DRONE DESIGN AND FABRICATION BASED UPON OPEN SOURCE SOFTWARE AND HARDWARE ARCHITECTURE AIMING TOWARDS INDIGENISATION

A THESIS

submitted by

EE19M012 YASH VARDHAN SINGH

for the award of the degree

of

MASTERS OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL

ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

JUNE 2021

THESIS CERTIFICATE

This is to certify that the thesis titled **DRONE DESIGN AND FABRICATION BASED UPON OPEN SOURCE SOFTWARE AND HARDWARE ARCHITECTURE AIMING TOWARDS INDIGENISATION**, submitted by **YASH VARDHAN SINGH**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Professor Ashok Jhunjhunwala
Research Guide
Dept. of Electrical Engineering
IIT Madras, 600 036

Place: Chennai
Date: **22 June 2021**

ACKNOWLEDGEMENTS

First of all, I would like to extend my heartfelt gratitude to Prof Ashok Jhunjhunwala, my guide and mentor, for providing me, along with two other project members, the opportunity to work on an implementational project with industry professionals at Centre for Battery Engineering and Electric Vehicles (CBEEV), IITM Research Park for the smart battery pack and Motorz, a startup at Telecom Centre of Excellence (TCOE), IITM Research Park for design and fabrication of motors for the drone. I would also like to take this opportunity to thank Prof Kaushal Jha at CBEEV and Prof Kannan Lakshminarayan at Motorz for extending all the necessary support and encouragement.

We are also indebted to Dr Satyanarayanan R. Chakravarthy, Professor at Aerospace department of IITM and e-Plane company, a drone and electric vehicle mobility based startup at IITM (co-founded by Prof), for providing me and my project members with much needed laboratory access for fabrication, ,material support, technical guidance and professional drone pilot support for testing our work in furtherance of the thesis.

I would also like to mention about the contribution of Mr Ibai Gorordo, PhD student at Kyushu Institute of Technology, Fukuoka, Japan for assisting me in slicing mm wave radar data and Mr Rishabh Singh, pursuing MSc (Mathematics) & B.E (Electronics and Instrumentation) at BITS Pilani for assisting me with sending proximity data from mmWave radar to Autopilot application running on FCU.

Last I would like to extend my gratitude to Tony Joseph and Akhil Sharma for being excellent team members and committed partners in implementing a project of this magnitude under pandemic conditions.

ABSTRACT

KEYWORDS: Flight Controller Unit (FCU), 4G LTE, Ground Control Station (GCS), Mission Planner, MAVlink, Virtual Private Network (VPN), YOLOv4 Artificial Intelligence (AI) Algorithm

Today most of the drone startups are actually system integrators utilizing the existing set of components to develop products which are quite similar due to common hardware and software architecture many of which are proprietary in nature. We at IITM under the guidance of our mentor and guide wanted to understand the implementational aspects of drone design with the aim of incorporating open source hardware and software as much as possible so as to eventually design and fabricate key components inhouse leading to indigenization. We began our project by studying the implementational aspects of existing drones available. After getting keen insight into the various aspects we began the process of customization. The first customization was to prepare Flight Controller Unit (FCU) on completely open source hardware and software. The flight controller being utilized for the drone is Beagle Bone Blue, a Linux based general purpose robotics platform with open source hardware design and open source operating system. All the major softwares including Ardupilot autopilot stack, Artificial Intelligence Algorithm YOLO , Linux Operating Systems (Beagle Bone Blue and companion computer) are open source softwares. We were able to integrate a 3D mm Wave radar as proximity sensor.

Thereafter, as part of inhouse design and fabrication, the propulsion system i,e motors with ESC combine have been designed. An inhouse designed and fabricated smart battery pack from CBEEV at IITMRP with Battery Management System (BMS) was tested on drone with live monitoring through a web-based dashboard. We as a project team have tried to discuss all these aspects in our thesis so that this project can be carried forward from hereon to build further and achieve complete open source hardware and software architecture with the aim towards indigenization of components i.e inhouse design and fabrication.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial Intelligence
AHRS	Attitude and Heading Reference Systems
BBB	Beagle Bone Blue
BITS	Birla Institute of Technology
BMS	Battery Management System
CBEEV	Centre for Battery Engineering and Electric Vehicles
CUDA	Compute Unified Device Architecture
EKF	Extended Kalman Filtering
FCU	Flight Controller Units
GCS	Ground Control Station
IITM	Indian Institute of Technology, Madras
IITMRP	Indian Institute of Technology Madras Research Park
IMU	Inertial Navigation Unit
ISP	Internet Service Provider
LTE	Long Term Evolution
MAVlink	Micro Air Vehicle Link
MP	Mission Planner
PID	Proportional–Integral–Derivative
Raspi	Raspberry Pi 4
SD	Secure Digital
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network
YOLO	You only look once
4G	Fourth Generation

NOTATION

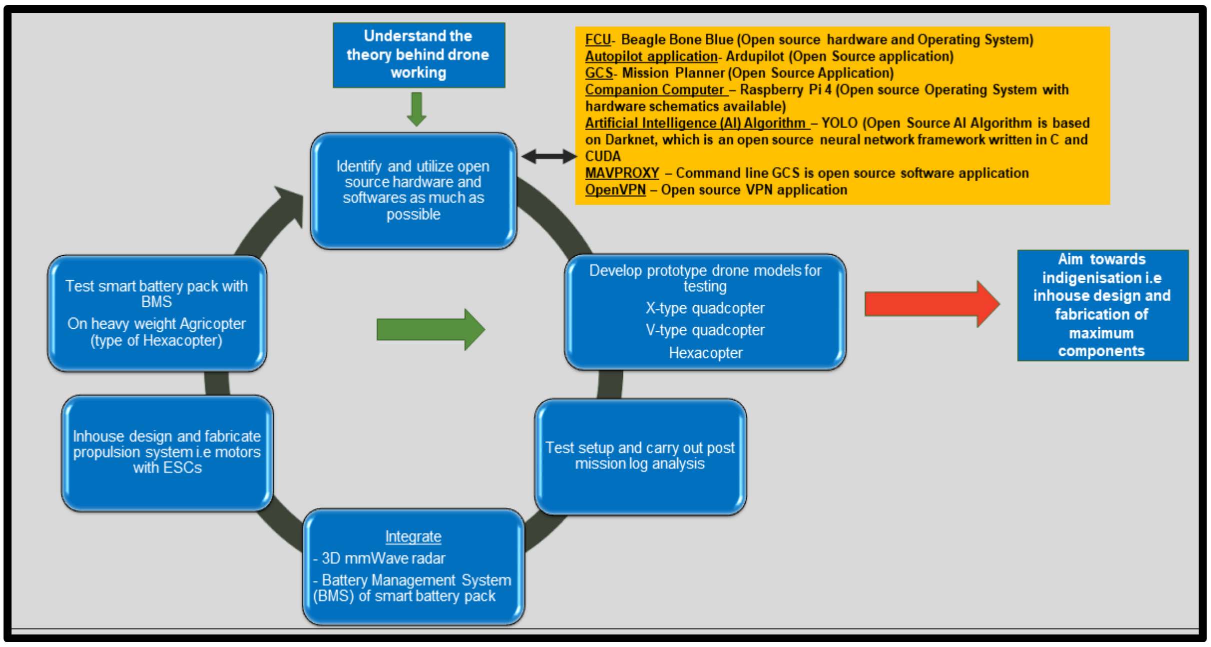
r	Radius, m
α	Angle of thesis in degrees
β	Flight path in degrees
ESC	Electronic Speed Control
D_p	Diameter of propellers
ρSSL	Density of air at sea level
V_{ROC}	Desired Rate Of Climb during Take Off
V_{ROD}	Desired Rate of Descend during landing
E_{TR}	Energy required (Total)
E_{VTO}	Energy required during vertical take-off
E_H	Energy required to hover
E_C	Energy required to cruise
E_{VLD}	Energy required for vertical landing
S_W	Surface area of wing
C_{D-TO}	Coefficient of drag during take-off
C_{D-LD}	Coefficient of drag during landing
η_p	Assumed Propeller efficiency during take off
η_m	Assumed Motor Efficiency during take off
η_{ESC}	Assumed Electronic Speed Controller (efficiency)
η_O	Overall Power plant efficiency
V_{ih}	The induced velocity during hover
V_{iVTO}	The induced velocity during vertical take-off
V_{iVTO}	The induced velocity during vertical landing
T_{VTO}	Thrust required for vertical take off
T_{LD}	Thrust required for landing
T_H	Thrust to hover
P_{VTO}	Power for vertical take-off
P_{VLD}	Power for vertical landing
A_{PROP}	Area of propeller
A_{PROPS}	Area of all propellers
η_0	Overall Power plant efficiency
η_P	Propeller efficiency
η_M	Motor efficiency
η_{ESC}	ESC efficiency

CHAPTER 1

INTRODUCTION

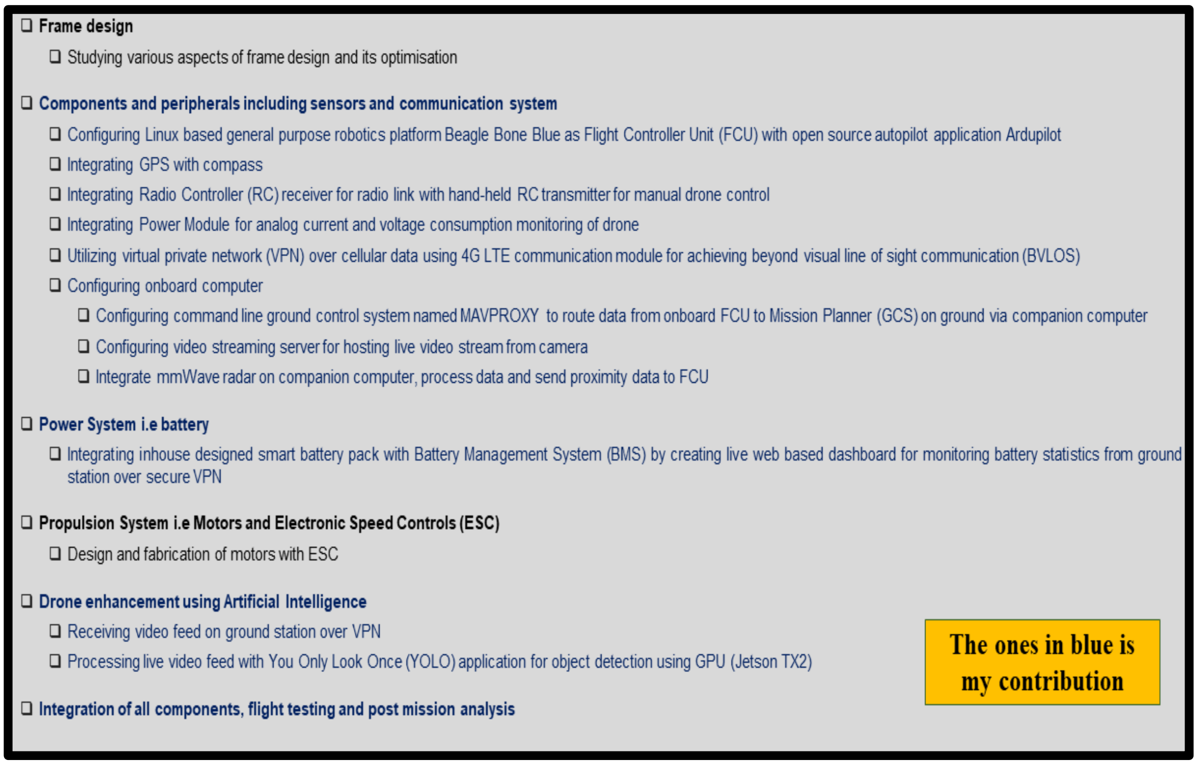
The research and development of unmanned aerial vehicles (UAV)s commonly known as drones all across the world has assumed gigantic proportions due to its ability to influence industries across varied spectrum from defense & homeland security to transportation of goods, search and rescue. The governments all across the world have identified its potential and are in the process of regulating and legalizing drone utilization across various sectors of the economy. As a result, there is a mad rush for development for drones with India seeing exponential rise of drone-based startups. As per Business Insider report, the industry is expected to grow to $63.6 billion by 2025. However, if we get a closer look at the development of drone industry, we realize that it is heavily skewed in favour of Chinese industries especially in terms of development of hardware components and associated software. Also, most drone startups are simply happy to integrate the existing hardware and software to produce drone solutions with little or no ingenuity towards developing inhouse solutions. This poses serious implications as it monopolizes industry in the favour of Chinese manufacturing and creates national security challenges. Against this backdrop, under the guidance of our guide and mentor we decided to undertake this project. Our aim was to first understand the nuances of designing a drone with the aim of incorporating as much as possible non-proprietary software and hardware components with the ultimate aim to carry out inhouse fabrication and design popularly known as indigenization of the process.

With that as the overall background, we began our study of drones. Considering the vast scope of the subject vis-à-vis the challenges of Covid pandemic, we limited ourselves to the study of multi-rotor copters namely Quadcopter (V and X-type) and Hexacopter covering key aspects such as Frame design, integrations of various components and peripherals including sensors and communication system, Power System i.e battery, Propulsion System i.e Motors and Electronic Speed Controls (ESC) and Drone enhancement using Artificial Intelligence. The overall objective of the project is listed in figure 1.1


Understand the theory behind drone working
Identify and utilize open source hardware and softwares as much as possible
FCU- Beagle Bone Blue (Open source hardware and Operating System)
Autopilot application- Ardupilot (Open Source application)
GCS- Mission Planner (Open Source Application)
Companion Computer – Raspberry Pi 4 (Open source Operating System with hardware schematics available)
Artificial Intelligence (AI) Algorithm – YOLO (Open Source AI Algorithm is based on Darknet, which is an open source neural network framework written in C and CUDA
MAVPROXY – Command line GCS is open source software application
OpenVPN – Open source VPN application
Develop prototype drone models for testing
X-type quadcopter
V-type quadcopter
Hexacopter
Aim towards indigenisation i.e inhouse design and fabrication of maximum components
Test setup and carry out post mission log analysis
Integrate
- 3D mmWave radar
- Battery Management System (BMS) of smart battery pack
Inhouse design and fabricate propulsion system i.e motors with ESCs
Test smart battery pack with BMS
On heavy weight Agricopter (type of Hexacopter)
Figure 1.1: The overall objective of project

Our project team comprised of Tony Joseph, Akhil Sharma and me, all being final year MTech students at IIT Madras. The aspect of frame design and post mission analysis was implemented by Tony Joseph while design and fabrication of Propulsion System including Motors and ESCs has been undertaken by Akhil Sharma. I have contributed to the project on the aspects of preparing the flight controller unit (FCU), logical integration of all peripherals and sensors including GPS, Radio controller, LIDAR, mm Wave radar, configuring the onboard companion computer (Raspberry Pi 4 AND Jetson Nano), communication over virtual private network (VPN) on 4G LTE module over cellular network, video streaming from onboard computer and processing live video feed using Artificial Intelligence Algorithm for object detection at ground station (using Jetson TX2). We have worked together as a team to integrate the various components and test the setup with multiple flight testing and post flight log analysis for optimizing design.

Centre for Battery Engineering and Electric Vehicles (CBEEV), IITM Research Park, was the nodal agency which assisted us with procurement of various components, technical support and provided us with inhouse designed and fabricated smart battery pack with intelligent Battery Management System (BMS) under guidance of Prof Kaushal Jha. For motor design and manufacturing, we were assisted by a startup named Motorz at Telecom Centre for Excellence (TCOE) under guidance of Prof Kannan Lakshminarayan. For frame design, fabrication, assembly and professional drone pilot support, was provided to us by a startup named e-plane company at IITM under guidance of Prof Satya Chakravarthy of Aerospace department. The scope of my thesis is listed in figure 1.2


- ❑ **Frame design**
  - ❑ Studying various aspects of frame design and its optimisation
- ❑ **Components and peripherals including sensors and communication system**
  - ❑ Configuring Linux based general purpose robotics platform Beagle Bone Blue as Flight Controller Unit (FCU) with open source autopilot application Ardupilot
  - ❑ Integrating GPS with compass
  - ❑ Integrating Radio Controller (RC) receiver for radio link with hand-held RC transmitter for manual drone control
  - ❑ Integrating Power Module for analog current and voltage consumption monitoring of drone
  - ❑ Utilizing virtual private network (VPN) over cellular data using 4G LTE communication module for achieving beyond visual line of sight communication (BVLOS)
  - ❑ Configuring onboard computer
    - ❑ Configuring command line ground control system named MAVPROXY to route data from onboard FCU to Mission Planner (GCS) on ground via companion computer
    - ❑ Configuring video streaming server for hosting live video stream from camera
    - ❑ Integrate mmWave radar on companion computer, process data and send proximity data to FCU
- ❑ **Power System i.e battery**
  - ❑ Integrating inhouse designed smart battery pack with Battery Management System (BMS) by creating live web based dashboard for monitoring battery statistics from ground station over secure VPN
- ❑ **Propulsion System i.e Motors and Electronic Speed Controls (ESC)**
  - ❑ Design and fabrication of motors with ESC
- ❑ **Drone enhancement using Artificial Intelligence**
  - ❑ Receiving video feed on ground station over VPN
  - ❑ Processing live video feed with You Only Look Once (YOLO) application for object detection using GPU (Jetson TX2)
- ❑ **Integration of all components, flight testing and post mission analysis**

The ones in blue is my contribution

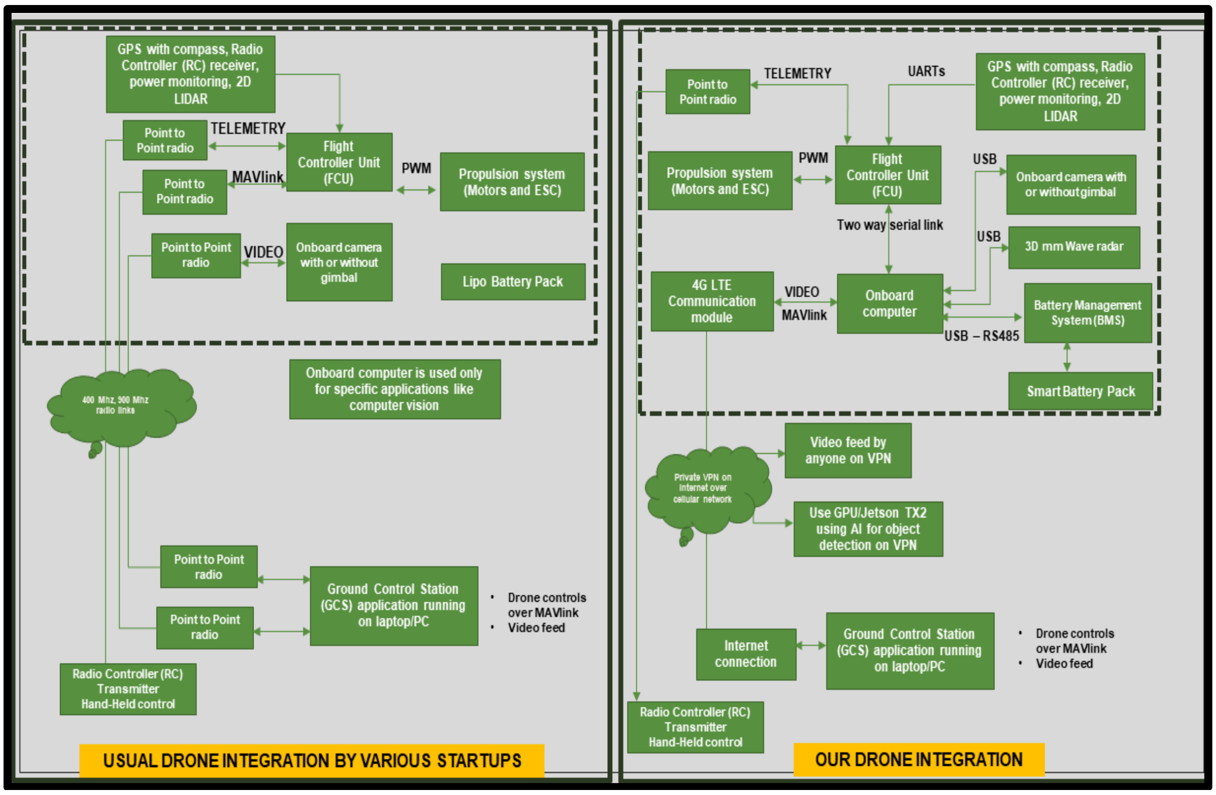
Figure 1.2: The scope of project for my thesis

Linux based Beagle Bone Blue general purpose robotics platform was configured as flight controller unit (FCU) running open source Ardupilot autopilot stack. Ardupilot autopilot application being open source, most widely used and best supported by a large community became the application of choice for FCU. This FCU has been tested for GPS, Compass, Radio Controller (RC) receiver and transmitter and LIDAR as sensor integration. The sensor hardware and firmware were procured commercially off the shelf. 4G LTE communication module was integrated as communication module utilizing VPN based internet connection as against the conventional point-to-point radios being used by most. Two different companion computers were configured and tested on drone. The companion computers were utilized for hosting onboard video streaming server, live battery dashboard and first-time integration of mm Wave radar as proximity sensor in India.

Under the head of Power System, a high capacity NMC battery with Battery Management System (BMS) designed at Centre for Battery Engineering and Electric Vehicles (CBEEV), IITM Research Park was integrated with the drone against the normal Lithium Polymer (LiPo) being widely used. We used LiPo batteries for X,V type quadcopter and

Hexacopter. The smart battery was tested on Agricopter (heavy weight Hexacopter provided by e-plane company). An intelligent dashboard has been created to monitor the battery statistics on a web-based dashboard available at ground station over VPN throughout the flight to provide greater situational awareness and intelligence to drone operator.

Under the Drone employability enhancement using Artificial Intelligence, an open source MJPEG server for live streaming video from onboard camera connected to companion computer was implemented. The live video was processed using open source You Only Look Once (YOLO) AI application utilized for image detection.



GPS with compass, Radio Controller (RC) receiver, power monitoring, 2D LIDAR
Point to Point radio
TELEMETRY
Flight Controller Unit (FCU)
PWM
Propulsion system (Motors and ESC)
Point to Point radio
MAVlink
Point to Point radio
VIDEO
Onboard camera with or without gimbal
Lipo Battery Pack
400 Mhz, 900 Mhz radio links
Onboard computer is used only for specific applications like computer vision
Point to Point radio
Point to Point radio
Ground Control Station (GCS) application running on laptop/PC
Drone controls over MAVlink
Video feed
Radio Controller (RC) Transmitter Hand-Held control
USUAL DRONE INTEGRATION BY VARIOUS STARTUPS
Point to Point radio
TELEMETRY
UARTs
GPS with compass, Radio Controller (RC) receiver, power monitoring, 2D LIDAR
Propulsion system (Motors and ESC)
PWM
Flight Controller Unit (FCU)
USB
Onboard camera with or without gimbal
Two way serial link
USB
3D mm Wave radar
4G LTE Communication module
VIDEO
MAVlink
Onboard computer
Battery Management System (BMS)
USB – RS485
Smart Battery Pack
Video feed by anyone on VPN
Private VPN on Internet over cellular network
Use GPU/Jetson TX2 using AI for object detection on VPN
Internet connection
Ground Control Station (GCS) application running on laptop/PC
Drone controls over MAVlink
Video feed
Radio Controller (RC) Transmitter Hand-Held control
OUR DRONE INTEGRATION

Figure 1.3: The difference in our implementation strategy

A total of three different types of drones were developed by our team during the course of the entire project to study the various aspects of the project. Two drones were of small drones' category with quadcopter configuration. One was X-Type and the other was V-Type which is a slight variation of the X-type quadcopter. The third drone is of the medium weight category i.e the Hexacopter with six motors. The smart battery pack which is much heavier in configuration was tested on Heavy Weight Hexacopter also popularly known as Agri copter taken from e-plane company, a startup at IIT Madras for testing purpose.

The X-Type Quadcopter was made to validate the configuration of a small drone with Beagle Bone Blue acting as both the Flight Controller Unit (FCU) and companion computer as it has Linux operating system on which Ardupilot autopilot application runs. It was tested with GPS, compass and Radio Controller (RC) receiver on board. The link between the drone and ground station called MAVlink was established over 4G LTE Communication module. Refer figure 1.4

The V Type Quadcopter was made to validate the configuration of a medium to heavy drone with onboard computer, camera and mmWave radar. It was taken as the miniature replica of the complete setup intended. Also the drone was integrated with the smart battery pack with Battery Management System (BMS) and intelligent Battery dashboard only on table top to validate configurations. However, due to the weight of the battery pack its field testing was possible only with an Agricopter. Refer figure 1.5

The Hexacopter was made to validate scaled up version of the drone with inhouse designed frame, Jetson Nano as companion computer and utilization of 360 degrees LIDAR as proximity sensor. This drone too was integrated with the smart battery pack with Battery Management System (BMS) and intelligent Battery dashboard on table top to validate configurations. However, again due to the weight of the battery pack its field testing was possible only with an Agricopter. Refer figure 1.6 and figure 1.7

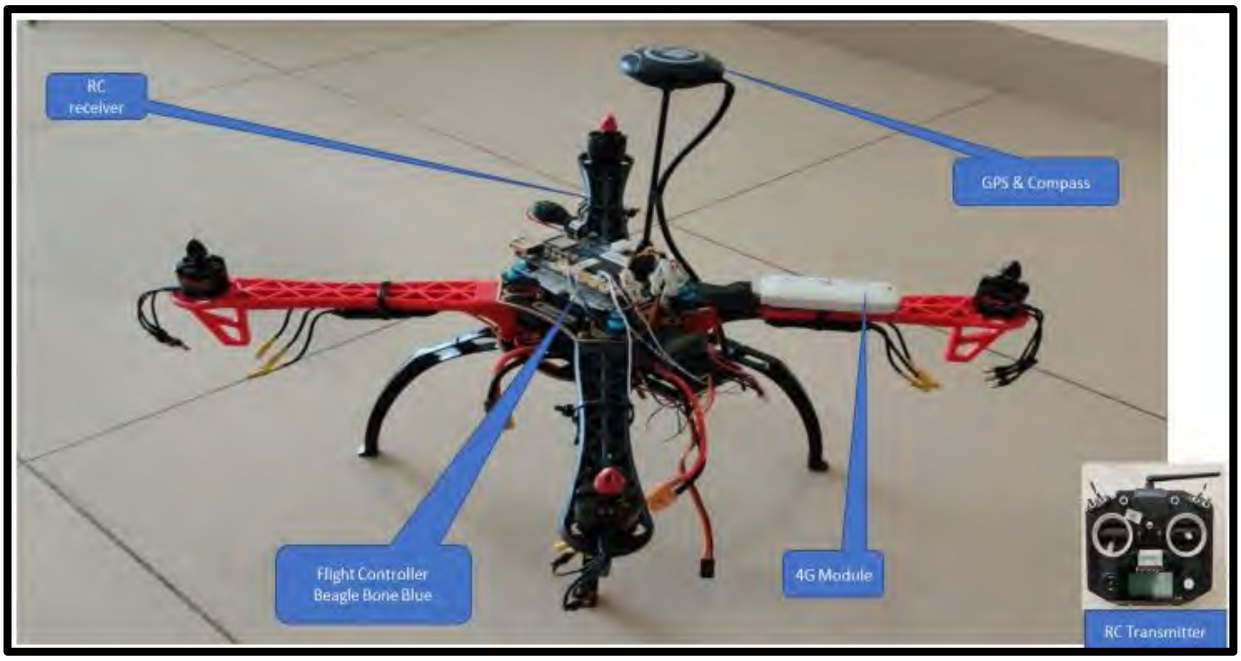


Figure 1.4: Light weight Quadcopter X-Type

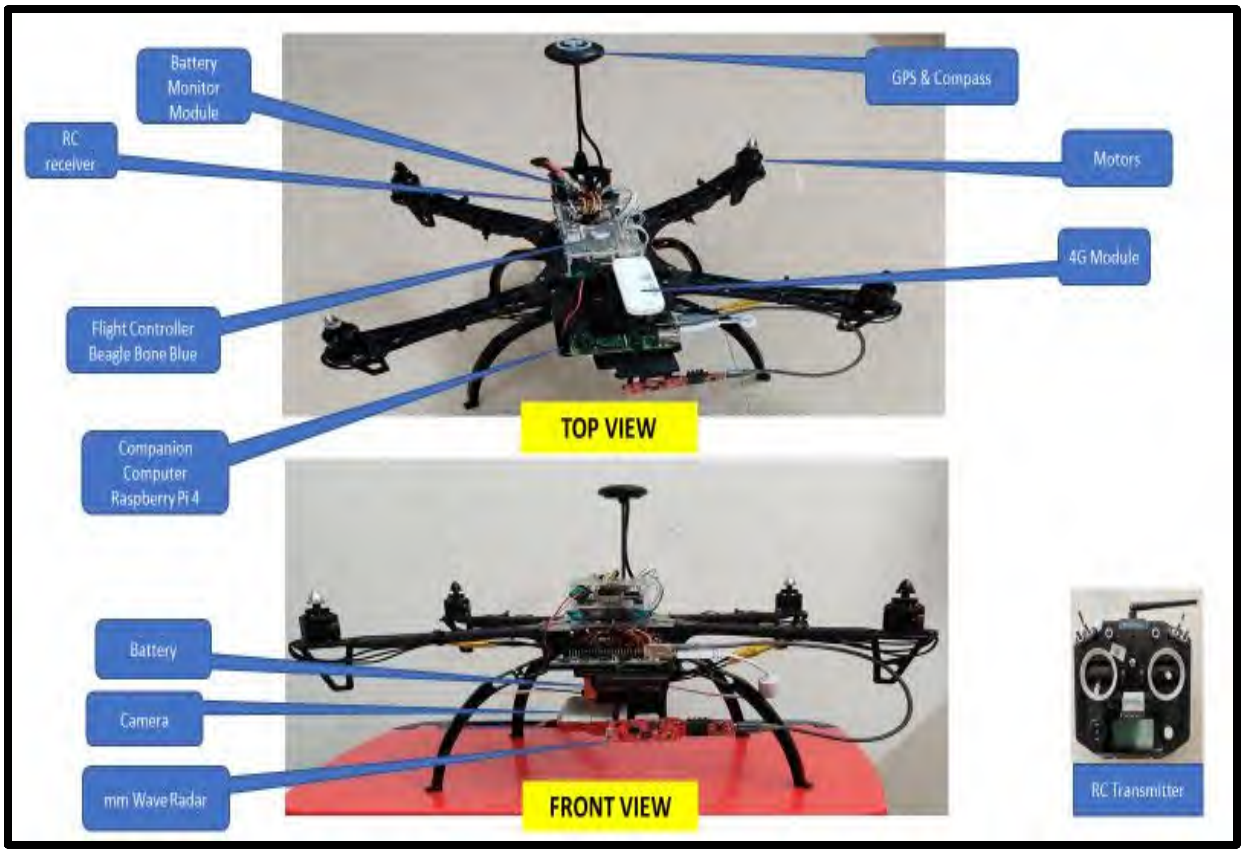


Figure 1.5: Light weight Quadcopter V-Type

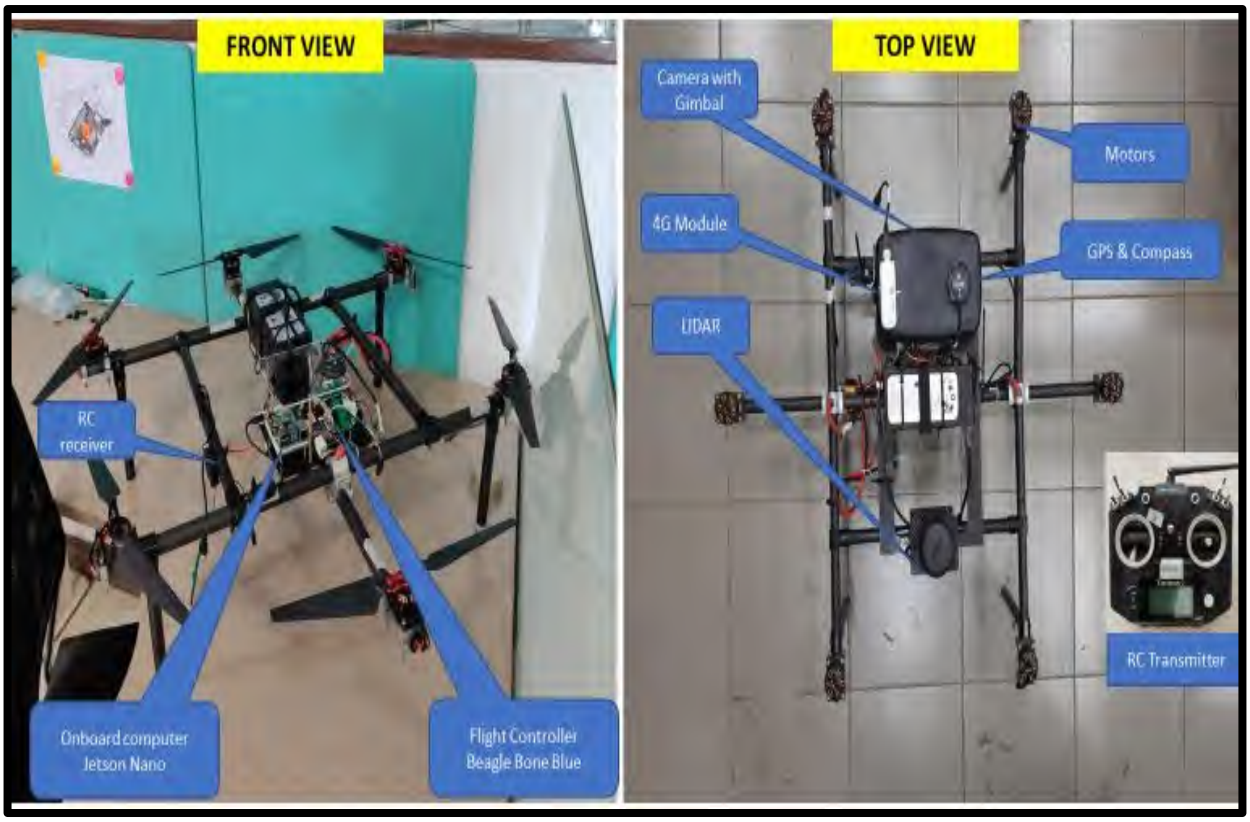


Figure 1.6: Medium weight Hexacopter

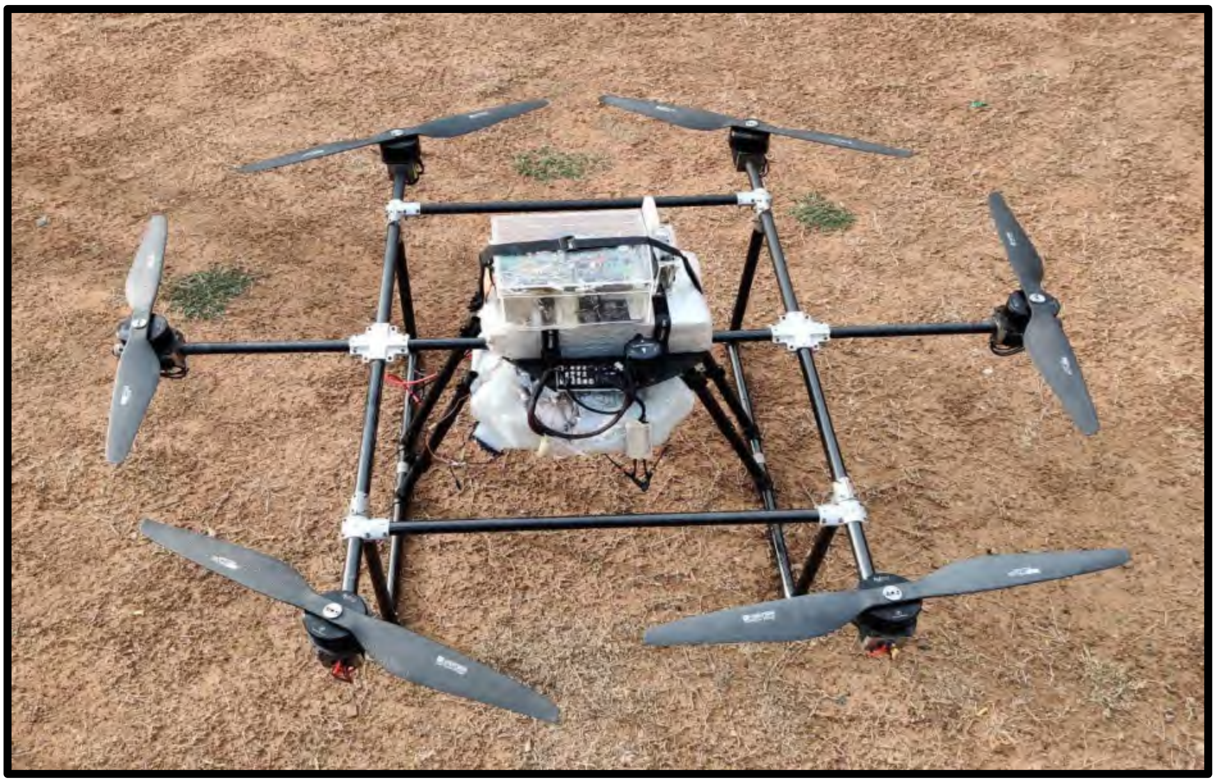


Figure 1.7: Heavy weight Agri Copter for smart battery pack testing

1.1 Understanding the physics of drones

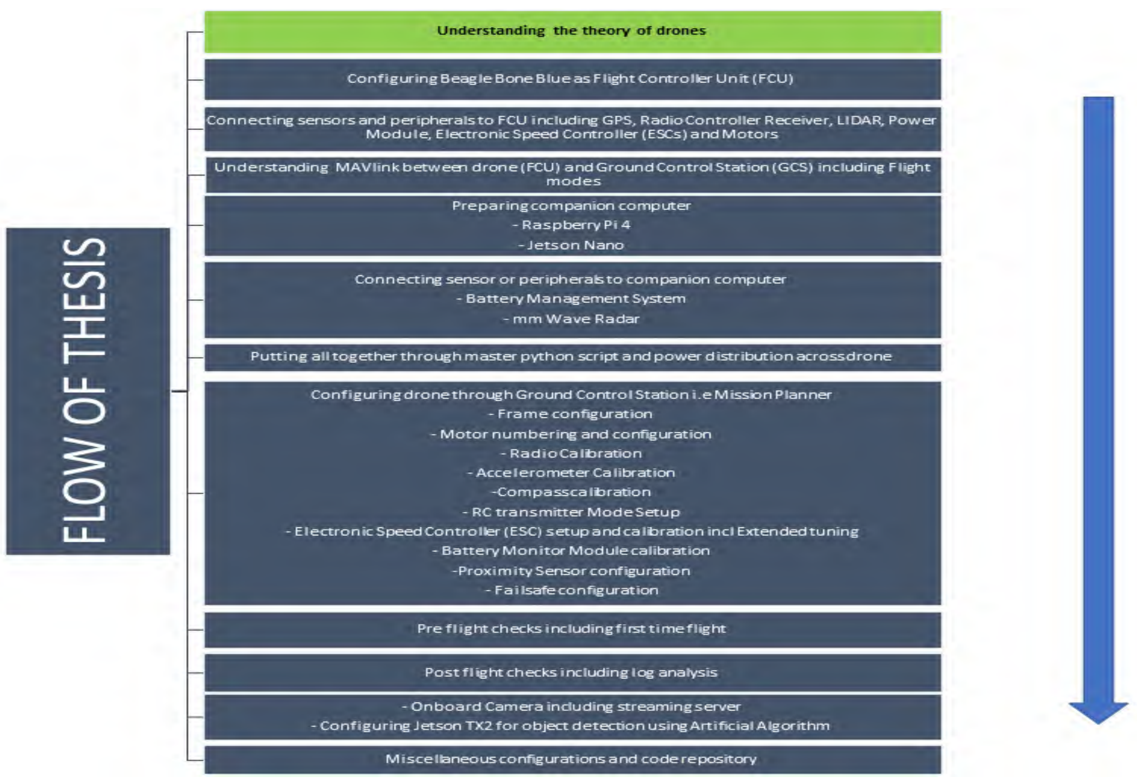

FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer - Raspberry Pi 4 - Jetson Nano
Connecting sensor or peripherals to companion computer - Battery Management System - mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner - Frame configuration - Motor numbering and configuration - Radio Calibration - Accelerometer Calibration -Compass calibration - RC transmitter Mode Setup - Electronic Speed Controller (ESC) setup and calibration incl Extended tuning - Battery Monitor Module calibration -Proximity Sensor configuration - Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server - Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

Figure 1.8: Flow of thesis until now

In order to be able to plan for drone integration and subsequent design and fabrication, the physics behind drone working needs to be well understood. The theory behind lift required to be generated by a Unmanned Aerial Vehicle (UAV) or commonly called drone is determined primarily by the weight of the drone. The weight of the drone is balanced by the lift produced by its motors. The normal force produced during the forward motion of the aircraft, at a given velocity, is called lift force. The required lift can be produced either by an increase of the forward velocity or by increase in the Angle Of Attack .

Angle of attack is the angle between the reference line of the aircraft (which runs along the longitudinal axis of the aircraft) and the relative wind. In the case of multi-copter drones, this angle is zero. The rotors must be able to create enough lift for the drone to take of at the required velocity of climb termed as Voc.

The Drag is the opposing force to the motion of drone produced by virtue of the viscosity of air. This drag force is overcome by the thrust force produced by the propeller. The total drag force has two components in it viz., - the zero lift drag and the induced drag. The zero lift drag is the force that is produced because of the friction caused by the air molecules while it flows over the aircraft surface and hence it is directly related to the smoothness of the surface and the area exposed to the airflow itself. The induced drag is caused by the downwash, which is influenced by aspect ratio of the rotors. Down wash is simply the downward velocity component induced beneath the rotor surface and hence altering the relative velocity itself. The zero lift drag coefficient (C_D).

The required thrust for drone to take off must overcome the force due to gravity and the drag force. This thrust required for drone to vertically take off = Weight of drone + Drag Force experienced during take-off

$$\mathbf{T_{VTO} = (M*g) + (0.5*\rho_{SSL}*V_{ROC}^2*S_W*C_{D\text{-}TO})}$$

The required thrust for drone to land must overcome the opposing force due to gravity and now this phase of the mission is aided by drag which in mathematical terms is defines as below: Thrust required for drone to vertically land = Weight of drone - Drag Force experienced during take-off

$$\mathbf{T_{VLD} = (M*g) - (0.5*\rho SSL*V_{ROD}^2*S_W*C_{D\text{-}LD})}$$

We know that thrust required during take-off is much more than thrust required for landing, hence we will calculate thrust per motor from thrust required for vertical take off which is given as,

Thrust per motor = (Thrust during take-off ($T_{VTO)}$ / No of motors on the drone)

This value must always be <= 0.5 times the maximum thrust of the motor to be utilized as per its data sheet. Also, C_{D-TO} and $C_{D\text{-}LD}$ are the coefficient of drag. I have taken value of 2 based on the results for drag of a flat plate, facing the flow at 90°. The thrust for hover is thrust required to maintain weight

$$\mathbf{T_H = (M*g)}$$

In terms of velocity we have two concepts, the induced velocity of hover denoted as V_{ih} and induced velocity for vertical take off $V_{iVTO.}$ This induced velocity is generally in a downward direction also called downwash. Its important consequences is that it modifies the flow of air around the frame and impacts its aerodynamic characteristics. Induced velocity of hover is given by:

$$\mathbf{V_{ih} = \sqrt{} T_H/(2 * \rho_{SSL} * A_{PROPS})}$$

Where area of Propellers (A_{PROP}) = $2*\pi*(D_P/2)^2$ and total area under all propellars (A_{PROPS}) = A_{PROP} * No of propellers

Similariliy, the induced velocity for vertical take off $\mathbf{(V_{iVTO}) = -(V_{ROC}/2) + \sqrt{}((V_{ROC}/2)^2 + V_{ih})}$ and induced velocity of landing $\mathbf{(V_{iLD}) = -(V_{ROD}/2) - \sqrt{}((V_{ROD}/2)^2 - V_{ih})}$

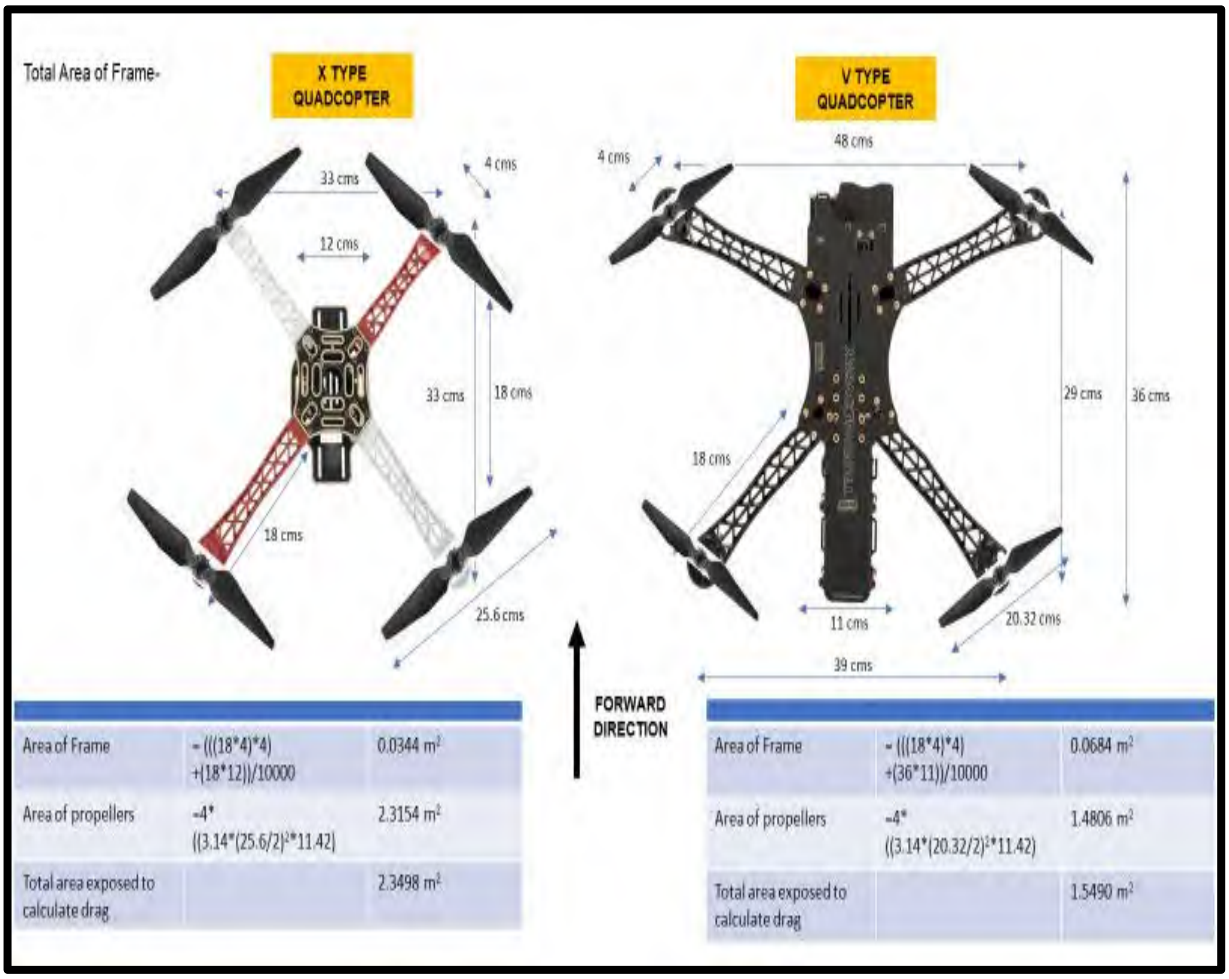
Now power required during vertical take-off denoted by $\mathbf{P_{VTO} = (T_{VTO} * V_i) / \eta_0}$ where η_0 is the overall efficicncy of the power plant. It is the product of efficiency of propellars, motors and ESCs ($\eta_0 = \eta_P * \eta_M * \eta_{ESC)}$. The typical values are 0.66*0.85*0.98 = 0.55 for η_0 Similarly the power during vertical landing denoted by $P_{VLD} = (T_{VLD} * V_i) / \eta_{0.}$ The power for hover denoted P_H and power to cruise denoted by P_C is related to take-off and landing by a simple intuitive relation.

$$\mathbf{P_{VTO} > P_C > P_H > P_{VLD}}$$

Thus, if we know what altitude we wish to achieve during vertical take-off based on rate of climb V_{ROC}, we can calculate energy E_{VTO} given by power * time. Similarly time for hover can give energy required for hover denoted by E_H while time for cruise can give energy denoted by E_C which is required to cruise from one point to another . The rate of descent V_{ROD} and the altitide from where descent begins can give total time for descent which can help us calculate the energy required for descent denoted by E_{VLD}. Thus, we see

Total energy required denoted by $\mathbf{E_{TR} = E_{VTO} + E_H + E_C + E_{VLD}}$ where $E_{TR} <= 0.85$ * Battery Energy Capacity

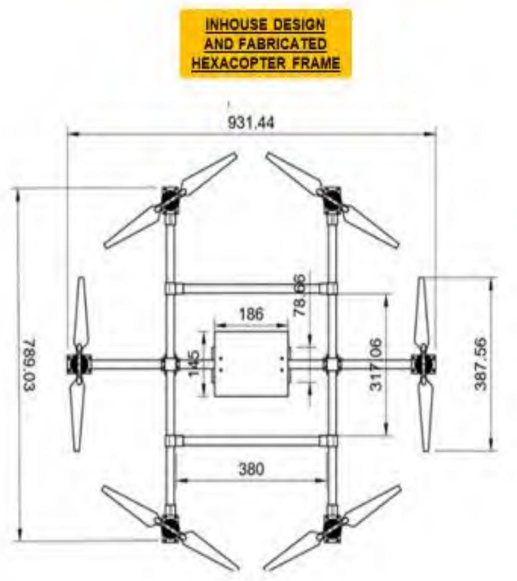
I have done sample calculations for the X-Type Quadcopter, V-Type Quadcopter and Hexacopter and given in figure 1.11 and figure 1.12. We can see that the thrust requirement per motor in each of the frame is <50% maximum trust capacity of motots selected.

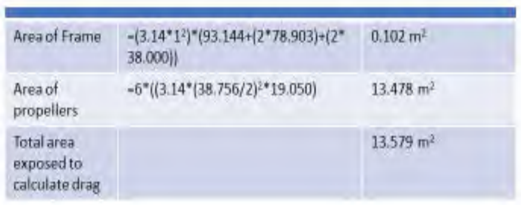


Area of Frame	= (((18*4)*4) +(18*12))/10000	0.0344 m²
Area of propellers	=4* ((3.14*(25.6/2)²*11.42)	2.3154 m²
Total area exposed to calculate drag		2.3498 m²

Area of Frame	= (((18*4)*4) +(36*11))/10000	0.0684 m²
Area of propellers	=4* ((3.14*(20.32/2)²*11.42)	1.4806 m²
Total area exposed to calculate drag		1.5490 m²

Figure 1.9: Dimensions of X-Type and V-Type Quadcopters



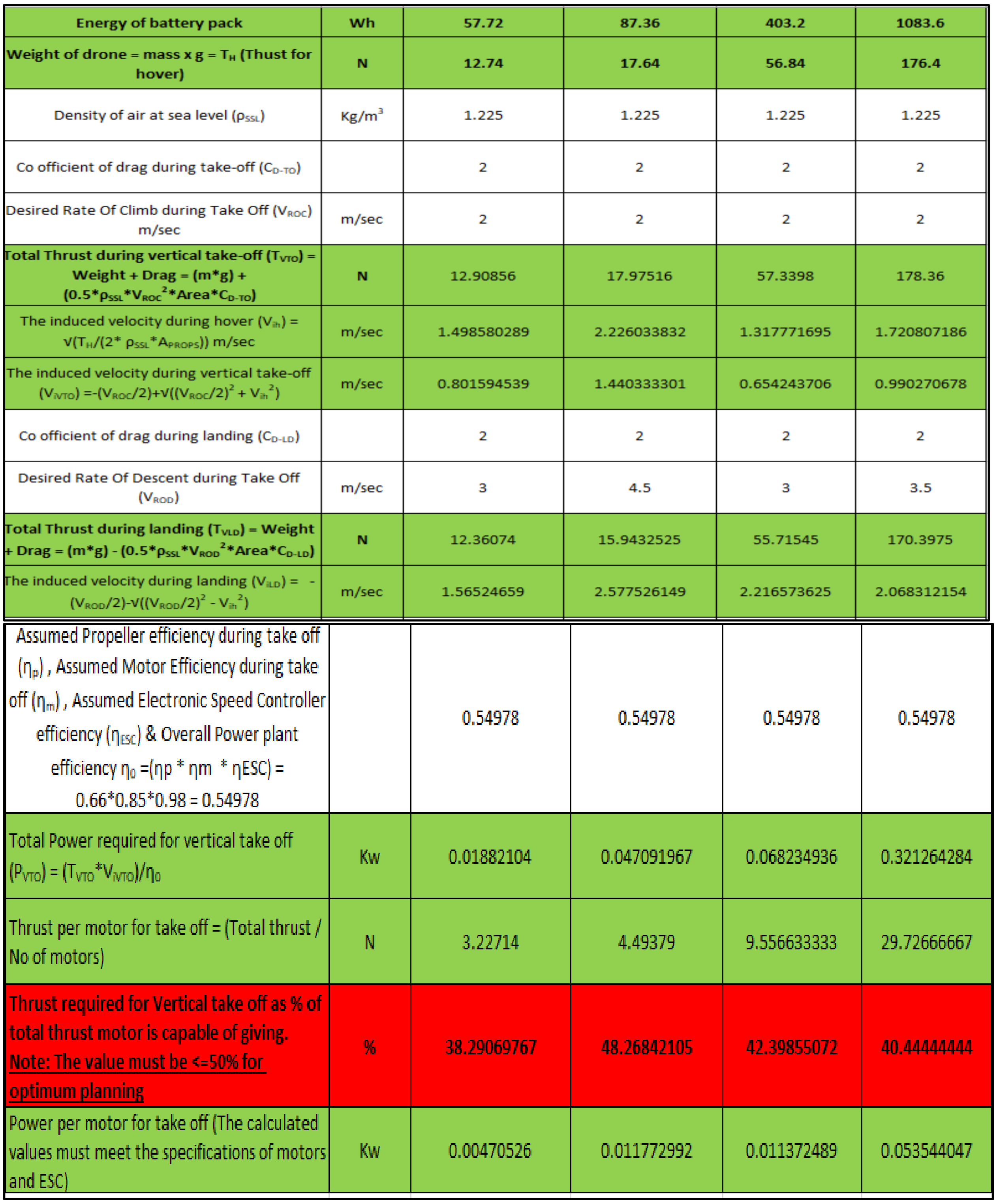


Area of Frame	=(3.14*1²)*(93.144+(2*78.903)+(2*38.000))	0.102 m²
Area of propellers	=6*((3.14*(38.756/2)²*19.050)	13.478 m²
Total area exposed to calculate drag		13.579 m²

Figure 1.10: Dimensions of Hexacopter

Items	Units	X type Quadcopter	V type Quadcopter	Hexacopter	Agricopter
Frame material		Carbon Fiber + Polyamide Nylon	Carbon Fiber + Polyamide Nylon	Carbon fiber	Carbon Fiber
Total Area	m^2	0.0344	0.0684	0.102	0.4
Gross mass	gms	1300	1800	5800	18000
Legs clearance	mm	150	150	125	560
Motor Model		Emax MT 2215	Emax MT2216	T Motor Anti Gravity 4005/6	T Motors P60
Motor Kv	Kv	935	810	380	340
Max Thrust as per data sheet (gms)	gms	860	950	2300	7500
Mx Thrust per motor in (N)	N	8.428	9.31	22.54	73.5
No of motors	No	4	4	6	6
Total thrust from motors (No of motors x Individual motor thrust) calculated as per datasheet	**N**	**33.712**	**37.24**	**135.24**	**441**
ESC		Emax BL Heli	Emax BL Heli	Favorite LittleBee	Favorite LittleBee
ESC rated current	Amps (A)	30	30	30	30
Prop Size	inch x inch	10x4.5	80x4.5	15x7.5	22x6.6
Prop Area (A_{PROP}) = π*(D/2)²*Pitch	**m^2**	**0.578873036**	**0.370154615**	**2.246252064**	**4.097469285**
Total prop area (A_{PROPS})= Prop area * No of props	**m^2**	**2.315492143**	**1.480618458**	**13.47751239**	**24.58481571**
Total area including frame and propellers	**m^2**	**2.349892143**	**1.549018458**	**13.57951239**	**24.98481571**
Battery Specifications		3s	4s	6s	6S
Battery Voltage (Max) Volts	Volts (V)	11.1	16.8	25.2	25.2
Current Capacity	mAh	5200	5200	16000	43000

Figure 1.11: Calculations of various parameters for each frame



Energy of battery pack	**Wh**	**57.72**	**87.36**	**403.2**	**1083.6**
Weight of drone = mass x g = T_H (Thust for hover)	**N**	**12.74**	**17.64**	**56.84**	**176.4**
Density of air at sea level (ρ_{SSL})	Kg/m^3	1.225	1.225	1.225	1.225
Co officient of drag during take-off (C_{D-TO})		2	2	2	2
Desired Rate Of Climb during Take Off (V_{ROC}) m/sec	m/sec	2	2	2	2
Total Thrust during vertical take-off (T_{VTO}) = Weight + Drag = (m*g) + (0.5*ρ_{SSL}*V_{ROC}^2*Area*C_{D-TO})	**N**	12.90856	17.97516	57.3398	178.36
The induced velocity during hover (V_{ih}) = $\sqrt{(T_H/(2^* \rho_{SSL} {}^*A_{PROPS}))}$ m/sec	m/sec	1.498580289	2.226033832	1.317771695	1.720807186
The induced velocity during vertical take-off (V_{iVTO}) $=-(V_{ROC}/2)+\sqrt{((V_{ROC}/2)^2 + V_{ih}^2)}$	m/sec	0.801594539	1.440333301	0.654243706	0.990270678
Co officient of drag during landing (C_{D-LD})		2	2	2	2
Desired Rate Of Descent during Take Off (V_{ROD})	m/sec	3	4.5	3	3.5
Total Thrust during landing (T_{VLD}) = Weight + Drag = (m*g) - (0.5*ρ_{SSL}*V_{ROD}^2*Area*C_{D-LD})	**N**	12.36074	15.9432525	55.71545	170.3975
The induced velocity during landing (V_{iLD}) = $-(V_{ROD}/2)-\sqrt{((V_{ROD}/2)^2 - V_{ih}^2)}$	m/sec	1.56524659	2.577526149	2.216573625	2.068312154
Assumed Propeller efficiency during take off (η_p) , Assumed Motor Efficiency during take off (η_m) , Assumed Electronic Speed Controller efficiency (η_{ESC}) & Overall Power plant efficiency η_0 =(ηp * ηm * ηESC) = 0.66*0.85*0.98 = 0.54978		0.54978	0.54978	0.54978	0.54978
Total Power required for vertical take off (P_{VTO}) = (T_{VTO}*V_{iVTO})/η_0	Kw	0.01882104	0.047091967	0.068234936	0.321264284
Thrust per motor for take off = (Total thrust / No of motors)	N	3.22714	4.49379	9.556633333	29.72666667
Thrust required for Vertical take off as % of total thrust motor is capable of giving. Note: The value must be <=50% for optimum planning	%	38.29069767	48.26842105	42.39855072	40.44444444
Power per motor for take off (The calculated values must meet the specifications of motors and ESC)	Kw	0.00470526	0.011772992	0.011372489	0.053544047

Figure 1.12: Calculations of various parameters for each frame

CHAPTER 2

CONFIGURING BEAGLE BONE BLUE AS FLIGHT CONTROLLER UNIT

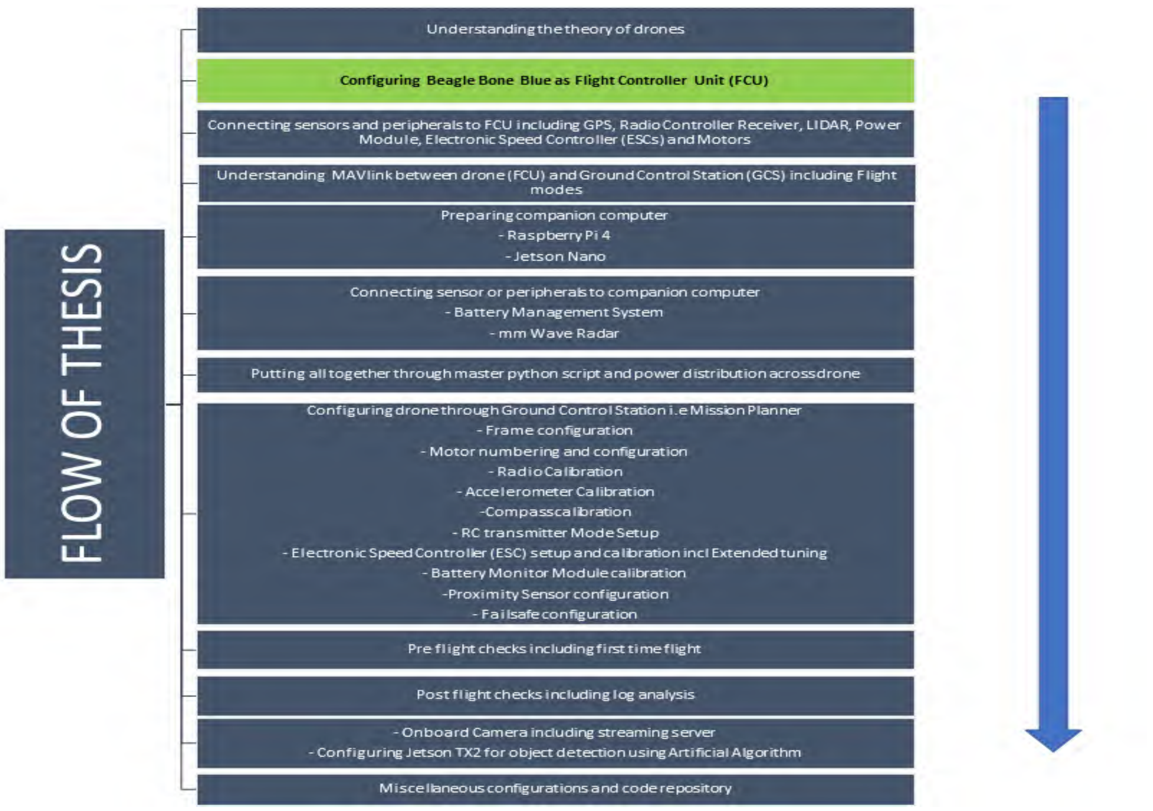
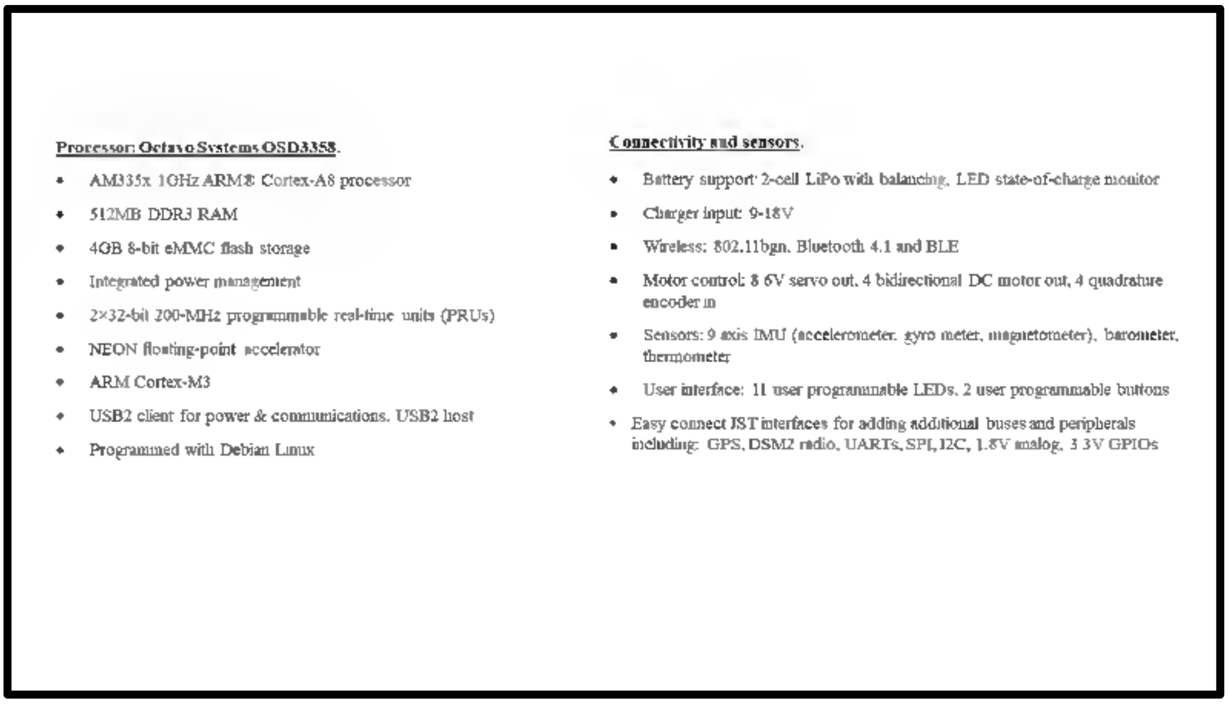

FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server
- Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

Figure 2.1: Flow of thesis until now

The most important component of any drone is the Flight Controller Unit (FCU). I have dedicated an entire chapter to explaining the configuration of Beagle Bone Blue as Flight Controller. All sensors and drone electrical components revolve mainly around FCU. Beagle Bone Blue (BBB) is an all-in-one Linux-based general purpose robotics computer with small form factor (3.5" x 2.15" board) running Octavo OSD3358 microprocessor. It also supports wifi/Bluetooth for networking, inbuilt IMU/barometer, power management and regulation. It has the option of being powered from 9-18 volts DC power source via barrel jack. It requires JST type connectors for interfacing with sensors It has all the commonly-needed buses for additional peripherals in embedded applications. The key aspect for the choice of this board is its fully open source and actively supported by a strong community, the real-time performance, flexible networking, and rich set of robotics-oriented capabilities making it versatile, fast, streamlined, affordable, and fun.



Processor: Octavo Systems OSD3358.

- AM335x 1GHz ARM® Cortex-A8 processor
- 512MB DDR3 RAM
- 4GB 8-bit eMMC flash storage
- Integrated power management
- 2×32-bit 200-MHz programmable real-time units (PRUs)
- NEON floating-point accelerator
- ARM Cortex-M3
- USB2 client for power & communications. USB2 host
- Programmed with Debian Linux

Connectivity and sensors.

- Battery support 2-cell LiPo with balancing, LED state-of-charge monitor
- Charger input: 9-18V
- Wireless: 802.11bgn. Bluetooth 4.1 and BLE
- Motor control: 8 6V servo out. 4 bidirectional DC motor out, 4 quadrature encoder in
- Sensors: 9 axis IMU (accelerometer, gyro meter, magnetometer), barometer, thermometer
- User interface: 11 user programmable LEDs. 2 user programmable buttons
- Easy connect JST interfaces for adding additional buses and peripherals including: GPS, DSM2 radio, UARTs, SPI, I2C, 1.8V analog, 3.3V GPIOs

Figure 2.2: Technical specifications of Beagle Bone Blue (BBB) list

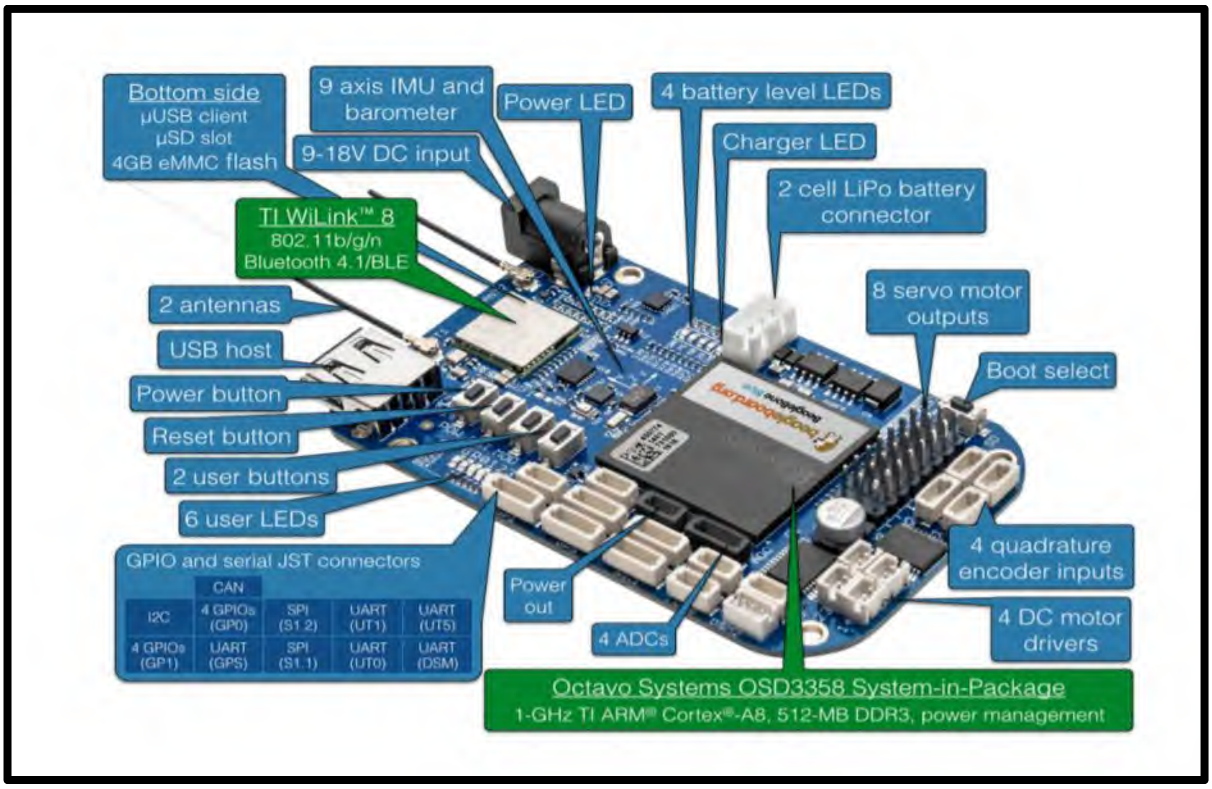


Figure 2.3: Technical specifications of Beagle Bone Blue (BBB) pictorial

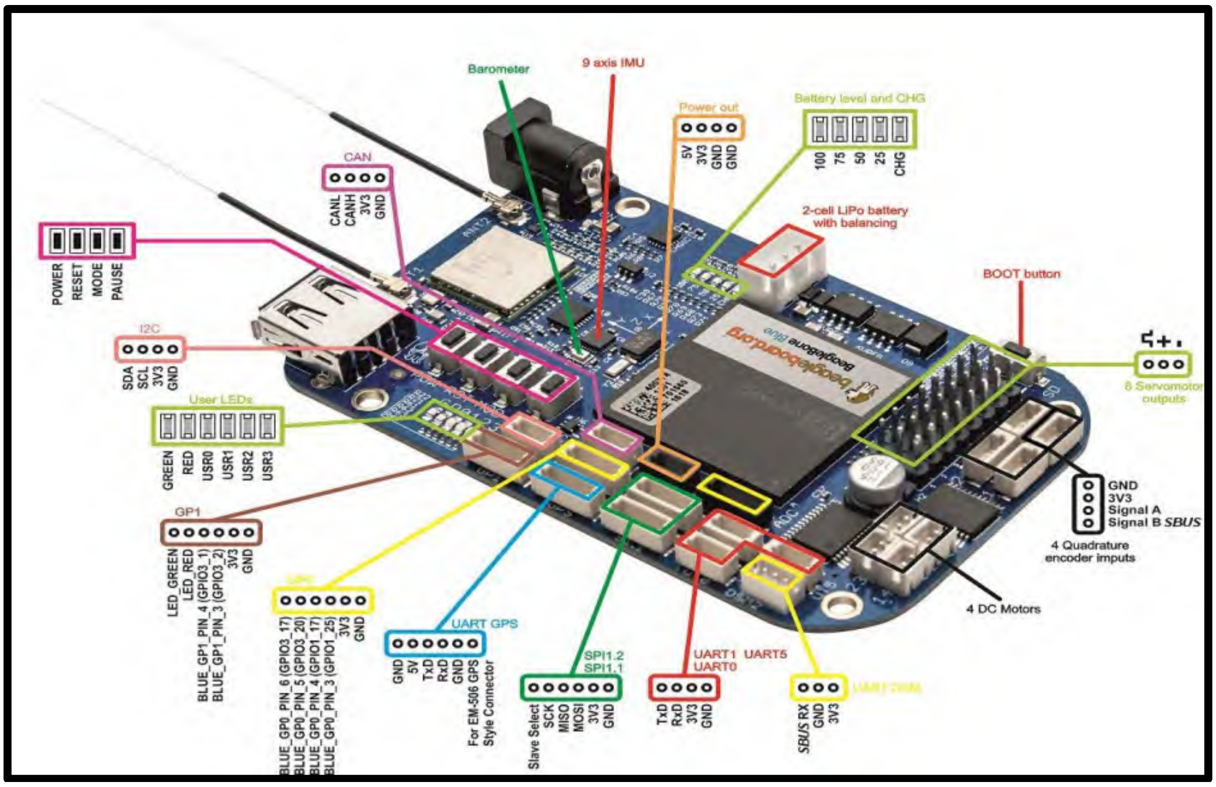
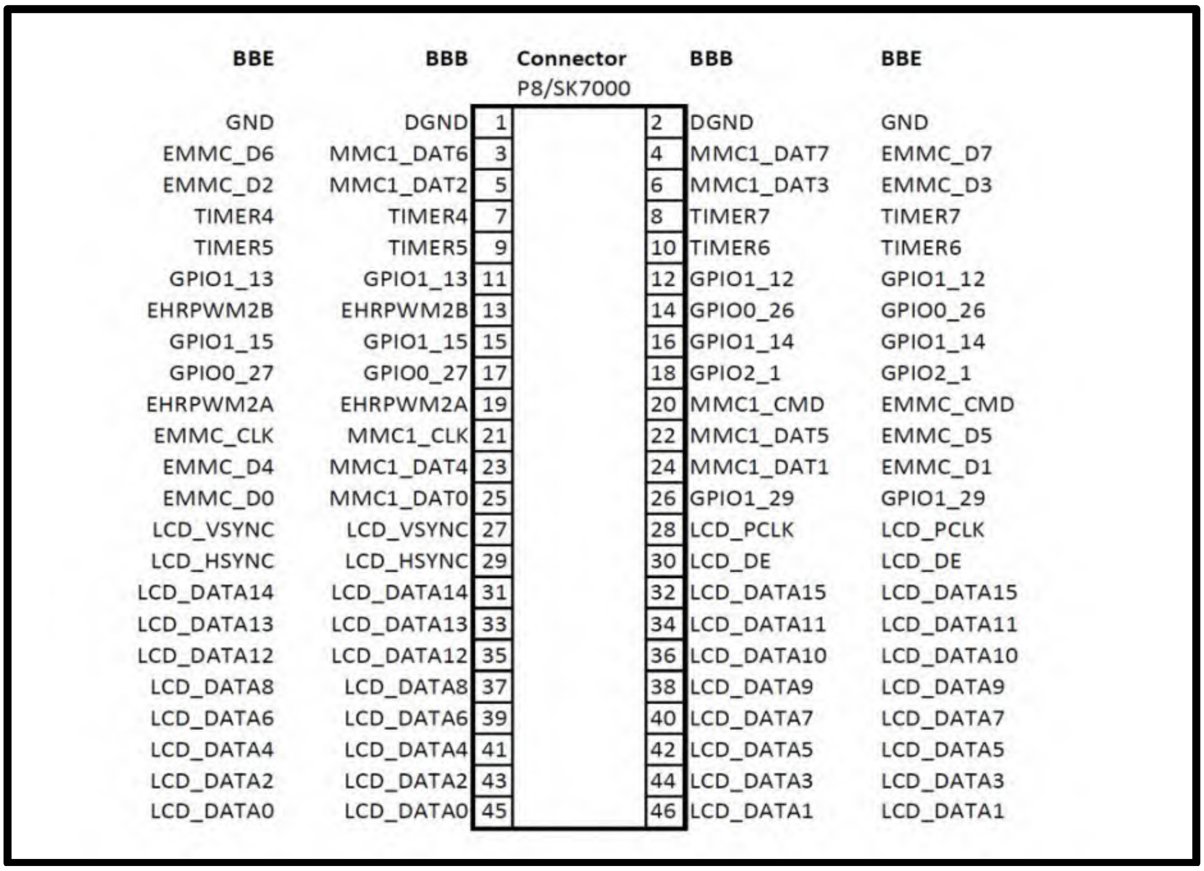


Figure 2.4: Port wise pinout details of Beagle Bone Blue (BBB)



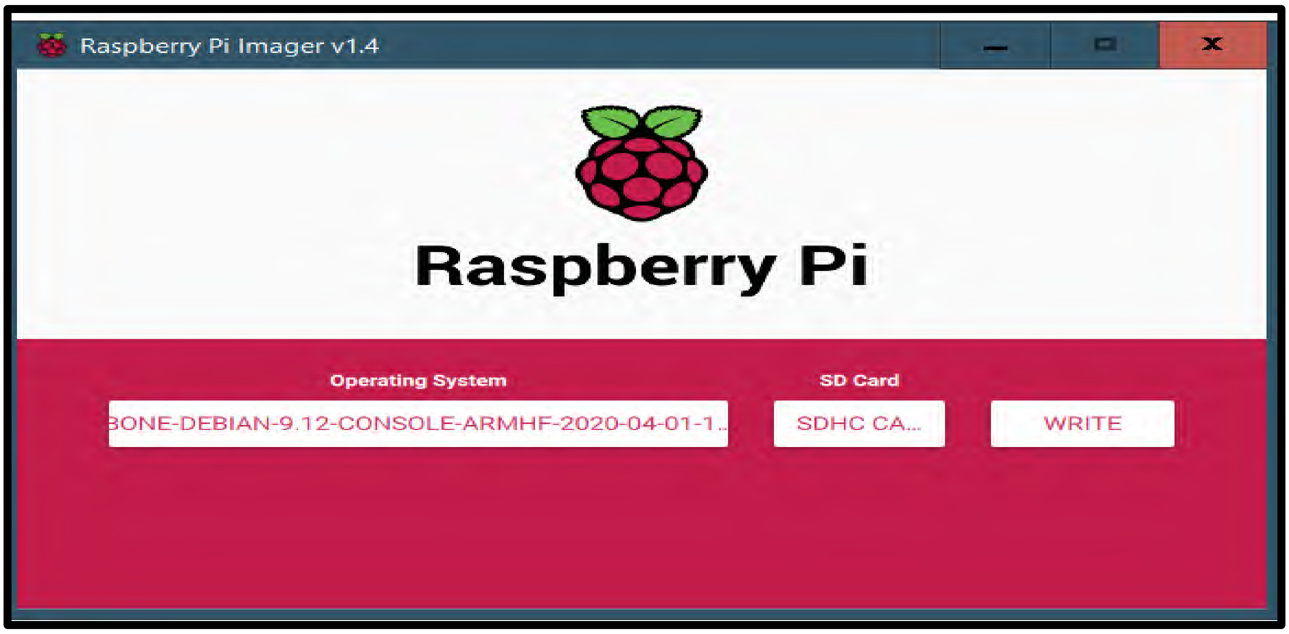
BBE	BBB	Connector P8/SK7000		BBB	BBE
GND	DGND	1	2	DGND	GND
EMMC_D6	MMC1_DAT6	3	4	MMC1_DAT7	EMMC_D7
EMMC_D2	MMC1_DAT2	5	6	MMC1_DAT3	EMMC_D3
TIMER4	TIMER4	7	8	TIMER7	TIMER7
TIMER5	TIMER5	9	10	TIMER6	TIMER6
GPIO1_13	GPIO1_13	11	12	GPIO1_12	GPIO1_12
EHRPWM2B	EHRPWM2B	13	14	GPIO0_26	GPIO0_26
GPIO1_15	GPIO1_15	15	16	GPIO1_14	GPIO1_14
GPIO0_27	GPIO0_27	17	18	GPIO2_1	GPIO2_1
EHRPWM2A	EHRPWM2A	19	20	MMC1_CMD	EMMC_CMD
EMMC_CLK	MMC1_CLK	21	22	MMC1_DAT5	EMMC_D5
EMMC_D4	MMC1_DAT4	23	24	MMC1_DAT1	EMMC_D1
EMMC_D0	MMC1_DAT0	25	26	GPIO1_29	GPIO1_29
LCD_VSYNC	LCD_VSYNC	27	28	LCD_PCLK	LCD_PCLK
LCD_HSYNC	LCD_HSYNC	29	30	LCD_DE	LCD_DE
LCD_DATA14	LCD_DATA14	31	32	LCD_DATA15	LCD_DATA15
LCD_DATA13	LCD_DATA13	33	34	LCD_DATA11	LCD_DATA11
LCD_DATA12	LCD_DATA12	35	36	LCD_DATA10	LCD_DATA10
LCD_DATA8	LCD_DATA8	37	38	LCD_DATA9	LCD_DATA9
LCD_DATA6	LCD_DATA6	39	40	LCD_DATA7	LCD_DATA7
LCD_DATA4	LCD_DATA4	41	42	LCD_DATA5	LCD_DATA5
LCD_DATA2	LCD_DATA2	43	44	LCD_DATA3	LCD_DATA3
LCD_DATA0	LCD_DATA0	45	46	LCD_DATA1	LCD_DATA1

Figure 2.5: Pinout details of Beagle Bone Blue (BBB)

This Beagle Bone Blue will be configured as Flight Controller Unit (FCU). It will run Debian Linux Operating System (OS) and Ardupilot autopilot stack as application. This process has been explained in subsequent sub chapters and paras.

2.1 Preparing Beagle Bone Board with OS (Debian Linux)

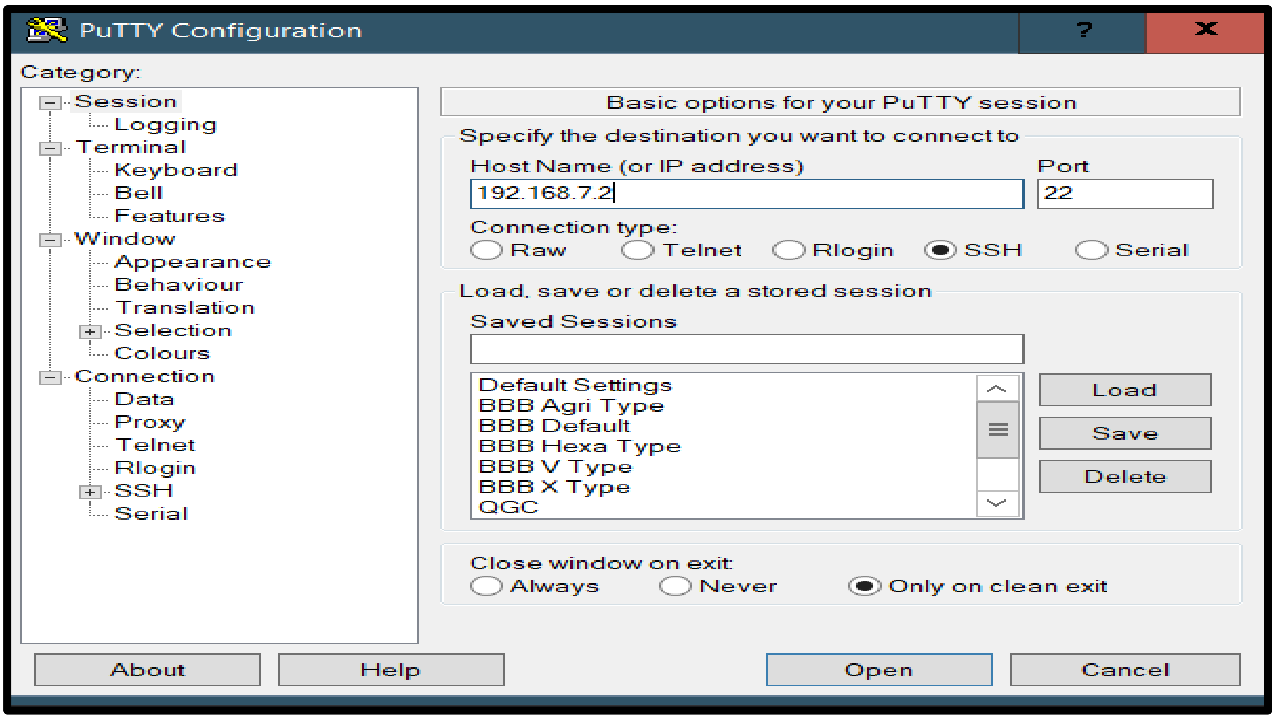
Preparing SD Card We need to format an SD Card preferably 16/32 GB and flash Linux Debian Operating System. Download ***bone-debian-9.12-console-armhf-2020-04-01-1gb.img*** and flash it on SD card of 16/32 Gb using Balena Etcher or any other such application. I have used Raspberry Pi imager to do the same.



Operating System: BONE-DEBIAN-9.12-CONSOLE-ARMHF-2020-04-01-1...
SD Card: SDHC CA...

Figure 2.6: Flashing SD card for Beagle Bone Blue through Raspberry Pi Imager

Insert SD card and boot Beagle Bone Blue Connect Beagle Bone Blue with micro-USB cable to USB port of laptop. Use putty application for ssh access of Beagle Bone Blue. Default ip access to access Beagle Bone Blue is 192.168.7.2



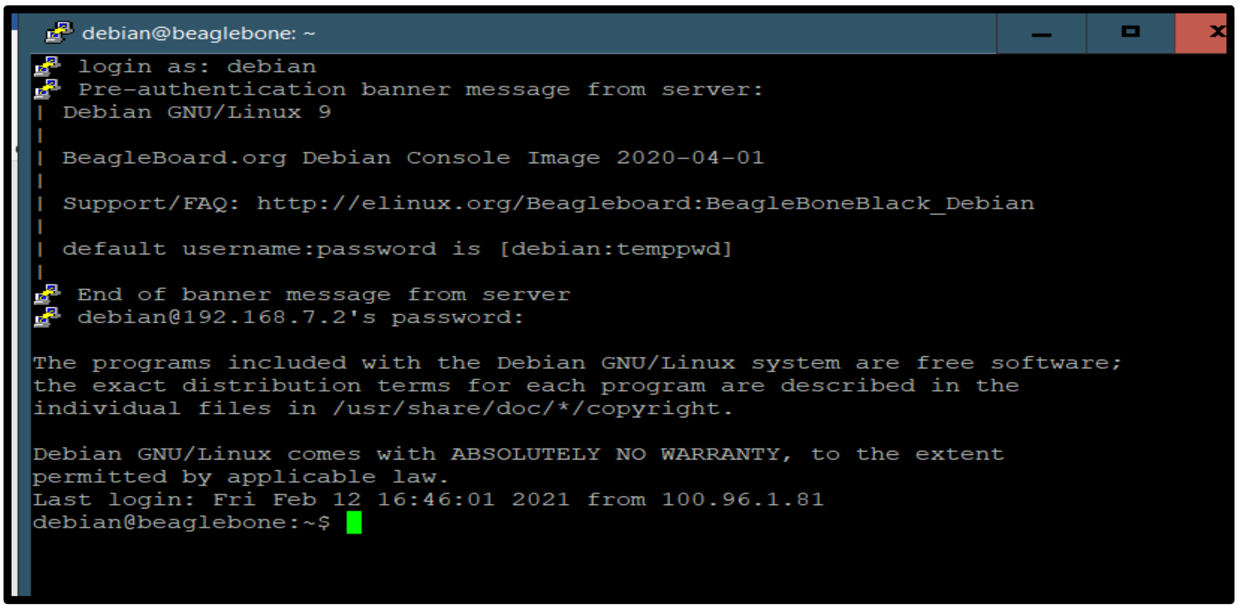
Host Name (or IP address): 192.168.7.2
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions
Default Settings
BBB Agri Type
BBB Default
BBB Hexa Type
BBB V Type
BBB X Type
QGC
Close window on exit: Always Never [✓ Only on clean exit]

Figure 2.7: Remote access of Beagle Bone Blue though Putty from windows laptop

Accessing Beagle Bone Blue on ssh Use below mentioned default credentials to access Beagle Bone Blue-

Default username – Debian

Default password – temppwd



```
debian@beaglebone: ~
login as: debian
Pre-authentication banner message from server:
| Debian GNU/Linux 9
|
| BeagleBoard.org Debian Console Image 2020-04-01
|
| Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian
|
| default username:password is [debian:temppwd]
|
End of banner message from server
debian@192.168.7.2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Feb 12 16:46:01 2021 from 100.96.1.81
debian@beaglebone:~$
```

Figure 2.8: Successful login into Debian Linux running on Beagle Bone Blue from the Windows Laptop

Connecting through wifi for wireless access in future After logging in to Beagle Bone Blue through ssh for the first time, use the below mentioned command to connect it on wifi for internet access. Also the board can also be accessed on wifi.

Step1 We need to allow the debian user to sudo without having to enter the password every (subsequent) time, enter the following command

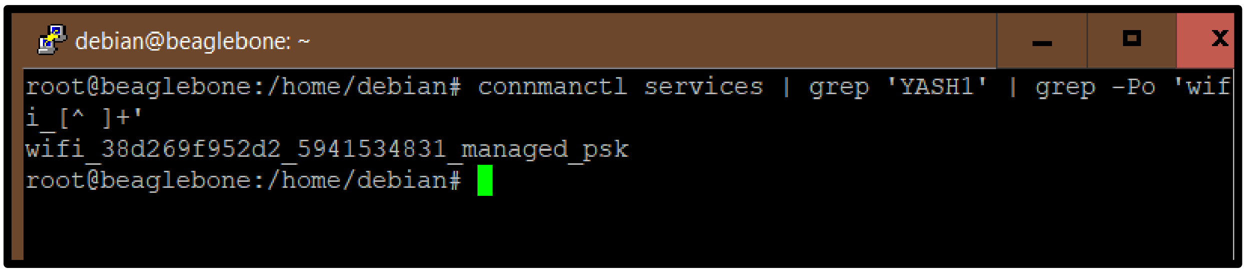
```
echo "debian ALL=(ALL) NOPASSWD: ALL" | sudo tee -a /etc/sudoers.d/debian
>/dev/null
```

Step 2 The Wifi SSID and password be automated and put in a script so that subsequent access can be on Wifi without the need of connecting with micro USB cable

```
sudo -s
```

Check Beagle Bone Blue detects the wifi, in my case wifi name is YASH1

```
connmanctl services | grep 'YASH1' | grep -Po 'wifi_[^ ]+'
```

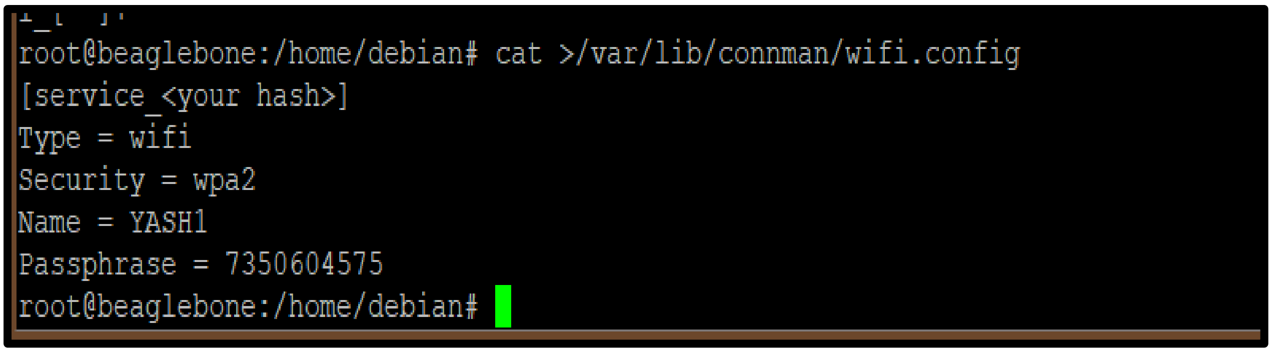


```
root@beaglebone:/home/debian# connmanctl services | grep 'YASH1' | grep -Po 'wif
i_[^ ]+'
wifi_38d269f952d2_5941534831_managed_psk
root@beaglebone:/home/debian#
```

Figure 2.9: Wifi network named YASH1 detected by Beagle Bone Blue

Step3 Add Wifi SSID and passphrase details

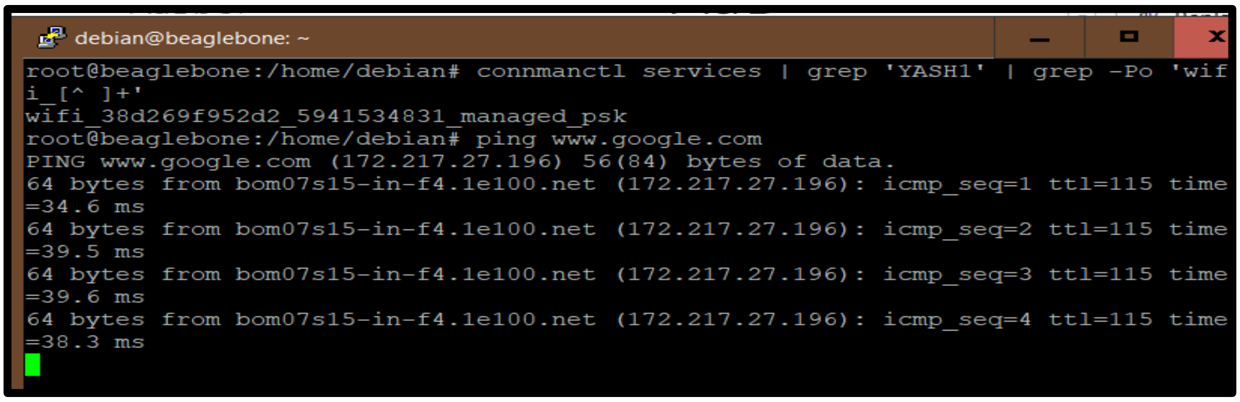
```
cat >/var/lib/connman/wifi.config
[service_<your hash>]
Type = wifi
Security = wpa2
Name = YASH1
Passphrase = 7350604575
```



```
root@beaglebone:/home/debian# cat >/var/lib/connman/wifi.config
[service_<your hash>]
Type = wifi
Security = wpa2
Name = YASH1
Passphrase = 7350604575
root@beaglebone:/home/debian#
```

Figure 2.10: Adding Wifi details to wifi.config file using cat command

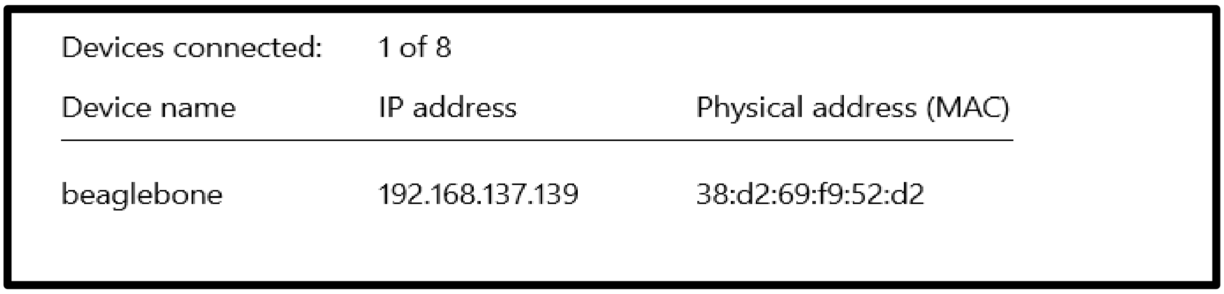
Step 4 Ctrl-D and ping www.google.com to check internet and DNS resolution is working successfully.



```
root@beaglebone:/home/debian# connmanctl services | grep 'YASH1' | grep -Po 'wif
i_[^ ]+'
wifi_38d269f952d2_5941534831_managed_psk
root@beaglebone:/home/debian# ping www.google.com
PING www.google.com (172.217.27.196) 56(84) bytes of data.
64 bytes from bom07s15-in-f4.1e100.net (172.217.27.196): icmp_seq=1 ttl=115 time
=34.6 ms
64 bytes from bom07s15-in-f4.1e100.net (172.217.27.196): icmp_seq=2 ttl=115 time
=39.5 ms
64 bytes from bom07s15-in-f4.1e100.net (172.217.27.196): icmp_seq=3 ttl=115 time
=39.6 ms
64 bytes from bom07s15-in-f4.1e100.net (172.217.27.196): icmp_seq=4 ttl=115 time
=38.3 ms
```

Figure 2.11: Connected to internet through YASH1 Wifi and successful DNS resolution by Beagle Bone Blue

Step 5 Restart Beagle Bone Blue. Use any IP Scanner application on wifi to detect IP of Beagle Bone Blue over wifi network.

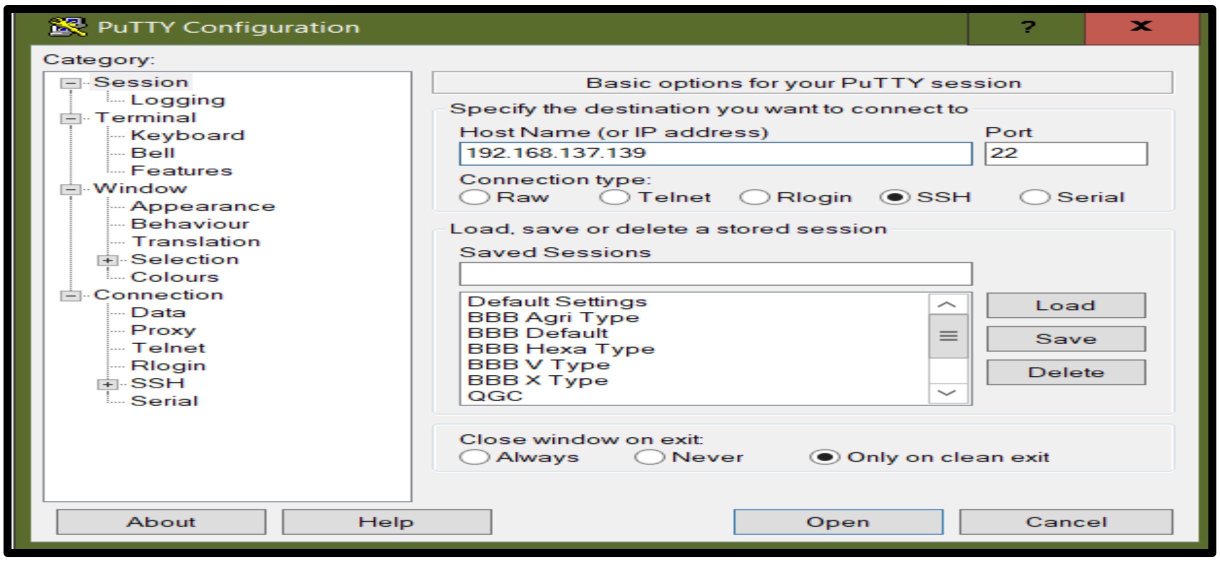


Devices connected: 1 of 8

Device name	IP address	Physical address (MAC)
beaglebone	192.168.137.139	38:d2:69:f9:52:d2

Figure 2.12: Beagle Bone Blue successfully detected over local LAN with allotted IP Address

Step 6. Use Putty to access it over that that IP which was scanned and found during step 5 above. Using Putty Application to access ssh over wifi. (No need of any micro-USB cable now). Use same login credentials as above.



PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.137.139

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

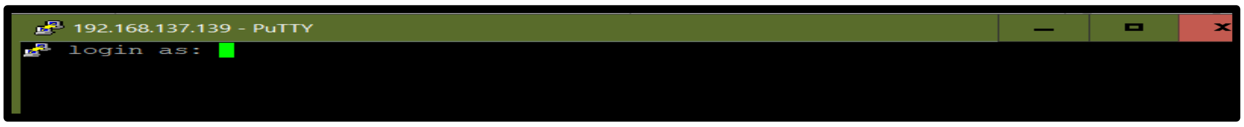
Default Settings
BBB Agri Type
BBB Default
BBB Hexa Type
BBB V Type
BBB X Type
QGC

Close window on exit: Always Never [✓ Only on clean exit]

Figure 2.13: Remote access to Beagle Bone Blue over Wifi using Putty application

Default username – Debian

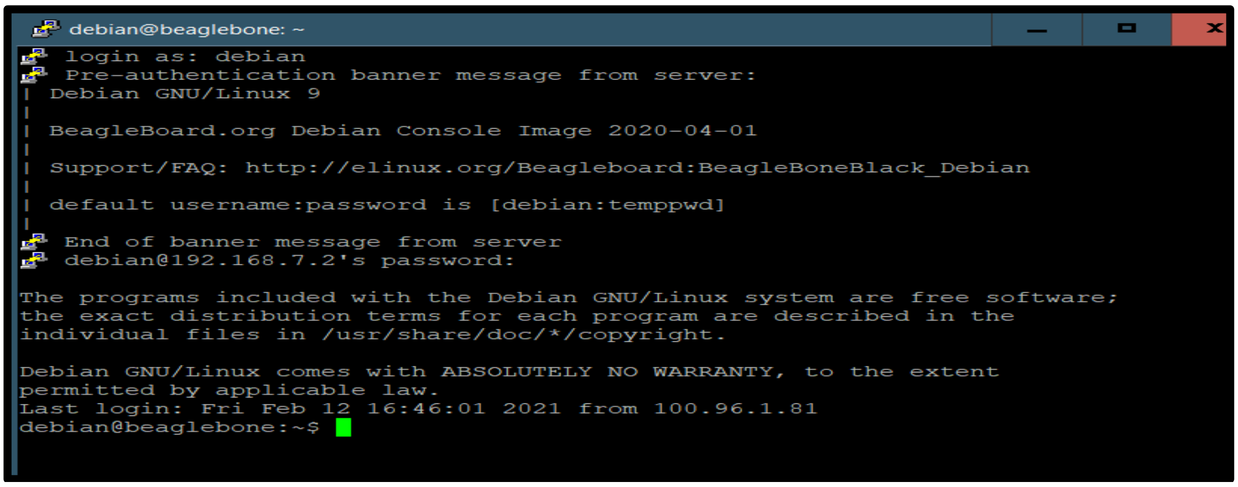
Default password – temppwd



192.168.137.139 - PuTTY

```
login as:
```

Figure 2.14: Successful access to Beagle Bone Blue over ssh over Wifi



debian@beaglebone: ~

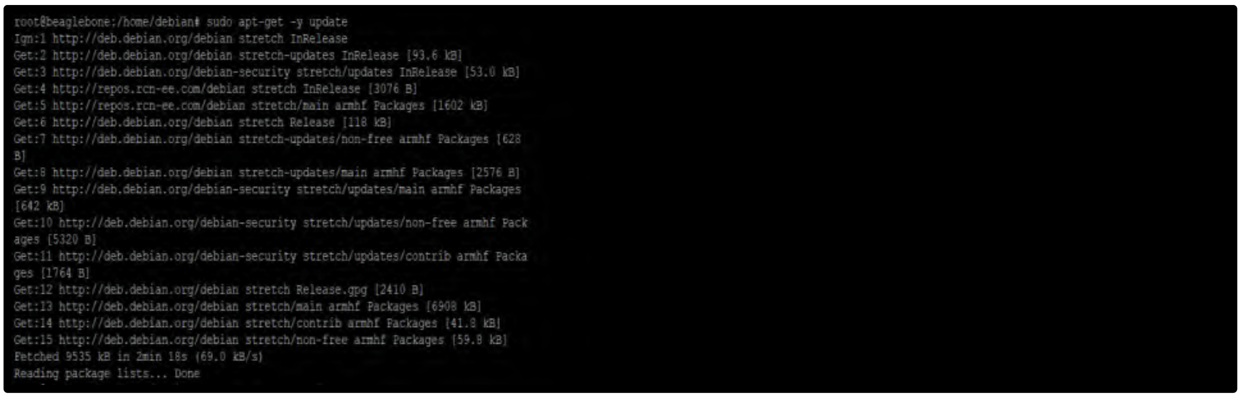
```
login as: debian
Pre-authentication banner message from server:
| Debian GNU/Linux 9
|
| BeagleBoard.org Debian Console Image 2020-04-01
|
| Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian
|
| default username:password is [debian:temppwd]
|
End of banner message from server
debian@192.168.7.2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Feb 12 16:46:01 2021 from 100.96.1.81
debian@beaglebone:~$
```

Figure 2.15: Successful login to Beagle Bone Blue over ssh over Wifi

2.2 Building autopilot stack on Beagle Bone Blue

Configuring OS on Beagle Bone Blue Update and upgrade Debian Linux OS and install requisite tools/applications

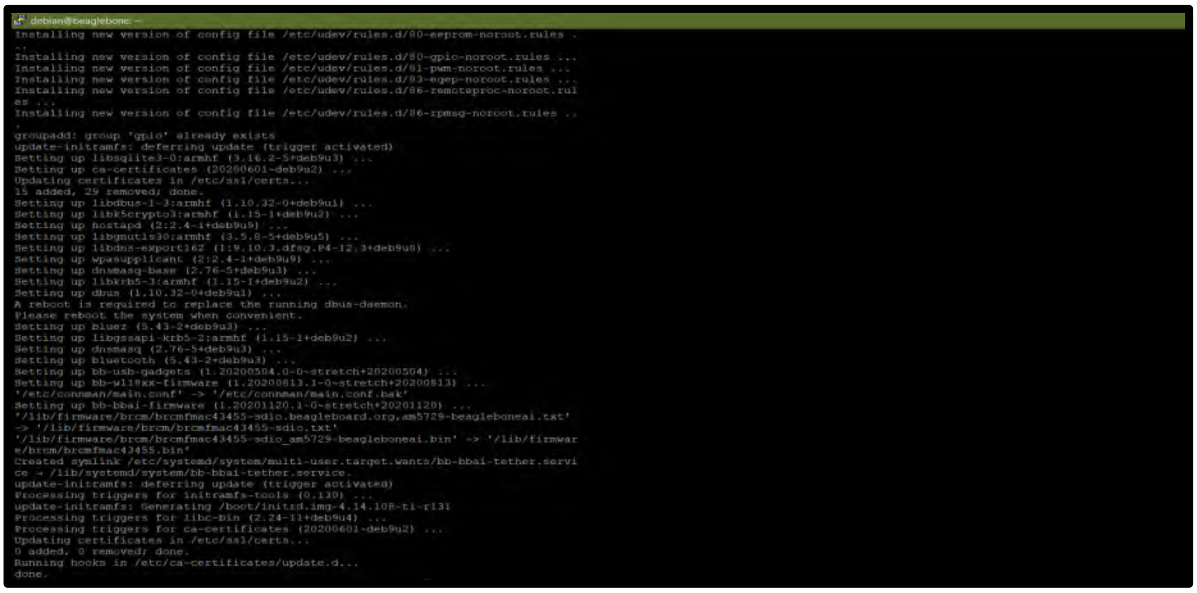
sudo apt-get -y update



```
root@beaglebone:/home/debian# sudo apt-get -y update
Ign:1 http://deb.debian.org/debian stretch InRelease
Get:2 http://deb.debian.org/debian stretch-updates InRelease [93.6 kB]
Get:3 http://deb.debian.org/debian-security stretch/updates InRelease [53.0 kB]
Get:4 http://repos.rcn-ee.com/debian stretch InRelease [3076 B]
Get:5 http://repos.rcn-ee.com/debian stretch/main armhf Packages [1602 kB]
Get:6 http://deb.debian.org/debian stretch Release [118 kB]
Get:7 http://deb.debian.org/debian stretch-updates/non-free armhf Packages [628
B]
Get:8 http://deb.debian.org/debian stretch-updates/main armhf Packages [2576 B]
Get:9 http://deb.debian.org/debian-security stretch/updates/main armhf Packages
[642 kB]
Get:10 http://deb.debian.org/debian-security stretch/updates/non-free armhf Pack
ages [5320 B]
Get:11 http://deb.debian.org/debian-security stretch/updates/contrib armhf Packa
ges [1764 B]
Get:12 http://deb.debian.org/debian stretch Release.gpg [2410 B]
Get:13 http://deb.debian.org/debian stretch/main armhf Packages [6908 kB]
Get:14 http://deb.debian.org/debian stretch/contrib armhf Packages [41.8 kB]
Get:15 http://deb.debian.org/debian stretch/non-free armhf Packages [59.8 kB]
Fetched 9535 kB in 2min 18s (69.0 kB/s)
Reading package lists... Done
```

Figure 2.16: Debian Linux successfully updated

sudo apt-get -y dist-upgrade

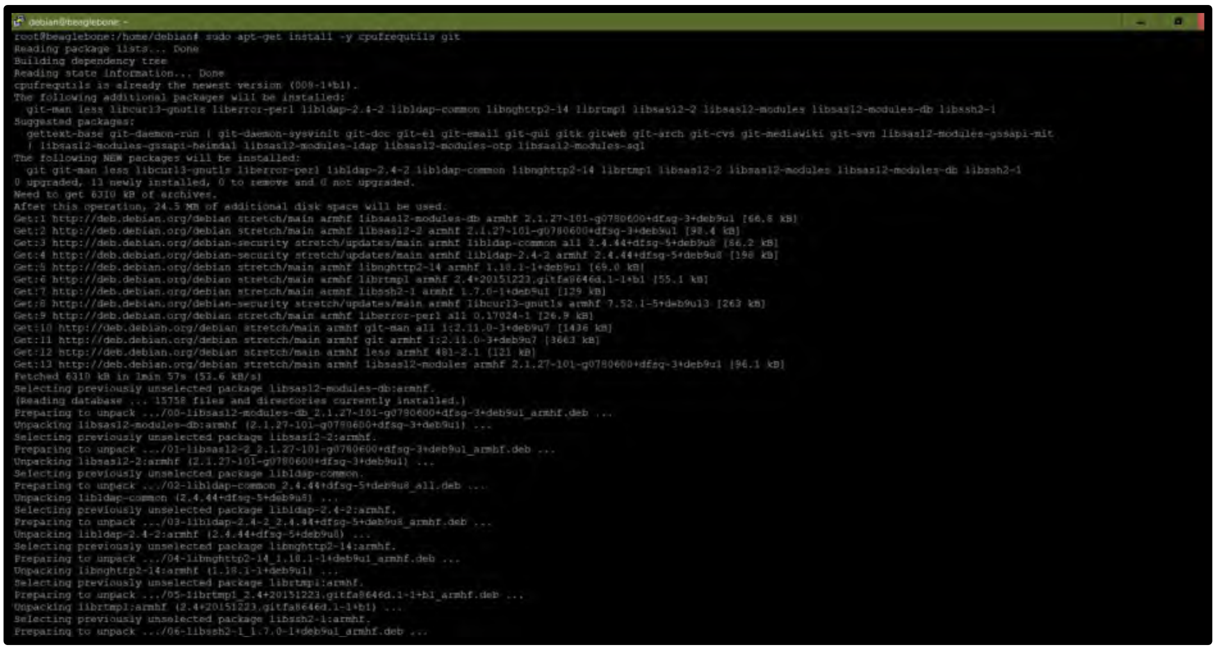


```
debian@beaglebone: ~
Installing new version of config file /etc/udev/rules.d/80-eeprom-noroot.rules .
..
Installing new version of config file /etc/udev/rules.d/80-gpio-noroot.rules ...
Installing new version of config file /etc/udev/rules.d/81-pwm-noroot.rules ...
Installing new version of config file /etc/udev/rules.d/83-eqep-noroot.rules ...
Installing new version of config file /etc/udev/rules.d/86-remoteproc-noroot.rul
es ...
Installing new version of config file /etc/udev/rules.d/86-rpmsg-noroot.rules ..
.
groupadd: group 'gpio' already exists
update-initramfs: deferring update (trigger activated)
Setting up libsqlite3-0:armhf (3.16.2-5+deb9u3) ...
Setting up ca-certificates (20200601-deb9u2) ...
Updating certificates in /etc/ssl/certs...
15 added, 29 removed; done.
Setting up libdbus-1-3:armhf (1.10.32-0+deb9u1) ...
Setting up libk5crypto3:armhf (1.15-1+deb9u2) ...
Setting up hostapd (2:2.4-1+deb9u9) ...
Setting up libgnutls30:armhf (3.5.8-5+deb9u5) ...
Setting up libdns-export162 (1:9.10.3.dfsg.P4-12.3+deb9u8) ...
Setting up wpasupplicant (2:2.4-1+deb9u9) ...
Setting up dnsmasq-base (2.76-5+deb9u3) ...
Setting up libkrb5-3:armhf (1.15-1+deb9u2) ...
Setting up dbus (1.10.32-0+deb9u1) ...
A reboot is required to replace the running dbus-daemon.
Please reboot the system when convenient.
Setting up bluez (5.43-2+deb9u3) ...
Setting up libgssapi-krb5-2:armhf (1.15-1+deb9u2) ...
Setting up dnsmasq (2.76-5+deb9u3) ...
Setting up bluetooth (5.43-2+deb9u3) ...
Setting up bb-usb-gadgets (1.20200504.0-0~stretch+20200504) ...
Setting up bb-wl18xx-firmware (1.20200813.1-0~stretch+20200813) ...
'/etc/connman/main.conf' -> '/etc/connman/main.conf.bak'
Setting up bb-bbai-firmware (1.20201120.1-0~stretch+20201120) ...
'/lib/firmware/brcm/brcmfmac43455-sdio.beagleboard.org,am5729-beagleboneai.txt'
-> '/lib/firmware/brcm/brcmfmac43455-sdio.txt'
'/lib/firmware/brcm/brcmfmac43455-sdio_am5729-beagleboneai.bin' -> '/lib/firmwar
e/brcm/brcmfmac43455.bin'
Created symlink /etc/systemd/system/multi-user.target.wants/bb-bbai-tether.servi
ce → /lib/systemd/system/bb-bbai-tether.service.
update-initramfs: deferring update (trigger activated)
Processing triggers for initramfs-tools (0.130) ...
update-initramfs: Generating /boot/initrd.img-4.14.108-ti-r131
Processing triggers for libc-bin (2.24-11+deb9u4) ...
Processing triggers for ca-certificates (20200601~deb9u2) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
```

Figure 2.17: Debian Linux successfully upgraded

Installing CPU Frequency Scaling Utility Now we need to install CPU Frequency scaling utility which enables Operating systems to scale up or down CPU frequency to save power. It can be scaled up incase of load on CPU

sudo apt-get install -y cpufrequtils git

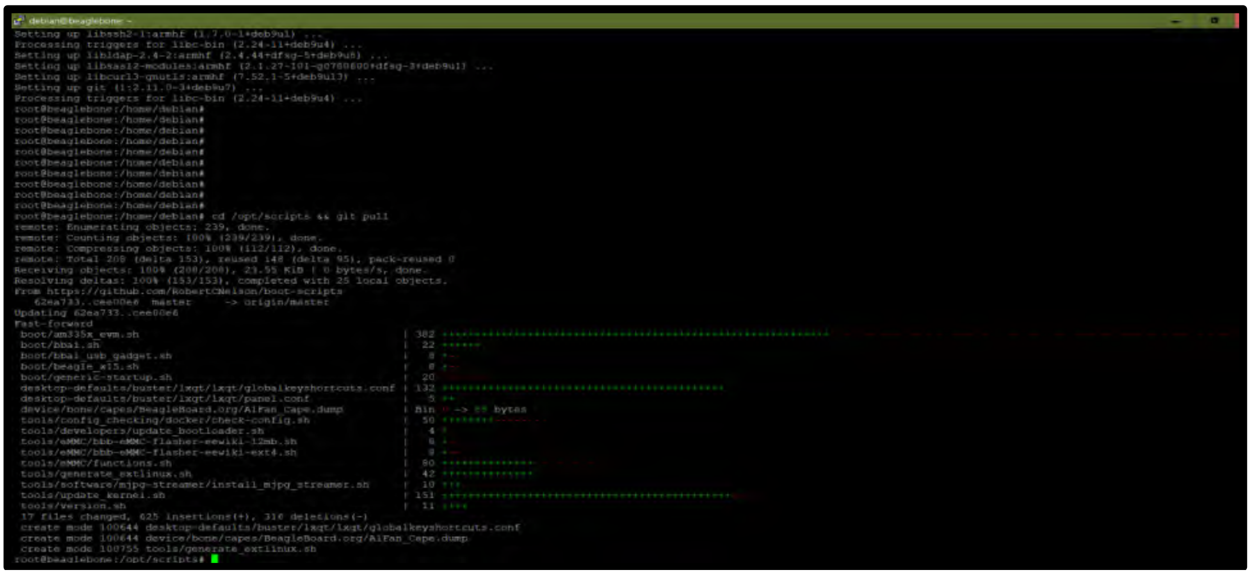


```
root@beaglebone:/home/debian# sudo apt-get install -y cpufrequtils git
Reading package lists... Done
Building dependency tree
Reading state information... Done
cpufrequtils is already the newest version (008-1+b1).
The following additional packages will be installed:
  git-man less libcurl3-gnutls liberror-perl libldap-2.4-2 libldap-common libnghttp2-14 librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libssh2-1
Suggested packages:
  gettext-base git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-arch git-cvs git-mediawiki git-svn libsasl2-modules-gssapi-mit
  | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap libsasl2-modules-otp libsasl2-modules-sql
The following NEW packages will be installed:
  git git-man less libcurl3-gnutls liberror-perl libldap-2.4-2 libldap-common libnghttp2-14 librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libssh2-1
0 upgraded, 13 newly installed, 0 to remove and 0 not upgraded.
Need to get 6310 kB of archives.
After this operation, 24.5 MB of additional disk space will be used.
Get:1 http://deb.debian.org/debian stretch/main armhf libsasl2-modules-db armhf 2.1.27~101-g0780600+dfsg-3+deb9u1 [66.8 kB]
Get:2 http://deb.debian.org/debian stretch/main armhf libsasl2-2 armhf 2.1.27~101-g0780600+dfsg-3+deb9u1 [98.4 kB]
Get:3 http://deb.debian.org/debian-security stretch/updates/main armhf libldap-common all 2.4.44+dfsg-5+deb9u8 [86.2 kB]
Get:4 http://deb.debian.org/debian-security stretch/updates/main armhf libldap-2.4-2 armhf 2.4.44+dfsg-5+deb9u8 [198 kB]
Get:5 http://deb.debian.org/debian stretch/main armhf libnghttp2-14 armhf 1.18.1-1+deb9u1 [69.0 kB]
Get:6 http://deb.debian.org/debian stretch/main armhf librtmp1 armhf 2.4+20151223.gitfa8646d.1-1+b1 [55.1 kB]
Get:7 http://deb.debian.org/debian stretch/main armhf libssh2-1 armhf 1.7.0-1+deb9u1 [129 kB]
Get:8 http://deb.debian.org/debian-security stretch/updates/main armhf libcurl3-gnutls armhf 7.52.1-5+deb9u13 [263 kB]
Get:9 http://deb.debian.org/debian stretch/main armhf liberror-perl all 0.17024-1 [26.9 kB]
Get:10 http://deb.debian.org/debian stretch/main armhf git-man all 1:2.11.0-3+deb9u7 [1436 kB]
Get:11 http://deb.debian.org/debian stretch/main armhf git armhf 1:2.11.0-3+deb9u7 [3663 kB]
Get:12 http://deb.debian.org/debian stretch/main armhf less armhf 481-2.1 [121 kB]
Get:13 http://deb.debian.org/debian stretch/main armhf libsasl2-modules armhf 2.1.27~101-g0780600+dfsg-3+deb9u1 [96.1 kB]
Fetched 6310 kB in 1min 57s (53.6 kB/s)
Selecting previously unselected package libsasl2-modules-db:armhf.
(Reading database ... 15758 files and directories currently installed.)
Preparing to unpack .../00-libsasl2-modules-db_2.1.27~101-g0780600+dfsg-3+deb9u1_armhf.deb ...
Unpacking libsasl2-modules-db:armhf (2.1.27~101-g0780600+dfsg-3+deb9u1) ...
Selecting previously unselected package libsasl2-2:armhf.
Preparing to unpack .../01-libsasl2-2_2.1.27~101-g0780600+dfsg-3+deb9u1_armhf.deb ...
Unpacking libsasl2-2:armhf (2.1.27~101-g0780600+dfsg-3+deb9u1) ...
Selecting previously unselected package libldap-common.
Preparing to unpack .../02-libldap-common_2.4.44+dfsg-5+deb9u8_all.deb ...
Unpacking libldap-common (2.4.44+dfsg-5+deb9u8) ...
Selecting previously unselected package libldap-2.4-2:armhf.
Preparing to unpack .../03-libldap-2.4-2_2.4.44+dfsg-5+deb9u8_armhf.deb ...
Unpacking libldap-2.4-2:armhf (2.4.44+dfsg-5+deb9u8) ...
Selecting previously unselected package libnghttp2-14:armhf.
Preparing to unpack .../04-libnghttp2-14_1.18.1-1+deb9u1_armhf.deb ...
Unpacking libnghttp2-14:armhf (1.18.1-1+deb9u1) ...
Selecting previously unselected package librtmp1:armhf.
Preparing to unpack .../05-librtmp1_2.4+20151223.gitfa8646d.1-1+b1_armhf.deb ...
Unpacking librtmp1:armhf (2.4+20151223.gitfa8646d.1-1+b1) ...
Selecting previously unselected package libssh2-1:armhf.
Preparing to unpack .../06-libssh2-1_1.7.0-1+deb9u1_armhf.deb ...
```

Figure 2.18: Installing CPU Frequency scaling utility

To get pre-built kernels. Real-time kernel is software which manages the time of microprocessor thereby ensuring that time-critical events required as Flight Controller Unit (FCU) can be processed efficiently. It simplifies the design of embedded systems and makes Linux act like Real Time Operating System as it allows the system to be divided into multiple independent elements called tasks. These scheduling of tasks can be done in real-time.

