DIY IOT Lab — Penguicon 2017

Author Dave Putz <dwputz@gmail.com>
Author Connie Sieh <cjsieh@gmail.com>

Your Station Name

DIY IOT Lab Outline

ESP8266 Arduino IDE , sensor, Data Broker and Web Server Basics

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 to a MQTT Broker

Lab 4
Program the ESP8266 to publish DHT11 Temperature sensor data to a MQTT Broker

Lab 5
Program the ESP8266 to be a Web Server

Lab 6
Program the ESP8266 to be a Access Point and a Web Server

Lab 7
Program the ESP8266 to display DHT11 sensor data via a local web server as a Access Point

Data Flow with Node-RED

Lab 8
Familiarize yourself with Node-RED

Lab 9
Setup ESP8266 to publish to MQTT Broker, and configure a Node-RED flow to subscribe to MQTT and send email

ESP8266 No/Little Programming Frameworks

Lab 10
Upload and configure ESPeasy to send DHT11 sensor data to MQTT Broker

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

1. Log into the lab system with
   - user: arduino
   - password: arduino
2. Plug in the ESP8266 using the USB cable
3. Click on the Arduino IDE icon to bring up the IDE, might be under
   - *Activities→Show Programs*
4. Click on *Tools→Board* and pick
   - “NodeMCU 1.0 (ESP 12-E Module)”
5. Click on Tools and make sure
   - "Upload Speed" is set to
     - 115200
   - "Port" is set to
     - /dev/ttyUSB0 or
     - /dev/ttyUSB1
6. Click on *File→Examples→ESP8266→Blink* This should bring up a code window with the “Blink” program loaded.
7. Take a look at this small program(sketch). Note that it has two main functions . All ESP8266 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
8. 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.
9. Edit the program to change the timing of the blinks. This will be done with the two lines that currently read
   - change it to whatever you like.

```
delay(1000);
```

10. Compile and upload the modified program. TIP: Can just click on the "upload arrow", and the IDE will compile before uploading
11. 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*
2. uncomment(remove the //) on the
    - define line for DHTTYPE DHT11

```
//#define DHTTYPE DHT11
```

- to

```
#define DHTTYPE DHT11
```

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

```
#define DHTTYPE DHT22
```

- to

```
//#define DHTTYPE DHT22
```

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

4. change

```
#define DHTPIN 2
```

- to

```
#define DHTPIN D5
```

Note The DHT11 sensor is connected to GPIO(General Purpuse IO) pin D5

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

```
Serial.begin(9600);
```

- to

```
Serial.begin(115200);
```

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* to save this version <<<

LAB 3 :: MQTT

1. 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 local MQTT Broker called Mosquitto .
3. Configure the ESP8266 Wifi to use the local lab network. Change the program at lines 24 and 25
    - WLAN_SSID: iotlab (SSID of our created lab network)
    - 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. Change line 29 and 31
    - AIO_SERVER
        - Use your station 192.168.x.x ipaddress
            - “ip addr” to get ip addresses of your station in a terminal window
    - AIO_USERNAME
        - User your station name
    - AIO_KEY (empty)
5. Change the name of your MQTT topic, which is done in the call to
    - Adafruit_MQTT_Publish(). Line 48
        - Change the name to “temp”
        - Change the literal from “feeds/photocell” to “feeds/temp”

```
Adafruit_MQTT_Publish photocell = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/photocell");
```

- to

```
Adafruit_MQTT_Publish temp = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/temp");
```

1. Change line 107 to call "temp.publish" instead of "photocell.publish" since we changed the name to temp earlier .

```
if (! photocell.publish(x++)) {
```

- to

```
if (! temp.publish(x++)) {
```

1. Put in a 15-second delay between each publishing
2. Click on *File→Save As* to save this version
3. Compile and upload the sketch(program)
4. Keep the serial monitor window open so you can see any displayed messages or errors
5. You should now be publishing the resukt of x++ to a *mosquitto MQTT broker* on your lab station
6. You can check this by
    - Switch to the “iotlab” network by going to “Network Manager” Icon at top right of screen and selecting
        - SSID “iotlab”
        - PASS “iotlabpass”
    - open a terminal window on the system and entering the command

```
mosquitto_sub -h localhost -v -u stationxx -P stationxxpass-t '#'
```

- where xx is your station number

NOTE:

```
-h localhost  -- <name/ipaddress of mqtt broker system>
-v            -- <verbose>
-t '#'        -- <topic> '#' means all topics
```

LAB 4 :: MQTT & DHT11

Add/Change DHT code to MQTT code from LAB 3

1. Add in the necessary DHT code. You can cut & paste from the DHT example.
    - Definitions

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

- Initialization of the DHT
    - Add to setup(); the following code

```
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

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

- Change the temp.publish line from "x++" to "f" since the DHT11 temp is in "f"
- from

