ESP32-S3 DevKitC Pinout Refence Guide: GPIOs Explained

The ESP32-S3 DevKitC is one of the most popular development boards with the ESP32-S3 chip. This chip comes with 45 pins and has a different pinout than the boards with the ESP-WROOM-32 chip. In this article, we'll take a look at the ESP32-S3 DevkitC board Pinout and explain the different functions of its GPIOs.

ESP32-S3 Peripherals

The ESP32-S3 peripherals include:

- 20 Analog-to-Digital Converter (ADC) channels
- 4 SPI interfaces
- 3 UART interfaces

- 2 I2S interfaces
- 14 Capacitive sensing GPIOs

Due to the ESP32 chip multiplexing feature, you can assign almost any function to the GPIO of your choice. However, there are pins with assigned functions by default. We'll take a look at that in this pinout.

ESP32-S3 DevKitC-1 Pinout

The following picture shows the pinout of the ESP32-S3 DevKitC-1 board, one of the most popular development boards with the S3 chip. There are different versions of the same board with slightly different pinouts. Always double-check the pin location and the GPIO label before connecting any peripherals.

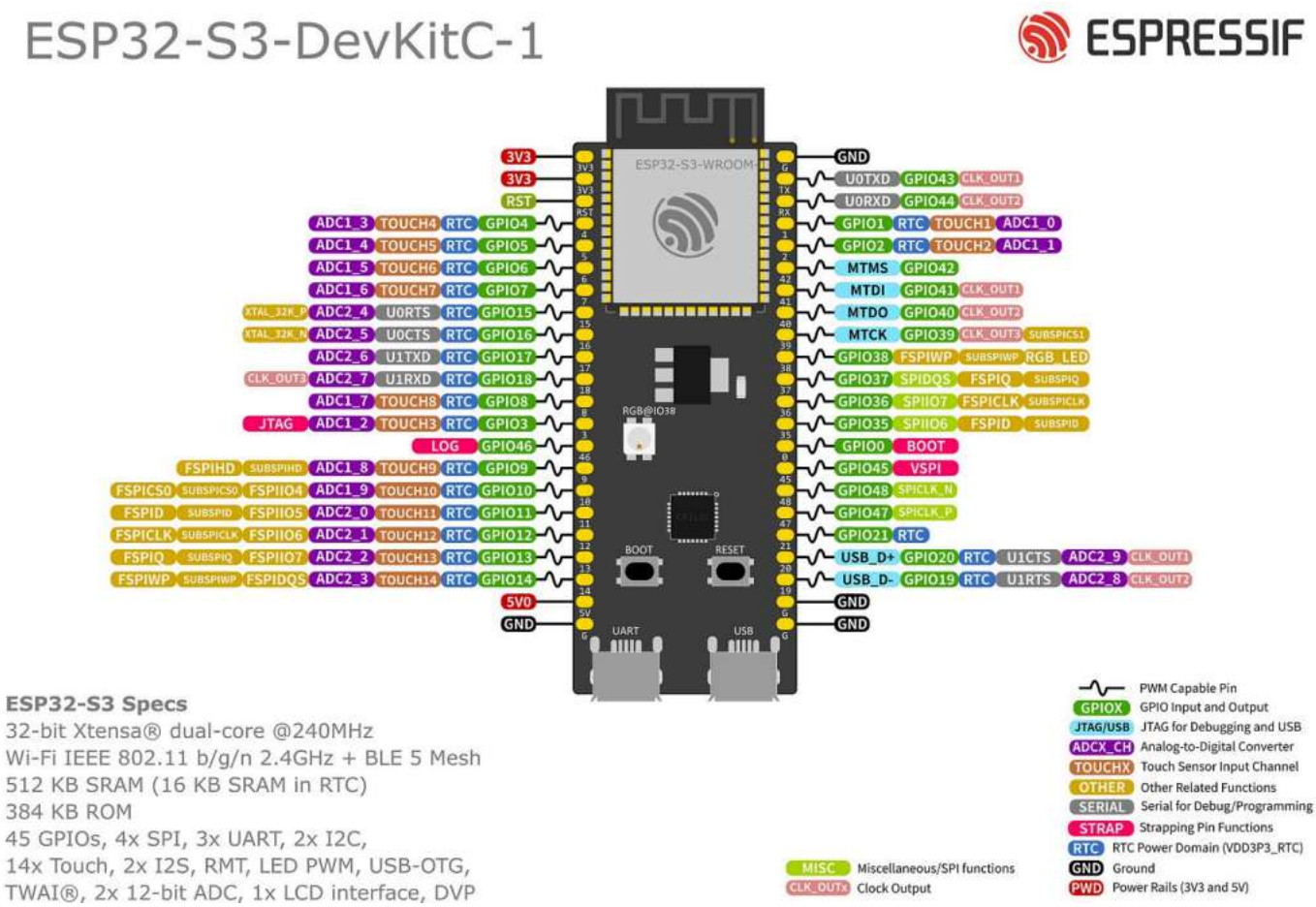


Image source

For more information about this board, check the official documentation here.

GPIOs 26 to 32 are connected to the integrated SPI flash and PSRAM and are not recommended for other uses. They are not exposed in this particular board, but if they are exposed on your board, avoid using them:

- GPIO 26 (Flash/PSRAM SPICS1)
- GPIO 27 (Flash/PSRAM SPIHD)
- GPIO 28 (Flash/PSRAM SPIWP)
- GPIO 29 (Flash/PSRAM SPICS0)
- GPIO 30 (Flash/PSRAM SPICLK)
- GPIO 31 (Flash/PSRAM SPIQ)
- GPIO 32 (Flash/PSRAM SPID)

Capacitive touch GPIOs

