

Grip Force Measurement For Driver Fatigue Detection

A Thesis

Submitted By

SACHIN SHARMA

for the award of the degree

of

MASTER OF TECHNOLOGY

Under the guidance of

Prof. Jagadeesh Kumar V



**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS**

May 2014

THESIS CERTIFICATE

This is to certify that the thesis titled “**Grip Force Measurement For Driver Fatigue Detection**”, submitted by **Mr. Sachin Sharma**, to the Indian Institute of Technology Madras, Chennai for the award of the degree of **Master of Technology**, is a bona fide record of research work done by him under my supervision. The contents of this thesis, in full or a part has not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Jagadeesh Kumar V

Research Guide

Professor (CEC Head)

Dept. of Electrical Engineering

IIT-Madras, Chennai-600036

Place: Chennai

Date: 23 May 2014

ACKNOWLEDGEMENTS

I express my sincere and heartfelt gratitude to my guide, Dr. Jagadeesh Kumar V, Professor (CEC Head) for giving me this highly rewarding opportunity to work with him. His courses and insightful assignments helped me in understanding the finer aspects of Sensor selection and using them. His suggestions were sound and timely, which ensured the smooth progress of this work. His eye for detail, patience, meticulous nature, devotion to work is always a source of inspiration for me.

I am extremely grateful to Dr. Bobby George, Assistant Professor, Department of Electrical Engineering, IIT Madras, for his moral support and encouragement.

I thank all the teaching and non-teaching staff of the Department especially from Measurements and Instrumentation Lab, for their great help and also thank to my friends Rahul Bharadwaj, Rahul Tyagi, Anurag Singh, Suaib Danish, Piyush Kumar, Gaurav Chandrakar, Aditya Arya, Rohit Kalla, Ritesh Pal Singh and Aniket Anil More, for giving me their valuable time in taking data from the sensors and for their timely assistance in the completion of this work.

Last but not the least, I would like to thank Almighty God, My parents and family for their support and for uncountable blessings due to which I was able to complete the project on time.

Sachin Sharma

EE12M097

ABSTRACT

Statistics shows driver fatigue is a major cause of road accidents all around the world. This project presents a simple and reliable method of driver fatigue detection by continuously monitoring the driver's grip force on the steering wheel. Four FSR (Force Sensing Resistor) sensors attached to the steering wheel continuously measure the grip force on the steering wheel exerted by the driver. The sensor output is read by a microcontroller unit for storage and onward transmission to a personal computer. Data analysis is performed on the PC and warning signals are generated by detecting fatigue through the changes in the grip force. The alertness of the driver is assessed by using an algorithm based on ANOVA (Analysis of Variance) test. This sensor fusion technology can be used in future smart vehicles to prevent road accidents.

TABLE OF CONTENTS

LIST OF TABLES	vi
LIST OF FIGURES	vii
ABBREVIATIONS	viii

CHAPTER 1: INTRODUCTION

1.1. Need for Measurement System.....	1
1.2. Literature Survey.....	2
1.2.1. Road Safety Activities undertaken during the Financial Year 2011.....	3
1.2.2. Initiatives of NHAI for Road Safety Activities.....	4
1.3. Objective and Scope of work	5

CHAPTER 2: HARDWARE AND DESIGN SPECIFICATION

2.1. Block Diagram.....	6
2.2. Hardware Selection.....	7
2.2.1. Gaming Console.....	7
2.2.2. Sensor.....	8
2.2.3. Arduino Due Board.....	9
2.2.4. Buzzer.....	10
2.2.5. LED.....	10
2.2.6. Connecters and Wires.....	10
2.3. Design Procedure.....	11
2.4. Constraints.....	12
2.5. Arduino Due features.....	13

CHAPTER 3: MEASUREMENT TECHNIQUE

3.1. Hardware Connections.....	18
3.2. Sensor Calibration.....	19
3.3. Programming.....	21

CHAPTER 4: DATA ANALYSIS

4.1. Testing Hardware.....	22
4.1.1. Testing an FSR.....	22
4.1.2. Testing an Arduino Due Board.....	23
4.1.3. Receiving Grip Force Signals.....	24
4.2. Matlab Graphical Analysis.....	25
4.2.1. Test Driver 1.....	25
4.2.2. Test Driver 2.....	27
4.3. Anova Test in Excel.....	28
4.3.1. Test Driver 1.....	28
4.3.2. Test Driver 2.....	29
4.3.3. Test Driver 3.....	29
4.3.4. Test Driver 4.....	30
4.3.5. Test Driver 5.....	30
4.3.6. Test Driver 6.....	31

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1. Conclusion.....	32
5.2. Future Scope.....	33
A. Op-Code Used in Arduino Programming.....	34
B. Syntax and Operators Used in Arduino Programming.....	38
C. Experimental Setup.....	41
D. Testing an Arduino Due Board.....	42
E. Data Logging in Matlab.....	43
REFERENCES.....	46
Index.....	47

LIST OF TABLES

Table 4.3.1: Test Driver 1 Single Factor Anova.....	28
Table 4.3.2: Test Driver 1 Two-Factor with Replication Anova.....	28
Table 4.3.3: Test Driver 2 Single Factor Anova.....	29
Table 4.3.4: Test Driver 2 Two-Factor with Replication Anova.....	29
Table 4.3.5: Test Driver 3 Single Factor Anova.....	29
Table 4.3.6: Test Driver 3 Two-Factor with Replication Anova.....	29
Table 4.3.7: Test Driver 4 Single Factor Anova.....	30
Table 4.3.8: Test Driver 4 Two-Factor with Replication Anova.....	30
Table 4.3.9: Test Driver 5 Single Factor Anova.....	30
Table 4.3.10: Test Driver 5 Two-Factor with Replication Anova.....	30
Table 4.3.11: Test Driver 6 Single Factor Anova.....	31
Table 4.3.12: Test Driver 6 Two-Factor with Replication Anova.....	31

LIST OF FIGURES

Fig 1.1.1: Causes of Road Accidents in India.....	2
Fig 2.1.1: Block Diagram.....	6
Fig 2.2.1: Gaming Console.....	7
Fig 2.2.2: FSR-406 Sensor.....	8
Fig 2.2.3: Arduino Due Board.....	9
Fig 2.2.1: Design Procedure.....	11
Fig 3.1.1: Hardware Connections.....	18
Fig 3.2.1: Sensor Calibration.....	20
Fig 4.1.1: Testing of an FSR Sensor.....	22
Fig 4.1.2: Testing of an Arduino Due Board.....	23
Fig4.1.3: Receiving Grip force Signals.....	24
Fig 4.2.1.1: Test Driver 1 Left Hand Plot.....	25
Fig 4.2.1.2: Test Driver 1 Right Hand Plot.....	25
Fig 4.2.1.3: Test Driver 1 Combined Plot.....	25
Fig 4.2.1.4: Zoomed version of Left Hand Plot.....	26
Fig 4.2.1.5: Zoomed version of Right Hand Plot.....	26
Fig 4.2.1.6: Test Driver 1 Zoomed Combined Plot.....	26
Fig 4.2.2.1: Test Driver 2 Left Hand Plot.....	27
Fig 4.2.2.2: Test Driver 2 Right Hand Plot.....	27
Fig 4.2.2.3: Test Driver 2 Combined Plot.....	27
Fig C.1: Experimental Setup.....	45
Fig C.2: Connections at Microcontroller Unit.....	45

ABBREVIATIONS

AC	Alternating Current
ADC	Analog to Digital Converter
AIMTC	All India Motor Transport Congress
AITWA	All India Transporter's Welfare Association
ARM	Advanced RISC (Reduced Instruction Set) Machine
ATMS	Advanced Traffic Management System
CEC	Central Electronic Centre
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DC	Direct Current
DMA	Direct Memory Access
ECG	Electrocardiography
EEG	Electroencephalography
EMG	Electromyography
FM	Frequency Modulation
FIAA	Federation of Indian Automobile Associations
FSR	Force Sensing Resistors
GND	Ground
IRF	International Road Federation
LED	Light Emitting Diode
MCU	Microcontroller Unit
N	Newton
NHAI	National Highways Authority of India
NHDP	National Highways Development Project
PTF	Polymer Thick Film

PWM	Pulse Width Modulation
R	Resistance
RAM	Random Access Memory
RC	Registration Certificate
ROM	Read Only Memory
RTO	Regional Transport Office
RX	Receiver
PC	Personnel Computer
SIAM	Society of Indian Automobile Manufacturers
SRAM	Static Random Access Memory
TTL	Transistor-Transistor Logic
TV	Television
TX	Transmitter
UART	Universal Asynchronous Receiver/Transmitter
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UT	Union Territories
V	Voltage
WIAA	Western Indian Automobile Association

CHAPTER 1

INTRODUCTION

1.1.Need for Measurement System

Driver fatigue has long been identified as one of the major causes of road accidents.

Thus, developing intelligent systems for assessing the driver's vigilance level is becoming a central issue in the field of active road safety research ^[1]. Such systems are based on the assumption that the occurrence of fatigue or drowsiness can be related to measurable changes in driver's state and behavior. A crucial point is that the methods employed to detect driver's fatigue must be reliable and non-intrusive ^[4].

The approaches presented in this literature can be grouped into two classes.

On one hand, there are methods based on signals from the driver ^[1]. These include physiological parameters, such as ECG, EEG and EMG, whose measurement usually requires electrodes to be applied to the driver. Other driver-related signals are eye movement, head positioning and facial expressions, which can be accessed using cameras and computer vision.

On the other hand, analyzing the vehicle's behavior including its speed, lateral position and distance from the other vehicle is monitored ^[2].

The measurement of driver-related signals generally results in more difficult and invasive.

1.2.Literature Survey

In India motor vehicle population is growing at a faster rate than the economic and population growth. The surge in motorization coupled with expansion of road network has brought with it the challenge of addressing adverse factors such as the increase in road accidents.

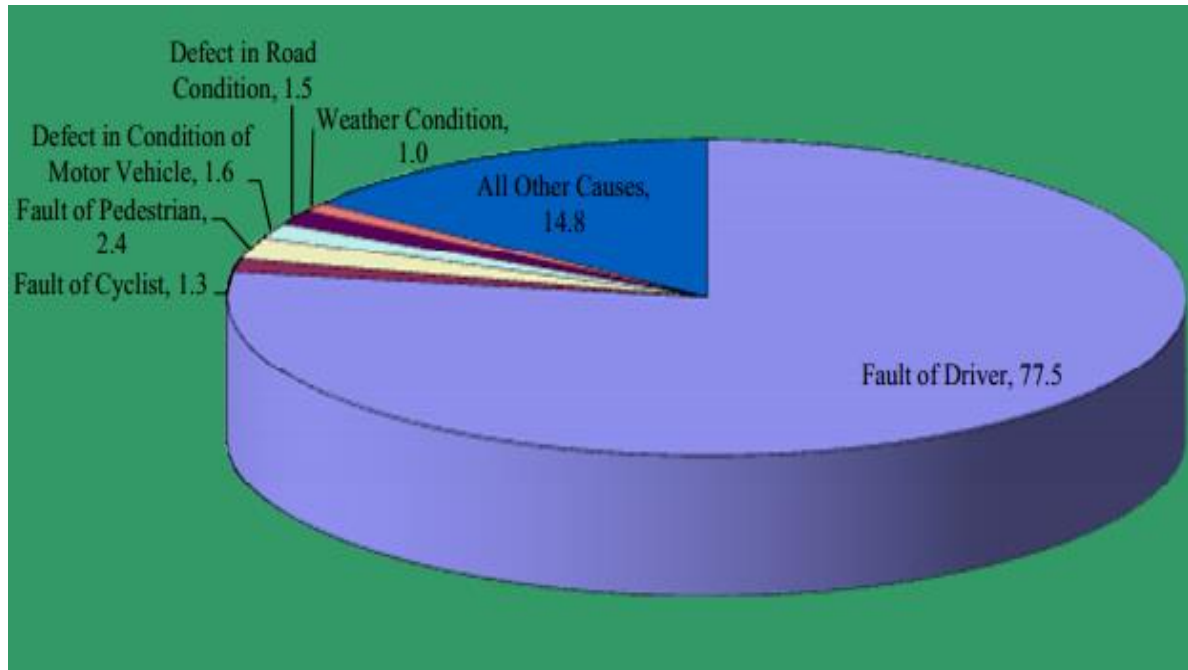


Fig 1.2.1: Causes of Road Accidents in India ^[3]

Road accidents are a human tragedy ^[3]. It involves high human suffering and socioeconomic costs in premature deaths, injuries, loss of productivity etc. It is heartening to note that there has been a marginal decline in road accidents during 2011.

