

Project Report

**Detection of Knee Varus Thrust using IMU Sensors**

*Submitted by*

**Jinesh R**

Second Year (Electrical and Electronics Engineering, NIT Trichy)

*Over the Period*

**May 20 2019 – July 20 2019**

*In*

**Neuromechanics Laboratory, IIT Madras**

CONTENTS

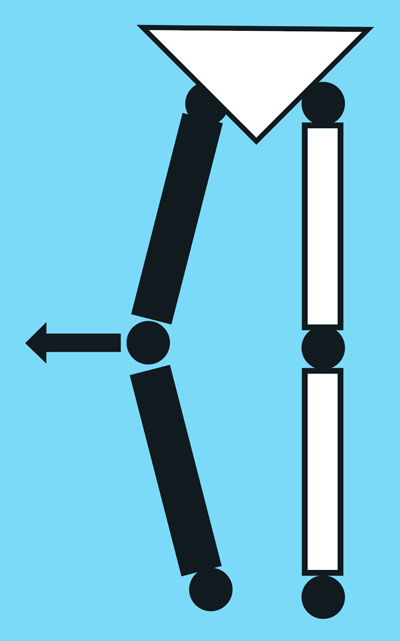
1. ABSTRACT---------------------------------------------------------------------3
2. INTODUCTION ----------------------------------------------------------------3
3. PLATFORMS USED-----------------------------------------------------------4
4. METHODOLOGY

1. Interfacing MPU-9250 breakout board with Arduino Uno--------4
2. Using 9DoF Razor IMU board --------------------------------------5
3. Bluetooth and SD card module----------------------------------------6
4. Communication between two microcontrollers---------------------7
5. Storage in SD card------------------------------------------------------7
6. Python Programming---------------------------------------------------8
7. CODES
8. Arduino with Sparkfun-------------------------------------------------9
9. Python ------------------------------------------------------------------15
10. Arduino -----------------------------------------------------------------17
11. CONCLUSION ---------------------------------------------------------------32
12. REFERENCES ----------------------------------------------------------------33

**Abstract**

This project aims to build a prototype that monitors the gait movement of the elderly people legs, who are likely to have knee varus thrust instead of going to a sophisticated gait laboratory.

Varus thrust is the visualized dynamic bowing out of the knee laterally while the limb is weightbearing (stance) during ambulation, with the return to a less varus alignment during nonweightbearing (swing). Varus thrust can be assessed in a gait lab, where it is defined as the difference in the hip-knee-ankle angle between heel strike and the first varus peak.



**Fig 1**

**Introduction**

The movement of the leg about the frontal or coronal plane of the human anatomy can be tracked using Sen14001- The SparkFun 9DoF Razor IMU which combines a SAMD21 microprocessor with an MPU-9250 9DoF (9 Degrees of Freedom) sensor was used in this project, which has a 3 Accelerometer, a 3 axis Gyroscope and a 3 axis Magnetometer. By using these sensors values Euler angles were calculated.

The two SparkFun sensors were mounted on simple straps and located on the shank and thigh. The data from IMU sensors is processed by ATSAM21 microcontroller and after applying Kalman filter the Euler angles are calculated. This data is stored in an SD card.

**Platforms Used**

1. Arduino Uno
2. 9DoF Razor IMU board
3. Sparkfun- SEN14001

**Methodology**

1. **Interfacing MPU-9250 breakout board with Arduino Uno**

I2C protocol was used to establish communication between the MPU-9250 breakout board and Arduino Uno. The raw data values of accelometer and gyroscope sensors from MPU-9250 via I2C protocol were stored in the SD card by using SPI protocol with an external micro SD card shield at a sampling rate of 40Hz. The obtained data is saved as a text file in the micro SD card.

The IMU sensor is calibrated to make the initial position as the reference by keeping the sensor still in the starting position for the initial 10 seconds. Accelerometer, gyroscope data were calibrated. Angular position is obtained by Euler angle form accelometer and integrating gyroscope data (angular velocity) time. Complimentary filter is used to smoothen the angular position. The filtered Euler angles were printed in serial monitor and verified for different orientation of MPU-9250

Fig2. Complementary filter output for pitch(red), yaw(green) and roll(blue)

1. **Using 9DoF Razor IMU board (SEN14001)**

The Sparkfun 9DoF Razor IMU board comprises of ATSAMD21 microcontroller and MPU-9250 along with other peripherals. The module was designed to store data in SD card. It is a standalone module while can be powered and programmed through USB cable. It has a provision to be powered by LiPo battery. The microcontroller used ATSAMD21 has 256KB Flash Memory,32KB SRAM.

This module is built solely for the purpose of getting IMU data. The library SparkFun MPU-9250 library enables the sensors digital motion processing (DMP) features. The data from IMU sensor is received using I2C protocol. The values received through I2C protocol from the IMU sensors will be raw data, which have to be processed to obtain the pitch and roll. Kalman filters were used to reduce the noise and smoothen the values of pitch and roll and Kalman filters turned out to be a better filtering method compared to the Complimentary filters.

