CALIFORNIA STATE UNIVERSITY

SACRAMENTO

College of Engineering & Computer Sciences
6000 J St, Sacramento, CA 95819, USA

SOLAR LIGHTING

Team 8: The Solar Lighting Team

Team Members: Scott Carrick, Tony DiMichele, Dylan Lee, Amelia Vo

scottcarrick@csus.edu
tonydimichele@csus.edu
dylanchonglee@csus.edu
avo2@csus.edu

TABLE OF CONTENTS

TABLE OF FIGURES

Elevator Pitch

The Solar Lighting system integrates a solar tube with reactive lighting, allowing a room to be illuminated by natural light in conjunction with artificial light. The unit will maintain constant luminosity using sensory data and microcontrollers while recording and reporting the amount of energy saved. The system will be combined with a solar panel and charge controller to enable off grid use.

Executive Summary

Before designing any elements of the project described in this report, it was necessary to establish a clear issue to encompass our design. Each group member had pitched an idea to base the project around, but the idea that captured us the most was integrating natural light and using it to replace unneeded electrical lighting. That concept, coupled with the use of a solar tube, was the beginning of our project idea.

After establishing the societal issue we wished to address, we began to brainstorm the kind of features our product would have in order to actually impact the societal problem. The rough concept was to have a controller monitor and maintain the lighting in a space, and have that be set by the user. This meant we needed a sensor to monitor the lighting, a controller, some sort of mechanism to change the lighting, and an interface for the user to interact with the device. Additionally, we would need a space where we could install and test both our device and solar tube integration. Thankfully a team member had a small shed that we were given permission to make alterations to.

As we began the process of documenting our design idea, we chose to make our design solar powered, as it would further enhance the power saving capabilities of the product. A closure of some sort was also added to the features to ensure that the space could be made dark again if the user desired it to be. In order to interact with the system we opted to design a mobile application that would also report energy data from the system. Lastly, we added a physical switch to emulate the same experience a normal electrical lighting system produces.

With our features in place, we all began the process of designing our assigned components. In addition to this, we also had to alter the shed in order to install both the solar tube and solar panel to the roof. By the end of the first semester we had completed a proof of concept design in which all of our features could be demonstrated. Most features were built on breadboards and the system as a whole was not suitable to be installed in the shed.

For the second semester the design stayed the course and work was focused on completing the integration of all features. The controller board design was fabricated on a PCB and 3D enclosures were designed for all physical components of the project. The light sensor was redesigned to allow for it to be a battery powered stand-alone device. The application was beautified and more functionality was added to allow for it to report data from the controller and to send commands to the controller.

The device as a whole was completely installed into the shed and was tested for edge cases to ensure it would function as intended. The design passed all testing cases and proved to be a successful design. Power saving capabilities of our design were verified, confirming that our design did in fact address the wasted electrical lighting issue.

ABSTRACT — **Implementing a system that can react and adjust to sunlight can increase the efficiency of lighting. Adding solar tube technology, which can capture and tunnel sunlight from the roof, into a desired area, allows for solar light to be utilized as a light source. This is beneficial in that the energy that is currently being used to light residential and commercial buildings is not utilizing natural light, allowing energy savings of up to 30%. Implementing a novel system which can automatically adjust artificial light in response to the luminosity of natural light can be used to reach this desired effect and energy savings. Light-to-digital sensors provide feedback data which in turn controls two motors, one which adjusts the brightness of the LED, and another which partially covers the sun tube via a flap. The designing of such a product is discussed in detail in this report, including planning, execution, and testing of the system.**

Keywords—automation, solar tube, microcontrollers, artificial light, natural light, control system, testing

I. INTRODUCTION

When it was first discovered, the use of a lightbulb allowed daytime to be extended by artificially recreating the light from the sun. While this is still a major use of this technology, lightbulbs can be found illuminating rooms during daylight hours. What once was an extension of the day, has become a replacement for it. Artificial lighting is a significant source of energy consumption in the United States, and in this report, we will demonstrate our plan to reduce its strain on the grid. This will be accomplished by utilizing natural light and adjusting the level of artificial light autonomously, ensuring the user has a consistent lighting experience whilst consuming much less electrical power. The design detailed in this report uses a combination of sensors and a controller to monitor and adjust the amount of electrical light in a space. Additionally, a solar tube will be used to increase the amount of daylight added into the total ambient light of the room. This design will help reduce the amount of energy used to light a room whilst also increasing natural light.

In this project, a work breakdown structure assists in breaking a complex project or idea into definitive features. This allows those who are working on the project to better estimate time of completions as well as identify the best route of actions. In goals that can then be assigned and completed by the group. Organizing the project this way increases efficiency whilst also forcing the team to analyze all aspects of the project and how they need to be completed, to prevent unforeseen complications in the design process.

In addition, a detailed timeline aims to structure the project by sub activities and assess the time needed to complete each activity. The timeline uses the work breakdown structure to further detail when the project should be completed in an effort to reduce any unforeseen time crunches. While the timeline indicates the timing and completion dates of all tasks, the milestones of the project are also used to track team progress. These milestones are directly related to the work breakdown structure and allow the team to recognize definitive and testable progress.

In this report, we will define the design idea and technologies needed. Also, we will detail the needed resources, project budget, prototype features, and measurable metrics of our project.

After completing the assembly of different aspects of this project, the next necessary step is to test the functionality of the unit. These tests will ensure that the product is fully functional and can withstand the expected normal use. While it is worthwhile to test the product as a whole to ensure it works as intended, this test plan breaks the project into testable units to ensure functionality in each feature.

This report details the testing plans for the automatic lighting control, the user application, the physical switch, the solar tube flap, and lastly the solar power integration. These tests will cover the function of the individual system as well as its

integration points with other features of the product.

II. SOCIETAL PROBLEM

A. Energy Consumption From Lighting

One of the challenges facing modern societies is the ability to effectively and efficiently supply power to all. As the population continues to exponentially increase, the need for solutions to power's efficiency, storage, and creation are ever present. While there are many ways power is utilized, lighting is a significant usage of energy in the United States.

A report by the U.S. Energy Information Administration indicated that the U.S. uses "219 billion kilowatthours (kWh) of electricity for lighting" which accounts for "6% of total U.S. electricity consumption" [1]. While this is not the largest sector of energy use, it is an area where improvements can be made and a more stable grid can be realized. Furthermore, Fig. 1 shows that the commercial industry itself uses around 10% of its total energy on lighting alone, while maintaining operational hours that mainly coincide with sunlight hours. For these reasons, we believe that incorporating sunlight as a main resource for interior lighting can better equip our growing societies energy demands.

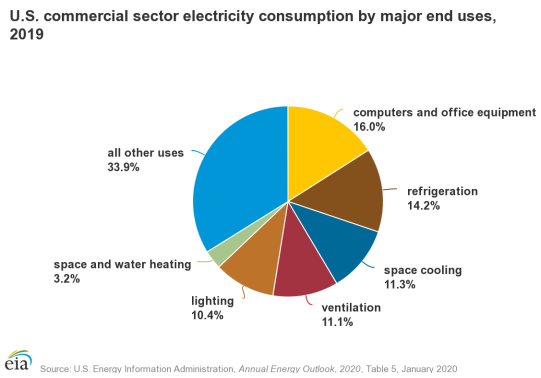


Fig. 1. U.S. Commercial Sector Electricity Consumption.

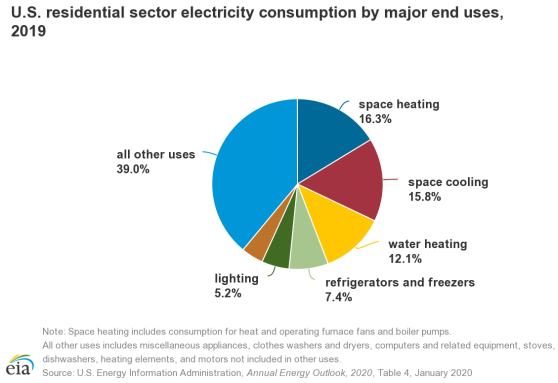


Fig. 2. U.S. Residential Electricity Consumption.

Using a system that can adjust the amount of artificial light through sensory data can minimize the energy necessary to illuminate a room. In fact, the effort to control lighting in relation to the amount of sunlight present can "save lighting energy use by up to 30%" [2]. If we compare that value with the national usage of 219 billion kWh of energy for lighting, the U.S. can save up to 66 billion kWh of energy per year. The savings in energy usage can help ensure that electrical systems have the necessary reserves to protect from unforeseen complications, such as inclement weather.

B. Power Grid Reliability

Maintaining a power grid requires a certain amount of flexibility, as demands in power fluctuate with time of day as well as the seasons. While lighting usage does change in accordance with the time of day, as the seasons change the lighting needs are not all that affected. In contrast, as the weather changes, other forms of energy consumption such as space heating and cooling increase.

In August of 2020, California saw record high temperatures, which in turn created peak demands for energy that were beyond what the current system could handle. In response, CAISO, a non-for-profit group tasked with managing the electrical systems in California, had to introduce rolling black outs to prevent damage to the infrastructure. These black outs are illustrated in the tables below.

Table 3.1: CPUC-Jurisdictional Customers Affected by August 14 Rotating Outages

	Customers	CAISO-initiated rotating outage (MW)	IOU actual response (MW)	Time (in mins)	Start	Finish
SCE	132,000	400	400	63	6:56 PM	7:59 PM
PG&E	300,600	460	588	~150	6:38 PM	~9:08 PM
SDG&E	59,000	71.6	84	~15-60		
Total	**491,600**	**931.6**	**1,072**	**15 to 150 mins**		

Fig. 3. CAISO Rolling Blackout Data from August 14th,2020[3].

Table 3.2: CPUC Jurisdictional Customers Affected by August 15 Rotating Outages

	Customers	CAISO-initiated rotating outage (MW)	IOU actual response (MW)	Time (in mins)	Start	Finish
SCE	70,000	200	200	8	6:43 PM	6:51 PM
PG&E	234,000	230	459	~90	6:25 PM	~7:55 PM
SDG&E	17,000	35.8	39	~15-60		
Total	**321,000**	**465.8**	**698**	**8 to 90 mins**		

Fig. 4. CAISO Rolling Blackout Data from August 15th,2020[3].

According to Fig. 3 and Fig. 4, the amount of energy that was cutoff in order to protect the power grid was 1072 MW and 698 MW respectively. Using the estimate that 6% of energy is used by lighting as stated by reference [1] as well as the estimate in reference [2], that 30% of lighting power can be saved using a reactive system, more than enough energy would have been saved to have prevented these black outs.

While the California rolling black outs was just one instance of infrastructure failure, many more events like these happen all across the nation. Lighting may not make up a drastic amount of energy usage but by increasing the efficiency of these systems, the power grid can operate more effectively.

C. Artificial Light vs. Natural Light

According to the US EPA (United States Environmental Protection Agency), many humans in modern cities spend up to 90% of their lives indoors [4]. This can have adverse effects on an individual's health as it can affect the levels of vitamin D within the body. Vitamin D deficiency is one of the most common conditions worldwide with more than 1 billion people at risk [5].

The major cause of this is lack of vitamin D in the individual's diet or from insufficient sun exposure. A few of the side effects caused by a lack of vitamin D are improper formation of bones, growth retardation in children, as well as increasing the risk of autoimmune, cardiovascular, and infectious diseases [5][6].

Using natural light during times when artificial lighting is unnecessary can positively impact an individual. According to a report published in the Building and Environment Journal, work was done to see employee's thoughts on using DGS (Daylight Guidance Systems) in the core of a building. The findings concluded that with increased daylight penetration due to DGS, well-being, mood, and productivity were positively influenced [7]. Additionally, those who had perceived their lighting as being higher quality rated their space as more attractive. This led to a more positive mood and a greater well-being at the end of the work day [7].

III. Design Idea

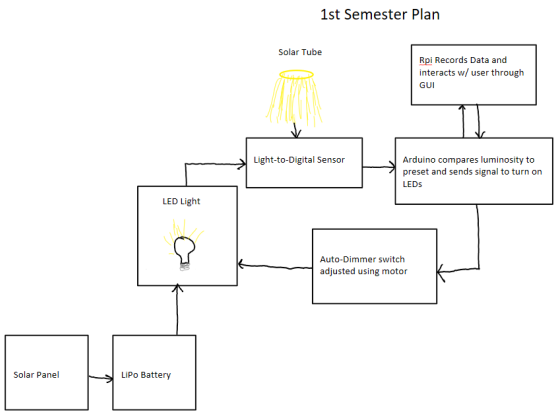


Fig. 5. Block Diagram of The Solar Light Design

A. First Semester Design

The Solar Lighting system will be controlled using an Arduino and Raspberry Pi. A GUI system will be available for users to control brightness. On normal days, the tube will use direct sunlight as its main source for lighting the space. If the sun is too bright, the flap will be closed to reduce luminosity. For power, the system will utilize a solar panel and battery. On

cloudy days and at nighttime the solar panel, which is installed on the roof, will charge the battery which will be used to power the LED.

Fig. 6. Solar Tube Illustrated

The top of the tube that is placed above the roof is shaped as a hemisphere, which helps capture sunlight in any direction. The inside of the tube is made of reflective material, one-way acrylic mirror film. It allows captured light to move through the tube and maintain brightness. In addition, the film will block UV rays from the sun and only let natural sunlight into the space, creating a comfortable and safe environment to stay in all day.

