CRICKET BOWLING ANALYSER

ABSTRACT

This project is an alternative for the expensive performance evaluating systems available in the market today. Individual bowlers will be able to evaluate their performances on their own with the help of statistics and graphs provided by our system. More specifically, individuals will be able to keep a check on their consistency as a bowler based on bowling parameters like line, length, and speed.

In this system, we will be developing a grid of velostat polymer which will be used as a sensor for the detection of location of ball's impact on the pitch. This will help us to provide the line and length of the delivery.

The data collected from this sensor will be accumulated which can be later used to present to the user using Data Visualization techniques which will make this data intelligible for the user.

The final aim of this project is to provide a performance analyzing and evaluating system which can be utilized by any layman for the betterment of the sport

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1. INTRODUCTION

1.1 Introduction of project

Cricket is a bat and ball game played between two teams, 11 players each, on a field which has a rectangular 22-yard-long pitch in the centre. The basic purpose of the game is to score more runs than your opposing team.

A Cricket match is divided into periods called innings. During the innings, one team bats while the other fields and vice-versa. All 11 players on the fielding team are on the pitch at the same time however only two batsmen are the field at any one time.

Cricket fields tend to be oval in shape. The end which is marked off is called the boundary, with the rectangle “pitch” in the center.

Fig 1.1 Cricket Pitch

Fig 1.2 Pitch Dimensions

The essence of the sport is that a bowler delivers (i.e., bowls) the ball from his end of the pitch towards the batsman who, armed with a bat is "on strike" at the other end.

Bowling, in cricket, is the action of propelling the ball toward the wicket defended by a batsman. A player skilled at bowling is called a bowler. A single act of bowling the ball towards the batsman is called a ball or a delivery

The bowler's job is to take as many wickets as possible. That job will be a lot easier if they have control of their line and length, making the batsman's job of scoring runs that much more difficult. The line is the direction of the ball in accordance with the batsman's stumps, while the length is the area of the pitch where the ball bounces.

The length of the ball will determine whether the batsman will play on the front or back foot.

There are five areas the bowler can aim for:

Short length (8 or more meters in front of stumps)

Good length (6 to 8 meters in front of stumps)

Full length (2 to 6 meters in front of stumps)

Yorker(0 to 2 meters in front of stumps)

Different lines that the ball may be said to be traveling on maybe towards off stump, middle stump or leg stump, outside leg stump, or outside off stump.Line controls how much room the batsman has to play various shots, and sometimes dictates what shot he must play.

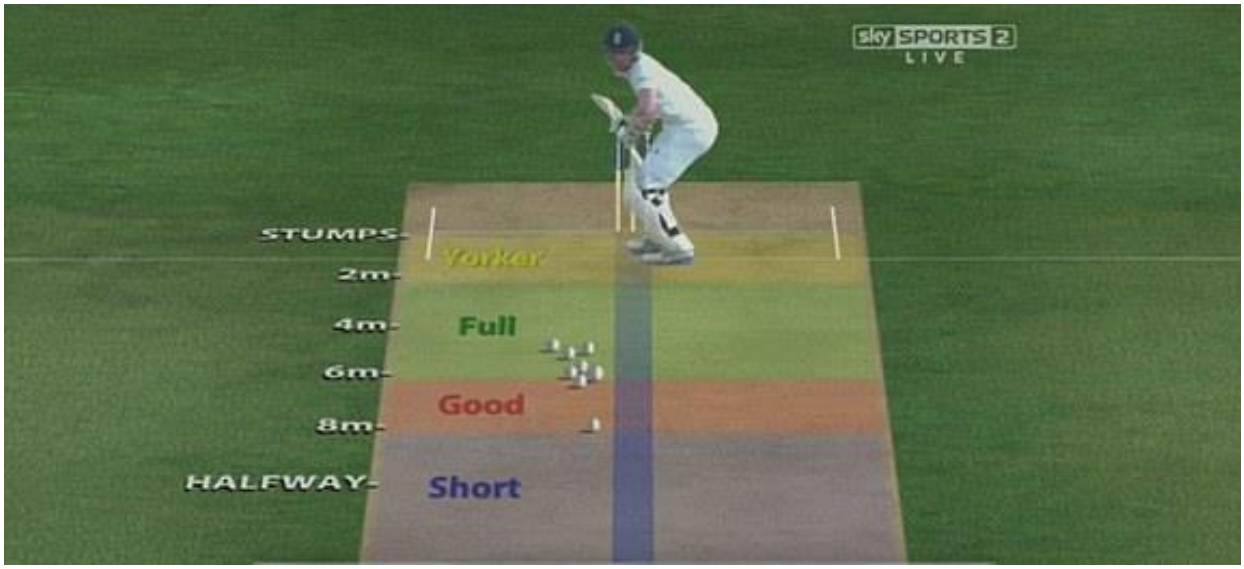


Fig 1.3 Length of delivery

In the past, cricketers could not see their performance parameters like bowling length, speed and line; and batting parameters too. If a cricketer was not performing in a match or was lacking in specific parameter,the fault could not be figured , Like is the bowler injured, less stamina or learning is wrong, that is bad muscle memory.

With the fast-paced improvement in technology nowadays it is integrated with sports. Even in cricket there are major changes due to this, like the hawk eye system, sensor bat, led stumps and stump microphone and camera. In this modern time where we are inclined towards perfection, we need to work on smart systems that can help sportspeople improve their gameplay that's easy for them and are less time consuming.

This Project "Cricket bowling analyser" is a reliable sytem that takes over the task of evaluating parameters of bowlers in the cricket very accurately and notifies about the parameters and fault delivery in the bowlers bowling via graphical and statistical data. This system will consist of VELOSTAT-based pressure mat which will be fabricated in a grid like structure and laid on the cricket pitch, for speed it will be radar gun. When the bowler's balls, the ball will fall on a square part of grid which will give the perfect spot on pitch and once it hits stump mat, the deviation from the original landing spot and final spot will give the line.

This system can also be enhanced by providing video processing through cameras which can be recorded and viewed later on. The cricket bowling analyser is a simple sytem that can turn out to be very helpful for the kids as well as adults who want to learn bowling without external assistance or in absence of coach

1.2 Aim and Objective of project

1. To provide a technology driven aid in the field of cricket that can help budding cricketers to enhance their performance.
2. Tracking statistics of players with a modern digital approach.
3. Provide a low cost easy to use alternative

1.3 Scope of the Project

1. To provide a technologically driven aid that will help in monitoring bowling performance by measuring line, length and speed of a delivery.

2. A low-cost system that will give necessary features of high cost professional system that are not easily affordable.

3. This Application will help a weaker section of society to portray their skills.

4. The main target audience of this project are school academies and small clubs (gymkhanas).

5. This can also be used by individual to target a particular spot to improve bowling and analyse himself.

CHAPTER 2

2. Literature Survey

2.1 Currently available Technology:

2.1.1 Hawk eye:

Hawk-Eye is a computer system used in numerous sports such as cricket, tennis, Gaelic football and volleyball, to visually track the trajectory of the ball and display a profile of its statistically most likely path as a moving image

Cost- 1.2 crores INR

Fig2.1 Hawk Eye

2.1.2 Pitch Vision:

Pitch vision is an innovative cricket training platform that facilitates an analytical system consisting of motion tracking and video analysis

Cost- 3 -10 Lakhs INR

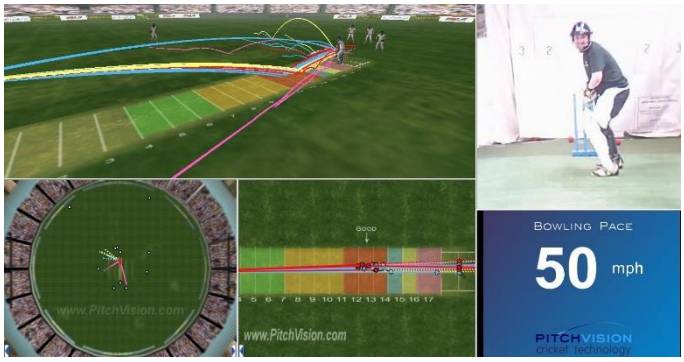


Fig 2.2 Pitch Vision

SYSTEM	TECHNOLOGY	HARDWARE	APPLICATION	COST (INR)
HAWK EYE	IMAGE PROCESSING	HIGH END CAMERA	MATCH	2.5CR
PITCH VISION	EMBEDDED SYSTEM & IMAGE PROCESSING	CAMERA, SENSOR MAT	MATCH & TRAINING	3LAKHS
CRICKET BOWLING ANALYZER	EMBEDDED SYSTEM	SENSOR MAT	TRAINING	30K

Table 2.1 Technology Comparison

The currently available systems considered for cricket, work on the principle of Image processing and require expensive hardware, and very high processing power and storage. Hence, we have designed a less expensive embedded system which requires less processing power and storage.

The main requirement of the embedded system is to collect data on line, length and speed for this purpose we have studied various sensors which can be used to perform the desired task.

2.2 Study of Sensor:

2.2.1 Piezoelectric sensor:

A piezoelectric sensor is a device that uses the piezoelectric effect, to measure changes in pressure, acceleration, temperature, strain, or force by converting them to an electrical charge. Piezoelectric sensors are versatile tools for the measurement of various processes. They are used for quality assurance, process control, for research and development in many industries.

Usability-Piezoelectric sensor are inexpensive pressure sensors but the output of these sensor is not stable so they are not suited for measurement of line and length of delivery as they fail to detect impact of ball and also they are easily prone to damage by high impact.

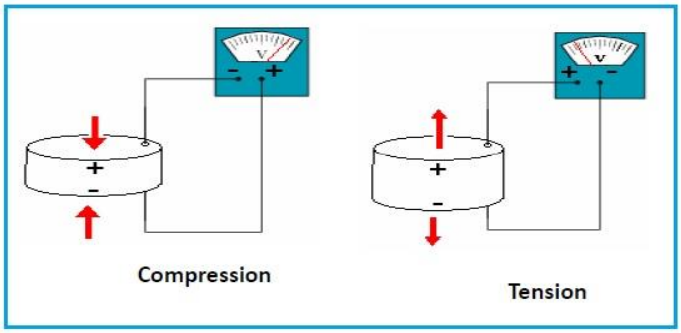


Fig 2.3 piezoelectric sensor

2.2.2 Force sensing resistor:

Force-sensing resistors consist of a conductive polymer, which changes resistance in a predictable manner following application of force to its surface. They are normally supplied as a polymer sheet or that can be applied by screen-printing. The sensing film consists of both electrically conducting and non-conducting particles suspended in matrix. The particles are sub-micrometer sizes, and are formulated to reduce the temperature dependence, improve mechanical properties and increase surface durability.

Usability-These are expensive sensors which are excellent for detection of impact but it is practically difficult and expensive to design a pressure sensing grid using these sensors.

Fig2.4 Force sensing resistor

2.2.3 Ultra sonic sensor:

Ultrasonic sensors are a type of acoustic sensor divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert electrical signals into ultrasound, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound.

Usability- Work on ultrasound and have a field of 120 degree so they are used for measurement of speed of delivery

Fig 2.5 Ultrasonic sensor

2.2.4 IR sensor:

An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measures only infrared radiation, rather than emitting it that is called as a passive IR sensor.

Usability-Due to low range of operation they are not advisable for use.

Fig 2.6 IR sensor

2.2.5 Velostat

Velostat is a piezoresistive material, meaning it's electrical resistance decreases when pressured. When sandwiched between two conductive layers, it has a wonderful range for making pressure and bend sensors. It can also be used for resistive sensing over distance, position sensing. Velostat is a back, opaque, volume-conductive, carbon-impregnated polyolefin. The electrical characteristics are not affected by age or humidity (but they do change when melted under the iron).

