DIY IOT Lab — Uniforum Chicago May 2018

Author Dave Putz <dwputz@gmail.com>

Author Connie Sieh <cjsieh@gmail.com>

License Creative Commons Attribution 4.0 International License

*Your Station ID is the word "station" followed by the two digits on the bottom of your breadboard or case *

DIY IOT Lab Outline

Topology of iotlab network

Lab 0
: Arduino IDE, add board support for ESP8266 and libraries needed by lab

Lab 1a
: Configure the Arduino IDE for the ESP8266

Lab 1
: Familiarize yourself with the Arduino IDE, compiling, and uploading a program to the ESP8266

Lab 2
: Program the ESP8266 to gather and report DHT11 data from a sensor

Lab 3
: Program the ESP8266 to publish DHT11 Temperature sensor data to a MQTT Broker

> **Note** The following labs are included here for your future reference, we will run labs 7 and 8 as demonstrations only

Lab 4
: Program the ESP8266 to be a Web Server

Lab 5
: Program the ESP8266 to be a Access Point and a Web Server

Lab 6
: Program the ESP8266 to display DHT11 sensor data via a local web server as a Access Point

Lab 7

Familiarize yourself with Node-RED

Lab 8

Setup ESP8266 to publish to MQTT Broker, and configure a Node-RED flow to subscribe to MQTT and send email

Lab 9

Configure a Node-Red flow to subscribe to a MQTT broker and publish to the cloud

Lab 10

Using ESPEasy to configure an ESP8266 with no coding necessary

LAB 0 :: Arduino IDE, install ESP8266 and libraries required for lab

This lab is available at https://github.com/siehputz/uniforum2018

1. Log into your system as the user you wish to run the "arduino IDE" as
    - Verfiy that the "arduino IDE" is installed
        - On linux
            - ls "Arduino/"
        - If not installed then go to
            - https://www.arduino.cc/en/main/software
            - Download the Arduino IDE
                - version 1.8.5
                - Versions that are availble via apt or yum are too old for this lab
            - Follow installation instructions
                - https://www.arduino.cc/en/Guide/HomePage Go to the "Install the Arduino Desktop IDE" section
2. Start Arduino IDE

3. If you have not previously set up the Arduino IDE for ESP8266, click on
    - Click "File→Preferences"
        - Check "Display Line Numbers"
        - Enter in the *Additional Boards Manager URLs* box the following
            - http://arduino.esp8266.com/stable/package_esp8266com_index.json
    - Click on "Tools→Board→Boards Manager" menu and install the *esp8266* package
        - You can enter "esp8266" in top right search box to help find the esp8266 entry
4. Some extra libraries (code provided by a 3rd party) need to be installed
    - Click on *Sketch→Include Libraries→Manage Libraries*
    - You can enter items in top right search box
        - The following libraries need to be installed:
            - ESP8266 examples (ESP8266 sketches examples)
            - Adafruit MQTT Library
            - DHT Sensor library by Adafruit
            - Adafruit Unified Sensor by Adafruit (dependency of DHT Sensor)
5. Verify that your arduino user is part of the "dialout" group if you are running Linux
    - grep dialout /etc/group
        - dialout:x:18:arduinouser
            - usermod -a -G dialout arduinouser
                - Logout of arduinouser and log back in to enable this group change
6. It is a good practice to restart the Arduino IDE after installing boards and libraries

LAB 1a :: Arduino IDE, Configure for the ESP8266 Nodemcu Lab board

1. Log into your system as the user that installed the Adruino IDE
2. Plug in the ESP8266 using the USB cable
3. Start the Arduino IDE
4. Click on Tools and verify
    - Board
        - `NodeMCU 1.0 (ESP 12-E Module)`
    - "Upload Speed" is set to
        - 115200
    - "Port" is set to
        - on Linux
            - /dev/ttyUSB0 or
            - /dev/ttyUSB1
        - on Windows
            - COMxx
        - on MAC
            - /dev/cu.CPxxxxxx

LAB 1 :: Arduino IDE, compiling, and uploading a program to the ESP8266

1. Click on *File→Examples→ESP8266→Blink* This should bring up a code window with the `Blink` program loaded.
2. Take a look at this small program(sketch). Note that it has two main functions . All Arduino IDE sketches will contain at least these two functions.
    - setup() function, which is run once when the sketch starts
    - loop() function, which is run in a loop after setup completes
3. Click on the 'Check` icon on the toolbar to compile the sketch(program).

- If you see no errors in the output window, click on the `Right Arrow` icon on the toolbar to upload the program to your ESP8266. You should see a progress indicator in the output window and the Blue LED (near antenna) blinking on the ESP8266.
- When the upload completes, the ESP8266 will start running the program. You should see the Red or Blue(near usb connector) LED turning off and on with a pause of one second between changes.

4. Edit the program to change the timing of the blinks. This will be done with the code at line 21 and 23

Line Num	Before Code	After Code	Notes
21	`delay(1000);`	`delay(1500)`	the value is in milliseconds , on time
23	`delay(2000);`	`delay(3000)`	the value is in milliseconds , off time

5. Compile and upload the modified program.
    - Can just click on the "upload arrow", and the IDE will compile before uploading
6. Verify that the LED blink rate on your ESP8266 has successfully changed.