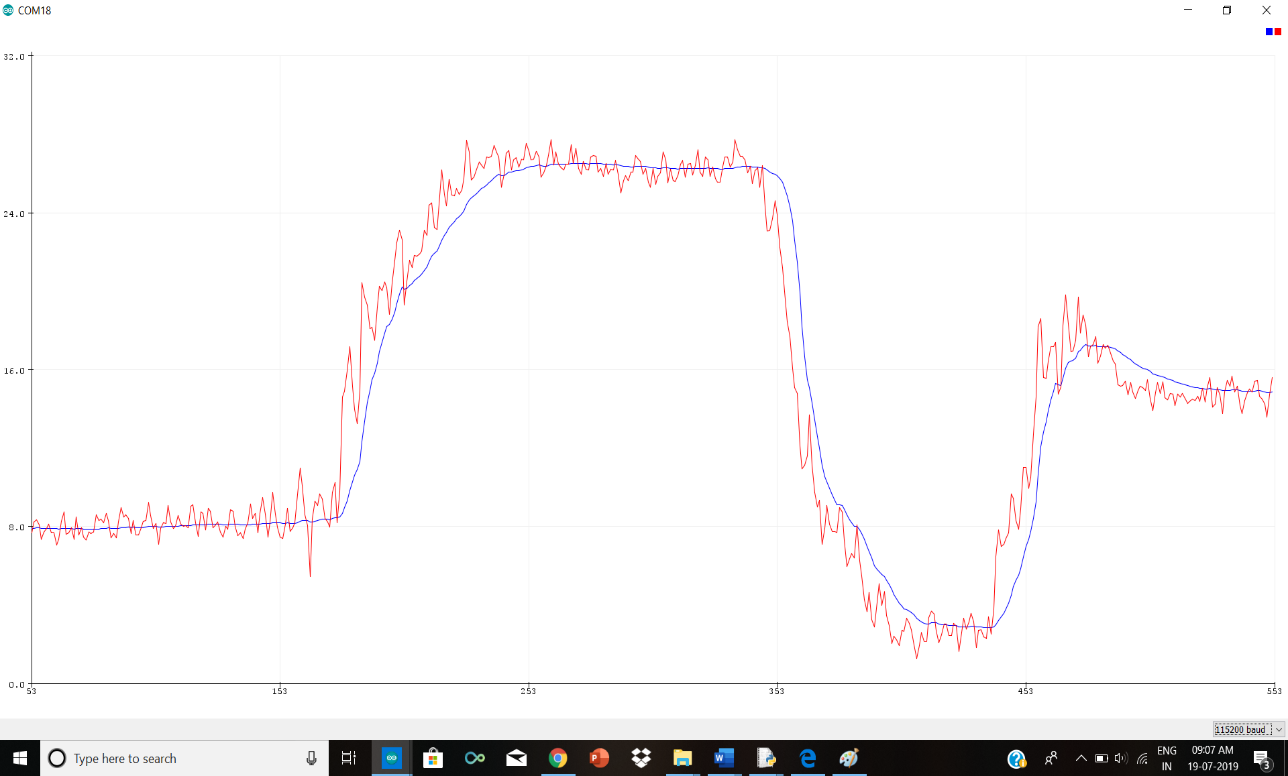


Fig3. Pitch rotation value from IMU. RED- raw pitch rotation, BLUE- Kalman filtered pitch rotation

1. **Bluetooth and SD card module**

The obtained data can be stored in one SD card by receiving the data from the other hand using Bluetooth communication. This method requires two Bluetooth modules, one to send and the other one to receive and one SD card module to store data in the SD card. This results in all these three units to be continuously in use and be powered. Bluetooth communication also consumes power.

Hence it is found better to use two SD card modules and store the two hands data in two separate SD cards (more economical in terms of power consumption).And when an external push button is pressed in the Master module all the data collected in the SD card of slave module gets transmitted through Bluetooth communication and stores the slave values in the SD card of Master module.

1. **Storage in SD card**

The IMU data processed in the microcontroller was stored as a text file in the SD card. The communication between the microcontroller and the SD card was done using the SPI Protocol.

