AI Sensors & Virtual/ Augmented Reality Technologies Homework1

Member: LIUCHAOWEI-A0263758N

Member: SHI YUFEI-A0263756U

Instructor: Vincent C. Lee

Team No: No12

Content

1. Explanations for the solutions and approaches:

1.1. Explore the influence of data transmission rates:

There are totally four Baud rates that need to be set in this lab: (Arduino board, two CC2530 modules, python) You can try to set the Baud rate as listed in the table, write down what happens for the received signal, and overall explain the reason. Design the Baud rate that you think is most appropriate (not required to be one of the combinations shown in the table), explain the reason.

	#1	#2	#3	#4	#5	#6	#7
Arduino Board	38400	38400	38400	38400	38400	2400	115200
TransmitterCC2530 Linked with Arduino	38400	19200	57600	38400	38400	2400	115200
ReceiverCC2530 Linked with laptop	38400	19200	57600	57600	19200	2400	115200
Python for reading The port	38400	19200	57600	57600	19200	2400	115200

#1 Successfully transmit and receive the data， because the transmit speed match the receive speed.

#2 Unsuccessfully receive the data, and the data is garbled. Because the speed of the Arduino Boad do not match the Transmitter of CC2530**.** due to too low speed ,the data can not be successful transmit.

#3 Unsuccessfully receive the data, and the data is garbled. Because the speed of the Arduino Boad do not match the Transmitter of CC2530**.** due to too low speed ,the data can not be successful transmit.

#4 Unsuccessfully receive the data, and the data is garbled. Because the speed of CC2530 transmitter do not match the Receiver of CC2530.

#5 Unsuccessfully receive the data, and the data is garbled. Because the speed of CC2530 transmitter do not match the Receiver of CC2530**,**

#6 Successfully transmit and receive the data， because the transmit speed match the receive speed.

#7 Successfully transmit and receive the data， because the transmit speed match the receive speed.

1.1.2 overall explain the reason:

Aim to successful transmitting, we just need to match the speed of the transmitter and receiver.And because of the exist of latency (100ms/patch) we defined in the Arduino, there is no obvious gap in different baud rates. The aim we set the latency in Arduino code is to make sure enough time to process the data to convert and transmission in case of the data loss.

However, the higher the baud rate is, the low quality of the transmitting. There are more possible to loss data during the data transmission when we choose the high baud rate.

1.1.3 Design the Baud rate that you think is most appropriate:

Due to exist of latency, we think the most appropriate Baud rate is 2400, because we have the latency of the code, so higher baud rate would not be faster than the Baud rate of 2400, but higher baud rate would make more data loss.

1.2. Transfer the obtained data to acutual acceleration and change the sensing range to ± 16g

a. Currently, the sense data, e.g., 12340, is not the actual acceleration value, transfer it to corresponding acceleration.

We inquire the datasheet of the MPU6050, we find that the MPU6050 have the Three-axis MEMS rate gyroscope sensor with 16-bit ADCs and signal conditioning Three-axis MEMS accelerometer sensor with 16-bit ADCs and signal conditioning. Because of the 16-bit ADCs, we can get the Range of data results between the -32767 and 32767.

So when we choose the sensing range : ±2g, and we get the sense data:12340, the actual acceleration value is equal to the 12340/32767 *2g=0.753g. when we choose the sensing range: ±16g, and we get the sense data:12340, the actual acceleration value is equal to the 12340/32767*16g=6.023 g. So we could get the following convert equation.

The actual acceleration = the sense data /32767 * the range of sense

b. Currently, the sensing range was set to ± 2g, while we can know from the data sheet that this inertial sensor have four different ranges: ± 2g, ± 4g, ± 8g, ± 16g. Change the sensing range to ± 16g.

We inquire the datasheet of the MPU6050, we find the ***setFullScaleAccelRange()*** function. We could choose the parameter of this function to set the sense range. And we inquire the datasheet, we find the following excel for the corresponding parameter identification conversion. For example, if we want to select the 16g range, we choose the MPU6050_ACCEL_FS_16 parameter.

corresponding parameter identification conversion for the acceleration range

The symbol	The data	The range
MPU6050_ACCEL_FS_2	0x00	± 2g
MPU6050_ACCEL_FS_4	0x01	± 4g
MPU6050_ACCEL_FS_8	0x02	± 8g
MPU6050_ACCEL_FS_16	0x03	± 16g

1.3. Use the sensed data to eliminate the influence of gravity.

1.3.1 Namely, when the acceleration is placed vertically, the obtained acceleration value on the z-axis should be zero.

If we keep the MPU6050 still and vertical don't turn it, we could just subtract 1 g from the Z-axis acceleration obtained by the sensor. And we could eliminate the influence of gravity.

1.3.2 We achieve a method to eliminate the influence of the gravity whatever the acceleration sensor is placed.

We think the normal and most simple way to eliminate the influence of gravity is to minus the gravitational acceleration (0.98) on the z-axis when the acceleration is placed vertically.