LAB 2 :: DHT11(Temperature and Humidity) Sensor

1. Load the example program for a DHT sensor.
    - Click on *File→Examples→DHT sensor library→DHTtester*

> **Tip** It is a good programming practice to use defined values for pin numbers rather than actual numbers

2. change

Line Num	Before Code	After Code	Note
6	`#define DHTPIN 2`	`#define DHTPIN D5`	DHT11 sensor is connected to GPIO(General Purpuse IO) pin D5

3. uncomment(remove the //) on the
    - define line for DHTTYPE DHT11

Line Num	Before Code	After Code	Note
9	`//#define DHTTYPE DHT11`	`#define DHTTYPE DHT11`	have a dht11 so uncomment , remove //

4. comment(Insert //) on the
    - line for DHTTYPE for DHT22

Line Num	Before Code	After Code	Note
10	`#define DHTTYPE DHT22`	`//#define DHTTYPE DHT22`	comment out dht22 line , add //

5. Because we are using a baud rate of 115200, you need to change

Line Num	Before Code	After Code	Note
27	`Serial.begin(9600);`	`Serial.begin(115200);`	Change speed of serial

6. Compile and upload the sketch(program).
7. Open the serial monitor window by clicking on *Tools→Serial Monitor*.
    - You should see the temperature and humidity readings being displayed. Note that the Serial.print() functions in your program will always display to this monitor window.
8. Try holding the sensor in your hand and/or blowing on it to see the readings change. Be careful not to detach any wires when doing this.
9. Click on *File→Save As* and save it as ‘lab2’ to save this version

LAB 3 :: MQTT & DHT11

1. Open the MQTT example - click on *File→Examples→Adafruit MQTT Library→mqtt_esp8266*
2. Read through the source code. Note that this example was written to use a specific MQTT broker (io.adafruit.com) (most MQTT Brokers are compatible). We are going to communicate with a "MQTT Broker" called Mosquitto running on a Raspberry Pi on the "iotlab" network.

3. Configure the ESP8266 Wifi to use the local "iotlab" network. Change the program as indicated below

Line Num	Before code	After code	Note
24	`#define WLAN_SSID "…your SSID…"`	`#define WLAN_SSID "iotlab"`	(SSID of our created lab network)
25	`#define WLAN_PASS "…your password…"`	`#define WLAN_PASS "iotlabpass"`	(PASS of our created lab network)

4. Configure the ESP8266 to communicate to a MQTT server. You will use a MQTT Broker that is on the local iotlab network. Make changes as indicate below

Line Num	Before code	After code	Note
29	`#define AIO_SERVER "io.adafruit.com"`	`#define AIO_SERVER "192.168.10.201"`	`IP address of the Raspberry Pi running a MQTT Broker`
31	`#define AIO_USERNAME "…your AIO username…"`	`#define AIO_USERNAME "stationxx"`	`where xx is your station number on bottom of breadboard/case of esp8266`
32	`#define AIO_KEY "…your AIO key…"`	`#define AIO_KEY ""`	`change to ""`

5. Change the name of your MQTT topic, which is done in the call to
    - "Adafruit_MQTT_Publish()"
        - Change the name to `temp`
        - Change the literal from `feeds/photocell` to `feeds/temp`

Line Num	Before Code
48	`Adafruit_MQTT_Publish photocell = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/pho`

- to

Line Num	After Code
48	`Adafruit_MQTT_Publish temp = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/temp");`

6. Add in the necessary DHT code. You can cut & paste from the code block below the table

- Add after line 20

New Line Num	New code
21	`#include "DHT.h"`
22	`#define DHTPIN D5`
23	`#define DHTTYPE DHT11`
24	`DHT dht(DHTPIN, DHTTYPE);`

```
#include "DHT.h"
#define DHTPIN D5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

- Initialization of the DHT
  - Add to *setup();* after line 85 the following code

New Line Num	New Code
86	`dht.begin();`

- Add the read from the DHT to loop();
- Above the comment line in the loop() that follows

```
// Now we can publish stuff!
```

- add the code from Lab 3 that reads the temperature from the DHT sensor after line 106 , cut and paste from the code block below the table

New Line Num	New code
107	`// Read temperature as Fahrenheit (isFahrenheit = true)`
108	`float f = dht.readTemperature(true);`
109	`if (isnan(f)) {`
110	`Serial.println("Failed to read from DHT sensor!");`
111	`}`

```
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);
if (isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
}
```

7. Change line 114 to "temp" instead of "photocell" since we changed the name to temp earlier .

Line Num	Before Code	After Code	N
114	`Serial.print(F("\nSending photocell val "));`	`Serial.print(F("\nSending temp val "));`	

8. Change line 115 to "f" instead of "x" since we changed the name to f earlier .

Line Num	Before Code	After Code	Note
115	`Serial.print(x);`	`Serial.print(f);`	

9. Change line 117 to call "temp.publish" instead of "photocell.publish" since we changed the name to temp earlier .

Line Num	Before Code	After Code	Note
117	`if (! photocell.publish(x++)) {`	`if (! temp.publish(f)) {`	`Change photocell to temp`

10. Add a "delay of 15 sec" between publishes
    - Hint , see lab 1
11. Compile and upload the sketch
12. Watch the instructor's screen
    - It is displaying all of the "publish" data to the MQTT Broker on the Raspberry Pi
      - Look for your stationxx entries
      - example
        - station12/feeds/temp
13. Click on *File→Save As* and save as 'lab3'

LAB 4 :: ESP8266 running a Web Server

1. Load the example ESP8266 web server sketch *File→Examples→ESP8266WebServer→HelloServer*
2. Look at the code. Note in particular that the loop() function consists of only a single function call.

