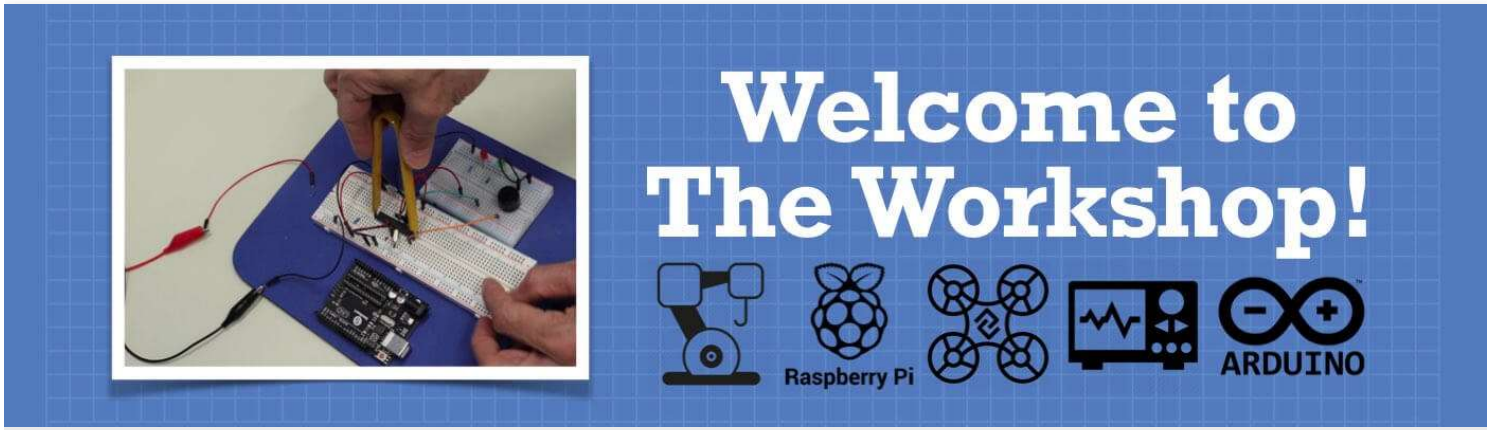


DroneBot Workshop

Build your own Electronics, IoT, Drones and Robots – Welcome to the Workshop!

Getting started with ESP32

It's time to move beyond the Arduino and work with a more modern microcontroller. Today we will get started with the popular ESP32.

Table of Contents

Introduction

When we think about using a microcontroller for a project we usually consider an Arduino. It's inexpensive, easy to use and has a generous number of digital I/O ports, and a few analog inputs as well.

But the Arduino, for all of its wonderful benefits, is lacking in a number of areas. The first one is speed, the popular Arduino AVR series of boards run at 16 MHz. That's certainly fast enough to build thousands of applications, but it's a bottleneck for others.

The Arduino certainly has enough digital outputs and inputs to satisfy most requirements, and its analog inputs are also useful. But adding features like WiFi and Bluetooth requires external components.

Let's face it, the Arduino has been around since 2005. That's fifteen years, which in terms of technology is eons.

The Arduino is well-loved here in the DroneBot Workshop and I'll continue to use it for many projects and experiments. But I also feel that it's time to explore other microcontrollers.

Introducing ESP32

The ESP32 is actually a series of microcontroller chips produced by Espressif Systems in Shanghai. It is available in a number of low-cost modules.

The ESP32 is an updated version of the ESP8266, which was a chip that took experimenters in the western world by “surprise” in 2014. The original ESP8266 was introduced on a module called the ESP-01, which had very little English documentation so it’s capabilities were largely unknown at the time. Once the documentation was translated into English many experimenters soon became aware of the power of the ESP8266, and it quickly became very popular.

The ESP32 improved upon the ESP32 design in a number of ways. It offers both Bluetooth and BLE (Bluetooth Low Energy), whereas the ESP8266 only has WiFi (which, of course, the ESP32 also has). It is faster and is available in a dual-core design. It is also capable of operating in an ultra-low-power mode, ideal for battery-powered applications.

Other features of the ESP32 include:

- Up to 18 12-bit Analog to Digital converters.
- Two 8-bit Digital to Analog converters.
- 10 capacitive touch switch sensors.
- Four SPI channels.
- Two I2C interfaces.
- Two I2S interfaces (for digital audio).
- Three UARTs for communications.
- Up to 8 channels of IR remote control.
- Up to 16 channels of LED PWM (pulse width modulation).
- An integrated Hall-effect sensor.
- An ultra-low-power analog preamp.
- An internal low-dropout regulator.

Note that many of the pins on the ESP32 share a number of the above functions, so not all of them are available concurrently.

ESP32 Modules

There are many ESP32 modules available for experimenters. Just about any of them can be used for the experiments here.

Many of these boards have an integrated micro-USB connector that will simplify programming. Some boards don’t have this feature and require an external FTDI adapter for programming.

Most of these boards are based upon the ESP32-WROOM chip.

Popular Articles

Getting started with the ESP32-CAM

May 24, 2020

By DroneBot Workshop | 134 Comments

Today we will look at the amazing ESP32-CAM module from A-Thinker. This 10-dollar

The ESP32 DEV KIT and ESP32 NODEMCU boards are quite popular and available at Amazon and eBay.

Adafruit makes the HUZZAH32 board.

Sparkfun has the ~~ESP32 Thing~~ board.

And the popular ESP32-Cam board integrates a small video camera and a microSD card socket along with an ESP32 (this board will require an FTDI adapter for programming).

Please note that these boards share many features but they don't have the same pinouts. In our experiments, I'll be referring to the pin function (i.e. GPIO 4) instead of an actual pin number. This will allow you to use a different board than the one I am using.

Programming the ESP32

The ESP32 can be programmed using many different development environments. Code can be written in C++ (like the Arduino) or in MicroPython.

To make use of all of the ESP32 features Espressif provided the Espressif IoT Development Framework, or ESP-IDF.

For beginners, an easy way to get started is by using the familiar Arduino IDE. While this is not necessarily the best environment for working with the ESP32, it has the advantage of being a familiar application, so the learning curve is flattened.

We will be using the Arduino IDE for our experiments.

Getting started with the Arduino IDE

Before we can use the Arduino IDE with an ESP32 board we will need to add the ESP32 boards using the Arduino IDE Board Manager.

If you look online for instructions for setting up the IDE you may run into a long and complex procedure that actually isn't necessary anymore. As long as you are using a recent version of the Arduino IDE you can use the following instructions, which are much simpler.

module features a 2MP camera, microSD card...

Getting started with ESP32

April 2, 2020

By DroneBot Workshop | 88 Comments

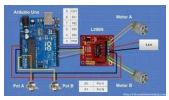
Time to move up to another microcontroller, the ESP32. This amazing device has multiple I/O ports, WiFi, Bluetooth and BLE,...

Stepper Motors with Arduino – Getting Started with Stepper Motors

February 10, 2018

By DroneBot Workshop | 97 Comments

Stepper motors are used in a variety of devices ranging from 3D printers and CNC machines to Blu Ray drives,...



Controlling DC Motors with the L298N Dual H-Bridge and an Arduino

March 11, 2017

ESP32 for Arduino IDE

In order to be able to work with the ESP32 you will need to add an additional source to the Arduino IDE Board Manager and then install the ESP32 boards.

This is actually a lot simpler than it sounds thanks to the folks at Espressif. They have provided a link to a JSON file that takes care of almost everything for you.

If you are not familiar with JSON it is a format of text file that allows structured information to be shared between computers. In this respect, it is similar to XML.

Add Boards Manager Entry

Here is what you need to do to install the ESP32 boards into the Arduino IDE:

1. Open the Arduino IDE. Make sure that you are at version 1.8 or higher, if not then update your IDE with the latest version.
2. Click on the *File* menu on the top menu bar.
3. Click on the *Preferences* menu item. This will open a Preferences dialog box.
4. You should be on the *Settings* tab in the Preferences dialog box by default.
5. Look for the textbox labeled "*Additional Boards Manager URLs*".
6. If there is already text in this box add a coma at the end of it, then follow the next step.
7. Paste the following link into the text box – ***https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json***
8. Click the OK button to save the setting.

The textbox with the JSON link in it is illustrated here:

By DroneBot Workshop | 164 Comments

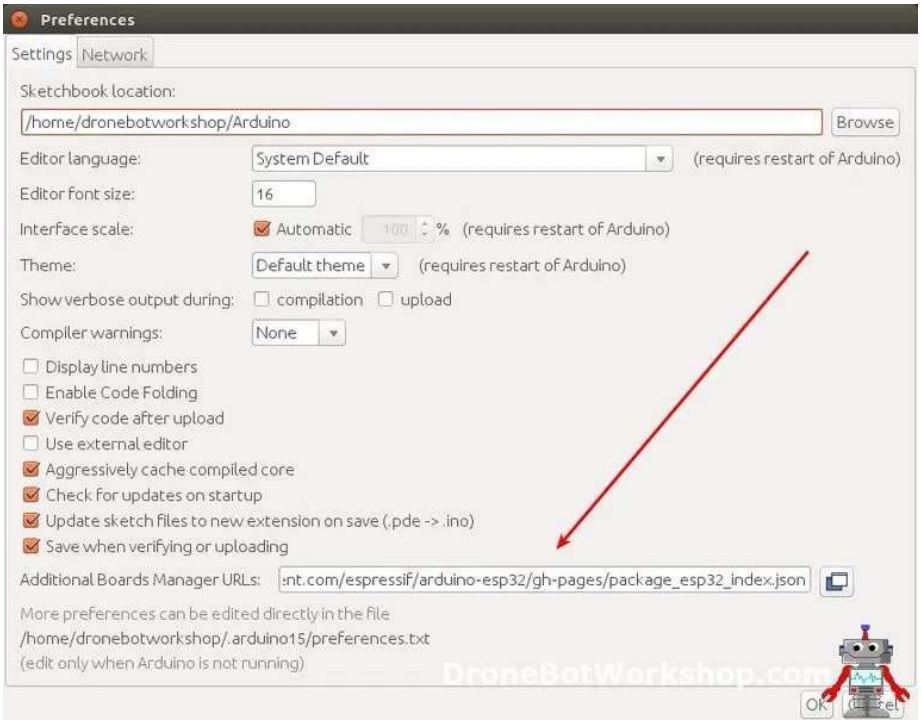
Controlling DC Motors is an essential skill for constructing robots and other hobby projects. An easy way to control DC...

