Sensor Activated Home Hub Curtains

ECE 445 Final Report

By

Daniel Chiu

Anusha Anumakonda

Rachel Fu

Team No. 53

TA: Bonhyun Ku

04 May 2021

Abstract

The primary goal of our project is to automate the process of opening and closing curtains in order to aid the regulation of a room's temperature. Our device interfaces with a mobile app through a series of get and post requests to get useful user preference information. The user preference information includes the decision state, alarm time, current time, user curtain state, and goal temperature. Our Sensor Activated Home Hub Curtains then uses this data to decide whether to keep the curtains open, closed or halfway open. The physical movement of the curtains is controlled by two stepper motors.

In the manual decision state, the device uses the data from a thermal and photo sensor to identify how much sunlight and heat to allow into the room. In the automatic decision state, the user's preference on curtain position overrules all decisions made by the device.

Table of Contents

1. Introduction

1.1 Objective:

Calls to help mitigate climate change is a well known issue, but many do not know how effective simply opening and closing curtains can do to help conserve energy. Simply "closing the curtains during the winter helps reduce up to 10 percent in heat loss from a warm room [1]." However, as effective as this simple solution is, the actual process of doing so is rarely undertaken. In addition, waking up through sunlight exposure is a very effective way to improve a person's circadian rhythm and has been proven to help to help "increases in the level of the hormone serotonin, which is important to sleep [2]" as well as reducing the stress hormone cortisol.

Our device shown in Figure 1 will aid the regulation of temperature and light in a room through automated curtains, thermostat and light control. The decision to open and close a set of curtains depending on the data input of several sensors. These sensors will monitor temperature, and sunlight. Depending on the readings front the sensors, our device will fully close the curtains, close the curtains halfways, and open the curtains, in order to best maintain the temperature and light preferences of the user.

Figure 1*: Finalized Prototype of Sensor Activated Home Hub Curtain System*

1.2 Background:

Similar products that aim to control curtains in residential homes have been limited to products like the Aqara curtain controller system that focuses on helping users "open or close curtains via [their] smartphone, cube controller, or wireless switch anywhere [3]." These products do not contain any sensors that would analyze the different factors that would minimize the need for centralized heating.

In addition, there have been a few products marketed for Industrial uses such as Curtains for Barns curtain system that does analyze the "temperature, humidity, wind direction and speed [4]." However, these are more focused on air quality for their animals and have not focused on more residential applications. Thus, while there are products that may be a bit similar in concept, our Smart Curtains device is the first of its kind to use automated curtain control to help the average person conserve energy.

1.3 High Level Requirements:

- The device should exhibit 0.2 N-m torque for a curtain rod with diameter of ¾ inches through the use of each motor, showing that the device is capable of opening and closing curtains that are up to 4 lbs.

- The device must have a reliable connection to the data server being hosted. In order to test such, the device must be able to make connections to the server at any time with 90% success rate. The server must have an uptime of at least 90% of the day.

- Using sensor data, such as lighting levels and temperature readings, the device should autonomously decide what state to put the curtains in (open, closed, partially opened, etc). Given the same environmental information, the decision the device makes should align with the user's decision at least 75% of the time (i.e Given 4 scenarios, 3 of the device's choice is the same as the user's).

2. Design

Our design includes six modules: the power module to power the PCB and motors, the control module to control sensor and motor activities, the sensor module to get light and temperature data, the output module to adjust the curtains, the server module to set up network connection, and the user interface module to allow users to control the system through a mobile application. The modules and how they are connected through power or data are shown in Figure 2, and the circuit schematic and the PCB layout can be found in Appendix C.

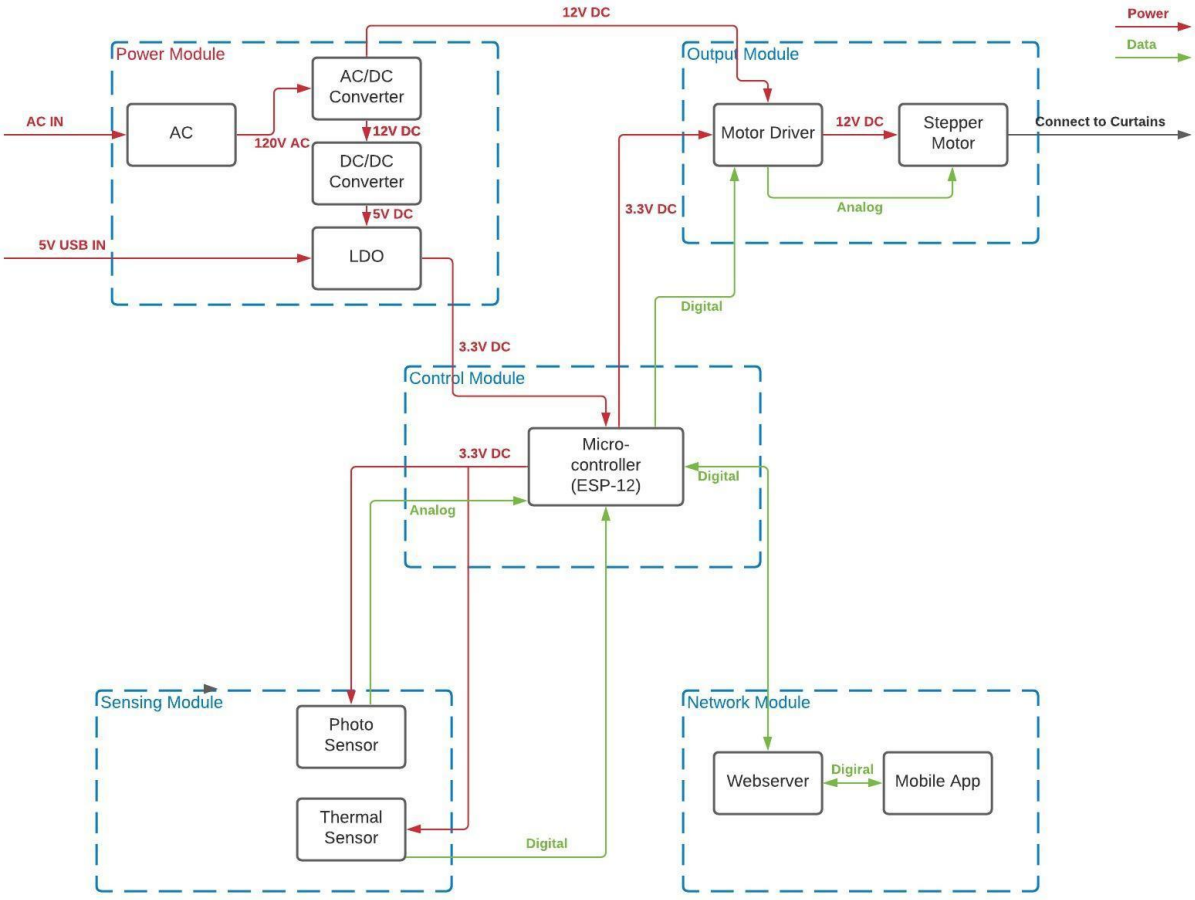

***Figure 2**: Block Diagram*

2.1 Power Module

The power module includes an AC/DC converter and a DC/DC step down module. The AC/DC converter can be plugged into the power outlet on walls and convert the 120V input to 12V output. It is chosen because 120V to 12V converters are common and 12V satisfies the operating voltage required by the motor driver. The 12V output from the converter goes to the stepper