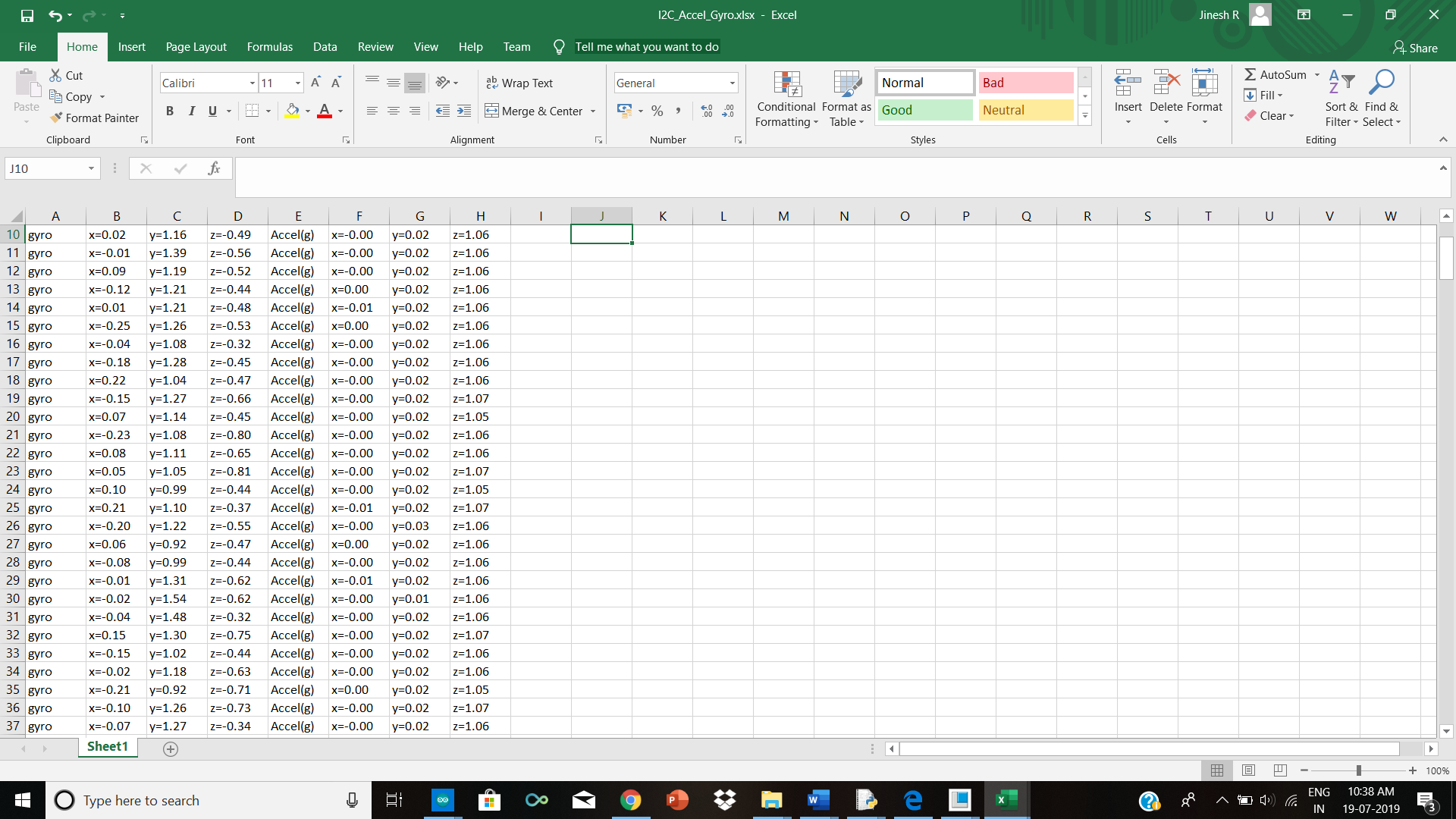


Fig4.values of IMU stored in SD card

1. **CAD designing**

The CAD design was done using Creo software. 2 models were made for this project, the **base** dimensions 6.5cm x 4.5cm x 2.5cm which is used for holding the batteries, sen-14001, Bluetooth together, a small slot is given for USB connection and for handling micro SD card in the module. The **top** dimensions 4.5cm x 4.5 cm which is used to cover the base 4 holes and a slot is given for placing the push button and Bluetooth. This CAD model is fed to a 3D printing software to get the 3D printed parts for placing the sensor modules in the thigh and shanks for measuring Knee Varus.

