FINAL REPORT

By

Dhruv Diddi

James Jia

Saksham Saini

Final Report for ECE 445, Senior Design, Spring 2017

TA: Eric Clark

03 May 2017

Project No. 82

Abstract

The Robotic Lamp is a device that combines the typical uses of a desk lamp with added automation abilities. The Robotic Lamp recognizes gestures through it's built-in camera, with each gesture paired with LED functionality and Servo Motor animations. While designing our product, we learned the computational intensity that was required for well-trained gesture recognition, and diverted from many of our design choices in the original Design Document. In this Final Report, you will read about the successes and shortcomings of our product, as well as our original design choices and the few modifications we made to them.

Contents

1 Introduction

Home automation continues to be on the forefront of technological advancement in 2016 and in 2017. While we have seen software-human interaction, such as Google Assistant, Siri, and Alexa, and we have seen automation in the form of smart lighting, thermostats, and more, we have yet to see many products that bring automation and interaction together. Specifically, we wanted to look at the office setting - lamps. We spend a lot of time at our office desks working on our computers, scribbling on our papers, and 46% of that time is spent after daylight hours[1]. While lighting in the office typically is through the use of lamps, these lamps are stationary, out of reach, and many on the market still lack capabilities such as dimmable lighting. These can be annoyances for many office workers - having to reach over their tables to simply interact with their lamp.

Our solution - a robotic lamp - aims to provide easy interactivity for anyone who spends their time at the office. The robotic lamp provides interaction through a camera interface. Rather than reaching over to change settings, one can simply use hand gestures for configurations such as increasing or decreasing the brightness of the lamp. The motors of the lamp, paired with hand gestures, provide animation and an additional layer of interactivity with the lamp.

In this Final Report, you will read about the successes and shortcomings of the Robotic Lamp. We made large modifications to how Gesture Recognition worked, and decided to use detection through localization as a proof-of-concept as opposed to original Design choices. Additionally, you will read of the challenges faced with the ability to animate the lamp due to it's weight. Overall, the Lamp was a success, with the caveat that many design choices had to be modified. There will be additional explanation on design modifications and choices we would/will make in future iterations of the product.

2 Design

2.1 Block Diagram

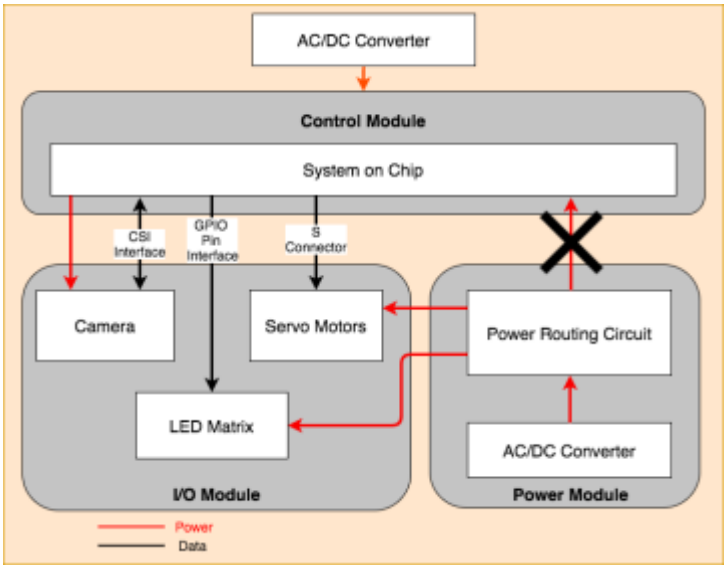

Figure 1. Block Diagram

Our final design of the Lamp followed the exact Block Diagram shown in Figure 1, with one change. Prior to the demonstration of the Robotic Lamp, we prepared our System on Chip (Raspberry Pi Zero[2]) to be powered by the Power Routing Circuit. However, we did not follow the recommended current for powering the Raspberry Pi Zero, and fried the board. We only had one more System on Chip on-hand to use (Raspberry Pi 3[18]), and we decided to not take our chances at another mistake. The System on Chip at it's final stages was powered by an AC/DC Wall to MicroUSB Adapter.

