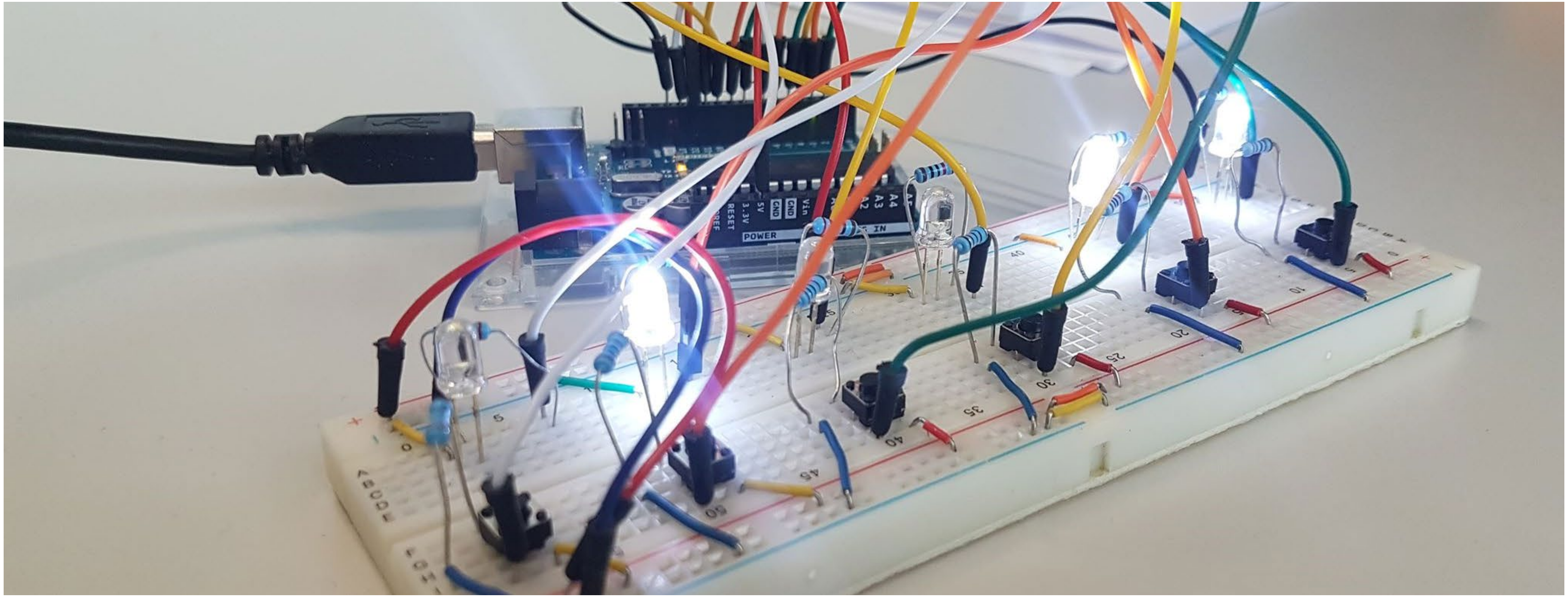
IAT 884 – Week 9 – Workshop 8

Alissa Antle and Annemiek Veldhuis (*ahv1@sfu.ca*)



IAT 884 – Tangible Technology

Wireless Communication

Workshop 8

Wireless Communication

Options with Arduino

Infrared

Pro's:
Cheap
DIY

Cons:
Directional – Must face receiver
Short range (10-meter max)
One way
Little information can be passed

Wireless Communication

Options with Arduino

XBees/Zigbee are small low-power digital *radios*.

Pros:
Multi-point networking
Mesh networks
Range: 10 to 100 meters but can be extended by adding more devices.
Low power
Secure networking

Cons:
Need atleast 3 devices: Zigbee coordinator, Zigbee router and end device
Expensive
Complex to configure

Wireless Communication

Options with Arduino

Radio communication

Pros:
Range: 10 to 150 meters depending on power output and environment.
Cheap
Widely popular, so many resources online
Reliable
Can broadcast to many receivers

Cons:
Slow data rate

Wireless Communication

Options with Arduino

Bluetooth

Pros:
Simple setup
Universal

Cons:
Can only connect 2 devices
Limited range: 10 meters max
Expensive
Size

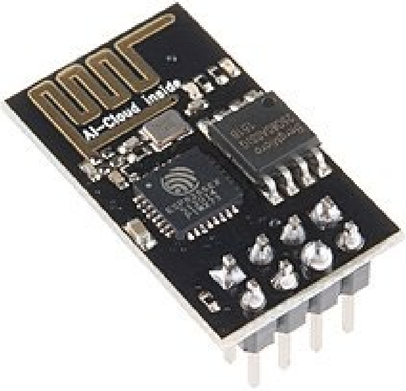
Wireless Communication

Options with Arduino

ESP8266 (WiFi)

Pros:
Range: Long
Cheap
Ability to connect your hardware to web-interfaces
Can connect lots of devices together (Mesh network)
Can work as router and receiver

Cons:
Can be more difficult to set up
Needs additional knowledge of web protocols



IAT 884 – Tangible Technology

Internet of Things

(ESP8266)

Workshop 8

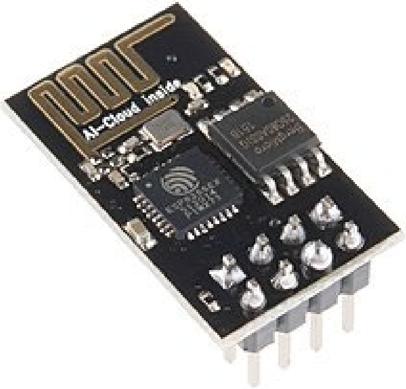
Internet of Things

The **Internet of Things** (IoT) refers to a system of interrelated, internet-connected objects that can collect and transfer data over a wireless network without human intervention

Internet of Things

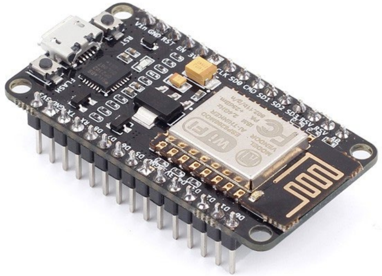
The ESP8266 was released in late 2014. It consists out of a microcontroller with 11 I/O pins and a WiFi transceiver.
Its cost was about 7 CAD, making it a perfect tool for makers.

Since then, many boards based on the ESP8266 chips were developed.



