University of Central Florida

Computer Science Senior Design I

Animatronic Lamp

Final Document

Authors:
Timothy Allen
Ian Lasky
Raphael Miller
Joshua Schroeder
Kevin Tran

Coordinator:
Dr. Mark Heinrich

May 16, 2018

Contents

List of Figures

List of Tables

Executive Summary

Self-formed and self-led, our project for this Senior Design course sequence is to create an animatronic desk lamp capable of interacting with the human user in a fun and playful manner. Our vision is for our animatronic desk lamp to have the capability of being described as sentient, that is, being capable of expressing emotions. Our inspiration for this project was founded from Disney's Pixar lamps, Luxo, Sr. and Luxo, Jr.

The lamp model will be a balanced-arm lamp, sometimes called a *floating arm lamp*. A floating arm lamp consists of an adjustable folding arm with two joints. The goal is to give the lamp the capability of moving its joints and head, also known as a shroud.

The lamp will have a stationary base. A Raspberry Pi computer will be installed inside the base of the lamp. This Raspberry Pi will control and manage the movement of the lamp.

The lamp's center light bulb will be taken out to place a web camera inside its shroud. In lieu of the light bulb, an LED ring will be installed around the camera within the shroud.

The idea is to make a lamp that is capable of listening to user voice commands with the aid of the Google Assistant API. Receiving the user's voice as input, the lamp will perform a corresponding action as output. For instance, the lamp shall be able to turn off at the user's command. This section shall be further detailed in 3.1.

One of the core functionalities is for this lamp to be able to detect the user's face using computer vision and face detection algorithms. In order to emulate human behavior, the lamp will look at the user's face and dim its circling LED lights as to not stress the user's eyes. Then, the lamp will eagerly wait for the user to give a command, such as to light up an object in a certain direction.

In order for this lamp to move, five servo motors will be attached with 3D printed brackets at five existing mounting points of the lamp. The lamp will learn to move smoothly to the desired point in space using machine learning. The lamp will be trained based on a feedback system from the developers, in which case the developers will rate each movement attempt from the lamp on a scale.

1 Introduction

The project internally formulated by the group is to create an animatronic lamp capable of tasks such as being able to recognize, track, and interact with its environment. Through these actions, the lamp will have the ability to interact with the user, express emotions, and adapt to new situations. Technologies including hardware implementations, computer vision algorithms, and machine learning will be implemented to see the success of this project.

Within this group, every member possesses their own individual motivation and reasoning for seeking the success of the project at hand. As such, the remaining paragraphs below are the detailed and tailored responses from each group member to the inquisition of what reasoning they have for participating and their expectations for the final result.

1.1 Statements of Motivation

Timothy Allen: When I heard the proposal for the animatronic lamp it really stuck out because it was unlike any other project that I had worked on before. I have never programmed an actual "physical thing" to do something before. Even though computer vision and machine learning interests me I have never tried to tackle either. In fact, that is a major reason that I chose this project. It sounded like a good opportunity to get some experience with both computer vision and machine learning and have fun doing it. I'm hoping to not only contribute a bunch to this project, but to also learn new and exciting things.

Ian Lasky: Since I was a kid I have always had a fascination with animation. Around two years ago in the middle of my computer science curriculum, I revisited my childhood interests and thought about the challenges that would be involved in bringing the ubiquitous "Pixar lamp" to life. At that point in time, it was far too extensive of a project to take on alone with the limited time I possessed. Now, however, with a team and more computer science knowledge under my belt, I believe the goal is achievable. The challenges are still challenging and the end result just as impressive. For me the project is presented more as a technological showcase of "can it be done?", yet at the same time it could present uses previously unconsidered which will be detailed in later sections.

Raphael Miller: The nature of robotics and computer vision has always been of interest to me. In this time of unlimited information and newer and more interesting ideas coming to the forefront we can assume that the field of robotics has been underrepresented. The world of science fiction has produced promises of artificial intelligence and machines that can not only perform tasks that we command but to do it in a way that it allows the users relative safety and comfort. From Issac Asimov's Laws of Robotics to the Tryell Corporation we have seen a vision of the future that founders of those concepts could have only dreamed of. In the 21st century, we finally have an opportunity to create those useful machines of yesteryear. Simple playthings in the minds of directors and writers come to life in real and practical forms. This lamp is more then just a Senior Design project, it helps bridge the neces-

sary gap to create the automatons that everyone hopes to own and use.

Joshua Schroeder: Since I saw the initial proposal for this project I knew I would be interested. After viewing all of the other project proposals I knew this was the one for me. The thought of combining hardware with computer learning has always interested me but I never had the right project to bring it to life. If we can successfully combine hardware with software to control this lamp and give it a personality, I'll be satisfied. Hardware and hardware interfacing has always been a large interest of mine, so getting to do that and working with others is awesome.

Kevin Tran: This project is very new and exciting for me, and I'm excited to help implement the machine learning portion. My interest is primarily motivated by the tangibility of the final product. For Senior Design, I would love to be able to showcase a real, physical artifact of work rather than a series of slides proving that a certain concept works. I would also like to work on something that is very unique and has not been done before in previous Senior Design semesters. Senior Design is the time and place to chase after fun new projects that may or may not have actual implementation and usage in the real world, and therefore would otherwise never be developed in the real world as a result of a lack of potential profit. All in all, I really wanted to make something different from the usual database-website-mobile app project. Furthermore, I chose this project for the group. In particular, I had a great semester with Ian when we worked on the

same project for the class COP4331 (Processes for Object-Oriented Software Development). Something I find integral in Senior Design is to be able to work with a good group of colleagues, and so I find it just as vitally important as choosing the project itself to also choose a fantastic team to see our Senior Design project to the end. Aside from Senior Design and group motivation, personally, I believe it is very rewarding to bring into this world a dynamically moving lamp that can interact with the user and bop its head to music. This lamp has the potential to be the life of any party, and it may also serve practical uses in-home by providing a unique lighting solution different from all of the rest of currently on the market.

2 Broader Impacts

An intrinsic aspect of the project at hand is the interaction of human and robot. This subject is still a poorly understood area of research and therefore the resulting outcome of this project could be of use in broadening its knowledge base. Such human-robot interaction has the potential to improve aspects of the user's daily life including their mood and productivity. Within the project, once the team believes they have reached a point of automation they are content with, they will field test the lamp with other students, perhaps faculty, and determine which aspects are seen useful and which can be improved or removed.

Additionally, this project could prove useful to those with a movement disability. The hands-free aspect would allow such users to control and interact with the lamp without needing to strain themselves.

3 Project Design

3.1 Required Functionality

Within this section will be the discussion and listing of required functionalities of the project. These requirements are non-negotiable and are the most basic actions that the completed project must be able to perform. These requirements are broken into two sections: quantitative and qualitative requirements. The former being measurable and definitive metrics set by the project team and the latter being more subjective functionality requirements unable to be attached to such a metric. The project team believes these requirements are both challenging and achievable within the scope of time allotted for the project.

The functionalities will be expressed in terms of a matrix with the first column being the functionality, the second being the importance to the final project, the third being the expected difficulty of the task, and the fourth being the priority of the task. The priority of the task is calculated from the product of the importance and the difficulty. This task matrix will help define the final project and determine the functionalities on which the team will focus. Both the scale for importance and difficulty range from one, being the least important or least difficult, to ten being the most difficult or most important. Thus, the priority can range from one to one hundred. Each of the team members completed their individual importance/difficulty matrix and the results were averaged. The requirements can be seen in the next section.

3.1.1 Quantitative Requirements

Table 1: Quantitative Requirements

Functionality	**Importance**	**Difficulty**	**Priority**
The system must attain a minimum of eighty percent (80%) accuracy when tracking objects	10.0	5.0	50.0
The system must attain five (5) degrees of freedom	9.4	2.2	20.7
The system must be able to capture and process a minimum of twenty (20) frames per second	9.4	2	18.8
The system must be able to emit colors in the RGB spectrum from (0,0,0) to (255,255,255)	3.8	1.4	5.3
The system must be able to emit a brightness value measured in lumens in the spectrum from 5 to 800	5.6	1.2	6.7
The System must be able to, with over 80% accuracy, detect an object.	5.0	4.0	20.0

3.1.2 Qualitative Requirements

Table 2: Qualitative Requirements

Functionality	Importance	Difficulty	Priority
The system must be able to smoothly move along its five (5) degrees of freedom	8.6	8.6	74
The system must be able to illuminate objects on objects of interest	9	3	27
The system must be able to detect a user's face within the viewing space	7.4	4	29.6
The system must be able to convey basic emotions through movement	7.2	8.6	61.9
The system must be able to listen to commands from the user	8	4	32
The system must be able to turn on and off at the user's command	10	1	10
The system must show minimal latency in movement or tracking	8.6	8.6	74
The system must integrate the Google Assistant API	8	2.8	22.4
The System must be able to detect the front of the face of any person	10.0	4.0	40.0
The System must be able to detect the side of the face of any person	9.0	4.0	36.0
The System must be able to detect writing instruments	7.0	7.0	49

The System must be able to detect paper on desks	7.0	8.0	3.0
The System must be able to interact with the Google Assistant API.	6.0	10.0	60.0
The System must be able to utilize the full list of Google Assistant Features	4.0	7.0	28.0
The System must be able to add custom features relevant to the lamp and its movement	8.0	5.0	40.0
The System must be able to add features that affect the lamps movement	1.0	10.0	10.0
The System must be able to connect using Google Assistant to other Google Home appliances	3.0	5.0	15.0
The System must be able to interact with other Google Assistant devices on the network	6.0	4.0	24.0
The System must be able to connect to the network using Wi-Fi 802.11a/g	10.0	1.0	10.0
The System must be able to, in using Google Assistant, convey and understand commands	10.0	3.0	30.0
The System must be able to, in using Google Assistant, convert those commands to actions done by the lamp	10.0	4.0	40.0
The System must be able to, in using Google Assistant, convey and understand commands	9.0	7.0	63.0

3.2 Optional Functionality

Along with the required functionality of the project detailed in Section 3.1, the team has compiled a list of desirable functionalities that, if time permits, would result in a more robust system. The following functionalities will be included within the project time line presented in Section 7 to motivate the team towards the completion of not only the required functionality but also the optional ones.

3.2.1 Optional Functionalities

Table 3: Optional Functionalities

Functionality	Importance	Difficulty	Priority
The system shall be able to shake its shroud in response to a question asked by a human user.	3.4	5	17
The system shall be able to 'bop' with the music being played by the user	3.2	6.6	21.1
The system shall be able to learn new commands	4.8	6	28.8
The system shall be able to play "catch".	1.2	10	12

4 Block Diagrams

4.0.1 Overall High Level Schematic

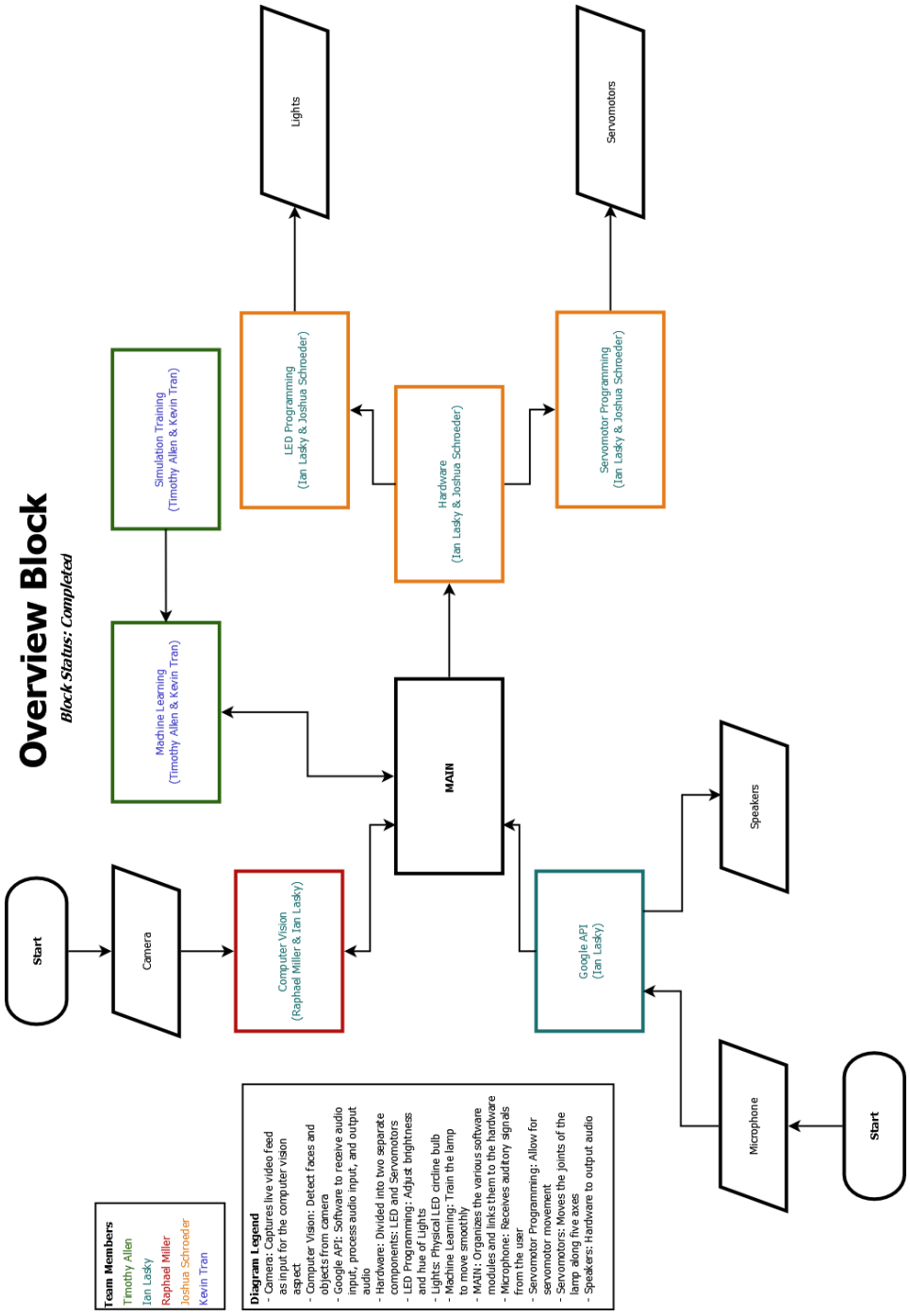

Figure 1: Overall Organization Flow Chart

4.0.2 Software Block Diagram

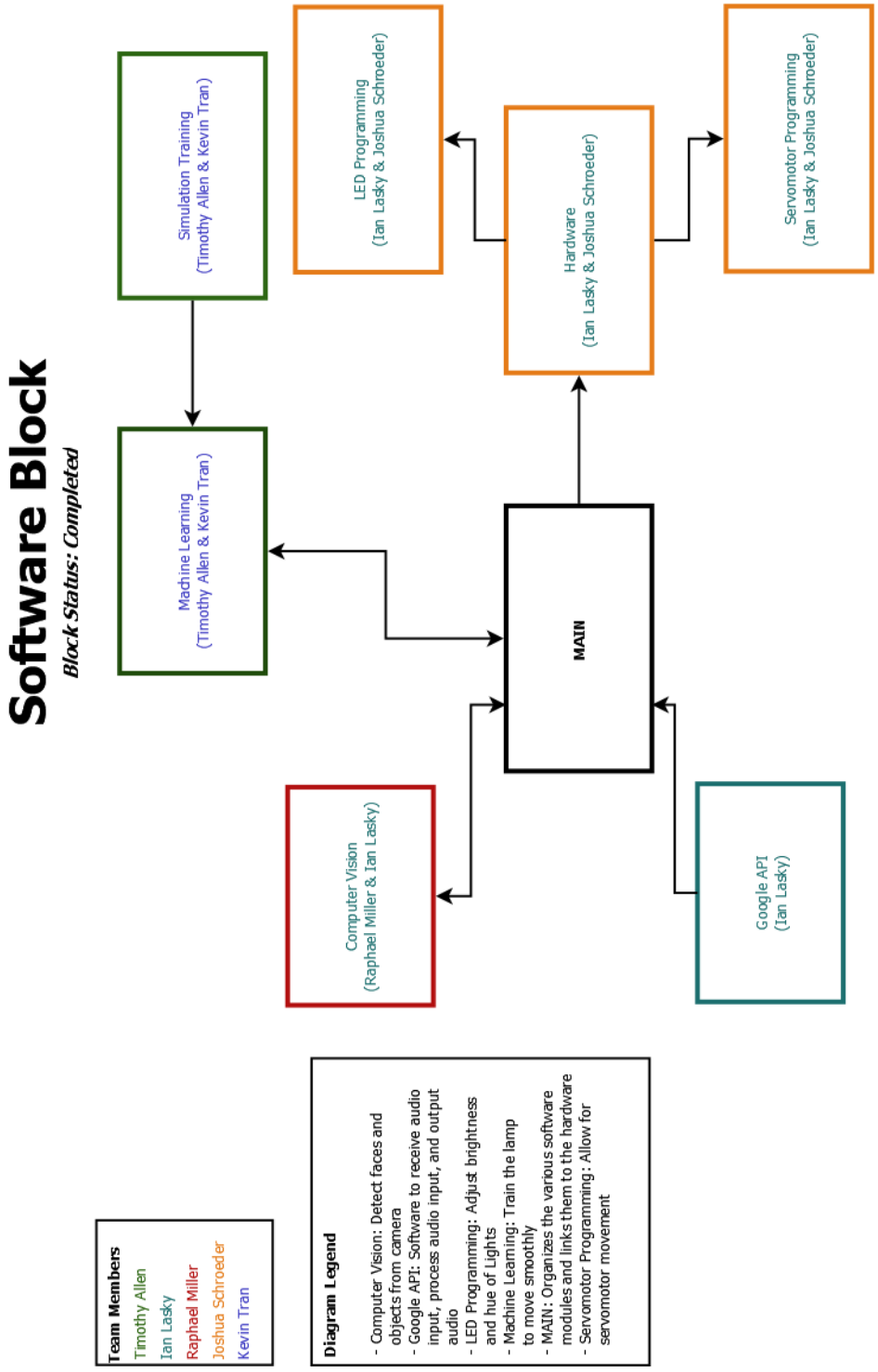

Figure 2: Software Organization Flow Chart

4.0.3 Hardware Block Diagram

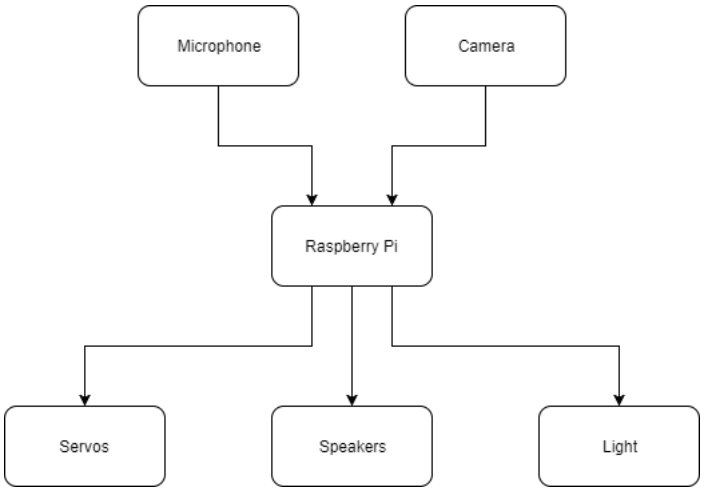

Figure 3: Hardware Organization Flow Chart

5 Specifications

Our project consists of six primary modules. Each module will be described below in a numbered list format with no particular preference to the ranking (i.e., the number next to each item merely serves as a counter rather than as an indicator of importance relative to other modules).

We define *module* as a part of a program [1]. Our program consists of four parts that must be combined in the main module to fully implement all the required functionality and features specified in 3.1. Thus far, we have accounted for five of the modules. The sixth module is the **Simulated Training** module which will directly feed into the **Machine Learning** module, which is one of the four modules that will feed into the main module to complete the animatronic lamp.

Without further adieu, here is the description of each module's intended functionality and how it relates to each requirement from 3.1 as well as its particular specifications and constraints which will describe how the module shall be implemented while taking into account the constraints and restraints it must abide by, possibly due to hardware or technological limitations, but not limited to only those limitations:

5.1 Modules

1. **Hardware**

    - The hardware shall consist of the following physical components:

Figure 4: Raspberry Pi 3 model B

– Raspberry Pi model 3B
    * The Raspberry Pi is a small computer running a Debian-based distro of Linux.
    * Specifications
        · 1.2Ghz quad-core 64-bit CPU
        · 1 GB RAM
        · 4 USB Ports
        · Wireless/Bluetooth onboard
        · 40 IO pins (digital)

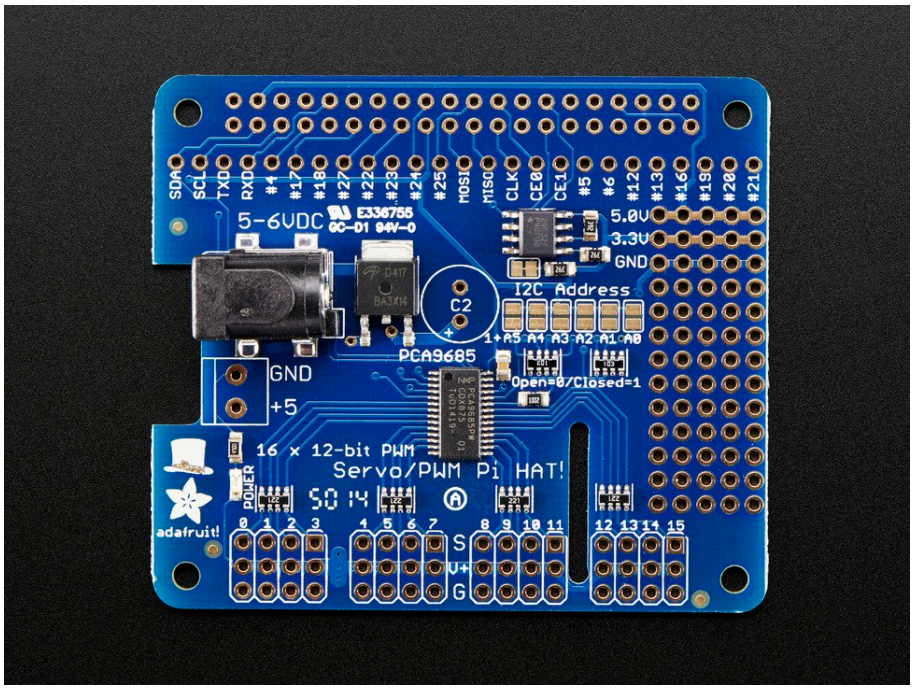

Figure 5: Adafruit 16-channel Servo HAT

- Adafruit 16-channel Servo HAT
  - HAT stands for "Hardware Attached on Top"[2]
  - Will handle PWM modulation for servo control to free up resources for other tasks

Figure 6: Logitech C525 Webcam

- Logitech HD USB Webcam C525

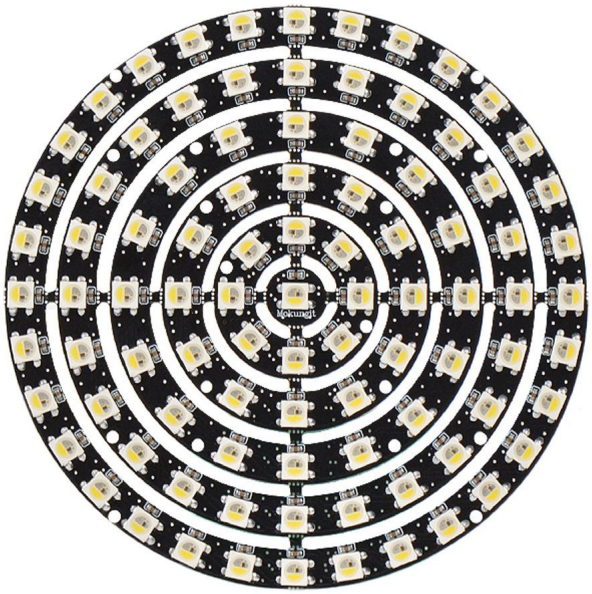

Figure 7: SK6812 RGBW LED Ring

- LED ring
  - 84/71 SK6812 RGBW LEDs depending on final configuration

Figure 8: Futaba S3004 Servo

- Servomotors
  - Futaba S3004 Standard Ball Bearing Servo
- Custom 3D printer servo brackets
- USB microphone

  - Power supply
    * 5v 15A

- Assembly

  - Because of height constraints in the lamp base, the team will be de-soldering all components except for power input from the Raspberry Pi and Servo HAT and connecting them using small wires. Being that the HAT uses a serial connection, only two wires will be required to connect to the Raspberry pi.
  - After deciding to use a single power supply for simplicity of setup, we'll be using a splitter between the Raspberry Pi and Servo HAT power input. The LED ring will piggyback off of the servo HAT's power input pins since it's a direct connection between barrel plug and power input pins.
  - The LED ring will require an 800Khz PWM signal to address, and the servo HAT can only supply up to 100hz PWM. Because of this the team will be using the SPI interface on the Raspberry Pi to address the LED ring. It does not need constant addressing, so should be a very small load on the Raspberry Pi when being changed and almost no impact when not being addressed. If this proves to be a problem we can stack an LED controller on top of the servo HAT's serial connection as a contingency.
  - The servos will attach to the lamp at places where hinges already exist to simplify implementation. The servos will be attached using custom 3D printers brackets, where the output shaft will replace a pivot hinge on the lamp.

2. **Computer Vision**

- The computer vision module contains these algorithms.
  - Camera Control

        * The algorithm shall be able to utilize all of the camera's functionality.
        * The algorithm shall be able to turn the camera on and off on command according to the process requests.
        * The algorithm shall be able to focus (if required) on objects of interest.
    - Face Detection
        * This algorithm will be able to detect a face from the front of the face or the side view of the face.
    - Object Detection
        * This algorithm shall be able to detect an object of the original specification.
        * It should be able to differentiate between each object located.
    - Object Tracking
        * The algorithm should be able to track an object within the Lamp's field of view
        * The algorithm should be able to aid the detection of particular objects of interest.
    - Object Coordinate Recording
        * This algorithm will be able to detect and interpret a coordinate system in order to calculate position, velocity, and acceleration.
    - Position, Velocity, and Acceleration Recording
        * This algorithm shall be able to calculate in real time, the position, velocity and acceleration of the object in question
        * The algorithm should have these values ready to be given to the main module upon request.

3. **Google Assistant API**

- The Google Assistant API module will contain these algorithms.
    - Google OAuth Authorization

    * The algorithm will be able to authenticate the lamp with Google's Web Servers using Google OAuth authorization system.
    * The algorithm shall be able to use multiple user accounts for multiple users in order to connect many users to their accounts.
- Google Assistant SDK
    * The algorithm shall be able to access the Google Assistant SDK in order to connect with the user for User Interface interactions.
    * The algorithm shall be able to utilize the full functionality of the Google Assistant SDK suite.
- Google Natural Language Processing
    * The algorithm should be able to listen and convert speech to text
    * The algorithm should be able to understand the user in a relevant way.
    * The algorithm should be able to respond to the user in a meaningful way
    * The algorithm should be able to listen to commands
    * The algorithm should be able to understand commands
    * The algorithm should be able to obey commands
- Google JSON Requests
    * The algorithm should be able to utilize Google's RESTful API service to process requests.
    * The algorithm should be able to send and receive information to Google's web servers.
- Custom Commands
    * The algorithm should be able to listen to custom commands.
    * The algorithm should be able to identify and interpret custom commands.
    * The algorithm should be able to use custom commands to move the camera.

  - Google Home Suite
    * The algorithm should be able to connect to the Google Home Suite
    * The algorithm should be able to interact with other devices on the network.
    * The algorithm should be able to interact with third party applications in Google Home.
    * The algorithm should be able to control other third party appliances on the Google Home Network.

4. **Machine Learning**

   - As will be discussed in more detail in 9.3, the lamp will be trained through machine learning to move its lower arm, upper arm, and head.
   - Machine learning will be able to train the lamp to yield optimal results and result in more smooth and natural movement that will make the lamp to appear more lifelike and polished. For this reason, hard-coding the lamp's physical movements was not desired. In addition. hard-coding the lamp's physical movement disallows the lamp from being able to effectively locate the speaker when being spoken to.
   - The approach to training the lamp is to first train a model of the lamp through a virtual environment.
     - This approach allows for easy, fast, and flexible training.
     - This approach will also help the developers more easily observe the lamp's training and facilitate the process of debugging.
     - Another advantage of modeling and simulating the lamp's training is the ability to create numerous virtual lamps to each give them a separate training procedure in order to determine which training procedure is the best.
   - Through simulated training, the real physical lamp will learn to set the values of its five (5) servos to move the camera to a new position relative to its current position.

