Design and Implementation of a Projection Seeking Robot

by

Morgan R. Stewart

Submitted to the
Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 2015

Signature of Author: Signature redacted

Department of Mechanical Engineering
May 22, 2015

Certified by: Signature redacted

David Trumper
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by: Signature redacted

Anette Hosoi
Professor of Mechanical Engineering
Undergraduate Officer

Design and Implementation of a Projection Seeking Robot

by

Morgan R. Stewart

Submitted to the Department of Mechanical Engineering
on May 8, 2015 in Partial Fulfillment of the
Requirements for the Degree of

Bachelor of Science in Mechanical Engineering

ABSTRACT

A design project on the development of an interactive robot system for display in the MIT Museum Studio gallery. The system consists of a moving projection surface, a projector, and an external camera with motion detection. When the external camera senses motion in the hallway outside of the gallery, it takes a picture. If a face is visible in the image, the image is cropped and displayed on the projector. The external controller notifies the moving surface that it should try to orient itself within a projection field. The projection surface uses its internal camera to continuous take images through the translucent projection material and determines whether it or not it needs to move to be within the projection field.

Through this series of interactions, the robot essentially "borrows" the faces of those who pass the studio, catching projected imagery of their faces on the "face" of the projection surface. With this interaction in mind, the projection surface was designed to mimic human qualities, with an abstract head, neck, and organic body. This thesis encompasses the design and documentation of the physical robot, as well as the design and programming of one series of interactions that could be performed by this robot.

This thesis represents a snapshot of the current progress of this project. Besides performing the interaction sequence described above, a secondary goal of this project is to a create an easily programmable and well documented project that future students of the MIT Museum Studio can modify for new displays and projects.

Thesis Supervisor: David Trumper
Tile: Professor of Mechanical Engineering

Table of Contents

List of Figures

List of Tables

1. Project Overview

This project includes the design, construction and programming of a robot for the MIT Museum Studio. The mobile robot carries a projection surface, and uses light sensing and commands from an external controller to capture light on its projection surface "head". The robot observes patterns of light and dark based on captured imagery and determines if adjustments to its position are necessary to better orient itself within the projected light field. The result is a moving screen that turns and dances based on the imagery it shares with the viewer.

The end product is a 2.5' robot that rolls on the floor in the front of the gallery. It moves around on four omnidirectional wheels. It uses sensors and cameras to determine whether or not it needs to move, and also responds to moment commands from an external controller. This robot is still under development, and an outline for next steps is in section 4. Updated control code can be found at: https://github.com/morganrstewart/museumbot.

The current system includes an assembled and mobile robot and a networking system between the controller and robot. The camera controller is able to detect motion, take and edit pictures, and send motion commands to the robot which moves in response. Next steps include developing a website that displays the images taken by the camera and refreshes so the projector does not have to be plugged in to the camera/controller assembly. Also, further refinement of the motion commands and environmental responses is in progress.

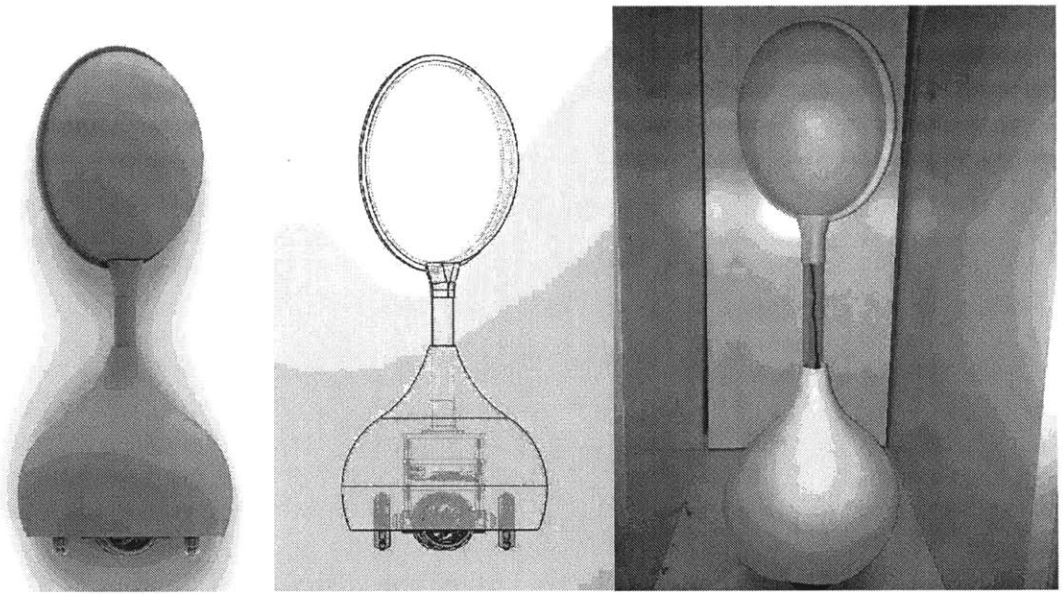


Figure 1-1: The assembled robot. The neck piece is interchangeable based on the height of the projector. In this models an 8" neck is used, while the photo shows a 12" neck. The oval dome on top is a matte acrylic projection surface, mounted with custom 3D printed parts to a polycarbonate neck. The neck connects to the electronics tower which is seated on the wheel base. Collectively, this assembly is

known as the robot. The external camera and computer are together known as the controller. Future work includes finishing and painting the entire assembly.

