

Programming Manual of MYOSA

Basic Information of MYOSA and its working principle.

MYOSA board: MYOSA (Make Your Own Sensors Applications) is a plug-and-play, multipurpose, multi-sensor system for learning purpose. The motherboard of MYOSA is an Arduino derived open source development board. It is like a Lego set with which students can play and learn about the sensors. More importantly, it provides the users ready-to-apply connections of sensor-boards and the motherboard so that the user doesn't have to worry about wires and can directly jump on various creative applications. Various blocks include different sensors, actuators and an OLED display that shows the output. The motherboard also has a Bluetooth module through which the output data gets transferred to the MYOSA mobile app.

Working Principle: MYOSA works completely over I2C communication. I2C is a serial communication protocol, so data is serially transferred bit by bit along a single wire viz. the SDA line which is synchronised by clock. With I2C, data is transferred in *messages*. Messages are broken up into *frames* of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted by the sensor.

So, basically in brief, every sensor has one address identifier and sends data collected from the environment. Thereby MYOSA board collects information from each address which will be displayed to an OLED screen and also sends it to an android Bluetooth App.

MYOSA Kit:

- MYOSA Motherboard
- Sensors
 - Luminous Sensor
 - Barometric Pressure & Altitude Sensor
 - Air Quality Sensor
 - Particle Sensor
 - Magnetometer
 - RGB & Gesture Sensor
 - Gyroscope & Accelerometer
 - Real Time Clock
 - Temperature & Humidity Sensor
- OLED display
- Actuators
 - RGB led
 - Motor Driver
 - Buzzer
 - Relay
- Wireless Communication
 - Bluetooth Module
 - Wi-Fi Daughter Board

GETTING STARTED WITH MYOSA BOARD

WELCOME TO MYOSA BOARD! BEFORE YOU START MAKING APPLICATIONS USING THE BOARD, LET'S LEARN ABOUT SOME BASIC CODING IN OUR BOARD.

First of all, you'll need to set up the software to program your MYOSA board. MYOSA motherboard is developed as a derivative of Arduino. Arduino is an open-source electronics platform based on easy-to-use hardware and software. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

You can install the Arduino IDE using the following link:

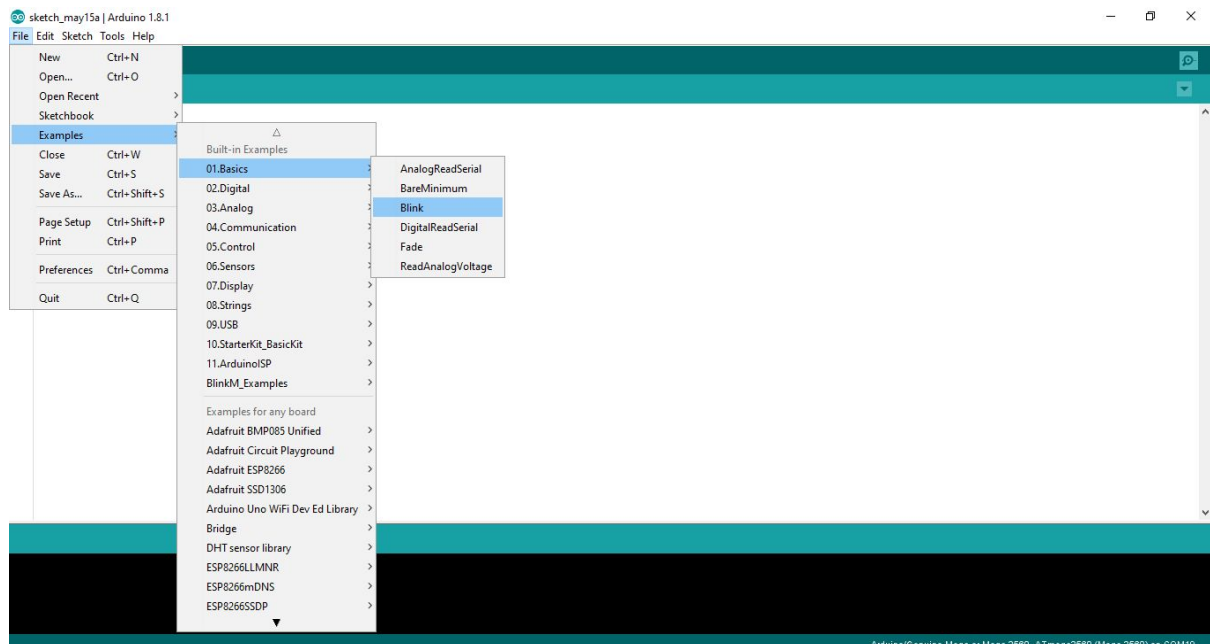
<https://www.arduino.cc/en/Guide/HomePage> → Select the OS you are using and download compatible version from the page.

Now, after you have installed Arduino IDE by following the instructions, let's get started with the basic application of LED blinking and learn how to use Arduino IDE.

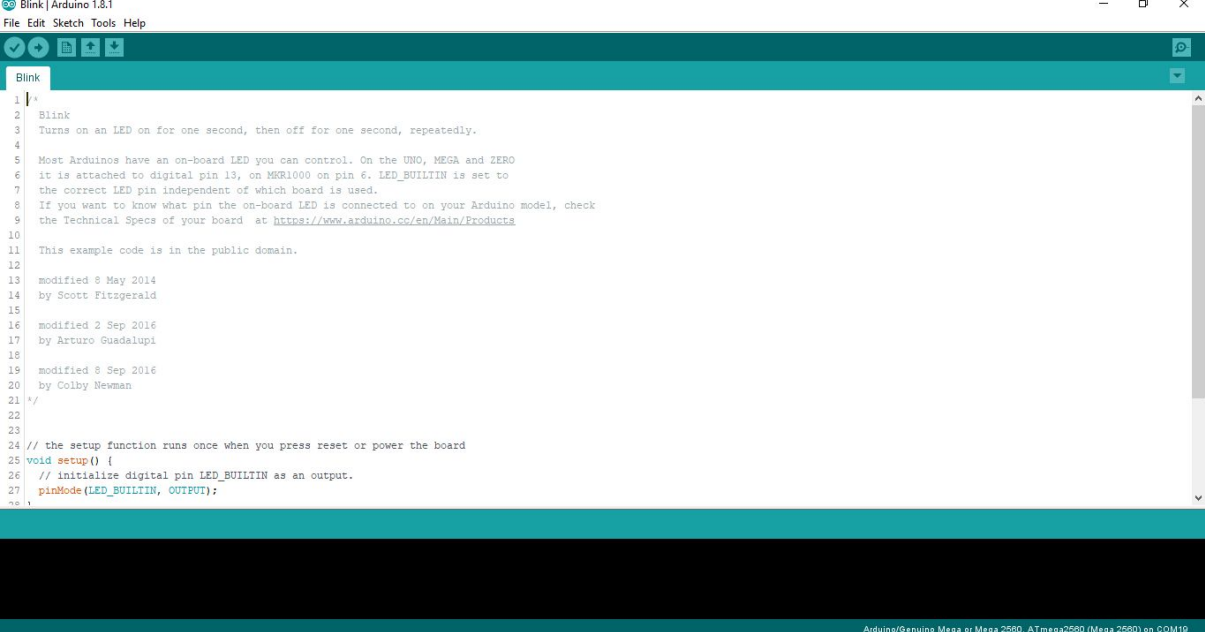
Example Code - BLINK: Turn an LED on and off

This is an example code in Arduino IDE. To open the blink sketch, follow the below mentioned steps.