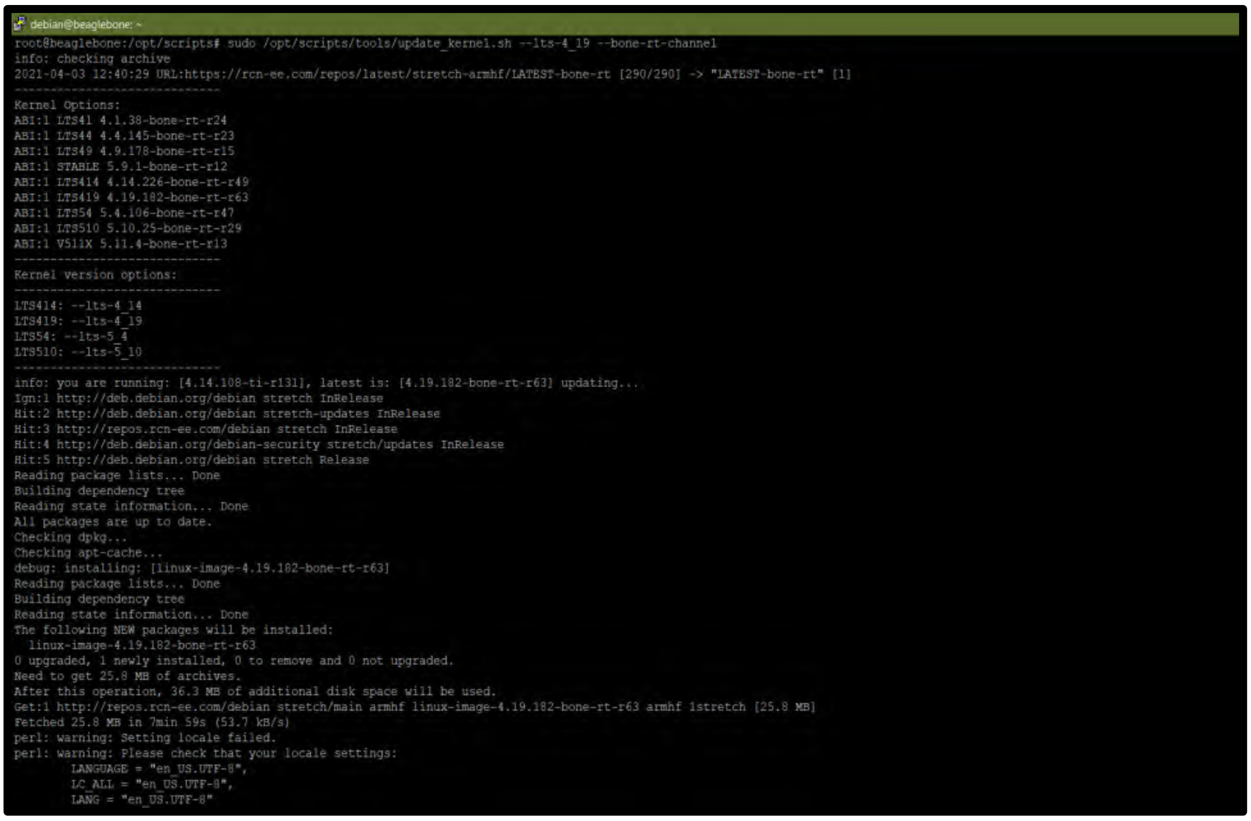
cd /opt/scripts && git pull



```
Setting up libssh2-1:armhf (1.7.0-1+deb9u1) ...
Processing triggers for libc-bin (2.24-11+deb9u4) ...
Setting up libldap-2.4-2:armhf (2.4.44+dfsg-5+deb9u8) ...
Setting up libsasl2-modules:armhf (2.1.27~101-g0780600+dfsg-3+deb9u1) ...
Setting up libcurl3-gnutls:armhf (7.52.1-5+deb9u13) ...
Setting up git (1:2.11.0-3+deb9u7) ...
Processing triggers for libc-bin (2.24-11+deb9u4) ...
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian#
root@beaglebone:/home/debian# cd /opt/scripts && git pull
remote: Enumerating objects: 239, done.
remote: Counting objects: 100% (239/239), done.
remote: Compressing objects: 100% (112/112), done.
remote: Total 208 (delta 153), reused 148 (delta 95), pack-reused 0
Receiving objects: 100% (208/208), 23.55 KiB | 0 bytes/s, done.
Resolving deltas: 100% (153/153), completed with 25 local objects.
From https://github.com/RobertCNelson/boot-scripts
   62ea733..cee00e6  master     -> origin/master
Updating 62ea733..cee00e6
Fast-forward
 boot/am335x_evm.sh                                         | 382 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++------
 boot/bbai.sh                                               |  22 ++++++
 boot/bbai_usb_gadget.sh                                    |   8 +--
 boot/beagle_x15.sh                                         |   8 +--
 boot/generic-startup.sh                                    |  20 -----
 desktop-defaults/buster/lxqt/lxqt/globalkeyshortcuts.conf  | 132 ++++++++++++++++++++++++++++++++++++++++
 desktop-defaults/buster/lxqt/lxqt/panel.conf               |   5 ++
 device/bone/capes/BeagleBoard.org/AlFan_Cape.dump          | Bin 0 -> 55 bytes
 tools/config_checking/docker/check-config.sh               |  50 ++++++++--------
 tools/developers/update_bootloader.sh                      |   4 +
 tools/eMMC/bbb-eMMC-flasher-eewiki-12mb.sh                 |   6 +-
 tools/eMMC/bbb-eMMC-flasher-eewiki-ext4.sh                 |   9 +--
 tools/eMMC/functions.sh                                    |  80 ++++++++++++++-------
 tools/generate_extlinux.sh                                 |  42 ++++++++++++++
 tools/software/mjpg-streamer/install_mjpg_streamer.sh      |  10 +++
 tools/update_kernel.sh                                     | 151 ++++++++++++++++++++++++++++++++++++++++--
 tools/version.sh                                           |  11 ++++
 17 files changed, 625 insertions(+), 316 deletions(-)
 create mode 100644 desktop-defaults/buster/lxqt/lxqt/globalkeyshortcuts.conf
 create mode 100644 device/bone/capes/BeagleBoard.org/AlFan_Cape.dump
 create mode 100755 tools/generate_extlinux.sh
root@beaglebone:/opt/scripts#
```

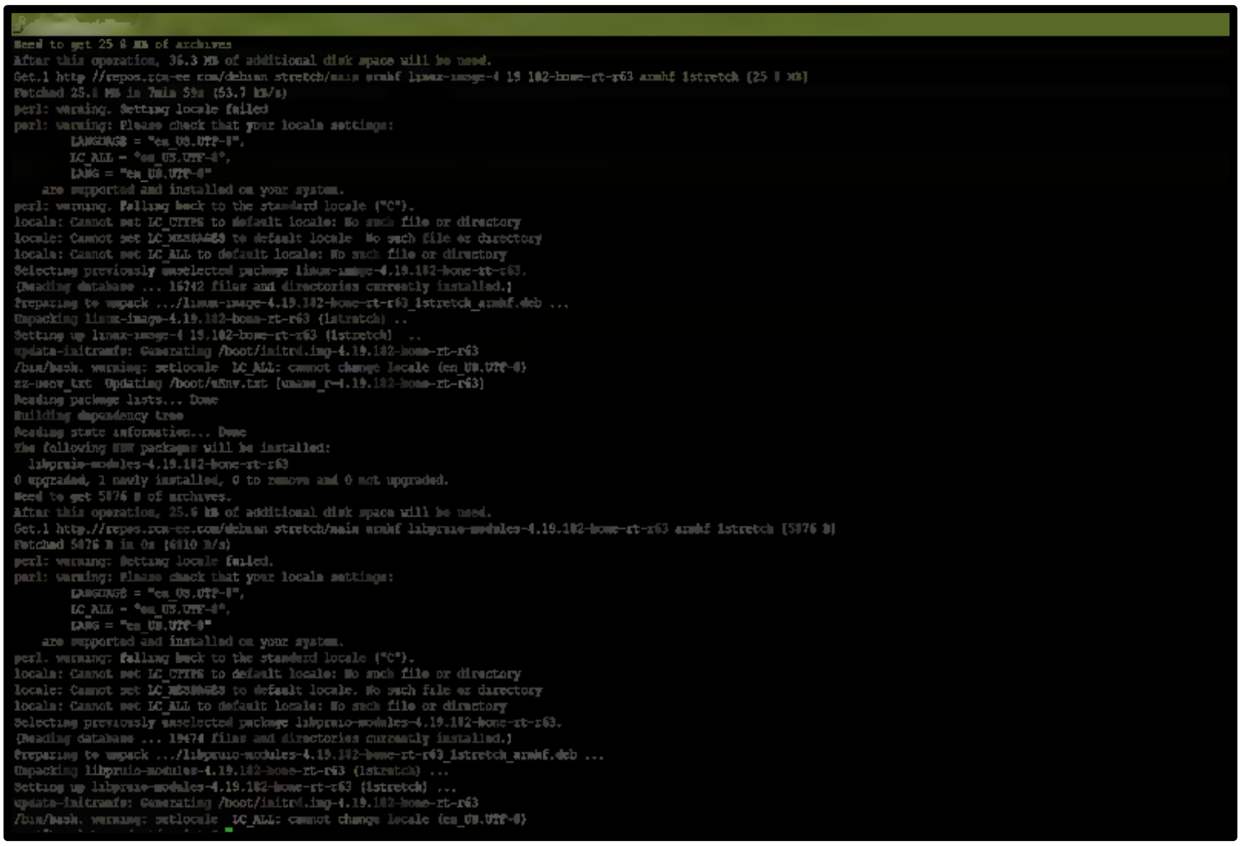
Figure 2.19: Installing real-time kernels

sudo /opt/scripts/tools/update_kernel.sh --lts-4_19 --bone-rt-channel



```
root@beaglebone:/opt/scripts# sudo /opt/scripts/tools/update_kernel.sh --lts-4_19 --bone-rt-channel
info: checking archive
2021-04-03 12:40:29 URL:https://rcn-ee.com/repos/latest/stretch-armhf/LATEST-bone-rt [290/290] -> "LATEST-bone-rt" [1]
-----------------------------
Kernel Options:
ABI:1 LTS41 4.1.38-bone-rt-r24
ABI:1 LTS44 4.4.145-bone-rt-r23
ABI:1 LTS49 4.9.178-bone-rt-r15
ABI:1 STABLE 5.9.1-bone-rt-r12
ABI:1 LTS414 4.14.226-bone-rt-r49
ABI:1 LTS419 4.19.182-bone-rt-r63
ABI:1 LTS54 5.4.106-bone-rt-r47
ABI:1 LTS510 5.10.25-bone-rt-r29
ABI:1 V511X 5.11.4-bone-rt-r13
-----------------------------
Kernel version options:
-----------------------------
LTS414: --lts-4_14
LTS419: --lts-4_19
LTS54: --lts-5_4
LTS510: --lts-5_10
-----------------------------
info: you are running: [4.14.108-ti-r131], latest is: [4.19.182-bone-rt-r63] updating...
Ign:1 http://deb.debian.org/debian stretch InRelease
Hit:2 http://deb.debian.org/debian stretch-updates InRelease
Hit:3 http://repos.rcn-ee.com/debian stretch InRelease
Hit:4 http://deb.debian.org/debian-security stretch/updates InRelease
Hit:5 http://deb.debian.org/debian stretch Release
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
Checking dpkg...
Checking apt-cache...
debug: installing: [linux-image-4.19.182-bone-rt-r63]
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  linux-image-4.19.182-bone-rt-r63
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 25.8 MB of archives.
After this operation, 36.3 MB of additional disk space will be used.
Get:1 http://repos.rcn-ee.com/debian stretch/main armhf linux-image-4.19.182-bone-rt-r63 armhf 1stretch [25.8 MB]
Fetched 25.8 MB in 7min 59s (53.7 kB/s)
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en_US.UTF-8",
        LC_ALL = "en_US.UTF-8",
        LANG = "en_US.UTF-8"
```

Figure 2.20: Updating real-time kernels 4_19



```
Need to get 25.8 MB of archives.
After this operation, 36.3 MB of additional disk space will be used.
Get:1 http://repos.rcn-ee.com/debian stretch/main armhf linux-image-4.19.182-bone-rt-r63 armhf 1stretch [25.8 MB]
Fetched 25.8 MB in 7min 59s (53.7 kB/s)
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en_US.UTF-8",
        LC_ALL = "en_US.UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
Selecting previously unselected package linux-image-4.19.182-bone-rt-r63.
(Reading database ... 16742 files and directories currently installed.)
Preparing to unpack .../linux-image-4.19.182-bone-rt-r63_1stretch_armhf.deb ...
Unpacking linux-image-4.19.182-bone-rt-r63 (1stretch) ...
Setting up linux-image-4.19.182-bone-rt-r63 (1stretch) ...
update-initramfs: Generating /boot/initrd.img-4.19.182-bone-rt-r63
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
zz-uenv_txt: Updating /boot/uEnv.txt [uname_r=4.19.182-bone-rt-r63]
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  libpruio-modules-4.19.182-bone-rt-r63
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 5876 B of archives.
After this operation, 25.6 kB of additional disk space will be used.
Get:1 http://repos.rcn-ee.com/debian stretch/main armhf libpruio-modules-4.19.182-bone-rt-r63 armhf 1stretch [5876 B]
Fetched 5876 B in 0s (6010 B/s)
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en_US.UTF-8",
        LC_ALL = "en_US.UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
Selecting previously unselected package libpruio-modules-4.19.182-bone-rt-r63.
(Reading database ... 19474 files and directories currently installed.)
Preparing to unpack .../libpruio-modules-4.19.182-bone-rt-r63_1stretch_armhf.deb ...
Unpacking libpruio-modules-4.19.182-bone-rt-r63 (1stretch) ...
Setting up libpruio-modules-4.19.182-bone-rt-r63 (1stretch) ...
update-initramfs: Generating /boot/initrd.img-4.19.182-bone-rt-r63
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
```

Figure 2.21: Updating real-time kernels 4_19

Now we need to specify the device tree binary which is to be used at Linux startup

```
sudo sed -i 's/#dtb=/dtb=am335x-boneblue.dtb/g' /boot/uEnv.txt
```

Now we need to specify the device tree overlays

```
sudo sed -i 's|#dtb_overlay=/lib/firmware/<file8>.dtbo|dtb_overlay=/lib/firmware/BB-
I2C1-00A0.dtbo\n#dtb_overlay=/lib/firmware/BB-UART4-
00A0.dtbo\n#dtb_overlay=/lib/firmware/BB-ADC-00A0.dtbo|g' /boot/uEnv.txt
```

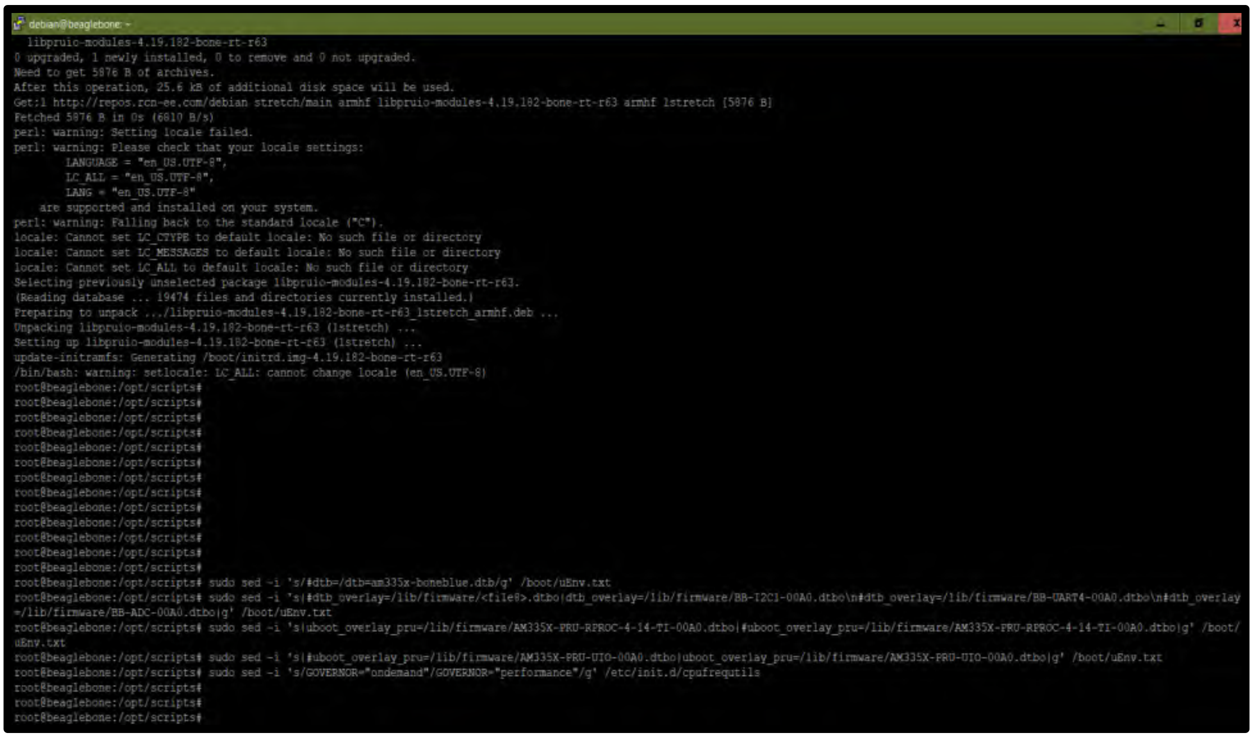
Now we need to specify the U-Boot overlay

```
sudo sed -i 's|uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-14-TI-
00A0.dtbo|#uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-14-TI-
00A0.dtbo|g' /boot/uEnv.txt
```

```
sudo sed -i 's|#uboot_overlay_pru=/lib/firmware/AM335X-PRU-UIO-
00A0.dtbo|uboot_overlay_pru=/lib/firmware/AM335X-PRU-UIO-00A0.dtbo|g'
/boot/uEnv.txt
```

To set clock frequency

```
sudo sed -i 's/GOVERNOR="ondemand"/GOVERNOR="performance"/g'
/etc/init.d/cpufrequtils
```



```
debian@beaglebone: ~
  libpruio-modules-4.19.182-bone-rt-r63
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 5876 B of archives.
After this operation, 25.6 kB of additional disk space will be used.
Get:1 http://repos.rcn-ee.com/debian stretch/main armhf libpruio-modules-4.19.182-bone-rt-r63 armhf 1stretch [5876 B]
Fetched 5876 B in 0s (6810 B/s)
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en_US.UTF-8",
        LC_ALL = "en_US.UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_MESSAGES to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
Selecting previously unselected package libpruio-modules-4.19.182-bone-rt-r63.
(Reading database ... 19474 files and directories currently installed.)
Preparing to unpack .../libpruio-modules-4.19.182-bone-rt-r63_1stretch_armhf.deb ...
Unpacking libpruio-modules-4.19.182-bone-rt-r63 (1stretch) ...
Setting up libpruio-modules-4.19.182-bone-rt-r63 (1stretch) ...
update-initramfs: Generating /boot/initrd.img-4.19.182-bone-rt-r63
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts# sudo sed -i 's/#dtb=/dtb=am335x-boneblue.dtb/g' /boot/uEnv.txt
root@beaglebone:/opt/scripts# sudo sed -i 's|#dtb_overlay=/lib/firmware/<file8>.dtbo|dtb_overlay=/lib/firmware/BB-I2C1-00A0.dtbo\n#dtb_overlay=/lib/firmware/BB-UART4-00A0.dtbo\n#dtb_overlay
=/lib/firmware/BB-ADC-00A0.dtbo|g' /boot/uEnv.txt
root@beaglebone:/opt/scripts# sudo sed -i 's|uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-14-TI-00A0.dtbo|#uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-14-TI-00A0.dtbo|g' /boot/
uEnv.txt
root@beaglebone:/opt/scripts# sudo sed -i 's|#uboot_overlay_pru=/lib/firmware/AM335X-PRU-UIO-00A0.dtbo|uboot_overlay_pru=/lib/firmware/AM335X-PRU-UIO-00A0.dtbo|g' /boot/uEnv.txt
root@beaglebone:/opt/scripts# sudo sed -i 's/GOVERNOR="ondemand"/GOVERNOR="performance"/g' /etc/init.d/cpufrequtils
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
```

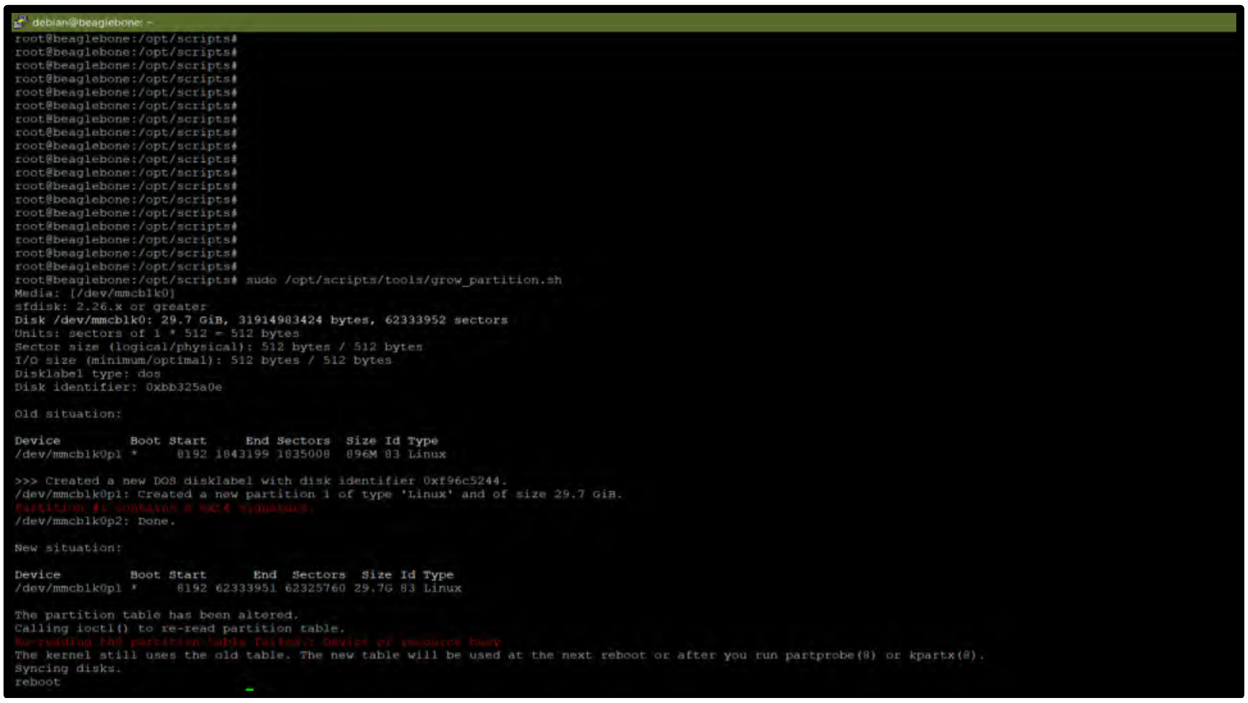
Figure 2.22: Setting clock frequency

We can disable Bluetooth as we don’t need it for FCU.

sudo systemctl disable bb-wl18xx-bluetooth.service

We can maximize the SD cards existing partition

sudo /opt/scripts/tools/grow_partition.sh



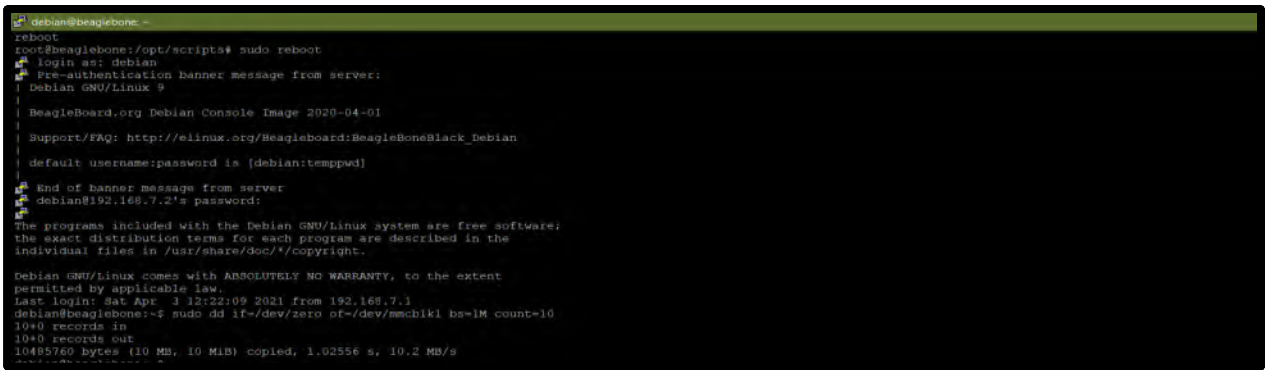
```
debian@beaglebone: ~
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts#
root@beaglebone:/opt/scripts# sudo /opt/scripts/tools/grow_partition.sh
Media: [/dev/mmcblk0]
sfdisk: 2.26.x or greater
Disk /dev/mmcblk0: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xbb325a0e

Old situation:

Device         Boot Start     End Sectors  Size Id Type
/dev/mmcblk0p1 *     8192 1843199 1835008  896M 83 Linux

>>> Created a new DOS disklabel with disk identifier 0xf96c5244.
/dev/mmcblk0p1: Created a new partition 1 of type 'Linux' and of size 29.7 GiB.
Partition #1 contains a ext4 signature.
/dev/mmcblk0p2: Done.

New situation:

Device         Boot Start      End  Sectors  Size Id Type
/dev/mmcblk0p1 *     8192 62333951 62325760 29.7G 83 Linux

The partition table has been altered.
Calling ioctl() to re-read partition table.
Re-reading the partition table failed.: Device or resource busy
The kernel still uses the old table. The new table will be used at the next reboot or after you run partprobe(8) or kpartx(8).
Syncing disks.
reboot
```

Figure 2.23: Maximize system partition

sudo reboot

I have always carried out wiping the eMMC boot sector to ensure resolution of RCOutputAioPRU.cpp:SIGBUS error

sudo dd if=/dev/zero of=/dev/mmcblk1 bs=1M count=10

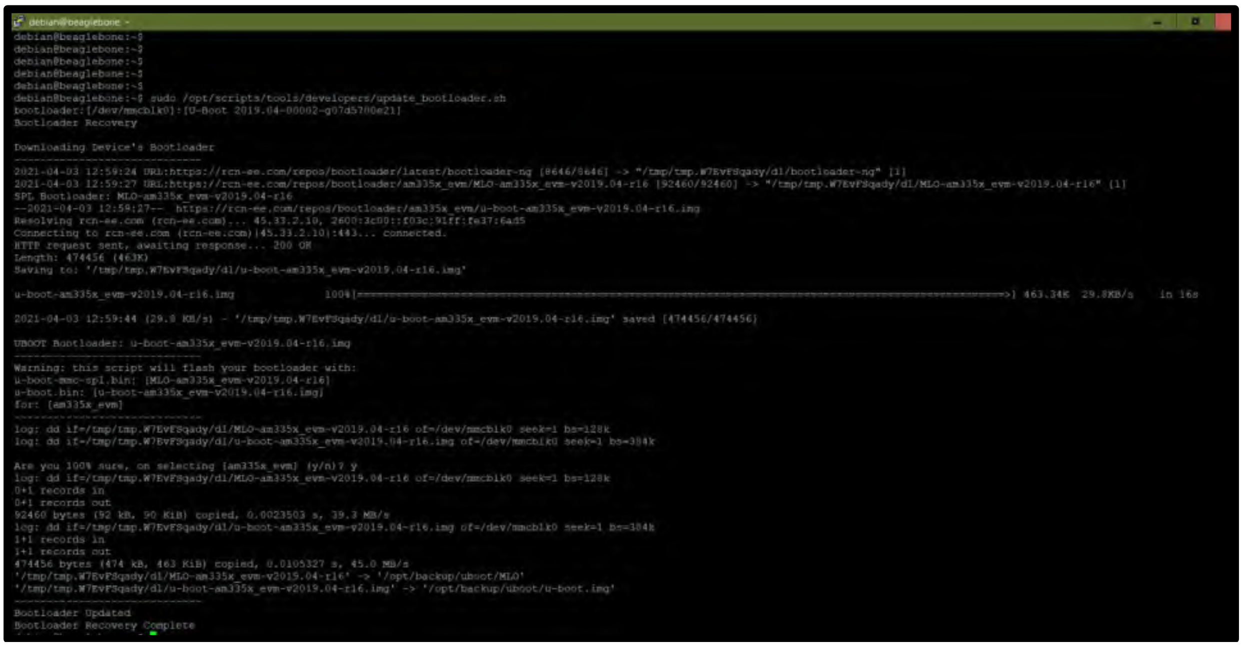


```
debian@beaglebone: ~
reboot
root@beaglebone:/opt/scripts# sudo reboot
  login as: debian
  Pre-authentication banner message from server:
| Debian GNU/Linux 9
|
| BeagleBoard.org Debian Console Image 2020-04-01
|
| Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian
|
| default username:password is [debian:temppwd]
|
  End of banner message from server
  debian@192.168.7.2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Apr  3 12:22:09 2021 from 192.168.7.1
debian@beaglebone:~$ sudo dd if=/dev/zero of=/dev/mmcblk1 bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 1.02556 s, 10.2 MB/s
```

Figure 2.24: Wiping of eMMC boot sector

Now we update the bootloader through the script

sudo /opt/scripts/tools/developers/update_bootloader.sh

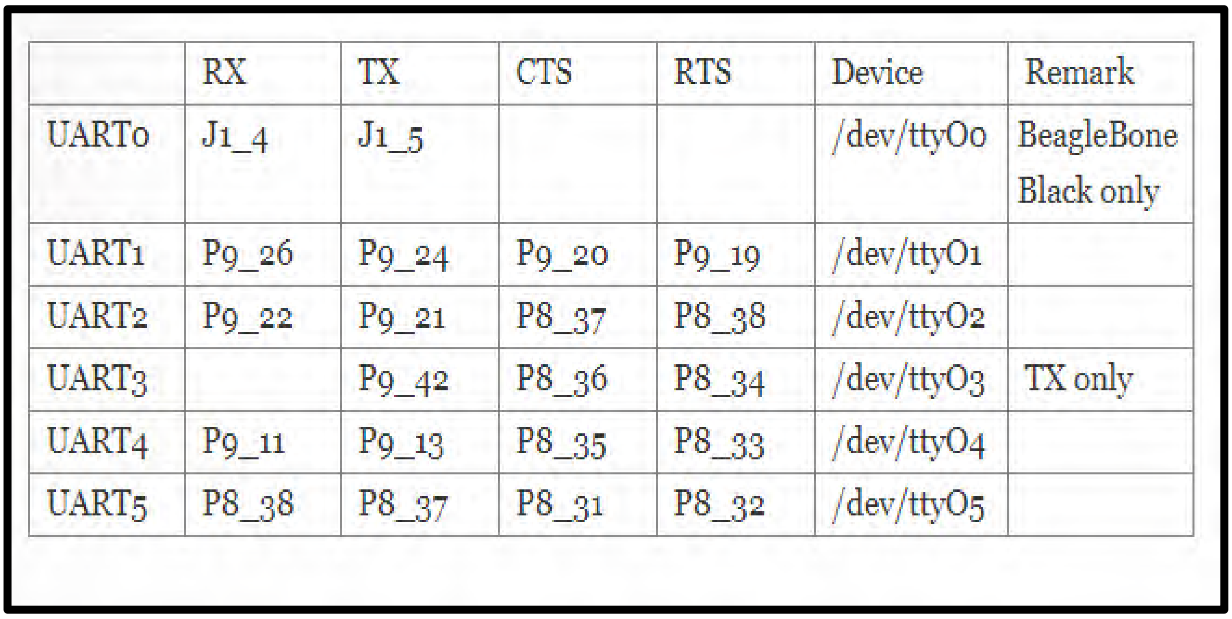


```
debian@beaglebone: ~
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$ sudo /opt/scripts/tools/developers/update_bootloader.sh
bootloader:[/dev/mmcblk0]:[U-Boot 2019.04-00002-g07d5700e21]
Bootloader Recovery

Downloading Device's Bootloader
-----------------------------
2021-04-03 12:59:24 URL:https://rcn-ee.com/repos/bootloader/latest/bootloader-ng [8646/8646] -> "/tmp/tmp.W7EvFSqady/dl/bootloader-ng" [1]
2021-04-03 12:59:27 URL:https://rcn-ee.com/repos/bootloader/am335x_evm/MLO-am335x_evm-v2019.04-r16 [92460/92460] -> "/tmp/tmp.W7EvFSqady/dl/MLO-am335x_evm-v2019.04-r16" [1]
SPL Bootloader: MLO-am335x_evm-v2019.04-r16
--2021-04-03 12:59:27--  https://rcn-ee.com/repos/bootloader/am335x_evm/u-boot-am335x_evm-v2019.04-r16.img
Resolving rcn-ee.com (rcn-ee.com)... 45.33.2.10, 2600:3c00::f03c:91ff:fe37:6ad5
Connecting to rcn-ee.com (rcn-ee.com)|45.33.2.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 474456 (463K)
Saving to: '/tmp/tmp.W7EvFSqady/dl/u-boot-am335x_evm-v2019.04-r16.img'

u-boot-am335x_evm-v2019.04-r16.img            100%[=================================================================================>] 463.34K  29.8KB/s    in 16s

2021-04-03 12:59:44 (29.8 KB/s) - '/tmp/tmp.W7EvFSqady/dl/u-boot-am335x_evm-v2019.04-r16.img' saved [474456/474456]

UBOOT Bootloader: u-boot-am335x_evm-v2019.04-r16.img
-----------------------------
Warning: this script will flash your bootloader with:
u-boot-mmc-spl.bin: [MLO-am335x_evm-v2019.04-r16]
u-boot.bin: [u-boot-am335x_evm-v2019.04-r16.img]
for: [am335x_evm]
-----------------------------
log: dd if=/tmp/tmp.W7EvFSqady/dl/MLO-am335x_evm-v2019.04-r16 of=/dev/mmcblk0 seek=1 bs=128k
log: dd if=/tmp/tmp.W7EvFSqady/dl/u-boot-am335x_evm-v2019.04-r16.img of=/dev/mmcblk0 seek=1 bs=384k

Are you 100% sure, on selecting [am335x_evm] (y/n)? y
log: dd if=/tmp/tmp.W7EvFSqady/dl/MLO-am335x_evm-v2019.04-r16 of=/dev/mmcblk0 seek=1 bs=128k
0+1 records in
0+1 records out
92460 bytes (92 kB, 90 KiB) copied, 0.0023503 s, 39.3 MB/s
log: dd if=/tmp/tmp.W7EvFSqady/dl/u-boot-am335x_evm-v2019.04-r16.img of=/dev/mmcblk0 seek=1 bs=384k
1+1 records in
1+1 records out
474456 bytes (474 kB, 463 KiB) copied, 0.0105327 s, 45.0 MB/s
'/tmp/tmp.W7EvFSqady/dl/MLO-am335x_evm-v2019.04-r16' -> '/opt/backup/uboot/MLO'
'/tmp/tmp.W7EvFSqady/dl/u-boot-am335x_evm-v2019.04-r16.img' -> '/opt/backup/uboot/u-boot.img'
-----------------------------
Bootloader Updated
Bootloader Recovery Complete
```

Figure 2.25: Updating the bootloader

Putting Ardupilot Autoilot Stack application on Beagle Bone Blue

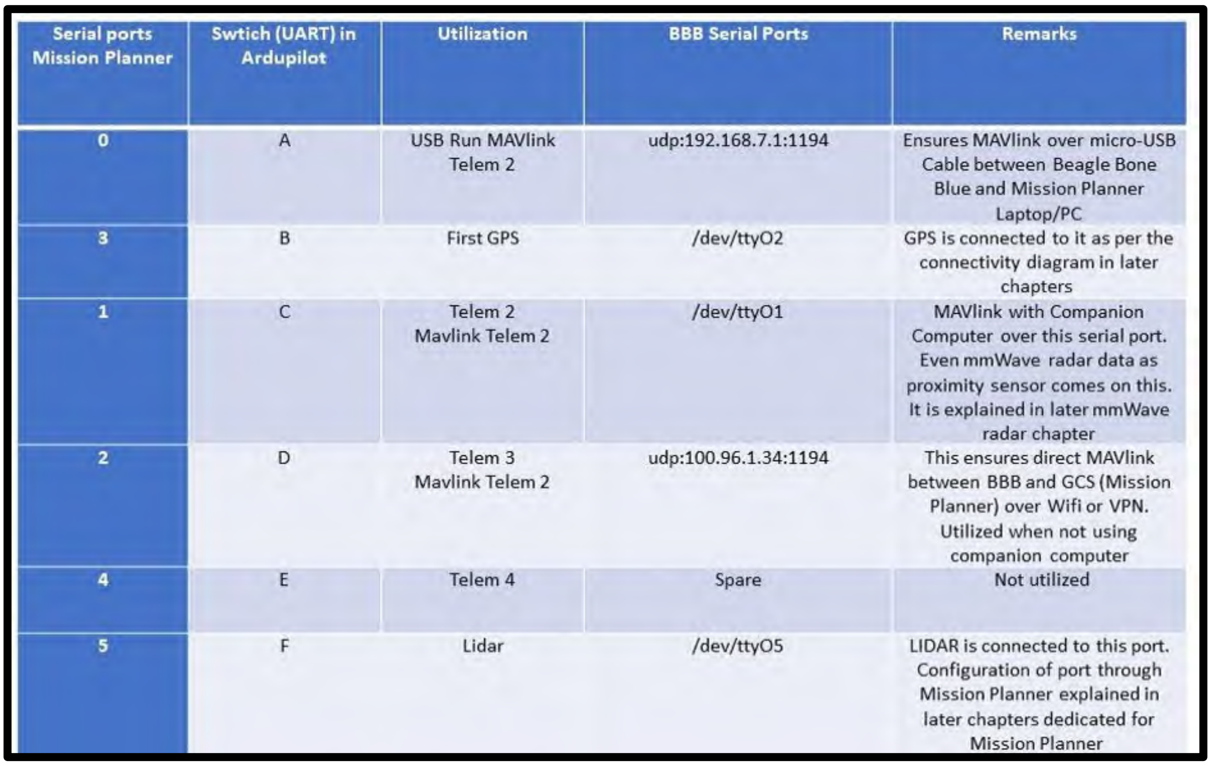
Step1 Before we begin the Ardupilot autopilot application stack build up, we need to understand the Ardupilot environment and its configuration file to be created at location /etc/default/ardupilot. The UARTS map to pin configuration on Beagle Bone Blue side is as below:-



	RX	TX	CTS	RTS	Device	Remark
UART0	J1_4	J1_5			/dev/ttyO0	BeagleBone Black only
UART1	P9_26	P9_24	P9_20	P9_19	/dev/ttyO1	
UART2	P9_22	P9_21	P8_37	P8_38	/dev/ttyO2	
UART3		P9_42	P8_36	P8_34	/dev/ttyO3	TX only
UART4	P9_11	P9_13	P8_35	P8_33	/dev/ttyO4	
UART5	P8_38	P8_37	P8_31	P8_32	/dev/ttyO5	

Figure 2.26: UART map to pins

The Ardupilot autopilot application supports upto 8 x UARTs. The mapping done by me for the utilization is as per table below: -



Serial ports Mission Planner	Swtich (UART) in Ardupilot	Utilization	BBB Serial Ports	Remarks
0	A	USB Run MAVlink Telem 2	udp:192.168.7.1:1194	Ensures MAVlink over micro-USB Cable between Beagle Bone Blue and Mission Planner Laptop/PC
3	B	First GPS	/dev/ttyO2	GPS is connected to it as per the connectivity diagram in later chapters
1	C	Telem 2 Mavlink Telem 2	/dev/ttyO1	MAVlink with Companion Computer over this serial port. Even mmWave radar data as proximity sensor comes on this. It is explained in later mmWave radar chapter
2	D	Telem 3 Mavlink Telem 2	udp:100.96.1.34:1194	This ensures direct MAVlink between BBB and GCS (Mission Planner) over Wifi or VPN. Utilized when not using companion computer
4	E	Telem 4	Spare	Not utilized
5	F	Lidar	/dev/ttyO5	LIDAR is connected to this port. Configuration of port through Mission Planner explained in later chapters dedicated for Mission Planner

Figure 2.27: Suggested UART mapping between Beagle Bone Blue and Ardupilot application

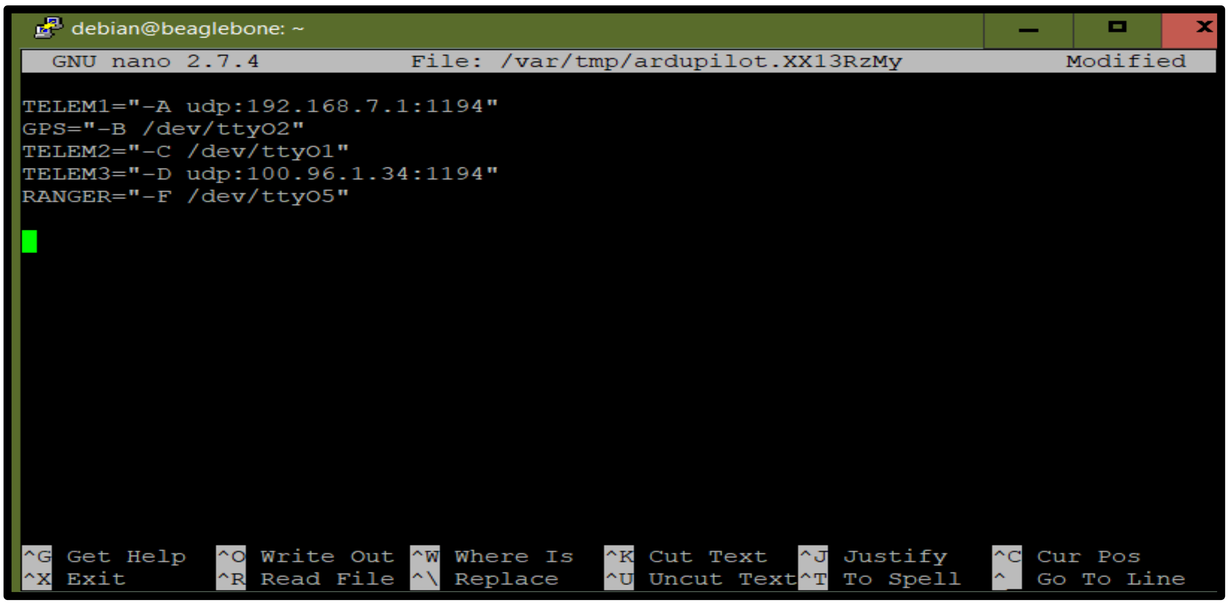
The IP Addresses in this table are of Ground Control Station (GCS) and must be changed as the existing network. This scheme of IP Addresses have been explained in later chapter on concept of MAVlink over VPN. The concept of MAVlink has also been explained in following chapters.

Step2 After getting the table ready as per our requirement, we now need to create ardupilot environment file with the data from table above: -

sudo nano /etc/default/ardupilot (later for editing use sudoedit /etc/default/ardupilot)

Add the blow mentioned lines

TELEM1="-A udp:192.168.7.1:1194"
GPS="-B /dev/ttyO2"
TELEM2="-C /dev/ttyO1"
TELEM3="-D udp:100.96.1.34:1194"
RANGER="-F /dev/ttyO5"



```
debian@beaglebone: ~
  GNU nano 2.7.4              File: /var/tmp/ardupilot.XX13RzMy                      Modified

TELEM1="-A udp:192.168.7.1:1194"
GPS="-B /dev/ttyO2"
TELEM2="-C /dev/ttyO1"
TELEM3="-D udp:100.96.1.34:1194"
RANGER="-F /dev/ttyO5"


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Figure 2.28: Creating Ardupilot environment file as per table above

Ctrl+O and enter to save configuration followed by and Ctrl+X to exit

Step 3 Create system file for running Arducopter (version of Ardupilot) as a service at startup

sudo nano /lib/systemd/system/arducopter.service

Add the below mentioned lines

[Unit]

Description=ArduCopter Service

After=networking.service

StartLimitIntervalSec=0

Conflicts=arduplane.service ardurover.service antennatracker.service

[Service]

EnvironmentFile=/etc/default/ardupilot

ExecStartPre=/usr/bin/ardupilot/aphw

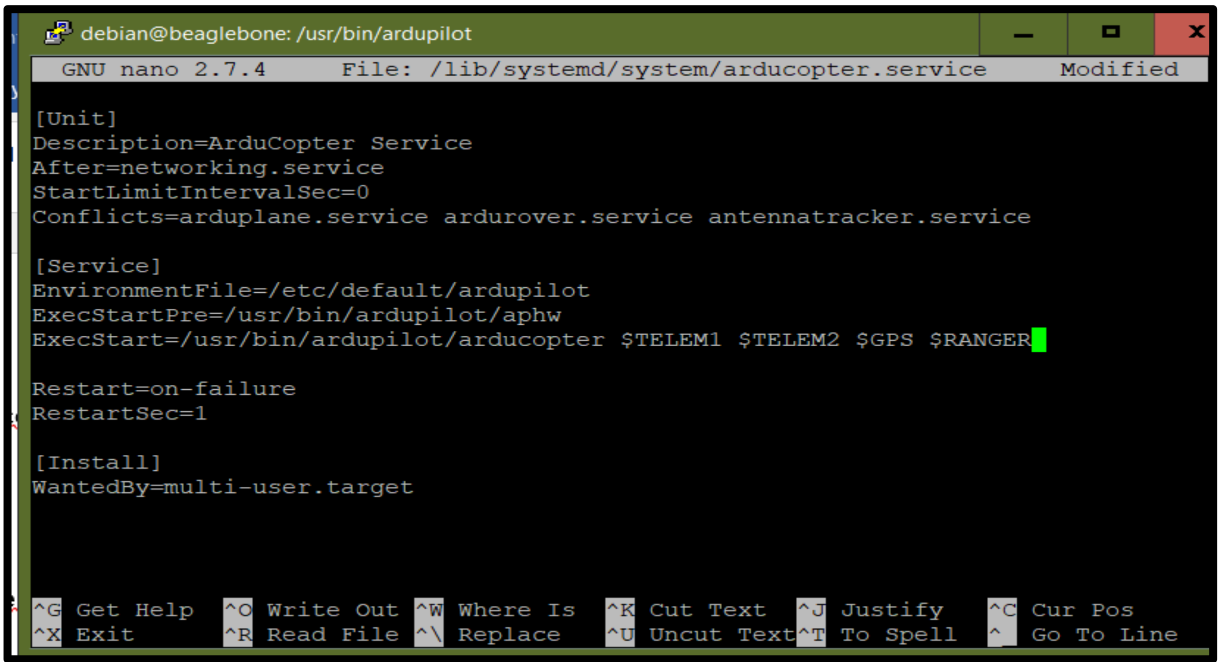
ExecStart=/usr/bin/ardupilot/arducopter $TELEM1 $TELEM2 $TELEM3 $GPS $RANGER

Restart=on-failure

RestartSec=1

[Install]

WantedBy=multi-user.target



```
debian@beaglebone: /usr/bin/ardupilot
  GNU nano 2.7.4        File: /lib/systemd/system/arducopter.service        Modified

[Unit]
Description=ArduCopter Service
After=networking.service
StartLimitIntervalSec=0
Conflicts=arduplane.service ardurover.service antennatracker.service

[Service]
EnvironmentFile=/etc/default/ardupilot
ExecStartPre=/usr/bin/ardupilot/aphw
ExecStart=/usr/bin/ardupilot/arducopter $TELEM1 $TELEM2 $GPS $RANGER

Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Figure 2.29: Creating arducopter.service for execution at startup as service

Ctrl+O and Ctrl+X to save configuration

Step 4 We need to **c**reate ardupilot directory where we will build and compile the Ardupilot autopilot stack.

```
sudo mkdir -p /usr/bin/ardupilot
```

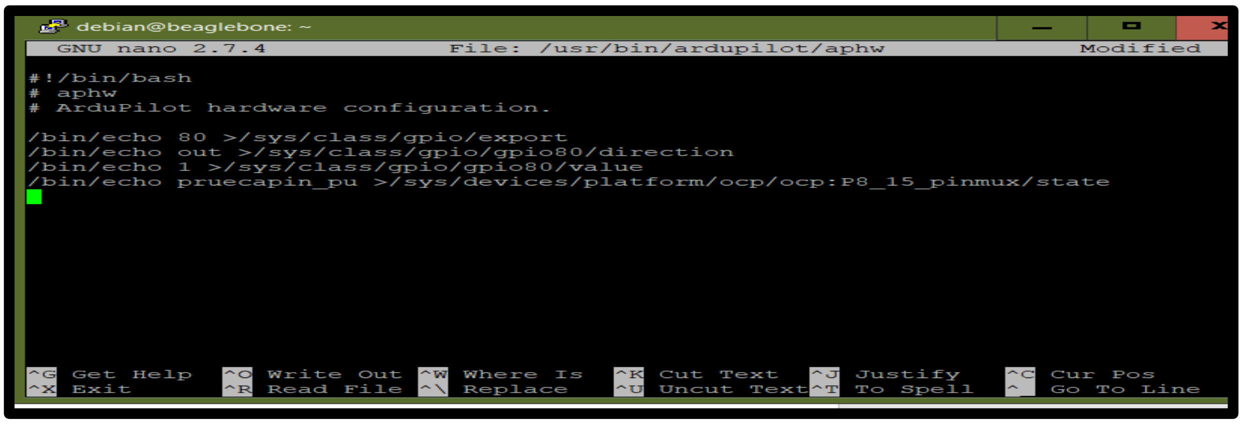
Step 5 We need to create Ardupilot hardware configuration file aphw at location /usr/bin/ardupilot/aphw, which is run by the services prior to running the Ardupilot executables.

```
sudo nano /usr/bin/ardupilot/aphw
```

Add the below mentioned lines

```
#!/bin/bash
# aphw
# ArduPilot hardware configuration.

#/bin/echo 80 >/sys/class/gpio/export
#/bin/echo out >/sys/class/gpio/gpio80/direction
#/bin/echo 1 >/sys/class/gpio/gpio80/value
/bin/echo pruecapin_pu >/sys/devices/platform/ocp/ocp:P8_15_pinmux/state
```

The lines 5 to 7 has been commented out by me as I don't wish to activate or switch on power to the Beagle Bone Blue's +5V servo rail. I have made a conscious effort to ensure that power of 5V is fed directly from DC-DC convertor to the sensor or peripheral and nothing is powered from Beagle Bone Blue. This ensures that Beagle Bone Blue is protected from any power related issues. However, the option of doing that exists in case we uncomment the lines. In order to draw power from Beagle Bone Blue esp 5V, the recommended input power is 12V-18V through DC Barrel jack. 5V snt available when powering BBB through micro USB cable. I have provided 12V to Beagle Bone Blue through Barrel jack just to ensure enough power. Also, the only module to draw power directly from Beagle Bone Blue is the 4G LTE module connected on the USB 2.0 of Beagle Bone Blue (5V and 500mA)



```
debian@beaglebone: ~
  GNU nano 2.7.4              File: /usr/bin/ardupilot/aphw                    Modified

#!/bin/bash
# aphw
# ArduPilot hardware configuration.

/bin/echo 80 >/sys/class/gpio/export
/bin/echo out >/sys/class/gpio/gpio80/direction
/bin/echo 1 >/sys/class/gpio/gpio80/value
/bin/echo pruecapin_pu >/sys/devices/platform/ocp/ocp:P8_15_pinmux/state






^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Figure 2.30: Creating arducopter hardware configuration file

Ctrl+O and enter to save configuration followed by and Ctrl+X to exit

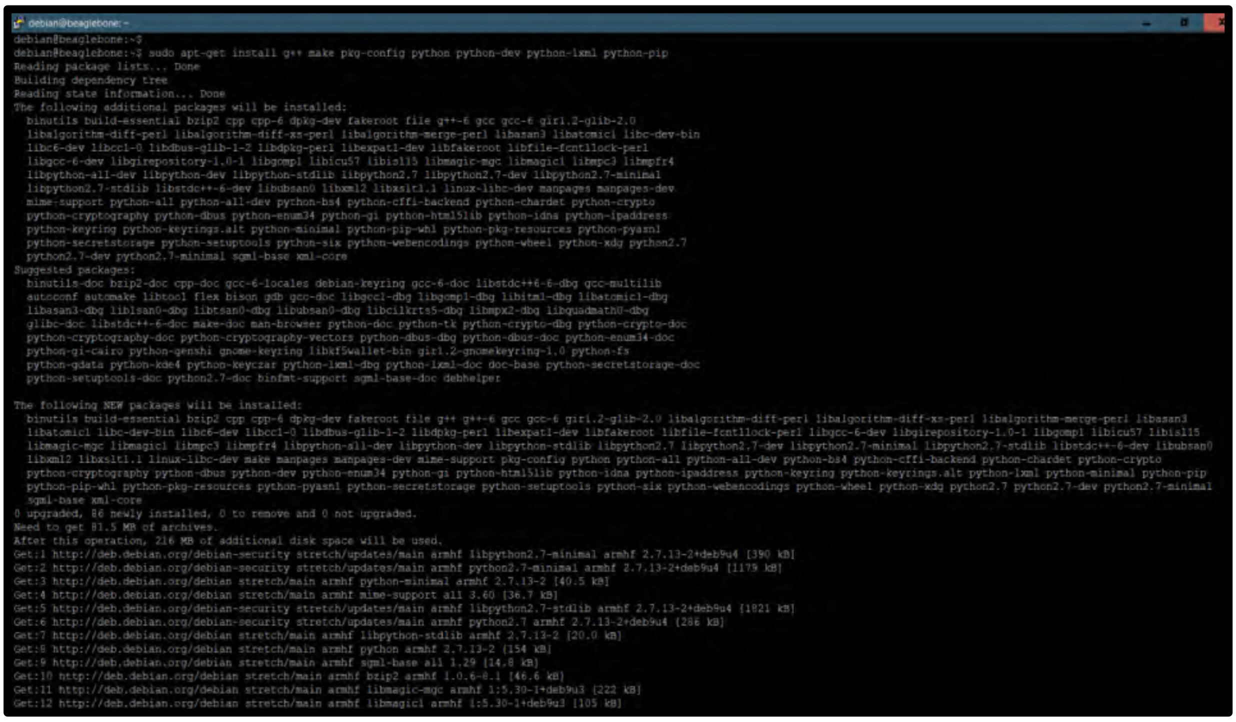
Step 6 Give permission to the file/folder

sudo chmod 0755 /usr/bin/ardupilot/aphw

sudo chmod 0755 /usr/bin/ardupilot/a*

Step 7 Compiling Ardupilot on Beagle Bone Blue itself. We need to install the requisite packages before:-

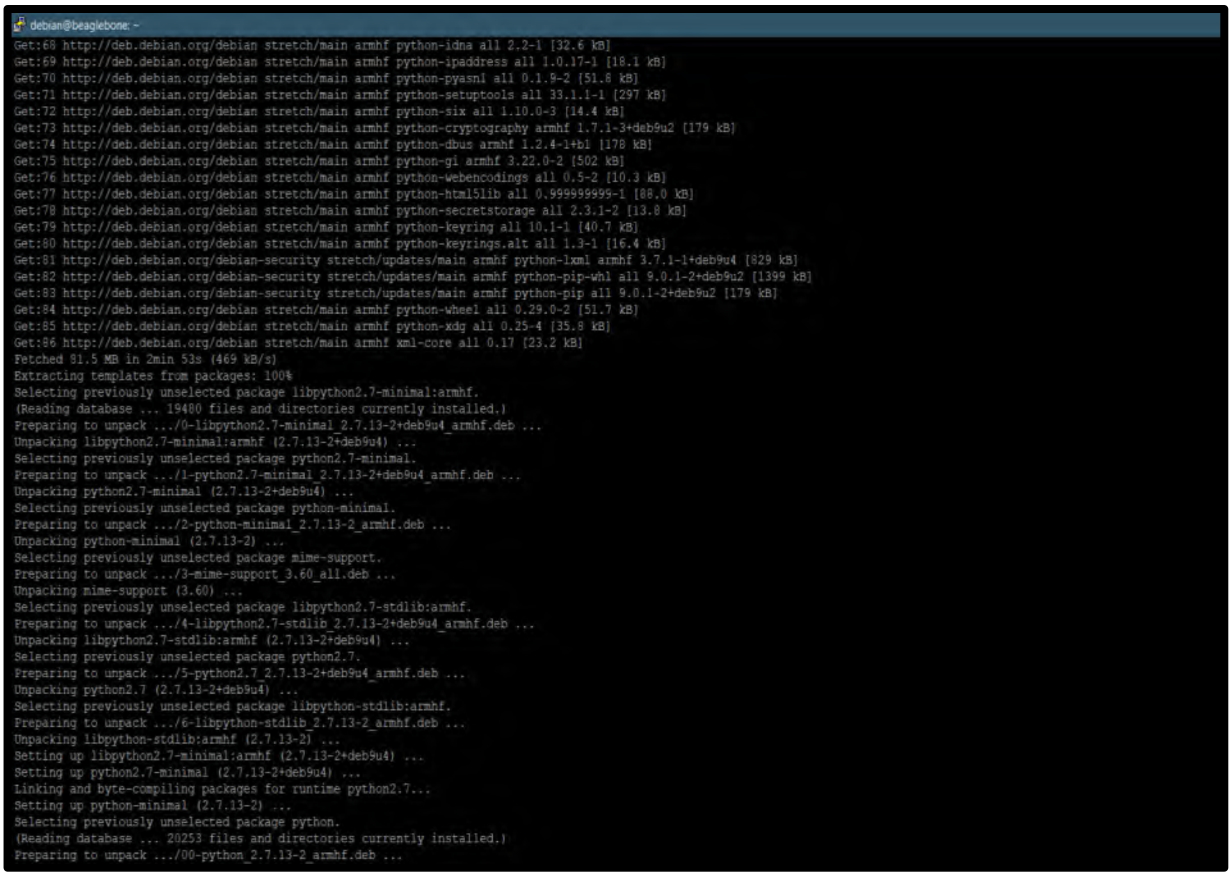
sudo apt-get install g++ make pkg-config python python-dev python-lxml python-pip



```
debian@beaglebone: ~
debian@beaglebone:~$
debian@beaglebone:~$ sudo apt-get install g++ make pkg-config python python-dev python-lxml python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils build-essential bzip2 cpp cpp-6 dpkg-dev fakeroot file g++-6 gcc gcc-6 gir1.2-glib-2.0
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan3 libatomic1 libc-dev-bin
  libc6-dev libcc1-0 libdbus-glib-1-2 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl
  libgcc-6-dev libgirepository-1.0-1 libgomp1 libicu57 libisl15 libmagic-mgc libmagic1 libmpc3 libmpfr4
  libpython-all-dev libpython-dev libpython-stdlib libpython2.7 libpython2.7-dev libpython2.7-minimal
  libpython2.7-stdlib libstdc++-6-dev libubsan0 libxml2 libxslt1.1 linux-libc-dev manpages manpages-dev
  mime-support python-all python-all-dev python-bs4 python-cffi-backend python-chardet python-crypto
  python-cryptography python-dbus python-enum34 python-gi python-html5lib python-idna python-ipaddress
  python-keyring python-keyrings.alt python-minimal python-pip-whl python-pkg-resources python-pyasn1
  python-secretstorage python-setuptools python-six python-webencodings python-wheel python-xdg python2.7
  python2.7-dev python2.7-minimal sgml-base xml-core
Suggested packages:
  binutils-doc bzip2-doc cpp-doc gcc-6-locales debian-keyring gcc-6-doc libstdc++6-6-dbg gcc-multilib
  autoconf automake libtool flex bison gdb gcc-doc libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg
  libasan3-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx2-dbg libquadmath0-dbg
  glibc-doc libstdc++-6-doc make-doc man-browser python-doc python-tk python-crypto-dbg python-crypto-doc
  python-cryptography-doc python-cryptography-vectors python-dbus-dbg python-dbus-doc python-enum34-doc
  python-gi-cairo python-genshi gnome-keyring libkf5wallet-bin gir1.2-gnomekeyring-1.0 python-fs
  python-gdata python-kde4 python-keyczar python-lxml-dbg python-lxml-doc doc-base python-secretstorage-doc
  python-setuptools-doc python2.7-doc binfmt-support sgml-base-doc debhelper

The following NEW packages will be installed:
  binutils build-essential bzip2 cpp cpp-6 dpkg-dev fakeroot file g++ g++-6 gcc gcc-6 gir1.2-glib-2.0 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan3
  libatomic1 libc-dev-bin libc6-dev libcc1-0 libdbus-glib-1-2 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-6-dev libgirepository-1.0-1 libgomp1 libicu57 libisl15
  libmagic-mgc libmagic1 libmpc3 libmpfr4 libpython-all-dev libpython-dev libpython-stdlib libpython2.7 libpython2.7-dev libpython2.7-minimal libpython2.7-stdlib libstdc++-6-dev libubsan0
  libxml2 libxslt1.1 linux-libc-dev make manpages manpages-dev mime-support pkg-config python python-all python-all-dev python-bs4 python-cffi-backend python-chardet python-crypto
  python-cryptography python-dbus python-dev python-enum34 python-gi python-html5lib python-idna python-ipaddress python-keyring python-keyrings.alt python-lxml python-minimal python-pip
  python-pip-whl python-pkg-resources python-pyasn1 python-secretstorage python-setuptools python-six python-webencodings python-wheel python-xdg python2.7 python2.7-dev python2.7-minimal
  sgml-base xml-core
0 upgraded, 86 newly installed, 0 to remove and 0 not upgraded.
Need to get 81.5 MB of archives.
After this operation, 216 MB of additional disk space will be used.
Get:1 http://deb.debian.org/debian-security stretch/updates/main armhf libpython2.7-minimal armhf 2.7.13-2+deb9u4 [390 kB]
Get:2 http://deb.debian.org/debian-security stretch/updates/main armhf python2.7-minimal armhf 2.7.13-2+deb9u4 [1179 kB]
Get:3 http://deb.debian.org/debian stretch/main armhf python-minimal armhf 2.7.13-2 [40.5 kB]
Get:4 http://deb.debian.org/debian stretch/main armhf mime-support all 3.60 [36.7 kB]
Get:5 http://deb.debian.org/debian-security stretch/updates/main armhf libpython2.7-stdlib armhf 2.7.13-2+deb9u4 [1821 kB]
Get:6 http://deb.debian.org/debian-security stretch/updates/main armhf python2.7 armhf 2.7.13-2+deb9u4 [286 kB]
Get:7 http://deb.debian.org/debian stretch/main armhf libpython-stdlib armhf 2.7.13-2 [20.0 kB]
Get:8 http://deb.debian.org/debian stretch/main armhf python armhf 2.7.13-2 [154 kB]
Get:9 http://deb.debian.org/debian stretch/main armhf sgml-base all 1.29 [14.8 kB]
Get:10 http://deb.debian.org/debian stretch/main armhf bzip2 armhf 1.0.6-8.1 [46.6 kB]
Get:11 http://deb.debian.org/debian stretch/main armhf libmagic-mgc armhf 1:5.30-1+deb9u3 [222 kB]
Get:12 http://deb.debian.org/debian stretch/main armhf libmagic1 armhf 1:5.30-1+deb9u3 [105 kB]
```

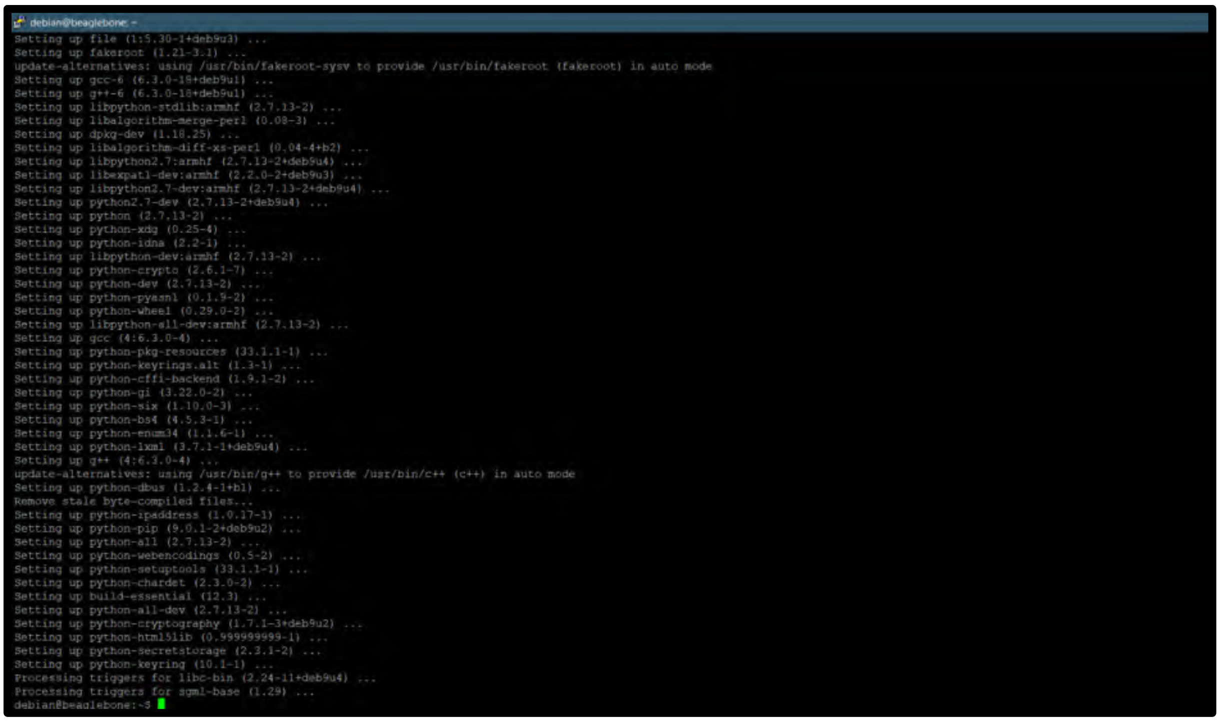
Figure 2.31: Installing pre-requisites for compiling Ardupilot application on Beagle Bone

Blue itself.



```
debian@beaglebone: ~
Get:68 http://deb.debian.org/debian stretch/main armhf python-idna all 2.2-1 [32.6 kB]
Get:69 http://deb.debian.org/debian stretch/main armhf python-ipaddress all 1.0.17-1 [18.1 kB]
Get:70 http://deb.debian.org/debian stretch/main armhf python-pyasn1 all 0.1.9-2 [51.8 kB]
Get:71 http://deb.debian.org/debian stretch/main armhf python-setuptools all 33.1.1-1 [297 kB]
Get:72 http://deb.debian.org/debian stretch/main armhf python-six all 1.10.0-3 [14.4 kB]
Get:73 http://deb.debian.org/debian stretch/main armhf python-cryptography armhf 1.7.1-3+deb9u2 [179 kB]
Get:74 http://deb.debian.org/debian stretch/main armhf python-dbus armhf 1.2.4-1+b1 [178 kB]
Get:75 http://deb.debian.org/debian stretch/main armhf python-gi armhf 3.22.0-2 [502 kB]
Get:76 http://deb.debian.org/debian stretch/main armhf python-webencodings all 0.5-2 [10.3 kB]
Get:77 http://deb.debian.org/debian stretch/main armhf python-html5lib all 0.999999999-1 [88.0 kB]
Get:78 http://deb.debian.org/debian stretch/main armhf python-secretstorage all 2.3.1-2 [13.8 kB]
Get:79 http://deb.debian.org/debian stretch/main armhf python-keyring all 10.1-1 [40.7 kB]
Get:80 http://deb.debian.org/debian stretch/main armhf python-keyrings.alt all 1.3-1 [16.4 kB]
Get:81 http://deb.debian.org/debian-security stretch/updates/main armhf python-lxml armhf 3.7.1-1+deb9u4 [829 kB]
Get:82 http://deb.debian.org/debian-security stretch/updates/main armhf python-pip-whl all 9.0.1-2+deb9u2 [1399 kB]
Get:83 http://deb.debian.org/debian-security stretch/updates/main armhf python-pip all 9.0.1-2+deb9u2 [179 kB]
Get:84 http://deb.debian.org/debian stretch/main armhf python-wheel all 0.29.0-2 [51.7 kB]
Get:85 http://deb.debian.org/debian stretch/main armhf python-xdg all 0.25-4 [35.8 kB]
Get:86 http://deb.debian.org/debian stretch/main armhf xml-core all 0.17 [23.2 kB]
Fetched 81.5 MB in 2min 53s (469 kB/s)
Extracting templates from packages: 100%
Selecting previously unselected package libpython2.7-minimal:armhf.
(Reading database ... 19480 files and directories currently installed.)
Preparing to unpack .../0-libpython2.7-minimal_2.7.13-2+deb9u4_armhf.deb ...
Unpacking libpython2.7-minimal:armhf (2.7.13-2+deb9u4) ...
Selecting previously unselected package python2.7-minimal.
Preparing to unpack .../1-python2.7-minimal_2.7.13-2+deb9u4_armhf.deb ...
Unpacking python2.7-minimal (2.7.13-2+deb9u4) ...
Selecting previously unselected package python-minimal.
Preparing to unpack .../2-python-minimal_2.7.13-2_armhf.deb ...
Unpacking python-minimal (2.7.13-2) ...
Selecting previously unselected package mime-support.
Preparing to unpack .../3-mime-support_3.60_all.deb ...
Unpacking mime-support (3.60) ...
Selecting previously unselected package libpython2.7-stdlib:armhf.
Preparing to unpack .../4-libpython2.7-stdlib_2.7.13-2+deb9u4_armhf.deb ...
Unpacking libpython2.7-stdlib:armhf (2.7.13-2+deb9u4) ...
Selecting previously unselected package python2.7.
Preparing to unpack .../5-python2.7_2.7.13-2+deb9u4_armhf.deb ...
Unpacking python2.7 (2.7.13-2+deb9u4) ...
Selecting previously unselected package libpython-stdlib:armhf.
Preparing to unpack .../6-libpython-stdlib_2.7.13-2_armhf.deb ...
Unpacking libpython-stdlib:armhf (2.7.13-2) ...
Setting up libpython2.7-minimal:armhf (2.7.13-2+deb9u4) ...
Setting up python2.7-minimal (2.7.13-2+deb9u4) ...
Linking and byte-compiling packages for runtime python2.7...
Setting up python-minimal (2.7.13-2) ...
Selecting previously unselected package python.
(Reading database ... 20253 files and directories currently installed.)
Preparing to unpack .../00-python_2.7.13-2_armhf.deb ...
```

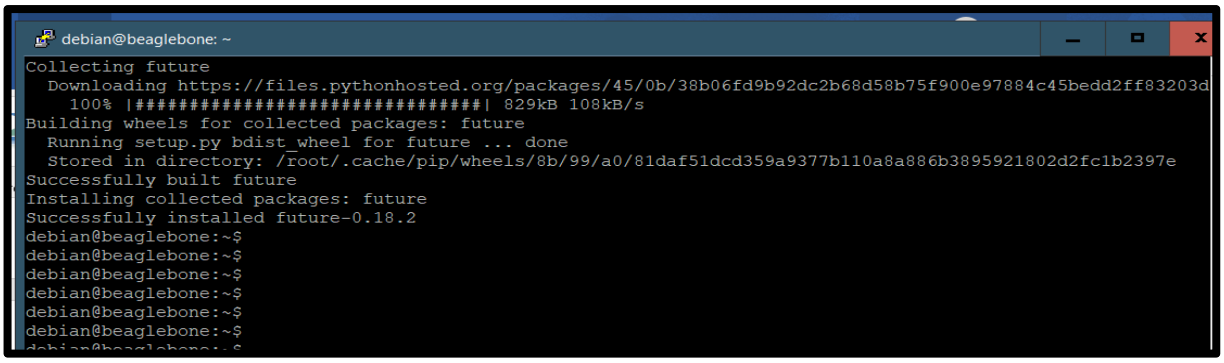
Figure 2.32: Installing pre-requisites for compiling Ardupilot application on Beagle Bone Blue itself



```
debian@beaglebone: ~
Setting up file (1:5.30-1+deb9u3) ...
Setting up fakeroot (1.21-3.1) ...
update-alternatives: using /usr/bin/fakeroot-sysv to provide /usr/bin/fakeroot (fakeroot) in auto mode
Setting up gcc-6 (6.3.0-18+deb9u1) ...
Setting up g++-6 (6.3.0-18+deb9u1) ...
Setting up libpython-stdlib:armhf (2.7.13-2) ...
Setting up libalgorithm-merge-perl (0.08-3) ...
Setting up dpkg-dev (1.18.25) ...
Setting up libalgorithm-diff-xs-perl (0.04-4+b2) ...
Setting up libpython2.7:armhf (2.7.13-2+deb9u4) ...
Setting up libexpat1-dev:armhf (2.2.0-2+deb9u3) ...
Setting up libpython2.7-dev:armhf (2.7.13-2+deb9u4) ...
Setting up python2.7-dev (2.7.13-2+deb9u4) ...
Setting up python (2.7.13-2) ...
Setting up python-xdg (0.25-4) ...
Setting up python-idna (2.2-1) ...
Setting up libpython-dev:armhf (2.7.13-2) ...
Setting up python-crypto (2.6.1-7) ...
Setting up python-dev (2.7.13-2) ...
Setting up python-pyasn1 (0.1.9-2) ...
Setting up python-wheel (0.29.0-2) ...
Setting up libpython-all-dev:armhf (2.7.13-2) ...
Setting up gcc (4:6.3.0-4) ...
Setting up python-pkg-resources (33.1.1-1) ...
Setting up python-keyrings.alt (1.3-1) ...
Setting up python-cffi-backend (1.9.1-2) ...
Setting up python-gi (3.22.0-2) ...
Setting up python-six (1.10.0-3) ...
Setting up python-bs4 (4.5.3-1) ...
Setting up python-enum34 (1.1.6-1) ...
Setting up python-lxml (3.7.1-1+deb9u4) ...
Setting up g++ (4:6.3.0-4) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up python-dbus (1.2.4-1+b1) ...
Remove stale byte-compiled files...
Setting up python-ipaddress (1.0.17-1) ...
Setting up python-pip (9.0.1-2+deb9u2) ...
Setting up python-all (2.7.13-2) ...
Setting up python-webencodings (0.5-2) ...
Setting up python-setuptools (33.1.1-1) ...
Setting up python-chardet (2.3.0-2) ...
Setting up build-essential (12.3) ...
Setting up python-all-dev (2.7.13-2) ...
Setting up python-cryptography (1.7.1-3+deb9u2) ...
Setting up python-html5lib (0.999999999-1) ...
Setting up python-secretstorage (2.3.1-2) ...
Setting up python-keyring (10.1-1) ...
Processing triggers for libc-bin (2.24-11+deb9u4) ...
Processing triggers for sgml-base (1.29) ...
debian@beaglebone:~$
```

Figure 2.33: Pre-requisites installed successfully

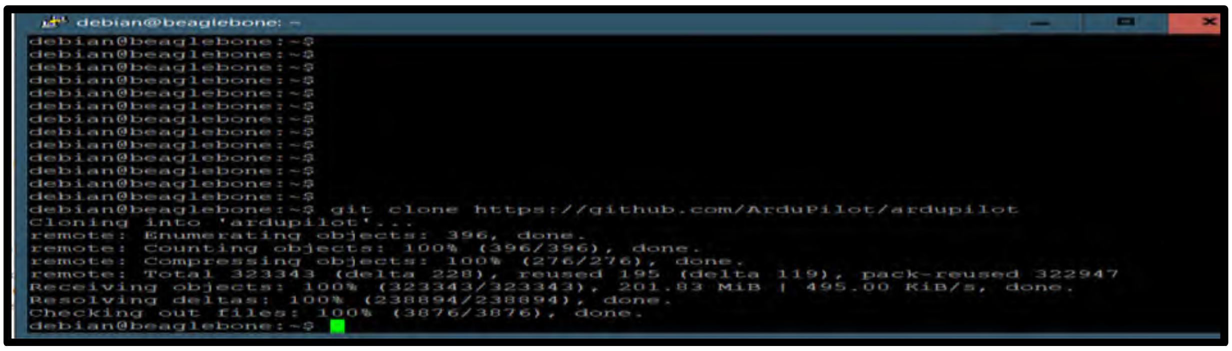
sudo pip install future



```
debian@beaglebone: ~
Collecting future
  Downloading https://files.pythonhosted.org/packages/45/0b/38b06fd9b92dc2b68d58b75f900e97884c45bedd2ff83203d
    100% |████████████████████████████████| 829kB 108kB/s
Building wheels for collected packages: future
  Running setup.py bdist_wheel for future ... done
  Stored in directory: /root/.cache/pip/wheels/8b/99/a0/81daf51dcd359a9377b110a8a886b3895921802d2fc1b2397e
Successfully built future
Installing collected packages: future
Successfully installed future-0.18.2
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
```

Figure 2.34: Installing Future

git clone https://github.com/ArduPilot/ardupilot

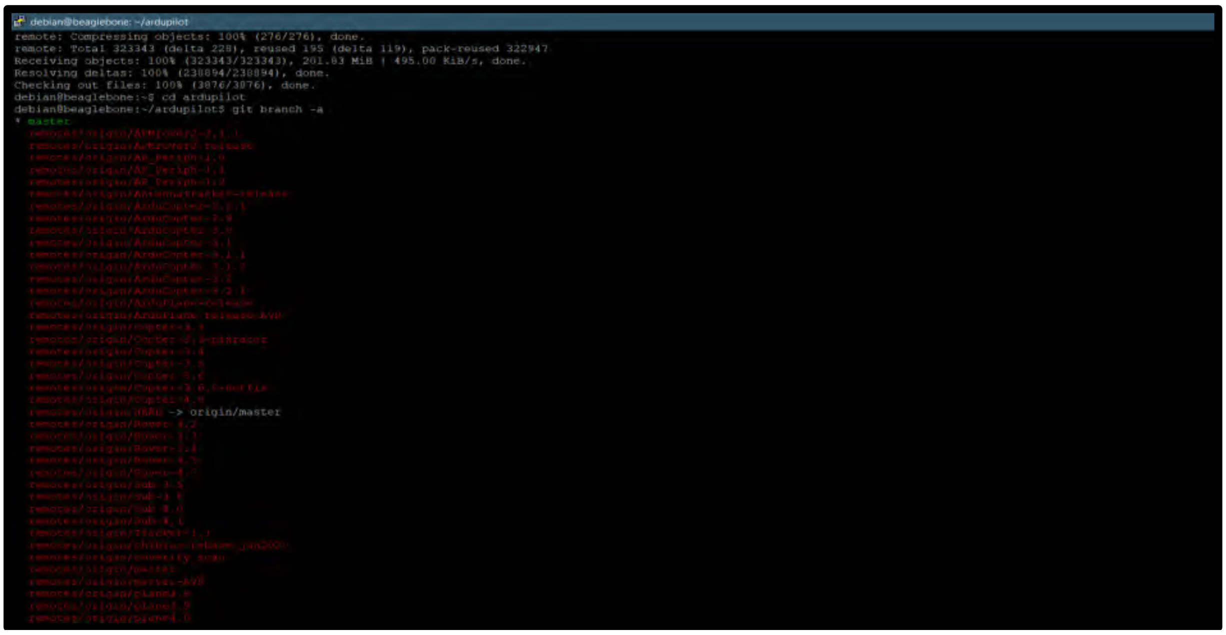


```
debian@beaglebone: ~
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$ git clone https://github.com/ArduPilot/ardupilot
Cloning into 'ardupilot'...
remote: Enumerating objects: 396, done.
remote: Counting objects: 100% (396/396), done.
remote: Compressing objects: 100% (276/276), done.
remote: Total 323343 (delta 228), reused 195 (delta 119), pack-reused 322947
Receiving objects: 100% (323343/323343), 201.83 MiB | 495.00 KiB/s, done.
Resolving deltas: 100% (238894/238894), done.
Checking out files: 100% (3876/3876), done.
debian@beaglebone:~$
```

Figure 2.35: Downloading Ardupilot repository from github successful

cd ardupilot

git branch -a # <-- See all available branches. Press q to exit.



```
debian@beaglebone: ~/ardupilot
remote: Compressing objects: 100% (276/276), done.
remote: Total 323343 (delta 228), reused 195 (delta 119), pack-reused 322947
Receiving objects: 100% (323343/323343), 201.83 MiB | 495.00 KiB/s, done.
Resolving deltas: 100% (238894/238894), done.
Checking out files: 100% (3876/3876), done.
debian@beaglebone:~$ cd ardupilot
debian@beaglebone:~/ardupilot$ git branch -a
* master
  remotes/origin/APMrover2-3.1.1
  remotes/origin/APMrover2-release
  remotes/origin/AP_Periph-1.0
  remotes/origin/AP_Periph-1.1
  remotes/origin/AP_Periph-1.2
  remotes/origin/AntennaTracker-release
  remotes/origin/ArduCopter-2.9.1
  remotes/origin/ArduCopter-2.9
  remotes/origin/ArduCopter-3.0
  remotes/origin/ArduCopter-3.1
  remotes/origin/ArduCopter-3.1.1
  remotes/origin/ArduCopter-3.1.2
  remotes/origin/ArduCopter-3.2
  remotes/origin/ArduCopter-3.2.1
  remotes/origin/ArduPlane-release
  remotes/origin/ArduPlane-release-AVR
  remotes/origin/Copter-3.3
  remotes/origin/Copter-3.3-pixracer
  remotes/origin/Copter-3.4
  remotes/origin/Copter-3.5
  remotes/origin/Copter-3.6
  remotes/origin/Copter-3.6.5-notify
  remotes/origin/Copter-4.0
  remotes/origin/HEAD -> origin/master
  remotes/origin/Rover-4.2
  remotes/origin/Rover-3.3
  remotes/origin/Rover-3.4
  remotes/origin/Rover-3.5
  remotes/origin/Rover-4.0
  remotes/origin/Sub-3.5
  remotes/origin/Sub-3.6
  remotes/origin/Sub-4.0
  remotes/origin/Sub-4.1
  remotes/origin/Tracker-1.1
  remotes/origin/chibios-rebase_jan2019
  remotes/origin/coverity_scan
  remotes/origin/master
  remotes/origin/master-AVR
  remotes/origin/plane3.8
  remotes/origin/plane3.9
  remotes/origin/plane4.0
```

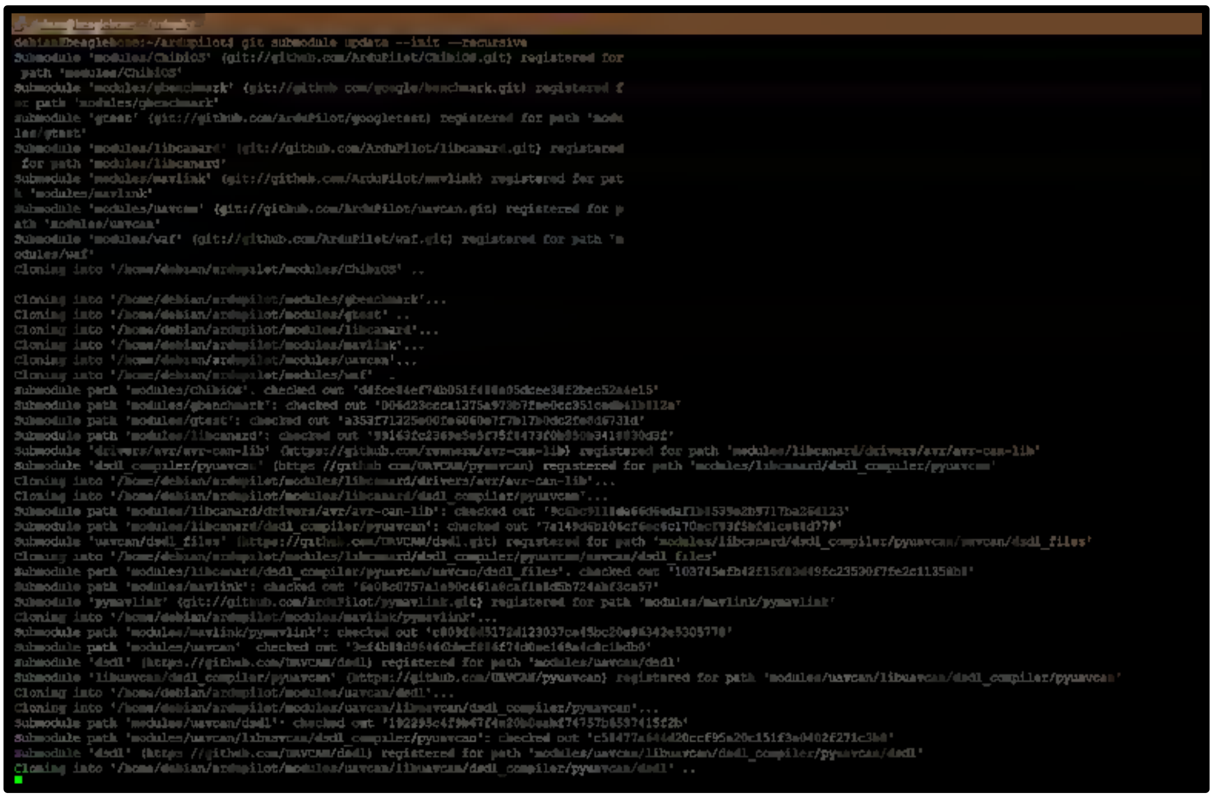
Figure 2.36: Selecting Ardupilot branch to compile. For me I have compiled latest 4.0.7 stable and also 4.1 (Beta)

I have compiled Ardupilot 3.6, 4.0.5, 4.0.7 all being stable versions. Later to support the mm Wave radar which is a 3D radar, I have compiled 4.1.0 (Beta) version. Hence, select the Ardupilot version one wishes to compile.

git checkout Copter-4.0 # <-- Arducopter 4.0 is selected by this command

Git submodule init to initialize our local configuration file, and git submodule is used to update to fetch all the data from Ardupilot github repository

git submodule update --init --recursive

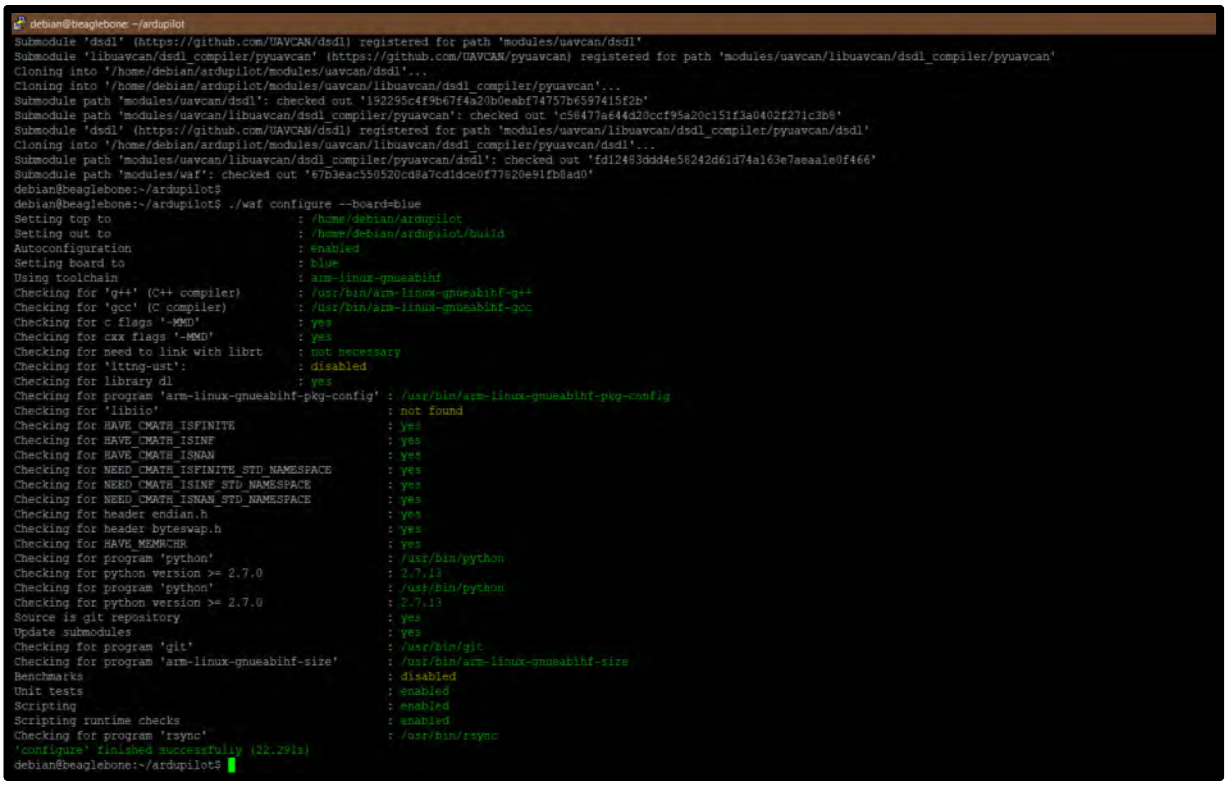


```
debian@beaglebone:~/ardupilot$ git submodule update --init --recursive
Submodule 'modules/ChibiOS' (git://github.com/ArduPilot/ChibiOS.git) registered for
 path 'modules/ChibiOS'
Submodule 'modules/gbenchmark' (git://github.com/google/benchmark.git) registered f
or path 'modules/gbenchmark'
Submodule 'gtest' (git://github.com/ArduPilot/googletest) registered for path 'modu
les/gtest'
Submodule 'modules/libcanard' (git://github.com/ArduPilot/libcanard.git) registered
 for path 'modules/libcanard'
Submodule 'modules/mavlink' (git://github.com/ArduPilot/mavlink) registered for pat
h 'modules/mavlink'
Submodule 'modules/uavcan' (git://github.com/ArduPilot/uavcan.git) registered for p
ath 'modules/uavcan'
Submodule 'modules/waf' (git://github.com/ArduPilot/waf.git) registered for path 'm
odules/waf'
Cloning into '/home/debian/ardupilot/modules/ChibiOS'...

Cloning into '/home/debian/ardupilot/modules/gbenchmark'...
Cloning into '/home/debian/ardupilot/modules/gtest'...
Cloning into '/home/debian/ardupilot/modules/libcanard'...
Cloning into '/home/debian/ardupilot/modules/mavlink'...
Cloning into '/home/debian/ardupilot/modules/uavcan'...
Cloning into '/home/debian/ardupilot/modules/waf'...
Submodule path 'modules/ChibiOS': checked out 'ddfce84ef74b051f488a05dcee38f2bec52a4e15'
Submodule path 'modules/gbenchmark': checked out '004d23ccca1375a973b7fae0cc351ca8b41b812a'
Submodule path 'modules/gtest': checked out 'a353f71325e00fe6060e7f7b17b0dc2fe8d6731d'
Submodule path 'modules/libcanard': checked out '99163fc2369e5e5f75f8473f0b850b3418030d3f'
Submodule 'drivers/avr/avr-can-lib' (https://github.com/rennerm/avr-can-lib) registered for path 'modules/libcanard/drivers/avr/avr-can-lib'
Submodule 'dsdl_compiler/pyuavcan' (https://github.com/UAVCAN/pyuavcan) registered for path 'modules/libcanard/dsdl_compiler/pyuavcan'
Cloning into '/home/debian/ardupilot/modules/libcanard/drivers/avr/avr-can-lib'...
Cloning into '/home/debian/ardupilot/modules/libcanard/dsdl_compiler/pyuavcan'...
Submodule path 'modules/libcanard/drivers/avr/avr-can-lib': checked out '9c8bc9118da66d6edaf1b4539e2b9717ba26d123'
Submodule path 'modules/libcanard/dsdl_compiler/pyuavcan': checked out '7a149d6b106cf6ec6c170acf93f5bfd1ce88d779'
Submodule 'uavcan/dsdl_files' (https://github.com/UAVCAN/dsdl.git) registered for path 'modules/libcanard/dsdl_compiler/pyuavcan/uavcan/dsdl_files'
Cloning into '/home/debian/ardupilot/modules/libcanard/dsdl_compiler/pyuavcan/uavcan/dsdl_files'
Submodule path 'modules/libcanard/dsdl_compiler/pyuavcan/uavcan/dsdl_files': checked out '103745afb42f15f03d49fc23530f7fe2c11358b8'
Submodule path 'modules/mavlink': checked out '6e08c0757a1a90c461a8caf1a8d5b724abf3ca57'
Submodule 'pymavlink' (git://github.com/ArduPilot/pymavlink.git) registered for path 'modules/mavlink/pymavlink'
Cloning into '/home/debian/ardupilot/modules/mavlink/pymavlink'...
Submodule path 'modules/mavlink/pymavlink': checked out 'c809f0d5172d123037ca45bc20e96342e5305778'
Submodule path 'modules/uavcan': checked out '3ef4b88d9646dbecf886f74d0ae169a4c8c1bdb0'
Submodule 'dsdl' (https://github.com/UAVCAN/dsdl) registered for path 'modules/uavcan/dsdl'
Submodule 'libuavcan/dsdl_compiler/pyuavcan' (https://github.com/UAVCAN/pyuavcan) registered for path 'modules/uavcan/libuavcan/dsdl_compiler/pyuavcan'
Cloning into '/home/debian/ardupilot/modules/uavcan/dsdl'...
Cloning into '/home/debian/ardupilot/modules/uavcan/libuavcan/dsdl_compiler/pyuavcan'...
Submodule path 'modules/uavcan/dsdl': checked out '192295c4f9b67f4e20b0eabf74757b6597415f2b'
Submodule path 'modules/uavcan/libuavcan/dsdl_compiler/pyuavcan': checked out 'c58477a644d20ccf95a20c151f3e0402f271c3b8'
Submodule 'dsdl' (https://github.com/UAVCAN/dsdl) registered for path 'modules/uavcan/libuavcan/dsdl_compiler/pyuavcan/dsdl'
Cloning into '/home/debian/ardupilot/modules/uavcan/libuavcan/dsdl_compiler/pyuavcan/dsdl'...
```

Figure 2.37: Updating submodule of Ardupilot

Now we use the waf command which is a build automation tool. It is designed to assist in the automatic compilation and installation of application/computer software. Now we name the compilation folder Blue as its Beagle Bone Blue.

./waf configure --board=blue # <-- BeagleBone Blue.

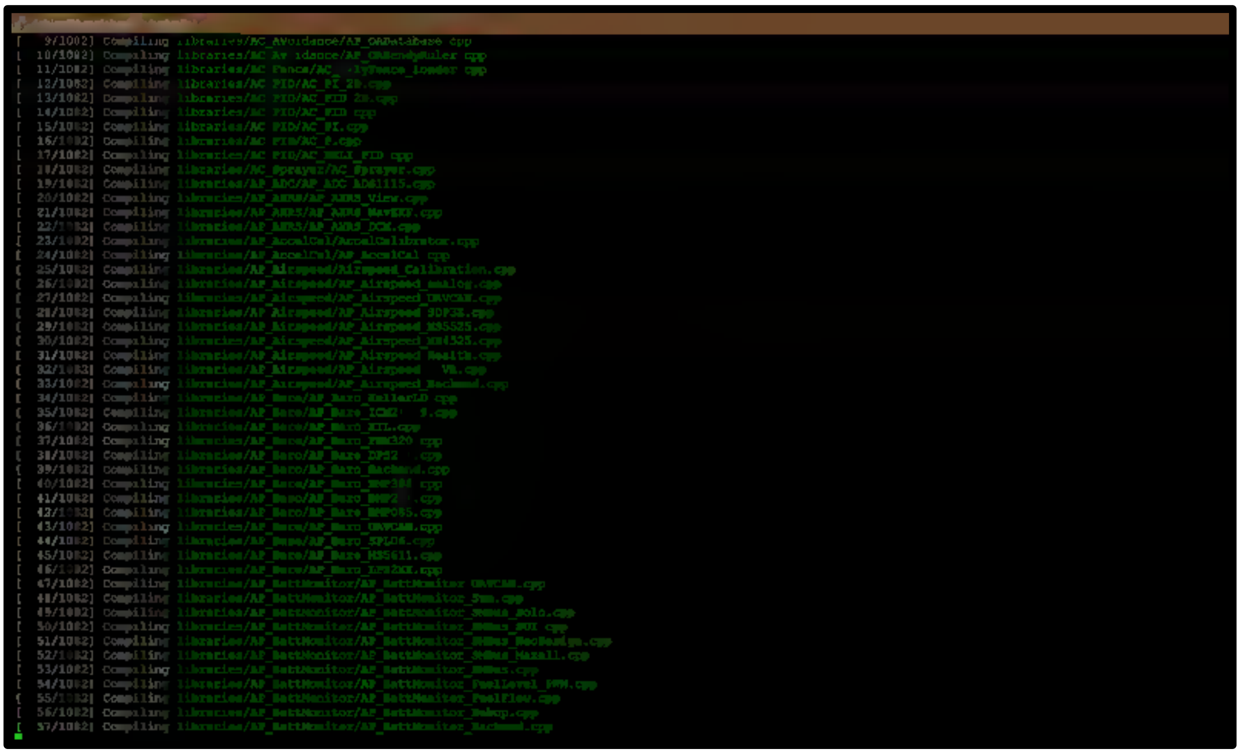


```
debian@beaglebone: ~/ardupilot
Submodule 'dsdl' (https://github.com/UAVCAN/dsdl) registered for path 'modules/uavcan/dsdl'
Submodule 'libuavcan/dsdl_compiler/pyuavcan' (https://github.com/UAVCAN/pyuavcan) registered for path 'modules/uavcan/libuavcan/dsdl_compiler/pyuavcan'
Cloning into '/home/debian/ardupilot/modules/uavcan/dsdl'...
Cloning into '/home/debian/ardupilot/modules/uavcan/libuavcan/dsdl_compiler/pyuavcan'...
Submodule path 'modules/uavcan/dsdl': checked out '192295c4f9b67f4a20b0eabf74757b6597415f2b'
Submodule path 'modules/uavcan/libuavcan/dsdl_compiler/pyuavcan': checked out 'c58477a644d20ccf95a20c151f3a0402f271c3b8'
Submodule 'dsdl' (https://github.com/UAVCAN/dsdl) registered for path 'modules/uavcan/libuavcan/dsdl_compiler/pyuavcan/dsdl'
Cloning into '/home/debian/ardupilot/modules/uavcan/libuavcan/dsdl_compiler/pyuavcan/dsdl'...
Submodule path 'modules/uavcan/libuavcan/dsdl_compiler/pyuavcan/dsdl': checked out 'fd12483ddd4e58242d61d74a163e7aeaa1e0f466'
Submodule path 'modules/waf': checked out '67b3eac550520cd8a7cd1dce0f77820e91fb8ad0'
debian@beaglebone:~/ardupilot$
debian@beaglebone:~/ardupilot$ ./waf configure --board=blue
Setting top to                           : /home/debian/ardupilot
Setting out to                           : /home/debian/ardupilot/build
Autoconfiguration                        : enabled
Setting board to                         : blue
Using toolchain                          : arm-linux-gnueabihf
Checking for 'g++' (C++ compiler)        : /usr/bin/arm-linux-gnueabihf-g++
Checking for 'gcc' (C compiler)          : /usr/bin/arm-linux-gnueabihf-gcc
Checking for c flags '-MMD'              : yes
Checking for cxx flags '-MMD'            : yes
Checking for need to link with librt     : not necessary
Checking for 'lttng-ust':                : disabled
Checking for library dl                  : yes
Checking for program 'arm-linux-gnueabihf-pkg-config' : /usr/bin/arm-linux-gnueabihf-pkg-config
Checking for 'libiio'                                 : not found
Checking for HAVE_CMATH_ISFINITE                      : yes
Checking for HAVE_CMATH_ISINF                         : yes
Checking for HAVE_CMATH_ISNAN                         : yes
Checking for NEED_CMATH_ISFINITE_STD_NAMESPACE        : yes
Checking for NEED_CMATH_ISINF_STD_NAMESPACE           : yes
Checking for NEED_CMATH_ISNAN_STD_NAMESPACE           : yes
Checking for header endian.h                          : yes
Checking for header byteswap.h                        : yes
Checking for HAVE_MEMRCHR                             : yes
Checking for program 'python'                         : /usr/bin/python
Checking for python version >= 2.7.0                  : 2.7.13
Checking for program 'python'                         : /usr/bin/python
Checking for python version >= 2.7.0                  : 2.7.13
Source is git repository                              : yes
Update submodules                                     : yes
Checking for program 'git'                            : /usr/bin/git
Checking for program 'arm-linux-gnueabihf-size'       : /usr/bin/arm-linux-gnueabihf-size
Benchmarks                                            : disabled
Unit tests                                            : enabled
Scripting                                             : enabled
Scripting runtime checks                              : enabled
Checking for program 'rsync'                          : /usr/bin/rsync
'configure' finished successfully (22.291s)
debian@beaglebone:~/ardupilot$
```

Figure 2.38: The compilation will happen in folder named blue

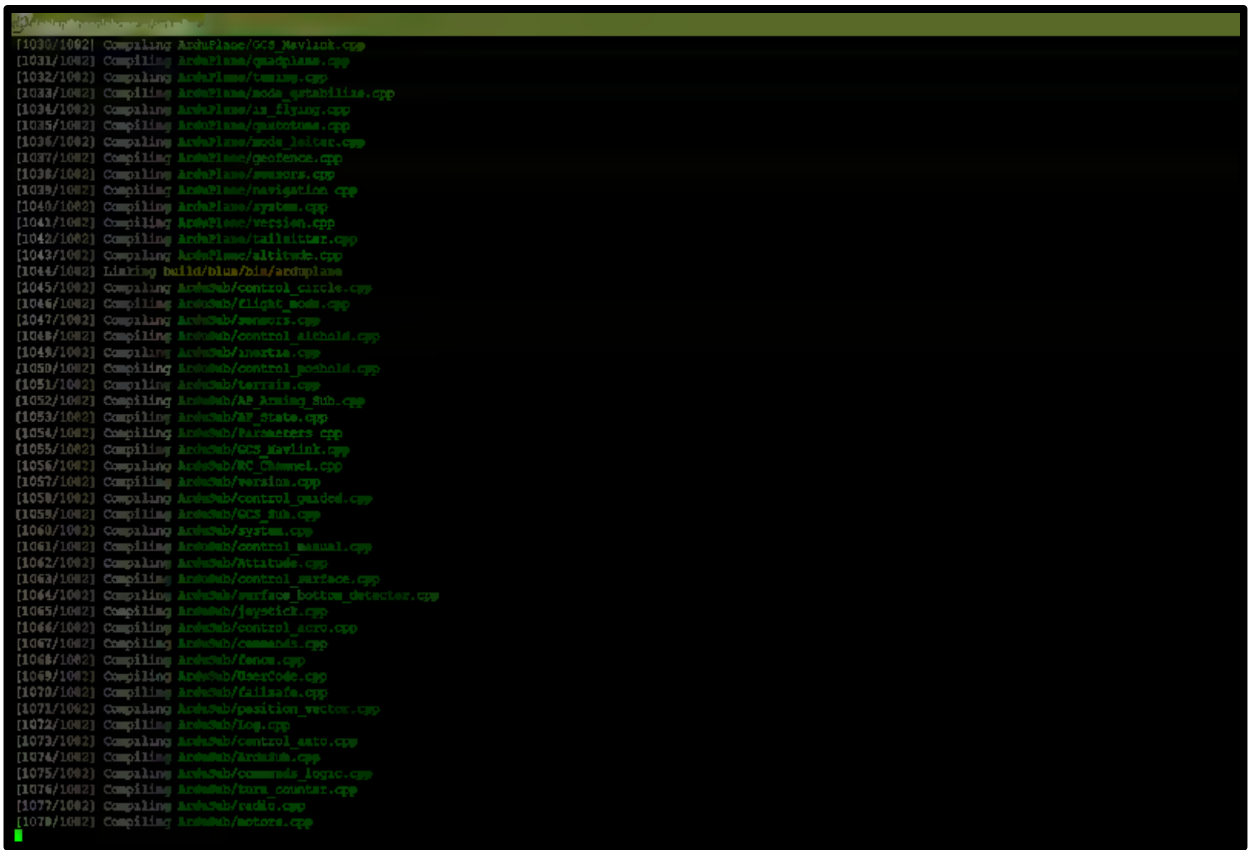
The compilation starts from the step below:-

./waf



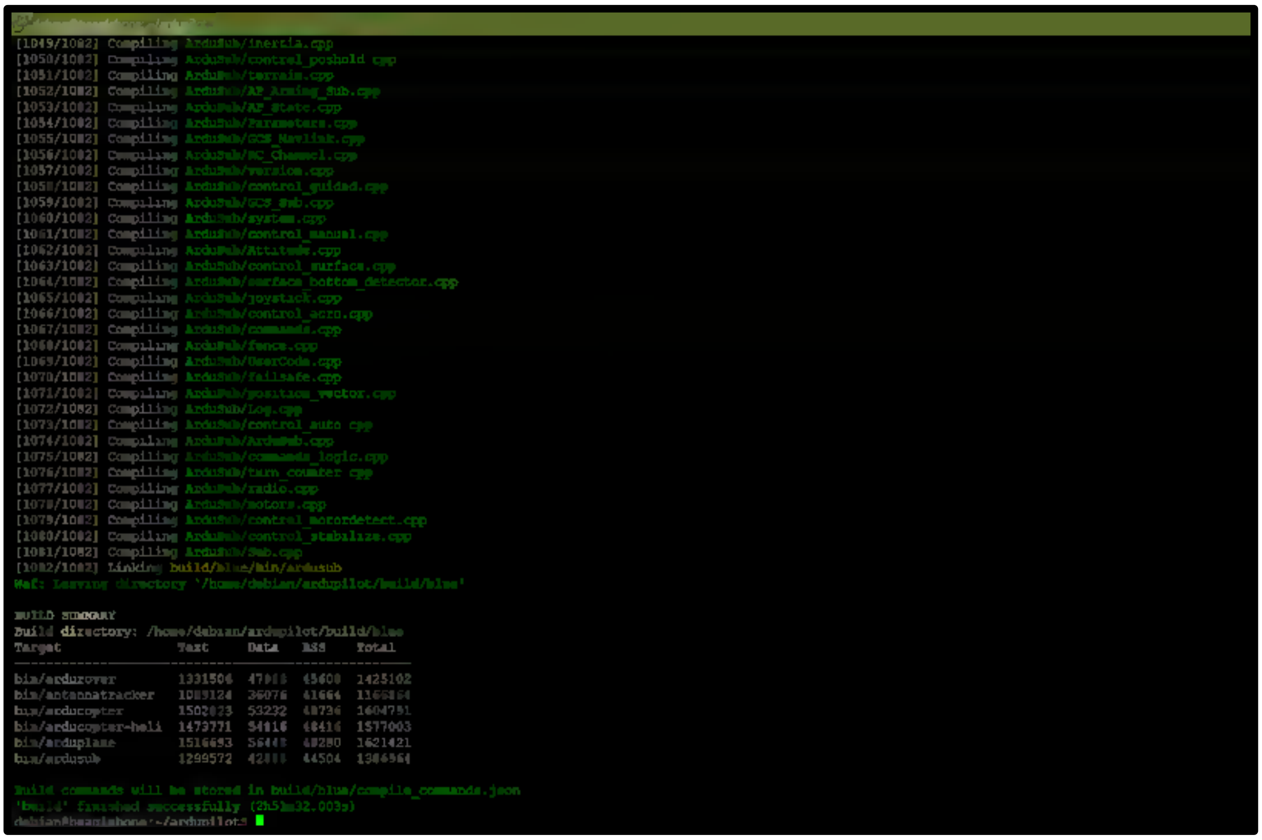
```
[   9/1082] Compiling libraries/AC_Avoidance/AP_OADatabase.cpp
[  10/1082] Compiling libraries/AC_Avoidance/AP_OABendyRuler.cpp
[  11/1082] Compiling libraries/AC_Fence/AC_PolyFence_loader.cpp
[  12/1082] Compiling libraries/AC_PID/AC_PI_2D.cpp
[  13/1082] Compiling libraries/AC_PID/AC_PID_2D.cpp
[  14/1082] Compiling libraries/AC_PID/AC_PID.cpp
[  15/1082] Compiling libraries/AC_PID/AC_PI.cpp
[  16/1082] Compiling libraries/AC_PID/AC_P.cpp
[  17/1082] Compiling libraries/AC_PID/AC_HELI_PID.cpp
[  18/1082] Compiling libraries/AC_Sprayer/AC_Sprayer.cpp
[  19/1082] Compiling libraries/AP_ADC/AP_ADC_ADS1115.cpp
[  20/1082] Compiling libraries/AP_AHRS/AP_AHRS_View.cpp
[  21/1082] Compiling libraries/AP_AHRS/AP_AHRS_NavEKF.cpp
[  22/1082] Compiling libraries/AP_AHRS/AP_AHRS_DCM.cpp
[  23/1082] Compiling libraries/AP_AccelCal/AccelCalibrator.cpp
[  24/1082] Compiling libraries/AP_AccelCal/AP_AccelCal.cpp
[  25/1082] Compiling libraries/AP_Airspeed/Airspeed_Calibration.cpp
[  26/1082] Compiling libraries/AP_Airspeed/AP_Airspeed_analog.cpp
[  27/1082] Compiling libraries/AP_Airspeed/AP_Airspeed_UAVCAN.cpp
[  28/1082] Compiling libraries/AP_Airspeed/AP_Airspeed_SDP3X.cpp
[  29/1082] Compiling libraries/AP_Airspeed/AP_Airspeed_MS5525.cpp
[  30/1082] Compiling libraries/AP_Airspeed/AP_Airspeed_MS4525.cpp
[  31/1082] Compiling libraries/AP_Airspeed/AP_Airspeed_Health.cpp
[  32/1082] Compiling libraries/AP_Airspeed/AP_Airspeed_DLVR.cpp
[  33/1082] Compiling libraries/AP_Airspeed/AP_Airspeed_Backend.cpp
[  34/1082] Compiling libraries/AP_Baro/AP_Baro_KellerLD.cpp
[  35/1082] Compiling libraries/AP_Baro/AP_Baro_ICM20789.cpp
[  36/1082] Compiling libraries/AP_Baro/AP_Baro_HIL.cpp
[  37/1082] Compiling libraries/AP_Baro/AP_Baro_FBM320.cpp
[  38/1082] Compiling libraries/AP_Baro/AP_Baro_DPS280.cpp
[  39/1082] Compiling libraries/AP_Baro/AP_Baro_Backend.cpp
[  40/1082] Compiling libraries/AP_Baro/AP_Baro_BMP388.cpp
[  41/1082] Compiling libraries/AP_Baro/AP_Baro_BMP280.cpp
[  42/1082] Compiling libraries/AP_Baro/AP_Baro_BMP085.cpp
[  43/1082] Compiling libraries/AP_Baro/AP_Baro_UAVCAN.cpp
[  44/1082] Compiling libraries/AP_Baro/AP_Baro_SPL06.cpp
[  45/1082] Compiling libraries/AP_Baro/AP_Baro_MS5611.cpp
[  46/1082] Compiling libraries/AP_Baro/AP_Baro_LPS2XH.cpp
[  47/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_UAVCAN.cpp
[  48/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_Sum.cpp
[  49/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_SMBus_Solo.cpp
[  50/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_SMBus_SUI.cpp
[  51/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_SMBus_NeoDesign.cpp
[  52/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_SMBus_Maxell.cpp
[  53/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_SMBus.cpp
[  54/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_FuelLevel_PWM.cpp
[  55/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_FuelFlow.cpp
[  56/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_Bebop.cpp
[  57/1082] Compiling libraries/AP_BattMonitor/AP_BattMonitor_Backend.cpp
```

Figure 2.39: The compilation process is on



```
[1030/1082] Compiling ArduPlane/GCS_Mavlink.cpp
[1031/1082] Compiling ArduPlane/quadplane.cpp
[1032/1082] Compiling ArduPlane/tuning.cpp
[1033/1082] Compiling ArduPlane/mode_qstabilize.cpp
[1034/1082] Compiling ArduPlane/is_flying.cpp
[1035/1082] Compiling ArduPlane/qautotune.cpp
[1036/1082] Compiling ArduPlane/mode_loiter.cpp
[1037/1082] Compiling ArduPlane/geofence.cpp
[1038/1082] Compiling ArduPlane/sensors.cpp
[1039/1082] Compiling ArduPlane/navigation.cpp
[1040/1082] Compiling ArduPlane/system.cpp
[1041/1082] Compiling ArduPlane/version.cpp
[1042/1082] Compiling ArduPlane/tailsitter.cpp
[1043/1082] Compiling ArduPlane/altitude.cpp
[1044/1082] Linking build/blue/bin/arduplane
[1045/1082] Compiling ArduSub/control_circle.cpp
[1046/1082] Compiling ArduSub/flight_mode.cpp
[1047/1082] Compiling ArduSub/sensors.cpp
[1048/1082] Compiling ArduSub/control_althold.cpp
[1049/1082] Compiling ArduSub/inertia.cpp
[1050/1082] Compiling ArduSub/control_poshold.cpp
[1051/1082] Compiling ArduSub/terrain.cpp
[1052/1082] Compiling ArduSub/AP_Arming_Sub.cpp
[1053/1082] Compiling ArduSub/AP_State.cpp
[1054/1082] Compiling ArduSub/Parameters.cpp
[1055/1082] Compiling ArduSub/GCS_Mavlink.cpp
[1056/1082] Compiling ArduSub/RC_Channel.cpp
[1057/1082] Compiling ArduSub/version.cpp
[1058/1082] Compiling ArduSub/control_guided.cpp
[1059/1082] Compiling ArduSub/GCS_Sub.cpp
[1060/1082] Compiling ArduSub/system.cpp
[1061/1082] Compiling ArduSub/control_manual.cpp
[1062/1082] Compiling ArduSub/Attitude.cpp
[1063/1082] Compiling ArduSub/control_surface.cpp
[1064/1082] Compiling ArduSub/surface_bottom_detector.cpp
[1065/1082] Compiling ArduSub/joystick.cpp
[1066/1082] Compiling ArduSub/control_acro.cpp
[1067/1082] Compiling ArduSub/commands.cpp
[1068/1082] Compiling ArduSub/fence.cpp
[1069/1082] Compiling ArduSub/UserCode.cpp
[1070/1082] Compiling ArduSub/failsafe.cpp
[1071/1082] Compiling ArduSub/position_vector.cpp
[1072/1082] Compiling ArduSub/Log.cpp
[1073/1082] Compiling ArduSub/control_auto.cpp
[1074/1082] Compiling ArduSub/ArduSub.cpp
[1075/1082] Compiling ArduSub/commands_logic.cpp
[1076/1082] Compiling ArduSub/turn_counter.cpp
[1077/1082] Compiling ArduSub/radio.cpp
[1078/1082] Compiling ArduSub/motors.cpp
█
```

Figure 2.40: The compilation process is on



```
[1049/1082] Compiling ArduSub/inertia.cpp
[1050/1082] Compiling ArduSub/control_poshold.cpp
[1051/1082] Compiling ArduSub/terrain.cpp
[1052/1082] Compiling ArduSub/AP_Arming_Sub.cpp
[1053/1082] Compiling ArduSub/AP_State.cpp
[1054/1082] Compiling ArduSub/Parameters.cpp
[1055/1082] Compiling ArduSub/GCS_Mavlink.cpp
[1056/1082] Compiling ArduSub/RC_Channel.cpp
[1057/1082] Compiling ArduSub/version.cpp
[1058/1082] Compiling ArduSub/control_guided.cpp
[1059/1082] Compiling ArduSub/GCS_Sub.cpp
[1060/1082] Compiling ArduSub/system.cpp
[1061/1082] Compiling ArduSub/control_manual.cpp
[1062/1082] Compiling ArduSub/Attitude.cpp
[1063/1082] Compiling ArduSub/control_surface.cpp
[1064/1082] Compiling ArduSub/surface_bottom_detector.cpp
[1065/1082] Compiling ArduSub/joystick.cpp
[1066/1082] Compiling ArduSub/control_acro.cpp
[1067/1082] Compiling ArduSub/commands.cpp
[1068/1082] Compiling ArduSub/fence.cpp
[1069/1082] Compiling ArduSub/UserCode.cpp
[1070/1082] Compiling ArduSub/failsafe.cpp
[1071/1082] Compiling ArduSub/position_vector.cpp
[1072/1082] Compiling ArduSub/Log.cpp
[1073/1082] Compiling ArduSub/control_auto.cpp
[1074/1082] Compiling ArduSub/ArduSub.cpp
[1075/1082] Compiling ArduSub/commands_logic.cpp
[1076/1082] Compiling ArduSub/turn_counter.cpp
[1077/1082] Compiling ArduSub/radio.cpp
[1078/1082] Compiling ArduSub/motors.cpp
[1079/1082] Compiling ArduSub/control_motordetect.cpp
[1080/1082] Compiling ArduSub/control_stabilize.cpp
[1081/1082] Compiling ArduSub/Sub.cpp
[1082/1082] Linking build/blue/bin/ardusub
Waf: Leaving directory `/home/debian/ardupilot/build/blue'

BUILD SUMMARY
Build directory: /home/debian/ardupilot/build/blue
Target                  Text     Data    BSS     Total
--------------------------------------------------------
bin/ardurover           1331506  47[illegible]  45600   1425102
bin/antennatracker      10[illegible]124  36076   41664   1166[illegible]
bin/arducopter          1502023  53232   48736   1604791
bin/arducopter-heli     1473771  54816   48416   1577003
bin/arduplane           1516693  56[illegible]  48280   1621421
bin/ardusub             1299572  42[illegible]  44504   1386964

Build commands will be stored in build/blue/compile_commands.json
'build' finished successfully (2h53m32.003s)
debian@beaglebone:~/ardupilot$ █
```

Figure 2.41: The compilation process is successfully completed

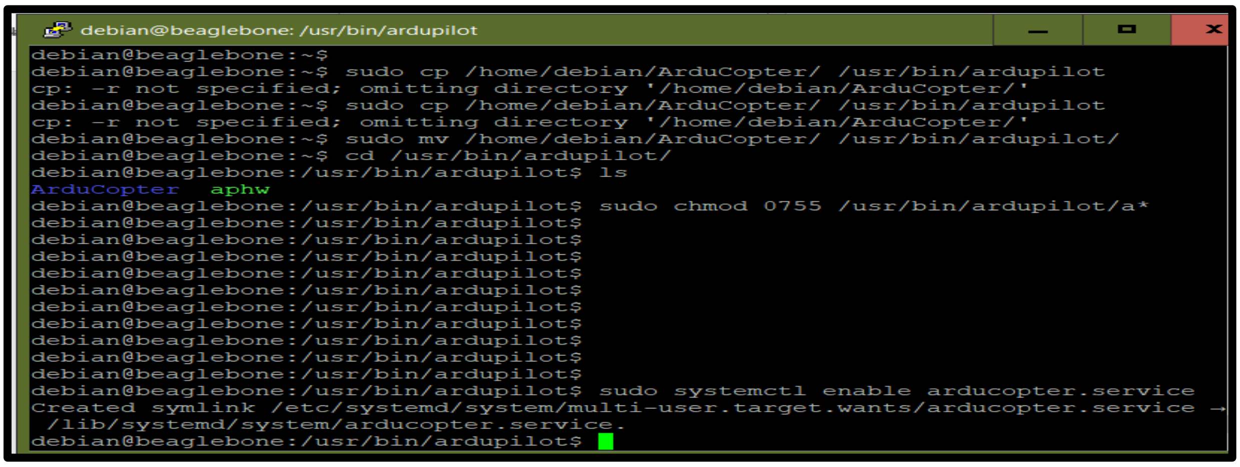
The compilation for Arducopter 3.6 took about 2 hours, while for Arducopter 4.0.5, (stable) Arducopter 4.0.7 (stable) and Arducopter 4.1.0 (Beta) took almost 12 hours !!!! Now we copy the compiled bin file to the execution folder and give permission.

sudo cp ./build/blue/bin/a* /usr/bin/ardupilot

sudo chmod 0755 /usr/bin/ardupilot/a*

Step 8. Now we enable Ardùcopter service after successfully compiling it on Beagle Bone Blue itself.

sudo systemctl enable arducopter.service