Using Inexpensive 433MHz Transmit and Receive Modules with Arduino

February 17, 2018

By DroneBot Workshop | 51 Comments

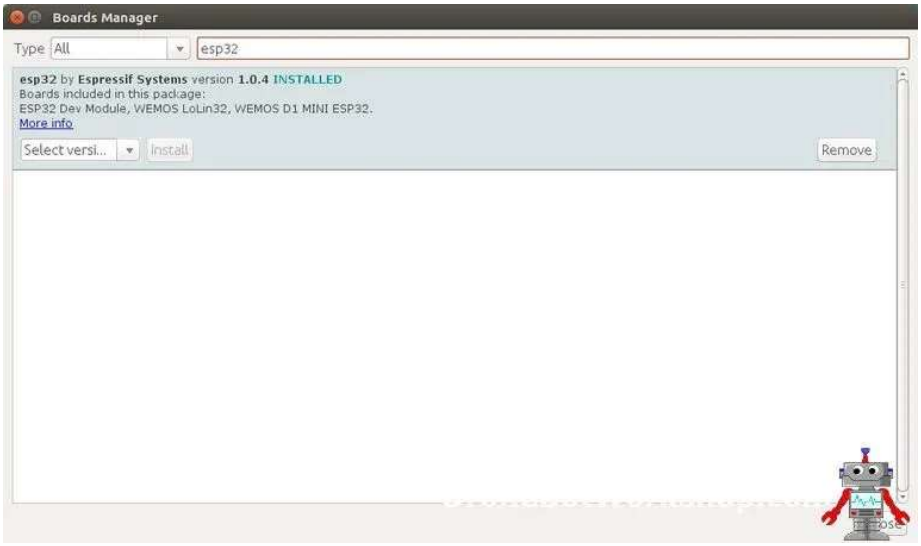
Those inexpensive RF transmitter and receiver modules that you can get on eBay and Amazon are perfect when you need...



Next, you will need to use the new entry to actually add the ESPO32 boards to your Arduino IDE. You do that by following this procedure:

1. In the Arduino IDE click on the *Tools* menu on the top menu bar.
2. Scroll down to the *Board:* entry (i.e. *Board: Arduino/Genuino Uno*).
3. A submenu will open when you highlight the *Board:* entry.
4. At the top of the submenu is *Boards Manager*. Click on it to open the Boards Manager dialog box.
5. In the search box in the Boards Manager enter "*esp32*".
6. You should see an entry for "*esp32 by Espressif Systems*". Highlight this entry and click on the *Install* button.
7. This will install the ESP32 boards into your Arduino IDE

88



If you go back into the *Boards:* submenu you should now see a number of ESP32 boards. You'll need to select the board that matches (or is equivalent to) the ESP32 board you have purchased.

For the record, I used an ESP-32S NodeMCU board and chose the "Node32s" board in the Boards Manager.

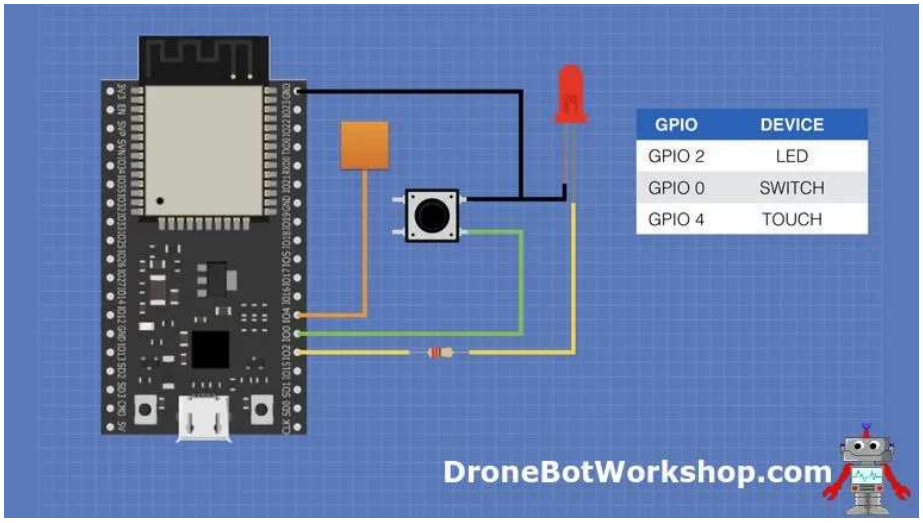
You have probably finished setting up your Arduino IDE! The reason I say "probably" is that it is possible that you may need to install some additional Python files.

The easiest way to figure that out is to compile and upload a program to your ESP32 board and see if you get an error. Don't worry, I'll show you how to fix that error if you do!

Experiment Hookup

Before we can upload a test program we will need to hookup our ESP32 board.

The following diagram shows the connections we'll need to perform all of the experiments and demos included in this article. Note that the ESP32 pins are specified by their GPIO names and not pin numbers, as different ESP32 boards will have different pin numbers.



You can use pretty well any LED, for a dropping resistor any values between 150 and 470 ohms will work well. Any momentary-contact push button will suffice.

The small gold square in the diagram is a Touchpad. You can use any metallic object that you can connect a wire to, a small piece of PCB would work great. I just used a small piece of bare wire on my breadboard.

Now that we are all hooked up it's time to try our first program.

Hello World – Blink for ESP32

When using a new development environment or programming for a new device it is traditional to create a "Hello World" program. Far be it from me to dispense with tradition!

The "Hello World" for microcontrollers is the Blink sketch which, as I'm sure you already know, simply flashes an LED on and off. While "Hello World" programs and sketches are usually of little practical use they serve a number of functions:

- They familiarize you with the operation of your development library. In this particular case they let you practice uploading programs to the ESP32 which, as you will soon see, is done a bit differently than an Arduino.
- They get you familiar with the programming syntax. In this case we are using the same C++ that the Arduino uses, so you are likely already familiar with the programming syntax.
- They let you verify that your hardware is working correctly.

Here is the classic Arduino Blink sketch, rewritten for the ESP32 module.

```
/*
  ESP32 Blink
  esp32-blink.ino
  Rewrite of classic Blink sketch for ESP32
  Use LED on GPIO2

  DroneBot Workshop 2020
  https://dronebotworkshop.com
*/

// LED on GPIO2
int ledPin = 2;

void setup()
{
    // Set LED as output
    pinMode(ledPin, OUTPUT);

    // Serial monitor setup
    Serial.begin(115200);
}

void loop()
{
    Serial.print("Hello");
    digitalWrite(ledPin, HIGH);

    delay(500);

    Serial.println(" world!");
    digitalWrite(ledPin, LOW);

    delay(500);
}
```

Aside from using GPIO pin 2 (instead of Arduino digital output pin 13) and adding the serial monitor it's no different from the Arduino Blink sketch.

Note that the serial monitor is run at a higher baud rate than most Arduino sketches. While it isn't really necessary with this sketch it is often done with ESP32 sketches, as the microcontroller runs much faster than the Arduino AVR boards do.

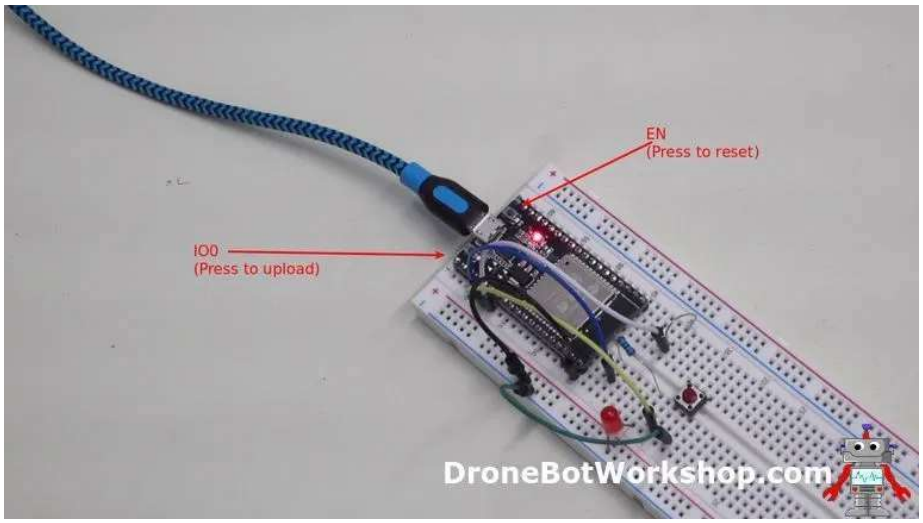
Uploading the program

After you load the sketch into your Arduino IDE it will need to be compiled and then transferred to the ESP32 board.

With an Arduino you just have to hit the *Upload* button and both of these steps will proceed automatically. With the ESP32,

however, there is an additional step.

You begin by doing exactly the same thing you would with an Arduino – hit the *Upload* button (the one with the right-pointing arrow). This will start the compiler. You'll probably notice that it takes longer to compile ESP32 programs than it does for Arduino sketches.

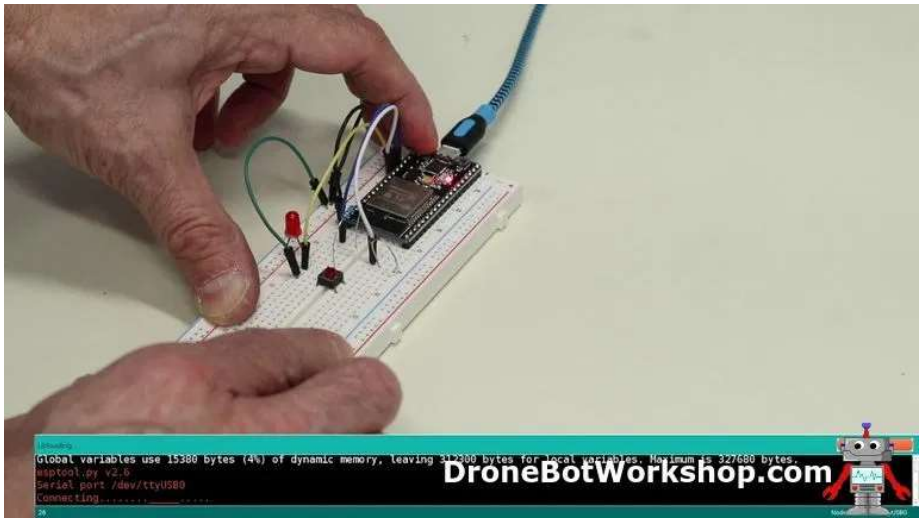


Watch the status screen at the bottom of the Arduino IDE. When the compiling process is finished it will print "Connecting", followed by a string of periods.

