Final Report

Light and Lock Automation System

Version 1.0

Prepared by: Nat Fortems

Liam Smeaton

Minh Tran

Table of Contents (Nat) 2

1 – Introduction

The primary goal of the “Light and Lock Automation Project” was to obtain practical knowledge, skill, and experience in all aspects of working on a more complex engineering project. The secondary objective was to produce a functional product according to the project proposal and progress schedule we had prepared at the beginning of this semester. While we have undoubtedly succeeded in the first goal, having learned, in one member’s case, three new programming languages, as well as all members learning more about power supply issues and general design problems, the completion of our secondary objective could be described as ‘tenuous’ at best. While our group did assemble a working project for presentation on the date it was due, the final product was lacking several systems that we had wanted to include and were described in our documentation. In short, we demonstrated an early functional prototype instead of any kind of finished product. In this document, we will attempt to examine and present our work on this project, and explain our failing and successes with the product we ended up presenting.

2 – Lock System

2.1 Introduction and Overview

The lock subsystem reached all minimum functionality requirements for this project, with few design flaws which will be covered in future sections. The basic functionality of the lock subsystem was to be able to lock and unlock a deadbolt when receiving a signal from the central hub, or the website. Another basic requirement was the ability to be attached to a door with screws by someone who has little to no hardware experience. This requirement was shown in the presentation as the lock subsystem was attached to a 2x4 wood block to display this requirement(FIG-2). The lock subsystem is not a finished project and should be seen as a first prototype to the project as there are many factors that need to be polished.

2.2 Comparisons Between Planned and Completed

The lock system had some major changes between the proposal, detailed design, and the outcome. First, the chassis was incomplete and in a second prototype would include a faceplate and room for circuitry. Second, the scope was adjusted to not include a sensor which would be able to tell the website and hub if it is locked or unlocked. Lastly, the lock subsystem did not include a gear ratio to increase its torque rating, the gears used had a ratio of 1-1. This section will highlight the differences between the proposal, detailed design, and the outcome of the project, and the changes that would be made in the future.

The chassis is a main component of the lock subsystem, as it contains the entire functionality of the mechanical aspect of the lock system(FIG—1). The biggest design flaw of the chassis is that I did not include it in the detailed design nor the requirements specification despite how important it is for the system. The chassis is meant to contain the entirety of the circuitry motor, power supply which is accessible, and the radio communication. The outcome for the prototype was a simple open concept that allowed us to manipulate parts and view what was happening in the inside easily(FIG—2). The main components that the chassis was missing in the presentation was the faceplate, and room for circuitry. The faceplate would have properly held the motor and handle setup in place without the use of hot glue which I was forced to use to prove functionality. The unfinished faceplate that was used is only half of what it should have been(FIG—7). The changes that need to be made are the implementation of a faceplate, and a removeable top to access the battery power supply.(FIGS 5—8)

We planned to include a sensor since the project proposal, but chose not to include it early in the design stage—there is no real reason for this and it should have been included. One feature the lock system lacks is the ability to communicate back to the central hub. When the hub locks the system, it toggles a local variable to keep itself aware if the door is locked or unlocked. This is clearly a flaw as it does not count for the door to be locked manually by a user and still function properly. This issue would have been solved with a position sensor detecting if the deadbolt was on the east-west axis, or the north-south axis and updating the central hub respectively. This change would have removed a pivotal design flaw and would be implemented in future prototyping.

Lastly, the gear ratio used was 1-1 with a motor too strong for its load. The detailed design included a gear ratio of 1.3 to increase the torque rating. The gear ratio of 1.3 was not included in the outcome of the project because I overlooked if they would mesh properly. Proper gear ratios would allow for the motor to have a lower holding torque and be smaller geometrically to downsize the chassis. Furthermore, the proper gear ratio would allow the motor driver to act in micro stepping mode for a smoother lock/unlock of the door. Also, the bore diameter was too large for the driver arm of the motor and would need to be decreased, or have a shim implemented. Perfecting the gear ratio would increase the overall functionality of the lock subsystem as it would remove many small issues such as gears disconnecting, motor being too big and too strong, motor being jittery, and the dimensions of the chassis.

The changes mentioned above would allow for a better second prototype which has better reached the requirements of our initial proposal. Although I am aware that implementation of these features would also cause failures of their own, they would solve all issues that I am aware of at this stage of testing. The gear ratio change would allow for a smaller motor, which would help downsize the chassis and help include the circuitry within it. The increased room inside the chassis would also make it easier to implement a sensor which solves the user-website clash of detecting what state the lock is in. For future work on this project we would plan to implement all these features and further test their conditions.

2.3 Testing Plan Analysis

The main component that required accessing our test plans was the link from the radio chip to the output of the Arduino. This was tested by verifying the output of the Arduino was as stated in the test plans. The test plans met their requirements as all the testing we did on the lock subsystem are noted in the test plans document. The only difference in the test plans that need to be noted is the change from a 6AA battery supply to a 5AA battery power supply which means the test value for the power supply should be 7.5V.

2.4 Circuit Design

The circuit design remains almost unchanged from the detailed design—the only two changes worth noting is the inclusion of the micro stepping pins in the future, as well as the switch from the ESP8266 to radio modules. With the implementation of a gear ratio in a future prototype it also opens the ability to use the micro stepping pins on the motor driver (MS1, MS2, MS3). These pins smooth out the stepping of the stepper motor by adding additional steps within a single step the motor receives(FIG). Smoother stepping of the motor would make the lock subsystem more user friendly by decreasing how loud and jumpy it is currently (FIGS 3—4).

2.5 Code

The coding design for the locking subsystem remains, for the most part, unchanged from the detailed design, with one exception. This one major exception was the complete amputation of any wi-fi connectivity code as detailed in both our requirements document and the detailed design. This code segment was subsequently replaced with basic radio connection code to provide the needed connection to the hub, and in turn, the website, that the wireless chip would have otherwise provided.

```
#include "Arduino.h"
#include <SPI.h>
#include <RF24.h>

int motorSpin();

// defines pins numbers
const int stepPin = 3;
const int dirPin = 4;
const int RESET = 5;

RF24 radio(7, 8);

byte addresses[][6] = {"Lock1","Lock2"};
char data;
```

```
void setup() {
  Serial.begin(9600);
  // Sets the two pins as Outputs
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  pinMode(RESET,OUTPUT);

  radio.begin();
  radio.setPALevel(RF24_PA_HIGH);
  radio.setDataRate(RF24_2MBPS);
  radio.setChannel(124);
}

void loop(){
  radio.openReadingPipe(1, addresses[1]);
  radio.startListening();

  if ( radio.available()) {
    while (radio.available()) {
      radio.read( &data, sizeof(char));
    }

    radio.stopListening();
    radio.openWritingPipe(addresses[0]);
    radio.write( &data, sizeof(char) );
    radio.startListening();

    Serial.print("Sent response ");
    motorSpin();
    Serial.println(data);

  }
}
int motorSpin(){
  if ( data == 'O' ){
    delay(1);
    digitalWrite(RESET,HIGH);
    delay(1);
    digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
```

```
   for(int x = 0; x < 75; x++) {
     digitalWrite(stepPin,HIGH);
     delay(10);
     digitalWrite(stepPin,LOW);
     delay(10);
     Serial.println("Turn would have happened");
   }
   digitalWrite(RESET,LOW);
   delay(1);
   directionToSpin = -1;
   Serial.println("Closing");
   data = 0;
 }
 if ( data == 'X' ){
   delay(1);
   digitalWrite(RESET,HIGH);
   delay(1);
   digitalWrite(dirPin,LOW); // Enables the motor to move in a particular direction
   // Makes 200 pulses for making one full cycle rotation
   for(int x = 0; x < 75; x++) {
     digitalWrite(stepPin,HIGH);
     delay(10);
     digitalWrite(stepPin,LOW);
     delay(10);
     Serial.println("Turn would have happened");
   }
   digitalWrite(RESET,LOW);
   delay(1);
   directionToSpin = 0;
 }
 Serial.println("Opening");
 data = 0;
}
```