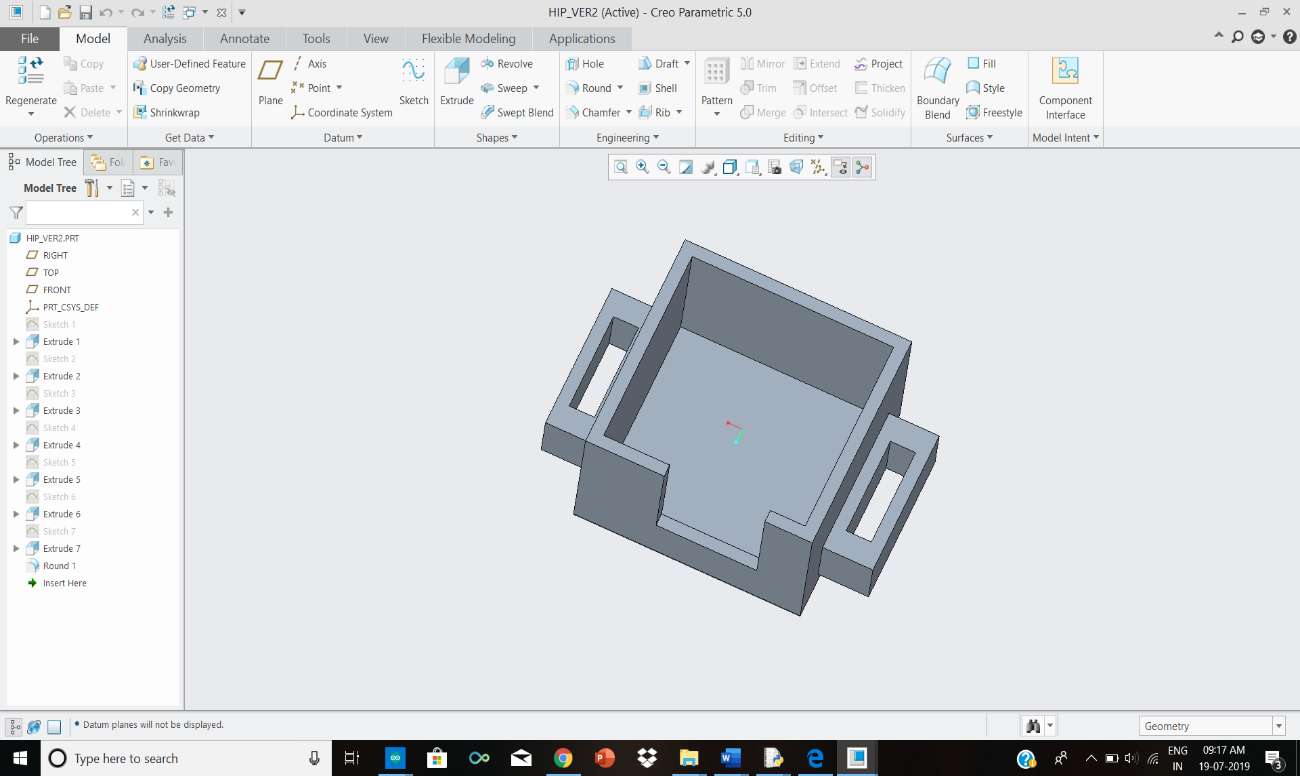


Fig5. Base part for mounting the sensor modules

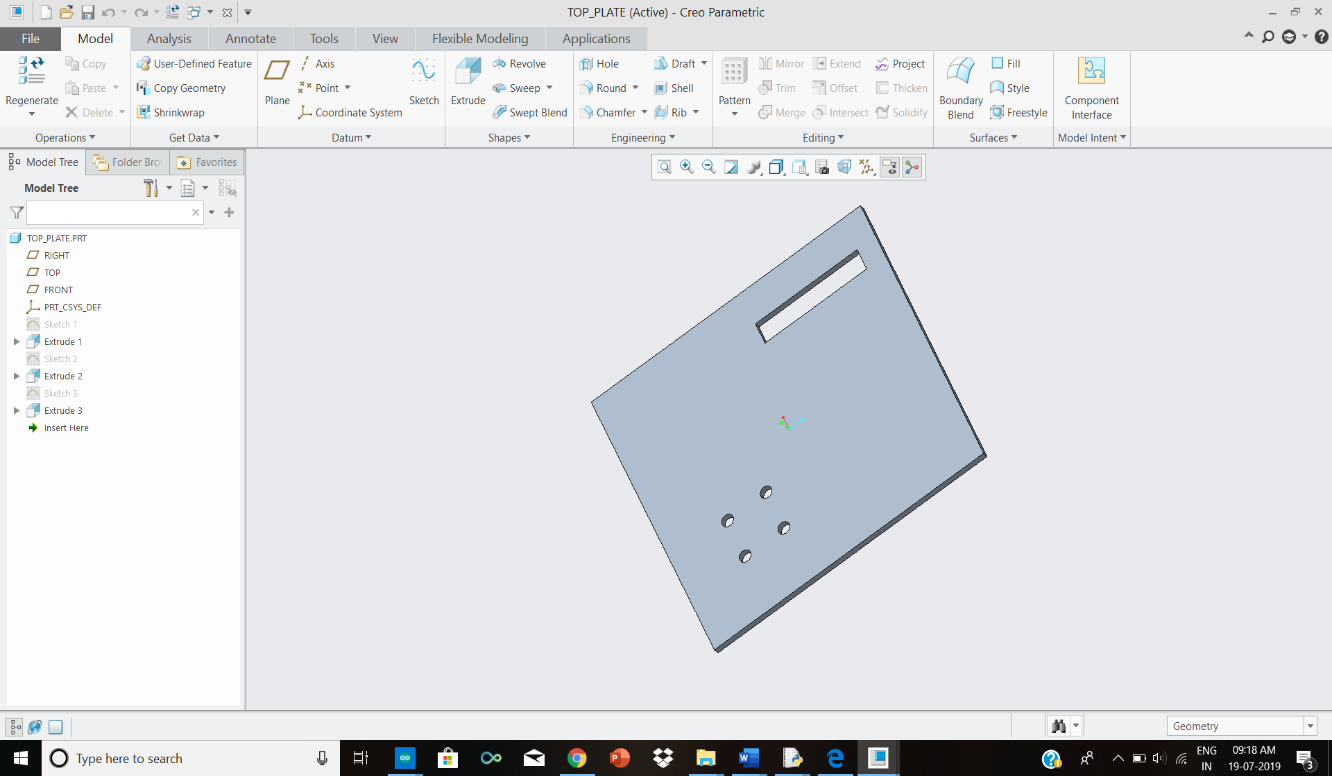


Fig6. TOP part of the sensor mounting module

1. **PYTHON Programming**

For getting the real time angle of the knee in varus thrust, Using 2 IMU sensors placed on thigh and shank of the leg, we will get the angles based on the movement of the leg, angle **a** and angle **b** respectively. Using these angles we can measure the angle made by the knee joint joining them in the frontal plane, for measuring Knee Varus which is **c.**

For Knee Varus,

**c = 180 – (a + b)** where c, a and b are in degrees.

The output values from two Sen14001 (a, b) are sent to the python terminal for measuring the angle c and real time graph is plotted.