- Open the Arduino Software.
- Go to File → Examples → Basics → Blink



- After you open the blink sketch, you'll see the window as below.

A screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.8.1". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening, saving, and running. The main text area displays the "Blink" sketch code, which is well-commented. The code includes a header section with a description of the sketch, a list of modifications by Scott Fitzgerald, Arturo Guadalupi, and Colby Newman, and a setup function that initializes the built-in LED. The status bar at the bottom indicates the board is "Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM19".

```
1 //  
2 Blink  
3 Turns on an LED on for one second, then off for one second, repeatedly.  
4  
5 Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO  
6 it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to  
7 the correct LED pin independent of which board is used.  
8 If you want to know what pin the on-board LED is connected to on your Arduino model, check  
9 the Technical Specs of your board at https://www.arduino.cc/en/Main/Products  
10  
11 This example code is in the public domain.  
12  
13 modified 8 May 2014  
14 by Scott Fitzgerald  
15  
16 modified 2 Sep 2016  
17 by Arturo Guadalupi  
18  
19 modified 8 Sep 2016  
20 by Colby Newman  
21 */  
22  
23  
24 // the setup function runs once when you press reset or power the board  
25 void setup() {  
26   // initialize digital pin LED_BUILTIN as an output.  
27   pinMode(LED_BUILTIN, OUTPUT);  
28 }
```

This example code is well commented so that you can easily understand it.

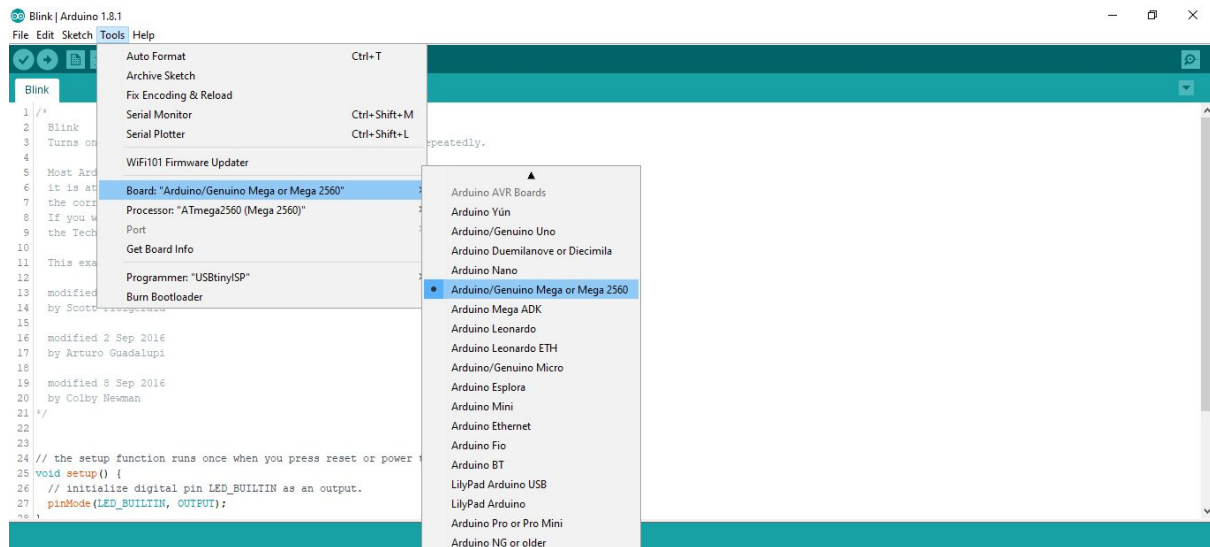
Now, we have our sketch ready. So, next step is to upload this code to MYOSA motherboard.

To upload the code, follow the below mentioned steps.

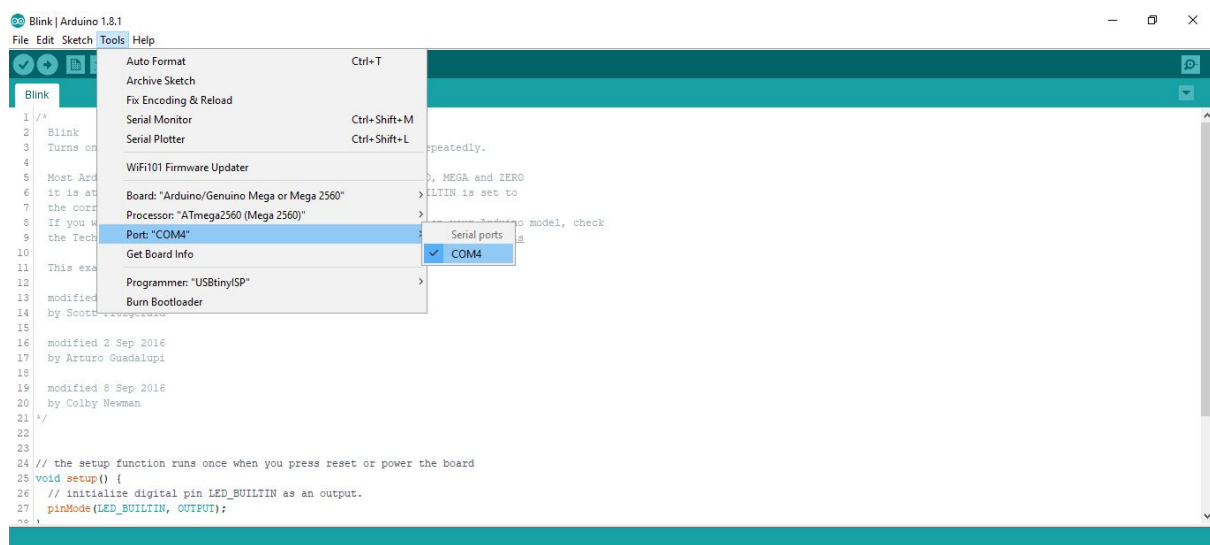
First, we need to select the board we are using.

As the heart of MYOSA motherboard is built using the same chip which is used for Arduino/Genuino Mega, we will select it.

- Go to Tools → Board → Arduino/Genuino Mega or Mega 2560



- After selecting this, we need to select the port. It is not necessary that you will get COM4, as shown below, you may get different COM port depending upon your system.



- Now, we have given the destination where the code is to be uploaded i.e., it will upload this code through the mentioned COM port.
- Final step is to upload the code. You can upload it by navigating through Sketch → Upload (OR) you can use the shortcut <Ctrl> + <u> (OR) just press a button in toolbar as highlighted below.



