IoT With ESP8266

Mr.Prarinya Ekapho[1]

AGENDA FOR 1 DAYS TRAINING

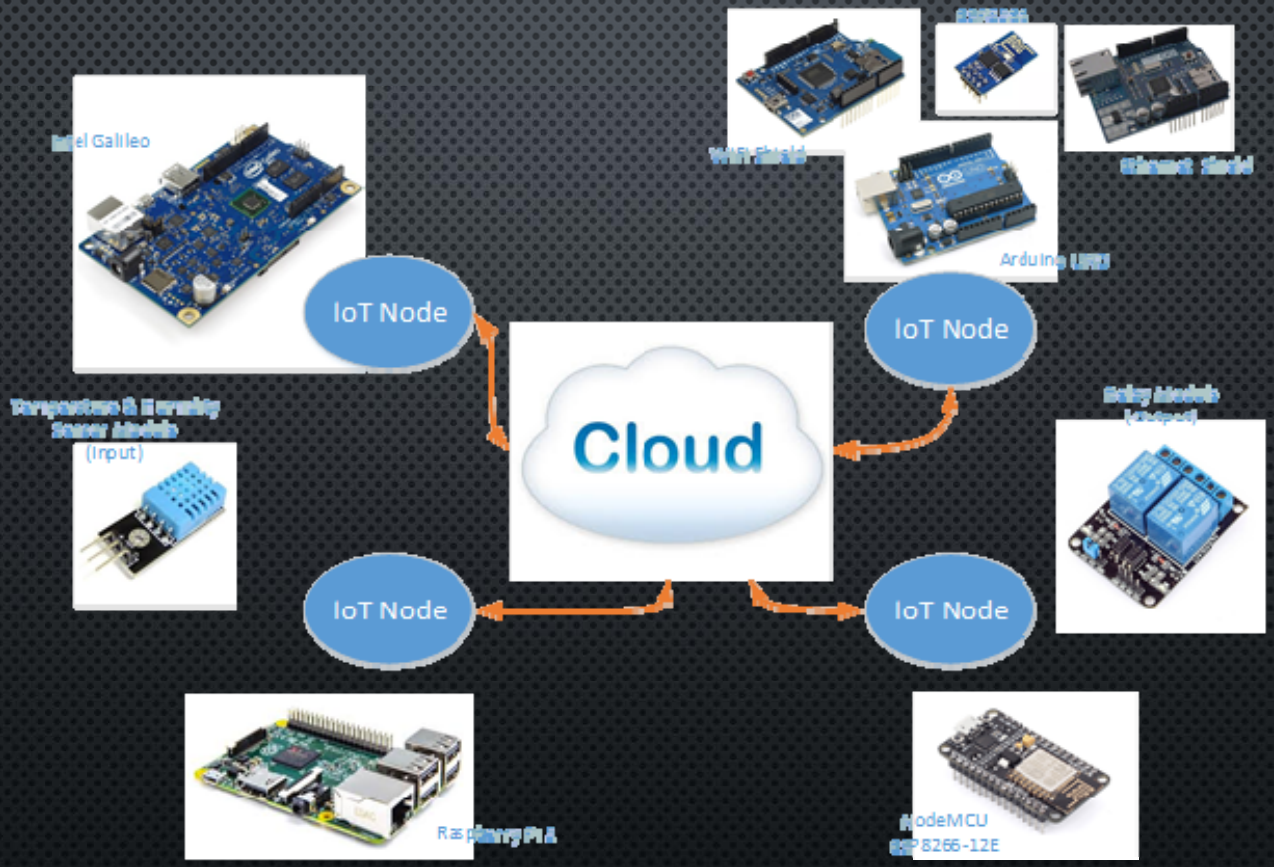
- IoT Concept and Applications
- Environment and Equipment of system
- Start with Developing tools kit
- Workshop#1 Test board NodeMCU V2
- Workshop#2 temperature and humidity sensor
- Workshop#3 Mini project IoT 2way switch control via Smart Phone

IoT Concept

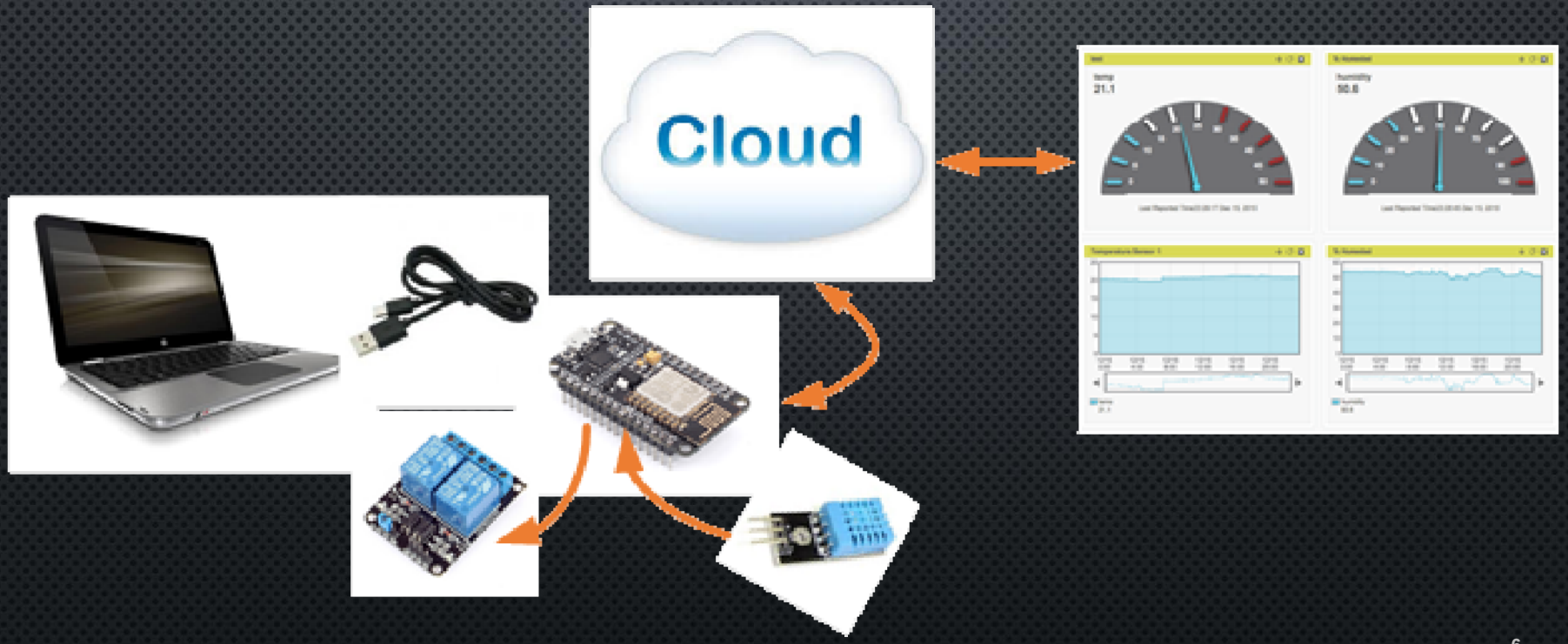
Internet of Things (IoT) Components

- Things
- Sensors & Controls I/O
- Internet connect devices
- Data
- Cloud Service

IoT Development guide



SYSTEM DESIGN CONCEPT



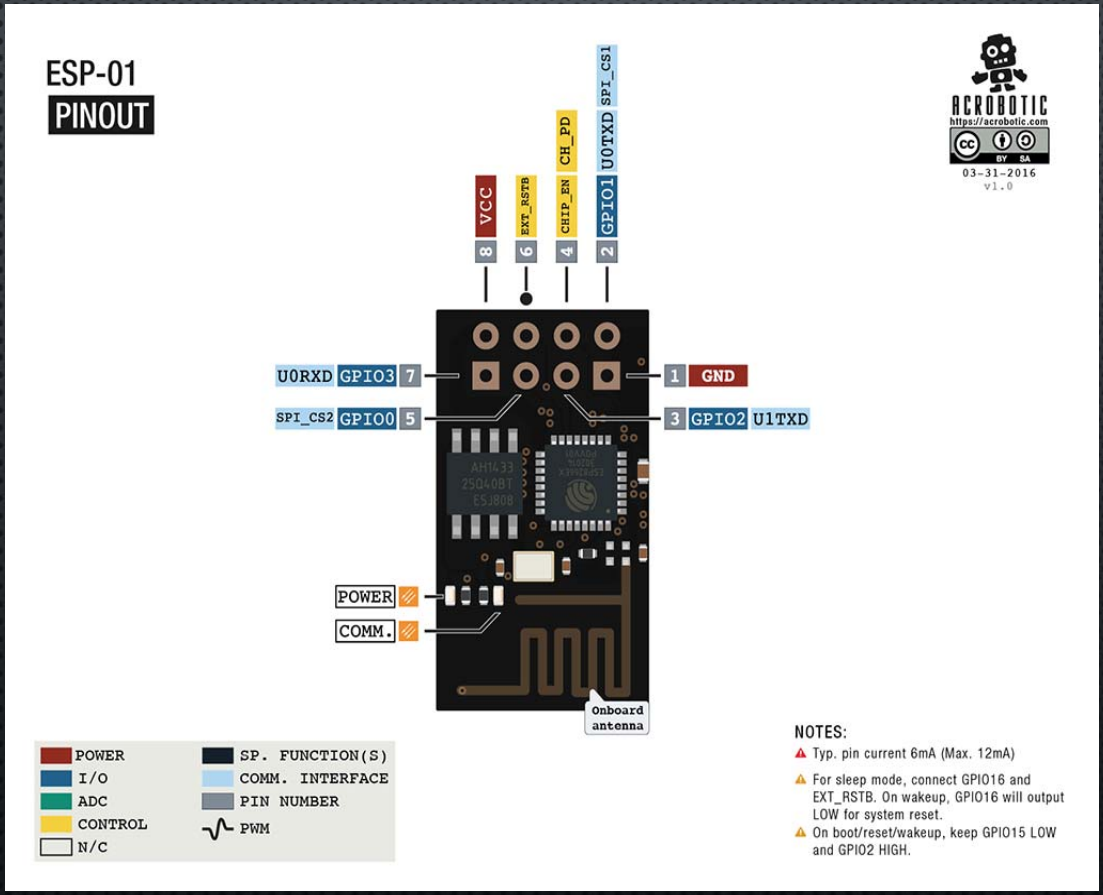
About NodeMCU ESP8266

ESP8266(ESP-01)

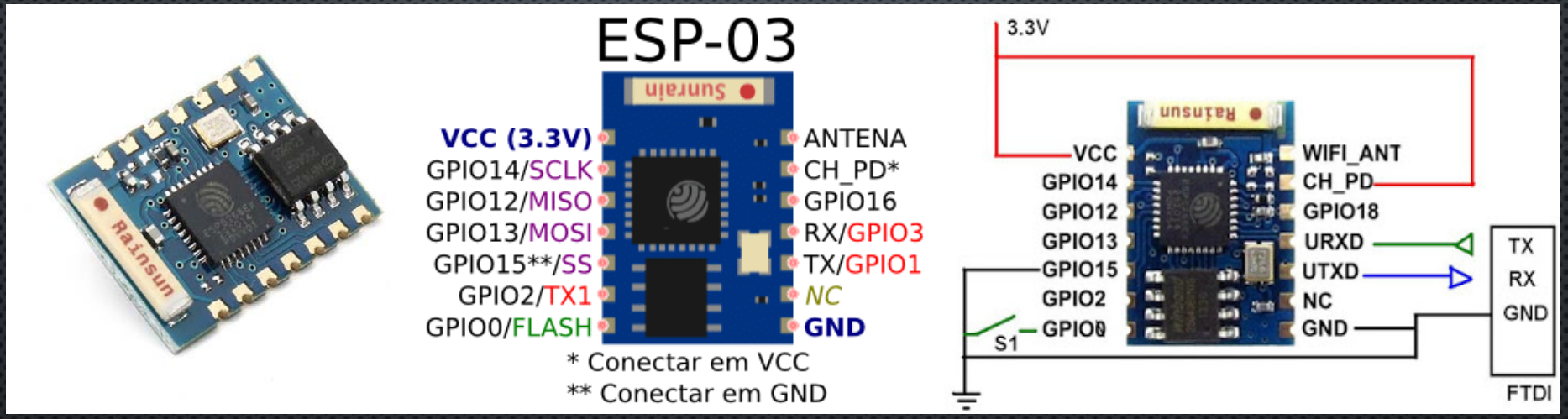
NodeMCU Devkit 0.9 (ESP-12) Version 1

NodeMCU Devkit 1.0 (ESP-12E) Version 2

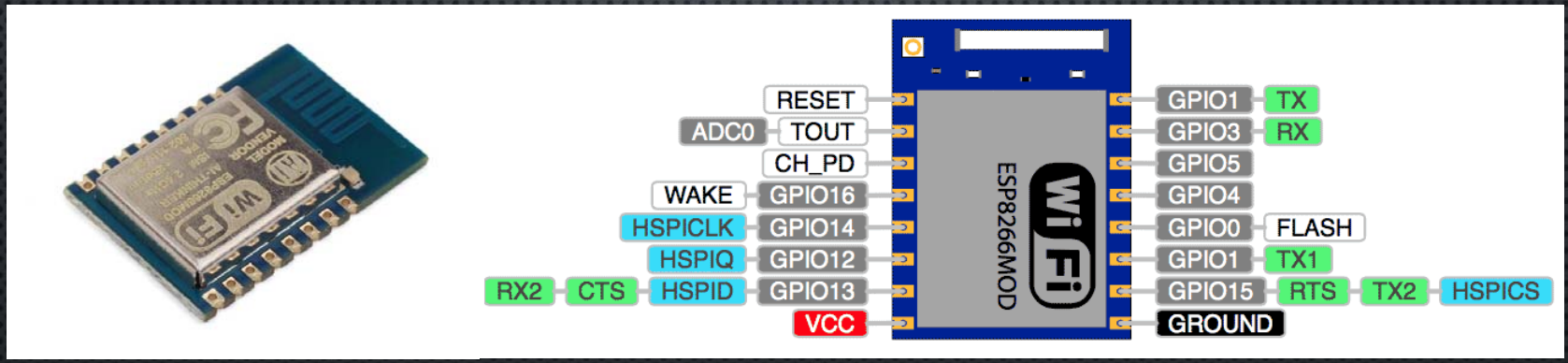
ESP8266 (ESP-01)



