

IoT With ESP8266



Mr.Prarinya Ekapho

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



3

Internet of Things (IoT) Components

Things

Sensors & Controls I/O

Internet connect devices

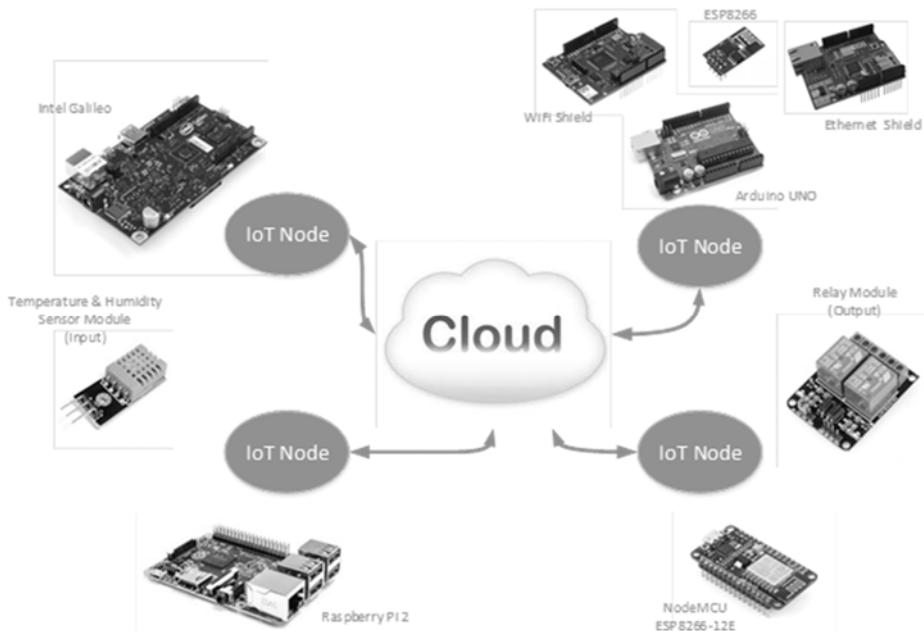
Data

Cloud Service



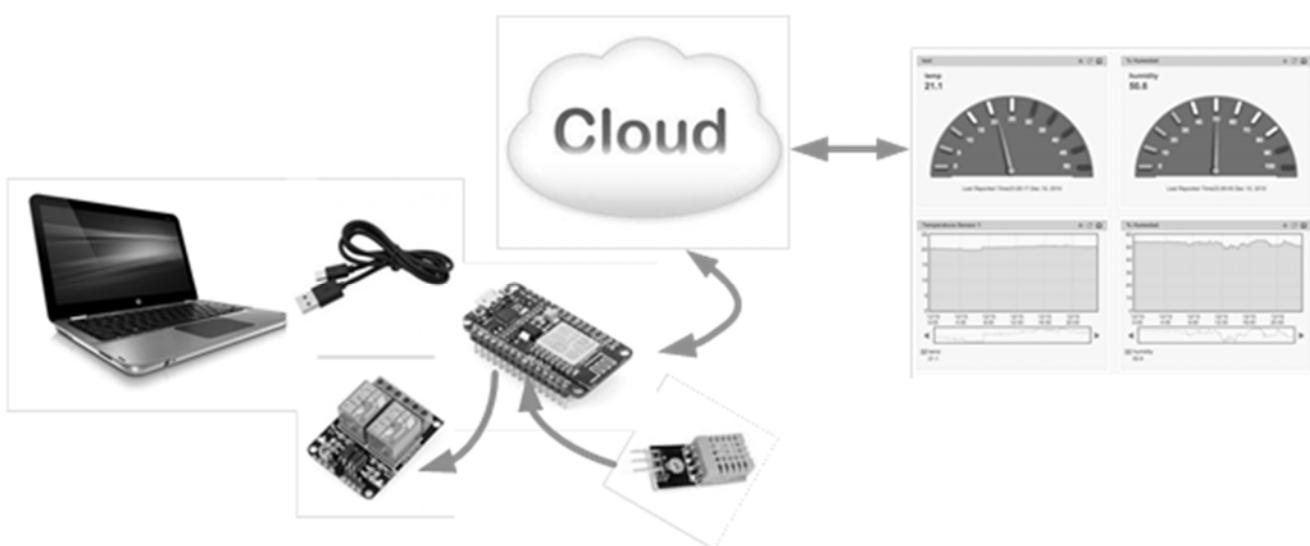
4

IoT Development guide



5

SYSTEM DESIGN CONCEPT

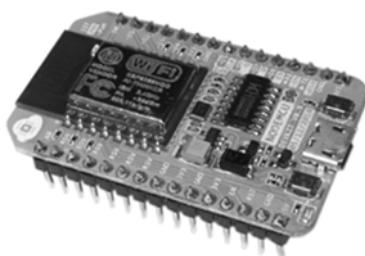


6

About NodeMCU ESP8266



ESP8266(ESP-01)



NodeMCU Devkit 0.9 (ESP-12) Version 1

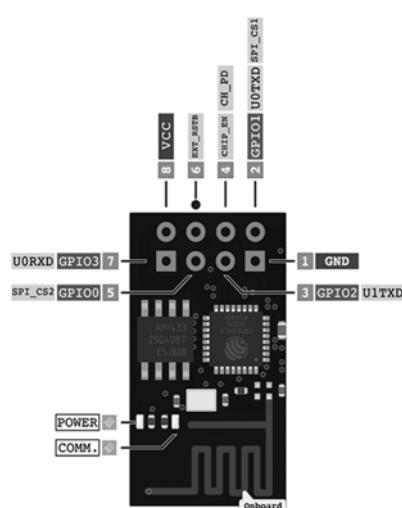


NodeMCU Devkit 1.0 (ESP-12E) Version 2

7

ESP8266 (ESP-01)

ESP-01
PINOUT



POWER	SP. FUNCTION(S)
I/O	COMM. INTERFACE
ADC	PIN NUMBER
CONTROL	PWM
N/C	

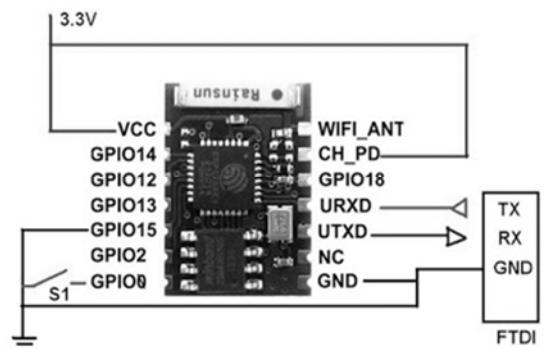
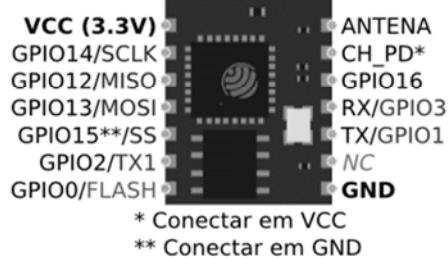
NOTES:
▲ Typ. pin current 6mA (Max. 12mA)
▲ For sleep mode, connect GPIO16 and EXT_RSTB. On wakeup, GPIO16 will output LOW for system reset.
△ On boot/reset/wakeup, keep GPIO15 LOW and GPIO2 HIGH.