The tube will be installed on the roof of a shed. One ambient light sensor will be placed inside the space to measure the luminosity. Users will be provided a GUI app to control the light as they want or leave it on auto mode.

B. Second Semester Design

Initially, the control program allowed for the stepper motors to consume about a half an amp to keep the stepper motors at their current position. By programming the MCU properly, the stepper motors will be powered off while not in operation. This helps our system last much longer since the entire application is meant to run off-grid and helps make our product less wasteful and more eco-friendly. In addition, the control program will be further improved to have faster stabilization times.

In an effort to make the system usable as well as scalable, it is important to create a light sensor that is able to run on self-contained battery power. This change to our design comes after the realization that other products that are currently out in the market include battery powered sensors. Our current design requires the light sensor to be connected to the main power of the structure, limiting the location of the sensor and increasing the installation difficulty. In a Medium article about product design, they write that "good products don't take up more space than they need to. They fit elegantly into the space they belong" [8]. If our goal is to decrease the amount of energy used in lighting, our product must be usable to a large group of people from different backgrounds. This understanding demands that we simplify the product as much as possible to allow for the user to easily install and use it, thus requiring an out-of-the-box usable sensor.

Another improvement on our design is the beautification and increased functionality of the user interface. The application will be able to turn the system off and on, as well as display useful data to the user. Currently that data consists of the amount of energy saved per month, and in turn, how much money the system saved the user.

Lastly, the system as a whole will be made more marketable by converting the breadboard circuits to PCB designs and enclosing them in 3D-printed units. The flap will be constructed out of plastic and will be contained in the same unit with the flap motor.

IV. Work Breakdown Schedule

The work necessary to complete the design of the Solar Lighting project can be divided into its distinct features. From there, the features can be broken down into smaller action items that are needed in order to complete the given feature.

The Solar Lighting system contains five main features, including automatic lighting control, a

user application, a physical switch to locally control the system, a flap to control the amount of solar light allowed into the space, and finally the ability for the system to be solar powered. Additionally, documentation is required to organize this project and to detail the steps taken on the journey to completing it.

A. Automatic Lighting Control

The overall goal of the automatic lighting system is to provide constant luminosity to a room at a fraction of the power necessary with purely artificial lighting. This can be achieved by auto-adjusting LED strips with a motor controlled by an Arduino Nano being supplied with luminosity measurements from a lux sensor. The entirety of the control system integration and development should take about 45 hours.

1. Measure Lighting

A Lux sensor will be placed into a corner of the shed and will be transmitting its measurements via bluetooth to the Arduino Nano. This parameter will be arbitrarily set by the user, and will serve as the set-point luminosity for the overall control scheme. This sensor will be powered by an Arduino Nano that is attached to it, and run directly off the 12V battery. Setting up the lux sensor will be done by Scott and should take around 12 hours.

2. Control Program

Lighting measurements will be fed into the Arduino and fed through the control logic program in order to actuate the dimmer switch motor. This system will autonomously control the overall amount of light in the shed at any given time relative to user defined parameters.

An off signal from the touch switch will also trigger the flap to close, and an on signal will trigger the flap to open. This helps prevent unnecessary heating of the shed when not in use. The control program and electronics will be developed by Tony and should take around 20 hours.

3. Motor Actuation

The circuit's overall power source will be a 12V lead-acid battery and the motors being used also run off of 12V. As such, no power step up/step down is required, but some actuation logic is. A DC 2-Phase motor controller will be used to control the actuation of the rotor.

The flap controls will be power regulated using the same controller setup, but will have two states only, which will be actuated by the motors (fully open or fully closed.) These systems will be developed by Tony and should take around 15 hours.

4. Electrical Lighting

The shed will need to be wired for power distribution and the various sensors, microcontrollers, and LEDs placed in their appropriate places. This will be primarily planned out by Dylan and the whole team will assist him with actual construction. This will take around 3 hours to complete.

B. User Application

The main controller of the system is an Android app which can accept user input and display the current settings. The user can toggle between text or slider input and will also display what the current light levels of the room are as well as what the user has the system set to. This feature should take about 40 hours to implement.

1. Building the Application

The application will be built using Java in Android Studio. The app will be running on Android version 11 with wifi as the means to communicate with the RPi. The base of the application should take around 10 hours and will be completed by Dylan.

2. Communication with RPi

The Raspberry Pi board will be running its own application that is waiting for input from the user or updates from the rest of the system based on the level of light in the room. If a message comes from the application, then the Raspberry Pi will take the information and translate it into something usable by the system. Once the translations are done the information will be sent to the system and changes will be made accordingly. The Raspberry Pi will then send the information to the application over wifi so the user can see it. This is programmed by Dylan and should take at least 20 or 30 hours to implement.

C. Physical Switch

In this part, the elements include Arduino, physical switch, LEDs, and a solar panel. The touch capacitive sensor switch is used to control the system's on/off state. The whole process is done by Amelia. This should take around 20 hours to complete.

1. Control program integration

The touch capacitive sensor switch is used to implement a circuit to control the system. In normal state, the module outputs low, but when the switch is touched, the module outputs high and changes the state of the system.

2. Switch hardware

For the physical hardware, a touch capacitive switch will be connected to the controlling Arduino. Arduino pins are used to power the switch control circuit. The whole system is powered by solar energy.

D. Solar Tube Flap

While the light coming from the solar tube will greatly assist in lighting the room, at times the user may want the room darker. In order to accomplish this, a flap will be designed in order to cover the solar tube to prevent unwanted light from coming into space.
This feature consists of three action items, including the design of the flap itself, the software integration with the control program, and the motor actuation circuit to drive the flap. Overall, the feature should take around 40 hours to implement.

1. Flap Design

For the design of the flap, the specific shape and the materials needed will be determined by Scott, with input from the rest of the group. The flap will need to take into account the stepper motor that is driving it, as well as the shape of the solar tube. The flap should be able to fully block the solar tube light allowing the space to return to its lowest lighted state. A completed covering is expected after 10 hours of work.

2. Control Program Integration

Since the flap will alter the amount of light in the space, it is necessary for the control program to integrate this change into the software. This will allow the system to actuate the flap if the light in the space is too bright for the user. Tony will be responsible for writing the necessary code to implement the flap with the control program. This task is expected to take close to 20 hours to complete, including testing and all necessary alterations.

3. Motor Actuation Circuit

The final component of the flap feature is designing the motor circuit and connecting it to the flap. This will allow the control program to actuate the flap and complete the necessary changes to the lighting to meet the users needs. This circuit will be completed by Tony and should take around 10 hours to complete.

E. Solar Power

A solar panel and battery will be used to power the entire system, including the lights and all microcontrollers. This part is completed by Amelia and should take around 25 hours to complete.

1. Hardware

The Solar panel connects to the charge controller which charges the battery. The battery then connects to the rest of the project to power the system.

2. Integrating with system

The solar panel is getting the energy from the sun and powering the whole system which includes: 2 Arduinos, a Raspberry Pi, 2 motors, a switch circuit and lighting. For lighting, the solar panel will charge the battery to power the whole circuit. The solar panel will allow the battery to consistently have enough charge to power the system.

F. Documentation

In order to keep track of the progress and maintain the necessary organization for this project, documentation is vital. Throughout the project, reports will be completed to document different aspects of the project, including the identification and explanation of a societal problem as well as the schedule of when activities need to be completed to meet the project's deadline.

1. Identifying Societal Problem

Before designing a project, it is important to determine the need for the design. In this project, we focused on societal issues and documented the most pressing societal issue that we wish to address for this project.

2. Design Idea Contract

After identifying the social issue the group wants to focus on, a technical idea is produced in order to address the societal problem. This idea is then documented in a contract in which the project is described and the feature list is solidified. This report will be the basis for determining whether or not the project is successful.

3. Work Breakdown Structure

In order to achieve all wanted features in our design, it is crucial to further dissect what each feature needs in terms of actions or activities. This allows the project to be organized in a clear working structure and assist in making a realistic timeline for the project's completion.

4. Project Timeline

In this report, each feature described in the work breakdown structure will have defined start and end dates to facilitate an organized timeline to the project's completion. In addition, milestones will be defined to keep track of project completion.

5. Risk Assessment

With each feature, as well as the project as a whole, there are certain risks that could prevent us from completing the project. To mitigate these we try to identify possible risks and make plans for mitigation such that the project can still be completed.

6. Public Prototype Presentation

At the halfway point of this project, the group will assemble visual aids as well as a video to present to the public demonstrating the prototype we've designed. The visual aids will assist in describing the features of our product.

7. Revised Societal Problem

After completing the prototype, the team will look back at the societal issue and make any necessary changes to ensure that the product that is produced, addresses the societal issue defined at the beginning.

8. Device Test Plan

With the prototype complete, the team can now assemble test plans to ensure the product will work as described in conditions that are

expected. This will assist in refining the design and making it close to market designs.

9. Market Review

Before launching a product into a market, a price must be considered. This price is contingent on the current market for products that are similar to ours. The report will detail what products currently exist as well as where are product lies in the

10. Feature Report

This report details the function of each feature of our project, describing the individual function and how the features combine to make a complete project.

11. Final Documentation Report

This report will be a culmination of all the documentation that took place throughout the project. It will detail each step of the design process as well the necessary reports, such as the risk assessment and market review. This will be a supplement to our design as a comprehensive explanation to how our project was completed.

