# NUS - 首页

# AI Sensors & Virtual/ Augmented Reality Technologies Homework1

|  |  |
| --- | --- |
| Member: | LIUCHAOWEI-A0263758N |
| Member: | SHI YUFEI-A0263756U |
| Instructor: | Vincent C. Lee |
| Team No: | No12 |

Content

[s](#_Toc112676280)

[AI Sensors & Virtual/ Augmented Reality Technologies Homework1 1](#_Toc112676281)

[1. Explanations for the solutions and approaches: 3](#_Toc112676282)

[1.1. Explore the influence of data transmission rates: 3](#_Toc112676283)

[1.1.2 overall explain the reason: 4](#_Toc112676284)

[1.1.3 Design the Baud rate that you think is most appropriate: 4](#_Toc112676285)

[1.2. Transfer the obtained data to acutual acceleration and change the sensing range to ± 16g 5](#_Toc112676286)

[1.3. Use the sensed data to eliminate the influence of gravity. 6](#_Toc112676287)

[1.3.1 Namely, when the acceleration is placed vertically, the obtained acceleration value on the z-axis should be zero. 6](#_Toc112676288)

[1.3.2 We achieve a method to eliminate the influence of the gravity whatever the acceleration sensor is placed. 6](#_Toc112676289)

[2. A screenshot/video to show the results/draw the sensed data separately: 9](#_Toc112676290)

[3. A statement about the contribution of the group member to each problem: 10](#_Toc112676291)

[3.1 The contribution of LIU CHAOWEI: 10](#_Toc112676292)

[3.2 The contribution of SHI YUFEI: 10](#_Toc112676293)

[4. Screen shot with comments: 11](#_Toc112676294)

[4.1 Python codes: 11](#_Toc112676295)

[4.2 Arduino codes: 13](#_Toc112676296)

# 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 | 38400 | 19200 | 57600 | 38400 | 38400 | 2400 | 115200 |
| Linked with Arduino |
| ReceiverCC2530 | 38400 | 19200 | 57600 | 57600 | 19200 | 2400 | 115200 |
| Linked with laptop |
| Python for reading | 38400 | 19200 | 57600 | 57600 | 19200 | 2400 | 115200 |
| The port |

**#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.



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.



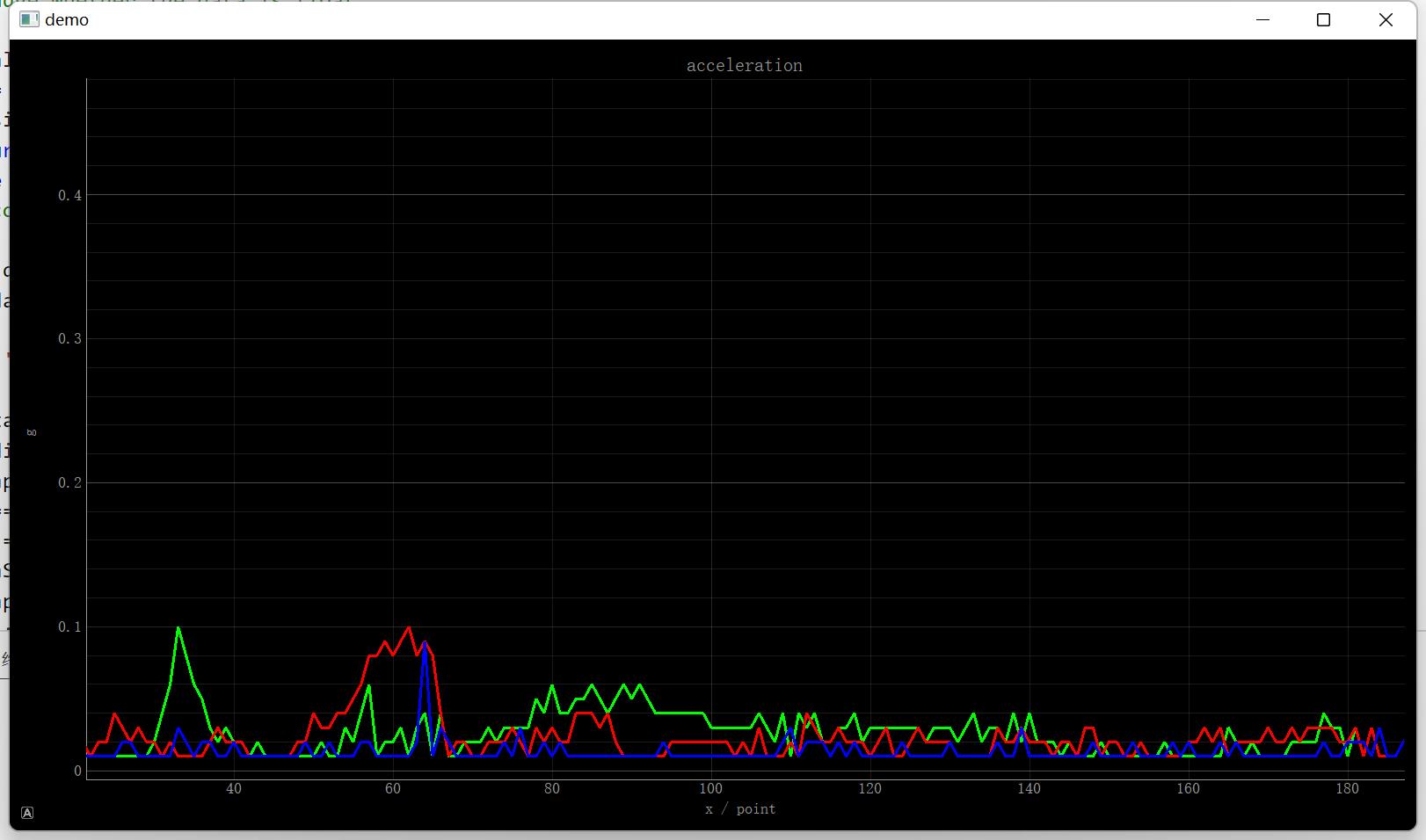
This rotation matrix is an orthogonal matrix. We could get the gravitational components P in the three directions in the coordinate system at the initial rest moment. And we just need to use the following formula. We could get the new gravitational components P1 in the coordinate after the rotation.