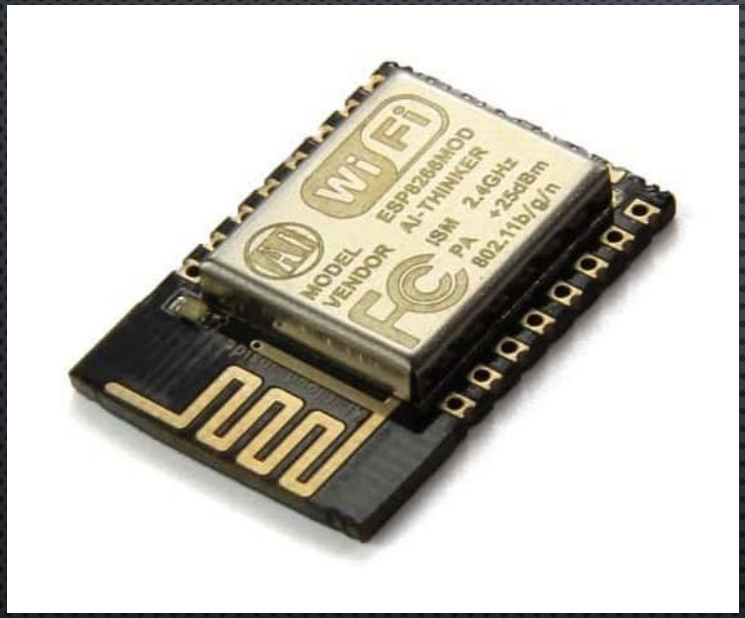
ESP8266 (ESP-03)

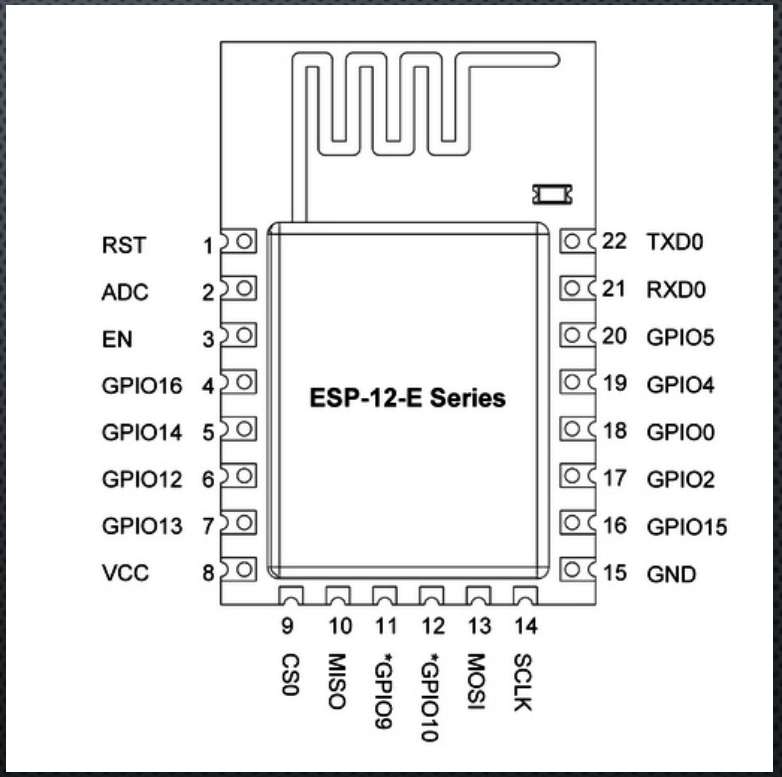


ESP8266 (ESP-07)



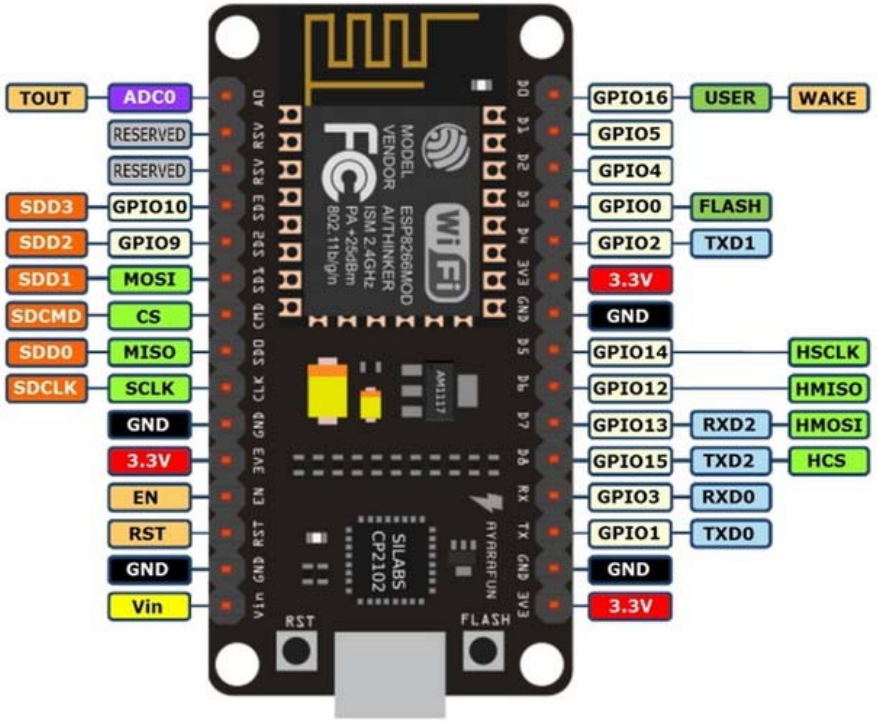
ESP8266 (ESP-12E)



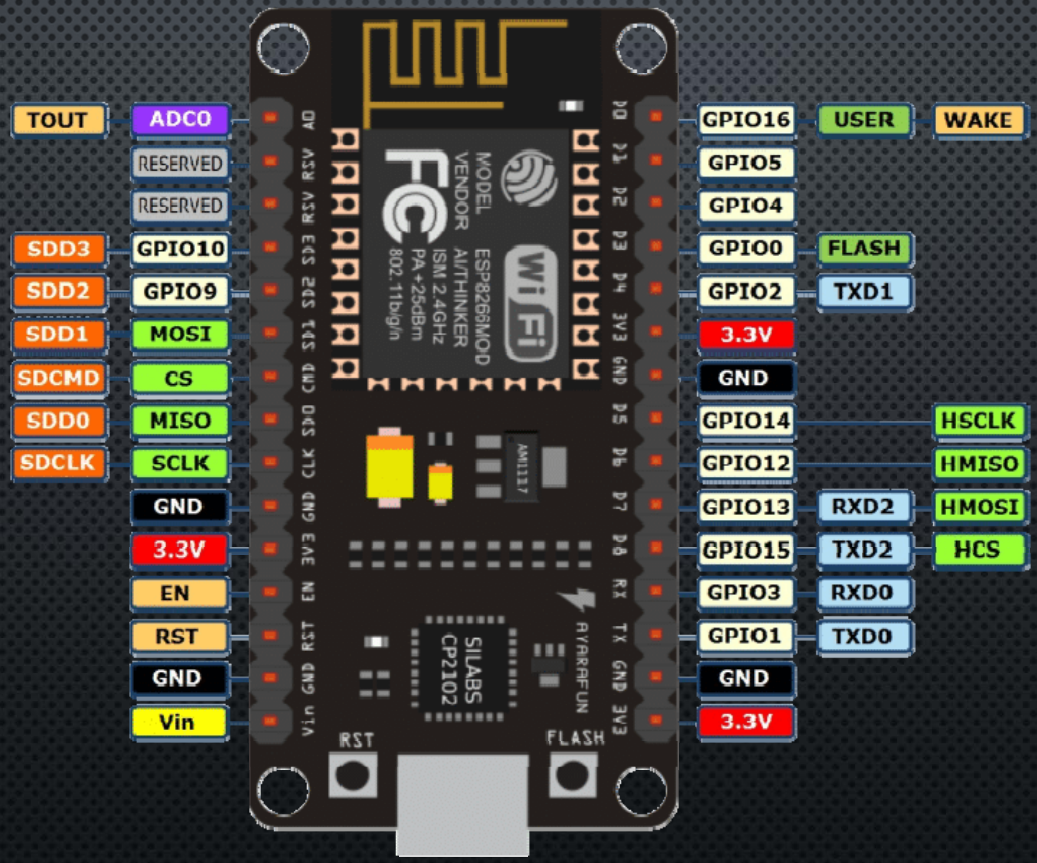


ESP8266 NodeMCU V1 Pin layout

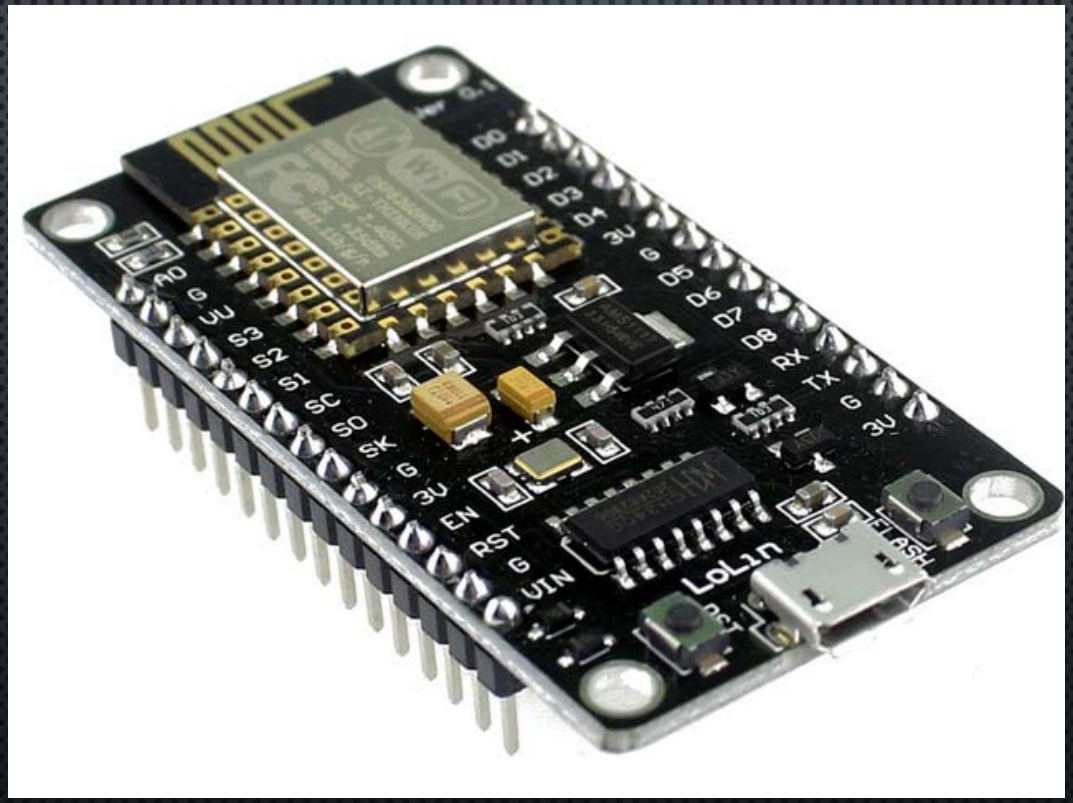
NodeMCU ESP12 Dev Kit V1.0 Pin Definition:



ESP8266 NodeMCU V2 Pin layout



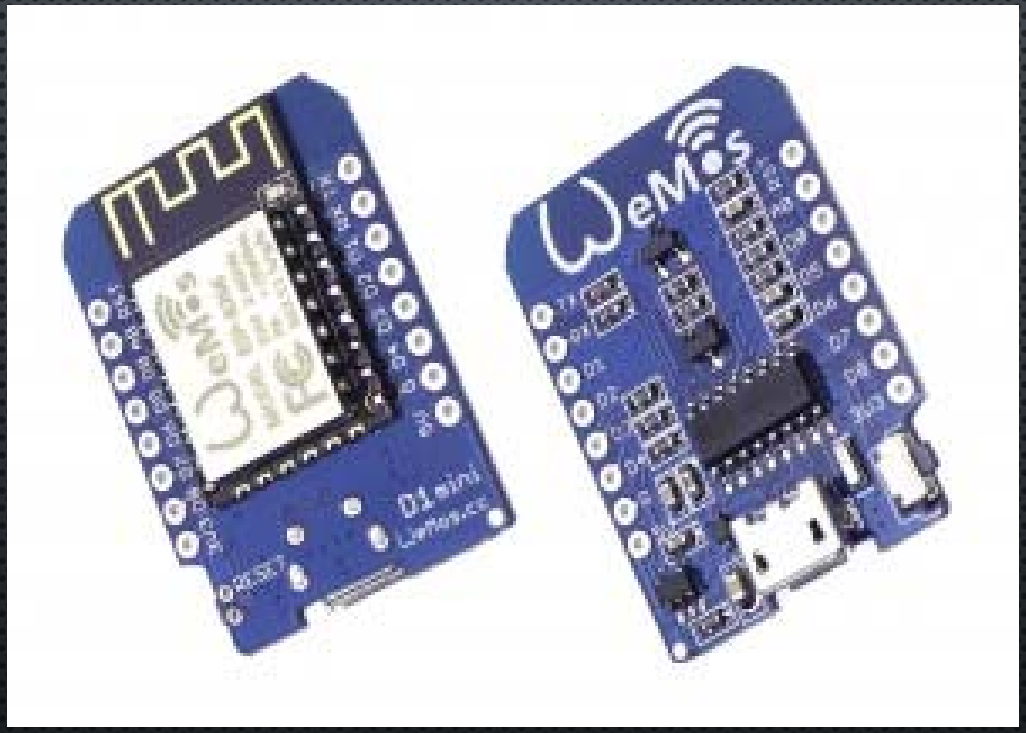
ESP8266 NodeMCU V3 Pin layout



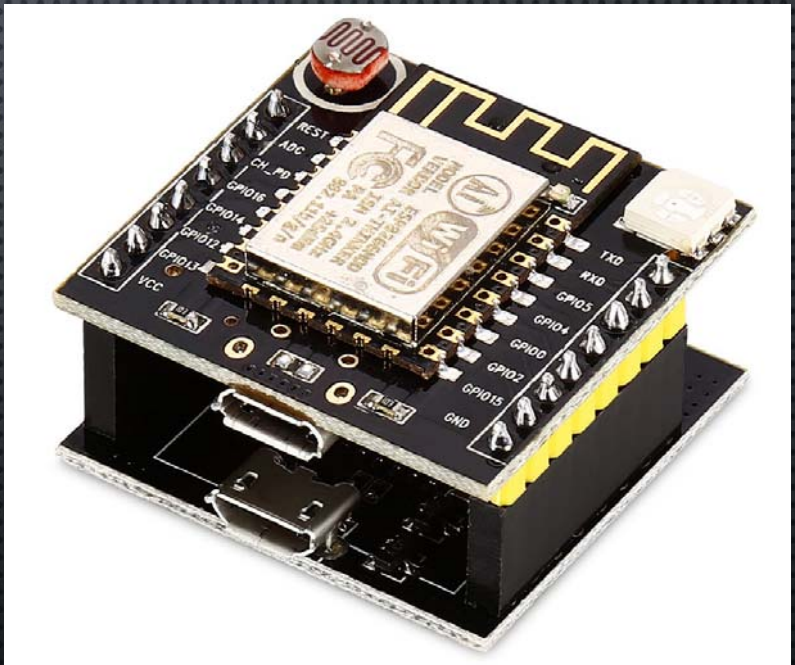
WeMos D1


RESET
SCL/D1
SDA/D2
GND
SCK/D5
MISO/D6
MOSI/D7
SS/D8
D7
D6
D5
D4
D3
SDA/D2
SCL/D1
D0
TX
RX
WeMos D1
ON
SCK
3V3 RX TX GND
3V3 SCL SDA GND
Powered by
WeMos.cc
IOREF RST 3V3 5V GND VIN
A0
SDA
SCL

