Final Report

GRAND CARE : Smart walking stick for Elderly (With alert system and health monitoring)

Team Members :

- Prakash nidhi verma 2016185 CSE
- Satyam dubey 2016232 ECE
- Samyak jain 2016224 CSE
- Raunak patel 2016203 ME
- Anshuman mishra 2016036 ECE
- Anusha Tharamal 2016508 Design

Electronics and Communication

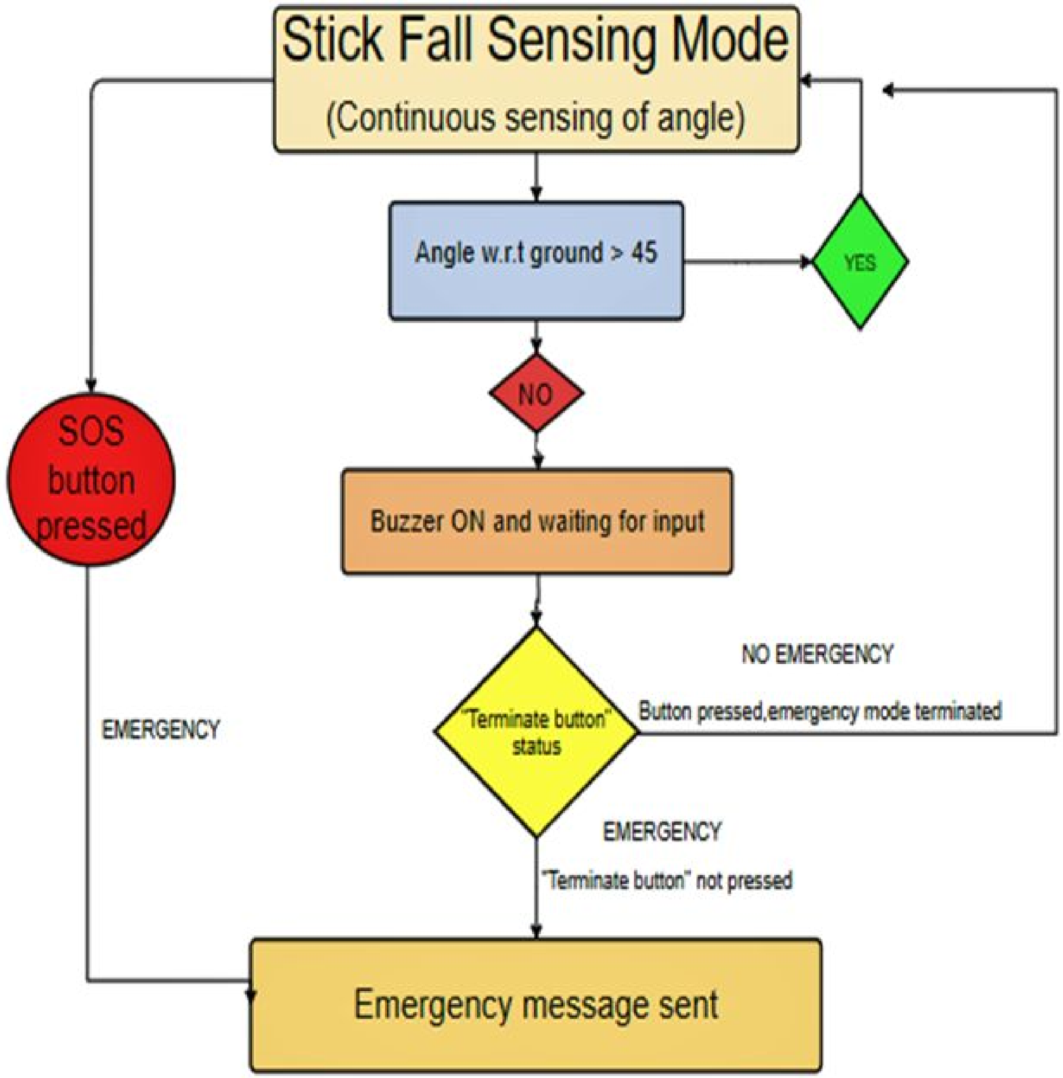
Members working :

- Satyam Dubey 2016232 ECE
- Anshuman Mishra 2016036 ECE

Original inside snap of the stick loaded with all sensors without cover

CIRCUIT 1:

Arduino interfaced GSM module (900A) and Gyroscope (MPU6050) also including a buzzer and a push button.



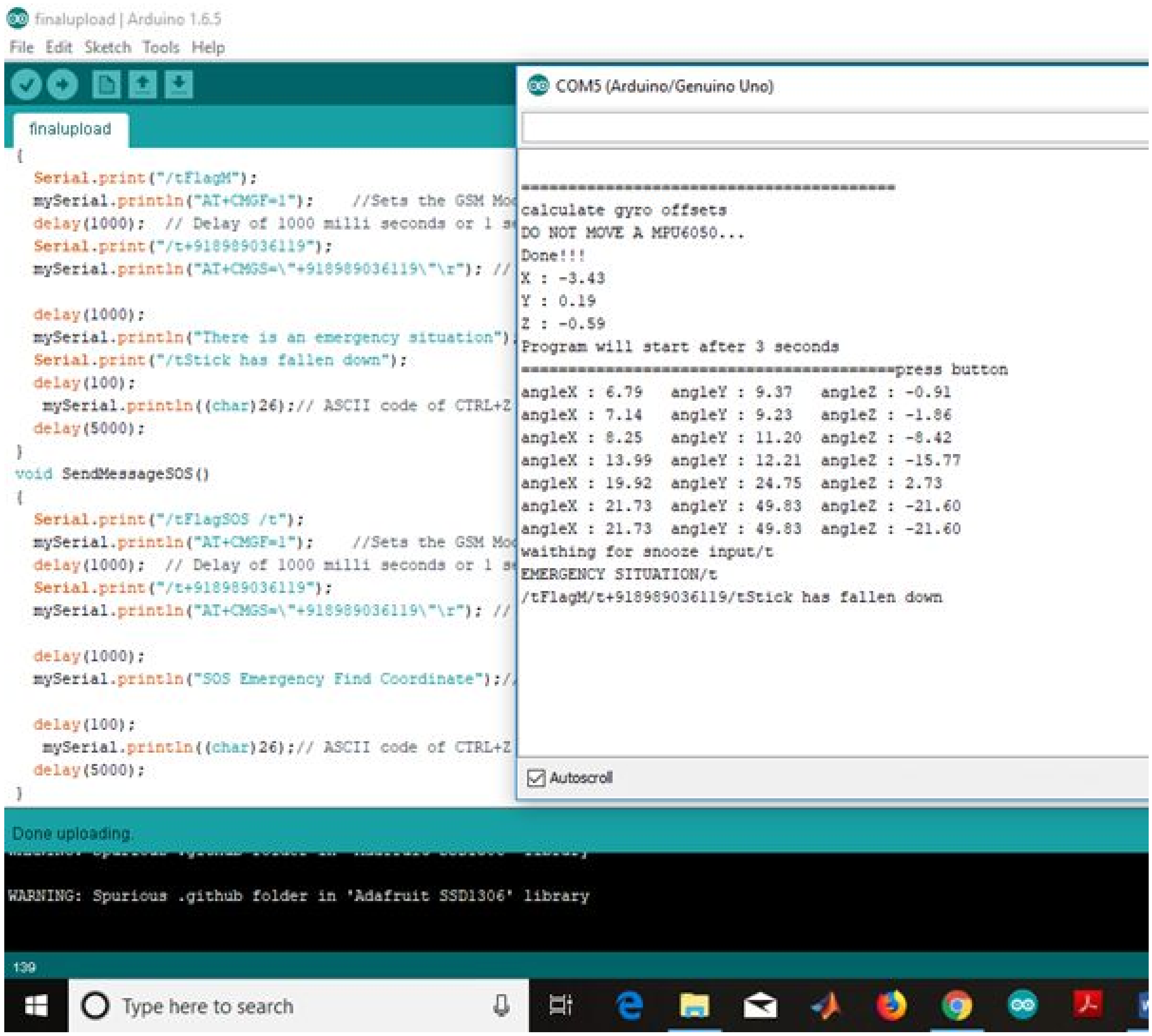
Flowchart to explain functioning of above components

DESCRIPTION:

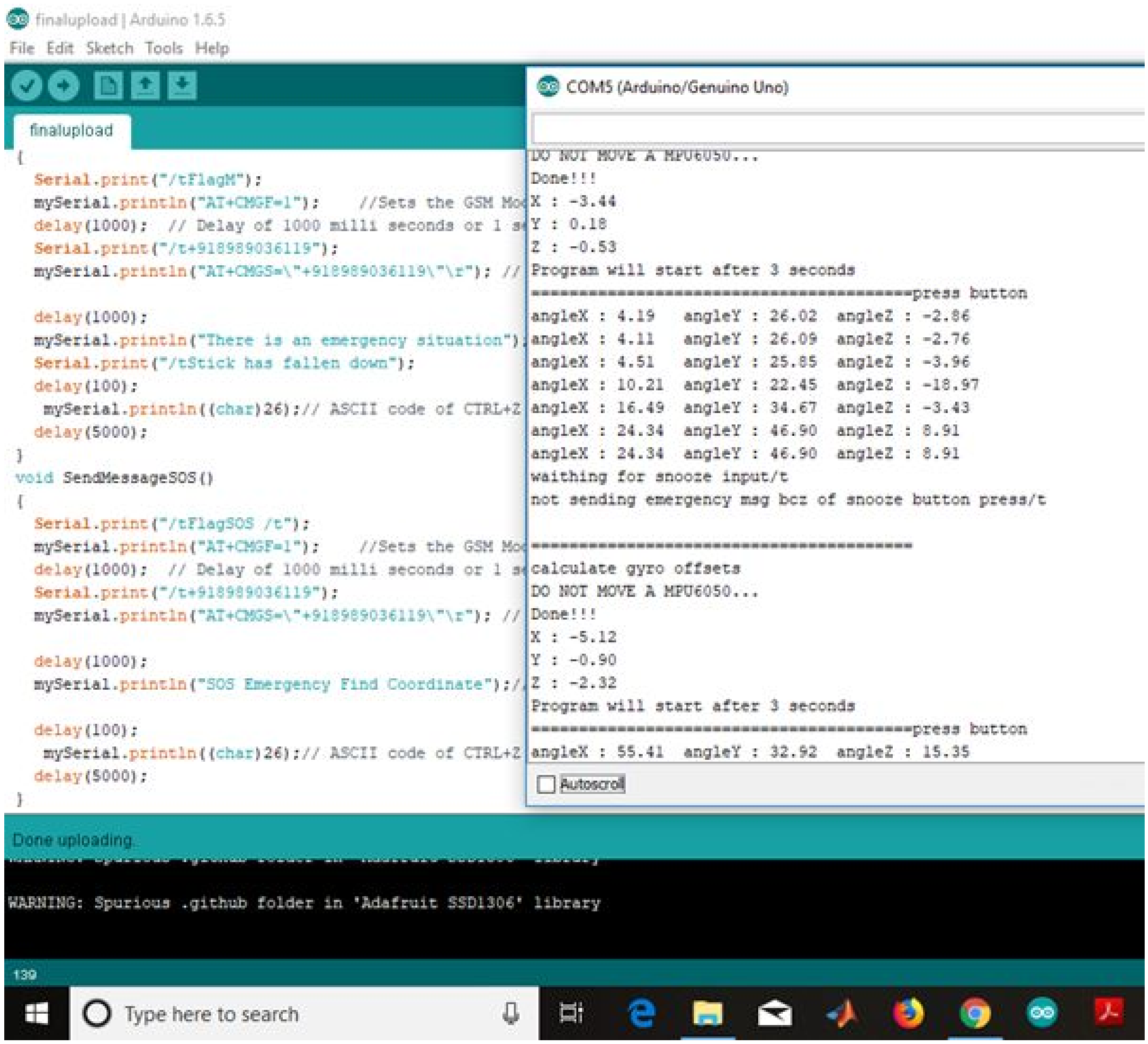
- Gyroscope placed on the stick is continuously measuring the angle of the stick with respect to horizontal.
- If the stick falls, the angle will be detected by the gyroscope which will turn ON the buzzer for 15 seconds, also this duration is used as a waiting time for an input from user to terminate the emergency mode in case of a false emergency situation.
- If the button is pressed the situation is sensed as normal and there is no emergency message and the system goes in angle checking mode.
- If there is no input in the given time window there is an emergency situation. Hence an emergency message is sent with the help of GSM module.
- Also, there is a SOS button placed which if pressed for more than 5 seconds gives an emergency message.

