

IoT With ESP8266



Mr.Prarinya Ekapho

Agenda for 2 days training

Day1

- IoT Concept and Applications
- Workshop#1 (Start with NodeMCU ESP8266)
- Workshop#2 (Use platform Thinkspeak)

Day2

- Workshop#3 (Use platform NETPIE)
- Workshop#4 (Use others application to control IoT)
- Conclude and guide to development

IoT Concept



3

Internet of Things (IoT) Components

Things

Sensors & Controls I/O

Internet connect devices

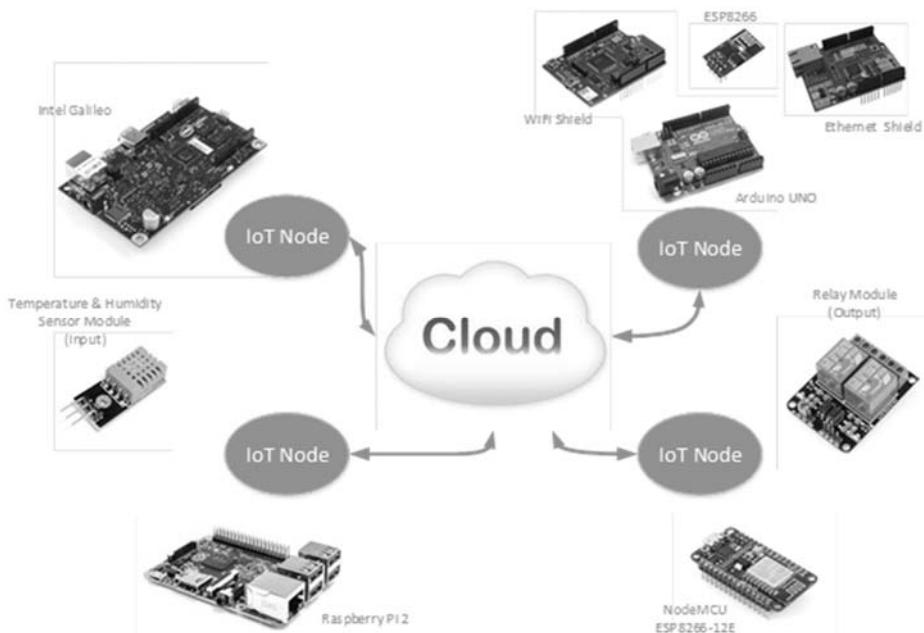
Data

Cloud Service



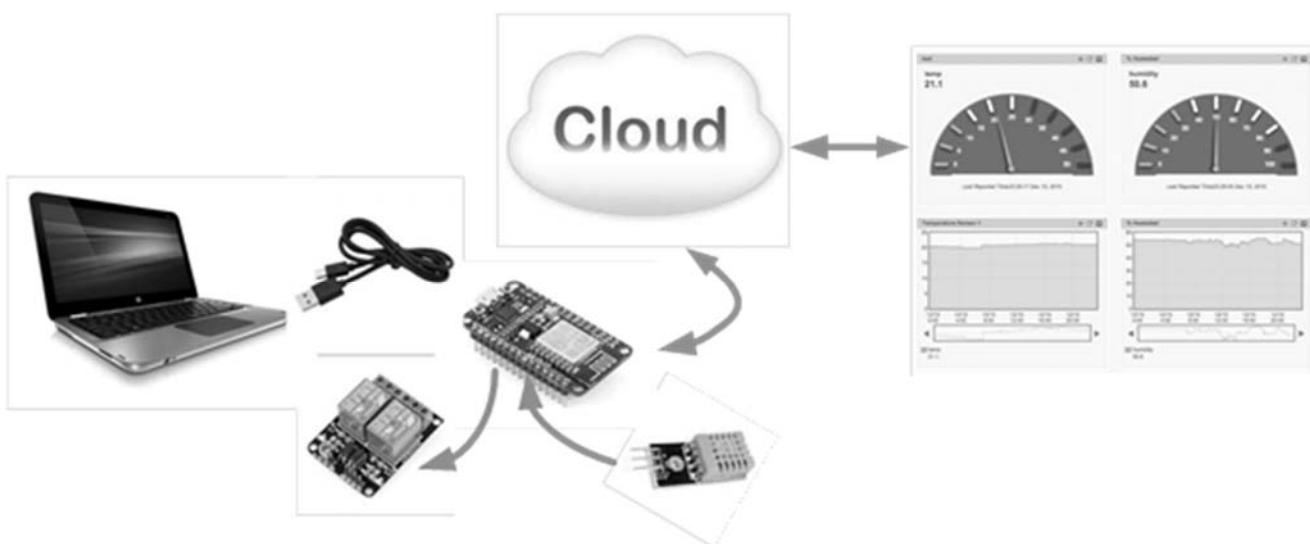
4

IoT Development guide



5

System Design concept

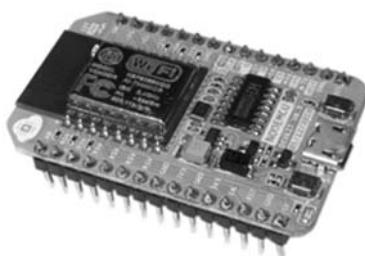


6

About NodeMCU ESP8266



ESP8266(ESP-01)



NodeMCU Devkit 0.9 (ESP-12) Version 1

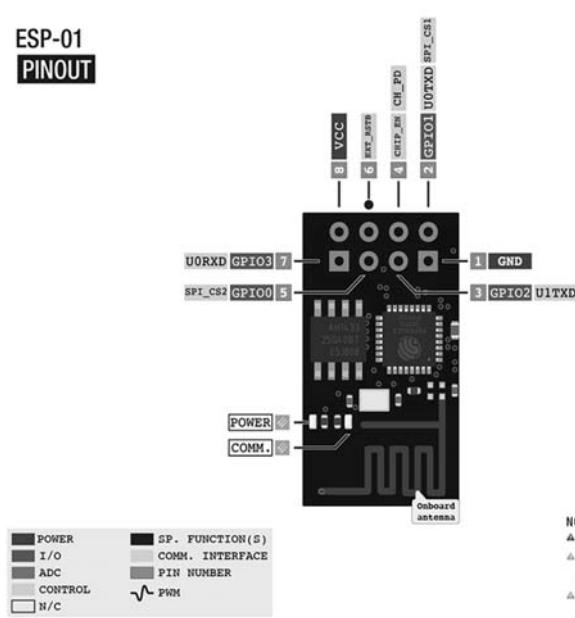


NodeMCU Devkit 1.0 (ESP-12E) Version 2

7

ESP8266 (ESP-01)

ESP-01
PINOUT



03-31-2016
v1.0

NOTES:

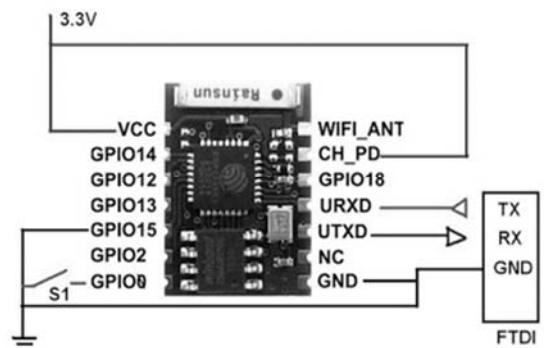
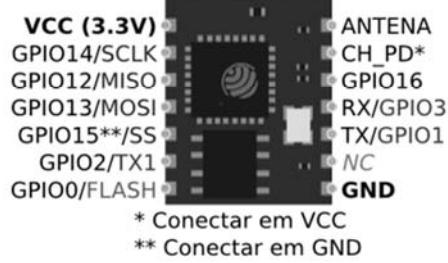
- ▲ Typ. pin current 6mA (Max. 12mA)
- ▲ For sleep mode, connect GPIO16 and EXT_RSTB. On wakeup, GPIO16 will output LOW for system reset.
- ▲ On boot/reset/wakeup, keep GPIO15 LOW and GPIO2 HIGH.

8

ESP8266 (ESP-03)

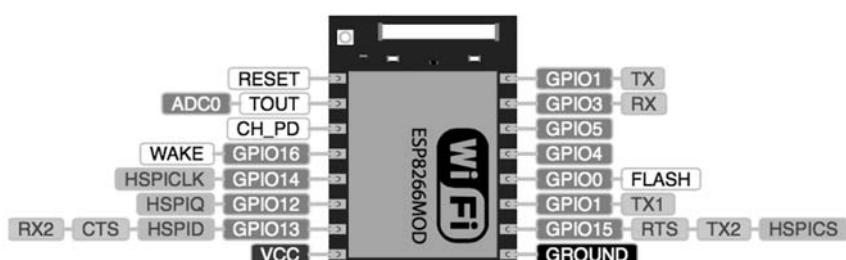


ESP-03



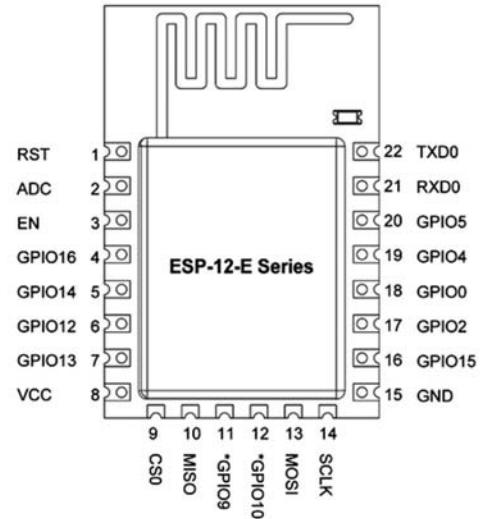
9

ESP8266 (ESP-07)



10

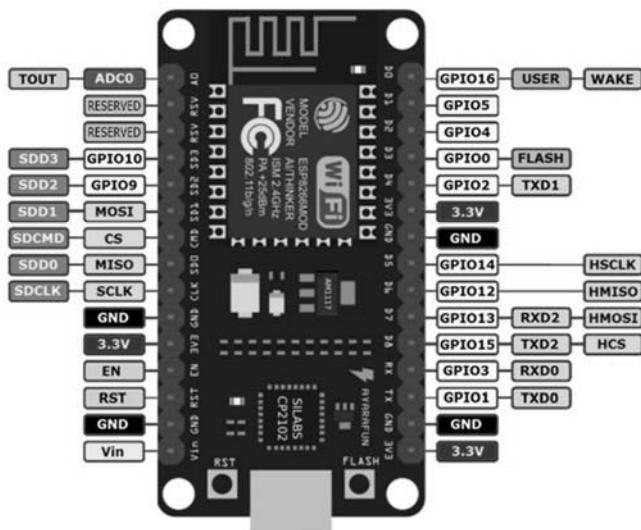
ESP8266 (ESP-12E)



11

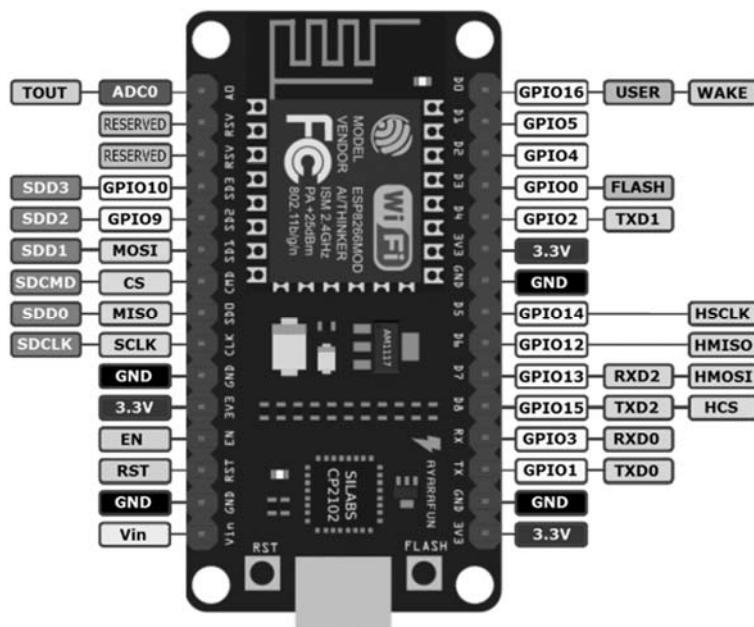
ESP8266 NodeMCU V1 Pin layout

NodeMCU ESP12 Dev Kit V1.0 Pin Definition:



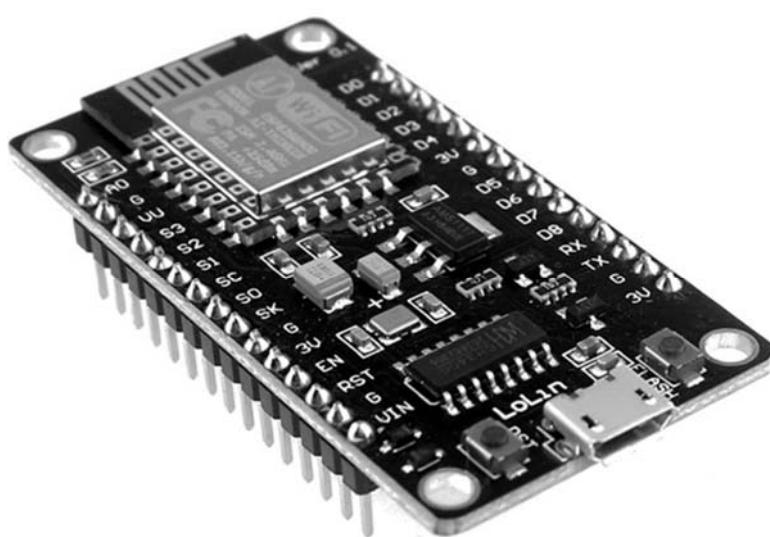
12

ESP8266 NodeMCU V2 Pin layout



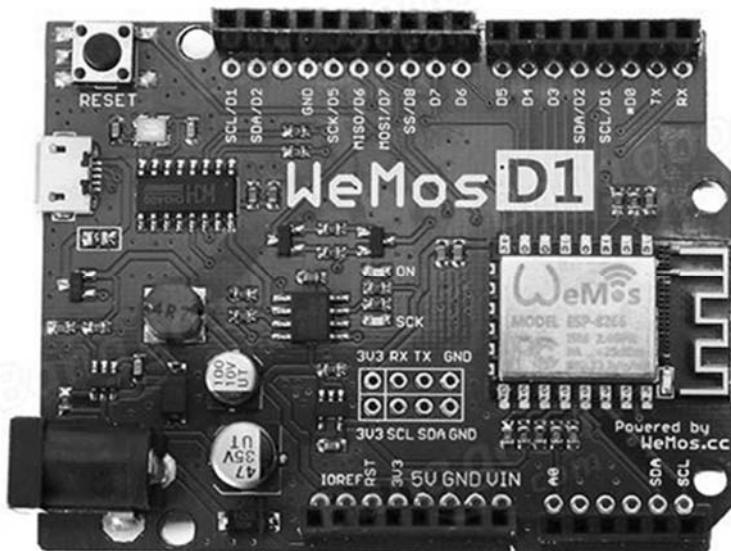
13

ESP8266 NodeMCU V3 Pin layout



14

WeMos D1



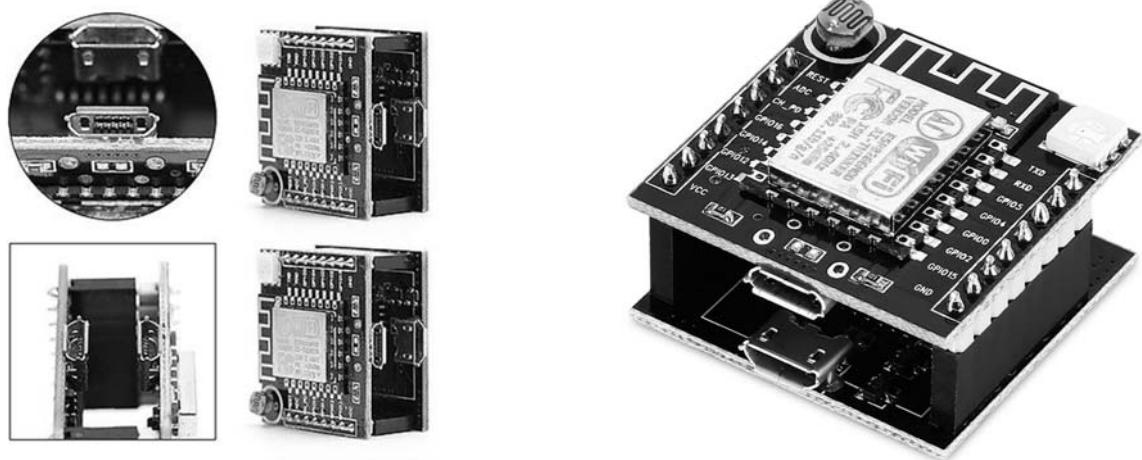
15

WeMos D1 Mini



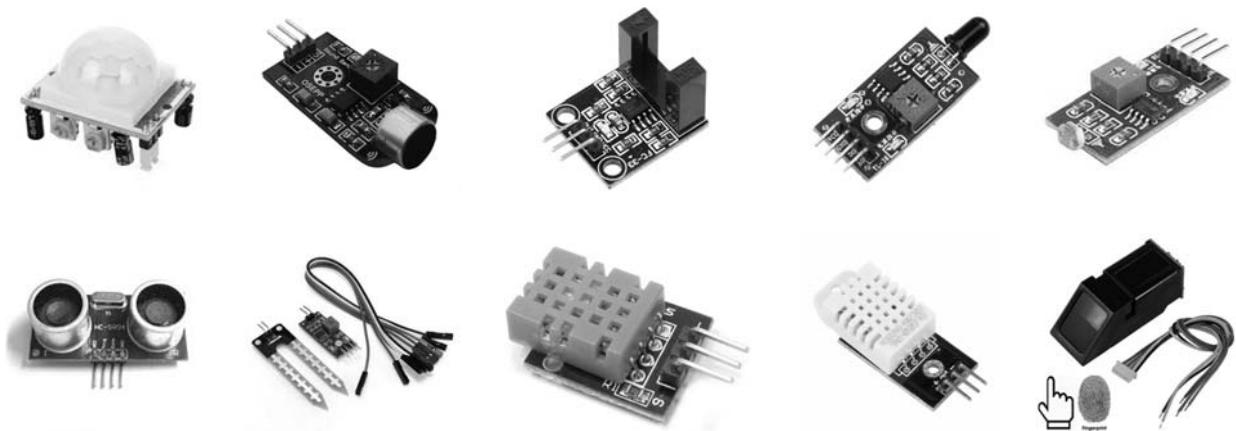
16

Witty cloud Mini NodeMCU



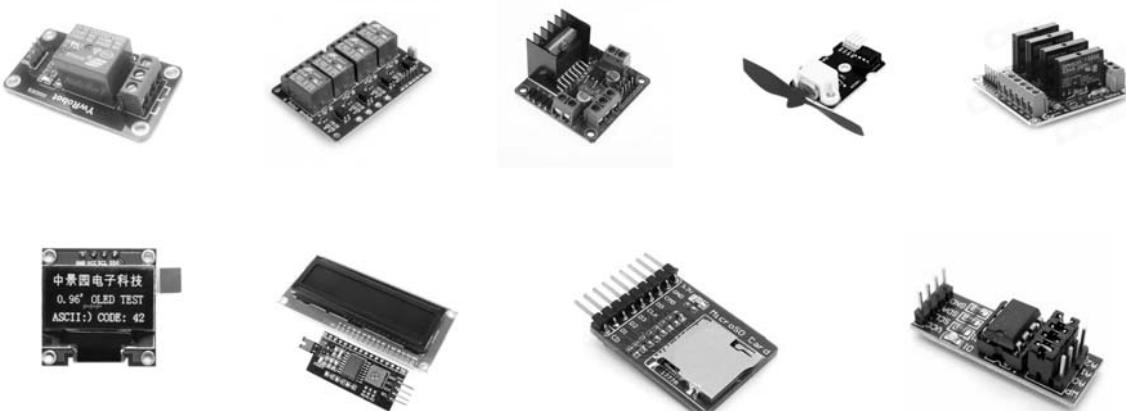
17

Sensor & Control modules



18

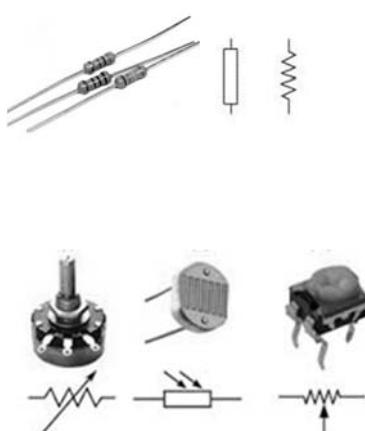
Sensor & Control modules



19

Basic electronic devices

Resistor



RESISTOR COLOR CODE GUIDE

A diagram of a resistor with four color bands. The bands are labeled 1st, 2nd, 3rd, and 4th from left to right. The colors correspond to the values: 1st band (Black), 2nd band (Brown), 3rd band (Black), and 4th band (Gold). The text above the resistor indicates a value of $1.0\text{ K}\Omega \pm 5\%$.

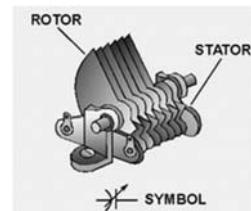
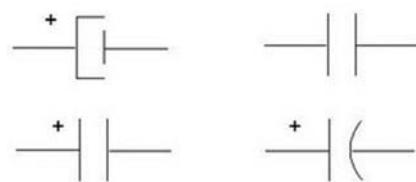
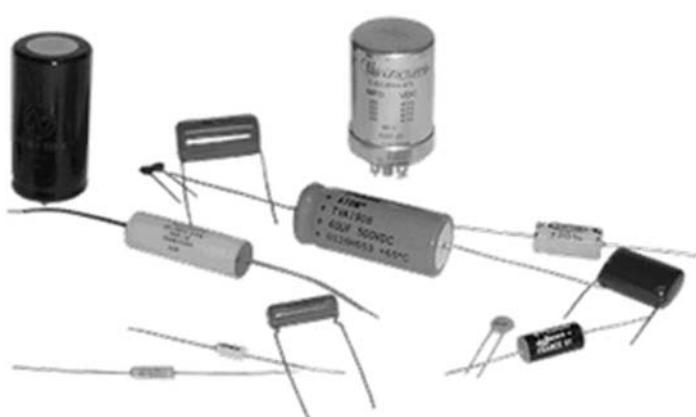
Color	1st Band	2nd Band	3rd Band	Decimal Multiplier	Tolerance
Black	0	0	0	1	1 %
Brown	1	1	1	10	$\pm 1\%$
Red	2	2	2	100	$\pm 2\%$
Orange	3	3	3	1K	1,000
Yellow	4	4	4	10K	10,000
Green	5	5	5	100K	100,000
Blue	6	6	6	1M	1,000,000
Violet	7	7	7	10M	10,000,000
Gray	8	8	8	100,000,000	
White	9	9	9	1,000,000,000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

5- Band Code

20

Basic electronic devices

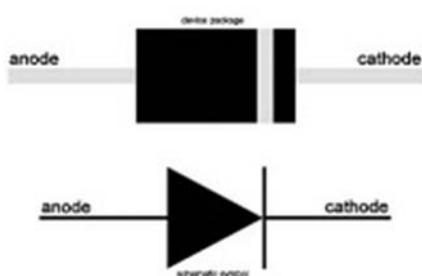
Capacitor



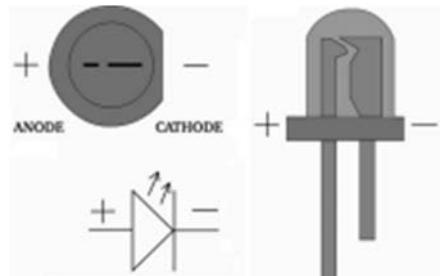
21

Basic electronic devices

Diode



LED



22



23

Tools for Develop

- 1 Arduino IDE and SDK for ESP8266
- 2 NETPIE Microgear library for ESP8266
- 3 CP2012 USB to UART driver
- 4 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

NETPIE Microgear library for ESP8266

Download from GitHub

- <https://github.com/netpieio/microgear-esp8266-arduino>

From Manage Libraries... Menu

- Sketch --> Include Library --> Manage Libraries...
- Search "Microgear" click Choose

26

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 »

27

Workshop#1

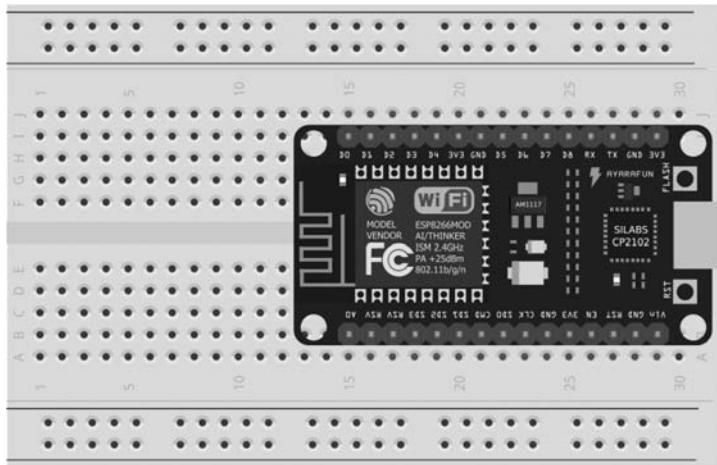
Start learning ESP8266 with Arduino IDE

- BLINK
- TOGGLE SWITCH
- WIFI SCAN
- WEB SERVER

28

Workshop#1

Prepare hardware snap in breadboard



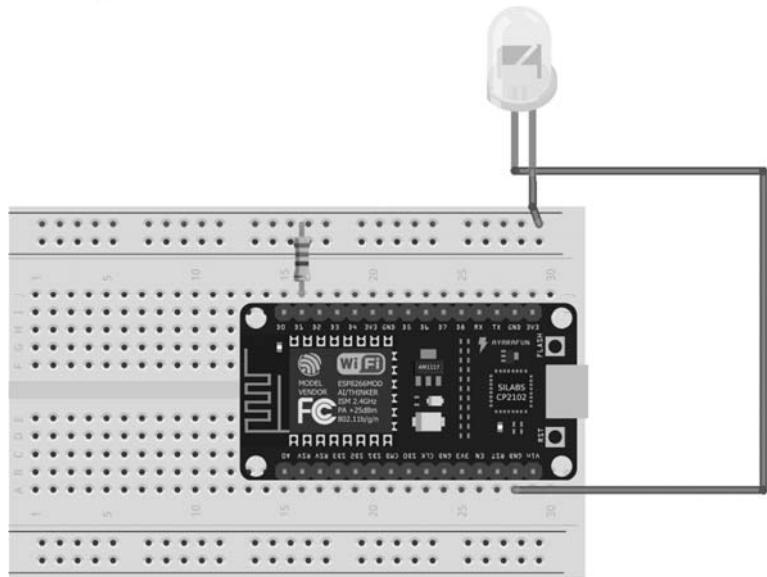
29

Workshop#1 [test board with blink Built in LED]

```
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)  
}
```

30

Workshop#1 [test board with blink GPIO5]



31

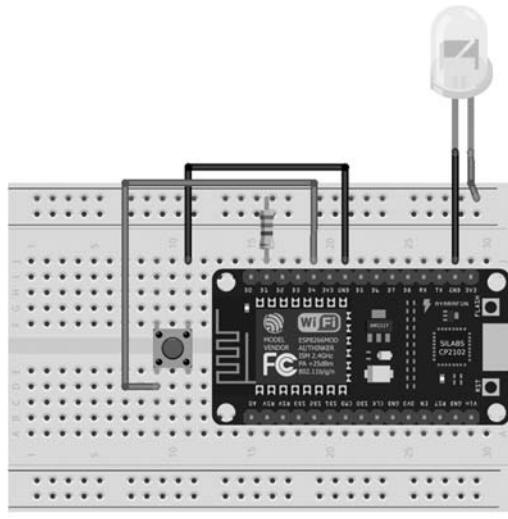
Workshop#1 [test board with blink GPIO5]

```
#define ledPin1 D1 //GPIO5
void setup() {
    pinMode(ledPin1, OUTPUT);    // Initialize the LED_BUILTIN pin as an output
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(ledPin1, HIGH);  // 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(ledPin1, LOW);   // Turn the LED off by making the voltage HIGH
    delay(2000);                // Wait for two seconds (to demonstrate the active low LED)
}
```

32

Workshop#1 [test Toggle Switch]



33

Workshop#1 [test Toggle Switch]

```
#define ledPin D1 // GPIO5
#define sw1 D4 // GPIO2
int st_sw = 0;
int st = 0;
int last_st_sw = 1;
void setup() {
    pinMode(ledPin, OUTPUT); // Set pin mode
    pinMode(sw1, INPUT_PULLUP);
}

void loop() {
    st_sw = digitalRead(sw1); // Read input port1
    if ((st_sw == 0) && (last_st_sw == 1)) // Check current status
    {
        st = ~st; // Toggle
        digitalWrite(ledPin, st); // Drive LED
        delay(250);
    }
    last_st_sw = st_sw; // Update current status
}
```

34

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);

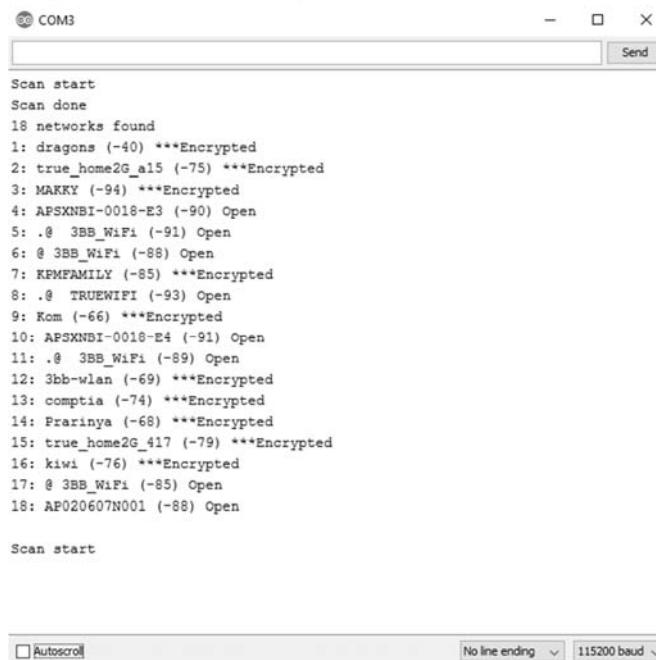
  Serial.println("Setup done");
}

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)? " Open": " ***Encrypted");
          delay(10);
        }
      Serial.println("");
      // Wait a bit before scanning again
      delay(5000);
    }
}
```

35

Workshop#1 [Wifi scan] – serial monitor



```
Scan start
Scan done
18 networks found
1: dragons (-40) ***Encrypted
2: true_home2G_a15 (-75) ***Encrypted
3: MARKY (-94) ***Encrypted
4: AFSXNBI-0018-E3 (-90) Open
5: .@ 3BB_WiFi (-91) Open
6: @ 3BB_WiFi (-88) Open
7: KMPFAMILY (-85) ***Encrypted
8: .@ TRUEWIFI (-93) Open
9: Kom (-66) ***Encrypted
10: AFSXNBI-0018-E4 (-91) Open
11: .@ 3BB_WiFi (-89) Open
12: 3bb-wlan (-69) ***Encrypted
13: comptia (-74) ***Encrypted
14: Frarinya (-68) ***Encrypted
15: true_home2G_417 (-79) ***Encrypted
16: kiwi (-76) ***Encrypted
17: @ 3BB_WiFi (-85) Open
18: AP020607N001 (-88) Open

Scan start
```

Autoscroll No line ending 115200 baud

36

Workshop#1 [WifiWeb server] – sketch GPIO5

```
#include <ESP8266WiFi.h>
#define ledPin D1 //GPIO5

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 GPIO5
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, 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());
```

37

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(ledPin, val);

  client.flush();

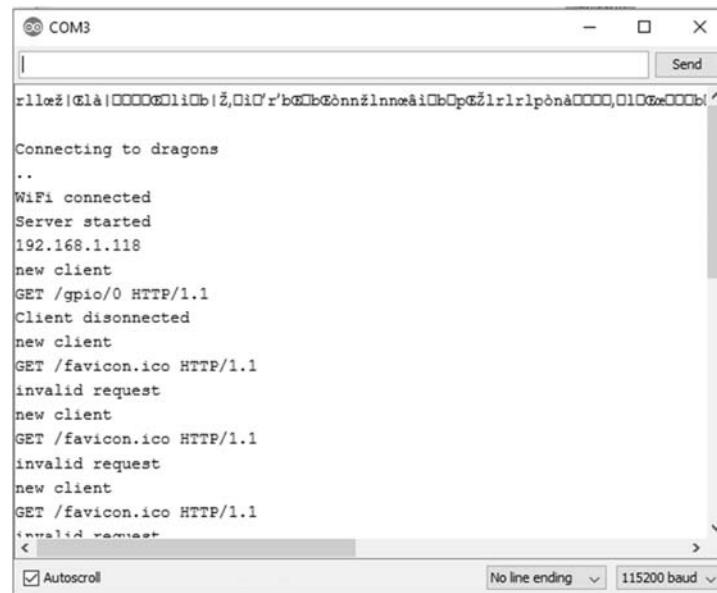
  // Prepare the response
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n<head>\r\n<title>ESP8266 GPIO Control</title>\r\n</head>\r\n<body>\r\n<h1>ESP8266 GPIO Control</h1>\r\n<p>GPIO 5 is now <strong>" + String(val) + "</strong></p>\r\n</body>\r\n</html>\r\n";
  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
}
```

38

Workshop#1 [WifiWeb server] – serial monitor



The screenshot shows a serial monitor window titled "COM3". The text output is as follows:

```
Connecting to dragons
..
WiFi connected
Server started
192.168.1.118
new client
GET /gpio/0 HTTP/1.1
Client disconnected
new client
GET /favicon.ico HTTP/1.1
invalid request
new client
GET /favicon.ico HTTP/1.1
invalid request
new client
GET /favicon.ico HTTP/1.1
invalid request
```

At the bottom of the window, there are checkboxes for "Autoscroll" and "No line ending", and a dropdown for "115200 baud".

