2018

Lab 5 Security

IOT IT515R CLASS

CJ CORNEL

Contents

I. Objective

Building off of Lab 4, code and create a website GUI for opening and closing the garage door and a circuit to act accordingly. The garage opener will not override what the Reed switch reads.

In addition, security must be checked over to ensure a safe, reliable network.

II. Materials

Adding on from previous materials are 1 Wemos, 2 headers, 3 male to male wires, and 1 D1 Relay Shield.

- Raspberry Pi
- 4 Wemos D1 Mini ESP-8266EX Board
- 1 Distance Sensor HC-SR04
- 1 Reed Switch
- 3 LEDs (Red, Green, and Yellow)
- 3 220 Ω Resisters
- 8 headers
- 13 Male to male wires
- 2 alligator cables
- 1 D1 Relay Shield

III. Procedure

This procedure is assuming that all previous labs have been completed beforehand.

A. Understanding How it Works

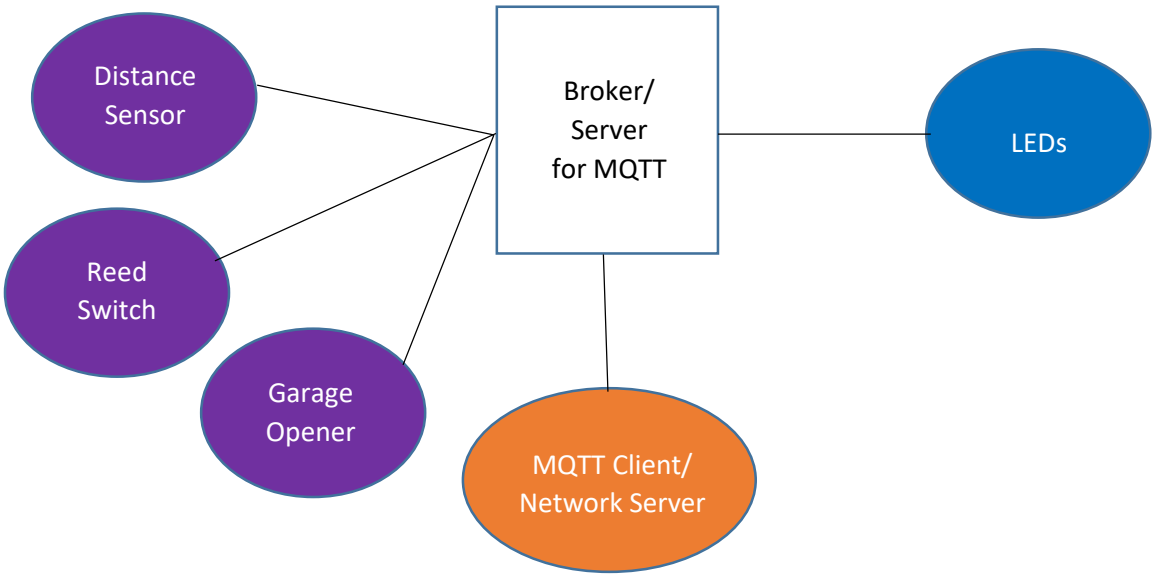

Figure 1: Adding the Garage Opener.

1. Garage Opener – Tracks if the garage has been opened or closed via the website
2. MQTT Client – In addition to its pre-existing abilities, it will also host a website for a Garage Door Opener GUI.