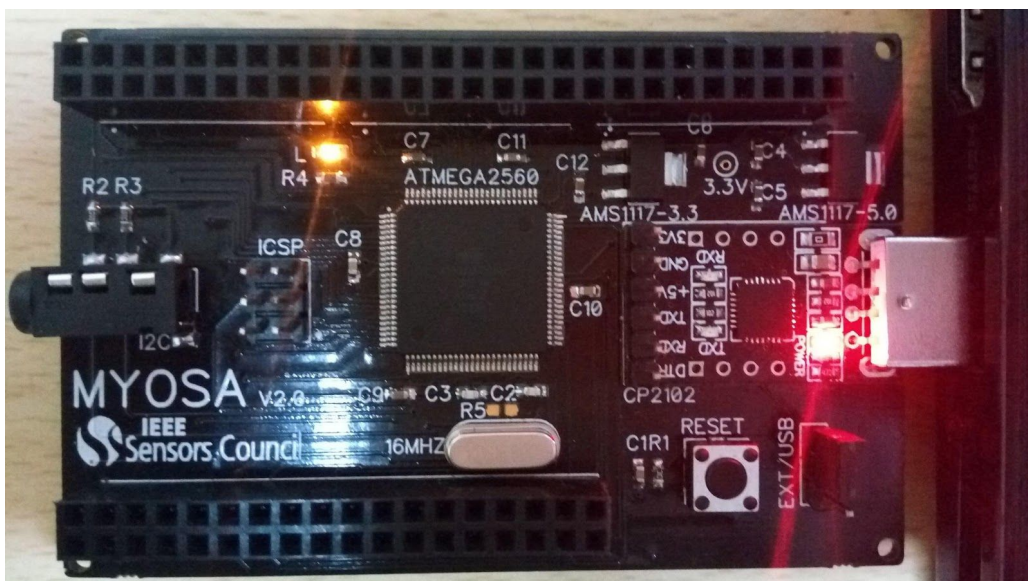
The screenshot shows the Arduino IDE interface with the 'Blink' sketch loaded. The code is as follows:

```

1 // Blink
2 // Turns on an LED on for one second, then off for one second, repeatedly.
3
4 // Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
5 // it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
6 // the correct LED pin independent of which board is used.
7 // If you want to know what pin the on-board LED is connected to on your Arduino model, check
8 // the Technical Specs of your board at https://www.arduino.cc/en/Main/Products
9
10 // This example code is in the public domain.
11
12 // modified 8 May 2014
13 // by Scott Fitzgerald
14
15 // modified 2 Sep 2016
16 // by Arturo Guadalupi
17
18 // modified 8 Sep 2016
19 // by Colby Newman
20
21 //
22
23
24 // the setup function runs once when you press reset or power the board
25 void setup() {
26   // initialize digital pin LED_BUILTIN as an output.
27   pinMode(LED_BUILTIN, OUTPUT);
28 }
29
30 // the loop function runs over and over again forever
31 void loop() {
32   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the positive voltage)
33   delay(1000);                     // wait for a second
34   digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the pin LOW (no voltage)
35   delay(1000);                     // wait for a second
36 }

```

- Now, the code is uploaded to MYOSA motherboard. You will see the built in LED (which is connected to D13) will start blinking.



- You can see the orange colour LED blinking.

You can tweak the code by changing the delay line which will change the on/off time of LED blinking. Also, you need to upload the code again.

Wohooo!!! You have now your first sketch running on MYOSA motherboard.....!!

INCLUDING LIBRARIES

Up till now, we executed the example code using MYOSA board.

WHAT IS A LIBRARY

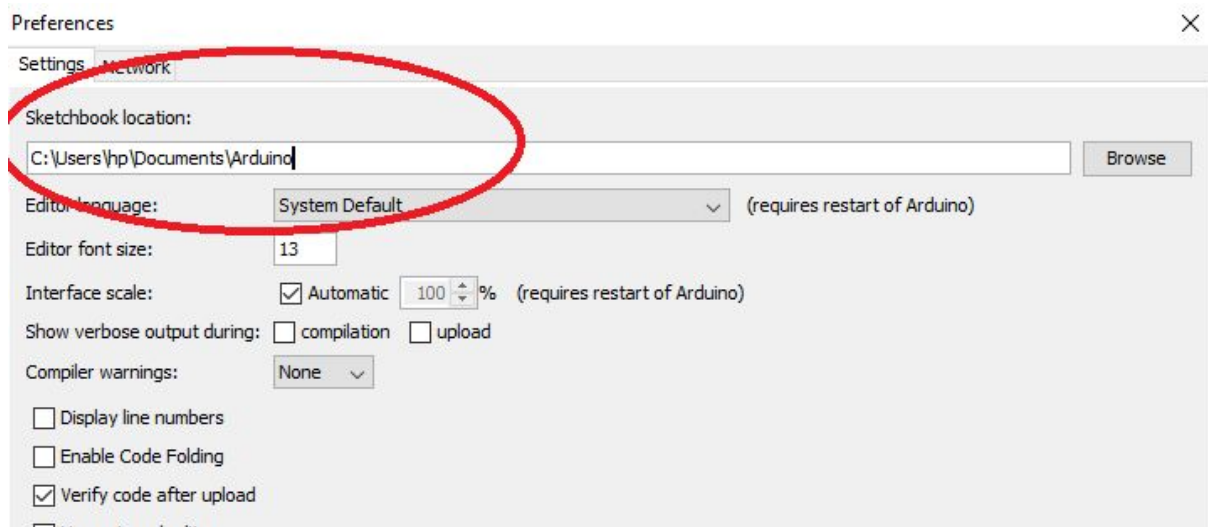
Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. Libraries make it easy to read sensor values without getting into the actual code. To use any library, you will need to download and install that.

Now let's learn how to include libraries for specific sensor, writing a code for reading the values and get its output on serial monitor.

