4-2022

LED Enhancements to the MIMIR Animatronic Figure

Collin J. DeVillier

LED Enhancements to the MIMIR Animatronic Figure

by

Collin J. DeVillier

Undergraduate honors thesis under the direction of

Dr. Shuangqing Wei

Department of Electrical and Computer Engineering

Submitted to the LSU Roger Hadfield Ogden Honors College in partial fulfillment of the Upper Division Honors Program.

April 2022

Louisiana State University
& Agricultural and Mechanical College
Baton Rouge, Louisiana

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

The MIMIR animatronic figure is robotic character developed as part of Louisiana State University's capstone design program. To fulfill the thesis requirements of the LSU Ogden Honors College, LED lights were added to the robotic system to enhance the character's storytelling abilities.

First, definitions of relevant terminologies are provided. Next, the background and objectives of the MIMIR animatronic figure are discussed. A detailed description of the MIMIR animatronic is provided, along with a description of the LED additions. Following, a description of the design and analysis of the LED system is provided. Finally, a description of the future applications of the system is provided, along with the system's code.

Definitions

Animatronic – a robotic system designed to tell a story. Animatronic figures are often found in theme parks and in the film and television industry.

Theme Park – "an amusement park in which the structures and settings are based on a central theme [1]."

Themed Entertainment – "the creation of an artificial environment where various elements bring to life a thematically driven story for immersing visitors in a strongly identified or branded setting [2]." Themed entertainment is inclusive of theme parks, museums, escape rooms, or other physical environments.

CHAPTER I

THE MIMIR ANIMATRONIC

Background

Legacy Animatronic Figures

Animatronic figures have been used in the themed entertainment industry for the last fifty years. The term “animatronic” is a derivative of *Audio-Animatronic*, the name trademarked by the Walt Disney Company to describe their new line of robotic characters in the middle of the twentieth century [3]. Walt Disney Imagineering, formerly known as WED Enterprises, developed the first Audio-Animatronic human for the 1964 New York World’s Fair [4]. The animatronic character featured was Abraham Lincoln, who delivered the Gettysburg Address to guests at the fair. In the decades since, animatronic characters have increased in complexity.

Figure 1:Animatronic Abraham Lincoln developed by the Walt Disney Company [4]

Current Animatronic Figures

In 2017, Disney unveiled their newest Audio-Animatronic, the Na'vi Shaman. This animatronic contains 42 degrees of freedom in the head alone, making it Disney's most complex animatronic at the time [5]. The Na'vi Shaman is an example of the growing possibilities of animatronic systems. The increasing complexity of the systems allow engineers and designers to give the characters more detailed facial expressions and mobility, enhancing their ability to immerse guests in compelling narratives.

Figure 2: Na'vi Shaman from Disney's Animal Kingdom theme park in Orlando, Florida [5]

Since the advent of the Audio-Animatronic, these robotic figures have only provided a uni-directional method of storytelling. Animatronic figures conveyed a pre-written story to guests; guests could not engage with an animatronic to change the animatronic figure's narrative. Even though these animatronic figures have become more complex in design, most animatronic figures that reside in theme parks follow this uni-directional storytelling method. Though the last decade has seen leaps in animatronic technologies, there remains a gap separating modern theme parks from the influences of bi-directional storytelling.

A Future of Animatronic Figures

In 2018, Disney unveiled a new autonomous figure as they opened *Guardians of the Galaxy - Mission BREAKOUT!* at Disney's California Adventure park in Anaheim, California. Called the Vyloo, these small animatronic figures interact with guests. When asked about the new animatronic figures, Leslie Evans, Senior Research and Development Imagineer said, "We were really interested in the idea of creating some little guys that could truly respond to and interact with guests [6]."

Figure 3:Vyloo, Disney's interactive animatronic figures located at Disney's California Adventure theme park in Anaheim, California [6]

It was recently revealed that in 2011, Walt Disney Imagineering developed an animatronic figure called Destini that used computer vision to interact with guests. According to Research and Development Imagineer Alfredo Ayala, the project was designed to answer the question, "What happens when our characters become reactive and responsive to our guests [7]?" This bi-directional form of storytelling allows guests to interact with the animatronic figures. The motivation of this project is to build on this foundational principle and showcase a potential future of animatronic interactions using current advancements in technology.

Objective

The objective of the MIMIR animatronic is to develop a story driven animatronic head for the themed entertainment industry that moves to track users and verbally respond to auditory cues. The animatronic will move its eyes, eyelids, mouth, and head. In addition, the animatronic will use a computer vision system to track a user as they move about a room. The system will leverage a speech recognition system to allow users to deliver specific key phrases that will trigger the movement and speech of the animatronic system.

Concept Overview

When a user wishes to interact with the MIMIR animatronic figure, the user will walk in front of the animatronic and pick up an ornate bowl. The MIMIR animatronic figure will move to follow the location of the bowl. The user will be free to walk about the entire space, and the animatronic figure will move its head to keep eye contact with the user. The user can speak any one of fourteen unique key phrases. Once spoken, the MIMIR animatronic figure will respond by moving its head and speaking to the user. The key phrases can be spoken in any order and with any frequency. Once the user completes their interaction with the animatronic figure, the user will return the bowl to its original location.

Movement Profile Overview

The MIMIR animatronic figure contains four distinct movement functions. The figure can move its head, eyes, eyelids, and mouth. The servos are connected to the computer vision system and speech recognition system in accordance with table 1.

Table 1: Servo listing with corresponding system that controls the servo's movement.

Servo Name	System Connection
Head Yaw	Computer Vision System
Eye Yaw	
Head Pitch	Speech Recognition System
Head Roll	
Eye Pitch	
Mouth	
Left Eyelid	Eyelid Randomizer System
Right Eyelid	

Computer Vision System

The MIMIR animatronic figure leverages a computer vision system to track a unique object. The computer vision system uses the MobileNet v2 algorithm as the basis for the model. A custom dataset was created with images of the ornate bowl from different distances, lighting environments, and object orientations. The dataset was trained on the MobileNet algorithm to produce a unique mode. When the computer vision algorithm detects the bowl, it places a bounding box around the object. An algorithm uses the bounding box to calculate the location of the object with respect to the center of the animatronic's head. This location, in degrees, is then sent to the head and eye yaw servos. Then, the eye and yaw servos receive the location and move to align with the user's location. Figure 4 depicts a flowchart of the computer vision system.

