ESP32 SPI Communication: Set Pins, Multiple SPI Bus Interfaces, and Peripherals (Arduino IDE)

This is a simple guide about SPI communication protocol with the ESP32 using Arduino IDE. We'll take a look at the ESP32 SPI pins, how to connect SPI devices, define custom SPI pins, how to use multiple SPI devices, and much more.

Table of Contents:

This tutorial focus on programming the ESP32 using the Arduino core, so before proceeding, you should have the ESP32 add-on installed in your Arduino IDE. Follow the next tutorial to install the ESP32 on the Arduino IDE, if you haven't already.

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

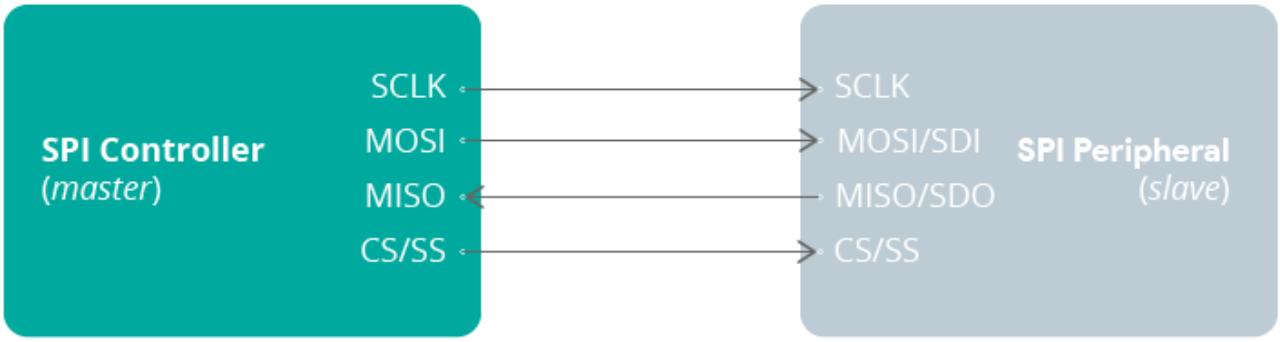
Alternatively, you can also use VS Code with the PlatformIO extension to program your boards using the Arduino core:

- Getting Started with VS Code and PlatformIO IDE for ESP32 and ESP8266 (Windows, Mac OS X, Linux Ubuntu)

Introducing ESP32 SPI Communication Protocol

SPI stands for **S**erial **P**eripheral **I**nterface, and it is a synchronous serial data protocol used by microcontrollers to communicate with one or more peripherals. For example, your ESP32 board communicating with a sensor that supports SPI or with another microcontroller.

In an SPI communication, there is always a **controller** (also called *master*) that controls the **peripheral** devices (also called *slaves*). Data can be sent and received simultaneously. This means that the master can send data to a slave, and a slave can send data to the master at the same time.



You can have *only one master*, which will be a microcontroller (the ESP32), but you can have multiple slaves. A slave can be a sensor, a display, a microSD card, etc., or

sensors, but the same sensor can't be connected to multiple ESP32 boards simultaneously.

SPI Interface

For SPI communication you need four lines:

- **MISO**: Master In Slave Out
- **MOSI**: Master Out Slave In
- **SCK**: Serial Clock
- **CS** /**SS**: Chip Select (used to select the device when multiple peripherals are used on the same SPI bus)

On a slave-only device, like sensors, displays, and others, you may find a different terminology:

- **MISO** may be labeled as **SDO** (Serial Data Out)
- **MOSI** may be labeled as **SDI** (Serial Data In)

ESP32 SPI Peripherals

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, and each bus can drive up to three SPI slaves.

ESP32 Default SPI Pins

Many ESP32 boards come with default SPI pins pre-assigned. The pin mapping for most boards is as follows:

VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

Warning: depending on the board you're using, the default SPI pins might be different. So, make sure you check the pinout for the board you're using. Additionally, some boards don't have pre-assigned SPI pins, so you need to set them on code.

Note: usually, when not specified, the board will use the VSPI pins when initializing an SPI communication with the default settings.

Whether your board comes with pre-assigned pins or not, you can always set them on code.