2.6 References

FIG 1 Assembled Chassis

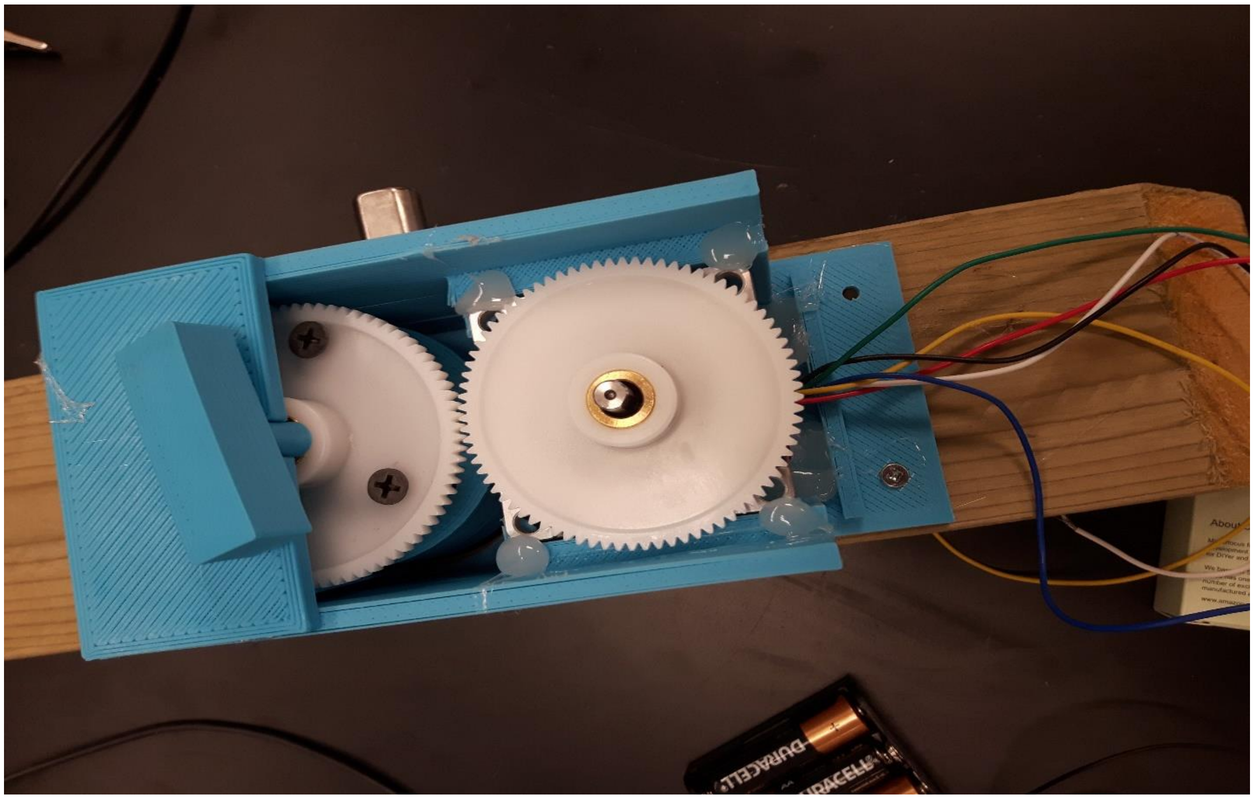


FIG 2—Lock subsystem

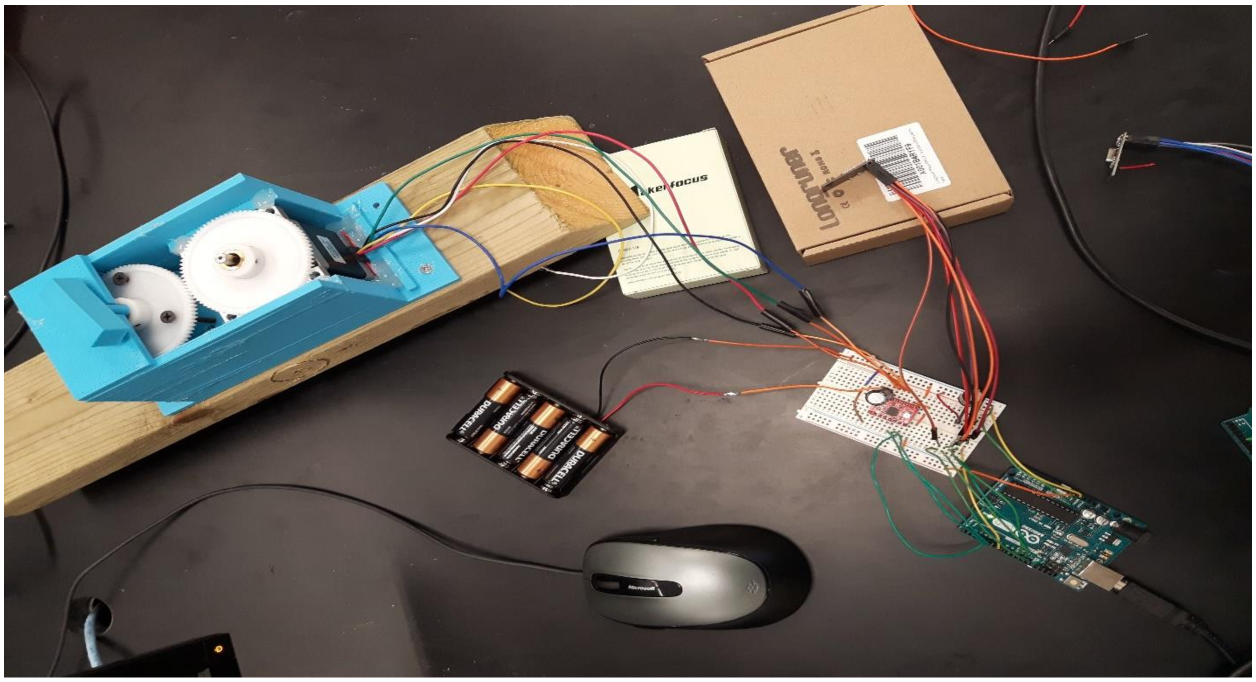


FIG 3—Driver Circuit

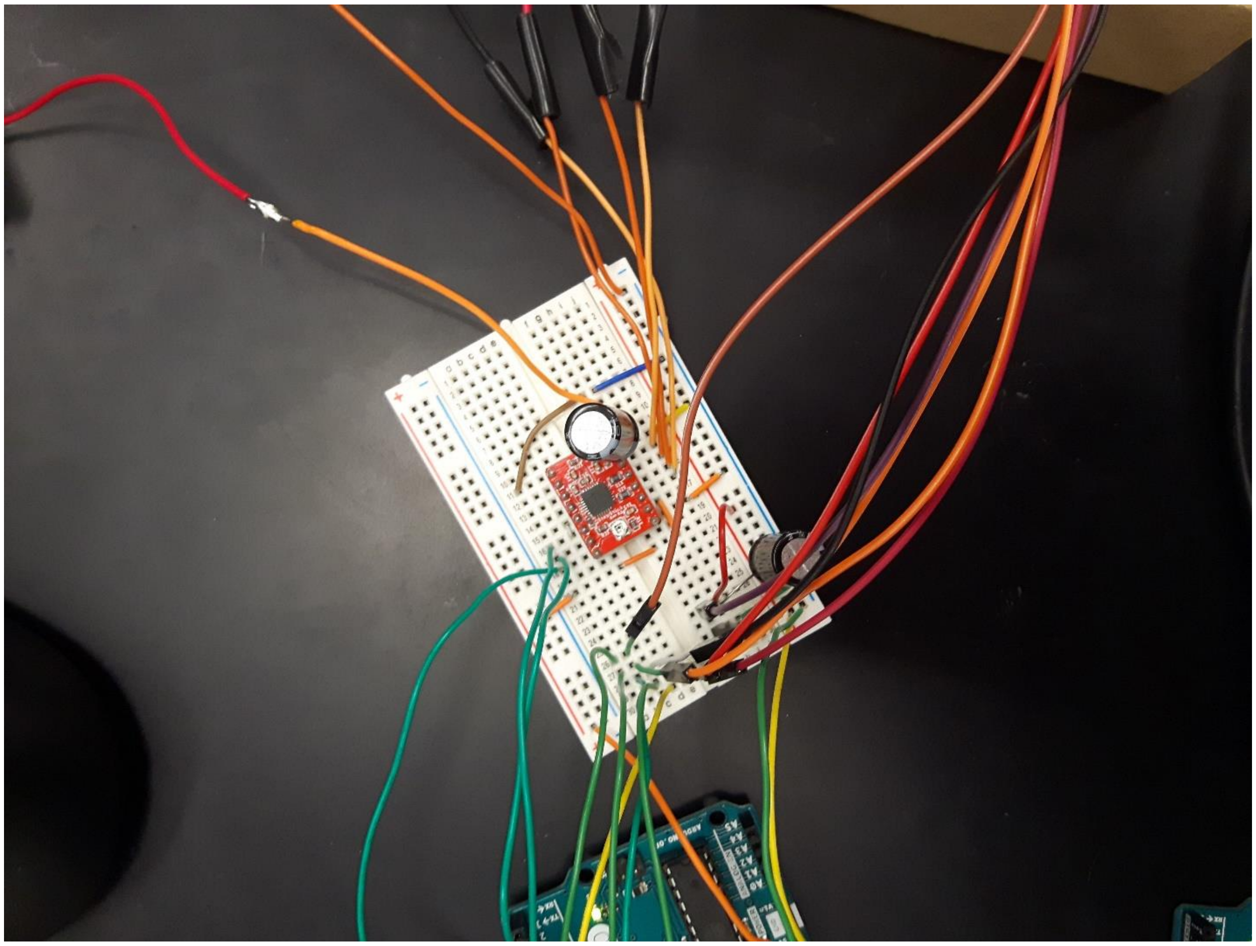


FIG 4—Circuit Schematic