2.2 Circuit Schematic

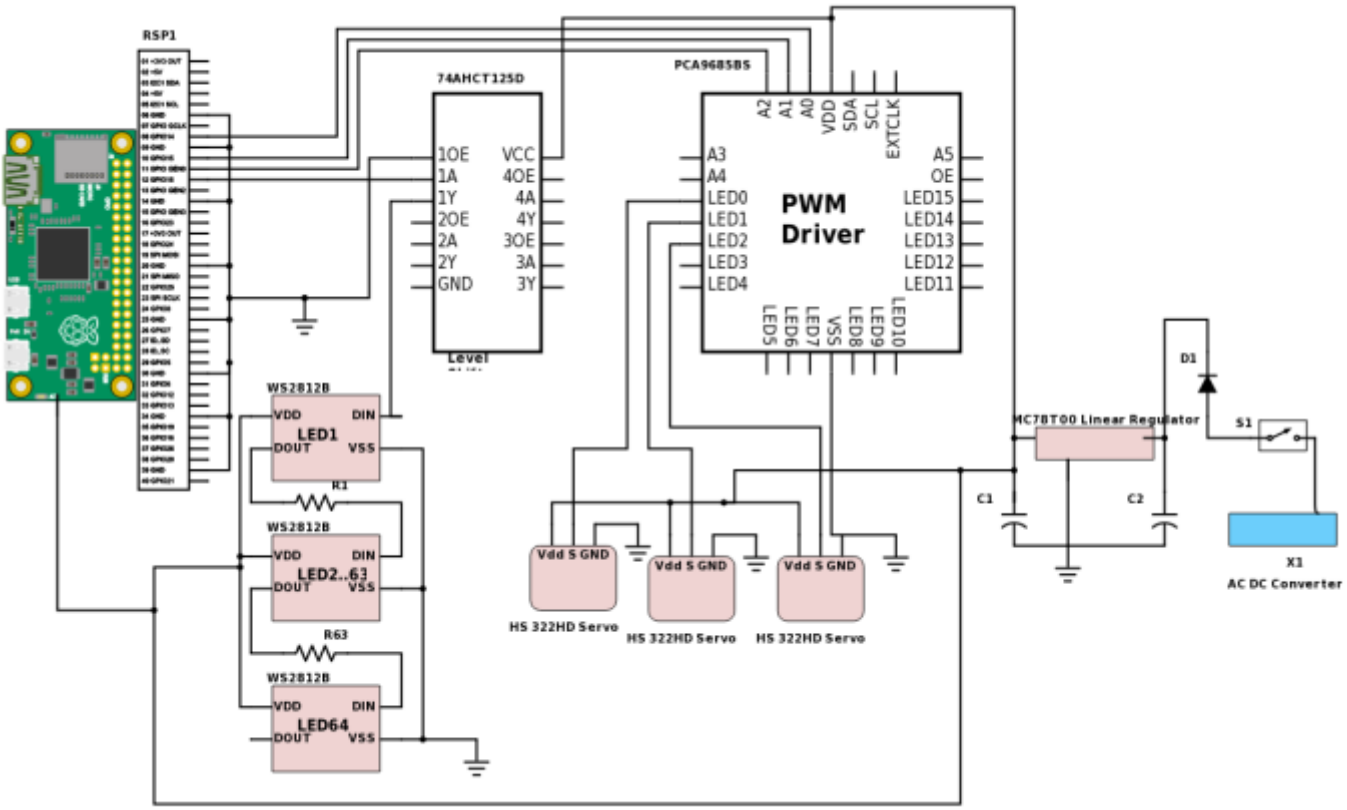

Figure 2. Circuit Schematic

The final circuit schematic is as follows. No changes were made from our initial circuit, except that the Raspberry Pi Zero would be replaced with a Raspberry Pi 3, and that it's power would come from the separate AC/DC Wall Converter. Due to issues with the MC78T00 Voltage Regulator[3] and not having a second on hand, we opted to replace it with the 7805 Voltage Regulator[4].

2.3 Control Module

Our final design included a Raspberry Pi Zero as the control module as it provided enough computational power, with its 1Ghz, 512 MB RAM specifications, for gesture recognition. Additionally, it was cheap, low power, and low weight. As prefaced in the "Block Diagram" section, we had a Raspberry Pi 3[18] as a backup and eventually used it for our demonstration, though the Raspberry Pi Zero[2] was sufficient for our needs.

The Raspberry Pi family was an overall decision to simplify portions of our product that would otherwise require much more circuitry and programming. For example, the Raspberry Pi includes a CSI Interface

that allows for easy interfacing to a CSI-compatible camera. Using an OV5647 camera[5] allows for a circuit-less connection with the camera, which is powered by the Raspberry Pi. Additionally, interfacing with the Servo Motors and LED Matrix was much simpler with the compatibility of a Servo Motor (PWM) Driver[6], as well as an LED (PWM) Driver[7]. This, however, resulted in a design that was too simple, in the sense that we did not need much circuitry for the design of the lamp.

In our "Camera" portion of this report, we will discuss many different gesture recognition techniques. Some of these techniques are very computationally intensive, and would require a much more powerful chip. In future design considerations, we will continue to test these gesture techniques on different systems, including both System-on-Chips, and desktop stations. We would also like to consider the possibility of a cheap microprocessor, though most likely not powerful enough for gesture recognition.

	Raspberry Pi Zero	**Raspberry Pi 3**
Cost	$5	$35
Weight	0.3 oz	2.4 oz (with case)
Processor	1 Ghz, 1 core	1.2 Ghz, 4 core
RAM	512 MB	1 GB
Power Consumption	0.4 - 0.7W	1.4 – 3.7W

Figure 3. Raspberry Pi Comparisons[8]

2.4 I/O Module

2.4.1 Camera

The camera module chosen for this project was the OV5647[5] module. It is one of the only two CSI Interface compatible cameras available on the market right now and was the cheaper alternative out of the two. The camera has a 5MP sensor capable of recording at 1080p at 30fps, which is more than enough for our needs. We capped the video stream to 240p at 12fps to ensure that the Raspberry Pi performs the gesture recognition in under 2 seconds, according to our requirements.
For recognizing the different gestures, we tried the following list of computer vision techniques:

2.4.1.1 Contour and Convex Hull Detection

This technique converts an image into a black and white image using a threshold value. The black and white image is then used to find continuous curves followed by convexity defects which can be used to calculate how many fingers the user is holding up in front of the camera. Example images can be seen in Figure 5.