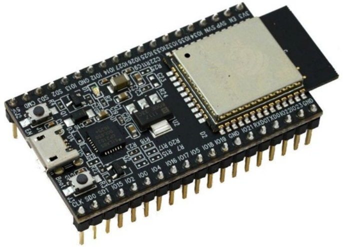
ESP8266

ESP-01

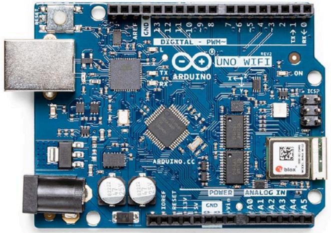


NodeMCU

Wemos D1 Mini



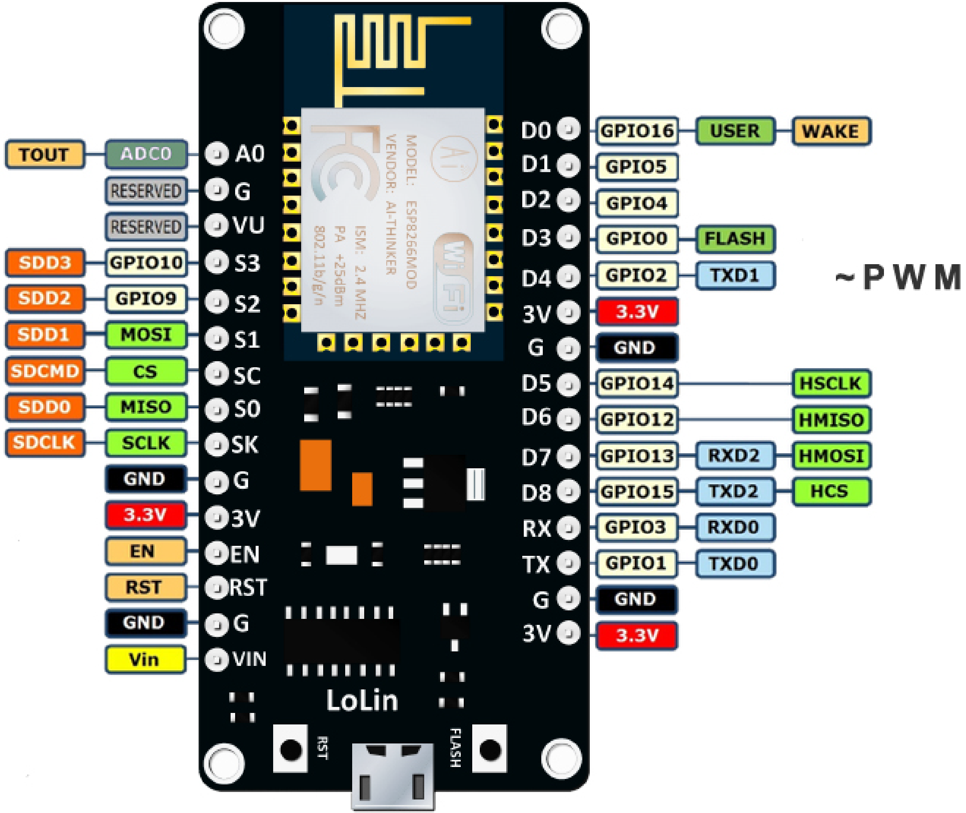
ESP-32



Arduino UNO WiFi

NodeMCU

Pinout



NodeMCU

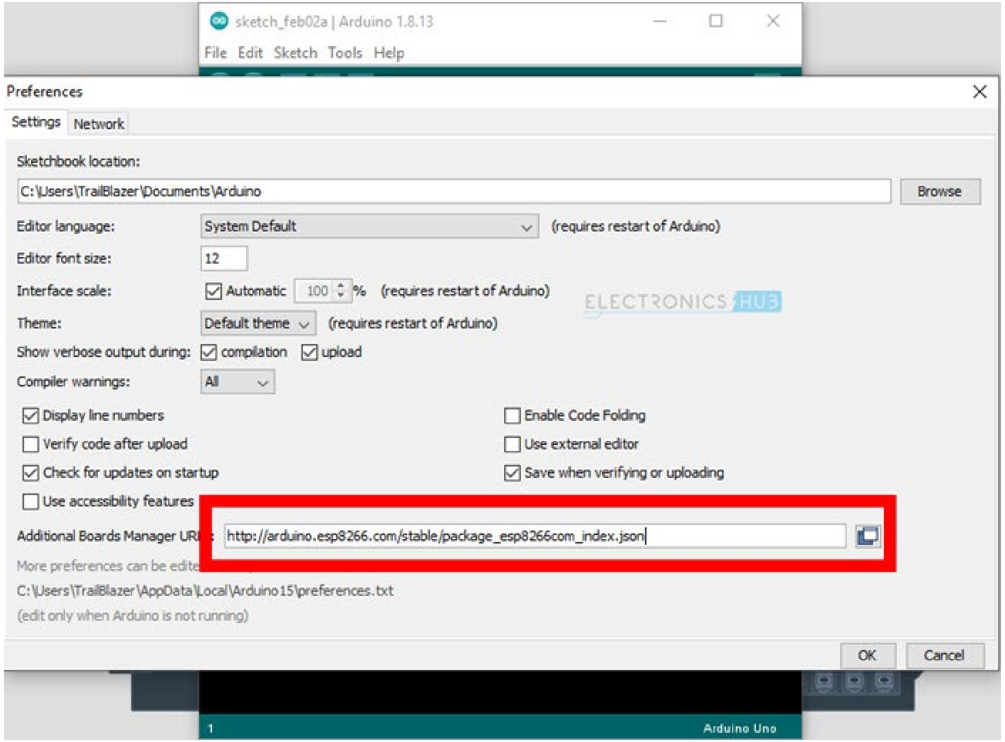
Prepare the Arduino IDE

Go to **File -> Preferences**

Paste in *Additional Boards Manager URLs*:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

(Make sure your laptop is connected to the internet)



Sketchbook location: C:\Users\TrailBlazer\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☑ compilation ☑ upload