V. GANTT CHART FOR SPRING 2021

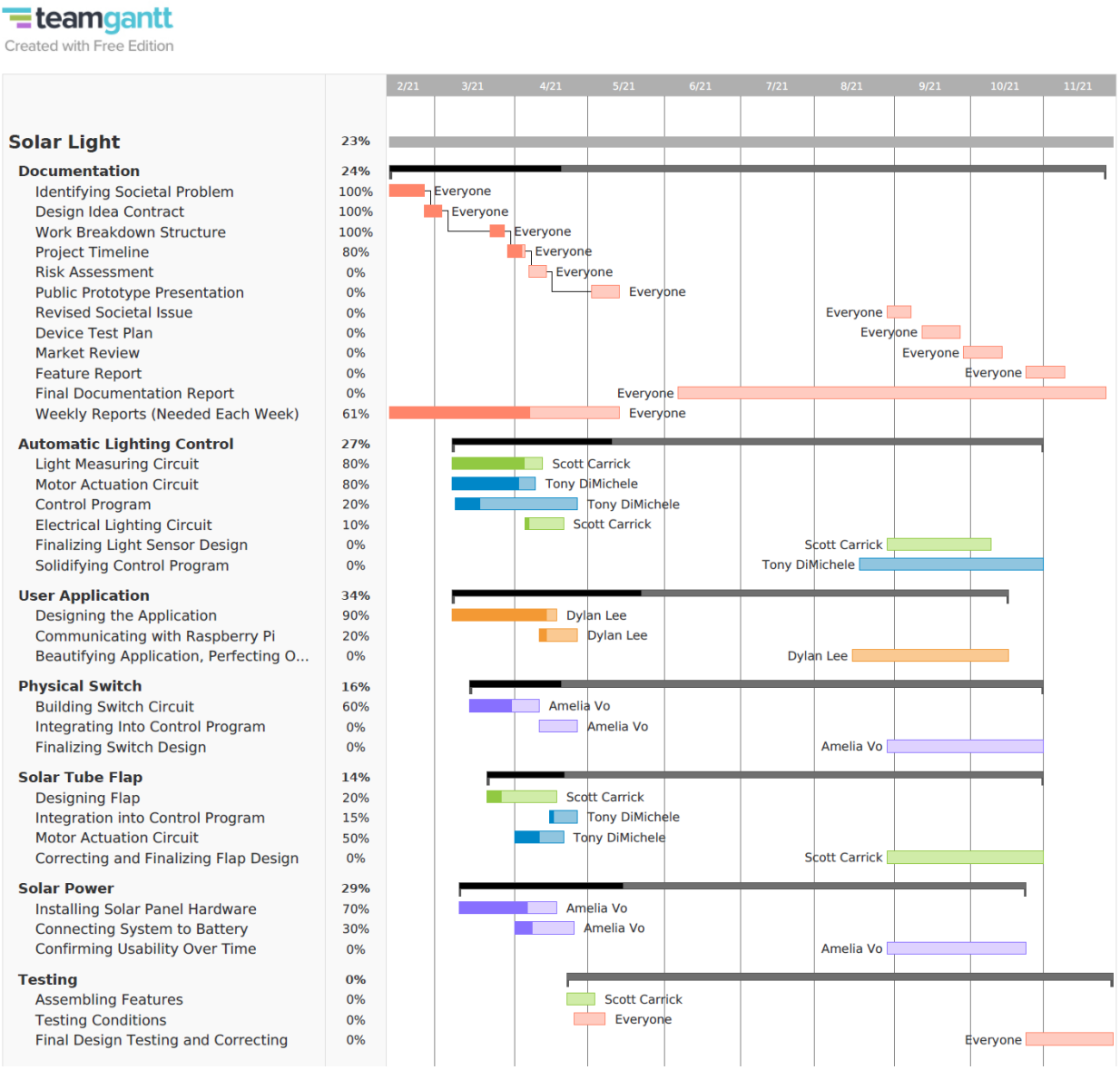


Fig 7. Gantt Chart for Spring Semester

VI.GANTT CHART FOR FALL 2021

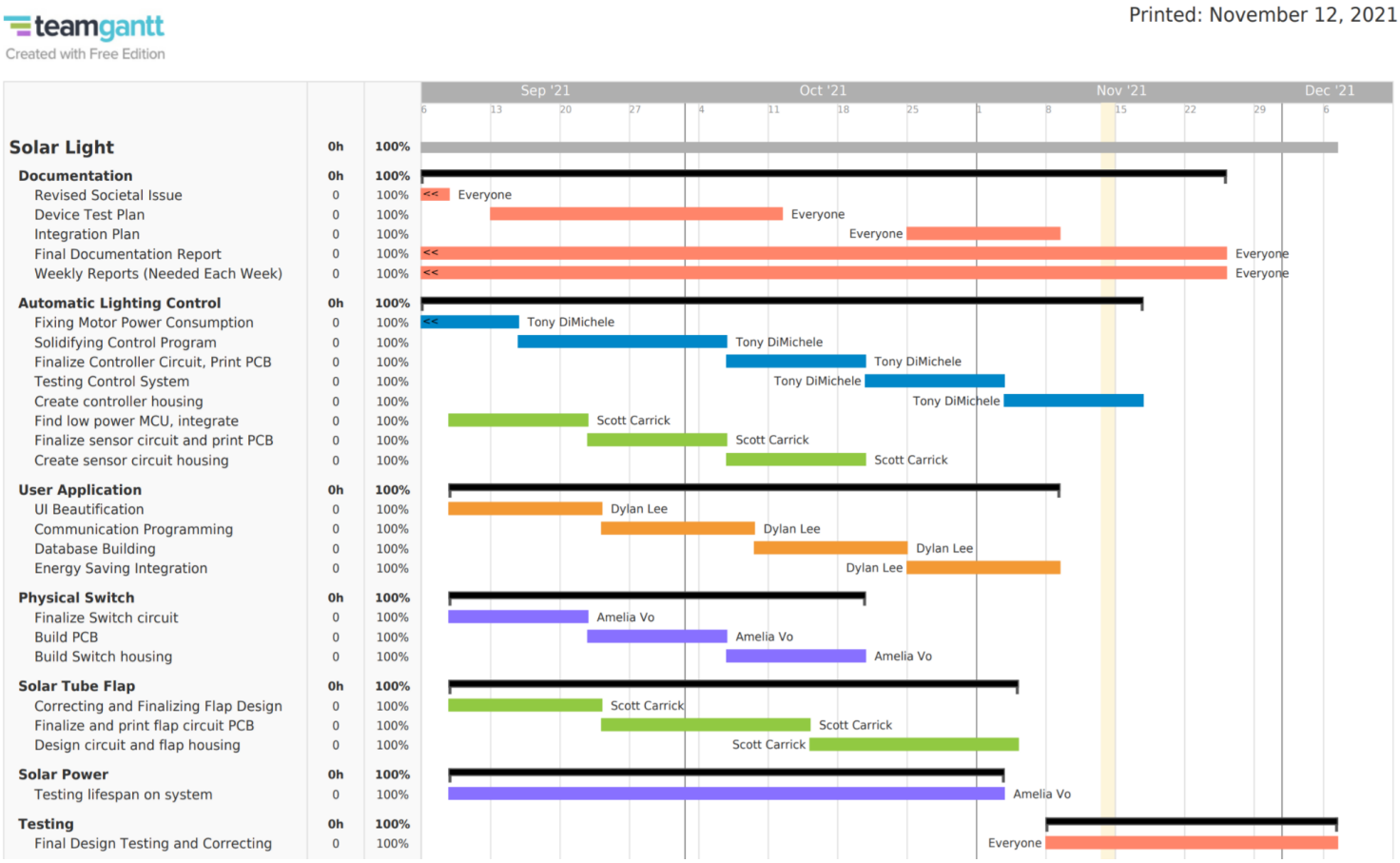


Fig 8. Gantt Chart for Fall Semester

VII. Timeline

The milestones listed in this report aim to identify recognizable accomplishments in the process of designing our project idea. These accomplishments are strongly related to the work breakdown structure and are used to confirm the completion or success of tasks performed by the group. The progress the team makes in the design can be compared to the amount of milestones accomplished, with a finished design having passed all milestones.

A. Installation of Solar Tube

The solar tube is an integral part of the project even though it is completely non-electric. The addition of natural lighting helps to curb total energy costs outright, and provides pleasant light to the room. This milestone was already achieved by the team in early March, and was found to be very effective at providing light to the room. Since this is completed, we can now focus on the more labor and intellectually intensive portions of the project requiring electrical engineering skills.

B. Measuring Accurate Light Data

In order to automate the lighting in a space, the controller will need to obtain an accurate measure of the current light level. Therefore, a significant moment in this project will be when the light sensor is able to measure the light accurately and repeatedly.

This milestone should occur early in the project as the completion of the controller program hinges on accurate light data. Reaching this point will allow us to enhance the design of the light sensor to make it more marketable and functional.

C. Controlling Motor Output With Incoming Data

The whole point of the project is to have some sort of control over the motors to influence the amount of light shining through the tube as well as the brightness of the electrical lighting. The motors will be connected to a controller that gets data from both the user and the light sensors and will actuate the tube flap or light brightness as needed.

The implementation for this milestone should be fairly early on but refining the process will take much more time to make it more robust and accurate.

D. Installing All Hardware

A major milestone in the series of events will be once all of the hardware for the project is installed completely. When everything is finally put together, then all of the measuring of data and programming of hardware can finally be fully measured and used accordingly.

This milestone will happen at the midpoint of the project as most of the actual work that needs to be done hinges on many controllers communicating at the same time. Additionally, the controllers and sensors are being designed modularly and thus will be connected once operating correctly individually.

E. Application Correctly Receives and Sends Data

As the main source of interaction with the system, the application needs to be first able to send data to the Raspberry Pi. The user will then see how their input affects the system and the updated values from the RPi will be sent back to the user.

This should occur once all hardware is installed and the proper modules and functions are programmed into the RPi and the application which should be somewhere in the middle of the whole project.

F. Switch Circuit Correctly Interrupts Program

A needed feature in our design was the users ability to control the system directly and without the need of the application. This means that a manual switch will need to be designed and its signal must properly interrupt the control program to accomplish the desired result.

The switch circuit can be built in a modular fashion but will need to be integrated into the control program software, thus a working program

will need to be completed before or alongside this milestone.

G. Flap Actuates Correctly in Response to Data

A significant feature in our design is a flap that will cover the solar tube in order to reduce the natural light as needed by the user. Getting the designed flap to be actuated by the control program will be a defining moment in the project. While the control program and motor actuation circuits can be built without the flap, getting to the point where the flap automatically closes will signify successful integration of several different tasks.

H. System Functions Properly Off Battery

With all components of our design operating off of a solar panel and battery, the completed design must be shown to be operable for a considerable amount of time. This accomplishment will verify the usability of the system operating completely off the power grid, an important aspect of our design.

This milestone will be completed near the end of the design phase, as all the elements of the project must be working and connected to battery power to pass this milestone.

I. System Completes Necessary Testing

One of the most obvious milestones in this project is when the system, after being fully assembled, responds to the testing parameters successfully. This will mark a completed project that can then be further beautified to prepare for the market. This milestone will most likely occur close to the end of our projected timeline.

VIII. FUNDING PROPOSALS

Quantity	Description	Cost per unit	Total	Purchased	Purchased By
2	Bluetooth (BLE) HM-10 Board	$9.99	$19.98	Yes	Scott
1	Lux sensor I2C Adafruit	$10.49	$10.49	Yes	Scott
1	Solar LightBlaster for Shed	$135.97	$135.97	Yes	Dylan
2	Stepper motor	$16.65	$33.70	Yes	Tony
1	Touch Capacitive Sensor Switch	$7.49	$7.49	Yes	Amelia
1	Solar Panel Kit	$49.99	$49.99	Yes	Amelia
2	Stepper Motor Controller	$8.46	$16.92	Yes	Tony
1	Dimmer Module	$9.99	$9.99	Yes	Tony
1	12V 20Ah battery	$36.90	$36.90	Yes	Tony
1	Arduino Nano Boards (pack of 3)	$13.99	$13.99	Yes	Scott
1	LED Light Strip	$16.99	$16.99	Yes	Scott
1	Screws			Yes	Tony
1	Rubber Washers			Yes	Tony
1	Electrical Tape			Yes	Tony
1	Arduino Uno	$24.27	$24.27	Yes	Tony
1	Stepper Motor Mounting Bracket	$8.96	$8.96	Yes	Scott
1	Nuts and Bolts	$1.00	$1.00	Yes	Scott
20	Wire	$0.49	$9.80	Yes	Scott
20	Wire	$0.49	$9.80	Yes	Scott
1	Sheet Metal	$7.74	$7.74	Yes	Scott
1	Ace Trip	$8.62	$8.62	Yes	Tony

Fig 9. Materials Required

Purchase History	Description	Purchaser	Total
3/10/2021	Solar Tube	Dylan	$146.51
3/11/2021	Lux sensor and Bluetooth boards	Scott	$32.82
3/17/2021	Motors	Tony	$33.70
3/19/2021	Switch and Solar Panel	Amelia	$62.51
3/20/21	Roof sealant and caulk gun	Scott	$11.43
3/28/2021	Arduino Nano boards	Scott	$15.07
3/30/2021	LED Light Strip	Scott	$18.31
3/28/2021	Dimmer Module	Tony	$9.99
3/31/2021	Stepper Motor Controller	Tony	$16.92
3/31/2021	12V 20Ah battery	Tony	$36.90
4/3/2021	Ace Hardware	Tony	$12.00
4/15/21	Arduino Uno	Tony	$24.27
4/15/2021	Stepper Motor Brackets	Scott	$9.65
4/17/2021	Hardware; Nuts, Bolts, Wires	Scott	$22.11
4/19/2021	Sheet Metal	Scott	$8.24
4/25/2021	Ace Hardware	Tony	$8.62
4/17/2021	Food	Dylan	$38.77
9/9/2021	Bluetooth Module	Scott	$10.38
9/26/2021	ATmega328 and components	Scott	$32.28
9/26/2021	Flap components	Scott	$22.24

Fig 10. Materials and Costs for Spring and Fall 2021

IX. Integration Plans

A. Solar Tube

The solar tube was installed and tested for leaks early on in the project. Since this feature has no moving parts and no electrical components, after installing and checking that the light came through and didn't leak, the integration for the solar tube was more or less finished.

B. Solar Panels, Solar Controller, and LEDs

Our team purchased an "off the shelf" solar panel and controller. The panel was mounted on the roof of the shed and the controller was mounted on a wall inside the shed. Wires were run from the panel to controller and a battery was purchased capable of receiving charge from the control unit. Since this system was purchased from a manufacturer, there was very little debugging needed in order to ensure this portion of the system was functioning properly.

The LED strips were then soldered according to our needs and mounted on two of the 2x4 beams inside of the shed. The entire system was then run into the "Load" terminal of the solar controller and functioned perfectly.

C. Control Box

The control box was comprised of an Arduino Nano, 2 DRV8833 motor controllers, one side of a HM-10 bluetooth serial module, a buck converter for stepping down the voltage of the battery to 9-V (the working voltage of the motor controllers), and the rotary dimmer switch which was connected to a stepper motor via a motor shaft coupler. The LED strips were then connected into the load terminal of the dimmer, and the motor was actuated and found to be able to hold its position solely from the mechanical friction of the system. This allowed the motors to be turned completely off after actuation and a small settling time which allowed the motors to settle. The same was also found to be true of the flap motor, which was controlled by the same box. It required us to use cardboard as the flap material and to orient the flap box in a specific way in order for the mechanical friction of the flap to maintain its position as well when the motor to it was also turned off. Being able to turn the motors off was critical as its holding current was extremely high and wasted too much power.

D. Solar Tube Flap

A housing made from PETG was modeled and 3-D printed and installed on the ceiling of the shed. The flap itself was made from cardboard since it is completely opaque but also very light and rigid enough for our application. The flap was then coupled to the second motor which was secured within the 3-D printed housing. The flap was controllable by turning on and off the system and was found to be functioning correctly.

E. Light Sensor

The light sensor, a VEML7700, was mounted on the lid of a 3-D printed enclosure and was the only transducer we used in this project. It collected the ambient light level of the room and converted it into a digital value, which was then sent via bluetooth serial to the other bluetooth module and sent to the Arduino Nano. The module was battery powered and portable so that the ideal location in the room could be found to sense the combination of light from the solar tube and from the LEDs.

F. User Application

The user application designed for Android was the way our project changed parameters inside the control program, allowing a user to adjust the light level in the room and turn the system on or off. The application would talk to a Raspberry Pi inside the shed and send light level parameters or on and off signals, which would then be serially

communicated via USB to the Arduino Nano inside the control box. The Nano would then send back the on/off state, the ambient light level, and a step position of the motor controlling the dimmer. The ambient light level would then be displayed to the user, and the stepper motor position would be used to estimate the power usage currently being used by the system based on experimental data.

G. Physical Switch

A physical switch was also implemented into the system so that a user could turn the system on and off inside the shed without the need of an application. A touch switch was chosen as it allowed the control program to simply change state instead of breaking the power connection. That way the app or switch could be used in conjunction with one another to turn on/off the system. The switch was connected to the controller board PCB with wires.

Conclusion

The following systems were modularly implemented into the Solar Lighting system in this order and were checked via experimentation to ensure that all the parts continued functioning as a cohesive project after each additional layer of complexity was added. This process of integrating a part, and then checking for functionality is an important step to make sure that all the pieces will work in a controlled sequence without error.

