IERG1000 Project 2 – Lab-3
Standalone Thermometer

Objectives

- To use OLED display module;
- To use Thermo sensor module;
- To build a standalone thermometer.

Equipment / Component / Software

- Arduino IDE (ready for ESP32);
- Libraries for OLED (download from Internet)
- ESP32 microprocess module (x1);
- OLED display module, SSD1306, size 128x64 (x1);
- Thermo sensor module, AHT (x1)
- Micro USB cable for ESP32 (x1);
- Breadboard (x1);
- A bunch of wires for breadboard;
- Tactile switch (x1);
- Thumb drive (x1, optional, if you don't have the right to write file to the computer)

Introduction

In Lab-2, we setup the Arduino IDE with ESP32 and did some basic experiments. In this Lab, we will use Thermo sensor (AHT-10) and OLED display (SSD1306) modules to make a standalone thermometer, and show temperature in centigrade on the OLED display module. For elite students, you can show Fahrenheit or do more as you want with the provided modules.

The thermo sensor and OLED display modules both are communicated with I2C protocol (https://en.wikipedia.org/wiki/i2c). Thanks to the contributors, the libraries for I2C have been written, so we can use it and start the project very easy.

Important:

- If you want to get higher mark, write enough comment in your program, it is good habit of a professional programmer.
- Plagiarism is not allowed, it is very easy to find out in programming course works.

Experiment 3.1: Connect OLED Display Module to ESP32

Use different color wires to connect the pins 3V3(VIN), GND, SCL and SDL between ESP32, OLED display and thermo sensor modules:

Import notes:

- The communication protocol between ESP32 to OLED display and thermo sensor modules is I2C. Use different color wires to indicate different signals.
    - RED = VIN and 3V3
    - BLACK = GND
    - WHITE = SCL
    - GREEN = SDA
- Incorrect corrections (e.g. SCL or SDA connected to 3V3 or GND) will damage the modules.
- Holes named a-e and f-j are five holes a group joining together with a metal strip (refer Lab-1 using breadboard).

Figure-1.1 0.96" OLED Display module

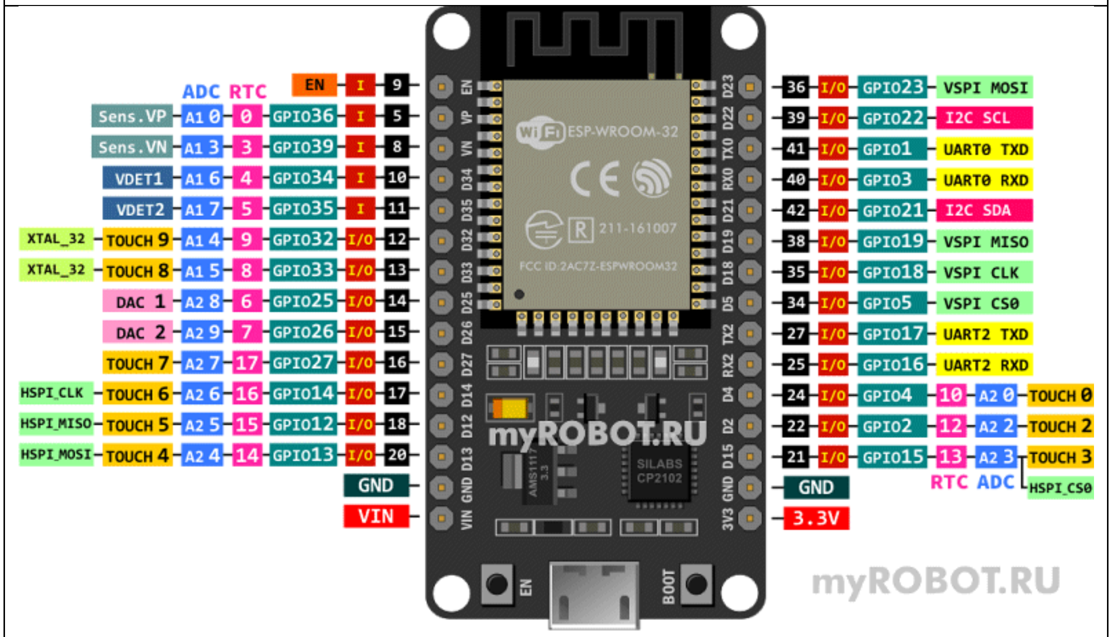

ADC RTC
EN I 9
Sens.VP A1 0 0 GPIO36 I 5
Sens.VN A1 3 3 GPIO39 I 8
VDET1 A1 6 4 GPIO34 I 10
VDET2 A1 7 5 GPIO35 I 11
XTAL_32 TOUCH 9 A1 4 9 GPIO32 I/O 12
XTAL_32 TOUCH 8 A1 5 8 GPIO33 I/O 13
DAC 1 A2 8 6 GPIO25 I/O 14
DAC 2 A2 9 7 GPIO26 I/O 15
TOUCH 7 A2 7 17 GPIO27 I/O 16
HSPI_CLK TOUCH 6 A2 6 16 GPIO14 I/O 17
HSPI_MISO TOUCH 5 A2 5 15 GPIO12 I/O 18
HSPI_MOSI TOUCH 4 A2 4 14 GPIO13 I/O 20
GND
VIN
EN VP VN D34 D35 D32 D33 D25 D26 D27 D14 D12 D13 GND VIN
D23 D22 TX0 RX0 D21 D19 D18 D5 TX2 RX2 D4 D2 D15 GND 3V3
ESP-WROOM-32
211-161007
FCC ID:2AC7Z-ESPWROOM32
myROBOT.RU
SILABS CP2102
EN
BOOT
36 I/O GPIO23 VSPI MOSI
39 I/O GPIO22 I2C SCL
41 I/O GPIO1 UART0 TXD
40 I/O GPIO3 UART0 RXD
42 I/O GPIO21 I2C SDA
38 I/O GPIO19 VSPI MISO
35 I/O GPIO18 VSPI CLK
34 I/O GPIO5 VSPI CS0
27 I/O GPIO17 UART2 TXD
25 I/O GPIO16 UART2 RXD
24 I/O GPIO4 10 A2 0 TOUCH 0
22 I/O GPIO2 12 A2 2 TOUCH 2
21 I/O GPIO15 13 A2 3 TOUCH 3
RTC ADC HSPI_CS0
GND
3.3V
myROBOT.RU