Compiler warnings: All

☑ Display line numbers

□ Verify code after upload

☑ Check for updates on startup

□ Use accessibility features

□ Enable Code Folding

□ Use external editor

☑ Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

C:\Users\TrailBlazer\AppData\Local\Arduino15\preferences.txt

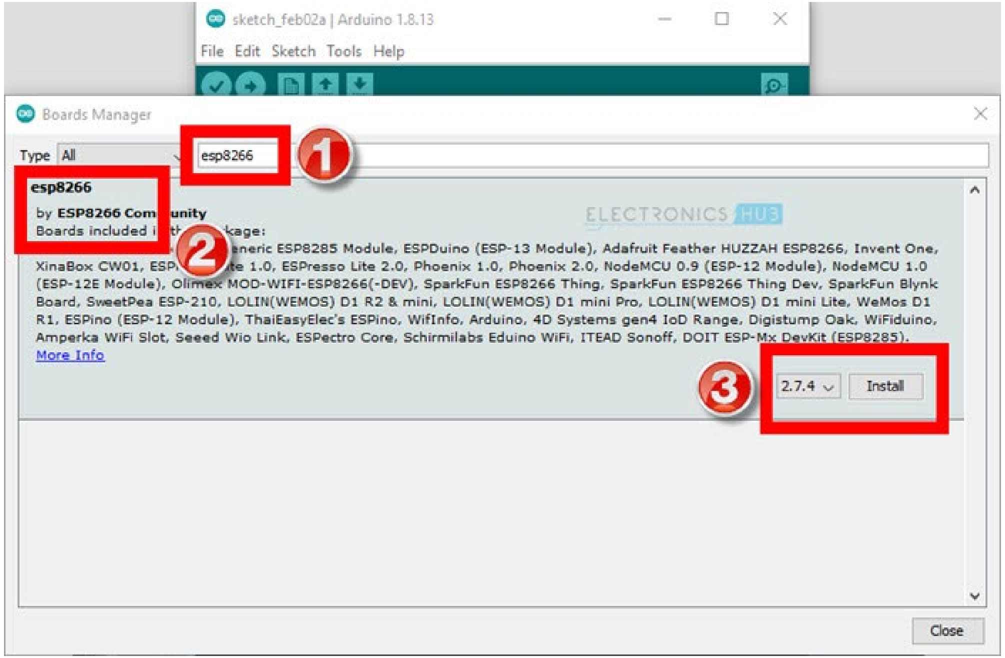
(edit only when Arduino is not running)

NodeMCU

Prepare the Arduino IDE

Go to **Tools -> Board -> Board Manager**

Search for “*esp8266*”



Type All esp8266

esp8266

by **ESP8266 Com unity**
Boards included i th kage:
eneric ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESP te 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).
More Info

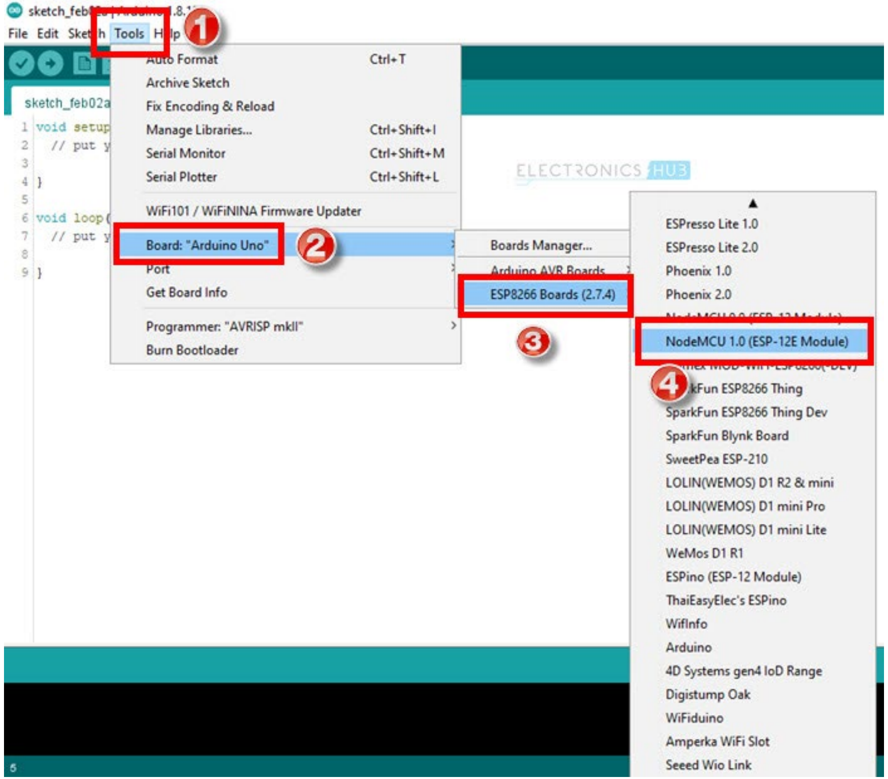
2.7.4 Install

NodeMCU

Prepare the Arduino IDE

Go to **Tools -> Board -> ESP8266 Boards**

Select *NodeMCU 1.0 ESP-12E Module*



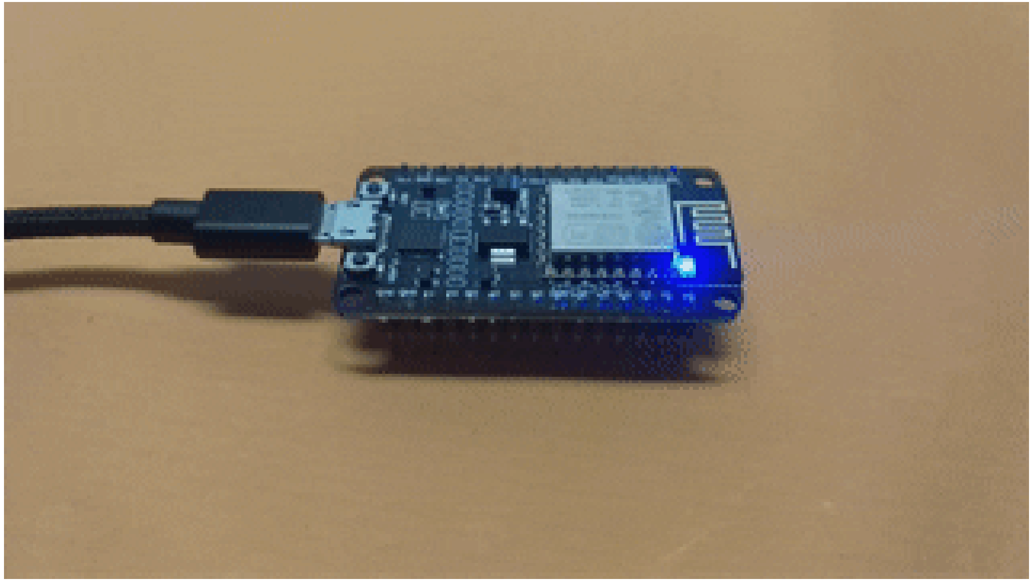
NodeMCU

Test the board connection

```
#define ledpin1 2
#define ledpin2 16

void setup(){
        pinMode(ledPin1, OUTPUT);
        pinMode(ledPin2, OUTPUT);
}

void loop(){
        digitalWrite(ledPin1, LOW);
        digitalWrite(ledPin2, HIGH);
        delay(1000);
        digitalWrite(ledPin1, HIGH);

}
```

