```
01.
#include <Wire.h> // I2C library, gyroscope
02.
03. // Accelerometer ADXL345
04. #define ACC (0x53)    //ADXL345 ACC address
05. #define A_TO_READ (6)        //num of bytes we are going to read each time (two bytes for each axis)
06.
07.
08. // Gyroscope ITG3200
09. #define GYRO 0x68 // gyro address, binary = 11101000 when AD0 is connected to Vcc (see schematics of your breakout board)
10. #define G_SMPLRT_DIV 0x15
11. #define G_DLPF_FS 0x16
12. #define G_INT_CFG 0x17
13. #define G_PWR_MGM 0x3E
14.
15. #define G_TO_READ 8 // 2 bytes for each axis x, y, z
16.
17.
18. // offsets are chip specific.
19. int a_offx = 0;
20. int a_offy = 0;
21. int a_offz = 0;
22.
23. int g_offx = 0;
24. int g_offy = 0;
25. int g_offz = 0;
26. ////////////////////////
27.
28. ////////////////////////
29. char str[512];
30.
31. void initAcc() {
32.   //Turning on the ADXL345
33.   writeTo(ACC, 0x2D, 0);
34.   writeTo(ACC, 0x2D, 16);
35.   writeTo(ACC, 0x2D, 8);
36.   //by default the device is in +-2g range reading
37. }
38.
39. void getAccelerometerData(int* result) {
40.   int regAddress = 0x32;    //first axis-acceleration-data register on the ADXL345
41.   byte buff[A_TO_READ];
42.
43.   readFrom(ACC, regAddress, A_TO_READ, buff); //read the acceleration data from the ADXL345
```

```
44.
45.   //each axis reading comes in 10 bit resolution, ie 2 bytes.  Least Significat Byte first!!
46.   //thus we are converting both bytes in to one int
47.   result[0] = (((int)buff[1]) << 8) | buff[0] + a_offx;
48.   result[1] = (((int)buff[3]) << 8) | buff[2] + a_offy;
49.   result[2] = (((int)buff[5]) << 8) | buff[4] + a_offz;
50. }
51.
52. //initializes the gyroscope
53. void initGyro()
54. {
55.   /**************************************
56.   * ITG 3200
57.   * power management set to:
58.   * clock select = internal oscillator
59.   *     no reset, no sleep mode
60.   *   no standby mode
61.   * sample rate to = 125Hz
62.   * parameter to +/- 2000 degrees/sec
63.   * low pass filter = 5Hz
64.   * no interrupt
65.   **************************************/
66.   writeTo(GYRO, G_PWR_MGM, 0x00);
67.   writeTo(GYRO, G_SMPLRT_DIV, 0x07); // EB, 50, 80, 7F, DE, 23, 20, FF
68.   writeTo(GYRO, G_DLPF_FS, 0x1E); // +/- 2000 dgrs/sec, 1KHz, 1E, 19
69.   writeTo(GYRO, G_INT_CFG, 0x00);
70. }
71.
72.
73. void getGyroscopeData(int * result)
74. {
75.   /**********************************
76.   Gyro ITG-3200 I2C
77.   registers:
78.   temp MSB = 1B, temp LSB = 1C
79.   x axis MSB = 1D, x axis LSB = 1E
80.   y axis MSB = 1F, y axis LSB = 20
81.   z axis MSB = 21, z axis LSB = 22
82.   **********************************/
83.
84.   int regAddress = 0x1B;
85.   int temp, x, y, z;
86.   byte buff[G_TO_READ];
87.
88.   readFrom(GYRO, regAddress, G_TO_READ, buff); //read the gyro data from the ITG3200
89.