```
debian@beaglebone:~$
debian@beaglebone:~$ sudo cp /home/debian/ArduCopter/ /usr/bin/ardupilot
cp: -r not specified; omitting directory '/home/debian/ArduCopter/'
debian@beaglebone:~$ sudo cp /home/debian/ArduCopter/ /usr/bin/ardupilot
cp: -r not specified; omitting directory '/home/debian/ArduCopter/'
debian@beaglebone:~$ sudo mv /home/debian/ArduCopter/ /usr/bin/ardupilot/
debian@beaglebone:~$ cd /usr/bin/ardupilot/
debian@beaglebone:/usr/bin/ardupilot$ ls
ArduCopter  aphw
debian@beaglebone:/usr/bin/ardupilot$ sudo chmod 0755 /usr/bin/ardupilot/a*
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$
debian@beaglebone:/usr/bin/ardupilot$ sudo systemctl enable arducopter.service
Created symlink /etc/systemd/system/multi-user.target.wants/arducopter.service →
 /lib/systemd/system/arducopter.service.
debian@beaglebone:/usr/bin/ardupilot$
```

Figure 2.42: The arducopter.service enabled successfully

To start Arducopter service,

sudo systemctl start arducopter.service

To check arducopter service status

sudo systemctl status arducopter.service

To disable and stop arducopter service

sudo systemctl disable arducopter.service

sudo systemctl stop arducopter.service

To restart service after making any change will require following command,

sudo systemctl daemon-reload

sudo systemctl restart arducopter. service

CHAPTER 3

SENSORS AND COMPONENTS

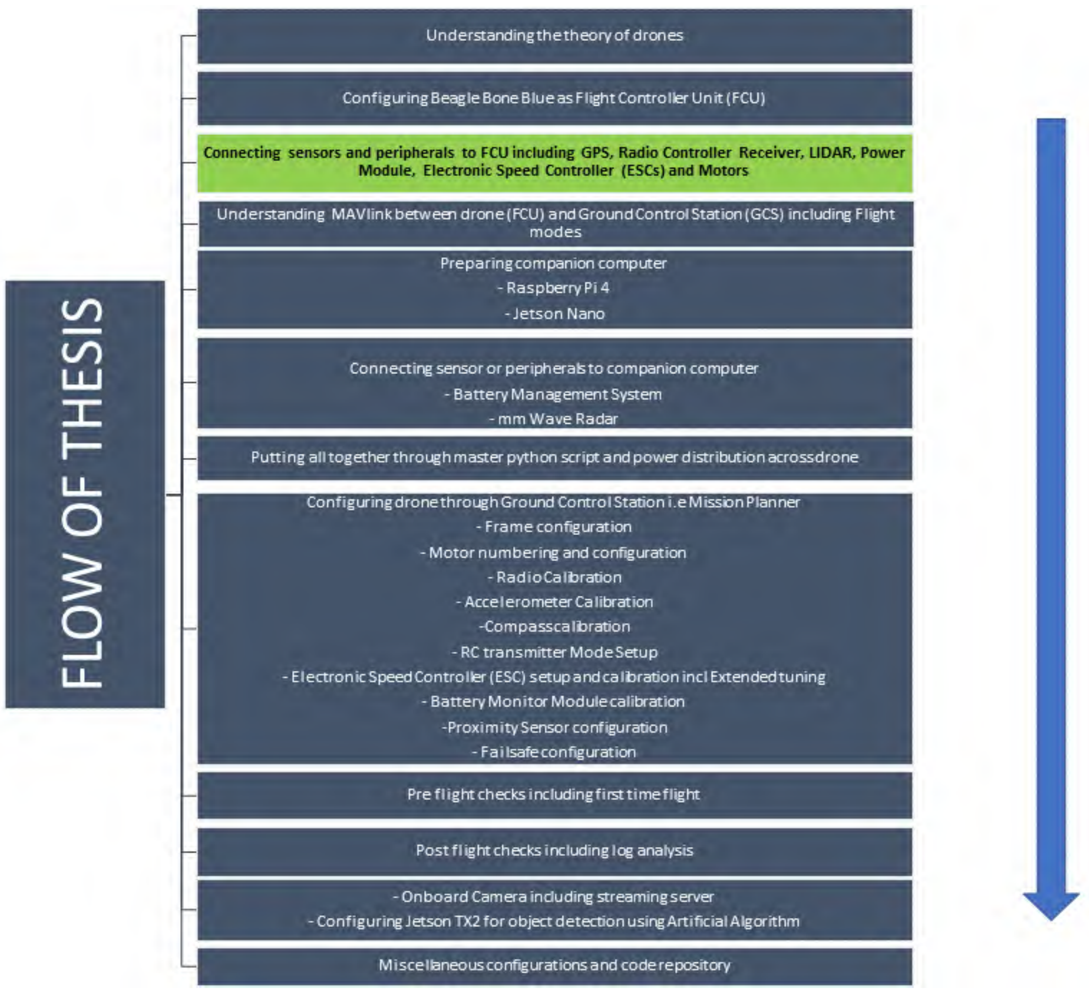

FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server
- Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

Figure 3.1: Flow of thesis until now

In the previous chapter I have worked on the Beagle Bone Blue as Flight Controller Unit (FCU) which is the brain and the most critical part of drone design. All sensors and components work around the choice of FCU, type of frame, type of mission and the overall budget. In this chapter I have focused on the sensors connected directly to FCU namely:

- Integration of GPS with compass with FCU
- Radio Controller (RC) Receiver integration with FCU
- LIDAR integration with FCU
- Power Module integration with FCU

Ardupilot autopilot application that we successfully compiled on Linux board Beagle Bone Blue supports a wide variety of sensors from many different manufacturers. Being an open source application, it is supported by developers all across the world and its scope of sensor integration is ever increasing. The application supports multiple communication protocols such as I2C, SPI, UART (or Serial) and CANBUS (in particular UAVCAN) protocols. Beagle Bone Blue provides hardware support for all these communication protocols. However, in the course of making our drones, I have utilized UART and I2C communication protocols only.

A brief introduction to all the supported protocols is as under:

I2C

- one master, many slaves possible
- a relatively simple protocol which is good for communicating over short-distances (i.e. less than 1m).
- bus runs at 100kHz or 400kHz but the data rate is relatively low compared to other protocols.
- only 3 pins are required (GND, SDA, SCL). VCC is only to power the sensor. [3]

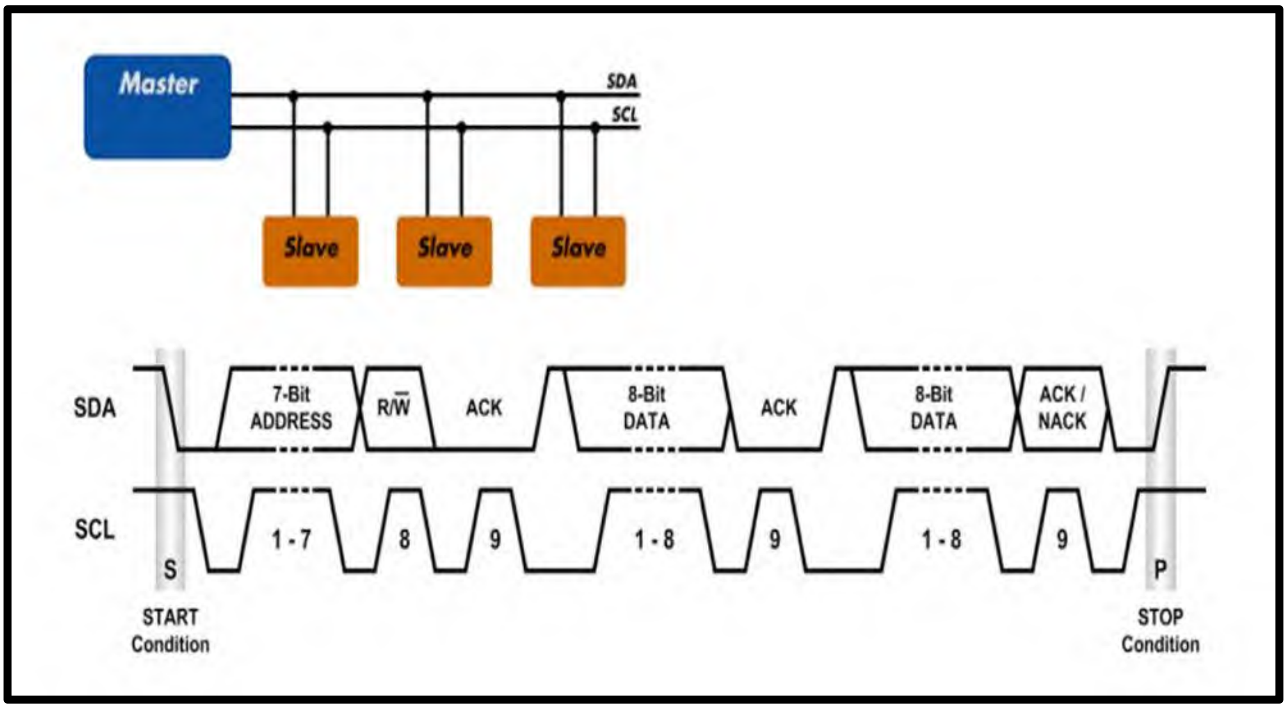


Figure 3.2: I2C Protocol [3]

SPI

- works in one master, one slave configuration
- The clock speed of 20Mhz+ speed much faster than I2C
- short distances protocol upto 10cm max
- requires at least 4 pins namely GND, SCLK, Master-Out-Slave-In, Master-In-Slave-Out + 1 slave select pin per slave. VCC is only to power the sensor. [3]

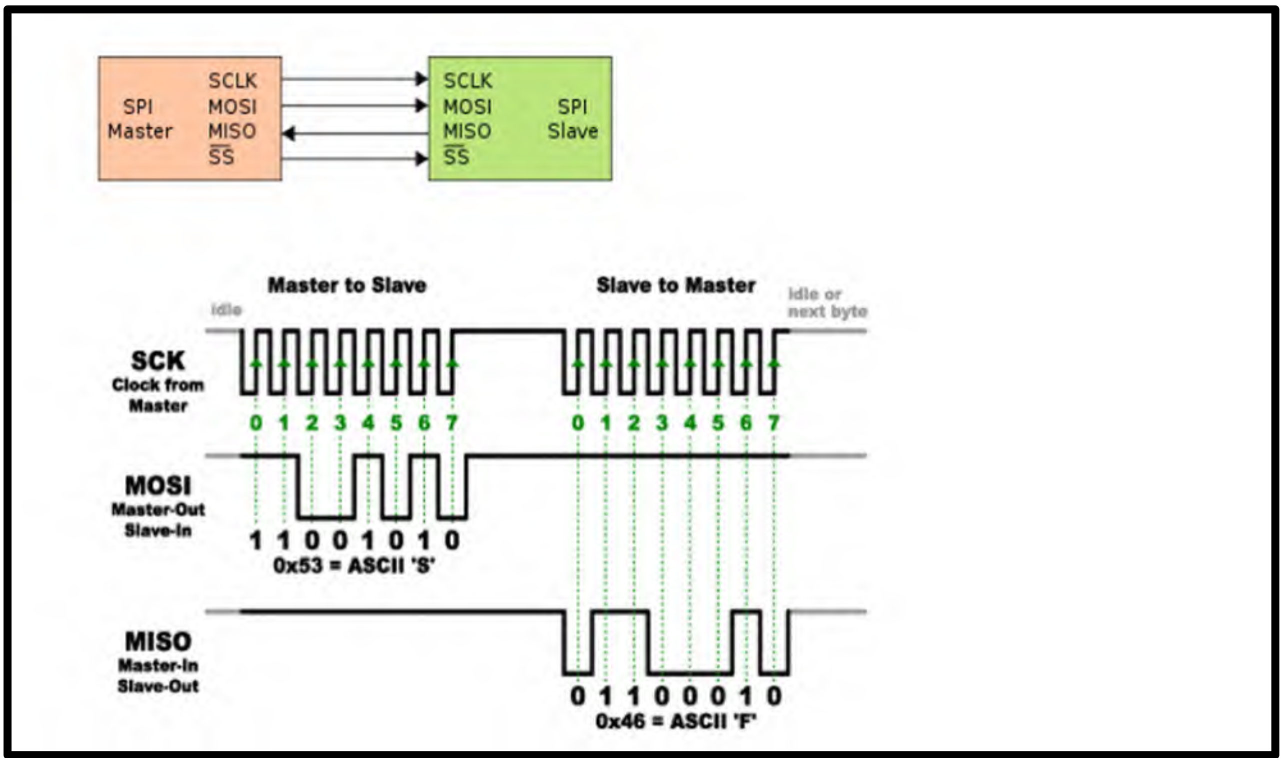


Figure 3.3: SPI protocol [3]

Serial / UART

- works in one master, one slave configuration
- being a character based protocol. Better than I2C and SPI for long distance communication (i.e. 1m)
- it is relatively fast at 57Kbps ~ 1.5Mbps
- it requires at least 3 pins (GND, TX, RX) Two pins (Clear-To-Send, Clear-To-Receive) are optional. VCC is only to power the sensor. [3]

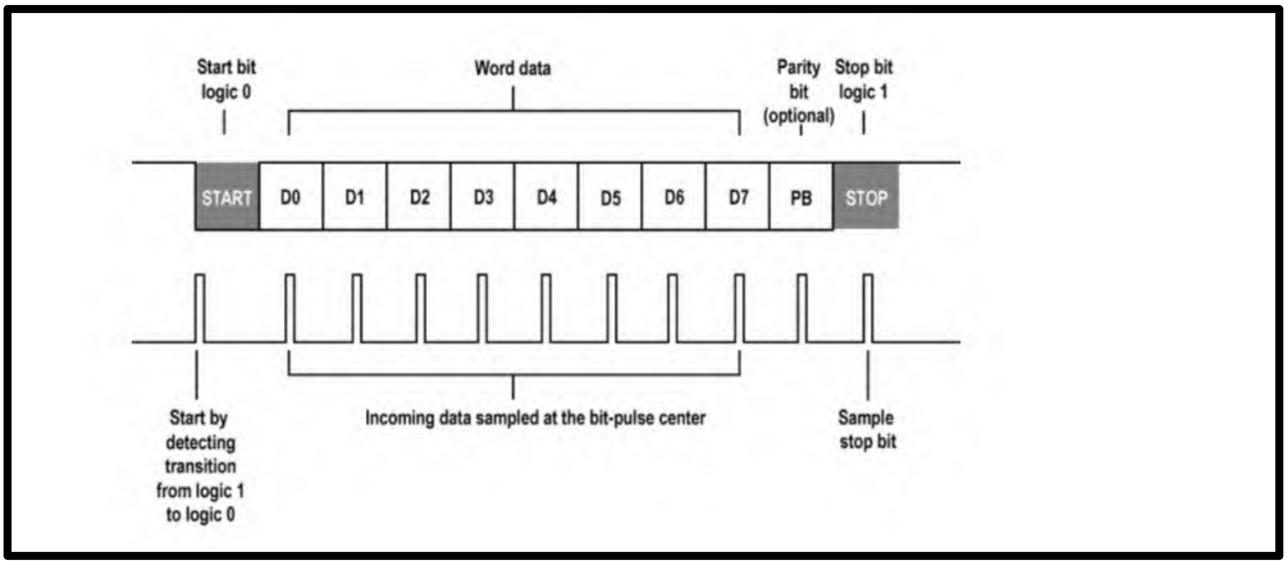


Figure 3.4: UART protocol

CAN bus with UAVCAN

- works in multimaster bus architecture as any node can initiate transmission of data as per need to do basis
- being a packet based protocol, its useful for very long distances
- can achieve high speed of the order of 1 Mb but only 50% of the bus bitrate is usable without major collisions
- requires at least 3 pins (GND, CAN HI, CAN LO). VCC is only to power the sensor.
- Recommended for point-to-point topology and not Star or stubs topology
- All end of busses need to be terminated [3]

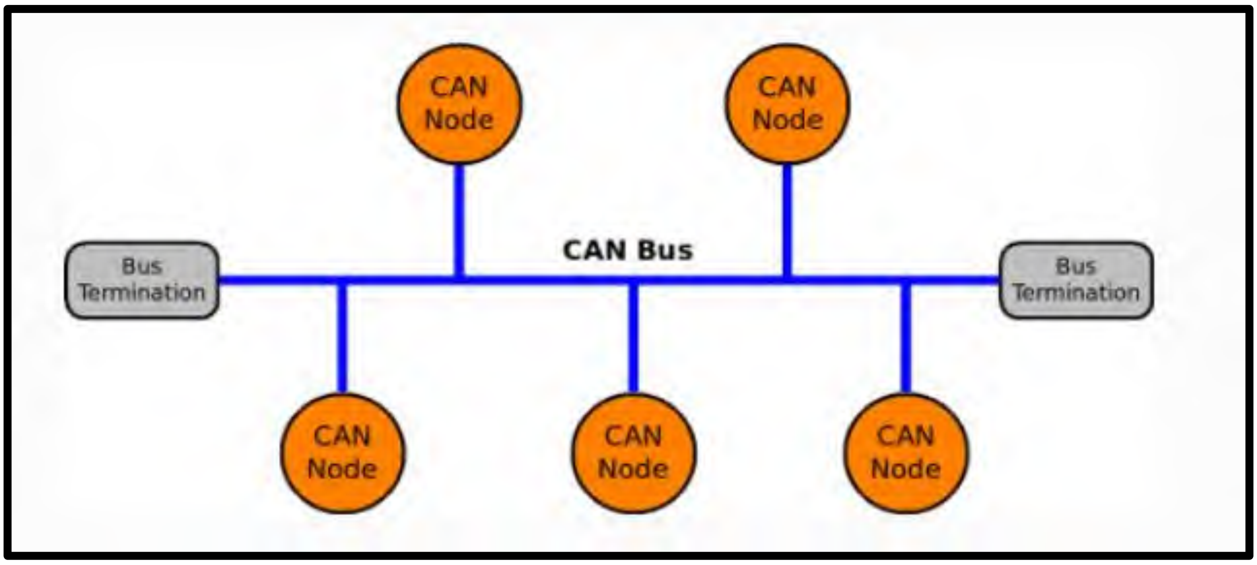


Figure 3.5: CAN Protocol [3]

Broad overview of code for sensor integration

An important concept adopted by Ardupilot autopilot application as part of its sensor driver architecture is the front-end and back-end split mode of implementation. The main Ardupilot code only calls into the front end libraries i.e front end sensor drivers. When the system starts up, the front-end creates one or multiple instances of back-ends based either on

- Automatic detection of the sensor (i.e., probing for a response on a known I2C address)
- Using the user defined _TYPE params (i.e. RNGFND_TYPE, RNGFND_TYPE2).

Pointers to each back-end is maintained by front-end. It is held within an array named drivers and incase of user settable parameters; it is always held within front-end. [3]

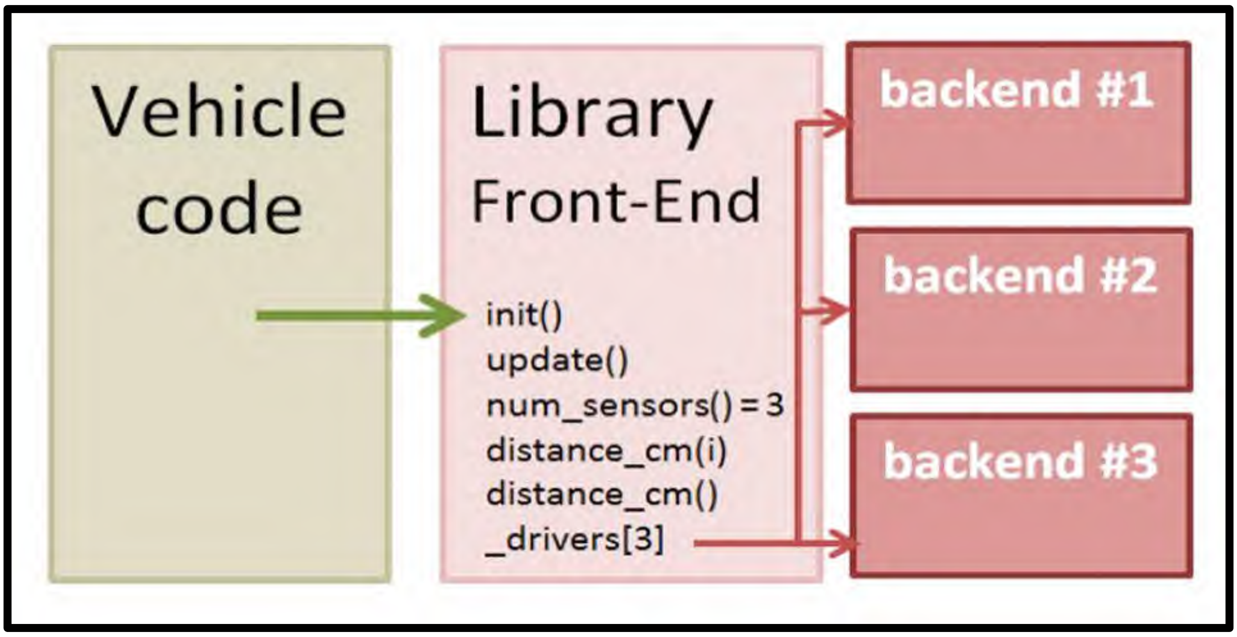


Figure 3.6: Overview of code interaction [3]

Understanding the back-end working of code which runs in background. Sensor raw data is collected and converted to standard units and then the values are held in buffers within drivers. Incase of copters, the vehicle main code thread runs regularly at 400Hz. It accesses latest available data through methods in drivers' front-end. For example , the AHRS/EKF pulls the latest accelerometer, gyroscope and compass information from the sensor drivers' front-ends. The running of background threads ensure that the high-rate communication with the sensor as it does not affect the main loop's performance. Serial (i.e UART) interfaced sensor codes are safe to run in the main thread as the underlying serial driver collects data in the background and includes a buffer. It does not affect the main vehicle code. Hence, I have preferred UART as the interface of choice for most sensors.[3]

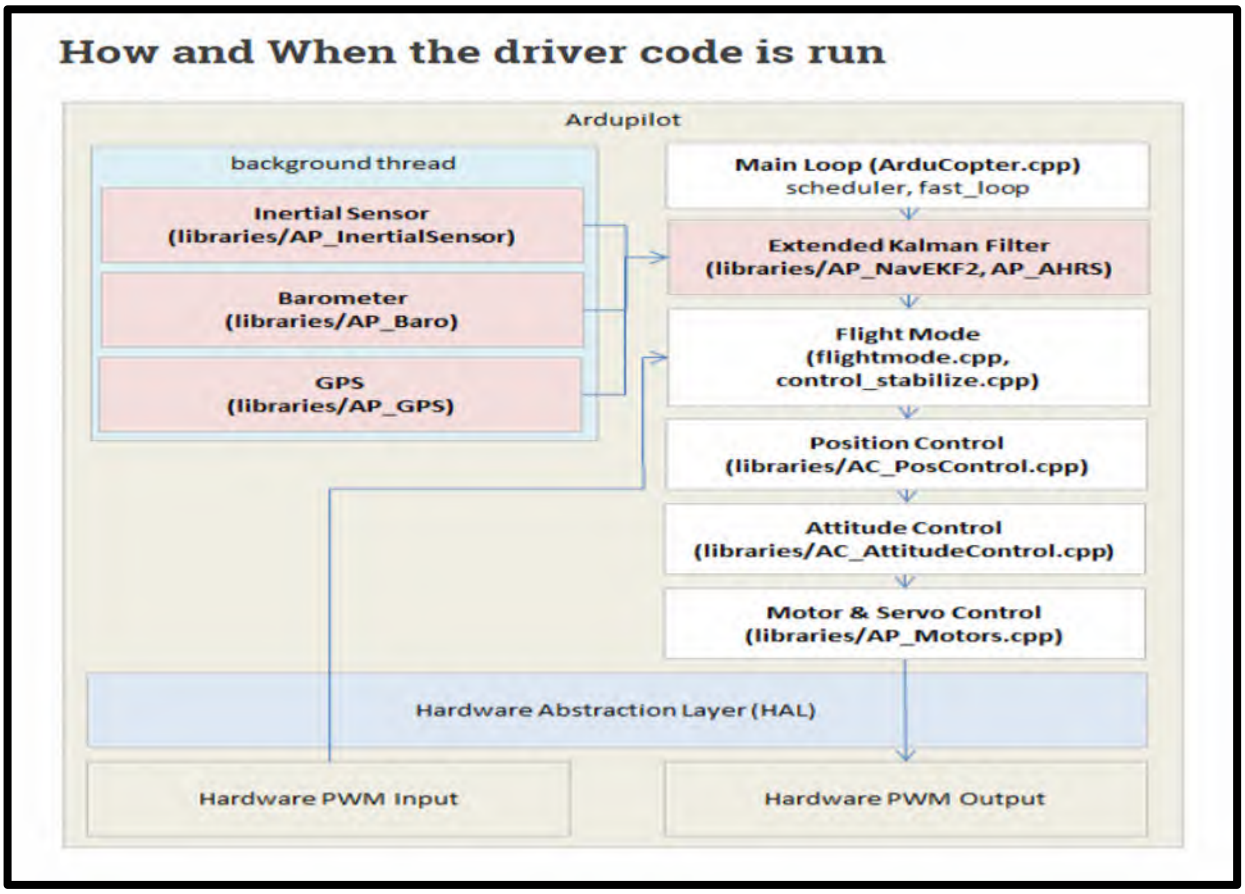

How and When the driver code is run
Ardupilot
background thread
Inertial Sensor (libraries/AP_InertialSensor)
Barometer (libraries/AP_Baro)
GPS (libraries/AP_GPS)
Main Loop (ArduCopter.cpp) scheduler, fast_loop
Extended Kalman Filter (libraries/AP_NavEKF2, AP_AHRS)
Flight Mode (flightmode.cpp, control_stabilize.cpp)
Position Control (libraries/AC_PosControl.cpp)
Attitude Control (libraries/AC_AttitudeControl.cpp)
Motor & Servo Control (libraries/AP_Motors.cpp)
Hardware Abstraction Layer (HAL)
Hardware PWM Input
Hardware PWM Output

Figure 3.7: Driver code interaction flowchart [3]

3.1 Connecting GPS using UART and Compass (inbuilt with GPS) using I2C

GPS with Compass features

- The GPS is driven by Ublox Neo-M8N module
- It has navigation sensitivity of 167 dBm
- It takes 26secs for GPS Cold starts
- Utilizes Low Noise Amplifier MAX2659ELT+
- Supported by a ceramic patch of 25 x 25 x 4 mm
- Supported with 3.3V low noise voltage regulator [7]

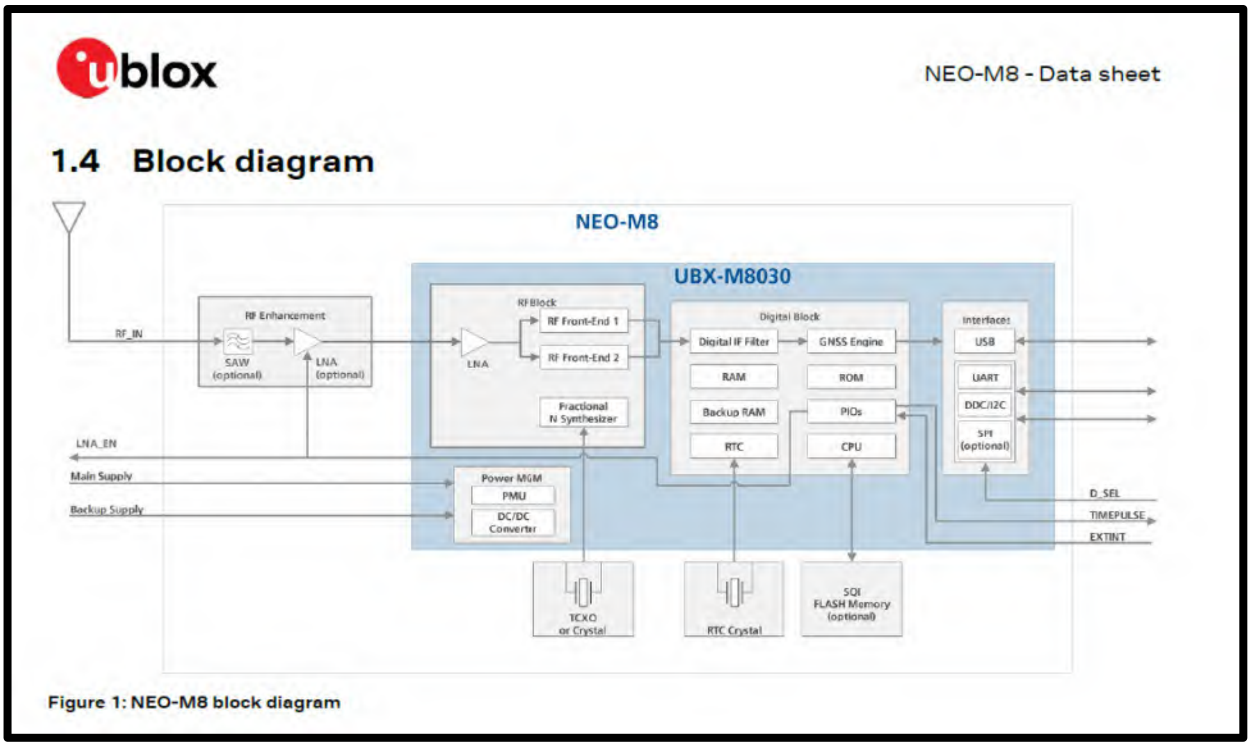


Figure 3.8: Block diagram of UBlox module as per datasheet [7]

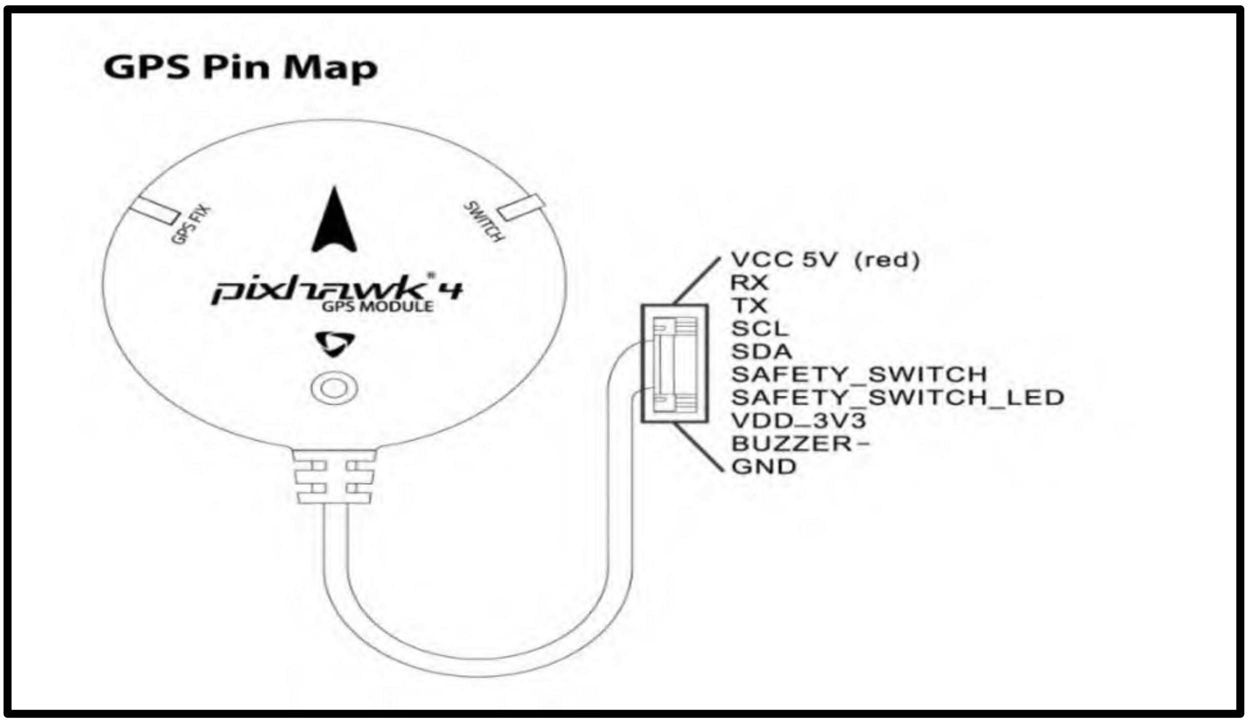


Figure 3.9: Pix Hawk4 GPS with compass pinout

The connectivity diagram for GPS (with Compass) to Beagle Bone Blue as Fight Controller Unit (FCU)

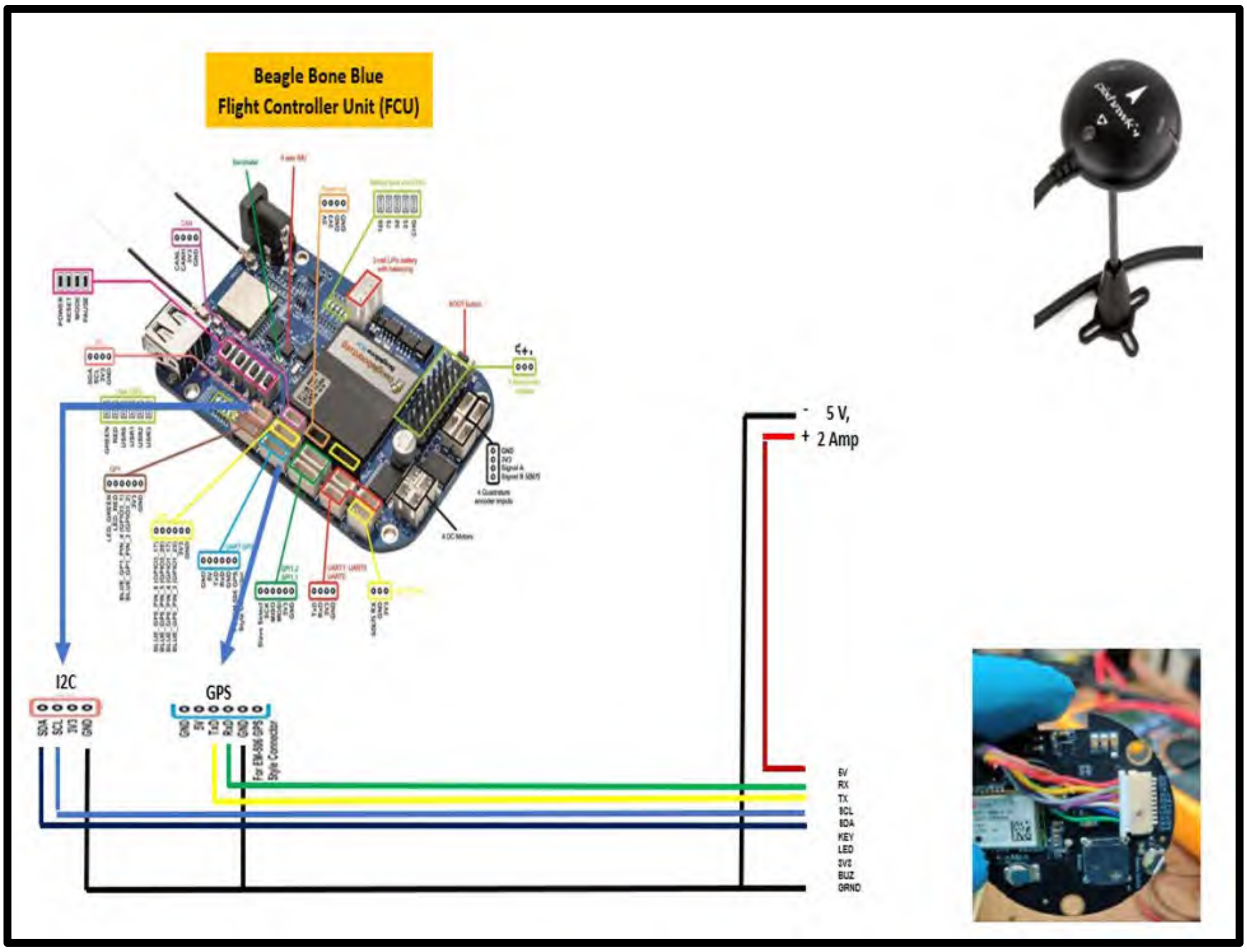


Figure 3.10: Connectivity diagram for physical integration of GPS with compass with Beagle Bone Blue which is the FCU

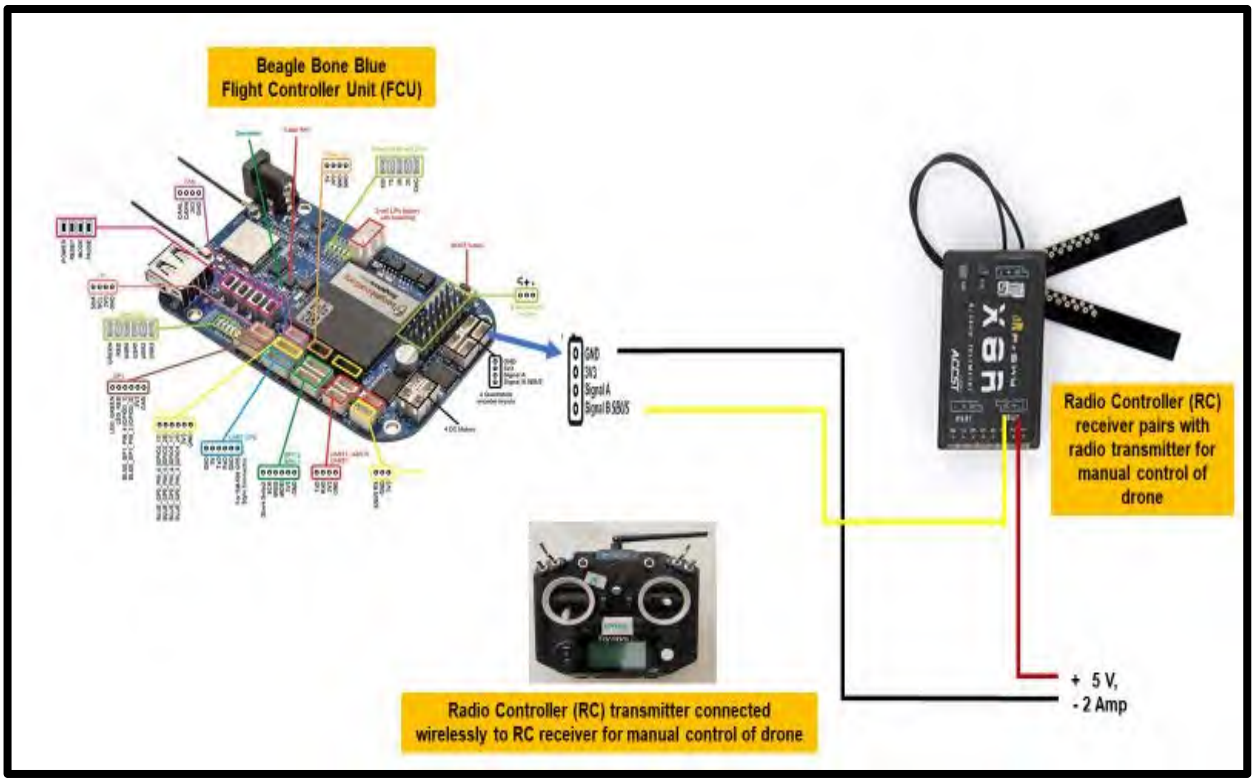
3.2 Connecting Radio Controller (RC) receiver on SBUS

GPS with compass is the bare minimum sensor required for reliable and stable flight operation of a drone. Now for a drone to be manually controller we need a controller. Incase of drones we have a Radio Controller (RC) Transmitter available in various channel configurations to control the drone operations manually from ground station. The RC transmitter is binded to a RC receiver on radio link. The RC receiver present onboard the drone is connected to Flight Controller over wired link running the option of multiple communication protocols.

RC Receiver protocols enable communication with the Flight Controller Unit (FCU). Some communication protocols are proprietary and some are non-proprietary. The list of protocols are PWM (universal), PPM or CPPM (universal), SBUS (Futaba, Frsky), IBUS (Flysky), XBUS (JR), MSP (Multiwii), CRSF (TBS Crossfire), SPEKTRUM1024 (Spektrum DSM2), SPEKTRUM2048 (Spektrum DSMX), FPort (Frsky). I am using the SBUS protocol for the choice of RC receiver i.e X8R. It is connected to SBUS interface on Beagle Bone Blue incase of all three drones i.e X-Type, V-Type and Hexacopter. It requires The 5V and 1-2 Amp power is provided separately through DC convertor and the overall connectivity diagram in Figure 3.10. SBUS is a bus protocol utilized by receivers in order to send commands. Its a digital loss-less bus architecture protocol which uses only 3 wires (signal, power, ground) for multiple channels. Compared to commonly used PPM which is a signal in time domain, SBUS protocols is digital and requires a serial port on the flight controller i.e UART. A single serial line supports upto 16 channels with each receiving a unique command. This SBUS protocol uses an inverted serial logic. It has a baud rate of 100000, 8 data bits, even parity, and 2 stop bits. The packet size of SBUS is 25 bytes and contains:

Byte[0]: SBUS header, 0x0F
Byte[1 -22]: 16 servo channels, 11 bits each
Byte[23]
Bit 7: channel 17 (0x80)
Bit 6: channel 18 (0x40)
Bit 5: frame lost (0x20)
Bit 4: failsafe activated (0x10)
Byte[24]: SBUS footer

RC Transmitter protocols enable communication with the RC Receiver onboard drone over radio link. Its communication protocol for radio transmitter and receiver (two way link). Some communication protocols are proprietary and some are non-proprietary. The list of protocols are ACCST (Frsky), ACCESS (Frsky), DSM (Spektrum), DSM2 (Spektrum), DSMX (Spektrum), AFHDS (Flysky), AFHDS 2A (Flysky), A-FHSS (Hitec), FASST (Futaba), Hi-Sky (Deviation / Devo), Frsky's TX Protocols. Frsky has two TX protocols, ACCST and ACCESS. I have utilized Frsky RC transmitter.


Beagle Bone Blue
Flight Controller Unit (FCU)
GND
3V3
Signal A
Signal B
Radio Controller (RC) receiver pairs with radio transmitter for manual control of drone
+ 5 V,
- 2 Amp
Radio Controller (RC) transmitter connected wirelessly to RC receiver for manual control of drone
Figure 3.11: Connectivity diagram for physical integration of Radio controller (RC) Receiver with Beagle Bone Blue which is the FCU. The Radio controller (RC) transmitter is connected to RC receiver over radio link.

3.3 Connecting LIDAR as proximity sensor on UART

A2 RP LIDAR Model: A2M3, A2M4 was integrated with the Hexacopter. Its **a** 360-degree 2D laser scanner (LIDAR) with 6 meter range. It supports up to 4000 samples of laser ranging per second with high rotation speed. It performs 360-degree 2D to generate 2D point cloud data which is used for mapping, localization and object/environment modeling. The scanning frequency is 10hz (600rpm) with best resolution of 0.9-Degrees. It adopts low-cost laser triangulation measurement system. It emits modulated infrared laser, during its ranging process. The signal and the laser signal is reflected by the object in its field of view. The reflected signal is sampled by vision acquisition system and then the embedded DSP starts processing the sample data with distance and angle between object and RPLIDAR through communication interface. The motor system, drives the range scanner core which rotates clockwise to perform the 360-degree scan [6].

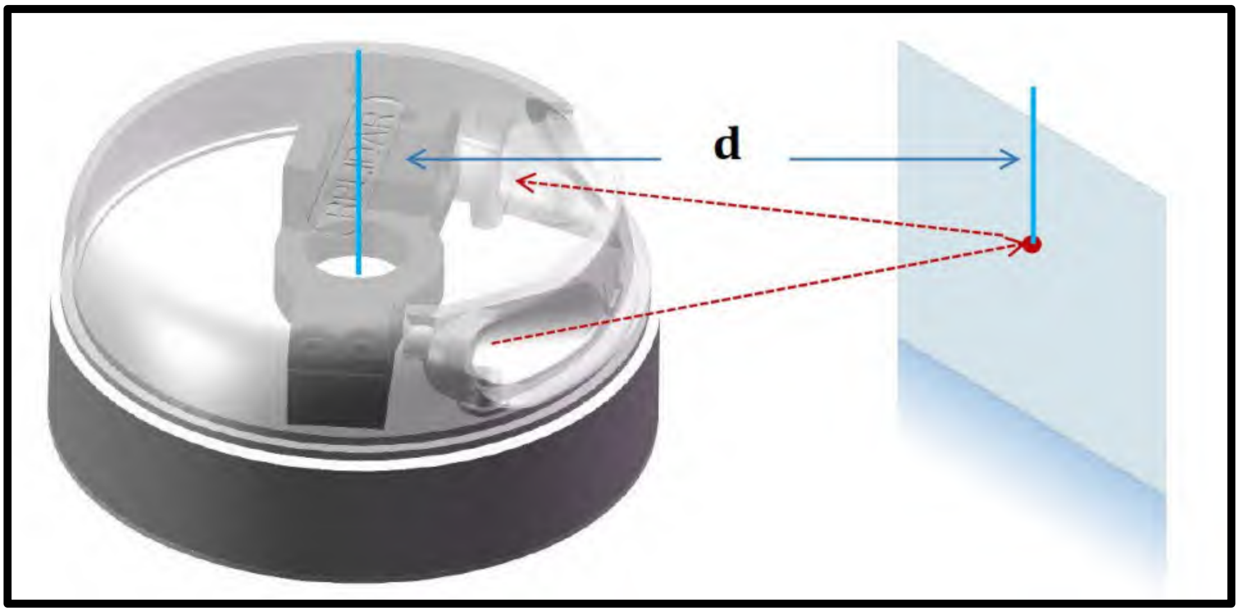
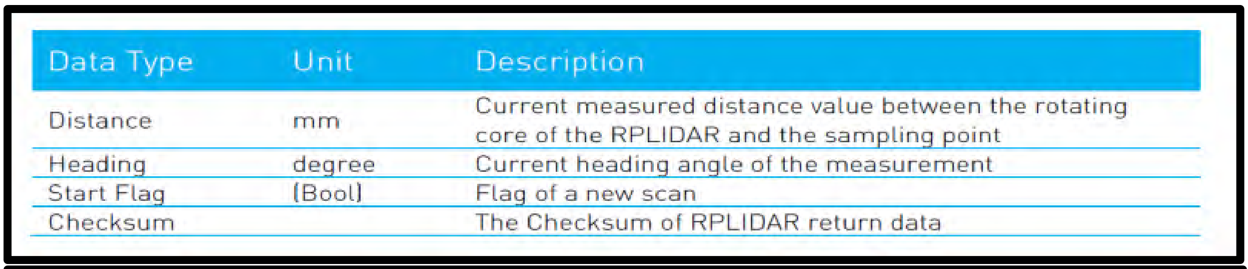


Figure 3.12: Pictorial representation of LIDAR working as per datasheet [6]

The light source of LIDAR system is a low power infrared laser. It drives the system by using modulated pulse. The duration of laser light emission is very short and complies with Class I laser safety standard as per product datasheet. The sample point data contains the information as shown in picture below [6]:



Data Type	Unit	Description
Distance	mm	Current measured distance value between the rotating core of the RPLIDAR and the sampling point
Heading	degree	Current heading angle of the measurement
Start Flag	(Bool)	Flag of a new scan
Checksum		The Checksum of RPLIDAR return data

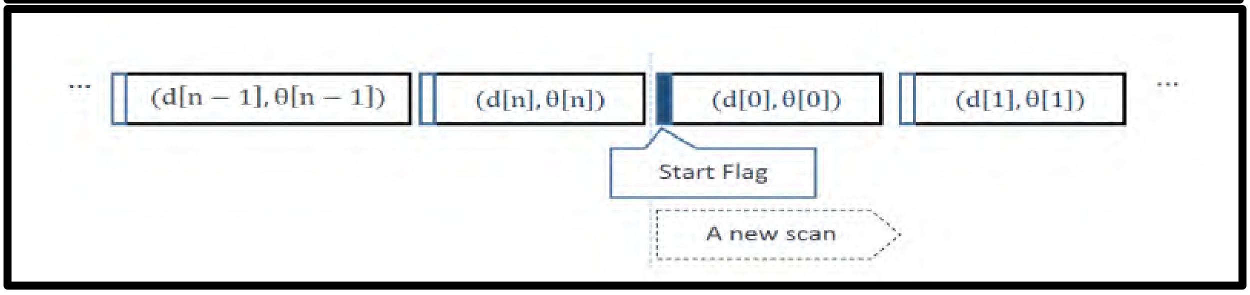


Figure 3.13: Distance and angle information sent as part of data sample as per datasheet [6]

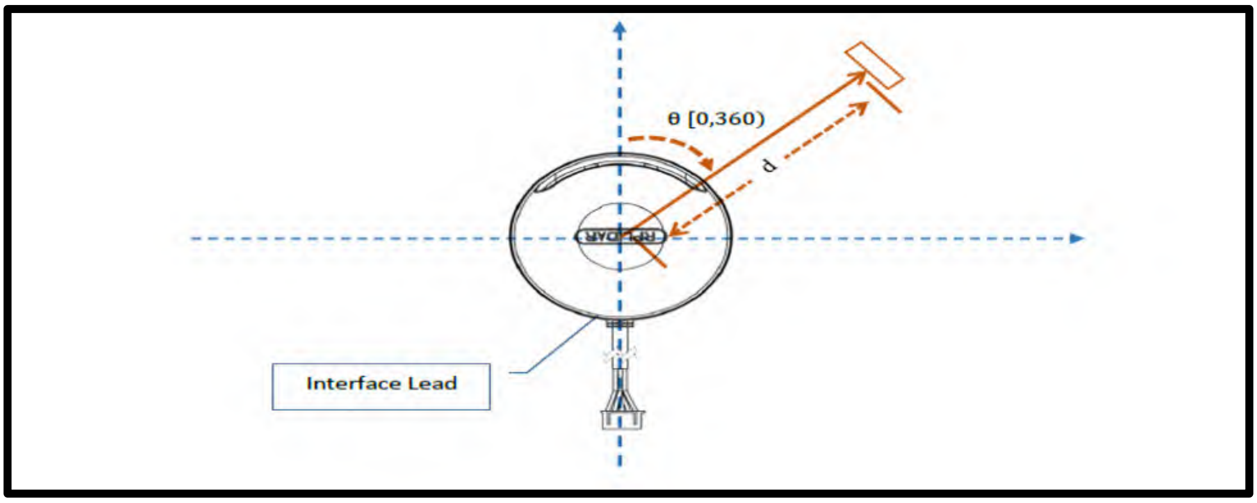
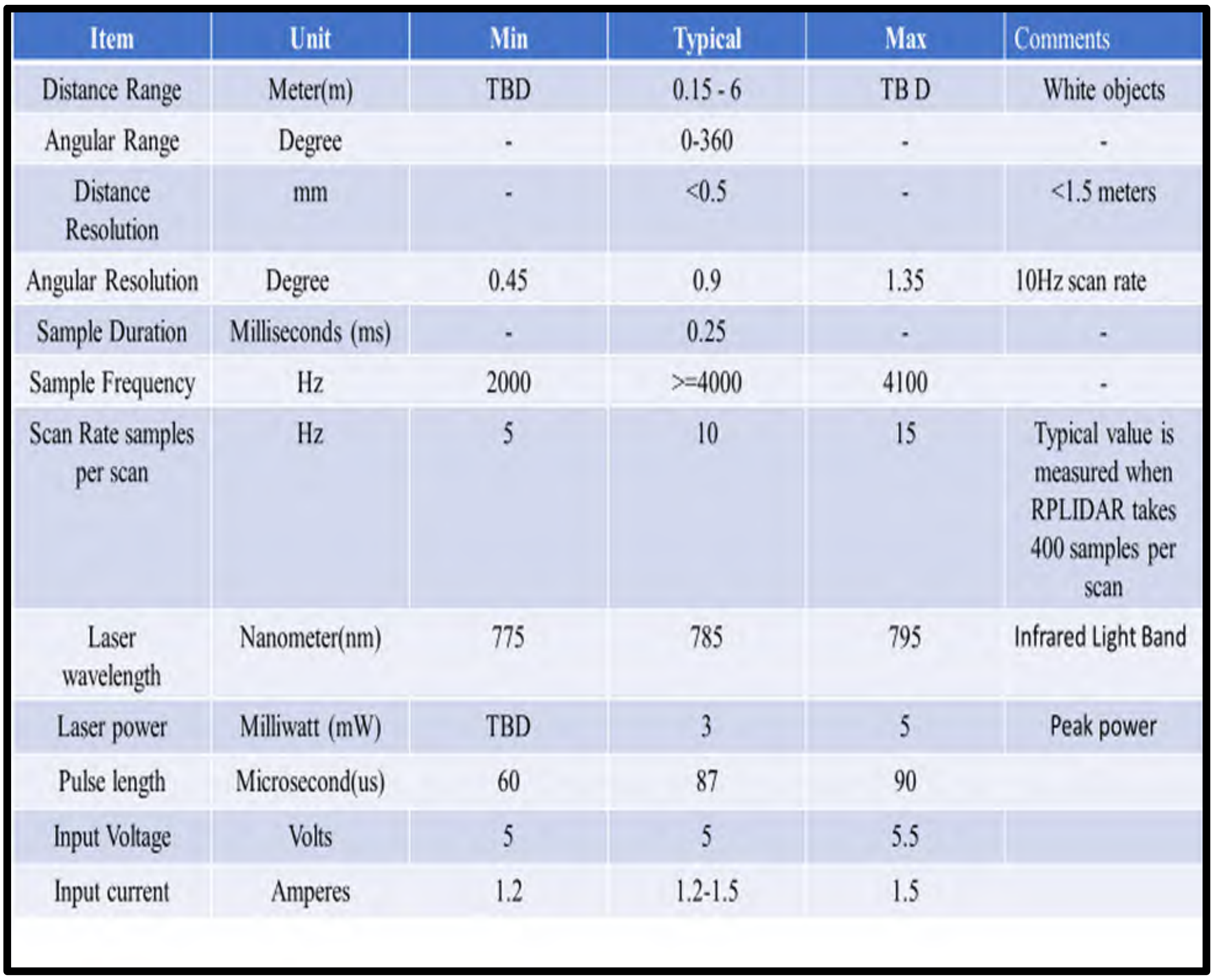
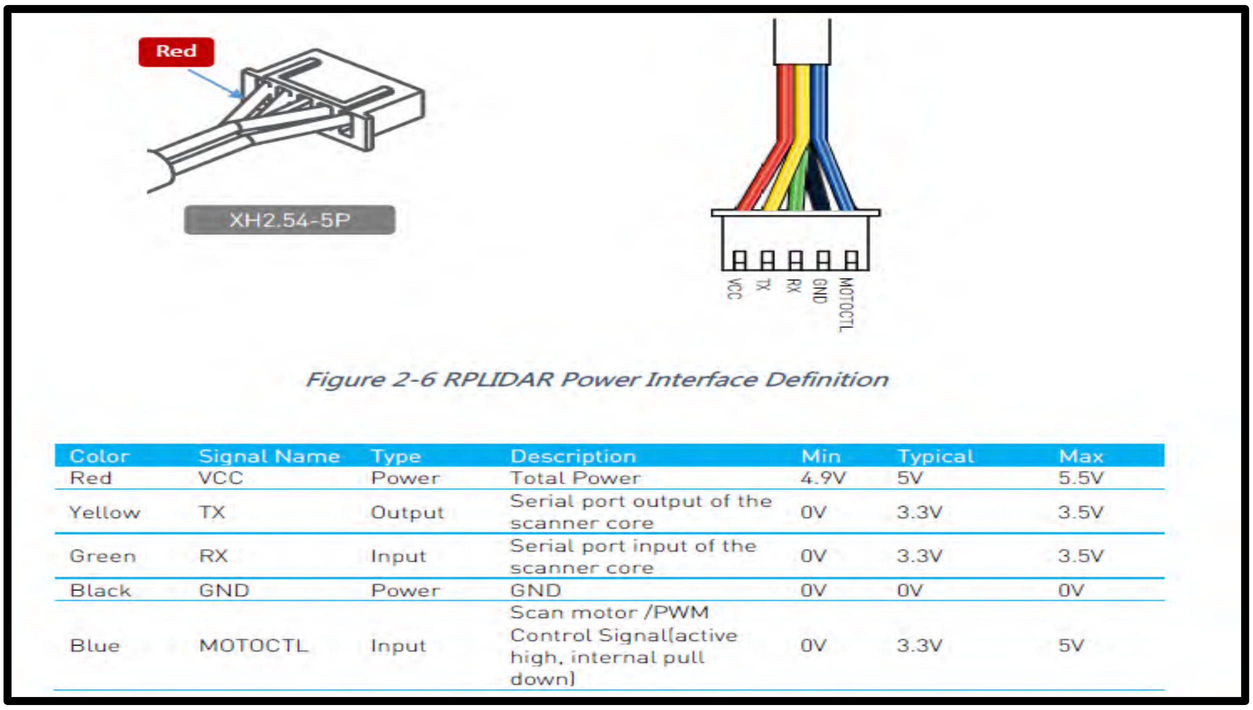


Figure 3.14: Rotation and angle calculation philosophy in LIDAR [6]



Item	Unit	Min	Typical	Max	Comments
Distance Range	Meter(m)	TBD	0.15 - 6	TB D	White objects
Angular Range	Degree	-	0-360	-	-
Distance Resolution	mm	-	<0.5	-	<1.5 meters
Angular Resolution	Degree	0.45	0.9	1.35	10Hz scan rate
Sample Duration	Milliseconds (ms)	-	0.25	-	-
Sample Frequency	Hz	2000	>=4000	4100	-
Scan Rate samples per scan	Hz	5	10	15	Typical value is measured when RPLIDAR takes 400 samples per scan
Laser wavelength	Nanometer(nm)	775	785	795	Infrared Light Band
Laser power	Milliwatt (mW)	TBD	3	5	Peak power
Pulse length	Microsecond(us)	60	87	90	
Input Voltage	Volts	5	5	5.5	
Input current	Amperes	1.2	1.2-1.5	1.5	

Figure 3.15: Operating values of LIDAR as per datasheet



Color	Signal Name	Type	Description	Min	Typical	Max
Red	VCC	Power	Total Power	4.9V	5V	5.5V
Yellow	TX	Output	Serial port output of the scanner core	0V	3.3V	3.5V
Green	RX	Input	Serial port input of the scanner core	0V	3.3V	3.5V
Black	GND	Power	GND	0V	0V	0V
Blue	MOTOCTL	Input	Scan motor /PWM Control Signal(active high, internal pull down)	0V	3.3V	5V

Figure 3.16: Pinout details of LIDAR connector as per datasheet [6]

LIDAR Connectivity diagram The LIDAR on drone has to be mounted horizontally. It could be on the top or bottom. The black cable must point towards the rear of the vehicle. The field of view of the sensor must remain unobstructed which includes the , drone frame parts like legs, GPS mast or any other component.

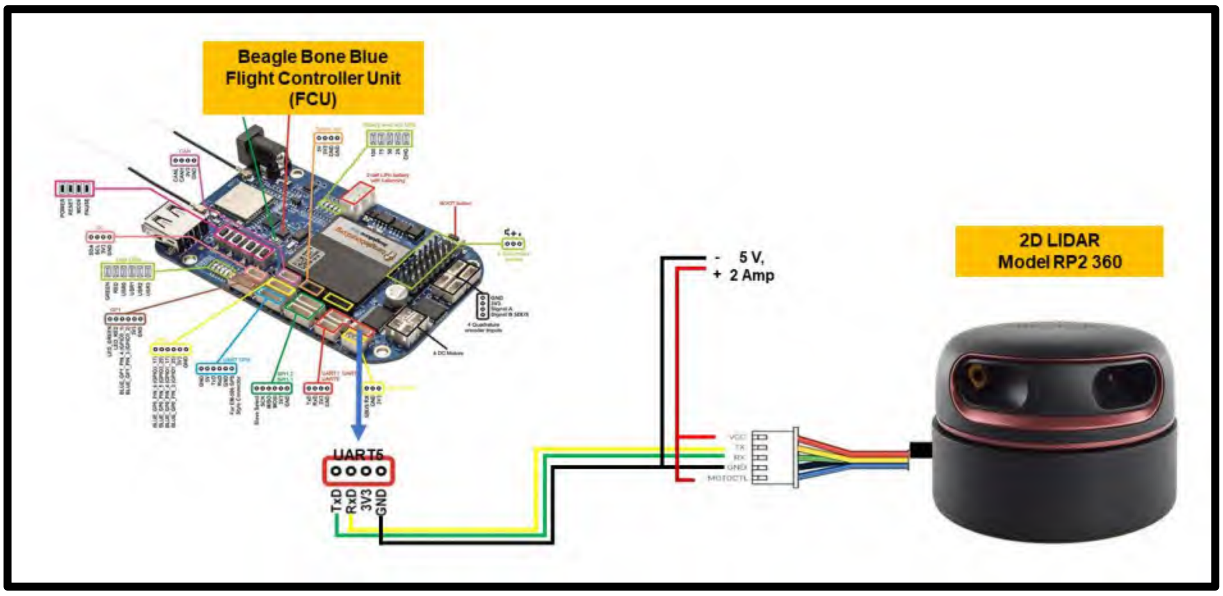
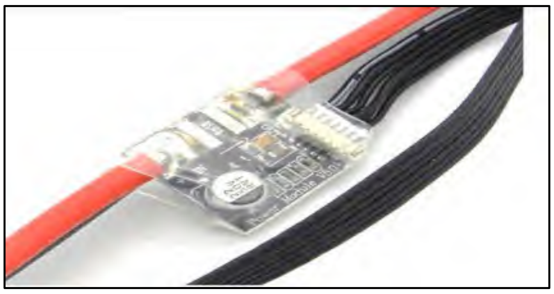


Figure 3.17: Connectivity diagram for physical integration of LIDAR with Beagle Bone Blue which is the FCU

3.4 Connecting power module to ADC of Beagle Bone Blue

Power Module The APM power Module is connected to ADC port of the Beagle Bone Blue. The power module sits between Input from battery and power distribution for drone components with XT60 Connectors, Male and Female connectors on respective sides. The APM module provides clean power from battery and also provide voltage measurement between 6-28V (upto 6s battery) and 90 Amps of current. The signal output for voltage measurement is 5V while ADC input range is 1.8V max. Thus, we use 1k and 2k resistors to keep signal voltage less than 1.8V. The pinout and connection details are in the connectivity diagram below:-



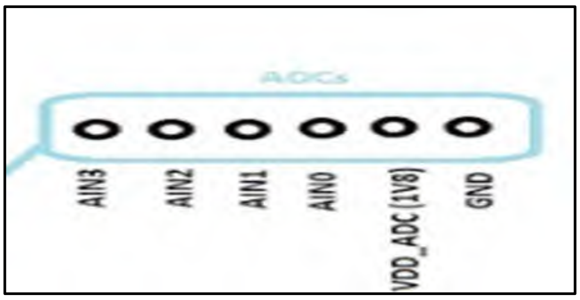
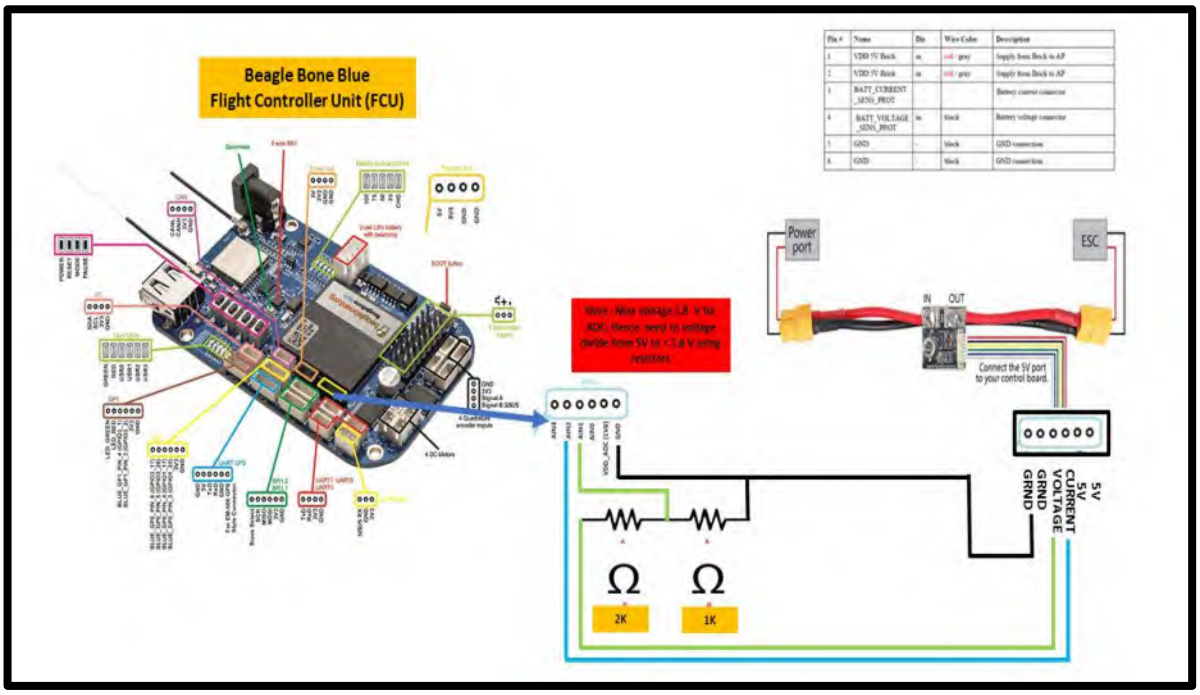


Figure 3.18: Power module pinout



Note : Max voltage 1.8 V for ADC. Hence need to voltage divide from 5V to < 1.8 V using resistors.

Pin #	Name	Dir	Wire Color	Description
1	VDD 5V Brick	in	red / gray	Supply from Brick to AP
2	VDD 5V Brick	in	red / gray	Supply from Brick to AP
3	BATT_CURRENT_SENS_PROT			Battery current connector
4	BATT_VOLTAGE_SENS_PROT	in	black	Battery voltage connector
5	GND	-	black	GND connection
6	GND	-	black	GND connection

Figure 3.19: Connectivity diagram for physical integration of power module with Beagle Bone Blue which is the FCU

3.5 Connecting Electronic Speed Control (ESC) and motor to PWM input of Beagle Bone Blue

Electronic Speed Control (ESC) and Motors The connectivity of Electronic Speed Control with motors and Flight Controller Unit i.e Beagle Bone Blue is illustrated in picture below.

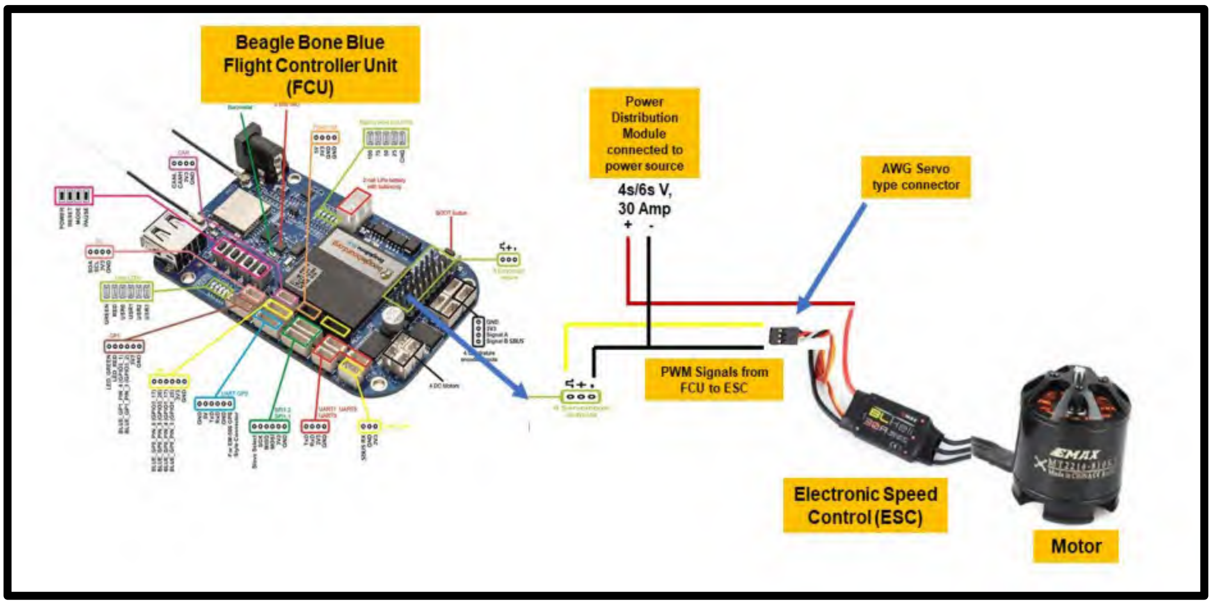

Beagle Bone Blue Flight Controller Unit (FCU)
Power Distribution Module connected to power source
4s/6s V, 30 Amp
+ -
AWG Servo type connector
PWM Signals from FCU to ESC
Electronic Speed Control (ESC)
Motor

Figure 3.20: Connectivity diagram for physical integration of Electronic Speed Control (ESCs) and motors with Beagle Bone Blue which is the FCU

We have utilized the Brushless DC motors procured off the shelf. The motor and ESC make and model details are available in figure 1.11 and 1.12. The BLDC motor meets the requirement for a mechanical commutator as required in drone application by reversing the motor set-up. As a result, the windings become the stator and the permanent magnets therefore become part of the rotor. These BLDC motors are powered by MOSFET bridge controllers popularly known as ESCs using pulse-width modulation (PWM). The windings are commutated in a controlled sequence. As a result of that, it produces a rotating magnetic field which “drags” the rotor around and consequently drives the attached load which in our case our propellers. We have tested propellers from carbon fiber to composite materials during our multiple flight tests. Learning from the existing ESC and motor combines being used by us, we have been able to simulate motor design and are in the final stages of fabricating motor with inhouse design. This aspect of understanding motor theory and inhouse design has been covered in detail by Akhil Sharma, the other student as part of this project in his thesis. The frame and the aerodynamic part of the drone is covered by Tony Joseph as part of his thesis in the project.

CHAPTER 4

UNDERSTANDING MAVLINK BETWEEN DRONE AND GROUND STATION (GCS)

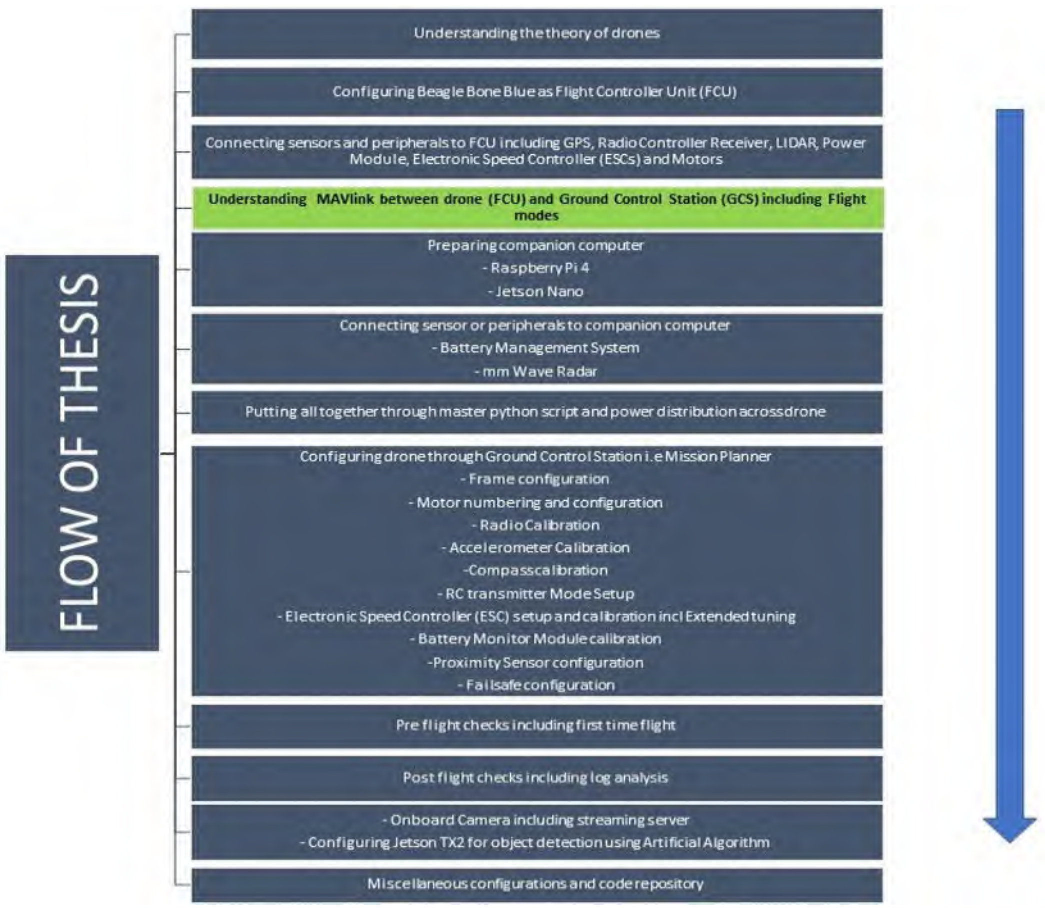

FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server
- Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

Figure 4.1: Flow of thesis until now

The Micro Air Vehicle Link (MAVlink) is a light-weight serialization communication protocol for bidirectional communication between unmanned aerial vehicles/drone and GCS for bi-direction communication. The MAVlink protocol defines a complete set of messages that are exchanged between drone and ground stations. The generated libraries in this protocol are MIT-licensed and can be used without limits in any open or closed-source application without the need for publishing the source code. MAVLink was first released in 2009 by Lorenz Meier and is being supported by a large community of developers. This protocol is utilized by most widely used ArduPilot and PX4. The most popular autopilot software Ardupilot actively supported by a battery of enthusiasts all across for world for over a decade has seamless integration with this communication protocol. [5]

The binary serialization makes it lightweight with minimal overhead. Due to its small size, it can be reliably utilized over wireless links including Wifi, low data rate telemetry links and even on internet. MAVlink protocol defines the mechanism on the structure of messages, how to serialize them at the application layer and way to forward to the lower layers (i.e., transport layer, physical layer) which can be transmitted on the network. As a result ,it can be transmitted through WiFi, Ethernet (i.e., TCP/IP Networks) or low-bandwidth serial telemetry links (namely 433 MHz, 868 MHz or 915 MHz). The use of these low-data rate frequencies allow for higher range of operations. MAVLink protocol ensures the reliability and integrity of messages by double checksum verification in its packet header. [5]

MAVLink follows the latest hybrid publish-subscribe and point-to-point design pattern

- Data streams are sent or published as topics
- All configuration sub-protocols adopted in MAVlink protocol (e.g mission protocol) or parameter protocol are defined as point-to-point with retransmission.

Messages are defined in XML files. The message set supported by a particular MAVLink system referred as a "dialect". The reference message set implemented by GCS and autopilots is defined in common.xml upon which most dialects are built upon. Code generators create software libraries for specific programming languages from XML message definitions. This is utilized by drones, ground control stations, and other MAVLink systems to communicate. All these features make MAVLink protocol the most utilized protocol. [5]

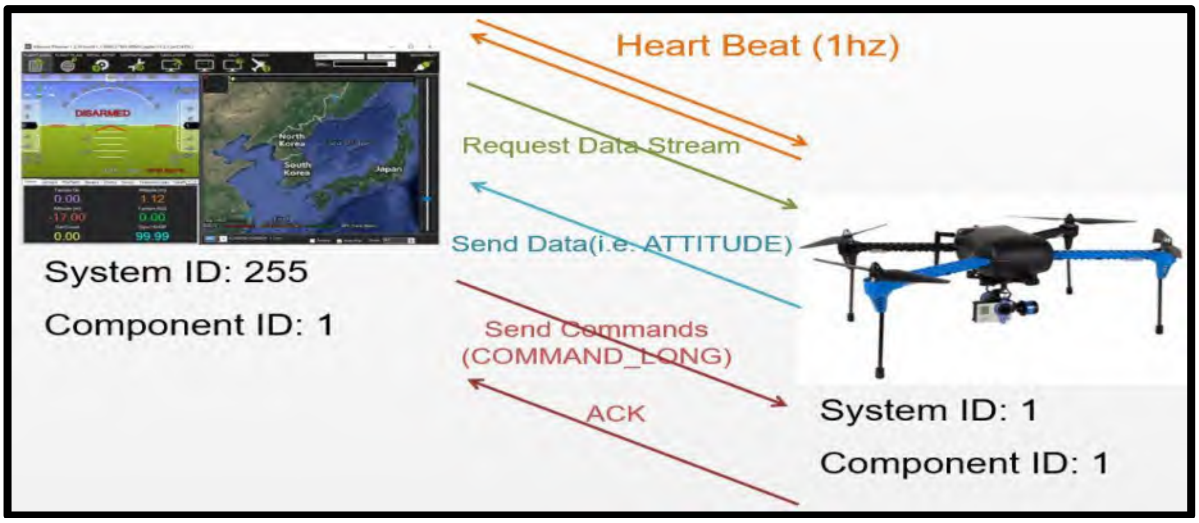
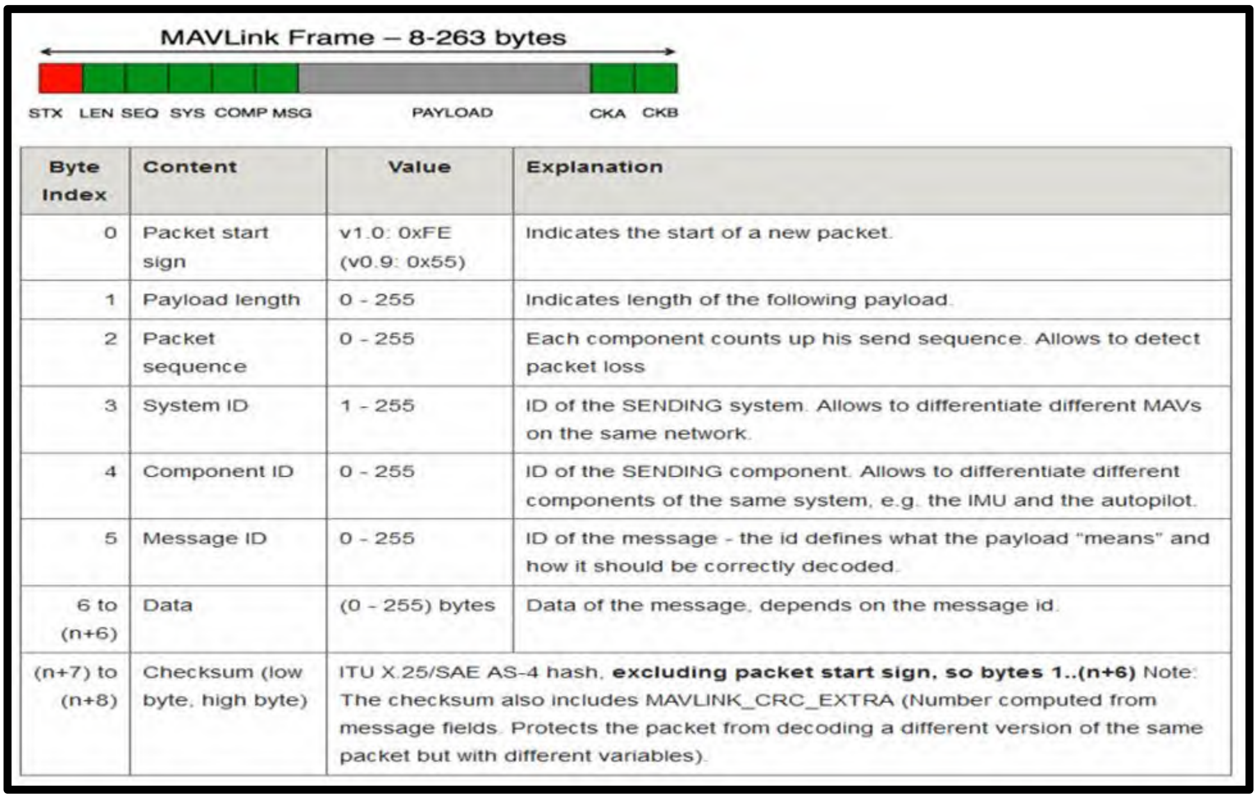


Figure 4.2: MAVlink between drone and Ground Control Station (GCS) application i.e Mission Planner in my case [3]

In our drone setup, I have utilized IP Network to stream MAVlink messages. The IP network utilizes the private VPN over internet which will be explained in detail in next chapter. MAVLink protocol typically supports both UDP and TCP connections at the transport layer between the ground station and the drone. The choice between the two depends upon the reliability of the network link end to end. UDP is a connectionless datagram protocol which requires no connection between the client and server. It has no acknowledgement mechanism between sender and receiver ,making it less tolerant to errors. However, it is light-weight, fast and real time. On the other hand TCP protocol which adopts a connection oriented approach leading to multiple retransmissions and overheads leading to congestion. As a result, I have preferred UDP over TCP protocol for MAVlink over private VPN through 4G LTE Modem. [5]

MAVLink being a binary telemetry protocol is designed for resource-constrained systems including bandwidth. It is deployed in two major versions: v1.0 and v2.0. v2.0 is backwards-compatible. The telemetry data streams are sent in a multicast design. The protocol aspects which change the system configuration requiring guaranteed delivery such as mission protocol or parameter protocol are point-to-point with retransmission capability. [5]

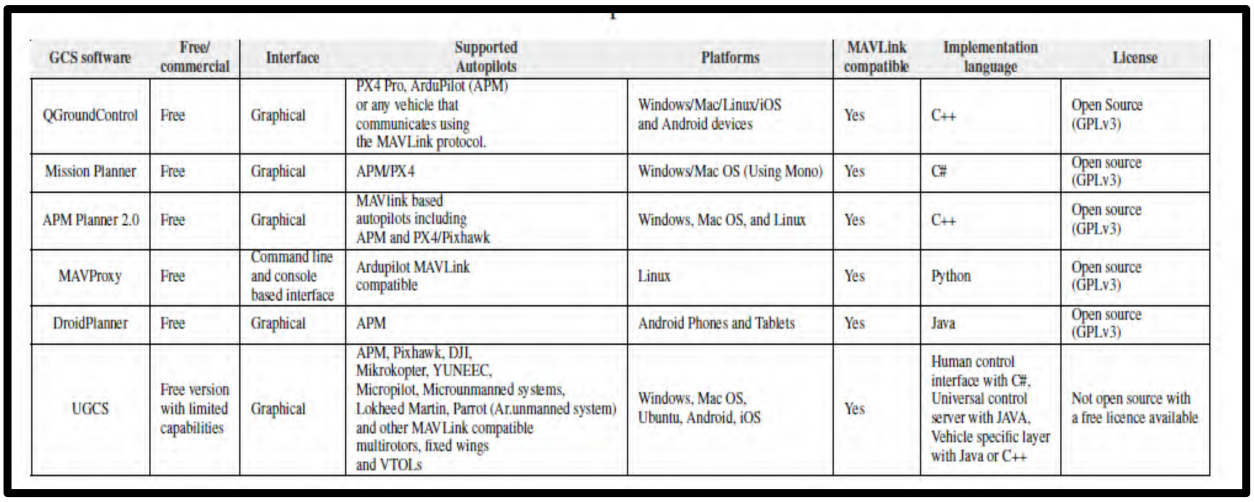

MAVLink Frame – 8-263 bytes
STX
LEN
SEQ
SYS
COMP
MSG
PAYLOAD
CKA
CKB

Byte Index	Content	Value	Explanation
0	Packet start sign	v1.0: 0xFE (v0.9: 0x55)	Indicates the start of a new packet.
1	Payload length	0 - 255	Indicates length of the following payload.
2	Packet sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5	Message ID	0 - 255	ID of the message - the id defines what the payload "means" and how it should be correctly decoded.
6 to (n+6)	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+7) to (n+8)	Checksum (low byte, high byte)	ITU X.25/SAE AS-4 hash, **excluding packet start sign, so bytes 1..(n+6)** Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).	

Figure 4.3: MAVlink frame v2 [3]

4.1 Introduction to Ground Control Station (GCS)

A Ground Control Station (GCS) is a software application which communicates with the drone through a serial or network interface by exchanging MAVLink messages. The communication either takes place over serial port through a telemetry device or via a network interface using UDP or TCP over Wifi (Local Area Network or internet) or 4G Communication module like in our case. The aspect of MAVlink over internet is explained in detail subsequently. Various types of GCS are as below:-



GCS software	Free/ commercial	Interface	Supported Autopilots	Platforms	MAVLink compatible	Implementation language	License
QGroundControl	Free	Graphical	PX4 Pro, ArduPilot (APM) or any vehicle that communicates using the MAVLink protocol.	Windows/Mac/Linux/iOS and Android devices	Yes	C++	Open Source (GPLv3)
Mission Planner	Free	Graphical	APM/PX4	Windows/Mac OS (Using Mono)	Yes	C#	Open source (GPLv3)
APM Planner 2.0	Free	Graphical	MAVlink based autopilots including APM and PX4/Pixhawk	Windows, Mac OS, and Linux	Yes	C++	Open source (GPLv3)
MAVProxy	Free	Command line and console based interface	Ardupilot MAVLink compatible	Linux	Yes	Python	Open source (GPLv3)
DroidPlanner	Free	Graphical	APM	Android Phones and Tablets	Yes	Java	Open source (GPLv3)
UGCS	Free version with limited capabilities	Graphical	APM, Pixhawk, DJI, Mikrokopter, YUNEEC, Micropilot, Microunmanned systems, Lokheed Martin, Parrot (Ar.unmanned system) and other MAVLink compatible multirotors, fixed wings and VTOLs	Windows, Mac OS, Ubuntu, Android, iOS	Yes	Human control interface with C#, Universal control server with JAVA, Vehicle specific layer with Java or C++	Not open source with a free licence available

Figure 4.4: Types of Ground Control Station (GCS) applications [5]

Mission Planner is one of the most popular GCS which allows a number of customizations and is completely open source. It was created and is regularly supported by Michael Oborne. It runs on Windows platforms only. It has host of features from planning an autonomous mission to taking full control of the drone. It provides support for video streaming and changing the internal parameters of the autopilot along with the ability for calibration of the sensors of the autopilot. We can download logs and analyze flight performance post mission with support for simulations to improve piloting. All configurations of drone (FCU) can be done from here and is stored in log files for analysis. I have used Mission Planner as GCS for doing the project and dedicated a chapter to understanding and utilizing Mission Planner later in the thesis. Any reference to GCS from hereon will mean Mission Planner for the purpose of my thesis. Also I am using UDP over network interface established over private VPN over 4G LTE modem. [5]

QGroundControl is also a very famous QGC which supports both Ardupilot and PX4 autopilot applications. It supports the MAVLink protocol and has several functionalities, including defining and planning autonomous missions, complete control of the drone, graphical visualization of the map and location tracking of the drone through its GPS coordinates. It provides support for video streaming and changing the internal parameters of the autopilot along with the ability for calibration of the sensors of the autopilot. QGroundControl can run on different platforms, namely, Windows, Mac OS, IOS, and Android devices [5]. However. QGroundControl has a lot of automated functions providing little less options than Mission Planner to customize drone settings. Hence, I preferred Mission Planner as GCS though the concept of utilization remains the same for both.

In **Small Drones** the MAVlink will be established directly between Beagle Bone Blue (BBB) which is a Linux based Flight Controller Unit (FCU) and Mission Planner which is the Ground Control Station (GCS) over UDP protocol. The link can be engineered over wifi (Local LAN) or over internet (VPN). This setup was tested with X-Type Drone as Beagle Bone Blue as acting like FCU and also onboard computer running Linux. The MAVlink is over private VPN using UDP protocol and 4G LTE Modem as hardware. This configuration is only possible with drones having Beagle Bone Blue as FCU.

In **Medium and Large Drones** the MAVlink is established indirectly between the FCU (in my case Beagle Bone Blue) and Mission Planner which is the Ground Control Station (GCS) via companion computer (Raspberry Pi 4 or Jetson Nano). The Flight controller Unit (FCU) is Beagle Bone Blue (BBB) but the configuration suggested is equally applicable to any other FCU (STM based running RTOS). The link between FCU and Companion computer will be through UART and companion computer to GCS over private VPN using UDP protocol and 4G LTE Modem. This setup was tested on both V-Type and Hexacopter. The difference was that V-Type has Raspberry Pi 4 as companion computer while Hexacopter has Jetson Nano as companion computer.

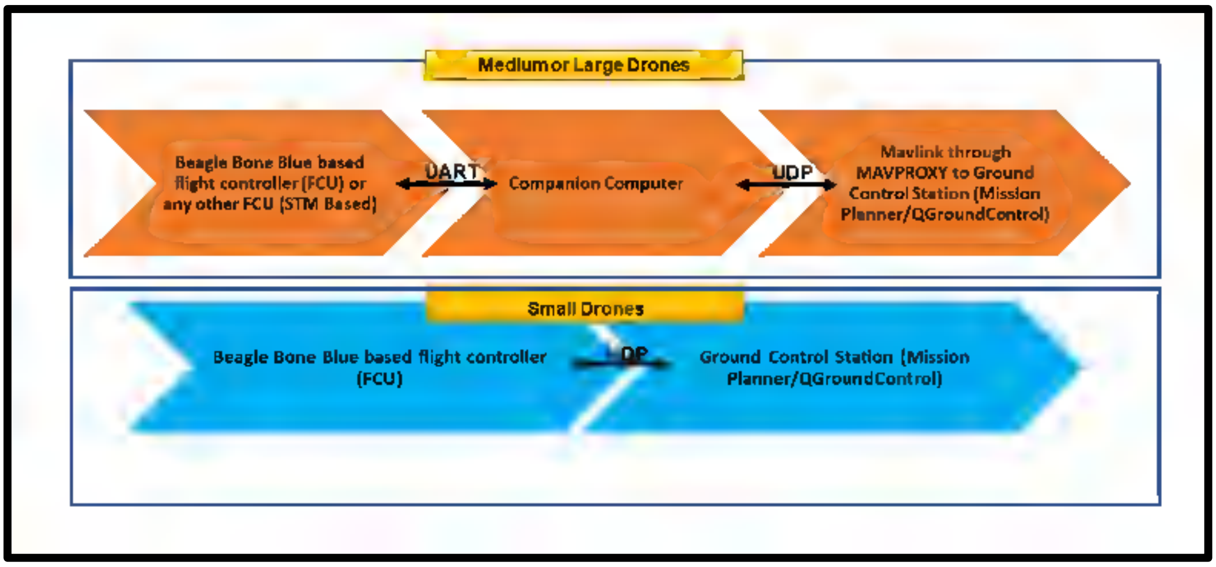
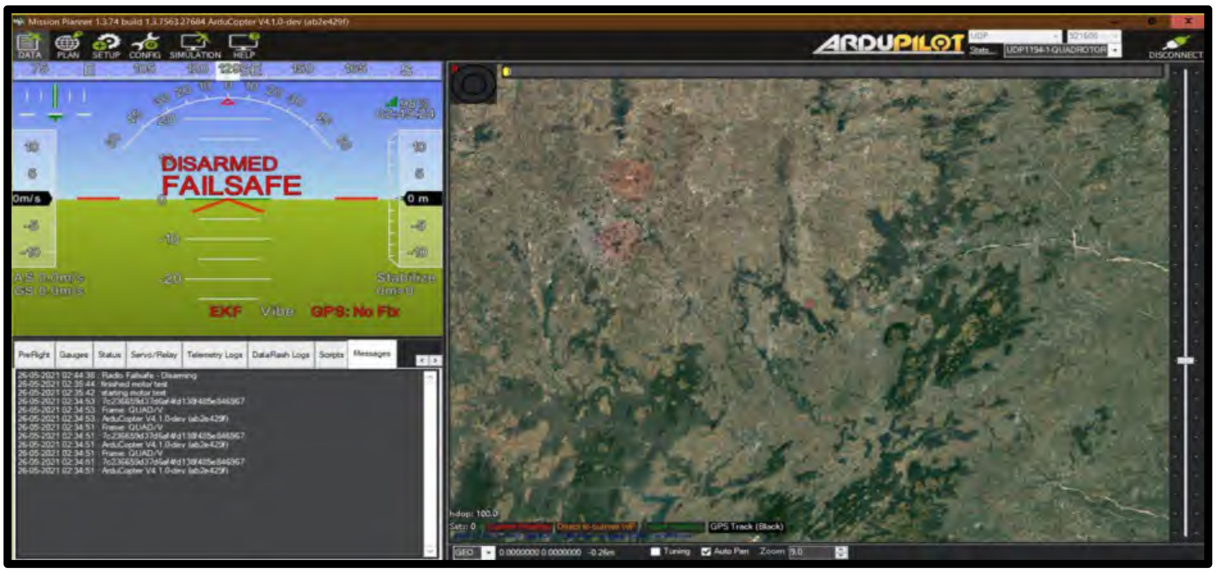

Medium or Large Drones
Beagle Bone Blue based flight controller (FCU) or any other FCU (STM Based)
UART
Companion Computer
UDP
Mavlink through MAVPROXY to Ground Control Station (Mission Planner/QGroundControl)
Small Drones
Beagle Bone Blue based flight controller (FCU)
UDP
Ground Control Station (Mission Planner/QGroundControl)

Figure 4.5: Flow of MAVlink



```
26-05-2021 02:44:38 : Radio Failsafe - Disarming
26-05-2021 02:35:44 : finished motor test
26-05-2021 02:35:42 : starting motor test
26-05-2021 02:34:53 : 7c236659d37d6af4fd138f405e846967
26-05-2021 02:34:53 : Frame: QUAD/V
26-05-2021 02:34:53 : ArduCopter V4.1.0-dev (ab2e429f)
26-05-2021 02:34:51 : Frame: QUAD/V
26-05-2021 02:34:51 : 7c236659d37d6af4fd138f405e846967
26-05-2021 02:34:51 : ArduCopter V4.1.0-dev (ab2e429f)
26-05-2021 02:34:51 : Frame: QUAD/V
26-05-2021 02:34:51 : 7c236659d37d6af4fd138f405e846967
26-05-2021 02:34:51 : ArduCopter V4.1.0-dev (ab2e429f)
```

Figure 4.6: Mission Planner as GCS

4.2 Concept of MAVlink over internet using 4G LTE modem

In order to be able to communicate with drone from ground station using MAVlink, we want to utilize the internet over 4G LTE Modem to achieve unlimited range of communication link. However, to be able to establish point to point link over internet we need to be able to route data packets over internet. This is possible with below mentioned options:-

Option 1 If we have dedicated IP addresses over internet purchased from Internet Service Provider (ISP) for each component namely FCU, Companion computer, GCS and any other networked station. Then we can route packets over internet between the FCU (in our case its Beagle Bone Blue) to GCS (in our case Mission Planner) either directly using MAVlink or indirectly through companion computer using MAVPROXY. MAVPROXY is a command line GCS and we will see the configuration of companion computer in later chapters. The advantage of this setup is its very simple and anybody can implement it. The disadvantage is that this setup requires purchase of IP Addresses which has monetary implications, the MAVlink is not encrypted and this setup results in dependency on a particular internet service provider.

Option II In the second option we have two variations. One is to use Dynamic DNS and a dedicated VPN Server to create own private network over Internet. This ensures encryption (MAVlink security) but requires dedicated infrastructure. The second variation of this option is to utilize VPN from a service provider through cloud-based services. i.e VPN as a service over cloud. This option allows the benefit of VPN (encrypted MAVlink between Drone and Ground Control Station) without any dedicated infrastructure needed to implement. Hence, I chose this option to implement my setup or rather demonstrate this proof of concept. I am utilizing cloud-based VPN services offered by OpenVPN. This website allows a registered user to create as many VPN clients but simultaneous use is restricted to three free clients. Beyond three one has to pay to avail services. The setup utilized by me is depicted in figure 4.7 . The X Type quadcopter was used to demonstrate the small drone implementation while V Type quadcopter and Hexacopter were utilized to demonstrate medium/large drone setup to validate the proof of concept.

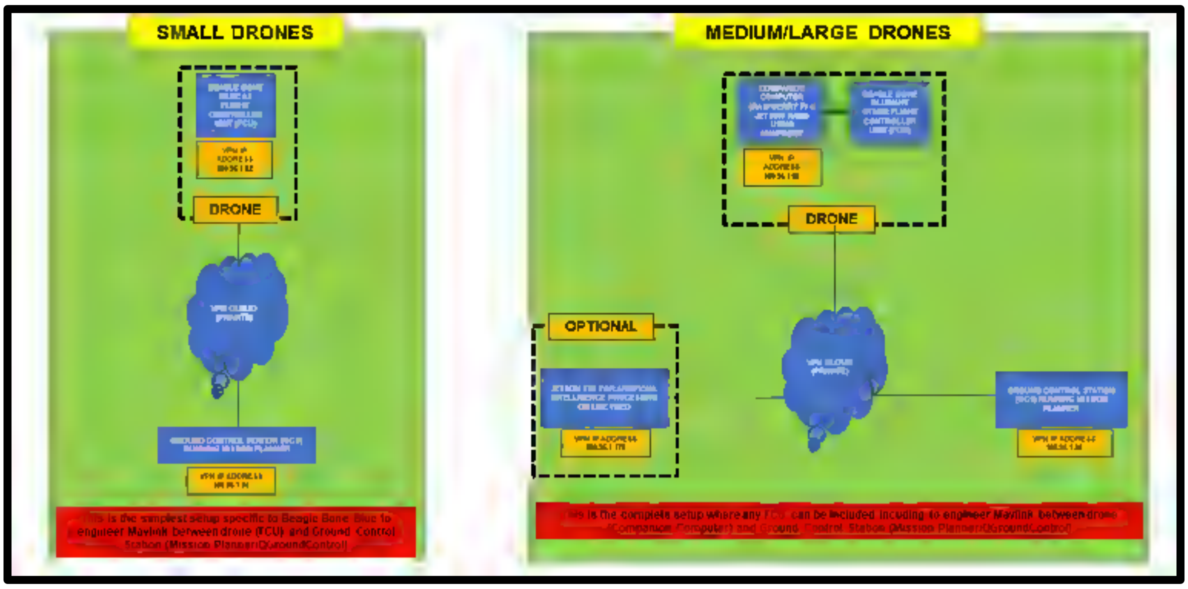
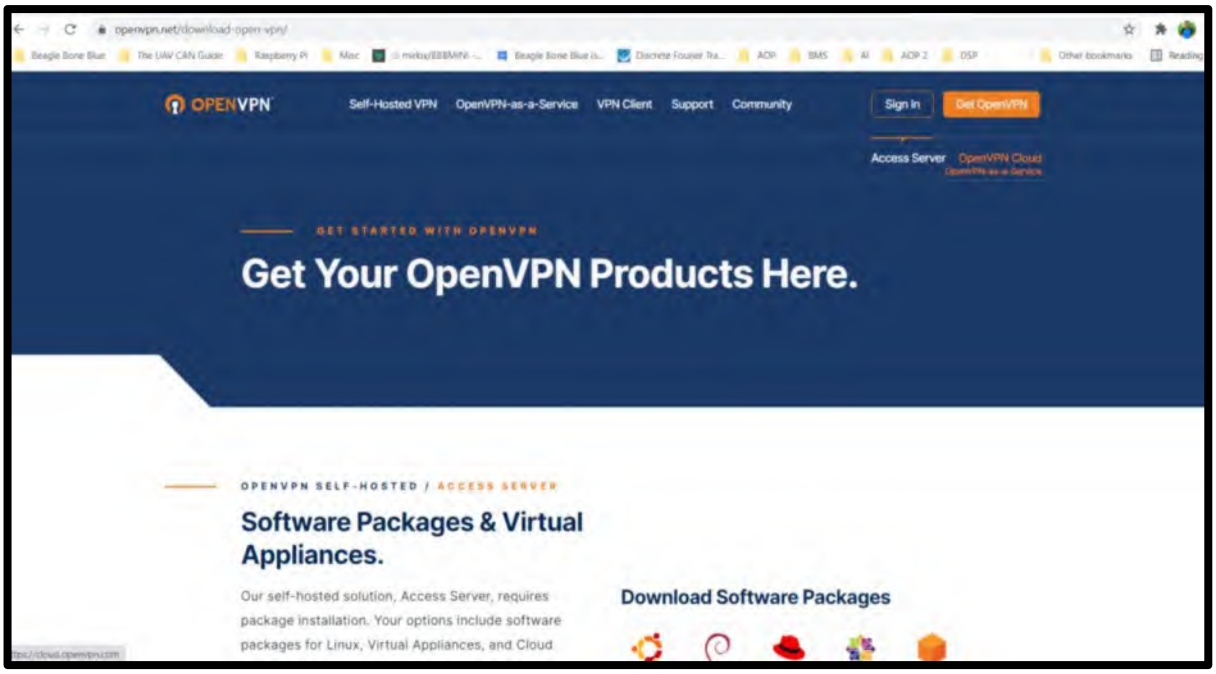

SMALL DRONES
DRONE
This is the simplest setup specific to Beagle Bone Blue to engineer Mavlink between drone (FCU) and Ground Control Station (Mission Planner/QGroundControl)
MEDIUM/LARGE DRONES
DRONE
OPTIONAL
This is the complete setup where any FCU can be included including to engineer Mavlink between drone (Companion Computer) and Ground Control Station (Mission Planner/QGroundControl)

Figure 4.7: Private VPN network created through OPENVPN Cloud based services

Step 1 Creating profile on OpenVPN cloud website https://myaccount.openvpn.com/ using email address.



GET STARTED WITH OPENVPN

Get Your OpenVPN Products Here.

OPENVPN SELF-HOSTED / ACCESS SERVER

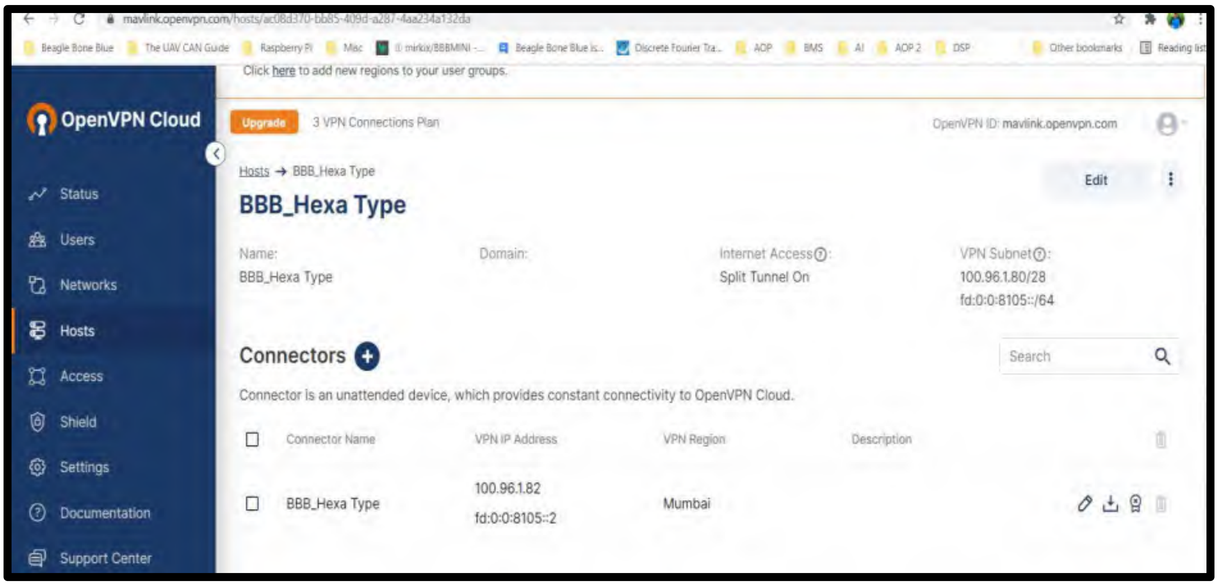
Software Packages & Virtual Appliances.

Our self-hosted solution, Access Server, requires package installation. Your options include software packages for Linux, Virtual Appliances, and Cloud

Download Software Packages

Figure 4.8: Creating my profile on OPENVPN portal for creating private VPN network

Step 2 Create host profile like the one below for each of the client i.e Beagle Bone Blue based Flight Controller Unit (FCU), ground station, one for companion computer and one for Ground Control Station.



Click here to add new regions to your user groups.

3 VPN Connections Plan

OpenVPN ID: mavlink.openvpn.com

Hosts → BBB_Hexa Type

BBB_Hexa Type

Name: BBB_Hexa Type

Domain:

Internet Access: Split Tunnel On

VPN Subnet: 100.96.1.80/28 fd:0:0:8105::/64

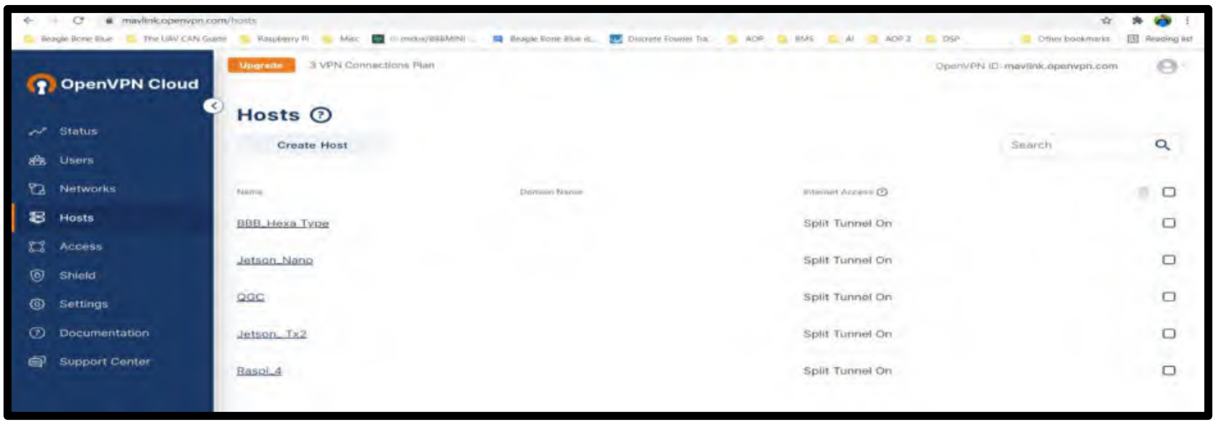
Connectors

Connector is an unattended device, which provides constant connectivity to OpenVPN Cloud.

Connector Name	VPN IP Address	VPN Region	Description
BBB_Hexa Type	100.96.1.82 fd:0:0:8105::2	Mumbai	

Figure 4.9: Creating VPN profiles on OPENVPN portal for various networking components i.e FCU, companion computer, GCS and Jetson TX2

Step 3 Download the profile in .ovpn format for each client profile on your machine. I have created a VPN Folder where I have downloaded all the .ovpn files. These files give all the configuration details for the device to auto connect with VPN cloud.



Hosts

Name	Domain Name	Internet Access
BBB_Hexa Type		Split Tunnel On
Jetson_Nano		Split Tunnel On
QGC		Split Tunnel On
Jetson_Tx2		Split Tunnel On
Raspi_4		Split Tunnel On

Figure 4.10: List of VPN Clients created

4.3 Installation 4G LTE communication modem and running VPN as a service on Beagle Bone Blue (BBB) FCU for small drones

4G LTE Modem integration with Beagle Bone Blue Flight Controller Unit (FCU)

Preparing Beagle Bone Blue to integrate with 4G LTE Modem over USB Connection.

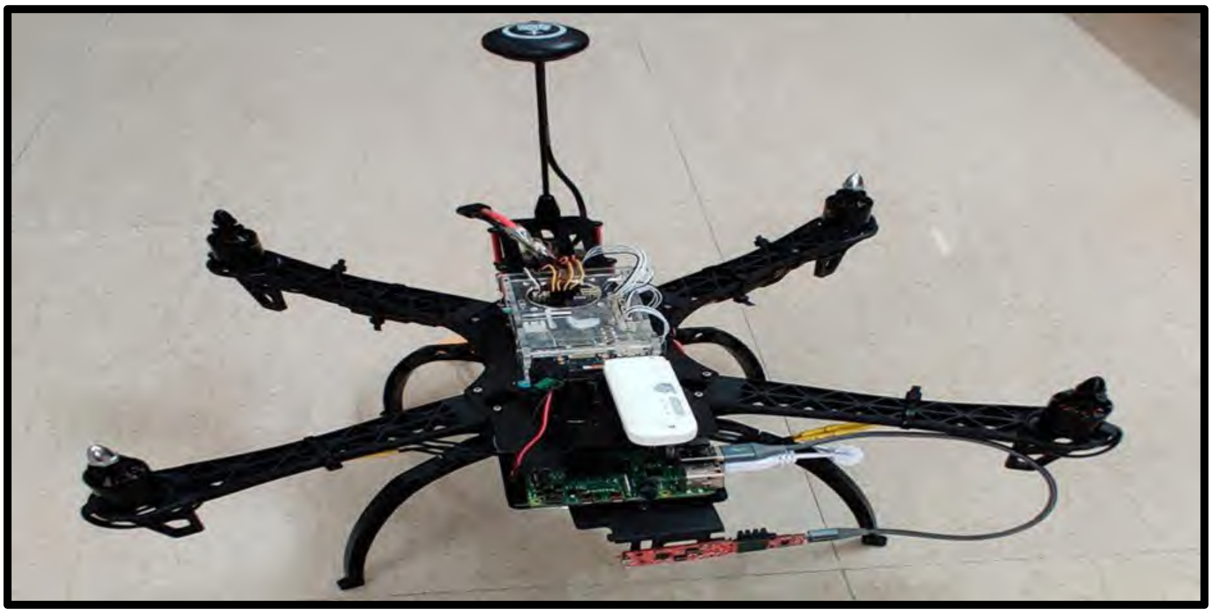
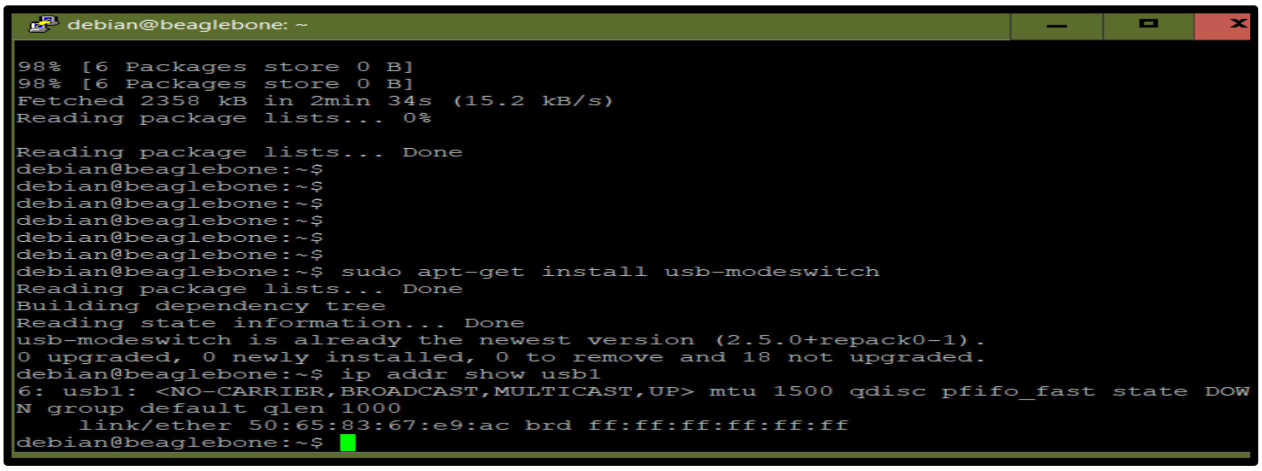


Figure 4.11: V-Type drone with 4G LTE Modem

Step 1 We need to prepare Beagle Bone Blue for connecting to 4G USB based dongle for internet access. Beagle Bone Blue does not connect to internet through 4G module via USB port but wifi as default option. But I don't trust wifi of Beagle Bone Blue, hence prefer to connect to internet via USB port. Hence, we need to do the below steps to enable that. This is one reason why in all drones irrespective of companion computer being present on drone or not, I have connected 4G Module to Beagle Bone Blue so that I can take remote ssh reliably at any time. Companion computer connected to 4G Module over wifi. We need usb mode switch to enable networking through usb port. Thereafter, we enable DHCP so that IP allotted by 4G LTE modem can be utilized by Beagle Bone Blue.

```
sudo apt-get update
sudo apt-get install usb-modeswitch
ip addr show usb1
```



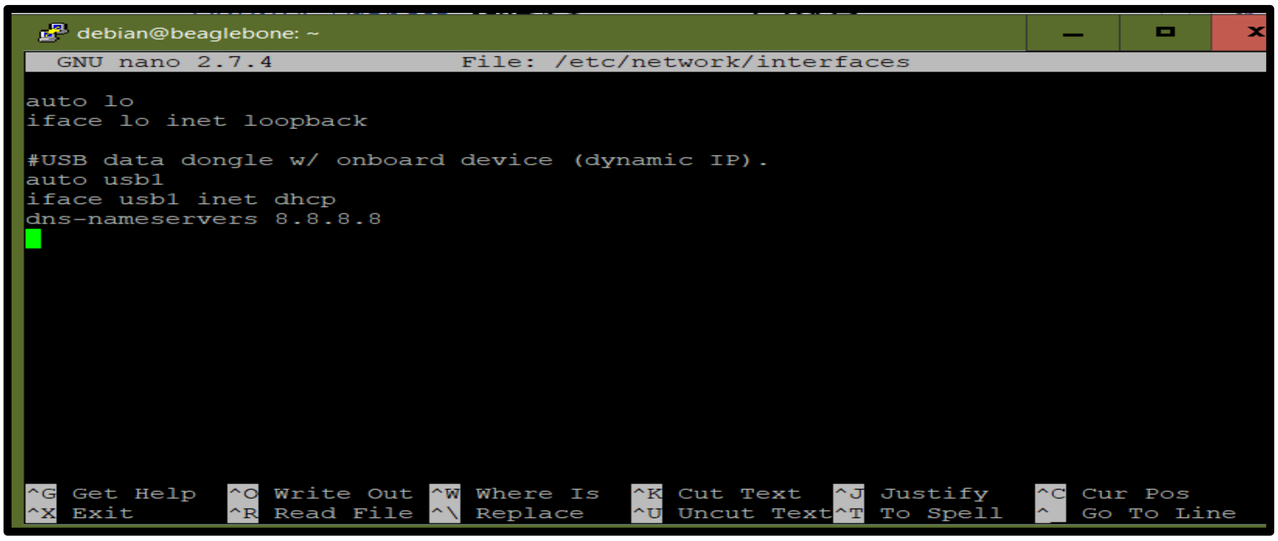
```
98% [6 Packages store 0 B]
98% [6 Packages store 0 B]
Fetched 2358 kB in 2min 34s (15.2 kB/s)
Reading package lists... 0%

Reading package lists... Done
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$ sudo apt-get install usb-modeswitch
Reading package lists... Done
Building dependency tree
Reading state information... Done
usb-modeswitch is already the newest version (2.5.0+repack0-1).
0 upgraded, 0 newly installed, 0 to remove and 18 not upgraded.
debian@beaglebone:~$ ip addr show usb1
6: usb1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOW
N group default qlen 1000
    link/ether 50:65:83:67:e9:ac brd ff:ff:ff:ff:ff:ff
debian@beaglebone:~$
```

Figure 4.12: Checking IP Address allotted on Eth (USB) to Beagle Bone Blue

```
sudo nano /etc/network/interfaces
```

Add the following lines

```
#USB data dongle w/ onboard device (dynamic IP).
auto usb1
iface usb1 inet dhcp
dns-nameservers 8.8.8.8
```



```
debian@beaglebone: ~
  GNU nano 2.7.4              File: /etc/network/interfaces

auto lo
iface lo inet loopback

#USB data dongle w/ onboard device (dynamic IP).
auto usb1
iface usb1 inet dhcp
dns-nameservers 8.8.8.8

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

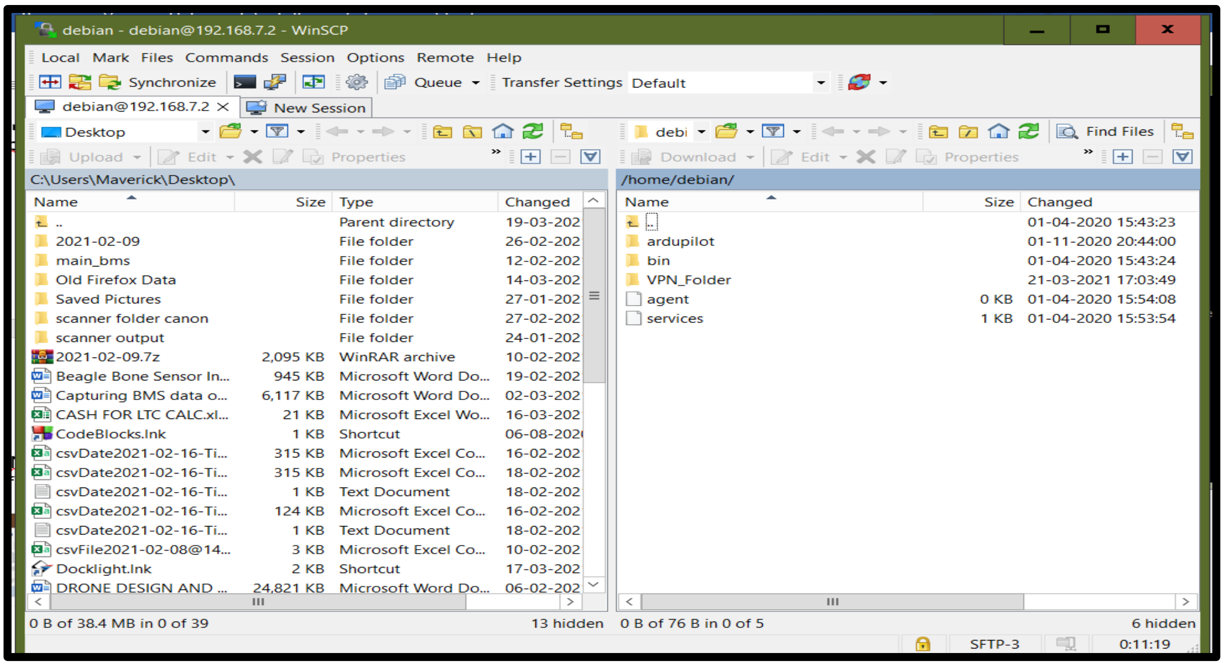
Figure 4.13: Enabling DHCP in /etc/network/interfaces

Ctrl+O and then Ctrl+X. The config gets saved. Now Beagle Bone Blue can connect to internet via USB 4G LTE modem connected to it directly through USB port.

Step 2 Install Open VPN application on Beagle Bone Blue This step is required incase we want Beagle Bone Blue to connect to Ground Control Station (GCS) directly without companion computer. This is recommended for smaller drones only. In.order to establish MAVlink between Beagle Blue as FCU and Mission Planner as GCS over internet using VPN we need to install OpenVPN on Beagle Bone Blue. I do this step irrespective to be able to take remote ssh of Beagle Bone Blue over internet for any remote configuration check or troubleshooting.

sudo apt-get install openvpn

Use WinSCP application on windows to transport the folder VPN_Folder that contains all my .ovpn files. Let me remind you that the VPN_Folder contains .ovpn files for FCU, Companion computer and GCS. I only need .ovpn file that I created for Beagle Bone Blue from the folder. However, I am transporting the entire folder just for the ease of it. I will do the same process for Companion computer too later in the document to maintain uniformity. Please note the relevant file in VPN_Folder for me is **bbbhexa_ssh.ovpn** which I renamed after download from OpenVPN website as above. The folder was first copied to location /home/debian being default directory as shown below:-


debian - debian@192.168.7.2 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
debian@192.168.7.2
New Session
Desktop
Upload
Edit
Properties
debi
Find Files
Download
Edit
Properties

C:\Users\Maverick\Desktop\

Name	Size	Type	Changed
..		Parent directory	19-03-202
2021-02-09		File folder	26-02-202
main_bms		File folder	12-02-202
Old Firefox Data		File folder	14-03-202
Saved Pictures		File folder	27-01-202
scanner folder canon		File folder	27-02-202
scanner output		File folder	24-01-202
2021-02-09.7z	2,095 KB	WinRAR archive	10-02-202
Beagle Bone Sensor In...	945 KB	Microsoft Word Do...	19-02-202
Capturing BMS data o...	6,117 KB	Microsoft Word Do...	02-03-202
CASH FOR LTC CALC.xl...	21 KB	Microsoft Excel Wo...	16-03-202
CodeBlocks.lnk	1 KB	Shortcut	06-08-202
csvDate2021-02-16-Ti...	315 KB	Microsoft Excel Co...	16-02-202
csvDate2021-02-16-Ti...	315 KB	Microsoft Excel Co...	18-02-202
csvDate2021-02-16-Ti...	1 KB	Text Document	18-02-202
csvDate2021-02-16-Ti...	124 KB	Microsoft Excel Co...	16-02-202
csvDate2021-02-16-Ti...	1 KB	Text Document	18-02-202
csvFile2021-02-08@14...	3 KB	Microsoft Excel Co...	10-02-202
Docklight.lnk	2 KB	Shortcut	17-03-202
DRONE DESIGN AND ...	24,821 KB	Microsoft Word Do...	06-02-202

0 B of 38.4 MB in 0 of 39 13 hidden

/home/debian/

Name	Size	Changed
..		01-04-2020 15:43:23
ardupilot		01-11-2020 20:44:00
bin		01-04-2020 15:43:24
VPN_Folder		21-03-2021 17:03:49
agent	0 KB	01-04-2020 15:54:08
services	1 KB	01-04-2020 15:53:54

0 B of 76 B in 0 of 5 6 hidden

SFTP-3 0:11:19

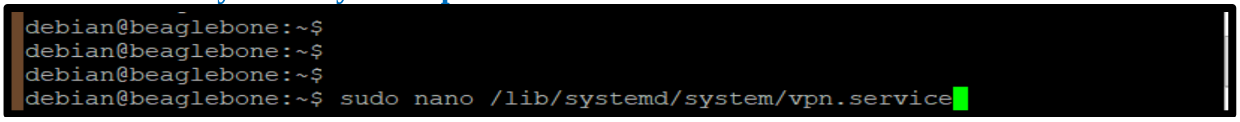
Figure 4.14: Using WinSCP to port VPN_Folder from Windows Laptop to default directory location of Debian Linux (Beagle Bone Blue)

Now move the folder from /home/debian to /etc/openvpn for execution

sudo mv /home/debian/VPN_Folder /etc/openvpn/

Step 3 Create systemd file for startup of VPN services at boot

sudo nano /lib/systemd/system/vpn.service



```
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$
debian@beaglebone:~$ sudo nano /lib/systemd/system/vpn.service
```

Figure 4.15: Using nano through remote ssh console to create vpn.service

Add the following lines

[Unit]

Description=VPN Service

After=networking.service

StartLimitIntervalSec=0

[Service]

Type=idle

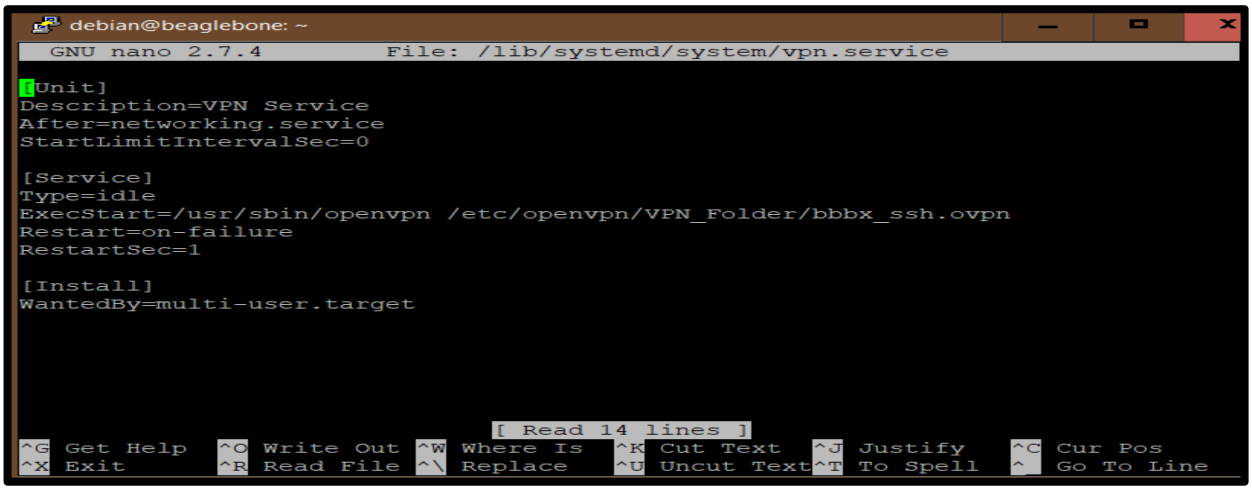
ExecStart=/usr/sbin/openvpn /etc/openvpn/VPN_Folder/bbbhexa_ssh.ovpn

Restart=on-failure

RestartSec=1

[Install]

WantedBy=multi-user.target



```
debian@beaglebone: ~
  GNU nano 2.7.4          File: /lib/systemd/system/vpn.service

[Unit]
Description=VPN Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
ExecStart=/usr/sbin/openvpn /etc/openvpn/VPN_Folder/bbbx_ssh.ovpn
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target

                              [ Read 14 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^T To Spell    ^  Go To Line
```

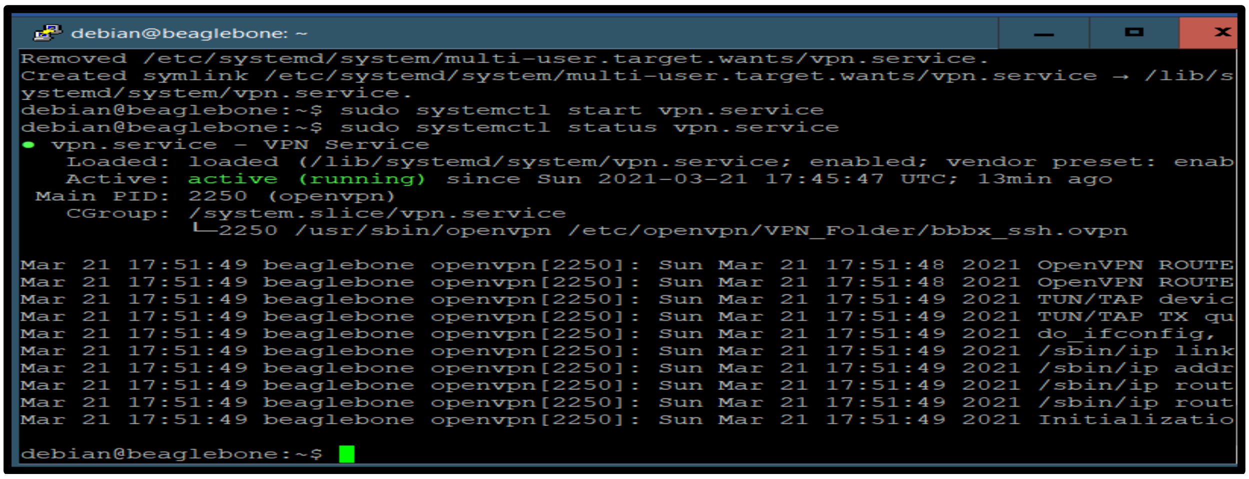
Figure 4.16: Creating vpn.service file at /lib/systemd/system directory location for VPN services to start at startup

sudo systemctl enable vpn.service

sudo systemctl start vpn.service

sudo systemctl status vpn.service

Now Beagle Bone Blue will connect to internet through 4G LTE Modem on startup and will be accessible over internet over VPN network. We need to find the IP Address of Beagle Bone Blue allotted to it over VPN Network. This will remain the defacto…………….. IP Address for taking ssh over internet or connecting to Ground Control Station (GCS) like mission planner explained later.



```
debian@beaglebone: ~
Removed /etc/systemd/system/multi-user.target.wants/vpn.service.
Created symlink /etc/systemd/system/multi-user.target.wants/vpn.service → /lib/s
ystemd/system/vpn.service.
debian@beaglebone:~$ sudo systemctl start vpn.service
debian@beaglebone:~$ sudo systemctl status vpn.service
● vpn.service - VPN Service
   Loaded: loaded (/lib/systemd/system/vpn.service; enabled; vendor preset: enab
   Active: active (running) since Sun 2021-03-21 17:45:47 UTC; 13min ago
 Main PID: 2250 (openvpn)
   CGroup: /system.slice/vpn.service
           └─2250 /usr/sbin/openvpn /etc/openvpn/VPN_Folder/bbbx_ssh.ovpn

Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:48 2021 OpenVPN ROUTE
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:48 2021 OpenVPN ROUTE
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:49 2021 TUN/TAP devic
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:49 2021 TUN/TAP TX qu
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:49 2021 do_ifconfig,
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:49 2021 /sbin/ip link
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:49 2021 /sbin/ip addr
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:49 2021 /sbin/ip rout
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:49 2021 /sbin/ip rout
Mar 21 17:51:49 beaglebone openvpn[2250]: Sun Mar 21 17:51:49 2021 Initializatio

debian@beaglebone:~$
```

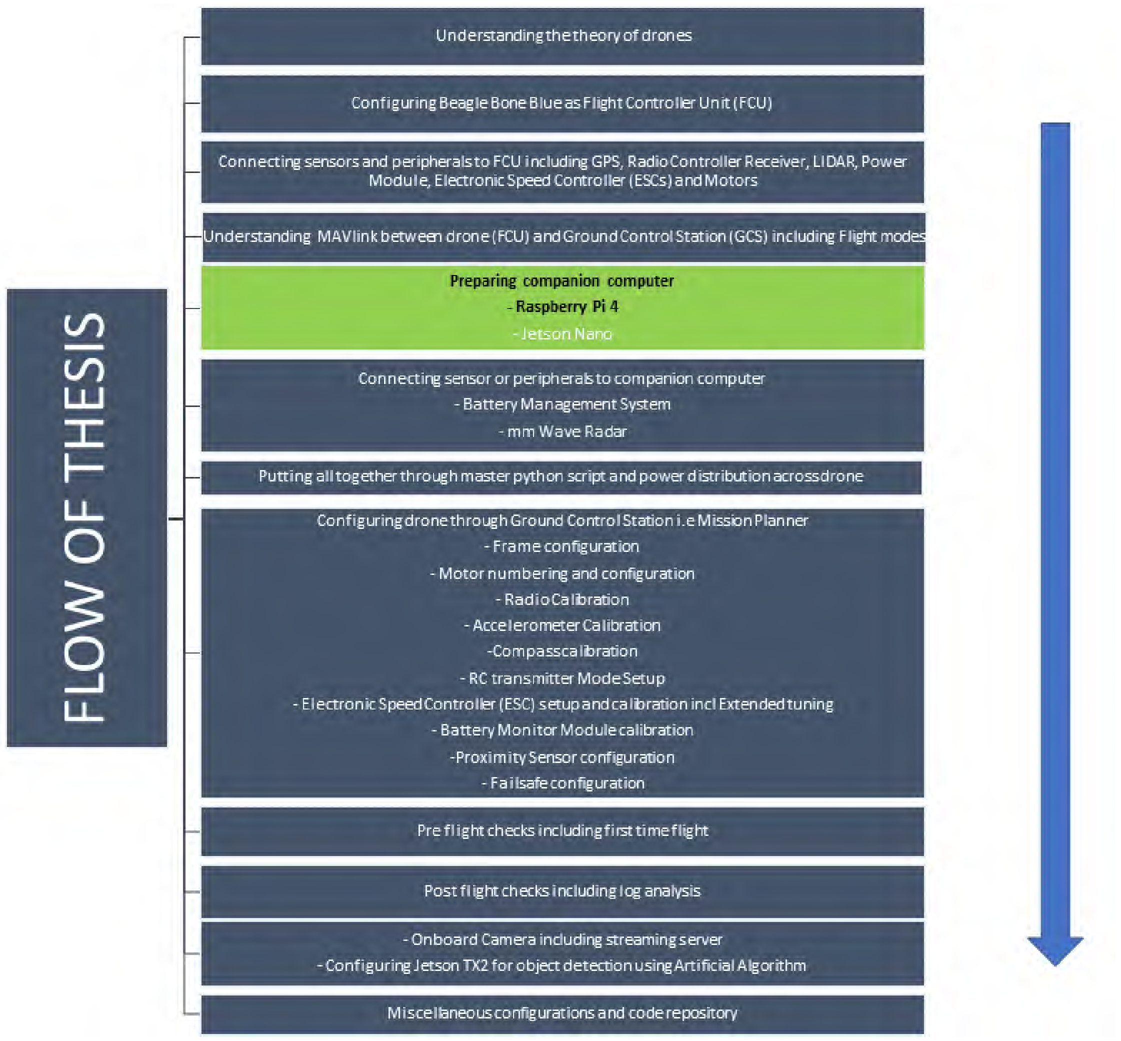
Figure 4.17: Enabling, starting and checking status of vpn.service. The service started successfully

sudo systemctl daemon-reload (only to reload service incase some changes are made)

Its important to note that VPN as a service on Beagle Bone Blue is required only in small drones where Beagle Bone Blue which is the FCU also acts as onboard computer running Linux and establishes MAVlink with GCS. However, incase of medium and larger drones this work of MAVlink over VPN is done by Companion computers like Raspberry Pi 4 or Jetson Nano which will be explained in subsequent chapters. However, if we have a VPN profile to spare, its always a good idea to install VPN on Beagle Bone Blue too so that remote ssh can be taken to troubleshoot any errors on FCU. Also, the choice of 4G LTE Modem location is totally on user. I have proposed the idea of 4G LTE Modem being connected to USB port of Beagle Bone Blue meaning the modem gets powered by Beagle Bone Blue and internet is accessed over USB. The companion computer utilizes the wifi to connect to internet. You can always reverse the order of plugging 4G LTE Modem to companion computer and Beagle Bone Blue can connect to internet on wifi (medium or large drones). The choice is entirely on user. However, from experience I find the wifi reliability of Companion computer is much better than that of Beagle Bone Blue, hence I connect 4G LTE Modem to Beagle Bone Blue

CHAPTER 5

RASPBERRY PI 4 AS COMPANION COMPUTER


FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server
- Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

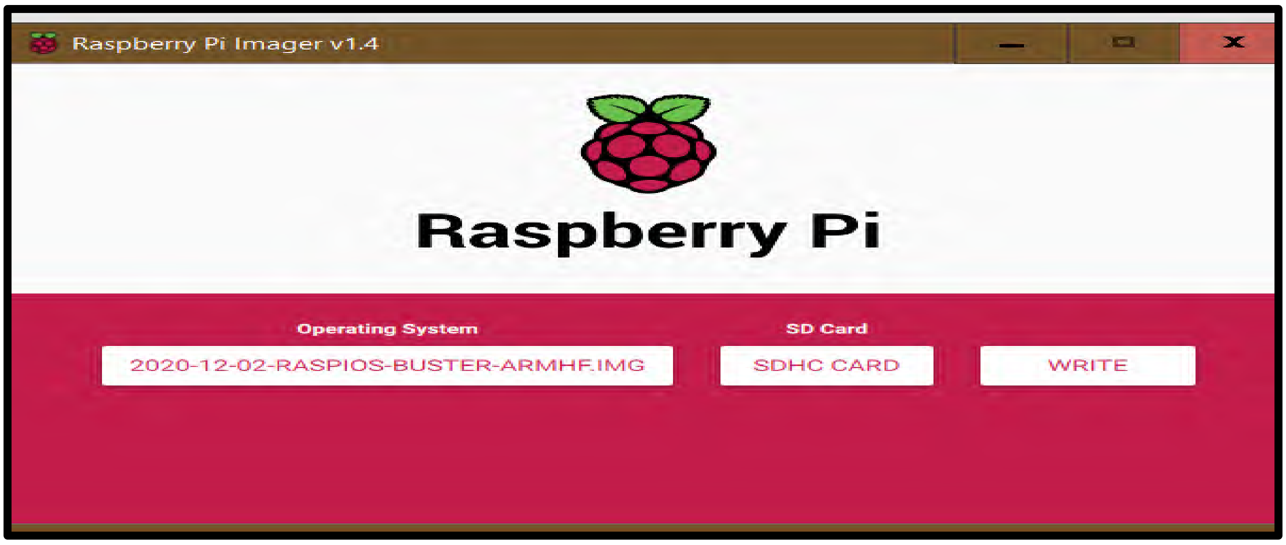
5.1: Flow of thesis until now

Companion Computers can be used to interface and communicate with ArduPilot on a flight controller unit (FCU) using the MAVLink protocol. By doing this the companion computer gets all the MAVLink data produced by the autopilot (including GPS data) and can use it to make intelligent decisions during flight and reduce computing load over FCU. In this chapter we will configure Raspberry Pi 4 as companion computer. The Raspberry Pi 4 is the latest product in the Raspberry Pi range, boasting an updated 64-bit quad core processor running at 1.4GHz with built-in metal heatsink, USB 3 ports, dual-band 2.4GHz and 5GHz wireless LAN, faster (300 Mbps) Ethernet, and PoE capability via a separate PoE HAT. Some of the features are as below-

- 4GB RAM
- Quad-Core 64-bit Broadcom 2711, Cortex A72 Processor
- WLAN 802.11 b/g/n/ac (2,4 + 5,0 GHz)
- LAN RJ45 10/100/1000 Mbit (Gigabit LAN over USB 3.0)
- Operating Power 5V@3A via USB Type-C Port
- Dual-Display Micro HDMI Ports which supports H 265 Decode for 4K Video @60p

5.1 Preparing Raspberry Pi 4

Step 1 Download the latest **raspios-buster-armhf.img** from official site. Use pi imager or any other SD card flashing software like Balena Etcher to flash SD card of 8/16/32 Gb. I am using Raspberry Pi imager and windows laptop to flash the SD card with Raspbian OS (Linux OS)

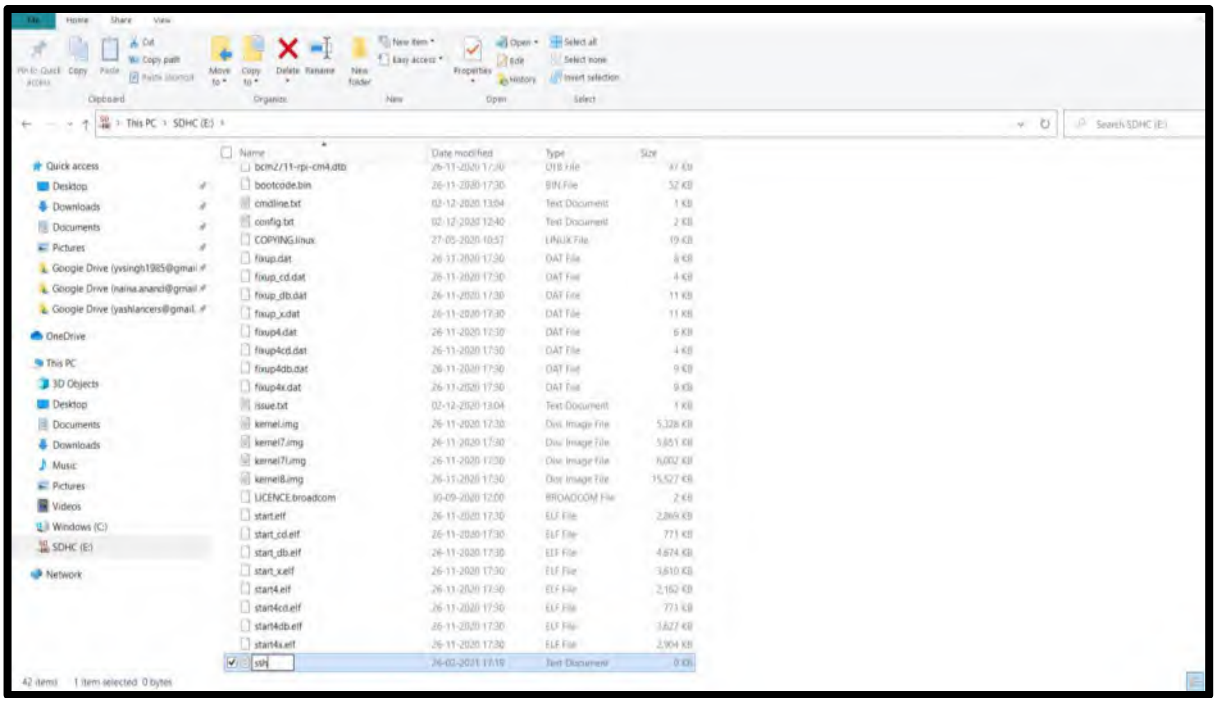


Operating System: 2020-12-02-RASPIOS-BUSTER-ARMHF.IMG

SD Card: SDHC CARD

Figure 5.2: Flashing SD card for Raspberry Pi 4 using Raspberry Pi Imager

Step 2 To access Raspberry Pi on ssh (headless), after flashing SD Card, open the directory and add a .txt file and rename it ssh without any file extension.



Name	Date modified	Type	Size
bcm2711-rpi-cm4.dtb	26-11-2020 17:30	DTB File	47 KB
bootcode.bin	26-11-2020 17:30	BIN File	52 KB
cmdline.txt	02-12-2020 13:04	Text Document	1 KB
config.txt	02-12-2020 12:40	Text Document	2 KB
COPYING.linux	27-05-2020 10:57	LINUX File	19 KB
fixup.dat	26-11-2020 17:30	DAT File	8 KB
fixup_cd.dat	26-11-2020 17:30	DAT File	4 KB
fixup_db.dat	26-11-2020 17:30	DAT File	11 KB
fixup_x.dat	26-11-2020 17:30	DAT File	11 KB
fixup4.dat	26-11-2020 17:30	DAT File	6 KB
fixup4cd.dat	26-11-2020 17:30	DAT File	4 KB
fixup4db.dat	26-11-2020 17:30	DAT File	9 KB
fixup4x.dat	26-11-2020 17:30	DAT File	9 KB
issue.txt	02-12-2020 13:04	Text Document	1 KB
kernel.img	26-11-2020 17:30	Disc Image File	5,328 KB
kernel7.img	26-11-2020 17:30	Disc Image File	5,651 KB
kernel7l.img	26-11-2020 17:30	Disc Image File	6,002 KB
kernel8.img	26-11-2020 17:30	Disc Image File	15,527 KB
LICENCE.broadcom	30-09-2020 12:00	BROADCOM File	2 KB
start.elf	26-11-2020 17:30	ELF File	2,869 KB
start_cd.elf	26-11-2020 17:30	ELF File	771 KB
start_db.elf	26-11-2020 17:30	ELF File	4,674 KB
start_x.elf	26-11-2020 17:30	ELF File	3,610 KB
start4.elf	26-11-2020 17:30	ELF File	2,162 KB
start4cd.elf	26-11-2020 17:30	ELF File	771 KB
start4db.elf	26-11-2020 17:30	ELF File	3,627 KB
start4x.elf	26-11-2020 17:30	ELF File	2,904 KB
ssh	26-02-2021 17:19	Text Document	0 KB

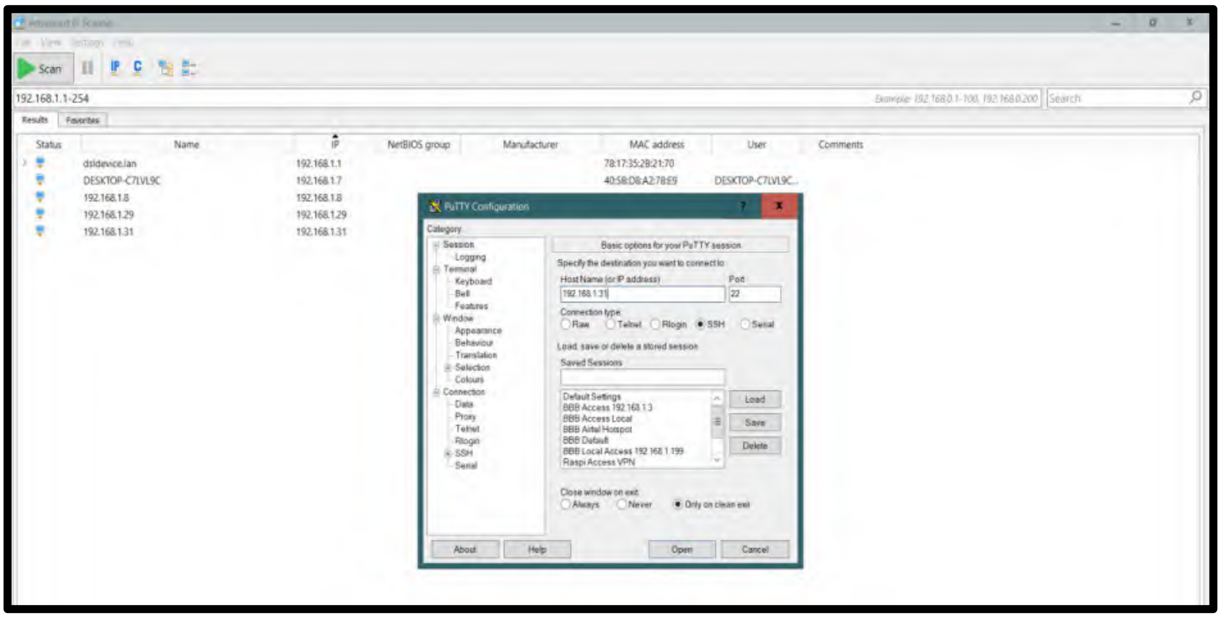
42 items 1 item selected 0 bytes

Figure 5.3: Adding ssh file with no extension to SD card for headless access to Raspberry Pi 4

Use this sd card to boot Raspi Pi 4

Identifying IP of Raspberry Pi 4 over local LAN for wireless ssh access

Step 1 Connect Raspberry Pi 4 on LAN through ethernet. Scan for its IP address using any IP Scanner tool to find IP allotted. Use that IP to access Raspberry Pi 4 on ssh using Putty client.



