B.L.I.N.D.S

Blocking Light IN Domestic Spaces

Jeff Chen, Elizabeth Chuei, Dianne Ge

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*—Sunlight often serves as an annoyance to people by obstructing their vision, and adversely affecting their visual comfort, eye health, and productivity. However, completely eliminating natural light has been shown to negatively impact a person's mood and mental health. To combat this issue, our project aims to create a system that will eliminate the manual process of turning blinds, and optimize a room's lighting. The system will automatically close certain blinds if it detects sunlight hitting a person in the room.**

***Index Terms*—LIDAR, Smart Blinds, Computer Vision, OpenCV, Arduino, Magnetometer, Photoresistor, Suncalc, GeoPy**

I. Introduction

Previous research shows that direct exposure to sunlight affects people negatively in a multitude of ways, including obstructing their vision and negatively impacting their eyesight [1]. Overexposure to the sun's harmful UV rays can also cause a variety of eye diseases [2]. However, completely removing sunlight is not an optimal solution, because eliminating natural light can prove detrimental to a person's mood, mental health, and productivity [3]. In other words, to maximize quality of life, one must continuously adjust a room's blinds in a way that minimizes the amount of light that hits one's face, but maximizes the amount of natural light in the room. However, continuously adjusting blinds can disrupt concentration and in turn, decrease one's productivity. To combat this, we aim to create blinds, for people who spend a lot of time at home, that will adjust automatically in a way that prevents light from hitting a person's face, but lets in maximal sunlight. This type of product is particularly relevant, as the number of people who work remotely and spend copious amounts of time at home has skyrocketed since the COVID-19 pandemic [4].

Although other motorized blinds exist, most require the use of a remote control that the user has to press to adjust the blinds. However, pressing a remote control also disrupts concentration, which conflicts with its original purpose of providing convenience and allowing the user to focus on other tasks. Other competing technologies include blinds that move according to the time the sun rises and sets. However, this type of system fails to take into account the position of the user in the room, and whether the light hits them at that location. Our proposed solution seeks to increase a person's quality of life by balancing the amount of sunlight they are exposed to, while solving the existing issues of alternate solutions.

II. Use-Case Requirements

We want our system to be beneficial enough so that a user may be able to focus on the task at hand without being hindered by sunlight. In order to quantify this, our group ran a short survey on a group of college students to ask what the minimum accuracy rate they would be able to tolerate for purchasing a system like this would be. The average came out to 90%. Given this answer, we aim to have our overall system be able to block sunlight from a person's face, by adjusting the blinds so that the light projected from the window into the room will only reach a person's face 90% of the time.

On the other hand, we do not want our blinds to be a source of distraction either. Many widely used face detection systems take under 1 second to detect a face, which is quite fast [5]. However, a constantly moving blinds system may be more of a nuisance than a source of convenience. It may also potentially frighten users and create an unpleasant experience. In the same survey as mentioned above, we asked participants whether they would prefer a constantly moving blinds system, or one that would wait for a set amount of time for a user to stop moving before adjusting the blinds. Additionally, if a participant answered that they preferred the system to wait, we asked how long of a period the system should standby before making the adjustment. The results showed that the participants would be startled if the blinds were constantly moving, and on average preferred a wait time of 10 seconds. Therefore, we want the feedback latency, which is the time it will take for our blinds to react after a user's movement, to be around 10 seconds.

We also want our physical latency (the time it takes for the blinds to physically adjust upwards or downwards) to be on par with existing motorized blinds. Popular motorized blinds on the market take around 60 seconds to fully roll upwards or downwards [6]. Therefore, our motorized blinds system should have at most a 60 second physical latency in order to compete with similar existing products.

Finally, we want to be able to achieve our accuracy goal within a reasonable workspace: the average bedroom size in the US, which is 132 ft^2 [7]. This means we want to be able to block sunlight from a user's face within our accuracy goal in a 132 ft^2 space.

Although we have gathered requirements pertaining to minimizing sunlight that directly hits the user's face, we currently lack knowledge about user satisfaction regarding maximizing sunlight in the room. Moving forward, we will conduct a survey in the future on students at Carnegie Mellon University about how far light should be from their face in order to maximize light in the room.

III. Architecture and/or Principle of Operation

Our full system consists of 3 primary interconnected systems, each with separate subsystems:

1. Sensor System
2. Processing Unit (RPi)
    a. User Position Algorithm
    b. Sun Calculation API

c. Light Area of Effect Algorithm (LAOE)

3. Motor Control

The sensor system (1) consists of 3 “sensors”, each for different purposes. The first sensor is the Intel RealSense Depth Camera D455. This integrated depth and camera system will provide information to determine the 3D coordinates of where a person is located within the room in question. Another sensor is the sunlight detection circuit, which uses photoresistors to determine if the weather is sunny enough to warrant adjusting the blinds. The last sensor is a magnetometer, which will tell the system the cardinal direction of the window, and thus help us determine the azimuth of the sun in respect to the window.