motor driver, which uses this voltage to power the stepper motors. Also, the 12V output goes into the DC/DC step down module, and is converted to 5V output. The output of the DC/DC step down module is chosen to be 5V because we want this voltage to match the voltage from the USB connection, so that the LDO can convert it to 3.3V, which is the required input voltage for the ESP12E chip. Since the DC/DC step down module and the LDO are soldered on the PCB, the AC/DC converter is connected to the PCB through wire to board connectors.

2.2 Control Module

Our control module includes a ESP12E chip and a stepper motor driver. At first, we wanted to use Atmega328P as our microcontroller, because it can be coded with the Arduino IDE and it's compatible with the sensors and motor drivers. However, because Atmega328P doesn't support wireless communication protocols and we need to connect to the web server for our project, we decided to use ESP12E. We thought about using ESP12E as a Wifi module and Atmega328P as our microcontroller. However, ESP12E and Atmega328P don't have the same clock speed [5], so to use them together, we would have to interface between them. To save that trouble, we decided to use ESP12E directly as our microcontroller, ESP12E, especially when we found out ESP12E could also support all the functionalities we wanted.

The ESP12E chip takes an input of 3.3V and gives a 3.3V output, which goes into the sensors and the logic input of the motor driver. The sensors are connected to and controlled by the ESP12E chip through I2C communication protocol, which allows the two sensors to be connected to the same output pins. The stepper motor driver is also powered and connected to the ESP12E chip. The ESP12E chip is also connected to a USB to UART converter, so that it can be flashed and it is programmable with a computer.

For the motor driver, at first we wanted to use Sparkfun Easy Driver for testing, and we planned to switch to L293D for the PCB to reduce the size. However, we found out that L293D wasn't ideal for stepper motors, as it couldn't limit the current to protect the motor and use too many output pins [6]. Thus, we decided to use the A4988 motor driver for our final design. Another reason to choose A4988 over Easy Driver is that A4988 can provide an output current to be as much as 2A [7], which is ideal for our stepper motor, whereas Easy Driver has a maximum output current of 750mA [8].

The motor driver is connected to the 12V DC output from the AC/DC converter. It's also connected to and controlled by the ESP12E chip as shown in Figure 3.

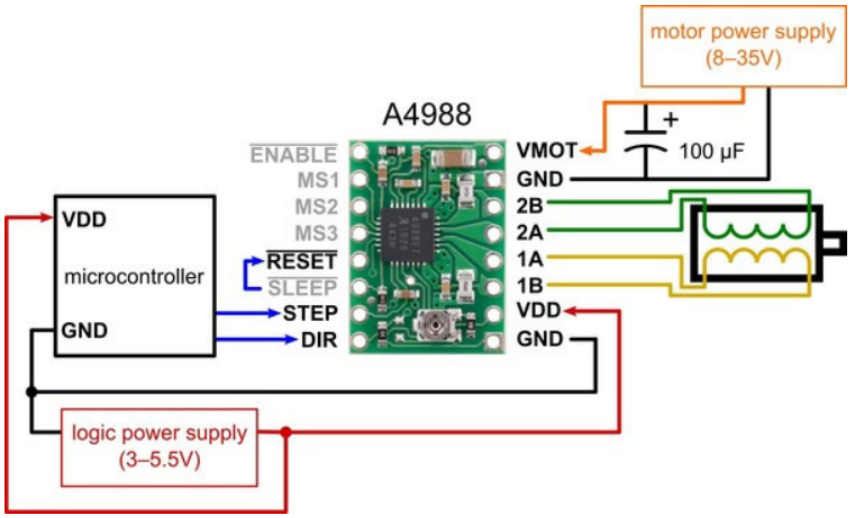

***Figure 3**: Motor Driver Connection [7]*

2.3 Sensor Module

We have a photo sensor (BH1750) and a thermal sensor (MLX90614) for our project. The reasons why we chose those two sensors are that both sensors have high accuracies and both of them use I2C communication protocol. The thermal sensor features contactless temperature sensing, which is ideal for our project, as the goal of it is to measure the room temperature. The photo sensor has a wide sensing range, which is needed for our project, because the sensor might receive from 0.0011 lux at night to 107527 lux from direct sunlight.

One concern about the light sensor is that it doesn't differentiate artificial and natural light. Since a very dark day can have as low as 107 lux and an average home has 150 lux [9] and we want to make sure the microcontroller makes correct decisions about if it is day time or night time, we need to make sure that the light sensor is placed close to the window, preferably facing the window.

Both sensors are connected to the ESP12E chip, and powered by its 3.3V output. The two SCL pins of the sensors are connected together, and the two SDA pins are connected together, to reduce the number of output pins of the microcontroller used.

2.4 Output Module

Our output module includes 2 Nema 17 stepper motors. We chose Nema 17 stepper motors because typically a Nema 17 motor can provide a maximum torque of 0.3N-m, which is 0.1N-m larger than the torque needed for the project. To decide the torque required, we did the following calculations. First we calculated the force needed to pull the curtains. The friction coefficient of

nylon on dry steel is around .40. The average heavy curtain to retain heat weighs in at around 2kg (4lb). Thus to the minimum force to pull the whole curtain is:

$$F = (m * g) * \mu$$

$$F = (2kg * g) *.4 = 7.848\,N$$

Then we calculated the torque required. With a curtain rod with a 1" (.0254m) circumference, we can calculate the torque required by the motor to be:

$$\tau = F * m$$

$$\tau = 7.848N * 0.0254\text{m} = .1993\ \text{N-m}$$

As such, with the assumption we are moving the whole mass all the time, the motor must operate at a capacity of at least .2 N-m of torque in order to move our curtains .

Ideally, we would have the stepper motors connected to a parallel board designed for stepper motors, so that they could be controlled by the microcontroller at the same time. They would be powered by 12V DC from the output of the AC/DC converter. However, although the parallel board functioned well, we couldn't set it up for our final prototype and the demonstration due to mechanical problems explained in section 5.2. Thus, we only set up and used 1 stepper motor in the end.