NodeMCU

Test the board connection

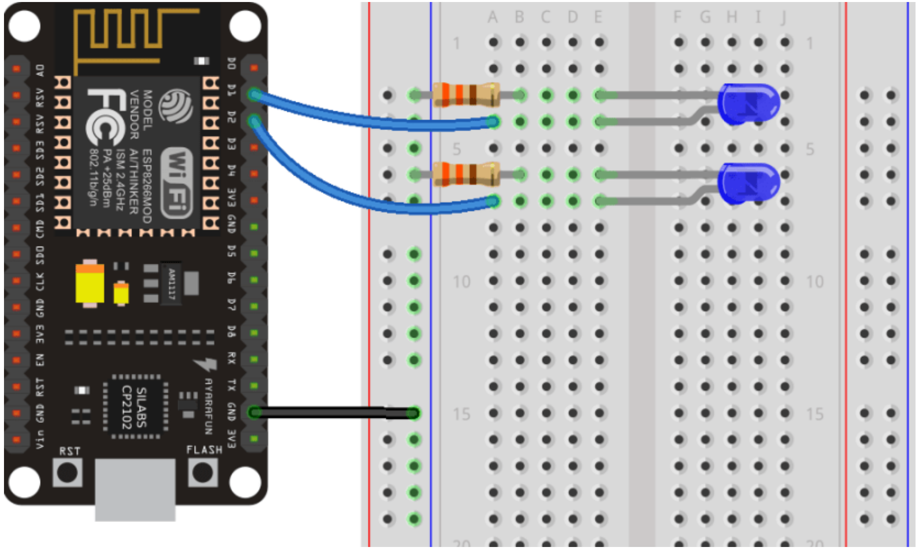


NodeMCU

Webserver

Control hardware through a website that runs off your NodeMCU

Step 1: Connect 2 LEDS to pin D1 and D2

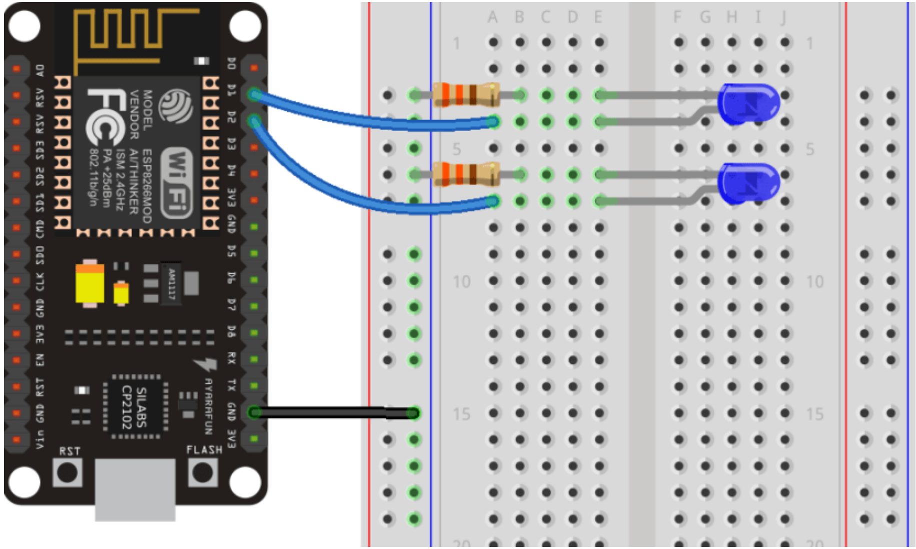


NodeMCU

Webserver

Control hardware through a website that runs off your NodeMCU

Step 2: Download the code from the wiki



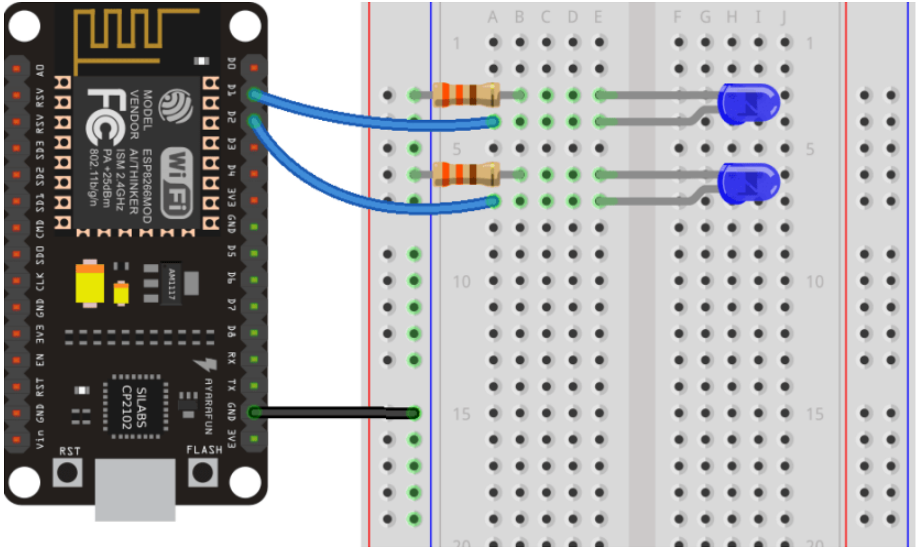
NodeMCU

Webserver

Control hardware through a website that runs off your NodeMCU

Step 3: Connect the NodeMCU to your WiFi network by changing two lines in the code

```
// Replace with your network
credentials
const char* ssid = "";
const char* password = "";
```