192.168.1.1-254

Status	Name	IP	NetBIOS group	Manufacturer	MAC address	User	Comments
	dsldevice.lan	192.168.1.1			78:17:35:2B:21:70		
	DESKTOP-C7LVL9C	192.168.1.7			40:58:D8:A2:78:E9	DESKTOP-C7LVL9C...	
	192.168.1.8	192.168.1.8					
	192.168.1.29	192.168.1.29					
	192.168.1.31	192.168.1.31					

PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.31

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions:
Default Settings
BBB Access 192.168.1.3
BBB Access Local
BBB Airtel Hotspot
BBB Default
BBB Local Access 192.168.1.199
Raspi Access VPN

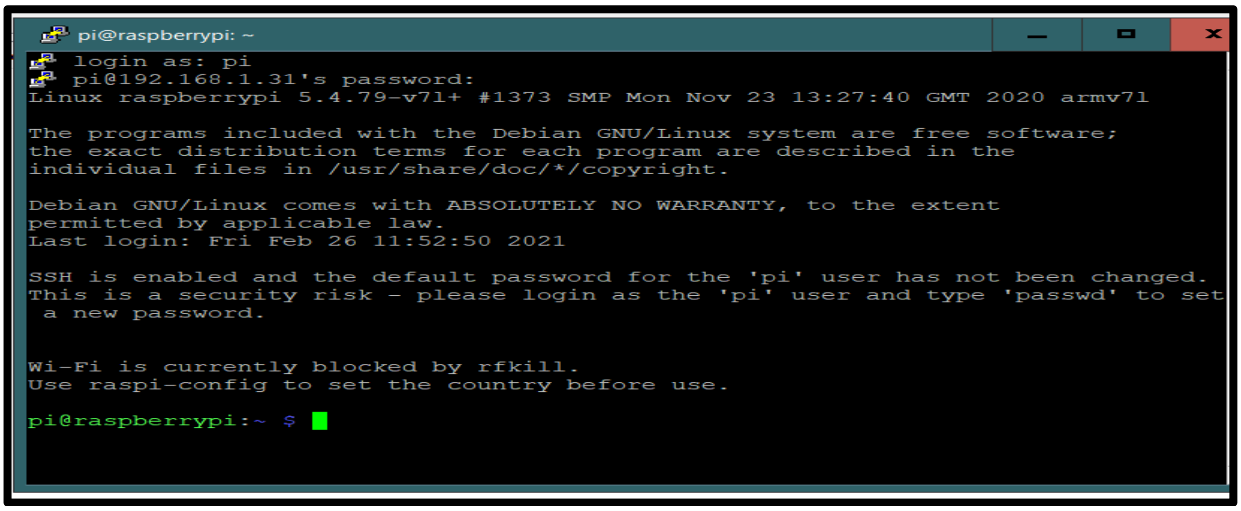
Close window on exit: Always Never [✓ Only on clean exit]

Figure 5.4: Using Advanced IP Scanner to detect IP Address allotted to Raspberry Pi 4 over LAN

Use that IP address to access Raspberry Pi 4 through Putty application

Default username- pi

Default password- raspberry



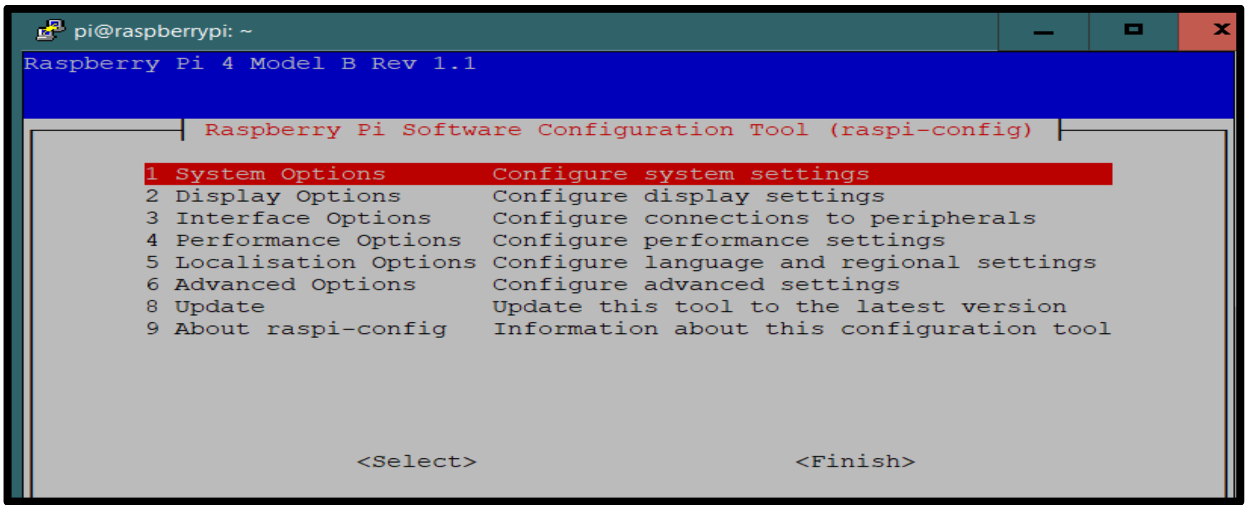
```
pi@raspberrypi: ~
login as: pi
pi@192.168.1.31's password:
Linux raspberrypi 5.4.79-v7l+ #1373 SMP Mon Nov 23 13:27:40 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Feb 26 11:52:50 2021

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.


Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.

pi@raspberrypi:~ $
```

Figure 5.5: Remote ssh access to Raspberry Pi 4 over local LAN

Step 2 Now we have been able to access Raspberry Pi 4 over LAN. Npw we p[rovide it with Wifi credentials to connect over Wifi in future. We need to access raspi-config to save Wifi credentials.

sudo raspi-config

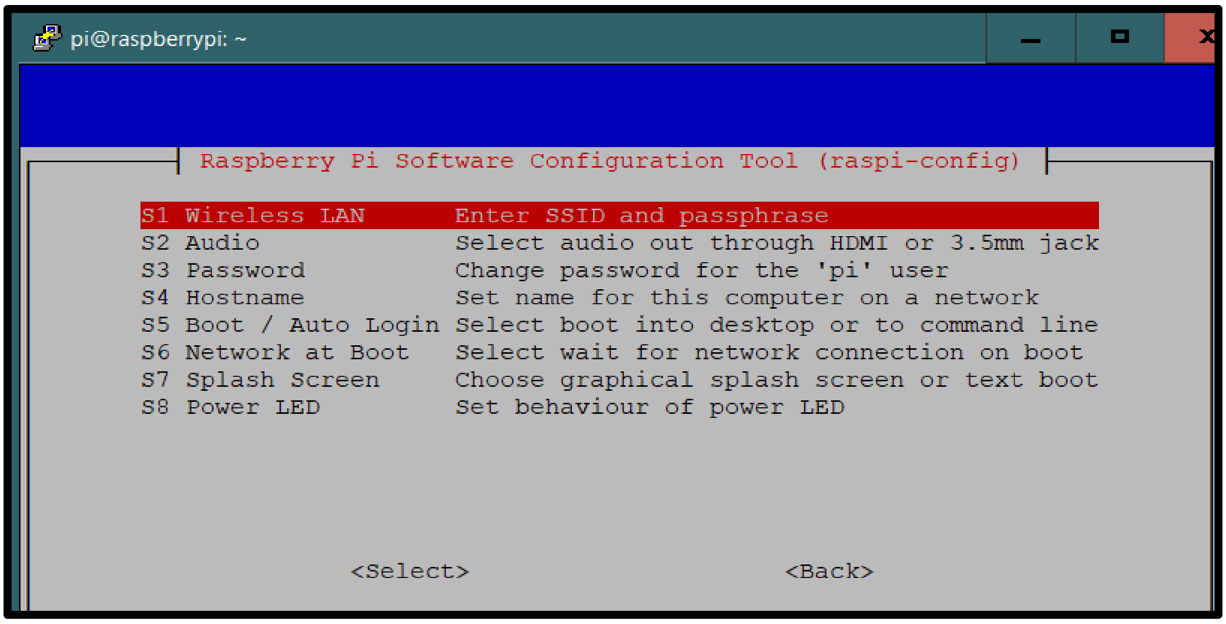
Step 3 Go to system options



```
pi@raspberrypi: ~
Raspberry Pi 4 Model B Rev 1.1

        Raspberry Pi Software Configuration Tool (raspi-config)

 1 System Options       Configure system settings
 2 Display Options      Configure display settings
 3 Interface Options    Configure connections to peripherals
 4 Performance Options  Configure performance settings
 5 Localisation Options Configure language and regional settings
 6 Advanced Options     Configure advanced settings
 8 Update               Update this tool to the latest version
 9 About raspi-config   Information about this configuration tool


              <Select>                     <Finish>
```

Figure 5.6: Select System Options

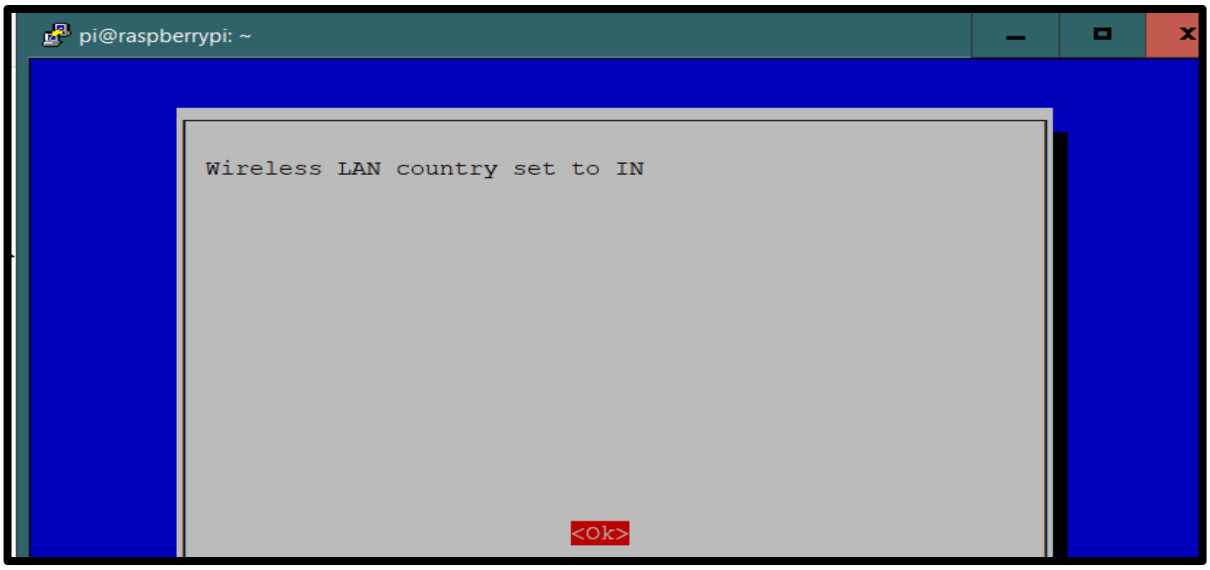
Step 4 Select wireless LAN



```
pi@raspberrypi: ~

Raspberry Pi Software Configuration Tool (raspi-config)

S1 Wireless LAN        Enter SSID and passphrase
S2 Audio               Select audio out through HDMI or 3.5mm jack
S3 Password            Change password for the 'pi' user
S4 Hostname            Set name for this computer on a network
S5 Boot / Auto Login   Select boot into desktop or to command line
S6 Network at Boot     Select wait for network connection on boot
S7 Splash Screen       Choose graphical splash screen or text boot
S8 Power LED           Set behaviour of power LED

              <Select>                  <Back>
```

Figure 5.7: Select Wireless LAN

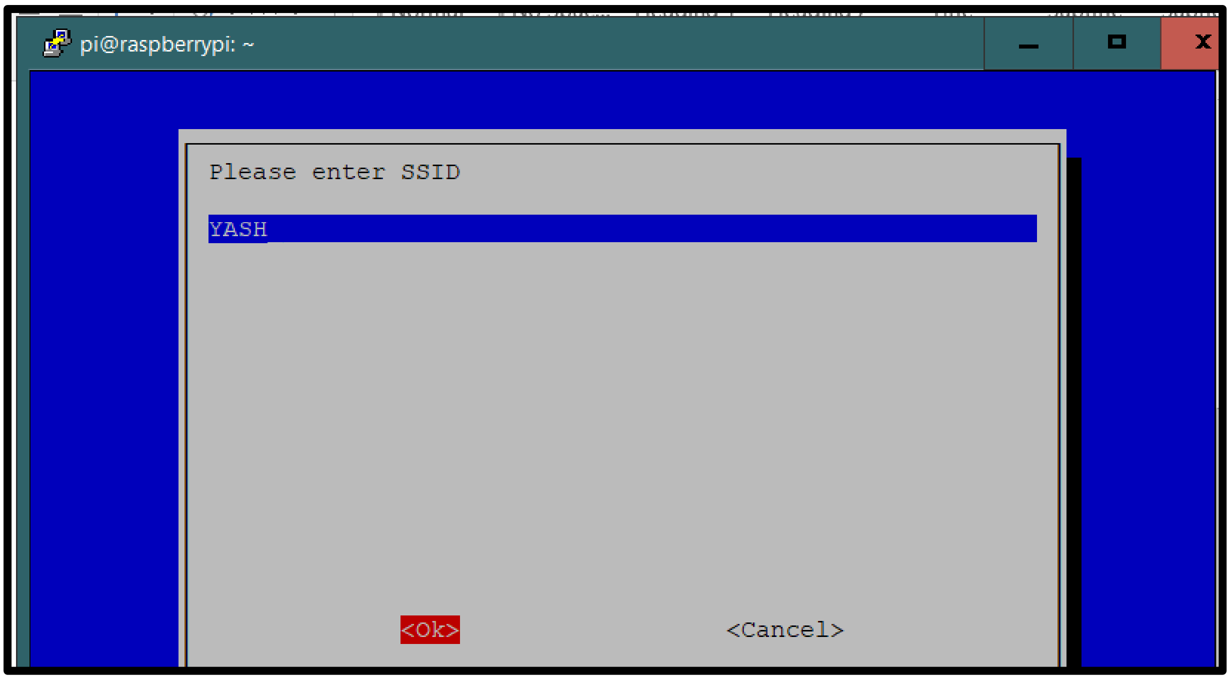
Select country to India



```
pi@raspberrypi: ~

Wireless LAN country set to IN

                    <Ok>
```

Figure 5.8: Select country option

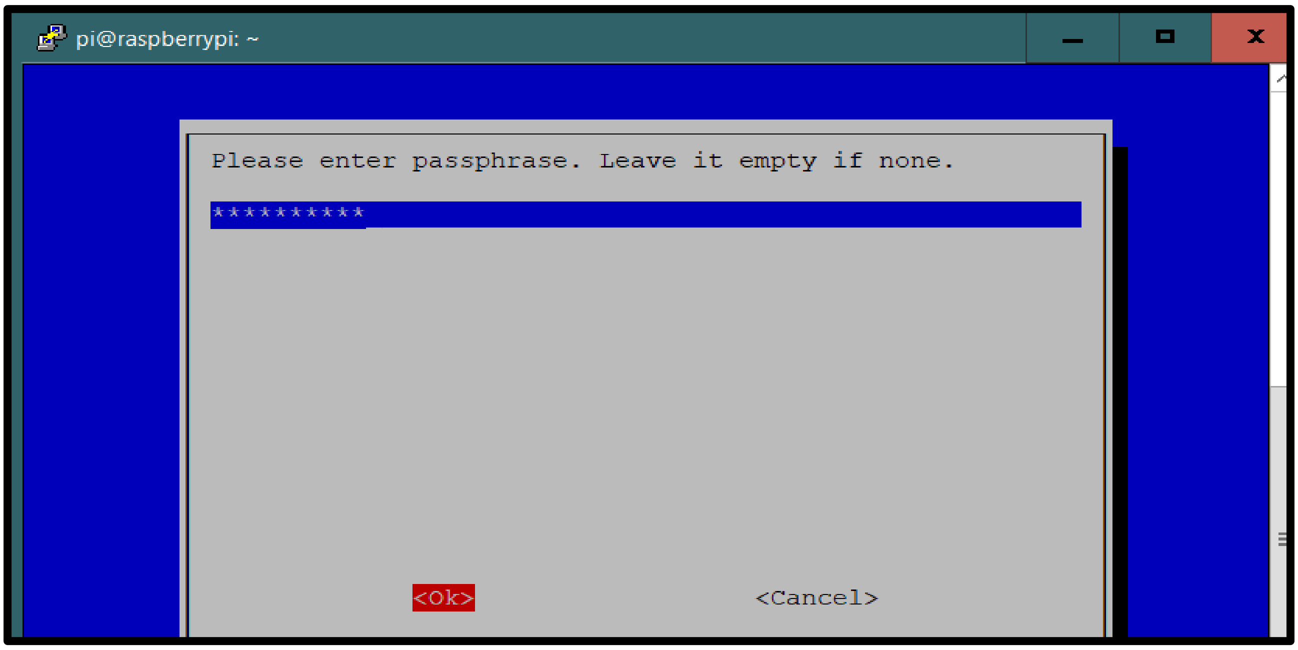
Enter SSID of wifi



```
Please enter SSID

YASH

<Ok>                    <Cancel>
```

Figure 5.9: Enter SSID option

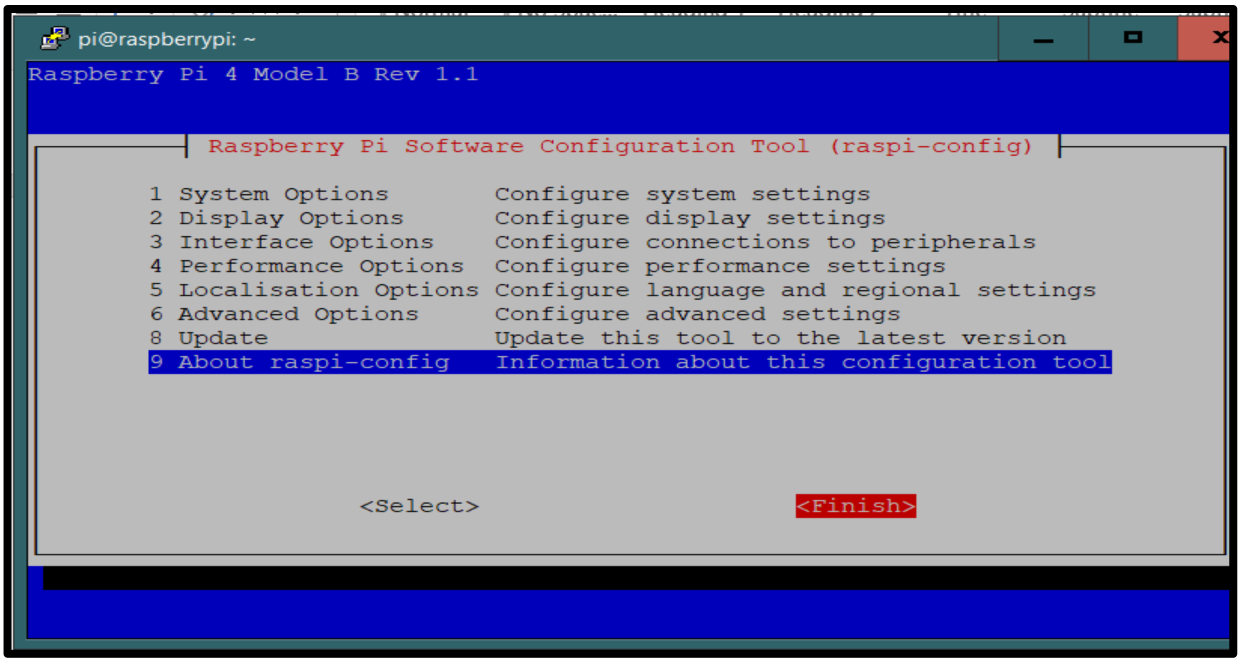
Enter password



```
Please enter passphrase. Leave it empty if none.

**********

<Ok>                    <Cancel>
```

Figure 5.10: Enabling, starting and checking status of vpn.service. The service started successfully

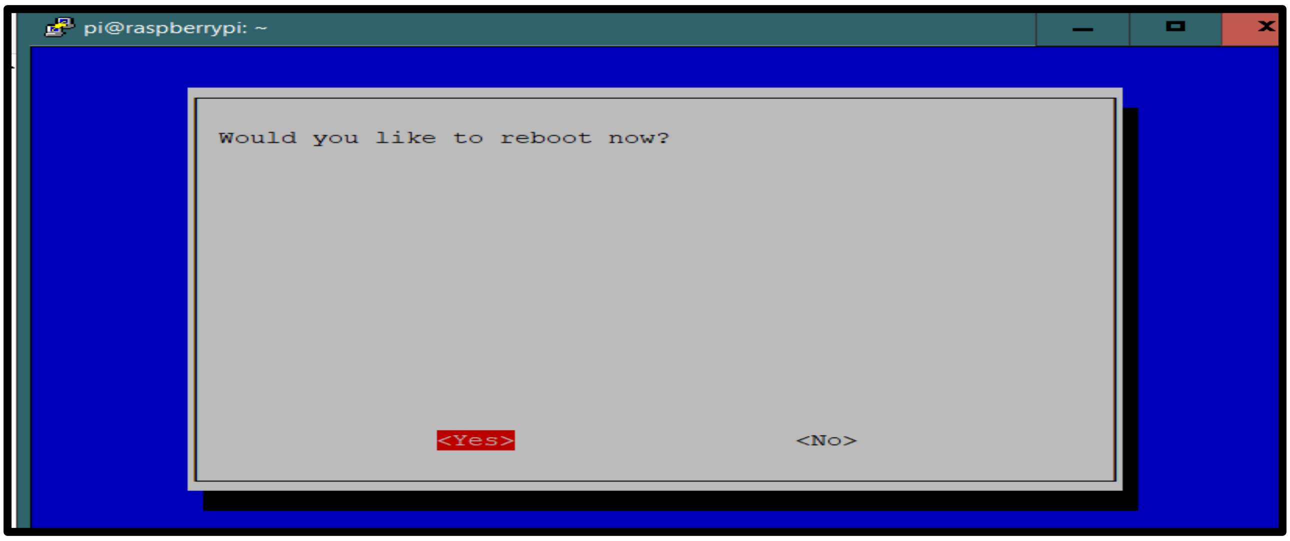
Select finish



```
pi@raspberrypi: ~
Raspberry Pi 4 Model B Rev 1.1

        Raspberry Pi Software Configuration Tool (raspi-config)

    1 System Options        Configure system settings
    2 Display Options       Configure display settings
    3 Interface Options     Configure connections to peripherals
    4 Performance Options   Configure performance settings
    5 Localisation Options  Configure language and regional settings
    6 Advanced Options      Configure advanced settings
    8 Update                Update this tool to the latest version
    9 About raspi-config    Information about this configuration tool

              <Select>                      <Finish>
```

Figure 5.11: Select Finish Option

Reboot system. The Raspberry Pi 4 will now connect on wifi.



```
pi@raspberrypi: ~

Would you like to reboot now?

        <Yes>                 <No>
```

Figure 5.12: Option to reboot now

Reboot the Raspberry Pi when you are done

sudo reboot

After reboot, Raspberry Pi 4 will connect to wifi. We need to find the IP Address by using any

IP Scanner tool and using Putty application take remote ssh for configuring Raspberry Pi 4.

Use below mentioned commands to update and upgrade Raspi OS.

```
sudo apt-get update
sudo apt-get upgrade
```

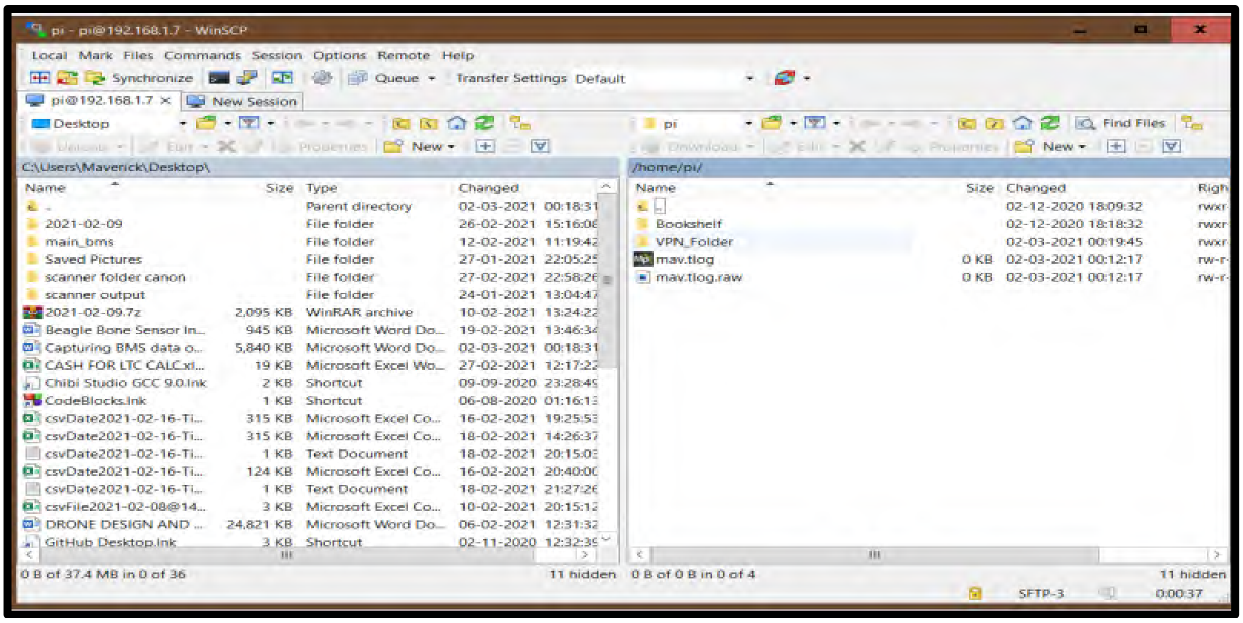
5.2 Installation of VPN as a service

As we want to establish MAVlink using UDP protocol over internet using 4G LTE Modem, we need VPN. To install VPN, the process is similar to the once explained above for Beagle Bone Blue in Chapter 4.

Step1 First install Open VPN application on Raspberry Pi 4

```
sudo apt-get install openvpn
```

Use WinSCP application on windows to transport the folder VPN_Folder that contains all my .ovpn files. Let me remind you that the VPN_Folder contains .ovpn files for FCU, Companion computer and Ground Control Station (GCS). I only need .ovpn file that I created for Raspberry Pi 4 from the folder. However, I am transporting the entire folder just for the ease of it. I have done the same process for Beagle Bone Blue above in Chapter 4. Please note the relevant file in VPN_Folder for me is **raspi_4.ovpn** which I renamed after download from OPENVPN website. The folder was first copied to location /home/pi being default directory as shown below:-


pi - pi@192.168.1.7 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
pi@192.168.1.7
New Session
Desktop
New
pi
Find Files
New

```
C:\Users\Maverick\Desktop\
Name                          Size  Type                  Changed
..                                  Parent directory      02-03-2021  00:18:31
2021-02-09                          File folder           26-02-2021  15:16:06
main_bms                            File folder           12-02-2021  11:19:42
Saved Pictures                      File folder           27-01-2021  22:05:25
scanner folder canon                File folder           27-02-2021  22:58:26
scanner output                      File folder           24-01-2021  13:04:47
2021-02-09.7z             2,095 KB  WinRAR archive        10-02-2021  13:24:22
Beagle Bone Sensor In...    945 KB  Microsoft Word Do...  19-02-2021  13:46:34
Capturing BMS data o...   5,840 KB  Microsoft Word Do...  02-03-2021  00:18:31
CASH FOR LTC CALC.xl...      19 KB  Microsoft Excel Wo... 27-02-2021  12:17:22
Chibi Studio GCC 9.0.lnk      2 KB  Shortcut              09-09-2020  23:28:49
CodeBlocks.lnk                1 KB  Shortcut              06-08-2020  01:16:13
csvDate2021-02-16-Ti...     315 KB  Microsoft Excel Co... 16-02-2021  19:25:53
csvDate2021-02-16-Ti...     315 KB  Microsoft Excel Co... 18-02-2021  14:26:37
csvDate2021-02-16-Ti...       1 KB  Text Document         18-02-2021  20:15:03
csvDate2021-02-16-Ti...     124 KB  Microsoft Excel Co... 16-02-2021  20:40:00
csvDate2021-02-16-Ti...       1 KB  Text Document         18-02-2021  21:27:26
csvFile2021-02-08@14...       3 KB  Microsoft Excel Co... 10-02-2021  20:15:12
DRONE DESIGN AND ...     24,821 KB  Microsoft Word Do...  06-02-2021  12:31:32
GitHub Desktop.lnk            3 KB  Shortcut              02-11-2020  12:32:39
0 B of 37.4 MB in 0 of 36                                 11 hidden
```

```
/home/pi/
Name                 Size  Changed              Righ
..                         02-12-2020 18:09:32  rwxr
Bookshelf                  02-12-2020 18:18:32  rwxr
VPN_Folder                 02-03-2021 00:19:45  rwxr
mav.tlog             0 KB  02-03-2021 00:12:17  rw-r
mav.tlog.raw         0 KB  02-03-2021 00:12:17  rw-r
0 B of 0 B in 0 of 4                            11 hidden
                              SFTP-3            0:00:37
```

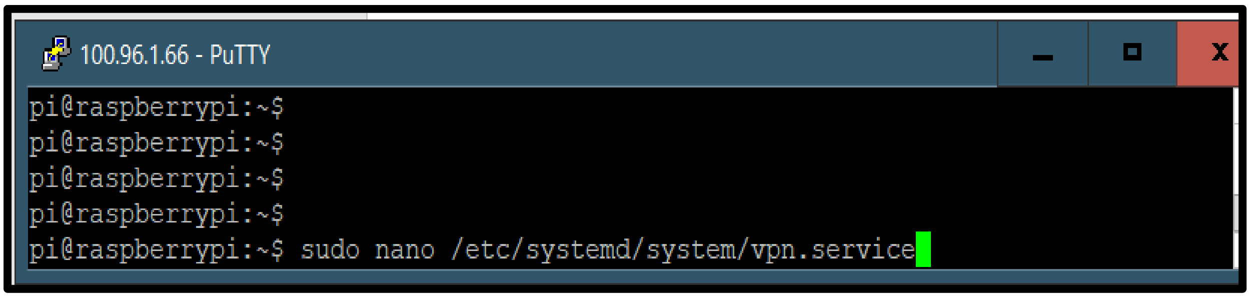
Figure 5.13: Using WinSCP to port VPN_Folder from windows Laptop to Default location in Raspberry Pi 4 OS

Now move the folder from /home/pi to /etc/openvpn for execution

sudo mv /home/pi/VPN_Folder /etc/openvpn/

Step 2 Create systemd file for startup at boot

sudo nano /etc/systemd/system/vpn.service



```
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo nano /etc/systemd/system/vpn.service
```

Figure 5.14: Creating vpn.service at /etc/systemd/system location

```
[Unit]
Description=VPN Service
After=networking.service
StartLimitIntervalSec=0
```

[Service]

Type=idle

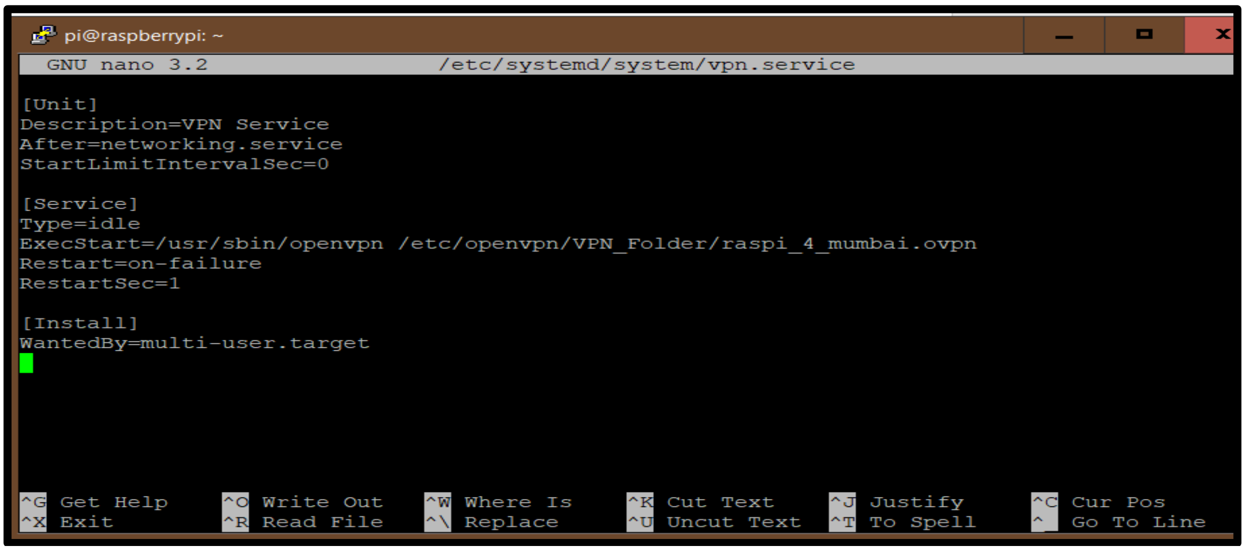
ExecStart=/usr/sbin/openvpn /etc/openvpn/VPN_Folder/raspi_4.ovpn

Restart=on-failure

RestartSec=1

[Install]

WantedBy=multi-user.target



```
  GNU nano 3.2                     /etc/systemd/system/vpn.service

[Unit]
Description=VPN Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
ExecStart=/usr/sbin/openvpn /etc/openvpn/VPN_Folder/raspi_4_mumbai.ovpn
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target
```

Figure 5.15: Creating content of vpn.service

sudo systemctl enable vpn.service

sudo systemctl start vpn.service

sudo systemctl status vpn.service

sudo systemctl daemon-reload (only to reload service incase some changes are made)

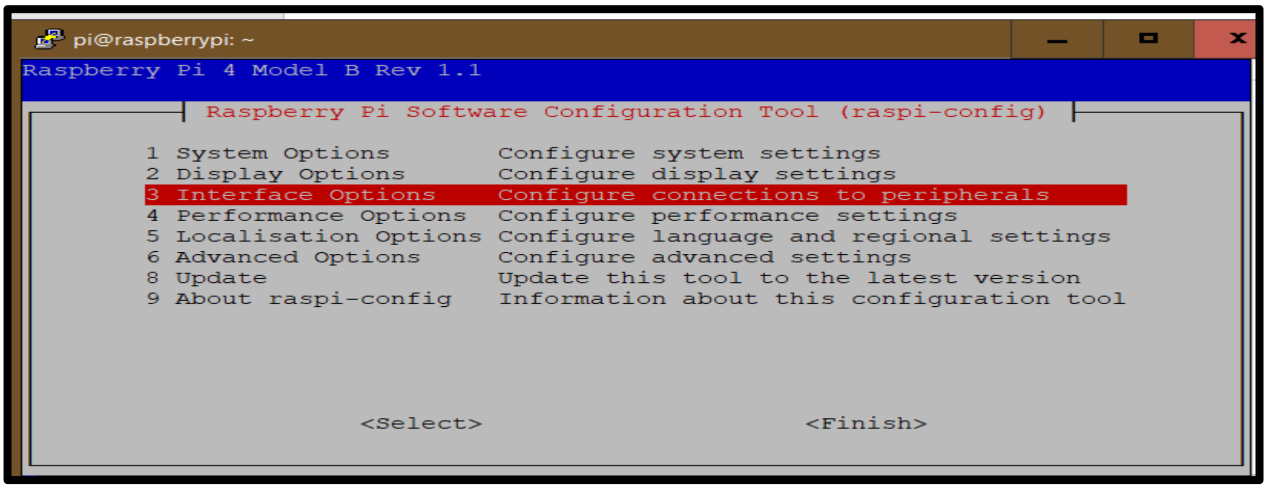
5.3 Installation of MAVPROXY (Command Line Ground Control Station) as a service

MAVPROXY is the command line Ground Control Station (GCS) to be installed in companion computer. In my case its Raspberry Pi 4. This MAVPROXY will receive MAVlink data (link between Ardupilot running on FCU i.e Beagle Bone Blue and companion computer) over serial interface and relay it back to the ground control station i.e Mission Planner running over laptop/PC. The MAVlink is installed over secure VPN using UDP protocol over internet connection extended by 4G LTE Modem.

Step 1 First disable SSH login on serial port

sudo raspi-config

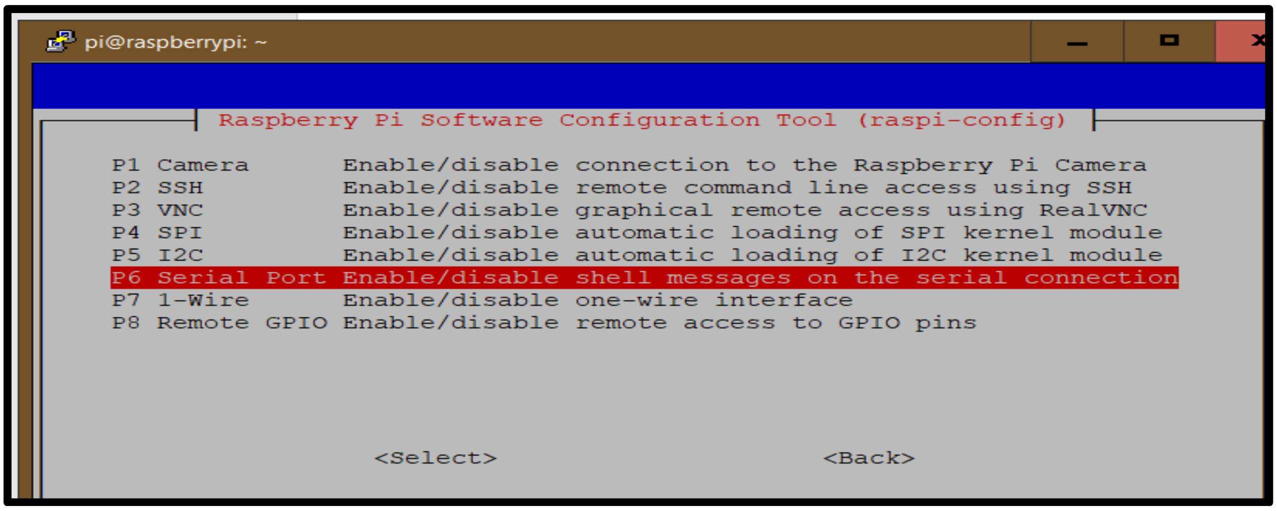
And in the utility, select "Interfacing Options":



```
pi@raspberrypi: ~
Raspberry Pi 4 Model B Rev 1.1
          Raspberry Pi Software Configuration Tool (raspi-config)

     1 System Options         Configure system settings
     2 Display Options        Configure display settings
     3 Interface Options      Configure connections to peripherals
     4 Performance Options    Configure performance settings
     5 Localisation Options   Configure language and regional settings
     6 Advanced Options       Configure advanced settings
     8 Update                 Update this tool to the latest version
     9 About raspi-config     Information about this configuration tool

                  <Select>                     <Finish>
```

Figure 5.16: Accessing Interfacing options through raspi-config

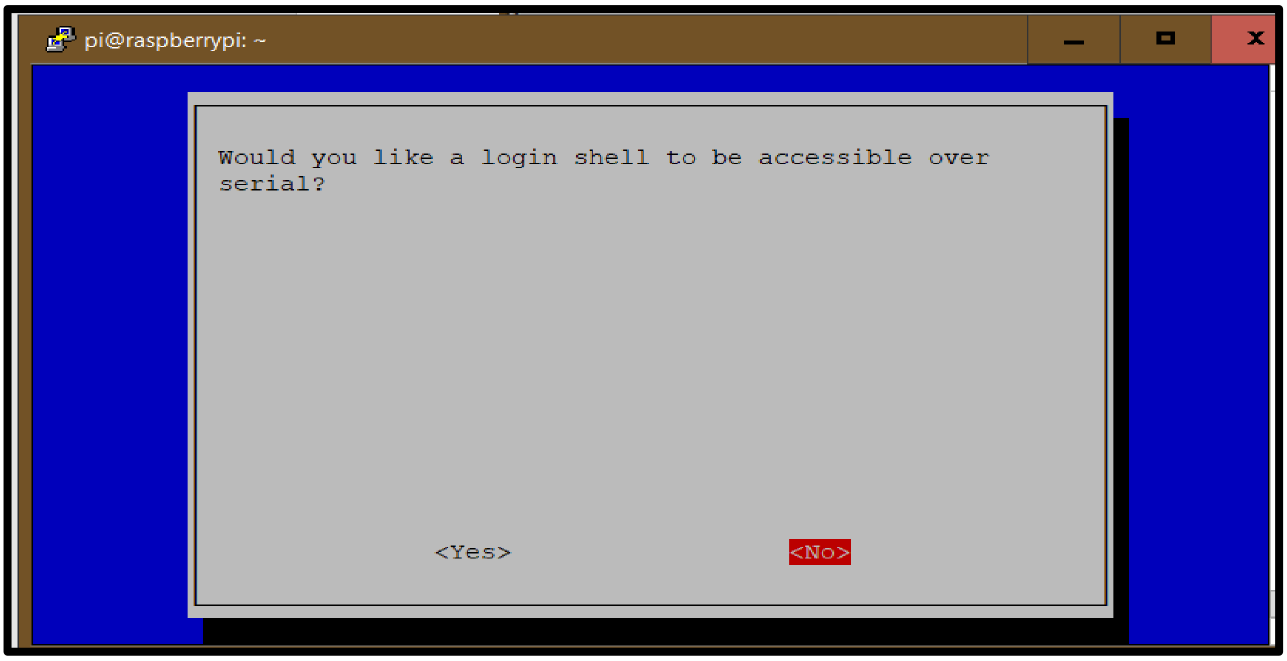
Step 2. **Select Serial Port option**



```
pi@raspberrypi: ~
          Raspberry Pi Software Configuration Tool (raspi-config)

   P1 Camera      Enable/disable connection to the Raspberry Pi Camera
   P2 SSH         Enable/disable remote command line access using SSH
   P3 VNC         Enable/disable graphical remote access using RealVNC
   P4 SPI         Enable/disable automatic loading of SPI kernel module
   P5 I2C         Enable/disable automatic loading of I2C kernel module
   P6 Serial Port Enable/disable shell messages on the serial connection
   P7 1-Wire      Enable/disable one-wire interface
   P8 Remote GPIO Enable/disable remote access to GPIO pins

                  <Select>                     <Back>
```

Figure 5.17: Disabling shell messages on serial port

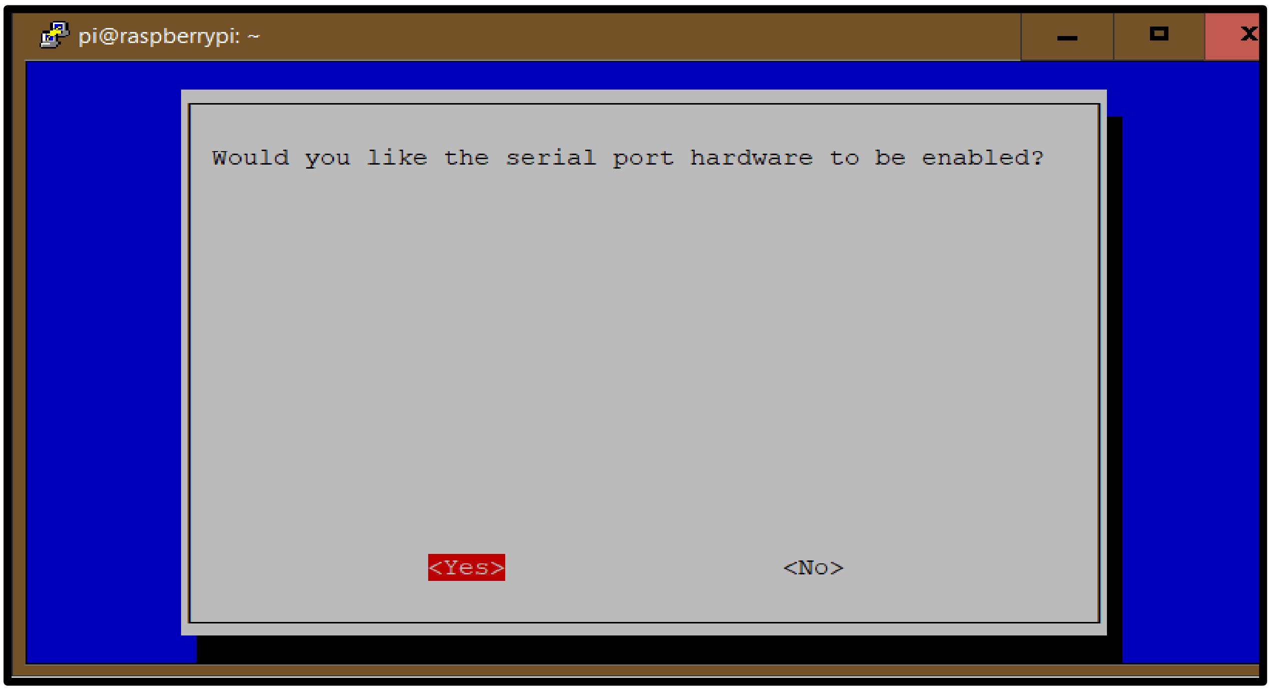
Step 3. **When prompted, select no to "Would you like a login shell to be accessible over serial?"**



```
Would you like a login shell to be accessible over
serial?
```

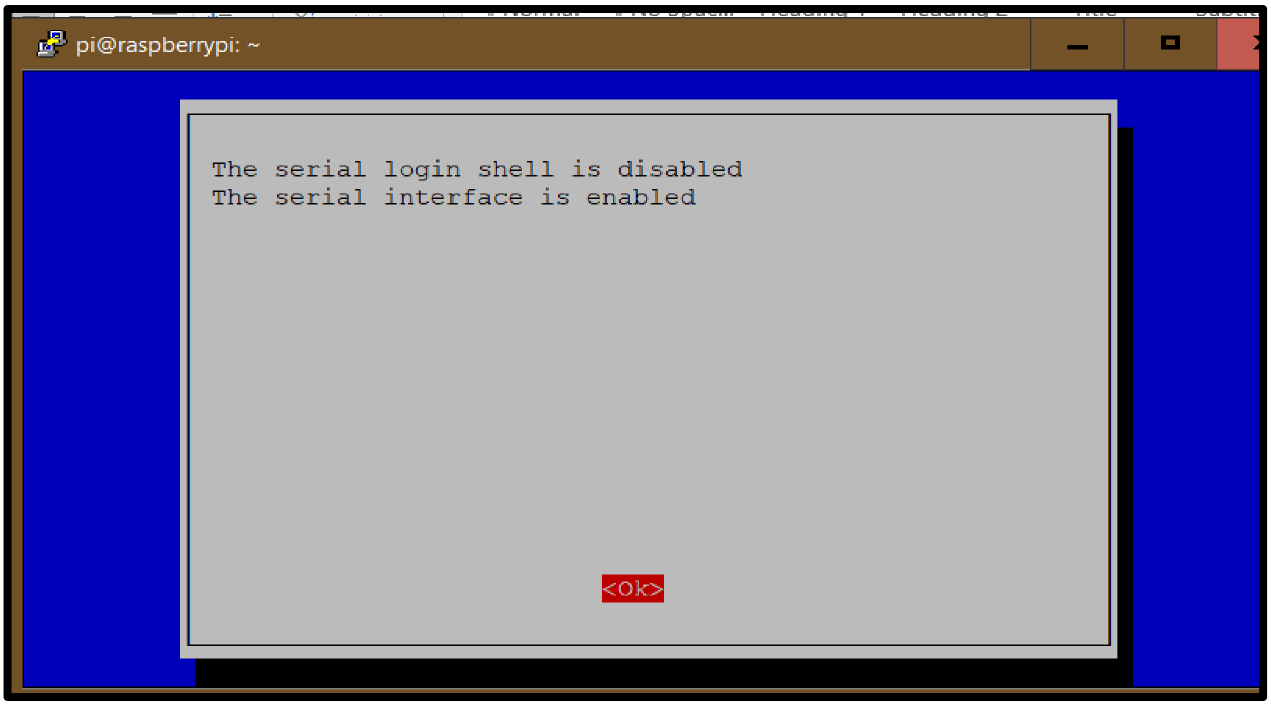
Figure 5.18: Disabling shell messages on serial port

Step 4 **When prompted, select yes to "Would you like serial port hardware to be enabled?**



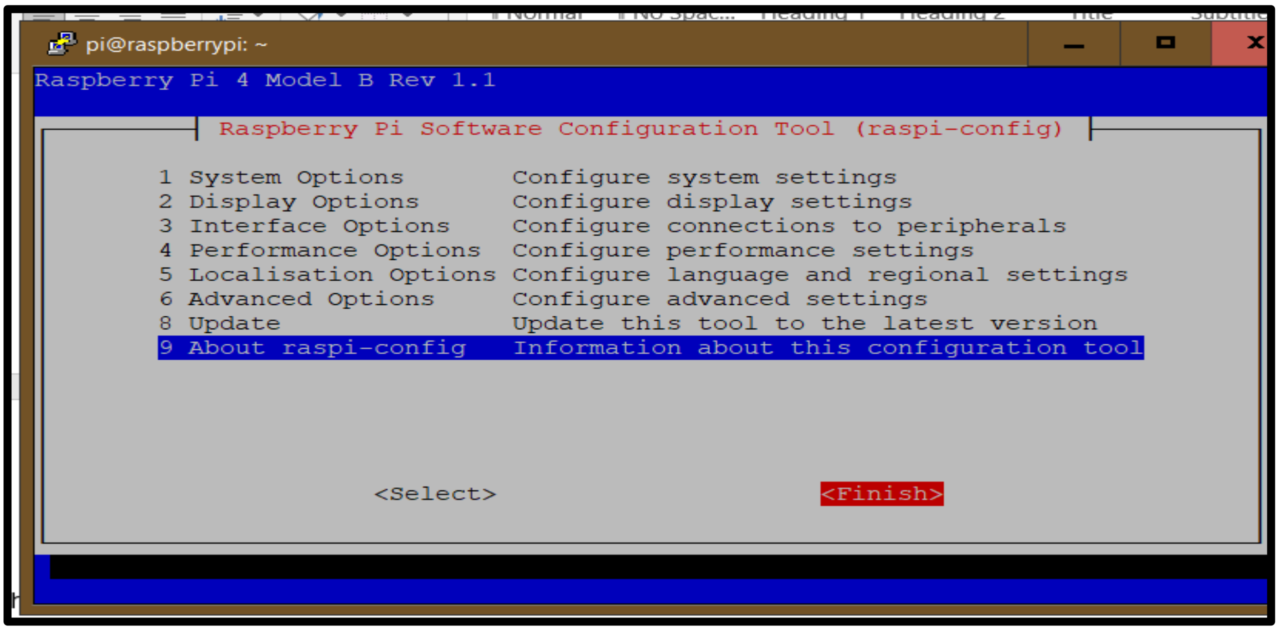
```
Would you like the serial port hardware to be enabled?
```

Figure 5.19: Enabling serial port hardware option



```
pi@raspberrypi: ~

The serial login shell is disabled
The serial interface is enabled

                    <Ok>
```

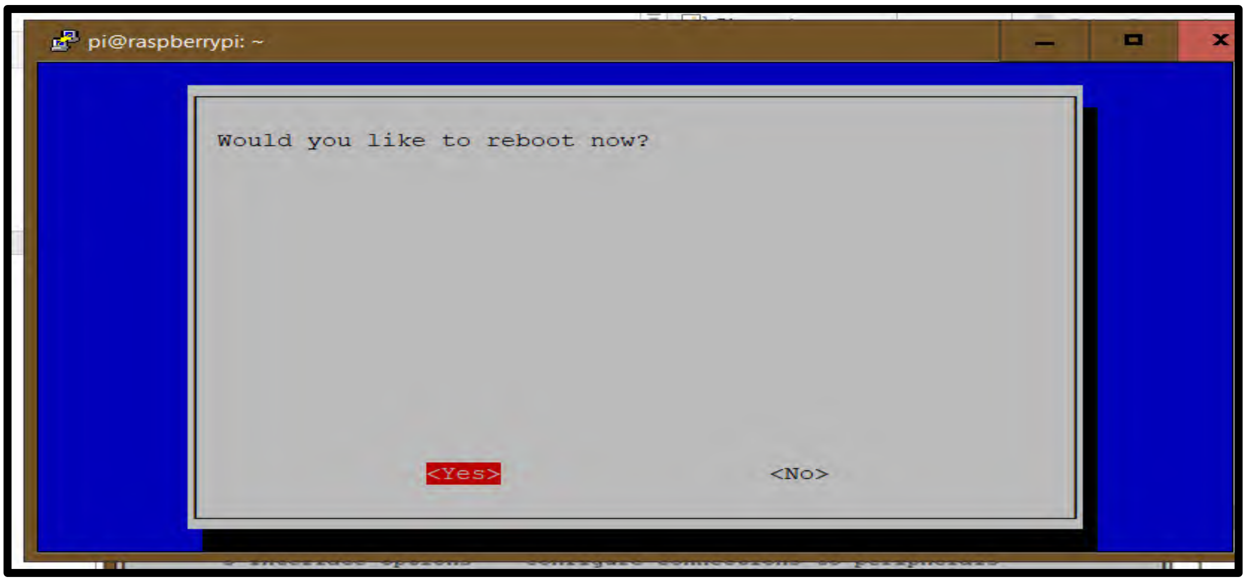
Figure 5.20: Final dialog box displaying login shell disabled and serial interface enabled



```
pi@raspberrypi: ~
Raspberry Pi 4 Model B Rev 1.1

        ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

       1 System Options       Configure system settings
       2 Display Options      Configure display settings
       3 Interface Options    Configure connections to peripherals
       4 Performance Options  Configure performance settings
       5 Localisation Options Configure language and regional settings
       6 Advanced Options     Configure advanced settings
       8 Update               Update this tool to the latest version
       9 About raspi-config   Information about this configuration tool

              <Select>                     <Finish>
```

Figure 5.21: Closing raspi.config option

Step 5 Reboot the system by clicking yes



```
pi@raspberrypi: ~

Would you like to reboot now?

<Yes>                    <No>
```

Figure 5.22: Reboot system after making changes to configuration file

Installing dependencies for MAVPROXY

sudo apt-get install python3-dev python3-opencv python3-wxgtk4.0 python3-pip python3-matplotlib python3-lxml python3-pygame

pip3 install PyYAML mavproxy --user

echo "export PATH=$PATH:$HOME/.local/bin" >> ~/.bashrc

Incase user permission issues (optional)

sudo usermod -a -G dialout <username>

To update an existing installation with the current release on Python 3 based systems

pip3 install mavproxy --user –upgrade

5.4 Interfacing with Beagle Bone Blue Flight Controller Unit (FCU)

Connectivity Diagram Raspberry Pi 4 serial interface will connect with UART1 of Beagle Bone Blue as mentioned below: -

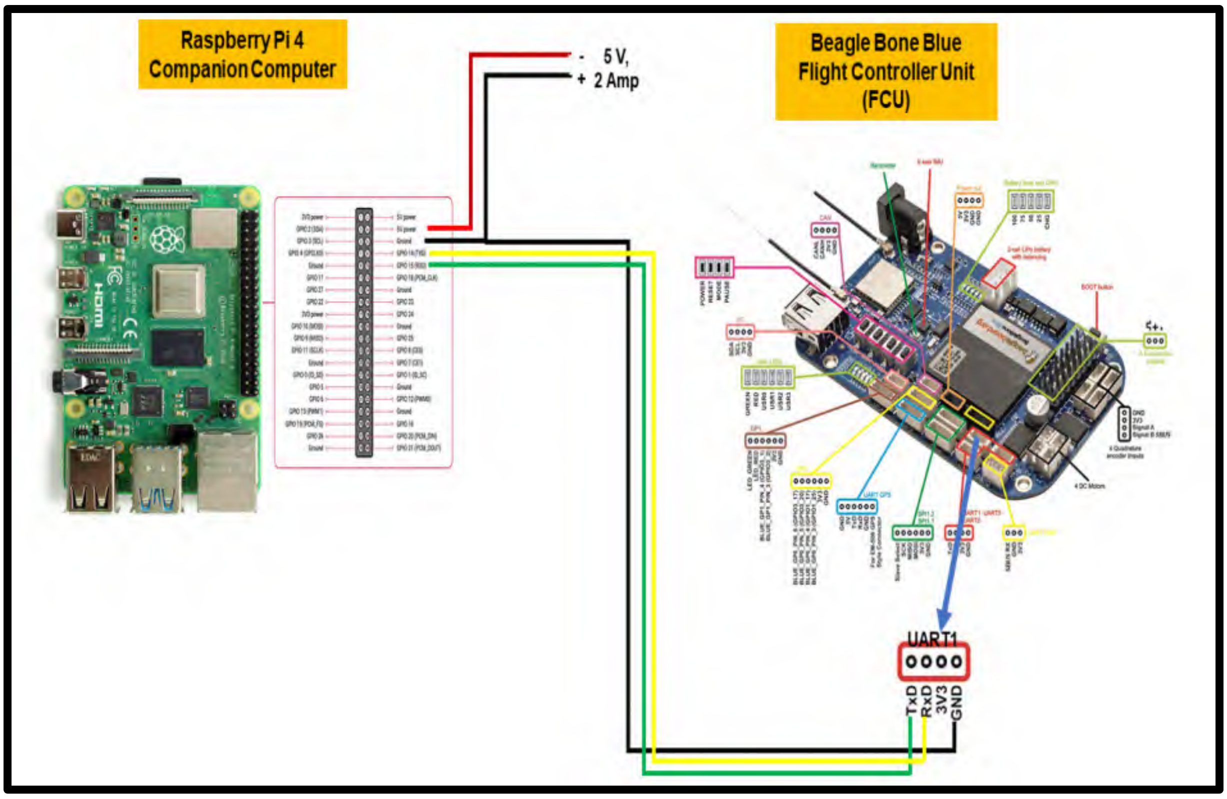


Figure 5.23: Connectivity diagram for physical interfacing between Raspberry Pi 4 as companion computer and Beagle Bone Blue as FCU

Testing MAVlink over MAVPROXY Testing MAVlink though MAVPROXY Command Line GCS. In order to test the configuration, the below mentioned command will have to be run in Terminal window of Raspberry Pi 4. Both Beagle Bone Blue and Raspberry Pi 4 have to be configured as explained till now to be able to test it.

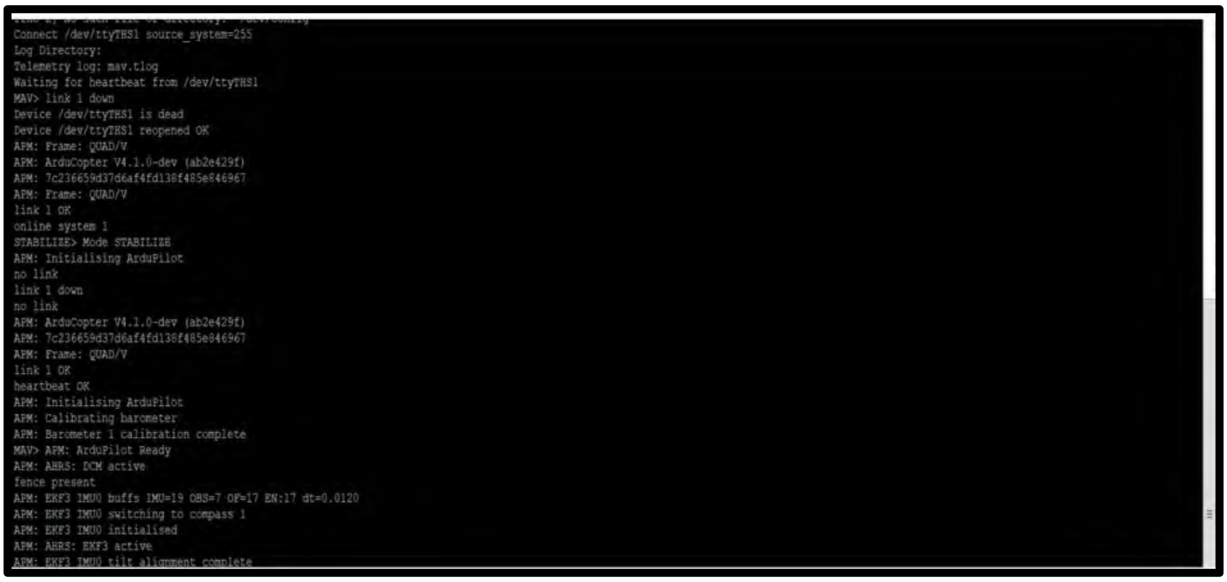
(a) Both Companion computer and GCS on local LAN, then use

python3 $HOME/.local/bin/mavproxy.py --master=/dev/ttyS0 –out=udp:192.168.1.7:14550

(b) VPN Command Example.

python3 $HOME/.local/bin/mavproxy.py --master=/dev/serial0 --out=udp:100.96.1.34:1194

Kindly note that the Raspberry Pi's serial port will now be usable on /dev/serial0 or **dev/ttyS0** and **1194** is port for **VPN** while **local LAN** uses **14550**. The IP Address 192.168.1.7 is the IP Address of GCS on local LAN while 100.96.1.34 is IP on VPN. Replace with your own IP Addresses.



```
Connect /dev/ttyTHS1 source_system=255
Log Directory:
Telemetry log: mav.tlog
Waiting for heartbeat from /dev/ttyTHS1
MAV> link 1 down
Device /dev/ttyTHS1 is dead
Device /dev/ttyTHS1 reopened OK
APM: Frame: QUAD/V
APM: ArduCopter V4.1.0-dev (ab2e429f)
APM: 7c236659d37d6af4fd138f485e846967
APM: Frame: QUAD/V
link 1 OK
online system 1
STABILIZE> Mode STABILIZE
APM: Initialising ArduPilot
no link
link 1 down
no link
APM: ArduCopter V4.1.0-dev (ab2e429f)
APM: 7c236659d37d6af4fd138f485e846967
APM: Frame: QUAD/V
link 1 OK
heartbeat OK
APM: Initialising ArduPilot
APM: Calibrating barometer
APM: Barometer 1 calibration complete
MAV> APM: ArduPilot Ready
APM: AHRS: DCM active
fence present
APM: EKF3 IMU0 buffs IMU=19 OBS=7 OF=17 EN:17 dt=0.0120
APM: EKF3 IMU0 switching to compass 1
APM: EKF3 IMU0 initialised
APM: AHRS: EKF3 active
APM: EKF3 IMU0 tilt alignment complete
```

Figure 5.24: MAXPROXY command output in console window

CHAPTER 6

JETSON NANO AS COMPANION COMPUTER

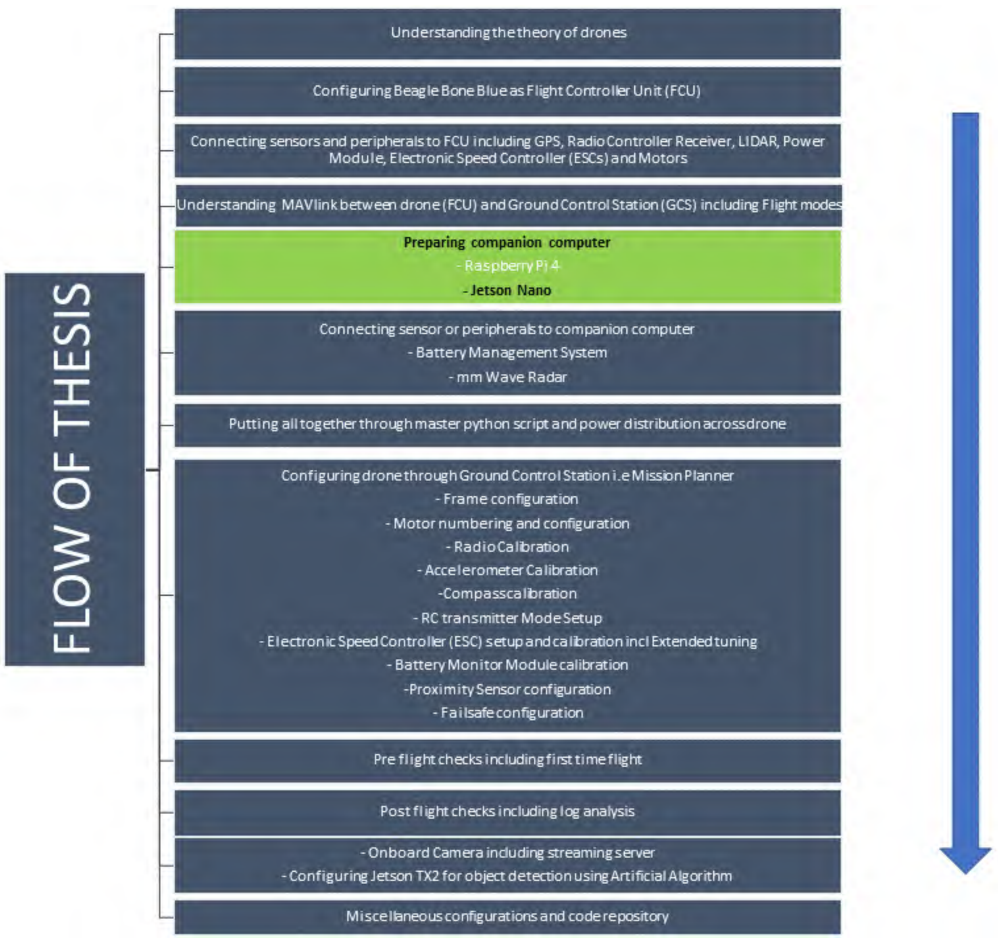

FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server
- Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

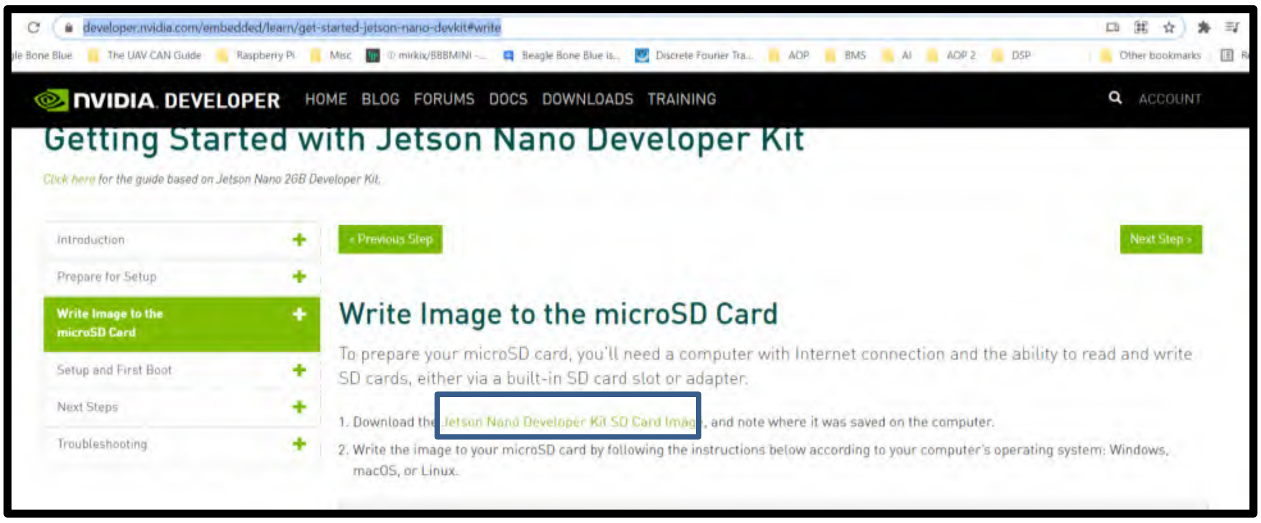
Figure 6.1: Flow of thesis until now

NVIDIA Jetson Nano Developer Kit B-02 is a small, powerful computer which supports running of multiple neural networks in parallel for applications such as image classification, object detection, segmentation, and speech processing with reasonably affordable price. With performance comes excellent power management and the option to run in less than 5 watts. It delivers 472 GFLOPS for running modern AI algorithms fast, with a quad-core 64-bit ARM CPU, a 128-core integrated NVIDIA GPU, as well as 4GB LPDDR4 memory. It has the ability to processes several high-resolution sensors simultaneously. So, for drones running Artificial Intelligence and computer vision specific roles, Jetson Nano is the choice of companion computer. The Jetson Nano will be running the Ubuntu 16.04 LTS with Jetpack 4.5.1.

6.1 Preparing Jetson Nano

Preparing SD Card for Jetson Nano

Step 1 Download the latest **sd-blob-b01.img** latest from **https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write** which is the official site.


developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write

Getting Started with Jetson Nano Developer Kit

Click here for the guide based on Jetson Nano 2GB Developer Kit.

Write Image to the microSD Card

To prepare your microSD card, you'll need a computer with Internet connection and the ability to read and write SD cards, either via a built-in SD card slot or adapter.

1. Download the Jetson Nano Developer Kit SD Card Image, and note where it was saved on the computer.
2. Write the image to your microSD card by following the instructions below according to your computer's operating system: Windows, macOS, or Linux.

Figure 6.2: Downloading OS for Jetson Nano from official website

Use any SD card flashing software like Balena Etcher or Pi imager to flash sd card of 8/16/32 Gb.

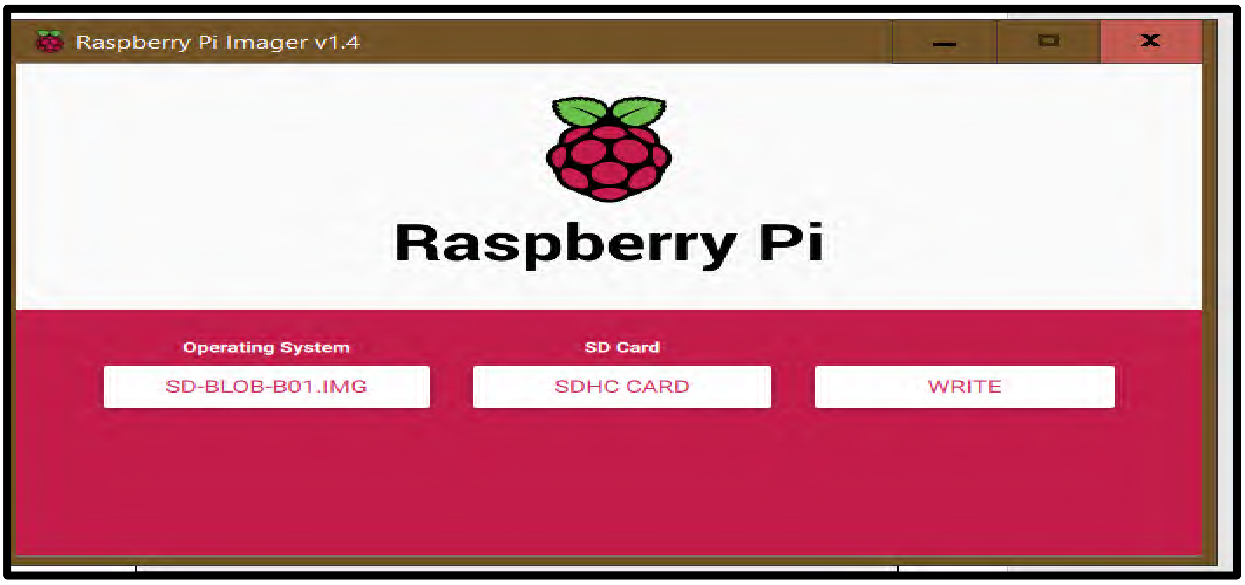
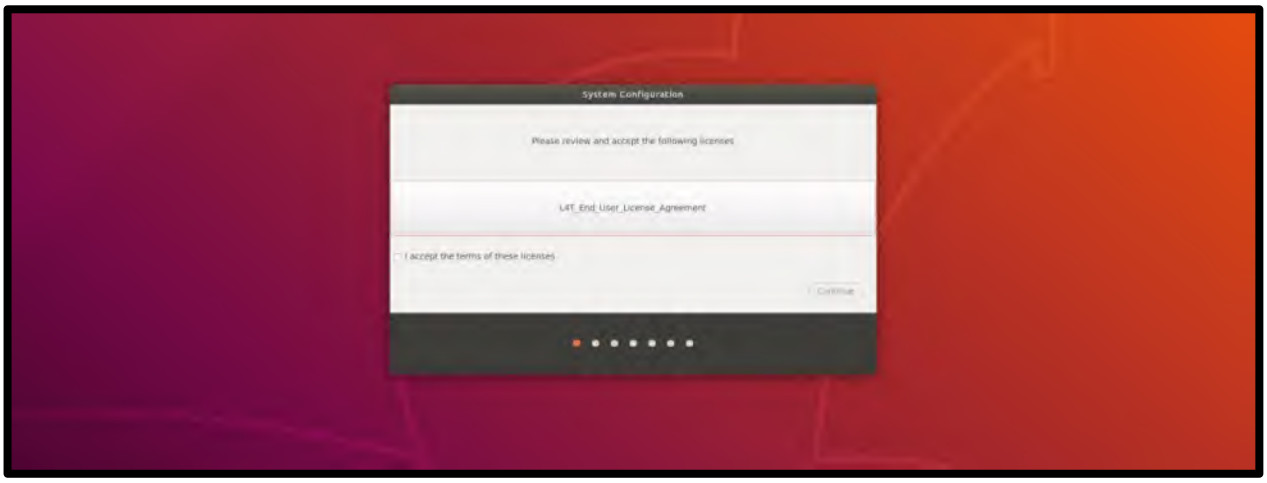


Figure 6.3: Flashing SD card for Jetson Nano using Raspberry Pi Imager

Step 2. Boot Jetson Nano from SD Card and connect with ethernet cable to local LAN with internet. Connect Monitor via HDMI port to do the startup process. Configure your Ubuntu installation progress (language, keyboard type, location, username & password etc.). Connect Jetson Nano to the internet to create username and password just like any Ubuntu startup and update/upgrade OS.


System Configuration

Please review and accept the following licenses

L4T_End_User_License_Agreement

☐ I accept the terms of these licenses

Figure 6.4: Configuring Ubuntu Operating System profile

sudo apt-get update
sudo apt-get upgrade

Use **ifconfig** command to check IP Address allotted to Jetson Nano over local LAN. This is now used for taking ssh also. The point to note is that Jetson Nano does not have an inbuilt wifi and requires ethernet based connection for internet access. So, any access over local network using ssh will require physical ethernet connection to Jetson Nano. Later I will explain how we can use USB to Wi-Fi adaptor to connect Jetson Nano over internet at startup in headless mode just like Raspberry Pi4. By default, Wi-Fi isn't enabled on Jetson Nano unless user logins to his/her profile. This problem has been solved later in the document as it's an important requirement for Jetson Nano as companion computer onboard drone. Username and password are defined by user during startup process. The same is required for taking ssh later on

6.2 Installation of VPN as a service

To install VPN, the process is similar to the once explained in Chapter4.

Step1. First install Open VPN application on Jetson Nano.

sudo apt-get install openvpn

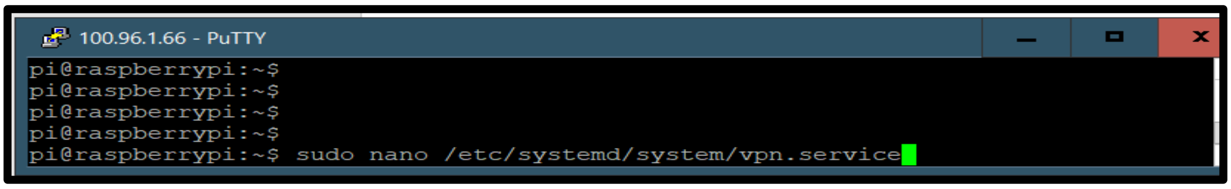
Use WinSCP application on windows to transport the folder VPN_Folder that contains all my .ovpn files. Let me remind you that the VPN_Folder contains .ovpn files for FCU, Companion computer and Ground Control Station (GCS). I only need .ovpn file that I created for Jetson Nano from the folder. However, I am transporting the entire folder just for the ease of it. I have done the same process for Raspberry Pi 4 in Chapter 5 above Please note the relevant file in VPN_Folder for me is **jetson.ovpn** which I renamed after download from OpenVPN website as shown in Chapter 4. The folder was first copied to location /home/yash being default directory as shown below:-

Now move the folder from /home/yash to /etc/openvpn for execution

sudo mv /home/yash/VPN_Folder /etc/openvpn/

Step 2. Create systemd file for startup at boot

sudo nano /etc/systemd/system/vpn.service



```
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo nano /etc/systemd/system/vpn.service
```

Figure 6.5: Creating vpn.service at location /etc/systemd/system in Ubuntu Operating System (OS)

```
[Unit]
Description=VPN Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
```

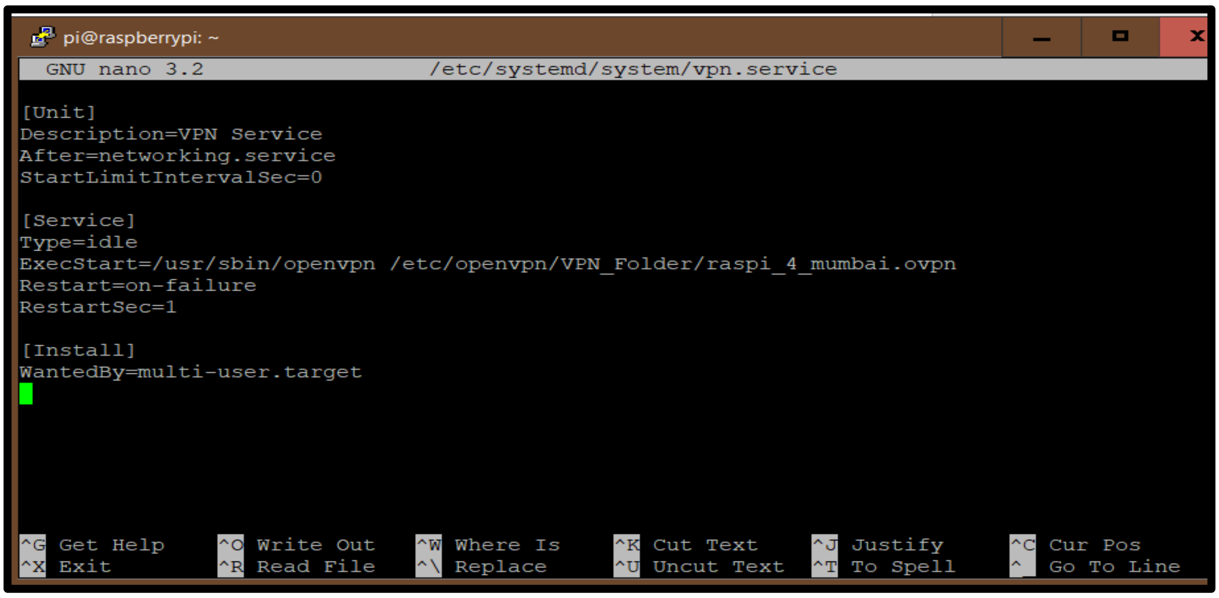
ExecStart=/usr/sbin/openvpn etc/openvpn/VPN_Folder/jetson.ovpn

Restart=on-failure

RestartSec=1

[Install]

WantedBy=multi-user.target



```
  GNU nano 3.2                    /etc/systemd/system/vpn.service

[Unit]
Description=VPN Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
ExecStart=/usr/sbin/openvpn /etc/openvpn/VPN_Folder/raspi_4_mumbai.ovpn
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target



^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Figure 6.6: Creating content of vpn.service for enabling VPN services on startup

sudo systemctl enable vpn.service

sudo systemctl start vpn.service

sudo systemctl status vpn.service

sudo systemctl daemon-reload (only to reload service incase some changes are made)

6.3 Installation of MAVPROXY (Command Line Ground Control Station) as a service

MAVPROXY is the command line Ground Control Station (GCS) to be installed in companion computer. This MAVPROXY will receive MAVlink data (link between Ardupilot running on FCU i.e Beagle Bone Blue and companion computer) over serial interface and relay it back to the ground control station i.e Mission Planner running over laptop/PC. The MAVlink is installed over secure VPN using UDP protocol over internet connection extended by 4G LTE

Modem. Kindly note that the serial port in Jetson Nano is identified as /**dev/ttyTSH0**

Installing dependencies for MAVPROXY

```
sudo apt-get install python3-dev python3-opencv python3-wxgtk4.0 python3-pip python3-matplotlib python3-lxml python3-pygame
pip3 install PyYAML mavproxy --user
echo "export PATH=$PATH:$HOME/.local/bin" >> ~/.bashrc
```

Incase user permission issues (optional)

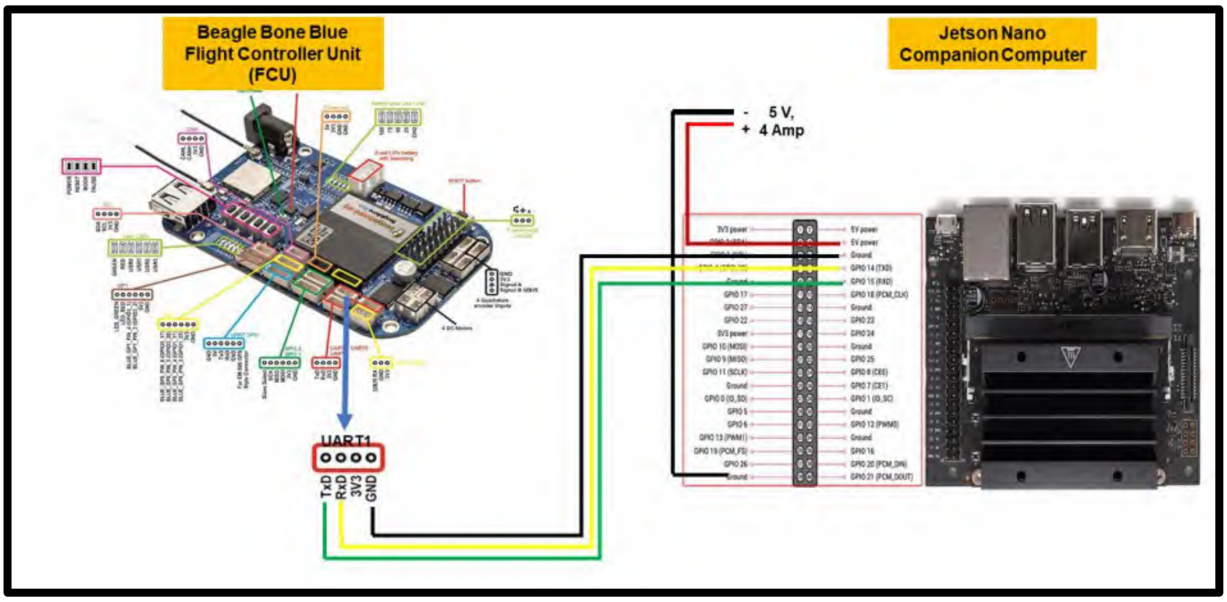
```
sudo usermod -a -G dialout <username>
```

To update an existing installation with the current release on Python 3 based systems

```
pip3 install mavproxy --user –upgrade
```

6.4 Interfacing with Beagle Bone Blue Flight Controller Unit (FCU)

Connectivity Diagram Jetson Nano serial interface will connect with UART1 of Beagle Bone Blue as mentioned below: -



Beagle Bone Blue Flight Controller Unit (FCU)
Jetson Nano Companion Computer
- 5 V,
+ 4 Amp
UART1
TxD
RxD
3V3
GND

Figure 6.7: Connectivity diagram for physical interfacing of Jetson Nano as companion computer with Beagle Bone Blue as FCU

Testing MAVlink over MAVPROXY Testing MAVlink though

MAVPROXY Command Line GCS. In order to test the configuration, the below mentioned command will have to be run in Terminal window of Jetson Nano 4. Both Beagle Bone Blue and Jetson Nano have to be configured as explained till now to be able to test it.

(a) Both Companion computer and GCS on local LAN, then use

sudo python3 $HOME/.local/bin/mavproxy.py --master=/dev/ttyTSH0 –out=udp:192.168.1.7:14550

(b) VPN Command Example.

sudo python3 $HOME/.local/bin/mavproxy.py --master=/dev/ttyTSH0 –out=udp:100.96.1.34:1194

Kindly note that the Raspberry Pi's serial port will now be usable on /dev/ttyTSH0 and **1194** is port for **VPN** while **local LAN** use **14550**. IP Address 192.168.1.7 is local LAN IP Address while 100.96.1.34 is the VPN IP Address of GCS. Replace with your own IP Address to test.

6.5 Running WIFI on headless startup (peculiar to Jetson Nano)

Unlike Raspberry Pi 4, the Jetson Nano does not have an inbuilt wifi and requires external adaptor (USB Wifi). In our case, the 4G LTE Modem is on Beagle Bone Blue, hence Jetson Nano is required to connect to internet on wifi on startup. We want Jetson Nano to connect with wifi onboard drone as an when the drone starts and allow us to take remote ssh over VPN and also establish MAVlink over VPN. Thus, we require to disable power saving mode and allow wifi connection on startup through below steps:-

Step 1. Update the Ubuntu OS on Jetson Nano first

sudo apt-get update

Step 2. Install dependencies

```
sudo apt-get install git linux-headers-generic build-essential dkms
```

<u>Step 3</u>. Git clone repository

```
git clone https://github.com/pvaret/rtl8192cu-fixes.git
```

The list of wifi adaptors supported for Jetson Nano

ASUSTek USB-N13 rev. B1 (0b05:17ab)
Belkin N300 (050d:2103)
D-Link DWA-121 802.11n Wireless N 150 Pico Adapter [RTL8188CUS]
Edimax EW-7811Un (7392:7811)
Kootek -RPWF (0bda:8176)
OurLink 150M 802.11n (0bda:8176)
Plugable USB 2.0 Wireless N 802.11n (0bda:8176)
TP-Link TL-WN725N (0bda:8176)
TP-Link TL-WN821Nv4 (0bda:8178)
TP-Link TL-WN822N (0bda:8178)
TP-Link TL-WN823N (only models that use the rtl8192cu chip)
TRENDnet TEW-648UBM N150

```
sudo dkms add ./rtl8192cu-fixes

sudo dkms install 8192cu/1.11

sudo depmod -a
sudo cp ./rtl8192cu-fixes/blacklist-native-rtl8192.conf /etc/modprobe.d/
sudo echo options rtl8xxxu ht40_2g=1 dma_aggregation=1 | sudo tee
/etc/modprobe.d/rtl8xxxu.conf
```

Now we need to enable auto login. This ensures that wifi will not lose connectivity on ssh.

```
sudo nano /etc/gdm3/custom.conf
```

In this file, uncomment the following:

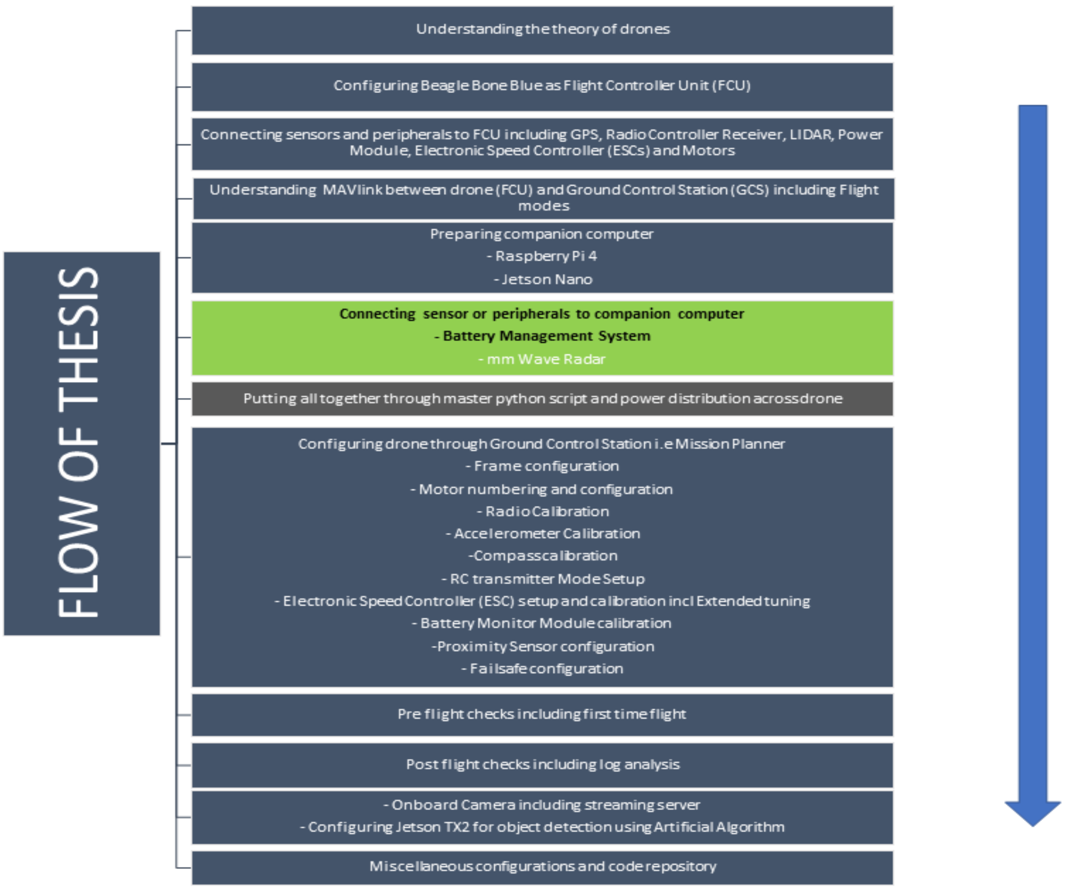
AutomaticLoginEnable = true

AutomaticLogin = user // put your user name here e.g. jetson

sudo reboot now

CHAPTER 7

INTEGRATING BATTERY MANAGEMENT (BMS) OF SMART BATTERY WITH COMPANION COMPUTER LIKE RASPBERRY PI 4 OR JETSON NANO TO PROVIDE LIVE DASHBOARD



FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer - Raspberry Pi 4 - Jetson Nano
Connecting sensor or peripherals to companion computer - Battery Management System - mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner - Frame configuration - Motor numbering and configuration - Radio Calibration - Accelerometer Calibration -Compass calibration - RC transmitter Mode Setup - Electronic Speed Controller (ESC) setup and calibration incl Extended tuning - Battery Monitor Module calibration -Proximity Sensor configuration - Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server - Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

Figure 7.1: Flow of thesis until now

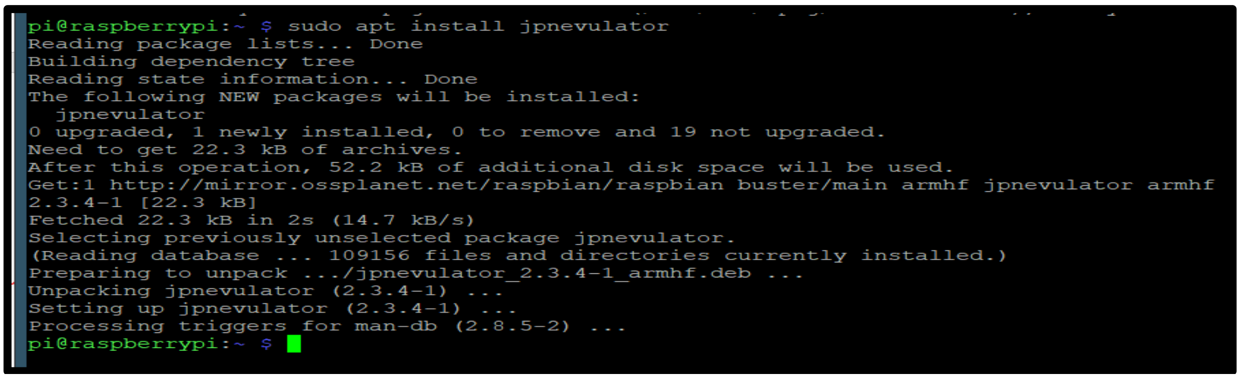
Basic working philosophy The Battery Management System (BMS) of the smart battery provides data in human readable form through RS485 interface. This from RS485 can be directly converted to USB or RS485 to TTL to USB for integrating on companion computer. This data is then captured through minicom application running on companion computer (Raspberry Pi 4 or Jetson Nano) and dumped into .csv file with date and time stamp. The .csv file data contains all the data instances (roughly 108 data packets per second). However, the python script captures one instance of data every 1 minutes to plot it on the dashboard.

7.1 Integrating smart battery with Battery Management System (BMS) with drone

The steps to install and configure the package is

Step 1 Install minicom application on companion computer

sudo apt-get install minicom

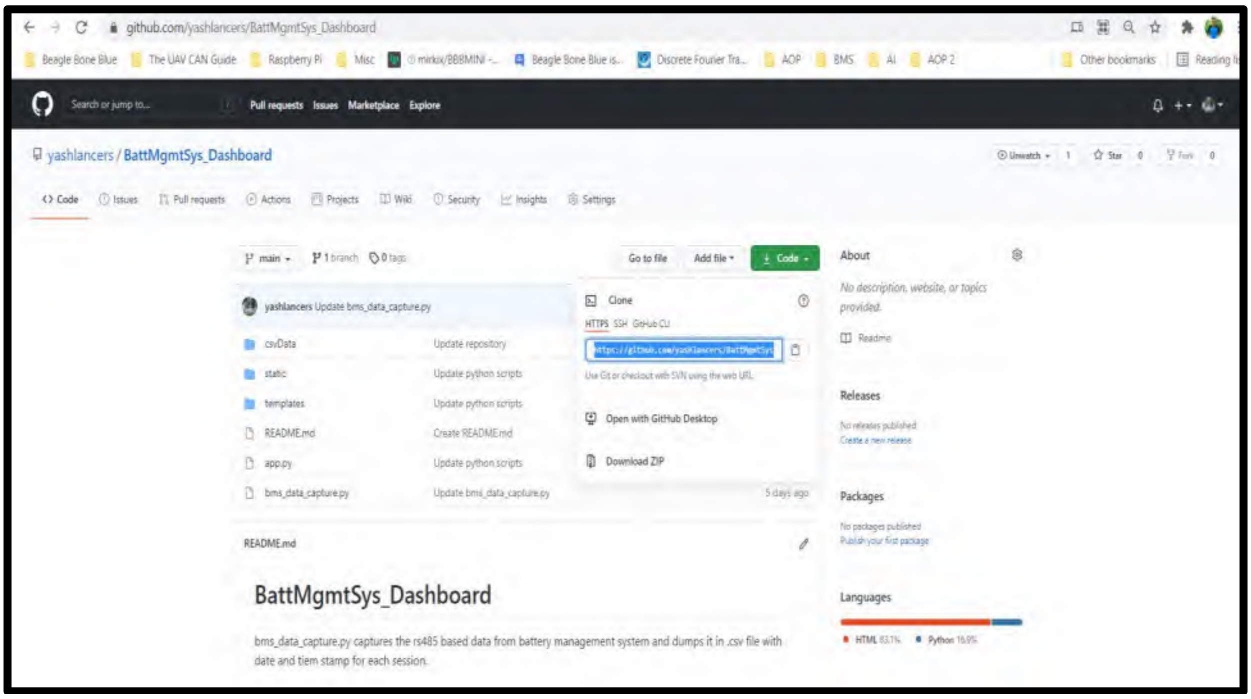


```
pi@raspberrypi:~ $ sudo apt install jpnevulator
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  jpnevulator
0 upgraded, 1 newly installed, 0 to remove and 19 not upgraded.
Need to get 22.3 kB of archives.
After this operation, 52.2 kB of additional disk space will be used.
Get:1 http://mirror.ossplanet.net/raspbian/raspbian buster/main armhf jpnevulator armhf
2.3.4-1 [22.3 kB]
Fetched 22.3 kB in 2s (14.7 kB/s)
Selecting previously unselected package jpnevulator.
(Reading database ... 109156 files and directories currently installed.)
Preparing to unpack .../jpnevulator_2.3.4-1_armhf.deb ...
Unpacking jpnevulator (2.3.4-1) ...
Setting up jpnevulator (2.3.4-1) ...
Processing triggers for man-db (2.8.5-2) ...
pi@raspberrypi:~ $
```

Figure 7.2: Installing minicom application for reading serial data from Battery Management System (BMS)

Step 2 Get the repository from my github account in the name of yashlancers

git clone https://github.com/yashlancers/BattMgmtSys_Dashboard.git



```
github.com/yashlancers/BattMgmtSys_Dashboard

yashlancers / BattMgmtSys_Dashboard

main   1 branch   0 tags                        Go to file   Add file   Code

yashlancers Update bms_data_capture.py
csvData              Update repository
static               Update python scripts
templates            Update python scripts
README.md            Create README.md
app.py               Update python scripts
bms_data_capture.py  Update bms_data_capture.py        5 days ago

Clone
HTTPS  SSH  GitHub CLI
https://github.com/yashlancers/BattMgmtSys
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP

README.md

BattMgmtSys_Dashboard

bms_data_capture.py captures the rs485 based data from battery management system and dumps it in .csv file with
date and tiem stamp for each session.

About
No description, website, or topics provided.
Releases
No releases published
Packages
No packages published
Languages
HTML 83.1%   Python 16.9%
```

Figure 7.3: Downloading Battery Dashboard repository from my github account yashlancers

Step 3 After successful download of the repository, move into the file directory of the folder downloaded.

cd BattMgmtSys_Dashboard

The folder contains following files

- csvData is for dumping serial data from BMS RS 485 to USB in .csv file
- bms_data_capture.py script is for capturing BMS data
- app.py script is for plotting the data from .csv file to dashboard
- template folder contains index.html file for dashboard
- static folder contains images and other resource of dashboard

Step 4 The intelligent Battery has a Battery Management System (BMS) designed inhouse by Centre for Battery Engineering and Electric Vehicles (CBEEV) at Indian Institute of Technology, Madras Research Park (IITMRP). The dashboard has been created by us as part of the project.

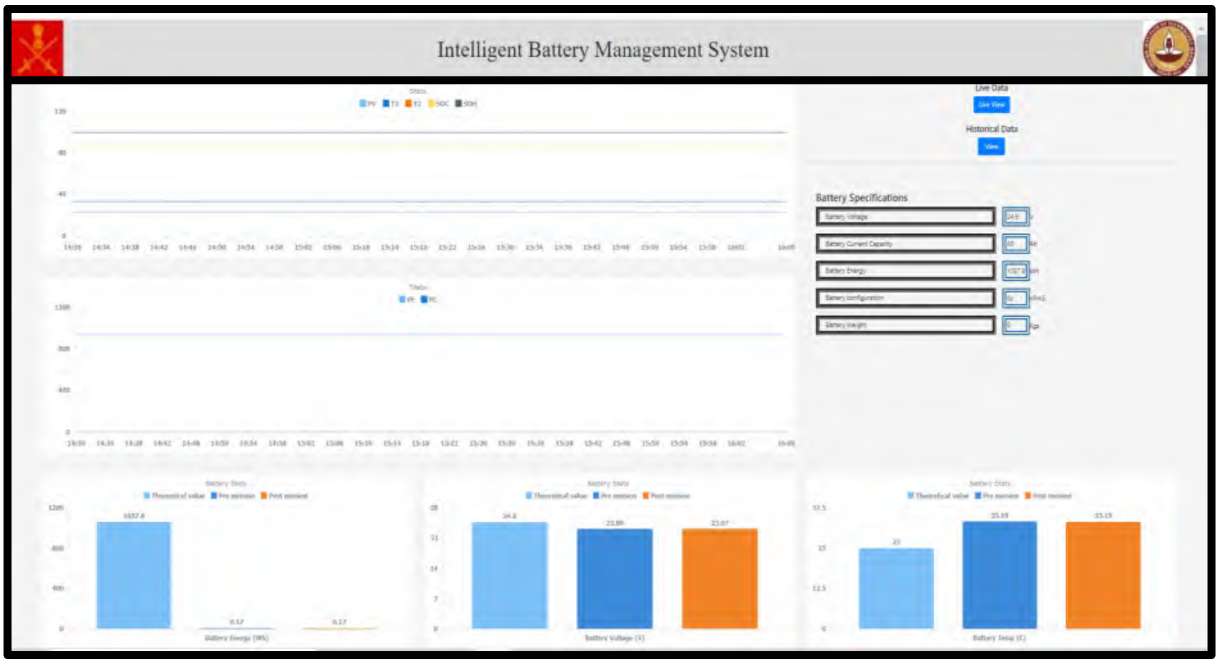
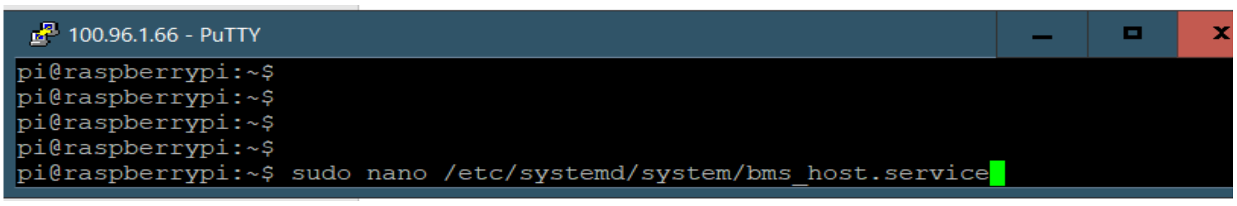


Figure 7.4: Screenshot of the intelligent dashboard for monitoring vital statistics of the battery

The BMS provides data about the Battery voltage, battery current, Battery pack energy consumed, Battery Temperature, State of Health (SoH) and State of Charge (SoC). The data from BMS is captured through bms_data_capture.py python script which is run on startup though a startup master_script.py explained in later chapters of the thesis. This captured data is dumped in csvData folder with date and time stamp. The latest data file is picked up by app.py python script and plotted on web-based dashboard using index.html and flash server. All these files are auto downloaded when we download the repository from github.

Step 5 We need to create a startup script for plotting of battery BMS data on startup of system.

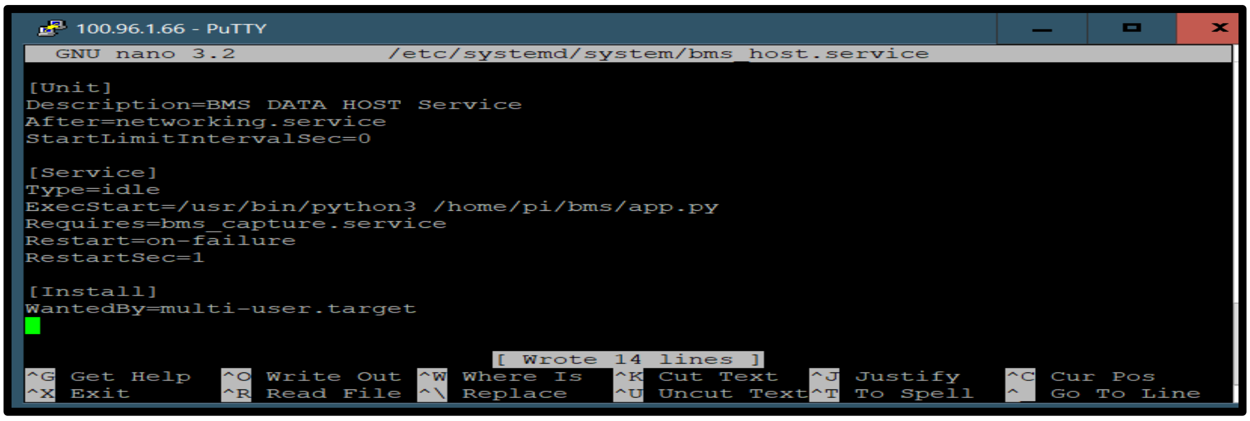
sudo nano /etc/systemd/system/bms_host.service



```
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo nano /etc/systemd/system/bms_host.service
```

Figure 7.5: Creating bms_host.service at /etc/systemd/system directory location Linux Operating System (OS) of companion computer

```
[Unit]
Description=BMS DATA HOST Service
After=networking.service
StartLimitIntervalSec=0

[Service]
ExecStartPre=/bin/sleep 59
ExecStart=/usr/bin/python3 /home/pi/ BattMgmtSys_Dashboard/app.py
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target
```



```
  GNU nano 3.2              /etc/systemd/system/bms_host.service

[Unit]
Description=BMS DATA HOST Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
ExecStart=/usr/bin/python3 /home/pi/bms/app.py
Requires=bms_capture.service
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target


                             [ Wrote 14 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Figure 7.6: Creating content of bms_host.service so that it can run the web-based dashboard on startup

Press Ctrl+O, enter and then Ctrl+X to save and exit. To enable bms_host.service, then start and check status. After status is successful, we are confirmed the service is good to go.

sudo systemctl enable bms_host.service

sudo systemctl start bms_host.service

sudo systemctl status bms_host.service

Important point to note Its important to note that the process for getting battery data is same for Raspberry Pi or Jetson Nano as companion computer. Only file paths and port interfaces will change as per the computer being used in python scripts.

7.2 Testing smart battery in field

This smart battery was tested on Agricopter and the data was recorded on the online dashboard over VPN to battery performance against peak motor throttle and sustained motor throttle. The battery was tested with motor throttle above 50% all times and taking it to peak 100% to test battery discharge capacity against peak requirements. Battery was tested for 18 minutes and reached peak discharge current of 122 Amps. The rate of energy consumption was 18Wh per min to test battery under stress conditions.

Battery type NMC

Battery pack voltage 24V (6s)

Battery Current Capacity 43 Ah

Battery Energy 1032 Wh

Peak current discharge at 100% Motor throttle 122 Amps

Average current discharge 50 Amps with throttle >50%

Total test time 18 minutes

Total energy consumption 188 Wh

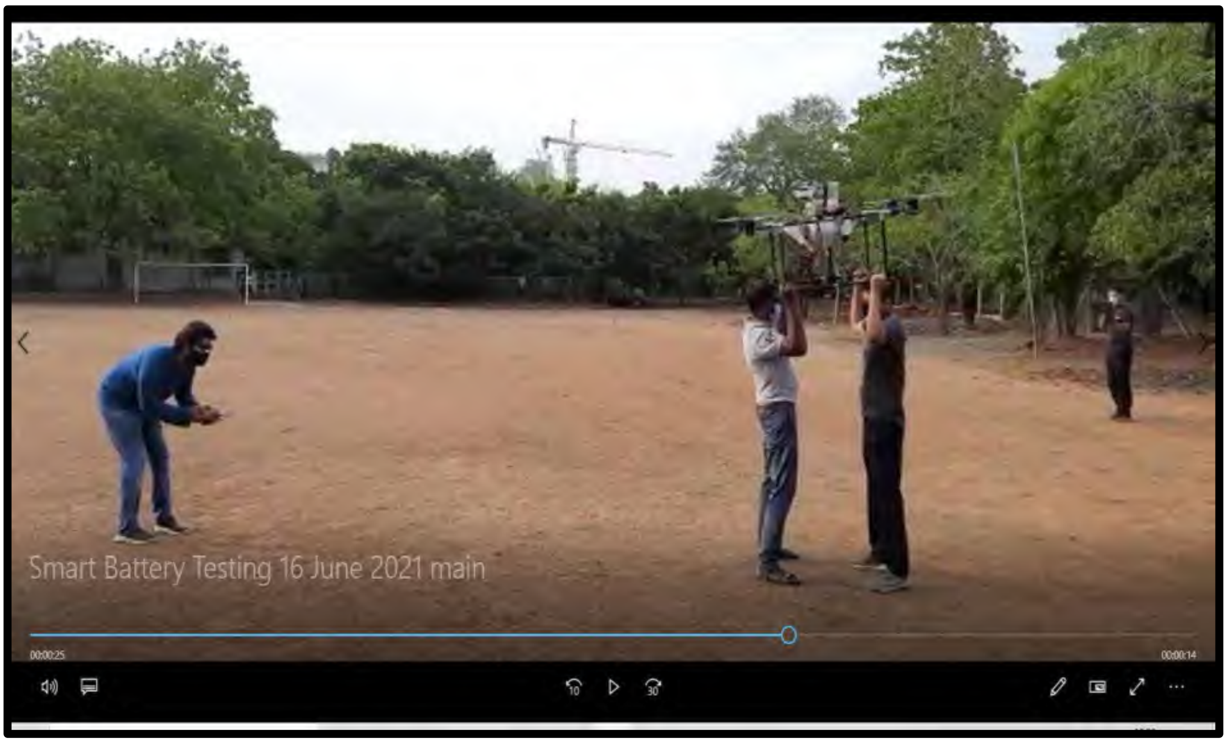
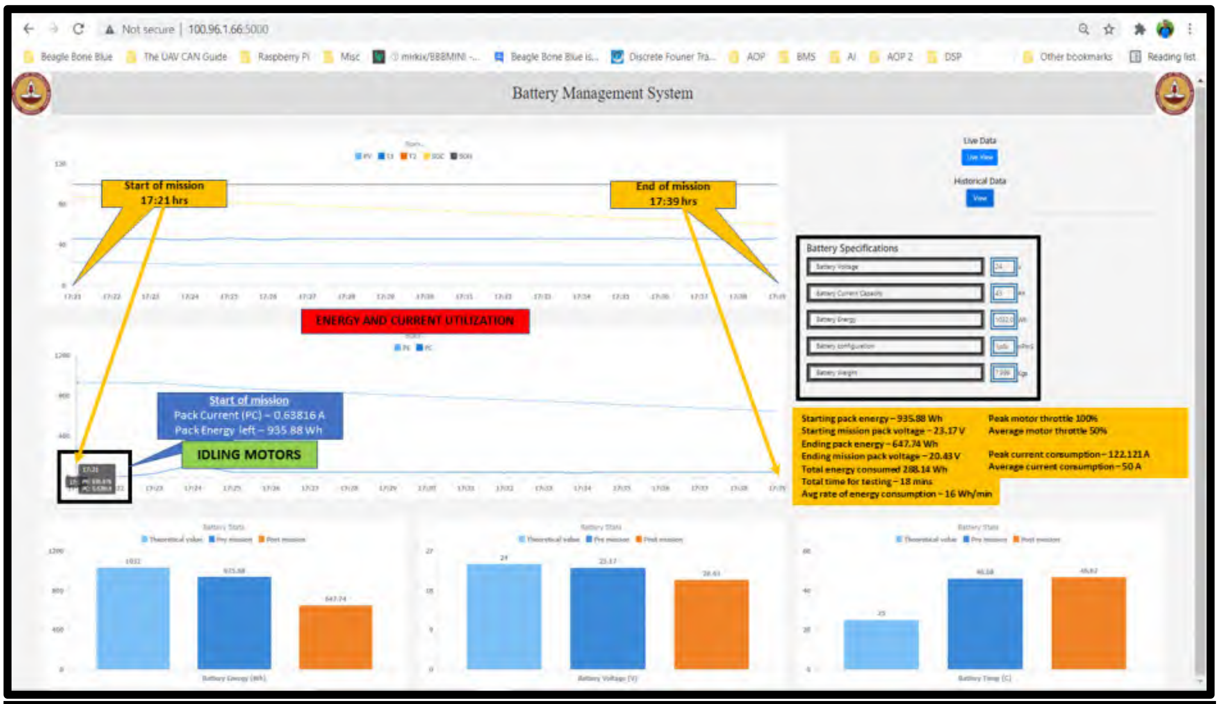


Figure 7.7: Testing smart battery with Battery Management System (BMS) on Agricopter



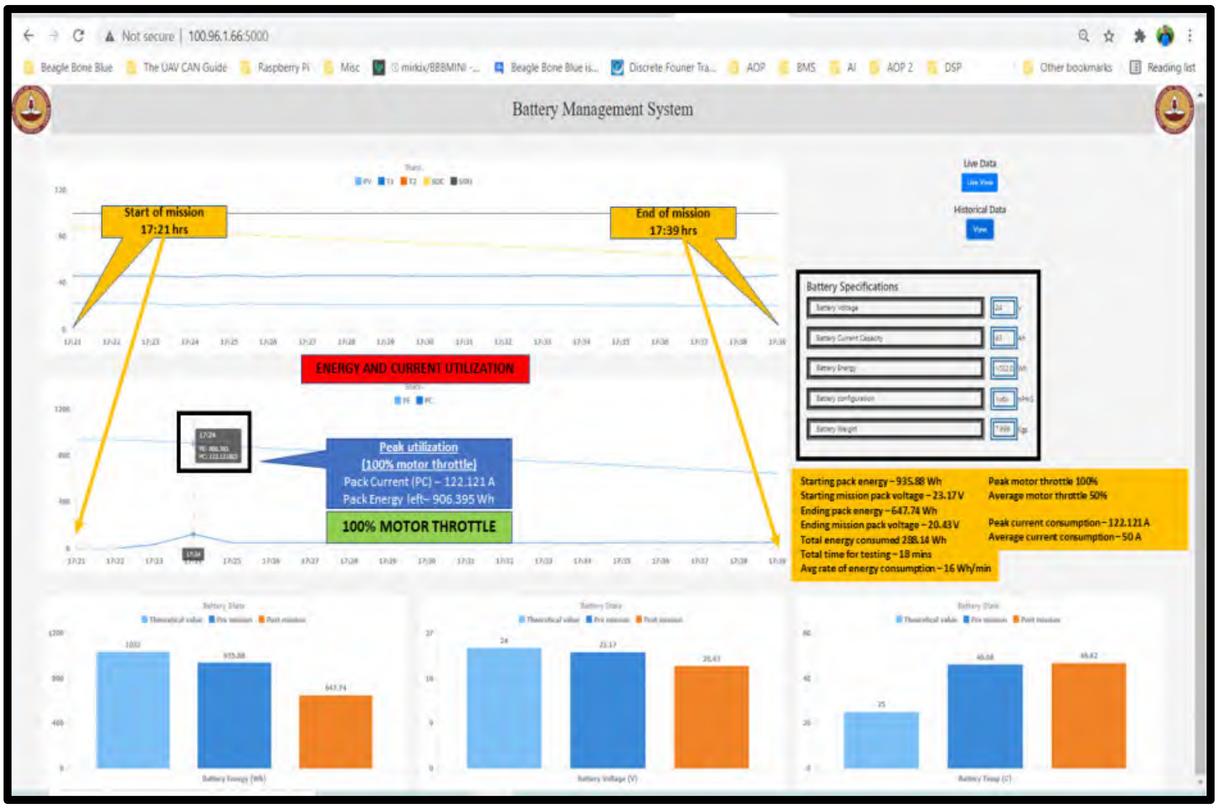
Start of mission
Pack Current (PC) – 0.63816 A
Pack Energy left – 935.88 Wh

Starting pack energy – 935.88 Wh
Starting mission pack voltage – 23.17 V
Ending pack energy – 647.74 Wh
Ending mission pack voltage – 20.43 V
Total energy consumed 288.14 Wh
Total time for testing – 18 mins
Avg rate of energy consumption – 16 Wh/min

Peak motor throttle 100%
Average motor throttle 50%

Peak current consumption – 122.121 A
Average current consumption – 50 A

Figure 7.8: Energy and current utilization on intelligent dashboard of smart battery with Battery Management System (BMS) during testing on agricopter

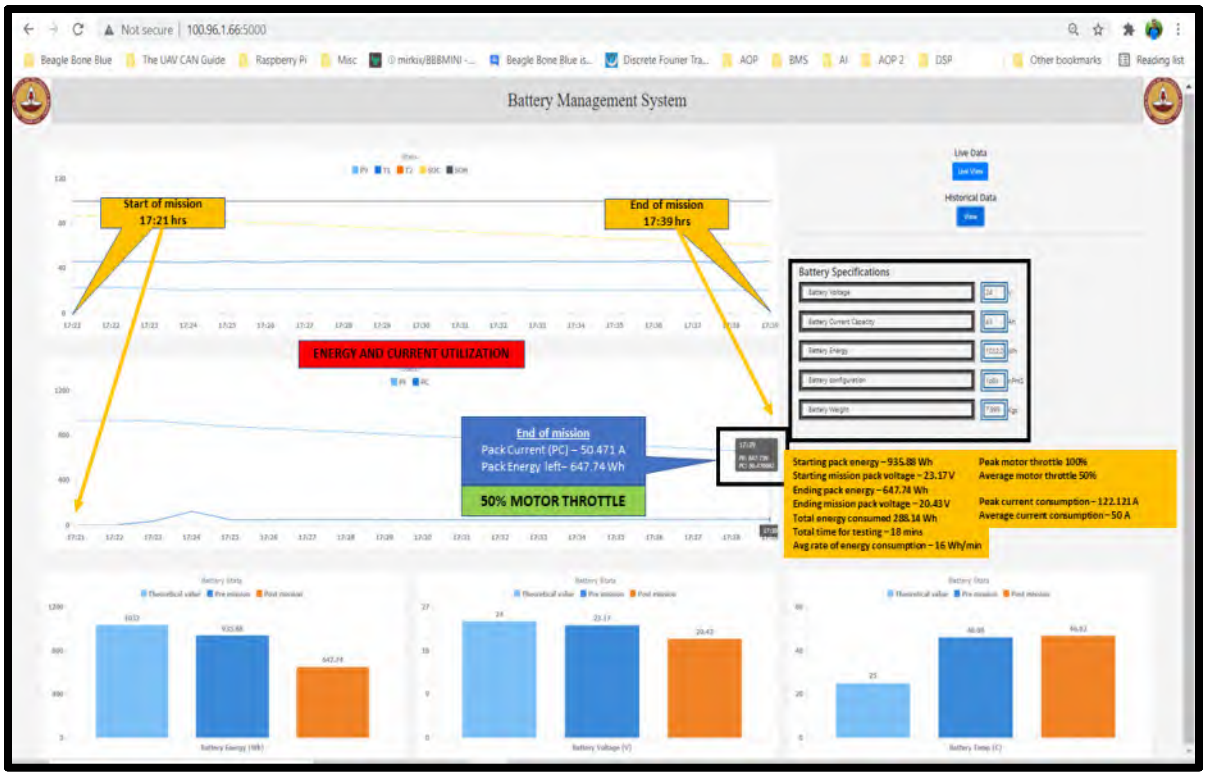


Starting pack energy – 935.88 Wh
Starting mission pack voltage – 23.17 V
Ending pack energy – 647.74 Wh
Ending mission pack voltage – 20.43 V
Total energy consumed 288.14 Wh
Total time for testing – 18 mins
Avg rate of energy consumption – 16 Wh/min

Peak motor throttle 100%
Average motor throttle 50%

Peak current consumption – 122.121 A
Average current consumption – 50 A

Figure 7.9: Energy and current utilization on intelligent dashboard of smart battery with Battery Management System (BMS) during testing on agricopter

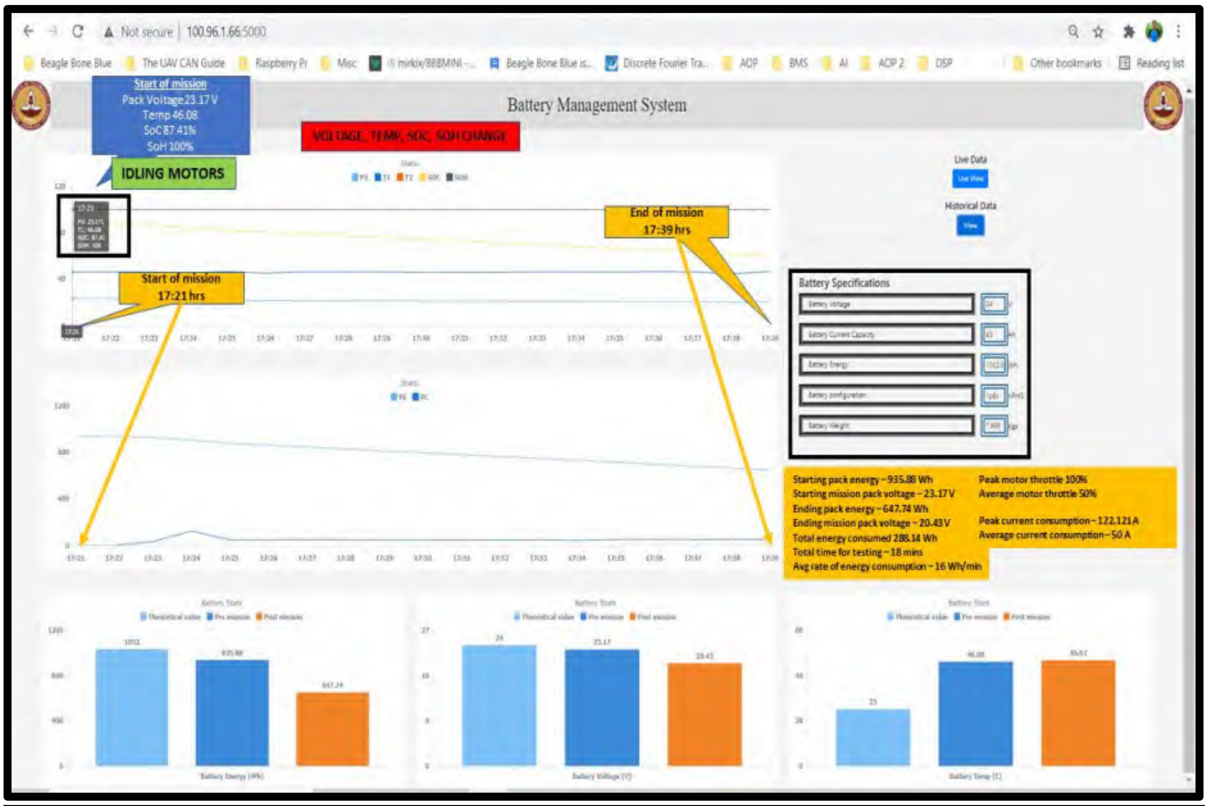


Starting pack energy – 935.88 Wh
Starting mission pack voltage – 23.17 V
Ending pack energy – 647.74 Wh
Ending mission pack voltage – 20.43 V
Total energy consumed 288.14 Wh
Total time for testing – 18 mins
Avg rate of energy consumption – 16 Wh/min

Peak motor throttle 100%
Average motor throttle 50%

Peak current consumption – 122.121 A
Average current consumption – 50 A

Figure 7.10: Energy and current utilization on intelligent dashboard of smart battery with Battery Management System (BMS) during testing on agricopter



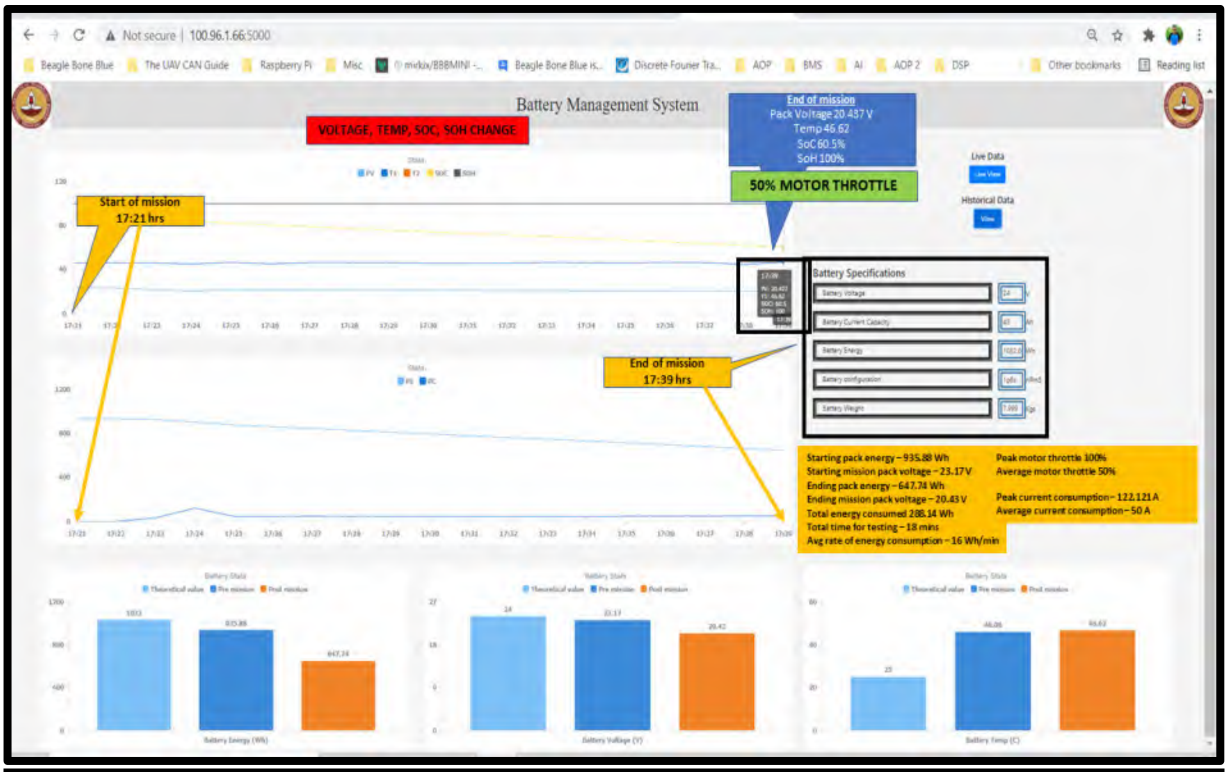
Start of mission
Pack Voltage 23.17 V
Temp 46.08
SoC 87.41%
SoH 100%

Starting pack energy – 935.88 Wh
Starting mission pack voltage – 23.17 V
Ending pack energy – 647.74 Wh
Ending mission pack voltage – 20.43 V
Total energy consumed 288.14 Wh
Total time for testing – 18 mins
Avg rate of energy consumption – 16 Wh/min

Peak motor throttle 100%
Average motor throttle 50%

Peak current consumption – 122.121 A
Average current consumption – 50 A

Figure 7.11: Voltage, SOC, SOH and temperature values on intelligent dashboard of smart battery with Battery Management System (BMS) during testing on Agricopter



End of mission
Pack Voltage 20.437 V
Temp 46.62
SoC 60.5%
SoH 100%

Starting pack energy – 935.88 Wh
Starting mission pack voltage – 23.17V
Ending pack energy – 647.74 Wh
Ending mission pack voltage – 20.43 V
Total energy consumed 288.14 Wh
Total time for testing – 18 mins
Avg rate of energy consumption – 16 Wh/min

Peak motor throttle 100%
Average motor throttle 50%

Peak current consumption – 122.121 A
Average current consumption – 50 A

Figure 7.12: Voltage, SOC, SOH and temperature values on intelligent dashboard of smart battery with Battery Management System (BMS) during testing on agricopter

CHAPTER 8

INTEGRATING ANTENNA ON PACKAGE (AOP) RADAR THROUGH COMPANION COMPUTER LIKE RASPBERRY PI 4 OR JETSON NANO TO FCU

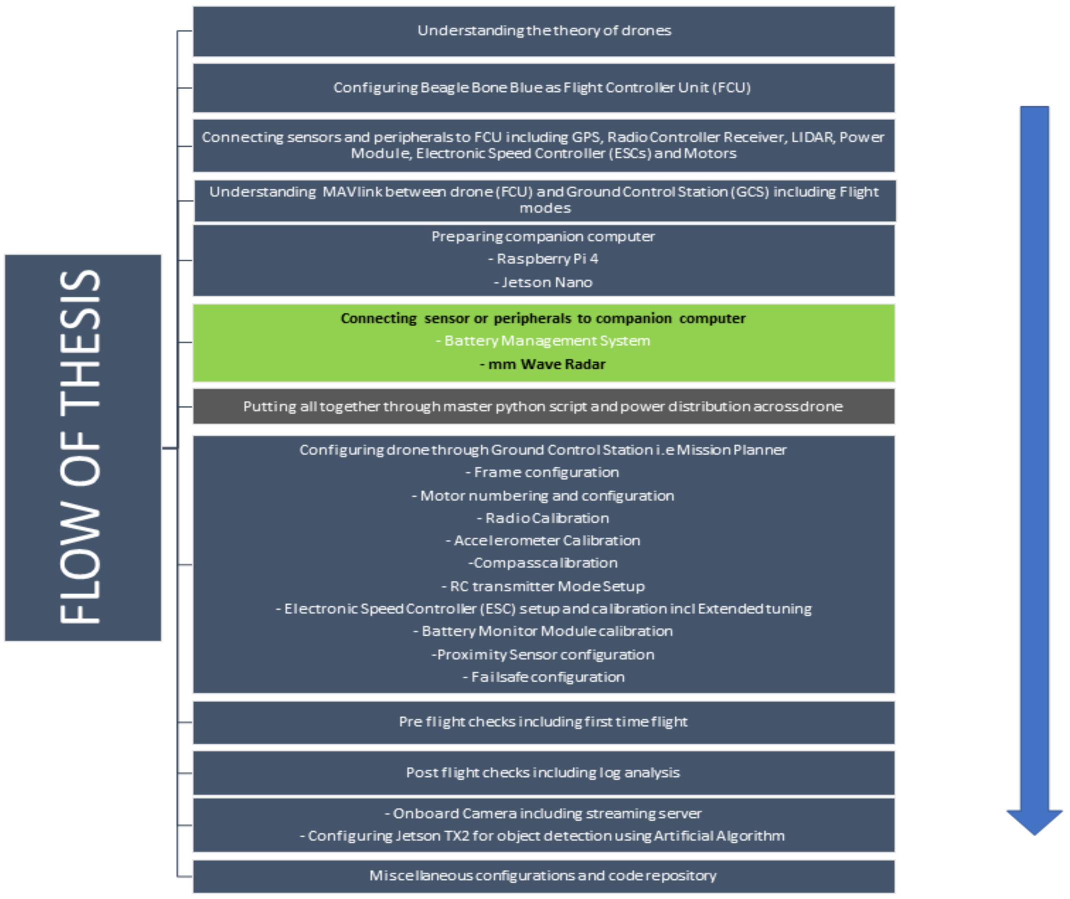

FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer - Raspberry Pi 4 - Jetson Nano
Connecting sensor or peripherals to companion computer - Battery Management System - mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner - Frame configuration - Motor numbering and configuration - Radio Calibration - Accelerometer Calibration - Compass calibration - RC transmitter Mode Setup - Electronic Speed Controller (ESC) setup and calibration incl Extended tuning - Battery Monitor Module calibration - Proximity Sensor configuration - Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server - Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

Figure 8.1: Flow of thesis until now

mmWave Radar IWR6843AOPEVM The IWR6843AOP device is antenna-on-package (AoP) evaluation module (EVM) which is a 60-GHz mmWave sensor evaluation platform. It integrates a wide 120-degrees field-of-view (FoV) antenna and enables access to point-cloud data and power over a USB interface. Examples of its existing applications are:

- Industrial level sensing
- Industrial automation sensor fusion with radar
- Traffic intersection monitoring with radar
- Industrial radar-proximity monitoring
- People counting
- Gesturing

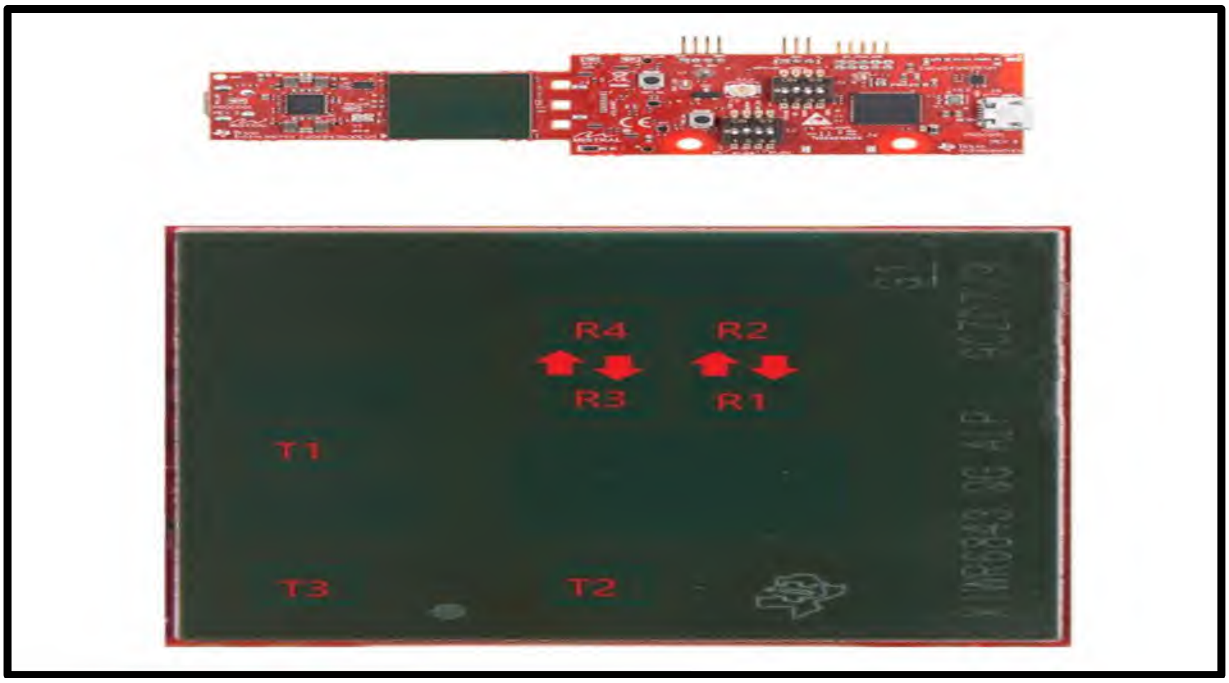


Figure 8.2: Texas Instruments mm Wave radar IWR843AOPEVM picture including antenna closeup

8.1 Understanding the hardware design

RF and Analog Subsystem includes the Radio Frequency and analog circuitry which comprises of the synthesizer, Power Amplifier, Low Noise Amplifier, mixer, IF and Analog to Digital Convertor. It also includes the temperature sensors and crystal oscillator. Two out of the three transmit channels can be operated simultaneously in 1.3-V mode. [1]

Clock Subsystem generates 60-64 GHz from an input reference of 40-MHz crystal. The clock subsystem has a built-in oscillator circuit which is followed by a clean-up PLL and RF synthesizer circuit. The output of the RF synthesizer is then processed by an X3 multiplier to create the required frequency in the 60 to 64 GHz spectrum. The required waveform for sensor operation by modulating the RF synthesizer output by the timing engine block. Thereafter, for the host processor, the clean-up PLL provides a reference clock after system wakeup. There is a built-in mechanism for detecting the presence of a crystal. It is also monitoring the quality of the generated clock. [1]

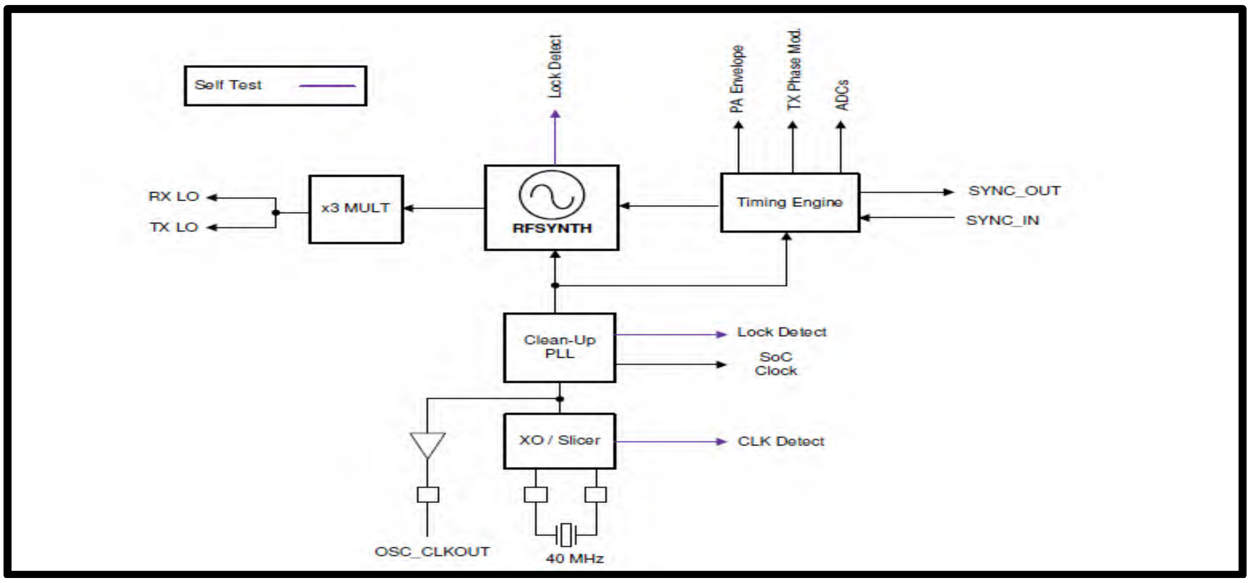


Figure 8.3: Block diagram of Clock Sub System as per datasheet [1]

Transmit Subsystem consists of three parallel transmit chains with each having its own independent amplitude and phase control supporting Transmit Beam forming applications, 6-bit linear phase modulation for Multiple Input Multiple Output (MIMO) radar and also interference mitigation. The programmable backoff supported transmit chains system optimization. [1]

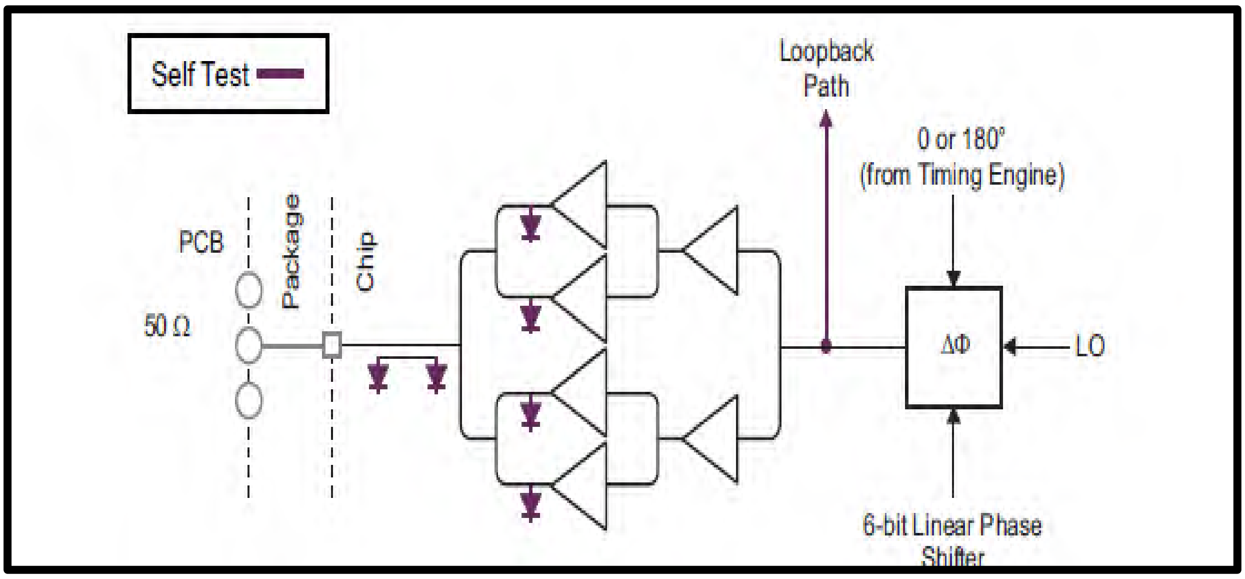


Figure 8.4: Block diagram of Transmission Sub System as per datasheet [1]

Receive Subsystem consists of four parallel channels. Every receive channel consists of an LNA, mixer, IF filtering, A2D conversion, and decimation as a complete unit. All four receive channels can function at the same time while an individual power-down option which ensures system optimization is also available. The complex baseband architecture of this device uses quadrature mixer and dual IF and ADC chains to deliver complex I and Q outputs for each of the receiver channel. It targeted for fast chirp systems. The band-pass IF chain has lower cutoff frequencies above 175 kHz which are configurable. It can support bandwidths up to 10 MHz. [1]

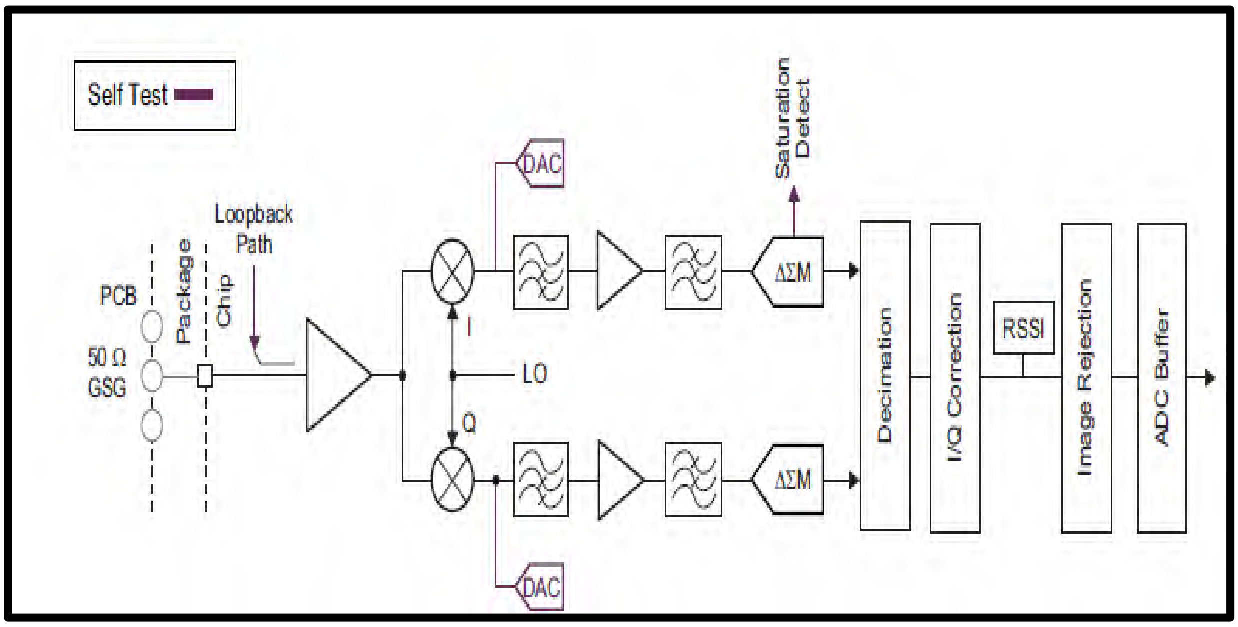


Figure 8.5: Block diagram of Receive Sub System as per datasheet [1]

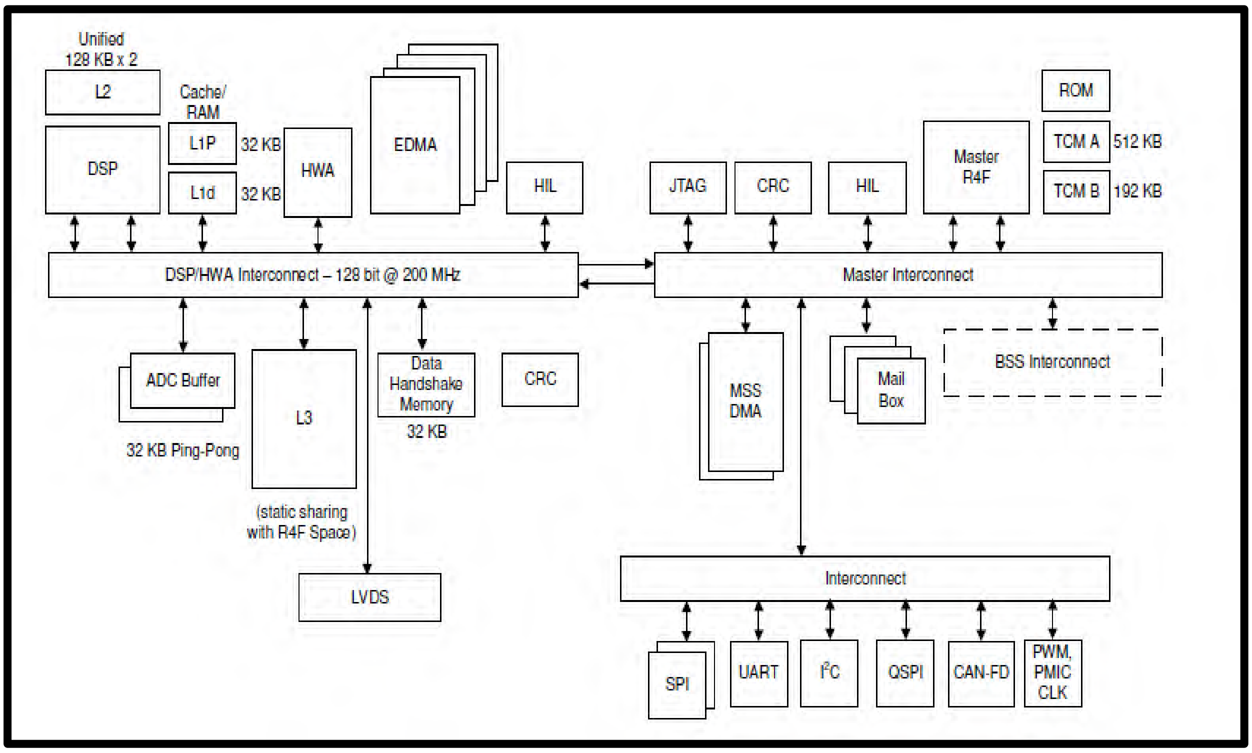


Figure 8.6: Block diagram of Processor Sub System as per datasheet [1]

Processor Subsystem At a high level it has two customer programmable subsystems. It is shown by a dotted line in figure 8.5. Left hand side is the DSP Subsystem which contains Texas Instruments high-performance C674x DSP, hardware accelerator, a high-bandwidth interconnect for high performance (128-bit, 200MHz), and associated peripherals – four DMAs for data transfer, LVDS interface for Measurement data output, L3 Radar data cube memory, ADC buffers, CRC engine, and data handshake memory (additional memory provided on interconnect). The right side shows the Master subsystem which is the master of the device and controls all the device peripherals and house-keeping activities of the device. It contains Cortex-R4F (Master R4F) processor and associated peripherals and housekeeping components such as DMAs, CRC and Peripherals (I2C, UART, SPIs, CAN, PMIC clocking module, PWM, and others) connected to Master Interconnect through Peripheral Central Resource. HIL module is seen in both the subsystems. It is used to perform the radar operations which involves feeding the captured data from outside into the device. It does not involve the RF subsystem. HIL. HIL on DSPSS is for high speed ADC data input while on master SS is for controlling the configuration. Both HIL modules uses the same IOs on the device. One additional IO (DMM_MUX_IN) allows for selecting between the two. **Host Interface** can be provided through a SPI, UART, or CAN-FD interface. [1] The IWR6843AOP device communicates with the host radar processor over the following main interfaces:

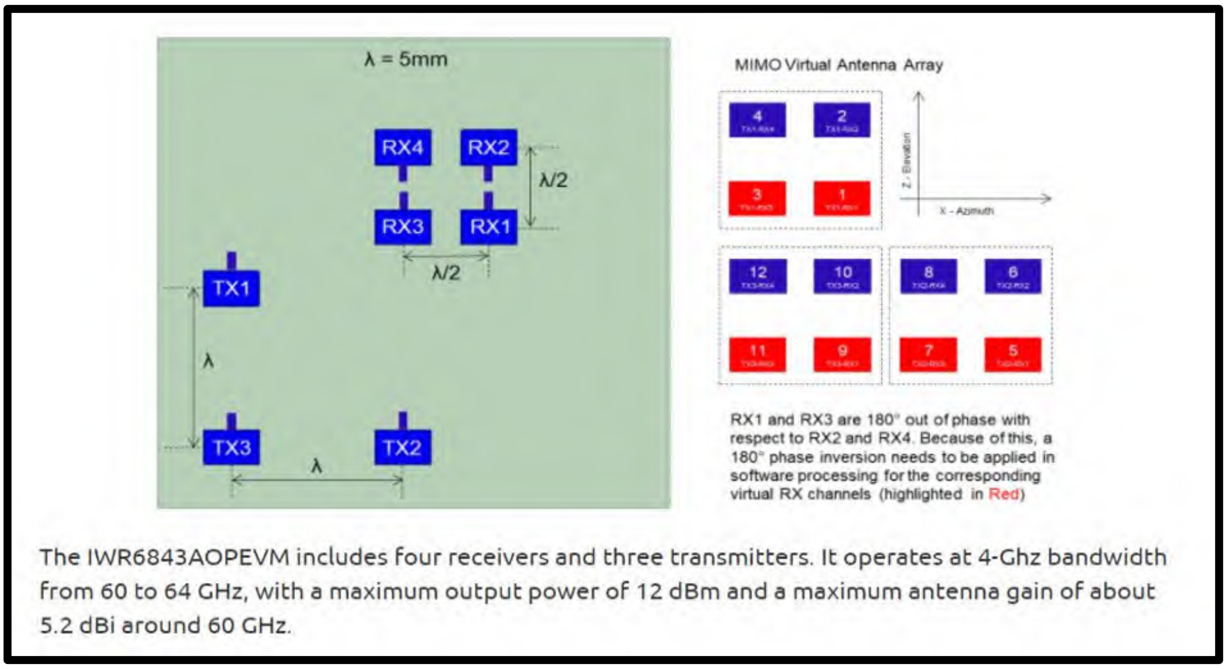
- Reference Clock is available for host processor after device wakeup
- Control – 4-port standard SPI (slave) for host control. All radio control commands (and response) flow through this interface.
- Reset – Active-low reset for device wakeup from host
- Host Interrupt - an indication that the mmwave sensor needs host interface
- Error – Used for notifying the host in case the radio controller detects a fault

Features of the device

- FMCW transceiver consisting of Integrated 4 receivers and 3 transmitters with Antennas-On-Package (AOP), Integrated PLL, transmitter, receiver, Baseband, and A2D. It has 60- to 64-GHz coverage with 4-GHz continuous bandwidth, supports 6-bit phase shifter for TX Beam forming and ultra-accurate chirp engine based on fractional-N PLL
- Built-in calibration and self-test with ARM® Cortex®-R4F-based radio control system, Built-in firmware (ROM) and self-calibrating system across frequency and

- C674x DSP for advanced signal processing
- Memory compression
- Hardware accelerator for FFT, filtering, and CFAR processing
- ARM-R4F microcontroller for object detection, and interface control
- Supports autonomous mode (loading user application from QSPI flash memory)
- Internal memory with ECC 1.75 MB, divided into MSS program RAM (512 KB), MSS data RAM (192 KB), DSP L1 RAM (64KB) and L2 RAM (256 KB), and L3 radardata cube RAM (768 KB)
- Other interfaces available to user application supports upto 6 ADC channels (low sample rate monitoring), 2 SPI ports, 2 UARTs and 1 CAN-FD interface, I2C and GPIOs. It also has 2 lane LVDS interface for raw ADC data
- Power management with Built-in LDO network for enhanced PSRR and I/Os support dual voltage 3.3 V/1.8 V
- Clock source 40.0 MHz crystal with internal oscillator and supports external oscillator at 40 MHz. It also supports externally driven clock (square/sine) at 40 MHz
- Easy hardware design with 0.8-mm pitch, 180-pin 15 mm × 15 mm FCBGA package (ALP) for easy assembly and low-cost PCB design ensuring small form factor
- Operated in the range of –40ºC to 105ºC [1]

Antenna Positions and Radiation patterns



RX1 and RX3 are 180° out of phase with respect to RX2 and RX4. Because of this, a 180° phase inversion needs to be applied in software processing for the corresponding virtual RX channels (highlighted in Red)

The IWR6843AOPEVM includes four receivers and three transmitters. It operates at 4-Ghz bandwidth from 60 to 64 GHz, with a maximum output power of 12 dBm and a maximum antenna gain of about 5.2 dBi around 60 GHz.