B. Channels – Subscribers & Publishers

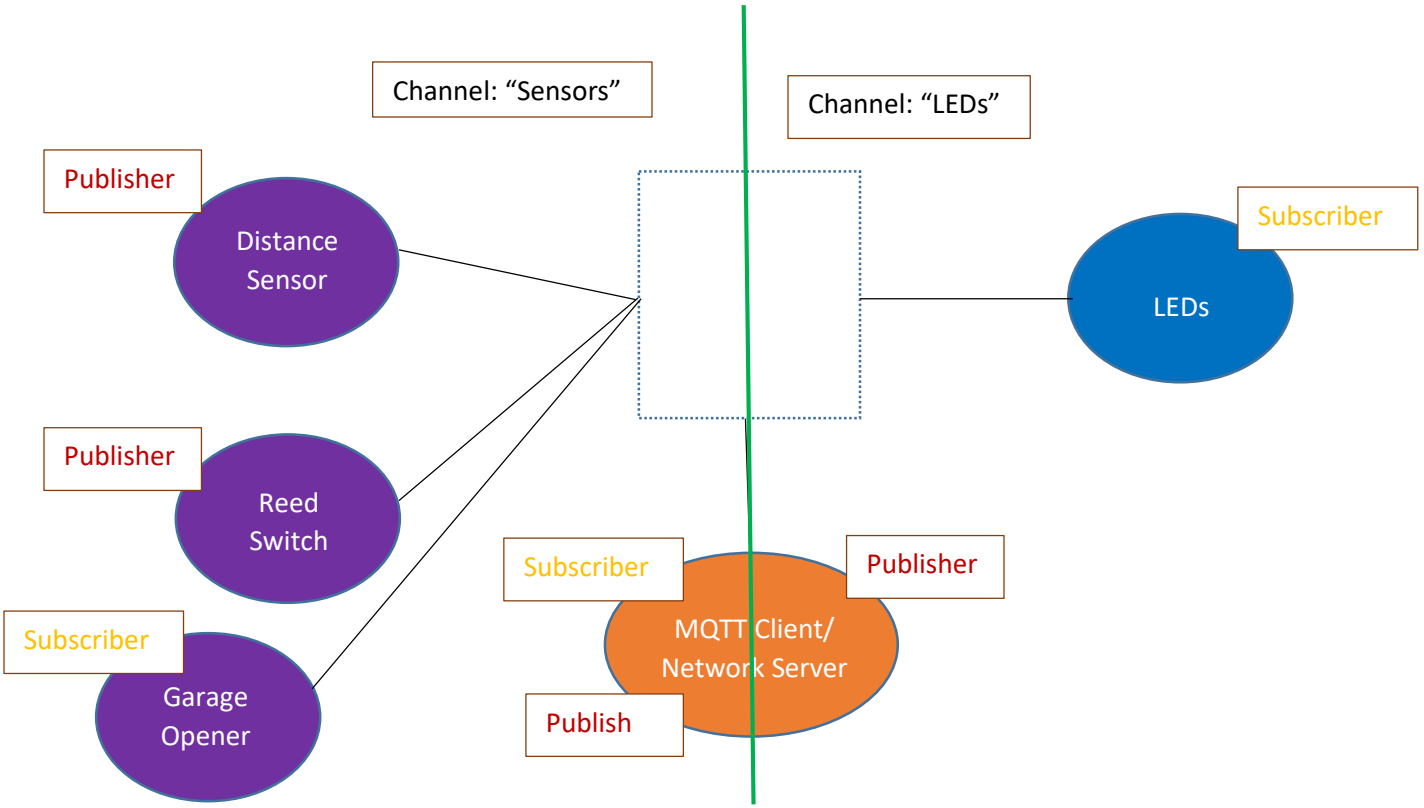

Figure 2: The Garage Opener Sensor has been added to the Channel “Sensors” as a subscriber, so as to discern if there has been a request to open or close the garage door. The MQTT Client is now also a publisher on “Sensors” because we are using a website to tell the garage door to open or close.

B. MQTT Broker Client & Server

1. Create a website to close or open a garage door
    a. Publish the option chosen to the “Sensors” channel
    b. Publish the result to LEDs
    c. Ensure that the website only properly opens to those on the home network
2. Website
    a. Restrict access to home subnet that the website is on

C. Garage Opener

1. Connect a Weemos to a D1 Relay Shield to simulate a garage door opener telling the garage door to open or close.

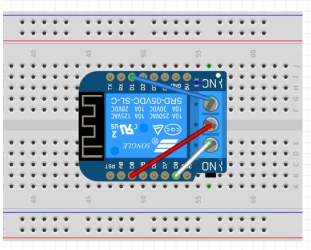

Figure 3: Garage Opener Configuration – The Shield is attached to the top of the Weemos. I attached wires to the NC and NO connections to read the output (essentially for testing purposes)