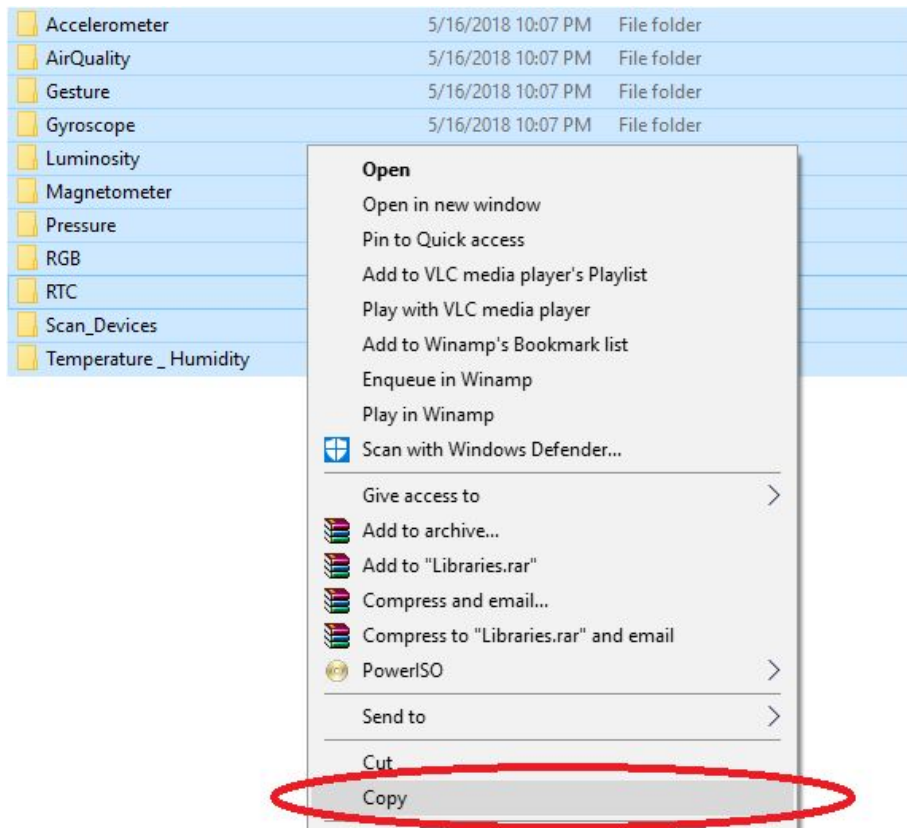
Go to <https://ieee-sensors.org/myosa/the-firmware/> and download library folder

Screenshot

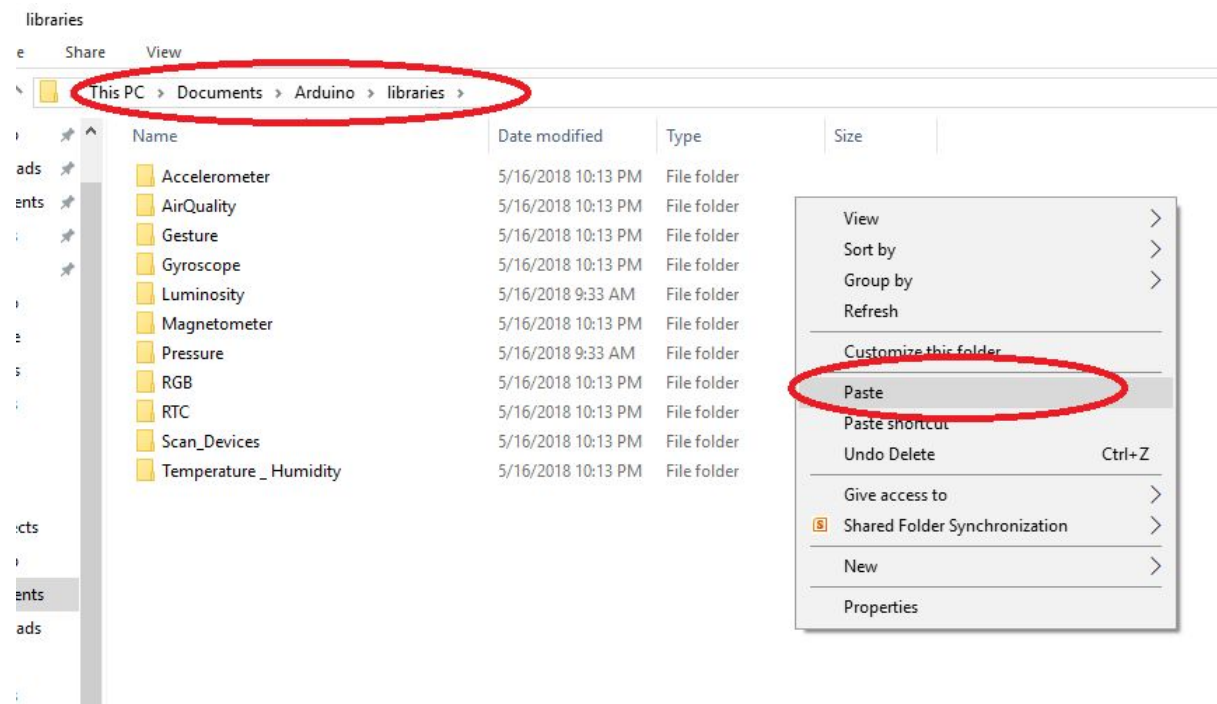
Extract the .zip file. You need to copy the subfolders of *Library* folder to the *libraries* folder of your sketchbook. You can find or change the location of your sketchbook folder at *File > Preferences > Sketchbook* location.



Open the extracted *Library* folder. Copy all subfolders.



Paste all subfolders to *libraries* folder of your sketchbook.



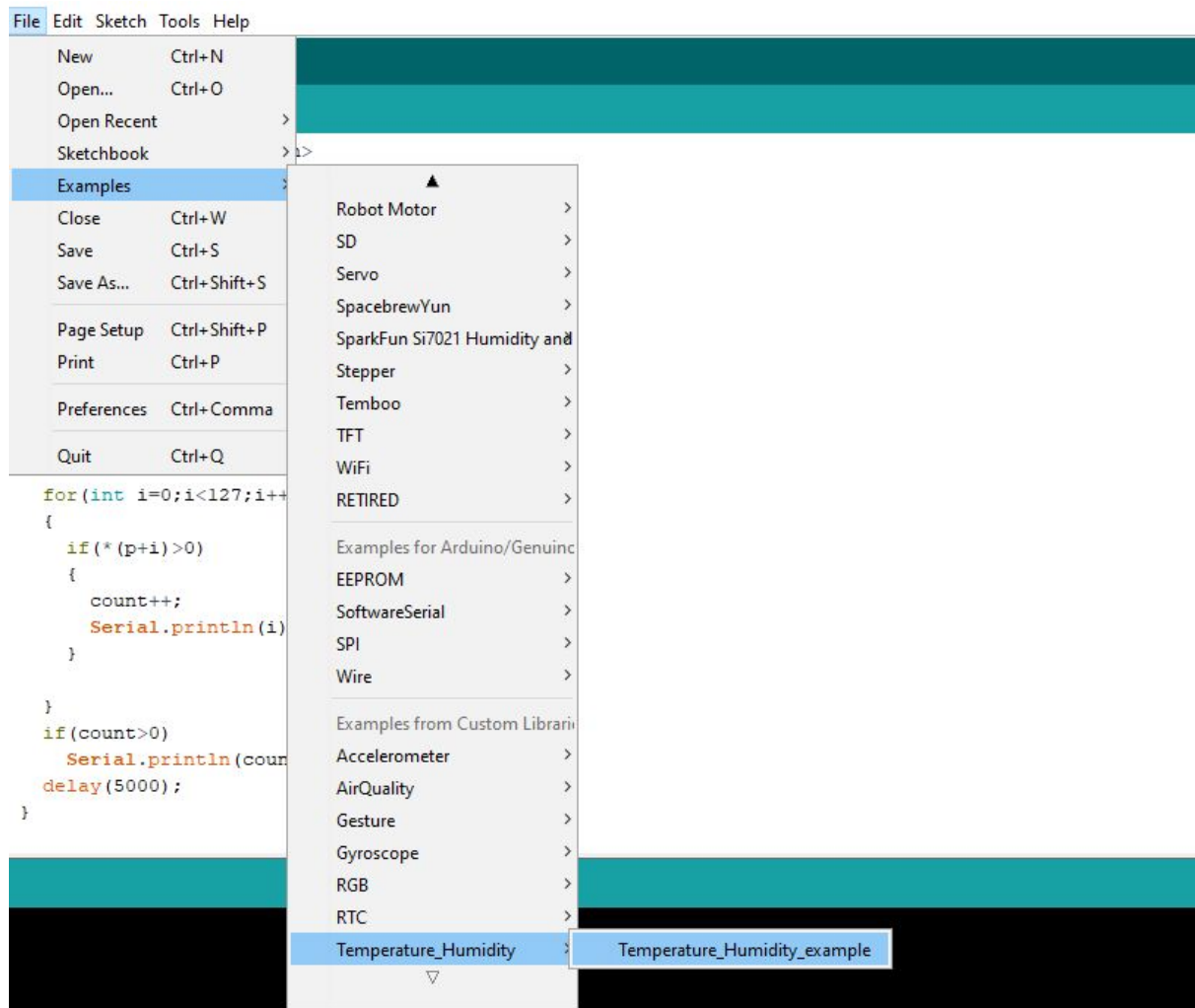
For more details of the installation of libraries, visit this page:

<https://www.arduino.cc/en/Guide/Libraries#toc2>

Write code using MYOSA libraries

Read temperature sensor:

First open your arduino IDE. Go to *File -> Examples -> Temperature_Humidity -> Temperature_Humidity_example*.



This will open a code in your IDE. The snippet of the code is shown below.


```

// Include Temperature_Humidity Library
#include <Temperature_Humidity.h>

// Make object of Temperature_Humidity class
Temperature_Humidity th;

//Setup your temperature and humidity sensor
void setup() {
    th.begin();
    Serial.begin(9600);
}

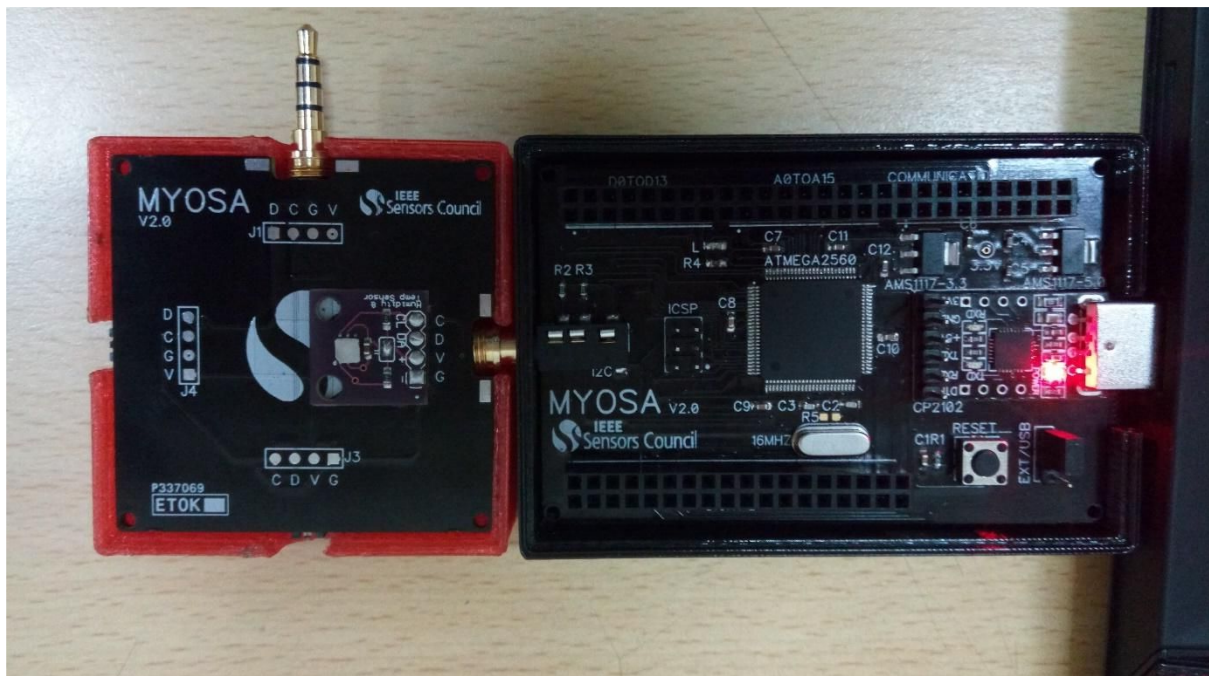
//Get readings of temperature and humidity sensor in loop
void loop() {

    float h=th.getHumidity();    // get humidity in percentage
    float tc=th.getTempC();      // get Temperature in Celcius
    float tf=th.getTempF();      // get Temperature in Fahrenheit

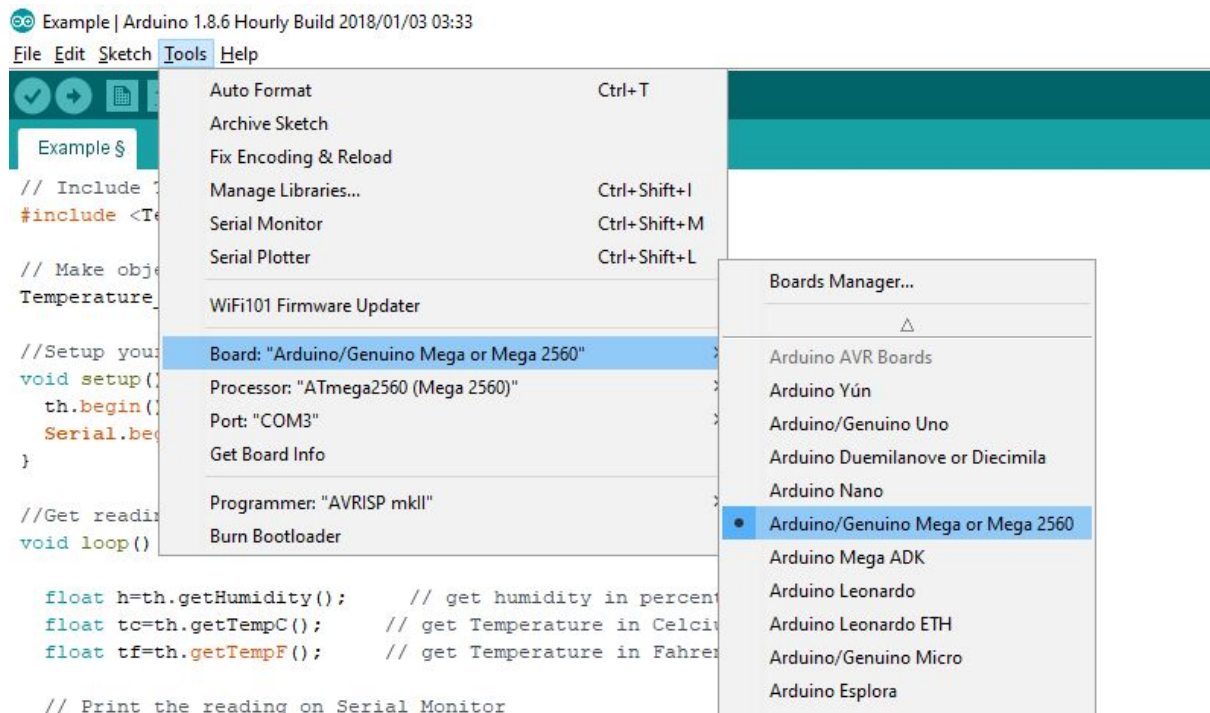
    // Print the reading on Serial Monitor
    Serial.print("Humidity : ");
    Serial.println(h);
    Serial.print("Temperature in C : ");
    Serial.println(tc);
    Serial.print("Temperature in F : ");
    Serial.println(tf);
}