Snapshot of Arduino simulation of all the condition

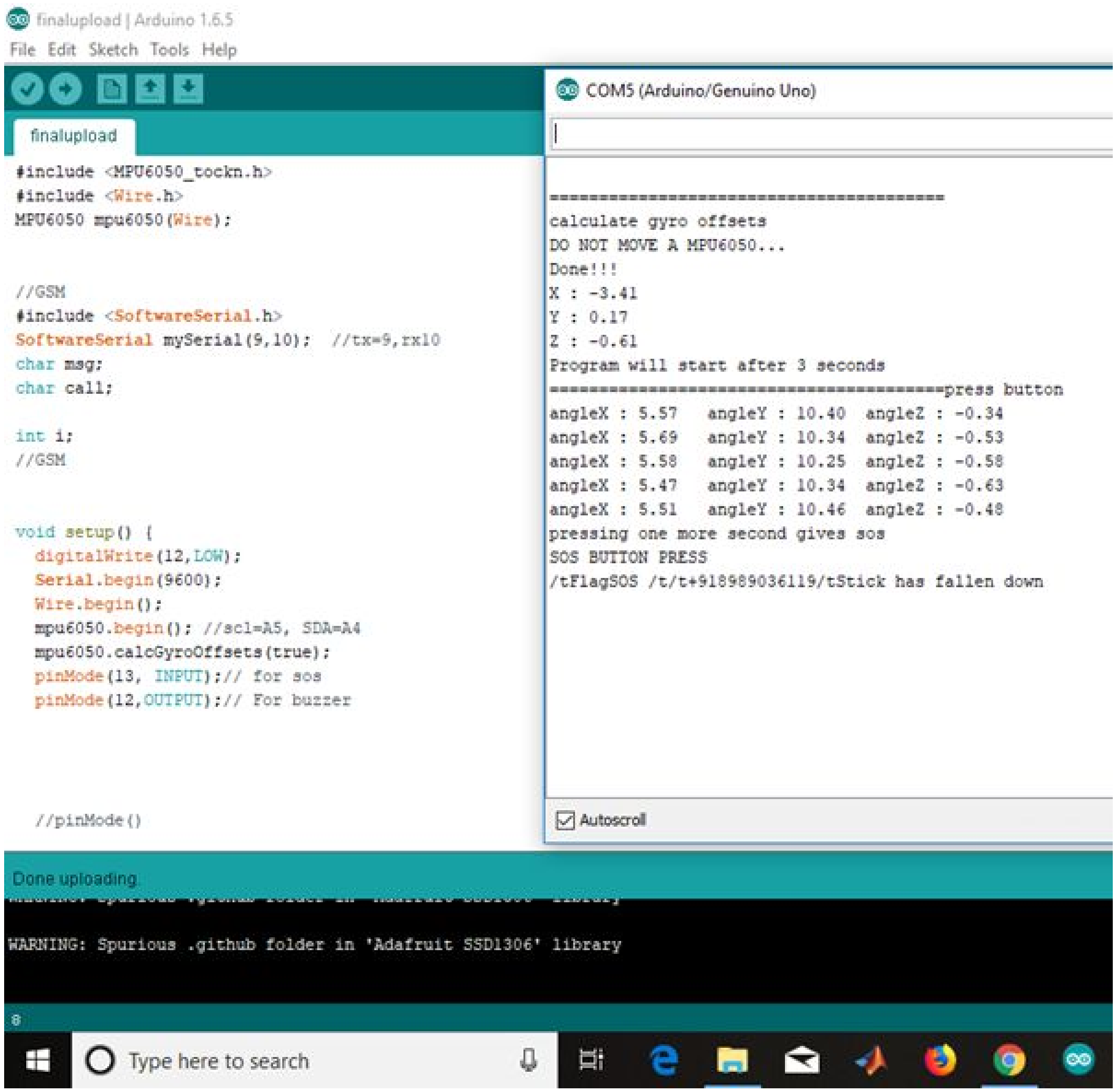
Case 1. Stick falls and results in alert sms



Case 2. False emergency situation and user terminates the emergency mode and back to normal angle tracking



Case3. User wants help and hits the sos button results in sos message alert



Components Used:

1. Arduino UNO:

Arduino Uno is a microcontroller board based on the ATmega328 chip, in which we give some input to get a desired output. It has 14 digital input/output pins, 6 analog inputs, a power jack, a USB connection, an ICSP header, a 16 MHz ceramic resonator and a reset button. It has got everything needed to make a microcontroller work; you just have to connect it to a computer with a USB cable or power it with a battery to start operating it.

Arduino Uno

2. Gyroscope MPU6050

The MPU 6050 is a 6 DOF (degrees of freedom) or a six-axis IMU sensor, which means that it gives six values as output: three values from the accelerometer and three from the gyroscope. The MPU 6050 is a sensor based on MEMS (micro electro mechanical systems) technology.
We have interfaced Arduino with gyroscope to measure angle with respect to horizontal.

MPU6050

3. GSM 900A

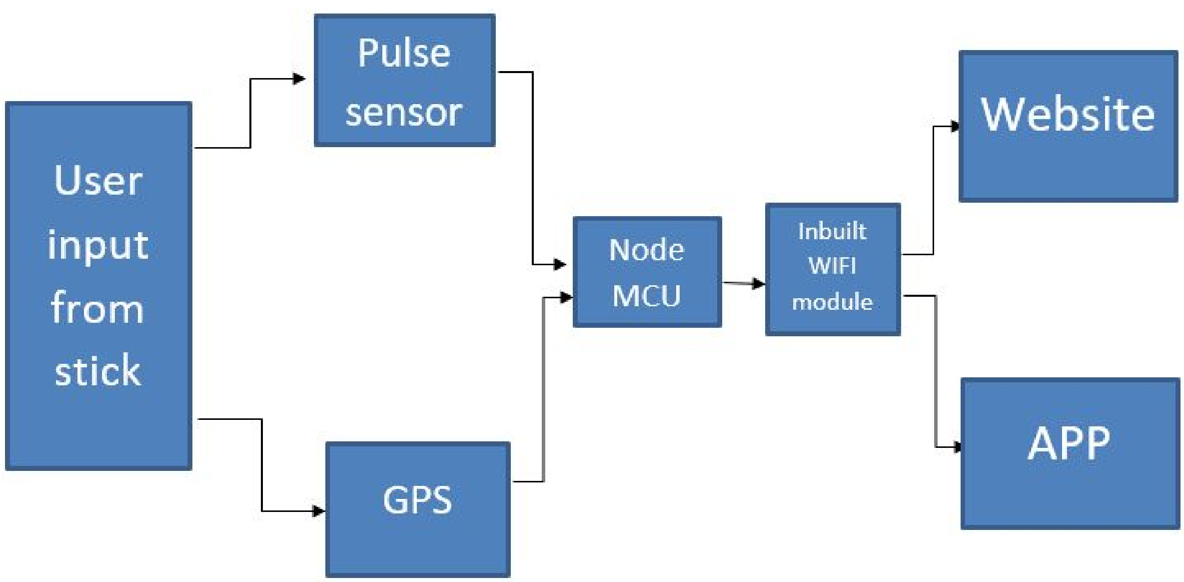
GSM/GPRS Modem is built with Dual Band GSM/GPRS engine- SIM900A, works on frequencies 900/ 1800 MHz The Modem is coming with RS232 interface, which allows you connect PC as well as microcontroller with RS232 Chip (MAX232).The GSM/GPRS Modem is having internal TCP/IP stack to enable you to connect with internet via GPRS. It is suitable for SMS, Voice as well as DATA transfer application in M2M interface. We have used this modem for sending message in case of emergency.

4. Buzzer and Push Button

Here buzzer is used to help user understand different modes of the stick. The push button is used to trigger the SOS mode of the stick also pressing at appropriate time terminates the emergency mode.

CIRCUIT 2:

Components: Node MCU, GPS, Pulse sensor



Flow chart

*Link to access the website:

https://pnidhi26.github.io/SmartCane/

Description:

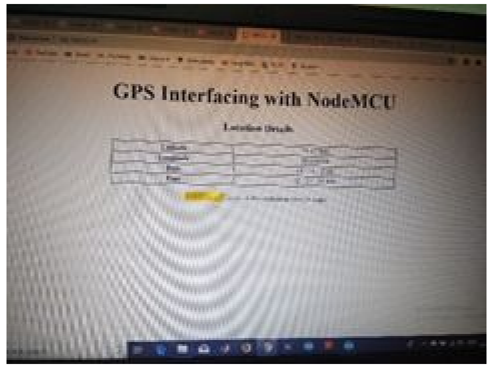
1. With the help of Node MCU microcontroller interfaced with GPS module the live location is tracked. Inbuilt wifi module uploads real time location of the user to the website (https://pnidhi26.github.io/SmartCane/)
2. GPS module is also being used to calculate real time distance from home, walking speed and direction.
3. Pulse sensor with the help of NodeMcu uploads real time data to website which can be monitored and stored for long time health analysis.

Components used:

1. NodeMcu:

The NodeMcu (Node Microcontroller Unit) is an open source software and hardware development environment that is built around a very inexpensive System-on-a-Chip (SoC) called the ESP8266. It contains all crucial elements of the modern computer: CPU, RAM, networking (Wi-Fi), and even a modern operating system and SDK.

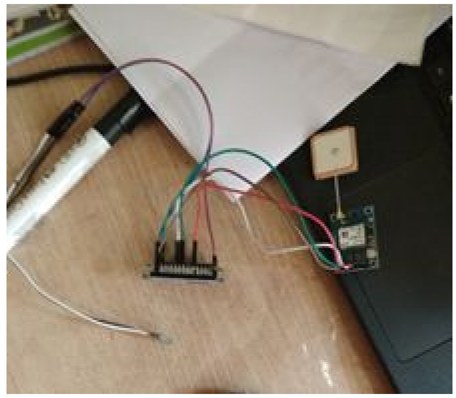
We have used to process the data input from sensor at the same time uploading it to website.



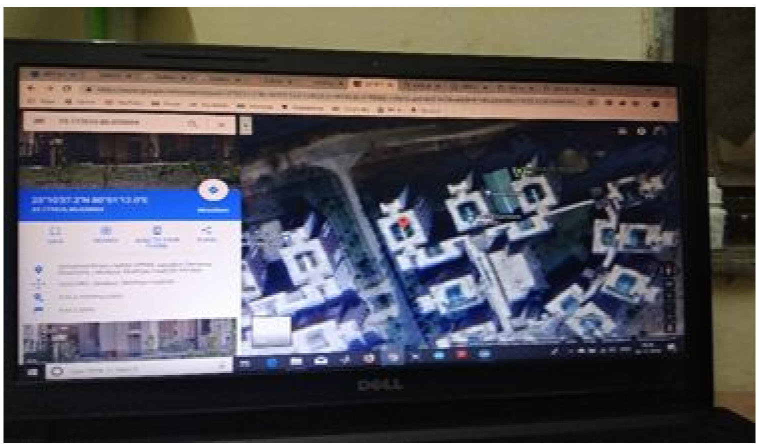
*actual image

2. GPS module

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high-performance u-blox 6 positioning engine. These flexible and cost-effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints. The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly.
We have interface GPS with microcontroller to track live location, distance from a fixed point and speed.