The Processing Unit, the RPi, (2) will be where the necessary calculations are done on the software side. One of the software systems is a user position calculation algorithm. This employs OpenCV and depth information from the LIDAR to calculate the 3D coordinates of a user in the room space. Another component of the software system is the sun calculation API. This function will use the location of the window (latitude, longitude) in addition to the time of day to find the azimuth and altitude of the sun relative to the blinds system. Finally, the last algorithm we will be writing is the LAOE algorithm. Using the measurements of the window, the user position, and the sun data, this algorithm will determine an adjustment so that sunlight is not in the user’s face, as well as maximizing the amount of sunlight entering the room.

The motor control, or the physical blinds system (3), will consist of a few working parts. This consists of an Arduino Uno, which takes in adjustments as inputs, and the motor system attached to the controls of the blind, which will receive the adjustments from the Arduino. The motor system will be attached through a 3-D printed gear that will hold tightly onto the beads of the blinds’ cord lock.

The relationship between these parts and a greater breakdown of each subsystem are illustrated and can be better seen in Appendix C at the end of this document.

IV. Design Requirements

We want our sensor subsystem to contain parts that are at least 90% accurate, to match with our overall accuracy requirements. To limit the impact of polling rate on feedback latency, we want our sensor system to poll for real-world information as much as possible. However, because feedback latency is mainly limited by processing unit limitations, which will take at least a second, it’s best to wait for the processing unit to perform its calculations before polling again. So we want our system to poll for information at least once per second.

As for the photoresistor circuit in this system, the threshold that determines whether light is considered sunlight, which determines whether the blinds move or not, will be determined in the future with thorough testing with real sunlight. Because this threshold is dependent on a variety of factors, including the type of photoresistor used, it has not yet been determined.

Our motor control system needs to be able to fully open/close the blinds in 60 seconds. Because the standard window height is 60 inches, the motors in our system need to be able to rotate at a fast enough speed to allow the blinds to move at a rate of 1 inch per second. The motor system must also be able to allow for precise position control by increments of 1 cm with a margin of 0.5cm. We currently believe that a 1 cm precision is precise enough to cut off light at the optimal level, but this is subject to change, as it depends on the results of the previously mentioned survey involving user satisfaction in maximizing the sunlight in the room.

To meet our overall accuracy requirement of 90% accuracy in a 132 ft^2 room, we want our processing unit to correctly identify whether the user’s position is in the area of direct sunlight identified by the LAOE algorithm 90% of the time. Although we know that the total accuracy a typical user would want is 90%, we do not know how much error is allowed within each subsystem to achieve this overall error of 90%. We do know that if a single subsystem is less than 90% accurate then the overall accuracy of our integrated system can’t reach 90%. Thus we are currently giving each subsystem a hard 90% accuracy minimum.

To be able to capture more real-world data and to increase the reaction speed to the user’s movements, we also want our processing and calculations to take less than 1 second; this will help us more closely reach a feedback latency of 10 seconds. And, once again, although the requirements for minimizing sunlight on the user’s face for this are fairly defined, more requirements will be made later on after the survey about the blinds’ position to maximize sunlight in the room has been conducted.

V. Design Trade Studies

A. User Position Extraction

There were other methods we were considering using to calculate the position of the user in the room. One such method was using multiple cameras around the room, and using triangulation to identify the position of the user in the room. This method would allow us to forgo looking for a LIDAR device, and would be fairly accurate. However, using multiple cameras complicates the design, as the particular angle and position of which the cameras are put up have to be taken into account. It would also provide inconvenience to the user when setting up our BLINDS. To slightly simplify the design, as well as provide more convenience to the user, we decided to use a LIDAR and camera integrated device. Another similar method that was under consideration was using Bluetooth beacons, but that requires the user to hold onto a device to communicate with, which is also inconvenient for the user, which contradicts our goal to maximize convenience for the user.

Another design choice we made was the choice of using Haar cascades over other forms of facial detection. Although Haar cascades tend to be slightly less accurate compared to other methods, it is also faster than other methods. This is also

particularly pertinent because its lower latency makes it not only run well on a microcontroller, but helps us achieve our latency requirements. Yet another design choice we made was using a Fast Super-Resolution Convolutional Neural Network (FSRCNN) model to increase the resolution of the camera image, over other models. We decided to use a FSRCNN model over an EDSR model because although EDSR's gives the best results in terms of accuracy, it is slower [10]. Because we want our BLINDS to react in real-time, it is better to use a model like FSRCNN.