WeMos D1 Mini

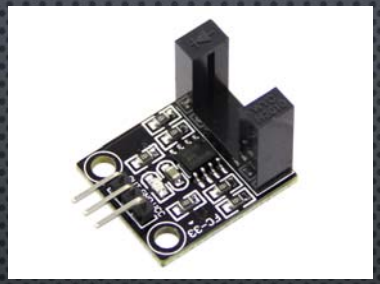
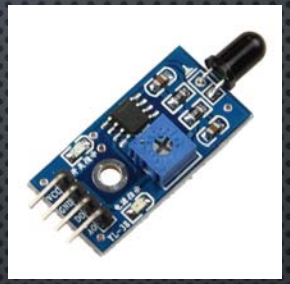
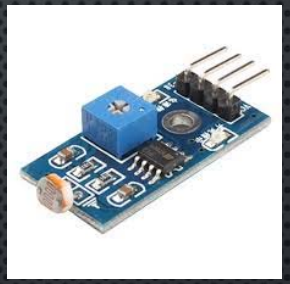


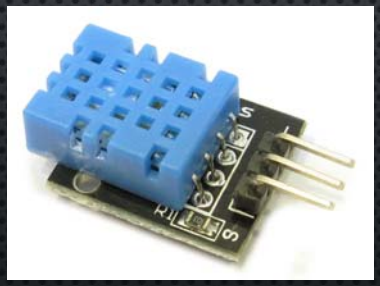
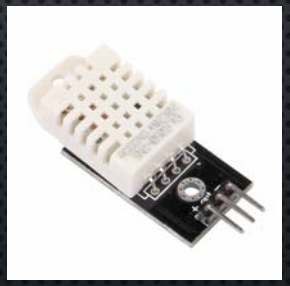
Witty cloud Mini NodeMCU



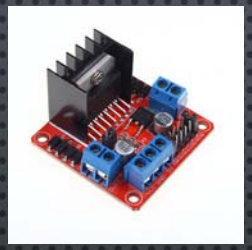
SENSOR & CONTROL MODULES



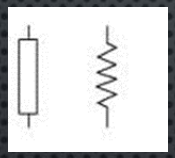




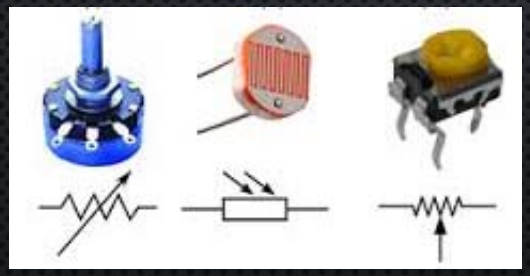

SENSOR & CONTROL MODULES



BASIC ELECTRONIC DEVICES

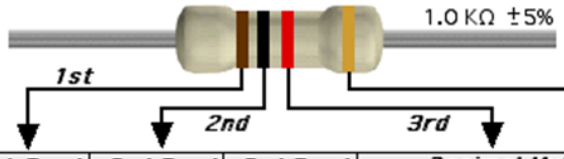
Resistor



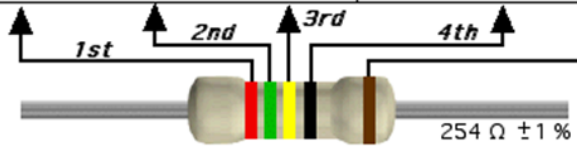


RESISTOR COLOR CODE GUIDE

4- Band Code



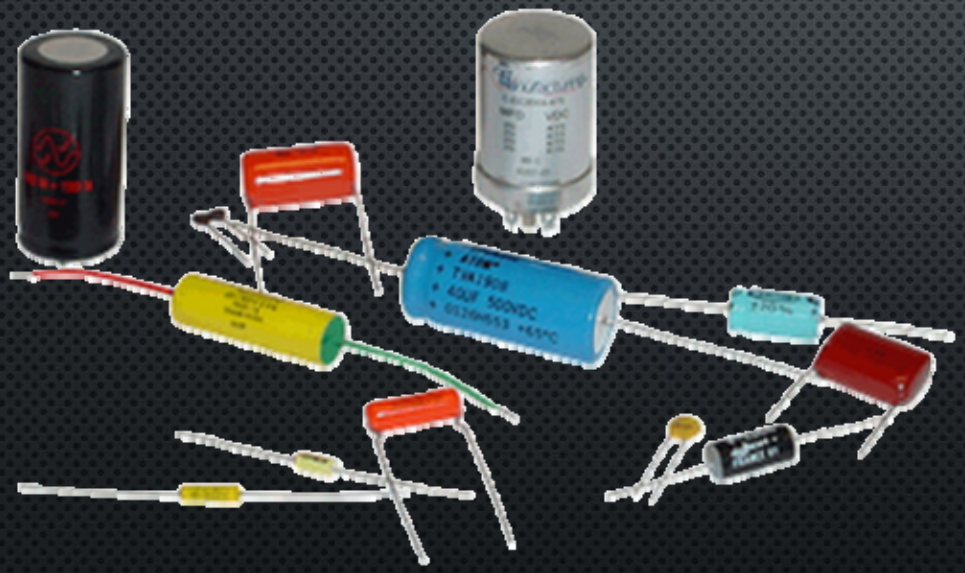
Color	1st Band	2nd Band	3rd Band	Decimal Multiplier	Tolerance
Black	0	0	0	1	
Brown	1	1	1	10	± 1 %
Red	2	2	2	100	± 2 %
Orange	3	3	3	1K	
Yellow	4	4	4	10K	
Green	5	5	5	100K	
Blue	6	6	6	1M	
Violet	7	7	7	10M	
Gray	8	8	8		
White	9	9	9		
Gold				0.1	± 5 %
Silver				0.01	± 10 %
None					± 20 %

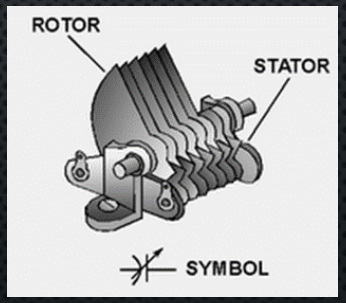


5- Band Code

BASIC ELECTRONIC DEVICES

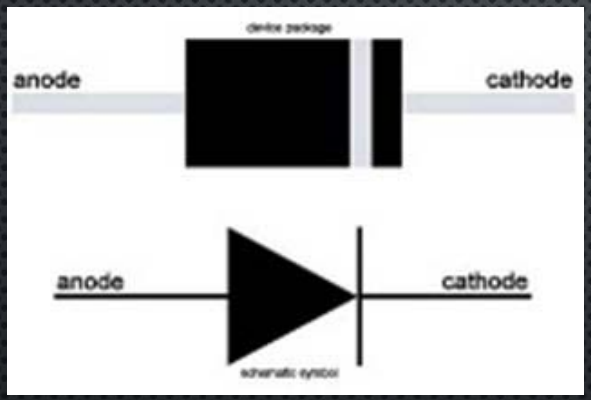
Capacitor



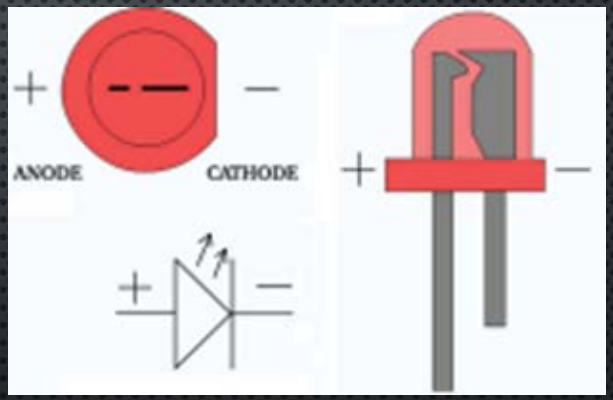


BASIC ELECTRONIC DEVICES

Diode



LED




Workshop

Tools for Develop

1. Arduino IDE and SDK for ESP8266
2. CP2012 USB to UART driver
3. Web Browser: Chrome or Firefox

Arduino IDE and SDK for ESP8266

Download and Install Arduino IDE

- https://www.arduino.cc/en/Main/Software

Additional Board Manager URLs

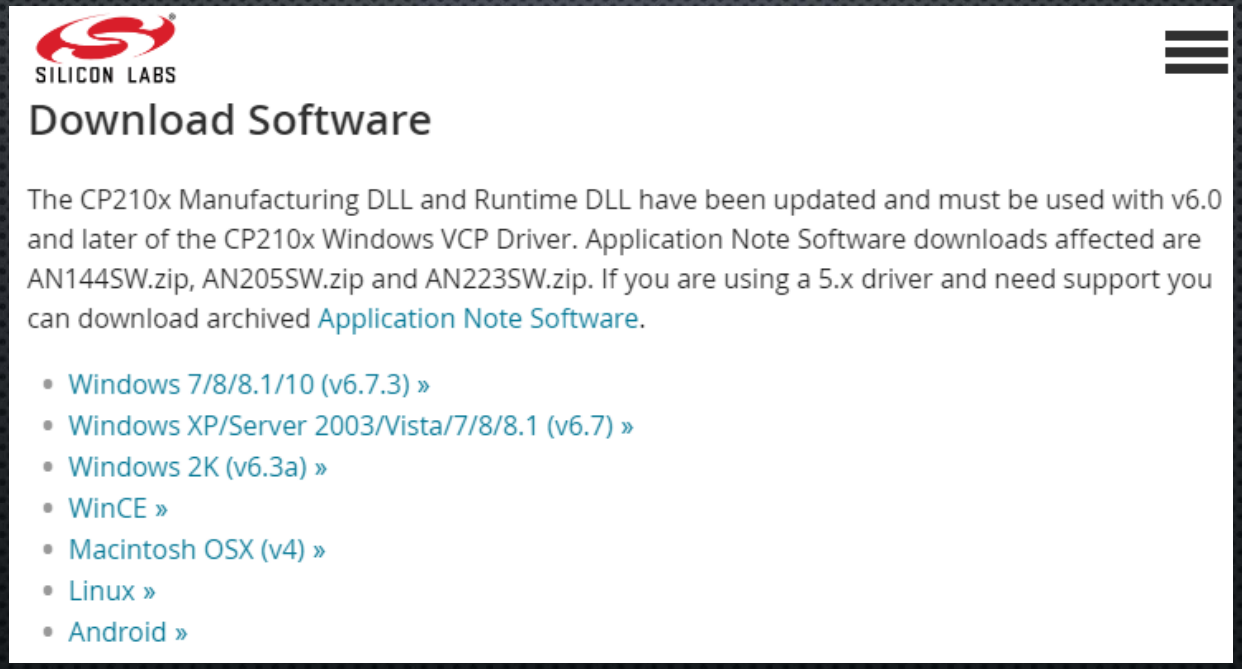
- http://arduino.esp8266.com/stable/package_esp8266com_index.json

Check Install ESP8266 in Boards Manager... Menu

- esp8266 by ESP8266 Community version 2.3.0 INSTALLED

CP2012 USB to UART driver

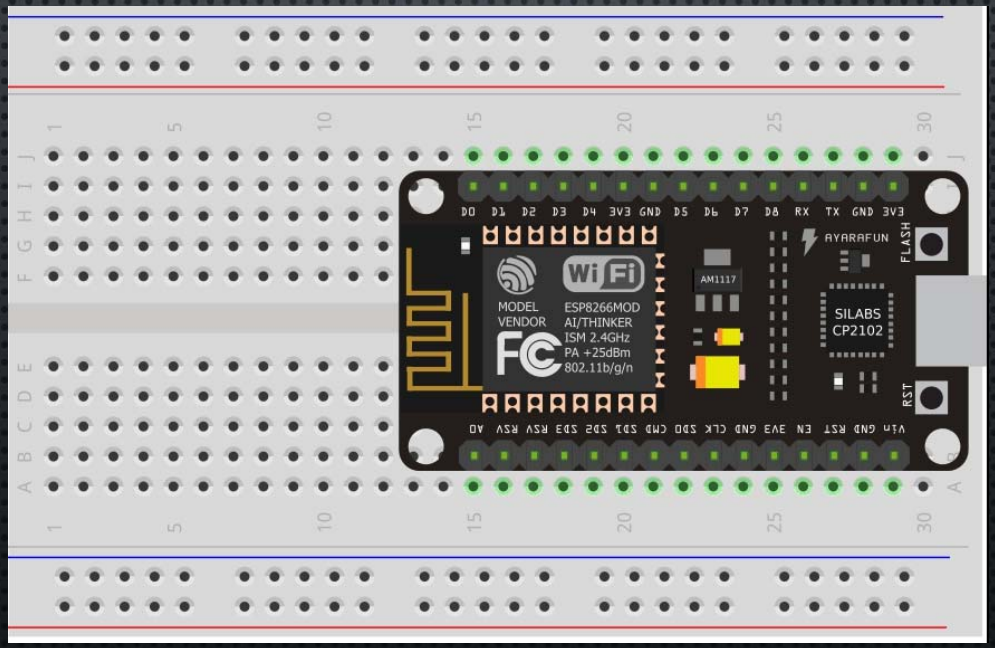
https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx



Workshop#1
Start learning ESP8266 with Arduino IDE

- Blink
- Toggle Switch
- WiFi Scan
- Web server

Workshop#1
Prepare hardware snap in breadboard



Workshop#1 [test board with blink]