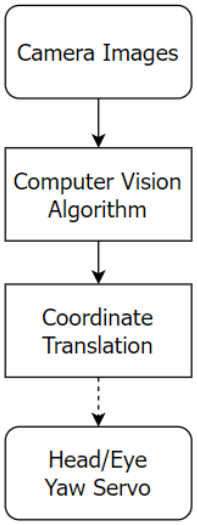

Figure 4: Computer vision system flowchart. Dotted lines represent ROS connections.

The computer vision system runs on an NVIDIA Jetson Nano, a microcontroller with an embedded graphical processing unit (GPU) that allows for the simultaneous filtering of multiple images. The speed of the processor reduces the delay between the user's motion and the animatronic figure's reaction.

Speech Recognition System

The MIMIR animatronic uses a speech recognition system to listen to the user's voice. When the user speaks a specific key phrase, the animatronic figure will respond to the user's prompt by moving and speaking. A microphone is embedded in the ornate bowl, allowing the user to speak while moving about the room. This microphone is connected to a receiver that will transmit the speech data to the speech recognition system. The MIMIR animatronic figure will contain pre-recorded dialogue that corresponds to the subject of the user's key phrase. The MIMIR animatronic contains fourteen unique dialogue outputs triggered by fourteen unique key phrases.

The speech recognition system uses the PocketSphinx speech recognition model developed by Carnegie Mellon University. This model functions offline and has been trained on an internally developed dictionary by CMU. The speech recognition system searches user data for specific key phrases. Each key phrase corresponds to an index number, according to table 2.

Table 2: Key phrases that will trigger the MIMIR animatronic figure to speak and move.

Key Phrase	**Index**
“greetings stranger”	0
“how may I address you”	1
“well of wisdom”	2
“why are we under a tree”	3
“speak about the gods”	4
“where is your body”	5
“tell me about your eyes”	6
“suitable offering”	7
“glistening gold”	8
“enchanted wood”	9
“apples of immortality”	10
“sands of time”	11
“senior design project”	12
“until next time”	13

When the user speaks the phrase, it will trigger the corresponding .mp3 dialogue file to play, as well as the corresponding servo movements. The available servo movements for the speech recognition system are mentioned in table 1. Figure 5 depicts a flowchart for the speech recognition system.

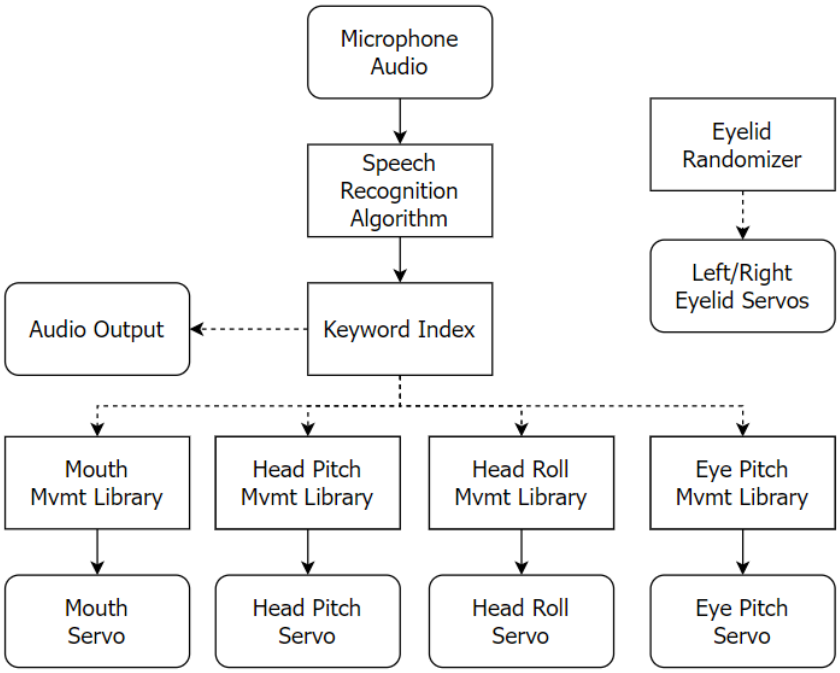

Figure 5: Speech recognition system flowchart, including eyelid randomizer. Dotted lines represent ROS communication.

The Raspberry Pi microcontroller will control all servo movements using two servo hats connected to the PWM signal pins on the microcontroller. In addition, the Raspberry Pi also houses a function that will randomly generate an eyelid actuation. A random number function will publish the value of a time delay between each eyelid movement.

Safety System Overview

The MIMIR animatronic figure was designed according to ASTM F2291-21: Standard Practice for Design of Amusement Rides and Devices [8]. This standard governs the design of themed entertainment attractions. In accordance with these standards, a five-foot barrier will be placed between the user and the animatronic figure. This barrier is commonly referred to as the system's reach envelope. Because of this requirement, a tethered operator control panel has been designed to allow the operator to stay outside of the reach envelope when powering and

monitoring the animatronic figure. This operator control panel contains an emergency stop pushbutton that will allow the operator to immediately remove power from the servo motors and shutoff the microcontrollers in case of an emergency.

System Integration

The Robot Operating System (ROS) is used to send and receive data from the computer vision and speech recognition systems. The ROS environment contains two major functions to allow this communication: publishers and subscribers. A publisher is a script that can transmit data. A subscriber can access published information and perform a corresponding action using the received data. The communication pipeline between the publisher and subscriber is called a topic. Multiple subscribers can subscribe to topics. The publisher and subscriber information, along with the affected servos and message types can be seen in table 3. The "comp_vision_angle" topic handles the information from the computer vision system. The "random_num" topic controls the actuation of the eyelids. The "mimir_keyword" topic handles the information from the speech recognition system.

Table 3: List of topics, publishers, and subscribers for the MIMIR animatronic figure.

Topic Name	Node Type	Node Name	Message Type	Servos Used
"comp_vision_angle"	Publisher	"comp_vision"	Int16	
	Subscriber	"servo_yaw"		Head Yaw, Eye Yaw
"random_num"	Publisher	"eyelid_randomizer"		
	Subscriber	"servo_eyelids"		Left Eyelid, Right Eyelid
"mimir_keyword"	Publisher	"speech_recog"		
	Subscriber	"dialogue_mp3"		
	Subscriber	"servo_head_roll"		Head Roll
	Subscriber	"servo_head_pitch"		Head Pitch
	Subscriber	"servo_eye_pitch"		Eye Pitch
	Subscriber	"servo_mouth"		Mouth

CHAPTER II

LED ENHANCEMENTS

Introduction

To add to the immersive capabilities of the MIMIR animatronic figure, LED eyes are added to the system. These LED eyes can change colors to match the dialogue of the MIMIR animatronic. These LED eyes will consist of four different modes, allowing the operator to alter the parameters by which the LED eyes will change colors.

