Automatic Light Tailoring Apparatus Instructing Radiance (ALTAIR)

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Peter Duchene Morris

Fall, 2021

Technical Project Team Members

Peter Morris

Mason Notz

Steven Peng

Alexander Tomiak

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Harry Powell, Department of Electrical and Computer Engineering

noFUN / ALTAIR

Peter Morris, Mason Notz, Steven Peng, Alexander Tomiak

December 17, 2021

Capstone Design ECE 4440 / ECE4991

Signatures

Peter Morris

Mason Notz

Steven Peng

Alexander Tomiak

Statement of work:

Peter Morris

As the lone electrical engineer in the capstone group, my work primarily focused on developing the PCB boards. My first task was to determine which components were needed for the design. After choosing our project, I researched and picked out the light sensor. I reviewed the datasheets and application notes of the components needed for the project in order to design the Multisim files. I used the resources to create new components for the database which were specific to our team's project. After I completed the Multisim files, I exported the files in Ultiboard to layout the board. Like in Multisim, I needed to create new components by reviewing figures and measurements presented in the datasheets. From the Ultiboard files, I created Gerber files to be checked on FreeDFM, and then submitted the boards for manufacturing. When the boards were returned, I tested the power supplies on the board after the boards were populated.

Steven Peng

From my extended mechanical experience with Virginia Motorsports Education, a 501(c)(3) non-profit that educates members of all experience levels through experiential learning, I was tasked with managing all the mechanical designs and manufacturing. My first task was to design a way to actuate the blinds, so I first measured the torque required to rotate the inner shaft when the blinds were fully open, which came out to be ~0.69 oz-in. I then chose a SER0046 Micro Servo from DFRobot which output 1.5 kg-cm of torque, which resulted in a safety factor of ~30. To attach the servo to the blinds, I designed a custom bracket in Autodesk Fusion 360 and manufactured it at Lacy Hall with the OMAX 1515 Waterjet and the Pan and Box brake. Next, I designed an adapter in Autodesk Fusion 360 to convert the splined output of the servo to the shaft input of the blinds tilting mechanism, and then made it on a manual lathe at Lacy Hall with the assistance of Mason Notz. Once the servo integration with the blinds was completed, I populated both of our PCB's with components to save time and money in comparison to outsourcing it to a local store.

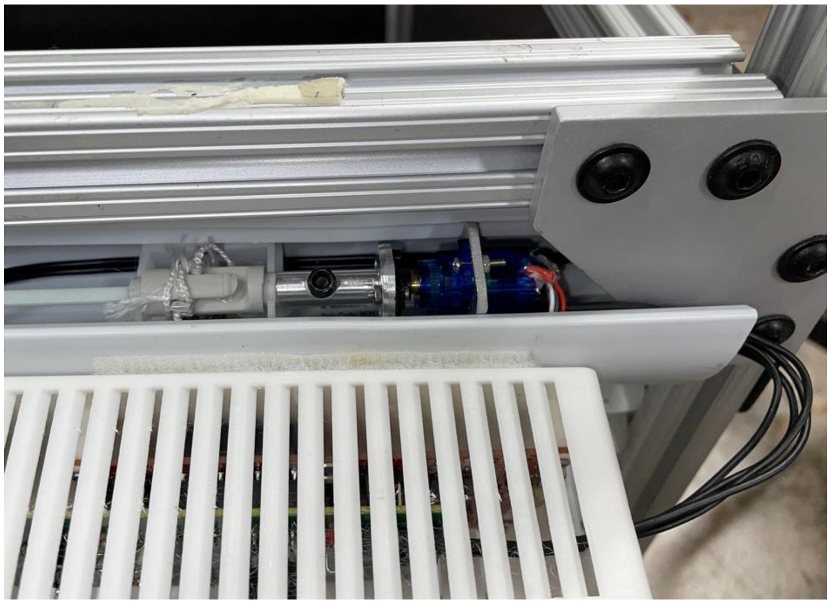


Figure 1: Servo Mounting and Adapter

Afterwards, housings for both PCBs needed to be designed and created, so I utilized Autodesk Fusion 360 once more. This time around, however, I used a Creality CR-10 Smart 3D printer to create the housings as they would contain exposed electrical components. By creating the housings out of PLA plastic instead of metal, it would mitigate the chance of either of the boards to short against the housing. Finally, to bring the entire project together I created a testing rig that would emulate a sealed room with a single window, along with an interior lamp for the smart light control algorithm and an exterior lamp that could emulate the sun as well. I also implemented the front-end user control web site hosted on the Raspberry Pi remote node, which could take in the user’s desired light level and process them through the control algorithm on the back end, which would then send the appropriate commands to adjust the lights and blinds to achieve the desired brightness.

Mason Notz

My work primarily focused on software and firmware. I designed and implemented the underlying control algorithm on the Raspberry Pi, as well as integrating the front end coded by Steven into a backend capable of controlling the servo and light system. I helped with the Raspberry Pi side of the communication to and from the CC3220. On the electrical side, I helped with specking out components, specifically related to power. Detailing power requirements for the board and picking appropriate devices to supply that power. On the firmware side, I wrote the PWM and GPIO drivers for the CC3220. On the mechanical side, I assisted Steven in creating the servo to shaft coupler at Lacy Hall with the lathe and mill. I also double checked some mechanical designs made by Steven.

Alex Tomiak

Because of my prior experience working with embedded systems at internships, and my continued interest in this field, my primary role was to design our embedded software/firmware architecture and manage communication between devices. In particular, I was responsible for developing the wireless communication between the CC3220 and the Raspberry Pi. Once we decided to go with Wi-Fi, I learned how to use the SimpleLink SDK for Wi-Fi firmware and software development on the CC3220. This involved figuring out what toolchain to use, understanding what different parts of the SDK did, and how it might apply to our project. I developed the TCP socket connection between the two devices, and developed the messaging schemes and behaviors that our project uses to pass data in a useful manner between the two devices. I also developed some tests and some wrappers for the ease of use of others who needed to use the wireless communication. Finally, I developed, tested, and integrated the driver and the exponential averaging for the VEML7700 light sensor on the CC3220, and I helped integrate Mason's PWM and GPIO CC3220 drivers to control the servo.

Table of Contents

Contents

Table of Figures

Table of Tables

Abstract

The project is an interior-setting, responsive window shade and light controller. The purpose of the project is to control the lighting level selected by the user at a desired location within a room. The light level is maintained by manipulating natural light coming through the window with a wall shade and interior lights. The window shade will be controlled by a motor connected to a Texas Instruments (TI) microcontroller. The light level at the desired location will be determined from a photodiode which is connected wirelessly to the motor/microcontroller. The interior lights will be connected as Internet of Things (IoT) devices to the motor/microcontroller which allows the user to specify his/her desired light level. This project aims to improve user comfort within a room and provide an efficient mechanism for controlling ambient light.

Background

Lighting is an important feature for a room. According to Zee and Cheung from Northwestern Medicine, "…exposure to light, especially during the day… is beneficial to your health via its effects on mood, alertness, and metabolism"[1]. Given the effects light has on each individual's life, it is important to provide an opportunity for those individuals to control the light in any area that they choose.

To ensure maximum individual comfort, the responsive window blind and light controller was developed. Photodetectors determine the current light level within a room for the project's microcontroller unit. In accordance with the user's preset light level, the microcontroller will control a motor to affect the amount of light coming through blinds on a window. If the light flux entering a window is not sufficient to fulfill the user's wishes, the microcontroller will interact with smart lights to provide the desired light level. The project is divided into a main hub which consists of the microcontroller, the motor, and a wireless transmitter/receiver, and a wireless photodetector which relays the light information for a desired position within a room to the microcontroller.

There are other attempts at making a responsive window shade. In 2020, a group of University of Virginia (UVA) students did a capstone project named Window Automated Natural Daylight Sensor, or WANDA for short[2]. The WANDA project took user input from a mobile application for a desired light level and transmitted that data to a microcontroller to control the window blinds. The WANDA system-based motor controls of the blinds on a light sensor as well as a proximity sensor. The light sensor tried to match the light level to the desired user settings, and the proximity sensor added a degree of security for the position of automatic blinds. Within industry, there are several companies that have tried to implement smart blinds. IKEA has electric blinds that are on a roller (as opposed to cellular blinds)[3]. The IKEA blinds are controlled by either a remote controller or a mobile app and can be programmed to follow a schedule for opening or closing. A product by Smarter Home features Bluetooth connectivity to an app and schedule setting[4]. A unique aspect of the Smarter Home is the shades are solar powered and can be voiced controlled. Blinds from Lutron and Graber have similar mobile application or remote-control accessibility, but are battery powered[4], [5].

Our project differentiates from current market options in two ways. The first distinction is the use of an interior light sensor. The interior light sensor provides feedback information for a desired location within the room. This would lead to the desired light level at a specific location like a desk or bed, rather than just being on a set schedule. Additionally, our project will be connected to smart lights. The goal of our project is to maintain a certain light level at the desired position through natural light and smart lights.

The scope of the project will draw from many courses required for Electrical and Computer Engineers (ECE) and Computer Scientists (CS). The ECE Fundamentals Sequences (ECE 2630/ECE 2660/ECE 3750) will be necessary for designing circuit boards which will provide power to the microcontroller, sensors, motor, and wireless receiver/transceiver components. Embedded Computing and Robotics (ECE 3501/ECE 3502) will be necessary for understanding/coding microcontroller functions as well as reading/understanding datasheets. Additionally, Real Time Embedded Systems (ECE 4501) was necessary for developing a program that guaranteed computations within hard deadlines. Cloud Computing (CS 4740) and Algorithms (CS 4102) will be necessary for writing efficient code on the microcontroller and connecting the wireless sensor to the microcontroller for data processing and environment manipulation.

Constraints

Design Constraints

The team consisted of a mix of three computer engineers and an electrical engineer. Due to the mix of the majors, the Capstone course imposed primary restraints: the groups must use a microcontroller and design a custom PCB board to incorporate digital and analog signals.

To complete our capstone project, the team decided to have the remote node and the and the window node communicate wirelessly. That limited the communication to Bluetooth, Wi-Fi, or RF. As none of the team members had experience with RF communication, it limited the choices down to Bluetooth and Wi-Fi. After research, the team decided to proceed with Wi-Fi communication. The capstone team that did the WANDA project used the TI CC-3200 microcontroller and had some issues. Our team chose the TI CC3220SF-LAUNCHXL Microcontroller; referred to as the CC3220 later[6]. The team used a Raspberry Pi 3A+for the remote node; referred to as the Pi later[7]. The CC3220 needed to communicate with the Pi and control the servo. The Pi needed to communicate with a light sensor, store the user interface, and communicate with the CC3220.

CPU Limitations

The CC3220 required a 3.3V supply to turn on. The microcontroller consisted of 40 pins in two sets of 2x10 on both sides of the board in a standard, TI microcontroller set up[8]. The CC3220 pin functionalities include General Purpose Input/Output (GPIO), Pulse Width Modulation (PWM), I^{2}C communication, SPI communication, and UART communication. The CC3220 also included a SimpleLink Wi-Fi on a chip. For the capstone project, the team needed to use the GPIO pins and PWM pins to control the servo connected to the blinds. I^2C communication was needed to communicate between the light sensor and the CC3220. The Wi-Fi chip was used to communicate with the Pi. The Pi required a 5V supply to turn on. The Pi consisted of 40 pins in two rows of 20. The Pi pin functionalities included GPIO, PWM, I^2C, and SPI. The I^2C pins communicated with a light sensor.

In order to test the capstone project, the team needed to design and construct a small-scale version of a room within the National Instruments (NI) Lab at the University of Virginia. The model allowed the team to test the different nodes of the project and their impact on the blinds.

Software Availability

In order to design the PCB board, NI's Multisim and Ultiboard were used. Both software packages were provided through the University of Virginia to electrical and computer engineering students. Multisim is a circuit design software[9]. Ultiboard is a PCB design software[10]. The PCB boards were designed in Multisim, exported to Ultiboard for layout, and then sent out for manufacturing.

Manufacturing Limitations

Due to the Covid-19 pandemic, there has been a shortage of electrical components[11], [12]. The supplies of components found on Digikey[13] and Mouser[14] were limited. The team ensured that all components ordered were not on marketplace sale or backorder.

Each of the PCB boards from the Ultiboard software had two layers: copper top and copper bottom. The boards also had silkscreen, solder mask top, and solder mask bottom layers. The Ultiboard Gerber files were exported and renamed to match the standards for a FreeDFM board check[15]. Hole sizes, silkscreen letter sizes, and board dimensions were all accurately reported for a successful report. The PCB board was received by the team about two weeks after submission.

Most of the system can be built with off-the-shelf components, with only the housing for the Printed Circuit Boards (PCB), the PCBs themselves, and the adapter between the servo and the blinds requiring custom fabrication. The system will be entirely controlled through a web interface hosted on the Raspberry Pi, which can be connected to through the web browser on a computer, cellular phone, etc. The web interface should be designed as simple as possible, with an option to fully increase or decrease the brightness of the room, and either buttons or a slider to gradually change the brightness in either direction as well.

The project components are currently available for mass production. All electronic components for this project are active products on Digikey. Mass producing custom PCB boards can be manufactured by a wide variety of manufacturers and in general is a solved problem. For example, DigPCB can bulk manufacture custom PCBs[16]. Mechanical components will be the source of some constraints. There is a custom connection from the servo to the blind actuating shaft. The piece could be redesigned to a plastic injection mold for mass production.

Economic and Cost Constraints

The major expenses for this project are the PCBs, the microcontrollers, and the materials required to manufacture custom components. The main economic benefit of this device is the potential saving on the energy bill, as described in the *Environmental impact* section, and the ease of use, where the user can spend more time working on productive tasks than having to manually adjust the blinds and lights to their liking. The benefit of this should be comparable to the cost. Similar products that only actuate the blinds can be purchased for over $500, so being able to create a product for less than that would make this technology, and the associated energy savings, more accessible to all.

Environmental Impact

There are many environmental considerations that need to be made during the manufacturing of the product, the usage of the product, and the disposal of the product. As we are creating a custom PCB, there are a lot of carcinogenic materials used during their production, such as glycol ethers, formaldehydes, and lead, which can cause long-term health issues to workers and wildlife alike[17]**.** As we are planning on creating a metal housing for the servo to reside in, the impact to the environment through the process of mining and processing iron to make steel is significant, as it requires a large amount of energy, which is made by generating a substantial amount of CO_2 which is then released into the environment[18]**.** Running the servo also takes energy, which would slowly but surely increase the user's carbon footprint as the motor will be powered through a conventional 120V outlet. The servo itself only requires 0.7 watts at full operating conditions[19]. As most of the device will be made from metal, it can be recycled easily. However, any plastic components created from 3D printing, for example, can introduce microplastics into the environment, which is known to be toxic to living organisms [20]. A benefit of our project will be the energy savings associated with blinds usage. The Department of Energy notes that most types of window treatments will result in energy savings[21]. Shades can help retain heat within homes during the winter and block glare in the summer.

Sustainability

The housing for the window and remote nodes was 3D printed using PLA filament. According to the safety specifications under the technical documentation, the PLA filament is biodegradable[22]. If the material is left within the environment, no toxicity nor bio accumulation was reported.

The model of a test room used aluminum from 80/20[23]. Aluminum is extracted from bauxite ore. As the aluminum ore is not buried deep in the ground, mining mainly occurs in open-pits[24]. Aluminum mining contributes to the biodiversity loss by clearing trees and grasses. During the mining of aluminum ore, there is the potential to uncover toxic or naturally radioactive elements which can pose a risk to the local environment and humans[25].

The PCBs are generally made of a fiberglass epoxy resin[26]. Fiberglass is not decomposable and not affect by "weather, saltwater, and most chemicals". The production of fiberglass can release air pollutants and is subject to EPA regulations. Finally, fiberglass has the potential to cause skin irritation is ingested or comes in contact with exposed skin.

Health and Safety

As this product contains moving parts, it can be dangerous if any of those components were exposed, especially to a child. To prevent any bodily harm, the entire drive unit will be fully enclosed, with some small slits for ventilation for the motor and the electronics. Since the unit resides indoors and will not be exposed to outside elements, it does not have to be fully sealed or weatherproofed and will follow the NEMA Type I rating for enclosures, as described in the *Standards* section.

If the device were to project during operation, there is a very slim chance that the user could be subjected to extremely low or extremely high sunlight exposure. Extremely low sunlight exposure would be harmful to the user. Sunlight causes the body to produce Vitamin D which is beneficial for bone health and immunity [27]. At the other extreme, the user could be subject to extreme sunlight. Too much sunlight can cause sunburn which has been linked to an increased risk for cancer[28].

External Standards

Our project will make use of circuit boards and will therefore have to follow IPC standards for the boards. As our project is a general electronic product, it falls into IPC Class I which corresponds to IPC-2221 standards[29]. These standards will guide how the boards are populated, design layout, testing procedures, and integration[30]. The project took advantage of a trace width calculator online by 7PCB[31]. Since our project will be indoors and not meant for exposure to the elements, our project should also conform with the NEMA Ratings for Enclosures: Type I[32]. This level of standards is to provide personal protection against hazardous parts within the project, and protection of project equipment from falling debris such as dust, dirt, and light ingress/splashing of water[33]. The wireless communication between the light sensor and the microcontroller will have to follow IEEE standards 802. Standards 802.11 and its derivations (a-n) are necessary for local area network, Wi-Fi connections. Standards 802.15 give the requirements for Bluetooth connected devices[34], [35]. Finally, the coding on the microcontroller should follow the standards from C programming by Barr[36]. The standards by Barr will ensure code readability for the other programmers as well as reduce the chance for bugs. For the model test environment, the frame used 80/20 T-Slot Aluminum pieces. Each of the pieces conforms to an industry standard of sizes[37].

