Final Project – Arduino Alarm

Online Links

This lab is available on my personal website at: http://aaronnelson95.com/IT441Final.php
The code is also available on my GitHub repository at https://github.com/AaronNelson95/IT441

Objective

The purpose of this project is to learn how to combine Arduino devices to create an alarm clock supplement. Three Arduino devices will be subscribed to an Adafruit IO MQTT feed. This feed will be turned on by an alarm set in IFTTT (through a Date & Time service). Once activated and a value is sent through the feed, a device connected to a light strip will turn on to provide light in a room. A second device will perform an HTTP Get request to a weather API. The JSON returned will then be parsed and displayed on an OLED display screen. The alarm can then be turned off through Google Assistant or any other IFTTT connected service, or by opening the bedroom door when a magnetic reed switch becomes disconnected. This lab will help one learn:

- How to use IFTTT to design applets that publish data to an Adafruit IO feed
- How to use the FastLED library to activate LED lights on a light strip
- How to connect to an API and receive its JSON information
- How to parse through JSON in Arduno
- How to display text on an OLED screen
- How to develop a practical product that can meet your needs by using Arduino devices

Materials

To complete this lab, you will need the following materials:

- A computer with the Arduino IDE
- Three Wemos D1 mini microcontrollers
- Three Micro USB power cords
- Three breadboards
- 7 Male-Male Jumper Wires
- A WS2812B LED Light Strip
- A SSD1306 Wemos OLED Shield (screen size 64x48)
- A Magnetic Reed Switch (such as the Aleph DC-1561)

The WS2812B LED light Strip

The OLED Shield

The Aleph DC-1561 Magnetic Reed Switch

References

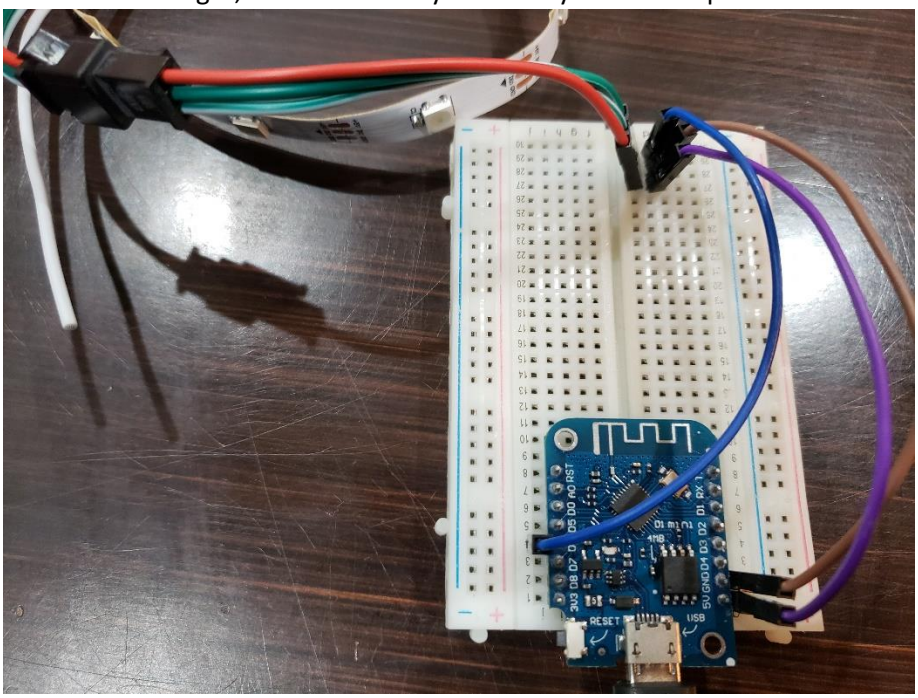
The following links may be helpful throughout this project:

- https://www.norwegiancreations.com/2018/01/programming-digital-rgb-led-strips-with-arduino-and-the-fastled-library/ - How to set up a LED light strip with Arduino
- https://www.youtube.com/watch?v=9KI36GTgwuQ – A more advanced tutorial on using a LED light strip with MQTT and Home Assistant
- https://create.arduino.cc/projecthub/officine/getting-weather-data-655720 - A tutorial on obtaining weather data from an online API and how to parse JSON data
- https://techtutorialsx.com/2016/07/17/esp8266-http-get-requests/ - A tutorial on using HTTP Get requests
- https://openweathermap.org/current#format – An online weather API that works across HTTP
- https://arduinojson.org/v6/example/http-client/ - An example on how to read and parse JSON data

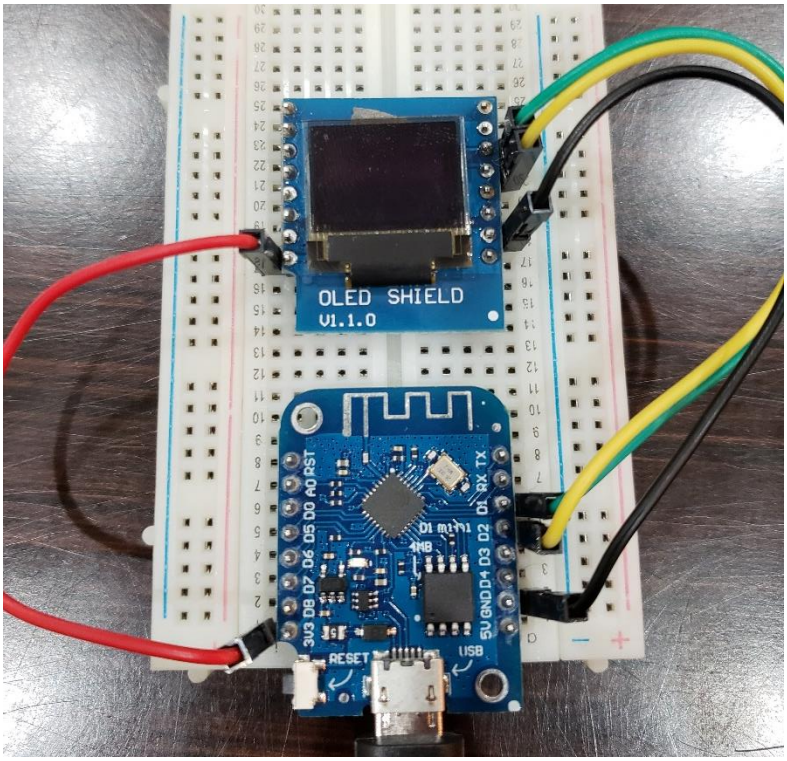
- https://arduinojson.org/v6/assistant/ - An essential tool for allocating memory and understanding how JSON is read with Arduino
- The Adafruit IO Feed Example at File -> Examples -> Adafruit MQTT Library -> mqtt_esp8266_callback
- The OLED screen example at File -> Examples -> Adafruit SSD1306 Wemos Mini OLED -> ssd1306_64x48_i2c
- https://ifttt.com/ - Used to combine services that will post a value to the Adafruit feed to trigger the alarm
- https://io.adafruit.com/ - An online MQTT Broker that can work with IFTTT and Arduino devices

Procedures

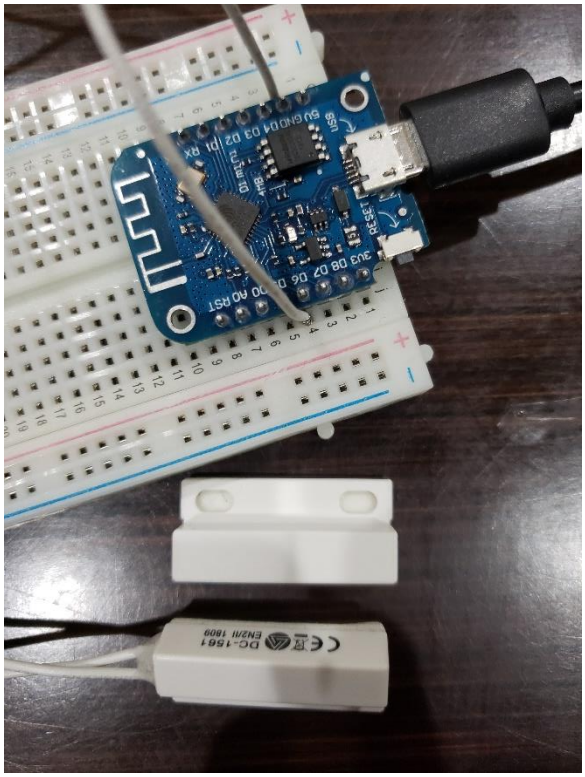
1. Design documents to help you understand the logical flow of the system.
    a. A system diagram will help you to understand how each device communicates with each other. Basically, when a Date & Time based applet runs in IFTTT, a value of "1" is posted to an Adafruit IO MQTT feed. The three Arduino devices are subscribed to this feed and will turn "on" when it receives this value. The LED light strip will turn on its lights. The weather screen device will make an HTTP Get request from an online API and will display its results on an OLED screen. The magnetic reed switch will now check for a door being opened. If the switch's door is opened, Google Assistant is spoken to, or a different IFTTT applet is activated, a value of "0" is posted to the feed and all devices will turn "off". You can view an example system diagram in the appendix of this report.
    b. Develop system flowcharts for each of the three Arduino devices. This should show an idea of how to program the microcontrollers, as they step through the startup() and loop() functions. You should also plan out how you would like to publish or subscribe to MQTT information. This will start out basic, but as you are working through the code you will have a better understanding of how the Arduino script should handle data. You can view example flow charts in the appendix.
2. Setup the three Arduino devices. The circuit diagrams for these three devices are shown in the appendix.
    a. Connect one Wemos board to the LED Light Strip. Wire colors depend on your model, but generally, one of these wires is meant to handle the power, and should be connected to 5V. Another should be connected to Ground, and a third wire works as data, and should be hooked up to a random pin, such as D6. It may be tempting to line these up and use pin D4, but be aware that this pin also controls the onboard built-in LED light, which will always be on if you use this pin for data.



