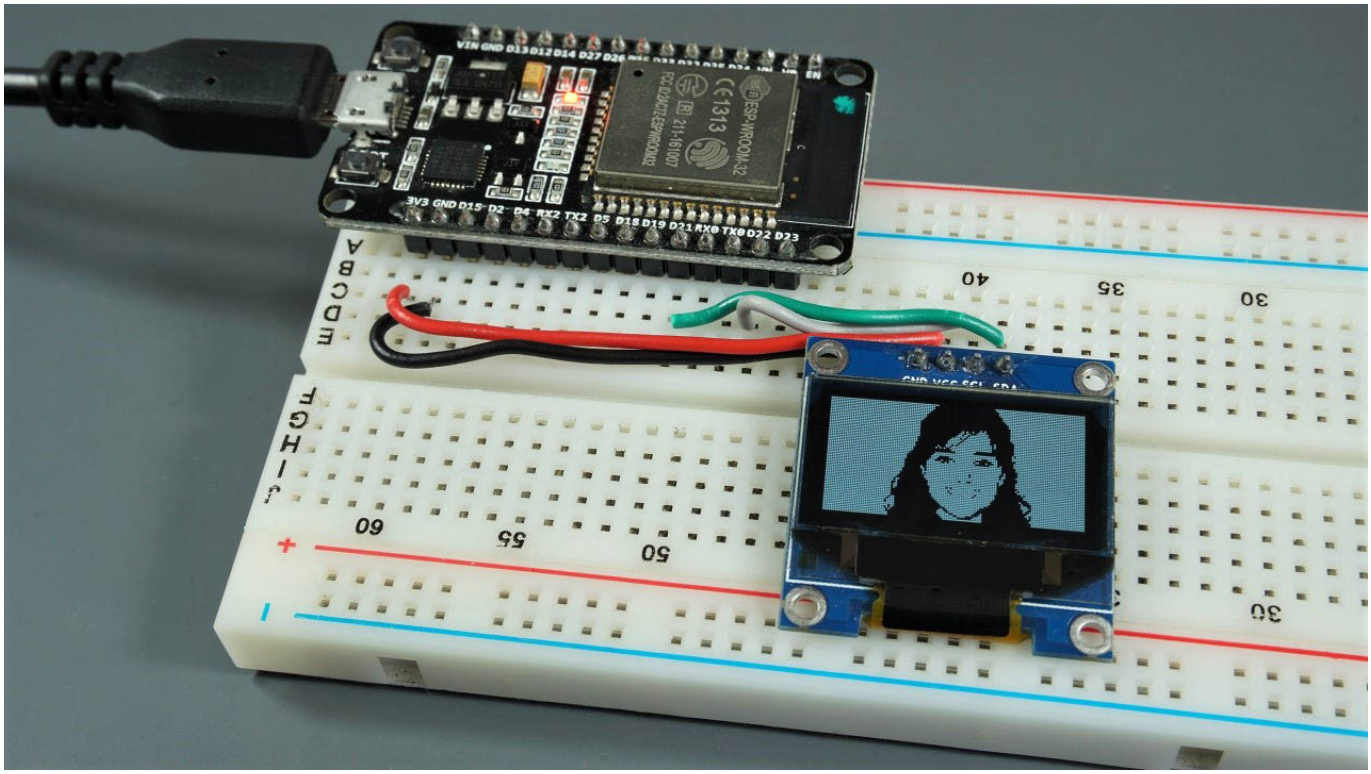
ESP32 OLED Display with Arduino IDE

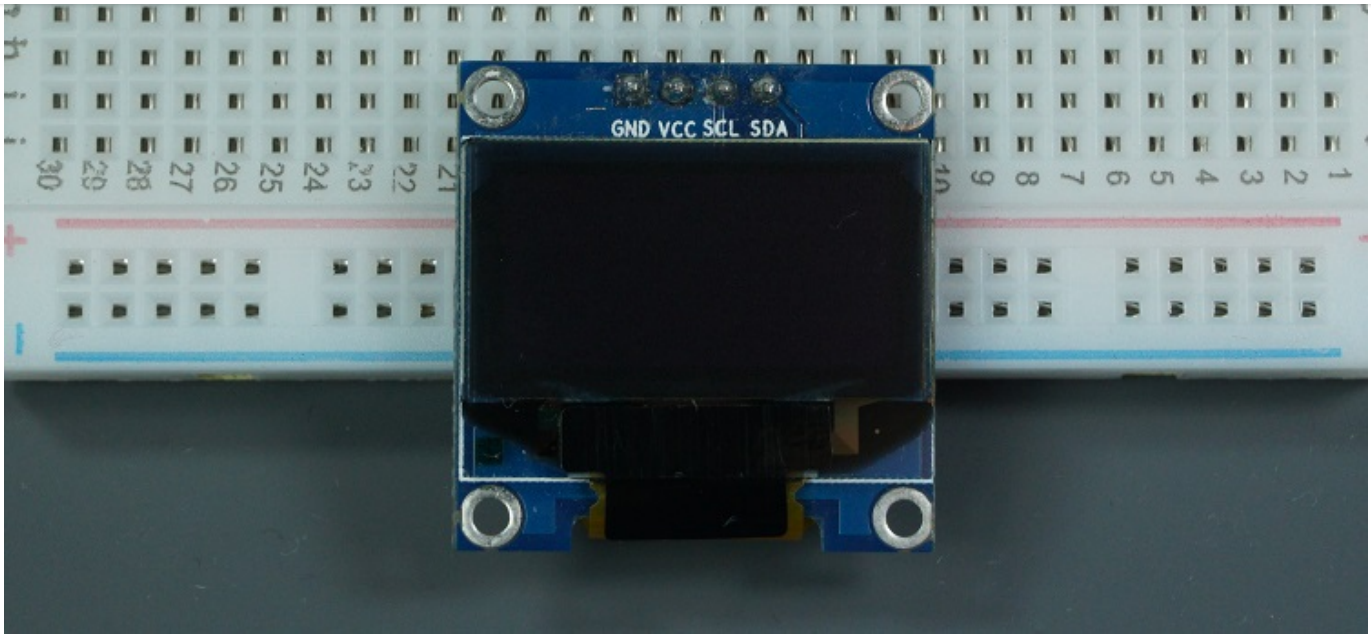
This guide shows how to use the 0.96 inch SSD1306 OLED display with ESP32 using Arduino IDE. We'll show you how to write text, set different fonts, draw shapes and display bitmaps images.



We also have a dedicated guide that shows how to display temperature and humidity readings using DHT sensor and ESP32.

Introducing 0.96 inch OLED Display

The OLED display that we'll use in this tutorial is the SSD1306 model: a monocolor, 0.96 inch display with 128×64 pixels as shown in the following figure.



