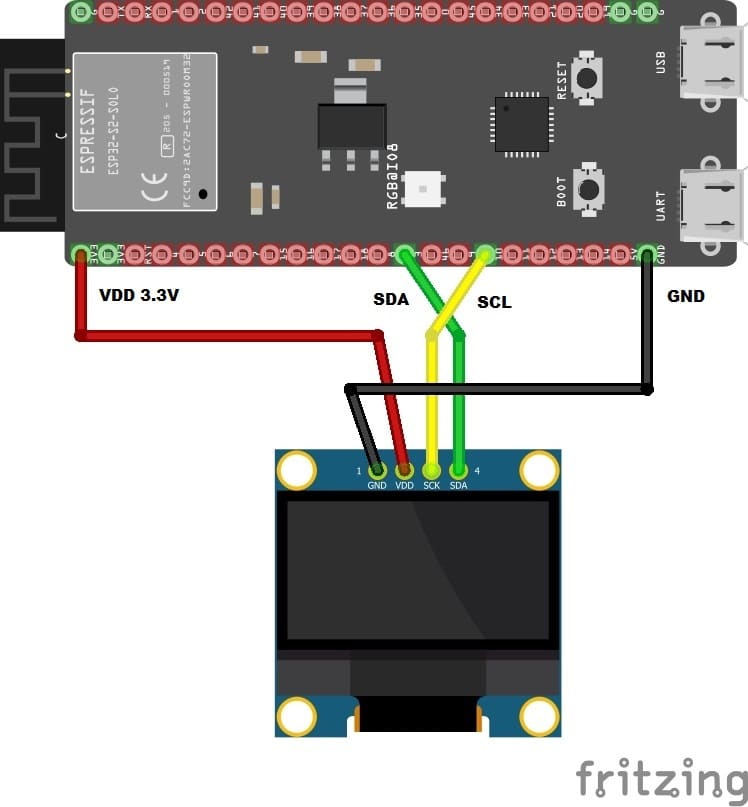
*Want to display text or graphics with your OLED display using an ESP32? Interested in knowing how to do it in VS Code? Read the article to find out.*

**INTRODUCTION**

OLEDs are a great way to display data from your microcontroller to the outside world. Some of our previous [blogs](https://www.phippselectronics.com/displaying-temperature-and-humidity-on-an-oled-screen/" \t "_blank) have already done and explained the process of doing it. However, how do you do it on an ESP32 device? Here, you’ll learn the specifics.

**CONNECTING YOUR OLED WITH YOUR ESP32 DEVICE**

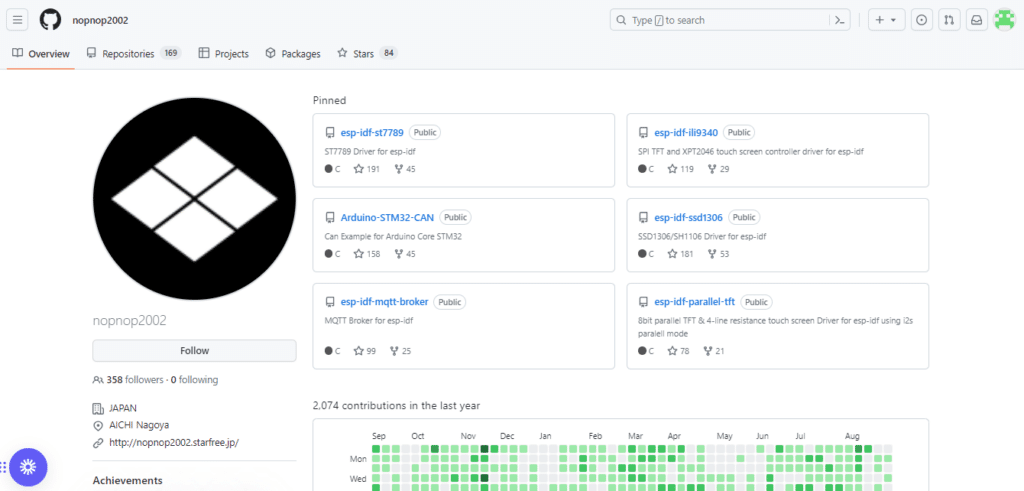
Wether it’s an ESP32, ESP32-S2, ESP32-S3, or other models, the process of connecting your OLED screen should be similar. What’s important is to find your ESP32’s I2C port. After that, directly connect this port to your OLED display’s I2C port. Here, we have pin 9 to be SCL and pin 8 to be SDA on ESP32-S2 or S3 DevKitC.