Figure-1.2 ESP32 pinout

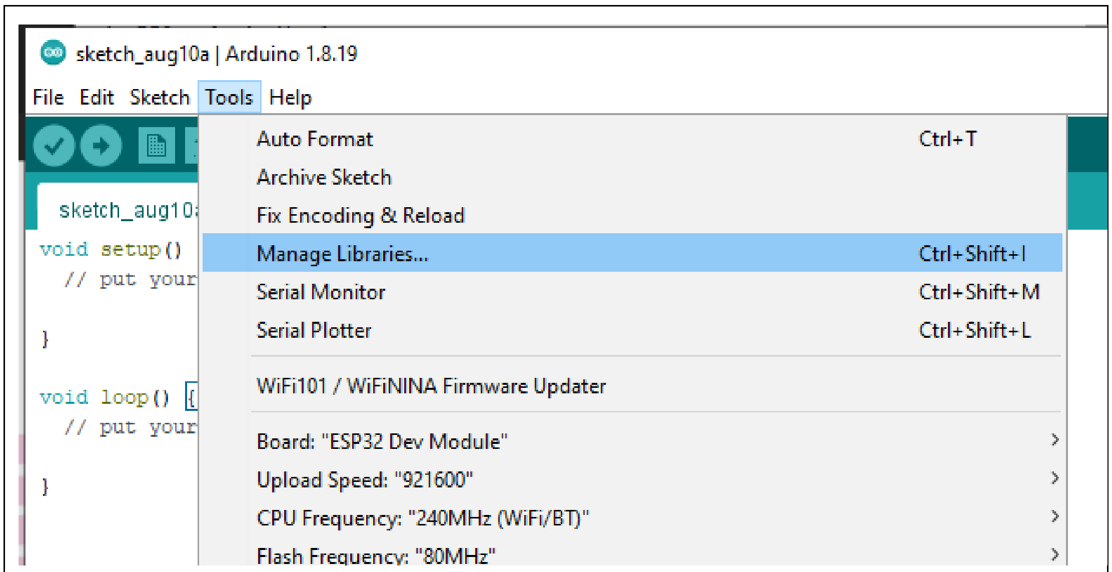
Procedures:

1. Find the OLED display module on the breadboard, it should be plugged to Zone-A column 1 to 11. The pins are on the breadboard A5a, A6a, A7a and A8a.
2. Identify which holes are connected to GND, VCC, SCL and SDA of the OLED display module. (refer Lab-1 Experiment-1.1.5)

Note: Will not remind you which procedure you have done in experiments anymore, you should remember them.

3. Connect OLED display module to ESP32, insert wires on the breadboard (refer Lab-1). Figure-4.2 shows ESP32-pin out.
    - OLED Display GND (A5e) – ESP32 GND (B2i)
    - OLED Display VCC (A6e) – ESP32 3V3 (B1i)
    - OLED Display SCL (A7e) – ESP32 D22 (B14i)
    - OLED Display SDA (A8e) – ESP32 D21 (B11i)

