

IoT and Touch-Based Home Automation

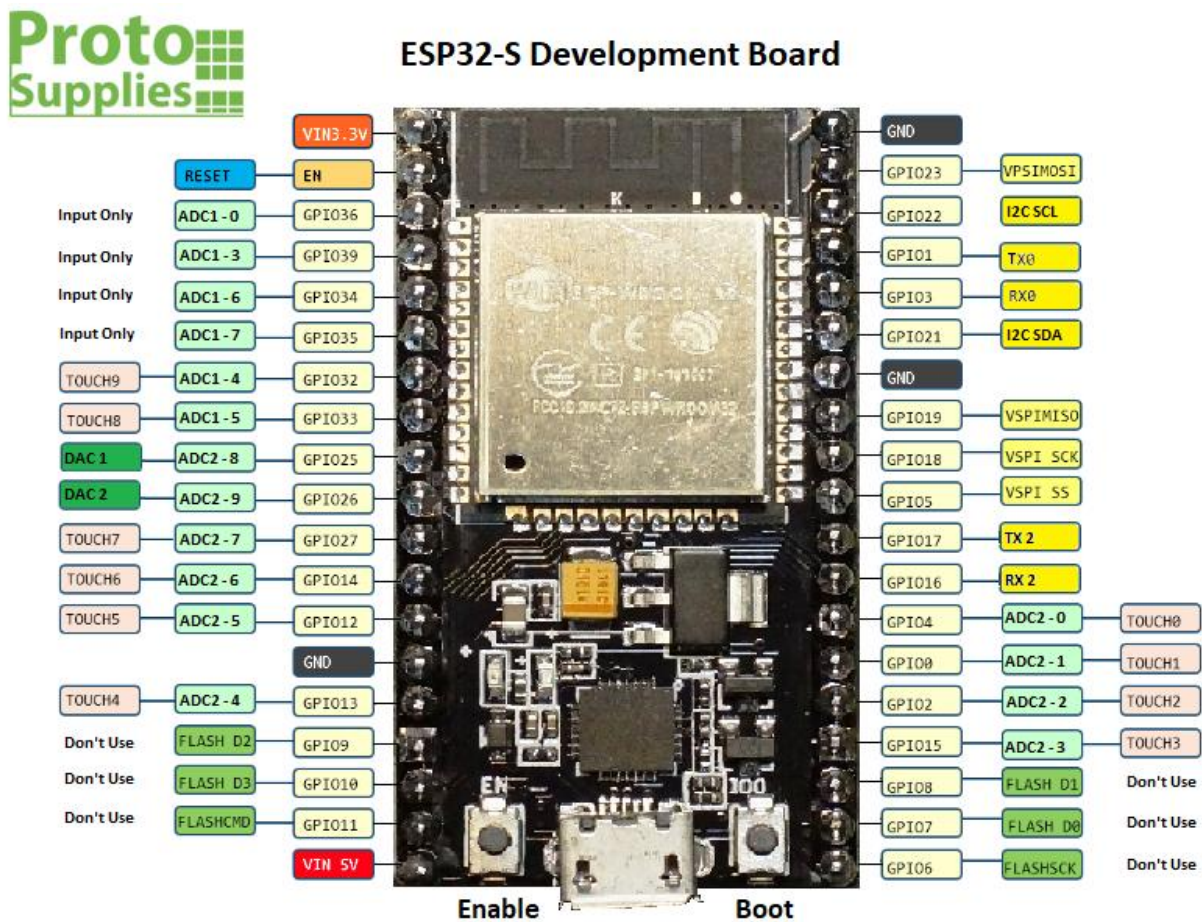
Ianoș Ștefan-Cristian, Grupa 6304

We have seen various applications of IoT but what about adding the touch to it. In this project, we will add simple touch buttons to the ESP-32 Wi-Fi module. ESP-32 is a great module to design IoT applications and adding touch to it will make it further smart. Talking about ESP-32, it is a micro-controller designed by Espressif mainly for IoT applications. It is so handy that even a novice can use it. ESP-32 contains Wi-Fi, Bluetooth, Inbuilt Touch sensing input pins, temperature and hall sensors on board which makes it fit for IoT and Smart home.