X. Device Test Plan

A. Automatic Lighting Control

1. Necessary Tests

The tests required for the lighting control actuation are: ensuring motor response to changes in ambient lighting, switch hardware interrupts/PCB verification, making sure the stepper-motor is properly tracking it's rotation and are turning off after actuation (no holding current), and finding a suitable location for the lux sensor to transmit the best possible light data to the control board via the bluetooth modules.

2. Test Timeline and Test Assignment

Test	Performed by	Date
Motor Response to Changing Light Parameters	Tony	11/03/2021
PCB handles all data and interrupts	Tony	11/03/2021
Program is properly tracking stepper motors and motors are not using holding current	Tony	11/03/2021
Bluetooth Send/Receiving Light Data	Tony	11/03/2021

Fig. 11. Automatic Lighting Control Test Timeline

B. User Application

1. Necessary Tests

The tests for the application include the following functionality: on/off button, the ability to edit the light setting with a custom or preset setting, and showing the user extra details such as energy/money saved. All of these tests should affect the backend database values and the changes should be shown to the user as they occur.

2. Test Timeline and Test Assignment

Test	Performed by	Date
Application affecting all database values	Dylan	11/03/2021
Application showing values to user	Dylan	11/03/2021
User input affecting the system	Dylan	11/03/2021

Fig. 12. User Application Test Timeline

C. Physical Switch

1. *Necessary Tests*

The physical switch test is to make sure it can control the whole system ON/OFF. Once the switch button is pressed, it triggers the system to turn ON and press again to turn OFF. When the system is OFF, the user application will not be able to interrupt it.

2. Test Timeline and Test Assignment

Test	Performed by	Date
Switch works as interruption	Amelia	11/03/2021
Switch turn ON/OFF	Amelia	11/03/2021

Fig. 13. Physical Switch Test Timeline

D. Solar Tube Flap

1. Necessary Tests

To confirm the proper functionality of the solar tube flap there are two main categories of testing. First, the correct software testing must be done to ensure that the flap is receiving the correct signals at the expected times. Second, the hardware and mechanics of the flap must be tested to ensure long term usage and for the effective result of blocking the solar light.

Using a variety of situations, we will test the software to make sure the flap actuates when needed. The points of interest include when the system turns on and off, through either the application or physical switch, and when the flap is actuated by the control program or user application.

The hardware and mechanics of the flap will be tested for strain to ensure the stepper motor can consistently operate the flap without fail. In addition, the flap will need to consistently maintain its ability to fully cover the solar tube and block the sunlight.

2. Test Timeline and Test Assignment

Test:	Performed by	Date
Software integration with application	Scott	11/03/2021
Software integration with control program	Scott	11/03/2021
Consistent ability to block sufficient light	Scott	11/03/2021
Mechanic functionality	Scott	11/03/2021

Fig. 14. Solar Tube Flap Test Timeline

E. Solar Power

1. *Necessary Tests*

Characterize the solar controllers power output when directly connected to the lights versus when our system is integrated into the lights. Find the expected power savings you should get out of the system, and document in the report.

2. Test Timeline and Test Assignment

Test	Performed by	Date
Battery integrates with the system	Amelia	11/03/2021
Battery goes the standby state when it's not used	Amelia	11/03/2021

Fig. 15. Solar Power Test Timeline

Conclusion

The use of a solar tube in conjunction with a reactive lighting system can help reduce the power used in an office or residential building, increase productivity and mental wellness, and save money by reducing the cost of lighting. Our product, the Solar Lighting system is a successful prototype of a system which can achieve these goals through the means discussed in this report.

Further reduction of power usage could be achieved by using more appropriate microcontrollers, different dimming actuation means (say purely electronic for example), more efficient battery storage, or even less computationally heavy code. Still, given these areas of improvement, the Solar Lighting system is a successful proof of concept that such a reactive lighting system is viable. Further research and engineering would be necessary in order to make this product completely market ready, but as it stands it completes the tasks we set out to achieve.

References

[1] U.S. Energy Information Administration, "How much electricity is used for lighting in the United States," U.S. Energy Information Administration, Feb. 3, 2021. [Online]. Available: https://www.eia.gov/tools/faqs/faq.php?id=99&t=3. [Accessed: Feb. 20, 2021].

[2] G. Y. Yun, H. Kim, and J. T. Kim, "Effects of occupancy and lighting use patterns on lighting energy consumption," in Energy and Buildings, Vol. 46, pp. 152-158, 2012.

[3] California Independent System Operator, "Preliminary root cause analysis mid-August 2020 heat storm," California Independent System Operator, Oct. 6, 2020. [Online]. Available: http://www.caiso.com/Documents/Preliminary-Root-Cause-Analysis-Rotating-Outages-August-2020.pdf. [Accessed: Feb. 20, 2021].

[4] United States Environmental Protection Agency, "Indoor air quality," EPA, [Online]. Available: http://www.epa.gov. [Accessed: Feb. 18, 2021].

[5] M. F. Holick, "Health benefits of vitamin D and sunlight: a D-bate," in Nature Reviews Endocrinology, Vol. 7, pp. 73-75, 2011.

[6] M. N. Mead, "Benefits of sunlight: A bright spot for human health," in Enviroment Health Perspectives, Vol. 116, Issue 4, pp. 421-572, April 2008.

[7] M. S. Mayhoub, and D. J. Carter, "The costs and benefits of using daylight guidance to light office buildings," in Building and Environment, Vol. 46, Issue 3, pp. 698-710, 2011.

[8] J. Davidson, "The 10 Commandments of Good Products," Medium, Aug. 28, 2018. [Online]. Available: https://medium.com/swlh/the-10-commandments-of-good-products-d1d0a97b30ee. [Accessed: Sept. 9, 2021].