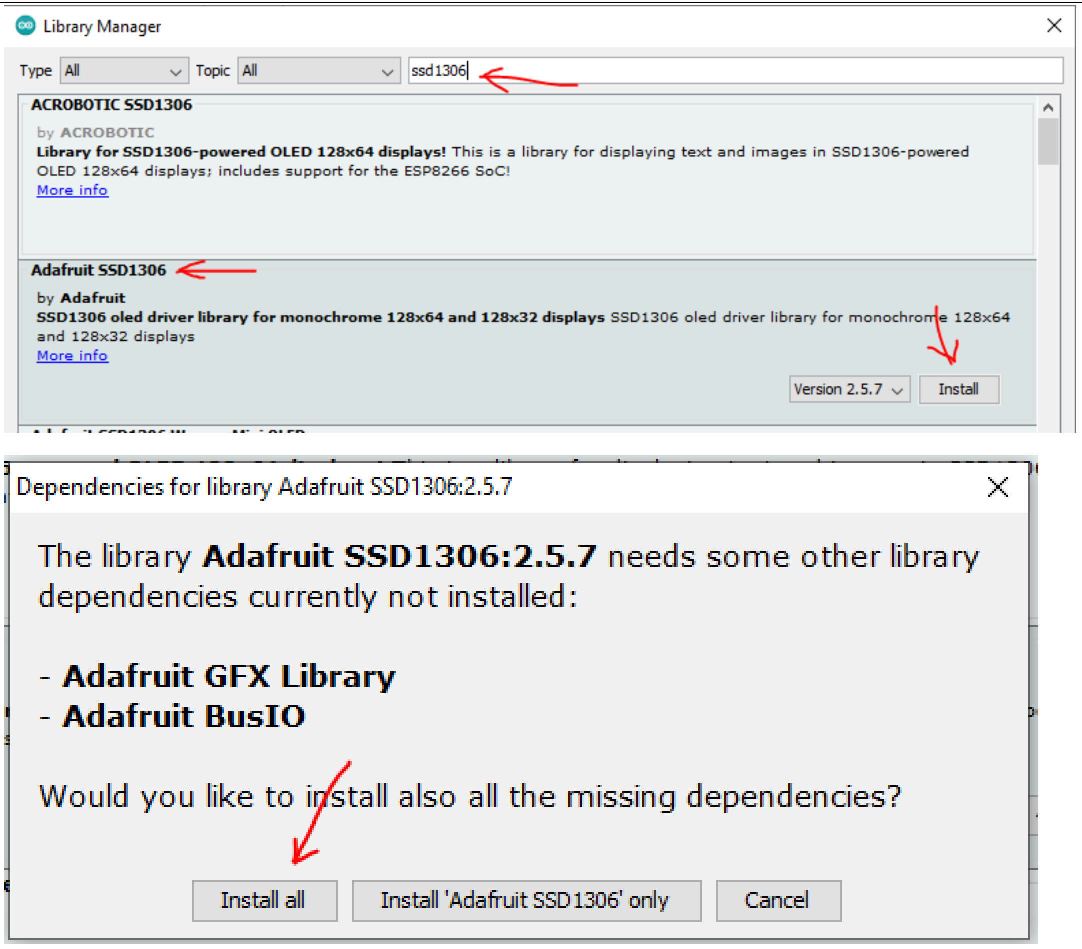
4. Start Arduino and open an empty project, we need to install some libraries. Select 'Tool' 'Manage Libraries…'. (Figure-1.3)
5. When the Manage Libraries dialogue is pop-up, it will connect to Internet to get the index, it will take a couple of minutes.
6. Key-in **SSD1306** and install the latest libraries for **Adafruit SSD1306 and Adafruit GFX Library** (as Figure-1.4). After installation, '**installed**' will be shown as in Figure-1.5. Press the button 'Close' to close the Library Manage.


sketch_aug10a | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
sketch_aug10

```
void setup() {
  // put your
}

void loop() {
  // put your
}
```

Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"

Figure-1.3 Mange Libraries


Library Manager

Type: All | Topic: All | ssd1306

ACROBOTIC SSD1306
by ACROBOTIC
Library for SSD1306-powered OLED 128x64 displays! This is a library for displaying text and images in SSD1306-powered OLED 128x64 displays; includes support for the ESP8266 SoC!
More info

Adafruit SSD1306
by **Adafruit**
SSD1306 oled driver library for monochrome 128x64 and 128x32 displays SSD1306 oled driver library for monochrome 128x64 and 128x32 displays
More info

Version 2.5.7 | Install

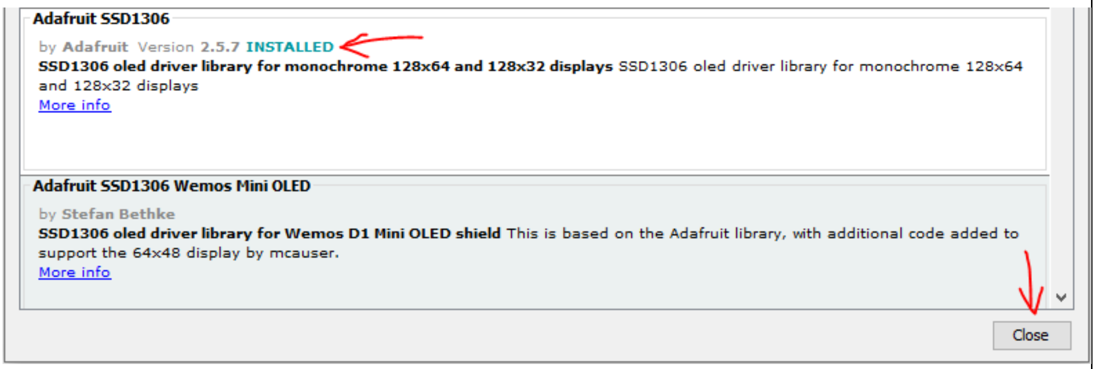
Dependencies for library Adafruit SSD1306:2.5.7

The library **Adafruit SSD1306:2.5.7** needs some other library dependencies currently not installed:

- **Adafruit GFX Library**
- **Adafruit BusIO**

Would you like to install also all the missing dependencies?

Figure-1.4 serarch SSD1306 on Manage Libraries and install it.


Adafruit SSD1306
by **Adafruit** Version **2.5.7 INSTALLED**
SSD1306 oled driver library for monochrome 128x64 and 128x32 displays SSD1306 oled driver library for monochrome 128x64 and 128x32 displays
More info

Adafruit SSD1306 Wemos Mini OLED
by **Stefan Bethke**
SSD1306 oled driver library for Wemos D1 Mini OLED shield This is based on the Adafruit library, with additional code added to support the 64x48 display by mcauser.
More info

Figure-1.5 **ESP8266 and ESP32 OLED driver for SSD1306** driver chip installed.