The ESP32-S3 has 14 internal capacitive touch GPIOs. These can sense variations in anything that holds an electrical charge, like the human skin. So they can detect variations induced when touching the GPIOs with a finger. These pins can be easily integrated into capacitive pads and replace mechanical buttons. The capacitive touch pins can also be used to wake up the ESP32 from deep sleep.

Those internal touch sensors are connected to these GPIOs:

- T1 (GPIO 1)
- T2 (GPIO 2)
- T3 (GPIO 3)
- T4 (GPIO 4)
- T5 (GPIO 5)
- T6 (GPIO 6)
- T7 (GPIO 7)
- T8 (GPIO 8)
- T9 (GPIO 9)
- T10 (GPIO 10)
- T11 (GPIO 11)
- T12 (GPIO 12)
- T13 (GPIO 13)

Analog to Digital Converter (ADC)

The ESP32 has 20x 12-bit ADC input channels. These are the GPIOs that can be used as ADC and respective channels:

- ADC1_CH0 (GPIO 1)
- ADC1_CH1 (GPIO 2)
- ADC1_CH2 (GPIO 3)
- ADC1_CH3 (GPIO 4)
- ADC1_CH4 (GPIO 5)
- ADC1_CH5 (GPIO 6)
- ADC1_CH6 (GPIO 7)
- ADC1_CH7 (GPIO 8)
- ADC1_CH8 (GPIO 9)
- ADC1_CH9 (GPIO 10)
- ADC2_CH0 (GPIO 11)
- ADC2_CH1 (GPIO 12)
- ADC2_CH2 (GPIO 13)
- ADC2_CH3 (GPIO 14)
- ADC2_CH4 (GPIO 15)
- ADC2_CH5 (GPIO 16)
- ADC2_CH6 (GPIO 17)
- ADC2_CH7 (GPIO 18)
- ADC2_CH8 (GPIO 19)
- ADC2_CH9 (GPIO 20)

The ADC input channels have a 12-bit resolution. This means that you can get analog readings ranging from 0 to 4095, in which 0 corresponds to 0 V and 4095 to 3.3 V. You can also set the resolution of your channels on the code and the ADC range.

RTC GPIOs

There is RTC GPIO support on the ESP32-S3. The GPIOs routed to the RTC low-power subsystem can be used when the ESP32 is in deep sleep. These RTC GPIOs

(ULP) coprocessor is running. The following GPIOs can be used as an external wake-up source.