Let's get more to Touch. In ESP-32, there are total 10 Touch Sensing general purpose Input Output (GPIO) pins. A touch-sensor system is built on a substrate which carries electrodes and relevant connections under a protective flat surface. When a user touches the surface, the capacitance variation is triggered, and a binary signal is generated to indicate whether the touch is valid.

ESP32 can provide up to 10 capacitive touch pads / GPIOs. The sensing pads can be arranged in different combinations (e.g. matrix, slider) so that a larger area or more points can be detected. The touchpad sensing process is under the control of a hardware-implemented finite-state machine (FSM) which is initiated by software or a dedicated hardware timer. We will learn how to handle these touch pins and try to make an IoT application around it. We will also integrate Wi-Fi control to it.



Material to get started with IoT and Touch Based Home Automation

The following is the list of components used for Touch based home automation system:

1. ESP32 NodeMCU (Check the datasheet from the Internet, if you are using a different version.)



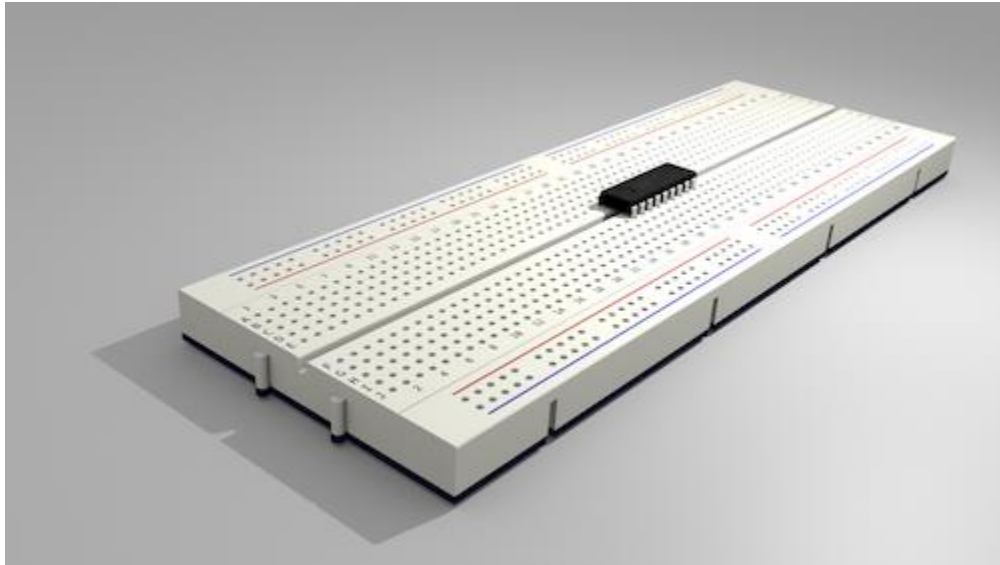
2. USB Type C cable to program ESP32 from a laptop or PC—most Android phones use this type of cable.