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on (Note that LOW is the voltage
level
                                    // but actually the LED is on; this is because
                                    // it is acive low on the ESP-01)
  delay(1000);                      // Wait for a second
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);                      // Wait for two seconds (to demonstrate the active low LED)
}
```

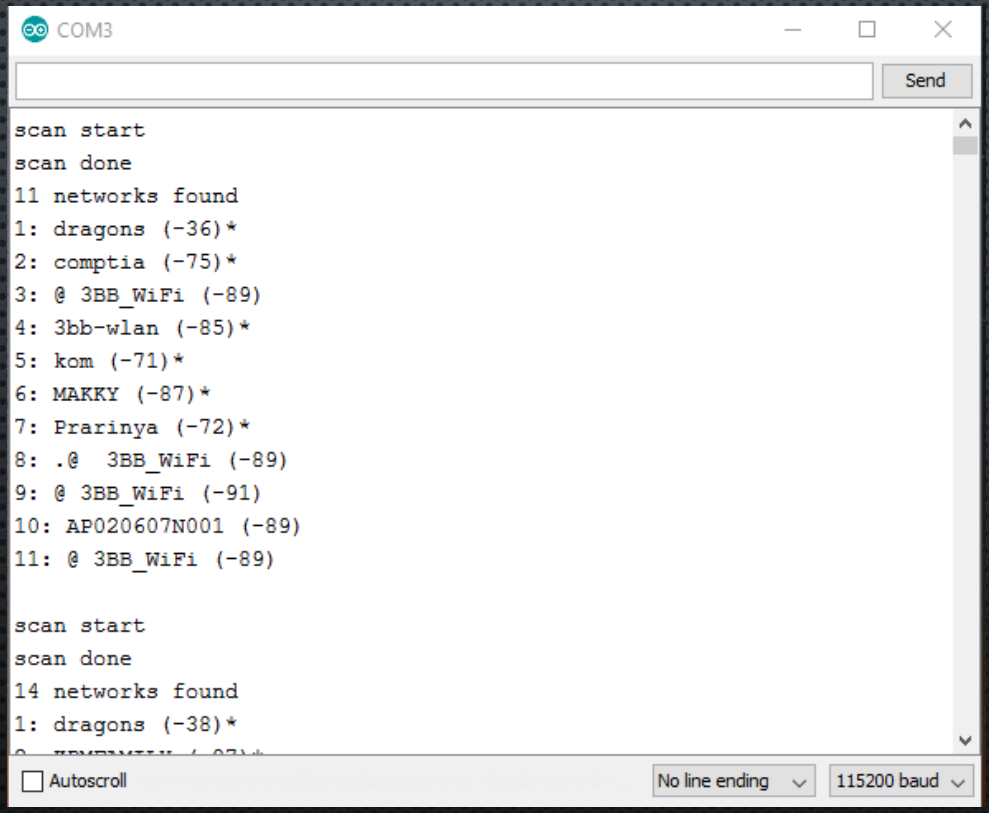
Workshop#1 [test Toggle Switch]

```
#define sw1 D4 // Declare all variables
int st_sw1 = 0;
int st_1 = 0;
int last_st_sw1 = 1;
void setup() {
  pinMode(LED_BUILTIN, OUTPUT); // Set pin mode
  pinMode(sw1, INPUT_PULLUP);
}

void loop() {
  st_sw1 = digitalRead(sw1); // Read input port1
  if ((st_sw1 == 0) && (last_st_sw1 == 1)) // Check current status
  {
    st_1 = ~st_1; // Toggle
    digitalWrite(LED_BUILTIN, st_1); // Drive LED1
    delay(250);
  }
  last_st_sw1 = st_sw1; // Update current status
}
```

Workshop#1 [Wifi scan] - sketch

```
#include "ESP8266WiFi.h"

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

  // Set WiFi to station mode and disconnect from an AP if it
was previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

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

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

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

Workshop#1 [Wifi scan] – serial monitor



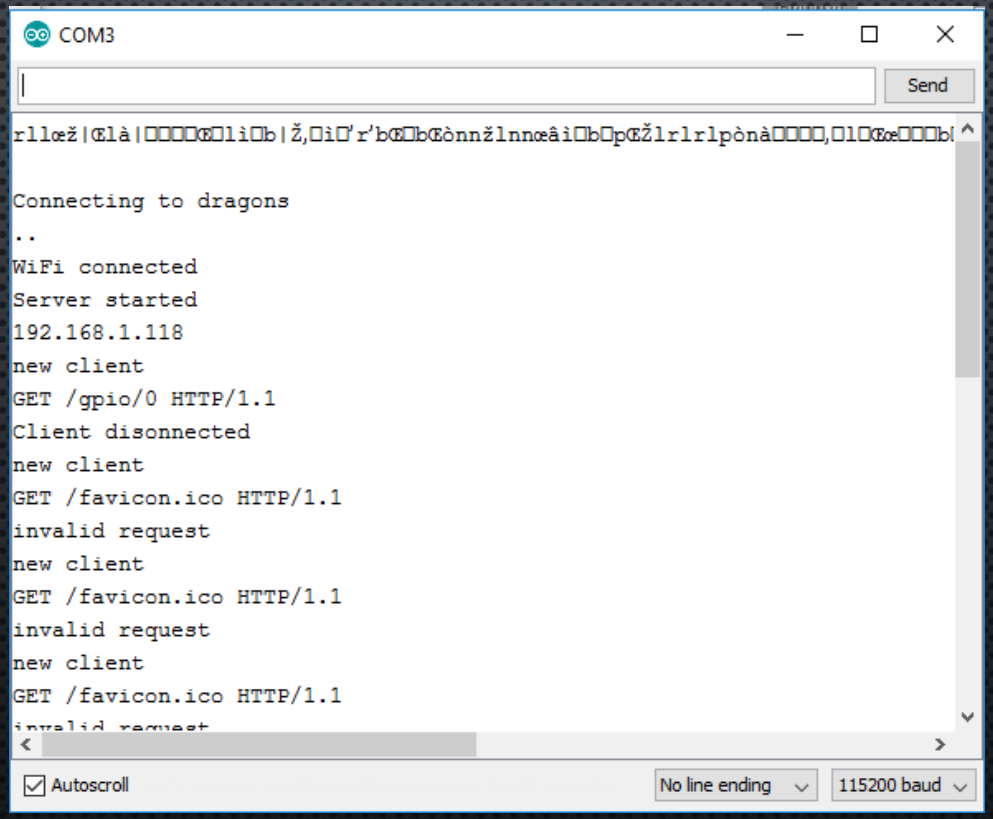
Workshop#1 [wifiWeb server] – sketch

```
#include <ESP8266WiFi.h>

const char* ssid = “your-ssid";
const char* password = “your-password";

// Create an instance of the server
// specify the port to listen on as an argument
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);

  // prepare GPIO2
  pinMode(2, OUTPUT);
  digitalWrite(2, 0);

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

  WiFi.begin(ssid, password);

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

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.println(WiFi.localIP());
}
```

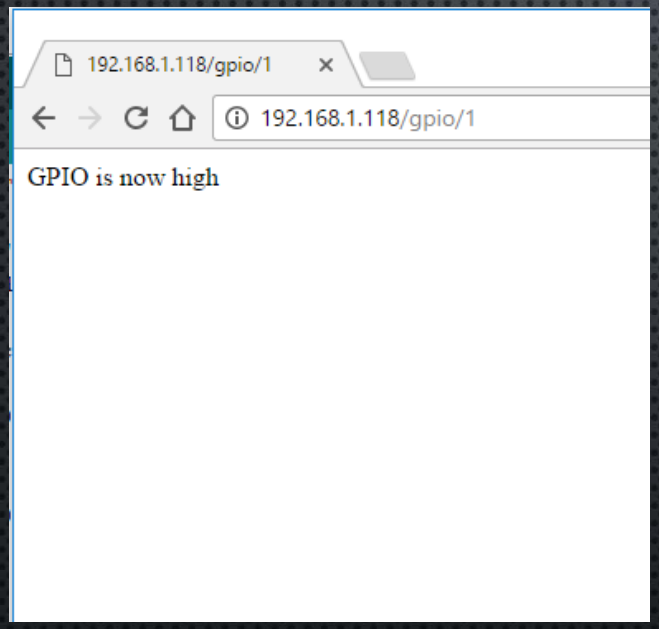
Workshop#1 [wifiWeb server] – sketch

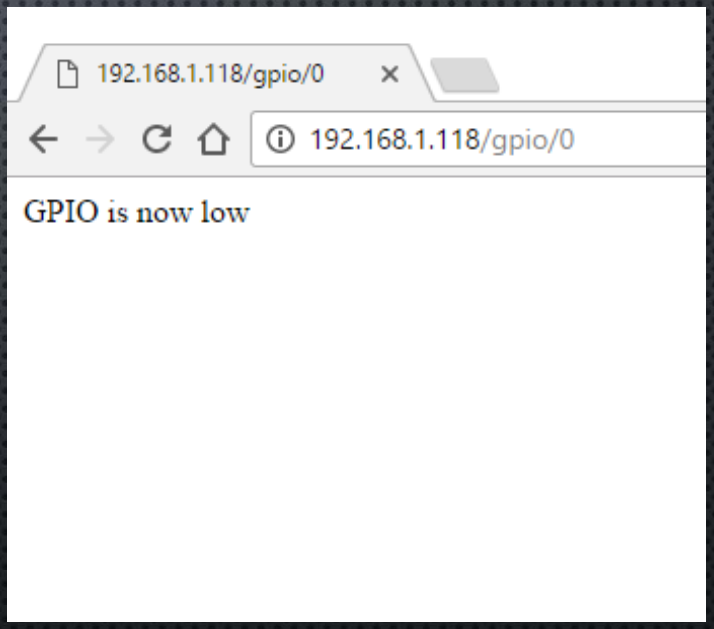
```
void loop() {
 // Check if a client has connected
 WiFiClient client = server.available();
 if (!client) {
  return;
 }

 // Wait until the client sends some data
 Serial.println("new client");
 while(!client.available()){
  delay(1);
 }

 // Read the first line of the request
 String req = client.readStringUntil('\r');
 Serial.println(req);
 client.flush();

 // Match the request
 int val;
 if (req.indexOf("/gpio/0") != -1)
  val = 0;
 else if (req.indexOf("/gpio/1") != -1)
  val = 1;
 else {
  Serial.println("invalid request");
  client.stop();
  return;
 }

  // Set GPIO2 according to the request
  digitalWrite(2, val);

  client.flush();

  // Prepare the response
  String s = "HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\nGPIO is now ";
  s += (val)?"high":"low";
  s += "</html>\n";

  // Send the response to the client
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");

  // The client will actually be disconnected
  // when the function returns and 'client' object is detroyed
}
```