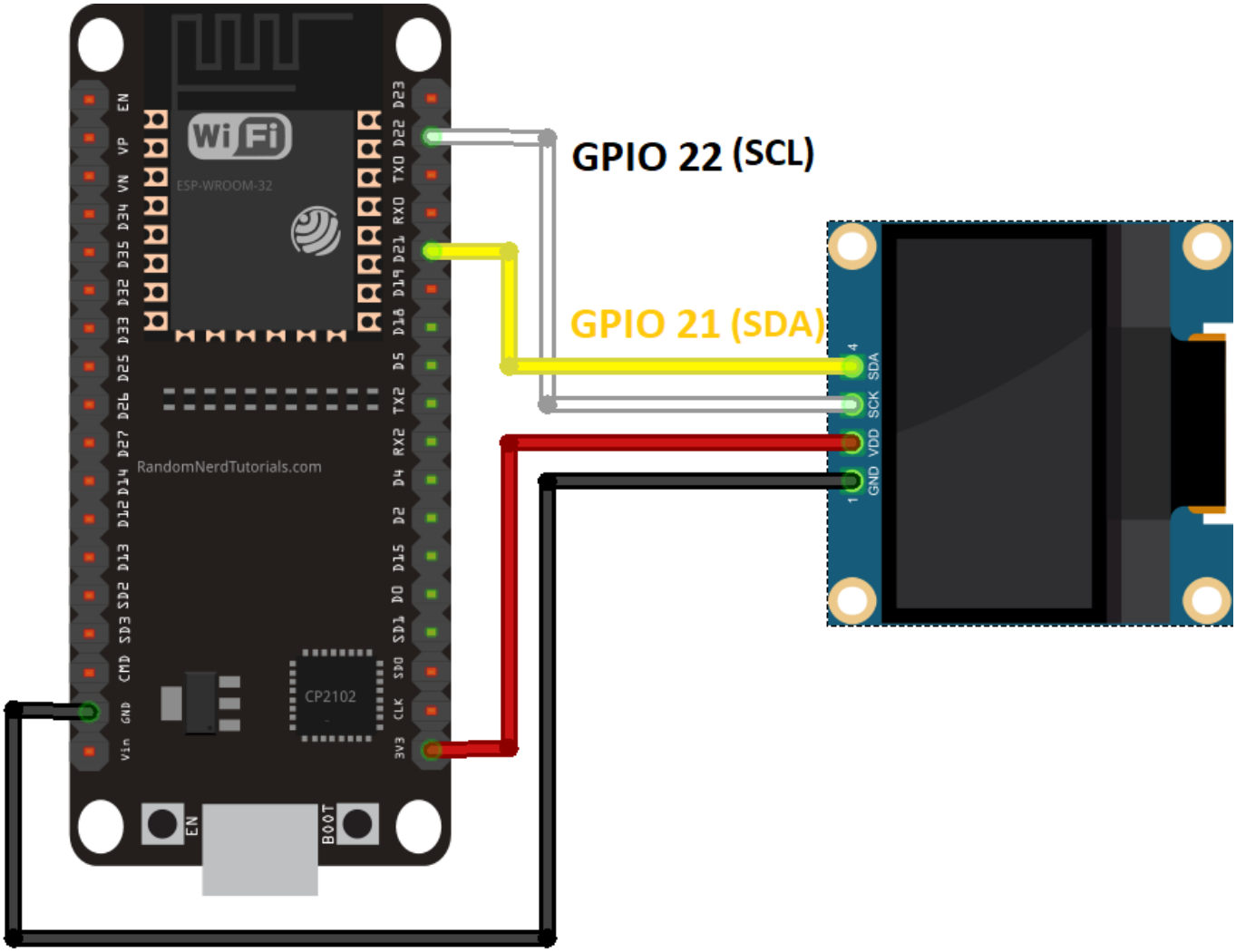
The OLED display doesn't require backlight, which results in a very nice contrast in dark environments. Additionally, its pixels consume energy only when they are on, so the OLED display consumes less power when compared to other displays.

The model we're using has four pins and communicates with any microcontroller using I2C communication protocol. There are models that come with an extra RESET pin or that communicate using SPI communication protocol.

OLED Display SSD1306 Pin Wiring

Because the OLED display uses I2C communication protocol, wiring is very simple. You can use the following table as a reference.

Pin	ESP32
Vin	3.3V
GND	GND
SCL	GPIO 22
SDA	GPIO 21



In this example, we're using I2C communication protocol. The most suitable pins for I2C communication in the ESP32 are GPIO 22 (SCL) and GPIO 21 (SDA).

If you're using an OLED display with SPI communication protocol, use the following GPIOs.

- GPIO 18: CLK
- GPIO 19: MISO
- GPIO 23: MOSI
- GPIO 5: CS

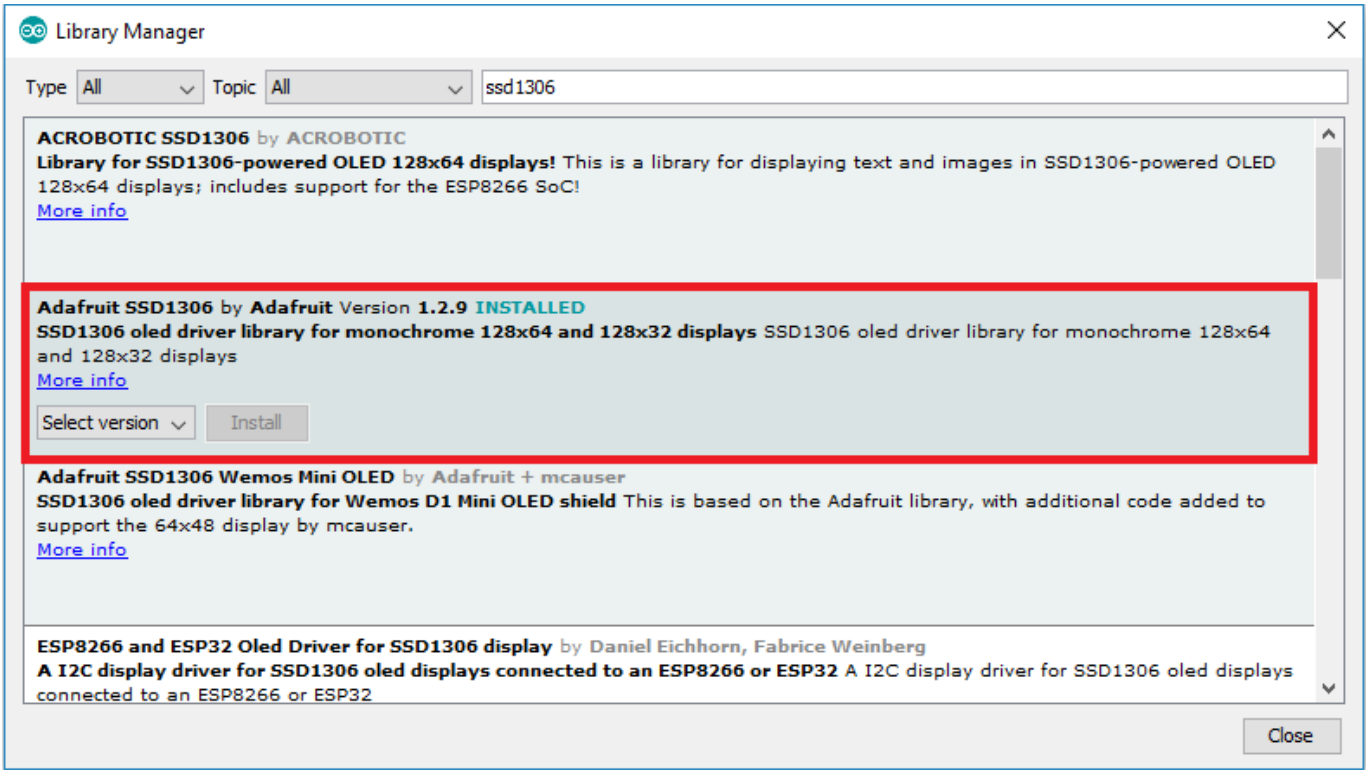
Read our ESP32 Pinout Reference Guide to learn more about the ESP32 GPIOs.

There are several libraries available to control the OLED display with the ESP32. In this tutorial we'll use two Adafruit libraries: Adafruit_SSD1306 library and Adafruit_GFX library.

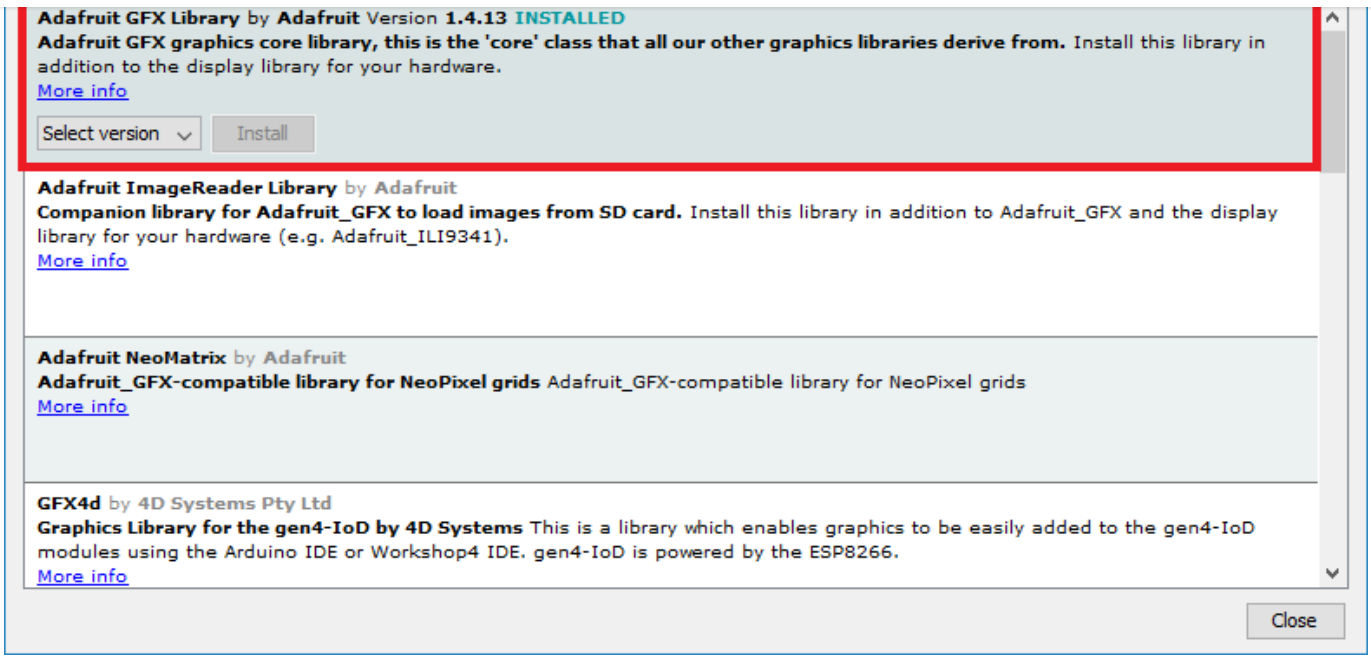
Follow the next steps to install those libraries.

