ECE 445

Spring 2024

Final Report

An Auto-Hand Chasing Lamp With Hand Gesture Control

Team 49
Feiyang Liu (feiyang5)
Yiyan Zhang (yiyanz3)
Jincheng Yu (jy54)

TA: Luoyan Li
April 30th, 2024

Abstract

This document is team 49's final report for ECE445 Senior Design. Team 49's project is a smart lamp with hand movement tracking and gesture control. The report covered the purpose, design, testing, and final build of the smart lamp. Readers can refer to the graphs and formulas shown in the corresponding section to understand the building process of the project. The conclusion at the end of the report summarized our achievements and future recommendations for the project.

Contents

1. Introduction

1.1 Problem

In industrial workspaces, inadequate lighting, especially from shadows, hinders precision tasks like soldering, impacting work quality and efficiency. Traditional lighting lacks flexibility and fails to meet specific task needs, leading to a demand for a smart, adaptable desk lamp.

1.2 Solution

Our design is a desk lamp equipped with the capability to autonomously track the user's hand/handlike object movements, thereby eliminating the common issue of shadows obscuring the work area. This is achieved through a combination of a camera, several servo motors, and a flexible mechanical arm, all of which work in tandem to reposition the lamp in real-time

The system utilizes gesture recognition technology, powered by an ESP32 module, to interpret hand gestures. This allows users to control the lamp's functionalities—ranging from adjusting brightness and color temperature to switching the lamp on and off. Furthermore, the lamp is capable of executing more advanced commands, such as controlling the RGB matrix to generate light to create atmosphere.

1.3 Functionality

Our project achieved the following high-level requirements and passed verification:

i. **Autonomous Tracking Accuracy:** The smart desk lamp can accurately detect and follow a human hand's movement when a hand is placed at least 15cm away from the camera. The lamp should differentiate human hands from objects and not move when a pen is placed in front of the camera.

ii. **Gesture Recognition Responsiveness:** The user can show the following hand gestures to control the light:

    1. Pointing up or down to change light intensity

2. Control lamp movement by pointing left or right
3. Activate tracking mode by showing a fist

iii. **Light Adjustment Range:** The lamp can change between 2 brightness and 2 colors when different hand gestures are captured through the camera.

1.4 Subsystem Overview

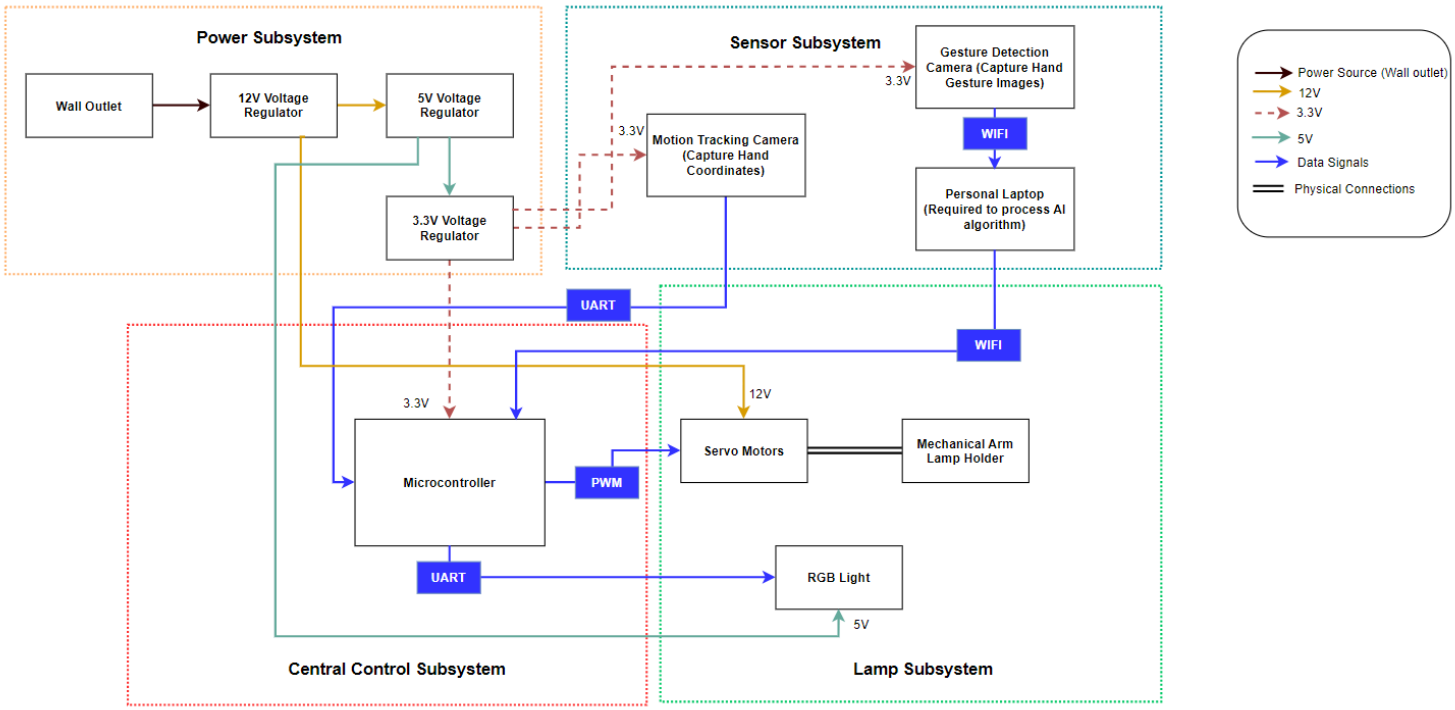

Figure 1: Auto-Hand Chasing Lamp's Block Diagram

Our initial block diagram included four subsystems: power, sensor, central control, and lamp systems. The function of each subsystem is described in the next section.

1.4.1 Sensor Subsystem

The sensor subsystem is divided into two systems, the first is the object detection system, we will use this to capture images in ESP32S3Cam, run the pre-trained object detection AI locally in ESP32S3, and pass the offset to the central control system. Eventually, the model may be trained using the model hand rather than the real hand according to various limitations.

In addition, the gesture system will use another ESP32S3cam for image capture. If the accuracy of the training model is too low, the picture will be transmitted to the computer through WIFI, and the command will be sent to the central control system after the picture is processed on the computer.

1.4.2 Central Control Subsystem

This system contains a microcontroller that integrates the ESP32 module and necessary I/O modules. It needs to process the input data from the sensor system like the offset of the servo and the data into the RGB light. It can also communicate remotely with a computer to control specific programs through WIFI. The lamp subsystem will receive outputs from the microcontroller and change mechanical arm movement and light bulb activities based on the detected hand gestures. Detectable gestures will include fists and raising, directions of thumbs, and other interesting gestures. The gestures will be able to turn on/off, increase/decrease brightness, and increase/decrease light temperature and other features.

1.4.3 Lamp Subsystem

The lamp subsystem makes up the physical lamp that will be used to track the user's hand/ hand-like object movement. It contains two key parts, which are a mechanical arm lamp holder and a light bulb. Three servo motors and a linear potentiometer are used to control the rotation and location of the lamp. The lighting bulb should be adjustable in terms of color temperature and brightness through instructions from the central control system.

1.4.4 Power Subsystem

The power subsystem provides the power needed for the sensors, microcontroller, and lamp system. We use a 12V voltage adapter to power the core lamp system. A separate 3.3V voltage adapter will be used to power smaller components including the camera sensor and microcontroller. The power needs to be sufficient to support the three servo motors in moving the physical-mechanical arms, and a wall outlet would be used as the primary energy source.