Tools Employed

A variety of both hardware and software tools were utilized to design, create, and test the hardware and software of ALTAIR. Tools used in each category are described below.

Hardware

To design the Window and Remote Node PCB, National Instrument's Multisim[9] and Ultiboard[10] were utilized. Multisim was used to design the window node and remote node circuitry. Several of the components needed to design the circuit were not in the standard library of parts. In order to build the circuit, the nonstandard parts were individually created. Following the Multisim design, the circuits were exported to Ultiboard. Ultiboard was used to create a layout for the PCBs. Similar to Multisim, several components were not in the generic libraries.

To create the parts in Ultiboard, datasheets and reference sheets were consulted to determine the correct dimensions for the parts. As for designing the various hardware components such as the housings for both nodes, servo mounting and attachment to the blinds, and parts for the test rig, Autodesk Fusion 360[38] was used. Steven was proficient enough to get preliminary versions of each part designed in Fusion 360, but he had to learn new features within the software to create a polished final product. To manufacture the designed housings for both nodes, the Creality CR-10 Smart 3D Printer [39, p. 10] and the Ultimaker Cura [40] slicer was used. For the servo mounting, an OMAX Maxiem 1515 Waterjet [41] was utilized to turn a 0.090" sheet of aluminum into a bolt-on bracket to the blinds. As Steven was new to 3D printing, he had a quick learning curve of how to use it along with what some of the settings did to make a part that would be suffice in quality. As for using the Waterjet, Steven has over 3 years of experience using it and did the job in a quick and timely manner.

Firmware

To create the firmware for the CC3220SF, TI's Code Composure Studios (CCS) [42] was used. Alex and Mason had previous experience with CCS, and thus were able to work efficiently with it, although they both had to learn new features of CCS when it came time to switch output pins around. Along with CCS, the TI SimpleLink™ Wi-Fi® CC32xx software development kit (SDK) [43] was heavily utilized for the Wi-Fi connectivity between the CC3220SF and the Raspberry Pi. While this was a very useful tool, there was a learning curve as Alex and Mason had never worked with any Wi-Fi module before. Finally, a NI VirtualBench [44] was also used to assist in debugging the custom drivers. As Alex and Mason also had previous experiences with the VirtualBox through the Fundamental of Electrical Engineering series, they had no issues using the tool to troubleshoot.

Software

To implement the user interface, the micro web framework Flask [45] was used in conjunction with Python[46], as the connected Philips Hue Hub had an easy-to-access API that could easily be accessed with Python, along with power efficiency reasons. To share the code between us, GitHub [47] was also set up so we all could view the code at any time, along with having access to previous versions of the code as well.

Ethical, Social, and Economic Concerns

In its general form, this project wouldn't have any more ethical concerns that traditional curtain/blinds. The purpose of our project was to maintain a constant light level within a room by wirelessly actuating the blinds and controlling lights. The same task could have been accomplished a person manually controlling the blinds and lights. Blinds can act as a form of security for the user's houses, and therefore add some benefit to io functionality.

When looking at the substance of the project, that is where ethical questions arise. In the traditional setting, the blinds and the lights could only be affected by an individual inside the house. Therefore, residents would know the last positions of the blinds and if they had been moved/disturbed by an intruder. In the current state of the project, the connection between the user, blinds, and lights have been transferred to a local network. Therefore, in order to connect with the blinds and change them, the user must have access to the user interface from a computer. This provides the opportunity for malicious actors to hack into the local network and disturb the resident. The blinds could be opened at unusual time, such as the middle of the night, without the consent of the user. The CC3220SF comes with some basic security protocols to protect against such actors, but the risk is never extinguished. The project is not connected to the network which provides a layer of protection; another layer is added by setting a password to the local network so that not everyone could join on will. However, if someone did manage to get into the network, the data is not encrypted and the malicious actor would have access to all the data. One thing to note is that the malicious actor would need to be in close proximity with the local network, such as in the house, in order to have the opportunity to get onto the local network.

At a low level of production, such as the current state of the project, the economic cost would be high to implement for all people. If all components worked and there was no need for extras, the cost would be around $250. The cost would be comparable to the price of what is considered high quality, motorized blinds. As such, the product would mainly be bought by the middle and upper classes. For low class citizens, this product would be a nonessential good as traditional blinds could fulfill the same job. However, if the price can be reduced by increasing the scale of production as noted in the Costs section, compacting the PCBs, and decreasing the computational power necessary from the microcontroller unit, the project could be on par with a good pair of traditional blinds, around $150.

Intellectual Property Issues

The “Thermochromic Window Structure”[48] is added onto windows that let’s light pass through or blocks light from entering. The patent claims at least one pane of a window with a sunlight responsive, thermochromic layer. Based on the position of the sun rays’ incident on the window will cause the structure to warm at varying rates due to convection currents within the window structure. As the window heats, the thermochromic windowpane will darken and therefore let less light pass through as the window structure warms. When the sun is not out at a high angle such that the light rays are not directly incident on the window, light will pass through. When the sun is at a low angle and therefore almost directly incident on the window, the thermochromic layer will darken, and less light will pass through. This patent provides a relatively simple solution to stop natural light from entering an area.

The “Method and Device for Lighting Control in Space Inside of Room”[49] determines how to control the light within a room. The device within the patent can take in several different parameters that affect the amount of daylight coming into a room, such as, but not limited to: geographic location, local weather, cloud coverage, and climate. The device uses the local information to synthesize a light profile for a room. The device is designed to illuminate a room located on the interior of a building with no access to natural lighting, and to replicate the lighting condition an occupant would receive if near a window.

The “Solar Radiation Adjusting Device for Lighting Louver”[50] is a full window attachment designed to block sunlight. The device is attached to the outside of the window. The main components are the louver and a sunshade. The louver is designed to be raised and lowered and allow for different amounts of natural light to be let into a room. The slats on the louver can also be rotated. All louver motion is controlled by a motor. The sunshade is connected to the bottom edge of the louver device in a separate case. The sunshade serves to block the heat from entering the interior. The height of the sunshade is also controlled by a motor.

The “Motorizable Tilt Shade System and Method”[51] is a manual and mechanical method for controlling the height and tilt of slats within blinds. The patent assesses the risks and limitations of solely a manual or mechanical system such as ease of operation, danger of hanging cords, limit space around a window, and the technical skills required to personally convert a purely manual system into purely mechanical system. This patent attempts to address these issues. The system consists of a header, driveshaft, suspension chord, tilt chord, motor driver, battery holder, and power delivery subsystems. With the limit space and for aesthetic appeal, the battery holder, motor, motor encoder, controller, and driveshaft are located within the header. The slats have a tilt freedom of 180°. Either the chords or the motor system may be used to adjust the height and/or tilt of the slats.

The “System to Control Daylight and Electric Light in a Space”[52] is an illumination maintenance system in a desired area. The patent attempts to maintain an illumination profile within the area through exterior and interior lighting. The patent sets up one or more zones within a room which each have a light sensor and a dimmable lamp. Each of the zones are preset to maintain a constant light level throughout the day through the combination of exterior light and interior dimmable light. The system is also built to implement window treatments, such as shades, as another means of controlling the illumination. The patent does note that occupants within the room may affect the light sensor readings, thereby disrupting the feedback control loop. Techniques such as averaging multiple light sensors are presented in order to remedy this drawback.

The first, fourth, and fifth patents/paragraphs within the Intellectual Property Issues sections are US patents. The second and third patents/paragraphs are foreign patents that provide more context into the current state of the art. The first patent and the ALTAIR system both provide methods for reducing the amount of light entering an area. The first patent provides a single step solution for reducing light but has no way of maintaining the light within an area. The second patent provides a method for building a light profile within a room. That patent is concerned with building the local light profile to mirror exterior conditions all day, as opposed to maintaining constant conditions. The third patent provides a mechanical solution to reduce the amount of light entering a window. This patent is intended to be in an outside environment and is not intended to maintain a constant light level within an area. The fourth patent is another mechanical solution to reducing light entering through a window. The patent is intended to be placed on the interior of a building and has the benefit of being a manual or a mechanical system. Like the first and third patents, the fourth patent doesn’t maintain a light level. The final patent is almost exactly the system from the ALTAIR project. Both systems use a zone model with light sensors. The light sensors are used to gauge the illumination levels, and control either dimmable lights or a window treatment to maintain a preset level. Based on the current state of the art, it would be difficult to receive a patent. The first four patents listed would pose no interference. The final patent's system is very similar to ours. It is unlikely that implementing an IoT interface introduces enough novelty into the design to warrant receiving a new patent.

Detailed Technical Description of Project

The goal of the project was to develop a system for controlling the amount of light within a designated area. The project was based between two nodes: a window node and a remote node. The window node was centered around a TI CC3220SF-LAUNCHXL microcontroller, servomotor, and light sensor. The remote node was responsible for actuating the servomotor to control the blinds. The remote node was centered around a Raspberry Pi 3A+ and a light sensor. The remote node is responsible for hosting the user interface collecting the ambient light data for the region of interest within the room and communicating with the window node and smart lights. The nodes communicated wirelessly via Wi-Fi to avoid extraneous wires within the room. The overall communication structure is shown in Figure 4: System Diagram.

Our project is an IoT device that automatically adjusts the window blinds and interior lights to achieve a desired brightness within a room. It works by first obtaining a desired light level from the user with a web application, then it determines the current light level of the room using the remote light sensor. If the current light level sensed by the remote light sensor is below the desired level, then the room will open the blinds using a servo connected to the shaft of the blinds. If the light coming through the window is not enough to satisfy the requested light level, then smart lights around the room will turn on and increase in brightness until the desired level is reached.

Mechanical

The mechanical aspect of our project is actuating the blinds with electronics. While the team tossed around a few ideas of what type of motor to use that would be the most optimal for the project, a micro servo was the dominant choice. There were a few factors that led to this decision: affordability, simplicity, size, and power. To determine how much power was required for the motor to produce, the amount of torque required to rotate shaft that tilts the slits of the blinds when the blinds are fully extended was measured by using a custom test rig, along with the torque equation $\tau = rFsinq$.

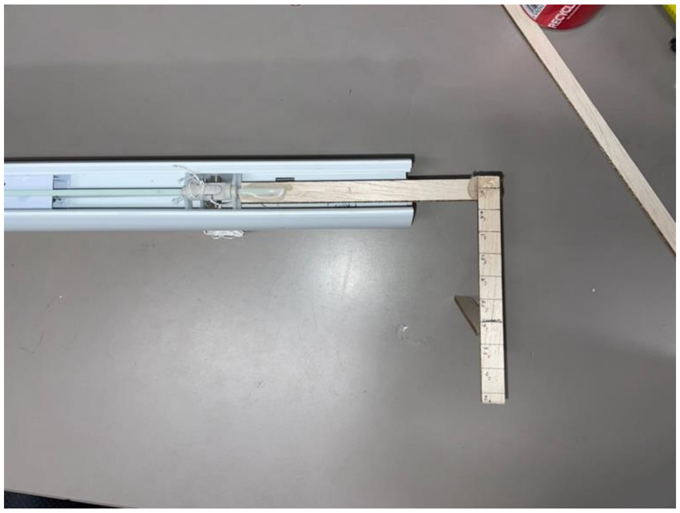


Figure 2: Blinds Torque Test

By using the rig in Figure 2: Blinds Torque Test with coins of a set weight, it was possible to calculate the worst-case torque required to turn the blinds, which was ~0.69 oz-in. With the torque requirement now determined, the packaging of the servo was the next issue to work out. The blinds stock housing was measured and found to have ~33mm of height to spare and ~22mm width to spare. After some research on Digikey, it was found that DFRobot offered a micro servo (SER0046) that produced 1.5 kg-m (20.83 oz-in) of torque and was 33.4mm tall and 12.2 mm wide. While the height was a little over than what was ideal, it was still workable and the most affordable at $6.90 each, which includes all the accessory horns as well which would later be utilized.

Once the servo was decided upon, a mounting solution had to be designed so that the servo would not shift around when rotating the shaft. First, a CAD of the general size of the blinds along with the servo was created. Afterwards, a few ideas for attaching the servo to the blinds were brainstormed, some of which included a direct drive splined adapter to the shaft, a male to female splined adapter that went from the servo straight to the cylinder the shaft was rotating, and a horn-adapter that would bolt to one of the existing horns of the servo and would slide over the shaft. The third option was the one that the team decided to use, primarily for the ease of manufacturing as well as the sturdiness of being made from aluminum instead of plastic, as the other two options required a spline connector to connect to the servo, which is extremely difficult to do using the subtractive manufacturing machines available at Lacy Hall. Fortunately, the third option was able to be made on a manual lathe as it was concentric. Using scraps of aluminum tubing lying around, a prototype of the adapter was able to be manufactured and was successfully validated.

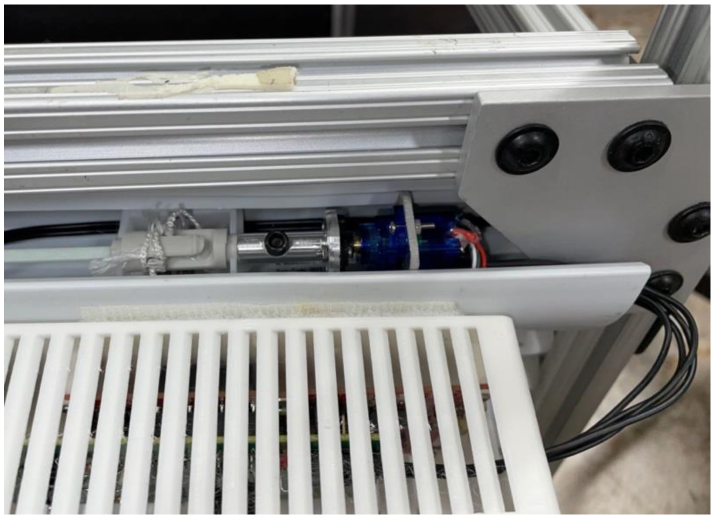


Figure 3: Servo Mount and Adapter

Next, the physical housings for both the window node and remote node were designed and created. The primary goal of the housings was to protect the PCB from physical damage, as well as hiding any exposed connectors internally and limiting the non-essential connections as well. The nodes were designed in Autodesk Fusion 360 after the PCBs came in, so that the actual sizes were accurately accounted for. The remote node has a single hole to allow for micro-USB type B power input, while the window node had two holes: one hole on one side for the power adapter, and one larger hole on the other side for both the debugging micro-USB input as well as the servo connector. Once designed, slits were added to the lid and the body of the window node to aid in ventilation, as well as allowing more light onto the light sensor integrated on the custom PCB. A similar approach was taken for the remote node in which ventilation holes on the lid were implemented to allow more light at different angles to hit the light sensor, rather than letting the system rely on light hitting the sensors head on. Figure 4 shows the CAD design for the window node housing and Figure 5 shows the CAD design for the remote node housing[53]. The 3D printer used can be found in Figure 30 in the appendix.