2.5 Server Module

2.5.1 Network Original Design

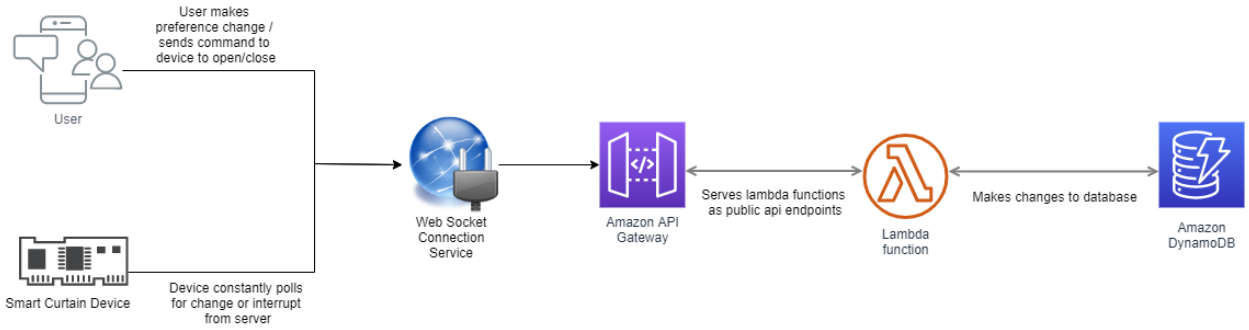

***Figure 4**: Original intended Network Diagram with websocket implementation*

In our original design for the network, it was planned to integrate a websocket over the serverless web server architecture. When the device connects to our server, it will be able to make a TCP connection that allows for bidirectional communication from the device and the server. Thus, when an update is made to the server, the server can tell the device directly a change has been made. Vice versa, if the device detects a change in its sensors, it can push that data to the server directly as well.

2.5.2 Final Network Design

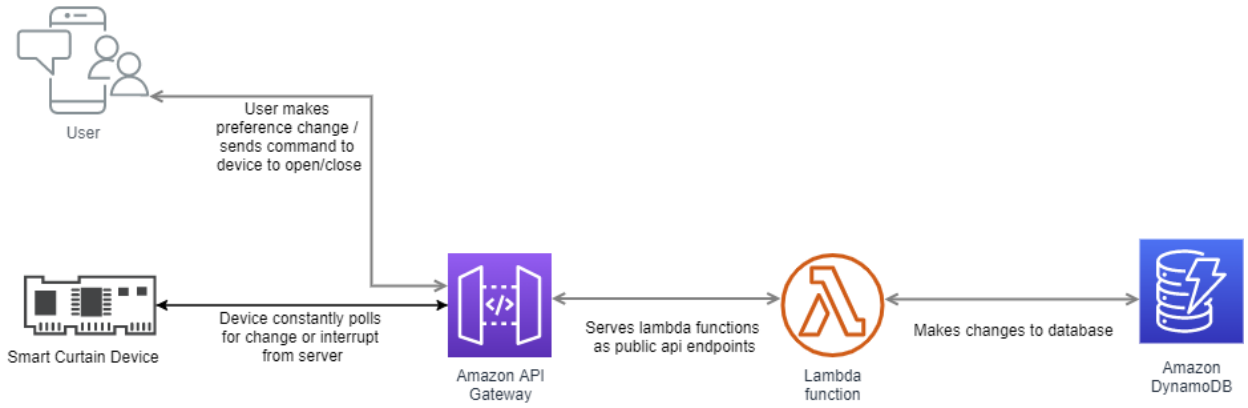

***Figure 5**: Diagram of actual implementation of network. Note removed web socket functionality.*

Instead, due to time constraints, we opted for a traditional REST api web server with data being obtained through GET request, and data updates being pushed through POST request. The difference in this implementation is that the device cannot make a direct TCP connection to the server, and thus the server is not able to send notifications of changes directly to the device. Thus, the device must now poll the server for changes.

2.6 User Interface Module

The user interface (UI) module is the main platform that the user will be able to dictate their preferences to the curtain control system. The UI is created as a React Native application, allowing for compatibility with IOS, android, and on a web browser. The UI allows the user to set either a device autonomous mode, or set specific preferences for the device to follow. Upon submission, the application sends the request to the server, which the device will then poll for a change and make an action accordingly. The UI also polls the server for changes the device has sent, allowing the UI to also display the data collected by the device's sensors in real time.

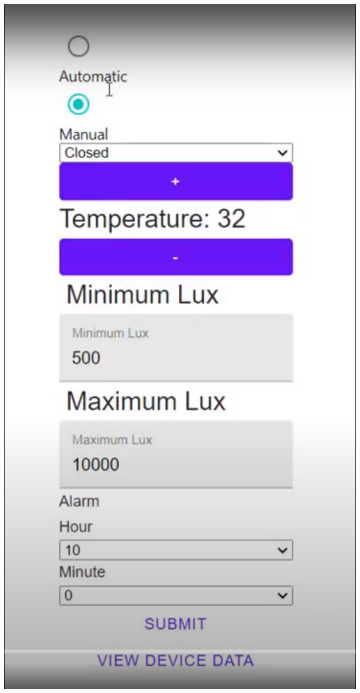

***Figure 6**: User Interface*

2.7 State Machine

Our project is designed to have three different modes: user control mode, device control mode and alarm control mode. Under user control mode, the user will be able to completely override the device's decision. As shown in Figure 7, from the start state, following the red arrows, if the user's preference is closing the curtains, the curtains will be closed. If the user's preference is opening the curtains, the curtains will be opened. If the user's preference is opening the curtains to halfways, the curtains will be opened halfways.

If the user chooses no preference, the curtain states will be decided by sensor ratings, which is our device control mode. In this mode, the photo sensor decides if it is day time or night time. If it is day time, the state of the curtains will be decided by the thermal sensor. If the temperature sensed is larger than the goal temperature set by the user, but very close to the goal temperature, the curtains will be opened halfways. If the temperature sensed is much larger than the goal temperature, the curtains will be closed for the room to cool down. If the temperature sensed is smaller than the goal temperature, the curtains will be opened to allow sunlight to heat the room up. In the previous state, if the photo sensor decides that it is night time, the curtains will be closed.

The user can choose whether or not to activate the alarm mode. If the alarm mode is activated, then thirty minutes before the user defined alarm time, the curtains will be opened so that the sunlight can be a natural alarm for the user. If the alarm mode is not activated, the curtains will remain closed.