```
server.handleClient();
```

1. Modify the values assigned to

- SSID “<your local ssid>”
- PASS “<your local ssid password>”

2. Compile and upload the sketch
3. Since multiple stations may be running this code at the same time, you will need to look at the serial monitor when the sketch starts running to see what IP address was assigned to it. The IPADDRESS is provided by the iotlab router via dhcp.
4. Using a smartphone or tablet or your station connect to
    - SSID: <your local ssid>
    - PASS: <your local ssid password>
5. enter your ESP8266’s IP address in the URL bar
    - You should see a screen back that contains just a hello message. Note that this response was created by the handleRoot() function in the sketch, which was assigned to the root page by the line:

```
server.on("/", handleRoot);.
```

6. Click on *File→Save As* to save this version

LAB 5 :: ESP8266 as an Access Point and Web Server

1. Load the example *File→Examples→ESP8266Wifi→WiFiAccessPoint* and look at the differences in the wifi initialization.
    - You will need to make your own "network" since the ESP will be a Access Point

Line Num	Before Code	After Code
38	`const char *ssid = "ESPap";`	`const char *ssid = "<accesspoint ssid>";`
39	`const char *password = "thereisnospoon";`	`const char *password = "<accesspoint passwd>";`

3. Insert
    - Since the default IPADDRESS is 192.168.4.1 you will need to change that if there are many EPS8266
    - Insert the following 3 lines; where xx makes the ip address unique in your network after line 55

Line Num	Before Code	Note
56	`IPAddress apIP(192,168,xx,1);`	Replace "xx" to make uni
57	`WiFi.mode(WIFI_AP_STA);`	
58	`WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));`	

```
IPAddress apIP(192,168,xx,1); // Replace "xx" to make unique ip address
WiFi.mode(WIFI_AP_STA);
WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
```

- It should now look like , assuming xx was replaced with 14

```
IPAddress apIP(192,168,14,1);
WiFi.mode(WIFI_AP_STA);
WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
WiFi,softAP(ssid,password);
```

3. Compile and upload the sketch
4. Using a smartphone or tablet or your station connect to
    - the network with the SSID and the password you created
    - Enter the IP address you specified in a web browser
        - You should see a screen back that contains just a hello message
5. Click on *File→Save As* and save it as 'lab5'

LAB 6 :: ESP8266 as a Access Point and Web Server displaying DHT11 data

Change the LAB 6 program to return temperature and humidity readings when a page other than the root page is used.

Copy & paste the necessary DHT lines from the DHT example as you did in Lab 2.

1. Add the DHT definition lines

```
#include "DHT.h"
#define DHTPIN D5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

2. Add the DHT initalize to setup();

```
dht.begin();
```

3. Add the code to read the DHT11 sensor to a new handleTemp() function

```
void handleTemp() {
  float h = dht.readHumidity();
  float f = dht.readTemperature(true);
}
```

4. Add server.send() call to handleTemp() to send the data back to the client when the web page requests it
    - https://links2004.github.io/Arduino/d3/d58/class_e_s_p8266_web_server.html

```
void handleTemp() {
  String message = "";
  float h = dht.readHumidity();
  float f = dht.readTemperature(true);
  message = "Temperature is : ";
  message += f;
  message += " Humidity is : ";
  message += h;
  message += "\n" ;
  server.send(200, "text/plain", message);
}
```

5. Call handleTemp in setup() after server.on("/", handleRoot) when user enters URL ending in **/temp**

```
server.on("/", handleRoot);
server.on("/temp", handleTemp);
```

6. Compile and upload the sketch.
7. Using a smartphone or tablet connect to the wifi network with the SSID and password you created.
8. Enter the IP address followed by `temp` in the URL bar. This should return your web page with the data from the DHT displayed.
9. Click on *Save As* and save this as 'lab6'

LAB 7 :: Familiarize Yourself with Node-RED

Create a basic Node-RED flow and see how to connect nodes

1. To start Node-RED
    - In a terminal window type
        - node node_modules/node-red/red
    - Using a browser
        - go to "http://127.0.0.1:1880/" to navigate to the Node-RED screen
2. Open a browser on your lab system.
3. Note that 1880 is the default port for Node-RED, and can be changed in the settings.js file.
4. You will see the Node-RED screen, with node choices on the left, the design screen in the middle, and info and debug tabs on the right.
5. Click and drag an ‘inject’ input node onto the design screen.
6. Double clicking on the new node will bring up an edit window on the screen. For now, just change the name to ‘Time’ and click ‘Done’.
7. Click and drag a ‘debug’ output node onto the design screen.
8. Change its name to ‘Test’.
9. Connect the two nodes by clicking on the right side of the ‘Time’ node and dragging a line to the left side of the ‘Test’ node.
10. Click on the ‘Deploy’ button on the top right. Note that in Node-RED no changes take effect until they are deployed.
11. Click on the ‘debug’ tab on the right side to show the debug screen.
12. Click on the button on the left side of the ‘Time’ node. This activates that node. Look at the debug window to see the timestamp displayed.

Part 2 ::Modify the flow to add a function node

1. Click on the line between the two nodes, and hit ‘delete’ on the keyboard to remove it.

2. Click and drag a function node between the two existing nodes. You can move the existing nodes around on the screen if needed to make space.
3. Double-click on the function node to bring up the edit window, and enter the following code:

```
// Create a Date object from the payload
var date = new Date(msg.payload);
// Change the payload to be a formatted Date string
msg.payload = date.toString();
// Return the message so it can be sent on
return msg;
```

4. Click 'Done', and then connect the output of the 'Time' node to the input of the function node, and the output of the function node to the input of the 'Test' node.
5. Deploy the flow, then click on the activate button on the 'Time' node. You should now see a formatted date in the debug window.