Advantages	**Limitations**
Fast Low computational time	Low accuracy

Figure 4. Contour and Convex Hull Detection Advantages and Limitations

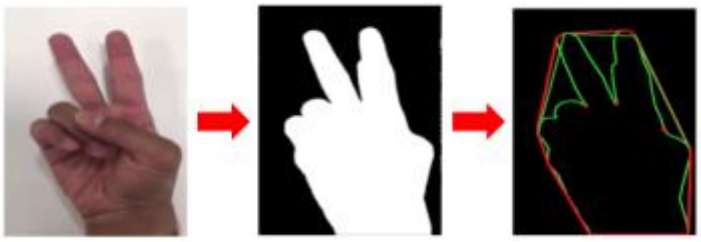

Figure 5. Threshold Effect to Contour/Convex Hull Detection

2.4.1.2 FASTFeature Recognition, ORB and SIFT Feature Recognition

FASTFeature Recognition uses corner detection based on the brightness of pixels to extract features in a circle of 16 pixels on the image for any random pixel *p*, as shown in Figure 6.

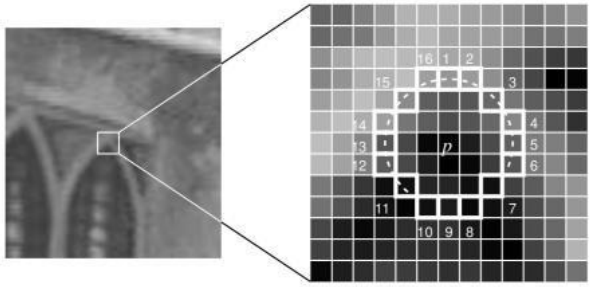

Figure 6. FASTFeature Recognition Example[9]

2.4.1.3 SIFT Feature Recognition

SIFT transforms the image into a large collection of keypoints, and then computing their descriptors. This algorithm compares each pixel to its neighbors to calculate these descriptors.

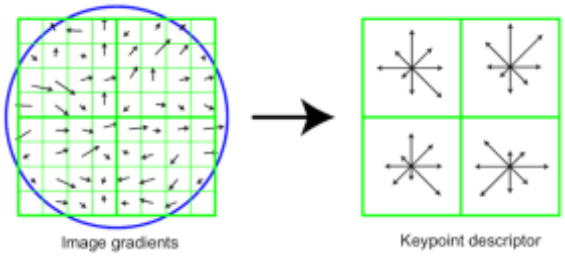

Figure 7. SIFT Feature Recognition Example[10]

2.4.1.4 ORB Feature Recognition

ORB is a combination of FAST and BRIEF, and is a good alternative for SIFT because of its low computational time. It extracts corners like FAST and keypoints like SIFT to generate features from an image.

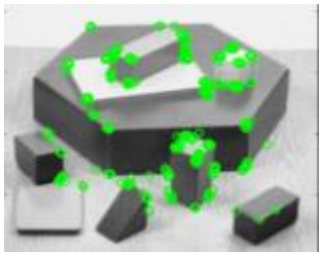

Figure 8. ORB Feature Recognition Example[11]

Technique	Advantages	Limitations
FAST	Low computational time	Low accuracy
ORB	Low computational time	Low accuracy
SIFT	High computational time	High accuracy

Figure 9. FASTFeature, ORB, SIFT Technique Advantages and Limitations

2.4.1.5 Convolutional Neural Network

Convolutional Neural Networks are based on the principle of how the visual cortex is organized in animals. It identifies the most prominent features first (curves, corners, etc.) and then starts narrowing down on what the input could be classified as based on the training dataset.

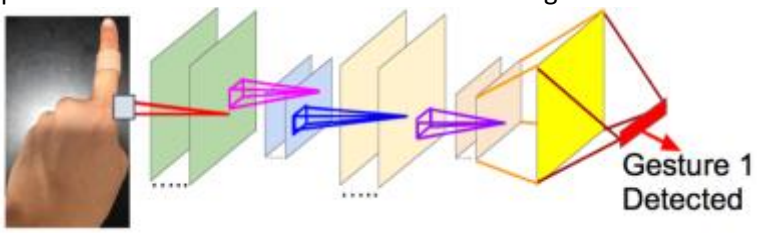

Figure 10. Convolutional Neural Network Gesture Detection Example

Advantages	Limitations
High accuracy	Very high computational time Requires huge dataset for training

Figure 11. Convolutional Neural Network Adavantages and Limitations

2.4.1.6 HAAR Cascade Classifiers

HAAR Cascade Classifiers generate a cascade of classifiers from extracted features which are based on the relative brightness of pixels.

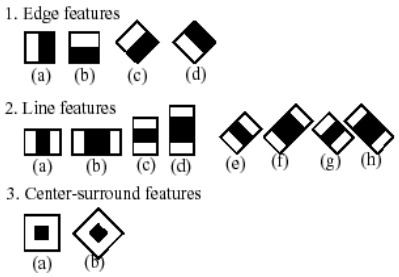

Figure 12. HAAR Cascade Example Features to Detect[12]

Advantages	Limitations
High accuracy	Training takes a long time Requires huge dataset for training

Figure 13. HAAR Cascade Classifiers Advantages and Limitations

2.4.1.7 Final Approach

After working with all these different techniques, we realized that the accuracy was not enough to meet our requirements. Therefore, we decided to take a different approach which involved using Contour and Convex Hull detection along with localization on the input screen to select the different gestures.

Figure 14. Contour Detection with Localization Example

The algorithm detects a five finger(open hand) gesture and locks onto a gesture when the hand is moved into one of the four squares. After this, we go into the "Enter Recognition" Mode which is followed by the execution of LED and motor functions.