However, the problem of road safety remains acute in India. During the year 2011, there were around 498k road accidents that killed 142k persons and injured more than 500k persons, many of them are disabled for the rest of their lives. These numbers suggest that few road accidents occur per minute, and each road accident death in every four minutes. Sadly, many of these victims are young people, those who are economically active.

The National Sleep Foundation also reported that 60% of adult drivers have driven while felling drowsy in the past year, and 37% have ever actually fallen asleep at the wheel ^[5].

1.2.1. Road Safety Activities undertaken during the Financial Year 2011-12 ^[3]

- The Ministry gave the slogan "Accidents bring tears, Safety brings Cheers" for the year.
- A National Road Safety Week was observed throughout the Country during January 1st to 7th, 2012 by involving all stakeholders.
- Media campaign was launched by the Ministry through the entire Doordarshan network including the Regional Centers and Regional channels, All India Radio, Vividh Bharati and all Regional stations, 35 Private TV Channels, Private FM radio stations throughout the country and in leading newspapers throughout the country with special emphasis on the Road Safety.
- The road safety material was dispatched in December 2011 consisting of children's activity books (for two age groups), road signage and posters in regional languages in 10,000 schools across the country in order to raise awareness of road safety. Calendars with Road Safety messages were distributed.
- All State Govts/ UTs and NHAI organized events for the Road Safety Week along with the stakeholders in their respective States such as SIAM, SRTUs, IRF, AIMTC, FIAA, WIAA, Tire Manufacturers and Auto Spare Parts Manufacturers.
- Valedictory Function to commemorate the successful observance of the 23 rd. Road Safety Week was held on Lal Chowk in Pragati Maidan on 9.12.2011 in association with SIAM and other stakeholders like AIMTC, AITWA, FIAA and IRF.

- For the first time, road safety campaign through online media i.e. Internet and SMS was carried out during 2011-12.
- Complete computerization of records of RTOs and issuance of driving license, RC and other documents on smart card through central assistance is being carried out. Approximately 95% of the work is complete linking most of the RTOs at the state level and linking of all state level records at the national level of national data base. Development of citizen friendly system - Vahan and Sarathi has already been already launched on 20th July 2011.
- A series of seminars/workshops across the country have been planned to sensitize all stakeholders and to work in an integrated manner. The First National Workshop on road safety was held at New Delhi on 03.04.2012. The theme was “improving the safety of more vulnerable road users”.

1.2.2. Initiatives of NHAI for Road Safety Activities^[3]

- Safety Measures are inbuilt in the projects during Design, Construction and O and M.
- State-of-the-Art Advanced Traffic Management System (ATMS) consisting of emergency call boxes, variable message signs, CCTVs, traffic counters cum classifiers, etc. has been provided on selected sections mostly under NHDP Phase V.
- The Comprehensive Road Safety Manual has been prepared and hosted on NHAI website.

1.3 Objective and Scope of Work

A promising approach is found in considering the data available at the interface between driver and the vehicle, i.e., the steering wheel. This may provide direct information about driver's state and can be easier been acquired.

In general, the grip force that a driver applied to the steering wheel has been used in driver's navigation and driving detection systems. It is important to note that the effectiveness of such systems can significantly be improved through the fusion of different kinds of sensors and data.

The objective of this system is to apply low-cost distributed intelligent sensors, which can easily be mounted on to a commercial steering wheel for analyzing real-time grip force measurement using MCU (Microcontroller Unit) by a simple digital interface.

As a consequence of this, the grip force data is integrated into the information data of the microcontroller, this data can be used to set the threshold value to buzz the alarm system to alert the driver in real-time.

CHAPTER 2

HARDWARE AND DESIGN SPECIFICATION

2.1. Block Diagram

Systematic steps in the system are shown in the block diagram.

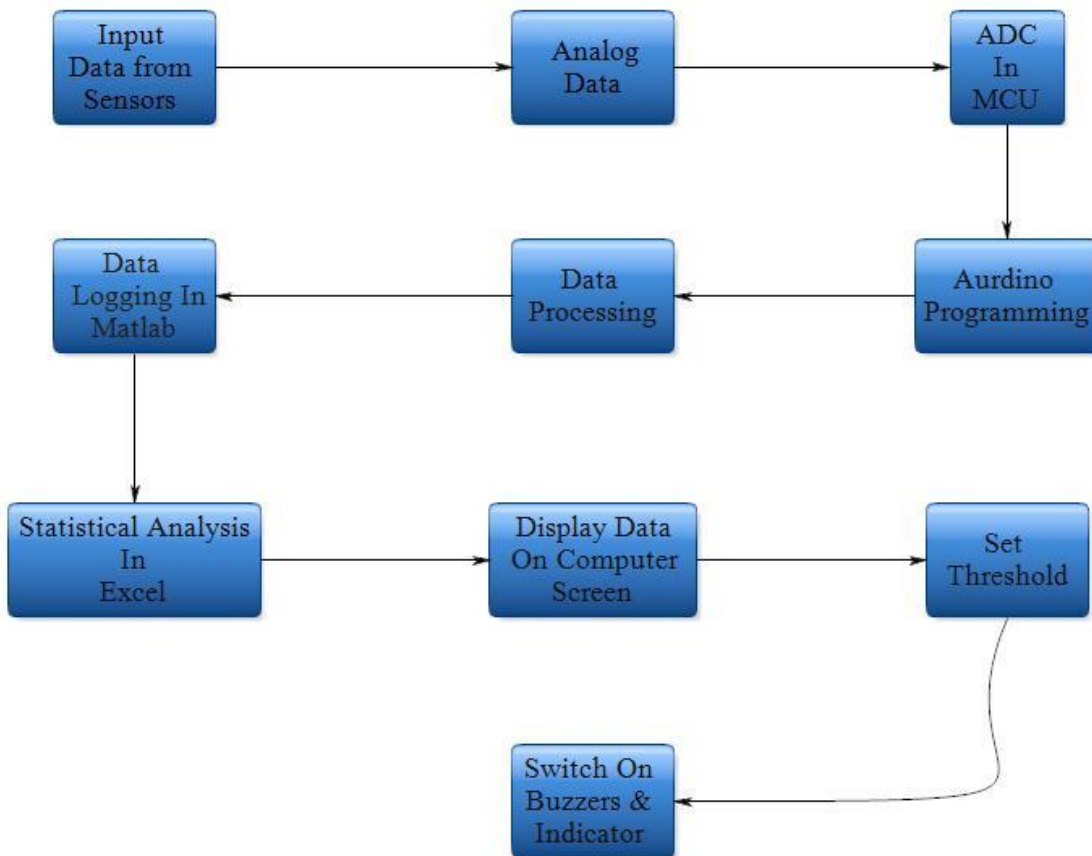


Fig 2.1.1: Block Diagram

2.2. Hardware Selection

To simulate the driver's grip force inside the laboratory we use the following hardware.

2.2.1. Gaming Console



Fig 2.2.1: Gaming Console ^[6]

This is 5-in-1 gaming wheel with brake and pedal can be connected to PC, PSX, PS, PS2 and PS3. The steering sensitivity and wheel angle are adjustable. It vibrates when your game is on, giving you an amazing and real experience. It supports Plug and Play and is compatible with windows 98/ME/NT4.0/2000/XP/VISTA/7 ^[6].

2.2.2. Sensor

Force Sensing Resistors ^[7] (FSR) is a polymer thick film (PTF) device which exhibits a decrease in resistance with an increase in the force applied to the active surface. Its force sensitivity is optimized for use in human touch control of electronic devices. FSRs are not a load cell or strain gauge, though they have similar properties.

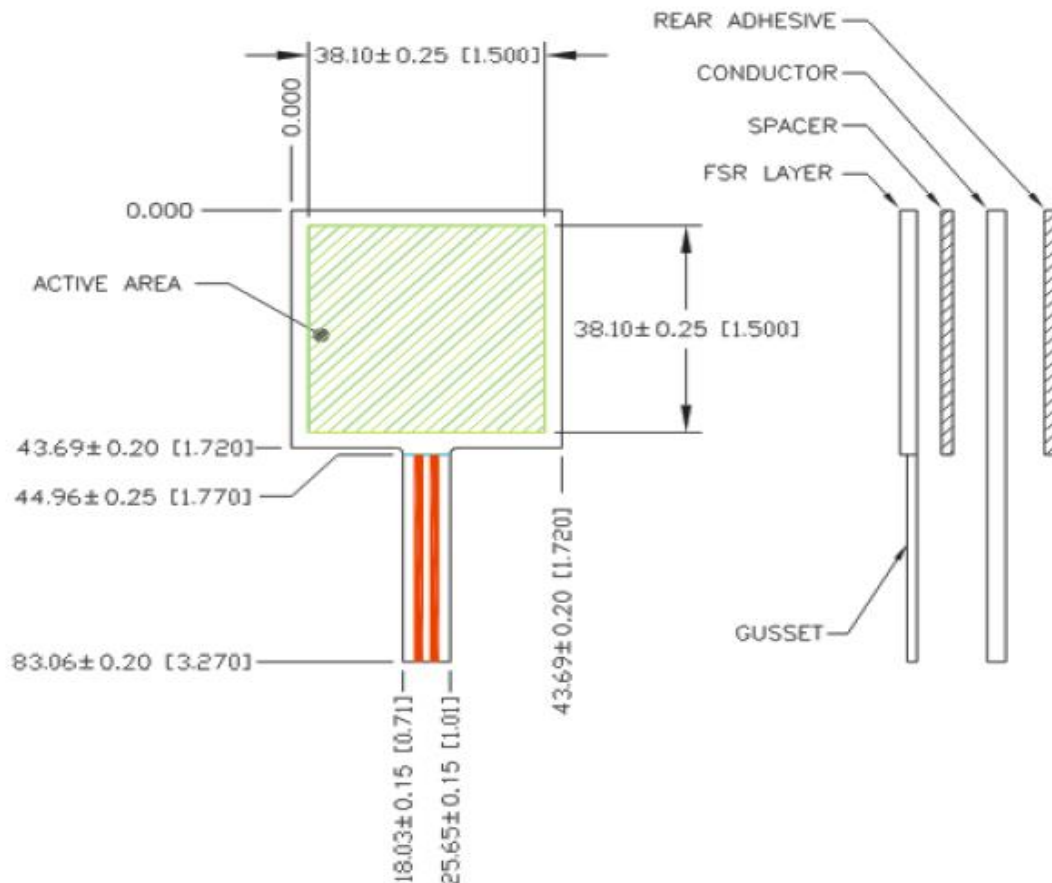


Fig 2.2.2: FSR-406 Sensor ^[7]