- The simulated training will involve reinforcement learning to give the developer the ability to choose desired behavior and movements.
- Unity3D and Unity ML-Agents will be utilized to train the lamp model. Unity ML-Agents utilize a Proximal Policy Optimization (PPO) algorithm.
- In addition, a headless version of the lamp model will be trained in the cloud on Amazon Web Service.
- More detail can be found immediately below in the Simulated Training section.

5. **Simulated Training**

   - Although this is a work in progress, the objective is to create a Unity3D model of the lamp. The following criteria must be kept in mind while creating the model in order to realistically portray the model of the lamp.
     - All aspects of the model's size shall be proportional to the lamp. To further elaborate, for instance, the model lamp's lower arm and upper arm shall have the same proportion in size relative to one another, and this proportion shall match the proportion pertaining to that of the actual lamp. Without loss of generality, all other parts of the lamp shall be correctly proportionally relative to each other.
     - The maximum and minimum bending angles at any part of the model lamp (e.g., lower arm, upper arm, head) must be consistent with the physical lamp.
   - The model lamp will be fed an input vector pertaining to position and angle, and the lamp will output a vector representing the positions of all of the lamp's axes. Through this method, the model lamp will be trained to achieve the desired result, at which point the model lamp's training data will serve to help the physical lamp to train more quickly. More detail will be discussed in 9.3.
     - Another constraint to consider while training the model lamp is to control the speed at which the lamp moves

from one point to the next. In a virtual environment, a model lamp can move its arm at a speed too fast to be considered realistic.

- Furthermore, the model lamp must account for weight distribution as it moves, as to not fall over. While the model lamp may not actually fall over, when the physical lamp repeats this movement learned from the model lamp, the physical lamp may actually become unbalanced and tip over.

6. **Main**

- The Main Module should contain the following algorithms
  - Process sharing
    * The algorithm should be able to control the processing power of the raspberry pi unit and allocate the processing power to meet the needs of the system.
    * The algorithm should be able to control threading implementation if necessary.
    * The algorithm should be able to utilize standard control functions to operate the modules.
  - Threading Implementation
    * The algorithm should be able to utilize standard threading patterns and processes
    * The algorithms should be able to handle threading errors and issues in order to keep the functionality of the lamp during the working process.
    * The algorithm should be able to maintain all activated modules within good operating conditions while the lamp is operational.
  - Module Interaction
    * The module should be be to receive commands from other modules to be sent to the various other modules.
    * The algorithm should be able to send commands to other modules upon request of those modules.

    * The algorithm should be able to allocate resources to other modules when said modules needs them.
    * The algorithm should be able to create a barrier between direct module interaction
- Function Pass-through
    * The algorithm should be able to allow function arguments to pass through the main module.
    * The algorithm should be able to utilize all modules within operation.
- Module errors
    * The algorithm should be able to detect and identify errors within a module.
    * The algorithm should be able to limit resources to a module if necessary.
    * The algorithm should be able to ”turn off” a module if said module is not working properly.
    * The algorithm should be able to turn on modules when required.
    * The algorithm should be able to identify various errors and not interrupt the hindrance of the overall working order of the module.
    * The algorithm, with the help of other modules, should notify the user if an error has occurred.

5.2 Image of Lamp

In addition to the listing of the capabilities of each module, a photo of the lamp is crucial to include within this document in order to have a reference for design and implementation. Furthermore, the image of the lamp shall serve as a useful figure for the reader to better understand the appearance of the lamp. The lamp in the image presented below has not been modified for the project yet.

Figure 9: Image of Original Lamp

5.3 Sketch of Lamp

In addition to the block diagrams provided in 4, requirements provided in 3.1, and specifications provided above, it is useful to have a drawn sketch of the lamp itself. For the sketch to be highly useful, the sketch should include the position of the lamp's various hardware components relative to each other.

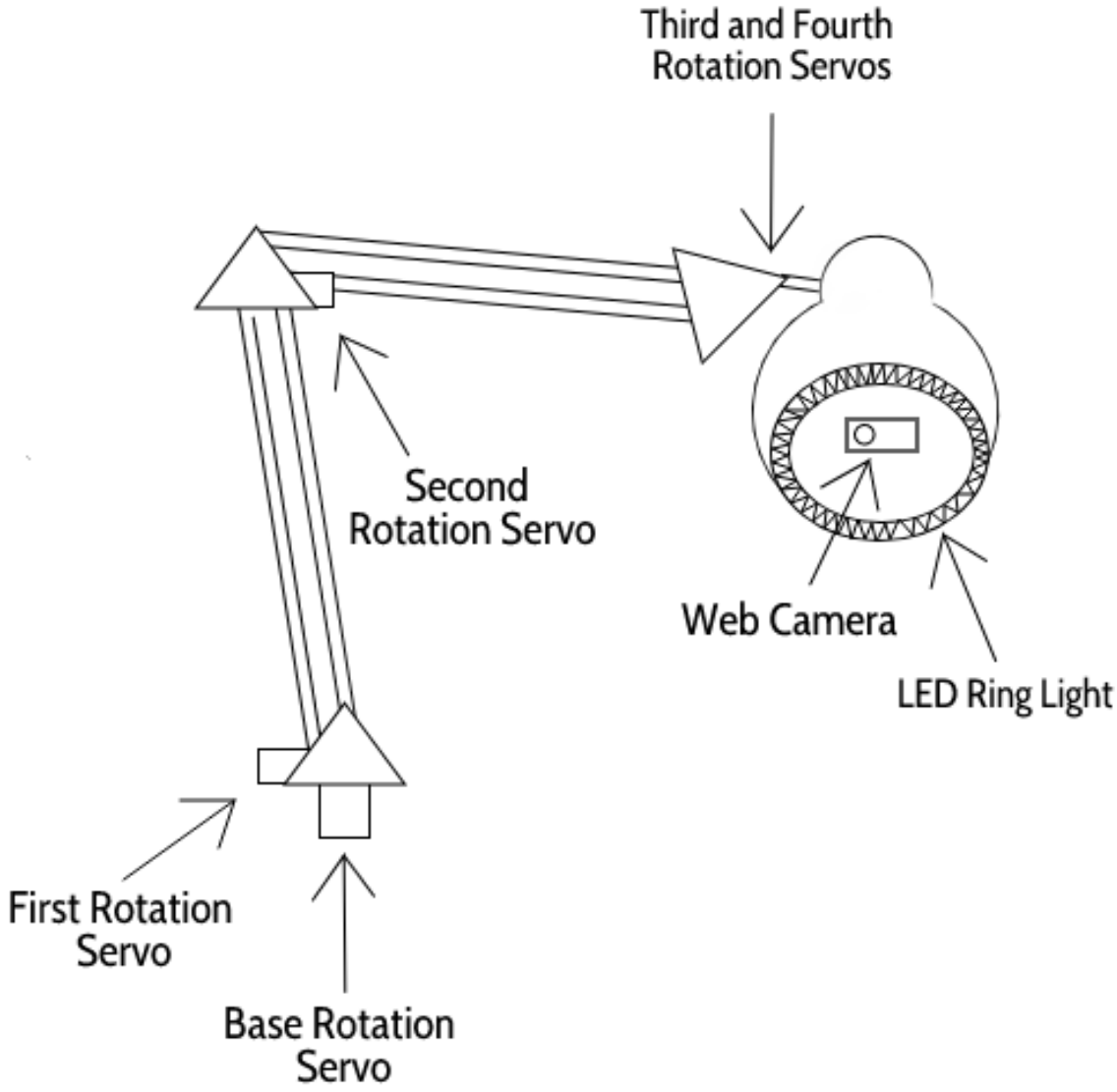

Figure 10: Diagram of Lamp

6 Budget and Financing

Table 4: Parts and Budget

Part	Price	Quantity Needed	Quantity Have	Quantity Left	Price Needed	Price Have	Price Difference
Lamp	$20	1	1	0	$20	$20	$0
Servo Motors	$14	5	3	2	$70	$42	$28
Raspberry Pi 3	$35	1	1	0	$35	$35	$0
Webcam	$22	1	1	0	$22	$22	$0
Servo HAT	$17.50	1	0	1	$18	$0	$18
Power Supply	$13	1	0	1	$13	$0	$13
LED Ring Light	$14	1	0	1	$14	$0	$14
Microphones	$6	1	0	1	$6	$0	$6
Speakers	$8	1	0	1	$8	$0	$8
Power Splitter	$3	1	0	1	$3	$0	$3
Adapter	$2	1	0	1	$2	$0	$2
TOTAL					$211	$119	$92

7 Proposed Project Milestones

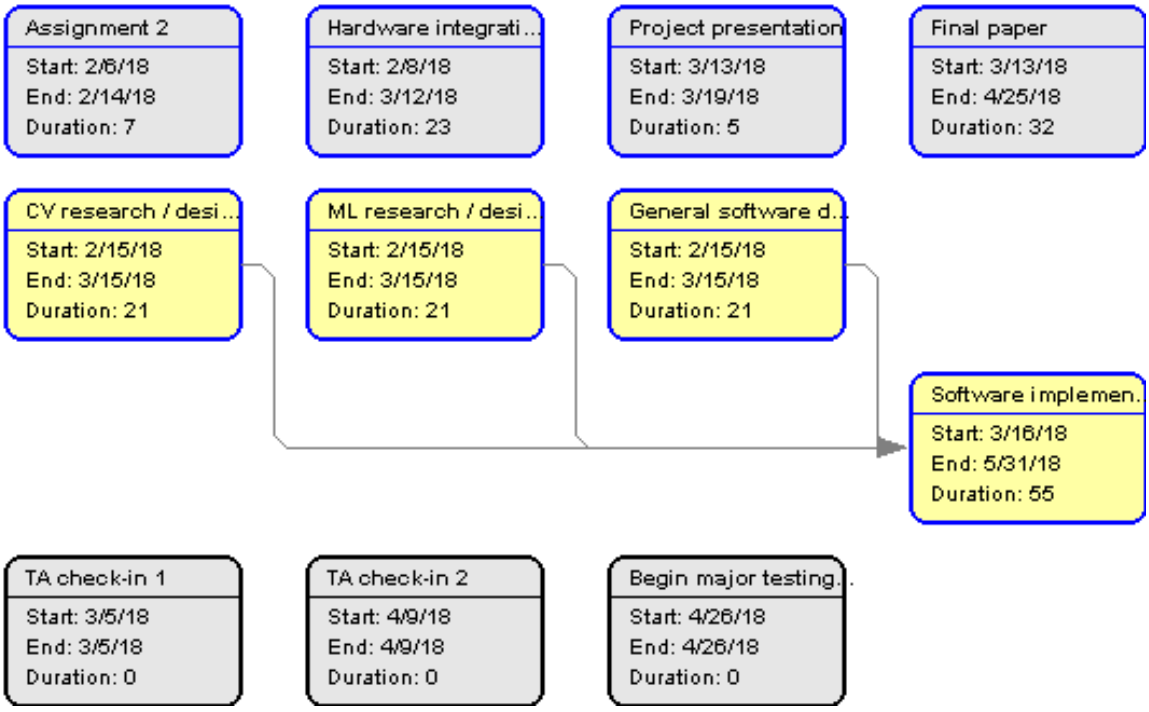

Figure 11: PERT Chart of Proposed Project Milestones

8 Administrative Technologies

The purpose for this section is to further describe our method and frequency of communication, coordination, and teamwork. Communication is an integral aspect of any team, and good communication leads to good understanding of both administrative duties and a comprehensive technical understanding of the overall system design. From communication springs forth coordination and teamwork which will yield the fruits of our labor: a fully functional animatronic lamp.

8.1 Medium of Communication: Slack

As the medium for all of our communication, we have chosen a instant messaging application known as Slack [3].

Figure 12: Logo of Slack

In addition to supporting instant messaging services, Slack also supports a wide variety of features that made this software stand out from the rest. In particular, Slack supports to-do lists, pinned messages, member pinging, designated channels for certain types of content, integration with GitHub, Google Drive, and Trello! Furthermore, another appealing factor of Slack is the lack of necessity to download the application on a desktop. Slack can be accessed via a web browser and has the capability to send desktop browser notifications to the user. Slack also has both Android and iOS support with comparable functionality to that of desktop Slack.

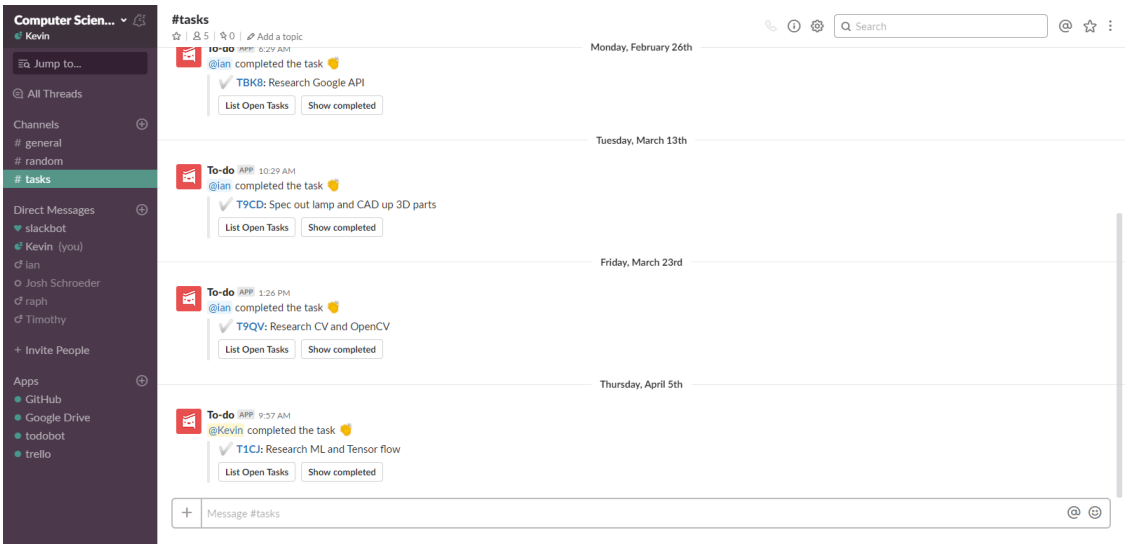

Figure 13: Slack's *tasks* channel to keep track of duties and responsibilities.

8.2 Task Delegation: Trello

Although Slack offers a task-keeping system to its users via the usage of the *todobot* bot, the system is not advanced enough to suit our needs. Therefore, in conjunction with using Slack, we have also been using a project management software application called Trello [4].

Figure 14: Logo of Trello

Trello uses the concept of *Kanban boards* to help software development teams organize their information and responsibilities. As defined by Leankit, a Kanban board is a "work and workflow visualization tool that enables you to optimize the flow of your work [5]." Traditionally, physical Kanban boards were used instead of virtual Kanban boards.

Sticky notes were stuck onto a whiteboard in a certain pattern to create a Kanban board. These sticky notes communicated status, progress, and issues [5]. The graphic below demonstrates an example of an effective Kanban board [5].

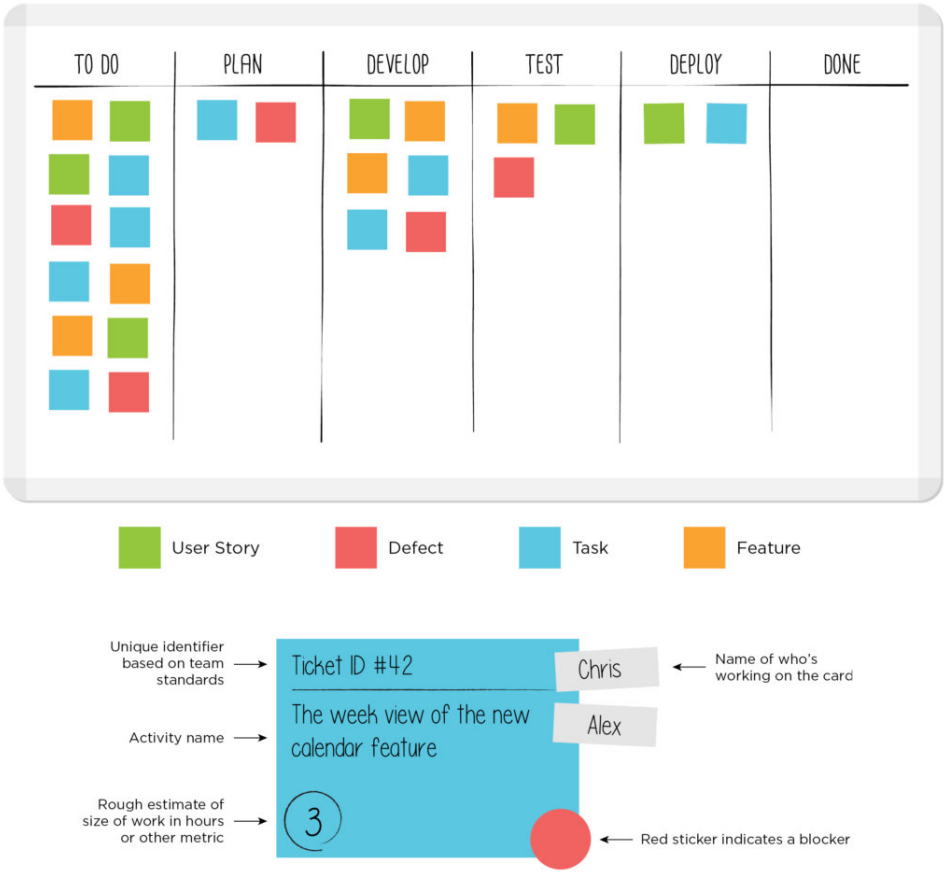

Figure 15: Example of an Effective Kanban Board

As we begin to delve more into the actual coding, we will begin to see more usage of Trello, but for the time being, attached below is a screenshot of all of our Kanban boards on Trello.

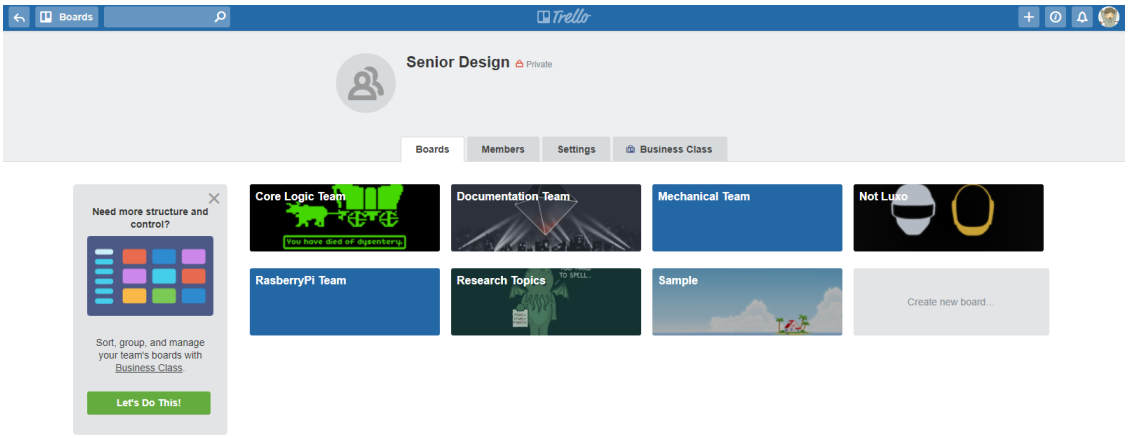

Figure 16: Showcase of Kanban boards on Trello

8.3 Document File Storage and Non-Formal Document Collaboration: G Suite

Well-known, reliable, and convenient, G Suite has proved to be incredibly helpful in aiding us with document creation and collaboration as well as storage. G Suite is a collection of software tools [6]. In particular, we are focused on three of G Suite's tools, namely: Google Drive, Google Docs, and Google Slides.

We have designated Google Drive as our choice of hub for the storage of most of our documents. On Google Drive, we have stored brainstorming ideas, meeting notes, quick collaborative technical documentation — which is later transferred over to Overleaf (see 8.4).

8.3.1 Presentation Slides

In addition, we have also stored our slides for the initial project pitch presentation to the class, and our work-in-progress Critical Design Review (CDR) slides are also on Google Drive. These slides were created with Google Slides for ease of collaboration. Beneath is an example of one of our slides developed from Google Slides with a modern, sleek, and elegant theme.

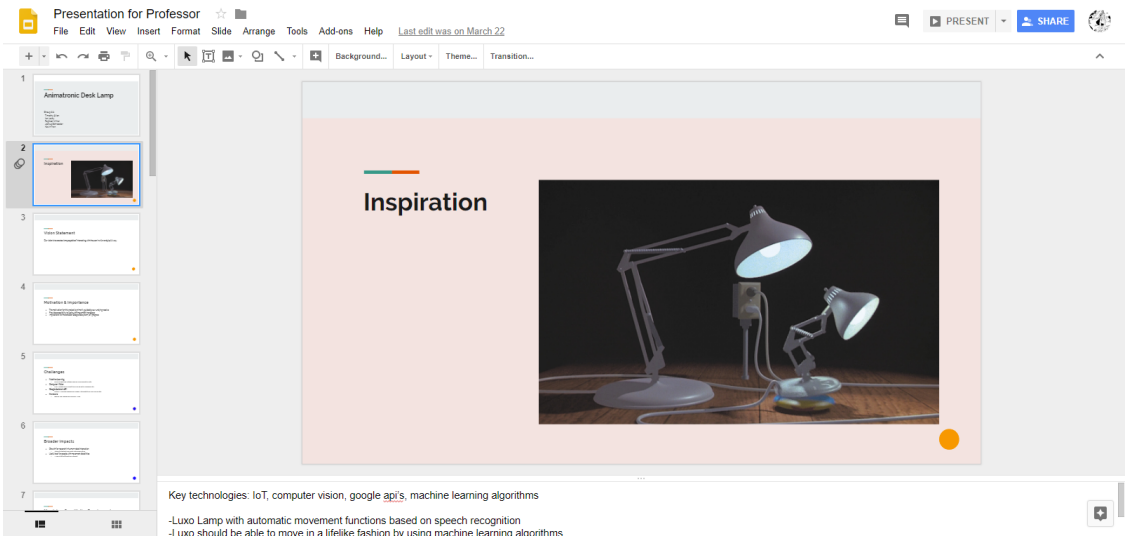

Figure 17: Image of inspiration slide

8.3.2 UML Diagrams with Dia

UML class diagrams for the entire system as well as each individual module and component is also being stored on Google Drive. Most UML class diagrams were created with a diagramming software called Dia. Dia is a free and open source software that has support for both Windows and Linux, which is why we chose to use this software. Below is a screenshot of one of our UML diagrams being developed in Dia.

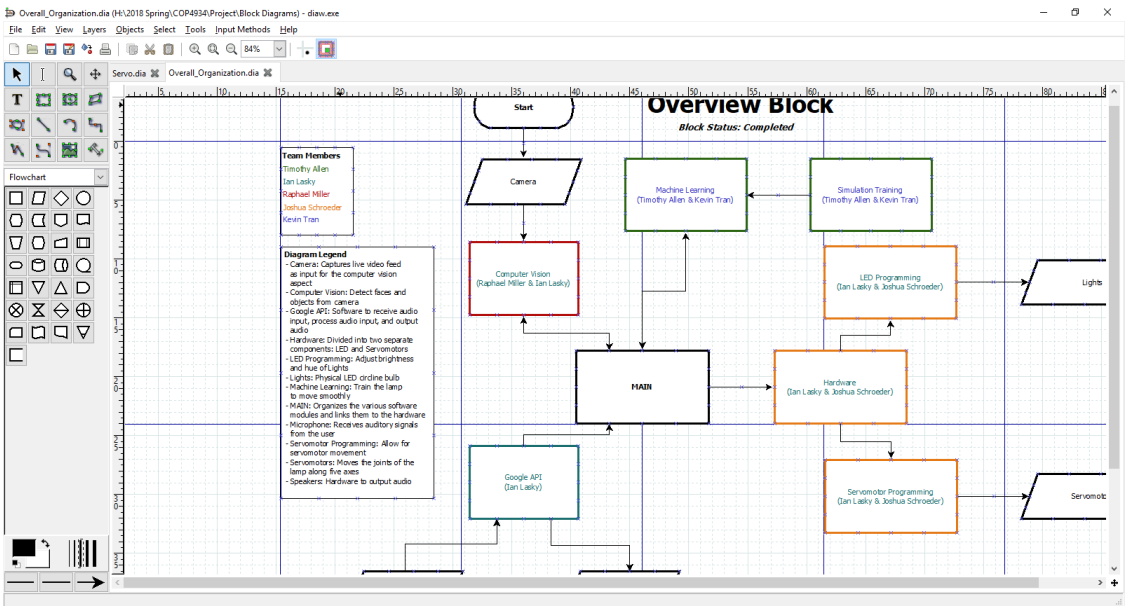

Figure 18: Overview Block UML diagram being developed in Dia

And for our convenience, copies of the lecture slides from the course is also being kept on our Google Drive. For backup reasons, a copy of our proposal paper is kept on Google Drive in the form of a portable document format (PDF) file.

8.3.3 Bootcamp Worksheet

Our team values worksheet from the Senior Design bootcamp is also kept within our Google Drive folder to serve as a reminder and reference for all of us.

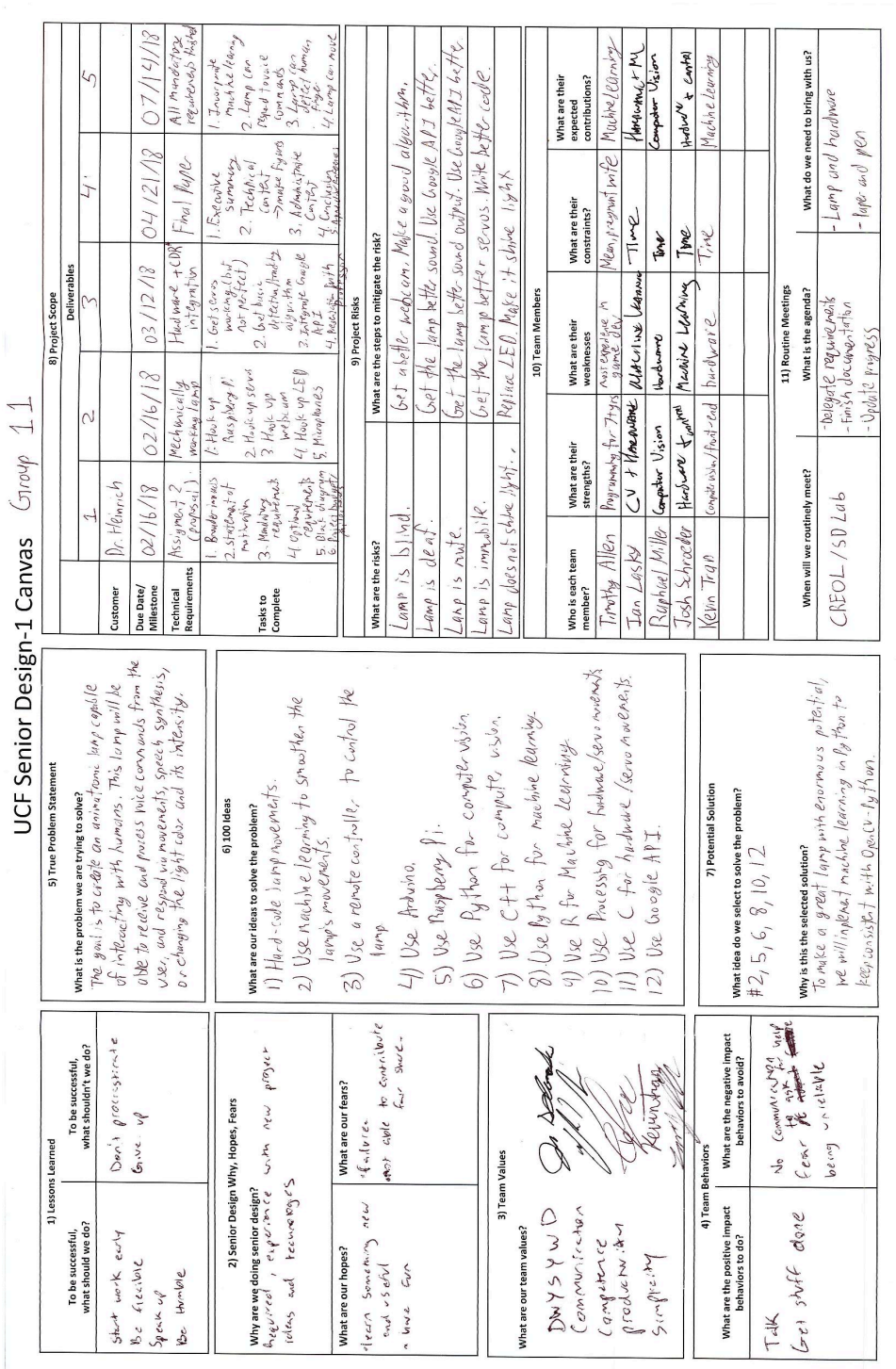

Figure 19: Display of our worksheet from the 5-hour Senior Design bootcamp

8.4 Formal Documentation Write-up: Overleaf

For any formal documentation, such as the Proposal or the Final Design Document, our team has been collaborating on Overleaf, which is an online LaTeXcollaborative writing and publishing tool. Our choice for using LaTeXto create our final design document is rooted in our beliefs of submitting a highly polished paper in appearance. LaTeXis also highly customizable and allows for great flexibility with the formatting of our document. In particular, LaTeXprovides us with an easy way to include and replace figures and diagrams as the occasion occurs. LaTeXis also able to help us easily point the reader to another section of the paper via coded integration of section headers and the table of contents.

