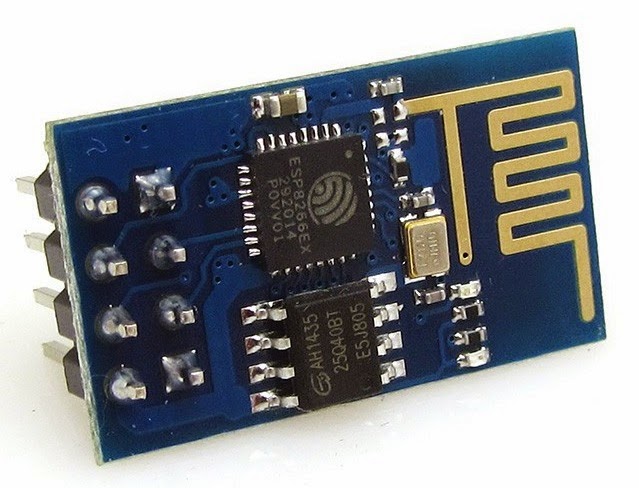
# Introduction to Nodemcu using Arduino Platform

**Introduction to ESP8266 WiFi Module**

**What is the ESP8266?**

****

ESP8266 is a WiFi SOC (system on a chip) produced by [Espressif Systems](http://espressif.com/). It is a highly integrated chip designed to provide full internet connectivity in a small package.

ESP8266 is an impressive, low cost WiFi module suitable for adding WiFi functionality to an

Existing microcontroller project via a UART serial connection. The module can even be reprogrammed to act as a standalone WiFi connected device–just add power!

ESP8266 is a complete and self-contained Wi-Fi network solutions that can carry software applications, or through another application processor uninstall all Wi-Fi networking capabilities. ESP8266 when the device is mounted and as the only application of the application processor, the flash memory can be started directly from an external Move. Built-in cache memory will help improve system performance and reduce memory requirements. Another situation is when wireless Internet access assume the task of Wi-Fi adapter, you can add it to any microcontroller-based design, the connection is simple, just by SPI / SDIO interface or central processor AHB ([Advanced High-performance Bus](https://en.wikipedia.org/wiki/Advanced_Microcontroller_Bus_Architecture#Advanced_High-performance_Bus_.28AHB.29)) bridge interface. Processing and storage capacity on ESP8266 powerful piece, it can be integrated via GPIO ports sensors and other applications specific equipment to achieve the lowest early in the development and operation of at least occupy system resources. The ESP8266 highly integrated chip, including antenna switch balun, power management converter, so with minimal external circuitry, and includes front-end module, including the entire solution designed to minimize the space occupied by PCB. The system is equipped with ESP8266 manifested leading features are: energy saving VoIP quickly switch between the sleep / wake patterns, with low-power operation adaptive radio bias, front-end signal processing functions, troubleshooting and radio systems coexist characteristics eliminate cellular / Bluetooth / DDR / LVDS / LCD interference.

## What is it good for?

ESP8266 can be used as an external WiFi module, using the standard AT Command set Firmware by connecting it to any microcontroller using the serial UART, or directly serve as a WiFi-enabled micro controller, by programming a new firmware using the provided SDK.

The GPIO pins allow Analog and Digital IO, plus PWM, SPI, I2C, etc.

This board has been around for almost a year now, and has been used mostly in IoT contexts, where we want to add connectivity for example to an Arduino project. A wide adoption has been facilitated by the very modest price, ranging from 2.50 to 10 USD depending on the features offered by the manufacturers.

## Technical Features

## 802.11 b / g / n

## Wi-Fi Direct (P2P), soft-AP

## Built-in TCP / IP protocol stack

## Built-in TR switch, balun, LNA, power amplifier and matching network

## Built-in PLL, voltage regulator and power management components

## 802.11b mode + 19.5dBm output power

## Built-in temperature sensor

## Support antenna diversity

## Off leakage current is less than 10uA

## Built-in low-power 32-bit CPU: can double as an application processor

## SDIO 2.0, SPI, UART

## STBC, 1×1 MIMO, 2×1 MIMO

## A-MPDU, A-MSDU aggregation and the 0.4 within wake

## 2ms, connect and transfer data packets

## Standby power consumption of less than 1.0mW

## Architecture

Espressif’s ESP8266EX delivers a highly integrated Wi-Fi SoC solution to meet the continuous demand for efficient power usage, compact design and reliable performance in the industry. With its complete and self-contained Wi-Fi networking capabilities, ESP8266EX can perform either as a standalone application, or as a slave to a host MCU. When ESP8266EX hosts an application, it promptly boots up from the external flash. The integrated highspeed cache optimizes the system's performance and memory. Also, ESP8266EX can be applied to any micro-controller design as a Wi-Fi adaptor through SPI/SDIO or I2C/UART interfaces. Besides the Wi-Fi functionalities, ESP8266EX also integrates an enhanced version of Tensilica’s L106 Diamond series 32-bit processor and on-chip SRAM. It can be interfaced with external sensors and other devices through the GPIOs, resulting in low development cost at early stage and minimum footprint. Software Development Kit (SDK) provides sample codes for various applications. ESP8266EX integrates antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters and power management modules. The compact design minimizes the PCB size and the external circuitry. ESP8266EX enables sophisticated features, such as: • Fast switching between sleep and wake-up modes for efficient energy use; • Adaptive radio biasing for low-power operation; • Advanced signal processing; • Spur cancellation; • Radio co-existence mechanisms for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

## Related image

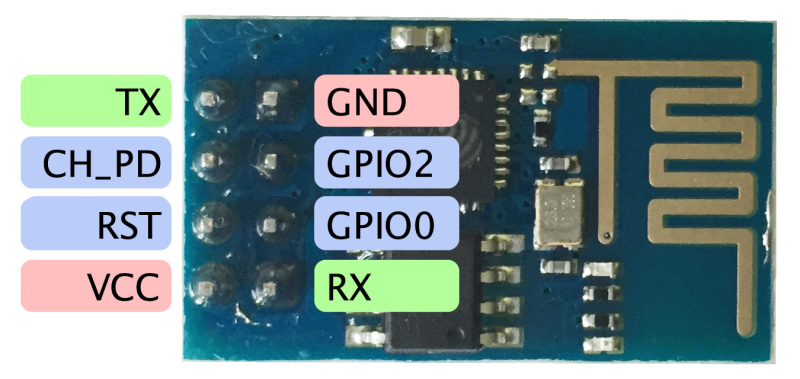
## Variants

## ESP8266-based boards are available from several vendors and with different breakout boards. Most of the boards differ in the number of pins made available, the amount of Flash memory for storing program and data, and the shielding on the SOC for certified boards.

## Some boards also support an external uFL antenna connector as well as the build-in on-chip antenna.

## The most common variants are:

## ESP8266-01

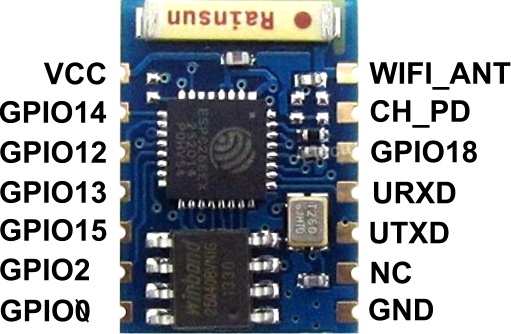
****

This is the first and simplest board using the ESP8266. It allows to attach serial lines, and only breaks out two GPIO pins for native usage.

Pin out details

* Vcc = 3.3v
* GND
* TX = serial TX
* RX = serial RX
* RST = reset
* CH\_PD=FLASH reset (3.3v)
* GPIO0,GPIO1=I/O pins

### ESP-03



This is the second generation board, breaking out more GPIO pins, and using a different antenna, plus an external antenna connector.

**NodeMCU**

NodeMCU is an open source [IoT](https://en.wikipedia.org/wiki/Internet_of_Things) platform. It includes firmware which runs on the [ESP8266](https://en.wikipedia.org/wiki/ESP8266) [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) [SoC](https://en.wikipedia.org/wiki/System_on_a_chip) from [Espressif Systems](https://en.wikipedia.org/w/index.php?title=Espressif_Systems&action=edit&redlink=1), and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the dev. kits. It combined features of WIFI accesspoint and station + microcontroller and uses simple [LUA](http://www.lua.org/) based programming language.

The ESP8266 is made by a privately held company in China called [Espressif](http://espressif.com/).   They are a fabless semiconductor company that just came out of nowhere and shook up the whole industry.   Now all the major players are working on inexpensive versions of an IOT chip with WiFi connectivity.  And they are all struggling to make it as inexpensive as the ESP8266.

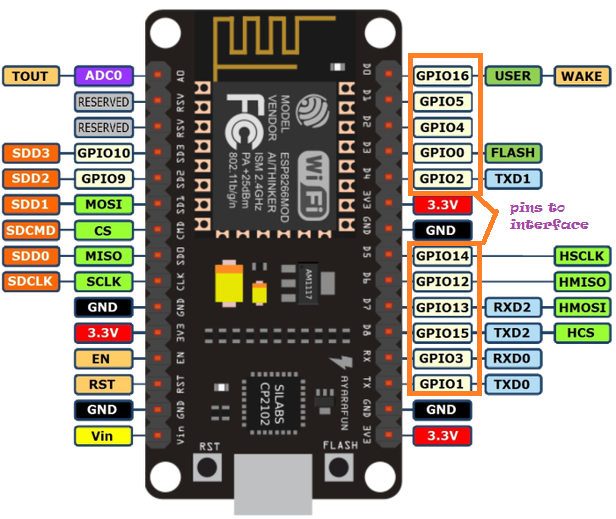
The ESP8266 chip was originally designed for connected lightbulbs (with functionality like the Phillips Hue we used in the [iBeacon BeaconAir project](http://www.switchdoc.com/2015/01/beaconair-reading-ibeacons-raspberry-pi/)), but soon was used in a variety of applications.  While the ESP8266 has huge functionality and a good price, the amount of current consumed by the chip makes battery powered solutions problematic, but with very clever programming, possible in some applications.

****

Here are the main features of the chip.

* SDIO 2.0, SPI, UART, I2C
* Integrated RISC processor, on-chip memory and external memory interfaces
* Integrated MAC/baseband processors
* I2S interface for high fidelity audio applications
* Fully integrated WiFi solution
* Open-source
* Interactive
* Programmable
* Low cost
* USB-TTL included
* Plug & Play
* 10 GPIO, every GPIO can be PWM, I2C, 1-wire
* PCB antenna

Recently, there has been interest in programming ESP8266 systems using Arduino IDE. Programming, of ESP8266 using Arduino IDE is not very straight forward, until it is properly configured. Especially because, the Input and output pins have different mapping on NodeMCU than those on actual ESP8266 chip.

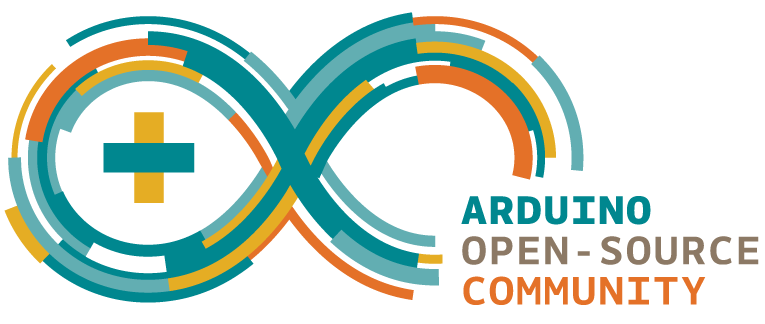


The GPIO pins are the pins to interface. Here the pin layout is different. When you need to connect pin D2 you need to program for pin as 4.

Getting Started with NodeMCU using Arduino Platform.

The Arduino Integrated Development Environment - or Arduino Software (IDE). The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.



