Efficient Light Control System for Plant Growth

By

Christelle Seri (seri2),

Heonjang Lee (hl8),

Sungjoo Chung (sungjoo2)

Final Report for ECE 445 Senior Design, Fall 2022

TA: Zhicong Fan

12/07/2022

Project 5

Abstract

The current market for grow lights for plants mostly only features products that control the light modules by the user. This is inefficient in the sense that during daytime, there is plenty of light coming from the sun that we can utilize and inconvenient in the sense that the user has to manually control the grow lights. Our project aims to create a product that automatically controls a motorized blinds and a grow light module to maximize the electricity efficiency of the product, using light from the sun when it is available and sufficient, and otherwise supplementing it with the LED modules.

High level requirement of our product would provide the plants with lights of wavelength that are proven to be most beneficial for plant growth for an extended period of time. Moreover, the photosensor should accurately measure the light intensity, and based on this data along with the target luminosity, the software should accurately instruct our system to reach the target luminosity through instructing the motorized blinds and the LED modules appropriately. Lastly, our application should cover enough modes of plants to be user friendly.

Contents

1. Introduction

1.1. Problem

Greenhouses in the industry are essential in the agricultural fields, but also costly. Greenhouses are proven to be an effective solution to growing plants, but over time, the electricity costs will begin to add up. According to research, a 200ft * 100 ft greenhouse costs about $6000 per month [1].

1.2. Solution

This project proposes an energy efficient blinds system with LED lights as a solution to simulate a cost effective greenhouse system. A sensor can be placed on the plant vase to measure the amount of light received. The blinds aim to adjust via an attached motor so as to optimize the amount of light to the plant. The LED lights will turn on when the maximum sunlight from the blinds is insufficient.

Thus the LED lights would only be used when strictly necessary, cutting down on electricity costs as a result. Additionally, the blinds system could be scheduled and adjusted to user needs as well.

This system will be easily controlled by a user using an application, and also statistics will be provided on the application.

1.3. Visual Aid

Figure 1: Model of the system

1.4. High Level Requirements

- The artificial light-source, combined with available natural light, should provide light of wavelength 400-700 nm, which has been proven to be optimal for plant growth [2], and a maximum of 3,500 lux over a 12 hour period [3], to provide sufficient light for high-light plants, when needed.
- The photosensors on the vase should correctly calculate the illumination on the plant to minimize the discrepancy between the actual illumination on the plant and the expected illumination within ±5%.
- The application should have enough modes to cover various types of plants including cactus, tropical plants, conifers, etc

2. Design

2.1. Block Diagram

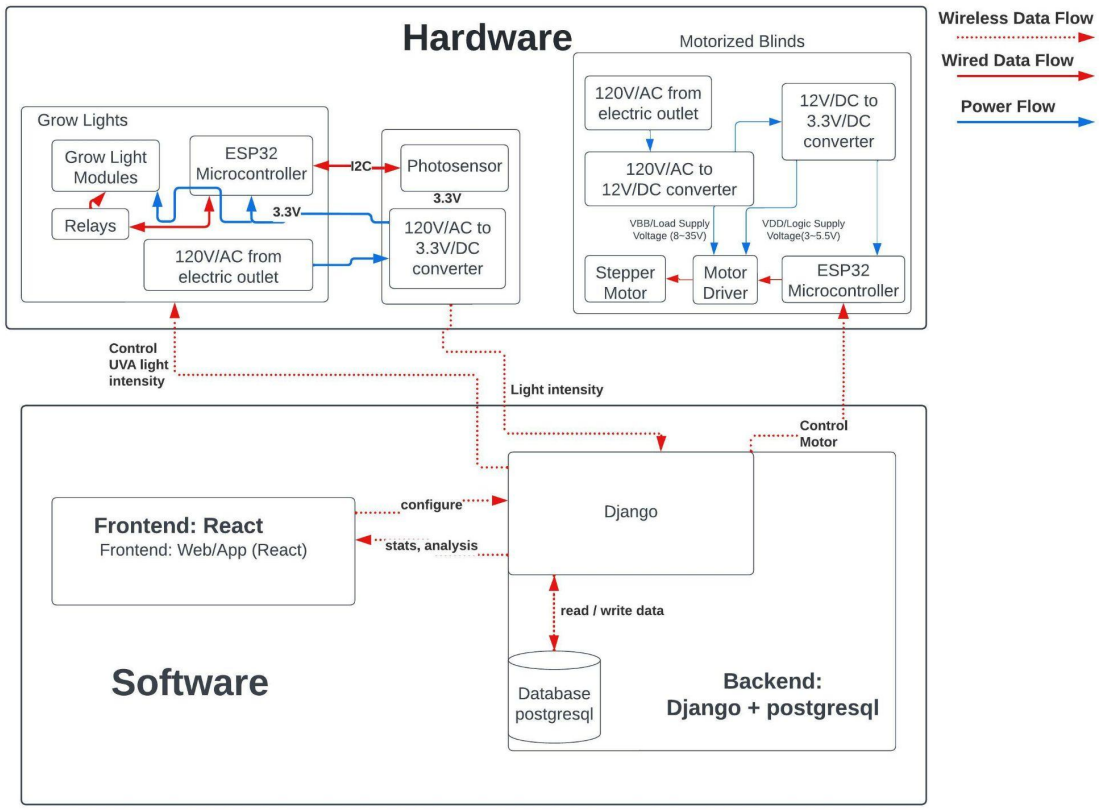


Figure 2: Block Diagram of the Project

2.2. Subsystem Overview

Photosensor

The photosensor used is a BH1750FVI Luminosity sensor. The advantage of this sensor is that it is a precise digital device, and compact in size as well. The ESP32-WROOM-32E communicates directly with the BH1750FVI via I2C. Luminosity data is transmitted to the app through wifi. The ESP32 was chosen as it is a well documented microcontroller with both bluetooth and wifi capabilities. This subsystem will share a microcontroller with the grow light subsystem. The microcontroller acts as a wireless access point, from which photosensor values can be read via GET requests.