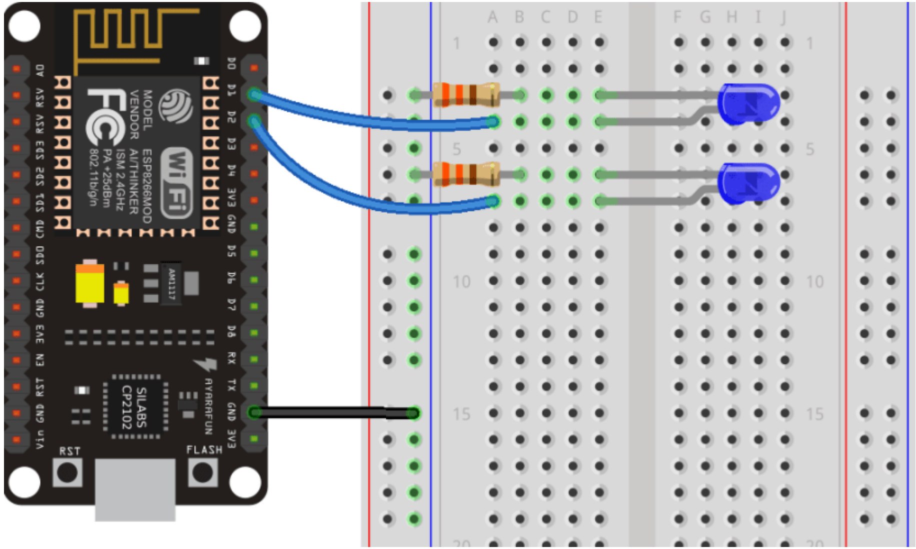


NodeMCU

Webserver

Control hardware through a website that runs off your NodeMCU

Step 4: Upload code to board

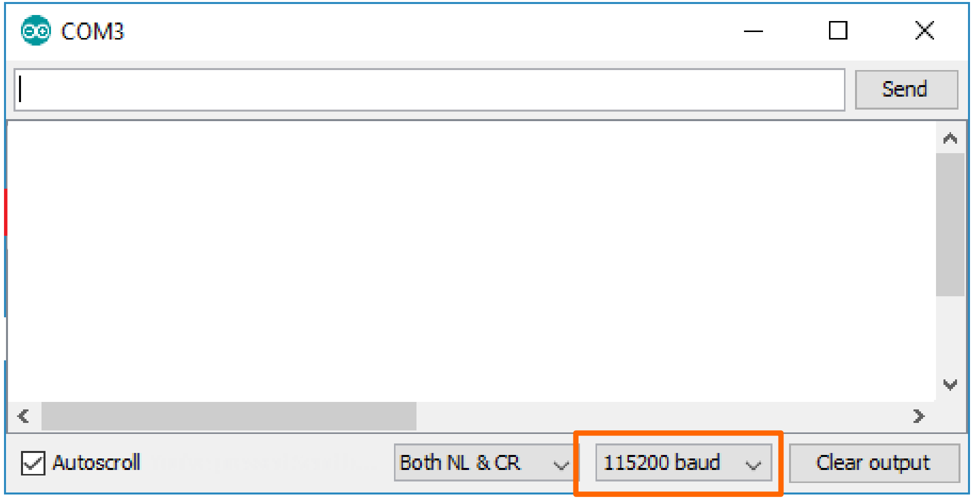


NodeMCU

Webserver

Control hardware through a website that runs off your NodeMCU

Step 5: Open the Serial Monitor and set the baud rate to 115200



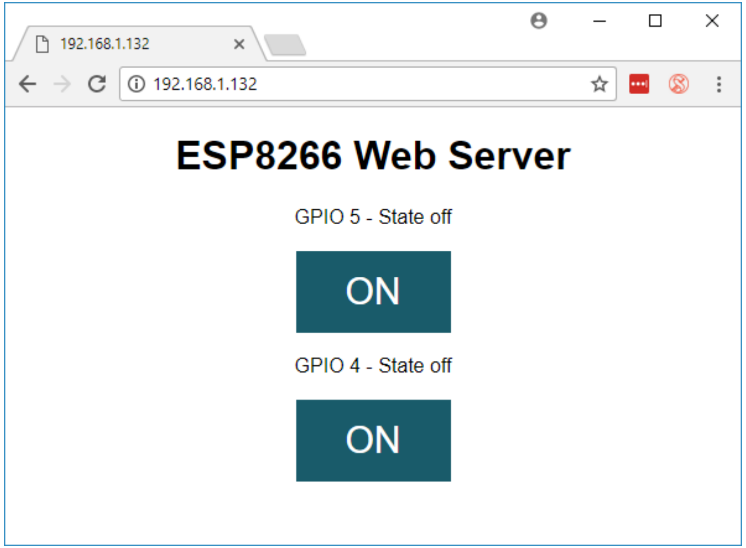
NodeMCU

Webserver

Control hardware through a website that runs off your NodeMCU

Step 7: Copy the IP address in your browser window.

This page is sent by the ESP8266 when you make a request on the ESP IP address.



ESP8266 Web Server

GPIO 5 - State off

GPIO 4 - State off

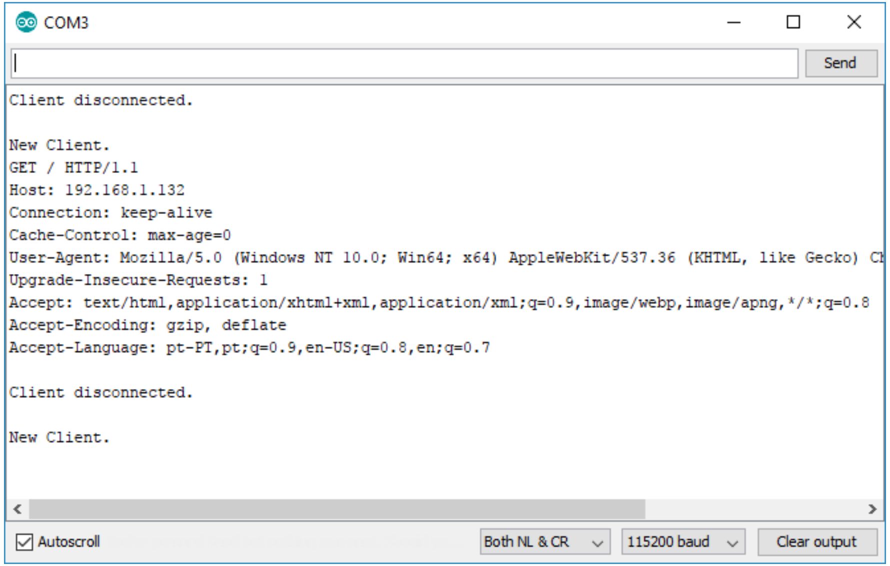
NodeMCU

Webserver

Control hardware through a website that runs off your NodeMCU

Step 8: Look at the serial monitor to see what is going on.

The ESP receives an HTTP request from a new client – in this case, your browser.



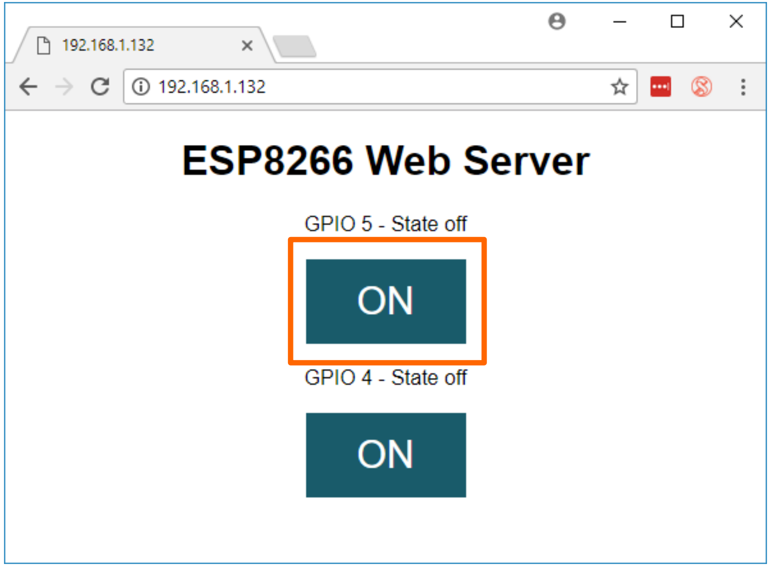
```
Client disconnected.

New Client.
GET / HTTP/1.1
Host: 192.168.1.132
Connection: keep-alive
Cache-Control: max-age=0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7

Client disconnected.

New Client.
```

NodeMCU

Webserver

Control hardware through a website that runs off your NodeMCU

Step 9: Click on the button to turn GPIO 5 on. Look at the LEDs and the serial monitor.