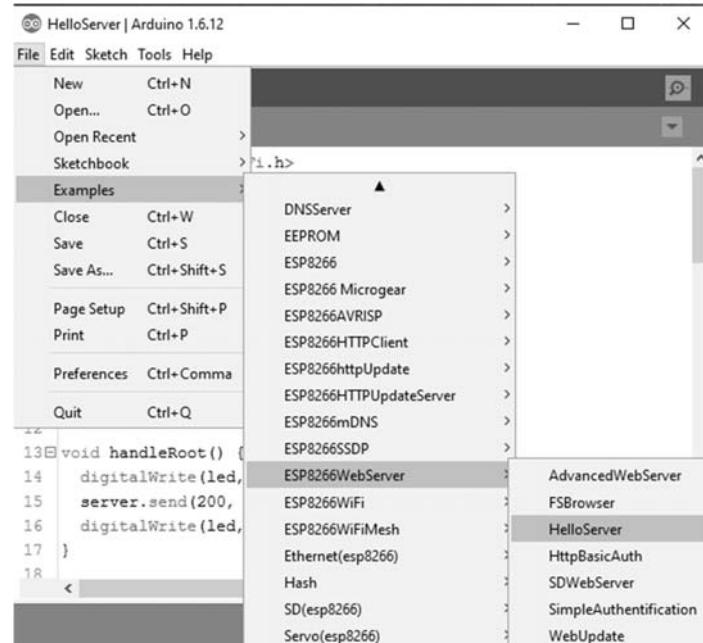
39

Workshop#1 [WifiWeb server] – web browser



40

Workshop#1 [HelloServer]



41

Workshop#1 [HelloServer]

The screenshot shows the Arduino IDE code editor with the sketch named "HelloServer". The code includes #include statements for WiFi.h, WiFiClient.h, ESP8266WebServer.h, and ESP8266mDNS.h. It defines constants for SSID and password, initializes a WebServer on port 80, and sets up pin 13 as an output. The handleRoot() function turns the LED on and off and sends a response. The handleNotFound() function returns a "File Not Found" message. The code editor has a toolbar with icons for file operations like new, open, save, and upload.

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

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

ESP8266WebServer server(80);

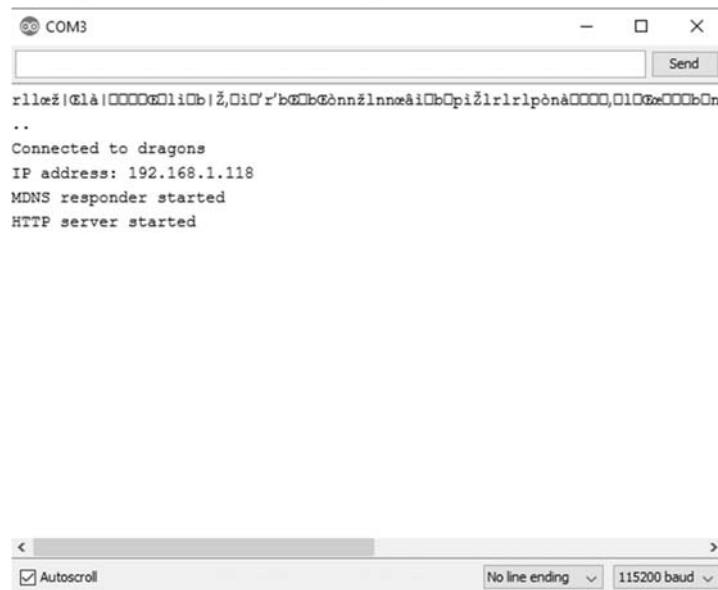
const int led = 13;

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";
}
```

42

Workshop#1 [HelloServer]-serial monitor



43

Workshop#1 [HelloServer]-web browser



44

Workshop#2

up data to cloud (Thinkspeak)

- Apply Thinkspeak
- Test send data
- Customize visual board

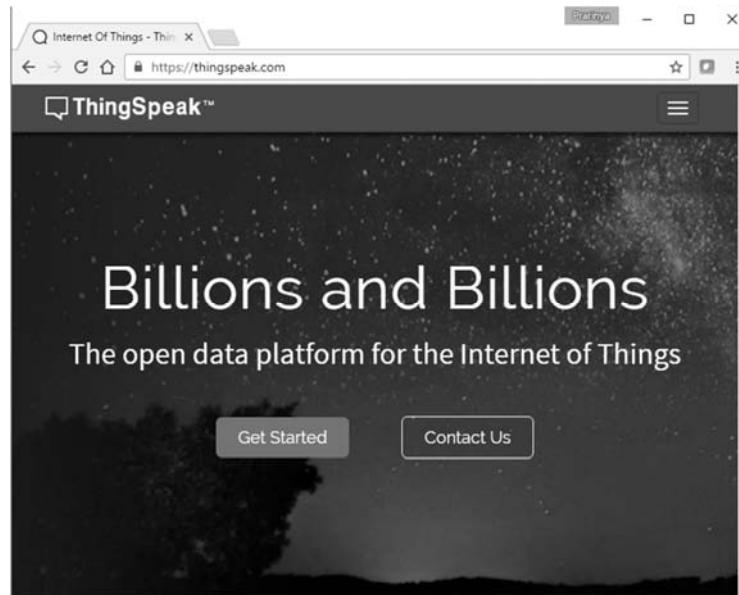
45

Workshop#2 Apply Thinkspeak

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

46

Workshop#2 Apply Thinkspeak



47

Workshop#2 Apply Thinkspeak

A screenshot of the ThingSpeak sign-in page. At the top is a navigation bar with the 'ThingSpeak' logo, 'Channels', 'Apps', 'Blog', and 'Support'. Below the navigation bar are fields for 'User ID' and 'Password', each with a corresponding input box. Underneath the password field is a link 'Forgot your password?'. Below the input boxes is a checkbox labeled 'Remember my User ID'. At the bottom of the form is a 'Sign In' button. To the right of the 'Sign In' button is a link 'Sign In With MathWorks Account' with an arrow pointing towards it. Below these links is a message 'Don't have a ThingSpeak account?' followed by a circled 'Sign Up' link, which also has an arrow pointing towards it.

48

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 

49

Workshop#2 Create New Channel

My Channels

New Channel 

Name	Created
AMS Machine	2015-08-13

[Private](#) [Public](#) [Settings](#) [API Key](#) [Data Import / Export](#)

50

Workshop#2 Create New Channel

ThingSpeak Channels ▾ Apps Blog Support ▾

New Channel

Name	<input type="text"/>	Metadata	<input type="text"/>
Description	<input type="text"/>	Tags	<input type="text"/> (Tags are comma separated)
Field 1	<input type="text"/> Field Label 1 <input checked="" type="checkbox"/>	Latitude	<input type="text"/>
Field 2	<input type="text"/> <input type="checkbox"/>	Longitude	<input type="text"/>
Field 3	<input type="text"/> <input type="checkbox"/>	Elevation	<input type="text"/>
Field 4	<input type="text"/> <input type="checkbox"/>	Make Public?	<input checked="" type="checkbox"/> 
Field 5	<input type="text"/> <input type="checkbox"/>	URL	<input type="text"/>
Field 6	<input type="text"/> <input type="checkbox"/>	Video ID	<input type="text"/> <input type="radio"/> YouTube <input type="radio"/> Vimeo
Field 7	<input type="text"/> <input type="checkbox"/>	Save Channel	
Field 8	<input type="text"/> <input type="checkbox"/>	51	

Workshop#2 Create New Channel

Data Import / Export

Update Channel Feed - GET
`GET https://api.thingspeak.com/update?api_key=G7EEA1W3NU7CAG7H&field1=0`

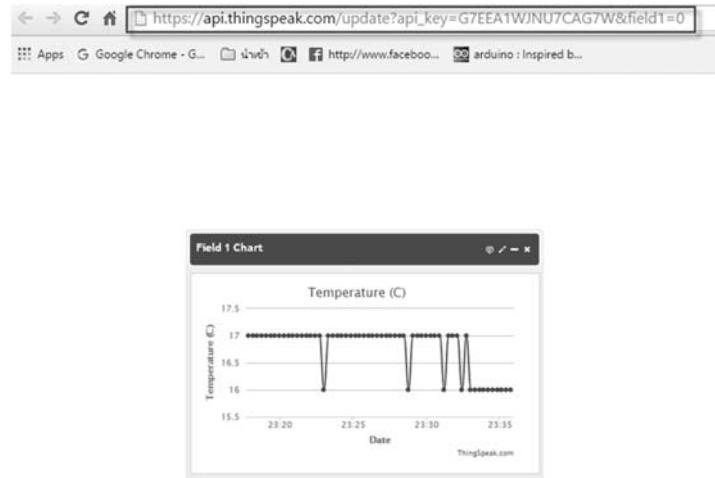
Update Channel Feed - POST
`POST https://api.thingspeak.com/update.json
api_key=G7EEA1W3NU7CAG7H
field1=73`

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`

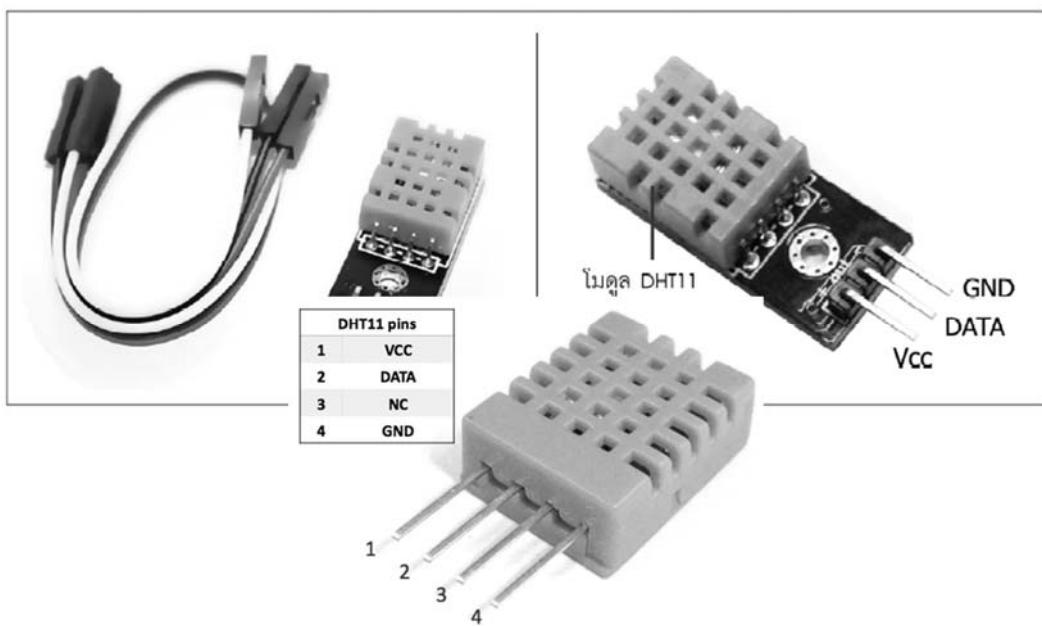
Get Status Updates
`GET https://api.thingspeak.com/channels/51581/status.json`

Workshop#2 Create New Channel



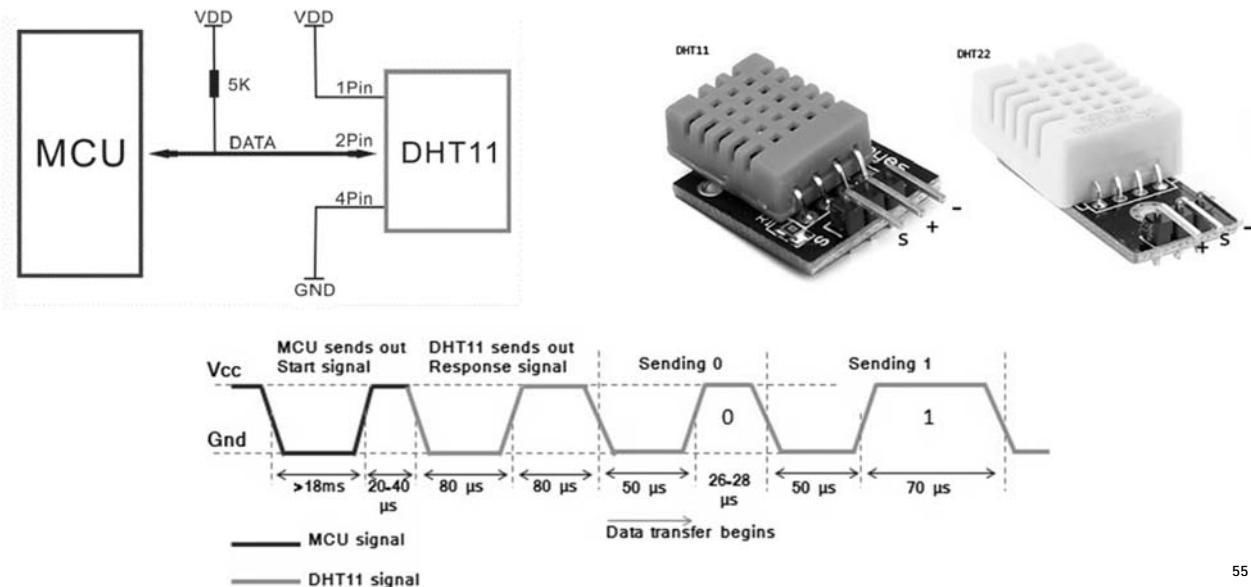
53

Workshop#2 DHT xx Temperature and Humidity Sensor



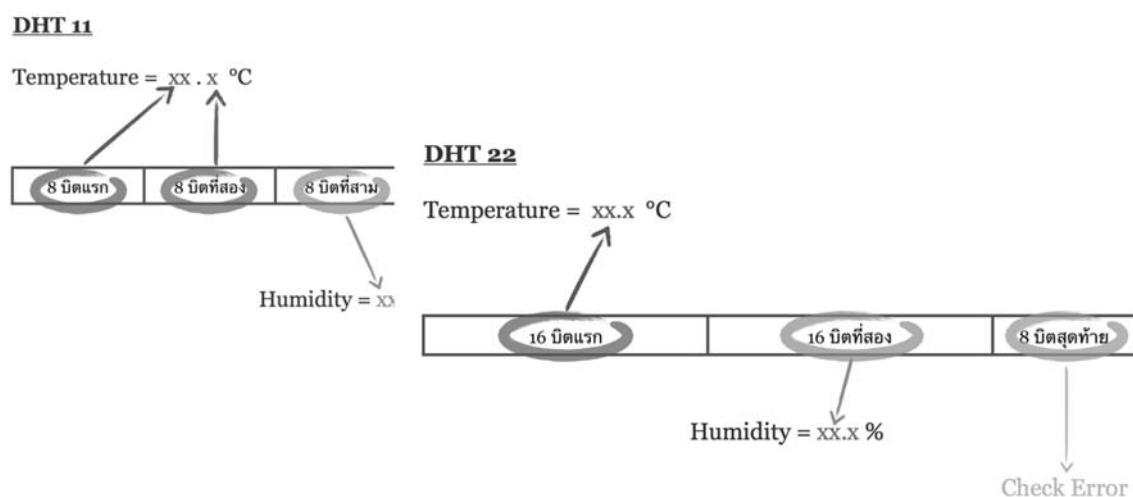
54

Workshop#2 DHT xx Temperature and Humidity Sensor



55

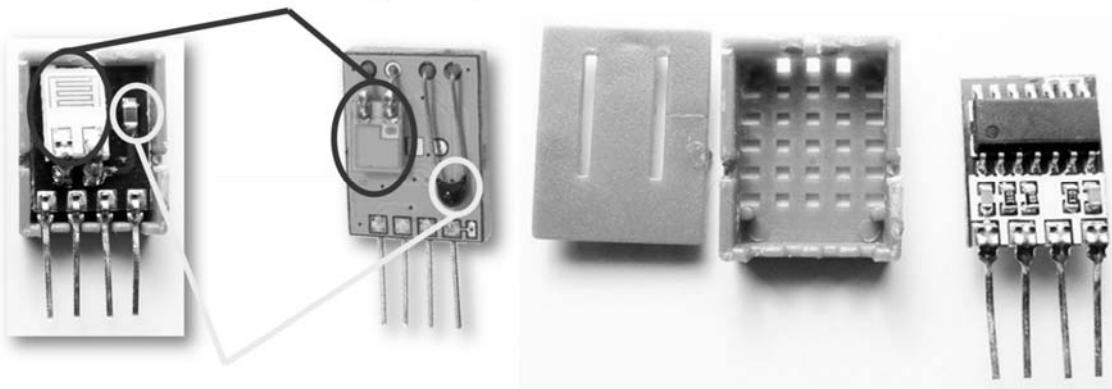
Workshop#2 DHT xx Temperature and Humidity Sensor



56

Workshop#2 DHT xx Temperature and Humidity Sensor

Resistive Humidity Sensing Component



NTC Temperature Sensor Thermistor

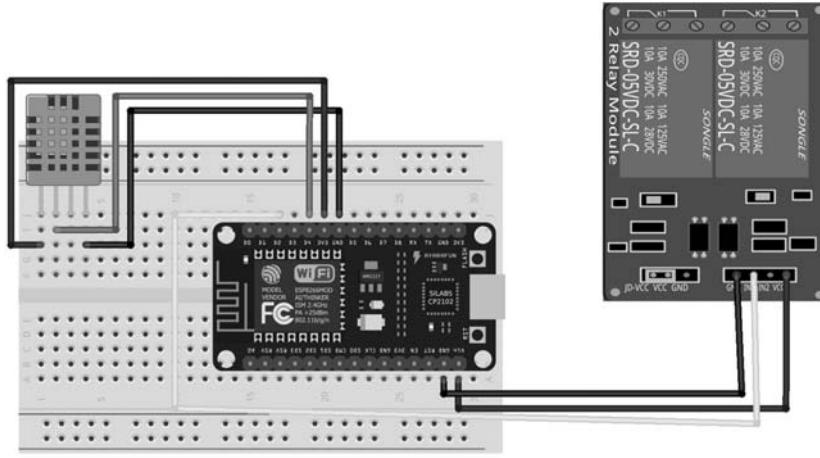
57

Workshop#2 DHT xx Temperature and Humidity Sensor

ความแตกต่าง	DHT11	DHT22
ขอบเขตและความคลื่อนในการวัดความชื้นสัมพัทธ์	ความชื้น : 20 - 80 % ความคลาดเคลื่อน : 5 %	ความชื้น : 0 - 100 % ความคลาดเคลื่อน : 5 %
ขอบเขตและความคลื่นในการวัดอุณหภูมิ	อุณหภูมิ : 0 - 50 °C ความคลานเคลื่อน : ± 2 °C	อุณหภูมิ : -40 - 125 °C ความคลานเคลื่อน : ± 0.5 °C
ความถ่วงไวในการประมวลผล	1 Hz (1 ครั้งต่อวินาที)	0.5 Hz (2 ครั้งต่อวินาที)
ขนาด	กว้าง x ยาว x สูง : 5.5 mm x 12 mm x 15.5 mm	กว้าง x ยาว x สูง : 7.7 mm x 15.1 mm x 25 mm
ราคา	ประมาณ 100 บาท	ประมาณ 200 บาท

58

Workshop#2 connect hardware



59

Workshop#2 sketch for test send data

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

#define PUMP_RLY 14 // output drive relay for pump GPIO14 (D5)
#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 > 50 || 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();
}
```