Workshop#1 [wifiWeb server] – serial monitor

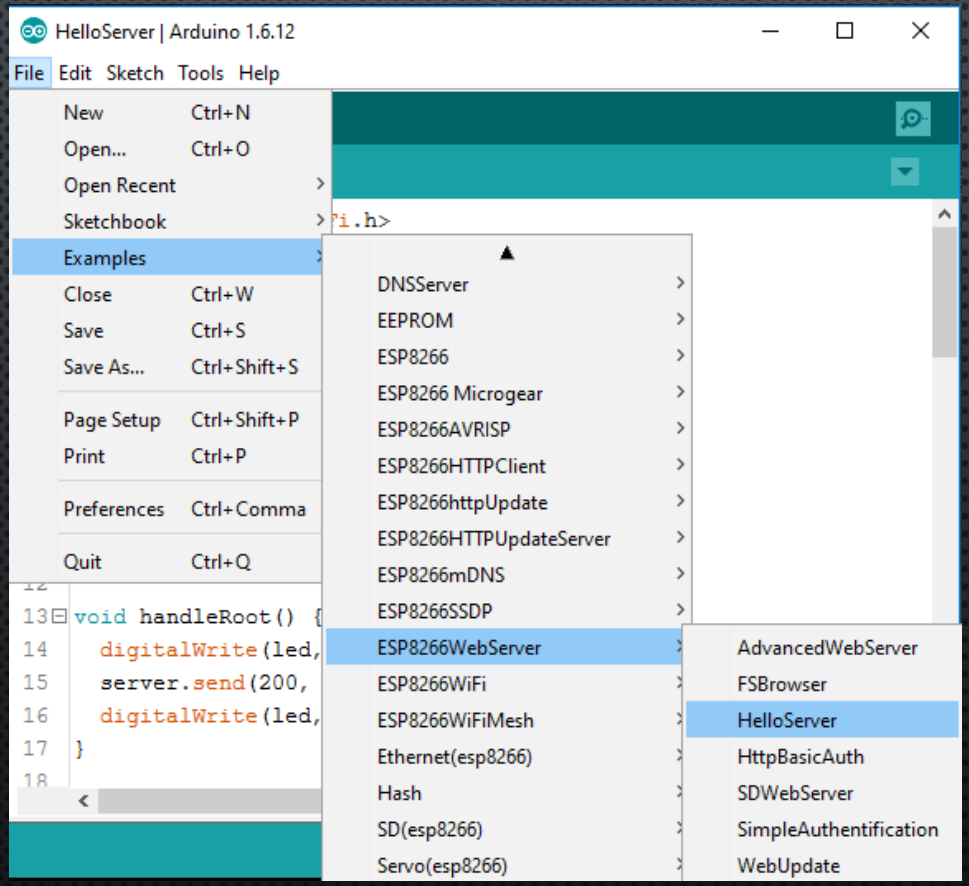


Workshop#1 [wifiWeb server] – web browser

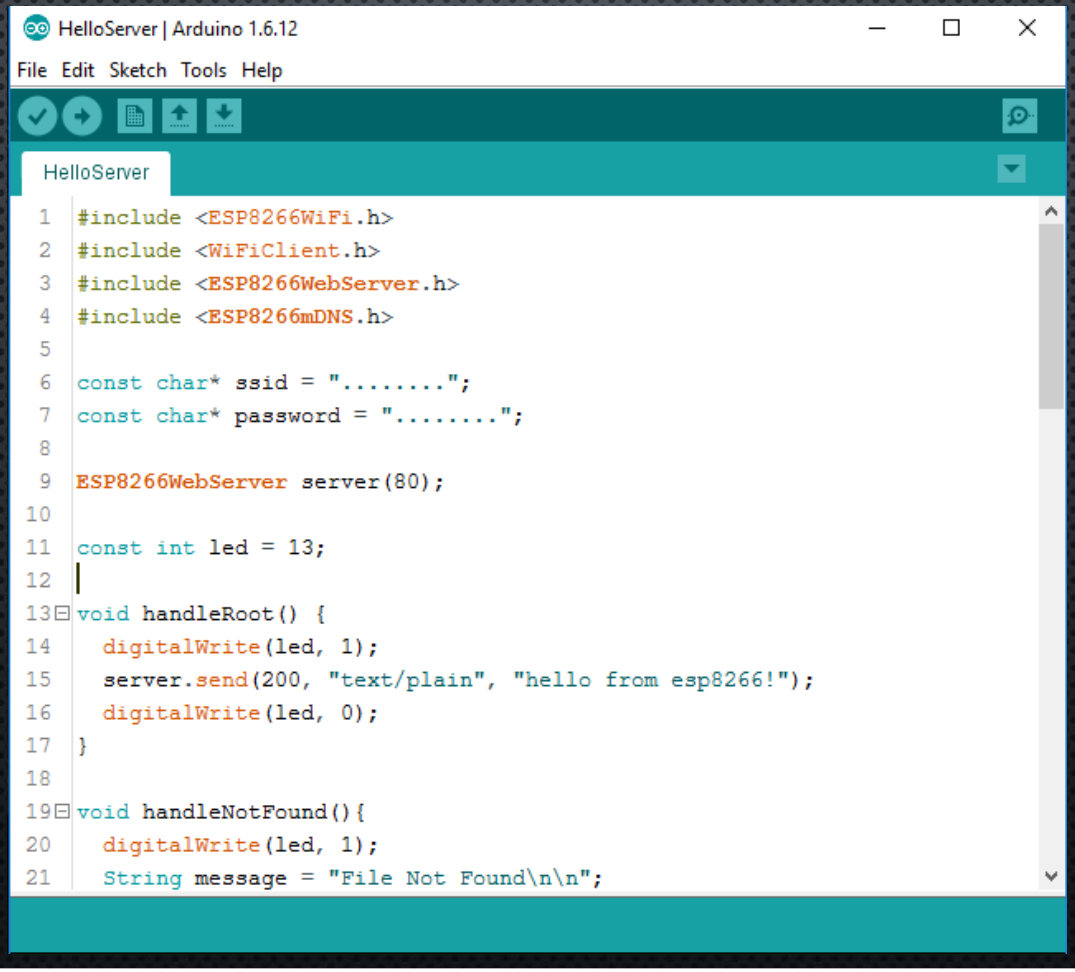




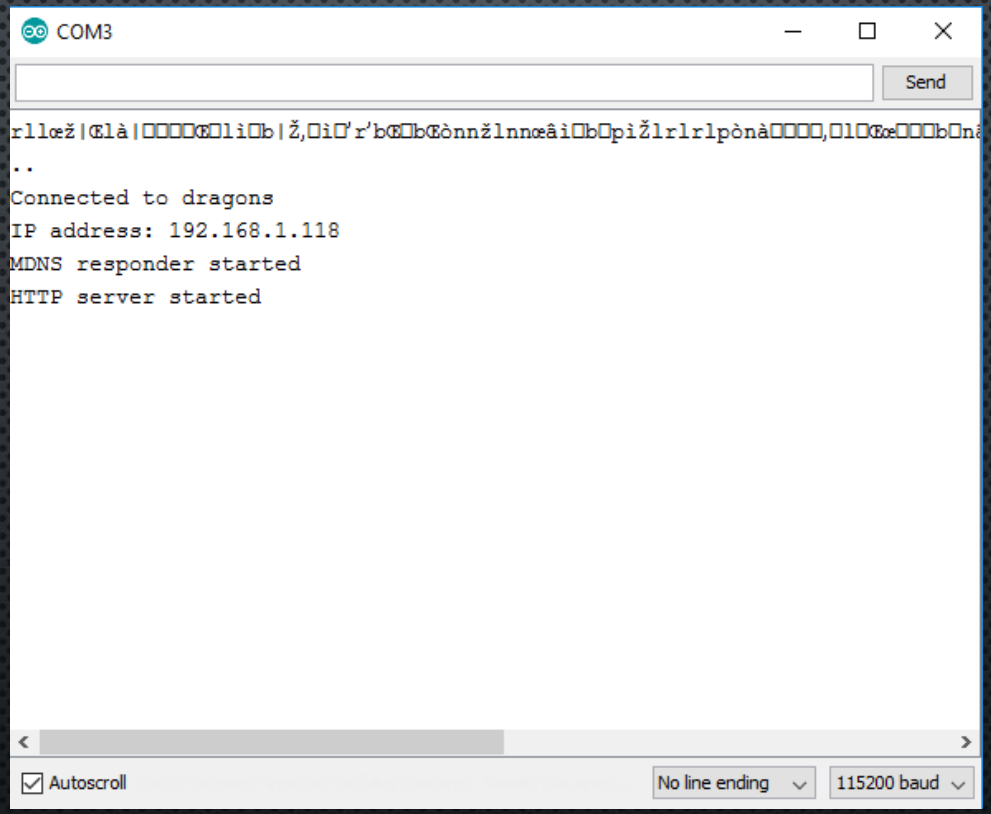
Workshop#1 [helloserver]



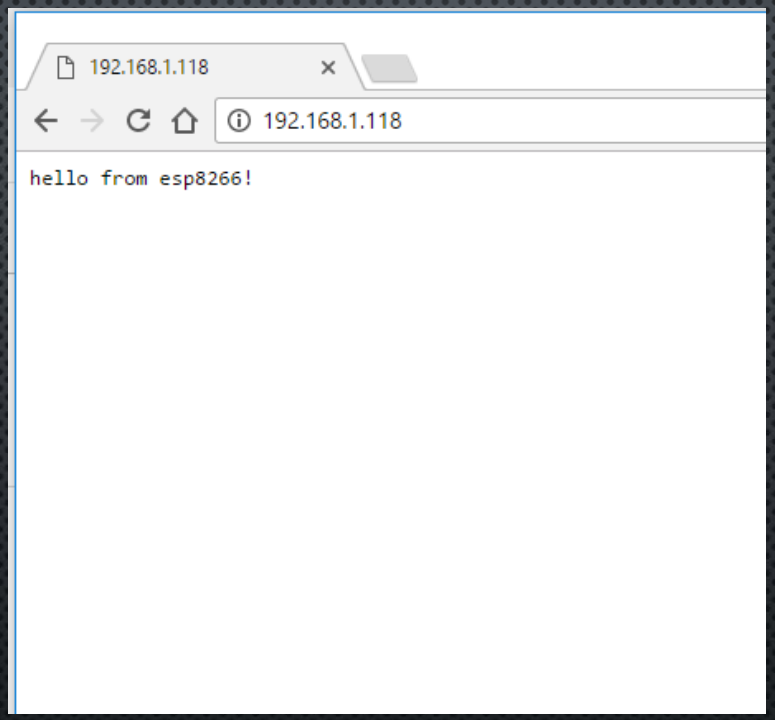
Workshop#1 [helloserver]



Workshop#1 [helloserver]-serial monitor



Workshop#1 [helloserver]-web browser



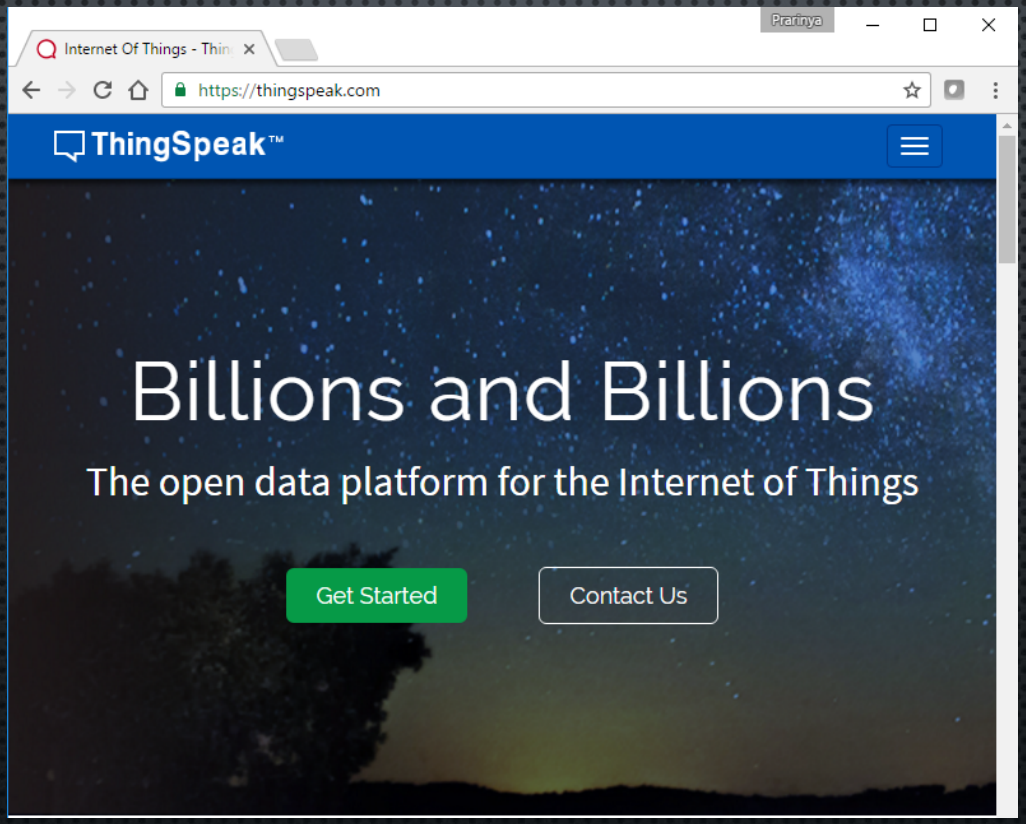
Workshop#2
upload data to cloud (thinkspeak)

