SmartStand

Sebastian Garcia, Mary Rose Rubino, and Olivia Yang

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—Laptops, while portable, are not ergonomic to use as the screen height is several inches lower than eye level when placed on a table. This less-than-ideal screen height can cause muscle strain and poor posture as users often hunch over for extended periods of time in order to better see the screen. We aim to fix this issue by creating a portable laptop stand that is able to calibrate and rise to the ergonomic height for any user. Additionally, we will provide a graphical user interface that can be minimized which will track a user's eye movement and posture over time. These trackers will run in the background and notify users when they have been sitting for long periods of time and when they should take a break to prevent eye strain.**

***Index Terms*—Arduino, Computer Vision, Eye Tracking, Facial Landmark Detection, Neural Network**

I. Introduction

Laptops are incredibly convenient and portable; they're lightweight, compact, and can provide the ability for a user to work and communicate with others from practically any location. However, laptops are usually not ergonomic since the screen sits just above the surface that they are placed atop whereas the ergonomic height for a computer screen is slightly below eye level. For a user sitting straight with good posture, the screens provided by laptops are often several inches lower than the ideal ergonomic height. For some groups of individuals that are often required to spend long periods of time using their laptops such as students and professionals that work remotely, this screen height can cause several health issues. Since these groups may spend many hours at a time working on their laptop, they can experience neck pain and muscle strain since they have to crane their neck at an uncomfortable angle, and they may often find themselves hunching or slouching in order to better see the screen.

Our project aims to solve this issue by creating a portable laptop stand that will automate the height and angle adjustment to face parallel to a user at an ergonomic height for them. This will help their posture by allowing them to use their laptops without constantly looking down at the screen. We also aim to improve a user's health by determining when they have been seated for long periods of time in order to remind them to stretch. Additionally, to reduce the eye strain from long periods of screen time, we will use computer vision to track the user's eye movement and send notifications for them to look away after 20 minutes of intense focus.

While portable laptop stands currently exist, most are not adjustable to match a user's height. Additionally, our system will serve multiple purposes to help the wellbeing of the user rather than just being a stand on which to place a laptop. Our project aims to help reduce the strain and discomfort of using a laptop for long periods of time.

II. Use-Case Requirements

Based on our system's user base, we compiled a list of requirements that guided our design process:

- Our device is intended to be portable, so it will need to be light enough for the user to carry without struggling. We determined that 5 lbs would be a reasonable maximum weight, as this is slightly more than the weights of some heavier fixed laptop stands.
- Our device is intended to be more convenient than existing fixed laptop stands, which includes its calibration time. Because we use a feedback loop to gain the most accurate calibration results, it is difficult to estimate setup times. However, we determined that 1 minute would be a reasonable total calibration length, including landmark measurements as well as lifting and tilting of the stand).
- Since our device is meant to be portable, we determined that we would need either a battery or a simple wall outlet connector. We opted for an outlet connector because of its simplicity and lighter weight compared to a standalone battery.
- Our device monitors the user's line of sight to limit eye strain from prolonged staring at the laptop screen. After a user looks at the screen for 20 minutes, it is recommended that they look away for 20 seconds or longer. We will sample at a rate of once every 3 seconds. This rate eliminates glances away from the screen that do not contribute to reducing eye fatigue.
- One of the intended uses of our device is to notify users when they are slouching. We will sample camera data to obtain the user's current position. We will then compare this to the landmarks from the calibration stage, using an algorithm to keep track of the user's landmarks as the stand changes height and angle. We will sample this data every 15 seconds, which will eliminate changes in position that are unrelated to posture, such as bending down.
- Another use of our device is to display the user's posture progress in the app so that the user can view the number of negative changes over time. We determined that information about the number of times the user changed posture per day should be recorded and displayed as a bar graph that includes the past 30 days.

III. Architecture and/or Principle of Operation

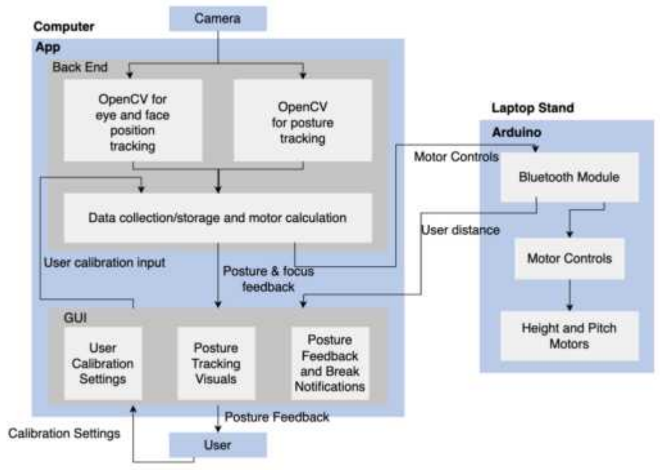


Fig. 1. Detailed overall block diagram of the device.

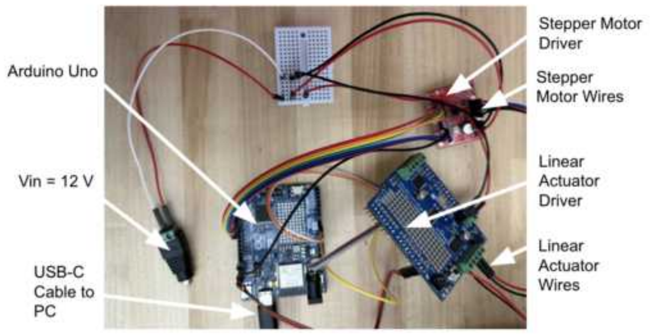


Fig. 2. Hardware components.