B. Sun Position Calculation

Initially, we considered a few different options for calculating the data on the sun. One was the use of a sun sensor or a sundial to determine where the sun is in respect to the window. The benefit of these physical forms of data measurement is that it can help us determine whether or not sunlight is being blocked from the window in some way, such as by weather or by objects such as trees or buildings. For instance, a lighter shadow or lack of shadow for the sundial may indicate that it is cloudy. However, there are quite a few drawbacks. The sun sensor would be an expensive option for finding information that could be calculated with just location and time of day. There would also be the issue of where exactly such a device would be placed, as putting it outside would make it separate from the entire blinds system. Our team also considered the use of a sundial, but methods of retrieving data and data accuracy would also be quite messy. We needed something that would be as accurate as possible, so we decided to simply calculate the location of the sun using Python modules, which would also only require the time, date and location.

C. Sensor System

To support the software solution we chose in the previous section, the sensors we need are a magnetometer, a photoresistor, Microcontroller with analog pins, and a Depth Camera. For our magnetometer we chose the Adafruit triple-axis Magnetometer MMC5603 because it is a reliable device with a heading accuracy of 1 degree. Although other magnetometers listed on Amazon are cheaper, their reviews in terms of reliability were quite low, and we did not want to take that risk. For our photoresistor, we chose the Elagoo photoresistor, as this is a component that we already own and will work sufficiently for our purposes. Finally, we chose the Arduino Uno for our microcontroller because this was also a device that we already had our hands on, and a component we have confidence and experience with in the past.

For our depth camera, we were originally contemplating between the Astra Depth Camera from Orbbec 3D, Intel RealSense Depth Camera L515, and RealSense Depth Camera D455. Our original plan was to use the Intel RealSense Depth Camera L515 because the CMU ECE inventory conveniently already owned one, and its range of up to 10 meters would fit our design requirements. Unfortunately, after some initial testing, we learned that the L515's effective range is reduced down to 1.5 meters in sunlight, which is a huge problem given the nature of our project. We then considered the Astra Depth Camera from Orbbec 3D, as it was relatively cheap and had a range up to 8 meters. However, we decided against it because there were no reviews for this depth camera, and therefore we were not able to confirm if the range was reliable. The Astra also did not have good documentation which we feel would be a huge issue. Thus, we settled with the more expensive camera, the Intel RealSense Depth Camera D455, which has many reviews affirming its 4.2 meter range even in direct sunlight and is well-documented enough for us to quickly learn how to utilize.

D. Motor Control

For our motorized blinds, we were choosing between the Automatic Window Roller Blinds [11] and the Arduino Automatic Blinds Opener [12]. The Automatic Window Roller Blinds is a design plan to make our own custom blinds, where the stepper motor will directly turn the rollers the blind's cloth is attached to. On the other hand, the Arduino Automatic Blinds Opener is a design which has an external motor with a gear that hooks onto the beaded string of the shade and controls the shade through the string. Both designs take 60 seconds to fully unroll/roll up. We decided to go with the Blinds Opener over the Roller Blinds because the Blinds Opener seems more versatile as it works with every blind as long as it has beaded strings. The Blinds Opener also fits our project better because the motor can sit on the window sill; this way, we don't need a long wire from our RPi all the way to the top of the window, which would have been necessary if we were to use the other design.

VI. System Implementation

A. Sensor System

The sensor subsystem consists of the Intel Realsense Depth Camera D455, Adafruit MMC5603 Magnetometer, and light detection circuit. The Intel Realsense Depth Camera D455 provides both a RGB camera feed and also lidar data at a resolution of 1240px x 720 px at 30fps as shown below in figure x. The Camera would be connected to the RPi via a USB-C to USB 3.1 cable. The sample output of this device can be seen in Fig. 1.

The second component of the sensor system is the Adafruit MMC5603 Magnetometer. This would be connected to the RPi via wires with I2C communication protocol. The Adafruit MMC5603 Magnetometer would generate and send magnetic orientation data to the RPi.

The last component of the sensor system is the light detection circuit. The light detection circuit is constructed by connecting a 10 kΩ resistor in series with a photoresistor to form a voltage divider. The voltage reading would be connected to the Arduino Uno analog pin with a wire. The

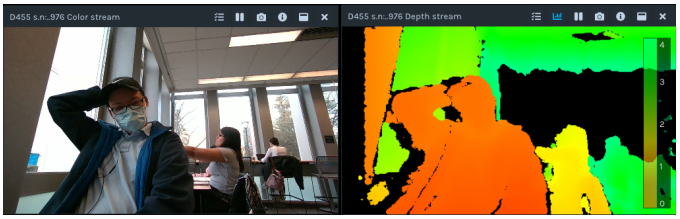


Figure 1: Sample Output of the Intel Realsense Depth Camera D455

Arduino Uno would be constantly communicating with the RPi via USB to USB via the Arduino Uno Serial Monitor if the measured voltage level is over the threshold for the light to be considered direct sunlight.

B. Processing Unit - User Position Extraction Algorithm

From the sensor system, the processing unit (the RPi) obtains real-world information. This information includes, but is not limited to, an image of the user and the room, as well as data from the LIDAR, which depicts the distance of the system to objects in the room, as shown in the previous section.