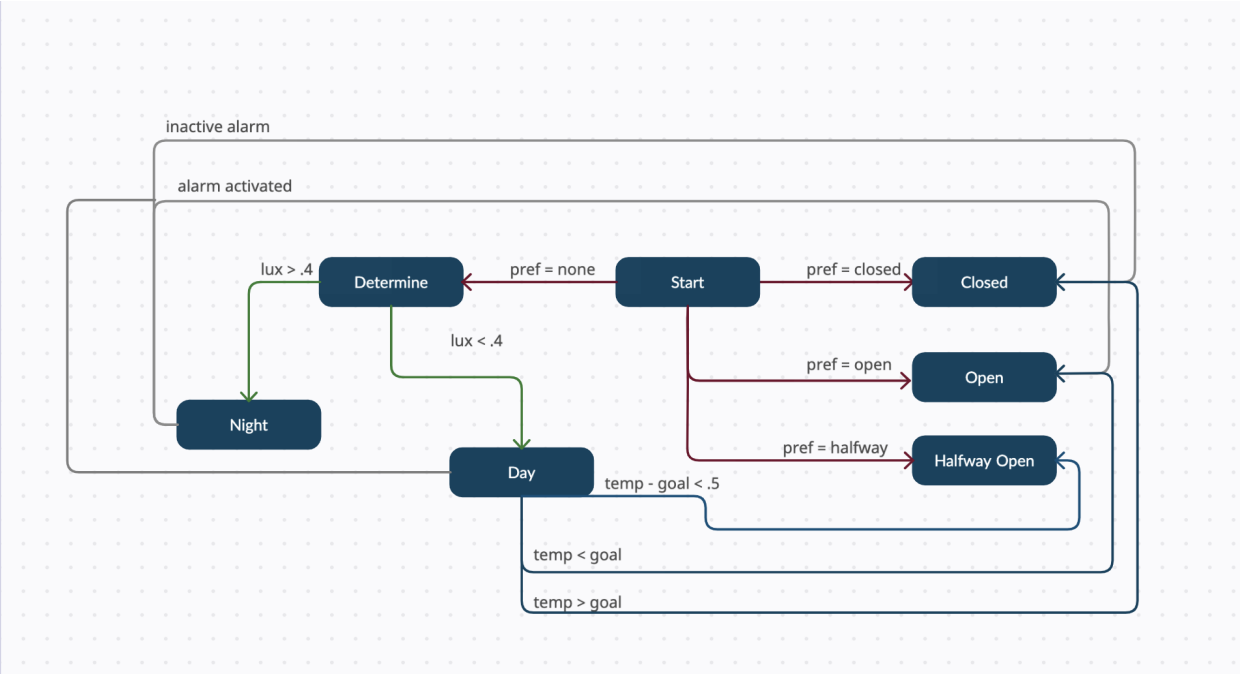

***Figure 7**: State Machine*

3. Verification

For the simplicity of the testing and the reliability of the results, we did different kinds of verification for different modules. For some modules, we used component-based verification and for some modules, we used module-based verification. We also combined the verification of the control module and the output module together, as the motor in the output module is driven by the controller in the control module. A specific list of component-based requirements and verification tables can be found in Appendix B.

3.1 Power Module

We first did component-based verification for the power module. For each component, we used adjustable power supply to provide the designed input, and used a multimeter to measure the output voltage. The AC/DC converter and LDO have set output voltages so we just made sure that they were not faulted. The DC/DC step down module has adjustable output voltage, so we adjusted the potentiometer on it to achieve the desired 5V output voltage. After the component-based verification, we did module-based verification. To do the module-based verification, we first powered the PCB with the computer and then powered it with a power outlet on the wall. With lit LEDs on the AC/DC converter, ESP12E and DC/DC step down module, we knew that the system was powered. By testing the voltage of the 3.3V output pins of ESP12E and the output voltage of the LDO and the DC/DC step down module, we could conclude that the power module functioned as expected.

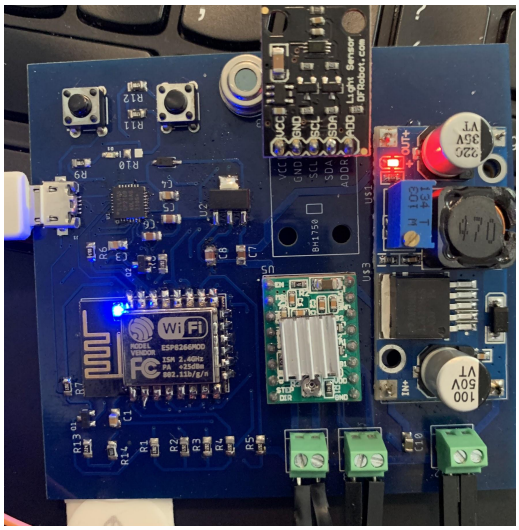

***Figure 8**: LEDs Light Up When the PCB Is Plugged In*

3.2 Control Module and Output Module

The ESP12E chip was tested first, and then the control module and the output module were tested together. To test the ESP12E chip, we simply connected it to Wifi. When it was connected, “check” was shown in the serial monitor.

***Figure 9**: Successful Wifi Connection*

To test the motor driver and the motors, We ran a piece of code for the stepper motors with ESP12E. We tested and verified that the motors could run in both directions with different speeds.

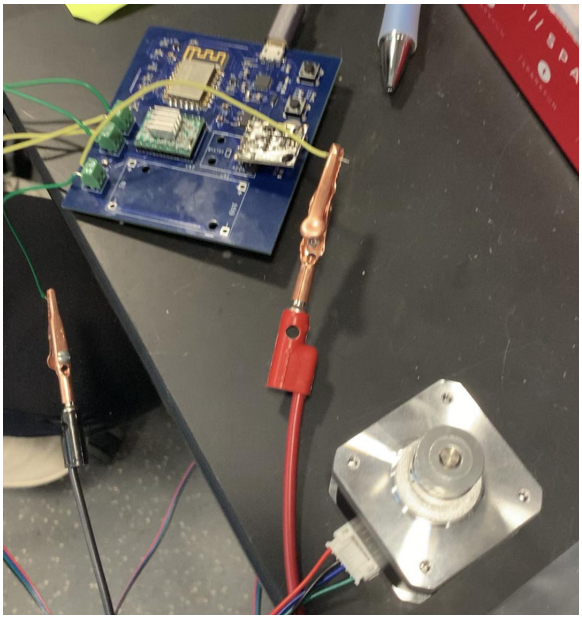

***Figure 10**: Motor and Motor Driver Testing*

3.3 Sensor Module

In order to verify that the sensors detected room conditions with adequate accuracy, we decided to perform modular based testing. I tested both the ambient and object accuracy for the MLX90614 Temperature Sensor to determine which attribute our device would base it’s curtain position on. As can be seen from Table 2, the ambient temperature had a lower error. Thus, we decided to base our automated decisions on the ambient reading from the Temperature Sensor.