However, this solution can't deal with all conditions, for example the initial place is not vertical, and we turn the sensor to other direction and in above way, we can't just minus the gravitation acceleration to eliminate the influence of gravity. In this case, we need to solve this problem from three parts.

The First part, we need to get the translation angle from the initial position. To solve this problem, firstly, we need to get the angle speed of the 3 axis and Angle transfer time. The Euler angles equal to the angle speed* angle transfer time.

After we inquire the datasheet, we find that we can get the 3 axis angle speed data by the function: getRotation(),and we could set the rotation range of this function by the function: setFullScaleGyroRange(),we could choose the parameter by the following table.

corresponding parameter identification conversion for the angle speed range

The symbol	The data	The range
MPU6050_GYRO_FS_250	0x00	250
MPU6050_GYRO_FS_500	0x01	500
MPU6050_GYRO_FS_1000	0x02	1000
MPU6050_GYRO_FS_2000	0x03	2000

To get the angle transfer time, we could use the function: ***millis()*** to get the current time stamp. And we just need to calculate the difference between two times. We could get the angle transfer time. And then we calculate the product of two numbers we could get the yaw, roll, pitch, the Euler angles. And we need to get the rotation angles in three directions accumulated from the initial moment. So we just need to use the following formulas.

$$\begin{cases} yaw = yaw + yaw(t) \\ roll = roll + roll(t) \\ pitch = pitch + pitch(t) \end{cases}$$

In the second part, we need to decompose the gravity component of the sensor into a new rotated coordinate system to obtain the effect of the gravity component on the three axes in the new rotated coordinate system.

We need to get the Rotation Matrix from the Euler angles(yaw, pitch, roll) by the following equation.

$$R(\alpha,\beta,\gamma) = R_Z(\alpha) R_Y(\beta) R_X(\gamma)$$

$$= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

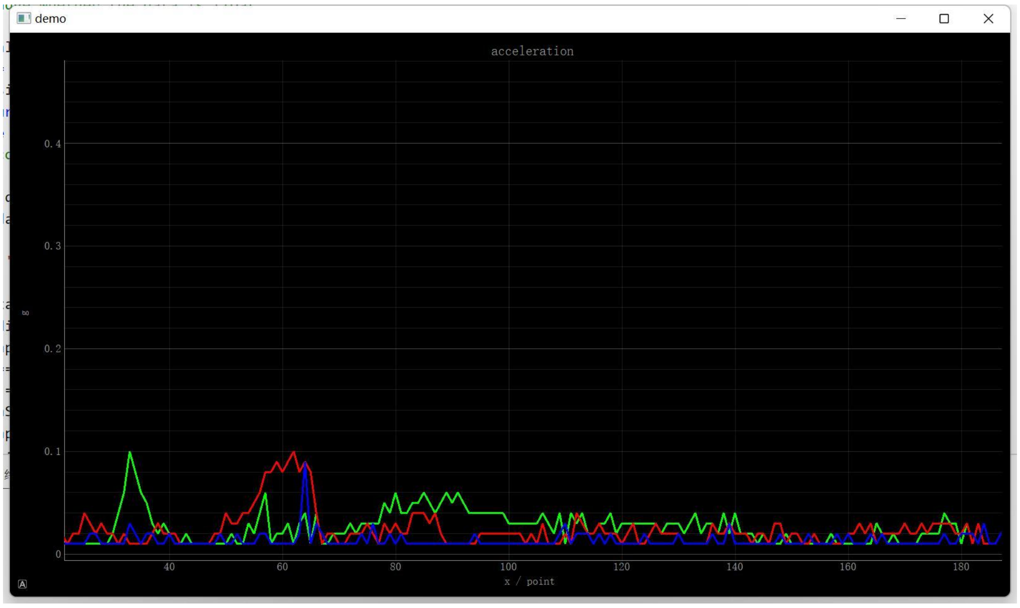
This rotation matrix is an orthogonal matrix. We could get the gravitational components P in the three directions in the coordinate system at the initial rest moment. And we just need to use the following formula. We could get the new gravitational components P1 in the coordinate after the rotation.

P1=R*P2

In the end part, we just need to make the current acceleration components in the three axis minus the current gravitational components and we could eliminate the influence of the gravitation. No matter what orientation the initial sensor is in, no matter what orientation the sensor is rotated to.

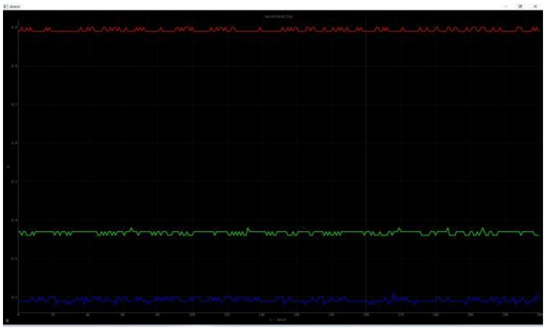
However, even if the sensor if still, the angle speed is not 0. There are some system hardware zero drift. We need to eliminate the influence of the zero drift. To solve these problems, we sample 10 times at the first time and calculate the mean of these ten angel speed data. And we just need to minus these mean in the following angel speed sampling point.