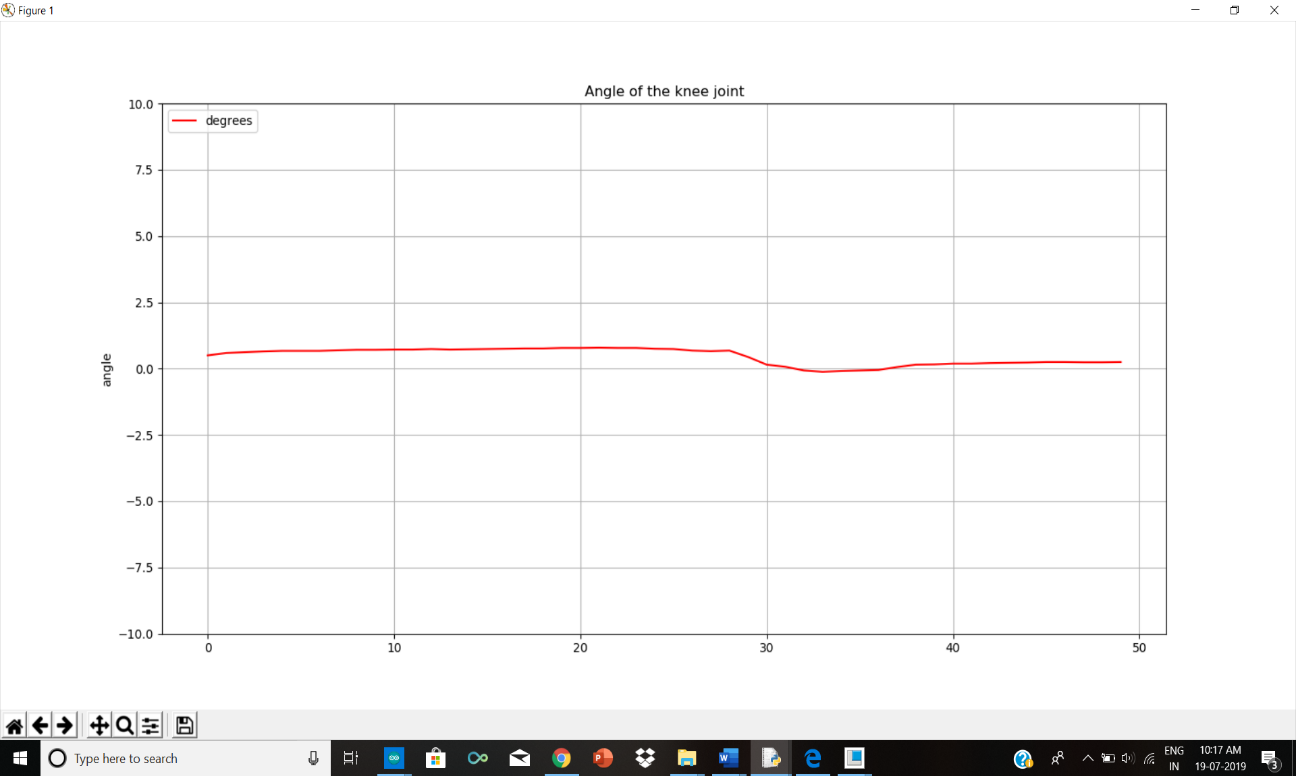


Fig7. Plotting the angle in Python terminal using Matplotlib

**CODES**

**-->Arduino with Sparkfun**

**1.**Using Sen14001 printing pitch value after applying Kalman filter

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MPU9250\_Basic

Basic example sketch for MPU-9250 DMP Arduino Library

Jim Lindblom @ SparkFun Electronics

original creation date: November 23, 2016

https://github.com/sparkfun/SparkFun\_MPU9250\_DMP\_Arduino\_Library

This example sketch demonstrates how to initialize the

MPU-9250, and stream its sensor outputs to a serial monitor.

Development environment specifics:

Arduino IDE 1.6.12

SparkFun 9DoF Razor IMU M0

Supported Platforms:

- ATSAMD21 (Arduino Zero, SparkFun SAMD21 Breakouts)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <SparkFunMPU9250-DMP.h>

#include <SimpleKalmanFilter.h>

#define SerialPort SerialUSB

MPU9250\_DMP imu;

SimpleKalmanFilter roll\_k(1, 1, 0.05);

SimpleKalmanFilter pitch\_k(1, 1, 0.075);

void setup()

{

SerialPort.begin(115200);

// Call imu.begin() to verify communication with and

// initialize the MPU-9250 to it's default values.

// Most functions return an error code - INV\_SUCCESS (0)

// indicates the IMU was present and successfully set up

if (imu.begin() != INV\_SUCCESS)

{

while (1)

{

SerialPort.println("Unable to communicate with MPU-9250");

SerialPort.println("Check connections, and try again.");

SerialPort.println();

delay(5000);

}

}

// Use setSensors to turn on or off MPU-9250 sensors.

// Any of the following defines can be combined:

// INV\_XYZ\_GYRO, INV\_XYZ\_ACCEL, INV\_XYZ\_COMPASS,

// INV\_X\_GYRO, INV\_Y\_GYRO, or INV\_Z\_GYRO

// Enable all sensors:

imu.setSensors(INV\_XYZ\_GYRO | INV\_XYZ\_ACCEL);

// Use setGyroFSR() and setAccelFSR() to configure the

// gyroscope and accelerometer full scale ranges.

// Gyro options are +/- 250, 500, 1000, or 2000 dps

imu.setGyroFSR(2000); // Set gyro to 2000 dps

// Accel options are +/- 2, 4, 8, or 16 g

imu.setAccelFSR(16); // Set accel to +/-2g

// Note: the MPU-9250's magnetometer FSR is set at

// +/- 4912 uT (micro-tesla's)

imu.dmpBegin(DMP\_FEATURE\_GYRO\_CAL |DMP\_FEATURE\_SEND\_CAL\_GYRO|DMP\_FEATURE\_SEND\_RAW\_ACCEL, 40); // Set DMP rate to 10 Hz

// setLPF() can be used to set the digital low-pass filter

// of the accelerometer and gyroscope.

// Can be any of the following: 188, 98, 42, 20, 10, 5

// (values are in Hz).

imu.setLPF(10); // Set LPF corner frequency to 5Hz

// The sample rate of the accel/gyro can be set using

// setSampleRate. Acceptable values range from 4Hz to 1kHz

imu.setSampleRate(40); // Set sample rate to 10Hz

// Likewise, the compass (magnetometer) sample rate can be

// set using the setCompassSampleRate() function.

// This value can range between: 1-100Hz

// imu.setCompassSampleRate(10); // Set mag rate to 10Hz

}

void loop()

{

if ( imu.fifoAvailable() )

{

// Use dmpUpdateFifo to update the ax, gx, mx, etc. values

if ( imu.dmpUpdateFifo() == INV\_SUCCESS)

{

printIMUData();

}

}

/\* // dataReady() checks to see if new accel/gyro data

// is available. It will return a boolean true or false

// (New magnetometer data cannot be checked, as the library

// runs that sensor in single-conversion mode.)

if ( imu.dataReady() )

{

// Call update() to update the imu objects sensor data.

// You can specify which sensors to update by combining

// UPDATE\_ACCEL, UPDATE\_GYRO, UPDATE\_COMPASS, and/or

// UPDATE\_TEMPERATURE.