The robot was designed iteratively since there was an unclear set of design and performance constraints. With each prototype, questions were answered regarding how the robot should look, move, and interact with people and the environment. The final assembly uses primarily off the shelf parts except for the custom made connecting components and outer shell.

The main components of this project include mechanical, electrical, and software prototyping, packaging the robot for display, power management, and the development of controls to make sure the robot is safe and does not harm people or other exhibits in the gallery. It requires iteration of prototypes and testing to see how the robot functions within the space. This project focuses on hardware and software design of robotics for the museum gallery.

1.1 Developing Context: Robotics and the MIT Museum Studio

The MIT Museum Studio is a gallery, teaching space, and workshop that re-opened in fall 2014. The studio features a large glass-front gallery where student and professional art and engineering projects are shown. This project is a student and studio collaboration to create a piece that utilizes the unique location and structural aspects of the gallery more effectively than traditional museum-style exhibition.

The MIT Museum Studio has a history of artist-engineer collaborations involving imagery and robotics. The most recent example of one of these large scale collaborations is the continued development of artist Otto Piene's Light Ballet robots. These large shapes on wheels feature swirling lights within a 2' cube (or other shapes) with perforated walls. When they perform, the robots use steel and omni-directional wheel bases to twirl and glide, throwing light patterns on the walls. The design of the projection seeking robot is influenced by the Light Ballet robots.

Figure 1-2: Light Ballet robots at the Miller Yzerski Gallery on April 15, 2015 [1]

1.1.1 Mission of the MIT Museum Studio and Gallery

The goal of the MIT Museum Studio and Gallery is to provide a space for students, engineers, and artists to collaborate and create innovative projects that cross the boundaries between fields and extend into unexplored territory. "Display drives communication at the MIT Museum Studio. Materializing ideas and getting feedback enables students to see themselves and their work from multiple perspectives." [2]

This project fits the guiding principles of the studio as it combines the exploration of projected imagery and technical work while also experimenting with new methods of communication. This iteration of the robot relies on light and facial recognition to drive interactions, but the robot and controller are designed to be reconfigurable, a platform for other students to modify and evolve. This thesis aims to document the design and creation of the first robot platform.

1.1.2 MIT Museum Studio Layout

This robot is specifically designed to work in the MIT Museum Studio. Further work could include a more flexible calibration system that allows the robot to operate in less controlled conditions. The space features a large glassed in gallery with a poured concrete floor. The robot will roam the space between the gallery wall and the glass front.

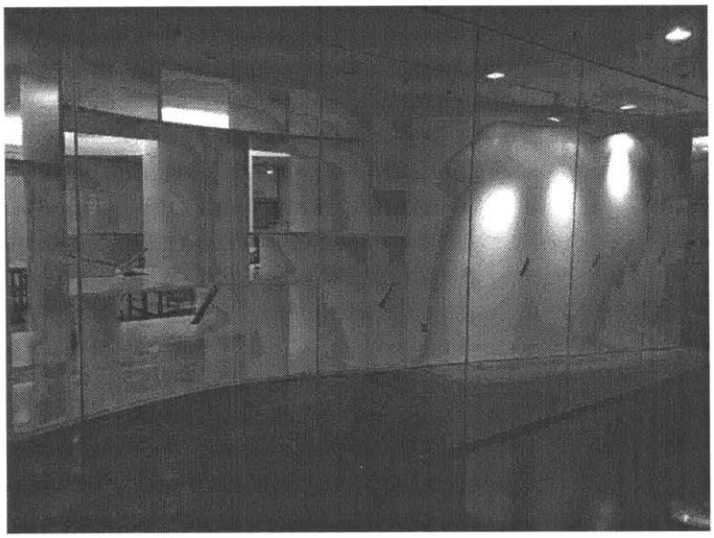


Figure 1-3: The MIT Museum Studio gallery space, room 10-150 at MIT. The robot will capture imagery through the glass and move on the floor.

2. Design of the Robot and Controller System

2.1 Overall System Layout

The physical robot must work with its external controller module and a projector in order to effectively "perform" for viewers. The controller is the most active part of the system. It is the only component that "sees" people and it drives the robot to react to the presence of a person.

The projector, the second component in the chain simply displays the image created by the controller. The robot has two layers of actions. The robot can stand in place until a signal is received, or can wander. When a signal is received from the controller, the robot follows the corresponding sequence of actions while checking for obstacles.