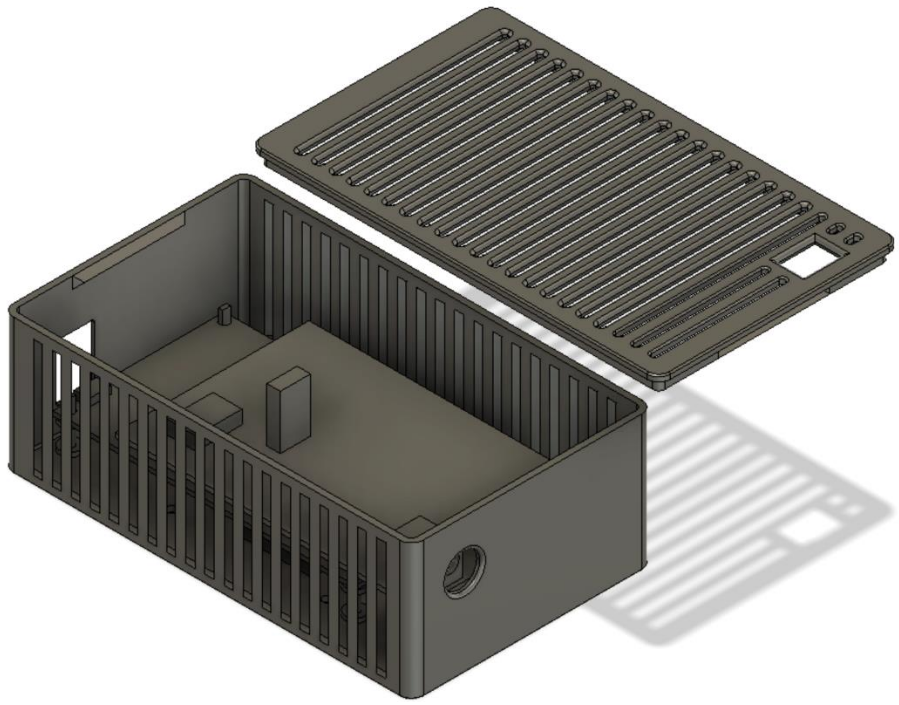


Figure 4: Window Node Housing CAD Design

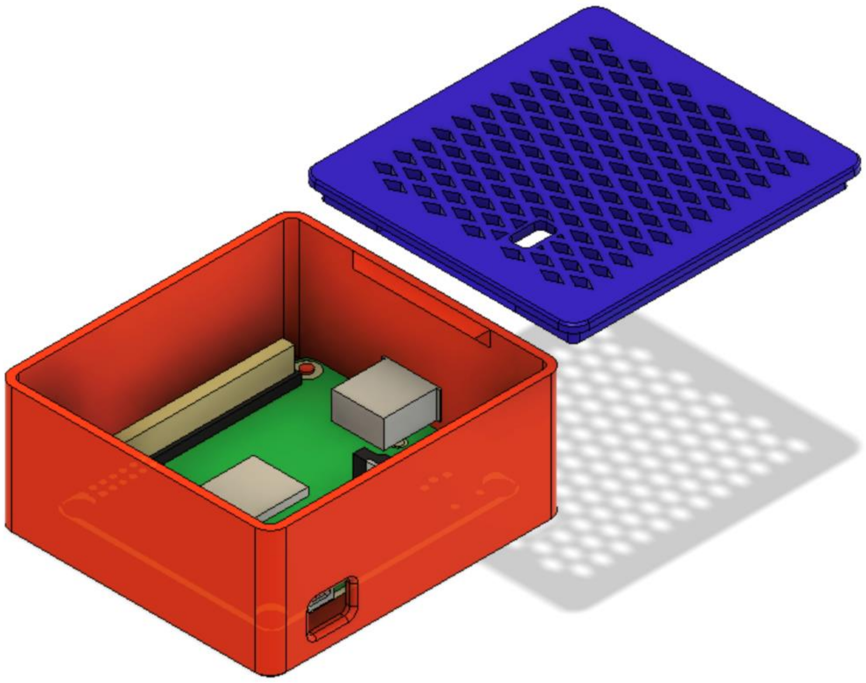


Figure 5: Remote Node Housing CAD Design

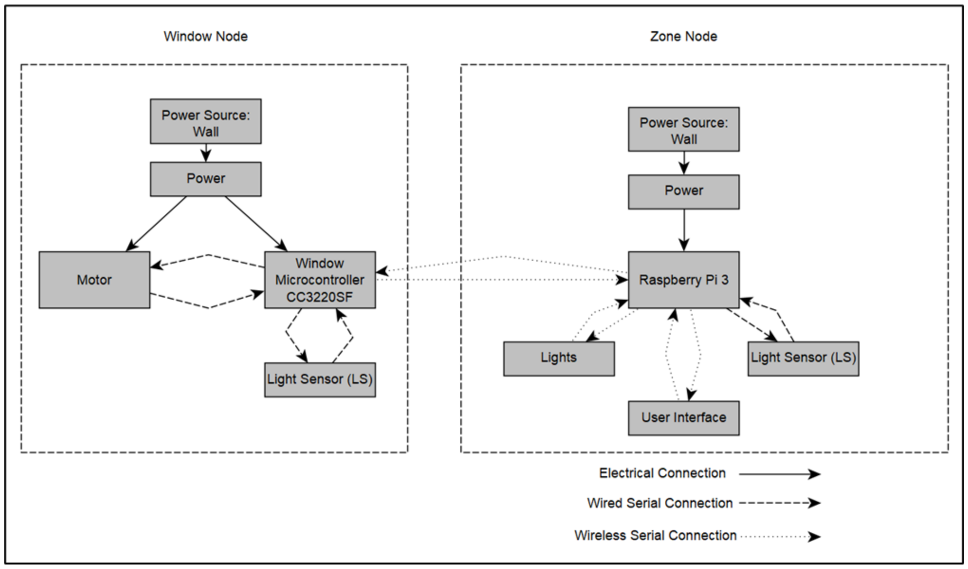


Figure 6: System Diagram

Window Node

The window node was designed to be placed at the top of a window to control the blinds. The window node consisted of the TI CC3220SF-LAUNCHXL Microcontroller, Window Node PCB, DFRobot DC Servomotor, and wall power supply. The overall design of the window node is centered around the PCB, and the PCB design from Multisim is shown in Figure 7.

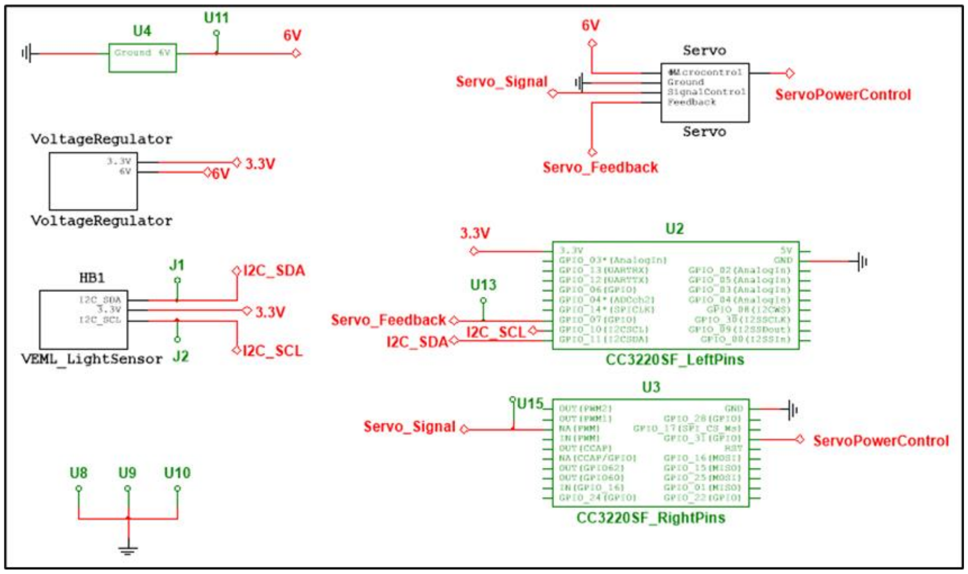


Figure 7: Window Node Schematic

The PCB was segmented into five parts: input electrical power from the wall, the voltage regulator, the VEML7700 light sensor, the servomotor, and the CC3220SF. The first thing we the team looked at was the supply power to the system. Looking at the other components, the SER0046 servomotor required 6V, the CC3220 required 3.3V, and the VEML7700 light sensor required 3.3V. To determine the power necessary, the team looked at the maximum current draws for the same three devices. The SER0046 has a peak current draw of 1.1A, the CC3220 has a peak current draw of 0.286A, and the VEML7700 has a peak current draw of 45µA. The combined current draw is calculated to be around 1.4A. The team chose a 6V, 12W wall power supply[54]. The current supplied would be about 2A, which provides about a 0.6A buffer. To connect the wall power supply to the PCB board, the team used a PCB barrel jack[55].

From the 6V power supply, the voltage needed to be taken down to 3.3V in order to supply power to the VEML7700 and the CC3220. The voltage regulator subsection was designed and is shown in Figure 8. The specific voltage regulator used was the TI LM1086CT-3.3NOPB[56]. From the datasheet, the minimum voltage required to maintain a 3.3V output was 5V; the 6V input from the wall power supply should be sufficient. Capacitors were added on the input and output sides of the regulator to account for any transients that may occur.

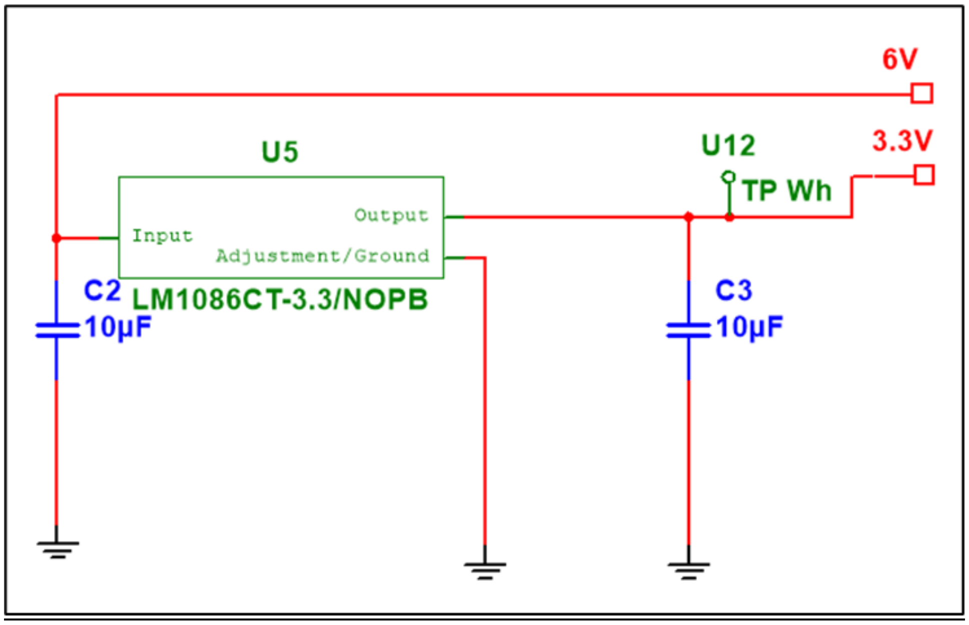


Figure 8: Voltage Regulator Schematic

At the window node, the light levels are measured by the VEML7700 light sensor [57]; the schematic to power and send the data is shown in Figure 9.[57]; the schematic to power and send the data is shown in Figure 9. From the datasheet, the supply voltage to the sensor must be between 2.5V and 3.6V. As the CC3220 can also be run 3.3V, which is in the range of the VEML7700, the team chose to run the light sensor at that 3.3V. The VEML7700 light sensor itself was chosen from a several potential sensors. A primary reason for choosing the VEML7700 was its spectral response to light. The team wanted a sensor that had a spectral response that was similar to that of the human eye in order to achieve the most accurate light level. From the datasheet, there is significant overlap between the spectral response curves of the sensor and the human eye. The human eye is slightly more sensitive to light with wavelengths between 550nm and 650nm, but the difference doesn't pose significant concern. The VEML7700 is a digital sensor. The sensor has an analog to digital conversion process which outputs a 16-bit integer. This is beneficial over an analog sensor which would require the team to use the ADC on the CC3220. Therefore, the sensor can communicate directly with the CC3220 via I^2C protocol.

The light sensor sub-schematic was based off a common application note within the datasheet. The pullup resistors, noted as R1 and R2 within Figure 9, were given a range of 2.2kΩ to 4.2kΩ. The team had access to several resistors from the ECE Fundamentals sequence lab kit and decided to use 3.3kΩ resistors. The SDA and SCL lines are connected to corresponding lines on the CC3220.

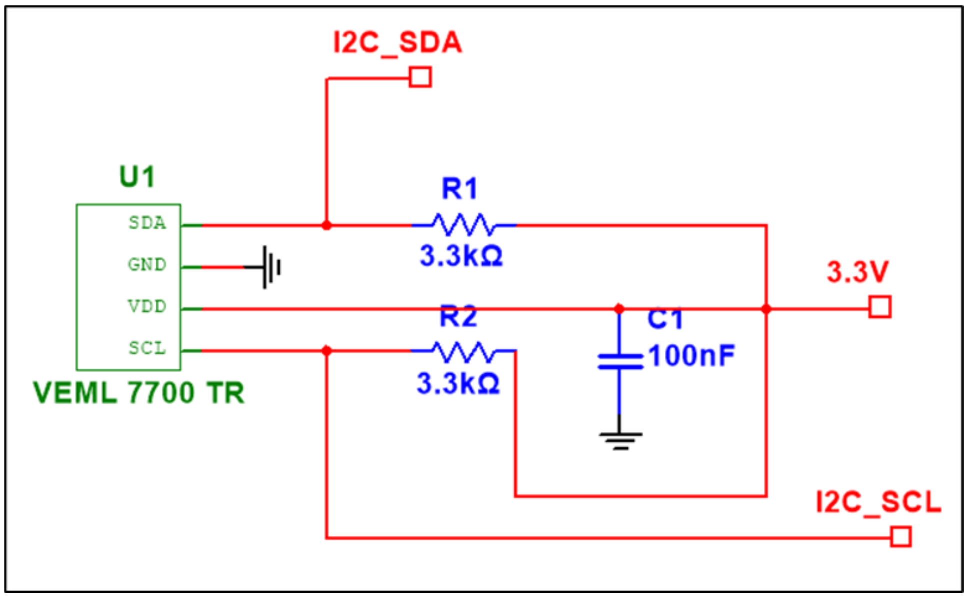


Figure 9: Light Sensor Sub-Schematic

The window shade position is controlled by the DFRobot SER0046 servomotor, which receives commands from the CC3220. The servomotor circuit makes use of the 2N3904 NPN transistor[58], LP0701 PMOS[59], and 1N5817-TP diode[60]. The 2N3904 is connected to the CC3220 microcontroller. When a signal is sent from the CC3220 to the transistor, the PMOS is activated such that the servomotor is powered by the 6V. This feature was added so that the servomotor is not constantly connected to a power supply. The diode is connected between the 6V power supply and the ground terminals. Inductive energy stored within the servomotor will be dissipated as it flows through circuit. The servomotor has four terminals: power supply, ground, signal, and feedback. The power terminal, as discussed above, needs 6V. The signal control and feedback terminals are connected to the CC3220. A pulse width modulation (PWM) signal is sent from the CC3220 to the servomotor on the signal control line. This line provides commands to the servomotor to adjust the position of the blinds. The feedback line sends a 0V to 3.3V signal from the servomotor to the CC3220. The feedback line relates the voltage to the head position of the servomotor. The analog signal from the feedback line will be deciphered by the CC3220 using the analog to digital convertor.

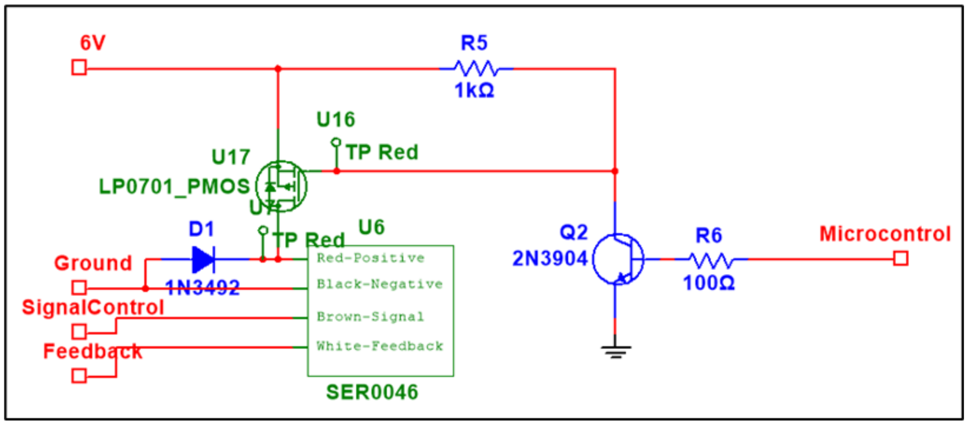


Figure 10: Servomotor Sub-Schematic

The CC3220SF-LAUNCHXL microcontroller has two sets of 20 pins connected to various functions as shown in Figure 7. The CC3220 user guide[6], datasheet[61, p. 32], and quick start guide[62] were all referenced in order to determine the functions of each pin. The power pin, ground pins, and I^{2}C communication pins were all determined from a limited number of pins. The servo power control, servo feedback, and servo signal pins all had multiple options from many general-purpose input/output pins.

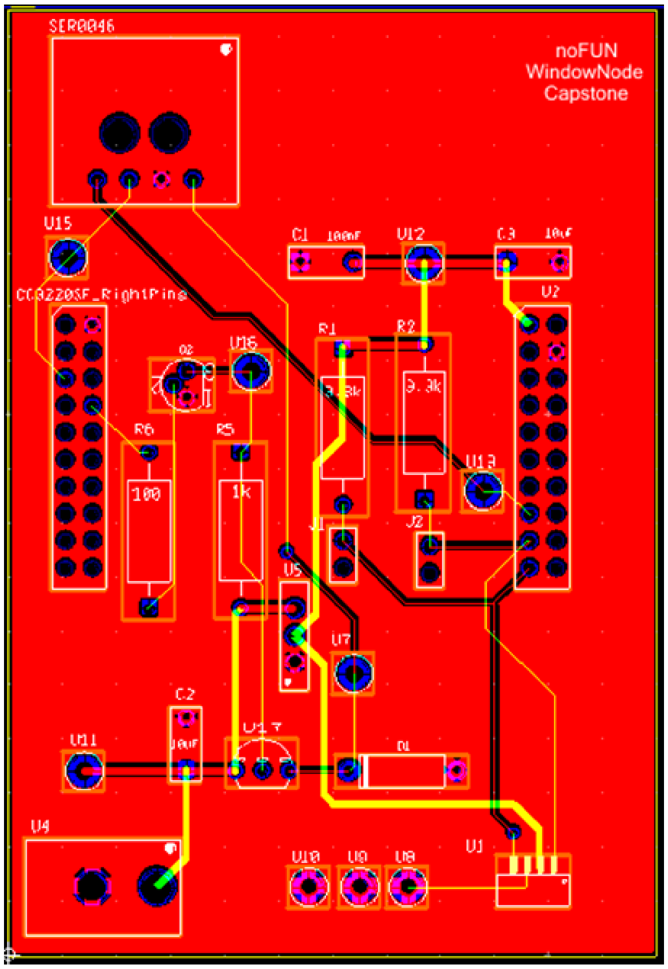


Figure 11: Window Node Ultiboard Design

After completing the Multisim schematic for the window node, the circuit was transferred into Ultiboard for the PCB layout. Due to the limited database, custom components were made for the power barrel jack, light sensor, PMOS, CC3220 microcontroller, and servomotor connector. Traces that trended in the north-south direction on the board were generally routed on the copper top layer. Traces that trended east-west were generally routed on the copper bottom layer. It was decided that the servomotor connecting should be placed on the north side of the board in order to have access to the header of the blinds. To measure the light and the window node, the light sensor was placed at the southern end of the PCB in order to reduce shadowing effects, and thus get the best representation of light at the window node. The microcontroller was laid over top of the other components in between the servomotor connector and light sensor.

After the board was manufactured, the team noted a couple items for future iterations. Visual inspection noted that the distance between components was generally too much. Future iterations should have the components packed closer together. The width was limited by the width of the CC3220 which would be placed on top of the PCB, but the height of the board could be easily reduced. If the boards were fabricated on a per area basis, cost could be saved by a more compact design. Similarly, a more compact design would be beneficial when mounting on the window. During testing of the PCB board, it was determined that the pin originally used for the servomotor PWM control would not be available. Therefore, output originally on pin 17 was moved to pin 64. Despite these hiccups in the board design, the team opted to not send out the board for remanufacturing in order to conserve the budget. The final product after manufacturing is shown in Figure 12: Manufactured Window Node.