// (The update function defaults to accel, gyro, compass,

// so you don't have to specify these values.)

imu.update(UPDATE\_ACCEL | UPDATE\_GYRO);

printIMUData();

}\*/

}

void printIMUData(void)

{

// After calling update() the ax, ay, az, gx, gy, gz, mx,

// my, mz, time, and/or temerature class variables are all

// updated. Access them by placing the object. in front:

// Use the calcAccel, calcGyro, and calcMag functions to

// convert the raw sensor readings (signed 16-bit values)

// to their respective units.

float accelX = imu.calcAccel(imu.ax);

float accelY = imu.calcAccel(imu.ay);

float accelZ = imu.calcAccel(imu.az);

float gyroX = imu.calcGyro(imu.gx);

float gyroY = imu.calcGyro(imu.gy);

float gyroZ = imu.calcGyro(imu.gz);

//imu.computeEulerAngles();

float p, rollk, pitchk ;

float y ;

float r ;

// accelX = aX\_k.updateEstimate(aX);

// accelY = aY\_k.updateEstimate(aY);

r = atan(accelY / sqrt(pow(accelX, 2) + pow(accelZ, 2))) \* 180 / PI;

p = atan(-1 \* accelX / sqrt(pow(accelY, 2) + pow(accelZ, 2))) \* 180 / PI;

rollk = roll\_k.updateEstimate(r);

pitchk = pitch\_k.updateEstimate(p);

//float magX = imu.calcMag(imu.mx);

//float magY = imu.calcMag(imu.my);

//float magZ = imu.calcMag(imu.mz);

/\* SerialPort.print(accelX);

SerialPort.print('\t');

SerialPort.print(accelY);

SerialPort.print('\t');

SerialPort.print(accelZ);

SerialPort.print('\t');

SerialPort.print(gyroX);

SerialPort.print('\t');

SerialPort.print(gyroY);

SerialPort.print('\t');

SerialPort.print(gyroZ);

SerialPort.print('\t');\*/

SerialPort.println(pitchk);

//SerialPort.print('\t');

//SerialPort.println(gyroY);

//SerialPort.println(pitchk);

// SerialPort.print('\t');

// SerialPort.println(y);

// SerialPort.print('\t');

/\* SerialPort.println("Accel: " + String(accelX) + ", " +

String(accelY) + ", " + String(accelZ) + " g");

SerialPort.println("Gyro: " + String(gyroX) + ", " +

String(gyroY) + ", " + String(gyroZ) + " dps");

SerialPort.println("Mag: " + String(magX) + ", " +

String(magY) + ", " + String(magZ) + " uT");

SerialPort.println("Time: " + String(imu.time) + " ms");

SerialPort.println();\*/

}

**-->Python**

**1.**Using Python to get 2 pitch values form 2 Sen14001 using pyserial and plotting the graph of the calculated value real time

import serial

import numpy

import matplotlib.pyplot as plt

from drawnow import \*

import threading

tempF = []

tempF2 = []

tempD = []

pressure = []

arduinoData = serial.Serial('COM17',115200)

arduinoData2 = serial.Serial('COM18',115200)

plt.ion()

cnt=0

def makeFig():

plt.ylim(-180,180)

plt.title('Angle of the knee joint')

plt.grid('True')

plt.ylabel('angle')

plt.plot(tempD,'r-',label='degrees')

plt.legend(loc='upper left')

temp=0

temp2=0

def drawFig():

while True:

drawnow(makeFig)

plt.pause(.001)

def data1take():

global temp

while True:

dataArray = arduinoData.readline()

temp = float(dataArray)

def data2take():

global temp2

while True:

dataArray2 = arduinoData2.readline()

temp2 = float(dataArray2)

#def datadifftake():

# while True:

t2=threading.Thread(target=data1take)

t2.start()

t3=threading.Thread(target=data2take)

t3.start()

t1=threading.Thread(target=drawFig)

t1.start()

while True:

while(arduinoData.inWaiting()==0):

pass

# dataArray = arduinoData.split('.')

tempd =180-(temp+temp2)

# P = float( dataArray[1])

tempD.append(tempd)

print(str(tempd)+str('\t')+str(temp)+str('\t')+str(temp2))

# pressure.append(p)

cnt=cnt+1

if(cnt>50):

tempD.pop(0)

# pressure.pop(0)

**🡪ARDUINO**

**2.** When a push button is pressed in master the IMU data from slave’s SD card is transmitted through Bluetooth and stored in master’s SD card which now has both the IMU data from master and slave.

MASTER CODE:

#include<Wire.h>

#include <SPI.h>

#include <SD.h>

const uint8\_t chipSelect = 10;

String yep,s;

char ch,cx,cy[80];

long t1=0,t2=0;

uint16\_t i=0,k=0,l=0,m=0,n=0,count;

uint16\_t accelx,accely,accelz;

float gforcex,gforcey,gforcez;

uint16\_t gyrox,gyroy,gyroz;

float rotx,roty,rotz;

void setup()

{pinMode(2, INPUT);

Serial.begin(38400);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

Serial.print("Initializing XD card...");

// see if the card is present and can be initialized:

if (!SD.begin(chipSelect)) {

Serial.println("Card failed, or not present");

// don't do anything more:

while (1);

}

Serial.println("card initialized.");

SD.remove("aster.txt");

SD.remove("master.txt"); // remove is a function in SD library to delete a file

if (!SD.exists("master.txt")) //If the file still exist display message exist

Serial.println("The zest");

Wire.begin();

setupMPU();

}

void loop()

{

if(millis()<10000)

{

recordaccelregisters();

recordgyroregisters();

printdata();

//serial\_srew(); just checking

}

else if(k==0)

{

serial\_srew();

// sd\_screw();}

}}

void serial\_srew()

{

if(digitalRead(2))

{ k++;

Serial.write('a');

//{myfile.print("slave");

Serial.println("slave");

while(Serial.available())

{

cx=Serial.read();

l++;

delayMicroseconds(50);

if(l>=64)

break;

//while(Serial.read()=='\n')

//delayMicroseconds(500);

/\* ch=Serial.read();

myfile.print(ch);

delayMicroseconds(50);

//myfile.print("chaa");

Serial.write(ch);

count=0;

while(Serial.available()==0){

delayMicroseconds(1000);

//count+=1;

}\*/

}

while(Serial.available()==0)

{ //Serial.print("kee");

delayMicroseconds(50);

//count+=1;

}

while(Serial.available())

{

ch=Serial.read();

Serial.write(ch);

s+=ch;