2. Design

2.1 Visual Design

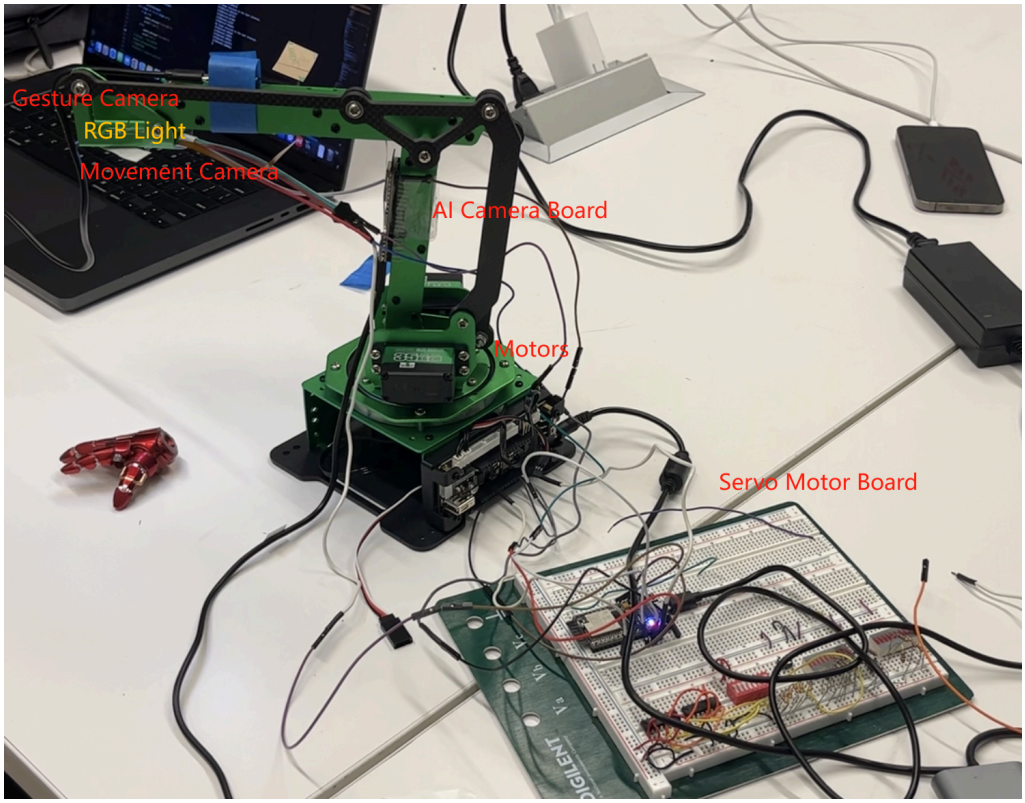

Figure 2: Final Product's Physical Design

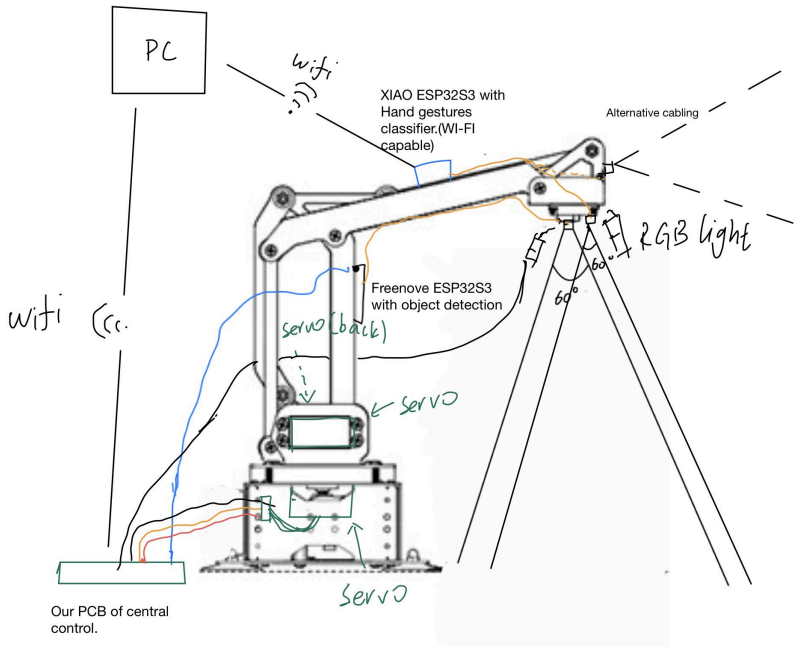

Figure 3: Smart Lamp's Design Layout

2.2 Hardware Design Description & Justification

2.2.1 PCB Diagrams & Schematics

As shown in figures 1 to 3, the hardware design for our project were focused on the PCB and mechanical arm design. Our PCB design included ESP32 microcontroller, programming circuits, power adapter, switches, and motor connectors that are used to fulfill the requirements for each subsystem. The detailed schematic and layout are shown in the figures 4 and 5. Up on completing the PCB, we could be able to perform image processing on ESP32 and output motor signals to control the mechanical lamp holder. However, there were design and assembly issues that caused the PCB to receive incorrect power and resulted in an unprogrammable PCB. We eventually decided to replace the PCB with two ESP32 development boards. By having one board controlling the motors and the other performing image processing, we enabled communication between the microcontroller and two front cameras in the front of the mechanical arm. The replaced ESP32 development boards passed all verifications and worked as we

expected.