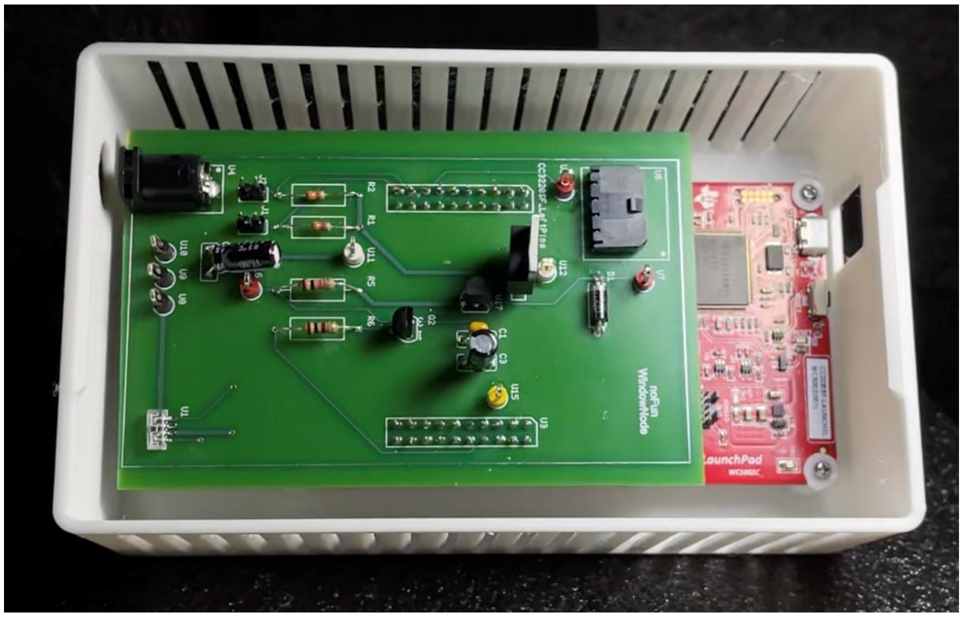


Figure 12: Manufactured Window Node

Remote Node

The remote node was designed to be compacted so that it could be placed within a room. The window node consisted of a Raspberry PI 3A+, VEML7700 light sensor, and power supply.

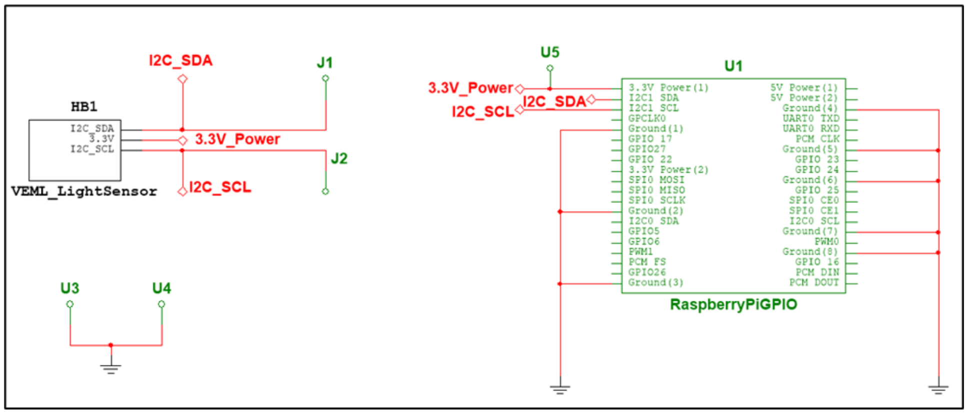


Figure 13: Remote Node Schematic

For the remote node, power is supplied directly to the PI through a dedicated USB cable. Power to the VEML7700 light sensor is provided from the 3.3V output pin on the PI. The PCB board is placed on top of the header of the PI. The light sensor sub-schematic shown in Figure 9: Light Sensor Sub-Schematic, is the same sub-schematic from the window node, shown in Figure 9:

Light Sensor Sub-Schematic. The only difference is the power supplied to the light sensor is from a PI pin rather than the voltage regulator.

After the circuit design was completed in Multisim, the schematic was transferred into Ultiboard, and the final design is shown in Figure 14: Remote Node Ultiboard Design. The remote node PCB size is dominated by the length of the header of the PI. Compared to the window node, the remote node features fewer components.

After the board was manufactured, the team noted a couple of items for future iterations. A visual inspection noted that the horizontal distance between pins was too much. Unlike the window node which could have been shrunk in 2 dimensions, the size of the remote node could have only been reduced in the east-west direction. As the PCB is placed on top of the PI, there would be no part interference, and a smaller board would not have impacted the design. As noted in the window node section, a smaller board would save materials and cost for large scall production. However, the size change to the board didn't warrant remanufacturing the board.

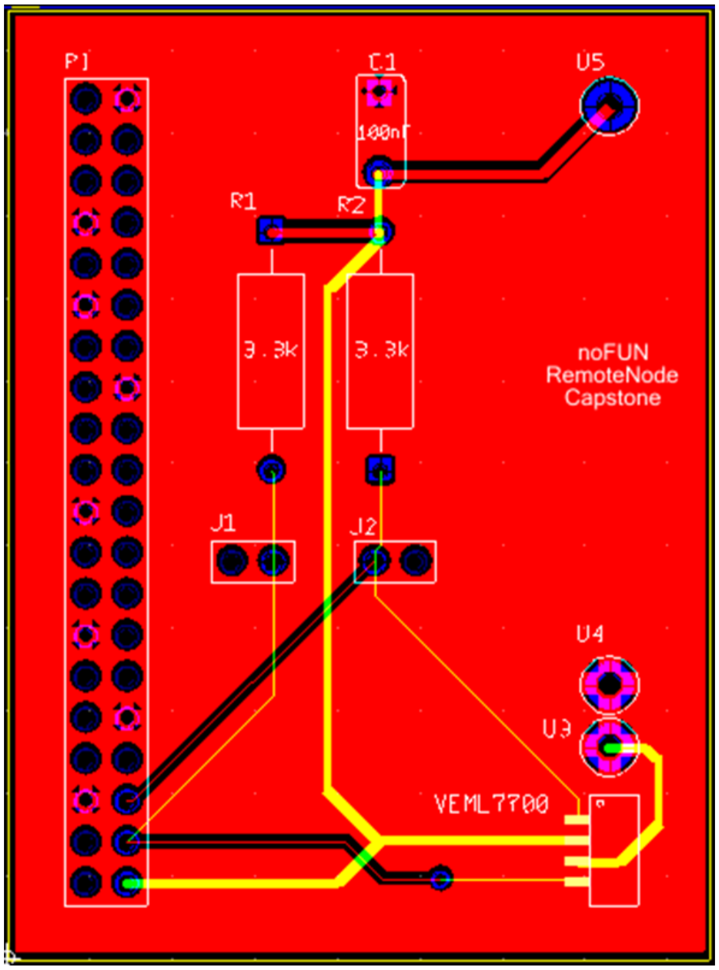


Figure 14: Remote Node Ultiboard Design

The manufacturing Remote Node is shown in Figure 15: Manufactured Remote Node.

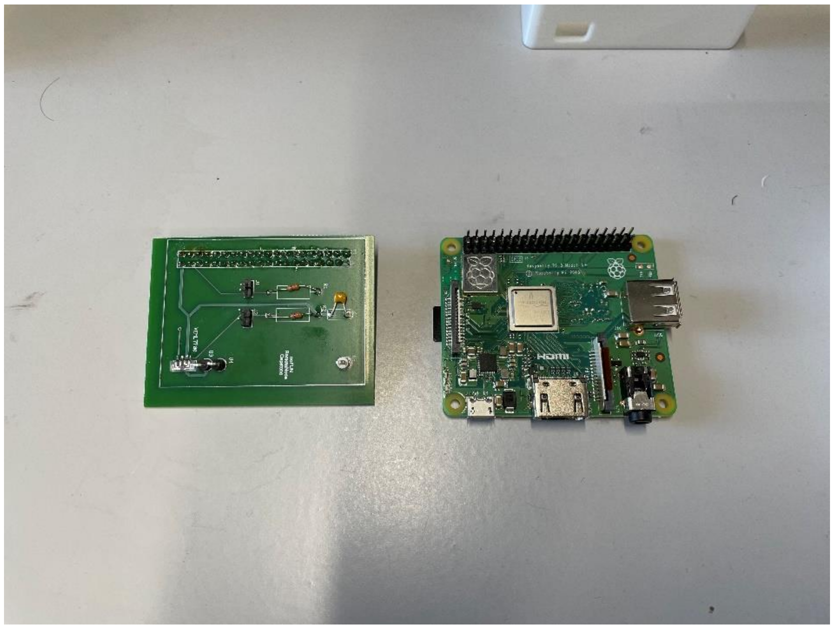


Figure 15: Manufactured Remote Node

Firmware

The firmware refers to the device drivers and abstractions and the wireless communication drivers and abstractions on the microcontroller and the Raspberry Pi. The Raspberry Pi drivers are covered in the software section below. Thus, this section will cover the servo driver, the ambient light driver, the digital filter for the light data, and the wireless communication driver on the microcontroller—all shown in Figure 16: Data Flow Diagram for Window Node. This section will also cover the multithreading running on the microcontroller.

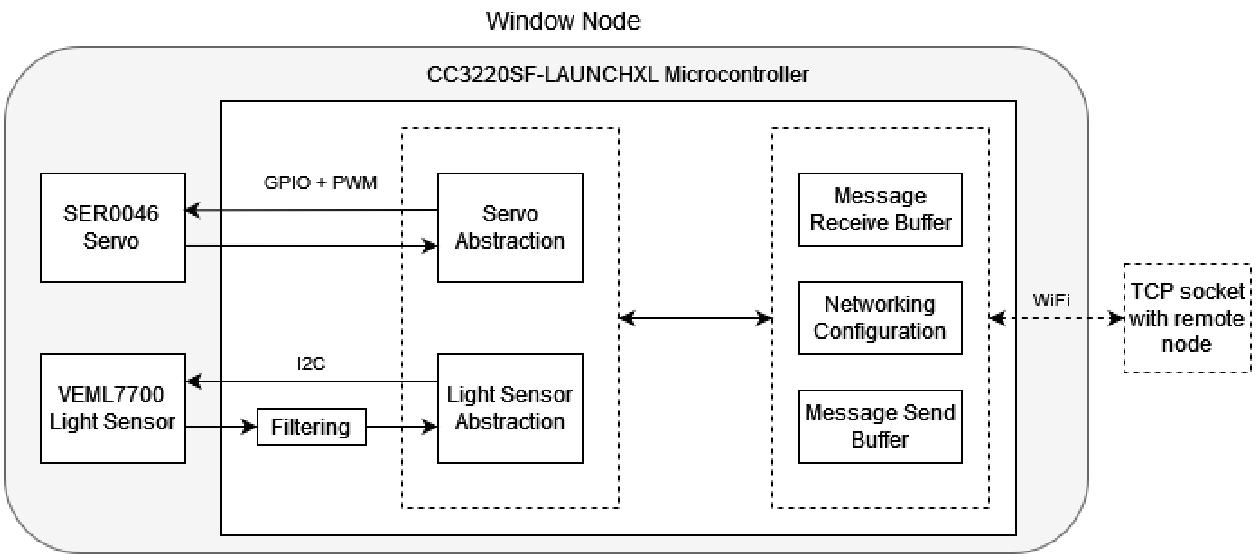


Figure 16: Data Flow Diagram for Window Node

The wireless communication utilized the SimpleLink SDK[43] in order to connect to a Wi-Fi network and a TCP socket as a client. Our messaging scheme was composed of a receive buffer that was populated by data sent from the remote node and a send buffer that was populated by function calls and whose data was sent to the remote node. The buffer scheme was as follows:

Table 1: Send buffer

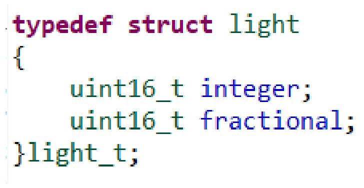
ACK	Light Sensor (high byte)	Light Sensor (low byte)	Server on	Server feedback	END

Table 2: Receive buffer

ACK	Want server on	Servo position	END

A driver utilizing one of the Pulse Width Modulation (PWM) modules and a GPIO pin was developed to control the SER0046 Servo[19]. This servo took a range of valid duty cycle values from 500 to 2500. A design tradeoff made was to only use 8 bits to store the position of the servo—meaning that only one byte in the receive buffer was needed to store the servo position. This was good for messaging, but limited the number of possible servo positions. However, this was a tradeoff found worth taking, as using only 200 servo unique positions was found to be more than adequate to give a smooth spread of possible servo positions to be used in the control algorithm. Thus, the values passed from the control algorithm were limited to be from 0 to 199, and were mapped to the valid duty cycles by the function $f(x) = 10x + 500$. The servo driver also used a GPIO pin to supply power to the device. This was to conserve energy consumption when the servo was not being actuated much throughout today.

The light sensor driver relied on using the I2C bus to interface with the VEML7700 light sensor[57]. Because the system should have a smooth approximation of what the human eye sees, the data was fed into a digital filter; specifically, an exponential average. The formula for getting a new data value in an exponential average is ((current value)*(15/16) + (read value)*(1/16)). This makes sure that big but short changes in ambient light do not wreak havoc on the system's control algorithm and output. In order to do exponential averaging, the system needed to shift right 4 times (divide by 16). However, the microcontroller did not support floating point, but because there was addition and multiplication alongside the division the system could not disregard the decimals without very poorly approximating the exponential averaging. Thus, the management of this filtering is done by using a custom floating-point notation called light_t, shown in Figure 17. When an integer light value is filtered, its 4 low bits shifted out of the integer (when it is divided by 16) are put into the fractional component. Later, when the additional and multiplication are applied to the integer portion, they are also applied to the fractional portion. Any overflow from the fractional portion is added to the integer portion for the final result of the filter operation. This preserves the decimal values to be used in arithmetic without actually storing them as decimals.



```
typedef struct light
{
    uint16_t integer;
    uint16_t fractional;
}light_t;
```

Figure 17: Light data struct for exponential averaging

The filtered light sensor value is stored in a 16 bit unsigned integer—so it takes up two bytes in the send buffer.

There are three threads running on the window node: one main thread, an I2C thread, and a messaging thread. The main thread initially configures the board, peripherals, Wi-Fi, and TCP socket connection, and then manages the UART terminal (useful for system debugging). The messaging thread manages the TCP socket connection with the remote node and drives the servo. Specifically, it waits to receive a message from the remote node. When it does, it parses the data, drives the servo (if necessary), grabs the current filtered ambient light value, populates the send buffer, and sends a response back to the remote node. The I2C thread periodically polls the I2C device to update the currently stored filtered ambient light value.

Software

The software is entirely on the Raspberry Pi. The diagram in Figure 17**Error! Reference source not found.** shows an overview of this software on the Raspberry Pi. The Raspberry Pi hosts a website with flask to serve a user interface to the end user. The Raspberry Pi is connected to the router and user can connect to it through the LAN. The code is broken up into four main sections. A communication section to handle the interface to the CC3220 over a TCP socket. A state section to handle the state of the whole system including tracking state variables and updating the state of the system. A control thread to make decision on updating the state. And finally, a user interface to allow the user to change the device.