Characteristics of FSR ^[7]

- Force sensitivity range <1N to>10N
- Force Resolution better than 0.5% of full scale
- Standoff Resistance >10kΩ
- Temperature Range -30⁰C to +70⁰C
- Sensitivity to Noise/Vibration not significantly affected
- Lifetime > 10 million actuations

2.2.3. Arduino Due Board

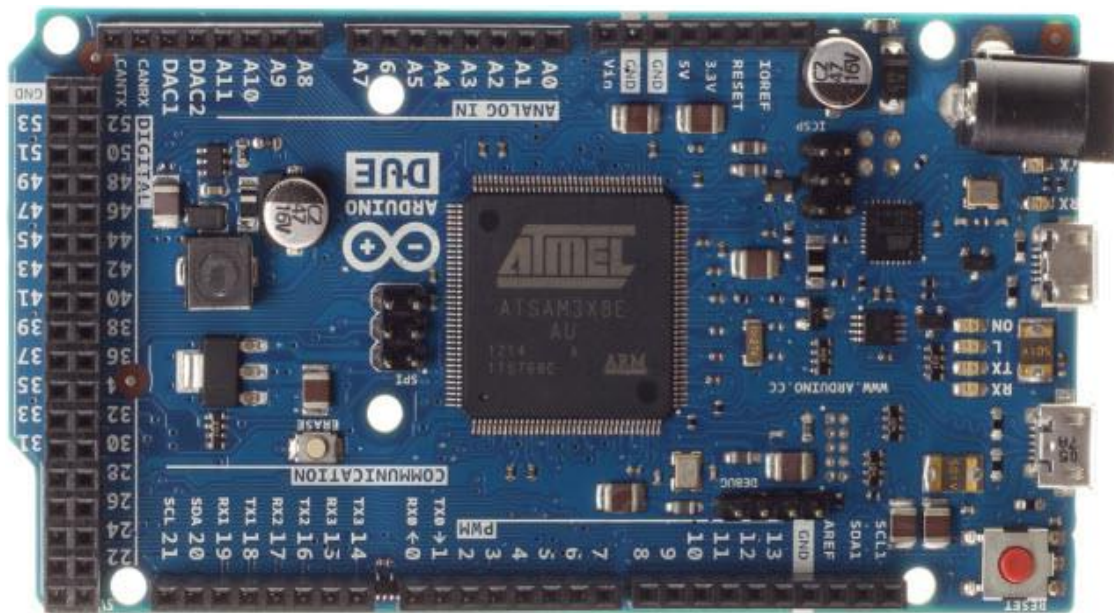


Fig 2.1.3: Arduino Due Board ^[8]

Summary:

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Outputs Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz

2.2.4. Buzzer^[9]

A buzzer is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.

Electromagnetic Buzzer (model: SB_PB5) - 5v (active buzzer / Piezo Buzzer) (operational range 2V - 5 V) is used to indicate the driver whether it grip loosened or not in real-time.

2.2.5. LED

Light-emitting diode (LED) is a two-lead semiconductor device that resembles to a basic pn-junction diode, except that LED can also emit light. When a forward voltage is applied to the LED's anode lead, current flows through it and it emits light energy. This phenomenon occurs because they have large band gaps compared to conventional diode.

In LED electrons are able to recombine with holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the material used in making the LED and energy band gap of the semiconductor.

2.2.6. Connecters and Wires

A five pin male-female connector is used to connect the Arduino Due board pins to buzzers, LEDs and ground, and also a two pin connector is used to acquire signals from the FSR sensors. Wires are used to connect the input-output assembly of the setup.

2.3. Design Procedure

The complete system can work in the manner shown below:

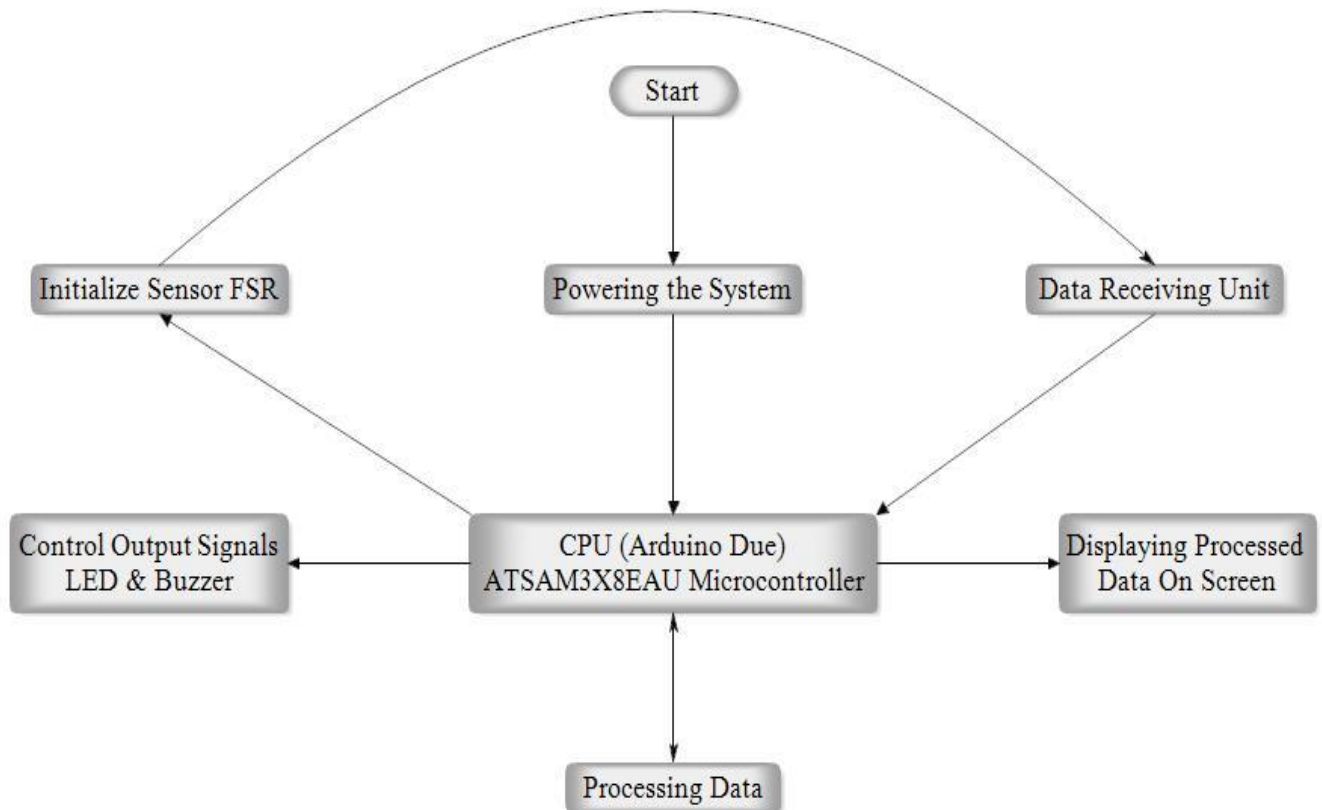


Fig 2.3.1: Design Procedure

2.4. Constraints

To make our system highly sensitive we need high processing power, so that at an instant it will give us the processing result.

The average power consumed is less than 350 mW.

After knowing our requirement we have an idea of what type of microcontroller we need to have. We should also know the application in which the microcontroller is going to be used. In our case it is Grip Force Measurement, so we need to check some of the primary requirements like:

Computational Performance: Generally microcontroller performance is measured in terms of MIPS. This is the only system which is going to perform all computational tasks so it should have high MIPS and higher operating frequency.

Power Consumption: We get low power consumption devices, so all subsystems should consume as much as less power as possible.

To market we get many microcontrollers of similar specifications but different features and resolutions. We select a chip that has all the features with the resolution as per our needs.

Availability of components: Components used in this design needs to be easily and commercially available for a long time to come.

Memory: We should have enough memory in the microcontrollers to program it and to make any changes in the future.

Online support: Selecting a microcontroller which has good online support will help us with your ideas and solve most of your problems as the experience of other users is available for you.

History: We have to check whether anyone has already used these types of hardware in the space and what are their ratings.

After considering all these we have chosen ARM Cortex-M3 CPU based Atmel AT91SAM3X8E microcontroller based evaluation board i.e. Arduino Due.

2.5. Arduino Due features^[8]

Overview:

The Arduino Due is based on the Atmel SAM3X8E ARM Cortex-M3 microcontroller. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.

Warning:

Unlike other Arduino boards, the Arduino Due board runs at 3.3V. The maximum voltage that the I/O pins can tolerate is 3.3V.

The board contains everything needed to support the microcontroller; simply connect it to a computer with a micro-USB cable or power it with an AC-to-DC adapter or battery to get good performance.

ARM Core benefits:

The Due has a 32-bit ARM core that can outperform typical 8-bit microcontroller boards. The most significant differences are:

1. A 32-bit core, that allows operations on 4 bytes wide data within a single CPU clock
2. CPU Clock at 84 MHz
3. 96 K Bytes of SRAM
4. 512 K Bytes of Flash memory for code
5. A DMA controller, that can relieve the CPU from doing memory intensive tasks

Power:

The Arduino Due can be powered via computer's USB connector or with an external DC power supply. The external (non-USB) power can come either from an AC-to-DC adapter or battery. The adapter jack can be connected by plugging a 2.1mm center-positive plug into the Due board's power jack. The leads of a battery can be inserted in the Ground and V_{in} pin headers of the power connector.

The board can operate on an external DC supply of 6 to 20V. If we supply it with a less than 7V, the 5V pin may not supply five volts and the board may be unstable. If we are using more than 12V, the voltage regulator may be overheat and damage the board. The recommended range is from 7 to 12V.

The power pins ^[8] are as follows:

V_{in} : The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). We can supply voltage through this pin, or if supplying voltage via the power jack, access it through this pin.

5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

3.3V: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 800mA. This regulator can also provide the power supply to the SAM3X microcontroller.

GND: Ground pins.

IOREF: This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Memory:

The SAM3X has 512 KB (2 blocks of 256 KB) of flash memory for storing Arduino code. The boot loader is pre-burned in a factory from Atmel and is stored in a dedicated ROM memory. The available SRAM is 96 KB in two contiguous banks of 64 KB and 32 KB. All the available memory (Flash, RAM and ROM) can be accessed directly as a flat addressing space.

It is possible to erase the Flash memory of the SAM3X with the on-board erase button. This will remove the currently loaded sketch from the MCU. To erase, press and hold the erase button for a few seconds while the board is powered.

Input and Output:

Digital I/O: Pins from 0 to 53

Each of the 54 digital pins on the Due can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 3.3 volts. Each pin can provide (source) a current of 3 mA or 15 mA, depending on the pin, or receive (sink) a current of 6 mA or 9 mA, depending on the pin. They also have an internal pull-up resistor (disconnected by default) of 100 kΩ.

