Development of $\frac{1}{10}$th scale Autonomous vehicle: Motor controls and PCB design

Indian Institute of Technology, Madras.

Anudeep EE16B010

13 June 2020

Acknowledgements

I wish to express my sincere gratitude to Dr. Ramkrishna Pasumarthy, Associate Professor, Electrical Engineering for providing me an oppurtunity to do my project work in "Development of $\frac{1}{10}$th scale Autonomous vehicle: Motor controls and PCB design"

I sincerely thank Mr. Subhadeep Kumar, Ph.D scholar for his guidance and encouragement in carrying out this project work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Self-driving cars have attracted enormous attention from the automotive community in the last decade. The electrical and electronics sector has seen a boom in development of high capacity micro-scale brushed and brushless DC motors, servo motors, stepper motors, lithium-ion batteries with large mAh and smaller sizes, enhanced embedded microcontrollers, different kind of sensing devices for vehicles, faster and robust wireless communication channels and numerous other technologies to equip electric vehicles and make them capable of autonomous motion. While there is a lot of research and development going on around the world to make the dream of self-driving cars true, one of the significant challenges in between is to test control algorithms for the autonomous motion of these vehicles on the road. An alternative option is to develop a scaled-down testing facility to perform experiments and check the feasibility of the control algorithms. However, the final objective of developing such an experimental setup begins with the development of small scale electric vehicles. Numerous ready-to-use small scale electric vehicles, such as electric RC cars, are available in shops and markets. But, the hardware present in those vehicles is not suitable enough to implement autonomous control algorithms.

Further modifications are necessary or such vehicles need to be developed indigenously. Many researchers have followed these methodologies in the past. Georgia Tech's AutoRally[3], f1tenth[4] and JetsonHacks autonomous car[5], MIT's autonomous RC car[6] are few examples of small-scaled electric vehicles developed for autonomous motion. The common thing amongst all these vehicles is the developers modified the chassis to fit necessary hardware and added requisite controllers, sensors and communication devices but kept the chassis base, suspensions, steering mechanism and most importantly the driving and steering motor intact. Although these vehicles perform sufficiently well, they are solely developed for autonomous motion with utmost non-cooperative behaviour. More specifically, these vehicles are designed to fulfil some particular objective, such as free motion in a given track or autonomous racing, etc. So, precise control of either of the driving or steering motor is not necessary for these applications.

Usually, in electric RC cars, an RC receiver receives the command signal from a remote and passes on to the Electronic Stability Control (ESC) unit for the driving motor and directly to the steering motor as PPM signals. The ESC unit supplies the necessary signal to the driving motor, which is a brushed or brushless DC motor. The voltage conversion logic or circuitry of the ESC unit is explicitly essential for precise control of the driving motor as is the parameters of the DC motor. Neither of that information is usually available from the RC car manufacturers because those components are designed and manufactured for a much simpler purpose. Standard usage of these RC cars is as a toy or for hobby RC racing.

On the other hand, the steering motor is a servo motor and hence turns the steering wheels as per the duty cycle of the PPM signal. These servos are designed for high torque supply and fast responses. Precision and accuracy are humbly ignored since the control is upon a human user who modifies the input based on visual feedback. On the contrary, in the case of autonomous motion control precision and accuracy of the motor outputs is crucial, and the motors that come with the RC cars are not suitable for the purpose. Moreover, neither of these motors have the provision to add sensors for getting feedback. Without measurements of the outputs, it is not possible to design any controller for controlling the motion of the vehicles. Hence, both the driving and steering motors need to be replaced with precision motors exhibiting good performances.

For generating the driving torque in the small-scale electric vehicle to be developed, a micro-scale brushless DC motor of high load capacity was chosen for meeting the desired specifications and performances, namely Maxon ECX 22M motor. The motor consists of Hall sensors and a temperature sensor which requires a separate controller, Maxon DEC Module 50/5, to extract the feedback information. The controller sends back the angular velocity of the motor shaft to the connected microcontroller or embedded computer. The controller also receives PPM signals from the embedded computer or microcontroller and transmits it to the motor input terminals. For steering control, a Dynamixel XL-320 servo motor was chosen. It has an inbuilt PID controller for position control over a range of 300 degrees and has the provision for tuning the gain values. The motor is capable of holding the motor shaft at the desired position due to its high torque capacity and also has a fast response time. The motor provides feedback of position, input voltage, load at the motor, the temperature within the motor, and many other variables. The inputs for the motors are computed by an Nvidia Jetson Nano small board computer (SBC) and sent to an Arduino Due microcontroller. The Arduino Due microcontroller is tasked with sending the command signals from the SBC to the motors and receiving the sensor feedback from the motors and forward it to the SBC. In case of the driving motor, it is necessary to test the control, adaptability, communication of the ECX 22M motor and the DEC Module with the Arduino Due and develop circuitry for this purpose. However, in the case of the steering motor, the scenario is more complicated. The Dynamixel XL-320 motor communicates using a half-duplex asynchronous serial communication protocol which is not supported by an Arduino Due microcontroller. Therefore, before testing the compatibility of Dynamixel XL-320 motor with Arduino Due microcontroller, it is necessary to develop a solution for converting the information transmitted via I2C or UART or CAN protocol in Arduino Due to the half-duplex asynchronous serial communication protocol in Dynamixel XL-320 and vice versa.

On the other hand, the electric vehicle will be equipped with numerous sensors such as IMU, Ultrasonic sensors, Hall sensors, etc. as well as other electronic devices. All these devices have different power requirements while there is a single primary power source, i.e. the LiPo battery. For example, the ultrasonic sensors, hall sensors require 5V power supply while the Arduino Due microcontroller requires 7.4V power supply. This differences in power requirement is fulfilled by a dedicated power circuit. Also, all the sensors, microcontrollers, and SBC have a dedicated circuit for establishing communication amongst themselves. All together, these circuits brings in a large number of wires which is not suitable for several reasons such as power loss, difficulty in fault detection, providing ruggedness in the vehicle, etc. To overcome these problems, all the nuclear circuits and electrical paths are to be designed on a PCB board which will serve as the motherboard of the electric vehicle.

The objective of this project is (i) to study the features of Maxon ECX 22M motor and DEC 50/5 controller, (ii) to test the compatibility and of ECX 22M motor and DEC 50/5 controller with Arduino Due microcontroller, (iii) to develop a circuit for controlling a ECX 22M motor using an Arduino Due microcontroller via a DEC 50/5 controller module, (iv) to study the features of a Dynamixel XL-320 servo motor, (v) to develop a circuit for establishing communication between an Arduino Due microcontroller and a Dynamixel XL-320 motor, and (vi) to design a PCB board for all power and communication circuits.

1.1 Overview

- The protocols that are required for communicating with XL-320 servo motor are understood and incorporated in the Arduino Due embedded code. A test circuit is made on breadboard with Arduino Due microcontroller, Dynamixel XL-320 servo motor, SN74LS241 IC, connecting wires, and a resistor to test the hardware and software.

- A software is developed in the Arduino IDE for controlling Maxon ECX 22M brushless DC motor with Maxon DEC 50/5 motor controller. A stripboard circuit is made in which the motor and the controller are mounted for testing the hardware and software. This knowledge was helpful in designing a printed circuit board in the later part of my project.

- A level shifter circuit is designed and simulated on the LTspice® software. DC-DC converter circuits are simulated in WEBENCH® designer. The lower PCB houses the circuitry for the servo motor, Inertial measurement unit (IMU) and the upper one houses the motor controller, Arduino Due microcontroller, DC-DC converters, Ultrasonic sensors, Hall sensors, IR sensors, and other power and communication circuits. Schematic and layout is designed in Autodesk EAGLE software.

Chapter 2

Dynamixel XL-320 with Arduino Due

2.1 Introduction

A servo motor is a self-contained electrical device, that rotate parts of a machine with high efficiency and with great precision. It comprises of a control circuit, motor, shaft, amplifier, encoder and often a gear assembly. The output shaft of the motor can be moved to a particular angle, position and velocity which is not possible just with a regular motor. A servo motor uses a regular motor and couples it with a sensor for position feedback, and the motor is controlled by a controller which determines the input voltage considering the reference and the feedback signal. The purpose of the controller is to control the rotational or linear velocity and position of the motor shaft precisely.

Servo motors or servos are used for steering small scaled electric RC cars. For autonomous steering, the controller in the servo needs to rotate the motor shaft to a certain angular displacement and hold the shaft at that angle. Due to road disturbances and other disturbances from the vehicle and the tyres it becomes a challenging task. The controller needs to adjust the input to the motor continuously with time or at certain time intervals. Servo motors available on the RC cars are not reliable as their performance is subpar. They are designed for hobby racing which demands only fast response time.