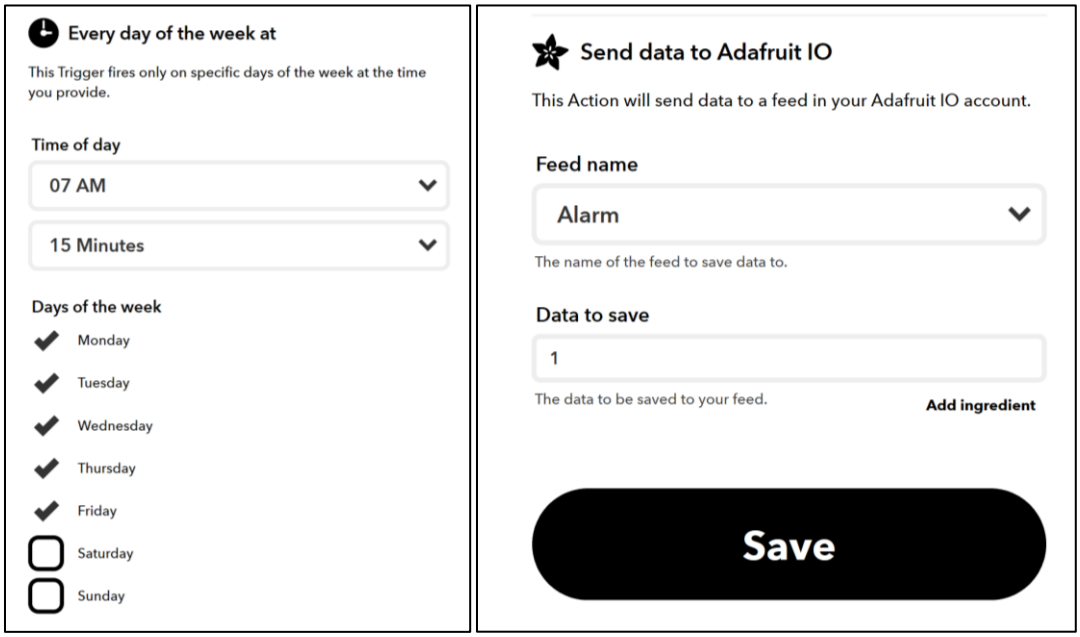
b. Connect another Wemos board to the OLED Shield. If soldering allows, you can just place the shield directly on top of the board. If it doesn't, however, you need to connect the 3V, Ground, D1 (which is SCL), D2 (which is SDA) pins.



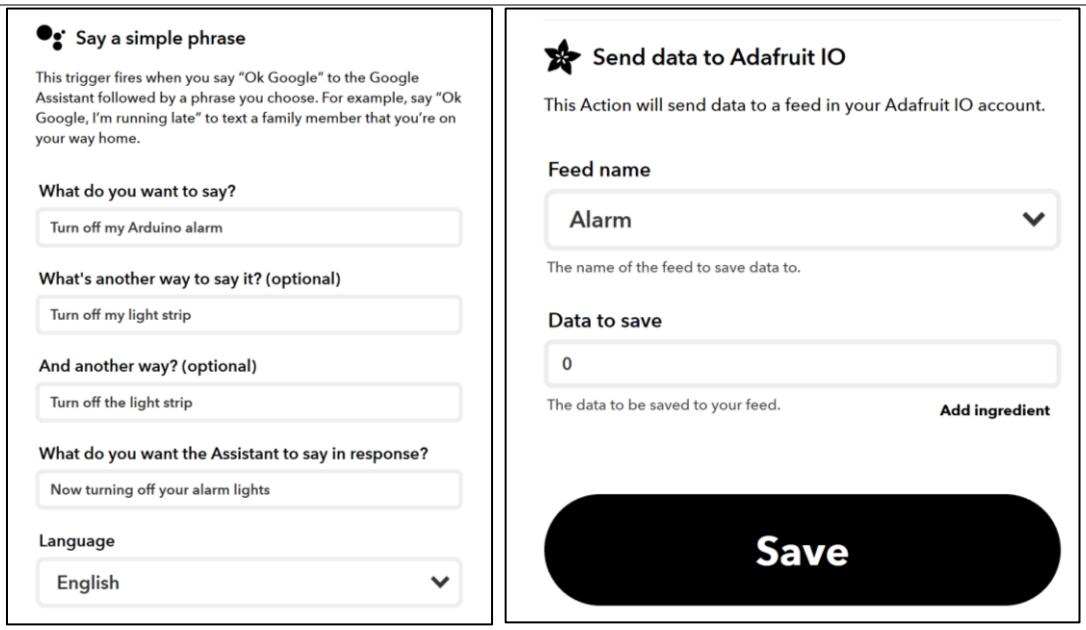
c. Connect a third board to the magnetic reed switch. One wire will be connected to a random pin, such as D6, and the other will be connected to GND. It does not matter which wire is connected to which pin.



3. Setup a connection between IFTTT and Adafruit IO on your computer or phone
    a. Create an account (if you haven't done so already) at https://io.adafruit.com. Once signed in, click the "IO" tab on the top of the page, then select the "Feeds" tab and click "view all"
    b. Click "Actions" and "Create a new Feed". This is the "topic" that the relay will subscribe to. Once created (name is "Alarm"), view that feed. Notice how the URL to this topic goes "<your username>/feeds/<topic name>". Later in the Arduino code, you will need to specify both of these parameters to access your information.
    c. Click the AIO Key tab on the top right of the page. Copy the "Active Key" and save it for later. This will be used to give permission for your Arduino device to access the Adafruit IO feed.
    d. Sign into IFTTT and go to https://ifttt.com/adafruit. Connect your Adafruit service to your IFTTT account. Now you can create and experiment with various IFTTT triggers to post a value to your Adafruit IO feed. For increased security, you can come up with a specific value that would cause the relay to be activated (and all others will be ignored). The "That" portion of your applet will work with Adafruit IO and send data to the feed name (given in step 4b), and the activating value you would like to send (such as a simple "1" to turn the alarm on).
        i. To simulate an alarm, create multiple "Date & Time" events. Pick "Every day of the week at" and specify the time and days for that alarm to run.



    e. You should also setup triggers to turn the alarm *off* by posting another value (such as "0") into the feed.
        i. You can trigger this same action with multiple, separate triggers, for example, you can use an IFTTT button (which is activated by a phone widget) to do this action.
        ii. You can also link IFTTT with Google Assistant and specify phrases you would like to say to trigger the action such as "Turn off my Arduino alarm" or "Turn off my light strip".
    f. Test that your triggers work. When they are activated, you should see the table populate with the value in your Adafruit IO Feed page.



4. In the Arduino IDE, install the libraries you will need to work with this project. All three devices will use the "Adafruit MQTT Library" by Adafruit. The light strip will also need the "FastLED" library by Daniel Garcia. The weather screen will use "Adafruit GFX Library" by Adafruit, "Adafruit SSD1306 Wemos Mini OLED" library by Adafruit + mcauser, and "ArduinoJson" library by Benoit Blanchon.
5. Program the LED light strip device. It may be helpful to first verify that you can connect to the Adafruit IO MQTT feed to receive data. Then you can work with the FastLED library to turn on lights. Although you may have many lights on the strip, use caution when powering the strip through your board! Trying to power all of the LEDs with bright lights may wear down your device. Instead, using 15-30 lights is generally safer unless your strip can connect to an external power source.
6. Program the weather screen device. Once again, ensure you can establish a connection to Adafruit IO. It may be helpful to split the program further and develop 3 separate functions for the device's function.
    a. First, see if you can use the ESP8266HTTPClient library to connect to an API and store the JSON into a variable. Generally, if you are using an online API service, you will need to obtain a key (available for free typically, but they work to limit the calls you make in a period of time). If using OpenWeather, create a free account and record your key value. Explore their documentation (https://openweathermap.org/current) to see examples of API calls, which simply look like typical URLs with additional parameters attached (such as "?zip=55555"), build the URL that returns the data you want, and test it in the web browser to see if you can view all of your information.
    b. Converting JSON data in Arduino is a little tricky and will require help from the ArduinoJson library. Examples of how to use this is in the above References section. You will likely need to go to https://arduinojson.org/v6/assistant/ to calculate your program's script. Here, you paste in the JSON code from your API (copied from the web browser). Lower in the code, there is a "Parsing program" section. Use the information here to build your Arduino program. You will need to find the size of the JSON (which it calculates for you), build an object with it, deserialize the data, and access the values you want to obtain. It may be tricky at first, but this site will help you a lot.
    c. Finally, display this information to the screen. It may help to play around with the example at File -> Examples -> Adafruit SSD1306 Wemos Mini OLED -> ssd1306_64x48_i2c to understand how this library works. Don't forget to run "display.display();" to actually commit the text and changes to the board.

7. Feel free to program any other devices you would like. For example, you can use the magnetic reed switch to act as another method to turn your alarm off (rather than Google Assistant and IFTTT). If you sleep with your door shut, you can know you're awake when the door becomes opened, which this switch can easily test. Try to program the logic of these devices to only activate when the alarm is already running. When it should be activated, simply publish the off value ("0") to the Adafruit feed. You can also use the echo distance sensor to detect when you are moving or walking around. Have fun exploring new and creative ways to confirm that you are awake.
8. Verify that the entire system works by manually submitting your "on" value to Adafruit. The light strip should turn on and the OLED screen should now display your weather data. Tweak your programs as you see fit. Ultimately, they should work like the system diagram designed earlier (provided in the appendix). Enjoy this practical and useful way to use Arduino devices to enhance your daily life!

Thought Questions

1. How well did the project solve your initial problem?

   A challenge I have always faced is with waking up on time. It is a problem I share with many other college students. I am often up too late in the night working on homework, and then I only have a few hours to sleep before I need to make it to my early class. In such a tired state, and combined with the darkness of my room, I have a hard time waking up, even to an alarm clock buzzing off! By the time I finally get moving (snoozing as many times as I can), I am too much in a rush to get out the door. Without having time to properly prepare, I am not aware of the day's current weather forecast, and I may be poorly dressed, unprepared for the elements. I wanted to build a device that could help me with both waking up because of dark conditions, and for knowing what I should wear for the rest of the day. Using Arduino devices, I was able to accomplish both of these challenges! When the alarm goes off, LED strip lights activate, providing me with enough illumination to see around my room. My other device works extremely fast, and provides me with a quick look at the weather conditions and the temperature at a glance. I also built this system independent of my audible alarm clock, so it can remain on and help discourage me from snoozing my clock. Testing this system out has worked pretty well. It has gone off in the mornings, within less than a minute of the time I had scheduled. I have found it very nice and convenient to have more light in my room, and the various methods I have incorporated to turn off the alarm require me to either move out of my bed or be cognitive enough to speak to my phone. I am proud of the way my project turned out, and I am thrilled that I was able to build something practical that can help me with my everyday needs.

2. What new technologies did you incorporate?

   I used many new technologies in this project. I had to borrow the devices and look around at quite a bit of examples to gain an understanding of how they work. One of these technologies is an LED light strip. It was interesting to me to learn how you can control and manage individual lights on a strand or use functions and math to make cool and creative patterns appear throughout the strip. I also learned about using Arduino to show text on a screen. It was quite a bit simpler than I imagined, but the screen also has so many additional graphical capabilities as well. I learned about API web requests and gathering data from them and storing it. I also learned about parsing JSON information with an Arduino board, which was actually more complicated than I anticipated. It was confusing to calculate how much space you needed to allocate for the object, but luckily there are great resources on the internet that can figure this out for you and how you can find the find the information you want.