8

ESP8266 (ESP-03)

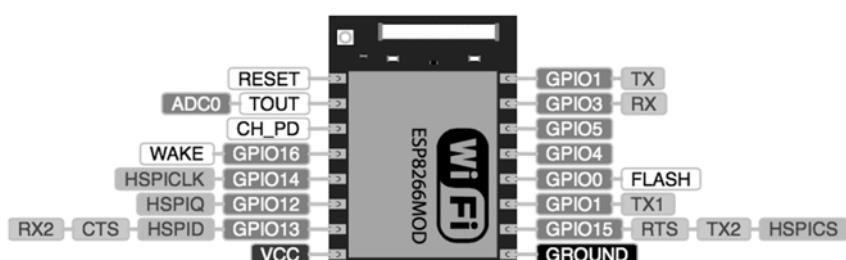


ESP-03



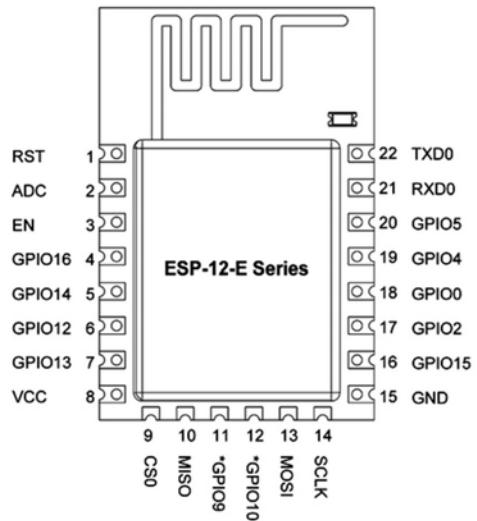
9

ESP8266 (ESP-07)



10

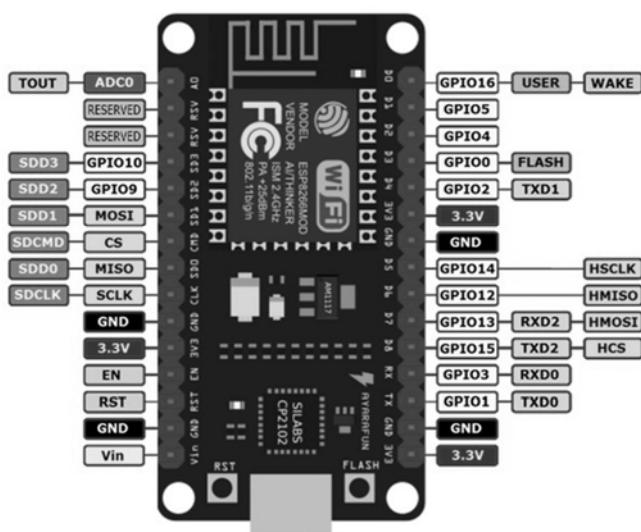
ESP8266 (ESP-12E)



11

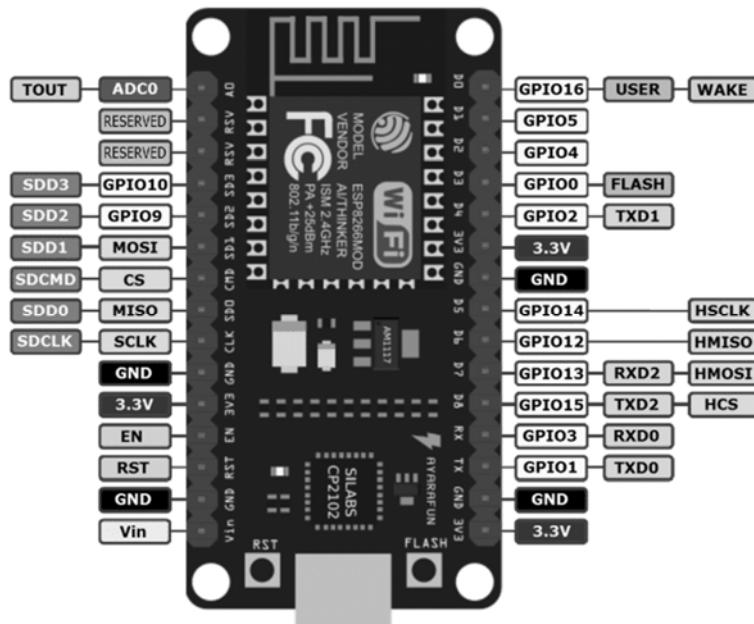
ESP8266 NodeMCU V1 Pin layout

NodeMCU ESP12 Dev Kit V1.0 Pin Definition:



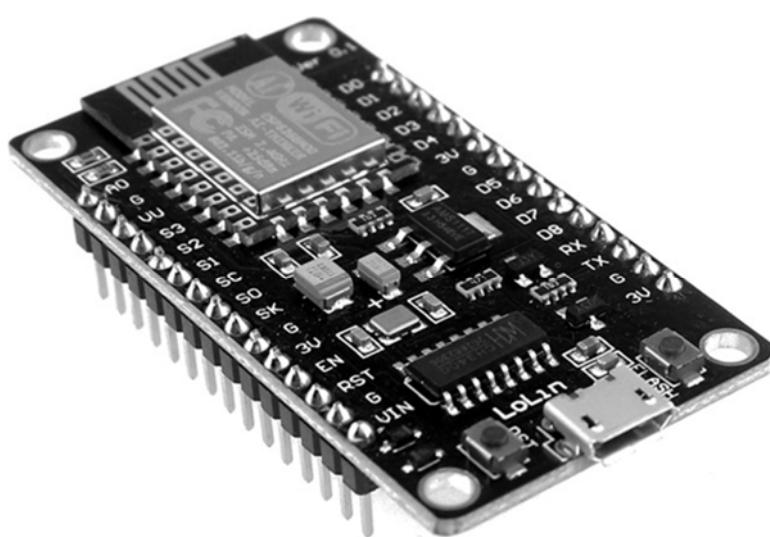
12

ESP8266 NodeMCU V2 Pin layout



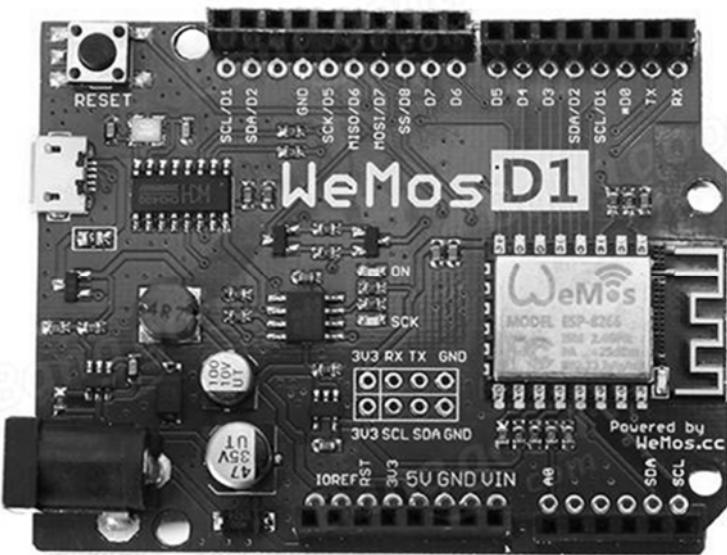
13

ESP8266 NodeMCU V3 Pin layout



14

WeMos D1



15

WeMos D1 Mini



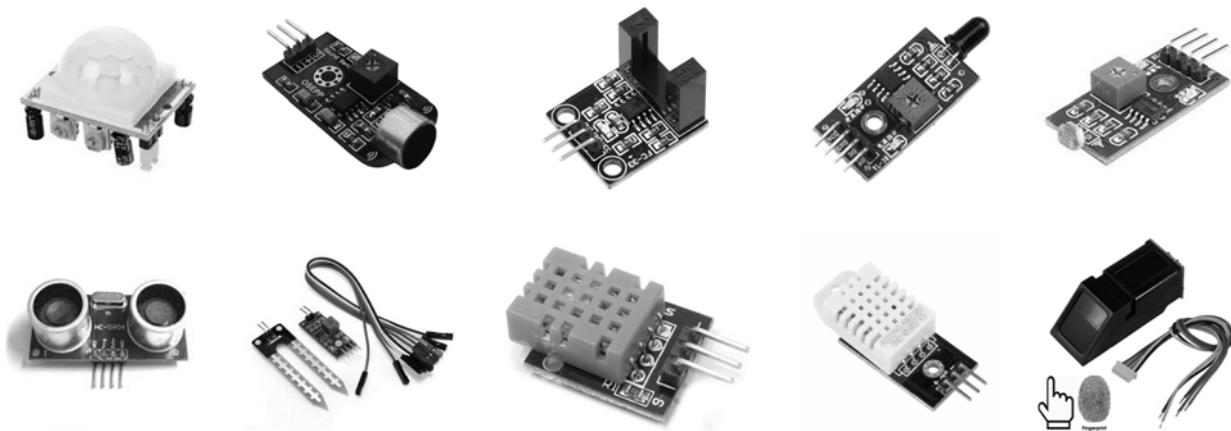
16

Witty cloud Mini NodeMCU



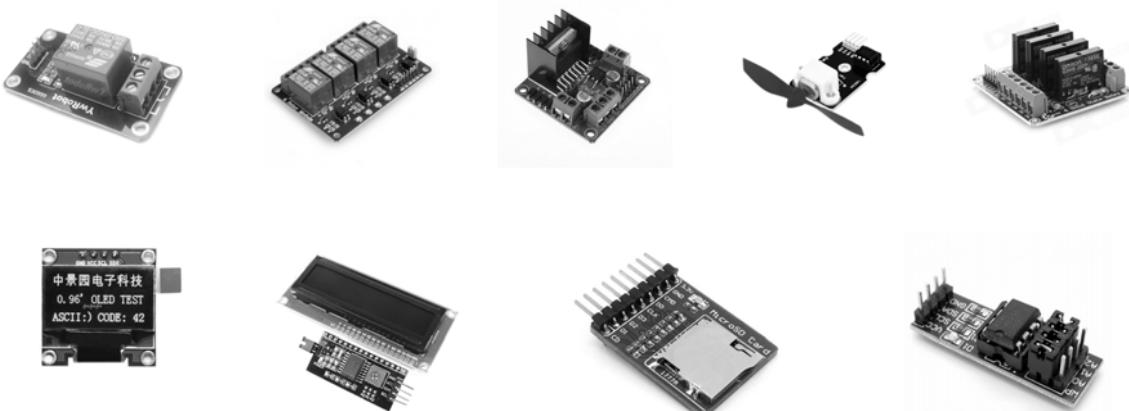
17

SENSOR & CONTROL MODULES



18

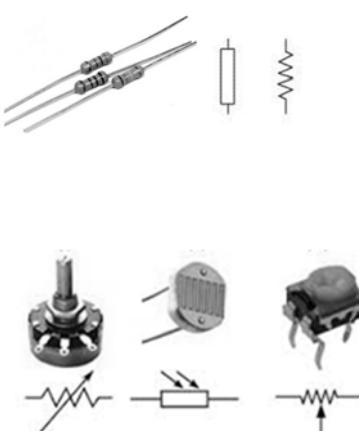
SENSOR & CONTROL MODULES



19

BASIC ELECTRONIC DEVICES

Resistor



RESISTOR COLOR CODE GUIDE

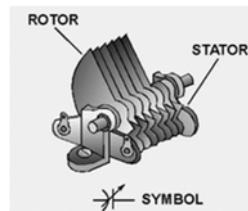
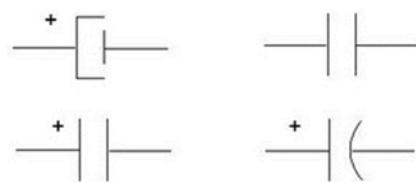
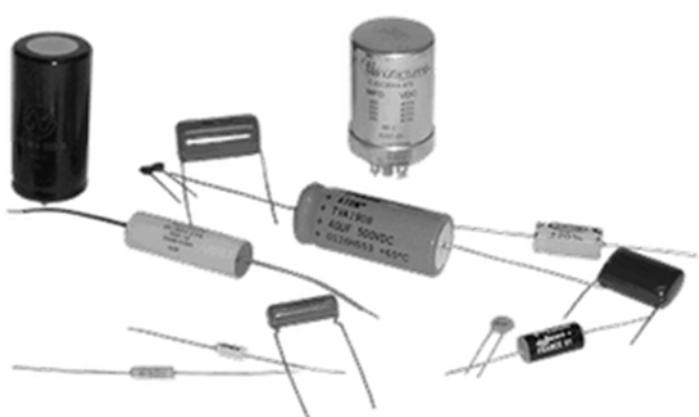
The diagram illustrates the resistor color code for both 4-band and 5-band resistors. For a 4-band resistor, the first four bands represent the value, the fifth band represents the decimal multiplier, and the sixth band represents the tolerance. For a 5-band resistor, the first five bands represent the value, and the sixth band represents the tolerance. The chart provides a reference for the color codes and their corresponding numerical values.

Color	1st Band	2nd Band	3rd Band	Decimal Multiplier	Tolerance
Black	0	0	0	1	1 ± 1%
Brown	1	1	1	10	10 ± 2%
Red	2	2	2	100	100 ± 5%
Orange	3	3	3	1K	1,000 ± 10%
Yellow	4	4	4	10K	10,000 ± 20%
Green	5	5	5	100K	100,000 ± 10%
Blue	6	6	6	1M	1,000,000 ± 5%
Violet	7	7	7	10M	10,000,000 ± 2%
Gray	8	8	8	100M	100,000,000 ± 1%
White	9	9	9	1G	1,000,000,000 ± 0.5%
Gold					0.1 ± 5%
Silver					0.01 ± 10%
None					± 20%

20

BASIC ELECTRONIC DEVICES

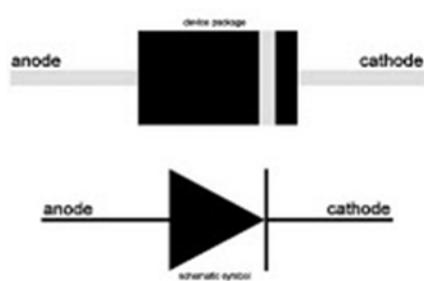
Capacitor



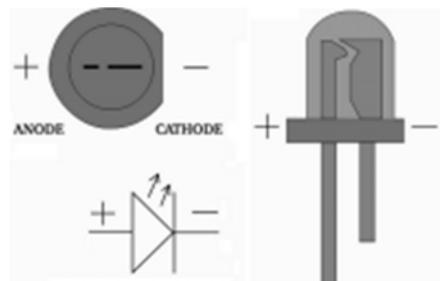
21

BASIC ELECTRONIC DEVICES

Diode



LED



22



23

Tools for Develop

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

24

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

25

CP2102 USB to UART driver

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



Download Software

The CP210x Manufacturing DLL and Runtime DLL have been updated and must be used with v6.0 and later of the CP210x Windows VCP Driver. Application Note Software downloads affected are AN144SW.zip, AN205SW.zip and AN223SW.zip. If you are using a 5.x driver and need support you can download archived Application Note Software.