Usually, these servos have a PID controller inside and a sensor for measurement of angular displacement. To improve the performance of the motor the gains of the PID controller needs to be tuned. The new values of gains can be determined using some algorithm which will require at least the position of the motor shaft. For better performances and increased precision, knowledge of load torque at the motor and absolute input voltage to the motors are also necessary. Neither the gains of the PID controller can be tuned nor the feedback values can be read from the servos which comes with RC cars. So, these motors cannot be used for achieving autonomous steering control in electric RC cars. A servo motor which allows to tune the parameters of the controller inside the servo over time and provides feedback of at least the shaft position is necessary for the purpose. The Dynamixel XL-320 servo motor was chosen to meet the requirements.

2.2 Dynamixel XL-320

The Dynamixel XL-320 is a robot smart actuator with a fully integrated DC Motor, Reduction Gearhead, Controller, Driver, Network in one small DC servo module. It is designed to operate in 'Wheel Mode' and 'Joint Mode'. In the *wheel mode*, it operates as a normal motor rotating a wheel with endless turns. On the contrary, in the *joint mode* it operates for position control of the motor shaft. It includes a small microprocessor inside which analyses the received signals and processes them into high frequency voltage pulses to the servo motor. It has the ability of defining the maximum torque that can be applied. This parameter can be used to avoid breaking mechanical parts. XL-320 is equipped with sensors for measuring the position of the motor shaft, angular velocity, input voltage, load on the shaft, internal temperature, etc. The sensor feedback can be read using a microcontroller or a computer. It consists of a PID controller and the controller gains can be tuned externally using a microcontroller. Moreover, the dimensions of the motor makes it fit suitably in a $\frac{1}{10}$th RC Car. An additional mechanical structure may be required to mount the motor properly in the chassis. However, the motor horn is designed for other robotic applications which is not capable of fitting the steering linkage with the spur gear at the end of the motor shaft. Hence, a horn has to be designed for connecting the XL-320 with the steering arm.

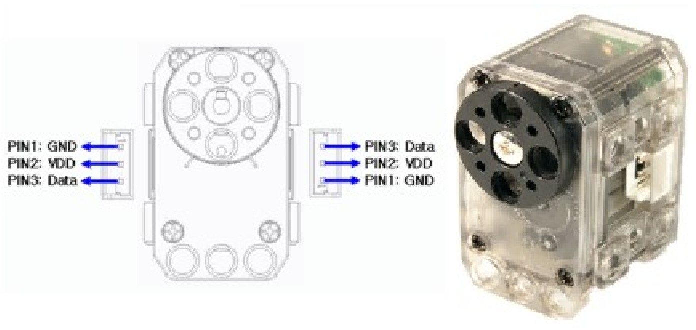


Figure 2.1: Dynamixel XL-320

The XL-320 model specifications are summarized below:

- Weight : 16.7g
- Dimension : 24mm x 36mm x 27mm
- Resolution : 0.29°
- Motor : Cored Motor
- Gear Reduction Ratio : 238 : 1
- Stall Torque : 0.39 N·m (at 7.4V)
- No load speed : 114 rpm (at 7.4V)
- Running Degree : 0° - 300°
- Running Temperature : -5°C - +70°C
- Voltage : 6 - 8.4V (Recommended Voltage 7.4V)
- Command Signal : Digital Packet
- Protocol Type : Half duplex Asynchronous Serial Communication
- Link (Physical) : TTL Level Multi Drop (daisy chain type Connector)
- ID : 253 ID (0-252)
- Communication Speed : 7343bps - 1 Mbps
- Feedback: Position, Temperature, Load, Input Voltage, etc.
- Material : Engineering Plastic

0xFF	0xFF	0xFD	0x00	ID	LEN_L	LEN_H	INST	Param1	...	ParamN
Header			Reserved	ID	Packek length		Instruction	Parameter		

Table 2.1: Communication protocol

The communication protocol used by the servo is as follows:

An Arduino Due microcontroller was selected for interfacing with most of the sensors present in the vehicle as well as for the driving and steering motor. So, the XL-320 motor is to be controlled using an Arduino Due microcontroller. The implementation of the protocol shown above in the Arduino Due microcontroller requires a library in order to establish communication with the servo. The functions to be written in the library should convert the input arguments into a format which the servo can process. Similarly, the feedback returned by servo has to be decoded back into readable data by the library. Arduino Due supports I2C, SPI, CAN communication protocols and UART Interface. On the other hand, XL-320 uses Half Duplex Asynchronous Serial Communication which can be interfaced via UART. Some of the aforementioned protocols and interfaces are described briefly in the following subsections.

2.2.1 UART

UART stands for 'Universal Asynchronous Receiver/Transmitter'. It is not a communication protocol like I2C, but a physical circuit in a microcontroller, or a stand-alone IC. The main purpose of a UART is to transmit and receive serial data. UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits. Parity bit helps in one bit error detection. Data is transmitted byte by byte. It supports lower data rate. 2.2 depicts two ICs communicating with UART Interface. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART.

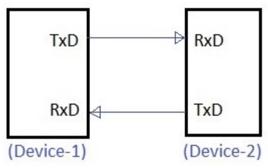


Figure 2.2: UART Interface

2.2.2 I^{2}C

I^{2}C stands for 'Inter-IC bus'. It is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line). It is used to run signals between ICs mounted on the same PCB (Printed Circuit Board). It uses only two lines between multiple masters and multiple slaves viz. SDA (Serial Data) and SCL (Serial Clock). It is synchronous unlike UART, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master I^{2}C supports various data rates ranging from 100 Kbps, 400 Kbps, 1 Mbps to 3.4 Mbps. Pull up resistors are necessary as shown in 2.3 when multiple devices load the bus.

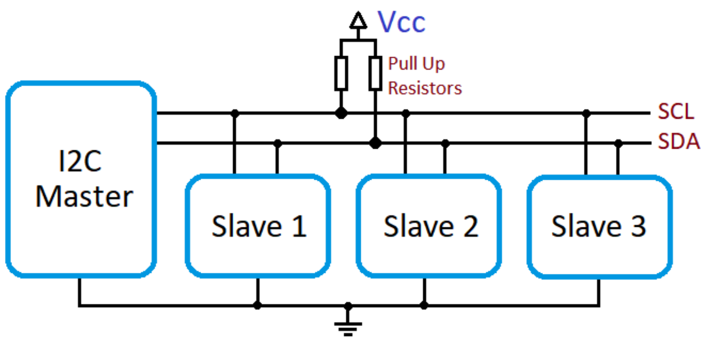


Figure 2.3: I^{2}C Interface.

2.2.3 Duplex Communication

A duplex communication system is a point-to-point system composed of two or more connected devices that can communicate with one another in both directions. Duplex communication systems are classified as (i) full-duplex communication system, and (ii) half-duplex communication system. The communication between sender and receiver occurs in both directions in Half duplex transmission, but only one at a time. The sender and receiver can both send and receive the information, but only one is allowed to send at any given time. For example, in walkie-talkies, the speakers at both ends can speak, but they have to speak one by one. They cannot speak simultaneously. A three terminal switch is required for communication between a Full duplex and a Half duplex device.

In Full duplex transmission mode, the communication between sender and receiver can occur simultaneously. The sender and receiver can both transmit and receive at the same time. For example, in a telephone conversation, two people communicate, and both are free to speak and listen at the same time. UART can be full duplex or half duplex. I^{2}C has a single data wire so it can perform only duplex communication. I^{2}C and UART can use simplex communication, in which one device transmits and the others can only listen.

Arduino Due supports Full duplex interface but XL-320 supports only half duplex interface so it is necessary to use additional circuitry to make communication possible between the Arduino Due and XL-320. We require an extra circuitry that behaves like a three terminal switch as shown in 2.4 connecting the XL-320 with the Arduino Due and thereby control the XL-320 motor using the Arduino Due microcontroller.

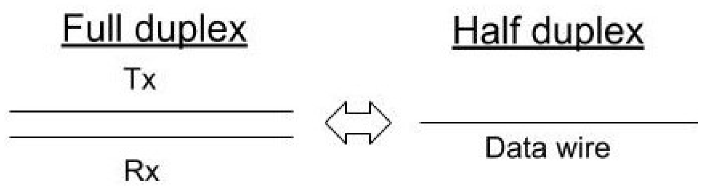


Figure 2.4: Three terminal switch

2.3 Hardware setup

Instead of a standard Serial connection, XL-320 uses a single wire, half-duplex serial. This makes communicating with the servo very difficult. We can use the OpenCM9.04 Microcontroller which is manufactured by the same company,Robotis, and is specially designed to interface with all Dynamixel Servos. However, both the microcontroller and the reference library provided in the OpenCM IDE are not fully functional. There were numerous problems with the microcontroller, including intermittent communication failures that seemed to be due to poor grounding on the controller, and poorly made electrical connections. I was unable to receive messages back from the XL-320. Walker Gosrich discussed a few approaches on using Arduino Due microcontrollers to communicate with the XL-320 servo motor, in his blog [2]. A few of them are discussed in the subsequent subsections.