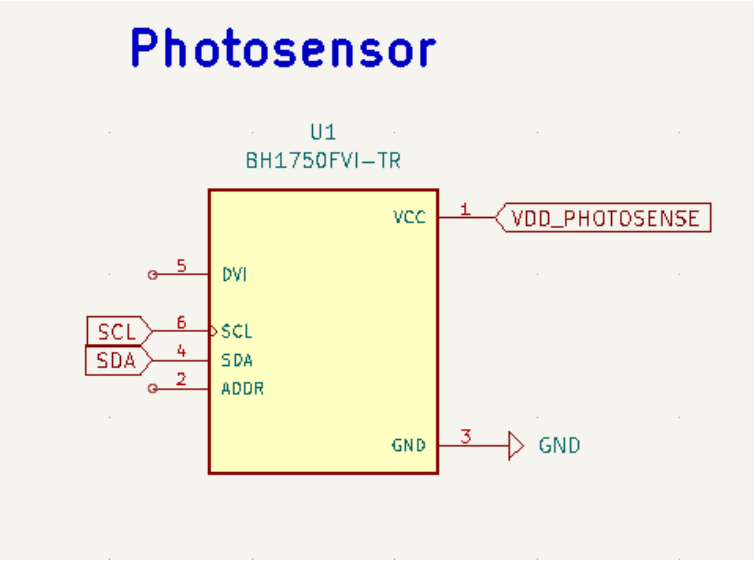


Figure 3: Photosensor Schematic

Grow Lights

To maximize the energy efficiency of the system, an adjustable LED light circuit will be implemented. LEDs were chosen for their energy efficiency, low cost and low heat emission [4]. Implementing an LED grow light circuit allows for optimization of the light provided to the plant. Below is a table showing the benefits of different types of LEDs [2].

Table 1: Improvement on Plant Growth from Different LEDs

Blue Light: increases chlorophyll production	**Yellow-Red Light:** improves chlorophyll absorption, germination and bud development
Green Light: helps with photosynthesis and can improve plant size	**UVA Light:** can enhance plant pigmentation, and thicken leaves

Due to cost limitations, the final design does not include a UVA light. To provide a combination of Blue, Green and Red light,GW P9LR35 mid power white LEDs were selected. The primary benefit being the high luminosity output as well as the resulting design simplification of having only one LED. For one grow light module, there will be three white LEDs. To adjust light to the plant these grow light modules will be switched

on and off using transistors. Primarily, the output of the microcontroller will switch the transistors from cutoff (OFF) to saturation(ON).

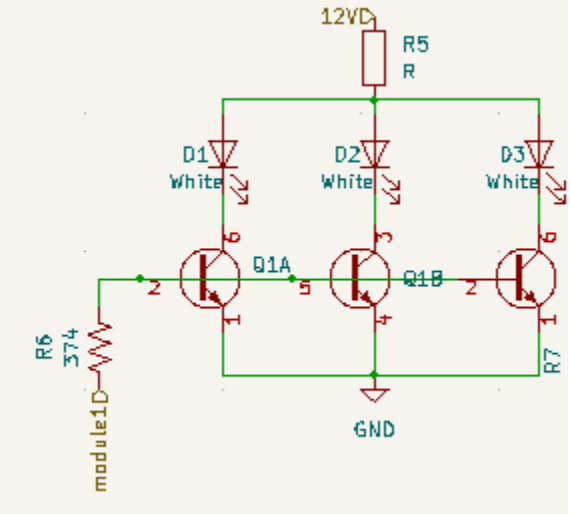


Figure 4: Single Grow Light Module

The microcontroller will switch modules on and off accordingly to adjust the light to the plant. To provide sufficient light to different types of plants the required modules are as follows,

Table 2: Modules Per Plant Type

Objective	Lumens Required	Required Number of Modules
Low-Light	538-2690	4
Mid-Light	8,070	14
High-Light	10,760	18

Due to time limitations, the prototype included only 7 modules, but more could be added as necessary

To power the lights, 120V AC is drawn from the wall and converted to 12V DC through an AC/DC converter. If there is ever a need to power off the prototype in case of an emergency, pressing the reset button of the microcontroller will switch all the LEDs off. Additionally, the website hosted by the ESP32 access point has an emergency shutoff implemented by following the link: 192.168.4.1./All_OFF.

In order to be able to program the microcontroller properly, a USB peripheral as well as Enable and Boot circuits are implemented. Referencing the design of Team 47 from Spring 2022, the boot and enable circuit were implemented and were a success [5]. To flash the code, both the Enable and Boot buttons are pushed at the same time.

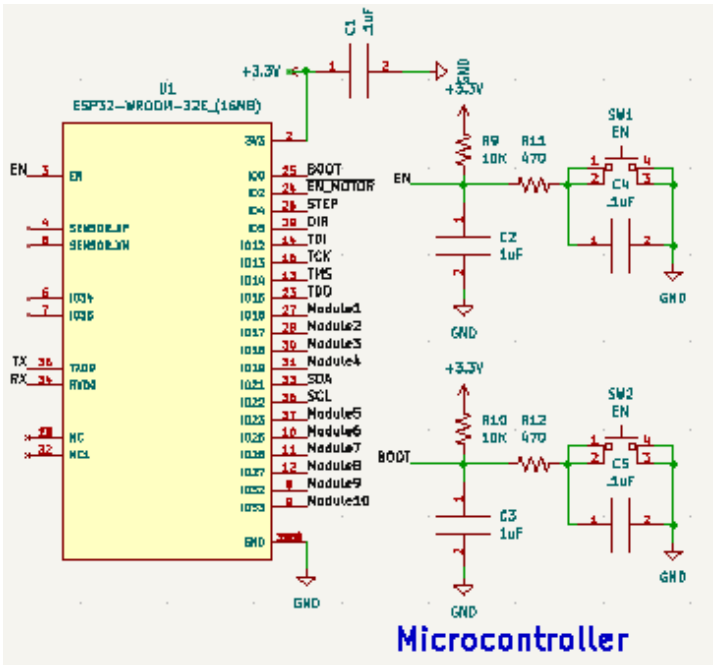


Figure 5: Microcontroller Connections

Motorized Blinds

The motorized blinds subsystem will be in charge of controlling the tilt angle of the blinds for the plants to receive the desired amount of light. The system includes a ESP32 microcontroller, stepper motor, A4988 motor driver, LM2596 buck converter and a 12V DC power supply.