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

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

|  |  |
| --- | --- |
| aa5cef3c34f28abf4c86d683b74dddc | cd684567a5b4d7b8ced5407b1e71c82 |
| 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 |
| 6a5559d9f6a911201786eccc24e8fbc | 7d0bafd53c8c61723399d5dc569ce19 |
| X-axis accelerations in one figure with no motion | X-axis accelerations in one figure with motion |
| 65000fd6dc00511fe4400f28d9cccb9 | 18e1e659e76968dddf3583f794a2250 |
| z-axis accelerations in one figure with no motion | z-axis accelerations in one figure with motion |
| 7066caa3cdfaf29b38f0d4c6acc78a1 | d000cdbb15ecbb36a7b8c514ab38fb2 |
| y-axis accelerations in one figure with no motion | y-axis accelerations in one figure with motion |
| 043250f2a4da44dadbf6a2cd4a5735f | |
| Use the sensed data to eliminate the influence of gravity. | |

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

## 3.1 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.

## 3.2 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. Finishing report

# 4.Screen shot with comments:

## 4.1 Python codes:

1. from signal import signal
2. import serial
3. import io
4. import pyqtgraph as pg
5. import array
6. import numpy as np
7. Data = serial.Serial()
8. Data.baudrate = 38400
9. Data.port = 'COM3'
10. Data.timeout =0.01
11. Data.open()
12. *# sio  = io.TextIOWrapper(io.BufferedRWPair(Data,Data,1),encoding='ascii',nweline='\r')*
13. *# app=pg.mkQApp()*
14. *# win = pg.GraphicsWindow()*
15. *# win.setWindowTitle('demo')*
16. *# win.resize(1600,900)*
17. *# xLength = 300*
18. *# fig1 = win.addPlot()*
19. *# fig1.showGrid(x=True, y=True)*
20. *# fig1.setRange(xLength=[0,xLength],padding=0)*
21. *# fig1.setLabel(axis = 'left',text='g')*
22. *# fig1.setLabel(axis = 'bottom',text='x / point')*
23. *# fig1.setTitle('acceleration')*
24. *# curve1 = fig1.plot()*
25. *# curve2 = fig1.plot()*
26. *# curve3 = fig1.plot()*
27. *# data = [np.zeros(xLength).\_\_array\_\_('d'),*
28. *#         np.zeros(xLength).\_\_array\_\_('d'),*
29. *#         np.zeros(xLength).\_\_array\_\_('d')]*
30. *# def canFloat():*
31. *#     try:*
32. *#         float(data)*
33. *#         return True*
34. *#     except:*
35. *#         return False*
36. *# def dataProcess(data):*
37. *#     data = str(data)*
38. *#     dataSet = []*
39. *#     datapoint = ''*
40. *#     for i in data:*
41. *#         if i.isdigit() or i =='.':*
42. *#             datapoint +=1*
43. *#         elif i == "," and canFloat(datapoint):*
44. *#             dataSet.append(float(datapoint))*
45. *#             data = ''*
46. *#     return dataSet*
47. *# def plotData():*
48. *#     global signal*
49. *#     single = Data.readline()*
50. *#     singal = dataProcess(singal)*
51. *#     if(len(singal)==3):*
52. *#         for i in range(len(data)):*
53. *#             if len(data[i]) < xLength:*
54. *#                 data[i].append(singal[i])*
55. *#         else:*
56. *#             data[i][:-1] = data[i][1:]*
57. *#             data[i][-1] = signal[i]*
58. *#             curve1.setData(data[0],pen=pg.mkPen('g',width=3))*
59. *#             curve2.setData(data[1],pen=pg.mkPen('r',width=3))*
60. *#             curve3.setData(data[2],pen=pg.mkPen('b',width=3))*
61. while True:
62. print(Data)
63. *# signal = Data.readline()*
64. *# print(signal)*
65. timer = pg.QtCore.QTimer()
66. timer.timeout.connect(plotData)
67. timer.start(1)
68. app.exec()