Figure 8.7: IWR6843AOPEVM mm Wave radar antenna positions [1]

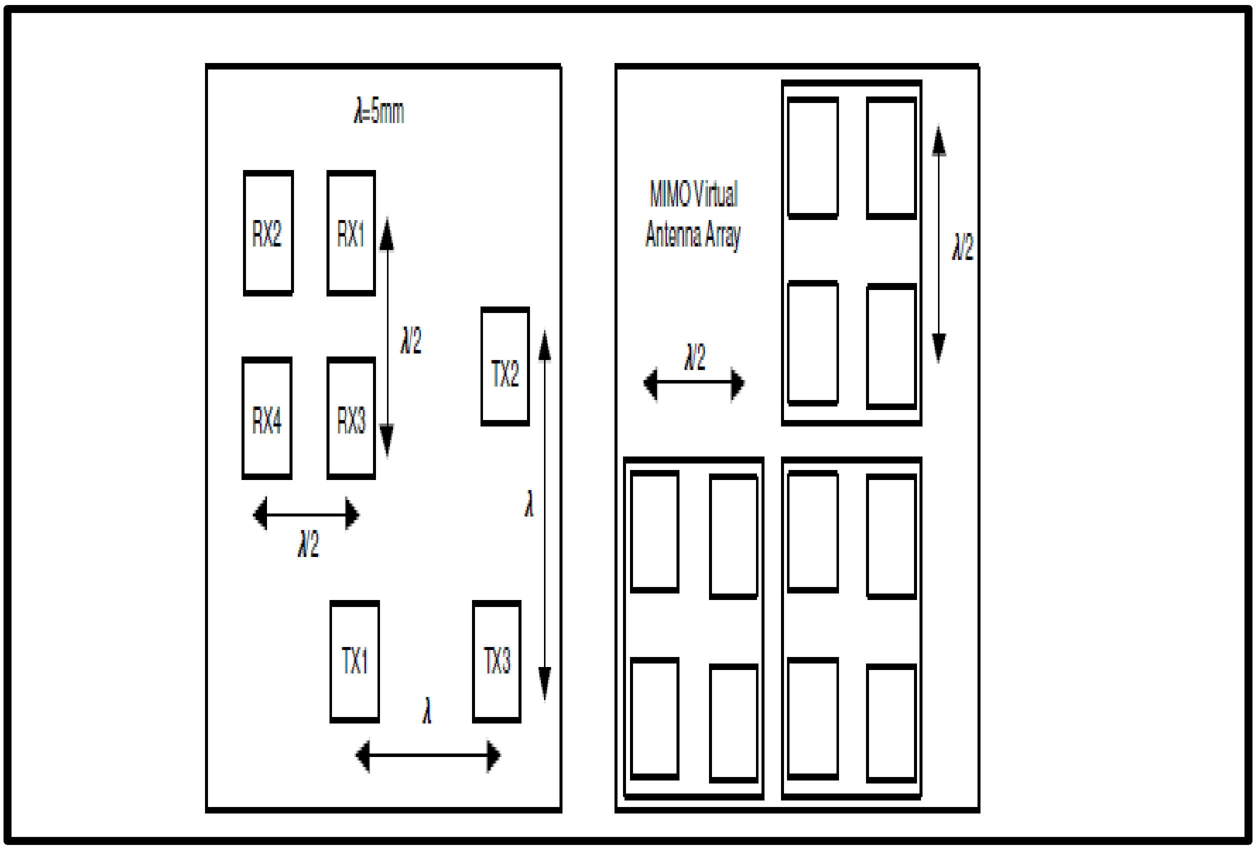


Figure 8.8: IWR6843AOPEVM mm Wave radar antenna positions [1]

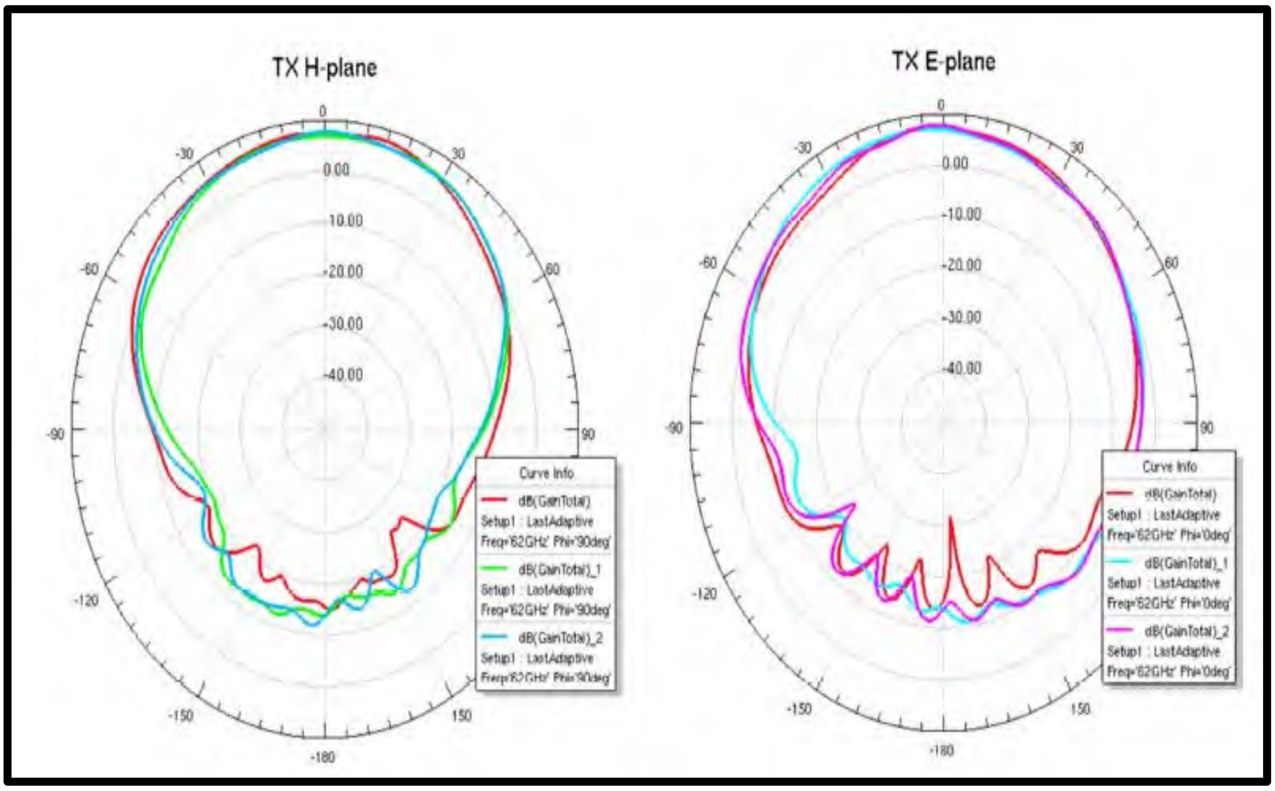


Figure 8.9: IWR6843AOPEVM mm Wave radar Transmission antenna radiation patterns [1]

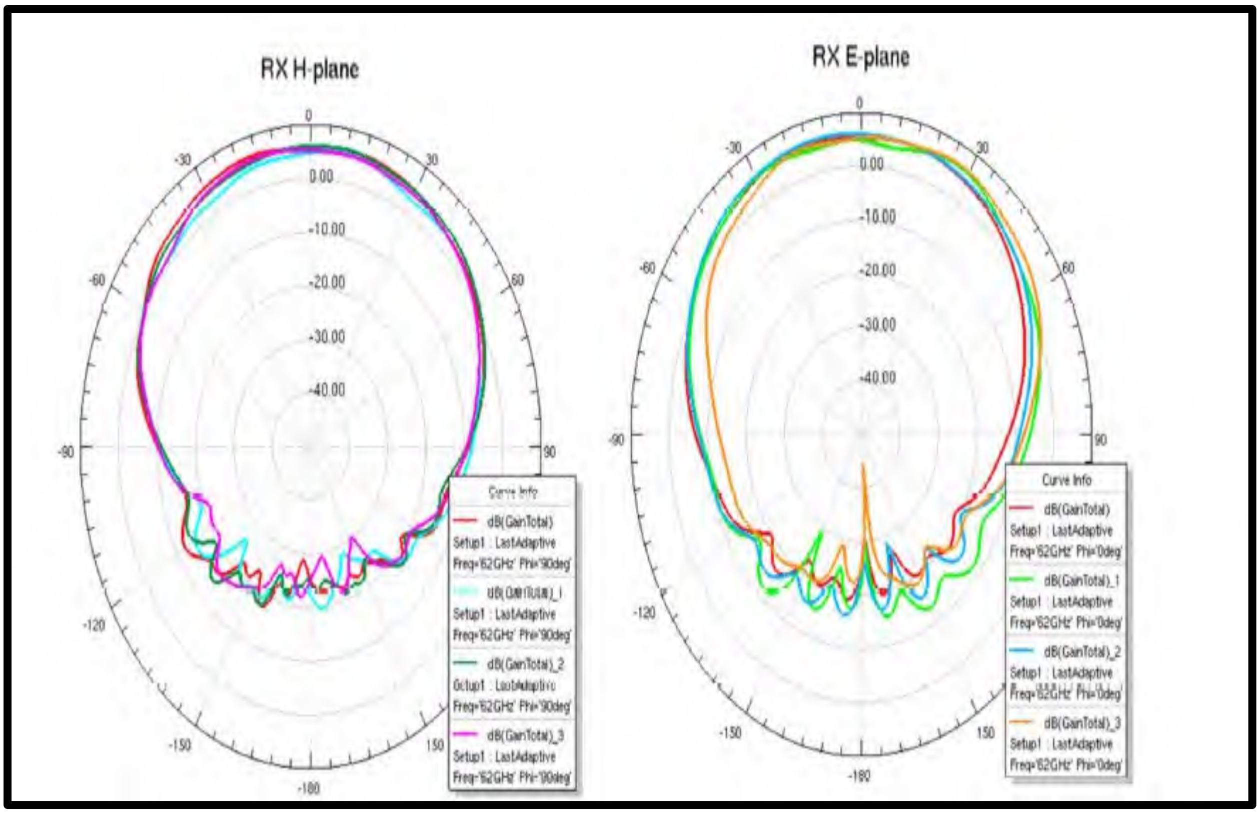


Figure 8.10: IWR6843AOPEVM mm Wave radar receive antenna radiation patterns [1]

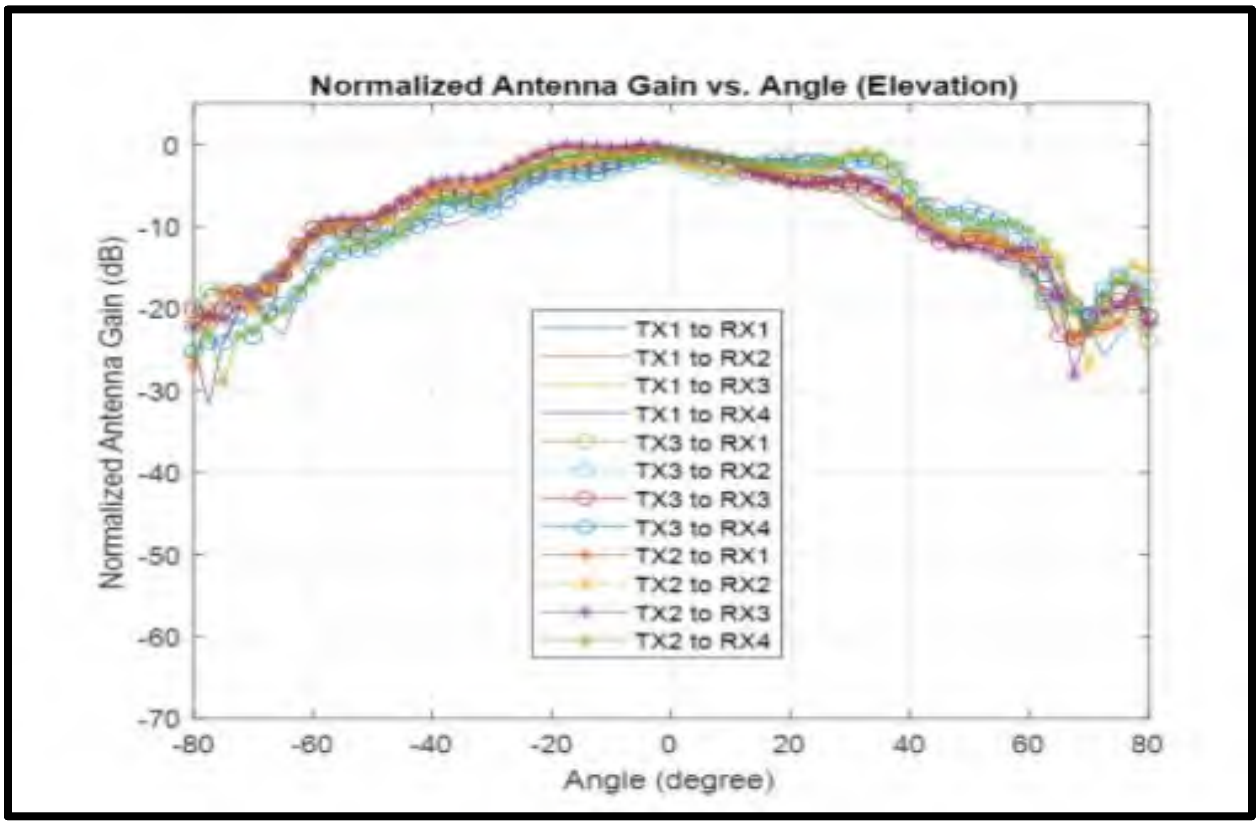


Figure 8.11: IWR6843AOPEVM mm Wave radar Normalized Antenna Gain vs Angle (Elevation) [1]

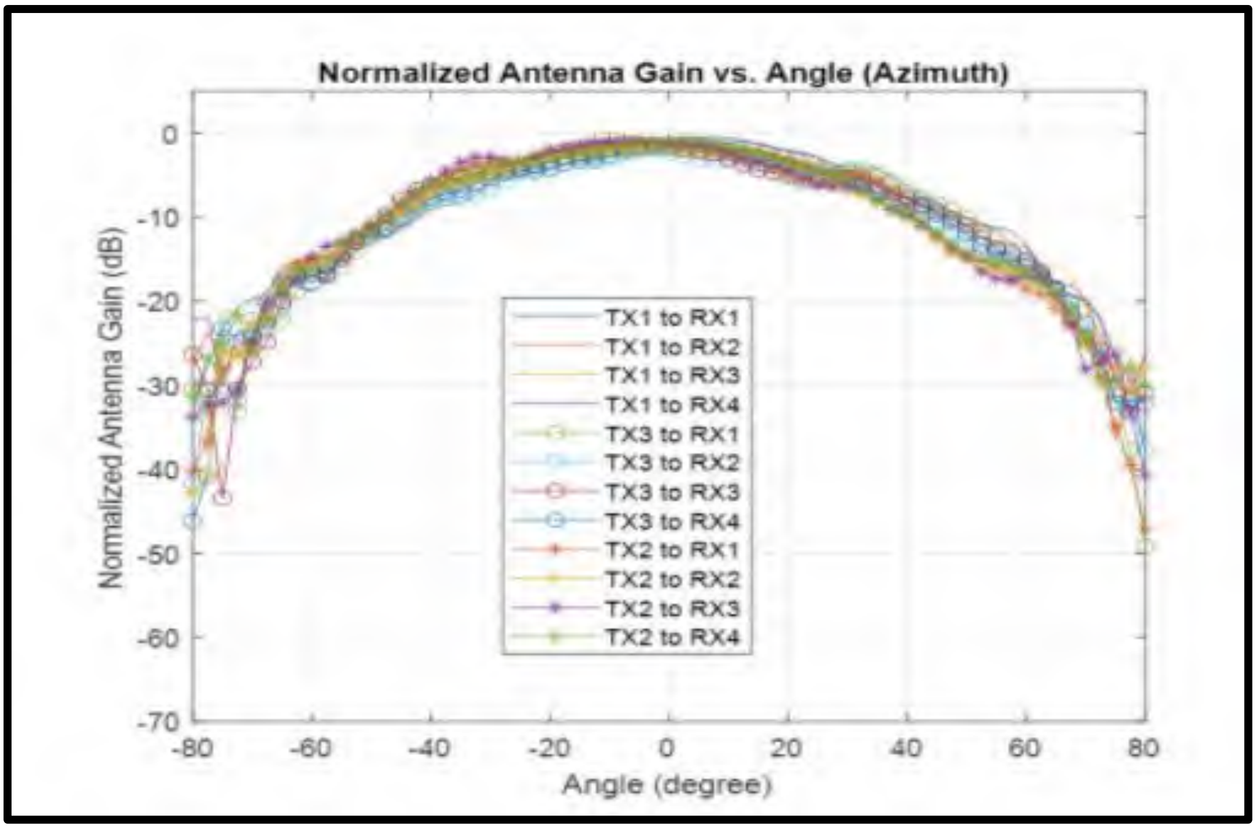


Figure 8.12: IWR6843AOPEVM mm Wave radar Normalized Antenna Gain vs Angle (Azimuthal) [1]

8.2 mmWave radar integration with companion computer

The mmWave antenna is to be connected to the companion computer via USB port. This USB interface supports two UART links, one for configuring the radar and other for data transfer. The configuration file needs to be created via Texas Instruments visualizer and .cfg file then can be saved on the computer. Now we need to send the configuration file each time the mmWave radar starts and then data starts flowing which has to be interpreted and then forwarded to Ardupilot running on Flight Controller Unit. Hence, python script does this job of sending configuration file to mm Wave radar to open sensor and start receiving data. The incoming data is sliced and interpreted. This logical data is sent to MAVPROXY by the python script. The MAVPROXY Command Line Ground Control Software (GCS) packetizes the data as part of MAVlink to send it as proximity sensor data to Flight Controller Unit which in my case is Beagle Bone Blue. This setup can work with any FCU with companion computer.

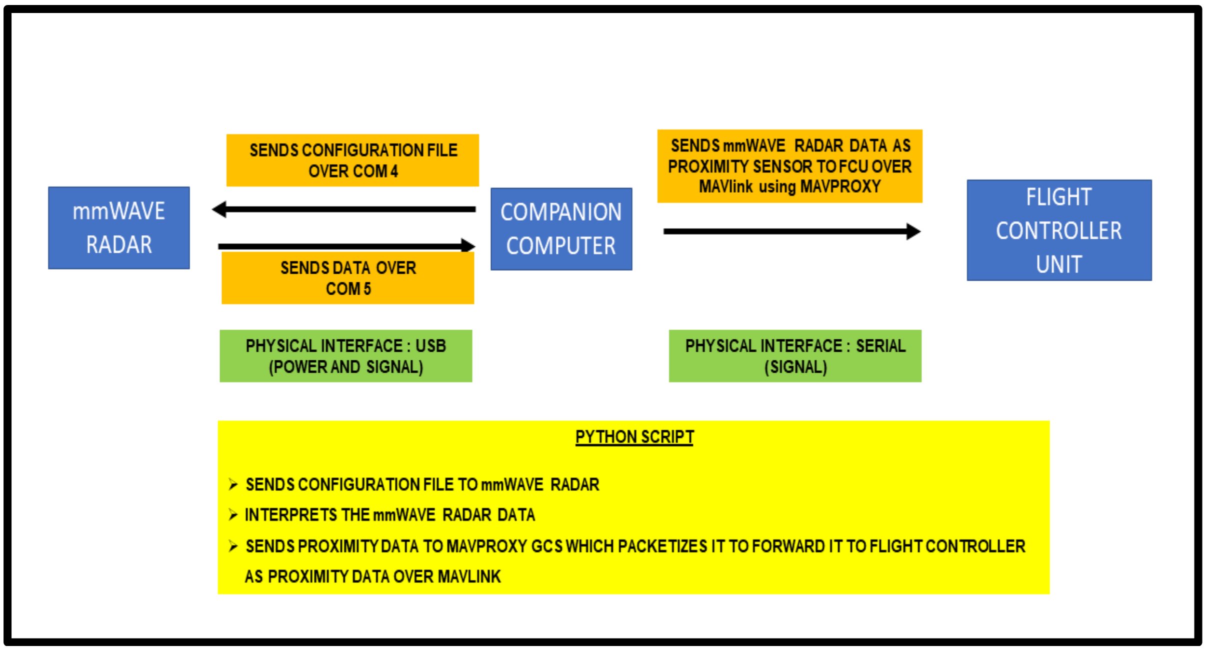

SENDS CONFIGURATION FILE OVER COM 4
SENDS mmWAVE RADAR DATA AS PROXIMITY SENSOR TO FCU OVER MAVlink using MAVPROXY
mmWAVE RADAR
COMPANION COMPUTER
FLIGHT CONTROLLER UNIT
SENDS DATA OVER COM 5
PHYSICAL INTERFACE : USB (POWER AND SIGNAL)
PHYSICAL INTERFACE : SERIAL (SIGNAL)
PYTHON SCRIPT
SENDS CONFIGURATION FILE TO mmWAVE RADAR
INTERPRETS THE mmWAVE RADAR DATA
SENDS PROXIMITY DATA TO MAVPROXY GCS WHICH PACKETIZES IT TO FORWARD IT TO FLIGHT CONTROLLER AS PROXIMITY DATA OVER MAVLINK

Figure 8.13: IWR6843AOPEVM mm Wave radar data flow from companion computer to Beagle Bone Blue FCU

Implementation steps In order for mmWave radar to successfully communicate with the companion computer, we need to carry out below mentioned actions.

We need to install latest python and pip3 package on companion computer. I have built Python 3.7 from source on my companion computer. To build Python package from source.

sudo apt update

sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev curl libbz2-dev

Download the latest python release source code using curl command:

curl -O https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tar.xz

After download, we need to extract the tarball :

tar -xf Python-3.7.3.tar.xz

Navigate to the Python source directory and run the configure script which performs number of checks to ensure all the dependencies are available on system. The optimizations option runs multiple tests to check system readiness but makes the process slower.

```
cd Python-3.7.3
./configure --enable-optimizations
```

Run make to start the build process:

```
make -j 4
```

To achieve quicker build time, we can modify the -j flag according to the processor. Depends on number of cores and for the processor how many cores it has, can be found by typing the command nproc. Once the build is done we install the Python binaries.

```
sudo make altinstall
```

We must avoid using standard make install as it will overwrite the default system python3 binary which we don’t want. At this point, Python 3.7 is installed on my Debian system and is ready. We can verify it

```
python3 --version
Python 3.7.3
```

Next we want to install pip3. First we update the package list for Linux OS

```
sudo apt update
```

Next, we install pip for Python 3 with all of its dependencies

```
sudo apt install python3-pip
```

To erify the installation

```
pip3 --version
```

The version number could vary something like below. Hence we must upgrade pip.

```
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.5)
```

Upgrade pip package

```
python3 -m pip install –upgrade
```

Install dependencies

```
pip3 install pyserial
```

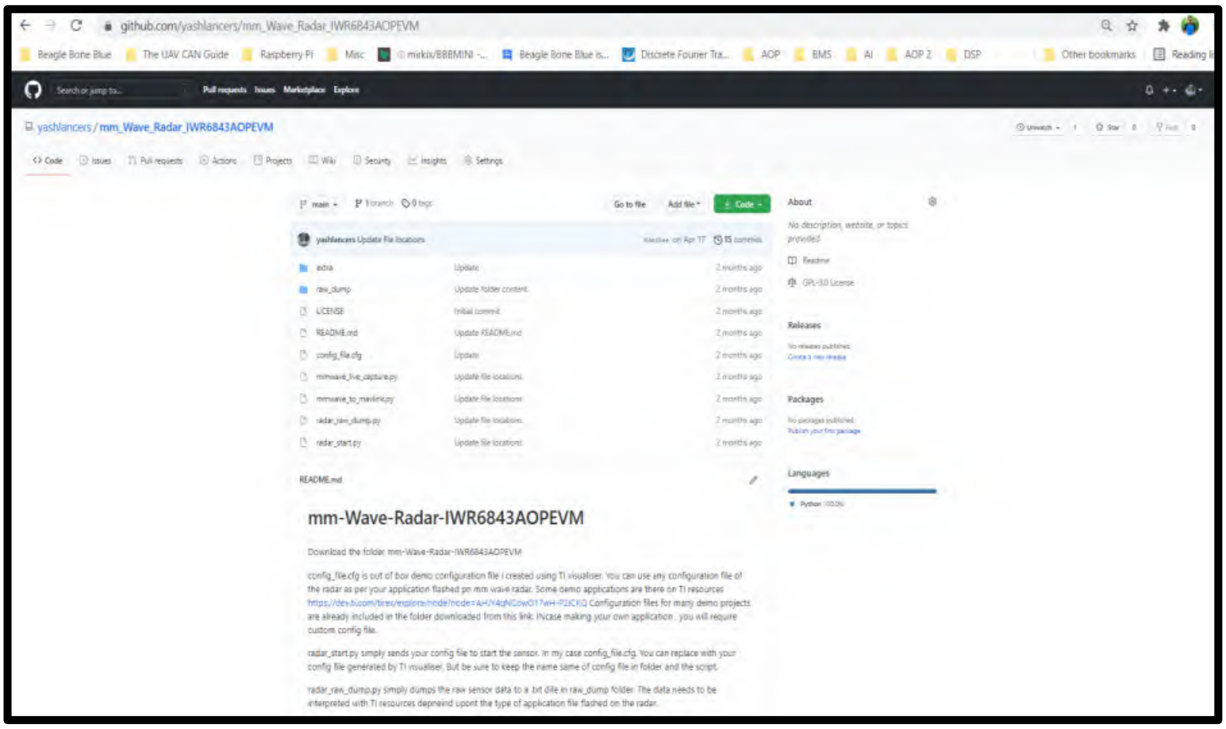
```
pip3 install apscheduler
pip3 install numpy
sudo apt-get install libatlas-base-dev
```

Getting mavlink ready (uninstall pymavlink older version if any and install latest version)

```
pip3 uninstall pymavlink
git clone https://github.com/ArduPilot/ardupilot
cd ardupilot
git branch master
git submodule update --init –recursive
cd /ardupilot/modules/mavlink/pymavlink
python3 setup.py install --user
```

Get mm wave radar code folder from my github repository

```
git clone https://github.com/yashlancers/mm_Wave_Radar_IWR6843AOPEVM.git
```



github.com/yashlancers/mm_Wave_Radar_IWR6843AOPEVM

yashlancers / mm_Wave_Radar_IWR6843AOPEVM

yashlancers Update file locations — on Apr 17 — 15 commits

extra	Update	2 months ago
raw_dump	Update folder content	2 months ago
LICENSE	Initial commit	2 months ago
README.md	Update README.md	2 months ago
config_file.cfg	Update	2 months ago
mmwave_live_capture.py	Update file locations	2 months ago
mmwave_to_mavlink.py	Update file locations	2 months ago
radar_raw_dump.py	Update file locations	2 months ago
radar_start.py	Update file locations	2 months ago

About: No description, website, or topics provided. Readme. GPL-3.0 License.

Releases: No releases published. Create a new release

Packages: No packages published. Publish your first package

Languages: Python 100.0%

README.md

mm-Wave-Radar-IWR6843AOPEVM

Download the folder mm-Wave-Radar-IWR6843AOPEVM

config_file.cfg is out of box demo configuration file i created using TI visualiser. You can use any configuration file of the radar as per your application flashed pn mm wave radar. Some demo applications are there on TI resources https://dev.ti.com/tirex/explore/node?node=AHJY4qNCowO17wH-P2lCKQ Configuration files for many demo projects are already included in the folder downloaded from this link. INcase making your own application , you will require custom config file.

radar_start.py simply sends your config file to start the sensor. In my case config_file.cfg. You can replace with your config file generated by TI visualiser. But be sure to keep the name same of config file in folder and the script.

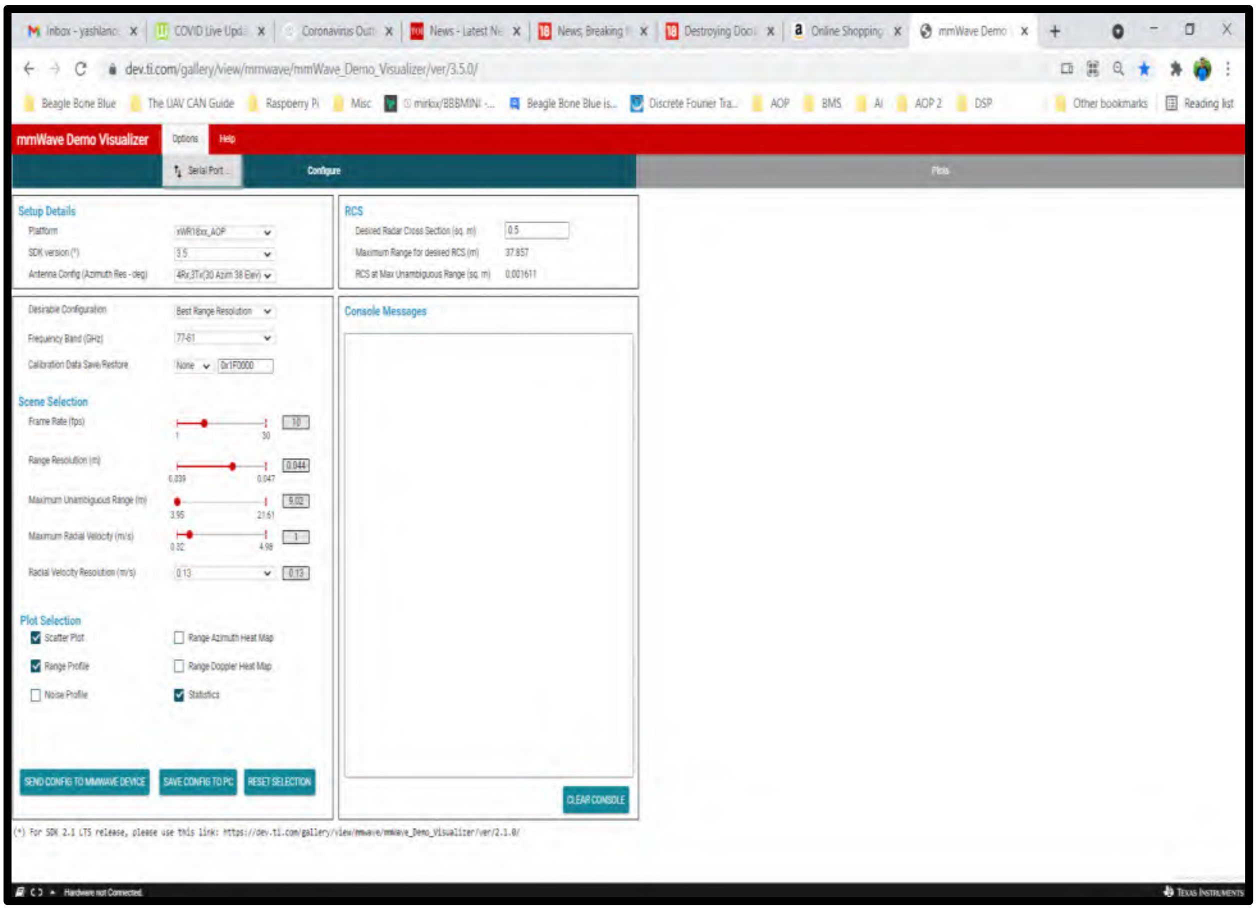
radar_raw_dump.py simply dumps the raw sensor data to a .txt dile in raw_dump folder. The data needs to be interpreted with TI resources depneind upont the type of application file flashed on the radar.

Figure 8.14: Downloading IWR6843AOPEVM mm Wave radar working repository from my github account containing all codes including mmWave radar configuration file

8.3 Creating mmWave radar configuration file

Configuration file for mmWave radar from TI visualizer (optional) The github repository that we downloaded contains the configuration file .cfg that's tried and tested for proximity sensor usage of mmWave radar. However, incase it needs to be generated again keeping any specific requirement in mind, the process below explains it with details. We will use the Texas Instruments visualizer to create configuration file for the mmWave radar. This action has to be done once. It has to be repeated only when we want to make changes to parameters of mmWave radar. Using Google Chrome and access **https://dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/3.5.0/** If prompted, follow the on-screen instructions for installing TI Cloud Agent (this is needed the first time on a new PC)

In the GUI menu, select Options → Serial Port



dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/3.5.0/

Setup Details

Platform	xWR18xx_AOP
SDK version (*)	3.5
Antenna Config (Azimuth Res - deg)	4Rx,3Tx(30 Azim 38 Elev)
Desirable Configuration	Best Range Resolution
Frequency Band (GHz)	77-81
Calibration Data Save/Restore	None 0x1F0000

Scene Selection

Frame Rate (fps)	1	30	10
Range Resolution (m)	0.039	0.047	0.044
Maximum Unambiguous Range (m)	3.95	21.61	9.02
Maximum Radial Velocity (m/s)	0.32	4.98	1
Radial Velocity Resolution (m/s)	0.13		0.13

Plot Selection

☑ Scatter Plot ☐ Range Azimuth Heat Map
☑ Range Profile ☐ Range Doppler Heat Map
☐ Noise Profile ☑ Statistics

RCS

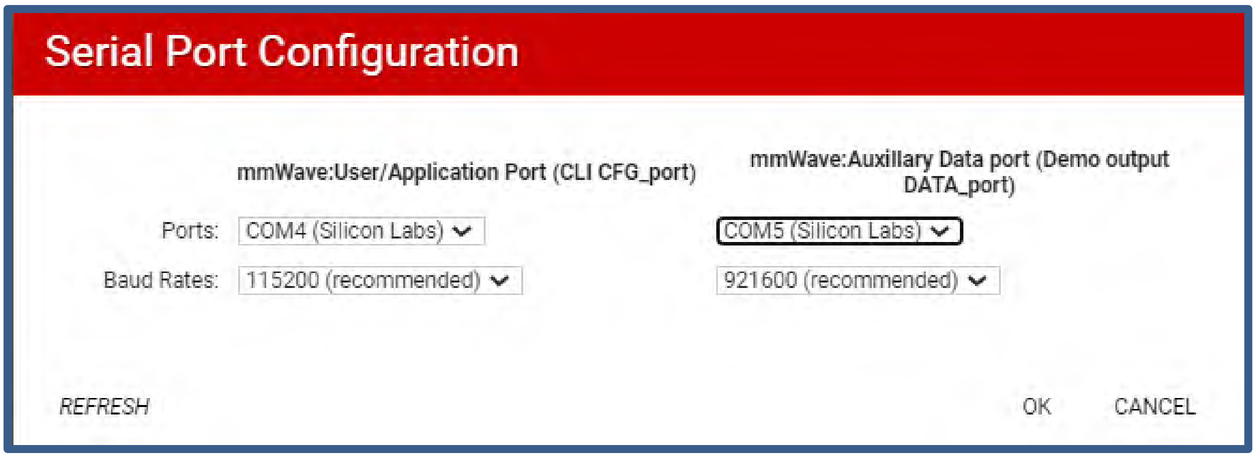
Desired Radar Cross Section (sq. m)	0.5
Maximum Range for desired RCS (m)	37.857
RCS at Max Unambiguous Range (sq. m)	0.001611

Console Messages

(*) For SDK 2.1 LTS release, please use this link: https://dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/2.1.0/

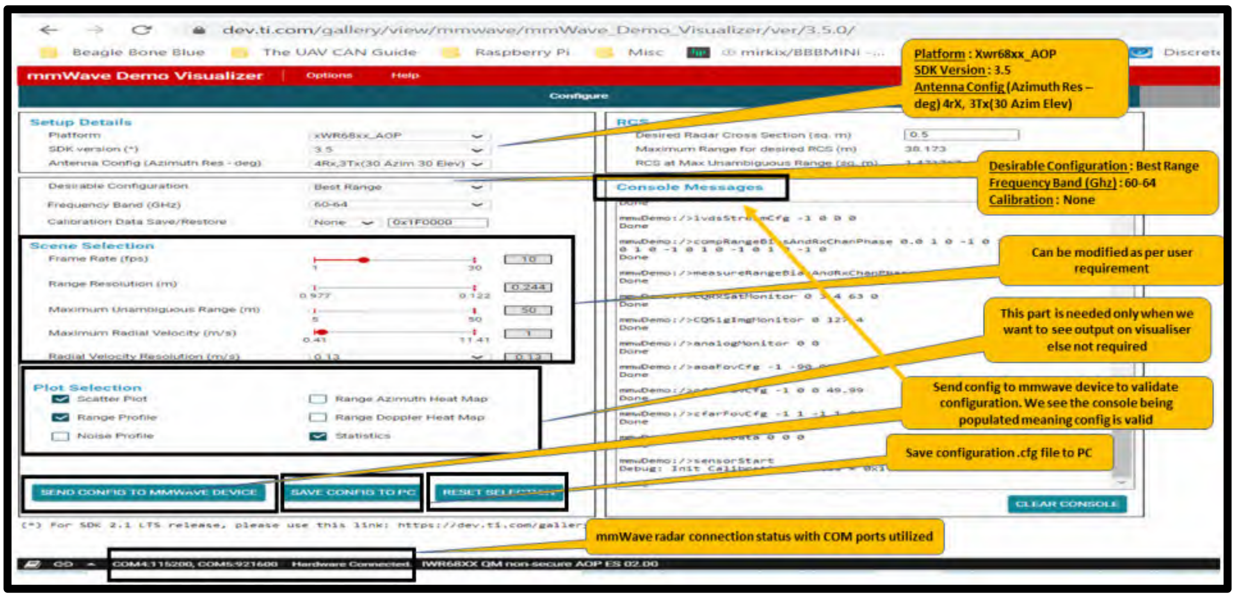
Figure 8.15: Using Texas Instruments visualizer to connect to IWR6843AOPEVM mm Wave radar for generating radar configuration file

In the serial port window, COM4 for configuration (11500 bauds) and COM5 (921600) for data.


Serial Port Configuration

	mmWave:User/Application Port (CLI CFG_port)	mmWave:Auxillary Data port (Demo output DATA_port)
Ports:	COM4 (Silicon Labs)	COM5 (Silicon Labs)
Baud Rates:	115200 (recommended)	921600 (recommended)

Figure 8.16: COM Port selection for connecting to mm Wave radar IWR6843AOPEVM though Texas Instruments visualizer tool


dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/3.5.0/

Setup Details

Platform	xWR68xx_AOP
SDK version (*)	3.5
Antenna Config (Azimuth Res - deg)	4Rx,3Tx(30 Azim 30 Elev)
Desirable Configuration	Best Range
Frequency Band (GHz)	60-64
Calibration Data Save/Restore	None 0x1F0000

Scene Selection

Frame Rate (fps)	1	30	10
Range Resolution (m)	0.977	0.122	0.244
Maximum Unambiguous Range (m)	5	50	50
Maximum Radial Velocity (m/s)	0.41	11.41	1
Radial Velocity Resolution (m/s)	0.13		0.13

Plot Selection

- ✓ Scatter Plot
- ✓ Range Profile
- ☐ Noise Profile
- ☐ Range Azimuth Heat Map
- ☐ Range Doppler Heat Map
- ✓ Statistics

RCS

Desired Radar Cross Section (sq. m)	0.5
Maximum Range for desired RCS (m)	38.173
RCS at Max Unambiguous Range (sq. m)	1.4

Console Messages

```
Done
mmwDemo:/>lvdsStreamCfg -1 0 0 0
Done
mmwDemo:/>compRangeBiasAndRxChanPhase 0.0 1 0 -1 0
0 1 0 -1 0 1 0 -1 0 1 0 -1 0
Done
mmwDemo:/>measureRangeBiasAndRxChanPhase
Done
mmwDemo:/>CQRxSatMonitor 0 3 4 63 0
Done
mmwDemo:/>CQSigImgMonitor 0 127 4
Done
mmwDemo:/>analogMonitor 0 0
Done
mmwDemo:/>aoaFovCfg -1 -90
Done
mmwDemo:/>cfarFovCfg -1 0 0 49.99
Done
mmwDemo:/>cfarFovCfg -1 1 -1
Done
mmwDemo:/>                  Data 0 0 0
mmwDemo:/>sensorStart
Debug: Init Calibration                 0x1
```

(*) For SDK 2.1 LTS release, please use this link: https://dev.ti.com/galler

COM4:115200, COM5:921600 Hardware Connected. IWR68XX QM non-secure AOP ES 02.00

Platform : Xwr68xx_AOP
SDK Version : 3.5
Antenna Config (Azimuth Res – deg) 4rX, 3Tx(30 Azim Elev)

Desirable Configuration : Best Range
Frequency Band (Ghz) : 60-64
Calibration : None

Can be modified as per user requirement

This part is needed only when we want to see output on visualiser else not required

Send config to mmwave device to validate configuration. We see the console being populated meaning config is valid

Save configuration .cfg file to PC

mmWave radar connection status with COM ports utilized

Figure 8.17: Understanding Texas Instruments visualizer for generating radar configuration file for IWR6843AOPEVM mm Wave radar

The configuration file can be generated by selecting save to PC. The configuration file is already part of the repository that we downloaded from github. We can even see the output of radar on Texas Instruments visualizer as standalone sensor

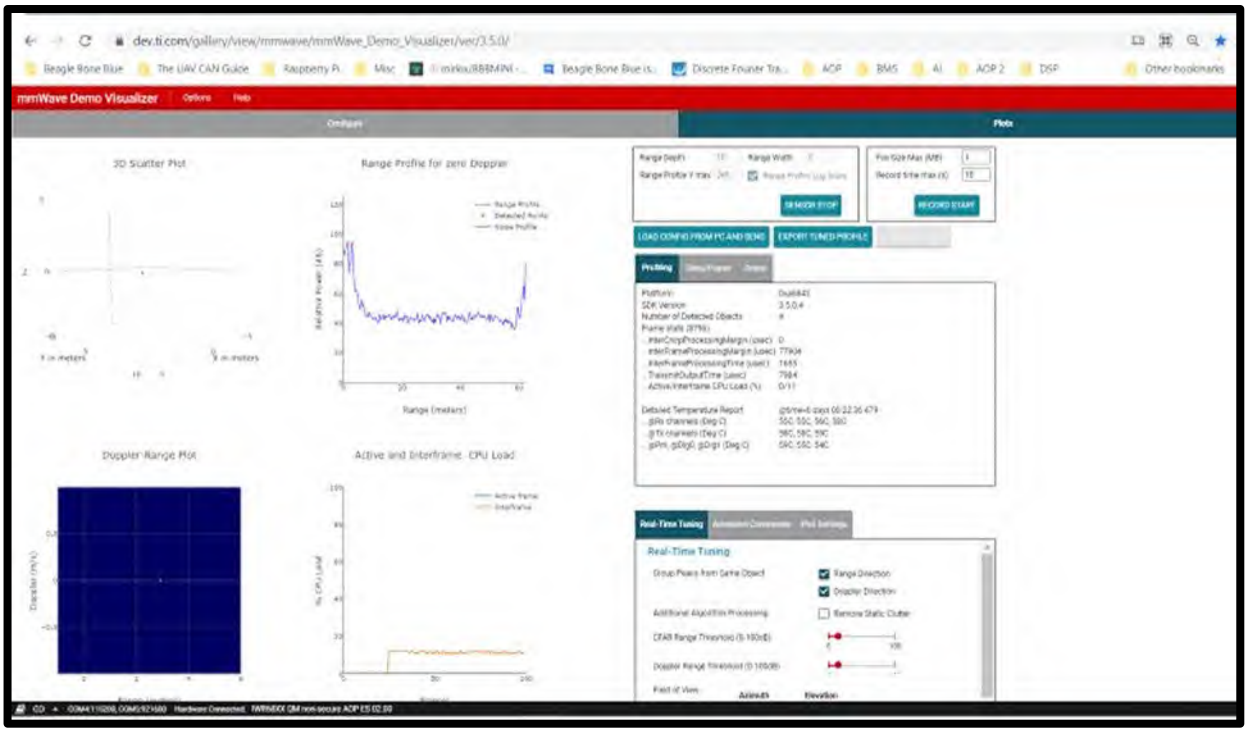


Figure 8.18: Output of IWR6843AOPEVM mm Wave radar on dashboard of Texas Instruments Visualizer application/tool

Testing python script to open sensor (mmWave radar) and get sliced data output. Connect mmWave radar to companion computer using USB port. Make sure that the interface details (COM Ports in Linux) are updated correctly in python script. It can be checked with

sudo ls /dev/ttyUSB*

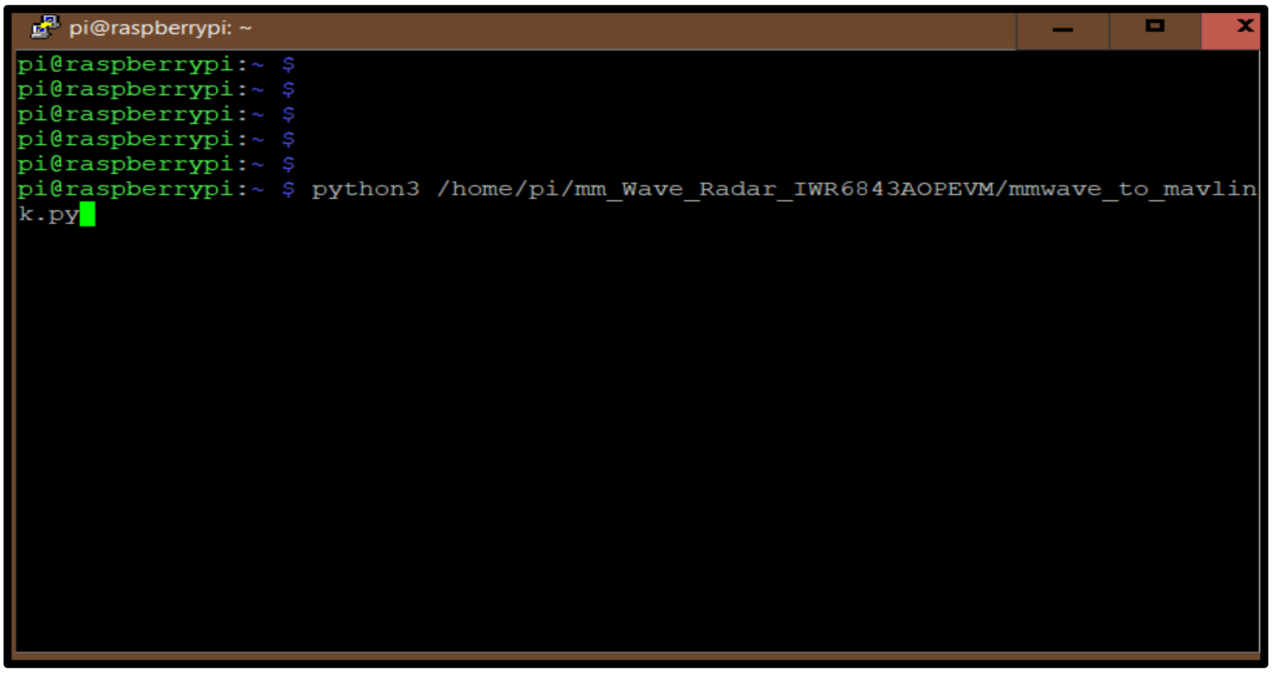
Incase of Raspberry Pi 4

python3 /home/pi/mm_Wave_Radar_IWR6843AOPEVM/mmwave_to_mavlink.py

Incase of Jetson Nano

python3 /home/yash/mm_Wave_Radar_IWR6843AOPEVM/mmwave_to_mavlink.py

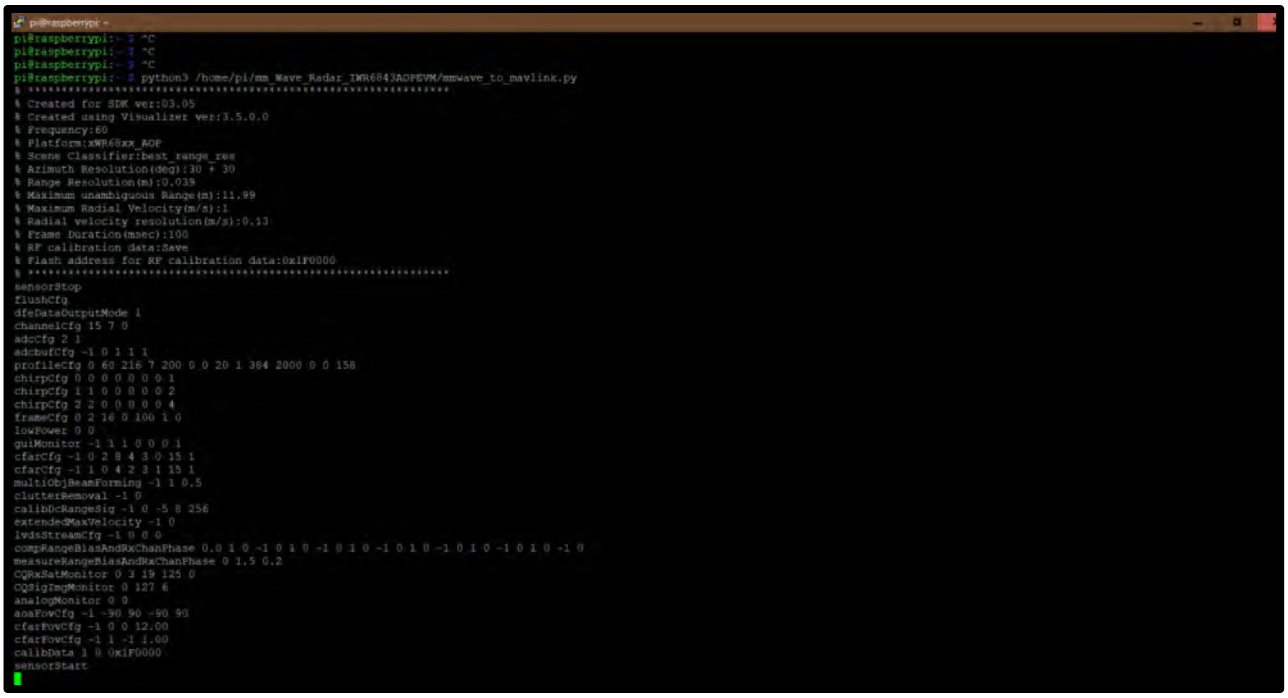
Replace yash by your username



```
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $
pi@raspberrypi:~ $ python3 /home/pi/mm_Wave_Radar_IWR6843AOPEVM/mmwave_to_mavlin
k.py
```

Figure 8.19: Running python script from the downloaded repository from github on caompanion computer to check output

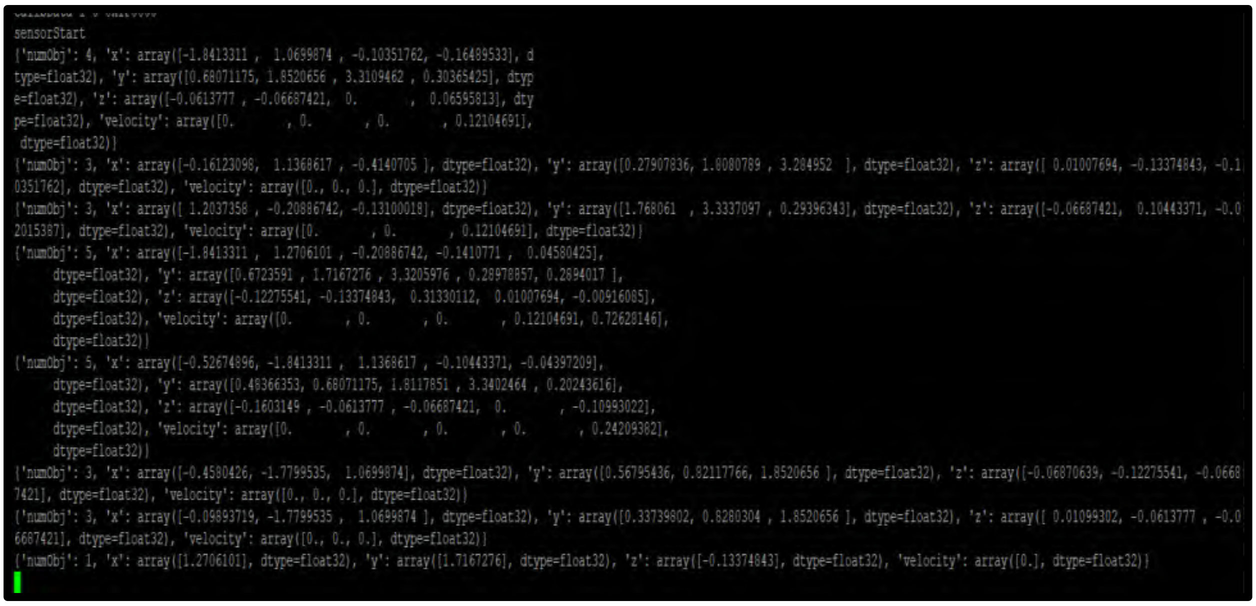
First the script mmWave_to_mavlink.py sends configuration to mmWave radar to start/open the sensor



```
pi@raspberrypi:~ $ ^C
pi@raspberrypi:~ $ ^C
pi@raspberrypi:~ $ ^C
pi@raspberrypi:~ $ python3 /home/pi/mm_Wave_Radar_IWR6843AOPEVM/mmwave_to_mavlink.py
% ***************************************************************
% Created for SDK ver:03.05
% Created using Visualizer ver:3.5.0.0
% Frequency:60
% Platform:xWR68xx_AOP
% Scene Classifier:best_range_res
% Azimuth Resolution(deg):30 + 30
% Range Resolution(m):0.039
% Maximum unambiguous Range(m):11.99
% Maximum Radial Velocity(m/s):1
% Radial velocity resolution(m/s):0.13
% Frame Duration(msec):100
% RF calibration data:Save
% Flash address for RF calibration data:0x1F0000
% ***************************************************************
sensorStop
flushCfg
dfeDataOutputMode 1
channelCfg 15 7 0
adcCfg 2 1
adcbufCfg -1 0 1 1 1
profileCfg 0 60 216 7 200 0 0 20 1 384 2000 0 0 158
chirpCfg 0 0 0 0 0 0 0 1
chirpCfg 1 1 0 0 0 0 0 2
chirpCfg 2 2 0 0 0 0 0 4
frameCfg 0 2 16 0 100 1 0
lowPower 0 0
guiMonitor -1 1 1 0 0 0 1
cfarCfg -1 0 2 8 4 3 0 15 1
cfarCfg -1 1 0 4 2 3 1 15 1
multiObjBeamForming -1 1 0.5
clutterRemoval -1 0
calibDcRangeSig -1 0 -5 8 256
extendedMaxVelocity -1 0
lvdsStreamCfg -1 0 0 0
compRangeBiasAndRxChanPhase 0.0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0
measureRangeBiasAndRxChanPhase 0 1.5 0.2
CQRxSatMonitor 0 3 19 125 0
CQSigImgMonitor 0 127 6
analogMonitor 0 0
aoaFovCfg -1 -90 90 -90 90
cfarFovCfg -1 0 0 12.00
cfarFovCfg -1 1 -1 1.00
calibData 1 0 0x1F0000
sensorStart
```

Figure 8.20: Configuration for opening IWR6843AOPEVM mm Wave radar sent by mmWave_to_mavlink.py python script

Python script reading data in terms of no of objects detected, x,y,z coordinates , velocity, acceleration and azimuth angle.



```
sensorStart
{'numObj': 4, 'x': array([-1.8413311 ,  1.0699874 , -0.10351762, -0.16489533], d
type=float32), 'y': array([0.68071175, 1.8520656 , 3.3109462 , 0.30365425], dtyp
e=float32), 'z': array([-0.0613777 , -0.06687421,  0.        ,  0.06595813], dty
pe=float32), 'velocity': array([0.        , 0.        , 0.        , 0.12104691],
 dtype=float32)}
{'numObj': 3, 'x': array([-0.16123098,  1.1368617 , -0.4140705 ], dtype=float32), 'y': array([0.27907836, 1.8080789 , 3.284952  ], dtype=float32), 'z': array([ 0.01007694, -0.13374843, -0.1
0351762], dtype=float32), 'velocity': array([0., 0., 0.], dtype=float32)}
{'numObj': 3, 'x': array([ 1.2037358 , -0.20886742, -0.13100018], dtype=float32), 'y': array([1.768061  , 3.3337097 , 0.29396343], dtype=float32), 'z': array([-0.06687421,  0.10443371, -0.0
2015387], dtype=float32), 'velocity': array([0.        , 0.        , 0.12104691], dtype=float32)}
{'numObj': 5, 'x': array([-1.8413311 ,  1.2706101 , -0.20886742, -0.1410771 ,  0.04580425],
      dtype=float32), 'y': array([0.6723591 , 1.7167276 , 3.3205976 , 0.28978857, 0.2894017 ],
      dtype=float32), 'z': array([-0.12275541, -0.13374843,  0.31330112,  0.01007694, -0.00916085],
      dtype=float32), 'velocity': array([0.        , 0.        , 0.        , 0.12104691, 0.72628146],
      dtype=float32)}
{'numObj': 5, 'x': array([-0.52674896, -1.8413311 ,  1.1368617 , -0.10443371, -0.04397209],
      dtype=float32), 'y': array([0.48366353, 0.68071175, 1.8117851 , 3.3402464 , 0.20243616],
      dtype=float32), 'z': array([-0.1603149 , -0.0613777 , -0.06687421,  0.        , -0.10993022],
      dtype=float32), 'velocity': array([0.        , 0.        , 0.        , 0.        , 0.24209382],
      dtype=float32)}
{'numObj': 3, 'x': array([-0.4580426, -1.7799535,  1.0699874], dtype=float32), 'y': array([0.56795436, 0.82117766, 1.8520656 ], dtype=float32), 'z': array([-0.06870639, -0.12275541, -0.0668
7421], dtype=float32), 'velocity': array([0., 0., 0.], dtype=float32)}
{'numObj': 3, 'x': array([-0.09893719, -1.7799535 ,  1.0699874 ], dtype=float32), 'y': array([0.33739802, 0.8280304 , 1.8520656 ], dtype=float32), 'z': array([ 0.01099302, -0.0613777 , -0.0
6687421], dtype=float32), 'velocity': array([0., 0., 0.], dtype=float32)}
{'numObj': 1, 'x': array([1.2706101], dtype=float32), 'y': array([1.7167276], dtype=float32), 'z': array([-0.13374843], dtype=float32), 'velocity': array([0.], dtype=float32)}
```

Figure 8.21: Proximity data being generated by IWR6843AOPEVM mm Wave radar in the console window.

Now the mmWave radar is working as proximity sensor sending data from companion computer to Ardupilot running on FCU which in my case is Beagle Bone Blue. If we connect FCU to GCS like Mission Planner , we can see the radar output as seen in next image. We need to make changes to Ardupilot running on FCU through Mission Planner to be able to see output of sensors in Mission Planner which is the GCS. Hence, I have dedicated an entire chapter 10 to understanding configuration of Ardupilot through Mission Planner later in the thesis . For now, the next image shows the mmWave radar output in Mission Planner.

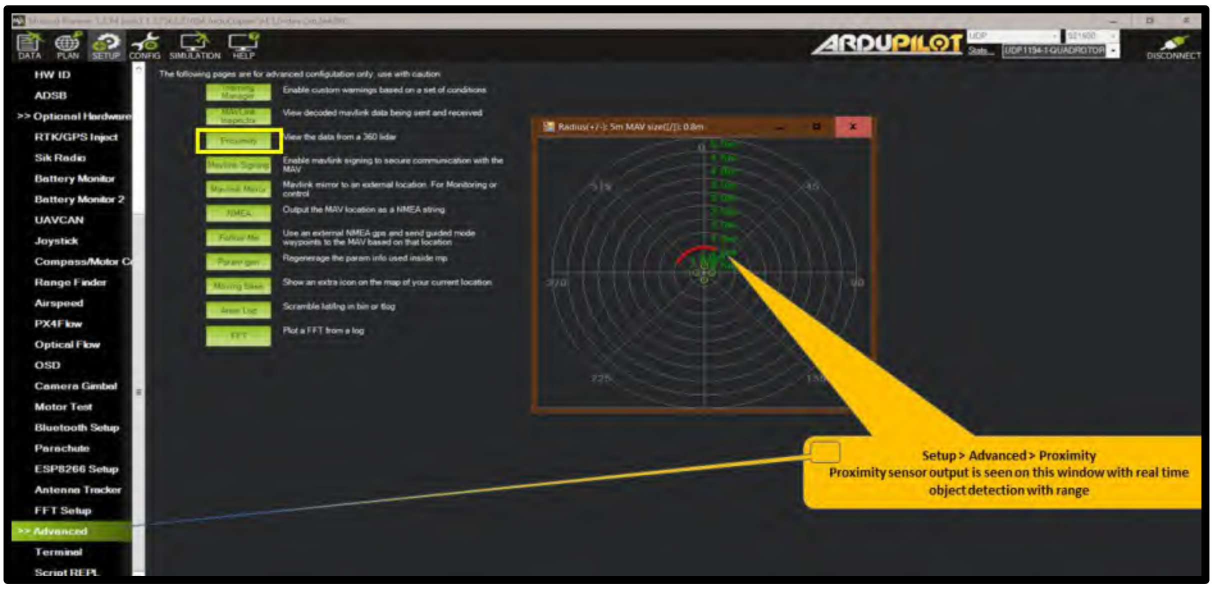
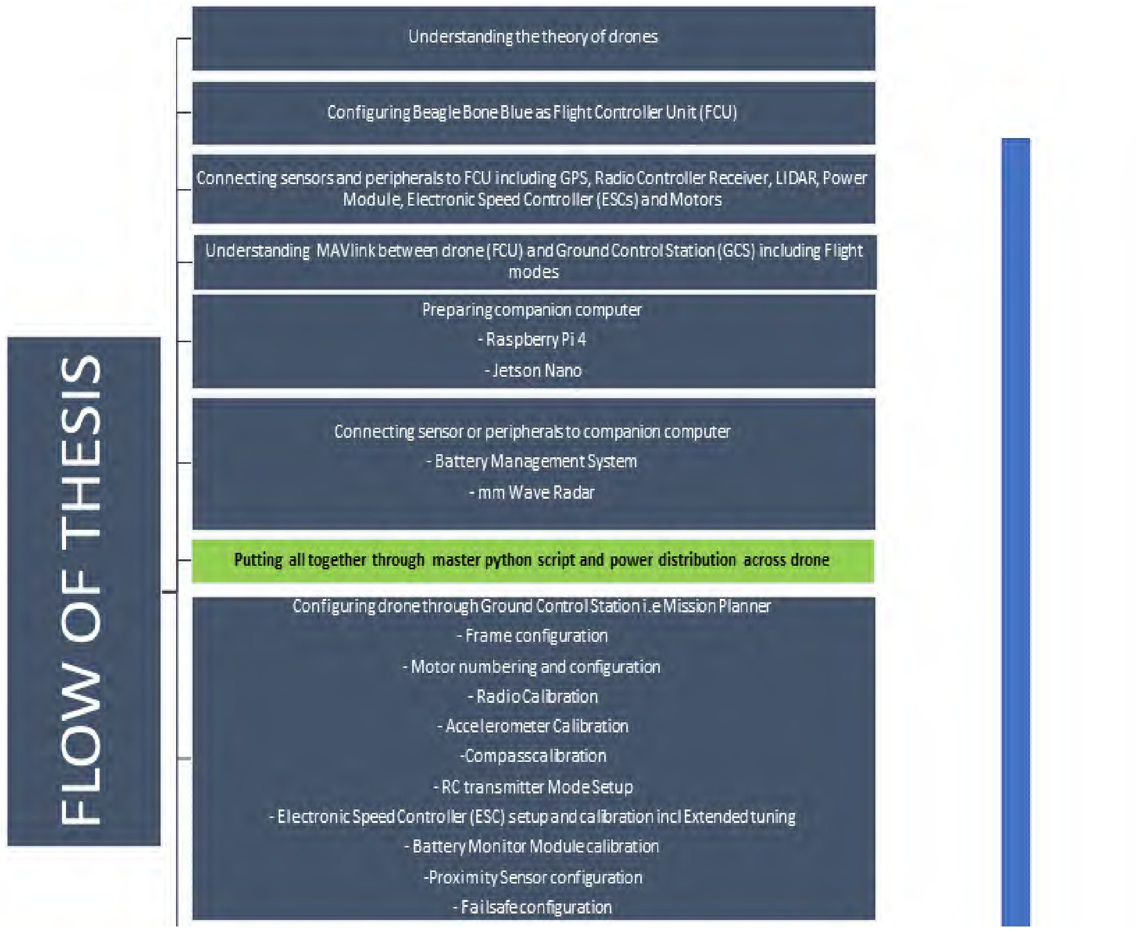

DATA
PLAN
SETUP
CONFIG
SIMULATION
HELP
ARDUPILOT
DISCONNECT
HW ID
ADSB
>> Optional Hardware
RTK/GPS Inject
Sik Radio
Battery Monitor
Battery Monitor 2
UAVCAN
Joystick
Compass/Motor C
Range Finder
Airspeed
PX4Flow
Optical Flow
OSD
Camera Gimbal
Motor Test
Bluetooth Setup
Parachute
ESP8266 Setup
Antenna Tracker
FFT Setup
>> Advanced
Terminal
Script REPL
The following pages are for advanced configuration only, use with caution
Enable custom warnings based on a set of conditions
View decoded mavlink data being sent and received
Proximity
View the data from a 360 lidar
Enable mavlink signing to secure communication with the MAV
Mavlink mirror to an external location. For Monitoring or control
NMEA
Output the MAV location as a NMEA string
Use an external NMEA gps and send guided mode waypoints to the MAV based on that location
Regenerage the param info used inside mp
Show an extra icon on the map of your current location
Scramble lat/lng in bin or tlog
Plot a FFT from a log
Radius(+/-): 5m MAV size([/]): 0.8m
0
45
90
135
225
270
315
Setup > Advanced > Proximity
Proximity sensor output is seen on this window with real time object detection with range

Figure 8.22: Proximity data being generated by IWR6843AOPEVM mm Wave radar is seen in Mission Planner

Also we need to run this python script automatically on start up. Hence, I have written a master_python.py script in next chapter (Chapter 8) to run all above individual python scripts with one master python script.

CHAPTER 9

RUNNING ALL THE ABOVE PYTHON SCRIPTS AS SINGLE MASTER PYTHON SCRIPT ON STARTUP ON COMPANION COMPUTER


FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Figure 9.1: Flow of thesis until now

We have seen that VPN services incase of Beagle Bone Blue or companion computer are auto running on startup using systemd file. But the MAVPROXY for MAVlink, BMS data capture and mm Wave radar python scripts need to be run on start up. Hence, we make a master python script which will run all the three python scripts on startup using .bashrc file on startup. This master python script will auto run incase of Raspberry Pi 4 but incase of Jetson Nano, the user will have to login to ssh before all are master python script is triggered on due to higher security settings in Jetson Nano. This allows better user control and security of drone incase of Jetson Nano.

Master python script is used to run MAVPROXY, BMS Data capture and mm Wave Radar python scripts on startup though .bashrc file. Once can always omit the part not required by commenting the relevant section of this python script.

For running master_script.py on boot, amend bash.rc

For Raspberry Pi 4

sudo nano /home/pi/.bashrc

Add the lines

echo Running at boot

python3 master_script.py

For Jetson Nano

sudo nano /home/yash/.bashrc

Add the lines

echo Running at boot

sudo python3 master_script.py

Note: Replace yash with your username on Nano

9.1 Master_script.py for Raspberry Pi 4

```
#!/usr/bin/env python3

##  Running these two scripts simultaneously:       ##
##   - mmwave_to_mavlink.py                          ##
##   - bms_data_capture.py                           ##
##   - mavproxy.py                                   ##
##                                                   ##

import os
import os.path
import threading
import serial

connection_in_port = "/dev/ttyS0"       # Serial port connected to FCU
connection_in_baud = "921600"           # As per Mission Planner settings
```

```
connection_out_p01 = “127.0.0.1:14550”   # Radar
connection_out_p02 = “100.96.1.34:1194”  # Mission Planner


def mavproxy_create_connection():
    os.system(“python3 $HOME/.local/bin/mavproxy.py” + \
           “ –master=”   + connection_in_port + \
           “ –baudrate=” + connection_in_baud + \
           “ –out udp:”  + connection_out_p01 + \
           “ –out udp:”  + connection_out_p02)


def run_radar():
   os.system(“python3 ~/mm_Wave_Radar_IWR6843AOPEVM/mmwave_to_mavlink.py”)

def run_battcapture():
   os.system(“python3 ~/BattMgmtSys_Dashboard/bms_data_capture.py”)


thread1 = threading.Thread(target=mavproxy_create_connection)
thread1.start()

thread2 = threading.Thread(target=run_radar)
thread2.start()

thread3 = threading.Thread(target=run_battcapture)
thread3.start()
```

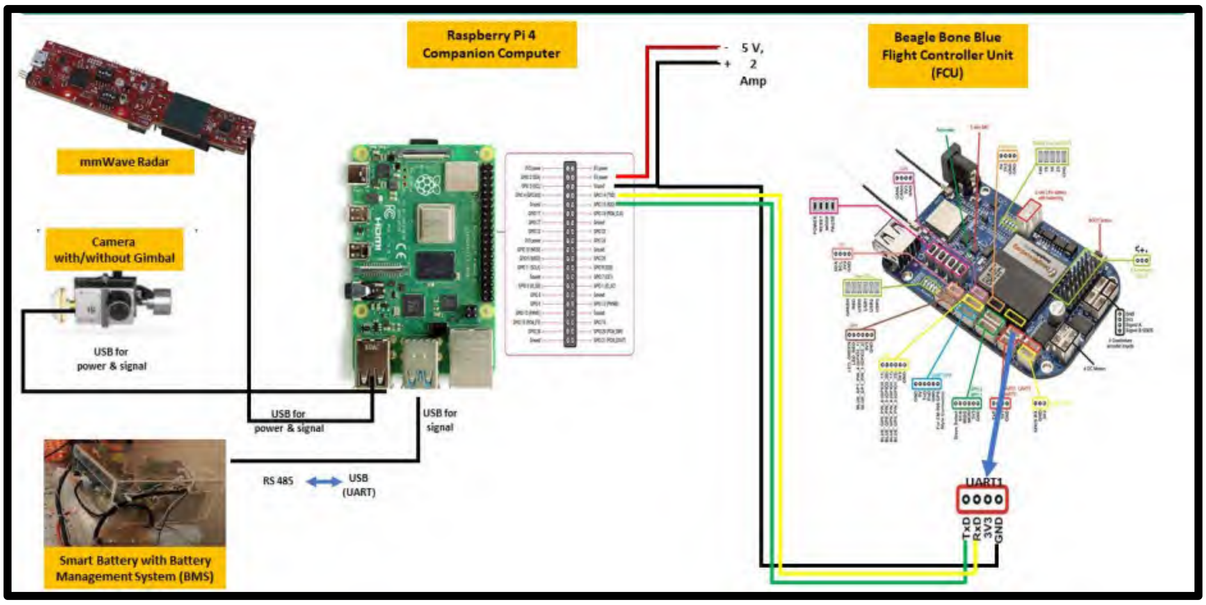
9.2 Master_script.py for Jetson Nano

```
#!/usr/bin/env python3

##########################################################
##  Running these two scripts simultaneously:       ##
##   - mmwave_to_mavlink.py                         ##
##   - bms_data_capture.py                          ##
##   - mavproxy.py                                  ##
##                                                  ##
##########################################################


import os
import os.path
import threading
import serial

connection_in_port = “/dev/ttyTHS1”        # Serial port connected to FCU
connection_in_baud = “921600”              # As per Mission Planner settings
connection_out_p01 = “127.0.0.1:14550”     # Radar
connection_out_p02 = “100.96.1.34:1194”    # Mission Planner


def mavproxy_create_connection():
    os.system(“python3 /home/yash/.local/bin/mavproxy.py” + \
        " –master="   + connection_in_port + \
        " –baudrate=" + connection_in_baud + \
        " –out udp:"  + connection_out_p01 + \
        " –out udp:"  + connection_out_p02)


def run_radar():
os.system("python3 /home/yash/mm_Wave_Radar_IWR6843AOPEVM/mmwave_to_mavlink.py")

def run_battcapture():
 os.system("python3 /home/yash/BattMgmtSys_Dashboard/bms_data_capture.py")


thread1 = threading.Thread(target=mavproxy_create_connection)
thread1.start()

thread2 = threading.Thread(target=run_radar)
thread2.start()

thread3 = threading.Thread(target=run_battcapture)

thread3.start()
```

9.3 Overall companion computer connectivity diagram

Overall Connectivity diagrams So now we see the overall connectivity diagram of companion computers with connected peripherals/sensors namely proximity sensor IWR6843AOPEVM mm Wave radar, onboard camera and smart battery with BMS.

For Raspberry Pi 4 as companion computer



mmWave Radar
Raspberry Pi 4 Companion Computer
- 5 V, + 2 Amp
Beagle Bone Blue Flight Controller Unit (FCU)
Camera with/without Gimbal
USB for power & signal
USB for power & signal
USB for signal
RS 485
USB (UART)
UART1
TxD RxD 3V3 GND
Smart Battery with Battery Management System (BMS)

Figure 9.2: The connectivity diagram for physical integration of Companion computer

(Raspberry Pi 4) with proximity sensor IWR6843AOPEVM mm Wave radar, onboard camera and smart battery with BMS

For Jetson Nano as companion computer

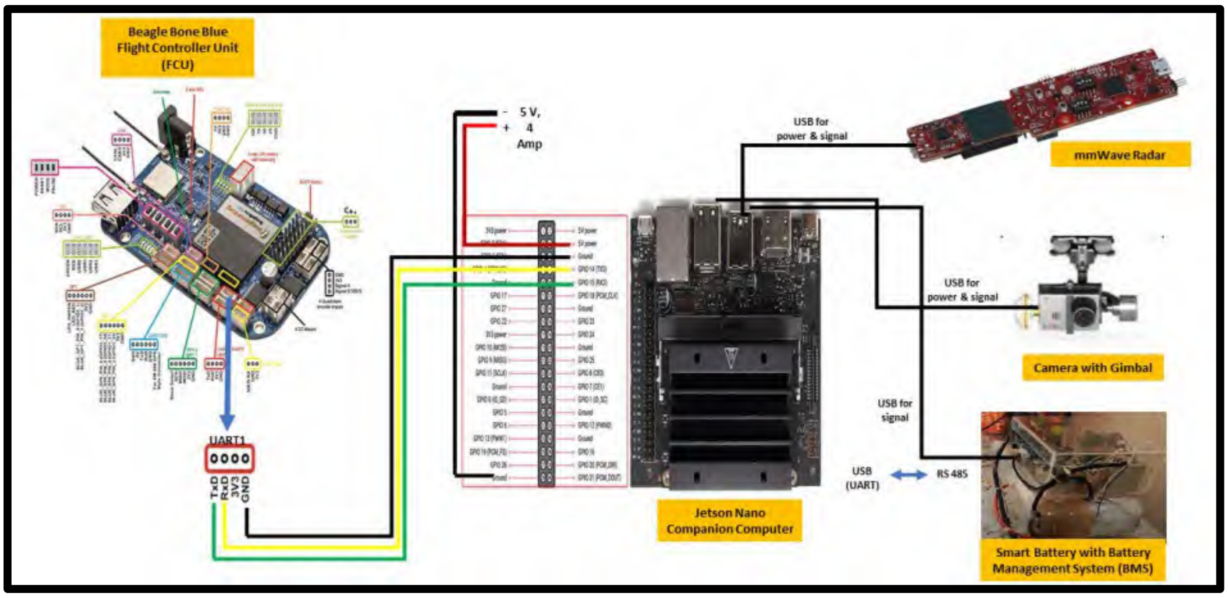


Figure 9.3: The connectivity diagram for physical integration of Companion computer (Jetson Nano) with proximity sensor IWR6843AOPEVM mm Wave radar, onboard camera and smart battery with BMS

The overall power distribution connectivity block diagram of the drones have been depicted below. It is important to note that a dedicated effort was done to keep the power and signal connections separate with no sensor being powered through Beagle Bone Blue. Though Beagle Bone Blue is capable of providing 5V power to sensors like GPS and 5V power output option, it was intentionally avoided to prevent any damage to Beagle Bone Blue which being the Flight Controller Unit (FCU) is the most critical component of a drone. Hence, the power to each sensor or component has been distributed separately through power distribution board or DC-DC convertor. The setup was tested on all three drones.

Small X-Type quadcopter The small X-type drone is powered through 3s (11.1V nominal and 12.6V peak) LiPo battery pack. Here, the power was distributed through a power distribution panel with frame itself for Electronic Speed Controls which in turn are powering the motors. The sensors have been powered by 5V DC-DC convertor. There are only two sensors/peripherals on board namely GPS with compass and radio controller 9RC) receiver. The flight controller is Beagle Bone Blue and it acts like both the FCU and companion computer. The 4G communication module is connected to Beagle Bone Blue. The broad connectivity diagram for power distribution is as below

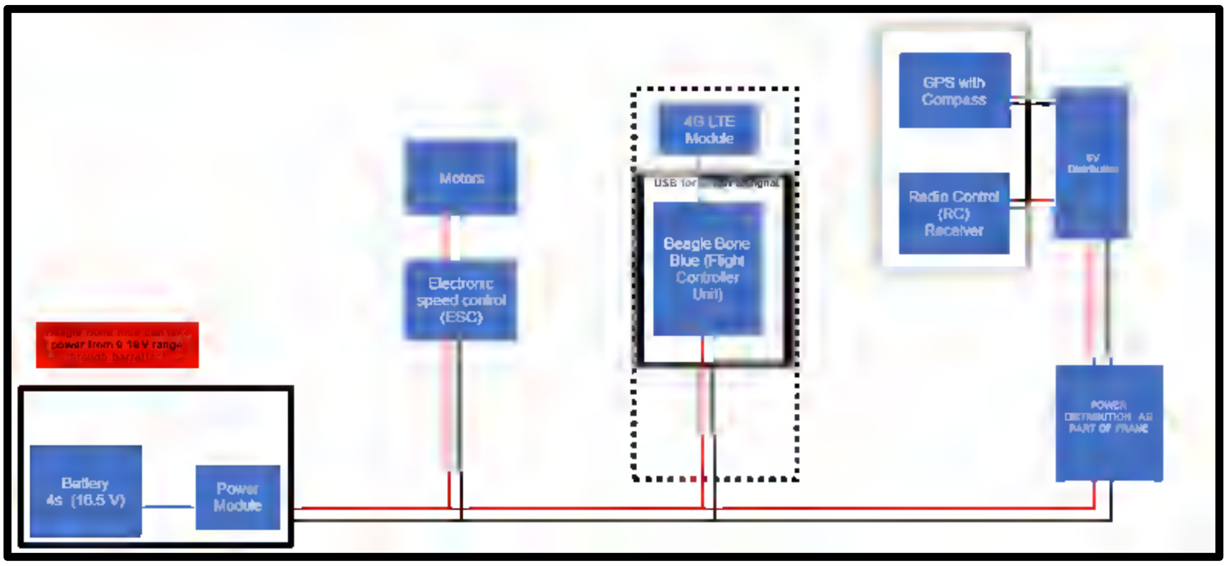
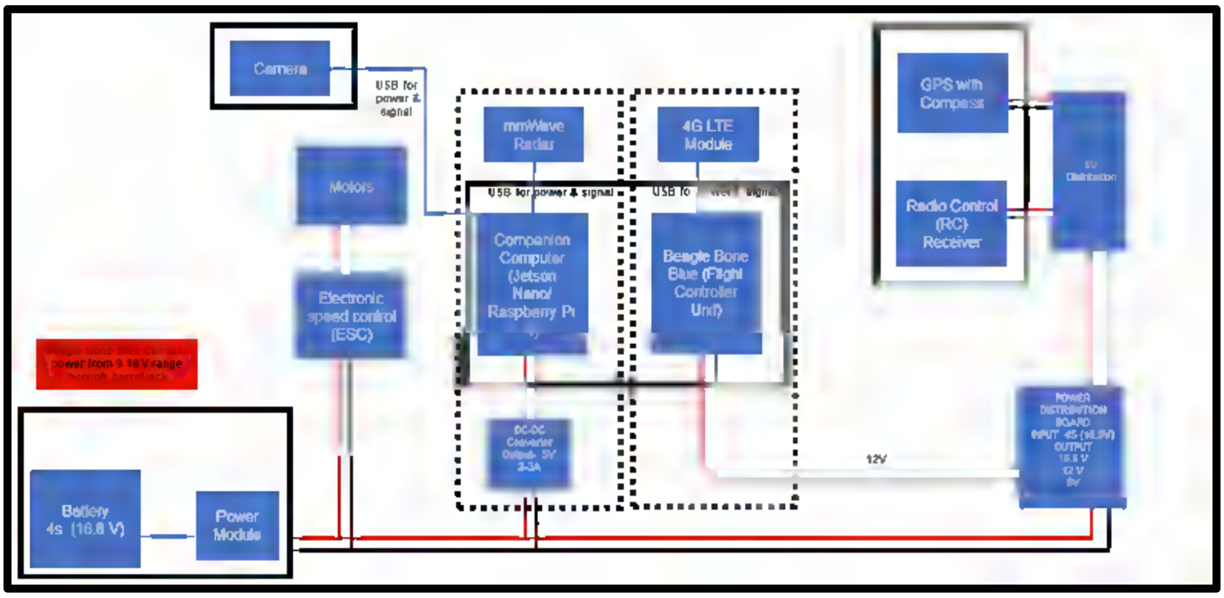


Figure 9.4: Overall power distribution block diagram of X-Type Quadcopter

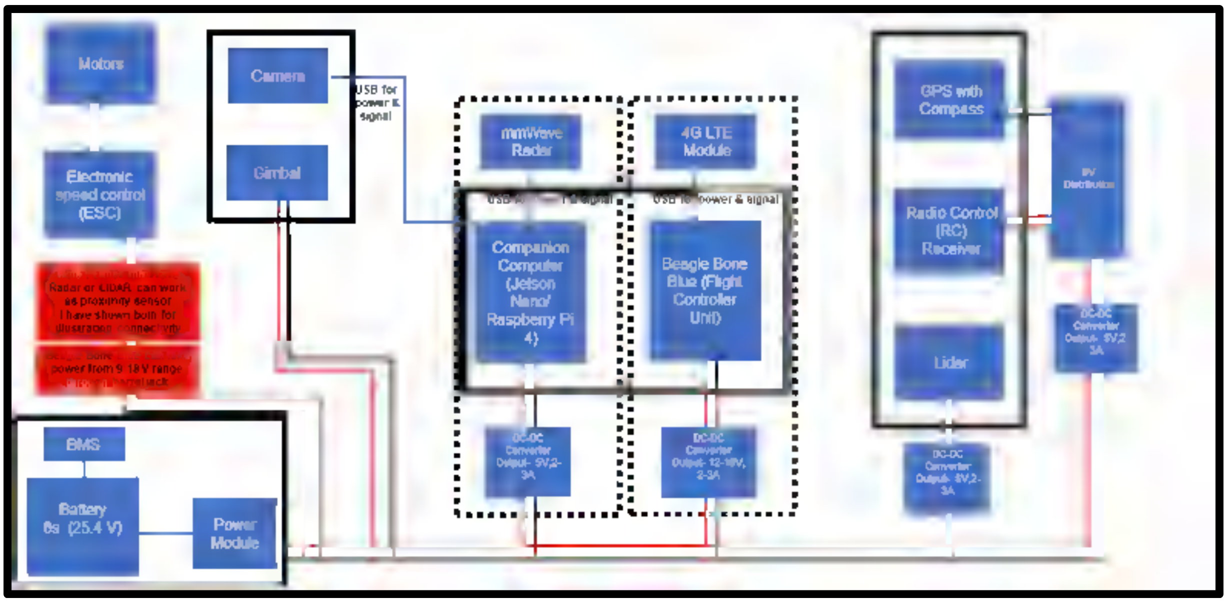
Small V Type quadcopter The small V type drone is powered through 4s (14.8V nominal and 16.8V peak) LiPo battery pack. Here the power was distributed through a power distribution module which takes input of 4s (16.5V) and gives 16.5V for distribution to ESCs and DC-DC converted 12V and 5V output for powering sensors. Only Companion computer was powered through another dedicated DC-DC convertor giving 5V and 2-3 Amps as companion computer was powering the mmWave radar through USB and camera through USB. The board connectivity diagram is as below:



Camera
USB for power & signal
GPS with Compass
mmWave Radar
4G LTE Module
Motors
USB for power & signal
USB for power & signal
5V Distribution
Radio Control (RC) Receiver
Companion Computer (Jetson Nano/ Raspberry Pi)
Beagle Bone Blue (Flight Controller Unit)
Electronic speed control (ESC)
power from 9 18 V range
through barrel jack
DC-DC Converter Output- 5V 2-3A
12V
POWER DISTRIBUTION BOARD INPUT 4S (16.5V) OUTPUT 16.5 V 12 V 5V
Battery 4s (16.8 V)
Power Module

Figure 9.5: Overall power distribution block diagram of V-Type Quadcopter

Medium weight drone (Hexacopter) The medium weight Hex copter type drone is powered through 6s (22.2V nominal and 25.2V peak) LiPo battery pack. Here the power was distributed through dedicated DC-DC convertor giving 12V output for Beagle Bone Blue while 5V for all other sensors including companion. The board connectivity diagram is as below


Motors
Electronic speed control (ESC)
Radar or LIDAR can work as proximity sensor I have shown both for illustration connectivity
power from 9-18 V range
BMS
Battery 6s (25.4 V)
Power Module
Camera
Gimbal
USB for power & signal
mmWave Radar
Companion Computer (Jetson Nano/ Raspberry Pi 4)
DC-DC Converter Output- 5V,2-3A
4G LTE Module
USB for power & signal
Beagle Bone Blue (Flight Controller Unit)
DC-DC Converter Output- 12-18V, 2-3A
GPS with Compass
Radio Control (RC) Receiver
Lidar
DC-DC Converter Output- 5V,2-3A
5V Distribution
DC-DC Converter Output- 5V,2 3A
Figure 9.6: Overall power distribution block diagram of Hexacopter

CHAPTER 10

UNDERSTANDING MISSION PLANNER FOR CONFIGURING DRONE FOR FLIGHT

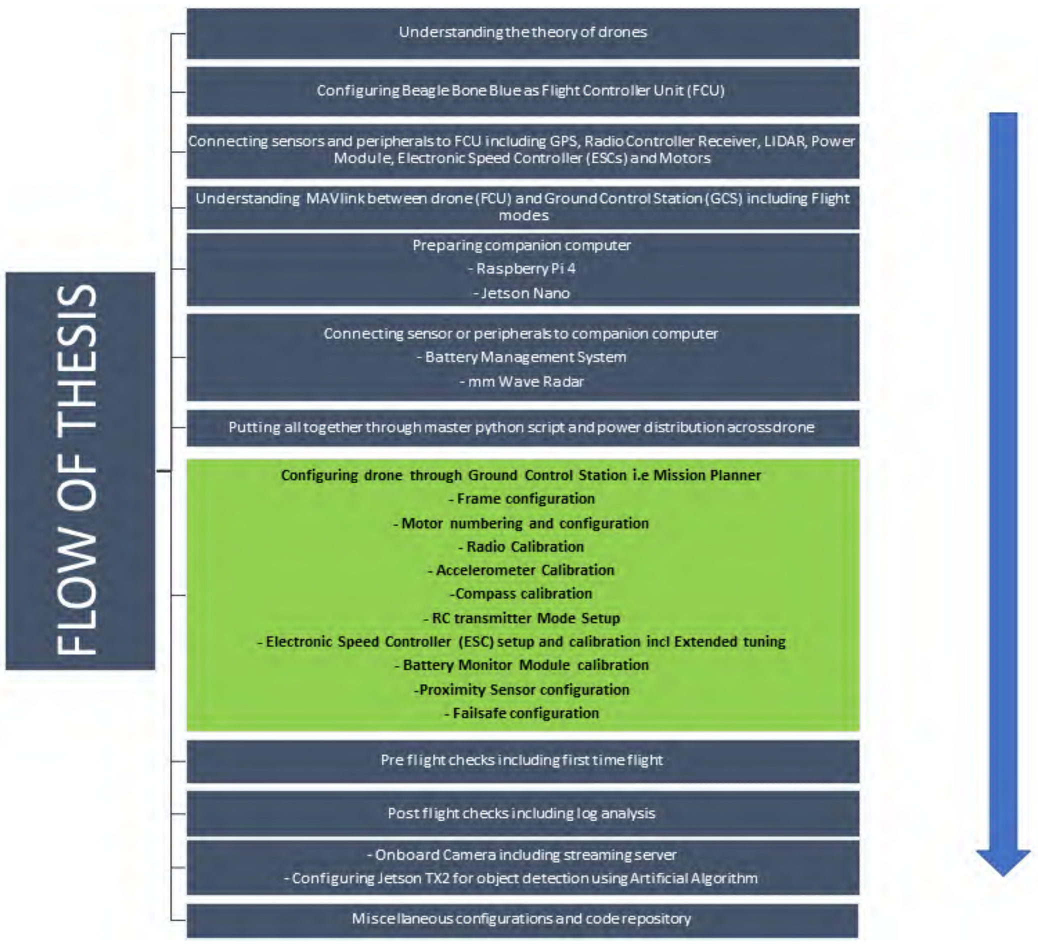
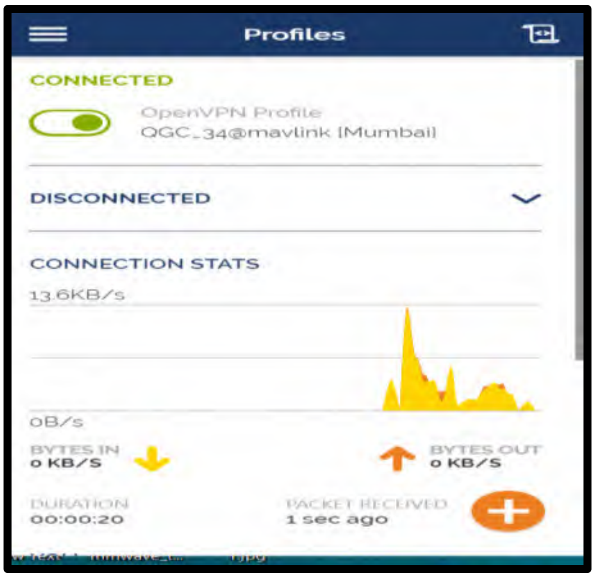

FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer - Raspberry Pi 4 - Jetson Nano
Connecting sensor or peripherals to companion computer - Battery Management System - mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner - Frame configuration - Motor numbering and configuration - Radio Calibration - Accelerometer Calibration -Compass calibration - RC transmitter Mode Setup - Electronic Speed Controller (ESC) setup and calibration incl Extended tuning - Battery Monitor Module calibration -Proximity Sensor configuration - Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server - Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

Figure 10.1: Flow of thesis until now

Now we come to the most important part of the configuration. We assume that all hardware connections have been made as per information provided in previous chapters. Each individual part has been tested and validated for hardware and software configuration. Now we install the Mission Planner Ground Control Application on our Laptop or PC. We connect internet to the laptop and follow the steps below in the order to avoid running errors.

Step 1 Install Mission Planner GCS Application on laptop or PC.

Step 2. Connect laptop to internet and download OpenVPN windows client. Use the .ovpn file that we created in chapter IV. We had created .ovpn files for each component from FCU, Companion computer to GCS and placed them in a folder VPN_Folder on the laptop. We can drag and drop on OVPN client our gcs.ovpn file and connect to VPN like below:-



Profiles

CONNECTED

OpenVPN Profile
QGC_34@mavlink [Mumbai]

DISCONNECTED

CONNECTION STATS

13.6KB/s

0B/s

BYTES IN
0 KB/S

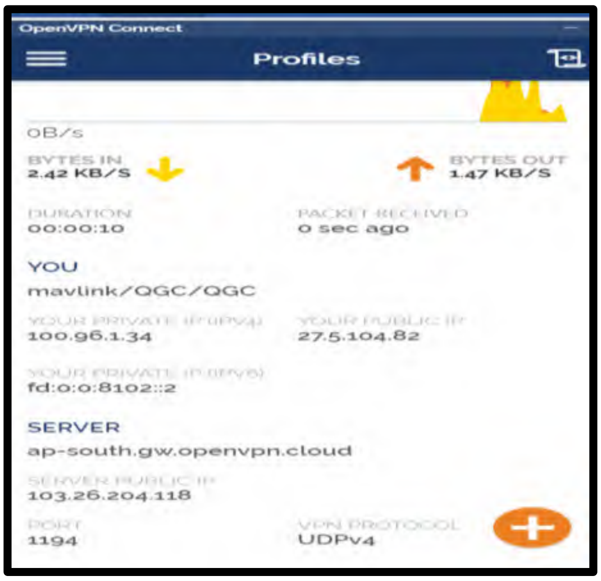
BYTES OUT
0 KB/S

DURATION
00:00:20

PACKET RECEIVED
1 sec ago

Figure 10.2: Windows Client of OpenVPN installed and connected on laptop/PC running Mission Planner Ground Control Station (GCS) application

Step 3 Scroll down and note VPN IP Address. This IP Address is the IP Address of our ground control station (GCS). In my case its 100.96.1.34 (Kindly refer the VPN diagram in Chapter IV). All configuration of MAVlink in chapters above has assumed this IP Address. ***Please note that the IP Addresses will depend on your VPN service provider and your account created.***



OpenVPN Connect

Profiles

0B/s

BYTES IN
2.42 KB/S

BYTES OUT
1.47 KB/S

DURATION
00:00:10

PACKET RECEIVED
0 sec ago

YOU
mavlink/QGC/QGC

YOUR PRIVATE IP (IPv4)
100.96.1.34

YOUR PUBLIC IP
27.5.104.82

YOUR PRIVATE IP (IPv6)
fd:0:0:8102::2

SERVER
ap-south.gw.openvpn.cloud

SERVER PUBLIC IP
103.26.204.118

PORT
1194

VPN PROTOCOL
UDPv4

Figure 10.3: VPN connection details of laptop/PC running Mission Planner Ground Control

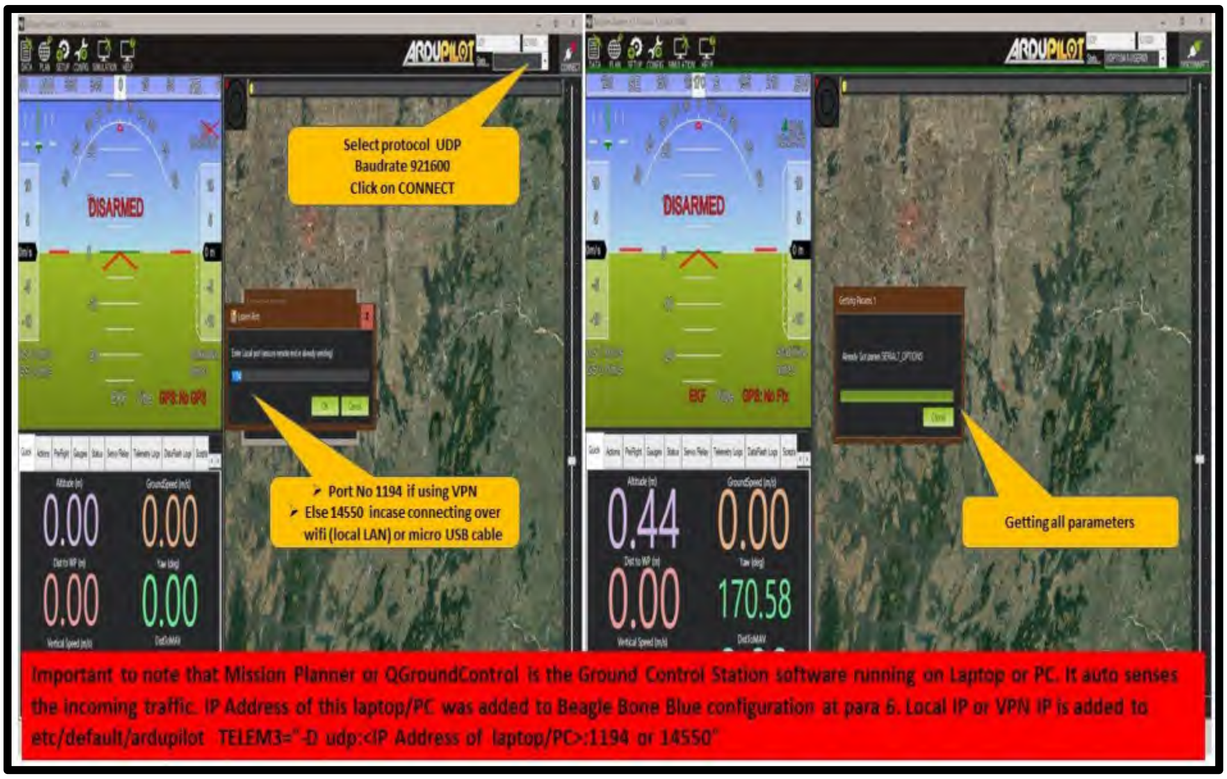
Station (GCS) application

Now, power up the drone, ensure all components works (no visible hardware error) and 4G LTE Modem is connected to service provider. Make sure there are no propellers on the drone. One can take remote ssh of Beagle Bone Blue and Companion Computer to check whether all working ok. If all OK lets connect Mission Planner GCS to the Drone. Here, it is important to note that it does not matter whether you are connecting MAVlink to Beagle Bone Blue directly incase of small drones or Companion computer incase of medium or large drones. The procedure remains the same. Because even when we establish MAVlink via companion computer using MAVPROXY, the link is indirectly between Ardupilot running on Flight Controller Unit which is Beagle Bone Blue and GCS i.e Mission Planner.

10.1 Configuring the Drone (FCU) as per frame design, calibration of sensors and validating outputs

Connecting MAVlink between drone (FCU) to Ground Control Station (GCS) i.e Mission Planner MAVPROXY via companion computer or directly to Ardupilot (FCU). When we run Mission Planner with Admin rights it ensures access through windows firewall or Anti-virus Firewall. As we have defined the IP Address of this laptop while configuring FCU and companion computer, the MAVlink traffic will be directed to it over local LAN or VPN depending upon the scheme adopted. Here, we simply need to select UDP as protocol, 921600 as baudrate and connect. UDP is the internet protocol to connect to FCU directly or indirectly via companion computer at 921600 baudrate. (Refer Chapter 4). Hence, from now we will refer to link with Ardupilot running on FCU directly or via companion computer as MAVLink assuming we refer both unless there is a need to specify the exact arrangement.

Mission Planner (GCS) is used to configure Ardupilot autopilot application running on FCU. We configure the FCU (Ardupilot application) as per our drone design, calibrate the sensors onboard, validate the outputs and be ready to fly by choosing the flight options. All this will be covered in the thesis from hereon.



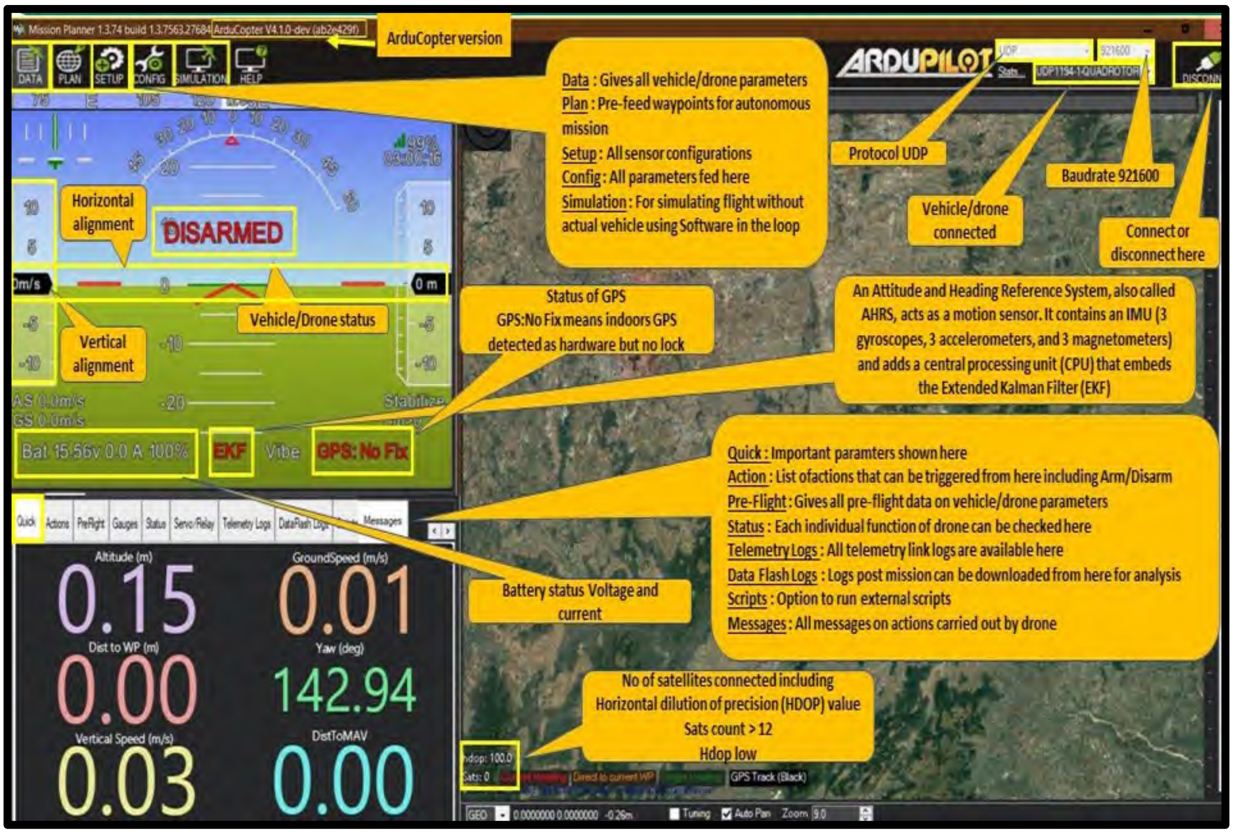
Select protocol UDP
Baudrate 921600
Click on CONNECT

➤ Port No 1194 if using VPN
➤ Else 14550 incase connecting over wifi (local LAN) or micro USB cable

Getting all parameters

Important to note that Mission Planner or QGroundControl is the Ground Control Station software running on Laptop or PC. It auto senses the incoming traffic. IP Address of this laptop/PC was added to Beagle Bone Blue configuration at para 6. Local IP or VPN IP is added to etc/default/ardupilot TELEM3="-D udp:<IP Address of laptop/PC>:1194 or 14550"

Figure 10.4: Connecting Mission Planner (GCS) application to MAVlink



ArduCopter version

Data : Gives all vehicle/drone parameters
Plan : Pre-feed waypoints for autonomous mission
Setup : All sensor configurations
Config : All parameters fed here
Simulation : For simulating flight without actual vehicle using Software in the loop

Protocol UDP

Baudrate 921600

Vehicle/drone connected

Connect or disconnect here

Horizontal alignment

Vertical alignment

Vehicle/Drone status

Status of GPS
GPS:No Fix means indoors GPS detected as hardware but no lock

An Attitude and Heading Reference System, also called AHRS, acts as a motion sensor. It contains an IMU (3 gyroscopes, 3 accelerometers, and 3 magnetometers) and adds a central processing unit (CPU) that embeds the Extended Kalman Filter (EKF)

Quick : Important paramters shown here
Action : List ofactions that can be triggered from here including Arm/Disarm
Pre-Flight : Gives all pre-flight data on vehicle/drone parameters
Status : Each individual function of drone can be checked here
Telemetry Logs : All telemetry link logs are available here
Data Flash Logs : Logs post mission can be downloaded from here for analysis
Scripts : Option to run external scripts
Messages : All messages on actions carried out by drone

Battery status Voltage and current

No of satellites connected including Horizontal dilution of precision (HDOP) value
Sats count > 12
Hdop low

Figure 10.5: Understanding Mission Planner Ground Control Station (GCS) dashboard

Frame configuration After successfully establishing connection for the first time, between Drone FCU (Beagle Bone Blue) and GCS (Mission Planner) (irrespective or direct or indirect MAVlink), we need to select the frame type as per design of drone. This is not a repetitive step but done for the first time or after any major reset.

Mission Planner > Setup > Mandatory Hardware > Frame Selection

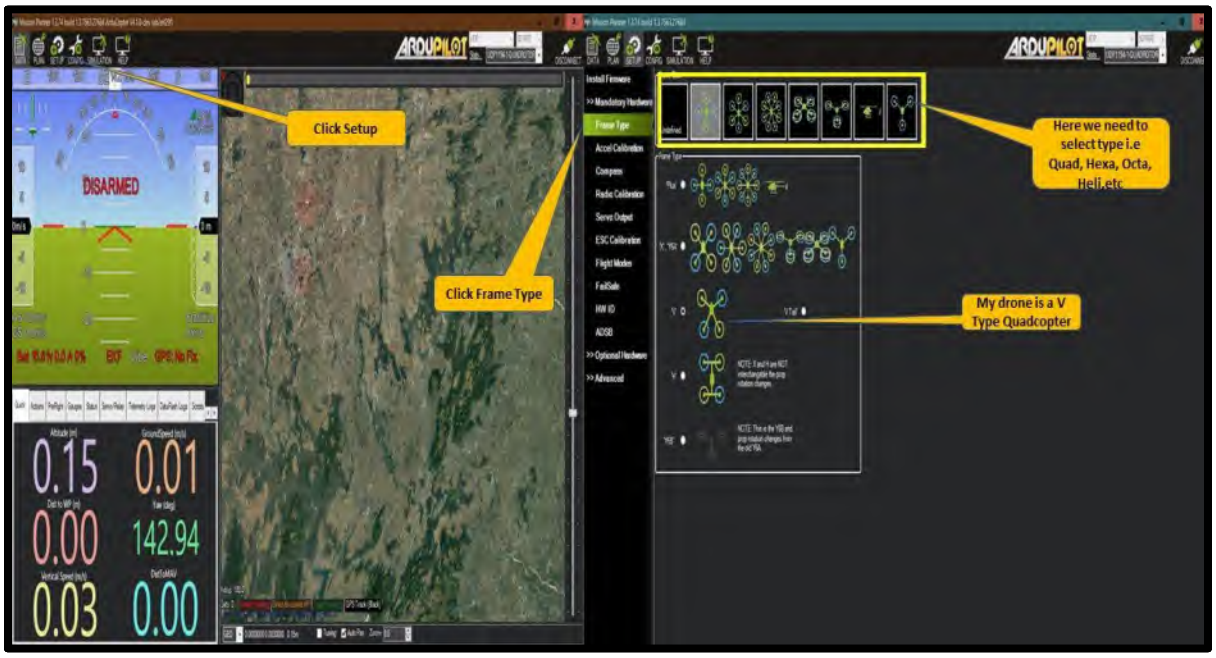

Click Setup
Click Frame Type
Here we need to select type i.e Quad, Hexa, Octa, Heli,etc
My drone is a V Type Quadcopter
DISARMED
0.15
0.01
0.00
142.94
0.03
0.00

Figure 10.6: Selecting frame type in Mission Planner (GCS application) to send configuration instructions to FCU (Beagle Bone Blue) through MAVlink

After doing the frame selection, we need to do motor numbering and select the correct orientation. Incase of quadcopter, the number of motors are 4 where motor A/motor 1 and motor C/motor 2 are clockwise while motor B/motor 4 and motor D/motor 3 are anticlockwise. The pairwise ordering has to be same, however we can reverse the clockwise to anticlockwise as per our choice. The main point is A-C or 1-2 will rotate in same direction while B-D or 3-4 will rotate in other direction opposite to the other combination.

Incase of Hexacopter, the number of motors are 6 where motor A/motor 2, motor C/motor 4 and motor E/motor 7 are clockwise while motor B/motor 1, motor D/motor 3 and motor F/motor 8 are anticlockwise. The pairwise ordering has to be same, however we can reverse the clockwise to anticlockwise as per our choice. The main point is A-C-E or 2-4-7 will rotate in same direction while B-D-F or 1-3-8 will rotate in other direction opposite to the other combination.

The frame can be of X type or plus type depends upon the direction of forward move of the drone. It has been observed that X type frames are much more stable than the Plus type frame as the front move is handled by two motors instead of a single motor incase of plus. The detailed analysis on the frames, its types and other considerations have been discussed in the thesis of my project partner. The pictorial view of the frame motor numbering is given below to allow physical connection and propeller type to be put as per schematic given below:-

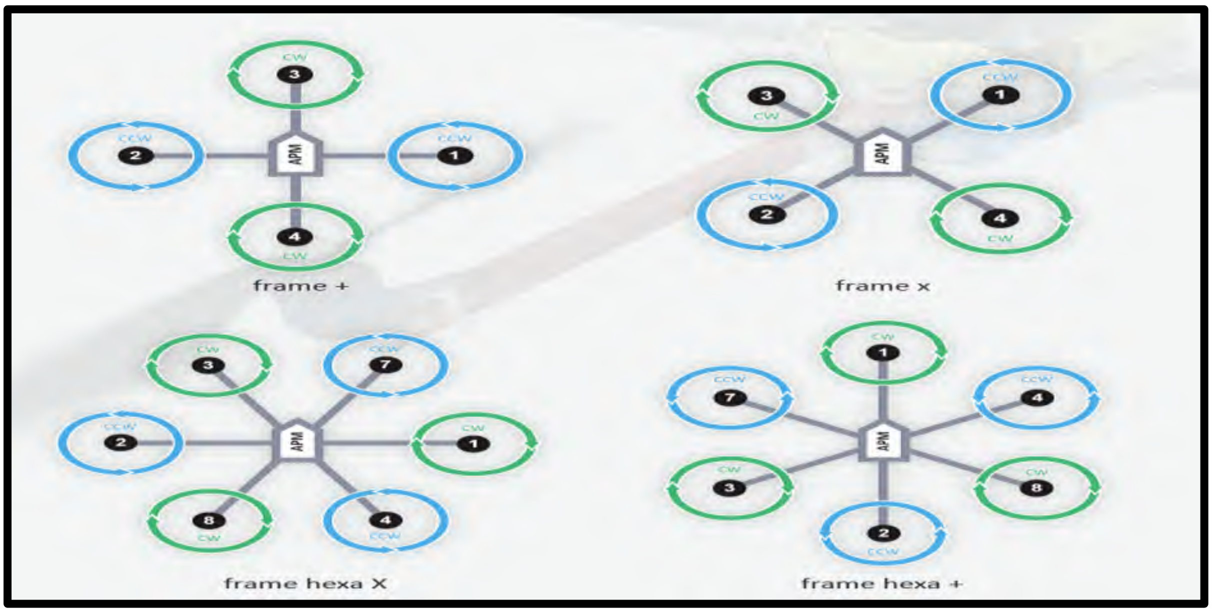
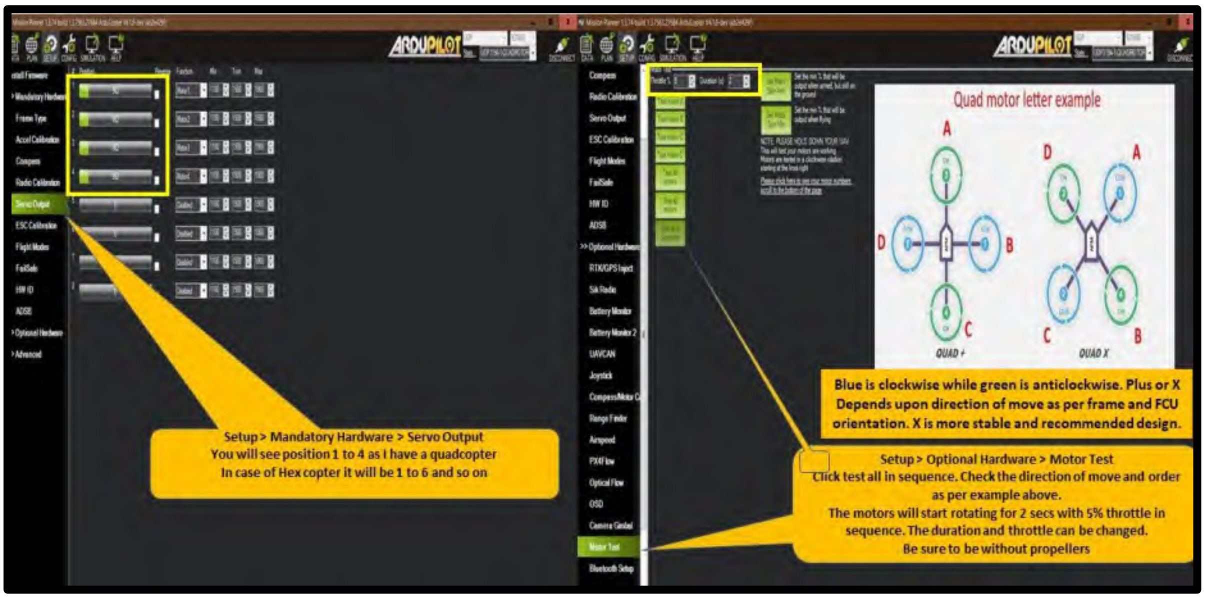


Figure 10.7: The ordering of motors with direction of rotation



Setup > Mandatory Hardware > Servo Output
You will see position 1 to 4 as I have a quadcopter
In case of Hex copter it will be 1 to 6 and so on

Blue is clockwise while green is anticlockwise. Plus or X Depends upon direction of move as per frame and FCU orientation. X is more stable and recommended design.