Additionally, connect the VCC and GND terminals of the OLED display to your ESP32’s system power. A typical 128×64 OLED can be powered with a +3.3V or 5V supply. With this,  connect your OLED’s power terminals to the +3.3V and GND terminals of your ESP32 component or devkit.

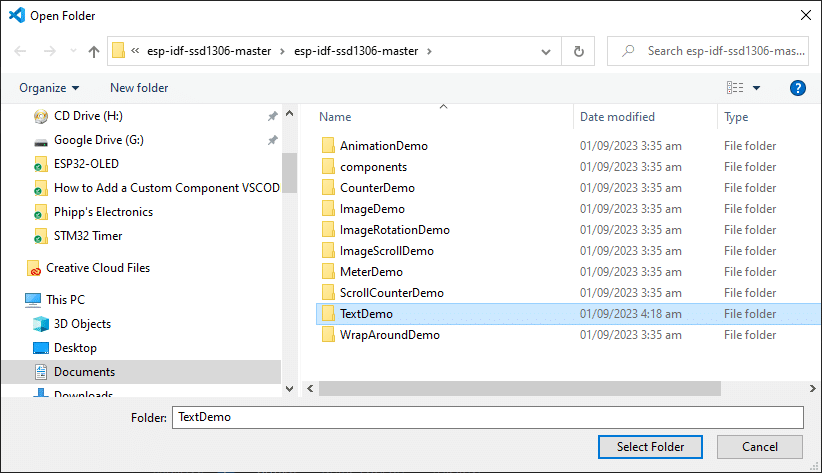
**USING VSCODE TO PROGRAM THE DISPLAY OF YOUR OLED**

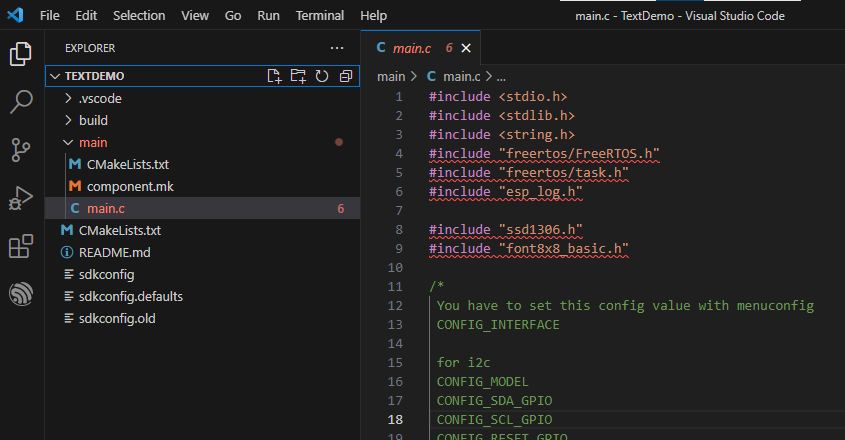
Previously, the way to use [components on ESP-IDF](https://www.phippselectronics.com/how-to-add-a-component-in-esp-idf-in-vscode/" \t "_blank) was discussed. This was followed by an example of adding a component utilizing an OLED screen from the ESP-IDF Component Registry. However, this time around, a third-party component will be used. A popular maker from Japan nicknamed [nopnop2002](https://github.com/nopnop2002" \t "_blank)creates ESP-IDF components. We’ll use his [esp-idf-ssd1306 component](https://github.com/nopnop2002/esp-idf-ssd1306), which uses a menu configuration to configure OLED parameters. This provides a better visual interaction with the user as he or she adjusts the parameters.