For the BH1750 Photo Sensor, I tested the accuracy of different relevant conditions included in the documentation. As can be seen in Table 1, the experimental error was minimal with the exception of when the sensor was placed directly in front of the window. We used this data to determine the best position at which to place our PCB.

Condition	Expected Value	Actual Value	Experimental Error
Night	.001-.02	.0	0%
Sunny Indoor in Front of Window	100-1000	1343.33	34.33%
Sunny Indoor by Window Side	100-1000	122.74	0%
Sunny Indoor a Foot Away from the Window	100-1000	293.52	0%
Video Illumination	100-1000	133.33	0%

***Table 1**: Photo Sensor Data*

Condition	Expected Value	Actual Value	Experimental Error
Ice Cube - Object	30	23.95	20 %
Hot Pack - Object	110	109.07	0.85%
Room Temp - Amb	74	80	8.1%
Cold Room - Amb	60	68.59	14.32%

***Table 2**: Temperature Sensor Data*

3.5 Server Module

For testing our server module, we performed simple queries to the server and manually observed if the server was operational. The first test we had was a simple hello world query to the server to check for server uptime. If the server was working properly, then it would return a hello world response, as shown in Figure 11. If the server was offline for any reason, we would receive an

error instead. We could periodically check this throughout the day to ensure that the server is running properly.

***Figure 11**: Hello World Query*

The next verification we performed on the server was to check the business logic of the server, as shown in Figure 12. By simulating the types of query push and retrievals to and from the server, we could check if the server is able to maintain accuracy in data. In these tests, we again manually check by sending a request with specific data, then pulling that data immediately after to inspect if the data is correct.

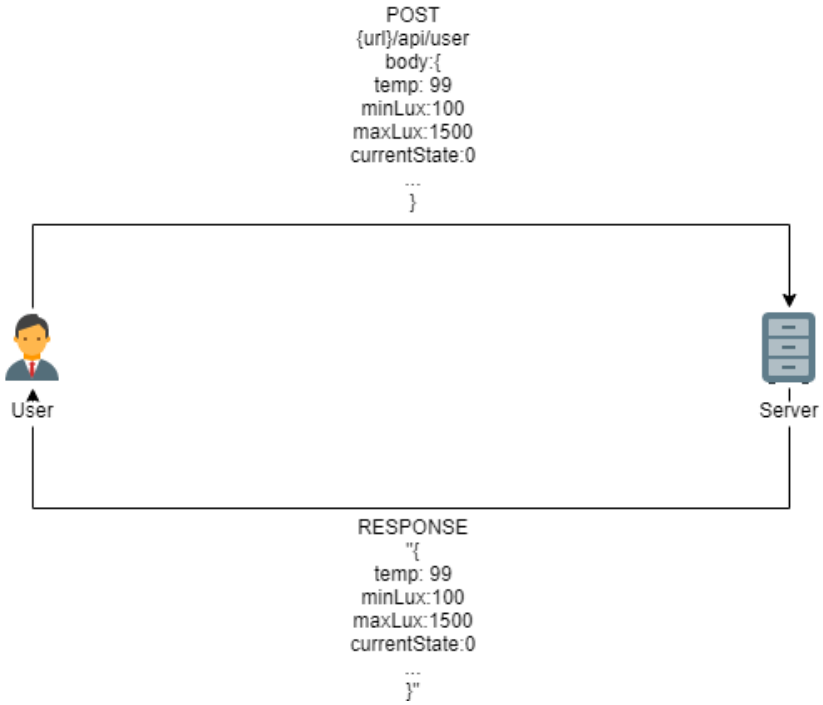

***Figure 12**: Business Logic Check*

These tests can be performed from all three of our data vectors: the UI, the device, and an outside source (such as a personal computer), as shown in Figure 13. In these tests, it also allows us to

not only diagnose the server itself, but the vectors in case one of them is having difficulties making contact with the server.

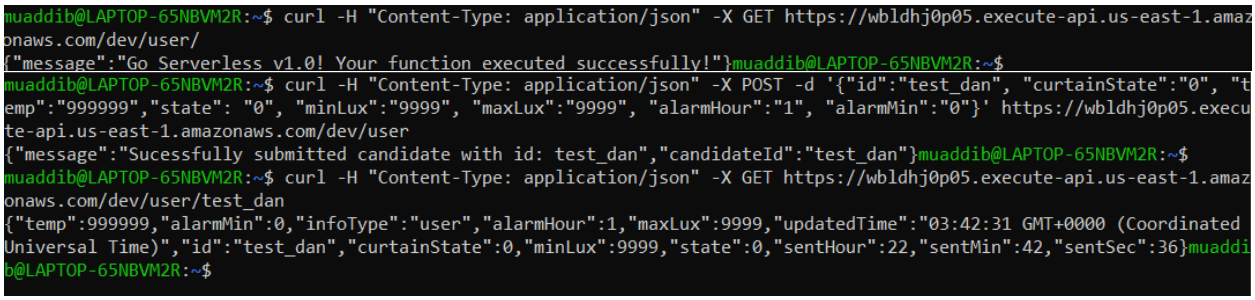

***Figure 13**: Tests of Data Vectors*

3.6 User Interface Module

The UI we tested manually once again by having specific instructions for what we wanted the device to do, and setting those preferences in the UI to see if they made the corresponding changes. For instance, if we wanted to close the curtains, we would manually set the curtains to close on the UI and observe if the device closed the curtains after. We tried this type of test across the temperature parameter, the alarm parameter, and the manual open or close parameter.

4. Cost and Schedule

4.1 Cost Analysis

4.1.1 Labor Cost

Assume hourly salary for a newly employed electrical engineer is $50/hr and assume an average workload is 10 hrs/week. With a total of 16 weeks of work this semester, the total labor cost would be:
$50 × 2.5 × 10 hrs/week × 16 weeks = $20000.