In addition, we are also using BibTeX as our reference management software to format our citations and list of references. BibTeX enables us to quickly and easily cite our sources within the text without the necessity of scrolling down to the references list to find the source we are looking to cite. BibTeX also automatically organizes our list of references by order of appearance in the paper; this automation greatly improves our workflow. Together, BibTeX and LaTeXform the backbone of our Final Design Documentation.

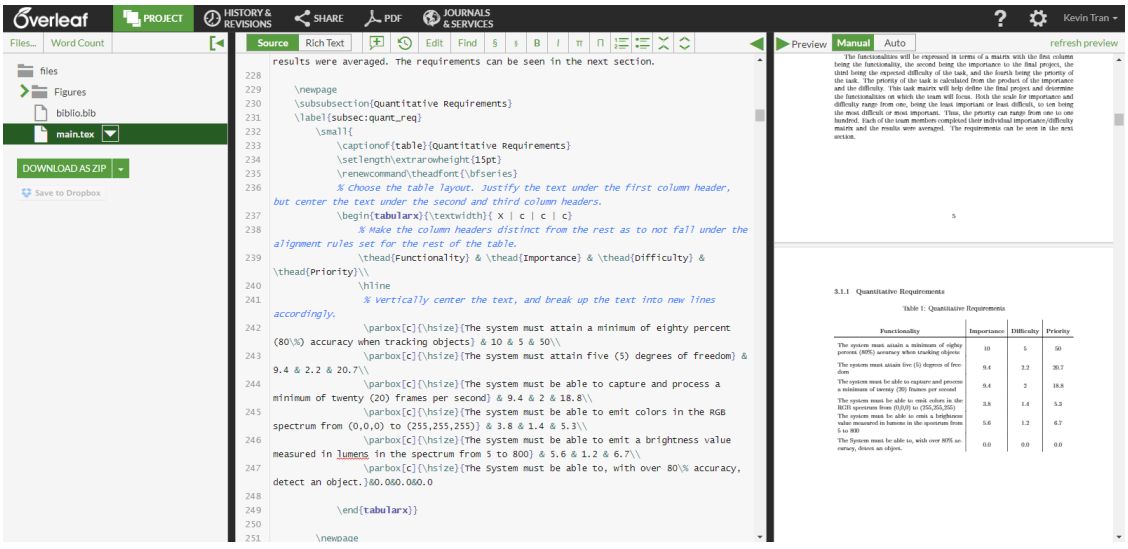

Figure 20: Showcase of our Overleaf workflow setup

8.5 Version Control: GitHub

Our choice for code collaboration and version control is GitHub, which is a web-based hosting service for version control using git [7]. It is important to note that *git* is different from GitHub itself. Git is a version control system for tracking changes in files and coordinating work on those files among multiple people [8].

Our GitHub repository — defined as a directory or storage space for projects — is accessible here: `https://github.com/imlasky/Animatronic-Lamp` [9]. First and foremost, our repository contains a *README.md* file which gives a brief overview of our project. Currently, within our repository, we have three folders. The first folder is dedicated to containing formal documentation and block diagrams for ease of access to the reader. The second folder contains computer vision code, and the third folder contains Processing language code for a model of the lamp.

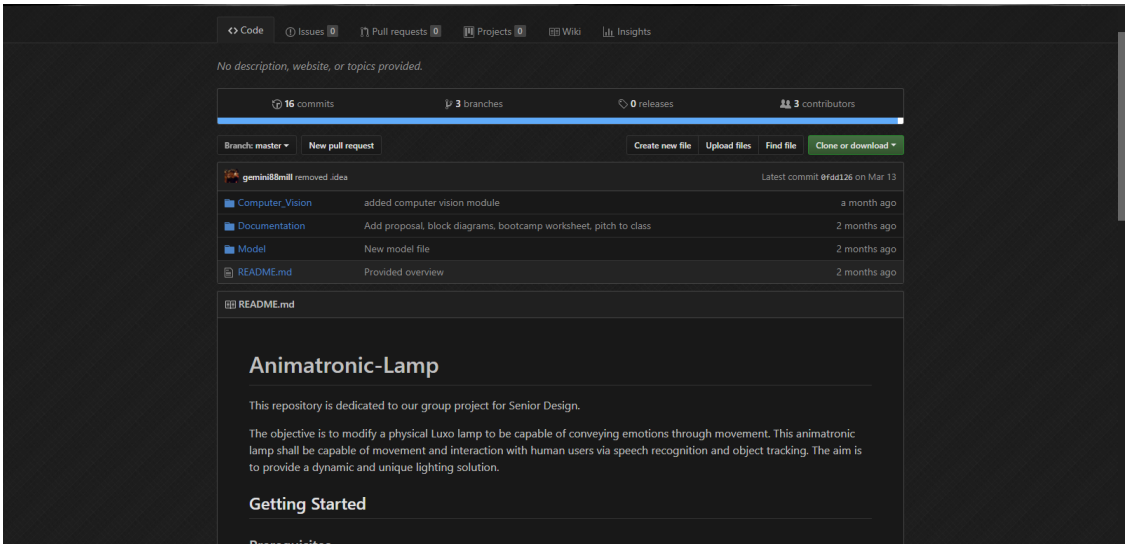

Figure 21: Overview of `/imlasky/Animatronic-Lamp` repo

9 Project Implementation

9.1 Hardware and Movement

The fundamental basis of the project at hand is hardware. The physical lamp is required to look, move, analyze, and respond to external stimuli. The requirements directly correlating to hardware will be discussed within this section and result in a definitive explanation of the implementation choices and why the aforementioned choices are optimal. The implementations are tested according to the procedure outlined in Section 10 and the results of this testing will be discussed in Section 11.

9.1.1 Movement

The most basic function of the lamp is movement. The lamp itself is comprised of five (5) distinct points of movement, one for each of the axes of movement:

- Entire lamp body rotation
- Small scale forward and backward movement
- Large scale upward and downward body movement
- Small scale leftward and rightward head movement
- Small scale upward and downward head movement

At each point of movement, the component is only allowed to move within one plane of motion. Thus, there must exist only five (5) degrees of freedom within the lamp. As such, the lamp requires five distinct pieces of hardware capable of setting and maintaining a prescribed position.

An intrinsic part of the design is moment balancing and torque correction. That is, at each point of movement there is an associated torque required to move along that degree of movement. Therefore, to rotate one axis, the driving device must overcome that of the torque. In addition to overcome the torque requirements, the device must be able to maintain to a fine degree its position with minimal slipping.

Two natural solutions to this posed problem are the use of stepper motors or the use of servomotors. The main difference between a servomotor and a stepper motor is the behavior on power up. A stepper motor will go to a predetermined zero position while a servomotor will move to any input position without returning to a zero position. This fundamental difference drove the decision to choose servomotors over stepper motors.

In addition, both hardware and software applications were more available for servomotors. The servo HAT is an addition to the Raspberry Pi specifically designed for the PWM output smoothing of a servomotor. This will free up the Raspberry Pi for other things as it will not have to maintain each servo's position repeatedly since the HAT is performing that function autonomously.

The servomotor is capable of high torque applications allowing for it to be well-suited for moving and maintaining set positions. Additionally, smooth movement is facilitated by the availability of a continuous motion. That is, the servomotor is unique in that it allows for continuous steps from zero (0) to one-hundred-eighty (180) degrees. The movement of the servo is controlled through a process called Pulse Width Modulation (PWM). PWM allows for the digital interface to create an analog signal for the servomotor to interpret as a voltage and turn to the prescribed position. A diagram of the details about PWM can be seen in Figure 22 The servo motor of choice was a Futaba S3004 Standard Ball Bearing Servo due to limited cost, availability, and torque specifications matching the needs of the project.

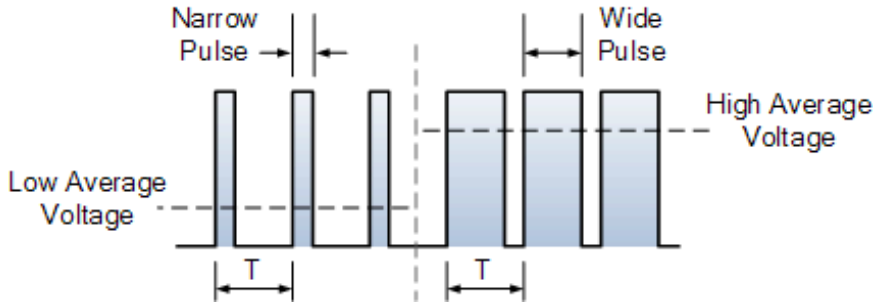

Figure 22: An example of the concept behind pulse width modulation.[10]

Assembling and connecting the servos to the lamp presented a challenge. To construct the lamp required servos at each point of rotation. Because there were no natural connections to the lamp, brackets and connections needed to be designed to fix the servomotors to the lamp. The lamp is was deconstructed, and all applicable measurements were made using a VINCA DCLA-0605 digital caliper with an error of ± 0.02 mm. Additionally, measurements of the servomotors were taken in order to design a servomotor mount.

The easiest and most intuitive approach to mounting the servomotors to the lamp was to model the parts in a 3D modeling program such as SolidWorks, 3D print the models and test fit the motors to the lamp. This process allows for rapid development, adjustment, and testing of servomotor mounting. There are multiple mounting points that need to be considered for the lamp. Each is slightly different and so a mount must be designed specifically for each. In Figure 23, one servo mount is seen for the lower hinge of the lamp.

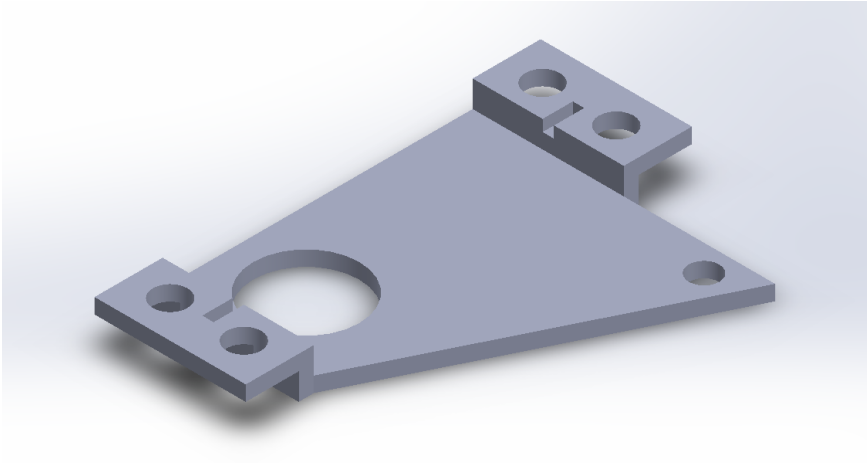

Figure 23: The mount designed to attach the servomotor to the lamp at one location.[11]

A more intensive design task was to design the mount responsible for the head movement. This mount must allow for two axis movement. That is, the head must be able to move up, down, left, and right. So, one servo will essentially be linked to another. This linkage is vital to the project as many of the actions and motions will be through the movement of the

Once the servomotors are affixed to the lamp, the task is then to determine a method to distribute the torque efficiently and consistently to the moment arm. The main contributor to the torque delivered by the servomotor is the shape of the connection between the servomotor horn and the servomotor. As seen in Figure 24, the gear-like shape allows for increased surface area and locking between the servomotor and the horn. This allows for minimal slipping and more efficient torque delivery to the servomotor horn. This is paramount when designing the connection between the servomotor and the lamp arm. Initial tests were conducted attempting to discern whether or not shear friction from a screw into the servomotor would provide enough leverage to turn the arm and it was proven insufficient. As such, the team decided to modify an existing servo arm to attach to the lamp arm allowing for minimal slipping and full torque delivery to the lamp.

Figure 24: A common servo design noting the geared connector at the top.[11]

9.1.2 Illumination

By nature of the name of the project, the lamp must function as just that: a lamp. Therefore, an illumination source was required. The light bulb and assembly that was stock with the lamp was unable to be utilized for two-fold: there was no built in way to control the luminosity of the bulb and there was no space for electronics, specifically the camera, to be mounted within the head of the lamp. Therefore, stock electronics needed to be removed from the lamp.

There were multiple choices of illumination source to be considered, including:

- LED Ring Light
- LED Circular Strip
- Circular Fluorescent Bulb

The requirements indicating that the lamp must be able to convey emotion led to the possibility that changing the output light color to

indicate mood could be implemented. This eliminated the possibility of the use of a fluorescent bulb. In addition, the versatility and size of a fluorescent bulb proved to be incompatible with the task at hand. Therefore, the choice was limited to the LED ring light and the LED circular strip.

Between the two LED choices, the main differences are that the ring is one singly addressable piece of hardware. That is, it may only be one color at a time. Additionally, the average cost of a ring led is far beyond that of the budget of the group. This too eliminated the possibility of utilizing the ring.

Therefore, the best option for illumination is an individually addressable LED circular strip. The strips are low cost, can be made to be variable radii to fit the lamp appropriately, and can vary in both brightness and color allowing for the respective requirements to be met.

Furthermore, LEDs are known to be energy efficient. This is in favor to the project team as the controller that will be powering and controlling the output of the LEDs will also be performing calculations regarding the camera, and servomotor positions. Therefore, the least amount of computing and power requirements that are needed for the illumination the better. Additionally to lessen the load of controlling these lights, a hardware shield could be used between the Raspberry Pi and the strip's data input so we don't have to dedicate a thread and hardware interface to controlling the lights. Given the demands of the project, this will be decided later.

9.1.3 Vision

One of the two inputs to the system is the vision system. Providing the system vision allows for tracking and detection of desired objects. This ability is vital to the success of the project and thus must be carefully considered.

The vision system is in essence a web camera that is connected to the main processing board. Modern web cameras are capable of sending

high definition video frames at approximately thirty (30) frames per second. This combination of frame rate and high definition would allow for better tracking, identification, and other vision implementations such as velocity and acceleration inferences, discussed more in Section 9.2.

The choice to position the camera within the head of the lamp was a natural one. This would allow simpler coordinate changing later in the processing as well as allow the team to model the view of the lamp similarly within the 3D model discussed within Section 9.3. The primary reason for positioning of the camera in this fashion allows the lamp to directly look and position itself for towards the object of interest, whether it be a person, face, or another object. This has the drawback of the lamp only having knowledge of things within its frame, but thought a method of scanning or remembering where things are at positionally we believe this can be overcome.

9.1.4 Processing

We were faced with a serious problem when thinking about module integration and how closely integrated they need to be to do their job correctly and efficiently. There are many possible solutions, but the ones that we narrowed it down to were direct call integration, where each module receives and outputs from all modules require and mediated integration, where we have one ”master” module manage the inputs and outputs of all of the other modules. One of these has a significant performance cost over the other, with the ”master” module method having the possibility to require much more overhead. After considering this, we thought about the amount of work it would take for each module to handle its own input and out and found that it would take a multiplicity of the work to do that rather than have a single module handle it due to thread-to-thread communication and waiting. This could be compounded depending on the load of each module and cause a runaway waiting period, then a crash. Rather than have each component tightly integrated, we've decided to go the central processor route where there is one main module that handles input and output

of the rest of the modules. This will allow easier development of individual modules as we can tailor the main interface to our needs and have a single unified interface. This will help prevent conflicts of state between modules having multiple inputs and outputs between one another as well as thead timing between several different threads. We believe this is the best option given that we have enough processing power from our quad-core Raspberry Pi 3b+.

9.2 Vision and Detection

Computer Vision is a vast field that the computer vision module will base most, if not all, its concurrent processes. By Definition [12]

Wikipedia: Computer Vision

> **Definition 1:** Computer vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do [12]."

For the lamp, one of the most important aspects of the design is the implementation of the vision and detection system. The main function for the lamp is not only to move and shake around, but to recognize objects and have more then just a novelty value to the user. Even if the focus is only novelty, the vision and detection systems are vital to providing the information for the system to function properly.

The lamp is essentially a system that can move in different ways to allow the user to think that the machine is "alive". The vision module will provide vital data to the other aspects of the system in order to operate start up functions, allow for the machine learning algorithm to create a usable movement model, and detect users and interact with them in a way that would otherwise be impossible without this component. This module will also have most of the computing power of the computer allocated to it since, most if not all of the time that the lamp is turned on, the camera will be on and be required to almost immediately to detect and locate various objects. This will require constant runtime in order to work properly. The system must also be optimized in order to comply with the quantitative and qualitative requirements (3.1.1 and 3.1.2). The core of the design will be based on the necessity of threading to offer aspects of control within the system.

9.2.1 OpenCV

Figure 25: OpenCV logo

OpenCV is the main driving force of this section. OpenCV as stated on their landing page:[13]

> OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Since our entire design flow will be done in python, OpenCV should have no major difficulty in communication with any aspect of the module flow and design implementations.

The library is used under a Linux base using rasberrianOS (raspberry pi Operating System) the library is built with the OpenCL platform. This allows for the GPU of the system to be involved within the workflow of the system by default. Although, raspberry pi does not include a dedicated graphics suite, if there is ever an implementation in the future then the transition should be easy to append to the overall system functions.

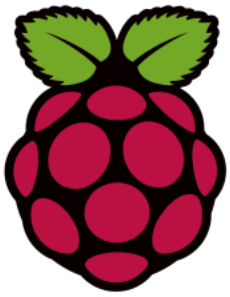

Figure 26: Raspberry Pi Logo

We have a few aspects of the design that are to be discussed within this section. These are facial detection, object detection, velocity and acceleration inferencing, and threading. Within these sections there is a solid foundation of how the design is implemented, how the methods and classes are formulated, and how the system is implemented in comparison to the rest of the modules discussed in this document. According with the design plan of the module system that is implemented for the project, each module acts independently from each other and the only connection between each module is the main function.

The main function acts as a communication function for any aspect of the system that needs to communicate with any other system (i.e. providing coordinate information to movement module). Therefore, the entire Vision and Detection module can be viewed as one major function within the system. The input that the vision and detection module expects is the operation of the camera and consistency of the camera feed. The overall goal for the computer vision module is to first establish a constant video feed for the computer vision module to analyze, then once that has been established, use that feed to compute and detect objects and users, independently from one another. Once that is established, take the relevant information from that process and send the data to the main function for data collection or further processing. The output of the computer vision module should be minimal. If the system has an object in view the system should have a detection value ready to return, with an x,y coordinate to display. The system will also be able to provide a constant stream of velocity and acceleration values (as needed for the hardware module). While in operation the module

should be able to operate with a maximum of three threads. One for the camera feed, the second and third for the sending of specific data requested by other operations in other modules.

9.2.2 Camera Process Flow

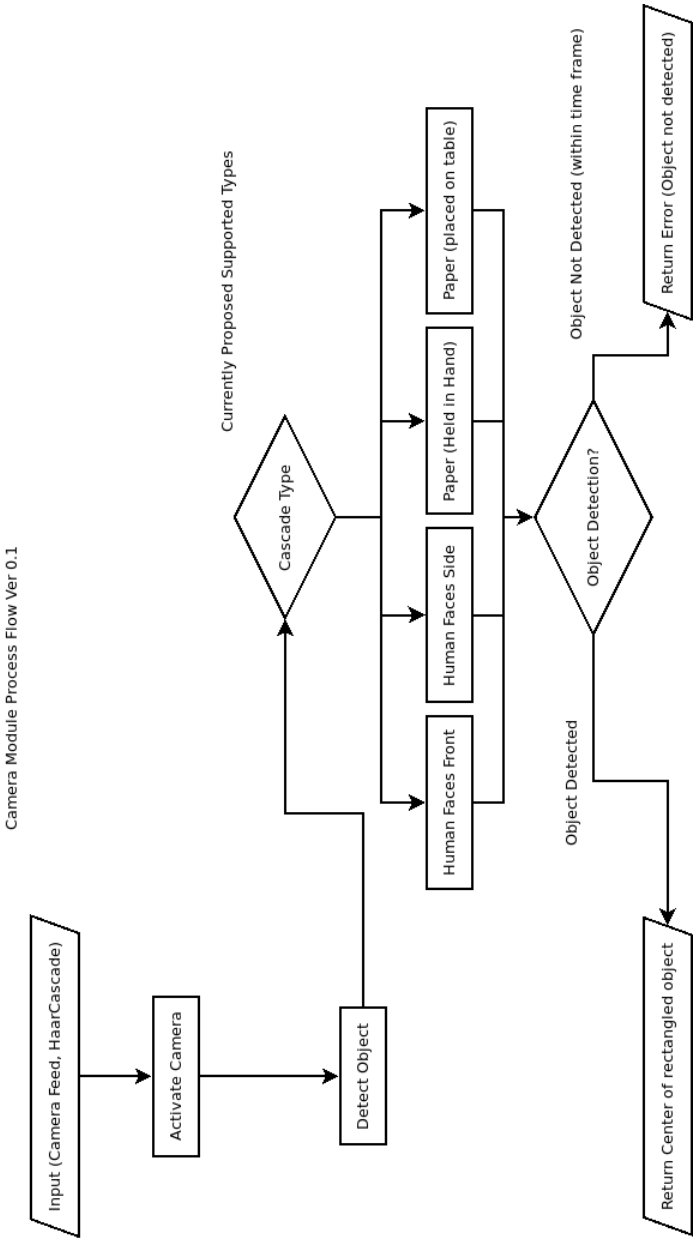

Figure 27: Camera Process Flow

From the figure above we can visually view the camera process flow. The commencement of the process assume that the camera is turned on and its ready for use. Upon activation of the camera the module, by default attempts to recognize one of four different categories of objects: frontal face, profile view of face, paper, and pens. This is the preliminary categories of detection and tracking portion of the module.

If the module detects one of the four objects displayed within the flow, the relevant values are then sent to the attribution globals within the class. If nothing is detected the class attributes will remain at default. Currently there is no method to reset to default within the process flow. This might cause an error if other modules gathering the data wish to make calculations on present time events. Further testing shall be concluded if this becomes an issue.

Each object module will have its own set of attributes, with face occupying both the front and side face cascade filter. [14]

During the module, the detection software is in a constant loop. Only ending if a value has been collected and a "kill" command has been issued from another module or the main function. This allows for a constant collection of relevant detection data to be utilized.

In the process of the camera module, we have a series of functions that will aid to the overall flow of the design. The camera has two entrance points. Either a command from the main module to activate the the camera, or a call from the main method to collect the information in order to use in other modules. The input will activate the camera module and either activate the camera entirely or have the camera in a standby mode. 9.2.16

Once the camera has received a command, depending on the command type certain functions will be performed. For example, if the camera is in motion and needs tracking information to be sent to the main module. The camera will have an array of functions at the main module's disposal and will be able to access them at any point during the activation of the camera module. Additionally, the camera acts as a data gathering point for other module systems. A few examples include:

- object tracking
- object localization
- object detection
- facial recognition
- velocity calculation
- acceleration calculation
- depth calculation

Although these samples are what the main camera would be geared towards, it does not preclude the camera from having updated features in order to better accommodate the system based on the needs of the user, and/or user base in the case of commercialization.

The first prototype model of the camera will have four unilateral methods for detection of certain objects. These objects are:

- front of face detection
- side of face detection
- paper in hand detection
- paper on desk detection

These detection methods will be used to identify items relevant to the camera module and useful to the lamp. Specifically the facial detectors will be used in tandem in order to track and identify a user while the camera is in "tracking" mode. In order for these systems to work effectively, the systems built in are very complex and require a lot of additional libraries and dependencies within the project structure. Therefore it is necessary to create a system for easy look up and localization. 9.2.3 This process will allow for easy look up, not only within

the file system but also with the overall performance of the camera module.

Finally, the camera will produce values according to the function called by the main module. It is good to note that the file system does not allow any module to directly interact with each other in order not to interfere with processes within the system and causing additional errors. Any call from another function must first be sorted through the main module and sent to the camera module. The camera module then sends back the result to the main module where it can then be utilized by the other processes. For example, if the lamp wants to turn the light on the steps in order to have the lamp turn on the camera are as such:

1. Microphone activation (this step is assumed since the lamp is turned on and ready for input from the user.)
2. Microphone sends information collected through to main in order to have the process run through the Google API module.
3. Main module sends data to Google API module.
4. Google API module interprets the data and formulates a process in order to turn on the camera of the camera.
5. Google API sends information back to main in order to turn on the camera.
6. Parses the feedback from the camera in order to prepare for the camera.
7. Camera module receives the command and activates the camera using the proper function.

From the example above we see that the main method is evoked multiple times. Although this initially seems to be inefficient, the reality is this this type of process will allow the system to check for general errors at multiple times during the overall process. That way the lamp will be more equipped to handle complex tasks without making an error in the pathways of said lamp. This means a more beneficial and better experience for the end user. (Section 9.5)

9.2.3 File Organization

File organization is an overlooked but important concept that should be discussed in detail. There are many styles that can be used and considering that all modules are relatively independent, we can view each file system within itself for its merits and corrections. From the Hitchhiker's Guide to Python: [15]

> By "structure" we mean the decisions you make concerning how your project best meets its objective. We need to consider how to best leverage Python's features to create clean, effective code. In practical terms, "structure" means making clean code whose logic and dependencies are clear as well as how the files and folders are organized in the file-system.

From this block we can get a good view of the methodology within the file system. From a top level in the Computer Vision Module Folder we have the following directories:

- /docs
- /tests
- /vision
- LICENSE.txt
- README.txt

Top Level Directories The top level directories allow us to us a tentative view of the current filesystem structure. Starting from the two files README and LICENSE. These are the these are the relevant information for outside associates to debug and start the module in a stand alone fashion. The LICENSE is all the relevant information to the various copyrights for certain aspects for the code. For example:

The Haar Cascade filter library (Section: 9.2.6) for faces is owned by The Intel Corporation and therefore requires a license in order to be used. Any information pertaining to the Computer Vision Module will be placed in the License and Readme sections of the file directory. [16]

/vision The vision directory is where the main program is housed, or the actual computer vision module. Since it is multiple files it contains its own directory. Otherwise it would be at the base level of the project along with README and LICENSE.

/tests Tests are where the testing suite, files, and results are stored. Any errors are also stored until final release of the system.

/docs Docs are where any files related to the documentation of the module is stored. Examples would be program notes, logs or links to the wiki or other online documentation.

9.2.4 Haar Cascades

For the facial and object detection, Haar Cascades are being utilized. Haar Cascades can be used to detect any specific object using a vast amount of positive and negative images. Positive being images with the object that the algorithm is looking for, and negative being without the object. The Haar cascade is a machine learning based approach where a cascade function is trained from the images described previously.

A Haar-like feature can be defined as the difference of the sum of pixels of areas in the rectangle. This can be at any position within the original image. A Haar-like feature has the basic premise that certain spaces of rectangles are darker then other portions of the same rectangle within the image. This is known to occur in all images and therefore can be used to detect various objects. [17]

9.2.5 Haar Feature

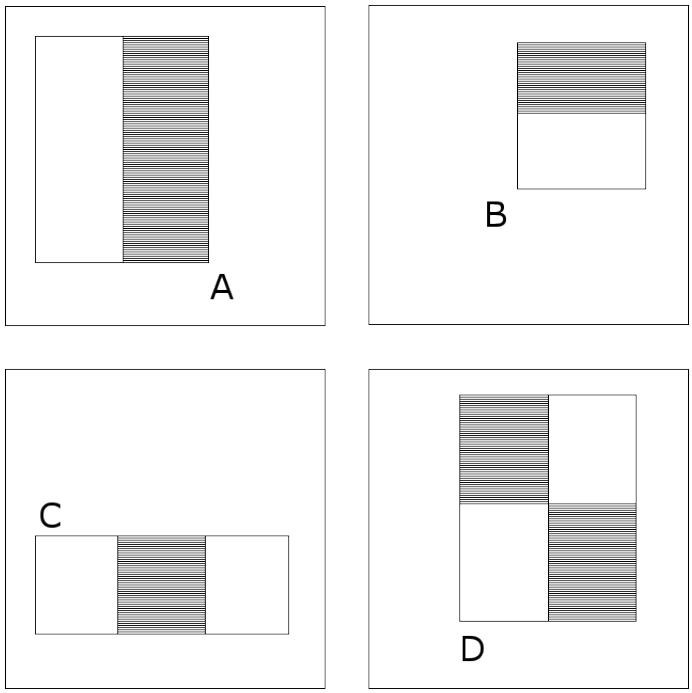

Figure 28: Haar Feature

These features are used to train the algorithm to implement the function to find various objects. The benefits for this method is that the algorithm (Viola-Jones Algorithm) is robust, done in real time, and only detection.