Hardware

LED Hardware

The LED eyes use NeoPixel Rings. These lights contain 12 RGB LEDs, which can be individually addressed. These lights were chosen due to their small, circular shape. Adafruit, the company who developed the NeoPixel family of LED lights, also created a library for controlling their LEDs.

Microcontroller Analysis

To control the LED lights, an Arduino Uno is used. The Arduino microcontroller is both inexpensive and compatible with the NeoPixel LED library. When designing the power distribution system for the MIMIR animatronic figure, the power requirements for the Arduino were taken into consideration. According to the Arduino datasheet, a fuse in the USB port

connected to a computer will blow if the Arduino's current exceeds 500mA [9]. The power load was calculated to be 2.5W. Figure 6 depicts the wiring diagram for the LED eye system. The LEDs are connected to pin 6 for communication, while the pushbutton is attached at pin 2.

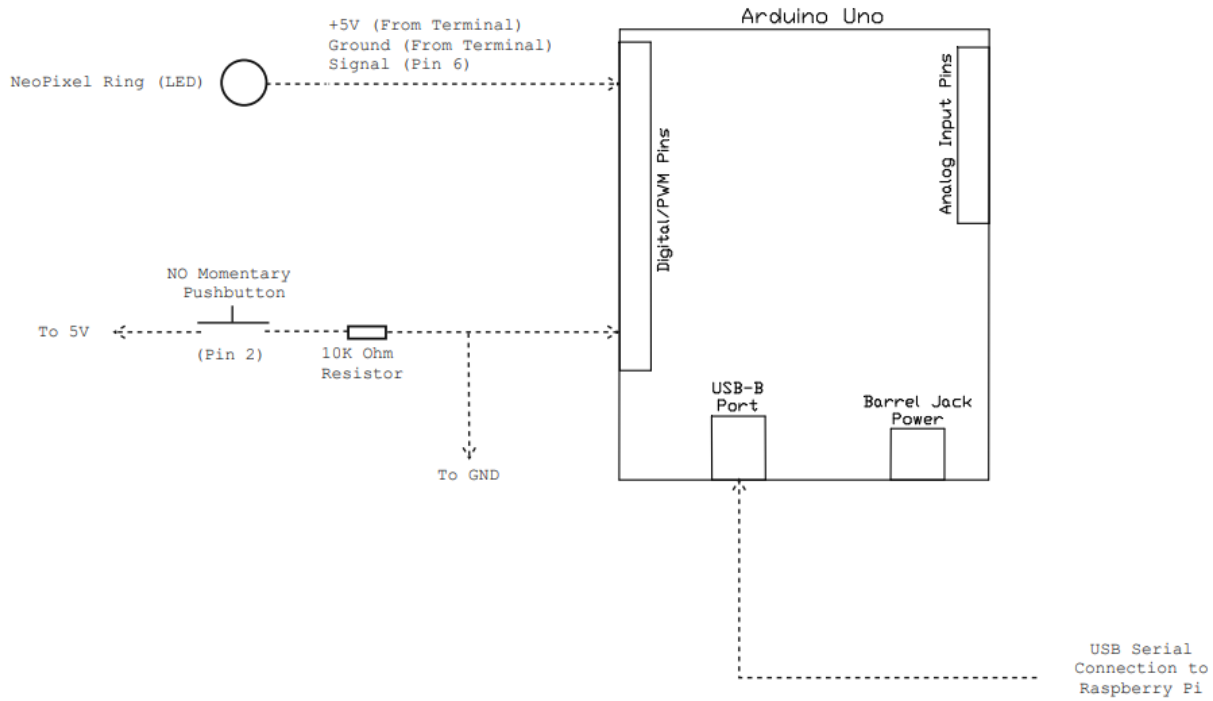

Figure 6: Wiring diagram for LED eye system.

LED Placement

The LED lights will be placed behind the 3D printed pupils. The eyes were printed with a transparent ABS filament, allowing for the light to shine through the eyes. A benefit of using transparent ABS is that the final print appears opaque. This allows the light to shine through while obscuring the user's view of the LED mechanism. This allows the light to appear to emanate from a point light source.

Changing Modes

For the user to switch between LED eye modes, a momentary pushbutton will be placed on the operator control panel. The user will be able to cycle between the four different eye modes. Once the user has decided upon the eye mode for the MIMIR animatronic, they will hold the pushbutton for 1 second to initiate the mode. To change eye modes again, the MIMIR animatronic figure would need to be restarted.

Software

Custom Color Library

The lights can be programmed using an Adafruit NeoPixel library. The library offers a method of programming the LED color using an HSV methodology. HSV, is a method of controlling multiple aspects of the LED's parameters, including the color's hue, saturation, and brightness. This offers a robustness when programming the LEDs to match the MIMIR animatronic's dialogue.

In the HSV function, the hue variable has a range of 0 to 65536. Each value represents a point on the color spectrum, moving from red to yellow to green to blue to purple. The saturation variable represents the purity of the hue. The range is from 0 to 255, moving from grayscale to pure. The brightness variable can be altered from 0 to 255, moving from off to complete brightness.

Because the LED eyes needed to shift colors during the dialogue enhancement mode, a function was written to transition between two colors' HSV values. The function takes in seven parameters. These parameters include the old HSV value, the new HSV value, and a time delay to govern the speed of the transition. The code for the program can be seen below:

```
void HSV(int hue, int sat, int bright, int hue_new, int sat_new, int
bright_new, int time_delay)
{
  while(hue != hue_new || sat != sat_new || bright != bright_new) //While the
old value is not equivalent
  {
    if (hue < hue_new) //If the new hue is greater, increment
    {
      hue++;
    }
    else if (hue > hue_new) //else, decrement
    {
      hue--;
    }
    if (sat < sat_new) //If the new saturation is greater, increment
    {
      sat++;
    }
    else if (sat > sat_new) //else, decrement
    {
      sat--;
    }
    if (bright < bright_new) //If the new brightness is greater, increment
    {
      bright++;
    }
    else if (bright > bright_new) //else, decrement
    {
      bright--;

    }
    int color = hue*(65536/360); //Color is the actual hue value, from 0-
65536.
    eye.fill(eye.ColorHSV(color,sat,bright),0,12);
    eye.show();
    delay(time_delay);
  }
}
```

LED Eye Modes

The MIMIR animatronic figure has four distinct eye modes. The first eye mode is called the “dialogue enhancement mode.” In this mode, the eyes change color to reflect the MIMIR animatronic figure’s dialogue. The eyes are pre-programmed and timed to the pre-recorded

dialogue. When the MIMIR animatronic figure is not producing dialogue, the LED eyes turn white. The second eye mode is a “vision eye mode.” In this eye mode, the location of a user within the camera will determine the color of the eye. As the user sweeps across the view of the camera, the eye will change colors, from red to purple. The third LED eye mode is a “random color mode.” In this mode, the LED eyes will change colors each time the animatronic figure blinks. The fourth eye mode is a “rainbow mode.” In this mode, the LED eyes shift colors through the rainbow. This color shift exists for both when the MIMIR animatronic figure is delivering an output and when the figure is at rest. Figure 7 depicts the flowchart for the LED eye modes.