The decision was made to use a stepper motor, more specifically the 28BYJ-48 stepper motor, rather than a servo motor, due to the requirements of this subsystem. In our system, the blinds will be adjusted mostly at low speed, which the stepper motor excels at as it provides high torque, reliability and precision, at a much affordable price than the servo motor [6]. This motor will be used to control the tilt of the blinds.

To stay consistent with the choice of microcontroller, this subsystem will also be operating through the ESP32 microcontroller due to the reasons discussed in the above section. The microcontroller will be in charge of receiving instructions from the application, and controlling the tilt of the blinds via the motor depending on the current state of the system.

A motor driver is implemented between the microcontroller and the stepper motor because the microcontroller operates in low current whereas the motor operates in high current. The A4988 Stepper Motor Driver was chosen as it is compatible with our stepper motor, it allows the control of maximum current output which translates to maximum voltage for the motor and that it has an over-temperature thermal shutdown system for safety measures [7].

The LM2596 buck converter and 12V DC power supply are needed in order to supply appropriate voltages to the components above.

2.3. Software Design

2.3.1. Procedure

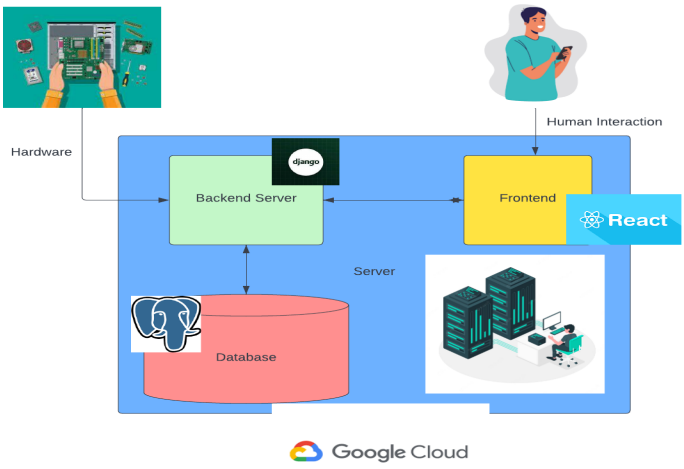


Figure 6: Overall Software Structure

The role of the software part in the greenhouse system is to accept the data, analyzed the passed data, and then control the hardware parts based on the analysis. The design of The strength of such a design is the hardware components, which are difficult

to simulate and debug, do not need to be bothered with all complicated business logic. By using the software system as a black box, the hardware parts can only focus on ensuring its fundamental functionalities.

In order to do so, the server should have a backend server, a frontend server, and a database server so that it can accept data from the hardware, store those data points, and then show the status and statics to the system users. This whole system is containerized using Docker and their images are uploaded to the Google Cloud Platform. There, the whole system runs in a cloud so that it is connected to the internet.

2.3.1.1. Backend server

The backend server is responsible for accepting data points, calculating and analyzing passed data points, making decisions, and sending appropriate commands to the hardware components, and lastly, reporting the history and the current statics. For this goal, Django is used. Django is a python backend library that is one of the most popular backend libraries. There are many alternative popular backend frameworks such as Spring. However, Django was chosen for many reasons. First, it is a python library. Python is known as the most productive programming language due to its speedy development. Unlike other languages like Java, it does not require complex programming syntaxes. Also, it is an interpreted language in that it does not have to compile the entire program before executing it. Therefore, it is handier to write test cases. Second, Django has a rich community of developers due to its popularity. Therefore, it is possible to write a more sophisticated product with a help of proven libraries.

2.3.1.2. Database

The database server is responsible for storing the data securely and offering the stored data when requested with performance. While many types of databases are possible such as relational databases, NoSQL databases, or graph databases, a relational database is chosen in order to support the dependencies between data as well as aggregation of them. For example, data points have a relationship with a photosensor. In order to figure out aggregate the data points by a sensor, it is necessary to use a relational database. Also, due to the high frequency of data points, graph databases are not suitable due to their poor performance in transactions.

Out of many other relational databases, such as MySQL or SQLite, Postgresql is chosen because of its high compatibility with Django. Also, it has a rich development community as well due to its popularity.

2.3.1.3. Frontend

The frontend server is responsible for the interaction between human users and the greenhouse system. There are a number of popular frontend frameworks such as Vue.js, Angular, and React. Out of these frameworks, this project was implemented with React because it is the most mobile-friendly framework. It is compatible with ReactNative which is a mobile application framework. Because the mobile application will be crucial for better accessibility from the users' point of view, this feature was important in this project. Also, because it is the most popular frontend framework as of now, it has the richest development community.

2.3.2. Design details

2.3.2.1. Backend

The backend consists of three big subsystems: data acquisition, analyzer, and decision maker. These three subsystems run independently(as a unique thread) of each other with their own responsibility. Therefore, the functionality of each subsystem is well-modularized so that each of them has a concise purpose. This lowers the development difficulty and strengthens the stability of each subsystem. Data acquisition - collect data and save it into the database

1. **Data acquisition**

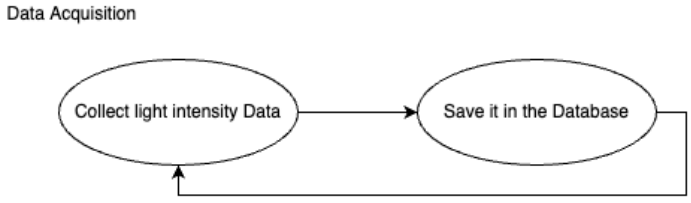


Figure 7: Logical Flowchart of Data Acquisition

The system's all business logic is fundamentally dependent on the light-intensity data collected from the photosensor in the hardware system. Without that information, it is not possible to analyze the current status or offer any useful information to users. Likewise, this process is the most crucial part of the greenhouse system. In order to

promise stability, this process is separated into one individual thread so that it can run without external interference.