// Serial.print(s);

if(s.length()>=160)

{sd\_screw();

// Serial.print("Heeeyyyyy");

// Serial.println(s.length());

}

//cy[m]=ch;

//m++;

//if(ch=='\n') //if(m>70)

//{

// for(int n=0;n<m;n++)

// s+=cy[n];

// m=0;

// myfile.print(ch);

// delayMicroseconds(50);

//myfile.print("chaa");

count=0;

t1=millis();

while(Serial.available()==0)

{delayMicroseconds(10);

//Serial.print("yooo");

t2=millis();

if(t2-t1>3000)

{

if(!(s.length())==0)

sd\_screw();

break;

}

//count+=1;

}}

Serial.println(s.length());

/\* Serial.print("m=");

Serial.println(m);

Serial.print("n=");

Serial.println(n); \*/

delayMicroseconds(1000);

Serial.print("Screwed it");

File xfile = SD.open("master.txt", FILE\_READ);

if(xfile)

{//delayMicroseconds(1000);

while(xfile.available())

{ Serial.write(xfile.read());

//delayMicroseconds(600);

}}

else

Serial.println("OOPs xxxxfile not able to open lave");

xfile.close();

}

}

void sd\_screw()

{n++;

File myfile = SD.open("master.txt", FILE\_WRITE);

if(myfile)

{ //for(int n=0;n<m;n++)

m++;

myfile.print(s);

//s="";

myfile.close();

}

else

Serial.print("Oops file didnt open master");

s.remove(0);

}

// if the file isn't open, pop up an error:

void setupMPU()

{

Wire.beginTransmission(0b1101000);//68 ad0 pull down to zero

Wire.write(0x6B);

Wire.write(0b00000000);

Wire.endTransmission();

Wire.beginTransmission(0b1101000);

Wire.write(0x1B);//accessing the register of gyrosope configuration

Wire.write(0x000000000); // seting the gyro to full scale +/- 250

Wire.endTransmission();

Wire.beginTransmission(0b1101000);

Wire.write(0x1C); //accessing the register 1c of acc configuration

Wire.write(0b00000000); //setting theaccel to +/-2g

Wire.endTransmission();

}

void recordaccelregisters()

{

Wire.beginTransmission(0b1101000);

Wire.write(0x3B);//starting register for accel readings

Wire.endTransmission();

Wire.requestFrom(0b1101000,6); //request accelregisters(3B-40)

while(Wire.available()<6);

accelx=Wire.read()<<8|Wire.read();//store first 2 bytes into x

accely=Wire.read()<<8|Wire.read();//store nxt 2 bytes into y

accelz=Wire.read()<<8|Wire.read();//store nxt 2 bytes into z

processacceldata();

}

void processacceldata()

{

gforcex=accelx/16384.0;

gforcey=accely/16384.0;

gforcez=accelz/16384.0;

}

void recordgyroregisters()

{

Wire.beginTransmission(0b1101000);

Wire.write(0x43); //starting register for gyro reading

Wire.endTransmission();

Wire.requestFrom(0b1101000,6);

while(Wire.available()<6);

gyrox=Wire.read()<<8|Wire.read();

gyroy=Wire.read()<<8|Wire.read();

gyroz=Wire.read()<<8|Wire.read();

processgyrodata();

}

void processgyrodata()

{

rotx=gyrox/131.0;

roty=gyroy/131.0;

rotz=gyroz/131.0;

}

void printdata()

{i++;

yep=String(i)+String('.')+String("gyro")+String('\t')+String("x=")+String(rotx)+ String('\t')+String("y=")+String(roty)+String('\t')+String("z=")+ String(rotz)+ String('\t')+ String("Accel(g)")+String('\t')+String("x=")+String(gforcex)+String('\t')+String("y=")+ String(gforcey)+ String('\t')+String("z=")+String(gforcez);

// yep=String(i)+String('\t')+String(rotx)+String(roty)+ String(rotz)+ String('\t')+String(gforcex)+ String(gforcey)+String(gforcez);

// Serial.print(i);Serial.print('\t')+Serial.print(rotx)+Serial.print(roty)+ Serial.print(rotz)+ Serial.print('\t')+Serial.print(gforcex)+ Serial.print(gforcey)+Serial.println(gforcez);

File dataFile = SD.open("master.txt", FILE\_WRITE);

if (dataFile) {

// dataFile.println();

Serial.println(yep);

// Serial.print(i);Serial.print('\t');Serial.print(rotx);Serial.print(roty); Serial.print(rotz); Serial.print('\t');Serial.print(gforcex); Serial.print(gforcey);Serial.println(gforcez);

// dataFile.print(i);dataFile.print('\t');dataFile.print(rotx);dataFile.print(roty); dataFile.print(rotz); dataFile.print('\t');dataFile.print(gforcex);dataFile.print(gforcey);dataFile.println(gforcez);

dataFile.println(yep);

// print to the serial port too:

}

else {

Serial.println("error opening datalog.txt");

} // yep=String(i)+String('.')+String("gyro")String(rotx)String(roty)+String('\t')+String("z=")+ String(rotz)+ String('\n');//+ String("Accel(g)")+String('\t')+String("x=")+String(gforcex)+String('\t')+String("y=")+ String(gforcey)+ String('\t')+String("z=")+String(gforcez);

dataFile.close();

}

SLAVE CODE:

#include<Wire.h>

#include <SPI.h>

#include <SD.h>

const uint8\_t chipSelect = 10;

String yep;

uint16\_t i=0,k=0;

uint16\_t accelx,accely,accelz;

float gforcex,gforcey,gforcez;

uint16\_t gyrox,gyroy,gyroz;

float rotx,roty,rotz;

void setup()

{

Serial.begin(38400);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

Serial.print("Initializing SD card...");