Using this information, the User Position Extraction Algorithm will be run on the processing unit. Its goal is to obtain the location of the user's chin, which will be used in tandem with the LAOE algorithm to determine if the user is hit by direct light or not. First, it will use OpenCV's Haar cascades to perform face detection on the camera input which it will obtain using the pyrealsense2 API, to detect the location of the user's face. Both the frontal and side profile cascades will be used, so that we can identify the user's face in multiple orientations. However, face detection issues may arise if the user is too far from the camera, and the resolution of the image of their face is not high enough. In this case, image processing to increase image resolution will be done using OpenCV's Super Resolution module and TensorFlow's Fast Super-Resolution Convolutional Neural Network (FSRCNN) model. Using the processed image, face detection will be performed again. We can convert the face detection output to pixels on the user's chin from the camera input. Given the pixels of the user's chin, we can map this pixel to pixel to the LIDAR data using the pyrealsense2 API, and from that we can identify the radial distance the user's chin is from the system. Then, by using the number of pixels away the user is from the center of the image, we can identify the angle of the user to the system. Using the distance and angle of the user, we can determine the 3-D coordinates of the user, as represented by the coordinates of their chin.

C. *Processing Unit - LAOE Algorithm*

As shown in the full architecture diagram in Appendix C and described in Section III of this document, the LAOE algorithm receives data on the user's position, the sun (azimuth, altitude, photoresistors), and the window's orientation. The user position is found from the algorithm above, the sun data is found through the python modules Suncalc.py and Geopy.py as well as the photoresistor, and the window orientation is found through the magnetometer. We want to check first whether or not the weather is affecting the sunlight. This is done through the data sent by the photoresistors. If there is no direct sunlight, we will keep the windows open/open them fully as the sunlight will not bother anyone. Otherwise, in order to find the adjustment that needs to be made to the blinds, our algorithm will be divided into a few functions. First, it will figure out if the user's position lies within the "area of effect" of the sun. This area, or more accurately, volume, is a trapezoidal prism of the projection of the sun from the window into the room. The idea is rooted in the fact that light travels in straight rays from the sun [8]. The rays will hit the corners of the window and travel into the room, at certain angles of altitude and azimuth. The projection of light traversing into the room at each of the four corners is what forms this box-shaped prism. It is better visualized and illustrated in Fig 2.

The first function maps the four coordinates of the window into the projection of the same four coordinates onto the floor by using the azimuth and altitude of the sun as the directions of the rays as they enter the room. Because the sun is very far away from the window in question, we can estimate all rays entering the room to have the same altitude and azimuth. The function will return the four coordinates of the corners of the light cast by the window. The coordinates are represented with the x-value representing the distance to the left or right of the window, the y-value representing the distance from the wall where the window is located, and the z-value being the physical height within the room. The origin is set at the floor under the window, with x value at the middle of the window.

The second function determines whether or not there is an intersection between the user and the LAOE. Using the eight corners we have found so far (four from the corners of the window + four from the corners of light projected onto the floor), we now have the eight vertices of the trapezoidal prism formed from the light. We can simplify the intersection to a calculation of whether or not the point lies between the two lines existing in each plane (XY plane and YZ plane). The equations for the four lines (two for each plane) can be found from the two coordinate points we have for each line (the previously calculated eight total points). This is illustrated in Fig. 3 and 4, and the intersection is calculated with a simple inequality intersection calculation.