**Installation of Arduino IDE Software**

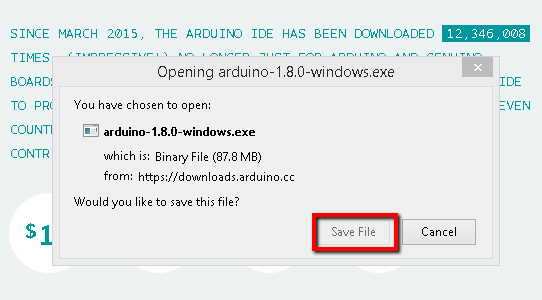
1. Installation steps for “ Arduino IDE”

Download the Arduino IDE Software from Google [Arduino IDE](https://www.arduino.cc/en/Main/Donate).

Click on just download.

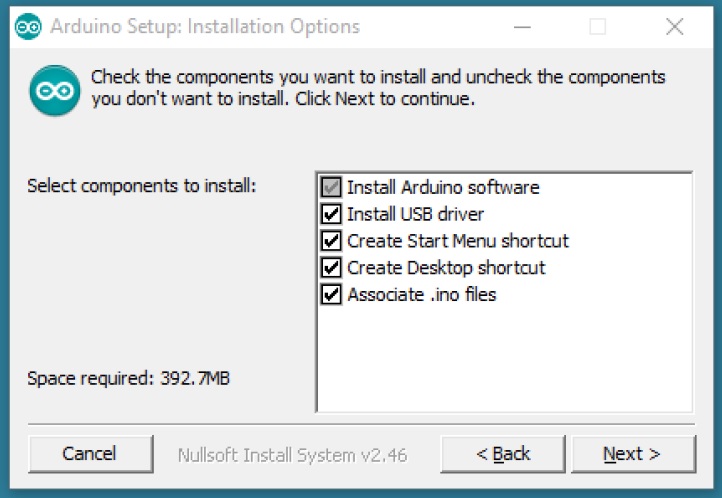


Will get a pop-up asking for saving the file, click on save file. It will start downloading.

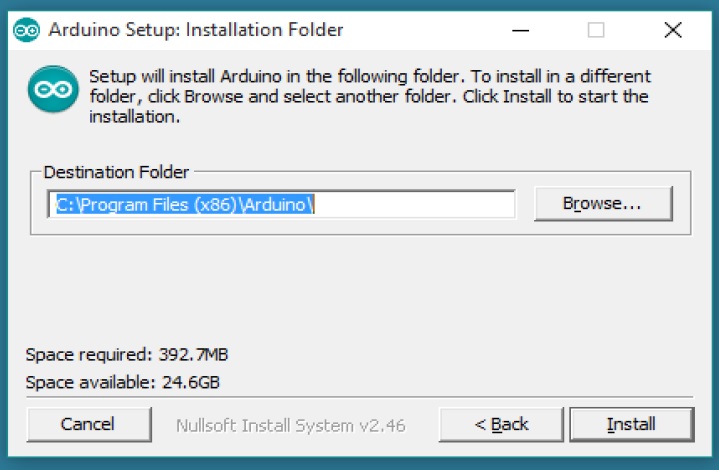


After the download finishes, proceed with the installation and you get a confirmation pop-up window with Yes and No buttons and select Yes button. Please allow the driver install.

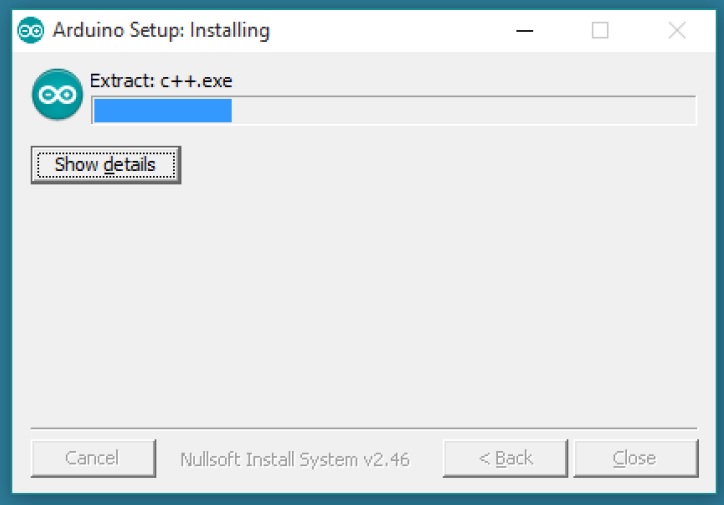
Choose the components to install and Click on next button as shown below.



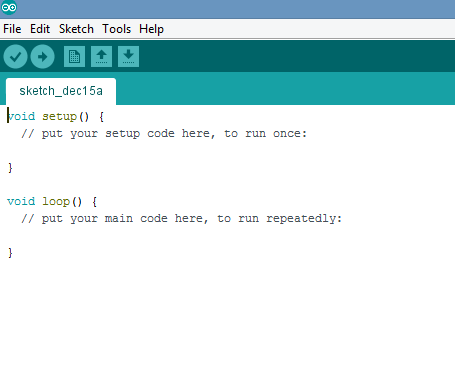
Choose the drive or directory where the software to be installed and click on install.



The process will extract and install all the required files to execute the Arduino Software.



Once the installation is completed, click on Arduino icon (we can see in Desktop), it will open the Arduino IDE as shown below.

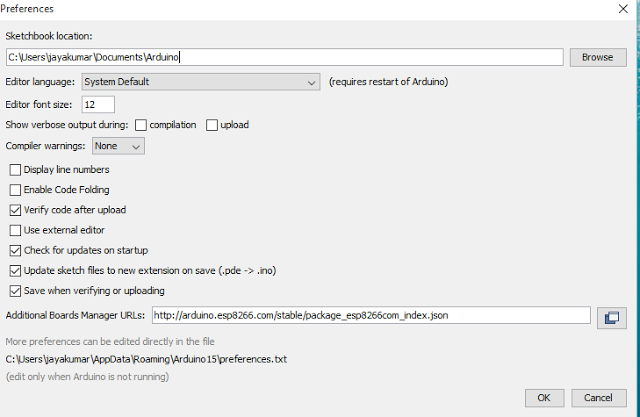


**Adding board packages Arduino ide**

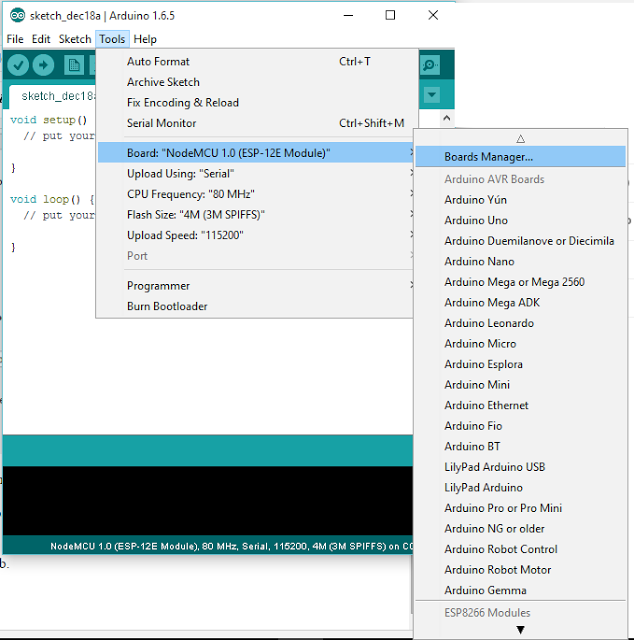
Let’s begin with installing the Esp8266 support for the Arduino and see how to blink an LED.

Firstly open the Arduino IDE  
Go to files and click on the preference in the Arduino IDE.

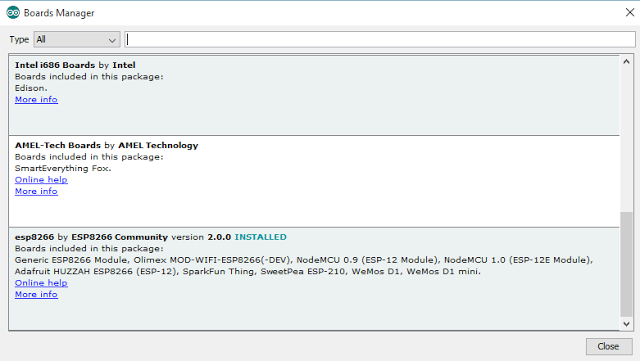
1. Copy the below code in the Additional boards Manager  
   Http://arduino.esp8266.com/stable/package\_esp8266com\_index.json  
   click OK to close the preference Tab.

[](http://1.bp.blogspot.com/-OV1A_EMzm00/VnPsxvqxr_I/AAAAAAAAAuI/qlswMsf3Reo/s1600/preference.PNG)

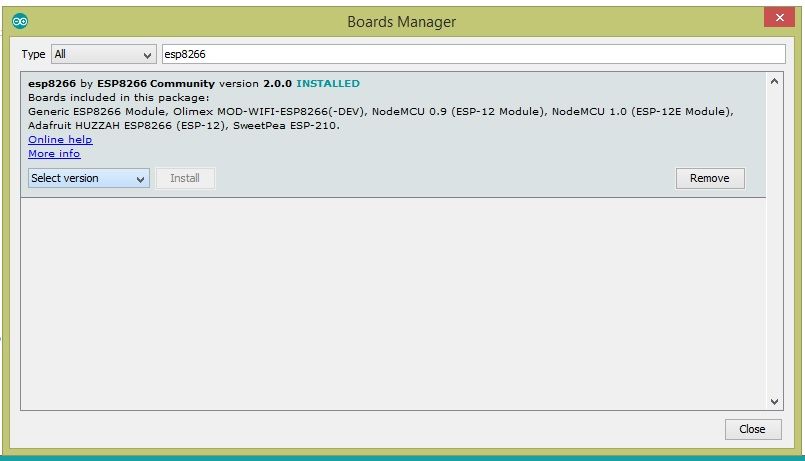
1. After completing the above steps , go to Tools and board, and then select board Manager

[[](http://1.bp.blogspot.com/-u-opNqS4BiI/VnPuR5Pc5uI/AAAAAAAAAuU/wGg5PkYGE3M/s1600/board+manager.png)](http://1.bp.blogspot.com/-u-opNqS4BiI/VnPuR5Pc5uI/AAAAAAAAAuU/wGg5PkYGE3M/s1600/board%2Bmanager.png)

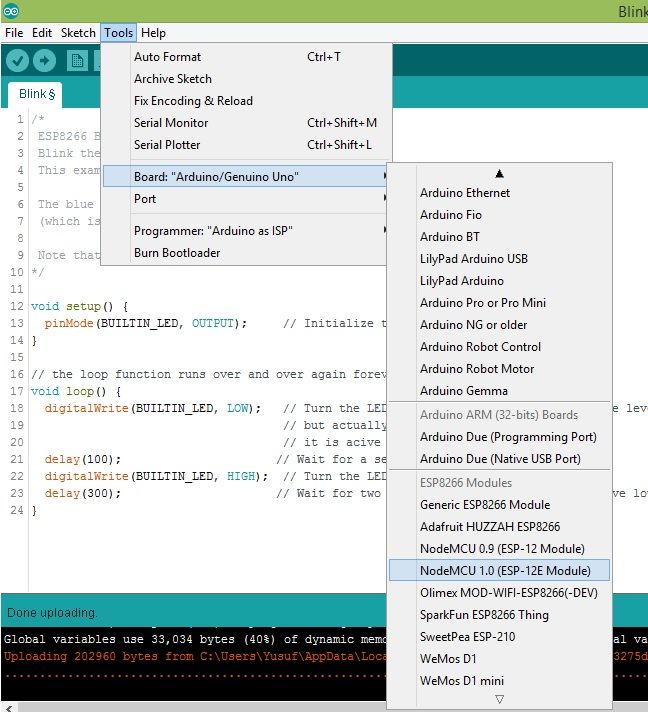
1. Navigate to esp8266 by esp8266 community and install the software for Arduino.