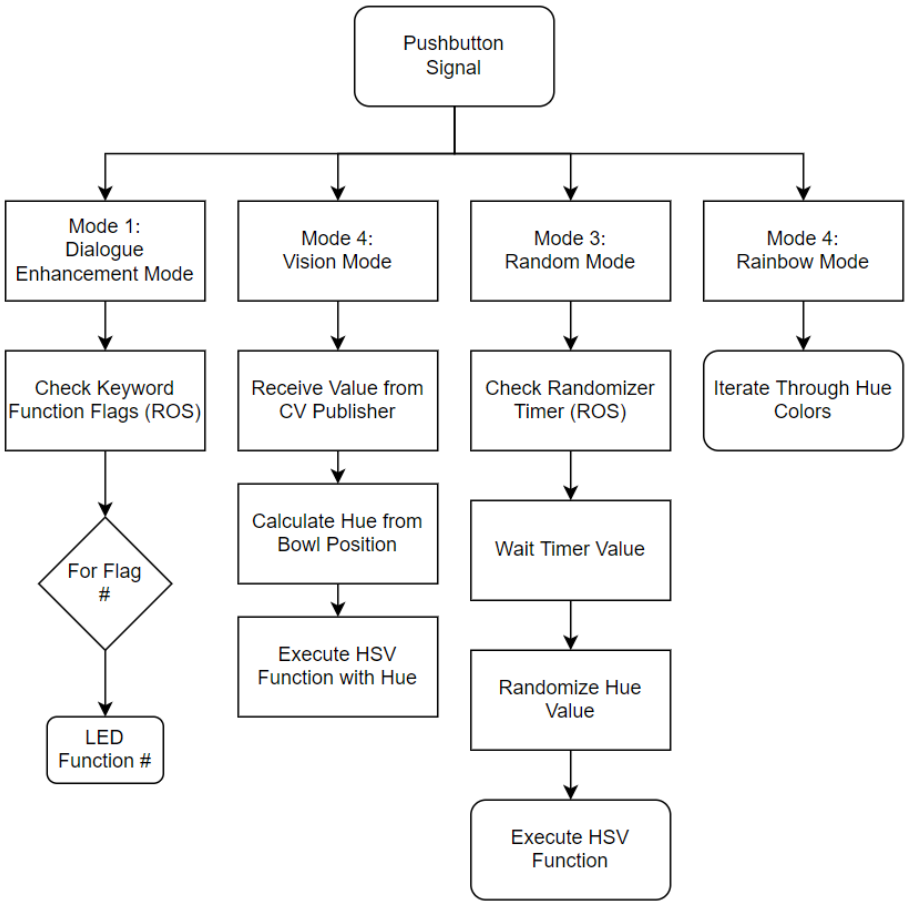

Figure 7: Flowchart for LED eye system.

Each of the eye modes is listed in table 4, along with when the eye mode is turned on.

Table 4: LED Eye modes and descriptions.

Eye Mode	Description	Timing
“dialogue eye mode”	The LED eyes change colors to match the dialogue output.	During dialogue only.
“rainbow eye mode”	The LED eyes iterate through the colors of the rainbow.	Always
“random color mode”	The LED eyes change colors each time the eyelids blink.	Always
“vision eye mode”	The LED eyes change colors based on the location of the user in the camera.	When object is detected.

ROS Additions

Because these eye modes require certain information from the computer vision and speech recognition systems, additional subscribers are added to ROS. Table 5 describes the subscriber additions.

Table 5: Subscriber additions for LED eye modes.

Eye Mode	Subscribes To	Node Name	Message Type	System
“dialogue eye mode”	“speech_recog”	“dialogue_eye”	Int16	Speech Recognition
“rainbow eye mode”				
“random color mode”	“rand_num”	“rand_eye”	Int16	Eyelid Randomizer
“vision eye mode”	“cv_angle”	“vision_eye”	Int16	Computer Vision

Future Considerations

The purpose of the LED addition was to provide an enhanced experience for users of the MIMIR animatronic figure. In the near future, a user survey will be generated to measure the experience that user’s have with the MIMIR animatronic. Questions on the efficacy of the LED eyes will be added to the survey to provide a concrete analysis of the system.

One of the major challenges with programming the LED eye system was the coupling of the speech recognition and computer vision systems. The programming of the LED modes was predicated on the implementation of the vision and speech systems, which delayed progress. Once the MIMIR animatronic figure is complete, further testing can be done to tune the LED parameters to provide a more immersive experience for users.

Conclusion

The LED enhancements to the MIMIR animatronic is a culmination of the programming knowledge and engineering experience gained over the last five years. The stated goal of the LED enhancements was to provide a more immersive experience from users. The system successfully contains four separate modes that communicate with the major systems of the MIMIR animatronic. This success is evidence that a future of bi-directional, storytelling animatronic figures is possible.