When you see this you'll need to press the *BOOT* button (sometimes labeled *IO0*) and hold it down until you see the upload progressing, which you'll know is happening when you see the upload progress displayed in percentage.

After the program is uploaded you will need to press the *ENABLE* or *RESET* button to start it (some boards don't require this step).

88



And if all goes well you'll be rewarded with a flashing LED!

But it's also possible that you'll get an error message about a missing module named "serial". If that happens don't worry, there is a fix.

Problem with "No module named serial"

This issue occurs because one or more of the underlying Python programs is not installed on your computer. The compiler is looking for a program called *pyserial* and is not finding it.

On Mac and Linux machines this is usually already installed, but not always – in fact the computer I used in the video (Ubuntu Linux 16.04) didn't have *pyserial* installed. So I downloaded and installed it using this (very long) command-line string:

sudo usermod -a -G dialout $USER && sudo apt-get install git && wget https://bootstrap.pypa.io/get-pip.py && sudo python get-pip.py && sudo pip install pyserial && mkdir -p ~/Arduino/hardware/espressif && cd ~/Arduino/hardware/espressif && git clone https://github.com/espressif/arduino-esp32.git esp32 && cd esp32 && git submodule update –init –recursive && cd tools && python3 get.py

As you need to sudo you'll need to enter your password to supply permissions.

Windows users may have additional difficulties installing *pyserial*. Adafruit published some detailed instructions that should be of assistance.

Using WiFi

Now that we have our programming environment working we can start experimenting with the ESP32.

When you install the Board Manager update and then select an ESP32 board you'll be provided with a large number of test programs that illustrate how to work with the many features of the ESP32.

We will go through some of these programs, starting with the ones that utilize the WiFi features.

WiFi on ESP32

The ESP32, like its predecessor the ESP8266, supports WiFi on the 2.4 GHz band. It supports WiFi protocols 802.11 b/g/n with a maximum data transfer rate of 150 Mbps.

The device has an adjustable transmit power of up to 20.5 dBm, using lower power will decrease the current requirements as the WiFi radio can consume a fair amount of current.

An advanced feature of the ESP32 WiFi is "antenna diversity". This allows you to use some of the GPIO pins to control an external RF switch connected to multiple antennas, and then switch the WiFi radio to the most appropriate antenna (and change it when signal conditions vary).

WiFi Modes

The ESP32 can be used in two different WiFi modes.

Station (STA) Mode

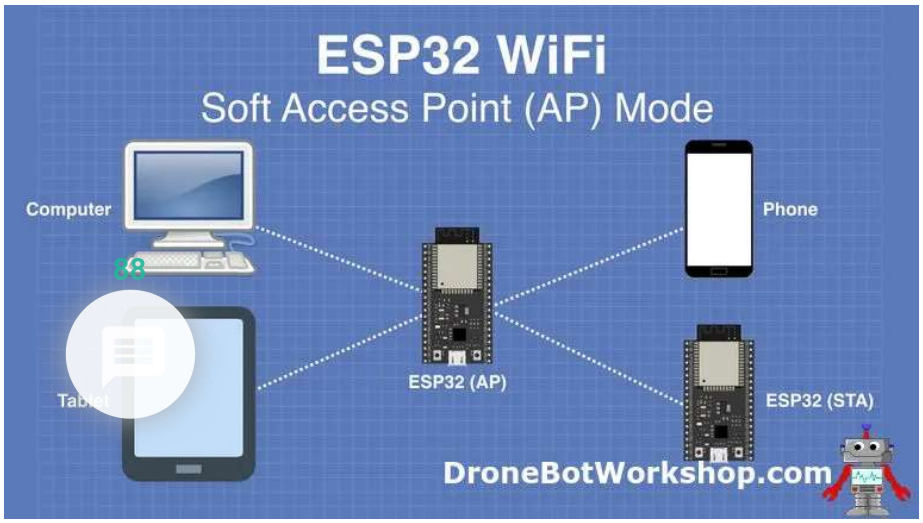
In STA, or Station, mode the ESP32 acts as a WiFi station or client. The arrangement is illustrated here:

In this mode the ESP32 needs to know the SSID and Password to access the WiFi network (on an unsecured network there is no password requirement).

The ESP32 is provided with an network IP address using the routers internal DHCP server. It can then be accessed using that address. It is also possible to assign a fixed IP address to the ESP32, which is useful if you are using it as a web server where a changing IP address would cause difficulties for other clients.

Soft Access Point (AP) Mode

In Soft Access Point, or AP, mode, the ESP32 provides a WiFi connection for external devices. These devices can be computers, phones, tablets, IoT devices or even other ESP32's configured in STA mode.



The ESP32 can support a maximum of five external devices in AP mode. It has a default IP address of 192.168.4.1 and it will provide DHCP services to the externally connected devices. The default IP address can be changed if it conflicts with existing devices.

This allows the ESP32 to create its own IP network, independent of any existing WiFi networks. You can secure the network with a password and choose the SSID (network name).

Incidentally, this is referred to as a “soft” access point as the ESP32 does not provide a hardwired connection to the Internet or other existing networks.

WiFi Scanner

The first example sketch we will look at is the WiFi Scanner. As its name would imply, this program scans for local WiFi networks. It then prints the results on the serial monitor. The results include the network SSID (name), the signal strength in dBm and an indicator if the network is secured.

Finding the Example Programs

This sketch, along with all of the other demonstration programs we’ll be examining, can be accessed as follows:

- Open the Arduino IDE.
- Select the *File* menu from the menu bar at the top of the IDE.
- Scroll down to *Examples*. A submenu will open.
- Scroll down through the examples until you find a section of examples for the ESP32. This will be labeled according to the ESP32 board you selected, the one I’m using says *Examples for Node32s*.
- You will see a number of menu items below this. Each one has its own submenu with one or more examples.

88

Load WiFiScan

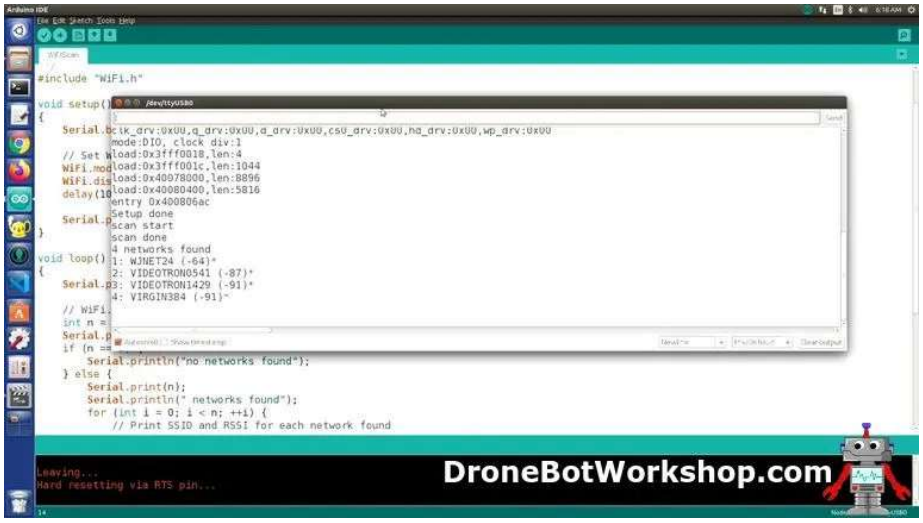
The example we are going to use now is under *WiFi* and is called *WiFiScan*.

```
1  /*
2   *  This sketch demonstrates how to scan WiFi networks.
```

```
 *  The API is almost the same as with the WiFi Shield libr
 *  the most obvious difference being the different file yo
 */
#include "WiFi.h"

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

    // Set WiFi to station mode and disconnect from an AP i
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);

    Serial.println("Setup done");
}

void loop()
{
    Serial.println("scan start");

    // WiFi.scanNetworks will return the number of networks
    int n = WiFi.scanNetworks();
    Serial.println("scan done");
    if (n == 0) {
        Serial.println("no networks found");
    } else {
        Serial.print(n);
        Serial.println(" networks found");
        for (int i = 0; i < n; ++i) {
            // Print SSID and RSSI for each network found
            Serial.print(i + 1);
            Serial.print(": ");
            Serial.print(WiFi.SSID(i));
            Serial.print(" (");
            Serial.print(WiFi.RSSI(i));
            Serial.print(")");
            Serial.println((WiFi.encryptionType(i) == WIFI_
            delay(10);
        }
    }
    Serial.println("");

    // Wait a bit before scanning again
    delay(5000);
}
```

The code makes use of the WiFi library and is fairly self-explanatory. The command WiFi.scanNetworks scans the available networks and enters them into an array. The array is then stepped through to produce the scan results.

Load the code to the ESP32, remember to hold down the BOOT button when required. Then open your serial monitor (be sure it is set to 115,200 baud).



When you first open the monitor there is a good chance you will just see gibberish, this is because the serial signal has not synched up correctly. This can easily be resolved by simply pressing the *RESET* button on the ESP32.

You should now see a scan of the networks available in your location.

Brownout Detector Enabled

When using the built-in radio for WiFi or Bluetooth you may encounter a “Brownout Detector” message when the ESP32 is reset. This generally means that the ESP32 is not receiving enough current.

This generally is not cause for alarm and you can proceed as normal. If it becomes problematic try switching USB cables or using a powered USB hub, in order to provide extra current to the ESP32.

WiFi Access Point

The next example we will look at uses the ESP32 in AP, or Soft Access Point, mode. You’ll find this example with the previous one, in the WiFi submenu. It is called *WiFiAccessPoint*.