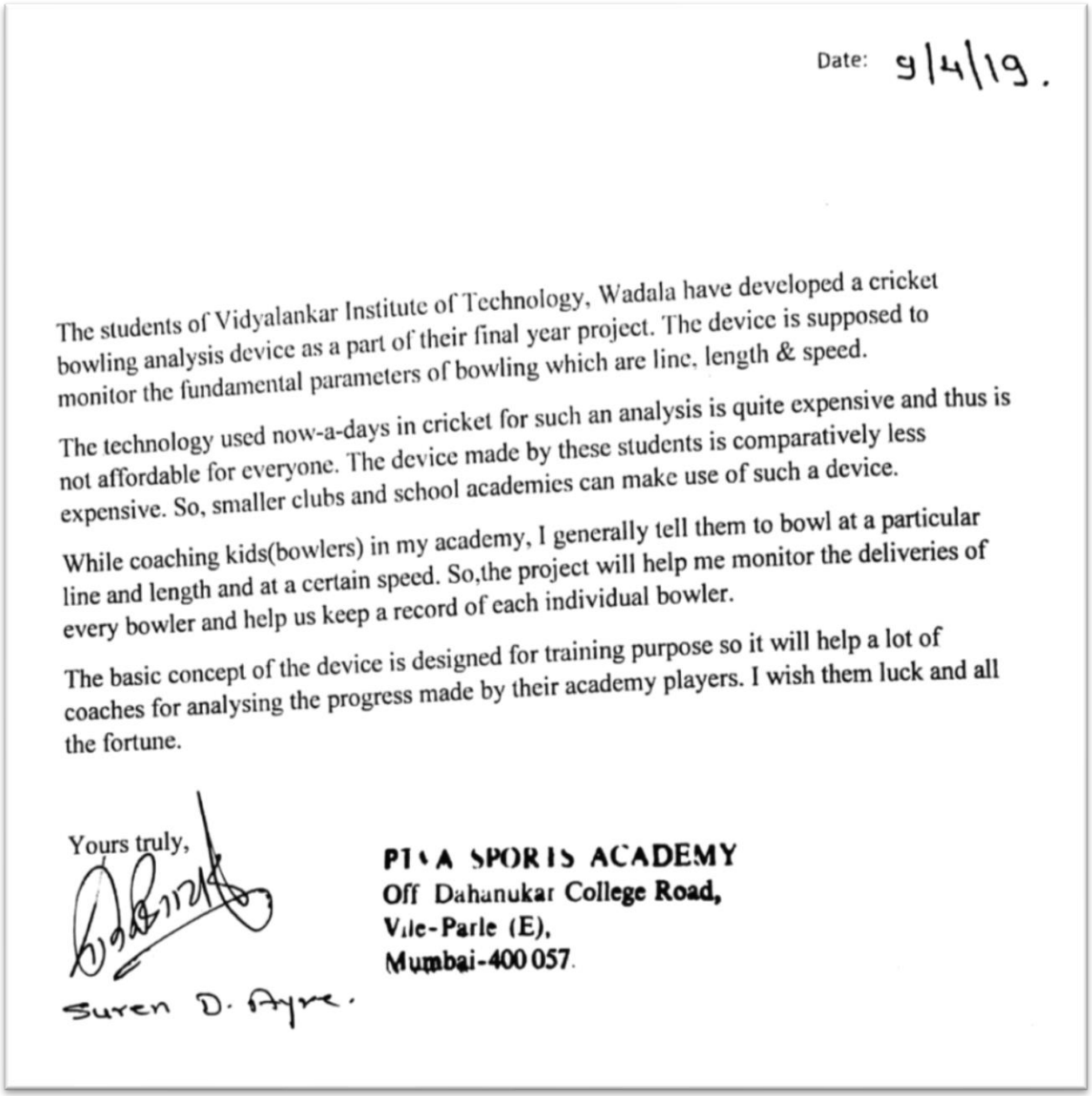
Usability-This is well suited for design of pressure sensor as per the requirement of the project. This polymer is soft, flexible, light, thin and can handle heavy impacts.

Fig 2.7 Velostat polymer

2.3 Market Research:

During the process of system designing we met with a few cricket clubs and explained them our system idea to get their feedback and suggestion on practical application of the proposed system.

We visited PTVA Sports Acadmey,Vile Parle,Mumbai to get a feedback on our idea .We met with their head coach Mr.Suren D.Ayre



Date: 9/4/19.

The students of Vidyalankar Institute of Technology, Wadala have developed a cricket bowling analysis device as a part of their final year project. The device is supposed to monitor the fundamental parameters of bowling which are line, length & speed.

The technology used now-a-days in cricket for such an analysis is quite expensive and thus is not affordable for everyone. The device made by these students is comparatively less expensive. So, smaller clubs and school academies can make use of such a device.

While coaching kids(bowlers) in my academy, I generally tell them to bowl at a particular line and length and at a certain speed. So,the project will help me monitor the deliveries of every bowler and help us keep a record of each individual bowler.

The basic concept of the device is designed for training purpose so it will help a lot of coaches for analysing the progress made by their academy players. I wish them luck and all the fortune.

Yours truly,

PTVA SPORTS ACADEMY
Off Dahanukar College Road,
Vile-Parle (E),
Mumbai-400 057.

Suren D. Ayre.

Fig 2.8 Feedback

CHAPTER 3

3. Problem Statement

To provide a cheaper alternative for the expensive systems available for cricket bowling performance analysis and help individual bowlers to evaluate their own consistency and progress.

CHAPTER 4

4. System Development

4.1 Block Diagram

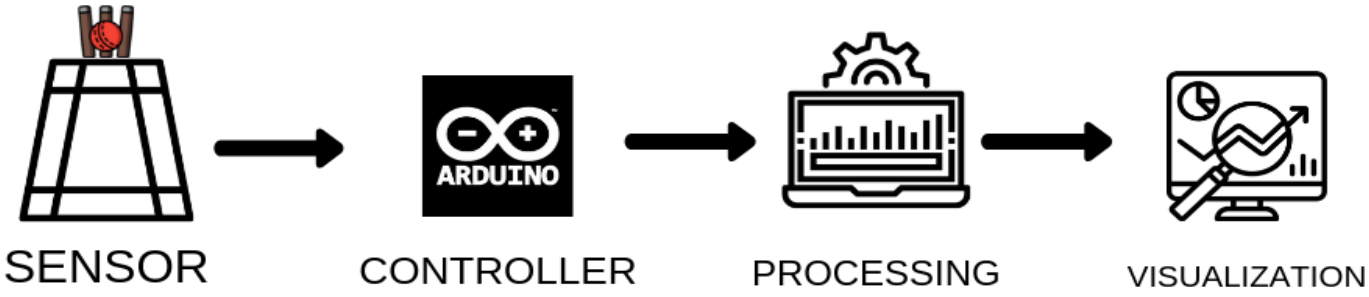


Fig 4.1 Block Diagram

The main goal of the system is to monitor and analyze cricket bowling parameters viz. line, length and speed which are the essential parameters of the multi-faceted sport for any budding cricketer. This will help the player to access and improve upon his/her bowling engendering higher performances. Due to this system, the fundamentals of any cricket bowler will be tested and polished.

An embedded system is designed that uses pressure sensors developed using velostat to detect the location of ball impact, which is then interfaced with the controller, and the acquired data is processed appropriately and is used to give proper information about the line, length, and length of the delivery.

This processed data will be stored in the storage device and specific directory will be maintained for every user for all the overs bowled and this data can be processed using data visualization techniques and can be represented in the form of graphs and chart which provide a summary of the performance and help in better understanding the data instead of mere table or spread sheets. Data can be stored for a longer duration which will help in tracking a performance for a period which gives all the insights in the various trends that has taken place during this period.