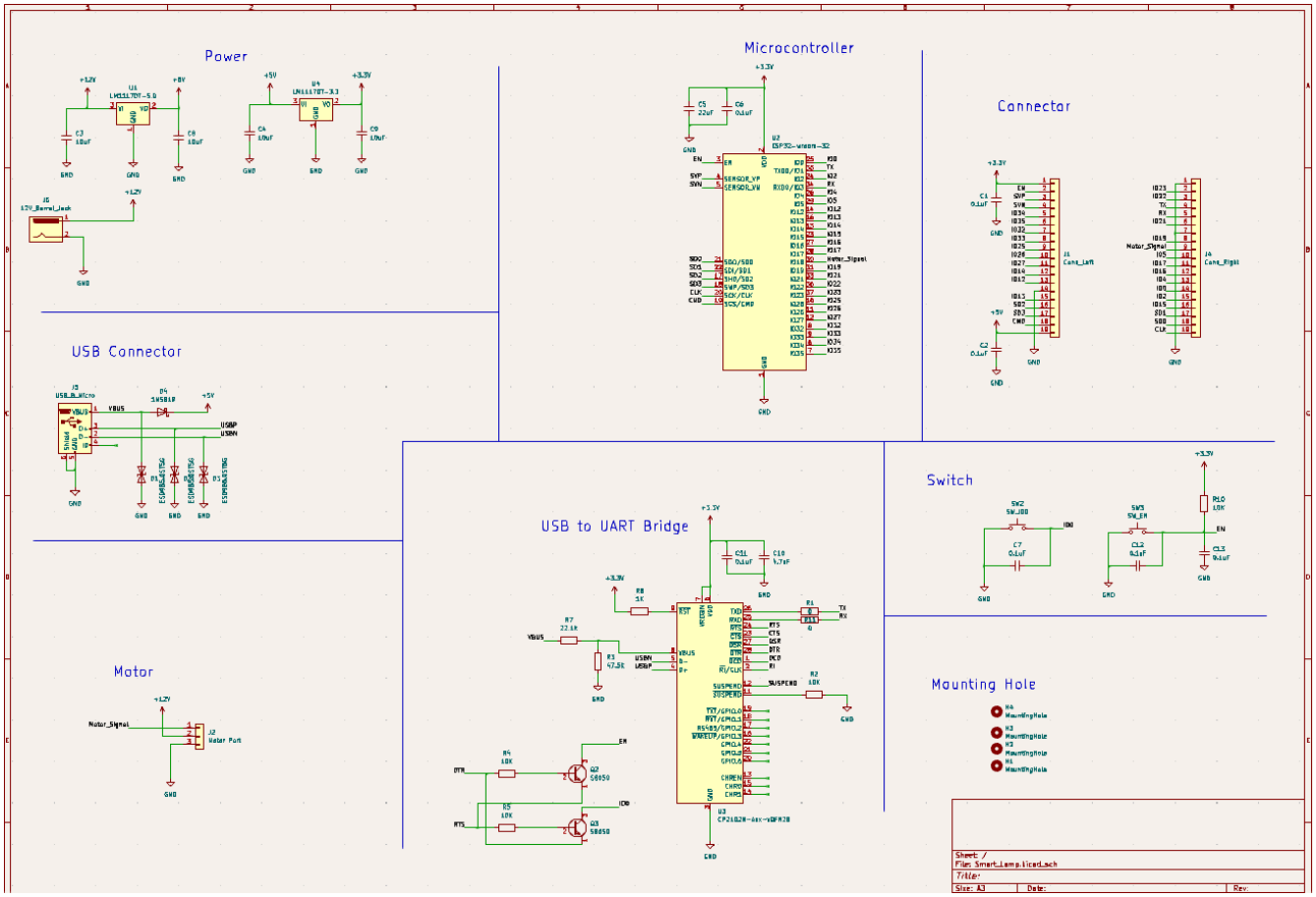

Figure 4: Original PCB Schematic

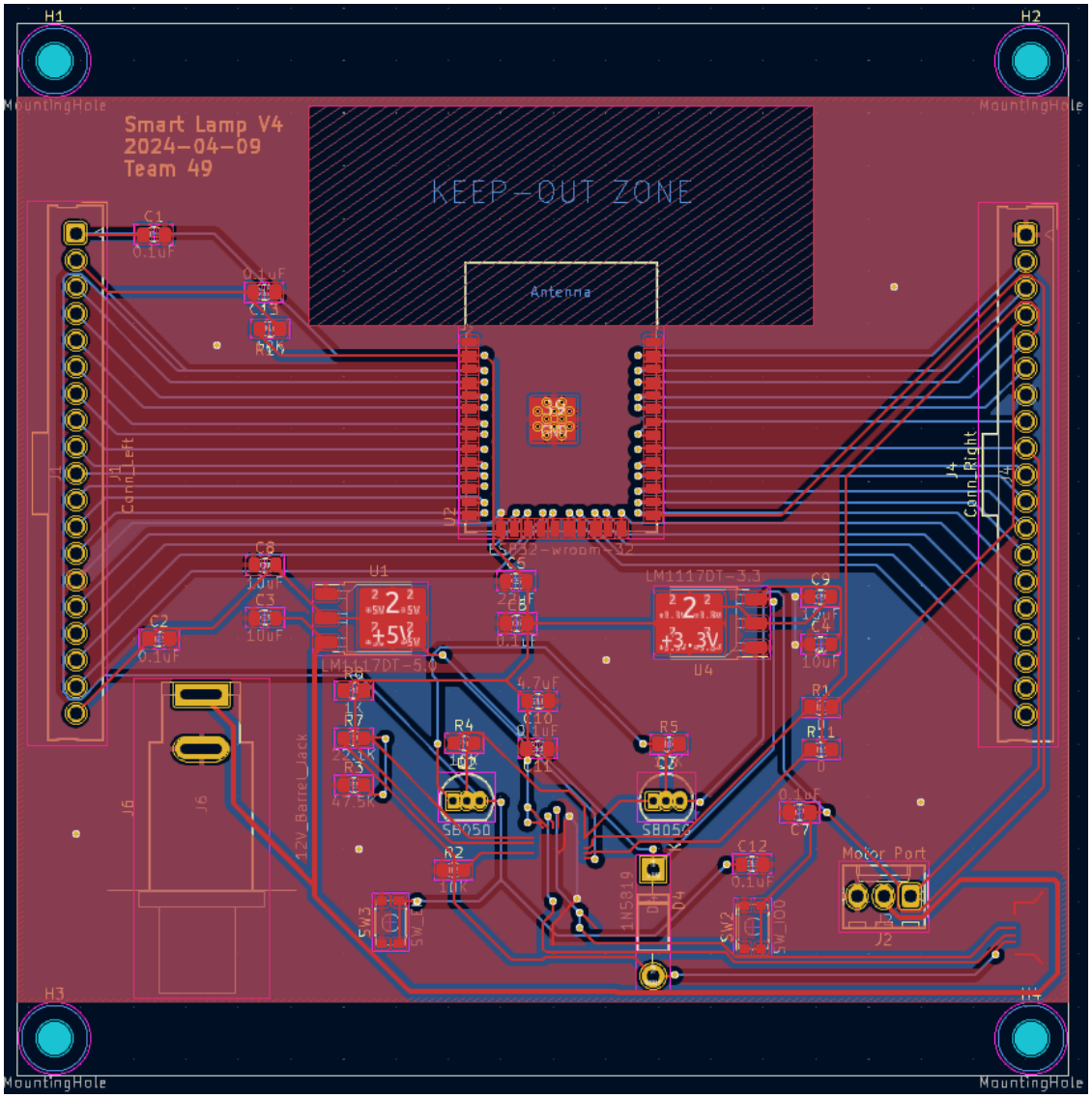

Figure 5: Original PCB Layout

2.2.2 Mechanical Arm

In the development of a precision-controlled mechanical arm for a desk lamp, three hts35h servo motors were employed to meticulously guide the arm's movement in the x, y, and z directions. Initially, the focus was on ensuring the correct operation of these servo motors. Subsequently, the coordination of their movements became paramount due to the arm's structural design, where excessive extension could notably lower the LED light's position, rendering some areas ineffective for illumination. Through rigorous testing and evaluation, a feasible range of motion was determined for the arm, which was calibrated on a scale from 400 to 750 out of 0-1000.

Furthermore, an interface was devised that translates the movements of two servos into a polar coordinate system: **`void LobotSerialServoControl::moveToAngleAndLength(float angle, float length).`** This system utilizes angles ranging from 0 to 240 degrees and distances extending up to 1000 units, facilitating precise and efficient control over the LED's positioning. By inputting specific coordinates, the light can be directed accurately to any desired location, thereby enhancing the functionality of the lamp. This integration addresses the initial challenge of converting the point needed to two motor's movement every time.

The greatest challenge lies in calculating the angle offset, and the following figure provides a clear explanation.