*actual image



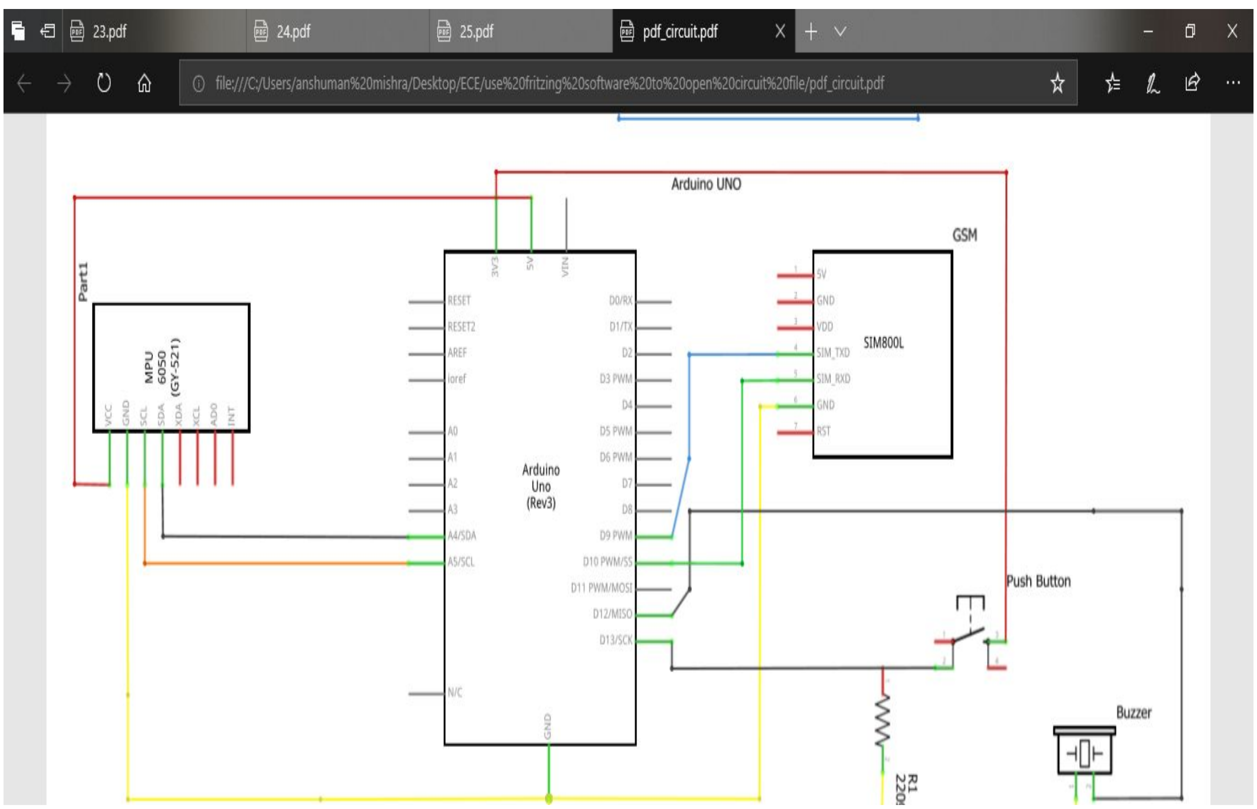
*actual image

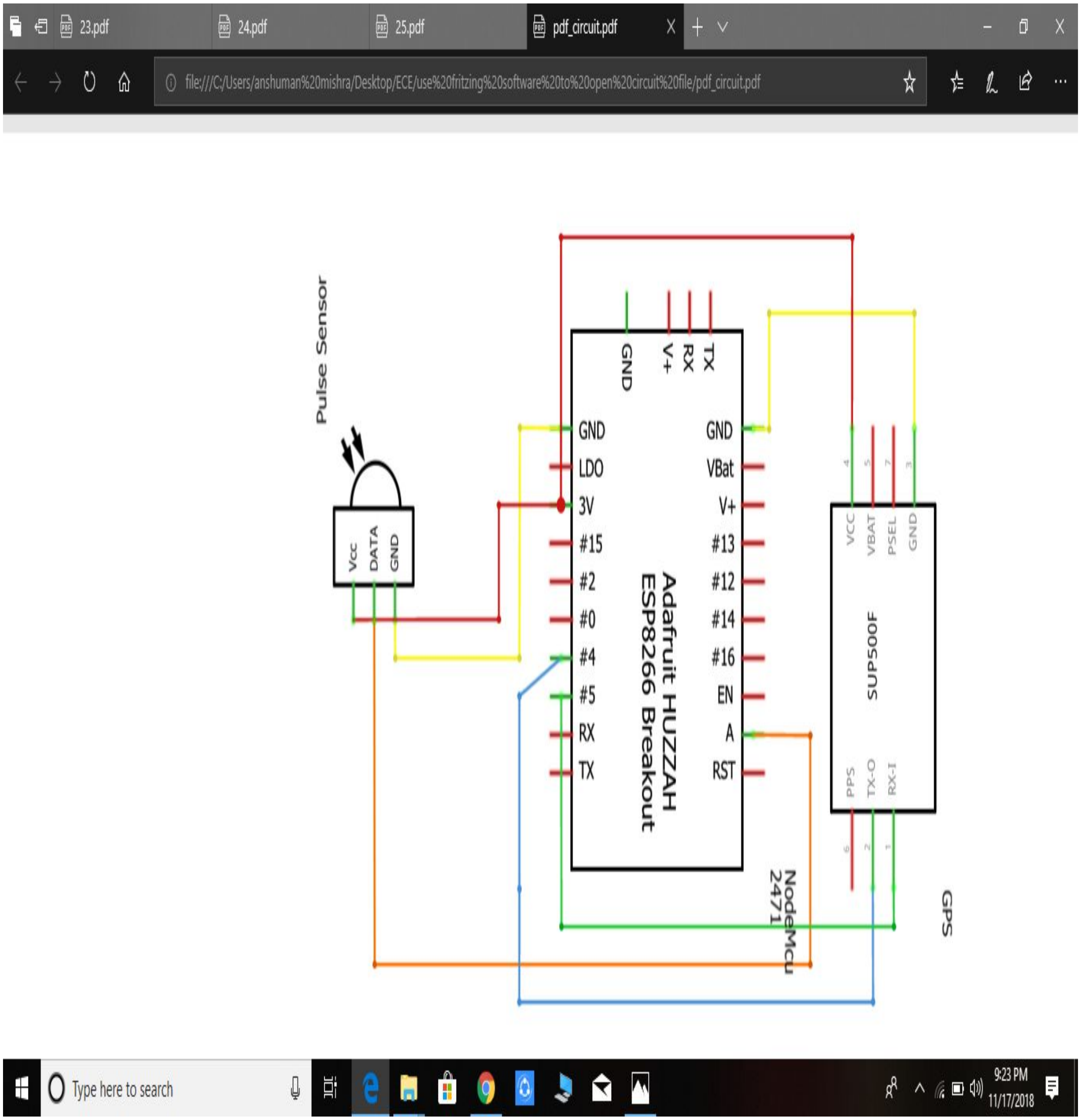
3. Pulse sensor

Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heartrate data into their projects. The sensor clips onto a fingertip or earlobe and plugs right into Arduino with some jumper cables. It also includes an open-source monitoring app that graphs your pulse in real time.

*actual image

Screenshot of circuit diagram is shown below made by fritzing software




23.pdf
24.pdf
25.pdf
pdf_circuit.pdf
file:///C:/Users/anshuman%20mishra/Desktop/ECE/use%20fritzing%20software%20to%20open%20circuit%20file/pdf_circuit.pdf
Pulse Sensor
Vcc
DATA
GND
Adafruit HUZZAH
ESP8266 Breakout
GND
V+
RX
TX
GND
LDO
3V
#15
#2
#0
#4
#5
RX
TX
GND
VBat
V+
#13
#12
#14
#16
EN
A
RST
NodeMcu
2471
SUP500F
VCC
VBAT
PSEL
GND
PPS
TX-O
RX-I
GPS
Type here to search
9:23 PM
11/17/2018

Ece Code1

```
#include <MPU6050_tockn.h>
#include <Wire.h>
MPU6050 mpu6050(Wire);
//GSM
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9,10);  //tx=9,rx10
char msg;
char call;
int i;
//GSM

void setup() {
  digitalWrite(12,LOW);
  Serial.begin(9600);
  Wire.begin();
  mpu6050.begin(); //scl=A5, SDA=A4
  mpu6050.calcGyroOffsets(true);
  pinMode(13, INPUT);// for sos
  pinMode(12,OUTPUT);// For buzzer


 //pinMode()

  //GSM
   mySerial.begin(9600);   // Setting the baud rate of GSM Module
  Serial.begin(9600);// Setting the baud rate of Serial Monitor (Arduino)
  Serial.println("press button");
  //GSM

}
```

```
void loop() {
  digitalWrite(12,LOW);
  mpu6050.update();
  Serial.println("reinitiated");

  Serial.print("angleX : ");
  Serial.print(mpu6050.getAngleX());
  Serial.print("\tangleY : ");
  Serial.print(mpu6050.getAngleY());
  Serial.print("\tangleZ : ");
  Serial.println(mpu6050.getAngleZ());

  delay(1000);

  if((abs(mpu6050.getAngleX())>45 ||abs(mpu6050.getAngleY()>45))  )
  {
    delay(1000);
    Serial.print("angleX : ");
  Serial.print(mpu6050.getAngleX());
  Serial.print("\tangleY : ");
  Serial.print(mpu6050.getAngleY());
  Serial.print("\tangleZ : ");
  Serial.println(mpu6050.getAngleZ());
      if((abs(mpu6050.getAngleX())>45 ||abs(mpu6050.getAngleY()>45))  )
         {Serial.println("waithing for snooze input/t");
          for(i=1;i< 10;i++)       //  ten sec wait for snooze input
          {
            digitalWrite(12,HIGH);
           delay(1000);
              if(digitalRead(13) == HIGH)
              {
```

```
            digitalWrite(12,LOW);
            Serial.println("not sending emergency msg bcz of snooze button press/t");
           setup();

            loop();
          }

        }
        digitalWrite(12,LOW);
       }

                    if((abs(mpu6050.getAngleX())>45 ||abs(mpu6050.getAngleY()>45)) )
                       {
                         digitalWrite(12,LOW);
                          Serial.println("EMERGENCY SITUATION/t");
                          SendMessage();

                          setup();
                          digitalWrite(12,LOW);
                           loop();

                        }

   }
    if(digitalRead(13) == HIGH)
  {
    delay(1000);
    Serial.println("pressing one more second gives sos");
      if(digitalRead(13))
        delay(1000);
                      if(digitalRead(13) == HIGH)
                        {
                          Serial.println("SOS BUTTON PRESS");
```

```
                         SendMessageSOS();
                          setup();
                          loop();
                        }

    }
}



//GSM
void SendMessage()
{
  Serial.print("/tFlagM");
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  Serial.print("/t+918989036119");
  mySerial.println("AT+CMGS=\"+918989036119\"\r");

  delay(1000);
  mySerial.println("There is an emergency situation");
  Serial.print("/tStick has fallen down");
  delay(100);
   mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(5000);
}
void SendMessageSOS()
{
  Serial.print("/tFlagSOS /t");
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  Serial.print("/t+918989036119");
  mySerial.println("AT+CMGS=\"+918989036119\"\r"); //destination number
  delay(1000);
```

```
  mySerial.println("SOS Emergency Find Coordinate");
  Serial.print("/tStick has fallen down");
  delay(100);
   mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(5000);
}
```

Ece code2

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>

/*
   pins 4(rx) and 3(tx).
*/
static const int RXPin = 12, TXPin = 13;// tx=D6 Rx=D7
static const uint32_t GPSBaud = 9600;
const double GADERI_LAT = 23.1658;
const double GADERI_LNG = 80.0528;
int N = 530;

const char* ssid     = "sat";
const char* password = "86868686";

unsigned long myChannelNumber = 612794;
const char * myWriteAPIKey = "A11PG8H4R9IZZLUI";

// The TinyGPS++ object
TinyGPSPlus gps;
WiFiClient  client;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

void setup()
{
  Serial.begin(115200);
```

```
  ss.begin(GPSBaud);
  Serial.println(F("DeviceExample.ino"));
  Serial.println(F(" TinyGPS++ with an attached GPS module"));
  Serial.print(F("Testing TinyGPS++ library v. "));
Serial.println(TinyGPSPlus::libraryVersion());
  Serial.println();

  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
   Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("Netmask: ");
  Serial.println(WiFi.subnetMask());
  Serial.print("Gateway: ");
  Serial.println(WiFi.gatewayIP());
  ThingSpeak.begin(client);

}

void loop()
{
  // This sketch displays information every time a new sentence is correctly encoded.
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
      displayInfo();

  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while(true);
  }
}
```

```
void displayInfo()
{
 // Serial.print(F("Location: "));
  if (gps.location.isValid())
  {

    double latitude = (gps.location.lat());
    double longitude = (gps.location.lng());
    double speed1 = (gps.speed.kmph());
    double heart = analogRead(A0);
    if(heart < N){
      heart = heart * 0;
    }
    else
    {
      heart = heart * 0.078;
    }
    double distanceKm =
  gps.distanceBetween(
    gps.location.lat(),
    gps.location.lng(),
    GADERI_LAT,
    GADERI_LNG) / 1000.0;

    String latbuf;
    latbuf += (String(latitude, 6));
    Serial.println(latbuf);

    String lonbuf;
    lonbuf += (String(longitude, 6));
    Serial.println(lonbuf);

    String spdbuf;
    spdbuf += (String(speed1, 6));
    Serial.println(spdbuf);

    String hrtbuf;
    hrtbuf += (String(heart, 6));
```

```
  Serial.println(hrtbuf);

  String dstbuf;
  dstbuf += (String(distanceKm, 3));
  Serial.println(dstbuf);
// if(dstbuf > 3.5){
//         buzzer = high
//         delay(1000);

  ThingSpeak.setField(1, latbuf);
  ThingSpeak.setField(2, lonbuf);
  ThingSpeak.setField(3, spdbuf);
  ThingSpeak.setField(4, hrtbuf);
  ThingSpeak.setField(5, dstbuf);
  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  delay(20000);

}
else
{
  Serial.print(F("INVALID"));
}

Serial.print(F("  Date/Time: "));
if (gps.date.isValid())
{
  Serial.print(gps.date.month());
  Serial.print(F("/"));
  Serial.print(gps.date.day());
  Serial.print(F("/"));
  Serial.print(gps.date.year());
}
else
{
  Serial.print(F("INVALID"));
}

Serial.print(F(" "));
if (gps.time.isValid())
```

```
  {
    if (gps.time.hour() < 10) Serial.print(F("0"));
    Serial.print(gps.time.hour());
    Serial.print(F(":"));
    if (gps.time.minute() < 10) Serial.print(F("0"));
    Serial.print(gps.time.minute());
    Serial.print(F(":"));
    if (gps.time.second() < 10) Serial.print(F("0"));
    Serial.print(gps.time.second());
    Serial.print(F("."));
    if (gps.time.centisecond() < 10) Serial.print(F("0"));
    Serial.print(gps.time.centisecond());
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.println();
}
```

Computer Science

Members working :

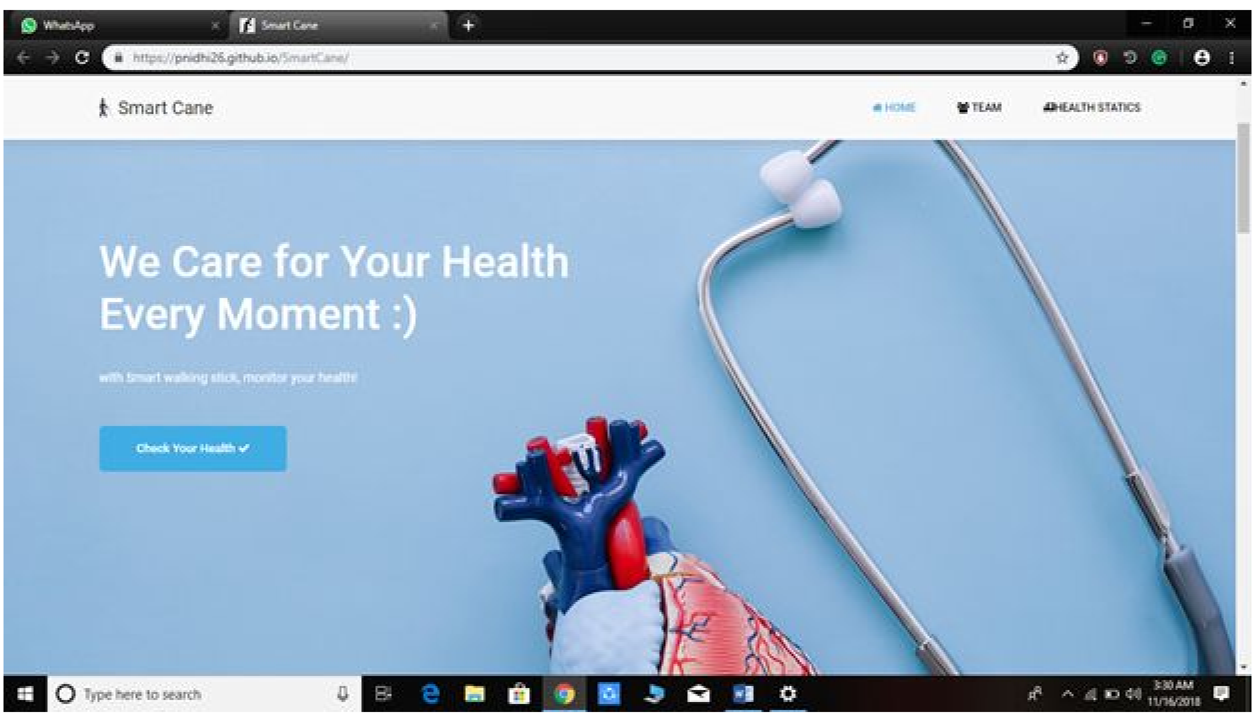
- Samyak Jain 2016224 CSE
- Prakash nidhi verma 2016185 CSE

Description:

Our project aims at easing the daily life of elderly people across the nation. Also, providing medication has became a dreamy requirement these days. As we know that providing medication these days is too costly, thus providing the same at a low cost and on daily basis solves the problem to a greater extent. Thus, understanding the problem we aim to provide a solution. Our smart cane, embedded with several kinds of functionalities provides the user a way to keep track of his/her health. But, the point stands when we ask about how the user will see his/her progress. Here comes the interfacing we are providing. An app and a website made to keep the life seamless and easy for the user.

Both the interfaces can be accessed anytime, from anywhere and at zero cost. Also, the UI of both the type of interfaces has been kept simple and easy to use understanding that our major target of users is the people with age > 60. Though, simple and sober, the UI has all the major functionalities one wish from it.

App Interface (Fig. 1)



Website Interface (Fig.2)

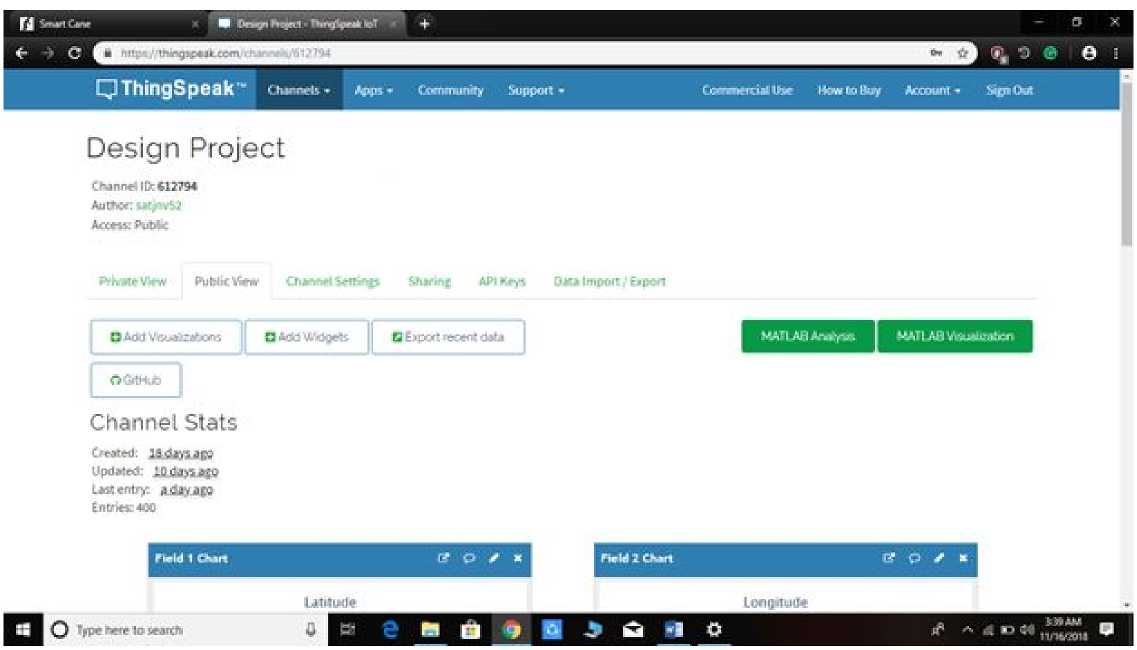
Below is the description of the interfaces:

1> [Use of Thingspeak.io API:]

The data produced from various sensors embedded in our product to provide user with different kinds of functionalities like measuring pulse, providing the exact location along with latitudes and longitudes, speed, distance from the source etc. is provided to the thingspeak API which keeps track of the data. Also, the data is changed dynamically so the user can have the most précised experience.

The data from thingspeak IO is fetched onto our website and mobile application using channel ID, READ API keys, and a simple line of code (given below).