APPENDIX A

CODE

```
/*
LED Enhancements to the MIMIR Animatronic Figure
By Collin DeVillier
4.4.2022
*/

#include <ros.h>
#include <std_msgs/Int16.h>
#include <Adafruit_NeoPixel.h>

#define LED_PIN 6 //Arduino LED Pin
#define PIXELS 12 //Number of pixels in NeoPixel Ring
#define PUSHBUTTON_PIN 2 //Arduino Pushbutton Pin

Adafruit_NeoPixel eye(PIXELS,LED_PIN,NEO_GRB + NEO_KHZ800); //Creates class of
LED called eye

ros::NodeHandle  nh;

boolean toggle = LOW; //Toggle button for debounce
boolean select = LOW; //Select mode for when pushbutton is held down
int eyeMode = 0; //Holds the value for the desired eye mode
int timer = 0; //Holds the number of milliseconds that the pushbutton is held
int old_color = 0; //Holds the value of the previous color for vision eye mode

int dialogue_flag = 0; //Flag for dialogue eye mode
int vision_flag = 0; //Flag for vision eye mode
int random_flag = 0; //Flag for random eye mode


//Code for dialogue eye mode callback function
void dialogue_color( const std_msgs::Int16& dialogue_msg)
{
  if ((dialogue_flag == 1) && (eyeMode == 1)); //If the eye mode is set to 1
and the dialogue flag is initiated
  {
    int dialogue_var = dialogue_msg.data; //Creates variable that holds
keyphrase index value
    if (dialogue_var == 0)
    {
      keyphrase0();
```

```
    }
    if (dialogue_var == 1)
    {
      keyphrase1();
    }
    if (dialogue_var == 2)
    {
      keyphrase2();
    }
    if (dialogue_var == 3)
    {
      keyphrase3();
    }
    if (dialogue_var == 4)
    {
      keyphrase4();
    }
    if (dialogue_var == 5)
    {
      keyphrase5();
    }
    if (dialogue_var == 6)
    {
      keyphrase6();
    }
    if (dialogue_var == 7)
    {
      keyphrase7();
    }
    if (dialogue_var == 8)
    {
      keyphrase8();
    }
    if (dialogue_var == 9)
    {
      keyphrase9();
    }
    if (dialogue_var == 10)
    {
      keyphrase10();
    }
    if (dialogue_var == 11)
    {
      keyphrase11();
    }
    if (dialogue_var == 12)
    {
      keyphrase12();
    }
```

```
    if (dialogue_var == 13)
    {
      keyphrase13();
    }
  }
}


//Code for vision eye mode callback function
void spectrum_color(const std_msgs::Int16& angle)
{
  if ((vision_flag == 1) && (eyeMode == 2)) //If eye mode set to two and
vision flag initiated
  {
    int new_color = (angle.data - 20)*(18/7); //Sets value from 0-360
    HSV(old_color,255,255,new_color,255,255,1); //Sets to new color from
previous color
    old_color = new_color; //Update old_color flag to equal to the current
color
  }
}


//Code for random eye mode callback function
void random_color(const std_msgs::Int16& time_delay)
{
  if ((random_flag == 1) && (eyeMode == 3)) //If eye mode equal to three and
random flag initiated
  {
    int sleep = time_delay.data;

    int spectrum[] = {0,30,60,90,120,150,180,210,240,270,300,330}; //Sets
values of different colors

    int index = random(0,12); //Chooses random number from 0 to 11
    HSV(spectrum[index],255,0,spectrum[index],255,255,1); //Random number
chooses index for color
    delay(sleep); //Sleeps for the published value
  }
}

ros::Subscriber<std_msgs::Int16> dialogue_eye("mimir_keyword",
&dialogue_color); //Sets dialogue_eye subscriber
ros::Subscriber<std_msgs::Int16> vision_eye("comp_vision_angle",
&spectrum_color); //Sets vision_eye subscriber
ros::Subscriber<std_msgs::Int16> random_eye("random_num", &random_color);
//Sets random_eye subscriber

void setup()
```

```
{
  eye.begin(); //Initate LED eyes
  pinMode(PUSHBUTTON_PIN, INPUT_PULLUP); //Pushbutton pin initalization


  /*EYE MODE SELECTION
  The operator will press the pushbutton to cycle through the available eye
modes:
  RED -- DIALOGUE EYE MODE
  YELLOW -- VISION EYE MODE
  WHITE -- RANDOM EYE MODE
  BLUE -- RAINBOW EYE MODE

  To select an eye mode, the pushbutton should be held for 1 second.
  Once the selection has been made, the LEDs will flash three times in the
  corresponding mode color before starting the sequence.

  For the rainbow mode, the sequence will start immediately, for the other
modes,
  the corresponding subscriber will be initialized.*/
  while (select == LOW) //While pushbutton hasn't been held for 1 second
  {
    if ((digitalRead(PUSHBUTTON_PIN) == HIGH) && (toggle == LOW)) //if button
was pressed, but previous state was off...
    {
      delay(100); //debounce
      if (digitalRead(PUSHBUTTON_PIN) == HIGH) //if pushbutton is still high
      {
        eyeMode++; //increment eyeMode
        if (eyeMode == 1)
        {
          HSV(0,255,0,0,255,255,1); //turn eyes from off to red -- Dialogue
Eye Mode
        }
        if (eyeMode == 2)
        {
          HSV(0,255,255,60,255,255,1); //turn eyes from red to yellow --
Rainbow Mode
        }
        if (eyeMode == 3)
        {
          HSV(60,255,255,180,255,255,1); //turn eyes from yellow to white --
Random Mode
        }
        if (eyeMode == 4)
        {
          HSV(180,255,255,240,255,255,1); //turn eyes from white to blue --
Vision Mode
        }
```

```
if (eyeMode == 5) //Reset eyeMode
{
  eyeMode = 0;
  HSV(240,255,255,0,255,0,1); //turn eyes from blue to off
}
while(digitalRead(PUSHBUTTON_PIN) == HIGH)
{
  timer++; //increment timer
  delay(1); //count every millisecond
  if (timer >= 1000) //after 1 second
  {
    if (eyeMode == 1)
    {
      for (int i = 0; i <= 3; i++) //blink red three times
      {
        HSV(0,255,255,0,255,255,1);
        delay(250);
        HSV(0,255,255,0,255,0,1);
      }
      dialogue_flag = 1; //turn dialogue mode flag on
    }
    if (eyeMode == 2)
    {
      for (int i = 0; i <= 3; i++) //blink yellow three times
      {
        HSV(60,255,255,60,255,255,1);
        delay(250);
        HSV(60,255,255,60,255,0,1);
      }
      vision_flag = 1; //turn vision mode flag on
    }
    if (eyeMode == 3)
    {
      for (int i = 0; i <= 3; i++) //blink white three times
      {
        HSV(180,255,255,180,255,255,1);
        delay(250);
        HSV(180,255,255,180,255,0,1);
      }
      random_flag = 1; //turn random mode flag on
    }
    if (eyeMode == 4)
    {
      for (int i = 0; i <= 3; i++) //blink blue three times
      {