// see if the card is present and can be initialized:

if (!SD.begin(chipSelect)) {

Serial.println("Card failed, or not present");

// don't do anything more:

while (1);

}

Serial.println("card initialized.");

SD.remove("slave.txt"); // remove is a function in SD library to delete a file

if (!SD.exists("slave.txt")) //If the file still exist display message exist

Serial.println("The test");

Wire.begin();

setupMPU();

}

void loop()

{

if(millis()<10000)

{

recordaccelregisters();

recordgyroregisters();

printdata();

// serial\_check(); just to be sure it works properly then run it in infinite time

}

else if(millis()>1500&&k==0)

{serial\_check();

delay(50);}

}

void serial\_check()

{ if(Serial.read()=='a')

{ k++;

File myfile = SD.open("slave.txt", FILE\_READ);

if(myfile)

{//delayMicroseconds(1000);

while(myfile.available())

{ Serial.write(myfile.read());

delayMicroseconds(900);

}}

else

Serial.println("OOPs file not able to open Slave");

myfile.close();

// if the file isn't open, pop up an error:

}

else

{

//Serial.println("no problem"); cos its printing values in serial monitor of master even if its not displayed in serial monitor of slave

delay(50);

}

}

void setupMPU()

{

Wire.beginTransmission(0b1101000);//68 ad0 pull down to zero

Wire.write(0x6B);

Wire.write(0b00000000);

Wire.endTransmission();

Wire.beginTransmission(0b1101000);

Wire.write(0x1B);//accessing the register of gyrosope configuration

Wire.write(0x000000000); // seting the gyro to full scale +/- 250

Wire.endTransmission();

Wire.beginTransmission(0b1101000);

Wire.write(0x1C); //accessing the register 1c of acc configuration

Wire.write(0b00000000); //setting theaccel to +/-2g

Wire.endTransmission();

}

void recordaccelregisters()

{

Wire.beginTransmission(0b1101000);

Wire.write(0x3B);//starting register for accel readings

Wire.endTransmission();

Wire.requestFrom(0b1101000,6); //request accelregisters(3B-40)

while(Wire.available()<6);

accelx=Wire.read()<<8|Wire.read();//store first 2 bytes into x

accely=Wire.read()<<8|Wire.read();//store nxt 2 bytes into y

accelz=Wire.read()<<8|Wire.read();//store nxt 2 bytes into z

processacceldata();

}

void processacceldata()

{

gforcex=accelx/16384.0;

gforcey=accely/16384.0;

gforcez=accelz/16384.0;

}

void recordgyroregisters()

{

Wire.beginTransmission(0b1101000);

Wire.write(0x43); //starting register for gyro reading

Wire.endTransmission();

Wire.requestFrom(0b1101000,6);

while(Wire.available()<6);

gyrox=Wire.read()<<8|Wire.read();

gyroy=Wire.read()<<8|Wire.read();

gyroz=Wire.read()<<8|Wire.read();

processgyrodata();

}

void processgyrodata()

{

rotx=gyrox/131.0;

roty=gyroy/131.0;

rotz=gyroz/131.0;

}

void printdata()

{i++;

yep=String(i)+String('.')+String("gyro")+String('\t')+String("x=")+String(rotx)+ String('\t')+String("y=")+String(roty)+String('\t')+String("z=")+ String(rotz)+ String('\t')+ String("Accel(g)")+String('\t')+String("x=")+String(gforcex)+String('\t')+String("y=")+ String(gforcey)+ String('\t')+String("z=")+String(gforcez);

// yep=String(i)+String('\t')+String(rotx)+String(roty)+ String(rotz)+ String('\t')+String(gforcex)+ String(gforcey)+String(gforcez);

// Serial.print(i);Serial.print('\t')+Serial.print(rotx)+Serial.print(roty)+ Serial.print(rotz)+ Serial.print('\t')+Serial.print(gforcex)+ Serial.print(gforcey)+Serial.println(gforcez);

File dataFile = SD.open("slave.txt", FILE\_WRITE);

if (dataFile) {

// dataFile.println();

Serial.println(yep);

// Serial.print(i);Serial.print('\t');Serial.print(rotx);Serial.print(roty); Serial.print(rotz); Serial.print('\t');Serial.print(gforcex); Serial.print(gforcey);Serial.println(gforcez);

// dataFile.print(i);dataFile.print('\t');dataFile.print(rotx);dataFile.print(roty); dataFile.print(rotz); dataFile.print('\t');dataFile.print(gforcex);dataFile.print(gforcey);dataFile.println(gforcez);

dataFile.println(yep);

// print to the serial port too:

}

else {

Serial.println("error opening datalog.txt");

} // yep=String(i)+String('.')+String("gyro")String(rotx)String(roty)+String('\t')+String("z=")+ String(rotz)+ String('\n');//+ String("Accel(g)")+String('\t')+String("x=")+String(gforcex)+String('\t')+String("y=")+ String(gforcey)+ String('\t')+String("z=")+String(gforcez);

dataFile.close();

}

**Conclusion**

The raw data values- accelerometer and gyroscope was collected from two Sen14001, placed at Thigh and Shank of the leg and the Euler angles were calculated. Kalman Filters were implemented for smoothening the angles and these values were imported to python terminal through pyserial and further calculations were made to find the Knee Varus angle in real time and displayed in a Graphical Interface.

**References**

1. <https://lermagazine.com/article/the-role-of-varus-thrust-in-knee-osteoarthritis>

2. <https://ieeexplore.ieee.org/document/6091743>

3. <https://learn.sparkfun.com/tutorials/9dof-razor-imu-m0-hookup-guide?_ga=2.232555167.1286290895.1526376730-1382921196.1526376730>

4. <https://howtomechatronics.com/tutorials/arduino/how-to-configure-pair-two-hc-05-bluetooth-module-master-slave-commands/>

5. <https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/>