```
iframe width="450" height="260" style="border: 1px solid #cccccc;" src="https://thingspeak.com/channels/Your Channel ID /charts/1?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=50&type=line&update=15">
```



Thingspeak IO API (Fig. 3)

We are given with a specific set of Channel ID and READ API keys. (Shown below)

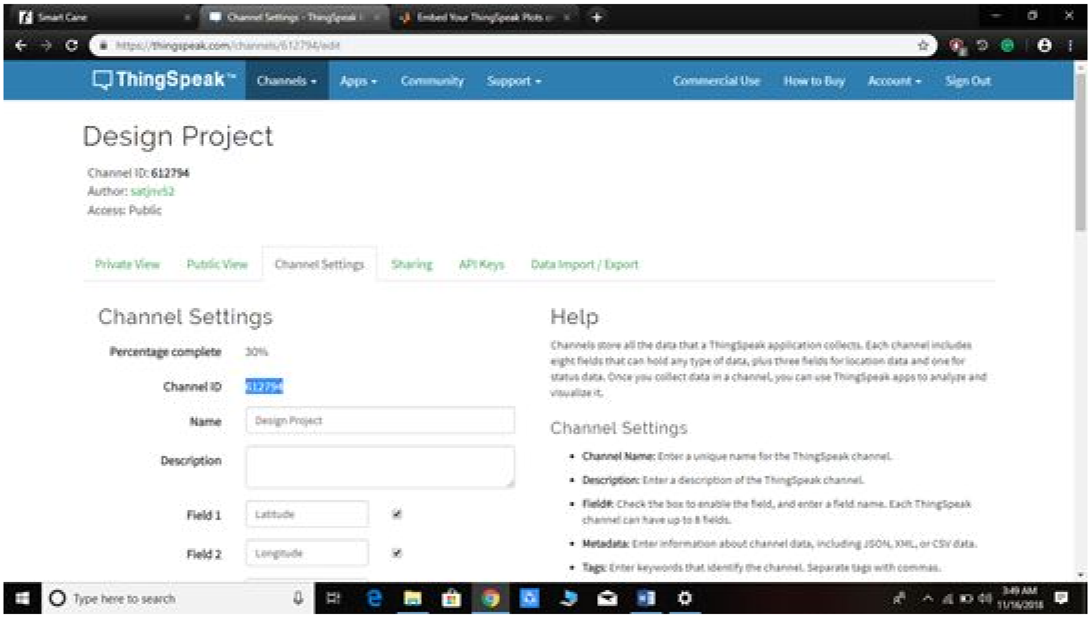
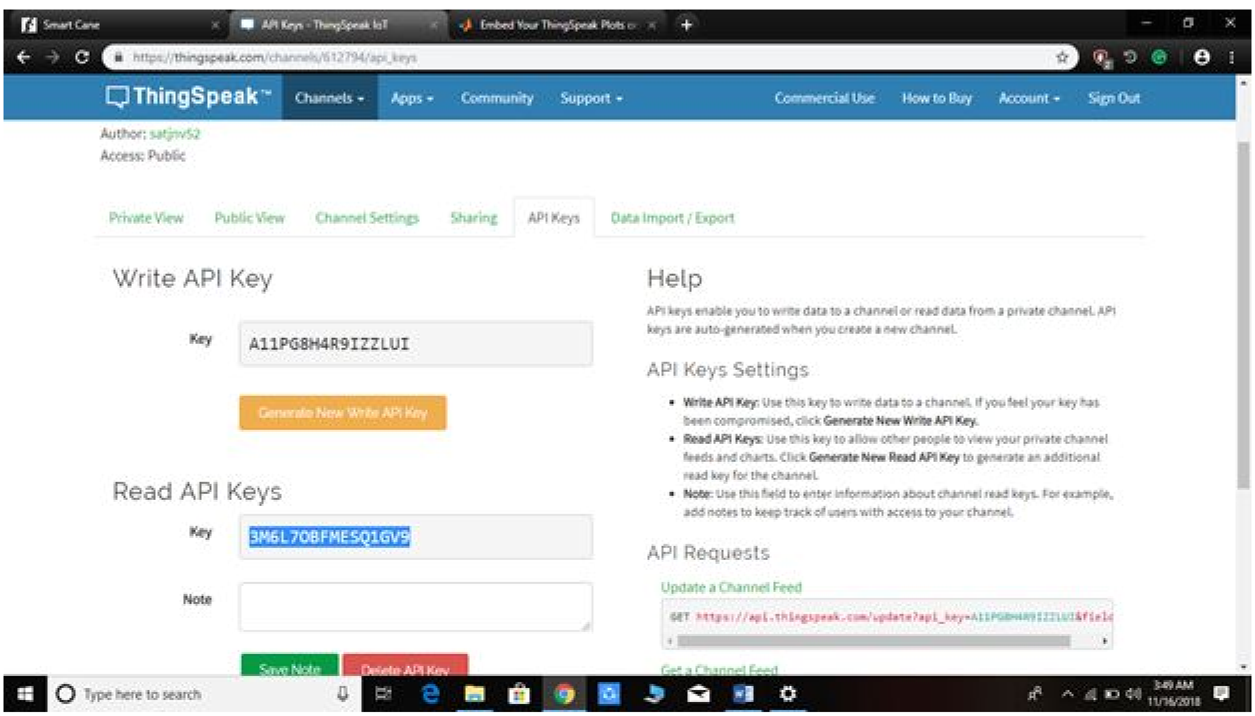


Fig.5



(Shown above is the CHANNEL id and READ API keys highlighted) (Fig. 6)

2> Developed Mobile Application:

The user at any point of time can see the health stats using the mobile application developed. Global and dynamic use of the application makes it an essential feature in the development of the product further. The data displayed in the application is directly fetched from the Thingspeak IO, and thus is more accurate and precise.

App specification:

Name of the application: Design_Project

Version: 1.0

Size: 17.05 MB

UI design: https://thunkable.com/

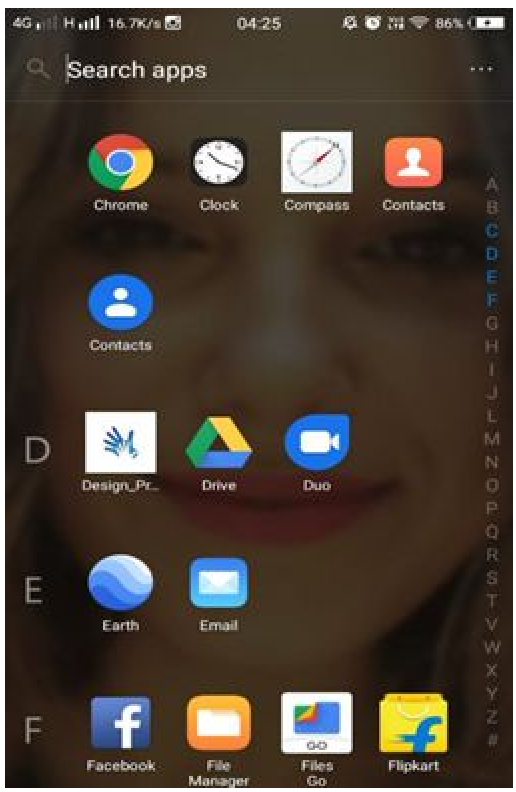
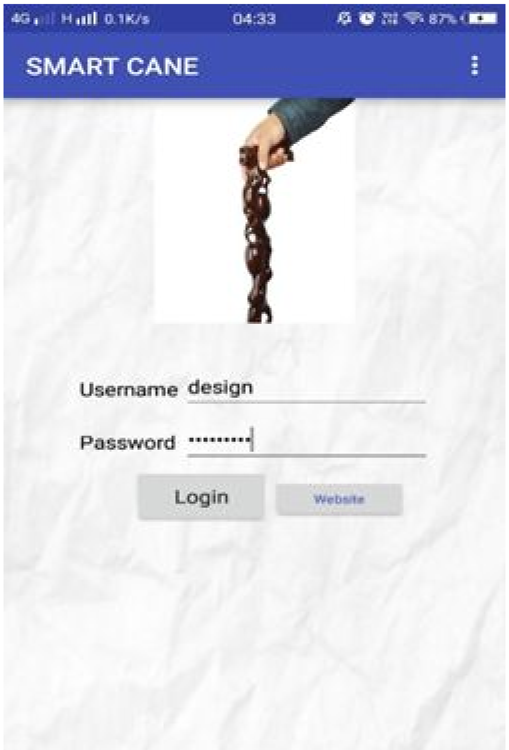


Fig. 7(app named Design_Project**)**



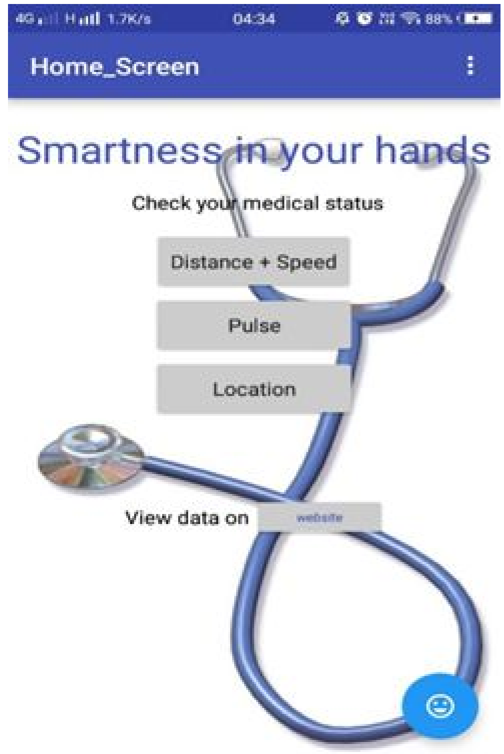


Fig. 8 **Fig. 9**

For initial purpose the username and password has been fixed, but there can be multiple logins at a time once firebase is taken into use.

Username: design
Password: login@123

In **Fig. 8** one can view the Website developed by clicking onto the website button. It will follow the url: https://pnidhi26.github.io/SmartCane/

Or else if after clicking on the login button in **Fig. 8, Fig. 9** will be displayed which is the home screen for the user. The screen follows with 3 buttons labeled as **Distance +Speed, pull**. After clicking on any of the three buttons the user will get the data particular to the label of the button.
For ex. After clicking on the Distance +Speed button, the user will be redirected to the screen as shown below in **Fig. 10**