4.1.2 Cost of Parts

Name of Part	Product Number	Manufacturer	Quantity	Total Cost ($)	Link
ESP12E Chip	2491	Adafruit Industries LLC	2	13.9	ESP12E
USB to UART Converter	CP2109-A01-GM	Silicon Labs	3	8.18	USB to UART
DC/DC Step Down	VA-139-52	Valefod	6	10.99	DC/DC Step Down
Motor Driver	A4988	Aoicrie	5	7.99	Motor Driver
Stepper Motor	Nema 17HS4023	Usongshine	3	25.99	Stepper Motor
Motor Mount	PRO766221	Anndason	6	14.99	Motor Mount
AC/DC Adapter	B07VBBMZBJ	VeeDoo	1	8.99	AC/DC Adapter
Motor Parallel Board	37	Aokin	2	6.59	Parallel Board
Infrared	MLX90614	Melexis	1	29.95	Infrared

Thermometer					Therm
Photo sensor	BH1750	DFRobot	1	4.50	Photo sensor
BJT	SS8050-G	Comchip Company	7	1.61	BJT
Micro USB	MCR-B-S-RA-TSMT-NP-CS2-T/R	Adam Tech	3	1.5	Micro USB
LDO	NCP1117ST33T3G	On Semiconductor	10	5.14	LDO
Button	COM-00097	Sparkfun	4	1.4	Button
Diode	1N5819HW-7-F	Diodes Incorporated	5	2.15	Diode
Blue LED	SMLE12BC7TT86	Rohm Semiconductor	2	1	Blue LED
PCB (2 days)	N/A	PCBWay	5	39	N/A
Terminal Block	1725656	Phoenix Contact	6	11.94	Terminal Block
Resistor and Capacitor	N/A	N/A	N/A	21.9	N/A

***Table 3**: List of Parts*

Note: Please see Appendix A for information of resistors and capacitors.

Total price of parts:217.71 + tax.

4.1.3 Total Cost

Total Cost = Labor + Parts = 6400 + 217.71 + tax = 20217.71 + tax.

4.2 Schedule

Week	Daniel	Rachel	Anusha

3/1	Setup web server api	Design circuit schematic	
3/8	Create database schema for data	Design PCB	Research on ideal sensor values
3/15	Setup database	Modify PCB	Setup motor control
3/22	Setup functions to make changes to database	Test Motor control	Setup sensor control
3/29	Setup endpoints database functions as api for application and device	Install and adjust PCB and wiring design	Finalize Arduino code
4/5	Create mobile app front end	Testing and troubleshooting	Setup server communication
4/12	Connect mobile app to send data between server and device	Debugging with mobile app	Debugging with mobile app
4/19	Prepare for demonstration		
4/26	Prepare for presentation and start final paper		
5/3	Finish final paper		

***Table 4**: Schedule*

5. Conclusion

5.1 Accomplishment

Overall, our project had a working prototype that could be used in any household. We had a functioning PCB and a well designed application. We also accomplished all the functionalities we desired. The device was able to be controlled by all three control modes: device control, user control and alarm control. Although there are a few possible improvements, we would consider our project to be a success.

5.2 Goals Not Met

During the development of the system's network, the original intent was to integrate a web socket with the server. By using a websocket, the device would be able to make a direct TCP connection with the server, allowing bidirectional communication between the two. This would allow the device to receive seemingly instant updates whenever a change is made on the server database. In addition, by integrating the serverless architecture, we would only be billed on the actual api calls of when a change happened. However, due to the unforeseen difficulty of integrating serverless web socket functionality, that idea was pivoted in the interest of clearing production blockers due to the dependence on the network module. Instead, we decided to use a more traditional webless server interaction of REST apis at an endpoint. In this implementation, vectors must poll the server every interval to check for an update, instead of the server being able to broadcast an update to all devices. In this result, the overhead cost of api calls would increase due to the waste polling calls when no update is actually made yet.

Another goal that we did not meet was using 2 motors to control the curtains instead of only one. While we had the motor parallel board which technically would allow the connections between both motors and the PCB, we did not talk to the ECE shop to figure out possible setups. We were concerned that if we were to use two belts, they would be in each other's ways and if we were to use only one belt, we could not set up the idlers on the wall to make it work.

Also, we did not have time to design a cover for the PCB. Since the PCB needs to be placed by the window, it is possible to contact water from rain or condensation. Thus, to protect the device and for safety concerns, it is ideal to have a cover for the PCB. We were thinking of designing a 3D printed cover, but we could not achieve this goal as we ran out of time.

5.3 Future Prospects

Aside from the goals not met, some other features we would hope to accomplish in the future is adding interchangeable curtains and adding an thermo anemometer. Currently, our device is only

set up to operate on one type of curtain. However, in the future, we would like the device to choose from a variety of curtains for different situations. For instance, detecting low temperatures, the device would be able to switch out light, daytime curtains for heavier thermal curtains for the night.

The addition of an anemometer would allow the device to start detecting the presence of a draft through the window. By doing this, we add another vector for our device to make measurements on. Detecting draft will allow the device to make assumptions of heat loss out of the house, or even heat loss into the house.

5.4 Ethics

One of the main concerns that we needed to consider is that of security. Since this device interacts with the physical barrier that maintains privacy in a home's windows. Although heating and lighting are the forethoughts of this project, we must keep in mind to not infringe upon a home by reducing the control of the user to use curtains as a means to not allow outsiders a view on their homes. In order to maintain this, the user must always feel in control of when the curtains are open or closed, and allow the user to disable any behavior that automatically opens or closes the blinds. This is provided through the user override mode in our design.

In this way, we uphold the IEEE Code of Ethics, #1: "to hold paramount the safety, health, and welfare of the public…[10]."In order to expand on this ethical consideration, we should also add extra security measures such as a password to protect other people from changing the user preferences.

Another consideration that we focused on was one of sustainability. According to the IEEE Code of Ethics, we should "strive to comply with ethical design and sustainable development practices[10]." We believe we upheld this ethical consideration since our project will help individual households attain a more sustainable level of energy consumption by decreasing heating and cooling costs.

6. References

[1] NRDC. 2020. ‘How to Keep Warm and Save on Your Energy Bills This Winter’. [Online] Available at: https://www.nrdc.org/stories/how-keep-warm-and-save-your-energy-bills-winter [Accessed 05 May 2021].

[2] VeryWellHealth. 2020. ‘Get Morning Sunlight and You'll Sleep Better’. [Online] Available at: https://www.verywellhealth.com/morning-sunlight-exposure-3973908 [Accessed 05 May 2021].

[3] Aqara. 2020. ‘Curtain Controller’. [Online] Available at: https://www.aqara.com/en/smart_curtain_motor.html [Accessed 05 May 2021].

[4] Curtains For Barns. n.d. ‘Fully Automated Curtain Systems’[Online] Available at: https://curtainsforbarns.com/complete-systems/ag-curtain-automation/ [Accessed 05 May 2021].

[5] DiyIOt. 2020. ‘Arduino vs ESP12E vs ESP12E32 Microcontroller Comparison’. [Online] Available at: https://diyi0t.com/technical-datasheet-microcontroller-comparison/ [Accessed 05 May 2021].

[6] RepRap. n.d. ‘NEMA stepper motor with L293 drive - would it work?’ [Online] Available at: https://reprap.org/forum/read.php?1%2C259540 [Accessed 05 May 2021].