2.3.1 No Extra Circuitry

This approach comes from Hackerspace Adelaide on Github. It implements a software-only attempt for sending to and receiving from the XL320, by simply wiring the TX and RX pins on the Arduino Due both to the XL320's data pin. Sending some commands to the XL320 was done successfully (LED blink). However, it doesn't work for implementing all commands in the XL320, and receiving feedback from the XL320 is not yet implemented. Overall, it serves as good skeleton code for sending commands to the XL320, and should work with a few fixes. However, receiving messages from the servo without any extra circuitry haven't been successfully implemented.

2.3.2 Passive Transistor Circuit

This approach comes from Nerd Ralph, and uses a passive and minimal single-transistor circuit to control communication with a half-duplex serial device. It works by cutting off the Arduino Due TX port when the half-duplex device (the XL320) is transmitting, and cutting off the Arduino Due RX port when the TX is transmitting a '0', and 'idling' the circuit when the TX is transmitting a '1'. It works for transmitting to the XL320, whether the XL-320's transmissions will make it through the transistor if the TX voltage is not actively controlled high is not discussed.

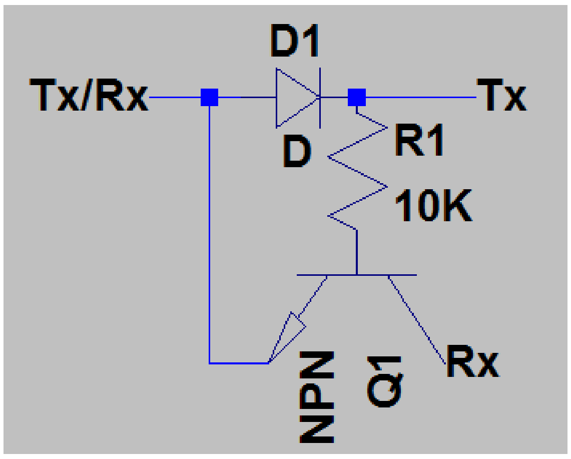


Figure 2.5: Passive Transistor [2]

2.3.3 Tri-State Buffer

Communication with the XL-320 was successfully made with the following method. The Half Duplex communication is realised with a tri-state buffer, 74LS241. The Data control pin is tied to one of the digital pins on the Arduino Due. When the data control pin is set to high value, the transmission channel turns on and commands can be sent to the motor. Setting the data control pin to low value turns on the receiver channel, thereby allowing feedback. This methodology will allow us to have full communication with the XL-320. The default value of the delay in the response of XL-320 motor is 0.5ms. Keeping this value to its default is suitable for proper communication between the Arduino Due and the XL-320. However, if the delay time is reduced to a low value then the Arduino Due might not be able to receive the feedback from XL-320.

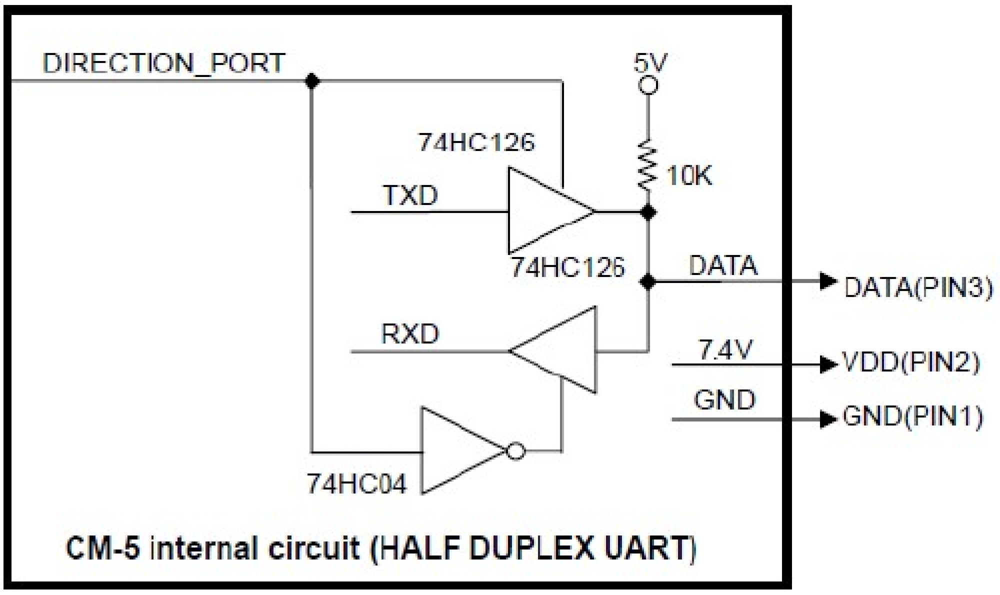


Figure 2.6: Internal Circuit of Half Duplex [1]

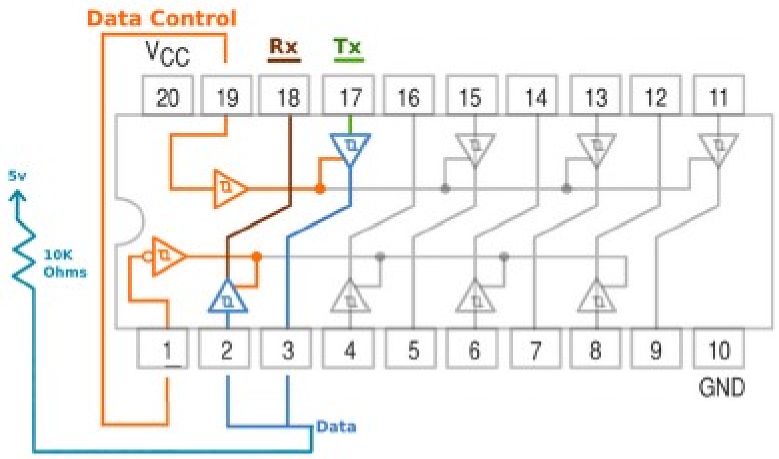


Figure 2.7: Tri state buffer, 74LS241[Sán16]

Arduino Due has four serial ports blue and three of them is required for this application. One port is for the XL-320, one for the real time clock (RTC) and the other is for serial monitor on the computer. The DC power supply present in the vehicle supplies a voltage of 7.4V. The 74LS241 IC requires 5V whereas the XL-320 motor requires 7.4V to function. Therefore, the 74LS241 IC and the XL-320 motor are powered separately. Arduino Due has an on-board voltage regulator which can be accessed with the V_{in} pin. 7.4V is stepped down to 5V by powering the V_{in} pin to 7.4V. So, the 74LS241 IC is connected to the 5V supply in the Arduino Due and the XL-320 motor is powered by a Battery.

Components:

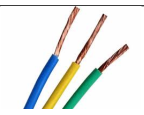
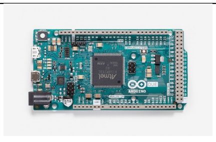
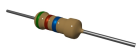
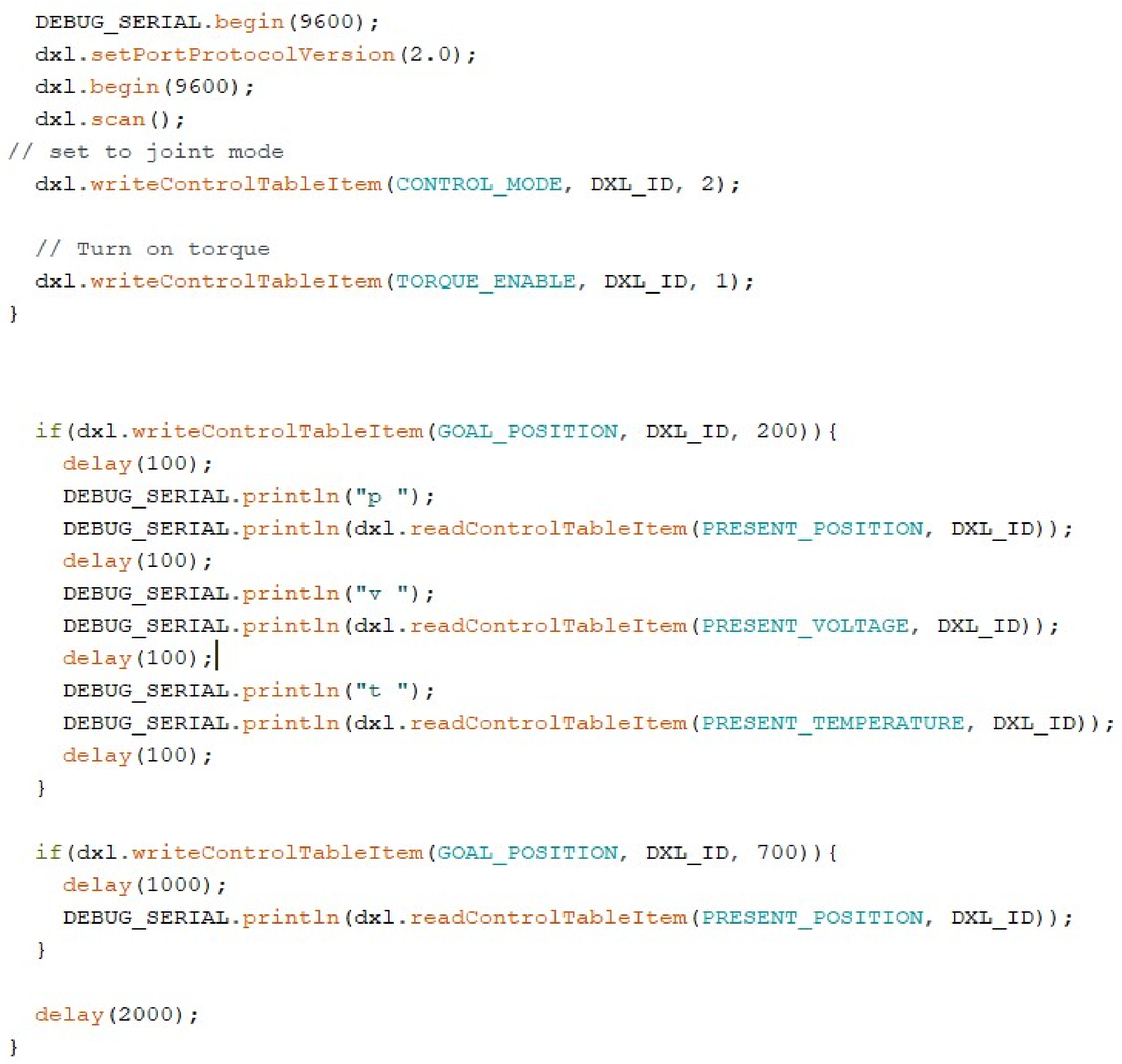
Components	Units	Description	Image
Dynamixel XL-320	1	Servo motor	
Copper wire	3	wires connecting Arduino Due and XL-320	
SN74LS241	1	Buffer for HD-ASC	
Arduino Due	1	Microcontroller	
Resistor	1	10 Kilo Ohm Resistor	

Table 2.2: Electronic Components list

A test circuit was made on a breadboard with a 74LS241 IC, a 10k resistor, connecting wires, an Arduino Due microcontroller and a XL-320 servo motor. The 74LS241 IC in 2.7 connects the Arduino Due microcontroller to the XL-320 servo motor. The intermediate circuit acts as a switch connecting the Arduino Due to the XL-320. Data Control is connected to a digital I/O pins on the Arduino Due . The Direction port which connects Tx of Arduino Due to Data wire of XL-320 should be turned ON before sending commands from Arduino Due to the XL-320. When the direction port is high, output of the inverter is low, which turns off the Rx buffer. A pull up resistor is added to avoid the data wire of XL-320 from floating. When the direction port is low, Rx buffer is switched ON, allowing data transfer from XL-320 to the Arduino Due.

2.4 Software setup

The microcontroller inside XL-320 consists of EEPROM and RAM for data storage. Instructions are given to the servo motor are stored in the RAM area. For example, the servo can be instructed to move to a position having 150 deg by changing the data at address ‘30’ to ‘512’. address ‘30’ corresponds to the parameter ‘Goal Position’. EEPROM data can only be changed when motor torque is disabled.   XL-320 can handle a maxi-



```
  DEBUG_SERIAL.begin(9600);
  dxl.setPortProtocolVersion(2.0);
  dxl.begin(9600);
  dxl.scan();
// set to joint mode
  dxl.writeControlTableItem(CONTROL_MODE, DXL_ID, 2);

  // Turn on torque
  dxl.writeControlTableItem(TORQUE_ENABLE, DXL_ID, 1);
}



  if(dxl.writeControlTableItem(GOAL_POSITION, DXL_ID, 200)){
    delay(100);
    DEBUG_SERIAL.println("p ");
    DEBUG_SERIAL.println(dxl.readControlTableItem(PRESENT_POSITION, DXL_ID));
    delay(100);
    DEBUG_SERIAL.println("v ");
    DEBUG_SERIAL.println(dxl.readControlTableItem(PRESENT_VOLTAGE, DXL_ID));
    delay(100);
    DEBUG_SERIAL.println("t ");
    DEBUG_SERIAL.println(dxl.readControlTableItem(PRESENT_TEMPERATURE, DXL_ID));
    delay(100);
  }

  if(dxl.writeControlTableItem(GOAL_POSITION, DXL_ID, 700)){
    delay(1000);
    DEBUG_SERIAL.println(dxl.readControlTableItem(PRESENT_POSITION, DXL_ID));
  }

  delay(2000);
}
```

mum data transfer speed of $1Mbps$. A few parameters are to be set before initiating the communication with XL-320.

- Step 1:The protocol version should be set to 2.0.
- Step 2:The baud rate should be set to 9600.
- Step 3:The control mode should be kept in Joint mode of operation.
- Step 4: Torque mode should be turned ON.

Dynamixel2Arduino library is installed in the Arduino Due IDE. It has *dxl* class.

- *setPortProtocolVersion* function sets the protocol to 2.0.
- *writeControlTableItem* and *readControlTableItem* are functions in the *dxl* class.
- *writeControlTableItem* function writes data to the specified address.

- *readControlTableItem* function reads data from the specified address.
- *dxl.writeControlTableItem(GOAL_POSITION,DXL_ID, 200)* moves the motor shaft's *GOAL POSITION* to 200.
- *dxl.readControlTableItem(PRESENT_POSITION, DXL_ID)* returns feedback data of the motor's *PRESENT POSITION*.
- The arguments of *readControlTableItem* are changed to *PRESENT_VOLTAGE*, *PRESENT_TEMPERATURE* to get feedback data of the Voltage and Temperature respectively.

writeControlTableItem and *readControlTableItem* follow a protocol to write or read data from the RAM area. The Header field indicates the start of the Packet. The Packet field indicates the ID of the Device that should receive the Instruction Packet and process it. *LEN1* and *LEN2* indicate the length after the Packet Length field (Instruction, Parameter, CRC fields). Packet Length = number of Parameters + 3. *INST* indicates whether the instruction is READ or WRITE etc. starting *PARAM* is the address at which the operation has to be executed and the final *PARAM* indicates the data to be written during the write instruction or the data read from the RAM area during the read instruction. CRC field checks if the Packet has been damaged during communication. 2.3 is the read protocol for reading Present Position. 2.4 is the write protocol for writing 512 to Goal Position.

H1	H2	H3	RSRV	ID	LEN1	LEN2	INST	P1	P2	P3	P4	CRC1	CRC2
0xFF	0xFF	0xFD	0x00	0x01	0x07	0x00	0x02	0x84	0x00	0x04	0x00	0x1D	0x15

Table 2.3: Read Protocol

H1	H2	H3	RSRV	ID	LEN1	LEN2	INST	P1	P2	P3	P4	CRC1	CRC2
0xFF	0xFF	0xFD	0x00	0x01	0x09	0x00	0x03	0x74	0x00	0x00	0x02	0xCA	0x89

Table 2.4: Write Protocol

The functions available in the *dxl* class take inputs for read and write, convert them into the aforementioned protocols to be processed by the XL-320. I adjusted the input parameters to change Goal Position, moving speed, disabling and enabling the Torque. I received feedback for Present Position, Temperature, Voltage, Speed. Read function takes two arguments, address of the parameter and the device ID. Write function takes three arguments, address of the parameter, device ID, data to be written. Decimal equivalent Addresses are predefined in the *dxl* parent class. Direction port is pulled high or low during Write or Read respectively. Library takes care of this by mentioning what pin on Arduino Due is assigned to Direction port.

2.5 Observations/Results

1. Dynamixel XL-320 servo motor is not fully functional with the OpenCM 9.04 Microcontroller from ROBOTIS.
2. Dynamixel XL-320 servo motor works with Arduino Due when connected via a SN74LS241 IC.
3. *Dynamixel2Arduino* Library sent commands and received feedback successfully.