APPENDIX

A. HARDWARE

1. Block Diagram

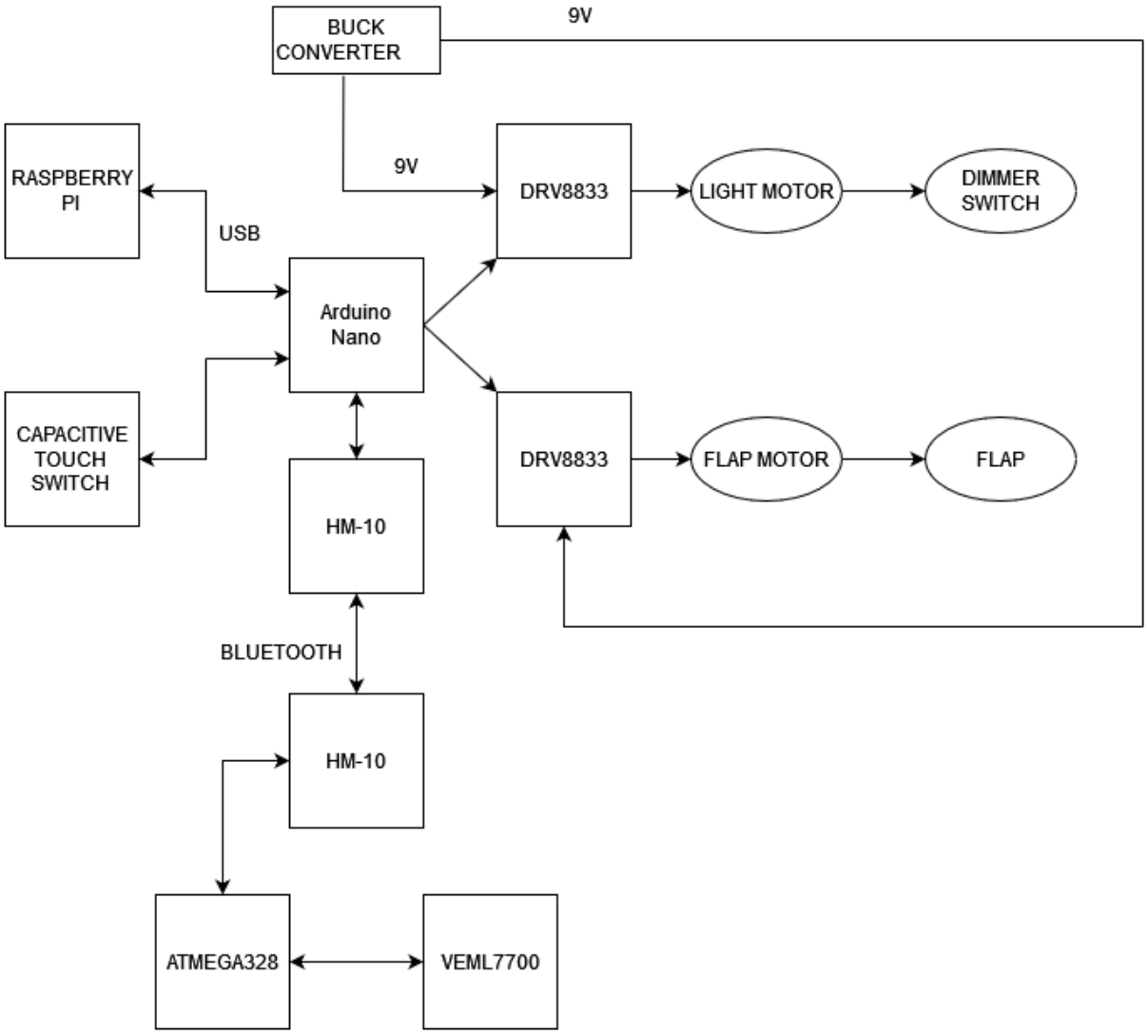


Fig. 16. Hardware Block Diagram

2. *Hardware Schematics*

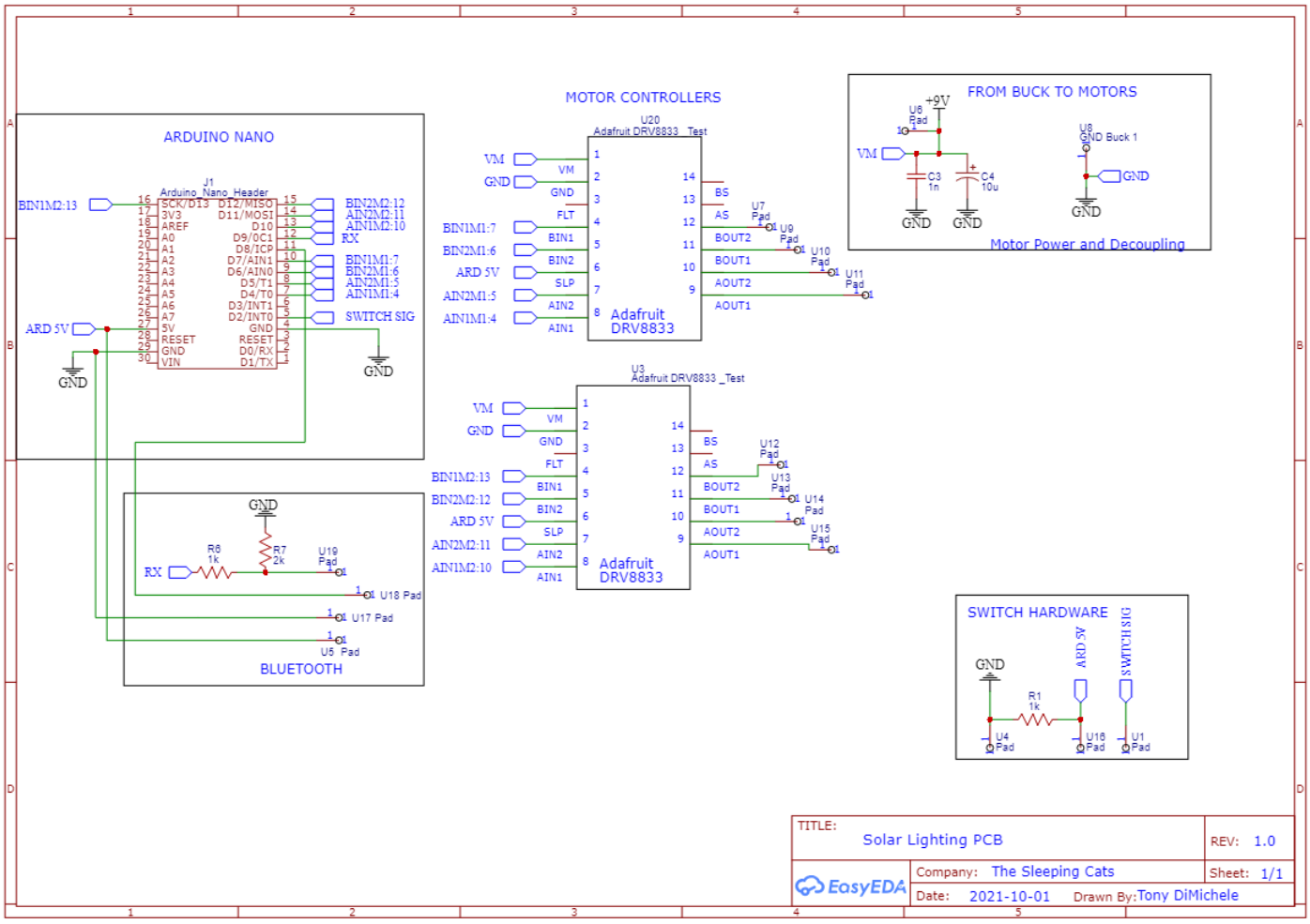


Fig. 17. Solar Lighting PCB Schematic

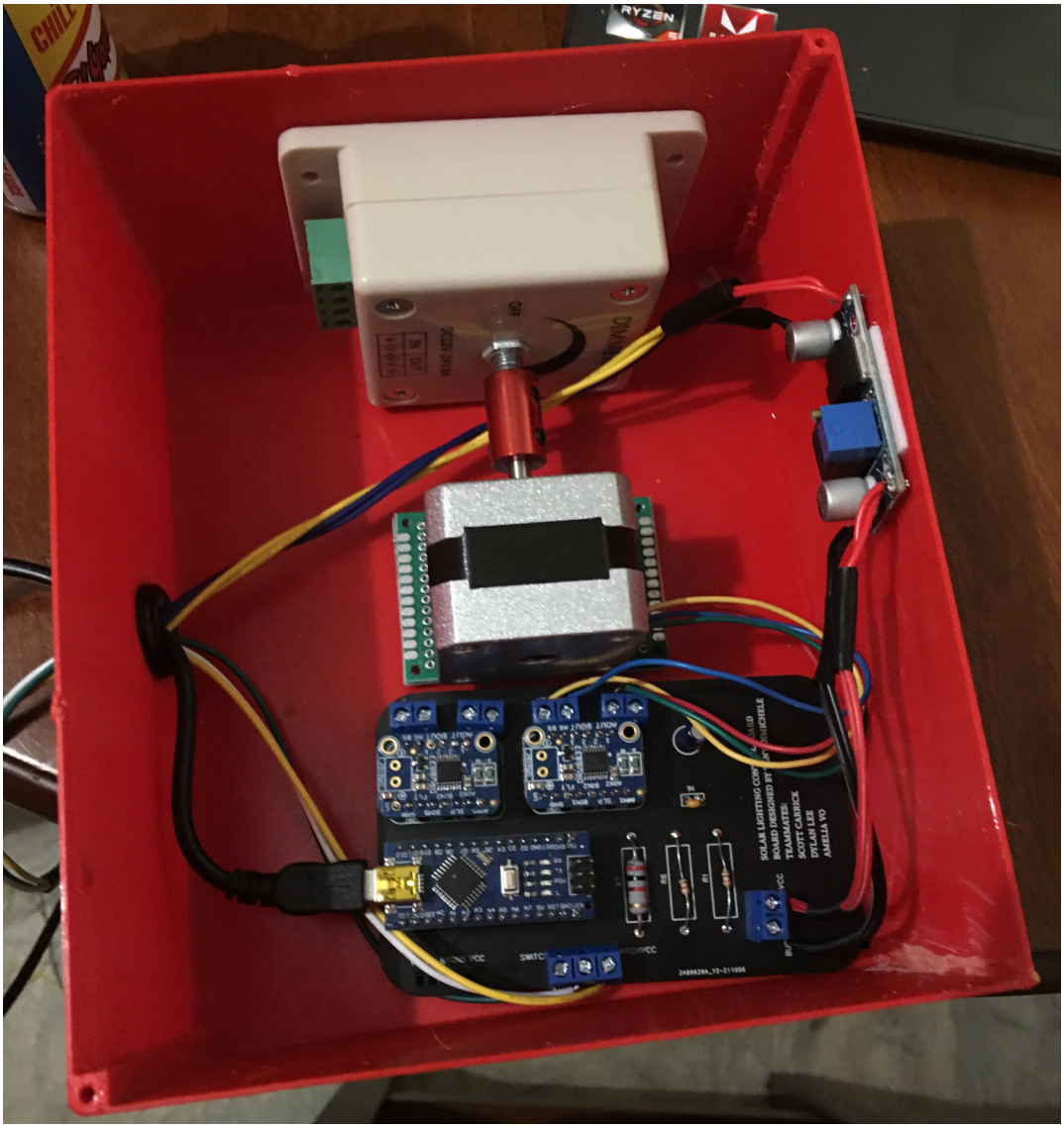


Fig 18. Control Box Hardware

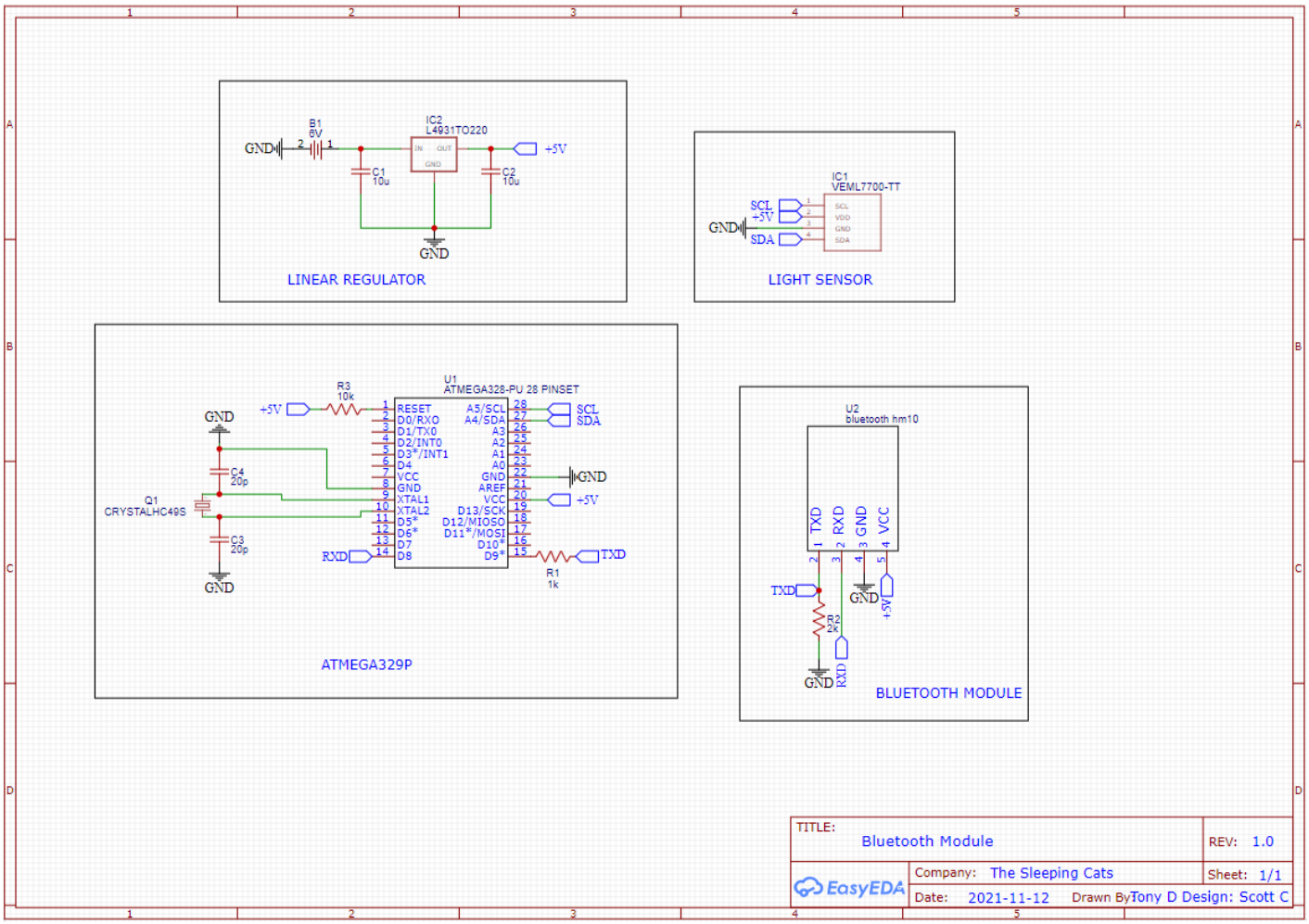


Fig. 19. Bluetooth Module Schematic

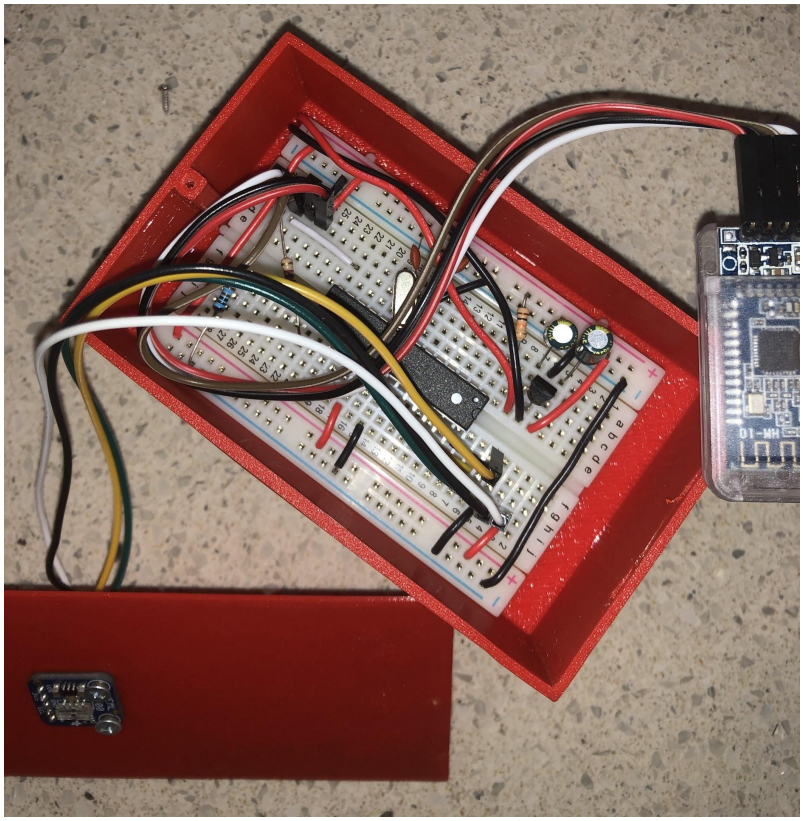


Fig. 20. Bluetooth Module Circuit

B. SOFTWARE

1. Block Diagram

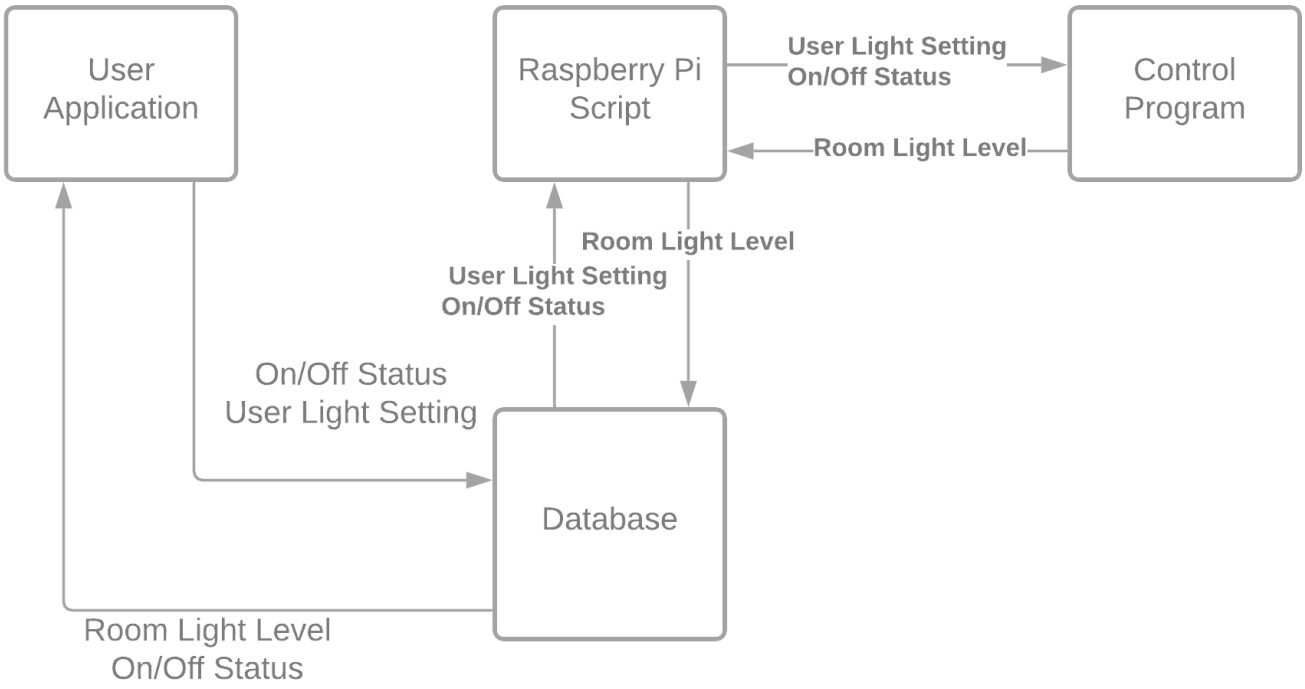


Fig. 21. Software Block Diagram

2. Raspberry Pi Script Block Diagram

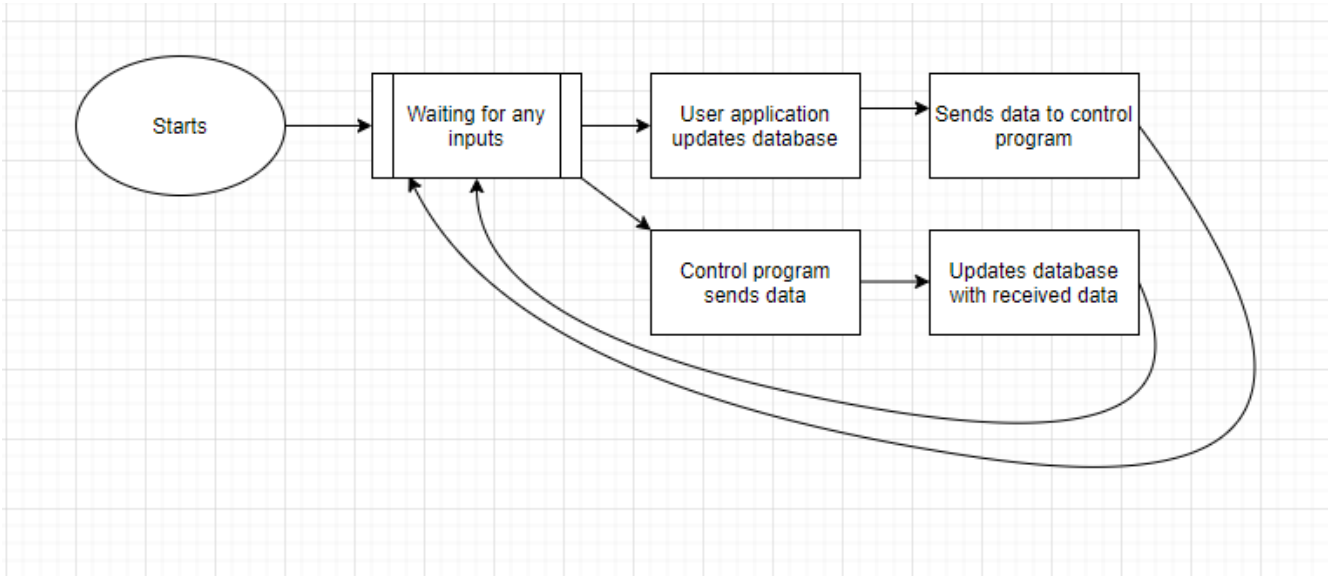


Fig. 22. Raspberry Pi Script Block Diagram

3. User Application Block Diagram

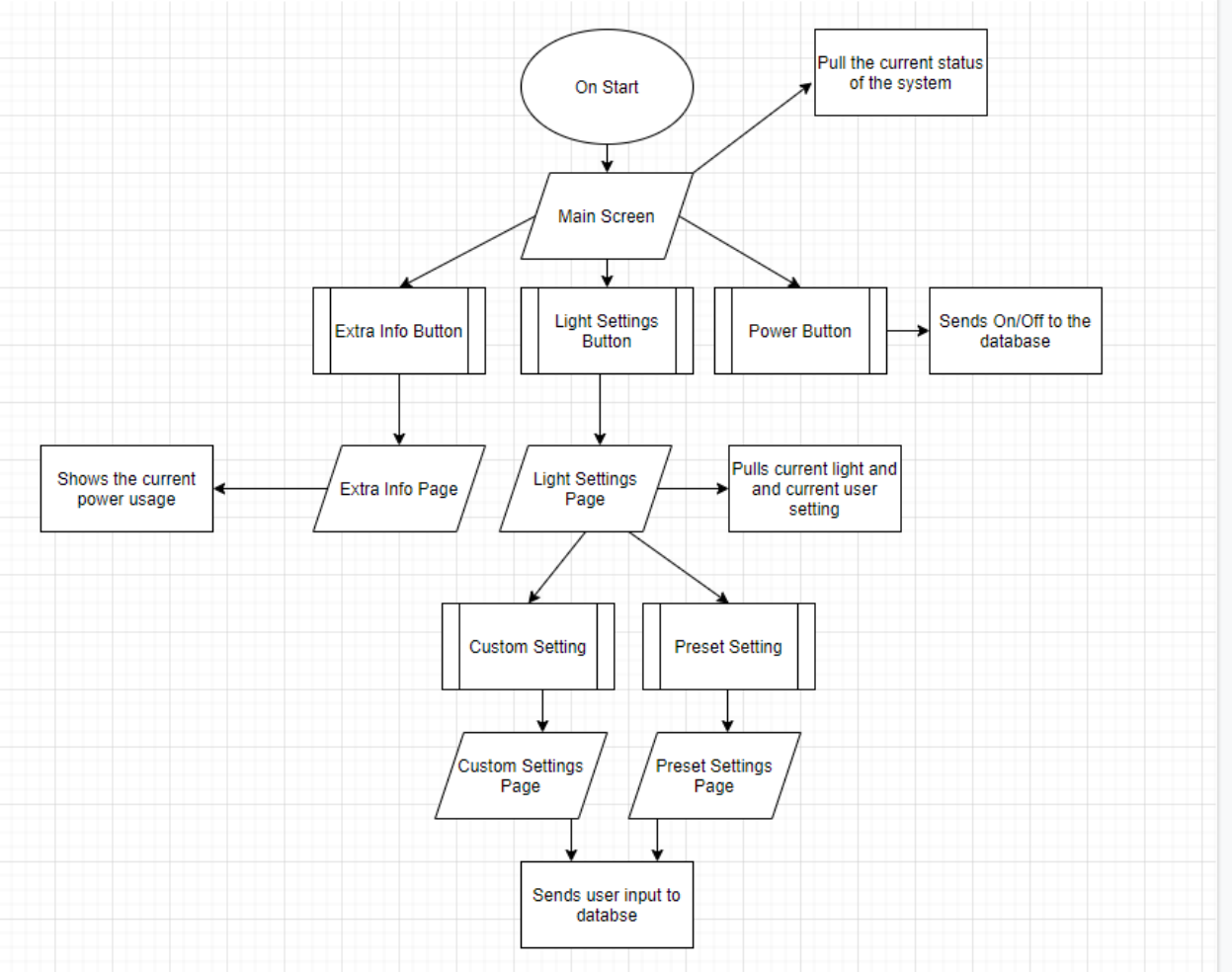


Fig. 23. User Application Block Diagram

4. Code Spring 2021

I. User Application Main

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.SeekBar;
import android.widget.TextView;

import com.chaquo.python.PyObject;
import com.chaquo.python.Python;
import com.chaquo.python.android.AndroidPlatform;
import com.chaquo.python.android.PyApplication;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.FileOutputStream;

import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.jsoup.Connection.Response;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.nio.charset.StandardCharsets;
import java.nio.file.Path;


public class MainActivity extends AppCompatActivity {
```

```
    int lightLevel = 0;
    int seekVal;
    EditText mEdit;
    TextView progText;
    SeekBar sBar;
    ProgressBar pBar;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sBar = (SeekBar)findViewById(R.id.seekBar);
        progText = (TextView)findViewById(R.id.seekProg);
        pBar = (ProgressBar)findViewById(R.id.progressBar2);
        initPython();

        String clientID = MqttClient.generateClientId();
        MqttAndroidClient client = new
MqttAndroidClient(getApplicationContext(), "broker.emqx.io", clientID);


        try {
            //MQTT Connect
            client.connect().setActionCallback(new IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken asyncActionToken) {
                    System.out.println("Connecting Success");
                }

                @Override
                public void onFailure(IMqttToken asyncActionToken, Throwable
exception) {
                    System.out.println("Connecting Failed");
                }
            });

            client.subscribe("Test_DylanGetLL",1);


            client.disconnect().setActionCallback(new IMqttActionListener()
{
                @Override
```

```
                public void onSuccess(IMqttToken asyncActionToken) {
                    System.out.println("Disconnecting Success");
                }

                @Override
                public void onFailure(IMqttToken asyncActionToken, Throwable
exception) {
                    System.out.println("Disconnecting Failed");
                }
            });
        } catch(MqttException e){

        }



        //Shows what setting is currently on the seek bar.
        sBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                pBar.setProgress(progress);
                progText.setText(""+progress+"%");
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });
    }

    private void initPython() {
        if(!Python.isStarted()){
            Python.start(new AndroidPlatform(this));
            System.out.println("Python Started");
        }
```

```
    }
    private void getPython(int x){
        Python py = Python.getInstance();
        PyObject number = py.getModule("srproj");
        number.callAttr("test", x);
        return;
    }
    //Upon clicking send Parameter button.
    public void sendParameter(View view) throws MqttException {
        //These are integers
        lightLevel = getParameter();
        seekVal = sBar.getProgress();
        System.out.printf("Seekbar currently at: %d\n", seekVal);
        String valx = Integer.toString(seekVal);

        //MQTT W/ Python Script Test
        getPython(seekVal);

        //MQTT Java Test

        //MQTT Init
        String clientID = MqttClient.generateClientId();
        MqttAndroidClient client = new
MqttAndroidClient(getApplicationContext(), "broker.emqx.io", clientID);
        MqttMessage msg = new MqttMessage(valx.getBytes());


        try {
            //MQTT Connect
            client.connect().setActionCallback(new IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken asyncActionToken) {
                    System.out.println("Connecting Success");
                }

                @Override
                public void onFailure(IMqttToken asyncActionToken, Throwable
exception) {