4.2 Circuit Diagram:

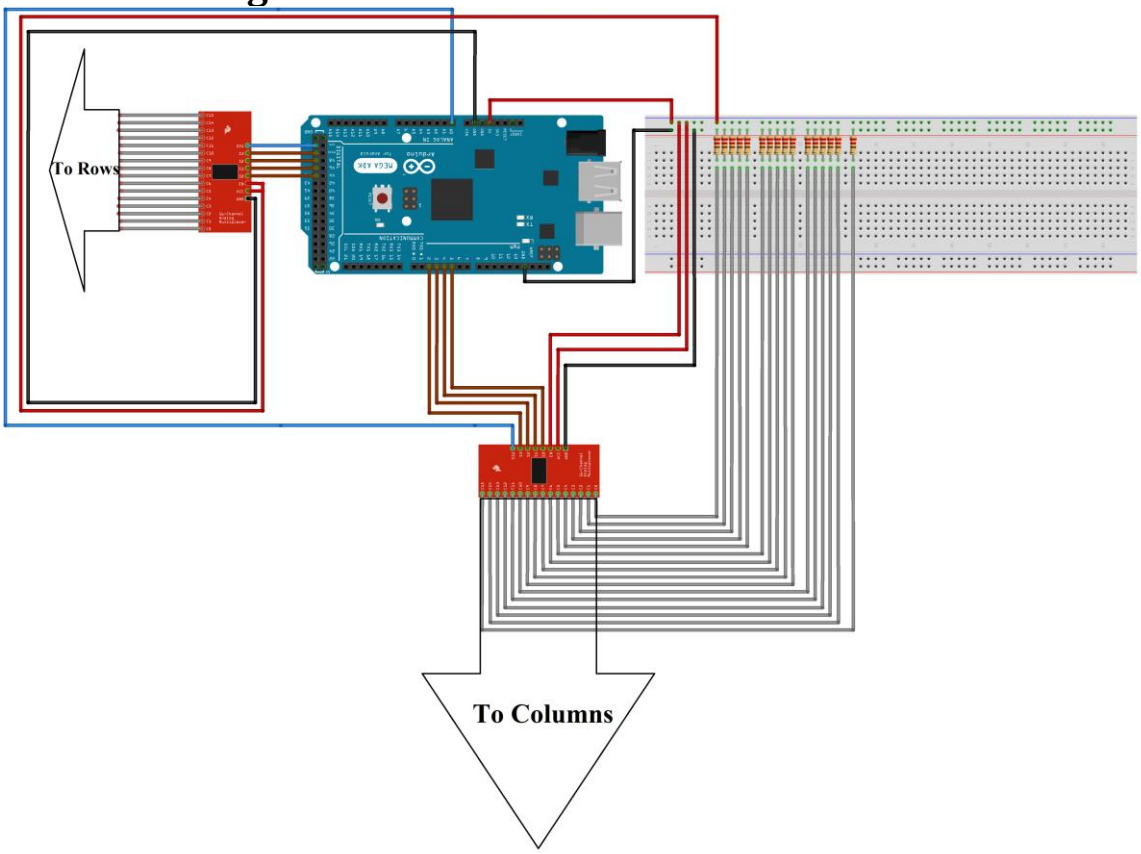


Fig 4.2Main Circuit

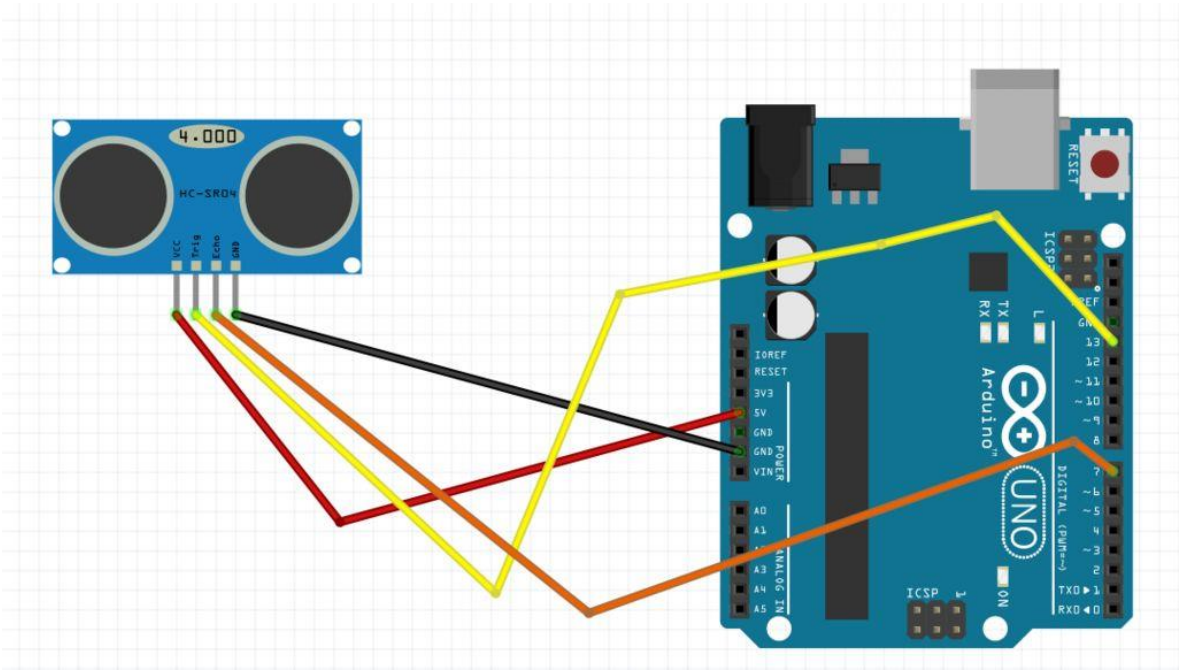


Fig 4.3 Speed Circuit

4.3 PCB

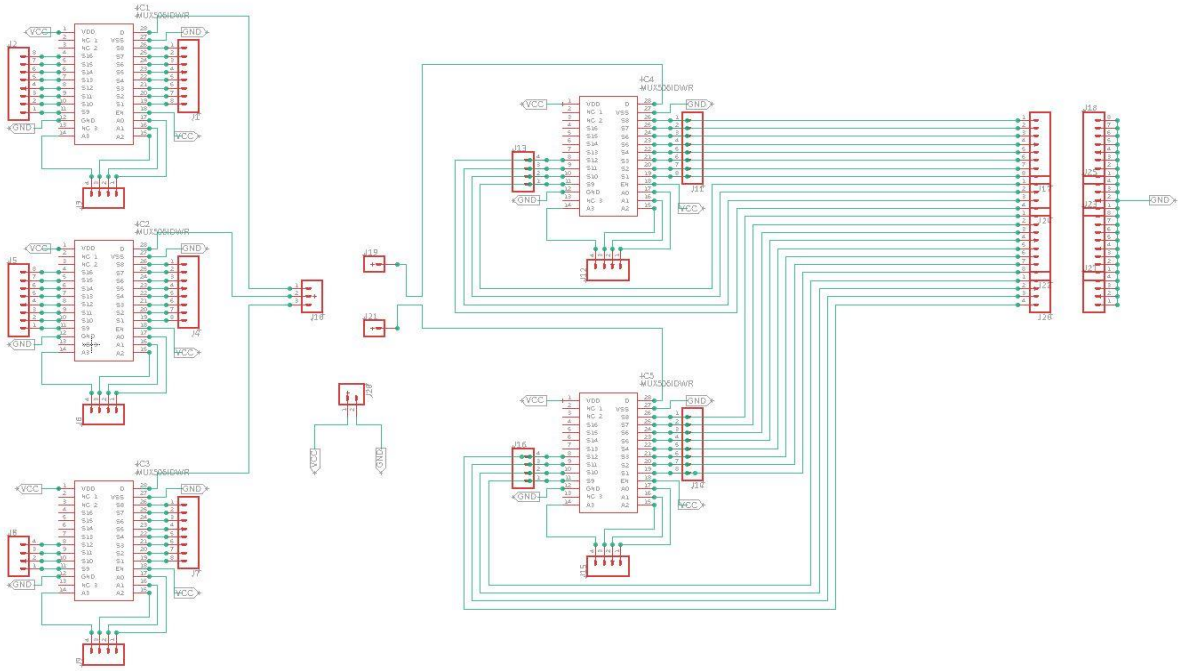


Fig 4.4 PCB schematic

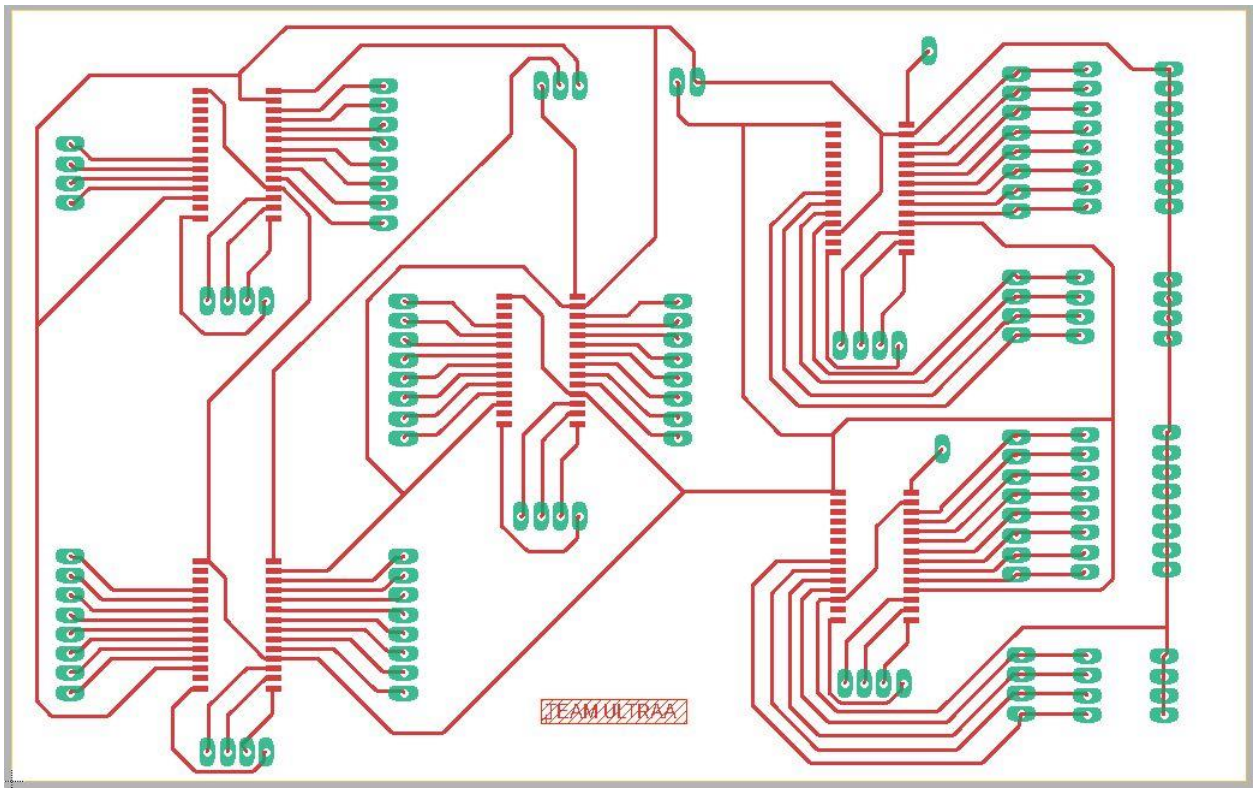


Fig 4.5 PCB

4.4 Explanation:

The circuit is designed for measuring line and length of ball delivery, it consists of 2 pair of 16:1 mux interfaced with Arduino. One pair of mux is interfaced with Pitch pressure sensing matrix for measuring length of delivery and another pair is connected with back screen matrix for measuring line of delivery.

The 16 channels of 1 mux are connected to rows of matrix and the 16 channels of other mux are connected to columns of same matrix. Select pins of multiplexers are connected to Arduino, switching of multiplexer channels is done via Arduino.

A voltage divider section is created on the output side and the output of this section is fed to 16 channels of second mux then this Output is given to analog pin of Arduino.

An ultrasonic sensor is interfaced with Arduino for measuring the speed of delivery and the speed data is also fed to csv file.

4.5 Working:

The primary task of the system is to measure line, length, speed of delivery. For measurement different sensors are interfaced with microcontroller to acquire the data.

4.5.1 Pressure sensing mat:

BIG MAT				
COPPER STRIP		LENGTH (cm)	BREADTH (cm)	NO OF STRIPS
	ROW	75	3.5	30
	COLUMN	180	3.5	12
VELOSTAT		LENGTH (cm)	BREADTH (cm)	
		180	75	
SMALL MAT				
COPPER STRIP		LENGTH (cm)	BREADTH (cm)	NO OF STRIPS
	ROW	75	3.5	12
	COLUMN	75	3.5	12
VELOSTAT		LENGTH (cm)	BREADTH (cm)	
		75	75	

Table 4.1 Senor mat dimensions

Pressure sensing mat is designed using velostat polymer. The aim of this sensor is to give the exact location at which pressure is applied. Matrix structure of the mat is helpful in finding the location of pressure change. The sensor will return 2D coordinates of the location of pressure change. A number of copper strips are used as rows and columns and the velostat polymer is sandwiched between the copper strips to create the pressure sensing matrix. The mat used in the system consists of a matrix structure with 30 rows and 12 columns. The data given by this matrix is used to track the length of delivery. Another matrix of 12 rows and 9 columns is used to measure the line of the delivery. The 30 rows X 12 columns mat used for determining the length of the delivery is placed on the pitch and the 12 rows X 9 columns mat is placed behind the stumps on a vertically oriented screen, which helps in determining the line of delivery.

The matrix structure is interfaced with the microcontroller using analog multiplexer mux506. The multiplexers are used to control the input and output of the matrix. The rows and columns of the matrix are connected to the channels of separate set of multiplexers responsible for controlling rows and columns independently. The row section of the matrix is used as input and the columns are used to measure the output.

The operation of pressure sensing matrix is dependent on change in impedance of the polymer, according to the ohm's law voltage is directly proportional to impedance, using this law the voltage change at the location of ball's impact is monitored using the mux.

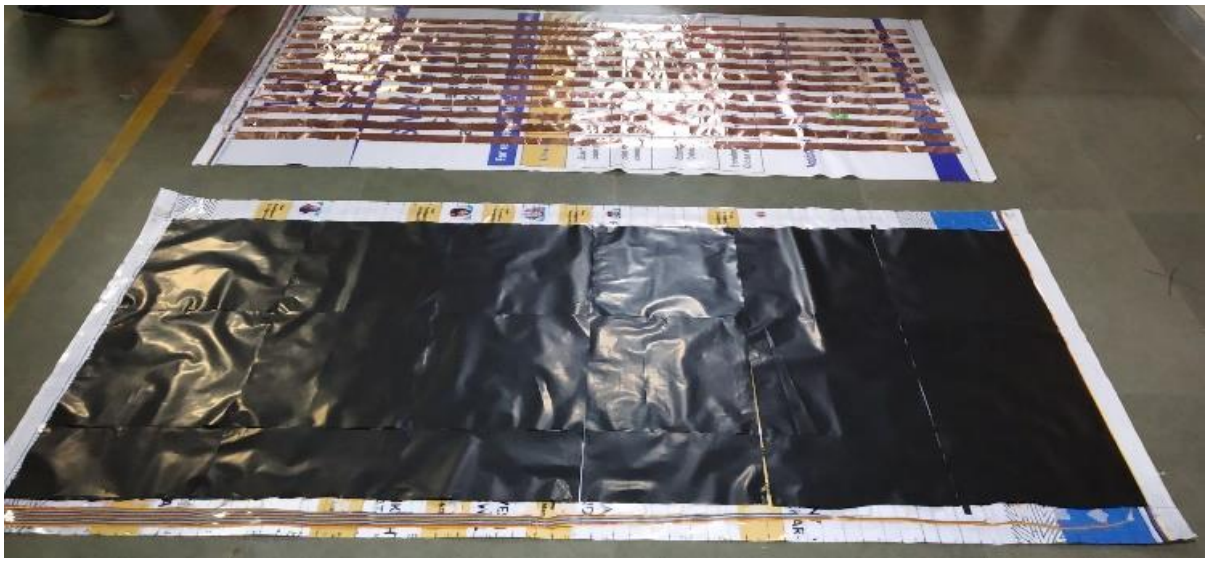


Fig 4.6 Implemented Sensor

The microcontroller sends a 5V input signal to the multiplexers,which are controls, the switching of rows. The signal is transferred to one channel at a time and the columns are continuously monitored for detecting any major surge in voltage. This voltage surge from the multiplexers is fed into the microcontroller.