2.4.2 LED Matrix

The WS2812B LEDs had a Raspberry Pi Driver[7] that handled all of the low-level logic. This included timing to turn on and off LEDs, and set the RGB Values of each individual one. Since PWM signals require very specific timing, this driver was an essential piece of software. On top of this Driver was an "Adafruit_Neopixel" API[13] that handled all driver calls, which made it very easy to set and get RGB values for each LED, as well as increase and decrease the brightness of the LEDs. We decided on creating and designing a Python API for the Robotic Lamp that would sit on top of the "Adafruit_Neopixel" API. All code is accessible on Github[14].

2.4.2.1 High Level (LEDLogic.py)

LEDLogic.py was the process that handled all LED commands. One challenge we faced when creating a top-level program (to run all of our logic for Gesture Recognition and LED/Servo Motor Response) was that Gesture Recognition had to be run in a virtual environment to prevent corruption of the filesystem and memory. However, the LEDs and Servo Motors required superuser access in order to output data via the GPIO pins. This became a dilemma, as spawning a subprocess[15] in Python allowed for no parallel processing, but rather sequential processing. We decided to go for the pexpect Python library[16] that allowed to spawn child processes, and send string data to the child process. Therefore, our LED logic was all encapsulated in a child process, where commands would be inputted as a string to it.

2.4.2.2 Detection Layer (RecognitionMode.py)

This layer is continuously ran in the background while waiting for a gesture to be recognized. It cycles through "windows" to indicate waiting for a gesture. Once a gesture has been recognized for the first time, it displays a symbol indicating the accuracy level. This was the only functionality that appeared in our final demonstration. However, code is in place to handle showing a "progress bar" and symbol to indicate the accuracy level of the gesture on the screen, and give a running total over multiple frames.

2.4.2.3 Command Layer (BrightnessControl.py, LEDClock.py, ColorRotator.py)

The Command Layer is an interface that pairs different LED functionalities to different gestures. After future polishing, that would mean that any class would be interchangeable in code by simply changing which class you are initializing. This interface follows a simple formula - Enter, Execute, Exit, as shown in Figure 15.

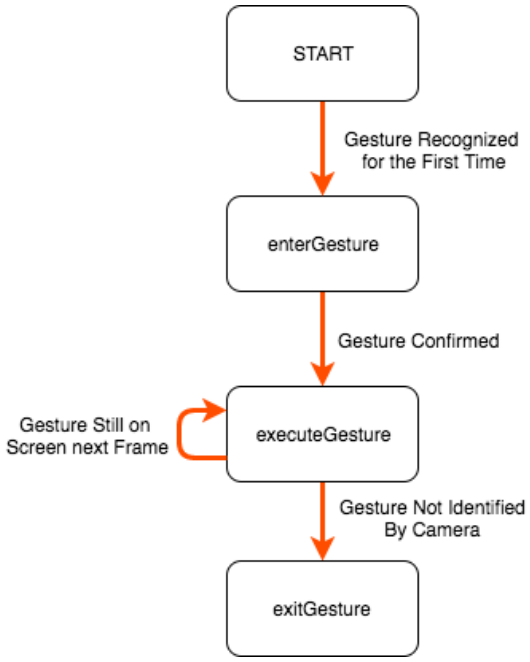

Figure 15. Gesture Interface Software Flow

After a gesture is recognized, the gesture's "enter" method begins, where the class takes an LED Snapshot and saves other necessary information in the class. The "execute" method performs the gesture's LED action, and this method would run each time the gesture is recognized past the first time. The "exit" method returns the LEDs to their original or future changed state, after the gesture is not on the screen anymore. Currently there are 4 gestures programmed with the Command Layer: brightness up (BrightnessControl.py), brightness down (BrightnessControl.py), a Time Clock (LEDClock.py), and a Color Rotation simulation (ColorRotator.py).

2.4.2.4 Low Level (Symbols.py, Colors.py)

These Low-Level programs sit on top of the "Adafruit_Neopixel" class, providing easy extensibility for the 8x8 Matrix we are using. The Symbols.py file provides access to users for adding symbols to the matrix. A "Symbol" 8-byte array is used to represent which LEDs are on and off, with each byte corresponding to one row of the LEDs. Internal logic is used to convert the "Symbol" to an internal representation of the on/off LEDs. By doing this, users can add fonts and multiple animations by adding byte-representation rather than bit-representation.

Colors.py provides getters and setters for the RGB Values of the LEDs. Typically one would use multiple for loops to set each LED a certain color, but with this, future users can change the colors in multiple ways with a simple method call.

2.4.3 Servo Motors

The servo motors that we chose for this product were HS 322HD[17], since they are lightweight and provide a stall torque of 3kg/cm. The motors are controlled by a PWM module that is discussed later in

this report. The motors receive a PWM signal from the PWM module which tells each servo motor what angle to turn to. We had three motors in the Robotic Lamp to provide 3 axes of rotational motion. However, with the additional weight added by the Raspberry Pi 3[18] and the wires that we didn't account for in our requirements and verifications, we were only able to get full motion(30°-40°) in one servo with very limited motion (<10°) in the other two. During our design phase, we considered the load torque and angular velocity of the servo motors. Since we understood that the relationship between torque and rotational speed is linear, we made estimations for the lamp head's torque to be:

$$\tau = 3 - 0.544\omega$$

τ = Torque (kg/cm)

ω = Angular Velocity (rads/second)

Since the motors did not reach full range of motion (180°), we find that our estimations of the lamp head weight and radius were not accurate after the change in components. These numbers are hard to measure due to the nature of the already-built lamp. In future iterations, we will consider an alternative motor that supports a stall torque of more than 6 times the HS 322HD, such as the MG958 Servo Motor[19].