In addition, some pins have specialized functions:

Serial 0: 0 (RX) and 1 (TX)

Serial 1: 19 (RX) and 18 (TX)

Serial 2: 17 (RX) and 16 (TX)

Serial 3: 15 (RX) and 14 (TX)

Used to receive (RX) and transmit (TX) TTL serial data (with 3.3 V level). Pins 0 and 1 are connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

PWM: Pins 2 to 13

Provide 8-bit PWM output with the analogWrite() function. The resolution of the PWM can be changed with the analogWriteResolution() function.

SPI: SPI header

These pins support SPI communication using the SPI library. The SPI pins are broken out on the central 6-pin header, which is physically compatible with the Due, Uno, Leonardo and Mega2560. The SPI header can be used only to communicate with other SPI devices, not for programming the SAM3X with the In-Circuit-Serial-Programming technique.

CAN: CANRX and CANTX

These pins support the CAN communication protocol but are not yet supported by Arduino APIs.

LED: Pin 13

There is a built-in LED connected to digital pin 13. When the pin is HIGH, the LED is on, when the pin is LOW, it's off. It is also possible to dim the LED because the digital pin 13 is also a PWM output.

TWI 1: 20 (SDA) and 21 (SCL)**TWI 2: SDA1 and SCL1.**

Support TWI communication using the Wire library.

Analog Inputs: Pins from A0 to A11

Arduino Due has 12 analog inputs, each of which can provide 12 bits of resolution (i.e. 4096 different values). By default, the resolution of the readings is set at 10 bits, for compatibility with other Arduino boards.

It is possible to change the resolution of the ADC with [analogReadResolution\(\)](#). The Due's analog inputs pins measure from ground to a maximum value of 3.3V. Applying more than 3.3V on the Due's pins will damage the SAM3X chip. The `analogReference()` function is ignored on the Due.

The AREF pin is connected to the SAM3X analog reference pin through a resistor bridge. To use the AREF pin, resistor BR1 must be disordered from the PCB.

DAC:

These pins provide true analog outputs with 12-bits resolution (4096 levels) with the `analogWrite()` function. These pins can be used to create an audio output using the Audio library.

Other pins on the board:

AREF:

Reference voltage for the analog inputs. Used with `analogReference()`.

Reset:

Bring this line LOW to reset the microcontroller.

Communication ^[3]

The Arduino Due has a number of facilities for communicating with a computer, other microcontrollers, and different devices like smart phones, tablets, cameras and so on. The SAM3X provides one hardware UART and three hardware USARTs for TTL (3.3V) serial communication.

The Programming port is connected to an ATmega16U2, which provides a virtual COM port to software on a connected computer (To recognize the device, Windows machines will need a .inf file, but OSX and Linux machines will automatically recognize the board as a COM port.). The ATmega16U2 is also connected to the SAM3X hardware UART. Serial on pins RX0 and TX0 provides Serial-to-USB communication for programming the board through the ATmega16U2 microcontroller.

The Arduino Due software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX (Receiving) and TX (Transmitting) LEDs on the board will flash when data is being transmitted via the ATmega16U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

CHAPTER 3

MEASUREMENT TECHNIQUES

3.1. Hardware Connections

The following circuit is built according to the given schematic.

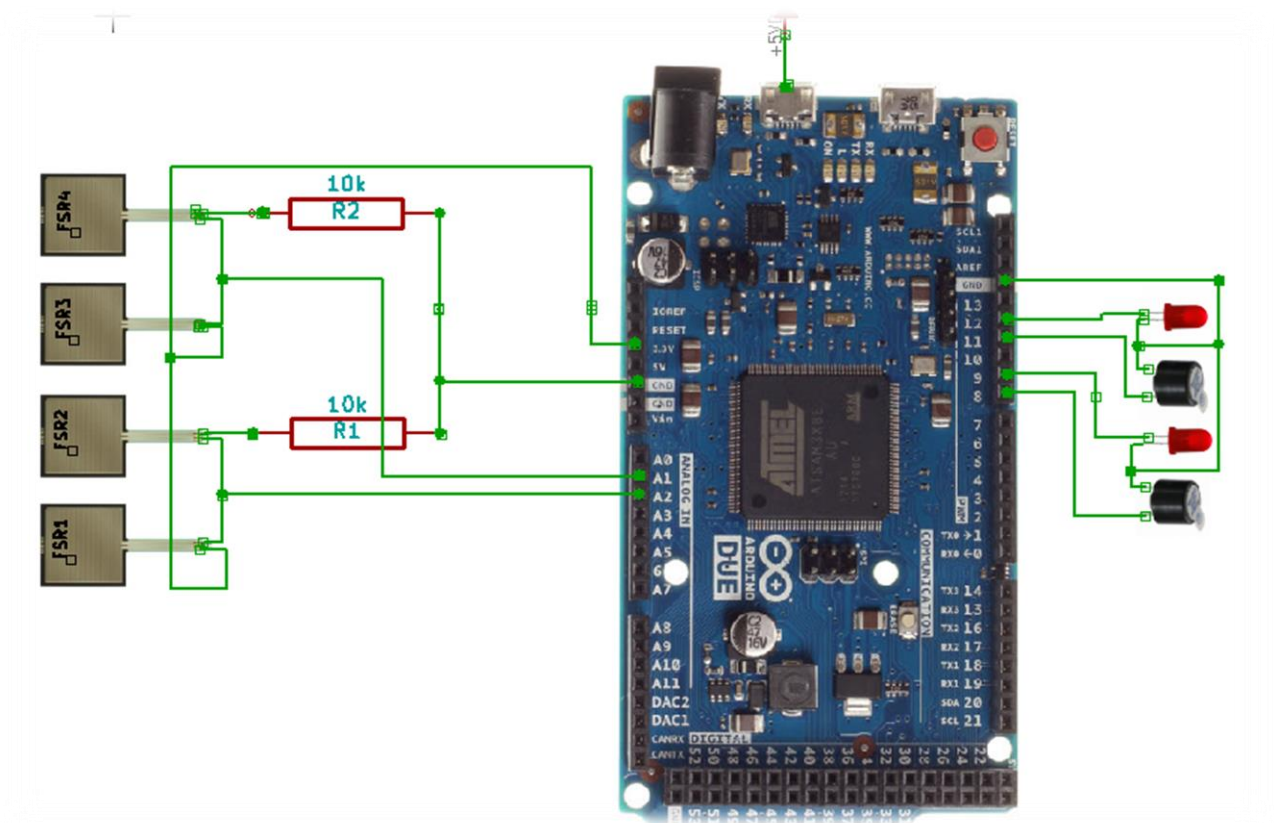


Fig 3.1.1: Hardware Connections ^[10]

3.2. Sensor Calibration

In human interface applications where fingers or Palm are the mode of sensing or actuation, good control of these parameters is not generally possible. Since, human force sensing is somewhat inaccurate; it is rarely sensitive to detect differences of less than $\pm 70\%$.

For dynamic measurements of FSR or Variable Control, a current to voltage converter is required; this produces an output voltage inversely proportional to FSR resistance; however the FSR resistance is approximately inversely proportional to the applied force.

The end result is a direct proportionality between the force and voltage; in another way, this circuit roughly gives a linear relationship between output voltage and applied force. This linearization of the response optimizes the resolution and simplifies data interpretation.

Three methods can be utilized: Gain and Offset Trimming, Curve Fitting and Analog Voltage Reading Method

1. Gain and offset trimming can be used as an easy method for calibration. The reference voltage and feedback resistor of the current to voltage converter are adjusted for each FSR to pull their responses closer to the nominal curve.
2. Curve fitting is the most appropriate calibration method. The parametric curve fit is done for the nominal curve from a set of FSR devices, and the resultant equation is saved for further use. Fit parameters are then established for each individual FSR (or sending element in an array) in the set.

These parameters, along with measured sensor resistance (or voltage), are put into the equation to obtain the force reading. If required, temperature compensation can also be included in that equation.

3. Analog Voltage reading method is the simplest way to measure the resistive sensor whose one end is connected to the power and other to a pull-down resistor and ground.

The midpoint between the fixed pull down resistor and the variable FSR resistor is connected to the analog input of a microcontroller such as an Arduino Due shown below.

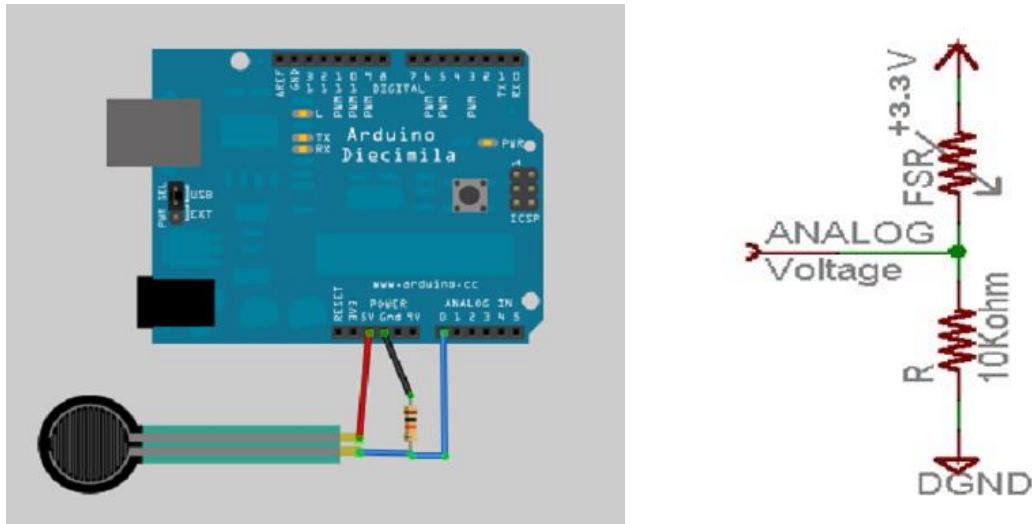


Fig 3.2.1: Sensor Calibration ^[11]

The way this circuit works is that as the resistance of the FSR decremented, the total resistance of the FSR and the pull-down resistor also decremented from about 10kΩ. It means that the current flowing through both resistors enhances which in turn causes the voltage across the fixed 10 kΩ resistor to increase.

This method takes somehow linear resistivity but does not provide linear voltage, which is because the voltage equation ^[11] for this is.

$$V_O = V_{CC} \times \frac{R}{R + FSR}$$

It means the voltage is proportional to the inverse of FSR resistance.

3.3. Programming

This system is computer controlled, so in order to correct working of this system, we program its 32-bit AT91SAM3X8E microcontroller in C Language in Arduino Compiler. We also introduce some Matlab command in this program, to be able to take the data into Matlab Environment for analysis purposes. The Following Block Codes ^[12] is written below.

- Initialize Variables
- Select Input and Output Pins
- Initialize the Function Loop
- Start Reading the Analog Data
- Convert Analog to Digital Data
- Processed the Given Data
- Setting the Threshold Values
- Applying Various Infinite Loops
- Inserting Matlab Commands
- Generates Delays

CHAPTER 4

TESTING AND RECEIVING DATA

4.1. Testing Hardware

After making necessary connections we are ready to test our hardware.

4.1.1. Testing an FSR

The simplest way to determine how FSR works ^[7] is to connect a multimeter in resistance measurement mode, connect multimeter probes to the tabs of sensor and see how the resistance changes. Because the resistance varies a lot, an auto-ranging meter works well here. Otherwise, you may change the tab on your own.