And we can get the following picture for the mpu6050 in the different direction. We can find that the 3 axis gravitation components are close to 0.

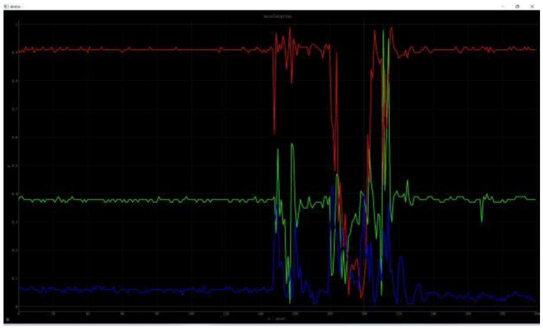


The three axis acceleration without the influence of the gravitation

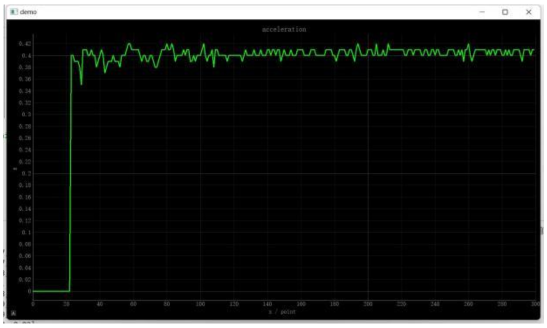
2. A screenshot/video to show the results/draw the sensed data separately:



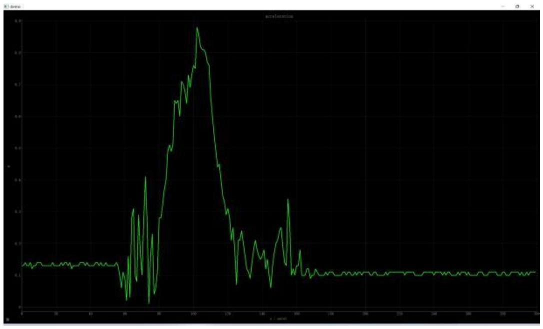
the three-axis accelerations at the same time in one figure with no motion



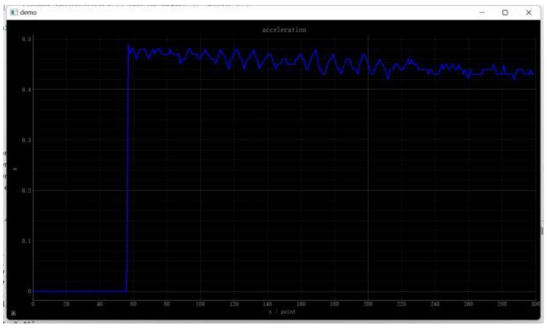
the three-axis accelerations at the same time in one figure with motion



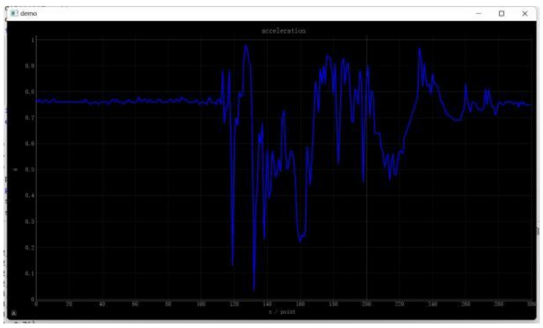
X-axis accelerations in one figure with no motion



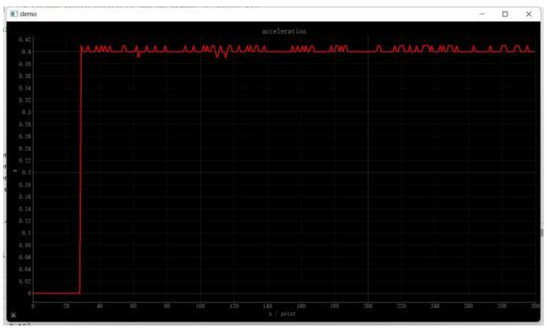
X-axis accelerations in one figure with motion



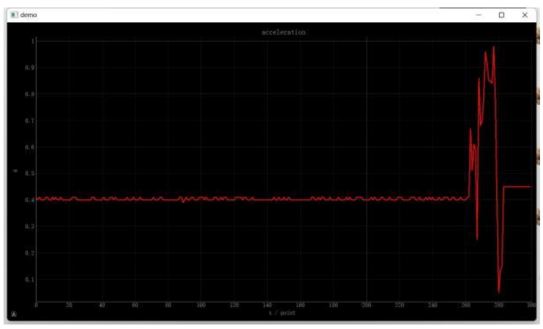
z-axis accelerations in one figure with no motion