ESP8266 Web Server

GPIO 5 - State off

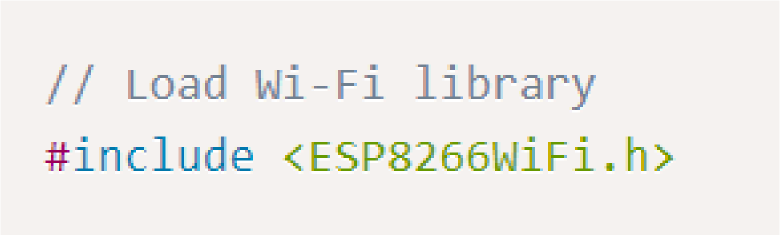
GPIO 4 - State off

NodeMCU

Webserver

Code:

Include the ESP WiFi library



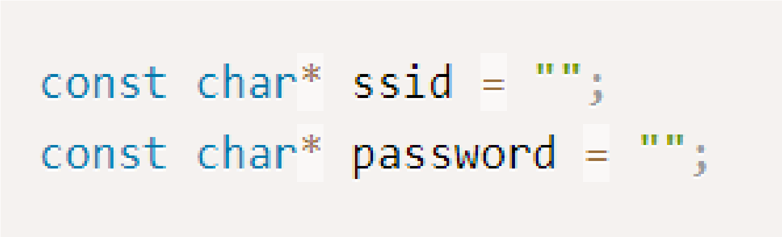
```
// Load Wi-Fi library
#include <ESP8266WiFi.h>
```

NodeMCU

Webserver

Code:

insert your ssid and password



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

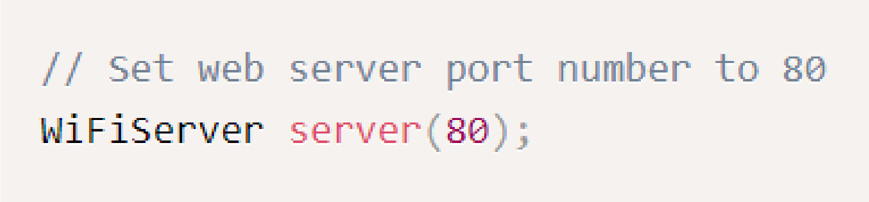
NodeMCU

Webserver

Code:

Then, you set your web server to port 80.

Port 80 is one of the most used port numbers. Any Web/HTTP client, such as a Web browser, uses port 80 to send and receive requested Web pages from a HTTP server.



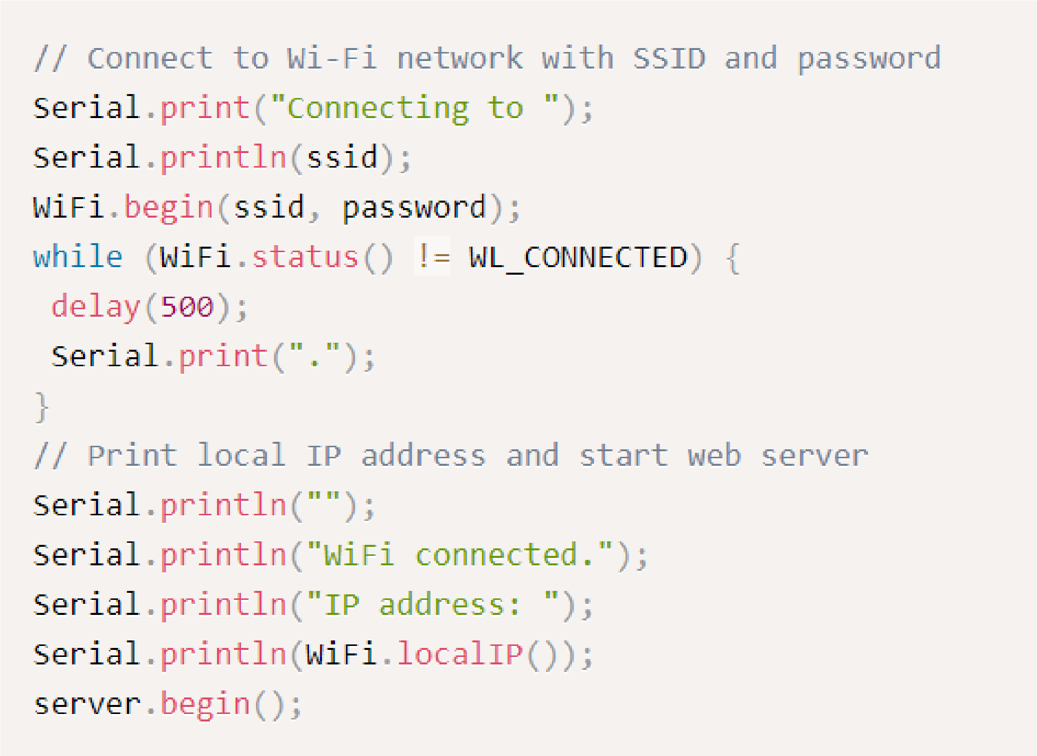
```
// Set web server port number to 80
WiFiServer server(80);
```

NodeMCU

Webserver

Code:

The following lines begin the Wi-Fi connection, wait for a successful connection and prints the ESP IP address in the Serial Monitor.



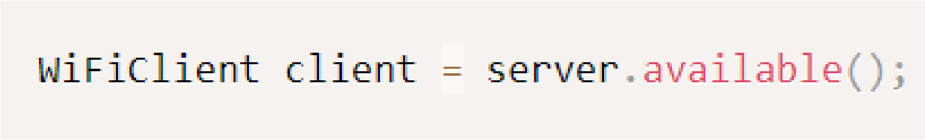
```
// Connect to Wi-Fi network with SSID and password
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
 delay(500);
 Serial.print(".");
}
// Print local IP address and start web server
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
server.begin();
```

NodeMCU

Webserver

Code:

The ESP is always listening for incoming clients.