1. Open your Arduino IDE and go to **Sketch** > **Include Library** > **Manage Libraries**. The Library Manager should open.

2. Type "**SSD1306**" in the search box and install the SSD1306 library from Adafruit.



3. After installing the SSD1306 library from Adafruit, type "**GFX**" in the search box and install the library.



4. After installing the libraries, restart your Arduino IDE.

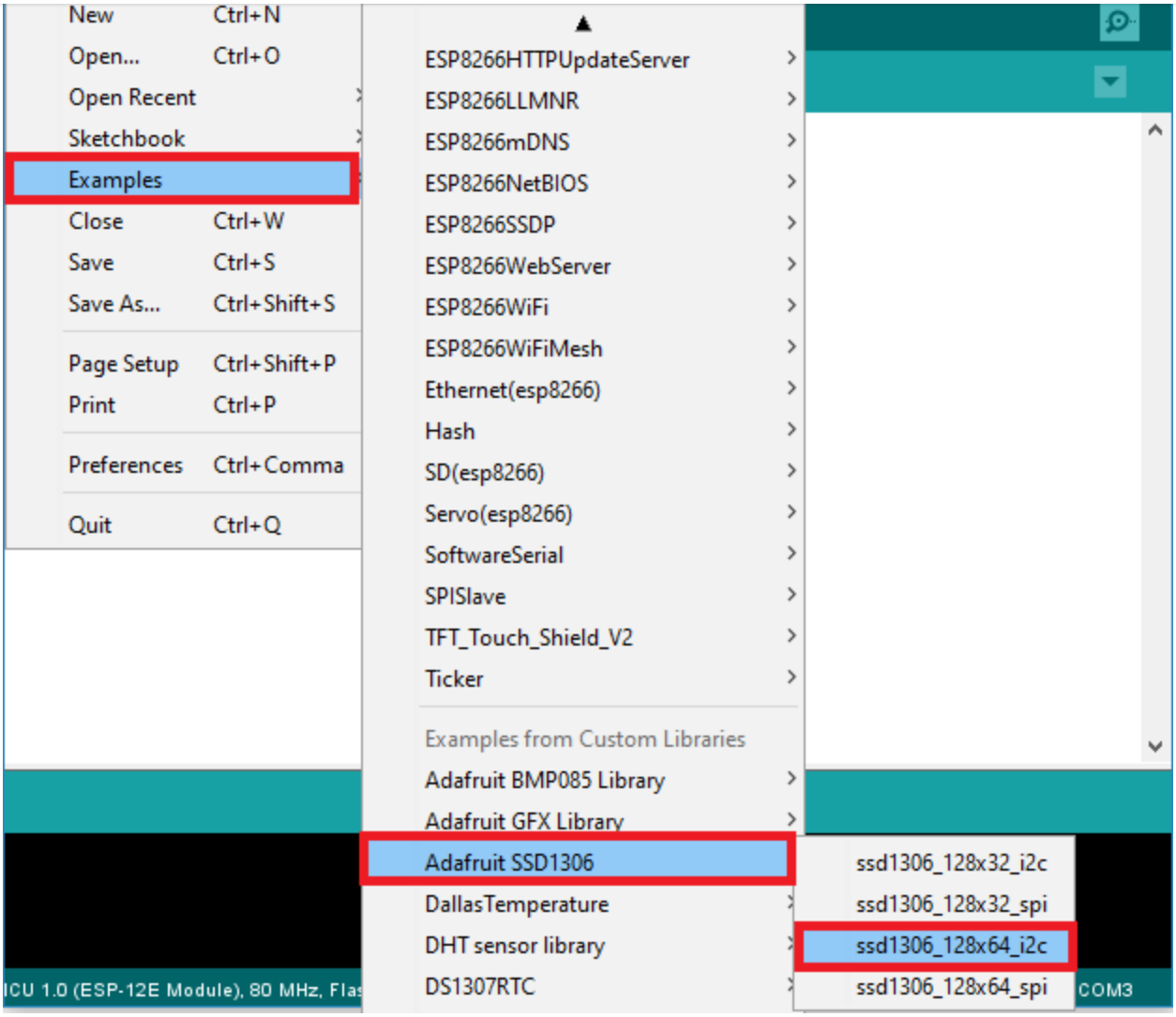
We'll program the ESP32 using Arduino IDE, so you must have the ESP32 add-on installed in your Arduino IDE. If you haven't, follow the next tutorial first:

- Install the ESP32 Board in Arduino IDE (Windows instructions)
- Install the ESP32 Board in Arduino IDE (Mac OS X and Linux instructions)

Testing OLED Display with ESP32

After wiring the OLED display to the ESP32 and installing all required libraries, you can use one example from the library to see if everything is working properly.

In your Arduino IDE, go to **File** > **Examples** > **Adafruit SSD1306** and select the example for the display you're using.



The following code should load:

```
  B11111110, B11111000,
  B01111110, B11111111,
  B00110011, B10011111,
  B00011111, B11111100,
  B00001101, B01110000,
  B00011011, B10100000,
  B00111111, B11100000,
  B00111111, B11110000,
  B01111100, B11110000,
  B01110000, B01110000,
  B00000000, B00110000 };

void setup() {
  Serial.begin(115200);
```

View raw code

If your OLED doesn't have a RESET pin, you should set the OLED_RESET variable to -1 as shown below:

```
#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino
```

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

Upload the code to your ESP32 board. Don't forget to select the right board and COM port in the **Tools** menu.

00:26

If your OLED display is not showing anything:

- Check that the OLED display is properly wired to the ESP32
- Double-check the OLED display I2C address: with the OLED connected to the ESP32, upload this code and check the I2C address in the Serial Monitor

You should change the OLED address in the following line, if necessary. In our case, the address is 0x3C.

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
```

Write Text – OLED Display

The Adafruit library for the OLED display comes with several functions to write text. In this section, you'll learn how to write and scroll text using the library functions.

The following sketch displays "Hello, world!" message in the OLED display.

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, S
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -

void setup() {
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  // Display static text
  display.println("Hello, world!");
  display.display();
}

void loop() {

}
```

View raw code

After uploading the code, this is what you'll get in your OLED:

Let's take a quick look on how the code works.

Importing libraries

First, you need to import the necessary libraries. The `Wire` library to use I2C and the Adafruit libraries to write to the display: `Adafruit_GFX` and `Adafruit_SSD1306`.

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

Initialize the OLED display

Then, you define your OLED width and height. In this example, we're using a 128×64 OLED display. If you're using other sizes, you can change that in the `SCREEN_WIDTH`, and `SCREEN_HEIGHT` variables.

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -
```

The (-1) parameter means that your OLED display doesn't have a RESET pin. If your OLED display does have a RESET pin, it should be connected to a GPIO. In that case, you should pass the GPIO number as a parameter.

In the `setup()`, initialize the Serial Monitor at a baud raute of 115200 for debugging purposes.