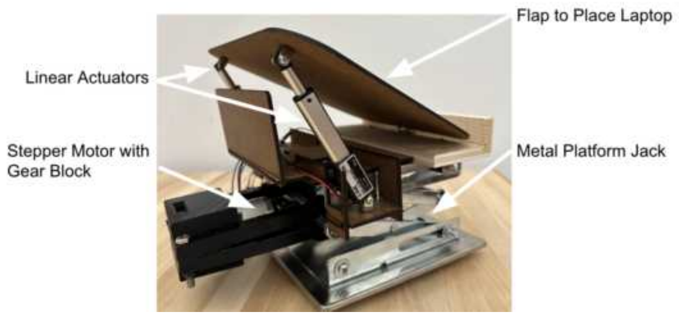


Fig. 3. Overall structure of SmartStand.

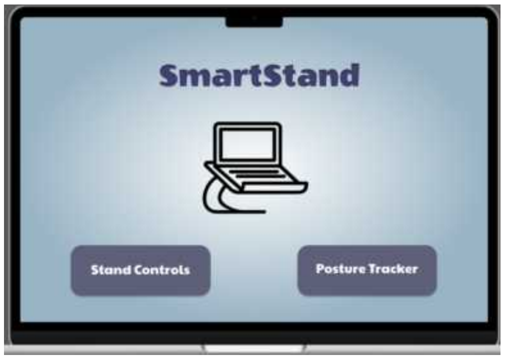


Fig. 4. GUI Homepage.

Our system's design and subsystem interactions are shown in Fig 1. Overall our system is composed of hardware, software, and mechanical components. On the software side, all components are included as part of the GUI. On the hardware side, all components, including the motors, linear actuators, and the Bluetooth module will be included on the stand. Our mechanical components refer to the stand itself.

Changes from the design report include: using a standard wall outlet for power (12 V, 3 A) instead of using a standalone battery. The motor we used ended up being much heavier than expected (because we ended up attaching a gearbox for added torque), so we didn't want to weight the stand down even further. Another important change was building a shelf for the linear actuators so they would be shifted below the top plate of the platform jack. The change we made is more clear by comparing Fig. 5. below with Fig. 3.

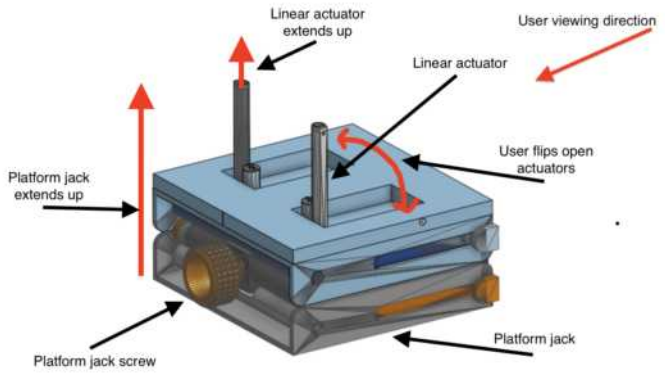


Fig. 5. Old Linear Actuator Design

PI 1-3: For engineering, our system used many subsystems and interfaces between hardware, software, and firmware. As such, we used many engineering techniques to integrate these systems as well as fabricate mechanical designs and components.

PI 1-4: For science, we were required to use many different motor drivers in order to match the exact type of motor due to the way they need specific current or voltage drives, which led us to use a chopper driver which was specifically designed to maximize the torque of a stepper motor.

PI 1-5: For mathematics, we needed to calculate the torque required for the platform jack with our given weight of a laptop and the peripheral equipment used.

IV. Design Requirements

Based upon our use case requirements, we determined a set of design requirements that ensure our device will be effective.

- The stand must lift to 12 inches. This height was determined based upon the average desk height, and the upper average person's sitting height to determine a maximum height that would allow for most people to be able to use the product.
- The stand must angle up to 45 degrees. This angle was what we determined to be a reasonable degree of freedom to account for different sizes of laptops that open to different maximum angles.
- Slouch detection must be determined in less than one second.The current data must be compared to the calibration data within this period. This was determined to be a reasonable amount of time so that the user would receive a notification quickly.
- Eye detection must be determined in less than one second. This ensures that notifications about how long the user looked at the screen are sent in a timely manner.
- The current ideal height must be reached within 5 seconds. This means that during one cycle of the height-raising feedback loop, the height sent to the Arduino from the laptop must be reached within 5 seconds. This was determined in case multiple cycles are necessary. Making this less than 5 seconds ensures that our calibration time remains low.
- The current ideal angle must be reached within 5 seconds. This means that during one cycle of the tilting feedback loop, the angle sent to the Arduino from the laptop must be reached within 5 seconds. This was determined in case multiple cycles are necessary. Making this less than 5 seconds ensures that our calibration time remains low.
- The margin of error for the laptop stand's angle mechanism must be less than 5 degrees. Meaning, that when the stand is fully calibrated, the angle of the laptop in relation to the user's face will be within the range of 85 to 95 degrees. We determined this angle based on the tradeoffs between usability (whether the angle is accurate) vs calibration time.
- The margin of error for the laptop stand height must be less than 3 inches. Meaning that when the stand is finished calibrating, the user's laptop will at max be +3 inches or -3 inches off of the ideal position. We determined this angle based on the tradeoffs between usability (whether the tilt is accurate) vs calibration time.

V. Design Trade Studies

Our design involved many tradeoffs mainly in the following areas: platform jack choice, battery pack, linear actuators, wired connection between arduino and PC, neural network for posture detection.