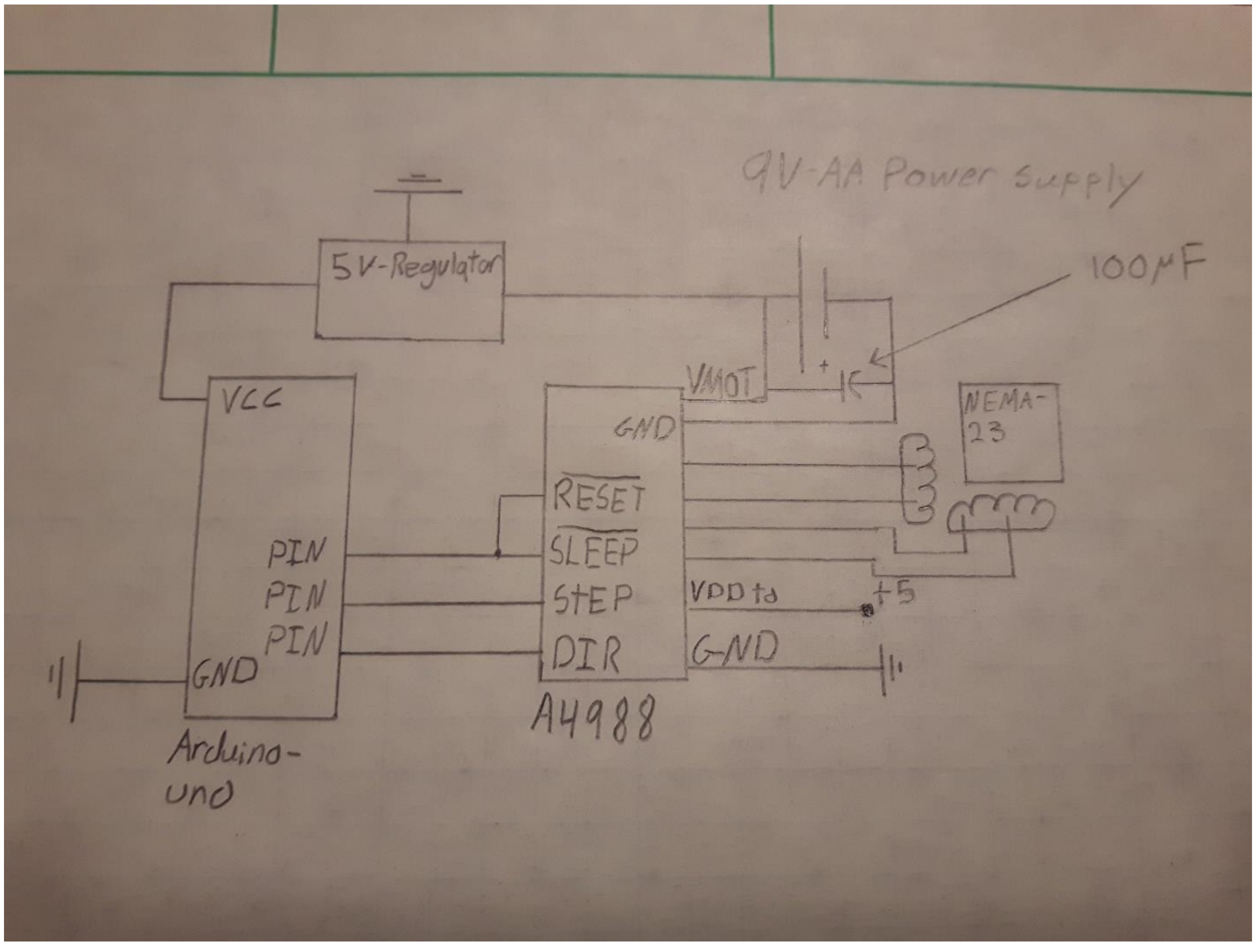


FIG 5—Handle

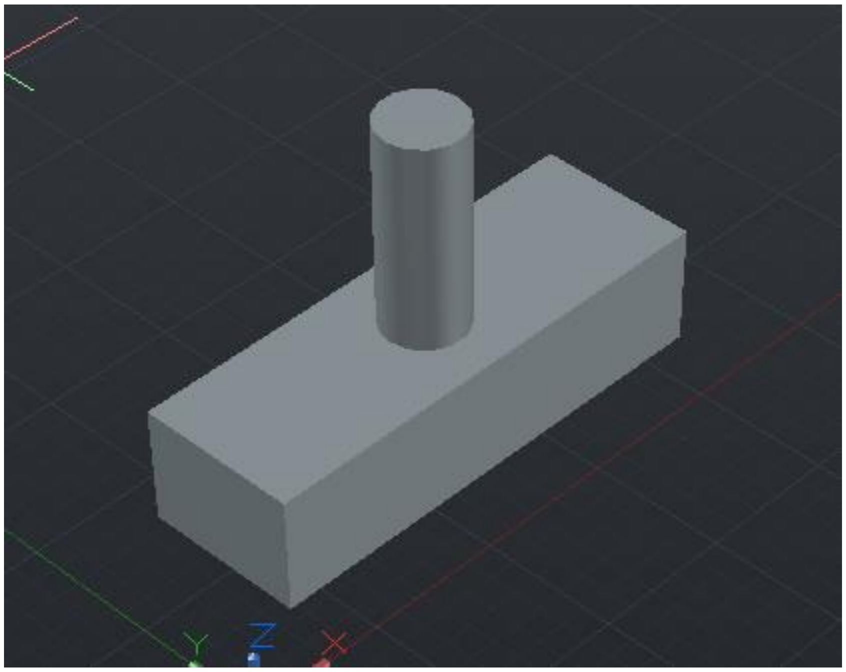


FIG 6—Attacher

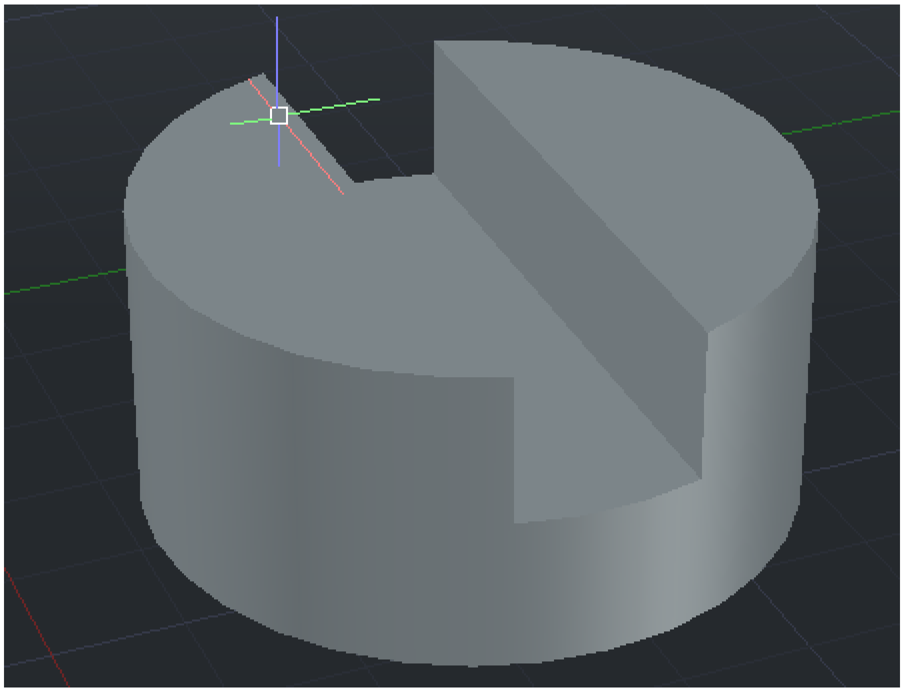


FIG 7—Face Plate

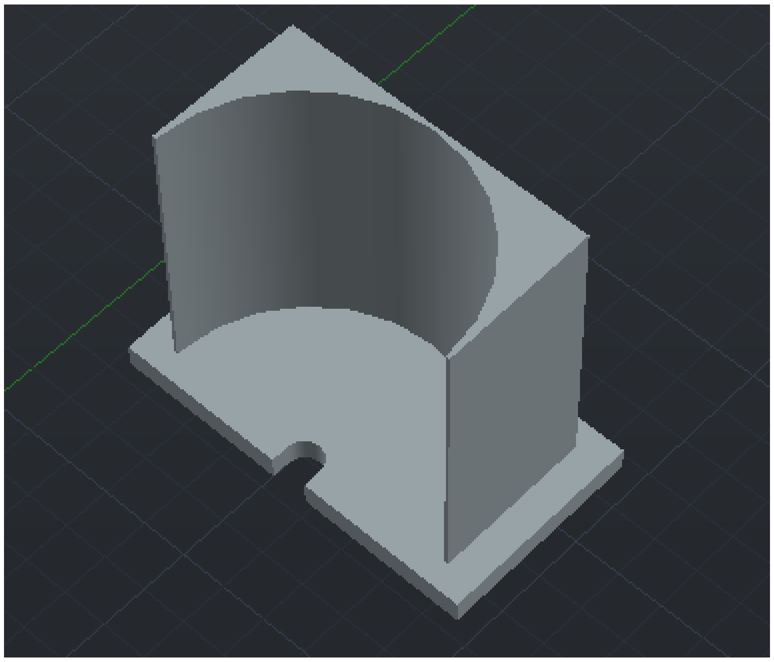
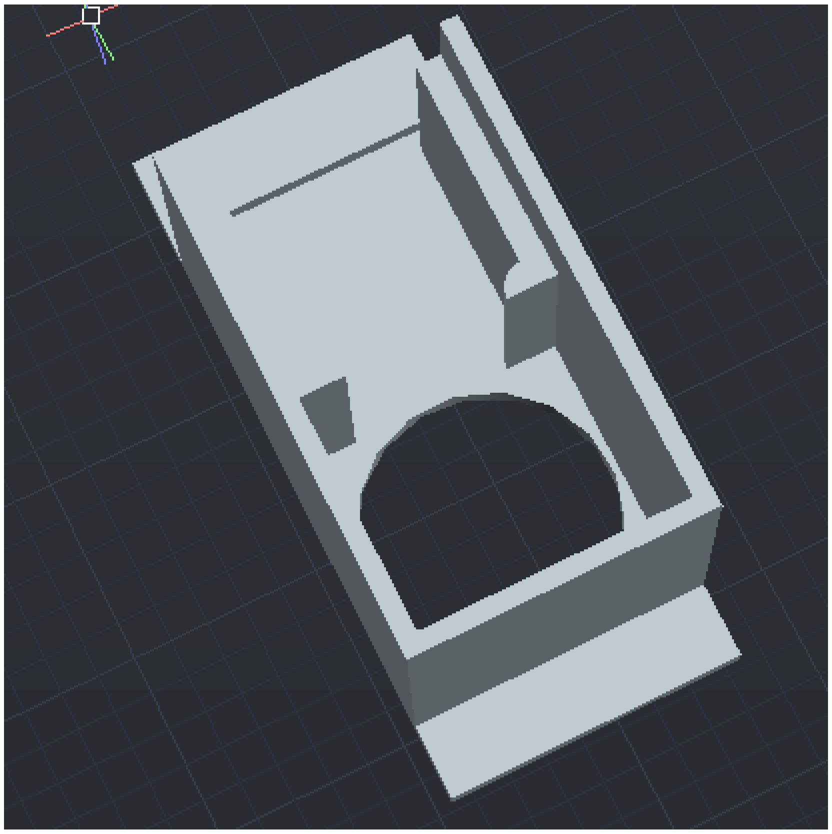