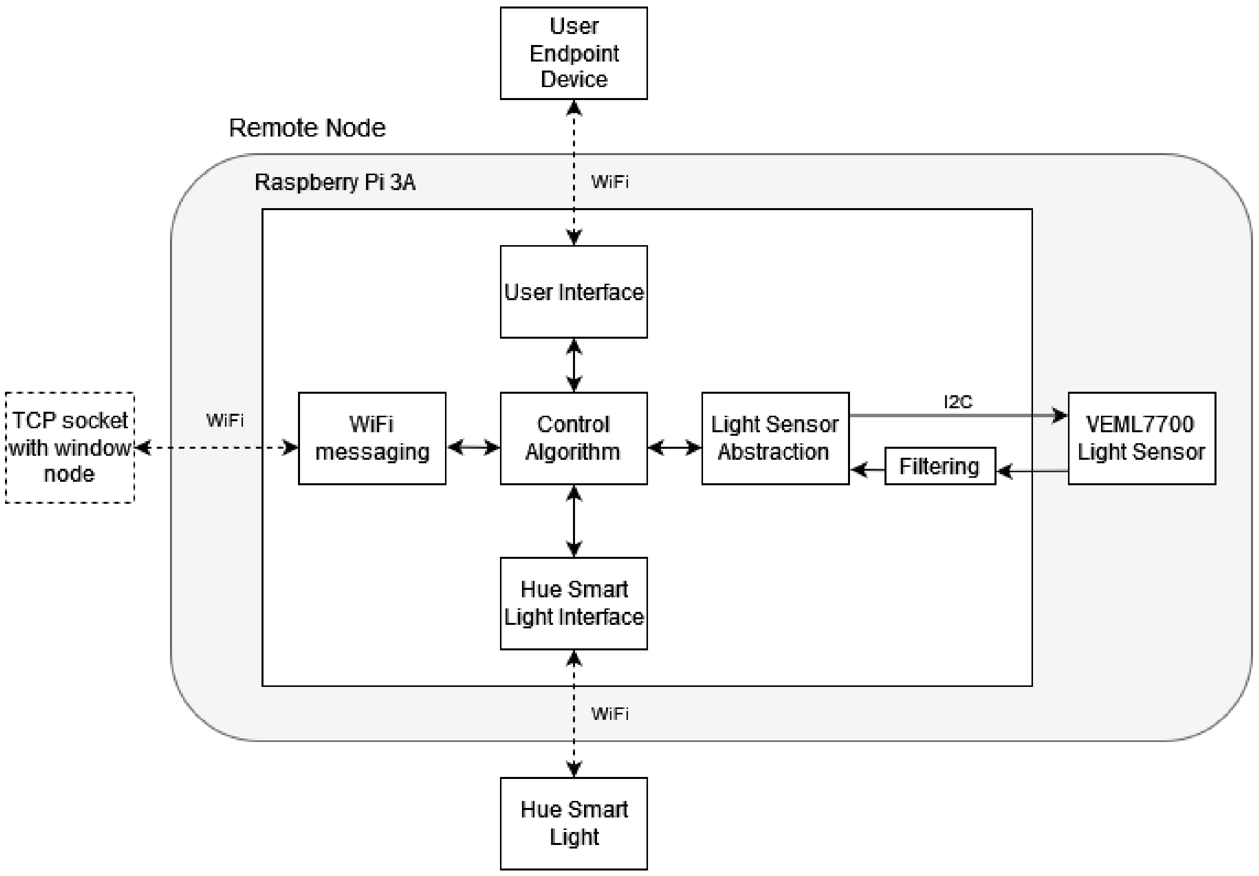


Figure 18: Data Flow Diagram for Remote Node

<u>Wi-Fi Messaging</u>

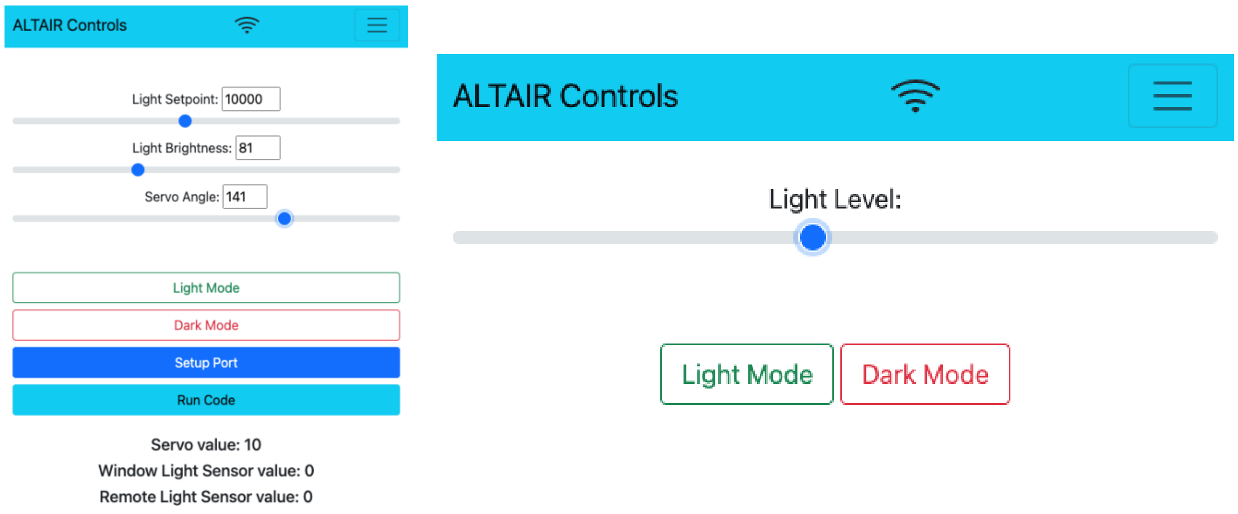
Wi-Fi messaging is accomplished using a TCP socket. The CC3220 tries to connect to an open socket on the Raspberry Pi. Once this connection is successful the Raspberry Pi can send messages to the CC3220. The Raspberry Pi sends a 4 byte messages to the CC3220 consisting of an ack byte, whether the servo should be turned on or off, a servo position, and an end of packet byte. The CC3220 will then respond with a packet containing the measured position of the servo and the current light value measured at the window.

<u>Hue Smart Light</u>

The IoT smart light is a Phillips Hue lightbulb. This lightbulb is controlled by a Philips Hue Bridge with connects directly to the router via ethernet. The Phillips Hue Bridge communicates with the actual lightbulb over Zigbee. From a software perspective, we communication with the bridge with a restful API. To control a lightbulb, we send a push request to the Hue Bridge containing a json object specifying the ID of the light bulb and the requested state of the lightbulb.

<u>User Interface</u>

The user interface is hosted on a micro web framework Flask server. We used Flask since it is very light weight and energy efficient. On top of that, we are using the React.js[63] framework to deliver a responsive webpage, along with the Bootstrap Framework[64] for quick and easy styling. Figure 19: User Interface shows a screen shot of the user interface. Each button corresponds to a Python function runs code to execute the function. To begin a connection with the window blind, a user must first select “Setup Port,” which will cause the Raspberry Pi to host a TCP connection that the CC3220 can connect to. In addition, the Raspberry Pi will spin up the controller thread that decides the blind and IoT light state for achieving the desired light level. After that a user can turn the lights on and off and control the set point of the system with the left slider.



Light Setpoint: 10000
Light Brightness: 81
Servo Angle: 141

Servo value: 10
Window Light Sensor value: 0
Remote Light Sensor value: 0

Figure 19: User Interface

Control Thread

The control thread decides what light and blind state to do to achieve the desired light intensity. It begins by running the control loop in **Error! Reference source not found.** This control loop starts by comparing the current light intensity to the desired light intensity. If the controller needs to increase the value, it will begin by increasing the blinds if they are not maxed out and then waiting a 400 ms for the value to settle down. If the blinds are already opened to the maximum extent, the device will instead increase the light intensity. Similarly, if the controller needs to decrease the light intensity it will first do so for the lightbulb and then decrease the natural light coming from the blinds. In this way, the system will always prioritize natural light.

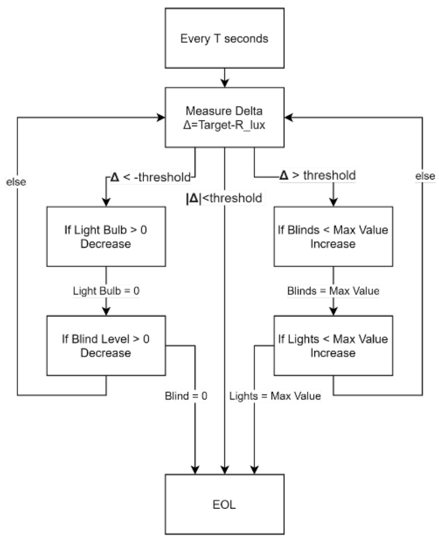
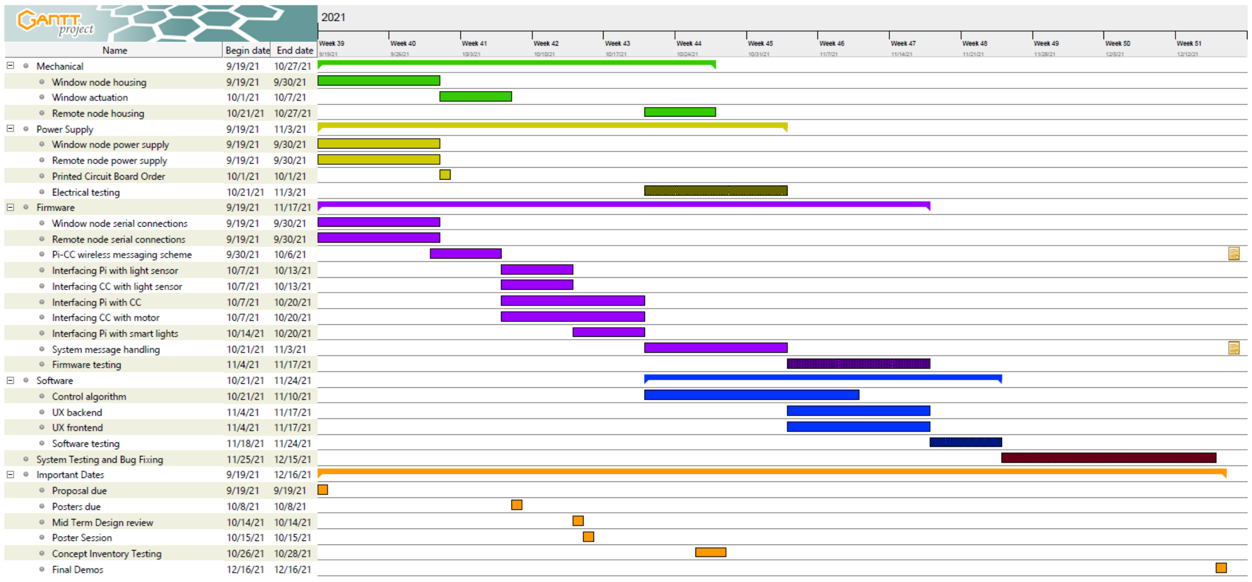


Figure 20: Control Loop

Project Time Line

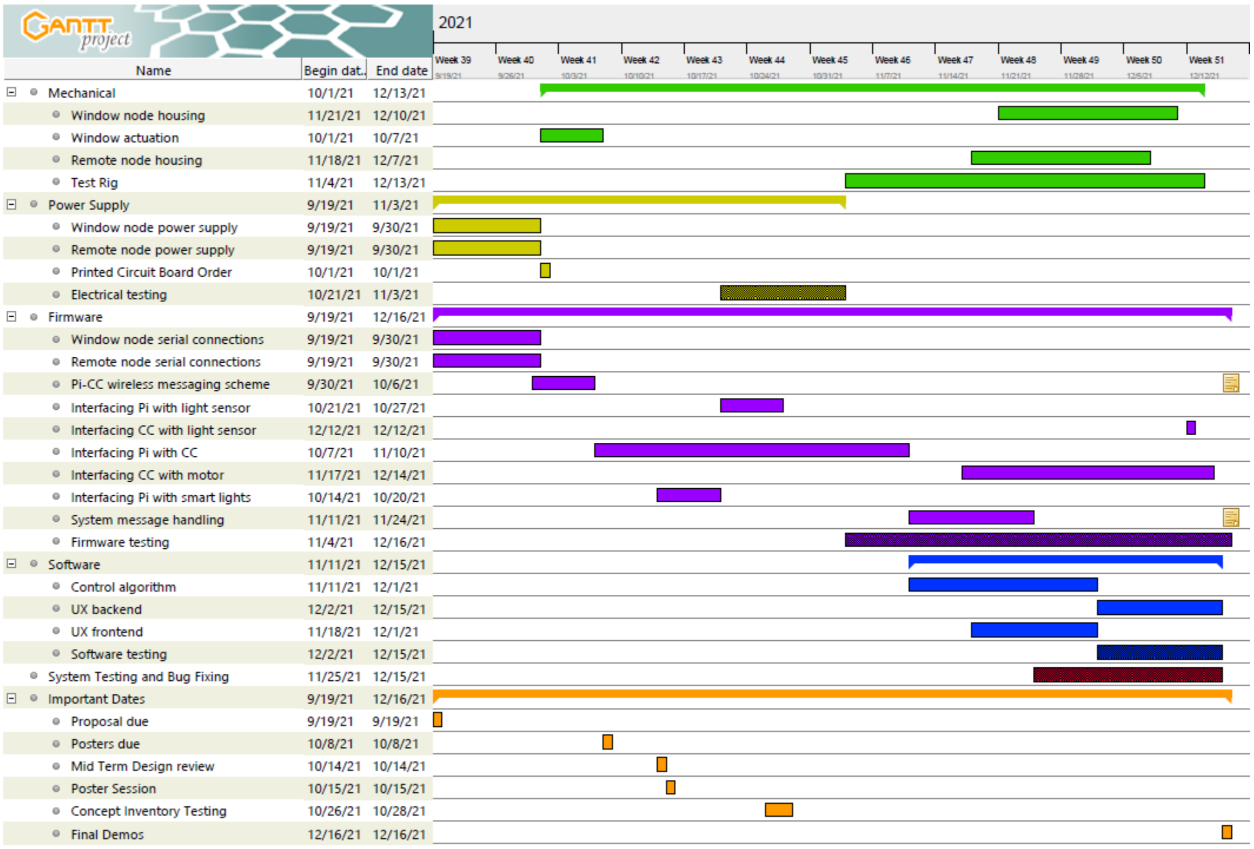
There were stark differences between our original planned timeline, shown in Figure 21: Gantt Chart created as part of the capstone proposal in September 2021, and the actual development of the project, shown in Figure 22: Gantt Chart created in December 2021 that reflects progress made throughout the semester. There were three main factors that influenced these differences. The first was parts ordering and supply chain issues due to Covid-19. The pandemic crippled the supply chain for many electronic devices. This not only limited part selection, but also meant severely increased shipping times for many parts. Our firmware development was most impacted by this, as for several devices we were unable to make significant progress without having the hardware to interface with. Our mechanical development was also impeded by supply chain issues and was further hindered by the time it took to print 3D parts. The second main factor was that our project is designed to control light in a room; however, we did not have easy access to an empty room with a single window, and the amount of sunlight is not a reasonably controlled factor for us to use in rigorous testing. Thus, we needed to create an intensive testing rig that required substantially more mechanical development than we originally anticipated. The final main factor was that no member of the group had experience with communication over Wi-Fi before working on this project. While the SimpleLink SDK[61] used was extremely useful, we found that using it in a meaningful way as applied to our project came with a significant learning curve. Interfacing the Raspberry Pi[7] with the CC3220SF-LAUNCHXL[6, p. 32] took more than a month—but the majority of that time was mostly spent learning how to use the toolchain and provided drivers. Our failure to anticipate this learning curve severely impacted the discrepancies between our projections and the reality of firmware development.

Fortunately, despite these setbacks, we were able to still meet benchmarks (notably the Midterm Design Review and the Final Demonstration). The Final Demonstration is in two days, and we are on track to have the system functionality we originally proposed at the beginning of the semester.



Name	Begin date	End date
Mechanical	9/19/21	10/27/21
Window node housing	9/19/21	9/30/21
Window actuation	10/1/21	10/7/21
Remote node housing	10/21/21	10/27/21
Power Supply	9/19/21	11/3/21
Window node power supply	9/19/21	9/30/21
Remote node power supply	9/19/21	9/30/21
Printed Circuit Board Order	10/1/21	10/1/21
Electrical testing	10/21/21	11/3/21
Firmware	9/19/21	11/17/21
Window node serial connections	9/19/21	9/30/21
Remote node serial connections	9/19/21	9/30/21
Pi-CC wireless messaging scheme	9/30/21	10/6/21
Interfacing Pi with light sensor	10/7/21	10/13/21
Interfacing CC with light sensor	10/7/21	10/13/21
Interfacing Pi with CC	10/7/21	10/20/21
Interfacing CC with motor	10/7/21	10/20/21
Interfacing Pi with smart lights	10/14/21	10/20/21
System message handling	10/21/21	11/3/21
Firmware testing	11/4/21	11/17/21
Software	10/21/21	11/24/21
Control algorithm	10/21/21	11/10/21
UX backend	11/4/21	11/17/21
UX frontend	11/4/21	11/17/21
Software testing	11/18/21	11/24/21
System Testing and Bug Fixing	11/25/21	12/15/21
Important Dates	9/19/21	12/16/21
Proposal due	9/19/21	9/19/21
Posters due	10/8/21	10/8/21
Mid Term Design review	10/14/21	10/14/21
Poster Session	10/15/21	10/15/21
Concept Inventory Testing	10/26/21	10/28/21
Final Demos	12/16/21	12/16/21

Figure 21: Gantt Chart created as part of the capstone proposal in September 2021



Name	Begin dat..	End date
Mechanical	10/1/21	12/13/21
Window node housing	11/21/21	12/10/21
Window actuation	10/1/21	10/7/21
Remote node housing	11/18/21	12/7/21
Test Rig	11/4/21	12/13/21
Power Supply	9/19/21	11/3/21
Window node power supply	9/19/21	9/30/21
Remote node power supply	9/19/21	9/30/21
Printed Circuit Board Order	10/1/21	10/1/21
Electrical testing	10/21/21	11/3/21
Firmware	9/19/21	12/16/21
Window node serial connections	9/19/21	9/30/21
Remote node serial connections	9/19/21	9/30/21
Pi-CC wireless messaging scheme	9/30/21	10/6/21
Interfacing Pi with light sensor	10/21/21	10/27/21
Interfacing CC with light sensor	12/12/21	12/12/21
Interfacing Pi with CC	10/7/21	11/10/21
Interfacing CC with motor	11/17/21	12/14/21
Interfacing Pi with smart lights	10/14/21	10/20/21
System message handling	11/11/21	11/24/21
Firmware testing	11/4/21	12/16/21
Software	11/11/21	12/15/21
Control algorithm	11/11/21	12/1/21
UX backend	12/2/21	12/15/21
UX frontend	11/18/21	12/1/21
Software testing	12/2/21	12/15/21
System Testing and Bug Fixing	11/25/21	12/15/21
Important Dates	9/19/21	12/16/21
Proposal due	9/19/21	9/19/21
Posters due	10/8/21	10/8/21
Mid Term Design review	10/14/21	10/14/21
Poster Session	10/15/21	10/15/21
Concept Inventory Testing	10/26/21	10/28/21
Final Demos	12/16/21	12/16/21

Figure 22: Gantt Chart created in December 2021 that reflects progress made throughout the semester

Despite the timeline setbacks outlined above, the high parallelization of tasks meant that we were still able to make progress and meet deadlines even when bottlenecked at certain points. Some specific examples of this include the development of the power supply and electric connections alongside the firmware and first mechanical tasks. Later in the semester, after the electrical testing was finished, the group was simultaneously working on mechanical, firmware, and software problems. The specific division of tasks is summarized as follows:

Peter primarily worked on the Printed Circuit Board design and layout, and secondarily worked on testing the power supply.

Mason primarily worked on the control algorithm, and secondarily worked on some device drivers and the remote node backend codebase.

Steven primarily worked on mechanical design and front end of the website, and secondarily worked with electrical connection testing.