60

Workshop#2 sketch for test send data

```
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!"));
}
```

61

Workshop#2 sketch for test send data

```
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;
    }
}
```

62

Workshop#2 sketch for test send data

```
// 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;
}
```

63

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.

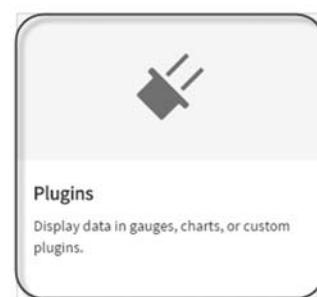
Analytics



MATLAB Analysis
Explore and transform data.



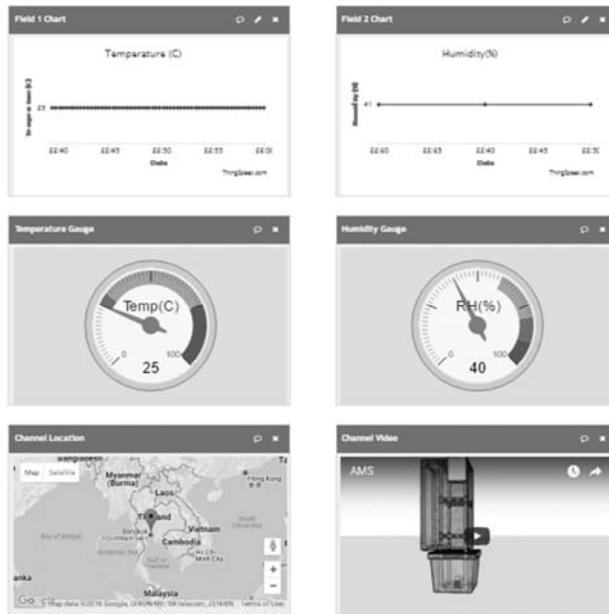
MATLAB Visualizations
Visualize data in MATLAB plots.



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

64

Customize visual board



65

Workshop#3

- Display and control device
- Apply to use NETPIE
- Test send data
- Customize visual board and control

66

Workshop#3 LCD with I²C Module [Key feature]

Display 16x2 Character

I²C Interface with MCU

2 Address Bus of I²C

- Chip PCF8574 -> 0x27
- Chip PCF8574A -> 0x3F

Use 4 Wires Vcc(+),GND(G),SDA,SCL

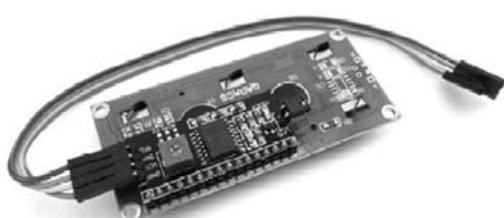
Use +5V for Power

67

Workshop#3 LCD with I²C Module



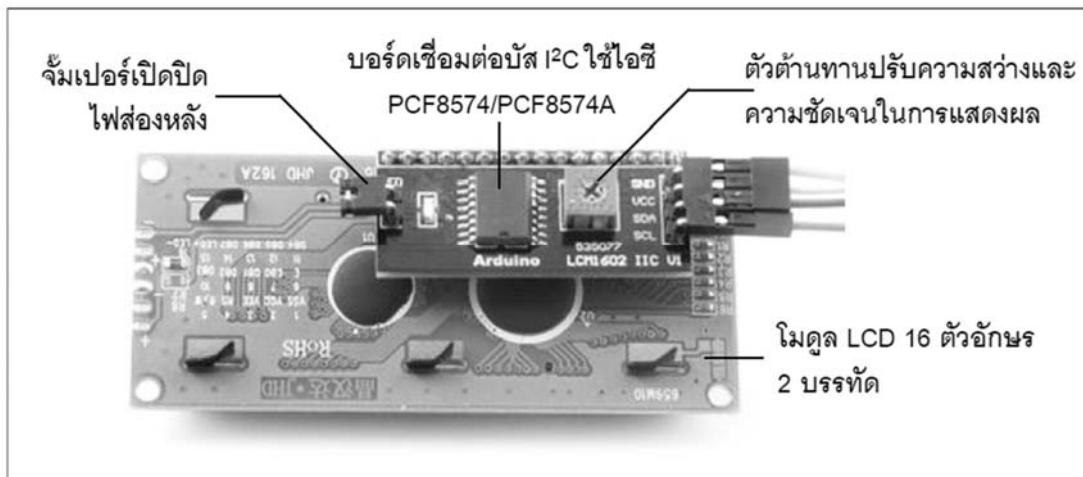
(ก) ภาพด้านหน้าของโมดูล LCD แบบบัส I²C



(ข) ภาพด้านหลังของโมดูล LCD แบบบัส I²C แสดงให้เห็นถึงบอร์ดเชื่อมต่อบัส I²C ที่ใช้อีซีเบอร์ PCF8574 หรือ PCF8574A รวมถึงตัวต้านทานปรับค่าได้สำหรับปรับความชัดเจนในการแสดงผล

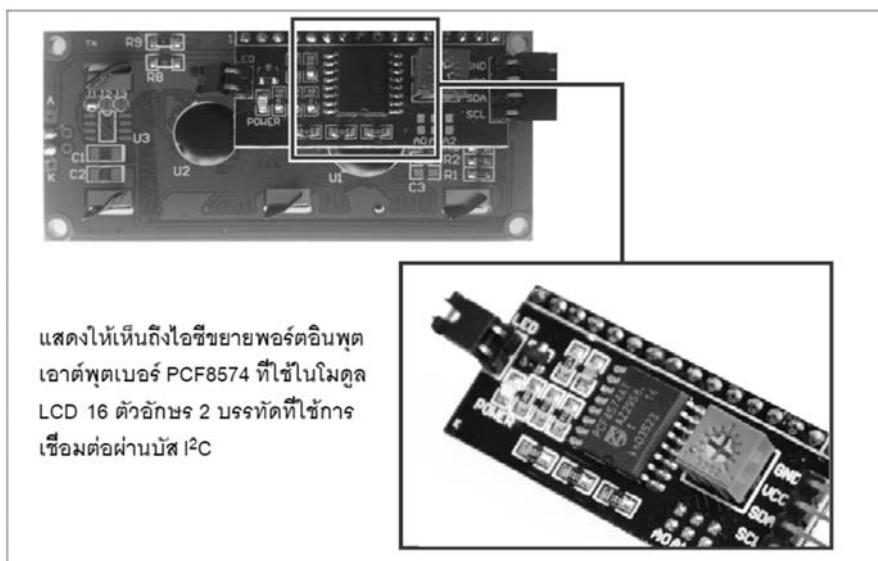
68

Workshop#3 LCD with I²C Module



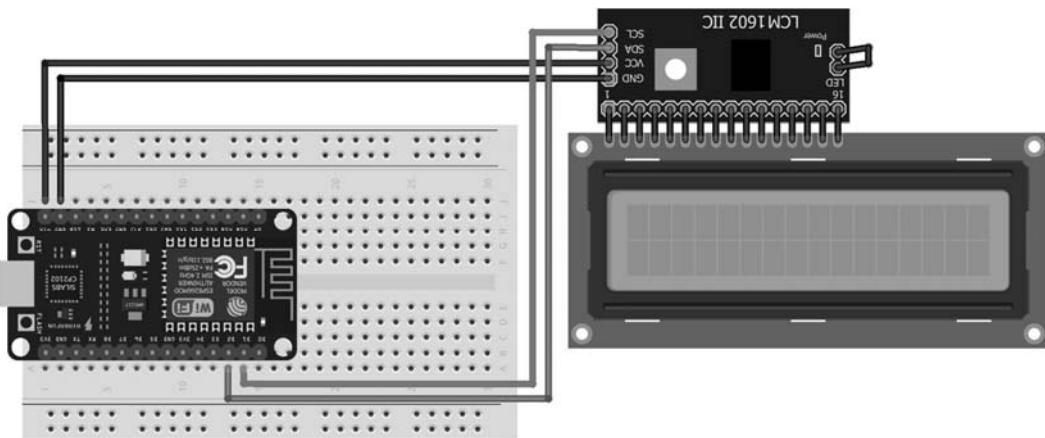
69

Workshop#3 LCD with I²C Module



70

Workshop#3 connect LCD with I²C



71

Workshop#3 sketch for test LCD

```
//SCL as D1/GPIO5
//SDA as D2/GPIO4
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Initial I2C-LCD
// Address is 0x27 (for PCF8574) or 0x3F (for PCF8574A)
// Type 16 characters 2 lines
//LiquidCrystal_I2C lcd(0x3F, 16, 2);
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup()
{
lcd.init(); // Start
lcd.backlight(); // Enable LED backlight
lcd.setCursor(0,0); // Set home cursor
lcd.print("L1-Hello World"); // Display message on line 1 (upper)
lcd.setCursor(0,1); // Set new position
lcd.print("L2-IoT Training"); // Display message on line 2 (lower)
}
void loop(){
// put your main code here, to run repeatedly:
}
```

72

Workshop#3 LCD I²C serial workshop

```
#include <Wire.h> // Comes with Arduino IDE
#include <LiquidCrystal_I2C.h>

//LiquidCrystal_I2C lcd(0x3F, 16,2);
LiquidCrystal_I2C lcd(0x27, 16,2);

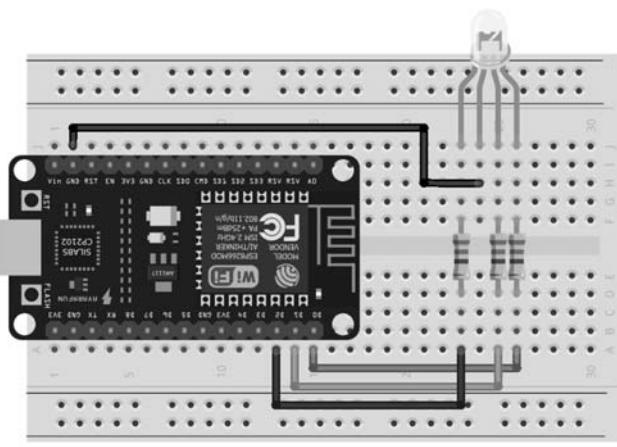
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  lcd.init();
  for(int i = 0; i<3; i++)
  {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print(".....");
  delay(1000);
}

lcd.setCursor(0,1);
lcd.print(".....");
delay(8000);
}

void loop()
{
{
  // when characters arrive over the serial port...
  if(Serial.available()){
    // wait a bit for the entire message to arrive
    delay(300);
    // clear the screen
    lcd.clear();
    // read all the available characters
    while(Serial.available()>0)
    {
      // display each character to the LCD
      lcd.write(Serial.read());
    }
  }
}
}
```

73

Workshop#3 connect LED 3 color (RGB)



74

Workshop#3 NodeMCU LED 3 color

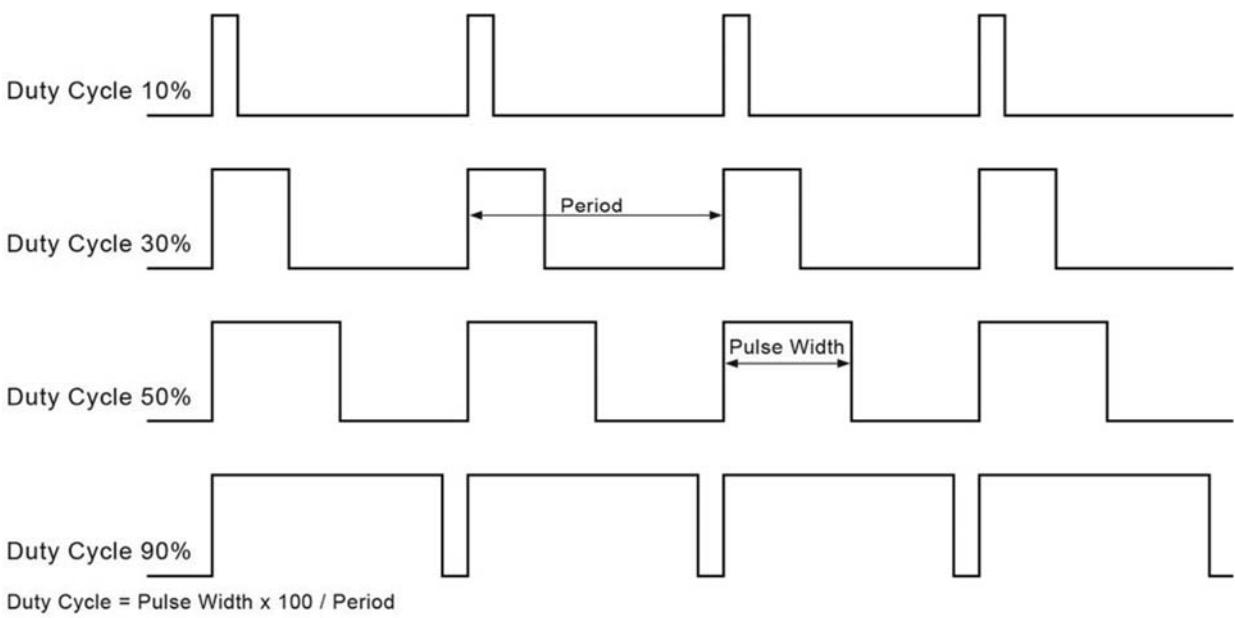
```
#define B_pin D0 //Blue
#define G_pin D1 //Green
#define R_pin D2 //Red
int st_B = 0;
int st_G = 0;
int st_R = 0;
void setup() {
    // put your setup code here, to run once:
    pinMode(B_pin, OUTPUT); // Set pin as output
    pinMode(G_pin, OUTPUT);
    pinMode(R_pin, OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    displayRGB(0, 0, 0); // Drive RGB LED with 3-bit data
    delay(2000); // Delay 2 seconds
    displayRGB(1, 0, 0);
    delay(2000);
    displayRGB(1, 1, 0);
    delay(2000);
    displayRGB(0, 1, 0);
    delay(2000);
    displayRGB(0, 0, 1);
    delay(2000);
    displayRGB(0, 1, 1);
    delay(2000);
    displayRGB(1, 0, 1);
    delay(2000);
    displayRGB(1, 1, 1);
    delay(2000);
}

void displayRGB(int R, int G, int B)
// Send 3-bit digital data to 3 output pins function
{
    digitalWrite(B_pin, B);
    digitalWrite(G_pin, G);
    digitalWrite(R_pin, R);
}
```