```
Serial.begin(115200);
```

Initialize the OLED display with the `begin()` method as follows:

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println("SSD1306 allocation failed");
  for(;;); // Don't proceed, loop forever
}
```

This snippet also prints a message on the Serial Monitor, in case we're not able to connect to the display.

```
Serial.println("SSD1306 allocation failed");
```

In case you're using a different OLED display, you may need to change the OLED address. In our case, the address is `0x3C`.

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
```

After initializing the display, add a two second delay, so that the OLED has enough time to initialize before writing text:

```
delay(2000);
```

Clear display, set font size, color and write text

After initializing the display, clear the display buffer with the `clearDisplay()` method:

```
display.clearDisplay();
```

Before writing text, you need to set the text size, color and where the text will be displayed in the OLED.

Set the font size using the `setTextSize()` method:

```
display.setTextSize(1);
```

Set the font color with the `setTextColor()` method:

```
display.setTextColor(WHITE);
```

`WHITE` sets white font and black background.

Define the position where the text starts using the `setCursor(x,y)` method. In this case, we're setting the text to start at the (0,0) coordinates – at the top left corner.

Finally, you can send the text to the display using the `println()` method, as follows:

```
display.println("Hello, world!");
```

Then, you need to call the `display()` method to actually display the text on the screen.

```
display.display();
```

Scrolling Text

The Adafruit OLED library provides useful methods to easily scroll text.

- `startscrollright(0x00, 0x0F)` : scroll text from left to right
- `startscrollleft(0x00, 0x0F)` : scroll text from right to left
- `startscrolldiagright(0x00, 0x07)` : scroll text from left bottom corner to right upper corner
- `startscrolldiagleft(0x00, 0x07)` : scroll text from right bottom corner to left upper corner

The following sketch implements those methods.

```
  delay(100);

}

void loop() {
  // Scroll in various directions, pausing in-between:
  display.startscrollright(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrolldiagright(0x00, 0x07);
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
}
```

View raw code

The text scrolls as shown in the following short video.

00:06

Using Other Fonts – OLED Display

The Adafruit GFX library allows us to use some alternate fonts besides the built-in fonts. It allows you to chose between Serif, Sans, and Mono. Each font is available in bold, italic and in different sizes.

The sizes are set by the actual font. So, the `setTextSize()` method doesn't work with these fonts. The fonts are available in 9, 12, 18 and 24 point sizes and also contain 7-bit characters (ASCII codes) (described as 7b in the font name).

You can chose from the next selection of fonts:

```
FreeMono12pt7b.h                FreeSansBoldOblique12pt7b.h
FreeMono18pt7b.h                FreeSansBoldOblique18pt7b.h
FreeMono24pt7b.h                FreeSansBoldOblique24pt7b.h
FreeMono9pt7b.h                 FreeSansBoldOblique9pt7b.h
FreeMonoBold12pt7b.h            FreeSansOblique12pt7b.h
FreeMonoBold18pt7b.h            FreeSansOblique18pt7b.h
FreeMonoBold24pt7b.h            FreeSansOblique24pt7b.h
FreeMonoBold9pt7b.h             FreeSansOblique9pt7b.h
FreeMonoBoldOblique12pt7b.h     FreeSerif12pt7b.h
FreeMonoBoldOblique18pt7b.h     FreeSerif18pt7b.h
```

```
FreeMonoOblique18pt7b.h         FreeSerifBold18pt7b.h
FreeMonoOblique24pt7b.h         FreeSerifBold24pt7b.h
FreeMonoOblique9pt7b.h          FreeSerifBold9pt7b.h
FreeSans12pt7b.h                FreeSerifBoldItalic12pt7b.h
FreeSans18pt7b.h                FreeSerifBoldItalic18pt7b.h
FreeSans24pt7b.h                FreeSerifBoldItalic24pt7b.h
FreeSans9pt7b.h                 FreeSerifBoldItalic9pt7b.h
FreeSansBold12pt7b.h            FreeSerifItalic12pt7b.h
FreeSansBold18pt7b.h            FreeSerifItalic18pt7b.h
FreeSansBold24pt7b.h            FreeSerifItalic24pt7b.h
FreeSansBold9pt7b.h             FreeSerifItalic9pt7b.h
```

The fonts that work better with the OLED display are the 9 and 12 points size.

To use one of those fonts, first you need to include it in your sketch, for example:

```
#include <Fonts/FreeSerif12pt7b.h>
```

Next, you just need to use the `setFont()` method and pass as argument, the specified font:

```
display.setFont(&FreeSerif12pt7b);
```

After specifying the font, all methods to write text will use that font. To get back to use the original font, you just need to call the `setFont()` method with no arguments:

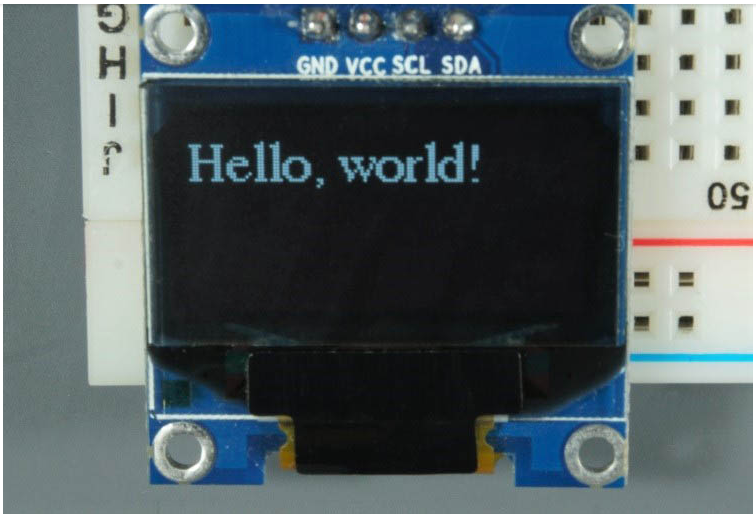
```
display.setFont();
```

Upload the next sketch to your board:

```
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, S
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -

void setup() {
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("SSD1306 allocation failed");
    for(;;);
  }
  delay(2000);