```

```
  result[0] = ((buff[2] << 8) | buff[3]) + g_offx;
  result[1] = ((buff[4] << 8) | buff[5]) + g_offy;
  result[2] = ((buff[6] << 8) | buff[7]) + g_offz;
  result[3] = (buff[0] << 8) | buff[1]; // temperature

}


float xz=0,yx=0,yz=0;
float p_xz=1,p_yx=1,p_yz=1;
float q_xz=0.0025,q_yx=0.0025,q_yz=0.0025;
float k_xz=0,k_yx=0,k_yz=0;
float r_xz=0.25,r_yx=0.25,r_yz=0.25;
  //int acc_temp[3];
  //float acc[3];
  int acc[3];
  int gyro[4];
  float Axz;
  float Ayx;
  float Ayz;
  float t=0.025;
void setup()
{
  Serial.begin(9600);
  Wire.begin();
  initAcc();
  initGyro();

}

//unsigned long timer = 0;
//float o;
void loop()
{

  getAccelerometerData(acc);
  getGyroscopeData(gyro);
  //timer = millis();
  sprintf(str, "%d,%d,%d,%d,%d,%d", acc[0],acc[1],acc[2],gyro[0],gyro[1],gyro[2]);

  //acc[0]=acc[0];
  //acc[2]=acc[2];
  //acc[1]=acc[1];
  //r=sqrt(acc[0]*acc[0]+acc[1]*acc[1]+acc[2]*acc[2]);
  gyro[0]=gyro[0]/ 14.375;
  gyro[1]=gyro[1]/ (-14.375);
```

```
  gyro[2]=gyro[2]/ 14.375;


  Axz=(atan2(acc[0],acc[2]))*180/PI;
  Ayx=(atan2(acc[0],acc[1]))*180/PI;
  /*if((acc[0]!=0)&&(acc[1]!=0))
    {
      Ayx=(atan2(acc[0],acc[1]))*180/PI;
    }
    else
    {
      Ayx=t*gyro[2];
    }*/
  Ayz=(atan2(acc[1],acc[2]))*180/PI;


//kalman filter
  calculate_xz();
  calculate_yx();
  calculate_yz();

  //sprintf(str, "%d,%d,%d", xz_1, xy_1, x_1);
  //Serial.print(xz);Serial.print(",");
  //Serial.print(yx);Serial.print(",");
  //Serial.print(yz);Serial.print(",");
  //sprintf(str, "%d,%d,%d,%d,%d,%d", acc[0],acc[1],acc[2],gyro[0],gyro[1],gyro[2]);
  //sprintf(str, "%d,%d,%d",gyro[0],gyro[1],gyro[2]);
    Serial.print(Axz);Serial.print(",");
    //Serial.print(Ayx);Serial.print(",");
    //Serial.print(Ayz);Serial.print(",");
  //Serial.print(str);
  //o=gyro[2];//w=acc[2];
  //Serial.print(o);Serial.print(",");
  //Serial.print(w);Serial.print(",");
  Serial.print("\n");


  //delay(50);
}
void calculate_xz()
{

xz=xz+t*gyro[1];
p_xz=p_xz+q_xz;
k_xz=p_xz/(p_xz+r_xz);
xz=xz+k_xz*(Axz-xz);
```

```
p_xz=(1-k_xz)*p_xz;
}
void calculate_yx()
{

  yx=yx+t*gyro[2];
  p_yx=p_yx+q_yx;
  k_yx=p_yx/(p_yx+r_yx);
  yx=yx+k_yx*(Ayx-yx);
  p_yx=(1-k_yx)*p_yx;

}
void calculate_yz()
{
  yz=yz+t*gyro[0];
  p_yz=p_yz+q_yz;
  k_yz=p_yz/(p_yz+r_yz);
  yz=yz+k_yz*(Ayz-yz);
  p_yz=(1-k_yz)*p_yz;

}


//---------------- Functions
//Writes val to address register on ACC
void writeTo(int DEVICE, byte address, byte val) {
   Wire.beginTransmission(DEVICE); //start transmission to ACC
   Wire.write(address);        // send register address
   Wire.write(val);        // send value to write
   Wire.endTransmission(); //end transmission
}


//reads num bytes starting from address register on ACC in to buff array
void readFrom(int DEVICE, byte address, int num, byte buff[]) {
  Wire.beginTransmission(DEVICE); //start transmission to ACC
  Wire.write(address);        //sends address to read from
  Wire.endTransmission(); //end transmission

  Wire.beginTransmission(DEVICE); //start transmission to ACC
  Wire.requestFrom(DEVICE, num);    // request 6 bytes from ACC

  int i = 0;
  while(Wire.available())    //ACC may send less than requested (abnormal)
  {
    buff[i] = Wire.read(); // receive a byte
```

```
228.     i++;
229.   }
230.   Wire.endTransmission(); //end transmission
231. }
```