75

Workshop#3 Pulse Width Modulation (PWM)



Workshop#3 NodeMCU LED 3 color PWM

```
#define B_pin D0 //Blue
#define G_pin D1 //Green
#define R_pin D2 //Red
int st_R = 255;
int st_G = 0;
int st_B = 0;
void setup() {
    // put your setup code here, to run once:
    pinMode(B_pin, OUTPUT); // Set pin as output
    pinMode(G_pin, OUTPUT);
    pinMode(R_pin, OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    for (int i = 0; i < 256; i++) // Set loop
    {
        st_B = 0; // Clear BLUE data
        st_G++; // Increase GREEN data
        displayRGB(st_R, st_G, st_B); // Drive LED with PWM
        delay(30); // Short delay
    }

    for (int i = 0; i < 256; i++)
    {
        st_R--; // Decrease RED data
        displayRGB(st_R, st_G, st_B); // Drive LED with PWM
        delay(30); // Short delay
    }

    for (int i = 0; i < 256; i++)
    {
        st_G--; // Decrease GREEN data
        displayRGB(st_R, st_G, st_B); // Drive LED with PWM
        delay(30); // Short delay
    }
}
```

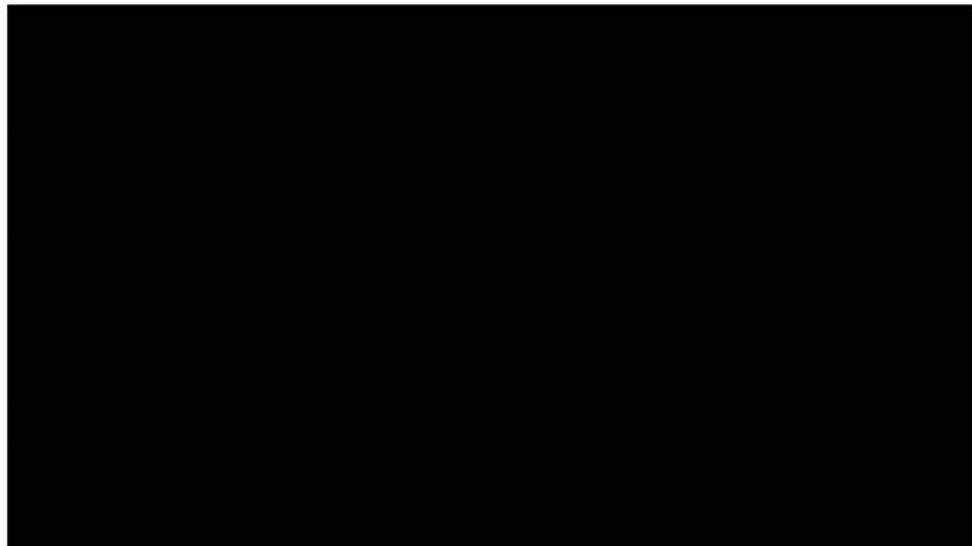
77

Workshop#3 NodeMCU LED 3 color PWM

```
for (int i = 0; i < 256; i++)
{
    st_G = 0; // Clear GREEN data
    st_R++; // Increase RED data
    displayRGB(st_R, st_G, st_B); // Drive LED with PWM
    delay(30); // Short delay
}
for (int i = 0; i < 256; i++)
{
    st_B--; // Decrease BLUE data
    displayRGB(st_R, st_G, st_B); // Drive LED with PWM
    delay(30); // Short delay
}
void displayRGB(int R, int G, int B)
// Drive LED with PWM by using analogWrite function
{
    analogWrite(B_pin, B);
    analogWrite(G_pin, G);
    analogWrite(R_pin, R);
}
```

78

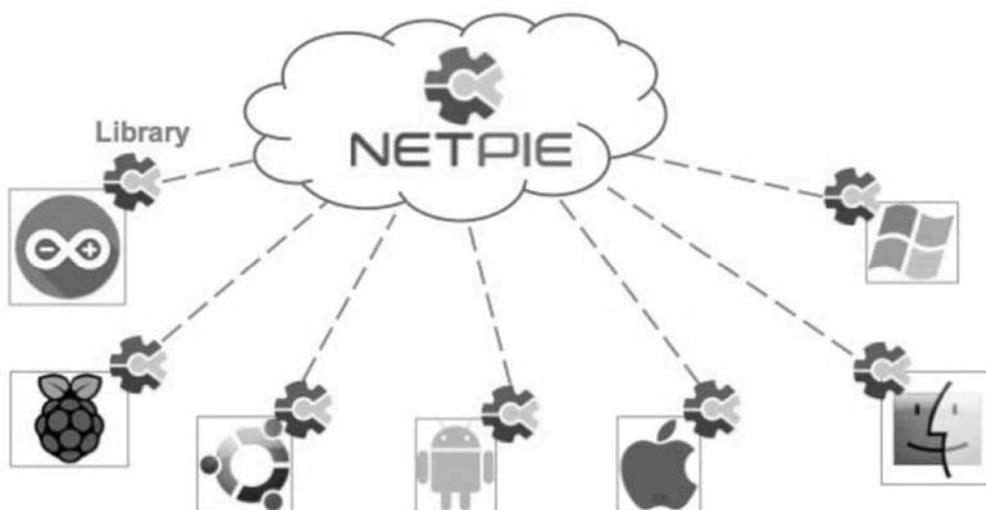
What is “NETPIE”



79

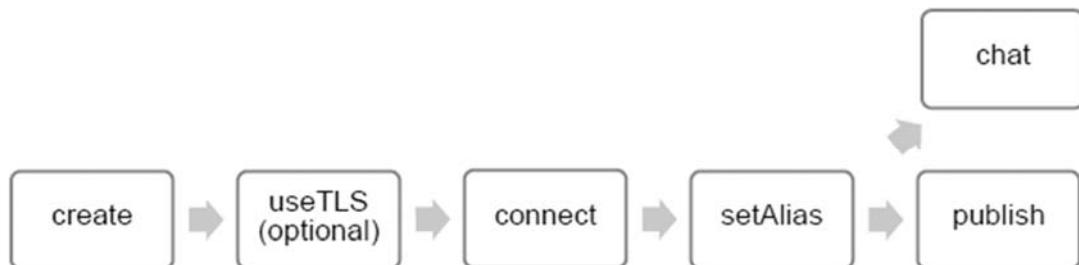
About NETPIE

NETPIE = Cloud Platform + Microgear Libraries



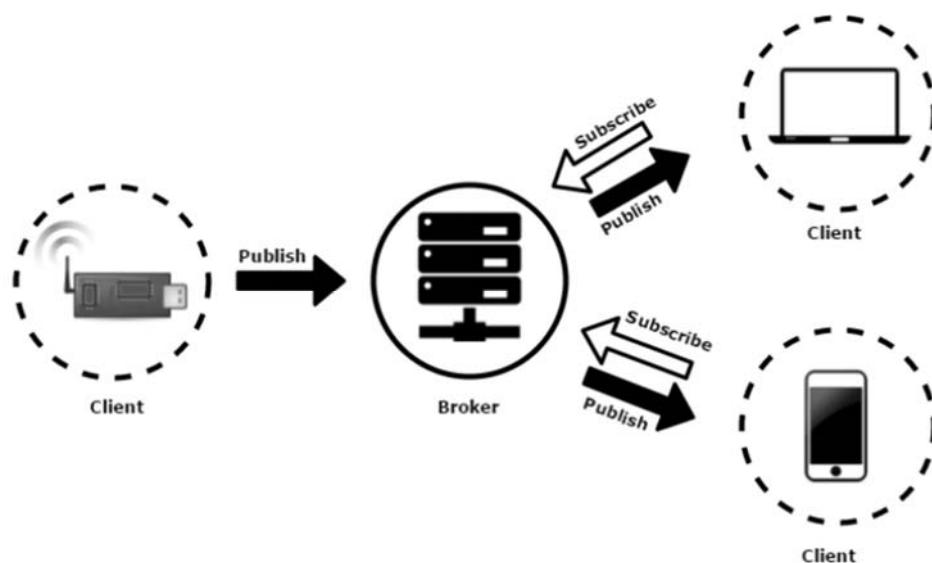
80

Basic function of Microgear



81

MQTT (MQ TELEMETRY TRANSPORT)



82

MQTT Topics

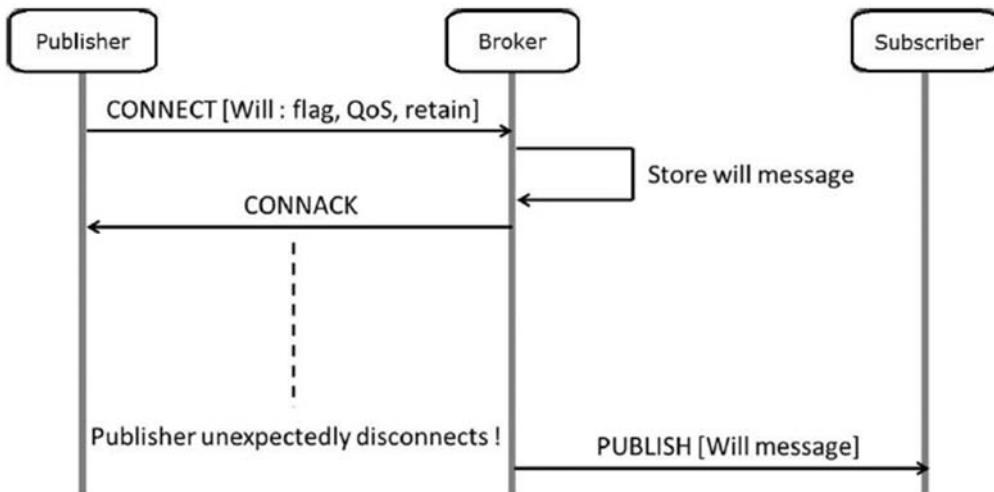
- UTF-8 String
- File Path -> *myhome/floor-one/room-c/temperature*
- Single-Level Wildcard (+)
 - *myhome/floor-one/+/temperature*
- Multi-Level Wildcard (#)
 - *myhome/floor-one/#*

83

MQTT Connections

Control Packets	Sender		Description
	Broker	Client	
CONNECT		X	Request connect
CONNACK	X		Acknowledge connect
PUBLISH	X	X	Message for Publish
PUBACK	X	X	Published (QoS Level 1)
PUBREC	X	X	Published (QoS Level 2)
PUBREL	X	X	Published and Remove message(QoS Level 2)
PUBCOM	X	X	Publish complete and delete status (QoS Level 2)
SUBSCRIBE		X	Request Subscribe
SUBACK	X		Acknowledge Subscribe
UNSUBSCRIBE		X	Cancel Subscribe
UNSUBACK	X		Acknowledge Cancel Subscribe
PINGREQ		X	PING Request
PINGRESP	X		PING Response
DISCONNECT		X	Cancel connect

Last Will Message



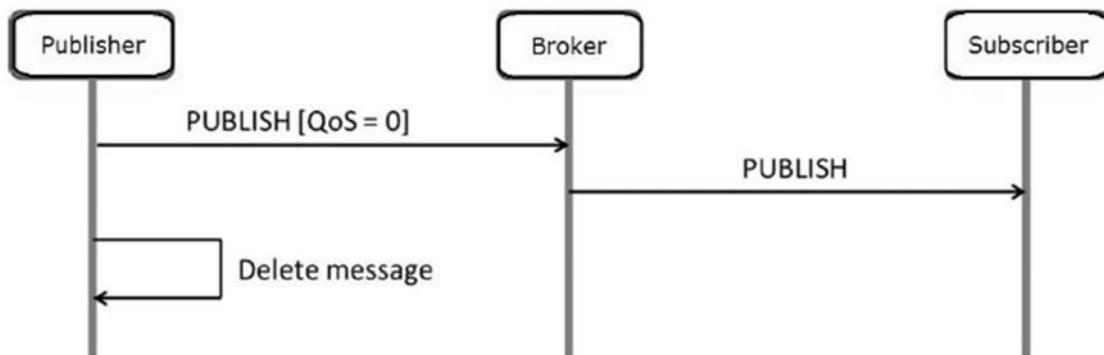
85

MQTT Quality of Service (QoS)