[](http://1.bp.blogspot.com/-njmbyb_yr_U/VnPuv7WAjRI/AAAAAAAAAus/nj0gR7SyFuE/s1600/board+manager+open.PNG)

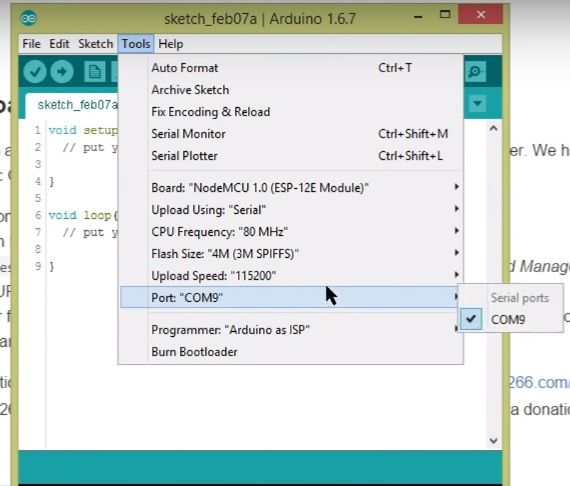
1. Click on install after selecting the recent version.



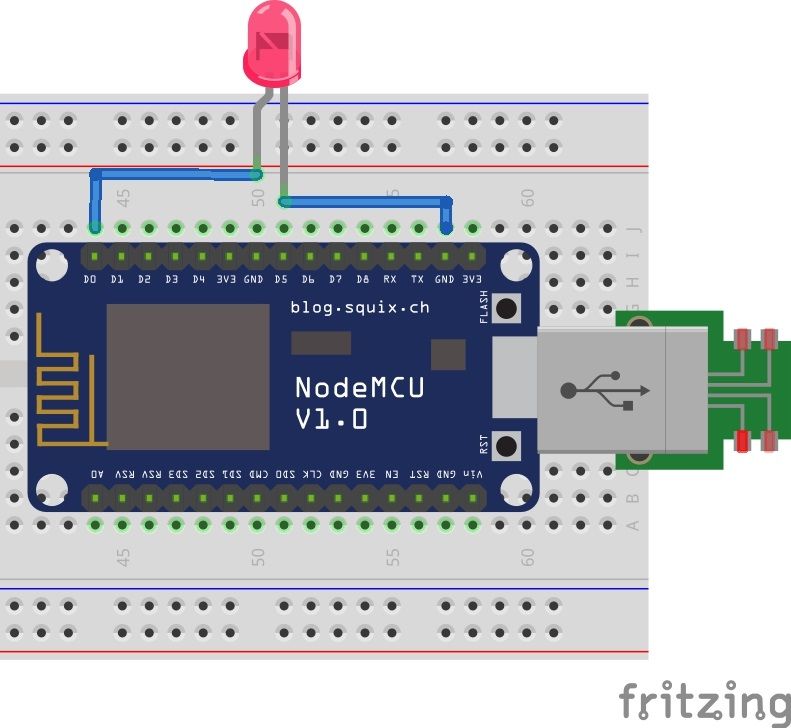
1. Selecting the NodeMCU Board in Arduino IDE



1. Select the com port



1. LED Blink connection for External LED



1. Led BLINK- Example Sketch



**Basic Arduino Programming Keywords**

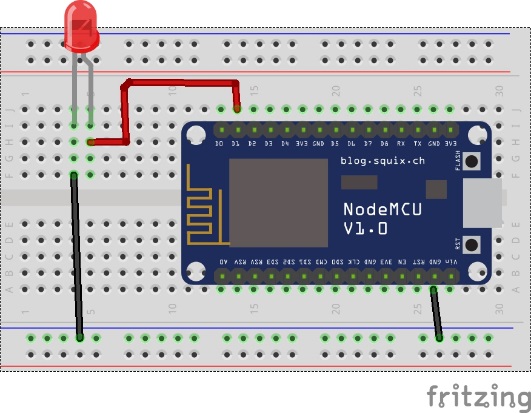
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Keywords** | | **Syntax** | **Example** | **Description** |
| **Data**  **types** | **int**  **float**  **String**  **boolean**  **char**  **byte**  **array** | **int** variable\_name;  **float** variable\_name;  **string** variable\_name;  **boolean** variable\_name;  **char** variable\_name;  **byte** variable\_name;  **array** variable\_name[size]; | **int** m; // int m=8  **float** p; // float p=5.035  **string** n; // n="college"  **boolean** b;// b=1 or 0  **char** p; //p='x'  **byte** r;  **int array** a[10];// 'a' can store 10 elements from 0 to 9 | To define integer numbers. (16 bits in size)  to define floating point numbers(32 bits )  to define array of characters  defines simple logical true/false(high or low)  To define characters (signed or unsigned)  unsigned number from 0-255  collection of identical elements of same type |
| **pinMode()** | | **pinMode**(Var1,OUTPUT/INPUT) | **Pinmode**(ledpin ,OUTPUT);  **Pinmode**(13, INPUT); | Configures the specified pin to behave either as an input or output |
| **digitalRead()** | | **digitalRead**(pin); | **Int** v1=12;  Val= **digitalRead**(v1); | Reads the value from a specified digital pin , either HIGH or LOW |
| **digitalWrite()** | | **digitalWrite**(pin, value); | **digitalWrite**(13, HIGH); | Write HIGH or LOW value to digital pin |
| **analogRead()** | | **analogRead**(pin); | **int** m=**analogRead**(A0);  **int** n=**analogRead**(A1); | Reads the analog value from a specified analog pin(A0-A5) |
| **analogWrite()** | | **analogWrite**(pin); | **int** m=**analogWrite**(A0);  **int** n=**analogWrite**(A1); | Write the analog value from a specified analog pin(A0-A5) |
| **delay()** | | **delay**(value)**;**  **millis**(value)**;**  **micros**(value)**;** | **delay**(1000)**;** //one sec dealy  **millis**(1000)**;** //one milli sec dealy  **micros**(1000)**;** //one micro sec dealy | Pauses the program for the amount of time specified |
| **Map()** | | **map(**variable, alow, ahigh, mlow ,mhigh**);** | **y=map(**x,0,1023,1,200**);** | Re-maps a number from one range to another |
| **Serial.begin()** | | **Serial.begin**(baud rate) | **Serial.begin**(9600) | Sets the data rate in bits per second (baud) for serial data transmission.(9600,14400 etc) |
| **Serial.print()** | | **Serial.print(**data**)** | **Serial.print(**78**) //gives 78**  **Serial.print(**"college") **//prints**  **College**  **Int** v**=3; Serial.print(**v**) // prints 3** | Prints data to the serial port as human readable ASCII text |
| **Serial.println()** | | **Serial.println()** | **Serial.println(**56**);** | Prints the value 56 fallowed by a carriage return character(ASCII 13 or '\n' ) i.e new line character |
| **Serial.read()** | | **Variable= Serial.read(**variable**)** | **v= Serial.read(**p**);**  **m=Serial.read(**A0**);** | Reads incoming serial data |
|  | |  |  |  |
| **Serial.available()** | | **Serial.available()** | **if (Serial.available()** > 0**)**  **{**  **int** incomingByte **= Serial.read();    }** | Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes) |
| **Serial.write()** | | **Serial.write(val)** | **Serial.write(45);**  **int** bytesent=**Serial.write("HELLO");** | Writes the binary data to the serial port  And it will return the number of bytes written on serial monitor |
| **Serial.println(val, format)** | | **Serial.println(val, format)** | **Serial.print(**78, BIN**) Serial.print(**78, OCT**)**  **Serial.print(**78, DEC**)**  **Serial.print(**78, HEX**)** | gives "1001110"  gives "116"  gives "78"  gives "4E |
| **Arrays** | | Data\_type array\_name[size]; | **int** mypins[]**=**{2,4,8};  **int** myval[4]**=**{2,4,-8,3}**;** | An array defines the pin numbers of 2,4,8 which acts as input or output  An array initializes four values of type **int** |
| **lcd.begin()** | | **lcd.begin(**cols,rows**);** | **lcd.begin(16, 2);** //defines for 16,2 LCD display | Initializes the interface to the lcd screen  **Note**: use #include <LiquidCrystal.h> to interface lcd with arduino |
| **lcd.print()** | | **lcd.print(**data**)** | **lcd.print(**"Hello world"**)** | Prints text to LCD |
| **lcd.cursor()** | | **lcd.cursor(**col,row**)** | **lcd.cursor(5,1**)//points to 5th coloumn,2nd row | an underscore (line) at the position to which the next character will be written |
| **lcd.clear()** | | **lcd.clear()** | **lcd.clear()** | Clears the LCD screen |
| **IPAddress()** | | **IPAddress**(address) | **IPAddress** ip(192, 168, 0, 2); | Defines an IP address. It can be used to declare both local and remote addresses. |
| **WiFi.config()** | | **WiFi.config**(ip);  **WiFi.config**(ip, dns);  **WiFi.config**(ip, dns, gateway);  **WiFi.config**(ip, dns, gateway, subnet); | **IPAddress** ip(192, 168, 0, 2);  **WiFi.config**(ip); | WiFi.config() allows you to configure a static IP address as well as change the DNS, gateway, and subnet addresses on the WiFi shield.  **Dns**: The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network.  Gateway: A gateway is a [network node](http://searchnetworking.techtarget.com/definition/node) connecting two networks that use different [protocols](http://searchnetworking.techtarget.com/definition/protocol).  **Subnet:** A subnetwork or subnet is a logical subdivision of an IP network. |
|  | |  |  |  |
| **WiFi.begin()** | | **WiFi.begin**();  **WiFi.begin**(ssid);  **WiFi.begin**(ssid, pass);  **WiFi.begin**(ssid, keyIndex, key); | **WiFi.begin**();  **WiFi.begin**(ssid);  **WiFi.begin**(ssid, pass);  **WiFi.begin**(ssid, keyIndex, key); | Initializes the WiFi library's network settings and provides the current status.  **Ssid**: Wifi name(Wifi Name)  **Pass**: password of the ssid (wifi router Pass) |
| **WiFiServer()** | | **Server**(port);  port: the port to listen on (int) | **WiFiServer** server(80); | Creates a server that listens for incoming connections on the specified port. |
| **WiFi.status()** | | **WiFi.status**(); | **WiFi.status**(); | Return the connection status. |
| **server.begin();** | | **server.begin**() | **server.begin**(); | Tells the server to begin listening for incoming connections. |
| **WiFiClient()** | | **WiFiClient**() | **WiFiClient** client;  **client.connect(**server, 80) | Creates a client that can connect to to a specified internet IP address and port as defined in client.connect(). |
| **server.available()** | | **server.available**() | **WiFiClient** client = **server.available**(); | Gets a client that is connected to the server and has data available for reading. The connection persists when the returned client object goes out of scope; |
| **client.connect()** | | **client.connect**() **client.connect**(ip, port) **client.connect**(URL, port) | **client.connect**(server, 80) | Connects to a specified IP address and port. The return value indicates success or failure. Also supports DNS lookups when using a domain name.  **ip**: the IP address that the client will connect to (array of 4 bytes)  **URL**: the domain name the client will connect to (string, ex.:"arduino.cc")  **port:** the port that the client will connect to (int) |
| **client.read()** | | **client.read**() | **client.read**() | Read the next byte received from the server the client is connected to |
| **client.println()** | | **client.println**()  **client.println**(data)  **client.print**(data, BASE) | **client.println**(“Hello World”) | Print data, followed by a carriage return and newline, to the server a client is connected to. Prints numbers as a sequence of digits, each an ASCII character |
| **WiFi.SSID()** | | **WiFi.SSID**();  **WiFi.SSID**(wifiAccessPoint) | char ssid[] = "yourNetwork";  **WiFi.begin**(ssid); | Gets the SSID of the current network  **wifiAccessPoint**: specifies from which network to get the information |
| **WiFi.localIP()** | | **WiFi.localIP**(); | **IPAddress** ip;  ip = **WiFi.localIP**(); **Serial.println**(ip); | Gets the WiFi shield's IP address |
| **WiFi.RSSI()** | | **WiFi.RSSI**();  **WiFi.RSSI**(wifiAccessPoint); | long rssi = **WiFi.RSSI**(); **Serial.print**("RSSI:"); **Serial.println**(rssi); | Gets the signal strength of the connection to the router |
| **WiFi.subnetMask()** | | **WiFi.subnet**(); | **IPAddress** subnet;  subnet = WiFi.subnetMask(); | Gets the WiFi shield's subnet mask |
| **WiFi.gatewayIP()** | | **WiFi.gatewayIP**(); | **IPAddress**gateway;  gateway = **WiFi.gatewayIP**();  **Serial.print**("GATEWAY: ");  **Serial.println**(gateway); | Gets the WiFi shield's gateway IP address. |
| **WiFi.encryptionType()** | | **WiFi.encryptionType**();  **WiFi.encryptionType**(wifiAccessPoint); | **byte** encryption = **WiFi.encryptionType**();   **Serial.print**("Encryption Type:");   **Serial.println**(encryption,HEX); | Gets the encryption type of the current network |
| **WiFi.scanNetworks();** | | **WiFi.scanNetworks**(); | **byte** numSsid = **WiFi.scanNetworks**();  **Serial.print**("SSID List:");   **Serial.println**(numSsid); | Scans for available WiFi networks and returns the discovered number |
| **WiFi.macAddress(mac);** | | **WiFi.macAddress**(mac); | **WiFi.macAddress**(mac);   **Serial.print**("MAC: ");   **Serial.print**(mac[5],**HEX**);   **Serial.print**(":");   **Serial.print**(mac[4],**HEX**);   **Serial.print**(":");   **Serial.print**(mac[3],**HEX**);   **Serial.print**(":");   **Serial.print**(mac[2],**HEX);   Serial.print**(":"); **Serial.print**(mac[1],**HEX);   Serial.print**(":"); **Serial.println**(mac[0],**HEX**); | Gets the MAC Address of your WiFi shield  **mac**: a 6 byte array to hold the MAC address |
| **client.flush()** | | **client.flush**() | **client.flush**() | This function discards any bytes that have been written to the client but not yet read. |
| **client.write()** | | **client.write**(val)  **client.write**(buf, len) | Client myClient;  **myClient.write**("Hi there"); | Write data to the server the client is connected to. This data is sent as a byte or series of bytes.  **val**: a value to send as a single byte (byte or char)  **buf**: an array to send as a series of bytes (byte or char)  **len**: the length of the buffer |
| **client.stop()** | | **client.stop**() | **client.stop**() | Disconnect the client from the server. It closes the established connection. |