This program places the ESP32 in AP mode and then creates a very simple web server and web page. The web page has two links on it, and these links can be used to control the LED status on the ESP32.

```
/*
  WiFiAccessPoint.ino creates a WiFi access point and provi

  Steps:
  1. Connect to the access point "yourAp"
  2. Point your web browser to http://192.168.4.1/H to turn
     OR
     Run raw TCP "GET /H" and "GET /L" on PuTTY terminal wi

  Created for arduino-esp32 on 04 July, 2018
  by Elochukwu Ifediora (fedy0)
*/

#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

#define LED_BUILTIN 2   // Set the GPIO pin where you conne

// Set these to your desired credentials.
const char *ssid = "yourAP";
const char *password = "yourPassword";

WiFiServer server(80);


void setup() {
  pinMode(LED_BUILTIN, OUTPUT);

  Serial.begin(115200);
  Serial.println();
  Serial.println("Configuring access point...");

  // You can remove the password parameter if you want the
  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  server.begin();

  Serial.println("Server started");
}

void loop() {
  WiFiClient client = server.available();   // listen for i

  if (client) {                             // if you get a
    Serial.println("New Client.");           // print a mes
    String currentLine = "";                // make a Strin
    while (client.connected()) {            // loop while t
      if (client.available()) {             // if there's b
        char c = client.read();             // read a byte,
        Serial.write(c);                    // print it out
        if (c == '\n') {                    // if the byte

          // if the current line is blank, you got two newl
          // that's the end of the client HTTP request, so
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response co
            // and a content-type so the client knows what'
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // the content of the HTTP response follows the
```

```
            client.print("Click <a href=\"/H\">here</a> to
            client.print("Click <a href=\"/L\">here</a> to

            // The HTTP response ends with another blank li
            client.println();
            // break out of the while loop:
            break;
          } else {    // if you got a newline, then clear c
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything els
          currentLine += c;      // add it to the end of th
        }

        // Check to see if the client request was "GET /H"
        if (currentLine.endsWith("GET /H")) {
          digitalWrite(LED_BUILTIN, HIGH);               //
        }
        if (currentLine.endsWith("GET /L")) {
          digitalWrite(LED_BUILTIN, LOW);                //
        }
      }
    }
    // close the connection:
    client.stop();
    Serial.println("Client Disconnected.");
  }
}
```

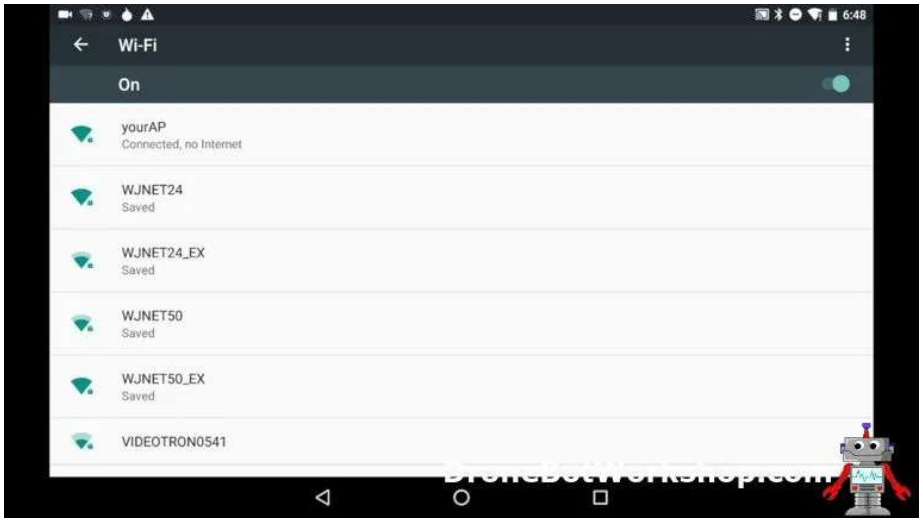
By default the SSID of the access point is “yourAP” and the password is “yourPassword”. You can modify both of these constants if you wish.

By default the IP address of the web server will be 192.168.4.1. You’ll need to know that to test out the sketch.

Load the sketch to your ESP32 and reset it. You can monitor the reset using the serial monitor.

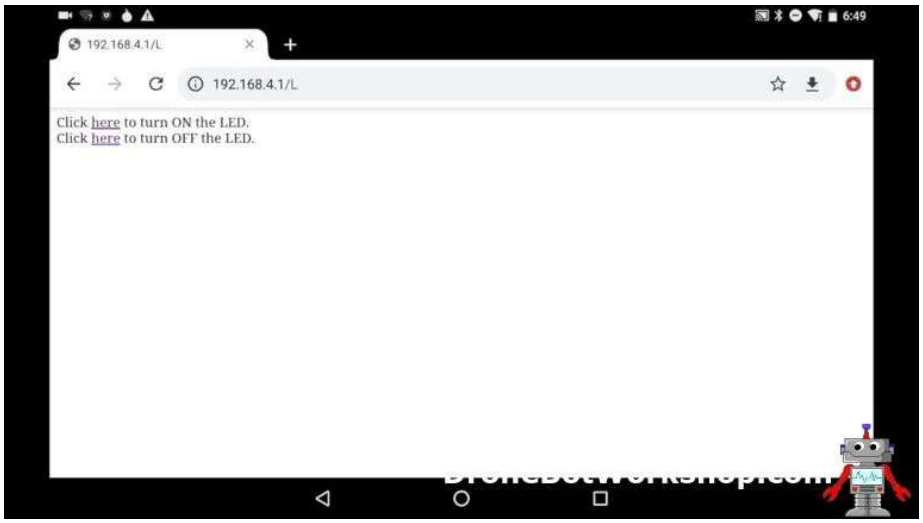
Now use a device like a notebook, tablet or phone and scan your available WiFi networks. You should see the “yourAP” network listed. Connect to the network with your device, using the password “yourPassword”.

88



Once the connection is established, open a web browser on the connected device and navigate to 192.168.4.1. You should see a web page with two links on it, one to turn the LED on and one to turn it off.

Click the links and observe the LED that you wired to the ESP32 (or the built in LED on the module itself). You should be able to control it from the web page.



This simple example can be used as the basis for code of your own.

88

Simple WiFi Server

The Simple WiFi Server example performs a similar function to the previous sketch. The difference is that in this case the ESP32 is used in STA (station) mode, so it creates a web server

on your own network. You can control the LED from any computer or device connected to your network.

You'll also find this sketch in the WiFi submenu, it is called SimpleWiFiServer.

```
/*
 WiFi Web Server LED Blink

 A simple web server that lets you blink an LED via the we
 This sketch will print the IP address of your WiFi Shield
 to the Serial monitor. From there, you can open that addr
 to turn on and off the LED on pin 5.

 If the IP address of your shield is yourAddress:
 http://yourAddress/H turns the LED on
 http://yourAddress/L turns it off

 This example is written for a network using WPA encryptio
 WEP or WPA, change the Wifi.begin() call accordingly.

 Circuit:
 * WiFi shield attached
 * LED attached to pin 5

 created for arduino 25 Nov 2012
 by Tom Igoe

ported for sparkfun esp32
31.01.2017 by Jan Hendrik Berlin

 */

#include <WiFi.h>

const char* ssid     = "yourssid";
const char* password = "yourpasswd";

WiFiServer server(80);

void setup()
{
    Serial.begin(115200);
    pinMode(2, OUTPUT);      // set the LED pin mode

    delay(10);

    // We start by connecting to a WiFi network

    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
```

```
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());

    server.begin();

}

int value = 0;

void loop(){
 WiFiClient client = server.available();   // listen for i

  if (client) {                             // if you get
    Serial.println("New Client.");           // print a me
    String currentLine = "";                // make a Stri
    while (client.connected()) {            // loop while
      if (client.available()) {             // if there's
        char c = client.read();             // read a byte
        Serial.write(c);                    // print it ou
        if (c == '\n') {                    // if the byte

          // if the current line is blank, you got two new
          // that's the end of the client HTTP request, so
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response c
            // and a content-type so the client knows what
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // the content of the HTTP response follows th
            client.print("Click <a href=\"/H\">here</a> to
            client.print("Click <a href=\"/L\">here</a> to

            // The HTTP response ends with another blank l
            client.println();
            // break out of the while loop:
            break;
          } else {    // if you got a newline, then clear
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything el
          currentLine += c;      // add it to the end of t
        }

        // Check to see if the client request was "GET /H"
        if (currentLine.endsWith("GET /H")) {
          digitalWrite(2, HIGH);               // GET /H t
        }
        if (currentLine.endsWith("GET /L")) {
          digitalWrite(2, LOW);                // GET /L t
        }
      }
    }
    // close the connection:
    client.stop();
    Serial.println("Client Disconnected.");
}
```

You'll have to make a few edits to this sketch before you can put it to use:

- You will need to enter your network SSID (line 30).
- You will need to enter your network password (line 31).
- You will need to edit the *pinMode* statement and change the “5” to a “2” (line 38).
- You will need to edit the *digitalWrite* statement and change the “5” to a “2” (line 105).
- You will need to edit the digitalWrite statement and change the “5” to a “2” (line 108).

The last three edits are because the sketch author, unfortunately, chose to hard-code the GPIO pin numbers instead of using a constant or variable. Another way around this would be to just move the LED to GPIO 5 and avoid those edits.

Once you have it edited load it to your ESP32. Open your serial monitor and reset the ESP32.

The serial monitor will display the IP address that the ESP32 was assigned from your routers DHCP server. Copy this address and then paste it into a web browser.

You should see a webpage very similar to the one displayed in the last experiment. Clicking on the links will control the LED.

Once again this is a good sample sketch from which to create your own web-controlled devices with the ESP32.

Using Bluetooth

One of the big advances that the ESP32 has over its predecessor the ESP8266 is its integrated Bluetooth capabilities.

With Bluetooth the ESP32 is capable of interfacing with numerous other Bluetooth devices, opening up a myriad of potential applications.

Bluetooth and BLE on ESP32

The ESP32 has both classic Bluetooth and BLE, or Bluetooth Low Energy. The device can act as either a Bluetooth client or server.

If you're unfamiliar with the difference between classic Bluetooth and BLE here is a brief explanation.

Classic Bluetooth 4.2

As its name would indicate Classic Bluetooth has been around for quite a while and is likely the Bluetooth that you are most familiar with.

Classic Bluetooth is used for continuous data transmission. It can be used for wireless audio applications, computer peripherals such as mice and keyboards, and for exchanging files between devices wirelessly.

We will look at an example of exchanging data with classic Bluetooth in a moment.

Bluetooth Low Energy

Bluetooth Low Energy, or BLE for short, exchanges data in short bursts. As its name would suggest it consumes very little current.