Chapter 3

Driving Motor with Arduino Due

3.1 Introduction

The driving motor on the RC car is a brushed or brushless DC motor which provides torque to the wheels. It drives the vehicle forward and reverse. The motor is operated remotely through a remote which transmits the commands to the receiver on the car. An electronic speed control(ESC) is an electronic circuit plugged into the receiver's throttle control channel which converts the received PPM signals into corresponding PPM voltage and transmits it to the brushless DC motor. Its purpose is to vary the motor's speed and direction. It also allows smoother and more precise variation of the motor speed.

Due to lack of information regarding the operational characteristics of the motor and ESC, it is not possible to determine the necessary input to the motor in case of autonomous control. Also the motor has very low torque and high rpm values. It takes large current when running at low rpm with high torque. The ESC doesn't provide feedback and no sensor can be mounted externally. Moreover, the ESC does not have under-voltage and over-voltage protection. Also, there is no protection against reverse polarity.

An electronically commutated(EC) maxon motor is chosen, which has the desired torque characteristics and wide speed range. It is controlled by a 1-quadrant digital controller, DEC Module $50/5$. The controller is connected to Arduino Due which receives commands from Jetson Nano, an embedded computer.

3.2 Hardware Setup

3.2.1 Components

Driving Motor

Maxon ECX 22M is chosen to meet the desired requirements. It is a high power brushless electronically commutated DC motor rated at a torque of 20.3mNm and a angular velocity of $60,000$rpm. It has Hall sensors for measuring angular velocity and a thermistor for temperature sensing. It is a 1 pole pair machine and has a total of 10 connections 3.3, of which three are for motor windings, five are for hall sensors and two are for thermistor.

Figure 3.1: Maxon ECX 22M

Motor Specifications :

Parameter	**Value**	**Units**
Nominal Voltage	18	V
No load speed	50900	rpm
No load current	324	mA
Nominal speed	48200	rpm
Nominal torque	20.3	mNm
Nominal current	6.28	A
Stall torque	454	mNm
Stall current	135	A
Max. efficiency	90.6	%
Terminal resistance	0.133	Ω
Terminal inductance	0.00978	mH
Torque constant	3.37	mNm/A
Speed constant	2830	rpm/V
Speed/torque gradient	112	rpm/mNm
Mechanical time constant	2.53	ms
Rotor inertia	2.15	gcm^2
Number of pole pairs	1	NA
Number of phases	3	NA
Weight of motor	98	g

Table 3.1: Maxon ECX 22M brushless DC motor Specifications

Motor Controller

A Motor Controller is a device that acts as intermediary between a microcontroller and motors. A motor controller is necessary because a microcontroller can only provide $1A$ of current whereas the motor require several ampere of current. The current a motor consumes is related to the torque it can provide therefore a controller is chosen only after the motor is chosen.

DEC module $50/5$ was chosen to meet the desired requirements. It is a small 1-quadrant digital controller for the control of brushless DC motors (Electronic Commutated) rated up to 250W. Digital hall sensors of EC motors can be connected. It operates as *closed loop* or *open loop* speed controller. It has protective functions for current limit, thermal overload protection, under-voltage, over-voltage, voltage transients and short-circuits in the motor cable.

Figure 3.2: DEC Module 50/5

Controller Specifications :

Parameter	**Value**	**Units**
Operating voltage(Min)	6	V
Operating voltage(Max)	50	V
Output current(Max)	10	A
PWM clock frequency of power stage	46.8	kHz
Sampling rate PI speed controller	0.25	kHz
Max. efficiency	94	%
Max. speed Control	80000	rpm
Built-in motor choke per phase	0	μH
Weight	9	g

Table 3.2: DEC Module 50/5 Specifications

3.2.2 Circuit

On the vehicle, Jetson Nano is the small board computer where all the computations and decision making happens. It sends commands and takes feedback from the Arduino Due microcontroller which controls the DEC Module 50/5 connected to the driving motor. All the electronic components are powered by a $25.2V$ Li-Po battery. The Arduino Due microcontroller and the DEC Module 50/5 are mounted on a PCB. The brushless motor is mounted on the chassis of the vehicle. The Jetson Nano is placed on the top of the Vehicle.

In the test setup, a strip board circuit is made on which the motor and the DEC module are mounted. The DEC Module is connected to an Arduino Due via Jumper wires and the Arduino Due is connected to a Laptop. An External power supply powers the DEC Module and the Laptop powers the Arduino Due. The motor is tested at zero load condition.

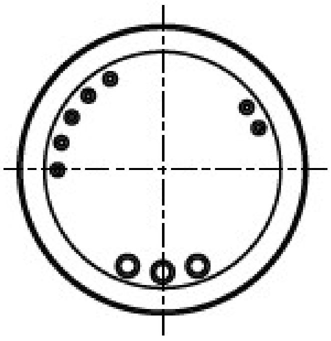


Figure 3.3: Pin diagram of Maxon ECX 22M brushless DC motor

Wire	Description
Red	Winding 1
Black	Winding 2
White	Winding 3
Orange	V_hall
Blue	Ground
Yellow	hall sensor 1
Brown	hall sensor 2
Grey	hall sensor 3
Purple wires(2)	Thermistor

Table 3.3: Maxon ECX 22*M* Pin description

Motor windings and the hall sensors are connected to the respective pins on the DEC Module 50/5. Thermistor acts like a variable resistor whose resistance is a function of temperature. It is connected to a digital I/O Pin on the Arduino Due where the effect of change in resistance is measured as a change in voltage.

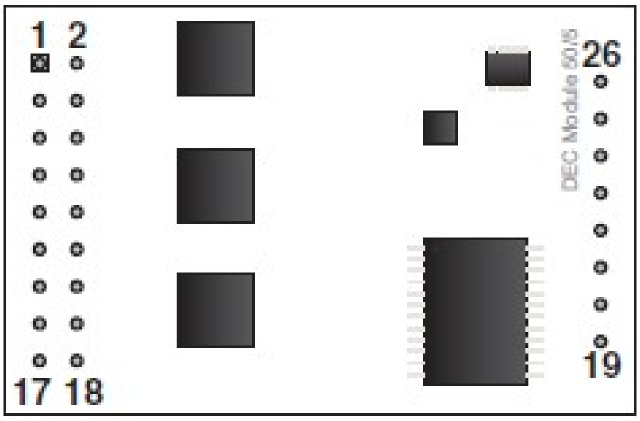


Figure 3.4: DEC Module 50/5 Pin Diagram

Pin	**Functionality**
1,2	Motor winding 1
3,4	Motor winding 2
5,6	Motor winding 3
7,8	Supply voltage 6...50 VDC
9,10	Ground
13	Hall sensor 1
15	Hall sensor 2
17	Hall sensor 3
14,16	Ground
18	Feed back pin, at what speed the motor is actually rotating
19	Status, whether the motor is ready or any error has occurred
20,21	To control the motor in open loop or closed loop
22	To enable or disable the motor
23	Control the rotational direction of motor
25	Set maximum current the motor can take
26	PWM input, instructing the motor

Table 3.4: DEC Module Pin description

Pins 18,26 are connected to analog I/O pins on the Arduino Due. Pins 19,20,21,22,23 are connected to digital I/O pins on the Arduino Due. Pin 25 is pulled to ground via a $56k$ resistor so that the continuous output current is limited to $7A$.

The digital I/O pins 20 and 21 on the DEC module are connected as follows to change the operating mode of the motor:

Digital Pin 20	**Digital Pin 21**	**Mode**
0	0	Open loop
1	0	$500-5000$ rpm
0	1	$500-20000$ rpm
1	1	$500-80000$ rpm

Table 3.5: Open loop and Closed loop

3.3 Software Setup

During the test, the signal given to the controller is 3.5. Max voltage is 5V and the duty cycle is set to 25%, so the mean voltage is 1.25. Velocity can be calculated as follows:

$$n = \left[\left(\frac{V_{set} - 0.1}{4.9} \right) (N_{max} - N_{min}) \right] + N_{min} \tag{3.1}$$

where Nmin, Nmax are the minimum and maximum values for a particular mode as shown in 3.5. For the test, the digital Pins 20 and 21 are set to ‘HIGH’ and $V_{set} = 1.25V$. The angular speed of the motor was found to be $19158.163rpm$.

Digital Pin 18 gives information on the actual angular speed(n) 3.2 of the motor shaft.