Finding your ESP32 Board's Default SPI Pins

If you're not sure about your board's default SPI pins, you can upload the following code to find out.

```
  Complete project details at https://RandomNerdTutorials.com/espi

  Permission is hereby granted, free of charge, to any person obta
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be i
  copies or substantial portions of the Software.
*/

//Find the default SPI pins for your board
//Make sure you have the right board selected in Tools > Boards
void setup() {

  // put your setup code here, to run once:
```

```
  Serial.println(MOSI);
  Serial.print("MISO: ");
  Serial.println(MISO);
  Serial.print("SCK: ");
  Serial.println(SCK);
  Serial.print("SS: ");
  Serial.println(SS);
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

View raw code

Important: make sure you select the board you're using in **Tools** > **Board**, otherwise, you may not get the right pins.

After uploading the code, open the Serial Monitor, RST your board and you'll see the SPI pins.

Using Custom ESP32 SPI Pins

When using libraries to interface with your SPI peripherals, it's usually simple to use custom SPI pins because you can pass them as arguments to the library constructor.

For example, take a quick look at the following example that interfaces with a BME280 sensor using the `Adafruit_BME280` library.

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp
  Based on the Adafruit_BME280_Library example: https://github.co

  Permission is hereby granted, free of charge, to any person obt
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be
  copies or substantial portions of the Software.
*/

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#include <SPI.h>
#define BME_SCK 25
#define BME_MISO 32
#define BME_MOSI 26
#define BME_CS 33
#define SEALEVELPRESSURE_HPA (1013.25)

//Adafruit_BME280 bme; // I2C
//Adafruit_BME280 bme(BME_CS); // hardware SPI
Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // softw
```

View raw code

You can easily pass your custom SPI pins to the library constructor.

```
Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);
```

In that case, I was using the following SPI pins (not default) and everything worked as expected:

```
#define BME_SCK 25
#define BME_MISO 32
#define BME_MOSI 26
#define BME_CS 33
```

If you're not using a library, or the library you're using doesn't accept the pins in the library constructor, you may need to initialize the SPI bus yourself. In that case, you would need to call the `SPI.begin()` method on the `setup()` and pass the SPI pins as arguments:

```
SPI.begin(SCK, MISO, MOSI, SS);
```

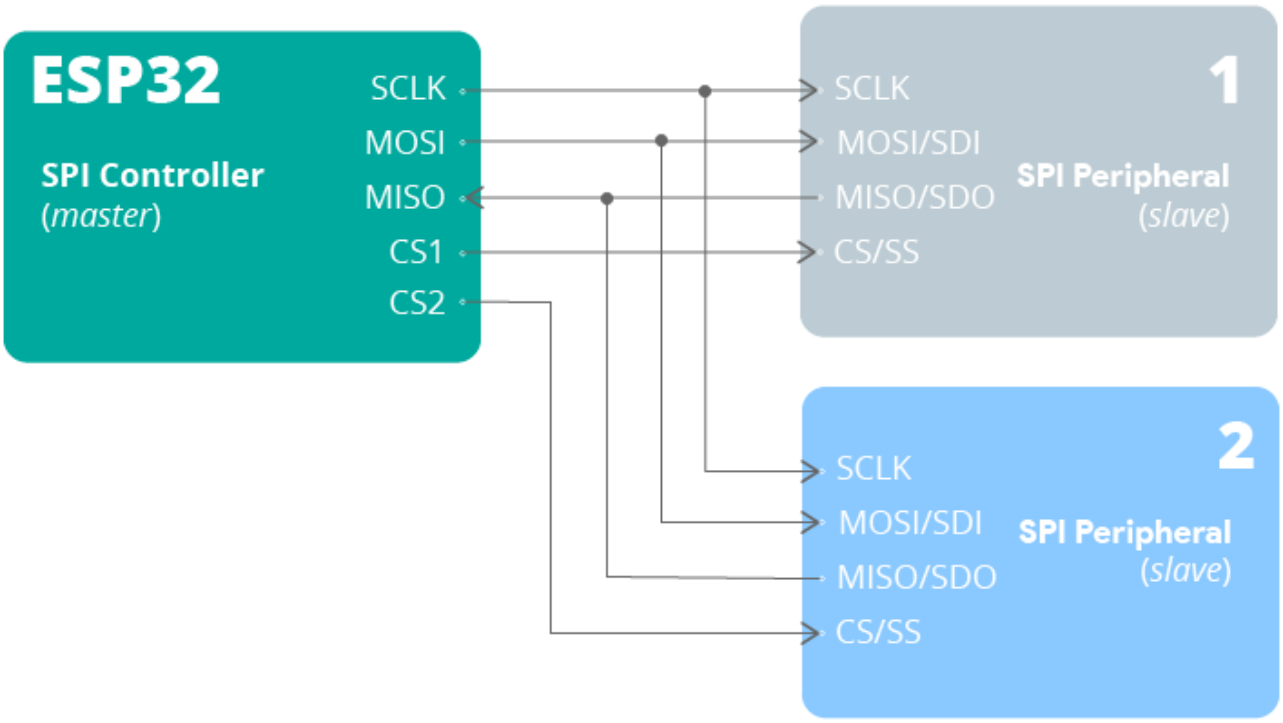
You can see an example of this scenario in this tutorial, in which we initialize an SPI LoRa transceiver that is connected to custom SPI pins. Or this example showing how to use custom SPI pins with a microSD card module.