3. LED with Resistor(1K) – To test the touch



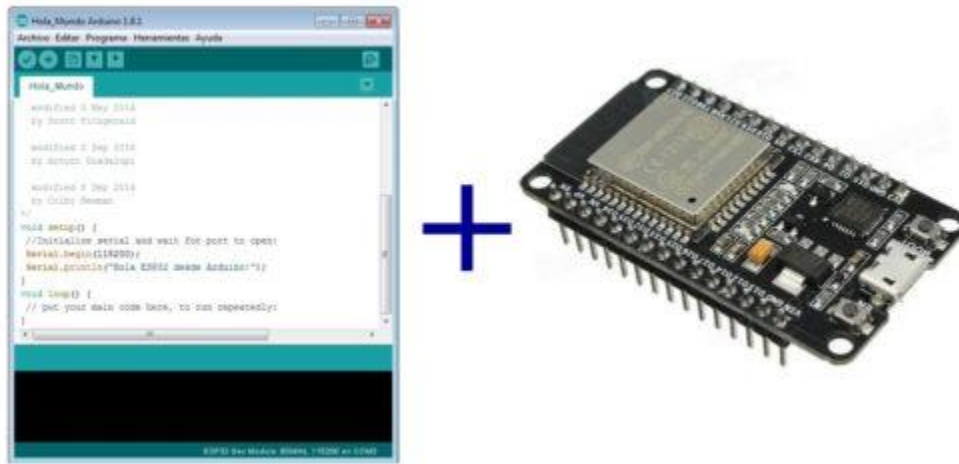
4. Breadboard – To place the components



5. Any metal plate to sense touch. You can even use aluminium foil by connecting a wire to it.



Steps for the software setup:



Here is the code for the ESP32: we need an Integrated Development Environment and we will use Arduino IDE software. Arduino IDE is a cross-platform application. It is written in Java and coded in C/C++ with some special rules. To download the latest Arduino IDE from here.

Arduino IDE does not contain support of ESP32 family so to install the ESP-32 Board in Arduino IDE, you can refer here.

The Code for Touch Based Home Automation System

```
#include <WiFi.h>
const char* ssid      = "xxxx";           // Replace with your network credentials within the double quotes
const char* password = "xxxx";
WiFiServer server(80);                   //Web server port is set to 80

String header;                           // Variable to store the HTTP request
bool state=true;
String output5State = "off";             // Auxiliar variables to store the current output state
const int output5 = 5;                   // Assign an Output variable to declare GPIO pins
int s1=0;
void setup()
{
  Serial.begin(115200);
```

The library contains all the Wi-Fi functions used in the code.

You must replace your Wi-Fi credentials here within the double quotes.

```
const char* ssid = "xxxx";  
const char* password = "xxxx";
```

and make global declarations here.

In the `Void Setup()` here

```
void setup()  
{  
  Serial.begin(115200);  
  pinMode(output5, OUTPUT);           // Initialize the output variables as output  
  digitalWrite(output5, LOW);        // Set output pin to LOW  
  Serial.print("Connecting to ");    // Connect to Wi-Fi network with SSID and password  
  Serial.println(ssid);  
  WiFi.begin(ssid, password);  
  while (WiFi.status() != WL_CONNECTED)  
  {  
    delay(500);  
    Serial.print(".");  
  }  
  Serial.println("");                // Print local IP address and start web server to Serial Monitor  
  Serial.println("WiFi connected.");  
  Serial.println("IP address: ");  
  Serial.println(WiFi.localIP());  
  server.begin();  
}
```

We will set the Baud Rate at 115200(default speed), set outputs and initialize the Wi-Fi to connect it only one. All the code we are placing in `Void Setup()` runs only once after every reset.

In the `void loop()`, we place our main code that needs to run repeatedly.

```
void loop()
{
  WiFiClient client = server.available(); // Listen for any incoming client
  s1=touchRead(T0); // Read the touch GPIO state and save it to variable s1
  if(s1<30) // then check the LED state to toggle LED
  {
    state=!state; // Toggle LED state
    Serial.print("State: ");
    Serial.println(state);
    delay(500);
    digitalWrite(output5,state); // Write the state to the LED
  }
  if (client)
  {
    Serial.println("New Client."); // If a new client connects,
    String currentLine = ""; // print the message in the serial port
    while (client.connected()) // make a String to hold incoming data from the client
    { // loop while the client's connected
      if (client.available()) // if there's bytes to read from the client,
      {
        char c = client.read(); // read a byte, then
        Serial.write(c); // print it out the serial monitor
        header += c;
        if (c == '\n') // if the byte is a newline character
        {
          if (currentLine.length() == 0) // if the current line is blank, you got two newline characters in a row.
          { // that's the end of the client HTTP request, so send a response:
            client.println("HTTP/1.1 200 OK"); // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            client.println("Content-type:text/html"); // and a content-type so the client knows what's coming, then a blank line:
          }
        }
      }
    }
  }
}
```

Done Saving.