  display.setFont(&FreeSerif9pt7b);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,20);
  display.println("Hello, world!");
  display.display();
  delay(2000);
}
```

View raw code

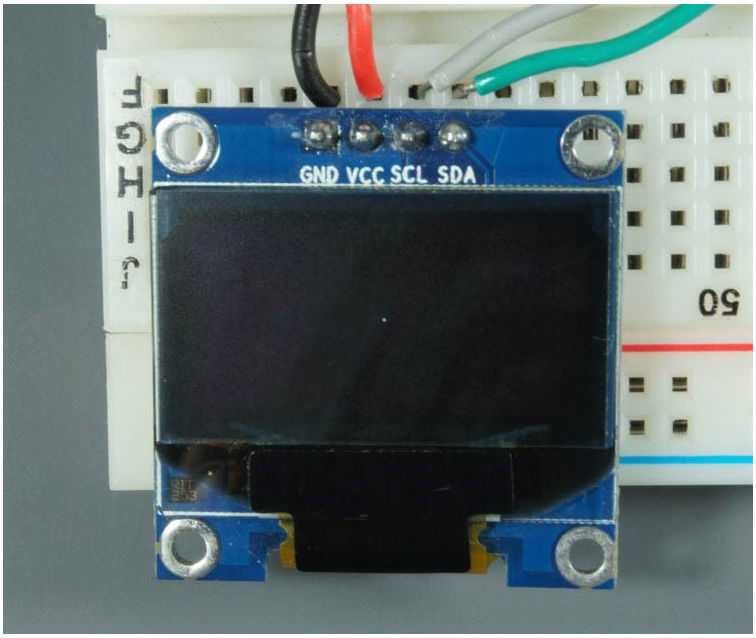
Now, your display prints the “Hello, world!” message in FreeSerif font.



Draw Shapes in the OLED Display

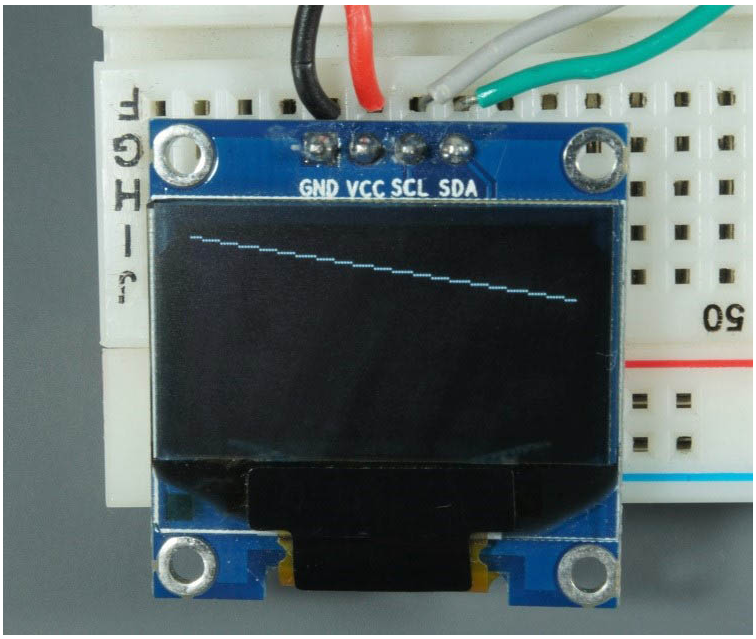
The Adafruit OLED library provides useful methods to draw pixels, lines and shapes. Let's take a quick look at those methods.

Draw a pixel



To draw a pixel in the OLED display, you can use the `drawPixel(x, y, color)` method that accepts as arguments the x and y coordinates where the pixel appears, and color. For example:

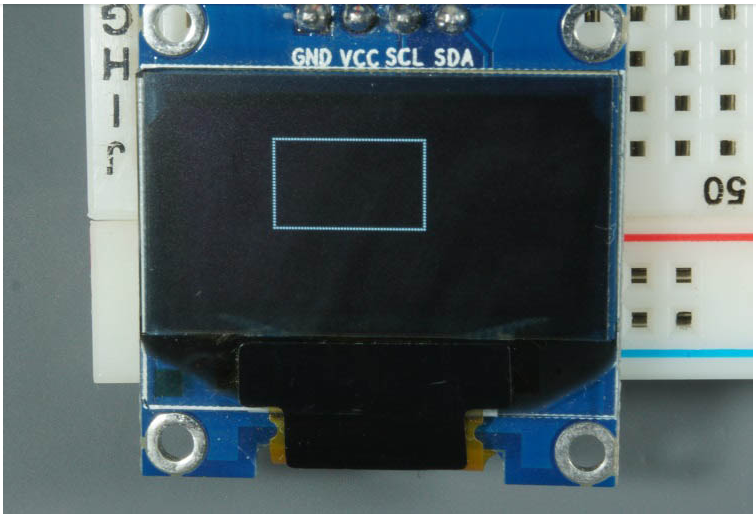
Draw a line



Use the `drawLine(x1, y1, x2, y2, color)` method to create a line. The (x1, y1) coordinates indicate the start of the line, and the (x2, y2) coordinates indicates where the line ends. For example:

```
display.drawLine(0, 0, 127, 20, WHITE);
```

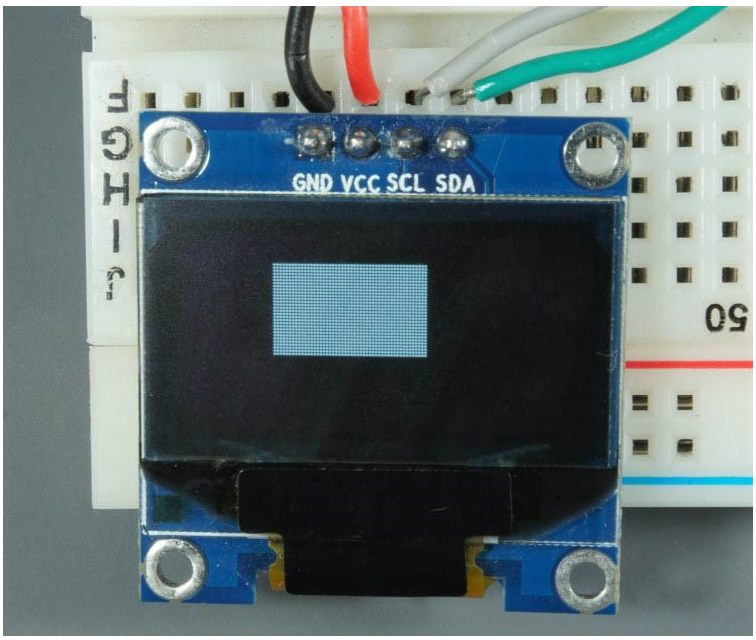
Draw a rectangle



The `drawRect(x, y, width, height, color)` provides an easy way to draw a rectangle. The (x, y) coordinates indicate the top left corner of the rectangle. Then, you need to specify the width, height and color:

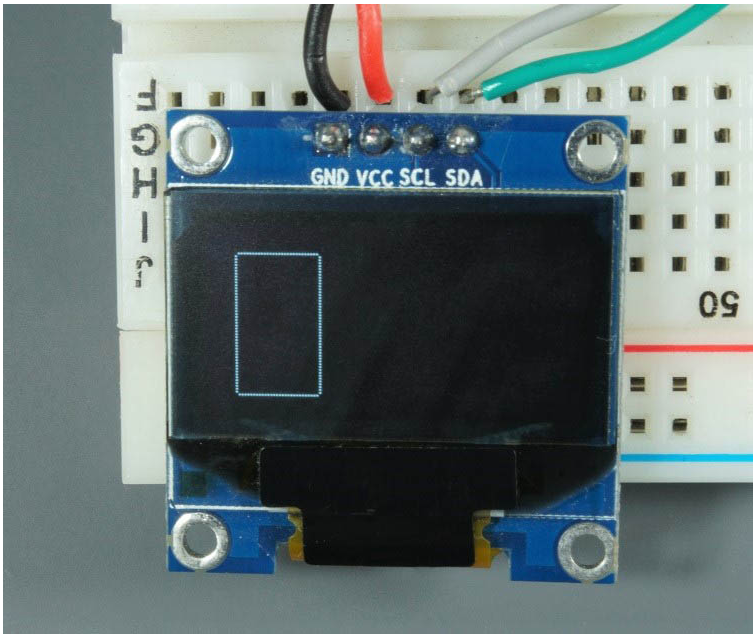
```
display.drawRect(10, 10, 50, 30, WHITE);
```

You can use the `fillRect(x, y, width, height, color)` to draw a filled rectangle. This method accepts the same arguments as `drawRect()` .



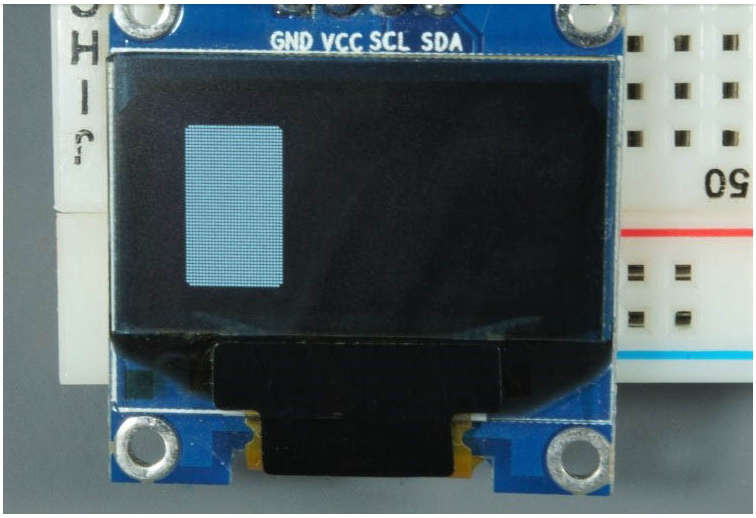
arguments as previous methods plus the radius of the corner. For example:

```
display.drawRoundRect(10, 10, 30, 50, 2, WHITE);
```

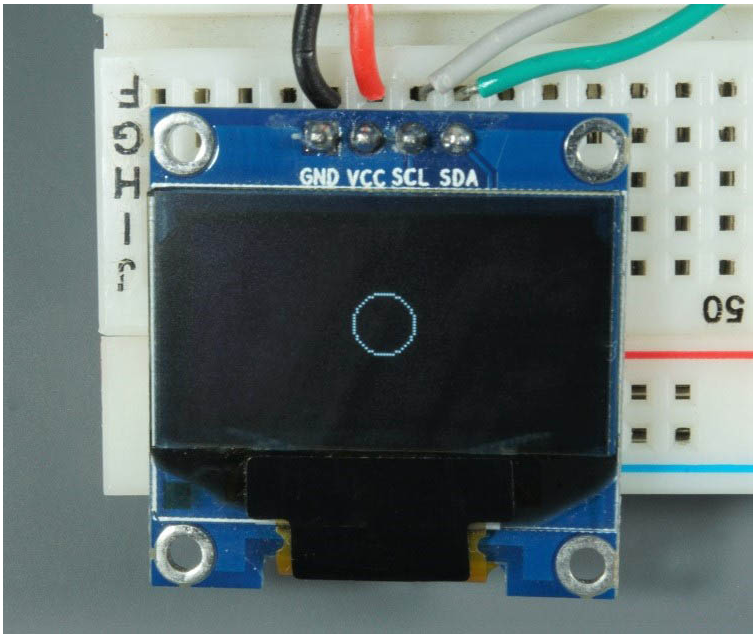


Or a filled round rectangle:

```
display.fillRoundRect(10, 10, 30, 50, 2, WHITE);
```



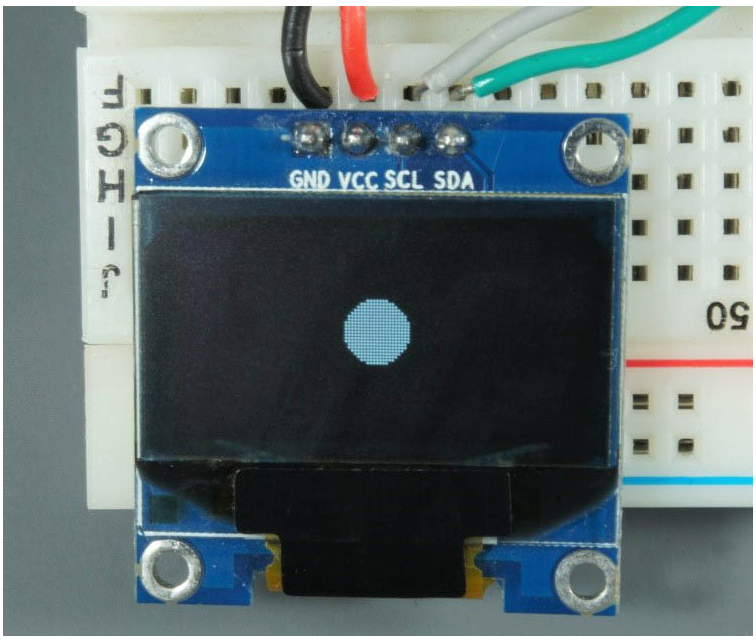
Draw a circle



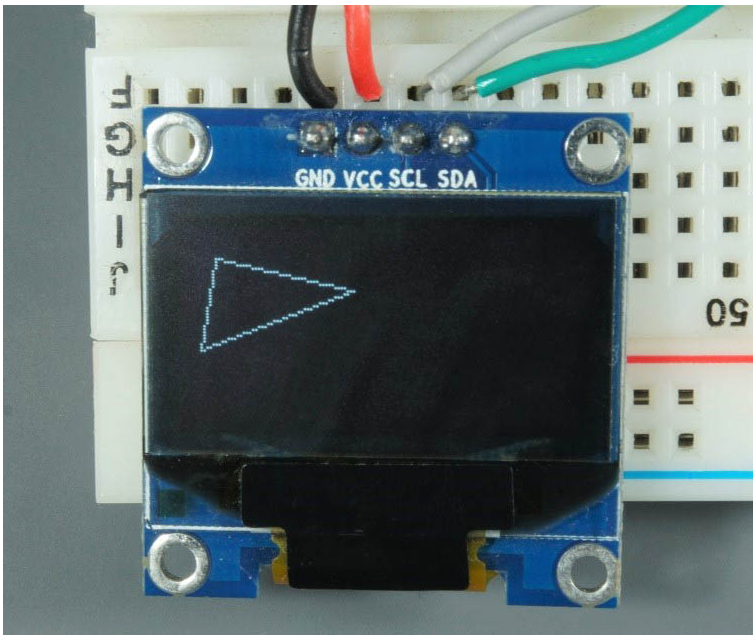
To draw a circle use the `drawCircle(x, y, radius, color)` method. The (x,y) coordinates indicate the center of the circle. You should also pass the radius as an argument. For example:

```
display.drawCircle(64, 32, 10, WHITE);
```

In the same way, to build a filled circle, use the `fillCircle()` method with the same arguments:



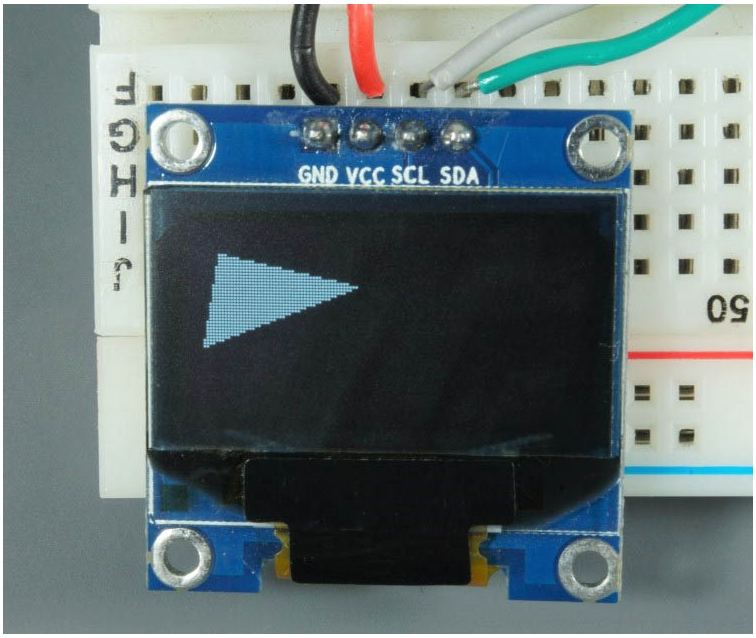
Draw a triangle



Use the the `drawTriangle(x1, y1, x2, y2, x3, y3, color)` method to build a triangle. This method accepts as arguments the coordinates of each corner and the color.

```
display.drawTriangle(10, 10, 55, 20, 5, 40, WHITE);
```

```
display.fillTriangle(10, 10, 55, 20, 5, 40, WHITE);
```



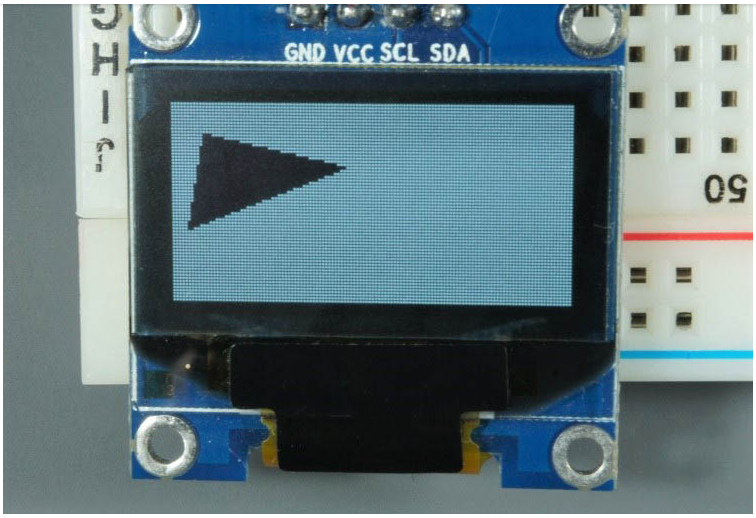
Invert

The library provides an additional method that you can use with shapes or text: the `invertDisplay()` method. Pass `true` as argument to invert the colors of the screen or `false` to get back to the original colors.

If you call the following command after defining the triangle:

```
display.invertDisplay(true);
```

You'll get an inverted triangle as follows:



Code – Draw Shapes

Upload the following sketch that implements each snippet of code we've covered previously and goes through all the shapes.