FIG 8—Chassis



3 – Light System

3.1 Introduction and Overview

While, fortunately, a basic version of the light control system was able to be presented and tested to check for code and circuit functionality, the product as displayed represents a mere first-stage prototype of what was outlined in the requirements, detailed design, and proposal document submitted. The overall issues of time constraint, project management, and planning that plague all other components of the Lock and Light Automation System are especially notable and relevant here, and this section of the project is clearly missing several subsystems.

3.2 Structural and Mechanical Design

In our original plan, we had specified that we would have a relatively lightweight and small casing for the body of the light control. Unfortunately, this plan never came to fruition, both because we never properly set up our relay and used a LED to demonstrate functionality instead and because of a lack of time and large/inefficient breadboarding.

3.3 Circuit Design

Unfortunately, not one of the circuitry requirements for the light control system outlined in the system requirements and detailed design document remain as originally conceived. These major changes were caused by our last-week-of-semester switch to NRF24L01 radio modules for communication with the hub instead of the planned ESP8266, as well as the removal of the relay from the demonstration in place of a simple LED, as we were unable to procure a lamp or other plug-in light operating off of 120 V AC to mutilate and insert our relay in. As a result of these changes, the light control system currently lacks a functional power source and cannot operate when disconnected from an outlet, as we never wired our plugin to take AC power from the light we were powering.

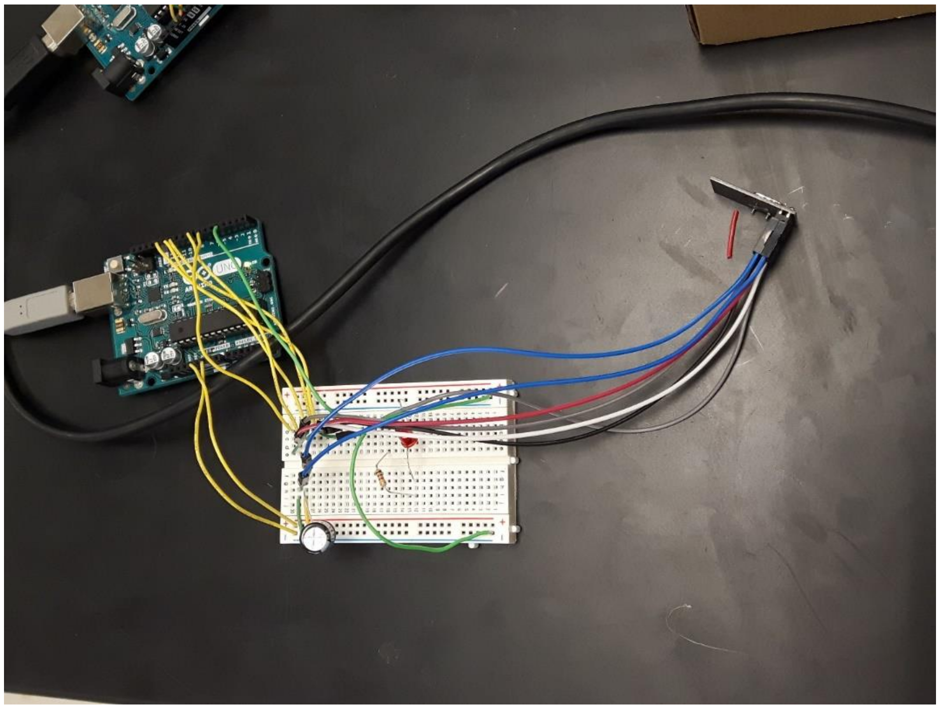


FIG 3.1 – Final Light Circuit

3.4 Code

```
#include "Arduino.h"
#include <SPI.h>
#include <RF24.h>

int lightSwitch();
char directionToSpin;

// defines pins numbers
const int ledPin = 4;
```

```
RF24 radio(7, 8);

byte addresses[][6] = {"Lght1","Lght2"};
char data;

void setup() {

  Serial.begin(9600);
  // Sets the two pins as Outputs
  pinMode(ledPin ,OUTPUT);

  radio.begin();
  radio.setPALevel(RF24_PA_HIGH);
  radio.setDataRate(RF24_2MBPS);
  radio.setChannel(124);
}
void loop(){

  radio.openReadingPipe(1, addresses[1]);
  radio.startListening();

  if ( radio.available()) {
    while (radio.available()) {
      radio.read( &data, sizeof(char));
    }

    radio.stopListening();
    radio.openWritingPipe(addresses[0]);
    radio.write( &data, sizeof(char) );
    radio.startListening();

    Serial.print("Sent response.");
    lightSwitch();
    Serial.println(data);
  }
}
int lightSwitch(){
  if (data=='O'){
    digitalWrite(ledPin, HIGH); // Enables the motor to move in a particular direction
    data = 0;
  }
  if (data=='X'){
   digitalWrite(ledPin, LOW); // Enables the motor to move in a particular direction
   data = 0;
  }
}
```

3.5 Testing Review

Input(s):	Expected Output(s):	Testing:
Relay "Toggle"	Nearly constantly at 0V, with periods of 5V signals	Verify that the relay is toggling the "Load Circuitry" path by ensuring that a 5V signal will toggle the current state of the "Load Circuitry" output to its inverse.
Relay "Load Circuitry":	Expected to alternate between 120 V AC and 0 V AC based on state of switch.	Verify that the load circuitry is functional by confirming that it can conduct a 120V signal through to its opposite side. (switch state may need to be altered)
Testing: Testing was unsuccessful, as relay was never successfully integrated into the circuit. An LED was, however, implemented into the circuit instead, which works on the same principal as toggling a relay’s state (high for open, low for close), so that part of testing was at least partially successful.		
Function int checkLightStatus() in microcontroller code, as well as Pin 12 on microcontroller.	Integer holding status of power to the light: '1' for on, and '0' for off.	Verify presence of either 0V or 5V DC signal to pin 12 on microcontroller through 120-240V AC. 2. Run function, and check outputs. It should produce a '0' if the input to the pin is zero, and '1' if the input to the pin is 5V DC.
Testing: Testing was unsuccessful, as the function was never implemented into our code.		
Function sendInfo(int Request) in microcontroller code	Signal sent to central hub over wireless connection.	Verify that hub successfully connects to light microcontroller as a client, then verify that light microcontroller can communicate requested information to hub (possible data sent is string containing 'id' of light controller, or int containing status of light)
Testing: Testing was partially successful, as the ESP8266 no longer functions as the communication device between the lock and light controls and the hub, as well as the hub not truly needing any communication back from the light and lock controls in its present		

state. This communication/connection can be made, however, and is currently used for confirmation of signal/data sent.		
Function recieveSignal()	Data updated in microcontroller, or microcontroller sends signal back to hub containing integer indicating status of light.	Verify that hub successfully connects to light microcontroller as a client, then verify that hub can communicate requested information to hub (possible data received is string containing new 'id' of light controller, integer 0 interpreted as request for status of light, or integer 1 interpreted as request to toggle light)
Testing: Testing was mostly successful, however the 'ID' functionality has been removed from our code, and again, our system uses radio modules instead of the ESP8266 to communicate.		
Function void toggleState()	Signal sent through pin 13 on microcontroller to solid-state relay.	Verify that pin 13 temporarily outputs a 5V signal to the solid-state relay when toggleState() is called.
Testing: Function works mostly as intended, however, the solid state relay has been removed from our circuit and been replaced with an LED.		

4 – Central Hub System

4.1 Introduction and Overview

The central hub system achieved most of the minimum functionality established in the design and requirements document, with a notable flaw being our lack of any real structure or casing for it. The basic functionality of the central hub system was to be able to send a signal to the lock and light control systems that tells them to lock and unlock a deadbolt or turn on and off a light. This signal was able to be sent over a touchscreen or over the website, although we did encounter minor errors with the interaction between the touchscreen and website.

4.2 Circuit Design

The central hub circuitry is, unfortunately, messy, due to a lack of access to the milling machine. Despite being messy, however, the circuitry on the protoboard connects all necessary devices and components and fulfills all the requirements and functionalities listed on our requirements and design documents.