- Windows 7/8/8.1/10 (v6.7.3) »
- Windows XP/Server 2003/Vista/7/8/8.1 (v6.7) »
- Windows 2K (v6.3a) »
- WinCE »
- Macintosh OSX (v4) »
- Linux »
- Android »

26

Workshop#1

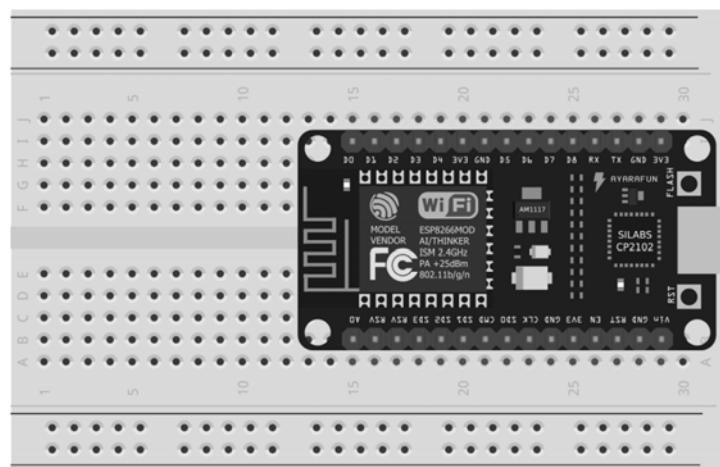
Start learning ESP8266 with Arduino IDE

- BLINK
- WiFi SCAN
- WEB SERVER

27

Workshop#1

Prepare hardware snap in breadboard



28

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 active 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)
}
```

29

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);

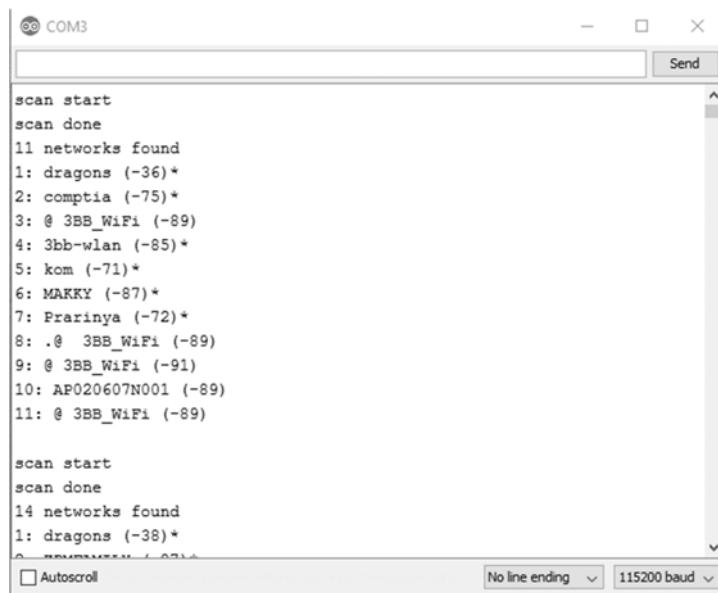
    Serial.println("Setup done");
}

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);
}
```

30

Workshop#1 [Wifi scan] – serial monitor



The screenshot shows a Windows-style serial monitor window titled "COM3". It displays two separate WiFi scan results. The first scan found 11 networks, including "dragons (-36)*", "comptia (-75)*", and "3BB_WIFI (-89)". The second scan found 14 networks, including "dragons (-38)*". The window includes standard controls like "Send", "No line ending", and "115200 baud".

```
scan start
scan done
11 networks found
1: dragons (-36)*
2: comptia (-75)*
3: @ 3BB_WIFI (-89)
4: 3bb-wlan (-85)*
5: kom (-71)*
6: MAKKY (-87)*
7: Prarinya (-72)*
8: .@ 3BB_WIFI (-89)
9: @ 3BB_WIFI (-91)
10: AP020607N001 (-89)
11: @ 3BB_WIFI (-89)

scan start
scan done
14 networks found
1: dragons (-38)*
```

31

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());
}
```

32

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\n<GPIO is now ";
  s += (val)? "high": "low";
  s += "</html>\r\n";

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

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

33

Workshop#1 [wifiWeb server] – sketch GPIO5

```
#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() {
  WiFi.begin(ssid, password);

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

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

// 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());
```

34

Workshop#1 [wifiWeb server] – sketch GPIO5

```
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 GPIO5 according to the request
    digitalWrite(5, val);

    client.flush();

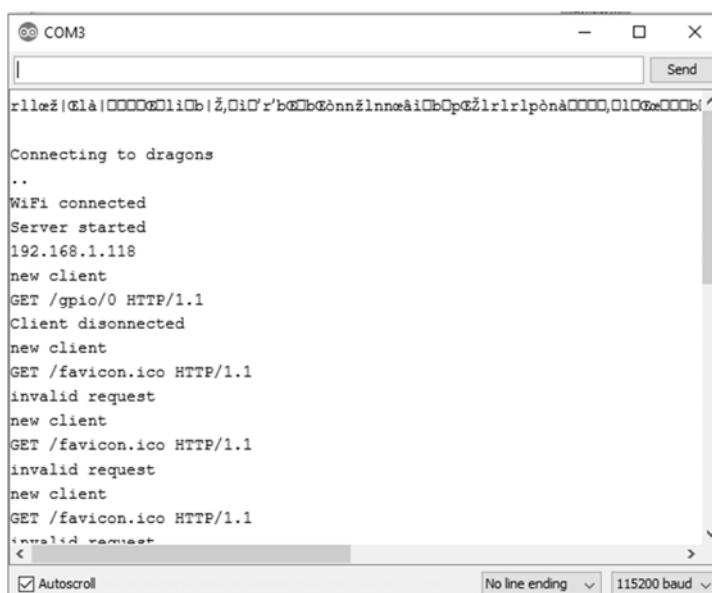
    // Prepare the response
    String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n<GPIO is now ";
    s += (val) ? "high" : "low";
    s += "</html>\r\n";