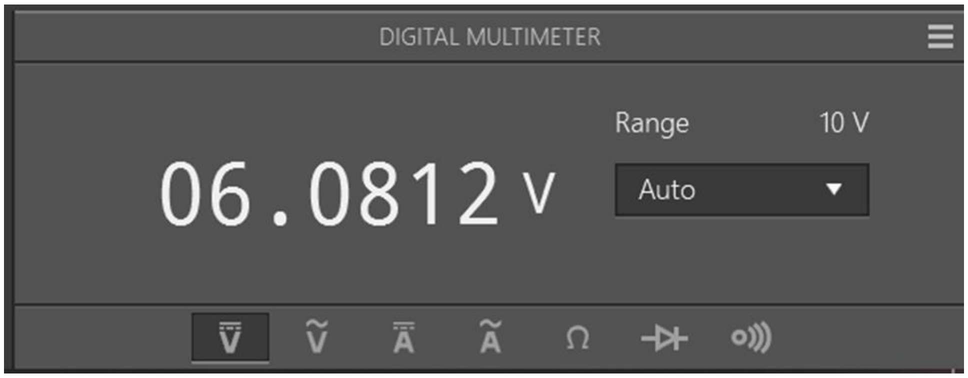
Alex primarily worked on device drivers and the Wi-Fi connection between two nodes, and secondarily helped with software testing.

Test Plan

From the Midterm Review, our test plan was split into hardware and software testing. The hardware testing was split between the window and remote nodes. The software testing was split between light sensor readings, servo actuation, communication, and user interface.

Hardware

The first step in the hardware test plan was to ensure that power was being received by all the components. For the window node, the team needed to ensure that the 6V supply from the wall was being received by the servomotor and at the input to the voltage regulator. From the voltage regulator, the output needed to be 3.3V which would be used to power the light sensor and microcontroller. The team used probes on a virtual bench in the NI lab to take the measurements. Figure 23: 6V Power Supply Verification shows the measurement for the 6V node. The measurement is slightly above the nominal output, but the percent error is 1.4%. Figure 24: 3.3V Power Supply Verification shows the measurement for the 3.3V node. The measurement is slightly below the nominal value, but the percent error is less than a single percentage point.

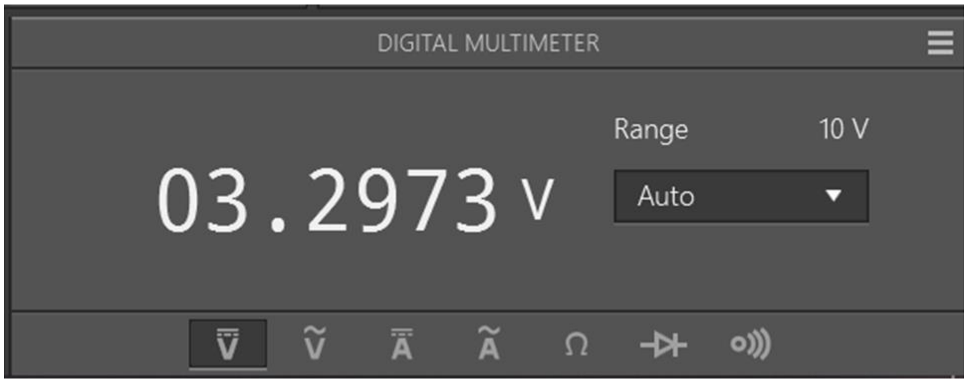

DIGITAL MULTIMETER

Range 10 V

06.0812 V

Auto

Figure 23: 6V Power Supply Verification


DIGITAL MULTIMETER

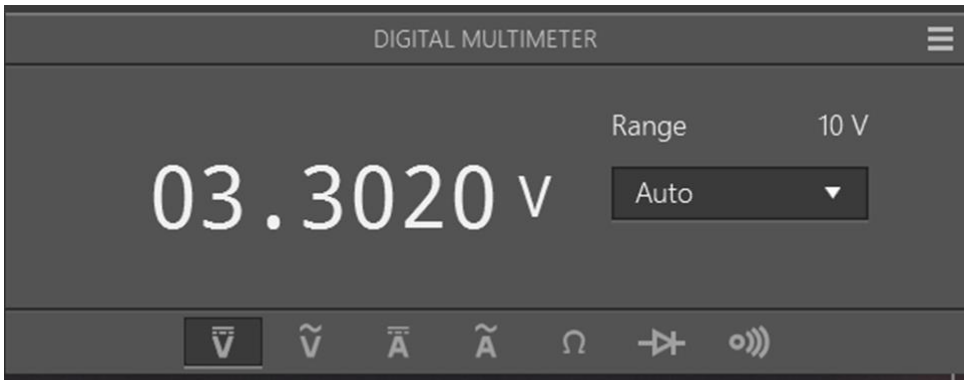
03.2973 V

Range 10 V

Auto

Figure 24: 3.3V Power Supply Verification

For the remote node, a similar procedure was followed. Power was supplied directly to the PI, so the only node that needed to be tested was the 3.3V output pin which supplied power to the light sensor. The results from the measurement are shown in Figure 25: 3.3V Power Supply Verification (Remote)**Error! Reference source not found.**. The output is slightly above the nominal 3.3V, but the percent error is again less than a single percentage point.


DIGITAL MULTIMETER

03.3020 V

Range 10 V

Auto

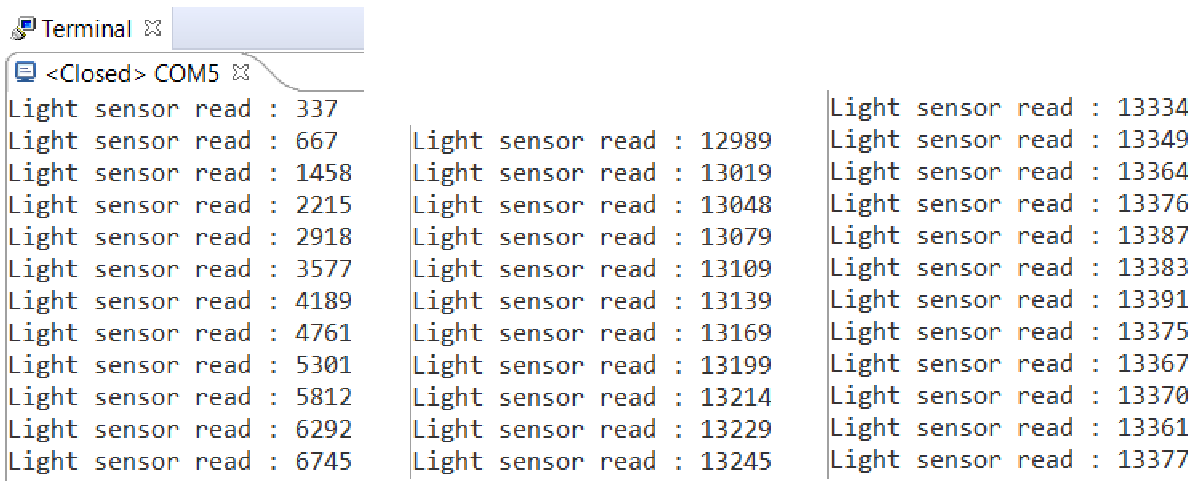
Figure 25: 3.3V Power Supply Verification (Remote)

Communication

Our communication testing lasted a significant amount of time. It began with getting each of the remote node (Raspberry Pi[7]) and the window node (CC3220SF-LAUNCHXL[6, p. 3]) configured to connect to a Wi-Fi network. After that, we began to test the TCP socket connection between the two devices. These tests simply involved using print statements to get visual confirmation that the two devices were connected. Once we had established that the TCP socket connection was valid, we tested messaging by sending dummy data back and forth between the devices. We do not include screenshots of these data captures because in tests below, we successfully send real data from devices back and forth, which validates both those device drivers but also the wireless communication between the nodes.

Device Drivers

In order to test our light sensor driver and filter, we took screen captures (Figure 26: Filtered Light Sensor Readings) of light sensor data and ensured that the values read matched our



```
Light sensor read : 337
Light sensor read : 667
Light sensor read : 1458
Light sensor read : 2215
Light sensor read : 2918
Light sensor read : 3577
Light sensor read : 4189
Light sensor read : 4761
Light sensor read : 5301
Light sensor read : 5812
Light sensor read : 6292
Light sensor read : 6745
```

```
Light sensor read : 12989
Light sensor read : 13019
Light sensor read : 13048
Light sensor read : 13079
Light sensor read : 13109
Light sensor read : 13139
Light sensor read : 13169
Light sensor read : 13199
Light sensor read : 13214
Light sensor read : 13229
Light sensor read : 13245
```

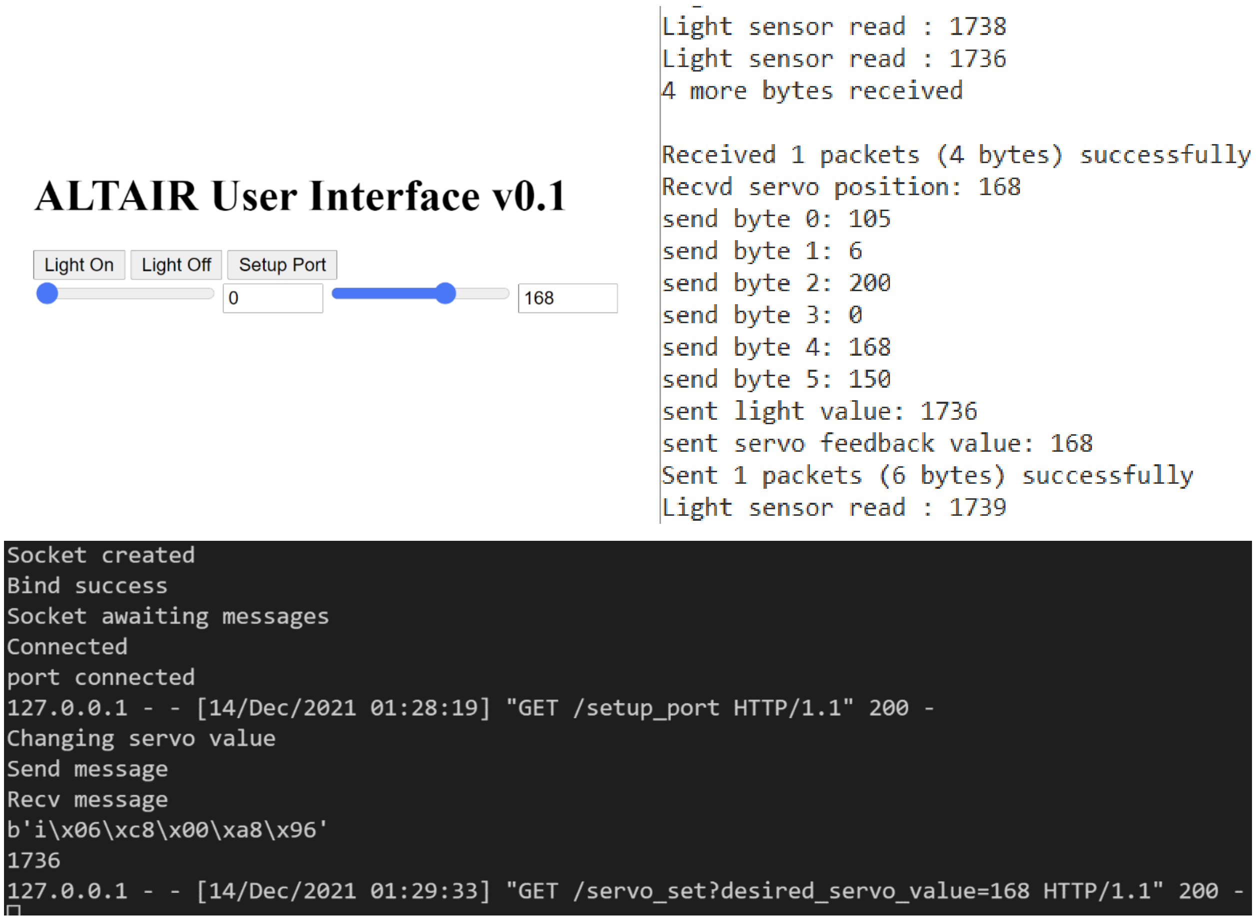
```
Light sensor read : 13334
Light sensor read : 13349
Light sensor read : 13364
Light sensor read : 13376
Light sensor read : 13387
Light sensor read : 13383
Light sensor read : 13391
Light sensor read : 13375
Light sensor read : 13367
Light sensor read : 13370
Light sensor read : 13361
Light sensor read : 13377
```

expectations under set conditions.

Figure 26: Filtered Light Sensor Readings

The leftmost capture in Figure 26: Filtered Light Sensor Readings shows the light sensor capture immediately after it is uncovered and exposed to the ambient light in the room. The middle capture shows a capture closer to the end of the light sensor filter stabilization, and the rightmost capture shows the filtered light sensor data stabilizing. We described in the previous section our method of filtering this data: we want to show that the stored value is equal to (current value)*(15/16) + (newly read value)*(1/16). From the rightmost capture, we know that in this environment the new light value being read was around 13370, since this is what the filtered value began to stabilize around. On the leftmost capture, we will disregard the first data read since the light sensor had just been uncovered and we do not have a good estimate of what light value it read in as an input. Thus, we can look at the jump of the light value to 667 to 1458. Notice that (15/16)*667 is 625, and that 13370/16 = 835. The sum of these (625+835) is equal to 1460, which is extremely close to the next stored light value. In the middle capture, we can see that the jumps between stored data are much smaller than they are on the left, since they are closer to the value being read in so the transitions are less sharp. If we look at the jump from 12989 to 13019, we see (12989*(15/16) + (13370/16)) = 12177 + 835 = 13012; a response that again is very close to what we expect. This test both validates that we are properly reading the light sensor and that our digital filtering method is working correctly.

To test our servo driver and to demonstrate the system’s wireless messaging capabilities, we set the servo position from the remote node and then sent the updated position and the current filtered light value back to the pi. This sequence is shown in Figure 27: Servo and Messaging Test Results. On the top left, the user interface is shown to have set the desired value of the servo to 168. The bottom of the figure (the remote node terminal) says that a message has been sent to the window node. The window node terminal (top right) shows that it received a message, and specifically received the new servo position of 168 as sent by the user interface. The servo was then rotated to that position by the microcontroller’s pulse width modulation module, and then the window node put together the 6-byte packet shown in its terminal. That packet was sent to the remote node, whose terminal shows that packet being received, its bytes, and the light sensor value it parsed from that packet. Notice that the light sensor value is 1736, which is the last light sensor value printed to the window node terminal. Thus, this sequence of events captured in the screen snip shows that the user interface correctly delivered a desired value to the servo driver, that the 16 bit light value is properly packetized into the buffer, that the window node sends a packet that is received by the remote node, and that the remote node successfully parses the packet to get the correct light value. Events not captured by a screen snip demonstrated that the servo driver works correctly (namely, that the servo responded correctly to the value of 168 passed to it by the user interface over Wi-Fi).



```
Light sensor read : 1738
Light sensor read : 1736
4 more bytes received

Received 1 packets (4 bytes) successfully
Recvd servo position: 168
send byte 0: 105
send byte 1: 6
send byte 2: 200
send byte 3: 0
send byte 4: 168
send byte 5: 150
sent light value: 1736
sent servo feedback value: 168
Sent 1 packets (6 bytes) successfully
Light sensor read : 1739
```

```
Socket created
Bind success
Socket awaiting messages
Connected
port connected
127.0.0.1 - - [14/Dec/2021 01:28:19] "GET /setup_port HTTP/1.1" 200 -
Changing servo value
Send message
Recv message
b'i\x06\xc8\x00\xa8\x96'
1736
127.0.0.1 - - [14/Dec/2021 01:29:33] "GET /servo_set?desired_servo_value=168 HTTP/1.1" 200 -
```

Figure 27: Servo and Messaging Test Results

Software

No data was collected for software testing, only visual responses from the light controlling system that matched behavior we expected. The software was tested on a room model that can be seen in Figure 28. The frame was built using 80/20 aluminum T-slots and personally manufactured brackets which are shown in the appendix in Figure 31.