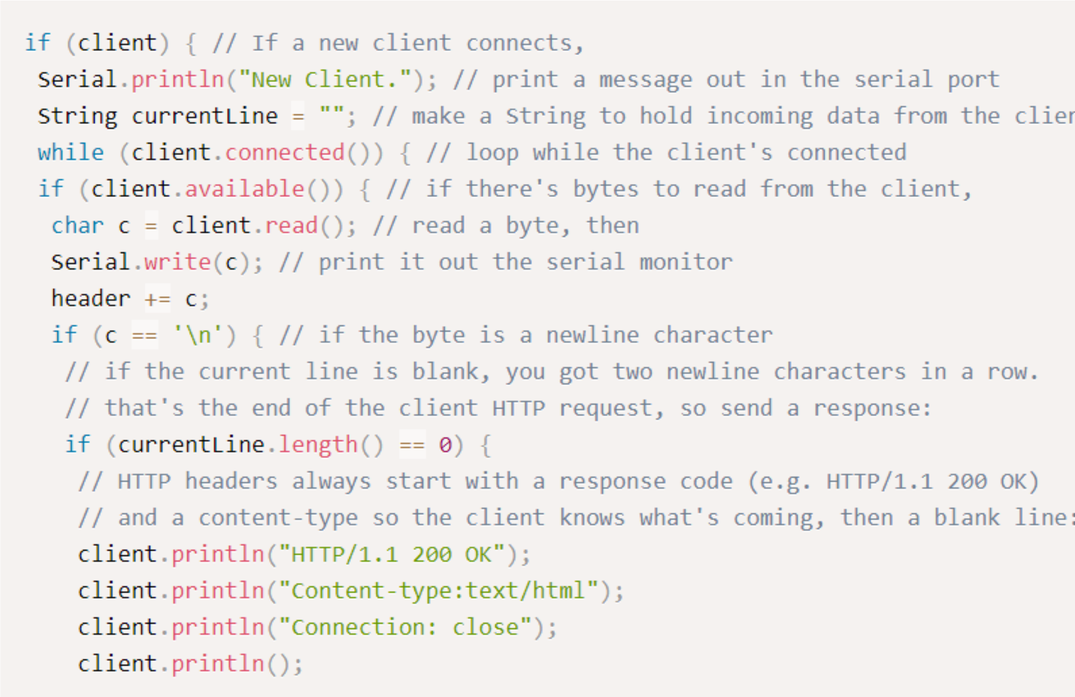
```
WiFiClient client = server.available();
```

NodeMCU

Webserver

Code:

When a request is received from a client, we'll save the incoming data. The while loop that follows will be running if the client stays connected.



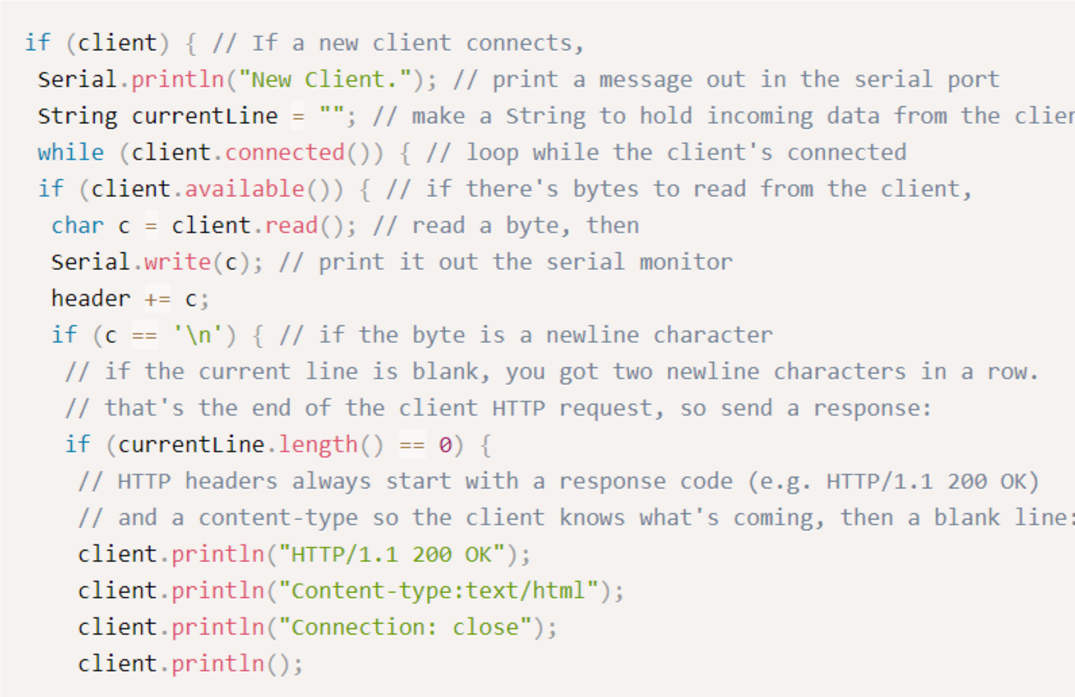
```
if (client) { // If a new client connects,
 Serial.println("New Client."); // print a message out in the serial port
 String currentLine = ""; // make a String to hold incoming data from the clier
 while (client.connected()) { // loop while the client's connected
 if (client.available()) { // if 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 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:
   if (currentLine.length() == 0) {
    // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
    // and a content-type so the client knows what's coming, then a blank line:
    client.println("HTTP/1.1 200 OK");
    client.println("Content-type:text/html");
    client.println("Connection: close");
    client.println();
```

NodeMCU

Webserver

Code:

When a request is received from a client, we'll save the incoming data. The while loop that follows will be running if the client stays connected.



```
if (client) { // If a new client connects,
 Serial.println("New Client."); // print a message out in the serial port
 String currentLine = ""; // make a String to hold incoming data from the clien
 while (client.connected()) { // 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 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:
   if (currentLine.length() == 0) {
    // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
    // and a content-type so the client knows what's coming, then a blank line:
    client.println("HTTP/1.1 200 OK");
    client.println("Content-type:text/html");
    client.println("Connection: close");
    client.println();
```

IAT 884 – Tangible Technology

Networking

(ESP8266)

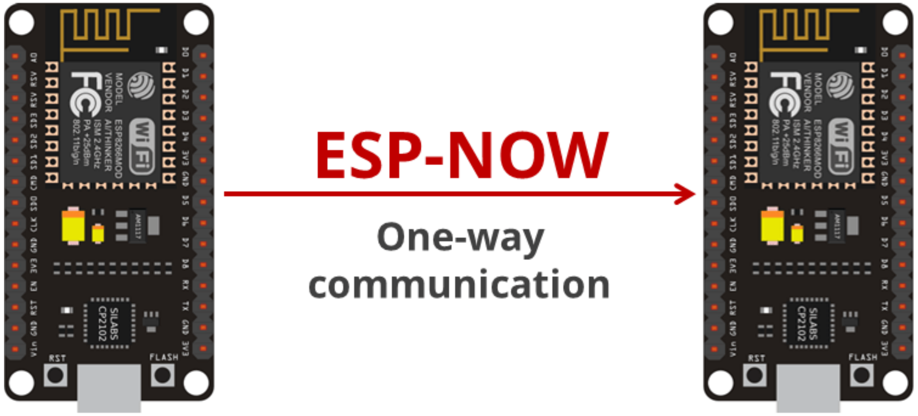
Workshop 8

Networking

ESP-NOW

One ESP8266 board sending data to another ESP8266 board.