- RTC_GPIO0 (GPIO0)
- RTC_GPIO1 (GPIO1)
- RTC_GPIO2 (GPIO2)
- RTC_GPIO3 (GPIO3)
- RTC_GPIO4 (GPIO4)
- RTC_GPIO5 (GPIO5)
- RTC_GPIO6 (GPIO6)
- RTC_GPIO7 (GPIO7)
- RTC_GPIO8 (GPIO8)
- RTC_GPIO9 (GPIO9)
- RTC_GPIO10 (GPIO10)
- RTC_GPIO11 (GPIO11)
- RTC_GPIO12 (GPIO12)
- RTC_GPIO13 (GPIO13)
- RTC_GPIO14 (GPIO14)
- RTC_GPIO15 (GPIO15)
- RTC_GPIO16 (GPIO16)
- RTC_GPIO17 (GPIO17)
- RTC_GPIO18 (GPIO18)
- RTC_GPIO19 (GPIO19)
- RTC_GPIO20 (GPIO20)
- RTC_GPIO21 (GPIO21)

PWM

The ESP32-S3 has an LED PWM controller with 8 PWM channels that can be configured to generate PWM signals with different properties. All pins that can act as outputs can be used as PWM pins.

To set a PWM signal, you need to define these parameters in the code:

- Signal’s frequency;

- GPIO where you want to output the signal.

I2C

When using the ESP32-S3 with the Arduino IDE, these are the ESP32 I2C default pins:

- GPIO 8 (SDA)
- GPIO 9 (SCL)

Learn more about I2C with our tutorials:

- ESP32 I2C Communication: Set Pins, Multiple Bus Interfaces and Peripherals (Arduino IDE)
- ESP32: I2C Scanner (Arduino IDE) – Finding the Address of I2C Devices
- ESP32 I2C Master and Slave (I2C Communication Between Two ESP32) – Arduino IDE

SPI

The ESP32 integrates 4 SPI peripherals: SPI0, SPI1, SPI2 (commonly referred to as HSPI), and SPI3 (commonly referred to as VSPI).

SP0 and SP1 are used internally to communicate with the built-in flash memory, and you should not use them for other tasks.

You can use **HSPI** and **VSPI** to communicate with other devices. HSPI and VSPI have independent bus signals.

SPI	MOSI	MISO	CLK	CS
HSPI (SPI 2)	GPIO 11	GPIO 13	GPIO 12	GPIO 10
VSPI (SPI 3)	GPIO 35	GPIO 37	GPIO 36	GPIO 39

Learn more about SPI communication protocol: ESP32 SPI Communication: Set Pins, Multiple SPI Bus Interfaces, and Peripherals (Arduino IDE).

Interrupts

All GPIOs can be configured as interrupts.

UART Pins – Serial Communication

The ESP32-S3 supports multiple UART (Universal Asynchronous Receiver-Transmitter) interfaces that allow serial communication with various devices. The ESP32 supports up to three UART interfaces: **UART0**, **UART1**, and **UART2**, depending on the ESP32 board model you're using.

Like I2C and SPI, these UARTs can be mapped to any GPIO pin, although they have default pin assignments on most board models.

The following table shows the default UART0, UART1, and UART2 RX and TX pins for the ESP32-S3:

UART Port	TX	RX	Remarks
UART0	GPIO 43	GPIO 44	Cannot be changed
UART1	GPIO 17	GPIO 18	Can be assigned to other GPIOs
UART2	—	—	Assign any pins of your choice

Strapping Pins

The ESP32-S3 chip has the following strapping pins:

- GPIO 3
- GPIO 45
- GPIO 46

These pins are used to put the ESP32 into bootloader or flashing mode. On most development boards with built-in USB/Serial, you don't need to worry about the state of these pins. The board puts the pins in the right state for flashing or boot mode. However, you should avoid using these pins on your projects.

Enable (EN/RST)

The Enable (EN/RST) is the 3.3V regulator's enable pin. It's pulled up, so connect to GND to disable the 3.3V regulator. This means that you can use this pin connected to a pushbutton to restart your ESP32, for example.

Wrapping Up

We hope you find this reference guide for the ESP32-S3 GPIOs useful. If you have more tips about the ESP32-S3 GPIOs, please share by writing a comment below.