```

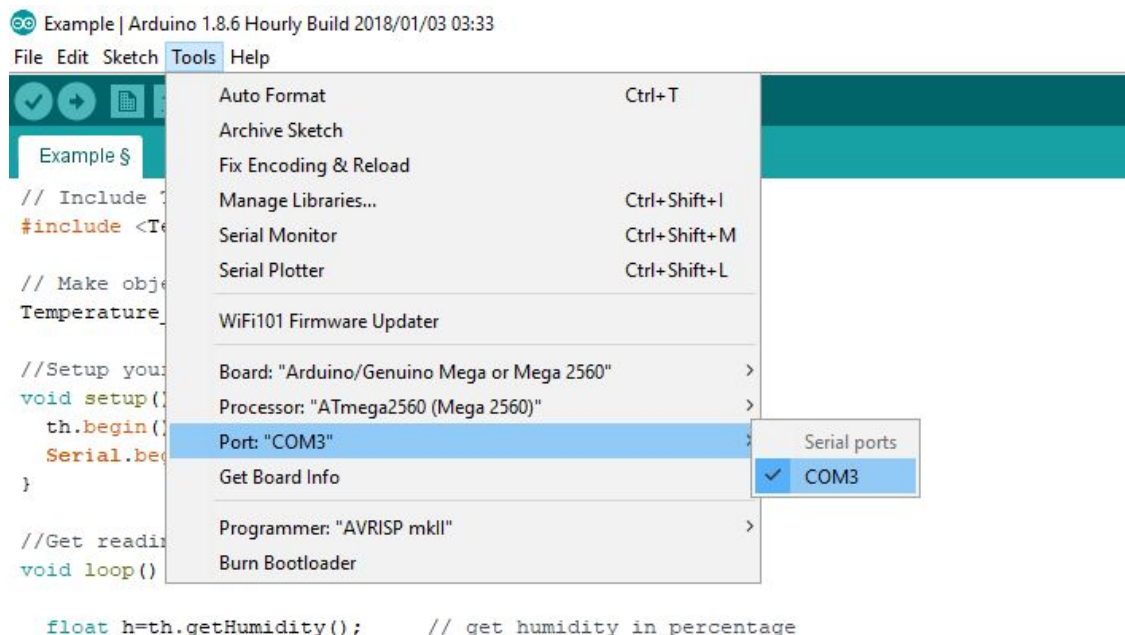
Now connect MYOSA motherboard to your computer. Connect Temperature Sensor to MYOSA motherboard.



After connecting MYOSA motherboard to computer using USB port, go to *tools* -> *Board* and choose *Arduino/Genuino Mega or Mega 2560*.



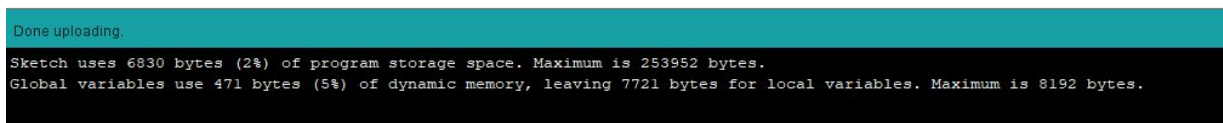
Now go to *tools* -> *port* and choose COM on which board is connected. It is not necessary that you will get the same COM port. You may get different COM port.




You can upload it by navigating through Sketch → Upload (OR) you can use the shortcut <Ctrl>+<u> (OR) just press a button in toolbar as shown below.



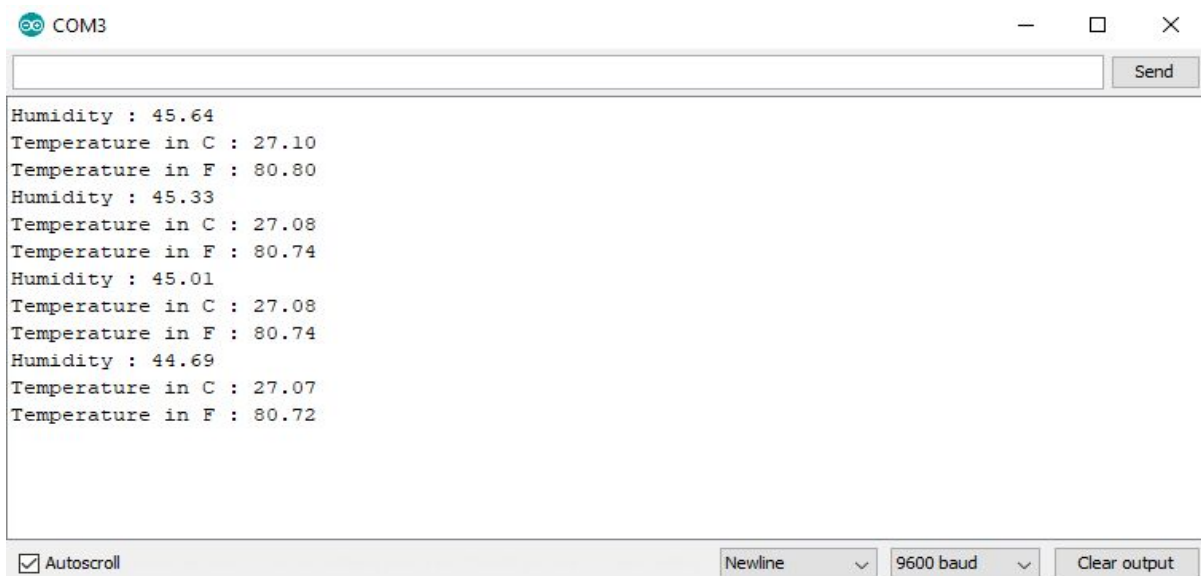
After uploading the code you can see the status of uploading at the *status bar* below the *Edit* area



Once you see *Done uploading* on status bar, open the Serial Monitor by clicking on  icon at the right top of *Button* bar.