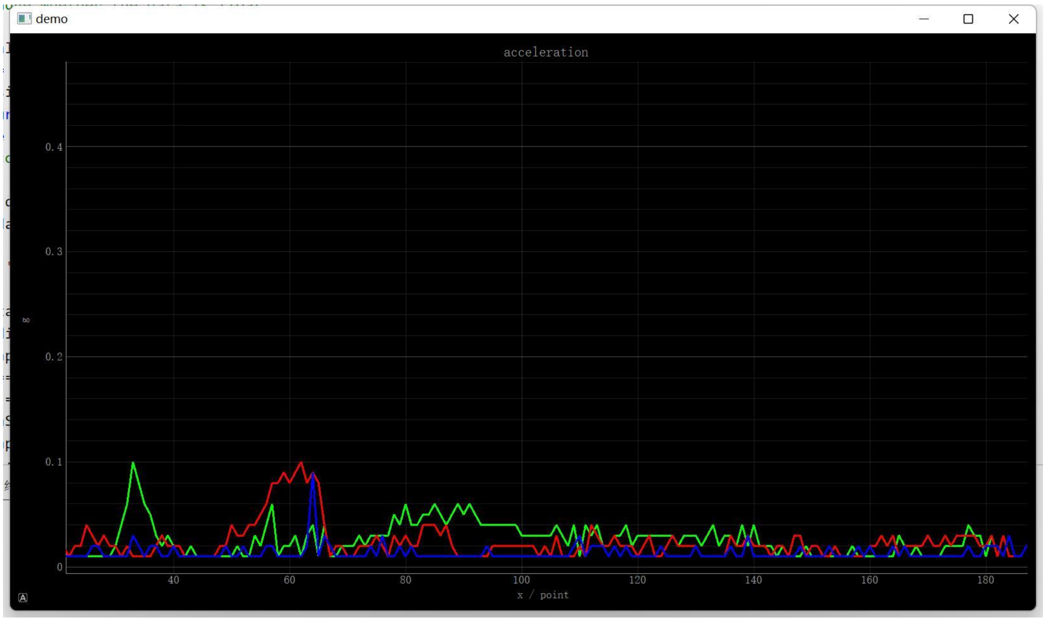
z-axis accelerations in one figure with motion



y-axis accelerations in one figure with no motion



y-axis accelerations in one figure with motion



Use the sensed data to eliminate the influence of gravity.

3. Our Expand work

We find that it also has some mistake when we Calculate the rotation angle of the coordinate axis by integrating the angular velocity because There is some noise here, so that even if we perform the initialization and zeroing operation, when we stop the sensor, there is still angular velocity, which eventually leads to the continuous occurrence of rotation angle. Over time, the accumulated error will cause the deviation to gradually increase.To solve the above problem, We made two improvements.

3.1 DMP library

Firstly, we choose to use the DMP Library. We find that the MPU6050 sensor has the special Digital Motion Processor which could directly get the Quaternion. Reduces workload on peripheral microprocessors and avoids tedious filtering and data fusion. For the question eliminate the influence of gravity, when we use the function from DMP Library, we could get the precise outcome about the quaternion. And we need to Convert Quaternion to Euler Angles. The conversion formula is as follows:

$$\begin{cases} pitch = \arcsin\left(2(q_0 * q_2 - q_1 * q_3)\right) \\ roll = \arctan\left(\dfrac{2(q_0 * q_1 + q_2 * q_3)}{{q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2}\right) \\ yaw = \arctan\left(\dfrac{2(q_0 * q_3 + q_1 * q_2)}{{q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2}\right) \end{cases}$$

After we get the Euler Angles, we rotate the coordinates, decompose the gravity coordinates into the new three direction axes, and subtract the decomposed value from the three-axis acceleration value at this time, the component of the gravity acceleration can be excluded. After our test, it can adapt to our normal Changes.