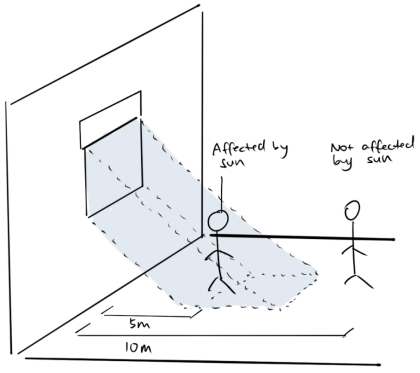


Figure 2: Visualization of Light Area of Effect

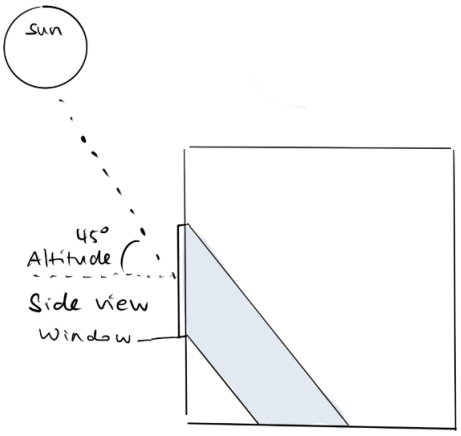


Figure 3: View from the YZ plane of LAOE

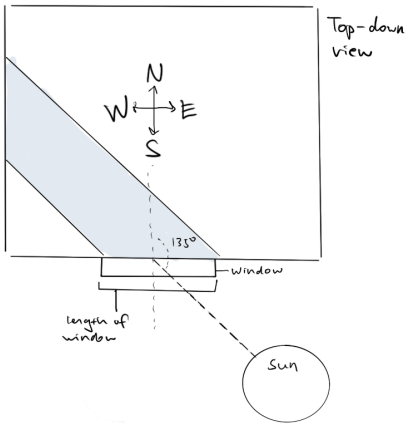


Figure 4: View from XY plane of LAOE

If we did not find an intersection, we can open the blinds fully so that light is maximized. However, if we did find an intersection, we must adjust the blinds. The last function calculates the adjustment that must be made in order to prevent light from hitting the user's face. To do this, our algorithm will backwards trace a ray hitting a target point (the bottom of the person's face) back until it reaches the wall where the window is located. This ray will use the same altitude and azimuth for its direction. This will allow us to calculate what the z-value of the bottom of the blinds should be (bottom of the blinds = top of the window), and adjust the blinds accordingly to that position. The result is sent to the motor control.

I. Motor Control

For our motorized blinds, we were choosing between the Automatic Window Roller Blinds [11] and the Arduino Automatic Blinds Opener [12]. The Automatic Window Roller Blinds is a design plan to make our own custom blinds, where the stepper motor will directly turn the rollers the blind's cloth is attached to. On the other hand, the Arduino Automatic Blinds Opener is a design which has an external motor with a gear that hooks onto the beaded string of the shade and controls the shade through the string. Both designs take 60 seconds to fully unroll/roll up. We decided to go with the Blinds Opener over the Roller Blinds because the Blinds Opener seems more versatile as it works with every blind as long as it has beaded strings. The Blinds Opener also fits our project better because the motor can sit on the window sill; this way, we don't need a long wire from our RPi all the way to the top of the window, which would have been necessary if we were to use the other design.

A. Tests for Sensor System

We will be testing the Depth Camera by making a person stand 1 meters, 2 meters, and 3 meters away from the camera at 0 degree, 30 degree, and 60 degree angles from the camera center. If all the distance measured by the depth camera is within ~2% of the actual distance, we can say the Depth Camera is fully functional because the D455 datasheet said to expect roughly 2% error. We can also check manually to make sure the RGB camera feed matches the real life scene it is filming.

We will test the Light Detection Circuit by moving in and out of different environments and checking the response. The environments we will be testing the circuit against are: in normal room light, in a dark room, in direct sunlight, and in sunlight that passes through clouds. The circuit is considered working if it only returns the result that there is light under the condition with direct sunlight hitting the photoresistor.

The Magnetometer would be tested against the result from the compass app on our phones. We will turn the magnetometer around every angle in a circle with 1 degree increments. It needs to be 1 degree increments because every degree of error would severely impact our LAOE prediction. The Magnetometer would be considered functional if it matches the result from our compass app for all of the angles.

B. Tests for User Position Extraction

Like the tests for the Depth Camera, we will test the User Position Extraction by making a person stand at set distances (1 meter, 2 meters, and 3 meters) away from the system. Then, we will run our User Position Extraction algorithm and compare its outputs to the actual measured angle and distances of the user. We want to achieve two different types of goals, and thus will have two different quantitative targets, when testing: Maximizing sunlight in the room, and minimizing the sunlight that hits the user's face. For maximizing sunlight, we need to conduct the previously mentioned survey regarding the amount of light the blinds should let in for optimal user satisfaction, so this section will mainly focus on minimizing light that hits the user's face.

For minimizing sunlight, we want to meet our overall accuracy requirement of 90% accuracy in a 132 ft^2 space. If the user is in the area in the room that gets hit by direct sunlight, as found by the LAOE algorithm, we want to be able to accurately identify that 90% of the time. To meet this 90%, the User Position Extraction algorithm can be broken down into three parts that can be tested separately: Determining where the chin of the user is using OpenCV, using that

information to find the radial distance of the user using LIDAR data, and calculating the angle of the user using pixels to the center of the image. When determining where the chin of the user is, we want our found location of the chin given an image, to be at or below the actual chin 90% of the time, to ensure light does not hit above the chin. The accuracy regarding how far below the chin our calculations should stay within pertains to maximizing sunlight in the room and will be determined after the aforementioned survey. Finding the radial distance of the user using LIDAR data is solely determined by how accurate the LIDAR data is, and we want this distance to be within ~2% of the actual distance, as mentioned in the sensor system section. For calculating the angle, we want the angle of the user to be as precise as possible, down to the pixel, as this can heavily affect whether the user is determined to be standing in the light or not, and affect our accuracy goals. Using the Depth Camera D455's maximum resolution and angle of view, and linear interpolation, we can calculate each pixel to be approximately 0.0431 degrees, so we would want our calculated angle to be within that angle range [13].

C. *Tests for LAOE*

Similar to the tests for User Position Extraction, we will be using real-life scenarios where a person is affected by sunlight to conduct testing. We will use data with user positions and times of day where people are affected by light and when they are not, and testing if the algorithm returns the correct adjustment accordingly.

