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.

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.

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:
Write code using MYOSA libraries

Read temperature sensor:

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

```
for(int i=0;i<127;++i)
{
  if(TMP>0)
  {
    count++;
    Serial.println(i);
  }
}
if(count>0)
    Serial.println(count);
    delay(5000);
```

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 Celsius
    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.

<table>
<thead>
<tr>
<th>Humidity</th>
<th>45.64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature in C</td>
<td>27.10</td>
</tr>
<tr>
<td>Temperature in F</td>
<td>80.80</td>
</tr>
<tr>
<td>Humidity</td>
<td>45.33</td>
</tr>
<tr>
<td>Temperature in C</td>
<td>27.08</td>
</tr>
<tr>
<td>Temperature in F</td>
<td>80.74</td>
</tr>
<tr>
<td>Humidity</td>
<td>45.01</td>
</tr>
<tr>
<td>Temperature in C</td>
<td>27.08</td>
</tr>
<tr>
<td>Temperature in F</td>
<td>80.74</td>
</tr>
<tr>
<td>Humidity</td>
<td>44.69</td>
</tr>
<tr>
<td>Temperature in C</td>
<td>27.07</td>
</tr>
<tr>
<td>Temperature in F</td>
<td>80.72</td>
</tr>
</tbody>
</table>
To write a code for any other sensor visit [https://ieee-sensors.org/myosa/sensorboards/](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.

```c
//Inclusion of libraries
#include <SPI.h>
#include <Adafruit_GPIO.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include "SparkFunCCS811.h"
#include <SparkFunISL29020.h>
#include <SparkFun_APDS9960.h>
#include "BlinkM_funcs123.h"
#include "MAX10105.h"
#include "heartRate.h"
#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].

```c
#define DS18B7_ADDRESS 0x28
#define magnet_addr 0x1E  // Magnetometer - HMC883L
#define led_addr 0x09     // I2C Led - BlinkM
#define air_addr 0x5B     // Air Quality Sensor - CCS811
#define gest_addr 0x39    // Gesture Sensor - ARDD9660
#define pressure_addr 0x77 // Pressure Sensor - BMP180
#define ch_addr 0x40      // Temperature and Humidity Sensor - Si7021
#define luxi_addr 0x29    // Luminous Sensor - TSL2561
#define particle_addr 0x57 // Particle Sensor - Max30105
#define max_value 127
#define ssd_addr 0x3C      // Maximum No. of I2C Address
#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
- $0 \rightarrow$ Sensor is Not Connected and $1 \rightarrow$ Sensor is Connected

```c
//array of devices which require initialization. 0 --> don't require initialization, 1 --> require initialization.
int initDevices[0x745][max_value];

//array of currently connected devices. 0 --> Not connected, 1 --> connected
int devices[0x587][max_value] = {0};

//devices connected in previous iteration of loop.
int prevIter[0x527][max_value] = {0};

//******************************************* Variables for Sensors and actuators *******************************************

//class object
CC3011 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 prevIter 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.

```cpp
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.

```cpp
14 //Address definitions
15 #define SI307_ADDRESS 0x68 //Magnetometer - HMC5883L
16 #define led_addr 0x09 //I2C Led - BlinkM
17 #define s_q_addr 0x3B //Air Quality Sensor - CCS811
18 #define gest_addr 0x3F //Gesture Sensor - APDS9960
19 #define pressure_addr 0x77 //Pressure Sensor - BME180
20 #define th_addr 0x40 //Temperature and Humidity Sensor - SI7021
21 #define lux_addr 0x29 //Luminous Sensor - ISL2912
22 #define particle_addr 0x57 //Particle Sensor - Max50105
23 #define max_value 127 //Maximum No. of I2C Address
24 #define xwt_addr 0x3C
25 #define xpu_addr 0x69
26 #define new_sensor/actuator addr 0xA //Address of new sensor/actuator
```

- Declare the Global variables which are used by new sensor or actuator.
In `void setup()`, you need to add a line for the initialization of sensor/actuator if it requires.

```c
// initialize the initialization array
initDevices[mag_addr] = 1;
initDevices[air_addr] = 1;
initDevices[ges_addr] = 1;
initDevices[pressure_addr] = 1;
initDevices[ultr_addr] = 1;
initDevices[particle_addr] = 1;
initDevices[ch_addr] = 1;
initDevices[add_addr] = 1;
initDevices[mpp_addr] = 1;
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.

```c
void initFunction(int addr)
{
    // initialize the initialization array
    Wire.beginTransmission(addr); // open communication with HMC5883
    Wire.write(0x02); // select mode register
    Wire.write(0x26); // continuous measurement mode
    Wire.endTransmission(true);

    // initialize new sensor/actuator
    else if (addr == new_sensor_addr)
    {
        // write the code for initialization of new sensor/actuator
        //
    }

    // initialize Air Quality Sensor
    else if (addr == air_addr)
    {
        //
    }
```
- You need to add an “else if” block in `readFunction()` for reading the data from the sensor.

```cpp
int32_t b5;
Serial.print("Temperature: ");
Serial.print(T, 2);
Serial.print("°C, ");
P = pressure(b5);
Serial.print("Pressure: ");
Serial.print(P, 2);
Serial.print(" mbar, ");
Serial.print(F / 0.756063795541921, 2);
Serial.print(" Pascal");

else if(addr == new_sensor/actuator_addr)
{
    //Write code for reading the sensor data
}
```

- 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.

```cpp
void OLED_display(int addr)
{
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0, 0);
if(addr == luxi_addr) //Light Intensity
{
    display.clearDisplay();
display.println("Luminous Sensor");
display.println("Infrared: ");
display.println(ch1);
display.println("Visible: ");
display.println(ch0-ch1);
display.println("UX: ");
display.println(lux_value);
display.display();
delay(500);
}
else if(addr == new_sensor/actuator)
{
    //Write code for displaying data to OLED display
}
```
**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.**

```c
void send_data() {
    // Luminosity sensor
    Serial.println(Serial1.read());
    if(devices[lumi_addr]==0) {
        Serial.print("*,*,*,*");
    } else {
        Serial.print(ch0 - ch1);
        Serial.print(", ");
        Serial.print(ch1);
        Serial.print(", ");
        Serial.print(Lux_value);
        Serial.print(", ");
    }

    // New Sensor
    if(devices[new_sensor/actuator_addr]==0) {
        Serial.print("*,*,*");
    } else {
        // Write code for sending the string to app
    }

    // pressure sensor
    if(devices[pressure_addr]==0) {
        Serial.print("*,*,*,*");
    } else {
        Serial.print(T, 2);
    }
    Serial.println(",");
}  
```