We can directly read the touch GPIOs using `touchRead()` function. We can save it to any variable and here we have saved it in the `s1` variable.

Our aim is to control LED with both Touch and Wi-Fi and hence we will merge the functions in the `Void loop()`. An HTML page is made using the HTML script in the code here.

```
digitalWrite(output5, state);
if(state==true)
{
  Serial.println("GPIO 5 on");
  output5State = "on";
}
else if(state==false)
{
  Serial.println("GPIO 5 off");
  output5State = "off";
}
}
client.println("<!DOCTYPE html><html>"); // Display the HTML web page
client.println("<head><meta name='viewport' content='width=device-width, initial-scale=1'>");
client.println("<link rel='icon' href='data:;'>");
client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;};"); // CSS to style the on/off buttons
client.println(".button { background-color: #195B6A; border: none; color: white; padding: 16px 40px;");
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;});");
client.println(".button2 {background-color: #77878A;}</style></head>");
client.println("<body><h1>ESP32 with Touch</h1>"); // Web Page Heading
client.println("<p>Light is " + output5State + "</p>"); // Display current state, and ON/OFF buttons for GPIO 5
if (output5State=="off") // If the output5State is off, it displays the ON button
{
  client.println("<p><a href='/'><button class='button'>ON</button></a></p>");
}
else
{
  client.println("<p><a href='/'><button class='button button2'>OFF</button></a></p>");
}
}
```

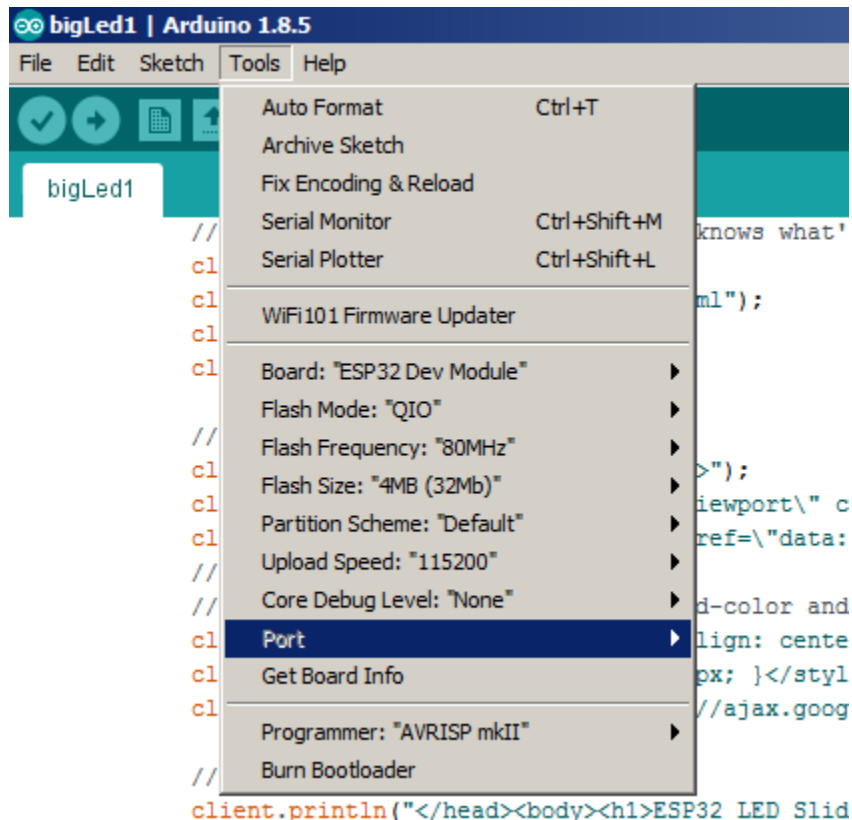

You may even change this as per your application. You will see something like this in your web browser.

ESP32 with Touch

Light is On

ON

Upload this code to the ESP-32 and do remember to select ESP-32 DEV Module and COM Port from Tools menu before uploading the code to the board.