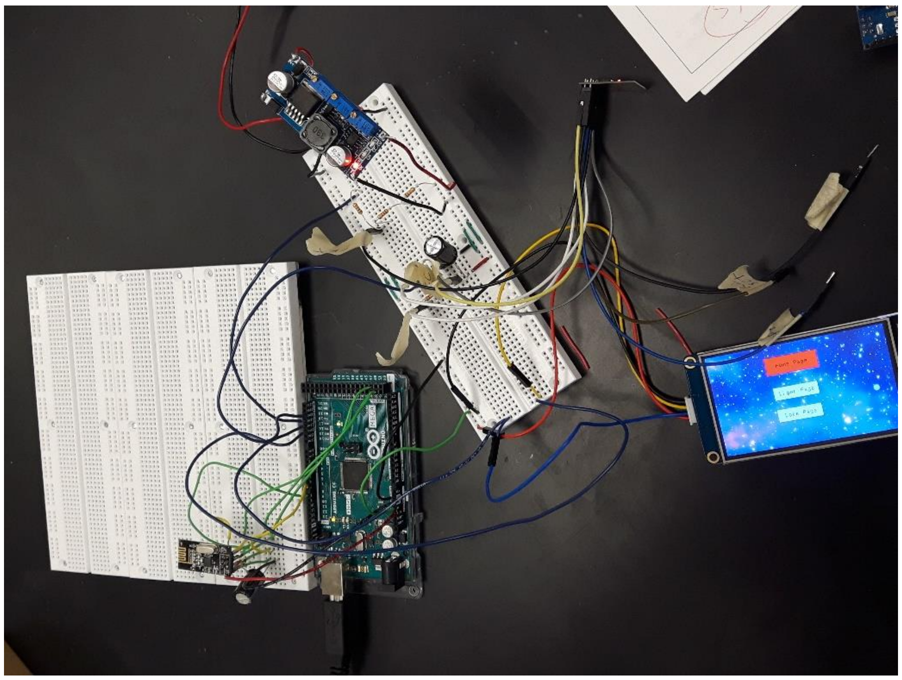


Fig. 4.0 – Circuit Diagram

4.3 Mechanical Design

The central hub possesses no moving parts, due to the resistive touchscreen we used instead of buttons. Unlike the plans developed in the functional requirements and detailed design documents, we lack any casing or housing structure, partially due to a lack of time and partially due to the bulkiness of the breadboarded circuit we had created.

4.4 Code

```
#include <SoftwareSerial.h>
#include <Nextion.h>
#define DEBUG true
#define esp8266 Serial1
#define nexSerial Serial2

#include "Arduino.h"
#include <SPI.h>
#include <RF24.h>

int sendSignal( char data, int toOpen );
int bounceBack( char data, int toOpen );
String templock = "";
String templight = "";

RF24 radio(7, 8);
```

```
byte addresses[][6] = {"Lock1", "Lock2", "Lght1", "Lght2"};
int check;

//NexText lockstatus = NexText(1, 2, "lockstatus");
//NexText lightstatus = NexText(2, 6, "lightstatus");
NexButton block = NexButton(1,3,"block");
NexButton bunlock = NexButton(1,4,"bunlock");
NexButton bon = NexButton(2,3,"bon");
NexButton boff = NexButton(2,4,"boff");
NexButton lightpage = NexButton(0,4,"lightpage");
NexButton lockpage = NexButton(0,2,"lockpage");
String lock = "";
String light = "";
int page = 0 ;
int count = 0;
NexTouch *nex_listen_list[] =
{
  &block,//add button
  &bunlock,//add button
  &bon,// add button
  &boff,//add button
  &lightpage,// going to light page
  &lockpage, //going to lock page
  NULL// String terminate
};

void setup() {
  Serial.begin(9600);
  esp8266.begin(115200);
  nexInit();
  bon.attachPush(Pushon,&bon);
  boff.attachPush(Pushoff,&boff);
  block.attachPush(Pushlock,&block);
  bunlock.attachPush(Pushunlock,&bunlock);
  lightpage.attachPush(gotolight,&lightpage);
  lockpage.attachPush(gotolock,&lockpage);
  radio.begin();
  radio.setPALevel(RF24_PA_HIGH);
  radio.setDataRate(RF24_2MBPS);
  radio.setChannel(124);
}

void loop() {

  nexLoop(nex_listen_list);
```

```
if(!templock.equals(lock)){
  String cmd = "";
  lock = templock;
  cmd += 'Y';
  cmd += "lock=";
  cmd += lock;
  cmd += 'X';
  esp8266.println(cmd);
  Serial.println(cmd);
  if(lock.equals("Locked")){
        Serial.println("Executing");
        check = sendSignal( 'X', 1 );
        check = bounceBack( 'X', 0 );
       }
  else if(lock.equals("Unlocked")) {
        Serial.println("Executing");
        check = sendSignal( 'O', 1 );
        check = bounceBack( 'O', 0 );
      }
}
if(!templight.equals(light)){
  String cmd = "";
  light = templight;
  cmd += 'Y';
  cmd += "light=";
  cmd +=  light;
  cmd += 'X' ;
  esp8266.println(cmd);
  Serial.println(cmd);
  if(light.equals("on")){
      Serial.println("Executing");
      check = sendSignal( 'O', 3 );
      check = bounceBack( 'O', 2 );
  }
  else if(light.equals("off")){
      Serial.println("Executing");
      check = sendSignal( 'X', 3 );
      check = bounceBack( 'X', 2 );
  }
}

if(page == 1)
  {updateLock();}
else if(page == 2)
```

```
    {updateLight();}
}

int sendSignal( char data, int toOpen ){
  radio.openWritingPipe(addresses[toOpen]);
  radio.stopListening();
  char number = random(0, 254);
  radio.stopListening();
  if (!radio.write( &data, sizeof(char) )) {
    Serial.println("No acknowledgement of transmission - receiving radio device connected?");
  }
}

int bounceBack( char data, int toOpen ){
  radio.openReadingPipe(1, addresses[toOpen]);
   radio.startListening();
  unsigned long started_waiting_at = millis();

  while ( ! radio.available() ) {
    // Oh dear, no response received within our timescale
    if (millis() - started_waiting_at > 200 ) {
      Serial.println("No response received - timeout!");
      return;
    }
  }
  unsigned char dataRx;
  radio.read( &dataRx, sizeof(char) );
  // Show user what we sent and what we got back
  Serial.print("Sent: ");
  Serial.print(data);
  Serial.print(", received: ");
  Serial.println(dataRx);
  // Try again 1s later
  delay(1000);
}

 // check if the esp is sending a message
void serialEvent1(){
  if(esp8266.available()){
    String esp = esp8266.readStringUntil('\n');
    if(esp.equals("Status:")){
      esp8266.readStringUntil('=');
      templight = esp8266.readStringUntil('\n');
      Serial.println(light);
      esp8266.readStringUntil('=');
      templock = esp8266.readStringUntil('\n');
      delay(10);
      Serial.println(lock);
    }
  }
}
```

4.5 Testing Report:

Input(s):	Expected Output(s):	Testing
120-240 V AC power from 'main' of house.	Regulated 9V DC signal to Vin pin on Arduino.	Verify existence of both 240V AC input and 9V DC output.
Testing: Testing was successful, the power supply performed as expected.		

Hub Functions Test Plan			
Function:	Input(s):	Expected Output(s):	Testing:
Lock_toggle()	Recieves a lock toggle HTTP request from the website or when the Toggle button on lock page of touchscreen is pressed.	The hub microcontroller connects as a client to a specified lock microcontroller, then sends integer 1 to the lock microcontroller.	Can be tested by verifying that the hub connects and sends information successfully to the lock system microcontroller.
Light_toggle()	Recieves a light HTTP request from the website or when the Toggle button on light page of touchscreen is pressed.	The hub microcontroller connects as a client to a specified light microcontroller, then sends integer 1 to the lock microcontroller.	Can be tested by verifying that the hub connects and sends information successfully to the light system microcontroller.
Testing: Testing was mostly successful, however, since the lock and website both send distinct open and close requests, the toggling function works for both the lock and light controls, and the toggling function sends a character ‘X’ or ‘O’ as an open or close request instead of integers.			
Lock_update()	When Lock_toggle() is called	Match the website's lock status label with the Hub's status.	
Light_update()	When Light_toggle() is called	Match the website's light status label with the Hub's status.	
Testing: Testing was unsuccessful, as we were unsuccessful with enabling communication from the Arduino Mega to the ESP8266.			
Get_lockTime()	When a new time is enter into the lock time's textfield	Add the entered time to the Lock_schedule[] Array	Can be test by printing the entire Lock__schedule[]

			array after calling this function
Get_LightTime()	When a new time is enter into the lock time's textfield	Add the entered time to the Light_schedule[] Array	Can be test by printing the entire Light__schedule[] array after calling this function
LockTime_Delete()	When pressing a delete button	The coresponding Lock time will be remove from the Lock_schedule[] array	Can be test by printing the entire Lock__schedule[] array after calling this function
LockTime_Delete()	When pressing a delete button	The coresponding Light time will be remove from the Lock_schedule[] array	Can be test by printing the entire Light__schedule[] array after calling this function
Check_Time()	When a minute has pass	If the clock's current time match with any of the time in Lock_schedule[], call Lock_toggle() if the door is currently unlock. Do the same thing with the Light_schedule[] array and call Light_toggle() if the light is on.	This function should work even when the Hub is in sleep mode.
Testing: Testing was unsuccessful, as we were unsuccessful with implementing a timekeeping or schedule system using the ESP8266 and Arduino Mega.			