Interfacing sensors with NodeMCU

1. Blinking of LED using NodeMCU

This example shows the simplest thing you can do with a NodeMCU to see physical output it blinks an LED.

1. Component Used
2. NodeMCU
3. Breadboard
4. LED
5. Connecting Wires
6. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU on the breadboard.
3. Connect the LED on the Breadboard.
4. Connect the VCC of led to D1 of the Nodemcu and GND to GND respectively.
5. Upload the code.
6. Code

int LED=5;

void setup() // the setup function runs once when you press reset or power the board

{ pinMode(LED, OUTPUT); // initialize digital pin D1 as an output.

}

void loop() // the loop function runs over and over again forever

{ digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}

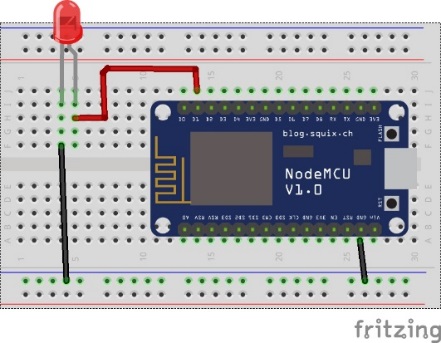
1. **Fading of LED using NodeMCU**

The example will demonstrate the fading of LED from Low brightness to high brightness and going back to low using NodeMCU.

1. Component Used

As earlier Program.

1. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU on the breadboard.
3. Connect the LED on the Breadboard.
4. Connect the VCC of led to **D1** of the Nodemcu and GND to GND respectively.
5. Upload the code.
6. Code

int led = 5; // the PWM pin the LED is attached

int brightness = 0; // how bright the LED is

int fadeAmount = 5; // how many points to fade the LED by

void setup()

{

pinMode(led, OUTPUT);

}

void loop()

{

analogWrite(led, brightness);

brightness = brightness + fadeAmount; // change the brightness for next time through the loop

if (brightness == 0 || brightness == 255) // reverse the direction of the fading at the ends of the fade:

{

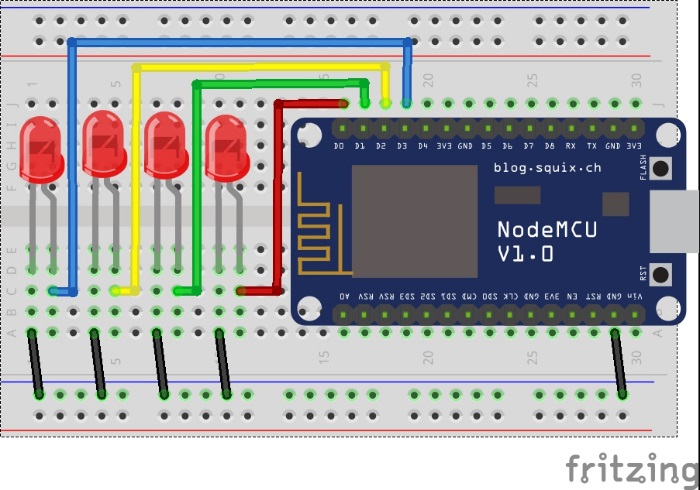
fadeAmount = -fadeAmount ;}

delay(30); // wait for 30 milliseconds to see the dimming effect }

1. **Running Led pattern 1 using NodeMCU**

This experiment will demonstrate to create a pattern using LED for example running LED from left to rightusing NodeMCU.

1. Component required
2. NodeMCU
3. Breadboard.
4. LED
5. Connecting Wires
6. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU on the Breadboard.
3. Place all the 4 led on the breadboard.
4. Connect all the GND pin of led with GND rail on breadboard.
5. Connect the 1st led with D0, 2nd with D1, 3rd with D2 and 4th with D3 pin of the NodeMCU.
6. Upload the code.
7. Code

int led[4]={ 16,5,4,0}; // pin defined for all the 4 led

void setup()

{

for(int i=0; i<4; i++)

{

pinMode(led[i], OUTPUT); // initialing all the array of 4 pin as output

}

}

void loop()

{

for(int i =0; i<4;i++)

{

digitalWrite(led[i], HIGH); // turning ON the led from left to right

delay(200); // Creating a delay

digitalWrite(led[i], LOW); // turning OFF the led from left to right

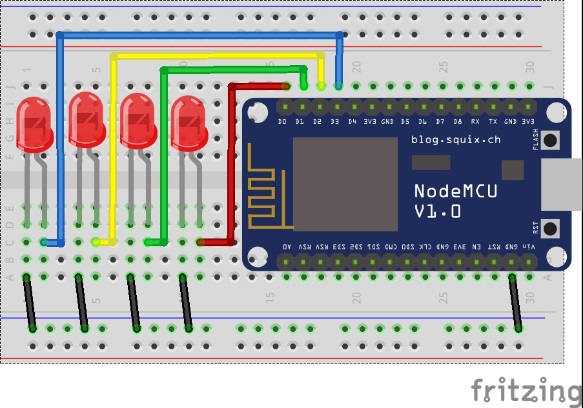
}

}

1. **Running Led pattern 2 using NodeMCU**

This experiment will demonstrate to create a pattern with LED using NodeMCU.

1. Component required
2. NodeMCU
3. Breadboard.
4. LED
5. Connecting Wires
6. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU on the Breadboard.
3. Place all the 4 led on the breadboard.
4. Connect all the GND pin of led with GND rail on breadboard.
5. Connect the 1st led with D0, 2nd with D1, 3rd with D2 and 4th with D3 pin of the NodeMCU.
6. Upload the code.
7. Code

int ledPin[] = {16,5,4,0}; // Defining the Pin

void setup()

{

for (int i =0;i<4;i++)

{

pinMode(ledPin[i], OUTPUT); // initialing the pin as output

}

}

void loop()

{

byte nums[] = {0, 1, 3, 6, 4, 12, 8, 12, 4, 6, 3, 1, 0}; // numbers to be displayed

for (byte i = 0; i<13;i++)

{

displayBinary(nums[i]); // calling the function and by passing the values

delay(25);

}

}

void displayBinary(byte numToShow) // the function to convert the integer value to binary led display

{

/\*the bitRead is used to read the 4 bit binary number and then in for loop i. the i read the position of the bit and turn on the led ex: 1001 then the led connected at position 0 and 3 should be turn On rest OFF.\*/

for (int i =0;i<4;i++)

{

if(bitRead(numToShow, i)==1)

{

digitalWrite(ledPin[i], HIGH);

}

else

{

digitalWrite(ledPin[i], LOW);

}

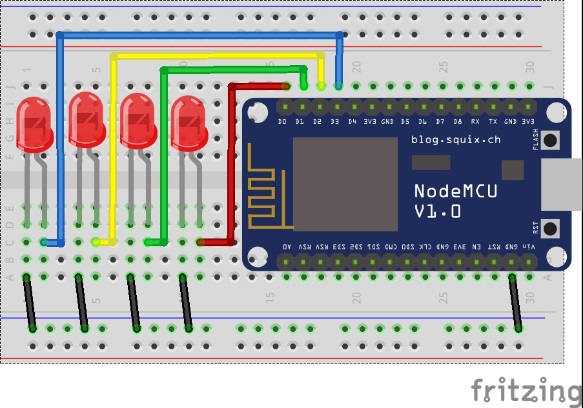
}

}

1. Hexadecimal counter with led using nodemcu

This experiment is going to demonstrate the hexadecimal counter countdown from 0 to 15 using led and NodeMCU.

1. Component required
2. NodeMCU
3. Breadboard.
4. LED
5. Connecting Wires
6. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU on the Breadboard.
3. Place all the 4 led on the breadboard.
4. Connect all the GND pin of led with GND rail on breadboard.
5. Connect the 1st led with D0, 2nd with D1, 3rd with D2 and 4th with D3 pin of the NodeMCU.
6. Upload the code.
7. Code

int ledPin[] = {16, 5, 4, 0}; // Defining the pins

void setup()

{

for (int i = 0; i < 4; i++)

{

pinMode(ledPin[i], OUTPUT); // initializing the pins

}

}

void loop()

{

for (byte counter = 0; counter <= 15; counter++) // counting from 0 to 15

{

displayBinary(counter); // function calling and passing the value

delay(1000);

}

}

void displayBinary(byte numToShow)

{

/\*the bitRead is used to read the 4 bit binary number and then in for loop i. the i read the position of the bit and turn on the led ex: 1001 then the led connected at position 0 and 3 should be turn On rest OFF.\*/

for (int i = 0; i < 4; i++)

{

if (bitRead(numToShow, i) == 1)

{

digitalWrite(ledPin[i], HIGH);

}

else

{

digitalWrite(ledPin[i], LOW);

}

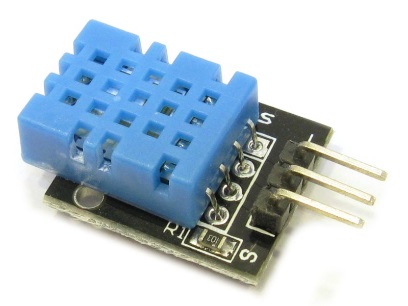
}

}

1. Interfacing Nodemcu with Temperature and Humidity Sensor (Dht11).

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).