ESP32 with Multiple SPI Devices

As we've seen previously, you can use two different SPI buses on the ESP32 and each bus can connect up to three different peripherals. This means that we can connect up to six SPI devices to the ESP32. If you need to use more, you can use an SPI multiplexer

Multiple SPI Devices (same bus, different CS pin)

To connect multiple SPI devices, you can use the same SPI bus as long as each peripheral uses a different CS pin.



To select the peripheral you want to communicate with, you should set its `CS` pin to `LOW`. For example, imagine you have peripheral 1 and peripheral 2. To read from peripheral 1, make sure its `CS` pin is set to `LOW` (here represented as `CS_1`):

```
digitalWrite(CS_1, LOW); // enable CS pin to read from peripheral 1

/*
 use any SPI functions to communicate with peripheral 1
*/
```

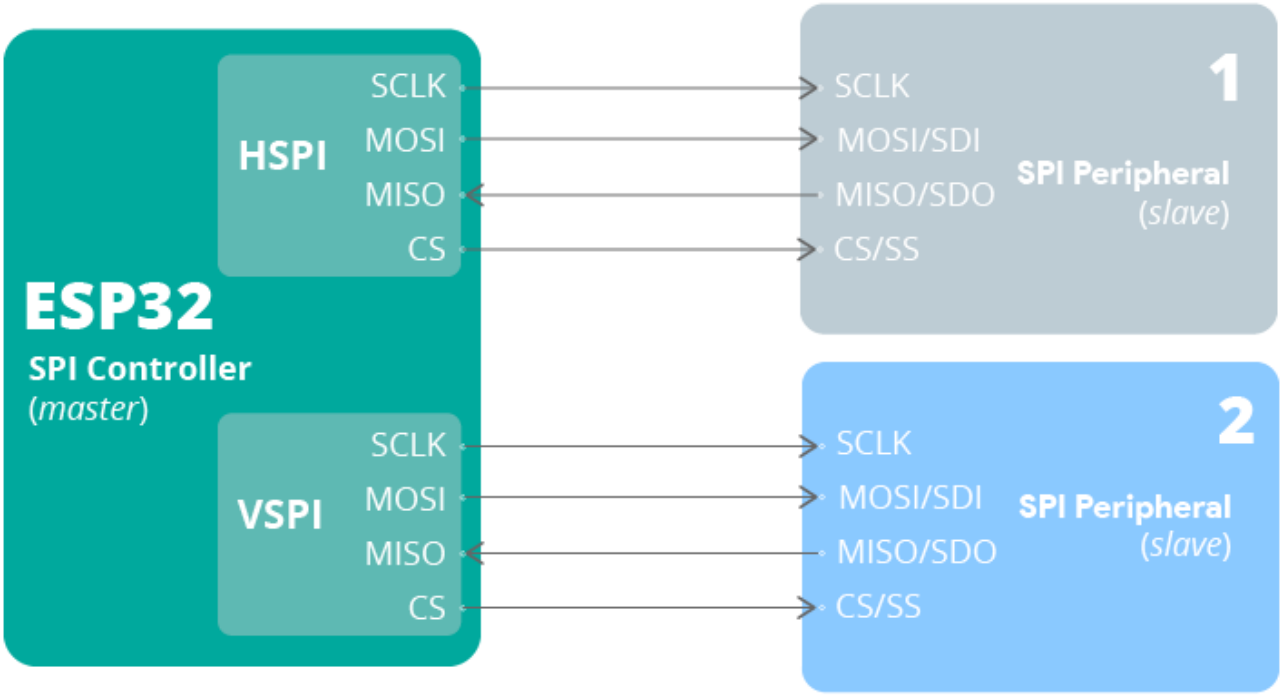
Then, at same point, you'll want to read from peripheral 2. You should disable peripheral 1 `CS` pin by setting it to `HIGH`, and enable peripheral 2 `CS` pin by setting it to `LOW`:

```
digitalWrite(CS_1, HIGH); // disable CS pin from peripheral 1
digitalWrite(CS_2, LOW);  // enable CS pin to read from peripheral

/*
 use any SPI functions to communicate with peripheral 2
*/
```