Table 2-1: Input sources for major system components.

	Controller	**Projector**	**Robot**
Input	Motion	Image	Command
Source	Motion Sensor	Controller	Controller (WIFI)
Secondary Input	Image		Distance
Source	Camera		Ultrasound Sensor

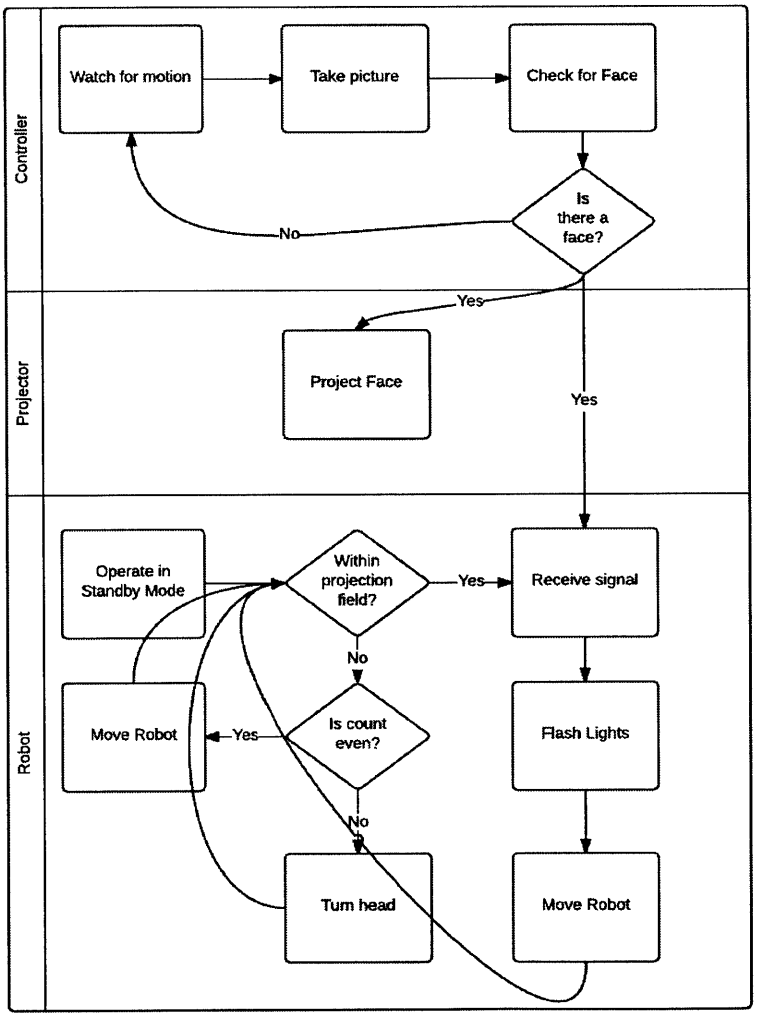


Figure 2-1: System overview chart

The controller box is a Raspberry Pi with Wi-Fi enabled, housed within a plastic project enclosure. For this iteration, a USB camera takes the pictures, though future iterations could include a higher quality camera. Also, the interaction experience for the user could be enhanced by creatively disguising the control box and using the robot, lights, or some other feature to lure people into the correct position. In the current state, the viewer must consciously choose to stand in front of the camera to get a clear picture. An ideal implementation would seamlessly integrate the controller camera into another feature of the gallery to make the experience more of a surprise and less like a photo booth.

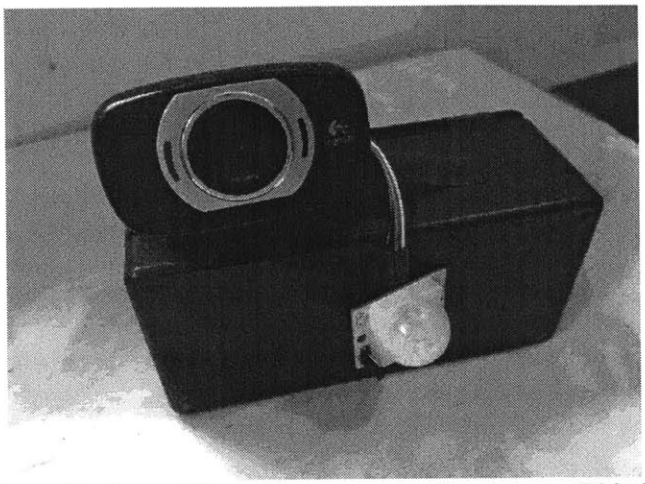


Figure 2-2: Controller box with motion sensor and webcam. This is placed along the glass wall to capture motion down the hallway. The sensor range is 30 feet.

2.2 Robot Body Design

The robot body was designed to have a smooth, abstract version of a human face. The robot has three main components: the base, neck, and projection surface as shown below in Figure 2-3.