Easy to implement
Send sensor readings or ON/OFF commands

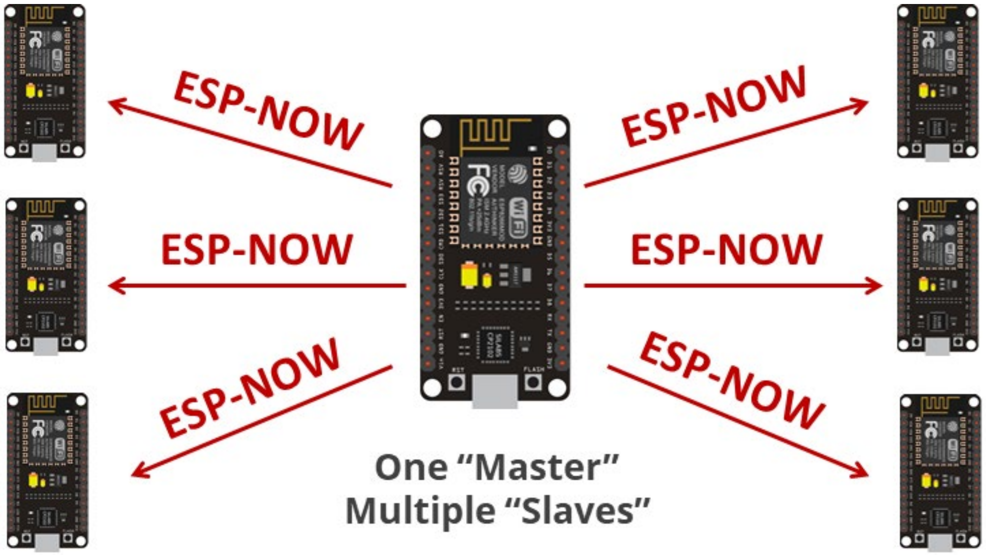


Networking

ESP-NOW

A “master” ESP8266 sending data to multiple ESP8266 “slaves”

Home automation remote control

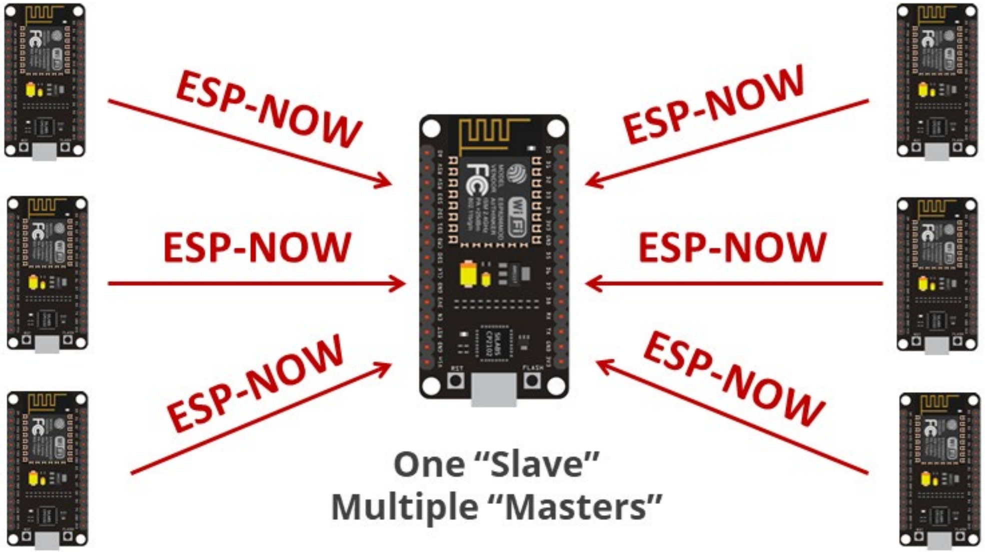


Networking

ESP-NOW

One ESP8266 “slave” receiving data from multiple “masters”

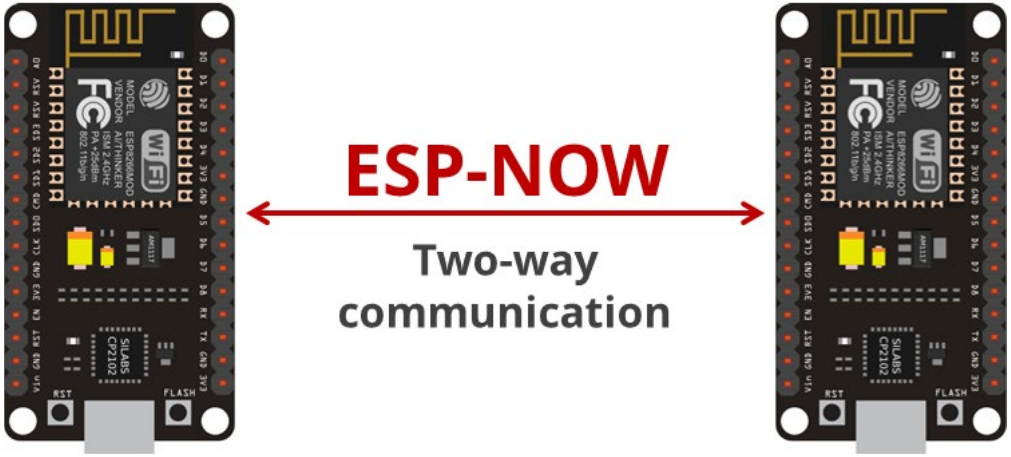
Collect data from multiple sensor nodes and display them through 1 webserver



Networking

ESP-NOW

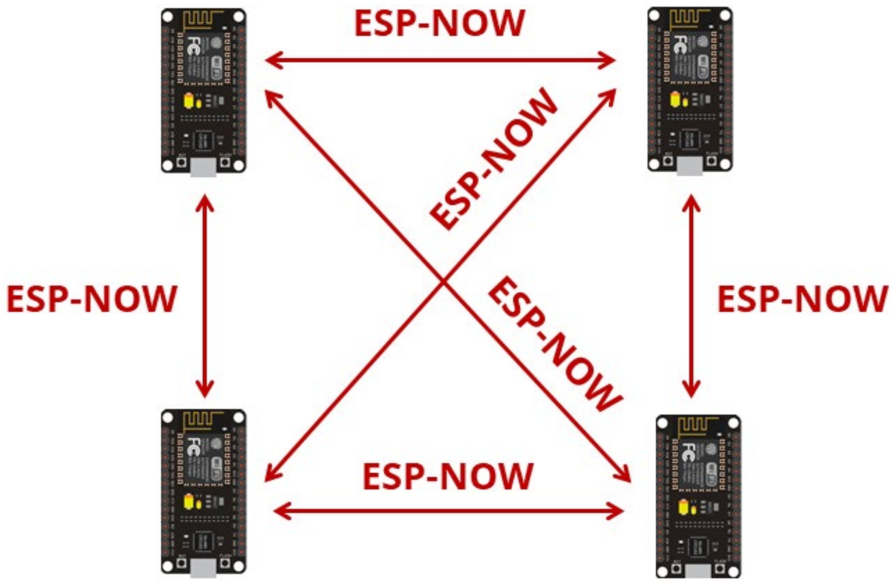
Two boards communicating with each other. Each board is a sender and receiver.



Networking

ESP-NOW

Create a Star or Mesh network.


ESP-NOW
ESP-NOW
ESP-NOW
ESP-NOW
ESP-NOW
ESP-NOW