88

This makes BLE ideal for remote sensors and IoT devices, as these applications only require short bursts of data and are often battery-powered.

Using BLE it is possible to design products that measure battery life in years instead of months or days!

Another great application for BLE is targeted promotions. An example of this is a restaurant that can broadcast its menu or daily specials to potential customers in the vicinity. Apple's iBeacon protocol makes use of this feature.

Serial to Serial BT

To test the Bluetooth on our ESP32 we will use a simple demonstration sketch that is included in the examples, one called *SerialToSerialBT*. You'll find this sketch in the *BlueToothSerial* submenu.

```
1  //This example code is in the Public Domain (or CC0 license
2  //By Evandro Copercini - 2018
3  //
4  //This example creates a bridge between Serial and Classica
5  //and also demonstrate that SerialBT have the same function
6
7  #include "BluetoothSerial.h"
8
9  #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROI
10 #error Bluetooth is not enabled! Please run `make menuconfi
11 #endif
12
13 BluetoothSerial SerialBT;
14
15 void setup() {
16   Serial.begin(115200);
17   SerialBT.begin("ESP32test"); //Bluetooth device name
```

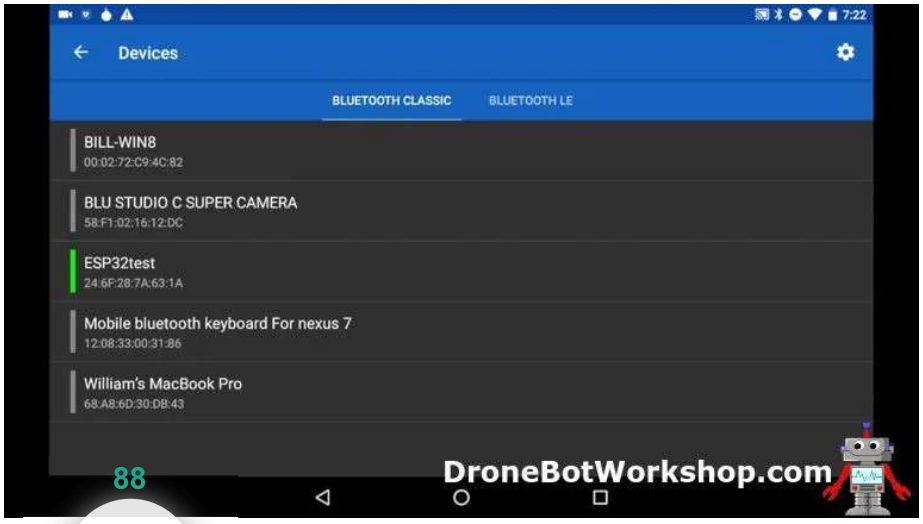
```
18    Serial.println("The device started, now you can pair it w
19  }
20
21  void loop() {
22    if (Serial.available()) {
23      SerialBT.write(Serial.read());
24    }
25    if (SerialBT.available()) {
26      Serial.write(SerialBT.read());
27    }
28    delay(20);
29  }
```

The sketch is actually pretty simple, as once again a library does all of the “heavy lifting“ for us.

The function of this program is to exchange data between the Arduino IDE Serial Monitor and an external Bluetooth Serial Terminal. What is written to one device is read by the other, and it works in both directions.

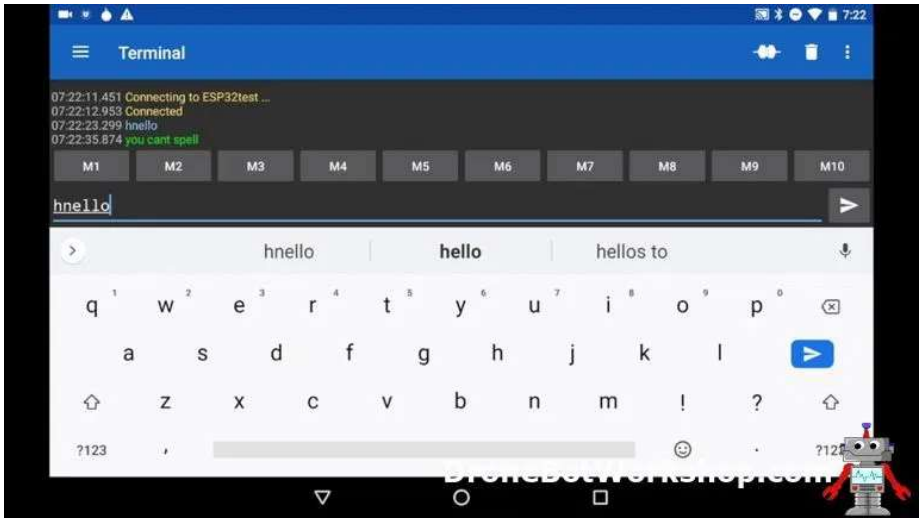
I’m sure you can see how useful this will be when developing your own Bluetooth applications.

In order to test the sketch you’ll need to have a serial Bluetooth terminal of some sort. If you have an Android device you can use the Serial Bluetooth Terminal App, which is free on the Google Play store. Apple IOS device users can try the Bluetooth Terminal App or ~~BlueTerm~~, both available on the Apple App Store.



Once you have a terminal you can load the sketch and pair your device with the ESP32. Look for a device named “ESP32test” (you can change this name if you like by modifying line 17 of the code).

Now use your Bluetooth terminal to type something. You should see the text you typed displayed on the Arduino IDE Serial Monitor. You can also type text into the serial monitor textbox, it should appear on your Bluetooth terminal.



More ESP32 Features

As we saw at the beginning of this article the ESP32 has a wealth of features aside from WiFi and Bluetooth.

The best way to learn about using these features is by running the examples provided by Espressif, justy as we did when learning about the WiFi and Bluetooth features.

You can find all of these examples under the ESP32 menu in the examples. Highlighting this menu will bring up a submenu, which in turn contains more submenus!

The following is a small sample of these example sketches.

Simple Time

One of the features of the ESP32 is its internal Real Time Clock. The SimpleTime sketch, which is found in the Time submenu, illustrates how you can set the clock using a NTP (Network Time Protocol) server on the internet.

```
#include <WiFi.h>
#include "time.h"

const char* ssid       = "YOUR_SSID";
const char* password   = "YOUR_PASS";

```

```
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 3600;
const int   daylightOffset_sec = 3600;

void printLocalTime()
{
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
}

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

  //connect to WiFi
  Serial.printf("Connecting to %s ", ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
  }
  Serial.println(" CONNECTED");

  //init and get the time
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  printLocalTime();

  //disconnect WiFi as it's no longer needed
  WiFi.disconnect(true);
  WiFi.mode(WIFI_OFF);
}

void loop()
{
  delay(1000);
  printLocalTime();
}
```

Since this program needs to attach to the internet you'll need to edit the code with your WiFi SSID and Password.

Calculating the GMT and Daylight Savings Offset

In order to get the clock set to your local timezone you will need to provide it with two values, which will be dependent upon your location.

- GMT Offset – The number of seconds your timezone differs from GMT. This can be a positive or negative value.
- Daylight Offset – If your area observes Daylight Savings Time then this is the number of seconds you advance the clock for DST.

To calculate the GMT Offset you'll need to know how many hours your timezone is offset from Greenwich Mean Time. You probably already know that value, but if you don't you can calculate using one of several calculators on the internet. Your value may be positive (if you are east of the Greenwich Observatory) or negative (if you live on the west side, i.e. North and South America).

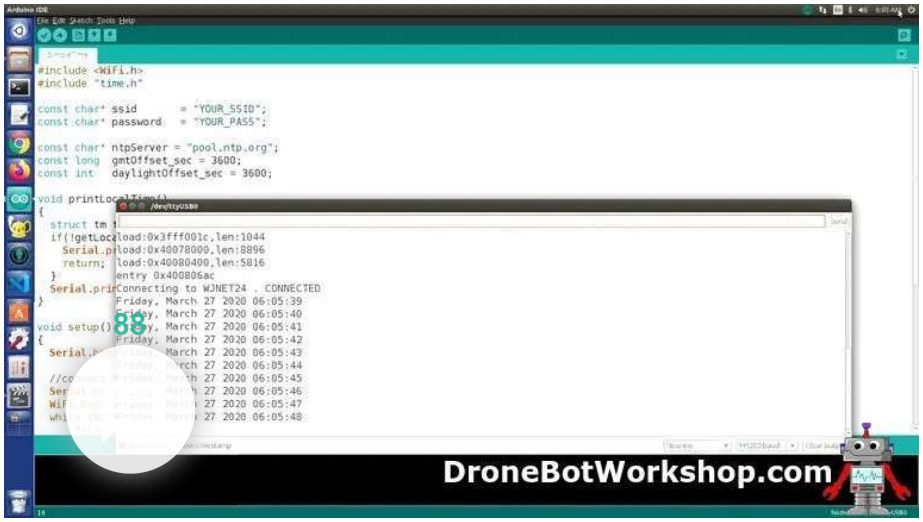
Next you will need to convert that figure into seconds, which means you'll need to multiply it by 3600, as there are 3600 seconds in an hour (60 x 60). The result is the GMT Offset, which you will enter on line 8 of the code.

I'll use my area as an example. I live in Montreal, which is in Eastern Time – same time zone as New York, Miami and Toronto. Our GMT offset is -5 hours, and we observe daylight savings time.

So my GMT offset will be -18000, which is 3600 x -5. Note that you don't use a comma within the number.

The Daylight Offset is 3600, which means that during daylight savings time our clocks are set one hour (3600 seconds) ahead. For most areas that observe daylight savings time this is the correct figure.

After editing the code with your WiFi parameters and the correct offset all you need to do is load the sketch to the ESP32, open the serial monitor and press reset.



You should see the ESP32 connect to your WiFi and then display the time every second. Since most computers are also synchronized to an internet time server the time value should match the value on your computer.

The uses for a sketch like this one are pretty obvious!

Hall Sensor

A Hall Sensor is a device that uses the Hall Effect to sense a magnetic field. I have covered these devices before in the article and video Stepper Motor with Hall Effect Limit & Homing Switches.

The ESP32 has an integrated Hall Effect sensor, so you can use it to detect the presence (or absense) of a magnetic field.

The *HallSensor* example sketch shows how to read the value of the integrated Hall sensor.