## 4.2 Arduino codes:

1. *// Arduino Wire library is required if I2Cdev I2CDEV\_ARDUINO\_WIRE implementation*
2. *// is used in I2Cdev.h*
3. #include "Wire.h"
5. *// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files*
6. *// for both classes must be in the include path of your project*
7. #include "I2Cdev.h"
8. #include "MPU6050.h"
10. *// class default I2C address is 0x68*
11. *// specific I2C addresses may be passed as a parameter here*
12. *// AD0 low = 0x68 (default for InvenSense evaluation board)*
13. *// AD0 high = 0x69*
14. MPU6050 accelgyro;
16. int16\_t ax, ay, az;
17. int16\_t x,y,z;
18. int16\_t times=0;
19. float x1,y1,z1;
20. float  bx,by,bz,b1,b2,b3;
21. int16\_t x3,y3,z3;
22. #define LED\_PIN 13
23. bool blinkState = false;
24. float gravity[]={0,0,0.98};
25. float linear\_acceleration[]={0,0,0};
26. float alpha=0.8;
27. float previousTime,currentTime,elapsedTime;
28. float gyroAngleX,gyroAngleY,gyroAngleZ;
29. float yaw,roll,pitch,roll2,yaw2,pitch2;
30. float accx\_no\_gra,accy\_no\_gra,accz\_no\_gra;
31. int16\_t accx,accy,accz;
32. void setup() {
33. *// join I2C bus (I2Cdev library doesn't do this automatically)*
34. Wire.begin();
36. *// initialize serial communication*
37. *// (38400 chosen because it works as well at 8MHz as it does at 16MHz, but*
38. *// it's really up to you depending on your project)*
39. Serial.begin(38400);
41. *// initialize device*
42. Serial.println("Initializing I2C devices...");
43. accelgyro.initialize();
45. *// verify connection*
46. Serial.println("Testing device connections...");
47. Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");
49. *// configure Arduino LED for*
50. pinMode(LED\_PIN, OUTPUT);
51. accx,accy,accz=set\_argle\_speed\_zero();
52. Serial.println(accx);
53. }
55. void loop() {
56. *// read raw accel/gyro measurements from device*
57. *//accelgyro.setFullScaleAccelRange(11);  //we choose the range is 2g*
58. accelgyro.setFullScaleAccelRange(MPU6050\_ACCEL\_FS\_16);  *//MPU6050\_ACCEL\_FS\_16 means that we choose the range is 16g*
59. accelgyro.setFullScaleGyroRange(MPU6050\_GYRO\_FS\_250);  *//Select the smallest range for angular velocity*
60. accelgyro.getAcceleration(&ax, &ay, &az);
61. accelgyro.getRotation(&x, &y, &z);



66. previousTime = currentTime;        *// Previous time is stored before the actual time read*
67. currentTime = millis();
68. elapsedTime = (currentTime - previousTime) / 1000; *// Divide by 1000 to get seconds*
69. *// display tab-separated accel x/y/z values*
70. bx = float(ax);
71. by = float(ay);
72. bz = float(az);
73. bx = float(bx / 2048);
74. by = float(by / 2048);
75. bz = float(bz / 2048);
77. if(times==0){
78. b1=bx;
79. b2=by;
80. b3=bz;}
81. times=times+1;

84. *// display tab-separated angles speed x/y/z values*
85. x1 = float(x);
86. y1 = float(y);
87. z1 = float(z);
88. x1 = x1-accx;
89. y1 = y1-accy;
90. z1 = z1-accz;
92. x1 = float(x1 / 131);
93. y1 = float(y1 / 131);
94. z1 = float(z1 / 131);
96. *// Correct the outputs with the calculated error values*