In order to meet our overall accuracy goal of 90%, we want the algorithm to both be able to determine whether or not a person lies within the area of effect or not 90% of the time, and also ensure that the face is covered 90% of the time after an adjustment is made. We can do this by manually finding the minimum height at which the blinds can be lowered to and cover the face, and seeing if the adjustment covers more than the minimum height or not.

Additionally, we want to test whether the light is in a range such that the light entering the room is maximized. Once we have completed the previously mentioned survey to gauge how often and within what range the light from the adjusted blinds should end at in respect to the person (ex. 80% of the time within 5 cm of the person's chin), we want to collect data to determine what the adjustment should be in a real-life scenario. On a sunny day, we can find the upper and lower limits at which the height of the blinds should be at so that the light falls within an acceptable range to maximize light, and then test those height values against the output of the LAOE algorithm given the time of day, user position, and location at which we took the data.

D. *Tests for Motor Control System*

We will test that our Motor Control System can move the Blinds to the directed position within a 2 cm margin of error. There is a 2 cm margin of error because the stepper motor's smallest step size is ⅛ turn so it is not possible to move the blinds to every position. We will send the blind controls request to set the Blinds to every position from the top of the blinds downwards at 5 cm intervals. The blinds control would have to be able to do this in downwards, upwards, and mixed up order to move a spontaneous position (i.e. "move to ¾ closed then to ¼ closed then to ½ closed"). The Motor Control System is only considered functional if it works 100% of the time.

E. *Tests for Overall Integrated System*

The fully integrated system would be tested in a real life setting where we will make a person walk into sunlight and stand there and see if the blinds move so that the direct sunlight cuts off within the 10cm area below the person's chin. The person would then walk outside of the sunlight and we will see if the blinds fully open to let more sunlight into the room. If the blinds passed the first two tests then we would move to the final test where a person would be sitting in the room all day on a sunny day and we would observe if the blind adjusts properly throughout the day to block direct sunlight from hitting the person's face. The overall system would be considered functional if 90% of the time when the user is stationary, the sunlight is cut off at the optimal position as described in the requirement section.

VII. Project Management

A. *Schedule*

Our project is on schedule, and no adjustments are currently necessary. The Gantt chart for our schedule is shown below in Appendix B. Some tasks have already been started, such as the physical building of the blinds, as well as the development of the LAOE algorithm and User Position Extraction algorithm. Moving forward, we will continue to adapt and code our LAOE algorithm, develop our User Position Extraction algorithm, and continue to build our hardware.

B. *Team Member Responsibilities*

All the group members have rough experience in all the subsystems of the project, so we will all have a secondary task of helping each other with subtasks. However to utilize the specializations of each group member, the project work has been divided into two main parts: The hardware component, which involves working with the Arduino and motors, and the software component, which largely consists of implementing the LAOE and User Position Extraction algorithm on the Raspberry Pi. The work has been split up such that Jeff is in charge of the hardware portion of the project, and Dianne and Elizabeth will work closely together on the software components. All group members will be responsible for integration of the different components of the project.

C. *Bill of Materials and Budget*

A breakdown of the materials bought and used can be found in Appendix A below.

D. Risk Mitigation Plans

There are two risks our project may be facing: 1) the lack of experience with working Raspberry Pi 2) the difficulty of conducting real life tests. The Raspberry Pi 4 in our design is the microprocessor to perform the User Position detection, LAOE algorithm, and communication with the arduino. Therefore, it is a critical part of our project and a bulk of the computational work. The mitigation plan in this first case is to use a laptop as the processor instead. The second risk of our project, however, is more tricky since the weather is outside of our control. Our mitigation plan includes getting the MVP done early so we have a long window of time to test, and therefore more sunny days to work with. We are also collecting data throughout the semester, so we can test with past data, as well as utilizing artificial light to trick our light sensing circuit and whether or not the blinds will block the artificial light in the same way it blocks sunlight.

VIII. Related Work

Serena Shades by Lutron [9] are the only smart blinds on the market that have a light optimization feature built into the blinds. The blinds, over the course of the day, automatically tilt themselves at an angle that prevents direct sunlight from entering the house, but allows natural light to fill the room. The Natural Light Optimization algorithm that Serena Shades implements utilizes a person's address and window orientation, information which has to be manually inputted via the Lutron app, to calculate the angle the blinds should be tilted at.

Our project differs from Serena Shades by a few key factors. For one, when calculating how our blinds should move in order to optimize light in a room, our project not only takes in a person's address and window orientation, but it also takes into account a user's position within a room. This means that our BLINDS allows more natural light in a room, by letting in direct sunlight that does not negatively impact or hit the user. Additionally, our BLINDS also considers if light is really coming through the window or not, as a factor of whether or not an adjustment should be made, while Serena Shades functions the same regardless of actual light. Our system will only adjust if it detects the presence of direct sunlight through our light sensing circuit, thereby expanding on Serena Shades' Natural Light Optimization algorithm.