More ESP32 content:

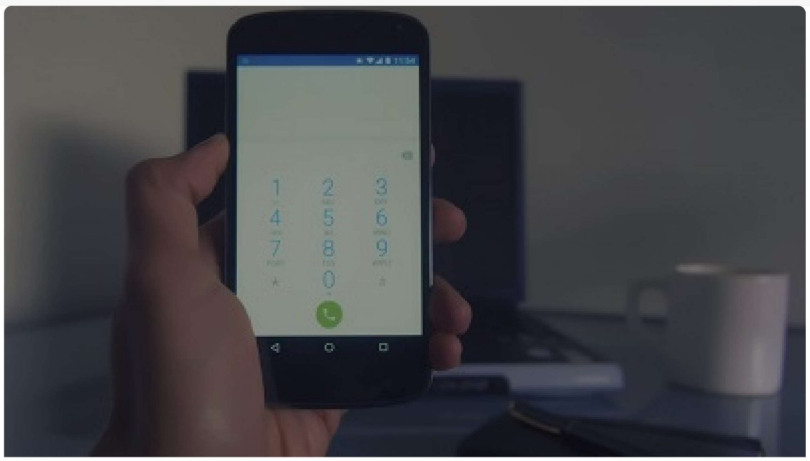
- **Learn ESP32 with Arduino IDE eBook**
- Free ESP32 Projects and Tutorials

Thanks for reading.

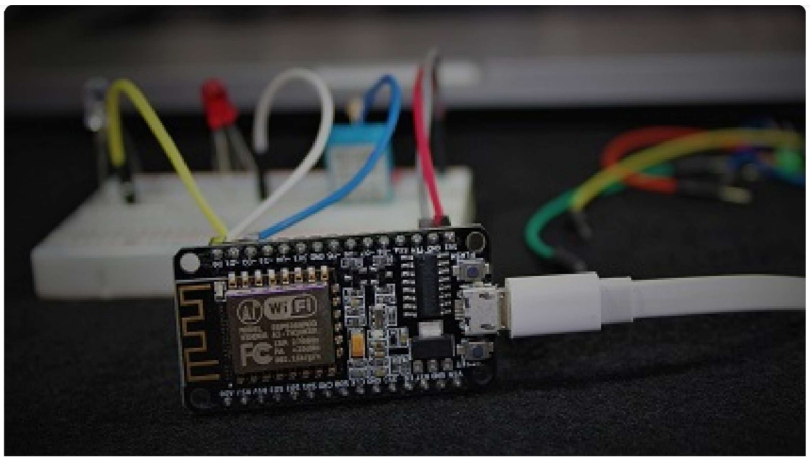
SMART HOME with Raspberry Pi, ESP32, ESP8266 [eBook]

Learn how to build a home automation system and we'll cover the following main subjects: Node-RED, Node-RED Dashboard, Raspberry Pi, ESP32, ESP8266, MQTT, and InfluxDB database **DOWNLOAD »**

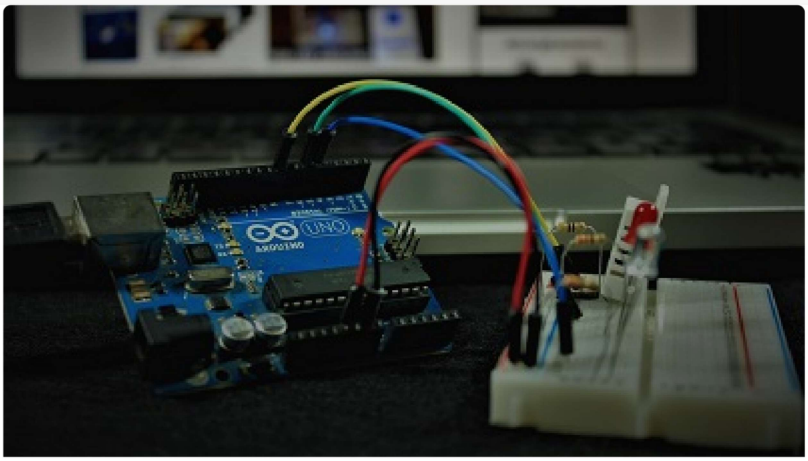
Recommended Resources



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED



Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.