LAB 8 :: Setup ESP8266 to publish to MQTT broker, and configure a Node-RED flow to subscribe to MQTT and send email

1. Reload the MQTT sketch from Lab 3 into your ESP8266.
    - Using a browser
        - go to "http://127.0.0.1:1880/" to navigate to the Node-RED screen
            - If Node-RED is not running see the begining of lab 8
2. On your Node-RED screen, click on the "+" button on the top right of the design screen to open another flow.
3. Double-click on the "flow name" to bring up its edit window. Change the name to "MQTT-Test". Note that this is also where you would delete a flow that you no longer want. Click "Done".
4. Drag a "mqtt node" from the input list onto the workspace screen. Note that there is also a mqtt node in the output section, which we do not want to use here.
5. Double-click on the "mqtt node" to bring up the edit window.

- If your mosquitto broker and Node-RED are running on the same system, click on the edit icon and enter a servername of ‘localhost’, otherwise enter the name or IP address of the system running mosquitto; then click ‘Add’.

6. Enter the topic name you used from Lab 3 in the "Topic box".
7. Change the "QOS" from 2 to 0.
8. Change the "name" to “MQTT subscribe” and hit “Done”.
9. Click and drag an "e-mail output node" from the “social” group. Note that you may have to scroll down to see this one. Also, make sure you get the output node (connector on the left) and not the input node (connector on the right).
10. Double-click the "email node" to bring up the edit window.
11. Enter your own email address in the “To” box, “penguiconlab1” in the Userid box, and “Test.123!” in the password box.
12. Change the name to gmail-out, and click “Done”
13. Connect the two nodes, and DEPLOY.
14. Log into your email account, and check for incoming messages. Note that you could be getting one every 15 seconds, so if it is working you may want to unplug the ESP8266 after a few emails, or change the delay timer in your sketch.

Further possibilities - If time permits, you may want to try out other nodes. Some suggestions:

A) Add a switch node to test on the message from the MQTT broker; and only email if it exceeds some value. Send all other values to a debug node.

B) Add a function node to convert the temperature from what your are getting (either Fahrenheit or Celsius) to the other scale. One possible function to do this

```
// convert Celsius to Farenheit
var newtemp = new int (msg.payload);
msg.payload= (newtemp * 1.8) + 32;
return msg;
```

C) Insert an rbe (report-by-exception) node to only pass on the message if the temperature has changed.

LAB 9 :: Setup ESP8266 to publish to MQTT broker, and configure a Node-RED flow to subscribe to MQTT and upload to the cloud

1. If you do not have an account at adafruit.io, you will need to use a web browser and go to https://io.adafruit.com/. Click on the 'Get Started for Free' box on the top right of the screen. Enter the required info (Name, email, username, and password) and click on 'Create Account'
2. Reload the MQTT sketch from Lab 4 or the ESPEASY Framework from lab 10 into your ESP8266.
3. If you did Lab 9, you can use that flow as a base for this one. Delete the email node, and go to Step 11 below.
4. On your Node-RED screen, click on the '+' button on the top right of the design screen to open another flow.
5. Double-click on the flow name to bring up its edit window. Change the name to 'Cloud-Test'. Note that this is also where you would delete a flow that you no longer want. Click 'Done'.
6. Drag a mqtt node from the input list onto the design screen. Note that there is also a mqtt node in the output section, which we do not want to use here.
7. Double-click on the mqtt node to bring up the edit window. Since your mosquitto broker and Node-RED are running on the same system, enter 'localhost:1883' in the Server box.
8. Enter the topic name you used from Lab 4 or Lab 10 in the Topic box.
9. Change the QOS from 2 to 0.
10. Change the name to 'MQTT subscribe' and hit 'Done'.
11. Click and drag an mqtt output node from the 'output' group. Note that you may have to scroll down to see this one. Make sure you get the output node (connector on the left) and not the input node (connector on the right).
12. Double-click the MQTT output node to bring up the edit window.

13. Select ‘Add new mqtt-broker’ in the Server drop-down. Click on the edit icon to the right to configure a new MQTT broker.
14. Under the Connection tab enter `io.adafruit.com' in the `Server' box and `8883' in the 'Port' box. Check the `Enable secure (SSL/TLS) connection box.
15. Select ‘Add new tls-config’ in the TLS Configuration box, and the click the edit icon to the right. Enter the following paths: Certificate - $HOME/node_modules/adafruit-io/node_modules/mqtt/examples/tls client/tls-cert.pem Private Key - $HOME/node_modules/adafruit-io/node_modules/mqtt/examples/tls client/tls-key.pem CA Certificate - $HOME/node_modules/adafruit-io/node_modules/mqtt/examples/tls server/crt.server1.pem
16. Uncheck the `Verify server certificate box and enter a name (such as `My Cert') in the Name box.
17. Click the ‘Update’ button on the top right and make sure your new name is chosen in the TLS Configuration box. Click the ‘Add’ button on the top right.
18. Click on the `Security` tab to bring up the login screen.
19. Enter your Adafruit username and AIO key and click ‘Add’
20. You should now be back in the MQTT node edit panel. Make sure io.adafruit.com:8883 is in the ‘Server’ box; and enter <your adafruit login>/feeds/IOTLab in the ‘Topic’ and ‘Name’ boxes. Click ‘Done’ on the top right.
21. Connect the new *MQTT out node* to your previous *MQTT in node* and DEPLOY.
22. Log into your Adafruit account, and check for your Topic under Feeds. Note that you could be getting one every 15 seconds, so if it is working you may want to unplug the ESP8266 after a few minutes, or change the delay timer in your sketch.

LAB 10 :: ESP8266 Using ESPeasy Framework

Install and Configure ESPeasy to send DHT11 sensor data to a MQTT broker

1. Look at https://www.letscontrolit.com/wiki/index.php/ESPEasy#Get_started
2. Upload the precompiled ESPeasy firmware to esp8266

> **Note** Precompiled image are located at http://www.letscontrolit.com/downloads/ESPEasy_R120.zip . You will need to install this firmware to the esp8266 as mentioned at https://www.letscontrolit.com/wiki/index.php/Tutorial_ESPEasy_Firmware_Upload. Newer versions are available.