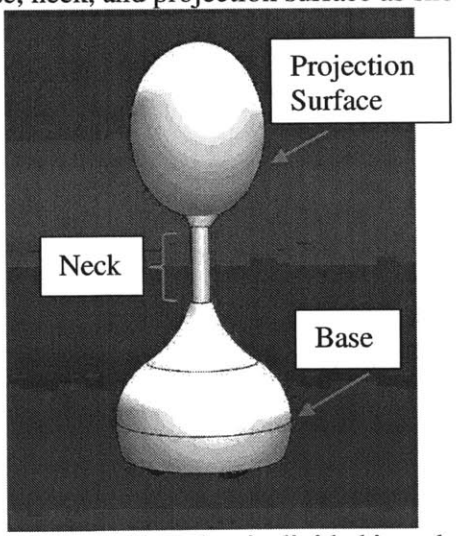


Figure 2-3: Labeled robot body. The robot is divided into three main sections: the projection surface, neck, and base.

Projection Surface

The projection surface is made of vacuum formed white acrylic, sanded with a fine grit sandpaper to reduce shine. It has a 10" major axis and 8" minor axis, and is an oblong dome. A camera is mounted flush with the back edge (looking into the bowl) of the projection surface.

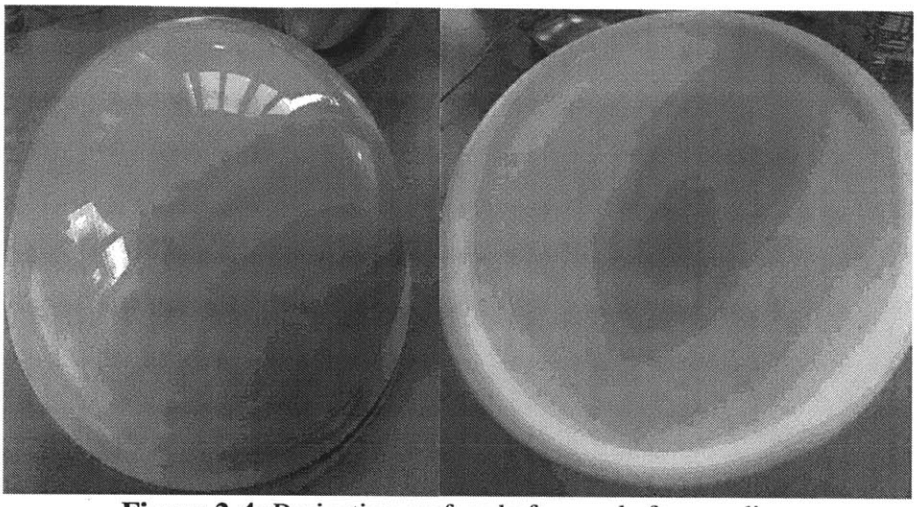


Figure 2-4: Projection surface before and after sanding

Head-Neck Connector

The head and neck are connected by a 3D printed piece that mates with the neck tube and curves to sit flush against the surface of the oval dome. Details regarding the head-neck connector can be found in Appendix B. The connector has a hole for mounting and a slot for routing the camera and cord through the connector and dome so the cord can be protected inside of the neck.

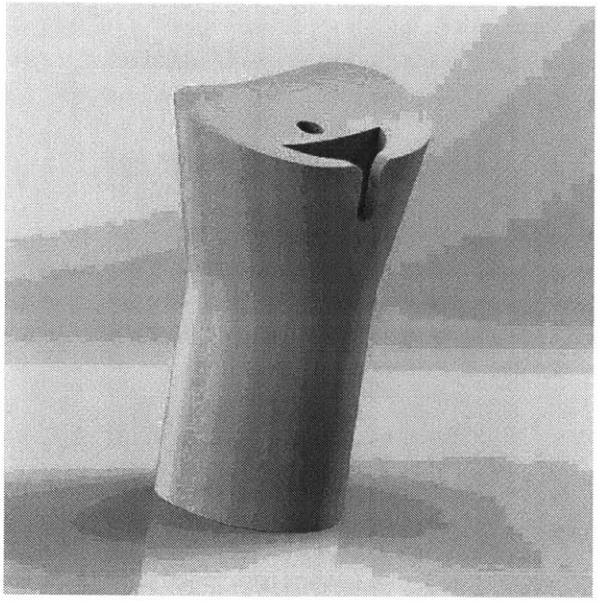


Figure 2-5: Render of the 3D printed head-neck connector piece. More information is in Appendix A.

Neck

The neck is made from 1" outer diameter polycarbonate tubing. It was purchased clear, then sanded to have a similar texture as the body base and projection surface. It was then painted white.

While not installed on the current model, a turning "turret" design was developed to allow the projection surface to turn independently of the base. This was desirable as it gives the robot more control when finding the best position to capture light. This will be the next feature implemented on the robot. Items related to the turret design can be found in Appendix A. Currently, a static neck to base connector piece is used instead.

