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→01. Basics→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 for DHTPIN
        - change 2 to D5

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

5. Compile and upload the sketch(program).
6. 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.
7. 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.

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 -t '#'
```

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 function that reads the DHT sensor

```
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);
```

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

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

- to

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

2. 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

```
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

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

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

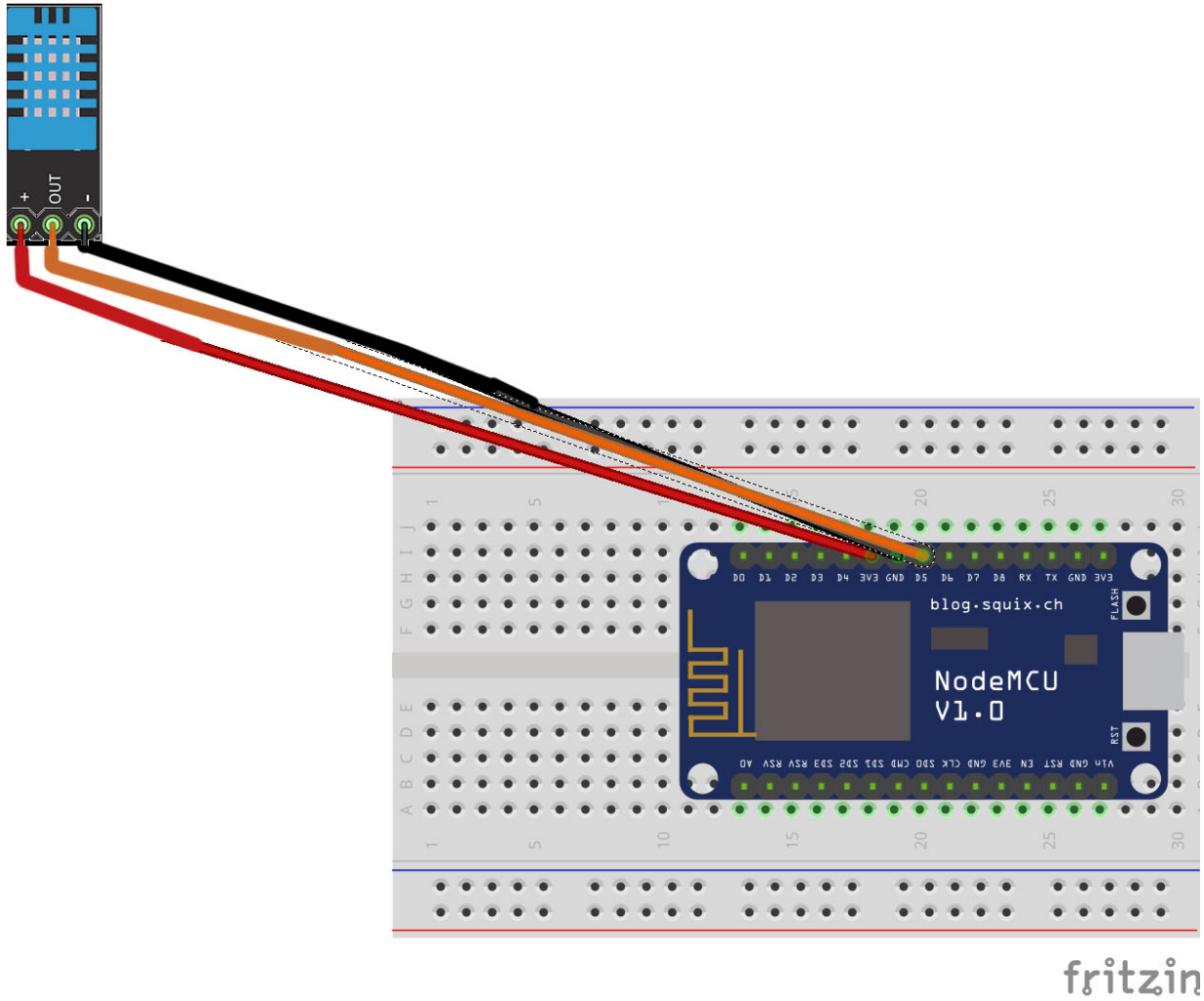
```
server.on("/", handleRoot);
server.on("/temp", handleTemp);
```

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.

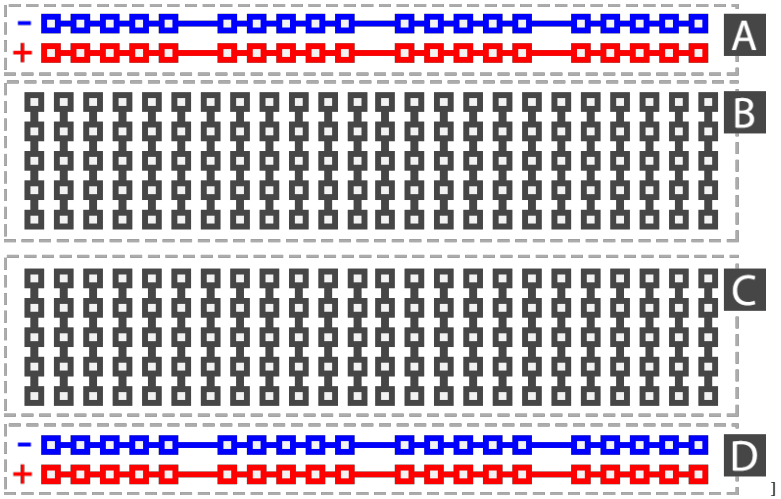
LAB SETUP :: These have already been done

- 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 2017-04-19 08:25:53 CDT