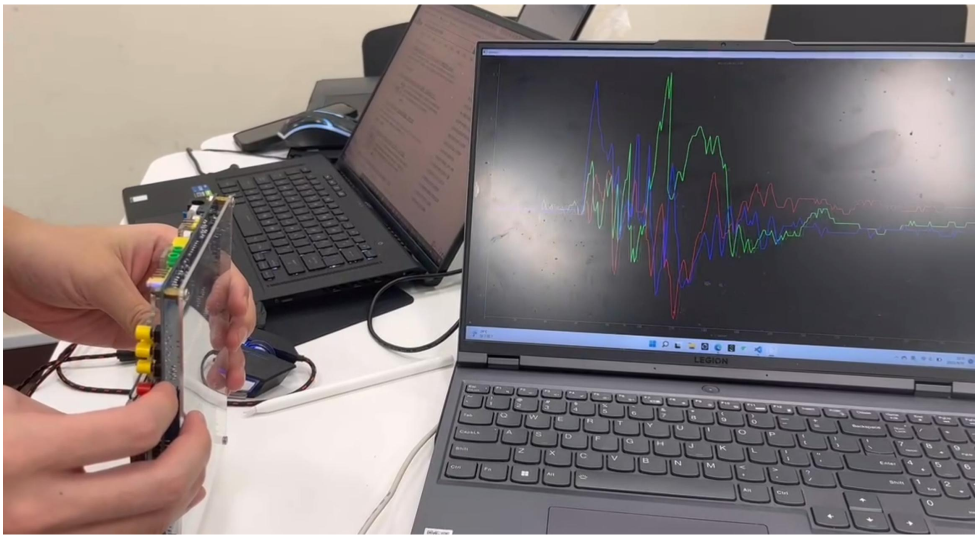
3.2 STM32F407 Embedded platform

Secondly, and to avoid the impression of the noisy and get the higher running speed, we test the whole project on the STM32F407 Embedded chip and its highest frequency can reach 168MHz, for the Arduino just the 16MHZ. Higher frequencies can support higher sampling rates, resulting in higher sensitivity and accuracy. And for our STM32F407, It solders a piece of MPU6050 in the peripheral circuit, and the superb soldering process makes it have higher anti-interference ability than the multi-state line connection. For the arm chip STM32F407, we will debug it using the Keil platform,
Download program via Flymcu software.

We could use the function: ***MPU_set_Accel_Fsr()*** to change the range of the acceleration. When we choose the different parameters, such as 0, we could get the different range, such as 2g.

We could use the function: ***MPU_Get_Accerometer()*** to get the 3 axis acceleration,
We could use the function: ***MPU_Get_Gyroscope()*** to get the 3 axis angle speed

3.3 our result



Use the sensed data to eliminate the influence of gravity.

There are more information in our video, you can click the following link to look it.
result.mp4
Root: ~/result.mp4

4. A statement about the contribution of the group member to each problem:

4.1 The contribution of LIU CHAOWEI:

1. Finishing python code including software environment, and some algorithm eg. How to determine the data is valid.
2. Designing the algorithm to deal with the problem of data loss during the Wi-Fi transmitting.
3. Achieving the data result shows separately in three figures and one figure.
4. Finishing report.

4.2 The contribution of SHI YUFEI:

1. Finishing Arduino code including hardware environment
2. Designing the algorithm to eliminate the influence of gravity
3. Achieving the range conversion
4. Achieving the conversion from the sensing data to the actual acceleration data.

5. Transplanting the entire embedded project from the arduino platform to the STM32F407 platform for testing
6. Finishing report

5.Screen shot with comments:

5.1 Python codes:

```
from signal import signal
import serial
import io
import pyqtgraph as pg
import array
import numpy as np

Data = serial.Serial()
Data.baudrate = 38400
Data.port = 'COM3'
Data.timeout =0.01
Data.open()
# sio  = io.TextIOWrapper(io.BufferedRWPair(Data,Data,1),encoding='ascii',nweline='\r')

# app=pg.mkQApp()
# win = pg.GraphicsWindow()
# win.setWindowTitle('demo')
# win.resize(1600,900)
# xLength = 300
# fig1 = win.addPlot()
# fig1.showGrid(x=True, y=True)
# fig1.setRange(xLength=[0,xLength],padding=0)
# fig1.setLabel(axis = 'left',text='g')
# fig1.setLabel(axis = 'bottom',text='x / point')
# fig1.setTitle('acceleration')

# curve1 = fig1.plot()
# curve2 = fig1.plot()
# curve3 = fig1.plot()

# data = [np.zeros(xLength).__array__('d'),
#          np.zeros(xLength).__array__('d'),
#          np.zeros(xLength).__array__('d')]


# def canFloat():
```

```
#     try:
#         float(data)
#         return True
#     except:
#         return False

# def dataProcess(data):
#     data = str(data)
#     dataSet = []
#     datapoint = ''
#     for i in data:
#         if i.isdigit() or i =='.':
#             datapoint +=1
#         elif i == "," and canFloat(datapoint):
#             dataSet.append(float(datapoint))
#             data = ''
#     return dataSet

# def plotData():
#     global signal
#     single = Data.readline()
#     singal = dataProcess(singal)
#     if(len(singal)==3):
#         for i in range(len(data)):
#             if len(data[i]) < xLength:
#                 data[i].append(singal[i])
#         else:
#             data[i][:-1] = data[i][1:]
#             data[i][-1] = signal[i]
#             curve1.setData(data[0],pen=pg.mkPen('g',width=3))
#             curve2.setData(data[1],pen=pg.mkPen('r',width=3))
#             curve3.setData(data[2],pen=pg.mkPen('b',width=3))

while True:
    print(Data)
    # signal = Data.readline()
    # print(signal)


timer = pg.QtCore.QTimer()
timer.timeout.connect(plotData)
timer.start(1)
app.exec()
```