The main benefit for use in the application for the automated lamp is that in real world examples only 2 frames per second need be processed with very low false positive rates.

9.2.6 Facial Detection

Facial Detection and recognition, are operated by and utilized by the camera system installed in the system. A brief overview would be that the camera utilized will a standard Logitech web camera operated by the raspberry pi module. The end result will be, as discussed above

the x,y coordinate system for the object's location, and for the velocity and acceleration for the object in motion [18].

Figure 29: Logitech web camera used for the implementation of the project

Furthermore, Facial recognition (along with object detection 9.2.10) utilize OpenCV for computer vision functions and parameters. Along with Python for the overall programming language and environment. OpenCV is an open source computer vision library for applications of various functions for the utilization of computer vision techniques. The main method for gathering object and facial recognition data is the Haar Feature-Based Cascade Filters.

In the example used in the OpenCV tutorial, the Haar Cascading algorithm is implemented using an already pre-defined set of positive and negative images. OpenCV comes with a trainer as well as a classifier. The face detector will use the already supplied classifier for facial recognition. This will mean that we will have to include a license since the initial facial classifier is owned by the Intel Corporation. The lamp will operate under a BSD license scheme.

Haar Cascade Filters is an object detection method that uses machine learning algorithms to detect various objects. The algorithm is trained to look at a vast amount of positive and negative images. and to learn what an object looks like in order to detect the object. In the case of facial detection. The Cascade requires positive images (pictures

with faces) and negative images (pictures without faces) in order to "train" the algorithm to detect faces. There are Haar Cascade libraries already included with OpenCV and for the detection system we will identify two separate Haar Filters. One for a profile face view, and the second for the side face view. With both of these filters in place the system should be able to track or locate any person within the frame.

Using the cascade filters, the system will be able to return various values for use in other modules. the three most important aspects for data would be position, velocity and acceleration (velocity and acceleration explained in further detail in Section: 9.2.12). With the position, the haar cascade algorithm offers a box where it sees the object in view. Upon the calculation of the box the algorithm retrieves the middle position of that box and stores it for other modules to collect.

9.2.7 Functional View of Camera Module

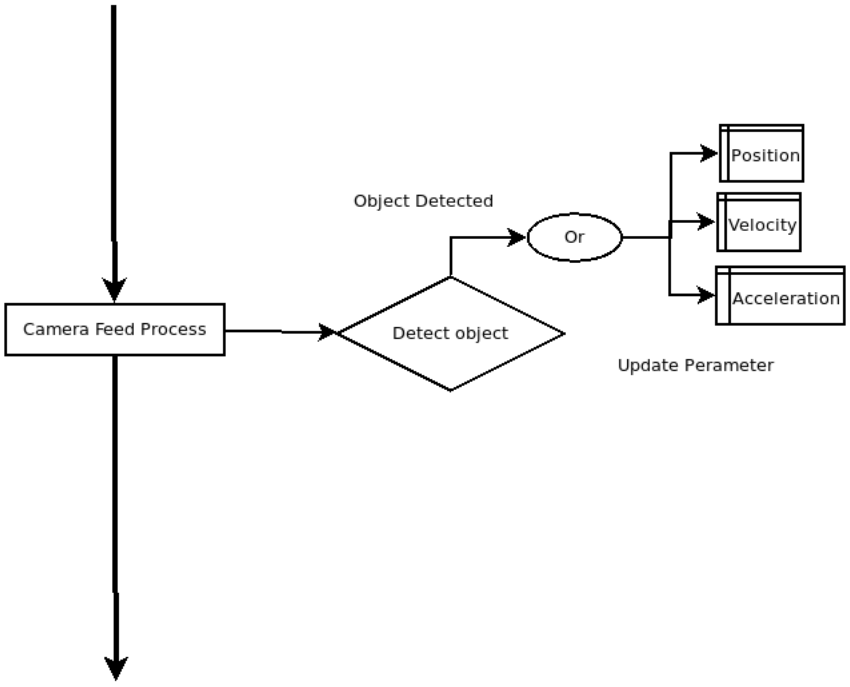

Figure 30: Camera Function Description Diagram

The Camera values are updated in real time, and always available during the on time of the system. This will allow for the camera to

always output relevant information to the other modules. [18]

9.2.8 Proof of Concept Facial Detection

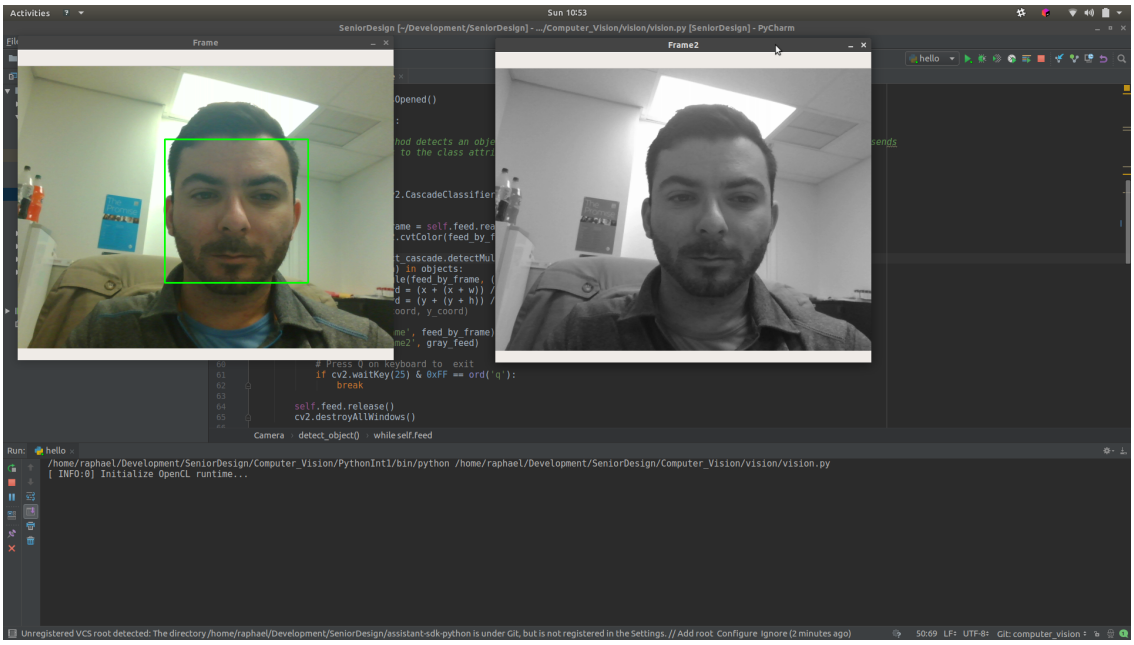

Figure 31: Glasses off Test

From the figure above, we have a test for the facial recognition with no apparent objects on the face. From this test alone it was apparent that there are a few issues. The Facial Detection has its own set of problems. On first testing it has difficulty recognizing a full range of faces from different ethnicities. While my face is easily recognizable, on other faces that I have tested, both male and female, it has had a hard time in recognition.

This is fixable using using a thresholding value built into the Haar Cascade Filter method. Another thing to look at is the lighting, since the Haar Cascade filter uses the difference of lighting within the photo being viewed. Sometimes, background light can view false positives within the system. After through testing we can give a good appropriate thresholding value for real world conditions. [19]

9.2.9 Proof of Concept Facial Detection (With Glasses)

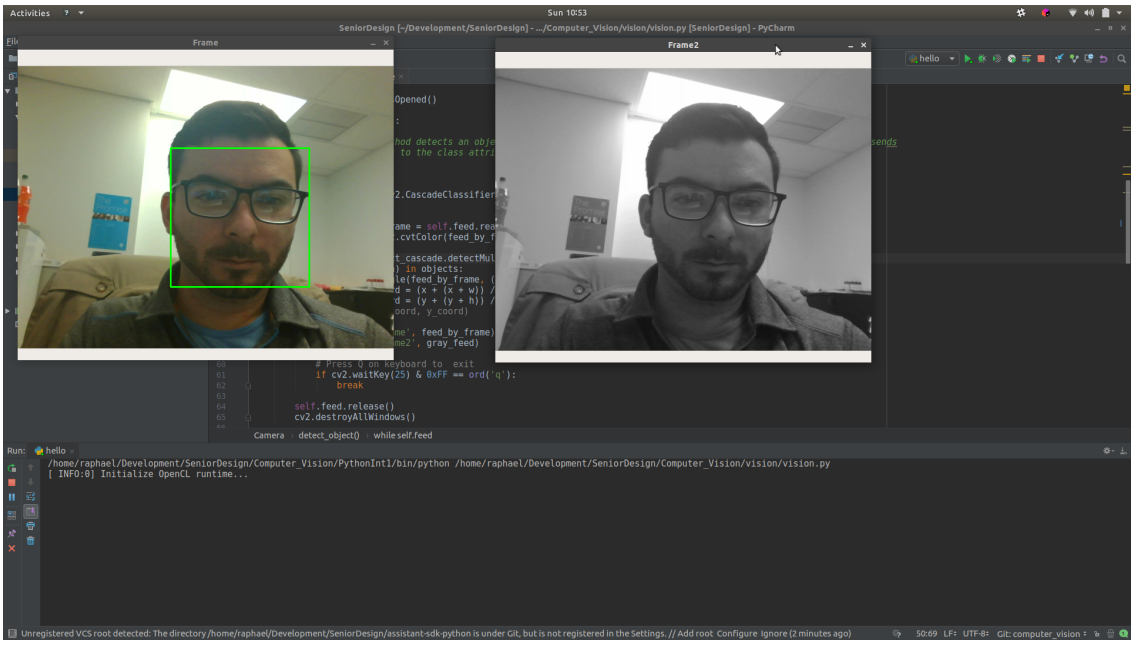

Figure 32: Glasses On Test

The above diagram shows the same process under the same conditions as before, but this time with glasses. Many time certain objects can inhibit the system from finding the face. In this test, the facial recognizer was not fooled but additional testing is required in order to get the correct thresholding for the facial recognizer to operate under normal conditions.

9.2.10 Object Detection

Object detection works with the same method as facial recognition. Therefore it might be useful to understand that both are using the same process flow as each other but with a few differences. Both Facial and object recognition have different problems that require tackling. With facial recognition, the Haar cascade filter, utilized already predefined cascade filters to achieve its detection algorithm. However, with the object recognizer, we would need a cascade filter for each individual object. [14]

9.2.11 Haar Faces

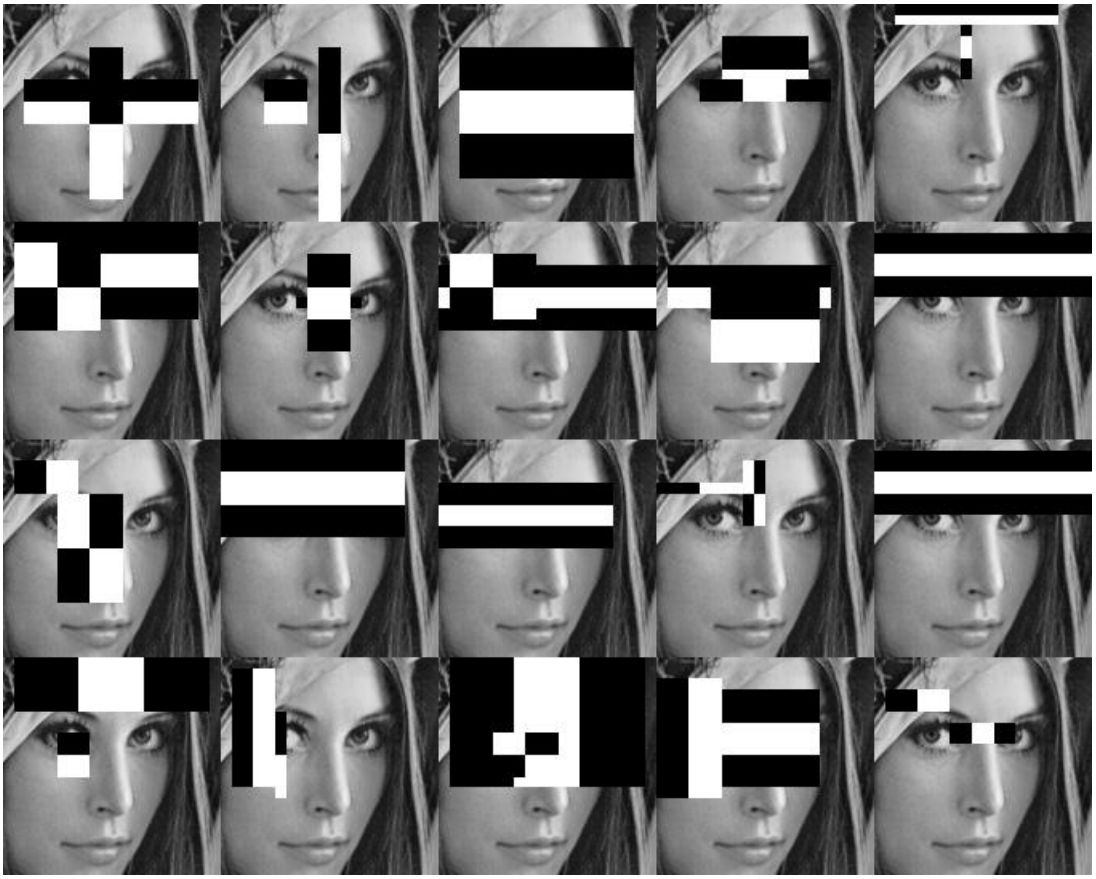

Figure 33: Haar Faces

The example above shows a figure of Haar Features in action on a face used for testing. The difference in facial detection and object detection is minimal but not inconsequential. For the creation of the Haar Cascade the system will use two tutorials, each for different objects [18].

- https://memememememememe.me/post/training-haar-cascades/
- http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html

One tutorial focuses on phones in the view of the camera, the other is for bananas. Both tutorials will be useful for the interesting features

that both objects provide. In order to create a capable Haar Cascade the filter function will have to be custom built [20].

For the lamp it can recognize two objects with addition to faces and people: pens and papers. The reason for this was the original purpose of the Luxo Lamp. The purpose of the lamp and its inception was to light a desk. Since it found popularity with designers, animators and architects, it would be fitting to detect objects that were relevant to them.

For each additional object that requires detection, the system would require a new Haar cascade to train and detect that object. Therefore, there is nothing inhibiting the lamp to support additional detection features. For its inception it would be wise to keep to the objects currently held limited to two [13].

There are particular difficulties that involve paper as an object. in a lot of cases. Paper is easy to hide or blend with the background. Unless there are stark contrasting views, the paper will blend into the background of the table. It would be wise to suggest that for optimal detection conditions, that the object is to be placed in a background that is not white. An ideal example would be a white paper with a dark table.

9.2.12 Object Tracking

Not only does the system need to detect various objects, but it also needs to track the object and save that data for the other modules. This requires an object tracking algorithm in order to complete this task. The object tracking algorithm will be able to detect an object, and track its movement within the frame of the camera view. It should also be able to provide position, velocity, and acceleration information. An additional benefit to having an object tracking algorithm is that if the detection is lost, the tracked object can take the place of the detection algorithm and predict where the object is going to be in order to have a more concrete view of the object at all times [19].

Specifically, optical flow techniques will be implemented. A running time-window of approximately fifteen (15) frames will be kept to determine velocity and acceleration vectors from frames. Optical flow is the process of determining vector velocities from frame to frame movement. More specifically, it would be able to extract the velocity, acceleration for each object that would be moving within the frame. A pyramid processing scheme will be implemented so that information necessary to process is limited and the processing time is therefore expedited. This will allow the system to track more effectively and show minimal latency with respect to the movement of the servomotors.

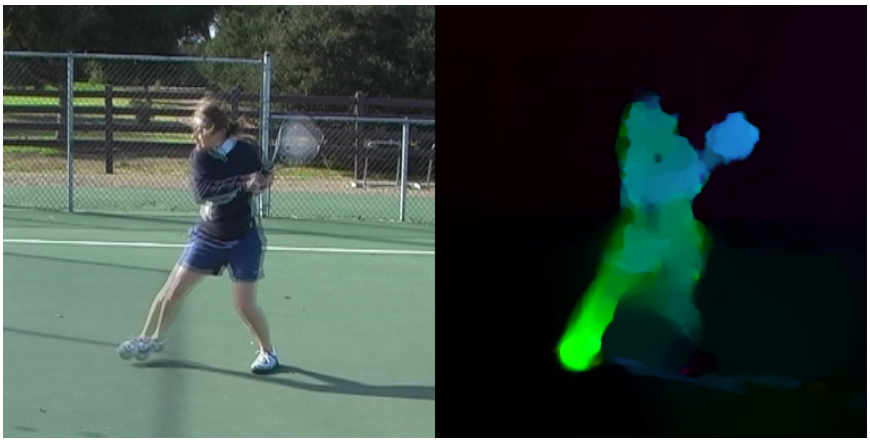

Figure 34: An example of optical flow techniques.[21]

The purpose of utilizing optical flow to calculate velocity and acceleration vectors is two-fold: to allow lamp movement if the object moves out of frame too quickly and also so that the lamp has smooth ramp-up and ramp-down times with respect to movement.

The object tracker is use is also used and held by OpenCV, The KCF Tracker. The KCF tracker algorithm utilizes properties of circulant matrix to enhance the processing speed. Based on the older BOOSTING tracker, the KCF has some advantages that the BOOSTING tracker does not. Since the BOOSTING tracker is based of the Adaboost algorithm, which the same algorithm that the Haar Cascade Filter is based off of, it should seamlessly work with the detection technology that is implemented. The KCF tracker is more of an optimization of the BOOSTING tracking algorithm [22].

From the tracking tutorial:

> KCF Tracker KFC stands for Kernelized Correlation Filters. This tracker builds on the ideas presented in the previous two trackers. This tracker utilizes that fact that the multiple positive samples used in the MIL tracker have large overlapping regions. This overlapping data leads to some nice mathematical properties that is exploited by this tracker to make tracking faster and more accurate at the same time.
>
> Pros: Accuracy and speed are both better than MIL and it reports tracking failure better than BOOSTING and MIL. If you are using OpenCV 3.1 and above, I recommend using this for most applications.
>
> Cons : Does not recover from full occlusion. Not implemented in OpenCV 3.0.

Since this is a built-in function in OpenCV, there are little operations to manipulate within the system in order to get the tracker to work effectively. This will allow the most time allocated towards testing of the device once the modules have been implemented.

9.2.13 Object Tracking Example

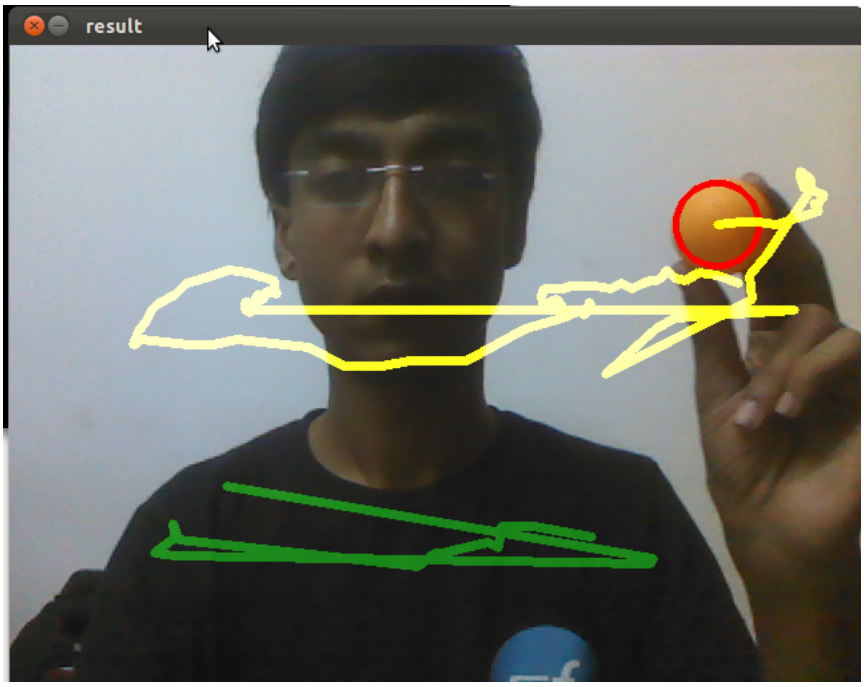

Figure 35: Object Tracking Example using multiple colored balls

From the figure above the feature allows us to track multiple objects within the space of the video shot and additionally, allows us to create a trail for them to be used within the camera frame space. Once this is implemented the relevant information can be sent to the machine learning module and aid the algorithm to create a smooth movement for the lamp to follow and track movements of certain objects within frame.

The tracking algorithm should by final calculations give useful values not only for rediscovering lost detections, but also allow more utilizations for the Lamp to move and flow seamlessly throughout its runtime.

9.2.14 Threading

Threading is a fundamental concept in Computer Science and is a core concept in optimal runtime for the Lamp to operate properly without stalling. Since the raspberry pi model (Section: 4.0.3) in use has a multi-core processor. Therefore, all threads can operate concurrently with the rest of the system as needed with little lag. Within the system the entire computer vision module is able to run on two to three threads [23].

9.2.15 Thread Basics

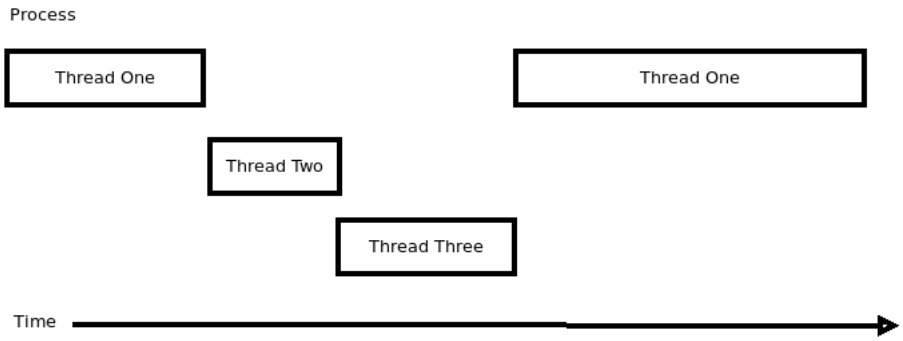

Figure 36: Example Computer Vision Module Thread

In the diagram above the thread is implemented using Time-Slicing. This allocates the threads to share computing power between individual threads. Since the raspberry pi is a multi core system. the threads can use multiple processors in order to do concurrent tasks at the same time. The computer vision module will most likely be the most use of computing power and therefore should have at least one dedicated processor to the thread. This allows for the camera to be always on and ready for a command from the main module.

Since multi-core threading is going to be used there are some obvious drawbacks that come with implementation: [23]

- Synchronization

- Thread Crashing

Synchronization Synchronization becomes a glaring issue. Without the time sharing of a single thread process, the system de-sync and any files in reading or writing within the queue, could potentially be overwritten in another stack. For example, if one process dictates that an object is in a certain location but fails to notify the other thread, then the values sought out by any additional processes could yield incorrect values. This could casue a potential problem to the rest of the system that depends on the computer vision module to display the correct values.

Thread Crashing Thread Crashing occurs when any thread that performs an illegal process. Since multi-threading allows for multiple processes to run concurrently, if there is one thread that crashes this causes the entire process to crash. It is more likely for critical failures to occur with a multi-threaded process then a single-threading process for these reasons.

Fortunately, for the system, python uses a system for scheduling in order to prevent these critical failures to occur. From WikiBooks: [23]

> Threading in python is used to run multiple threads (tasks, function calls) at the same time. Note that this does not mean that they are executed on different CPUs. Python threads will NOT make your program faster if it already uses 100 % CPU time. In that case, you probably want to look into parallel programming. If you are interested in parallel programming with python, please see here.
>
> Python threads are used in cases where the execution of a task involves some waiting. One example would be interaction with a service hosted on another computer, such as a webserver. Threading allows python to execute other code while waiting; this is easily simulated with the sleep function.

In case of any issues within the system, these threads will be managed accordingly.

The first thread will operate all of the camera functions. The start and stop of the camera along with the initial camera feed. It should also handle the external calls to the other aspects of the module. During the camera process, the system shall create and update tracking and detection for the various objects within the frame.

9.2.16 Thread Diagram

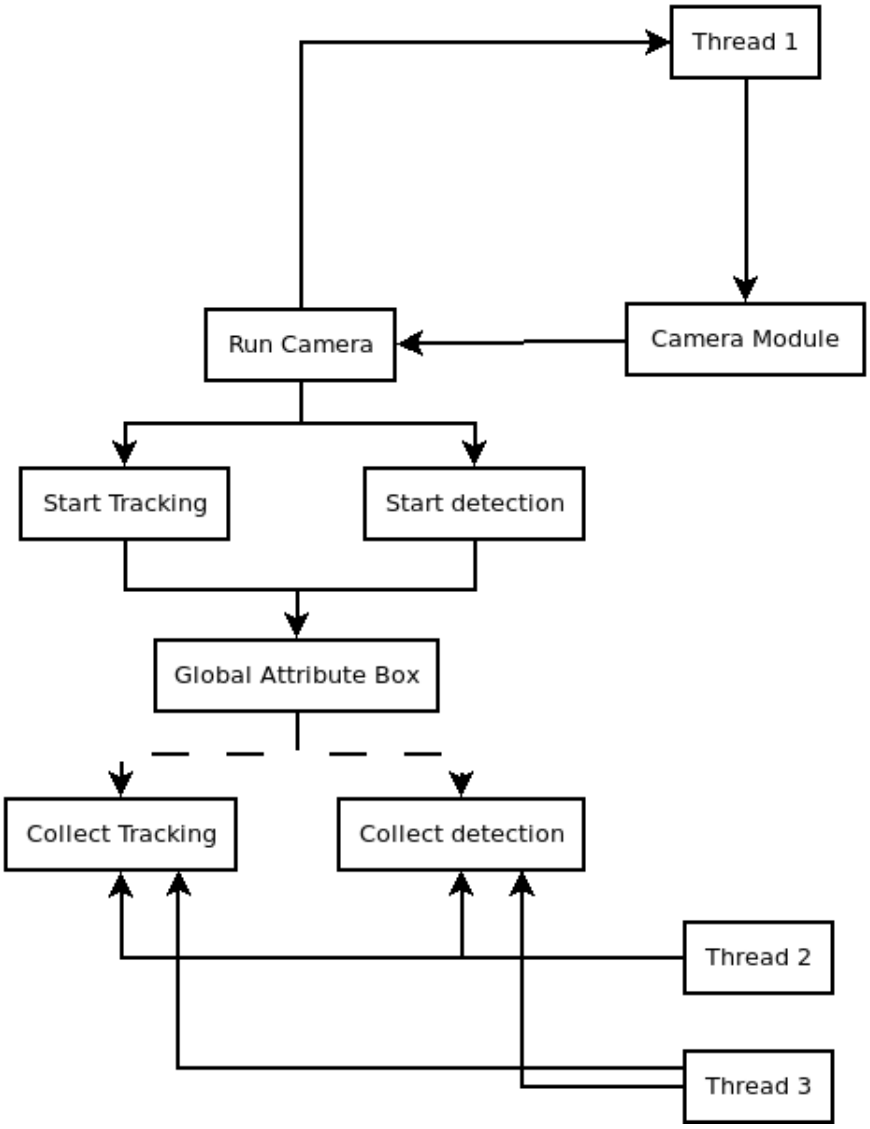

Figure 37: Thread Description Diagram

The second and third thread would be utilized for any other processes that would require the attributes that the tracker and detector create. These processes would be from other modules needing information from the camera or the camera itself to update its status. No more then 3 threads should be utilized at any given time while the camera is operational.

If there is a case where the camera is not operational, the first thread should be halted and waiting for the commencement of its operation. This however should not be the case considering that the camera is vital to the operation of the lamp.

9.2.17 Error Handling

Finally, given the dependence of the project on the computer vision module, it is essential that there are fail safes implemented in order to make sure that the system does not have a catastrophic failure. This will be a series of conditions that will have to be met before accessing or sending data to other modules within the system. Python does have a system for error handling and all errors that the program receives will be handled through those channels.

