//Gyro - Arduino UNO R3

//VCC - 5V

//GND - GND

//SDA - A4

//SCL - A5

//INT - port-2

#include <LiquidCrystal.h>

#include <Wire.h>

//Declaring some global variables

int gyro\_x, gyro\_y, gyro\_z;

long gyro\_x\_cal, gyro\_y\_cal, gyro\_z\_cal;

boolean set\_gyro\_angles;

long acc\_x, acc\_y, acc\_z, acc\_total\_vector;

float angle\_roll\_acc, angle\_pitch\_acc;

float angle\_pitch, angle\_roll;

int angle\_pitch\_buffer, angle\_roll\_buffer;

float angle\_pitch\_output, angle\_roll\_output;

long loop\_timer;

int temp;

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

Wire.begin(); //Start I2C as master

setup\_mpu\_6050\_registers(); //Setup the registers of the MPU-6050

for (int cal\_int = 0; cal\_int < 1000 ; cal\_int ++){ //Read the raw acc and gyro data from the MPU-6050 for 1000 times

read\_mpu\_6050\_data();

gyro\_x\_cal += gyro\_x; //Add the gyro x offset to the gyro\_x\_cal variable

gyro\_y\_cal += gyro\_y; //Add the gyro y offset to the gyro\_y\_cal variable

gyro\_z\_cal += gyro\_z; //Add the gyro z offset to the gyro\_z\_cal variable

delay(3); //Delay 3us to have 250Hz for-loop

}

// divide by 1000 to get avarage offset

gyro\_x\_cal /= 1000;

gyro\_y\_cal /= 1000;

gyro\_z\_cal /= 1000;

Serial.begin(115200);

loop\_timer = micros();

lcd.begin(16, 2);

lcd.print(" GONIO");

//Reset the loop timer

}

void loop(){

read\_mpu\_6050\_data();

//Subtract the offset values from the raw gyro values

gyro\_x -= gyro\_x\_cal;

gyro\_y -= gyro\_y\_cal;

gyro\_z -= gyro\_z\_cal;

//Gyro angle calculations . Note 0.0000611 = 1 / (250Hz x 65.5)

angle\_pitch += gyro\_x \* 0.0000611; //Calculate the traveled pitch angle and add this to the angle\_pitch variable

angle\_roll += gyro\_y \* 0.0000611; //Calculate the traveled roll angle and add this to the angle\_roll variable

//0.000001066 = 0.0000611 \* (3.142(PI) / 180degr) The Arduino sin function is in radians

angle\_pitch += angle\_roll \* sin(gyro\_z \* 0.000001066); //If the IMU has yawed transfer the roll angle to the pitch angel

angle\_roll -= angle\_pitch \* sin(gyro\_z \* 0.000001066); //If the IMU has yawed transfer the pitch angle to the roll angel

//Accelerometer angle calculations

acc\_total\_vector = sqrt((acc\_x\*acc\_x)+(acc\_y\*acc\_y)+(acc\_z\*acc\_z)); //Calculate the total accelerometer vector

//57.296 = 1 / (3.142 / 180) The Arduino asin function is in radians

angle\_pitch\_acc = asin((float)acc\_y/acc\_total\_vector)\* 57.296; //Calculate the pitch angle

angle\_roll\_acc = asin((float)acc\_x/acc\_total\_vector)\* -57.296; //Calculate the roll angle

angle\_pitch\_acc -= 0.0; //Accelerometer calibration value for pitch

angle\_roll\_acc -= 0.0; //Accelerometer calibration value for roll

if(set\_gyro\_angles){ //If the IMU is already started

angle\_pitch = angle\_pitch \* 0.9996 + angle\_pitch\_acc \* 0.0004; //Correct the drift of the gyro pitch angle with the accelerometer pitch angle

angle\_roll = angle\_roll \* 0.9996 + angle\_roll\_acc \* 0.0004; //Correct the drift of the gyro roll angle with the accelerometer roll angle

}

else{ //At first start

angle\_pitch = angle\_pitch\_acc; //Set the gyro pitch angle equal to the accelerometer pitch angle

angle\_roll = angle\_roll\_acc; //Set the gyro roll angle equal to the accelerometer roll angle

set\_gyro\_angles = true; //Set the IMU started flag

}

//To dampen the pitch and roll angles a complementary filter is used

angle\_pitch\_output = angle\_pitch\_output \* 0.9 + angle\_pitch \* 0.1; //Take 90% of the output pitch value and add 10% of the raw pitch value

angle\_roll\_output = angle\_roll\_output \* 0.9 + angle\_roll \* 0.1; //Take 90% of the output roll value and add 10% of the raw roll value

Serial.print(" | Angle = "); Serial.println(angle\_pitch\_output);

lcd.setCursor(1,1);

lcd.print("angle: " + String(angle\_pitch\_output) + " deg");

while(micros() - loop\_timer < 4000); //Wait until the loop\_timer reaches 4000us (250Hz) before starting the next loop

loop\_timer = micros();//Reset the loop timer

}

void setup\_mpu\_6050\_registers(){

//Activate the MPU-6050

Wire.beginTransmission(0x68); //Start communicating with the MPU-6050

Wire.write(0x6B); //Send the requested starting register

Wire.write(0x00); //Set the requested starting register

Wire.endTransmission();

//Configure the accelerometer (+/-8g)

Wire.beginTransmission(0x68); //Start communicating with the MPU-6050

Wire.write(0x1C); //Send the requested starting register

Wire.write(0x10); //Set the requested starting register

Wire.endTransmission();

//Configure the gyro (500dps full scale)

Wire.beginTransmission(0x68); //Start communicating with the MPU-6050

Wire.write(0x1B); //Send the requested starting register

Wire.write(0x08); //Set the requested starting register

Wire.endTransmission();

}

void read\_mpu\_6050\_data(){ //Subroutine for reading the raw gyro and accelerometer data

Wire.beginTransmission(0x68); //Start communicating with the MPU-6050

Wire.write(0x3B); //Send the requested starting register

Wire.endTransmission(); //End the transmission

Wire.requestFrom(0x68,14); //Request 14 bytes from the MPU-6050

while(Wire.available() < 14); //Wait until all the bytes are received

acc\_x = Wire.read()<<8|Wire.read();

acc\_y = Wire.read()<<8|Wire.read();

acc\_z = Wire.read()<<8|Wire.read();

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

gyro\_x = Wire.read()<<8|Wire.read();

gyro\_y = Wire.read()<<8|Wire.read();

gyro\_z = Wire.read()<<8|Wire.read();

}