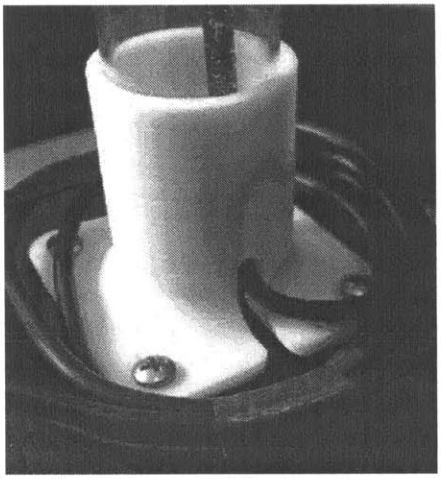


Figure 2-6: Fixed neck to base mount with slots for cord routing. This piece is mounted to the top of the tower, but has a 4 rubber washers to dampen vibrations between the head/neck system and the moving tower and base.

Base

The base of the robot contains the white matte shell and the electronics tower contained within. The tower is mounted on the cart, which is four servo and omni-directional wheels mounted in a tight square. Although 3 wheels would have allowed for all directions of travel and reduced energy use, the directional controls are less intuitive than with the current setup.

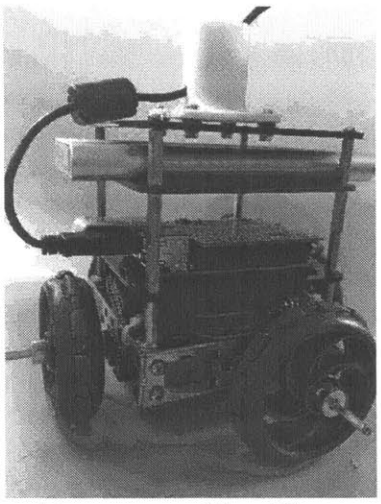


Figure 2-7: Square base and tower assembly. These components are covered by the white plastic shell when the robot is fully assembled. The shelves are laser cut 1/8" acrylic.

2.3 Sensors and Introspective Camera

The robot will be operating in an environment that has obstacles of varying position depending on the day and configuration of the gallery. The large glass wall of the gallery is also a defining feature of the space and provides the interface between robot and viewer. During the decision making process regarding robot actions, it was determined that the robot should be able to operate within the space without harming objects or people in its path. The first step to accomplishing this goal was to limit the speed of the robot so that if collisions occur, they are not damaging to objects or the robot. The second step was to determine a method of obstacle detection, either after a soft bump or before collision occurs. Six options were proposed as potential solutions.

Mechanical Contact Switches

Option 1: Surround the robot base with a low-force tape switch. While the design of the tape switch is a simple flexible moving contact switch, the tape switch is expensive compared to other options. To surround the robot base with tape switch in all directions, the cost would be $64, more than quadruple the cost of other options.

Option 2: A series of micro switches distributed around the base of the robot. This option is essentially the same as the tape switch, with more wiring but less cost.

Both mechanical switch options posed a problem with placement. With the robot design tapering at the bottom plate, placing the switches around the bottom edge of the robot would not suffice as the switches would not be the first part of the robot to contact an object. Placing the switches at the widest section of the robot would be an effective solution, but would change the aesthetic of the robot.

Sensor Solutions

Option 3: Gyro/Accelerometer to detect if the robot is stuck. This option did not detect obstacles or avoid collisions, rather it would be used to detect when the robot wheels are moving but the robot is not moving. At this point, the onboard computer would send a signal to the controller to interrupt the normal operation pattern and try to rotate and back up until the robot is able to move again for a certain amount of time. This is an interesting feature that should be considered for further development. However, it also allows the robot to plow over objects, and is not a sufficient solution on its own. Paired with another sensing method, this could be a valuable addition to the robot.

Option 4: Tape/Color boundaries. Another proposed idea was to mask off the boundaries of the robot using colored tape. The tape could be placed to help the robot avoid objects in the gallery. Static boundaries would not resolve collisions with any other active objects in the space.

Option 5: Infrared distance sensors were considered as a way to avoid active and passive obstacles within the space. Without knowing whether the glass windows of the gallery were coated with an IR reflective coating, this option was eliminated.

Option 6: Ultrasonic distance sensors are currently mounted on the robot to detect the distance from the glass windows and obstacles within the gallery. The main challenge with the ultrasonic distance sensors is determining the orientation and number of sensors necessary to adequately navigate the environment. For this version of the robot, four sensors oriented 90° apart were used.

A combination of gyro/accelerometer and ultrasonic distance sensing seems to be the best approach to obstacle detection, avoidance, and determining when the robot is stuck.

Introspective Camera

To determine whether or not the robot is within the projection field, an internal camera is mounted within the oval dome. It takes pictures of the inside of the dome and turns the image into sections of light and dark. Based on the percent of the image that is dark, the robot turns to find a section with more light. The mount for this camera had to be minimal to avoid casting shadows on the inside of the dome. Since the light/dark sensing is not an exact way to determine the boundaries of the projection field, the camera can move a little without affecting the quality of the results.