7. Open the official example **ssd1306_128x64_i2c** as in Figure-1.6. It is at the bottom of **Examples**.
8. Modify the line “#define SCREEN_ADDRESS 0x3D” to “#define SCREEN_ADDRESS **0x3C**” (Figure-1.7). The address of SSD1306 is incorrect in the official example.
9. **Save the project to a writable folder**. Compile and upload the project. You will see a demo on screen. It includes a few subroutines, could you find them in the program? Don’t worry, no need to understand them now, just to know what can be shown on screen and some C-language syntaxes.
   - testdrawline(); // Draw many lines
   - testdrawrect(); // Draw rectangles (outlines)
   - testfillrect(); // Draw rectangles (filled)
   - testdrawcircle(); // Draw circles (outlines)
   - testfillcircle(); // Draw circles (filled)
   - testdrawroundrect(); // Draw rounded rectangles (outlines)
   - testfillroundrect(); // Draw rounded rectangles (filled)
   - testdrawtriangle(); // Draw triangles (outlines)
   - testfilltriangle(); // Draw triangles (filled)
   - testdrawchar(); // Draw characters of the default font
   - testdrawstyles(); // Draw 'stylized' characters
   - testscrolltext(); // Draw scrolling text
   - testdrawbitmap(); // Draw a small bitmap image

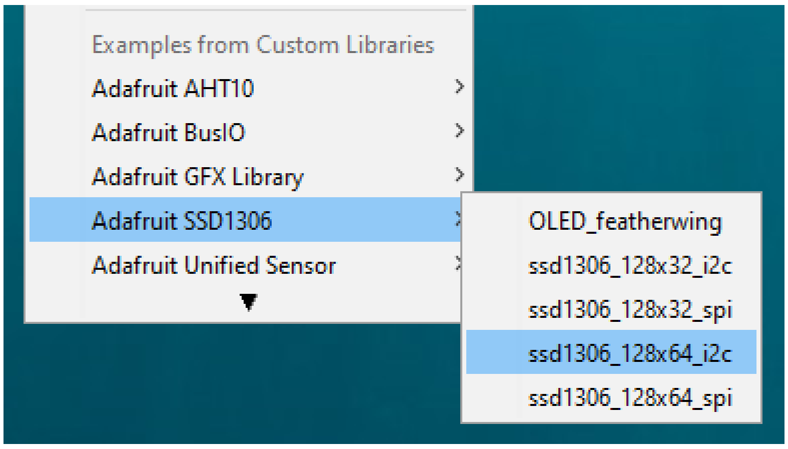
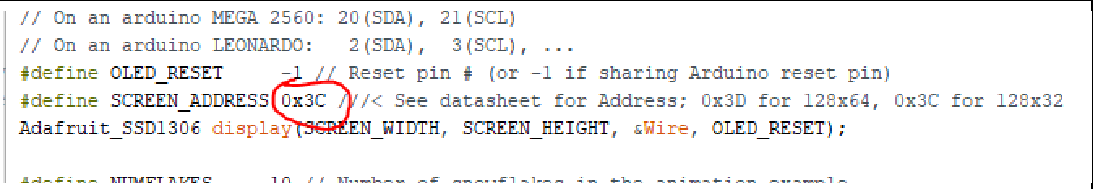


Figure-1.6 Open example ‘ssd1306_128x64_i2c’



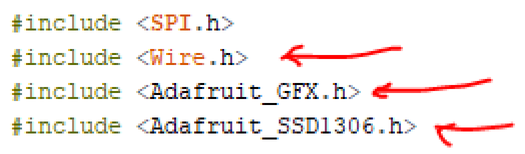
```
// On an arduino MEGA 2560: 20(SDA), 21(SCL)
// On an arduino LEONARDO:   2(SDA),  3(SCL), ...
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define NUMFLAKES     10 // Number of snowflakes in the animation example
```

Figure-1.7 Modify the address of SSD1306

10. Understand the program, only a few lines are important.

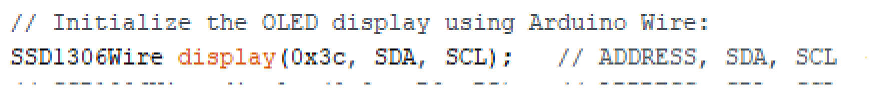
- #include means including the file into the project. (Figure-1.7).
- ‘SPI.h’ is not essential. (Figure-1.8)
- ‘Wire.h’ is for I2C protocol.
- ‘Adafruit_GFX.h’ and ‘Adafruit_SSD1306.h’ are for OLED display module.
- Figure-1.9, Initialize the SSD1306 OLED module address 0x3C.
- In setup( ) (Figure-1.10), that checks OLED display module installed or not.
- In the main program, loop( ) is empty, it means that the last statement ‘testanimate’ of setup() will work and loop forever.

A lot of subroutines start with ‘void’ after loop(). They are the bodies of functions (subroutine) and their details. Only ‘testdrawstylesr(void)’ we are interested in. (Figure-1.11)