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

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

35

Workshop#1 [wifiWeb server] – serial monitor



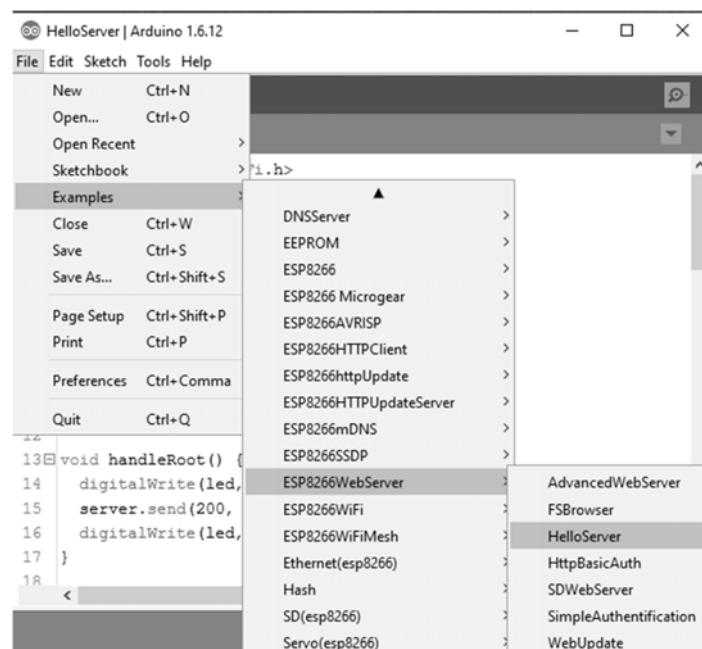
36

Workshop#1 [wifiWeb server] – web browser



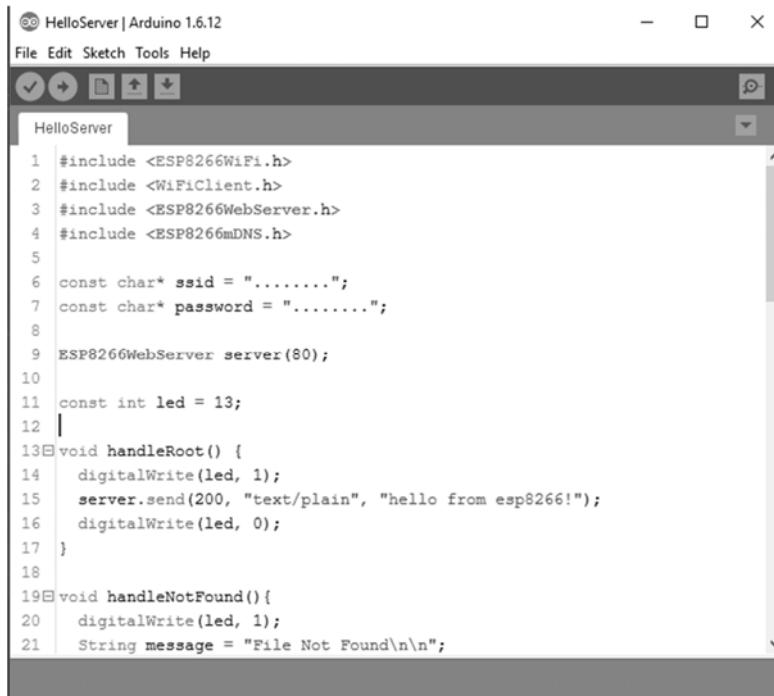
37

Workshop#1 [helloserver]



38

Workshop#1 [helloserver]



The screenshot shows the Arduino IDE interface with the title bar "HelloServer | Arduino 1.6.12". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, refresh, and other functions. The main code editor window contains the following C++ code:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

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

ESP8266WebServer server(80);

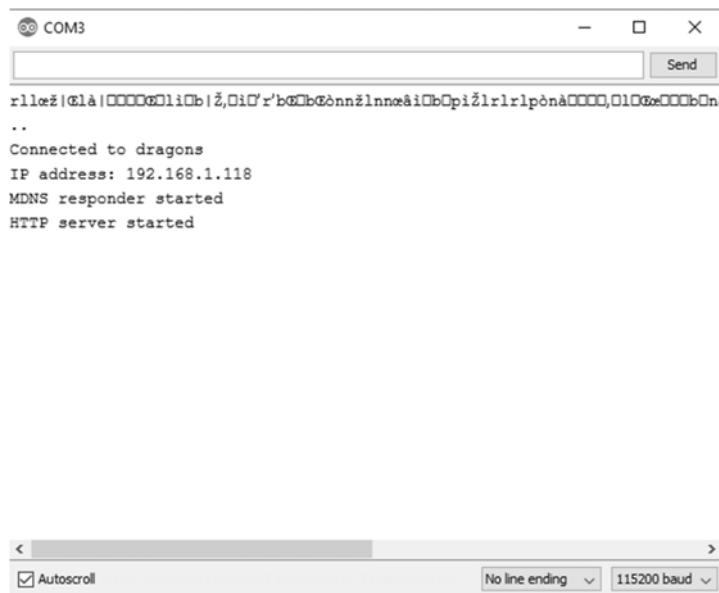
const int led = 13;

void handleRoot() {
    digitalWrite(led, 1);
    server.send(200, "text/plain", "hello from esp8266!");
    digitalWrite(led, 0);
}

void handleNotFound() {
    digitalWrite(led, 1);
    String message = "File Not Found\n\n";
}
```

A small number "39" is visible in the bottom right corner of the code editor.

Workshop#1 [helloserver]-serial monitor

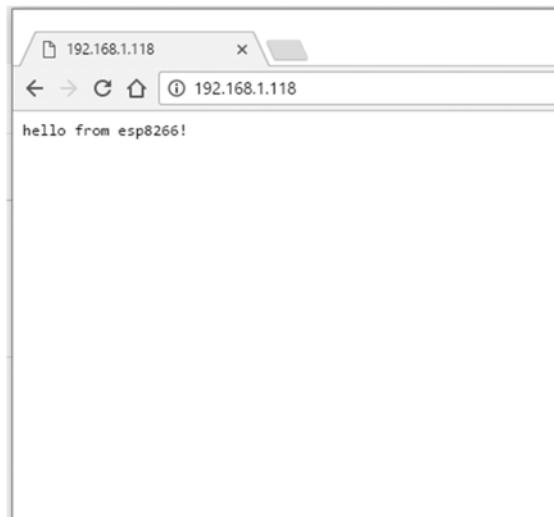


The screenshot shows the Serial Monitor window titled "COM3". The window displays the following text output:

```
rllœž|Glà|000000li0b|ž,0i0'r'b00bGonnžlnnøæi0bOpížrlrlpønå0000,010æe0000bOna
..
Connected to dragons
IP address: 192.168.1.118
MDNS responder started
HTTP server started
```

At the bottom of the window, there are controls for "Autoscroll" (checked), "No line ending", and "115200 baud".

Workshop#1 [helloserver]-web browser



41

Workshop#2

upload data to cloud (thinkspeak)

- Apply Thinkspeak
- Test send data
- Customize visual board

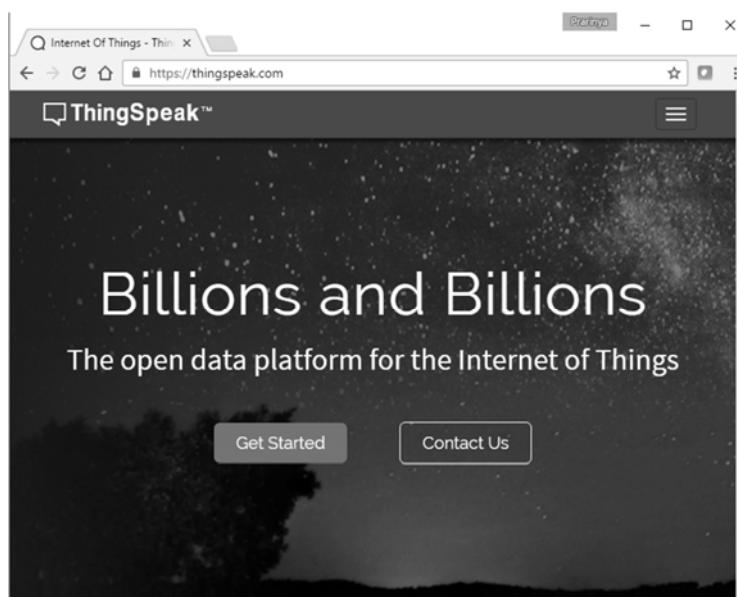
42

Workshop#2 Apply Thinkspeak

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

43

Workshop#2 Apply Thinkspeak



44

Workshop#2 Apply Thinkspeak

User ID

Password

[Forgot your password?](#)

Remember my User ID

[Sign In](#)

[Sign In With MathWorks Account](#)

Don't have a ThingSpeak account? [Sign Up](#)

45

Workshop#2 Apply Thinkspeak

Sign up to start using ThingSpeak

User ID

Email

Time Zone

Password

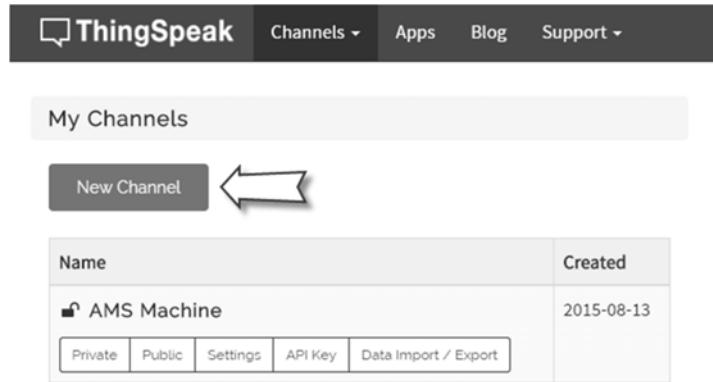
Password Confirmation

By signing up, you agree to the [Terms of Use](#) and [Privacy Policy](#).

[Create Account](#)

46

Workshop#2 Create New Channel



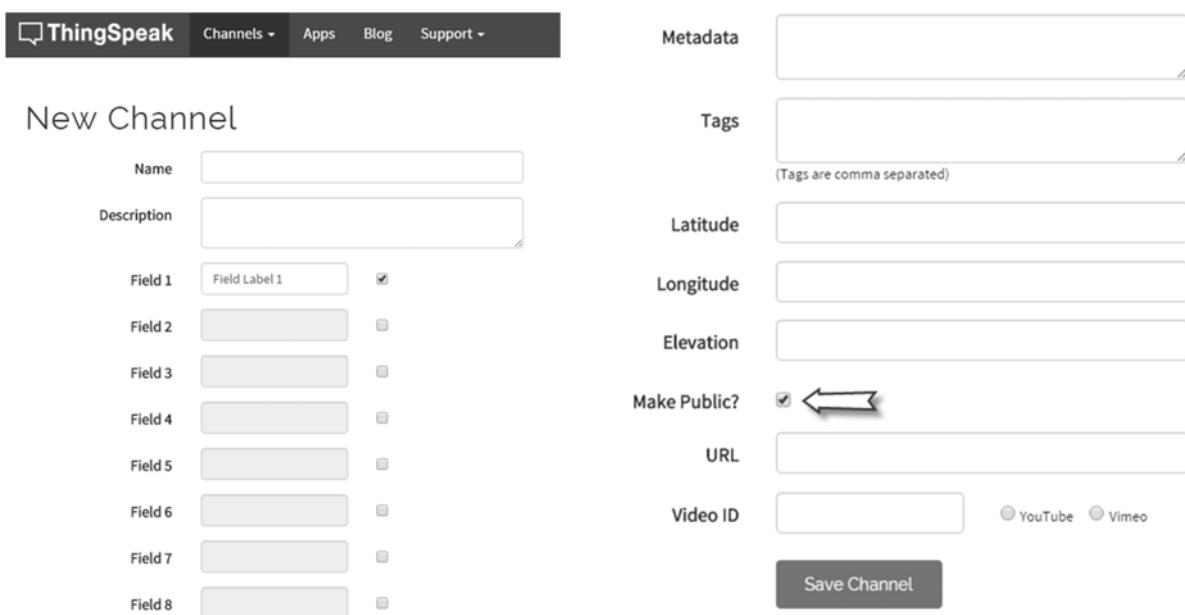
The screenshot shows the ThingSpeak interface. At the top is a navigation bar with the logo, 'Channels', 'Apps', 'Blog', and 'Support'. Below it is a section titled 'My Channels' containing a table. The table has two columns: 'Name' and 'Created'. One row shows a channel named 'AMS Machine' created on '2015-08-13'. Below the table are five buttons: 'Private', 'Public', 'Settings', 'API Key', and 'Data Import / Export'. A white arrow points from the 'New Channel' button on the left to the 'AMS Machine' row.

Name	Created
AMS Machine	2015-08-13

New Channel

47

Workshop#2 Create New Channel



The screenshot shows the 'New Channel' creation form. At the top is a navigation bar with the logo, 'Channels', 'Apps', 'Blog', and 'Support'. The main form area has sections for 'Metadata' (with a note '(Tags are comma separated)'), 'Tags' (with a note '(Tags are comma separated)'), 'Latitude', 'Longitude', 'Elevation', and 'URL'. There are also fields for 'Video ID' and checkboxes for 'YouTube' and 'Vimeo'. On the left, there's a list of 'Field 1' through 'Field 8' each with a 'Field Label' input and a checked checkbox. In the center, there's a 'Make Public?' checkbox with a checked value and a white arrow pointing to the 'Save Channel' button at the bottom right. The 'Save Channel' button is highlighted with a dark grey background.

New Channel

Name

Description

Field 1 Field Label 1

Field 2

Field 3

Field 4

Field 5

Field 6

Field 7

Field 8

Metadata

Tags
(Tags are comma separated)

Latitude

Longitude

Elevation

URL

Video ID

YouTube Vimeo

Save Channel

48

Workshop#2 Create New Channel

Data Import / Export

Update Channel Feed - GET
GET https://api.thingspeak.com/update?api_key=G7EEA1WJNU7CAG7W&field1=0

Update Channel Feed - POST
POST <https://api.thingspeak.com/update.json>
api_key=G7EEA1WJNU7CAG7W
field1=73

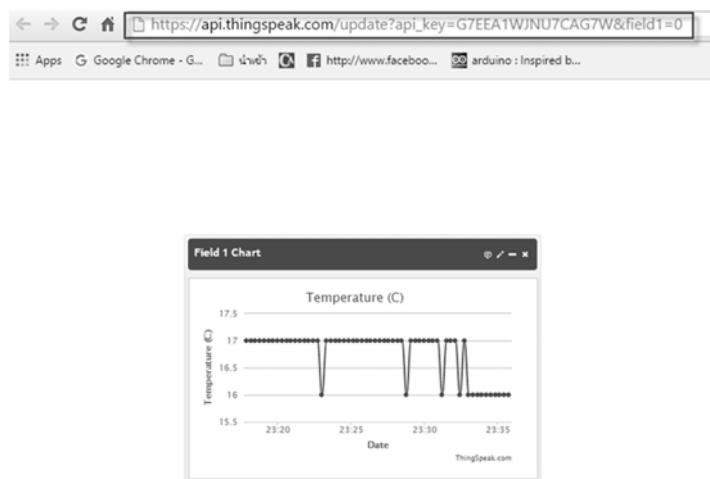
Get a Channel Feed
GET <https://api.thingspeak.com/channels/51581/feeds.json?results=2>

Get a Channel Field Feed
GET <https://api.thingspeak.com/channels/51581/fields/1.json?results=2>

Get Status Updates
GET <https://api.thingspeak.com/channels/51581/status.json>

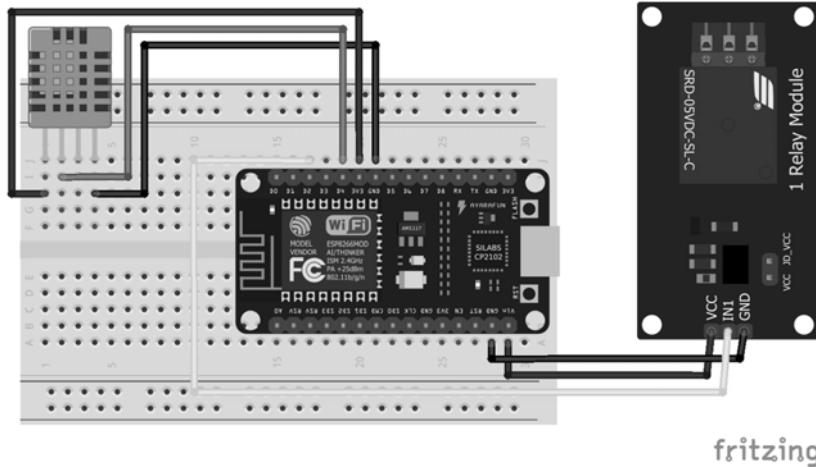
49

Workshop#2 Create New Channel



50

Workshop#2 connect hardware



51

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

#ifndef 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();
}
```

52

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!"));
}
```

53

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("[dht1] 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;
    }
}
```

54

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\nt");
DEBUG_PRINT("Heat index: ");
DEBUG_PRINT(hi);
DEBUG_PRINTLN(" *F");

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

55

Customize visual board



The image shows the top navigation bar of the ThingSpeak website. It includes the logo "ThingSpeak™", links for "Channels", "Apps", "Community", and "Support", and buttons for "How to Buy", "Account", and "Sign Out".