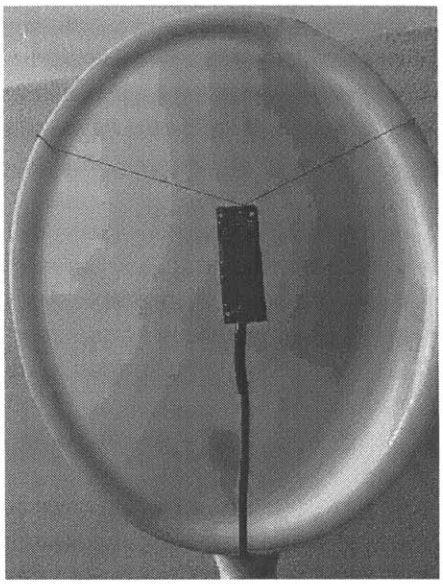


Figure 2-8: Minimalist attachment for the introspective camera. Initially a plastic bracket was designed but under certain light conditions it would cast a large shadow onto the inside of the dome, affecting the robot's ability to detect light. When in use, an opaque sheet covers the back of the dome to reduce shadow from behind the head.

2.4 Code Development

2.4.1 Robot-Controller Communication

Initially, robot-controller communication was accomplished through the use of an ad hoc network. However, after concerns about leaving an open network running a server-client system was developed instead. The robot acts as the server, and the controller is the client. This system effectively allows the controller to send wireless commands to the robot, telling the robot to move. For more details regarding the development of this system, see appendices D-G.

Controller	Robot
Detect Motion Take Picture of Subject Crop Face Determine Size of Image Determine Appropriate Motion Send Motion Command	Move according to controller command Take picture of light inside "head" Determine percent of light v. dark Use percentage to determine how much to move

Figure 2-5: Division of commands between controller and robot. The controller is responsible for all actions except for the ambient light detection, which is processed by the onboard robot computer.

3. Integration and Testing

The success of the proposed system depends on many subsystems working together. At the current state, the subsystems have been tested individually under the previous communication system. As the new communication system is written, each subsystem is tested again to make sure the functionality has not been unintentionally modified. In this section, the evaluation processes for different subassemblies are described, along with a set of questions and challenges to continue to evaluate during integration.

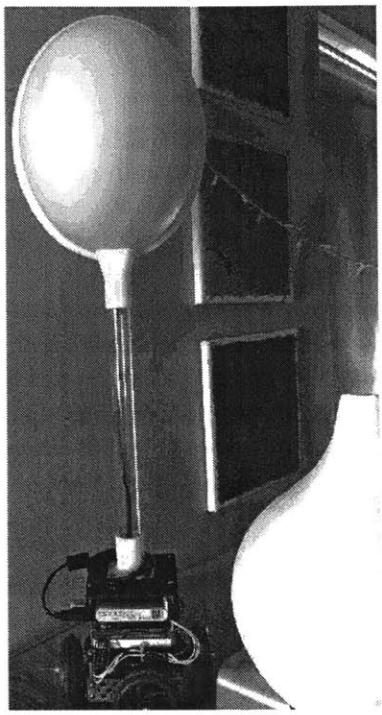


Figure 3-1: Assembled robot without cover. Cover is sitting on the right.

Controller Tests:
The first step in the system is detecting motion. The motion detector used specified it has a range of thirty feet and a 120 degree detection range. This was easily tested by moving objects at different locations while running a test script on an attached computer. The results confirmed the range of the motion detector.

The next step was to deal with small faces and crowds of people. Originally, the face detection program identified and highlighted all of the faces in a picture. While this was not the desired final behavior, it was a useful step to determine if crowds could be identified. To test the size limits of faces in crowds, multiple people stood near and far from the camera. The program is better at identifying larger and closer faces, but clarity is more important than size. A person whose hair hangs in front of a facial feature poses more of a problem than a small face. In one extreme test, a person held their state ID. The program successfully highlighted both their real face and photo. The smallest faces typically looked the worst when cropped, so the script was modified to only crop and use the largest face in the frame. There no current procedure for identifying if the robot has misidentified an object as a face.

Robot Tests:

To develop the light and dark separation code, a webcam was taped to the side of a translucent container with about 4" between the camera and the surface exposed to light. By varying the position of the light and the ambient light conditions, a value was found that best identified when the container was within the light of the projector. This was repeated with the final projection surface. However, since the gallery has movable lamps, recalibration is necessary if there are lighting changes within the space. Making the robot less sensitive to changes is a challenge that will be addressed as work continues on this project.

