Detailed Firmware Guide

Make Your Own Sensors Application (MYOSA v3.0)

OLED Display

Library Name	oled.h
i2c address	0x3C
Datasheet of IC used (SSD1306)	

Demo Code Flow

```
#include <library>
Create an object of the class

void setup(void)
{
  Serial communication and Wire Begin.
  Try setting up the sensor.

  if display connected
  {
    //display is connected
  }
  else
  {
    //display is dis-connected
  }
}

void loop(void)
{
  if(Ping == display is connected)
  {
    //display data
  }
}
```

Easy to use functions available
oLed(uint8_t w, uint8_t h); Called when an object of the class <oLed> is created.
bool begin(); Does the initial setup required. Returns true if the sensor is connected, else returns false.
void drawPixel(int16_t x, int16_t y, uint16_t color); As the name suggests, it draws a pixel at specified position.
There are few other handy functions for graphics as below, drawLine; drawRect; drawCircle; drawTriangle; drawBitmap; drawChar etc... Above are just handy names of those functrions and not their declarations. Please refer library to learn use them.
void setCursor(int16_t x, int16_t y); Sets the cursor at desired position, to display any kind of message on the OLED.
void setTextSize(int _size); Sets the text size of the next buffer.
void setRotation(int x); Rotates the display for different applications. Acceptable values - 0, 1, 2, 3. Each value rotates display by 90°.
void print(); Prints the custom message passed as an argument to the function.
void display(); This function actually displays the text/graphics. Before this function is called, data is stored in the buffer.
void clearDisplay(); Clears the display screen of any text or graphics.

Note: There are few other advance functions in library for advance usages.

Temperature and Humidity

Library Name	TempAndHumidity.h
i2c address	0x40
Datasheet of IC used (SI7021)	

Demo Code Flow

```
#include <library>
Create an object of the class
void setup(void)
{
  Serial communication and Wire Begin.
  Try setting up the sensor.
}

void loop(void)
{
  if(Ping == Sensor is connected)
  {
    //display data
  }
  else
  {
    //sensor is dis-connected
  }
}
```

Easy to use functions available

TempAndHumidity(); Called when an object of the class <TempAndHumidity> is created.
bool begin(void); Does the setup required for the sensor if the sensor is connected, else returns false.
bool ping(void); Pings the sensor to check if its connected or not. Returns boolean value. In case of disconnection in the loop, ping function also does the begin() to setup the sensor.
float getRelativeHumdity(bool print=true); Returns RelativeHumidity levels in percentage.
float getTempC(bool print=true); Returns temperature in °C.
float getTempF(bool print=true); Returns temperature in °F.
float getHeatIndexC(bool print=true); The heat index (HI) is an index that combines air temperature and relative humidity, in shaded areas, to posit a human-perceived equivalent temperature a.k.a "felt air temperature". Function returns the HI value in °C

Note: There are few other advance functions in library for advance usages.

BarometricPressure

Library Name	BarometricPressure.h
i2c address	0x77u
Datasheet of IC used (BMP180)	

Demo Code Flow

```
#include <library>
Create an object of the class

void setup(void)
{
  Serial communication and Wire Begin.
  Try setting up the sensor.

  if sensor connected
  {
    //sensor is connected
  }
  else
  {
    //sensor is dis-connected
  }
}

void loop(void)
{
  if(Ping == Sensor is connected)
  {
    //display data
  }
}
```

Easy to use functions available
BarometricPressure(bmp180AccuracyMode_t=ULTRA_LOW_POWER); Called when an object of the class <BarometricPressure> is created.
bool begin(); Does the setup required. Returns true if the sensor is connected, else returns false.
int32_t getPressure(void); Returns pressure from sensor in Pascal.
float getPressurePascal(bool print=true); Returns pressure in Kilo-Pascal of current location.
float getPressureHg(bool print=true); Returns pressure in mmHg of current location.
float getPressureBar(bool print=true); Returns pressure in Bar of current location.
float getTempC(bool print=true); Returns temperature of current location in celcius.
float getAltitude(float p0, bool print=true); Returns altitude of the current location with refrence to sea level.
float getSeaLevelPressure(float altitude, bool print=true); Returns pressure with refrence to sea level.
bool ping(void); Pings the sensor to check if its connected or not. Returns boolean value.

Note: There are few other advance functions in library to calibrate the sensor for different applications.