3. Look at https://www.letscontrolit.com/wiki/index.php/Tutorial_ESPEasy_Firmware_Upload
    - You have a nodemcu version of esp8266 so we have to select
        - 4096K version of the ESPeasy firmware in the next step
4. Start a terminal window
5. Run espeasyflash.sh
    - Available in our github repo
    - espeasyflash script will ask for
        - USB Serial port to upload to
            - /dev/ttyUSB0 or
            - /dev/ttyUSB1
                - Determine which serial port with

```
ls /dev/ttyUSB*
```

    - Firmware to upload
        - 0 ESPEasy_R147_4096.bin

```
cd diyiot/Downloads/images/esp8266-frameworks/espeasy
./espeasyflash.sh
```

6. After flash is done
    - Start Arduino Serial Console

- **AFTER** all of the flash clearing is done and you see
    - *wdt reset*
        - Reboot esp8266 via reset button

7. Configure ESPeasy running on your esp8266 via ESP Easy web interface
    - Configure Network Settings
        - https://www.letscontrolit.com/wiki/index.php/ESPEasy#Config_page
            - Using a tablet or smart phone or your station look for WIFI networks with a SSID of ESP_0 (this might be confusing if many do this at the same time)
            - Connect to ESP_0 wifi network
                - Password is *configesp*
            - In a web browser go to 192.168.4.1 (default ip address , might be confusing if many do this at the same time)
                - Should see *Welcome to ESP Easy:newdevice*
                    - Click on *iotlab* Network
                    - Password *iotlabpass*
                - Click on *Connect*
                    - Should get a "Please wait for ... while trying to connect"
                    - Note the IP ADDRESS given by dhcp
                - Using a tablet, smart phone or your station switch to the Network with SSID
                    - iotlab
                    - Password *iotlabpass*
        - Review this first screen
        - Click on *Config*
            - Change "Name" to

        - easyxx
            - where xx is your station number
    - Change "Admin Password" to
        - easyxxpass
            - where xx is your station number
    - "SSID" should already be
        - iotlab
    - "WPA Key" should already be
        - iotlabpass
    - Change "Protocol"
        - "OpenHab MQTT"
    - Change "Controller IP"
        - your station 192.168.10.x" address, address was assigned via dhcp
    - Should already have "Controller Port" set to
        - 1883
            - MQTT default port
    - Change "Controller User"
        - stationxx
            - where xx is your station number
    - Change "Controller Password"
        - stationxxpass
            - where xx is your station number
- Click on "**Submit**" at botton of page to save settings to esp8266 flash
- Bring up Arduino console window to determine IP address assigned via dhcp on next reboot

        - Reboot by pressing "reset" button or via "Tools" tab "Reboot" button
        - Login to esp8266 as with ipaddress that Arduino Console shows
            - esp8266 is now on iotlab network
- Configure Hardware settings Boot State (GPIO pins)
    - https://www.letscontrolit.com/wiki/index.php/ESPEasy#Config_page
    - There should not be any need to change these for this lab
- Configure Devices settings
    - https://www.letscontrolit.com/wiki/index.php/ESPEasy#Devices_page
        - Click on the 'Devices" tab
            - Click on "Edit" of "Task 1"
                - Select Device
                    - "Temperature & Humidity - DHT"
                - Name
                    - <what ever you want>
                - Delay
                    - time in seconds
                    - 15
                - IDX/Var
                    - 1 — It is the first so 1 is good
                - 1st GPIO
                    - GPIO-14
                    - which is (D5) (look on Pin diagram)
                - Send Data
                    - checked

                - sends data to protocol/controller defined in "Network Settings"
            - Optional Settings
                - Formula Temperature - Since DHT library defaults to Celcius could change
                    - %value% * 9/5 + 32
                - Value Name 1 — Will show as name used in MQTT
                    - Should already be *Temperature*
                - Value Name 2 — Will show as name used in MQTT
                    - Should already be *Humidity*
        - Click on **Submit** to save
- Configure Tools settings
    - https://www.letscontrolit.com/wiki/index.php/ESPEasy#Tools_page
    - Click on the 'Tools" tab
        - System/Advanced
            - Look it over — no changes needed for lab
        - Settings/Save
            - Can save setting for later reload via Settings/Load
            - Good idea so as to not have to enter by hand again
        - Firmware Load
            - Allows for OTA (Over the Air) updates
        - Command
            - Can enter commands , mostly for testing
                - See https://www.letscontrolit.com/wiki/index.php/ESPEasy_Command_Reference

        - Click on **Submit** to save
        - Bring up Arduino console window to note boot messages
        - Reboot by pressing "reset" button or via "Tools" tab "Reboot" button
8. Check that esp8266 is reading the DHT11 and sending data to MQTT
    - Login to esp8266 as with ipaddress that Arduino Console shows
        - esp8266 is now on iotlab network
    - Click on Tools/Advanced
        - Change "Serial Log Level"
            - 4
        - Change "Web Log Level"
            - 4
    - Click on **Submit** to save
    - Bring up Arduino Serial console window to see serial log messages
    - Reboot by pressing "reset" button or via "Tools" tab "Reboot" button
        - Should see MQTT messages (along with others)
9. Check that MQTT on your station is receiving data from the esp8266
    - On your station in a terminal window
        - mosquitto_sub -h localhost -u stationxx -P stationxxpass -v -t '#'
            - where xx is your station number
            - Note '#' is the wildcard for all topics