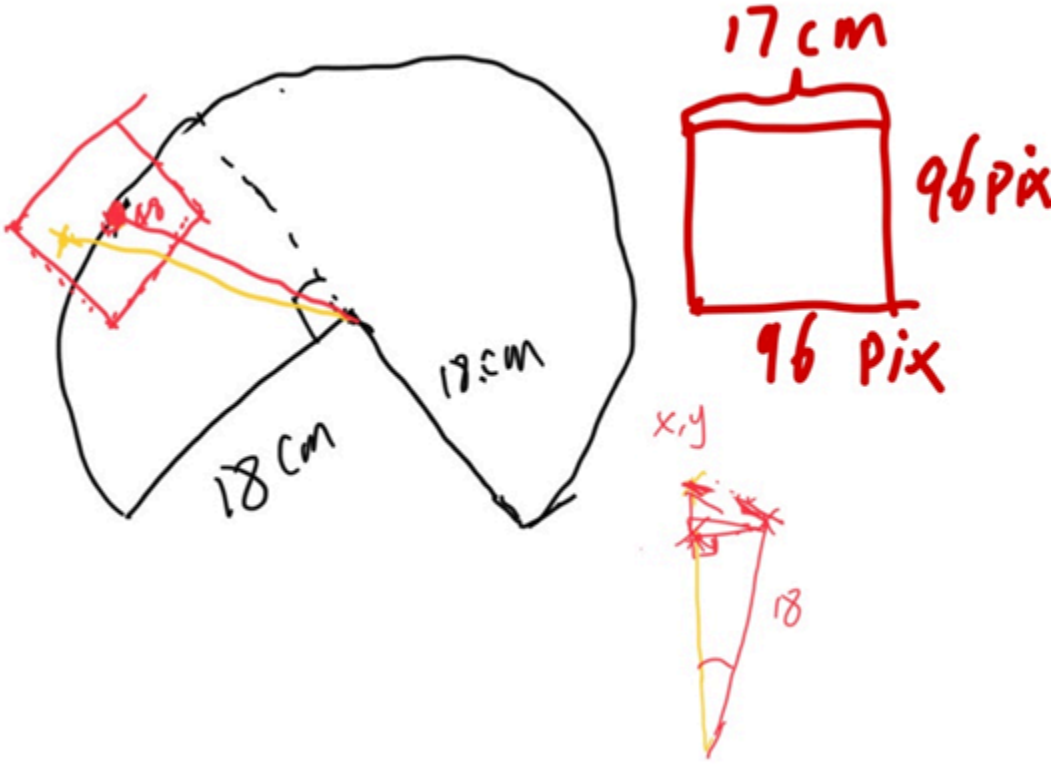

Figure 6: Hand Angle Draft

The yellow lines and x's indicate the position of the model's hand, the 270-degree fan is the Angle that the motor can support, and the red box is the area that the camera can capture. According to the 96*96 pictures I streamed out, it is about 17cm. Therefore, I calculated that the distance between each pixel is 17/96 cm. Currently, we only consider the rotation Angle of the main servo. I used the length of the robotic arm (18cm) and found the Angle through the following formula.

$$Angle_{offset} = (\frac{(x-48)*\frac{17}{96}}{18}) \qquad (1)$$

From this, our robotic arm can successfully find the Angle and keep track.

2.3 Software Design Description & Justification - Object Detection

2.3.1 Object Detection

Our first task was to choose the right development platform for the team and the AI model that could be deployed on that platform. First, we chose the ESP32 series from the beginning. Especially after learning about a lot of AI projects based on ESP32S3, we found that this chip can largely achieve our requirements. Then, finding small AI models that can fit into them is the next challenge.

Our team looked for various possible solutions on the web, such as converting the TensorFlow model to the ONNX model based on the ESP-DL deep learning library. This was our initial direction, however we ran into deployment issues with the Freenove ESP32S3Cam we purchased. At the same time there are some necessary dependencies that we can never install correctly. So we have to give up. The next attempt is to stream the captured images to the computer for processing via MicroPython or ESP32 webserver. we completed Wifi-based image streaming, but it was abandoned as a last resort because it went against our desire to run as locally as possible. As we continued to read about the ESP32S3, we noticed the FOMO model from Edge Impulse. It seems to do much of what we need.

The FOMO model[1] enables object detection models to be run on constrained devices and brings real-time object detection, tracking, and counting to microcontrollers. Most importantly, it is able to run smoothly on very small amounts of RAM. This is what we need.

2.3.2 Model training and deployment

Fortunately, we found full instructions and a relatively easy training environment on the edge impulse website. we need to upload the object that needs to be identified and add the corresponding label. Then fill in the corresponding image parameters (height and width are set to 96 for ESP32S3 processing). After that, background is needed as a control group to better train the model.

To simulate the true image quality of the ESP32S3cam as much as possible, we deployed a web server on the esp32s3 to capture the images and download them to the computer. However, because the height of the robot arm we purchased is relatively low, and the FOV of the ov2640 is not excellent. It is difficult for us to fully slap the palm into the camera at a normal height. We need to use a set of Iron Man anti-Hulk armor model hands as a training object.

Figure 7: Pictures Taken by OV2640 Camera

First, we uploaded the picture data to the website and manually typed most of the pictures with the correct label. We can then output the eigenvalues of all images for subsequent convolution. We use RGB as the Color depth in the hope that the red color of the hand model can be perfectly distinguished from the background and increase the accuracy. Then, follow the system to download the Lib package in the form of Arduino Lib, deploy it into Arduino, and finally deploy it on ESP32S3.

However, when we first practiced it, because the background was all white desktops and the light was not enough, the accuracy was quite low on other colored desktops, and it also had a high probability of recognizing the shadow as a hand. This is hard for us to accept. First, we try to combine the three identifications, that is, record the data of the three identifications and compare them before output, and output if the three identifications are close to each other. But this still cannot meet our requirements, and will slow down our detection speed to a certain extent.

So, we did a second and third model training, adding black and well-lit datasets.

Figure 8: Graph of Features

In this image, each cluster represents an image, such as the manipulator on the far left, which has a mostly black background. The good news is that each cluster is relatively concentrated.

```
if (fomo.first.proba > 0.6  ) {
    Serial.printf(
        "Found %s at (x = %d, y = %d) (size %d x %d). "
        "Proba is %.2f\n",
        fomo.first.label,
        fomo.first.x,
        fomo.first.y,
        fomo.first.width,
        fomo.first.height,
        fomo.first.proba
    );
    mySerial.print("x:");
    mySerial.print(fomo.first.x);
    mySerial.print(" y:");
    mySerial.println(fomo.first.y);
    delay(1000);
}
```

Figure 9: Code Snippet

This method was used to send the detected coordinates to the GPIO-2 of the master via the TX pin of the ESP32S3 in the loop.

2.4 Software Design Description & Justification - Hand Gesture Detection

We want gesture recognition to be able to control the on/off of the RGB strip, the color, to be able to switch the tracking mode of the lamp on and off, and to be able to control the position of the robotic arm when the tracking mode is off.

2.4.1 Model and Development Platform Selection

I originally wanted to be able to shoot and define a simple gesture model in a similar way to objection detection. However, this does not seem to be the same thing. Because most of the features of the hand are similar when it comes to gestures. we took a lot of shots for palm and fist and tried to train and deploy on the ESP32S3 in a similar way. However, although the model can be successfully deployed and can distinguish between the fist and the palm in some cases, the

accuracy is very low if the background is different or the position of the hand is slightly changed. This is hard for us to accept.

We started thinking about using more mature datasets, such as an ONNX model based on the Kaggle gesture recognition dataset. However, the models in these datasets are very large, taking about 20 seconds for the ESP32S3 to process an image. This is not good news either.

At the same time, what makes us most hesitant is that most of the data sets support only partial gestures, such as the palm, fist, and index finger. That's still not enough for us. We want to be able to reverse control the mainframe and the computer through gestures. This requires us to be able to implement gestures that are less common, but more intuitive.

Finally, we chose to send the picture to the IP address of the computer at a rate of once per second through the Camera and Socket in Micropython, and then the computer monitored 0.0.0.0:9090, parsed the picture through OpenCV and sent it to MediaPipe for analysis.

The reason for using MediaPipe is that it can analyze the Landmark of each joint for hand photos with relative accuracy and provide corresponding coordinates. This allows us to create a lot of possibilities.