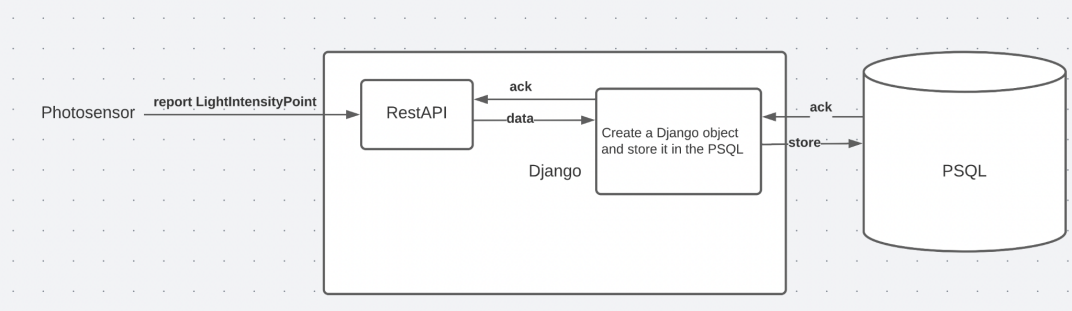


Figure 8: Structural Diagram of Data Acquisition

In this thread, the backend server sends a request to the hardware system to respond with a photosensor value. When the thread receives the value, it saves in the database. This process runs in an infinite loop with a 100ms delay after each execution.

2. Analyzer

The Analyzer's goal is to calculate how much light intensity the system is lacking or overloaded. The calculated number will be saved into the database so that the other thread(Adjuster) can take an action accordingly to achieve the system's goal. In order to synchronize with the Adjuster which runs every 10 minutes, the system will also wait 10 minutes after one cycle.

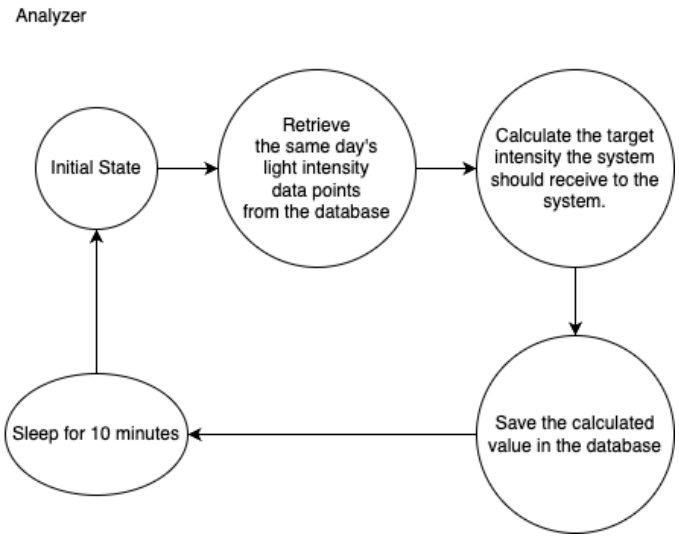


Figure: 9 Logic Flowchart for Analyzer

Because the project goal is to maximize the electricity consumption efficiency, the calculation logic focuses on maximizing the natural light source usage. To do so, the system computes the optimal insolation graph from sunrise to sunset and tries to adjust the system based on that.

$$F = F_0 \times cos\theta_0 \tag{2.1}$$

[9] Sun's Flux Density Formula

In order to calculate the insolation value, the system uses the flux density formula (2.1). In the formula F_0 is the strongest insolation of one day and the θ represents the angle between the system's zenith and the center of the sun. At sunrise, it is -180 degrees and at sunset, it is 180 degrees. Using this formula, the analyzer can compute the target illumination bell curve as well as the difference between the current light intensity value and the target intensity value. Refer to Appendix A for a visualization of the calculation result;

2.3.3. Adjuster

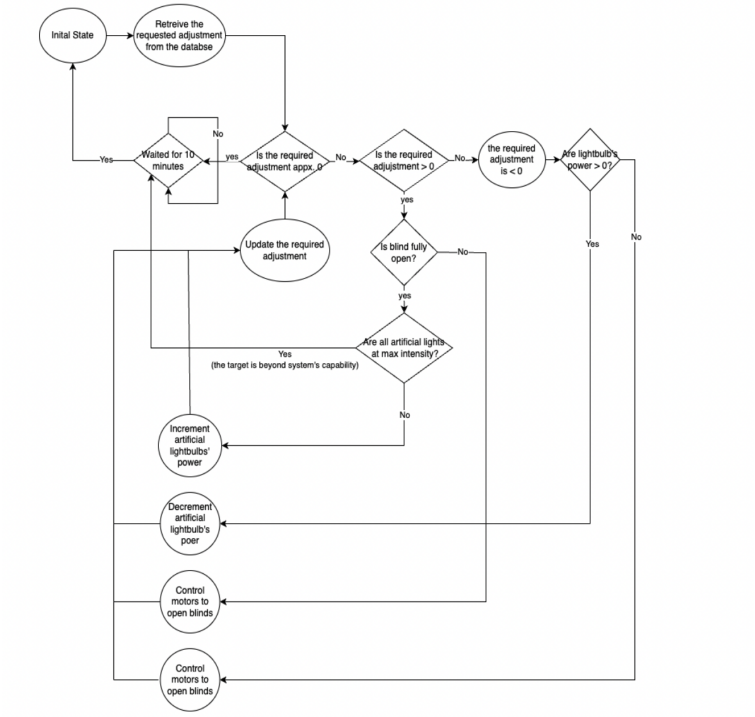


Figure 10: Logic Flowchart for Adjuster

2.3.3.1. Frontend

The frontend's role is to deliver data generated from the backend server to a user in a readable format and also to deliver commands from the user to the backend server. Therefore, the frontend consists of 4 tabs: the home page, system configuration page, report controller page, and history page. For screenshots of each page, refer to Appendix A.

1. **Home**

   On this page, the server shows the current configuration of the system to the user. Therefore, the user does not need to actually see the greenhouse system in person to check its current status.