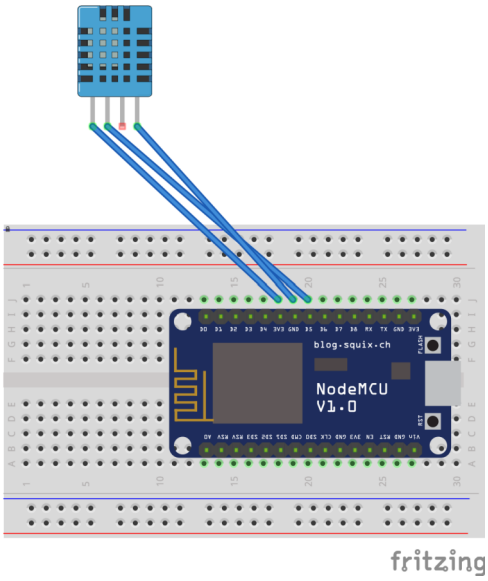
LAB SETUP :: These are the steps that would be needed to setup the lab environment on an Ubuntu Linux system.

- SOFTWARE
    - The Arduino IDE would need to be installed.

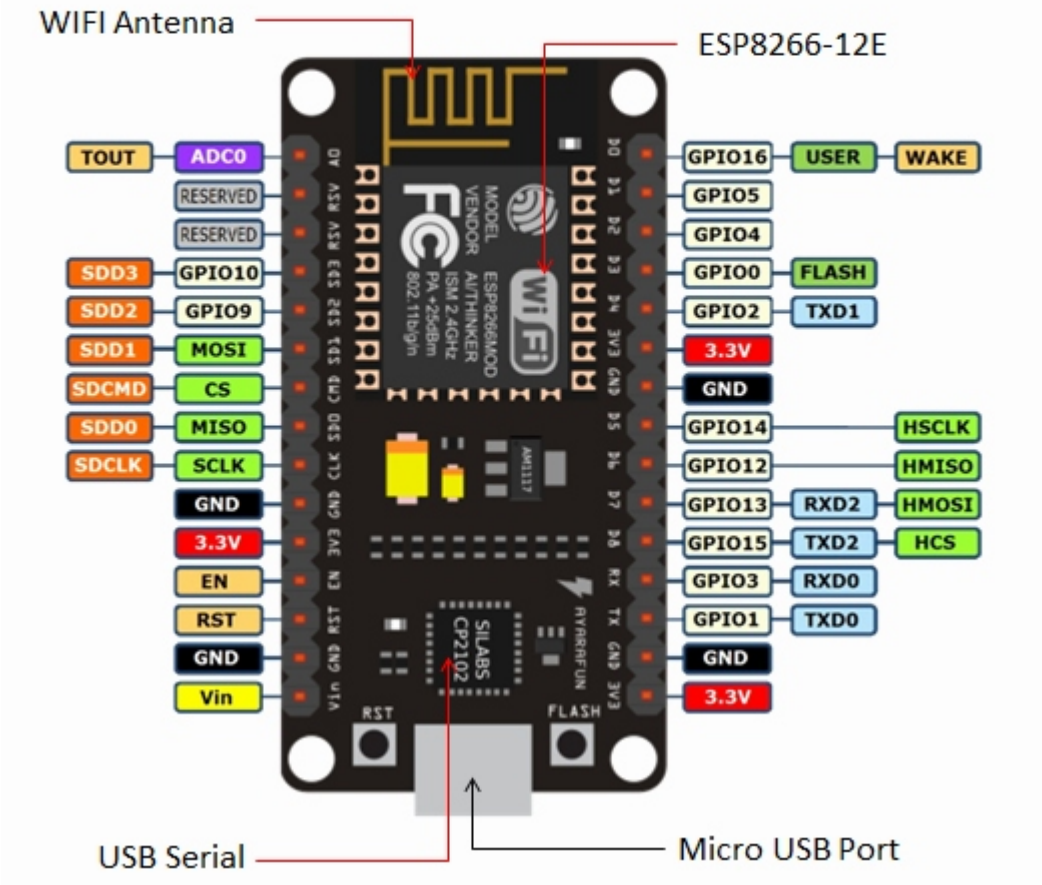
        - This can be found online at https://www.arduino.cc/en/main/software
            - Version 1.8.5 was installed
    - The Arduino IDE should be configured to pick up the ESP8266 board definition by adding *http://arduino.esp8266.com/stable/package_esp8266com_index.json* to *File→Preferences→Settings* "Additional Boards Manager URLS".
        - http://arduino-esp8266.readthedocs.io/en/latest/installing.html#boards-manager
    - Arduino libraries and ESP8266 specific libraries have been installed
        - In the Arduino IDE, Click on *Sketch→Include Libraries→Manage Libraries*
            - The following libraries need to be installed:
                - ESP8266
                - Adafruit MQTT Library
                - DHT Sensor library by Adafruit
                - Adafruit Unified Sensor by Adafruit (dependency of DHT Sensor)
        - ESP8266 Arduino reference doc
            - http://arduino-esp8266.readthedocs.io/en/latest/
    - The mosquitto MQTT broker needs to be installed. This can be found online at https://mosquitto.org/download/ or by using `apt-get mosquitto`.
    - The Node-RED flow manager software needs to be installed. Instructions on how to do this can be found at https://nodered.org/docs/getting-started/installation .
- HARDWARE
    - A NodeMCU ESP8266 development board needs to be mounted on a breadboard and wired up to a DHT11 temperature/humidity sensor.
        - NodeMCU ESP8266 v1.0