4.2 Arduino codes:

```
// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#include "Wire.h"

// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"
#include "MPU6050.h"

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 accelgyro;

int16_t ax, ay, az;
int16_t x,y,z;
int16_t times=0;
float x1,y1,z1;
float  bx,by,bz,b1,b2,b3;
int16_t x3,y3,z3;
#define LED_PIN 13
bool blinkState = false;
float gravity[]={0,0,0.98};
float linear_acceleration[]={0,0,0};
float alpha=0.8;
float previousTime,currentTime,elapsedTime;
float gyroAngleX,gyroAngleY,gyroAngleZ;
float yaw,roll,pitch,roll2,yaw2,pitch2;
float accx_no_gra,accy_no_gra,accz_no_gra;
int16_t accx,accy,accz;
void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)

  Wire.begin();

  // initialize serial communication
  // (38400 chosen because it works as well at 8MHz as it does at 16MHz, but
```

```
    // it's really up to you depending on your project)
    Serial.begin(38400);

    // initialize device
    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();

    // verify connection
    Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

    // configure Arduino LED for
    pinMode(LED_PIN, OUTPUT);
    accx,accy,accz=set_argle_speed_zero();
    Serial.println(accx);
  }

  void loop() {
    // read raw accel/gyro measurements from device
    //accelgyro.setFullScaleAccelRange(11);  //we choose the range is 2g
    accelgyro.setFullScaleAccelRange(MPU6050_ACCEL_FS_16);  //MPU6050_ACCEL_FS_16 means that we choose the range is 16g
    accelgyro.setFullScaleGyroRange(MPU6050_GYRO_FS_250);  //Select the smallest range for angular velocity
    accelgyro.getAcceleration(&ax, &ay, &az);
    accelgyro.getRotation(&x, &y, &z);




    previousTime = currentTime;        // Previous time is stored before the actual time read
    currentTime = millis();
    elapsedTime = (currentTime - previousTime) / 1000; // Divide by 1000 to get seconds
    // display tab-separated accel x/y/z values
    bx = float(ax);
    by = float(ay);
    bz = float(az);
    bx = float(bx / 2048);
    by = float(by / 2048);
    bz = float(bz / 2048);
```

```

    if(times==0){
       b1=bx;
       b2=by;
       b3=bz;}
    times=times+1;


    // display tab-separated angles speed x/y/z values
    x1 = float(x);
    y1 = float(y);
    z1 = float(z);
    x1 = x1-accx;
    y1 = y1-accy;
    z1 = z1-accz;

    x1 = float(x1 / 131);
    y1 = float(y1 / 131);
    z1 = float(z1 / 131);

    // Correct the outputs with the calculated error values


    gyroAngleX = gyroAngleX + x1 * elapsedTime; // deg/s * s = deg
    gyroAngleY = gyroAngleY + y1 * elapsedTime;
    roll =  yaw + z1 * elapsedTime;
    // Complementary filter - combine acceleromter and gyro angle values
    pitch = 0.96 * gyroAngleX + 0.04 * bx;
    yaw = 0.96 * gyroAngleY + 0.04 * by;

   //convert to angle
  // roll2=roll/180*3.14;
  // pitch2=pitch/180*3.14;
  // yaw2=yaw/180*3.14;
   roll2=roll/57.3;
   pitch2=pitch/57.3;
   yaw2=yaw/57.3;
  //z rollα y yawβ x pitchγ
    accx_no_gra=bx-b3*(cos(roll2)*sin(yaw2)*cos(pitch2)+sin(yaw2)*sin(roll2))
                    -b1*(cos(roll2)*cos(yaw2))-b2*(cos(roll2)*sin(yaw2)*sin(pitch2)-sin(yaw2)*cos(roll2));
```

```

    accy_no_gra=by-b3*(sin(roll2)*sin(yaw2)*cos(pitch2)-cos(roll2)*sin(yaw2))
                             -b1*(sin(roll2)*cos(yaw2))-b2*(sin(roll2)*sin(yaw2)*sin(pitch2)-cos(yaw2)*cos(roll2));

    accz_no_gra=bz-b3*cos(yaw2)*cos(pitch2)+b1*sin(yaw2)-b2*cos(yaw2)*sin(pitch2);
//  gravity[0] = alpha * gravity[0] + (1 - alpha) * bx;
//  gravity[1] = alpha * gravity[1] + (1 - alpha) * by;
//  gravity[2] = alpha * gravity[2] + (1 - alpha) * bz;
//
//  linear_acceleration[0] = bx - gravity[0];
//  linear_acceleration[1] = by - gravity[1];
//  linear_acceleration[2] = bz - gravity[2];
  //  ax=float(ax/16384)
  //  ay=float(ay/16384)
  //  az=float(az/16384)
  //angle speed
//  Serial.print(x1);
//  Serial.print(",");
//  Serial.print(y1);
//  Serial.print(",");
//  Serial.print(z1);
//  Serial.print(",");

  //roll angle
//  Serial.print("/");
//  Serial.print(pitch);
//  Serial.print("/");
//  Serial.print(yaw);
//  Serial.print("/");
//  Serial.println(roll);
//  Serial.print("/");

  //acc
//  Serial.print(bx);
//  Serial.print(",");
//  Serial.print(by);
//  Serial.print(",");
//  Serial.print(bz);
//  Serial.print(",");

  //acc no gravitate
```

```
    Serial.print(accx_no_gra);
    Serial.print(",");
    Serial.print(accy_no_gra);
    Serial.print(",");
    Serial.print(accz_no_gra);
    Serial.print(",");
    Serial.println("");

    delay(100);
  //  Serial.print(bx);
  //  Serial.print(",");
  //  Serial.print(by);
  //  Serial.print(",");
  //  Serial.print(bz);
  //  Serial.print(",");
  //  Serial.println("");

    //The accuracy of the acceleration measurement is 16 bits, so it is also int16, so the range is: -32768~32767
    //Range selection if 16g
    //So the sensitivity is:32767/16 =  2048 g
    //The final calculation formula isACC_X  / 2048 g
    //The final calculation formula is 2g  ACC_X  / 16384 g
    // blink LED to indicate activity
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
  }

  // Calibrated angular velocity
  int16_t set_argle_speed_zero()
  {
    for(int i=0;i<10;i++)
    {
    accelgyro.getRotation(&x3, &y3, &z3);
    accx=accx+x3;
    accy=accy+y3;
    accz=accz+z3;
    }
    accx=accx/10;
    accy=accy/10;
    accz=accz/10;
    return accx,accy,accz;
  }
```