Arduino Step-by-Step Projects » Build 25 Arduino projects with our course, even with no prior experience!

What to Read Next…

MicroPython: DS18B20 Temperature Sensor with ESP32 and ESP8266

ESP32 Bluetooth Classic with Arduino IDE – Getting Started

ESP8266 Voltage Regulator (LiPo and Li-ion Batteries)

Enjoyed this project? Stay updated by subscribing our newsletter!

Your Email Address

SUBSCRIBE

19 thoughts on “ESP32-S3 DevKitC Pinout Refence Guide: GPIOs Explained”

Renzo

October 2, 2024 at 5:29 pm

If pin GPIO 8 (SDA) , GPIO 9 (SCL) are occupied it is possible to use some

Thanks
Renzo

Reply

Joe
October 2, 2024 at 7:31 pm

Hi Sara,
Thank you very much for this short tutorial.
I have the exact questions Boris request.
Using the 16MB Flash would really be great.

For Renzo,
Try some variant of the following code to redefine I2C pins for any ESP32.

The following code can be used to redefine I2C & reads an analog signal using an I2C device, The analog signal on pin A0 of the ADS1115 is from a light sensor that detects a laser pulse at high resolution.
The ADS1115 is an I2C device with 16 bit resolution and a multiplexed 4 channel Analog to Digital Converter.
Adafruit has some excellent info on how to use and on programmable gain for the ADS1115.
Note that the 20 ADC channels on ESP32 are 12 bit.
Also the following code does not include all the initialization for the variables, or all the calls, or all the functions.
It is hoped that it will help illustrate the concepts for redefining I2C pins, etc.)

```
// Include the I2C library
#include <Wire.h>

// Include a library for the I2C device, for example an I2C using a 16bit ADC
```

```
// Define I2C pins
#define SDA_PIN 21 // Replace with your desired SDA pin
#define SCL_PIN 22 // Replace with your desired SCL pin

Adafruit_ADS1115 ads; // Create an instance of the ADS1115 ADC
const int analogInputPin = 0; // Initialize the A0 pin on ADS1115 device

// Initialize global variables
const int numSamplesForBaseline = 100;
const int numSamplesPerPulse = 10;
unsigned long previousMicros = 0;
unsigned long sampleIntervalMicros = 500;
int baselineThreshold = 0;
int analogSum = 0;
int pulseCount = 0;
int averageAmplitude = 0;

void setup() {
Serial.begin(115200);
Wire.begin(SDA_PIN, SCL_PIN); // Initialize I2C w specified SDA & SCL
pins
ads.begin();

// Calculate baseline threshold during initialization
calculateBaselineThreshold();

// Insert other code for setup

}

void loop() {
// Read and process analog signal
processAnalogSignal(currentMicros);

// Main loop code Example: Print results every 10 seconds
```

```
printResultsToSerial();
sendResultsToThingSpeak();
previousMicros = currentMicros;
}

}

void processAnalogSignal(unsigned long currentMicros) {
int rawAnalogValue = ads.readADC_SingleEnded(analogInputPin);
baselineThreshold = (baselineThreshold + rawAnalogValue) / 2;

if (rawAnalogValue > baselineThreshold) {
if (currentMicros – previousMicros >= sampleIntervalMicros) {
for (int i = 0; i < numSamplesPerPulse; i++) {
rawAnalogValue = ads.readADC_SingleEnded(analogInputPin);
analogSum += rawAnalogValue;
}

averageAmplitude = analogSum / numSamplesPerPulse;
analogSum = 0;
pulseCount++;
previousMicros = currentMicros;
}

}
}

void printResultsToSerial() {
Serial.println("Average Amplitude: " + String(averageAmplitude));
Serial.println("Pulse Count: " + String(pulseCount));
}
```

Best regards,
Joe

Joe

October 3, 2024 at 5:15 am

Hi Renzo,
Random Nerds has a great I2C tutorial at
https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/
It shows how to redefine ESP32 pins for I2C.
Best regards,
Joe

Reply

Boris

October 2, 2024 at 5:46 pm

Hello guys!!