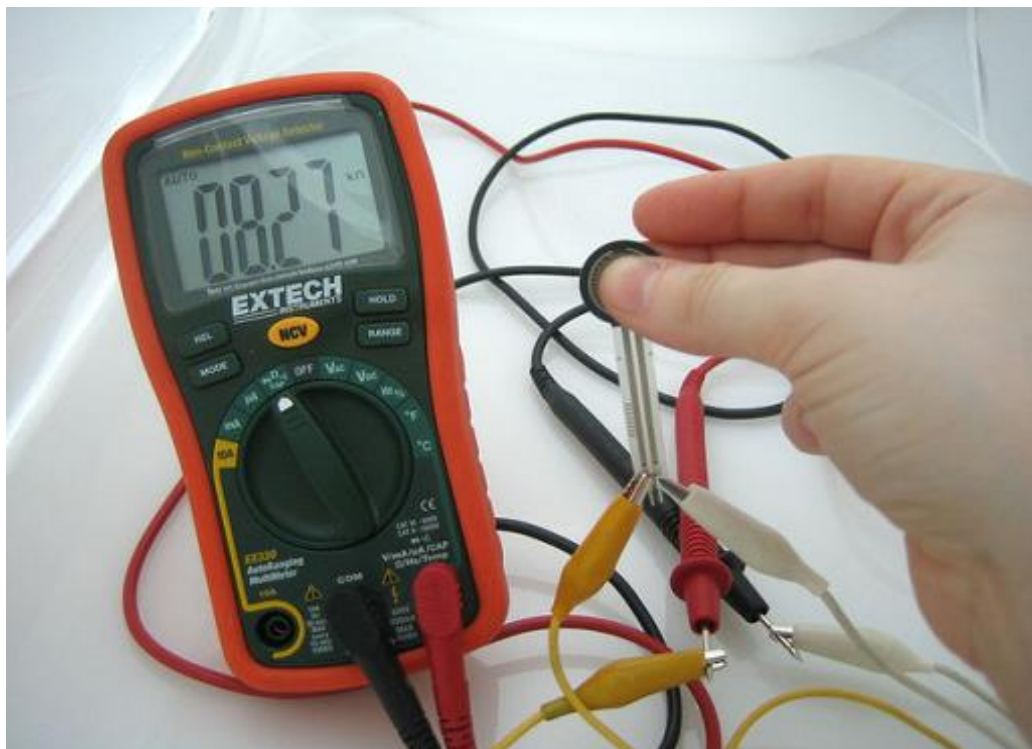


Fig 4.1.1: Testing of an FSR Sensor

4.1.2. Testing an Arduino Due Board

We make a small program of serial data transfer to blink an LED and see that on the Arduino 1.5.3 software serial monitor.

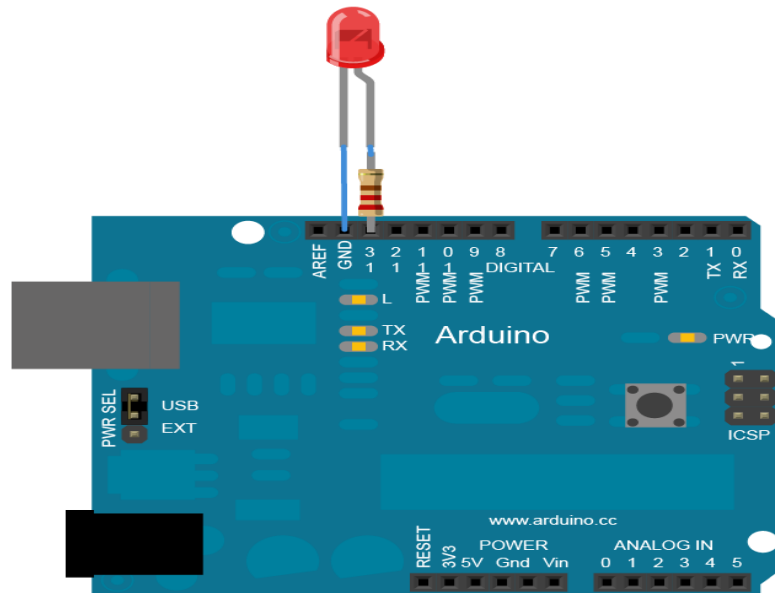


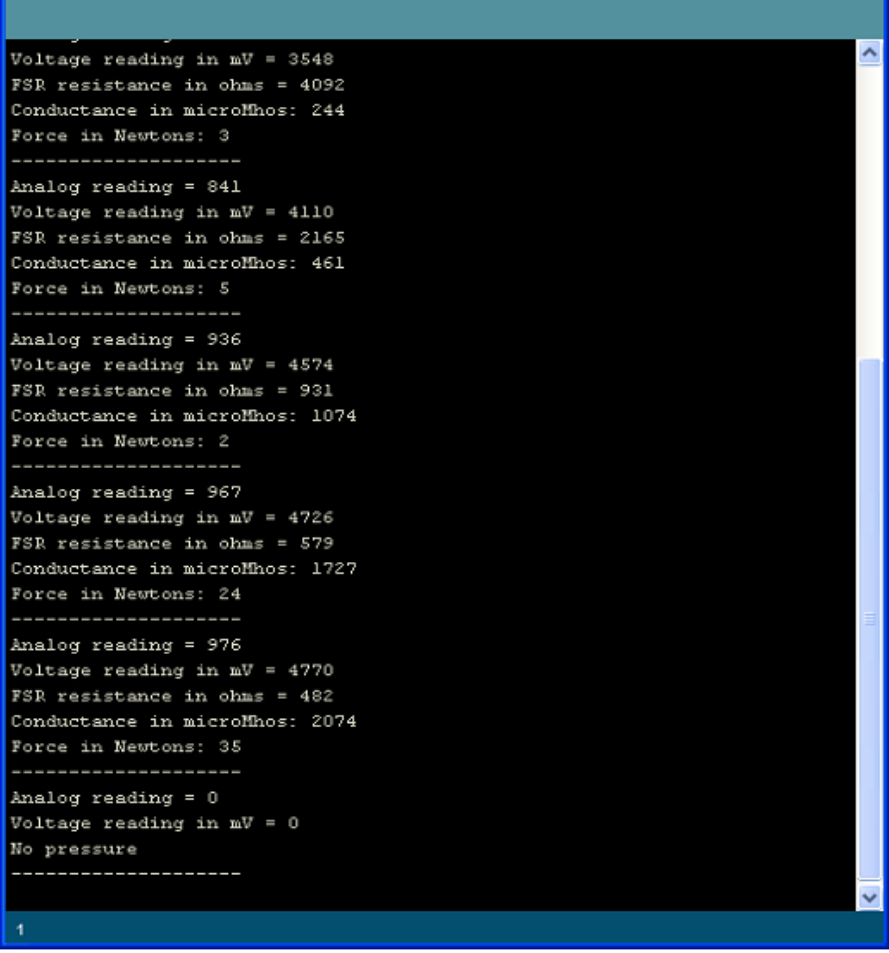
Fig 4.1.2: Testing of an Arduino Due Board ^[1]

To build the circuit, attach a 220Ω resistor to pin 13 of the Arduino board, then attach long leg of an LED (Positive leg, called the anode) to the 220Ω resistor. Attach a short leg (Negative leg, called the cathode) to ground. Then we plug the Arduino board into our computer, open the program window, and enter the code sketch in Appendix-D.

4.1.3. Receiving Grip Force Signals

After attaching FSR sensor to the steering wheel and making complete hardware connections as shown in figure 3.1.1. Attach the USB micro side of the USB cable to the Due's *Programming* port (this is the port closer to the DC power connector). To upload a sketch, then open the serial monitor on the software environment.

Start gripping the FSR sensors to see if readings are shown or not, if everything goes well then you will see some readings on the serial monitor according to your program ^[13].



```
Voltage reading in mV = 3548
FSR resistance in ohms = 4092
Conductance in microMhos: 244
Force in Newtons: 3
-----
Analog reading = 841
Voltage reading in mV = 4110
FSR resistance in ohms = 2165
Conductance in microMhos: 461
Force in Newtons: 5
-----
Analog reading = 936
Voltage reading in mV = 4574
FSR resistance in ohms = 931
Conductance in microMhos: 1074
Force in Newtons: 2
-----
Analog reading = 967
Voltage reading in mV = 4726
FSR resistance in ohms = 579
Conductance in microMhos: 1727
Force in Newtons: 24
-----
Analog reading = 976
Voltage reading in mV = 4770
FSR resistance in ohms = 482
Conductance in microMhos: 2074
Force in Newtons: 35
-----
Analog reading = 0
Voltage reading in mV = 0
No pressure
-----
1
```