IX. Summary

Our BLINDS project aims to improve users' quality of life by freeing them from the duty of manually adjusting blinds. By automatically adjusting to the optimal position that will let in the most beneficial light to the user, our BLINDS will simultaneously protect users from the sun's harmful UV rays, and encourage users to let in more natural light into their room, improving their mental health. To do this, our smart blinds will adjust its position based on a multitude of factors, such as whether it detects light coming through the window, the angle of the sun, the user's address, and their position in a room. Our BLINDS will also only adjust after a user has maintained the same position, to avoid startling the user and to provide a smooth experience that they will enjoy.

One of the challenges that need to be tackled include developing our algorithms in a way that meets our project's accuracy requirements, as both our LAOE and User Position algorithm inherently have some amount of error. This is because our LAOE models light in a simplified way, and our User Position algorithm utilizes OpenCV which isn't always very accurate. To reduce error in our algorithms, we will focus our efforts on testing and continually adapting our algorithms to improve their accuracy. Another challenge that exists is the difficulty of performing real-life testing, as the amount of testing we can do with real sunlight is dependent on a multitude of volatile factors, such as weather. Although mitigation plans have been implemented to counteract this challenge, it is possible that these plans may not proceed as expected. Finally, one last challenge is the integration of the very different subsystems in this project. However, we will do our best to stay on schedule, which provides some leeway in case problems occur.

Glossary of Acronyms

API - Application Programming Interface
LAOE - Light Area Of Effect Algorithm
RPi – Raspberry Pi

References

[1] Boyce, Peter R. "Review: The Impact of Light in Buildings on Human Health." Indoor and Built Environment, vol. 19, no. 1, Feb. 2010, pp. 8–20, 10.1177/1420326x09358028.

[2] Young RW. The family of sunlight-related eye diseases. Optometry and Vision Science : Official Publication of the American Academy of Optometry. 1994 Feb;71(2):125-144. DOI: 10.1097/00006324-199402000-00013. PMID: 8152745.

[3] Shishegar, N, and M Boubekri. "Natural Light and Productivity: Analyzing the Impacts of Daylighting on Students' and Workers' Health and Alertness." International Journal of Advances in Chemical Engineering and Biological Sciences, vol. 3, no. 1, 21 May 2016, 10.15242/ijacebs.ae0416104.

[4] Mitchell, Travis. "Covid-19 Pandemic Continues to Reshape Work in America." Pew Research Center's Social & Demographic Trends Project, Pew Research Center, 23 Mar. 2022, https://www.pewresearch.org/social-trends/2022/02/16/covid-19-pandemic-continues-to-reshape-work-in-america/.

[5] Kazanskiy, Nikolay, et al. 'Performance Analysis of Real-Time Face Detection System Based on Stream Data Mining Frameworks'. Procedia Engineering, vol. 201, 2017, pp. 806–816, https://doi.org10.1016/j.proeng.2017.09.602.

[6] Graywind, "Graywind Motorized Blackout Roller Shades." Graywind, 2020, https://www.graywindblinds.com/collections/roller-shades/products/smart-shades-blackout-designed

[7] Helling, Author Andrew. "Average Bedroom Sizes in a Home - The Ultimate Guide." REthority, 14 Feb. 2023, https://rethority.com/average-bedroom-size/.

[8] Bird, J. O., and P. J. Chivers. '31 - Light Rays'. Newnes Engineering and Physical Science Pocket Book, edited by J. O. Bird and P. J. Chivers, Newnes, 1993, pp. 248–255, https://doi.org10.1016/B978-0-7506-1683-6.50034-5.

[9] Serena Shades. SerenaShades, Lutron, 2016, https://www.serenashades.com/.