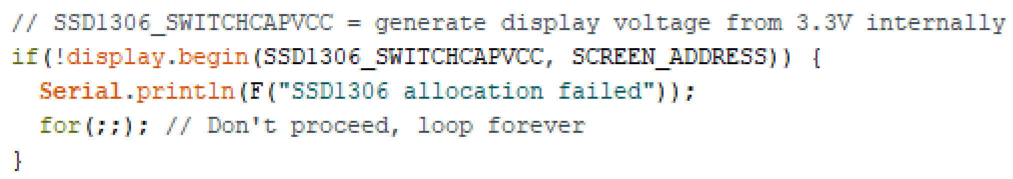
```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Figure-1.8 Use Wire.h



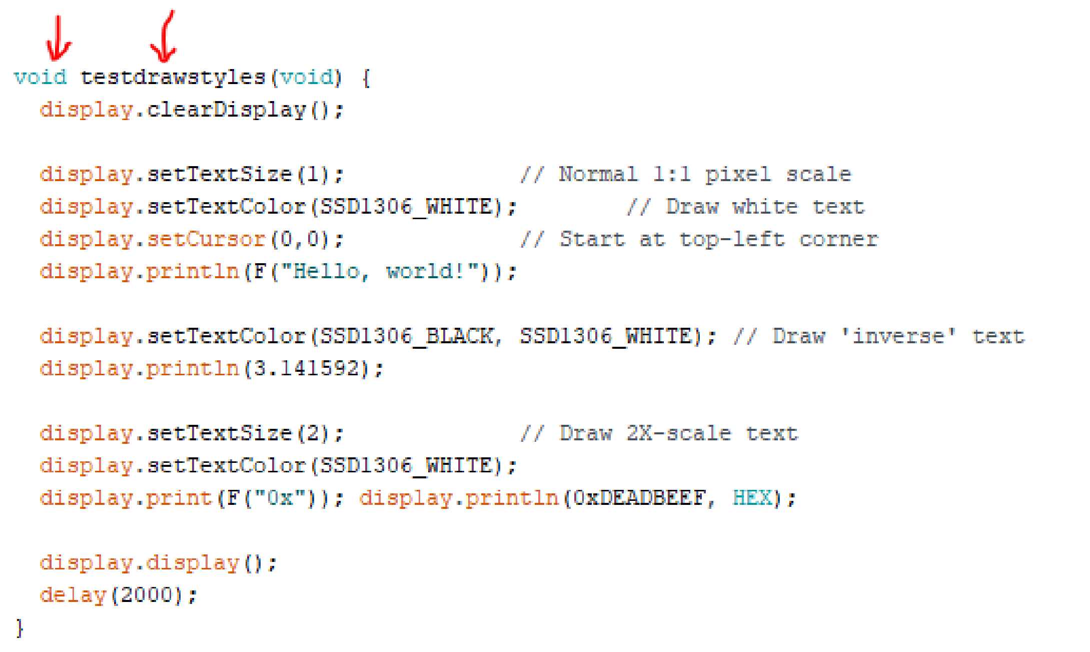
```
// Initialize the OLED display using Arduino Wire:
SSD1306Wire display(0x3c, SDA, SCL);   // ADDRESS, SDA, SCL
```

Figure-1.9 Create object SSD1306Wire with I2C address 0x3C



```
// SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
  Serial.println(F("SSD1306 allocation failed"));
  for(;;); // Don't proceed, loop forever
}
```

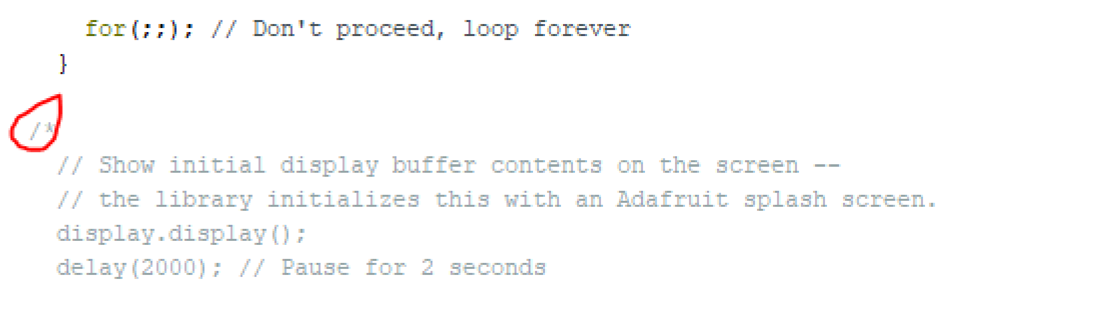
Figure-1.10 Init SSD1306



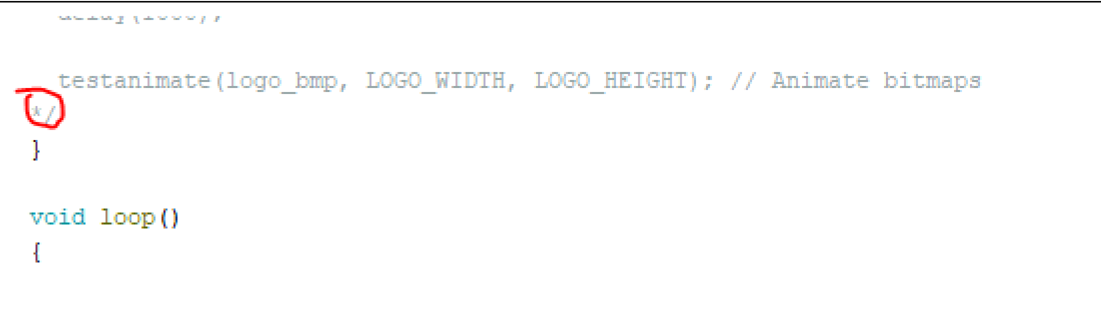
```
void testdrawstyles(void) {
  display.clearDisplay();

  display.setTextSize(1);             // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);        // Draw white text
  display.setCursor(0,0);             // Start at top-left corner
  display.println(F("Hello, world!"));

  display.setTextColor(SSD1306_BLACK, SSD1306_WHITE); // Draw 'inverse' text
  display.println(3.141592);

  display.setTextSize(2);             // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.print(F("0x")); display.println(0xDEADBEEF, HEX);