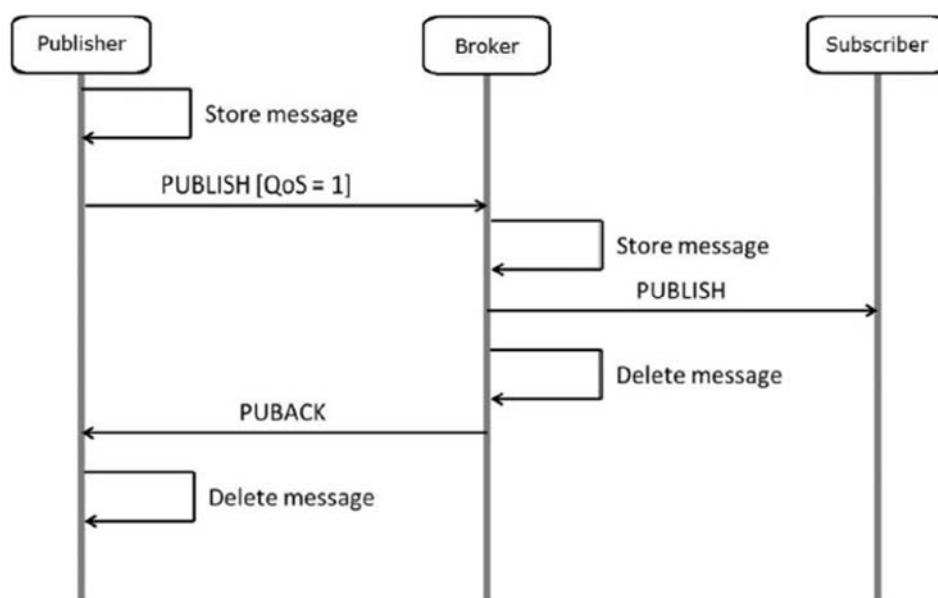
86

At Most Once QoS 0



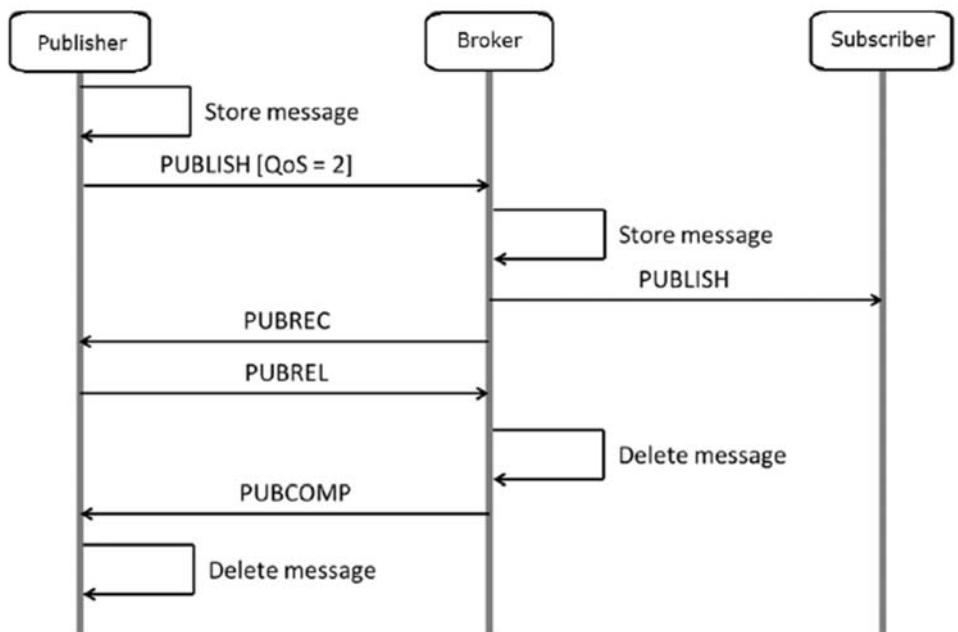
87

At Least Once QoS 1



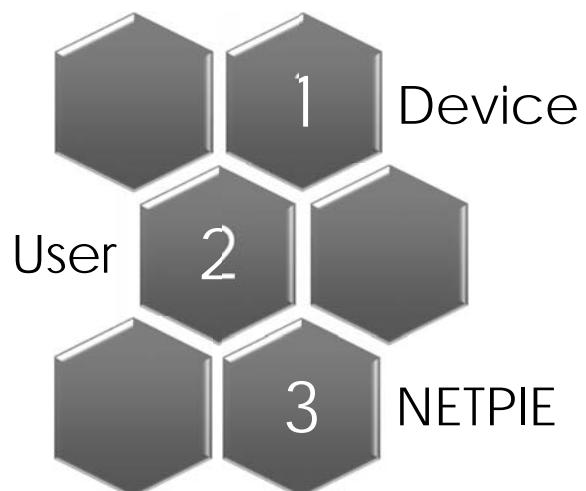
88

Exactly Once QoS 2



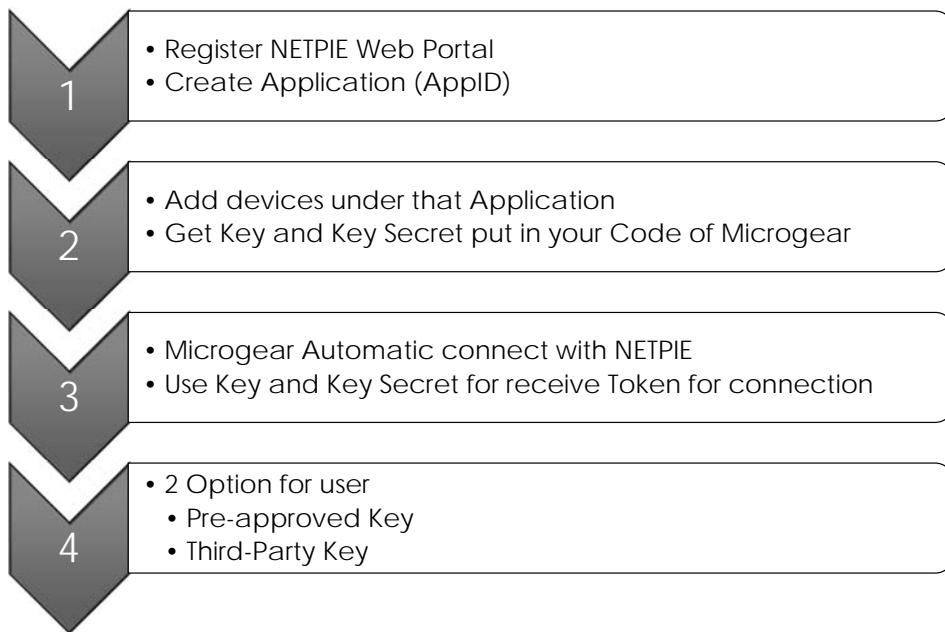
89

Authorization and Authentication of NETPIE



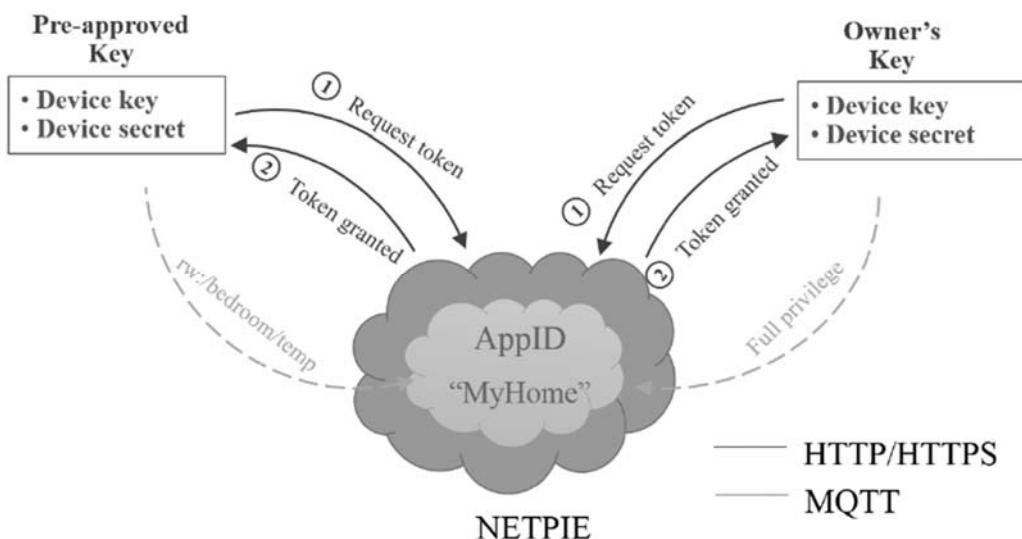
90

NETPIE Connection process



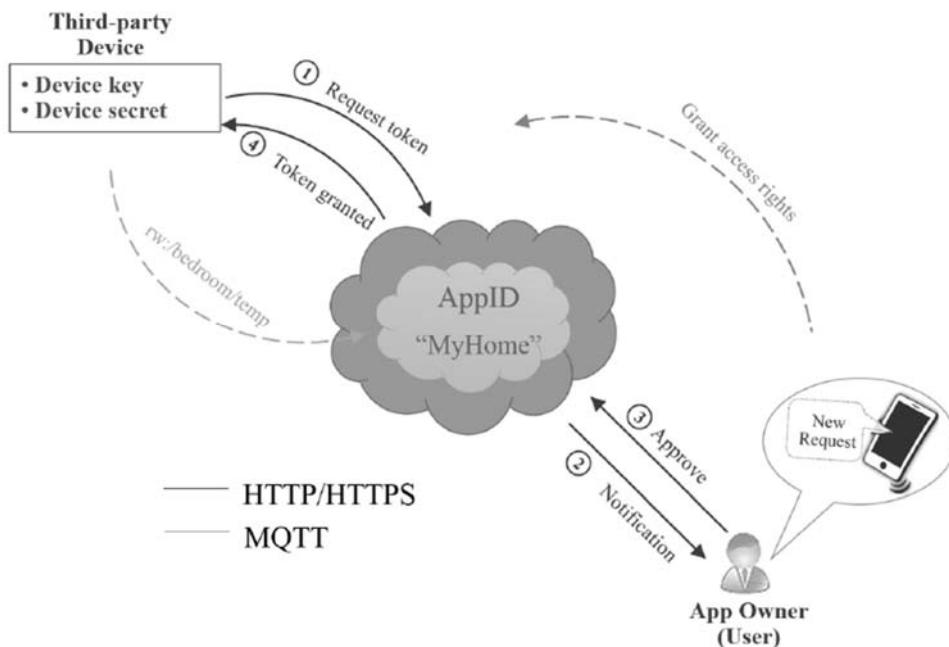
91

Pre-approved Key



92

Third-Party Key



93

Type of Key



94

Scope for access NETPIE MQTT Broker

- 1 • rw:/topic
• rw:/home/bedroom/temperature
- 2 • r:/topic
• r:/home/#
- 3 • w:/topic
• w:/home/+/temperature

95

Apply Member NETPIE (https://netpie.io/sign_up)



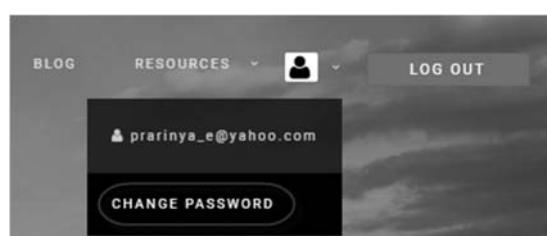
96

Waiting SMS for OTP First Login



97

Change Password

A screenshot of the "CHANGE PASSWORD" form. The form is set against a background of a city skyline at night. It contains three input fields: "CURRENT PASSWORD", "NEW PASSWORD", and "CONFIRM PASSWORD". Below the fields are two buttons: "CANCEL" and "SAVE".

98

Create Application

The screenshot shows the NETPIE Applications dashboard. At the top, there are navigation links: HOME, DEVELOPERS, BLOG, RESOURCES, and LOG OUT. A prominent button labeled "APPLICATIONS" is visible. Below the navigation, a message indicates "Credits available : 97 / 100". The main area displays two large cards: one for "APPLICATIONS" containing the number "2" and another for "THINGS" containing the number "3". Under the APPLICATIONS card, there is a list titled "APPLICATION" with two items: "KobSmartHome" and "TrainIoT".

Create Application

The screenshot shows the process of creating a new application. A modal dialog box is open, prompting for a unique application ID, with the placeholder text "Your unique application ID, e.g., 'PaulSmartHome'". Below the input field are "CREATE" and "CANCEL" buttons. This dialog is overlaid on the main NETPIE Applications dashboard. The dashboard shows the "KobSmartHome" application has been created and is listed under the APPLICATION section. The application card for "KobSmartHome" displays its name and a "DELETE" button. To the right, a card for "APPLICATION KEY" shows the value "0" and a note about Device Key and Session Key. A "CREATE" button is also present here.

Create Application Key

Name your application key, e.g., "my kitchen lamp"

-- select type --

CREATE **CANCEL**

APPLICATION



KobSmartFarm

DELETE

1

APPLICATION KEY

Device Key, Session Key

APPLICATION KEY

+ **X**

KobNodeMCU12E [online:0]

101

Create Application Key

KobNodeMCU12E

Key : SIpLYGJzIBD7ZRI

Secret :  

REST API auth :  

RENAME **CANCEL**

Workshop#3 NodeMCU NETPIE blink

```
/* NETPIE ESP8266 blink */  
  
#include <ESP8266WiFi.h>  
#include <MicroGear.h>  
  
const char* ssid    = "SSID"; //your ssid  
const char* password = "PASSWORD"; //your password  
  
#define APPID  "YOUR_APPID" //your AppID  
#define KEY    "YOUR_KEY" //your Key  
#define SECRET "YOUR_SECRET" //your Secret  
#define ALIAS   "netpieblink"  
  
WiFiClient client;  
  
int timer = 0;  
char state = 0;  
MicroGear microgear(client);  
  
/* If a new message arrives, do this */  
void onMsgHandler(char *topic, uint8_t* msg, unsigned int msglen) {  
    Serial.print("Incoming message --> ");  
    msg[msglen] = '\0';  
    Serial.println((char *)msg);  
    // If message = 1 On LED, If 0 LED off  
    if(*(char *)msg == '1'){  
        digitalWrite(LED_BUILTIN, LOW); // LED on  
    }else{  
        digitalWrite(LED_BUILTIN, HIGH); // LED off  
    }  
}  
  
/* When a microgear is connected, do this */  
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen) {  
    Serial.println("Connected to NETPIE...");  
    /* Set the alias of this microgear ALIAS */  
    microgear.setAlias(ALIAS);  
}
```

103

Workshop#3 NodeMCU NETPIE blink

```
void setup() {  
    /* Add Event listeners */  
    /* Call onMsgHandler() when new message arrives */  
    microgear.on(MESSAGE,onMsgHandler);  
  
    /* Call onConnected() when NETPIE connection is  
    established */  
    microgear.on(CONNECTED,onConnected);  
  
    Serial.begin(115200);  
    Serial.println("Starting...");  
  
    pinMode(LED_BUILTIN, OUTPUT); //Set LED Built-in to  
    Output  
  
    /* Initial WIFI, this is just a basic method to configure WIFI  
    on ESP8266. */  
    /* You may want to use other method that is more  
    complicated, but provide better user experience */  
    if (WiFi.begin(ssid, password)) {  
        while (WiFi.status() != WL_CONNECTED) {  
            delay(500);  
            Serial.print(".");  
        }  
    }  
}
```

```
Serial.println("WiFi connected");  
Serial.println("IP address: ");  
Serial.println(WiFi.localIP());  
  
/* Initial with KEY, SECRET and also set the ALIAS here */  
microgear.init(KEY,SECRET,ALIAS);  
  
/* connect to NETPIE to a specific APPID */  
microgear.connect(APPID);  
}  
  
void loop() {  
    /* To check if the microgear is still connected */  
    if (microgear.connected()) {  
        Serial.println("connected");  
  
        /* Call this method regularly otherwise the connection  
        may be lost */  
        microgear.loop();  
    }  
}
```

104

Workshop#3 NodeMCU NETPIE blink

```
if (timer >= 1000) {  
    Serial.println("Publish...");  
  
    //chat myself with invert stage  
    if(state==0){  
        microgear.chat(ALIAS,state);  
        state = 1;  
    }else{  
        microgear.chat(ALIAS,state);  
        state = 0;  
    }  
    timer = 0;  
}  
else timer += 100;  
}  
else {  
    Serial.println("connection lost, reconnect...");  
    if (timer >= 5000) {  
        microgear.connect(APPID);  
        timer = 0;  
    }  
    else timer += 100;  
}  
delay(100);  
}
```

105

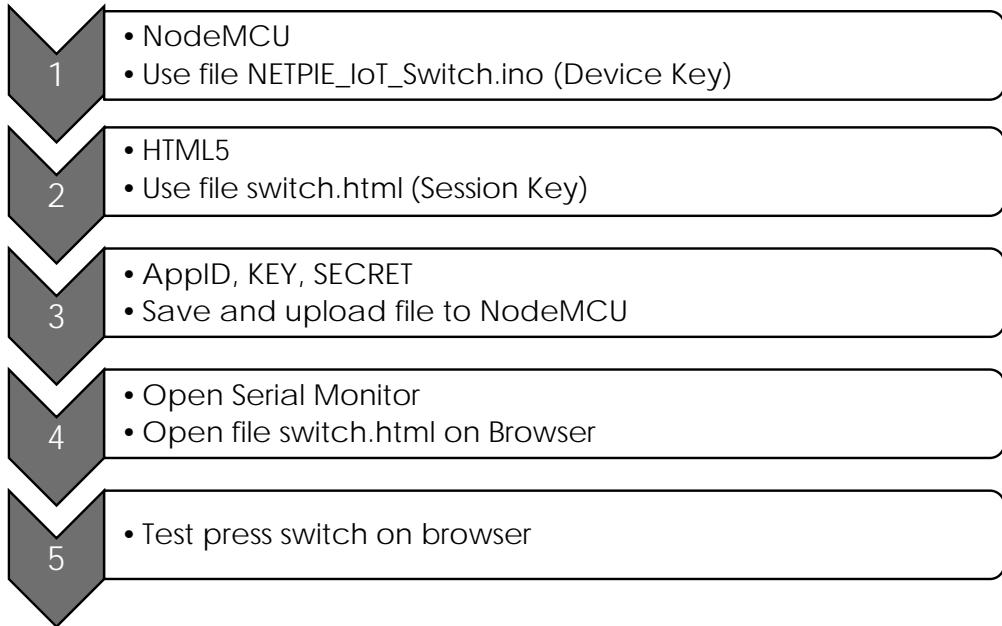
Workshop#4

Another way to control IoT devices

- Control IoT via NETPIE
- Control IoT via Blynk

106

Workshop#4 IoT Switch use NETPIE



107

Workshop#4 NETPIE_IoT_Switch.ino

```
/* NETPIE ESP8266 blink */  
#include <ESP8266WiFi.h>  
#include <MicroGear.h>  
  
const char* ssid = "....."; //your ssid  
const char* password = "....."; //your password  
  
#define APPID "....." //your AppID  
#define KEY "....." //your Device Key  
#define SECRET "....." //your Device Secret Key  
#define ALIAS "....." //your Alias  
  
WiFiClient client;  
  
int timer = 0;  
MicroGear microgear(client);  
  
/* If a new message arrives, do this */  
void onMsghandler(char *topic, uint8_t* msg, unsigned int msglen){  
    Serial.print("Incoming message --> ");  
    msg[msglen] = '\0';  
    Serial.println((char*)msg);  
}  
  
// If messaage = 1 On LED, If 0 LED off  
if(*(char *)msg == '1'){  
    digitalWrite(LED_BUILTIN, LOW); // LED on  
    microgear.chat("switch","1");  
}else{  
    digitalWrite(LED_BUILTIN, HIGH); // LED off  
    microgear.chat("switch","0");  
}  
  
/* When a microgear is connected, do this */  
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)  
{  
    Serial.println("Connected to NETPIE...");  
    /* Set the alias of this microgear ALIAS */  
    microgear.setAlias(ALIAS);  
}  
  
void setup()  
{  
    /* Add Event listeners */  
  
    /* Call onMsghandler() when new message arraives108 */  
    microgear.on(MESSAGE,onMsghandler);  
}
```

Workshop#4 NETPIE_IoT_Switch.ino

```
/* Call onConnected() when NETPIE connection is
established */
microgear.on(CONNECTED,onConnected);

Serial.begin(115200);
Serial.println("Starting...");

pinMode(LED_BUILTIN, OUTPUT); //Set LED Built-in to
Output

/* Initial WIFI, this is just a basic method to configure WIFI
on ESP8266. */
/* You may want to use other method that is more
complicated, but provide better user experience */
if (WiFi.begin(ssid, password)) {
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
}

Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
```

```
/* Initial with KEY, SECRET and also set the ALIAS here */
microgear.init(KEY,SECRET,ALIAS);
/* connect to NETPIE to a specific APPID */
microgear.connect(APPID);
}

void loop() {
    /* To check if the microgear is still connected */
    if (microgear.connected()) {
        Serial.println("connected");

        /* Call this method regularly otherwise the connection
may be lost */
        microgear.loop();
        timer = 0;
    } else {
        Serial.println("connection lost, reconnect...");
        if (timer >= 5000) {
            microgear.connect(APPID);
            timer = 0;
        }
        else timer += 100;
    }
    delay(100);
}
```

109

Workshop#4 switch.html

```
<script src="https://cdn.netpie.io/microgear.js"></script>
<script>
    const APPID = "....."; //your AppID
    const KEY = "....."; //your Session Key
    const SECRET = "....."; //your Session Secret Key
    const ALIAS = "....."; //your Alias
var microgear = Microgear.create({
    key: KEY,
    secret: SECRET,
    alias : ALIAS
});
function toggle() {
    if(document.getElementById("button").innerText=="off"
"){
        microgear.chat('KobNodeMCU12E','1');
    } else{ // Alias of your device
        microgear.chat('KobNodeMCU12E','0');
    }
}
microgear.on('message',function(topic,msg) {
    document.getElementById("data").innerHTML = msg;
    if(msg=="1"){

```

```
        document.getElementById("button").innerText="on";
    }else if(msg=="0"){
        document.getElementById("button").innerText="off";
    });
}
microgear.on('connected', function() {
    microgear.setAlias(ALIAS);
    document.getElementById("data").innerHTML =
"Now I am connected with netpie...";
});
microgear.connect(APPID);
</script>
<div id="data">____</div>
<center>
<button onclick="toggle()" id="button">off</button>
</center>
```

110

Workshop#4 IoT Switch

The screenshot displays three windows related to an IoT switch project:

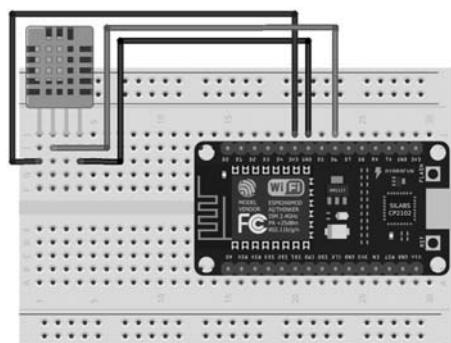
- Serial Monitor (COM3):** Shows the following text:

```
..WiFi connected
IP address:
192.168.1.106
Connected to NETPIE...
connected
connected
connected
connected
connected
connected
connected
connected
Incoming message --> 1
connected
connected
connected
connected
Incoming message --> 0
connected
connected
connected
connected
Incoming message --> 1
connected
connected
connected
connected
Incoming message --> 0
connected
connected
```
- Browser Window 1 (switch.html):** Shows the value "1" and an "on" button.
- Browser Window 2 (switch.html):** Shows the value "0" and an "off" button.

