**EMBEDDED SYSTEMS**

**(EL-419)**

# LABORATORY MANUAL

# Spring 2019



**Engr. Aneela Sabir**

## Interfacing Accelerometer/Gyro & Flex Sensors with Arduino Uno

**(LAB # 03)**

Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_**MOIZ(15i-0402),Aqib(15i-0443),Kamran(14i-0420)**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section: \_A\_\_\_

Date performed: \_\_\_\_**2/6**\_\_\_\_\_\_\_\_\_, 2019

**MARKS AWARDED: \_\_\_\_\_\_\_\_ / 10**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES, ISLAMABAD**

|  |  |  |  |
| --- | --- | --- | --- |
| Prepared by: | Engr. Ahsan Khan |  | Version: 1.0.2 |
| Last Edited by: | Engr. Aneela Sabir |  | February 5, 2019 |
| Verified by: | Dr. Durdana Habib |  | Updated: Spring 2019 |

**Lab # 03: Interfacing Accelerometer/Gyro and Flex Sensors with Arduino Uno**

#### Lab Objective:

The purpose of this lab is to interface different sensors with microcontrollers.

**Equipment Required:**

* Arduino Uno/Mbed LPC1768
* Accelerometer
* Flex Sensor
* LEDs
* Resistors
* Breadboard

**Introduction:**

#### Serial Communication with Arduino

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): **Serial**. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

#### Serial.begin() function

This function sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200. You can, however, specify other rates - for example, to communicate over pins 0 and 1 with a component that requires a particular baud rate.

An optional second argument configures the data, parity, and stop bits. The default is 8 data bits, no parity, one stop bit.

Its syntax is as follows:

Serial.begin(speed) Serial.begin(speed, config) where,

speed: in bits per second (baud) - *long* config: sets data, parity, and stop bits.

#### Serial.available() function

This function gets the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes). Its syntax is

Serial.available(), it takes no parameters and returns the number of bytes available to read.

#### Serial.read() function

This function reads the incoming serial data. Its syntax is serial.read()

It takes no parameters and returns the first byte of incoming serial data available (or -1 if no data is available).

#### Serial.write() function

This function writes binary data to the serial port. This data is sent as a byte or series of bytes. Its syntax is as follows:

Serial.write(val)

Serial.write(str) Serial.write(buf, len) where,

val: a value to send as a single byte str: a string to send as a series of bytes buf: an array to send as a series of bytes len: the length of the buffer

This function will return the number of bytes written, though reading that number is optional.

**Example 1**

void setup()

{

Serial.begin(9600);

}

void loop() {

Serial.write(45); // send a byte with the value 45. This will be ASCII converted. If you send 65,

//„A‟ will appear on Serial Monitor.

delay(5000);

int bytesSent = Serial.write(“hello”); //send the string “hello” and return the length of the string.

delay(5000);

}

#### Echoing with Arduino Serial Communication

Parts Required:

* Computer
* USB cable
* Arduino UNO
* Arduino IDE

The following code will make the Arduino ECHO anything you send to it. Therefore, if you type a 3, the Arduino will send back a 3. If you type a letter F, the Arduino will send back a letter F.

Enter the following code into your Arduino IDE and upload it to your Arduino.

#### Example 2

/\* Simple Serial ECHO script : Written by ScottC 03/07/2012 \*/

/\* Use a variable called byteRead to temporarily store

the data coming from the computer \*/

byte byteRead;

void setup() {

// Turn the Serial Protocol ON

Serial.begin(9600);

}

void loop() {

/\* check if data has been sent from the computer: \*/

if (Serial.available()) {

/\* read the most recent byte \*/

byteRead = Serial.read();

/\*ECHO the value that was read, back to the serial port. \*/

Serial.write(byteRead);

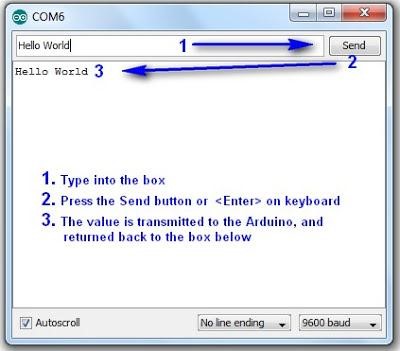
}

}

Once the Arduino sketch has been uploaded to the Arduino, open the Serial monitor, which looks like a magnifying glass at the top right section of the Arduino IDE. Please note, that you need to keep the USB connected to the Arduino during this process, as the USB cable is your communication link between your computer and the Arduino.