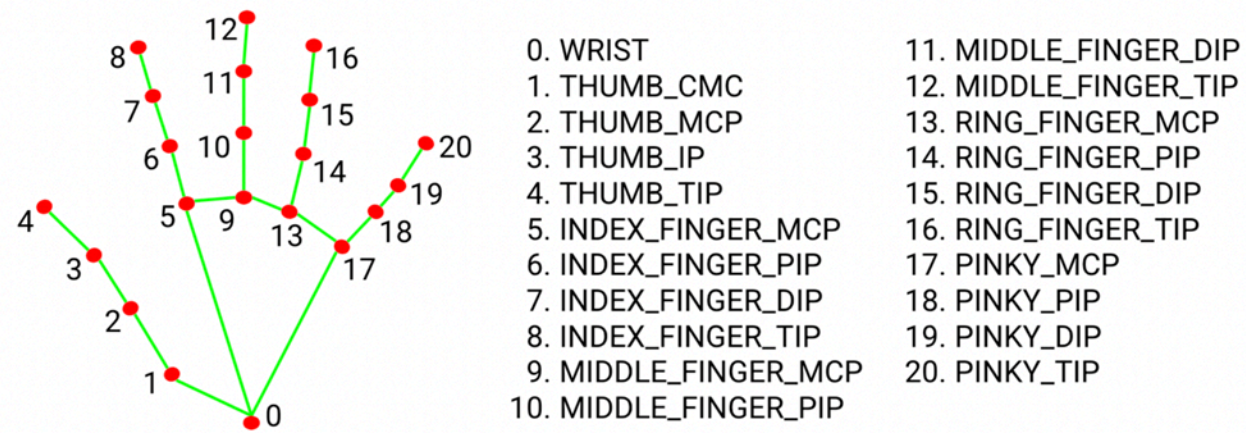

Figure 10: Hand Landmarks by MediaPipe [2]

2.4.2 Gesture analysis

First we want to be able to tell if each finger is in an active state. We judged the Euclidean distance between the coordinates of the fingertips and the coordinates of the wrist. If it is greater than the distance between the Previous Joint and the coordinates of the wrist and the Euclidean distance, it means that the fingers have been extended. For the index, middle, ring and pinky fingers, this algorithm is sufficient. However, this is not true for the thumb, because the distance between the fingertip and the wrist in the contracted state of the thumb is sometimes still longer than the distance between the previous joint and the wrist. At the same time, because of the different posture of the fist, sometimes the change in the distance from the thumb to the wrist is not obvious.

So, we try to calculate the slope of the previous joint to the fingertip and the wrist to the previous joint and judge the Angle between them to determine whether the thumb is bent. This allows me to judge the status of thumb.

Then we have a basic framework that we can use to implement gestures. First, we added thumb-based directions when only the thumb was active. For example, with a thumbs-up gesture, we can calculate the direction of my thumb by comparing the x coordinates and y coordinates of my thumb and wrist and output up, down, left and right. we plan to use this gesture to control motor movement when tracking mode is turned off.

At the same time, we created palm (all fingers active), fist (all fingers deactive) and so on. In the future, these gestures could also be used to control motors. These gestures will eventually output a string, which is sent over WIFI to the master ESP32 for control.

2.5 LED

In the design and implementation of the Smart LED Control System for Gesture-Responsive Lighting, the WS2812B LED strip was selected for its balance of cost-effectiveness and capability to meet the project's technical requirements. This type of LED strip is advantageous as it requires only power and a data input, thereby simplifying the circuitry design. A matrix of four

WS2812B LEDs was configured into a square layout and soldered to ensure stable and consistent illumination, as illustrated in figure 12.

The control of the LEDs was managed using the Adafruit NeoPixel library, renowned for its precise tuning of color (RGB) and light intensity. The choice of this library was driven by its extensive support and ease of programming LED behaviors, which are crucial for achieving desired lighting effects.

The LED matrix was integrated with the development board and configured with WiFi connectivity through the software, employing the HTTP protocol for communication. It allows the LEDs to wirelessly receive commands such as turning on or off, adjusting brightness, or changing colors.

The incorporation of WiFi and HTTP protocols significantly enhanced the system's adaptability and enabled real-time interaction with the LEDs. This setup is pivotal for the system's readiness to integrate with gesture recognition technologies, aiming to provide a responsive and user-friendly lighting experience that dynamically adjusts to users' needs based on their inputs.

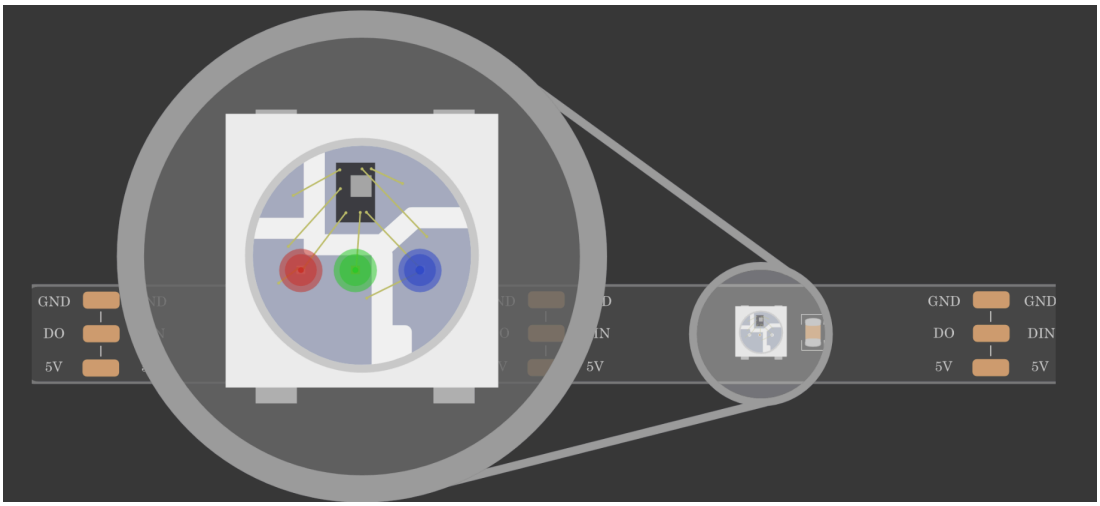

Figure 11: WS2812B

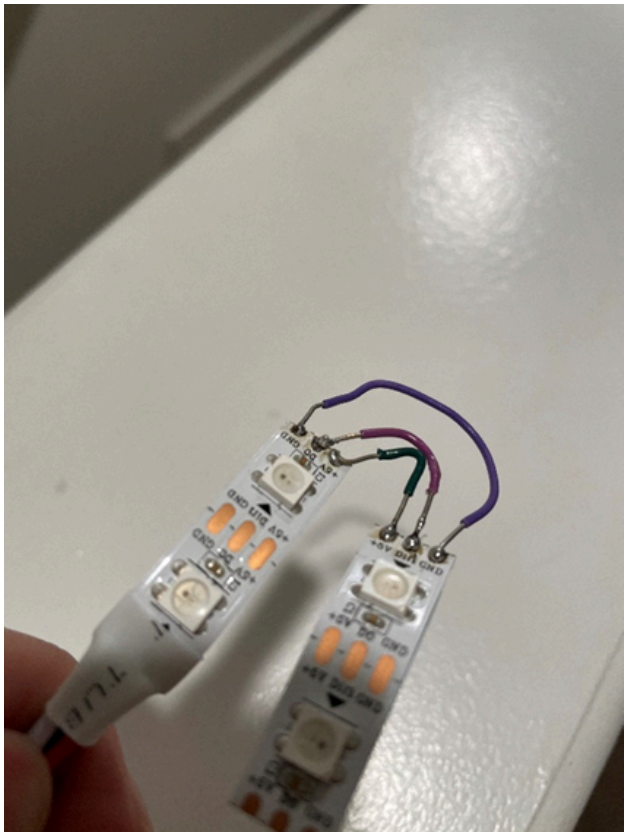

Figure 12: Led Matrix

2.6 Design Alternatives

Several design changes were made to the initial design. The first change was replacing the PCB with two development boards. During our testing phase, we realized that the 5V voltage adapter failed to drop the 12V power input to 5V. The multimeter showed that 10V entered the ESP32 microcontroller's power pin, which resulted in the burning of our initial microcontroller. We first ordered other PCBs and separated the power modules from microcontroller to ensure that power modules could be tested separately. This design however caused our PCB to end up unprogrammable due to potential soldering and designing mistakes. As a result, we decided to replace the custom PCB with ESP32 development boards so that we could still program the microcontroller and output the motor signals. This solution worked and allowed us to continue implementing the gesture control and object detection features.