3. What challenge did you face with this project?

   The biggest challenge I faced was in trying to obtain JSON data from an online API. I was able to use a HTTP Get request, which worked great, but I could never figure out how to properly do a HTTPS request. The main API I wanted to incorporate, which displayed all of the information I wanted accurately, only worked across HTTPS. I spent a couple of hours trying to figure out these requests. I followed online tutorials and examples, which suggested using certain libraries, or even directly copying a certificate directly to be used. However, I was unsuccessful. I ultimately decided to use a different weather API, which was more basic, but it worked across HTTP. When making a request to this type of page, I had no problems whatsoever. A future goal is to learn how to obtain the HTTPS information from the API I originally wanted to use.

   I also had a problem with the LED Light Strip and the used Arduino board attached to it. I noticed, when plugging it in, that it put off a burning smell. Afraid of any potential problems, I disconnected the board from power. As I did more research online, I discovered that even though the 5V is supposed to directly pass into the light strip, it is not enough power depending on the number of lights available, and it can cause overheating with the Arduino board. Using less lights on the strip (such as 15-30) and incorporating lower LED brightness can make your device much safer to use. It is recommended to use an external source to provide power to your LED lights, rather than through the Arduino board.

4. What are future goals with the project?

   I was able to get my project to a state that worked for my needs. However, there are other things that I would like to incorporate someday. As mentioned, I would like to incorporate a Get request with a HTTPS secured page. I just really liked the weather information from another API better, and it was much more accurate as well. However, I simply could not figure it out, but I will spend more time in trying to. I also would love to control more of the LED light strip lights. This will require attaching them to an external power source (which I will need to learn more about), but the FastLED library has so many fun, colorful possibilities I would like to play and experiment with. Eventually purchasing a larger OLED display screen (48x64 is just too small) can also open up a world of possibilities. With this device, you can perform all sorts of functions and calculations because of unlimited number of things that can be shown on a screen. It seems like it could be the most practical Arduino device I have used yet! I really enjoyed working with these new technologies, and I believe I can implement them in many more helpful ways in my life.

5. Estimate the time you spent on this lab and report.

   I spent around 10 ½ hours designing and building the Arduino devices for this lab. I stretched myself with learning new technologies and libraries in fun, creative ways. It took some time, but I was eventually able to solve or find alternative solutions to my problems. I spent around 6 hours working on the lab report, and additional time in preparing my final PowerPoint presentation. Overall, I invested around 18 hours in this project.

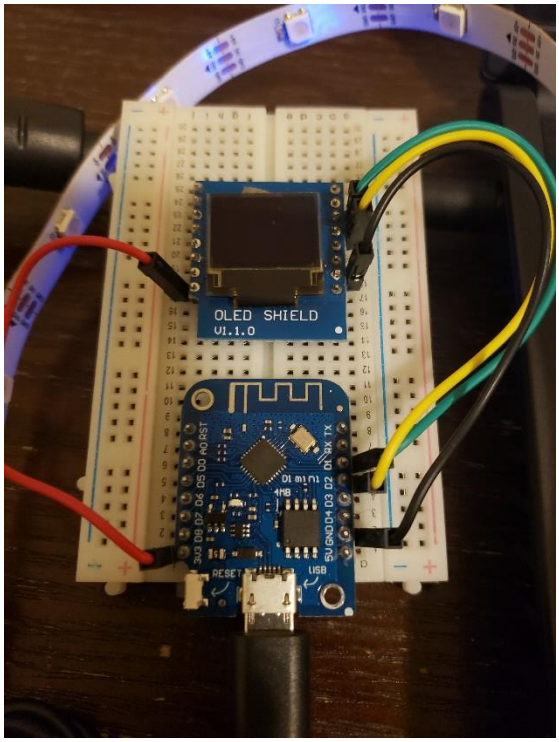
Certification of Work

I certify that this lab represents my own work. Any code that I used was basic public knowledge and the examples and resources I utilized are provided in the code comments.

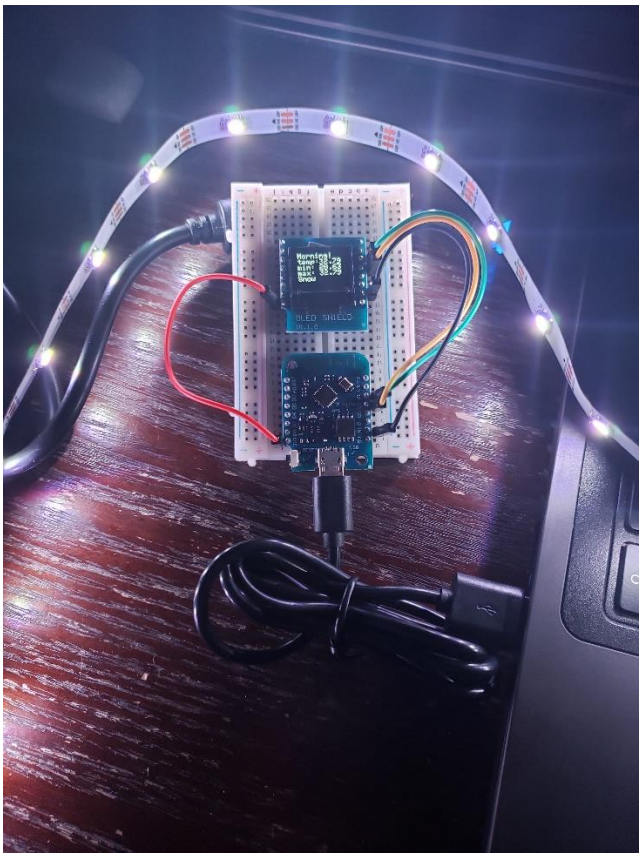
– Aaron Nelson

Appendix

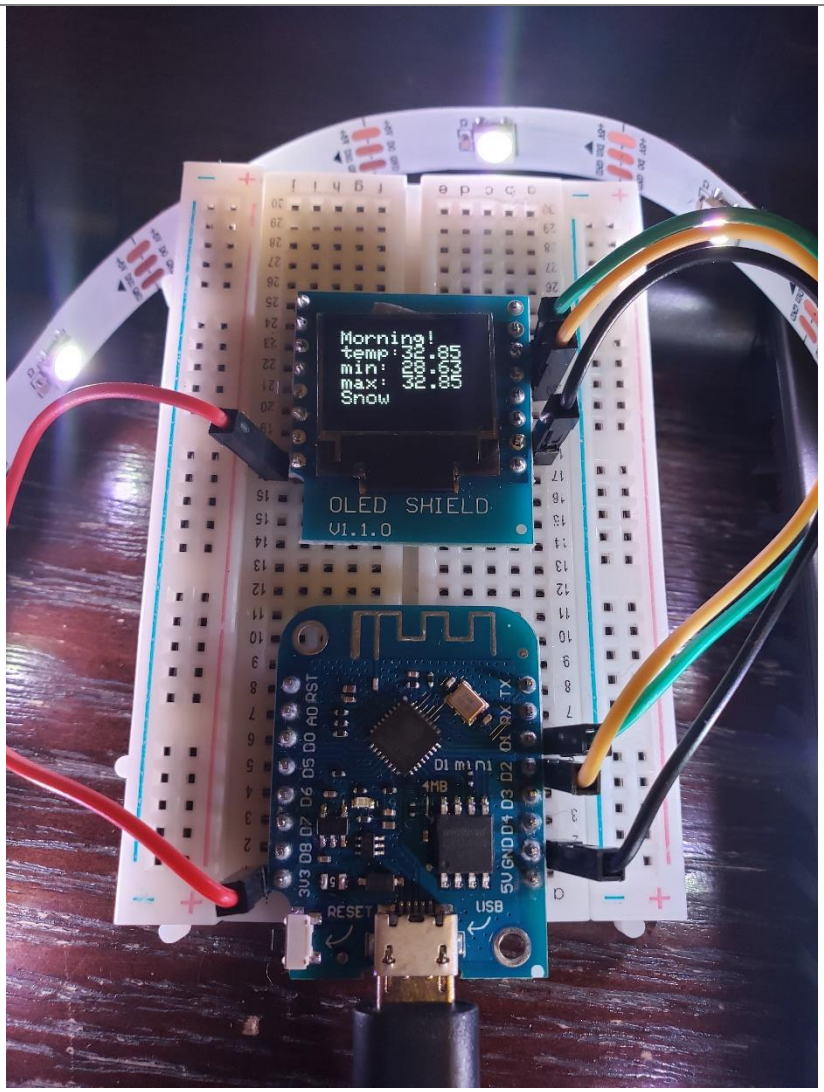
Images of Final Product



When the alarm is not set on, the screen display is blank and the light strip is off, appearing to be disconnected from power

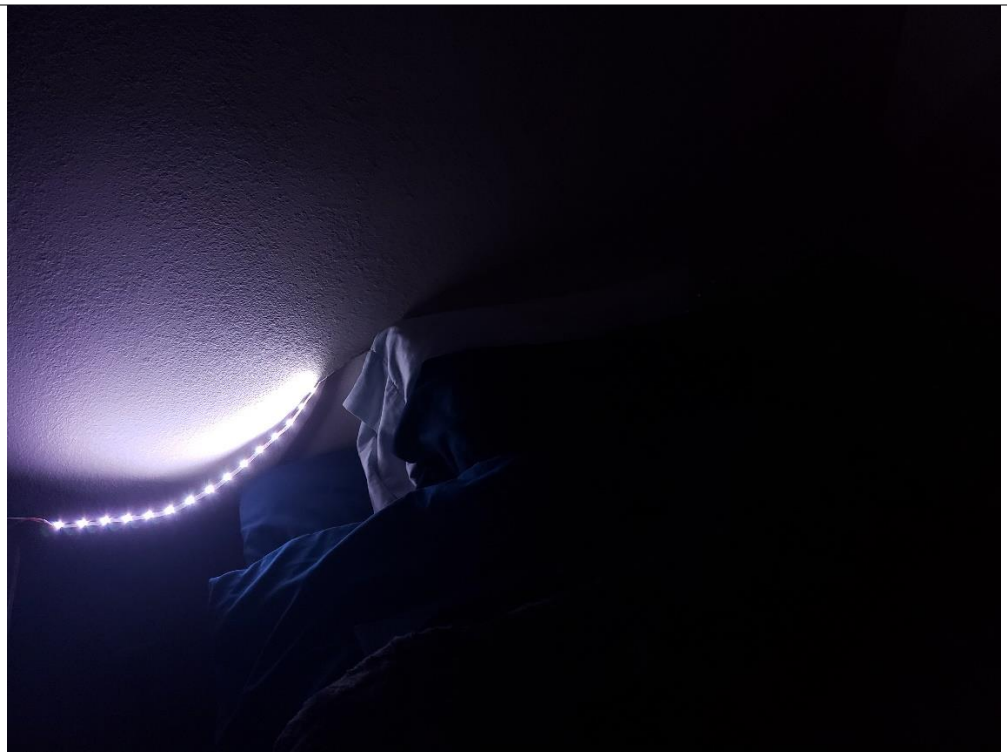


Once the feed receives a value of "1", turning the alarm on, the strip lights up and the screen shows weather information