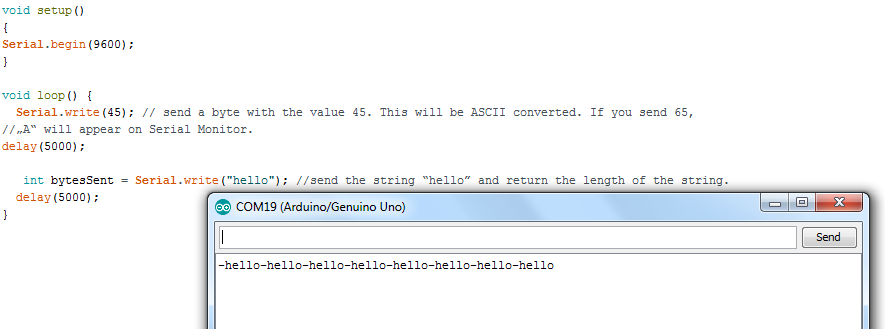


Type anything into the top box of the Serial Monitor and press <Enter> on your keyboard. This will send a series of bytes to the Arduino. The Arduino will respond by sending back your typed message in the larger textbox.



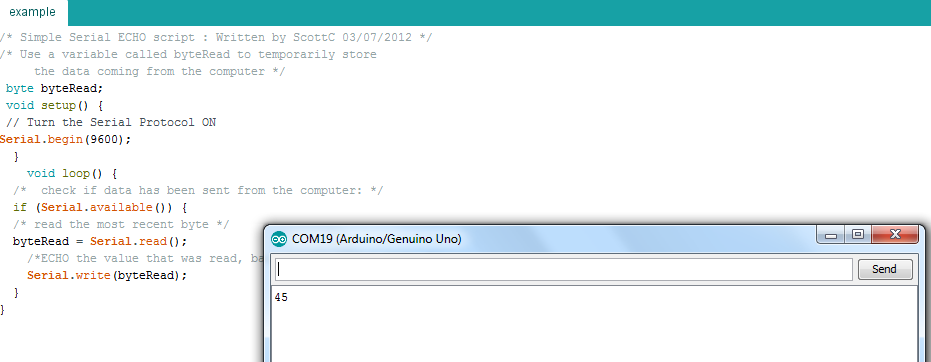
**Task 3.1**

**Implement Example 1 on the Arduino Serial Monitor and observe the output.**



**Task 3.2**

**Implement Example 2 on the Arduino Serial Monitor and observe the output.**



#### Triple Axis Accelerometer and Gyro Breakout - MPU-6050: SEN-11028

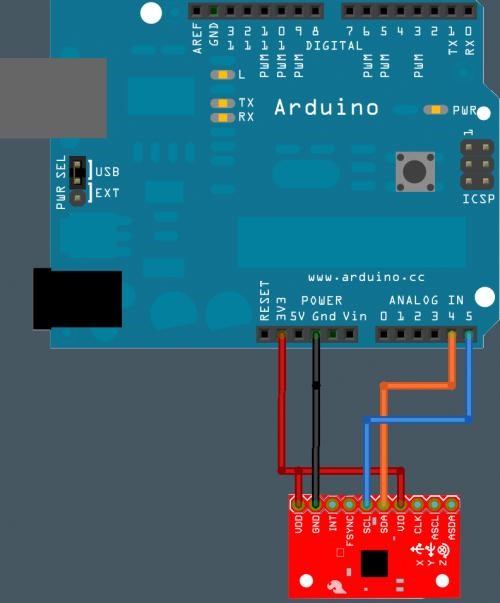
The MPU-6050 is a serious little piece of motion processing tech! By combining a MEMS 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor™ (DMP™) capable of processing complex 9-axis MotionFusion algorithms, the MPU-6050 does away with the cross-axis alignment problems that can creep up on discrete parts.

**``**

Its features are:

* I2C Digital-output of 6 or 9-axis MotionFusion data in rotation matrix, quaternion, Euler Angle, or raw data format
* Input Voltage: 2.3 - 3.4V
* Selectable Solder Jumpers on CLK, FSYNC and AD0
* Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps
* Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g  Digital Motion Processing™ (DMP™) engine offloads complex MotionFusion, sensor timing synchronization and gesture detection
* Embedded algorithms for run-time bias and compass calibration. No user intervention required
* Digital-output temperature sensor

**Interfacing- MPU-6050 with Arduino:**



#### Example 3

// MPU-6050 Short Example Sketch

// Arduino

// 2018

// Testing

#include<Wire.h>

const int MPU=0x68; // I2C address of the MPU-6050

int16\_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

void setup(){

Wire.begin();

Wire.beginTransmission(MPU);

Wire.write(0x6B); // PWR\_MGMT\_1 register

Wire.write(0); // set to zero (wakes up the MPU-6050)

Wire.endTransmission(true);

Serial.begin(9600);