I got the exact same board. I would really appreciate a tutorial on how to use the full 16mb flash memory.
I tried a lot of things but I never could use more than 3.5mb something no matter what settings I selected in the Arduino IDE.

Have a great day!!

Boris

Frank De Hert
October 3, 2024 at 7:37 am

I have the same problem. Tried about everything I could find on the Web to no avail. Would really like a tutorial on that.

Thanks in advance.
Frank

Reply

Marius
October 2, 2024 at 7:28 pm

Should MOSI of HSPI (SPI 2) not be GPIO11 instead of GPIO35?

Reply

Sara Santos
October 7, 2024 at 2:02 pm

Yes. You're right.
It's fixed now.
Regards,
S

Reply

Hiren

October 3, 2024 at 4:08 am

Why don't you guys make projects using ESP-IDF? Your step by step guide will be really helpful for others.

Reply

tDaniel

October 3, 2024 at 11:10 am

Many questions come up to mind, I guess most of your projects should be reviewed using this board, especially the ones about memery management and usage of sd cards. Thank you for this first coverage of this board.

Reply

tDaniel

October 3, 2024 at 8:37 pm

Does anyone has an idea about how to achieve debugging with this board

Reply

steve

October 4, 2024 at 4:36 pm

I have been using this exact DevKit, a true Espressif version, not one of the countless variations, for over a year and have discovered via much frustration that some of the pin-outs you list are not correct or should be avoided, at least with MicroPython firmware. You should not/cannot use TX0/RX0, HSPI is GPIO9-11, GPIO48 is the RGB led, and GPIO35-37 can be used on the N8 variant (no PSRAM) but not on the N8R8 variant (with PSRAM).

Reply

Sara Santos

October 7, 2024 at 8:21 am

Hi.
Thanks for your feedback.
I'll that a look into that.
Regards,
Sara

Reply

David

I2C: When using the ESP32-S3 with the Arduino IDE, these are the ESP32 I2C default pins: GPIO 8 (SDA) & GPIO 9 (SCL)

THANK YOU! I have been trying to find the default GPIO pin numbers for BOTH of the ESP32-S3 default built-in I2C ports (four pins total) for days on end. And here you have two of the four, and I tested them both with an I2C scanner utility, and they work!

Is there any way you can provide the GPIO pin numbers for the second default I2C port on the ESP32-S3-DevKitC-1 board? I do not have the ESP-IDF installed so I cannot snoop around in source and header files. And of-course the ESP documentation does not list the GPIO numbers, at least I can't find them.

Thank You, David in Florida

Reply

David

December 8, 2024 at 6:30 am

Here is how I scanned one default I2C port on the ESP32-S3-DevKitC-1 board to verify the I2C port works with Rui's GPIO numbers:

```
/**********************************
ESP32-S3 I2C Scanner v1.01.txt
ESP32-S3 I2C Scanner
OLED ESP32-S3
+———-+ +———-+
| GND|—-|GND |
```

```
| SCL|—-|GPIO9 |
| SDA|—-|GPIO8 |
+————-+ +————-+

Posted by Rui Santos: https://randomnerdtutorials.com/esp32-s3-devkitc-pinout-guide/
I2C: When using the ESP32-S3 with the Arduino IDE, these are the ESP32 I2C default pins:

GPIO 8 (SDA)

GPIO 9 (SCL)

Select these two pins like this in your sketch:
//Wire.begin(I2C_SDA, I2C_SCL);
Wire.begin(8, 9);

Serial Monitor says (this is a 0.96″ I2C OLED SSD1306 display):
Scanning…
I2C device found at address 0x3C

Hardware:
ESP32-S3-DevKitC-1: https://www.amazon.com/dp/B0D9W4Y3F3
0.96″ White SSD1306 I2C OLED Display:
https://www.amazon.com/dp/B09T6SJBV5

Problem: The onboard addressable RGB LED on GPIO38 is always
on bright white. I cannot find a way to simply disable it in software.

**********/

#include <Wire.h>

#define rgbledPin 38
```

```
Wire.begin(8, 9); //For ESP32-S3-DevKitC-1 board
//Wire.begin(5, 4);
//Wire.begin(33, 32);
pinMode(rgbledPin, OUTPUT);
//pinMode(rgbledPin, INPUT);
Serial.begin(115200);
Serial.println("\nI2C Scanner Started…");
}

void loop() {
byte error, address;
int nDevices;
Serial.println("Scanning…");
digitalWrite(rgbledPin, LOW);
//digitalWrite(rgbledPin, HIGH);
nDevices = 0;
for(address = 1; address < 127; address++ ) {
Wire.beginTransmission(address);
error = Wire.endTransmission();
if (error == 0) {
Serial.print("I2C device found at address 0x");
if (address<16) {
Serial.print("0");
}
Serial.println(address,HEX);
nDevices++;
}
else if (error==4) {
Serial.print("Unknow error at address 0x");
if (address<16) {
Serial.print("0");
}
Serial.println(address,HEX);
}
```

```
Serial.println("No I2C devices found\n");
}
else {
Serial.println("done\n");
}
//digitalWrite(ledPin, LOW);
delay(2000);
}
```