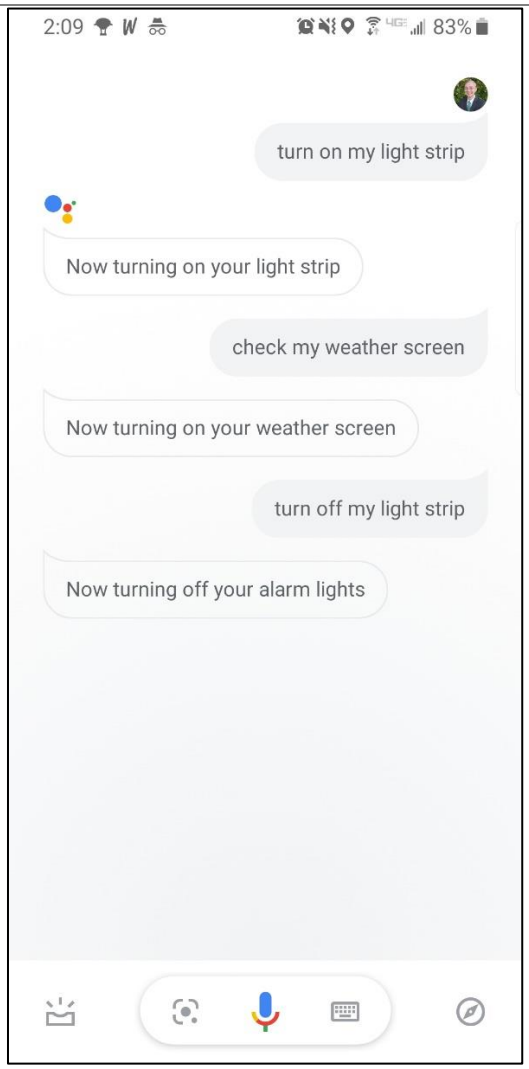
The OLED Screen shows information such as the temperature, the minimum temperature, the maximum, and the weather conditions

On a different day, it shows different values, verifying that the API request is working. This is the screen displayed in the morning



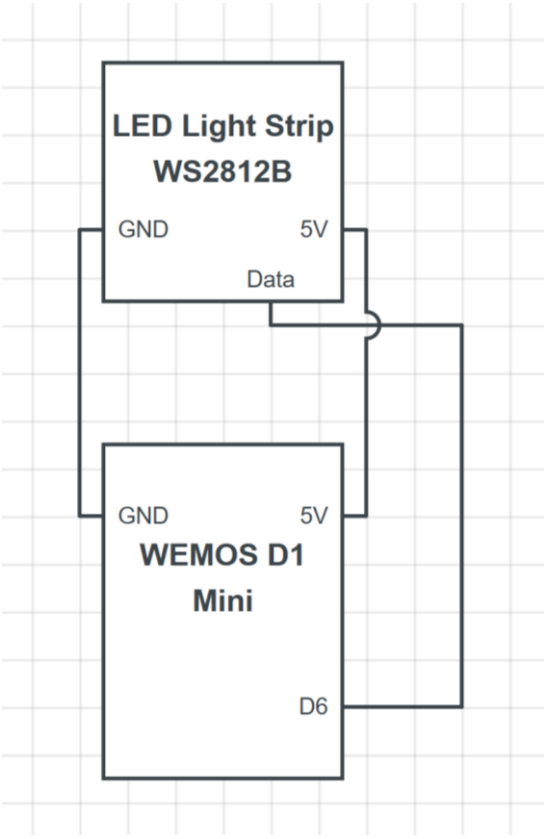
Even with only 15 LED lights turned on in the strip, powered at a lower brightness as well, sufficient light can be seen to illuminate a room

The Adafruit IO MQTT Feed. As can be seen with some of our precise time stamps, the IFTTT alarm trigger is working

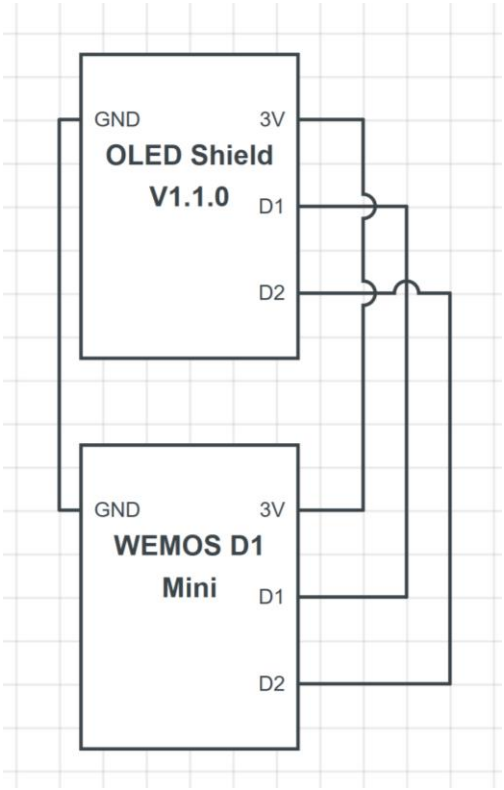


Google Assistant can be used to demo the light strip (value = "Lights"), show the weather temporarily (value = "Weather"), or can be used to turn off the alarm (value = "0"). The first two functions are extra features I implemented just for fun