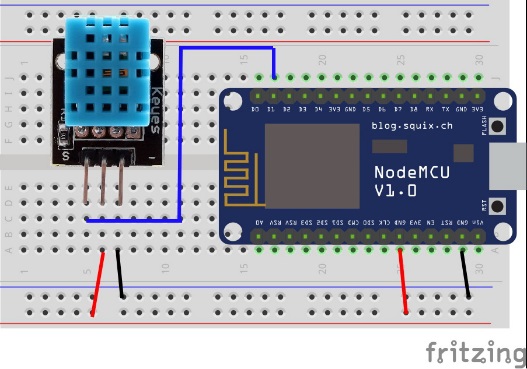
The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes. The DHT11 measures temperature with a surface mounted [NTC temperature sensor](http://buy.geni.us/Proxy.ashx?TSID=13213&GR_URL=http%3A%2F%2Fwww.amazon.com%2Fgp%2Fproduct%2FB00GD471PO%2Fref%3Das_li_qf_sp_asin_il_tl%3Fie%3DUTF8%26camp%3D1789%26creative%3D9325%26creativeASIN%3DB00GD471PO%26linkCode%3Das2%26tag%3Dcircbasi-20%26linkId%3DUJTHZ5Z3JDMKGOCK) (thermistor) built into the unit.



Technical Specification:

* Humidity Range: 20-90% RH
* Humidity Accuracy: ±5% RH
* Temperature Range: 0-50 °C
* Temperature Accuracy: ±2% °C
* Operating Voltage: 3V to 5.5V

1. Components Required
2. NodeMCU
3. DHT11
4. Breadboard
5. Connecting Wires
6. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU on the breadboard.
3. Place the DHT11 Temperature and Humidity Sensor on the breadboard (assuming DHT11library is installed).
4. Connect VCC and GND of DHT11with 3.3v and GND to NodeMCU.
5. Connect data pin from DHT11 to D2 of the NodeMCU.
6. Upload the code.
7. Code

#include <DHT11.h> // including the library for dht11

int pin = 5; //Defining the pin D1

DHT11 dht11(pin); //initialization of pin

void setup()

{

Serial.begin(9600);

}

Void loop()

{

int err;

float temp, humi;

if ((err = dht11.read(humi, temp)) == 0) // check if dht11 reads the value

{

Serial.print("temperature:");

Serial.print(temp); //Displaying the Temperature

Serial.print(" humidity:");

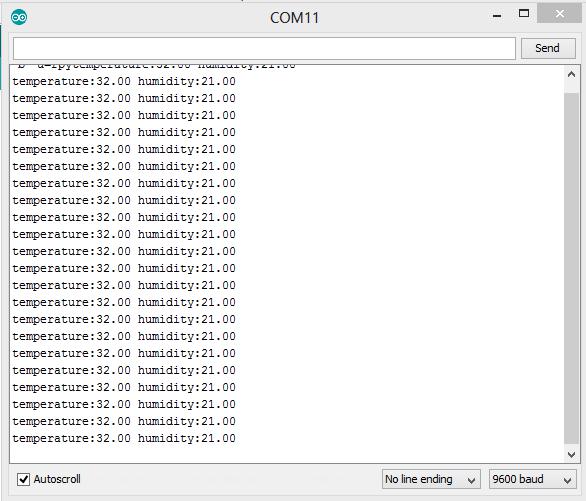
Serial.print(humi); //Displaying the Humidity

Serial.println();

}

}

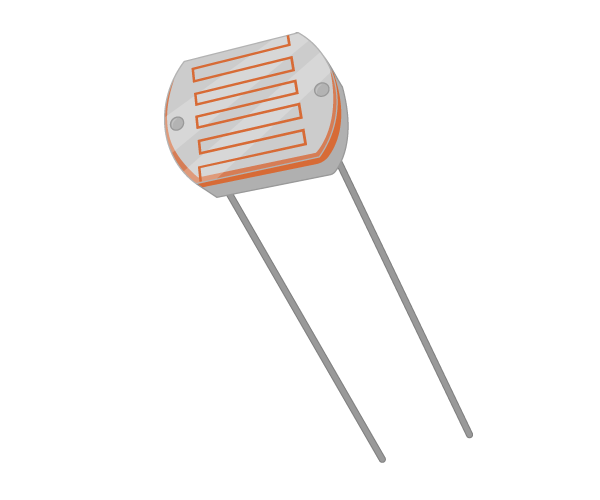
1. Output



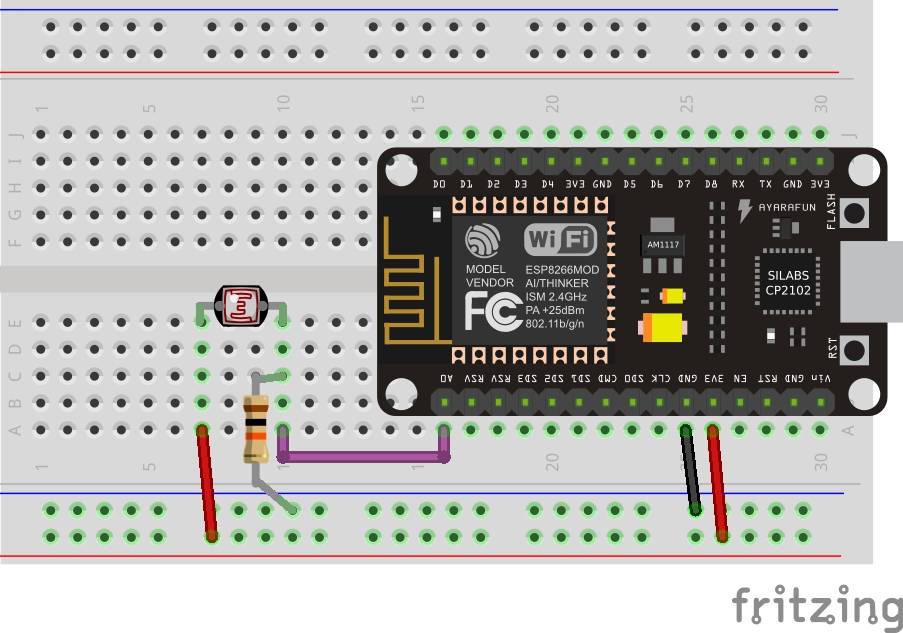
1. Application
2. Attention to chemical materials inside the chemical drums.
3. Restoration process.
4. Temperature Affect.
5. Monitoring Cold storage vehicle.
6. Home Automation.
7. Interfacing ldr with nodemcu

A **Light Dependent Resistor** (LDR) or a photo [resistor](https://www.electrical4u.com/types-of-resistor-carbon-composition-and-wire-wound-resistor/) is a device whose [resistivity](https://www.electrical4u.com/electrical-resistance-and-laws-of-resistance/) is a function of the incident electromagnetic radiation. Hence, they are light sensitive devices. They are also called as photo conductors, photo conductive cells or simply photocells. They are made up of [semiconductor](https://www.electrical4u.com/theory-of-semiconductor/) materials having high resistance.

A **light dependent resistor** works on the principle of photo conductivity.Photo conductivity is an optical phenomenon in which the materials conductivity is increased when light is absorbed by the material. When light falls i.e. when the photons fall on the device, the electrons in the valence band of the semiconductor material are excited to the conduction band. These photons in the incident light should have energy greater than the band gap of the semiconductor material to make the electrons jump from the valence band to the conduction band. Hence when light having enough energy strikes on the device, more and more electrons are excited to the conduction band which results in large number of [charge carriers](https://www.electrical4u.com/mobility-of-charge-carrier/). The result of this process is more and more [current](https://www.electrical4u.com/electric-current-and-theory-of-electricity/) starts flowing through the device when the circuit is closed and hence it is said that the resistance of the device has been decreased.



1. Component Used
2. NodeMCU
3. LDR(**Light Dependent Resistor)**
4. 10K resistor
5. Wire
6. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU (ESP8266) on the breadboard.
3. Take LDR and place it on Breadboard.
4. Connect VCC (3.3V) toward any one pin (say pin 1) of the LDR.
5. Connect Pull down Resistor with ground on the other pin (say pin 2) of LDR.
6. Connect a wire from the other pin (pin 2) to A0 of NodeMCU.
7. Code

void setup()

{

Serial.begin(9600); //setting up the serial monitor

}

void loop()

{

int sensorValue = analogRead(A0); //Accepting the analog value

Serial.println(sensorValue); //Printing the value on the serial monitor

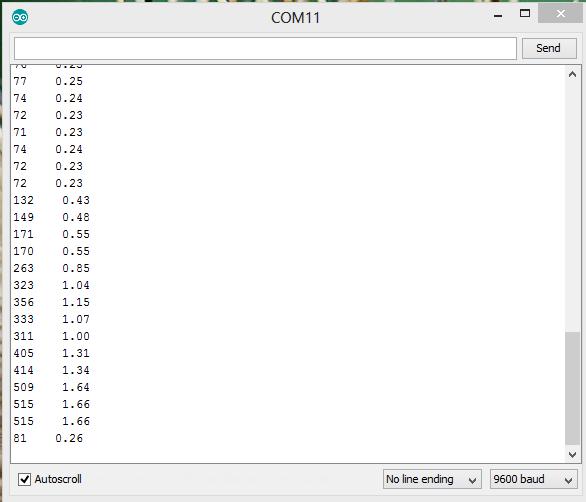
float voltage = sensorValue \* (3.3 / 1023.0); //Converting the value to Volt

Serial.println(voltage); // Printing Voltage

delay(1000);

}

1. Output



1. Application
2. Camera Exposure Control
3. Photocopy Machines - density of toner
4. Street Light Control
5. Automatic Headlight Dimmer
6. Night Light Control
7. Interfacing Ultrasonic Sensor with nodemcu

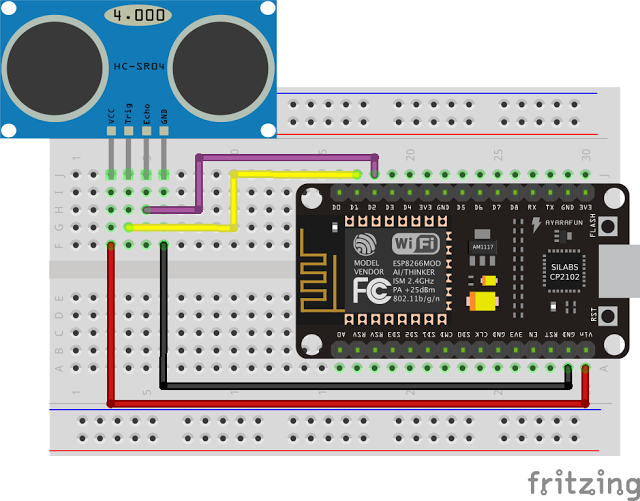
This is the HC-SR04 ultrasonic ranging sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.

There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground). You will find this sensor very easy to set up and use for your next range-finding project.

It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.