5 – Website and Internet

5.1 Introduction (Website and Internet):

In order to interface the system with the internet, we decide to use the esp8266. For demonstration, I set up a local host server which host the html and JSON files of the website. The esp8266 can parse the information of the JSON file by sending a http get request. The esp8266 can also overwrite the contents of the JSON file by sending a http post request to the html file. Ideally the user can change the status of the lock and light freely from accessing the website. The user can also change the status of lock and light displayed on the website via a touchscreen that connect to the hub.

5.2 Comparison Between Planned and Done:

Originally, we plan to include a schedule functionality that allow the lock and light to operate automatically using a set schedule that the user can freely edit from the hub and the website. However, we were unable to implement this function due to time constraint. We also cannot establish a bi-directional communication between the Hub and the internet since we were not able to figure out how to parse the messages sent from the hub to the esp8266. Due to the issue with bi-directional communication, we cannot display the status of the lock and light from the website. Furthermore, whenever we change the status of the lock or light on the hub, the website will always overwrite them making the touchscreen unusable when the website is being accessed.

We also planned to be able to change the status of light and lock from anywhere we want as long as we have access to the internet. But due to not having access to the school router, we cannot port forward the IP address of the esp8266. This allow websites to access the esp8266 directly despite them not being in the same domain.

5.3 Changes from Original Design:

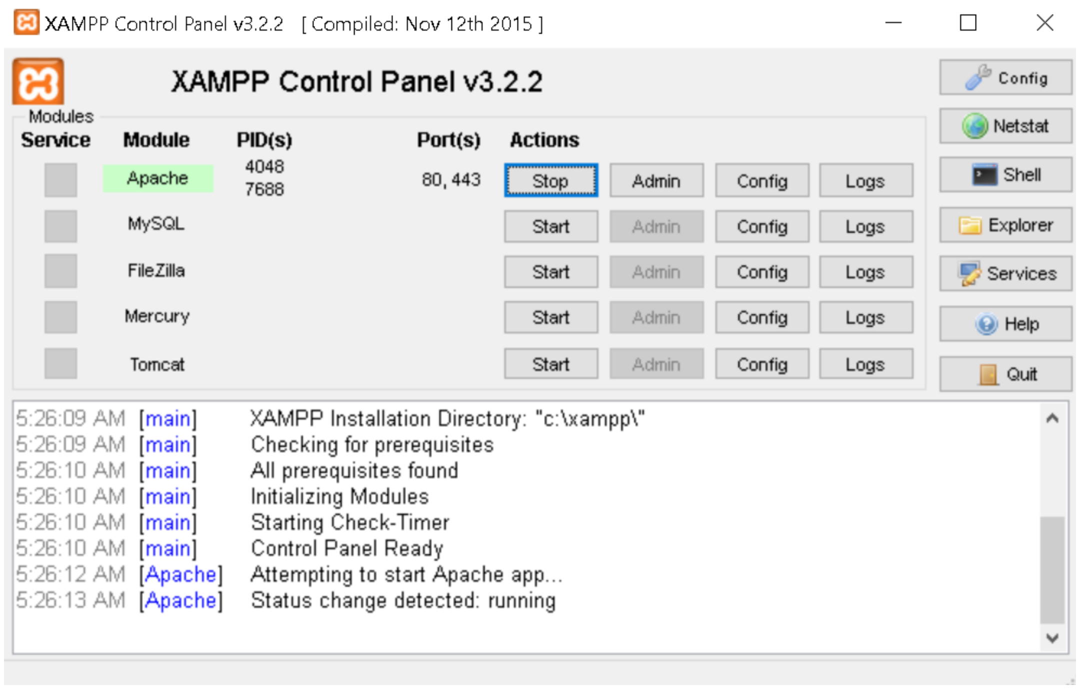
Originally, my design for the lock and page of the website only have a label that display the status of lock and light and a toggle button that send a get request to the hub with the id of the button which the hub can process and act accordingly to the id of the request. For example: if the id is “a1”, the esp8266 will toggle light. If the id is “b2, the esp8266 will toggle lock. Then hub will send a post request back to the website and display the new lock/light state on the label.

There are several issues with the original design, the most obvious one is the hub missing toggling request from the website. To fix this issue, I decide to implement two JSON files which act as the host for the lock and light status on the web interface. When accessing a light or lock page, the user has to option to turn the light on/off or turn the lock locked/unlocked from pressing a button. By pressing these button, the website will send a get request to itself which overwrite the JSON file according to the request. The esp8266 will constantly be parsing the content of the JSON files and switch the lock and light state according to them. The downside to this fix is it will take a bit more time for the hub to switch the lock/light state from the website interface.

Another change I make is programming the esp8266 and the Arduino mega separately, so I can program the esp8266 with out using AT commands. The project can totally work if we program the esp8266 directly from the Arduino mega using AT command, but it is a very complicated process considering the scope of what we want to do. Programming the esp8266 and Arduino mega separately is much easier since there are built in library that help with programming the esp8266. However, this is when I find out that sending messages from the Arduino mega to the esp8266 is problematic, especially when we are not sending an AT command. This result in the esp8266 cannot parse the messages we send from the Arduino mega.

5.4 Website Interface and Code:

To locally host the website, we use the apache module on XAMPP.

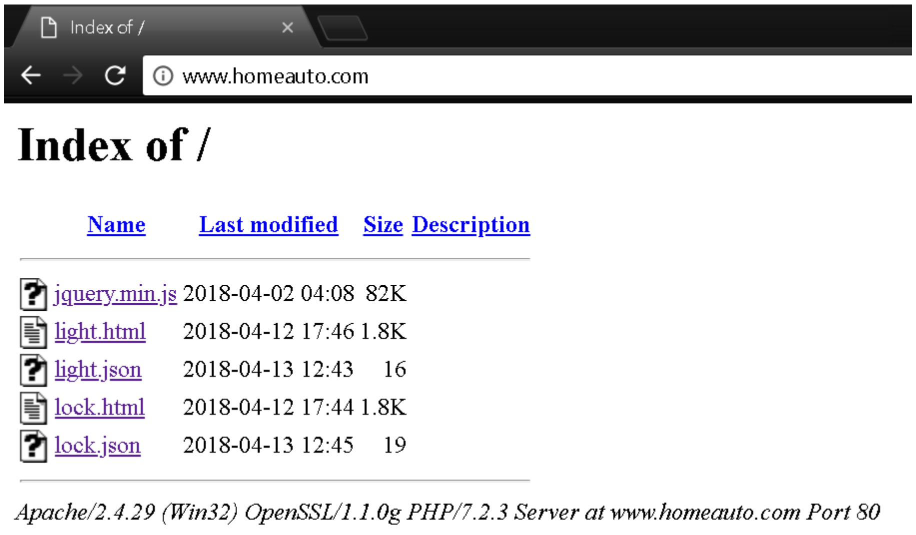

XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Config
Netstat
Shell
Explorer
Services
Help
Quit
Modules
Stop
Start
Admin
Config
Logs

Service	Module	PID(s)	Port(s)	Actions
	Apache	4048 7688	80, 443	Stop Admin Config Logs
	MySQL			Start Admin Config Logs
	FileZilla			Start Admin Config Logs
	Mercury			Start Admin Config Logs
	Tomcat			Start Admin Config Logs

```
5:26:09 AM  [main]     XAMPP Installation Directory: "c:\xampp\"
5:26:09 AM  [main]     Checking for prerequisites
5:26:10 AM  [main]     All prerequisites found
5:26:10 AM  [main]     Initializing Modules
5:26:10 AM  [main]     Starting Check-Timer
5:26:10 AM  [main]     Control Panel Ready
5:26:12 AM  [Apache]   Attempting to start Apache app...
5:26:13 AM  [Apache]   Status change detected: running
```

Figure 5.1: XAMPP Control Panel

After starting the XAMPP server, typing www.homeauto.com will take you to the main page of the webserver. Normally, you will need to type IP address and port number of the local host to be able to access the website, but I have mask the IP address using an URL so that the user can access it with ease as long as they are in the same domain as the local host.



Index of /

Name	Last modified	Size	Description
jquery.min.js	2018-04-02 04:08	82K	
light.html	2018-04-12 17:46	1.8K	
light.json	2018-04-13 12:43	16	
lock.html	2018-04-12 17:44	1.8K	
lock.json	2018-04-13 12:45	19	

Apache/2.4.29 (Win32) OpenSSL/1.1.0g PHP/7.2.3 Server at www.homeauto.com Port 80

Figure 5.2: Main page of the Website

Normally I would have a index.html file which determine what the main page will look like. But for testing purpose I need to be able to access the json files directly. The jquery.min.js is a library for the methods I am using in the html code.

When clicking on the light.html file, the user will be taken to this page.