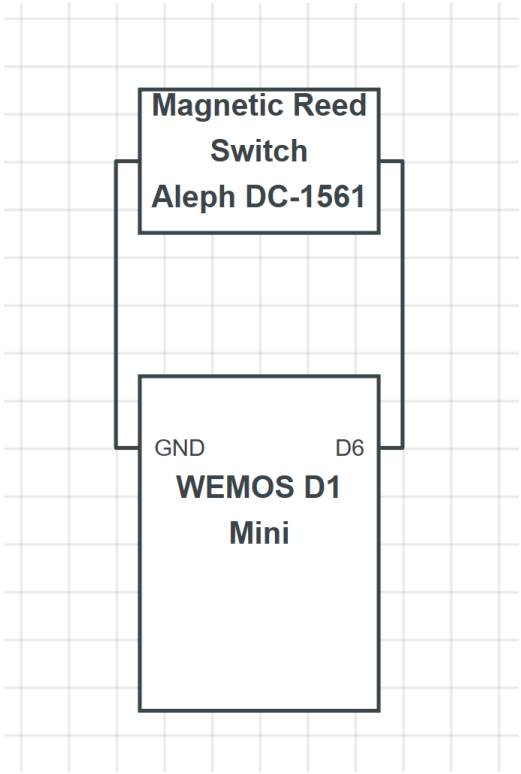
Circuit Diagrams



The LED Light Strip

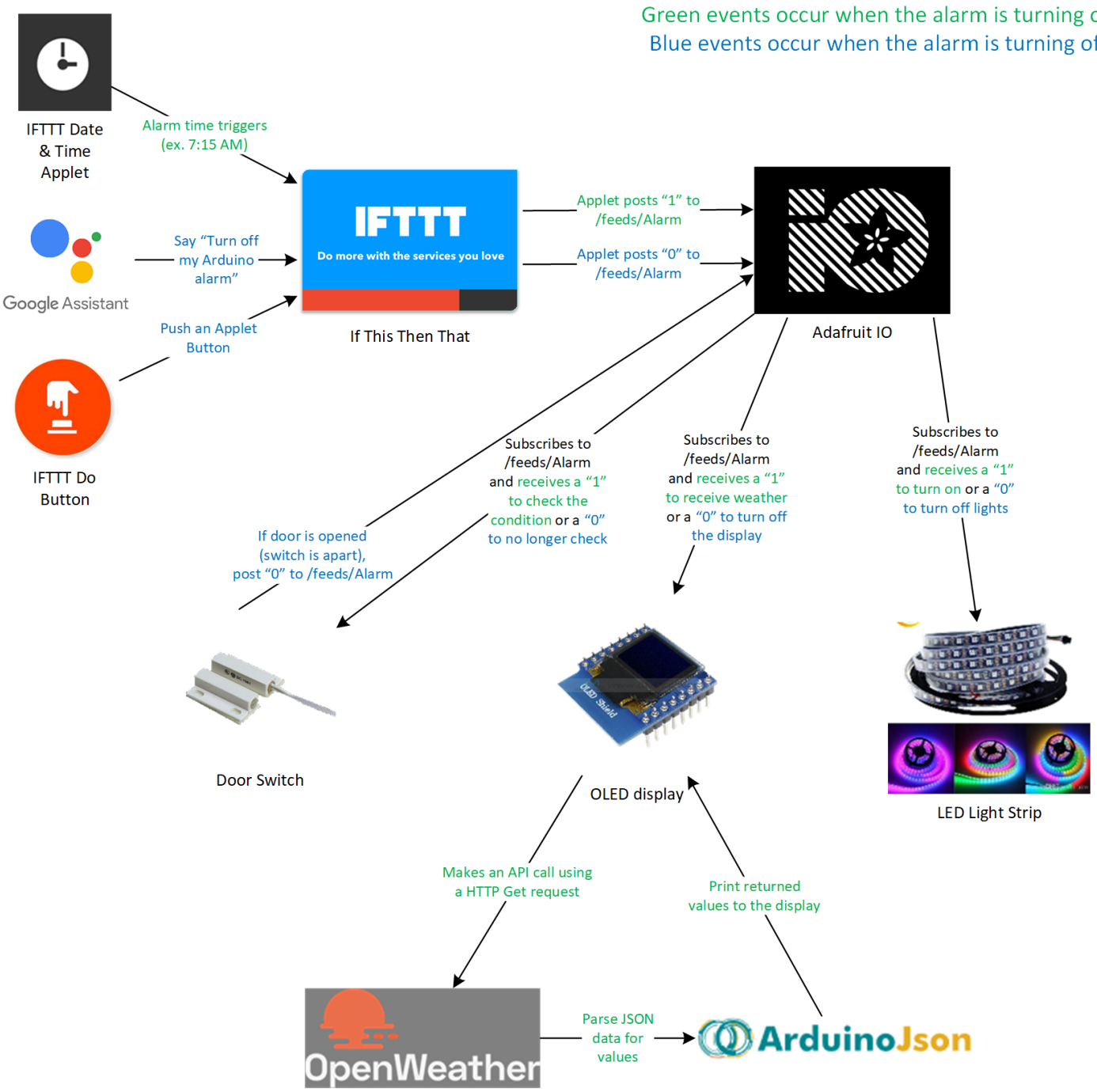


The OLED Shield Display



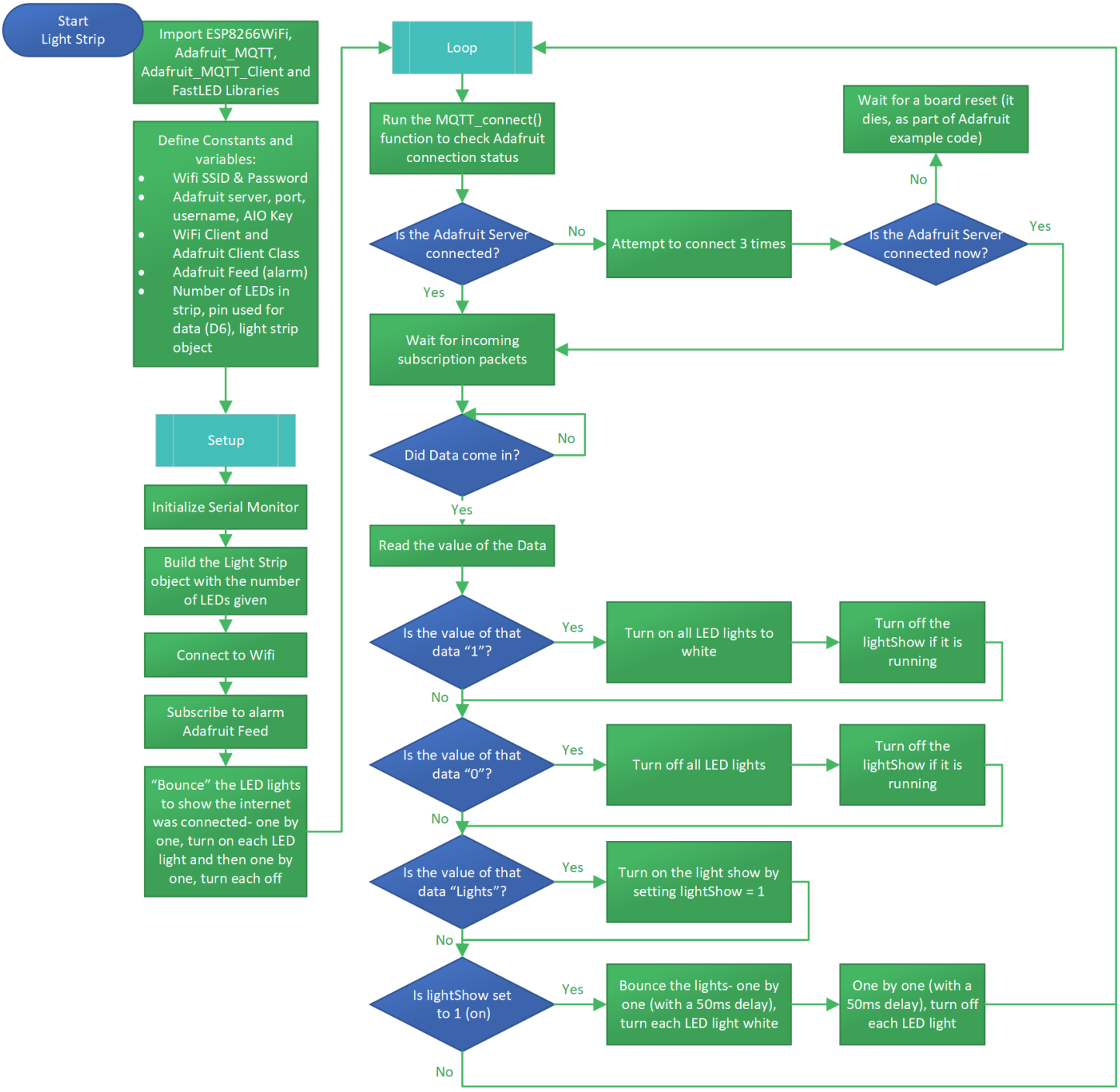
The Magnetic Reed Switch

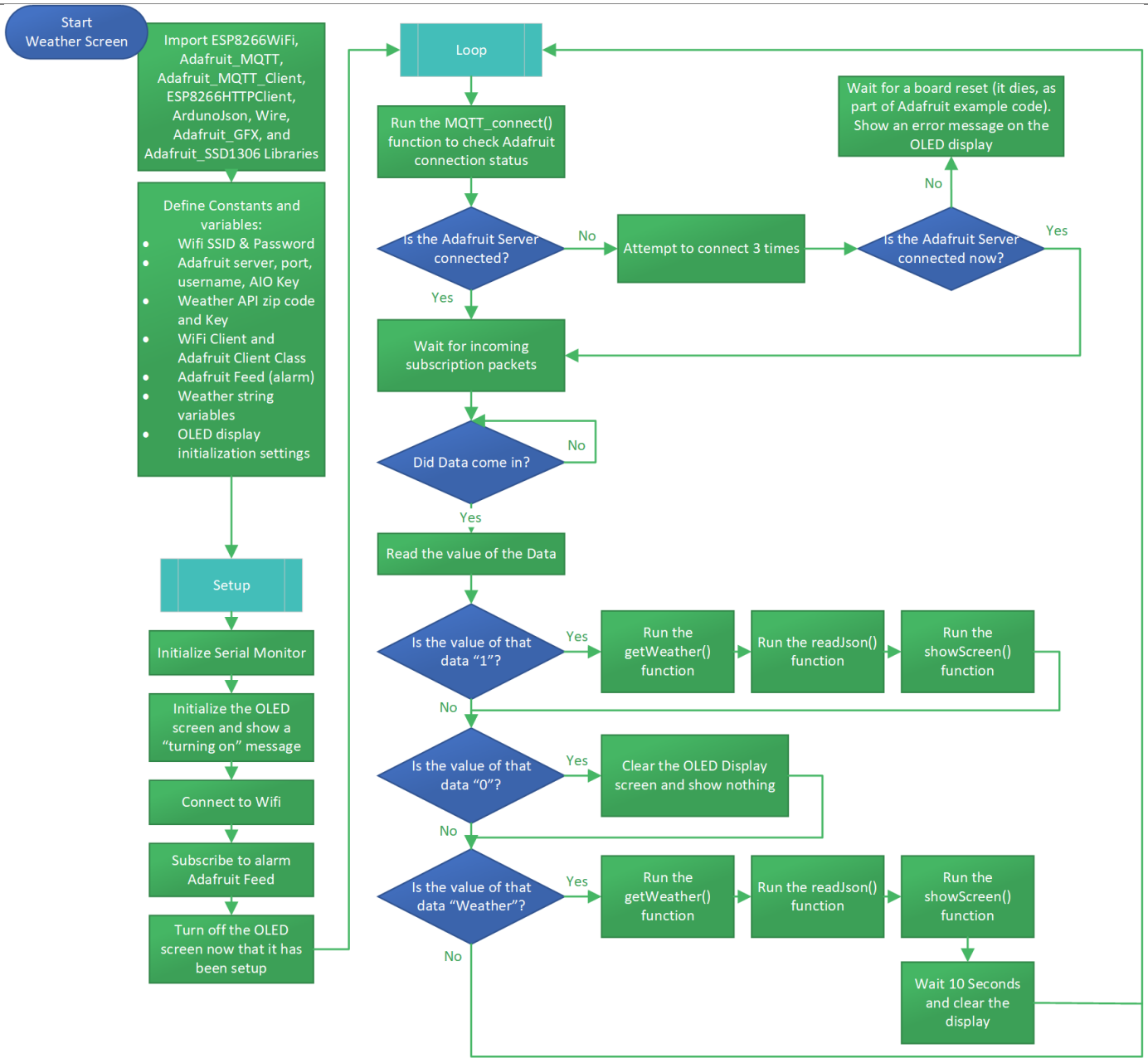
System Diagram



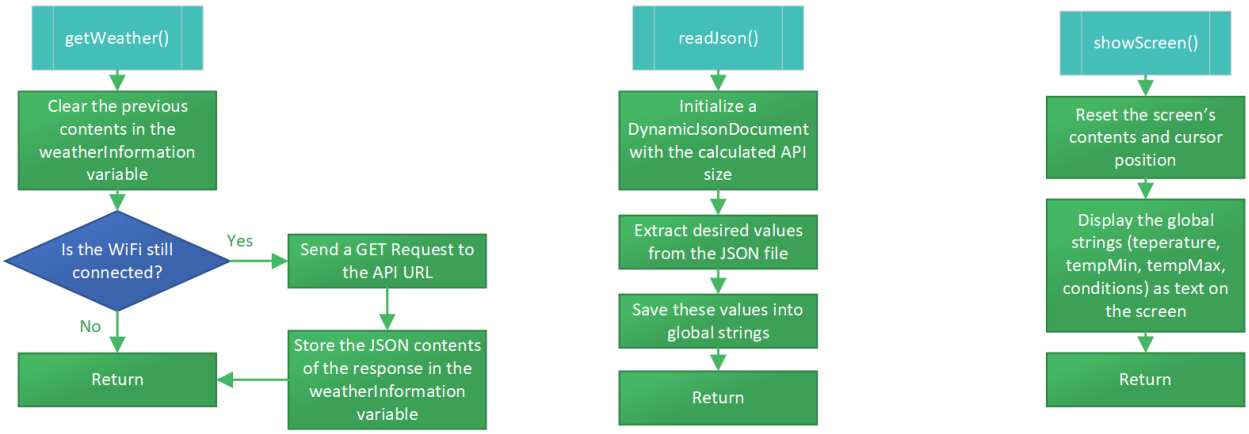
System Flowchart

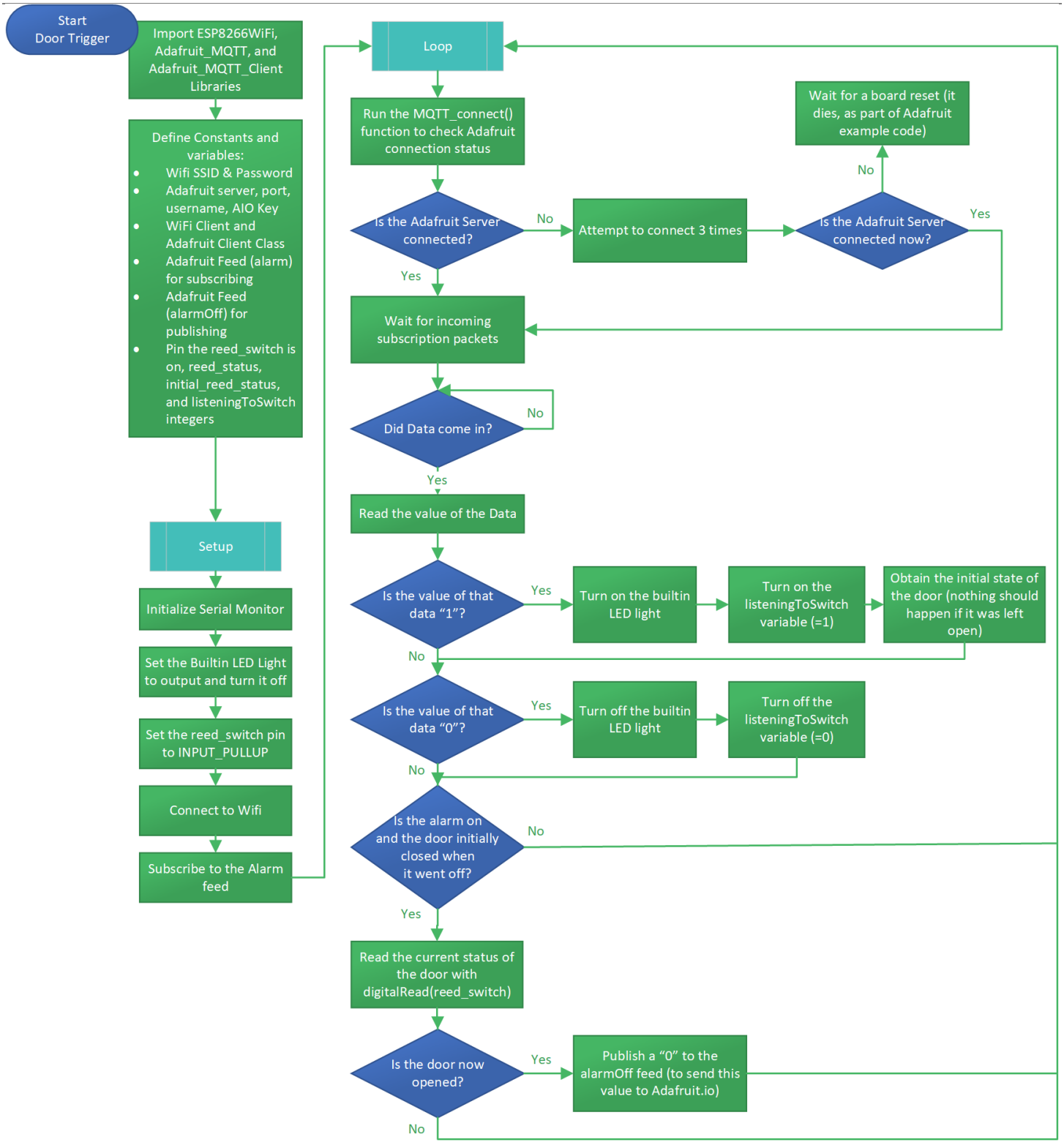
NOTE: A larger version of my flowcharts are available on my website.




Start Weather Screen
Import ESP8266WiFi, Adafruit_MQTT, Adafruit_MQTT_Client, ESP8266HTTPClient, ArdunoJson, Wire, Adafruit_GFX, and Adafruit_SSD1306 Libraries
Define Constants and variables:
Wifi SSID & Password
Adafruit server, port, username, AIO Key
Weather API zip code and Key
WiFi Client and Adafruit Client Class
Adafruit Feed (alarm)
Weather string variables
OLED display initialization settings
Setup
Initialize Serial Monitor
Initialize the OLED screen and show a "turning on" message
Connect to Wifi
Subscribe to alarm Adafruit Feed
Turn off the OLED screen now that it has been setup
Loop
Run the MQTT_connect() function to check Adafruit connection status
Is the Adafruit Server connected?
No
Attempt to connect 3 times
Is the Adafruit Server connected now?
No
Wait for a board reset (it dies, as part of Adafruit example code). Show an error message on the OLED display
Yes
Yes
Wait for incoming subscription packets
Did Data come in?
No
Yes
Read the value of the Data
Is the value of that data "1"?
Yes
Run the getWeather() function
Run the readJson() function
Run the showScreen() function
No
Is the value of that data "0"?
Yes
Clear the OLED Display screen and show nothing
No
Is the value of that data "Weather"?
Yes
Run the getWeather() function
Run the readJson() function
Run the showScreen() function
No
Wait 10 Seconds and clear the display

Functions


getWeather()
Clear the previous contents in the weatherInformation variable
Is the WiFi still connected?
Yes
Send a GET Request to the API URL
No
Return
Store the JSON contents of the response in the weatherInformation variable
readJson()
Initialize a DynamicJsonDocument with the calculated API size
Extract desired values from the JSON file
Save these values into global strings
Return
showScreen()
Reset the screen's contents and cursor position
Display the global strings (teperature, tempMin, tempMax, conditions) as text on the screen
Return


Start
Door Trigger
Import ESP8266WiFi, Adafruit_MQTT, and Adafruit_MQTT_Client Libraries
Define Constants and variables:
Wifi SSID & Password
Adafruit server, port, username, AIO Key
WiFi Client and Adafruit Client Class
Adafruit Feed (alarm) for subscribing
Adafruit Feed (alarmOff) for publishing
Pin the reed_switch is on, reed_status, initial_reed_status, and listeningToSwitch integers
Setup
Initialize Serial Monitor
Set the Builtin LED Light to output and turn it off
Set the reed_switch pin to INPUT_PULLUP
Connect to Wifi
Subscribe to the Alarm feed
Loop
Run the MQTT_connect() function to check Adafruit connection status
Is the Adafruit Server connected?
No
Attempt to connect 3 times
Is the Adafruit Server connected now?
No
Wait for a board reset (it dies, as part of Adafruit example code)
Yes
Yes
Wait for incoming subscription packets
Did Data come in?
No
Yes
Read the value of the Data
Is the value of that data "1"?
Yes
Turn on the builtin LED light
Turn on the listeningToSwitch variable (=1)
Obtain the initial state of the door (nothing should happen if it was left open)
No
Is the value of that data "0"?
Yes
Turn off the builtin LED light
Turn off the listeningToSwitch variable (=0)
No
Is the alarm on and the door initially closed when it went off?
No
Yes
Read the current status of the door with digitalRead(reed_switch)
Is the door now opened?
Yes
Publish a "0" to the alarmOff feed (to send this value to Adafruit.io)
No

Code (available at https://github.com/AaronNelson95/IT441)

Arduino Code

Final_Light_Strip.ino

```
/*
 * Created by Aaron Nelson
 * Final Project - Arduino Alarm - Light Strip
 * 12-9-19
 * This script is to be used with a WS2812B LED light strip. It connects to the
     Adafruit.io MQTT server and listens for a value to come into the 'Alarm' feed.
     When a '1' comes in, it lights up all of the LED lights. A '0' turns off these
     lights.
 *
 *
 * The Adafruit.io portion of this script was obtained from the Arduino example (after
     adding the Adafruit Library) at File -> Examples -> Adafruit MQTT Library ->
     mqtt_esp8266_callback
 *
 * The LED light strip portion was built from the example at
     https://www.norwegiancreations.com/2018/01/programming-digital-rgb-led-strips-
     with-arduino-and-the-fastled-library/
 *
    WS2812B LED light strip pins:
        Red Wire (power): +5VDC
        White Wire (ground): GND
        Green Wire (trigger): D6 (or any as specified in the code)
 */

#include <ESP8266WiFi.h>            // This contains the libraries that allows the board
                                        to connect to wifi
#include "Adafruit_MQTT.h"          // Library to access data from an Adafruit.io online
                                        feed
#include "Adafruit_MQTT_Client.h"   // Used to subscribe to an Adafruit.io online feed
#include "FastLED.h"                // Library to manipulate light strip colors


/************************ WiFi Access Point *********************************/
const char* ssid     = "ENTER WIFI HERE";       // Specify the name of your wifi
const char* password = "ENTER PASSWORD HERE";   // Specify the password for your wifi


/************************ Adafruit.io Setup *********************************/
#define AIO_SERVER      "io.adafruit.com"       // Pulling data from the Adafruit
                                                    website
#define AIO_SERVERPORT  1883                    // use 8883 for SSL
#define AIO_USERNAME    "ENTER ADAFRUIT NAME"   // Username for Adafruit (goes before
                                                    the /feed/# MQTT feed)
#define AIO_KEY         "ENTER ADAFRUIT KEY"    // Obtained by going to io.adafruit.com
                                                and clicking AIO Key link in top right.
                                                Copy the "Active Key" here


/************************** Feeds ***********************************/
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
```

```
// Setup the MQTT client class by passing in the WiFi client and MQTT server and login