3 Cost Analysis & Schedule

The cost analysis for this project is primarily divided into two main components: labor costs and parts costs. We have estimated the hourly wage for each project partner at $56, based on market research for graduates from Electrical and Computer Engineering (ECE) at the University of Illinois. Below is a detailed cost analysis:

3.1 Labour Costs

- The assumed hourly wage for each partner is $56.
- The total hours each person should use for the project is 20 hours/person, And we assume to finish the project in 8 weeks.
-
- The formula for total labor cost is: `56 x 2.5 x 20 x 3 x 8 = 67200`.

3.2 Parts Cost

The following table lists all the parts required for the project, including description, manufacturer, part number, quantity, and cost, along with estimated machine shop labor hours needed to complete the project.

Item	Manufacture	Part Number	Quantity	Unit Price(USD)	Total Price(USD)
ESP32-S3-CAM	Freenove	-	1	20	20
HTS-35H	Hiwonder	-	3	23	69
Robot Arm kit	Hiwonder	-	1	50	50
ESP32-S3-CAM	XIAO	-	1	23	23
ESP-WROOM-32	Espressif	N16CT-ND	1	3	3
Power Jack Connector	CUI Devices	CP-037A-ND	1	0.59	0.59
5V Linear	Texas	LM1117DT-5.0	1	1.81	1.81

Item	Manufacture	Part Number	Quantity	Unit Price(USD)	Total Price(USD)
ESP32-S3-CAM	Freenove	-	1	20	20
HTS-35H	Hiwonder	-	3	23	69
Robot Arm kit	Hiwonder	-	1	50	50
Regulator	Instruments				
USB to UART Bridge	Silicone Labs	336-5889-ND	1	4.66	4.66
NPN Transistor	Onsemi	SS8050CBU-ND	2	0.41	0.82
3.3V Linear Regulator	Texas Instruments	LM1117DT-3.3	1	1.81	1.81
Single Diode	STMicroelectronics	497-6610-1-ND	1	0.38	0.38
47.5K Ohm Resistor	YAGEO	311-47.5KCRCT-ND	1	0.1	0.1
22.1 Ohm Resistor	YAGEO	311-22.1KCRCT-ND	1	0.1	0.1
TVS Diode	Onsemi	ESD9B5.0ST5G	3	0.17	0.51
Tacile Switch	C&K	401-1426-1-ND	2	0.57	1.14
22UF Capacitor	Samsung	1276-1100-1-ND	1	0.13	0.13
0 Ohm Resistor	ECE Supply Shop	-	2	0	0
1k Ohm Resistor	ECE Supply Shop	-	1	0	0
10k Ohm Resistor	ECE Supply Shop	-	4	0	0
0.1uF Capacitor	ECE Supply Shop	-	7	0	0

10 uF Capacitor	ECE Supply Shop	-	4	0	0
4.7 uF Capacitor	ECE Supply Shop	-	1	0	0
Micro USB-B Connector	ECE Supply Shop	-	1	0	0
3 Position Header Connector	ECE Supply Shop	-	1	0	0
18 Position Header Connector	ECE Supply Shop	-	2	0	0
Total					177.05

Table 1: Costs of Components

3.3 Grand total

The grand total will include the sum of the labor costs and the parts costs. The specific labor cost will need to be calculated based on the actual hours worked by each partner. The total parts cost is $177.05. All labor costs amounted to $67,200.At the same time, we also need to consider the expenditure of PCB board design completion. This will be updated after we decide on the design, tentatively set at $20.
The total cost is therefore approximately 67,400.

3.4 Schedule

Week 1 (Feb 22 - Feb 28): Initial Planning and Architecture

Task: Select architecture and finalize project scope.

Members Involved: Jincheng (Lead), Yiyan, Feiyang.

Deliverables: Project scope and architecture plan

Week 2 (Mar 1 - Mar 7): Detailed Design Phase

Task: Design sensor subsystem; Yiyan begins ESP32-CAM integration.

Members Involved: Feiyang (Lead on sensor design), Yiyan (ESP32-CAM), Jincheng.

Deliverables: Sensor subsystem design, initial ESP32-CAM setup.

Week 3 (Mar 8 - Mar 14): Control and Lamp Design

Task: Design central control subsystem; start lamp subsystem design.

Members Involved: Jincheng (Lead on central control), Feiyang (Lead on lamp design), Yiyan.

Deliverables: Control and lamp subsystem designs.

Week 4 (Mar 15 - Mar 21): Parts Procurement and Testing

Task: Order and test individual components; finalize ESP32-CAM integration.

Members Involved: Yiyan (Lead on ESP32 and testing), Jincheng, Feiyang.

Deliverables: Components ordered, initial component tests, ESP32 integration.

Week 5 (Mar 22 - Mar 28): Assembly of Subsystems

Task: Assemble sensor and central control subsystems.

Members Involved: Feiyang (Lead on assembly), Jincheng, Yiyan.

Deliverables: Assembled sensor and control subsystems.

Week 6 (Mar 29 - Apr 4): Lamp Assembly and System Integration

Task: Assemble lamp subsystem and integrate with control system.

Members Involved: Jincheng (Lead on lamp assembly and integration), Feiyang, Yiyan.

Deliverables: Lamp assembly, integrated system.

Week 7 (Apr 5 - Apr 11): System Refinement and Demo Preparation

Task: Refine integrated system and prepare for mock demo.

Members Involved: Feiyang (Lead on refinement), Jincheng, Yiyan.

Deliverables: Refined system, prepared demo.

Week 8 (Apr 12 - Apr 15): Final Testing and Demo Execution

Task: Conduct final system testing and execute mock demo.

Members Involved: Yiyan (Lead on final testing), Jincheng, Feiyang.

Deliverables: Final testing completion, successful mock demo.

4 Requirements & Verification

4.1 Subsystem Verification

4.1.1 Sensor Subsystem

In the sensor subsystem, we employed two camera sensor, specifically the ESP32S3-CAM, which is capable of capturing images accurately. The hand gesture tracking and hand movement tracking were conducted directly on the ESP32S3-CAM, leveraging its onboard processing capabilities to facilitate local model deployment. We conducted input power testing on the two camera sensors and verified the correct 5V power input. Image processing was conducted on PC

and demonstrated correct detection of hand movement and hand gestures. In figure 11, a performance score of 98.8% is achieved, which indicated that our sensor detection and image processing were successful. We also set up the motion tracking camera to take 96 x 96 pixels pictures and set the center coordinate to 48 x 48 pixels to compute offsets with the new coordinate of hands. This allowed us to receive the correct coordinate of hand location and track its movement through the camera sensor. The corresponding R&V table is shown as Table 2 in Appendix A.