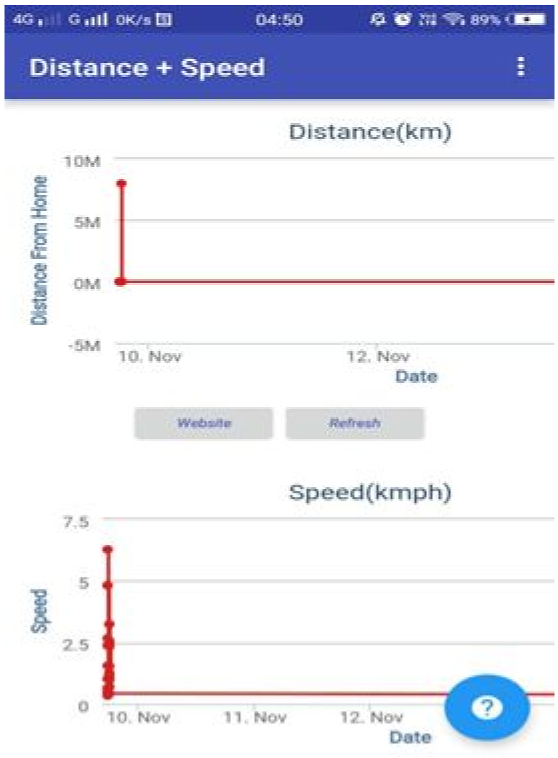


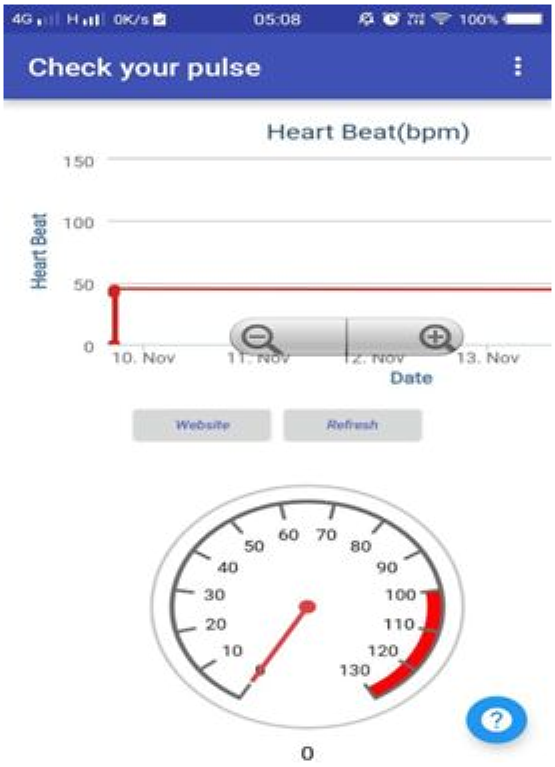
Fig. 10

Fig. 11

The user can refresh the data to view the stats at that moment of time. Also, a button labeled as website is provided so that the user can look upon the data directly on the website.

If the product is offline at any point of time and if the user looks upon the data, the data last fetched will be displayed. Once, the product is connected to the network, the data is updated.
Also, to facilitate the user about the how's and where's of functionalities, a screen action button is provided which when pressed opens another screen labeled as "Help" as shown in **Fig. 11.**

Similarly, once the user is back home, he/she can see the stats for some other health factors such as pulse. (images attached below)



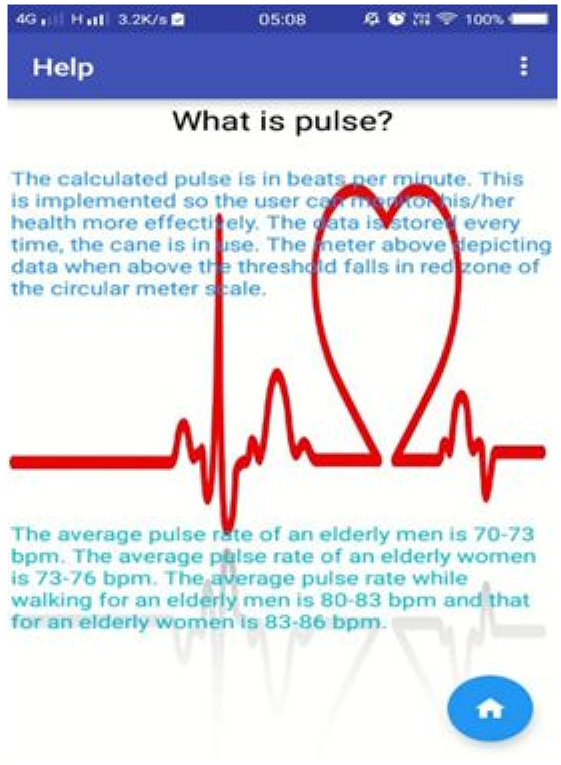


Fig. 12 **Fig. 13**

The location labeled button on the home screen of the application once clicked takes the user to the browser to open the google maps. It will take 30-40 sec of time for the page to get loaded as we have used the google maps API. The location marked will be the current location of the product. This page when refreshed will display the updated location. Also when the product is off network, the location data last fetched will be displayed, On network and here goes the dynamic data.
(images attached below)

(Fig. 14) The current location of the stick is displayed
(This page is hosted on our github profile.)

An overview of different stages of Mobile App Development Lifecycle is shown below **(Fig. 15)**

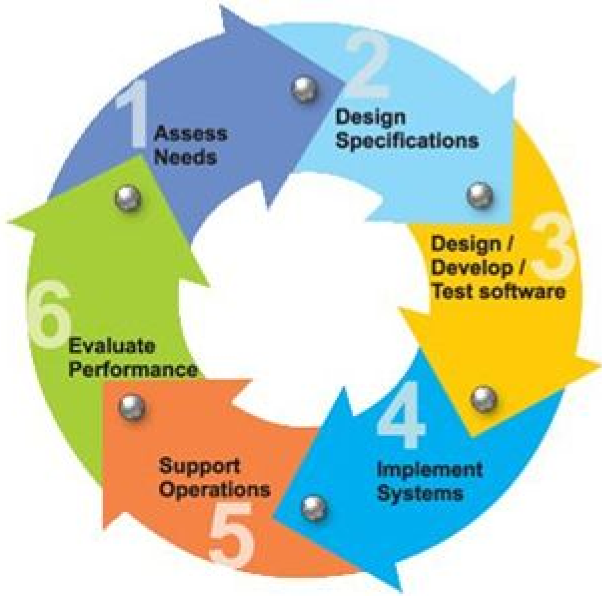


Fig. 15

The APK of the app is attached for reference

3> Developed Web Interface:
(https://pnidhi26.github.io/SmartCane/)