Apps

ThingSpeak channels store data. Upload data from the web or send data from devices to a ThingSpeak channel. Use these apps to transform and visualize data or trigger an action. See Tutorial: ThingSpeak and MATLAB to create a channel. Learn more about MATLAB® inside ThingSpeak.

Analytics



MATLAB Analysis
Explore and transform data.



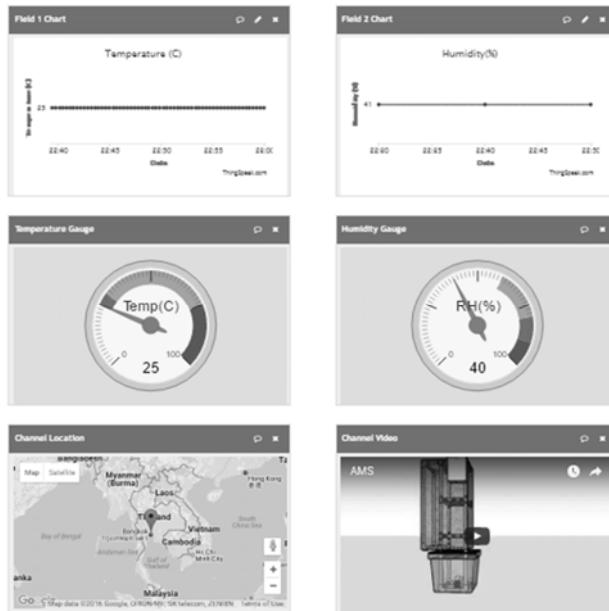
MATLAB Visualizations
Visualize data in MATLAB plots.



Plugins
Display data in gauges, charts, or custom plugins.