A. Trade Study: Platform Jack

One of our main design requirements was that SmartStand is compact. Initially, our plan was to have a 3D printer type construction where z-axis movement was accomplished by using screwed stepper motors similar to how a 3D printer moves the print nozzle in the z axis. However, we felt that this would be too cumbersome to carry around and would detract from the user being able to comfortably use his computer. So instead we opted for a platform jack which is extremely compact in its closed state (great for transportation), has incredibly simple vertical control (only requires motor rotation of a screw), and is capable of lifting heavy loads as demonstrated by the use of jacks in industrial settings.

B. Trade Study: Power Supply

We initially wanted to use a battery pack because we wanted the system to be completely wireless. The user connects to SmartStand via bluetooth and the user does not need to connect a power cable to an outlet. However, when we measured the weight of SmartStand with its finished mechanical design, the weight came out to 10 lbs, 6 lbs over our weight limit. So we decided to opt for the lighter, wired power supply choice.

C. Trade Study: Linear Actuators

We conducted a lot of research to decide on how to change the angle of the computer so that eventually the screen is parallel to the plane of the user's face. We considered using stepper motors with screws similar to a 3D printed but again for the same reasons stated in trade study A we felt that this design would be too invasive for the user. Additionally, we were not confident that the stepper motors would be fast enough. Since we aimed for a 5 second setup phase for SmartStand, we knew that the linear actuators we chose that are capable of 2 inches per second would meet the requirement. Also, the linear actuator implementation is extremely simple. We will simply use a GPIO switch from the Arduino to toggle the actuators.

D. Trade Study: Neural Network for Posture Detection

The input data to our slouch classifier is simply the deviation of body landmarks from a known "good posture" state. From this statement you could argue that the classifier could simply be a step function. For example, if landmark A (e.g. the shoulder landmark) is 10 pixels away from the ideal position, then slouch is true, else slouch is false. While this scheme is simple, it fails to take into account the big picture, that is all the other landmark data (e.g. the other shoulder, and facial landmarks). Furthermore, the thresholder fails to assign weights to these other landmarks.

We chose a neural network design because it would assign weights automatically, would be able to input all the landmark data, and it would be straightforward to train because our team could generate the training data.

E. Nema 17 Stepper with 5:1 Gearbox

Initially, we chose to use a Nema 17 without a gearbox so it had an rpm of about 100. We could manually adjust the rpm in the Arduino firmware but this would decrease the performance of the motor. With this high-speed motor, we were able to lift the platform jack fully in about 5 seconds. It was blazing fast and made us meet the time-based design requirement that the stand completely adjust within 5 seconds. However, once we put the laptop on the stand, the additional weight made the motor stall because it lacked the torque necessary.

So, we opted for the same motor but with a 5:1 gearbox, this drastically slowed the speed of our system, but with a torque of 3 Nm according to the specification, this ensured that our system was reliable. In the end, we traded off speed for torque so we had a more reliable and performant system.

We needed a way to calculate the minimum torque necessary to lift the computer so we would not over-sacrifice torque for speed. We used the equation below.

$$\tau = \left(\frac{F(tan(\lambda) + f)}{1 - f tan(\lambda)}\right)r$$

In the above equation, the torque (tau) required to rotate the lead screw of the platform jack depends on the weight pressing down on the platform (F), the lead angle (lambda) of 2.85 degrees for the type of screw we used, the coefficient of friction between the screw and the threaded hole (f), and the radius of the screw (r).

V. System Implementation

The entire system will consist of a modified platform jack which will contain the arduino, motor drivers, and motors.

A. Subsystem A: Platform Jack

The platform jack will be responsible for lifting and lowering the computer along the z axis. This will be accomplished by having a motor rotate a screw inside the platform jack, converting rotational motion to linear motion. A diagram depicting the system is shown in *Figure 3.*

The platform jack itself was manufactured by Amazing-us. It weighs approximately 3.5 pounds which meets our use case requirement for weight, and leaves some room for our modifications.

The top of the stand is 8" by 8", which is large enough to balance a laptop. This platform jack also has enough vertical elevation change to meet the height requirement (12 inches of elevation to accommodate almost all users and the platform jack is capable of 13 inches) The platform jack before any modifications is shown in *Figure 6.*

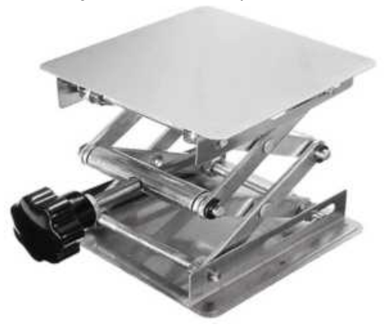


Figure 6. Platform Jack purchased from manufacturer Amazing-us

However, this stand does not meet all of the requirements of the product. To facilitate the angle raising mechanism, ⅛" thick wood was laser cut to create a shelf. In addition, a wooden lip to prevent the laptop from sliding off of the stand, and a flap to keep the laptop steady while the stand is being adjusted were also crafted from wood. To keep the stepper motor steady while it raises the stand, a 3D manufactured motor mount was created. This mount is held in place using screws. On the cosmetic side, we manufactured a wall and a small shelf to hold the arduino and motor drivers, to keep them out of sight for the user. These additional components are shown and labeled in *Figure 3*.

A. Subsystem B: Linear Actuators

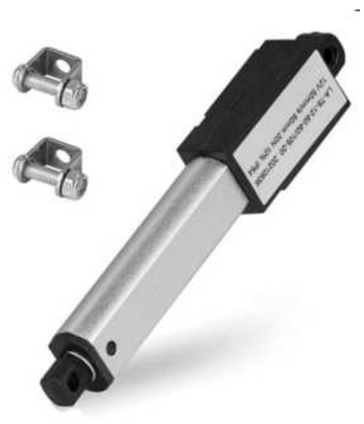


Figure 7: Linear Actuators Selected, manufactured by ECO LLC

The linear actuators are the angling mechanism, which will tilt the laptop until it is parallel to the user's face. As previously mentioned, these linear actuators will be housed in the shelf shown in *Figure 3*.