The system takes approximately **64 ms** to scan both the mats with Arduino Mega 2560 (16 MHz crystals). After removing the pre-scaler, the scanning time is reduced to **11 ms**. This helps us to detect the impact of the ball.

4.5.2 Ultrasonic speed gun:

The speed gun is used for measuring the speed of delivery and it is designed using ultrasonic sensor. Ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception.

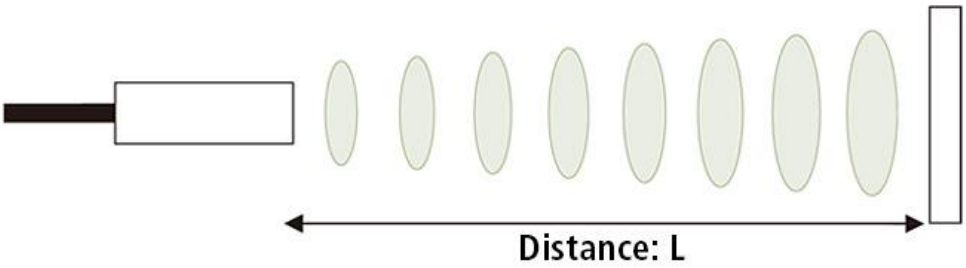


Fig 4.7 ultrasonic waves

An ultrasonic sensor uses a single ultrasonic element for both emission and reception. In a reflective model ultrasonic sensor, a single oscillator emits and receives ultrasonic waves alternately. This enables miniaturization of the sensor head.

Ultrasonic sensor module comprises of one transmitter and one receiver. The transmitter can deliver 40 KHz ultrasonic sound while the maximum receiver is designed to accept only 40 KHz sound waves. The receiver ultrasonic sensor that is kept next to the transmitter shall thus be able to receive reflected 40 KHz, once the module faces any obstacle in front. Thus, whenever any obstacles come ahead of the ultrasonic module it calculates the time taken from sending the signals to receiving them since time and distance are related for sound waves passing through air medium at 343.2m/sec.

Distance calculation:

The distance can be calculated with the following formula:

Distance L = 1/2 × T × C

where L is the distance, T is the time between the emission and reception, and C is the sonic speed. (The value is multiplied by 1/2 because T is the time for go-and-return distance.)

Speed measurement:

The ultrasonic sensor is placed on the stumps facing towards the bowler. The sensor continuously emits ultrasound, when the ball is bowled the sensor detects the ball twice, one at distance d1 and one more time at distance d2. So, only two waves reflected back at the distances d1 and d2 are considered. This enables us to determine the time between the two distances and hence the speed of the delivery.

4.5.3 Data Acquisition and Visualization:

The data collected from sensor matrix is fed to Arduino where processing of the data takes places. Further, the processed data is then sent serially to Processing 3 software which is used to format the data and load in CSV file. At the same time the formatted data is visualized in real time. CSV files are used to store the data on line, length and speed of delivery. This accumulated data is used for long-term analysis of trends in performance. The data from CSV files is loaded in python and analyzed using graphs to show the tabular data in simple form. Finally, a GUI is developed to give user a smooth access to this data.

CHAPTER 5

5. HARDWARE DETAILS

5.1Hardware Requirments:

SR NO	COMPONENTS	SPECIFICATION
1	Arduino Nano	ATmega328P,16MHz
2	Arduino Mega	ATmega 2560 AU
3	Velostat polymer	50,000 ohm/sq metre
4	MUX 506	16:1 Analog Mux
5	Ultrasonic Sensor	Detection range: 2cm – 1000cm
6	Copper Stripes	Conductivity: 5.96 x 10^7 S/m

Table 5.1: Component Specifications

5.1.1Arduino Nano:

Arduino Nano is an open source platform for programming. The Arduino Nano is a small, complete, and breadboard-friendly board based on the Atmega328P.It has 22 digital input/output pins (of which 6 can be used as PWM outputs), 8 analog inputs, a 16 MHz ceramic resonator, a USB connection, an ICSP header, and automatic reset.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Arduino Nano has many facilities for communicating with a computer, another Arduino, or other microcontrollers. The Atmega328P provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). The Arduino Nano can be programmed with the Arduino software (download). The Atmega328P on the Arduino Nano comes preburnt with a boot loader that allows you to upload new code to it without the use of an external hardware programmer.

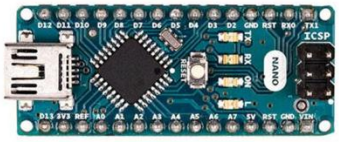


Fig 5.1 Arduino Nano

5.1.2 Arduino Mega:

Arduino is an open-source physical computing platform based on a simple i/o board and a development environment that implements the Processing/Wiring language. Arduino can be used to develop stand-alone interactive objects or can be connected to software on your computer (e.g. Flash, Processing, MaxMSP). The open-source IDE can be downloaded for free (currently for Mac OS X, Windows, and Linux).

The Arduino Mega is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. Never fear for accidental electrical discharge, either since since the Mega also includes a plastic base plate to protect it!

The Mega 2560 R3 also adds SDA and SCL pins next to the AREF. In addition, there are two new pins placed near the RESET pin. One is the IOREF that allow the shields to adapt to the voltage provided from the board. The other is a not connected and is reserved for future purposes. The Mega 2560 R3 works with all existing shields but can adapt to new shields which use these additional pins.

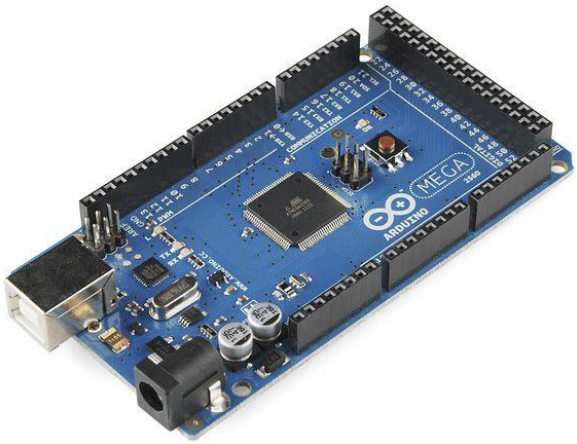


Fig 5.2 Arduino Mega

5.1.3 Velostat Polymer:

Velostat (50000-ohm, 4 mil), which is produced by 3M and consists of a polymeric foil (11μm thickness) impregnated with carbon black. For the electrodes, we use a copper sheet strip (40 microns), which is also commonly available. The outer protection layer is made of a plastic flex.

Due to its properties of changing its resistance with either flexing or pressure it is becoming popular with hobbyists for making inexpensive sensors for microcontroller experiments. One example of this is to make shoes which light up when the wearer takes a step. Since the resistance in the circuit is reduced when pressure is applied, this reading can indicate when weight is applied or removed from the shoes.

Fig 5.3 Velostat

5.1.4 Texas Instrument Mux506:

The MUX506 and MUX507 (MUX50x) are modern complementary metal-oxide semiconductor (CMOS) precision analog multiplexers (muxes). The MUX506 offers 16:1 single-ended channels, whereas the MUX507 offers differential 8:1 or dual 8:1 single-ended channels. The MUX506 and MUX507 work equally well with either dual supplies (±5 V to ±18 V) or a single supply (10 V to 36 V). These devices also perform well with symmetric supplies (such as VDD = 12 V, VSS = –12 V), and unsymmetric supplies (such as VDD = 12 V, VSS = –5 V). All digital inputs have transistor-transistor logic (TTL) compatible thresholds, providing both TTL and CMOS logic compatibility when operating in the valid supply voltage range.

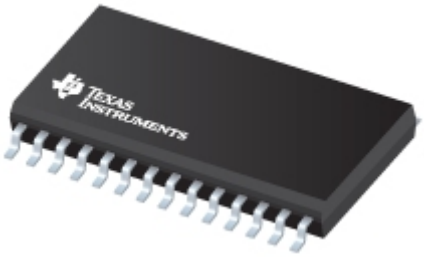


Fig 5.4 Mux506

5.1.6 Ultrasonic Sesnor:

Ultrasonic sensor module comprises of one transmitter and one receiver. The transmitter can deliver 40 KHz ultrasonic sound while the maximum receiver is designed to accept only 40 KHz sound waves. The receiver ultrasonic sensor that is kept next to the transmitter shall thus be able to receive reflected 40 KHz, once the module faces any obstacle in front. Thus, whenever any obstacles come ahead of the ultrasonic module it calculates the time taken from sending the signals to receiving them since time and distance are related for sound waves passing through air medium at 343.2m/sec.

Fig 5.5 Ultrasonic Sensor

CHAPTER 6

6. SOFTWARE DETAILS

6.1 Software Requirements:

6.1.1 Processing3:

Fig 6.1 Processing 3

Processing is an open-source graphical library and integrated development environment (IDE) / playground built for the electronic arts, new media art, and visual design communities with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context.

Processing uses the Java language, with additional simplifications such as additional classes and aliased mathematical functions and operations. As well as this, it also has a graphical user interface for simplifying the compilation and execution stage.

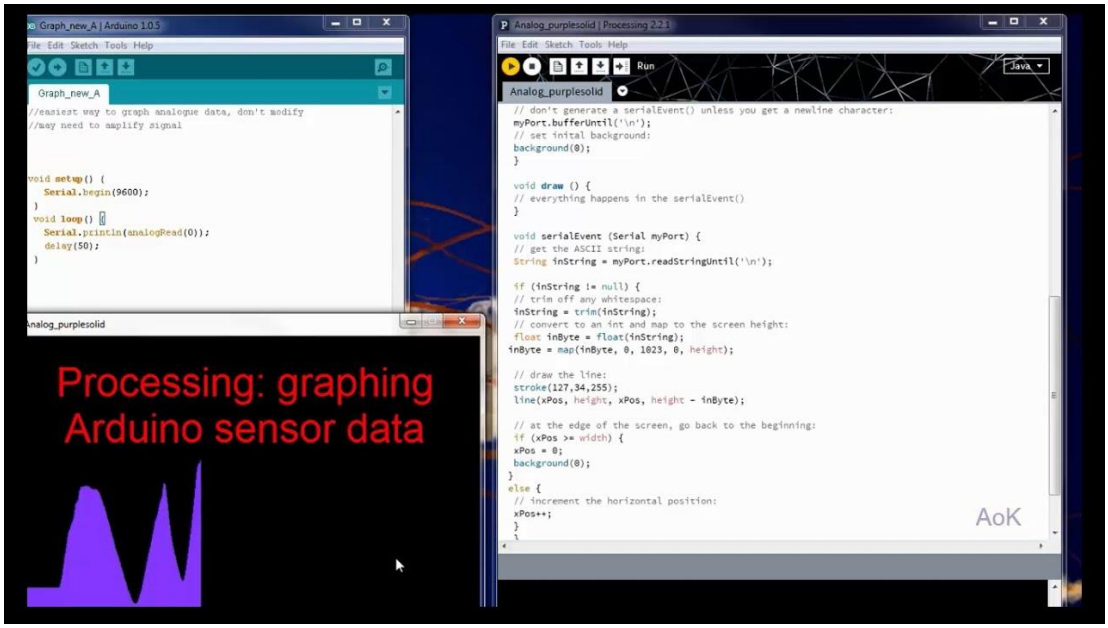


Fig 6.2 Processing 3 display

Interfacing Arduino with Processing3 using a simple example of turning a LED ON & OFF

Arduino Circuit

The connections for this tutorial are very easy. Connect the two end pins of the 1k potentiometer to the 5V and the GND pin of the Arduino. Then connect the middle pin of the potentiometer to the A0 on Arduino. Then connect the positive pin of the LED to pin 7 on the Arduino and the negative pin of the LED to the GND pin through the 220 ohm resistor.