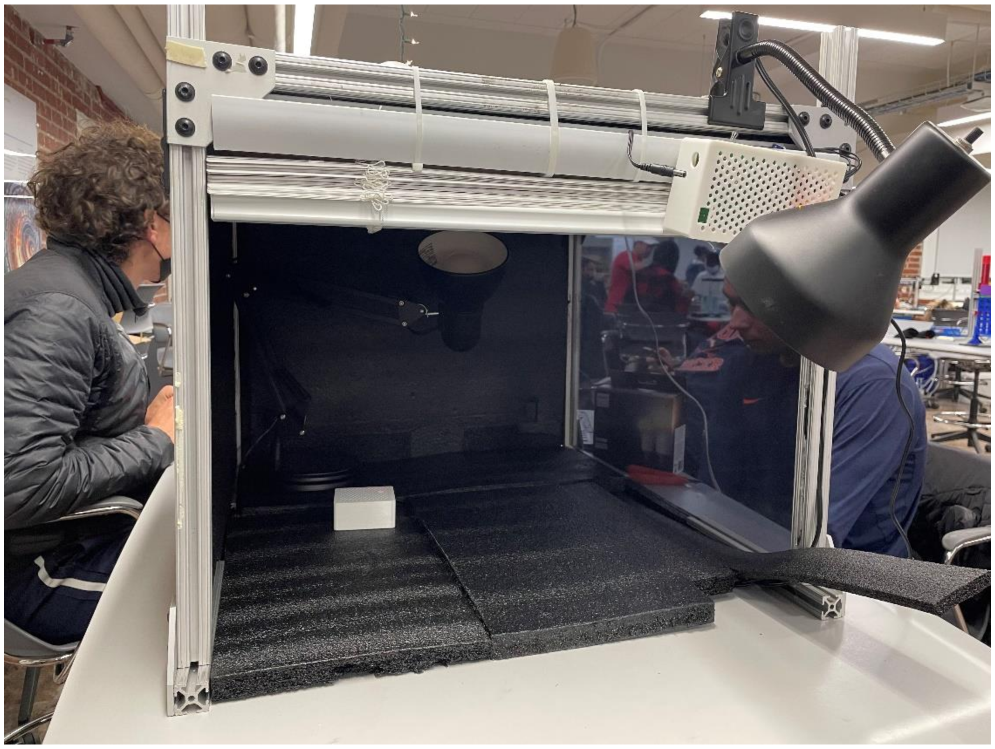


Figure 28: Model Room for Testing

Final Results

Table 33: Proposal expectations

Points	Window Node	Remote Node	User Interface
3	Can receive light sensor data from the Raspberry Pi **AND** control the blinds **AND** smart lights to achieve the desired light level	Can properly read the light sensor **AND** transmit the data to the microcontroller over wirelessly	User can easily navigate the interface **AND** successfully control the desired light level **AND** have that data transmitted to the microcontroller
2	Can receive light sensor data from the Raspberry Pi **AND** control the blinds **OR** adjust the lights to be close to the desired light level	Can properly read the light sensor **BUT CANNOT** transmit the data to the Raspberry Pi wirelessly	User **CANNOT** easily navigate the interface **BUT CAN** successfully control the desired light level **AND** have that data transmitted to the microcontroller
1	Cannot receive light sensor data from the Raspberry Pi **BUT CAN** adjust either the blinds **OR** lights	Cannot properly read the light sensor output **BUT CAN** transmit data to the Raspberry Pi wirelessly	User can easily navigate the interface **BUT CANNOT** control the desired light level **AND** no data is transmitted to the microcontroller

0	Cannot receive information from the Raspberry Pi **AND CANNOT** control the blinds **NOR** the lights	Cannot properly read the light sensor **AND CANNOT** transmit data to the Raspberry Pi wirelessly	User CANNOT easily navigate the interface **AND CANNOT** control the desired light level **AND** no data is transmitted to the microcontroller

Above in Table 33 are shown the expectations put forth for this project for our proposal. We have successfully completed each of the level 3 rubric items. Thus, our project meets all our expectations. Currently, the control algorithm is simplistic, but we expect to make modifications to improve performance over the next two days before our final demonstration.To test our servo driver and to demonstrate the system's wireless messaging capabilities, we set the servo position from the remote node and then sent the updated position and the current filtered light value back to the pi. This sequence is shown in Figure 27: Servo and Messaging Test Results. On the top left, the user interface is shown to have set the desired value of the servo to 168. The bottom of the figure (the remote node terminal) says that a message has been sent to the window node. The window node terminal (top right) shows that it received a message, and specifically received the new servo position of 168 as sent by the user interface. The servo was then rotated to that position by the microcontroller's pulse width modulation module, and then the window node put together the 6-byte packet shown in its terminal. That packet was sent to the remote node, whose terminal shows that packet being received, its bytes, and the light sensor value it parsed from that packet. Notice that the light sensor value is 1736, which is the last light sensor value printed to the window node terminal. Thus, this sequence of events captured in the screen snip shows that the user interface correctly delivered a desired value to the servo driver, that the 16 bit light value is properly packetized into the buffer, that the window node sends a packet that is received by the remote node, and that the remote node successfully parses the packet to get the correct light value. Events not captured by a screen snip demonstrated that the servo driver works correctly (namely, that the servo responded correctly to the value of 168 passed to it by the user interface over Wi-Fi).

Figure 27: Servo and Messaging Test Results

Software

No data was collected for software testing, only visual responses from the light controlling system that matched behavior we expected. The software was tested on a room model that can be seen in Figure 28. The frame was built using 80/20 aluminum T-slots and personally manufactured brackets which are shown in the appendix in Figure 31.

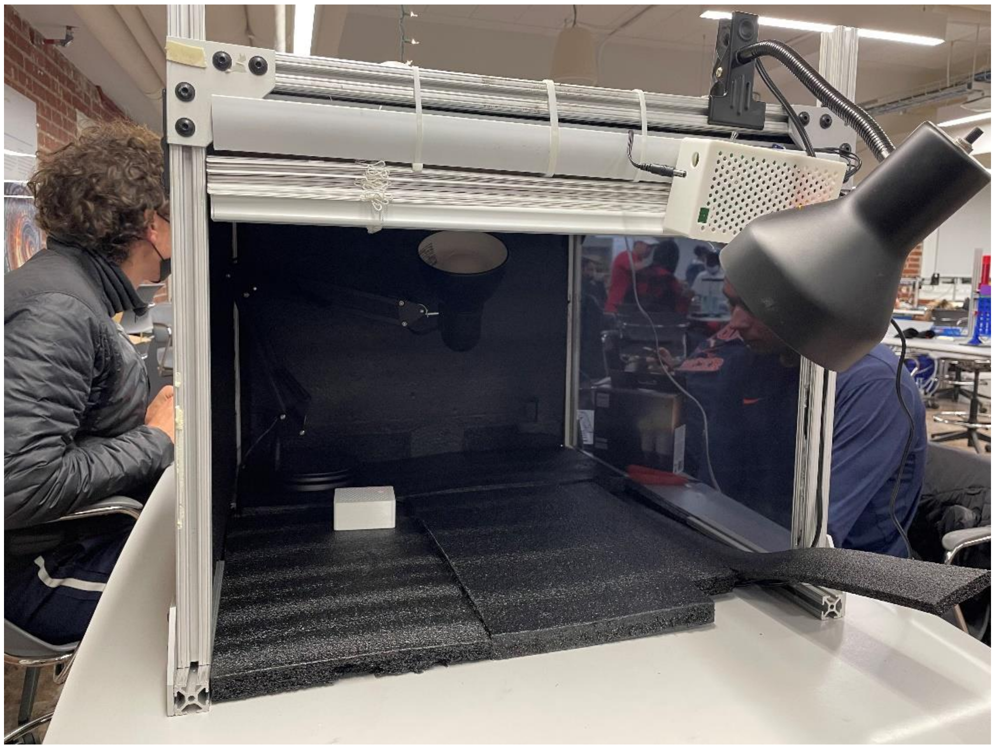


Figure 28: Model Room for Testing

Final Results

Table 33

Costs

The costs for the team's project can be separated into four categories: PCB fabrication, microcontrollers, components, and mechanical/miscellaneous items. The price for PCB fabrication was $33 per board. A single board was made for the window node and the remote node. The total spent on PCB fabrication was $66. The team decided it would be beneficial to buy two TI CC3220s for the project even though only one CC3220 would end up on the window node. Each microcontroller was $59.99 for a total of $119.98. For the remote node, a Raspberry Pi 3A+ was used which costed $25.00. The cost for all the components to populate the boards with was $76.46. The team budgeted to get replacements for each component in the case of issues. The cost of the miscellaneous items was $141.49. A detailed spreadsheet with the cost for all individual items can be found in Figure 29: Complete Bill of Materials in the appendix.

Crude calculations were performed to estimate the cost of producing 10,000 units. Based on the original cost of components, the total cost for 10,000 units would be $229,800. All the components were then looked up on Digikey to see the cost for buying in bulk. The new cost for producing 10,000 units would be $144,947.80; a savings of $84,852.20. In addition to the components, costs could be saved on the microcontroller. The CC3220 meets the design requirements of the project, but many of the peripheral functions are unused, and are therefore an unnecessary cost for the project. If 10,000 units were to be produced, a different microcontroller unit with fewer features could be implemented to save costs. As noted in the technical documentation section, the size of the PCB boards designed was too big. While the cost for single manufacturing was set at a fixed price of $33, it is more likely that the cost for the PCB boards at large scale would be based upon the size. Therefore, making the PCB design more compact would be beneficial. From the FreeDFM report, the cost of making 150 boards for the window node would drop the cost down to $5.29. For 10,000 boards, the savings would be $277,100 compared to buying the same board at $33/board. The cost for making 150 boards for the remote node would drop the cost down to $3.63. For 10,000 boards, the savings would be $293,700. The PCB boards designed are not very complex. The most difficult part to solder onto the board would be the light sensors. However, at high production rates, automation would likely benefit the team's product and reduce costs.

Future Work

From the current state of the project, the team believes there are several expansions which build on and improve the project.

Energy Efficiency Algorithm

An improvement to the current project would create an algorithm to optimize the blinds position for natural light and energy efficiency. According to the Department of Energy, between 25 and 30% of residential energy usage is due to heat gain or loss through windows[21]. This extension would add a temperature sensor and a clock synchronized to the local time. The temperature sensor would provide feedback on how to operate the system. If the temperature recorded was below a desired temperature set by the user, the windows would open to allow sunlight to naturally heat the room. If the temperature was above a desired temperature, the windows would close in order to block sunlight and avoid warming. The clock would be used to during the day and night cycles. In the winters, shades are an insulating barrier. During the day, it would be beneficial to have the shades wide open to allow for natural heating. As day turns to night, the shades should close in order to trap warm air inside and keep cold air out. As night turns into day, the windows should reopen. In the summer, the shades should close during the day to avoid excessive heating and open during the night to release trapped air.

Manual Controls

In the current system, the blinds can only be controlled through the servomotor. It would be beneficial to include a method for the user to operate the blinds manually. Additionally, the user would be able to manually control the smart lights through the user interface.

Retrofittable System

The current design of the project is only applicable to the venetian blinds with slats. The slats can only be rotated in order to block or let in sunlight. The first improvement to the system would be to add a functionality such that the slats could be twisted, but also the raised and lowered. Secondly, the project could be improved by making the nodes applicable to all types of blinds. Roller blinds are an example of another blind that would work in the current system. The servomotor would need to be replaced with a stepper motor that has more degrees of rotational freedom. This would ensure the blinds could be raised and lowered to let in enough light.

Outside of the blinds, it would be convenient if the user could add any brand of smart lights be controlled through Wi-Fi. The current lights were chosen because they came with documentation on how to connect the lights in our Wi-Fi system. The general coding could be abstracted away to include all IoT lights.

Alternative Power Sources

An improve on the current design would be to add other power sources. As both the window node and the remote node are powered by a wall supply, there is only a limited range in which the nodes can be placed. Two supply options are available. The first solution would be to use batteries. For the window node, and assuming the size of the PCB board reduced, the batteries and the entire system could be held in the header of the blinds. For the remote node, a small battery pack to run the light sensor and transceiver would allow a degree of freedom.

A second solution would have a solar powered system. Solar panels would be added to both the window and the remote node. Due to the efficiency of solar panels, it is likely that the system would have to be optimized for the most natural light possible throughout the day. To run at night, the solar powered system would need to have a secondary power source, most likely a battery pack.

Remote Access

The final improvement would be to allow access of the system beyond the local network. The at-home system would connect through to the internet to the user interface to allow for extended range, which can allow users to manually set “vacation times” and turn on some lights at night when they are away, or to close the blinds in case they forget to after leaving the house.

References

[1] M. Paul, "Natural Light in the Office Boosts Health," Aug. 08, 2014. https://news.northwestern.edu/stories/2014/08/natural-light-in-the-office-boosts-health

[2] K. Tenkorang, E. Agyeman, and B. Chan, "Window Automated Natural Daylight Assistant (WANDA)," Mesgana, University of Virginia, Charlottesville, 2020. [Online]. Available: file:///C:/Users/student/Downloads/Tenkorang_Kwadwo_Technical_Report%20(2).pdf

[3] IKEA, "Motorized shades," 2021. https://www.ikea.com/us/en/cat/electric-blinds-44531/

[4] Lutron, "Smart Shade Products," 2021. https://www.serenashades.com/products/#smart-wood-blinds

[5] Graber, "Z-Wave Smart Shades," 2021. https://www.graberblinds.com/why-graber/z-wave-motor/

[6] "CC3220SF-LAUNCHXL Development kit | TI.com." https://www.ti.com/tool/CC3220SF-LAUNCHXL?keyMatch=&tisearch=search-everything&usecase=hardware (accessed Dec. 12, 2021).

[7] R. P. Ltd, "Buy a Raspberry Pi 3 Model A+," *Raspberry Pi*. https://www.raspberrypi.com/products/raspberry-pi-3-model-a-plus/ (accessed Dec. 12, 2021).

[8] "slaa542.pdf." Accessed: Dec. 12, 2021. [Online]. Available: https://www.ti.com/lit/an/slaa542/slaa542.pdf?ts=1639360216108&ref_url=https%253A%252F%252Fwww.ecosia.org%252F

[9] "Multisim Download." https://www.ni.com/en-us/support/downloads/software-products/download.multisim.html (accessed Dec. 12, 2021).

[10] "Ultiboard." https://www.ni.com/en-us/shop/software/products/ultiboard.html (accessed Dec. 12, 2021).

[11] M. Group, "Electronic Component Shortages Update -- 2022 and Beyond." https://blog.matric.com/electronic-component-shortages-update (accessed Dec. 13, 2021).

[12] C. Correll, "Price Hikes and Shortages: Production Uncertainty for Electronic Parts in 2021," Jun. 16, 2021. https://www.z2data.com/insights/electronic-component-shortage-price-hike-2021 (accessed Sep. 19, 2021).

[13] "DigiKey Electronics - Electronic Components Distributor." https://www.digikey.com/ (accessed Dec. 13, 2021).

[14] "Electronic Components Distributor - Mouser Electronics." https://www.mouser.com/ (accessed Dec. 13, 2021).

[15] "FreeDFM - A Service of Advanced Circuits." https://www.my4pcb.com/net35/FreeDFMNet/FreeDFMHome.aspx (accessed Dec. 13, 2021).

[16] "China PCB Prototype and Mass-fabrication Quote Online." http://www.digpcb.com/ (accessed Dec. 14, 2021).

[17] J. LaDou, "Printed circuit board industry," *Int. J. Hyg. Environ. Health*, vol. 209, no. 3, pp. 211–219, May 2006, doi: 10.1016/j.ijheh.2006.02.001.

[18] A. D. Schino, "Environmental Impact of Steel Industry," in *Handbook of Environmental Materials Management*, C. M. Hussain, Ed. Cham: Springer International Publishing, 2018, pp. 1–21. doi: 10.1007/978-3-319-58538-3_101-1.

[19] "SER0046 DFRobot | Motors, Solenoids, Driver Boards/Modules | DigiKey." https://www.digikey.ie/en/products/detail/dfrobot/SER0046/11613083 (accessed Dec. 13, 2021).

[20] “What are the drivers of microplastic toxicity? Comparing the toxicity of plastic chemicals and particles to Daphnia magna | Elsevier Enhanced Reader.” https://reader.elsevier.com/reader/sd/pii/S0269749120360802?token=214551C81C1D304D7EEFC7E6748B82DCFB6DF22B6F83D1D92D35D7FA529A5ED3E5C0D650D3510FDEF4C117BF517717C7&originRegion=us-east-1&originCreation=20211213212232 (accessed Dec. 13, 2021).

[21] “Update or Replace Windows,” *Energy.gov*. https://www.energy.gov/energysaver/update-or-replace-windows (accessed Dec. 14, 2021).

[22] “Ultimaker White PLA Filament - 2.85mm (0.75kg),” *MatterHackers*. https://www.matterhackers.com/store/l/ultimaker-pla-3d-printing-filament-285mm-075kg/sk/MX6CY6J2 (accessed Dec. 14, 2021).

[23] “80/20 T-slot Aluminum Building System.” https://8020.net/ (accessed Dec. 14, 2021).

[24] “The Environmental Impact of Aluminum (And Why it’s Still Better Than Plastic),” *The Student Conservation Association*, Sep. 21, 2017. https://www.thesca.org/connect/blog/environmental-impact-aluminum (accessed Dec. 14, 2021).

[25] “Aluminium production & environmental impact.” http://www.greenspec.co.uk/building-design/aluminium-production-environmental-impact/ (accessed Dec. 14, 2021).

[26] Megan, “Is Fiberglass Biodegradable? (Eco & Health Facts You Should Know),” *Citizen Sustainable*, Jul. 01, 2020. https://citizensustainable.com/fiberglass/ (accessed Dec. 14, 2021).

[27] M. Wacker and M. F. Holick, “Sunlight and Vitamin D,” *Dermatoendocrinol.*, vol. 5, no. 1, pp. 51–108, Jan. 2013, doi: 10.4161/derm.24494.

[28] “Sunburns and Damage to Your Body,” *Healthline*, Jun. 28, 2018. https://www.healthline.com/health-news/heres-how-much-damage-a-really-bad-sunburn-can-do (accessed Dec. 14, 2021).

[29] IPC, “IPC Checklist for Producing Rigid Printed Board Assemblies,” Jun. 2019. [Online]. Available: https://www.ipc.org/media/3418/download

[30] IPC, “IPC 2221: Generic Standard on Printed Board Design,” IL, 1998. [Online]. Available: http://us.jetpcb.com/Cht/Document/ipc-2221all.pdf

[31] 7PCB, “PCB Trace Width Calculator,” *7PCB*. https://www.7pcb.com/trace-width-calculator.php (accessed Dec. 13, 2021).

[32] NEMA, “NEMA Ratings for Enclosures.” https://www.nemaenclosures.com/enclosure-ratings/nema-rated-enclosures.html (accessed Sep. 10, 2021).