```
if (! photocell.publish(x++)) {
```

- to

```
if (! temp.publish(f)) {
```

2. Compile and upload the sketch. Check your mosquitto_sub screen from Lab 3 to see that the sequential numbers have changed to temperatures.
3. Click on *File→Save As* to save this version

LAB 5 :: 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 “iotlab”
   - PASS “iotlabpass”
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: iotlab
   - PASS: iotlabpass
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 6 :: ESP8266 as a 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
        - ssid (use your station name)
        - password (your choice)
2. Insert
    - Since the default IPADDRESS is 192.168.4.1 you will need to change that since there are many EPS8266 in the lab.
    - where xx is replaced by your station number , just to make it unique

```
IPAddress apIP(192,168,xx,1);
WiFi.mode(WIFI_AP_STA);
WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
```

- before

```
WiFi.softAP(ssid, password);
```

- It should now look like

```
IPAddress apIP(192,168,xx,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* to save this version <<<

LAB 7 :: 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 4.

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 or lab station 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.

LAB 8 :: Familiarize Yourself with Node-RED

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

1. Open a browser on your lab system.
2. Enter “localhost:1880” in the URL bar. Note that 1880 is the default port for Node-RED, and can be changed in the settings.js file.
3. 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.
4. Click and drag an “inject” input node onto the design screen.
5. 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”.
6. Click and drag a “debug” output node onto the design screen.
7. Change its name to “Test”.
8. 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.
9. Click on the “Deploy” button on the top right. Note that in Node-RED no changes take effect until they are deployed.
10. Click on the debug tab on the right side to show the debug screen.
11. 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 9 :: 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.
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 design 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. Since your mosquitto broker and Node-RED are running on the same system, enter "localhost:1883" in the Server box.
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 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 is located at http://www.letscontrolit.com/downloads/ESPEasy_R120.zip . This has already been downloaded and unziped to your stations. I included the Linux script to install this firmware to the esp8266 as mentioned at https://www.letscontrolit.com/wiki/index.php/Tutorial_ESPEasy_Firmware_Upload .

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
    - 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
    - 4 ESPEasy_R147_4096.bin

```
cd Downloads/arduinoimages/esp8266-frameworks/espeasy
./espeasyflash.sh
```

6. After flash is done
    - Start Arduino Serial Console
        - **AFTER** all of the flash clearing is done and your see
            - *wdt reset*
                - Reboot esp8266 via reset button
7. Configure ESPeasy running on your esp8266 via ESP Easy web interface

a. 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

b. 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

c. 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 miliseconds
        - 500
      - 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

d. Configure Tools settings

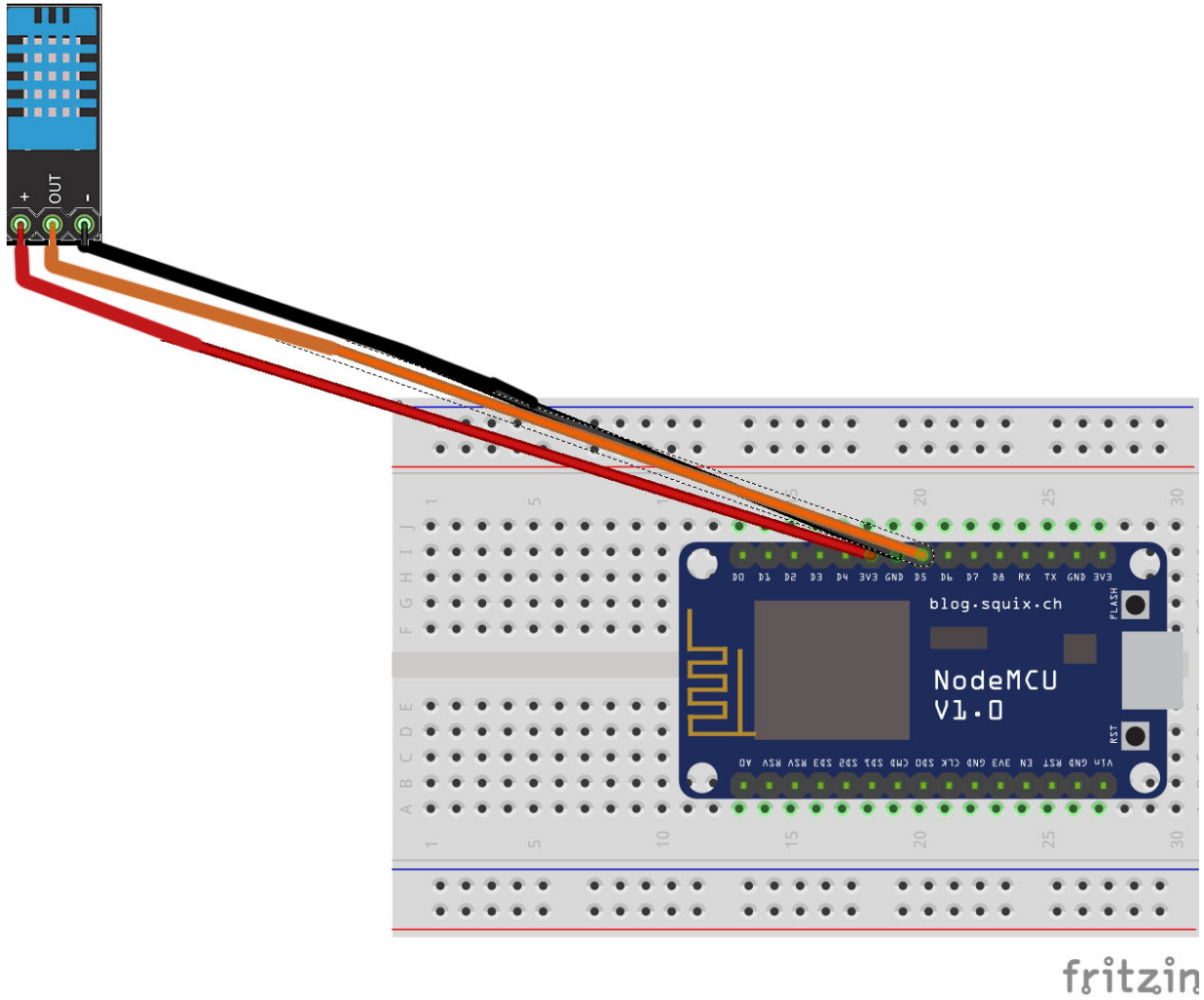
- https://www.letscontrolit.com/wiki/index.php/ESPEasy#Tools_page
- Click on the 'Tools" tab
  - System/Advanced
    - Look it over — no changes needed for lab
  - Settings/Save
    - Can save setting for later reload via Settings/Load
    - Good idea so as to not have to enter by hand again
  - Firmware 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