- Apply Thinkspeak
- Test send data
- Customize visual board

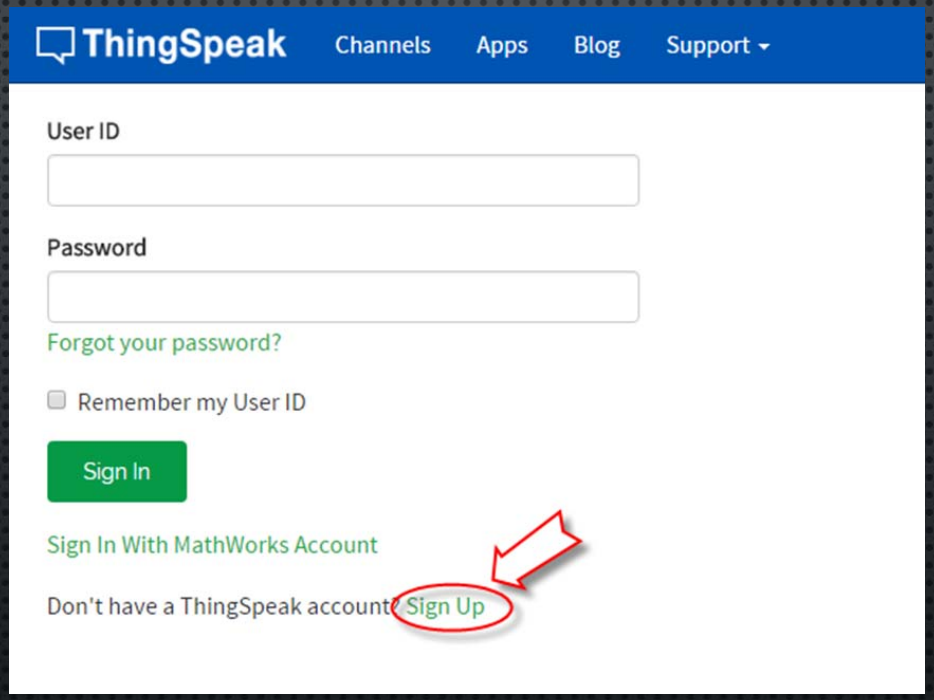
Workshop#2 Apply Thinkspeak

1. Register Thingspeak
2. Create New Channel
3. Receive API Key
4. Use API Key send data to server

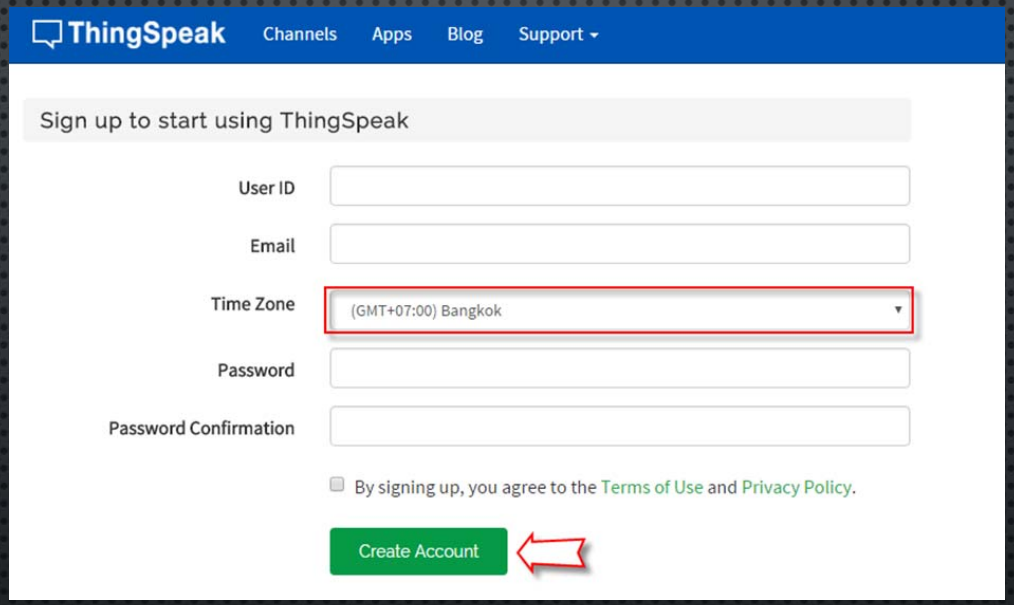
Workshop#2 Apply Thinkspeak



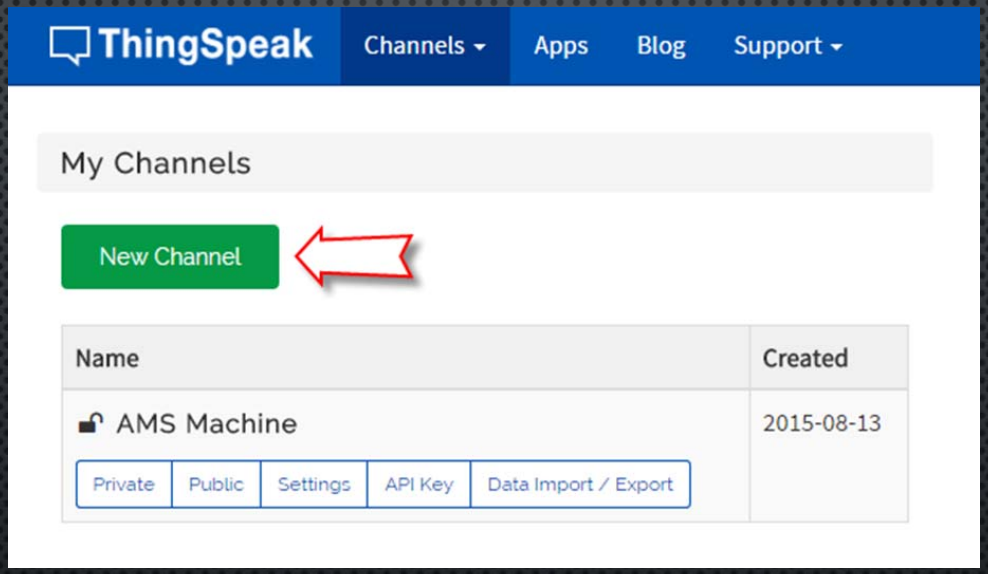
Workshop#2 Apply Thinkspeak



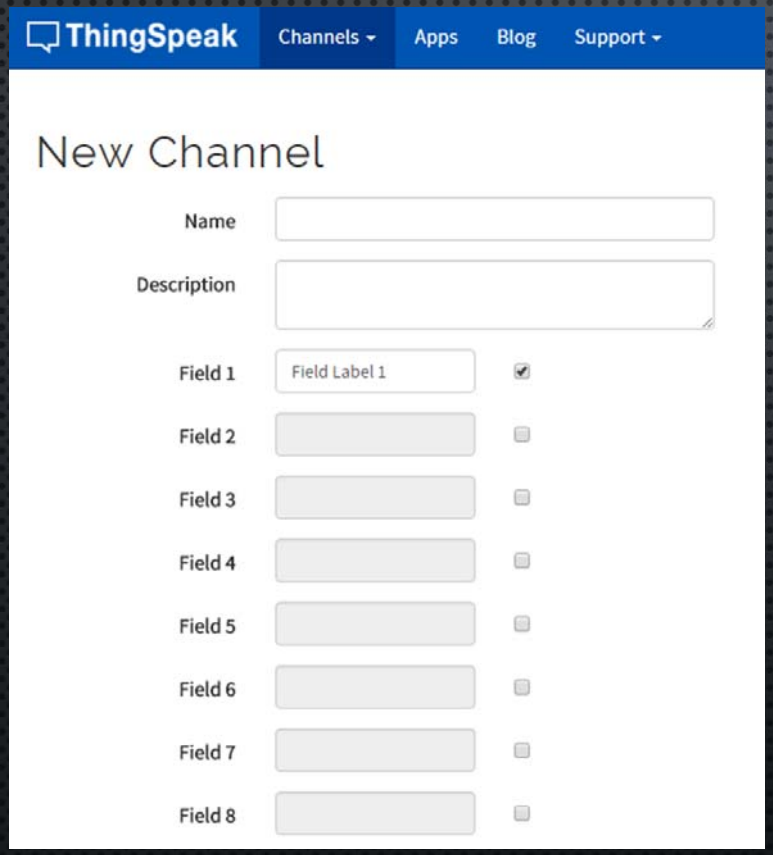
Workshop#2 Apply Thinkspeak

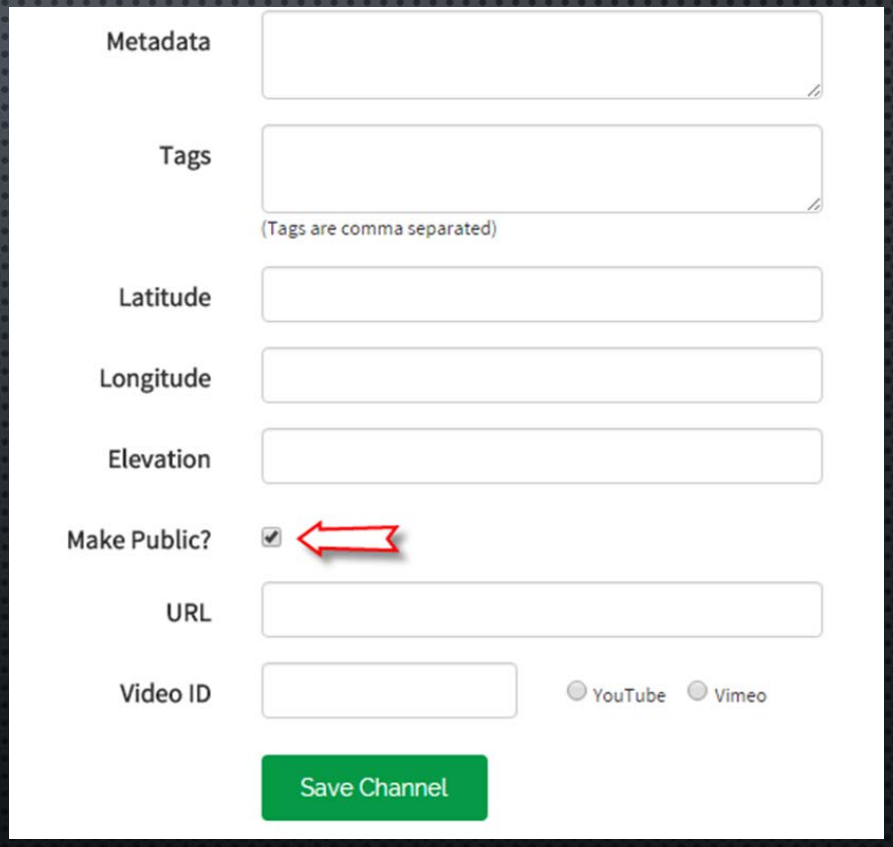


Workshop#2 Create New Channel

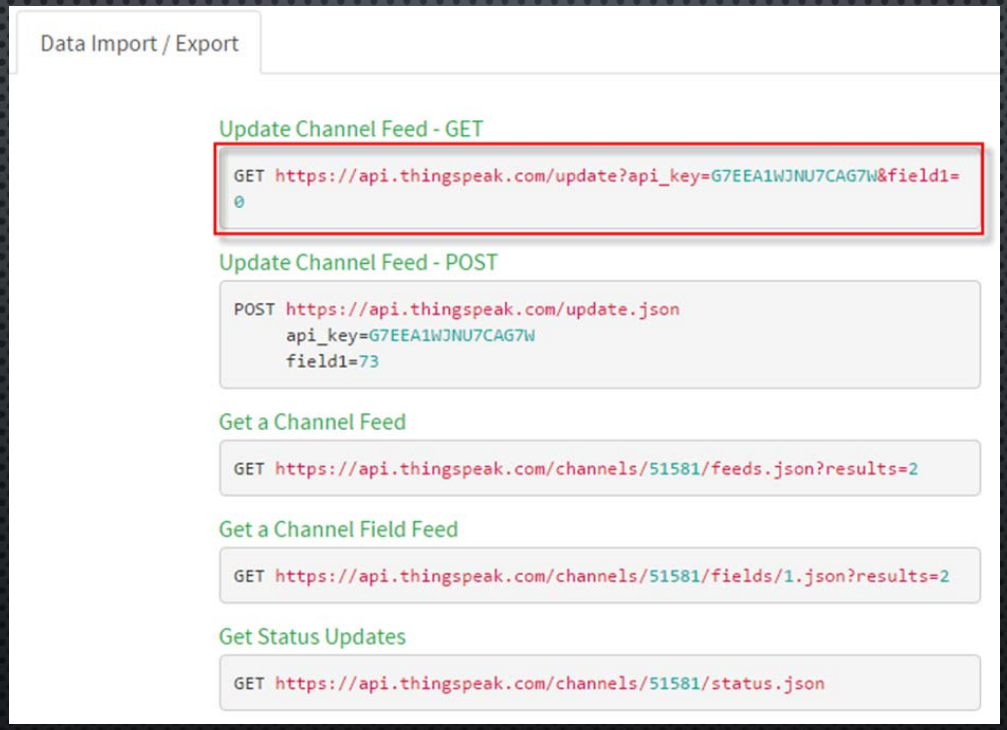


Workshop#2 Create New Channel

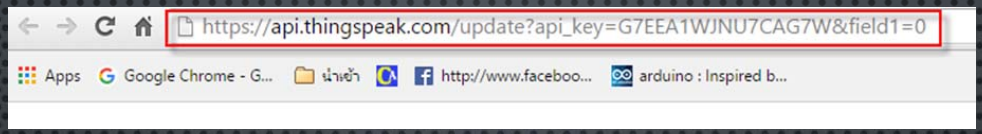


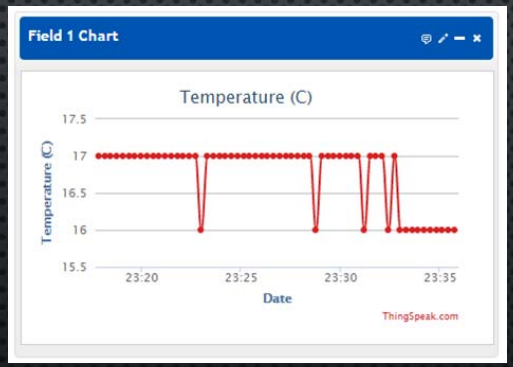


Workshop#2 Create New Channel

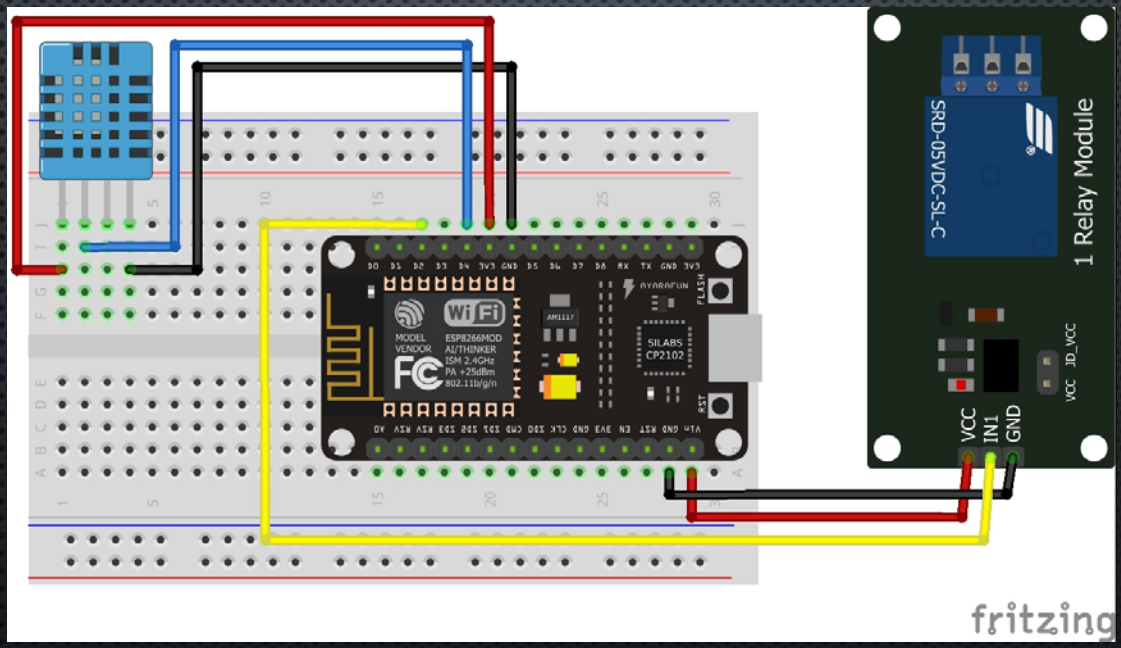


Workshop#2 Create New Channel





Workshop#2 connect hardware



Workshop#2 sketch for test

```
#include "DHT.h"
#include <ESP8266WiFi.h>

#define PUMP_RLY 4    // output drive relay for pump GPIO4 (D2)
#define DHTPIN 2     // what pin we're connected to GPIO2 (D4)
#define DHTTYPE DHT11   // DHT 11

#define DEBUG

#define DEBUG_PRINTER Serial

#ifdef DEBUG
#define DEBUG_PRINT(...) { DEBUG_PRINTER.print(__VA_ARGS__); }
#define DEBUG_PRINTLN(...) { DEBUG_PRINTER.println(__VA_ARGS__); }
#else
#define DEBUG_PRINT(...) {}
#define DEBUG_PRINTLN(...) {}
#endif

const char* ssid     = “SSID"; //Use own ssid
const char* password = “PASSWORD"; //use password of AP

DHT *dht;

void connectWifi();
void reconnectWifiIfLinkDown();
void initDht(DHT **dht, uint8_t pin, uint8_t dht_type);
void readDht(DHT *dht, float *temp, float *humid);
void uploadThingsSpeak(float t, float h);

void setup() {
  Serial.begin(115200);
  delay(10);
  pinMode(PUMP_RLY, OUTPUT); // Initialize the PUMP_RLY(4) pin as
an output
  digitalWrite(PUMP_RLY, HIGH);// Make sure relay is normal off

  connectWifi();

  initDht(&dht, DHTPIN, DHTTYPE);
}

void loop() {
  static float t_dht;
  static float h_dht;

  readDht(dht, &t_dht, &h_dht);
  if(h_dht < 30 || t_dht > 26) // condition for make relay on
  {
   digitalWrite(PUMP_RLY, LOW); //If condition true do this!
  } else
  {
   digitalWrite(PUMP_RLY, HIGH);
  }
  uploadThingsSpeak(t_dht, h_dht);

  // Wait a few seconds between measurements.
  delay(10 * 1000);
  reconnectWifiIfLinkDown();
}
```

Workshop#2 sketch for test

```
void reconnectWifiIfLinkDown() {
  if (WiFi.status() != WL_CONNECTED) {
    DEBUG_PRINTLN("WIFI DISCONNECTED");
    connectWifi();
  }
}

void connectWifi() {
  DEBUG_PRINTLN();
  DEBUG_PRINTLN();
  DEBUG_PRINT("Connecting to ");
  DEBUG_PRINTLN(ssid);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    DEBUG_PRINT(".");
  }

  DEBUG_PRINTLN("");
  DEBUG_PRINTLN("WiFi connected");
  DEBUG_PRINTLN("IP address: ");
  DEBUG_PRINTLN(WiFi.localIP());
}

void initDht(DHT **dht, uint8_t pin, uint8_t dht_type) {
  // Connect pin 1 (on the left) of the sensor to +5V
  // NOTE: If using a board with 3.3V logic like an Arduino Due
connect pin 1
    // to 3.3V instead of 5V!
    // Connect pin 2 of the sensor to whatever your DHTPIN is
    // Connect pin 4 (on the right) of the sensor to GROUND
    // Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the
sensor

    // Initialize DHT sensor for normal 16mhz Arduino
    // NOTE: For working with a faster chip, like an Arduino Due or
Teensy, you
    // might need to increase the threshold for cycle counts
considered a 1 or 0.
    // You can do this by passing a 3rd parameter for this threshold.
It's a bit
    // of fiddling to find the right value, but in general the faster the
CPU the
    // higher the value.  The default for a 16mhz AVR is a value of 6.
For an
    // Arduino Due that runs at 84mhz a value of 30 works.
    // Example to initialize DHT sensor for Arduino Due:
    //DHT dht(DHTPIN, DHTTYPE, 30);

    *dht = new DHT(pin, dht_type, 30);
    (*dht)->begin();
    DEBUG_PRINTLN(F("DHTxx test!"))  ;
}
```