2. **System Configuration**

   On this page, the user can update the current status of the system, and the update is immediately applied so that the target illumination changes correspondingly.

3. **Remote Controller**

   On this page, the user can directly control the system remotely.

4. **History**

   On this page, the user can see the history of the light intensity stored in the system, Through this, the user can know whether the system is behaving as intended or not.

3. Design Verification

3.1. Photosensor

The photosensor was tested by adjusting the light to the photosensor and verifying the changes through the website. Mainly, a flashlight was shined onto the photosensor, then turned off, and later the photosensor itself was covered by a hand. The measurements were transmitted to the website accurately and were as expected: displaying a peak, a fall and then an even sharper fall.

3.2. Grow Lights

The grow lights were tested both manually and through the software's logic. Mainly, modules could be switched on or off by redirecting to the 192.168.4.1/Module{#}_{ON or OFF}. Through these manual tests, the light to plant visibly changed as well as was reflected in the photosensor measurements. They also reacted as expected to directions from the software turning on and off to supplement the light to the plant as necessary.

3.3. Motorized Blinds

The motorized blinds subsystem was tested by sending appropriate signals to turn the motors both ways from the ESP32 and verifying that. After this was functionality was verified, the capability of ESP32 to control the motor to angle the blinds at a desired angle was tested by letting the ESP32 to instruct the blinds to tilt at an arbitrary angle and confirming that the blinds is angled in the desired angle within a $2.5°$ error. This was also verified. Lastly, for the responsiveness of the subsystem, the time it takes for the motors to start moving after the ESP32 has passed on its instruction was measured and verified to be less than 5 seconds.

3.4. Software

On the software side, all tests are done as unit tests so that they can be tested without hardware dependency. Using a python program, it prepared a set of data or instructions to simulate scenarios and checked whether the software system behaves as expected. For more detailed information about the verification, refer to Appendix C.

3.4.1. Data Acquisition

While its goal and logic are simple, its tolerance and stability were the most crucial. Therefore, its verification focuses on guaranteeing those features. For each requirement, simulators are created so that this functionality was tested every time before the application's deployment.

3.4.2. Analyzer

The key requirement of this subsystem is that the calculation must be accurate. Therefore, scenarios with different conditions are manually calculated, and then the correct results are compared with the result from Analyzer

3.4.3. Adjuster

The key to this subsystem is that it should make a correct decision in various circumstances and that decision should be executable for the hardware. (i.e., the light power should not exceed the hardware's hard limit)

3.4.4. Frontend

The frontend's role is not complex. Because it only needs to deliver commands from a user and show the data to the user in a human-readable way, it mainly needs to be verified whether data are corrected and delivered from the server to the user and vice versa.

3.5. Integration

As a whole, software, and hardware, the greenhouse system should either open the blind or turn on LEDs whenever the target illumination is greater than the current illumination on the photosensor and close or turn off LEDs whenever there is more illumination than the expectation.

4. Cost and Schedule

UIUC's ECE AY20-21 grad students have an average starting salary of $92824. Our team consists of 3 members and each of us worked 10 hrs/per week (Appendix D). A full-time employee works 40 hrs/week and this was a 10-week project so the total human resource cost was $92824 per year / 52 weeks * 10 weeks * (10 hours / 40 hours) * 3 = $13388

The cost of all the components combined is $102. (Appendix D)

This brings the total cost of our project to $13490.

5. Conclusion

5.1. Successes and Failures

In conclusion, most of our project worked as it was intended. It accurately senses the luminosity value from the photosensor, calculates the difference between the target luminosity and actual luminosity and controls the LED modules to achieve the target luminosity in the end. Furthermore, manual controls from the application such as turning the blinds and controlling the LED modules also worked as intended. However, the ESP32 for this subsystem was unsuccessful due to a lack of connection. Thus, separate verification tests were run with the subsystem being instructed by an Arduino board rather than an ESP32 to confirm the functionality of it.

5.2. Further Work

Going forward, one of the features that could be improved is a potentiometer in the grow light modules. Currently, the luminosity values coming from the LEDs differ by increments of the brightness of each of the LED modules, meaning that there is less precision in the control of luminosity. If a potentiometer is added to each grow light module, the system would be able to more precisely achieve the desired luminosity by adjusting the resistance to adjust the brightness of each LED module.

Furthermore, better quality connectors and better wire management will definitely improve the quality of the product. The connectors that were ordered were not very stable, resulting in bad connections. This was further worsened by the poor wire management the design faced, which made it even more difficult to improve on the connections of the circuit.

5.3. Ethics and Safety

This project is subject to the ACM Code of Ethics 1.3*Be honest and trustworthy* [8]. Because we are integrating all existing technologies into one system, we are destined to borrow ideas or approaches made by other people. Therefore, we should cite those ideas properly to recognize the original author. Also, privacy should be taken seriously which is related to the ACM Code of Ethics 1.6 [11]. Our system stores user's authentication and they use histories in our database. This should be most firmly protected to avoid any privacy leakage.

This project is also subject to potential fire accidents at an industry level. Both the lights, and motor are never overloaded to comply with the ACM Code of Ethics 1.2 *Avoid harm* [11]. In addition to that, the LEDs may get hot when they are turned on for a certain period of time. The users of this system should avoid touching the system directly so that they do not get burned on their hands. The housing frame of the system encapsulates the lights.

Lastly, the light bulbs cluster should be located far enough away from the plants. Plants touching the bulbs will cause fire which will damage not only the plant, but also the system itself and potentially beyond. Because this is an automated system which has a minimum human interaction, it is possible that plant growth can occur without the user's expectation.