Fig 4.1.3: Receiving Grip force Signals

4.2. Matlab Graphical Analysis ^[14]

4.2.1. Test Driver 1

Left Hand Plot

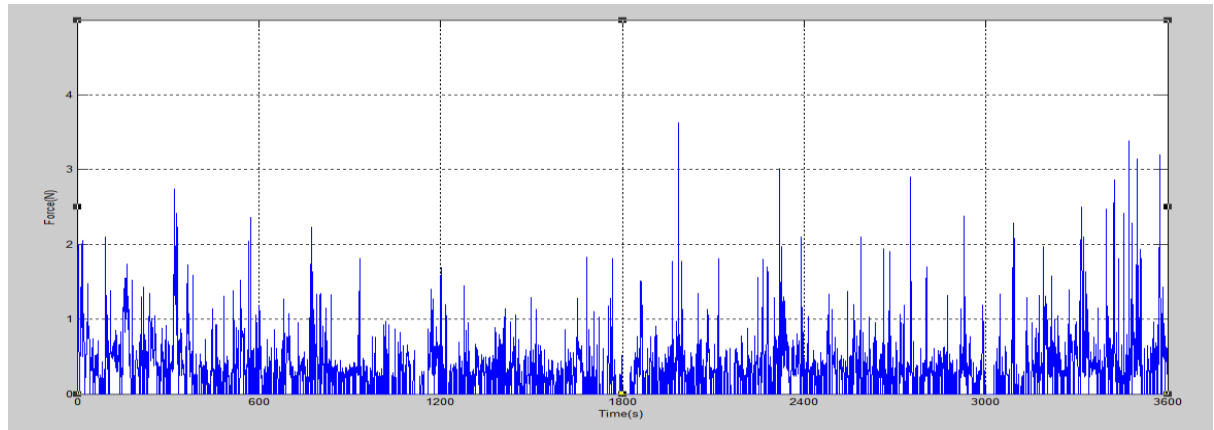


Fig 4.2.1.1: Test Driver 1 Left Hand Plot

Right Hand Plot

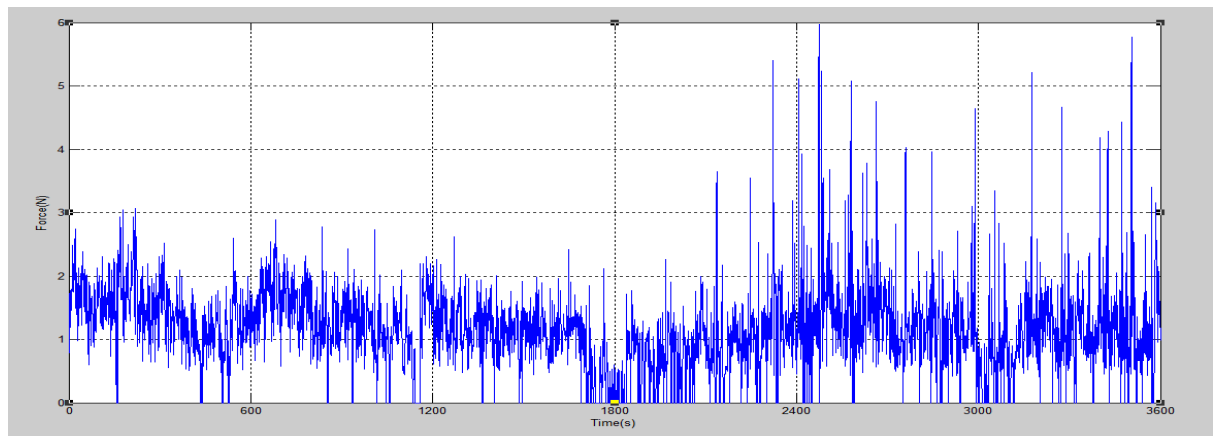


Fig 4.2.1.2: Test Driver 1 Right Hand Plot

Combined Plot

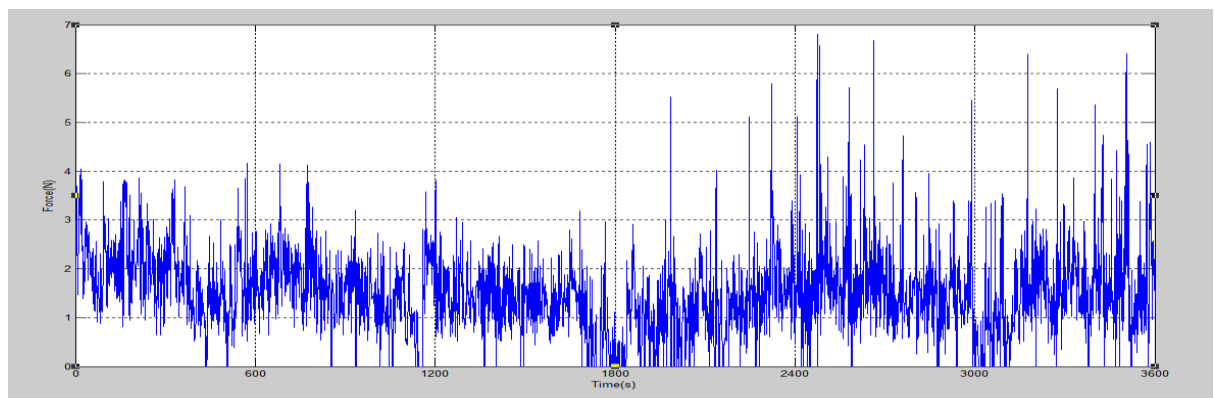


Fig 4.2.1.3: Test Driver 1 Combined Plot

Zoomed version of Left Hand Plot

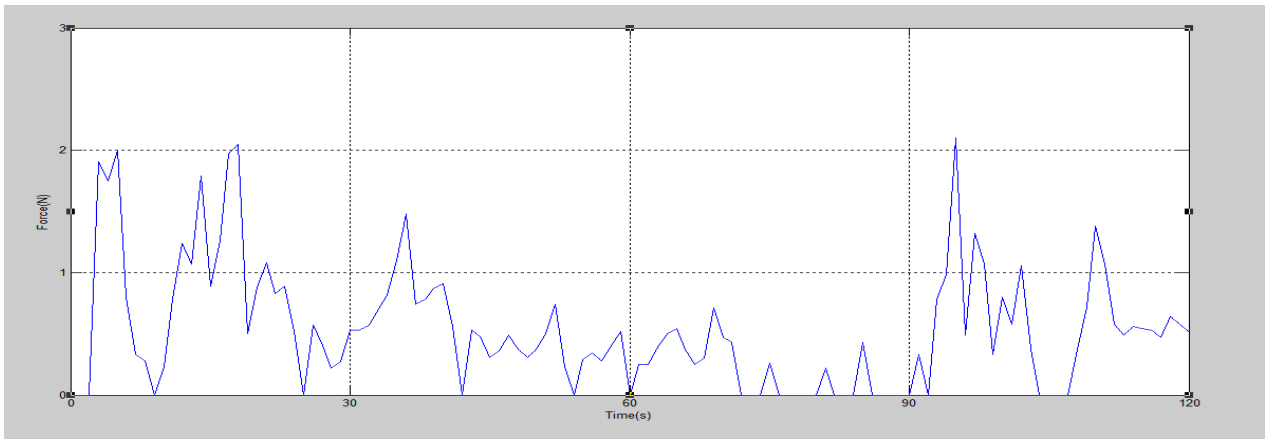


Fig 4.2.1.4: Zoomed version of Left Hand Plot

Zoomed version of Right Plot

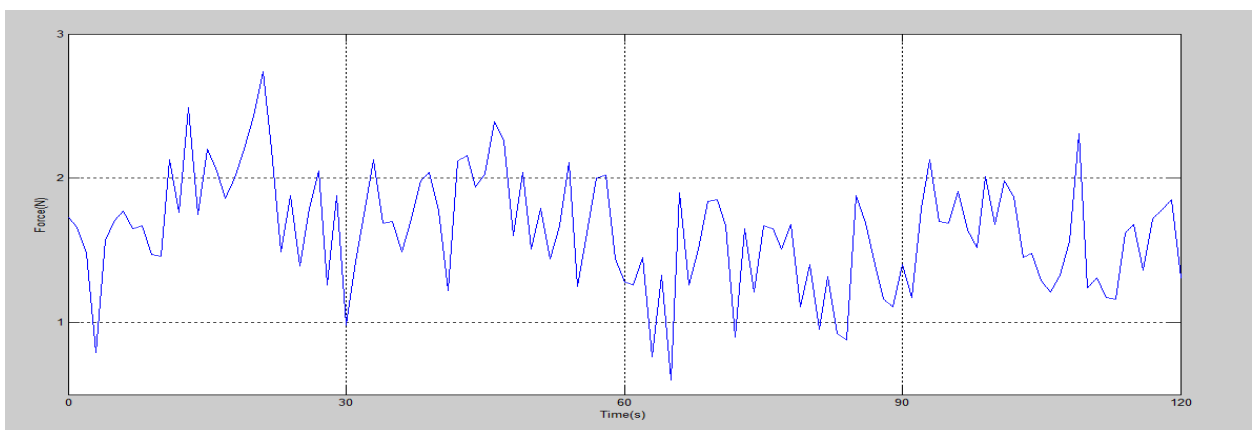


Fig 4.2.1.5: Zoomed version of Right Hand Plot

Zoomed version of Combined Plot

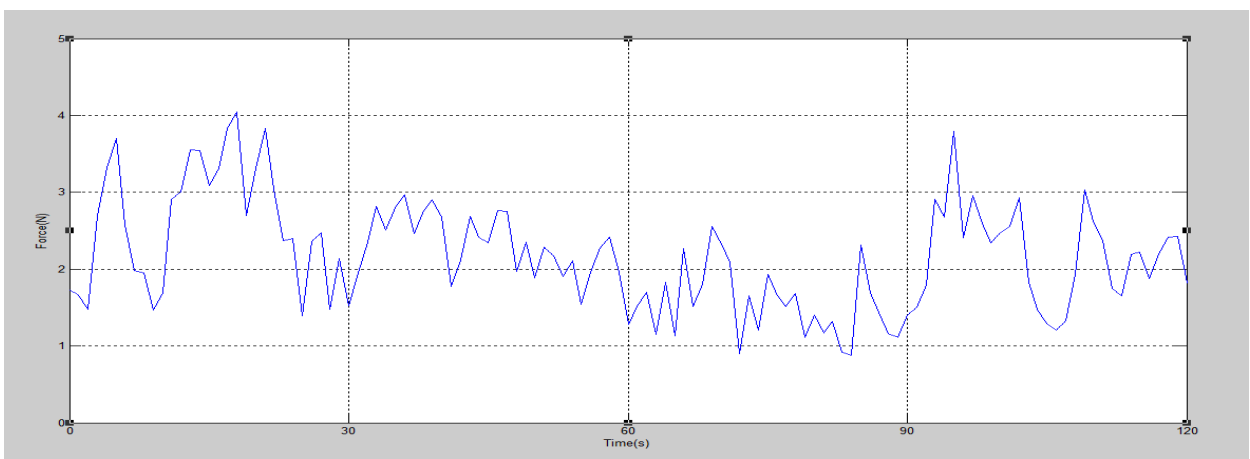


Fig 4.2.1.6: Test Driver 1 Zoomed Combined Plot

4.2.2. Test Driver 2

Left Hand Plot

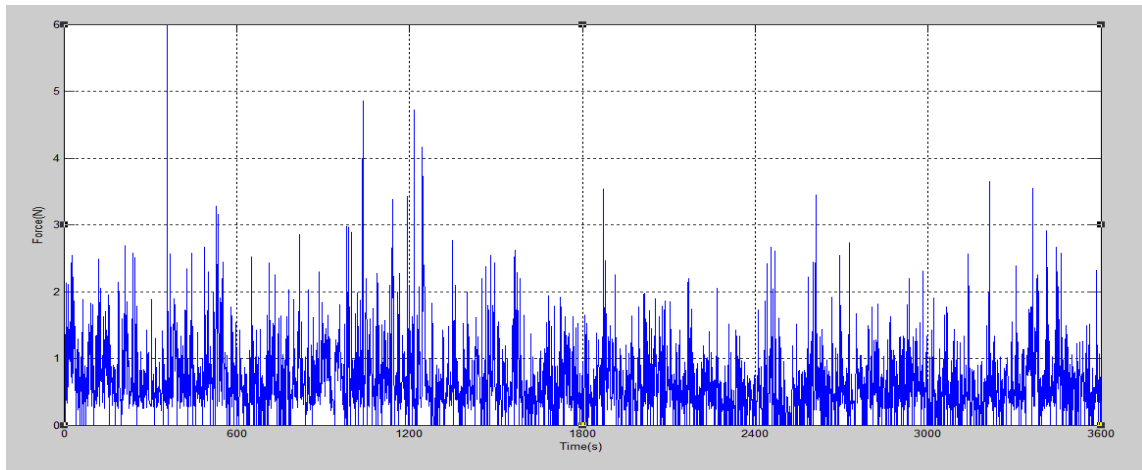


Fig 4.2.2.1: Test Driver 2 Left Hand Plot

Right Hand Plot

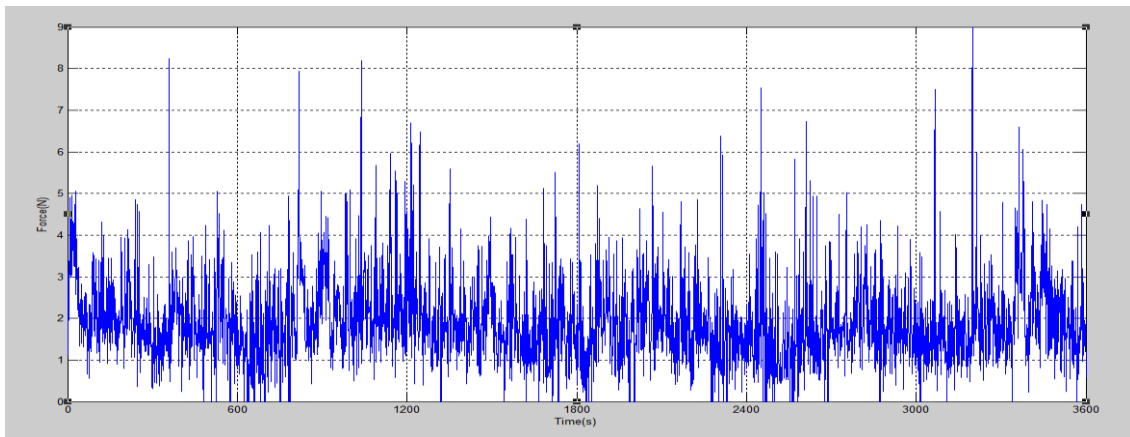


Fig 4.2.2.2: Test Driver 2 Right Hand Plot

Combined Plot

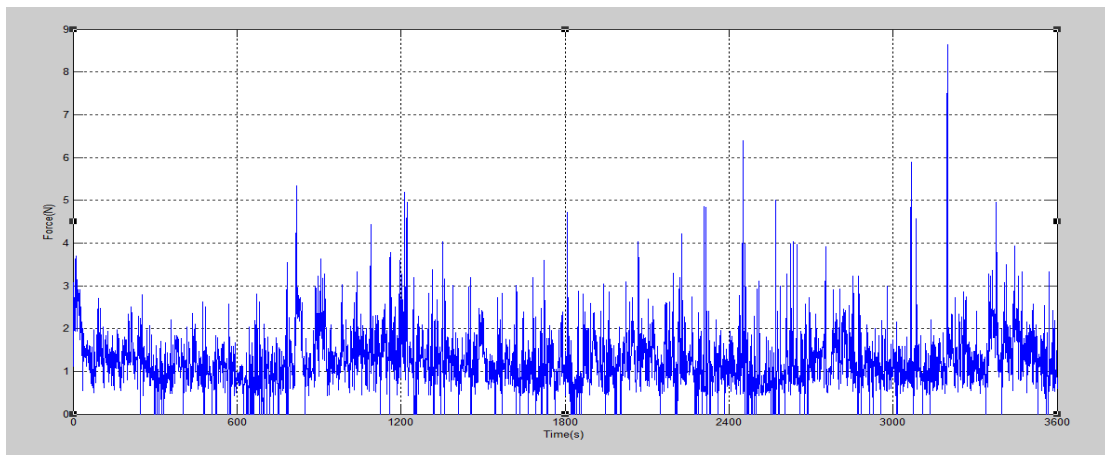


Fig 4.2.2.3: Test Driver 2 Combined Plot

4.3. Anova Test in Excel ^[15]

We have huge amount of data to analyze, we have to maintain this data in Excel sheets. If you want to see this data please click on the link below.

[Raw Test Drivers Readings.xlsx](#)

The reason for doing an ANOVA is to see if there is any difference between groups on some variable. The equation of sample variance ^[17] is

$$s^2 = \frac{1}{n-1} \sum (y_i - \bar{y})^2$$

where, the divisor is called the degrees of freedom (DF), the summation is called the sum of squares (SS), the result is called the mean square (MS) and the squared terms are deviations from the sample mean. The critical value of F is a function of the numerator degrees of freedom. If $F \geq F_{\text{Critical}}$, then reject the null hypothesis ^[17]. The computer method calculates the probability (p-value) of a value of F greater than or equal to the observed value. The null hypothesis is rejected if the p-value is less than or equal to the significance level.

4.3.1. Test Driver 1

Anova: Single Factor

Source of Variation	SS	DF	MS	F	P-value	F crit
Between Groups	8.74E+09	3	2.91E+09	10781.9	0	1.547405
Within Groups	3.89E+09	14400	270225.4			
Total	1.26E+10	14403				

Table 4.3.1: Test Driver 1 Single Factor Anova

Anova: Two-Factor with Replication

Source of Variation	SS	DF	MS	F	P-value	F crit
Sample	450.8744	9	50.09715	121.7813	5.3E-219	1.360569
Columns	2647.601	2	1323.8	3218.029	0	1.609678
Interaction	118.9794	18	6.609965	16.06818	4.32E-50	1.264816
Within	4430.454	10770	0.41137			
Total	7647.909	10799				

Table 4.3.2: Test Driver 1 Two-Factor with Replication Anova

4.3.2. Test Driver 2

Anova: Single Factor

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	8.74E+09	3	2.91E+09	10778.81	0	1.547405
Within Groups	3.89E+09	14400	270225.5			
Total	1.26E+10	14403				

Table 4.3.3: Test Driver 2 Single Factor Anova

Anova: Two-Factor with Replication

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Sample	235.5883	9	26.17648	43.55808	1.98E-77	1.360569
Columns	2808.849	2	1404.424	2336.985	0	1.609678
Interaction	57.59408	18	3.199671	5.324305	1.48E-12	1.264816
Within	6472.293	10770	0.600956			
Total	9574.324	10799				

Table 4.3.4: Test Driver 2 Two-Factor with Replication Anova

4.3.3. Test Driver 3

Anova: Single Factor

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	8.74E+09	3	2.91E+09	10775.28	0	1.547405
Within Groups	3.89E+09	14400	270226.3			
Total	1.26E+10	14403				

Table 4.3.5: Test Driver 3 Single Factor Anova

Anova: Two-Factor with Replication

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Sample	1285.361	9	142.8179	92.39892	1.1E-166	1.360569
Columns	3959.74	2	1979.87	1280.917	0	1.609678
Interaction	292.2508	18	16.23616	10.50431	3.01E-30	1.264816
Within	16646.82	10770	1.545666			
Total	22184.18	10799				

Table 4.3.6: Test Driver 3 Two-Factor with Replication Anova

4.3.4. Test Driver 4

Anova: Single Factor

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	8.74E+09	3	2.91E+09	10787.17	0	1.547405
Within Groups	3.89E+09	14400	270225.4			
Total	1.26E+10	14403				

Table 4.3.7: Test Driver 4 Single Factor Anova

Anova: Two-Factor with Replication

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Sample	855.9851	9	95.10945	297.3024	0	1.360569
Columns	607.0637	2	303.5318	948.8095	0	1.609678
Interaction	162.3861	18	9.021451	28.20013	6.04E-94	1.264816
Within	3445.41	10770	0.319908			
Total	5070.845	10799				

Table 4.3.8: Test Driver 4 Two-Factor with Replication Anova

4.3.5. Test Driver 5

Anova: Single Factor

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	8.74E+09	3	2.91E+09	10787.12	0	1.547405
Within Groups	3.89E+09	14400	270225.4			
Total	1.26E+10	14403				

Table 4.3.9: Test Driver 5 Single Factor Anova

Anova: Two-Factor with Replication

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Sample	366.088	9	40.67644	102.3818	1.4E-184	1.360569
Columns	624.8742	2	312.4371	786.3977	0	1.609678
Interaction	73.03815	18	4.057675	10.21308	3.18E-29	1.264816
Within	4278.939	10770	0.397302			
Total	5342.939	10799				

Table 4.3.10: Test Driver 5 Two-Factor with Replication Anova

4.3.6. Test Driver 6

Anova: Single Factor

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	8.74E+09	3	2.91E+09	10784.54	0	1.547405
Within Groups	3.89E+09	14396	270075.2			
Total	1.26E+10	14399				

Table 4.3.11: Test Driver 6 Single Factor Anova

Anova: Two-Factor with Replication

<i>Source of Variation</i>	<i>SS</i>	<i>DF</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Sample	292.1832	9	32.4648	111.6982	3.6E-201	1.360569
Columns	610.0887	2	305.0444	1049.534	0	1.609678