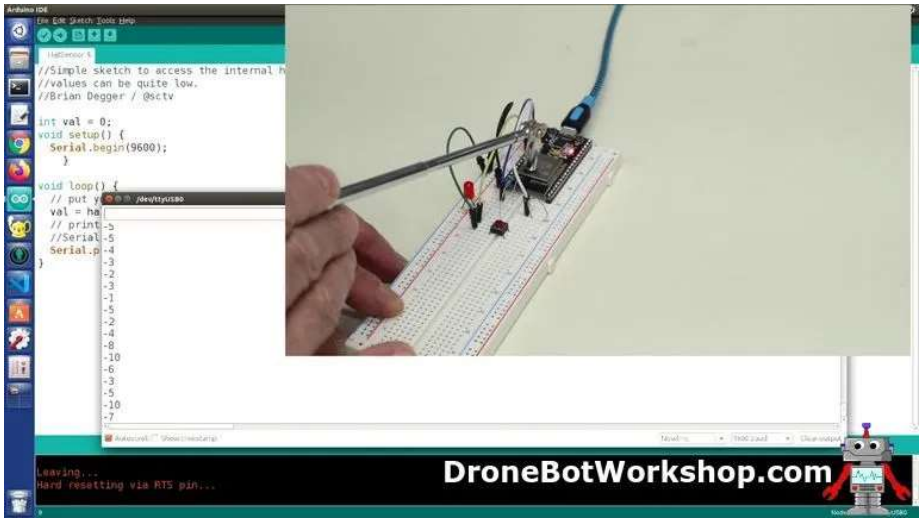
```
//Simple sketch to access the internal hall effect detector
//values can be quite low.
//Brian Degger / @sctv

int val = 0;
void setup() {
  Serial.begin(9600);
    }

void loop() {
  // put your main code here, to run repeatedly:
  val = hallRead();
  // print the results to the serial monitor:
  //Serial.print("sensor = ");
  Serial.println(val);//to graph
}
```

This sketch is about as simple as the Blink sketch, in fact in some ways it's even simpler!

The *hallRead()* function does all of the work, producing an integer value that indicates the hall sensor output.

Just load this sketch to the ESP32, open the serial monitor (note that this sketch runs at 9600 baud,unlike the others) and press reset. You'll get a string of numbers.

Now bring a magnet near the ESP32 module and observe the numbers. You should notice a definite change when the magnet is brought near the sensor.



For a more visual experience open the Serial Plotter instead of the Serial Monitor. You should notice a graph that responds to the presence or absence of your magnet.

LED Software Fade

Changing the brightness of an LED is a pretty popular task for a microcontroller. With the Arduino we use the *analogWrite* function, which provides a PWM (pulse width modulation) output to PWM-enabled output pins. Varying the duty-cycle of the PWM signal will control the LED brightness.

However *analogWrite* will not work with the ESP32. Instead we will need to use a different method to control LED brightness.

The ESP32 has 16 internal channels to control LED brightness. You can control the precision of the PWM timer and its frequency, allowing you precision control of the LED brightness.

Although you'll find the *LEDCSoftwareFade* sketch in the *AnalogOut* submenu the output is actually digital PWM.

```
/*
  LEDC Software Fade

  This example shows how to software fade LED
  using the ledcWrite function.

  Code adapted from original Arduino Fade example:
  https://www.arduino.cc/en/Tutorial/Fade

  This example code is in the public domain.
  */

// use first channel of 16 channels (started from zero)
```

```
14 #define LEDC_CHANNEL_0     0
15
16 // use 13 bit precission for LEDC timer
17 #define LEDC_TIMER_13_BIT  13
18
19 // use 5000 Hz as a LEDC base frequency
20 #define LEDC_BASE_FREQ     5000
21
22 // fade LED PIN (replace with LED_BUILTIN constant for buil
23 #define LED_PIN            5
24
25 int brightness = 0;    // how bright the LED is
26 int fadeAmount = 5;    // how many points to fade the LED b
27
28 // Arduino like analogWrite
29 // value has to be between 0 and valueMax
30 void ledcAnalogWrite(uint8_t channel, uint32_t value, uint3
31   // calculate duty, 8191 from 2 ^ 13 - 1
32   uint32_t duty = (8191 / valueMax) * min(value, valueMax);
33
34   // write duty to LEDC
35   ledcWrite(channel, duty);
36 }
37
38 void setup() {
39   // Setup timer and attach timer to a led pin
40   ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_B
41   ledcAttachPin(LED_PIN, LEDC_CHANNEL_0);
42 }
43
44 void loop() {
45   // set the brightness on LEDC channel 0
46   ledcAnalogWrite(LEDC_CHANNEL_0, brightness);
47
48   // change the brightness for next time through the loop:
49   brightness = brightness + fadeAmount;
50
51   // reverse the direction of the fading at the ends of the
52   if (brightness <= 0 || brightness >= 255) {
53     fadeAmount = -fadeAmount;
54   }
55   // wait for 30 milliseconds to see the dimming effect
56   delay(30);
57 }
```

The sketch works by cycling the duty-cycle of the PWM output, which causes the LED to fade and brighten and then repeat.

In order to work with our ESP32 hookup you will need to change the LED_PIN constant (on line 23) to a “2” instead of “5”, as our LED is attached to GPIO 2.

Note that the sketch defines a function called *ledcAnalogWrite* that operates in a similar fashion to the Arduino *analogWrite* function. You might want to use that in programs that you create to control LEDs.

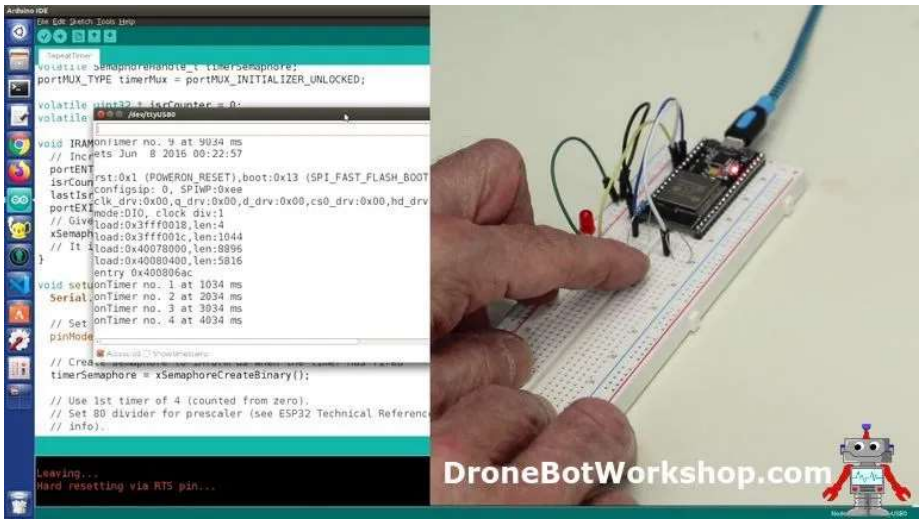
Repeat Timer

The Repeat Timer example is essentially a basic stopwatch. When the Reset (ENABLE) button on the ESP32 module is pressed it starts counting seconds, which are displayed on the serial monitor. The count continues until the push button we attached to our module is pressed. It can be reset to start again.

```
/*
 Repeat timer example

 This example shows how to use hardware timer in ESP32. The
 function every second. The timer can be stopped with butto
 (IO0).

 This example code is in the public domain.
 */

// Stop button is attached to PIN 0 (IO0)
#define BTN_STOP_ALARM    0

hw_timer_t * timer = NULL;
volatile SemaphoreHandle_t timerSemaphore;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

volatile uint32_t isrCounter = 0;
volatile uint32_t lastIsrAt = 0;

void IRAM_ATTR onTimer(){
  // Increment the counter and set the time of ISR
  portENTER_CRITICAL_ISR(&timerMux);
  isrCounter++;
  lastIsrAt = millis();
  portEXIT_CRITICAL_ISR(&timerMux);
  // Give a semaphore that we can check in the loop
  xSemaphoreGiveFromISR(timerSemaphore, NULL);
  // It is safe to use digitalRead/Write here if you want t
}

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

  // Set BTN_STOP_ALARM to input mode
  pinMode(BTN_STOP_ALARM, INPUT);

  // Create semaphore to inform us when the timer has fired
  timerSemaphore = xSemaphoreCreateBinary();

  // Use 1st timer of 4 (counted from zero).
  // Set 80 divider for prescaler (see ESP32 Technical Refe
  // info).
  timer = timerBegin(0, 80, true);

  // Attach onTimer function to our timer.
  timerAttachInterrupt(timer, &onTimer, true);

  // Set alarm to call onTimer function every second (value
  // Repeat the alarm (third parameter)
  timerAlarmWrite(timer, 1000000, true);

  // Start an alarm
  timerAlarmEnable(timer);
```

```
}

void loop() {
  // If Timer has fired
  if (xSemaphoreTake(timerSemaphore, 0) == pdTRUE){
    uint32_t isrCount = 0, isrTime = 0;
    // Read the interrupt count and time
    portENTER_CRITICAL(&timerMux);
    isrCount = isrCounter;
    isrTime = lastIsrAt;
    portEXIT_CRITICAL(&timerMux);
    // Print it
    Serial.print("onTimer no. ");
    Serial.print(isrCount);
    Serial.print(" at ");
    Serial.print(isrTime);
    Serial.println(" ms");
  }
  // If button is pressed
  if (digitalRead(BTN_STOP_ALARM) == LOW) {
    // If timer is still running
    if (timer) {
      // Stop and free timer
      timerEnd(timer);
      timer = NULL;
    }
  }
}
```

The *RepeatTimer* example is in the *Timer* submenu, and it illustrates how to create an interrupt handler for the ESP32. The interrupt handler is a function called onTimer and it is attached to the timer and triggered every second.



88

The button is read in the Loop of the sketch and calls the built-in *timerEnd* function to stop the timer.

This is a great example to build upon.

Touch Read

The final example that we will look at uses the built-in touch switch of the ESP32.

A touch switch is simply a conductive plate that you can touch to activate. This is a capacitive touch switch, so touching it causes the capacitance to change. This change is measured and that change is used to activate the switch.

On my breadboard, I just used a small loop of wire as my “touch switch”. Feel free to get creative and use something else!