5.3 STM32F407 codes:

```
#include "sys.h"
#include "delay.h"
#include "usart.h"
#include "led.h"
#include "key.h"
#include "lcd.h"
#include "mpu6050.h"
#include "inv_mpu.h"
#include <math.h>
#include "inv_mpu_dmp_motion_driver.h"



//Serial port 1 sends 1 character
//c: character to send
void usart1_send_char(u8 c)
{

    while(USART_GetFlagStatus(USART1,USART_FLAG_TC)==RESET);
    USART_SendData(USART1,c);

}
//Transfer data to anonymous four-axis PC software
//fun: function word. 0XA0~0XAF
//data: data buffer, up to 28 bytes!!
//len: the number of valid data in the data area
void usart1_niming_report(u8 fun,u8*data,u8 len)
{
    u8 send_buf[32];
    u8 i;
    if(len>28)return;    //Up to 28 bytes of data
    send_buf[len+3]=0;  //Checksum is set to zero
    send_buf[0]=0X88;   //frame header
    send_buf[1]=fun;    //function word
    send_buf[2]=len;    //Data length
    for(i=0;i<len;i++)send_buf[3+i]=data[i];         //copy data

    for(i=0;i<len+3;i++)send_buf[len+3]+=send_buf[i];     //Calculate checksum
```

```
        for(i=0;i<len+4;i++)usart1_send_char(send_buf[i]);    // Send data to serial port 1
}
//Send accelerometer data and gyroscope data
//aacx,aacy,aacz:x,y,zAcceleration values in three directions
//gyrox,gyroy,gyroz:x,y,zGyroscope values in three directions
void mpu6050_send_data(short aacx,short aacy,short aacz,short gyrox,short gyroy,short gyroz)
{
    u8 tbuf[12];
    tbuf[0]=(aacx>>8)&0XFF;
    tbuf[1]=aacx&0XFF;
    tbuf[2]=(aacy>>8)&0XFF;
    tbuf[3]=aacy&0XFF;
    tbuf[4]=(aacz>>8)&0XFF;
    tbuf[5]=aacz&0XFF;
    tbuf[6]=(gyrox>>8)&0XFF;
    tbuf[7]=gyrox&0XFF;
    tbuf[8]=(gyroy>>8)&0XFF;
    tbuf[9]=gyroy&0XFF;
    tbuf[10]=(gyroz>>8)&0XFF;
    tbuf[11]=gyroz&0XFF;


    usart1_niming_report(0XA1,tbuf,12);//Custom Frame, 0XA1
}
//Report the posture data after settlement to the computer through serial port 1
//aacx,aacy,aacz:x,y,z Acceleration values in three directions
//gyrox,gyroy,gyroz:x,y,z Gyroscope values in three directions
//roll: Roll angle. The unit is 0.01 degrees. -18000 -> 18000 corresponds to -180.00 -> 180.00 degrees
//pitch: Pitch angle. The unit is 0.01 degrees. -9000 - 9000 corresponds to -90.00 -> 90.00 degrees
//yaw: heading angle. The unit is 0.1 degrees 0 -> 3600 corresponds to 0 -> 360.0 degrees
void usart1_report_imu(short aacx,short aacy,short aacz,short gyrox,short gyroy,short gyroz,short roll,short pitch,short yaw)
{
    u8 tbuf[28];
    u8 i;
    for(i=0;i<28;i++)tbuf[i]=0;//clear 0
    tbuf[0]=(aacx>>8)&0XFF;
    tbuf[1]=aacx&0XFF;
```

```
        tbuf[2]=(aacy>>8)&0XFF;
        tbuf[3]=aacy&0XFF;
        tbuf[4]=(aacz>>8)&0XFF;
        tbuf[5]=aacz&0XFF;
        tbuf[6]=(gyrox>>8)&0XFF;
        tbuf[7]=gyrox&0XFF;
        tbuf[8]=(gyroy>>8)&0XFF;
        tbuf[9]=gyroy&0XFF;
        tbuf[10]=(gyroz>>8)&0XFF;
        tbuf[11]=gyroz&0XFF;
        tbuf[18]=(roll>>8)&0XFF;
        tbuf[19]=roll&0XFF;