Interaction	54.73209	18	3.040672	10.46172	4.25E-30	1.264816
Within	3130.272	10770	0.290647			
Total	4087.276	10799				

Table 4.3.12: Test Driver 6 Two-Factor with Replication Anova

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Conclusion

Repeatability of results is confirmed by performing six successive measurements. By performing ANOVA (Analysis of Variance) Test in Microsoft Excel and also by doing a graphical analysis in Matlab we conclude that the average force of left hand is 0.4 N because most of the times we use our left hand for changing the gear. Right hand average force is around 1 N and combined average force of both the hands is around 1.7 N.

A threshold value is to be set by using this analysis, i.e. for left hand threshold value is 0.22 N and 0.51 N for right hand. If the readings are below these thresholds, then microcontroller buzzes the alarm to alert the driver in real time. For the left hand, we also provide five second delay to re-grip the sensor after changing the gear, if it is not so, microcontroller buzz the alarm to alert the driver.

More tests will be conducted on the system with the advice of medical personals in order to locate the accurate change in steering grip force in order to minimize false alarm. However, fatigue is a complex and safety-related matters and its detection should not be based on steering grip force monitoring alone ^[16].

Consequently, the driver fatigue detection system needs to be merged into a sensor decision-making system that integrates the outputs from other fatigue detection system currently being developed.

Perhaps the percentage of road fatalities will fall tremendously and thus making our roads a safe passage to all our destinations when this system will be available in the market. The system may be a reminder to all drivers that they should be in the best of alertness ^[5] when they are behind the wheels or otherwise the system will automatically remind them.

5.2. Future Scope

1. To measure the grip force that lies in the geometry of sensor, that are not easily available according to our requirement, but the sensor manufacturing company will make it if we give them bulk order. Because for measuring force we have to symmetrically place the sensors on the steering wheel that are different for different vehicles.
2. Since the upper coating of our sensor is delicate and need special precaution. We put paper tape on it that causes an error in readings. So, we calibrate our program to subtract this force from applying force. Sensor manufacturing companies will place a good protective covering on the sensors to work under rugged conditions.
3. In the future by using appropriate algorithms, we can know that how much time the driver left the steering wheel and using this data; the vehicle's intelligent system can control the vehicle in auto driver mode to avoid fatal accidents. That system also alerts the driver and nearby vehicles by giving special light and sound signal, that are approved by the Government for sleeping drivers.

APPENDIX-A

Op-Code Used in Arduino Programming

- `setup()`

The `setup()` function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

- `loop()`

After creating a `setup()` function, which initializes and sets the initial values, the `loop()` function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

- `if/else`

`if/else` allows greater control over the flow of code than the basic `if` statement, by allowing multiple tests to be grouped together. For example, an analog input could be tested and one action taken if the input was less than 500, and other action taken if the input was 500 or greater.

- `for`

`For` statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. `For` statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

- `while`

`while` loops will loop continuously, and infinitely, until the expression inside the parenthesis, `()` becomes false. Something must change the tested variable, or the `while` loop will never exit. This could be in your code, such as an incremented variable, or an external condition, such as testing a sensor.

- break

Break is used to exit from a do, for, or while loop, bypassing the normal loop condition. It is also used to exit from a switch statement.

- void

The void keyword is used only in function declarations. It indicates that the function is expected to return no information to the function from which it was called.

- int

Integers are your primary data-type for number storage. On the Arduino Due, an int stores a 32-bit (4-byte) value. This yields a range of -2,147,483,648 to 2,147,483,647 (minimum value of -2^{31} and a maximum value of $(2^{31}) - 1$).

int's store negative numbers with a technique called 2's complement method. The highest bit, sometimes referred to as the "sign" bit, flags the number as a negative number. The rest of the bits are inverted and 1 is added.

- float

The data type for floating-point numbers, a number that has a decimal point. Floating-point numbers are often used to approximate analog and continuous values because they have greater resolution than integers. Floating-point numbers can be as large as 3.4028235E+38 and as low as -3.4028235E+38. They are stored as 32 bits (4 bytes) of information.

Floats have only 6-7 decimal digits of precision. That means the total number of digits, not the number to the right of the decimal point. Unlike other platforms, where you can get more precision by using a double (e.g. up to 15 digits), on the Arduino, double is the same size as a float.

- pinMode()

Configure the specified pin to behave either as an input or an output. See the description of digital pins for details on the functionality of the pins.