6. Appendix A: Software

Figure 11: Target Illumination Graph computed by the Analyzer

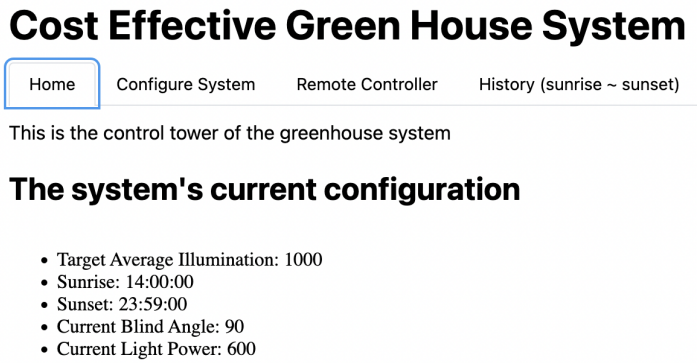


Figure 12: Home page

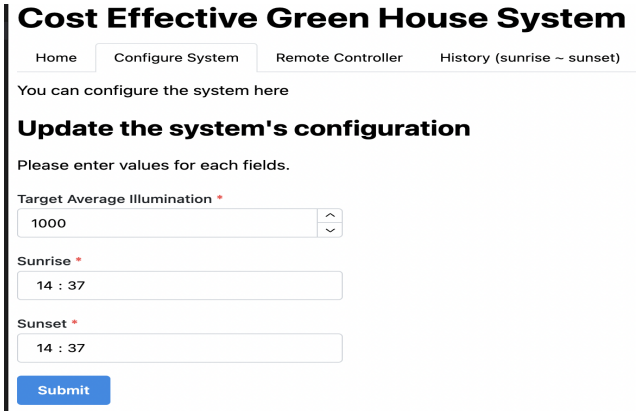


Figure 13: System Configuration

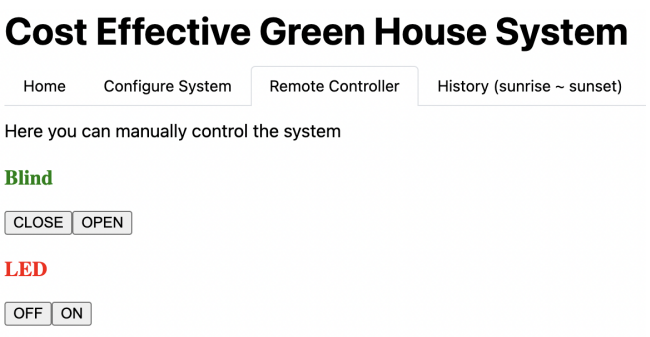


Figure 14: Report Controller

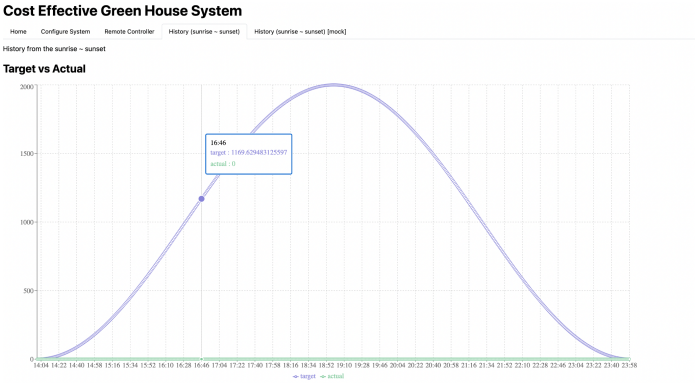


Figure 15: History Page

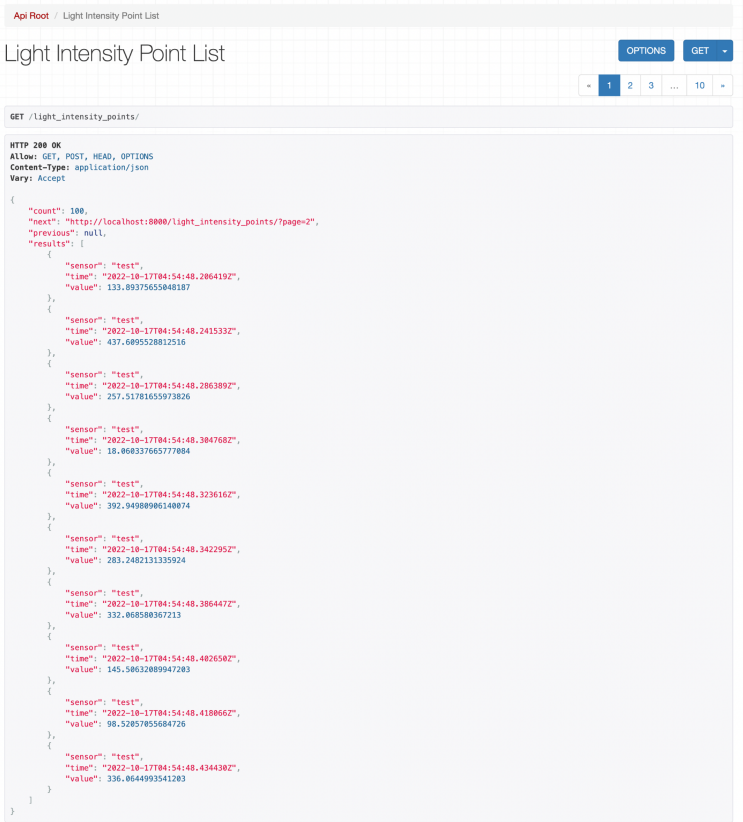


Figure 16: Data Acquisition Test result 1

```
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 81
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 81
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 81
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 81
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 79
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 81
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 81
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 81
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 81
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:14] "POST /light_intensity_points/ HTTP/1.1" 201 80
lcs-server | [17/Oct/2022 04:52:15] "GET /light_intensity_points/test/points/ HTTP/1.1" 200 8115
lcs-server | [17/Oct/2022 04:52:15] "GET /light_intensity_points/test/delete_all/ HTTP/1.1" 200 3
```

```
> python light_intensity/creation_simulator.py
Delete all points previously stored with the same sensor
Data points simulated
The number of intensity points matches the expectation
All points matched the expectation

 ~/school/LightControlSystem/src/server/simulator  on impl
>
```