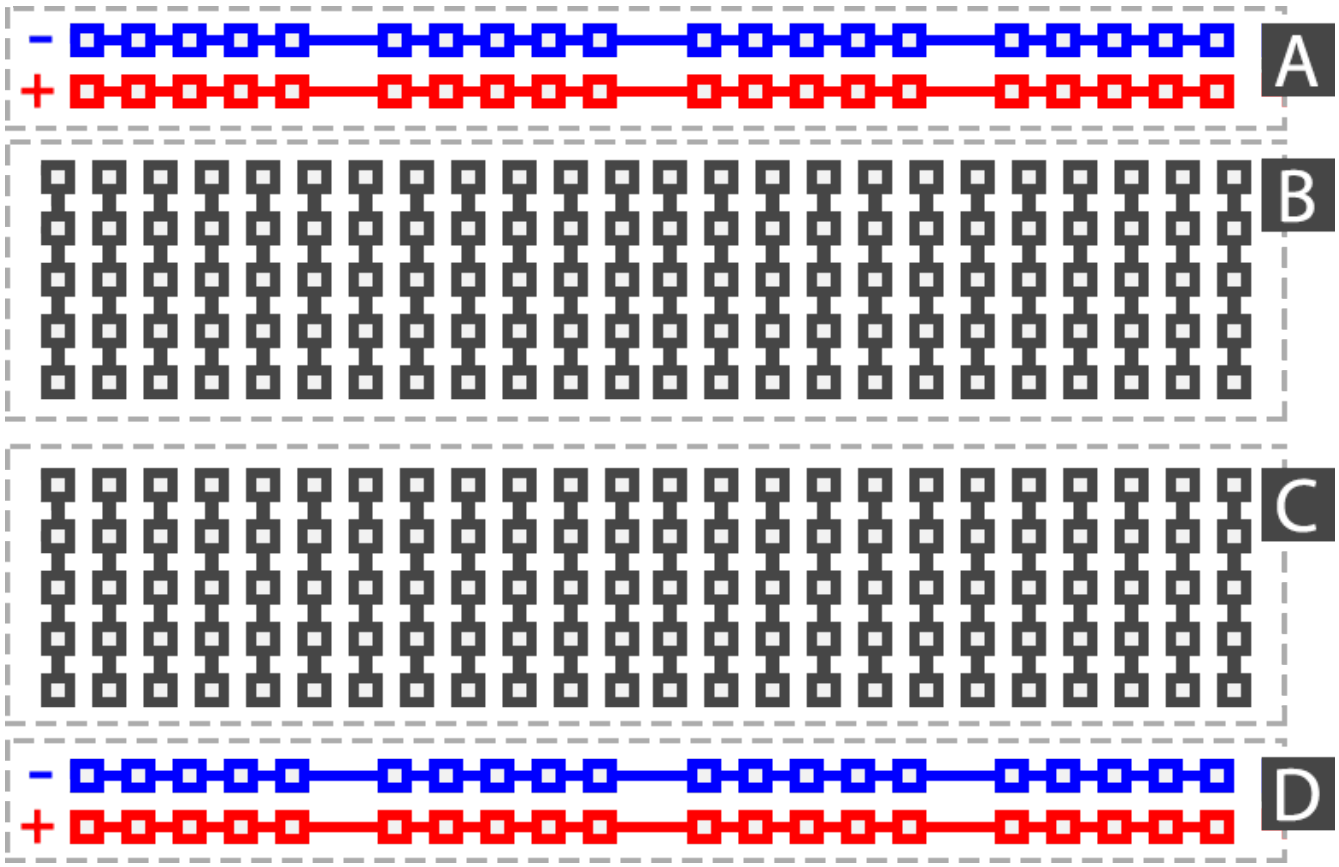
    - https://github.com/nodemcu/nodemcu-devkit-v1.0
- DHT11 breakout board
    - The + pin of the DHT11 goes to 3v3
    - The – pin of the DHT11 goes to GND
    - The OUT pin of the DHT11 goes to D5