Connections:

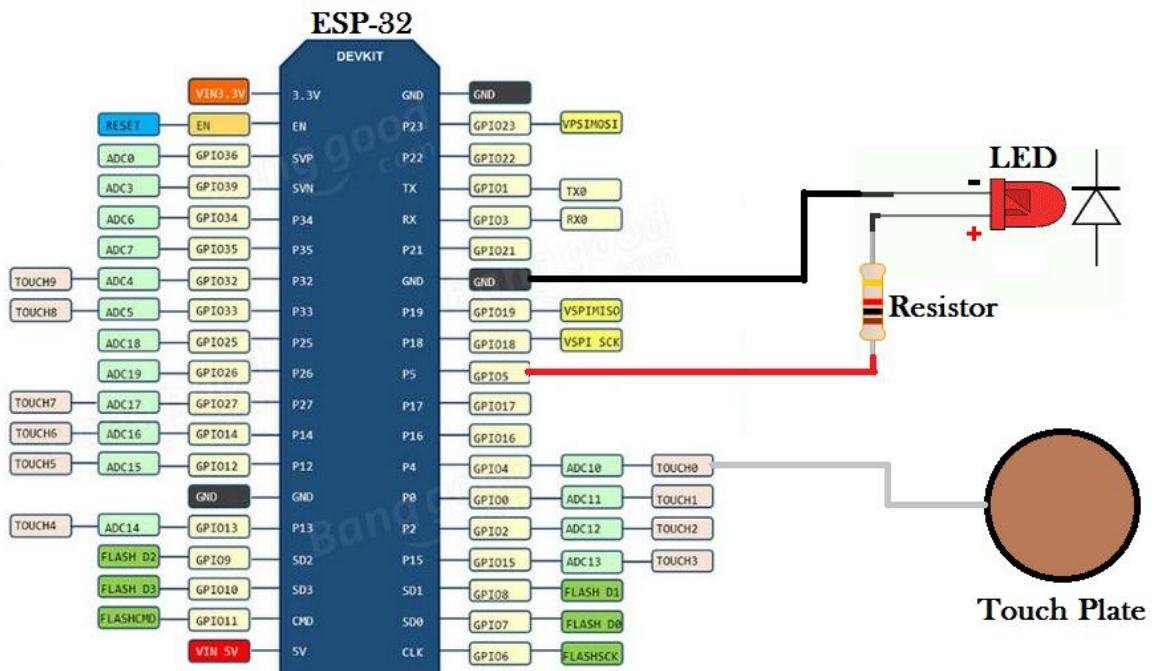
There are only one Input (Touch plate) and one Output (LED) in the circuit.

ESP-32 Pin 5 -> Resistor

ESP-32 Pin 4 -> Touch Plate(any aluminium foil or metal piece would work)