A web interface to keep check of the health is a key feature of the product. The web interface developed is based on several technologies as mentioned below:

- HTML5
- CSS3
- Java Script
- Bootstrap 4
- JQuery
- PHP
- Github

An ER Diagram made is easy to understand:

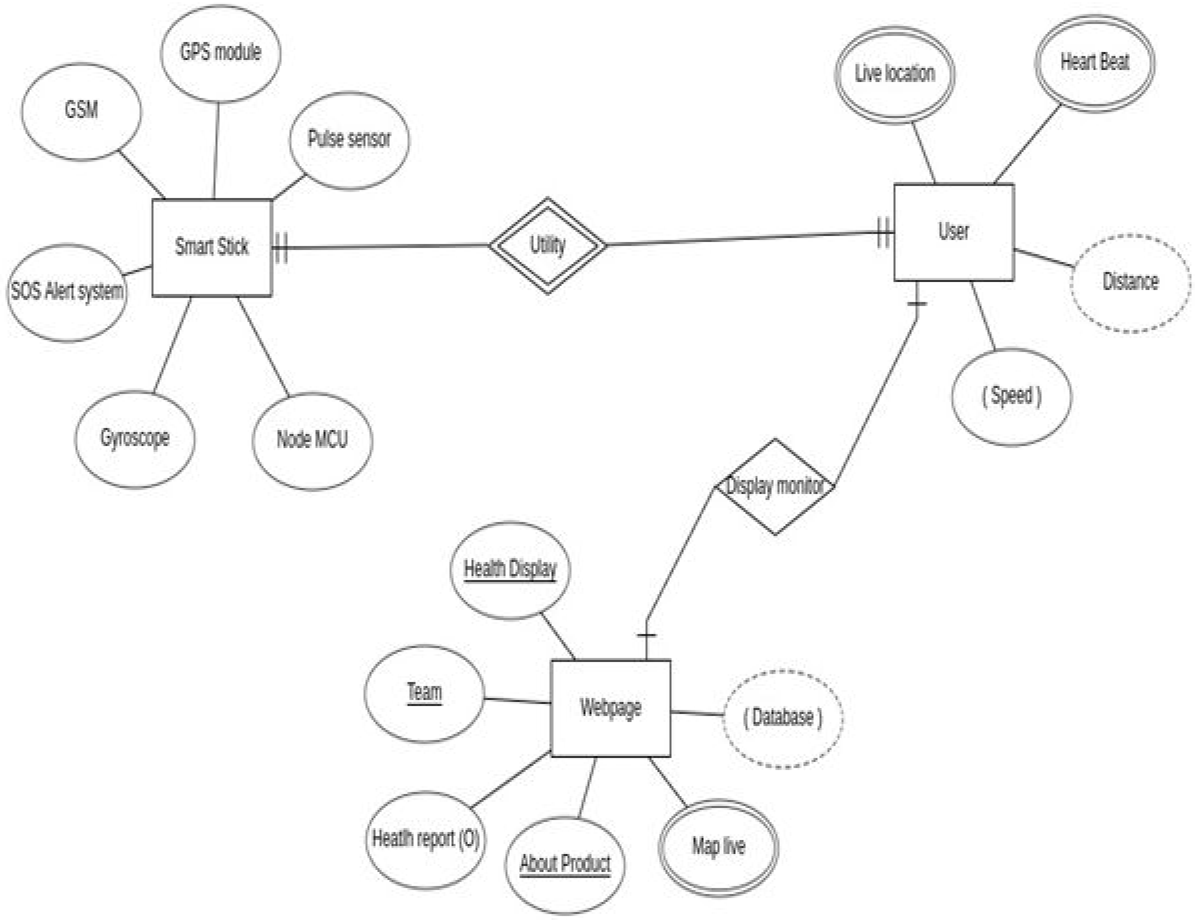


Fig. 16

The home page of the website have three hovers labeled as :

<u>A</u>: SPEED(KMPH)/DISTANCE(KM)

<u>B</u>: PULSE(BPM)/HEARTBEAT

<u>C</u>: LIVE LOCATION/ Latitude and longitude

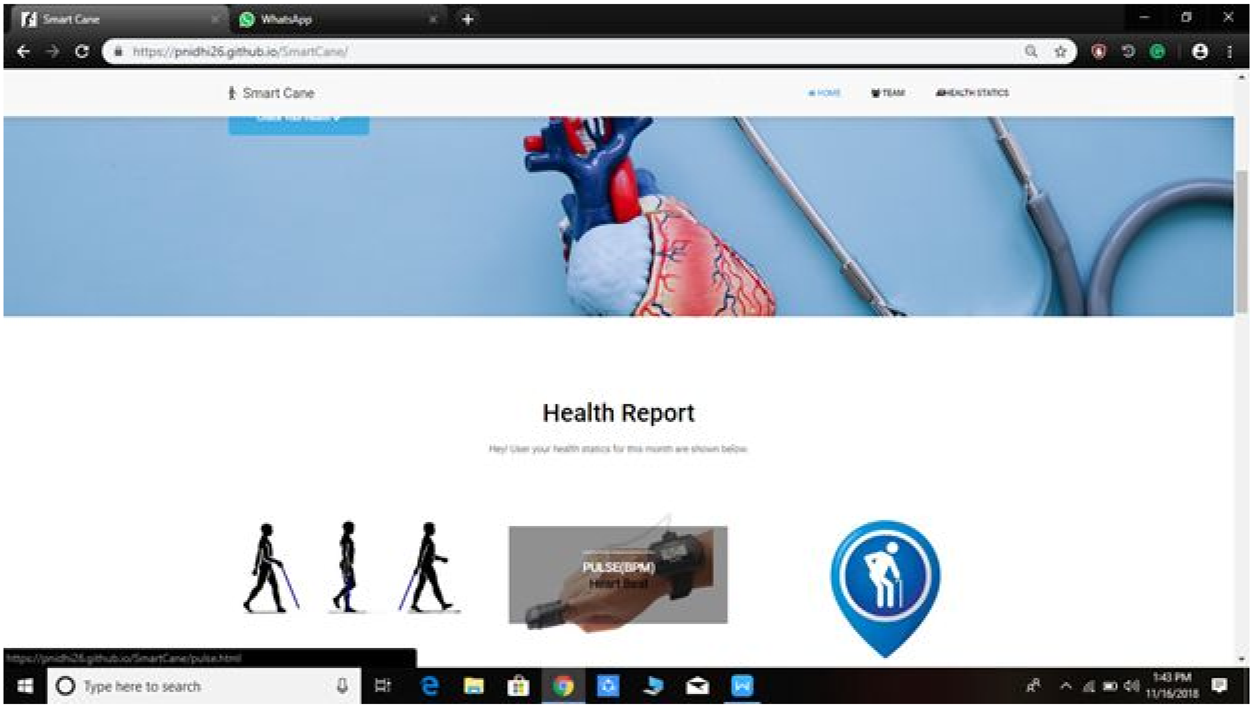


Fig. 17

Also, on the home screen we have some bootstrap containers displaying the number of users we have of our product, the current number of companies manufacturing the same type of product etc.
These all things are part of the research we had done.

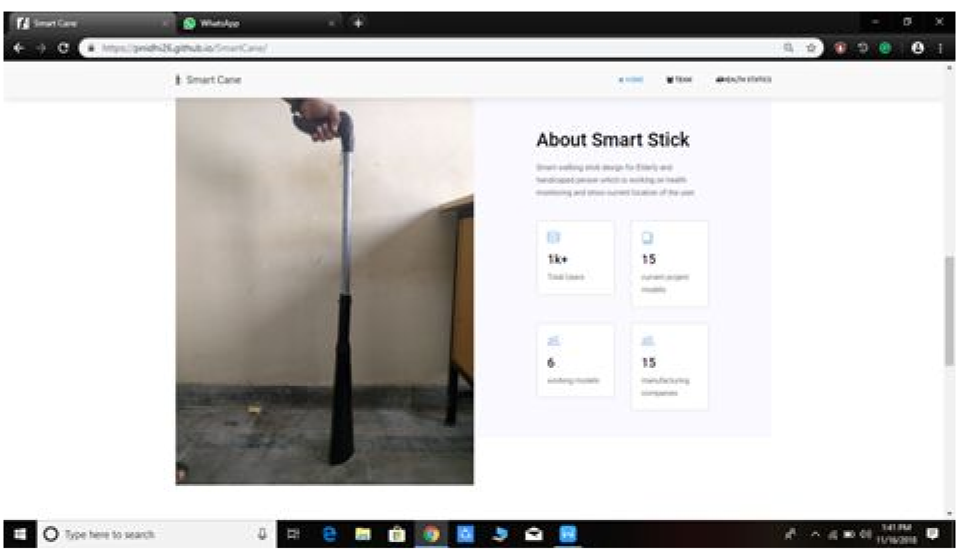


Fig. 18

Also we provide customer support. For this the user can connect to us using the mail option. The back-end for this is done in PHP.

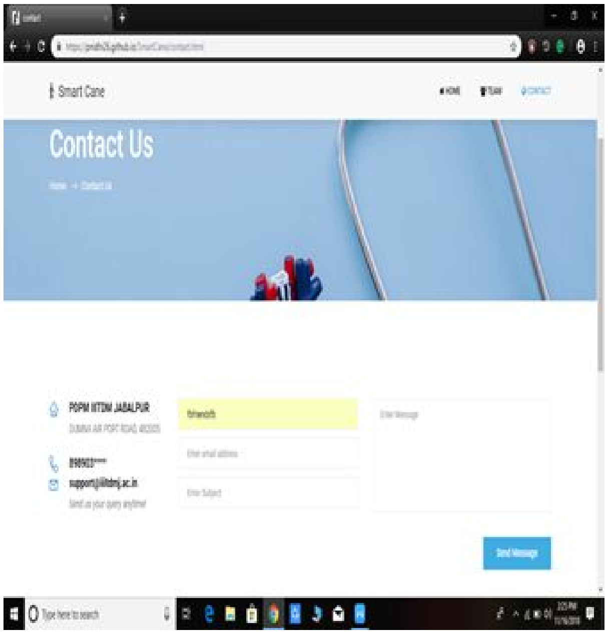


Fig. 19

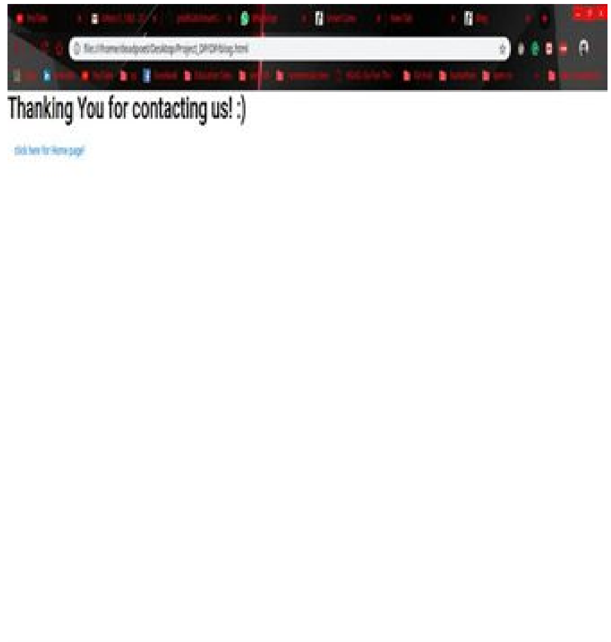


Fig. 20

Now clicking onto the hovers labeled as SPEED(KMPH)/DISTANCE(KM), PULSE(BPM)/HEARTBEAT, LIVE LOCATION/ Latitude and longitude we will be redirected to the page containing the data of your health particular to the label. This data is fetched directly from Thingspeak IO API. This data fetched is dynamic and can be viewed at any instant of time. This is done by inserting the following piece of code in our base code. (pulse.html/locatiton.html/speed.html)

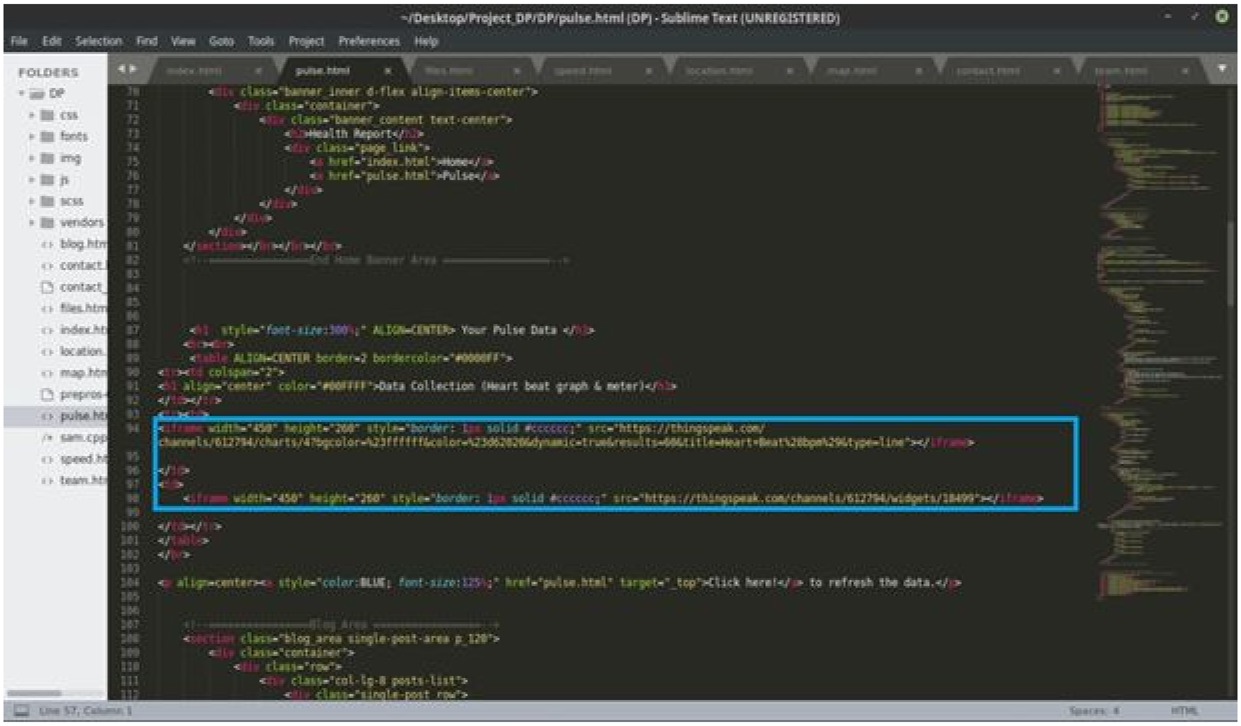


Fig. 21

The below page is opened once the hover is clicked on the home page.

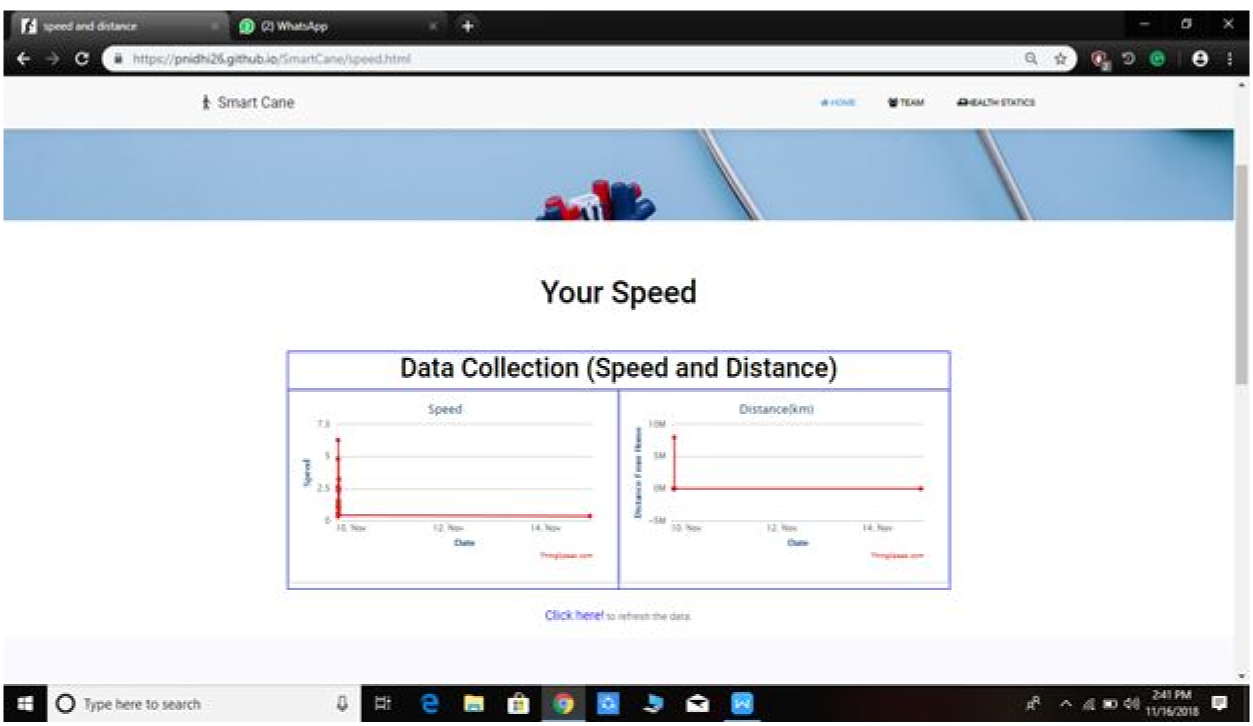
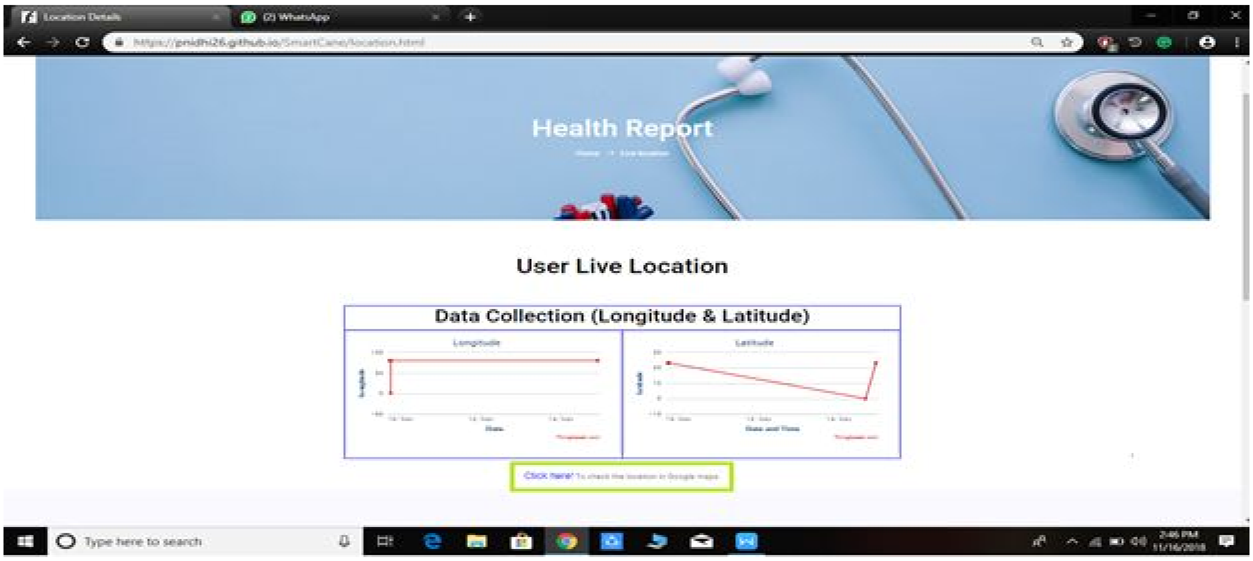


Fig. 22

One can refresh the data to see the accurate data at that moment of time. Also, the exact location of the user at the time can also be viewed. We give proper longitudes and latitudes for the more precised experience.