111

Workshop#4 Send data to NETPIE Freeboard

Hardware

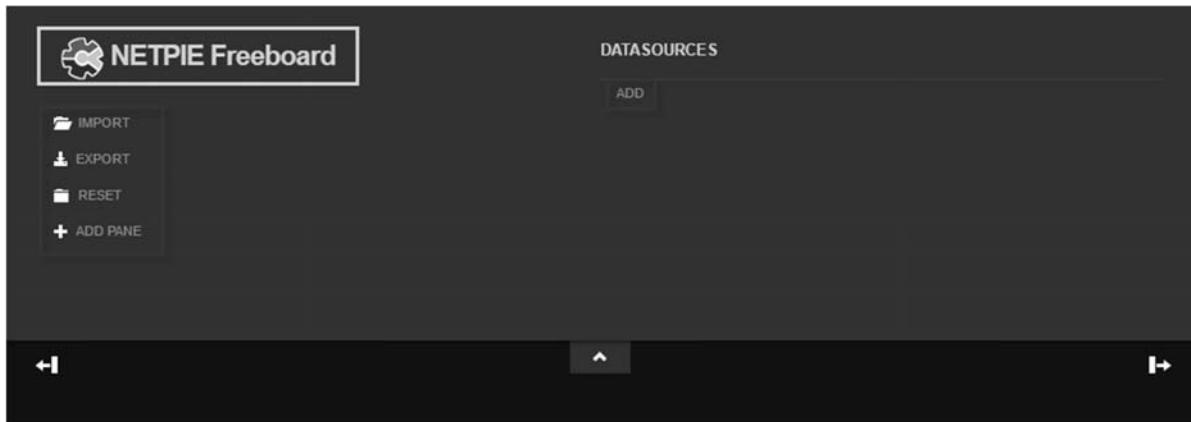


Software

Upload program NETPIE_Freeboard.ino to board

112

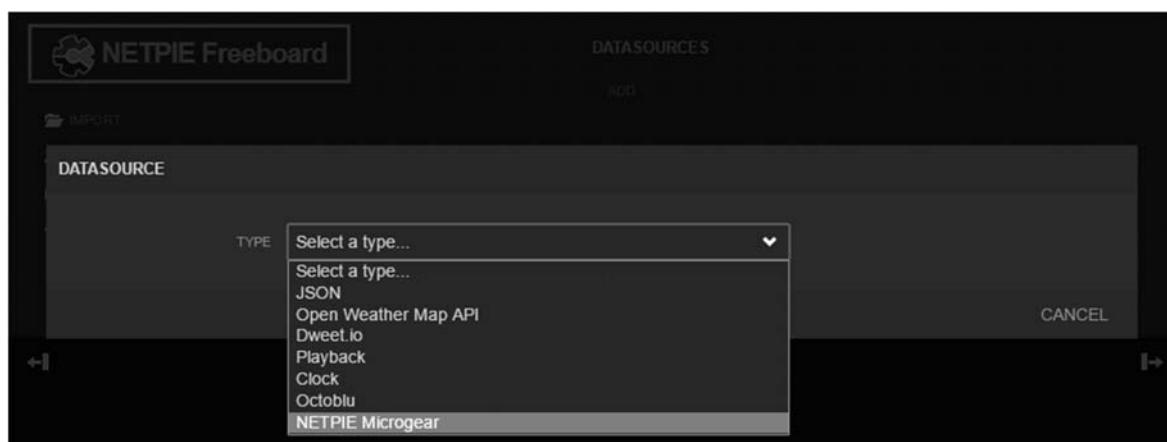
Workshop#4 NETPIE Freeboard



Before to next step Upload file NETPIE_Freeboard.ino to board

113

Workshop#4 NETPIE Freeboard



114

Workshop#4 NETPIE Freeboard

DATASOURCE

Connect to NETPIE as a microgear to communicate real-time with other microgears in the same App ID. The microgear of this datasource is referenced by microgear[DATASOURCENAME]

TYPE: NETPIE Microgear

NAME: dashboard_mon

APP ID: TrainIoT
NETPIE App ID obtained from <https://netpie.io/app>

KEY: Ct0kTCsmAEQphIk

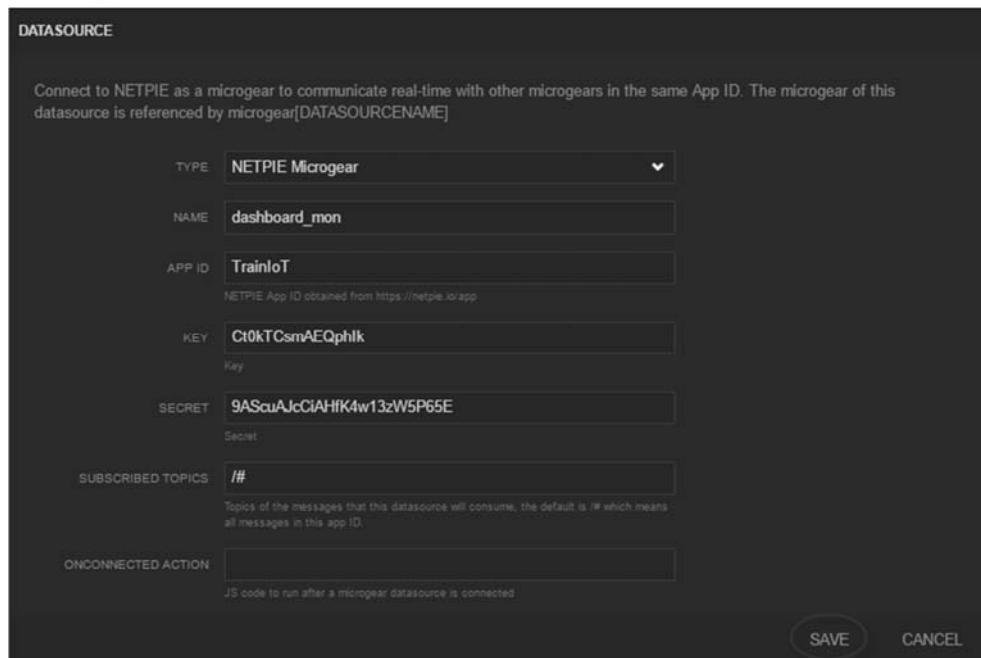
SECRET: 9AScuAJcCiAHfK4w13zW5P65E

SUBSCRIBED TOPICS: /#
Topics of the messages that this datasource will consume, the default is /# which means all messages in this app ID.

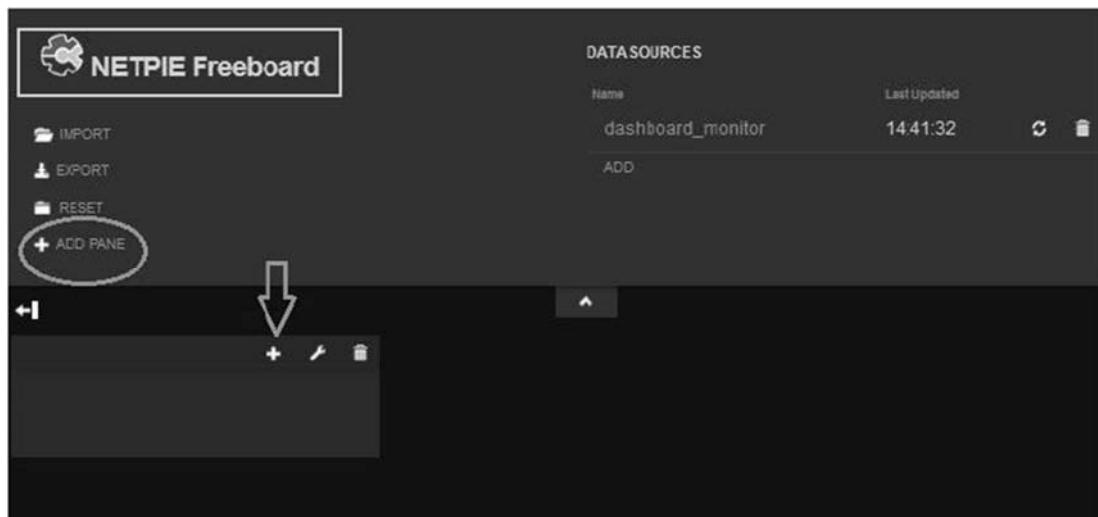
ONCONNECTED ACTION:
JS code to run after a microgear datasource is connected

SAVE CANCEL

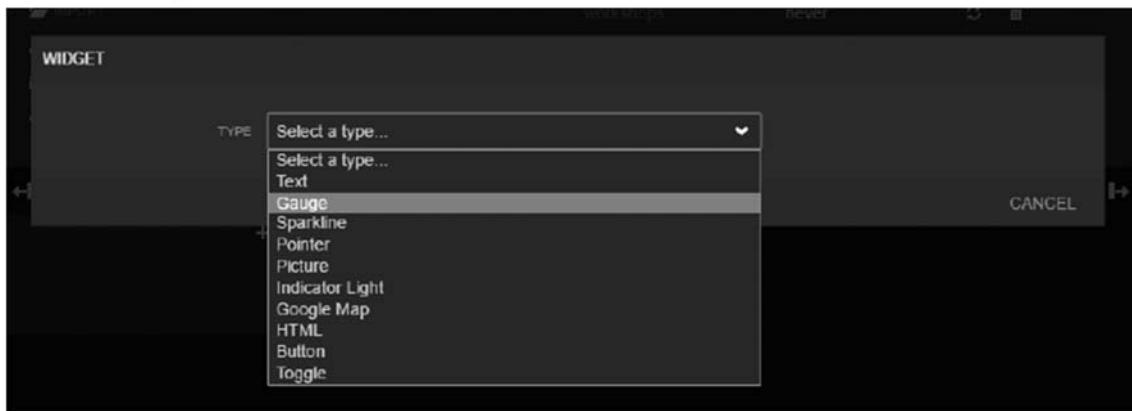
115



Workshop#4 NETPIE Freeboard



Workshop#4 NETPIE Freeboard



117

Workshop#4 NETPIE Freeboard

The screenshot shows the 'Widget' configuration dialog for a 'Gauge' type. The configuration fields are as follows:

- TYPE:** Gauge
- TITLE:** Temperature
- VALUE:** datasources["dashboard_mon"]["/TrainIoT/dht"].split(",")[1]
- UNITS:** C
- MINIMUM:** 0
- MAXIMUM:** 100

At the bottom right of the dialog are 'SAVE' and 'CANCEL' buttons.

118

Workshop#4 NETPIE Freeboard

Widget 1:

- TITLE : Humidity
- VALUE : datasources["YourDatasourceName"]["/YourAppID/dht"].split(",")[0]
- UNIT : %
- MINIMUM : 0
- MAXIMUM : 100



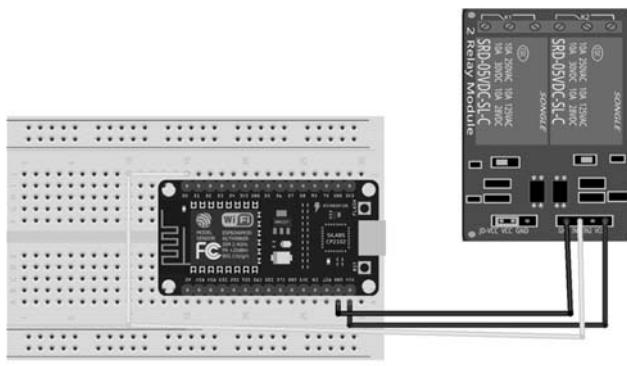
Widget 2:

- TITLE : Temperature
- VALUE : datasources["YourDatasourceName"]["/YourAppID/dht"].split(",")[1]
- UNIT : C
- MINIMUM : 0
- MAXIMUM : 50

119

Workshop#4 Control devices with NETPIE Freeboard

Hardware

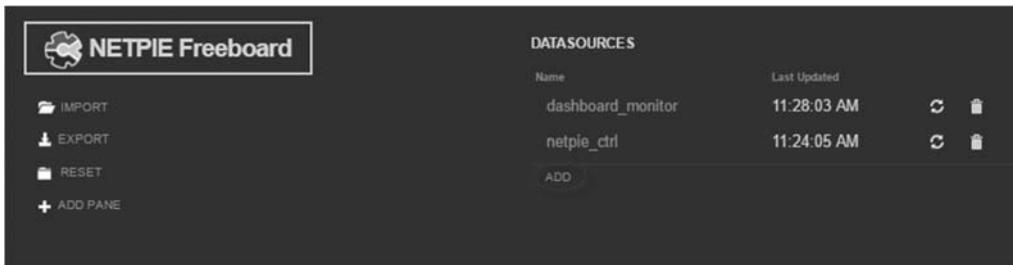


Software

Upload program NETPIE_Ctrl_Freeboard.ino to board

120

Workshop#4 Control devices with NETPIE Freeboard



121

Workshop#4 Control devices with NETPIE Freeboard

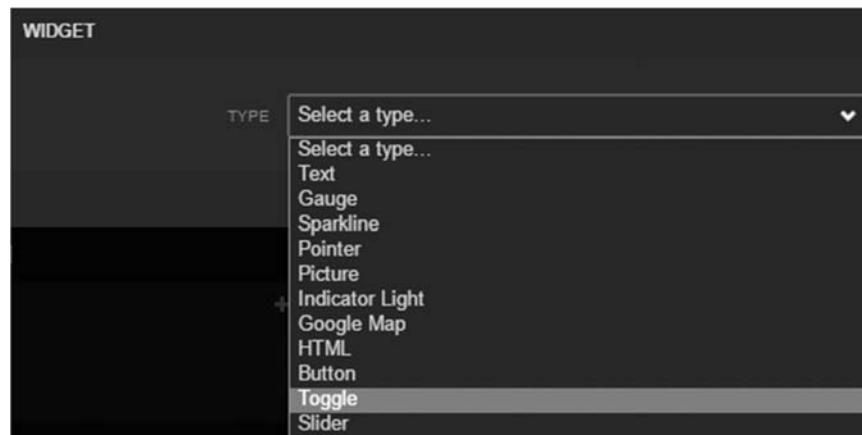
This screenshot shows the 'DATA SOURCE' configuration dialog for a 'netpie_ctrl' datasource. The dialog has the following fields:

TYPE	NETPIE Microgear
NAME	netpie_ctrl
APP ID	TrainIoT
KEY	Ct0KTCsmAEQphIk
SECRET	9AScuAJcCiAHfK4w13zW5P65E
SUBSCRIBED TOPICS	/piectrl/state
ONCONNECTED ACTION	(empty)

Below the form, there's a note: "Topics of the messages that this datasource will consume, the default is /# which means all messages in this app ID." At the bottom right, there are 'SAVE' and 'CANCEL' buttons.