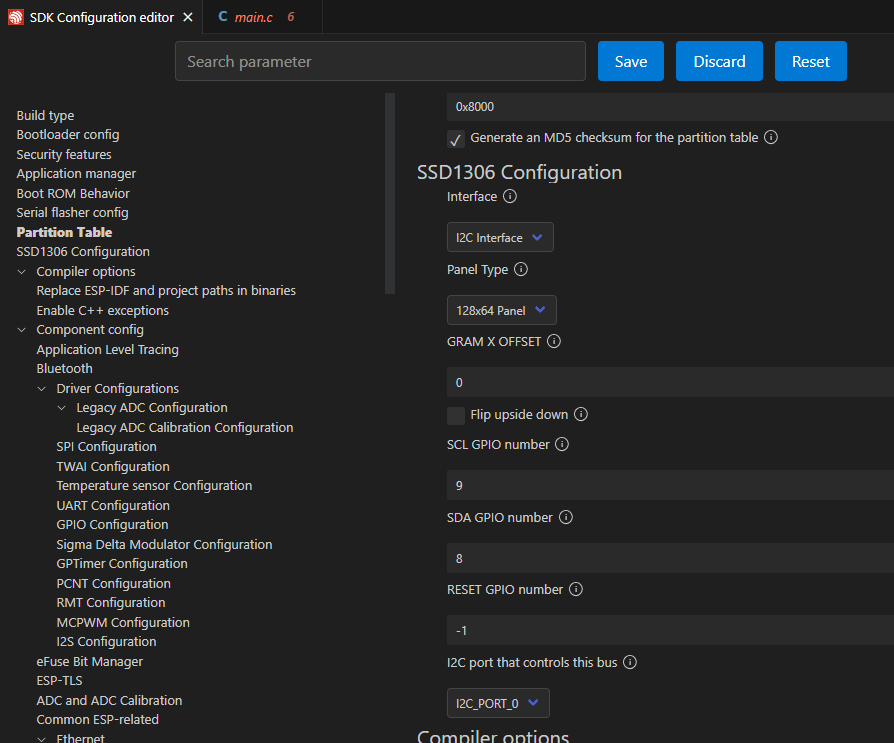


**DEMO RUN OF NOPNOP2002'S SSD1306 COMPONENT**

Download [nopnop2020/esp-idf-ssd1306](https://github.com/nopnop2002/esp-idf-ssd1306" \t "_blank) and then open the TextDemo folder in Visual Studio Code through *File -> Open Folder…* Having read our previous blog about [running programs in ESP-IDF in VS Code](https://www.phippselectronics.com/run-your-first-esp32-program-in-visual-studio-code/" \t "_blank),  run menu configuration and ensure you have pin 8 as SDA and pin 9 as SCL (or use what you have). You can put -1 on RESET if your OLED does not have a reset pin. Choose a Panel type of 128×64 OLED screen if you have that kind of panel, otherwise, choose your own.



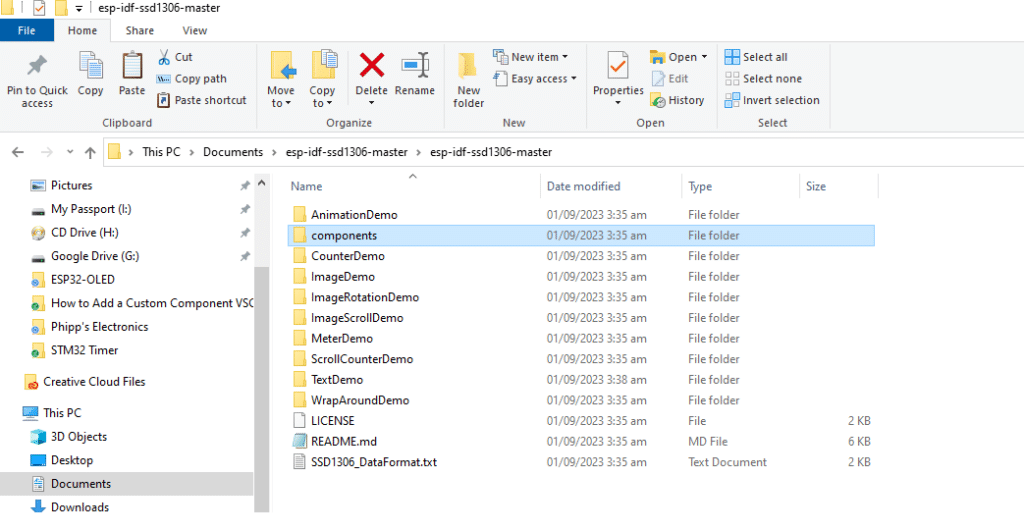


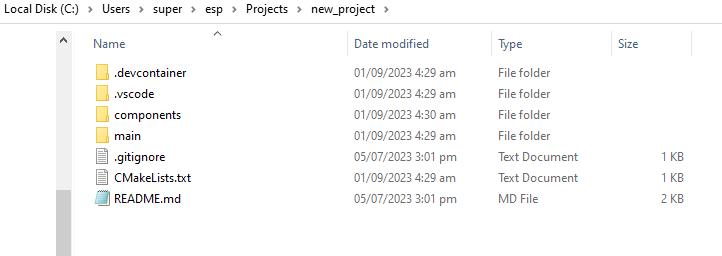


[Build and Run](https://www.phippselectronics.com/run-your-first-esp32-program-in-visual-studio-code/" \l "build-flash-monitor) it on your ESP32 board to see how the OLED Text demo runs.