[7] Pololu. n.d. ‘Pololu - a4988 stepper motor Driver Carrier’. [Online] Available at: https://www.pololu.com/product/1182 [Accessed 05 May 2021].

[8] Sparkfun. n.d. ‘EasyDriver - stepper motor driver’. [Online] Available at: https://www.sparkfun.com/products/12779 [Accessed 05 May 2021].

[9] Engineering ToolBo. 2004. Illuminance - Recommended Light Level. [online] Available at: https://www.engineeringtoolbox.com/light-level-rooms-d_708.html [Accessed 05 May 2021].

[10] Ieee.org. 2021. ‘IEEE Code of Ethics’. [Online] Available at: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 05 May 2021]

Appendix A - List of Resistors and Capacitors

Name of Part	Product Number	Manufacturer	Quantity	Total Cost ($)	Link
Res 12kΩ	RMCF0805FT12K0	Stackpole Electronics Inc	25	0.68	Res 12kΩ
Res 470Ω	RMCF0805FT470R	Stackpole Electronics Inc	15	0.41	Res 470Ω
Res 220kΩ	RMCF0805JT220K	Stackpole Electronics Inc	10	0.2	Res 220kΩ
Res 100kΩ	RMCF0805FG100K	Stackpole Electronics Inc	10	0.27	Res 100kΩ
Res 4.7kΩ	RMCF0805FT4K70	Stackpole Electronics Inc	10	0.27	Res 4.7kΩ
Cap 100uF 2.5V	PMK212BBJ107MG-T	Taiyo Yuden	5	8.2	Cap 100uF 2.5V
Cap 0.1uF 16V	C0805Y104K4RAC AUTO	KEMET	15	5.46	Cap 0.1uF 16V
Cap Tant 100uF 6.3V	TAJB107M006RNJ	AVX Corporation	3	1.5	Cap Tant
Cap 10uF 10V	CGA4J3X7S1A106 M125AB	TDK Corporation	10	3.16	Cap 10uF 10V
Cap 10uF 25V	C2012X5R1E106K1 25AB	TDK Corporation	5	1.75	Cap 10uF 25V

***Table 1**: List of Resistors and Capacitors*

Appendix B - Requirements and Verification Tables

Requirement	Verification
1. AC/DC Adaptor: The Adapter is able to convert the incoming power into 12V DC, at a rate of 12V +- 5% at a rate of 2A. 2. Step Down Module: The step down module is able to convert 12V DC to 5V DC with output current being 2A. 3. LDO: The LDO converts 5V DC from the step down module or the USB connection to 3.3V, to power the ESP12E chip.	1. Using a multimeter, we can measure the output voltage to be 12V when the adaptor is plugged into the wall. 2. Using a multimeter, we can measure the output voltage to be 5V when the input voltage is 12V. 3. Using a multimeter, we can measure the output voltage to be 3.3V when the input voltage is 5V.

***Table 1**: Requirements and Verification of Power Module*

Requirement	Verification
1. ESP12E: ESP12E must be able to run provided instructions at any time given the provided power is supplied 2. Motor Control: Motor control can interpret the state given by the ESP12E chip and start sending power, continue sending power, or stop sending power to the stepper motor.	1. Arbitrary code, such as changing the states of the onboard LEDs on the chip, can run upon power being supplied to the ESP12E chip. 2. The ESP12E chip will send three different states to the motor control: begin, continue, and end. The motor can be observed to see if it follows these three states.

***Table 2**: Requirements and Verification of Control Module*

Requirement	Verification
1. Photo Sensor:	1. Check if the signal given by the

Photosensor is able to detect lux value within an accuracy of 20%, correctly identifying the light state of the environment 2. Thermal Sensor: Thermal sensor is able to detect the temperature of a surface with accuracy of 15%	ESP12E chip determines what the lighting of the room is correct in three different scenarios: pitch black, dimly lit, and fully lit. 2. Using two object with controlled temperature surfaces (ie. ice cube, hair dryer, etc), check the temperature read by the sensor and compare it with temperature read with a thermometer.

***Table3**: Requirements and Verification of Sensor Module*

Requirement	Verification
1. Stepper Motor: Able to supply 0.2N-m torque given 12V power supply	1. We can attach a scale and have the motor spin, pulling on the scale to measure the provided force to ensure it meets the given requirements.

***Table4**: Requirements and Verification of Output Module*

Requirement	Verification
1. Server Module Have reliable uptime and allow for successful connection at anytime to store, process, and send data accurately 2. Wifi Module: Is able to successfully contact the server to push and pull data consistently, reliably and return the data to device with no packet error or loss	1. Have a constant API request to the server every 3 hours (running from the device or a personal workstation). If the api returns an error instead of an expected information payload, the server is then known to be down. If such an error does occur, viewing the AWS Cloudwatch logs can allow us to debug the problem. 2. Upon device startup, the device will send an API request to the "helloworld" endpoint of the api. The server will send a data packet that the device can then use to compare and check for errors. If there is error, then

	the network is deemed unreliable and another connection is reinitiated, and the same test is run.

***Table 5**: Requirements and Verification of Server Module*

Requirement	Verification
1. User Control: Using the app, users can successfully send commands and preferences to the device to which the device executes the specified commands. 2. Data Viewer With the device the user is able to successfully view real time data changes to from the device and sensors	1. By manually using the app, we can change settings and physically observe if the change sent to the device coincides with the user choice, checking if the device rESP12Eonds to the correct curtain state, as well as sensor preferences 2. Using the device, we can physically alter the device environment to check if the data is correctly displaying the change in environment (eg. cover the light sensor or hold the temperature sensors to warm it)