99. gyroAngleX = gyroAngleX + x1 \* elapsedTime; *// deg/s \* s = deg*
100. gyroAngleY = gyroAngleY + y1 \* elapsedTime;
101. roll =  yaw + z1 \* elapsedTime;
102. *// Complementary filter - combine acceleromter and gyro angle values*
103. pitch = 0.96 \* gyroAngleX + 0.04 \* bx;
104. yaw = 0.96 \* gyroAngleY + 0.04 \* by;
106. *//convert to angle*
107. *// roll2=roll/180\*3.14;*
108. *// pitch2=pitch/180\*3.14;*
109. *// yaw2=yaw/180\*3.14;*
110. roll2=roll/57.3;
111. pitch2=pitch/57.3;
112. yaw2=yaw/57.3;
113. *//z rollα y yawβ x pitchγ*
114. accx\_no\_gra=bx-b3\*(cos(roll2)\*sin(yaw2)\*cos(pitch2)+sin(yaw2)\*sin(roll2))
115. -b1\*(cos(roll2)\*cos(yaw2))-b2\*(cos(roll2)\*sin(yaw2)\*sin(pitch2)-sin(yaw2)\*cos(roll2));
117. accy\_no\_gra=by-b3\*(sin(roll2)\*sin(yaw2)\*cos(pitch2)-cos(roll2)\*sin(yaw2))
118. -b1\*(sin(roll2)\*cos(yaw2))-b2\*(sin(roll2)\*sin(yaw2)\*sin(pitch2)-cos(yaw2)\*cos(roll2));
120. accz\_no\_gra=bz-b3\*cos(yaw2)\*cos(pitch2)+b1\*sin(yaw2)-b2\*cos(yaw2)\*sin(pitch2);
121. *//  gravity[0] = alpha \* gravity[0] + (1 - alpha) \* bx;*
122. *//  gravity[1] = alpha \* gravity[1] + (1 - alpha) \* by;*
123. *//  gravity[2] = alpha \* gravity[2] + (1 - alpha) \* bz;*
124. *//*
125. *//  linear\_acceleration[0] = bx - gravity[0];*
126. *//  linear\_acceleration[1] = by - gravity[1];*
127. *//  linear\_acceleration[2] = bz - gravity[2];*
128. *//  ax=float(ax/16384)*
129. *//  ay=float(ay/16384)*
130. *//  az=float(az/16384)*
131. *//angle speed*
132. *//  Serial.print(x1);*
133. *//  Serial.print(",");*
134. *//  Serial.print(y1);*
135. *//  Serial.print(",");*
136. *//  Serial.print(z1);*
137. *//  Serial.print(",");*
139. *//roll angle*
140. *//  Serial.print("/");*
141. *//  Serial.print(pitch);*
142. *//  Serial.print("/");*
143. *//  Serial.print(yaw);*
144. *//  Serial.print("/");*
145. *//  Serial.println(roll);*
146. *//  Serial.print("/");*
148. *//acc*
149. *//  Serial.print(bx);*
150. *//  Serial.print(",");*
151. *//  Serial.print(by);*
152. *//  Serial.print(",");*
153. *//  Serial.print(bz);*
154. *//  Serial.print(",");*
156. *//acc no grativate*
157. Serial.print(accx\_no\_gra);
158. Serial.print(",");
159. Serial.print(accy\_no\_gra);
160. Serial.print(",");
161. Serial.print(accz\_no\_gra);
162. Serial.print(",");
163. Serial.println("");
165. delay(100);
166. *//  Serial.print(bx);*
167. *//  Serial.print(",");*
168. *//  Serial.print(by);*
169. *//  Serial.print(",");*
170. *//  Serial.print(bz);*
171. *//  Serial.print(",");*
172. *//  Serial.println("");*
174. *//The accuracy of the acceleration measurement is 16 bits, so it is also int16, so the range is: -32768~32767*
175. *//Range selection if 16g*
176. *//So the sensitivity is:32767/16 =  2048 g*
177. *//The final calculation formula isACC\_X  / 2048 g*
178. *//The final calculation formula is 2g  ACC\_X  / 16384 g*
179. *// blink LED to indicate activity*
180. blinkState = !blinkState;
181. digitalWrite(LED\_PIN, blinkState);
182. }
184. *// Calibrated angular velocity*
185. int16\_t set\_argle\_speed\_zero()
186. {
187. for(int i=0;i<10;i++)
188. {
189. accelgyro.getRotation(&x3, &y3, &z3);
190. accx=accx+x3;
191. accy=accy+y3;
192. accz=accz+z3;
193. }
194. accx=accx/10;
195. accy=accy/10;
196. accz=accz/10;
197. return accx,accy,accz;
198. }