Resistor -> LED +ve

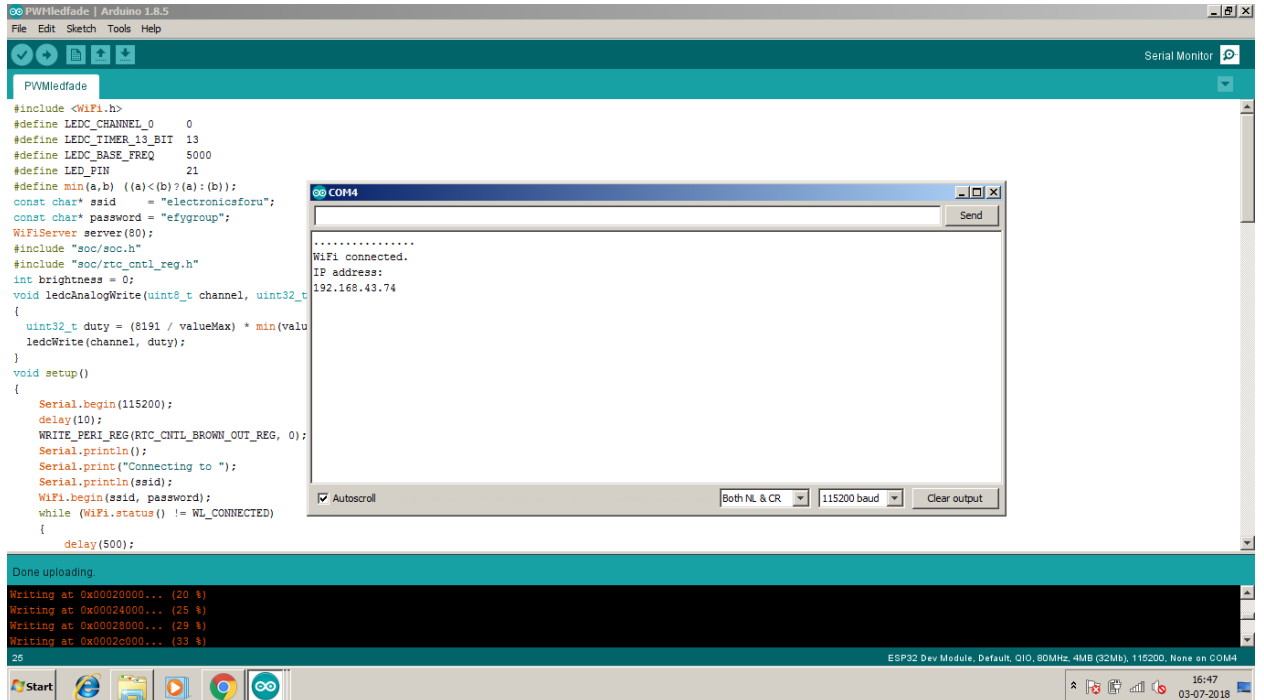
LED -ve -> Ground



Now, power up ESP-32 with USB or a 5Volts supply and let the magic happen.

Connecting the Web server

After uploading the code, go to Open Tools>Serial Monitor. ESP32 will try to connect to Wi-Fi and display its IP address on Arduino serial monitor.



```
PWMledfade
#include <WiFi.h>
#define LEDC_CHANNEL_0 0
#define LEDC_TIMER_13_BIT 13
#define LEDC_BASE_FREQ 5000
#define LED_PIN 21
#define min(a,b) ((a)<(b)?(a):(b));
const char* ssid = "electronicsforu";
const char* password = "efygroup";
WiFiServer server(80);
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
int brightness = 0;
void ledcAnalogWrite(uint8_t channel, uint32_t
{
  uint32_t duty = (8191 / valueMax) * min(valu
  ledcWrite(channel, duty);
}
void setup()
{
  Serial.begin(115200);
  delay(10);
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);

```

Done uploading.

```
Writing at 0x00020000... (20 %)
Writing at 0x00024000... (25 %)
Writing at 0x00028000... (29 %)
Writing at 0x0002c000... (33 %)
25
```