***Table 6**: Requirements and Verification of User Interface Module*

Appendix C - Circuit Schematics, PCB Layout and PCB Board

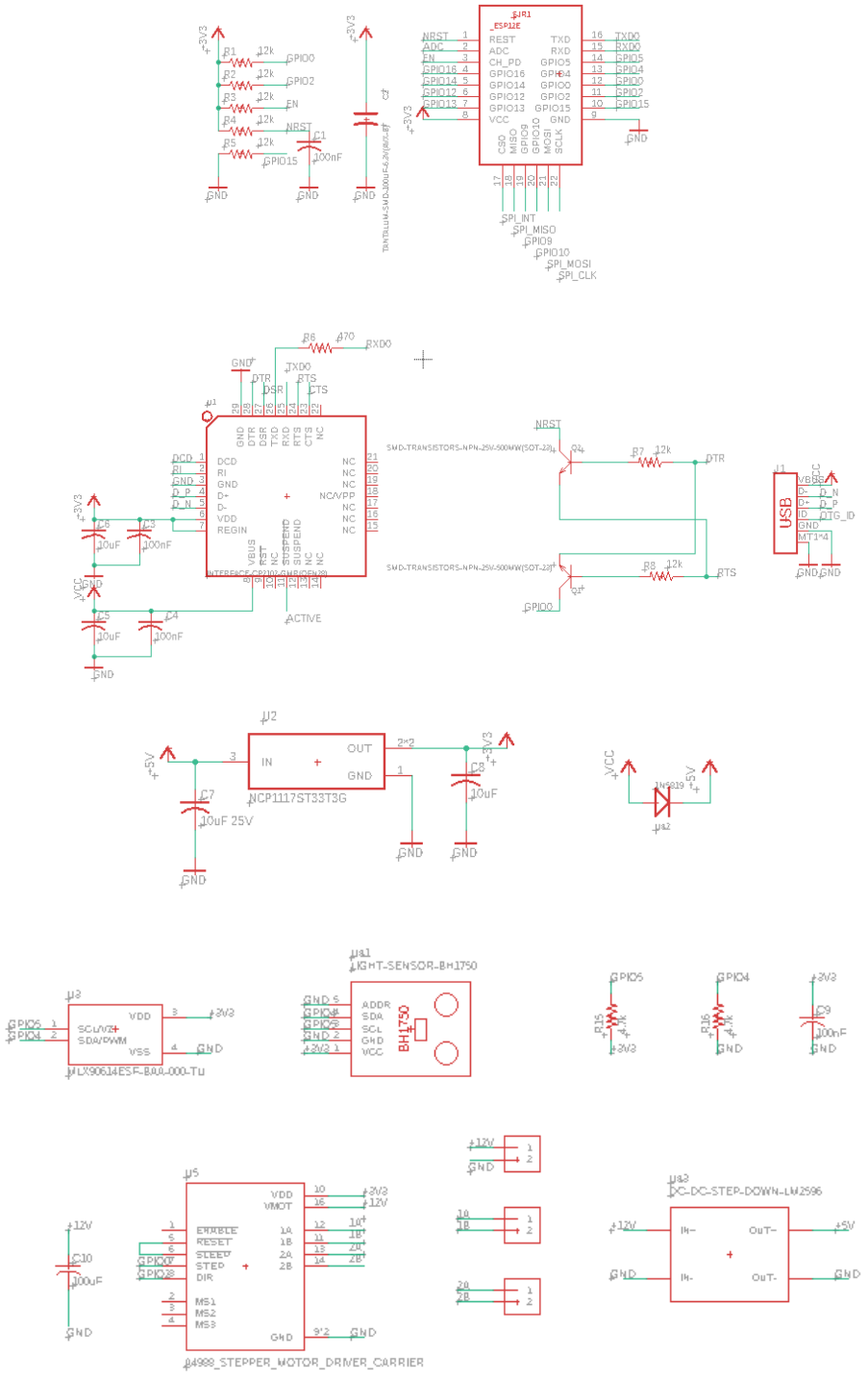

***Figure 1**: Circuit Schematics*

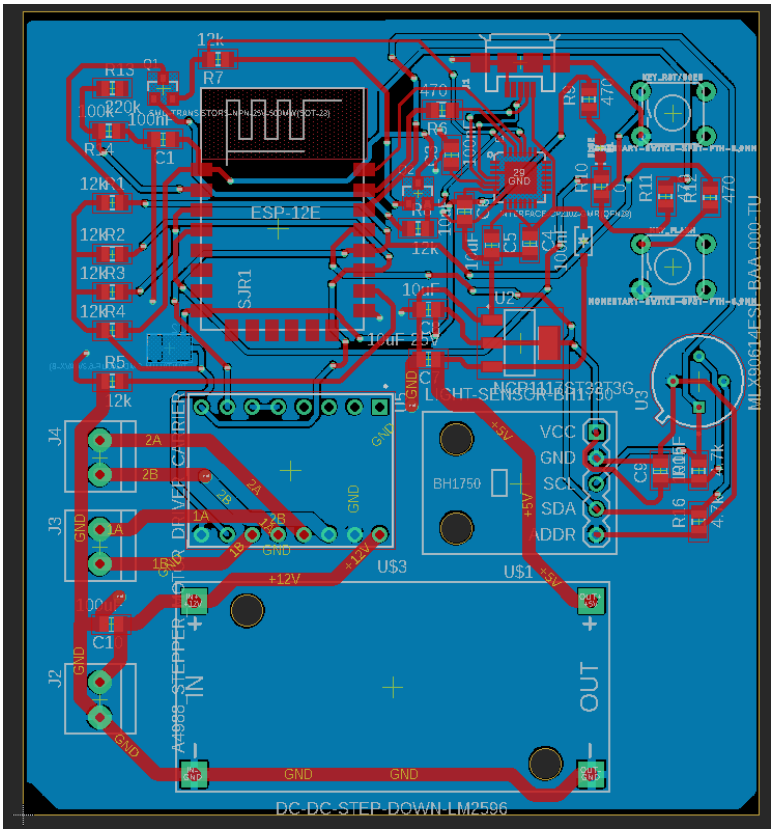

***Figure 2**: PCB Layout*

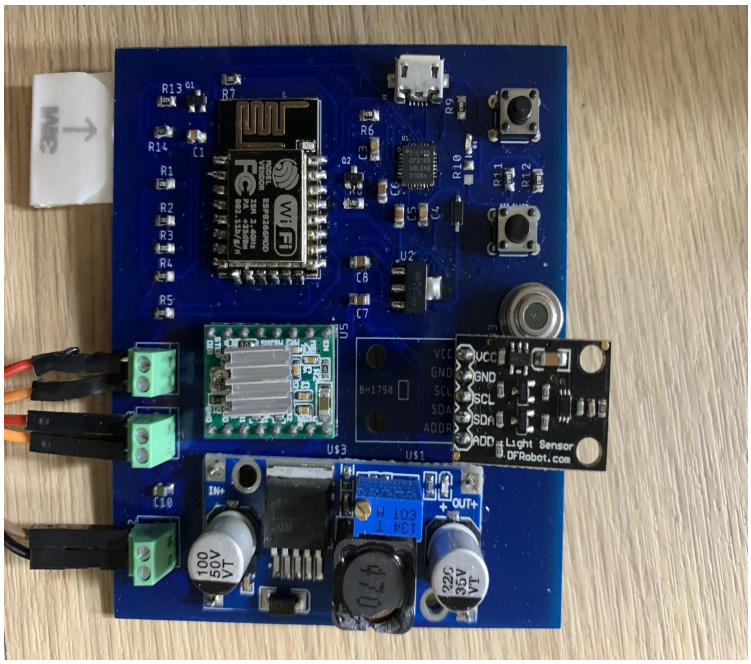

***Figure 3**: PCB Board*