  display.display();
  delay(2000);
}
```

Figure-1.11 Some subroutines

11. Modify (trim down) the demo program for our project. You should save it to a writable folder (Procedure-8).
    - Add comments to all subroutines between after setup() after SSD1306 initialization (Figure-1.10), use /* */ syntax. Start of comment multiple-line with /*, end of comment multiple-line with */. Everything between /* and */ will be ignored. (Figure-1.12) and all commented lines will be changed to gray.
    - Copy the content of subroutine testdrawstyle(void) to loop and add delay(2000) to slowdown the loop rate as in Figure-1.13. Compile and upload the program.

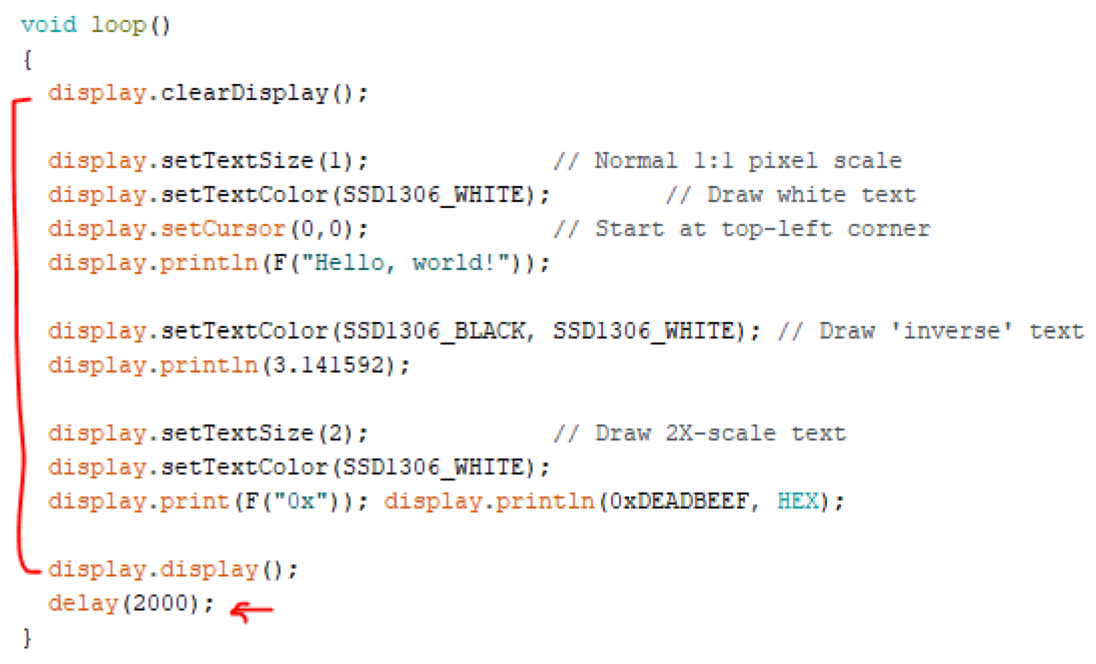


```
    for(;;); // Don't proceed, loop forever
  }

/*
  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(2000); // Pause for 2 seconds
```



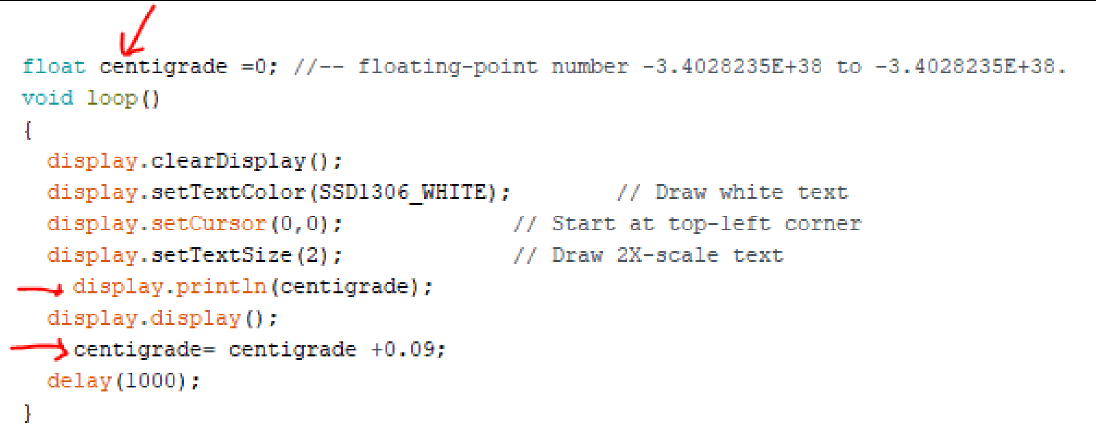
```
  testanimate(logo_bmp, LOGO_WIDTH, LOGO_HEIGHT); // Animate bitmaps
*/
}

void loop()
{
```

Figure-1.12 Comment multiple lines /* */



```
void loop()
{
  display.clearDisplay();

  display.setTextSize(1);             // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);        // Draw white text
  display.setCursor(0,0);             // Start at top-left corner
  display.println(F("Hello, world!"));

  display.setTextColor(SSD1306_BLACK, SSD1306_WHITE); // Draw 'inverse' text
  display.println(3.141592);

  display.setTextSize(2);             // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.print(F("0x")); display.println(0xDEADBEEF, HEX);

  display.display();
  delay(2000);
}
```

Figure-1.13 New program in loop( )