Additionally, since the linear actuator's top screws are attached to the flap, rubber stickers were used to prevent the bottom of the user's laptop from becoming scratched.

These linear actuators are rated for 4.5 lbs of force. Since most laptops weigh about 3.5 lbs and there are two actuators, this means each actuator will be responsible for about 1.75 lbs which is well below the max rating. The angle mechanism's safety factor is approximately 2.

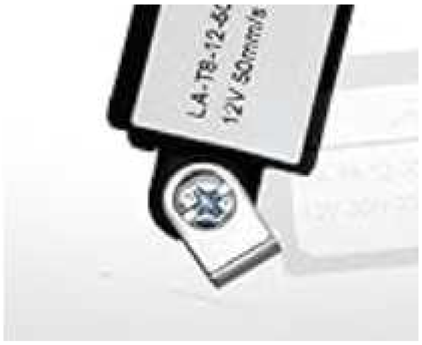


Figure 7: Bracket for Rotation

Because the linear actuators are screwed into the shelf, and attached to the flap, they need to rotate slightly in order angle the flap while maintaining two points of contact. Thus, we made use of the brackets shown in *Figure 7.* These brackets will allow for smooth rotation of the actuators coming out of the platform jack. They will be attached to the ends of the linear actuator, and then using the hole in the bracket, will be screwed into both the flap and the shelf, so this component will remain fixed while the stand is in motion.

B. Subsystem C: Posture Detection

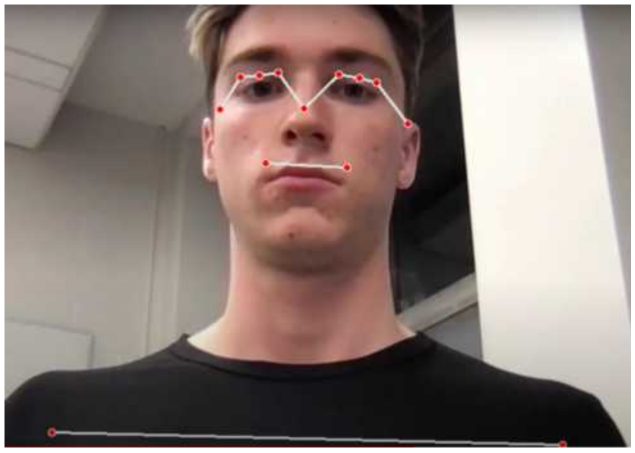


Figure 8: OpenCV for Posture Landmarks

To notify the user when their posture suffers, we will be running a Python program that the user will launch from their machine. This program will stream camera frames from the webcam and pass them through an off-the-shelf neural net that detects landmarks in the face and shoulders of the user.

To determine when the user is slouching, we will have a calibration stage after SmartStand has transformed the computer position and orientation to the most ergonomic state. In this stage, we will ask the user to sit with the most perfect posture they can, and the program running will take a "screenshot" of the posture, essentially recording the landmark data. Then the Python program will be taking the difference in positions between the ideal landmark coordinates and the current coordinates for every frame received by the camera. When there is sufficient deviation in the landmark coordinates, the user will receive a notification to fix their posture.

We plan on using a neural network for the slouch classification because it will handle the weighting of the various landmarks automatically and will be easy to train. All the members of team B2 will supply hundreds of "bad posture" and "good posture" photos with which to train the model.

Specifically, posture detection was implemented by taking the distances shown by the green lines in the image below.

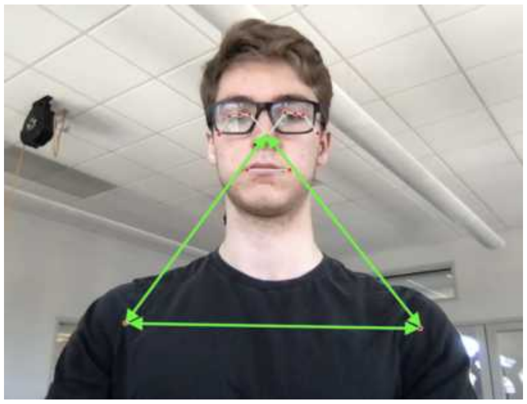


Figure 9: Posture Detection Distances

During the calibration stage, these distances are recorded. Then during the posture detection stage, these distances are again measured, and the difference between the distances from the current video frame and the “reference” video frame are passed to the neural network model. These differences in landmark distances were the data passed to the posture model.

We opted to use this data rather than the absolute positions of each landmark because with this method of data collection, if the user shifts their body left or right, the lengths of the green arrows shown above will be the same, so theoretically the posture detector output will be the same. If we had used absolute positions (which we did initially), we might get the incorrect prediction. We could have used absolute landmark positions, but this would have required a much deeper neural network and vastly higher amounts of training data which was not available to us.

C. *Subsystem D: User Calibration*

The adjustment process will start with a calibration stage. In this stage, a user will navigate to the calibration setup page on the GUI, and then raise or angle their laptop until the screen is flush with their face. This is necessary in order to get a clear reference image of their face without skew from having their camera at an angle.