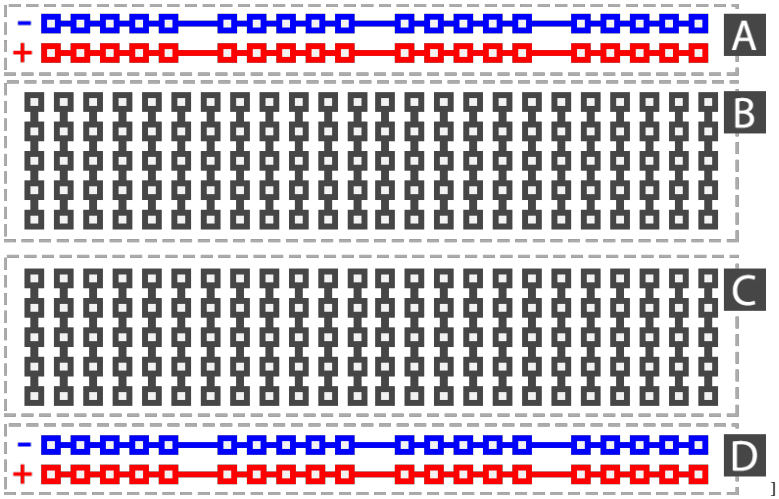
LAB SETUP :: These have already been done

- SOFTWARE
  - The Arduino IDE has been installed on the lab system.
    - This can be found online at https://www.arduino.cc/en/main/software
      - Version 1.8.1 was installed
  - The Arduino IDE has been 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://esp8266.github.io/Arduino/versions/2.3.0/doc/installing.html
  - Arduino libraries and ESP8266 specific libraries have been installed
    - In the Arduino IDE, Click on *Sketch→Include Libraries→Manage Libraries*
      - http://esp8266.github.io/Arduino/versions/2.3.0/doc/libraries.html
      - The following libraries have been installed:
        - ESP8266
        - Adafruit MQTT Library
        - DHT Sensor library by Adafruit
        - Adafruit Unified Sensor by Adafruit (dependency of DHT Sensor)
    - ESP8266 Arduino reference doc
      - http://esp8266.github.io/Arduino/versions/2.3.0/
  - The mosquitto MQTT broker has been installed. This can be found online at https://mosquitto.org/download/ or by using "apt-get mosquitto".
  - The Node-RED flow manager software has been installed. Instructions on how to do this can be found at https://nodered.org/docs/getting-started/installation .
- HARDWARE
  - A NodeMCU ESP8266 development board has been mounted on a breadboardand 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
    - Frizing diagram of the ESP8226 lab hardware

NodeMCU/DHT11 for DIY IoT Lab



- Diagram of the wiring of the Solderless Breadboard



]

Last updated 2017-04-25 18:13:32 CDT