122

Workshop#4 Control devices with NETPIE Freeboard



123

Workshop#4 Control devices with NETPIE Freeboard

A screenshot of the NETPIE Freeboard configuration interface for a 'Toggle' widget. The configuration fields are as follows:

- TYPE:** Toggle
- TOGGLE CAPTION:** CONTROL BUTTON
- TOGGLE STATE:** `datasources['netpie_ctrl']['TrainIoT/pietctrl/state']==1` (with '+ DATASOURCE' and 'JS EDITOR' buttons)
- ON TEXT:** ON
- OFF TEXT:** OFF
- ONTOGGLEON ACTION:** `microgear['netpie_ctrl'].chat('pietctrl','1')` (with 'JS code to run when a toggle is switched to ON')
- ONTOGGLEOFF ACTION:** `microgear['netpie_ctrl'].chat('pietctrl','0')` (with 'JS code to run when a toggle is switched to OFF')
- ONCREATED ACTION:** (empty field)

At the bottom right are 'SAVE' and 'CANCEL' buttons.

124

Workshop#4 Control devices with NETPIE Freeboard



125

Workshop#4 Control devices with NETPIE Freeboard

A screenshot of the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a New button. The main window shows a sketch titled "NETPIE_Ctrl_Freeboard". The code is as follows:

```
13 char stateOutdated = 0;
14 char buff[16];
15
16 MicroGear microgear(client);
17
18 void sendState() {
19   if (state == 0)
20     microgear.publish("/piectrl/state","0");
21   else
22     microgear.publish("/piectrl/state","1");
23   Serial.println("send state...");
24   stateOutdated = 0;
25 }
26
27 void updateIO() {
28   if (state >= 1) {
29     digitalWrite(TD, HIGH);
30   }
31 }
```

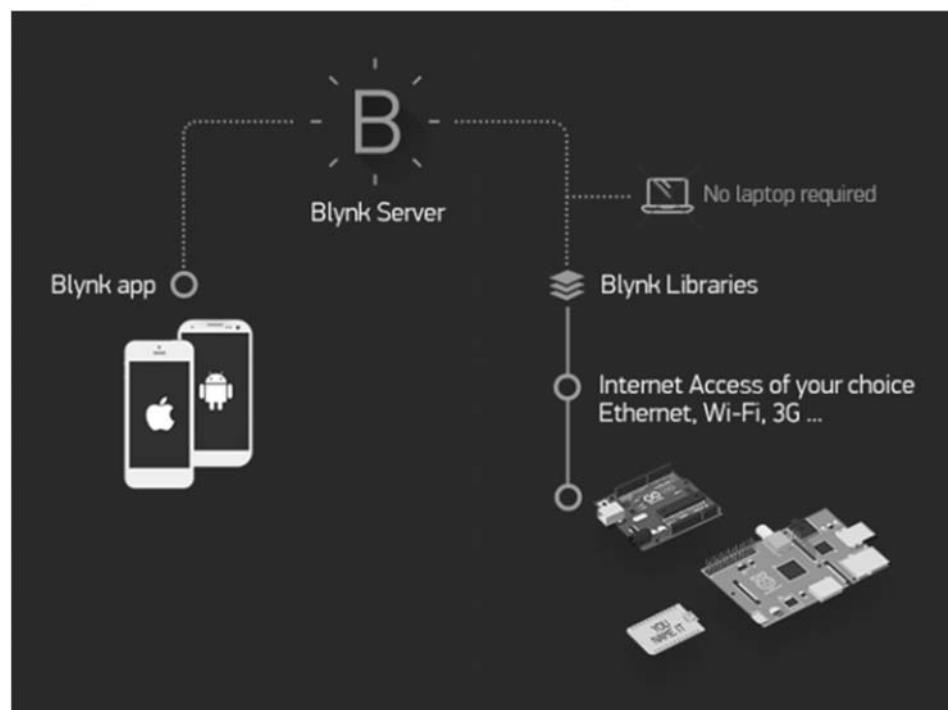
The bottom-left panel shows the upload progress: "Done uploading." followed by "setting serial port timeouts to 1 ms", "setting serial port timeouts to 1000 ms", and "flush complete". The bottom-right panel is a serial monitor window titled "COM3" with a "Send" button. It displays the following log:

```
r$0Ül,dà|000000el&0"!zf0i0('80p0p-Gozlonz&0#Op
..WiFi connected
IP address:
192.168.1.112
Connected to NETPIE...
send state...
Incoming message --> 1
send state...
Incoming message --> 1
send state...
Incoming message --> 0
send state...
Incoming message --> 1
send state...
Incoming message --> 1
send state...
Incoming message --> 0
send state...
Incoming message --> 1
send state...
Incoming message --> 0
send state...
Incoming message --> 1
send state...
```

At the bottom of the serial monitor, there are checkboxes for "Autoscroll", "No line ending", and a baud rate dropdown set to "115200 baud".

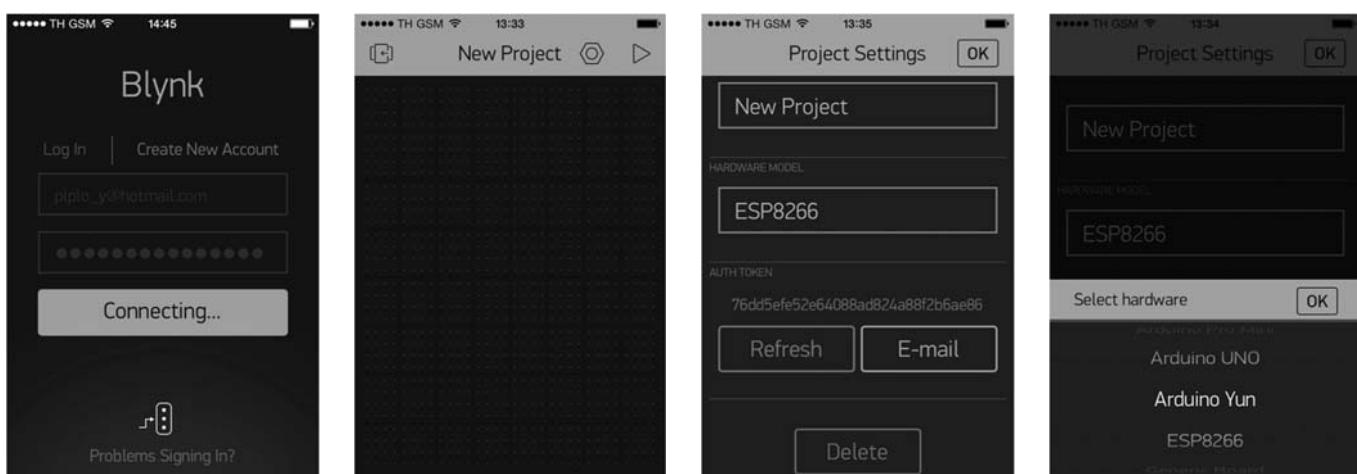
126

Workshop#4 Control IoT via Blynk



127

Workshop#4 Control IoT via Blynk



128

Workshop#4 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>

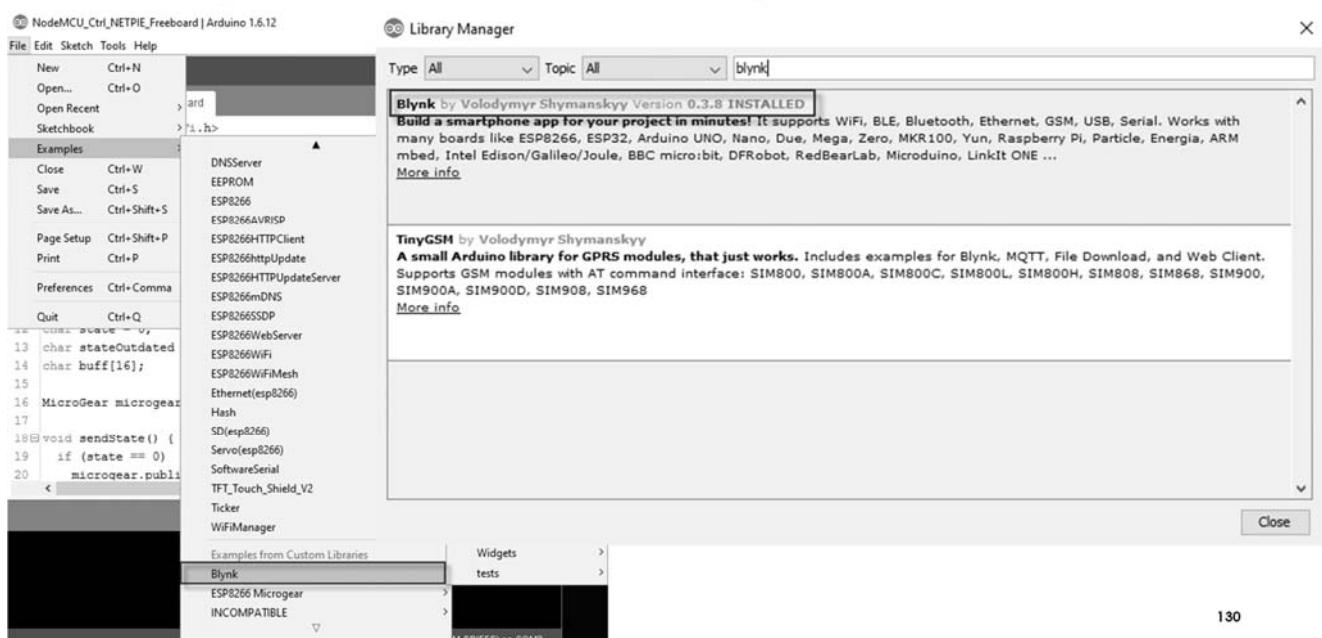
- <http://www.blynk.cc>

twitter.com/blynk_app

www.facebook.com/blynkapp

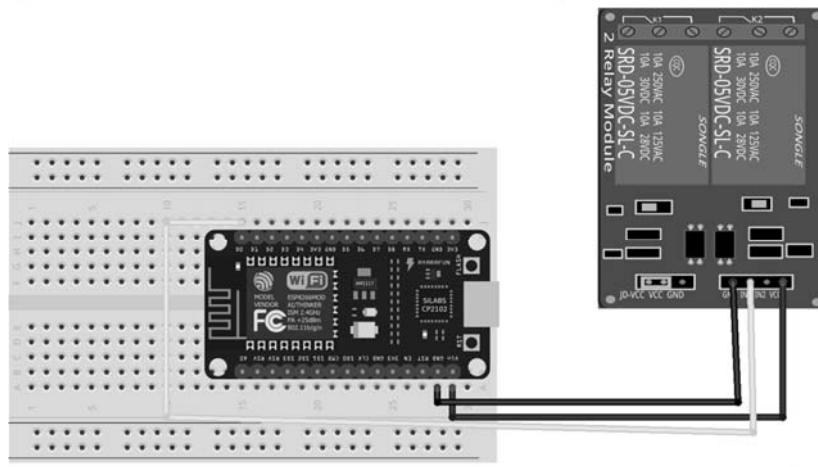
129

Workshop#4 Control IoT via Blynk



130

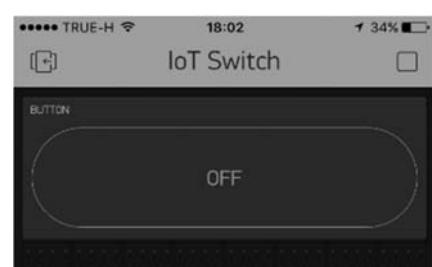
Workshop#4 Control IoT via Blynk



131

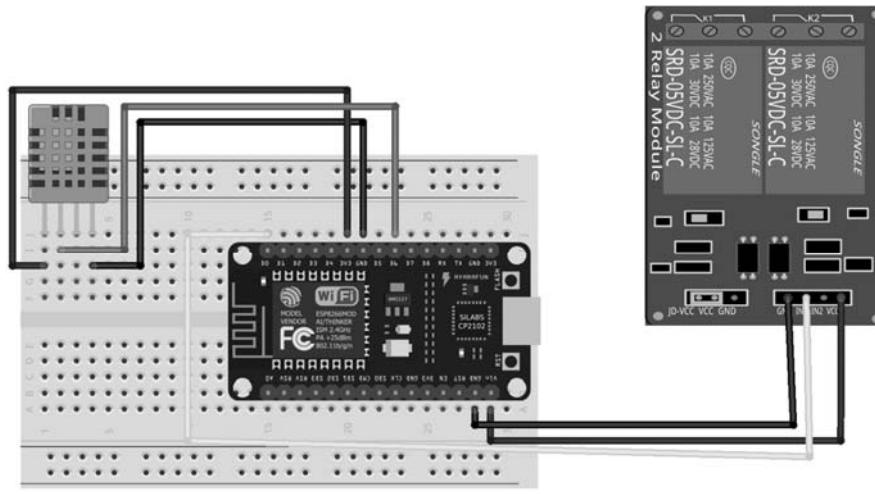
Workshop#4 Control IoT via Blynk

```
#define BLYNK_PRINT Serial // Comment this out to  
// disable prints and save space  
#include <ESP8266WiFi.h>  
#include <BlynkSimpleEsp8266.h>  
  
// You should get Auth Token in the Blynk App.  
// Go to the Project Settings (nut icon).  
char auth[] = "your_authentication_Key";  
  
// Your WiFi credentials.  
// Set password to "" for open networks.  
char ssid[] = "....."; // your SSID  
char pass[] = "....."; //your password  
  
void setup()  
{  
    Serial.begin(115200);  
    Blynk.begin(auth, ssid, pass);  
}  
  
void loop()  
{  
    Blynk.run();  
}
```



132

Workshop#4 Control IoT via Blynk



133

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 //GPIO12 Or D6 pin
#define DHTTYPE DHT11

DHT_Unified dht(DHTPIN, DHTTYPE);

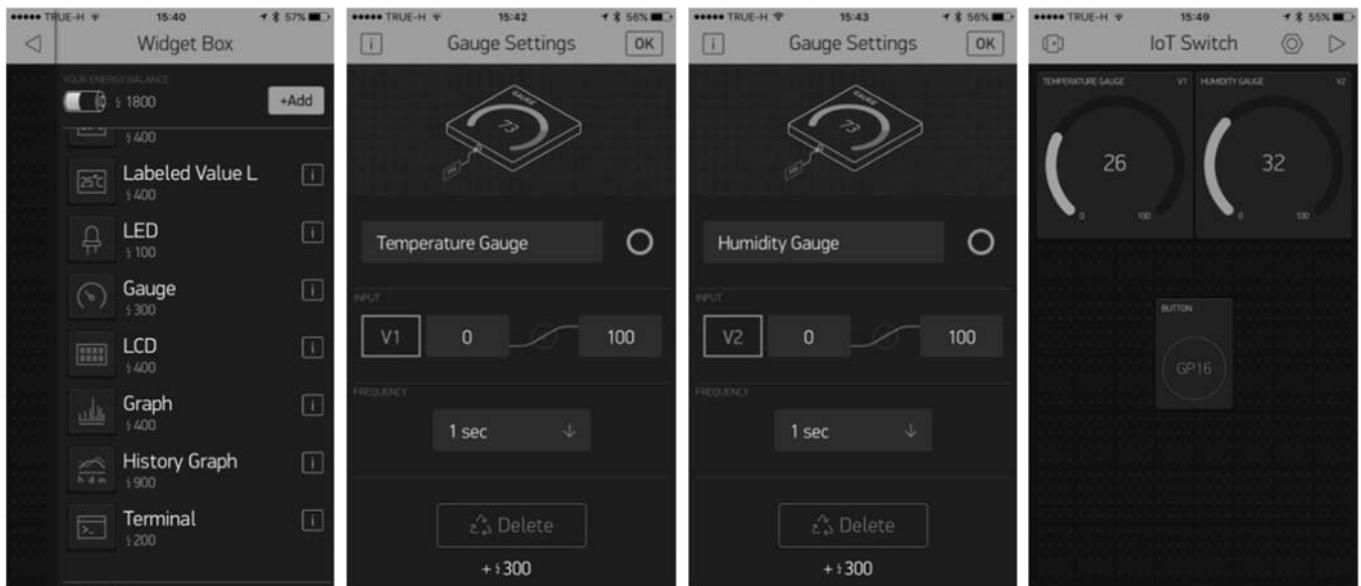
char auth[] = "your_Authentication_Key";

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

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

134

Workshop#4 Control IoT via Blynk



135

Question....



136

Further resources

- o <https://www.arduino.cc/>
- o <https://thingspeak.com/>
- o <https://netpie.io/>
- o <https://espressif.com/en/products/hardware/esp8266express/overview>
- o <http://www.esp8266.com/>
- o <http://www.blynk.cc/>
- o <https://www.rs-online.com/designspark/rs-toolbox>

137

Course feedback and instructor evaluation



EXCELLENT
 GOOD
 AVERAGE
 POOR

138

Final Question....



Email: prarinya.e@gmail.com

139



140