Accelerometer and Gyroscope

Library Name	AccelAndGyro.h
i2c address	0x69
Datasheet of IC used	(MPU6050)

Demo Code Flow

```
#include <library>
Create an object of the class

void setup()
{

  Serial communication and Wire Begin.
  Try setting up the sensor.

  if sensor connected
  {
    //sensor is connected
  }
  else
  {
    //sensor is dis-connected
  }
}

void loop()
{
  if(Ping == Sensor is connected)
  {
    //display data
  }
}
```

Easy to use functions available
AccelAndGyro(); Called when an object of the class <AccelAndGyro> is created.
bool begin(); Does the setup required. Returns true if the sensor is connected, else returns false.
bool ping(void); Pings the sensor to check if its connected or not. Returns corresponding boolean value.
float getAccelX(bool print=true); getAccelY(bool print=true); getAccelZ(bool print=true); Returns Acceleration in the X/Y/Z direction (RAW Data).
float getGyroX(bool print=true); getGyroY(bool print=true); getGyroZ(bool print=true); Returns Angular motion in the X/Y/Z direction (RAW Data).
float getTiltX(bool print=true); getTiltY(bool print=true); getTiltZ(bool print=true); Returns Tilt angle in X/Y/Z Direction (RAW Data).
float getTempC(bool print=true); getTempF(bool print=true); Returns Temperature reading in Celcius/Fahrenheit.
bool getMotionStatus(bool print=true); Returns boolen value true if any motion is detected, else returns false.

Note: There are few other advance functions in library to calibrate the sensor for different applications.

Light, Proximity and Gesture

Library Name	LightProximityAndGesture.h
i2c address	0x39
Datasheet of IC used	(APDS9960)

Demo Code Flow

```
#include <library>
Create an object of the class
void setup(void)
{
  Serial communication and Wire Begin.
  Try setting up the sensor.

  if sensor connected
  {
    //sensor is connected
  }
  else
  {
    //sensor is dis-connected
  }
}

void loop(void)
{
  if(Ping == Sensor is connected)
  {
    //display data
  }
}
```

Easy to use functions available
LightProximityAndGesture(); Called when an object of the class <LightProximityAndGesture> is created.
bool begin(); Does the setup required for the sensor if the sensor is connected, else returns false."
char *getGesture(bool print=true); Returns if gesture is detected or not. If gesture is detected also prints the direction of gesture
float getProximity(bool print=true); Returns a value from 1-255. 1 means there is no object in proximity. 255 means object is very near to the sensor.
uint16_t *getRGBProportion(bool print=true); Returns proportion of RGB in the Ambient Light.
uint16_t getAmbientLight(bool print=true); Return LUX of the Ambient Light.
bool enableAmbientLightSensor(STATE_t interrupt = DISABLE); enableProximitySensor enableGestureSensor Enables the ambient light / proximity / gesture sensing engine of the sensor
bool disableAmbientLightSensor(); disableProximitySensor disableGestureSensor Disables the ambient light / proximity / gesture sensing engine of the sensor

Note: There are few other advance functions in library for advance usages.

Air Quality

Library Name	AirQuality.h
i2c address	0x5B
Datasheet of IC used (CCS811)	

Demo Code Flow

```
#include <library>
Create an object of the class

void setup(void)
{
  Serial communication and Wire Begin.
  Try setting up the sensor.

  if sensor connected
  {
    //sensor is connected
  }
  else
  {
    //sensor is dis-connected
  }
}

void loop(void)
{
  if(Ping == Sensor is connected)
  {
    //display data
  }
}
```

Functions Available
AirQuality(); Called when an object of the class <AirQuality> is created.
CCS811_STATUS_t begin(); Does the setup required. Returns true if the sensor is connected, else returns false.
uint16_t getTVOC(); Returns TVOC in ppb
uint16_t getCO2(); Returns CO2 concrentration in ppm
float getTemperature(); Returns temperature readings in celsius.

Note: There are few other advance functions in library to calibrate the sensor for different applications.

Actuator

Library Name	Actuator.h
i2c address	0x41
Datasheet of IC used (PCA9536)	

Demo Code Flow

```
#include <library>
Create an object of the class

void setup()
{
  Serial communication and Wire Begin.
  Try setting up the sensor.

  if actuator connected
  {
    //actuator is connected
  }
  else
  {
    //actuator is dis-connected
  }
}

void loop()
{
  if(Ping == Actuator is connected)
  {
    //display data
  }
}
```

Easy to use functions available
Actuator(); Called when an object of the class <Actuator> is created.
bool ping(); Does the setup required. Returns true if the actuator is connected, else returns false.
PIN_MODE_t getMode(PCA_PIN_t pin); Returns the mode configured for the pin. Every pin can be configured as Input or Output.
PIN_STATE_t getState(PCA_PIN_t pin); Returns the current state for the pin. Every pin can be at HIGH or LOW.
void setMode(PCA_PIN_t pin, PIN_MODE_t newMode); sets the desired mode for specified pin. <setMode(0, IO_OUTPUT)> sets the pin 0 for OUTPUT.
void setState(PCA_PIN_t pin, PIN_STATE_t newState); sets the desired state for specified pin. <setState(0, IO_HIGH)> sets the pin 0 as HIGH.

Note: There are few other advance functions along with private functions detailed in library.