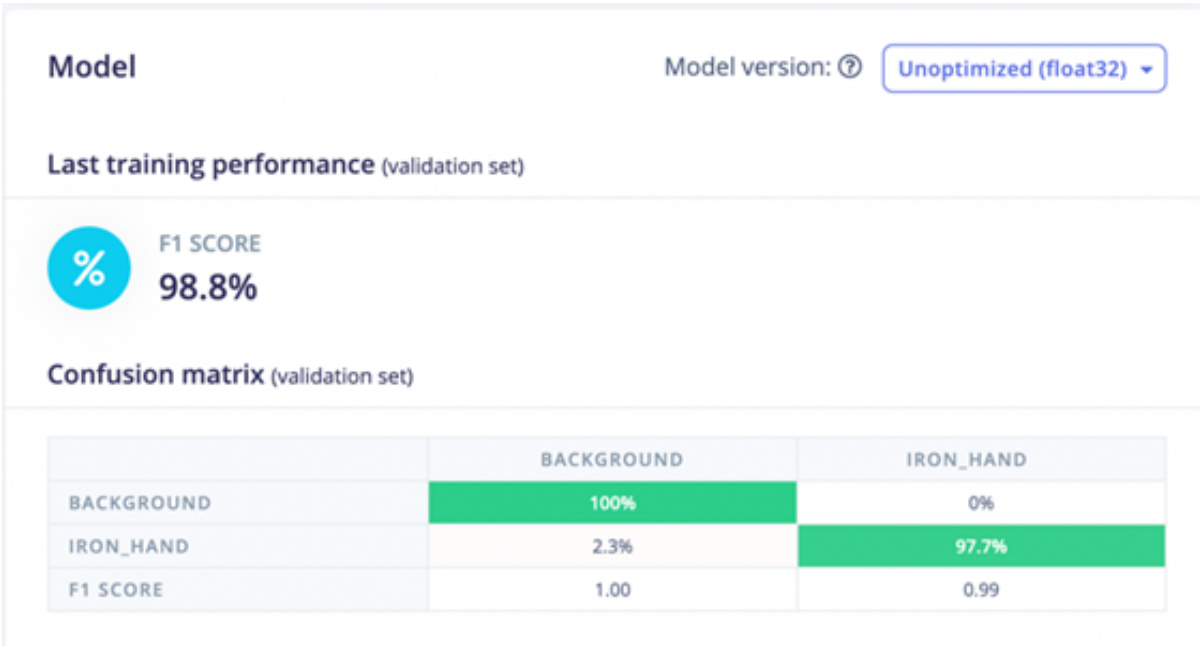

Figure 13: Image Processing Performance Metrics

4.1.2 Central Control Subsystem

A selected microcontroller, ESP-WROOM-32 Microcontroller, was ysed to process captured images from the ESP32 sensor and connect with the lamp system to determine mechanical arm movement and lamp lighting. However, we did encounter issues with verification of the central control microcontroller. After we finished soldering of custom PCB, we set up and programmed the ESP-WROOM-32 Microcontroller through Arduino IDE. Then we connected ADALM2000 between the ports used to transfer PWM and UART data to verify if signals are sent to sensor

and lamp subsystems. However, the ESP32 microcontroller ended up to be not programmable. As a result, we had to replace the central control with ESP32 Development Boards. After the replacement, we performed the same verification steps again and could successfully program the microcontroller. Another requirement was reducing the delay between image capturing and mechanical arm movement to under 5 seconds. This was verified using timer, and the average delay was under 5 seconds which matches our requirement. Table 3 in Appendix A shows the detailed requirements.

Figure 14: Delay Timer Results

4.1.3 Lamp Subsystem

The lamp system ensures the movement of the light bulb and controllable lighting. The lamp uses a mechanical arm to move the light bulb to track hand movement. We measured voltage across three HTS35h servo motors and were able to activate the motors with 12V. Figure 15 shows the rotational angles and distance that were verified by measuring The mechanical arm was able to move at a total angle of 240 degrees and extend for 17cm. The resulting measurement fulfills our requirements and allows the light to move in x and y directions.

Multimeter tests were also performed on the RGB light strip to ensure the light is receiving stable 5V power input. Table 4 in Appendix A shows detailed requirements.

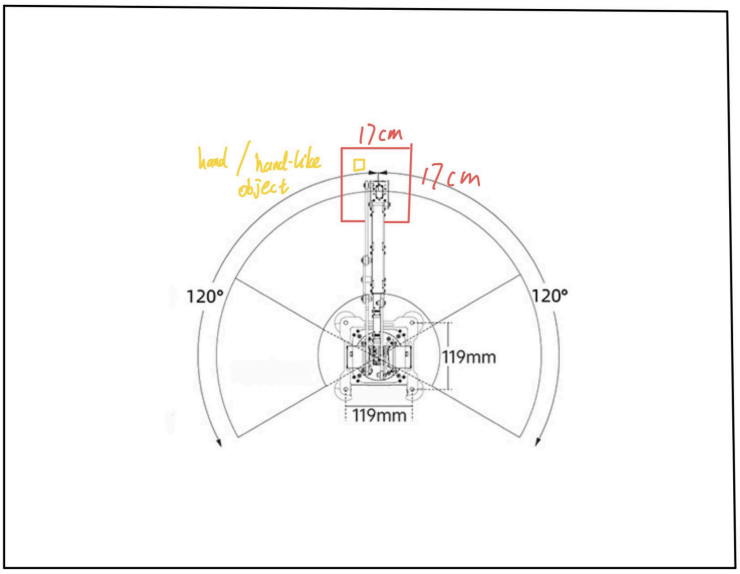

Figure 15: Mechanical Arm's Rotational Angle and Distance

4.1.4 Power Subsystem

The power system supports the energy needed for the lamp, sensor, and central control subsystems. A wall outlet will be the primary energy source. Extra 12V, 5V, and 3.3V voltage adapters are used to power up separate components in the design. We used a multimeter on the voltage regulator output to verify if it supports the corresponding output. Our final design decided to use an external power adapter since there were issues with the PCB design, and the multimeter tests showed that our external power source could support 12V +/- 5%, 5V +/- 5%, 3.3V +/- 5% power supply which satisfied our needs. Table 5 shows the detailed verification steps.

4.2 Tolerance Analysis

In our design, we will be utilizing the ESP32 controller, which has a minimum load requirement

of 0.5A. Additionally, we will also be using three HTS-35H motors with 35kg high torque to control the robotic arm.

$$\tau = F \cdot r \cdot \sin\theta$$

Referring to the torque formula, the entire lamp would weigh about 2kg. Assuming a rotation radius of 20cm and an angle of 270 degrees. We calculate that the minimum torque required to move the lamp would be $\tau = 9.81*2*0.2*\sin(90) = 3.5$ N-M. HTS-35H motor could support torque greater than 5 N-M and is expected to meet our needs.

We have also planned to use a power supply capable of providing 2A of current, so we can safely utilize this power source. Furthermore, we need to ensure that the cameras' field of view (FOV) can capture as much of the scene as possible to avoid losing track. For the ESP32-CAM, we are using the OV2640, which offers around 65 degrees of FOV, sufficient to capture adequate scene information. Additionally, the maximum resolution can reach 1622x1200. When used separately from a computer, the ESP32-S3 is capable of shooting at 400 × 300 pixels and running relatively small AI models at the same time. This gives us enough computing power and pixels to find the hand bones.

5 Conclusion

5.1 Accomplishments

Overall, we were able to accomplish most of the requirements we had. We successfully delivered a final smart lamp product that satisfied all high-level requirements. Our final project were able to perform movement tracking and gesture recognition so that our users can use the smart lamp to minimize the shadow cast on their hands. In addition, our RGB light could correspond to gestures for users to change the light’s brightness and temperature by raising fingers up and down. Some of the design changes we made included replacing PCB with development boards, but the replacement allowed us to finish all remaining requirements fo the project and we did deliver the high level requirements we initially planned.

5.2 Uncertainties

The challenges with the project were mainly related to PCB and the limitation of camera angles. Our PCB design initially did not take into consideration of possible burning, and due to soldering issues we failed to drop 12V power supply to 5V, which resulted in the burning of the microcontroller. This issue could be avoided by performing adequate testing and separating power module from the control module. In addition, due to the limited access we have to a professional camera, the ESP32-CAM module we purchased had a limited field of view, which

caused the camera to fail to capture the entire hand image. As a result, we had to use a smaller model hand to ensure the camera could capture the entire hand image. This issue could also be resolved by purchasing more expensive professional camera with a larger field of view.