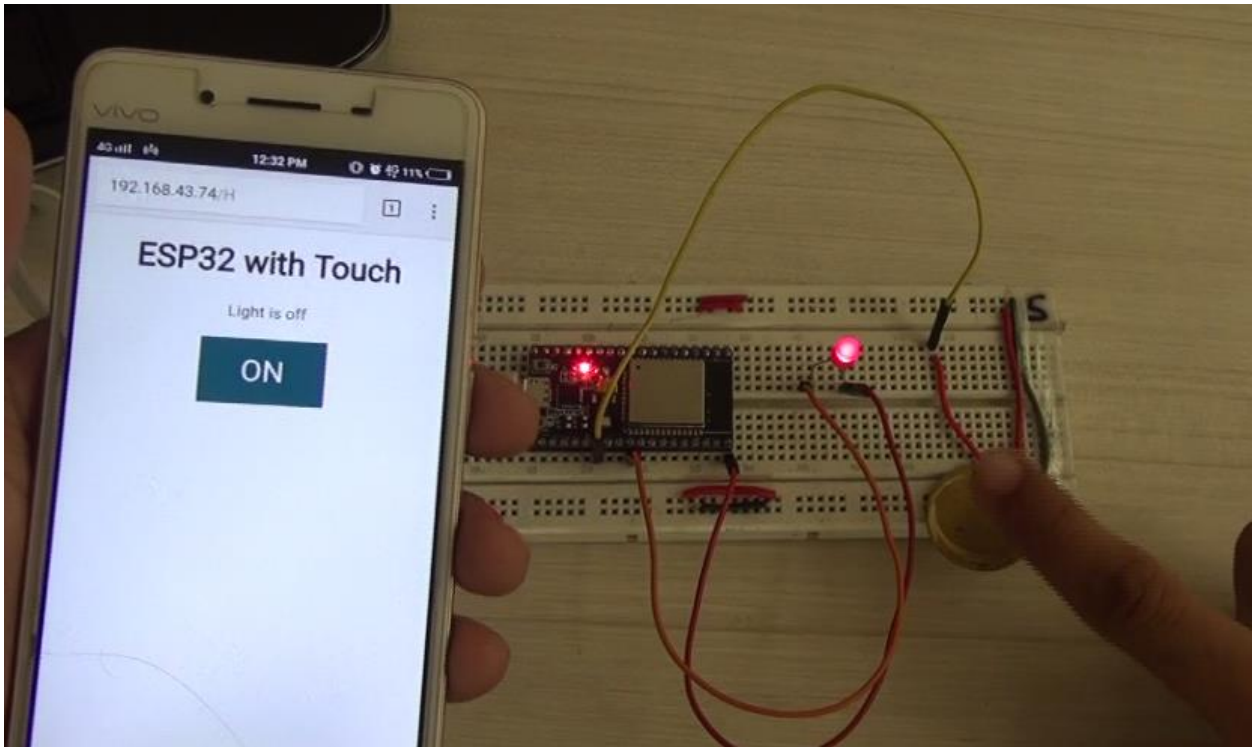
ESP32 Dev Module, Default, Q10, 80MHz, 4MB (32MB), 115200, None on COM4

16:47
03-07-2018

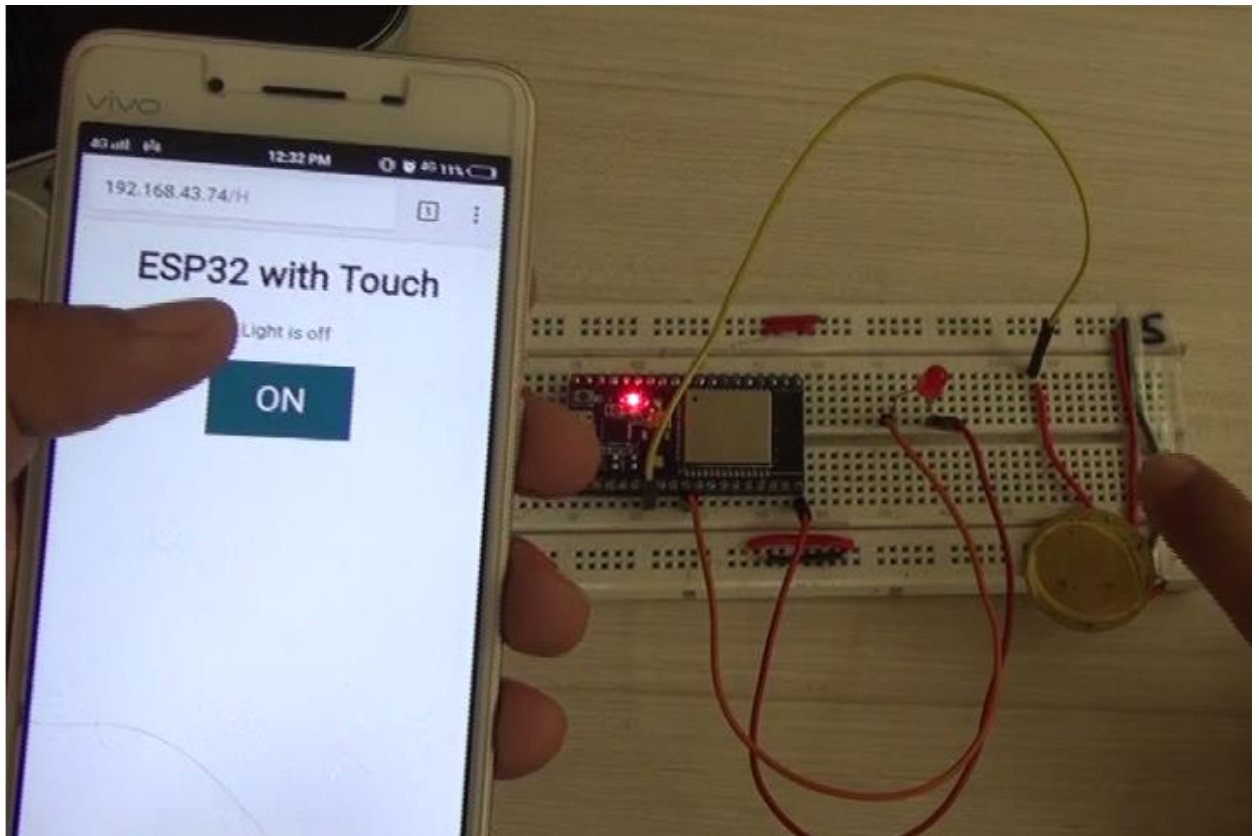
Make sure that the Wi-Fi router to be connected is already open. Hit this IP address in the browser of the device connected to the same Wi-Fi.