}

void loop(){

Wire.beginTransmission(MPU);

Wire.write(0x3B); // starting with register 0x3B (ACCEL\_XOUT\_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU,14,true); // request a total of 14 registers

AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL\_XOUT\_H) & 0x3C (ACCEL\_XOUT\_L)

AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL\_YOUT\_H) & 0x3E (ACCEL\_YOUT\_L)

AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL\_ZOUT\_H) & 0x40 (ACCEL\_ZOUT\_L)

Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP\_OUT\_H) & 0x42 (TEMP\_OUT\_L)

GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO\_XOUT\_H) & 0x44 (GYRO\_XOUT\_L)

GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO\_YOUT\_H) & 0x46 (GYRO\_YOUT\_L)

GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO\_ZOUT\_H) & 0x48

(GYRO\_ZOUT\_L)

Serial.print("AcX = "); Serial.print(AcX);

Serial.print(" | AcY = "); Serial.print(AcY);

Serial.print(" | AcZ = "); Serial.print(AcZ);

Serial.print(" | Tmp = "); Serial.print(Tmp/340.00+36.53); //equation for temperature in degrees C from datasheet

Serial.print(" | GyX = "); Serial.print(GyX);

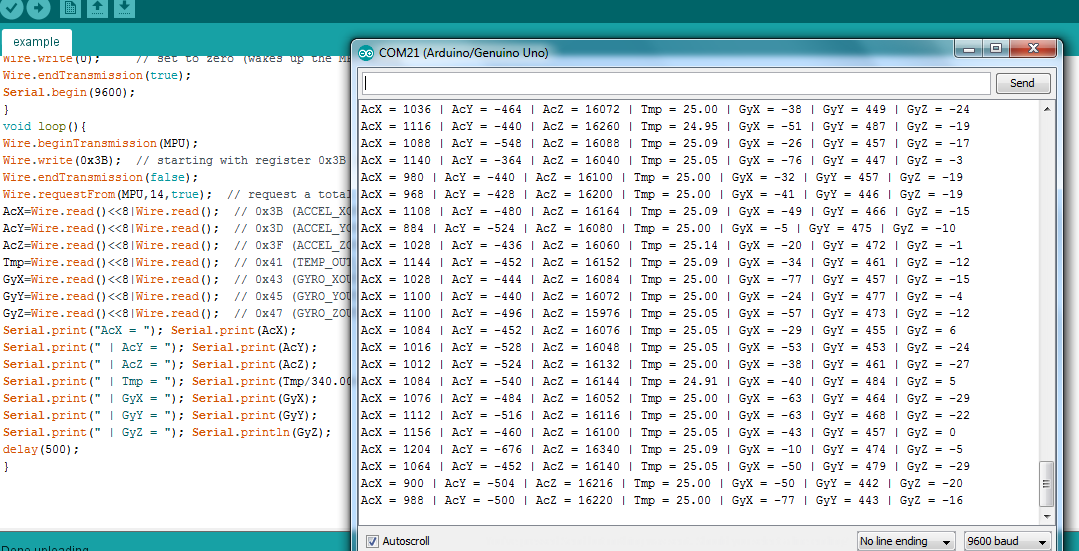
Serial.print(" | GyY = "); Serial.print(GyY);

Serial.print(" | GyZ = "); Serial.println(GyZ);

delay(500); }

**Task 3.3**

**Implement Example 3 on the Arduino Serial Monitor and observe the three dimensional readings of the accelerometer, gyro-meter and the temperature sensor reading.**

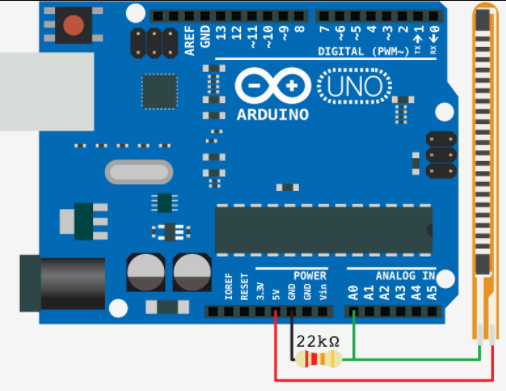


#### Flex Sensor 4.5" SEN-08606

A simple flex sensor 4.5" in length. As the sensor is flexed, the resistance across the sensor increases. The resistance of the flex sensor changes when the metal pads are on the outside of the bend (text on inside of bend). Connect it is series like any resistor (or variable resistor in this case). Connect one end to the power supply, and the other to ground (or in series with other resistors and then to ground).