      details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

// Watch a feed called 'alarm' and subscribe to changes.
Adafruit_MQTT_Subscribe alarm = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
                                                                    "/feeds/Alarm");


 /************************ Light Strip Settings ******************************/
// How many leds in your strip?
#define NUM_LEDS 15    // this actually has 150, up to 30 should be safe to use through
                          the board's voltage
#define DATA_PIN D6    // 2 is pin D4, but this will turn on the onboard LED as well
                          (which isn't desired)
CRGB leds[NUM_LEDS];  // Sets up the lights on the strip (up through the number
                          specified)

int lightShow = 0;    // Variable that manages if the special "Light" command was passed
                          in the feed (just a fun demo mode)




void setup() {
  Serial.begin(115200);             // This allows serial information when connected to
                                       a computer

  FastLED.addLeds<WS2812B, DATA_PIN, RGB>(leds, NUM_LEDS);

  /* Connect to WiFi */
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");    // This is shown on the serial if connected to a
                                       computer
  Serial.println(ssid);             // It displays the wifi it is trying to connect to

  WiFi.mode(WIFI_STA);              // It sets the wifis mode to "Station" (rather than
                                       "AP" which is the Access Point mode)
  WiFi.hostname("Light Strip");     // Hostname to uniquely identify our device
  WiFi.begin(ssid, password);       // Attempts to connect to wifi using the provided
                                       credentials

  while (WiFi.status() != WL_CONNECTED) {   // While the wifi is not connected yet:
    delay(500);                     // every half second,
    Serial.print(".");              // it prints a dot on the Serial Monitor to show it
                                       is still trying to connect

  }
  Serial.println("");
  Serial.println("WiFi connected"); // When it does connect, show that there was success

  mqtt.subscribe(&alarm);           // Setup MQTT subscription for alarm feed
```

```
  /* Bounce the LED Light colors to show it is connected to the internet */
  for (int i=0; i < NUM_LEDS; i++) {
    // One by one, turn on each LED light and keep it on
    leds[i] = CRGB::White;
    FastLED.setBrightness( 50 );
    FastLED.show();
    delay(50);
  }
  delay(1000);
  for (int i=NUM_LEDS; i > -1; i--) {
    // One by one, turn off each LED light in the reverse order they turned on
    leds[i] = CRGB::Black;
    FastLED.show();
    delay(50);
  }
}




void loop() {
  // Ensure the connection to the MQTT server is alive (this will make the first
  // connection and automatically reconnect when disconnected).
  MQTT_connect();

  // this is our 'wait for incoming subscription packets' busy subloop
  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(5000))) {
    if (subscription == &alarm) {
      // If something came through the "alarm" feed in Adafruit.io
      Serial.print(F("Got: "));
      String information = "";        // A string to insert the characters read by the
                                          feed
      char* ch = (char *)alarm.lastread;
      information.concat(ch);         // Add the last read character to the information
                                          string to get the full value
      Serial.println(information);    // Show the value that came in to the Serial
                                          Monitor

      // A "1" was received (the alarm is on)
      if (information == "1") {
        // This simply turns on every light in the LED strip to white (at 50 brightness)
        Serial.println("Starting the alarm!");
        lightShow = 0;
        for (int i=0; i < NUM_LEDS; i++) {
          leds[i] = CRGB::White;
          FastLED.setBrightness( 50 );
          FastLED.show();
          // delay(50);   // enable to turn the lights on one by one
        }
      }

      // A "0" was received (the alarm was turned off)
      if (information == "0") {
```

```
        // This simply turns off every light in the strip
        Serial.println("Ending the alarm!");
        lightShow = 0;
        for (int i=NUM_LEDS; i > -1; i--) {
          leds[i] = CRGB::Black;
          FastLED.show();
          // delay(50);   // enable to turn the lights off one by one
        }
      }

      // The word "Lights" was received
      if (information == "Lights") {
        // This turns on a light show that will bounce the path the LED lights turn on
            and off, end to end
        Serial.println("Showing Lights Show");
        lightShow = 1;
      }
    }
  }

  // This is a special "demo" that will flash the lights- having them turn on one by one
      until they are all on, and then they will turn off in reverse
  // It is only activated when the word "Lights" is passed into the alarm feed
  if (lightShow == 1) {
    for (int i=0; i < NUM_LEDS; i++) {
      // One by one, turn on each LED light and keep it on
      leds[i] = CRGB::White;
      FastLED.setBrightness( 50 );
      FastLED.show();
      delay(50);
    }
    delay(2000);
    for (int i=NUM_LEDS; i > -1; i--) {
      // One by one, turn off each LED light in the reverse order they turned on
      leds[i] = CRGB::Black;
      FastLED.show();
      delay(50);
    }
  }
}




// Function to connect and reconnect as necessary to the Adafruit.io MQTT server
// Should be called in the loop function and it will take care of connecting and
      reconnecting
void MQTT_connect() {
  int8_t ret;

  // Stop if already connected and return back to the loop
  if (mqtt.connected()) {
    return;
  }
```

```
  Serial.print("Connecting to Adafruit... ");

  uint8_t retries = 3;                    // It will attempt to connect 3 times before
                                                predicting the internet is down
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying Adafruit connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;    // Count against one of the retries because there was no success
       if (retries == 0) {
         // basically die and wait for the board to be reset
         while (1);
       }
  }
  Serial.println("Adafruit Connected!");
}
```

