ESP 8266 WORKSHOP W/ ARDUINO

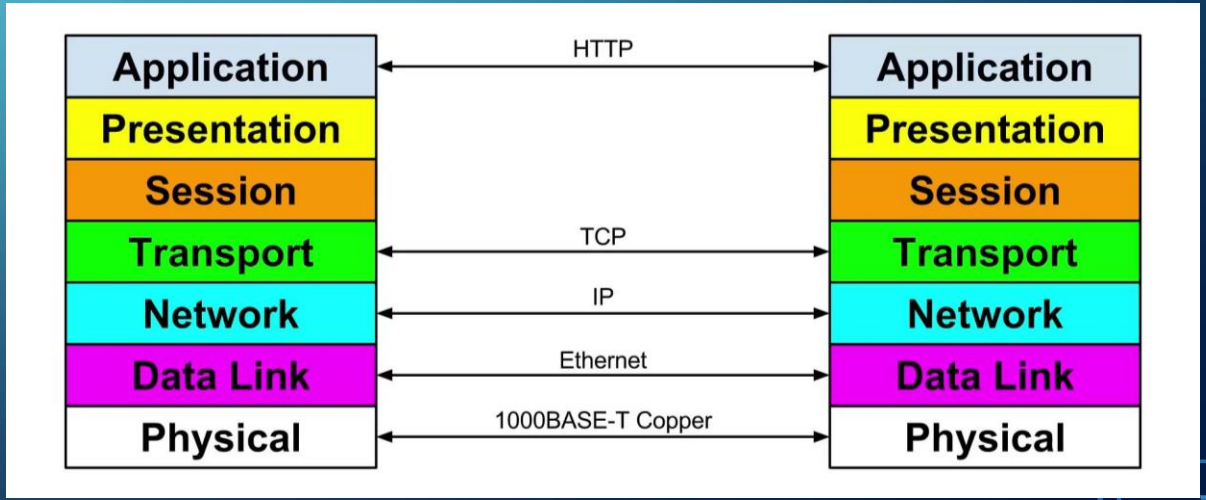
BY IEEE SUTD

ESP 8266 - NODEMCU

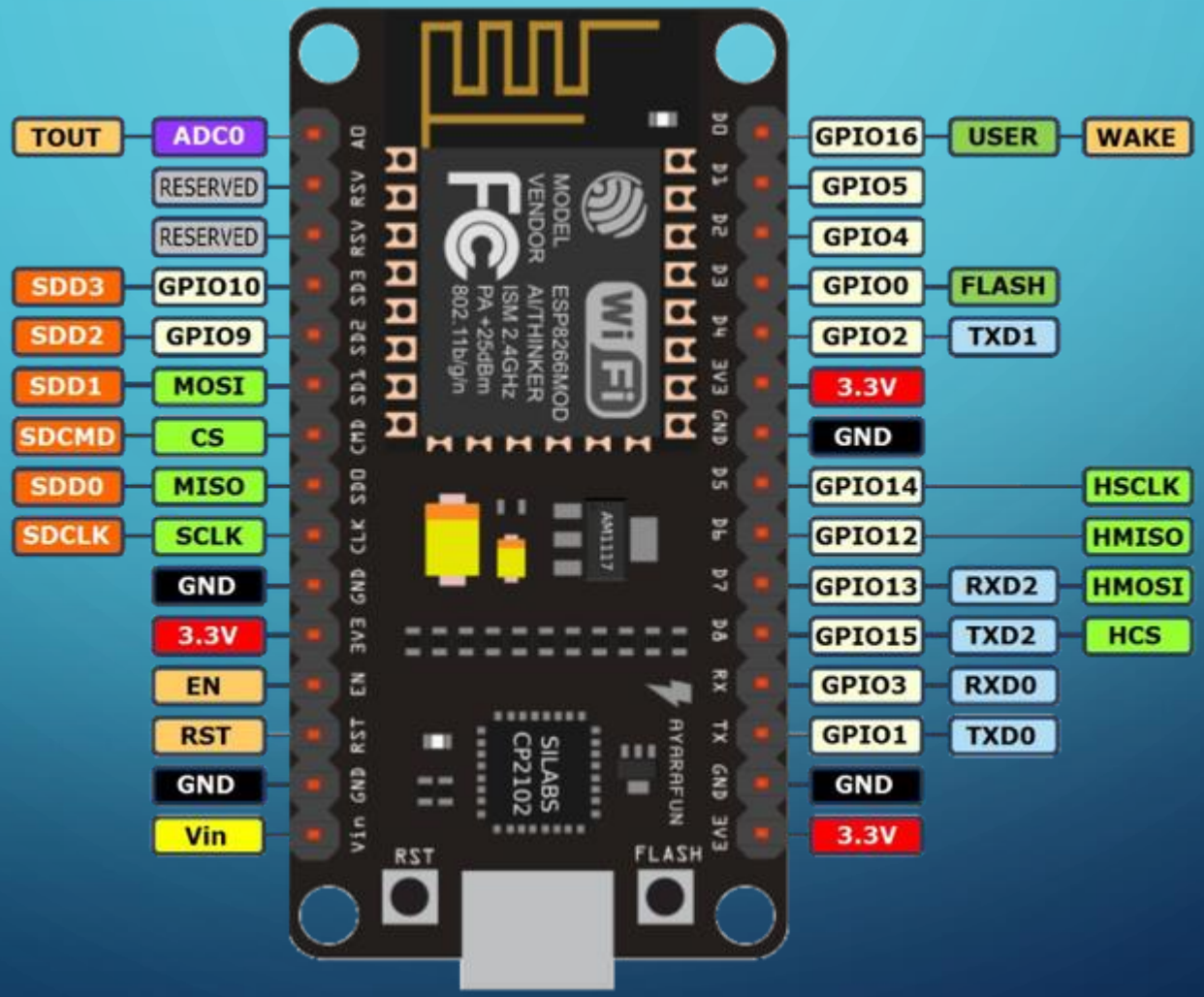
- A SOC with WiFi built-in
- Great for IOT and communication with the Internet
- Really cheap!
  - Makes it awesome for hacking & mischief

NETWORKS - OVERVIEW

- HTTP: Hypertext Tranfer Protocol
  - Usually used over TCP
  - Conforms to URI:
    `scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]`
- TCP: Transmission Control Protocol
  - A Transport Layer Protocol used to transfer data



NODEMCU - PINOUT



When you access the pins through Arduino, they refer to the GPIO pins not the D-pins. Instead, access them through

ADDING ARDUINO SUPPORT FOR ESP 8266

- Download its driver from this link CH341SER.zip
- Download Arduino IDE.
- Start Arduino and open Preferences window.
- Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into Additional
- Board Manager URLs field.
- Open Boards Manager from Tools
- Enter esp8266 intro the search field to install ESP8266 platform
- Go to Tools > Board menu, then select your ESP8266 board.
- Go to Tools > Port > “COM_” OR tty_

ESP 8266 DOCUMENTATION

- https://arduino-esp8266.readthedocs.io/en/latest/reference.html
- Libraries: https://arduino-esp8266.readthedocs.io/en/latest/libraries.html
- File System: https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html
- Using ESP WiFi as Client:
  https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/client-examples.html
- Using ESP WiFi as AP:
  https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/soft-access-point-examples.html