**Interfacing Flex Sensor with Arduino Uno:**



**Example 4:**

const int flexPin = A0; //pin A0 to read analog input

//Variables:

int value; //save analog value

void setup(){

pinMode(LED\_BUILTIN, OUTPUT);

Serial.begin(9600); //Begin serial communication

}

void loop(){

value = analogRead(flexPin); //Read and save analog value from potentiometer

Serial.println(value); //Print value

if (value>=1015)

{digitalWrite(LED\_BUILTIN, HIGH);

delay(1000); }

else

{digitalWrite(LED\_BUILTIN, LOW);

delay(1000);}

// value = map(value, 700, 900, 0, 255);//Map value 0-1023 to 0-255 (PWM)

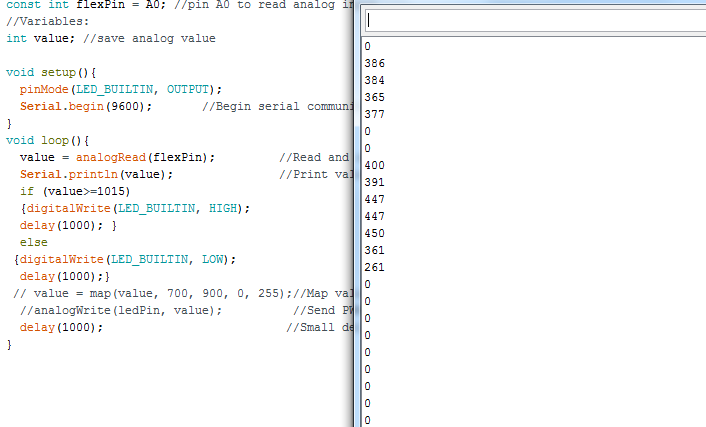
//analogWrite(ledPin, value); //Send PWM value to led

delay(1000); //Small delay

}

**Task 3.4**

**Implement Example 4 on the Arduino Serial Monitor and observe the LED status by changing theposition of Flex Sensor.**

****

**Task 3.4**

You are to simulate a racing car game. The MPU-6050 will be the controller. The user will use it to drive the car straight, turn it left or right. Similarly, the MPU-6050 will also provide the user the option to set different racing speeds or to apply the brakes and stop the car. Interface the flex sensor as a car horn. When the user flexes the sensor in one particular direction, the car horn will blow.

// MPU-6050 Short Example Sketch

// Arduino

// 2018

// Testing

#include<Wire.h>

int forward = 6;

int right = 5;

int left = 3;

const int MPU=0x68; // I2C address of the MPU-6050

int16\_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ,ab;

void setup(){

Wire.begin();

Wire.beginTransmission(MPU);

Wire.write(0x6B); // PWR\_MGMT\_1 register

Wire.write(0); // set to zero (wakes up the MPU-6050)

Wire.endTransmission(true);

Serial.begin(9600);

}

void loop(){

Wire.beginTransmission(MPU);

Wire.write(0x3B); // starting with register 0x3B (ACCEL\_XOUT\_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU,14,true); // request a total of 14 registers

AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL\_XOUT\_H) & 0x3C (ACCEL\_XOUT\_L)

AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL\_YOUT\_H) & 0x3E (ACCEL\_YOUT\_L)

AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL\_ZOUT\_H) & 0x40 (ACCEL\_ZOUT\_L)

Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP\_OUT\_H) & 0x42 (TEMP\_OUT\_L)

GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO\_XOUT\_H) & 0x44 (GYRO\_XOUT\_L)

GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO\_YOUT\_H) & 0x46 (GYRO\_YOUT\_L)

GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO\_ZOUT\_H) & 0x48 (GYRO\_ZOUT\_L)

Serial.print("AcX = "); Serial.print(AcX);

Serial.print(" | AcY = "); Serial.print(AcY);

Serial.print(" | AcZ = "); Serial.print(AcZ);

Serial.print(" | Tmp = "); Serial.print(Tmp/340.00+36.53); //equation for temperature in degrees C from datasheet

Serial.print(" | GyX = "); Serial.print(GyX);

Serial.print(" | GyY = "); Serial.print(GyY);

Serial.print(" | GyZ = "); Serial.println(GyZ);

AcY = map(AcY, -3000, 12000, 0, 150);//Map value 0-1023 to 0-255 (PWM)

// Serial.print(AcY);

analogWrite(forward, AcY); //Send PWM value to led

AcX = map(AcX, -10000,16000, 150, 0);//Map value 0-1023 to 0-255 (PWM)

// Serial.print(AcX);

analogWrite(right, AcX); //Send PWM value to led

ab = map(AcX, -16000, 10000, 0,255);//Map value 0-1023 to 0-255 (PWM)

Serial.print(ab);

analogWrite(left, ab); //Send PWM value to led

delay(500); }