1. Components used
2. NodeMCU
3. Ultrasonic Sensor
4. Wire
5. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU (ESP8266 ) on the breadboard.
3. Take Ultrasonic Sensor and place it on Breadboard.
4. Connect VCC (3.3V) and GND along with TRIG and ECHO.
5. Connect TRIG to pin D1 and ECHO to pin D2.
6. Upload the code given.
7. Code

#define TRIGGER 5 // defining trigger pin

#define ECHO 4 // defining echo pin

void setup()

{Serial.begin(9600);

pinMode(TRIGGER, OUTPUT); //initializing trigger as output

pinMode(ECHO, INPUT); //initialing trigger as input }

void loop()

{

int duration, dist;

digitalWrite(TRIGGER, LOW); // make trigger low

delayMicroseconds(2);

digitalWrite(TRIGGER, HIGH); // make trigger high

delayMicroseconds(10); //give 10 microsec delay

digitalWrite(TRIGGER, LOW);

duration = pulseIn(ECHO, HIGH);

dist = (duration / 2) / 29.1; // calibrate the distance using pulse

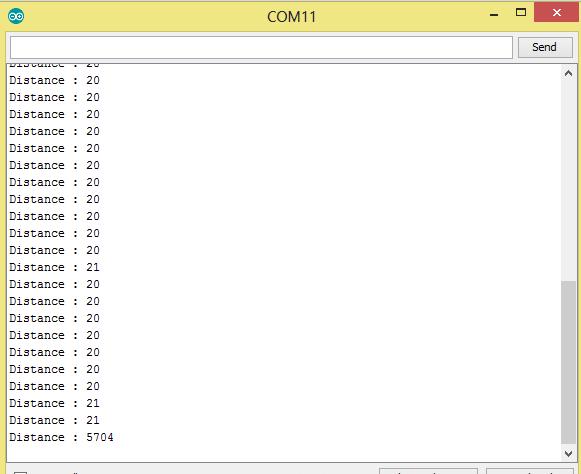
Serial.print(dist); // Print the distance value

Serial.println(" cm");

delay(500);

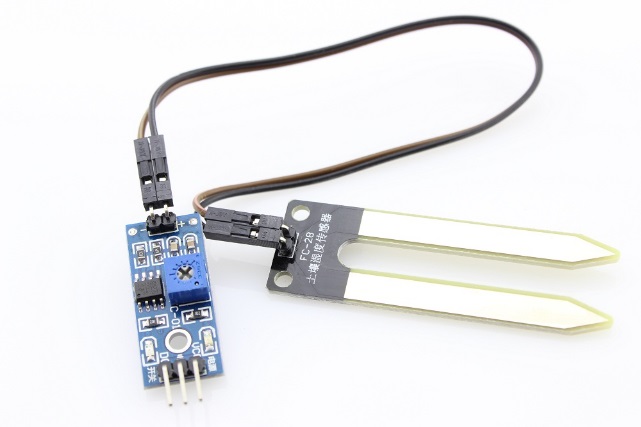
}

1. Output

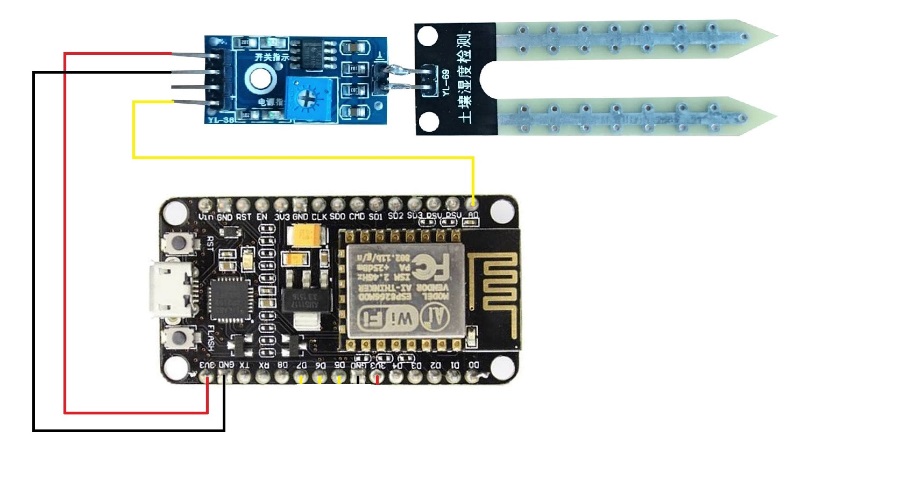


1. Application
2. [Level Measuring](https://www.pepperl-fuchs.com/global/en/22534.htm)
3. [Detecting Boom Height on Agricultural Machine](https://www.pepperl-fuchs.com/global/en/22494.htm)
4. [Ultrasonic Sensors for Anti-Collision Detection on Aerial Work Platforms](https://www.pepperl-fuchs.com/global/en/23828.htm)
5. [Bottle Counting on Drink Filling Machines](https://www.pepperl-fuchs.com/global/en/24176.htm)
6. People detection counting
7. Interfacing Soil moister Sensor with nodemcu

Soil moisture sensors measure the volumetric [water content](https://en.wikipedia.org/wiki/Water_content) in [soil](https://en.wikipedia.org/wiki/Soil). Since the direct [gravimetric measurement](https://en.wikipedia.org/wiki/Gravimetric_analysis) of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with [neutrons](https://en.wikipedia.org/wiki/Neutron), as a proxy for the moisture content. The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, [temperature](https://en.wikipedia.org/wiki/Temperature), or [electric conductivity](https://en.wikipedia.org/wiki/Electric_conductivity). Reflected [microwave](https://en.wikipedia.org/wiki/Microwave) radiation is affected by the soil moisture and is used for [remote sensing](https://en.wikipedia.org/wiki/Remote_sensing) in [hydrology](https://en.wikipedia.org/wiki/Hydrology) and agriculture. Portable probe instruments can be used by farmers or gardeners.



1. Component Required
2. NodeMCU
3. Soil moisture sensor
4. Connecting Wires
5. Breadboard
6. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU (ESP8266 ) on the breadboard.
3. Take Ultrasonic Sensor and place it on Breadboard.
4. Connect VCC and GND with 3.3 v and GND of nodemcu.
5. Connect the A0 with the A0 of the nodemcu.
6. Code

void setup()

{

Serial.begin(9600);

}

void loop()

{

int soil\_moisture=analogRead(A0); // read from analog pin A0

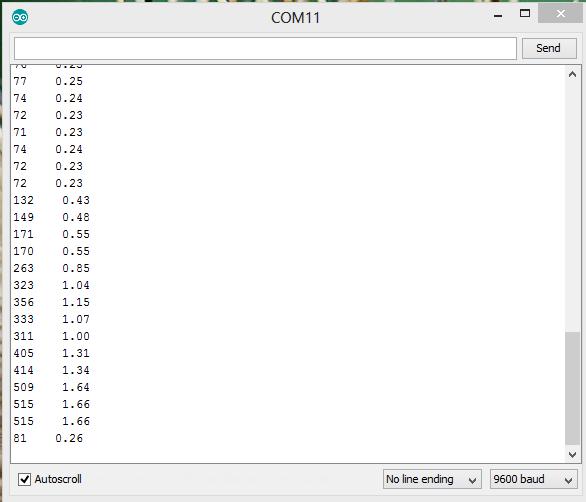
Serial.print("analog value: ");

Serial.println(soil\_moisture);

delay(1000);

}

1. Output

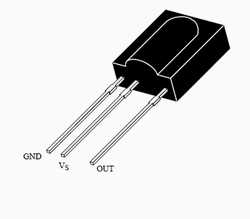


1. Interfacing TSOP (Infrared Receiver) with Nodemcu.

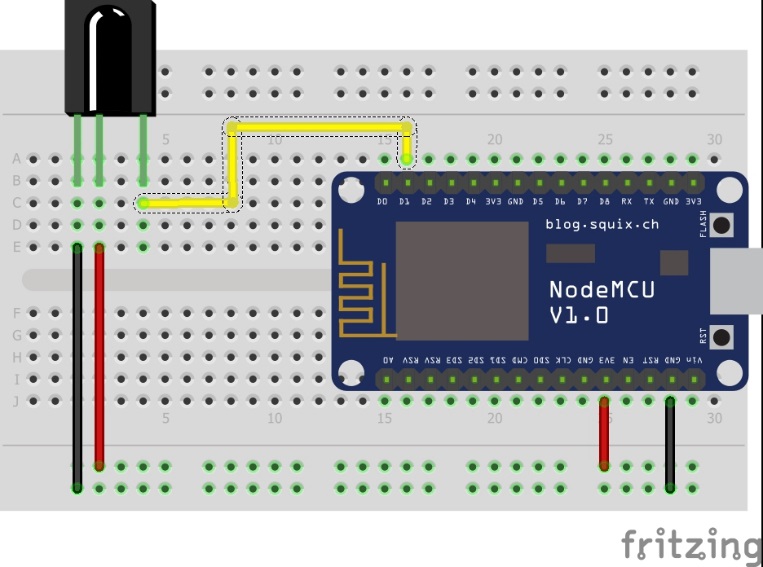
The TSOP 1738 is a member of IR remote control receiver series. This IR sensor module consists of a PIN diode and a pre amplifier which are embedded into a single package. The output of TSOP is active low and it gives +5V in off state. When IR waves, from a source, with a center frequency of 38 kHz incident on it, its output goes low. Lights coming from sunlight, fluorescent lamps etc. may cause disturbance to it and result in undesirable output even when the source is not transmitting IR signals. A band pass filter, an integrator stage and an automatic gain control are used to suppress such disturbances.

TSOP module has an inbuilt control circuit for amplifying the coded pulses from the IR transmitter. A signal is generated when PIN photodiode receives the signals. This input signal is received by an automatic gain control (AGC). For a range of inputs, the output is fed back to AGC in order to adjust the gain to a suitable level. The signal from AGC is passed to a band pass filter to filter undesired frequencies. After this, the signal goes to a demodulator and this demodulated output drives an npn transistor. The collector output of the transistor is obtained at pin 3 of TSOP module.

Members of TSOP17xx series are sensitive to different center frequencies of the IR spectrum. For example TSOP1738 is sensitive to 38 kHz whereas TSOP1740 to 40 kHz center frequency.



1. Component Required
2. NodeMCU
3. TSOP
4. Connecting Wires
5. BreadBoard
6. Remote Control
7. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU on the Breadboard.
3. Place the TSOP on the breadboard.
4. Connect the 1st pin to GND followed by VCC rail.
5. Connect the Data line to the pin D1 of the NodeMCU.
6. Install the Library “IRremoteESP8266.h “.
7. Upload the code.
8. Code

#include <IRremoteESP8266.h> //library include

int RECV\_PIN = 5; //defining pin for IR receiver

IRrecv irrecv(RECV\_PIN);

decode\_results results;

long lastMsg = 0;

char msg[50];

int value = 0;

void setup()

{

Serial.begin(9600);

irrecv.enableIRIn(); // Start the receiver

}

void loop()

{

long now = millis();

if (now - lastMsg > 2000)

{

if (irrecv.decode(&results))

{

Serial.println(results.value, HEX); // printing the hex code or key

irrecv.resume(); // Receive the next value

}

}

}

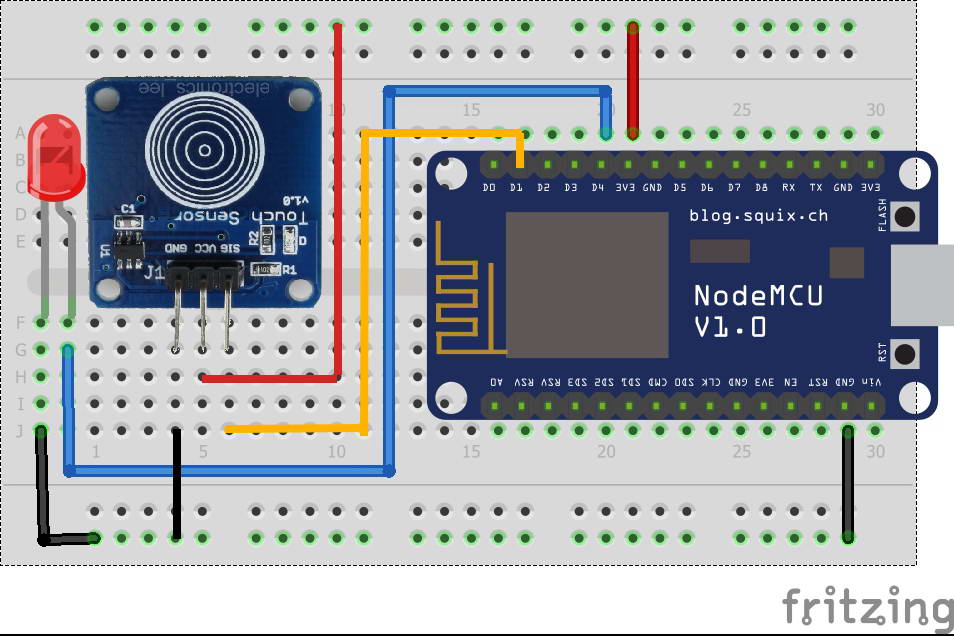
1. Output



1. Application
2. Controlling fan and light with tv remote
3. Remote access of other appliance.
4. Interfacing Touch Sensor with Nodemcu.

This device uses your body as part of the circuit.  When you touch the sensor pad, the capacitance of the circuit is changed and is detected.  That detected change in capacitance results in the output changing states.