There are a few points of failure that can be easily described within the document before testing:

- Camera and Connection Failure
- Threading De-synchronization
- Class and Method De-organization
- Facial and Object Misreads
- Null-pointers and other common errors

Camera and Connection Failure

The first and biggest error is if the camera fails to operate in the first place. This error could happen due to a variety of different issues and should let the user knows that this error has happened. If this is the case the system shall operate with a response message asking the user to check all the connections within the system. Once the System has performed the error catch request the system should then shut off in order to conserve power [24].

Threading De-Synchronization

Threading de-synchronization occurs when the program that allows for multiple concurrent threads to occur at the same time. This can cause a synchronization issue when files are being read or written to. If a de-sync occurs then the result is another aspect of the system getting the wrong values when operating. Unlike the camera failure system failure, this error would be inherent in the coding of the system and therefore will not have a user notification. This error should not only be caught before the system goes into general production but also during the testing and debugging phase of the development cycle. The avoidance of this would be to apply proper threading techniques in order to verify that all threads are complete with the current process before reading or writing to the files within the system [24].

Class and Method De-organization

Class and method de-organization occurs when the system has a few methods that are not reusable within said system. It can also occur when one method does more than one task and therefore another process does not have the result to that information unless it activates that whole method process. This is also an error in coding structure and therefore must be checked before the testing stage is implemented. This allows for other aspects of the system to get the data provided by the computer vision module, and stop using the computer vision process [24].

Facial and Object Misreads

Facial and object recognition misreads occur when there is a false-positive for the facial recognition suite when looking for faces. This occurs depending on a variety of different environments when the picture is being taken, or when the video is being analyzed. This is the one aspect of the project that will be taken care of during testing. Note, however, that although when can come very close to 100% detectability we can never each a complete facial recognition. There needs to be a metric for allowing an error ratio for this type of error. This error ratio, in this case is 80 % effectivness in normal situations and conditions. During the testing and debugging stages the system will be optimized to meet the needs for the system.

Other Common Errors

Other common errors within the code can occur. This is why it is vital for the system to go through a unit testing process. This will be done using UnitTest, that way the system will be able to comply with standards as to not let the other aspects of the system fail. This should conclude any foreseeable errors within the program, if there is any unnoticed errors they are to be corrected during the testing and debugging stages of the software development cycle.

9.3 Machine Learning

9.3.1 Introduction to Machine Learning

There are many definitions and explanations of machine learning on the Internet. A search for a simple definition of machine learning can quickly become overwhelming or even confusing for newcomers to the subject. Here are a few definitions that can be found with a simple Google search:

Wikipedia: Machine Learning

Definition 2: "Machine learning is a field of computer science that gives computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed [25]."

Nvidia: Machine Learning

Definition 3: "Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world [26]."

Expert System: Machine Learning

Definition 4: "Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed [27]."

Google Cloud: Machine Learning

Definition 5: "Machine learning is functionality that helps software perform a task without explicit programming or rules [28]."

So what exactly is machine learning? An aspect that virtually all definitions agree on is that machine learning somehow gives computer systems the ability to learn to execute a task without being explicitly programmed to do that specific task. It is also agreed that the computer system that is learning needs some data to make predictions from. At a basic level:

Machine Learning

Definition 6: Machine learning is a field of computer science that is focused on giving computer systems the ability make decisions and predictions on complex data that would otherwise be difficult to explicitly program.

9.3.2 What is Artificial Intelligence?

Now that we have established a definition for machine learning, it is worth pondering what *artificial intelligence* is all about, and how it relates to machine learning, as these two terms (machine learning and artificial intelligence) are often used together. It is therefore imperative to establish a clear definition of artificial intelligence and define a clear distinction between these two terms.

Techopedia: Artificial Intelligence

Definition 7: Artificial intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans [29].

Stopad: Artificial Intelligence

Definition 8: Artificial intelligence (AI) is the ability of a machine or a computer program to think and learn. The concept of AI is based on the idea of building machines capable of thinking, acting, and learning like humans [30].

Taking these two definitions together, artificial intelligence can be wholly summed as follows:

Artificial Intelligence

Definition 9: Artificial intelligence is simply a field of computer science that delves into the topic of a computer's ability to understand and approach a problem with human-like intelligence.

Clearly, a definition is not enough to encapsulate what artificial intelligence truly is, and so therefore, there is much more to discuss about artificial intelligence on a conceptual, theoretical, and practical level.

9.3.3 Where is Artificial Intelligence used, and how far has it come?

Artificial intelligence exists to tackle areas such as speech recognition, data analysis, and problem solving. It is noteworthy to make note of today's current progress with artificial intelligence research. Arguably

AI's most popular accomplishment could be contributed to Google's AI computer program AlphaGo which was able to establish itself as the top champion of a board game called Go [31]. Beginning in March of 2016, AlphaGo defeated master Go player Lee Se-dol 3-0 in a best-of-five competition [32]. Then in May of 2017, AlphaGo defeated the human champion of Go, Ke Jie [31].

Currently, a significant amount of artificial intelligence research has been invested into self-driving trucks [33]. There is also ongoing research at the Massachusetts Institute of Technology's Computer Science and Artificial Intelligence Laboratory (CSAIL) for developing a new sensor technology, called Gelsight, that uses physical contact with an object to provide a detailed 3-D map of its surface [34]. This technology is aimed for applicability in precision surgery.

9.3.4 What is the difference between Artificial Intelligence and Machine Learning?

The difference is quite simple: Artificial intelligence is a broader field while machine learning is a sub-topic of artificial intelligence [30]. There are also more subcategories of artificial intelligence rather than machine learning. For example, natural language processing, inference algorithms, and neural networks are all technologies under the artificial intelligence tree [30, 35].

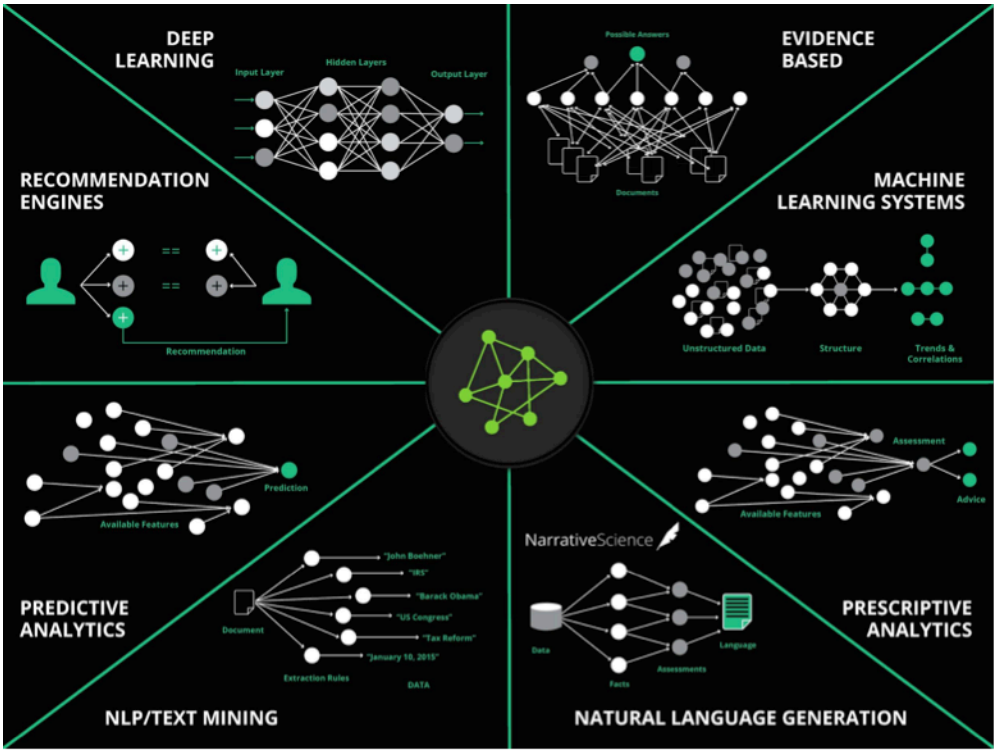

Figure 38: Major Subcategories of AI. Credit: Narrative Science

For the purposes of our project, our focus is on machine learning. Therefore, it is worth viewing this diagram in order to obtain a more clearer understanding of machine learning versus artificial intelligence.

Figure 39: Artificial Intelligence vs. Machine Learning

9.3.5 How is Machine Learning Relevant to Us

Our lamp has five axes, all at unique joints, that can be manipulated to achieve a position that will point its camera to a desired relative location. The number of distinct positions are vast and explicitly programming the manipulations of the lamp's axes would be extremely difficult. Because of this we will instead use machine learning to "teach" our lamp to move on its own without us explicitly programming in those movements. The lamp will be given data about its current state and environment, from which it will determine the positions that each of its axes should be at.

There are many methods that we could use. The most common learning methods are supervised, unsupervised, and reinforcement learning [36].

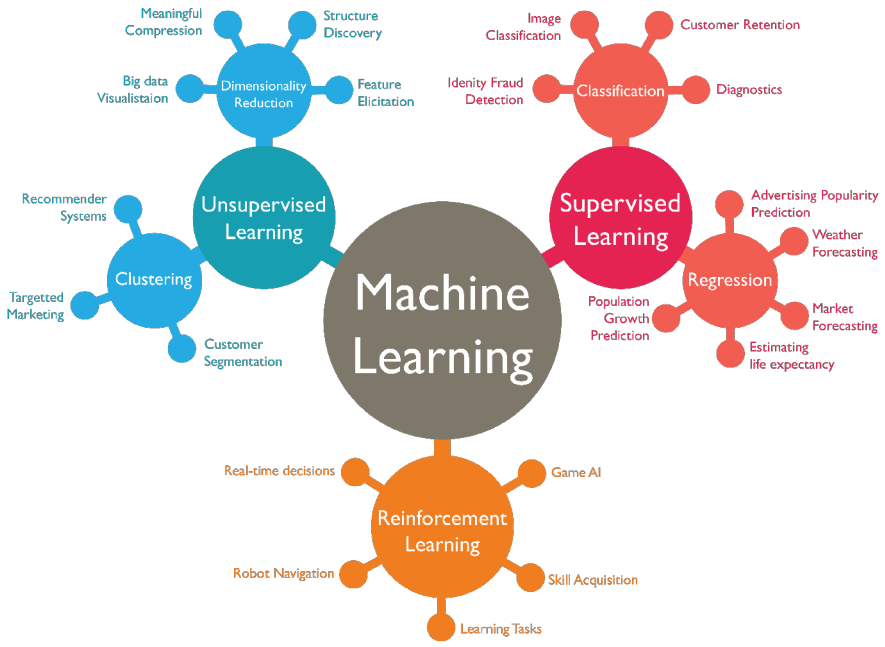

Figure 40: Machine Learning Diagram

9.3.6 Supervised Learning

Supervised learning is a method where a computer system's "learning" is guided by a human. The learning algorithm is provided a training data set that consists of example inputs and their correlated outputs.

An example of this would be when training image recognition. The learning algorithm would be given a training set of images paired with their correct labels [37]. If the algorithm was being trained to label an image as a dog or cat, its training set would contain many different images of dogs and cats (each labeled as such). The algorithm would then create a relation between the images and their labels, building a data structure (model) to be used to classify an image as a dog or cat. The model would then be tested using a validating data set. When an image is fed into the data structure, either a dog label or a cat label would be outputted (or "neither" if trained to do so). The model's output would be validated using the validating data set. If the model was incorrect with its prediction, the model would then be "tuned" [38].

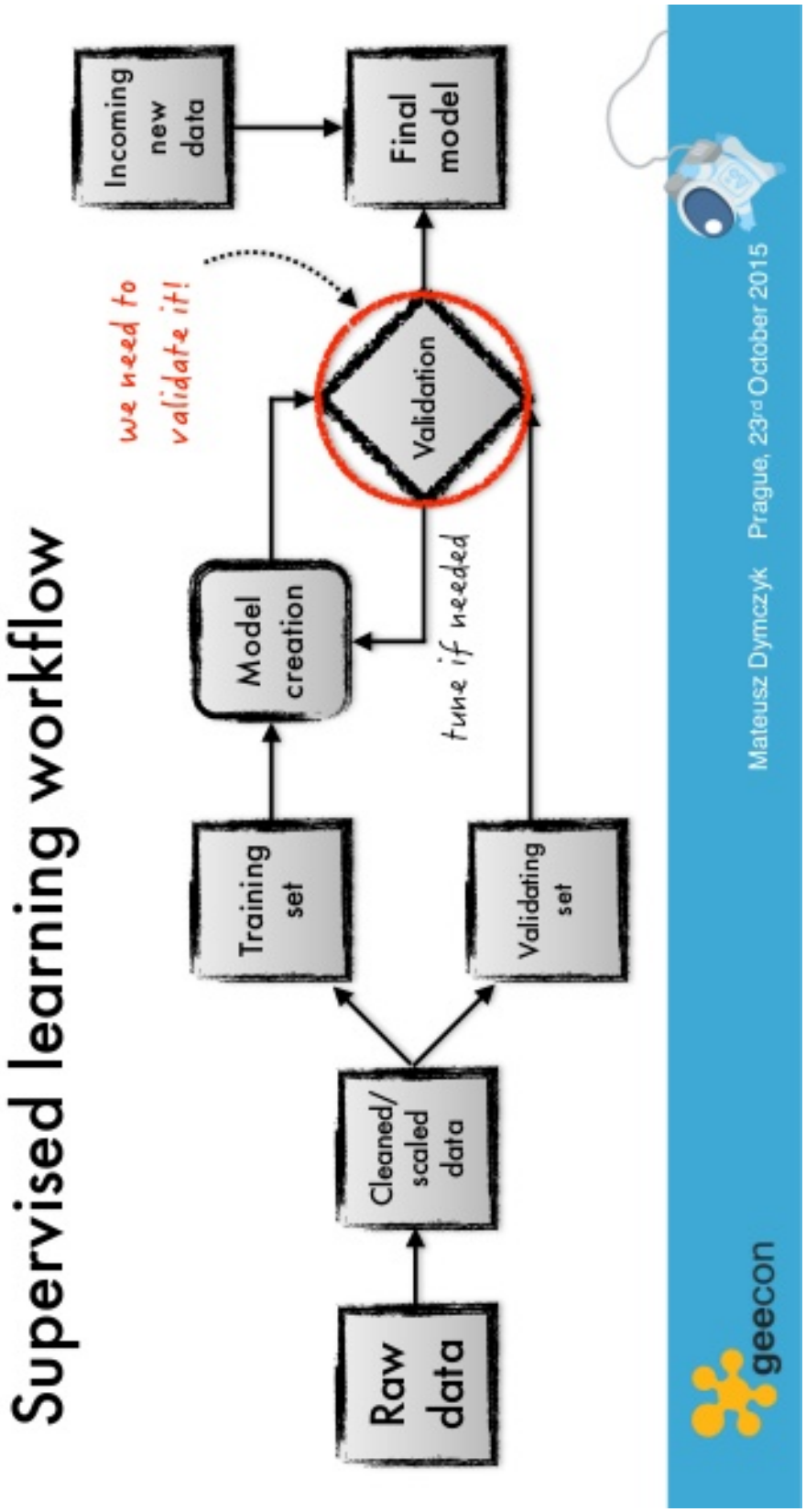

Figure 41: Supervised Learning Workflow Diagram

9.3.7 Unsupervised Learning

Unsupervised learning is very similar to supervised learning. The main difference between the two is that unsupervised learning is not guided by a human. Instead, during the training phase, the unsupervised learning algorithm is only given input and not a correlated output. The goal of the learning algorithm is to look at similarities of all its inputs and create its own groupings (outputs).

As an example for when unsupervised learning may be used would be for custom segmentation [37]. A company could train a model that would group its customers so that the company could more easily employ things like targeted marketing [39, 40].

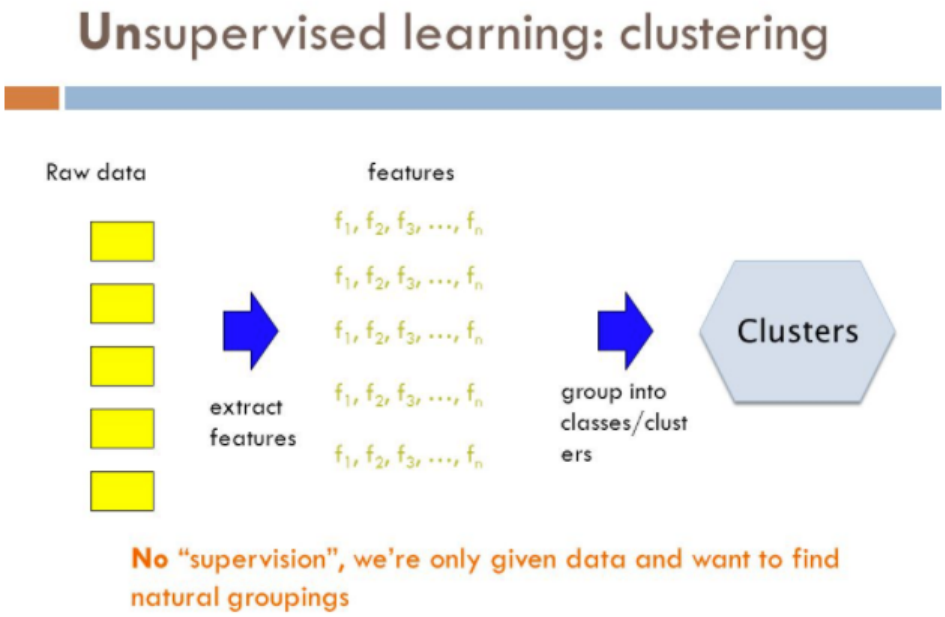

Figure 42: Unsupervised Learning

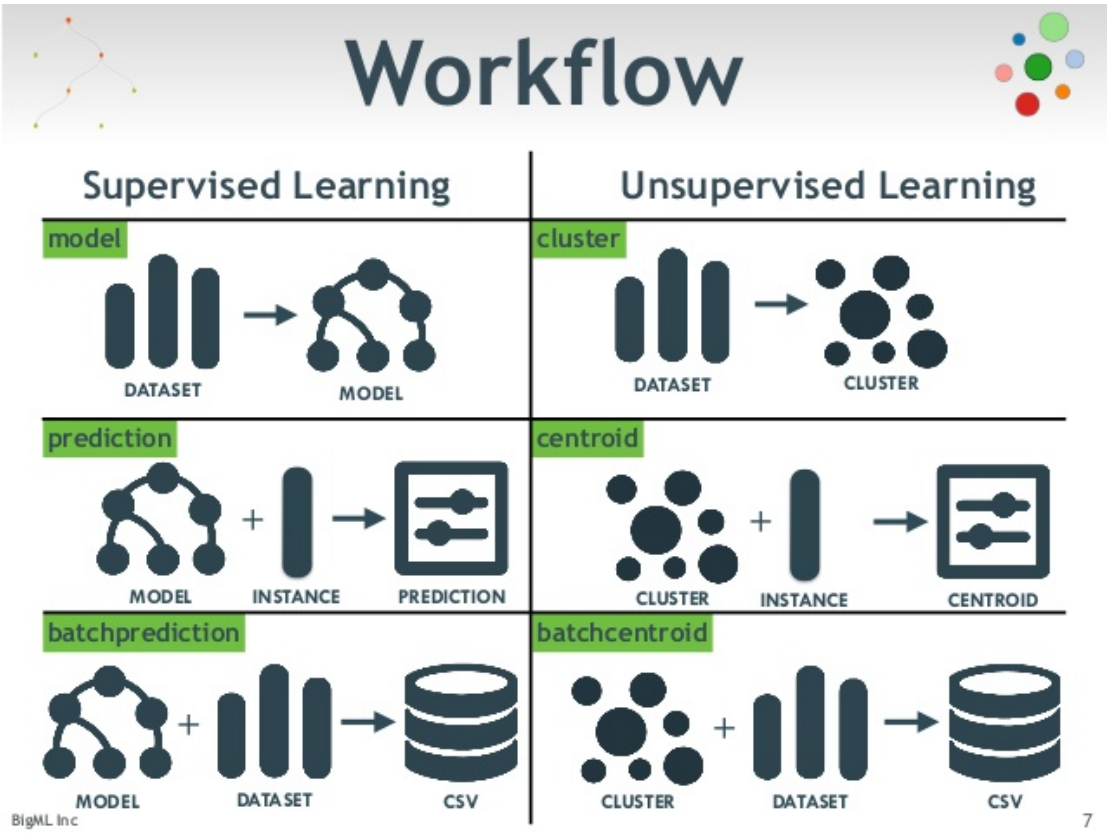

Figure 43: Supervised Learning vs Unsupervised Learning

9.3.8 Reinforcement Learning

Unlike supervised and unsupervised learning, reinforcement learning does not learn from some previously collected data that is fed into the learning algorithm. In reinforcement learning, the agent (thing trying to learn) is trained by trial and error. The agent takes an action based on its observations of its environment. It will then receive a reward based on the action it took [41]. The value of the reward will influence how likely the agent is to take the action again. This mimics how humans and animals learn in the real world.

Imagine a dog being potty trained by its owner. If the dog goes in the house, the dog will be scolded and put outside for some time. If the dog instead holds it until it is let out, the dog will be given a treat and told how good of a dog it is. Overtime, the dog makes a connection between going inside and getting scolded at, and going outside and

getting a treat. Since the dog would much prefer a treat over being scolded, it learns to go outside and is less likely to go inside [41].

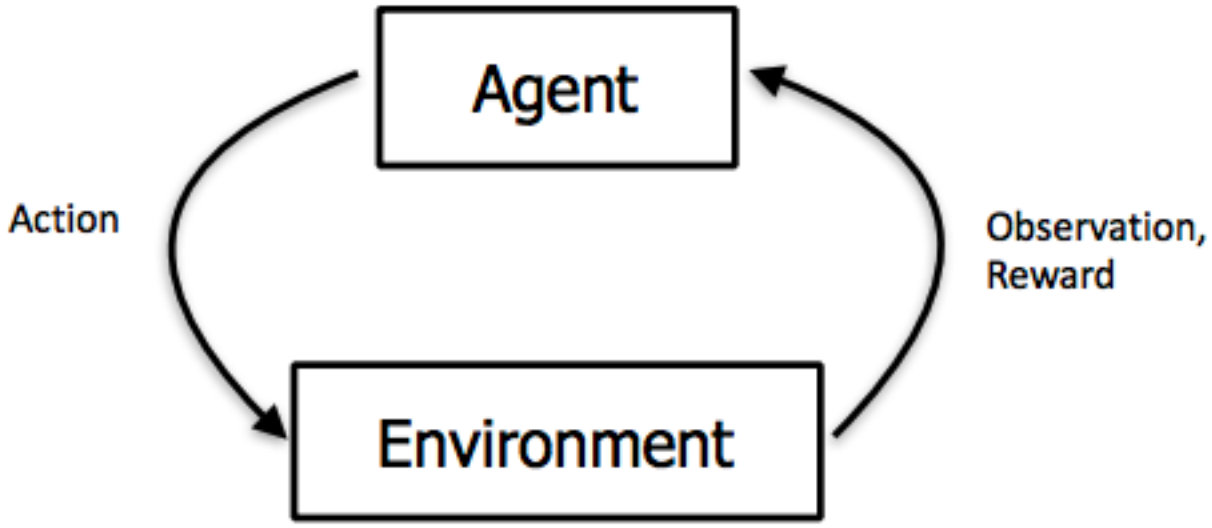

Figure 44: Agent and Environment Interaction

9.3.9 Choosing Reinforcement Learning

When researching the three methods mentioned above, we decided to use reinforcement learning to train our lamp. There were several factors that lead us to make this decision. We needed to decide what data was necessary for the lamp to learn movements. The lamp will be required to not only rotate to stationary objects / points of interest, it will be required to track objects that might be in motion. Either way, we have decided that the only information that the lamp absolutely needs to learn movement is a simple x and y offset that the point of interest is from the camera's center view and the current angular position of all five of the lamps axes. So at a minimum, the learning algorithm would need an input vector of length 7.

Input Vector

Formula 1:

$$input = (x, y, \theta_0, \theta_1, \theta_2, \theta_3, \theta_4)$$

After deciding the minimum input, we considered what information each of the learning methods needed and what the output would be. We observed that there would be many different "valid" movements that the lamp could use to look at a particular point in space. It would be preferred that there was only one movement, but since that is not an option, we needed some way to determine if the movement was not only valid but desired (more detail later). Given that we wanted to somehow "influence" the learning algorithm to output desirable movements, we never gave unsupervised learning much thought.

Initially we considered using a supervised learning method. We thought that we could give it some example inputs and outputs of desired movements and the algorithm could learn the "between" movements to "fill in the gaps". This sounded promising at first, but upon closer inspection, we realized it really wasn't feasible because we would need a lot of training data which would take a significant amount of time to collect. At this point we realized that a supervised learning method wasn't going to be sufficient.

With "learning by example" out of the question we thought we could somehow rate the lamp on how desirable the movement was. Immediately we started to consider a reinforcement learning method. But we weren't really thrilled about the idea of the lamp making thousands, if not hundreds of thousands, of random movements before it finally made a valid and desirable movement because of how much time that would consume. We really did not want to spend days training the lamp just to realize that we did something wrong and need to debug it and attempt to train it again. Therefore, the solution that we came up with was to train the lamp in a simulated environment. In the next section, we will explore the benefits and negatives of using a simulated environment to train our lamp.

9.3.10 Simulated Training - Pros and Cons

Pros:

- Fast, flexible training
- Easy to observe and debug
- Only one physical lamp but can create and train many simulated lamps
- Compare the results of different training procedures side by side
- Training on a GPU or Amazon Web Services Cloud

Cons:

- Possible issues when moving trained model from simulated to real lamp

9.3.11 Simulated Training Using Unity3D

When searching for a simulation software to build a lamp model, we decided to go with Unity3D. A couple of factor influenced this decision. First, we have a member in our design team that has experience using Unity3D so there would be no valuable time consumed learning a new tool. Second, Unity3D makes it very easy to build a sufficient model of our lamp, making it less time consuming. Lastly, Unity recently released a beta version of Unity Machine Learning Agents which employs a reinforcement learning algorithm and components to interface with that algorithm. The discovery of Unity Machine Learning Agents was completely unanticipated but strengthened our choice for using Unity3D for our simulated training environment.

Machine Learning Agents (ML-Agents) is an SDK that allows for the training of agents using machine learning methods through a simple

to use Python API. There are three main types of objects in a ML-Agents learning environment: *agent*, *brain*, and *academy* [42]. [1]

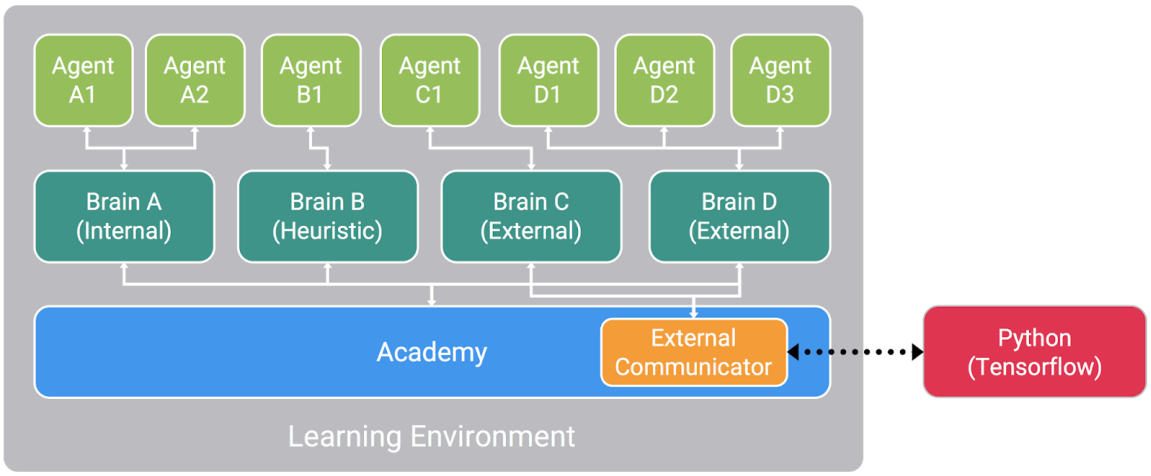

Figure 45: A visual depiction of how a Learning Environment might be configured within ML-Agents.

An *agent* is an autonomous entity that interacts with an environment. An agent takes actions which are influenced by its unique state and observations of its environment. An agent will receive rewards for events within its environment, ultimately rewarding the agent for the actions that led to the event. Actions that an agent takes are decided by the brain that the agent is linked to [43].

A *brain* controls the decision making process for any agents that are linked to it and must be a child of an academy. Every agent must be linked to a brain and multiple agents can be linked to the same brain while still having its own unique set of actions and observations [44].