Now you can see the values of sensor on Serial Monitor.



To write a code for any other sensor visit <https://ieee-sensors.org/myosa/sensorboards/> and start exploring.

Understanding the MYOSA Code:

Inclusion of Libraries - The first step is to include libraries for various sensor and display used. This block of code includes all the libraries which are used in the later part of the code.

```
1 //Inclusion of libraries
2 #include <SPI.h>
3 #include <Adafruit_GFX.h>
4 #include <Adafruit_SSD1306.h>
5 #include <Wire.h>
6 #include "SparkFunCCS811.h"
7 #include <SparkFunTSL2561.h>
8 #include <SparkFun_APDS9960.h>
9 #include "BlinkM_funcs123.h"
10 #include "MAX30105.h"
11 #include "heartRate.h"
12 #include "SparkFun Si7021 Breakout Library.h"
```

Address definition - Addresses of all the sensors are defined in this block of code in hexadecimal format. The address of any sensor is a 7/10-bit binary code. Hence the value of address ranges from [1-127] ([1-1023]). All the sensors/actuators/display used in MYOSA possesses 7-bit addressing and hence the value ranges from [1-127].

```
14 //Address definitions
15 #define DS1307_ADDRESS 0x68
16 #define mag_addr 0x1E //Magnetometer - HMC5883L
17 #define i2cled_addr 0x09 //I2C Led - BlinkM
18 #define air_addr 0x5B //Air Quality Sensor - CCS811
19 #define gest_addr 0x39 //Gesture Sensor - APDS9960
20 #define pressure_addr 0x77 //Pressure Sensor - BMP180
21 #define th_addr 0x40 //Temperature and Humidity Sensor - Si7021
22 #define lumi_addr 0x29 //Luminous Sensor - TSL2561
23 #define particle_addr 0x57 //Particle Sensor - Max30105
24 #define max_value 127 //Maximum No. of I2c Address
25 #define ssd_addr 0x3C
26 #define mpu_addr 0x69
```

Variables & Object declaration - This section declares all the Global Variables used by various functions. Definition and purpose of each variable are defined as below,

- max_value = 127
 - Maximum number of I2C addresses → 7-bit
- initDevices[max_value]
 - Binary array of all the devices which require initialization
 - 0 → Sensor is Not Connected and 1 → Sensor is Connected
- devices[max_value]
 - Binary array of currently connected devices
 - 0 → Sensor is Not Connected and 1 → Sensor is Connected
- prevIter[max_value]

- Sensor connected in previous iteration
- 0 → Sensor is Not Connected and 1 → Sensor is Connected

```

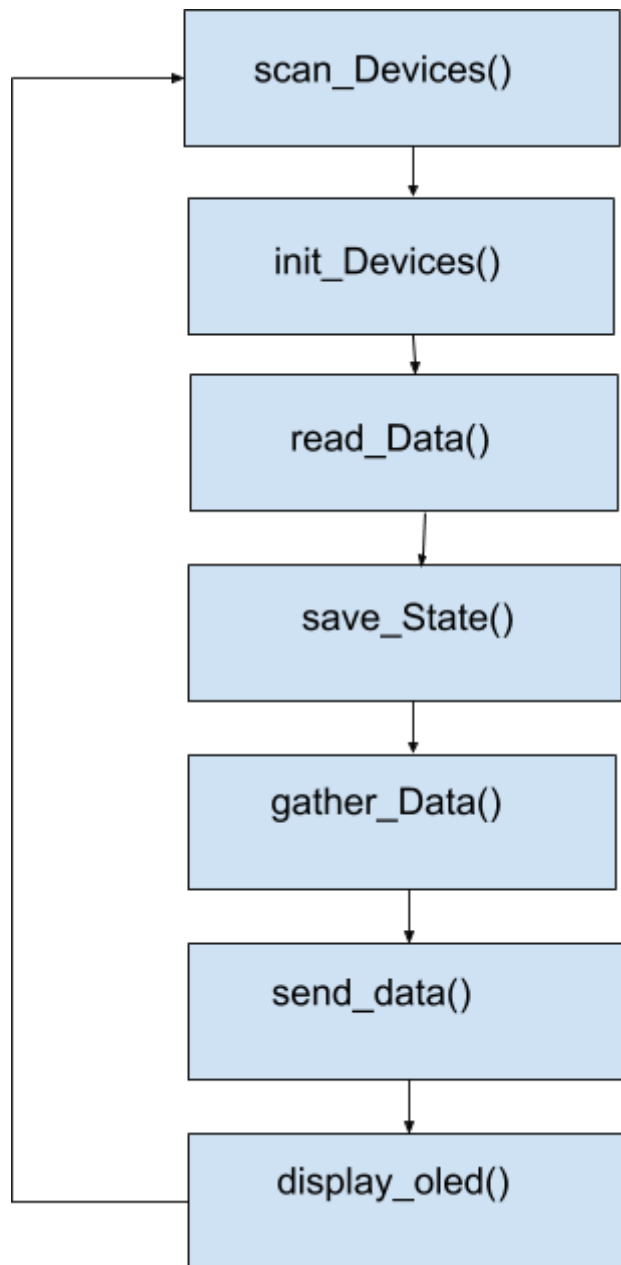
83 //array of devices which require initialization. 0 --> don't require initialization, 1 --> require initialization.
84 int initDevices[max_value];
85
86 //array of currently connected devices. 0 --> Not connected, 1 --> connected
87 int devices[max_value] = {0};
88
89 //devices connected in previous iteration of loop.
90 int prevIter[max_value] = {0};
91
92 //***** Variables for Sensors and actuators *****
93 //ccs object
94 CCS811 mySensor(air_addr);

```

void setup()

- initDevices[addr] = 1
 - set the entries corresponding to all the available sensors in the initialization array to 1. "addr" denotes the address, i.e. mag_addr, air_addr, etc.
- display.begin()
 - //initialize the OLED display
- Serial.begin(115200)
 - Beginning Serial Communication at 115200 baud rate for serial monitor.
- Serial1.begin(115200)
 - Beginning Serial Communication at 115200 baud rate for Bluetooth communication.

void loop() – After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board. The complete Algorithm of MYOSA is described in the block below. Each of the component block is explained in detail below the image.



scan_Devices() - Scan for the available devices on the bus. If the device is found, then the entry of the corresponding address in the devices array is set to 1, else it is set to 0. Scanning of devices is done by looping through all possible address and checking for connected device and if found, the initialization array and device array entries corresponding to that device is set to 1.

init_Devices() - After the bus has been scanned for the connected devices, there are some devices which are connected or reconnected and which are to be initialized. This function initializes the devices which are to be initialized by comparing the devices connected during the current iteration and the devices connected during the previous iteration.

read_data() – This function reads the value from all the connected devices (sensors)

save_data() - Save the data of devices array to prevlter array so that it can used to determine the devices connected during the next iteration.

gather_data() - Gather the data collected from the Bluetooth to send it to the actuator. The actuator then behaves accordingly.

send_data() – This function sends the collected data to MYOSA application through Bluetooth.

display_oled() - Display the collected data on the oled. The data for each connected device will be displayed one by one for a fixed time.