Integration Challenges:

The main challenges for integration are to make sure the system works as expected, and determine how to best move the robot to make the imagery either clear and recognizable, or so distorted that it is interesting. Through experimentation with the size adjustments made independently by the robot and the amount that the controller tells the robot to play in the light, a different "personality" could be created for the robot. Another integration challenge is to make sure that all of the actions happen quickly enough to be interesting. Image processing is a slower task, so there is the possibility that the timing of interactions may have to be changed so that the viewer is still interested in what is going on, even if there is a delay while their picture is being identified. Potentially the robot could move around, flash lights, or even both.

While running the robot in the museum space, it was found that the robot moves well across the floor surface. The floor does have some bumps that cause the robot to vibrate since the platform is significantly narrower than the total height. The distance between opposite wheels is about 6" and the robot is over 2' tall. The robot does not tip over due to the weight distribution and low speed. Small changes in the floor are reflected by motion at the top of the neck. This is visible and does not majorly affect image quality since the robot is already moving (the image quality is already low and quite variable).

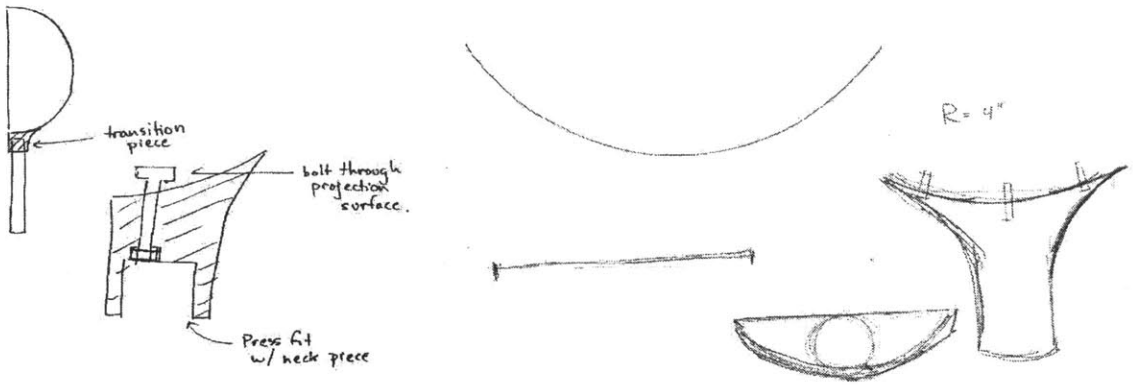
4. Summary and Conclusion

This project is an experiment with light, projection, and moving robots. One of the biggest factors that will determine the success of the project is how people react to the robot. The organic form of the robot and slightly humanoid characteristics intend to create a connection with the viewer, especially when man and machine share the same face.

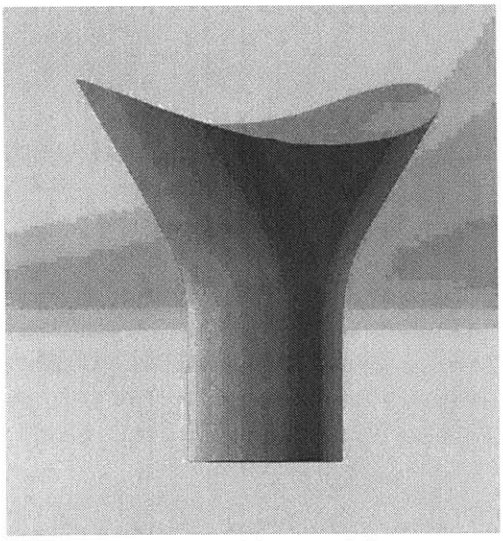
The next steps include the continued development of effective movement commands from controller to robot. The communication method is effective, that is, the robot can move on command from the controller. However, viewer feedback and further experimentation is necessary to determine which movement patterns are the most visually interesting. Also, the controller needs to be able to send images to a constantly refreshing website that the projector can access to project faces. While the on robot camera sensing system and distance sensors were tested (and determined functional) as subsystems, they have yet to be integrated into the final system. Continued information about integration can be found at https://github.com/morganrstewart/museumbot.

Appendix A: Head-Neck Connector

The piece that connects the 1" polycarbonate tube and the oval dome projection surface is a 3D printed piece designated the "neck interface". To design this piece, the edge of the oval dome was traced to determine if the actual dimensions were as specified when purchased. The 8" x 10" dimensions were confirmed. From here, a model was created from initial sketch ideas and then 3D printed. The fit of the interface piece was very important, as a loose fit would allow the projection surface to move a lot as the robot moved.



Initial sketches of neck interface piece.