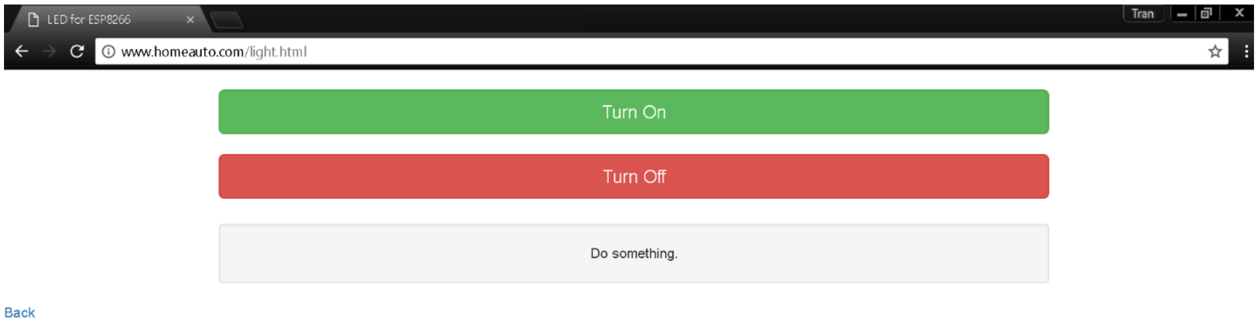


Figure 5.3: Light Page of the Website

Pressing Turn On and Turn Off will change the contents of the light.json file for example:

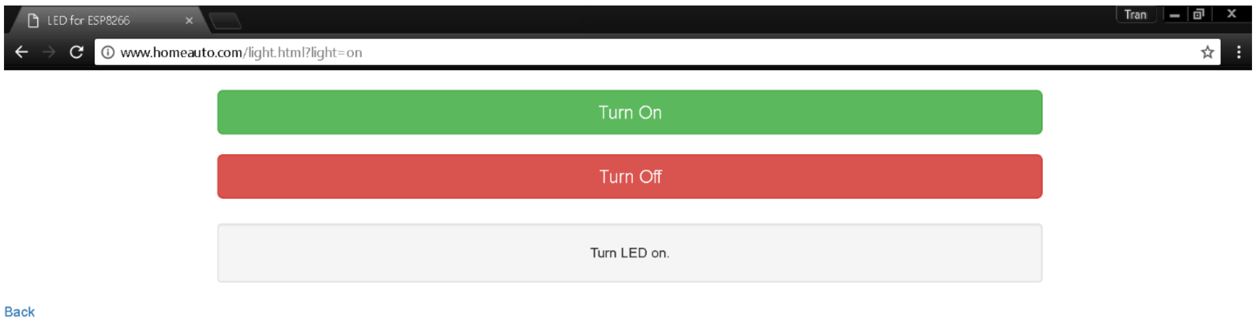
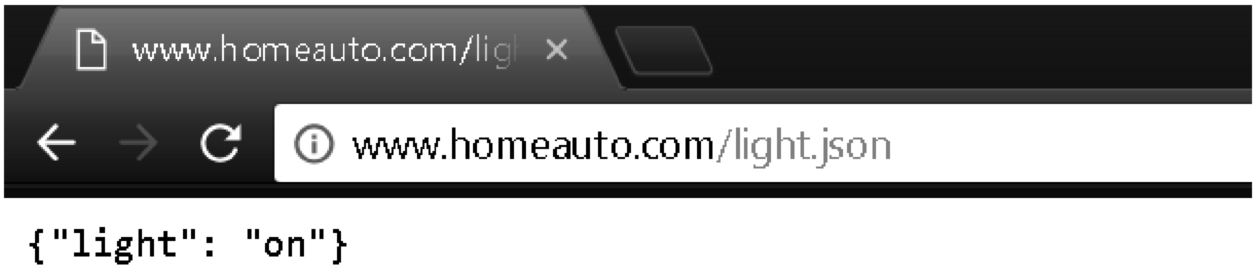


Figure 5.4: Light Page of the Website (After pressing Turn On)



```
{"light": "on"}
```

Figure 5.5 -- Content of light.json file after pressing Turn On

Furthermore, the user can also change the content of lock.json file from the lock page.

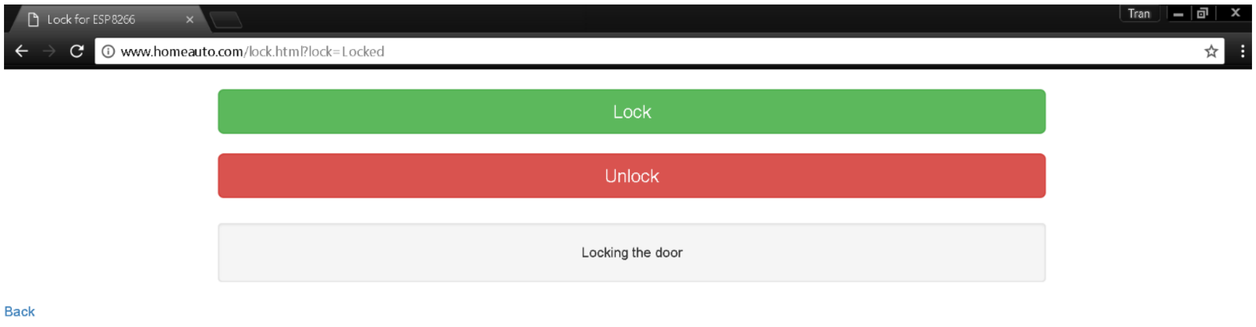
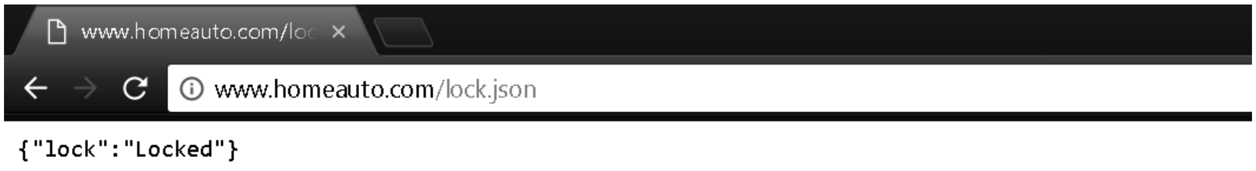


Figure 5.6: Lock Page of the website (after pressing Lock)



```
{"lock":"Locked"}
```

Figure 5.7 -- Content of lock.json file after pressing Lock Button

5.4.1 Here is the content of light.html file:

```
<?php
error_reporting(0);
//This section of the code check for and parse post and get request
if ($_SERVER['REQUEST_METHOD'] === 'POST') //check if there is incoming post request from
Hub
        {$light = $_POST['light'];}
else
        {$light =$_GET['light'];}  //if there is no post request, parse get request

/*This section of the code overwrite the json file depend on the contents
of receiving post and get request */
if($light == "on") {
  $file = fopen("light.json", "w") or die("can't open file");
  fwrite($file, '{"light": "on"}');
```

```
 fclose($file);
}
else if ($light == "off") {
 $file = fopen("light.json", "w") or die("can't open file");
 fwrite($file, '{"light": "off"}');
 fclose($file);
}
?>

<html>
 <head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <title>LED for ESP8266</title>

  <script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-
awesome.min.css">

 </head>
 <body>
  <div class="row" style="margin-top: 20px;">
   <div class="col-md-8 col-md-offset-2">
//This section will send a get request from the website to itself
    <a href="?light=on" class="btn btn-success btn-block btn-lg">Turn On</a>
    <br />
    <a href="?light=off" class="led btn btn-danger btn-block btn-lg">Turn Off</a>
    <br />
    <div class="light-status well" style="margin-top: 5px; text-align:center">
     <?php
       //This section label the status of the light depend on the value of $light variable
      if($light=="on") {
       echo("Turn LED on.");
      }
      else if ($light=="off") {
       echo("Turn LED off.");
      }
      else {
            echo ("Do something.");
           }
          ?>
        </div>
      </div>
    </div>
            //The backbutton take the user back to the main page
<a href="http://www.homeauto.com/">Back</a>
  </body>
</html>
```

5.4.2 Here is the content of lock.html file:

```
<?php
error_reporting(0);
//This section of the code check for and parse post and get request
if ($_SERVER['REQUEST_METHOD'] === 'POST') //Check if there is incoming post request from
Hub
        {$lock = $_POST['lock'];}
else
        {$lock = $_GET['lock'];} //if there is no post request, parse get request

/*This section of the code overwrite the json file depend on the contents
of receiving post and get request */
if($lock == "Locked") {
  $file = fopen("lock.json", "w") or die("can't open file");
  fwrite($file, '{"lock":"Locked"}');
  fclose($file);
}
else if ($lock == "Unlocked") {
  $file = fopen("lock.json", "w") or die("can't open file");
  fwrite($file, '{"lock":"Unlocked"}');
  fclose($file);
}
?>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Lock for ESP8266</title>

    <script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css"
rel="stylesheet">
  <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-
awesome.min.css">

 </head>
 <body>
  <div class="row" style="margin-top: 20px;">
   <div class="col-md-8 col-md-offset-2">

         //This section will send a get request from the website to itself
    <a href="?lock=Locked" class="btn btn-success btn-block btn-lg">Lock</a>
    <br />
    <a href="?lock=Unlocked" class="led btn btn-danger btn-block btn-lg">Unlock</a>
    <br />
    <div class="lock-status well" style="margin-top: 5px; text-align:center">
     <?php
               //This section label the status of the light depend on the value of $lock
variable
      if($lock=="Locked") {
       echo("Locking the door");
      }
      else if ($lock=="Unlocked") {
       echo("Unlocking the door");
      }
      else {
       echo ("Do something.");
      }
     ?>
    </div>
   </div>
  </div>
        //The backbutton take the user back to the main page
<a href="http://www.homeauto.com/">Back</a>
 </body>
</html>
```