2.4.3.1 PWM Module and Servo Control (SimpleTest.py)

The PWM module that we used for our project was the Adafruit PCA9685 [20]. The functions in this file allow you to address and control all three motors using the Set PWM function (pwm.set_pwm(motorNumber, 0, Tick)), where the first argument(motorNumber) tells which motor to address and the third argument(Tick) tells how much the motor should turn. The value of Tick can be any integer between 0-4096.

2.5 Power Module

The AC/DC converter that we used was a 110V AC to 12V DC converter which delivered a steady 12V DC output within an error range of (±0.15V). This 12V output was stepped down to 5V using two 7805 voltage regulators[4], which were readily available at the ECE Shop, to power the LED matrix and the servo motors. We had previously considered a buck converter but decided that the voltage regulator would suffice. However, issues with heat did come up, which resulted in burning out one of the voltage regulators.

3 Design Verification

3.1 Control Module

The first requirement tested to make sure the System on Chip would be able to process a 10-second video stream in less than 500ms to a raw format. On the Raspberry Pi Zero[2], we were consistently able to achieve 400-450ms. On the Raspberry Pi 3[18], numbers were consistently under 200ms. Note that these numbers are with the Graphical User Interface on and the video being shown on the interface - running this test without the interface on would result in much faster processing and a lower time.

The four additional requirements consisted of testing each gesture's execution from recognition, to action, and to de-recognition. Though we were unable to test these requirements on the Raspberry Pi Zero, we found that the time taken to execute each gesture using our final hand recognition technique,

Contour and Convex Hull Detection, was within the limitations of our requirements (under 10 seconds). On the Raspberry Pi 3, each step of the control logic would be performed on a frame before the next arrived. Thus, since frames per second were set at 12, actions were being performed at under 0.083333 seconds each. However, due to our drastic change to the design of our gesture recognition to achieve higher accuracy, we considered these requirements failed. The gesture recognition was more so detecting hand location rather than hand gesture, which was not the intentions of our initial design. Furthermore, requirement 5 lacked the full rotations of the motors.

3.2 I/O Module

3.2.1 Camera

The camera exceeded basic requirements. The first requirement for the Camera required a completion time of under 8.333333 seconds, where we were consistently obtaining times under 2 seconds. These tests were even ran with a Graphical User Interface in the background, which would increase time due to needing to process the camera video on the screen. For the second requirement, the camera was able to obtain a field-of-view of 83.2° x 52.6°, far exceeding basic requirements.

3.2.2 LED Matrix

Verifications specific for the LED Matrix were to confirm that the Matrix we used was not defective. The verification tests simply indicated that the LEDs were able to shine brightly for long periods of time, and rapidly change RGB values. These were both confirmed in our verifications.

3.2.3 Servo Motors

Though our servo motors did not have full range of motion after including gauge wires and the Raspberry Pi 3, we were able to test the motors beforehand, and were able to see full range of motion in the head of the lamp, head module, and upper neck of the lamp. As stated in our Design, it is hard to measure the torque of the lamp after construction, so exactly what torque the lamp was holding is unidentifiable. In terms of temperature measurements for the second requirement, we had no issue with a high temperature of the motors, as persistent use of the servo motor stayed in between 79°F and 80°F.

3.3 Power Module

We had no issue implementing an override switch and having it properly work, both in our demonstration and for simple on/off capabilities. However, since we did not have an additional MC78T00[3] on hand, we had to use the 7805 Voltage Regulators[4], which supply 1.5A of current each/3A in total because we had two 7805s in the circuit. Therefore, we passed both the requirements for the power module which required a steady output current of 3A with 3% error at 5V and the ability to turn the whole circuit off using the override switch.

4 Costs

Development costs are based on a $45/hour salary, working 8 hours a week for 3 people. We believe that to be production-ready there are a lot of added features that are out-of-scope for this course, but can be implemented, so only 50% of our final design is appropriated in the labor.

$$3 * \frac{\$35}{1\ hour} * \frac{8\ hours}{1\ week} * \frac{15\ weeks}{} * 2.5 * 2 = \$63,000$$

The machine shop will be helping to place the servo motors within our pre-bought lamp. The cost are as follows:

$$\frac{\$50}{1\ hour} * \frac{10\ hours}{} = \$500$$

The cost chart is as follows. Note that these parts are for a single prototype, and do not include parts that we needed to replace.

Part	Cost (prototype)	Cost (bulk,100 pieces)
Raspberry Pi 3 System on Chip	$37.00	$35.00
OV5467 Camera Module	$16.99	$6.84
8x8 WS2812B LED Matrix	$15.62	$4.30
74AHCT125 Level Shifter	$1.50	$1.20
Architect Lamp	$22.00	$5.94
2x 7805 Voltage Regulators	$1.12	$0.88
3x HITEC HS 322HD Servo Motors	$43.35	$43.35
Sunforce GM1200 AC/DC Converter	$22.04	$11.50
10Ω Resistors	$0.40	$0.10
1N4148WXTPMSCT-ND Diodes	$0.14	$0.08
0.33FCapacitor	$0.20	$0.12
PCA9685 PWM Driver	$14.95	$11.96
TRD13D10WL Switch	$1.00	$0.89
Total	**$176.32**	**$122.16**

5 Conclusion

5.1 Accomplishments

Although we had to make significant design changes to ensure integration and completion of our product, we were very proud to overcome the many individual and group challenges, especially in the later stages of our product. We were able to overcome many circuit-related issues at very short notice - this included burning out our Raspberry Pi Zero, Camera, First LED Matrix, and the Voltage Regulator. Additionally, many individuals advised that this project was too complex in nature for three software engineers with no robotics or Computer Vision experience: this came to be completely true, especially for our robotics portion. However, we strived to completely understand the Computer Vision portion of

the project, where we learned how each method worked. We find that the learning experience was more of an accomplishment than the product itself.