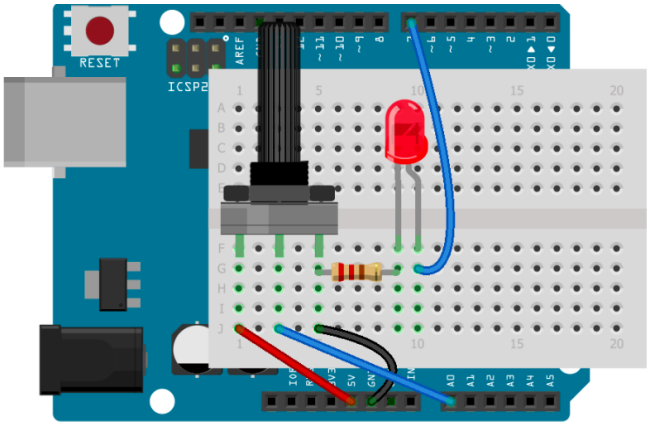


Fig 6.3 Led Arduino interfacing

Working

The Arduino IDE and the Processing IDE will communicate with each other through serial communication. The Processing IDE has a serial library which makes it easy to communicate with the Arduino.

When we move the potentiometer knob, the Arduino will send a value between 0 and 255 to the Processing IDE. The Processing IDE will then change the color of the serial window according to the movement of the potentiometer knob.

Similarly, when we press the mouse button in the serial window of the Processing IDE, the Processing IDE will send a '1' or '0' depending on the left or right mouse button to the Arduino IDE. The Arduino IDE will then turn the LED ON or OFF according to the button pressed.

Arduino Code

```
int led_pin = 7;    // Initializing the LED pin
int pot_pin = A0;   // Initializing the Potentiometer pin
int pot_output;     // Declaring a variable for potentiometer output
void setup ( ) {
pinMode(led_pin, OUTPUT); // Declaring the LED pin as output pin
Serial.begin(9600);       // Starting the serial communication at 9600 baud rate
}
void loop ( ) {
pot_output = analogRead (pot_pin); // Reading from the potentiometer
int mapped_output = map (pot_output, 0, 1023, 0, 255); // Mapping the output of potentiometer to 0-
255 to be read by the Processing IDE
Serial.println (mapped_output);     // Sending the output to Processing IDE

if (Serial.available ( ) > 0) {   // Checking if the Processing IDE has send a value or not
char state = Serial.read ( );    // Reading the data received and saving in the state variable
if(state == '1')             // If received data is '1', then turn on LED\
{
digitalWrite (led_pin, HIGH);
}
if (state == '0') {     // If received data is '0', then turn off led
digitalWrite (led_pin, LOW);
}
}
delay(50);
}
```

Processing

The Processing IDE will accept the data from the Arduino IDE through the Serial communication and will change the color of the serial window according to the data. It will also send ‘1’ or ‘0’ depending on the mouse button pressed. The Processing IDE has a serial library which makes it able to communicate with the Arduino IDE.

```
import processing.serial.*;    // Importing the serial library to communicate with the Arduino
Serial myPort;      // Initializing a vairable named 'myPort' for serial communication
float background_color ;   // Variable for changing the background color
void setup ( ) {
size (500,  500);     // Size of the serial window, you can increase or decrease as you want
myPort  =  new Serial (this, "COM3",  9600); // Set the com port and the baud rate according to the Arduino IDE
myPort.bufferUntil ( '\n' );   // Receiving the data from the Arduino IDE
}
void serialEvent  (Serial myPort) {
background_color  =  float (myPort.readStringUntil ( '\n' ) ) ;  // Changing the background color according to received data
}
void draw ( ) {

background ( 150, 50, background_color );   // Initial background color, when we will open the serial window

if ( mousePressed  &&  ( mouseButton  ==  LEFT ) ) { // if the left mouse button is pressed

myPort.write ( '1' ) ;       // send a '1' to the Arduino IDE

}

if  ( mousePressed  &&  ( mouseButton == RIGHT ) ) {  // if the right mouse button is pressed

myPort.write ( '0' ) ;     // Send a '0' to the Arduino IDE
```

6.1.2 Python 3:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python features a comprehensive standard library, and is referred to as "batteries included".

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model. Python and CPython are managed by the non-profit Python Software Foundation.

6.4 Python

6.1.3 PyQt:

PyQt is one of the most popular Python bindings for the Qt cross-platform C++ framework. PyQt developed by Riverbank Computing Limited. Qt itself is developed as part of the Qt Project. PyQt provides bindings for Qt 4 and Qt 5. PyQt is distributed under a choice of licences: GPL version 3 or a commercial license.

PyQt is available in two editions: PyQt4 which will build against Qt 4.x and 5.x and PyQt5 which will only build against 5.x. Both editions can be built for Python 2 and 3. PyQt

contains over 620 classes that cover graphical user interfaces, XML handling, network communication, SQL databases, Web browsing and other technologies available in Qt.

The latest iteration of PyQt is v5.11.3. It fully supports Qt 5.11.2.

PyQt4 runs on Windows, Linux, Mac OS X and various UNIX platforms. PyQt5 also runs on Android and iOS.

6.1.4 Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

CHAPTER 7

7. Cost of Materials

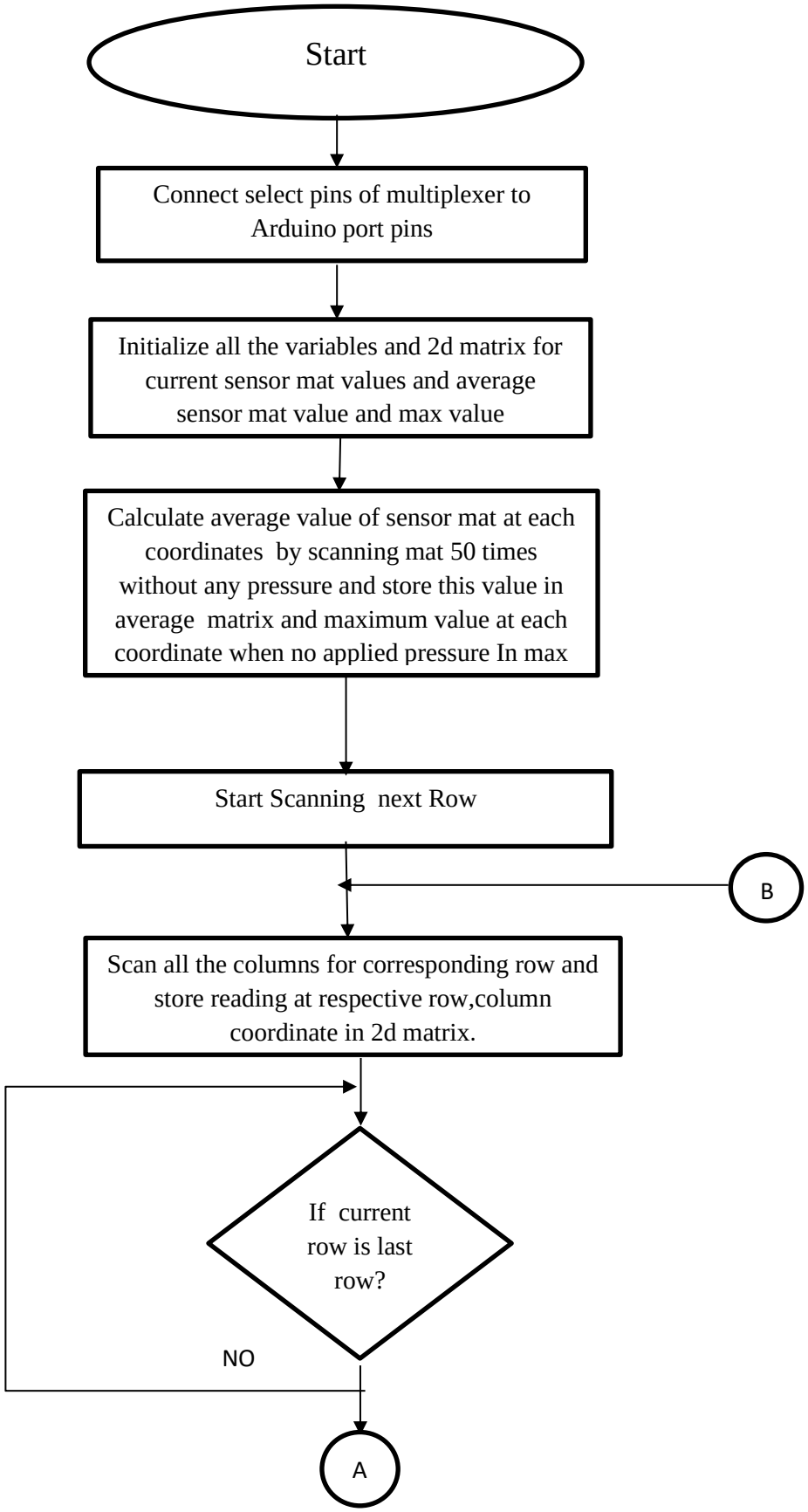
7.1 Costing Details:

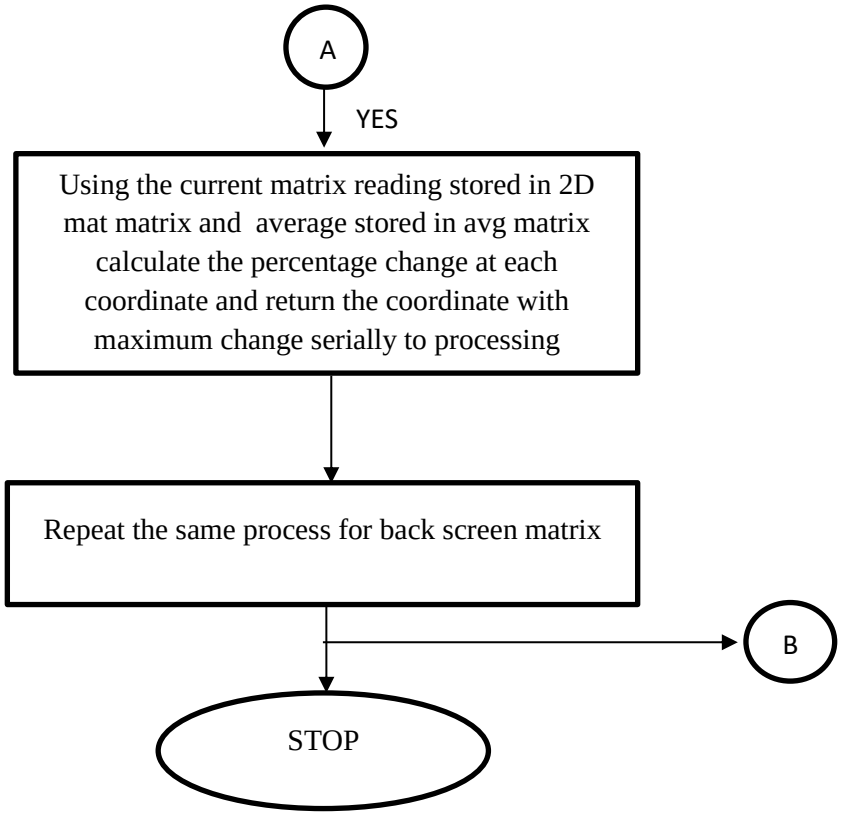
Sr no	Name	Quantity	Cost(INR)
1	Velostat	1.8 sq.meter	11000
2	Arduino Mega	1	800
3	Mux506	4	400
4	Aruino Nano	1	500
5	PCB	1	1000
6	Ultrasonic sensor	1	100
8	Copper sheet	30 sq feet	1800
9	Wires ,cables and accessories		6000
Total			**21,500**

Table 7.1 Cost of Materials

CHAPTER 8

8. Code Flow chart




A
YES
Using the current matrix reading stored in 2D mat matrix and average stored in avg matrix calculate the percentage change at each coordinate and return the coordinate with maximum change serially to processing
Repeat the same process for back screen matrix
B
STOP

CHAPTER 9
9. Results

Fig 9.1 System Connection

This is how the setup looks like when the whole system is assembled. The larger mat is overlaid on the ground (which gives the length) and the smaller mat is held behind (which gives the line).