The actual angular speed is available as a digital frequency signal($f_{Monitor}$). z_{pol} refers to the number of pole pairs of the motor. $z_{pol} = 1$ for Maxon ECX 22M.

$$n = \frac{f_{Monitor} * 20}{z_{pol}} \tag{3.2}$$

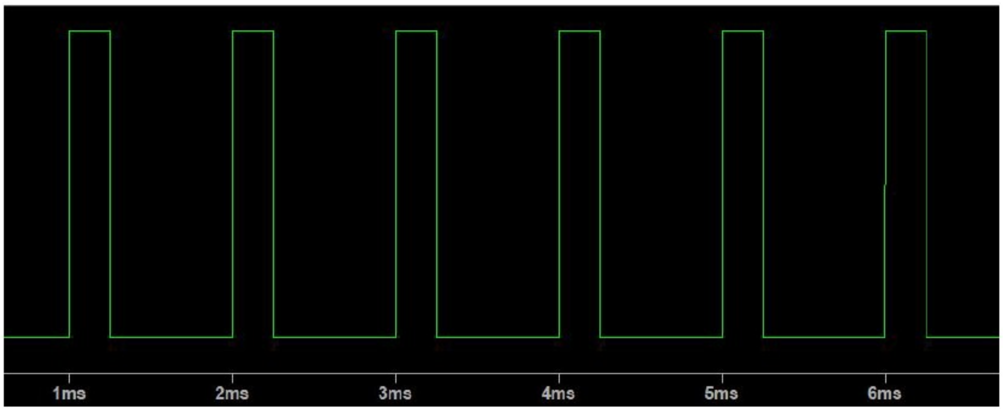


Figure 3.5: Input waveform

```
int digi_1= 8 ; //used for closed loop control
int digi_2= 9 ;
int enable = 10 ;
int speed_data=A9; //give PWM signal
int direction_pin= 11 ; //use this pin to change motor rotation
   direction
int feedback=A8; //frequency of this signal gives motor shaft
   velocity

void setup () {
Serial . begin ( 9600 );
digitalWrite(digi_1, HIGH );
digitalWrite(digi_2, HIGH );
digitalWrite(direction_pin, LOW );
digitalWrite(enable, HIGH );
}

void loop (){
analogWrite(speed_data, 50 );//set duty cycle
int data= pulseIn(feedback,HIGH);
Serial.println(data);
}
```

Variables *digi_1* and *digi_2* are for setting the operating mode of the motor. *enable* for turning ON the motor. *speed_data* is PWM input variable. *direction_pin* for changing the rotational direction of the motor. *feedback* for taking feedback from the controller. *analogWrite(speed_data,50)* generates a steady rectangular wave of the specified duty cycle. *pulseIn(feedback,HIGH)* waits for the voltage at specified pin to go from LOW to HIGH, starts timing, then waits for the voltage to go LOW and stops timing, returns the length of the pulse in microseconds.

3.4 Observations/Results

1. Maxon ECX 22*M* motor is successfully connected with the Maxon DEC Module $50/5$ controller.

2. The Software developed for the test circuit sent commands and received feedback successfully.

Chapter 4

PCB Design

4.1 Introduction

A printed circuit board (PCB) mechanically supports and electrically connects electrical or electronic components using conductive tracks and pads. A PCB has one or more sheet layers of copper laminated on and in between the sheet layers, they are insulated by a non-conductive substrates like *FR4*. Components are generally soldered onto the PCB to both electrically connect and mechanically fasten them to it.
On a printed circuit board, the interconnection between large number of components is made through copper tracks instead of using a number of current carrying wires. It makes the interconnections less bulky. The components are very small in size. It is impossible to connect these components together with wires without the aid of printed circuit boards. The components on a printed circuit board held fixed to the board. This is done by solder flux which does not allow them to move irrespective of the movement of the car itself. A PCB gives less electronics noise when properly laid. This is crucial as the sensors in the car are sensitive to electrical noise. The large ground planes on the PCB dissipate the generated heat thereby increasing the performance of the electronics. The placement of a component on the board is based on its Utility. As an example, Ultrasonic sensor is placed on the front so that an undesired object doesn't hinder the field of view. The dimensions of the chassis decide the dimensions of the board.
The electric vehicle will be equipped with numerous sensors such as IMU, Ultrasonic sensors, Hall sensors, etc. as well as other electronic devices. All these devices have different power requirements while there is a single primary power source, i.e. the $25.2V$ LiPo battery. For example, the ultrasonic sensors, hall sensors require $5V$ power supply while the Arduino Due microcontroller requires $7.4V$ power supply. This differences in power requirement is fulfilled by a dedicated power circuit. Also, all the sensors, micro-controllers, and SBC have a dedicated circuit for establishing communication amongst themselves. All together, these circuits brings in a large number of wires which is not suitable for several reasons such as power loss, difficulty in fault detection, providing ruggedness in the vehicle, etc. To overcome these problems, all the nuclear circuits and electrical paths are to be designed on a PCB board which will serve as the motherboard of the electric vehicle.

4.2 Design

The electronic components in the car are Jetson nano embedded computer, Maxon ECX $22M$ motor, Maxon DEC Module $50/5$ controller, Arduino Due microcontroller, DC-DC converters, Dynamixel XL-320 servo motor, SN74LS241 IC, Ultrasonic sensors, Hall and IR sensors, Analog IMU, Xsens IMU, RTC, Li-Po battery, MOSFETs, Diodes, Resistors, Capacitors and Switches.
Dynamixel XL-320 servo motor sits on the chassis of the car so a Lower PCB is designed on which the components related to the servo motor are mounted. The Lower PCB consists of a connector for the Dynamixel XL-320 servo motor, SN74LS241 IC, Analog IMU, Xsens IMU and a connector for power and data transfer from the Main PCB. The Lower PCB's dimensions are such that it fits on the front portion of the chassis of the car i.e, $Xdimension = 99.334mm$, $Ydimension = 42.354mm$.
The Main PCB consists of Arduino Due, Ultrasonic sensors, Hall and IR sensors, RTC, MOSFETs, Diodes, Resistors, Capacitors, Switches placed to one side and the power electronics components like DC-DC converters, Maxon ECX $22M$ motor, Maxon DEC Module $50/5$ controller, Li-Po battery connector placed to the other side to avoid interference and heating issues. Ground planes are made on the top and bottom side of the PCB for better heat dissipation. The Main PCB's dimensions are such that it fits on top of the chassis of the car i.e, $Xdimension = 177mm$, $Ydimension = 88mm$.
Design Net class parameters are:

- Width=$12mil$
- drill=$20mil$
- clearance=$10mil$

Design rules are:

- Copper thickness=$0.035mm$
- Sheet layer isolation=$1.5mm$
- Thermal isolation for vias=$10mil$

Different Signals			
X	Wire	Pad	Via
Wire	$10mil$	-	-
Pad	$10mil$	$6mil$	-
Via	$10mil$	$10mil$	$10mil$
Same Signals			
X	Smd	Pad	Via
Smd	$6mil$	$6mil$	$6mil$

Table 4.1: Design rules

4.3 DC-DC converters

A DC-to-DC converter is an electronic circuit or electromechanical device that converts a source of direct current (DC) from one voltage level to another. There is only one source to power all the electronics that operate at different voltages so two DC-DC converters are designed for powering Jetson nano, Servo and Arduino Due. Input power is supplied from a Battery rated at $25.2V$. The designs are taken from Texas Instruments WEBENCH® circuit designer. Resettable fuses are used to filter current transients and for protection. Chose the appropriate resettable fuses based on power requirements. Add a safety switch for the converter to enable/disable it.