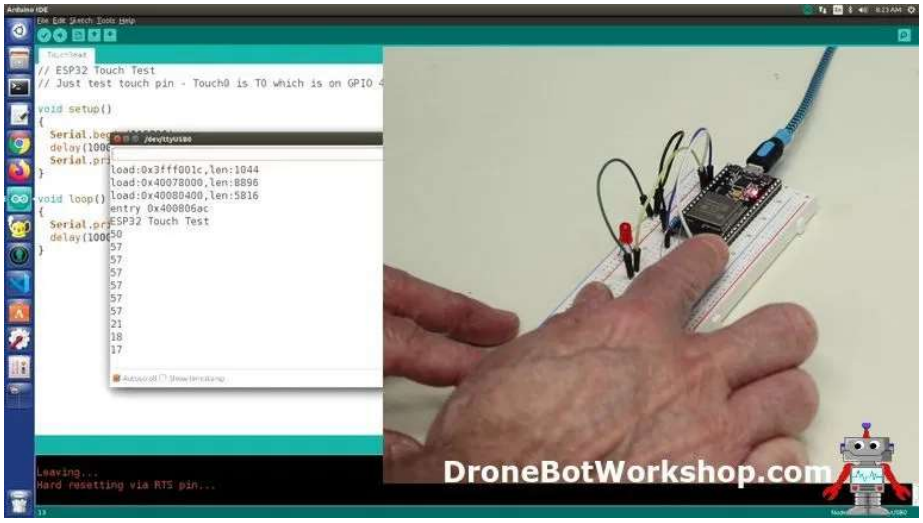
There are ten touch switch inputs on the ESP32, they share their function with ten of the GPIO pins. We are using touch switch 0, which is equivalent to GPIO 4.

```
// ESP32 Touch Test
// Just test touch pin - Touch0 is T0 which is on GPIO 4.

void setup()
{
  Serial.begin(115200);
  delay(1000); // give me time to bring up serial monitor
  Serial.println("ESP32 Touch Test");
}

void loop()
{
  Serial.println(touchRead(T0));  // get value using T0
  delay(1000);
}
```

The *TouchRead* sketch can be found in the *Touch* submenu. As you can see it’s another very simple sketch, making use of the *touchRead* function.

The sketch reads the value outputted from the *touchRead* function and displays it on the serial monitor. It’s as simple as that!

88



Load the sketch to your ESP32, open the serial monitor and reset the device. You should observe a change in value when you touch the “touch switch”.

You can use this example to add touch switches to your next ESP32 project.

Conclusion

As you can see from the number of example sketches provided with the ESP32 this is a very powerful microcontroller with a lot of capabilities.

With its amazing versatility and low price, the ESP32 is going to be making many more appearances here in the DroneBot Workshop. And it should be making an appearance in your own workshop very soon.

Resources

Article Code – The examples used in this article in a downloadable ZIP file.

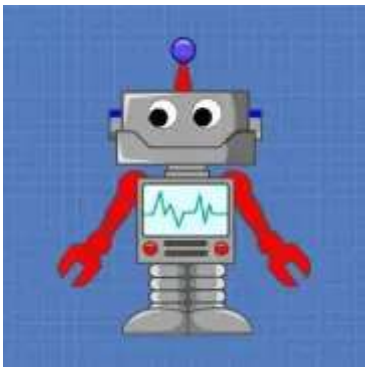
Arduino IDE Board Manager – The Board Manager extension for the ESP32 on GitHub.

Serial Bluetooth Terminal – The Android Serial Bluetooth Terminal app on the Google Play Store.

Bluetooth Terminal – The IOS Bluetooth Terminal all on the Apple App Store.

ESP32 Overview – Espressif Systems ESP32 page.

Summary

Article Name	Getting started with ESP32
Description	Learn to use the ESP32 microcontroller with the Arduino IDE. Today we will examine the many features of this amazing device, including its WiFi and Bluetooth capabilities.
Author	DroneBot Workshop
Publisher Name	DroneBot Workshop
Publisher Logo	

Tagged in: ESP32 Tutorial

DroneBot Workshop April 2, 2020 ESP32, Tutorial

88 Comments

← Peltier Effect Cooling Experiments

Getting started with the ESP32-CAM →

If you have a question...

Comments about this article are encouraged and appreciated. However, due to the large volume of comments that I receive, it may not be possible for me to answer you directly here on the website.

You are much more likely to get answers to technical questions by making a post on the DroneBot Workshop Forum. Your post will be seen not only by myself, but by a large group of tech enthusiasts who can quickly answer your question. You may also add code samples, images and videos to your forum posts.

Having said that, please feel free to leave constructive comments here. Your input is always welcome. Please note that all comments may be held for moderation.

Join the discussion

88 COMMENTS Oldest

A.J. Lenze 2 years ago

At about 15 minutes into your video, you show the process for downloading a sketch into your ESP32 module. As part of the process, you show how to press the IO0 button when you see “Connecting…”

on the Arduino IDE screen. There's a work-around for this, where you connect a 2.2uF capacitor between your EN pin and ground, which prevents you from having to press the button when downloading the sketch. You can read all about this at https://github.com/espressif/esptool/issues/136. Apparently, this is a bug in early ESP32 silicon, which has was later fixed, although there seems to be a lot… Read more »

+ 6 − Reply

Author

DroneBot Workshop 2 years ago

Reply to A.J. Lenze

Awesome A.J., thank you for sharing that! I'll have to check a few of my other ESP32 modules to see if they require the pushbutton.

+ 6 − Reply

David 2 years ago

TYPO
In 'Introducing ESP32', you say:
The ESP32 improved upon the ESP32
I think you mean
The ESP32 improved upon the ESP8266

+ 8 − Reply

David 2 years ago

Thanks for an amazing amount of work you must have put into the video and the write-up.
Having put my early ESP32s on one side after hitting so many problems with them, II have now brought them out and started loading the programs you kindly supplied in a zip (they didn't come up in

my Examples folders). And following your instructions, so far they all work! Brilliant!
Thanks. Now to get down to some serious work with them.

+ 2 − Reply

Teddy Chen 2 years ago

Thank you for your excellent Video. That is great!
For the uploading issue, you can use any capacitor between 2.2 to 10 uF.
BTW I noticed that you use the Menu to open the Serial Monitor. There is a shortcut below the “Close” button.

+ 5 − Reply

Suman Bhat 2 years ago

I do have an ESP32 module. I am thinking to make a clock using addressable LED with animation. My question is, will it be possible to use one core of ESP32 to run the clock mechanism, and another core to control LED animation (so the clock is not waiting in the meantime).

+ 0 − Reply

Martin Egan 2 years ago

excellent video and article looking forward to your next one keep up the good work
regards
Martin

+ 0 − Reply

Peter 2 years ago

Went out and bought one of these straight away (the Adafruit Feather version) Once It arrived I could not get it to work , it powered on but no communication with the Arduino IDE . It took me a while to realise the USB cable was power only 🙁 Once I got a new cable it was up and running) looking forward to using it and creating some animations (servos working etc) on a model railway. PS – I have used Arduino's and Raspberry Pi for years but the WiFi and Bluetooth will be an… Read more »

+ 2 — Reply

Nektarios Kourakis 2 years ago

thanks for tutorial..i suggest to move on with esp32 ..give us more option with this mcu..THANKS

+ 0 — Reply

JonG 2 years ago

Thanks for the very informative video. I am tinkering with an ESP32-DEVKITC-32D-F ($10 from DigiKey) and I ran into an issue using Arduino on my Mac. The drivers for the CP2102 USB to UART bridge on the ESP32 module were not installed on my Mac so I couldn't open a serial port. Apparently this is a common problem and I found this article that provides the steps for downloading the required drivers – https://techexplorations.com/guides/esp32/begin/cp21xxx/. It is a pretty simple download and once installed I was off and running. I just thought it might be helpful for anyone else who runs… Read more »

+ 5 — Reply

Phillip Neal 2 years ago

Great Article on the ESP32. Thank you. A BLE example client-server with another computer (not a phone) would be great !

+ 0 − Reply

Pedro A. Vazquez 2 years ago

+ 0 − Reply

Pedro A. Vazquez 2 years ago

I like that you go improving your site with the many processors that are common for today. You are a great teacher. I continue to think that you can publish a nice book. No not one, many!

+ 1 − Reply

John Anderson 2 years ago

Excellent web page. I don't know much about computers but I have bought an ESP32 board (NodeMCU-32S) to play with for 'lockdown' entertainment and your web site is my reference – when I find something I don't understand on another site I come back to yours and it usually explains it in understandable English. I found compiling on Windows 10 interminably slow, so now have installed Linux on an old laptop and it is a lot better – having finally got Arduino IDE to work on Linux. However whilst working on your set of example programs I get… Read more »

+ 0 − Reply

Ronnie 2 years ago

This is the best video for getting started on esp32 that I have seen. Outstanding in every way. After two years if multiple disappointments I have a esp32 up and blinking. Came here to download the other experimental code sketches to hook up Wifi tonight. And then Bluetooth and beyond. Thanks for such a concise tutorial.

+ 2 − Reply

juan garcia 2 years ago

+ 0 − Reply

Erwin Van Der Haar 2 years ago

Is there a way to readout the content of a programmed ESP32?

+ 0 − Reply

R.H. Campagne 2 years ago

TVM for your excellent workshop video's! Working with the ESP VROOM 32D DEVkit C V4 board, I came across an issue that I could not find in the Espressif documentation: The board must be powered by one of the three Power Supply Options Micro USB port(default), or 5V / GND header pins, or 3V3 / GND header pins I connected the micro USB to power the board, then I – surprisingly, because they should be power input only – measured 5V and 3.3V on the two power input Pins, so the board seems to supply power to the Pins once ... Read more »

+ 0 − Reply

George pederson 2 years ago

would it be possible to build a drone out of a esp32

+ 0 − Reply

Zeki 2 years ago

“Getting started with ESP32″ is perfect. Thank you very much.

+ 0 − Reply

LuigiChi 2 years ago

Very interesting! Thanks!

+ 0 − Reply

Dr. Hritu Raj Rohariya 2 years ago

You are the Best

+ 0 − Reply

TAREK ZAIN 2 years ago

GOOOOD

+ 0 − Reply

briefer 1 year ago

there is a typo with the address 182.168.4.1, should be 192.168.4.1

Last edited 1 year ago by briefer

88

+ 0 − Reply

asd 1 year ago

good

+ 0 − Reply

F Jin 1 year ago

First, thank you for the video and the detailed introduction!
For the *SerialToSerialBT example, I can pair it with Android phones (old or new), but not with any Apple devices (iPhone 7, SE, 11 and iPad). It doesn't show up in my Bluetooth list. Any suggestions?*
Many thanks!