Final_Weather_Screen.ino

```
/*
 * Created by Aaron Nelson
 * Final Project - Arduino Alarm - Weather Screen
 * 12-9-19
 * This script is to be used with an Arduino OLED Shield, screen size 64x48px. It
      connects to the Adafruit MQTT server and listens to the alarm feed. When it
      obtains a value of '1', the alarm will activate. This will pull HTTP information
      from a weather API, parse through its JSON, and show information on a OLED screen.
 *
 *
 * The Adafruit.io portion of this script was obtained from the Arduino example (after
      adding the Adafruit Library) at File -> Examples -> Adafruit MQTT Library ->
      mqtt_esp8266_callback
 *
 * The weather API is obtained from https://openweathermap.org/current
 *
 * The HTTP GET request code was obtained from
      https://techtutorialsx.com/2016/07/17/esp8266-http-get-requests/
 *
 * The JSON reader was obtained from https://arduinojson.org/v6/example/http-client/ and
      Size and parameters were generated from https://arduinojson.org/v6/assistant/
 *
 * The LED screen portion was obtained from the Arduino example (after adding the
      Adafruit GFX and Adafruit SSD1306 Wemos Mini OLED libraries) at File -> Examples
      -> Adafruit SSD1306 Wemos Mini OLED -> ssd1306_64x48_i2c
 *
    OLED Shield pins (if soldering does not allow direct placement):
        3V3: 3V3
        D1 (SCL): D1
        D2 (SDA): D2
        GND: GND
 */
```

```
#include <ESP8266WiFi.h>              // This contains the libraries that allows the board
                                         to connect to wifi
#include <ESP8266HTTPClient.h>        // Library to perform a HTTP GET request
#include <ArduinoJson.h>              // Library to read and process JSON objects
#include <Wire.h>                     // Used to communicate with I2C devices
#include <Adafruit_GFX.h>             // The Adafruit graphics core library to work with
                                         OLED displays
#include <Adafruit_SSD1306.h>         // Library that works with the OLED Shield displays
#include "Adafruit_MQTT.h"            // Library to access data from an Adafruit.io online
                                         feed
#include "Adafruit_MQTT_Client.h"     // Used to subscribe to an Adafruit.io online feed


/************************ WiFi Access Point *********************************/
const char* ssid     = "ENTER WIFI HERE";       // Specify the name of your wifi
const char* password = "ENTER PASSWORD HERE";   // Specify the password for your wifi


/************************ Adafruit.io Setup *********************************/
#define AIO_SERVER      "io.adafruit.com"       // Pulling data from the Adafruit
                                                   website
#define AIO_SERVERPORT  1883                    // use 8883 for SSL
#define AIO_USERNAME    "ENTER ADAFRUIT NAME"   // Username for Adafruit (goes before
                                                   the /feed/# MQTT feed)
#define AIO_KEY         "ENTER ADAFRUIT KEY"    // Obtained by going to io.adafruit.com
                                                and clicking AIO Key link in top right.
                                                Copy the "Active Key" here


/************************ Weather API Settings **********************************/
String zipCode = "55555";
String weatherAPIKey = "ENTER API KEY";         // API key from
https://openweathermap.org/current


/***************************** Feeds ***************************************/
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login
      details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

// Watch a feed called 'alarm' and subscribe to changes.
Adafruit_MQTT_Subscribe alarm = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
                                                                  "/feeds/Alarm");


/************************ Weather JSON Variables **********************************/
String weatherInformation = "";     // Variable to hold API JSON data
String temperature = "";            // Variable to hold the obtained temperature
String tempMin = "";                // Variable to hold the obtained min temperature
String tempMax = "";                // Variable to hold the obtained max temperature
String conditions = "";             // Variable to hold weather condition information
```

```
/********************** OLED display settings ******************************/
// This defines important variables for the OLED screen to work. If using a different
      screen size, obtain this information from the corresponding examples
#define OLED_RESET -1  // GPIO0
Adafruit_SSD1306 display(OLED_RESET);
#define XPOS 0
#define YPOS 1
#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH  16
#if (SSD1306_LCDHEIGHT != 48)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif




void setup() {
  Serial.begin(115200);             // This allows serial information when connected to
                                          a computer

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize the OLED display with the
                                                 I2C addr 0x3C (for the 64x48)
  // init done

  // Show image buffer on the display hardware. Since the buffer is initialized with an
      Adafruit splashscreen internally, this will display the splashscreen.
  // display.display();

  // Clear the buffer.
  display.clearDisplay();           // Show nothing on the OLED screen
  display.setTextSize(1);           // Set the text size for text (1 is appropriate as
                                          the screen has a small resolution)
  display.setTextColor(WHITE);      // Set the text color to white
  display.setCursor(0,0);           // Reset the text placement position to the top left
  display.println("Now");           // Write a message to the screen
  display.println("Turning");
  display.println("On...");
  display.display();                // Actually commit and show this message onto the
                                          display screen


  /* Connect to WiFi */
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");   // This is shown on the serial if connected to a
                                          computer
  Serial.println(ssid);             // It displays the wifi it is trying to connect to

  WiFi.mode(WIFI_STA);              // It sets the wifis mode to "Station" (rather than
                                          "AP" which is the Access Point mode)
  WiFi.hostname("Weather Monitor");     // Hostname to uniquely identify our device
  WiFi.begin(ssid, password);       // Attempts to connect to wifi using the provided
                                          credentials
```

```
  while (WiFi.status() != WL_CONNECTED) {   // While the wifi is not connected yet:
    delay(500);                             // every half second,
    Serial.print(".");                      // it prints a dot on the Serial Monitor to show it
                                                is still trying to connect
  }
  Serial.println("");
  Serial.println("WiFi connected"); // When it does connect, show that there was success

  mqtt.subscribe(&alarm);           // Setup MQTT subscription for alarm feed

  display.clearDisplay();           // Show nothing on the OLED display now that the
                                        device has been set up
  display.display();                // Reflect this on the screen
}




void loop() {
  // Ensure the connection to the MQTT server is alive (this will make the first
  // connection and automatically reconnect when disconnected).
  MQTT_connect();

  // this is our 'wait for incoming subscription packets' busy subloop
  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(5000))) {
    if (subscription == &alarm) {
      // If something came through the "alarm" feed in Adafruit.io
      Serial.print(F("Got: "));
      String information = "";          // A string to insert the characters read by the
                                            feed
      char* ch = (char *)alarm.lastread;
      information.concat(ch);           // Add the last read character to the information
                                            string to get the full value
      Serial.println(information);      // Show the value that came in to the Serial
                                            Monitor

      // A "1" was received (the alarm is on)
      if (information == "1") {
        Serial.println("Starting the alarm!");
          getWeather();                 // Run the function to perform a HTTP GET request
          readJson();                   // Run the function to interpret the received JSON
                                            file
          showScreen();                 // Show parsed information on the OLED display
      }

      // A "0" was received (the alarm was turned off)
      if (information == "0") {
        Serial.println("Ending the alarm!");
        display.clearDisplay();         // Show nothing on the display
        display.display();              // Commit the plan to clear the screen
      }
```

```
      // The word "Weather" was received
      if (information == "Weather") {
        // This will temporarily show the weather information on the screen, only for 10
            seconds
        Serial.println("Showing Weather Information");
          // Perform the same functions as when the alarm is on and running
          getWeather();
          readJson();
          showScreen();
          // Then wait for 10 seconds and turn off the display again
          delay(10000);
          display.clearDisplay();
          display.display();
      }
    }
  }
}




// Function to connect and reconnect as necessary to the MQTT server.
// Should be called in the loop function and it will take care of connecting and
      reconnecting.
void MQTT_connect() {
  int8_t ret;

  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to Adafruit... ");

  uint8_t retries = 3;                  // It will attempt to connect 3 times before
                                              predicting the internet is down
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying Adafruit connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;    // Count against one of the retries because there was no success
       if (retries == 0) {
         // basically die and wait for the board to be reset after showing an error
            message on the display screen
         display.clearDisplay();        // Clear the previous contents of the screen
         display.setCursor(0,0);        // Reset the text cursor to the top left
         display.println("ERROR");      // Display there was an error
         display.println("with MQTT!");
         display.display();             // Commit this to the screen
         while (1);
       }
  }
  Serial.println("Adafruit Connected!");
```

```
}




// Function to perform an HTTP Get request on a weather API website to obtain JSON data
void getWeather() {
  // Weather API obtained from https://openweathermap.org/current
  // Template from https://techtutorialsx.com/2016/07/17/esp8266-http-get-requests/
  Serial.println("Obtaining Weather Information... ");

  weatherInformation = "";                // Variable to hold our weather JSON (clear it
                                                if something was in it)
  if (WiFi.status() == WL_CONNECTED) {  //Check WiFi connection status
    HTTPClient http;                    //Declare an object of class HTTPClient
    http.begin("http://api.openweathermap.org/data/2.5/forecast?zip=" + zipCode +
                        ",us&cnt=1&units=imperial&appid=" + weatherAPIKey);
                                          //Specify request destination (the weather API
                                                with the user's ZIP and Key)
    int httpCode = http.GET();          // Send the request
    if (httpCode > 0) {                 //Check the returning code
      String payload = http.getString();    //Get the request response payload
      // Serial.println(payload);           //Print the response payload
      weatherInformation.concat(payload);   // Write the response payload to the
                                                weatherInformation variable

    }
    http.end();                         //Close the HTTP connection
  }
  Serial.println(weatherInformation);   // Show the weather JSON information on the
                                                Serial Monitor
}




// Function to parse through obtained JSON data
void readJson() {
  // Example provided at https://arduinojson.org/v6/example/http-client/
  // Size and parameters generated from https://arduinojson.org/v6/assistant/
  Serial.println("Parsing JSON Information... ");

  const size_t capacity = 2*JSON_ARRAY_SIZE(1) + 2*JSON_OBJECT_SIZE(1) +
            2*JSON_OBJECT_SIZE(2) + JSON_OBJECT_SIZE(4) + JSON_OBJECT_SIZE(5) +
            JSON_OBJECT_SIZE(6) + JSON_OBJECT_SIZE(7) + JSON_OBJECT_SIZE(8) + 320;   //
                                                                    Find this value online
  DynamicJsonDocument doc(capacity);    // You must specify the JSON document's size to
                                          allocate its memory pool in the heap

  // Parse JSON object
  DeserializationError error = deserializeJson(doc, weatherInformation);
  if (error) {
    // Could not read JSON information
    Serial.print(F("deserializeJson() failed: "));
```

```
    Serial.println(error.c_str());
    return;
  }


  // Extract and display values, the location and how to obtain these values are also
      obtained at the ArdunoJSON website
  JsonObject list_0 = doc["list"][0];
  JsonObject list_0_main = list_0["main"];
  float list_0_main_temp = list_0_main["temp"]; // 28.53
  float list_0_main_temp_min = list_0_main["temp_min"]; // 21.36
  float list_0_main_temp_max = list_0_main["temp_max"]; // 28.53

  JsonObject list_0_weather_0 = list_0["weather"][0];
  const char* list_0_weather_0_main = list_0_weather_0["main"]; // "Clouds"
  const char* list_0_weather_0_description = list_0_weather_0["description"]; // "few
                                                                                clouds"

  // Set the extracted values into our defined variables (also obtained from the
      website)
  temperature = String(list_0_main_temp);
  tempMin = String(list_0_main_temp_min);
  tempMax = String(list_0_main_temp_max);
  conditions = String(list_0_weather_0_main);

  // Show this content on the serial monitor
  Serial.println(temperature);
  Serial.println(tempMin + "-" + tempMax);
  Serial.println(conditions);
}




// Function to write extracted JSON variables on the OLED display
void showScreen() {
  // From the example code ssd1306_64x48_i2c

  display.clearDisplay();           // Show nothing on the OLED screen
  display.setCursor(0,0);           // Reset the text cursor to the top left
  display.println("Morning!");      // Write out the temperature messages
  display.println("temp:" + temperature);
  display.println("min: " + tempMin);
  display.println("max: " + tempMax);
  display.println(conditions);
  display.display();                // Commit this text to the board
}
```