EXERCISE 1:
CONNECT TO SCHOOL WIFI AND PRINT IP ADDRESS

```
#include <ESP8266WiFi.h>

//WiFi parameters to be configured:
const char* ssid = "YOUR WIFI NETWORK NAME";
const char* password = "YOUR WIFI PASSWORD";
```

```
void setup() {
  Serial.begin(9600);
  //Connect to WiFi
  WiFi.begin(ssid, password);

  //while wifi not conencted yet, print ','
  //after connected, get out of loop
  while(WiFi.status()!= WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }

  //print a new line, then tell you it's conencted + IP address
  Serial.println("");
  Serial.println("WiFi connected");
  //print the IP address
  Serial.print(WiFi.localIP());

}

void loop() {
  // nada

}
```

GET & POST REQUESTS

- **GET**
  - Safe & Idempotent
    - Same parameters = same results (regardless of number of repeated times)
  - Retrieve info
- **POST**
  - Request that URI does something with provided entity
  - Used to create new entities

HOSTING A SERVER ON YOUR ESP: BOILERPLATE CODE

```
//configuring
#include <ESP8266WiFi.h>  //this is the library you'll need for esp stuff
WiFiServer server(80); //Initialize the server on Port 80 (Global instance)

//setting up
void setup() {
  //boilerplate code
  WiFi.mode(WIFI_AP); //Our ESP8266-12E is an AccessPoint
  WiFi.softAP("Hello_IOT", "123456789"); // Provide the (SSID, password);
  server.begin(); // Start the HTTP Server
```

CONNECT TO YOUR SERVER!

- Open the "*Connect to network*" window. You should see your server with SSID "*Hello_IOT*" in the list. Select the Hello_IOT network, provide the *password/passphrase* and save it.

GET SERVER INFORMATION

Add the following code to the end of the setup() function

```
//setting up
void setup() {
  //boilerplate code
  WiFi.mode(WIFI_AP); //Our ESP8266-12E is an AccessPoint
  WiFi.softAP("Hello_IOT", "123456789"); // Provide the (SSID, password);
  server.begin(); // Start the HTTP Server

  //peeking under the hood: this is how we know what's going on
  //IMPORTANT: serial output baud rate has to match the one you manually select for your serial monitor
  Serial.begin(9600); //Start communication between the ESP8266-12E and the monitor window
  IPAddress HTTPS_ServerIP= WiFi.softAPIP(); // Obtain the IP of the Server
  Serial.print("Server IP is: "); // Print the IP to the monitor window
  Serial.println(HTTPS_ServerIP);
```

TALK TO YOUR SERVER

Enter the following code within the loop() function:

```
//this code gets run ALWAYS
void loop() {

  WiFiClient client = server.available();
  if (!client) {
    return; //keep searching if server isn't available!!
  }

  //Looking under the hood- so we're kept in the loop (heheh)
  Serial.println("Somebody has connected :)");
```

Compile and load to the ESP8266-E12.