1. Component Required
2. NodeMCU
3. Touch Sensor
4. Led
5. Wires
6. Breadboard
7. Circuit Diagram



1. Connection Detail
2. Place the Nodemcu on the breadboard.
3. Place the touch sensor and LED on the breadboard.
4. Connect VCC and GND of the touch sensor to 3.3v and GND of the Nodemcu.
5. Connect the –ve pin of the LED to the GND of the nodemcu.
6. Connect +ve pin of the LED to the **D4**(GPIO 2) of the Nodemcu.
7. Connect the SIG pin of the touch sensor to the **D1** (GPIO 5) of the nodemcu.
8. Code

int TouchSensor = 5; //connected to Digital pin D1

int led = 2; //connected to Digital pin D4

void setup()

{

Serial.begin(9600); // Communication speed

pinMode(led, OUTPUT);

pinMode(TouchSensor, INPUT);

}

void loop(){

if(digitalRead(TouchSensor)==HIGH) //Read Touch sensor signal

{

digitalWrite(led, HIGH); // if Touch sensor is HIGH, then turn on

Serial.println("Led ON");

}

else

{

digitalWrite(led, LOW); // if Touch sensor is LOW, then turn off the led

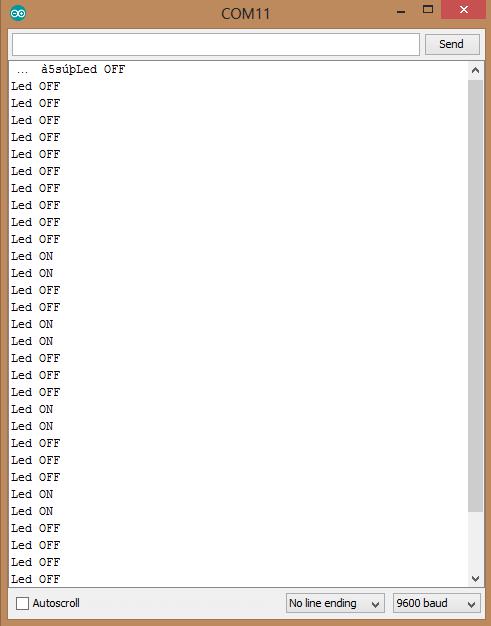
Serial.println("Led OFF");

}

delay(500); // Slow down the output for easier reading

}

1. Output



1. Applications
2. Remote controllers
3. Light switches
4. Sealed control panels
5. Home appliances
6. Door-lock systems
7. Interfacing Nodemcu to scan all the available WiFi network

This experiment scans all the available WiFi network and show the on the serial monitor.

1. Component Required
2. Nodemcu
3. Connection Detail
4. Connect the Nodemcu to the system.
5. Upload the code.
6. Code

#include "ESP8266WiFi.h"

void setup() {

Serial.begin(115200); // Set WiFi to station mode

WiFi.mode(WIFI\_STA);

WiFi.disconnect();

delay(100);

Serial.println("Setup done"); }

void loop()

{

Serial.println("scan start");

int n = WiFi.scanNetworks(); // WiFi.scanNetworks will return the number of networks found

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) == ENC\_TYPE\_NONE)?" ":"\*");

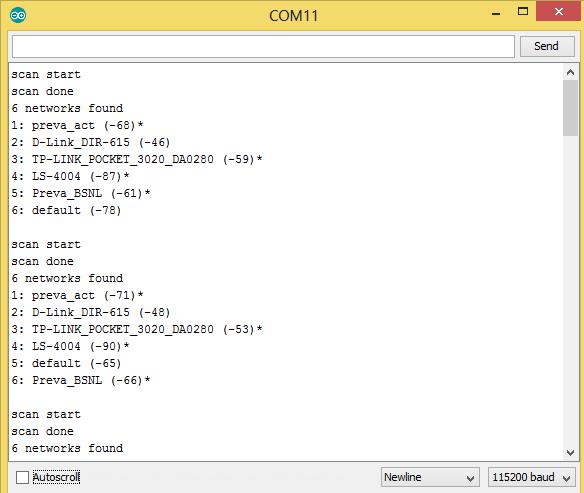
delay(10);

} }

Serial.println("");

delay(5000); }

1. Output



1. interfacing webserver program to say “Hello World” using Nodemcu

This Experiment is going to execute a web server program and display “Hello World” on the web using the IP address with the help of NodeMCU.

1. Component Required
2. Nodemcu
3. Connection Detail
4. Connect the Nodemcu to the system.
5. Upload the code.
6. Code

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include <ESP8266mDNS.h>

const char\* ssid = " YOUR SSID";

const char\* password = " YOUR PASSWORD ";

ESP8266WebServer server(80);

const int led = 16;

void handleRoot() {

digitalWrite(led, 1);

server.send(200, "text/plain", "Hello world!!! How are you today");

digitalWrite(led, 0);

}

void handleNotFound(){

digitalWrite(led, 1);

String message = "File Not Found\n\n";

message += "URI: ";

message += server.uri();

message += "\nMethod: ";

message += (server.method() == HTTP\_GET)?"GET":"POST";

message += "\nArguments: ";

message += server.args();

message += "\n";

for (uint8\_t i=0; i<server.args(); i++){

message += " " + server.argName(i) + ": " + server.arg(i) + "\n";

}

server.send(404, "text/plain", message);

digitalWrite(led, 0); }

void setup(void){

pinMode(led, OUTPUT);

digitalWrite(led, 0);

Serial.begin(115200);

WiFi.begin(ssid, password);

Serial.println("");

// Wait for connection

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

if (MDNS.begin("esp8266")) {

Serial.println("MDNS responder started");

}

server.on("/", handleRoot);

server.on("/inline", [](){

server.send(200, "text/plain", "this works as well");

});

server.onNotFound(handleNotFound);

server.begin();

Serial.println("HTTP server started");

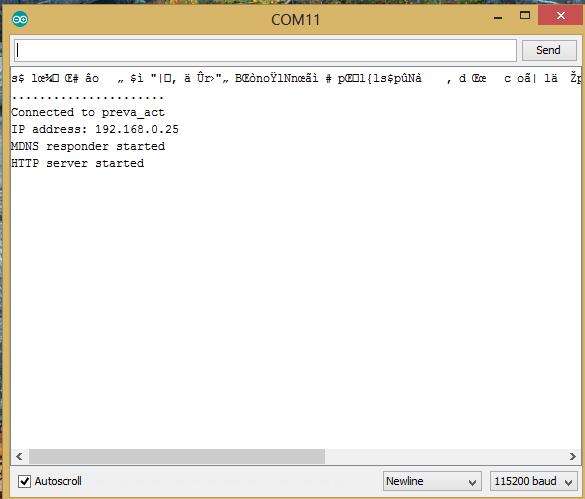
}

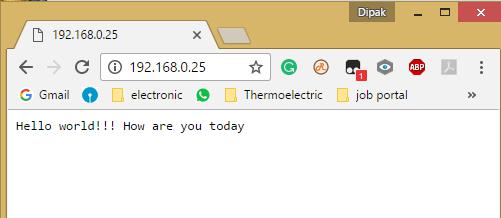
void loop(void){

server.handleClient();

}

1. Output

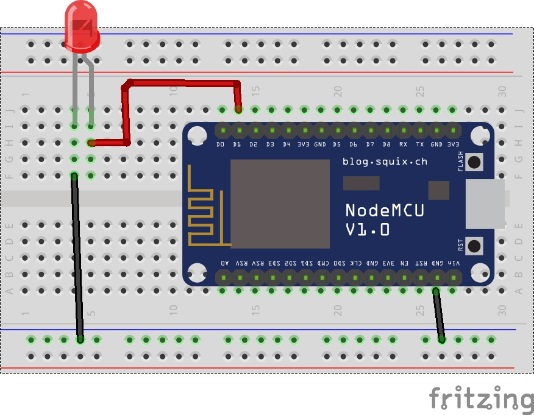




1. interfacing webserver program to turn ON and OFF the led using Nodemcu

This Experiment is going to execute a web server program to "Turn On and Turn Off" an LED that has connected on the web using the IP address with the help of NodeMCU.

1. Component Required
2. Nodemcu
3. LED
4. Connecting Wires
5. Connection Detail
6. Connect the Nodemcu to the system.
7. Upload the code.
8. Circuit Diagram



1. Code

#include <ESP8266WiFi.h>

const char\* ssid = "YOUR SSID";

const char\* password = "YOUR PASSWORD";

int ledPin = 5; // GPIO13

WiFiServer server(80);

void setup()

{

Serial.begin(115200);

delay(10);

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW);

Serial.println();

Serial.println();

Serial.print("Connecting to "); // Connect to WiFi network

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

server.begin(); // Start the server

Serial.println("Server started");

Serial.print("Use this URL to connect: "); // Print the IP address

Serial.print("http://");

Serial.print(WiFi.localIP());

Serial.println("/");

}

void loop()

{

WiFiClient client = server.available(); // Check if a client has connected

if (!client)

{

return;

}

Serial.println("new client");

while(!client.available()) // Wait until the client sends some data

{

delay(1);

}

String request = client.readStringUntil('\r');

Serial.println(request);

client.flush();

int value = LOW;

if (request.indexOf("/LED=ON") != -1)

{

digitalWrite(ledPin, HIGH);

value = HIGH;

}

if (request.indexOf("/LED=OFF") != -1)

{

digitalWrite(ledPin, LOW);

value = LOW;

}

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println(""); // do not forget this one

client.println("<!DOCTYPE HTML>");

client.println("<html>");

client.print("Led pin is now: ");

if(value == HIGH)

{

client.print("On");

}

else

{

client.print("Off");

}

client.println("<br><br>");

client.println("<a href=\"/LED=ON\"\"><button>Turn On </button></a>");

client.println("<a href=\"/LED=OFF\"\"><button>Turn Off </button></a><br />");

client.println("</html>");

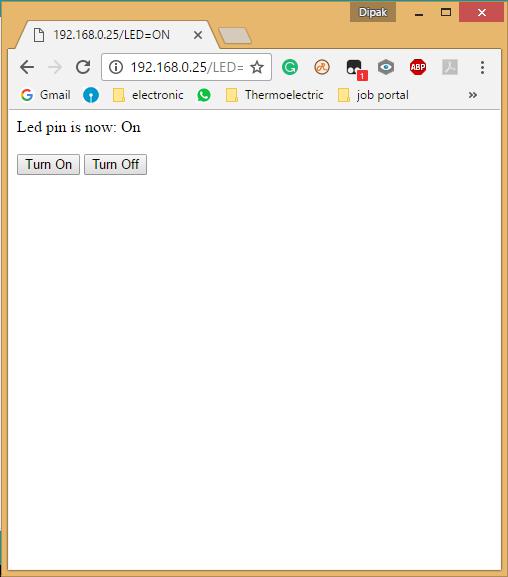
delay(1);

Serial.println("Client disonnected");

Serial.println("");

}

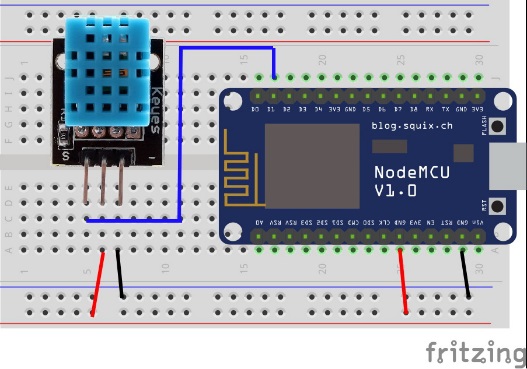
1. Output



1. interfacing webserver program to display the temperature and humidity on web using Nodemcu

This Experiment is going to execute a web server program to push the temperature and humidity sensor value on the web using the IP address with the help of NodeMCU.