ESP32 Using Two SPI Bus Interfaces (Use HSPI and VSPI simultaneously)

To communicate with multiple SPI peripherals simultaneously, you can use the ESP32 two SPI buses (HSPI and VSPI). You can use the default HSPI and VSPI pins or use custom pins.



Briefly, to use HSPI and VSPI simultaneously, you just need to.

1) First, make sure you include the SPI library in your code.

2) Initialize two `SPIClass` objects with different names, one on the HSPI bus and another on the VSPI bus. For example:

```
vspi = new SPIClass(VSPI);
hspi = new SPIClass(HSPI);
```

3) Call the `begin()` method on those objects.

```
vspi.begin();
hspi.begin();
```

You can pass custom pins to the `begin()` method if needed.

```
vspi.begin(VSPI_CLK, VSPI_MISO, VSPI_MOSI, VSPI_SS);
hspi.begin(HSPI_CLK, HSPI_MISO, HSPI_MOSI, HSPI_SS);
```

4) Finally, you also need to set the SS pins as outputs. For example:

```
pinMode(VSPI_SS, OUTPUT);
pinMode(HSPI_SS, OUTPUT);
```

Then, use the usual commands to interact with the SPI devices, whether you're using a sensor library or the SPI library methods.

You can find an example of how to use multiple SPI buses on the arduino-esp32 SPI library. See the example below:

```
 * them are available to use, HSPI and VSPI. Simply using the SPI
 * as illustrated in Arduino examples will use VSPI, leaving HSPI
 *
 * However if we simply intialise two instance of the SPI class fo
 * of these buses both can be used. However when just using these
 * way only will actually be outputting at a time.
 *
 * Logic analyser capture is in the same folder as this example as
 * "multiple_bus_output.png"
 *
 * created 30/04/2018 by Alistair Symonds
 */
#include <SPI.h>

// Define ALTERNATE_PINS to use non-standard GPIO pins for SPI bus

#ifdef ALTERNATE_PINS
  #define VSPI_MISO   2
  #define VSPI_MOSI   4
  #define VSPI_SCLK   0
  #define VSPI_SS     33

  #define HSPI_MISO   26
```

View raw code

Wrapping Up

This article was a quick and simple guide showing you how to use SPI communication with the ESP32 using the Arduino core—with the ESP32 acting as a controller (*master*).

In summary, the ESP32 has four SPI buses, but only two can be used to control peripherals, the HSPI and VSPI. Most ESP32 have pre-assigned HSPI and VSPI

You can use the HSPI and VSPI buses simultaneously to drive multiple SPI peripherals, or you can use multiple peripherals on the same bus as long as their CS pin is connected to a different GPIO.

We didn't dive deeply into examples, because each sensor, library, and case scenario is different. But, now you should have a better idea of how to interface one or multiple SPI devices with the ESP32.

For more detailed information about the SPI Master driver on the ESP32, you can check the espressif official documentation.

We didn't cover setting the ESP32 as an SPI slave, but you can check these examples.