5.2 Uncertainties

A few uncertainties with our product came through our different portions of hardware malfunctioning. While initially testing a cheaper LED Matrix that we found on eBay for $15, we had trouble lighting it up and performing simple animations. We had little to no issues when we purchased a more robust, expensive one from Adafruit [20]. However, prior to our demo, the Adafruit LEDs began malfunctioning, unbeknownst to us the reason. As we reverted to the cheaper matrix, we found ourselves confused as to the issue, as we were inputting the same data and power to both. Additionally, we are still unsure of the reason our Raspberry Pi Zero and Camera stopped working - due to time constraints and only having one back-up board available, we decided it was too much of a risk to figure out why.

5.3 Ethical Considerations

We wanted to highlight and consider two very important areas of ethics: privacy and safety. As spying on web and safety cameras continues to become more and more of an issue this decade, we look to IEEE Ethic #8 for a response to this pandemic: “to treat fairly all persons and to not engage in acts of discrimination…” [21]. We continue to regard this area of ethics as a top priority, designing our system to be preventative of discriminatory acts on both software and hardware. By choosing a System on Chip that has no internet access built in the hardware, nor any wireless capabilities (such as WiFi or Bluetooth), we are confident that a future product would not raise ethical concerns but boost confidence in a consumer that would otherwise have a camera staring at them on their desk.

Safety is now and always an utmost concern for everyone. We want to greatly consider IEEE Ethics #1: “to accept responsibility in making decisions consistent with the safety, health, and the welfare of the public…” [21], as well as IEEE Ethics #9: “to avoid injuring others, their property, reputation, or employment by false or malicious action;” [21]. Each motor is only capable of moving on a single axis, thereby limiting the lamp’s range of motion, but still covering a large area. This allows a consumer to better judge how the lamp would move. Although this is important, the more important issue is the force that the lamp can exert from each motor. As we described in the design of the Servo Motors, we deem the current product unable to cause harm. In future iterations of the product, we aim to increase the ranges of motion for each servo motor, but will limit the output of force per motor.

5.4 Future Work

There is a lot of ground to cover for future iterations of the product. As we have considered many different gesture recognition techniques, we will be spending more time diving into those techniques, and learning their algorithms, computational needs, training needs, and if any one of them would be fit for the lamp. We also realized that the choice of gestures also makes a lot of difference in accuracy (one finger vs. two fingers is harder to distinguish compared to one finger vs. five fingers). We will consider replacing the camera with a microphone, as the recent years have been very good to voice recognition techniques. We believe this is the most important steps to create improvements.

References

[1] National Oceanic and Atmospheric Administration. (2004). "RANKING OF CITIES BASED ON % ANNUAL POSSIBLE SUNSHINE IN DESCENDING ORDER FROM MOST TO LEAST AVERAGE POSSIBLE SUNSHINE" [Online]. Available FTP: https://www1.ncdc.noaa.gov Directory: pub/data/ccd-data File: pctposrank.txt

[2] Raspberry Pi. (2015, September 23). "RASPBERRY PI ZERO" [Online]. Available at: https://www.raspberrypi.org/documentation/hardware/raspberrypi/mechanical/rpi-zero-v1_2_dimensions.pdf

[3] ALLDATASHEET. "MC78T00 Series 3.0 A Positive Voltage Regulators" [Online]. Available at: http://pdf1.alldatasheet.com/datasheet-pdf/view/173631/ONSEMI/MC78T00.html

[4] Sparkfun. (2013, May) "µA7800 SERIES POSITIVE-VOLTAGE REGULATORS" [Online]. Available at: https://www.sparkfun.com/datasheets/Components/LM7805.pdf

[5] OmniVision. "OV5647 5-megapixel product brief" [Online]. Available at: http://www.ovt.com/uploads/parts/OV5647.pdf

[6] Github. "Adafruit Python PCA9685" [Online]. Avialable at: https://github.com/adafruit/Adafruit_Python_PCA9685

[7] Garf, Jeremy. "Userspace Raspberry Pi PWM library for WS281X LEDs" [Online]. Available at: https://github.com/jgarff/rpi_ws281x

[8] Raspberry Pi Dramble. "Power Consumption" [Online]. Available at: https://www.pidramble.com/wiki/benchmarks/power-consumption

[9] OpenCV. (2014, November 10) "FAST Algorithm for Corner Detection" [Online]. Available at: *http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html*

[10] OpenCV. (2014, November 10) "Introduction to SIFT (Scale-Invariant Feature Transform)" [Online]. Available at: *http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro*

[11] OpenCV. (2014, November 10) "ORB (Oriented FAST and Rotated BRIEF) [Online]. Available at: *http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html#orb*

[12] OpenCV. "Face Detection using Haar Cascades" [Online]. Available at: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