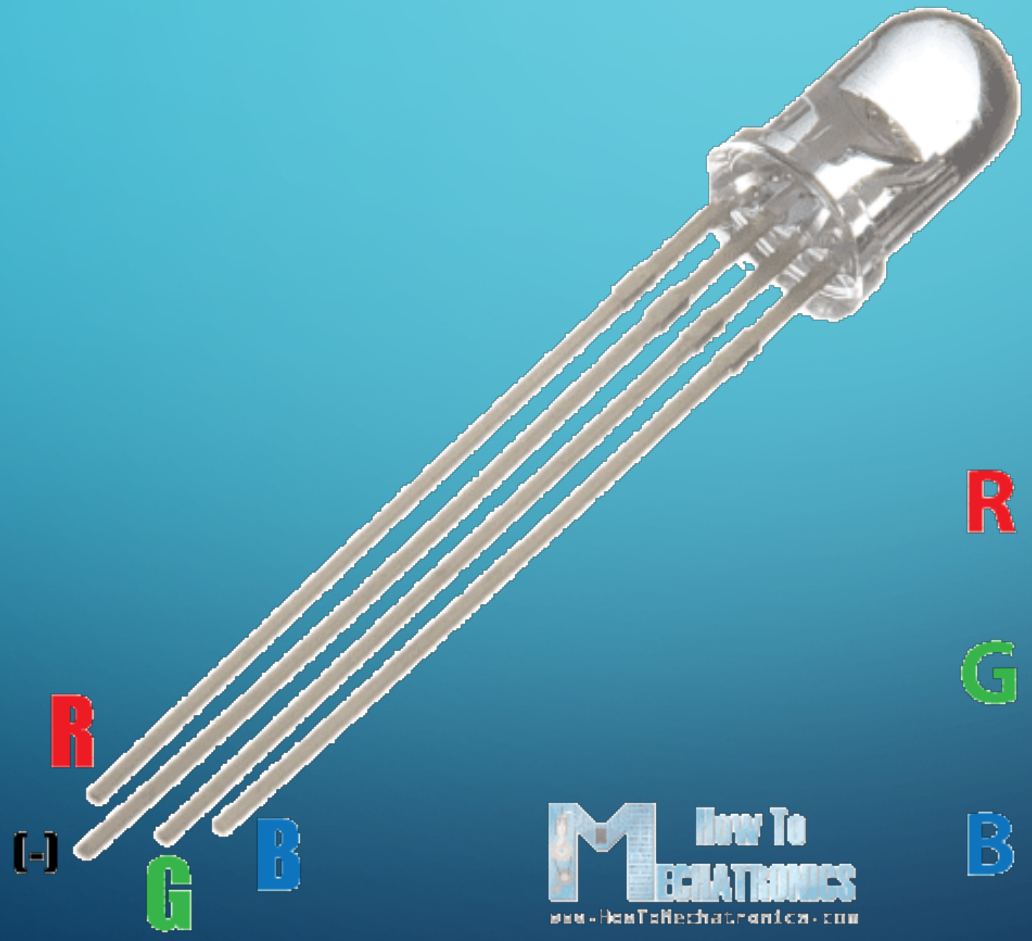
Open a browser window and enter http://192.168.4.1 and and hit enter.

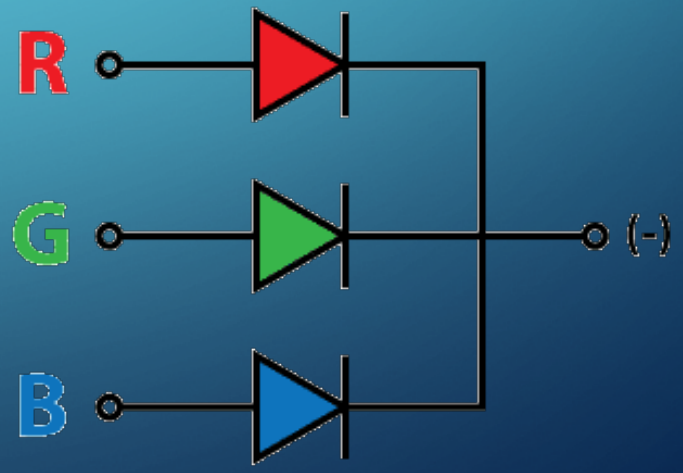
Observe your Monitor window to check for a connection.

```
//this code gets run ALWAYS
void loop() {

  WiFiClient client = server.available();
  if (!client) {
    return; //keep searching if server isn't available!!
  }

  //Looking under the hood- so we're kept in the loop (heheh)
  Serial.println("Somebody has connected :)");

  //Read what the browser has sent into a String class and print the request to the monitor
  String request = client.readStringUntil('\r');
  //Looking under the hood
  Serial.println(request);
```