```
float centigrade =0; //-- floating-point number -3.4028235E+38 to -3.4028235E+38.
void loop()
{
  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);        // Draw white text
  display.setCursor(0,0);             // Start at top-left corner
  display.setTextSize(2);             // Draw 2X-scale text
  display.println(centigrade);
  display.display();
  centigrade= centigrade +0.09;
  delay(1000);
}
```

Figure-1.14 the new program in loop( ) with variable

12. Although the characters on screen are unchanged, they are updated every 2 second. To prove that we try to print a variable. Modify the program in loop as in Figure-1.14.
    - Create a variable named 'centigrade' and the datatype is float.
    - Display the variable on screen with text size 2.
    - The variable will be increased by 0.09 in each loop cycle.
    - You can change the delay() to adjust the loop time.
13. Compile and upload the program, could the result be you expected?

Experiment 3.2: Temperature Sensor Module

The model of temperature sensor module is AHT10. Its interface is I2C bus (https://en.wikipedia.org/wiki/i2c) . All I2C devices can connect together in parallel as in Figure-2.1. AHT10 integrated temperature and humidity sensor have the default temperature output as centigrade.

Note: The pin names of AHT10 are printed on the PCB bottom side. You need to unplug it to check. You should find them in Lab-1. A gap on the guider holder to prevent placing it to an incorrect column when you re-plug it.

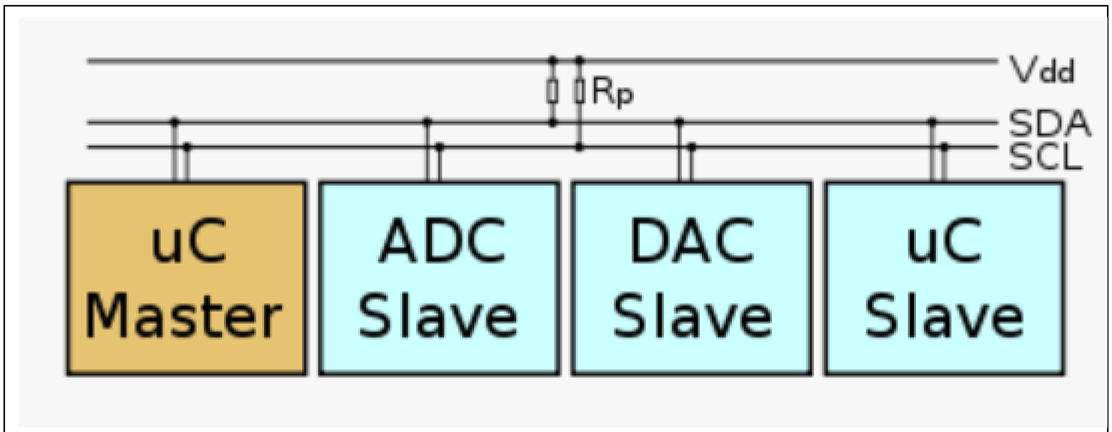


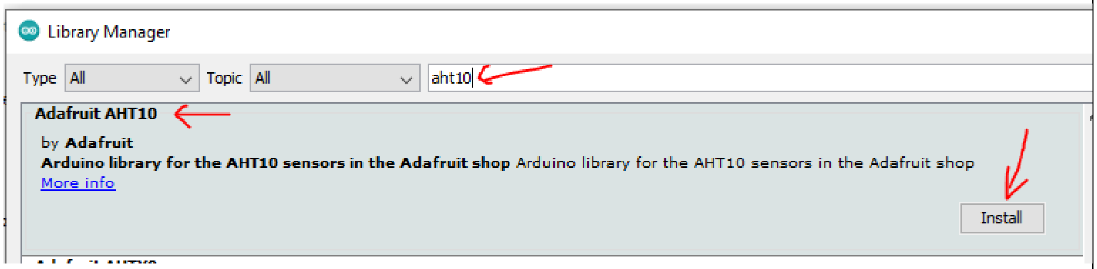
Figure-2.1 Multiple I2C device on one I2C bus

Procedures:

1. The AHT10 should be plugged at Zone-A, the pins are on the breadboard A25a(VIN), A26a(GND), A27a(SCL) and A28a(SDA). Add wires to connect them to OLED VIN, GND, SCL and SDA.

Important: the *pin order* of OLED and AHT10 are not the same.

2. Add library to Arduino for AHT10. 'Tools' 'Manage Libraries…'. Search with AHT10 and install Adafruit AHT10 library (Figure-2.2).
3. Open the official example 'afafruit_aht10_test', it is at the bottom of 'File' 'Examples' …
4. Save the project in a writable folder.
5. Check the setting of Arduino (refer to Lab-2).
6. Open the Serial Monitor of Arduino and change the baud rate to 115200 (refer to Lab-2).
7. Compile and upload the program to EPS32. It should have no error. Please check with Lab-2 procedures if there is any error.
8. You can see the temperature and humidity printed on Serial Monitor as Figure-2.4. Touch the AHT10 sensor (the metal case), the temperature will rise up 1~2 degree centigrade.


Library Manager

Type All Topic All aht10

Adafruit AHT10

by Adafruit

Arduino library for the AHT10 sensors in the Adafruit shop Arduino library for the AHT10 sensors in the Adafruit shop

More info

Install

Figure-2.2 Add AHT10 library

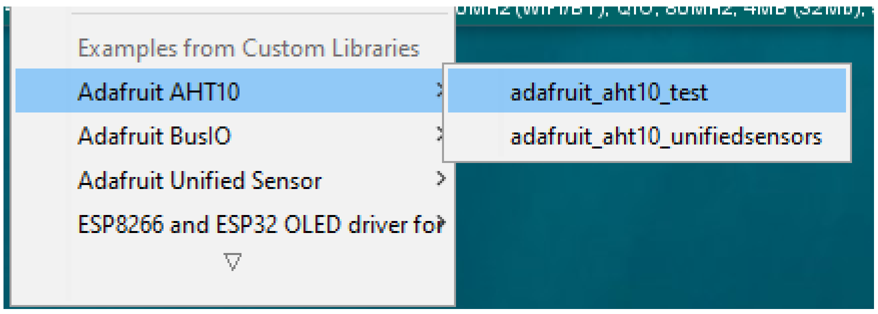
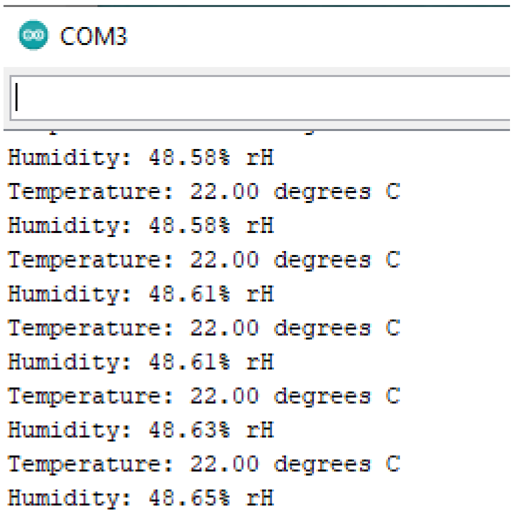

Examples from Custom Libraries
Adafruit AHT10
Adafruit BusIO
Adafruit Unified Sensor
ESP8266 and ESP32 OLED driver for
adafruit_aht10_test
adafruit_aht10_unifiedsensors

Figure-2.3 Open the official example ‘adafruit_aht10_test’

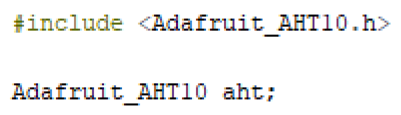

COM3