        HSV(240,255,255,240,255,255,1);
        delay(250);
        HSV(240,255,255,240,255,0,1);
      }
```

```
            }
            select = HIGH; //If pushbutton held for 1 second, turn select on,
exiting loop
          }
        }
        timer = 0; //If button held for less than 1 second, reset timer
        toggle = HIGH; //When button released, turn toggle on.
      }
      if ((digitalRead(PUSHBUTTON_PIN) == LOW) && (toggle == HIGH)) //If
toggle on and button released
      {
        toggle = LOW; //Turn toggle off
      }
    }
  }
  nh.initNode(); //Initializes ROS Serial node for Arduino. Occurs once.

  if (dialogue_flag == 1) //If dialogue mode chosen, turn subscriber on.
  {
    nh.subscribe(dialogue_eye);
  }
  else if (vision_flag == 1) //If vision mode on, turn subscriber on.
  {
    nh.subscribe(vision_eye);
  }
  else if (random_flag == 1) //If random mode on, turn subscriber on.
  {
    nh.subscribe(random_eye);
  }

}

void loop()
{
  nh.spinOnce();
  delay(1);
  while (eyeMode == 4) //If eye mode is set to rainbow mode, play sequence
  {
    rainbowEye();
  }
}

/*rainbowEye loops through the colors of the Rainbow using HSV function.
The delay can change the speed at which the colors loop.*/
void rainbowEye()
{
  for(int i=0; i<3276; i++)
  {
    int color = i*20;
```

```
    eye.fill(eye.ColorHSV(color,255,255));
    eye.show();
    delay(3);
  }
}


/*HSV is a function that takes in arguments for two separate hues,
saturations, and
brightnesses. The first three variables are the current HSV values. The second
three
variables are the new HSV values. The time delay sets how fast the transition
will be.
Hue has a range from 0 - 360 (the position on the color wheel.)
Saturation has a range from 0 - 255.
Brightness has a range from 0 - 255.*/

void HSV(int hue, int sat, int bright, int hue_new, int sat_new, int
bright_new, int time_delay)
{
  while(hue != hue_new || sat != sat_new || bright != bright_new) //While the
old value is not equivalent
  {
    if (hue < hue_new) //If the new hue is greater, increment
    {
      hue++;
    }
    else if (hue > hue_new) //else, decrement
    {
      hue--;
    }
    if (sat < sat_new) //If the new saturation is greater, increment
    {
      sat++;
    }
    else if (sat > sat_new) //else, decrement
    {
      sat--;
    }
    if (bright < bright_new) //If the new brightness is greater, increment
    {
      bright++;
    }
    else if (bright > bright_new) //else, decrement
    {
      bright--;

    }
```

```
    int color = hue*(65536/360); //Color is the actual hue value, from 0-
65536.
    eye.fill(eye.ColorHSV(color,sat,bright),0,12);
    eye.show();
    delay(time_delay);
  }
}

/*RGB is a similar function that takes in arguments for an old and new color
value, along with
a time delay. This function was never called, but it was written in case
certain color changes
were needed for the dialogue mode.*/

void RGB(int red, int green, int blue, int red_new, int green_new, int
blue_new, int time_delay)
{
  while(red != red_new || green != green_new || blue != blue_new)
  {
    if (red < red_new)
    {
      red++;
    }
    else if (red > red_new)
    {
      red--;
    }
    if (green < green_new)
    {
      green++;
    }
    else if (green > green_new)
    {
      green--;
    }
    if (blue < blue_new)
    {
      blue++;
    }
    else if (blue > blue_new)
    {
      blue--;

    }
    eye.fill(eye.Color(red,green,blue),0,12);
    eye.show();
    delay(time_delay);
  }
}
```

```
/*DIALOGUE MODE KEYPHRASES
 * The below keyphrase functions house the color changes with the
corresponding dialogue.*/

void keyphrase0()
{
  delay(800);
  HSV(240,0,50,240,50,255,1); //white, increasing brightness
  HSV(240,0,255,240,255,255,7); //white to blue, increasing saturation
  HSV(240,255,255,360,255,255,9); //blue to red, increasing hue
  delay(1600);
  HSV(360,255,255,360,255,50,2); //decreasing brightness
  HSV(360,255,50,360,0,50,6); //decreasing saturation
}

void keyphrase1()
{
  delay(200);
  HSV(0,0,50,0,0,255,2); //white, increaing brightness
  HSV(0,0,255,0,255,255,3); //white to red, increasing saturation
  delay(3300);
  HSV(0,255,255,120,255,255,5); //white to green, increasing hue
  delay(700);
  HSV(120,255,255,60,255,255,5); //green to yellow, decreasing hue
  delay(4000);
  HSV(60,255,255,0,255,255,15); //orange to red, decreasing hue
  delay(4000);
  HSV(360,255,255,320,255,255,10); //red to purple, decreasing hue
  delay(1700);
  HSV(320,255,255,240,255,255,5); //purple to blue, decreasing hue
  delay(2700);
  HSV(240,255,255,360,255,255,2); //blue to red, increasing hue
  delay(2800);
  HSV(0,255,255,120,255,255,9); //red to green, increasing hue
  delay(1000);
  HSV(120,255,255,140,255,255,7); //green to cyan, increasing hue
  delay(1700);
  HSV(140,255,255,140,255,50,2); //decreasing brightness
  HSV(140,255,50,140,0,50,2); //decreasing saturation
}

void keyphrase2()
{
  delay(1000);
  HSV(330,0,50,330,0,255,2); //white, increasing brightness
  delay(800);
  HSV(330,0,255,330,255,255,7); //white to pink, increasing saturation
  delay(300);
```

```
  HSV(330,255,255,330,0,255,4); //pink to white, decreasing saturation
  delay(500);
  HSV(0,0,255,0,255,255,4); //white to red, increasing saturation
  HSV(0,255,255,120,255,255,20); //red to green, increasing hue
  HSV(120,255,255,120,0,255,7); //green to white, decreasing saturation
  delay(800);
  HSV(60,0,255,60,255,255,12); //white to yellow, increasing saturation
  delay(3000);
  HSV(60,255,255,60,0,255,7); //yellow to white, decreasing saturation
  delay(1200);
  HSV(220,0,255,220,255,255,14); //white to blue, increasing saturation
  delay(4200);
  HSV(220,255,255,360,255,255,17); //blue to red, increasing hue
  delay(1300);
  HSV(0,255,255,0,255,50,2); //decreasing brightness
  HSV(0,255,50,0,0,50,2); //decreasing saturation
}

void keyphrase3()
{
  delay(1000);
  HSV(35,0,50,35,0,255,2); //white, increasing brightness
  delay(3600);
  HSV(35,0,255,35,255,255,3); //yellow-orange, increasing saturation
  delay(800);
  HSV(35,255,255,35,0,255,2); //decreasing saturation
  delay(700);
  HSV(240,0,255,240,255,255,3); //blue, increasing saturation