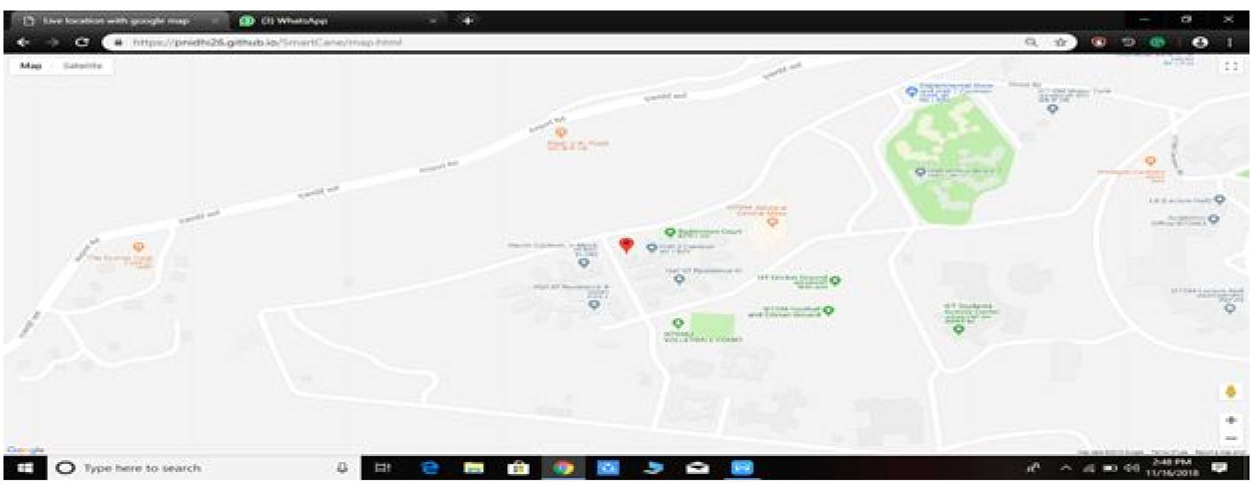


Fig. 23 & Fig. 24

The location of the product can be viewed on google maps too. **(Fig. 24)**

The code for this includes the use of Google API key, READ API keys from thingspeak IO and channel key.(map.html)

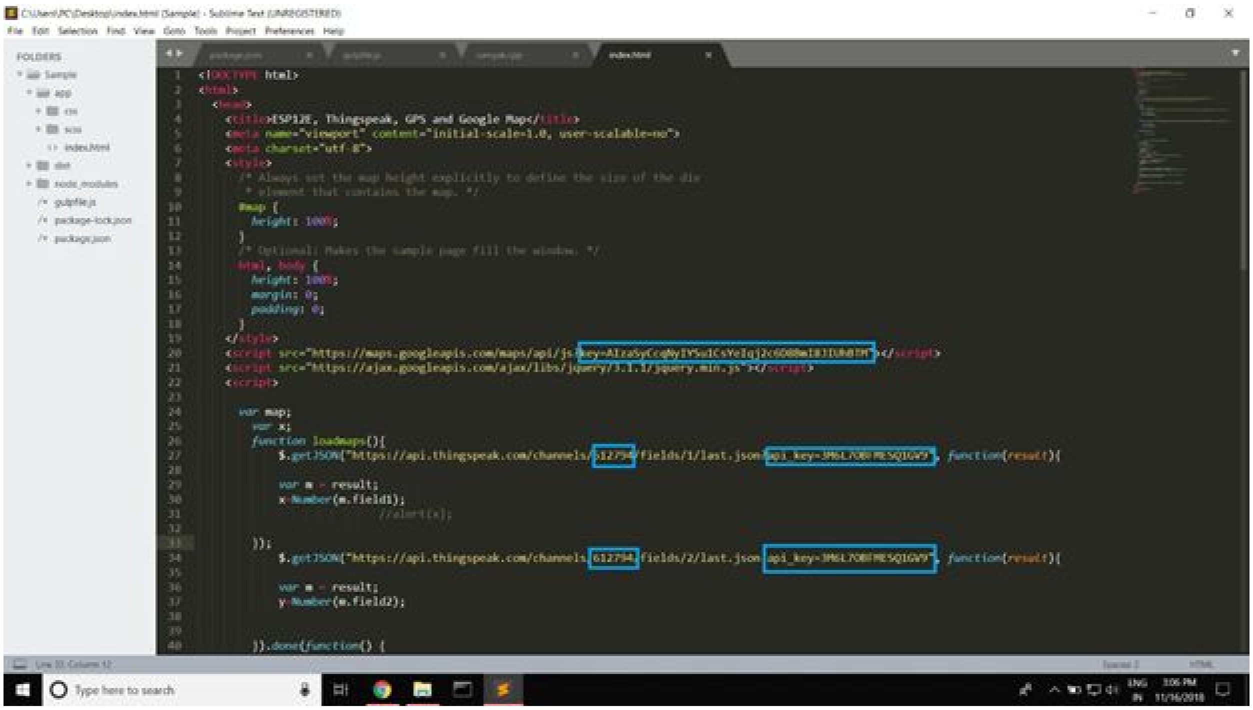


Fig. 25

The link to the project files is:

https://github.com/pnidhi26/SmartCane

Mechanical Engineering

Members Working :

- Raunak Patel 2016203 ME

Final product design

Placement of sensors:

- All the sensors are installed at the bottom of the stick. This lowers the centre of gravity of the whole system making it more stable.
- Pulse sensor is placed on handle which makes it easily accessible.
- SOS button is placed near to the neck of the stick which is free from mishandling and also accessible when needed.
- Bottom part is aerodynamic hence reduces air resistance.
- “FALLING IS A FEATURE”. We cannot use a fully stable bottom because it will affect the other feature of the stick. So bottom design is more like a golf club which makes it stable.

Specifications:

1. 3 level of height adjustment from 35 inch to 45 inch provides user customisation.
2. Suitable for user who needs to exert body weight directly on to the stick.
3. Rubber tip at the bottom which is anti-slip.
4. Grip to suit the natural wrist position.

Height Adjustment:

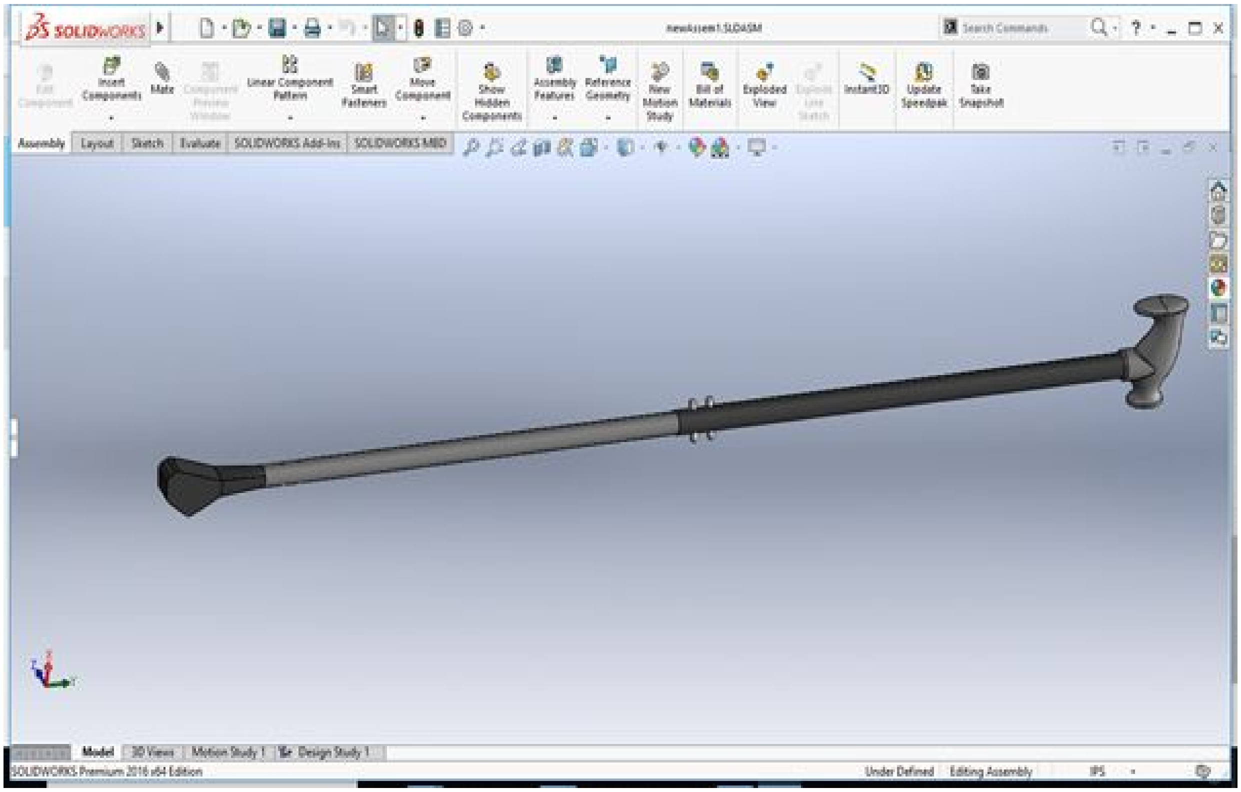
1. Loosen the locking collar.
2. Adjust the tubing to desired height manually.
3. Tighten the locking collar.

Initial telescopic design

Material for original product:

- Handle-high quality PC material(matte)
- Rod-aluminium alloy
- Foot-wear resistant soft rubber

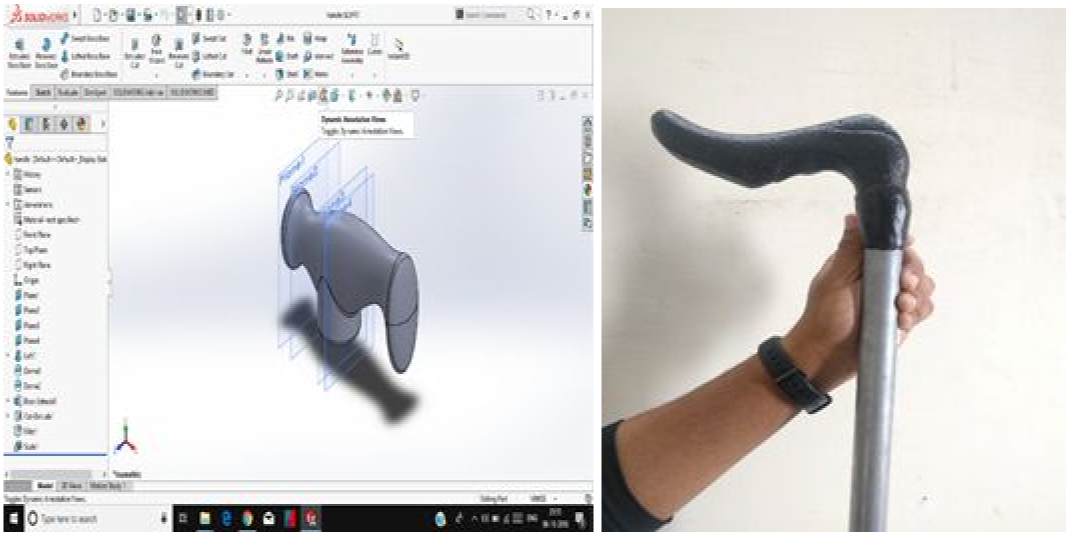
Photoview 360 rendering



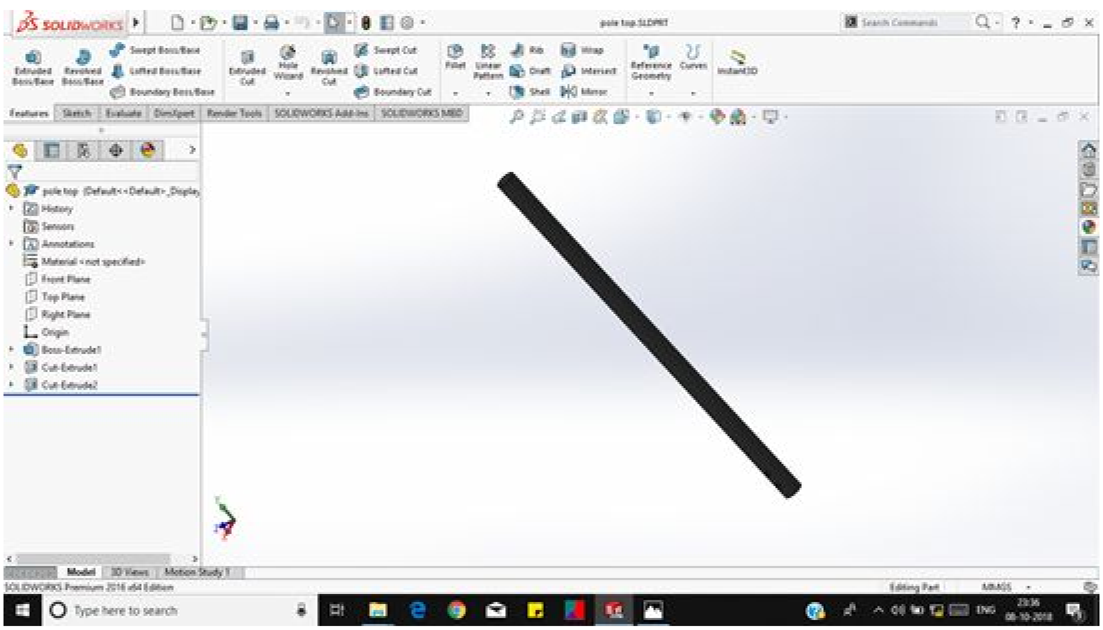
CAD model of the stick