[13] Garf, Jeremy. "rpi_ws281x/python" [Online]. Available at: https://github.com/jgarff/rpi_ws281x/tree/master/python

[14] Saini, Saksham. "RoboticLamp" [Online]. Available at: https://github.com/ssaini4/RoboticLamp/tree/master/main

[15] Python. “17.1. subprocess – Subprocess Management” [Online]. Available at: https://docs.python.org/2/library/subprocess.html

[16] Pexpect. “Pexpect version 4.2” [Online]. Available at: https://pexpect.readthedocs.io/en/stable/

[17] SERVOCITY.com. “HS-322HD Servo” [Online]. Available: https://www.servocity.com/hs-322hd-servo

[18] Raspberry Pi. “Raspberry Pi 3 Model B” [Online]. Available at: http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf

[19] Towerpro. “MG958” [Online]. Available at: http://www.towerpro.com.tw/product/mg958/

[20] Adafruit. “Adafruit Neopixel NeoMatrix 8x8 – 64 RGB LED Pixel Matrix” [Online]. Available at: https://www.adafruit.com/product/1487

[21] Ieee.org. "IEEE IEEE Code of Ethics". Available: http://www.ieee.org/about/corporate/governance/p7-8.html

Appendix A. Requirements and Verifications Table

Requirements	Verification	Status
	Control Module	
1. Can receive video stream from the camera module at a data rate of at least 0.9216Mbps (for 240p at 12fps) and process it in no more than 500ms.	1. a. Connect the camera module to the CSI interface on the Raspberry Pi. b. Run controlRV1.py script. This script is used to test the bitrate of the camera module. c. The program records a 10 second video at 240p resolution and shows the time it took to transfer and process the video to the Raspberry Pi to a raw H.264 Video Format. Ensure that the time does not exceed 500ms via the Terminal output "Process time was x ms."	Yes
2. Can identify and perform one iteration of the "Brightness Up" Control Logic within 6 seconds, which consists of: a. 2 seconds to recognize the hand gesture and indicate the gesture on the LEDs b. 2 seconds to increase the LED brightness c. 2 seconds to recognize the gesture removed and return LEDs to before the gesture (with brightness changed)	2. a. Connect the Raspberry Pi terminal via SSH or external monitor and keyboard. Run main.py. Ensure every LED is lit. b. Start a timer. Place marker image in front of the lamp's camera. LEDs should output, from the bottom of the matrix, a bar of lit LEDs for each level of brightness. Stop the timer, and ensure time is under 2 seconds c. After LEDs have changed from the previous step, start another timer. Keep the marker image in front of the camera. Another horizontal bar of LEDs should light up, and ensure the LEDs are visibly brighter. Stop the timer after this has happened at least once, and ensure time is under 2 seconds d. Start another timer. Remove the marker image from the camera's field of view. LEDs should return to the configuration in step a but visibly brighter.. Stop the timer, and ensure time is under 2 seconds	No
3. Can identify and perform one iteration of the "Brightness Down" Control Logic within 6 seconds, which consists of: a. 2 seconds to recognize the hand gesture and indicate the gesture on the LEDs	3. a. Connect the Raspberry Pi terminal via SSH or external monitor and keyboard. Run main.py. Ensure every LED is lit. b. Start a timer. Place marker image in front of the lamp's camera. LEDs should output, from the bottom of the matrix, a bar of lit LEDs for each level of brightness. Stop the timer, and ensure time is under 2 seconds c. After LEDs have changed from the previous step, start another timer. Keep the marker image in front of the camera. A horizontal bar of LEDs should turn off, and ensure the LEDs are visibly dimmer. Stop the timer after this has happened at least once, and ensure time is under 2	No

b. 2 seconds to decrease the LED brightness c. 2 seconds to recognize the gesture removed and return LEDs to before the gesture (with brightness changed)	seconds d. Start another timer. Remove the marker image from the camera's field of view. LEDs should return to the configuration in step a but visibly dimmer. Stop the timer, and ensure time is under 2 seconds	
4. Can identify and perform one iteration of the "Time Clock" Control Logic within 7 seconds, which consists of: a. 5 seconds to recognize the gesture and perform LED changes b. 2 seconds to exit the gesture	4. a. Connect the Raspberry Pi terminal via SSH or external monitor and keyboard. Run main.py. Ensure every LED is lit. b. Start a timer. Place marker image in front of the lamp's camera. LEDs should flash the "Hour" digits and then the "Minute" digits afterward. Ensure the time is correct. Stop the timer after both Hour and Minute have flashed at least once, and ensure the time is under 5 seconds. c. Start another timer. Remove the marker image from the camera's field of view. LEDs should return to the configuration in step a. Stop the timer, and ensure time is under 2 seconds	No
5. Can identify and perform one iteration of the "Color Rotation" Control Logic within 10 seconds, which consists of: a. 8 seconds to recognize the gesture, perform LED changes, and have each motor rotate 180 degrees b. 2 seconds to exit the gesture	5. a. Connect the Raspberry Pi terminal via SSH or external monitor and keyboard. Run main.py. Ensure every LED is lit. b. Start a timer. Place marker image in front of the lamp's camera. LEDs should output a visible "C" character. LEDs should begin cycling through RGB colors. Additionally, each of the 3 motors will take turns rotating through its 180 degree degrees of motion. After all 3 motors have performed their rotations at least once, stop the timer. Ensure the time is under 8 seconds. c. Notice the RGB color of the LEDs. Start another timer. Remove the marker image from the camera's field of view. LEDs should return to the configuration in step a, but with each lit LED now with the RGB color noticed at the beginning of this step. Stop the timer, and ensure time is under 2 seconds	No
	I/O Module - Camera	
1. Takes images at a resolution of at least 240p (320 x 240) and a frame rate of at least 12 frames per second	1. a. Connect OV5647 Camera to Raspberry Pi Zero via the CSI connector b. Connect monitor via mini-HDMI to the SoC, as well as a USB Keyboard to the SoC c. Run CameraRV1.py. This script will output if the image's size is 320 x 240.	Yes