An *academy* contains all the brains in an environment as children. There is only a single academy in an environment. An academy defines several parameters that describe its environment [42].

9.3.12 Training Scenarios

The Unity Machine Learning Agents SDK allows for several different training scenarios. Two are of particular interest to us: *single-agent* and *simultaneous single-agent*.

Single-agent is the simply training a single agent that is linked to a single brain. This is the traditional method of training an agent and is pretty straightforward to setup [42, 45].

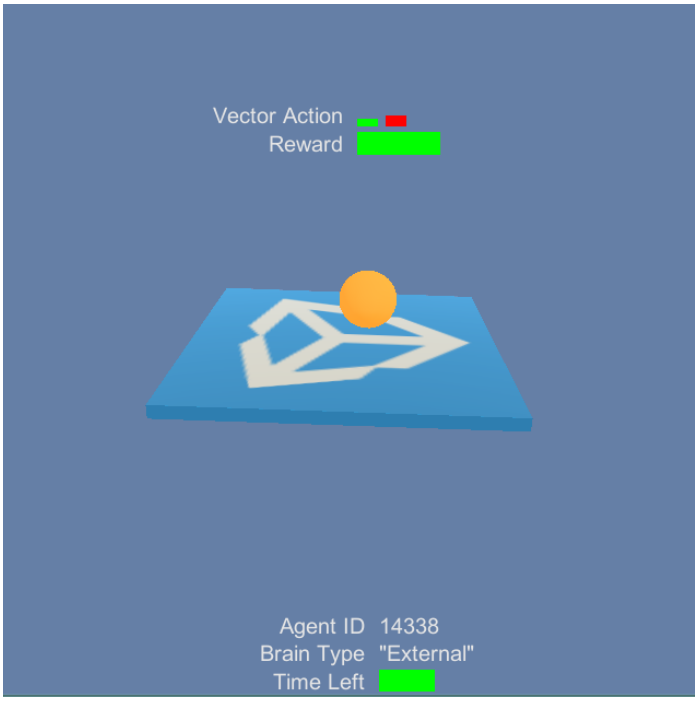

Figure 46: Single agent

Simultaneous single-agent is when several independent agents with independent reward functions are all linked to the same brain and trained in parallel. This scenario and dramatically speed up, and even stabilize, the training process as all agents are contributing to building the same model [42, 46].

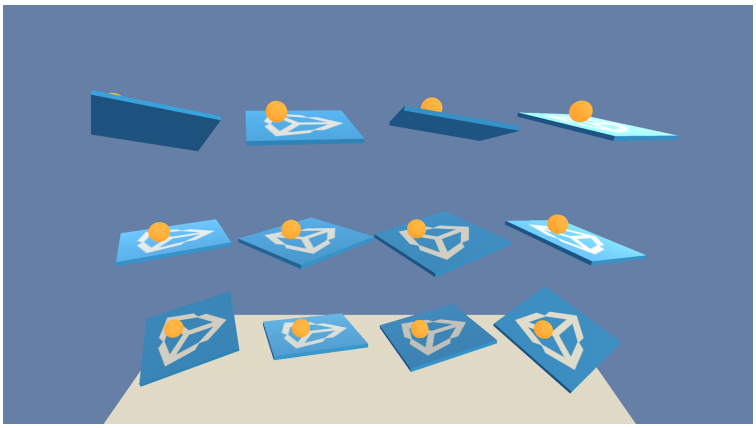

Figure 47: Multiple agents linked to one brain

9.3.13 Possible Training Procedures

There are numerous procedures that we could use to train our lamp. We will implement several different procedures when training our lamp so that we can compare the results of each and choose the "best" model. We categorize movement into three distinct groups: valid, optimal, and desired.

First and foremost, the lamp's movements have to be valid. A *valid movement* is any movement that results in the camera looking at the point of interest in space. For any one point in space, there will be many different movements that the lamp could execute that would be considered valid.

Beyond valid movements, are optimal movements. An *optimal movement* is a valid movement that achieves its goal in the least amount of movement. An optimal movement could be divided into two distinct measurements: how many joints were manipulated and how much were those joints moved. When deciding whether a movement was optimal or not, we could use one or both of those measurements.

Lastly, we there is a category of movement that doesn't necessarily have a distinct way to be measured. We call these movements, desired movements. A *desired movement* is any valid movement that we dub desirable. Maybe we desire a particular movement because it is optimal

or perhaps because its "human like". The reason behind a movement being desirable isn't really important but we must come up with some way to quantify those movements if we want to influence training in a manner that will produce the desired movements reliably.

Below we go into some details on possible procedures that might result in a particular category of movement. For all possible procedures, the output will be the same; a vector of length five representing the positions of all five of the lamp's axes.

Output Vector

Formula 2:

$$output = (\theta_0, \theta_1, \theta_2, \theta_3, \theta_4)$$

- Valid Movement
  - Input
  - Reward Function
- Optimal Movement
  - Input
  - Reward Function
- Desired Movement
  - Input
  - Reward Function

9.3.14 Python Tensorflow API

The intention is to use Python's Tensorflow Application Programming Interface (API) to code the actual machine learning algorithm in order to train the lamp and its Unity3D models. Our choice in using the

Python Tensorflow API mostly stems from the extensive documentation and tutorials. These resources will greatly help us develop the machine learning algorithm in a timely manner and fashion. Another reason for our particular choice in selecting Tensorflow as our machine learning library is driven by the fact that our entire project will be primarily implemented in Python. For example, the computer vision portion will be implemented in Python's OpenCV API. Therefore, choosing a machine learning library that can be written in Python code will tremendously help us integrate all of these different aspects of the project [47].

Figure 48: Python Tensorflow API

9.3.15 Expected Challenges

Like with any other project involving machine learning, there will be many initial hurdles to jump over. Below is a list of obstacles that are expected during the development of the machine learning and simulation training portion. Some remarks, including on how to possibly overcome each obstacle is also presented in following list.

- The lamp will need training data in order to train. Since there is no pre-existing data for the lamp to train on, the team is expected to generate their own dataset through the training of virtual lamps in Unity.
- The lamp needs an efficient way to train effectively and produce desired results. With the aid of simulated training, each model

can be given a different training algorithm for the developer to observe the results of each algorithm. Through careful observation and note keeping, the development team will be able to note the influences of each training factor, which will allow the machine learning algorithm code to be tweaked to yield optimal results for the lamp.

- There must be an indicative goal for the lamp to try to reach in order to learn. In other words, the lamp must have an objective to strive for, and this measurement should ideally be a quantity that is easily measurable in order to provide substantial feedback to the lamp.
  - While there is perhaps some indicator to measure smoothness of a movement, there is no real indicator to gauge whether a movement appears natural and appealing.
  - As a result, the team has determined that the best method to train the lamp is through the reinforcement learning (RL) approach. As discussed in 9.3.9, this approach will allow the developers to rate the lamp's movement on a numerical scale which will provide incentive for the lamp to do better on its next iteration.
  - The team recognizes that the reinforcement learning approach is not perfect, and the team is prepared to anticipate unpredictable movements from the lamp even after being thoroughly trained, since the lamp is simply taught to move better but not how to move better, or rather, what objectively makes a better movement. But the team plans to mitigate its unpredictability through extensive training for every test case, including edge cases, and if in the worst case scenario, the lamp does choose to move unpredictably in a less-than-ideal way, then depending on the rarity of this behavior, this behavior may help the lamp to be portrayed in a more dynamic, fun, and human-like fashion.

9.4 Google Assistant API

The Google Assistant API module is another access point for interaction with the Automated Lamp. The Google interface serves as an operational starting point along with the camera for a user interface that is completely keyboard-less and mouse-less. Most of the interface is a RESTful web service that is operated and maintained by Google with custom commands created by the Google Assistant API module team. With this in mind it is good to note that this module is the only module within the system that requires an internet connection.

Figure 49: Google Assistant Logo

The internet connection is facilitated by the raspberry pi and either operates on 802.11a/g wifi connection or a hard line RJ-45 Ethernet cable plug-in. Either connection protocol will be sufficient to allow the Google Assistant API to function properly [48].

The Google Assistant requires a few requirements and dependencies in order to operate on the raspberry pi system. The Google Assistant Software Development Kit (Google Assistant SDK) allows for the

prototyping of the device and its implementation on the raspberry pi system. The initial hardware components, as described by Google are as follows:

- Raspberry Pi Model 3b
- USB microphone
- Speaker
- RaspberrianOS or other Linux based Operating System

The Automated Lamp contains a variant of those devices. The microphone is embedded in the raspberry pi system, and the speaker is externally provided. In addition to the microphone the automated lamp has other hardware devices that extend the features of the Google Assistant. These include:

- A circular LED light array
- Movement operations
- Custom Commands (9.4.11)

With these items that the assistant requires along with the additional features added we can now discuss the overall process flow.

9.4.1 Google Module Project Work-flow

Along with any project work-flow, each module in the project will have its own selective work-flow. This is for a few reasons, firstly, each project module will have its own advances and pitfalls and secondly, each project is lead by an individual group member, and therefore, that group member should have the final say in the implementation of their work-flow [49].

The Google Assistant module work flow is as follows:

- Set up Hardware (Local Environment) and configure network access
- Configure and test audio
- Configure a developer project within the Google Cloud Service Platform
- Register the device model
- Run sample code and initialize the device
- Implement the system onto the Raspberry Pi Module.
- Run and configure the device onto the Raspberry PI module.
- Test basic commands for the Google Assistant
- Implement custom commands onto the the Google Assistant Module.
- Test Custom commands on the Raspberry PI Module.
- Test using other modules within project.

Each item on the module work-flow has its own unique challenges and will be explained in more detail below

Set up Hardware and configure network access

In order for the module to operate effectively, the device needs to be configured correctly. In the Google Tutorial documentation, The device in use needs at minimum a microphone, a speaker, a keyboard, mouse and monitor. As for the Raspberry PI implementation, the documentation requires the Raspberry Pi Module, speaker and an OS pre-installer. The Documentation also calls for keyboard and mouse in order to install the OS within the Raspberry PI, but assuming that all of that is complete we can move on to the network access.

The network access is taken care of in two ways; Wi-Fi and/or Ethernet cable. In the case of the lamp, in order to limit the number of cables required for basic functionality, Wi-Fi is the preferred method of use, but if for some reason Wi-Fi cannot be implemented, an Ethernet cable can be used to connect to the network. If neither is available, neither the initial setup nor the working implementation of the lamp using Google Assistant API will work properly.

Configure and Test Audio

Configuring and testing the audio is an important aspect to allowing the Google Assistant to work properly and communicate with the user. If this is not completed then the lamp will not be able to utilize its main feature for interacting with users, therefore eliminating the User Interface's ability to operate effectively.

For the testing environment, the steps to test the audio are relatively simple; we can use a few commands to allow the interface to listen and recognize the text being displayed. Using the commands below in the computers terminal application, we can test the speaker

Play a Test Sound

speaker−test −t wav

Record a Short Audio Clip

arecord −−format=S16_LE −−duration=5 −−rate=16000 −−file−type=raw out.raw

Check the recording by playing the audio clip

aplay −−format=S16_LE −−rate=16000 out.raw

Configure a developer project within the Google Cloud Service Platform

Configure an Actions Console project A Google Cloud Platform project, managed by the Actions Console, gives your device access to

the Google Assistant API. The project tracks quota usage and gives you valuable metrics for the requests made from your device.

To enable access to the Google Assistant API, do the following:

- Open the Actions Console.
- GO TO THE ACTIONS CONSOLE
- Click on Add/import project.

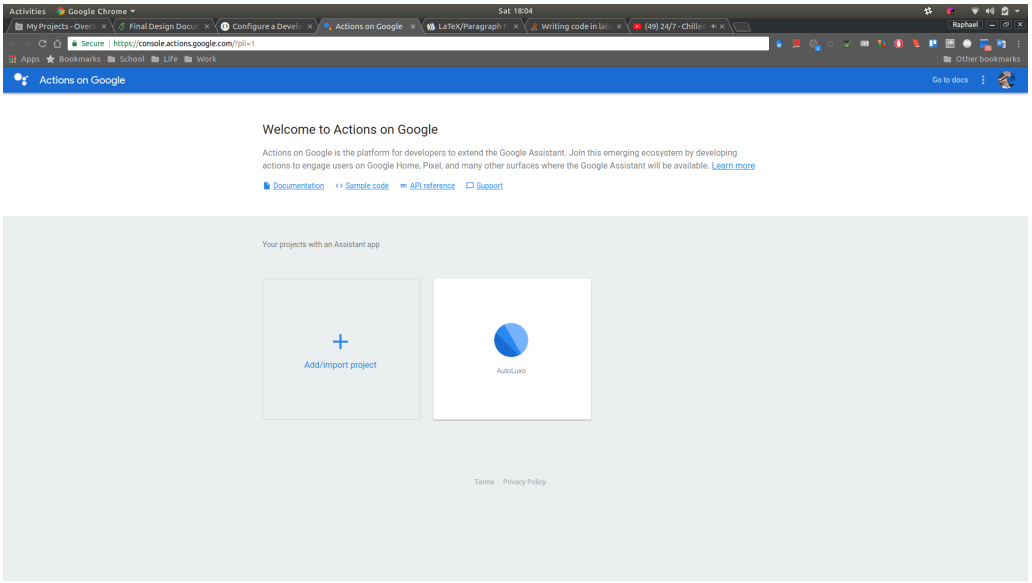

Figure 50: Actions on Google (Actions Console)

To create a new project, type a name in the Project name box and click CREATE PROJECT.

- Add/import project
- Enable the Google Assistant API on the project you selected (see the Terms of Service). You need to do this in the Cloud Platform Console.
- ENABLE THE API

- Click Enable.

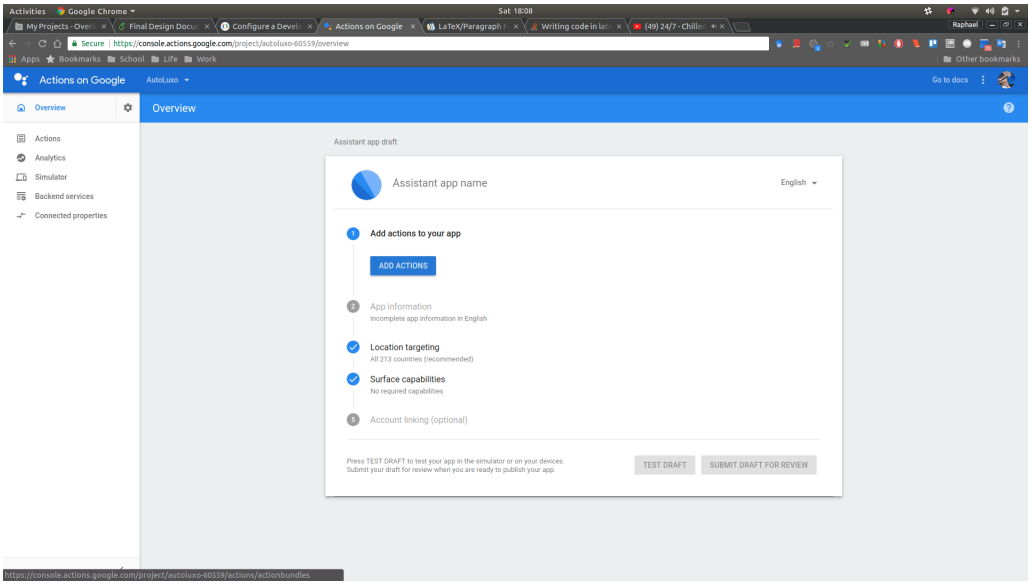

Figure 51: Add Actions (Actions Console)

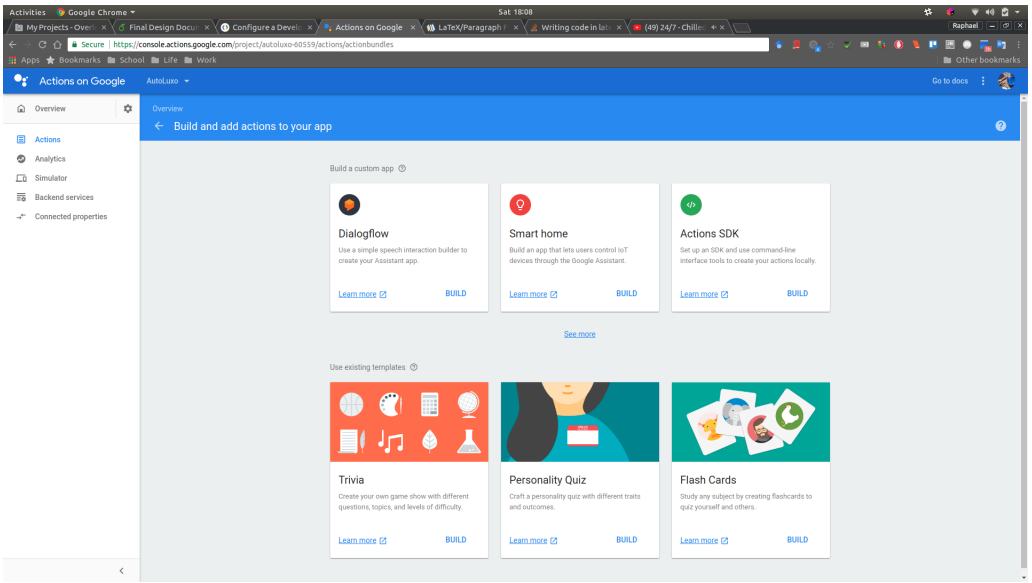

Figure 52: Actions Tab (Actions Console)

Set activity controls for your account In order to use the Google Assistant, you must share certain activity data with Google. The Google Assistant needs this data to function properly; this is not specific to the SDK.

Open the Activity Controls page for the Google account that you want to use with the Assistant. You can use any Google account, it does not need to be your developer account.

Ensure the following toggle switches are enabled (blue):

Web & App Activity In addition, be sure to select the Include Chrome browsing history and activity from websites and apps that use Google services checkbox. Device Information Voice & Audio Activity

Register the device model

Installing the SDK on the device is straight forward. Since both operating environments are using a version of Linux, we can implement the same step for each device. The below is used when using python 3+. This the the language that we have decided to use for the environment. If, for some reason, Python 2.7 is being utilized, please refer to: https://developers.google.com/assistant/sdk/guides/library/python/embed/install-sample, for more details.

In order to set up a Google Assistant SDK, we need to first establish a Python development environment. The steps in order to complete such a task in terminal are provided below:

```
$ sudo apt-get update
$ sudo apt-get install python3-dev python3-venv
$ python3 -m venv env
$ env/bin/python -m pip install --upgrade pip setuptools
    wheel
$ source env/bin/activate
```

Now that we have a development environment established, we need to install the Google SDK. First we need to install the system's dependencies. Note that from this point on we are inside the development environment and not using the traditional terminal bash.

```
(env) $ sudo apt-get install portaudio19-dev libffi-dev
    libssl-dev
```

Next we use pip (python's package manager) in order to download the the Google SDK.

```
(env) $ python -m pip install --upgrade google-assistant
    -library
(env) $ python -m pip install --upgrade google-assistant
    -sdk[samples]
```

Next we gave to give the environment credentials in order to work properly with Google's OAuth platform. OAuth is Google's way to authenticate various users utilizing their products.

```
(env) $ python -m pip install --upgrade google-auth-
    oauthlib[tool]
(env) $ google-oauthlib-tool --scope https://www.
    googleapis.com/auth
/assistant-sdk-prototype \
  --save --headless --client-secrets /path/to/
    credentials.json
```

At the end one should see a prompt that directs you to a URL to finish the OAuth activation process.

```
Please visit this URL to authorize this application: https
    ://...
```

This will bring you to the Google Applications Action Console. From there we can register the product from Google's Console in order to keep track of the device. This page requires the user to sign into their Google device. Since the lamp is meant to be a stand alone project, it would be ideal if we also gave the lamp its own email address and Google credentials. That way any testing will not be able to hurt one of the developers during the testing process.

This screen after the sign-in process will have an authorization code that will be entered into the environment API in the terminal. if seccessful we should see a prompt similar to the prompt below.

```
credentials saved: /path/to/.config/google-oauthlib-tool/
credentials.json
```

This allows us to start testing and running sample code on the local machine.

Run sample code and initialize the device

At this point, you are ready to run the sample and make a query.

- Replace my-dev-project with the Google Cloud Platform project ID for the Actions Console project you created. To find the project ID in the Actions Console, select the project, click the gear icon, and select Project settings.
- Replace my-model with the name of the model you created in the previous step.

```
(env) $ googlesamples-assistant-hotword --project_id my
    -dev-project
--device_model_id my-model
```

Say "Ok Google" or "Hey Google", followed by your query. These are a few sample commands built into the Google Assistant framework:

- Who am I?
- What time is it?
- What is the weather in San Francisco?

If there is another cast-enabled device on the network, we can interact with it using normal vernacular such as:

- "Ok Google, play Spotify on the Kitchen Speaker."

When you run the sample, it will generate a device instance for your particular device. This device instance will be associated with the device model that you specified to run the sample. Find the device instance ID in the output for the sample. You will use this ID when running the sample so you can use Device actions.

```
device_model_id: my-model
device_id: 1C3E1558B0023E49F71CA0D241DA03CF #
    Device instance ID
ON_MUTED_CHANGED:
{'is_muted': False}
ON_START_FINISHED
```

Since we have the sample code running we now have a Google Assistant capable device on the network. We can now start implementing the same service to the raspberry pi instance to the system and extend that system to understand different parameters and custom commands.

Implement the system onto the Raspberry Pi Module.

Implementation system into the raspberry pi module should be easier then setting up the entire process over again. The assumption is that after all the the sample code is run the only addition to the device would be the custom command architecture and code. After that simple troubleshooting and changing of device ID's should occur and the product should be in normal operation.

Run and configure the device onto the Raspberry PI module.

This should be a similar process above, with the exception of the raspberry PI module being tested and implemented with the raspberrianOS. This might cause a few transference issues but they should be minor considering that both platforms are operating under a similar kernel.

Test basic commands for the Google Assistant

The Google Assistant needs to be able to associate a query with a command to send to your device. For this to work, you need to declare what kinds of abilities your device supports. These abilities are known as traits. You declare these traits within your device model.

Google has already created a wide variety of common traits found on many devices. These traits are not tied to just one device type, you can use them as you choose.

Since we have previously defined a model, we can add a trait to it by issuing the following:

- Open the project in the Actions Console.
- Select the Connected properties tab from the left navbar.
- Select the DEVICE MODELS tab.
- Click a model from the list to edit it.
- Click the pencil in the Supported traits box to add the trait.
- Select the OnOff checkbox. Click SAVE.
- Make sure to save changes to the model. Click SAVE again.

Implement custom commands onto the the Google Assistant Module.

The custom commands interface structure is presented inside the Google Assistant environment. If we turn on the environment running the Google Assistant on the device we will be able to see a certain code block when creating a query:

```
ON_RECOGNIZING_SPEECH_FINISHED:
{'text': 'turn on'}
ON_DEVICE_ACTION:
{'inputs': [{'payload': {'commands': [{'execution':
```

```
[{'command': 'action.devices.commands.OnOff',
'params': {'on': True}}], 'devices':
[{'id': 'E56D39D894C2704108758EA748C71255'}]}]},
'intent': 'action.devices.EXECUTE'}], 'requestId':
    '4785538375947649081'}
Do command action.devices.commands.OnOff with params {'
    on': True}
```

This will provide a marker in order to modify the source code in order to create custom command inferences. Now we need the source code.

```
(env) $ git clone https://github.com/googlesamples/
    assistant-sdk-python
```

In the repository we can locate the hotword.py file to send requests.

```
(env) $ cd assistant-sdk-python/google-assistant-sdk/
    googlesamples/
assistant/library
(env) $ nano hotword.py
```

For Raspberry Pi a little more is required for normal functionality.

First we must install the GPIO Package:

```
(env) $ pip install RPi.GPIO
```

These allows the Google Assistant to communicate with pins on the raspberry Pi module directly. From here the raspberry pi module must conform directly to the information hardcoded into the individual pins. This can be used to associate certain functions and values that the raspberry pi is influencing, like the movement of the lamp or the lights.

Test Custom commands on the Raspberry PI Module.

Testing the custom commands on the Raspberry PI module will be a delicate process. The system in place will have to work in phases.

Each associating with the design flow of not only this module but the other modules within the system. Just like in the OpenCV modules, there will be unit testing involved in the process.

The process flow of the testing starts as such:

- Set up simple interface for the Raspberry Pi module.
- test each custom command for recognition
- test each custom command for feasibility

Simple Interface

Within the initial testing phase of the raspberry pi, we are assuming a few factors within the overall project work-flow.

- All other module are either incomplete or untested
- All other components are incomplete or untested

Therefore, we need to create a system where the Google API module can go ahead with testing before the other modules have been completed. This calls for the "Simple Interface". The Simple Interface consists of the raspberry pi and a light. This will allow for the raspberry pi to operate without the need of the lamp and to display hooks for later processes to connect in the future once testing is complete and the other modules are ready to be integrated.

Recognition

Testing for recognition done on the raspberry pi is set when the user or tester says a command and it is recognized by the system. This will be displayed within the system terminal user interface and able to provide feedback to the tester. If there is a command that is too

vague or a command that is too specific the tester can then go back in and change the phrase in order to better integrate the operation to the system.

Feasibility

Once the phrase has been accepted by the programmer and the Google API module, then the next test can proceed. The feasibility test occurs when the command is set to turn on a light on the raspberry pi module. That way we can determine that a function is being executed on the device on the phone.

Test using other modules within project.

Once the testing has been completed independent of the other modules we can now start to integrate all the modules together. In reality the only module that the Google API module needs to connect to is the main module, or the controller. But since the controller is the main method for the entire project it is vital that these two modules communicate effectively. For more information please refer to the Testing and quality control section 10.

9.4.2 Google API Process Flow

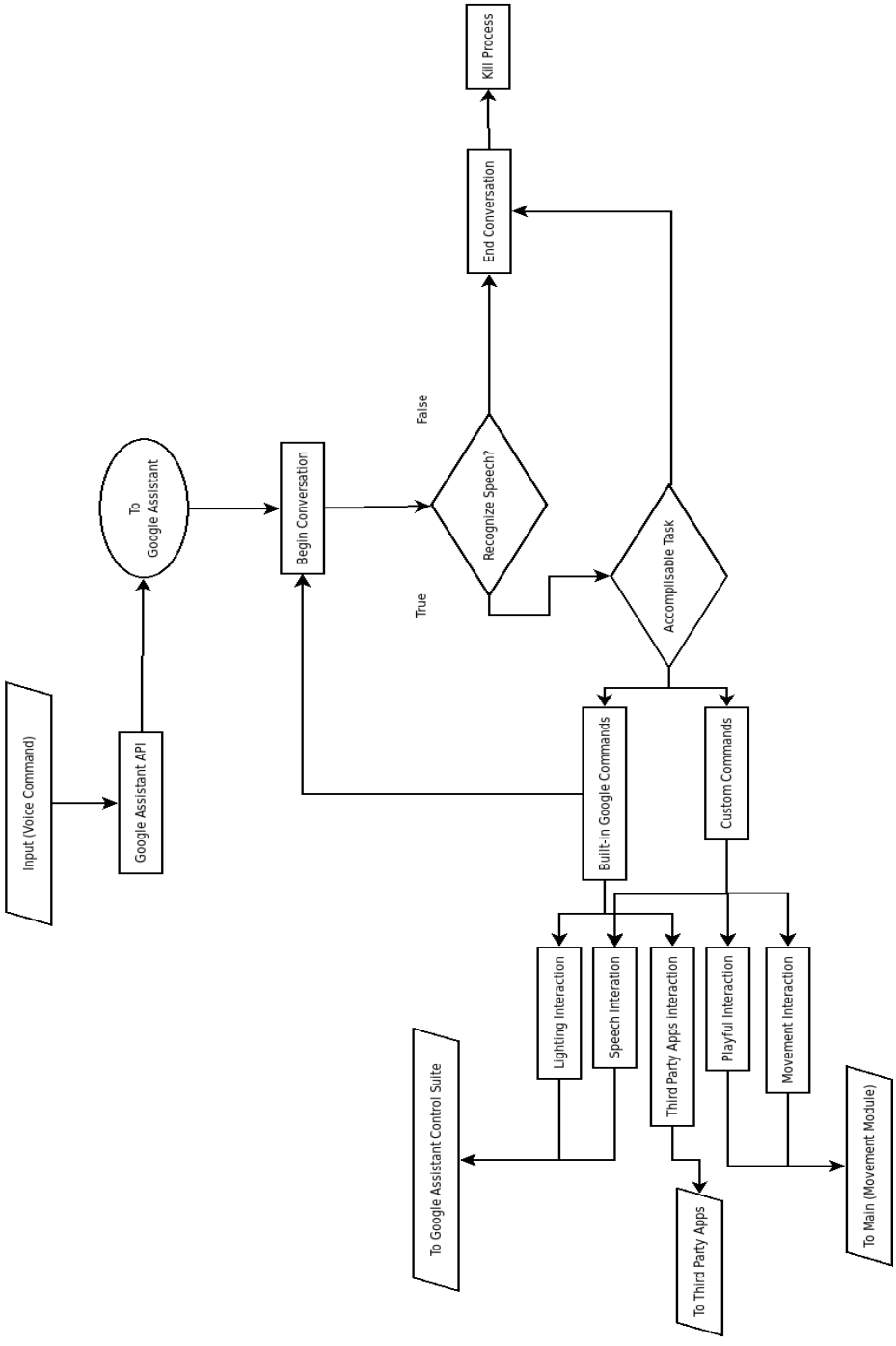

Figure 53: Google Assistant API Module Process Flow

The start of every process flow with the Google Assistant is the initial voice command or "wake up" command. With the standard Google Assistant Library this is achieved by saying "Okay, Google" or "Hey Google" within range of the microphone of the device. With the automated Lamp a new wake up command "Hey, Luxo" is introduced with the same operation as the traditional wake up command.