1. Components Required
2. NodeMCU
3. DHT11
4. Breadboard
5. Connecting Wires
6. Circuit Diagram



1. Connection Detail
2. Place the NodeMCU on the breadboard.
3. Place the DHT11 Temperature and Humidity Sensor on the breadboard (assuming DHT11library is installed).
4. Connect VCC and GND of DHT11with 3.3v and GND to NodeMCU.
5. Connect data pin from DHT11 to D2 of the NodeMCU.
6. Upload the code.
7. Code

#include <ESP8266WiFi.h>

#include <DHT11.h>

int pin = 5; //node d5

DHT11 dht11(pin);

const char\* ssid = "YOUR SSID";

const char\* password = "YOUR PASSWORD";

WiFiServer server(80); // Web Server on port 80

void setup() {

Serial.begin(115200);

delay(10);

Serial.println(); // Connecting to WiFi network

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");

server.begin(); // Starting the web server

Serial.println("Web server running. Waiting for the ESP IP...");

delay(10000);

Serial.println(WiFi.localIP()); // Printing the ESP IP address

}

void loop()

{

WiFiClient client = server.available(); // Listenning for new clients

if (client)

{

Serial.println("New client");

boolean blank\_line = true;

while (client.connected())

{

if (client.available())

{

char c = client.read();

if (c == '\n' && blank\_line)

{

int err;

float temp, humi;

if ((err = dht11.read(humi, temp)) == 0)

{

Serial.print("temperature:");

Serial.print(temp);

Serial.print(" humidity:");

Serial.print(humi);

Serial.println();

}

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println("Connection: close");

client.println();

// your actual web page that displays temperature and humidity

client.println("<!DOCTYPE HTML>");

client.println("<html>");

client.println("<head></head><body><h1>ESP8266 - Temperature and

Humidity</h1><h3>Temperature in Celsius: ");

client.println(temp);

client.println("\*F</h3><h3>Humidity: ");

client.println(humi);

client.println("%</h3><h3>");

client.println("</body></html>");

break;

}

if (c == '\n') {

// when starts reading a new line

blank\_line = true;

}

else if (c != '\r') {

// when finds a character on the current line

blank\_line = false;

}

}

}

// closing the client connection

delay(1);

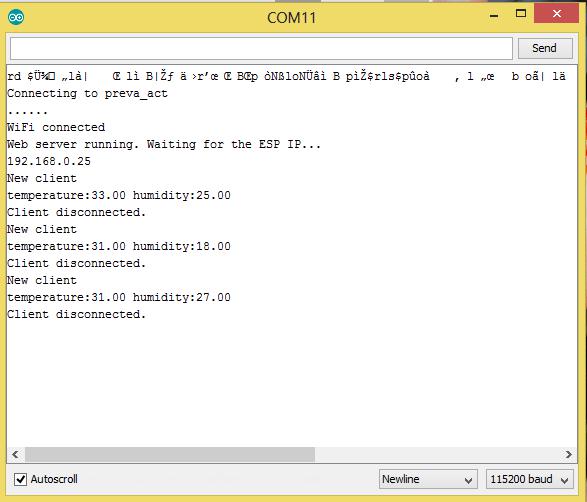
client.stop();

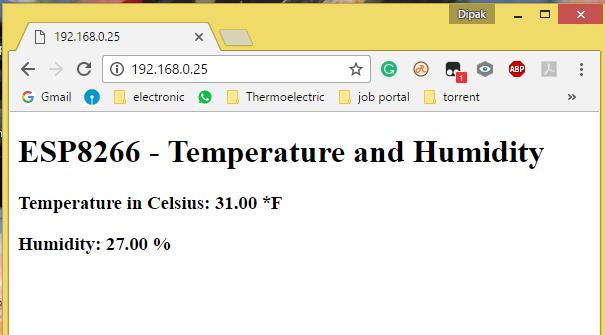
Serial.println("Client disconnected.");

}

}

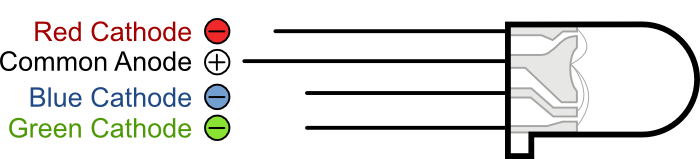
1. Output



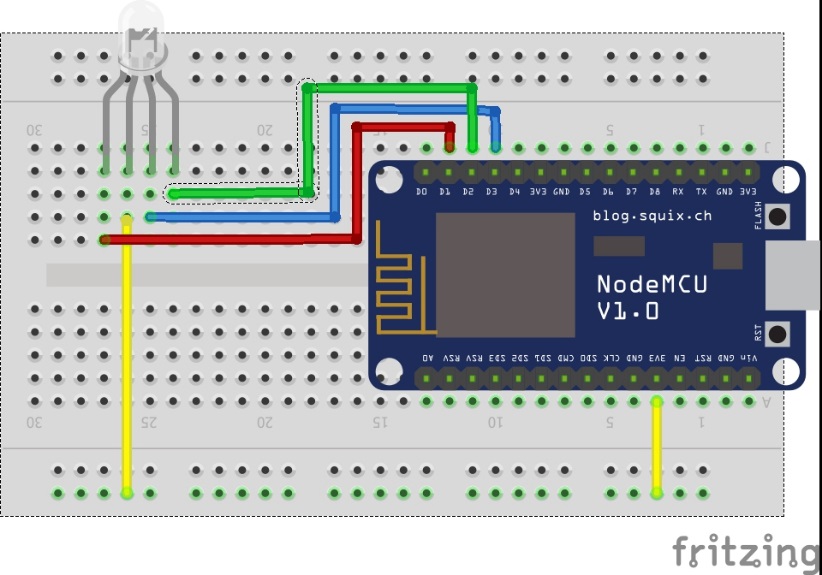


1. interfacing webserver program to control RGB led Through web using Nodemcu

Diffused 5mm tri-color LED with separate red, green and blue LED chips inside nice indicator, and fun to color-swirl. 60 degree viewing angle. We like diffused RGB LEDs because they color mix inside instead of appearing as 3 distinct LEDs. These are Common-Anode type which means you connect one pin to 5V or so and then tie the other three legs to ground through a resistor.



1. Component Required
2. RBG led
3. NodeMCU
4. Connecting Wire
5. Breadboard
6. Circuit Diagram



1. Connection
2. Place the Nodemcu on the Breadboard.
3. Place the RGB LED on the breadboard
4. The 1st pin is RED and will be connected to D1 (GPIO 5), 2nd to 3.3V, 3rd is blue and will be connected D3 (GPIO 2), and the 4th pin is green and will be connected to the D2 (GPIO 4).
5. Upload the code.
6. And open the Serial monitor to check the IP given. Copy and paste the IP on the browser.
7. Code

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include <ESP8266mDNS.h>

const char\* ssid = "YOUR SSID";

const char\* password = "YOUR PASSWORD";

ESP8266WebServer server(80);

const int R = 5;

const int G = 4;

const int B = 2;

void handleRoot()

{

String red = server.arg(0); // read RGB arguments

String green = server.arg(1);

String blue = server.arg(2);

analogWrite(R, red.toInt());

analogWrite(G, green.toInt());

analogWrite(B, blue.toInt());

Serial.println(red.toInt()); // for TESTING

Serial.println(green.toInt()); // for TESTING

Serial.println(blue.toInt()); // for TESTING

String webpage;

webpage += "<!DOCTYPE HTML>\r\n";

webpage += "<html>\r\n";

webpage += "<header><title>ESP8266 RGB LED</title><h1>ESP8266 RGBLED</h1></header>";

webpage += "<head>";

webpage += "<meta name='mobile-web-app-capable' content='yes' />";

webpage += "<meta name='viewport' content='width=device-width' />";

webpage += "</head>";

webpage += "<body style='margin: 0px; padding: 0px;'>";

webpage += "<canvas id='colorspace'></canvas></body>";

webpage += "<script type='text/javascript'>";

webpage += "(function () {";

webpage += " var canvas = document.getElementById('colorspace');";

webpage += " var ctx = canvas.getContext('2d');";

webpage += " function drawCanvas() {";

webpage += " var colours = ctx.createLinearGradient(0, 0, window.innerWidth, 0);";

webpage += " for(var i=0; i <= 360; i+=10) {";

webpage += " colours.addColorStop(i/360, 'hsl(' + i + ', 100%, 50%)');";

webpage += " }";

webpage += " ctx.fillStyle = colours;";

webpage += " ctx.fillRect(0, 0, window.innerWidth, window.innerHeight);";

webpage += " var luminance = ctx.createLinearGradient(0, 0, 0, ctx.canvas.height);";

webpage += " luminance.addColorStop(0, '#ffffff');";

webpage += " luminance.addColorStop(0.05, '#ffffff');";

webpage += " luminance.addColorStop(0.5, 'rgba(0,0,0,0)');";

webpage += " luminance.addColorStop(0.95, '#000000');";

webpage += " luminance.addColorStop(1, '#000000');";

webpage += " ctx.fillStyle = luminance;";

webpage += " ctx.fillRect(0, 0, ctx.canvas.width, ctx.canvas.height);";

webpage += " }";

webpage += " var eventLocked = false;";

webpage += " function handleEvent(clientX, clientY) {";

webpage += " if(eventLocked) {";

webpage += " return;";

webpage += " }";

webpage += " function colourCorrect(v) {";

webpage += " return Math.round(1023-(v\*v)/64);";

webpage += " }";

webpage += " var data = ctx.getImageData(clientX, clientY, 1, 1).data;";

webpage += " var params = [";

webpage += " 'r=' + colourCorrect(data[0]),";

webpage += " 'g=' + colourCorrect(data[1]),";

webpage += " 'b=' + colourCorrect(data[2])";

webpage += " ].join('&');";

webpage += " var req = new XMLHttpRequest();";

webpage += " req.open('POST', '?' + params, true);";

webpage += " req.send();";

webpage += " eventLocked = true;";

webpage += " req.onreadystatechange = function() {";

webpage += " if(req.readyState == 4) {";

webpage += " eventLocked = false;";

webpage += " }";

webpage += " }";

webpage += " }";

webpage += " canvas.addEventListener('click', function(event) {";

webpage += " handleEvent(event.clientX, event.clientY, true);";

webpage += " }, false);";

webpage += " canvas.addEventListener('touchmove', function(event){";

webpage += " handleEvent(event.touches[0].clientX, event.touches[0].clientY);";

webpage += "}, false);";

webpage += " function resizeCanvas() {";

webpage += " canvas.width = window.innerWidth;";

webpage += " canvas.height = window.innerHeight;";

webpage += " drawCanvas();";

webpage += " }";

webpage += " window.addEventListener('resize', resizeCanvas, false);";

webpage += " resizeCanvas();";

webpage += " drawCanvas();";

webpage += " document.ontouchmove = function(e) {e.preventDefault()};";

webpage += " })();";

webpage += "</script><html>\r\n";

server.send(200, "text/html", webpage);

}

void testRGB() { // fade in and out of Red, Green, Blue

analogWrite(R, 1023); // R off

analogWrite(G, 1023); // G off

analogWrite(B, 1023); // B off

fade(R); // R

fade(G); // G

fade(B); // B }

void fade(int pin) {

for (int u = 0; u < 1024; u++) {

analogWrite(pin, 1023 - u);

delay(1);

}

for (int u = 0; u < 1024; u++) {

analogWrite(pin, u);

delay(1);

} }

void handleNotFound() {

//digitalWrite(led, 1);

String message = "File Not Found\n\n";

message += "URI: ";

message += server.uri();

message += "\nMethod: ";

message += (server.method() == HTTP\_GET) ? "GET" : "POST";

message += "\nArguments: ";

message += server.args();

message += "\n";

for (uint8\_t i = 0; i < server.args(); i++) {

message += " " + server.argName(i) + ": " + server.arg(i) + "\n";

}

server.send(404, "text/plain", message);

//digitalWrite(led, 0); }

void setup(void) {

pinMode(R, OUTPUT);

pinMode(G, OUTPUT);

pinMode(B, OUTPUT);

analogWrite(R, 600); // R

analogWrite(G, 600); // G

analogWrite(B, 600); // B

Serial.begin(115200);

WiFi.begin(ssid, password);

Serial.println("");

// Wait for connection

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

if (MDNS.begin("esp8266")) {

Serial.println("MDNS responder started");

}

server.on("/", handleRoot);

testRGB();

server.onNotFound(handleNotFound);

server.begin();

Serial.println("HTTP server started");

}

void loop(void) {

server.handleClient();

}

1. Output