  delay(700);
  HSV(240,255,255,240,0,255,2); //decreasing saturation
  delay(800);
  HSV(0,0,255,0,255,255,4); //red, increasing saturation
  delay(2900);
  HSV(0,255,255,0,0,255,2); //decreasing saturation
  delay(1100);
  HSV(330,0,255,330,255,255,2); //purple-red, increasing saturation
  delay(1500);
  HSV(330,255,255,330,0,255,2); //decreasing saturation
  delay(3000);
  HSV(120,0,255,120,255,255,3); //green, increasing saturation
  delay(1000);
  HSV(120,255,255,120,255,50,2); //decreasing brightness
  HSV(120,255,50,120,0,50,2); //decreasing saturation
}

void keyphrase4()
{
  delay(500);
  HSV(330,0,50,330,0,255,2); //increasing brightness
```

```
  delay(600);
  HSV(330,0,255,330,255,255,2); //purple, increasing saturation
  delay(2400);
  HSV(330,255,255,330,0,255,2); //decreasing saturation
  delay(400);
  HSV(60,0,255,60,255,255,2); //yellow, increasing saturation
  delay(2400);
  HSV(60,255,255,60,0,255,2); //decreasing saturation
  delay(8000);
  HSV(0,0,255,0,255,255,3); //red, increasing saturation
  delay(2000);
  HSV(0,255,255,0,0,255,2); //decreasing saturation
  delay(5500);
  HSV(120,0,255,120,255,255,3); //green, increasing saturation
  delay(6500);
  HSV(120,255,255,0,255,255,5); //green to red, decreasing hue
  delay(2500);
  HSV(0,255,255,0,255,50,2); //decreasing brightness
  HSV(0,255,50,0,0,50,2); //decreasing saturation
}

void keyphrase5()
{
  delay(500);
  HSV(230,0,50,230,0,255,2); //increasing brightness
  delay(1000);
  HSV(230,0,255,230,255,255,5); //blue, increasing saturation

  for (int i = 0; i <= 2; i++) //pulsing blue color
  {
    HSV(230,255,255,230,255,100,13); //decreasing brightness
    HSV(230,255,100,230,255,255,13); //increasing brightness
  }
  delay(200);
  HSV(230,255,255,120,255,255,7); //green, decreasing hue
  delay(1000);
  HSV(120,255,255,120,255,50,2); //decreasing brightness
  HSV(120,255,50,120,0,50,2); //decreasing saturation
}

void keyphrase6()
{
  delay(500);
  HSV(180,0,50,180,0,255,2); //increasing brightness
  HSV(180,0,255,180,255,255,2); //white, increasing saturation
  delay(1400);
  for(int i = 0; i <= 3; i++) //rainbow for two iterations
  {
    for(int j = 0; j<=360; j++)
```

```
    {
      int color = j*(65536/360);
      eye.fill(eye.ColorHSV(color,255,255));
      eye.show();
      delay(2);
    }
  }
  delay(100);
  HSV(0,255,255,0,0,255,2); //decreasing saturation
  delay(6000);
  HSV(120,0,255,120,255,255,4); //green, increasing saturation
  delay(3000);
  HSV(120,255,255,280,255,255,12);//green to blue
  delay(2500);
  HSV(300,255,255,360,255,255,6); //blue to red
  for (int i = 0; i <= 21845; i++)
  {
    int color = 3*i;
    eye.fill(eye.ColorHSV(color,255,255));
    eye.show();
  }
  HSV(360,255,255,360,255,50,2); //decreasing brightness
  HSV(360,255,50,360,0,50,2); //decreasing saturation
}

void keyphrase7()
{
  delay(500);
  HSV(0,0,50,0,0,255,2); //increasing brightness
  delay(6400);
  HSV(35,0,255,35,255,255,2); //yellow, increasing saturation
  delay(1000);
  HSV(35,255,255,35,0,255,2); //decreasing saturation
  HSV(0,0,255,0,255,255,2); //red, increasing saturation
  delay(1100);
  HSV(0,255,255,0,0,255,2); //decreasing saturation
  HSV(120,0,255,120,255,255,2); //green, increasing saturation
  delay(1200);
  HSV(120,255,255,120,0,255,2); //decreasing saturation
  HSV(340,0,255,340,255,255,2); //purple, increasing saturation
  delay(1500);
  HSV(340,255,255,340,255,50,2); //decreasing brightness
  HSV(340,255,50,340,0,50,2); //decreasing saturation
}

void keyphrase8()
{
  delay(500);
  HSV(35,0,50,35,0,255,2); //increasing brightness
```

```
  delay(1200);
  HSV(35,0,255,35,255,255,2); //yellow, increasing saturation
  delay(16000);
  HSV(35,255,255,35,255,50,2); //decreasing brightness
  HSV(35,255,50,35,0,50,2); //decreasing saturation
}

void keyphrase9()
{
  delay(500);
  HSV(120,0,50,120,0,255,2); //increasing brightness
  delay(1200);
  HSV(120,0,255,120,255,255,2); //green, increasing saturation
  delay(16500);
  HSV(120,255,255,120,255,50,2); //decreasing brightness
  HSV(120,255,50,120,0,50,2); //decreasing saturation
}

void keyphrase10()
{
  delay(500);
  HSV(0,0,50,0,0,255,2); //increasing brightness
  delay(1000);
  HSV(0,0,255,0,255,255,2); //red, increasing saturation
  delay(18500);
  HSV(0,255,255,0,255,50,2); //decreasing brightness
  HSV(0,255,50,0,0,50,2); //decreasing saturation
}

void keyphrase11()
{
  delay(500);
  HSV(340,0,50,340,0,255,2); //increasing brightness
  delay(1000);
  HSV(340,0,255,340,255,255,2); //red, increasing saturation
  delay(12000);
  HSV(340,255,255,340,255,50,2); //decreasing brightness
  HSV(340,255,50,340,0,50,2); //decreasing saturation
}

void keyphrase12()
{
  delay(500);
  HSV(120,0,50,120,0,255,2); //increasing brightness
  delay(2200);
  HSV(120,0,255,120,255,255,2); //green, increasing saturation
  delay(1000);
  HSV(120,255,255,120,0,255,2); //decreasing saturation
  delay(3600);
```

```
  HSV(350,0,255,350,255,255,2); //pink, increasing saturation
  delay(1000);
  HSV(350,255,255,350,0,255,2); //decreasing saturation
  delay(2500);
  HSV(30,0,255,30,255,255,2); //yellow, increasing saturation
  delay(1000);
  HSV(30,255,255,30,0,255,2); //decreasing saturation
  delay(2000);
  HSV(220,0,255,220,255,255,2); //blue, increasing saturation
  delay(1200);
  HSV(220,255,255,220,0,255,2); //decreasing saturation
  delay(2500);
  HSV(300,0,255,300,255,255,2); //purple, increasing saturation
  delay(2000);
  HSV(300,255,255,300,0,255,2); //decreasing saturation
  delay(2700);
  HSV(0,0,255,0,255,255,2); //red, increasing saturation
  delay(2500);
  HSV(0,255,255,0,0,255,2); //decreasing saturation
  delay(2000);
  HSV(0,0,255,0,0,50,2); //decreasing brightness
}

void keyphrase13()
{
  delay(500);
  HSV(37,0,50,37,0,255,2); //increasing brightness
  delay(1000);
  HSV(120,0,255,120,255,255,2); //green, increasing saturation
  delay(1500);
  HSV(120,255,255,120,0,255,2); //decreasing saturation
  delay(800);
  HSV(37,0,255,37,255,255,3); //yellow, increasing saturation
  delay(2200);
  HSV(37,255,255,37,255,50,2); //decreasing brightness
  HSV(37,255,50,37,0,50,2); //decreasing saturation
}
```