```
/*********
  Rui Santos
  Complete project details at https://randomnerdtutorials.com
*********/

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

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Declaration for an SSD1306 display connected to I2C (SDA, S
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -

void setup() {
  Serial.begin(115200);
```

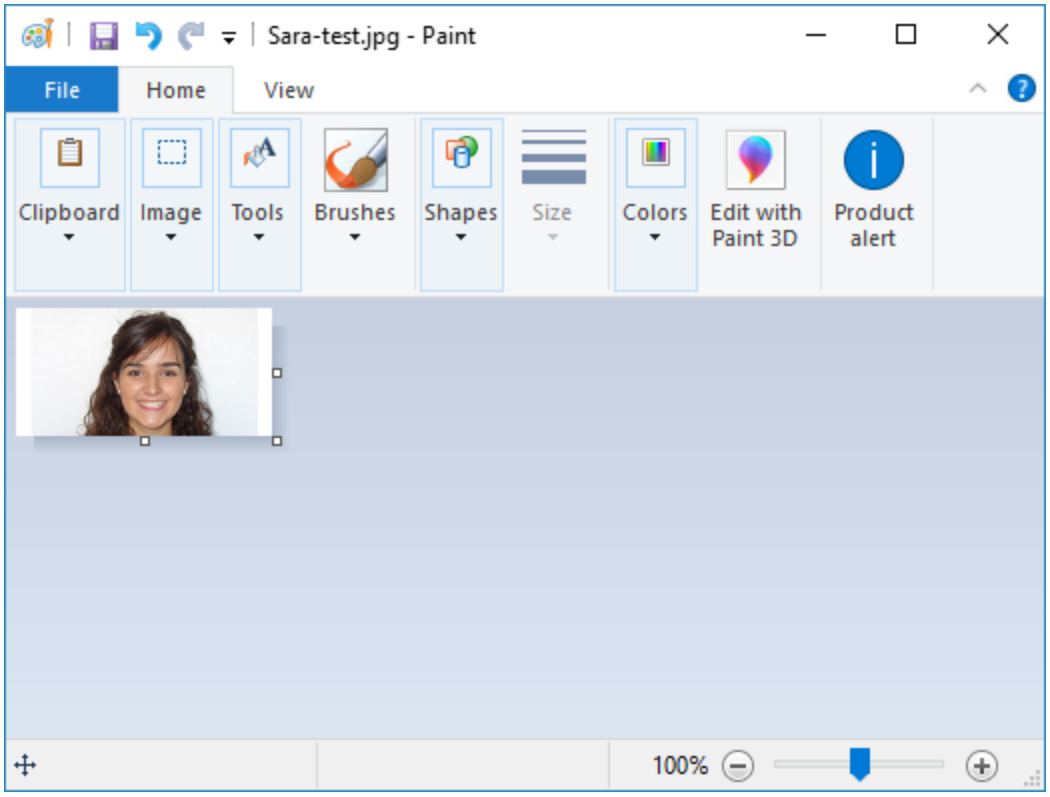
```
  for(;;);
}
delay(2000); // Pause for 2 seconds

// Clear the buffer
display.clearDisplay();
```

View raw code

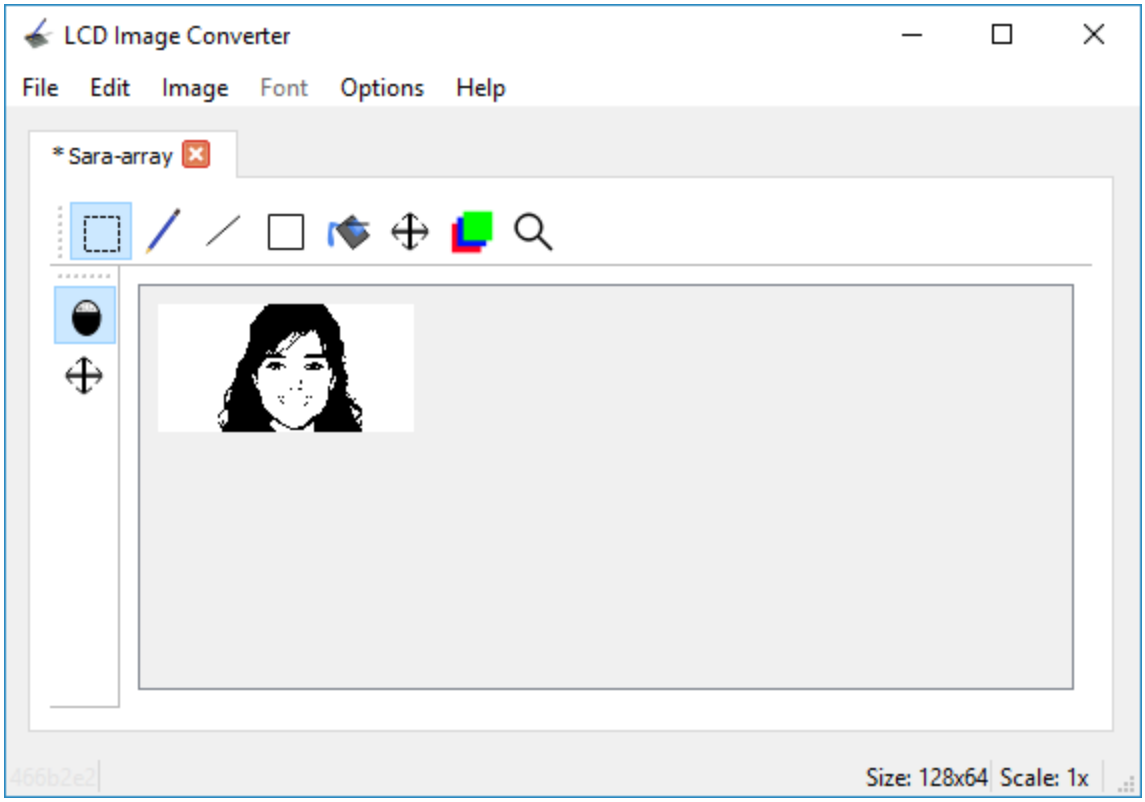
Display Bitmap Images in the OLED

You can display 128×64 bitmap monocolor images on the OLED display.

First, use an imaging program to resize a photo or picture and **save it as monochrome bitmap**. If you're on a Windows PC, you can use Paint.

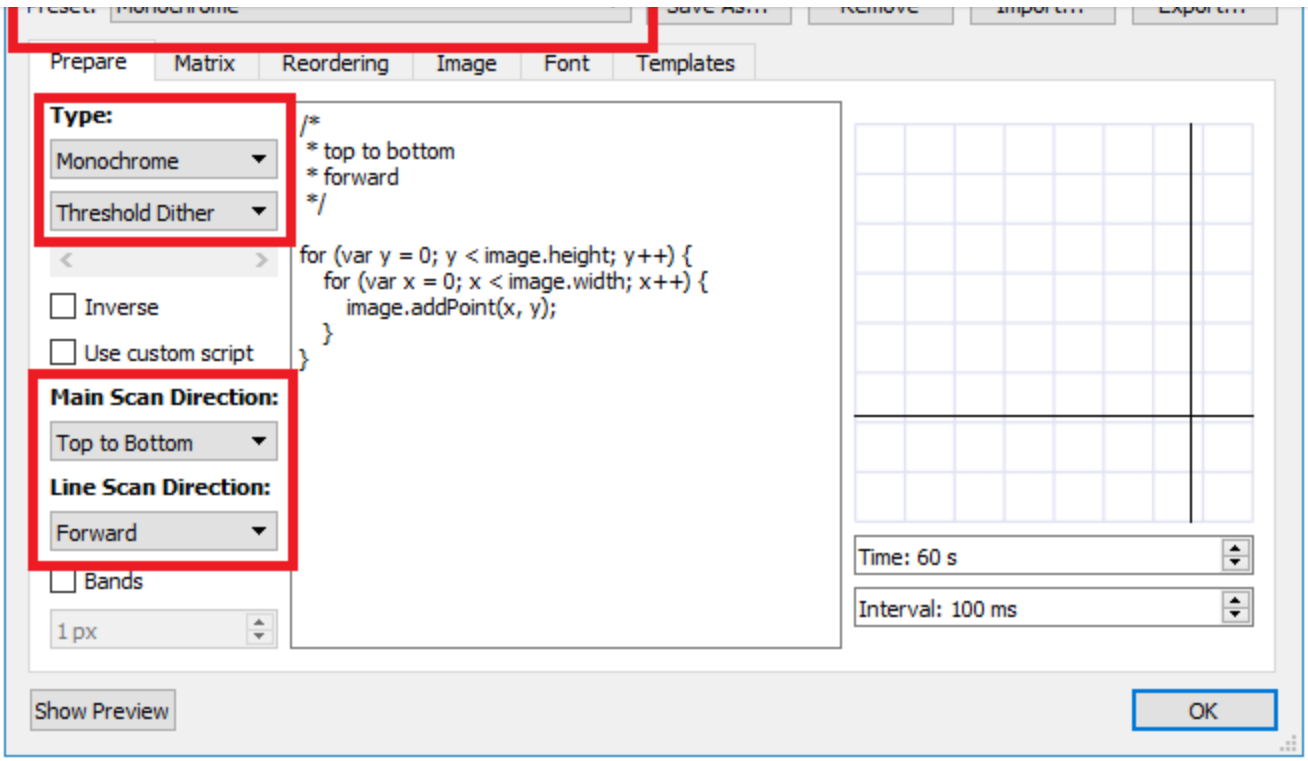


Run the program and start with a new image. Go to **Image** > **Import** and select the bitmap image you've created earlier.



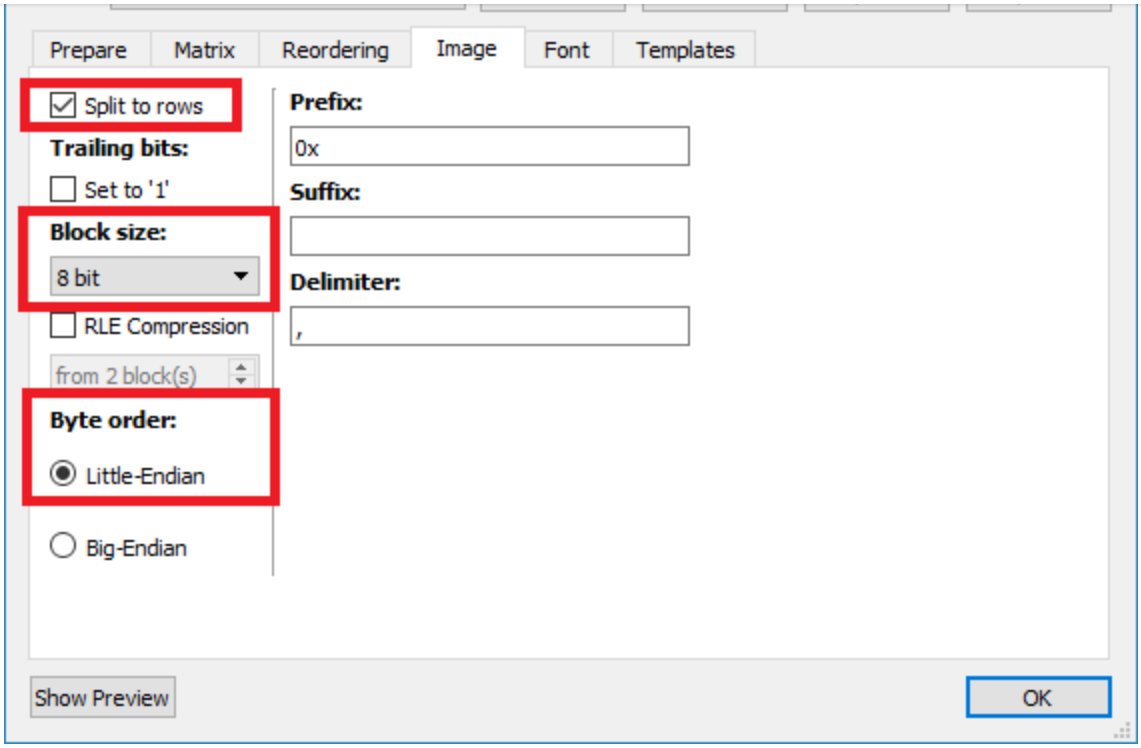
Go to **Options** > **Conversion** and in the **Prepare** tab, select the following options:

- **Type:** Monochrome, Threshold Dither
- **Main Scan Direction:** Top to Bottom
- **Line Scan Direction:** Forward



Go to the Image tab and select the following options:

- Split to rows
- **Block size:** 8 bit
- **Byte order:** Little-Endian



Then, click **OK**. Finally, in the main menu, go to **File** > **Convert**. A new file with *.c* extension should be saved. That file contains the C array for the image. Open that file with a text editor, and copy the array.

In our case, this is the array that we get:

```
static const uint8_t image_data_Saraarray[1024] = {
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00
```

```
  0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x0
  0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x14, 0x9
  0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x36, 0x3
  0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x6d, 0xf
  0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0xfb, 0xf
  0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x03, 0xd7, 0xf
  0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x07, 0xef, 0xf
  0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x0f, 0xdf, 0xf
  0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x0f, 0xbf, 0xf
  0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x1d, 0x7f, 0xf
  0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x01, 0x1b, 0xff, 0xf
  0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x02, 0xa7, 0xff, 0xf
```

View raw code

Copy your array to the sketch. Then, to display the array, use the `drawBitmap()` method that accepts the following arguments (x, y, image array, image width, image height, rotation). The (x, y) coordinates define where the image starts to be displayed.

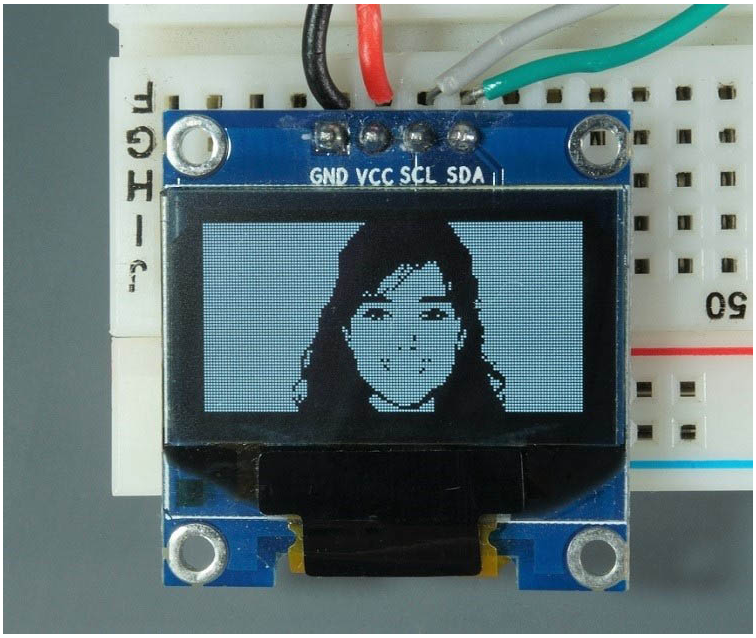
Copy the code below to display your bitmap image in the OLED.

```
/*********
  Rui Santos
  Complete project details at https://randomnerdtutorials.com
*********/

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

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
```

```
static const uint8_t image_data_Saraarray[1024] = {
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00
    0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00
```

View raw code

After uploading the code, this is what we get on the display.



Troubleshooting

If you get the “**SSD1306 allocation failed**” error or if the OLED is not displaying anything in the screen, it can be one of the following issues:

The I2C address for the OLED display we are using is 0x3C. However, yours may be different. So, make sure you check your display I2C address using an I2C scanner sketch.

SDA and SCL not connected properly

Please make sure that you have the SDA and SCL pins of the OLED display wired correctly. In case of the ESP32, connect SDA pin to GPIO 21 and SCL pin to GPIO 22 .

Wrapping Up

We hope you've found this guide about the OLED display with EPS8266 useful. Now, you can integrate the OLED display in your own projects. Proceed to the next tutorial to learn how to display sensor readings on the OLED display:

- ESP32 – Display Sensor Readings OLED Display

If you like ESP32, you'll certainly like our ESP32 resources:

- **Learn ESP32 with Arduino IDE (eBook and video course)**
- MicroPython Programming with ESP32 (eBook)
- ESP32 Pinout Reference: Which GPIO pins should you use?

Some of our most popular projects with ESP board:

- Build a Multisensor Shield for ESP8266
- ESP32 Web Server with Arduino IDE
- ESP32 DHT22 Web Server
- Low Power Weather Station Datalogger

Thanks for reading.