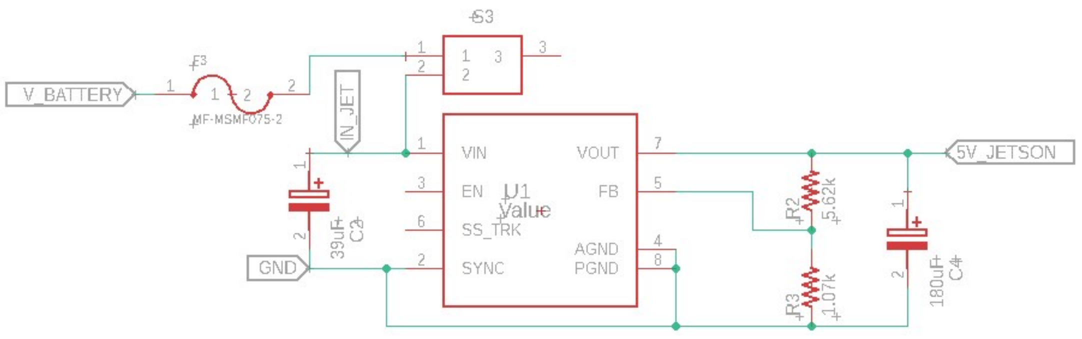


Figure 4.1: DC-DC converter for Jetson nano

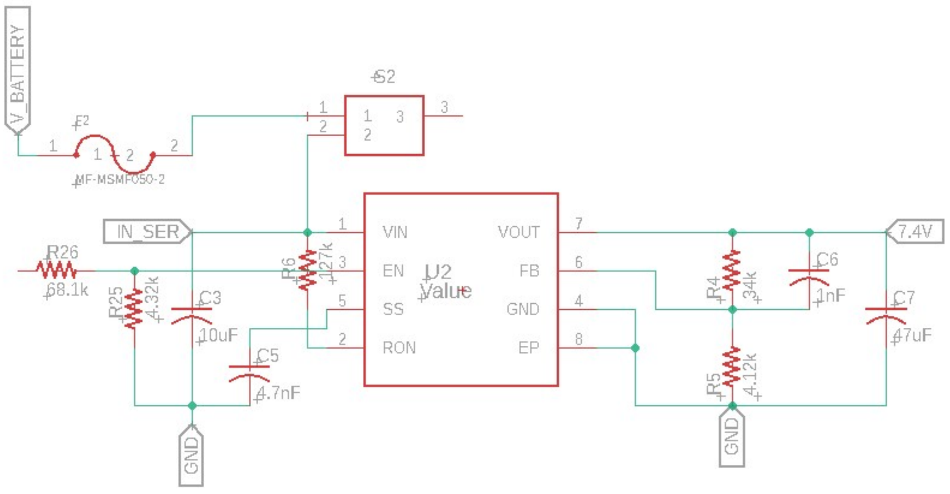


Figure 4.2: DC-DC converter for Arduino Due and Servo

4.4 Arduino Due

Arduino Due controls the Dynamixel XL-320 servo motor, Maxon DEC Module 50/5 controller, sensors and relays the feedback data to the Jetson nano. It is powered by supplying 7.4V at the Vin pin. Take the mirror image of the Arduino Due while placing the connector that holds it.
Digital pins 2-7, Analog pins 6, 7 are used for the DEC module.

Following is the naming convention I used for the connections between Arduino Due and Maxon ECX 22M :

- MOT_DATA_D7 refers to the feedback pin.
- READY_D6 refers to the status pin.
- IN_1_D5 is the digital input 1.
- IN_2_D4 is the digital input 2.
- EN_D3 is the enable pin.
- DIR_MOTOR_D2 is the direction pin.
- SET_A7 gives commands to the maxon motor.
- TEMP_A6 is for measuring temperature from the thermistor.

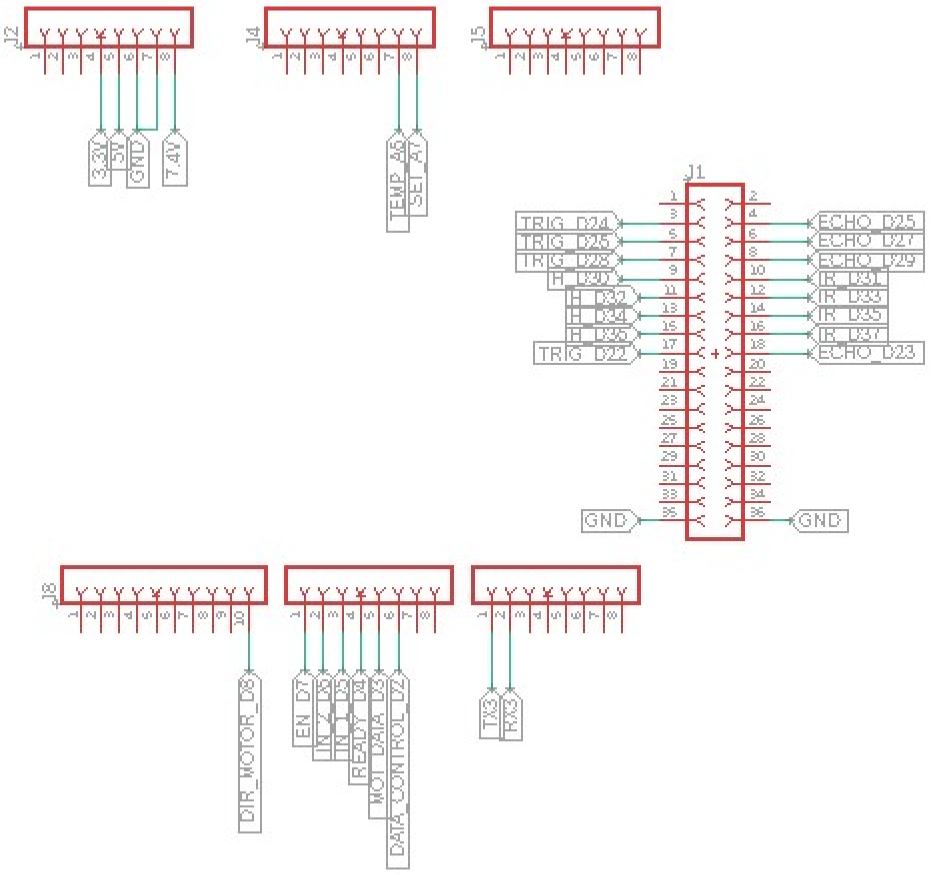


Figure 4.3: Arduino Due

4.5 Dynamixel XL-320 Servo motor

Digital pin 8 and a serial port(*TX3,RX3*) is used for the Dynamixel XL-320 servo motor. The Servo motor is placed on the lower board.

- DATA_CONTROL_D8 controls the serial to TTL protocol.
- TX3 and RX3 are the serial port pins for the XL320 motor.

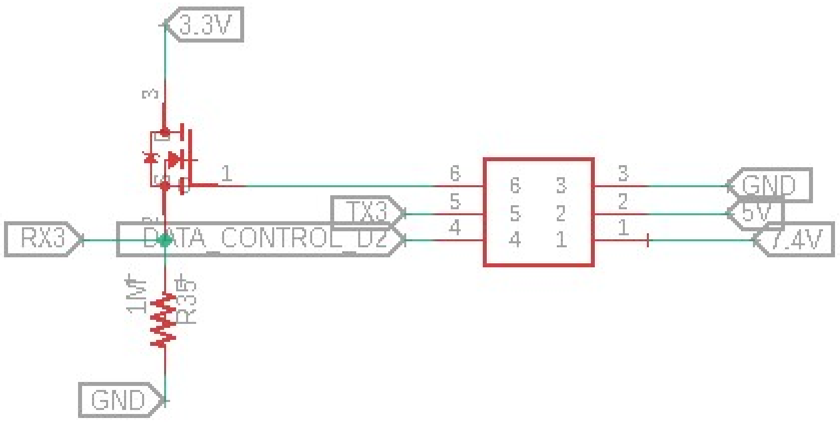


Figure 4.4: Connector for Servo

4.6 Maxon ECX $22M$ Motor and Controller

Transient voltage suppressing zener diode rated for $25.2V$. Fuse rated for a current of $7A$ and a decoupling capacitor of $220uF$ are used. Positive temperature coefficient resettable fuses are used.

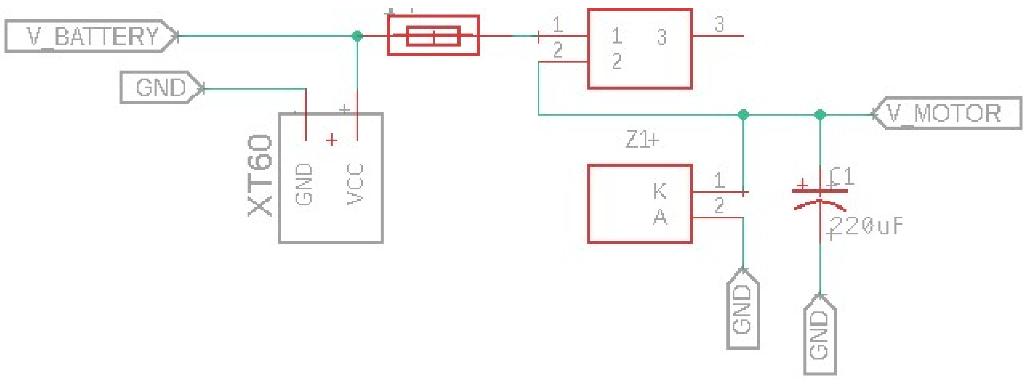


Figure 4.5: Battery Input

- The controller feedback is available at pin 18 which has to be connected to Arduino Due via a level shifter.
- Current limiter pin is tied to gnd with a 56k resistor which limits the current at 7A.