Setup > Optional Hardware > Motor Test
Click test all in sequence. Check the direction of move and order as per example above.
The motors will start rotating for 2 secs with 5% throttle in sequence. The duration and throttle can be changed.
Be sure to be without propellers

Figure 10.8: Checking motor/ESC detection and servo output and rotation direction including sequence

Mission Planner > Setup > Mandatory Hardware > Servo Output

Radio Calibration We need to calibrate the Radio Controller (RC) transmitter here to check for throttle, pitch, yaw, roll and flight modes are selected. We need to validate that the response to full range of controls for throttle, pitch, yaw and roll while toggle switches are programmed for flight mode selection and any other control we wish to have. This is important for all manual missions. Even for autonomous missions we need to test the drone locally before takeoff. Hence, it’s an important step. I will be using Taranis as Radio Controller (RC) transmitter and the X8R as the Radio controller (RC) receiver which we connected to FCU in chapter 3.

Mission Planner > Setup > Mandatory Hardware > Radio Calibration

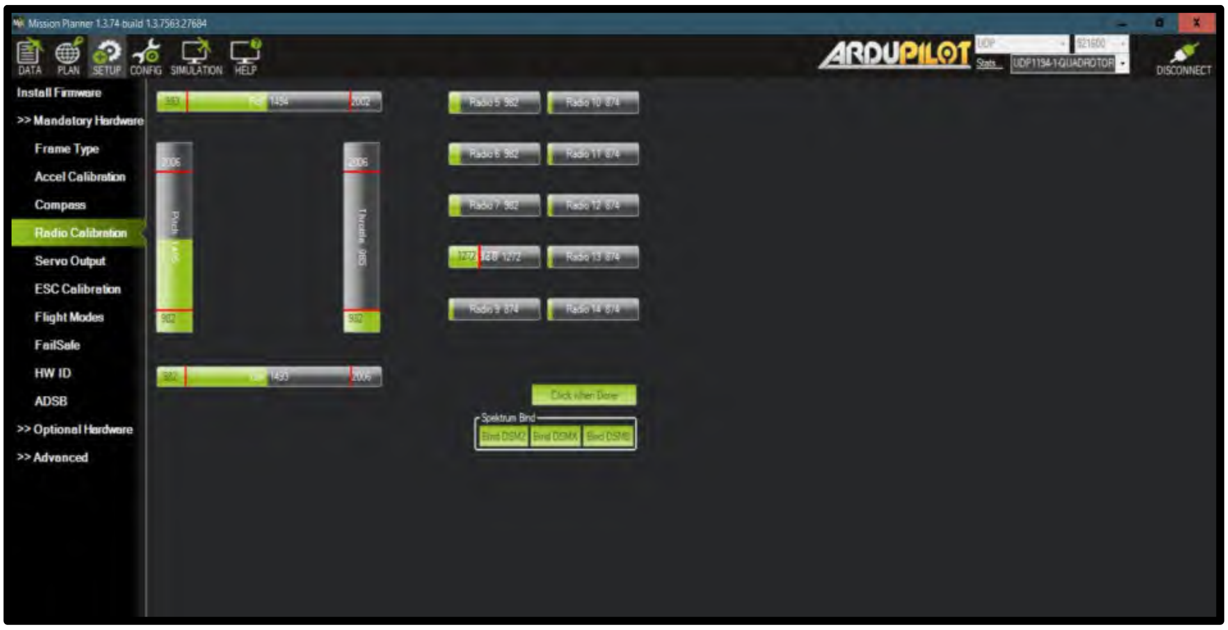


Figure 10.9: Calibrating Radio Controller (RC) Transmitter through Mission Planner (GCS application) through MAVlink

Before doing Accelerometer and compass calibration, we need to see the orientation of FCU with respect to the drone. In my case the FCU is in the line of aircrafts forward movement, hence the default AHRS orientation works for me. But incase the design of aircraft is such that FCU is facing in any other direction other than the direction of forward move of the drone then AHRS orientation needs to be done at

Mission Planner > Config > Full Parameter List > AHRS_ORIENTATION

Incase of any other orientation, refer to the link for its value **https://ardupilot.org/copter/docs/parameters.html#ahrs-orientation e.g** AHRS_ORIENTATION value is 1 for yaw45 degrees and so on. Default value is 0.

The Beagle Bone Blue has onboard 9-Axis Inertial Measurement Unit (IMU) which combines a 3-axis gyroscope, 3-axis accelerometer and 3-axis compass. This IMU works by detecting linear acceleration using accelerometers, rotational rate using gyroscopes and heading reference using magnetometer. It contains one accelerometer, gyro, and magnetometer per axis. This is for each of the three principal axes i.e pitch, roll and yaw. The raw IMU measurements is utilized to calculate altitude, angular rates, linear velocity and position relative to a global reference frame. Hence, we need to calibrate accelerometer and compass to provide the same. The Ardupilot autopilot application utilizes Extended Kalman Filter (EKF) algorithm to estimate vehicle position, velocity and angular orientation based on rate gyroscopes, accelerometer, compass, GPS, airspeed and barometric pressure measurements. The EKF is superior to the conventional complementary filter algorithms such as Inertial Navigation. It fuses all available measurements and is better able to reject measurements with considerable errors. This reduces the vulnerability of the drone to faults of a single sensor. This EKF algorithm also provides the option of integration of advanced optional sensors such as lasers and optical flow for more accurate navigation. [3]

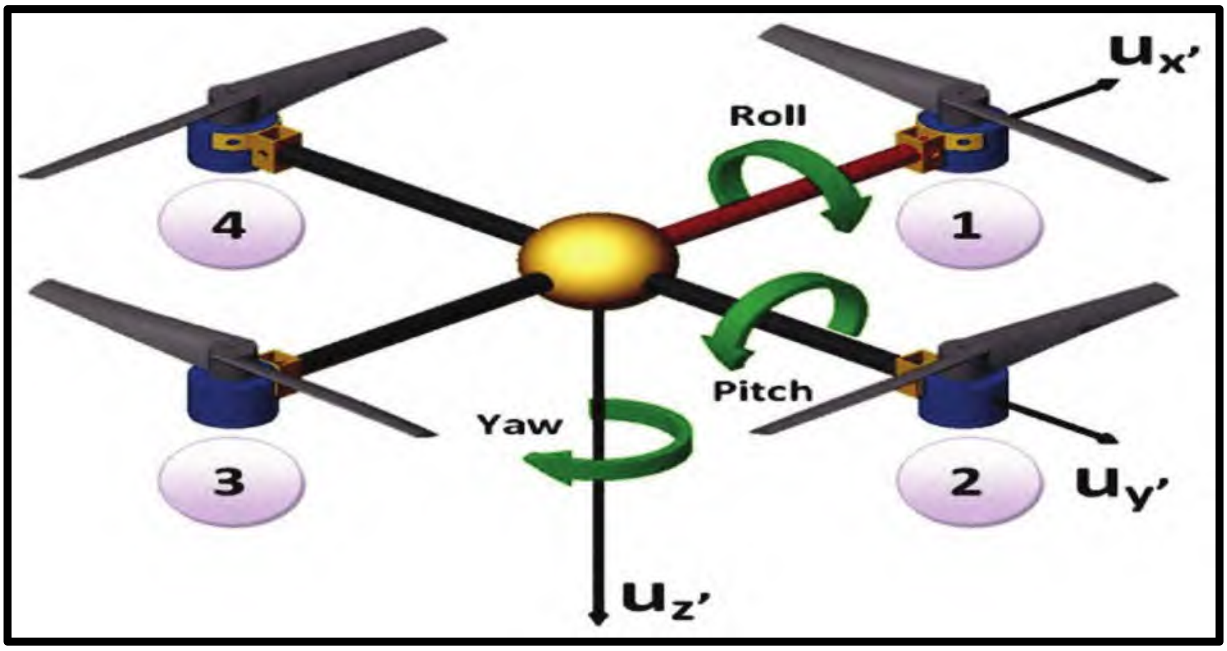


Figure 10.10: Understanding yaw, pitch and roll with respect to a drone frame

Accelerometer Calibration In accelerometer calibration, we need to orient the flight controller by calibrating accelerometer for positional awareness of the drone. The drone has to be moved in all directions namely level, left, right nose up, node down and back as is prompted by voice guide.

Mission Planner > Setup > Mandatory Hardware > Accelerometer Calibration

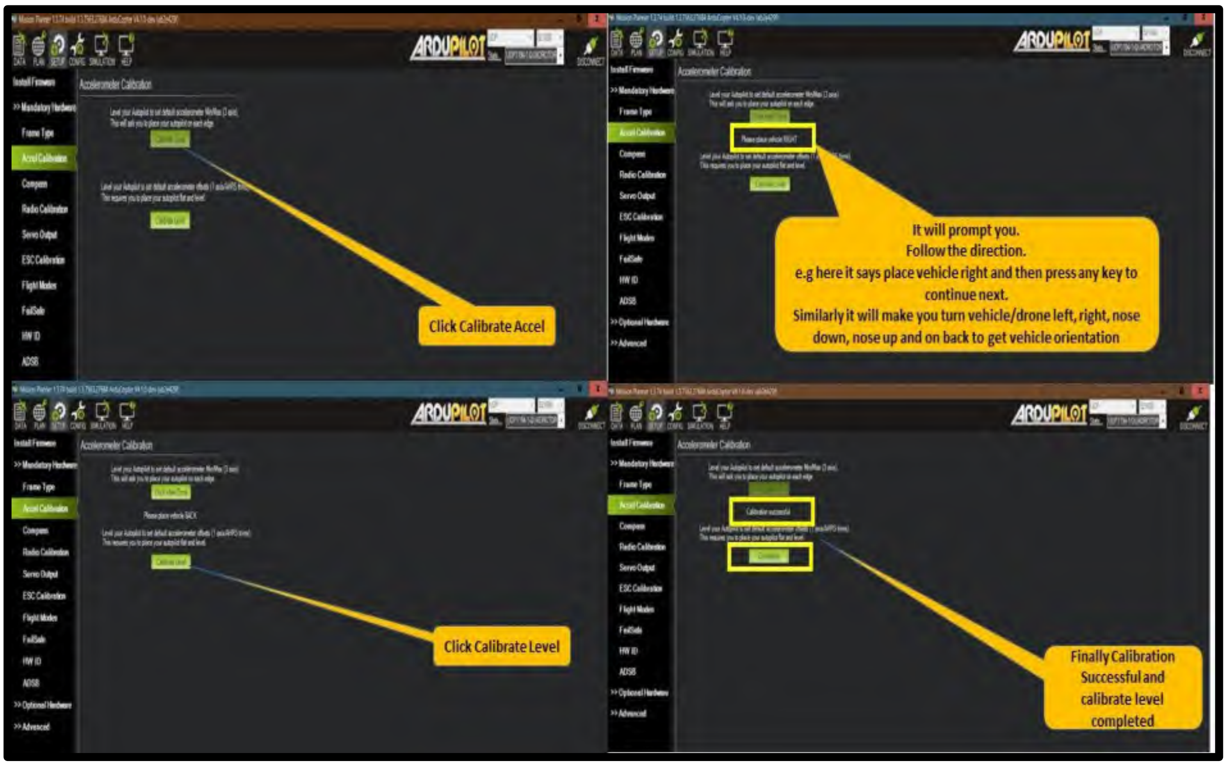

Click Calibrate Accel
It will prompt you. Follow the direction. e.g here it says place vehicle right and then press any key to continue next. Similarly it will make you turn vehicle/drone left, right, nose down, nose up and on back to get vehicle orientation
Click Calibrate Level
Finally Calibration Successful and calibrate level completed

Figure 10.11: Calibrating Accelerometer through Mission Planner (GCS application) to send configuration instructions to FCU (Beagle Bone Blue) through MAVlink

Compass and Magnetometer calibration Next, we need to calibrate compass and magnetometer. We need to remove any missing compass and reboot the vehicle/FCU. Then we can see the compass identified by the ArduPilot application. We then calibrate magnetometer and rotate the vehicle in all directions until success message is received. We now need to reboot the FCU for changes to get saved. We must also ensure that the propellers are never on during this entire process.

Mission Planner > Setup > Mandatory Hardware > Compass and Magnetometer calibration

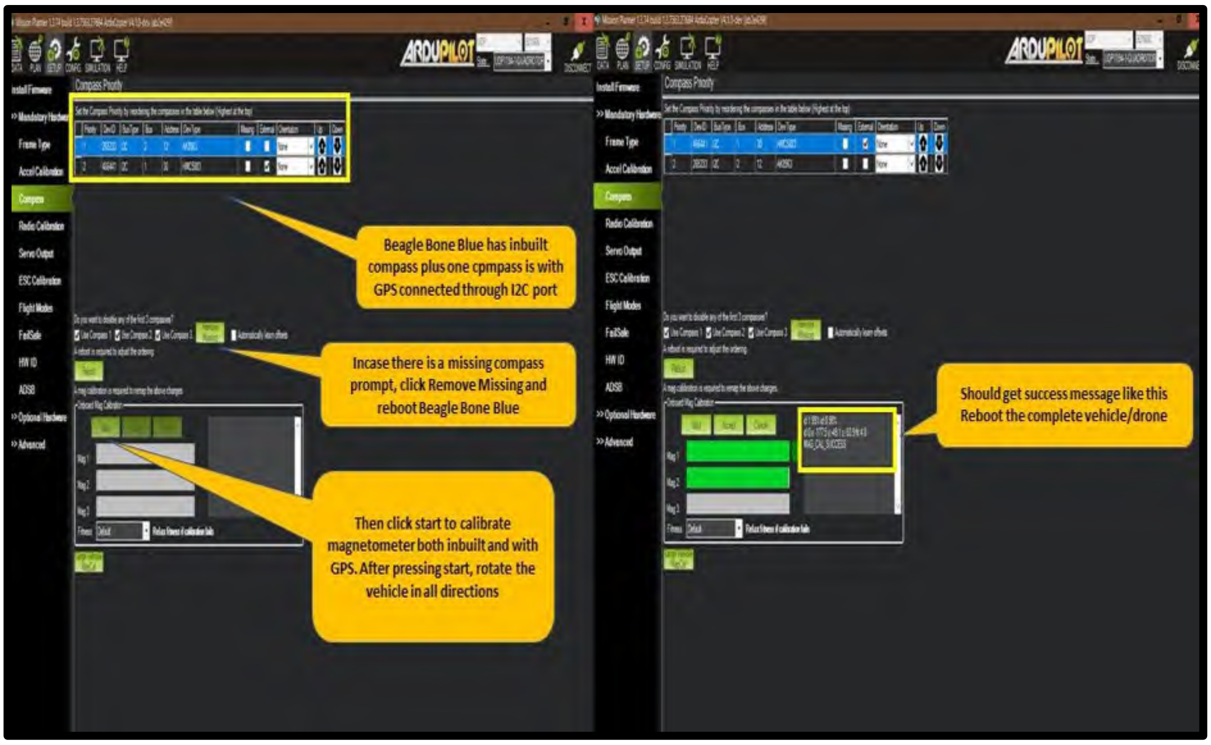
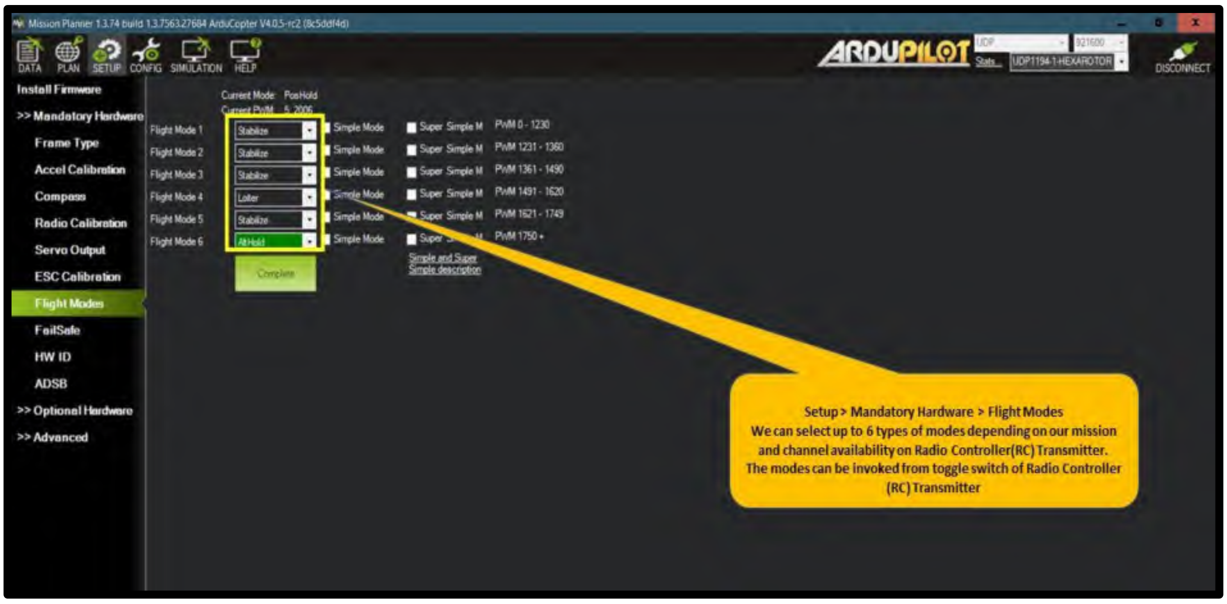

Beagle Bone Blue has inbuilt compass plus one cpmpass is with GPS connected through I2C port
Incase there is a missing compass prompt, click Remove Missing and reboot Beagle Bone Blue
Then click start to calibrate magnetometer both inbuilt and with GPS. After pressing start, rotate the vehicle in all directions
Should get success message like this Reboot the complete vehicle/drone

Figure 10.12: Calibrating Compass and Magnetometer through Mission Planner (GCS application) to send configuration instructions to FCU (Beagle Bone Blue) through MAVlink

Flight Modes. We can select upto six flight modes programmable on the Radio Controller (RC) Transmitter subject to the number of channels it supports. The communication between RC transmitter happens with FCU via RC receiver we saw in Chapter 3 above. The details on flight modes are available later in this Chapter.

Mission Planner > Setup > Mandatory Hardware > Flight Modes


Mission Planner 1.3.74 build 1.3.7563.27684 ArduCopter V4.0.5-rc2 (8c5ddf4d)

Current Mode: PosHold
Current PWM: 5: 2006

Flight Mode 1	Stabilize	Simple Mode	Super Simple M	PWM 0 - 1230
Flight Mode 2	Stabilize	Simple Mode	Super Simple M	PWM 1231 - 1360
Flight Mode 3	Stabilize	Simple Mode	Super Simple M	PWM 1361 - 1490
Flight Mode 4	Loiter	Simple Mode	Super Simple M	PWM 1491 - 1620
Flight Mode 5	Stabilize	Simple Mode	Super Simple M	PWM 1621 - 1749
Flight Mode 6	AltHold	Simple Mode	Super	PWM 1750 +

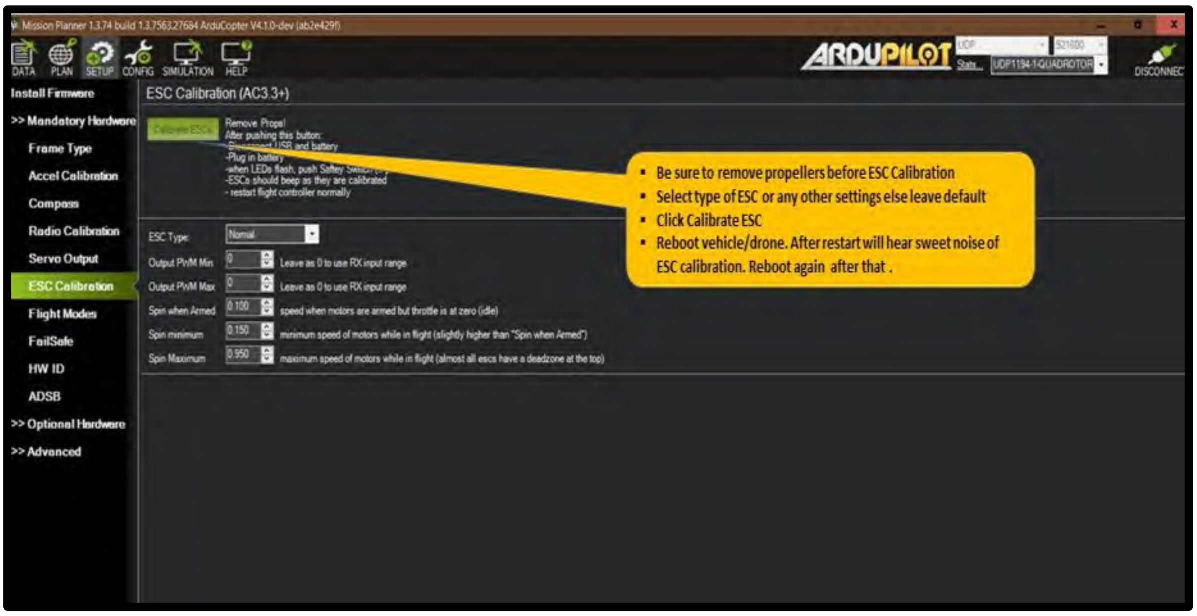
DATA
PLAN
SETUP
CONFIG
SIMULATION
HELP
ARDUPILOT
DISCONNECT
Install Firmware
>> Mandatory Hardware
Frame Type
Accel Calibration
Compass
Radio Calibration
Servo Output
ESC Calibration
Flight Modes
FailSafe
HW ID
ADSB
>> Optional Hardware
>> Advanced
Complete
Simple and Super Simple description
Setup > Mandatory Hardware > Flight Modes
We can select up to 6 types of modes depending on our mission and channel availability on Radio Controller(RC) Transmitter. The modes can be invoked from toggle switch of Radio Controller (RC) Transmitter

Figure 10.13: Selecting Flight Modes through Mission Planner (GCS application) to send

configuration instructions to FCU (Beagle Bone Blue) through MAVlink

Electronic Speed Control (ESC) calibration and motor test Next, we are required to calibrate ESCs so that we can define the minimum and maximum PWM signal for running motors. When we click calibrate ESC, we need to reboot the drone/vehicle (FCU). We hear a sweet sound of ESC calibration on reboot. We must reboot again for getting drone ready for motor test. Thus, we carry out reboot twice to get ESCs and motor ready for dry run. Note all this is done without propellers.

Mission Planner > Setup > Mandatory Hardware > ESC Calibration


Mission Planner 1.3.74 build 1.3.7563.27684 ArduCopter V4.1.0-dev (ab2e429f)

ESC Calibration (AC3.3+)

Remove Props!
After pushing this button:
-Disconnect USB and battery
-Plug in battery
-when LEDs flash, push Saftey Switch
-ESCs should beep as they are calibrated
- restart flight controller normally

ESC Type: Normal
Output PWM Min: 0 Leave as 0 to use RX input range
Output PWM Max: 0 Leave as 0 to use RX input range
Spin when Armed: 0.100 speed when motors are armed but throttle is at zero (idle)
Spin minimum: 0.150 minimum speed of motors while in flight (slightly higher than "Spin when Armed")
Spin Maximum: 0.950 maximum speed of motors while in flight (almost all escs have a deadzone at the top)

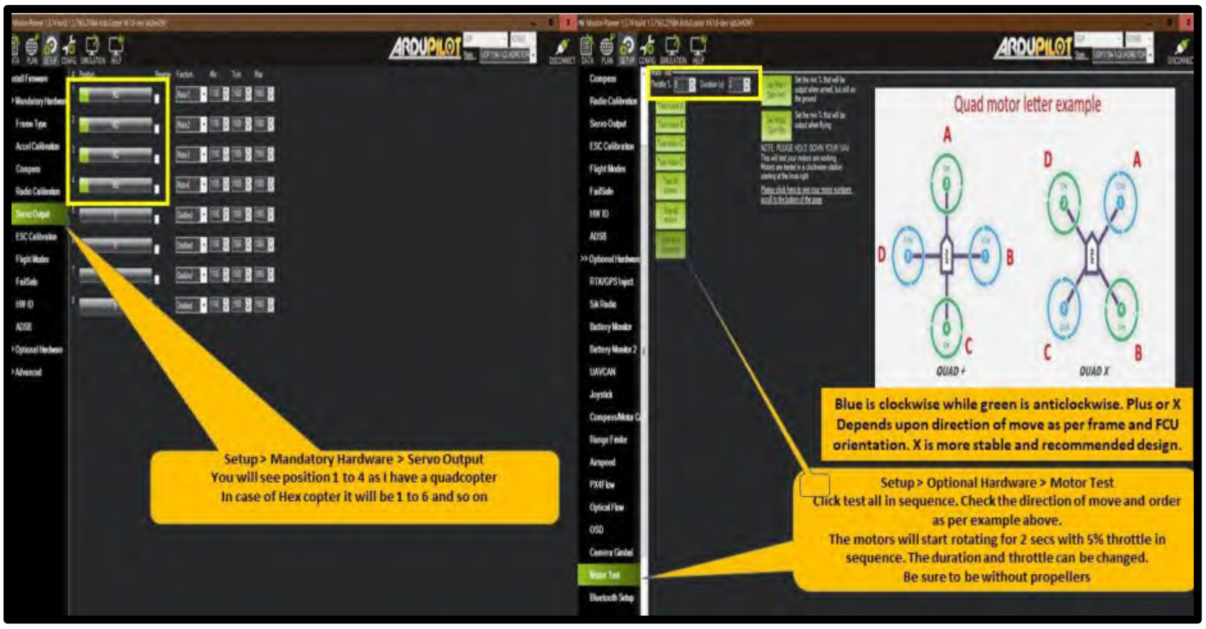
- Be sure to remove propellers before ESC Calibration
- Select type of ESC or any other settings else leave default
- Click Calibrate ESC
- Reboot vehicle/drone. After restart will hear sweet noise of ESC calibration. Reboot again after that .

Figure 10.14: Calibrating Electronic Speed Controller (ESC) through Mission Planner (GCS application) to send configuration instructions to FCU (Beagle Bone Blue) through MAVlink

Servo output. Next, we go to servo output to check if all ESCs have been detected. If it's a quadcopter then we will see the servos green from 1 to 4 and incase of Hexacopter it will be from 1 to 6. After validating, we now carry out ESC calibration and motor test.

Mission Planner > Setup > Mandatory Hardware > Servo Output

Mission Planner > Setup > Optional Hardware > Motor Test



Setup > Mandatory Hardware > Servo Output
You will see position 1 to 4 as I have a quadcopter
In case of Hex copter it will be 1 to 6 and so on

Blue is clockwise while green is anticlockwise. Plus or X Depends upon direction of move as per frame and FCU orientation. X is more stable and recommended design.

Setup > Optional Hardware > Motor Test
Click test all in sequence. Check the direction of move and order as per example above.
The motors will start rotating for 2 secs with 5% throttle in sequence. The duration and throttle can be changed.
Be sure to be without propellers

Figure 10.15: Checking motor/ESC detection and servo output and rotation direction including sequence

Click test all in sequence. Check the direction of move and order as per example above. The motors will start rotating for 2 secs with 5% throttle in sequence. The duration and throttle can be changed. Be sure to be without propellers. Check the picture to ascertain the expected direction of motor rotation and sequence. The clockwise and anti-clockwise motors have to be in their respective locations following the correct sequence as per the type of frame design.

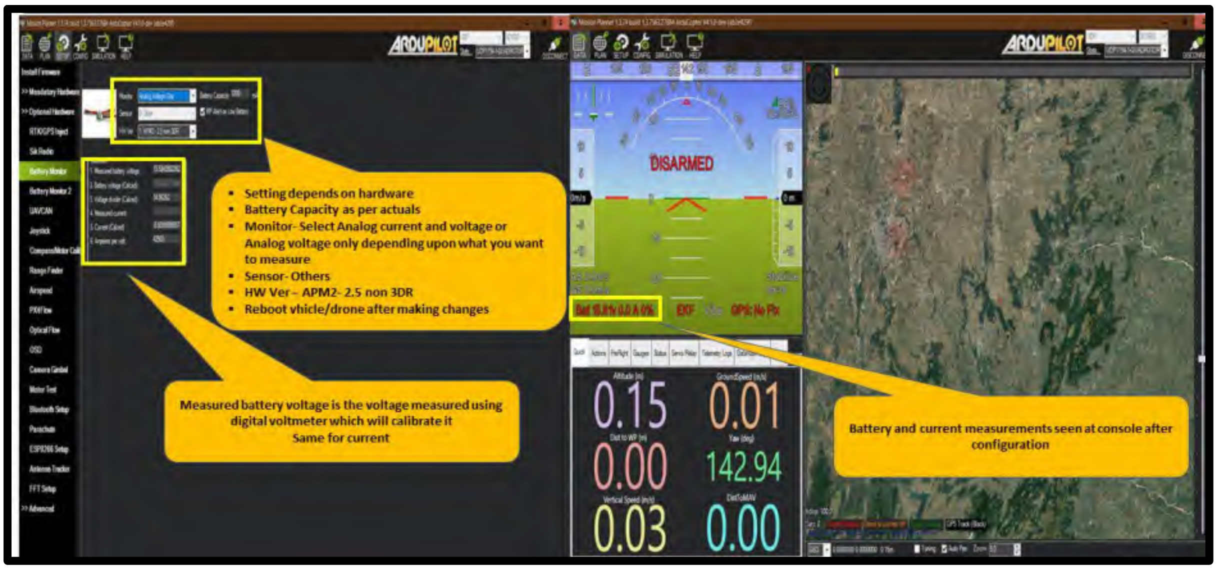
Power module calibration We are using the APM Power module which is between the battery and power distribution board to monitor the current and voltage drawn by the drone (all components). This power module gives battery state to ground operator as well as situational awareness to FCU. The smart battery with Battery Management System (BMS) has been explained in Chapter 7 above but is restricted to bigger drones. Most drones use LIPO batteries with no intelligence, hence this is an important step as battery is the most crucial part of drone working and single point of failure. Even when using smart batteries, having this power module is a good idea as overlay.

Mission Planner > Setup > Optional Hardware > Battery Monitor

Setting depends on hardware used. I have used generic APM power monitoring module. Make the below mentioned changes:_

- Battery Capacity as per actuals

- **Monitor**- Select Analog current and voltage or Analog voltage only depending upon what you want to measure
- **Sensor**- Select Others
- **HW Ver** – APM2- 2.5 non 3DR
- Reboot vehicle/drone after making changes


- Setting depends on hardware
- Battery Capacity as per actuals
- Monitor- Select Analog current and voltage or Analog voltage only depending upon what you want to measure
- Sensor- Others
- HW Ver – APM2- 2.5 non 3DR
- Reboot vhicle/drone after making changes

Measured battery voltage is the voltage measured using digital voltmeter which will calibrate it
Same for current

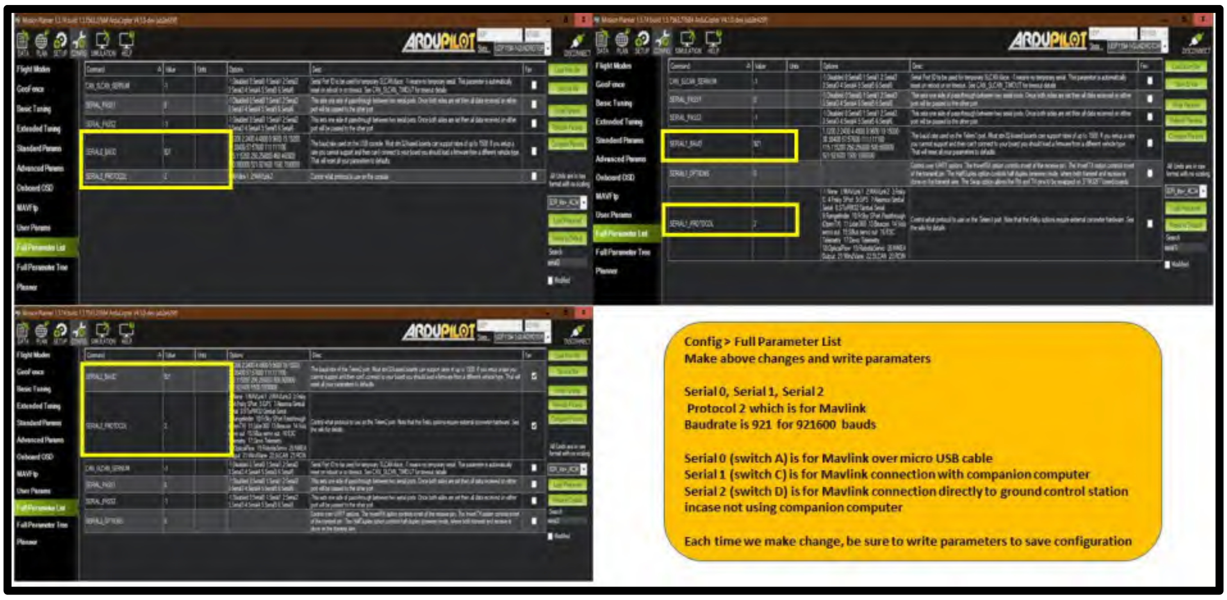
Battery and current measurements seen at console after configuration

Figure 10.16: Configuring battery monitoring through power module being done from Mission Planner (GCS application) and instructions sent to FCU (Beagle Bone Blue) through MAVlink

After reboot, go again and feed the measured values of voltage and current to calibrate the module. We can then see the output on the console as above. **Note**: This power module gives signal output of 5V and is connected to ADC of Beagle Bone Blue which can take only 1.8V max. Thus, we use resistors to divide voltage as shown in Chapter 3 connectivity diagram.

Configuration of parameter list for UARTs We need to configure the UARTs we had logically mapped in Chapter 2 while configuring the Beagle Bone Blue as the FCU. Just a recap that on Beagle Bone Blue, Serial 0 was MAVlink over micro-USB cable, Serial 1 was MAVlink with serial interface of Companion computer and Serial 2 is MAVlink directly with Beagle Bone Blue incase connecting directly to GCS (small drones). Now on the Mission Planner end, we need to define serial0, serial1 and serial2 protocols as MAVlink2 and baudrate as 921600. (Serial0 is configured by default so micro USB will work by default to connect for MAVlink). This configuration will enable the ports to act as MAVlink interfaces for Ardupilot application which is running on FCU. Make sure to write parameters after making changes.

Mission Planner > Config > Full Parameter List > Serial0 or Serial1 or Serial2


Config > Full Parameter List
Make above changes and write paramaters

Serial 0, Serial 1, Serial 2
Protocol 2 which is for Mavlink
Baudrate is 921 for 921600 bauds

Serial 0 (switch A) is for Mavlink over micro USB cable
Serial 1 (switch C) is for Mavlink connection with companion computer
Serial 2 (switch D) is for Mavlink connection directly to ground control station incase not using companion computer

Each time we make change, be sure to write parameters to save configuration

Figure 10.17: UART configuration through Mission Planner (GCS application) and instructions sent to FCU (Beagle Bone Blue) through MAVlink

Selection and configuration of proximity sensor (LIDAR or mmWave radar)_

LIDAR Configuration If we wish to use LIDAR as our proximity sensor, then we need to configure the serial 5 in full parameter list as given below. Refer Chapter 2 where we had logically mapped UART 5 of Beagle Bone Blue with UART 5 of ArduPilot (Switch F). The physical connection of LIDAR is also at UART 5 of Beagle Bone Blue FCU (Refer Chapter 3 for physical connectivity diagram). Change Serial5_PROTOCOL to any other make of LIDAR incase you wish to use some other make or model from the list of supported LIDARs.

- **Mission Planner > Config > Full Parameter List > Serial 5**
- **SERIAL5_PROTOCOL = “11” (“Lidar360”)**
- **SERIAL5_BAUD = “115” if using Serial1**
- **PRX_TYPE = “5”**
- **PRX_ORIENT = “0” if mounted on the top of the vehicle, “1” if mounted upside-down on the bottom of the vehicle.**

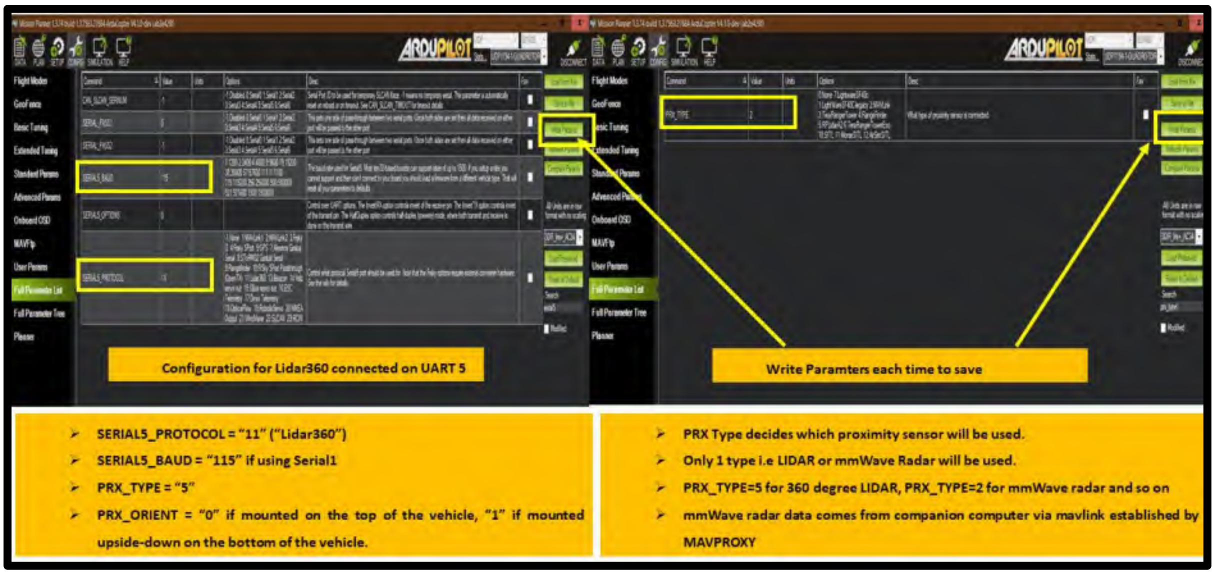
In our configuration above in the previous chapters, one can note that mmWave radar if connected to the drone (companion computer) will be sending data as master python script runs automatically on startup. But the choice to select and read proximity data from LIDAR or mmWave radar rests with Ardupilot running on FCU. It is the PRX_TYPE which determines the selection.

For LIDAR

Mission Planner > Config > Full Parameter List > PRX_TYPE=5

For mmWave radar, its 2 means data will come on MAVlink via Companion Computer

Mission Planner > Config > Full Parameter List > PRX_TYPE=2



- SERIAL5_PROTOCOL = "11" ("Lidar360")
- SERIAL5_BAUD = "115" if using Serial1
- PRX_TYPE = "5"
- PRX_ORIENT = "0" if mounted on the top of the vehicle, "1" if mounted upside-down on the bottom of the vehicle.

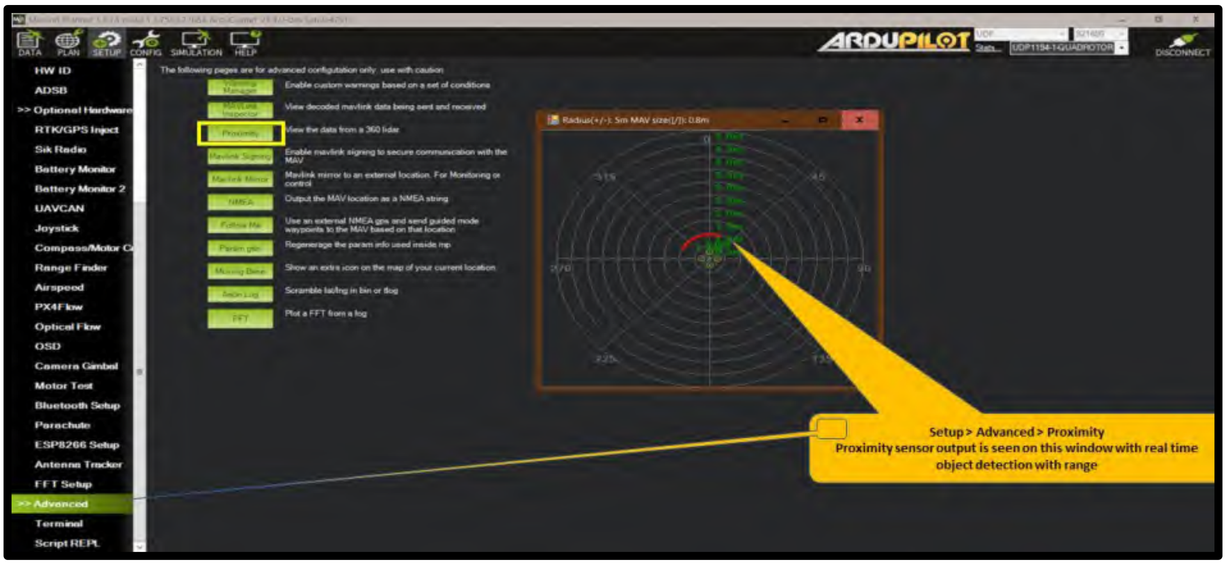
- PRX Type decides which proximity sensor will be used.
- Only 1 type i.e LIDAR or mmWave Radar will be used.
- PRX_TYPE=5 for 360 degree LIDAR, PRX_TYPE=2 for mmWave radar and so on
- mmWave radar data comes from companion computer via mavlink established by MAVPROXY

Figure 10.18: UART configuration for LIDAR through Mission Planner (GCS application) and instructions sent to FCU (Beagle Bone Blue) through MAVlink

The proximity sensor data can be seen on Mission Planner irrespective of the choice of sensor as seen below:-

Mission Planner > Setup > Advanced Hardware > Proximity

(Ctrl+F is the shortcut)



The following pages are for advanced configutation only. use with caution

Enable custom warnings based on a set of conditions
View decoded mavlink data being sent and received
View the data from a 360 lidar
Enable mavlink signing to secure communication with the MAV
Mavlink mirror to an external location. For Monitoring or control
Output the MAV location as a NMEA string
Use an external NMEA gps and send guided mode waypoints to the MAV based on that location
Regenerage the param info used inside mp
Show an extra icon on the map of your current location
Scramble lat/lng in bin or tlog
Plot a FFT from a log

Setup > Advanced > Proximity
Proximity sensor output is seen on this window with real time object detection with range

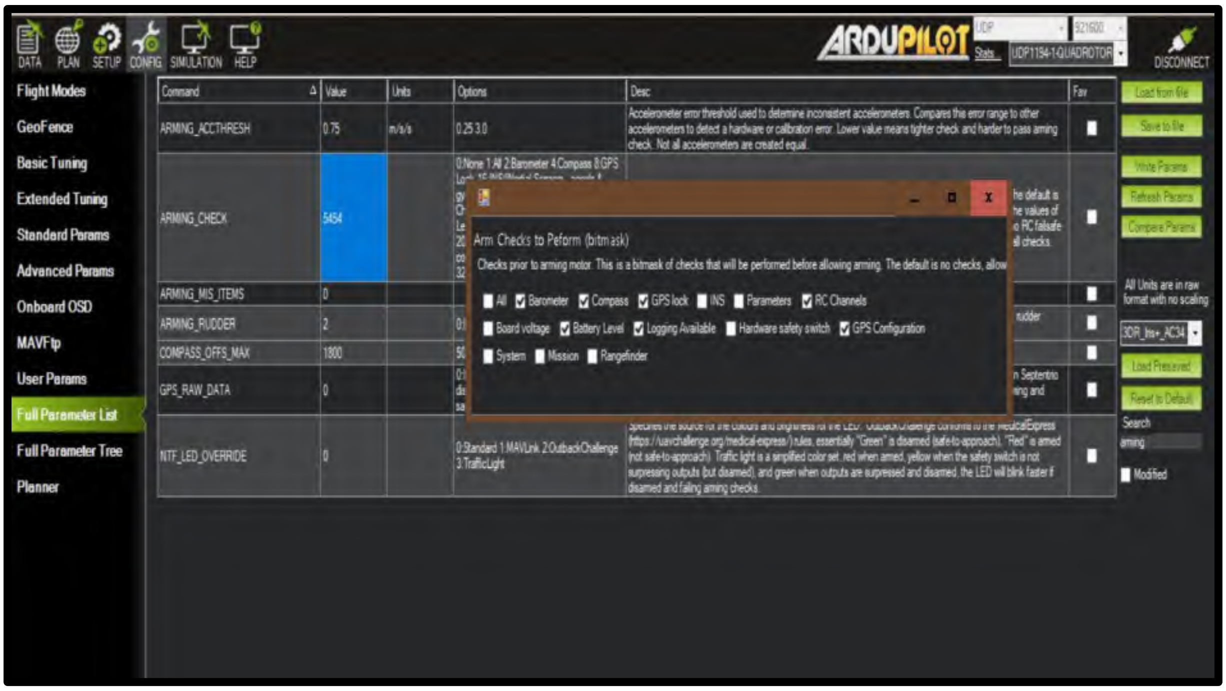
Figure 10.19: Proximity sensor output in Mission Planner (GCS application). The choice of proximity sensor between LIDAR and mmWave radar is as per instructions above. The instructions sent to FCU (Beagle Bone Blue) through MAVlink

10.2 Pre-Arm safety checks and failsafe configuration [3]

The ArduPilot autopilot application includes a set of Pre-arm safety checks. These ensure that the drone/vehicle will not arm i.e the motors will not fire up until all the set of conditions satisfied. It acts as a critical safety check to prevent any risk to drone or the operator. The points on the pre-arm safety checklist can be enabled or disabled as per user requirement. The pre-arm safety checks are performed by Ardupilot autopilot application, each time we power on the drone/vehicle. Thus we can select the arming checklist to ensure that the necessary actions are pre-requisite to arming of motors for safety of both the pilot and the drone. The settings can be done as per path given below

Mission Planner > Config > Full Parameter List > ARMING_CHECK

After making requisite changes, the parameters need to be written by clicking write parameters to save it permanently. The FCU needs to be rebooted after any change. The option to write parameters is given on right side as seen in the picture below:



Command	Value	Units	Options	Desc	Fav
ARMING_ACCTHRESH	0.75	m/s/s	0.25 3.0	Accelerometer error threshold used to determine inconsistent accelerometers. Compares this error range to other accelerometers to detect a hardware or calibration error. Lower value means tighter check and harder to pass arming check. Not all accelerometers are created equal.	☐
ARMING_CHECK	5454		0:None 1:All 2:Barometer 4:Compass 8:GPS		☐
ARMING_MIS_ITEMS	0				☐
ARMING_RUDDER	2				☐
COMPASS_OFFS_MAX	1800				☐
GPS_RAW_DATA	0				☐
NTF_LED_OVERRIDE	0		0:Standard 1:MAVLink 2:OutbackChallenge 3:TrafficLight	(https://uavchallenge.org/medical-express/) rules, essentially "Green" is disarmed (safe-to-approach), "Red" is armed (not safe-to-approach). Traffic light is a simplified color set, red when armed, yellow when the safety switch is not surpressing outputs (but disarmed), and green when outputs are surpressed and disarmed, the LED will blink faster if disarmed and failing arming checks.	☐

Arm Checks to Peform (bitmask)

Checks prior to arming motor. This is a bitmask of checks that will be performed before allowing arming. The default is no checks, allow

☐ All ☑ Barometer ☑ Compass ☑ GPS lock ☐ INS ☐ Parameters ☑ RC Channels
☐ Board voltage ☑ Battery Level ☑ Logging Available ☐ Hardware safety switch ☑ GPS Configuration
☐ System ☐ Mission ☐ Rangefinder

Figure 10.20: Arming check selection option

Whenever the drone is connected to the Mission Planner and the pilot tries to arm the aircraft using Radio Controller (RC) transmitter, the pre-arm check failure messages will be displayed to inform the pilot to take corrective action. Some of these messages are:-

GPS related failure messages and their reasons:

GPS Glitch : This message appears when the drone is in a flight mode such as Auto or Position hold which requires GPS. Hence, the user is advised to change mode to stabilize which is completely manual mode not requiring GPS and troubleshoot the issue of GPS glitch which could be temporary in nature or require hardware change (GPS) change.

Need 3D Fix : This message informs the pilot that again GPS hasn't got 3D fix. The GPS uses 3D trilateration to determine its on the earth's surface with a minimum of four satellites. The pilot must wait, take the drone to an open area and ensure 3D fix before using it in flight modes requiring GPS else operate the drone in stabilize mode where the controls are manual and does not require GPS.

Bad Velocity : This message informs the Pilot that the drones's velocity as per its inertial navigation system is above 50cm/s. Possible reasons for this error message are bad accelerometer calibration, GPS updating below 5Hzs or drone moving/dropping.

High GPS HDOP : The GPS's high dilution of precision (HDOP) message above 2.0 value is undesirable for safety of drone operation. The pilot needs to wait for this error to auto resolve on its own as the GPS finds more and more satellites or change position. Normally a count of 12 satellites is good enough to get the requisite HDOP. This error could also be due to GPS interferences which needs to be resolved by moving any such source of interference away.

The pilot has the option to disable the fence and take-off in a flight mode which does not require GPS such as stabilize or a AltHold and then switch to Loiter after arming. However, this is not recommended and GPS issues be resolved before takeoff.

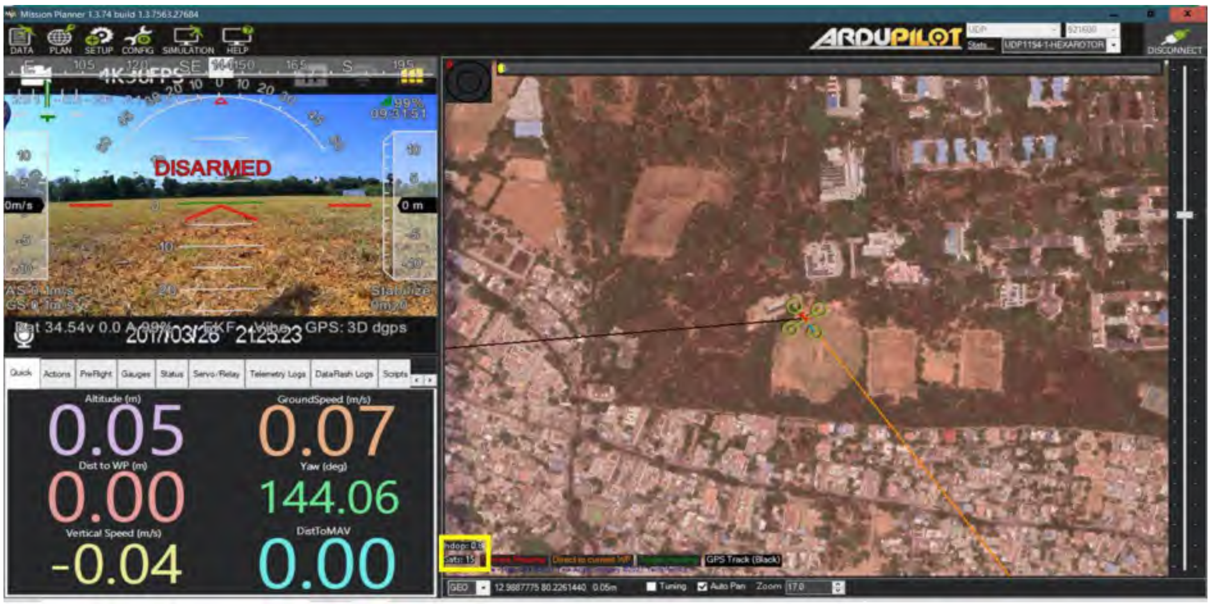


Figure 10.21: Satellite counts and HDoP value in yellow box

Inertial Navigation System (INS) related error messages:

INS not calibrated: This error message comes when one or all of the accelerometer's offsets are zero which requires the pilot to recalibrate the accelerometer as shown earlier in the chapter.

Accels or Gyros not healthy: This error message usually means a hardware issue or a recent firmware update related issue. The pilot needs to get this issue resolved by exploring both options before flying. Hardware issues are mostly resolved by replacing the faulty hardware but firmware update issues require either reverting back to the old firmware, or carrying out update again. Incase the firmware update issue is identified; it must be reported to Ardupilot developers on the forum to find possible solutions or issue fixes for resolving the same.

Accels inconsistent: This error message is again an indication to recalibrate accelerometer as shown earlier in the chapter or carry out hardware change incase recalibration does not help.

Gyro cal failed: This error message indicate that the gyro calibration has failed to capture the offsets. This in most cases is caused as a result of drone being moved during the calibration process. It can likely be resolved by unplugging and plugging the battery again without jostling the drone. If the issue remains, then it could also be due to sensors hardware failure such as spikes).

Gyros inconsistent: The error message is displayed when two gyroscopes are reporting drone rotation rates differing by more than 20deg/sec. This could be due to a hardware failure or caused due to bad gyro calibration.

Battery/Power Monitor message:

The power module calibration that we did earlier ensures that both voltage and current utilizations are tracked for the drone. The pilot or drone operator has the option of providing minimum voltage level as the failsafe value below which the drone will alert the drone pilot/operator and carry out emergency action i.e land or return to launch (RTL) as per user settings incase the drone is airborne. In pre-arm check it simply alerts the pilot/operator to check battery or replace faulty power module. We have the option of configuring minimum arming voltage and remaining capacity parameters for the battery/power monitor by adding values to the BATT_ARM_VOLT and BATT_ARM_MAH parameters in complete parameters list.

Mission Planner > Config > Full Parameter List > BATT_ARM_VOLT

Mission Planner > Config > Full Parameter List > BATT_ARM_MAH

This ensure and checks for the battery above failsafe levels and has enough capacity for flight operations.

Radio failsafe error message:

This message is displayed when the radio link between the drone and Radio Controller (RC) receiver is broken. During pre-arm it could be due to hardware failure such broken antenna or improper settings of RC transmitter. During flight this message could be due to drone being out of range. The drone has a setting to return to launch (RTL) incase of this error message. We will disable this message for auto or guided mission as our MAVlink will be on 4G LTE modem based internet and the range of drone operation much more that radio link range of RC transmitter and receiver. The disabling of the setting for auto flight mode will be covered in next sub chapter.

No proximity sensor:

This error message is displayed when proximity sensor settings are enabled but sensor signal is unavailable. We need to check for physical connectivity issues or hardware error to resolve this.

10.3 Understanding Flight modes [5]

After understanding the MAVlink protocol, we also require to understand the flight modes to be able to proceed with system configuration and piloting of unmanned system running Ardupilot and MAVLink protocol. Ardupilot defines multiple flight modes, some important ones are listed below:

The STABILIZE mode enables complete control to user. The drone will respond to every input provided by user using RC transmitter. Its like a manual override option available for user to take control from autopilot (autonomous mission) incase of issues with the drone. It requires careful handling of the drone and practice to operate the drone in this mode. We can analyze the quality of piloting of drone by analyzing logs which has been explained in detail in subsequent chapter on post-mission analysis.

ALTITUDE HOLD (ALT_HOLD) mode automatically controls the altitude of the unmanned system by the autopilot. However, it does not control the heading and position of the drone and is controlled by user instead. It’s a relatively easier mode to operate the drone as the user does not have to bother to maintain altitude. This mode is more recommended for beginners and does not require a GPS as it estimates the altitude with the barometer. It is possible to tune the setting of the proportional–integral–derivative (PID) controller of the

ALT_HOLD mode in GCS (Mission Planner). The altitude is maintained with proportional controller. It estimates the error between the desired altitude and the actual altitude and in turn tunes the vertical acceleration proportionally to the error. The proportional gain can be set through a GCS (Mission Planner) and will be discussed later in this chapter. The Proportional gain has to be selected keeping in mind the fact that a very high gain makes the control more aggressive and less stable, on the contrary a very low gain will make the control sluggish and nonresponsive.

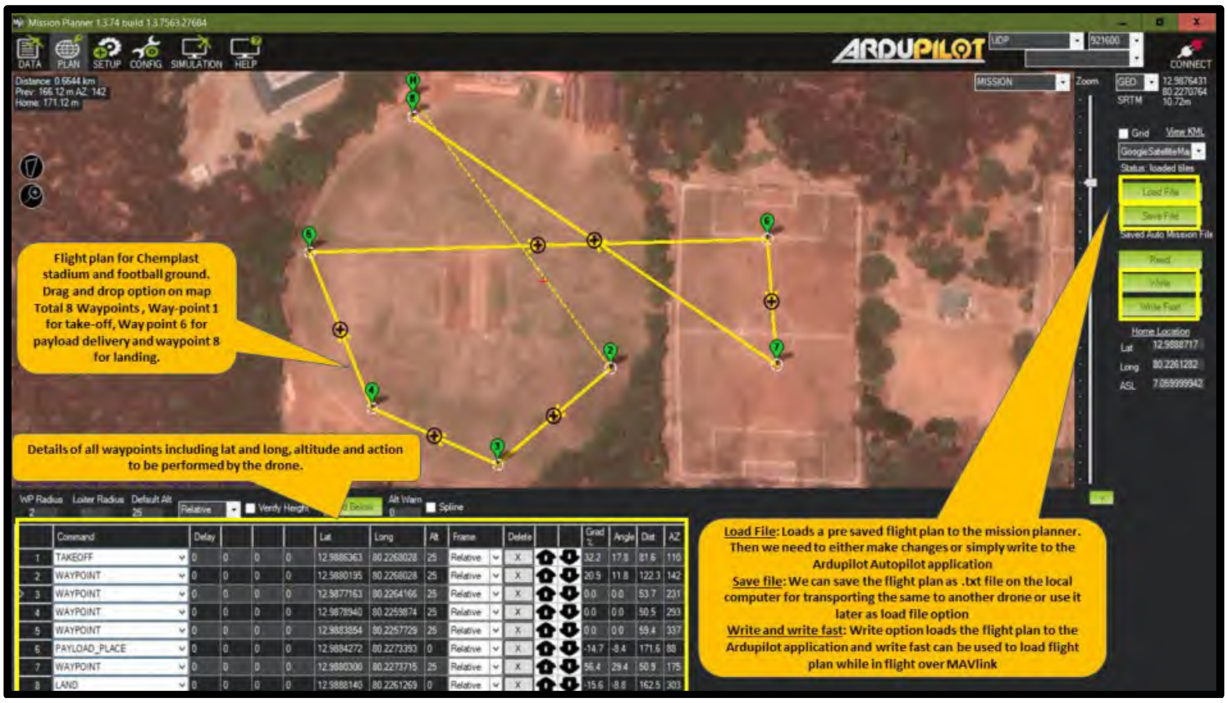
LOITER mode is even more convenient than the previous ALT_HOLD mode as the drone in this mode maintains the altitude, its position and heading once the user stops giving control signals through RC Transmitter. It kind of freezes in the place last left by user. In this mode the drone maintains the current location, orientation and altitude last operated by the user. So it's a combination mode with STABLIZE mode till the time user is operating and then goes in auto mode when user stops providing it any input though RC transmitter. The LOITER mode requires a GPS 3D fix and HDOP (Horizontal dilution of precision) is smaller than 2.0) for it to work or an optical flow. We cannot arm the vehicle in LOITER mode unless we meet the requirement. To achieve better performance, the drone must be in low magnetic interference of the compass and low vibrations. The PID controller gains can be tuned from the GCS (Mission Planner). The LOITER SPEED represents maximum horizontal speed in cm/s which is typically 5m/s. The default configuration is that the maximum acceleration is half of LOITER SPEED which is 2.5m/s2. The parameters of the LOITER mode can be configured through a GCS (Mission Planner) by setting PID control gains of the altitude, position and orientation (Yaw, Pitch and Roll) and is discussed later in the chapter.

The **LAND** mode forces the drone to land to the ground mostly incase of emergency.

The **RTL** (Return to Launch) mode forces the drone to return to start position from where it took-off. Both LAND and RTL mode are used in case of violation of navigation safety and geofence. An example of RTL is on loss of communication for a particular duration specified before take-off. The configuration of drone to LAND immediately or RTL automatically incase of battery going below threshold, is called GEOFENCE.

The **GUIDED** mode is one of the most important mode which operates only with GPS. The GPS of drone performs 3D fix and is activated, then the drone navigates autonomously to a particular GPS coordinate as provided by GCS. As the name suggests, the drone in the GUIDED mode is guided by the user to navigate autonomously to a specific waypoint as provided by the user. The GUIDED mode which works in conjunction with GPS, allows the user to send the drone to specified waypoint as defined by their GPS coordinates. The drone is not armed in the GUIDED mode until it achieves the GPS 3D fix. A GCS is used to send a navigation waypoint to the drone over MAVlink to navigate to it thereby requiring complete reliability over the link. We can click on map to provide next waypoint to the drone.

The **AUTO** mode refers to the autonomous mode, where the drone follows a predefined mission where a set of waypoints is fed and stored in ardupilot autopilot application. The drone follows the aerial path through waypoints in the same order as defined by the user before the start of mission. Utilizing the 4G LTE communication for Beyond Visual Line of Sight (BVLOS) communication, we intend to use the AUTO and guided modes without the need for RC transmitter. Hence, we need to make changes to the drone configuration to allow its operation in AUTO or GUIDED mode. When we are sure about our way points we use AUTO mode, but when we need to guide the drone with waypoints changing on the fly we prefer GUIDED mode.


Mission Planner 1.3.74 build 1.3.7563.27684
DATA PLAN SETUP CONFIG SIMULATION HELP
ARDUPILOT
CONNECT
Load File
Save File
Read
Write
Write Fast
Home Location

Flight plan for Chemplast stadium and football ground. Drag and drop option on map Total 8 Waypoints , Way-point 1 for take-off, Way point 6 for payload delivery and waypoint 8 for landing.

Details of all waypoints including lat and long, altitude and action to be performed by the drone.

WP Radius 2 | Loiter Radius | Default Alt 25 | Relative | Verify Height | Alt Warn 0 | Spline

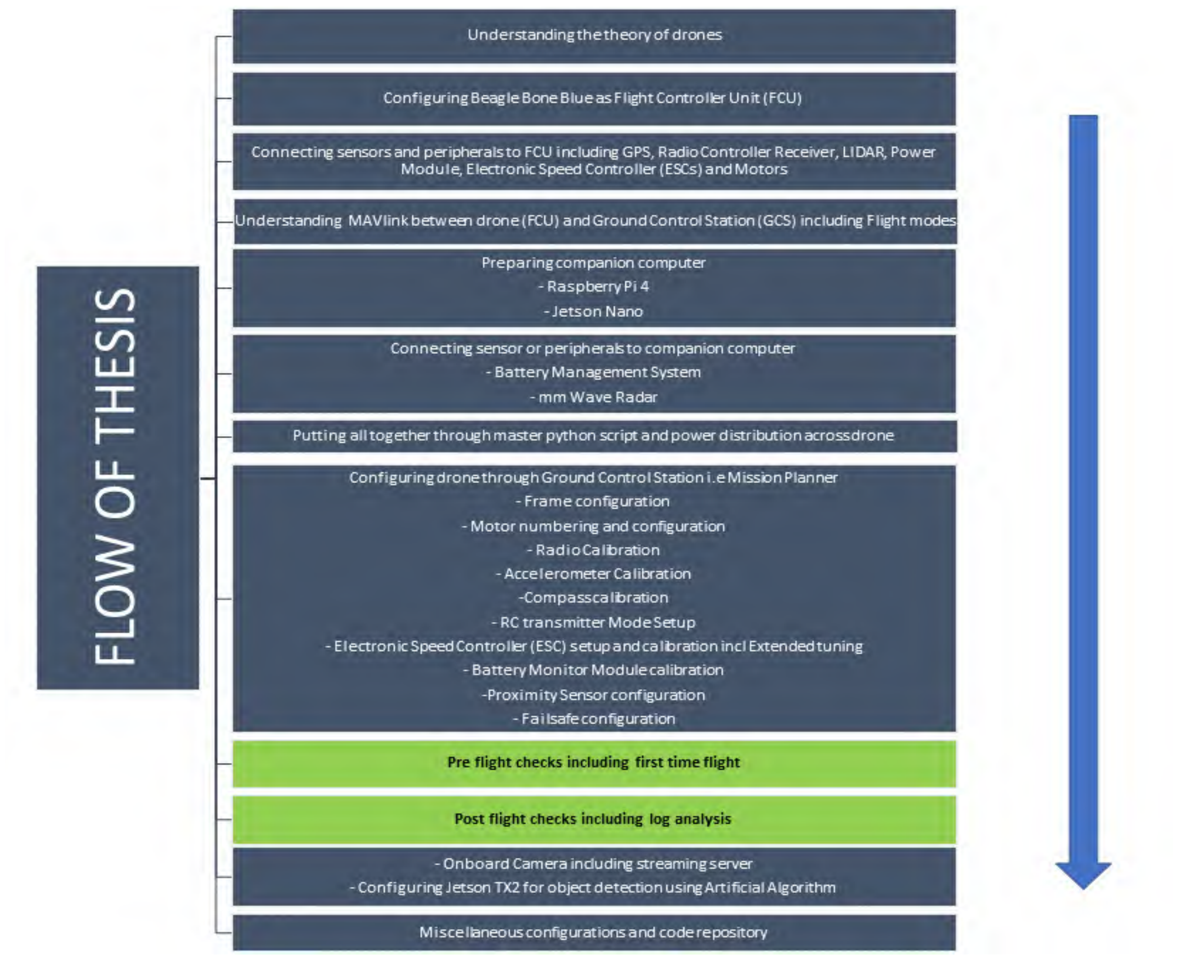
	Command	Delay				Lat	Long	Alt	Frame	Delete			Grad %	Angle	Dist	AZ
1	TAKEOFF	0	0	0	0	12.9886363	80.2268028	25	Relative	X	⬆	⬇	32.2	17.8	81.6	110
2	WAYPOINT	0	0	0	0	12.9880195	80.2268028	25	Relative	X	⬆	⬇	20.5	11.8	122.3	142
3	WAYPOINT	0	0	0	0	12.9877163	80.2264166	25	Relative	X	⬆	⬇	0.0	0.0	53.7	231
4	WAYPOINT	0	0	0	0	12.9878940	80.2259874	25	Relative	X	⬆	⬇	0.0	0.0	50.5	293
5	WAYPOINT	0	0	0	0	12.9883854	80.2257729	25	Relative	X	⬆	⬇	0.0	0.0	59.4	337
6	PAYLOAD_PLACE	0	0	0	0	12.9884272	80.2273393	0	Relative	X	⬆	⬇	-14.7	-8.4	171.6	88
7	WAYPOINT	0	0	0	0	12.9880300	80.2273715	25	Relative	X	⬆	⬇	56.4	29.4	50.9	175
8	LAND	0	0	0	0	12.9888140	80.2261269	0	Relative	X	⬆	⬇	-15.6	-8.8	162.5	303

Load File: Loads a pre saved flight plan to the mission planner. Then we need to either make changes or simply write to the Ardupilot Autopilot application
Save file: We can save the flight plan as .txt file on the local computer for transporting the same to another drone or use it later as load file option
Write and write fast: Write option loads the flight plan to the Ardupilot application and write fast can be used to load flight plan while in flight over MAVlink

Figure 10.22: Auto flight plan including guided flight mode option (Chemplast IITM)

Be sure to select Mission Planner > Setup > FailSafe > Radio > Enabled continue with mission in Auto Mode

10.4 Understanding error messages during flight and Post mission flight log analysis


FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server
- Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository
Figure 10.23: Flow of thesis until now

After doing are complete configuration and calibration of the drone, the drone is good enough to fly. However, we must carry out a limited duration flight test to evaluate the logs. The logs help us to understand the complete behavior of the drone for any possible error to prevent loss of aircraft or injury during its operation. Also the logs help us fine tune the various settings esp PID (Proportional - Integral - Derivative) tuning for optimum performance. The analysis of the logs must be done after each flight to improving the performance of drone iteratively and also prevent/avoid future failures by picking on the tell-tale signs as part of best practice.

The logs are stored in two ways during a flight as Telemetry Logs or Data Flash logs.

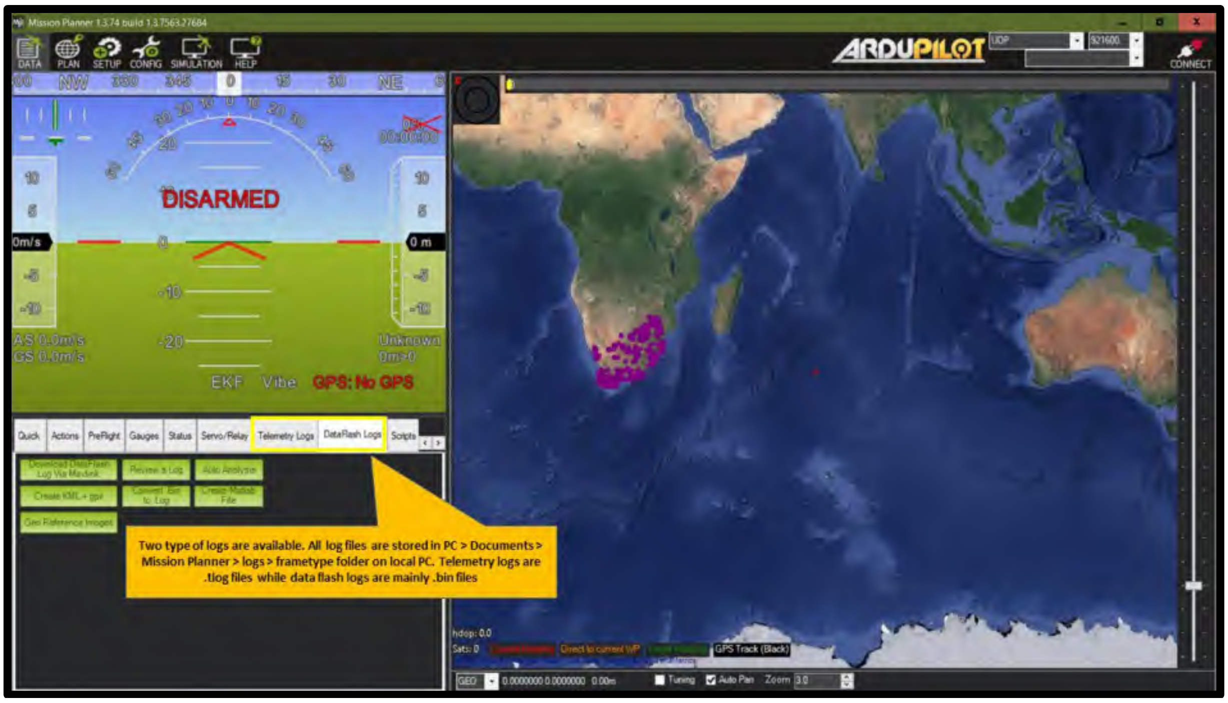
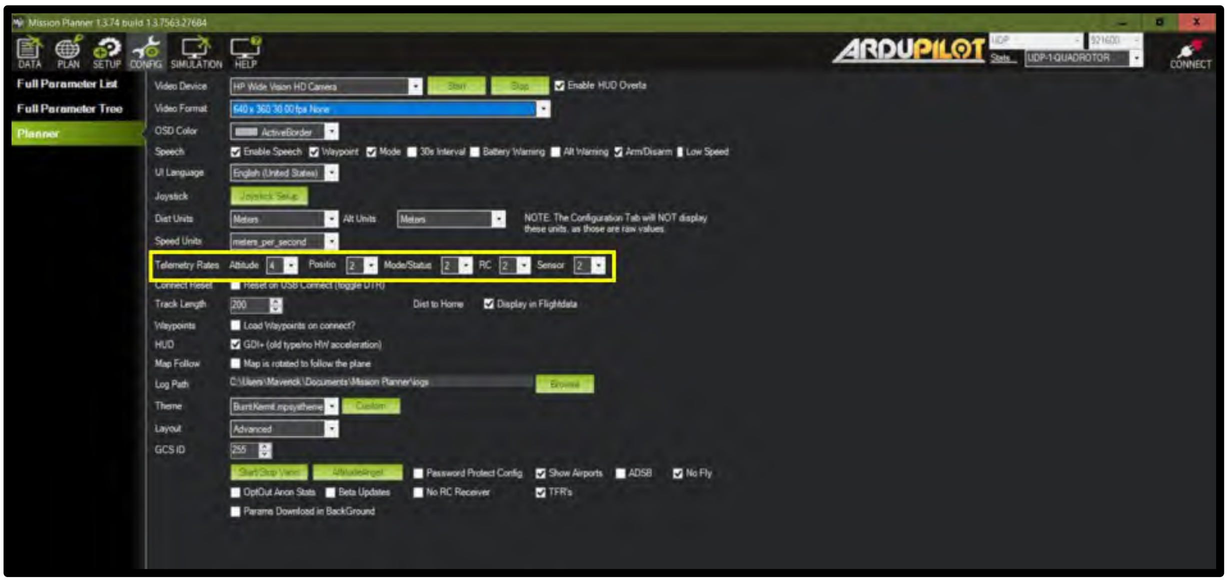

Two type of logs are available. All log files are stored in PC > Documents > Mission Planner > logs > frametype folder on local PC. Telemetry logs are .tlog files while data flash logs are mainly .bin files

Figure 10.24: Options to retrieve logs in Mission Planner

The desired data rate at which the data is sent from the autopilot to the ground station can be controlled through the mission planner though **Mission Planner > Config > Planner** As all the logging data is sent over the telemetry link (MAVlink).


Video Device	HP Wide Vision HD Camera; ✓ Enable HUD Overla
Video Format	640 x 360 30.00 fps None
OSD Color	ActiveBorder
Speech	✓ Enable Speech; ✓ Waypoint; ✓ Mode; 30s Interval; Battery Warning; Alt Warning; ✓ Arm/Disarm; Low Speed
UI Language	English (United States)
Dist Units	Meters; Alt Units Meters
Speed Units	meters_per_second
Telemetry Rates	Attitude 4; Positio 2; Mode/Status 2; RC 2; Sensor 2
Connect Reset	Reset on USB Connect (toggle DTR)
Track Length	200; Dist to Home ✓ Display in Flightdata
Waypoints	Load Waypoints on connect?
HUD	✓ GDI+ (old type/no HW acceleration)
Map Follow	Map is rotated to follow the plane
Log Path	C:\Users\Maverick\Documents\Mission Planner\logs
Theme	BurntKermit.mpsystheme
Layout	Advanced
GCS ID	255

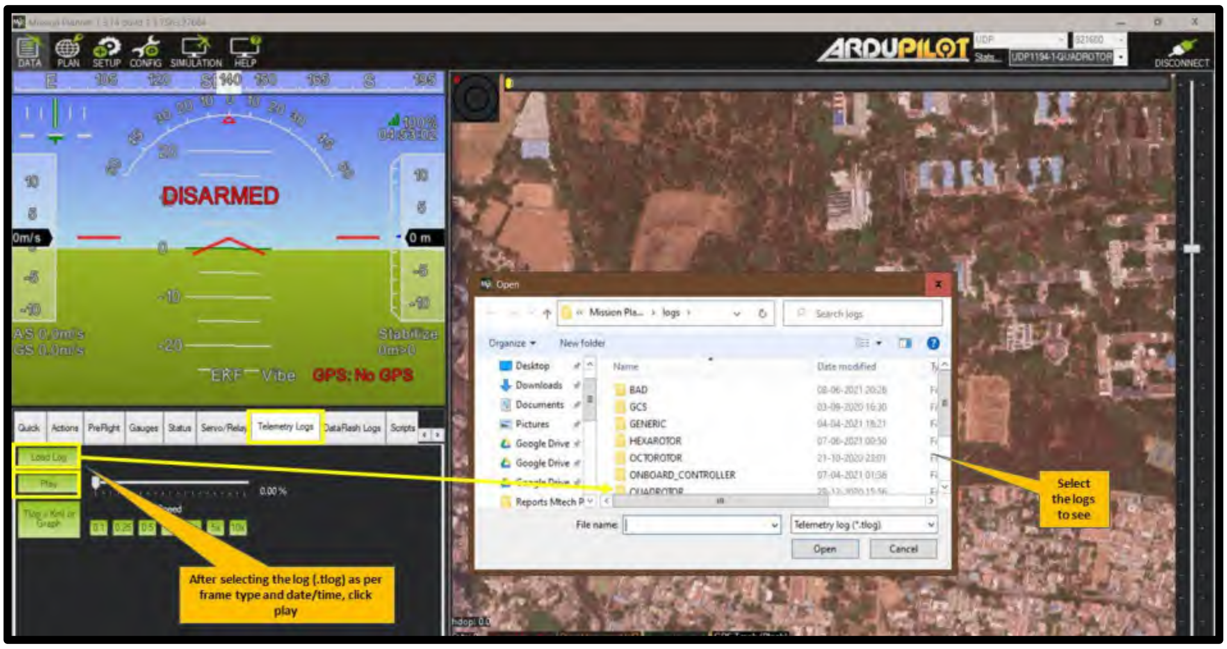
NOTE: The Configuration Tab will NOT display these units, as those are raw values.

Password Protect Config; ✓ Show Airports; ADSB; ✓ No Fly; OptOut Anon Stats; Beta Updates; No RC Receiver; ✓ TFR's; Params Download in BackGround

Figure 10.25: Option for changing data rate option for logs as per bandwidth requirement

Telemetry logs (also known as “tlogs”): These logs are recorded by the Mission Planner which is the GCS on the local laptop/PC running the application as and when drone is connected on telemetry link over radio or 4G LTE Modem. The log files are made in the format

YYYY-MM-DD hh-mm-ss.tlog and can be accessed from **Document** > **Mission Planner** > **log** > **frametype**.



Select the logs to see
After selecting the log (.tlog) as per frame type and date/time, click play
DISARMED
Telemetry Logs
Load Log
Play

Figure 10.26: Procedure to retrieve Telemetry Logs

Mission Planner > **Telemetry logs** >**Tuning** displays the parameters of the log we want to see. Incase we want to change the parameters of logs which we wish to see, double click on the tuning top bar and the select window opens.

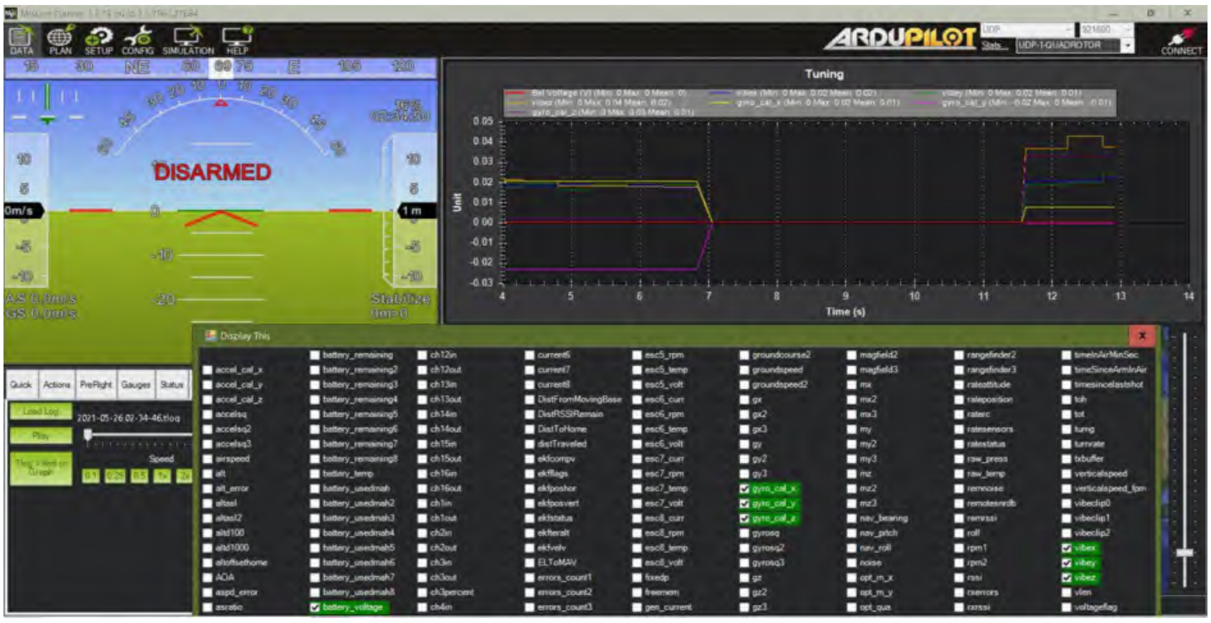
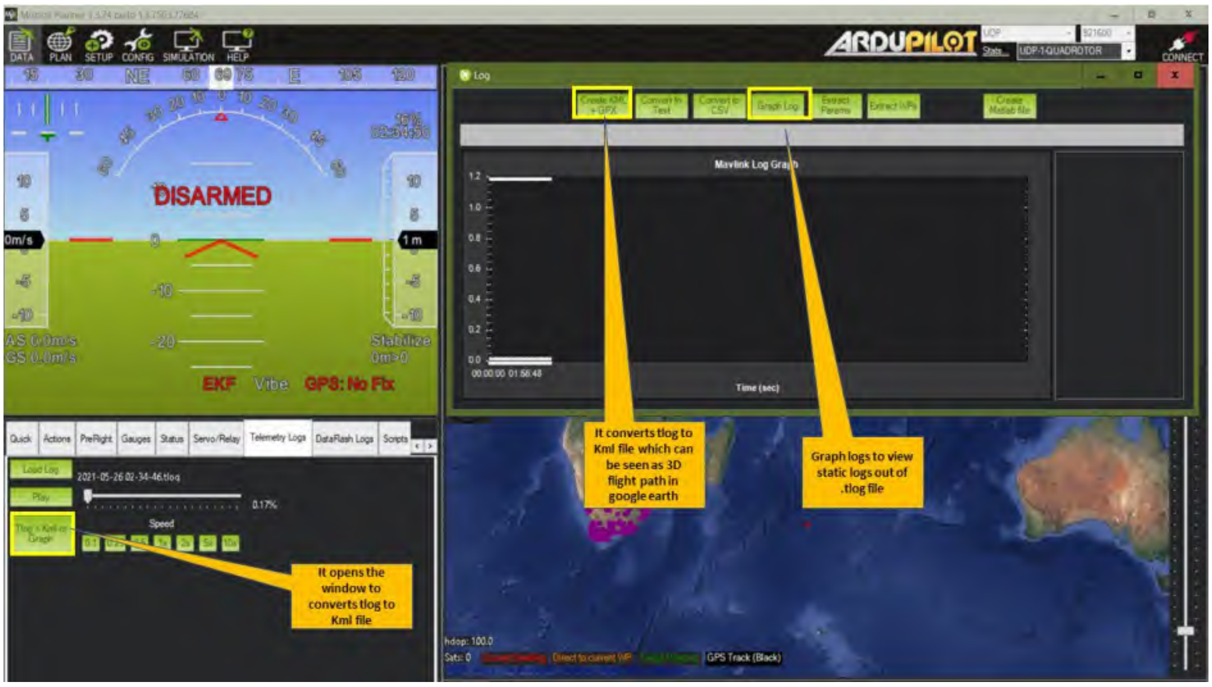


Figure 10.27: Procedure to customize Telemetry Logs

We can also convert .tlog to KML+GPX to view 3D flight plan in google earth. The converted file with .tlog.kml is in the same log sub folder. The graph log option can be used to view static graphs from .tlog file using the interactive window as shown below


Create KML +GPX
Graph Log
It converts tlog to Kml file which can be seen as 3D flight path in google earth
Graph logs to view static logs out of .tlog file
It opens the window to converts tlog to Kml file
Mavlink Log Graph
DISARMED
Figure 10.28: Converting tlogs to KML file or viewing static graphs

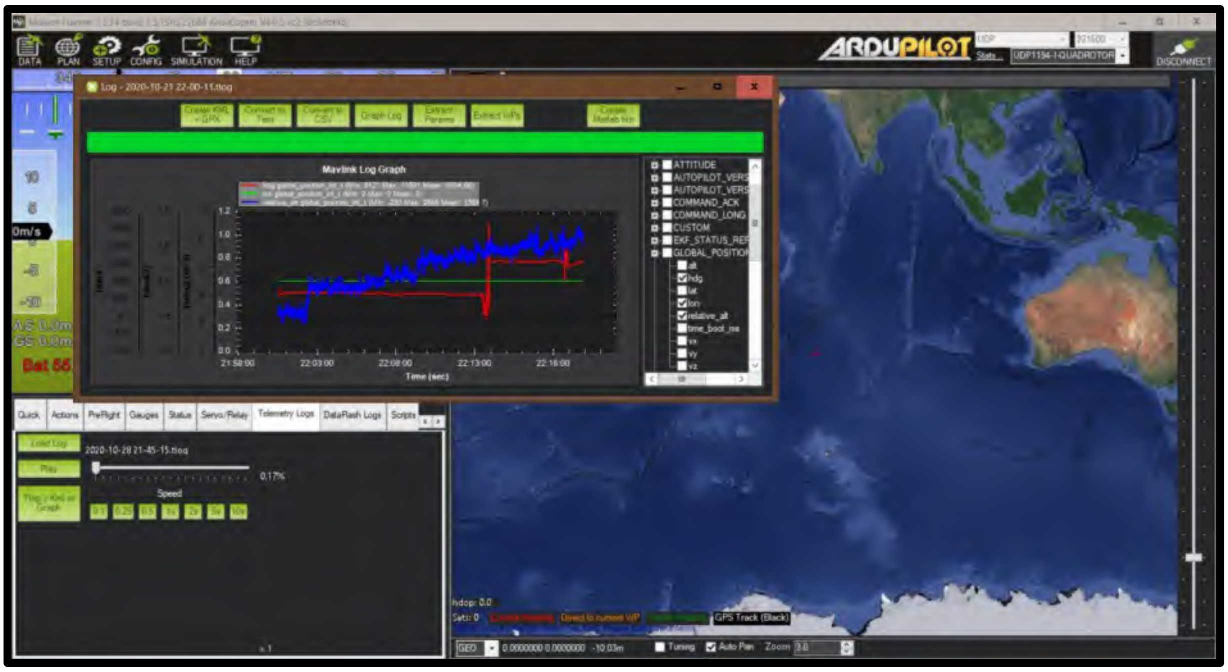
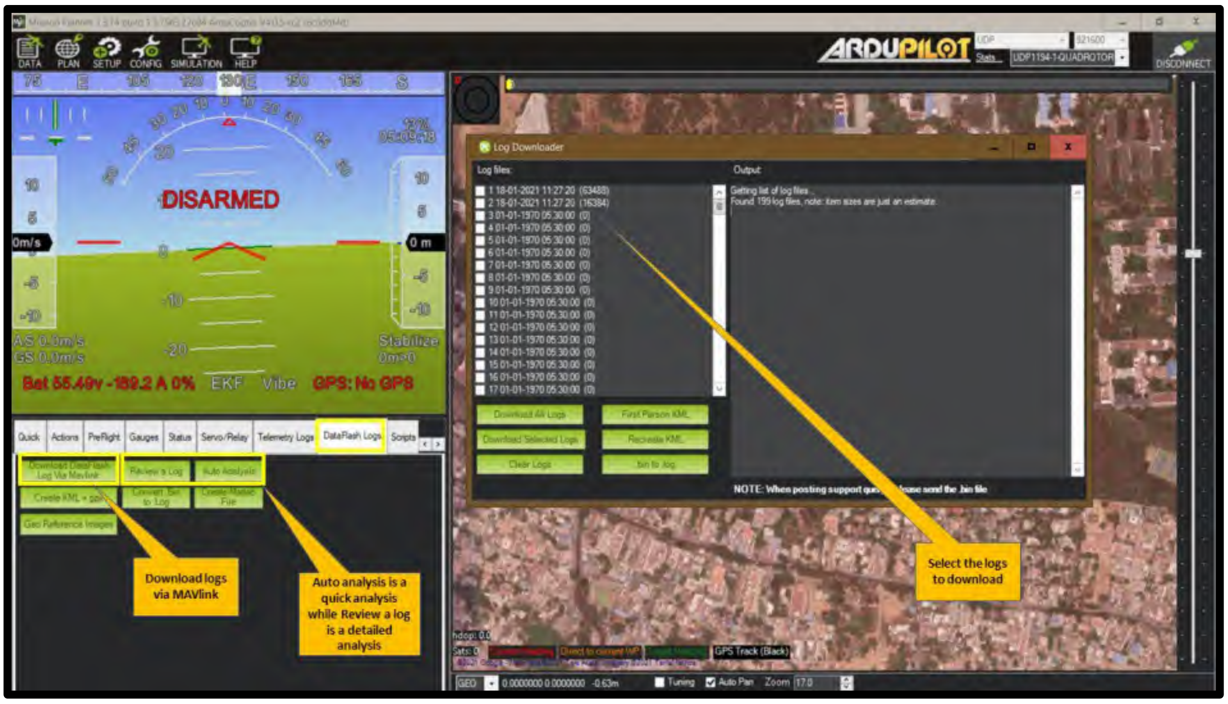


Figure 10.29: Window for viewing static graphs in telemetry logs window

Dataflash logs These logs are recorded on the Flight Controller i.e part of the ardupilot autopilot stack. They are downloaded using Mission Planner or going into the directory /var/lib/ardupilot incase of Beagle Bone Blue and can be downloaded directly. The log files are made in the format YYYY-MM-DD hh-mm-ss.bin and can be accessed from

Document > Mission Planner > log > frametype just like tlogs as seen above.



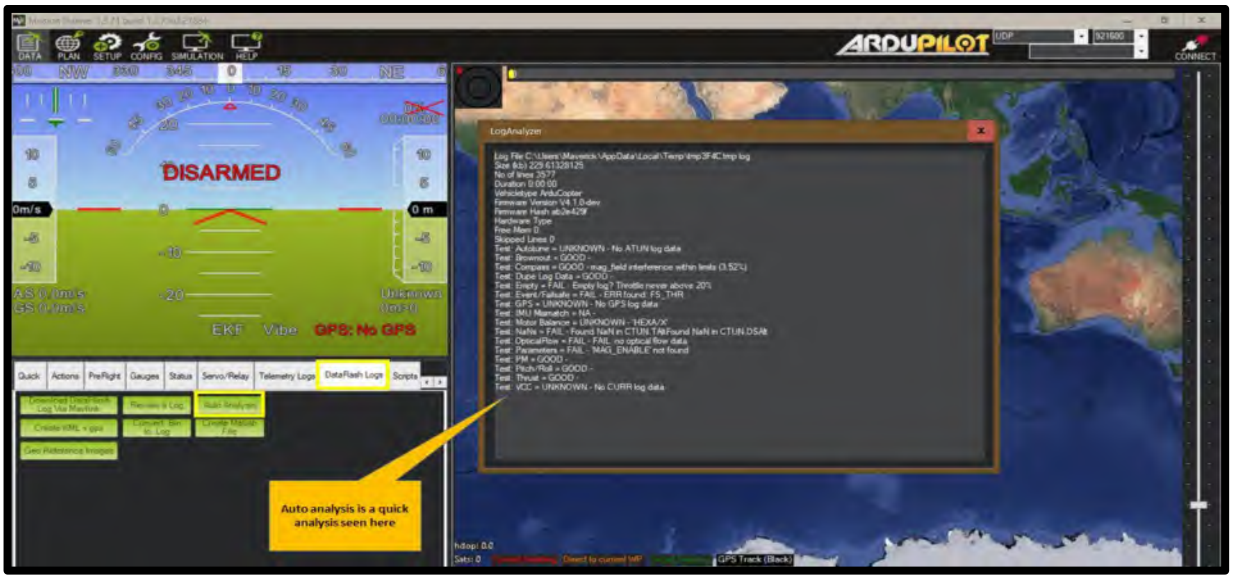
```
1 18-01-2021 11:27:20 (63488)
2 18-01-2021 11:27:20 (16384)
3 01-01-1970 05:30:00 (0)
4 01-01-1970 05:30:00 (0)
5 01-01-1970 05:30:00 (0)
6 01-01-1970 05:30:00 (0)
7 01-01-1970 05:30:00 (0)
8 01-01-1970 05:30:00 (0)
9 01-01-1970 05:30:00 (0)
10 01-01-1970 05:30:00 (0)
11 01-01-1970 05:30:00 (0)
12 01-01-1970 05:30:00 (0)
13 01-01-1970 05:30:00 (0)
14 01-01-1970 05:30:00 (0)
15 01-01-1970 05:30:00 (0)
16 01-01-1970 05:30:00 (0)
17 01-01-1970 05:30:00 (0)
```

```
Getting list of log files...
Found 199 log files, note: item sizes are just an estimate.
```

NOTE: When posting support qu[illegible]lease send the .bin file

Figure 10.30: Accessing data flash logs through Mission Planner

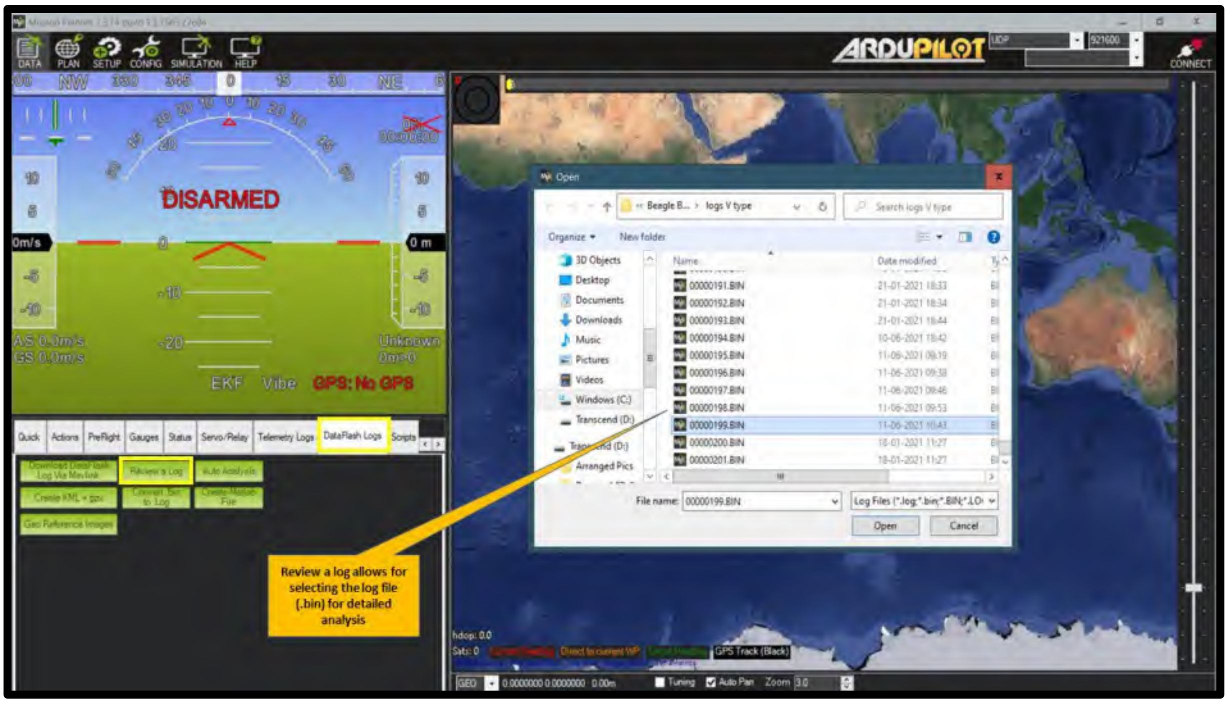
The data flash logs have two options namely auto analysis and review a log. The auto analysis is a quick log summary view and does not contain much details.



```
Log File C:\Users\Maverick\AppData\Local\Temp\tmp3F4C.tmp.log
Size (kb) 229.61328125
No of lines 3577
Duration 0:00:00
Vehicletype ArduCopter
Firmware Version V4.1.0-dev
Firmware Hash ab2e429f
Hardware Type
Free Mem 0
Skipped Lines 0
Test: Autotune = UNKNOWN - No ATUN log data
Test: Brownout = GOOD -
Test: Compass = GOOD - mag_field interference within limits (3.52%)
Test: Dupe Log Data = GOOD -
Test: Empty = FAIL - Empty log? Throttle never above 20%
Test: Event/Failsafe = FAIL - ERR found: FS_THR
Test: GPS = UNKNOWN - No GPS log data
Test: IMU Mismatch = NA -
Test: Motor Balance = UNKNOWN - 'HEXA/X'
Test: NaNs = FAIL - Found NaN in CTUN.TAltFound NaN in CTUN.DSAlt
Test: OpticalFlow = FAIL - FAIL: no optical flow data
Test: Parameters = FAIL - 'MAG_ENABLE' not found
Test: PM = GOOD -
Test: Pitch/Roll = GOOD -
Test: Thrust = GOOD -
Test: VCC = UNKNOWN - No CURR log data
```

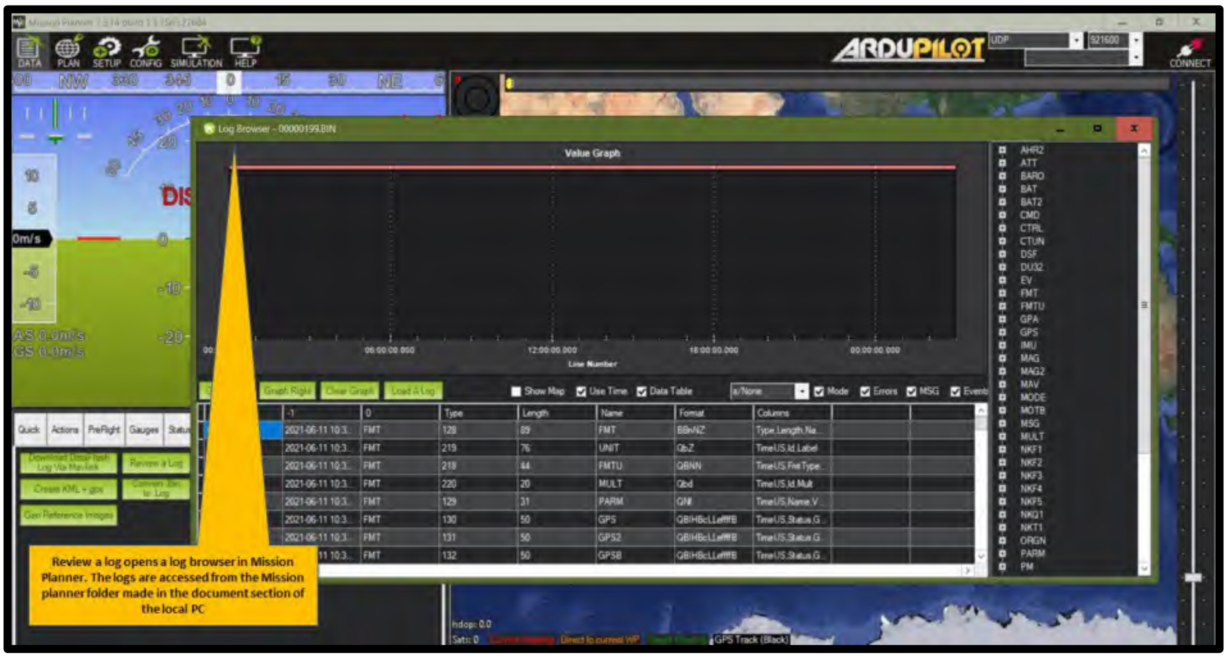
Figure 10.31: Auto analysis of data flash log i.e .bin file

Review a log is a more detailed analysis for in-depth understanding of functioning of components as logged by Ardupilot autopilot application. For the purpose of my thesis I will use review a log as it provides me with a log browser where I can review each parameter of the log file as static graphs to better understand the working of drone software and hardware during flight operations. Unlike tlogs which are moving logs like a live review, the data flash logs are static graphs.



Name	Date modified
00000191.BIN	21-01-2021 18:33
00000192.BIN	21-01-2021 18:34
00000193.BIN	21-01-2021 18:44
00000194.BIN	10-06-2021 18:42
00000195.BIN	11-06-2021 09:19
00000196.BIN	11-06-2021 09:38
00000197.BIN	11-06-2021 09:46
00000198.BIN	11-06-2021 09:53
00000199.BIN	11-06-2021 10:43
00000200.BIN	18-01-2021 11:27
00000201.BIN	18-01-2021 11:27

Figure 10.32: Review a log option of data flash log i.e .bin file



-1	0	Type	Length	Name	Format	Columns
2021-06-11 10:3...	FMT	128	89	FMT	BBnNZ	Type,Length,Na...
2021-06-11 10:3...	FMT	219	76	UNIT	QbZ	TimeUS,Id,Label
2021-06-11 10:3...	FMT	218	44	FMTU	QBNN	TimeUS,FmtType...
2021-06-11 10:3...	FMT	220	20	MULT	Qbd	TimeUS,Id,Mult
2021-06-11 10:3...	FMT	129	31	PARM	QNf	TimeUS,Name,V...
2021-06-11 10:3...	FMT	130	50	GPS	QBIHBcLLefffB	TimeUS,Status,G...
2021-06-11 10:3...	FMT	131	50	GPS2	QBIHBcLLefffB	TimeUS,Status,G...
2021-06-11 10:3...	FMT	132	50	GPSB	QBIHBcLLefffB	TimeUS,Status,G...

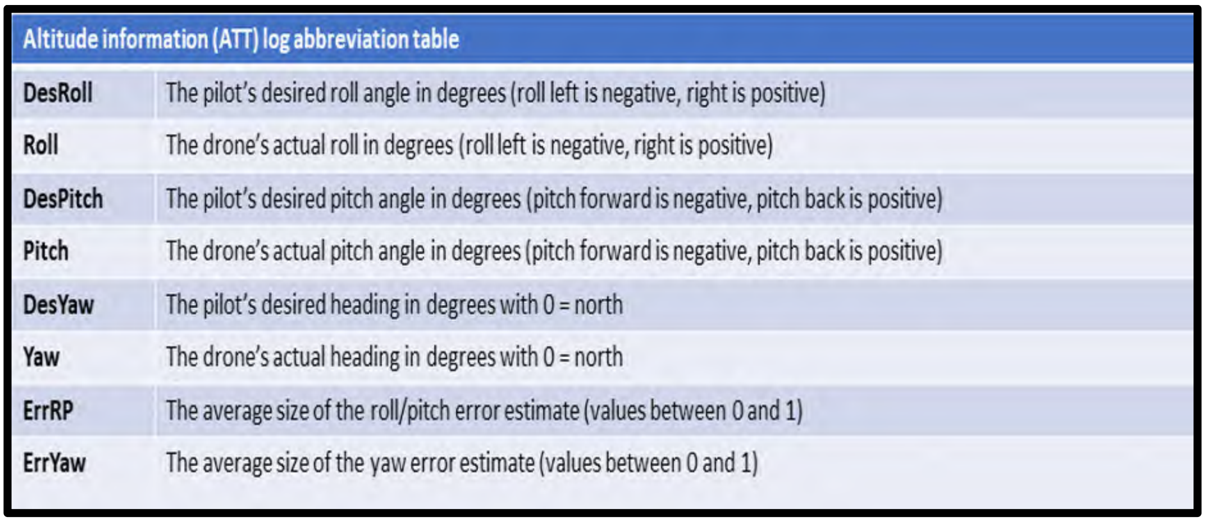
Figure 10.33: Log browser to access data flash log in review a log option i.e .bin

Understanding data flash logs using review a log option

In order to better understand the flight logs we need to be clear on the end aim of the analysis. Irrespective of the fact whether we use tlogs or data flash logs (review a log) option, the aim is to analyses the working of drone from key failure or error perspective to minimize failure of flight operations. Also, the nomenclature of log parameters more or less remain the same which means the same knowledge of data flash logs can help us analyze tlogs too. Its entirely on the drone operator or pilots choice to adopt which method for log analysis. The subsequent analysis of logs in the thesis is relevant for tlogs too.

There are six major types of errors to look for while analyzing logs. The analysis also helps us to optimize the design, improve configuration and do advanced tuning for improved flight operations. I will review actual logs in my thesis to explain the analysis and possible solutions during the course of my thesis.

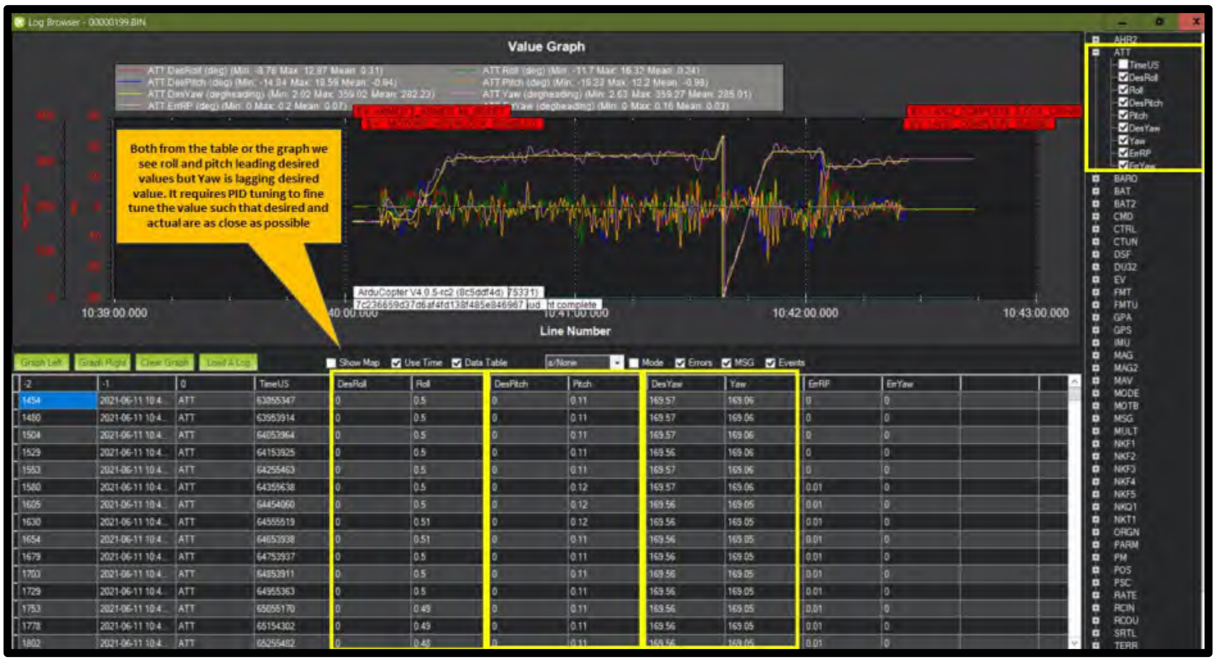
Mechanical Failures The most common error that can occur to any drone is due to mechanical fault such as motor or ESC failure, propeller damage etc resulting in sudden divergence in the desired roll/pitch/yaw to the actual roll/pitch/yaw which we will analyze through the log browser of data flash logs. This deviation from desired values can be seen in the Altitude information seen as ATT in log analysis. The details of this log settings are as below:



Altitude information (ATT) log abbreviation table	
DesRoll	The pilot's desired roll angle in degrees (roll left is negative, right is positive)
Roll	The drone's actual roll in degrees (roll left is negative, right is positive)
DesPitch	The pilot's desired pitch angle in degrees (pitch forward is negative, pitch back is positive)
Pitch	The drone's actual pitch angle in degrees (pitch forward is negative, pitch back is positive)
DesYaw	The pilot's desired heading in degrees with 0 = north
Yaw	The drone's actual heading in degrees with 0 = north
ErrRP	The average size of the roll/pitch error estimate (values between 0 and 1)
ErrYaw	The average size of the yaw error estimate (values between 0 and 1)

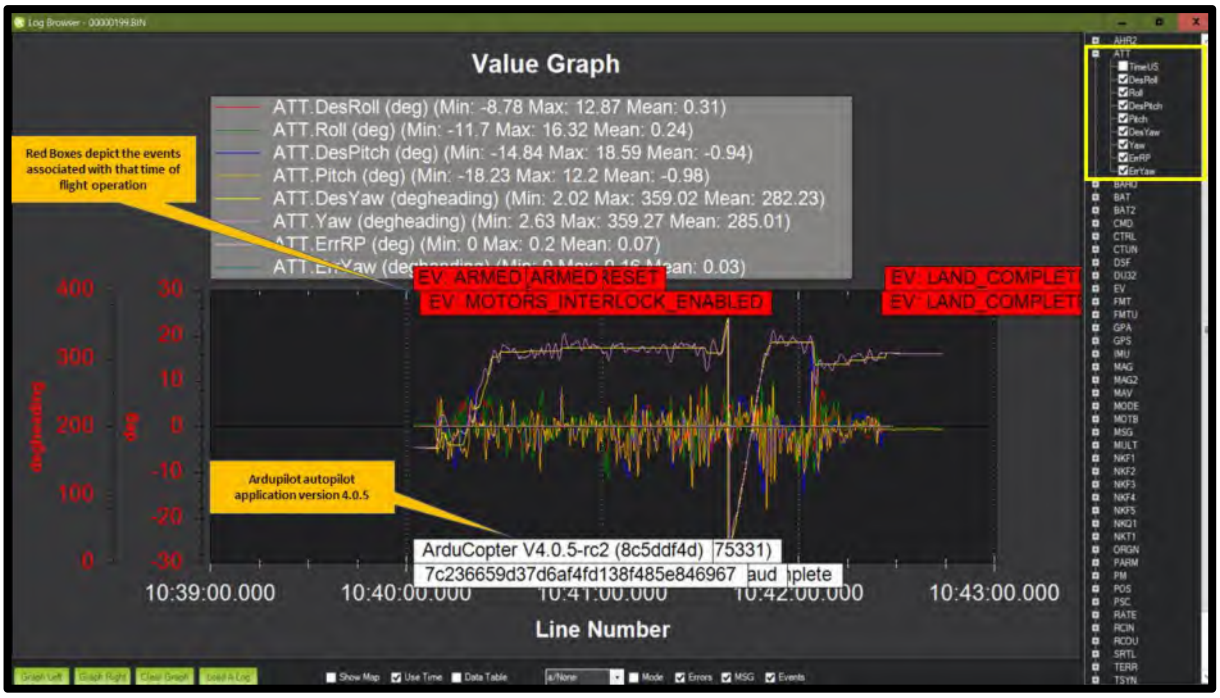
Figure 10.34: Altitude information ATT log abbreviation table

The log review of altitude information, ATT from log file of V type drone flown at Chemplast stadium of IITM. The x axis depicts the time of flight while the y axis is degrees for roll and pitch while degree heading for yaw. We see in the logs that both from the table or the pictorial graph we see roll and pitch leading desired values but Yaw is lagging desired value. It requires PID tuning to fine tune the value such that desired and actual are as close as possible. The PID tuning is discussed in the subsequent part of this chapter.


Log Browser - 00000199.BIN
Value Graph
Both from the table or the graph we see roll and pitch leading desired values but Yaw is lagging desired value. It requires PID tuning to fine tune the value such that desired and actual are as close as possible
ArduCopter V4.0.5-rc2 (8c5ddf4d) 75331)
7c236659d37d6af4fd138f485e846967
10:39:00.000
10:42:00.000
10:43:00.000
Line Number
Graph Left
Graph Right
Clear Graph
Load A Log
Show Map
Use Time
Data Table
Mode
Errors
MSG
Events

-2	-1	0	TimeUS	DesRoll	Roll	DesPitch	Pitch	DesYaw	Yaw	ErrRP	ErrYaw
1454	2021-06-11 10:4...	ATT	63855347	0	0.5	0	0.11	169.57	169.06	0	0
1480	2021-06-11 10:4...	ATT	63953914	0	0.5	0	0.11	169.57	169.06	0	0
1504	2021-06-11 10:4...	ATT	64053964	0	0.5	0	0.11	169.57	169.06	0	0
1529	2021-06-11 10:4...	ATT	64153925	0	0.5	0	0.11	169.56	169.06	0	0
1553	2021-06-11 10:4...	ATT	64255463	0	0.5	0	0.11	169.57	169.06	0	0
1580	2021-06-11 10:4...	ATT	64355638	0	0.5	0	0.12	169.57	169.06	0.01	0
1605	2021-06-11 10:4...	ATT	64454060	0	0.5	0	0.12	169.56	169.05	0.01	0
1630	2021-06-11 10:4...	ATT	64555519	0	0.51	0	0.12	169.56	169.05	0.01	0
1654	2021-06-11 10:4...	ATT	64653938	0	0.51	0	0.11	169.56	169.05	0.01	0
1679	2021-06-11 10:4...	ATT	64753937	0	0.5	0	0.11	169.56	169.05	0.01	0
1703	2021-06-11 10:4...	ATT	64853911	0	0.5	0	0.11	169.56	169.05	0.01	0
1729	2021-06-11 10:4...	ATT	64955363	0	0.5	0	0.11	169.56	169.05	0.01	0
1753	2021-06-11 10:4...	ATT	65055170	0	0.49	0	0.11	169.56	169.05	0.01	0
1778	2021-06-11 10:4...	ATT	65154302	0	0.49	0	0.11	169.56	169.05	0.01	0
1802	2021-06-11 10:4...	ATT	65255482	0	0.48	0	0.11	169.56	169.05	0.01	0

Figure 10.35: Altitude information ATT log view in log browser


Log Browser - 00000199.BIN
Value Graph
Red Boxes depict the events associated with that time of flight operation
EV: ARMED ARMED RESET
EV: MOTORS_INTERLOCK_ENABLED
EV: LAND_COMPLETE
Ardupilot autopilot application version 4.0.5
ArduCopter V4.0.5-rc2 (8c5ddf4d) 75331)
7c236659d37d6af4fd138f485e846967
degheading
deg
400
300
200
100
0
30
20
10
0
-10
-20
-30
10:39:00.000
10:40:00.000
10:41:00.000
10:42:00.000
10:43:00.000
Line Number

ATT.DesRoll (deg) (Min: -8.78 Max: 12.87 Mean: 0.31)
ATT.Roll (deg) (Min: -11.7 Max: 16.32 Mean: 0.24)
ATT.DesPitch (deg) (Min: -14.84 Max: 18.59 Mean: -0.94)
ATT.Pitch (deg) (Min: -18.23 Max: 12.2 Mean: -0.98)
ATT.DesYaw (degheading) (Min: 2.02 Max: 359.02 Mean: 282.23)
ATT.Yaw (degheading) (Min: 2.63 Max: 359.27 Mean: 285.01)
ATT.ErrRP (deg) (Min: 0 Max: 0.2 Mean: 0.07)
ATT.ErrYaw (degheading) (Min: 0 Max: 0.16 Mean: 0.03)

Figure 10.36: Altitude information ATT log view in log browser

Vibrations High vibrations results in drone's accelerometer based altitude and horizontal position estimates to drift far off from reality which leads to problems with altitude hold. This can result in the drone quickly shooting up in the sky in flight modes such as Loiter, PosHold, Auto, etc. Vibration levels below 30m/s/s are acceptable levels while above 30m/s/s may result in problems. The values above 60m/s/s will surely cause problems with flight modes position or altitude hold. Hence, this analysis is key to ensure safe flight. The real time vibrations can also be seen during flight while post flight vibrations graphs can be seen in VIBE in log analysis. The details of this log settings are **VibeX , VibeY and VibeZ** which denote measured standard deviation of the primary accelerometer's output in x, y and z axis respectively in m/s/s. **Clip0, Clip1** and **Clip2** are the values which increase each time one of the accelerometers reaches its maximum limit (16G)

The Realtime vibration measurements can be see from **Mission Planner** > **Vibe** Option

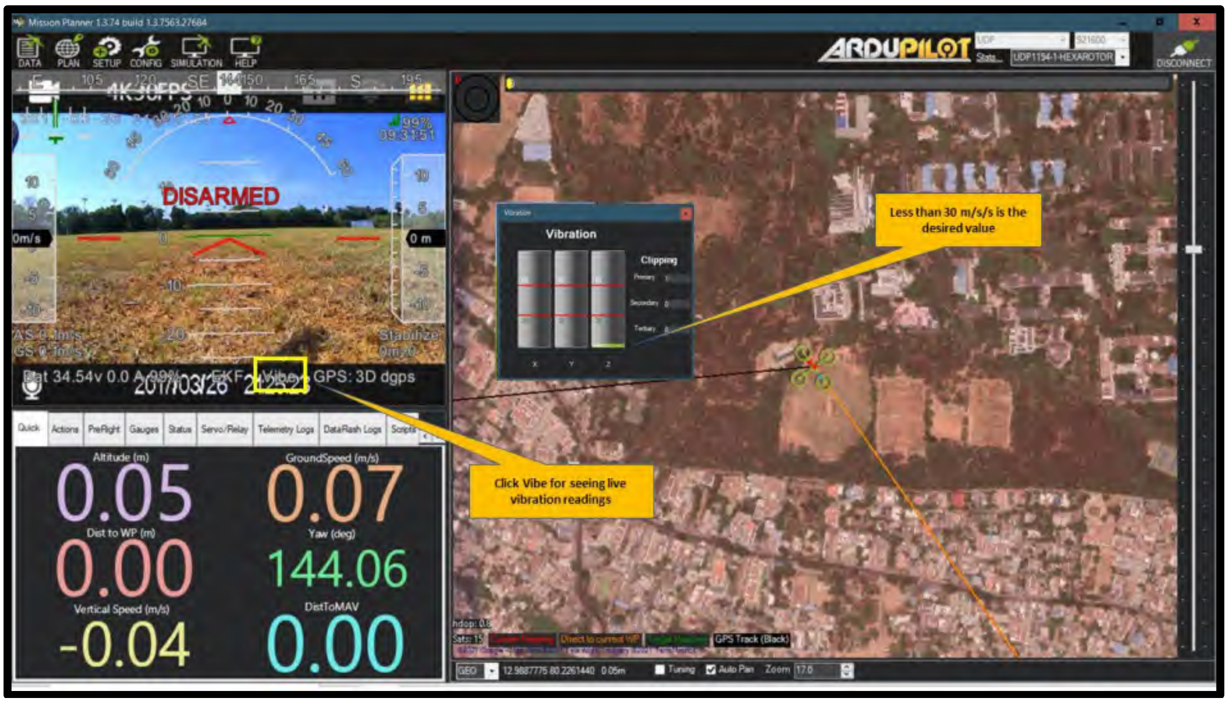
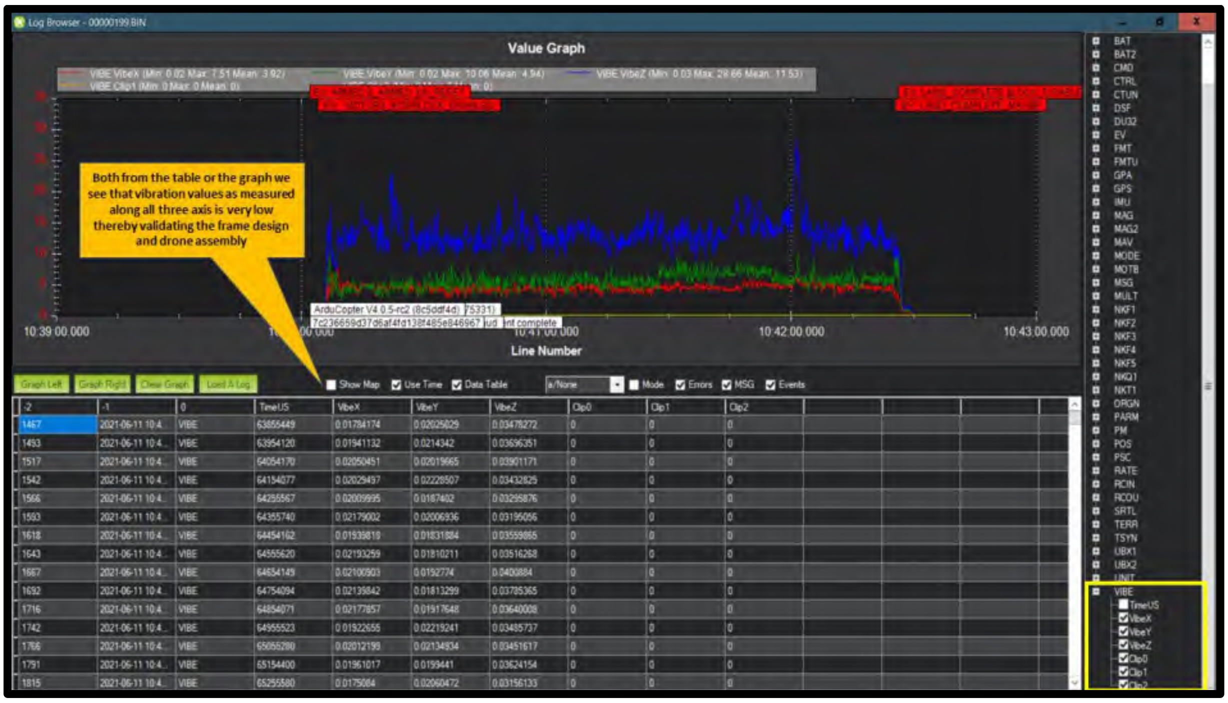


Figure 10.37: Live view of vibration measurements in Mission Planner

The log review of vibration information, VIBE from log file of V type drone flown at Chemplast stadium of IITM. The vibration measurement along x, y and z axis of drone is below 30 m/s/s which is the desired output. Both from the table or the pictorial graph we see that vibration values as measured along all three axis is very low thereby validating the frame design and drone assembly. The x axis in graph is the time of flight and y axis is vibration measured in m/s/s for each of the axis of drone frame (x,y and z)


Log Browser - 00000199.BIN
Value Graph
Both from the table or the graph we see that vibration values as measured along all three axis is very low thereby validating the frame design and drone assembly
ArduCopter V4.0.5-rc2 (8c5ddf4d) 75331)
7c236659d37d6af4fd138f485e846967
10:39:00.000
10:41:00.000
10:42:00.000
10:43:00.000
Line Number
Graph Left
Graph Right
Clear Graph
Load A Log
Show Map
Use Time
Data Table
a/None
Mode
Errors
MSG
Events
VIBE
TimeUS
VibeX
VibeY
VibeZ
Clip0
Clip1
Clip2

-2	-1	0	TimeUS	VibeX	VibeY	VibeZ	Clip0	Clip1	Clip2
1467	2021-06-11 10:4...	VIBE	63855449	0.01784174	0.02025029	0.03478272	0	0	0
1493	2021-06-11 10:4...	VIBE	63954120	0.01941132	0.0214342	0.03696351	0	0	0
1517	2021-06-11 10:4...	VIBE	64054170	0.02050451	0.02019665	0.03901171	0	0	0
1542	2021-06-11 10:4...	VIBE	64154077	0.02029497	0.02228507	0.03432825	0	0	0
1566	2021-06-11 10:4...	VIBE	64255567	0.02009995	0.0187402	0.03295876	0	0	0
1593	2021-06-11 10:4...	VIBE	64355740	0.02179002	0.02006936	0.03195056	0	0	0
1618	2021-06-11 10:4...	VIBE	64454162	0.01939819	0.01831884	0.03559865	0	0	0
1643	2021-06-11 10:4...	VIBE	64555620	0.02193259	0.01810211	0.03516268	0	0	0
1667	2021-06-11 10:4...	VIBE	64654149	0.02100903	0.0192774	0.0400884	0	0	0
1692	2021-06-11 10:4...	VIBE	64754094	0.02139842	0.01813299	0.03785365	0	0	0
1716	2021-06-11 10:4...	VIBE	64854071	0.02177857	0.01917648	0.03640008	0	0	0
1742	2021-06-11 10:4...	VIBE	64955523	0.01922655	0.02219241	0.03485737	0	0	0
1766	2021-06-11 10:4...	VIBE	65055280	0.02012199	0.02134934	0.03451617	0	0	0
1791	2021-06-11 10:4...	VIBE	65154400	0.01961017	0.0199441	0.03624154	0	0	0
1815	2021-06-11 10:4...	VIBE	65255580	0.0175084	0.02060472	0.03156133	0	0	0

Figure 10.38: Vibration information VIBE log view in log browser

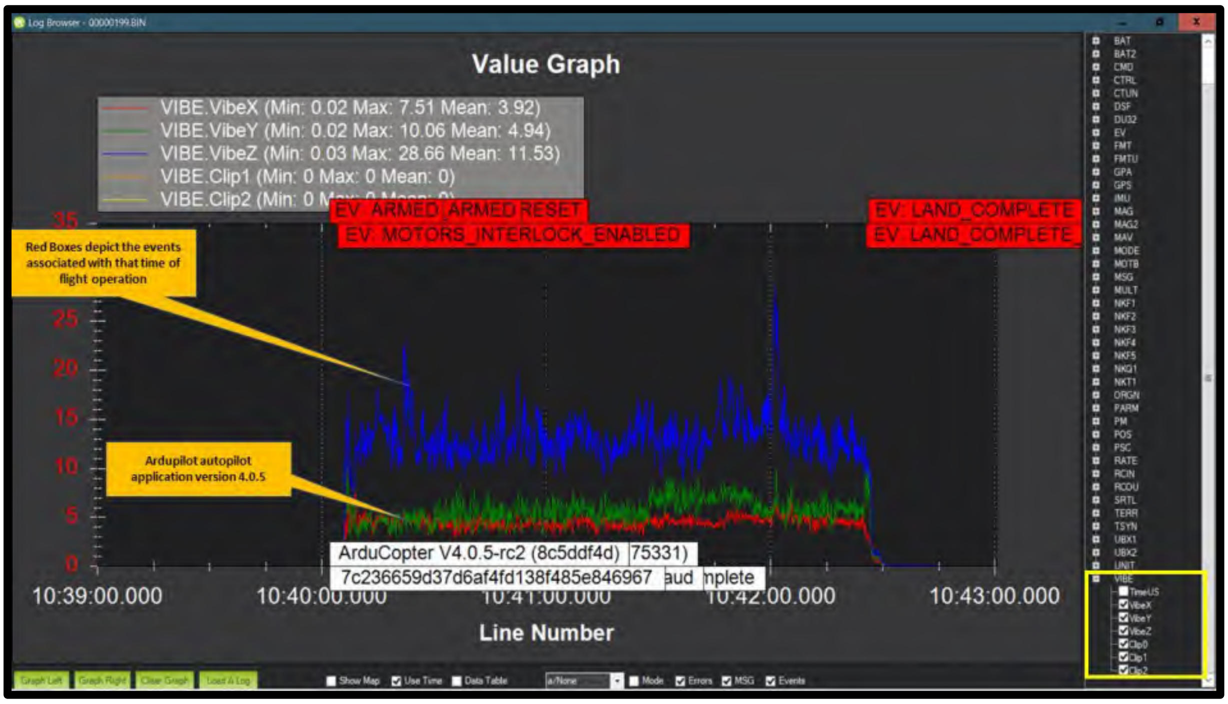
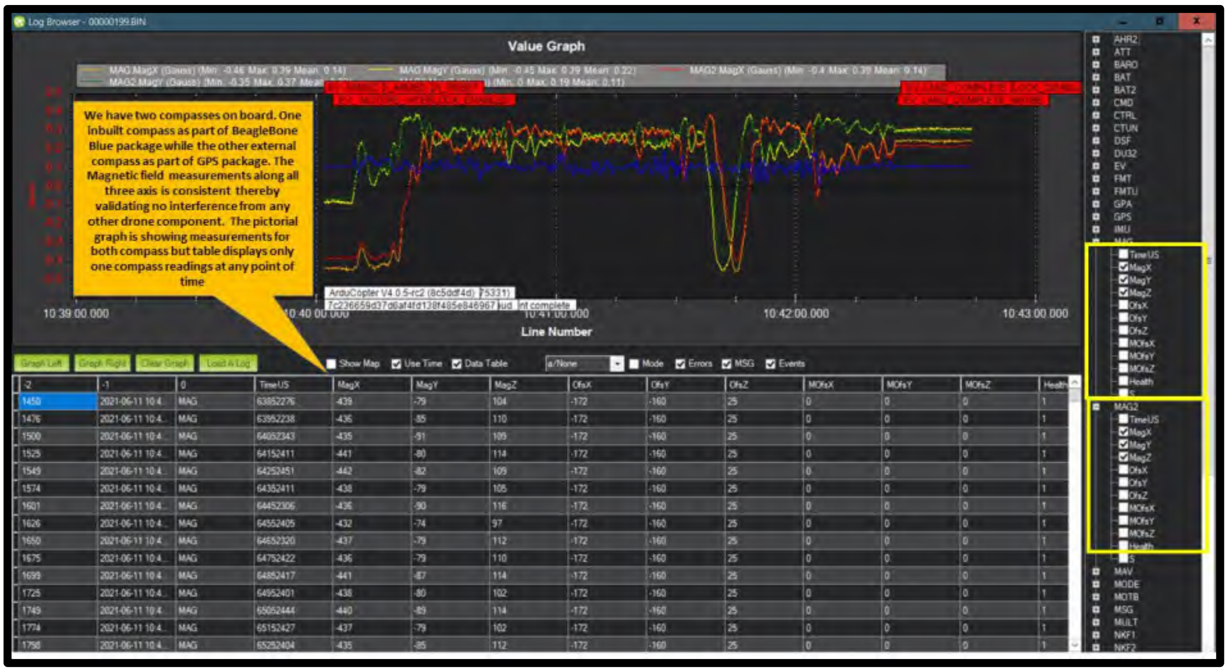

Log Browser - 00000199.BIN
Value Graph
VIBE.VibeX (Min: 0.02 Max: 7.51 Mean: 3.92)
VIBE.VibeY (Min: 0.02 Max: 10.06 Mean: 4.94)
VIBE.VibeZ (Min: 0.03 Max: 28.66 Mean: 11.53)
VIBE.Clip1 (Min: 0 Max: 0 Mean: 0)
VIBE.Clip2 (Min: 0 Max: 0 Mean: 0)
EV: ARMED ARMED RESET
EV: MOTORS_INTERLOCK_ENABLED
EV: LAND_COMPLETE
EV: LAND_COMPLETE
Red Boxes depict the events associated with that time of flight operation
Ardupilot autopilot application version 4.0.5
ArduCopter V4.0.5-rc2 (8c5ddf4d) 75331)
7c236659d37d6af4fd138f485e846967
35
25
20
15
10
5
0
10:39:00.000
10:40:00.000
10:41:00.000
10:42:00.000
10:43:00.000
Line Number
Graph Left
Graph Right
Clear Graph
Load A Log
Show Map
Use Time
Data Table
a/None
Mode
Errors
MSG
Events
VIBE
TimeUS
VibeX
VibeY
VibeZ
Clip0
Clip1
Clip2

Figure 10.39: Vibration information VIBE log view in log browser

Compass interference Interferences to compass can be caused on drone by power distribution board, motors, battery, ESCs and other electrical devices. These can disrupt compass heading resulting in circling or drone flying in completely wrong direction. It is a very important parameter to look for to ensure optimum design considerations often requiring to tune the frame design. Post flight this data can help understand the behavior of the drone.

The mag_field fluctuations are acceptable between 10-20% range when throttle is raised upto a max of 30%. Fluctuations above 60% is dangerous with increase in throttle. **MagX, MagY and MagZ** are raw magnetic field values for x, y and z axis respectively. **OfsX, OfsY and OfsZ** are raw magnetic offsets. They will only change if COMPASS_LEARN parameter is 1. **MOfsX, MOfsY and MOfsZ are** Compassmot compensation for throttle or current.