Then, using a Python program, the frames at this calibration setup are provided through an OpenCV camera stream collected from the user’s webcam. Using this reference frame, the 68 facial landmark detector will be used to analyze the face found in the frame, and locate the coordinates of these landmarks. The vertical distance between a landmark located between the user’s eyes (point 27 illustrated below) and one located at the tip of their nose (point 33 illustrated below) will be calculated. Similarly, the horizontal distance between the two landmarks - one located on the left eye (point 39 illustrated below), the other on the right (point 42 illustrated below) - will be calculated and recorded. Additionally, a ratio of vertical to horizontal landmark distance is calculated so that the user’s facial feature information can be used regardless of the distance they are from the camera, since being closer will result in larger distances overall than being far from the camera. After the user completes the instructions during the calibration process, this facial recognition and landmark distance data will be saved to a file so that this state can be saved for future reference without needing to recalibrate.

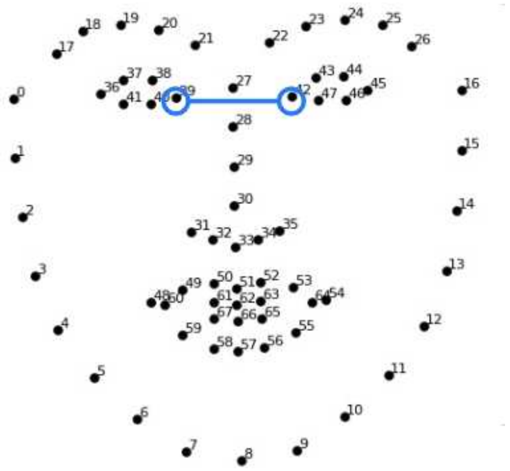


Figure 10: Facial landmarks used for horizontal distance.

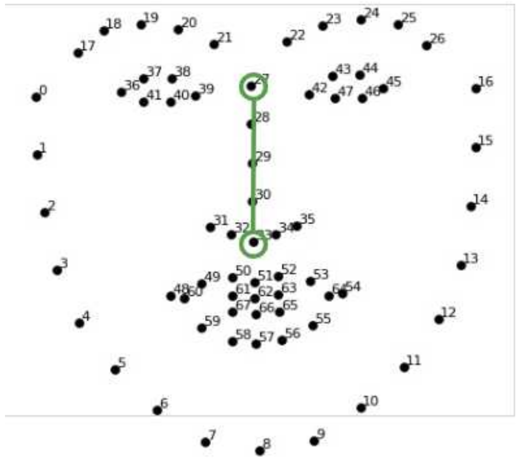


Figure 11: Facial Landmarks used for vertical distance.

D. *Subsystem E: Height and Angle Adjustment Process*

After this calibration stage, the user will be prompted to sit up straight with the best posture they can. After doing this, the distance between landmarks 27 and 33 as well as between landmarks 39 and 42 will change because the plane of the user’s face will be at an angle with respect to the plane of the computer screen. This difference in landmark distance is the basis for our SmartStand automatic adjustment.

In order to determine the ergonomic height and angle to lift and pitch the laptop for a given user, we used an iterative process that alternates between adjusting height and angle until certain thresholds are met as described below:

Adjust Height:

If the y-coordinate of the landmark between the user’s eyes (landmark 27) is located above the midpoint of the camera frame, then the laptop will be raised until it is within a threshold of the midpoint. Otherwise, if this landmark’s y-coordinate is below the midpoint of the frame, the laptop will be lowered until it is within a threshold of the midpoint. If it is within the threshold of the midpoint, then the state of the adjustment process is switched to angling the laptop.

Adjust Angle:

In this stage, the frame is processed through the 68 landmark detector to locate the coordinates of the user's face at this current position, and the vertical and horizontal landmark distances as mentioned above in Section D are calculated. Then, the current ratio of vertical to horizontal landmark distance is compared to the ratio saved from calibration, and if the current ratio is larger than the calibration ratio, the laptop is angled further towards the user. Alternatively, if the current ratio is smaller than the calibration ratio, then the laptop is angled farther away from the user. If the current ratio is within a threshold of the calibration ratio, then the process will switch back to adjusting height.

When both conditions are met - the current ratio is within a threshold of the calibration ratio and the landmark at the center of the user's eyes is within a threshold of the midpoint of the camera frame - then the adjustment process is finished and the ergonomic height and angle has been found.

Below is a state diagram of the various stages of the adjustment process that follow the calibration step.

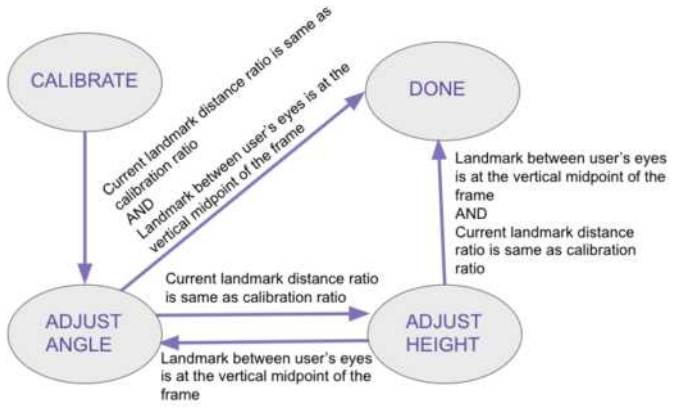

CALIBRATE
DONE
ADJUST ANGLE
ADJUST HEIGHT
Current landmark distance ratio is same as calibration ratio AND Landmark between user's eyes is at the vertical midpoint of the frame
Landmark between user's eyes is at the vertical midpoint of the frame AND Current landmark distance ratio is same as calibration ratio
Current landmark distance ratio is same as calibration ratio
Landmark between user's eyes is at the vertical midpoint of the frame

Figure 12: State Diagram of the Position Adjustment Algorithm

E. *Subsystem E: Python to Arduino Interface*

After the next adjustment step is decided by the adjustment algorithm detailed in Subsystem D, the linear actuators and stepper motor must be moved in the desired direction. This is achieved by sending a packet from Python to the Arduino over Bluetooth using the Python library Bleak which contains information such as whether the motor or linear actuators are to be adjusted, and in which direction they should move. The Python program will block and wait until it receives an acknowledgement packet from the Arduino over Bluetooth before continuing with the adjustment process.