On an average the ball takes approximately 11 ms to bounce off the pitch. Our system can detect whether the ball has bounced on the pitch if the bounce time in 11 ms. That is, the arduino scans the enter mat (both mats) in 11 ms.

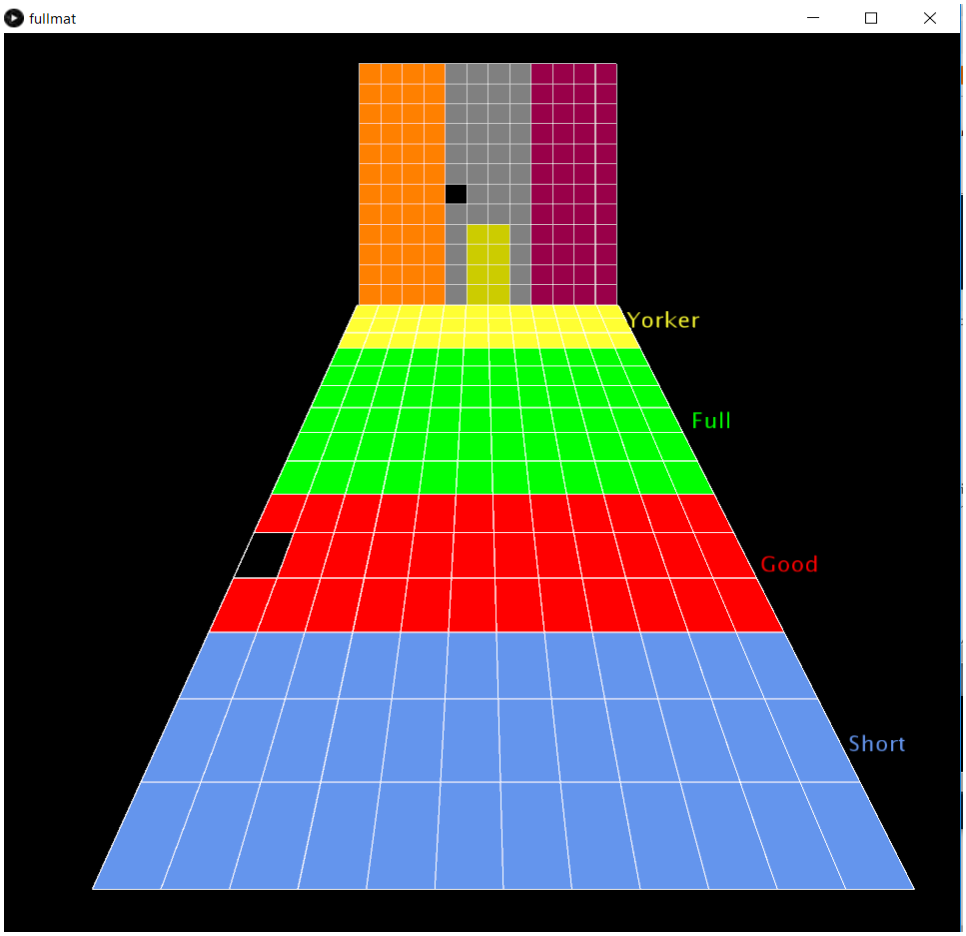
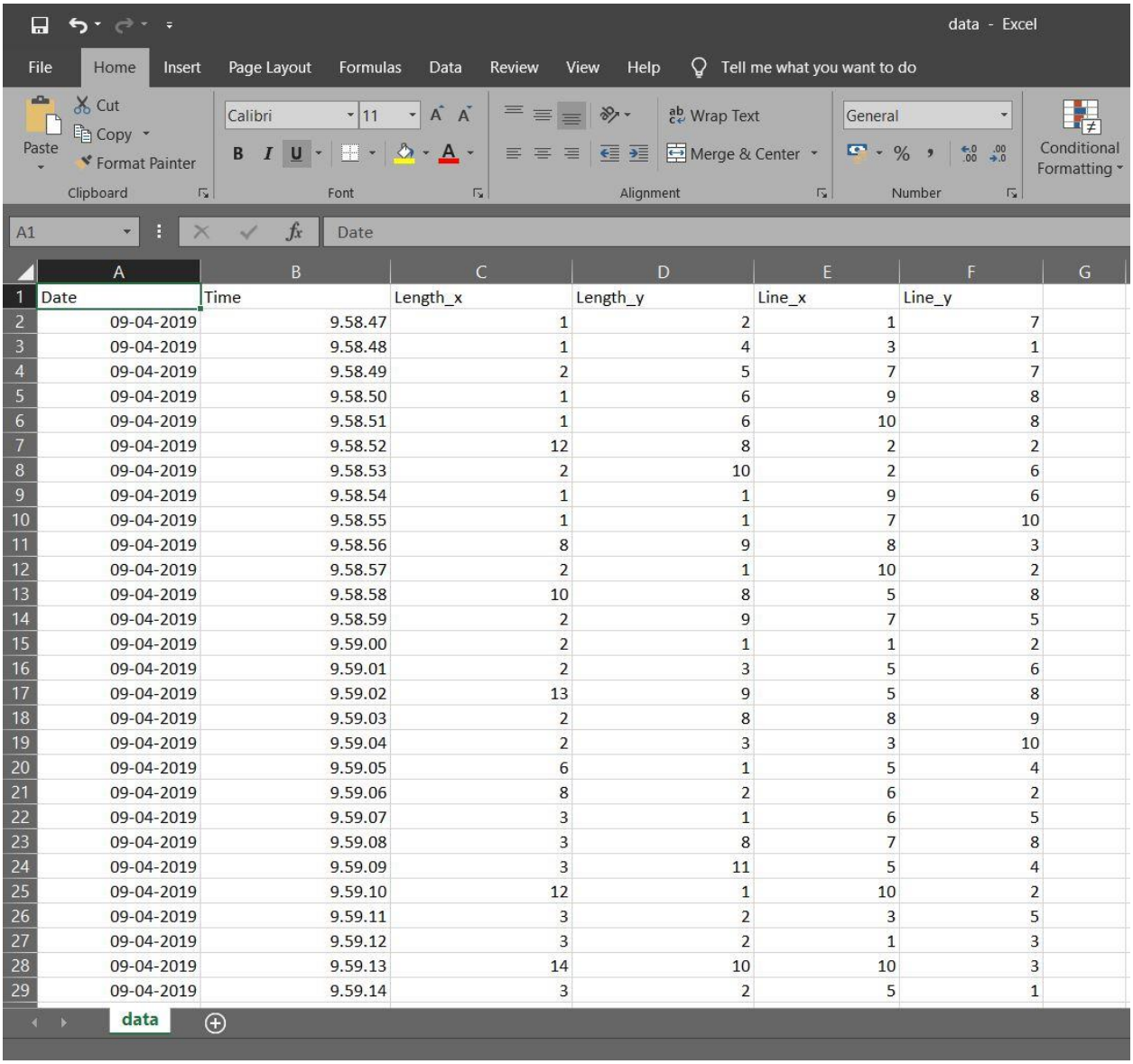


Fig 9.2 Processing3 visualization

This is virtual representation of the real setup developed using processing3.It shows live visualization of the real time delivery.The bottom pitch is divided into 4 section which shows the length of delivery viz. Yorker,full,good,short.The back screen is divided into 3 section which shows line of delivery viz. off,middle,leg.

The black box visible on the screen shows the exact coordinates of the delivery and the colour code helps in deciphering the length and line of delivery.processing3 also collects the data from Arduino and loads the data into csv file.



data - Excel

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

Cut Copy Format Painter Paste Clipboard

Calibri 11 Font

Wrap Text Merge & Center Alignment

General Number

Conditional Formatting

A1 Date

	A	B	C	D	E	F	G
1	Date	Time	Length_x	Length_y	Line_x	Line_y	
2	09-04-2019	9.58.47	1	2	1	7	
3	09-04-2019	9.58.48	1	4	3	1	
4	09-04-2019	9.58.49	2	5	7	7	
5	09-04-2019	9.58.50	1	6	9	8	
6	09-04-2019	9.58.51	1	6	10	8	
7	09-04-2019	9.58.52	12	8	2	2	
8	09-04-2019	9.58.53	2	10	2	6	
9	09-04-2019	9.58.54	1	1	9	6	
10	09-04-2019	9.58.55	1	1	7	10	
11	09-04-2019	9.58.56	8	9	8	3	
12	09-04-2019	9.58.57	2	1	10	2	
13	09-04-2019	9.58.58	10	8	5	8	
14	09-04-2019	9.58.59	2	9	7	5	
15	09-04-2019	9.59.00	2	1	1	2	
16	09-04-2019	9.59.01	2	3	5	6	
17	09-04-2019	9.59.02	13	9	5	8	
18	09-04-2019	9.59.03	2	8	8	9	
19	09-04-2019	9.59.04	2	3	3	10	
20	09-04-2019	9.59.05	6	1	5	4	
21	09-04-2019	9.59.06	8	2	6	2	
22	09-04-2019	9.59.07	3	1	6	5	
23	09-04-2019	9.59.08	3	8	7	8	
24	09-04-2019	9.59.09	3	11	5	4	
25	09-04-2019	9.59.10	12	1	10	2	
26	09-04-2019	9.59.11	3	2	3	5	
27	09-04-2019	9.59.12	3	2	1	3	
28	09-04-2019	9.59.13	14	10	10	3	
29	09-04-2019	9.59.14	3	2	5	1	

data

Fig 9.3 CSV File

The above figure shows the format in which data is stored in csv files. The line and length data from the processing software is stored along with date and time instantaneously, when it was acquired. This csv file helps in long term storage of data and this data can be retrieved later for further graphical analysis.length_x and length_y represents the x,y coordinate of lengthparticular delivery similarly line_x and line_y gives the x,y coordinate for line. Date-time stamp is added for aiding the processing of data for generating graphs.

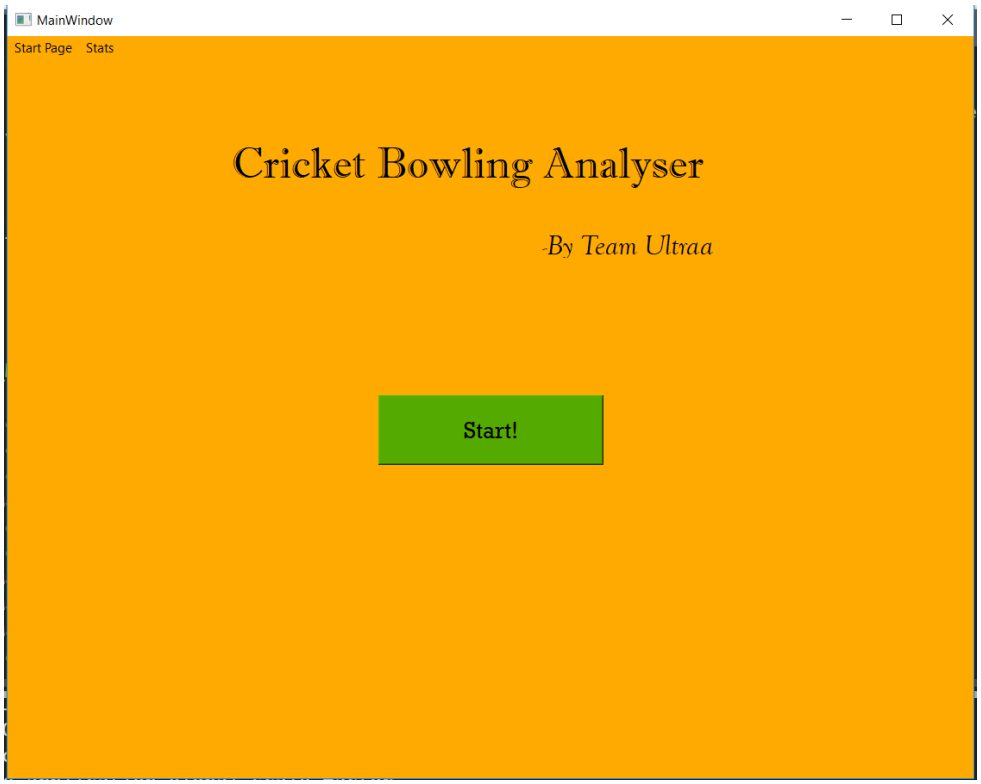


Fig 9.5 GUI Home Page

The above figure represents the homepage of GUI(Graphic User Interface)

Fig 9.5 GUI Second Page

Figure shows the second window of GUI where user can select any of the 2 given options as per his/her preference.