Base of the product modified in order to accomodate sensors

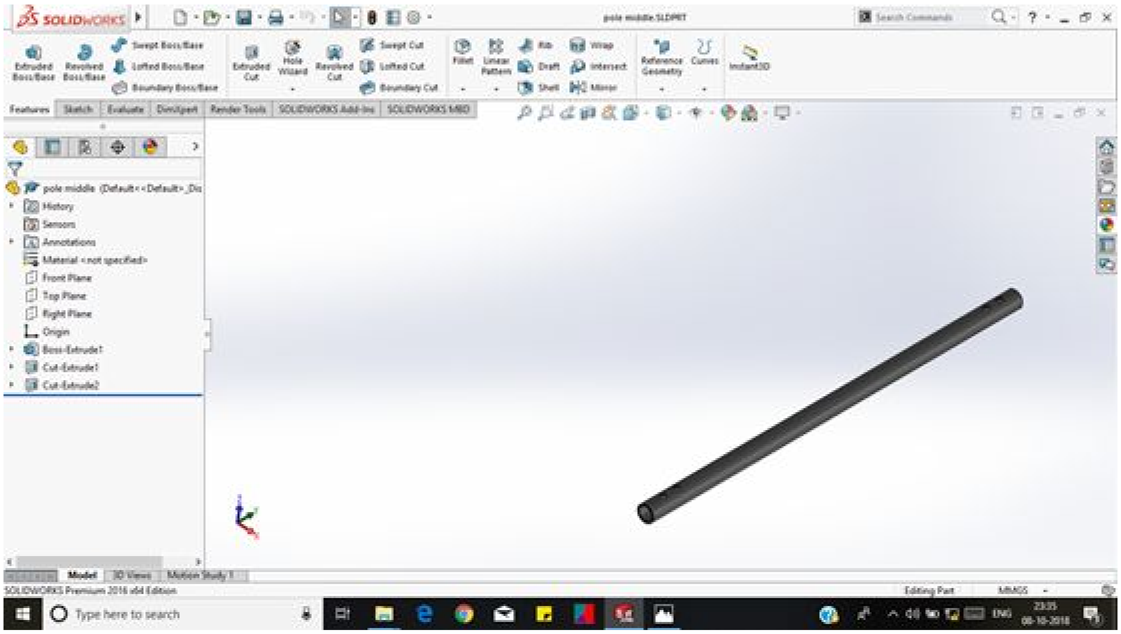
CAD model of different parts:



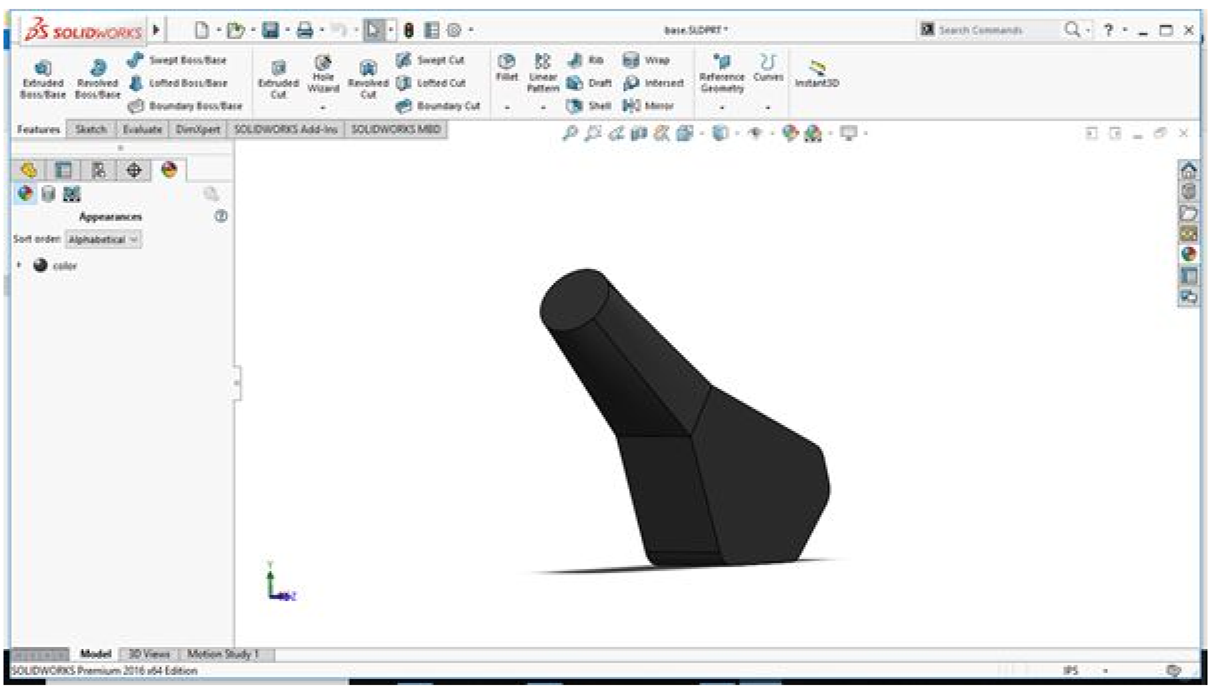
Handle



Top part



Bottom part



Base

Design Department

Members working:

Anusha Tharamal 2016508

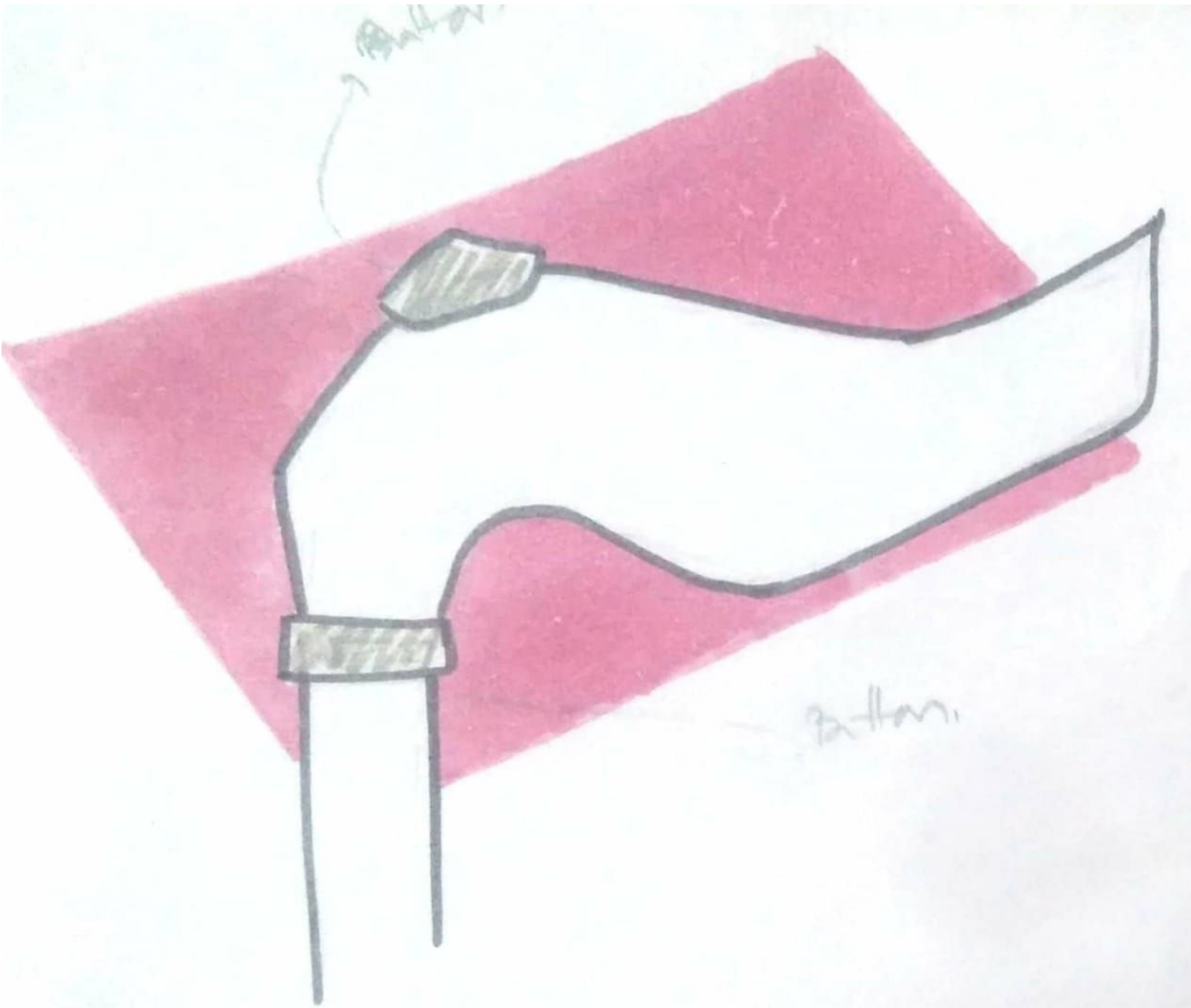
Description:

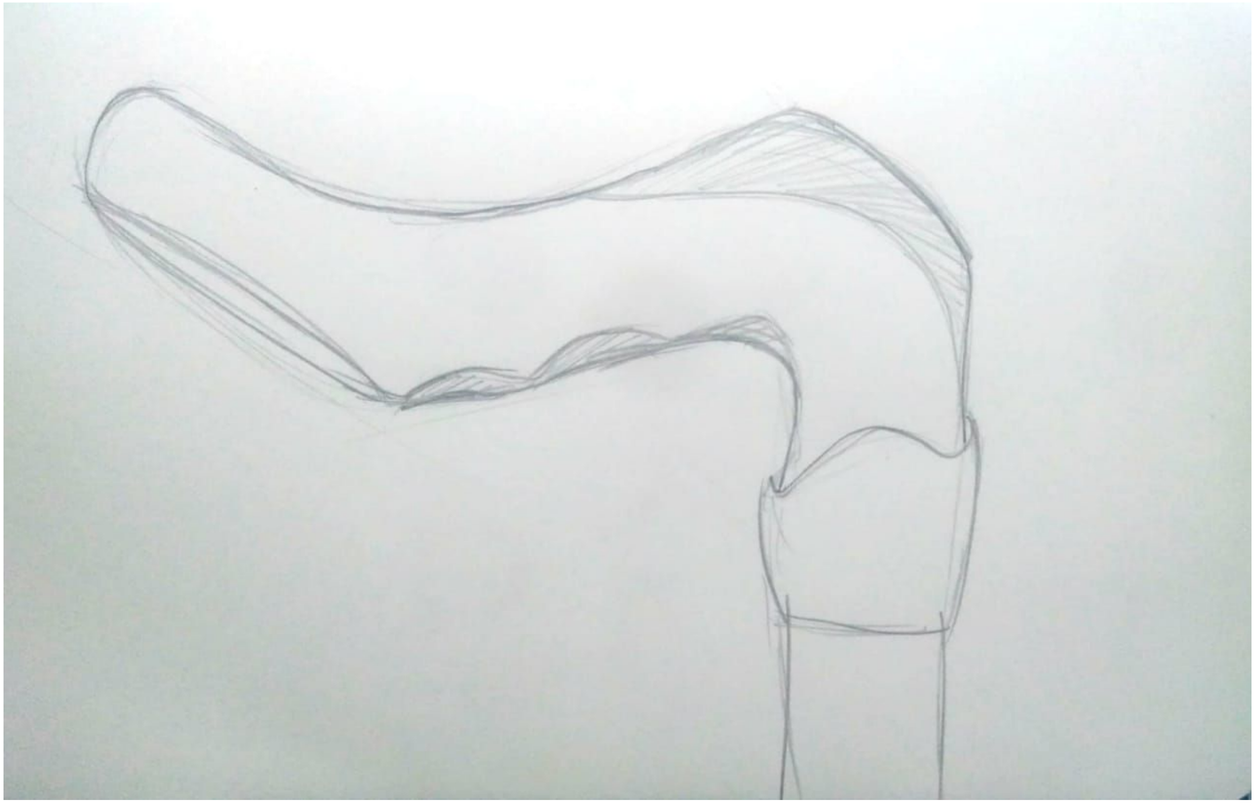
To make the walking stick more ergonomic and aesthetically pleasing so that the elderly may use it without any inconvenience, a new design was to be made.Our proposed design is made specifically keeping the elderly in mind, who, as a study has found, are interested in having a walking stick that is aesthetically pleasing. Each stick will be specifically designed for the person.

Our main objective was to make the existing walking stick smarter. For this using of a lot of sensors and circuits are required which could end up making our product look bulky, that can have a negative effect. But due to funding constraints we had to resort to using bulky sensors for prototyping.

Handle

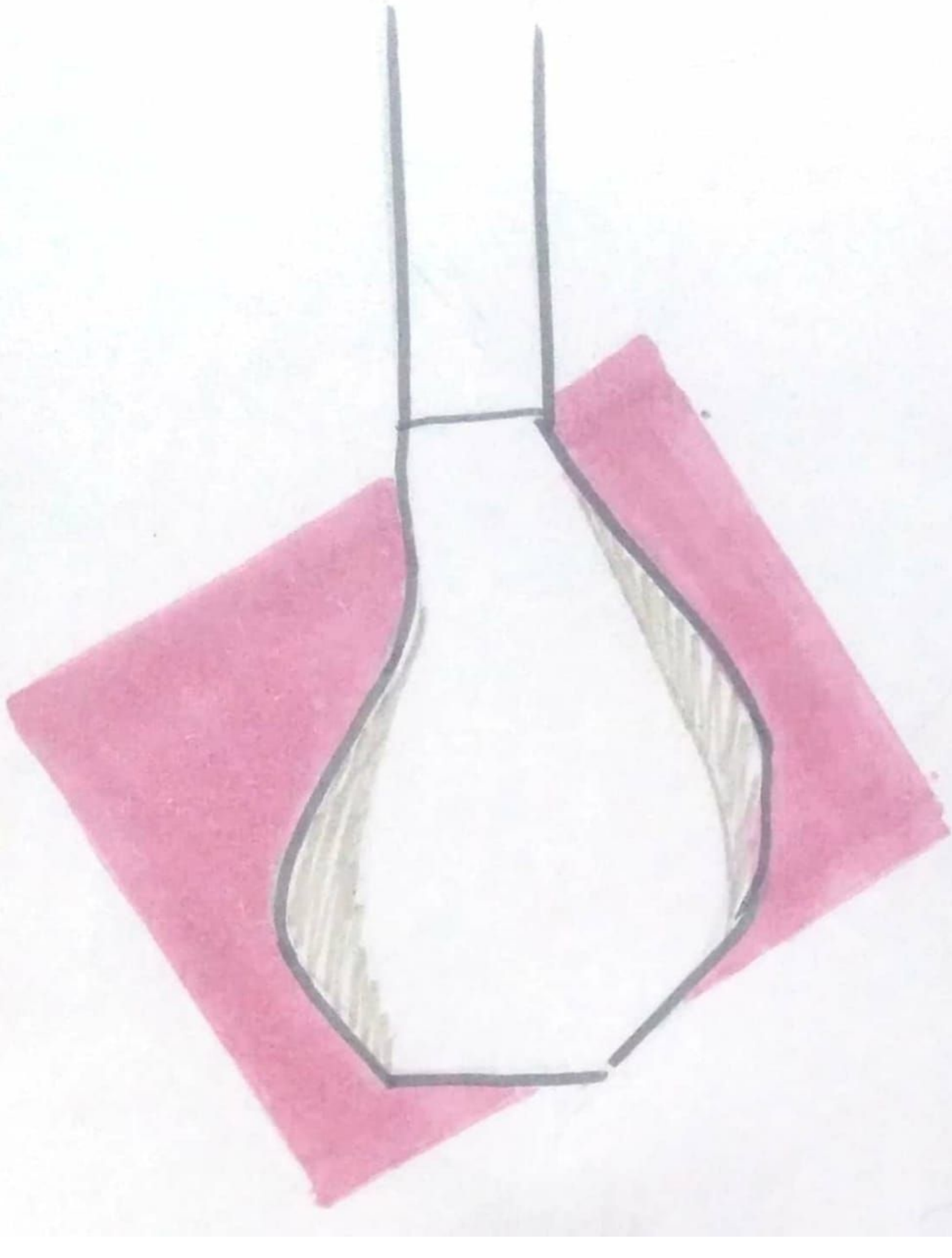
Handle is ergonomically designed for an individual as that is where most of the pressure goes. As our target users are elderly,it is important to make sure that while designing the walking stick, comfort is not to be compromised. Each handle will be specifically designed for the person depending on their anthropometric measurements that include Measurements of palm, Finger length and breadth, complete hand, Grip inner diameter, with dynamicity kept in mind.


Button.



Base

In Order to keep the sensors in place and working properly, the base of the stick is inspired from a golf club. A golf club was chose also because it is one thing that you can relate to an elderly person. Its kept simple and Geometric with less curves. It has a very light touch of future to it.



Renders