        tbuf[20]=(pitch>>8)&0XFF;
        tbuf[21]=pitch&0XFF;
        tbuf[22]=(yaw>>8)&0XFF;
        tbuf[23]=yaw&0XFF;
        usart1_niming_report(0XAF,tbuf,28);
    }

    int main(void)
    {
        u8 t=0,report=1;            //Reporting is enabled by default
        u8 key;
        int time=0;
        float pitch,roll,yaw;       //Euler angles
        short aacx,aacy,aacz;       //Accelerometer raw data
        short gyrox,gyroy,gyroz;    //Gyroscope raw data
        short temp;                       //temperature
        float x4,y4,z4,accx_no_gra,accy_no_gra,accz_no_gra;//Triaxial acceleration excluding the influence of gravity
        float b1,b2,b3;    //The acceleration value at the initial moment
        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//Set system interrupt priority group 2
        delay_init(168);          //Initialize delay function
        uart_init(38400);           //Initialize the serial port baud rate to 38400
        LED_Init();                 //Initialize the LEDs
        KEY_Init();                 //Initialize button
        MPU_Init();                 //Initialize MPU6050

        while(mpu_dmp_init())
        {
```

```
	
		delay_ms(200);
	
		delay_ms(200);
	}

	while(1)
	{
		key=KEY_Scan(0);
		if(key==KEY0_PRES)
		{
			report=!report;
			 if(report)LCD_ShowString(30,170,200,16,16,"UPLOAD ON");
			else LCD_ShowString(30,170,200,16,16,"UPLOAD OFF");
		}
		if(mpu_dmp_get_data(&pitch,&roll,&yaw)==0)
		{
			temp=MPU_Get_Temperature(); //get temperature value
			 MPU_Get_Accelerometer(&aacx,&aacy,&aacz);	//Get accelerometer data
			 MPU_Get_Gyroscope(&gyrox,&gyroy,&gyroz);	//get gyroscope data
			if(time==0)
		{
				MPU_Get_Accelerometer(&aacx,&aacy,&aacz);
				b1=(double)aacx/16384;
				b2=(double)aacy/16384;
				b3=(double)aacz/16384;
				time++;
			}
			 printf("%.2f,%.2f,%.2f,",(float)((float)aacx/16384),(float)((float)aacy/16384),(float)((float)aacz/16384));
			 //printf("gyrox %.3f,gyroy %.3f,gyroz %.3f ",((float)gyrox/16.384),((float)gyroy/16.384),((float)gyroz/16.384));
			//printf("roll %.3f,pitch %.3f,yaw %.3f\n",roll,pitch,yaw);
			//printf("\r\n\r\n");
			x4=roll/180*3.14;  //x4 pitch
			y4=pitch/180*3.14;  //z4 yaw
			z4=yaw/180*3.14;  // y4roll
			//printf("x4:%.3f cos(x4) %.3f ,b1 %.3f,b2 %.3f ,b3 %.3f",x4,cos(x4),b1,b2,b3);
			//printf("\r\n");
```

```
                accx_no_gra=(float)aacx/16384-(b1*(cos(y4)*cos(z4)-sin(x4)*sin(y4)*sin(z4))+b2*(cos(y4)*sin(z4)+sin(x4)*sin(y4)*cos(z4))+b3*(-sin(y4)*cos(x4)));
                accy_no_gra=(float)aacy/16384-(b1*(-cos(x4)*sin(z4))+b2*(cos(x4)*cos(z4))+b3*(sin(x4)));
                accz_no_gra=(float)aacz/16384-(b1*(sin(y4)*cos(z4)+sin(x4)*cos(y4)*sin(z4))+b2*(sin(y4)*sin(z4)-sin(x4)*cos(y4)*cos(z4))+b3*(cos(x4)*cos(y4)));
                //printf("%.2f,%.2f,%.2f,",accx_no_gra,accy_no_gra,accz_no_gra);
               printf("\r\n");

            }

        }
    }
```