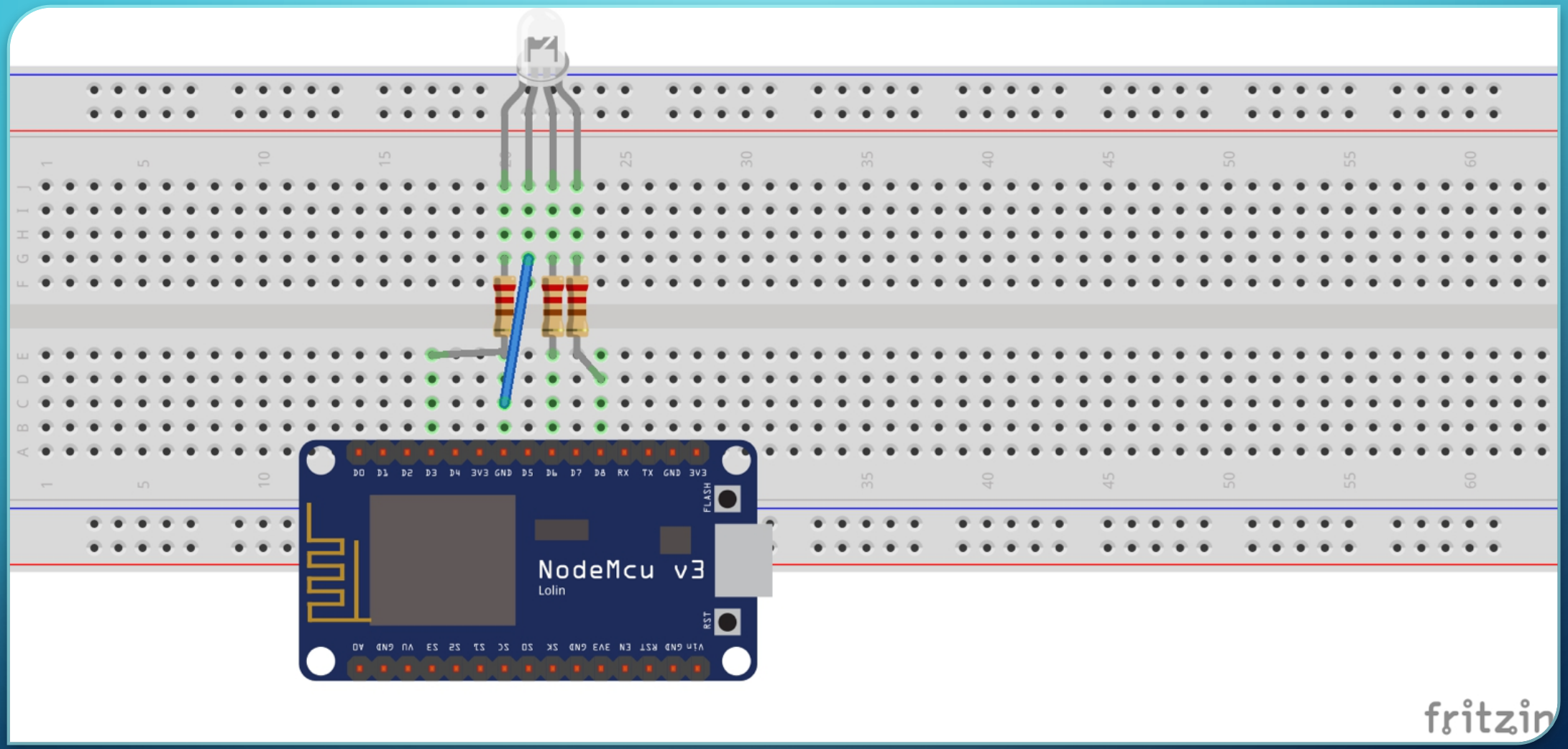
Add this to the end of the loop so you can read what the browser is sending you:

CONTROLLING LED VIA WEB BROWSER





Color	HTML/CSS Name	#RRGGBB	(R,G,B)
	Black	#000000	(0,0,0)
	White	#FFFFFF	(255,255,255)
	Red	#FF0000	(255,0,0)
	Lime	#00FF00	(0,255,0)
	Blue	#0000FF	(0,0,255)
	Yellow	#FFFF00	(255,255,0)
	Cyan / Aqua	#00FFFF	(0,255,255)
	Magenta / Fuchsia	#FF00FF	(255,0,255)
	Silver	#C0C0C0	(192,192,192)
	Gray	#808080	(128,128,128)
	Maroon	#800000	(128,0,0)
	Olive	#808000	(128,128,0)
	Green	#008000	(0,128,0)
	Purple	#800080	(128,0,128)
	Teal	#008080	(0,128,128)
	Navy	#000080	(0,0,128)


D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
NodeMcu v3
Lolin
RST
A0 GND VU S3 S2 S1 SC SD SK GND 3V3 EN RST GND Vin
fritzin

Add to top of program:

```
//configuring
#include <ESP8266WiFi.h>  //this is the library you'll need for esp stuff
WiFiServer server(80); //Initialize the server on Port 80 (Global instance)
//led config
int redPin = 0;
int greenPin = 12;
int bluePin = 15;
```

Add to bottom of program:

```
void setColour(int red, int green, int blue)
{

red = 255 - red;
green = 255 - green;
blue = 255 - blue;

analogWrite(redPin, red);
analogWrite(bluePin, blue);
analogWrite(greenPin, green);
}
```

At bottom of setup() function add:

```
//setting up
void setup() {
  //boilerplate code
  WiFi.mode(WIFI_AP); //Our ESP8266-12E is an AccessPoint
  WiFi.softAP("Hello_IOT", "123456789"); // Provide the (SSID, password);
  server.begin(); // Start the HTTP Server

  //peeking under the hood: this is how we know what's going on
  //IMPORTANT: serial output baud rate has to match the one you manually select for your serial monitor
  Serial.begin(9600); //Start communication between the ESP8266-12E and the monitor window
  IPAddress HTTPS_ServerIP= WiFi.softAPIP(); // Obtain the IP of the Server
  Serial.print("Server IP is: "); // Print the IP to the monitor window
  Serial.println(HTTPS_ServerIP);


  //here we set up the LED

  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
```

At bottom of loop() function:

```
//this code gets run ALWAYS
void loop() {

  WiFiClient client = server.available();
  if (!client) {
    return; //keep searching if server isn't available!!
  }

  //Looking under the hood- so we're kept in the loop (heheh)
  Serial.println("Somebody has connected :)");

  //Read what the browser has sent into a String class and print the request to the monitor
  String request = client.readStringUntil('\r');
  //Looking under the hood
  Serial.println(request);


  //for the LED: if someone tries to do something,
  // Handle the Request


  if (request.indexOf("/RED") != -1){
    setColour(255, 0, 0);
    Serial.println("I am red!");
    }
  else if (request.indexOf("/GREEN") != -1){
    setColour(0, 255, 0);
    Serial.println("I am green!");
    }
  else if (request.indexOf("/BLUE") != -1){
    setColour(0, 0, 255);
    Serial.println("I am blue!");
    }
```

In the address bar of your browse type the following URL:

http://192.168.4.1/RED

The LED on the ESP8266-E12 will turn RED.

Then type the following URL:

http://192.168.4.1/GREEN

The LED on the ESP8266-E12 will turn GREEN.

Then type the following URL:

http://192.168.4.1/BLUE

The LED on the ESP8266-E12 will turn BLUE.

RESOURCES

HTTP Server activity:

http://www.instructables.com/id/Programming-a-HTTP-Server-on-ESP-8266-12E/

Slightly more intermediate version of above activity:

https://www.allaboutcircuits.com/projects/how-to-make-an-interactive-tcp-server-nodemcu-on-the-esp8266/

AT COMMANDS

http://www.instructables.com/id/Get-Started-with-ESP8266-Using-AT-Commands-NodeMCU/

http://www.instructables.com/id/Get-Your-ESP8266-12-Ready-for-AT-Commands/\