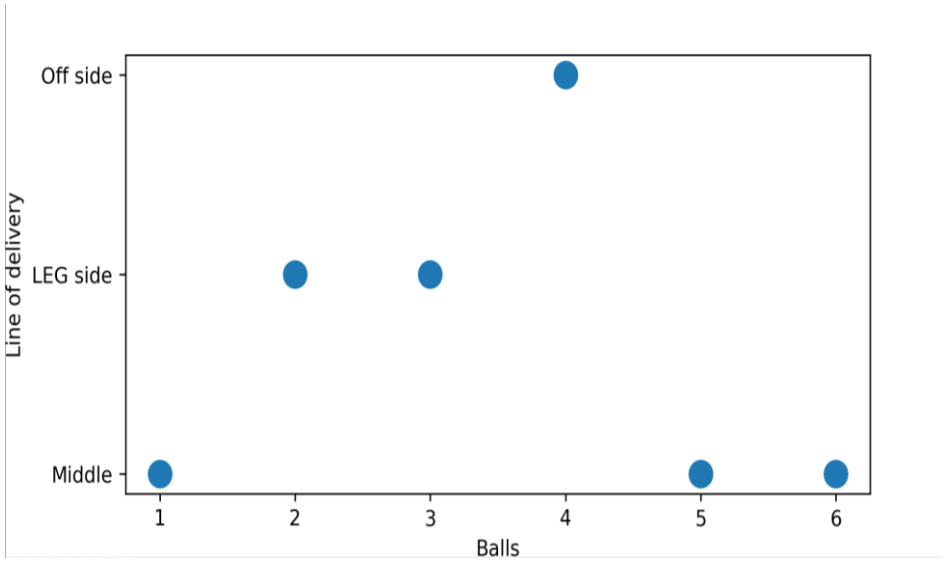


Fig 9.6 Line Graph

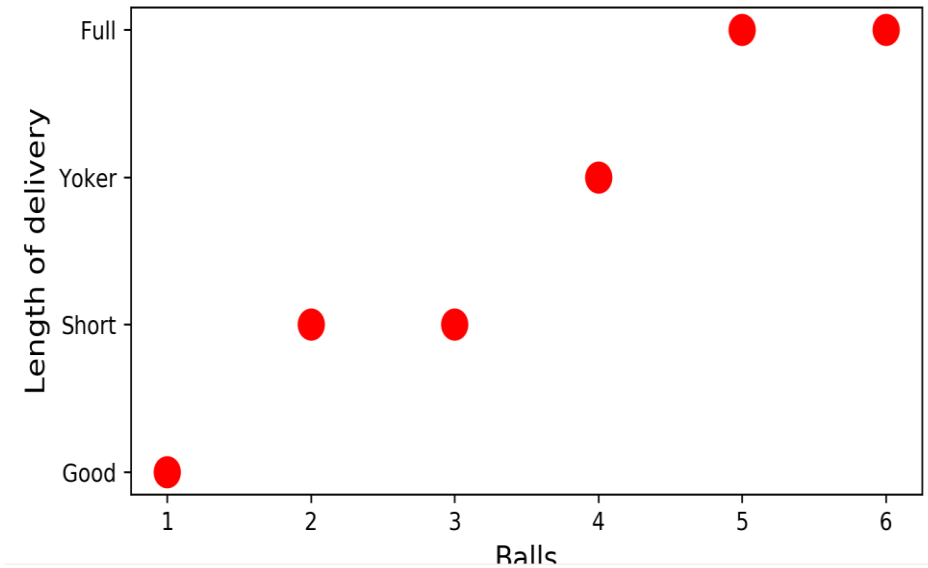


Fig 9.7 Length Graph

These graphs are used to provide a visual representation of the delivery to user for later use new graphs are generated after every over.

CHAPTER 10

10. Conclusion and Future Scope

10.1 Conclusion:

The proposed system Cricket Bowling is a low-cost alternative that can help in monitoring of bowling system which can effectively calculate the basic 3 parameters of the cricket coaching system. Self-fabricated Sensors are used to detect the delivery balls. Furthermore, the user can monitor the individual balls continuously and the status of ball's location is updated in real time. Therefore, this proposed system can overcome the traditional method of single coach system. At the same time, it offers cost savings and reliability which convince the users all time. The advantages consist of personalized cricket coaching system and tracking.

10.2 Future Scope:

- This system can further be improved by upgrading the speed measurement equipment, which currently works on the not so reliable ultrasonic sensor.
- The speed can be measured using a more reliable radar based system using the concept of doppler shift.
- Additional equipment can be utilized to provide a 3D trajectory of the delivery.

CHAPTER 11

References

WEBSITE:

[1]Pitch Vision: https://www.pitchvision.com/category/cricket#/

[2]Honeywell: https://sensing.honeywell.com/sensors/force-sensors (Force sensor)

[3]Piezoelectric: https://www.efxkits.us/piezoelectric-sensor-interfacing-arduino/

[4]Pressure mat: https://blog.hackster.io/yoga-mat-size-pressure-sensor-matrix-uses-lattepanda-and-velostat-to-save-money-49ee86950e5a

[5]Processing3: https://processing.org/reference/

[6]Python: https://www.python.org/doc/

[7]Arduino: https://www.arduino.cc/reference/en/

[8]Adafruit: https://www.adafruit.com/?q=velostat

IEEE PAPERS:

[1] Suprapto, Sena & Setiawan, A.W. & Zakaria, H & Adiprawita, W & Supartono, B. (2017). Low-Cost Pressure Sensor Matrix Using Velostat. 137-140. 10.1109/ICICI-BME.2017.8537720.

ANNEXURE

Arduino Code

```
const byte s0 = 26;//selection pins for bigmat columns (O/p section)
const byte s1 = 27;// EF
const byte s2 = 28;//port A
const byte s3 = 29;

const byte bc0 = 45;//o/p section for  small mat
const byte bc1 = 44;//port L
const byte bc2 = 43;
const byte bc3 = 42;

const byte t0 = 53;//big mat i/p section 1
const byte t1 = 52;// port b
const byte t2 = 51;//ip1
const byte t3 = 50;

const byte br0 = 33;//small mat ip section
const byte br1 = 32;
const byte br2 = 31;
const byte br3 = 30;//port C

int mu = 0;
int t = 0;
int star = 0;
float ttemp = 0;
//op variables
int a = 0;
int b = 0;
int c = 0;
int d = 0;
int ca=0,cb=0,cc=0,cd=0;

//mat "op  analog " pin default for arduino mega
#define SIG_pin A7
#define SIG_pin2 A2

// input to mux" pin default for arduino mini pro
const byte OUT_pin1 = 7;//ip pins for big mat 1
const byte OUT_pin3=9;//small mat i/p



byte
p3[16]={B00000000,B00010000,B00100000,B00110000,B01000000,B01010000,B01100000,B01110000,B10
000000,B10010000,B10100000,B10110000,B11000000,B11010000,B11100000,B11110000};
byte
p2[16]={B00000000,B00000001,B00000010,B00000011,B00000100,B00000101,B00000110,B00000111,B00
001000,B00001001,B00001010,B00001011,B00001100,B00001101,B00001110,B00001111};
```

```
int valor = 0;//variable for sending bytes to processing
int valorback=0;
int tt2=0;
//matrix for mat
int calibra[15][12]; //big
int calibra_b[12][9];//back
int maxsensor[15][12];//Calibration array for the min values of each od the 225 sensors.
int maxsensor_b[12][9];
int minsensor_b=1000;
int temp=0;
int temp2=0;
int rw=0;
int cl=0;
int ttemp2=0;
int star2=0;  //Variable for staring the min array
void setup()
{
pinMode(s0, OUTPUT);
pinMode(s1, OUTPUT);
pinMode(s2, OUTPUT);
pinMode(s3, OUTPUT);

pinMode(t0, OUTPUT);
pinMode(t1, OUTPUT);
pinMode(t2, OUTPUT);
pinMode(t3, OUTPUT);

pinMode(bc0, OUTPUT);
pinMode(bc1, OUTPUT);
pinMode(bc2, OUTPUT);
pinMode(bc3, OUTPUT);

pinMode(br0, OUTPUT);
pinMode(br1, OUTPUT);
pinMode(br2, OUTPUT);
pinMode(br3, OUTPUT);

pinMode(OUT_pin1, OUTPUT);
pinMode(OUT_pin3, OUTPUT);

digitalWrite(s0, LOW);
digitalWrite(s1, LOW);
digitalWrite(s2, LOW);
digitalWrite(s3, LOW);

digitalWrite(t0, LOW);
digitalWrite(t1, LOW);
digitalWrite(t2, LOW);
digitalWrite(t3, LOW);

digitalWrite(bc0, LOW);
```

```
digitalWrite(bc1, LOW);
digitalWrite(bc2, LOW);
digitalWrite(bc3, LOW);

digitalWrite(br0, LOW);
digitalWrite(br1, LOW);
digitalWrite(br2, LOW);
digitalWrite(br3, LOW);

digitalWrite(OUT_pin1, HIGH);
digitalWrite(OUT_pin3, HIGH);

Serial.begin(115200);
sbi(ADCSRA, ADPS2);
cbi(ADCSRA, ADPS1);
cbi(ADCSRA, ADPS0);

// Full of 0's of initial matrix
for(byte j = 0; j < 15; j ++)
{

for(byte i = 0; i < 12; i ++)
{calibra[j][i] = 0;
maxsensor[j][i]=0;

}
}
backscreendefine();

for(byte k = 0; k < 50; k++){
for(byte j = 0; j < 15; j ++){

writeMux1(j);
for(byte i = 0; i < 12; i ++)
{ temp=readMux(i);
calibra[j][i] = calibra[j][i]+temp;
if( temp>maxsensor[j][i])
{maxsensor[j][i]=temp;
}
}
}
}

for(byte j = 0; j < 15; j ++)
{
writeMux1(j);
for(byte i = 0; i < 12; i ++){
calibra[j][i] = calibra[j][i]/50;
}
}
backscreen_calib();
```

```
}
void loop()
{
sensing();
}

int readMux(byte channel){
PORTA=p3[channel];
int val = analogRead(SIG_pin);

//return the value
return val;
}
int readMux2(byte channel){
PORTL=p3[channel];
int val = analogRead(SIG_pin2);

//return the value
return val;
}


void writeMux1(byte channel){
PORTB=p2[channel];
}

void writeMux3(byte channel){
PORTC=p3[channel];
}


void backscreendefine()
{

for(byte p = 0; p < 12; p ++)
{
for(byte q = 0; q < 9; q ++)
{
calibra_b[p][q] = 0;
maxsensor_b[p][q]=0;

}
}
}
void backscreen_calib()
{
for(byte k = 0; k < 70; k++){
for(byte p = 0; p < 12; p ++){
writeMux3(p);
for(byte q = 0; q < 9; q ++){
temp2=readMux2(q);
calibra_b[p][q] = calibra_b[p][q]+temp2;
```

```
if( temp2>maxsensor_b[p][q])
{maxsensor_b[p][q]=temp2;
}}}}

for(byte j = 0; j < 12; j ++){
writeMux3(j);
for(byte i = 0; i < 9; i ++){
calibra_b[j][i] = calibra_b[j][i]/70;
}}}
```