Changes you need to make when you add a sensor or an actuator

When you add a sensor or an actuator you need to make the following changes.

- If sensor or actuator uses library then include that library in the code. This has to be added along with all other libraries in inclusion of libraries block.

```
1 //Inclusion of libraries
2 #include <SPI.h>
3 #include <Adafruit_GFX.h>
4 #include <Adafruit_SSD1306.h>
5 #include <Wire.h>
6 #include "SparkFunCCS811.h"
7 #include <SparkFunTSL2561.h>
8 #include <SparkFun_APDS9960.h>
9 #include "BlinkM_funcs123.h"
10 #include "MAX30105.h"
11 #include "heartRate.h"
12 #include "SparkFun_Si7021_Breakout_Library.h"
13 #include <new sensor/actuator library>
```

- Add a line in the Address definition block for the corresponding address of sensor or actuator.

```
14 //Address definitions
15 #define DS1307_ADDRESS 0x68
16 #define mag_addr 0x1E //Magnetometer - HMC5883L
17 #define i2cled_addr 0x09 //I2C Led - BlinkM
18 #define air_addr 0x5B //Air Quality Sensor - CCS811
19 #define gest_addr 0x39 //Gesture Sensor - APDS9960
20 #define pressure_addr 0x77 //Pressure Sensor - BMP180
21 #define th_addr 0x40 //Temperature and Humidity Sensor - Si7021
22 #define lumi_addr 0x29 //Luminous Sensor - TSL2561
23 #define particle_addr 0x57 //Particle Sensor - Max30105
24 #define max_value 127 //Maximum No. of I2c Address
25 #define ssd_addr 0x3C
26 #define mpu_addr 0x69
27 #define new sensor/actuator addr 0xAA //Address of new sensor/actuator
```

- Declare the Global variables which are used by new sensor or actuator.

```

161 //Temperature and Humidity sensor params
162 int tempf =0;
163 int humidity = 0;
164 //Temperature and Humidity sensor params finished
165
166
167
168 //rtc params
169 int second;
170 int minute;
171 int hour;
172 int weekDay;
173 int monthDay;
174 int month;
175 int year;
176 //rtc params finished
177
178 //New sensor/actuator params
179 int temp1;
180 double temp2;
181 //New sensor/actuator params finished
182

```

- In void setup (), you need to add a line for the initialization of sensor/actuator if it requires.

```

255 //initialize the initialization array
256 initDevices[mag_addr] = 1;
257 initDevices[air_addr] = 1;
258 initDevices[gest_addr] = 1;
259 initDevices[pressure_addr] = 1;
260 initDevices[i2cled_addr] = 1;
261 initDevices[lumi_addr] = 1;
262 initDevices[particle_addr] = 1;
263 initDevices[th_addr] = 1;
264 initDevices[ssd_addr]=1;
265 initDevices[mpu_addr]=1;
266 initDevices[new_sensor/actuator]=1;

```

- In void loop (), nothing has to be changed.
- You need to add an “else if” block in initFunction() if it requires initialization.

```

697 //***** Initialization Codes for all sensors *****
698 void initFunction(int addr)
699 {
700 //***** Initialise Magnetometer Sensor*****
701 if (addr == mag_addr)
702 {
703 Wire.beginTransmission(mag_addr); //open communication with HMC5883
704 Wire.write(0x02); //select mode register
705 Wire.write(0x00); //continuous measurement mode
706 Wire.endTransmission();
707 }
708
709 //*****Initialise new sensor/actuator*****
710 else if (addr == new_sensor_addr)
711 {
712 /*
713 Write the code for initialization of new sensor/actuator
714 */
715 }
716
717
718 //***** Initialise Air Quality Sensor *****
719 else if (addr == air_addr)
720 {
721 CCS811Core::status returnCode = mySensor.begin();
722

```

- You need to add an “else if” block in readFunction() for reading the data from the sensor.

```

1056 //***** Reading Pressure sensor *****
1057 else if (addr == pressure_addr)
1058 {
1059     int32_t b5;
1060     b5 = temperature();
1061     Serial.print("Temperature: ");
1062     Serial.print(T, 2);
1063     Serial.print("°C, ");
1064     P = pressure(b5);
1065     Serial.print("Pressure: ");
1066     Serial.print(P, 2);
1067     Serial.print(" mbar, ");
1068     Serial.print(P * 0.75006375541921, 2);
1069     Serial.print(" mmHg, ");
1070     Serial.print(P * 0.75006375541921 * 133.322387415);
1071     Serial.println(" Pascal");
1072 }
1073
1074 //***** Reading new sensor *****
1075 else if(addr == new_sensor/actuator_addr)
1076 {
1077
1078     /*
1079     *Write code for reading the sensor data
1080     */
1081
1082 }

```

- If you want to display the data to OLED then you need to add “else if” block in OLED_display().
- If you want to display the data to Serial monitor then you have to do the same in display_data() function.

```

1368 void OLED_display(int addr)
1369 {
1370     display.setTextSize(1);
1371     display.setTextColor(WHITE);
1372     display.setCursor(0,0);
1373     //display.println(names[flag-1]);
1374     if(addr==lumi_addr) //Light Intensity
1375     {
1376         display.clearDisplay();
1377         display.println("Luminous Sensor");
1378         display.print("Infrared: ");
1379         display.println(ch1);
1380         display.print("Visible: ");
1381         display.println(ch0-ch1);
1382         display.print("LUX: ");
1383         display.println(Lux_value);
1384         display.display();
1385         delay(500);
1386     }
1387
1388     else if(addr ==new_sensor/actuator)
1389     {
1390         //Write code for displaying data to OLED display
1391     }
1392
1393     else if(addr==mag_addr) //Magnetometer
1394     {

```

- If you want to see the data in mobile app then you need to add “if else” block to send_data() function.
- This function sends a string of 33 values separated by ‘,’(comma) and if a sensor is not connected then it will send ‘*’.
- This string sends the values in the specific order so you have to append the new values at the end of the string and not in the middle.

```

1601 void send_data() {
1602     /*****Luminosity sensor*****/
1603     Serial.println(Serial1.read());
1604     if(devices[lumi_addr]==0){
1605         Serial1.print("*,*,*,*");
1606     }else{
1607         Serial1.print(ch0 - chl);
1608         Serial1.print(",");
1609         Serial1.print(chl);
1610         Serial1.print(",");
1611         Serial1.print(Lux_value);
1612         Serial1.print(",");
1613     }
1614     //    New Sensor
1615     if(devices[new_sensor/actuator_addr]==0){
1616         Serial1.print("*,*,*");
1617     }else{
1618         //Write code for sending the string to app
1619     }
1620     //    pressure sensor
1621     if(devices[pressure_addr]==0){
1622         Serial1.print("*,*,*,*,*");
1623     }else{
1624         Serial1.print(T, 2);
1625         Serial1.print(",");

```