We hope you find this tutorial useful. We have a similar article, but about I2C communication protocol. Check it out on the following link:

- ESP32 I2C Communication: Set Pins, Multiple Bus Interfaces and Peripherals (Arduino IDE)

Learn more about the ESP32 with our resources:

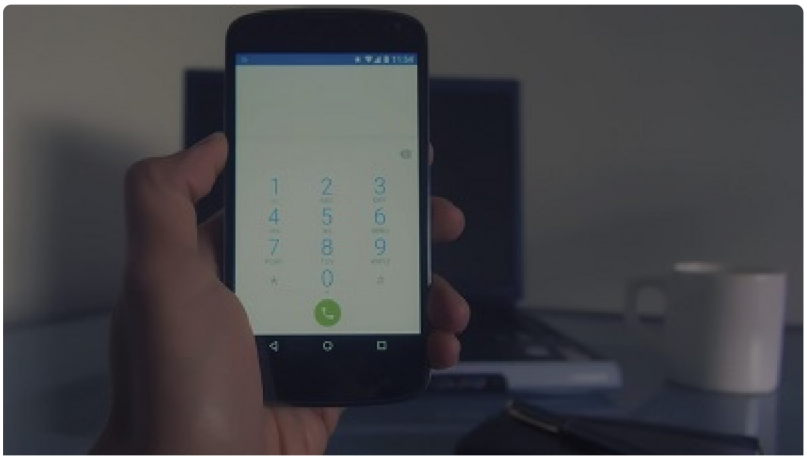
- Learn ESP32 with Arduino IDE
- Build Web Servers with ESP32 and ESP8266
- Firebase Web App with ESP32 and ESP8266
- Free ESP32 Projects and Tutorials

Thanks for reading.

[eBook] Build Web Servers with ESP32 and ESP8266 (2nd Edition)

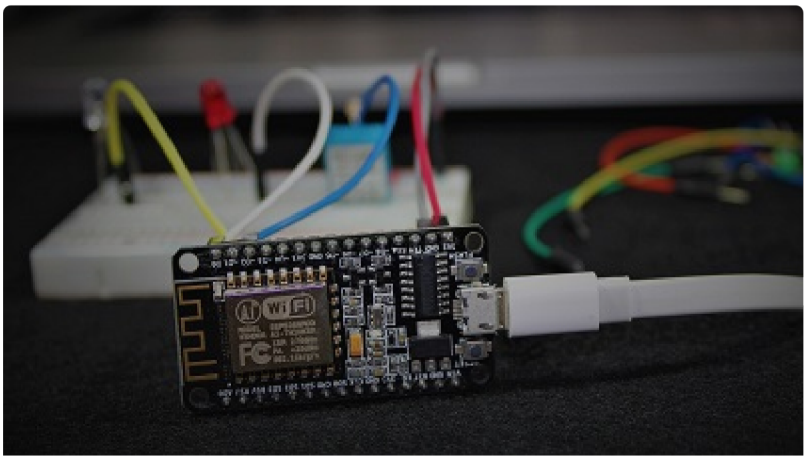
Build Web Server projects with the ESP32 and ESP8266 boards to control outputs and monitor sensors remotely. Learn HTML, CSS, JavaScript and client-server communication protocols **DOWNLOAD »**

Recommended Resources

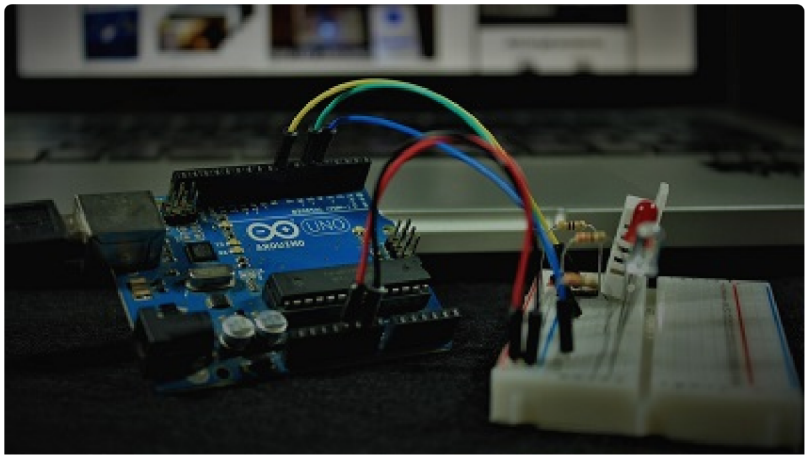


Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266

Menu



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…

ESP32: ESP-NOW Encrypted Messages

ESP32 MQTT – Publish DS18B20 Temperature Readings (Arduino IDE)

Install uPyCraft IDE – Windows PC Instructions

Enjoyed this project? Stay updated by subscribing our newsletter!