	d. Ensure the Terminal outputs “Size Check SUCCEEDED” along with the image size. e. The script will then take 100 photos at 320 x 240 resolution. f. Ensure the Terminal outputs “100 photos in x s. PASSED”, where x is a time under 8.333333 seconds (which is 12 frames per second)	
2. Provides a field-of-view of greater than or equal to 45° x 30°	2. a. Connect OV5647 Camera to Raspberry Pi Zero via the CSI connector b. Connect monitor via mini-HDMI to the SoC, as well as a USB Keyboard to the SoC c. Using the Linux Terminal on the SoC, run the command “sudo raspi-config” to enable the camera and configure image resolution d. Measure and stand 1 meter away from the center of the camera e. Run the command “raspistill -v -o test.jpg” in the terminal to begin a 5-second image preview begin f. Use a tape measure to span the horizontal axis of the photo frame, and record this value. g. Calculate the horizontal angle field-of-view using triangle identities to verify that it is greater than 45 degrees h. Repeat the process by instead measuring the vertical axis of the photo frame to verify that the vertical field-of-view is greater than 30 degrees	Yes
	I/O Module – LEDs	
1. At a voltage supply range from 3.5V-5.3V All LEDs light up white at for at least 30 minutes	1. a. Connect the 5V Power Supply to V_{DD} of the LED, as well as Ground to V_{SS} b. Connect GPIO Pin 18 of the Raspberry Pi Zero through a level shifter to raise the GPIO PWM output to 5V. Connect this 5V output to D_{IN} on the LED. Run ledRV1.py. This script will output white light (RGB 0xFFFFFF) at the brightest setting for 30 minutes. c. Ensure the lights stay bright white through the 30 minutes with no noticeable flickering.	Yes
2. LEDs are able to perform a “Color Wipe” Animation, where each LED one-by-one lights up bright red, then bright blue, then bright green.	2. a. Connect the Raspberry Pi terminal via SSH or external monitor and keyboard. Run ledRV2.py. b. This script will perform the “Color Wipe” Animation. Ensure each LED first obtains a bright red color. After every LED is light red, each LED should immediately change green, one after another. After every LED is light green, each LED should immediately change blue. Ensure there is no color distortion between red values, green values, and blue values.	Yes
	I/O Module – Servo Motors	
1. The torque requirement for the servo is at least a maximum of 3kg-cm. This	1. a. Connect the servo module to the power input(4.8V-6V) b. Attach drive gear to the servo end of the motor	Yes

implies that the servo can lift/control the load of the head of the lamp (50g), the head module(150g) and the load of the upper neck (1kg) of the lamp.	c. Since the project required complex weight distributed objects to be lifted with the motors, we connect the motor ends to threaded drives to parts of lamp with help from the engineers at ECE Machine Shop. d. We provide the motors peak voltage of 5V and signal simulated by the PWM driver simpletest.py code. e. Observe 3 axes of motion with the 3 motors present at the top, neck and middle joint for the lamp f. Observe motor behavior in being able to rotate the lamp parts.	
2. Temperature of the servo motors does not exceed 60°C in persistent(>2 minute) duty cycle changes.	2. a. Connect servo motors to GPIO pins of Raspberry Pi SoC b. Create duty cycle PWM signal using the testing PWM module provided by us. c. Use IR thermometer to measure the temperature of the motor casing. d. Collect and analyze temperature readings while the testing program run. This test program will continually rotate a motor for 2 minutes, followed by the next motors. Ensure temperature readings are below 60°C.	Yes
	Power Module – Power Routing Circuit	
1. Ensure that override switch turns the voltage off the level shifter and PWM driver when the switch is turned off	1. a. Connect functional generator to power routing circuit module b. Measure output voltage at the output of the linear voltage regulator with switch in ON position. The output should be 5V. c. Measure output voltage at the output of the linear voltage regulator with switch in OFF position. The output should be 0V. d. Confirm observation through repetition.	Yes
2. Ensure linear regulators work with the voltage conversion of 12V DC to 5V DC which maintaining the current to at least 2.54A	2. a. Connect functional generator to power routing circuit module b. Measure output voltage and current, before and after linear regulator using oscilloscope with switch in ON position. The current should be at least 2.54A at 5V within 3% error. c. Confirm observation through repetition.	Yes
	Power Module – AC/DC Converter	
1. Ensure that AC/DC voltage converted signal is consistent(± 0.15V) and accurate (intended voltage output of 12V) at least 2A.	1. a. Connect AC/DC voltage convertor to wall socket to receive 110V AC input b. Use multimeter to measure output DC voltage and ensure consistency and accuracy of 12V within 3% error. c. Test with resistors(5Ω, 100Ω, 500Ω, 5kΩ independently) and measure current across the resistors. d. Ensure the current measured is 2.4A, 0.12A, 24mA and 2.4mA respectively within 3% error.	Yes