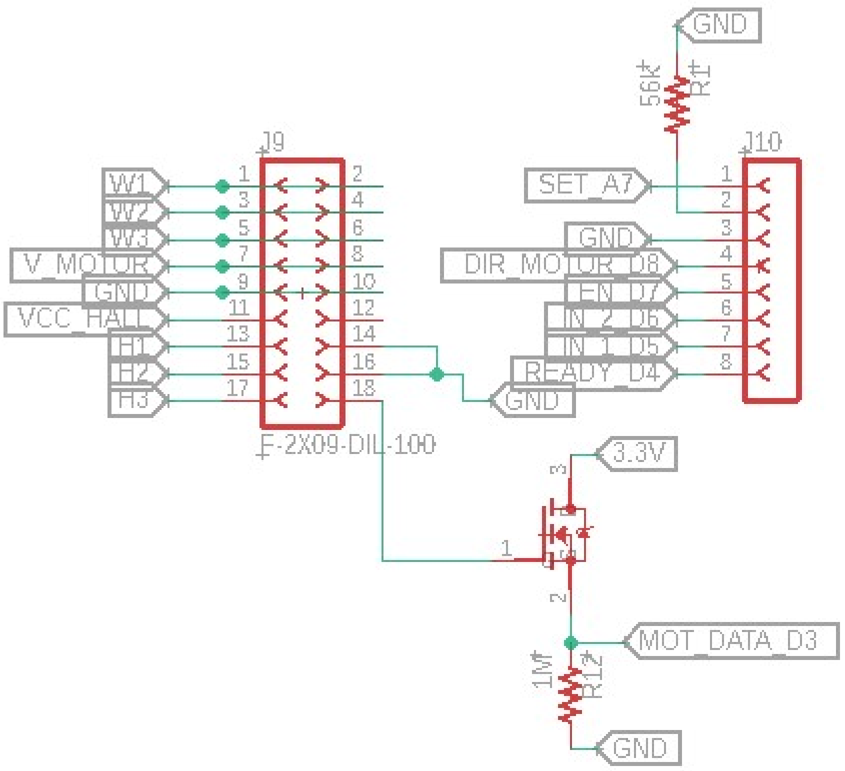


Figure 4.6: DEC Module 50/5connector

Layout guidelines for Maxon ECX $22M$ **motor :**

- The width and copper plating thickness of the power supply voltage and motor winding traces depend on the maximum current expected in the application.A minimum of 75 mil width at $70\mu m$ thickness is recommended.
- Make Top and Bottom planes as ground with polygons. This ensures proper heat dissipation.
- Place ground vias connecting top and bottom planes.
- The main PCB powers the Jetson nano so an output port has been placed for powering the Jetson.
- High power circuits viz; Motor and DC-DC converters are placed to a side on the board so as to avoid the inference with the low power circuits.

4.7 Ultrasonic sensor

Digital pin 22-29 on the Arduino Due are used for the ultrasonic sensor.

4.8 Hall and IR Sensors

Digital pins 30-37 on the Arduino Due are used for Hall and IR sensor.

4.9 RTC

RTC uses the I^2C protocol. It uses SDA1 and SCL1. For transferring the data from RTC to Jetson, a data port is placed.

4.10 Level shifter

Feedback pins on the DEC module and the data pins on the sensors operate at 5V but the pins on the Arduino Due are 3.3V tolerant so we need a Level shifter to stepdown the PWM data. I have used an N-MOSFET and a resistor for the Level shifter design. The data pin is connected to the gate of the NMOS, drain is connected to 3.3V voltage source and the source is pulled to ground via a 5 Kilo ohm resistor. Arrival of data switches ON the NMOS, allowing the current to follow from drain to source pulling the source to 3.3V.

Below is the schematic I used to test the circuit. The Input and Output waveforms are plotted below.

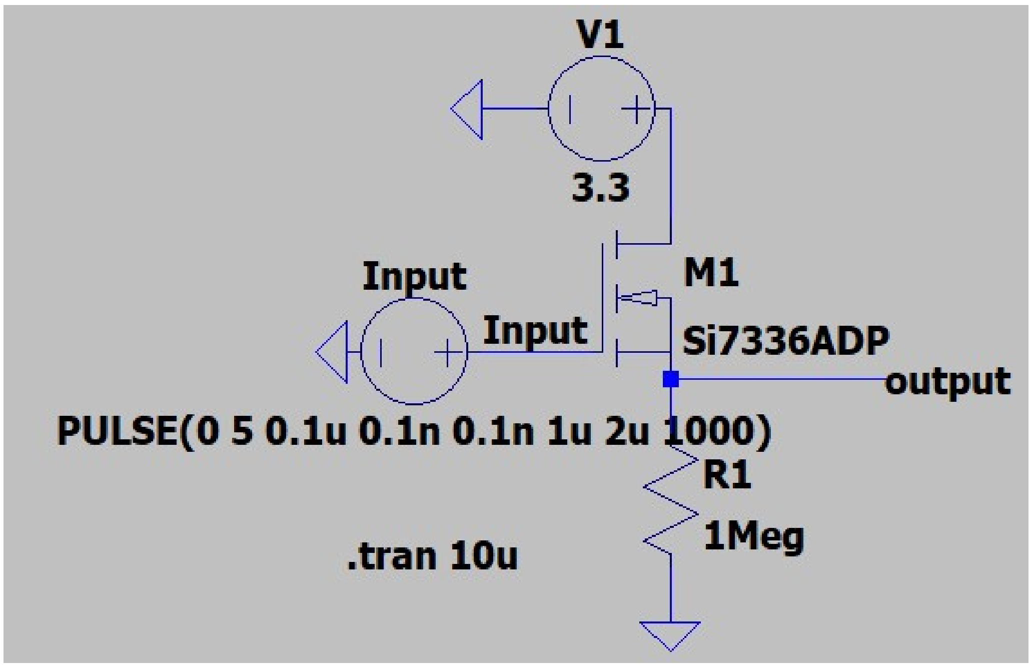


Figure 4.7: Level shifter simulation

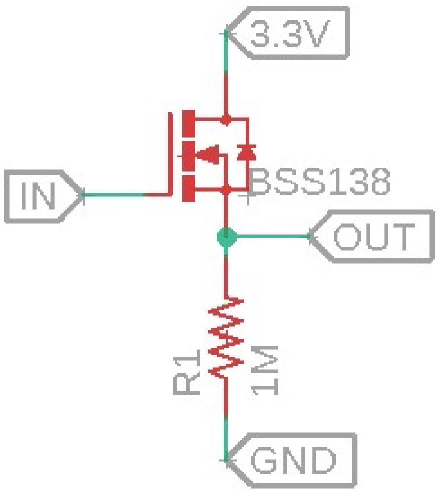


Figure 4.8: Level shifter schematic

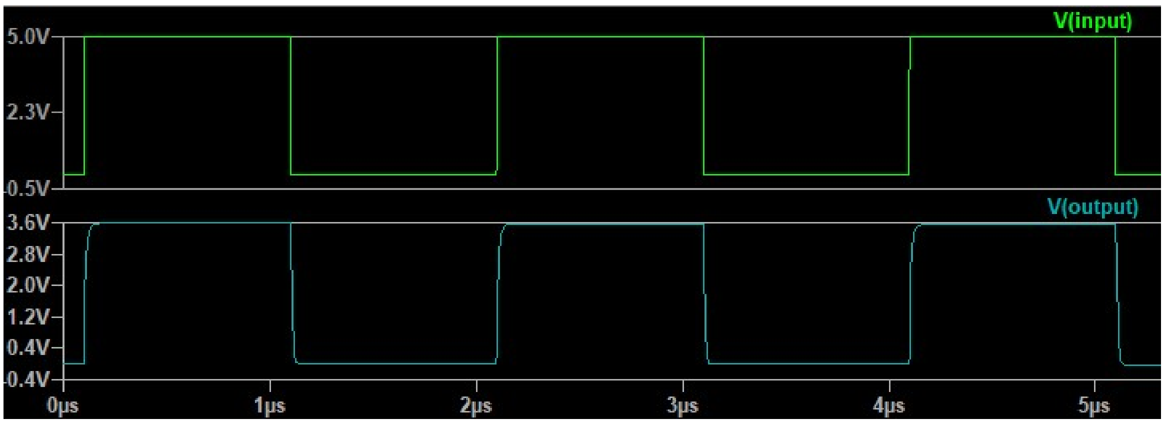


Figure 4.9: Simulated Waveform

References

[Sán16] Sergio Sánchez Gallego. “Mechanical and control design of an active coupling for a teleoperated surgical instrument”. MA thesis. Universitat Politècnica de Catalunya, 2016.

[1] 1. *Dynamixel*. URL: `http://emanual.robotis.com/docs/en/dxl/x/xl320/`.

[2] 2. *Walker’s Blog*. URL: `http://walker.gosrich.com/posts/xl320.html`.

[3] 3. *AutoRally*. URL: `https://autorally.github.io/`.

[4] 4. *F1TENTH*. URL: `https://f1tenth.org/`.

[5] 5. *Jetson Based Autonomous Race Car*. URL: `https://www.jetsonhacks.com/2016/06/13/jetson-based-autonomous-race-car-university-pennsylvania/`.

[6] 6. *MIT’s Autonomous RC Car*. URL: `https://mit-racecar.github.io/`.