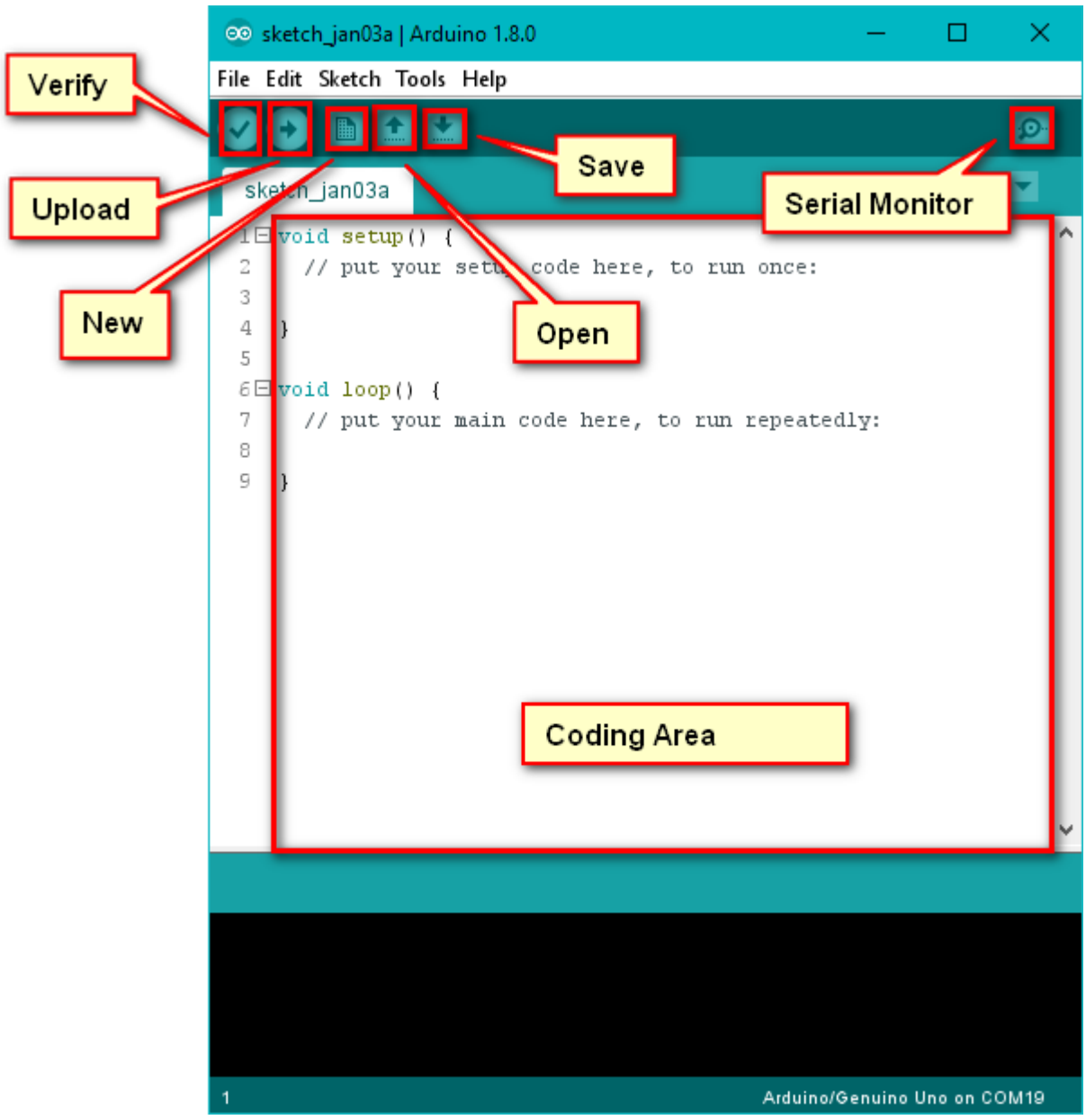
NodeMCU/DHT11 for DIY IoT Lab

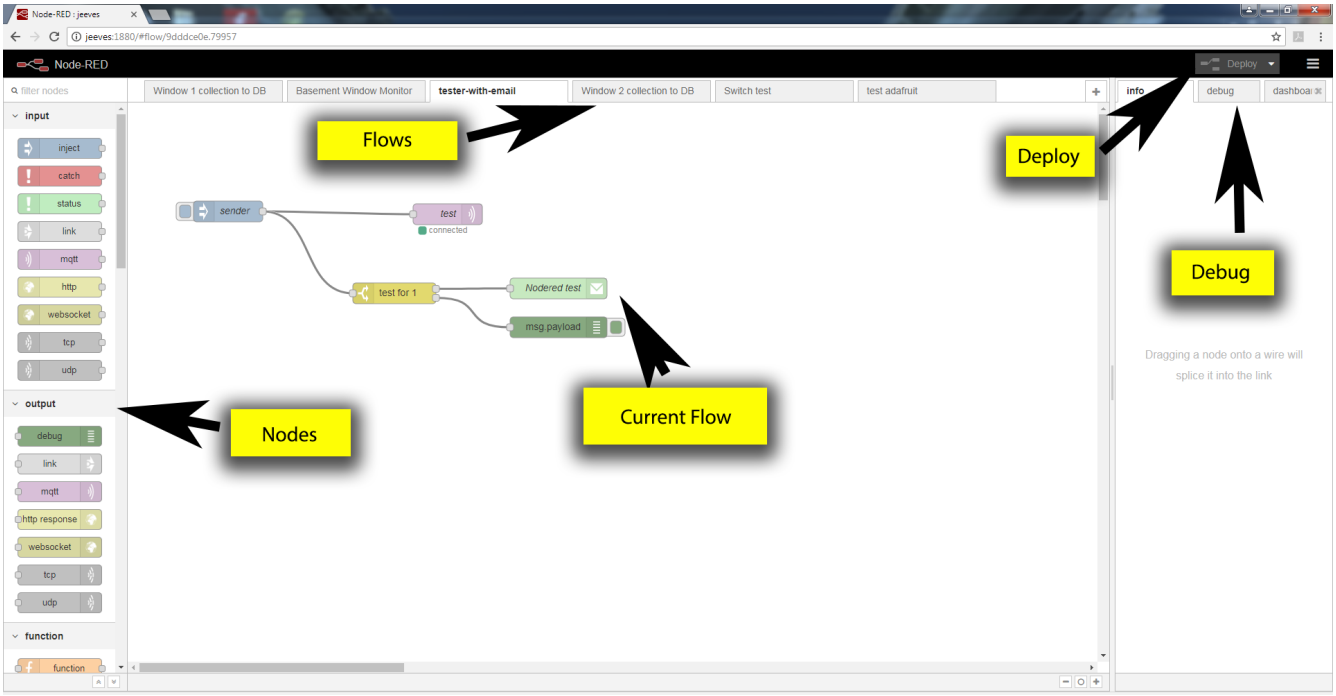


Node MCU Pinout – ESP8266 12E




-
+
A
B
C
-
+
D


sketch_jan03a | Arduino 1.8.0
File Edit Sketch Tools Help
Verify
Upload
New
Save
Open
Serial Monitor
sketch_jan03a
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
Coding Area
1
Arduino/Genuino Uno on COM19



Last updated 2018-05-20 18:32:13 CDT