F. *Subsystem F: Motor Selection*

Figure 13: 12 V Motor Selected: Nema 17 motor with 5:1 GearBox, manufactured by Stepperonline

This motor shown in *Figure 10* will be used to turn the screw to lift the platform. The torque calculations shown in the design tradeoffs section proved that we needed more than 3 Nm of torque in order to turn the screw. Thus, it is clear that a stepper motor with a Gearbox would be the only way to meet this requirement while also meeting the weight requirement, as some higher torque stepper motors require more space and also more voltage.

G. Subsystem G: Motor Coupler

To actually make the stepper rotate the lead screw in the platform jack, we had to couple the motion of the stepper motor rotor to the lead screw. This was accomplished with a coupler. This coupler was the ideal because the stepper motor rotor had a flat side for set screws. However, our lead screw did not, so we had to sand down a flat side of the lead screw to get the set screws of the coupler to hold the lead screw in place. This can be seen in the figure below.

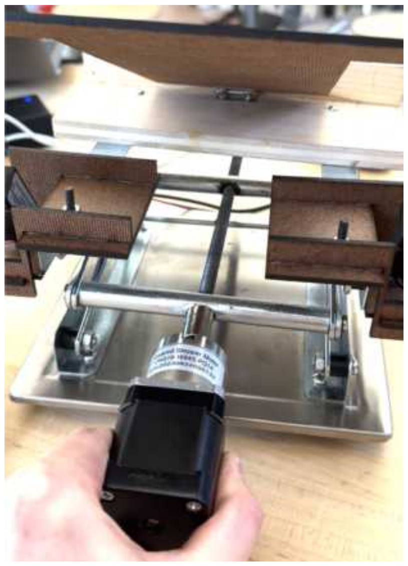


Figure 14: Motor Coupler in Action

II. Test, Verification and Validation

To test for the height adjustment delay, we will place the stand at a low height and test the ability of the stand to both raise to the maximum height and complete its adjustment cycle and settle at the appropriate height within 5 seconds.

The tests described below were run over 9 trials, using all 3 group members as test subjects in order to use different heights for testing. We also used different starting angles: one with the laptop opened fully, another with the screen slightly opened, and one with the screen already parallel with the vertical.

In order to test the total adjustment time, we timed the length of the process after calibration, and after choosing automatic ergonomic adjustment on the GUI. After the program ran and completed the adjustment process, sending a done signal to the arduino, we stopped timing. We found that the total adjustment time was around 92.57 seconds, which was much larger than we had set for the design requirement. However, the lower speed was a necessary trade off in order to gain higher torque from the gearbox stepper motor.

To test for the height adjustment accuracy, we had each test subject determine where their eyeline met the screen, and then measured the vertical distance between that point and the top of the laptop screen, since the ergonomic height for a computer screen is achieved when the eyeline meets the top of the screen. We found that the average deviation from this point was about 0.267 inches. This was within the acceptable range of height error as described in our design requirements.

To test for the angle adjustment accuracy, we measured the angle between the laptop screen and a complete 90 degree angle from the table, and we found that the average angle error was about 4.33 degrees - always angled in the direction away from the user. This was within the acceptable range of angle error as described in our design requirements.

To test for the latency of the posture detection, we will calibrate the stand with a user exhibiting good posture first, then having the user slouch and check that the Python program processed the image and categorized the posture type within one second.

To test the weight of the stand, we will weigh the stand when fully composed with the motor, linear actuators, battery, and Arduino with its motor driver and make sure that this weight is below 4 pounds.

Overall, the main quantitative design requirements are shown below.

Design Requirement	Target	Result
Total Adjustment Time	< 5 seconds	92.57 seconds
Height Error	< 3 inches	0.267 inches
Angle Error	< 5 degrees	4.33 degrees
Maximum Height Extension	>= 12 inches	12 inches
Maximum Angle Extension	>= 45 degrees	55 degrees

Noticeably, the total adjustment time is much longer than expected, this is because we used a gearbox on our stepper motor which gave us the torque we needed to lift the computer, but at the cost of speed. The height and angle errors are within our goals because of extensive tuning of the adjustment algorithm.

A. Test Results for Posture Detection

We did not end up integrating the posture detector with the overall GUI because we did not find that it was accurate enough. We aimed for 95% accuracy of the detector because we did not want to bother the user with spurious notifications to fix their posture when their posture was fine. During testing (on data I collected myself), the validation accuracy of the posture detector model was 98.5%. However, when I tested the same model on my teammates, accuracy dropped down to ~65%. Even when we all contributed training data to the model, it was unable to make intelligent predictions for a general audience. This was likely due to the simplicity of the model which had only one hidden layer and no regularization. If I could repeat the project, I would spend more effort to refine the model to improve general accuracy.

III. Project Management

A. *Schedule*

Our schedule is listed in Figure <>. The newest version of our schedule changed to no longer include time for creating a PCB, and instead included more time for the construction of the stand since the mechanical aspects of our project became increasingly more time consuming, and a PCB was not necessary or particularly helpful for our product.

B. *Team Member Responsibilities*

Sebastian's primary responsibility is to train the neural network with images of varying different types of posture and create the classification of "good" vs "bad" posture that will be used with real time images of the user as inputs. His secondary responsibility is to work the mechanical components of the stand, including the torque calculations, as well as purchasing new components, and combining components mechanically (such as the flap, the lip, and the shelf. This involved much assembly and general machining (e.g. drilling holes, sanding components).