Workshop#2 sketch for test

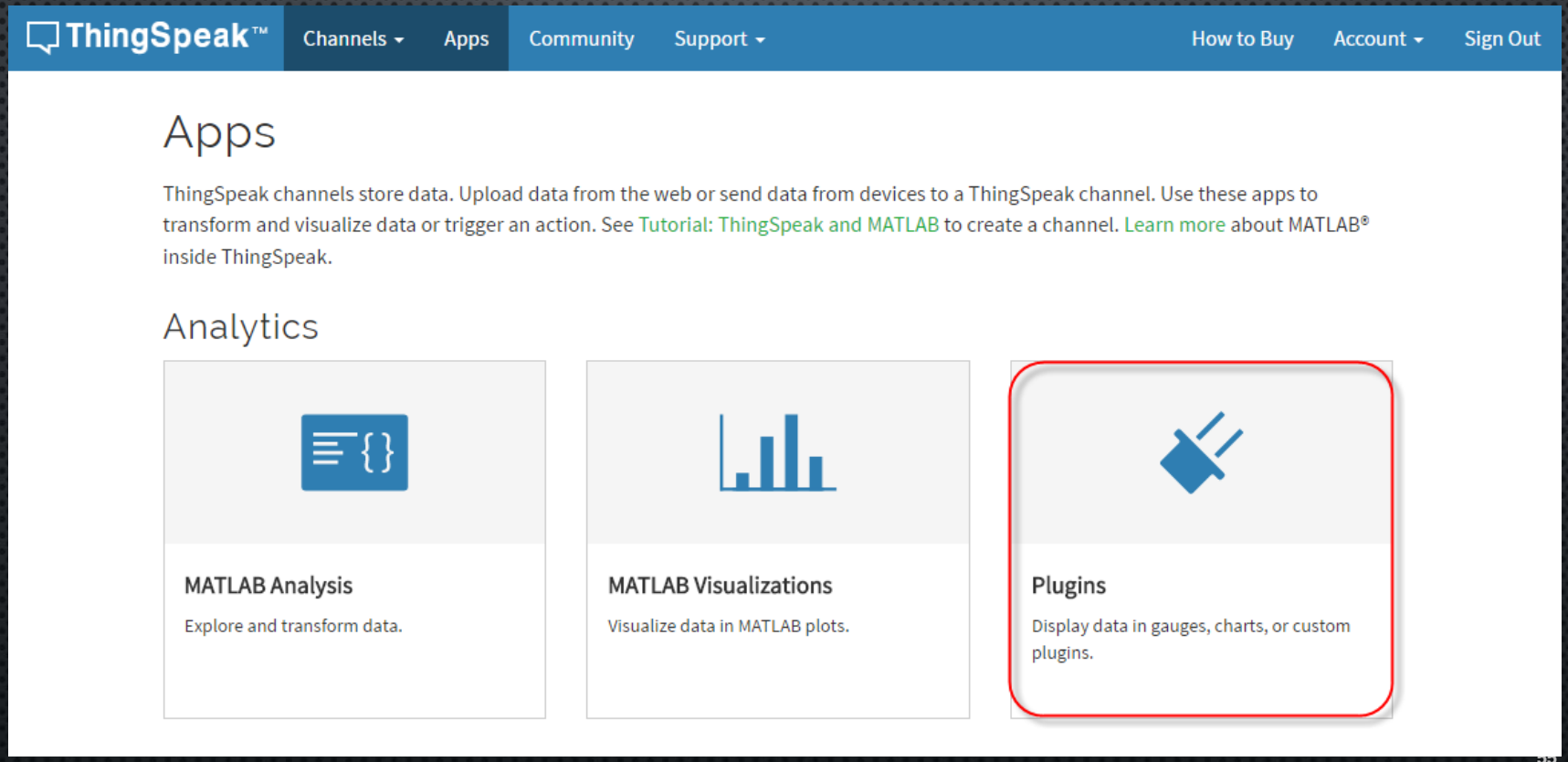
```
void uploadThingsSpeak(float t, float h) {
  static const char* host = "api.thingspeak.com";
  static const char* apiKey = "YOUR_APIKEY";

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    DEBUG_PRINTLN("connection failed");
    return;
  }
  // We now create a URI for the request
  String url = "/update/";
  //  url += streamId;
  url += "?key=";
  url += apiKey;
  url += "&field1=";
  url += t;
  url += "&field2=";
  url += h;

  DEBUG_PRINT("Requesting URL: ");
  DEBUG_PRINTLN(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
}
```

```
void readDht(DHT *dht, float *temp, float *humid) {

  if (dht == NULL) {
    DEBUG_PRINTLN(F("[dht11] is not initialised. please call initDht()
first."));
    return;
  }

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very
slow sensor)
  float h = dht->readHumidity();

  // Read temperature as Celsius
  float t = dht->readTemperature();
  // Read temperature as Fahrenheit
  float f = dht->readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    DEBUG_PRINTLN("Failed to read from DHT sensor!");
    return;
  }
```

Workshop#2 sketch for test

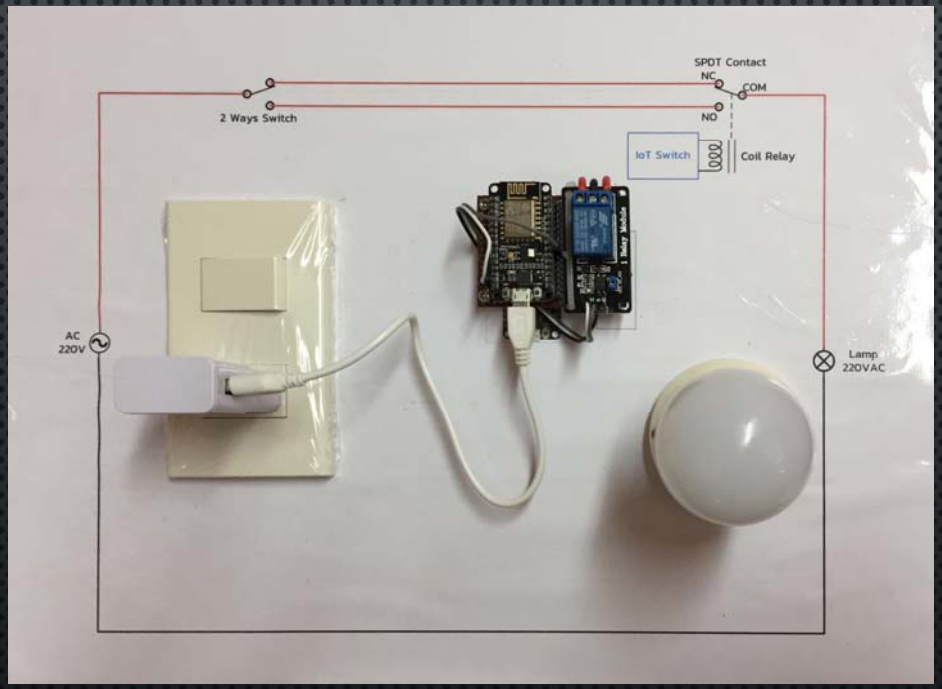
```
  // Compute heat index
    // Must send in temp in Fahrenheit!
    float hi = dht->computeHeatIndex(f, h);

    DEBUG_PRINT("Humidity: ");
    DEBUG_PRINT(h);
    DEBUG_PRINT(" %\t");
    DEBUG_PRINT("Temperature: ");
    DEBUG_PRINT(t);
    DEBUG_PRINT(" *C ");
    DEBUG_PRINT(f);
    DEBUG_PRINT(" *F\t");
    DEBUG_PRINT("Heat index: ");
    DEBUG_PRINT(hi);
    DEBUG_PRINTLN(" *F");

    *temp = t;
    *humid = h;
}
```