```
Humidity: 48.58% rH
Temperature: 22.00 degrees C
Humidity: 48.58% rH
Temperature: 22.00 degrees C
Humidity: 48.61% rH
Temperature: 22.00 degrees C
Humidity: 48.61% rH
Temperature: 22.00 degrees C
Humidity: 48.63% rH
Temperature: 22.00 degrees C
Humidity: 48.65% rH
```

Figure-2.4 AHT10 is reporting temperature and humidity.

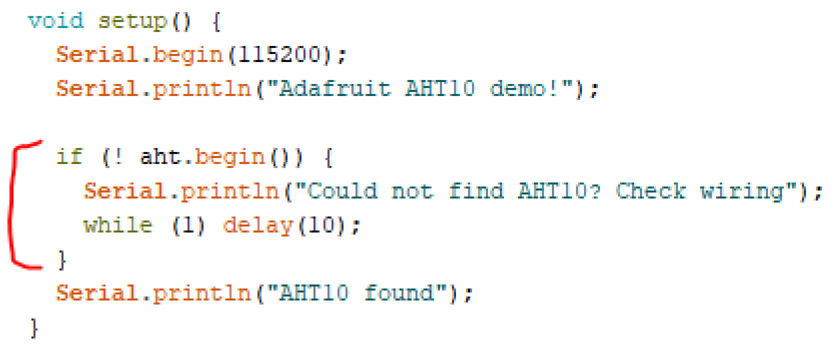
9. Understand the program, the following are new and important.
   - Figure-2.5, include AHT10 library and create an object ‘aht’.
   - Figure-2.6, in setup( ), the portion **if( )** check the AHT10 installed or not.
   - Figure-2.7, in loop( ), the first two lines get the sensor readings. ‘**temp.temperature**’ is datatype float.

.



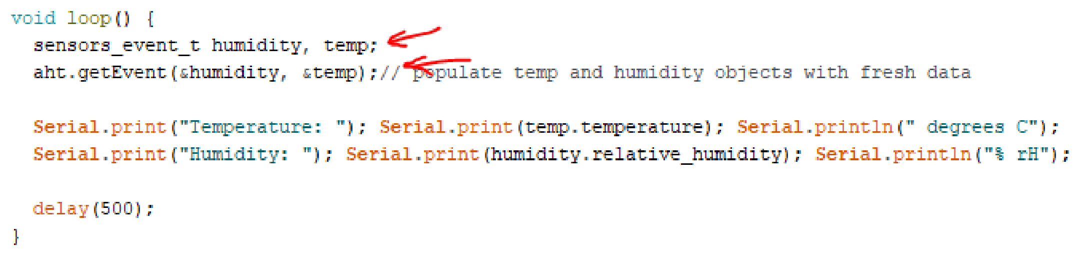
```
#include <Adafruit_AHT10.h>

Adafruit_AHT10 aht;
```

Figure-2.5 include AHT10 library and create object



```
void setup() {
  Serial.begin(115200);
  Serial.println("Adafruit AHT10 demo!");

  if (! aht.begin()) {
    Serial.println("Could not find AHT10? Check wiring");
    while (1) delay(10);
  }
  Serial.println("AHT10 found");
}
```

Figure-2.6 Check AHT10 installed correctly.



```
void loop() {
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);// populate temp and humidity objects with fresh data

  Serial.print("Temperature: "); Serial.print(temp.temperature); Serial.println(" degrees C");
  Serial.print("Humidity: "); Serial.print(humidity.relative_humidity); Serial.println("% rH");

  delay(500);
}
```

Figure-2.7 Get temperature and humidity for AHT10.

Experiment 3.3: Make a standalone thermometer

Up to now, you know how to get temperature from AHT10 and output something to OLED display module and Serial Monitor.

Combine Experiment 3.2 and 3.3 to make a standalone thermometer to show temperature on OLED display module and Serial Monitor.

Procedures:

1. Create a new empty project and save it to a writable folder.
2. Copy the portions #define, #include, initializations and the part of loop( ) from Experiments 2.2 and 2.3.
3. Compile and upload your program to ESP32.

Notes:

- '**temp.temperature**' is datatype float, you can print it on OLED as Figure-1.14.

~ END ~