2. Setup the connection with the MQTT server
3. Create a function for what to do when connected to the server and a payload is received
    a. According to the payload ("open" or "close") the state of the Ground (Com) will be high or low
        i. If Com is high, the Normally Closed (NC) port is high
        ii. If Com is low, the Normally Open (NO) port is high

IV. Links

- Github: https://github.com/Sh0rtyTheCircuit/Lab5_Security
- Blog: https://ideatethecreate.blogspot.com/2018/11/iot-lab-5-security.html

V. Thought Questions

1. How did you secure devices that are not inherently secure devices? How does layering security approaches help?

Restricting access to the website to the home subnet that the website is on is a good way to help secure inherently insecure devices. Layering security helps in the fact that more access points are guarded and make it harder to get in an otherwise open access point. Further suggestions include using 2 factor authentication (e.g. You can only access the program if using your phone's MAC address, etc.), changing the home wifi into 2 – 1 for guests and the other for family only, and putting the equipment in a secure location (e.g. put in a lock box, far from reach, etc.).

2. What was the biggest challenge you overcame in this lab?

Understanding how the garage opener is not to be confused with the one using the reed switch. The relay should not override what the reed switch says and should not be considered when determining what LEDs should be on or off.

3. Please estimate the total time you spent on this lab and report.

10 hours

VI. Sources

- https://www.instructables.com/id/Home-Automation-How-to-Add-Relays-to-Arduino/
- https://hobbycomponents.com/shields/865-wemos-v2-relay-shield
- https://gist.github.com/dergachev/7028596
- http://fritzing.org/home/
- Andrew Thomas, friend, M.S. candidate

VII. Appendix A

A. MQTT_Client.py

```
import paho.mqtt.client as mqtt

import time

import socket

import select

import sys

import threading


#Channel Topic

sensors = "Sensors"

LED = "LED"

GarageOpener = "GarageOpener"


#ip of localhost

mqtt_broker= "192.168.43.40"

mqtt_port = 1883


Accept_IP = "192.168.43."


#Define functions


def msg_rcv(sensors, user_data, msg):   #Interpret Msgs (Loops)

    print "Payload is " + str(msg.payload)

    if (str(msg.payload) == "close"):

        LED_color == "close"

    elif (str(msg.payload) == "open"):

        LED_color == "open"
```

```
        elif (str(msg.payload) == "on"):
                LED_color = "on"
        elif (str(msg.payload) == "off"):
                LED_color = "off"
        elif (str(msg.payload) == "red"): # or str(msg.payload) == "green" or str(msg.payload) == "yellow"):
                LED_color = "red"
#               LED_color = str(msg.payload)
        elif (str(msg.payload) == "yellow"):
                LED_color = "yellow"
        else:
                LED_color = "green"
        sensors.publish (LED, payload = LED_color, qos=0, retain=False) #(channel, msg to publish)
        print (LED_color)

def run_broker(client, user_data, flags, rc):                  #Subscribe to topics (Once)
        print "In the broker function"
        client.subscribe(sensors)                       #Listen to the Sensors channel
        print "Subscribed to " + topic
        client.subscribe(GarageOpener)

client = mqtt.Client()

#When message is received, run msg_rcv function
client.on_message = msg_rcv

#When connected to Broker, run run_broker function
client.on_connect = run_broker

#Begin connection to MQTT Broker
```

```
client.connect(mqtt_broker,mqtt_port)
print "connection to broker started"

## CREATE WEBSERVER ##

HOST, PORT = '', 9898

#httpd = BaseHTTPServer.HTTPServer((HOST, PORT), SimpleHTTPServer.SimpleHTTPRequestHandler)
#httpd.socket = ssl.wrap_socket (httpd.socket, keyfile= './key.pem', certfile='./server.pem', server_side=True)
#httpd.serve_forever()

listener = socket.socket(socket.AF_INET, socket.SOCK_STREAM)    #socket setup
listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
listener.bind((HOST,PORT))                          #Where the server is now located
listener.listen(1)                                  #Listen for this many clients

print 'Listener is set up'
stopper = threading.Event()                         #Handler for starting and stopping the thread

while True:
    client.loop()
    client_connection, client_address = listener.accept()   #Accept connection from user

    request = client_connection.recv(1024)              #Waiting for GET REQUEST(refresh/load website). Request is new request now.
    print request                                   #Request = whatever is Posted when btns are pressed
    check_status = request[0:13]                        #Look at URL in the GET REQUEST
    print check_status
```

```
    if check_status.find("close")>0:                #Look for "Green" in request. If not found, will return -
1 which breaks the code
        stopper.set()
        print ("closed")
        client.publish(GarageOpener,payload ="close")
    elif check_status.find("open")>0:
        stopper.set()
        print ("open")
        client.publish(GarageOpener,payload ="open")
    if client_address [0][0:11] == Accept_IP:          #If the client IP is in the same network
        disp_body = """\
<html>
    <title> Choose Wisely </title>
    <body>

        <form action="/close" method= "post">
            <button>Close</button>
        </form>
                                <form action="/open" method= "post">
            <button> Open </button>
        </form>
    <br>
    </body>

</html>
    """
    else:                                  #If the client IP is in a different network, do not show the page
        disp_body = """\
<html>
```

```
        <title> Choose Wisely </title>
        <body>
            <h1> Nice Try </h2>
        <br>
        </body>

</html> """
        display = """\
HTTP/1.1 302 OK
Content-Type: text/html
Connection: close \n
"""
        print (display)
        client_connection.sendall(display + disp_body)
        client_connection.close()
        print ("sent")
#Predefined functions
client.loop_forever()   #Client will keep itself alive
client.disconnect()     #Disconnect before dying (cntrl C or kill)
```