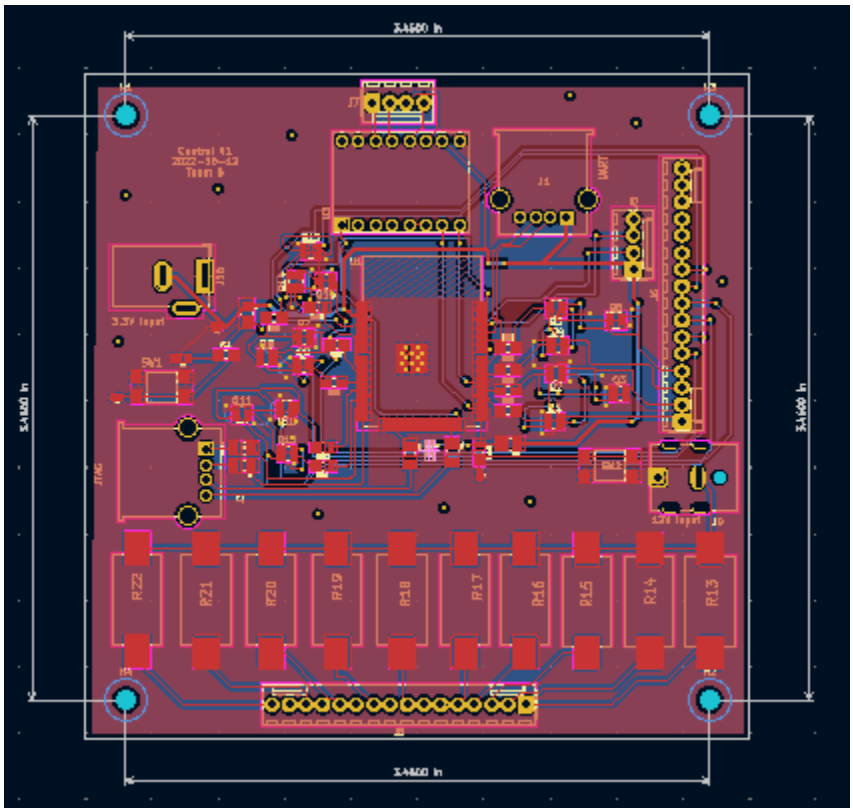
Figure 17: Data Acquisition Test result

```
target_average_illumination = 100
sunrise = 8 AM
sunset = 7 PM
period_in_seconds = 60

TIME     TARGET ILLUMINATION
08:00:00 0.0
08:01:00 0.004531464582713251
08:02:00 0.018125447647421297
08:03:00 0.040780717181076995
08:04:00 0.07249521995265072
08:05:00 0.11326608169920369
08:06:00 0.16308960738643513
08:07:00 0.22196128154349232
08:08:00 0.2898757686722541
08:09:00 0.3668269137308665
08:10:00 0.452807742691541
08:11:00 0.5478104631726709
08:12:00 0.6518264651449777
08:13:00 0.7648463217118981
08:14:00 0.8868597899638784
08:15:00 1.0178558119067314
08:16:00 1.1578225154637576
08:17:00 1.306747215551773
08:18:00 1.4646164152306902
08:19:00 1.6314158069267943
08:20:00 1.8071302737293289
08:21:00 1.9917438907606548
08:22:00 2.1852399266194307
08:23:00 2.3876008448969968
08:24:00 2.5988083057666524
08:25:00 2.8188431676458228
08:26:00 3.0476854889308727
08:27:00 3.285314529804283
08:28:00 3.5317087541144367
08:29:00 3.786845831327278
08:30:00 4.050702638550273
08:31:00 4.323255262627833
08:32:00 4.604479002308724
08:33:00 4.8943483704846455
08:34:00 5.192837096500045
08:35:00 5.499918128533153
08:36:00 5.815563636047526
08:37:00 6.1397450123144335
08:38:00 6.472432877005339
08:39:00 6.8135970788547695
08:40:00 7.163206698392742
08:41:00 7.521230050747083
08:42:00 7.887634688514986
08:43:00 8.262387404703661
08:44:00 8.645454235739923
```

Figure 18: Analyzer Test result

Appendix B: PCB Design



Appendix C: Requirement and Verification

C.1 Photosensor

Table 3 : Requirements and Verification for Photosensor Subsystem

Requirements	Verification	
1. The photosensor subsystem will transmit real time luminosity data to the rest of the system via bluetooth (every second) 2. The photosensor's measurements will reflect	1. The data from the photosensor will be polled for a period of 30 seconds at the end of which 30 accurate measurements should	1. Successful (Y)

changes to the light to the plant with a delay no greater than 2 seconds	be uploaded to the server 2. A phone's flashlight will be used to adjust light to the plant, in a range of none to the maximum possible light. The readings from the photosensor should reflect changes in the flashlight intensity.	2. Successful (Y)

C.2 Grow Lights

Table 4 : Requirements and Verification for Grow Lights

Requirements	Verification	Verification Status
1. The grow lights should be able to adjust to either increase or decrease the light to the plant 2. The grow light subsystem will transmit real time data on the power used (every second)	1. All of the grow light modules will be gradually switched on. The light to the plant should increase visibly. Additionally, the photosensor measurements should show an increase. 2. The data from the microcontroller will be polled for a period of 60 seconds, during which each of the grow lights will be switched on/off. At the end of this period, 60 measurements accurately reflecting the changes should be uploaded to the server.	1. Successful (Y) 2. Successful (Y)

C.3 Motorized Blinds Subsystem

Table 5: Requirements and Verification for Motorized Blinds Subsystem

Requirements	Verification	Verification Status
1. The microcontroller should instruct the motor to rotate in the desired direction to tilt the blinds 2. The microcontroller should be able to instruct the motors to angle the blinds at a desired angle within an error no bigger than $\pm$ 2.5°, which is half the minimum increment the blinds will be adjusted at 3. The motor should start rotating within 5 seconds, for both directions, of the UNO R3 receiving instruction from the application	1. Verification step for requirement 1 a. Have application to instruct UNO R3 to tilt the blinds in a specified orientation b. Confirm if the motor rotates in appropriate direction to perform its instruction c. Repeat steps a~b but with opposite orientation 2. Verification step for requirement 2 a. Have application to instruct UNO R3 to angle the blinds at an arbitrary angle b. Verify that the blinds are at an angle within the error margin using a protractor 3. Verification step for requirement 2 a. Have application to instruct UNO R3 to tilt the blinds in one orientation for an arbitrary amount b. Start the timer, preferably a stopwatch app or a digital timer c. Stop the timer when the motor	1. Y 2. Y 3. N

	starts rotating d. Repeat steps a~c but with opposite orientation	

C.4 Data Acquisition Verification

Table 6: Requirements and Verification for Data Acquisition

Requirements	Verification	Verification Status
1. Values should be correctly stored 2. All values should be safely stored while having high traffic	1. Tested with a simulator that sends requests with a random value and checks the database 2. Tested with a simulator that sends 100 requests in 5 seconds and checks the database whether all requests are successfully executed.	1. Y 2. Y