The log review of compass readings depicted by MAG and MAG2. We have two compasses on board. One inbuilt compass as part of BeagleBone Blue package while the other external compass as part of GPS package. The compass log information MAG and MAG2 from log file of V type drone flown at Chemplast stadium of IITM. The Magnetic field measurements along all three axis is consistent thereby validating no interference from any other drone component. The pictorial graph is showing measurements for both compass but table displays only one compass readings at any point of time. The x axis in graph is the time of flight and y axis is measured magnetic field in Gauss for each of the axis of drone frame (x,y and z) for both compasses onboard the drone.



Log Browser - 00000199.BIN
Value Graph
We have two compasses on board. One inbuilt compass as part of BeagleBone Blue package while the other external compass as part of GPS package. The Magnetic field measurements along all three axis is consistent thereby validating no interference from any other drone component. The pictorial graph is showing measurements for both compass but table displays only one compass readings at any point of time
ArduCopter V4.0.5-rc2 (8c5ddf4d) 75331)
10:39:00.000
10:40:00.000
10:41:00.000
10:42:00.000
10:43:00.000
Line Number
Graph Left
Graph Right
Clear Graph
Load A Log
Show Map
Use Time
Data Table
a/None
Mode
Errors
MSG
Events

-2	-1	0	TimeUS	MagX	MagY	MagZ	OfsX	OfsY	OfsZ	MOfsX	MOfsY	MOfsZ	Health
1450	2021-06-11 10:4...	MAG	63852276	-439	-79	104	-172	-160	25	0	0	0	1
1476	2021-06-11 10:4...	MAG	63952238	-436	-85	110	-172	-160	25	0	0	0	1
1500	2021-06-11 10:4...	MAG	64052343	-435	-91	109	-172	-160	25	0	0	0	1
1525	2021-06-11 10:4...	MAG	64152411	-441	-80	114	-172	-160	25	0	0	0	1
1549	2021-06-11 10:4...	MAG	64252451	-442	-82	109	-172	-160	25	0	0	0	1
1574	2021-06-11 10:4...	MAG	64352411	-438	-79	105	-172	-160	25	0	0	0	1
1601	2021-06-11 10:4...	MAG	64452306	-436	-90	116	-172	-160	25	0	0	0	1
1626	2021-06-11 10:4...	MAG	64552405	-432	-74	97	-172	-160	25	0	0	0	1
1650	2021-06-11 10:4...	MAG	64652320	-437	-79	112	-172	-160	25	0	0	0	1
1675	2021-06-11 10:4...	MAG	64752422	-436	-79	110	-172	-160	25	0	0	0	1
1699	2021-06-11 10:4...	MAG	64852417	-441	-87	114	-172	-160	25	0	0	0	1
1725	2021-06-11 10:4...	MAG	64952401	-438	-80	102	-172	-160	25	0	0	0	1
1749	2021-06-11 10:4...	MAG	65052444	-440	-89	114	-172	-160	25	0	0	0	1
1774	2021-06-11 10:4...	MAG	65152427	-437	-79	102	-172	-160	25	0	0	0	1
1798	2021-06-11 10:4...	MAG	65252404	-435	-85	112	-172	-160	25	0	0	0	1

Figure 10.40: Magnetic fluctuation information as MAG and MAG2 logs view in log browser

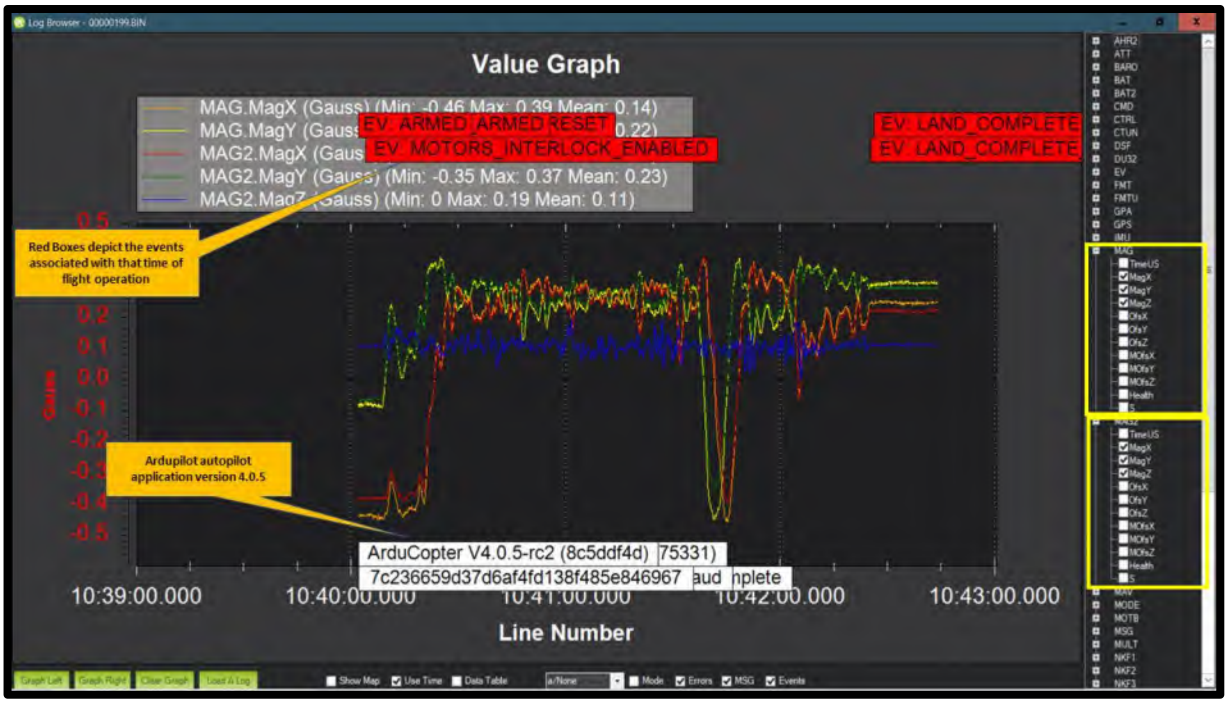
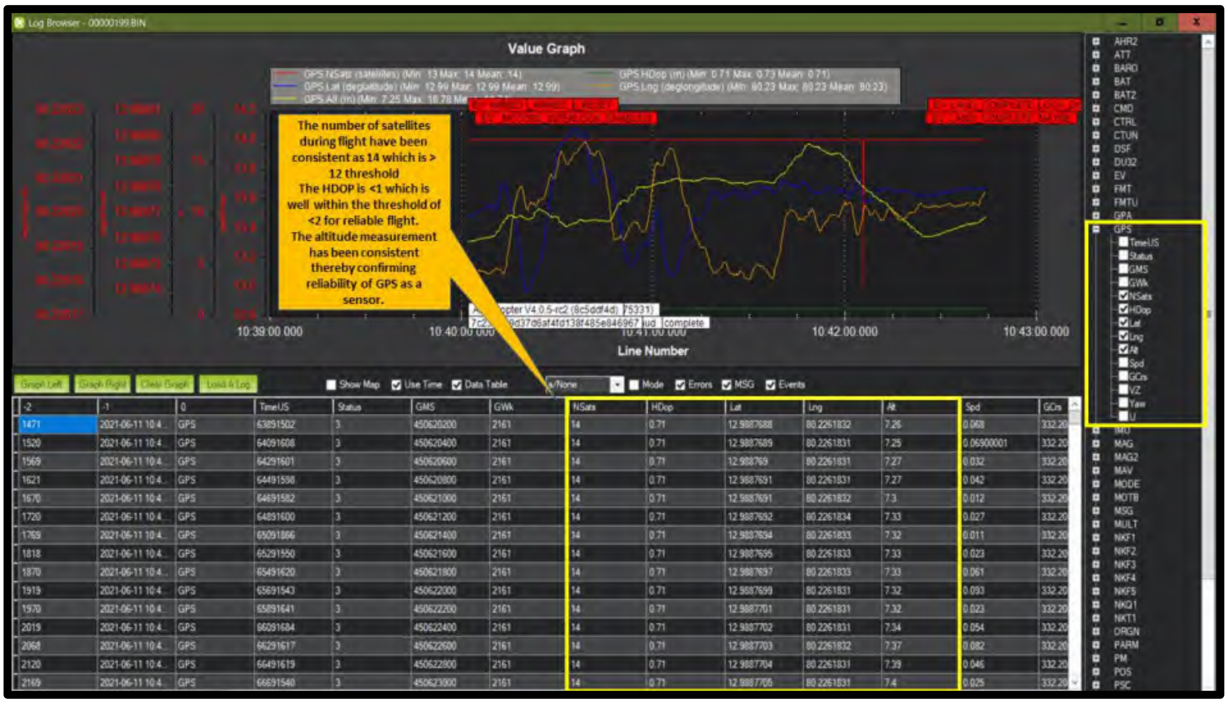


Figure 10.41: Magnetic fluctuation information as MAG and MAG2 logs view in log browser

GPS glitches The GPS glitch is an important parameter to evaluate during logs analysis especially when flying in flight modes requiring GPS input to make decision such as Loiter, RTL, Auto, etc. Position errors as a result of erroneous GPS readings can lead to unreliable or aggressive behavior of the drone. Two key fields to look for in GPS logs are Horizontal dilution of precision (HDop) and Number of satellites locked on to (NSats) values. Hdop < 1.5 is very good are very good while above 2.0 indicates a problem. The number of satellites < 12 is undesirable. The pilot must wait for these values to fall in range before taking off. A considerable change in these two values results in GPS position change. The GPS values during flight can be seen as GPS in log analysis. The details of this log settings are as below:

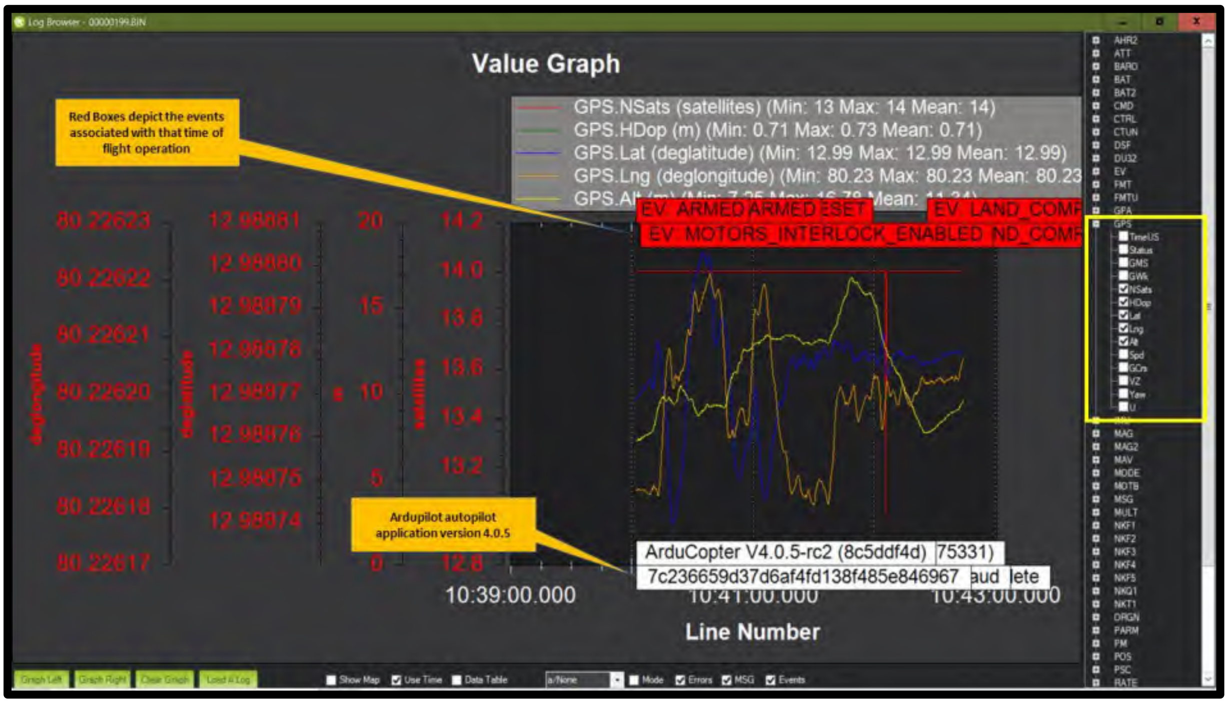
The log review of GPS readings depicted by GPS in log browser. The number of satellites during flight have been consistent as 14 which is > 12 (threshold value as discussed earlier). The HDOP is <1 which is well within the threshold of <2 for reliable flight. The altitude measurement has been consistent thereby confirming reliability of GPS as a sensor. The readings validate that there is no interference to GPS or GPS glitch thereby validating both the hardware status and the location of GPS on the drone has no interference from motors. The pictorial graph is showing measurements of important GPS parameters such as number of satellites, HDOP value, altitude value and latitude and longitude values. for both compass but table displays only one compass readings at any point of time. The x axis in graph is the time of flight and y axis is number for satellite count, decimal number for HDOP value, and latitude and longitude in degrees.


Log Browser - 00000199.BIN
Value Graph
GPS.NSats (satellites) (Min: 13 Max: 14 Mean: 14)
GPS.HDop (m) (Min: 0.71 Max: 0.73 Mean: 0.71)
GPS.Lat (deglatitude) (Min: 12.99 Max: 12.99 Mean: 12.99)
GPS.Lng (deglongitude) (Min: 80.23 Max: 80.23 Mean: 80.23)
GPS.Alt (m) (Min: 7.25 Max: 16.78 Mean
10:39:00.000
10:40:00.000
10:41:00.000
10:42:00.000
10:43:00.000
Line Number
Graph Left
Graph Right
Clear Graph
Load A Log
Show Map
Use Time
Data Table
Mode
Errors
MSG
Events

The number of satellites during flight have been consistent as 14 which is > 12 threshold The HDOP is <1 which is well within the threshold of <2 for reliable flight. The altitude measurement has been consistent thereby confirming reliability of GPS as a sensor.

-2	-1	0	TimeUS	Status	GMS	GWk	NSats	HDop	Lat	Lng	Alt	Spd	GCrs
1471	2021-06-11 10:4...	GPS	63891502	3	450620200	2161	14	0.71	12.9887688	80.2261832	7.26	0.068	332.20
1520	2021-06-11 10:4...	GPS	64091608	3	450620400	2161	14	0.71	12.9887689	80.2261831	7.25	0.06900001	332.20
1569	2021-06-11 10:4...	GPS	64291601	3	450620600	2161	14	0.71	12.988769	80.2261831	7.27	0.032	332.20
1621	2021-06-11 10:4...	GPS	64491598	3	450620800	2161	14	0.71	12.9887691	80.2261831	7.27	0.042	332.20
1670	2021-06-11 10:4...	GPS	64691582	3	450621000	2161	14	0.71	12.9887691	80.2261832	7.3	0.012	332.20
1720	2021-06-11 10:4...	GPS	64891600	3	450621200	2161	14	0.71	12.9887692	80.2261834	7.33	0.027	332.20
1769	2021-06-11 10:4...	GPS	65091866	3	450621400	2161	14	0.71	12.9887694	80.2261833	7.32	0.011	332.20
1818	2021-06-11 10:4...	GPS	65291550	3	450621600	2161	14	0.71	12.9887695	80.2261833	7.33	0.023	332.20
1870	2021-06-11 10:4...	GPS	65491620	3	450621800	2161	14	0.71	12.9887697	80.2261833	7.33	0.061	332.20
1919	2021-06-11 10:4...	GPS	65691543	3	450622000	2161	14	0.71	12.9887699	80.2261831	7.32	0.093	332.20
1970	2021-06-11 10:4...	GPS	65891641	3	450622200	2161	14	0.71	12.9887701	80.2261831	7.32	0.023	332.20
2019	2021-06-11 10:4...	GPS	66091684	3	450622400	2161	14	0.71	12.9887702	80.2261831	7.34	0.054	332.20
2068	2021-06-11 10:4...	GPS	66291617	3	450622600	2161	14	0.71	12.9887703	80.2261832	7.37	0.082	332.20
2120	2021-06-11 10:4...	GPS	66491619	3	450622800	2161	14	0.71	12.9887704	80.2261831	7.39	0.046	332.20
2169	2021-06-11 10:4...	GPS	66691540	3	450623000	2161	14	0.71	12.9887705	80.2261831	7.4	0.025	332.20

Figure 10.42: GPS information as GPS log view in log browser

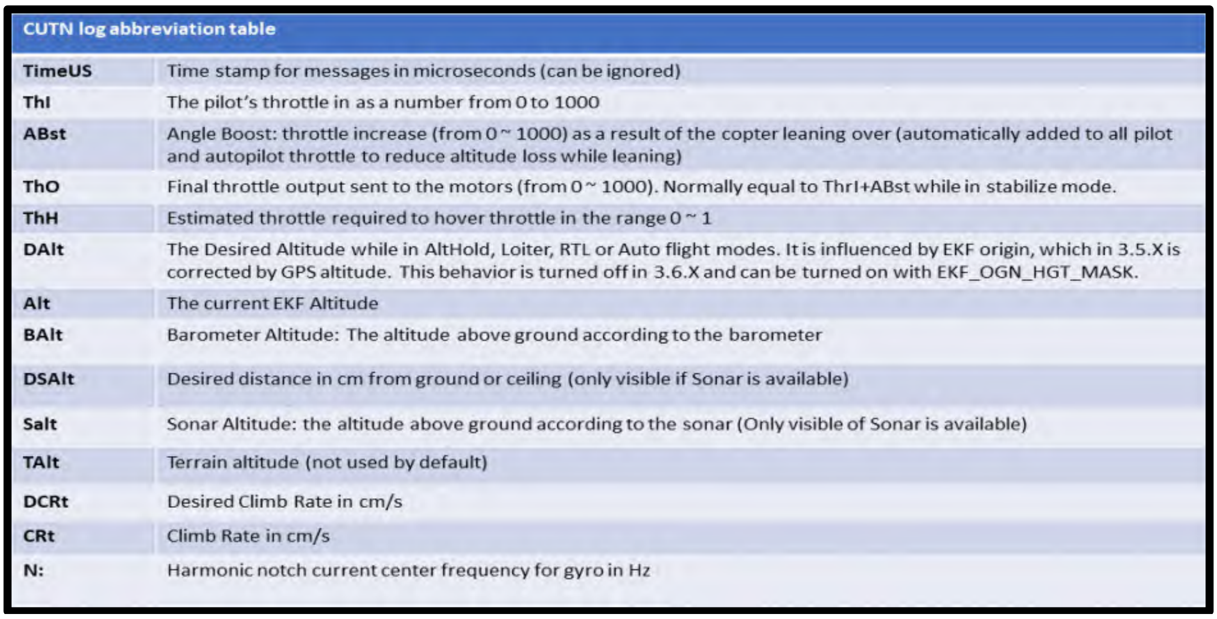

Log Browser - 00000199.BIN
Value Graph
GPS.NSats (satellites) (Min: 13 Max: 14 Mean: 14)
GPS.HDop (m) (Min: 0.71 Max: 0.73 Mean: 0.71)
GPS.Lat (deglatitude) (Min: 12.99 Max: 12.99 Mean: 12.99)
GPS.Lng (deglongitude) (Min: 80.23 Max: 80.23 Mean: 80.23
GPS.Alt (m) (Min: 7.25 Max: 16.78 Mean: 11.24)
EV_ARMED ARMED ESET
EV_MOTORS_INTERLOCK_ENABLED ND_COMP
EV_LAND_COMP
ArduCopter V4.0.5-rc2 (8c5ddf4d) 75331)
7c236659d37d6af4fd138f485e846967 aud lete
deglongitude
deglatitude
m
satellites
10:39:00.000
10:41:00.000
10:43:00.000
Line Number

Red Boxes depict the events associated with that time of flight operation

Ardupilot autopilot application version 4.0.5

Figure 10.43: GPS information as GPS log view in log browser

Altitude variations EKF errors The power to the all drone components is routed from the battery through the power module which also measures the voltage and current consumption if configured and calibrated. We need to see graphs of GPS, barometer altitude (BAlt) or altitude above home (CTUN Alt). The CTUN log gives us lot many details including throttle percentages ThH and ThO. It also gives us barometer altitude, desired altitude, terrain altitude and the climbed rates abbreviations are as below



CUTN log abbreviation table	
TimeUS	Time stamp for messages in microseconds (can be ignored)
ThI	The pilot's throttle in as a number from 0 to 1000
ABst	Angle Boost: throttle increase (from 0 ~ 1000) as a result of the copter leaning over (automatically added to all pilot and autopilot throttle to reduce altitude loss while leaning)
ThO	Final throttle output sent to the motors (from 0 ~ 1000). Normally equal to ThrI+ABst while in stabilize mode.
ThH	Estimated throttle required to hover throttle in the range 0 ~ 1
DAlt	The Desired Altitude while in AltHold, Loiter, RTL or Auto flight modes. It is influenced by EKF origin, which in 3.5.X is corrected by GPS altitude. This behavior is turned off in 3.6.X and can be turned on with EKF_OGN_HGT_MASK.
Alt	The current EKF Altitude
BAlt	Barometer Altitude: The altitude above ground according to the barometer
DSAlt	Desired distance in cm from ground or ceiling (only visible if Sonar is available)
Salt	Sonar Altitude: the altitude above ground according to the sonar (Only visible of Sonar is available)
TAlt	Terrain altitude (not used by default)
DCRt	Desired Climb Rate in cm/s
CRt	Climb Rate in cm/s
N:	Harmonic notch current center frequency for gyro in Hz

Figure 10.44: CUTN log abbreviations

Any sudden loss of altitude or abrupt reading co-related with GPS altitude readings can point to sudden power failure. The realtime EKF readings can be seen here:

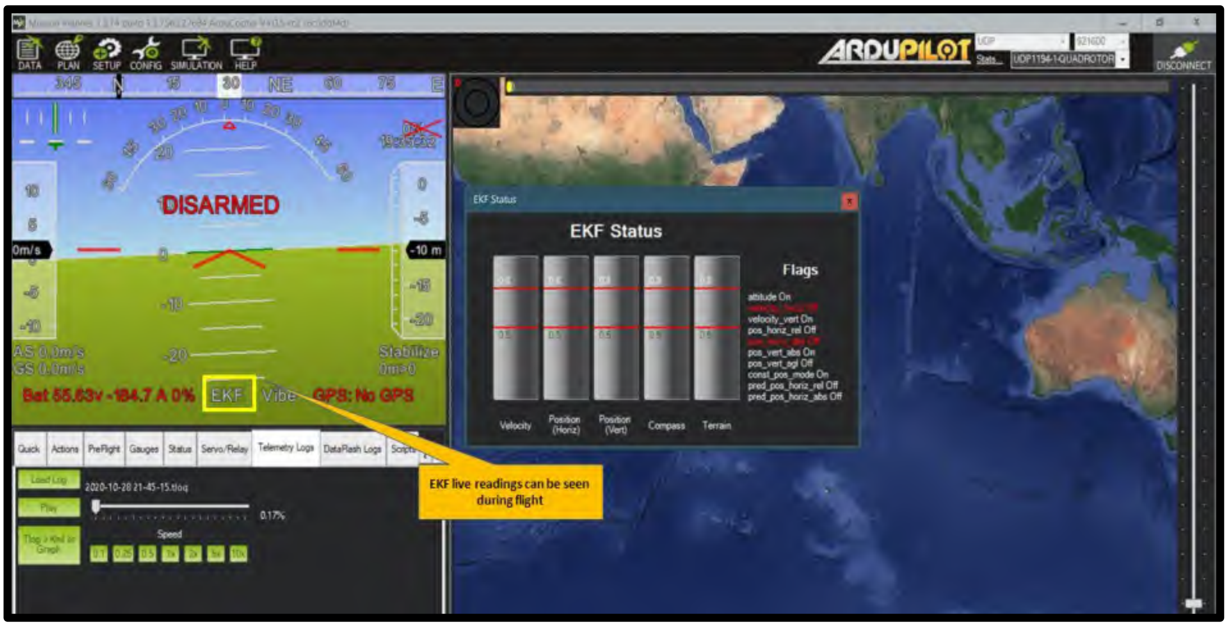


Figure 10.45: Live view of EKF measurements in Mission Planner

We also need to understand how Ardupilot takes altitude reading both from sensor and its own estimation readings. The altitude above (mean) Sea Level (ASL) is the altitude of the drone with respect to the mean sea level (MSL) of the world. Altitude above Ground Level (AGL) the altitude of the drone above its rest position. Relative altitude is the one measured above HOME/ORIGIN position and is what is displayed in the ground station. OSD as the vehicle's altitude. Terrain altitude (ALT) is the height above sea level (asl) of a terrain position. This may or may not any natural or man-made additions to the terrain's ground altitude. Estimated altitude (ALT) is the autopilot's EKF estimation of vehicle Relative altitude above ORIGIN. This measurement is used internally by the drone's Altitude Controller to maintain or obtain the Target ALT in flight modes such as ALTHOLD, LOITER/GUIDED etc where altitude is controlled. Target altitude (ALT) is the desired altitude in the above mentioned flight modes where altitude is maintained.

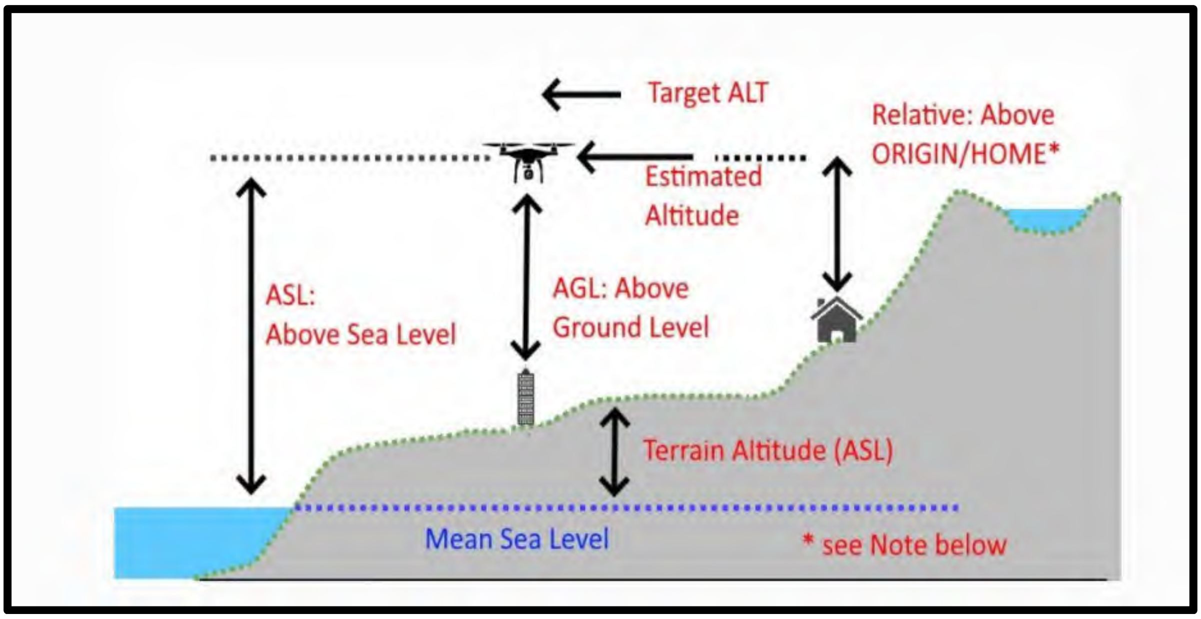


Figure 10.46: Understanding altitude information in Ardupilot autopilot stack [3]

The EKF takes the IMU, GPS, and BARO sensor inputs and integrates them to provide this Estimated altitude (ALT). The EKF subsystem is also responsible for estimating attitude, position and velocity of the drone which is utilized for navigation and control systems. The estimated altitude is then fed to the vehicle's altitude control system. It attempts to reach the Target ALT in flight modes where altitude is maintained as mentioned before. The altitude controller can be fed terrain altitude data or data from range finder to attempt to reach target altitude. The flow of data for altitude control as as below:

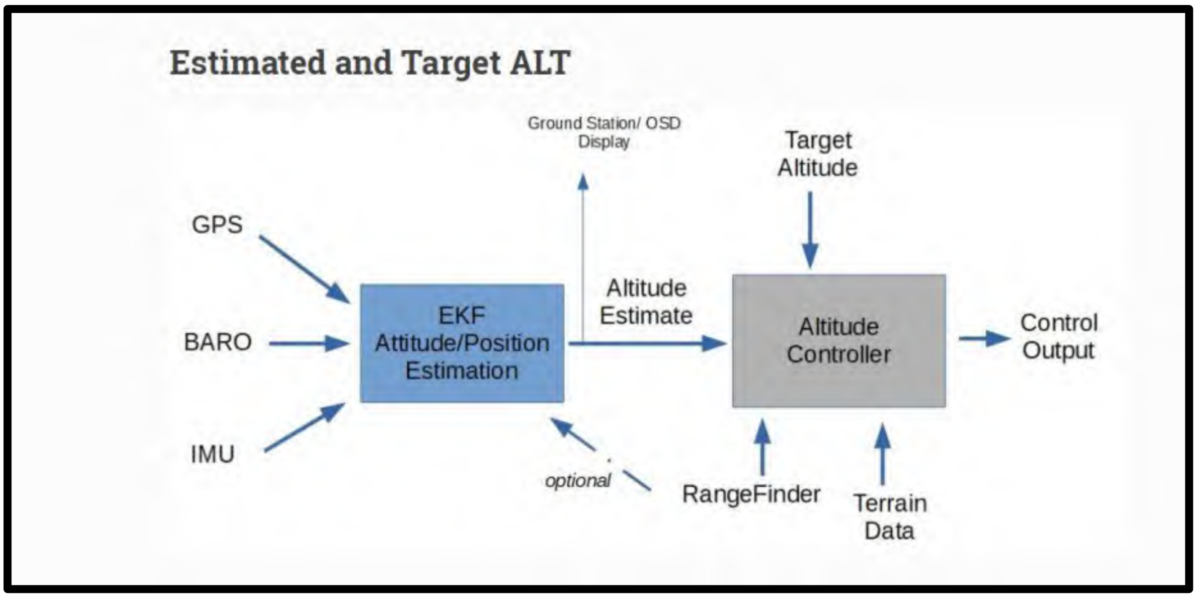
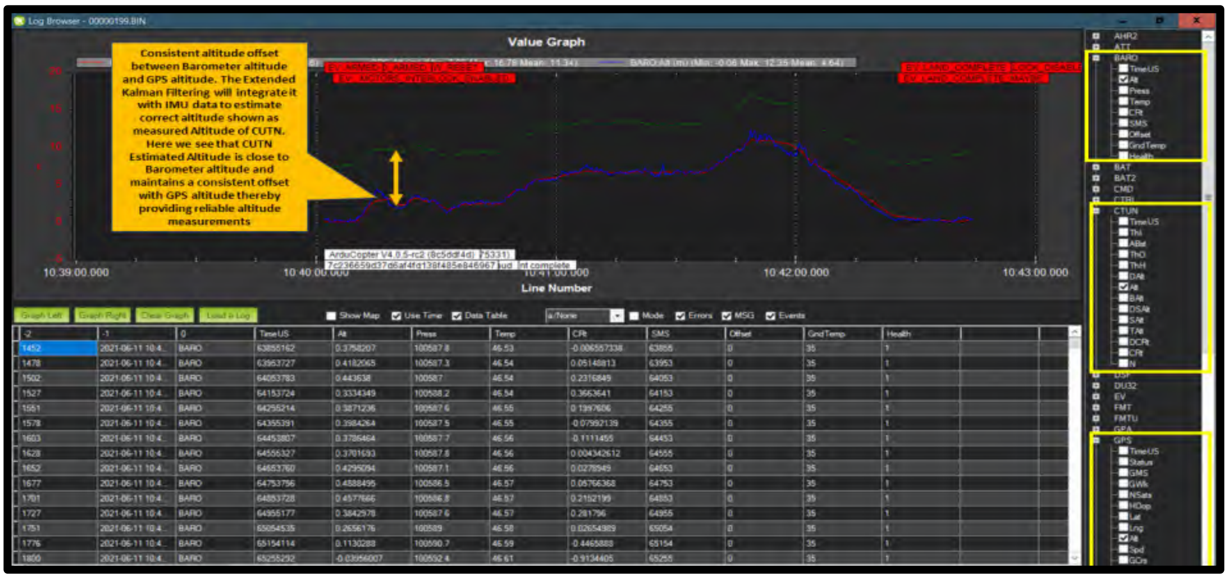


Figure 10.47: Altitude information flow in Ardupilot autopilot application stack [3]

The log review of ALT readings depicted by CUTN, BARO and GPS in log browser. We can see consistent altitude offset between Barometer altitude and GPS altitude. The Extended Kalman Filtering which integrates the measured values with the IMU data to estimate correct altitude shown as measured Altitude of CUTN. Here we see that CUTN Estimated Altitude is close to Barometer altitude and maintains a consistent offset with GPS altitude thereby providing reliable altitude measurements. Hence, the altitude estimation for the drone is reliable as per log analysis of V Type drone flown at Chemplast Stadium at IITM. The x axis in graph is the time of flight and y axis is the altitude in meters.



Consistent altitude offset between Barometer altitude and GPS altitude. The Extended Kalman Filtering will integrate it with IMU data to estimate correct altitude shown as measured Altitude of CUTN. Here we see that CUTN Estimated Altitude is close to Barometer altitude and maintains a consistent offset with GPS altitude thereby providing reliable altitude measurements

ArduCopter V4.0.5-rc2 (8c5ddf4d) 75331)
7c236659d37d6af4fd138f485e846967 ud Init complete

-2	-1	0	TimeUS	Alt	Press	Temp	CRt	SMS	Offset	GndTemp	Health
1452	2021-06-11 10:4...	BARO	63855162	0.3758207	100587.8	46.53	-0.006557338	63855	0	35	1
1478	2021-06-11 10:4...	BARO	63953727	0.4182065	100587.3	46.54	0.05148813	63953	0	35	1
1502	2021-06-11 10:4...	BARO	64053783	0.443638	100587	46.54	0.2316849	64053	0	35	1
1527	2021-06-11 10:4...	BARO	64153724	0.3334349	100588.2	46.54	0.3663641	64153	0	35	1
1551	2021-06-11 10:4...	BARO	64255214	0.3871236	100587.6	46.55	0.1397606	64255	0	35	1
1578	2021-06-11 10:4...	BARO	64355391	0.3984264	100587.5	46.55	-0.07992139	64355	0	35	1
1603	2021-06-11 10:4...	BARO	64453807	0.3786464	100587.7	46.56	-0.1111455	64453	0	35	1
1628	2021-06-11 10:4...	BARO	64556327	0.3701693	100587.8	46.56	0.004342612	64555	0	35	1
1652	2021-06-11 10:4...	BARO	64653760	0.4295094	100587.1	46.56	0.0278949	64653	0	35	1
1677	2021-06-11 10:4...	BARO	64753756	0.4588495	100586.5	46.57	0.05766368	64753	0	35	1
1701	2021-06-11 10:4...	BARO	64853728	0.4577666	100586.8	46.57	0.2152199	64853	0	35	1
1727	2021-06-11 10:4...	BARO	64955177	0.3842978	100587.6	46.57	0.281796	64955	0	35	1
1751	2021-06-11 10:4...	BARO	65054535	0.2656176	100589	46.58	0.02654989	65054	0	35	1
1776	2021-06-11 10:4...	BARO	65154114	0.1130288	100590.7	46.59	-0.4465888	65154	0	35	1
1800	2021-06-11 10:4...	BARO	65255292	-0.03956007	100592.4	46.61	-0.9134405	65255	0	35	1

Figure 10.48: Altitude information in log browser

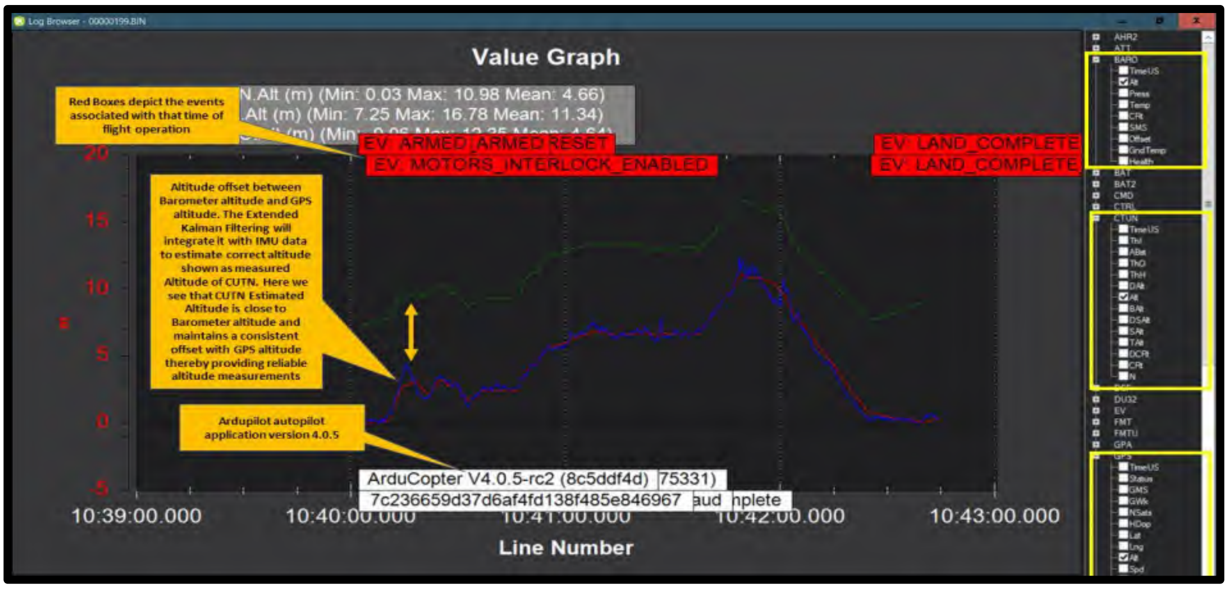
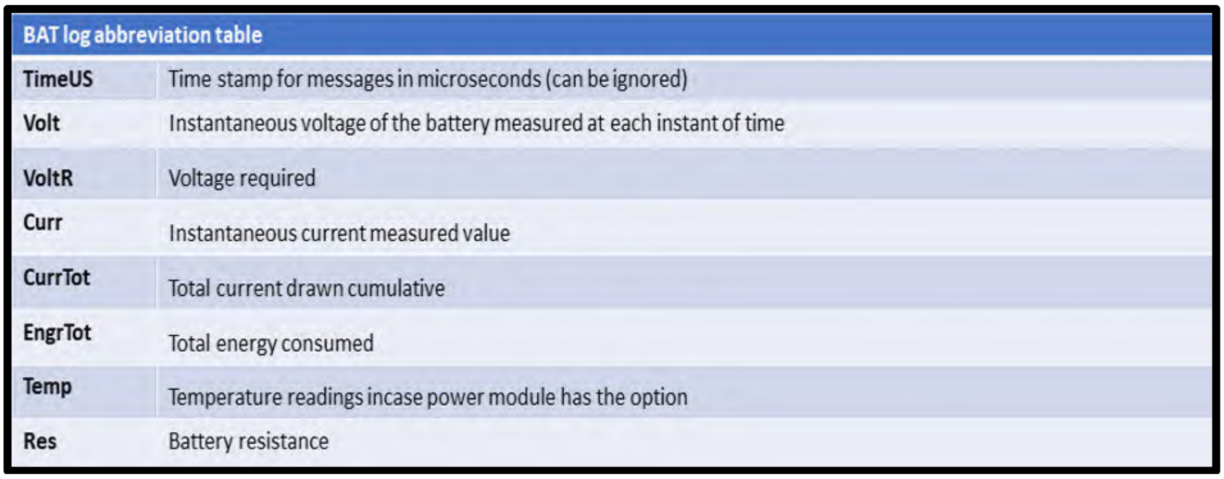

Log Browser - 00000199.BIN
Value Graph
Red Boxes depict the events associated with that time of flight operation
N.Alt (m) (Min: 0.03 Max: 10.98 Mean: 4.66)
Alt (m) (Min: 7.25 Max: 16.78 Mean: 11.34)
EV: ARMED ARMED RESET
EV: MOTORS_INTERLOCK_ENABLED
EV: LAND_COMPLETE
EV: LAND_COMPLETE
Altitude offset between Barometer altitude and GPS altitude. The Extended Kalman Filtering will integrate it with IMU data to estimate correct altitude shown as measured Altitude of CUTN. Here we see that CUTN Estimated Altitude is close to Barometer altitude and maintains a consistent offset with GPS altitude thereby providing reliable altitude measurements
Ardupilot autopilot application version 4.0.5
ArduCopter V4.0.5-rc2 (8c5ddf4d)
7c236659d37d6af4fd138f485e846967
20
15
10
5
0
-5
m
10:39:00.000
10:40:00.000
10:41:00.000
10:42:00.000
10:43:00.000
Line Number

Figure 10.49: Altitude information in log browser

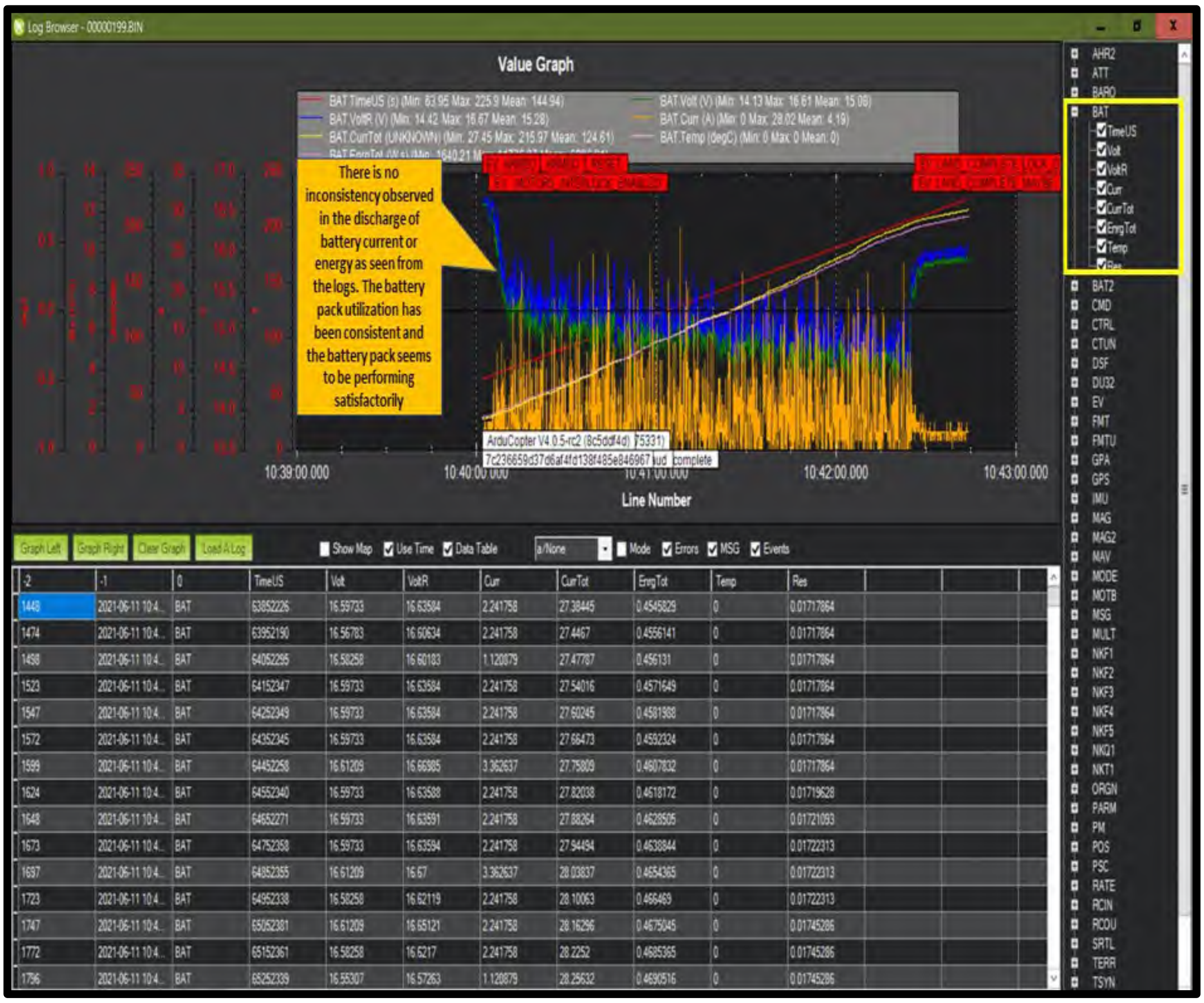
Batt monitor errors Battery monitor logs are the most important to view during the flight as well as after flight. The battery statistics are captured through APM power module between the battery pack and the power distribution board or DC-DC convertor. The battery consistency is a must for reliable drone operations. Also, the battery experiences the maximum wear and tear after each drone subject to multiple conditions. Hence, its important to monitor any abrupt variations in battery voltage , current or energy consumption to estimate the overall operating state of battery pack powering the drone. Failure of battery pack is a single point of failure for entire drone and hence the logs must be reviewed in detail each time drone is airborne. The details of battery monitor (BAT) log settings is as given below:



BAT log abbreviation table	
TimeUS	Time stamp for messages in microseconds (can be ignored)
Volt	Instantaneous voltage of the battery measured at each instant of time
VoltR	Voltage required
Curr	Instantaneous current measured value
CurrTot	Total current drawn cumulative
EngrTot	Total energy consumed
Temp	Temperature readings incase power module has the option
Res	Battery resistance

Figure 10.50: Battery BAT log abbreviation table

The log review of BAT readings in log browser shows there is no inconsistency observed in the discharge of battery current or energy as seen from the logs. The voltage reading is fairly consistent too. The battery pack is able to retain charge and is able to provide and keep up with the power requirements of the drone. Hence, we conclude after analyzing the BAT logs in log browser that the battery pack utilization has been consistent. The battery pack in this case 4s (16.8 Volts max) and 5200 mAh LiPo Lithium Polymer battery seems to be performing satisfactorily during the flight operation of the drone. These logs have been analysed after the flight of V Type drone flown at Chemplast Stadium at IITM. The x axis in the graphs is the time of flight while y axis is in volts for voltage readings, amperes for current, Watt second for energy and ohm for resistance.

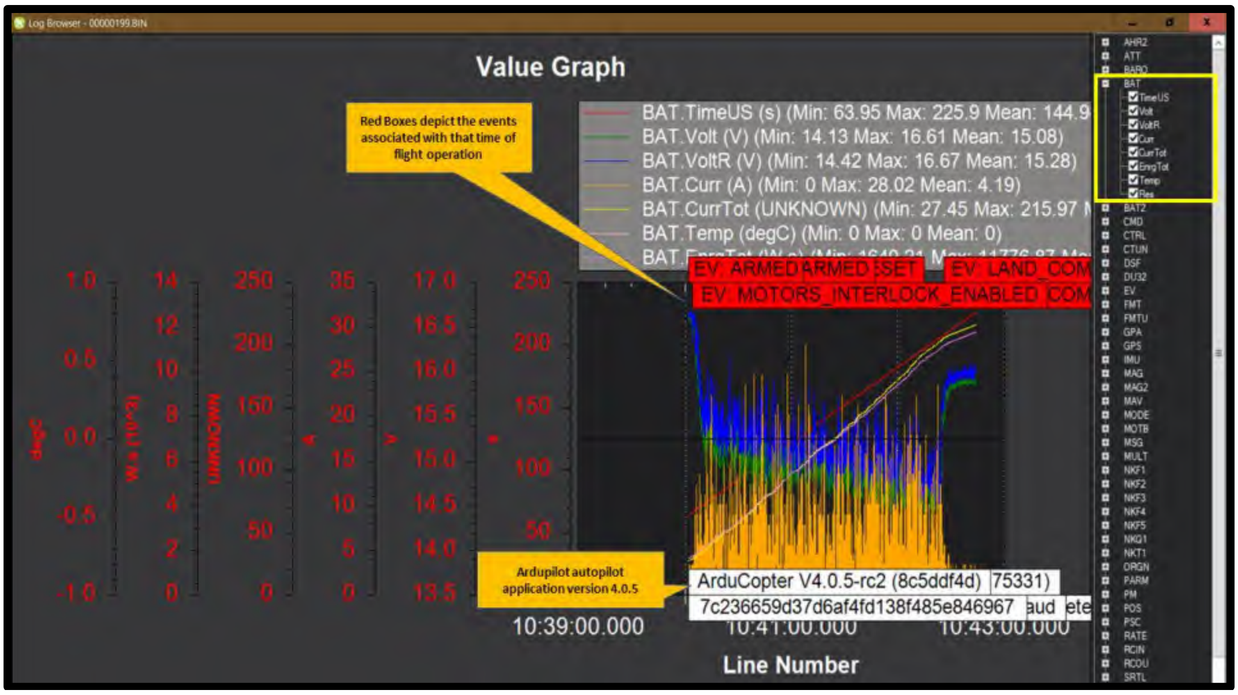


BAT.TimeUS (s) (Min: 63.95 Max: 225.9 Mean: 144.94)
BAT.VoltR (V) (Min: 14.42 Max: 16.67 Mean: 15.28)
BAT.CurrTot (UNKNOWN) (Min: 27.45 Max: 215.97 Mean: 124.61)
BAT.Volt (V) (Min: 14.13 Max: 16.61 Mean: 15.08)
BAT.Curr (A) (Min: 0 Max: 28.02 Mean: 4.19)
BAT.Temp (degC) (Min: 0 Max: 0 Mean: 0)

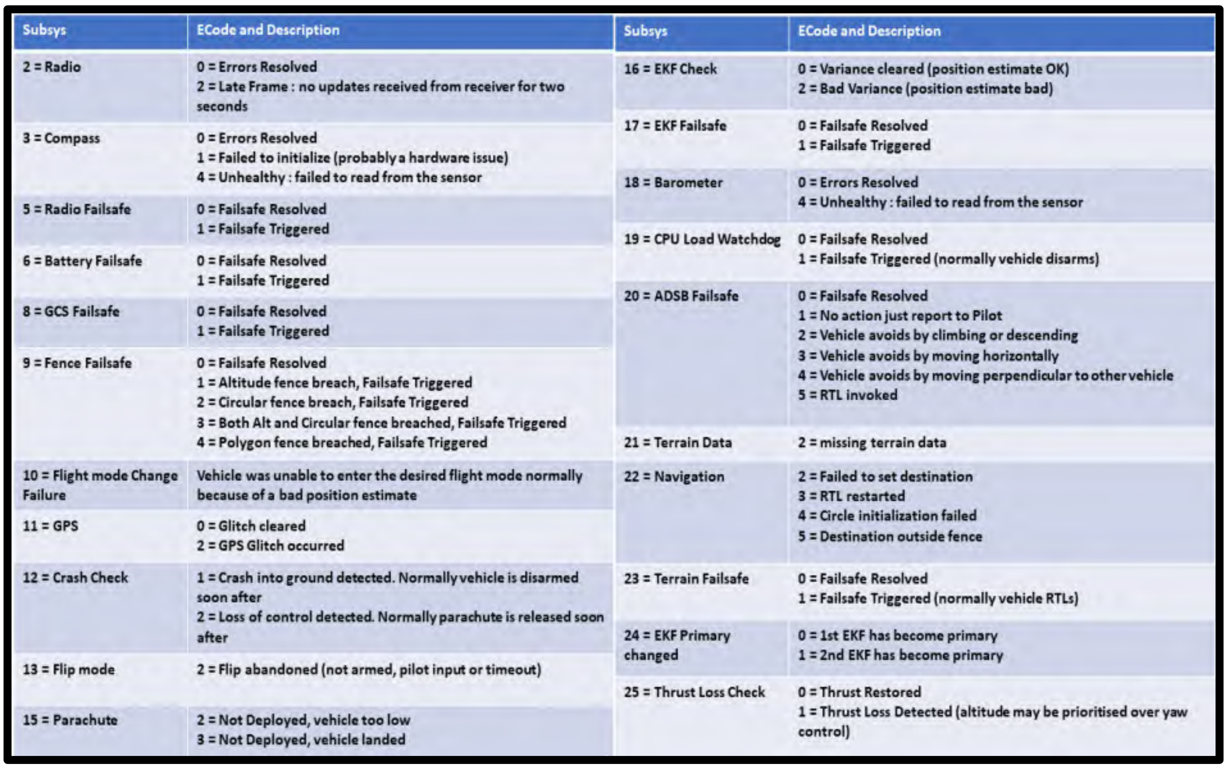
There is no inconsistency observed in the discharge of battery current or energy as seen from the logs. The battery pack utilization has been consistent and the battery pack seems to be performing satisfactorily

-2	-1	0	TimeUS	Volt	VoltR	Curr	CurrTot	EnrgTot	Temp	Res
1448	2021-06-11 10:4...	BAT	63852226	16.59733	16.63584	2.241758	27.38445	0.4545829	0	0.01717864
1474	2021-06-11 10:4...	BAT	63952190	16.56783	16.60634	2.241758	27.4467	0.4556141	0	0.01717864
1498	2021-06-11 10:4...	BAT	64052295	16.58258	16.60183	1.120879	27.47787	0.456131	0	0.01717864
1523	2021-06-11 10:4...	BAT	64152347	16.59733	16.63584	2.241758	27.54016	0.4571649	0	0.01717864
1547	2021-06-11 10:4...	BAT	64252349	16.59733	16.63584	2.241758	27.60245	0.4581988	0	0.01717864
1572	2021-06-11 10:4...	BAT	64352345	16.59733	16.63584	2.241758	27.66473	0.4592324	0	0.01717864
1599	2021-06-11 10:4...	BAT	64452258	16.61209	16.66985	3.362637	27.75809	0.4607832	0	0.01717864
1624	2021-06-11 10:4...	BAT	64552340	16.59733	16.63588	2.241758	27.82038	0.4618172	0	0.01719628
1648	2021-06-11 10:4...	BAT	64652271	16.59733	16.63591	2.241758	27.88264	0.4628505	0	0.01721093
1673	2021-06-11 10:4...	BAT	64752358	16.59733	16.63594	2.241758	27.94494	0.4638844	0	0.01722313
1697	2021-06-11 10:4...	BAT	64852355	16.61209	16.67	3.362637	28.03837	0.4654365	0	0.01722313
1723	2021-06-11 10:4...	BAT	64952338	16.58258	16.62119	2.241758	28.10063	0.466469	0	0.01722313
1747	2021-06-11 10:4...	BAT	65052381	16.61209	16.65121	2.241758	28.16296	0.4675045	0	0.01745286
1772	2021-06-11 10:4...	BAT	65152361	16.58258	16.6217	2.241758	28.2252	0.4685365	0	0.01745286
1796	2021-06-11 10:4...	BAT	65252339	16.55307	16.57263	1.120879	28.25632	0.4690516	0	0.01745286

Figure 10.51: Battery information through BAT logs in log browser


Log Browser - 00000199.BIN
Value Graph
Red Boxes depict the events associated with that time of flight operation
BAT.TimeUS (s) (Min: 63.95 Max: 225.9 Mean: 144.9
BAT.Volt (V) (Min: 14.13 Max: 16.61 Mean: 15.08)
BAT.VoltR (V) (Min: 14.42 Max: 16.67 Mean: 15.28)
BAT.Curr (A) (Min: 0 Max: 28.02 Mean: 4.19)
BAT.CurrTot (UNKNOWN) (Min: 27.45 Max: 215.97
BAT.Temp (degC) (Min: 0 Max: 0 Mean: 0)
EV: ARMED ARMED SET
EV: LAND_COM
EV: MOTORS_INTERLOCK_ENABLED COM
Ardupilot autopilot application version 4.0.5
ArduCopter V4.0.5-rc2 (8c5ddf4d) 75331)
7c236659d37d6af4fd138f485e846967
10:39:00.000
10:41:00.000
10:43:00.000
Line Number
BAT
TimeUS
Volt
VoltR
Curr
CurrTot
EnrgTot
Temp
Res
Figure 10.52: Battery information through BAT logs in log browser

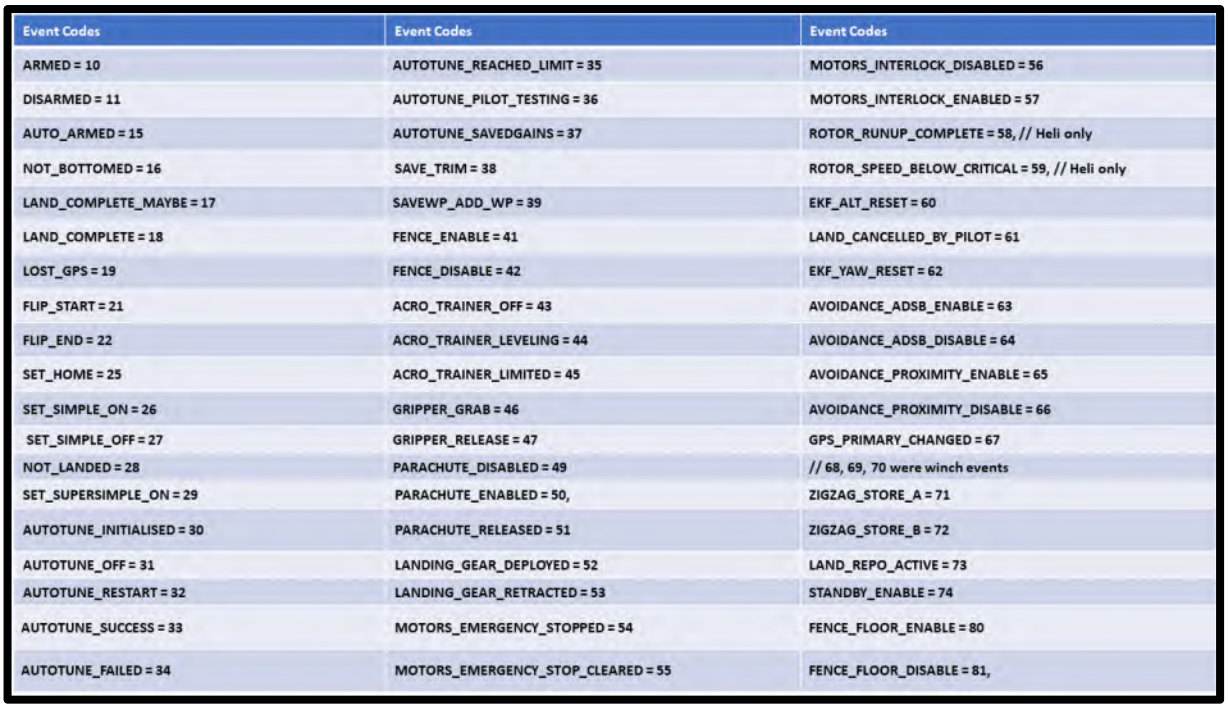
Unexpected ERRORS codes during log analysis including Failsafes The unexpected behavior of drone will generate the error codes by ERR for each log type and can be filtered out and analyzed for resolving them. The list of codes as below:



Subsys	ECode and Description	Subsys	ECode and Description
2 = Radio	0 = Errors Resolved 2 = Late Frame : no updates received from receiver for two seconds	16 = EKF Check	0 = Variance cleared (position estimate OK) 2 = Bad Variance (position estimate bad)
3 = Compass	0 = Errors Resolved 1 = Failed to initialize (probably a hardware issue) 4 = Unhealthy : failed to read from the sensor	17 = EKF Failsafe	0 = Failsafe Resolved 1 = Failsafe Triggered
5 = Radio Failsafe	0 = Failsafe Resolved 1 = Failsafe Triggered	18 = Barometer	0 = Errors Resolved 4 = Unhealthy : failed to read from the sensor
6 = Battery Failsafe	0 = Failsafe Resolved 1 = Failsafe Triggered	19 = CPU Load Watchdog	0 = Failsafe Resolved 1 = Failsafe Triggered (normally vehicle disarms)
8 = GCS Failsafe	0 = Failsafe Resolved 1 = Failsafe Triggered	20 = ADSB Failsafe	0 = Failsafe Resolved 1 = No action just report to Pilot 2 = Vehicle avoids by climbing or descending 3 = Vehicle avoids by moving horizontally 4 = Vehicle avoids by moving perpendicular to other vehicle 5 = RTL invoked
9 = Fence Failsafe	0 = Failsafe Resolved 1 = Altitude fence breach, Failsafe Triggered 2 = Circular fence breach, Failsafe Triggered 3 = Both Alt and Circular fence breached, Failsafe Triggered 4 = Polygon fence breached, Failsafe Triggered	21 = Terrain Data	2 = missing terrain data
10 = Flight mode Change Failure	Vehicle was unable to enter the desired flight mode normally because of a bad position estimate	22 = Navigation	2 = Failed to set destination 3 = RTL restarted 4 = Circle initialization failed 5 = Destination outside fence
11 = GPS	0 = Glitch cleared 2 = GPS Glitch occurred		
12 = Crash Check	1 = Crash into ground detected. Normally vehicle is disarmed soon after 2 = Loss of control detected. Normally parachute is released soon after	23 = Terrain Failsafe	0 = Failsafe Resolved 1 = Failsafe Triggered (normally vehicle RTLs)
		24 = EKF Primary changed	0 = 1st EKF has become primary 1 = 2nd EKF has become primary
13 = Flip mode	2 = Flip abandoned (not armed, pilot input or timeout)	25 = Thrust Loss Check	0 = Thrust Restored 1 = Thrust Loss Detected (altitude may be prioritised over yaw control)
15 = Parachute	2 = Not Deployed, vehicle too low 3 = Not Deployed, vehicle landed		

Figure 10.53: List of error codes ERR with their numbers

Event Codes The logs viewer also has the option of viewing events given by EV in the graph browser. The Event viewer codes are as given below:-



Event Codes	Event Codes	Event Codes
ARMED = 10	AUTOTUNE_REACHED_LIMIT = 35	MOTORS_INTERLOCK_DISABLED = 56
DISARMED = 11	AUTOTUNE_PILOT_TESTING = 36	MOTORS_INTERLOCK_ENABLED = 57
AUTO_ARMED = 15	AUTOTUNE_SAVEDGAINS = 37	ROTOR_RUNUP_COMPLETE = 58, // Heli only
NOT_BOTTOMED = 16	SAVE_TRIM = 38	ROTOR_SPEED_BELOW_CRITICAL = 59, // Heli only
LAND_COMPLETE_MAYBE = 17	SAVEWP_ADD_WP = 39	EKF_ALT_RESET = 60
LAND_COMPLETE = 18	FENCE_ENABLE = 41	LAND_CANCELLED_BY_PILOT = 61
LOST_GPS = 19	FENCE_DISABLE = 42	EKF_YAW_RESET = 62
FLIP_START = 21	ACRO_TRAINER_OFF = 43	AVOIDANCE_ADSB_ENABLE = 63
FLIP_END = 22	ACRO_TRAINER_LEVELING = 44	AVOIDANCE_ADSB_DISABLE = 64
SET_HOME = 25	ACRO_TRAINER_LIMITED = 45	AVOIDANCE_PROXIMITY_ENABLE = 65
SET_SIMPLE_ON = 26	GRIPPER_GRAB = 46	AVOIDANCE_PROXIMITY_DISABLE = 66
SET_SIMPLE_OFF = 27	GRIPPER_RELEASE = 47	GPS_PRIMARY_CHANGED = 67
NOT_LANDED = 28	PARACHUTE_DISABLED = 49	// 68, 69, 70 were winch events
SET_SUPERSIMPLE_ON = 29	PARACHUTE_ENABLED = 50,	ZIGZAG_STORE_A = 71
AUTOTUNE_INITIALISED = 30	PARACHUTE_RELEASED = 51	ZIGZAG_STORE_B = 72
AUTOTUNE_OFF = 31	LANDING_GEAR_DEPLOYED = 52	LAND_REPO_ACTIVE = 73
AUTOTUNE_RESTART = 32	LANDING_GEAR_RETRACTED = 53	STANDBY_ENABLE = 74
AUTOTUNE_SUCCESS = 33	MOTORS_EMERGENCY_STOPPED = 54	FENCE_FLOOR_ENABLE = 80
AUTOTUNE_FAILED = 34	MOTORS_EMERGENCY_STOP_CLEARED = 55	FENCE_FLOOR_DISABLE = 81,

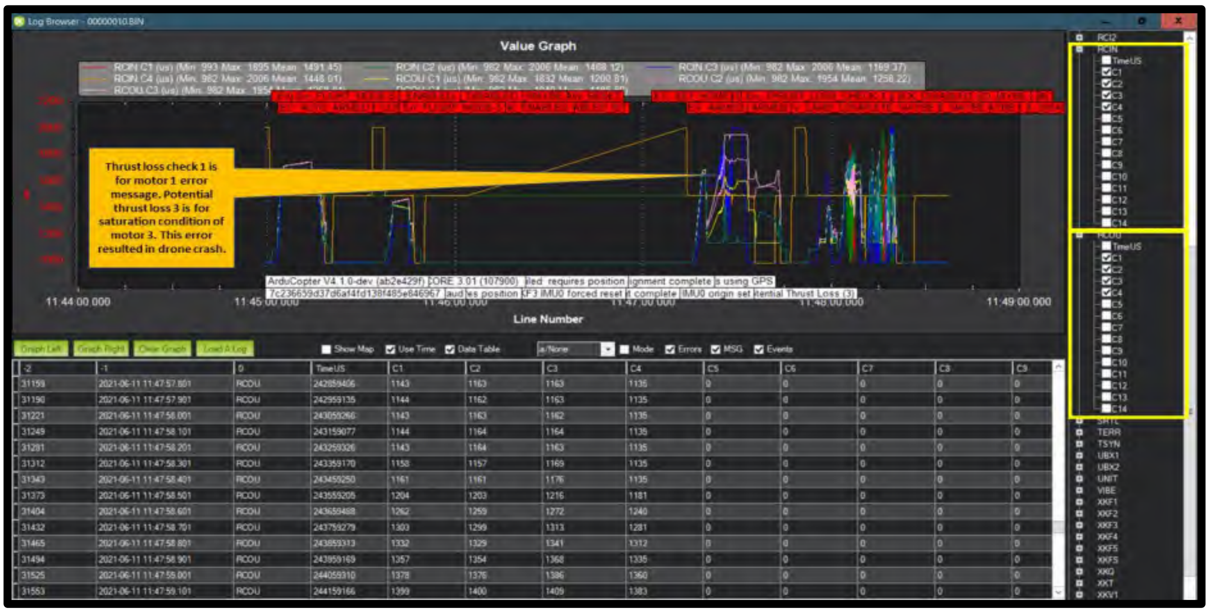
Figure 10.54: List of event EV with their numbers

PWM signal to motors and its output logs RCIN & RCOUT (Pulse Width Modulation (PWM) input and output to individual RC outputs ine motor/ESC combine. The RCIN and RCOUT logs give us a view in to performance of each ESC/motor combine during the flight. RC1, RC2, RC3, RC4 incase of quadcopter and so on for any frame type gives details of Pulse Width Modulation (PWM) command sent by Ardupilot autopilot application during the flight to the ESC/motor/RC input and output. The RCIN and RCOUT log graphs have to be read in conjunction with mechanical errors described before using ATT and RATE. RATE is similar to ATT logs as explained earlier but gives the rate of change of yaw, pich and roll. ATT gives degrees of deviation between desired and actual values while RATE gives degrees per second deviation between expected and actual values for change in yaw, pitch and roll. These logs analyzed together can give important analysis for finding out about mechanical failures as well as requirement of tuning to improve stability of drone. The PID tuning is done based on the output of these graphs and will be discussed subsequently.

Below are important warning messages which are displayed in Mission Planner during flight and the same can be seen during the log analysis of RCOUT, ATT and RATE. Understanding them is critical to reliable functioning of the drone.

Thrust Loss and Yaw Imbalance Warnings This warning is received mostly incase of incorrect hardware selection, hardware failure in propulsion system or improper tuning of PID values with respect to the frame type. If these warnings are displayed just with the take-off of aircraft then its more to do with hardware failure or poorly tuned aircraft but when received mid-flight its more likely a sudden hardware failure.

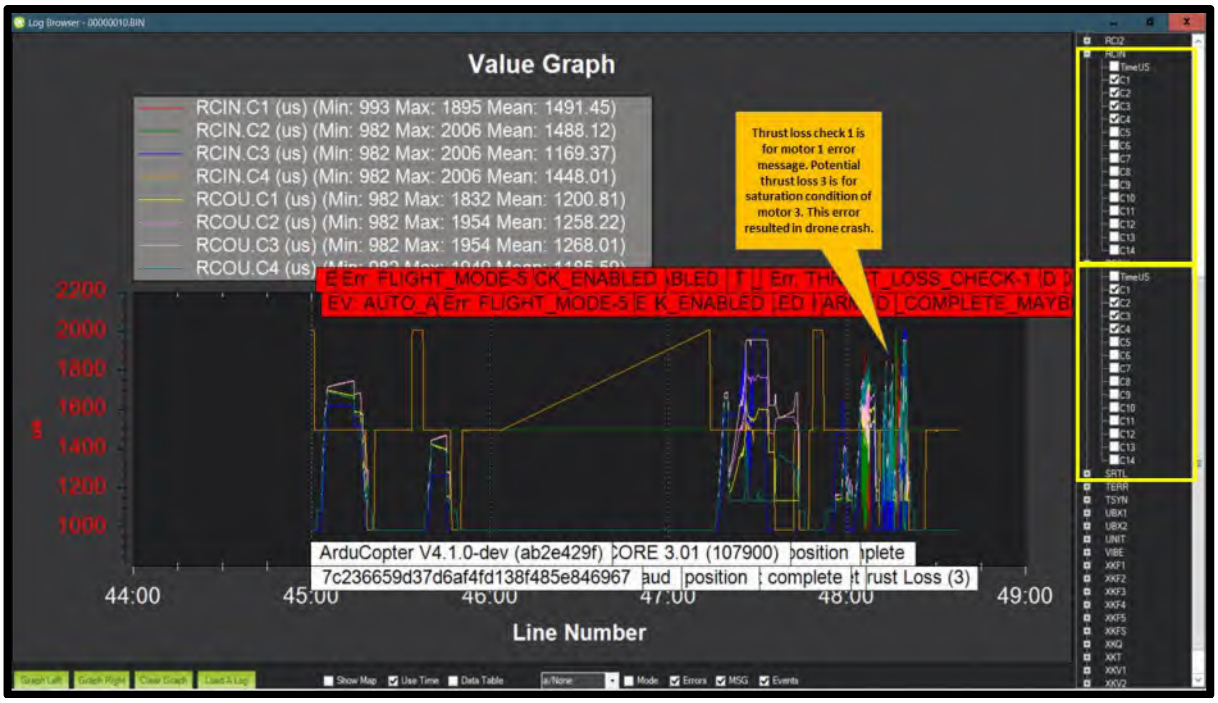
Potential Thrust Loss This message is displayed on GCS i.e Mission Planner or during log analysis signifies saturation of motor at 100% throttle. This mostly happens when one motor fails and the other tries to compensate for thrust loss of faulty motor/ESC combine. As a result of this saturation, the Ardupilot autopilot application is unable to achieve desired /required roll, pitch, yaw and throttle output. This can potentially result in drone crash. A message like **`Potential Thrust Loss (3)`** signifies potential thrust loss for Motor number 3. We need to evaluate the RCOUT to see which motor failed for which this motor tried to compensate. In the screenshot of drone crash on 11 June 2021 clearly depicts motor 1 failed resulting in Motor 3 trying to compensate and going into saturation with potential thrust loss. As a result the drone flipped on right and crashed. The reason for failure of motor will need further analysis to find the reason for it such as mechanical failure of motor or ESC or simply propeller damage. This can be done by physical examination and table top testing.



Log Browser - 00000010.BIN
Value Graph
RCIN C1 (us) (Min: 993 Max: 1695 Mean: 1491.45)
RCIN C2 (us) (Min: 982 Max: 2006 Mean: 1468.12)
RCIN C3 (us) (Min: 982 Max: 2006 Mean: 1169.37)
RCIN C4 (us) (Min: 982 Max: 2006 Mean: 1448.01)
RCOU C1 (us) (Min: 982 Max: 1832 Mean: 1200.81)
RCOU C2 (us) (Min: 982 Max: 1954 Mean: 1258.22)
RCOU C3 (us) (Min: 982 Max: 1954
Thrust loss check 1 is for motor 1 error message. Potential thrust loss 3 is for saturation condition of motor 3. This error resulted in drone crash.
ArduCopter V4.1.0-dev (ab2e429f)
7c236659d37d6af4fd138f485e846967
CORE 3.01 (107900)
EKF3 IMU0 forced reset
IMU0 origin set
tential Thrust Loss (3)
11:44:00.000
11:45:00.000
11:46:00.000
11:47:00.000
11:48:00.000
11:49:00.000
Line Number
Graph Left
Graph Right
Clear Graph
Load A Log
Show Map
Use Time
Data Table
Mode
Errors
MSG
Events

-2	-1	0	TimeUS	C1	C2	C3	C4	C5	C6	C7	C8	C9
31159	2021-06-11 11:47:57.801	RCOU	242859406	1143	1163	1163	1135	0	0	0	0	0
31190	2021-06-11 11:47:57.901	RCOU	242959135	1144	1162	1163	1135	0	0	0	0	0
31221	2021-06-11 11:47:58.001	RCOU	243059266	1143	1163	1162	1135	0	0	0	0	0
31249	2021-06-11 11:47:58.101	RCOU	243159077	1144	1164	1164	1135	0	0	0	0	0
31281	2021-06-11 11:47:58.201	RCOU	243259326	1143	1164	1163	1135	0	0	0	0	0
31312	2021-06-11 11:47:58.301	RCOU	243359170	1158	1157	1169	1135	0	0	0	0	0
31343	2021-06-11 11:47:58.401	RCOU	243459250	1161	1161	1176	1135	0	0	0	0	0
31373	2021-06-11 11:47:58.501	RCOU	243559205	1204	1203	1216	1181	0	0	0	0	0
31404	2021-06-11 11:47:58.601	RCOU	243659488	1262	1259	1272	1240	0	0	0	0	0
31432	2021-06-11 11:47:58.701	RCOU	243759279	1303	1299	1313	1281	0	0	0	0	0
31465	2021-06-11 11:47:58.801	RCOU	243859313	1332	1329	1341	1312	0	0	0	0	0
31494	2021-06-11 11:47:58.901	RCOU	243959169	1357	1354	1368	1335	0	0	0	0	0
31525	2021-06-11 11:47:59.001	RCOU	244059310	1378	1376	1386	1360	0	0	0	0	0
31553	2021-06-11 11:47:59.101	RCOU	244159166	1399	1400	1409	1383	0	0	0	0	0

Figure 10.55: RCIN and RCOUT PWM signal log information in log browser


Log Browser - 00000010.BIN
Value Graph
Thrust loss check 1 is for motor 1 error message. Potential thrust loss 3 is for saturation condition of motor 3. This error resulted in drone crash.
2200
2000
1800
1600
1400
1200
1000
44:00
45:00
46:00
47:00
48:00
49:00
Line Number
Graph Left
Graph Right
Clear Graph
Load A Log
Show Map
Use Time
Data Table
Mode
Errors
MSG
Events

RCIN.C1 (us) (Min: 993 Max: 1895 Mean: 1491.45)
RCIN.C2 (us) (Min: 982 Max: 2006 Mean: 1488.12)
RCIN.C3 (us) (Min: 982 Max: 2006 Mean: 1169.37)
RCIN.C4 (us) (Min: 982 Max: 2006 Mean: 1448.01)
RCOU.C1 (us) (Min: 982 Max: 1832 Mean: 1200.81)
RCOU.C2 (us) (Min: 982 Max: 1954 Mean: 1258.22)
RCOU.C3 (us) (Min: 982 Max: 1954 Mean: 1268.01)
RCOU.C4 (us)

```
E Err: FLIGHT_MODE-5 CK_ENABLED BLED T Err: THR T_LOSS_CHECK-1 D
EV: AUTO_A Err: FLIGHT_MODE-5 E K_ENABLED ED ARM D COMPLETE_MAYB
ArduCopter V4.1.0-dev (ab2e429f) CORE 3.01 (107900) position plete
7c236659d37d6af4fd138f485e846967 aud position complete t rust Loss (3)
```

Figure 10.56: RCIN and RCOUT PWM signal log information in log browser

If such error messages are received during hover or an altitude mode then it could probably be due to increased weight to thrust ratio requiring to retune to frame design and its components to reduce overall weight or change in propulsion system (motor/propellers). We must not that change in propulsion system must be done with proper drone calculations as higher propulsion system will require a bigger battery increasing the weight. This trade-off of thrust vs weight must be done as shown in chapter 1 earlier.

Incase of aggressive climbing and maneuvers, there is a requirement to lower the requested speed and acceleration.

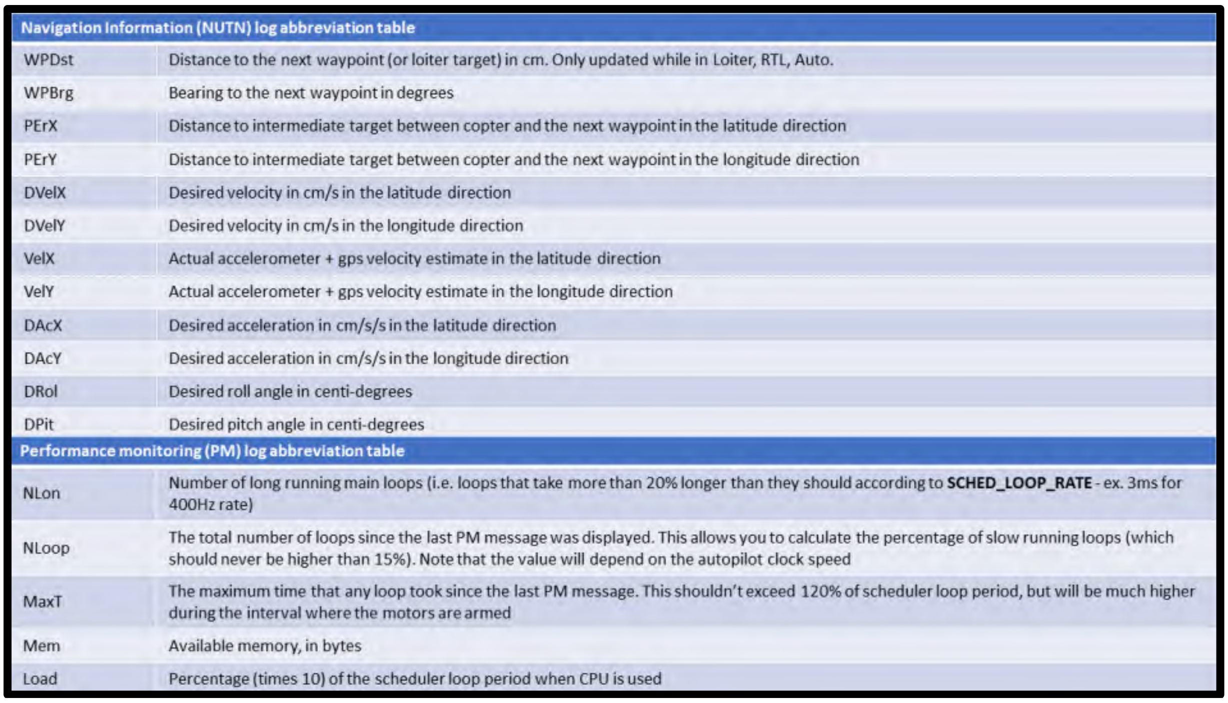
Yaw Imbalance The yaw imbalance warning depicts how much effort the drone is working on yaw. Incase of saturation, the ability of drone to maintain yaw is lost. 100% means complete saturation. The message reads something like this giving percentage of yaw imbalance. **`Yaw Imbalance 87%`.** The yaw imbalance mode in hover or fixed altitude flight mode, the issue is most probably with the hardware. We need to evaluate the logs for PWN input signal in RCIN log settings and compare the PWN between pars in opposing motors. If large throttle level difference between the clockwise and anti/counter clockwise motors is seen in the logs, then it is again likely due to hardware error. Most probably its when motors are not vertically on circular arms. This error can be cleared by rotating the motors to compensate for it as the thrust angle will assists the yaw in its rotation direction. If this yaw imbalance warning is seen during aggressive yaw maneuvers, the warning threshold need to be increased by raising **ATC_RAT_YAW_IMAX** in full parameter list**.**

Thus, after understanding all about configuration, fail safe pre arm checklist, understanding about logs and its impact on drone operations, all the logs messages we are now in a position to finally tune PID values for advanced configuration options.

Other log evaluation options

IMU (accelerometer and gyro information)

GyrX, GyrY, GyrZ	The raw gyro rotation rates in radians/second for each of x,y and z axis
AccX, AccY, AccZ	The raw accelerometer values in m/s/s for each of x,y and z axis



Navigation Information (NUTN) log abbreviation table	
WPDst	Distance to the next waypoint (or loiter target) in cm. Only updated while in Loiter, RTL, Auto.
WPBrg	Bearing to the next waypoint in degrees
PErX	Distance to intermediate target between copter and the next waypoint in the latitude direction
PErY	Distance to intermediate target between copter and the next waypoint in the longitude direction
DVelX	Desired velocity in cm/s in the latitude direction
DVelY	Desired velocity in cm/s in the longitude direction
VelX	Actual accelerometer + gps velocity estimate in the latitude direction
VelY	Actual accelerometer + gps velocity estimate in the longitude direction
DAcX	Desired acceleration in cm/s/s in the latitude direction
DAcY	Desired acceleration in cm/s/s in the longitude direction
DRol	Desired roll angle in centi-degrees
DPit	Desired pitch angle in centi-degrees
Performance monitoring (PM) log abbreviation table	
NLon	Number of long running main loops (i.e. loops that take more than 20% longer than they should according to **SCHED_LOOP_RATE** - ex. 3ms for 400Hz rate)
NLoop	The total number of loops since the last PM message was displayed. This allows you to calculate the percentage of slow running loops (which should never be higher than 15%). Note that the value will depend on the autopilot clock speed
MaxT	The maximum time that any loop took since the last PM message. This shouldn't exceed 120% of scheduler loop period, but will be much higher during the interval where the motors are armed
Mem	Available memory, in bytes
Load	Percentage (times 10) of the scheduler loop period when CPU is used

Figure 10.57: Other log parameters for information exploring in log browser

10.5 Advanced tuning option for optimized flight operations

In PID (Proportional - Integral - Derivative) tuning is a control loop mechanism by which the drone firmware stabilizes the drone/vehicle. P stands for Proportional which refers to the immediate Correction. The more we are off this value, the bigger is the correction. I stands for Integral which refers to over time or steady state correction. The additional corrections are added if we don’t make progress. D stands for derivative which means like take it easy correction. It slows down or dampen if the corrections are too fast.

Roll/Pitch tuning, the rate parameter is the most important. It convert the desired rate of rotation into motor output. The Rate Roll and Pitch P tuning page has most information about tuning them. The Stabilize Roll and Pitch P converts the required angle into a desired rotation rate. This rate is thereafter fed to the rate controller. A higher value makes the drone/copter more responsive the inputs while lower value will make it smoother. A very high value makes drone/copter to oscillate on the roll and or the pitch axis. A very low value makes drone/copter sluggish.

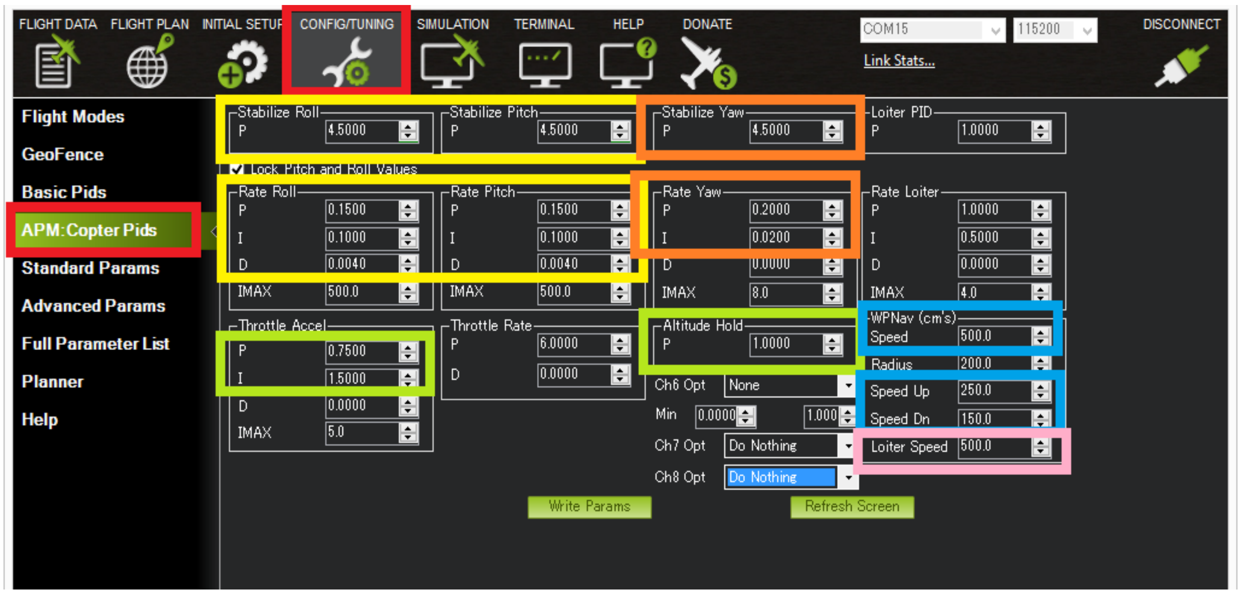
Yaw tuning, the Stabilize Yaw and Rate Yaw parameters are used to tune yaw which in most cases does not require tuning. It is similar to roll and pitch. If the value is too high the drone/copter will oscillate else will be unable to maintain heading if too low.

Altitude Tuning, hold related tuning parameters. Altitude Hold P is used to convert the altitude error which is the difference between the desired altitude and the actual altitude in order to maintain desired climb or descent rate. It the value is high, the drone/copter will make aggressive attempt to gain altitude and very high will give jerky throttle response to pilot inputs.

The Throttle Rate more than often does not require any change. It converts the desired climb or descent rate into a corresponding acceleration up or down. The Throttle Accel (Proportional - Integral - Derivative) PID gains convert the acceleration error. It is the difference between the desired acceleration and the actual acceleration into a motor output. The 1:2 ratio of P to I is recommended to be maintained.

Loiter Tuning, does not require much tuning in the case Roll and Pitch are tuned correctly with GPS working well and vibrations below accepted levels.

Mission Planner > Config > Extended Tuning



Stabilize Roll: P 4.5000
Stabilize Pitch: P 4.5000
Stabilize Yaw: P 4.5000
Loiter PID: P 1.0000

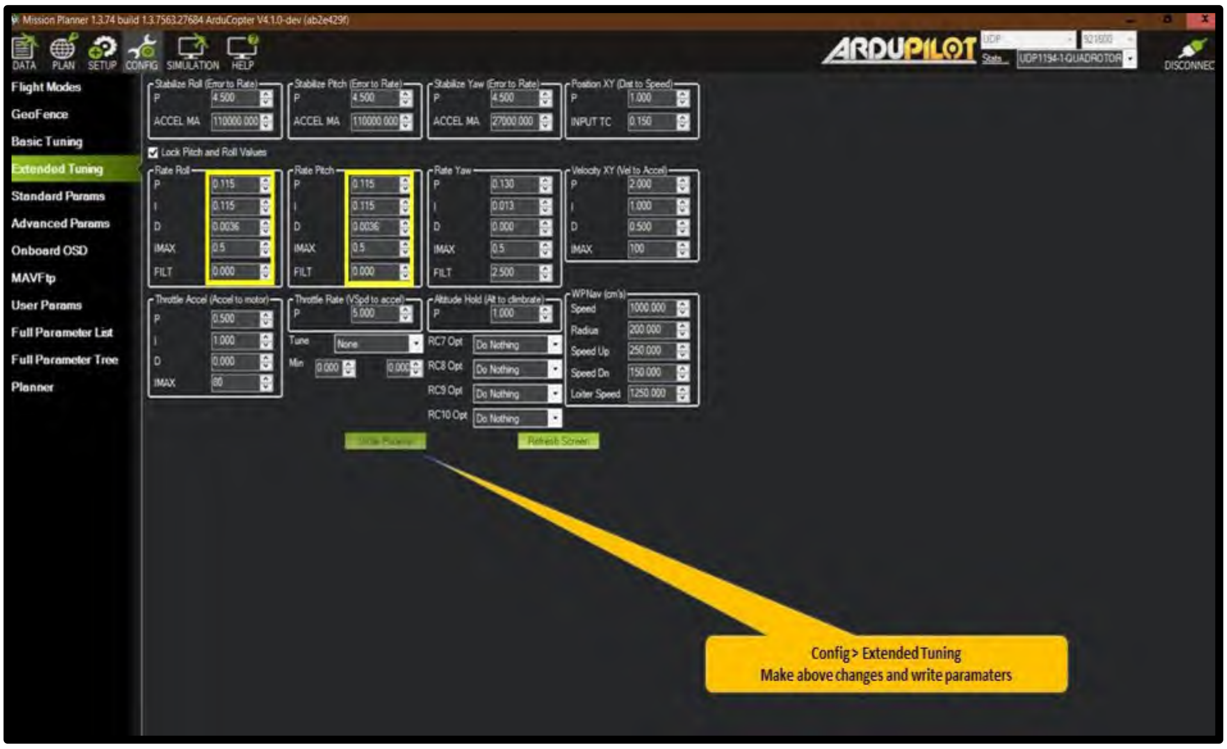
☑ Lock Pitch and Roll Values

Rate Roll: P 0.1500, I 0.1000, D 0.0040, IMAX 500.0
Rate Pitch: P 0.1500, I 0.1000, D 0.0040, IMAX 500.0
Rate Yaw: P 0.2000, I 0.0200, D 0.0000, IMAX 8.0
Rate Loiter: P 1.0000, I 0.5000, D 0.0000, IMAX 4.0

Throttle Accel: P 0.7500, I 1.5000, D 0.0000, IMAX 5.0
Throttle Rate: P 6.0000, D 0.0000
Altitude Hold: P 1.0000
WPNav (cm's): Speed 500.0, Radius 200.0, Speed Up 250.0, Speed Dn 150.0, Loiter Speed 500.0

Ch6 Opt: None
Min: 0.0000, 1.0000
Ch7 Opt: Do Nothing
Ch8 Opt: Do Nothing

Figure 10.58: Understanding PID tuning. The Roll or Pitch is in yellow box, Yaw is in orange box while, Altitude hold is in green box and Loiter is in pink box. The Waypoint navigation is in blue box. [3]



Mission Planner 1.3.74 build 1.3.7563.27684 ArduCopter V4.1.0-dev (ab2e429f)

Stabilize Roll (Error to Rate): P 4.500, ACCEL MA 110000.000
Stabilize Pitch (Error to Rate): P 4.500, ACCEL MA 110000.000
Stabilize Yaw (Error to Rate): P 4.500, ACCEL MA 27000.000
Position XY (Dist to Speed): P 1.000, INPUT TC 0.150

☑ Lock Pitch and Roll Values

Rate Roll: P 0.115, I 0.115, D 0.0036, IMAX 0.5, FILT 0.000
Rate Pitch: P 0.115, I 0.115, D 0.0036, IMAX 0.5, FILT 0.000
Rate Yaw: P 0.130, I 0.013, D 0.000, IMAX 0.5, FILT 2.500
Velocity XY (Vel to Accel): P 2.000, I 1.000, D 0.500, IMAX 100

Throttle Accel (Accel to motor): P 0.500, I 1.000, D 0.000, IMAX 80
Throttle Rate (VSpd to accel): P 5.000
Altitude Hold (Alt to climbrate): P 1.000
WPNav (cm's): Speed 1000.000, Radius 200.000, Speed Up 250.000, Speed Dn 150.000, Loiter Speed 1250.000

Tune: None
Min: 0.000, 0.000
RC7 Opt: Do Nothing
RC8 Opt: Do Nothing
RC9 Opt: Do Nothing
RC10 Opt: Do Nothing

Config > Extended Tuning
Make above changes and write paramaters

Figure 10.59: Suggested PID tuning values that need to be configured through Mission Planner (GCS application) and instructions sent to Ardupilot autopilot through MAVlink

CHAPTER 11

LIVE VIDEO STREAMING FROM DRONE OVER INTERNET (VPN) AND IMAGE DETECTION USING ARTIFICIAL INTELLIGENCE TOOL (YOLO v4 & YOLO v4 (TINY))

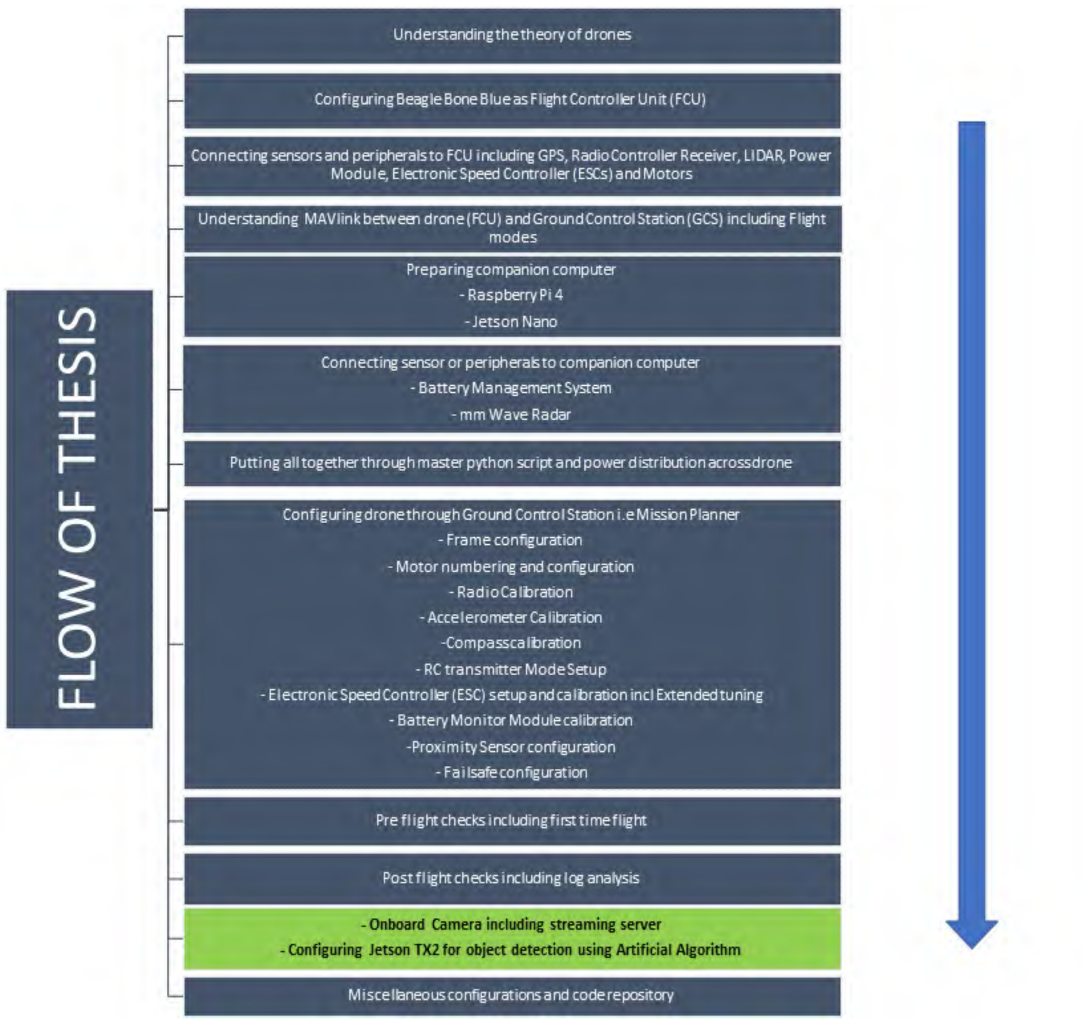

FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server
- Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository

Figure 11.1: Flow of thesis until now

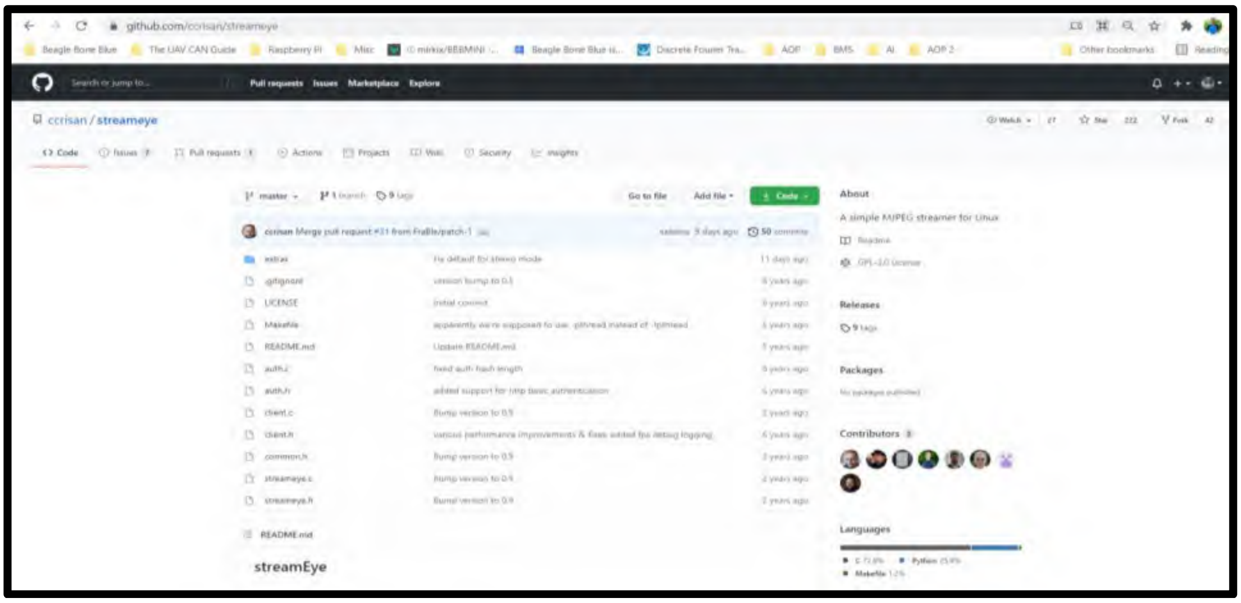
One of the most important aspect of any drone mission is the ability to live stream video feed from an onboard camera. This live feed can then be viewed at ground control station or be utilized by Artificial Intelligence (AI) tools like You Only Look Once (Yolo) for object detection. The AI algorithm can be trained for specific drone mission requirement such as surveillance, rescue, disaster relief or intelligent payload delivery. In this chapter I have illustrated the configuration for creating a streaming server onboard companion computer to live stream video feed over encrypted and secure VPN link. The configuration has been tested for both Raspberry Pi 4 and Jetson Nano. The live feed has been received at Jetson TX2 and processed for object detection using Yolo v4 Tiny. However, training the algorithm for specific dataset has not left out of the scope of this chapter.

11.1 Making video streaming server on companion computer like Raspberry Pi 4/Jetson Nano

Video streaming server on companion computer onboard drone In order to get live stream from drone, we need to create a streaming server on companion computer. The companion computer could be Raspberry Pi 4 or Jetson Nano, the process is same. We need to be careful with the directory paths which will be different for Raspberry Pi 4 or Jetson Nano as per your environment.

<u>Step 1</u>. Install MJPEG server on Raspberry Pi 4 or Jetson Nano. First update, upgrade OS followed by downloading streameye repository from github.

```
sudo apt-get update
sudo apt-get upgrade
git clone https://github.com/ccrisan/streameye.git
```



github.com/ccrisan/streameye

ccrisan / streameye

A simple MJPEG streamer for Linux

```
ccrisan Merge pull request #31 from FraBle/patch-1        9 days ago   50 commits
extras          fix default for stereo mode                                    11 days ago
.gitignore      version bump to 0.1                                            8 years ago
LICENSE         initial commit                                                 9 years ago
Makefile        apparently we're supposed to use -pthread instead of -lpthread 5 years ago
README.md       Update README.md                                               7 years ago
auth.c          fixed auth hash length                                         8 years ago
auth.h          added support for http basic authentication                    6 years ago
client.c        Bump version to 0.9                                            2 years ago
client.h        various performance improvements & fixes added for debug logging 6 years ago
common.h        Bump version to 0.9                                            2 years ago
streameye.c     Bump version to 0.9                                            2 years ago
streameye.h     Bump version to 0.9                                            2 years ago
```

README.md

streamEye

Figure 11.2: Downloading streameye github repository for MJPEG streaming server for

onboard companion computer

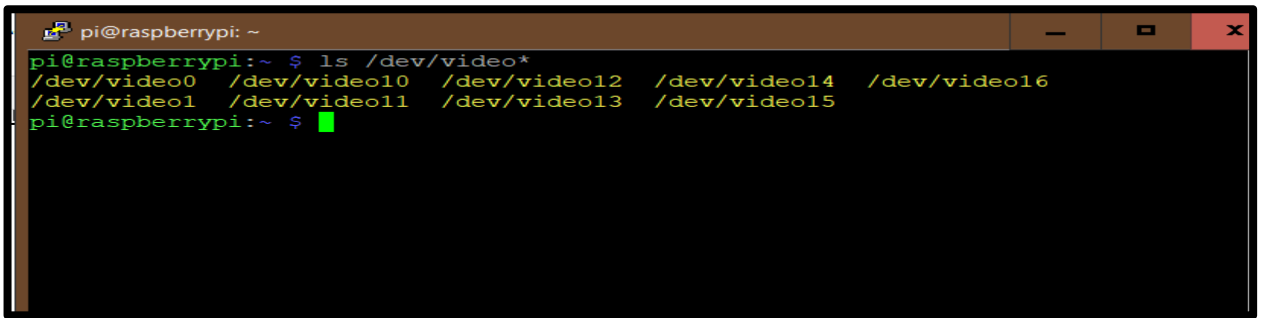
cd streameye

make

sudo make install

If there are no particular drivers required to be installed, all connected video devices / cameras will be displayed by using the following Linux command

ls /dev/video*



```
pi@raspberrypi:~ $ ls /dev/video*
/dev/video0   /dev/video10  /dev/video12  /dev/video14  /dev/video16
/dev/video1   /dev/video11  /dev/video13  /dev/video15
pi@raspberrypi:~ $
```

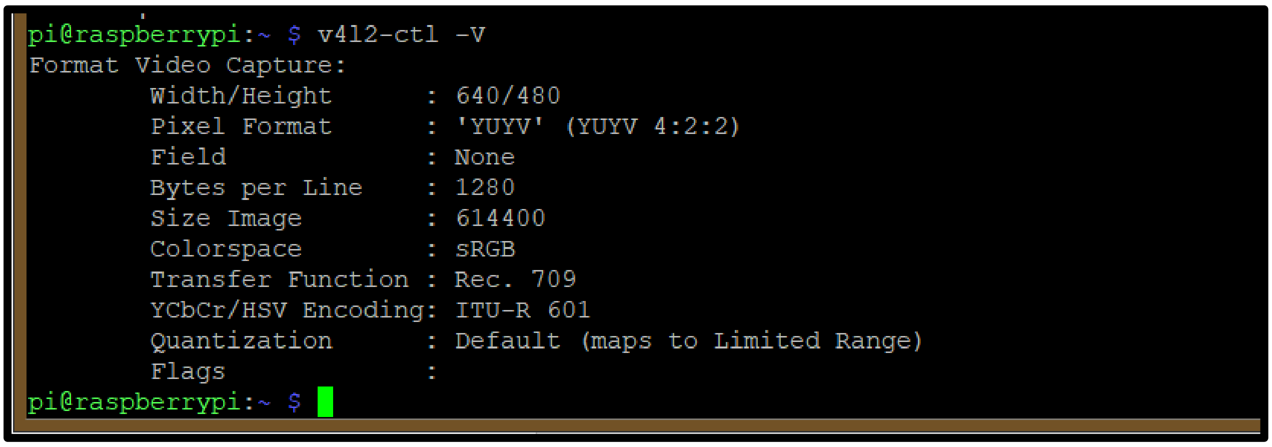
Figure 11.3: Detecting all video devices connected to companion computer

If you are using one of the official camera modules, it is important to do the following so that the camera is displayed immediately (preferably by autostart)

sudo modprobe bcm2835-v4l2

To get camera or HDMI capture card details on Companion Computer,

v4l2-ctl -V



```
pi@raspberrypi:~ $ v4l2-ctl -V
Format Video Capture:
        Width/Height      : 640/480
        Pixel Format      : 'YUYV' (YUYV 4:2:2)
        Field             : None
        Bytes per Line    : 1280
        Size Image        : 614400
        Colorspace        : sRGB
        Transfer Function : Rec. 709
        YCbCr/HSV Encoding: ITU-R 601
        Quantization      : Default (maps to Limited Range)
        Flags             :
pi@raspberrypi:~ $
```

Figure 11.4: Getting camera/capture card device information

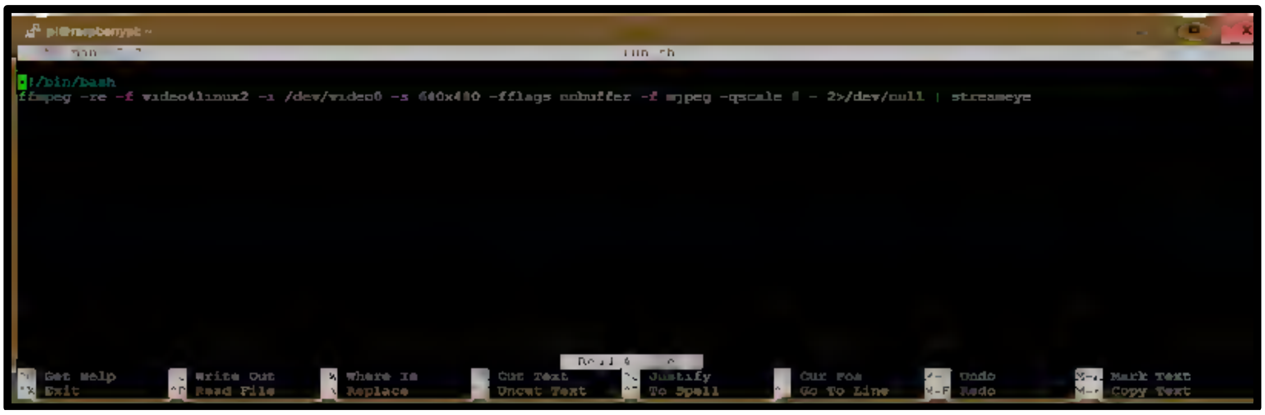
<u>Step 2</u>. Create a script to run the streameye application installed on companion computer with our required configuration.

sudo nano run.sh

Add the below commands

#!/bin/bash

ffmpeg -re -f video4linux2 -i /dev/video0 -s 640x480 -fflags nobuffer -f mjpeg -qscale 8 - 2>/dev/null | streameye

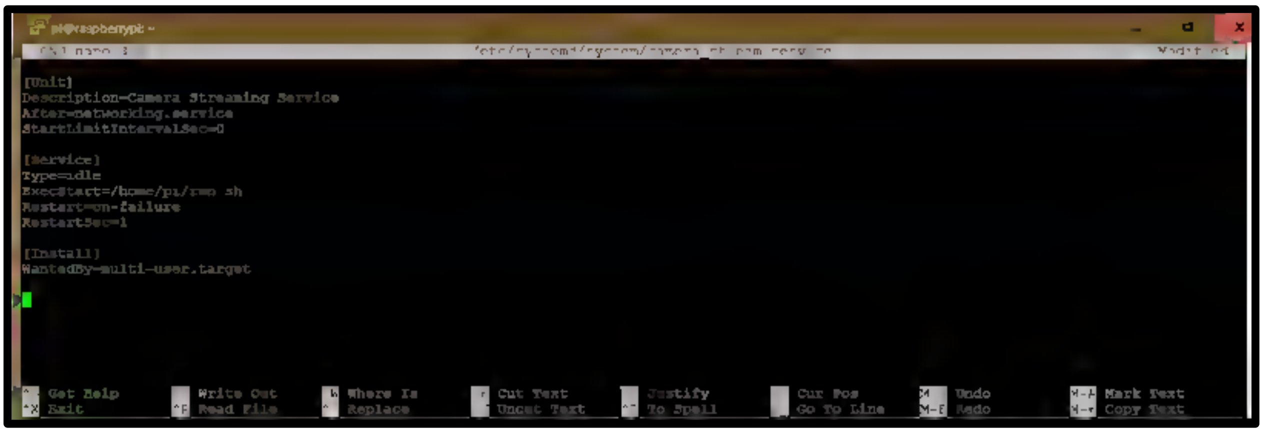


```
#!/bin/bash
ffmpeg -re -f video4linux2 -i /dev/video0 -s 640x480 -fflags nobuffer -f mjpeg -qscale 8 - 2>/dev/null | streameye
```

Figure 11.5: Creating content of run.sh, a script for triggering streameye MJPEG server with particular video streaming options

No we need to create a camera_stream.service to auto run the camera script run.sh on start up. For this use the command

sudo nano /etc/systemd/system/camera_stream.service



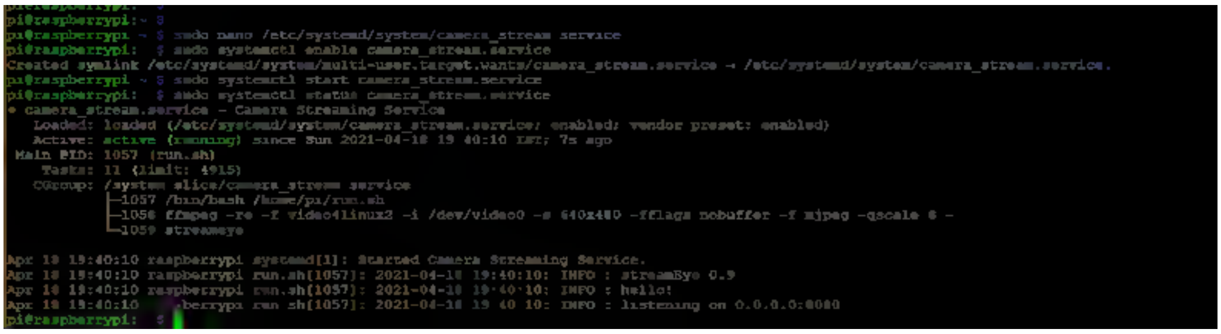
```
[Unit]
Description=Camera Streaming Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
ExecStart=/home/pi/run.sh
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target
```

Figure 11.6: Creating startup service file for running run.sh script automatically on startup using systemd

```
[Unit]
Description=Camera Streaming Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
ExecStart=/home/pi/run.sh
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target
```

Press **Ctrl+O**, enter and then **Ctrl+X** to save and exit. To enable camera service, then start and check status. After status is successful, we are confirmed the service is good to go.

```
sudo systemctl enable camera_stream.service
sudo systemctl start camera_stream.service
sudo systemctl status camera_stream.service
```



```
pi@raspberrypi:~ $
pi@raspberrypi ~ $ sudo nano /etc/systemd/system/camera_stream service
pi@raspberrypi:  $ sudo systemctl enable camera_stream.service
Created symlink /etc/systemd/system/multi-user.target.wants/camera_stream.service → /etc/systemd/system/camera_stream.service.
pi@raspberrypi ~ $ sudo systemctl start camera_stream.service
pi@raspberrypi:  $ sudo systemctl status camera_stream.service
● camera_stream.service - Camera Streaming Service
   Loaded: loaded (/etc/systemd/system/camera_stream.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2021-04-18 19 40:10 IST; 7s ago
 Main PID: 1057 (run.sh)
    Tasks: 11 (limit: 4915)
   CGroup: /system slice/camera_stream service
           ├─1057 /bin/bash /home/pi/run.sh
           ├─1058 ffmpeg -re -f video4linux2 -i /dev/video0 -s 640x480 -fflags nobuffer -f mjpeg -qscale 8 -
           └─1059 streameye

Apr 18 19:40:10 raspberrypi systemd[1]: Started Camera Streaming Service.
Apr 18 19:40:10 raspberrypi run.sh[1057]: 2021-04-18 19:40:10: INFO : streamEye 0.9
Apr 18 19:40:10 raspberrypi run.sh[1057]: 2021-04-18 19:40:10: INFO : hello!
Apr 18 19:40:10    berrypi run sh[1057]: 2021-04-18 19 40 10: INFO : listening on 0.0.0.0:8080
pi@raspberrypi:  $
```

Figure 11.7: camera_streaming.service successfully created, enabled and started with running status

11.2 Receiving video feed on ground control station application i.e Mission Planner

Receiving video feed on ground station i.e., Mission Planner

To view feed in Mission Planner, we need to firs install GStreamer version 1.14 for windows msi package. Then open Mission Planner, **right click on Mission Planner panel > video > Set GStreamer Source** and add the link

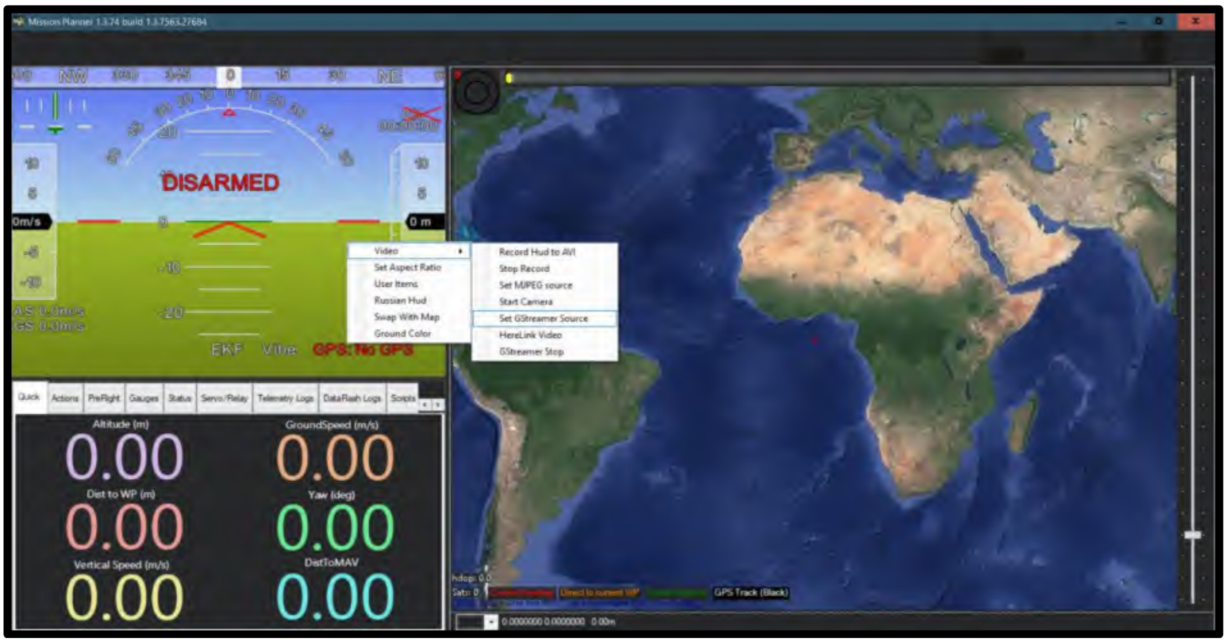
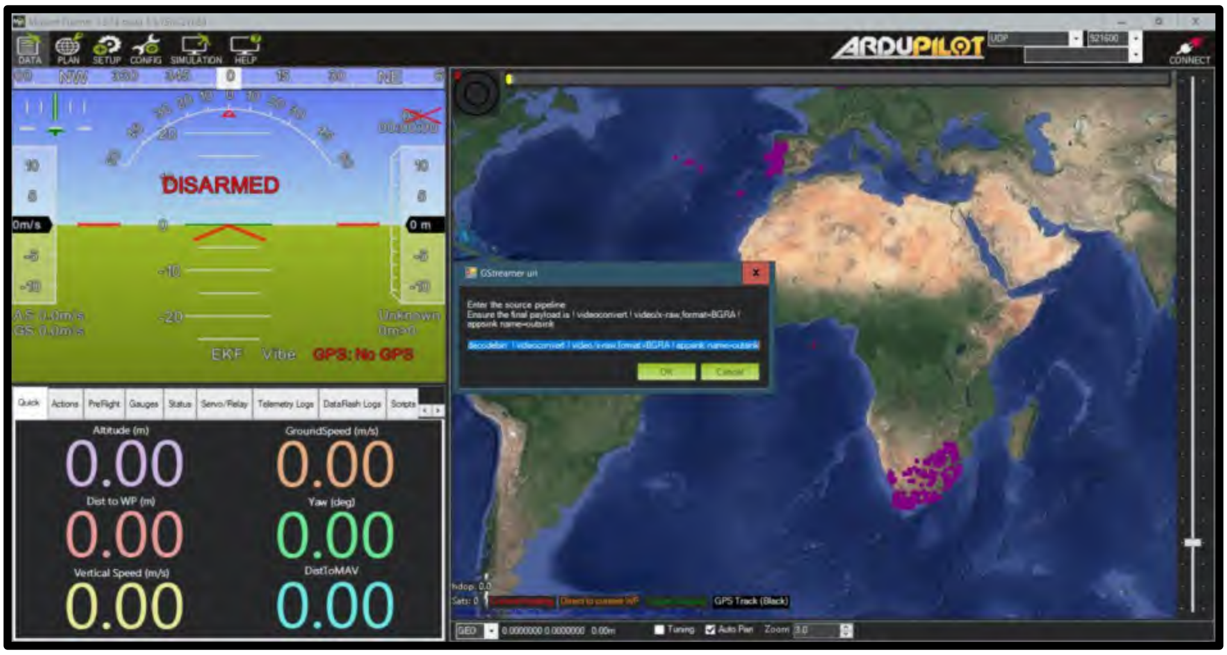


Figure 11.8: Importing video feed directly into Mission Planner GCS application
Add this link

souphttpsrc location=http://100.96.1.66:8080/mjpeg do-timestamp=true ! decodebin ! videoconvert ! video/x-raw,format=BGRA ! appsink name=outsink



Enter the source pipeline
Ensure the final payload is ! videoconvert ! video/x-raw,format=BGRA ! appsink name=outsink

decodebin ! videoconvert ! video/x-raw,format=BGRA ! appsink name=outsink

Figure 11.9: Importing video feed directly into Mission Planner GCS application

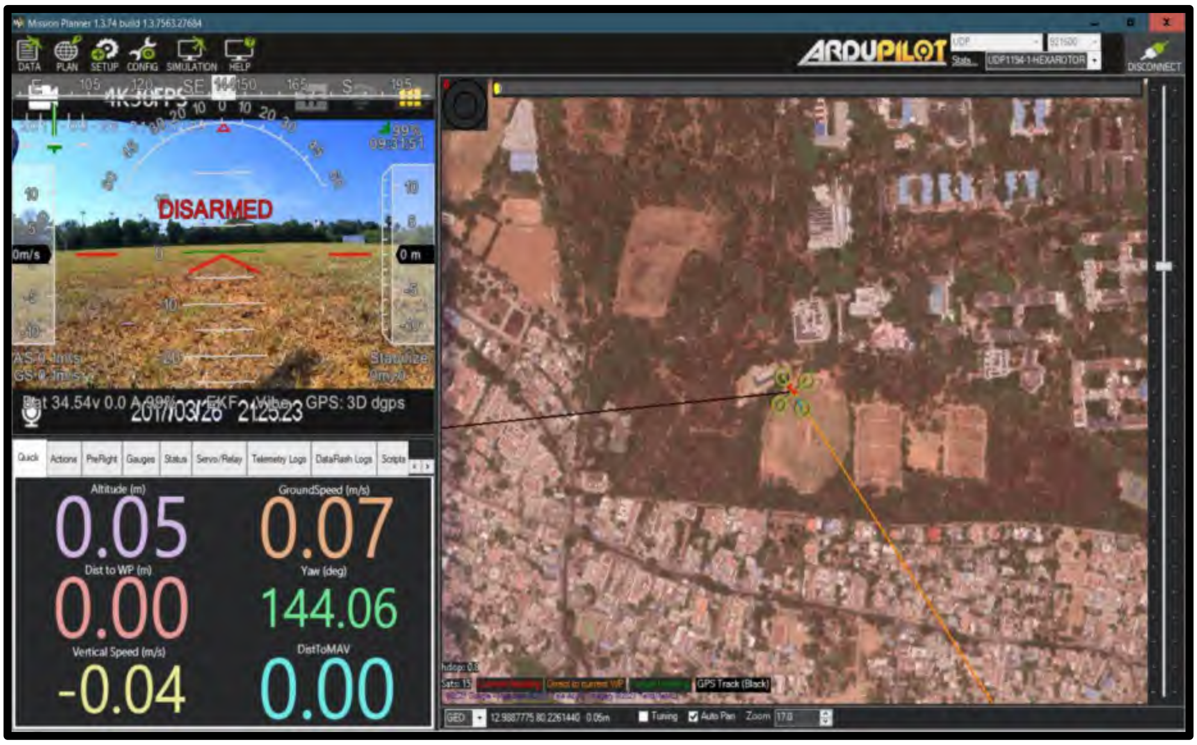


Figure 11.10: Successfully imported video feed directly into Mission Planner GCS application

The same feed is simultaneously available through browser. Open browser on ground station connected on OpenVPN. Type **http://100.96.1.66:8080/** to view live feed. If on local network then put local IP of Raspberry Pi 4 else VPN IP on VPN network or Public IP incase it's on open network.

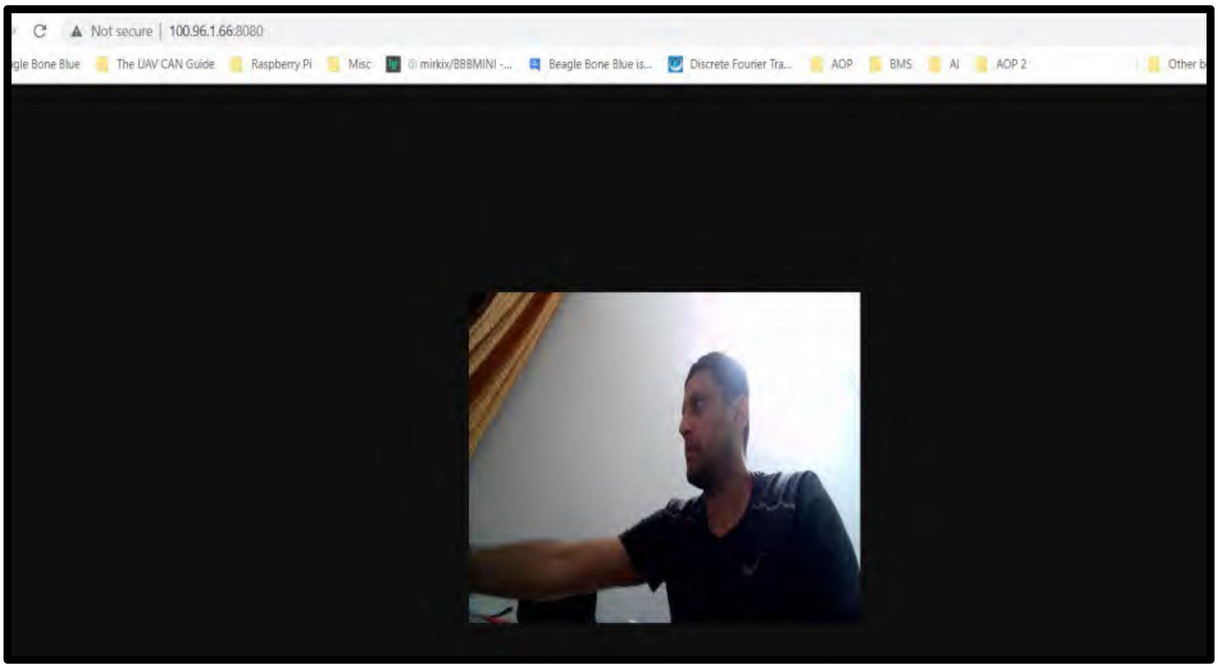


Figure 11.11: Video feed available in browser on any PC/Laptop connected to private VPN

11.3 Preparing Jetson TX2 with latest Ubuntu 16.04 LTS and Jetpack 4.5.1

Jetson TX2 is the computer on ground. It will receive the video feed from drone on VPN and do object detection on it using Yolov4 Darknet framework. Installing latest Jetpack 4.5.1 on JetsonTX2 to install all the pre-requisites. The list of pre-requisites required for YoloV4 are: -

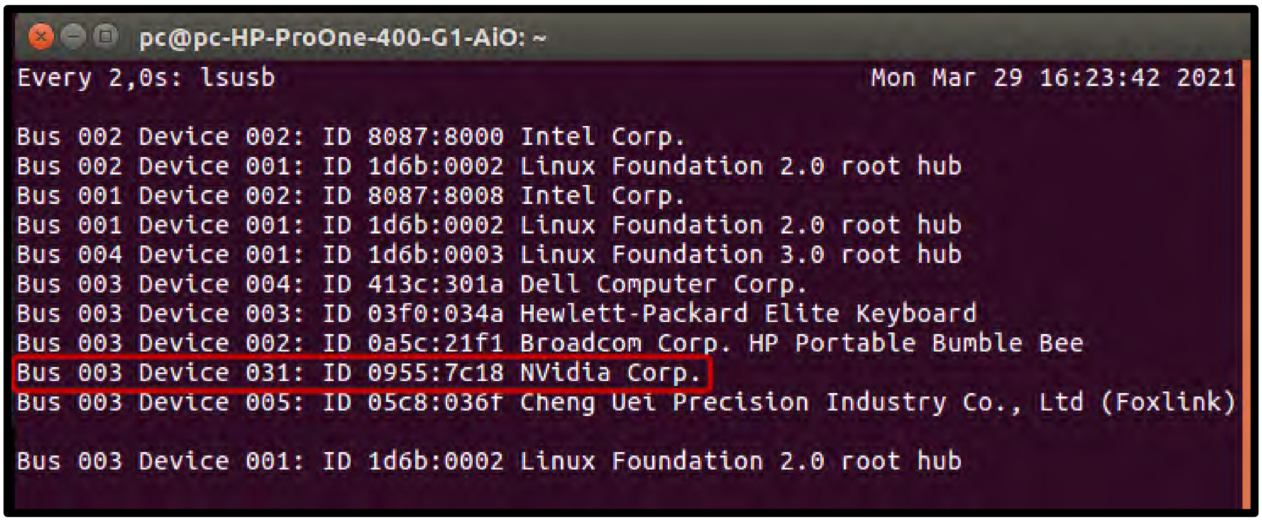
- **CMake >= 3.18**
- **CUDA >= 10.2**
- **OpenCV >= 2.4**
- **cuDNN >= 8.0.2**
- **GPU with CC >= 3.0**

Jetpack 4.5.1 installs all these prerequisites as a package. We will refresh the operating System Ubuntu 16.04 LTS with Jetpack 4.5.1 on Jetson TX2. We require a host Linux machine with 8 Gb RAM minimum running minimum Ubuntu 16.04 LTS. I am using my dual boot Laptop to do the same. Connect the Jetson TX2 to your laptop using micro-USB cable, while power to Jetson TX2 is provided by the power adaptor. Install Nvidia SDK Manager on Linux host machine.

Step 1 Power the Jetson TX2 with power adaptor and connect to Linux Laptop (Host Machine) over micro-USB cable.

- press power button then wait for boot led lights up.
- press reset & recovery buttons together
- release reset button
- and release the recovery button after 3 seconds later.