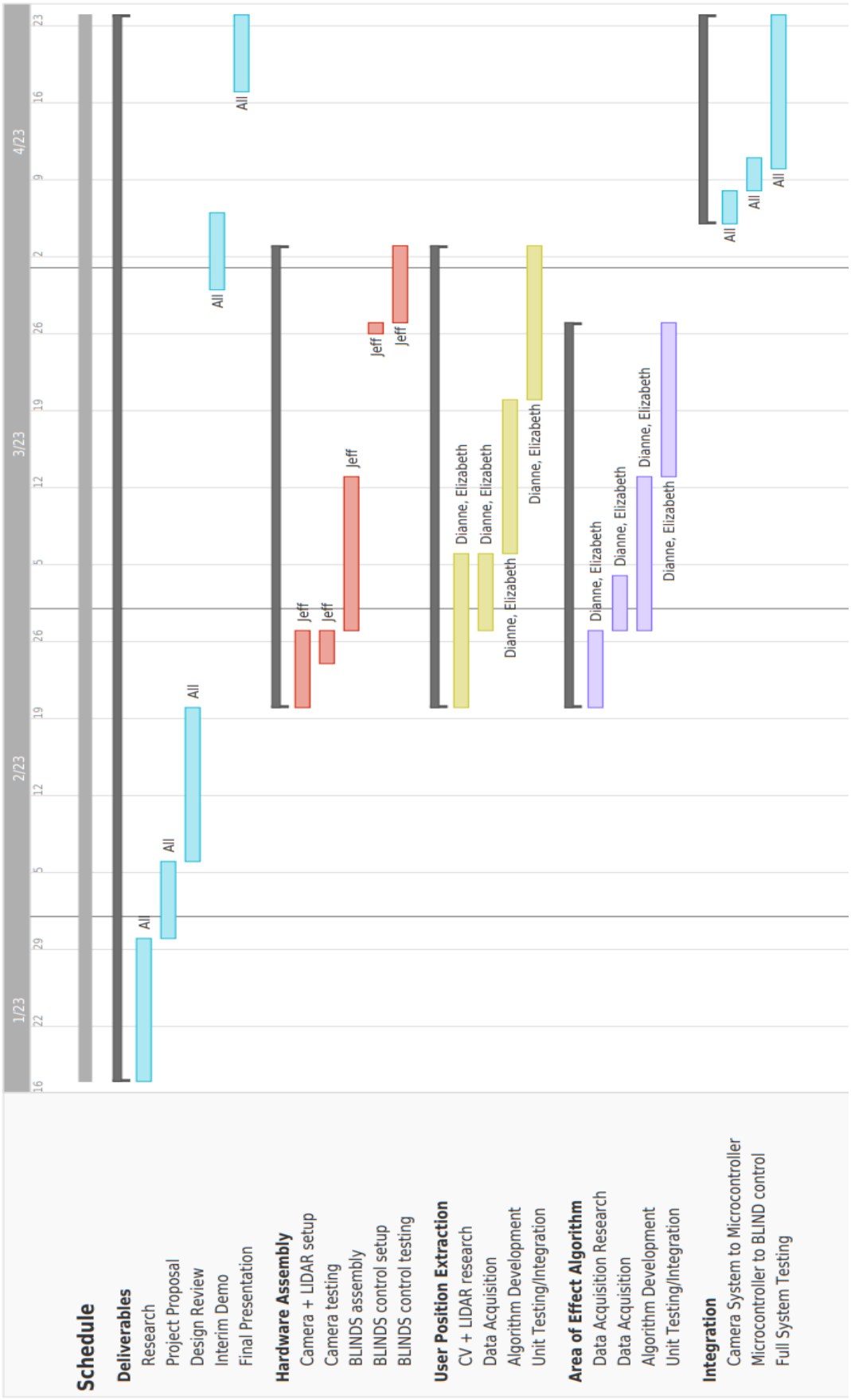
[10] Vardan Agarwal, Lipi Patnaik, et al. "Super Resolution in Opencv." LearnOpenCV, 18 Oct. 2021, https://learnopencv.com/super-resolution-in-opencv/#sec3.

[11] “Automatic Window Roller Blinds.” Projecthub.arduino.cc, 10 June 2021, projecthub.arduino.cc/twinsen01/85418824-2918-44ac-87c5-dc7deb4be4b5.
[12] Klements, Michael. “Arduino Automatic Blind Opener .” The DIY Life, 2 Apr. 2020, www.the-diy-life.com/arduino-automatic-blind-opener-works-with-a-remote-control-alexa/.
[13] Coffin, Jerry. “OpenCV: Calculate Angle between Camera and Pixel.” Stack Overflow, 1 May 1960, https://stackoverflow.com/questions/17499409/opencv-calculate-angle-between-camera-and-pixel.

Appendix A: Bill of Materials and Budget

Description	Model	Manufacturer	Source	Quantity	Cost
Lidar and Camera Integrated System	RealSense Depth Camera D455	Intel	Intel	1	$453.27
Window Blinds	Grey - 20" W x 72" H	Homebox	Amazon	1	$31.99
Screws kit	810Pcs M3 x 4/6/8/10/12/14/16/ 18/20 mm Screw Assortment Kit	VIGRUE	Amazon	1	$17.99
Stepper Motor Driver Module	A4988	HiLetgo	Amazon	1	$10.19
Stepper Motor	42BYGH	Usongshine	Amazon	1	$10.99
Main Computer	Raspberry Pi 4 8 GB	Raspberry Pi	ECE Receiving	1	$0
Motor Controller	Arduino Uno Rev3	Arduino	N/A	1	$0
Photoresistors	N/A	Elegoo	N/A	4	$0
Magnetometer	MMC5603	Adafruit	Adafruit	1	$5.95
3D Printed Pieces	N/A	N/A	TechSpark	1	$17.66
					$548.04

Appendix B: Gantt Chart of Schedule



Appendix C: Full Architecture Block Diagram