**HOW TO INCORPORATE NOPNOP'S SSD1306 COMPONENT INTO YOUR PROJECTS**

Adding nopnop’s SSD1306 component to your projects is easy. Copy the component folder from nopnop’s SSD1306 directory to your project’s directory. It’s as simple as that. You can test it out by creating a new project and then adding the component folder to your new project’s folder.





On your new project’s main.c file, don’t forget to include the ssd1306.h header file.

#include "ssd1306.h"

COPY

Go ahead and paste the code below to your main file. Configure, Build, and Run the code and see the output.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "esp\_log.h"

#include "ssd1306.h"

#include "font8x8\_basic.h"

/\*

You have to set this config value with menuconfig

CONFIG\_INTERFACE

for i2c

CONFIG\_MODEL

CONFIG\_SDA\_GPIO

CONFIG\_SCL\_GPIO

CONFIG\_RESET\_GPIO

for SPI

CONFIG\_CS\_GPIO

CONFIG\_DC\_GPIO

CONFIG\_RESET\_GPIO

\*/

#define tag "SSD1306"

void app\_main(void)

{

SSD1306\_t dev;

int center, top, bottom;

char lineChar[20];

#if CONFIG\_I2C\_INTERFACE

ESP\_LOGI(tag, "INTERFACE is i2c");

ESP\_LOGI(tag, "CONFIG\_SDA\_GPIO=%d",CONFIG\_SDA\_GPIO);

ESP\_LOGI(tag, "CONFIG\_SCL\_GPIO=%d",CONFIG\_SCL\_GPIO);

ESP\_LOGI(tag, "CONFIG\_RESET\_GPIO=%d",CONFIG\_RESET\_GPIO);

i2c\_master\_init(&dev, CONFIG\_SDA\_GPIO, CONFIG\_SCL\_GPIO, CONFIG\_RESET\_GPIO);

#endif // CONFIG\_I2C\_INTERFACE

#if CONFIG\_SPI\_INTERFACE

ESP\_LOGI(tag, "INTERFACE is SPI");

ESP\_LOGI(tag, "CONFIG\_MOSI\_GPIO=%d",CONFIG\_MOSI\_GPIO);

ESP\_LOGI(tag, "CONFIG\_SCLK\_GPIO=%d",CONFIG\_SCLK\_GPIO);

ESP\_LOGI(tag, "CONFIG\_CS\_GPIO=%d",CONFIG\_CS\_GPIO);

ESP\_LOGI(tag, "CONFIG\_DC\_GPIO=%d",CONFIG\_DC\_GPIO);

ESP\_LOGI(tag, "CONFIG\_RESET\_GPIO=%d",CONFIG\_RESET\_GPIO);

spi\_master\_init(&dev, CONFIG\_MOSI\_GPIO, CONFIG\_SCLK\_GPIO, CONFIG\_CS\_GPIO, CONFIG\_DC\_GPIO, CONFIG\_RESET\_GPIO);

#endif // CONFIG\_SPI\_INTERFACE

#if CONFIG\_FLIP

dev.\_flip = true;

ESP\_LOGW(tag, "Flip upside down");

#endif

#if CONFIG\_SSD1306\_128x64

ESP\_LOGI(tag, "Panel is 128x64");

ssd1306\_init(&dev, 128, 64);

#endif // CONFIG\_SSD1306\_128x64

#if CONFIG\_SSD1306\_128x32

ESP\_LOGI(tag, "Panel is 128x32");

ssd1306\_init(&dev, 128, 32);

#endif // CONFIG\_SSD1306\_128x32

#if CONFIG\_SSD1306\_128x64

top = 2;

center = 3;

bottom = 8;

#endif // CONFIG\_SSD1306\_128x64

#if CONFIG\_SSD1306\_128x32

top = 1;

center = 1;

bottom = 4;

#endif // CONFIG\_SSD1306\_128x32

// main code

while(1)

{

vTaskDelay(3000 / portTICK\_PERIOD\_MS);

// Hello World

ssd1306\_display\_text(&dev, 0, "Hello World", 12, false);

vTaskDelay(3000 / portTICK\_PERIOD\_MS);

ssd1306\_clear\_screen(&dev, false);

}

}

COPY