                    System.out.println("Connecting Failed");
                }
            });

            client.publish("TestDylan", msg);
```

```
            client.disconnect().setActionCallback(new IMqttActionListener()
{
                @Override
                public void onSuccess(IMqttToken asyncActionToken) {
                    System.out.println("Disconnecting Success");
                }

                @Override
                public void onFailure(IMqttToken asyncActionToken, Throwable
exception) {
                    System.out.println("Disconnecting Failed");
                }
            });
        } catch(MqttException e){

        }

        //This is all the jsoup stuff.
    }

    public int getParameter(){
        int val = 0;
        double val2 = 0;
        mEdit = (EditText)findViewById(R.id.editLL);
        mEdit.getText().toString();
        try {
            val = Integer.parseInt(mEdit.getText().toString());
        }
        catch(NumberFormatException e){
            if(TextUtils.isEmpty(mEdit.getText().toString())){
                val = 0;
            } else{
                val2 = Double.parseDouble(mEdit.getText().toString());
                val = (int)val2;
            }
        }
        if(val < 0 || val > 100) {
            System.out.println("Invalid Argument. Setting Light to 0");
            val = 0;
        }
        return val;
    }
```

II. Raspberry Pi Script

```
import paho.mqtt.client as mqtt
import serial
import time

#serial1 = serial.Serial('/dev/ttyACM1', 115200)

Lgt_Lvl = 10

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("TestDylan")


def on_message(client, userdata, msg):
    msg.payload = msg.payload.decode("utf-8")

    if int(msg.payload) <= 100:
        print("Correct Input.")
        print("Input: " +msg.payload)
         serial1.write(msg.payload.encode())
         time.sleep(1)
        for x in range(4):
            print("Printed")
            serial1.write(msg.payload.encode())
            time.sleep(1.4)

def send_lgtlvl():
    client.publish("TestDylan_GetLL", Lgt_Lvl);

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("broker.emqx.io", 1883, 60)

client.loop_start()
while True:
    send_lgtlvl()
    time.sleep(2)
```

III. Light Sensor Bluetooth

```
//  This code will serve as the controller
// arduino code. It will recieve the light
// sensor data from the Sensor arduino.

#include "Adafruit_VEML7700.h"
#include <AltSoftSerial.h>
AltSoftSerial BTserial;
// https://www.pjrc.com/teensy/td_libs_AltSoftSerial.html
Adafruit_VEML7700 veml = Adafruit_VEML7700();

String a = " ";
char c = ' ';
boolean NL = true;
char reply[30];
int pos;

void setup()
{
   Serial.begin(9600);
   Serial.print("Sketch:   "); Serial.println(__FILE__);
   Serial.print("Uploaded: "); Serial.println(__DATE__);
   Serial.println(" ");

   // Starting bluetooth serial
   BTserial.begin(9600);
   Serial.println("BTserial started at 9600");

   veml.begin();
   veml.setGain(VEML7700_GAIN_1);
   veml.setIntegrationTime(VEML7700_IT_800MS);

}

void loop()
{
   // Read light sensor data and send to write to bluetooth

      pos = veml.readLux();
      Serial.println(pos);
      BTserial.print(pos);
      delay(2000);
```

```
}
```

IV. Control Program

```
//

#include <Stepper.h>
#include <AltSoftSerial.h>
#include <avr/sleep.h> //this AVR library contains the methods that controls the sleep modes

#define interruptPin 2 //Pin we are going to use to wake up the Arduino

  AltSoftSerial BTserial;
  Stepper stepper1(200, 4, 5, 6, 7);
  Stepper stepper2(200, 10, 11, 12, 13);
  int step_parameter = 0;
  int step_position = 0;
  int actuation = 0;
  int light_data_int;
  int light_set_point = 250;
  int steps = 10 ;
  String light_data = " ";
  bool sleep_parameter = 0;
  String inByte;
  bool program_state = 0;
  int flap = 50;
void setup()
{
  Serial.begin(115200);
  Serial.println("Stepper test!");
  stepper1.setSpeed(30); // set the speed of the motor (RPM)
  stepper2.setSpeed(10);
  stepper2.step(flap);
  BTserial.begin(9600);
  Serial.println("BTserial started at 9600");
  delay(2000);

  // Interupt Pin Set-up
  pinMode(interruptPin,INPUT_PULLUP);//Set pin d2 to input using the buildin pullup resistor
}

void loop()
```

```
  {
   if(program_state == 1){

      // Attach Switch Interupt
        attachInterrupt(digitalPinToInterrupt(interruptPin), changeState, RISING);

      //Recieve Data From App
        inByte = (Serial.readString());
        Serial.println("inByte "); Serial.print(inByte);
        if(inByte != NULL) {
        light_set_point = (inByte.toInt());
        Serial.print("light_set_point "); Serial.println(light_set_point);
        delay(50);
        }