+ 0 − Reply

Ted Thisius 1 year ago

Reply to F Jin

You must use BLE with the Apple devices. I can use Bluetooth with my Android tablet but not with my iPhone or iPad. If I use BLE then I can get it to work with the Android tablet, my iPhone and iPad.

+ 0 − Reply

Cezary 1 year ago

Hello

I am trying t get my esp32 working but still same problem arising . After all steps you have shown in your tutorial etc in my sketch the (SIMPLE ESP32 library) is not highlighted in any colour ,so i assume that I've encountered an issue . And I don't know how to the solve it ?

I've changed arduino software and deleted and removed so many times libraries etc and cant find how to fix it .Any idea could you help me please ?

+ 0 − Reply

Mal 1 year ago

I am reading the Intro to ESP32 tutorial I am confused over the IP address for the AP sketch. Is the entry which says 182.168.4.1 a typo?

+ 0 − Reply

> **Jason** 1 year ago
>
> *Reply to Mal*
>
> 192.168.4.1 worked for me!
>
> + 0 − Reply

Jason 1 year ago

I've been having a hard time using this new to me ESP32 board.. I found your site and wanted to thank you for the great examples and documentation!

+ 0 − Reply

Eugenio Pessoa 1 year ago

Very interesting, simply perfect explanation!!!

+ 0 − Reply

Ted Thisius 1 year ago

Today was my first attempt with the ESP 32. I happened to buy the exact model that Bill shows in the video. I was able to upload the Blink/Hello world example. The onboard and external LEDs blinked as expected but there was gobbly guck on the Serial Monitor. Pushing the reset (EN) didn't change anything. By chance I noticed that my Serial Monitor (Win 10 computer) was still set to 9600 which was the only BAUD rate I had used in the past. I manually set it to the higher BAUD rate and it worked. I did the old test… Read more »

+ 0 − Reply

Ted Thisius 1 year ago

Reply to Ted Thisius

I was able to test the touch switch idea. I just used a piece of hookup wire with the end stripped back. It worked as shown in the video. The value shown on the serial monitor changed as I touched or released the wire. I then licked my finger. Then when I touched the wire the displayed value shown was even more of a change.

+ 0 − Reply

G. M. Oosta 1 year ago

I'd like to suggest a topic for your ESP32 section. I'm trying (so far unsuccessfully) to use an ESP32 with an LPD3806 rotary encoder. I use one of these encoders with one Nano to record position for either altitude or azimuth on my telescope. In examples on the web, I see libraries for multiple encoders and commands for ESP32 that I don't use on the Nano. Eventually, I'd like to combine NEMA17 motor control and position sensing, but have not found enough useful, clear examples to actually make it work. A DroneBot presentation would be helpful!

+ 1 − Reply

Angelo 1 year ago

Wonderful info on the esp32, thanks

+ 0 − Reply

André 1 year ago

Excellent video as usual! So easy to follow you. Your graphics are amazing, not to mention your explanations. A joy to behold…. Many thanks.

+ 0 − Reply

Ted Thisius 1 year ago

I have been trying the suggested example of setting up my ESP-32 as an Access Point. I have been successful in doing so and using my iPhone, iPad and Android tablet to turn ON/OFF the built in LED. It is difficult to touch the correct spot as the font is very small but it did work. Next I tried the suggestion of changing the SSID from the default of yourAP to Tednet. I changed line 21 from *ssid=”yourAP” to *sside=”Tednet”. I compiled this and loaded into onto the board. My devices still showed this as yourAP rather than Tednet. I… Read more »

+ 0 − Reply

Ted Thisius 1 year ago

Reply to Ted Thisius

Later, I tried the suggestion of not requiring a password by leaving password out of line 35. I then tried uploading and was able to use the sketch without a password. I then changed line 21 from yourAP to Bearnet and this time it was successful. I also tried inserting a couple of blank print lines after line 66 so the screen output was a few lines apart. This had no effect but the sketch ran properly. I'm not sure why I was unable to get the SSID to change earlier but it seems fine now.

88

+ 0 − Reply

Mike 1 year ago

This is an excellent intro to the ESP32, which I purchased after watching your video. After installing it, I set up each of the sketches you presented. I did have trouble establishing BT communication despite my best attempts to follow the directions. Also, I was not successful in using the repeat timer sketch. A couple of the sketches were rather advanced, and I think that my difficulty was probably due to my limited programming skillset rather than a flaw in the sketches. I also did not see change in the values in serial monitor or plotter during the Hall sensor… Read more »

+ 0 − Reply

Jason Strutland 1 year ago

Reply to Mike

I had issues with the repeat timer as well, and I traced it to the push button code that checks for LOW and changed it to HIGH (I'm guessing this is due to a different type of push button being used N/C vs N/O). Try changing this line of code:

```
1   if (digitalRead(BTN_STOP_ALARM) == LO
```

to

```
1   if (digitalRead(BTN_STOP_ALARM) == HI
```

+ -1 − Reply

Raagam Parmar 1 year ago

88

I encountered an error while compiling code for ESP32 DOIT Development boardexec: "python": executable file not found in $PATH (source of answer at: https://www.esp32.com/viewtopic.php?t=7616)If you encounter such error, there is a

solution that you can try out (this has worked for me on both of my Ubuntu machines, both of which gave this error) I saw this solution in a Github issue also, but unfortunately I am unable to find that particular page….. Open Terminal Type these commands: sudo apt install python-is-python3 sudo apt-mark hold python2 python2-minimal python2.7 python2.7-minimal libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib pip3 install pyserial 12345 sudo apt install… Read more »

+ 0 − Reply

Phil maser 1 year ago

How do you manage to provide such great videos; What equipment do you use, how does one set up at this level? Do you video editing people.

+ 0 − Reply

Oliver Bailey 1 year ago

You are incorrect about the PWM fade for the LED being digital. If it were digital, the LED could only be in an ON or OFF state. PWM creates a sine wave on the pin using minimum and maximum voltage. The pulse is what causes the LED to fade in an out.

+ -2 − Reply

Larry 1 month ago

Reply to Oliver Bailey

It provides a 0 or 1 for a specific part of each cycle. 50% duty cycle means it is on "1" as long as it is off '0'. 80% means it us a '1' for 80% of the cycle, then off for 20%. A'1′ could be 5 volts, and a '0' zero volts (or some other levels depending on the

logic family you are using). This is a digital output not analog.

+ 0 − Reply

Arno Solo 1 year ago

Very helpful, thanks.

+ 0 − Reply

Larry R Pare 1 year ago

What USB cable (PC to ESP-32S NodeMCU) is required?

+ 0 − Reply

Tom Powderly 1 year ago

Hi, great tutorial. I came here because the esp32 can be used as a hardware step generator for linuxcnc.(see https://github.com/Deotti-cl/linuxcnc-esp32)
Here's nice feature for the Bluetppth terminal…
I used the android text to speech (microphone iconn) to 'talk' to the esp32.
(though I still need to click the 'send' button).
thanks tomp

+ 0 − Reply

shantanu de 1 year ago

Hi,

88

I was doing my initial set-up for blink sketch, and I am receiving the following error.

I am using the ESP32-WROOM-32 board.

avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x8e

+ 0 − Reply

Chris Eaton 1 year ago

Hi, I setup my ESP32 using another persons intro tutorial. They point to this URL for the .json file.

https://dl.espressif.com/dl/package_esp32_index.json' defer onload='

The reason I mention it, is because this is a much shorter URL and it reveals exactly the same .json file information. Hope this helps.

+ 0 − Reply

Greg T 1 year ago

You do incredible work! Outstanding, 6-STAR performances. The most thorough and cogent detailed walkthroughs on this stuff on the Interweb:) Thanks.

+ 0 − Reply

Yuva Raj 1 year ago

Can i use esp32 for wireless cnc machine? what interface required to connect with esp32 to control stepper motor?

+ 0 − Reply

Dré Jansen 1 year ago

hi,
i am just learning the ESP works.

i am a novice beginner, so i do not have any comments yet
who knows, i will later on tell everything i know. later, much later…

+ 0 − Reply

Mayank 1 year ago

I wanted to know if it's possible to send audio to multiple devices at once using the esp32, like synchronizing audio of like 4 bluetooth speakers using the esp32 as the means to send audio from my phone..?

+ 0 − Reply

Chazim Fikri 1 year ago

i haven't found my esp32 board name in arduino IDE. i use TTGO T-Energy with esp32-wrover-B. what board name should I select in arduino IDE?.

+ 0 − Reply

Koji Fukuhara 1 year ago

Your tutorials are very informative and conclusive. I appreciate them!

+ 0 − Reply

Chuck Knox 1 year ago

Bill,
I don't want to be responsible for you thinking prideful thoughts. Even so, I hope you know you are the best teacher on the internet by orders of magnitude. Thank you, for doing this.
Peace,
Chuck Knox

+ 3 − Reply

Load More Comments

Projects

- Arduino Projects
- Drone Projects
- Electronics Projects
- Internet of Things Projects
- Raspberry Pi Projects
- Robot Projects

6-Wheel Rover

- 6-Wheel Rover – Introduction
- 6-Wheel Rover – Rover Base – Not wild about the Wild Thumper Chassis
- 6-Wheel Rover – Providing Power with Voltage Regulators
- 6-Wheel Rover – 11.4-Volt Battery Connections
- 6-Wheel Rover – ESCs and Motor Wiring

What's New?

- Driving DC Motors with Microcontrollers
- Measure Air Quality with Microcontrollers
- Pico W with the Arduino IDE
- Building a Dual-Boot Workstation
- Designing and Building Linear DC Power Supplies
- Using GC9A01 Round LCD Modules
- Sound with ESP32 – I2S Protocol
- Using Arduino Interrupts – Hardware, Pin Change and Timer
- Using the ESP32-CAM MicroSD Card

What's Popular

- Getting started with ESP32
- Getting started with the ESP32-CAM
- Using Servo Motors with ESP32
- Sound with ESP32 - I2S Protocol
- ESP32 WiFiManager - Easy WiFi Provisioning
- Controlling DC Motors with the L298N Dual H-Bridge and an Arduino

Workshop Connections

88

88