Customize visual board

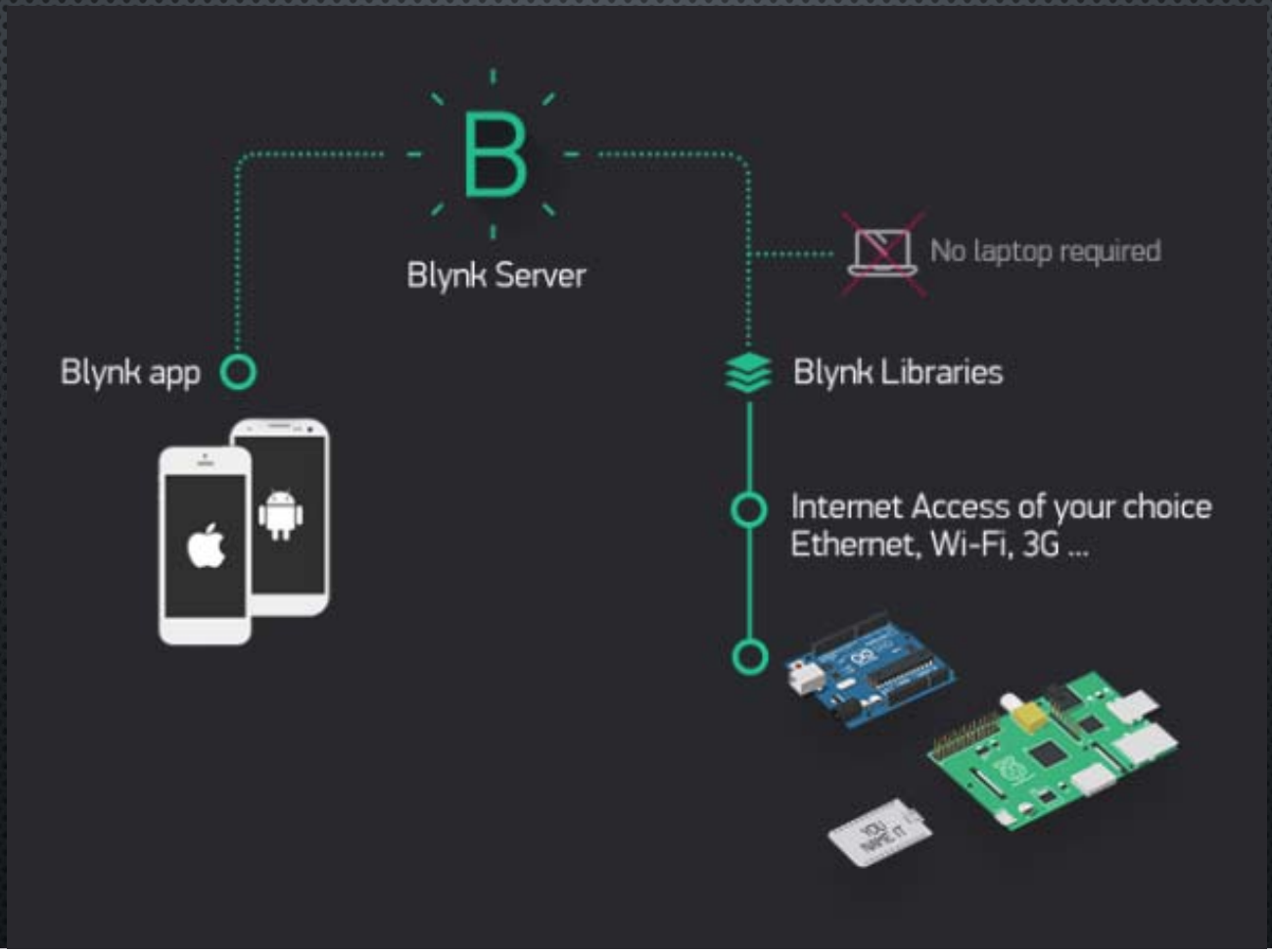


Customize visual board

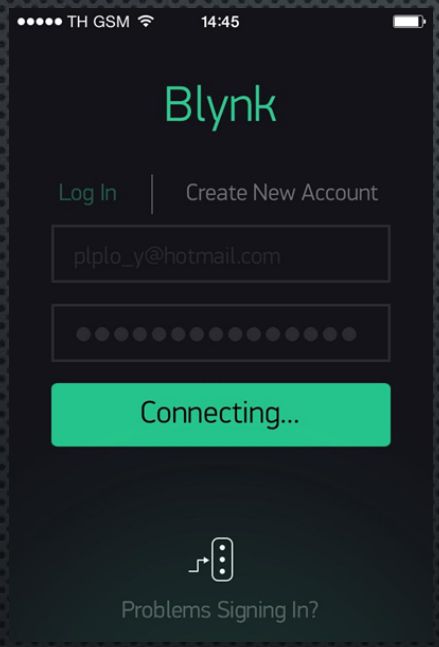
Workshop#3 Mini Project 2Way IoT Switch

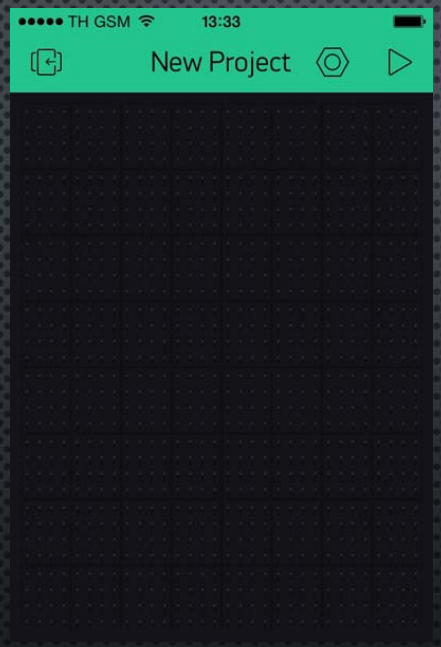


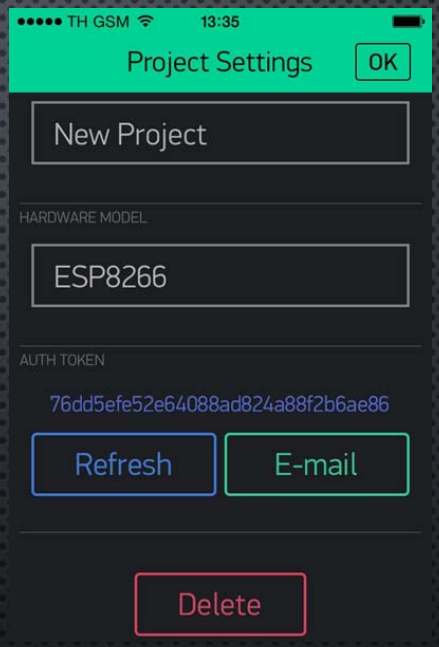
Workshop#3 Control IoT via Blynk

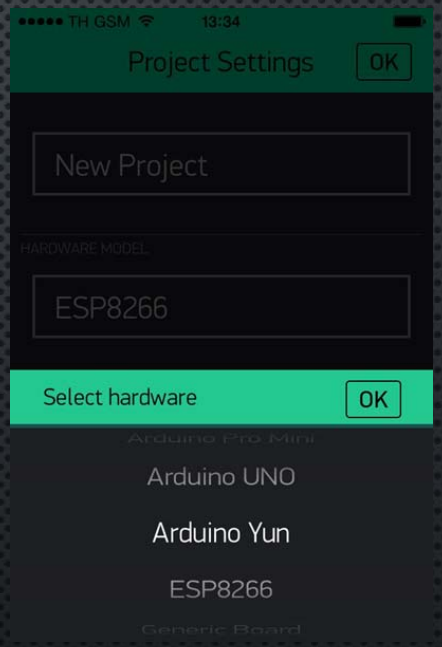


Workshop#3 Control IoT via Blynk









Workshop#3 Control IoT via Blynk

Auth Token for IoT Switch project and device New Device

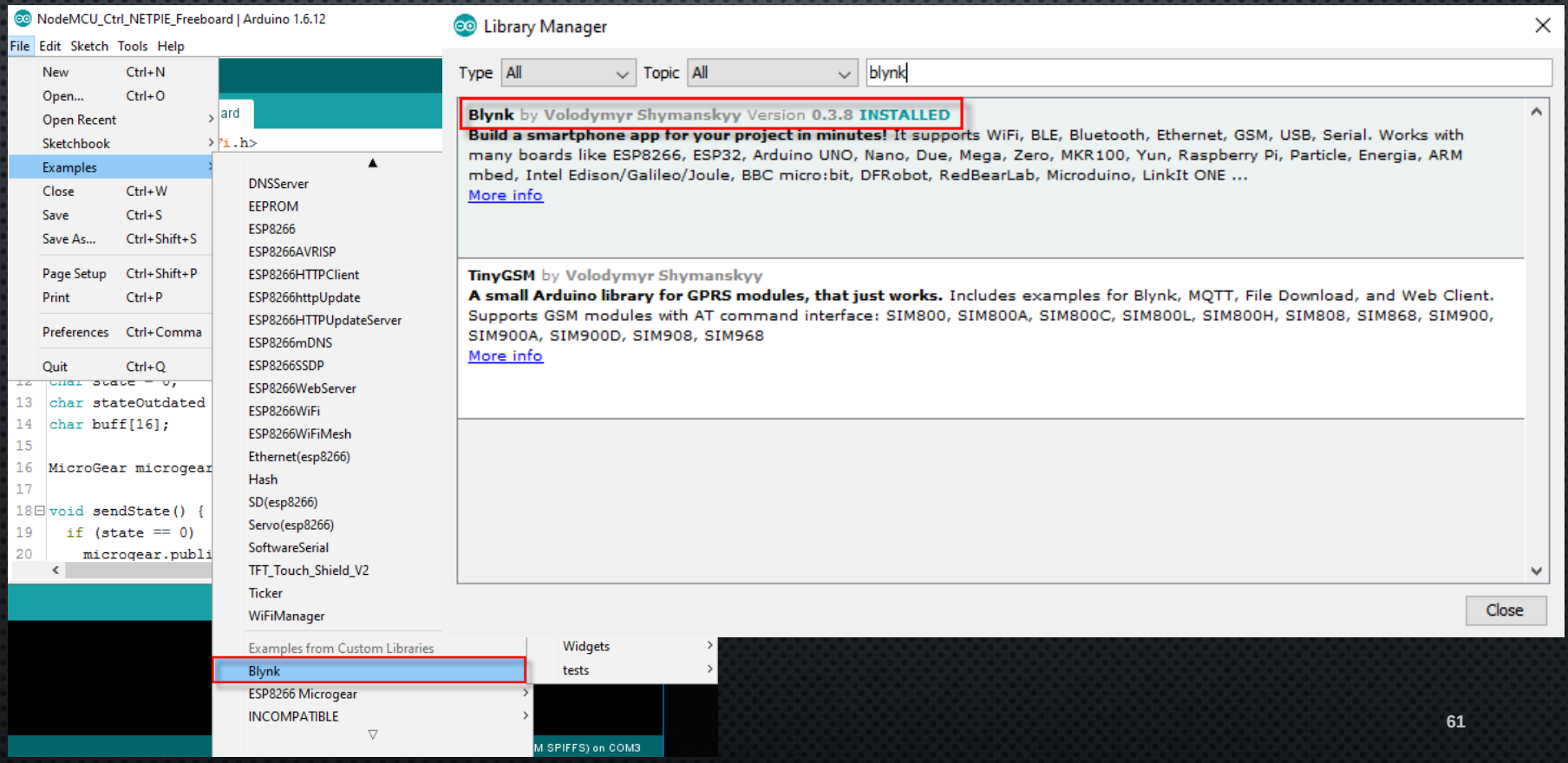
dispatcher@blynk.io

To prarinya_e@yahoo.com

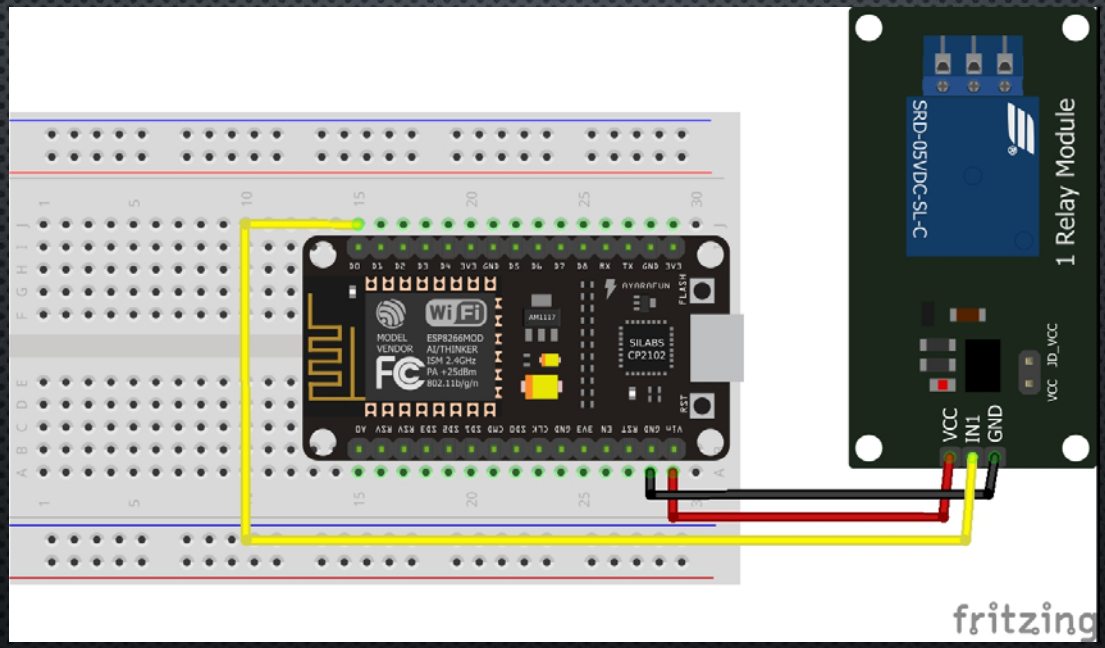
Auth Token for IoT Switch project "7c66a2b373a148b8a9b9470a0da38f87"

Happy Blynking!
-
Getting Started Guide -> http://www.blynk.cc/getting-started
Documentation -> http://docs.blynk.cc/
Latest Blynk library -> https://github.com/blynkkk/blynk-library/releases/download/v0.4.1/Blynk_Release_v0.4.1.zip
Latest Blynk server -> https://github.com/blynkkk/blynk-server/releases/download/v0.20.1/server-0.20.1.jar
-
http://www.blynk.cc
twitter.com/blynk_app
www.facebook.com/blynkapp

Workshop#3 Control IoT via Blynk

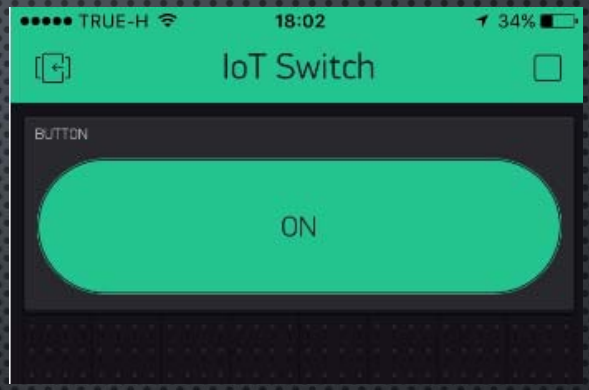


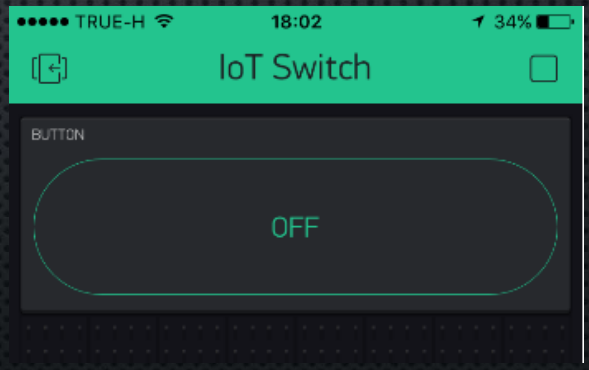
Workshop#3 Control IoT via Blynk



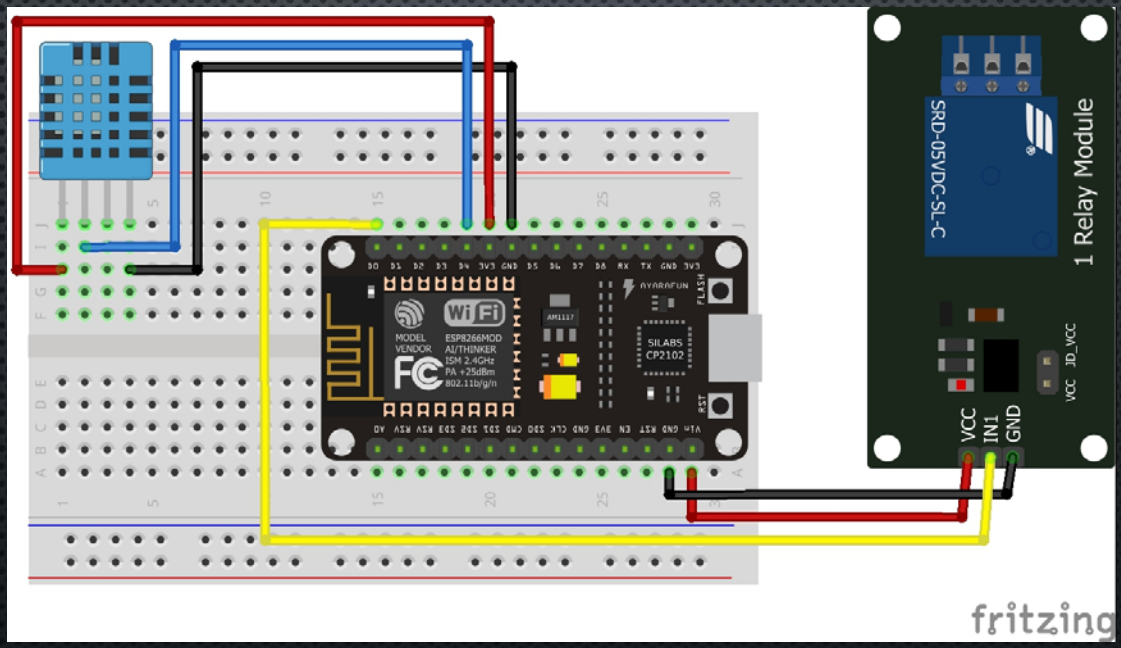
Workshop#3 Control IoT via Blynk

```
#define BLYNK_PRINT Serial    // Comment this out to
disable prints and save space
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "7c66a2b373a148b8a9b9470a0da38f87";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "comptia";
char pass[] = "comptiatc";

void setup()
{
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
}

void loop()
{
  Blynk.run();
}
```





Workshop#3 Temp Control IoT via Blynk



Workshop#4 Control IoT via Blynk

```
#include <ESP8266WiFi.h>
//https://github.com/esp8266/Arduino
//needed for library
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>
//https://github.com/tzapu/WiFiManager

#define BLYNK_PRINT Serial
#include <BlynkSimpleEsp8266.h>
char auth[] = "7c66a2b373a148b8a9b9470a0da38f87"; //
Your Authentication code of blynk account

//for LED status
#include <Ticker.h>
Ticker ticker;

void tick()
{
  //toggle state
  int state = digitalRead(BUILTIN_LED);  // get the current
state of GPIO1 pin
  digitalWrite(BUILTIN_LED, !state);     // set pin to the
opposite state
}

//gets called when WiFiManager enters configuration
mode
void configModeCallback (WiFiManager
*myWiFiManager) {
  Serial.println("Entered config mode");
  Serial.println(WiFi.softAPIP());
  //if you used auto generated SSID, print it
  Serial.println(myWiFiManager->getConfigPortalSSID());
  //Serial.println(myWiFiManager-
>autoConnect("2WaySwitchAP"));
  //entered config mode, make led toggle faster
  ticker.attach(0.2, tick);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);

  //set led pin as output
  pinMode(BUILTIN_LED, OUTPUT);
  // start ticker with 0.5 because we start in AP mode and
try to connect
  ticker.attach(0.6, tick);
}
```

Workshop#4 Control IoT via Blynk

```
 //WiFiManager
  //Local intialization. Once its business is done, there is no
need to keep it around
  WiFiManager wifiManager;
  //reset settings - for testing
  //wifiManager.resetSettings();

  //set callback that gets called when connecting to
previous WiFi fails, and enters Access Point mode
  wifiManager.setAPCallback(configModeCallback);

  //fetches ssid and pass and tries to connect
  //if it does not connect it starts an access point with the
specified name
  //here  "AutoConnectAP"
  //and goes into a blocking loop awaiting configuration
  if (!wifiManager.autoConnect()) {
    Serial.println("failed to connect and hit timeout");
    //reset and try again, or maybe put it to deep sleep
    ESP.reset();
    delay(1000);
  }
  //if you get here you have connected to the WiFi
  Serial.println("connected... :) to " + String(WiFi.SSID()));
  ticker.detach();
  //keep LED on
  digitalWrite(BUILTIN_LED, LOW);
  Blynk.config(auth);
}

void loop() {
  // put your main code here, to run repeatedly:
    Blynk.run();
}
```

QUESTION….

FURTHER RESOURCES

- https://www.arduino.cc/
- https://thingspeak.com/
- https://netpie.io/
- https://espressif.com/en/products/hardware/esp8266ex/overview
- http://www.esp8266.com/
- http://www.blynk.cc/
- https://www.rs-online.com/designspark/rs-toolbox

COURSE FEEDBACK AND INSTRUCTOR EVALUATION

FINAL QUESTION….

Email: prarinya.e@gmail.com


THANK
YOU !