As of Arduino 1.0.1, it is possible to enable the internal pull-up resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pull-ups.

- digitalWrite()

Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pull-up on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.

- digitalRead()

Reads the value from a specified digital pin, either HIGH or LOW.

- analogRead()

Reads the value from the specified analog pin. The Arduino board contains a 6 channels (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

- analogWrite()

Writes an analog value (PWM wave) to a pin. Can be used to light an LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite() on the same pin).

- `delay()`

Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

- `Serial`

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): `Serial`. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to `begin()`.

- `end()`

Disables serial communication, allowing the RX and TX pins to be used for general input and output. To re-enable serial communication, call `Serial.begin()`.

APPENDIX-B

Syntax and Operators Used in Arduino Programming

Syntax:

- ; (semicolon)

Used to end a statement

- {} (curly braces)

Curly braces (also referred to as just "braces" or as "curly brackets") are a major part of the C programming language. They are used in several different constructs, outlined below, and this can sometimes be confusing for beginners.

An opening curly brace "{" must always be followed by a closing curly brace "}". This is a condition that is often referred to as the braces being balanced.

The Arduino IDE (integrated development environment) includes a convenient feature to check the balance of curly braces. Just select a brace, or even click the insertion point immediately following a brace, and its logical companion will be highlighted.

- // (single line comment) and /* */ (multi-line comment)

Comments are lines in the program that are used to inform yourself or others about the way the program works. They are ignored by the compiler, and not exported to the processor, so they don't take up any space on the Atmega chip.

Comments only purpose are to help you understand (or remember) how your program works or to inform others how your program works.

- `#define`

`#define` is a useful C component that allows the programmer to give a name to a constant value before the program is compiled. Defined constants in Arduino don't take up any program memory space on the chip. The compiler will replace references to these constants with the defined value at compile time.

- `#include`

`#include` is used to include outside libraries in your sketch. This gives the programmer access to a large group of standard C libraries (groups of pre-made functions), and also libraries written especially for Arduino.

Operators:

Arithmetic Operators

- `=` (assignment operator)
- `+` (addition)
- `-` (subtraction)
- `*` (multiplication)
- `/` (division)
- `%` (modulo)

Comparison Operators

- `==` (equal to)
- `!=` (not equal to)
- `<` (less than)
- `>` (greater than)
- `<=` (less than or equal to)
- `>=` (greater than or equal to)

Boolean Operators

- `and` (and)
- `||` (or)
- `!` (not)

Bitwise Operators

- `and` (bitwise and)
- `|` (bitwise or)
- `^` (bitwise xor)
- `~` (bitwise not)
- `<<` (bitshift left)
- `>>` (bitshift right)

Compound Operators

- `++` (increment)
- `--` (decrement)
- `+=` (compound addition)
- `-=` (compound subtraction)
- `*=` (compound multiplication)
- `/=` (compound division)
- `and=` (compound bitwise and)
- `|=` (compound bitwise or)

APPENDIX-C

Experimental Setup

Overview of the complete system



Fig C.1: Experimental Setup

Through Preview of Microcontroller Unit

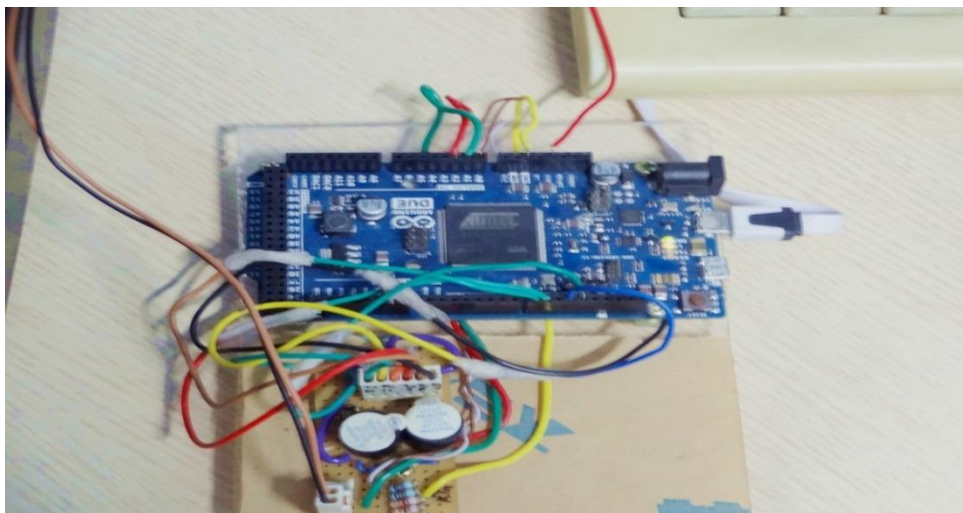


Fig C.2: Connections at Microcontroller Unit

APPENDIX-D

Testing an Arduino Due Board

/ Blink turns on an LED on for two and half second, then off for two and half second, repeatedly. */*

/ There is a built-in LED connected to digital pin 13. When the pin is at HIGH, the LED is on, when the pin is at LOW, it's off. We can also dim the LED because the digital pin13 is also a PWM output. */*

```
int led = 13;
```

```
// the setup routine runs infinitely:
```

```
void setup() {
```

```
Serial.begin(19200);    // Send debugging information through the Serial monitor
```

```
// Initialize the digital (PWM) pin as an output.
```

```
pinMode(led, OUTPUT);
```

```
} // End of function setup
```

```
// the loop routine runs infinitely forever:
```

```
void loop() {
```

```
digitalWrite(led, HIGH);    // turn on the LED (HIGH is the voltage level)
```

```
delay(2500);                // wait for two and half second
```

```
digitalWrite(led, LOW);     // turn off the LED by making the voltage LOW
```

```
delay(2500);                // wait for two and half second
```

```
} // End of Infinite loop
```

APPENDIX-E

Data Logging In Matlab

/ Matlab Program for data logging and graphical analysis */*

close all

clear all

t = 0:1:30;

data = zeros(length(t),10);

data_2 = zeros(1,6);

%% Serial Comm

s1 = serial ('COM10');

set (s1,'BaudRate',19200);

%set (s1,'Timeout', 0.01);

fopen(s1);

%% Main Program

for i= 1:length(t)

 idn = fgets(s1);

 if (~isempty(idn))

 C = textscan(idn, '%f32%f32%f32%f32%f32%f32%f32%f32', 'delimiter', ',');

 forceLeft = (C{1})

 conductanceLeft = (C{2})

 resistanceLeft = (C{3})

 forceRight = (C{4})

 conductanceRight = (C{5})

 resistanceRight = (C{6})

```

    forceCombined = (C{7})
    conductanceCombined = (C{8})
    resistanceCombined = (C{9})
    data(i,:) = [t(i) forceLeft conductanceLeft resistanceLeft forceRight conductanceRight
    resistanceRight forceCombined conductanceCombined resistanceCombined];

end

%   str = sprintf('%3.1f,%4.1,%3.1f\r',x);
%   fwrite(s1,str);          %% send data to arduino
%
%   xprev = fgets(s1); %% receive data from arduino
%   xdot= str2double(xprev);
%   x = x + xdot*Ts;

xlswrite('Data.xls',data);
deviation = std(data(:,2));
variance = var(data(:,2));
deviation2 = std(data(:,5));
variance2 = var(data(:,5));
deviation3 = std(data(:,8));
variance3 = var(data(:,8));

data_2(1,:) = [deviation variance deviation2 variance2 deviation3 variance3];
xlswrite('Std Var.xls',data_2);
figure,
plot(data(:,4),data(:,2));xlabel('Left Resistance(ohm)');ylabel('Left Force(N)'); grid on
figure,
plot(data(:,7),data(:,5));xlabel('Right Resistance(ohm)');ylabel('Right Force(N)'); grid on
figure,
plot(data(:,10),data(:,8));xlabel('Combined Resistance(ohm)');ylabel('Combined Force(N)');
grid on
figure,

```

```
plot(data(:,3),data(:,2));xlabel('Left Conductance(umho)');ylabel('Left Force(N)');grid on  
figure,
```

```
plot(data(:,6),data(:,5));xlabel('Right Conductance(umho)');ylabel('Right Force(N)');grid on  
figure,
```

```
plot(data(:,9),data(:,8));xlabel('Combined Conductance(umho)');ylabel('Combined  
Force(N)');grid on  
figure,
```

```
plot(t,data(:,2));xlabel('Time(s)');ylabel('Left Force(N)');grid on  
figure,
```

```
plot(t,data(:,5));xlabel('Time(s)');ylabel('Right Force(N)');grid on  
figure,
```

```
plot(t,data(:,8));xlabel('Time(s)');ylabel('Combined Force(N)');grid on  
fclose(s1);
```

REFERENCES

- [1] Q. Wang, J. Yang, et.al, “Driver fatigue detection: A survey,” in Proc. 6th World Congress on Intelligent Control and Automation (WCICA 2006), vol. 2, June 2006, pp. 8587–8591
- [2] Federico Baronti, et.al, 2009, Distributed Sensor for Steering Wheel Grip Force measurement in Driver Fatigue Detection, EDAA, 978-3-9810801-5-5
- [3] <http://www.morth.nic.in>
- [4] Y. Lin, et.al., 2007, An Intelligent Noninvasive Sensor for Driver Pulse Wave Measurement, IEEE Sensors Journal, Vol. 7, No. 5
- [5] Dr. Herlina Abdul Rahim, et.al, 2010, Grasp hand approach to detect the attentiveness of driver via vibration system, IEEE, 978-1-4244-7122-5/10
- [6] <http://www.zebronics.com>
- [7] <http://www.interlinkelectronics.com>
- [8] <http://arduino.cc>
- [9] <http://www.thefreedictionary.com/buzzer>
- [10] <http://www.kicad-pcb.org>
- [11] <https://learn.adafruit.com/force-sensitive-resistor-fsr>
- [12] <http://arduino.cc/en/Reference/HomePage>
- [13] <http://www.tekscan.com>
- [14] <http://www.mathworks.in>
- [15] <http://office.microsoft.com/en-us/excel/>
- [16] Tnum Chia Chieh, et.al. 2003, Driver Fatigue Detection using Steering Grip Force, IEEE, 0-7803-8173-4/03.
- [17] http://en.wikipedia.org/wiki/Analysis_of_variance

Index

A

Abstract, iii
Abbreviations, vii-ix
Acknowledgement, ii
Anova Test in Excel, 32-35
Arduino Due Board, 9
Arduino Due features, 13

B

Block Diagram, 6
Buzzer, 10

C

Conclusion, 36
Connecters and Wires, 10
Constraints, 12

D

Data Logging In Matlab, 26-28
Design Procedure, 11

F

Future Scope, 37

G

Gaming Console, 7

H

Hardware Selection, 7

I

Index, 47
Initiatives of NHAI for Road Safety Activities, 4

L

LED, 10
List of Figures, vii
List of Tables, vi
Literature Survey, 2-4

M

Matlab Graphical Analysis, 29-31

N

Need for Measurement System, 1

O

Objective and Scope of Work, 5

Op-Code Used in Arduino Programming, 38-41

P

Programming, 21

R

Receiving Grip Force Signals, 25
References, 46
Road Safety Activities, 3

S

Sensor, 8
Sensor Calibration, 19-20
Syntax and Operators Used in Arduino Programming, 42-44

T

Table of Contents, iv-v
Thesis Certificate, i
Testing Hardware, 22
Testing an FSR, 22
Testing an Arduino Due Board, 23-24