[33] NEMA, “NEMA Enclosure Types,” Virginia, 2005. [Online]. Available: https://www.nema.org/docs/default-source/products-document-library/nema-enclosure-types.pdf

[34] DHS, “A Guide to Securing Networks for Wi-Fi (IEEE 802.11 Family),” 1, Mar. 2017. [Online]. Available: https://us-cert.cisa.gov/sites/default/files/publications/A_Guide_to_Securing_Networks_for_Wi-Fi.pdf

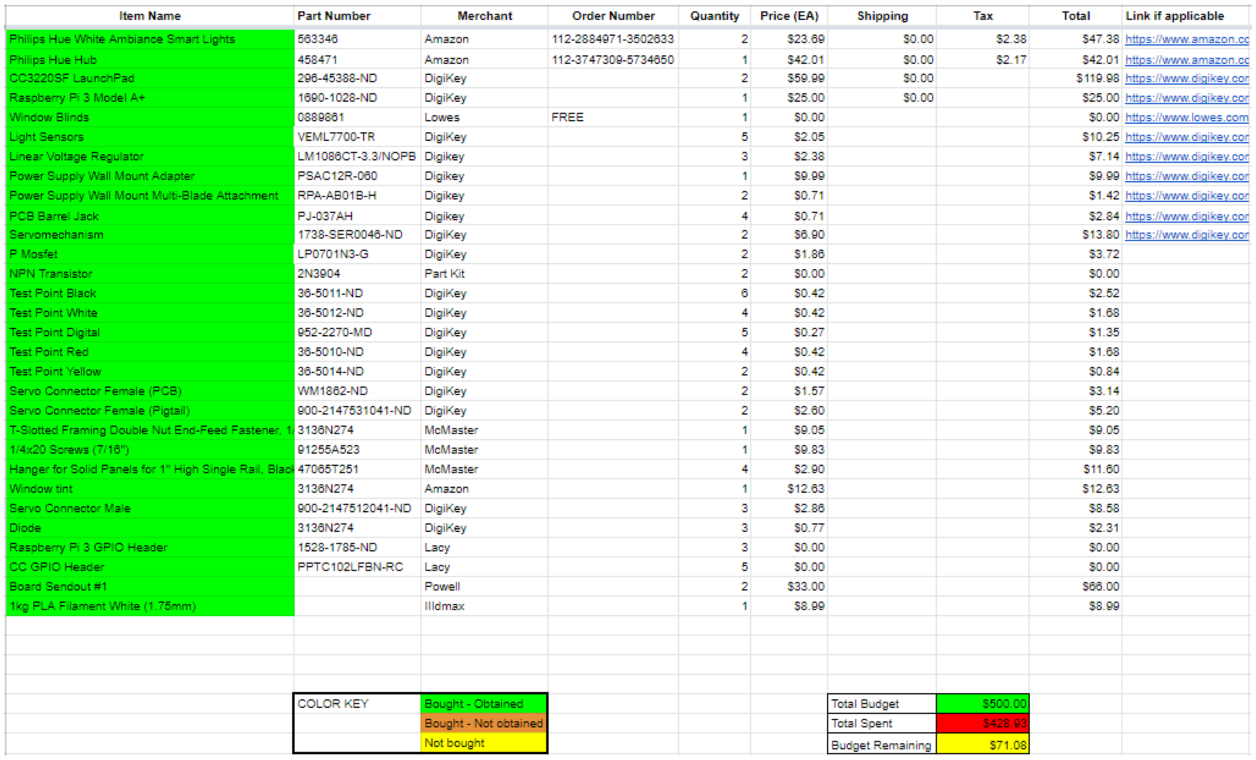
[35] A. Gillis, “IEEE 802 Wireless Standards,” *TechTarget*, Oct. 2020. https://www.techtarget.com/searchnetworking/reference/IEEE-802-Wireless-Standards-Fast-Reference (accessed Sep. 10, 2021).

[36] M. Barr, *Embedded C Programming Standards*. Germantown, MD: Barr Group, 2018. [Online]. Available: https://barrgroup.com/sites/default/files/barr_c_coding_standard_2018.pdf

[37] M. Media, “Metric T Slot Aluminum Profiles,” *Framing Technology Inc.* https://www.framingtech.com/products/metric-aluminum-profiles/ (accessed Dec. 14, 2021).
[38] “Fusion 360 | 3D CAD, CAM, CAE & PCB Cloud-Based Software | Autodesk.” https://www.autodesk.com/products/fusion-360/overview (accessed Dec. 14, 2021).
[39] “CR-10 Smart 3D Printer.” https://www.creality3dofficial.com/products/cr-10-smart-3d-printer (accessed Dec. 14, 2021).
[40] “Ultimaker Cura: Powerful, easy-to-use 3D printing software,” *ultimaker.com*. https://ultimaker.com/software/ultimaker-cura (accessed Dec. 14, 2021).
[41] “MAXIEM 1515 | Abrasive Water Jet Cutter Machine,” *OMAX Waterjet Cutting Machine Manufacturer*, Feb. 19, 2016. https://www.omax.com/maxiem-waterjet/1515 (accessed Dec. 14, 2021).
[42] “CCSTUDIO IDE, configuration, compiler or debugger | TI.com.” https://www.ti.com/tool/CCSTUDIO (accessed Dec. 13, 2021).
[43] “SIMPLELINK-CC32XX-SDK Software development kit (SDK) | TI.com.” https://www.ti.com/tool/SIMPLELINK-CC32XX-SDK (accessed Dec. 14, 2021).
[44] “VirtualBench.” https://www.ni.com/en-us/shop/hardware/products/virtualbench-all-in-one-instrument.html (accessed Dec. 13, 2021).
[45] “Welcome to Flask — Flask Documentation (2.0.x).” https://flask.palletsprojects.com/en/2.0.x/ (accessed Dec. 14, 2021).
[46] “Welcome to Python.org,” *Python.org*. https://www.python.org/ (accessed Dec. 14, 2021).
[47] “Build software better, together,” *GitHub*. https://github.com (accessed Dec. 14, 2021).
[48] F. A. Millett, M. D. B. Jr, H. J. Byker, and P. H. O. Jr, “Thermochromic window structures,” US8154788B2, Apr. 10, 2012 Accessed: Nov. 23, 2021. [Online]. Available: https://patents.google.com/patent/US8154788B2/en?q=responsive+window+blinds&oq=responsive+window+blinds
[49] Т. А. ЛАШИНА, Д. С. Б. М. ВАН, К. Д. ВАГЕНАР, Б. В. МЕРБЕК, Д. М. Д. ВАН, and Г. Х. А. Й. БРУКСТЕГ, “Способ и устройство для управления освещением в пространстве внутри помещения,” RU2642502C2, Jan. 25, 2018 Accessed: Dec. 12, 2021. [Online]. Available: https://patents.google.com/patent/RU2642502C2/en?q=light+controlled+blinds&oq=light+controlled+blinds
[50] 성욱주, 안병립, 이주윤, and 팽상호, “채광루버형 일사조절장치,” KR102129388B1, Jul. 09, 2020 Accessed: Nov. 30, 2021. [Online]. Available: https://patents.google.com/patent/KR102129388B1/en?q=responsive+window+blinds&oq=responsive+window+blinds
[51] W. J. Mullet, Y. Rodriguez, D. W. Brunk, R. S. Hand, and L. H. Oakley, “Motorizable tilt shade system and method,” US9091115B2, Jul. 28, 2015 Accessed: Dec. 12, 2021. [Online]. Available: https://patents.google.com/patent/US9091115B2/en?q=automatic+window+blinds&oq=automatic+window+blinds
[52] D. Veskovic, “System to control daylight and electric light in a space,” US7566137B2, Jul. 28, 2009 Accessed: Dec. 12, 2021. [Online]. Available: https://patents.google.com/patent/US7566137B2/en?q=light+controlled+blinds&oq=light+controlled+blinds&page=1

[53] “RASPBERRY PI 3 A+ SNAP FIT CASE,” *Fusion360 Gallery*. https://gallery.autodesk.com/fusion360/projects/136671/raspberry-pi-3-aplus-snap-fit-case (accessed Dec. 17, 2021).
[54] “PSAC12R-060 Phihong USA | Power Supplies - External/Internal (Off-Board) | DigiKey.” https://www.digikey.com/en/products/detail/phihong-usa/PSAC12R-060/5418512 (accessed Dec. 13, 2021).
[55] “PJ-037AH CUI Devices | Connectors, Interconnects | DigiKey.” https://www.digikey.com/en/products/detail/cui-devices/PJ-037AH/1644547 (accessed Dec. 13, 2021).
[56] “lm1086.pdf.” Accessed: Dec. 13, 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm1086.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1639445854162
[57] “VEML7700-TR Vishay Semiconductor Opto Division | Sensors, Transducers | DigiKey.” https://www.digikey.com/en/products/detail/vishay-semiconductor-opto-division/VEML7700-TR/5820243 (accessed Dec. 13, 2021).
[58] “ONSMS24978-1.pdf.” https://rocelec.widen.net/view/pdf/0kyoyfnmcq/ONSMS24978-1.pdf?t.download=true&u=5oefqw (accessed Dec. 13, 2021).
[59] “LP0701-P-Channel-Enhancement-Mode-Lateral-MOSFET-Data-Sheet-20005447A.pdf.” Accessed: Dec. 13, 2021. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/LP0701-P-Channel-Enhancement-Mode-Lateral-MOSFET-Data-Sheet-20005447A.pdf
[60] “1N5817-1N5819(DO-41).pdf.” Accessed: Dec. 13, 2021. [Online]. Available: https://www.mccsemi.com/pdf/Products/1N5817-1N5819(DO-41).pdf
[61] TI, “CC3220R, CC3220S, and CC3220SF SimpleLink^{TM} Wi-Fi® Single-Chip Wireless MCU Solutions,” TI, Online, Technical Documentation, May 2021. Accessed: Sep. 19, 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/cc3220sf.pdf?ts=1632026616156&ref_url=https%253A%252F%252Fdev.ti.com%252F
[62] “swau124.pdf.” Accessed: Dec. 13, 2021. [Online]. Available: https://www.ti.com/lit/ug/swau124/swau124.pdf?ts=1639414329807&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FCC3220SF-LAUNCHXL%253FkeyMatch%253D%2526tisearch%253Dsearch-everything%2526usecase%253Dhardware
[63] “React – A JavaScript library for building user interfaces.” https://reactjs.org/ (accessed Dec. 14, 2021).
[64] M. O. contributors Jacob Thornton, and Bootstrap, “Bootstrap.” https://getbootstrap.com/ (accessed Dec. 17, 2021).

Appendix



Item Name	Part Number	Merchant	Order Number	Quantity	Price (EA)	Shipping	Tax	Total	Link if applicable
Philips Hue White Ambiance Smart Lights	563346	Amazon	112-2884971-3502633	2	$23.69	$0.00	$2.38	$47.38	https://www.amazon.co
Philips Hue Hub	458471	Amazon	112-3747309-5734650	1	$42.01	$0.00	$2.17	$42.01	https://www.amazon.co
CC3220SF LaunchPad	296-45388-ND	DigiKey		2	$59.99	$0.00		$119.98	https://www.digikey.cor
Raspberry Pi 3 Model A+	1690-1028-ND	DigiKey		1	$25.00	$0.00		$25.00	https://www.digikey.cor
Window Blinds	0889861	Lowes	FREE	1	$0.00			$0.00	https://www.lowes.com
Light Sensors	VEML7700-TR	DigiKey		5	$2.05			$10.25	https://www.digikey.cor
Linear Voltage Regulator	LM1086CT-3.3/NOPB	Digikey		3	$2.38			$7.14	https://www.digikey.cor
Power Supply Wall Mount Adapter	PSAC12R-060	Digikey		1	$9.99			$9.99	https://www.digikey.cor
Power Supply Wall Mount Multi-Blade Attachment	RPA-AB01B-H	Digikey		2	$0.71			$1.42	https://www.digikey.cor
PCB Barrel Jack	PJ-037AH	Digikey		4	$0.71			$2.84	https://www.digikey.cor
Servomechanism	1738-SER0046-ND	DigiKey		2	$6.90			$13.80	https://www.digikey.cor
P Mosfet	LP0701N3-G	DigiKey		2	$1.86			$3.72	
NPN Transistor	2N3904	Part Kit		2	$0.00			$0.00	
Test Point Black	36-5011-ND	DigiKey		6	$0.42			$2.52	
Test Point White	36-5012-ND	DigiKey		4	$0.42			$1.68	
Test Point Digital	952-2270-MD	DigiKey		5	$0.27			$1.35	
Test Point Red	36-5010-ND	DigiKey		4	$0.42			$1.68	
Test Point Yellow	36-5014-ND	DigiKey		2	$0.42			$0.84	
Servo Connector Female (PCB)	WM1862-ND	DigiKey		2	$1.57			$3.14	
Servo Connector Female (Pigtail)	900-2147531041-ND	DigiKey		2	$2.60			$5.20	
T-Slotted Framing Double Nut End-Feed Fastener, 1/	3136N274	McMaster		1	$9.05			$9.05	
1/4x20 Screws (7/16")	91255A523	McMaster		1	$9.83			$9.83	
Hanger for Solid Panels for 1" High Single Rail, Black	47065T251	McMaster		4	$2.90			$11.60	
Window tint	3136N274	Amazon		1	$12.63			$12.63	
Servo Connector Male	900-2147512041-ND	DigiKey		3	$2.86			$8.58	
Diode	3136N274	DigiKey		3	$0.77			$2.31	
Raspberry Pi 3 GPIO Header	1528-1785-ND	Lacy		3	$0.00			$0.00	
CC GPIO Header	PPTC102LFBN-RC	Lacy		5	$0.00			$0.00	
Board Sendout #1		Powell		2	$33.00			$66.00	
1kg PLA Filament White (1.75mm)		Illdmax		1	$8.99			$8.99	
	COLOR KEY	🟩 Bought - Obtained				Total Budget	$500.00		
		🟧 Bought - Not obtained				Total Spent	$428.93		
		🟨 Not bought				Budget Remaining	$71.08		

Figure 29: Complete Bill of Materials

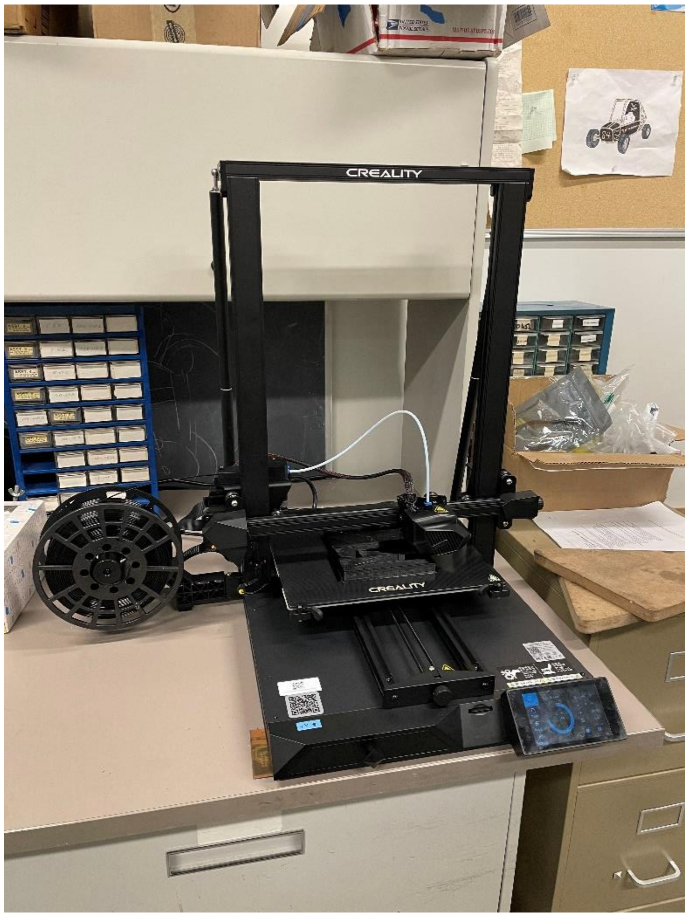


Figure 30: 3D Printer

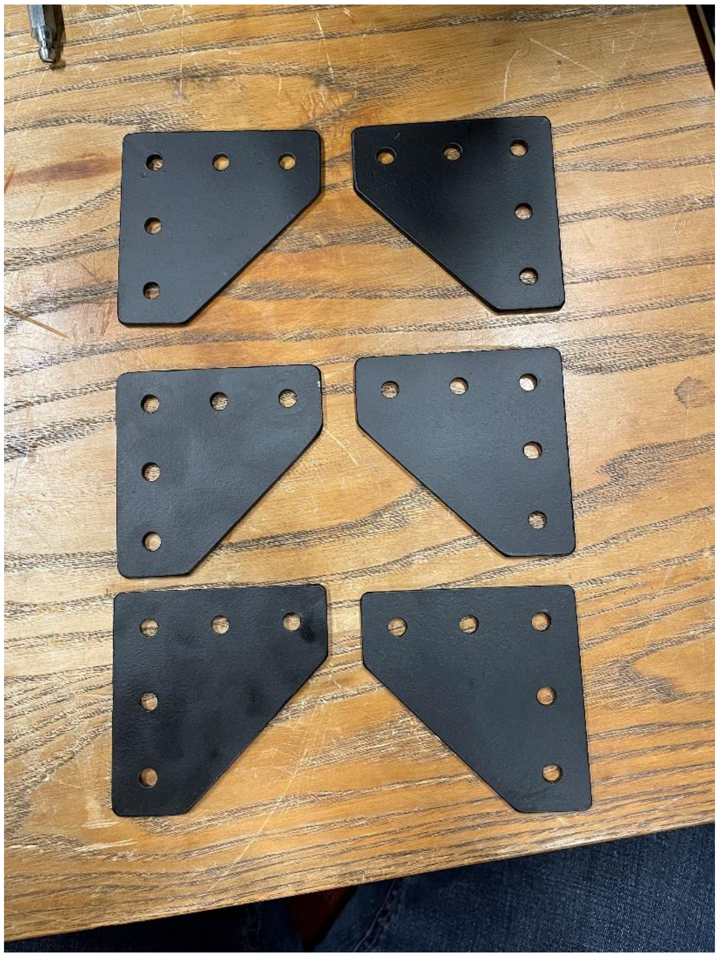


Figure 31: Manufactured Brackets