Your Email Address

SUBSCRIBE

5 thoughts on “ESP32 SPI Communication: Set Pins, Multiple SPI Bus Interfaces, and Peripherals (Arduino IDE)”

Dale Bartel

August 19, 2022 at 5:48 am

Can you please show how to use the ESP32 as an SPI device (slave).
I want to use it to read sensors for a Raspberry Pi as most sensors only have Arduino libraries.

Thank you
Dale

Reply

Sara Santos

August 19, 2022 at 9:32 am

Hi.
At the moment, we don't have any tutorials about setting the ESP32 as an SPI slave.
But you can find some examples on the following link:
https://github.com/hideakitai/ESP32SPISlave
Regards,
Sara

Reply

William Vancura
August 24, 2022 at 2:27 am

Is there a special version of SPI.h library? I compiled this and got this error several times

C:\ESP32_SPI_Example\ESP32_SPI_Example.ino: In function 'void setup()':
ESP32_SPI_Example:80:17: error: 'class SPIClass' has no member named 'pinSS'
pinMode(vspi->pinSS(), OUTPUT); //VSPI SS

Thanks
Bill

Reply

Sara Santos
August 24, 2022 at 10:26 pm

Hi.
Do you have an ESP32 selected in Tools>Board?
What's the version of your ESP32 installation?
Regards,
Sara

Reply

explainist
October 15, 2022 at 8:00 pm

the program above finds the HSPI pins, but not the VSPI pins.

On a TTGO loRa OlED SD card T3 V2.16 board, that gives me lORA pins but not SD card pins

Absolutely nothing I try gets me SD card on the T3 V2.16, or I2C on the T3 V2.16 or TTGO T Beam. I did get a DS3231 to work on the T Beam, but no BME280 , and OlED only on the T3 V2.16

Reply

Leave a Comment

Name *

Email *

Website

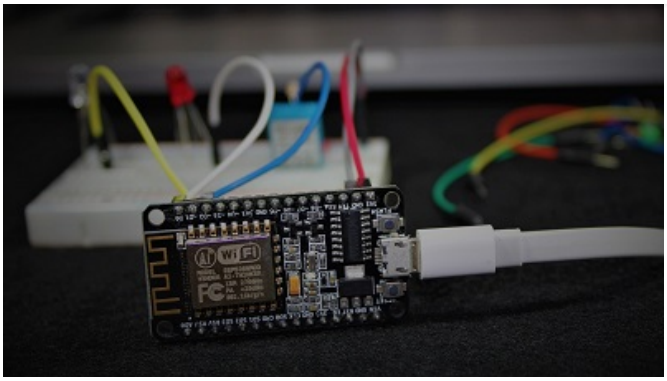
☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

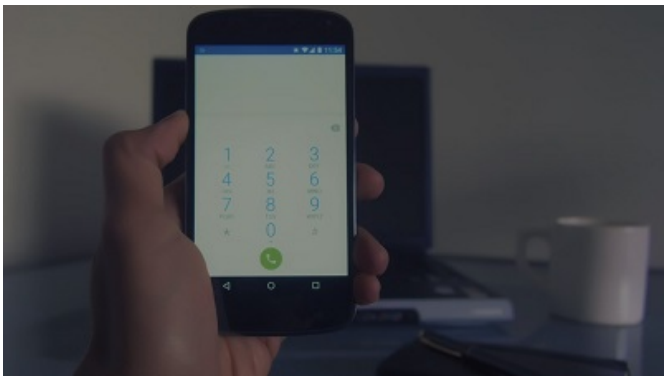
Post Comment

Visit Maker Advisor – Tools and Gear for makers, hobbyists and DIYers »

Menu



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



Build Web Servers with ESP32 and ESP8266 » boards to control outputs and monitor sensors remotely.