Speed:

```
#define trigPin 9
#define echoPin 10
float duration,distancem1,distancem2,speed0,distm,sum=0,i=0;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(13,OUTPUT);
  pinMode(7,OUTPUT);

}
  void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distancem1= duration*0.034/200;
//Serial.println(distancem1);
delay(150);
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distancem2= duration*0.034/200;
distm=distancem1-distancem2;
speed0=distm/0.15;
speed0=speed0*3.6;
if (distancem1<=300)  {
    if(distancem1>distancem2){
        sum=sum+speed0;
        i=i+1;
        }}
if(i==2)
{
  if(sum>0){
Serial.print("Speed of delivery is") ;
Serial.print(sum/2);
Serial.println();

i=0;
sum=0;
}
  }
```

Processing3 Code

```
import processing.serial.*;
 Serial myPort;  //the Serial port object
 String val;
 int[] input = new int [4];
 //int[] inputr = new int [2];
 int serialCount=0;
 int tiempoant;
boolean firstContact = false;
String sensorReading="";
import java.io.BufferedWriter;
import java.io.FileWriter;
String outFilename = "data.csv";
void setup() {
  size(960, 960,P3D); //make our canvas 200 x 200 pixels big

  myPort = new Serial(this, Serial.list()[0], 115200);
  appendTextToFile(outFilename,"Date,Time,Length_x,Length_y,Line_x,Line_y\n");
  }
void draw()  {
textSize(30);

fill(0,0,255);

    translate(800,800);
    rotateX(radians(60));
    background(0);


    for (int j=0; j<15; j++) {
      for (int i=0; i<12; i++) {
          stroke(255);
  if (j>=12 && j<=15){
  fill(255,255,51);
  rect(-60*i,-120*j,60,120);
  }
  else if(j>=6 && j<12){
    fill(0,255,0);
  rect(-60*i,-120*j,60,120);
  }
  else if(j>=3 && j<6){
    fill(255,0,0);
  rect(-60*i,-120*j,60,120);
  }
  else if(j<3){
    fill(100,149,237);
  rect(-60*i,-120*j,60,120);
  }
        }

      }
```

```
   int  a=input[1];
   int  b=input[0];
 fill(0,0,0);
  rect(-60*a,-120*b,60,120);
    pushMatrix();
    translate(-242,0,535);
     rotateX(radians(-60));
     scale(0.52);
       for (int i=0; i<12; i++) {
     for (int j=0; j<12; j++) {
         stroke(255);
       if (i>=8 && i<=11){
  fill(255,128,0);
   rect(-i*28,-j*28,28,28);
  }
  else if(i>=4 && i<8){
    fill(128,128,128);
  rect(-i*28,-j*28,28,28);
    if(i>=5 && i<=6 && j<4)
    {
      fill(204,204,0);
  rect(-i*28,-j*28,28,28);
    }
  }
  else if(i>=0 && i<4){
    fill(153,0,73);
  rect(-i*28,-j*28,28,28);
  }
   }
  }
    int  c=11-input[3];
    int  d=11-input[2];
  fill(0,0,0);
  rect(-28*c,-28*d,28,28);
  fill(255,255,51);
     text("Yorker", 40,60);
fill(0,255,0);
text("Full", 125, 200);
fill(255,0,0);
text("Good", 215, 400);
fill(100,149,237);
text("Short", 330, 650);
tint(255, 200);
//image(img, -700, -475);

     popMatrix();
}
  void serialEvent(Serial myPort) {sensorReading = myPort.readStringUntil('\n');
  if(sensorReading != null){
    int d=day();
    int m=month();
    int y=year();
```

```
  String dmy=str(d)+"-"+str(m)+"-"+str(y);
  int h=hour();
  int mi=minute();
  int sec=second();
  String time=str(h)+":"+str(mi)+":"+str(sec);
  sensorReading=trim(sensorReading);
  int abc[] = int(split(sensorReading, ','));
 appendTextToFile(outFilename,dmy+","+time+","+sensorReading);
 input[0]=abc[0];
 input[1]=abc[1];
 input[2]=abc[2];
 input[3]=abc[3];
 println(abc);

void appendTextToFile(String filename, String text){
 File f = new File(dataPath(filename));
 if(!f.exists()){
  createFile(f);
 }
 try {
  PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(f, true)));   // true to append the output
  out.println(text);
  out.close();                                  // clean close allows to read / copy file while processing runs
 }catch (IOException e){
    e.printStackTrace();
 }
}
void createFile(File f){
 File parentDir = f.getParentFile();
 try{
  parentDir.mkdirs();
  f.createNewFile();
   }
  catch(Exception e){
  e.printStackTrace();
  }
}
```

GUI Code in Python3

```
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QDialog, QApplication, QPushButton, QVBoxLayout, QLabel, QWidget
import csv
from PyQt5.QtGui import QPixmap
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.backends.backend_qt5agg import NavigationToolbar2QT as NavigationToolbar
import matplotlib.pyplot as plt

class Ui_MainWindow(object):
    def abc(self):
        self.window = QtWidgets.QMainWindow()
        self.ui = Ui_Dialog()
        self.ui.setupUi1(self.window)
        MainWindow.hide()
        self.window.show()
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1120, 860)
        MainWindow.setStyleSheet("background-color: rgb(255, 170, 0);")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(260, 90, 551, 61))
        font = QtGui.QFont()
        font.setFamily("Imprint MT Shadow")
        font.setPointSize(30)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(620, 200, 211, 31))
        font = QtGui.QFont()
        font.setFamily("Goudy Old Style")
        font.setPointSize(20)
        font.setItalic(True)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(430, 390, 261, 81))
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(16)
        self.pushButton.setFont(font)
        self.pushButton.setMouseTracking(True)
        self.pushButton.setAutoFillBackground(False)
        self.pushButton.setStyleSheet("background-color: rgb(85, 170, 0);")
        self.pushButton.setObjectName("pushButton")


    self.pushButton.clicked.connect(self.abc)
```

```
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1124, 26))
        self.menubar.setObjectName("menubar")
        self.menuStart_Page = QtWidgets.QMenu(self.menubar)
        self.menuStart_Page.setObjectName("menuStart_Page")
        self.menuStats = QtWidgets.QMenu(self.menubar)
        self.menuStats.setObjectName("menuStats")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionLine = QtWidgets.QAction(MainWindow)
        self.actionLine.setObjectName("actionLine")
        self.actionLength = QtWidgets.QAction(MainWindow)
        self.actionLength.setObjectName("actionLength")
        self.menuStats.addAction(self.actionLine)
        self.menuStats.addAction(self.actionLength)
        self.menubar.addAction(self.menuStart_Page.menuAction())
        self.menubar.addAction(self.menuStats.menuAction())

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label.setText(_translate("MainWindow", "Cricket Bowling Analyser"))
        self.label_2.setText(_translate("MainWindow", "-By Team Ultraa"))
        self.pushButton.setText(_translate("MainWindow", "Start!"))
        self.menuStart_Page.setTitle(_translate("MainWindow", "Start Page"))
        self.menuStats.setTitle(_translate("MainWindow", "Stats"))
        self.actionLine.setText(_translate("MainWindow", "Line"))
        self.actionLength.setText(_translate("MainWindow", "Length"))

##Stats page##
class Ui_Dialog(object):
    def abd(self):
        lst_leng=[]
        with open("people.csv", 'r') as csvfile:
            reader = csv.DictReader(csvfile)
            for row in reader:
                data=dict(row)
                lst_leng.append([data['lengthx'],data['lengthy']])

        csvfile.close()
        print(lst_leng)

        length=[]
        for lst in lst_leng:
            lst=int(lst[0])
```

```
            if(lst<=2):
                length.append('Yoker')
            elif(lst<=6):
                length.append('Full')
            elif(lst<=10):
                length.append('Good')
            else:
                length.append('Short')
        print(length)

        over=list(range(1,7))
        print(over)
        plt.figure(1, figsize=(18,18), dpi=250)
        plt.scatter(over,length,s=150,c='r')
        plt.xlabel("Balls",fontsize=13)
        plt.ylabel('Length of delivery',fontsize=13)
        plt.savefig('test1.jpg')

        plt.show()
        print(5)
       # self.window = QtWidgets.QDialog()
       # self.ui = Ui_Dialoge()
       # self.ui.setupUi2(self.window)
       # Dialog.hide()
       # self.window.show()
    def abline(self):

        lst_line=[]
        with open("people.csv", 'r') as csvfile:
            reader = csv.DictReader(csvfile)
            for row in reader:
                data=dict(row)

                lst_line.append([data['linex'],data['liney']])
        csvfile.close()

        line=[]


        for lst in lst_line:
            lst=int(lst[1])
            if(lst<=3):
                line.append('Off side')
            elif(lst<=6):
                line.append('Middle')
            else:
                line.append('LEG side')
        print(line)
        over=list(range(1,7))
        print(over)

        plt.figure(2, figsize=(18, 18), dpi=250)
```

```
        plt.scatter(over,line,s=150)
        plt.xlabel("Balls")
        plt.ylabel('Line of delivery')
        plt.savefig('test2.jpg')
        plt.show()
        print(5)
    #def abx(self):
      # self.window = QtWidgets.QDialog()
      # self.ui = Ui_Dialoge()
      # self.ui.setupUi2(self.window)
      # Dialog.hide()
      # self.window.show()
    def setupUi1(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(1120, 860)
        Dialog.setStyleSheet("background-color: rgb(255, 170, 0);")
        self.label = QtWidgets.QLabel(Dialog)
        self.label.setGeometry(QtCore.QRect(350, 90, 411, 101))
        font = QtGui.QFont()
        font.setFamily("Centaur")
        font.setPointSize(30)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.pushButton = QtWidgets.QPushButton(Dialog)
        self.pushButton.setGeometry(QtCore.QRect(400, 300, 291, 71))
        font = QtGui.QFont()
        font.setFamily("Modern No. 20")
        font.setPointSize(16)
        self.pushButton.setFont(font)
        self.pushButton.setStyleSheet("background-color: rgb(85, 170, 0);")
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(Dialog)
        self.pushButton_2.setGeometry(QtCore.QRect(400, 470, 291, 71))
        font = QtGui.QFont()
        font.setFamily("Modern No. 20")
        font.setPointSize(16)
        self.pushButton_2.setFont(font)
        self.pushButton_2.setStyleSheet("background-color: rgb(85, 170, 0);")
        self.pushButton_2.setObjectName("pushButton_2")

        self.pushButton.clicked.connect(self.abline)
        self.pushButton_2.clicked.connect(self.abd)

        self.retranslateUi1(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi1(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
        self.label.setText(_translate("Dialog", "Select Your Preference"))
        self.pushButton.setText(_translate("Dialog", "Line"))
        self.pushButton_2.setText(_translate("Dialog", "Length"))
```

```
        ##graph page##
class Ui_Dialoge(object):
    def abe(self):
        self.window = QtWidgets.QMainWindow()
        self.ui = Ui_Dialog()
        self.ui.setupUi1(self.window)
        Dialoge.hide()
        self.window.show()
    def setupUi2(self, Dialoge):
        Dialoge.setObjectName("Dialoge")
        Dialoge.resize(1120, 860)
        Dialoge.setStyleSheet("background-color: rgb(255, 170, 0);")
        self.pushButton = QtWidgets.QPushButton(Dialoge)
        self.pushButton.setGeometry(QtCore.QRect(490, 810, 93, 28))
        self.pushButton.setStyleSheet("background-color: rgb(85, 170, 0);")
        self.pushButton.setObjectName("pushButton")

        self.pushButton.clicked.connect(self.abe
        self.retranslateUi2(Dialoge)
        QtCore.QMetaObject.connectSlotsByName(Dialoge)
    def retranslateUi2(self, Dialoge):
        _translate = QtCore.QCoreApplication.translate
        Dialoge.setWindowTitle(_translate("Dialoge", "Dialoge"))
        self.pushButton.setText(_translate("Dialoge", "BACK"))
        #image#
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    Dialog = QtWidgets.QMainWindow()
    Dialoge = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_()
```