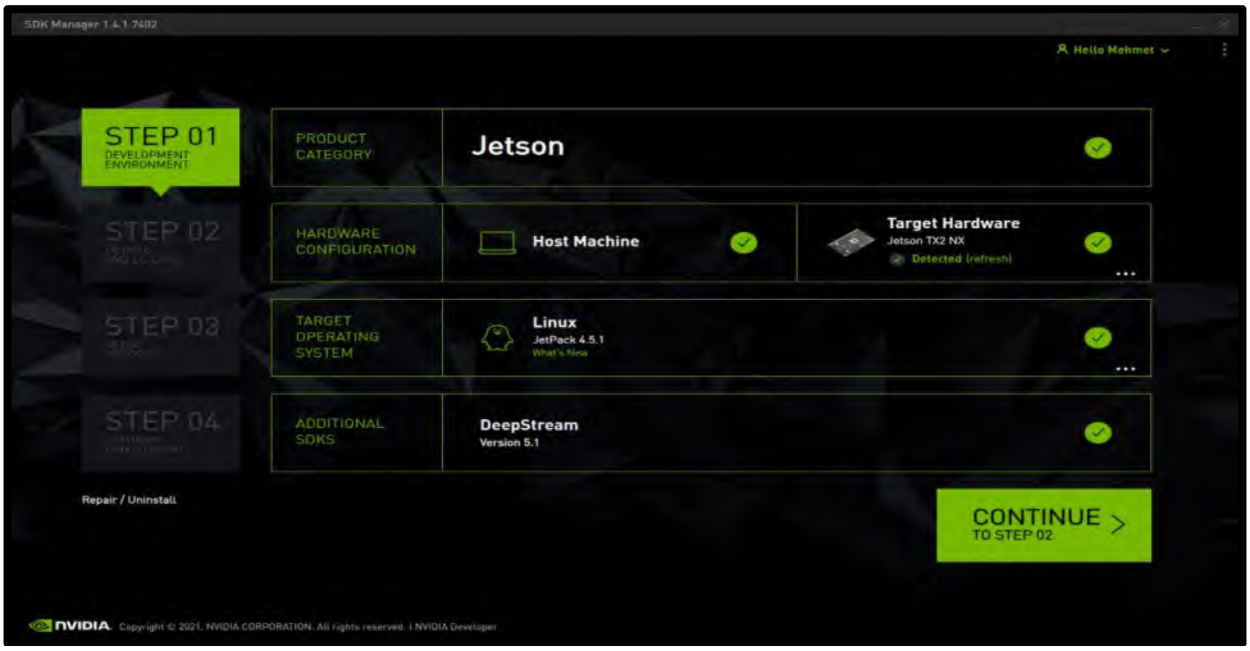
Step 2 Open terminal window on the host machine running Linux. Use command lsusb and you should see *"NVidia Corp."* title in the terminal.



```
Every 2,0s: lsusb                                      Mon Mar 29 16:23:42 2021

Bus 002 Device 002: ID 8087:8000 Intel Corp.
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 8087:8008 Intel Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 004: ID 413c:301a Dell Computer Corp.
Bus 003 Device 003: ID 03f0:034a Hewlett-Packard Elite Keyboard
Bus 003 Device 002: ID 0a5c:21f1 Broadcom Corp. HP Portable Bumble Bee
Bus 003 Device 031: ID 0955:7c18 NVidia Corp.
Bus 003 Device 005: ID 05c8:036f Cheng Uei Precision Industry Co., Ltd (Foxlink)

Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Figure 11.12: Detecting Jetson TX2 connected to Laptop/PC running Linux Ubuntu ready for configuration

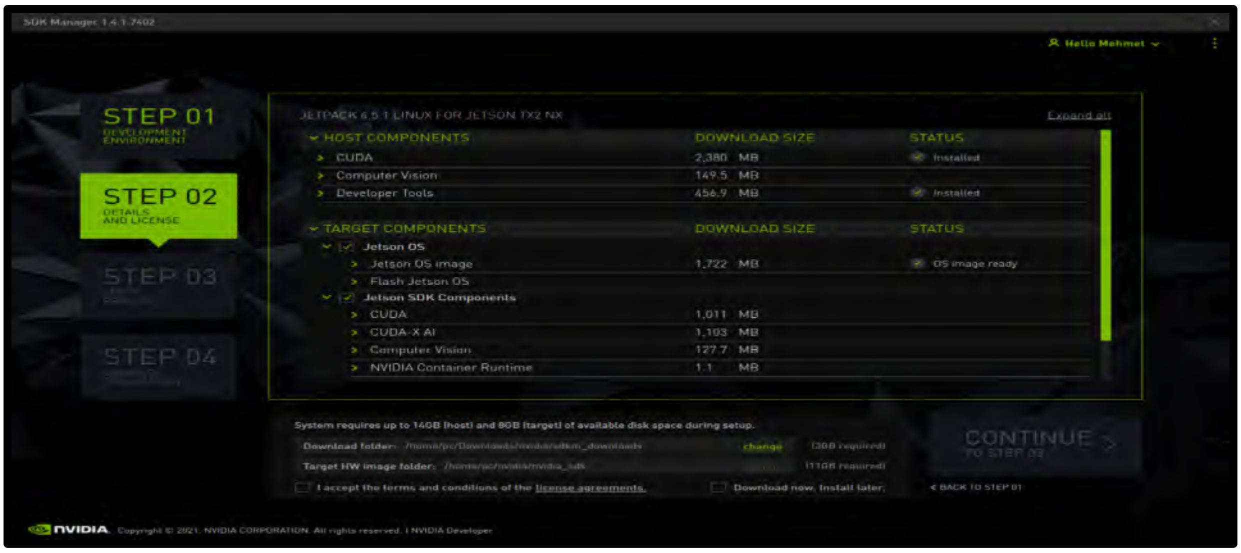
Step 3 Next, we need to set configurations in the Nvidia SDK Manager. We are required to set up the SDK Manager by making an account. Target Hardware must be autodetected as your Jetson TX2. Accept the license agreement and continue.


SDK Manager 1.4.1.7402
Hello Mehmet
STEP 01 DEVELOPMENT ENVIRONMENT
STEP 02
STEP 03
STEP 04
Repair / Uninstall
CONTINUE TO STEP 02
NVIDIA Copyright © 2021, NVIDIA CORPORATION. All rights reserved. | NVIDIA Developer

PRODUCT CATEGORY: Jetson
HARDWARE CONFIGURATION: Host Machine; Target Hardware Jetson TX2 NX Detected (refresh)
TARGET OPERATING SYSTEM: Linux JetPack 4.5.1 What's New
ADDITIONAL SDKS: DeepStream Version 5.1

Figure 11.13: Using Nvidia SDK Manager application running on host Linux PC/Laptop to configure Jetson TX2 via micro USB cable

Continue to next step


SDK Manager 1.4.1.7402

Hello Mehmet

JETPACK 4.5.1 LINUX FOR JETSON TX2 NX — Expand all

HOST COMPONENTS	DOWNLOAD SIZE	STATUS
CUDA	2,380 MB	Installed
Computer Vision	149.5 MB	
Developer Tools	456.9 MB	Installed

TARGET COMPONENTS	DOWNLOAD SIZE	STATUS
✓ Jetson OS		
Jetson OS image	1,722 MB	OS image ready
Flash Jetson OS		
✓ Jetson SDK Components		
CUDA	1,011 MB	
CUDA-X AI	1,103 MB	
Computer Vision	127.7 MB	
NVIDIA Container Runtime	1.1 MB	

System requires up to 14GB (host) and 8GB (target) of available disk space during setup.

Download folder: /home/pc/Downloads/nvidia/sdkm_downloads change (30GB required)

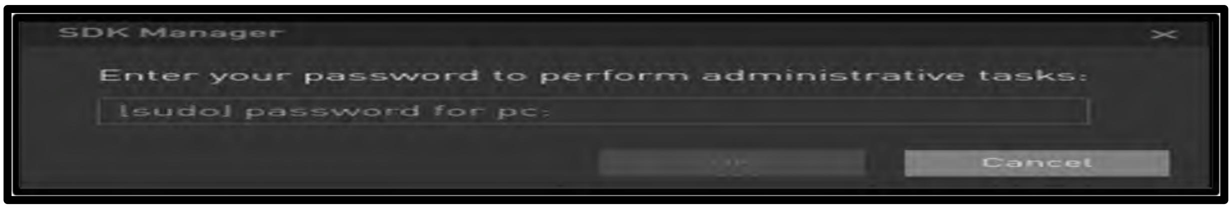
Target HW image folder: /home/pc/nvidia/nvidia_sdk (11GB required)

☐ I accept the terms and conditions of the license agreements. ☐ Download now. Install later.

NVIDIA. Copyright © 2021, NVIDIA CORPORATION. All rights reserved. | NVIDIA Developer

Figure 11.14: Installing Ubuntu 16.04 as Operating system with Jetpack 4.5.1

Step 4 Nvidia SDK Manager will for password to complete installation. Provide password to continue.

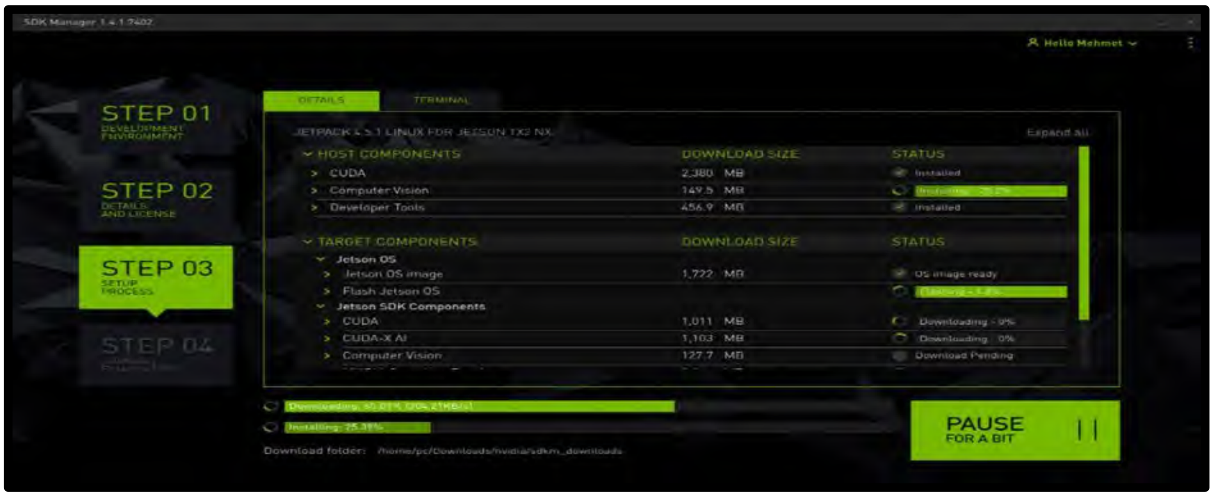

SDK Manager

Enter your password to perform administrative tasks:

[sudo] password for pc:

Figure 11.15: Authorization of superuser/administrator

Step 5 After all packages have been downloaded, Jetson OS will be installed on the Jetson TX2.


SDK Manager 1.4.1.7402

Hello Mehmet

JETPACK 4.5.1 LINUX FOR JETSON TX2 NX — Expand all

HOST COMPONENTS	DOWNLOAD SIZE	STATUS
CUDA	2,380 MB	Installed
Computer Vision	149.5 MB	Installing - 25.2%
Developer Tools	456.9 MB	Installed

TARGET COMPONENTS	DOWNLOAD SIZE	STATUS
Jetson OS		
Jetson OS image	1,722 MB	OS image ready
Flash Jetson OS		Flashing - 1.4%
Jetson SDK Components		
CUDA	1,011 MB	Downloading - 0%
CUDA-X AI	1,103 MB	Downloading - 0%
Computer Vision	127.7 MB	Download Pending

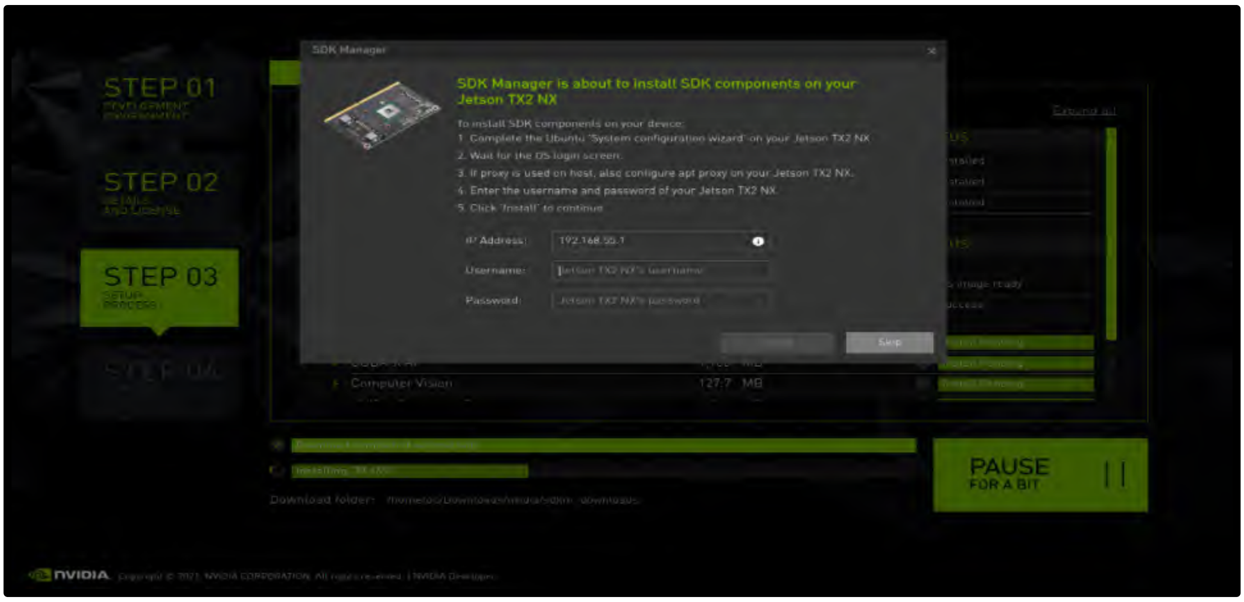
Downloading: 50.01% (204.21KB/s)

Installing: 25.39%

Download folder: /home/pc/Downloads/nvidia/sdkm_downloads

Figure 11.16: Downloading OS and Jetpack on Jetson TX2

Step 6 The SDK Manager asks your Jetson TX2 NX's username and password. Need to connect monitor and keyboard to Jetson TX2 and complete step 7 and then proceed here again.



SDK Manager

SDK Manager is about to install SDK components on your Jetson TX2 NX

To install SDK components on your device:

1. Complete the Ubuntu 'System configuration wizard' on your Jetson TX2 NX
2. Wait for the OS login screen.
3. If proxy is used on host, also configure apt proxy on your Jetson TX2 NX.
4. Enter the username and password of your Jetson TX2 NX.
5. Click 'Install' to continue

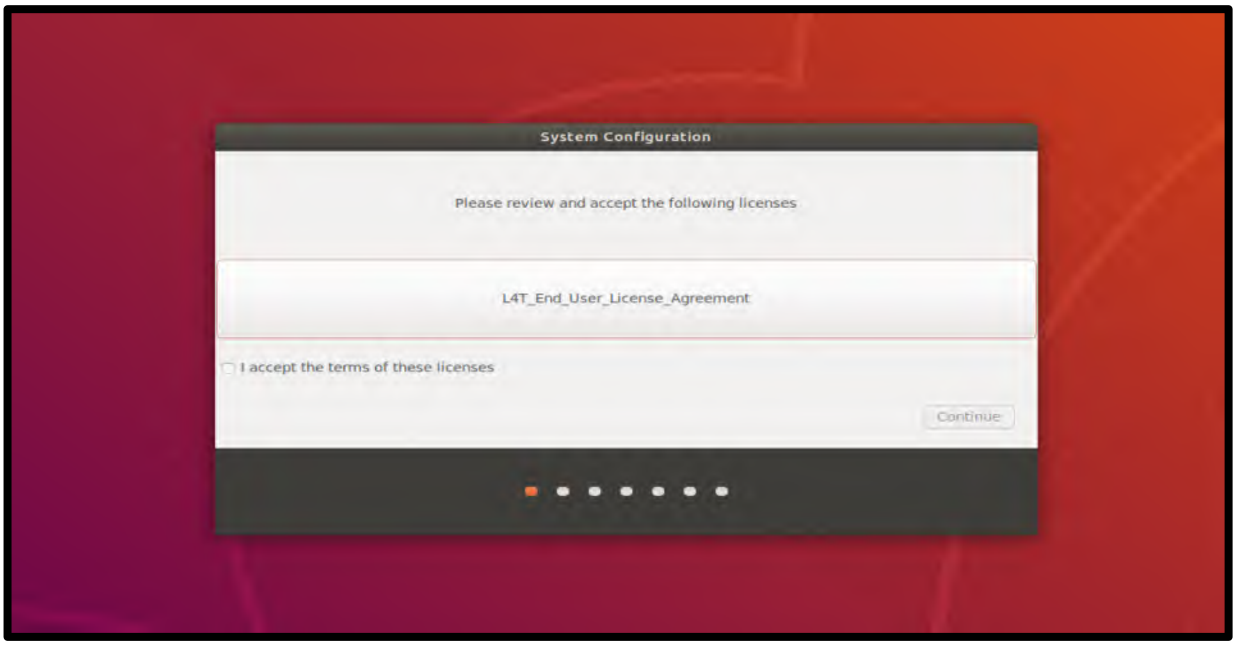
IP Address: 192.168.55.1

Username: Jetson TX2 NX's username

Password: Jetson TX2 NX's password

Figure 11.17: Login credentials of Jetson TX2 to proceed

Step 7 Complete the SDK Manager installation progress. Configure your Ubuntu installation progress (language, keyboard type, location, username & password etc.).



System Configuration

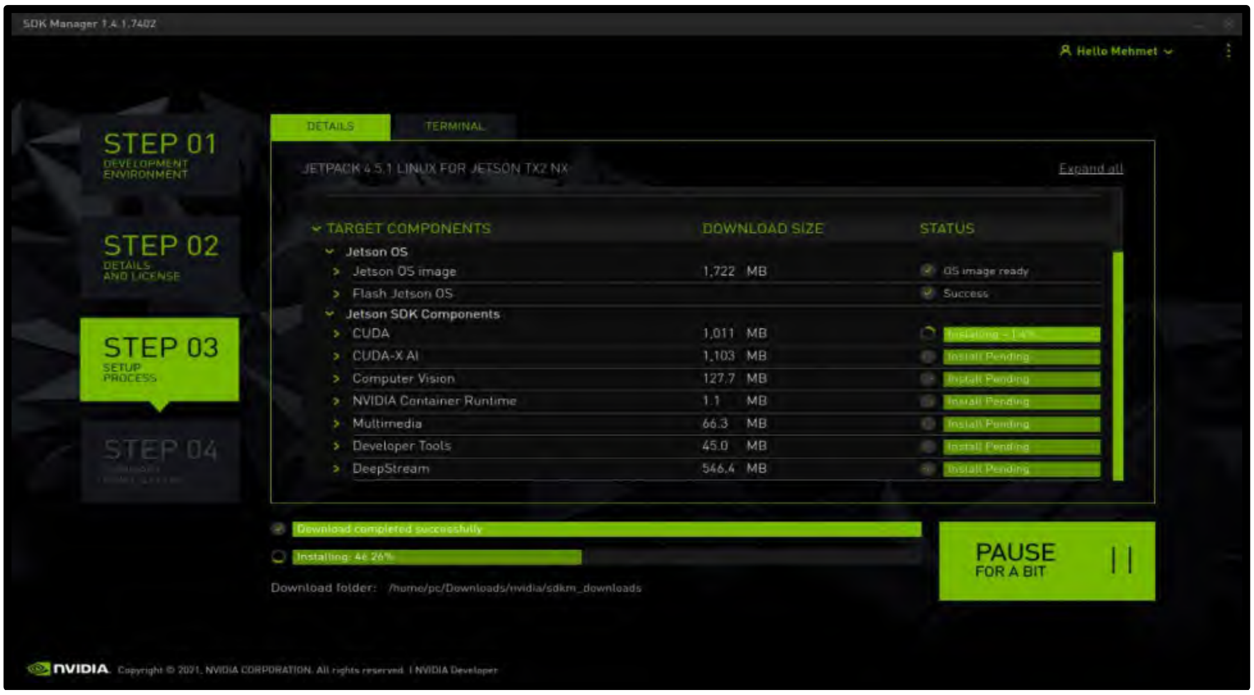
Please review and accept the following licenses

L4T_End_User_License_Agreement

I accept the terms of these licenses

Figure 11.18: Configuring user profile on Ubuntu 16.04 installed on Jetson TX2

Step 8 Type your username and password in SDK Manager then click "Install".



JETPACK 4.5.1 LINUX FOR JETSON TX2 NX

TARGET COMPONENTS	DOWNLOAD SIZE	STATUS
Jetson OS		
Jetson OS image	1,722 MB	OS image ready
Flash Jetson OS		Success
Jetson SDK Components		
CUDA	1,011 MB	Installing - 14%
CUDA-X AI	1,103 MB	Install Pending
Computer Vision	127.7 MB	Install Pending
NVIDIA Container Runtime	1.1 MB	Install Pending
Multimedia	66.3 MB	Install Pending
Developer Tools	45.0 MB	Install Pending
DeepStream	546.4 MB	Install Pending

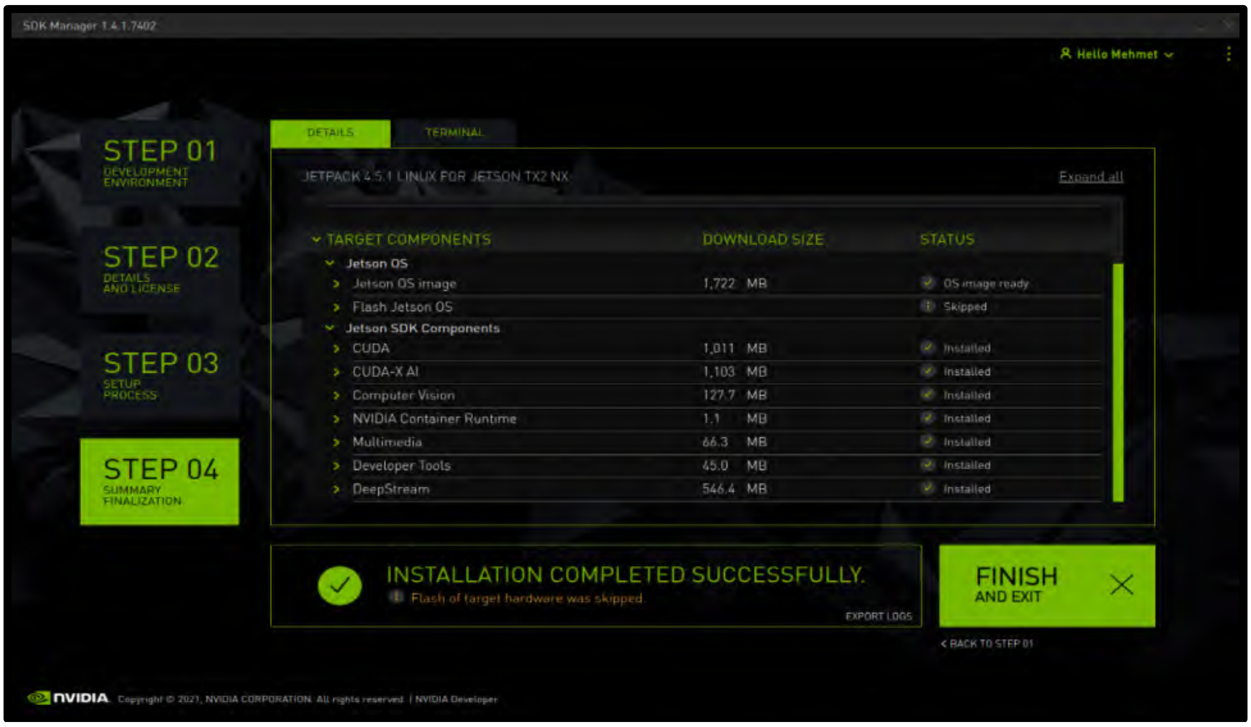
Download completed successfully

Installing: 46.26%

Download folder: /home/pc/Downloads/nvidia/sdkm_downloads

Copyright © 2021, NVIDIA CORPORATION. All rights reserved. | NVIDIA Developer

Figure 11.19: Downloading remaining packages



JETPACK 4.5.1 LINUX FOR JETSON TX2 NX

TARGET COMPONENTS	DOWNLOAD SIZE	STATUS
Jetson OS		
Jetson OS image	1,722 MB	OS image ready
Flash Jetson OS		Skipped
Jetson SDK Components		
CUDA	1,011 MB	Installed
CUDA-X AI	1,103 MB	Installed
Computer Vision	127.7 MB	Installed
NVIDIA Container Runtime	1.1 MB	Installed
Multimedia	66.3 MB	Installed
Developer Tools	45.0 MB	Installed
DeepStream	546.4 MB	Installed

INSTALLATION COMPLETED SUCCESSFULLY.

Flash of target hardware was skipped.

Copyright © 2021, NVIDIA CORPORATION. All rights reserved. | NVIDIA Developer

Figure 11.20: Ubuntu 16.04 with Jetpack 4.5.1 successfully installed

Now we have a fresh copy of Ubuntu 16.04 LTS with Jetpack 4.5.1 ready for installation of Yolov4 Darknet framework.

11.4 Installing VPN as a service to make Jetson TX2 as part of our private VPN

Now we need to install VPN to receive video feed on the private VPN from onboard companion computer. The process is similar to the one done in Chapter III above.

Step 1 First install Open VPN application on Jetson Nano.

sudo apt-get install openvpn

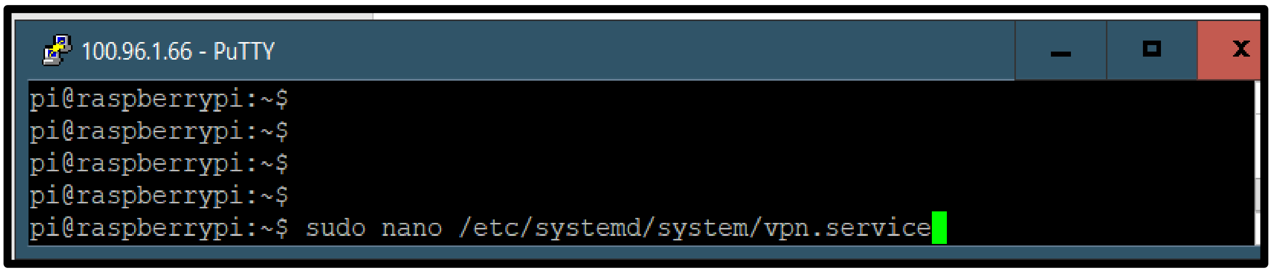
Use WinSCP application on windows to transport the folder VPN_Folder that contains all my .ovpn files. Let me remind you that the VPN_Folder contains .ovpn files for FCU, Companion computer and Ground Control Station (GCS). For doing this Artificial Intelligence based object detection, I created another profile for Jetson TX2 and named it **jetsontx2.ovpn**. Rest the process remains the same as shown in Chapter 4 of how to transport the .ovpn file. Again, I have transported the entire folder VPN_Folder first to default location using WinSCP. It allows transportation of files/folders from windows to Linux. Then using below mentioned command, I will move folder to location of ovpn application for execution.

Now move the folder from /home/yash to /etc/openvpn for execution

sudo mv /home/yash/VPN_Folder /etc/openvpn/

Step 2 Create systemd file for startup at boot

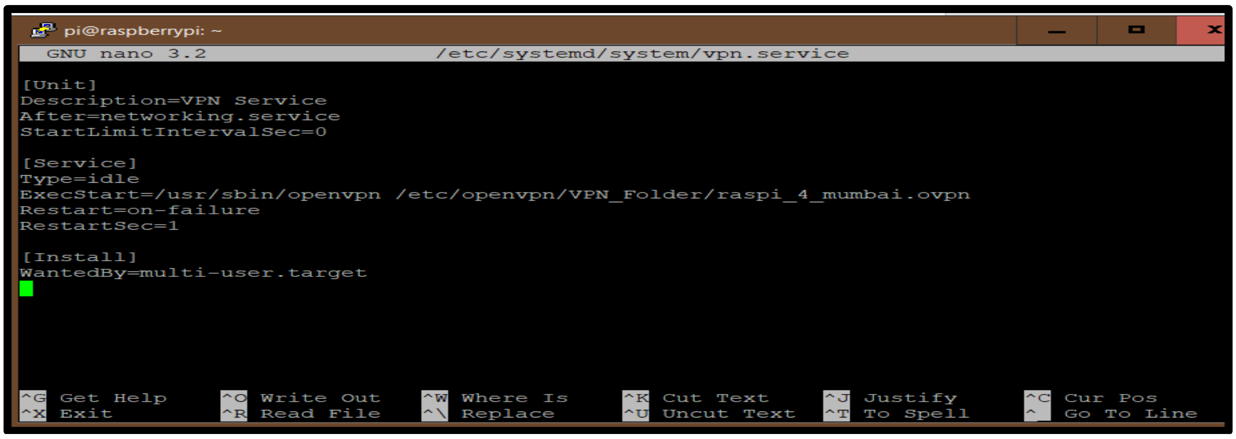
sudo nano /etc/systemd/system/vpn.service



```
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo nano /etc/systemd/system/vpn.service
```

Figure 11.21: Creating vpn.service at /etc/systemd/system for running VPN as a service on startup

```
[Unit]
Description=VPN Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
ExecStart=/usr/sbin/openvpn /etc/openvpn/VPN_Folder/jetson.ovpn
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target
```



```
pi@raspberrypi: ~
  GNU nano 3.2                    /etc/systemd/system/vpn.service

[Unit]
Description=VPN Service
After=networking.service
StartLimitIntervalSec=0

[Service]
Type=idle
ExecStart=/usr/sbin/openvpn /etc/openvpn/VPN_Folder/raspi_4_mumbai.ovpn
Restart=on-failure
RestartSec=1

[Install]
WantedBy=multi-user.target

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Figure 11.22: Creating content of vpn.service startup file

```
sudo systemctl enable vpn.service
sudo systemctl start vpn.service
sudo systemctl status vpn.service
```

sudo systemctl daemon-reload (only to reload service incase some changes are made)

11.5 Using Darknet framework and YOLO v4 artificial intelligence algorithm for detecting objects on live camera feed using Jetson TX2 located at ground station

Installation of Darknet framework Now after Drone and Jetson TX2 both are connected to VPN (internet), we need to install the Darknet framework to do object detection on live video stream. The installation process is as below:-

Step 1 Git clone the darknet repository from github

[**https://github.com/AlexeyAB/darknet.git**](https://github.com/AlexeyAB/darknet.git)

Step 2. Move inside Darknet folder

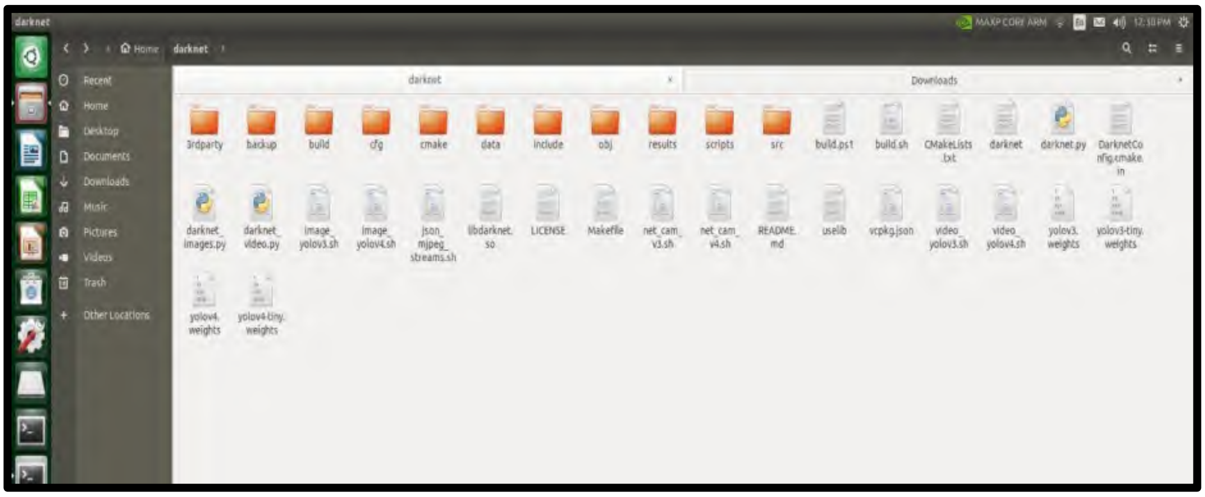
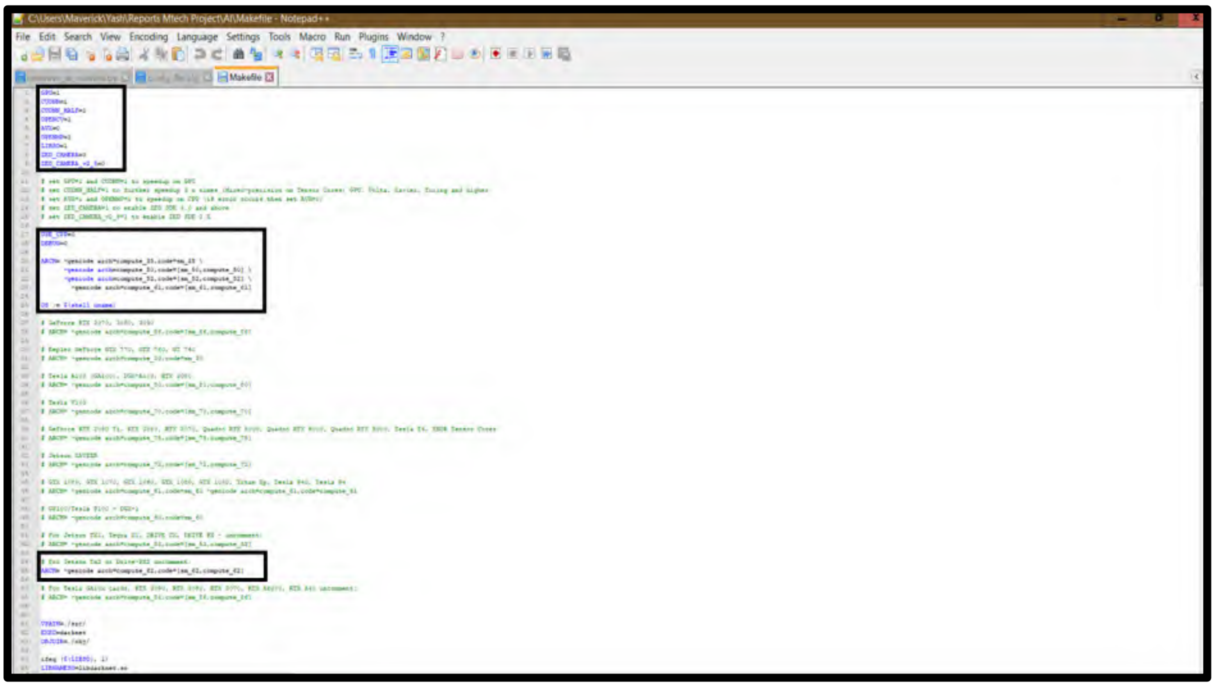


Figure 11.23: Darknet folder containing YOLO weights and other configuration file for Artificial Intelligence implementation on Jetson TX2


C:\Users\Maverick\Yash\Reports Mtech Project\AI\Makefile - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Makefile

```
GPU=1
CUDNN=1
CUDNN_HALF=1
OPENCV=1
AVX=0
OPENMP=1
LIBSO=1
ZED_CAMERA=0
ZED_CAMERA_v2_8=0

# set GPU=1 and CUDNN=1 to speedup on GPU
# set CUDNN_HALF=1 to further speedup 3 x times (Mixed-precision on Tensor Cores) GPU: Volta, Xavier, Turing and higher
# set AVX=1 and OPENMP=1 to speedup on CPU (if error occurs then set AVX=0)
# set ZED_CAMERA=1 to enable ZED SDK 3.0 and above
# set ZED_CAMERA_v2_8=1 to enable ZED SDK 2.X

USE_CPP=0
DEBUG=0

ARCH= -gencode arch=compute_35,code=sm_35 \
      -gencode arch=compute_50,code=[sm_50,compute_50] \
      -gencode arch=compute_52,code=[sm_52,compute_52] \
        -gencode arch=compute_61,code=[sm_61,compute_61]

OS := $(shell uname)

# GeForce RTX 3070, 3080, 3090
# ARCH= -gencode arch=compute_86,code=[sm_86,compute_86]

# Kepler GeForce GTX 770, GTX 760, GT 740
# ARCH= -gencode arch=compute_30,code=sm_30

# Tesla A100 (GA100), DGX-A100, RTX 3080
# ARCH= -gencode arch=compute_80,code=[sm_80,compute_80]

# Tesla V100
# ARCH= -gencode arch=compute_70,code=[sm_70,compute_70]

# GeForce RTX 2080 Ti, RTX 2080, RTX 2070, Quadro RTX 8000, Quadro RTX 6000, Quadro RTX 5000, Tesla T4, XNOR Tensor Cores
# ARCH= -gencode arch=compute_75,code=[sm_75,compute_75]

# Jetson XAVIER
# ARCH= -gencode arch=compute_72,code=[sm_72,compute_72]

# GTX 1080, GTX 1070, GTX 1060, GTX 1050, GTX 1030, Titan Xp, Tesla P40, Tesla P4
# ARCH= -gencode arch=compute_61,code=sm_61 -gencode arch=compute_61,code=compute_61

# GP100/Tesla P100 - DGX-1
# ARCH= -gencode arch=compute_60,code=sm_60

# For Jetson TX1, Tegra X1, DRIVE CX, DRIVE PX - uncomment:
# ARCH= -gencode arch=compute_53,code=[sm_53,compute_53]

# For Jetson Tx2 or Drive-PX2 uncomment:
ARCH= -gencode arch=compute_62,code=[sm_62,compute_62]

# For Tesla GA10x cards, RTX 3090, RTX 3080, RTX 3070, RTX A6000, RTX A40 uncomment:
# ARCH= -gencode arch=compute_86,code=[sm_86,compute_86]


VPATH=./src/
EXEC=darknet
OBJDIR=./obj/

ifeq ($(LIBSO), 1)
LIBNAMESO=libdarknet.so
```

Figure 11.24: Changes to Makefile for make install on Jetson TX2

You can amend the Makefile either through GUI by connecting keyboard and monitor to the Jetson TX2 or using remote ssh.

Incase doing through terminal window or remote ssh of Jetson TX2, use below commands. I have made changes to Makefile as per my hardware Jetson TX2 running Arch Linux. Need to amend as per hardware used.

cd darknet
sudo nano Makefile
Amend the Makefile as above and then **Ctrl+O and Ctrl+X** to save and exit.

Step 3. After making changes, to the Makefile, we need to make install it. Need to do this in terminal window or through remote ssh. We are still in darknet directory.

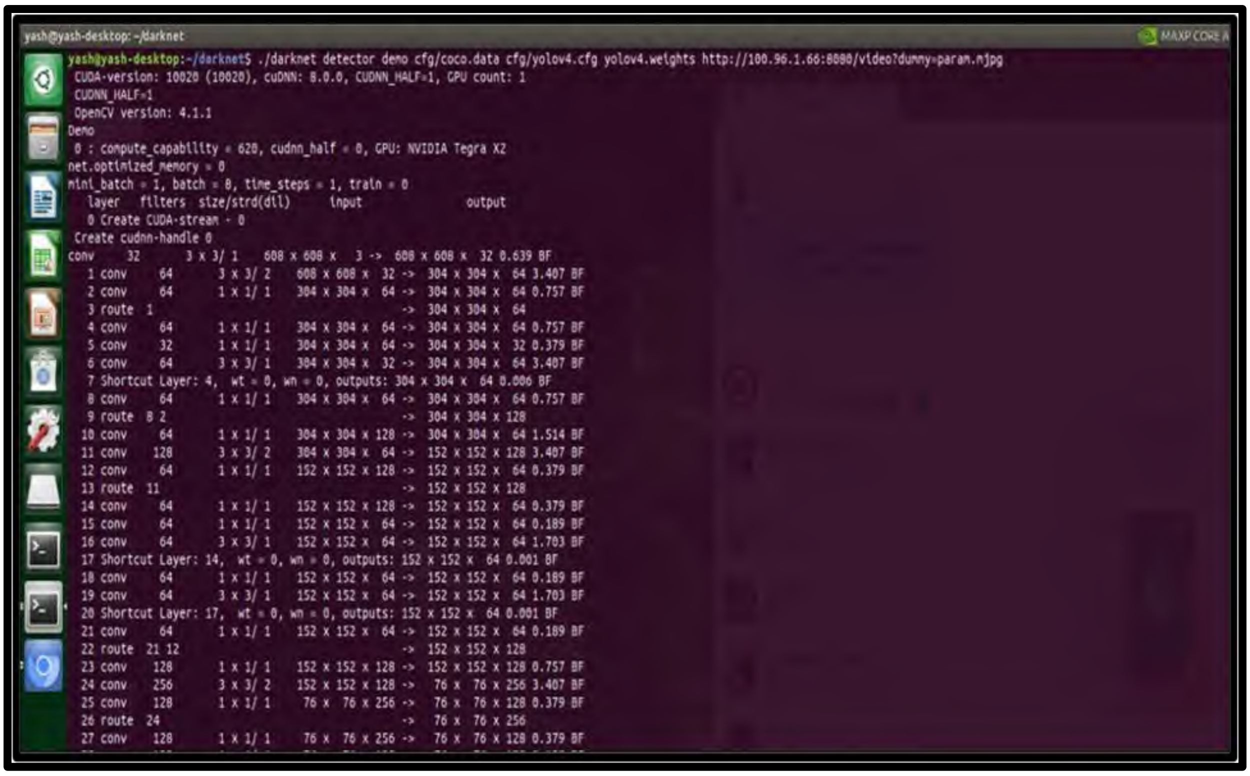
make

Step 4. If all goes well then, we are ready to test live video feed for image detection. First make sure companion computer is on and connected to onboard camera. It is accessible on VPN from Jetson TX2 over Internet. Jetson TX2 being on ground can utilize any type of internet connection from OFC based to 4G LTE Dongle as per the availability. Also, connecting monitor and keyboard to Jetson TX2 is suggested over taking remote ssh from GCS laptop. Open browser in Jetson TX2 and check availability of video feed from Companion computer on url **http://100.96.1.66:8080** You must put the IP Address of your companion computer (VPN or local LAN)

Step 5. Testing live video feed for object detection using YOLO version 4. Open terminal window.

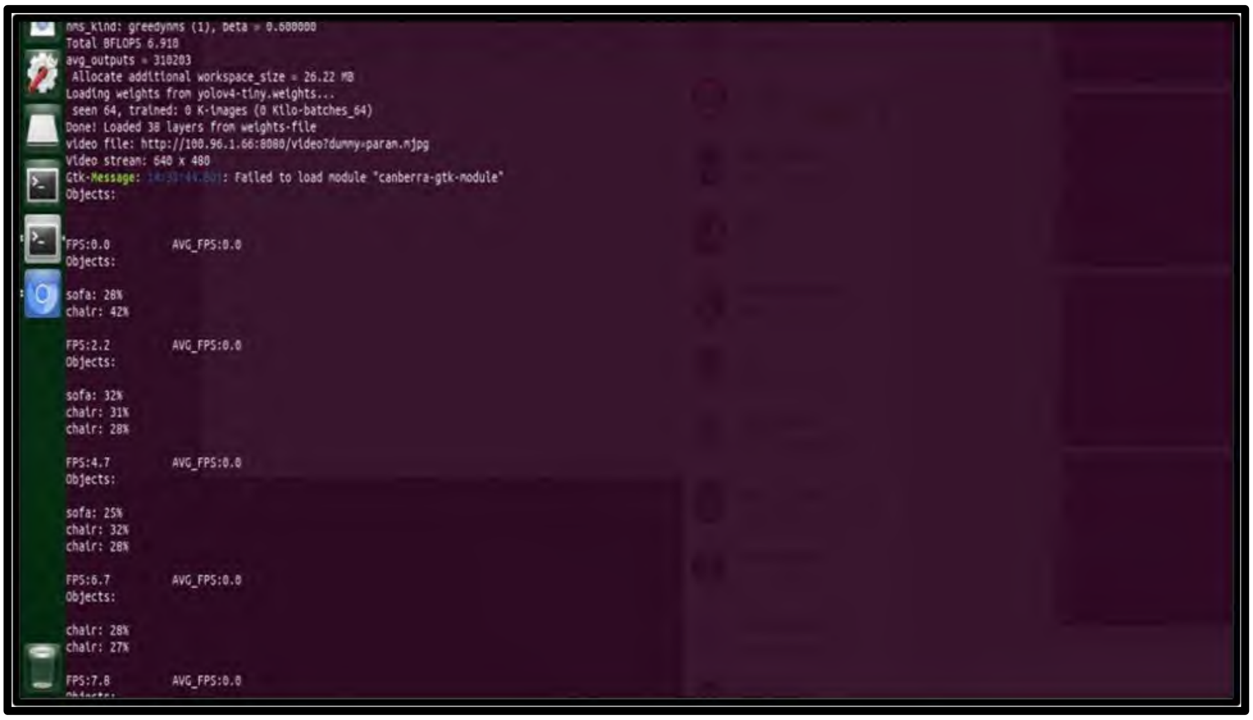
If we want to use YOLOv4

cd darknet
./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights
http://100.96.1.66:8080/video?dummy=param.mjpg



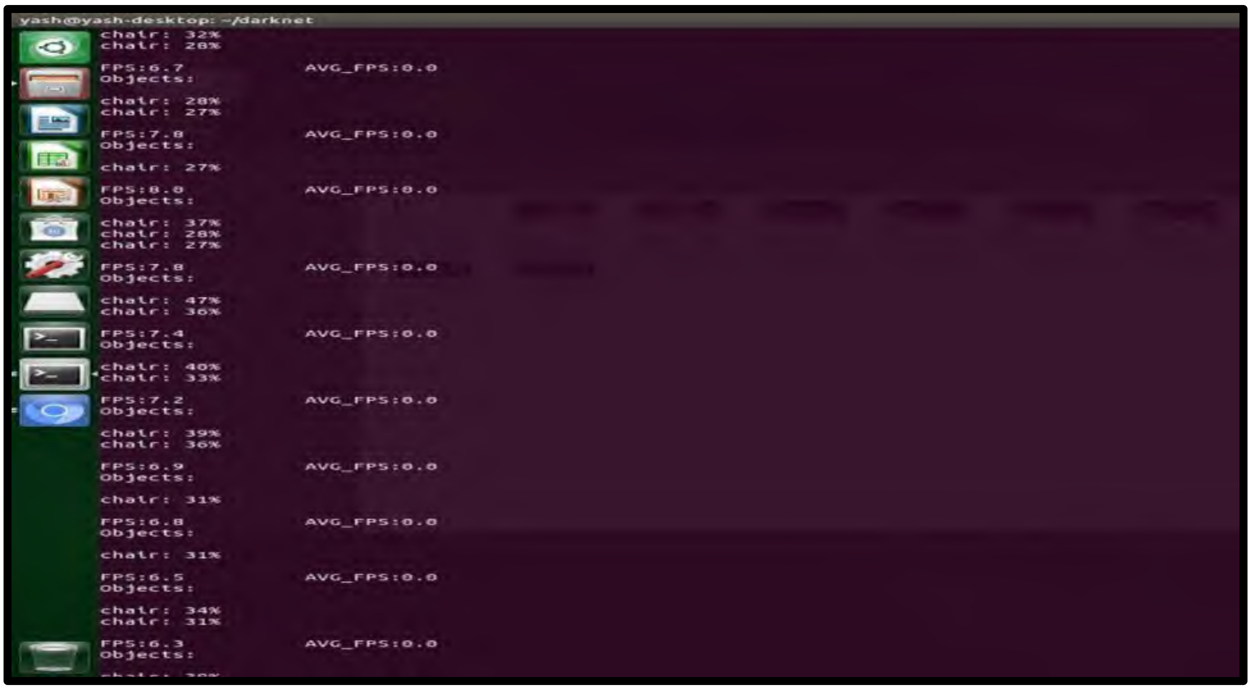
```
yash@yash-desktop: ~/darknet
yash@yash-desktop:~/darknet$ ./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights http://100.96.1.66:8080/video?dummy=param.mjpg
 CUDA-version: 10020 (10020), cuDNN: 8.0.0, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1
 OpenCV version: 4.1.1
Demo
 0 : compute_capability = 620, cudnn_half = 0, GPU: NVIDIA Tegra X2
net.optimized_memory = 0
mini_batch = 1, batch = 8, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
 Create cudnn-handle 0
conv     32       3 x 3/ 1    608 x 608 x   3 ->  608 x 608 x  32 0.639 BF
   1 conv     64       3 x 3/ 2    608 x 608 x  32 ->  304 x 304 x  64 3.407 BF
   2 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   3 route  1                                      ->  304 x 304 x  64
   4 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   5 conv     32       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  32 0.379 BF
   6 conv     64       3 x 3/ 1    304 x 304 x  32 ->  304 x 304 x  64 3.407 BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 304 x 304 x  64 0.006 BF
   8 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   9 route  8 2                                    ->  304 x 304 x 128
  10 conv     64       1 x 1/ 1    304 x 304 x 128 ->  304 x 304 x  64 1.514 BF
  11 conv    128       3 x 3/ 2    304 x 304 x  64 ->  152 x 152 x 128 3.407 BF
  12 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379 BF
  13 route  11                                     ->  152 x 152 x 128
  14 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379 BF
  15 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  16 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703 BF
  17 Shortcut Layer: 14,  wt = 0, wn = 0, outputs: 152 x 152 x  64 0.001 BF
  18 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  19 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703 BF
  20 Shortcut Layer: 17,  wt = 0, wn = 0, outputs: 152 x 152 x  64 0.001 BF
  21 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  22 route  21 12                                  ->  152 x 152 x 128
  23 conv    128       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x 128 0.757 BF
  24 conv    256       3 x 3/ 2    152 x 152 x 128 ->   76 x  76 x 256 3.407 BF
  25 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
  26 route  24                                     ->   76 x  76 x 256
  27 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
```

Figure 11.25: Executing YOLO version 4 for object detection through terminal on live feed from onboard camera over private VPN



```
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 6.910
avg_outputs = 310203
 Allocate additional workspace_size = 26.22 MB
Loading weights from yolov4-tiny.weights...
 seen 64, trained: 0 K-images (0 Kilo-batches_64)
Done! Loaded 38 layers from weights-file
video file: http://100.96.1.66:8080/video?dummy=param.mjpg
Video stream: 640 x 480
Gtk-Message: [illegible]: Failed to load module "canberra-gtk-module"
Objects:


FPS:0.0         AVG_FPS:0.0
Objects:

sofa: 28%
chair: 42%

FPS:2.2         AVG_FPS:0.0
Objects:

sofa: 32%
chair: 31%
chair: 28%

FPS:4.7         AVG_FPS:0.0
Objects:

sofa: 25%
chair: 32%
chair: 28%

FPS:6.7         AVG_FPS:0.0
Objects:

chair: 28%
chair: 27%

FPS:7.8         AVG_FPS:0.0
```

Figure 11.26: Executing YOLO version 4 for object detection through terminal on live feed from onboard camera over private VPN



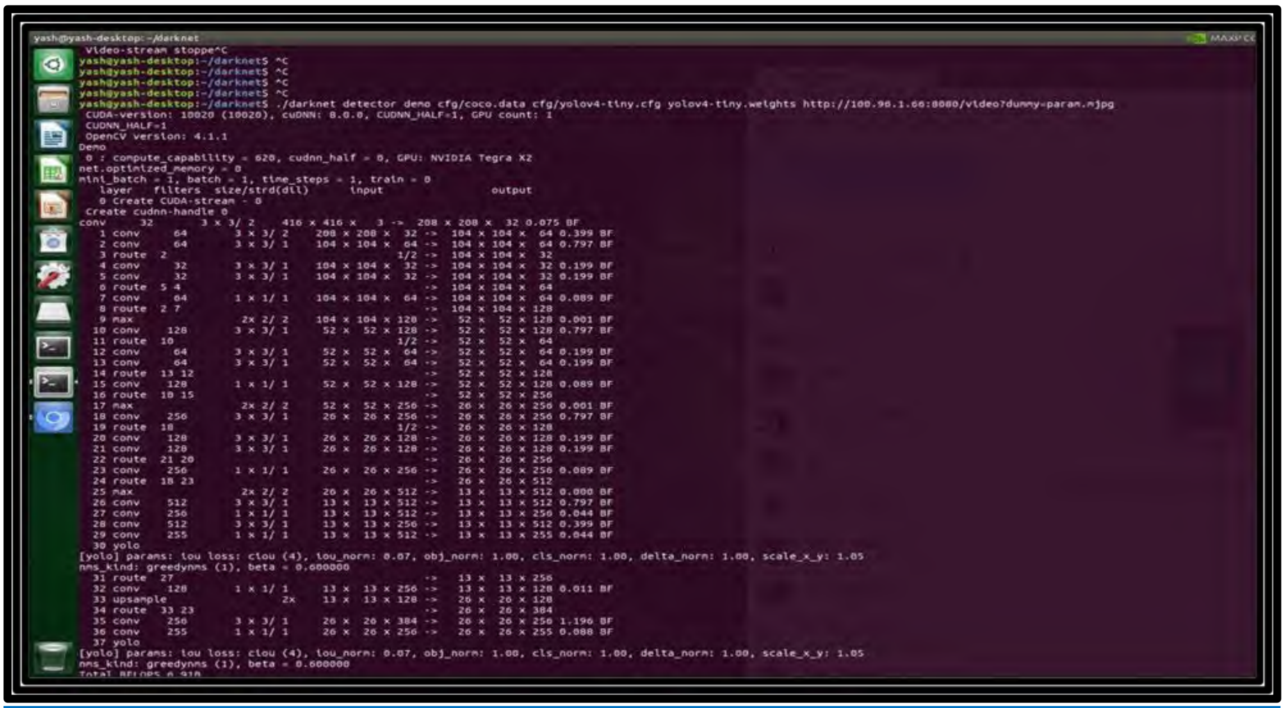
```
yash@yash-desktop: ~/darknet
chair: 32%
chair: 28%

FPS:6.7                  AVG_FPS:0.0
Objects:

chair: 28%
chair: 27%

FPS:7.8                  AVG_FPS:0.0
Objects:

chair: 27%

FPS:8.0                  AVG_FPS:0.0
Objects:

chair: 37%
chair: 28%
chair: 27%

FPS:7.8                  AVG_FPS:0.0
Objects:

chair: 47%
chair: 36%

FPS:7.4                  AVG_FPS:0.0
Objects:

chair: 40%
chair: 33%

FPS:7.2                  AVG_FPS:0.0
Objects:

chair: 39%
chair: 36%

FPS:6.9                  AVG_FPS:0.0
Objects:

chair: 31%

FPS:6.8                  AVG_FPS:0.0
Objects:

chair: 31%

FPS:6.5                  AVG_FPS:0.0
Objects:

chair: 34%
chair: 31%

FPS:6.3                  AVG_FPS:0.0
Objects:
```

Figure 11.27: Objects detected with Frame per second (FPS) statistics

If we want to use YOLOv4-tiny

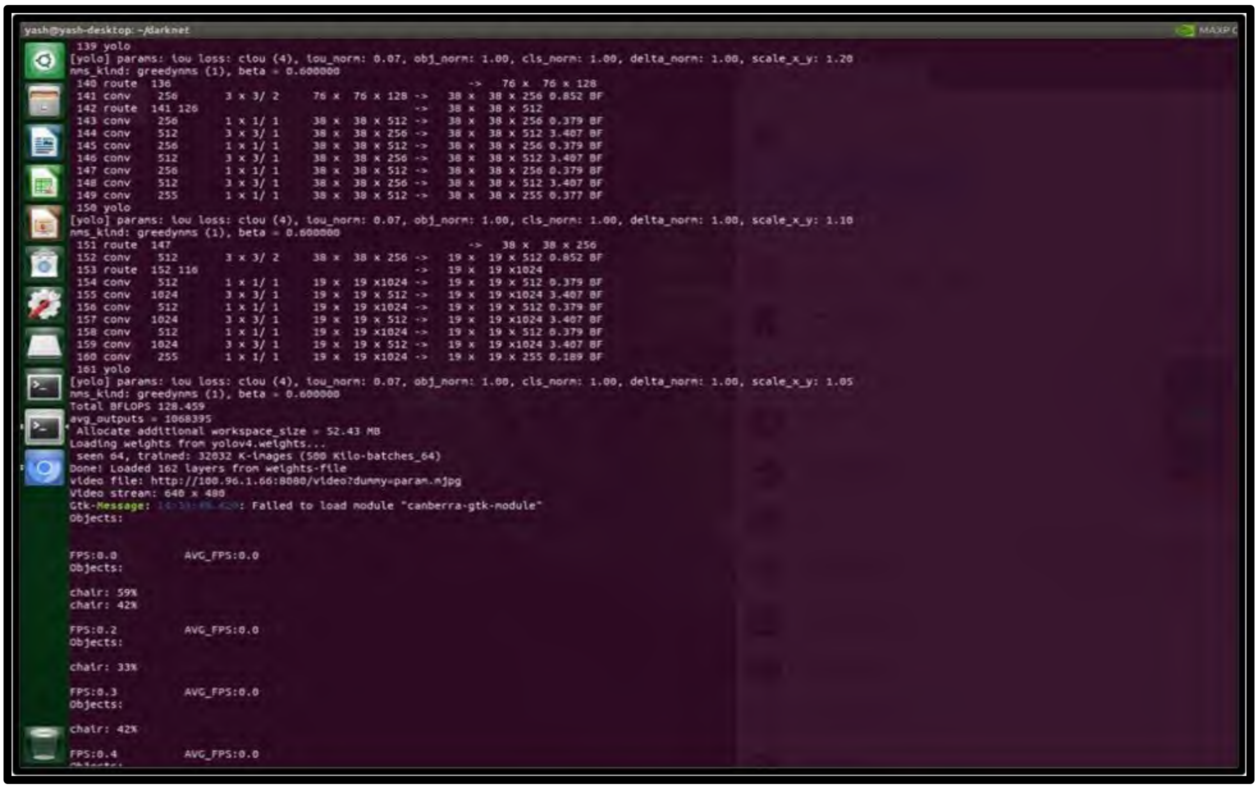
cd darknet

./darknet detector demo cfg/coco.data cfg/yolov4-tiny.cfg yolov4-tiny.weights http://100.96.1.66:8080/video?dummy=param.mjpg



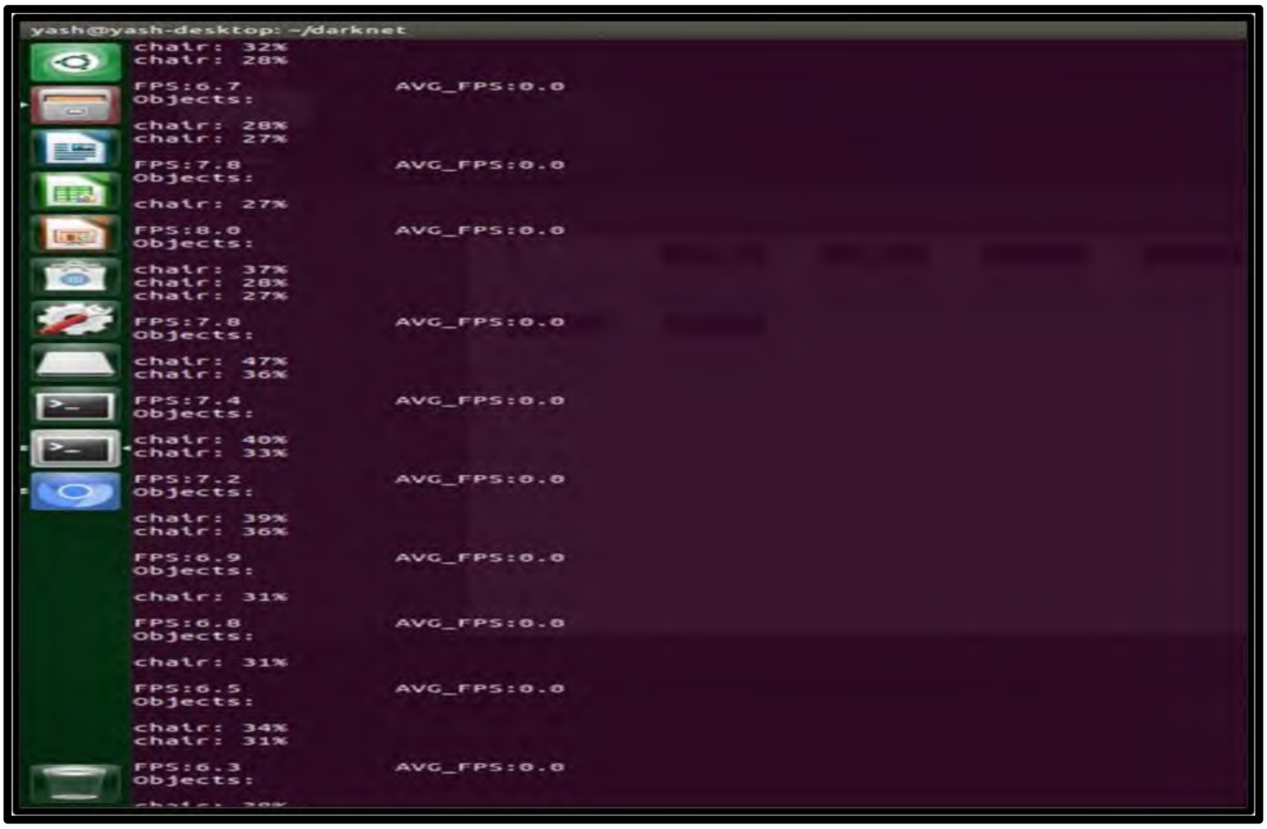
```
yash@yash-desktop: ~/darknet
 Video-stream stoppe^C
yash@yash-desktop:~/darknet$ ^C
yash@yash-desktop:~/darknet$ ^C
yash@yash-desktop:~/darknet$ ^C
yash@yash-desktop:~/darknet$ ^C
yash@yash-desktop:~/darknet$ ./darknet detector demo cfg/coco.data cfg/yolov4-tiny.cfg yolov4-tiny.weights http://100.96.1.66:8080/video?dummy=param.mjpg
 CUDA-version: 10020 (10020), cuDNN: 8.0.0, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1
 OpenCV version: 4.1.1
Demo
 0 : compute_capability = 620, cudnn_half = 0, GPU: NVIDIA Tegra X2
net.optimized_memory = 0
mini_batch = 1, batch = 1, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
 Create cudnn-handle 0
conv     32       3 x 3/ 2    416 x 416 x   3 ->  208 x 208 x  32 0.075 BF
   1 conv     64       3 x 3/ 2    208 x 208 x  32 ->  104 x 104 x  64 0.399 BF
   2 conv     64       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.797 BF
   3 route  2                                  1/2 ->  104 x 104 x  32
   4 conv     32       3 x 3/ 1    104 x 104 x  32 ->  104 x 104 x  32 0.199 BF
   5 conv     32       3 x 3/ 1    104 x 104 x  32 ->  104 x 104 x  32 0.199 BF
   6 route  5 4                                    ->  104 x 104 x  64
   7 conv     64       1 x 1/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
   8 route  2 7                                    ->  104 x 104 x 128
   9 max                2x 2/ 2    104 x 104 x 128 ->   52 x  52 x 128 0.001 BF
  10 conv    128       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
  11 route  10                                 1/2 ->   52 x  52 x  64
  12 conv     64       3 x 3/ 1     52 x  52 x  64 ->   52 x  52 x  64 0.199 BF
  13 conv     64       3 x 3/ 1     52 x  52 x  64 ->   52 x  52 x  64 0.199 BF
  14 route  13 12                                  ->   52 x  52 x 128
  15 conv    128       1 x 1/ 1     52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
  16 route  10 15                                  ->   52 x  52 x 256
  17 max                2x 2/ 2     52 x  52 x 256 ->   26 x  26 x 256 0.001 BF
  18 conv    256       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 256 0.797 BF
  19 route  18                                 1/2 ->   26 x  26 x 128
  20 conv    128       3 x 3/ 1     26 x  26 x 128 ->   26 x  26 x 128 0.199 BF
  21 conv    128       3 x 3/ 1     26 x  26 x 128 ->   26 x  26 x 128 0.199 BF
  22 route  21 20                                  ->   26 x  26 x 256
  23 conv    256       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 256 0.089 BF
  24 route  18 23                                  ->   26 x  26 x 512
  25 max                2x 2/ 2     26 x  26 x 512 ->   13 x  13 x 512 0.000 BF
  26 conv    512       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x 512 0.797 BF
  27 conv    256       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 256 0.044 BF
  28 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  29 conv    255       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 255 0.044 BF
  30 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
  31 route  27                                     ->   13 x  13 x 256
  32 conv    128       1 x 1/ 1     13 x  13 x 256 ->   13 x  13 x 128 0.011 BF
  33 upsample                 2x    13 x  13 x 128 ->   26 x  26 x 128
  34 route  33 23                                  ->   26 x  26 x 384
  35 conv    256       3 x 3/ 1     26 x  26 x 384 ->   26 x  26 x 256 1.196 BF
  36 conv    255       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 255 0.088 BF
  37 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
```

Figure 11.28: Executing YOLO-tiny version 4 for object detection through terminal on live feed from onboard camera over private VPN



```
yash@yash-desktop: ~/darknet
 139 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.20
nms_kind: greedynms (1), beta = 0.600000
 140 route  136                                    ->   76 x  76 x 128
 141 conv    256       3 x 3/ 2     76 x  76 x 128 ->   38 x  38 x 256 0.852 BF
 142 route  141 126                                ->   38 x  38 x 512
 143 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379 BF
 144 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407 BF
 145 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379 BF
 146 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407 BF
 147 conv    256       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 256 0.379 BF
 148 conv    512       3 x 3/ 1     38 x  38 x 256 ->   38 x  38 x 512 3.407 BF
 149 conv    255       1 x 1/ 1     38 x  38 x 512 ->   38 x  38 x 255 0.377 BF
 150 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.10
nms_kind: greedynms (1), beta = 0.600000
 151 route  147                                    ->   38 x  38 x 256
 152 conv    512       3 x 3/ 2     38 x  38 x 256 ->   19 x  19 x 512 0.852 BF
 153 route  152 116                                ->   19 x  19 x1024
 154 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379 BF
 155 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407 BF
 156 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379 BF
 157 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407 BF
 158 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379 BF
 159 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407 BF
 160 conv    255       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 255 0.189 BF
 161 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 128.459
avg_outputs = 1068395
 Allocate additional workspace_size = 52.43 MB
Loading weights from yolov4.weights...
 seen 64, trained: 32032 K-images (500 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
video file: http://100.96.1.66:8080/video?dummy=param.mjpg
Video stream: 640 x 480
Gtk-Message: [illegible]: Failed to load module "canberra-gtk-module"
Objects:


FPS:0.0         AVG_FPS:0.0
Objects:

chair: 59%
chair: 42%

FPS:0.2         AVG_FPS:0.0
Objects:

chair: 33%

FPS:0.3         AVG_FPS:0.0
Objects:

chair: 42%

FPS:0.4         AVG_FPS:0.0
```

Figure 11.29: Executing YOLO-tiny version 4 for object detection through terminal on live feed from onboard camera over private VPN



```
yash@yash-desktop: ~/darknet
chair: 32%
chair: 28%

FPS:6.7         AVG_FPS:0.0
Objects:

chair: 28%
chair: 27%

FPS:7.8         AVG_FPS:0.0
Objects:

chair: 27%

FPS:8.0         AVG_FPS:0.0
Objects:

chair: 37%
chair: 28%
chair: 27%

FPS:7.8         AVG_FPS:0.0
Objects:

chair: 47%
chair: 36%

FPS:7.4         AVG_FPS:0.0
Objects:

chair: 40%
chair: 33%

FPS:7.2         AVG_FPS:0.0
Objects:

chair: 39%
chair: 36%

FPS:6.9         AVG_FPS:0.0
Objects:

chair: 31%

FPS:6.8         AVG_FPS:0.0
Objects:

chair: 31%

FPS:6.5         AVG_FPS:0.0
Objects:

chair: 34%
chair: 31%

FPS:6.3         AVG_FPS:0.0
Objects:
```

Figure 11.30: Objects detected with Frame per second (FPS) statistics

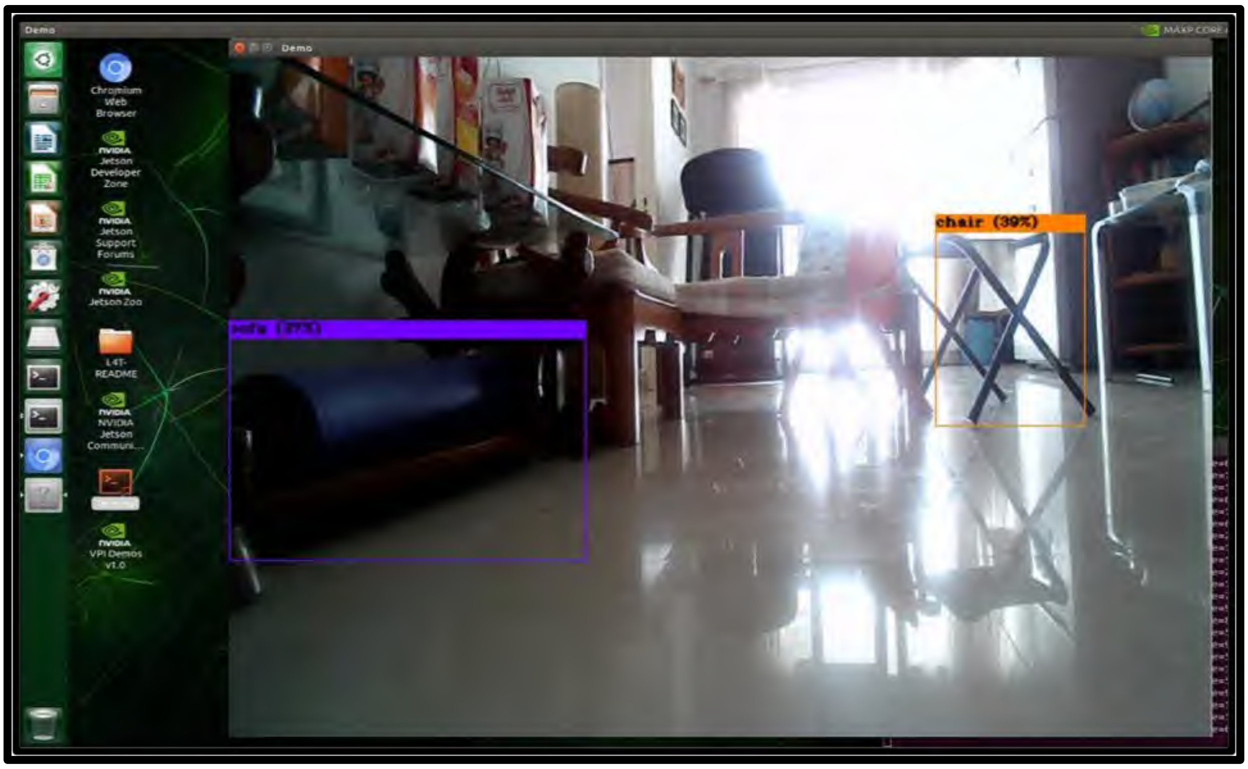


Figure 11.31: Objects detection on live video feed

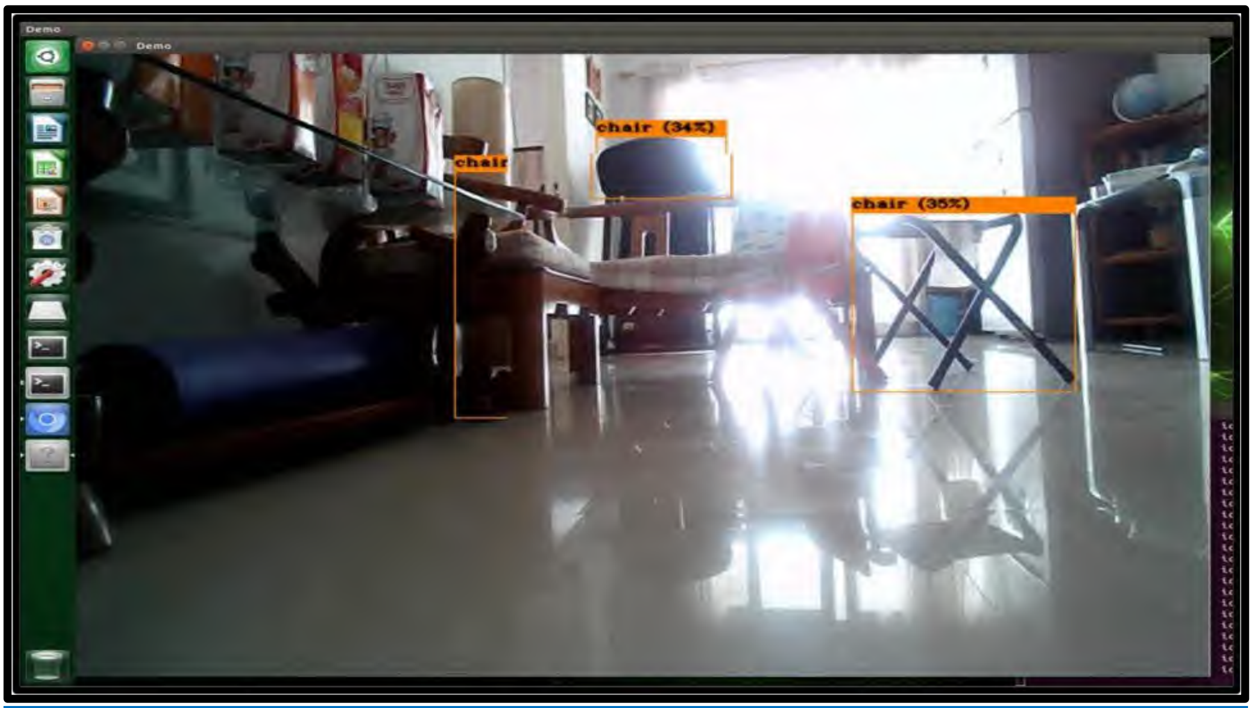


Figure 11.32: Objects detection on live video feed

YoloV4 is more accurate than YOLOv4-tiny, however YOLOv4-tiny is much faster than YOLOv4. It entirely depends upon the requirement and resources available. A

quick understanding of the concept is as per pictures below

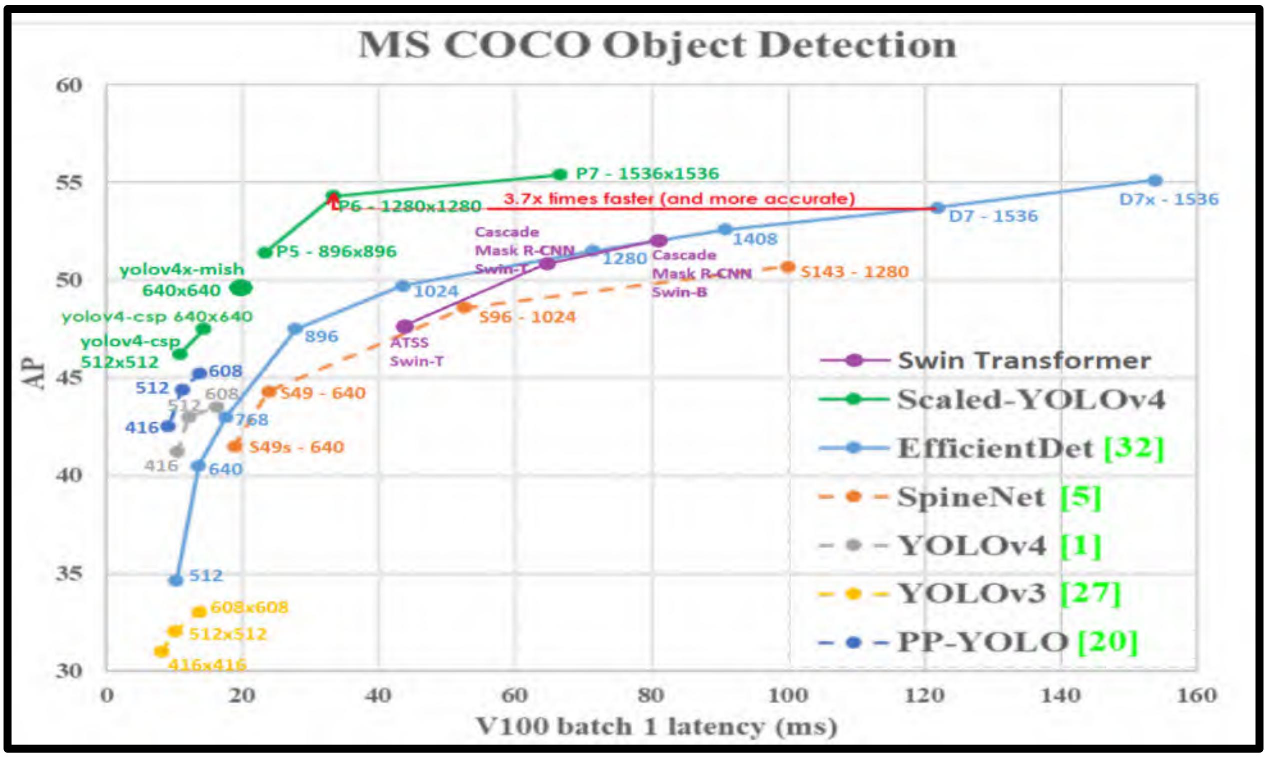
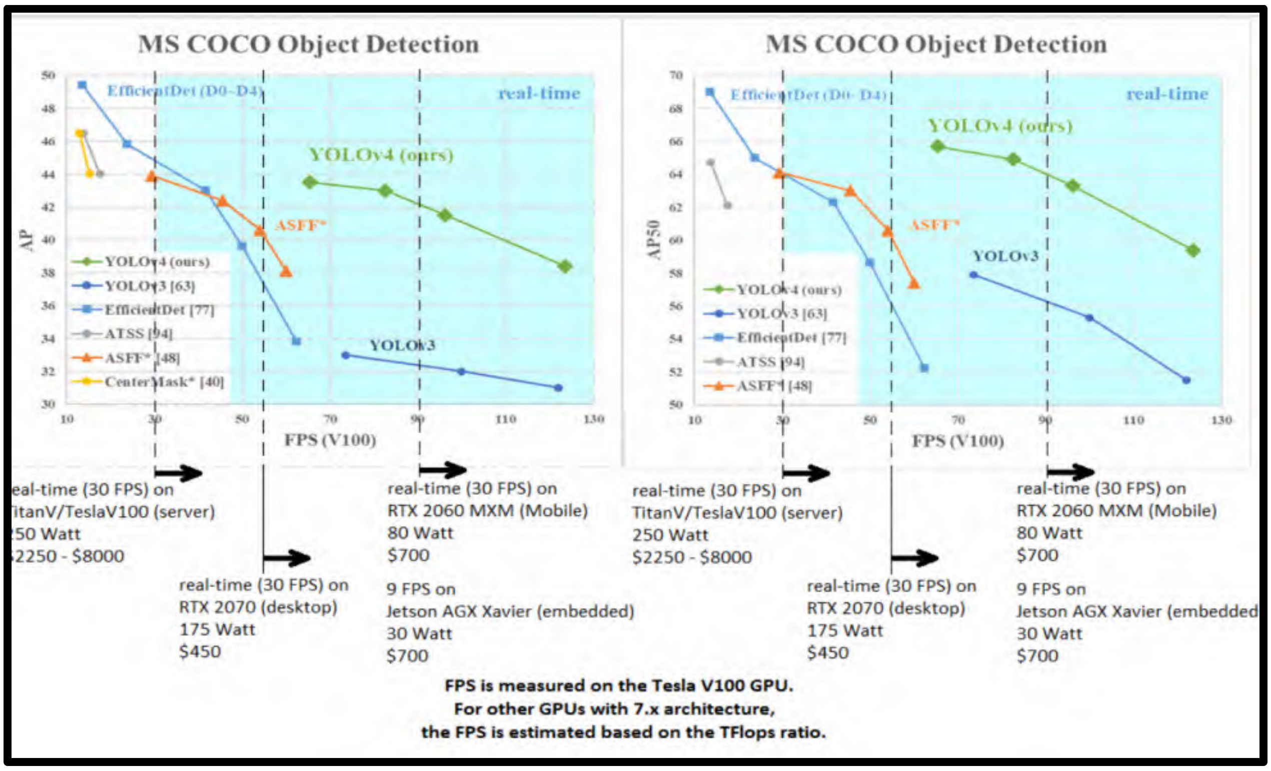


Figure 11.33: Speed vs accuracy comparison of YOLO versions



eal-time (30 FPS) on
itanV/TeslaV100 (server)
50 Watt
2250 - $8000

real-time (30 FPS) on
RTX 2070 (desktop)
175 Watt
$450

real-time (30 FPS) on
RTX 2060 MXM (Mobile)
80 Watt
$700

9 FPS on
Jetson AGX Xavier (embedded)
30 Watt
$700

real-time (30 FPS) on
TitanV/TeslaV100 (server)
250 Watt
$2250 - $8000

real-time (30 FPS) on
RTX 2070 (desktop)
175 Watt
$450

real-time (30 FPS) on
RTX 2060 MXM (Mobile)
80 Watt
$700

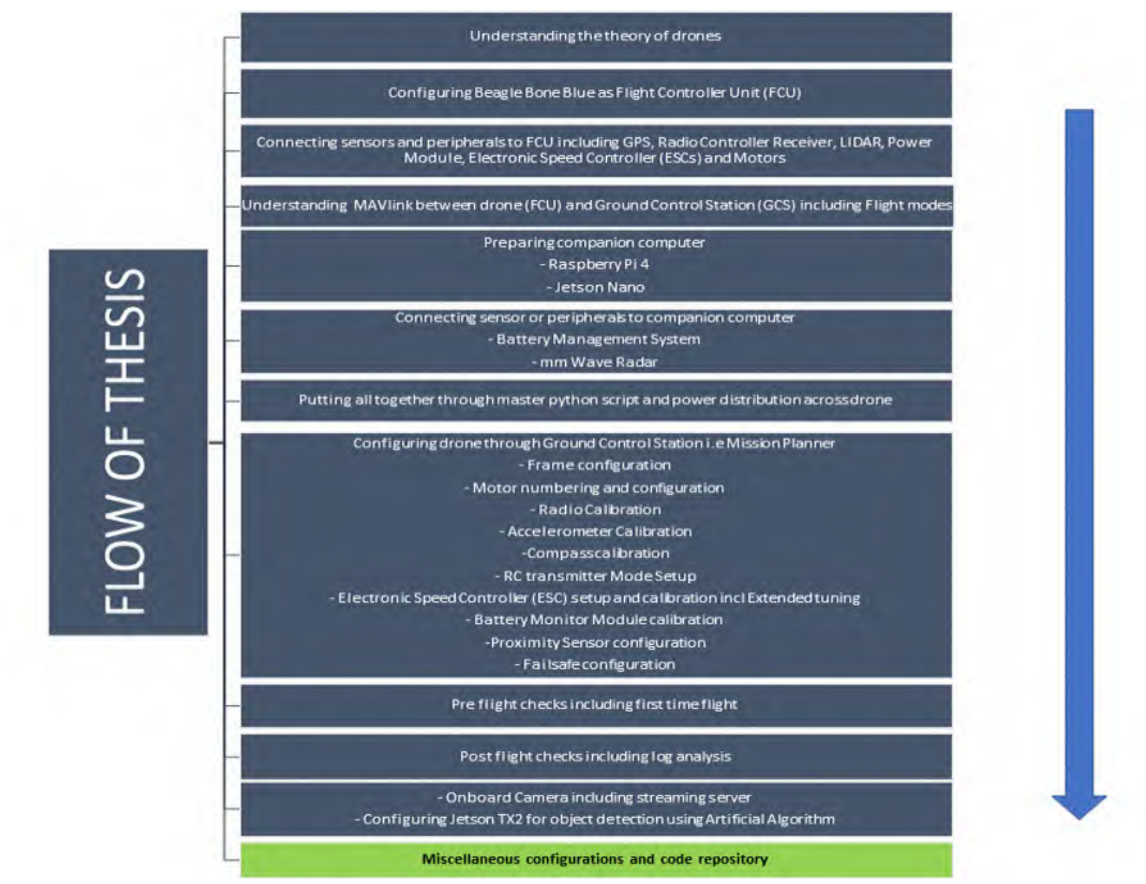
9 FPS on
Jetson AGX Xavier (embedded
30 Watt
$700

FPS is measured on the Tesla V100 GPU.
For other GPUs with 7.x architecture,
the FPS is estimated based on the TFlops ratio.

Figure 11.34: Speed vs accuracy comparison of YOLO versions

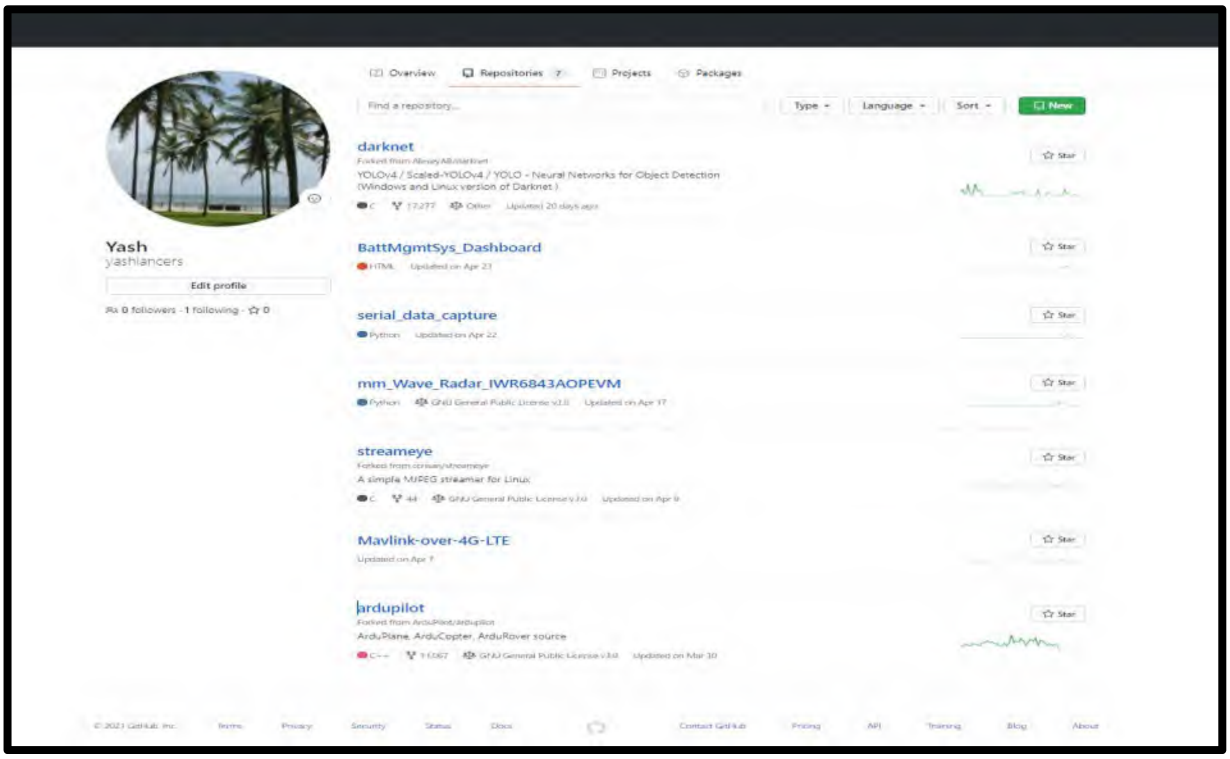
CHAPTER 12

CODES REPOSITORIES ON GITHUB


FLOW OF THESIS
Understanding the theory of drones
Configuring Beagle Bone Blue as Flight Controller Unit (FCU)
Connecting sensors and peripherals to FCU including GPS, Radio Controller Receiver, LIDAR, Power Module, Electronic Speed Controller (ESCs) and Motors
Understanding MAVlink between drone (FCU) and Ground Control Station (GCS) including Flight modes
Preparing companion computer
- Raspberry Pi 4
- Jetson Nano
Connecting sensor or peripherals to companion computer
- Battery Management System
- mm Wave Radar
Putting all together through master python script and power distribution across drone
Configuring drone through Ground Control Station i.e Mission Planner
- Frame configuration
- Motor numbering and configuration
- Radio Calibration
- Accelerometer Calibration
-Compass calibration
- RC transmitter Mode Setup
- Electronic Speed Controller (ESC) setup and calibration incl Extended tuning
- Battery Monitor Module calibration
-Proximity Sensor configuration
- Failsafe configuration
Pre flight checks including first time flight
Post flight checks including log analysis
- Onboard Camera including streaming server
- Configuring Jetson TX2 for object detection using Artificial Algorithm
Miscellaneous configurations and code repository
Figure 12.1 Flow of thesis until now

All the python scripts or codes/application dumps utilized during the implementation of the project is part of my github page https://github.com/yashlancers
Two repositories developed by me specifically for this project implementation are **BattMgmtSys_Dashboard** and **mm_Wave_Radar_IWR6843AOPEVM.** They have been developed after studying various different repositories available on github and their urls/links are part of the Bibliography.

Others repositories which I utilized directly from github have been forked to my account for easy reference. These are namely **Ardupilot** and **streameye.** Ardupilot was utilized to compile the autopilot software on Flight Controller Unit (FCU) i.e Beagle Bone Blue as seen in Chapter 2. Streameye repository was used for compiling MJPEG video streaming server on companion computer as seen in Chapter 11



Yash
yashlancers

0 followers · 1 following · 0

darknet
Forked from AlexeyAB/darknet
YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet)
C 17,277 Other Updated 20 days ago

BattMgmtSys_Dashboard
HTML Updated on Apr 23

serial_data_capture
Python Updated on Apr 22

mm_Wave_Radar_IWR6843AOPEVM
Python GNU General Public License v3.0 Updated on Apr 17

streameye
Forked from ccrisan/streameye
A simple MJPEG streamer for Linux
C 44 GNU General Public License v3.0 Updated on Apr 9

Mavlink-over-4G-LTE
Updated on Apr 7

ardupilot
Forked from ArduPilot/ardupilot
ArduPlane, ArduCopter, ArduRover source
C++ 11,067 GNU General Public License v3.0 Updated on Mar 10

© 2021 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Figure 12.2: My Github repository **https://github.com/yashlancers**

CHAPTER 13

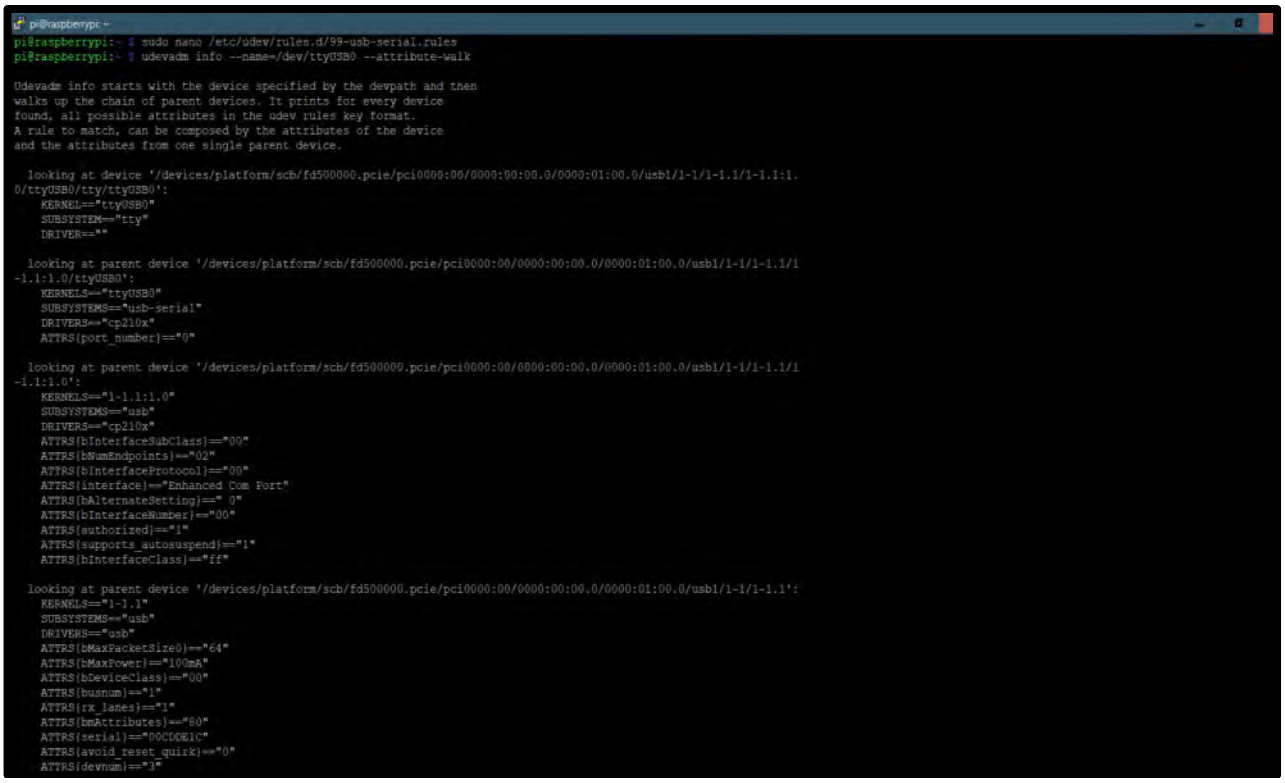
MISCELLANEOUS CONFIGURATIONS

13.1 To identify device as serial on USB

All devices connected to USB are allotted different identity /dev/ttyUSB* which can change depending upon various factors resulting in error while executing python scripts for mmWave radar orrs485 to USB convertor. In order to uniquely identify USB connected devices we need to define rules.

Step1 Connect all USB devices to companion computer and use the below command to identify unique identifiers.

udevadm info --name=/dev/ttyUSB0 --attribute-walk



```
pi@raspberrypi:~ $ sudo nano /etc/udev/rules.d/99-usb-serial.rules
pi@raspberrypi:~ $ udevadm info --name=/dev/ttyUSB0 --attribute-walk

Udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

  looking at device '/devices/platform/scb/fd500000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0/usb1/1-1/1-1.1/1-1.1:1.
0/ttyUSB0/tty/ttyUSB0':
    KERNEL=="ttyUSB0"
    SUBSYSTEM=="tty"
    DRIVER==""

  looking at parent device '/devices/platform/scb/fd500000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0/usb1/1-1/1-1.1/1
-1.1:1.0/ttyUSB0':
    KERNELS=="ttyUSB0"
    SUBSYSTEMS=="usb-serial"
    DRIVERS=="cp210x"
    ATTRS{port_number}=="0"

  looking at parent device '/devices/platform/scb/fd500000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0/usb1/1-1/1-1.1/1
-1.1:1.0':
    KERNELS=="1-1.1:1.0"
    SUBSYSTEMS=="usb"
    DRIVERS=="cp210x"
    ATTRS{bInterfaceSubClass}=="00"
    ATTRS{bNumEndpoints}=="02"
    ATTRS{bInterfaceProtocol}=="00"
    ATTRS{interface}=="Enhanced Com Port"
    ATTRS{bAlternateSetting}==" 0"
    ATTRS{bInterfaceNumber}=="00"
    ATTRS{authorized}=="1"
    ATTRS{supports_autosuspend}=="1"
    ATTRS{bInterfaceClass}=="ff"

  looking at parent device '/devices/platform/scb/fd500000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0/usb1/1-1/1-1.1':
    KERNELS=="1-1.1"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{bMaxPacketSize0}=="64"
    ATTRS{bMaxPower}=="100mA"
    ATTRS{bDeviceClass}=="00"
    ATTRS{busnum}=="1"
    ATTRS{rx_lanes}=="1"
    ATTRS{bmAttributes}=="80"
    ATTRS{serial}=="00CDDE1C"
    ATTRS{avoid_reset_quirk}=="0"
    ATTRS{devnum}=="3"
```

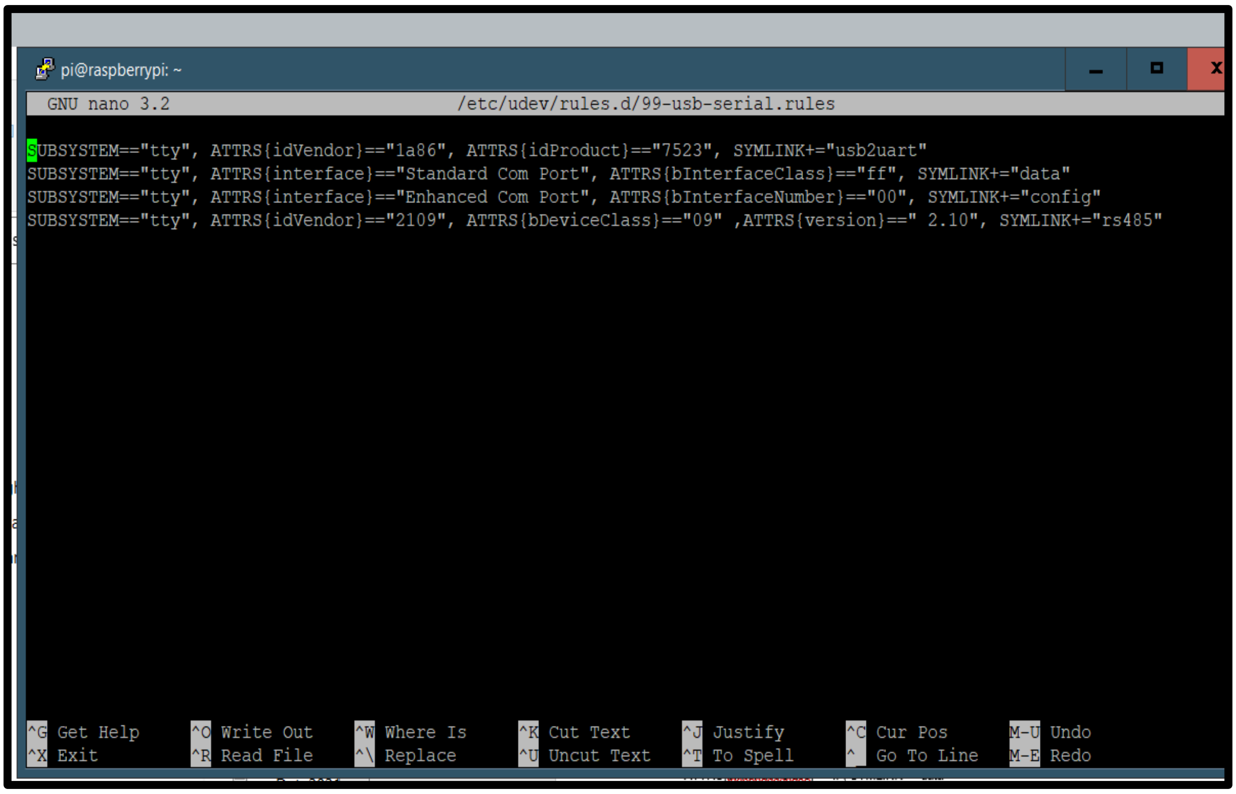
Figure 13.1: Console output of walkthrough for USB device connected

Similarly do for udevadm info --name=/dev/ttyUSB1 --attribute-walk, udevadm info --name=/dev/ttyUSB2 --attribute-walk, udevadm info --name=/dev/ttyUSB3 --attribute-walk and so on

Step2 Add the rules as given below to uniquely identify USB connected devices-

sudo nano /etc/udev/rules.d/99-usb-serial.rules

SUBSYSTEM=="tty", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", SYMLINK+="usb2uart"
SUBSYSTEM=="tty", ATTRS{interface}=="Standard Com Port", ATTRS{bInterfaceClass}=="ff", SYMLINK+="data"
SUBSYSTEM=="tty", ATTRS{interface}=="Enhanced Com Port", ATTRS{bInterfaceNumber}=="00", SYMLINK+="config"
SUBSYSTEM=="tty", ATTRS{idVendor}=="2109", ATTRS{bDeviceClass}=="09" ,ATTRS{version}==" 2.10", SYMLINK+="rs485"



```
  GNU nano 3.2                    /etc/udev/rules.d/99-usb-serial.rules

SUBSYSTEM=="tty", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", SYMLINK+="usb2uart"
SUBSYSTEM=="tty", ATTRS{interface}=="Standard Com Port", ATTRS{bInterfaceClass}=="ff", SYMLINK+="data"
SUBSYSTEM=="tty", ATTRS{interface}=="Enhanced Com Port", ATTRS{bInterfaceNumber}=="00", SYMLINK+="config"
SUBSYSTEM=="tty", ATTRS{idVendor}=="2109", ATTRS{bDeviceClass}=="09" ,ATTRS{version}==" 2.10", SYMLINK+="rs485"
```

Figure 13.2: Creating /etc/udev/rules.d/99-usb-serial.rules

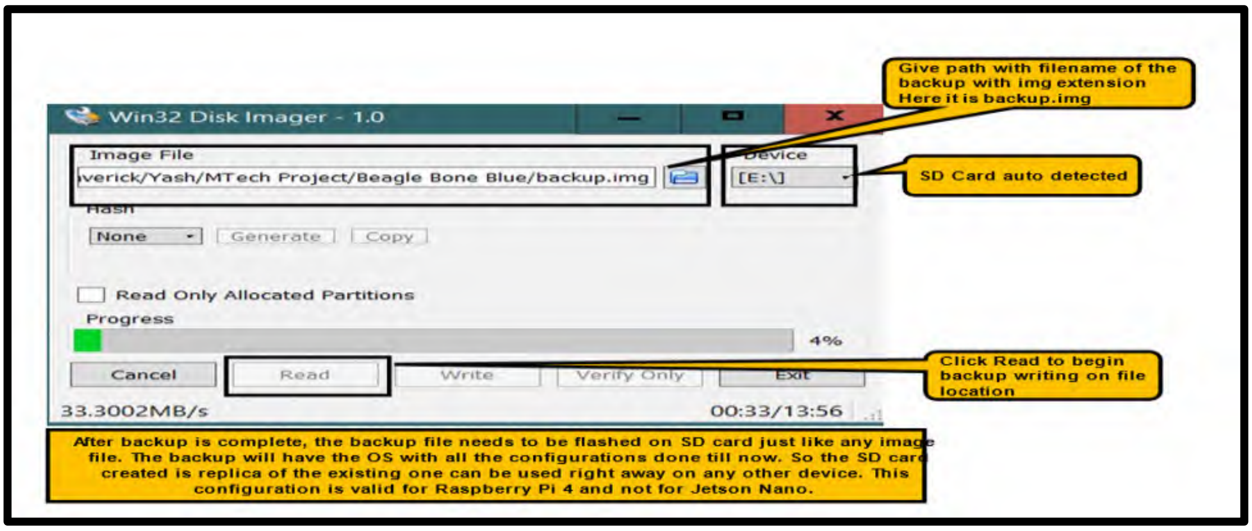
Step3 Update the rules

```
sudo udevadm control --reload-rules && udevadm trigger
```

Now for example in my case instead of /dev/ttyUSB0 for a convertor (RS485 to USB), I can use /dev/usb2uart in python script and it won't change.

13.2 Backup of Raspberry Pi 4 or Beagle Bone Blue

We require to take backup of SD card to save our OS with configuration from any SD card failure and the ability to quickly restore the configuration incase of SD card failure. I have used Win32DiskImager. This application can take backup of SD card of Beagle Bone Blue or Raspberry Pi 4but not for Jetson Nano. The process is explained in the photo below. Need to provide the directory path for backup including the destination filename i.e backup file name. The extension of backup must be .img (image file). We click Read to begin the backup process as SD card is auto detected unless working on more than two SD cards simultaneously. The backup generated is the replica of the original SD card containing the OS with all configurations done till now and can be readily used.


Give path with filename of the backup with img extension Here it is backup.img

Win32 Disk Imager - 1.0

Image File: verick/Yash/MTech Project/Beagle Bone Blue/backup.img

Device: [E:\]

SD Card auto detected

Hash: None

Read Only Allocated Partitions

Progress 4%

Click Read to begin backup writing on file location

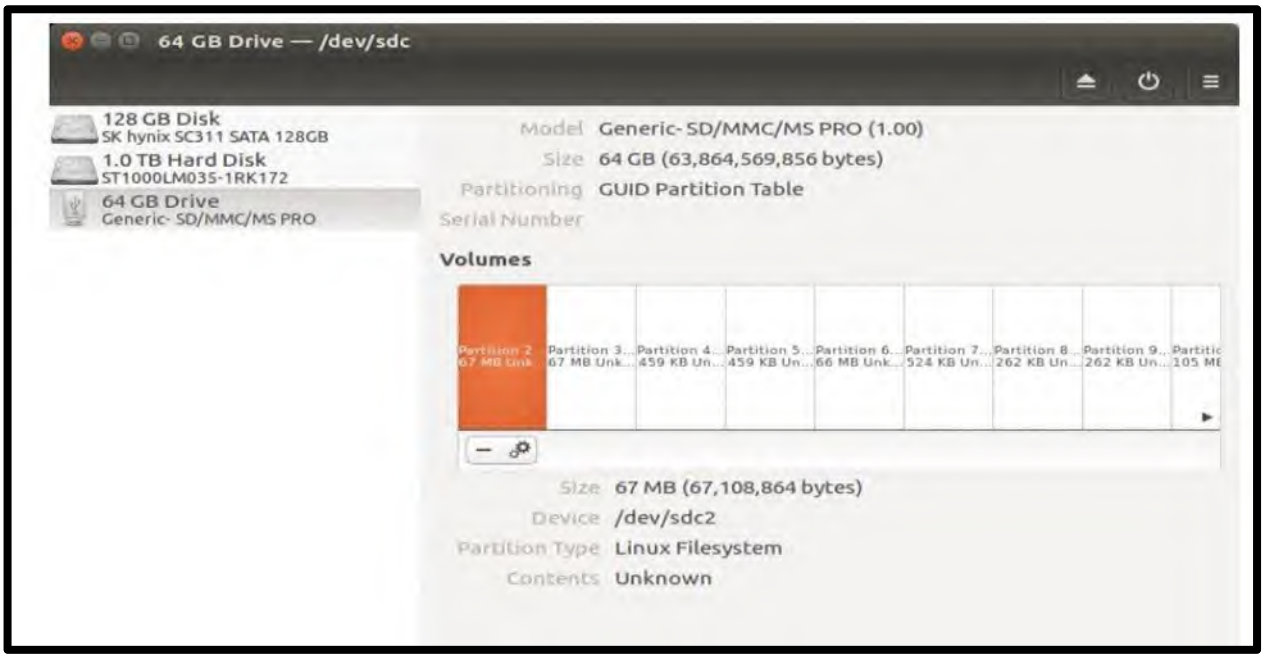
33.3002MB/s 00:33/13:56

After backup is complete, the backup file needs to be flashed on SD card just like any image file. The backup will have the OS with all the configurations done till now. So the SD card created is replica of the existing one can be used right away on any other device. This configuration is valid for Raspberry Pi 4 and not for Jetson Nano.

Figure 13.3: Using Win32 Disk Imager to take SD card backup for Beagle Bone Blue or Raspberry Pi 4

Backup of Jetson Nano SD Card The backup of Jetson Nano SD card is quite different from the process above. We need a Linux machine to take backup. We will use the dd command which is the oldest disk imaging tool on Linux.

Step 1 Use Disks GUI application to check for the SD card details still in use.


64 GB Drive — /dev/sdc
128 GB Disk
SK hynix SC311 SATA 128GB
1.0 TB Hard Disk
ST1000LM035-1RK172
64 GB Drive
Generic- SD/MMC/MS PRO
Volumes
Partition 2 67 MB Unk
Partition 3... 67 MB Unk...
Partition 4... 459 KB Un...
Partition 5... 459 KB Un...
Partition 6... 66 MB Unk...
Partition 7... 524 KB Un...
Partition 8... 262 KB Un...
Partition 9... 262 KB Un...
Partitic 105 ME

Model Generic- SD/MMC/MS PRO (1.00)
Size 64 GB (63,864,569,856 bytes)
Partitioning GUID Partition Table
Serial Number

Size 67 MB (67,108,864 bytes)
Device /dev/sdc2
Partition Type Linux Filesystem
Contents Unknown

Figure 13.4: SD card details on Linux PC/Laptop

Step 2 Make sure the SD card is unmounted.

```
sudo umount /dev/sdX
```

where X is SD card name

Step 3 Now we create backup in the home directory

```
sudo dd if=/dev/sdX conv=sync,noerror bs=64K | gzip -c > ~/backup_image.img.gz
```

Step 4. Now we can move this backup to windows and use any SD card flashing tool to flash SD card as discussed in earlier chapters.

Step 5 If we wish to restore back on SD card in Linux itself

```
sudo su
```

$ gunzip -c ~/backup_image.img.gz | dd of=/dev/sdX bs=64K

13.3 Remote Desktop in Raspberry Pi 4

Xrdp is an open-source implementation of the Microsoft Remote Desktop Protocol (RDP) .It allows GUI based system access. We login to Raspberry Pi and run the below mentioned commands to install Pixel desktop:

Update system

sudo apt update

sudo apt-get install raspberrypi-ui-mods xinit xserver-xorg

sudo reboot

As Xrdp package is available in the default Raspbian Buster repositories, we can install it right away

sudo apt install xrdp

After installation process is completed, the Xrdp service will automatically start. We verify it by

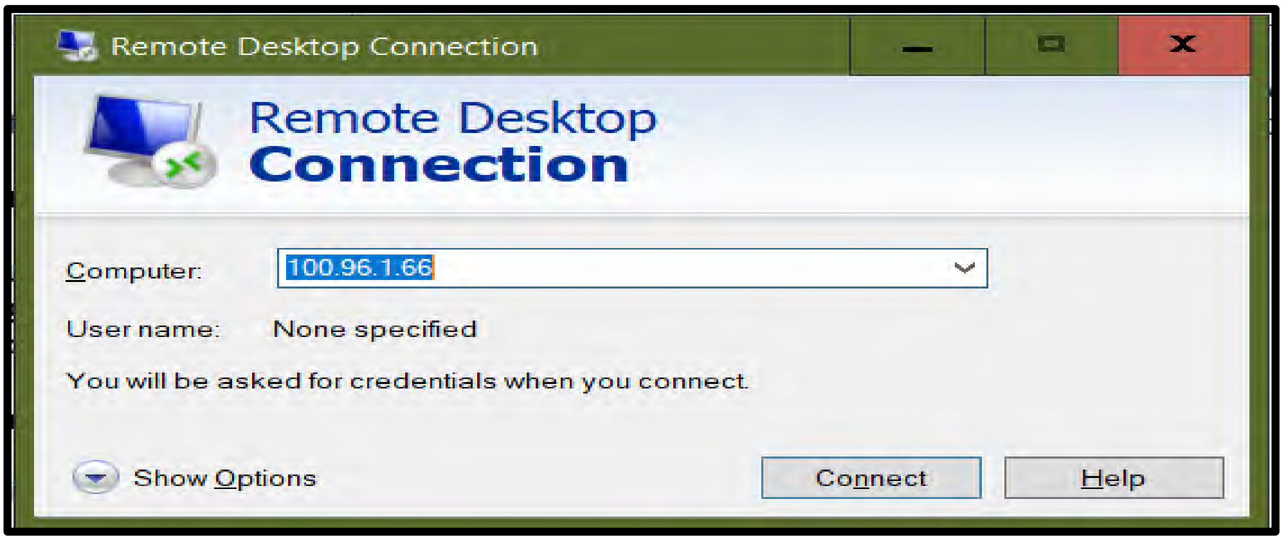
systemctl show -p SubState --value xrdp

Now we need to add the user to the group

sudo adduser xrdp ssl-cert

To access Raspberry Pi 4 through remote session

Open RDP client on Windows, add IP address and establish connection.



Remote Desktop Connection

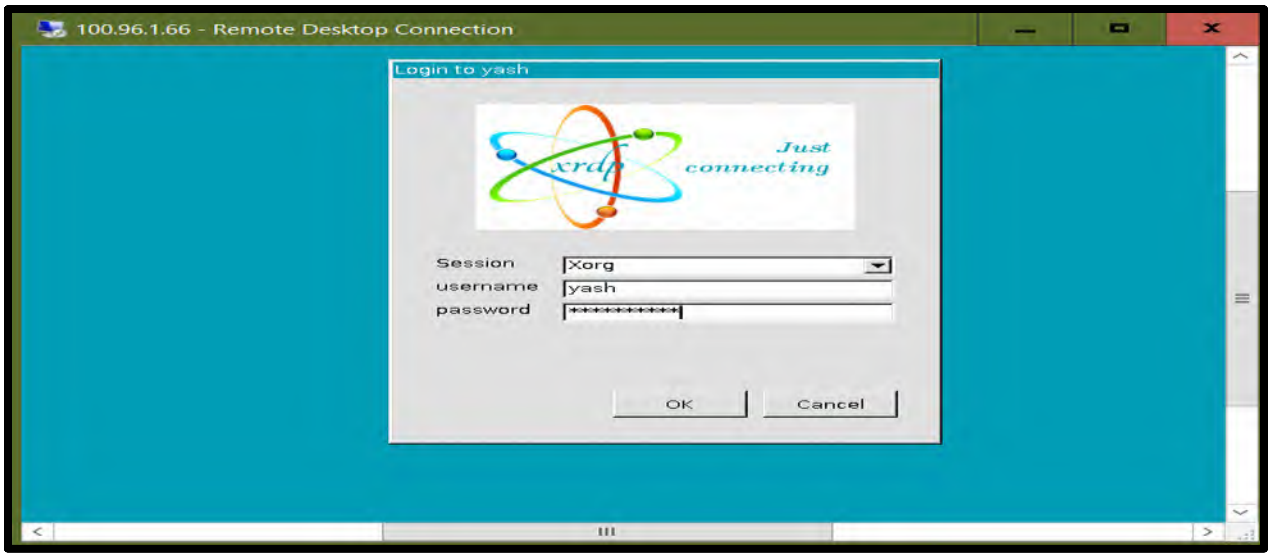
Computer: 100.96.1.66

User name: None specified

You will be asked for credentials when you connect.

Figure 13.5: For establishing Remote Desktop Session of Raspberry Pi 4 over VPN for remote configuration is quite helpful

Put user and password for remote connection



100.96.1.66 - Remote Desktop Connection

Login to yash

Session: Xorg

username: yash

password: ************

Figure 13.6: User credentials for establishing Remote Desktop Session of Raspberry Pi 4 over VPN for remote configuration is quite helpful

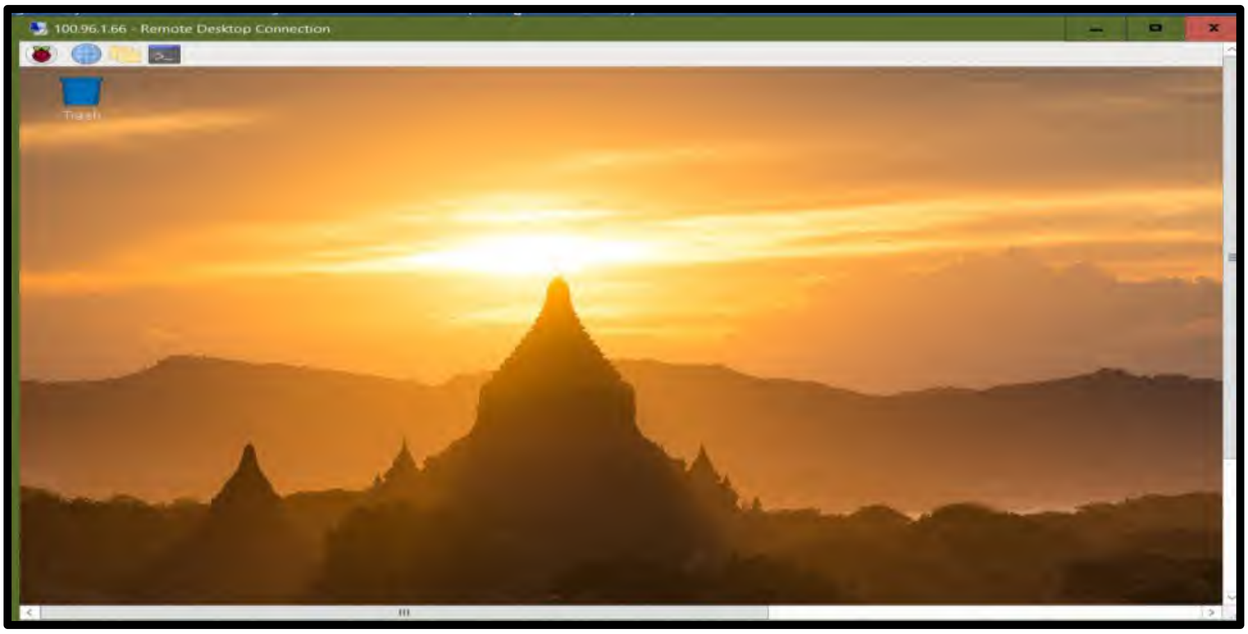


Figure 13.7: Remote Desktop Session of Raspberry Pi 4 over VPN established successfully

13.4 Bibliography

This project had many implementational aspects for which I referred to a number of blogs, websites and YouTube videos. I have tried to place some important links in Bibliography.

“4G LTE Cellular BVLOS Drone Control - XRD.” Botlink, https://botlink.com/4g-lte-cellular-bvlos-drone-control-xrd.

Adding a New MAVLink Message - Dev Documentation. https://ardupilot.org/dev/docs/code-overview-adding-a-new-mavlink-message.html.

AdrieSentosaFollow. “Raspberry Pi - Data Logging.” Instructables, https://www.instructables.com/Raspberry-Pi-Data-Logging/.

Advanced Tuning — Copter Documentation. https://ardupilot.org/copter/docs/tuning.html.

Alexey. AlexeyAB/Darknet. 2016. 2021. GitHub, https://github.com/AlexeyAB/darknet.

“Alro10/YOLO-Darknet-on-Jetson-TX2.” GitHub, https://github.com/Alro10/YOLO-darknet-on-Jetson-TX2.

“ArduPilot Discourse.” ArduPilot Discourse, https://discuss.ardupilot.org/.

ArduPilot Documentation — ArduPilot Documentation. https://ardupilot.org/ardupilot/index.html.

ArduPilot Firmware : /Tools/MissionPlanner/Gstreamer. https://firmware.ardupilot.org/Tools/MissionPlanner/gstreamer/.

ArduPilot/Ardupilot. 2013. ArduPilot, 2021. GitHub, https://github.com/ArduPilot/ardupilot.

Beagleboard:BeagleBoneBlack Debian - ELinux.Org. https://elinux.org/Beagleboard:BeagleBoneBlack_Debian#Flashing_eMMC.

BeagleBoard.Org - Getting-Started. https://beagleboard.org/getting-started#update.

Camera Configuration - Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/configuration/camera.md.

Canu, Sergio. “YOLO V3 - Install and Run Yolo on Nvidia Jetson Nano (with GPU).” Pysource, 29 Aug. 2019, https://pysource.com/2019/08/29/yolo-v3-install-and-run-yolo-on-nvidia-jetson-nano-with-gpu/.

“Clone SD Card - Jetson Nano and Xavier NX.” JetsonHacks, 8 Aug. 2020, https://www.jetsonhacks.com/2020/08/08/clone-sd-card-jetson-nano-and-xavier-nx/.

Shaun Taylor-Morgan Feed “The 7 Most Popular Ways to Plot Data in Python.” Opensource.Com, https://opensource.com/article/20/4/plot-data-python.

Communicating with Raspberry Pi via MAVLink — Dev Documentation. https://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html. Accessed 12 June 2021.

“Connect BeagleBone Black to Cellular with Huawei E303 Modem.” Hackster.Io, https://www.hackster.io/hologram/connect-beaglebone-black-to-cellular-with-huawei-e303-modem-0e14c2.

Copter Object Avoidance — Dev Documentation. https://ardupilot.org/dev/docs/code-overview-object-avoidance.html.

“Debian - Trying to Run a Python Script as a Service Using Systemd.” Unix & Linux Stack Exchange, https://unix.stackexchange.com/questions/364412/trying-to-run-a-python-script-as-a-service-using-systemd.

“Debian - What’s the Difference between /Usr/Lib/Systemd/System and /Etc/Systemd/System?” Unix & Linux Stack Exchange, https://unix.stackexchange.com/questions/206315/whats-the-difference-between-usr-lib-systemd-system-and-etc-systemd-system.

defgw. “Collecting Data from Serial Port and CSV Handling.” Defgw, 5 Apr. 2015, https://defgw.wordpress.com/2015/04/05/collecting-data-from-serial-port-and-csv-handling/.

Denecke, Mirko. Mirkix/Ardupilotblue. 2017. 2021. GitHub, https://github.com/mirkix/ardupilotblue.

Derek. “About the Book.” Exploring BeagleBone, http://exploringbeaglebone.com/.

Downloading and Analyzing Data Logs in Mission Planner — Copter Documentation. https://ardupilot.org/copter/docs/common-downloading-and-analyzing-data-logs-in-mission-planner.html.

“DroneePlotter - Drone Flight Log Analysis Tool Works on Web Browser.” Diydrones, 27 Oct. 2017, https://diydrones.com/profiles/blogs/droneeplotter-drone-flight-log-analysis-tool-works-on-web-browser.
EKF — Dev Documentation. https://ardupilot.org/dev/docs/ekf.html.

EKF2 Estimation System — Dev Documentation. https://ardupilot.org/dev/docs/ekf2-estimation-system.html.

Electronic Speed Controller (ESC) Calibration — Copter Documentation. https://ardupilot.org/copter/docs/esc-calibration.html.

Emmet. “How to Setup Raspberry Pi Remote Desktop.” Pi My Life Up, 24 Apr. 2019, https://pimylifeup.com/raspberry-pi-remote-desktop/.

“Enabling OpenVPN Configuration/Autostart on Ubuntu - DZone Integration.” Dzone.Com, https://dzone.com/articles/enabling-openvpn-configuration-autostart-on-ubuntu.

Extended Kalman Filter Navigation Overview and Tuning — Dev Documentation. https://ardupilot.org/dev/docs/extended-kalman-filter.html.

“Ffmpeg - Raspberry Pi USB Webcam Stream to Computer Using Gstreamer.” Stack Overflow,https://stackoverflow.com/questions/36023598/raspberry-pi-usb-webcam-stream-to-computer-using-gstreamer.

“Five Ways to Run a Program On Your Raspberry Pi At Startup.” Dexter Industries, https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/.

Fixing Wifi Connectivity on Nvidia Jetson Nano: An Experience to Share. https://www.datatobiz.com/blog/fixing-wifi-connectivity-nvidia-jetson-nano/.

Gorordo, Ibai. IbaiGorordo/AWR1843-Read-Data-Python-MMWAVE-SDK-3-. 2019. 2021. GitHub, https://github.com/ibaiGorordo/AWR1843-Read-Data-Python-MMWAVE-SDK-3-.

Grinberg, Miguel. How to Build and Run MJPG-Streamer on the Raspberry Pi. http://blog.miguelgrinberg.com/post/how-to-build-and-run-mjpg-streamer-on-the-raspberry-pi.

Running a Flask Application as a Service with Systemd. http://blog.miguelgrinberg.com/post/running-a-flask-application-as-a-service-with-systemd.

Stream Video from the Raspberry Pi Camera to Web Browsers, Even on IOS and Android. http://blog.miguelgrinberg.com/post/stream-video-from-the-raspberry-pi-camera-to-web-browsers-even-on-ios-and-android.

Ground Effect Compensation — Copter Documentation. https://ardupilot.org/copter/docs/ground-effect-compensation.html#ground-effect-compensation.

Gst-Launch-1.0. https://gstreamer.freedesktop.org/documentation/tools/gst-launch.html?gi-language=c.

“GStreamer with USB Webcam on Nvidia Jetson Nano.” Stack Overflow, https://stackoverflow.com/questions/65492424/gstreamer-with-usb-webcam-on-nvidia-jetson-nano.

Guoan, Xiao. “Using WPA_Supplicant to Connect to WPA2 Wi-Fi from Terminal on Ubuntu 16.04 Server.” LinuxBabe, 23 Mar. 2017, https://www.linuxbabe.com/command-line/ubuntu-server-16-04-wifi-wpa-supplicant.

Gus. “Raspberry Pi Port Forwarding & Dynamic DNS.” Pi My Life Up, 27 Apr. 2015, https://pimylifeup.com/raspberry-pi-port-forwarding/.

Hardware Setup of XWR6843AOPEVM for Flashing Mode. https://dev.ti.com/tirex/explore/content/mmwave_automotive_toolbox_3_2_0/mmwave_sdk_ccs_projects/common/docs/hardware_setup/hw_setup_aop_modular_mode_flashing.html.

Henry-Stocker, Sandra. “How to Share Files between Linux and Windows.” Network World, 24 Apr. 2018, https://www.networkworld.com/article/3269189/sharing-files-between-linux-and-windows.html.

“How to Set Up an OpenVPN Server on a Raspberry Pi - DZone IoT.” Dzone.Com, https://dzone.com/articles/how-to-setup-an-openvpn-server-on-a-raspberry-pi.

“How To Setup Autorun a Python Script Using Systemd.” TecAdmin, 28 Sept. 2017, https://tecadmin.net/setup-autorun-python-script-using-systemd/.

“How To Use Systemctl to Manage Systemd Services and Units.” DigitalOcean, https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units.

“HowTo: Setup RaspberryPi with OpenVPN Client.” A Guy, Named Urban Jack, 2 Dec. 2013, https://urbanjack.wordpress.com/2013/12/02/howto-setup-raspberrypi-with-openvpn-client/.

“IBM AIX.” IBM Developer, https://developer.ibm.com/components/aix/.

imfatant. Imfatant/Test. 2017. 2021. GitHub, https://github.com/imfatant/test.

Index of /Data/Pkg/Windows/1.14.4. https://gstreamer.freedesktop.org/data/pkg/windows/1.14.4/. Accessed 12 June 2021.

Installing Darknet. https://pjreddie.com/darknet/install/.

Installing the Latest Python 3 on Raspberry Pi. https://samx18.io/blog/2018/09/05/python3_raspberrypi.html.

JetPack 4.5.1 Installation for Jetson TX2 NX on DSBOARD-NX2 | Forecr.Io. https://www.forecr.io/blogs/installation/jetpack-4-5-1-installation-for-jetson-tx2-nx-on-dsboard-nx2.

Kapoor, Chetan. “How to Install Python 3.8 on Raspberry Pi (Raspbian).” Installvirtual, 15 Oct. 2019, https://installvirtual.com/how-to-install-python-3-8-on-raspberry-pi-raspbian/.

Linux - Autostart OpenVPN in Systemd (Ubuntu) - Getting Started OpenVPN — VPNSecure. https://support.vpnsecure.me/articles/getting-started/linux-autostart-openvpn-in-systemd-ubuntu.

“Linux - FFmpeg Starting Manually but Not with Systemd on Boot.” Stack Overflow, https://stackoverflow.com/questions/62879992/ffmpeg-starting-manually-but-not-with-systemd-on-boot.

“Linux - Systemd - Giving My Service Multiple Arguments.” Super User, https://superuser.com/questions/728951/systemd-giving-my-service-multiple-arguments.

Managing Gyro Noise with the Static Notch and Dynamic Harmonic Notch Filters — Copter Documentation. https://ardupilot.org/copter/docs/common-imu-notch-filtering.html.

Manual Roll and Pitch Tuning — Copter Documentation. https://ardupilot.org/copter/docs/ac_rollpitchtuning.html. Accessed 12 June 2021.

Messages (Common) · MAVLink Developer Guide. https://mavlink.io/en/messages/common.html#DISTANCE_SENSOR. Accessed 12 June 2021.

MmWave Demo Visualizer. https://dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/3.5.0/.

NanoDano. “Creating Systemd Service Files.” DevDungeon, 16 Sept. 2016, https://www.devdungeon.com/content/creating-systemd-service-files.

Ngo, Van Chan. Channgo2203/Test. 2019. 2019. GitHub, https://github.com/channgo2203/test.

Piggybank. “NVIDIA Jetson: JetsonTX2 - Installation OS(Ubuntu 18.04 LTS) and SDK.” NVIDIA Jetson, Winter 2019, https://spyjetson.blogspot.com/2019/11/jetsontx2-installation-osubuntu-1804.html.

Pre-Arm Safety Checks — Copter Documentation. https://ardupilot.org/copter/docs/common-prearm-safety-checks.html.

“PuTTY Equivalant on a Raspberry Pi.” Particle, 29 Mar. 2017, https://community.particle.io/t/putty-equivalant-on-a-raspberry-pi/30961/14.

“Python 3.7.” NVIDIA Developer Forums, 19 June 2019, https://forums.developer.nvidia.com/t/python-3-7/76655/7.

“Python Datalogger - Using PySerial to Read Serial Data Output from Arduino.” Maker Portal, https://makersportal.com/blog/2018/2/25/python-datalogger-reading-the-serial-output-from-arduino-to-analyze-data-using-pyserial.

Rajaraman, Jyotsna. “How to Read and Write from Serial Port Using Raspberry Pi?” IoTEDU, 14 Apr. 2020, https://iot4beginners.com/how-to-read-and-write-from-serial-port-using-raspberry-pi/.

Raspberry Pi Serial Data Logger with Display. http://www.davesmotleyprojects.com/raspi/raspi-data-logger/raspi-data-logger.html.

Raspberry Pi UART Communication Using Python and C | Raspberry Pi. https://www.electronicwings.com/raspberry-pi/raspberry-pi-uart-communication-using-python-and-c.

“Raspberry Pi Web-Based Data Logger Using MySQL and PHP.” Electronics-Lab.Com, https://www.electronics-lab.com/project/raspberry-pi-web-based-data-logger-using-mysql-php/.

Reading the Analog Inputs (ADC) - BeagleBone. http://beaglebone.cameon.net/home/reading-the-analog-inputs-adc.

RPLidar A2 360 Degree Lidar — Copter Documentation. https://ardupilot.org/copter/docs/common-rplidar-a2.html.

Serial Port Configuration Options — Copter Documentation. https://ardupilot.org/copter/docs/common-serial-options.html.

Serial Ports / UART - BeagleBone. https://sites.google.com/a/cameon.net/beaglebone/home/serial-ports-uart.

Simple Object Avoidance — Copter Documentation. https://ardupilot.org/copter/docs/common-simple-object-avoidance.html.

Staff, Embedded. “Comparing Real-Time Scheduling on the Linux Kernel and an RTOS.” Embedded.Com, 25 Apr. 2012, https://www.embedded.com/comparing-real-time-scheduling-on-the-linux-kernel-and-an-rtos/.

“Start Mavproxy as a Service.” ArduPilot Discourse, 15 Apr. 2020, https://discuss.ardupilot.org/t/start-mavproxy-as-a-service/54987.

Startup Options — MAVProxy Documentation. https://ardupilot.org/mavproxy/docs/getting_started/starting.html.
Startup Scripts — MAVProxy Documentation. https://ardupilot.org/mavproxy/docs/getting_started/mavinit.html.

Such, David. “Reefwing Robotics: The Falcon DS1 - BeagleBone Blue Drone (Part 1).” Reefwing Robotics, 21 June 2020, https://reefwingrobotics.blogspot.com/2020/06/the-falcon-ds1-beaglebone-blue-drone.html.

Systemd - Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/linux/usage/systemd.md.

“The Many Ways of Getting Data Into Charts.” CSS-Tricks, 1 May 2019, https://css-tricks.com/the-many-ways-of-getting-data-into-charts/.

“This Drone w/ a BBBlue and ArduPilot/ArduCopter.” BeagleBoard Projects, https://beagleboard.org/p/functt/this-drone-w-a-bbblue-and-ardupilot-arducopter-0ee27b.

“Tutorial - Auto Run Python Programs on the Raspberry Pi.” Dexter Industries, https://www.dexterindustries.com/howto/auto-run-python-programs-on-the-raspberry-pi/.

UART at Raspberry Pi GPIO Pinout. https://pinout.xyz/pinout/uart.

UARTs and the Console — Dev Documentation. https://ardupilot.org/dev/docs/learning-ardupilot-uarts-and-the-console.html.

Untitled. https://dev.ti.com/tirex/explore/.

Using MAVExplorer for Log Analysis — Dev Documentation. https://ardupilot.org/dev/docs/using-mavexplorer-for-log-analysis.html

Video Streaming Raspberry Pi Camera | Random Nerd Tutorials. 15 Aug. 2017, https://randomnerdtutorials.com/video-streaming-with-raspberry-pi-camera/.

Website:Follow, scottkildallScott Kildall. “Raspberry Pi: Launch Python Script on Startup.” Instructables, https://www.instructables.com/Raspberry-Pi-Launch-Python-script-on-startup/.

YOLOv4 on Jetson Nano. https://jkjung-avt.github.io/yolov4/.

REFERENCES

1. IWR6843AOP Single-Chip 60- to 64-GHz mmWave Sensor Antennas-On-Package (AOP) datasheet - URL **https://www.ti.com/product/IWR6843AOP**
2. IWR14xx/16xx/18xx/68xx/64xx Industrial Radar Family Technical Reference Manual - URL **https://www.ti.com/product/IWR6843AOP**
3. Ardupilot Development Site - URL **https://ardupilot.org/dev/**
4. Beagle board official webpage – URL **https://beagleboard.org/blue**
5. Koubaa, Anis & Allouch, Azza & Alajlan, Maram & Javed, Yasir & Belghith, Abdelfettah & Khalgui, Mohamed. (2019). Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey. – URL **https://www.researchgate.net/publication/334028415_Micro_Air_Vehicle_Link_MAVLink_in_a_Nutshell_A_Survey**
6. A2 RP LIDAR Datasheet - URL **https://www.digikey.com/htmldatasheets/production/2801237/0/0/1/rplidar-a2-a2m7-a2m8.html**
7. UBLOX GPS Datasheet - URL **https://www.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_UBX-15031086.pdf**