Mary Rose’s primary responsibility was to work on the firmware that runs on the Arduino to receive information and commands from the computer and translate those requests into motor movement and linear actuation. This included facilitating Bluetooth communication from the laptop’s end and the Arduino’s end using a simple packet system that she designed. Her secondary responsibility was to design mechanical pieces of the stand including the motor mount, the wooden shelves, the flap and the lip, and manufacture the wooden pieces.

Olivia’s primary responsibility was to work on the height and angle adjustment software using OpenCV and Python for facial landmark detection and eye tracking. This included developing an algorithm to use the landmark data to tell the Arduino when to lower and raise. Since that task was more vital to the project, she focused mainly on that aspect. Her secondary responsibility was to work on the Python GUI that receives input from the user and sends it to the Python program or the Arduino as well as analyze posture data. Included in this responsibility was finely tuning the GUI for user experience, and determining the best way to provide clear instructions to the user on how to calibrate the product.

C. Bill of Materials and Budget

The Bill of Materials is located at the end of the report.

D. Risk Management

Over the course of our device’s design process, we experienced many different types of project risks. First, determining the height and angle that the laptop should be raised to is highly difficult, as it is dependent on the exact positions users calibrated the stand in. To fix this, we added in a calibration stage with instructions on how the user should position themselves while in front of the camera. We also encouraged them to hold their computer up, as this was found to have more accurate results through our testing.

Another difficulty we encountered was related to the posture calibration. Determining good vs. bad posture both for the user and while training the network was more difficult than we anticipated. We planned to mitigate this risk by instructing the user to have good posture during the calibration process, however this component was never fully integrated into our final GUI.

Another risk we faced was related to the platform-jack itself. We bought several different motors, each of which were incapable of lifting the platform jack, with a laptop on it with our other modifications. This problem persisted until we purchased the Nema 17 with the 5:1 gearbox. Overall, We mitigated this risk by researching gearboxes. However, because we would not have had time to alter our project after we this motor was delivered, we planned to start the platform jack at a higher height, which requires less torque to lift, so that we would have been able to continue using the original Nema 17 motor we purchased, that did not have a gearbox.

IV. Ethical Issues

This device was designed with public welfare in mind, as one of its main goals is to improve user health by encouraging users to use healthier practices when sitting in front of their laptop screen. This element of user interactions and user trust meant that we needed to be careful when testing the accuracy of our product. However, it is possible to have some edge cases, as our implementation relies heavily on OpenCV facial landmarks. People who have above average or below average sitting height may have difficulty calibrating the product, as there is a minimum and maximum height and angle it can adjust to. In these cases, the device will not calibrate, and the user is made aware of this. We took this approach rather than allow the user to use our stand at an invalid height. Furthermore, our stand is also manually adjustable, a setting which would allow these particular users to adjust our stand to a height they are comfortable with.

Furthermore, because as our stand rises it becomes less stable, we balanced it carefully so that the stand would not tip over at its maximum height, preventing user injury or damage to the user’s laptop.

V. Related Work

Interestingly, there are no automatic computer stands on the market, only manual ones that require the user to adjust for different table/chair heights and readjust every time the user sets their computer up. After working on the project, this makes sense because having motors to make the stand automatic makes the stand very heavy and large.

VI. Summary

Overall, our system was able to meet many of our design specifications, including the height and angle errors. However, we were unable to meet the requirement for calibration time. It was a functional design but lacked many of the convenience features that would make it a usable product (e.g. weight, size, elegance).

We learned tremendous amounts about mechanical engineering and how to integrate software, hardware, mechanicals, all done by different members.

References

[1] Thingiverse.com. "Platform Jack [Fully Assembled, No Supports] by Intentional3D." *Thingiverse*, www.thingiverse.com/thing:925556. Accessed 1 Mar. 2024.

[2] Thingiverse.com. "Motorized Platform Jack by Mitchamccaskill." *Thingiverse*, www.thingiverse.com/thing:6161381. Accessed 1 Mar. 2024.

[3] Singh, Riddhi Kumari. "Real-Time Pose Estimation from Video Using MediaPipe and Opencv in Python." *Medium*, Medium, 26 Apr. 2023, medium.com/@riddhisi238/real-time-pose-estimation-from-video-using-mediapipe-and-opencv-in-python-20f9f19c77a6.

[4] "InsigniaTM." *BestBuy.Com*, www.bestbuy.com/site/best-buy-brands/insignia/pcmcat159800050007.c?id=pcmcat159800050007. Accessed 1 Mar. 2024.