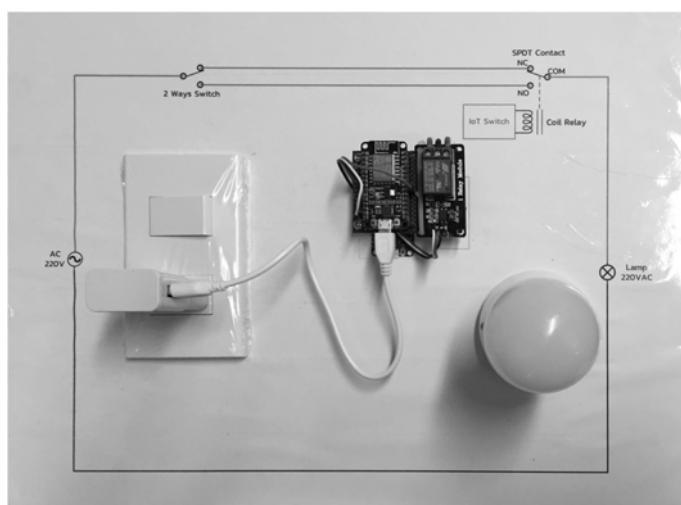
30

Customize visual board



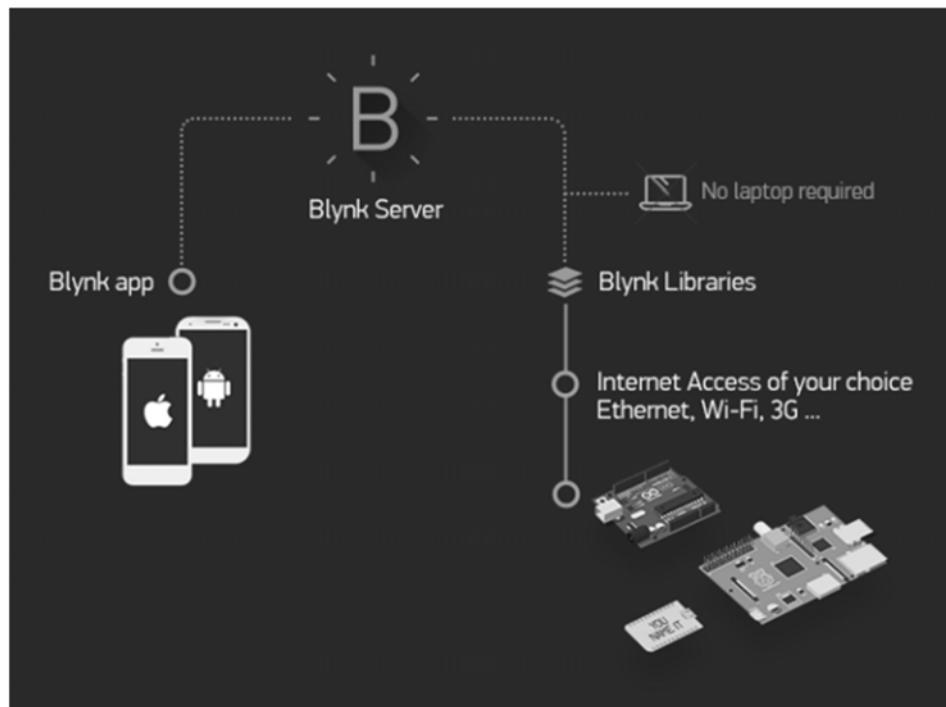
57

Workshop#3 Mini Project 2Way IoT Switch



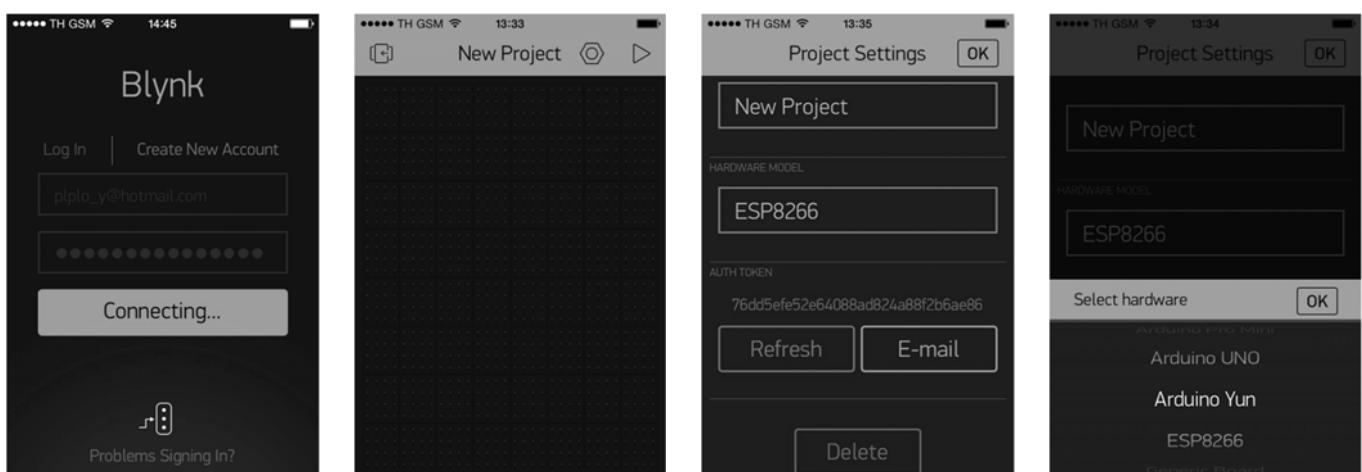
58

Workshop#3 Control IoT via Blynk



59

Workshop#3 Control IoT via Blynk



60

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>

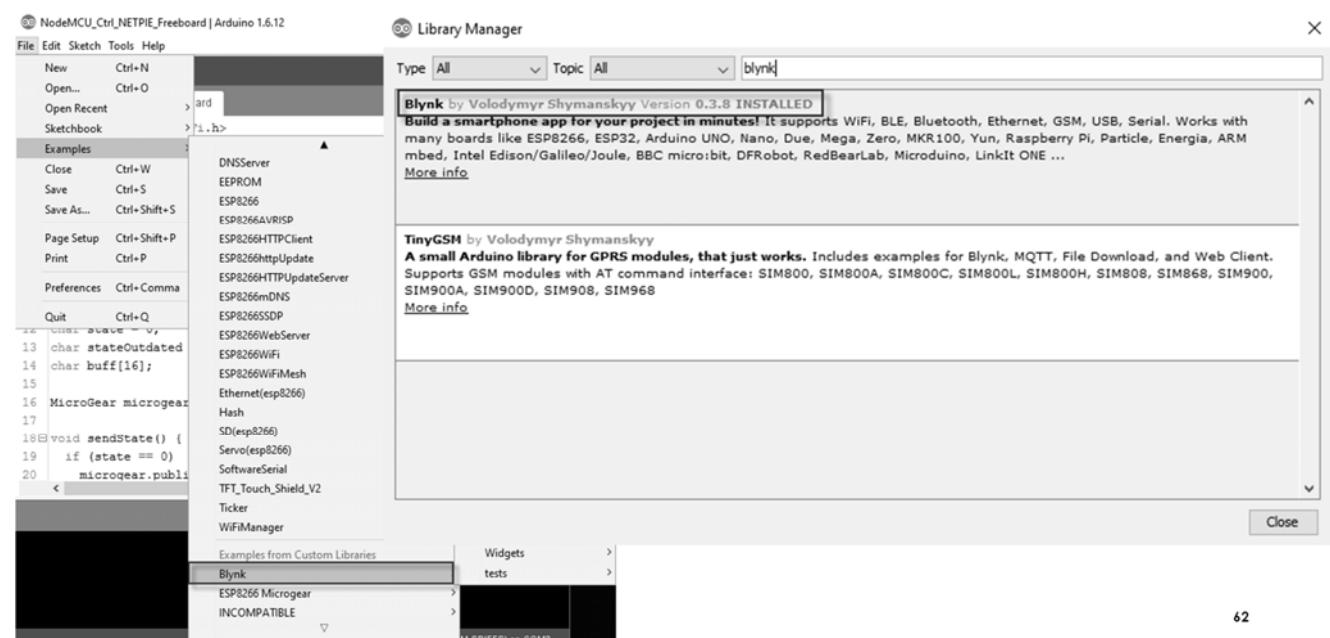
- <http://www.blynk.cc>

twitter.com/blynk_app

www.facebook.com/blynkapp

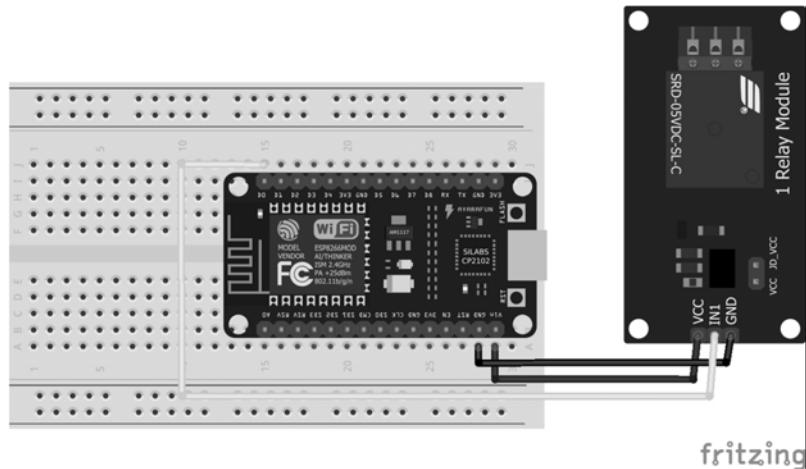
61

Workshop#3 Control IoT via Blynk



62

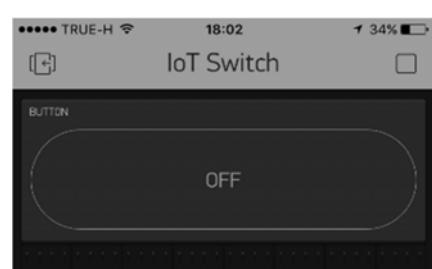
Workshop#3 Control IoT via Blynk



63

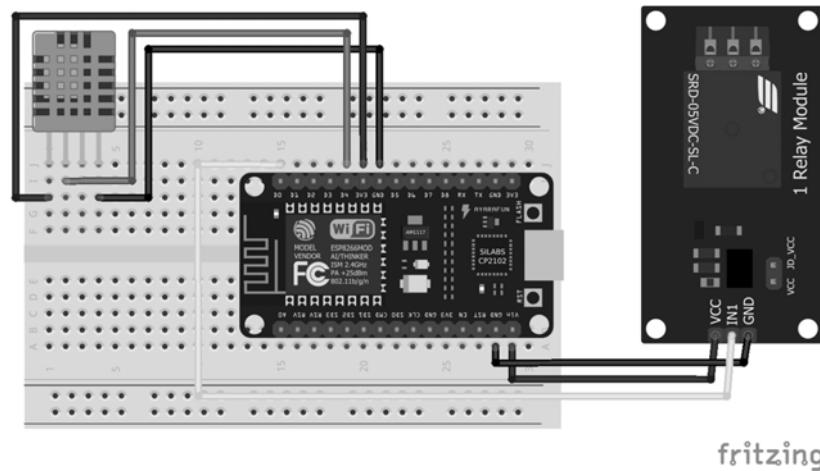
Workshop#3 Control IoT via Blynk

```
#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[] = "comptiatic";  
  
void setup()  
{  
    Serial.begin(115200);  
    Blynk.begin(auth, ssid, pass);  
}  
  
void loop()  
{  
    Blynk.run();  
}
```



64

Workshop#3 Temp Control IoT via Blynk



65

Workshop#4 Control IoT via Blynk

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

#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN 12
#define DHTTYPE DHT11

DHT_Unified dht(DHTPIN, DHTTYPE);

char auth[] = "8c028718194743109fd45f2fb720cc14";

SimpleTimer timer;
void setup()
{
  Serial.begin(115200);
  Blynk.begin(auth, "comptia", "comptiatac");
  dht.begin();
}

// Setup a function to be called every second
timer.setInterval(5000L, sendUptime);
}

void sendUptime()
{
  Blynk.virtualWrite(V5, millis() / 1000);
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (!isnan(event.temperature)) {
    Blynk.virtualWrite(V1, event.temperature);
  }
  dht.humidity().getEvent(&event);
  if (!isnan(event.relative_humidity)) {
    Blynk.virtualWrite(V2, event.relative_humidity);
  }
}

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

66

QUESTION....



67

FURTHER RESOURCES

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

68

COURSE FEEDBACK AND INSTRUCTOR EVALUATION



- EXCELLENT
- GOOD
- AVERAGE
- POOR

69

FINAL QUESTION....



Email: prarinya.e@gmail.com

70