From the wake up command the Google voice assistant is constantly operational and waiting for the next command which assumes that a question is being asked. The Assistant can search through Google's search engines any question that the user desires an answer to. Google's assistant also contains a conversation feature that allows for an extended more natural dialogue between the device and the user. It utilizes previous questions in reference in order to expand its search.

If the Google Assistant recognizes the question that is being asked then it will reply with the answer it chooses to be the best. If not, then it will respond with a canned response that notifies the user that the question is not understood, malformed, or needs to be repeated. The Assistant in the case of a error response will end the conversation flow and the Google Assistant will need to be woken up again by the wake up command.

If the response is understood, there are a few categories that go into what the assistant does. A variety of different command interactions are displayed that allow for the user to interact with the automated lamp. These are: [50]

- Lighting Interaction
- Speech Interaction
- Third Party Application (Apps) Interaction
- Playful Interaction
- Movement Interaction

All of these interactions are triggered by either the custom commands or the standard built-in Google commands.

Lighting

A lighting interaction uses the automated lamp's LED light array to perform the specific command. These commands are mostly custom commands given that the light is a device that comes from the Lamp itself.

Speech Interaction

The Speech interaction is any interaction that the user gets a response in the form of the Google Assistant answering the command or question. This is mostly completed by the built-in Google Assistant and allows the user to access all of the Google assistant command suite.

Third Party Application Integration

The third party application (Third party apps) allows the user to access different apps built into the system. The Google Assistant has access to support various applications on other devices within the Google Home Ecosystem. For example, from the automated lamp you can turn on a supported television and play a movie on Netflix, on that television. This allows for an expansion of the user interface to include not only the lamp but every device connected to the Google Home Ecosystem.

Playful Interaction

The Playful Interaction allows the movement of the Lamp to coincide with a number of various custom commands. These commands are one off commands and the direct result of the interaction is to revert back to the wait on wake up command status.

Movement Interaction

The Movement Interaction allows for the utilization of the tracking

and detection of the camera module. This would include a command like ”look at me” when the user is not in the center of the frame. The movement command is sent to the main module to be processed by the movement module.

All of these commands, once completed, revert back to the wake up/ standby status.

9.4.3 Natural Language Processing

The natural language processing service is what the Google assistant service is built upon, therefore it is essential to acquire a base level understand of how the process works in order to understand the limitations of the automated lamp embedded with the Google assistant. The Google Natural Language Understanding Team states: [28]

> Our team comprises multiple research groups working on a range of Natural Language Understanding (NLU) projects. We collaborate closely with teams across Google, leveraging efficient algorithms, neural networks, and graphical and probabilistic models to help guide product development and direction. In doing so, the Google NLU team enables natural and assistive communication with users, finds answers to user questions, analyzes app store reviews for developers, and more.

Simply put, the natural language processing service (NLP), along with the automatic speech recognition service (ASR), create the backbone for the Google Assistant. This allows for not only command recognition but also an easier conversational flow throughout the command structure. We can look at a base description of how the NLP and ASR work together to create a dialog with any user that uses the Google assistant.

9.4.4 Google SDK Flow

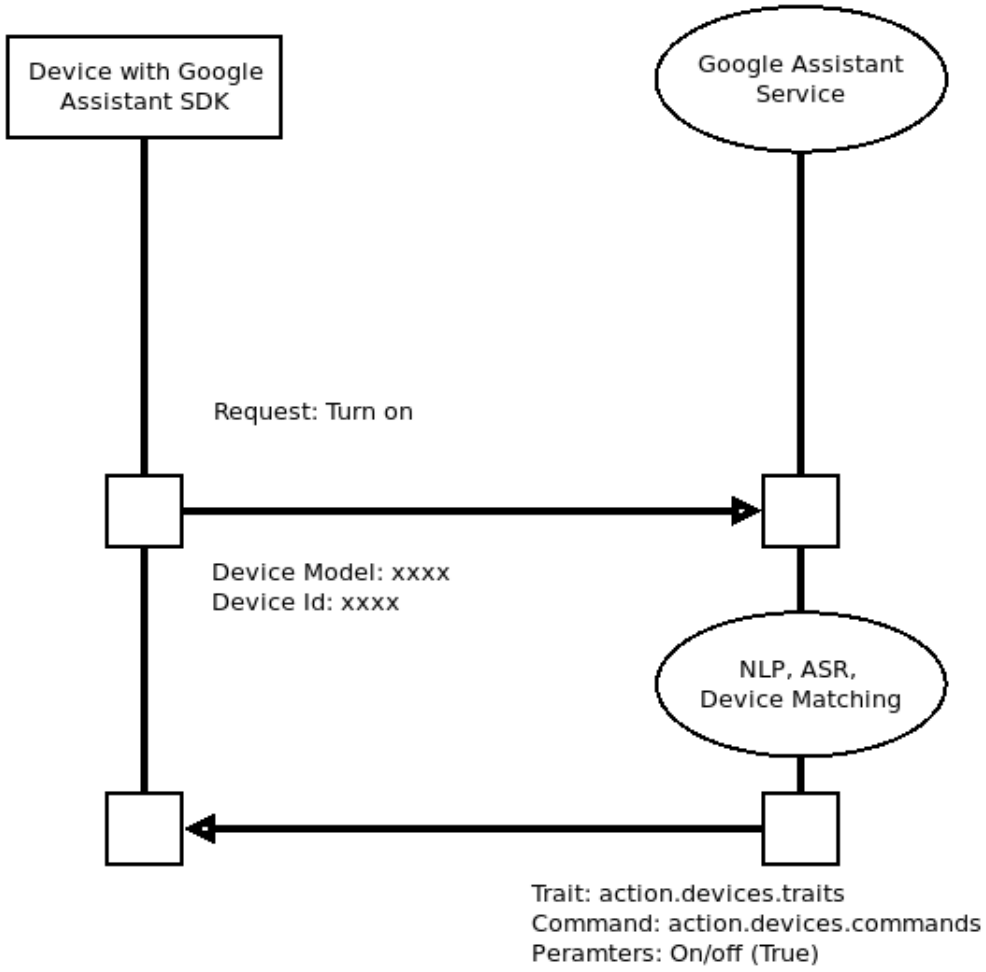

Figure 54: Google Assistant SDK NLP, ASR process flow

As shown above the NLP and ASR services are embedded into the Google Assistant service. Depending on the recognition of the Google assistant the user can expect certain responses and commands to be obey by using voice interaction.

9.4.5 Command Parsing

Command parsing is implemented through the natural language processing embedded into the Google Assistant service. The command parser runs through Google's Cloud Service. This allows the lamp to

have a low operational overhead while at the same time, utilizing powerful informational software from Google, all without cost to the team in research and development.

The command parser, built into the Google Assistant User Interface, extracts information from the Google Cloud Natural Language Service about people places and events that the Service recognizes. Not only does it understand information context, but it also understands intent and parses said intent. Which means that the lamp can have the ability to understand and respond back to emotional responses. This however is outside the per view for the initial release of the lamp.

Using the Natural Language Parser available from Google, we can understand better how the Cloud Service. The Cloud Service discovers 4 categories:

- Entities
- Sentiment
- Syntax
- Categories

If we were to use the sample sentence from Google: "Google, headquartered in Mountain View, unveiled the new Android phone at the Consumer Electronic Show. Sundar Pichai said in his keynote that users love their new Android phones." One sees the utilization of all categories to better understand the sentence.

9.4.6 Google Cloud Natural Language Service (Entities)

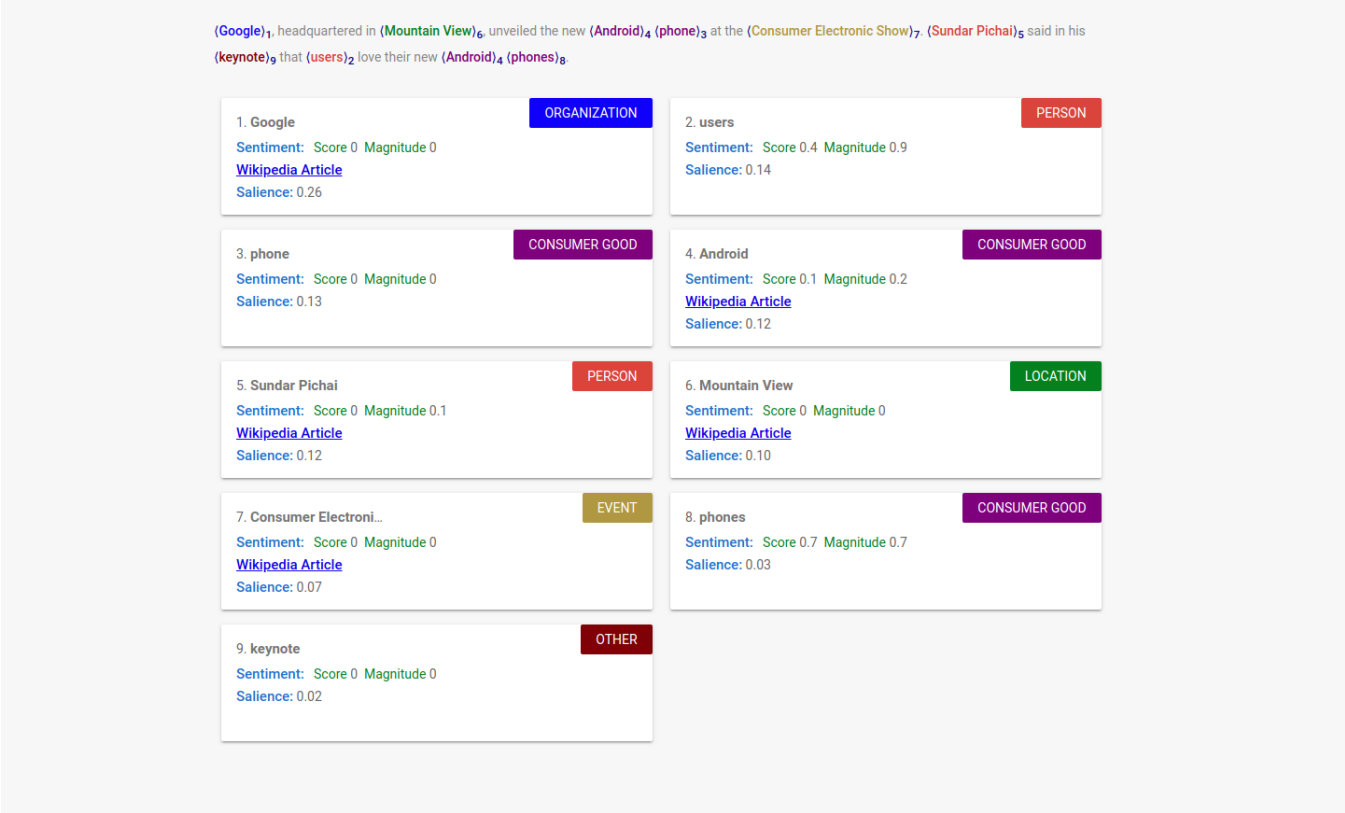

Figure 55: Screen-shot of the Google Natural language API Service in action

The Cloud Service collected all relevant words categorized into Entities, and blocked them into sub categories:

- Google (Organization)
- users (Person)
- phone (Consumer Good)
- Android (Consumer Good)
- Sundar Pachai (Person)
- Mountain View (Location)
- Consumer Electronics Show (Event)
- phones (Consumer Good)

- keynote (Other)

Each item collected gets tied to relevant information, perhaps it be a Wikipedia page or company website and is given a sentience and salience score. For sentience there are two types of scores, score and magnitude. Since each entity is given a score in order of salience. The Natural language parser can understand which part of the sentence is most important to the user.

9.4.7 Google Cloud Natural Language Service (Sentiment)

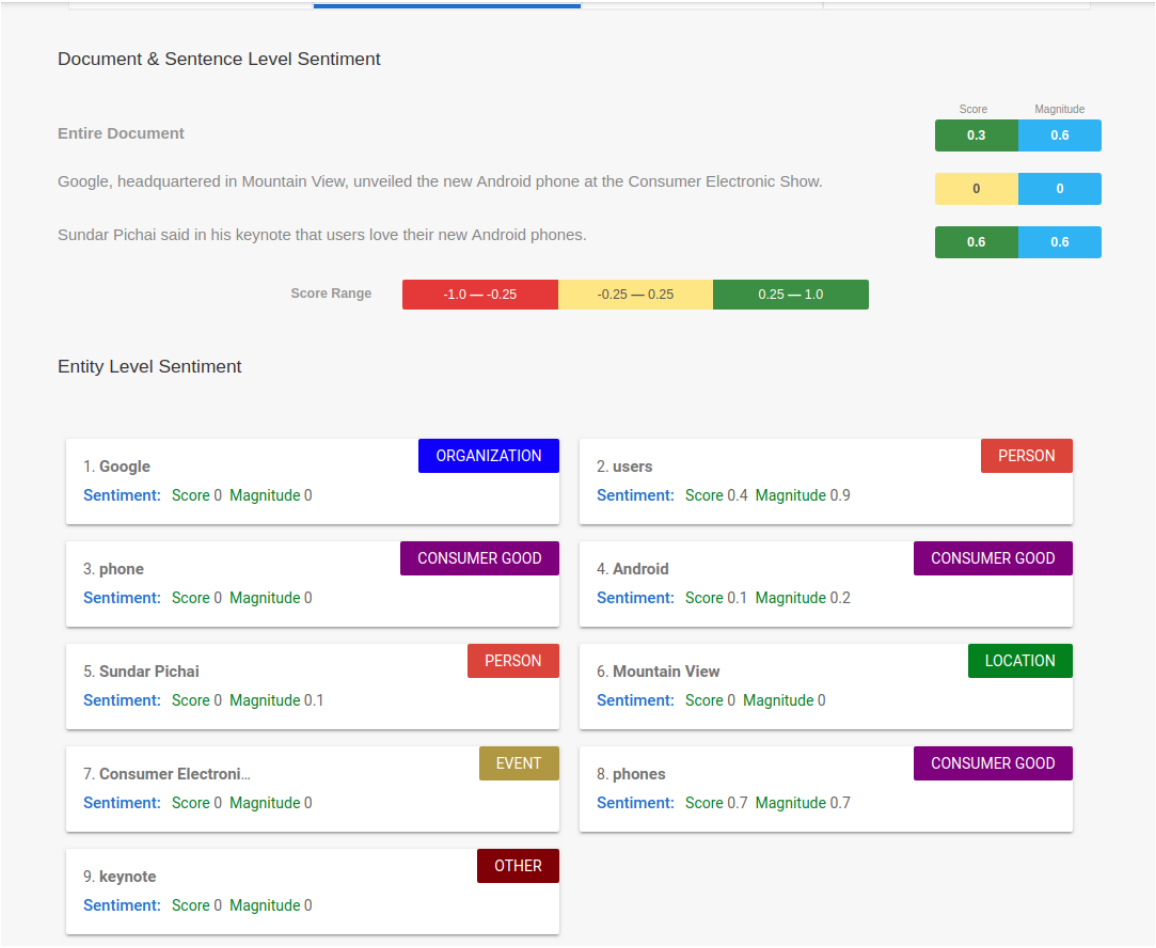

Figure 56: Google Cloud Service Sentiment evaluation

With the sentiment category, the service attempts to ascertain

the importance of the overall text. It rates each word with a level of sentiment in order to ascertain how important each sentence is in comparison with each other. In the example the sentence with the name of the Google CEO, Sundar Pachai, was rated with a slightly higher score then that of the other sentence which was in reference to the types of products the company (Google) sells. This is interesting considering that this suggests that the score and be subjective, respective to the user in question. For example, if we were searching for something and our interest was pointed to the product, then we are probably not looking for the Google CEO. However if we wanted to do more searches within Google's interface, having the CEO might be relevant. We can see that in certain respects the service is geared towards certain functional parameters.

9.4.8 Google Cloud Natural Language Service (Syntax)

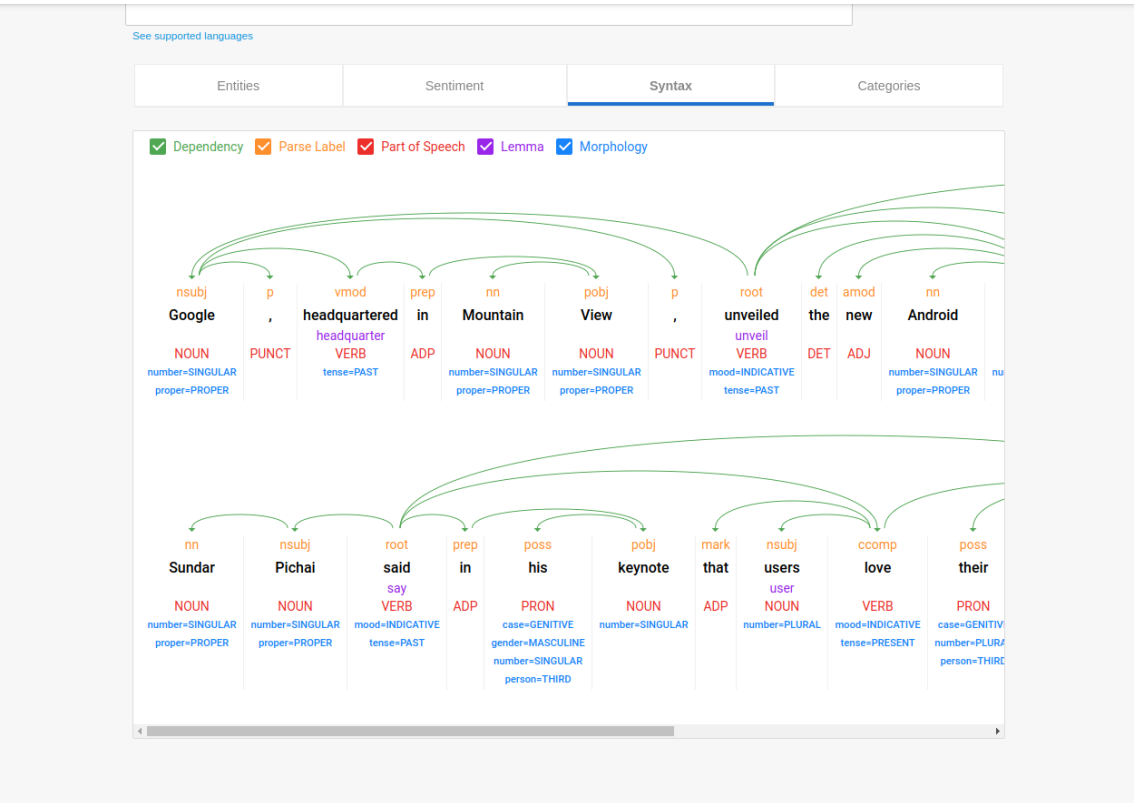

Figure 57: Google Cloud Service Syntax Parser

Along with most language recognition systems the Google syntax analyzer allows for parsing of syntactic systems within the language that we speak. The parser separates each sentence based on grammar rules and intensifies what category that the fragment belongs to. For English, each word is assigned a label according to its role within the speech (Noun, Verb, Adjective, etc.) and additionally given secondary characteristics. In the example above, in the NOUN category, the parser identifies a singular or plural noun, if its a proper noun, first, second or third person. For verbs there are different secondary categories, mood and tense, to indicate different dictates of speech. This allows the system to link together more information from a single sentence and allows the user to acquire more information from just one sentence.

9.4.9 Google Cloud Natural Language Service (Categories

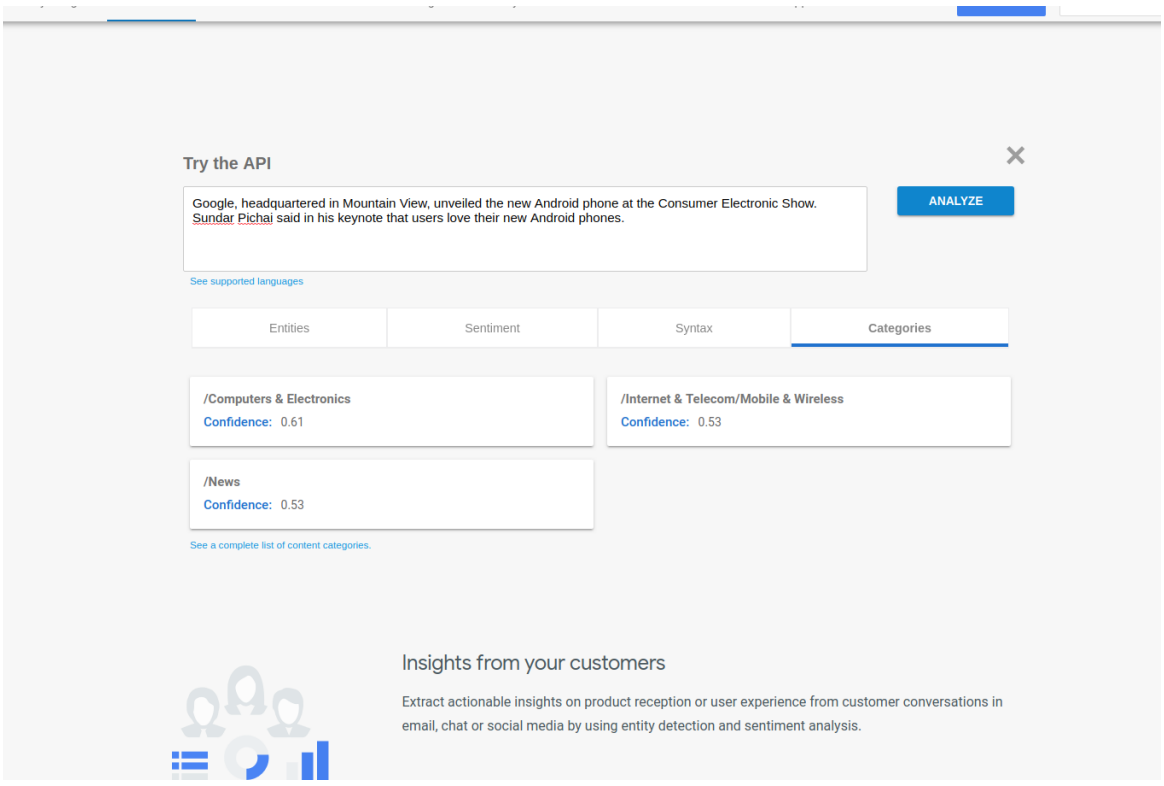

Figure 58: Google Cloud Service Category Parser

The category parser allows the Google Cloud Natural Language service to collect all the information from the previous sections and compile relevant information to the user. In this example, we are given three categories:

- Computers & Electronics
- Internet & Telecom/Mobile & Wireless
- News

Each category is given a confidence score. In reference to the text chosen the user, can be given relevant information in order to supplement the users query. This allows the parser to create more queries based on the text provided. This could supplement the lamps utilization in order to create a more hands-off experience.

The confidence score allows us to rank each category in order of importance to the block of text in question. This will allow the user to find relevant information about the subject at hand during interaction. A later adaptation of this system is to allow the natural language parser, through the Google Assistant to have a conversation with the user and inform the user about the day to day tasks, information, news, relevancy specific to each user profile. This will give the Automated Lamp a better and more clearer understanding of the user along with the user having a clearer understanding of the Lamp.

9.4.10 Commands of Interest

In order for the Automated lamp to operate properly for a general user, there needs to be a use for each command that is being implemented within the device along with the custom commands implemented within the module. Therefore, the commands of interest are any commands that can be utilized to create a better user experience. These commands should expand on the current Google Assistant base command list along with the understanding associated with each command. Some

of these commands will be implemented by prototype launch, while others would be commands that could be push through an update later on in the product life cycle or in a second version.

The commands that are part of the requirements in the system are as follows:

- Wake up Commands: ”Hey, Luxo” and ”Ok, Luxo”
- Light control commands: ”Dim the lights” or ”Brighten the Lights”
- Look commands: ”Look at me” or ”Look at this”

These will be used to create a basic user interface for all aspects of the device that requires user control. These commands allow the user to use certain functions that require a user to implement. The system, does not know what needs brightening or dimming according to the user. Additionally, the device does not need to be left on indefinitely. Specifically, because if any updates are implemented after launch there would be a restarted and rebooting process in order to install new hardware. There are, however, some commands that could be implemented once these commands have been throughly tested and the other minimum requirements have been met.

One aspect of commands that is under the optional requirements is the ability to utilize external music sources to facilitate interactions. These commands are ”Listen to this” or ”play this” types of commands. These commands serve two functions. They allow more user interaction by having the lamp interact with something more then the users voice. The ”listen to this” (More information in Section: 9.4.11) allows for the lamp to listen to music and bop its head once it recognizes a beat in the music. Paired with the built-in commands for music listening in Google, the user will be able to have a more involved interaction with the device.

Another possible interaction that is under the optional requirements is the connection and use for the Google Assistant to access

third party applications. For Example: ”Hey Luxo, open YouTube on my TV”. This allows the lamp to have access to the entire Google Home Suite. It also allows the user to be a part of the Google Home Ecosystem along with other products like Google Home, Google Home Mini, Chromecast, and Chromecast audio. The Google Assistant should be connected within the Google Home ecosystem and therefore, will have access to every app that has compatibility to the Google Home system. This could range from productivity applications like Todoist, to entertainment like Netflix. All of this without the necessity to add additional support for those systems.

Various commands for future use would be an ability to show a light array show for the user. For example, a light show command that would allow for the automated lamp to tilt its head up towards the ceiling and create a light show on the ceiling to display. This however would require a additional requirement for the hardware team and therefore has been left out of this particular schema. This does not mean that the Google Assistant does not have the possibility to perform these tasks.

9.4.11 Custom Commands

Custom commands are an expansion of the Google Assistant API. This is supported by Google and is aided by its natural language processing algorithm that comes pre-packaged with the software development kit. Below is a list of commands that we are deciding to implement for the lamp:

- ”Hey Luxo.”
- ”Ok, Luxo.”
- ”Look at me.”
- ”Look at this.”
- ”Listen to this.”

- ”Dim the lights.”
- ”Brighten the lights.”

Below is an overview diagram for the command structure process flow (figure: 9.4.12).

”Hey Luxo” and ”Okay, Luxo” is an additional wordage to the standard wake up commands. Generally, with the Google Assistant API, the phrase ”Hey, Google” or ”Okay, Google” activates the Google assistant and starts the listening service.

The ”Look at me” and ”Look at this” commands allow for a connection with the Google Assistant Module, the Camera Module (Section: 9.1.3) and the Movement Module (Section: 9.1). In tandem with the main module, the command is processed by the Google assistant, given to the camera module to look for the item of interest, in the case of ”Look at me” the object is the face, in the case of ”Look at this” the object is whatever the user is pointing to if the finger is in view, and finally use the movement module to move the lamp if needed.

”Listen to this” is an adaptation of the Google assistant built-in function, with an additional movement module functionality added. The Google assistant activates its music listening service, if the song is recognized, the lamp processes the beats associated with the music currently being played. It will then send a stream of that data to the movement module and proceed to bop ”its head” to the music.

The light control commands are for dimming and brightening the lights on command. ”Dim the lights” or ”Brighten the lights” allows for voice activated light control within the system.