C.5 Analyzer Verification

Table 7: Requirements and Verification for Analyzer

Requirements	Verification	Verification Status
1. Computation should correctly calculate the target illumination 2. The Analyzer should correctly save the value in the database.	1. Tested with a precalculated scenario and compared the result with the correct values. 2. Tested with a simulator with a precalculated scenario and checked	1. Y 2. Y

	whether values are correctly saved in the database.	

C.6 Adjuster Verification

Table 8: Requirements and Verification for Adjuster

Requirements	Verification	Verification Status
1. The decision made should not exceed limits. I.E., light power cannot be negative and exceed its hard limit. 2. The decision made should be correct	1. Tested with a simulator that requires the system lower/higher illumination while it already hit its limitation and checked whether the limitation is strictly kept. 2. Tested with a simulator with pre-computed scenarios. Checked whether the created decisions matched the pre-computed results.	1. Y 2. Y

C.7 Frontend Verification

Table 9: Requirements and Verification for Frontend

Requirements	Verification	Verification Status

1. The value showing on the homepage should be correct with the actual configuration 2. A command should be correctly applied to system 3. The graph should show data correctly with an adequate scale.	1. Created a set of configurations and check whether values are correct for each update 2. Created a set of commands and check whether the system execute the command correctly. 3. Prepared a list of the data point set and checked whether the graph showed the correct value with an adequate scale	1. Y 2. Y 3. Y

Appendix D: Cost and Schedule

Table 10: Component Costs

Component	Quantity	Manufacturer	Cost/Quantity($)	Total Cost($)
BH1750 Ambient Light Sensor	1	Rohm	4.50	4.50
Generic 4 Pin Connector	2	Molex	0.59	1.18
White LED	31	OSRAM OPTO	0,651	20.18
2 NPN Transistor	25	onsemi	0.221	5.53
10kΩ Resistor	4	EDGELEC	2.00	8
470Ω Resistor	4	EDGELEC	2.00	8
100Ω Resistor	4	EDGELEC	2.00	8
1uF Capacitor	10	KEMET	0.94	9.4

680uF Capacitor	1	KEMET	0.93	0.93
220uF Capacitor	1	KEMET	0.38	0.38
33uH Inductor	1	Bourns	0.86	0.86
1N5822 Schottky Diode	1	NTE Electronics	0.56	0.56
LM25-23B03 AC/DC Converter	1	Mornsun American	11.43	11.43
ESP32-WROOM-32E Microcontroller	2	Espressif Systems	3.00	6.00
A4988 Stepper Motor Driver	1	HiLetgo	1.98	1.98
28BYJ-48 Stepper Motor	1	HiLetgo	2.87	2.87
LM2596 Buck converter	1	Texas Instruments	3.20	3.20
12V Power Supply Adapter	1	GANGQI	9.00	9.00
Total Cost				102

Table 11: Schedule

	Major Deadlines	**Christelle**	**Sungjoo**	**Heonjang**
10/3	Design Review	Finalize PCB	Finalize PCB	Prepare a working environment (cloud setup)
10/10	**1st round PCBs**	Order Parts, Simulate Circuit	Finalize design and print 3D part	Implement a backend server
10/17		Soldering	Soldering	Soldering

10/24	Testing	Test the board, program the board to receive and send data	Test the board, program the board to receive and send data	Implement a frontend server
10/31	**2nd Round PCBs**	Order a second final board	Order a second final board	Integrate the software system with the hardware system and test
11/7		Soldering, testing	Soldering, testing	Implement statistics analyzer, soldering
11/14	**Mock Demo**	Prepare for Demo	Prepare for Demo	Prepare for Demo
11/21	**Break**			
11/28	**Final Demo**	Finalize adjustments for demo	Finalize adjustments for demo	Finalize adjustments for demo
12/5	**Final Presentation**	Finalize presentation	Finalize presentation	Finalize presentation

7. References

[1] Brumfield, Robin. (1992). Greenhouse Cost Accounting: A Computer Program for Making Management Decisions. HortTechnology. 2. 10.21273/HORTTECH.2.3.420.

[2] “The visible wavelength range and its impact on plant growth”, *Light Science Technologies*, https://lightsciencetech.com/visible-wavelength-range-plant-growth/

[3] Navvab, M. (2009, January). *Daylighting aspects for plant growth in interior environments*. ResearchGate. Retrieved September 30, 2022, from https://www.researchgate.net/publication/259043901_Daylighting_Aspects_for_Plant_Growth_in_Interior_Environments

[4] “LED Grow Lights for Plant Production” *OSU Extension*, https://extension.okstate.edu/fact-sheets/led-grow-lights-for-plant-production.html

[5] Xie, Ben & Goel, Pranav, & Wang, Honru. (2022) TimeTable Productivity Device. https://courses.engr.illinois.edu/ece445/getfile.asp?id=20494

[6] *Motors and Selecting the Right One*. Motors and selecting the right one. (n.d.). Retrieved September 29, 2022, from https://learn.sparkfun.com/tutorials/motors-and-selecting-the-right-one/all

[7] Allegro MicroSystems. (n.d.). *DMOS Microstepping Driver with Translator And Overcurrent Protection Datasheet*. Retrieved September 30, 2022, from https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf

[8] “Code of Ethics”, https://www.acm.org/code-of-ethics

[9] Ather, S. H. (2020, December 28). *How to calculate solar insolation*. Sciencing, https://sciencing.com/calculate-solar-insolation-8435082.html