      // This will read the serial data coming from the light sensor and convert it into an integer
        light_data = BTserial.readString();
        //Serial.println(light_data);  //debugging, comment out when working
        light_data_int = light_data.toInt();
        Serial.println(light_data_int);  //debugging, comment out when working

      // Clockwise Motor Actuation
        if((light_data_int < light_set_point) && (light_set_point-light_data_int > 10) && (step_position <
160)){
          stepper1.step(steps);
          step_position = step_position + 10;
          Serial.print("Actuating up!! Step Position = ");
          Serial.println(step_position);
        }

      // Counterclockwise Actuation
        else if(light_data_int > light_set_point  && (light_set_point-light_data_int < -10) &&
(step_position > 0)){
          stepper1.step(-steps);
          step_position = step_position - 10;
          Serial.print("Actuating down!! Step Position = ");
          Serial.println(step_position);
        }

      // DO NOT ACUTATE MOTOR
        else if(light_set_point-light_data_int >= 10 || light_set_point-light_data_int <= -10){
          Serial.println("Not Actuating!!");  // debugging, comment out when working
        }
```

```
        // reset the serial port
          while(Serial.read() >= 0);
          Serial.read();

        // reset Bluetooth serial
          while(BTserial.read() >= 0);
          BTserial.read();
      }

      if(program_state == 0){
        stepper2.step(-flap);
        stepper1.step(-step_position);
        step_position = 0;
        sleep_enable();//Enabling sleep mode
        attachInterrupt(digitalPinToInterrupt(interruptPin), changeState, RISING);
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);
        sleep_cpu();
        Serial.println("just woke up!");
        stepper2.step(flap);
      }
    }

    void changeState(){

      Serial.println("Interrrupt Fired");

//      if(program_state == 0){
//        //Open flap
//        Serial.println("Opening Flap");
//      }
//
//      else {
//        //Close Flap
//        Serial.println("CLosing Flap");
//      }

      program_state = !program_state;
      Serial.println(program_state);
      sleep_disable();
      detachInterrupt(0);
    }
```

5. Coding for Fall 2021

I. User Application Main

```
public class MainActivity extends AppCompatActivity {
    private Button lButton;
    private Button ooButton;
    private Button exButton;
    private TextView statText;
    private String value;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final RelativeLayout relativeLayout;
        relativeLayout = findViewById(R.id.bgr);

        //Button to open up light settings page
        lButton = (Button) findViewById(R.id.LightButton);
        //Button for on/off
        ooButton = (Button) findViewById(R.id.PowerButton);
        //Button to open up extra info
        exButton = (Button) findViewById(R.id.ExtraButton);

        lButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openLightSettings();
            }
        });

        exButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openExtraSettings();
            }
        });

        DatabaseReference ref;
        ref = FirebaseDatabase.getInstance().getReference().child("values");
```

```
        //Write on/off to database
        ooButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(value.equals("on")) {
                    ref.child("status").setValue("off");
                } else if(value.equals("off")){
                    ref.child("status").setValue("on");
                }
            }
        });

        // Attach a listener to read the data at our posts reference
        ref.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                value = dataSnapshot.child("status").getValue().toString();
                System.out.println(value);
                if(value.equals("on")){
                    relativeLayout.setBackgroundResource(R.color.on_color);
                } else if(value.equals("off")){
                    relativeLayout.setBackgroundResource(R.color.off_color);
                }
                updateText(value);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                System.out.println("The read failed: " +
databaseError.getCode());
            }
        });

    }

    public void openLightSettings() {
        Intent intent = new Intent(this, LightSettings.class);
        startActivity(intent);
    }

    public void openExtraSettings() {
        Intent intent = new Intent(this, ExtraInfo.class);
        startActivity(intent);
```

```
    }
    public void updateText(String status){
        statText = (TextView) findViewById(R.id.StatusText);
        statText.setText("Current Setting: "+ status);

    }



}
```

II. User Application Light Settings

```
public class LightSettings extends AppCompatActivity {
    private Button cusButton;
    private Button preButton;
    private TextView lText;
    private TextView sText;
    private TextView step;
    private boolean cusPress = false;
    private boolean prePress = false;
    private boolean test = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_light_settings);
        cusButton = (Button) findViewById(R.id.Custom);
        preButton = (Button) findViewById(R.id.Presets);

        //Get Database
        DatabaseReference ref;
        ref = FirebaseDatabase.getInstance().getReference().child("values");
        Toast.makeText(LightSettings.this, "Firebase Connection
Success",Toast.LENGTH_LONG).show();
        // Read from the database

        // Attach a listener to read the data at our posts reference
        ref.addValueEventListener(new ValueEventListener() {
```

```
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                //Updates the user light setting when changed
                String value =
dataSnapshot.child("light_level").getValue().toString();
                updateLText(value);

                //Updates the shed light when changed
                value = dataSnapshot.child("shed_light").getValue().toString();
                updateSText(value);

                //Updates the step when changed
                value = dataSnapshot.child("step").getValue().toString();
                updateStep(value);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                System.out.println("The read failed: " +
databaseError.getCode());
            }
        });

        cusButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View w) {
                openCustomSettings();

            }
        });

        preButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v){
                openPresetSettings();
            }
        });
        //runProg();
    }

    public void openCustomSettings() {
        Intent intent = new Intent(this, customa.class);
        startActivity(intent);
```

```
    }

    public void openPresetSettings(){
        Intent intent = new Intent(this, preseta.class);
        startActivity(intent);
    }
    public void updateLText(String Light){
        lText = (TextView) findViewById(R.id.LightSettings);
        lText.setText("Current User Light Setting: "+Light);

    }
    public void updateSText(String Light){
        sText = (TextView) findViewById(R.id.ShedLight);
        sText.setText("Current Shed Light Level: "+Light);

    }
    public void updateStep(String StepSet){
        step =  (TextView) findViewById(R.id.Step);
        step.setText("For Step Debugging Purposes; Step: "+ StepSet);
    }

}
```

III. User Application Custom Setting

```
public class customa extends AppCompatActivity {
    private EditText custom_set;
    private Button Send;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customa);
        custom_set = (EditText) findViewById(R.id.CustomLevel);
        Send = (Button) findViewById(R.id.SendButton);
        //Connect to database
        DatabaseReference ref;
        ref = FirebaseDatabase.getInstance().getReference().child("values");

        //Send Value to database
        Send.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View w) {

ref.child("light_level").setValue(custom_set.getText().toString());
            }
        });
    }
}
```

IV. User Application Extra Information

```
public class ExtraInfo extends AppCompatActivity {
    private String info = "Current Power Usage:";
    private TextView pwrText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_extra_info);

        DatabaseReference ref;
        ref = FirebaseDatabase.getInstance().getReference().child("values");

        ref.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                //Updates the power usage when step changes
                String value =
dataSnapshot.child("power_usage").getValue().toString();
                updatePwr(value);

            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                System.out.println("The read failed: " +
databaseError.getCode());
            }
        });
    }
```

```
    void updatePwr(String s){
        pwrText = (TextView) findViewById(R.id.pwrUsage);
        pwrText.setText("Current Power Usage:"+s+"W");
    }

}
```

V. Raspberry Pi Script

```
import pyrebase
import serial
import time

serial1 = serial.Serial('/dev/ttyUSB0', 115200)


config = {
  "apiKey": "AIzaSyDnOtrNEDTf7j1bZiy8oYv6M9J-hlSsTds",
  "authDomain": "sr-proj-light.firebaseapp.com",
  "databaseURL": "https://sr-proj-light-default-rtdb.firebaseio.com/",
  "storageBucket": "sr-proj-light.appspot.com",
  "serviceAccount": "/home/pi/Desktop/sr_proj/sr-proj-light-firebase-adminsdk-a92tf-8d53e32be1.json"
}

#Connect to the backend
firebase = pyrebase.initialize_app(config)
db = firebase.database()
auth = firebase.auth()

#Live update of the changed values
def stream_handler(message):
    light = db.child("values").child("light_level").get().val()
    status = db.child("values").child("status").get().val()
    #full = status + ";" + light
    full = message["data"]
    serial1.write(full.encode())
    print(full)


#Allows for live updates for changes in values
```

```
light_stream = db.child("values").child("light_level").stream(stream_handler, stream_id = "light")
status_stream = db.child("values").child("status").stream(stream_handler, stream_id = "status")

#This is how I can update values
#db.child("values").update({"light_level": "20"})

while True:
    temp = serial1.readline().decode("utf-8")
    split_hold = temp.split(":")
    temp_hold = []
    word_hold = ""
    upload_word = ""
    used_pwr = 0
    x = 0
    max_pwr = 21.195
    total_pwr = 0
    for element in split_hold:
        temp_hold.append(element.strip())
        word_hold = temp_hold[0]
        if(word_hold == "Light Data" and x == 1):
            upload_word = temp_hold[1]
            db.child("values").update({"shed_light": upload_word})
        if(word_hold == "Step Position" and x == 1):
            upload_word = temp_hold[1]
            db.child("values").update({"step": upload_word})
            if(upload_word == "30"):
                used_pwr = 6.75
                total_pwr = used_pwr - max_pwr
                db.child("values").update({"power_usage": total_pwr})
            if(upload_word == "40"):
                used_pwr = 8.1
                total_pwr = used_pwr - max_pwr
                db.child("values").update({"power_usage": total_pwr})
            if(upload_word == "50"):
                used_pwr = 9.18
                total_pwr = used_pwr - max_pwr
                db.child("values").update({"power_usage": total_pwr})
            if(upload_word == "60"):
                used_pwr = 10.395
                total_pwr = used_pwr - max_pwr
                db.child("values").update({"power_usage": total_pwr})
            if(upload_word == "70"):
                used_pwr = 11.475
```

```
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "80"):
            used_pwr = 12.555
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "90"):
            used_pwr = 13.635
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "100"):
            used_pwr = 14.85
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "110"):
            used_pwr = 16.335
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "120"):
            used_pwr = 17.55
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "130"):
            used_pwr = 19.305
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "140"):
            used_pwr = 20.925
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "150"):
            used_pwr = 23.49
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
        if(upload_word == "160"):
            used_pwr = 23.895
            total_pwr = used_pwr - max_pwr
            db.child("values").update({"power_usage": total_pwr})
    x = x+1
x = 0
```

VI. Control Program

```
#include <Stepper.h>
#include <AltSoftSerial.h>
#include <avr/sleep.h> //this AVR library contains the methods that controls the sleep modes

#define interruptPin 2 //Pin we are going to use to wake up the Arduino

  AltSoftSerial BTserial;
  Stepper stepper1(200, 4, 5, 6, 7);
  Stepper stepper2(200, 10, 11, 12, 13);
  int step_parameter = 0;
  int step_position = 0;
  int actuation = 0;
  int light_data_int;
  int light_set_point= 0;
  int steps = 10 ;
  String light_data = " ";
  bool sleep_parameter = 0;
  bool doOnce = 0;
  String inByte;
  bool program_state = 0;
  int flap = -50;
  char array_in[8];
  const char delim[2] = ";";
  char *ptr = NULL;

void setup()
{
  Serial.begin(115200);
  Serial.println("Stepper test!");
  stepper1.setSpeed(30); // set the speed of the motor (RPM)
  stepper2.setSpeed(10);
  stepper2.step(flap);
    digitalWrite(10,LOW);
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);
  BTserial.begin(9600);
  Serial.println("BTserial started at 9600");
  delay(2000);
  Serial.flush();
```

```
    // Interupt Pin Set-up
    pinMode(interruptPin,INPUT_PULLUP);//Set pin d2 to input using the buildin pullup resistor
   }

   void loop()
   {
    if(program_state == 0){
         if(doOnce == 1){
           stepper2.step(flap);
           doOnce = 0;
           Serial.println("MOTORSSSSSSS 0");
         }

        //Attach Switch Interupt
        attachInterrupt(digitalPinToInterrupt(interruptPin), changeState, RISING);

       //Recieve Data From App
        inByte =  Serial.readString();
          Serial.println("inByte "); //Serial.print(inByte);
          if(inByte != NULL) {
            if(inByte == "off"){
              Serial.println(inByte);
              changeState();
            }
            else if(inByte == "on"){
            } else {
              light_set_point = (inByte.toInt());
              Serial.println(inByte);
            }
          }

//
//          Serial.print("light_set_point "); Serial.println(light_set_point);
        // delay(50);


       // This will read the serial data coming from the light sensor and convert it into an integer
         light_data = BTserial.readString();
         light_data_int = light_data.toInt();
         Serial.print("Light Data: ");  //debugging, comment out when working
         Serial.println(light_data_int);

         //Serial.prinntln(light_data_int);  //debugging, comment out when working
```

```
        // Clockwise Motor Actuation
          if((light_data_int < light_set_point) && (light_set_point-light_data_int > 10) && (step_position <
160)){
            stepper1.step(steps);
            step_position = step_position + 10;
            delay(100);
//            Serial.print("Actuating up!! Step Position = ");
//            Serial.println(step_position);
          }

        // Counterclockwise Actuation
          else if(light_data_int > light_set_point  && (light_set_point-light_data_int < -10) &&
(step_position > 0)){
            stepper1.step(-steps);
            step_position = step_position - 10;
            delay(100);
//            Serial.print("Actuating down!! Step Position = ");
//            Serial.println(step_position);
          }

        // DO NOT ACUTATE MOTOR
          else if(light_set_point-light_data_int >= 10 || light_set_point-light_data_int <= -10){
            //Serial.println("Not Actuating!!");  // debugging, comment out when working
          }

          motorsOff();
          Serial.print("Step Position: ");  //debugging, comment out when working
          Serial.println(step_position);
//          // reset the serial port
//            while(Serial.read() >= 0);
//            Serial.read();