9.4.12 Google Custom Command Overview

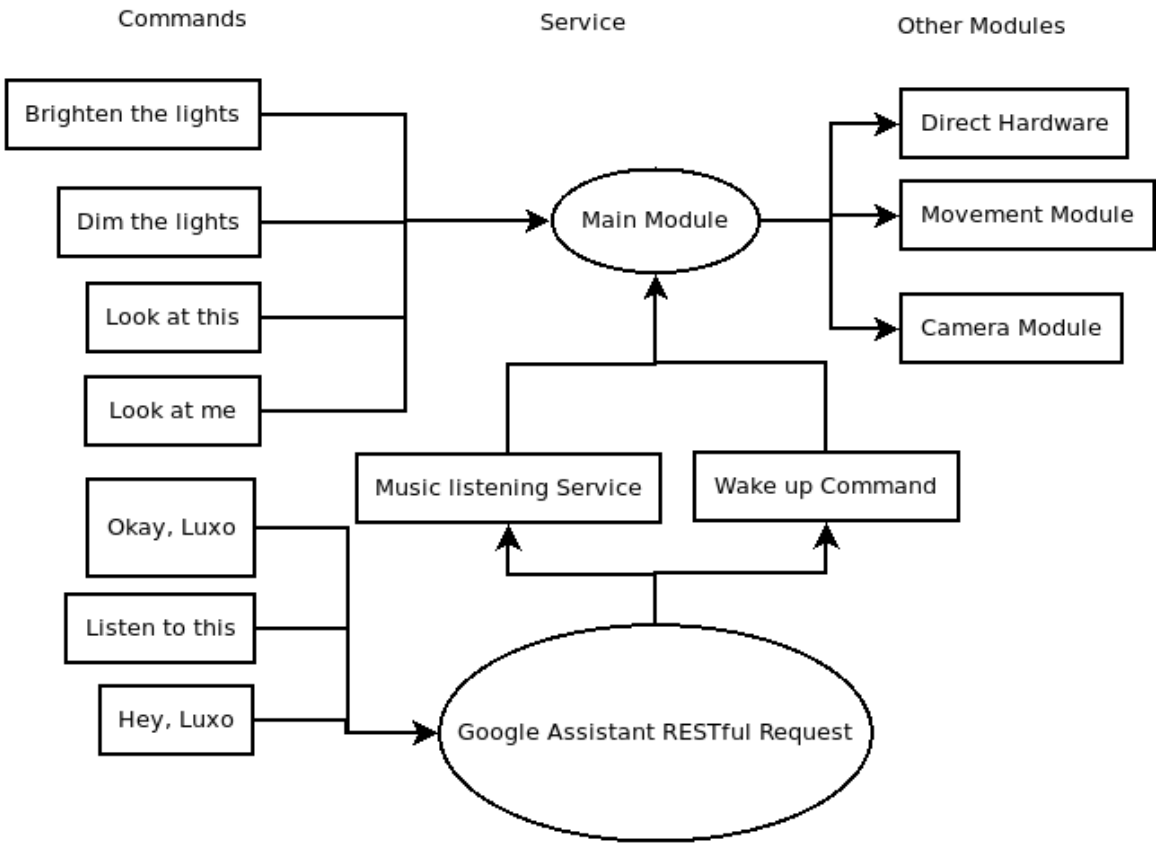

Figure 59: Google Assistant Custom Command Flow Overview

In summary, ideally one would like to see a system where the user has a genuine interaction with the device to the point where the user loses the sense that it is interacting with a machine.

9.5 The Main Module

The main method is the central processor that holds all modules together in order to allow the lamp to operate normally. This allows a good control structure and error handling hub to work its way into the project. The main method handles all throughput between the lamps various modules and allows the system to operate with an additional

check to verify that certain systems are functioning normally. This can offer a few advantages:

- The main method acts as a buffer for all modules
- It acts as an additional error checker in order to allow for easier handling of errors and error correction
- it can notify if a certain portion of the system is not working properly, and if so notify other aspects of the system in order to notify the user.

However, it also may offer some disadvantages:

- This process will be slower then interacting with the modules directly.
- The Main method itself can be a source for errors
- The main method can be more taxing to the rest of the system.

The team decided on this design process because of the benefits, in this case, would outweigh the disadvantages. It is therefore, necessary to offer some reasoning as to why we chose this model, what are its advantages in detail, and what are its drawbacks with solutions for combating said drawbacks.

9.5.1 Buffers

The main method acts as a buffer for all entry points onto the system. Although rare in this particular example, security is something of note during the forming of this lamp. The lamp connects to Google Home which leaves access to your gmail account and everything that the gmail account would have ties to, this could include:

- Netflix data
- Banking information
- Password information
- Organizational information
- Other Database related information

It is for these reasons that the lamp must not only be easy to use, but secure in doing so. The relevant information that your account has when using this device could be comprised if there is a back door inadvertently opened with the introduction of the lamp into the Google Home Ecosystem. The lamp must also act as the first line of defense.

We cannot directly tell what types of threats or vulnerabilities the lamp will face but we can use the main method to protect against these potential threats. Some being of future updates, others being items that we should include for its inception. These are:

- A function to check for erroneous speech patterns.
- A system for lamp placement in a secure area.
- A camera security measure upon focusing on the user.

The main method would be useful specifically in the case of the first item; erroneous speech patterns a well known tactic among cyber-security analysts is the database injection or allowing some script that would inadvertently allow access to the database. The main method defense would be in its transfer from one module to another. With this transfer there can be a quick check as to what information is going through and because there is no database that would trigger the script, the main method is safe. Therefore, a system to detect scripts should be put in place to defend against such an attack.

9.5.2 Additional Error Checking

As the additional buffer for the various modules to communicate with each other, the main module adds an additional error checker in order to verify the proper functions are being called during the runtime of the various processes. For example, if the microphone module collects a bit of information that does not make sense in terms of a command, the functions involved can send data of not only the data collected but additionally, a potential meaning with aid of the Google API module. There are various error handling functions that could be implemented in order to create a more error-free lamp.

In python, the main module will handle errors in the standard way according to python 3 documentation. Python allows error handling in a try/catch block. However, with python the coding is a little more streamline as opposed to other programming languages.

From Tutorials Point[51]:

> Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them
>
> - Exception Handling − This would be covered in this tutorial. Here is a list standard Exceptions available in Python: Standard Exceptions.
> - Assertions − This would be covered in Assertions in Python tutorial.

For python, error handling is performed with the standard exceptions listed below:

- Exception - Base class for all exceptions
- StopIteration - Raised when the next() method of an iterator does not point to any object.

- SystemExit - Raised by the sys.exit() function.
- StandardError - Base class for all built-in exceptions except StopIteration and SystemExit.
- ArithmeticError - Base class for all errors that occur for numeric calculation.
- OverflowError - Raised when a calculation exceeds maximum limit for a numeric type.
- FloatingPointError - Raised when a floating point calculation fails.
- ZeroDivisionError - Raised when division or modulo by zero takes place for all numeric types.
- AssertionError - Raised in case of failure of the Assert statement.
- AttributeError - Raised in case of failure of attribute reference or assignment.
- ImportError - Raised when an import statement fails.
- KeyboardInterrupt - Raised when the user interrupts program execution, usually by pressing Ctrl+c.
- LookupError - Base class for all lookup errors.
- IndexError - Raised when an index is not found in a sequence.
- KeyError - Raised when the specified key is not found in the dictionary
- NameError - Raised when an identifier is not found in the local or global namespace.
- UnboundLocalError - Raised when trying to access a local variable in a function or method but no value has been assigned to it.
- EnvironmentError - Base class for all exceptions that occur outside the Python environment.

- IOError - Raised when an input/ output operation fails, such as the print statement or the open() function when trying to open a file that does not exist.
- IOError - Raised for operating system-related errors.
- SyntaxError - Raised when there is an error in Python syntax.
- IndentationError - Raised when indentation is not specified properly.
- SystemError - Raised when the interpreter finds an internal problem, but when this error is encountered the Python interpreter does not exit.
- TypeError - Raised when an operation or function is attempted that is invalid for the specified data type.
- ValueError - Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified.
- NotImplementedError - Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented.

Error handling is either done with assertions or exceptions. Assertions are defined as a "sanity check" on the particular block of code in question. Instead of a if-then style exception much like the try-catch; the assertion is much like a raise-if coding style. For example, create exception if x is not working properly. A basic example of assertions can be viewed below.

```
def KelvinToFahrenheit(Temperature):
    assert  (Temperature >= 0),"Colder␣than␣absolute␣zero!"
    return ((Temperature−273)*1.8)+32
print KelvinToFahrenheit(273)
print int(KelvinToFahrenheit(505.78))
print KelvinToFahrenheit(−5)
```

With the above code executed, we can see that there will an issue because we know that -5 Kelvin does not make sense and therefore the assertion is implemented. The result follows:

```
32.0
451
Traceback (most recent call last):
File "test.py", line 9, in <module>
print KelvinToFahrenheit(-5)
File "test.py", line 4, in KelvinToFahrenheit
assert (Temperature >= 0),"Colder than absolute zero!"
AssertionError: Colder than absolute zero!
```

Exceptions are executed like events, and is set up in an if-then format within the code. This coding style provides a defense of the code being executed. Anything encapsulated within the block allows that block to be tested before continuation. The good thing about exceptions is that the error does not shutdown the entire program but rather isolates the problem and has the ability to continue if necessary. The syntax of the try-except block is shown below:

```
try:
    #code to be tried
except ExceptionI:
    #if code is caught by this exception execute this block of
        code
except ExceptionII:
    #if code is caught by this exception execute this block of
        code
else
    #if no exception execute this block of code.
```

Notice how we can create multiple exceptions in order to test for various instances. This allows for a way more versatility among coding the main module and its manipulation through all the various pieces of data. This could also be used to create a method for handling errors for handling module errors and unexpected shutdowns.

9.5.3 Non-functioning modules

Since we have various modules working in tandem with each other. The potential for modules to fail becomes greater as more and more functions are added to the device itself. This can become an issue if the situation does not address the potential pitfalls of the various aspects of the project. It is therefore, understandable that we have a system in place where the main module controls how systems work and is able to send potential information to other modules in order to keep the flow of the lamp from stopping functionality.

The main module is responsible for the various transfers of data and therefore will also be responsible for the implementation of sending data in case of a failure within the system. For example, if the Google API module cannot function for some reason, Google will release an error code in order to notify the user that the module is not working correctly. If this is the case the lamp should be able to utilize its movement generally, albeit with limited functionality. If this is the case, then the user should be notified, and the lamp should attempt to work normally. Much like an error log message from your computer.

There are a few general errors that the user should be notified of:

- Google API Module Failure
- Camera Hardware Failure
- Microphone Hardware Failure
- LED light array Failure
- Computer Vision Module Failure
- Machine Learning Module Failure
- Initial setup Failure
- Miscellaneous Failures

Although this list could vary depending on the types of issues we may encounter during the testing process. It is important that with the evolution of this device we can introduce artificial intelligence along with more functionality. One would also note that we should make sure that the main module controls its association on the user based on said users wants and needs.

10 Testing and Quality Control

10.1 Code Testing and Implementation

The project will primarily be programmed in the python development language. With this language comes certain aspects unique to assisting in the implementation, testing, and confirmation of results. One of the most important aspects of the language is the idea of a virtual environment. The idea behind a virtual environment is to create an environment capable running a particular version of python with libraries important to the running of the program. Therefore, there may be multiple environments created by the team with the use of testing different implementation schemes.

To maintain quality control across code and between programmers, the team decided to implement coding practices as well as coding standards to maintain the programs. More specifically, the team will implement Python's set style guide [15]. The key elements outline in the style guide include:

- Indentation
- Line length
- Blank line locations
- Proper import syntax
- White spacing
- Comment placement

Despite creating code that is well organized, following this guide would allow for the team to quickly identify, assess, and understand the purpose of a program with minimal effort. An example to demonstrate the rigor of the programming style is given below provided by the Python Software Foundation.

Yes:

```
# Aligned with opening delimiter.
foo = long_function_name(var_one, var_two,
                         var_three, var_four)

# More indentation included to distinguish this from the
    rest.
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)

# Hanging indents should add a level.
foo = long_function_name(
    var_one, var_two,
    var_three, var_four)
```

No:

```
# Arguments on first line forbidden when not using
    vertical alignment.
foo = long_function_name(var_one, var_two,
    var_three, var_four)

# Further indentation required as indentation is not
    distinguishable.
def long_function_name(
    var_one, var_two, var_three,
    var_four):
    print(var_one)
```

10.2 Code Testing

Within the Python classes, there is a framework called `unittest` that allows the programmer to test the code in a methodical, repeatable way. The idea behind unit testing is that there are predetermined cor-

rect and incorrect values as determined by the programmer. These correct and incorrect values are identified from desired output as well as the expected output from the program. In addition, there are deeper concepts in unit testing that allow for robust testing and implementation. Within the Python documentation, these concepts are described as follows:

Test Fixture

Definition 10: A *test fixture* represents the preparation needed to perform one or more tests, and any associate cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.

Test Case

Definition 11: A *test case* is the smallest unit of testing. It checks for a specific response to a particular set of inputs. unittest provides a base class, TestCase, which may be used to create new test cases.

Test Suite

Definition 12: A *test suite* is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.

Test Runner

Definition 13: A *test runner* is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

Therefore, unit testing is therefore a valuable asset in the program-

mer's toolbox. Thus, in creating the project, the team has identified specific areas of the project that would benefit from the unit testing framework. In particular, computer vision and hardware control.

More specifically, to determine the effectiveness of the team's computer vision implementation, especially object and facial detection, test images will be loaded into the unit testing framework. These default images will have present both objects, faces, or none at all. Therefore, with predetermined correct and incorrect responses, the team will run the algorithm through the unit testing allowing for a definitive yes or no answer as to whether the program is running effectively.

Inherent to the idea of machine learning is the idea of effectiveness. That is, how accurate the algorithm is at the task it was set to. Therefore, as described in Section 9.3, the testing of the machine learning algorithm is a part of the algorithm itself.

10.3 Physical Testing

Due to the unique nature of the project at hand, not only does the computer vision, machine learning, and hardware softwares need to cooperate, but they need to be effectively implemented to the physical lamp. More specifically, there are aspects to the lamp that would require extremely precise modeling to capture effectively and due to the time constraint of the project, it is difficult to do such. Therefore, the physical testing will be necessary to debug possible problems.

The primary difference between the 3D model and the physical model are the servos. Within in the machine learning algorithm, the joints are modeled as simple movement angles that are allowed to move at a velocity similar to that of the servos. What is difficult to portray are the torque and ramp-up times, and as such provide a major difference to be overcome by the design team.

Another inherent difference between the models is that the physical model will have weight and inertia to the movement. This is predicted based upon previous experiences to cause issues with jerkiness, sharp

movements, and possible tip-over. Therefore, when training the model, the team will attempt to best replicated the weight of each component to allow the learning algorithm to be as close to the physical model as possible.

In addition, servo limits are indeed present with the servos having the capability of -90 degrees to +90 degrees. Yet, due to the geometry of the lamp, there are restrictions on the extent of movement that the servos have within this domain. Therefore, the team has identified and measured these areas as possible problem areas that are anticipated to be simple limit fixes within the machine learning algorithm.

With respect to the hardware itself, there are multiple points of possible error to be identified. To start, the camera along with the raspberry pi could be slower than anticipated in transferring images and processing them in the computer vision algorithm. It is therefore, the team's responsibility in implemented algorithms, such as image pyramid processes with Gaussian filtering, to decrease the latency of the computer vision software. If that same task can be just as effective, to be tested by unit testing as described in Section 10.2, then it will be implemented.

In a similar vein, once the machine learning is complete, the output to be used with the output information of the main function could prove insufficient in the fact that the learning output is not optimal and therefore introducing lag into the system.

Strictly hardware, there are the possibilities of electronics errors due to faulty wiring, overload of amperage to the system, and insufficient processing power. All of these aspects could prove to be detrimental to the project and if found to be the case, the team has acquired multiple Raspberry Pis in order to continue with the project.

Testing the hardware is also a challenge. The idea of unit testing here would prove to be ill-equipped for the task. In general, besides testing limits of the lamp to determine the range, effectiveness of tracking, and output of tasks, the team is interested in creating smooth physical movement of the lamp. This is a subjective measure and therefore has no set defined metric. The traits that the team will be looking

for is minimal jerkiness, minimal latency between input and output, and focusing ability. With the information of this output, the team will then alter the machine learning algorithm as they see fit to correct with these problems. Overall, the process will be trial-and-error and therefore multiple iterations would be needed.

11 Results

11.1 Implementation Results

11.1.1 3D Model Training Results

The first step in determining the validity of implementation rests with whether the 3D model of the lamp was adequately able to learn. Thus, there will be particular characteristics of interest in the movement of the lamp model. There are two main categories of results: quantitative and qualitative. That is to say, there is a set of results pertaining to measurable, defined metrics outlined in the initial requirements. Then there is another set of results pertaining to the subjective opinion of the team in relation to the quality of the movement.

Due to the nature of the project and the state it is at, the project results are unable to be obtained at this time. The results will be included at the commencement of Senior Design II.

11.2 Requirements Check

Due to the nature of the project and the state it is at, the project results are unable to be obtained at this time. The results will be included at the commencement of Senior Design II.

12 Human-Robot Interaction Study

Humans will be living more and more closely with robots in the near future. Between assistant robots, delivery robots, and automation robots there will be new and interactive physical tools surrounding the everyday person. As such, studying the interaction, whether it be emotional or physical, between a human and a robot is paramount to successfully having both exist harmoniously. [52]

The idea of human-robot interaction was first conceptualized by Isaac Asimov in his novel, *I, Robot* [53], where he detailed the three laws of robotics:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey any orders given to it by human beings, except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Numerous studies of human-robot interaction have been conducted in the past [54–57]. The majority of which focus on the acceptable types of interactions a robot must exhibit with a human. Such interactions are gauged based upon the response of the human.

The project the team worked on involves a robot capable of interacting with and assessing situations presented to it. Namely, the robot will be able to assist with illuminating objects of importance, recognize commands given to it, and respond in a playful manner. This ability to convey emotions through movement, and possibly light, would mean that the human user would also respond with emotions towards the lamp.

The study here is an extraneous, but nevertheless important, aspect to the project team's task. Studying the interaction between the

lamp and the user and the effectiveness it may have with respect to easing workload or improving a mood of the user.

Within this section, a testing schedule is provided to guide the project team towards the goal and gather such information. This information could provide a valuable step to the community regarding the human-robot interaction and the effectiveness of certain interactions.

12.1 Interaction Goals

At the end of the project, the team desires for the result to prove useful to the user. That is, there is a burden that is eased or a task that is improved due to the presence of the teams product. Possible improvements could include ability to illuminate objects of interest, locate objects, assist in companionship, etc. Therefore, the interaction goals are as follows:

- Useful illumination
- Ability to show basic emotions
- Interact and understand commands from the user
- Show playfulness in actions
- Human does not need to continuously interact for useful operation

12.2 Test Procedures

There will be two main categories of testing: those concerning the lamp's personality and those concerning the lamp's usefulness. A proper tool is one that is easy to use, intuitive, and requires little to no instruction.

Therefore, to test the lamp's usefulness, users will be tasked with going about activities that would enact certain modes of the lamp. The user will then rate the lamp on how well it seemed to perform these tasks. The team will take this information and regroup to form new ways and possibly implementations that would enable better user interaction.

The tasks of interest to the group are:

- Look up at voice command
- Responds to tracking command
- Responds to music command
- Responds to internet search command

More specifically, the lamp must be able to adequately and seamlessly operate with minimal latency, error, or stagger. If commanded to illuminate a piece of paper or to look at something specifically, there will be no hesitation seen by the lamp or lag from command to action. This delay is anticipated to be the most difficult part of the project and its impact on the user. A delay of action would prove to be frustrating to users and therefore no interest in the use of the lamp.

13 Future Work and Recommendations

At the current point of the project, the team has formulated an approach to creating, programming, and training the animatronic lamp. The goal of the next semester is to implement the plans and prove the effectiveness of the approach taken. That is, as described in Section 11, the team will be executing the ability of the project to meet the original expectations formulated at the beginning of the project.

Recommendations for the project will be included at the conclusion of Senior Design II.

14 Concluding Remarks

In summary, the project proposed by the group represents the fusion of creativity, challenging concept design and implementation, and exploration in the fields of robotics, computer vision, and machine learning. To prepare for undertaking such a task, the team has set forth requirements, their respective importance to the final project, and the difficulty of each task. The result of these requirements is a set of deadlines which the project team is prepared to meet to see the success of the final project. The project team is willing to contribute their strengths to the project but also enter fields of which they are not familiar to make the project not just a capstone, but a learning experience.

The importance of this project not only pushes the field of robotics forward, but also creates a human-robot interaction platform for the everyday user. With proper implementation, marketing, and capital this project is something that could be seen in every household in the near future. The ground work, done by Google has allowed in the present day for common consumers to get more acclimated with speaking and interacting with devices they use everyday. This lamp would be an entry point to the business barrier in the Google Home ecosystem.

References

[1] module. *Webopedia*. URL https://www.webopedia.com/TERM/M/module.html. Accessed on April 20, 2018.

[2] Introducing raspberry pi hats. URL https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/.

[3] K. Walsh. Slack: A powerful, private (free) team tool that educators are adopting. November 2016. URL https://www.emergingedtech.com/2016/11/slack-a-powerful-private-free-team-tool-that-educators-are-adoptin

[4] About trello. URL https://trello.com/about/logo. Accessed on April 21, 2018.

[5] What is a kanban board? URL https://leankit.com/learn/kanban/kanban-board/. Accessed on April 21, 2018.

[6] G suite by google cloud. URL https://gsuite.google.com/. Accessed on April 21, 2018.

[7] Github. *Wikipedia*, April 2018. URL https://en.wikipedia.org/wiki/GitHub.

[8] Github. *Wikipedia*, April 2018. URL https://en.wikipedia.org/wiki/Git.

[9] L. Orsini. Github for beginners: Don't get scared, get started. *ReadWriteWeb*, September 2013. URL https://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1/.

[10] Pulse width modulation used for motor control, Feb 2018. URL https://www.electronics-tutorials.ws/blog/pulse-width-modulation.html.

[11] URL https://grabcad.com/library/servo-hxt900-micro.

[12] Computer vision. *Wikipedia*, April 2018. URL https://en.wikipedia.org/wiki/Computer_vision.

[13] S. Mallick. Object tracking using opencv (c++/python). *Learn OpenCV*, February 2017. URL https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/.

[14] Face detection using haar cascades. *OpenCV*, August 2017. URL https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html.

[15] Pep 8 – style guide for python code, . URL https://www.python.org/dev/peps/pep-0008/.

[16] K. Reitz. Structuring your project. *The Hitchhiker's Guide to Python*, 2016. URL http://docs.python-guide.org/en/latest/writing/structure/.

[17] T. Ball. Train your own opencv haar classifier. *Coding Robin*, July 2013. URL http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html.

[18] S. Tiwari. Face detection in python using a webcam. *Real Python*, July 2014. URL https://realpython.com/face-detection-in-python-using-a-webcam/.

[19] A. Saha. Read, write and display a video using opencv (c++/ python). *Learn OpenCV*, June 2017. URL https://www.learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/.

[20] R. Ajna and T. Hersan. Training haar cascades. *mememememe*. URL https://memememememememe.me/post/training-haar-cascades/. Accessed on April 20, 2018.

[21] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3): 500–513, 2011. URL http://lmb.informatik.uni-freiburg.de/Publications/2011/Bro11a.

[22] Person (https://stackoverflow.com/users/5134880/person). error: (-215) !empty() in function detectmultiscale. Stack Overflow. URL https://stackoverflow.com/questions/30508922/error-215-empty-in-function-detectmultiscale.

URL:https://stackoverflow.com/questions/30508922/error-215-empty-in-function-detectmultiscale (version 2018-04-20).

[23] Thread (computing). *Wikipedia*, April 2018. URL https://en.wikipedia.org/wiki/Thread_(computing)#Single_vs_multiprocessor_systems.

[24] Python programming/threading. *Wikibooks*, . URL https://en.wikibooks.org/wiki/Python_Programming/Threading. Accessed on April 20, 2018.

[25] Machine learning. *Wikipedia*, Mar 2018. URL https://en.wikipedia.org/wiki/Machine_learning.

[26] M. Copeland. What's the difference between artificial intelligence, machine learning, and deep learning? *Nvidia Blogs*, July 2016. URL https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-lea

[27] M. Varone and D. Mayer. What is machine learning? a definition. *Expert System*, Mar 2018. URL http://www.expertsystem.com/machine-learning-definition/.

[28] What is machine learning? *Google Cloud*, Nov 2017. URL https://cloud.google.com/what-is-machine-learning/.

[29] Artificial intelligence (ai). *Techopedia*. URL https://www.techopedia.com/definition/190/artificial-intelligence-ai. Accessed on August 21, 2018.

[30] What you need to know about artificial intelligence. *StopAd Blog*, January 2018. URL https://stopad.io/blog/artificial-intelligence-facts.

[31] Google ai defeats human go champion. *BBC News*, May 2017. URL http://www.bbc.com/news/technology-40042581.

[32] Artificial intelligence: Google's alphago beats go master lee sedol. *BBC News*, March 2016. URL http://www.bbc.com/news/technology-35785875.

[33] A. Madrigal. Could self-driving trucks be good for truckers? *The Atlantic*, Feb 2018. URL http://www.bbc.com/news/technology-35785875.

[34] Giving robots a sense of touch. *BBC News*, June 2017. URL http://news.mit.edu/2017/gelsight-robots-sense-touch-0605.

[35] Narrative science employs natural language generation. *Nanalyze*, January 2017. URL https://www.nanalyze.com/2017/01/narrative-science-natural-language-generation/.

[36] D. Shewan. 10 companies using machine learning in cool ways. *Wordstream Blog*, Feb 2018. URL https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications.

[37] N. Castle. Supervised vs. unsupervised machine learning. *DataScience.com*, July 2017. URL https://www.datascience.com/blog/supervised-and-unsupervised-machine-learning-algorithms.

[38] M. Dymczyk. Distributed machine learning on the jvm without a ph.d. *GeeCON Prague 2015*, Oct 2015. URL https://www.slideshare.net/MateuszDymczyk/geecon-prague-2015.

[39] D. Kauchak. Unsupervised learning. *CS451*, 2013. URL http://slideplayer.com/slide/4423102/.

[40] Poul Petersen. Bigml spring 2014 release webinar - clustering! *BigML Spring 2014 Release Webinar*, June 2014. URL https://www.slideshare.net/bigml/big-ml-spring-2014-webinar-clustering.

[41] V. Maini. Machine learning for humans, part 5: Reinforcement learning. *Medium*, Aug 2017. URL https://medium.com/machine-learning-for-humans/reinforcement-learning-6eacf258b265.

[42] A. Juliani. Introducing: Unity machine learning agents. *Unity Blog*, Sep 2017. URL https://blogs.unity3d.com/2017/09/19/introducing-unity-machine-learning-agents/.

[43] M. Mattar. Agents. *ML Agents*, Mar 2018. URL https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Design-Agents.md.

[44] A. Juliani. Brains. *ML Agents*, Mar 2018. URL https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Design-Brains.md.

[45] V. Pierre. Monitor. *ML Agents*, Mar 2018. URL https://github.com/Unity-Technologies/ml-agents/blob/master/docs/images/monitor.png.

[46] A. Juliani. Balance. *ML Agents*, Mar 2018. URL https://github.com/Unity-Technologies/ml-agents/blob/master/docs/images/balance.png.

[47] J. Portilla. Complete guide to tensorflow for deep learning with python. *Udemy*, Jan 2018. URL https://www.udemy.com/complete-guide-to-tensorflow-for-deep-learning-with-python/.

[48] Register the device model. *Google Developers*, March 2018. URL https://developers.google.com/assistant/sdk/guides/library/python/embed/register-device.

[49] Google assistant sdk overview. *Google Developers*, March 2018. URL https://developers.google.com/assistant/sdk/overview.

[50] Handle commands on the device. *Google Developers*, March 2018. URL https://developers.google.com/assistant/sdk/guides/library/python/extend/handle-device-commands.

[51] Python - exceptions handling. URL https://www.tutorialspoint.com/python/python_exceptions.htm.

[52] A. Clarke. *2001: A Space Odyssey*, volume 1. New American Library, 1968.

[53] Isaac Asimov. *I, robot*, volume 1. Spectra, 2004.

[54] Christopher Deligianis, Christopher John Stanton, Craig McGarty, and Catherine J Stevens. The impact of intergroup bias on trust and

approach behaviour towards a humanoid robot. *Journal of Human-Robot Interaction*, 6(3):4–20, 2017.

[55] Elizabeth Kathleen Phillips and Florian G Jentsch. Supporting situation awareness through robot-to-human information exchanges under conditions of visuospatial perspective taking. *Journal of Human-Robot Interaction*, 6(3):92–117, 2017.

[56] Florent Levillain and Elisabetta Zibetti. Behavioral objects: the rise of the evocative machines. *Journal of Human-Robot Interaction*, 6 (1):4–24, 2017.

[57] Gordon Briggs, Tom Williams, and Matthias Scheutz. Enabling robots to understand indirect speech acts in task-based interactions. *Journal of Human-Robot Interaction*, 6(1):64–94, 2017.