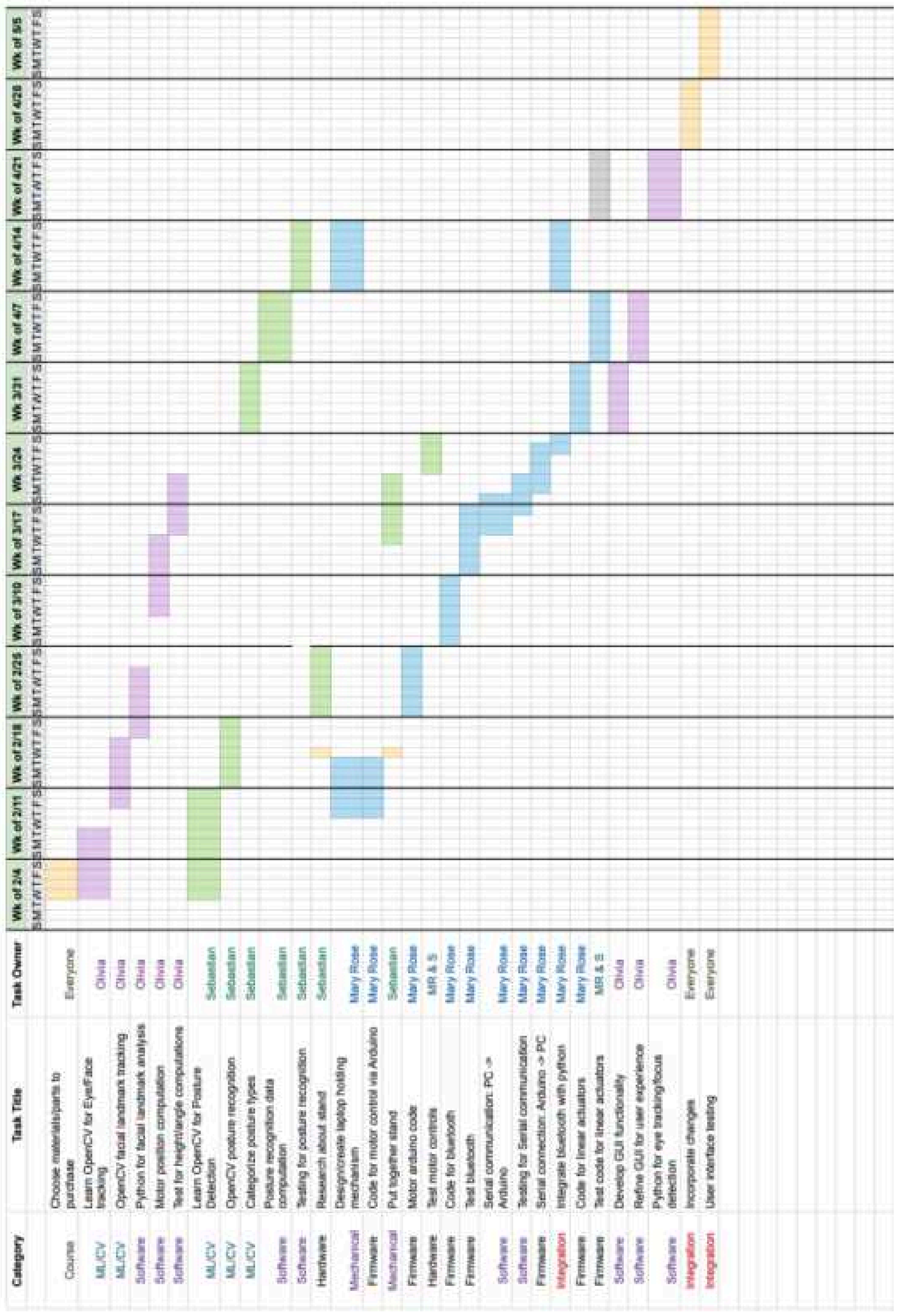
[5] "Power Screws - Torque to Force Relationships in Just over 10 Minutes!" *YouTube*, YouTube, 12 Oct. 2020, www.youtube.com/watch?v=BstZUC4tcOA.

[6] "Postural Awareness." *Postural Awareness – Stanford Environmental Health & Safety*, ehs.stanford.edu/subtopic/postural-awareness. Accessed 4 May 2024.

Item	Cost ($)
Screws: $7	7
3D printed jack: $30	30
Lin actuator 1: $35	35
Lin actuator 2: $30	30
Arduino Rev4: free	0
Arduino Rev3: free	0
Cable USB C: free	0
Stepper Motor: $13	13
Motor shield: $7.50	7.5
149 grams PLA for motor mount: $75	75
Metal platform jack: $28	28
Hardware (nuts and bolts): free	0
Original Motors: $20	20
9V Barrel jack power supply	10
Adafruit motor shield	21
headers	8
12 barrel jack	8
barrel jack connector	12
lin actuator 2	35
big easy driver	21.5
misc hardware	33.65
new nema 17	40
coupler	10
3d printed coupler	12.5
l298n motor driver	7
laser cut	10
AA batteries	5
Total	479.15
Remaining:	120.85

Table 1: Bill of Materials

Figure 15: Final Gantt Schedule:



Category	Task Title	Task Owner
Course	Choose materials/parts to purchase	Everyone
ML/CV	Learn OpenCV for Eye/Face tracking	Olivia
ML/CV	OpenCV facial landmark tracking	Olivia
Software	Python for facial landmark analysis	Olivia
Software	Motor position computation	Olivia
Software	Test for height/angle computations	Olivia
ML/CV	Learn OpenCV for Posture Detection	Sebastian
ML/CV	OpenCV posture recognition	Sebastian
ML/CV	Categorize posture types	Sebastian
Software	Posture recognition data computation	Sebastian
Software	Testing for posture recognition	Sebastian
Hardware	Research about stand	Sebastian
Mechanical	Design/create laptop holding mechanism	Mary Rose
Firmware	Code for motor control via Arduino	Mary Rose
Mechanical	Put together stand	Sebastian
Firmware	Motor arduino code	Mary Rose
Hardware	Test motor controls	MR & S
Firmware	Code for bluetooth	Mary Rose
Firmware	Test bluetooth	Mary Rose
Software	Serial communication: PC -> Arduino	Mary Rose
Software	Testing for Serial communication	Mary Rose
Firmware	Serial connection: Arduino -> PC	Mary Rose
Integration	Integrate bluetooth with python	Mary Rose
Firmware	Code for linear actuators	Mary Rose
Firmware	Test code for linear actuators	MR & S
Software	Develop GUI functionality	Olivia
Software	Refine GUI for user experience	Olivia
Software	Python for eye tracking/focus detection	Olivia
Integration	Incorporate changes	Everyone
Integration	User interface testing	Everyone