        // reset Bluetooth serial
          while(BTserial.read() >= 0);
          BTserial.read();
    }

    if(program_state == 1){
        if(doOnce == 0){
        stepper2.step(-flap);
        stepper1.step(1);
        stepper1.step(-step_position-1);
        step_position = 0;
```

```
      //turn motors off
      delay(500);
    motorsOff();
    Serial.println("MOTORSSSSSSS 1");
    doOnce = 1;
    }
    //sleep_enable();//Enabling sleep mode
    attachInterrupt(digitalPinToInterrupt(interruptPin), changeState, RISING);
    inByte =  Serial.readString();
       Serial.println("inByte "); //Serial.print(inByte);
       if(inByte != NULL) {
         if(inByte == "on"){
           Serial.println(inByte);
           changeState();
         }
       }
    //set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    //sleep_cpu();
    //stepper2.step(flap);
  }
}

void changeState(){
  Serial.println("Interrrupt Fired");
  program_state = !program_state;
  //Serial.println(program_state);
  sleep_disable();
  detachInterrupt(0);
}

void motorsOff(){
      digitalWrite(10,LOW);
      digitalWrite(11,LOW);
      digitalWrite(12,LOW);
      digitalWrite(13,LOW);
      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
      digitalWrite(6,LOW);
      digitalWrite(7,LOW);
}
```

C. User Manual

I. Welcome

Thank you for choosing the Solar Lighting reactive lighting system. This device uses a solar tube in combination with a robust lumen detection system to automatically control a lighting set-point in a room of roughly 200-sq ft. The control box and ambient light sensor are ready to go "out-of-the-box", and a professional technician will be deployed to install your solar tube, solar panel, light switch, and flap covering at no additional cost. Soon you'll have the benefits of cheaper electricity costs and free, natural lighting in your home or office!

Note: An android phone is required to control the set-point parameters and a download link will be sent to the email you registered with us at purchase.

II. Safe Handling Instructions

The Solar Lighting System is designed with the user in mind, but please note the handling instructions below to avoid damage to the system or injury to yourself

- Always plug in the wires to the system in the following order: Battery then Vin/Vout of the solar panel and then Load (including USB connections) and unplug the wires in the reverse order Load then Vin/Vout for solar panel and then the battery. Not following this scheme may damage your controller!
- Keep the battery out of direct sunlight
- Do not upload any new code to the control board
- Do not remove the caps of the lead-acid battery unless you are replacing it as this could lead to an accidental short if not handled properly, schedule for a technician if you are unsure about how to install a new battery.
- NEVER open the control box, this will void the warranty and could seriously affect the system's operation. The system requires precise knowledge of it's position and cannot account for you moving the motor shafts with your hands.

III. Included materials

The kit provided to you includes the following items:

1 solar panel with controller (installed by technician)

1 solar tube with refractive dome and diffuser plate (installed by a technician)

1 touch ON/OFF switch (installed by technician)

1 flap cover (installed by technician)

1 12V 20AH Lead-Acid Battery

1 control system box

1 light sensor box

1 Raspberry Pi 4 with USB cable

2 rows of 10-ft of Bright White LEDS

25 feet each of 18 gauge red, white, green, and yellow wire (installed by technician)

4 AA Batteries

IV. Setup and operation procedures

A. Setup

1. Find a suitable spot directly under both the solar tube and LEDs where the ambient light sensor will work most effectively. This is at a height generally between 5-7 feet off the floor for a 10-12 foot ceiling. Your technician will help you choose a suitable location. You must be able to access this location in the future to reinstall fresh batteries as needed. Turn off the light sensor when you leave the room via the switch on the outside of the box to maintain battery life.
2. Connect the lead-acid battery, solar panel, and lights to the solar controller (in that order). Disconnect in the opposite order. Connect the load + and load - wires of the control box to the same terminals the LEDs are connected to. Connect the Raspberry Pi to the USB outlet of the controller, and the USB from

the control box to any of the Raspberry Pi's USB ports.

3. Use the download link sent to your email to install the Solar Lighting app on your android device and connect the system to your home wifi.

B. Operation

1. Turn the system ON via the app button or the touch switch installed in your room.

2. To change the lighting set point of the room, enter a number into the set-point field within the app and adjust until a desired brightness of the room is found. After this the system will remember your desired brightness setting, but if you ever wish to change it you can do so at any time.

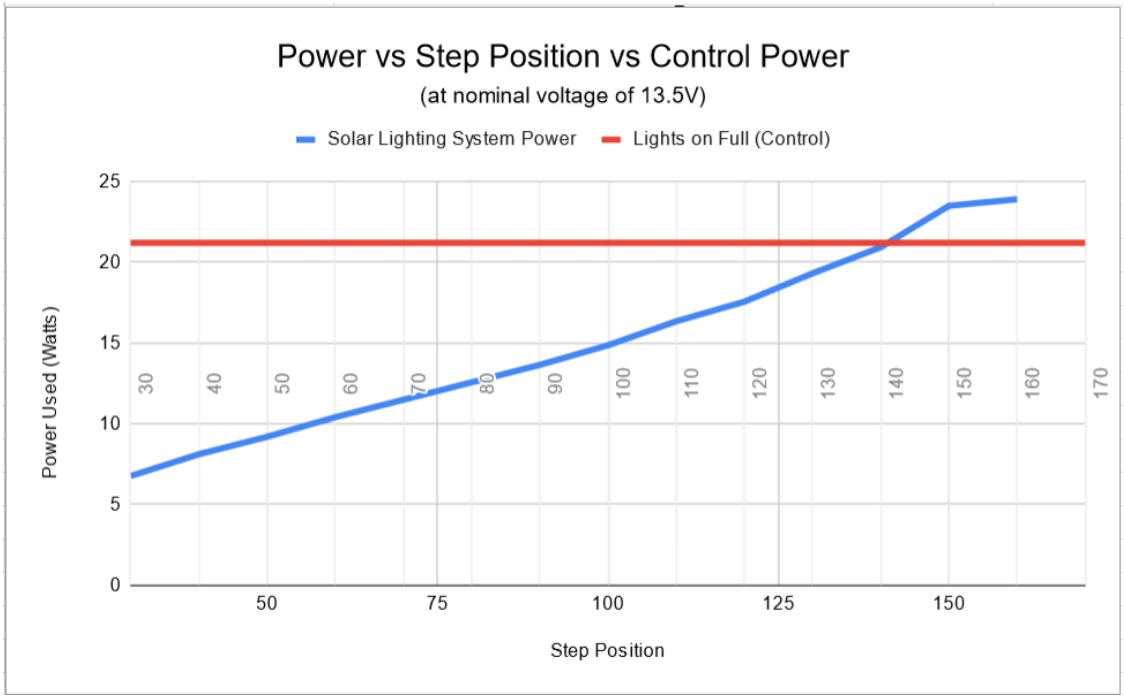
3. You may use the app at any time to track whether the system is ON or OFF and to see the current light level, light set-point, and power usage of the lighting system.

4. To turn the system OFF use either the app button or the touch switch.

D. SOLAR LIGHTING POWER SAVINGS DATA

The table and graph below show the power used in our system vs a control. If the system is actuated at 140 steps or less then there is a power savings versus the light in the room being simply on or off with a regular switch.

Step Position	A w/ ammeter Solar Lighting System	Solar Lighting System Power	Lights on Full (Control)	A with Lights all the way on, no Solar Lighting System
30	0.5	6.75	21.195	1.57
40	0.6	8.1	21.195	1.57
50	0.68	9.18	21.195	1.57
60	0.77	10.395	21.195	1.57
70	0.85	11.475	21.195	1.57
80	0.93	12.555	21.195	1.57
90	1.01	13.635	21.195	1.57
100	1.1	14.85	21.195	1.57
110	1.21	16.335	21.195	1.57
120	1.3	17.55	21.195	1.57
130	1.43	19.305	21.195	1.57
140	1.55	20.925	21.195	1.57
150	1.74	23.49	21.195	1.57
160	1.77	23.895	21.195	1.57



E. RESUMES

Tony DiMichele

❖ Sacramento, CA

WORK EXPERIENCE

Liquid Propulsion Group **Dec 2020 – Present Year**
Controls Hardware Engineer
Sacramento, CA

- Responsible for designing and testing of pressure transducer, thermocouple, and force transducer circuits
- Design, build, and test controls hardware in an interdisciplinary team of engineering students
- Install and troubleshoot computer and microcontroller hardware issues
- Assist with documentation of specifications and research

CSUS **Feb 2020 – June 2020**
Digital Logic Design Lab Assistant
Sacramento, CA

- Assist students, provide clarification, and give supportive instruction with homework and labs
- Administer lab sessions and grade demos
- Proctor lessons on Verilog and breadboard circuitry
- Monitor students to ensure lab safety

EDUCATION

CSUS **Dec, 2021**
BS. Electrical and Electronic Engineering *Sacramento, CA*

- Dean's Highest Honors 3.85/4.0 Major GPA

SKILLS & INTERESTS

- **Skills:** PSPICE/LTspice, MATLAB, Microcontroller Programming, ADS 2016, C Circuit Analysis, Multisim, Python, Teamwork, Interpersonal Communication
- **Interests:** Rockets, Movies, Hiking, Dogs, Astronomy, Craft Beer, Cooking, Music

Scott Carrick

Sacramento, CA

WORK EXPERIENCE

University Enterprises, Inc. **July 2018 – Present**
Student Assistant Engineer for Caltrans *Sacramento, CA*

- As Assist lead engineers on testing programs such as the Traffic System Control Program, the Field Master Program, and the Universal Ramp Metering Program.
- Manage database records for statewide radio communication equipment; conduct training for database usage

Tutoring Rocks **Oct. 2017 – July 2018**
Academic Mentor *El Dorado Hills, CA*

- Tutored and mentored students in chemistry, biology, and math

EDUCATION

California State University, Sacramento **December, 2021**
BS, Electrical and Electronic Engineering *Sacramento, CA*

- Completing upper division coursework with a focus in communication. GPA: 4.0/4.0.
- Member of the Liquid Propulsion Group, a student led group designing, manufacturing, and firing a liquid bi-propellant, pressure-fed rocket engine.

PROJECTS

- **Solar Lighting** *Reactive lighting system (in progress)*: Developing a system that integrates a solar tube with reactive lighting, allowing a room to be illuminated by natural light. The unit will maintain constant luminosity using sensory data and microcontrollers.
- **BrightBlinds** *Automated window blinds* : Developing a system that integrates a solar tube with reactive lighting, allowing a room to be illuminated by natural light. The unit will maintain
constant luminosity using sensory data and microcontrollers.

SKILLS

- Experience with AutoCAD and SolidWorks
- Experience with project planning and assembly
- Ability to collaborate in a professional environment
- Proficient in database management software
- Experience in Java, C++, and Python

REFRENCES

- **Ismael Briseno**: Supervisor Telecommunication Engineer, Caltrans; 1120 N Street, Sacramento, CA 95814 - (916) 651-2023
- **Jay Schultz**: Transportation Engineer/Electrical, Caltrans 1820 Alhambra Blvd., Sacramento, CA 95816 (916) 227-4662
- **Sergio Aguilar Rudametkin**: Engineering Professor, CSU Sacramento, sergioaguilarru@csus.edu, (916) 278-6873

Dylan Lee

WORK EXPERIENCE

Pleasant Grove High School **Sept. 2016 – Feb. 2018**
After School Music Instructor *Elk Grove, CA*

- Taught High School students with another instructor on how to play certain instruments as well as instructing the rest of the group on proper etiquette and performance.
- I was responsible for the rhythm section of the group which for our group was the backbone of keeping the front and back together.

EDUCATION

California State University, Sacramento **Dec, 2021**
BS. Computer Engineering *Sacramento, CA*

SKILLS & INTERESTS

- **Skills:** PSPICE, Java, C, Linux, Assembly, Verilog, Programming Microcontrollers
- **Interests:** Games, Anime, Brewing Coffee, Alcohol, Music, Cats

Amelia Vo

❖ Sacramento, CA

PROJECTS

Weather Station
Collaborated with a small group to create a simple weather station with multiple uses. Used sensors and Raspberry Pi to create a weather station measuring temperature, pressure, UV Ray, and humidity. Coded in Python and C language to make the system run which later sent a text message and an email to the user.

Traffic Light Signal
Designed a 2-way intersection with perpendicular road which allowed traffic on East-West until a car arrived at the North-South road. I was using C++ coding language and Nucleo board and its GPIO pins.

LED Blinks in S.O.S Signal
Wired 3 LEDs and a button using a raspberry Pi and GPIO pins. Programmed in Python so that once the button is pressed, the LED blinked in S.O.S morse signal.

Navigate and Park a Robot
Programmed a robot following an instruction route with music playing when it was required, also reversed parking. Used hardware Parallax Board and software BASIC Stamp

Creating Web Server
Created a webserver on LINUX using Python with Flask Framework.

EDUCATION

California State University, Sacramento — **Graduation** December, 2021
Bachelor of Science, Electrical and Electronics Engineering — *Sacramento, CA*
Participant in MESA Engineering Program, Sac State | Aug 2019-Present — CSUS GPA: 3.626

SKILLS & INTERESTS

- **Skills**: MATLAB, LTSPICE, language Python, C++; microcontrollers; soldering
- **Interests:** traveling; meditating; cooking; drawing; sleeping.