B. GarageOpener.ino

```
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

// #### LED Pin Setup #### //
int NO = D8;
int COM = D5;
```

```
int NC = D1;


// #### MQTT Server connection Setup - Raspberry Pi Broker #### //

char* mqtt_server = "192.168.43.40";

int mqtt_port = 1883;

char* topic = "GarageOpener";

char* state = " ";


WiFiClient Wifi;          //Setup Wifi object

PubSubClient client(Wifi);  //Object that gives you all the MQTT functionality, access objects in PubSubClient Library


// ##### Wifi Connection Setup #### //

char WifiName[] = "Verizon-SM-G935V";          //SSID

char Password[] = "password";


void Msg_rcv(char* topic, byte* payload, unsigned int length){     //Unsigned int = Positive numbers (more range)

 Serial.print ("payload is ");

 Serial.println((char)payload[0]);

 if ((char) payload[0] == 'o'){

  if ((char) payload[1] == 'p'){

   digitalWrite(COM,LOW);

   Serial.println("COM port is LOW");

  }

  }

 if ((char) payload[0] == 'c'){

  digitalWrite(COM,HIGH);

  Serial.println("COM port is HIGH");
```

```
  }
}

void setup() {
  // put your setup code here, to run once:
  pinMode (COM, OUTPUT);              //INPUT_PULLUP, Pin is high (3.3V) until low
  pinMode (NO, INPUT);
  pinMode (NC, INPUT);
  digitalWrite(COM,LOW);
  Serial.begin(9600);       //Begin serial monitor

  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(Msg_rcv);                //Send payload to function (Msg_rcv)

  // ### Begin Connection to Wifi ### //
  WiFi.begin(WifiName,Password);
  while (WiFi.status() !=WL_CONNECTED){         //If not connected to Wifi, delay until connected
    delay (2000);
    Serial.println("Finding a Connection...");
  }

  Serial.println("Connection Started");         //Begin Connection to Wifi
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());             //IP assigned to Server by host wifi

  while(!client.connect("Garage_Opener")){         //LED_board is name of Wemos/arduino connected to
code. Waiting to connect to Broker.
    Serial.println("Finding a Connection...");
  }
```

```
  client.subscribe(topic);
}

void loop() {
  // put your main code here, to run repeatedly:
  client.loop();
  if(!client.connected()){
    client.connect("GarageOpener");
  }
   if (digitalRead(NC) == HIGH){
    state = "close";
    Serial.println(state);
  }
  else if (digitalRead(NC) == LOW){
    state = "open";
    Serial.println(state);

  }
  //client.publish(topic, state);
  Serial.println(state);
  delay(1000);
}
```