Final_Door_Trigger.ino

```
/*
 * Created by Aaron Nelson
 * Final Project - Arduino Alarm - Door Trigger
```

```
 * 12-9-19
 * This script is to be used with a reed switch. It connects to the Adafruit.io MQTT
      server and listens for a value to come into the 'Alarm' feed. When a '1' comes in
      and the door opens, it should publish a value of '0' to turn off the alarm.
 *
 *
 * The Adafruit.io portion of this script was obtained from the Arduino example (after
      adding the Adafruit Library) at File -> Examples -> Adafruit MQTT Library ->
      mqtt_esp8266_callback
 *
    ReedSwitch Pins:
        One Wire: Pin12 (D6)
        The Other Wire: GND
 */

#include <ESP8266WiFi.h>            // This contains the libraries that allows the board
                                          to connect to wifi
#include "Adafruit_MQTT.h"          // Library to access data from an Adafruit.io online
                                          feed
#include "Adafruit_MQTT_Client.h"   // Used to subscribe to an Adafruit.io online feed


/************************ WiFi Access Point *********************************/
const char* ssid     = "ENTER WIFI HERE";       // Specify the name of your wifi
const char* password = "ENTER PASSWORD HERE";   // Specify the password for your wifi


/************************ Adafruit.io Setup *********************************/
#define AIO_SERVER      "io.adafruit.com"       // Pulling data from the Adafruit
                                                      website
#define AIO_SERVERPORT  1883                    // use 8883 for SSL
#define AIO_USERNAME    "ENTER ADAFRUIT NAME"   // Username for Adafruit (goes before
                                                      the /feed/# MQTT feed)
#define AIO_KEY         "ENTER ADAFRUIT KEY"    // Obtained by going to io.adafruit.com
                                          and clicking AIO Key link in top right. Copy
                                          the "Active Key" here


/************************** Feeds *******************************************/
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login
      details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

// Watch a feed called 'alarm' and subscribe to changes.
Adafruit_MQTT_Subscribe alarm = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
                                                                    "/feeds/Alarm");

// Prepare the feed 'alarm' to publish a '0' when the alarm should be turned off.
Adafruit_MQTT_Publish alarmOff = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
                                                                    "/feeds/Alarm");
```

```
 /********************** Reed Switch Settings ****************************/
int reed_switch = 12;               // Pin D6 is connected to one of the switch wires.
                                       If using a different pin, specify that here
int reed_status = 0;                // When this is 0, it means it is connected to the
                                    magnet. A 1 means it is disconnected and separated
int initial_reed_status = 0;        // Keeps the last value of the reed magnet (to only
                                       publish to the sensor when a CHANGE is made)
int listeningToSwitch = 0;          // When this is a 1 (alarm is on), it listens to the
                                 door switch and turns the alarm off if the door is opened




void setup() {
  Serial.begin(115200);             // This allows serial information when connected to
                                       a computer

  pinMode(LED_BUILTIN, OUTPUT);     // Prepares the builtin light pin for output (which
                                       turns on when the reed magnet is connected)
  digitalWrite(LED_BUILTIN, HIGH);  // Initially, turn off the builtin LED light
  pinMode(reed_switch, INPUT_PULLUP);     // Initialize the reed switch pin for input.
                                             Pullup is necessary for when the magnet
                                             becomes reconnected

  /* Connect to WiFi */
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");    // This is shown on the serial if connected to a
                                       computer
  Serial.println(ssid);             // It displays the wifi it is trying to connect to

  WiFi.mode(WIFI_STA);              // It sets the wifis mode to "Station" (rather than
                                       "AP" which is the Access Point mode)
  WiFi.hostname("Door Switch");     // Hostname to uniquely identify our device
  WiFi.begin(ssid, password);       // Attempts to connect to wifi using the provided
                                       credentials

  while (WiFi.status() != WL_CONNECTED) {   // While the wifi is not connected yet:
    delay(500);                     // every half second,
    Serial.print(".");              // it prints a dot on the Serial Monitor to show it
                                       is still trying to connect
  }
  Serial.println("");
  Serial.println("WiFi connected"); // When it does connect, show that there was success

  mqtt.subscribe(&alarm);           // Setup MQTT subscription for alarm feed
}




void loop() {
  // Ensure the connection to the MQTT server is alive (this will make the first
  // connection and automatically reconnect when disconnected).
```

```
  MQTT_connect();

  // this is our 'wait for incoming subscription packets' busy subloop
  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(5000))) {
    if (subscription == &alarm) {
      // If something came through the "alarm" feed in Adafruit.io
      Serial.print(F("Got: "));
      String information = "";        // A string to insert the characters read by the
                                          feed
      char* ch = (char *)alarm.lastread;
      information.concat(ch);         // Add the last read character to the information
                                          string to get the full value
      Serial.println(information);    // Show the value that came in to the Serial
                                          Monitor

      // A "1" was received (the alarm is on)
      if (information == "1") {
        // Start listening to the door switch's status. If it is opened, turn off the
            alarm
        Serial.println("Starting the alarm!");
        digitalWrite(LED_BUILTIN, LOW);         // The LED light turns on
        reed_status = digitalRead(reed_switch); // Read the current value of the magnet
                                                    switch
        initial_reed_status = reed_status;      // Store the initial value of the door
                                            (so if the door was opened when the alarm
                                            turned on, don't automatically turn it off)
        Serial.println(initial_reed_status);
        listeningToSwitch = 1;
      }

      // A "0" was received (the alarm was turned off)
      if (information == "0") {
        // Turn off the light and don't worry about collecting/publishing door data
        Serial.println("Ending the alarm!");
        digitalWrite(LED_BUILTIN, HIGH);        // The LED light turns off
        listeningToSwitch = 0;
      }
    }


  }

  if (listeningToSwitch == 1 && initial_reed_status == 0) {   // The alarm is on and the
                                            door is currently closed (which is together)
    reed_status = digitalRead(reed_switch);     // Check if the magnet is connected or
                                                    not
    if (reed_status == 1) {                     // If the door goes from closed to
                                                    opened
      Serial.println("switch just disconnected (the door opened)");
      if (! alarmOff.publish("0")) {            // Publish the value of "0" to turn the
                                                    alarm off
        Serial.println(F("Failed to turn off alarm"));
      } else {
        Serial.println(F("Alarm should turn off soon"));
```

```
      }
    }
  }
}




// Function to connect and reconnect as necessary to the Adafruit.io MQTT server
// Should be called in the loop function and it will take care of connecting and
      reconnecting
void MQTT_connect() {
  int8_t ret;

  // Stop if already connected and return back to the loop
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to Adafruit... ");

  uint8_t retries = 3;                  // It will attempt to connect 3 times before
                                          predicting the internet is down
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying Adafruit connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;    // Count against one of the retries because there was no success
       if (retries == 0) {
         // basically die and wait for the board to be reset
         while (1);
       }
  }
  Serial.println("Adafruit Connected!");
}
```