APPENDIX B

MIMIR DIALOGUE

Keyphrase 0: "greetings stranger"

And hello to you weary traveler. What wisdom have you come to seek?

Keyphrase 1: "how may i address you"

What is my name? Some call me the God of Wisdom. I am keeper of knowledge and guardian of the secrets of the nine realms. I know all and I see all. I am the custodian of memory and broker between dreams and reality. What is my name? I am MIMIR! What other knowledge do you seek?

Keyphrase 2: "well of wisdom"

Before me lies a well that contains limitless wisdom. I drink from this well daily in order to maintain access to the world's knowledge. This well also provides nourishment to the world tree, Yggdrasil.

Keyword 3: "why are we under a tree"

Ah yes, your memory fades, doesn't it? You have traveled a great distance to stand beneath the great ash tree, Yggdrasil. It is this tree, with its cathedral of glowing roots that gives life to the nine realms. Before me lies one of three wells that provides nourishment to the great tree. It is I who guards this well of infinite wisdom. Any more questions?

Keyword 4: "speak about the gods"

Well, there are two races of gods. The Aesir are the gods of Asgard. The Vanir are the gods of Vanaheim. These gods often clash, waging wars against one another. After one such skirmish, Hoesir, an old Aesir god, was placed in charge of the Vanir in hopes of brokering a peace.

Keyword 5: "where is your body"

Hoesir, an Aesir god, was placed in charge of the Vanir. I was appointed as counsel to Hoesir. Over time, the Vanir grew tired of Hoesir's tenure as ruler of the Vanir. As punishment, the Vanir thought fit to cut my head off. Odin, king of the Aesir, brewed a benevolent concoction of herbs and restored my consciousness. It is through his kindness that I speak to you today. Do you have any other questions?

Keyword 6: "tell me about your eyes"

Oh, you mean these beauties. (Pause as Mimir's eyes shimmer). Odin, king of the gods, also sought wisdom from my well. As guardian of the well, I struck a deal with Odin. In exchange for a taste of these divine waters, I requested that Odin bring me two orbs fashioned from the great rainbow bridge called the Bifrost. Having replaced my eyes with the orbs of the Bifrost, I was able to see into dimensions beyond our own.

Keyword 7: "suitable offering"

A choice lies before you. You may choose from four items: a bar of gold, an basket of apples, a bundle of wood, or a bowl of sand.

Keyword 8: "glistening gold"

You have chosen glistening gold from the ground below. I cannot accept this gift, for gold belongs to the dwarves of Svartalfheim. Their craftsmanship is unmatched in the nine realms.

Keyword 9: "enchanted wood"

You have chosen wood cut from the enchanted forests of Jotenheim. I cannot accept this gift, for these enchanted trees belong to the giants. The giants should not be trifled with.

Keyword 10: "apples of immortality"

You have chosen apples picked from the tree of immortality. I cannot accept this gift, for these apples belong to Idunn, goddess of rejuvenation. These apples provide immortality to the gods.

Keyword 11: "sands of time"

You have chosen the sands of time. I am most grateful for this gift, for you see, I am an old man and merely want to share my tales with travelers on their journey.

Keyword 12: "senior design project"

Ahem. My name is MIMIR, and I wanted to take a moment to praise the remarkable efforts of these five engineering students. In many ways their unparalleled creativity is matched only in their magnificent engineering ability. As the god of wisdom, I know all and I see all. And I see no greater feat of excellence than the handsome devil that speaks before you. That is all.

Keyword 13: "until next time"

And farewell to you kind seeker. May you find joy and peace in your journeys ahead.

REFERENCES

[1] merriam-webster.com, "theme park," Merriam-Webster, [Online]. Available: https://www.merriam-webster.com/dictionary/theme%20park. [Accessed 2022].

[2] T. Skees, "Themed Entertainment, the basics," The Designer's Creative Studio, 2 November 2020. [Online]. Available: https://www.designerscreativestudio.com/blog/Themed-Entertainment-the-basics. [Accessed 2022].

[3] D23, "Audio-Animatronics," D23, [Online]. Available: https://d23.com/a-to-z/audio-animatronics/. [Accessed 2022].

[4] D23, "The Birds, Beasts, and Beauty of Disney's Audio Animatronics Characters," D23, [Online]. Available: https://d23.com/audio-animatronics-disneyland-magic-kingdom-walt-disney-world/. [Accessed 2022].

[5] IEEE, "Na'vi Shaman," [Online]. Available: https://robots.ieee.org/robots/navishaman/. [Accessed 2022].

[6] M. Panzarino, "Disney has begun populating its parks with autonomous, personality-driven robots," TechCrunch, 8 February 2018. [Online]. Available: https://techcrunch.com/2018/02/08/disney-has-begun-populating-its-parks-with-autonomous-personality-driven-robots/. [Accessed 2022].

[7] *One Day at Disney (Shorts) - Alfredo Ayala: R&D Imagineer*. [Film]. United States: Walt Disney Studios, 2019.

[8] ASTM International, "Standard Practice for Design of Amusement Rides and Devices," [Online]. Available: https://www.astm.org/f2291-21.html. [Accessed 2022].

[9] Arduino, "Arduino Uno Rev3," [Online]. Available: http://store-usa.arduino.cc/products/arduino-uno-rev3. [Accessed 2022].