5.5 ESP8266 Functionality and Code:

The esp8266 handle the communication between the website and the Arduino mega. This include matching the lock and light status on the Arduino mega to the website and vice versa. The esp8266 will send get request to the JSON files and parse the response for the lock and light status on the website. The esp8266 can also send a post request to the html files on the website which prompt it to change the lock/light status on the JSON file. Here are the codes of the esp8266:

5.5.1 Setup and loop code:

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
const char* ssid     = "TZUMI8760"; // my hotspot SSID
const char* password = "7x3R&559"; //my hotspot password
String lightschedule[10] ;
String lockschedule[10];
String lightpath = "/light.json";  //Path to light.json
String lockpath = "/lock.json";  //Path to lock.json
int path = 0; // 0 to access light site and 1 to access lock site
WiFiClient client;
const char* host     = "192.168.137.1"; // Your domain

IPAddress ip;
ESP8266WebServer server(80);
String lightstatus = "off";
String lockstatus = "Unlocked";

void setup() {

  Serial.begin(115200);

  delay(100);
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password); //begin connection to wifi network
  int wifi_ctr = 0;
  while (WiFi.status() != WL_CONNECTED) {//connect to wifi in a while loop
    delay(500);
    //Serial.print(".");
  }
  Serial.println("WiFi connected");
  delay(5000);
  ip = WiFi.localIP();
  Serial.print("IP address: ");
  Serial.println(ip); //Print IP address
}
```

```
void loop() {
  Serial.print("connecting to ");
  Serial.println(host);
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) { // Connect to the webserver
    Serial.println("connection failed");
    return;
  }

  /*if(Serial.available()){
    receiveserial(); This section of code check for incoming message from the Hub but since we
cannot parse the incoming messages, I decide to comment it out.
     loop();
  }*/

  /*This section of code decide to call handlelock() or handlelight() according to the path
variable. This section is necessary since the lock need to reconnect to the client before sending
another post or get request.
   */
  else{
    if(path == 0) {
      handlelight();
      path = 1;
      }
    else {
      handlelock();
      path = 0;
    }
  }
  /* This block of code send a message with the new lock and light status to the arduino mega
    */
    String output = "";
    output += "Status:\nLight1=";
    output += lightstatus;
    output +="\nLock1=";
    output += lockstatus;
    output += "\n\r";
    Serial.println(output);
 }
```

5.5.2 handlelight() code:

```
/*Name: handlelight()
 Functionality: Send a get request to the light.json file of the website and parse the content of
 that json file. Then change the lightstatus variable to match the parsed content.
 */
void handlelight() {
 client.print(String("GET ") + lightpath + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Connection: keep-alive\r\n\r\n");
 delay(500); // wait for server to respond
 // read response
 String section="header";
 while(client.available()){
  String line = client.readStringUntil('\r');
  //Serial.print(line);
  //we'll parse the HTML body here
  if (section=="header") { // headers..
   //Serial.print(".");
   if (line=="\n") { // skips the empty space at the beginning
    section="json";
   }
  }
  else if (section=="json") {
   section="ignore";
   String result = line.substring(1);
   // Parse JSON
   int size = result.length() + 1;
   char json[size];
   result.toCharArray(json, size);
   StaticJsonBuffer<200> jsonBuffer;
   JsonObject& json_parsed = jsonBuffer.parseObject(json);
   if (!json_parsed.success())
   {
    Serial.println("parseObject() failed");
    return;
   }
   // Make the decision to turn off or on the LED
   if (strcmp(json_parsed["light"], "on") == 0) {
    lightstatus = "on";
   }
   else {
    lightstatus = "off";
   }
  }
  }
}
```

5.5.3 updatelight() code:

```
/*Name: updatelight()
 Functionality: Send a post request with the value of lightstatus to the light.html file of the
website
 so that it can change the contents of light.json according to the post request
 */

void updatelight(){

  HTTPClient http; //Declare object of class HTTPClient
  String request ="light=";
  String destination = host;
  request += lightstatus;
  destination += "http://192.168.137.1:80/light.html";
  http.begin(destination); //Specify request destination
  Serial.println("Connected to HTTPClient");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded"); //Specify content-
type header
  int httpCode=http.POST(request); //Send the request
  String payload = http.getString(); //Get the response payload
  //Serial.println(httpCode); //Print HTTP return code
  //Serial.println(payload); //Print request response payload
  http.end(); //Close connection
}
}
```

5.5.4 handlelock() code:

```
/*Name: handlelock()
 Functionality: Send a get request to the lock.json file of the website and parse the content of
 that json file. Then change the lockstatus variable to match the parsed content.
 */

void handlelock() {

  client.print(String("GET ") + lockpath + " HTTP/1.1\r\n" +
              "Host: " + host + "\r\n" +
              "Connection: keep-alive\r\n\r\n");

  delay(500); // wait for server to respond
  // read response
  String section="header";
  while(client.available()){
    String line = client.readStringUntil('\r');
    //Serial.println(line);
    //we'll parse the HTML body here

    if (section=="header") { // headers..
     //Serial.print(".");
     if (line=="\n") { // skips the empty space at the beginning
       section="json";
     }
    }

    else if (section=="json") {
     section="ignore";
     String result = line.substring(1);
     // Parse JSON
     int size = result.length() + 1;
     char json[size];
     result.toCharArray(json, size);
     StaticJsonBuffer<200> jsonBuffer;
     JsonObject& json_parsed = jsonBuffer.parseObject(json);

     if (!json_parsed.success())
     {
       Serial.println("parseObject() failed");
       return;
     }

     // Make the decision to turn lock or unlock
   if (strcmp(json_parsed["lock"], "Locked") == 0) {
       lockstatus = "Locked";
     }

     else {
       lockstatus = "Unlocked";
     }

    }
  }
  }
```

5.5.4 receiveserial() code:

```
/*Name: receiveserial()
 Functionality: Parse incoming message from the Arduino mega, if the parsed message has
"lock=" or "light="change lockstatus or lightstatus according to the message and call
updatelight() or updatelock() to update the json file.
 */

void receiveserial(){//Update the website when mega send a message
String feedback = Serial.readStringUntil('='); //read serial to see if there is "light=" or "lock="
String value =  Serial.readStringUntil('X');  //X is terminating  character value should be "on" or
"off" or "locked" or "unlocked"
  if(!feedback.equals("lock")&&!feedback.equals("light"))
  {return;} //return if light or lock is not part of the serial message
  Serial.println(feedback);
  Serial.println(value);  //this block test the serial communication

if(feedback.equals("lock")){ //determine to change lockstatus or lightstatus depend on the
messager
  lockstatus = value;
  updatelock();
}

else if(feedback.equals("light")){
  lightstatus = value;
  updatelight();
  }

}
```

5.6 Test Plan and Results

Since there are a lot of changes to our original design, notably the inclusion of on and off buttons on light page and lock and unlock buttons on lock page instead of a toggle button on each page. We also fail to implement the schedule functionality to our final product.

Setting up the xampp server and type in the url www.homeauto.com take you to the main page, from the main page the user can access light page from light.html or lock page from lock.html file.

From the light webpage, pressing on turn the LED on when it is off, the label of light status on the central hub also displays "on". Pressing off turn the LED off when it is on, the label of light status on the central hub also display "off". There is no change to the system when pressing on while the LED is on or off while the LED is off.

From the lock webpage, pressing lock turns the stepper motor if its current state is unlocked. This also update the lock status label on the central hub to display "Locked". Pressing unlock turns the stepper motor if its current state is locked. This also update the lock status label on the central hub to display "Unlocked". There is no change to the system when pressing lock while the lock state is locked or unlock while the lock state is unlocked.

Pressing any button on the touch screen do not update the new lock or light status to the website interface because there is no bi-directional communication between the central hub and the esp8266. The esp8266 can send message to the central hub but it cannot parse messages from the. Therefore, updating data on the website from the central hub does not work.

Since we do not include the schedule functionality to our final design, the user cannot enter a time in 24 hours in a text field in the lock and light page to automatically lock or turn off. Consequently, there is also no implementation of a delete button for each existing time in the schedule where the user has set up.

The user also cannot enter a new time or delete an existing time from the schedule when using the hub either. Therefore, the hub cannot update the new schedule to the website. When entering the lock or light page from the hub, there is no schedule being display and edit nor a text field that update the existing schedule.

5.7 Future Changes

To start off, I would use a microcontroller with built in WIFI instead of using an Arduino mega as a central hub and the esp8266 as a WIFI chip. This would solve our problem of sending messages from the hub to the website since the messages do not have to go through an intermediate WIFI chip. We would not have to worry about the esp8266 not being able to parse an incoming message from the hub. This will also help us implement the schedule functionality as we can easily send a post request from the hub to the website and vice versa with all the schedule variables instead of sending the schedules to the esp8266 and worry about it not parsing the message properly.

I would also want to use a microcontroller that has multi-threading capability. Since there are a lot of different pages that I want to parse and update at all time. Because of this, it takes around 10-15 seconds for the system to switch lock or light stage after a button is press on the website. Having multi-threading to keep track of every pages individually will help the system run a lot faster especially when we expand it to include more locks and lights. Multi-threading will be something I will look into going forward.

6 – Conclusion

For as much as our group botched and failed in the creation of the Light and Lock Automation system, we learned an equivalent amount. We learned important lessons on coding in different languages, project planning and adherence to schedules, construction of breadboarded circuits, and efficient teamwork. While the Light and Lock Automation system may not have ended as the product we had hoped to create, it did serve as an valuable lesson on necessary courses of action and pitfalls to avoid in future project work.