Reply

steve

December 8, 2024 at 3:38 pm

You can actually use any (more or less) pins for I2C with the ESP32-S3. Micropython allows you to specify them. I don't know if the Arduino platform also does. I strongly recommend you wean yourself from Arduino and use Micropython instead. One of the smartest moves I ever made.

Reply

p43lz3r

December 8, 2024 at 7:10 pm

Did this research so time ago as well but it seems there are no default pins for the second I2C interface.

```
int sda_pin = 6; // GPIO6 as I2C SDA
int scl_pin = 7; // GPIO7 as I2C SCL

void setup()
{
Wire.setPins(sda_pin, scl_pin); // Set the I2C pins before begin
Wire.begin(); // join i2c bus (address optional for master)
}
```

It’s important to call “Wire.setPins()” before “Wire.begin()”.

docs.espressif.com/projects/arduino-esp32/en/latest/tutorials/io_mux.html

Reply

Sara Santos

December 8, 2024 at 7:41 pm

Thanks for sharing this info.
Regards,
Sara

Reply

Matt

December 25, 2024 at 5:49 pm

Hello all,

couldn’t get any monitoring info .
Would you be able to assist me how to run smoothly esp32s3 on raspberry Pi ?
Thanks in advance.

Reply

Boris
December 28, 2024 at 1:03 pm

Sounds interesting… What do you want to achieve in detail? Control the ESP32 with the Raspberry Pi? Via SPI, I2C or serial?

Reply

Leave a Comment

Name *

Email *

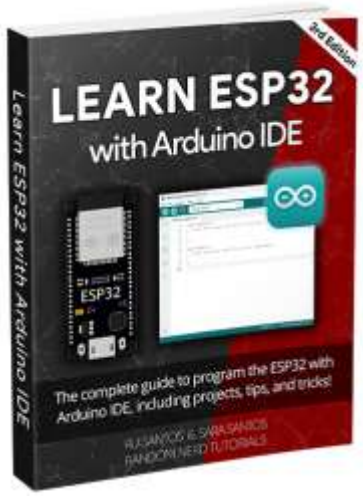
Website

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Post Comment

Affiliate Disclosure: *Random Nerd Tutorials is a participant in affiliate advertising programs designed to provide a means for us to earn fees by linking to Amazon, eBay, AliExpress, and other sites. We might be compensated for referring traffic and business to these companies.*



Learn ESP32 with Arduino IDE eBook »
Complete guide to program the ESP32 with Arduino IDE!

SMART HOME with Raspberry Pi, ESP32, and ESP8266 » learn how to build a complete home automation system.

Learn Raspberry Pi Pico/Pico W with MicroPython » The complete getting started guide to get the most out of the the Raspberry Pi Pico/Pico W (RP2040) microcontroller board using MicroPython programming language.

🔥 **Learn LVGL: Build GUIs for ESP32 Projects »** Learn how to build Graphical User Interfaces (GUIs) for ESP32 Projects using LVGL (Light Versatile Graphics Library) with the Arduino IDE.

About Support Terms and Conditions Privacy Policy Refunds Complaints' Book

MakerAdvisor.com Join the Lab

Copyright © 2013-2025 · RandomNerdTutorials.com · All Rights Reserved