Url: <http://192.168.xx.xx> (your IP displayed in Arduino serial monitor)

You will be able to see the HTML Web page mentioned in the code. Now, you can connect and test everything.

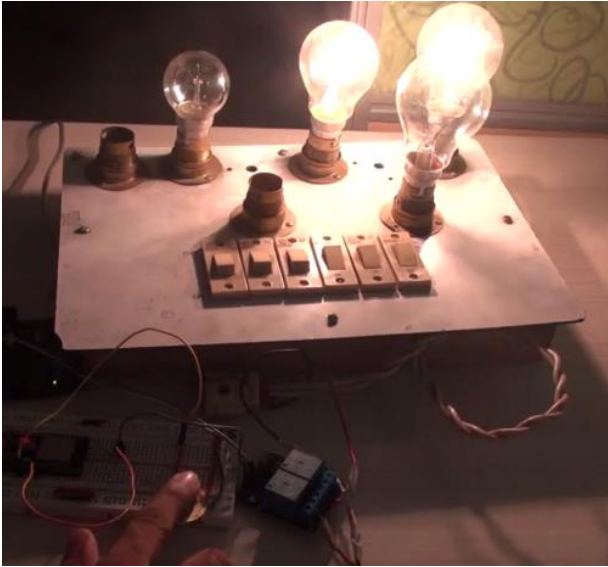


Led Demo 1

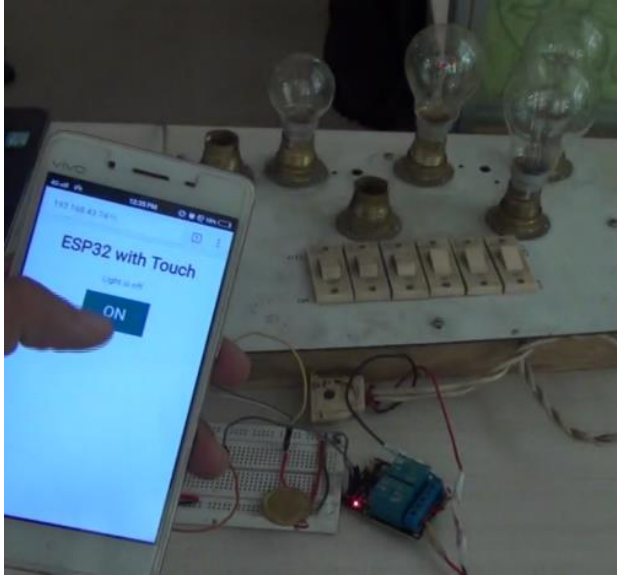


Led Demo 2

Further, you can also connect a relay instead of an LED. Try this out and have the touch fun.



Demo 1



Demo 2