5.3 Future Work

For the future, we could continue refining the tracking algorithm and conduct additional image processing training to improve the accuracy of object movement detection. Also, if we have more resources, we could purchase better cameras and motors to improve the flexibility of the lamp. As of now, we are using an RGB light strip as our lighting component due to its low costs. We could also improve the quality of lighting by purchasing brighter and larger light bulb to improve the lighting quality. In addition, we would order preassembled PCB board to ensure any potential soldering issues may not interfere with the functionality of the final product.

5.4 Ethics and Safety

In terms of ethics and morality, our team adheres to the IEEE Code of Ethics adopted by the IEEE Board of Directors in June 2020. We firmly believe that as members of UIUC, a world-renowned university, we need to hold ourselves to the highest academic and ethical standards. We hope to change the world with our technology. Therefore, when working in our team, we will take the following measures, including but not limited to:

> 1: Diligently Studying Technology and Actively Communicating with Guides (TAs, Professors): Within our team, we strive to learn as much technology as possible in this project, as well as how to successfully combine the knowledge we have learned into a viable project. We will seek various resources, including but not limited to the internet, videos, and books, and actively seek advice from professors with insights in this field. At the same time, we will also actively share and exchange experiences and insights within the group to help everyone gain sufficient knowledge and skills. In this task, we will learn about relatively cutting-edge fields such as web development, AI vision, firmware development, PCB research and development, and robotics. Perhaps our ability to delve deeply is temporarily limited, but at least we can explore these fields and gain insights. We will continue to learn and apply this knowledge in the future in academia or the industry.
>
> 2: Developing Comprehensive Feedback and Testing Plans: In our team, because there are many different areas of content, we will develop comprehensive testing plans for

subsystems including robotic arms and AI, and timely feedback these test results to TAs while planning the next strategy.

3:Treating Everyone with Respect and Kindness and Ensuring These Codes Are Adhered To: To ensure good teamwork and communication, we have set up chat groups, a GitHub repository, and Google Drive space to share resources. We ensure all technical details can be tracked.

Laboratory Usage and Safety

Adherence to Laboratory Rules: Strictly following all laboratory rules and guidelines is essential for ensuring safety. This includes wearing appropriate personal protective equipment (PPE), such as safety goggles, gloves, and lab coats, when necessary.

Supervised Mechanical Work: When developing or working on mechanical parts, supervision or collaboration is crucial. Working alone on mechanical components can increase the risk of accidents, so we'll ensure that team members are present or that a lab supervisor is available for assistance.

Equipment Training: Before using any lab equipment, team members will receive proper training. This training will cover the operation of the equipment, safety procedures, and emergency protocols.

Digital Security

Secure Wi-Fi and Bluetooth Connections: Given the project's reliance on Wi-Fi and Bluetooth for communication, ensuring these connections are secure is vital. We will implement encryption protocols such as WPA2 for Wi-Fi and LE Secure Connections for Bluetooth to protect against unauthorized access.

Data Privacy: Protecting the privacy of users interacting with our system is paramount. We will design the system to collect only the necessary data, ensuring it is stored securely and that users are informed about how their data is used.

Reference

[1] "Detect objects with FOMO | Edge Impulse Documentation," Edgeimpulse.com, Feb. 29, 2024. https://docs.edgeimpulse.com/docs/tutorials/end-to-end-tutorials/object-detection/detect-objects-using-fomo (accessed Mar. 27, 2024).

[2] *"Hand landmarks detection guide | MediaPipe," Google for Developers. https://developers.google.com/mediapipe/solutions/vision/hand_landmarker#models (accessed Mar. 27, 2024).*

[3] Eloquentarduino.com, 2024. https://eloquentarduino.com/ (accessed Mar. 27, 2024).

[4] *IEEE code of Ethics*. IEEE. (n.d.-a). https://www.ieee.org/about/corporate/governance/p7-8.html

Appendix A Requirements & Verification Tables

Requirement	Verification	Results
1. Able to capture the image when the sensor is powered on by 5V +/- 5%	• Set up both camera sensors on Arduino IDE and connect them to a 5V power source • Use an multimeter on the camera power input to verify that the power input is within the range 5V +/- 5% • Check on a personal laptop if Arduino IDE can receive images from the ESP32 sensor	Pass
2. Able to conduct model training for the tracking object	• Conduct a model test on ESP32S3, the processing time for each frame should be less than 200ms.	Pass
3. Able to accurately count stretched-out fingers within the range of 0-5	• Use the gesture detection camera to take 20 hand images with stretched-out fingers ranging from 0 - 5. • Check all trained images' properties to verify if the images are correctly tagged with finger numbers within the range of 0 - 5	Pass
4. Able to identify hand coordinates to enable hand movement tracking	• Set up the motion tracking camera to start taking 96 x 96 pixels pictures • Set the center coordinate to 48 x 48 pixels to compute offsets with the new coordinate of hands • Move one's hand in front of the camera to observe if the camera can move in the same direction as the hand's movement	Pass

Table 2: Sensor Subsystem Verification Table

Requirement	Verification	Result
1. Able to communicate data through PWM, and UART	• Set up and program the ESP-WROOM-32 Microcontroller through Arduino IDE • Connect ADALM2000 between the	PCB was unprogrammable, central control was replaced with

protocols	ports used to transfer PWM and UART data to verify if signals are sent to sensor and lamp subsystems	development board
2. Communicate with the mechanical arm to follow human hands with a delay < 0.5s	• Set up and assemble the complete control, lamp, sensor, and power systems • Initiate hand movement in front of ESP32 Cam • Use the timer app on your mobile phone to ensure the reaction speed delay of mechanical arm tracking is below 0.5s	Pass
3. Communicate with the light controller to switch on/off and adjust brightness with a delay < 0.5s	• Set up and assemble the complete control, lamp, sensor, and power systems • Stretch out one finger (turn on light gesture) in front of ESP32 Cam • Use the timer app on your mobile phone to ensure the light switching delay is below 0.5s	Pass

Table 3: Central Control Subsystem Verification Table

Requirement	Verification	Result
1. Servo motors can move the mechanical arm when powered on by 12V +/- 5%	• Set up servo motors and mechanical arms and connect servo motors to a 3.3V power source • Activate HTS35H motors and observe if mechanical arms can be moved by motors	Pass
2. Mechanical arm can move in x and y directions	• Set up servo motors and mechanical arms and connect servo motors to a 12V power source • Activate the servo motors and observe if the mechanical arm can move in the corresponding x and y directions	Pass
3. RGB Light strip can adjust brightness and temperature when powered on by 5V +/-	• Assemble the complete power, sensor, central control, and lamp subsystems and connect an RGB light strip to a 5V power source • Stretch out 2-3 fingers in front of ESP32	Pass

5%	Cam to verify if the light strip can increase/decrease brightness through dimming ● Stretch out 4-5 fingers in front of ESP32 Cam to verify if the light strip can change color temperature	

Table 4: Lamp Subsystem Verification Table

Requirement	Verification	Result
1. Primary voltage regulator enables 12V +/- 5% voltage output	● Connect a 12V voltage regulator to a wall outlet ● Use a multimeter on the voltage regulator output to verify if it supports 12V +/- 5% output	Passed using an external power adapter and development board
2. Primary voltage regulator enables 5V +/- 5% voltage output	● Connect a 5V voltage regulator to a wall outlet ● Use a multimeter on the voltage regulator output to verify if it supports 5V +/- 5% output	Pass
3. Secondary voltage regulator enables 3.3V +/- 5% voltage output	● Connect a 3.3V voltage regulator to a wall outlet ● Use a multimeter on the voltage regulator output to verify if it supports 3.3V +/- 5% output	Pass

Table 5: Power Subsystem Verification Table