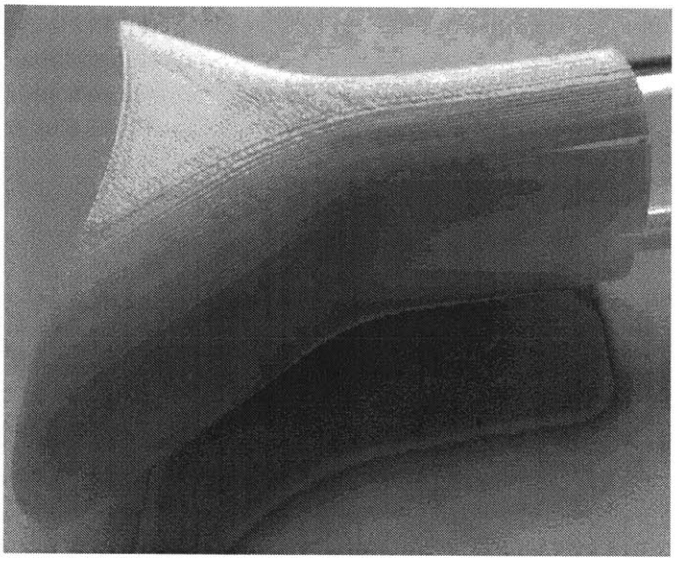
Render of 3D model.

The interface piece was printed on a FlashForge Creator Pro, a desktop FDM printer. Previous personal experiments determined that interfacing parts should be .007" larger for a press fit. Part orientation during the print was critical. The direction of the layers of the print greatly affected the strength of the print during installation.

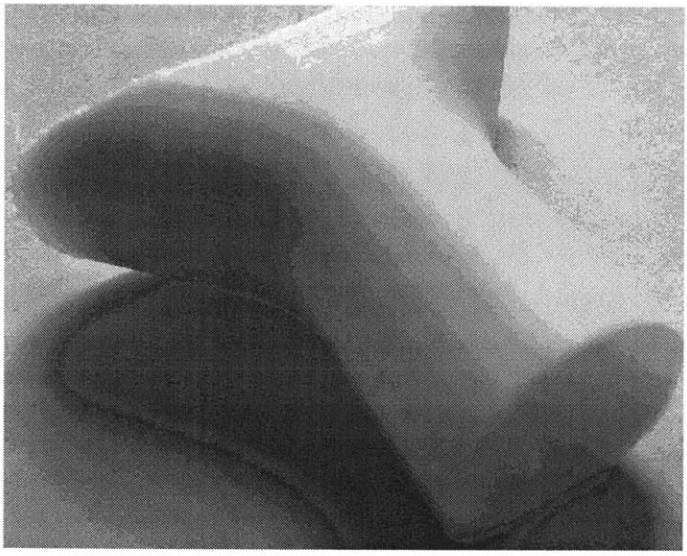
In the first attempt, the part was oriented on its side while printing. This was the fastest way to print the part, and overall surface finish was good. However, the round hole for the

polycarbonate tube warped and was not circular as the print went on. As such, it did not mate well with the polycarbonate tube, and the force from the tube cracked the print along the print layer boundaries.

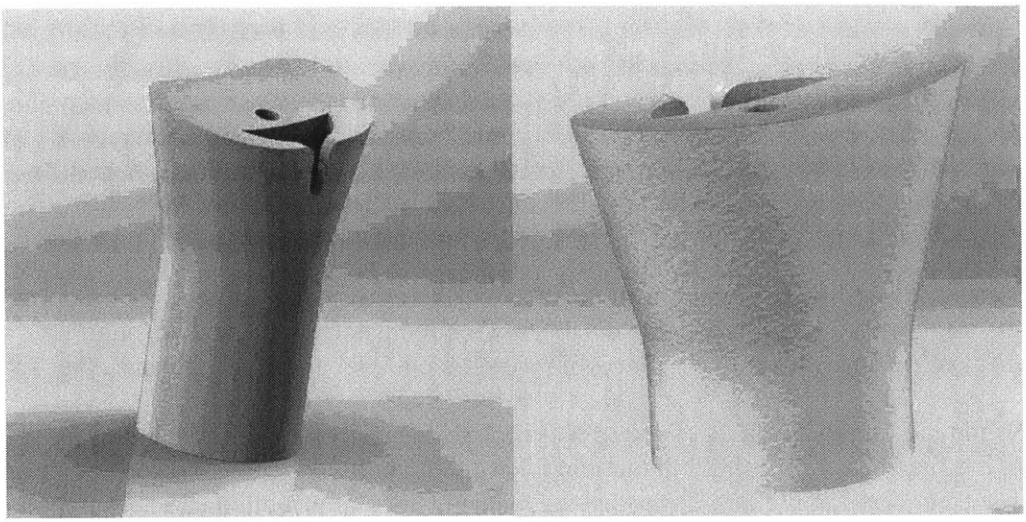
For the second attempt, the part was re-oriented so that the print layers would be in the opposite direction. The resulting print was twenty minutes slower and the overall surface finish on exposed areas was equivalent. The hole for the polycarbonate tube was circular, and able to be sized for a proper fit.



First print with layers parallel to the line of the polycarbonate tube, cracks present.



Layers are perpendicular to the polycarbonate tube, and the hole is more round.



Final two revisions of part. Here, a slot was added so the camera could be routed through the part, hiding the cable within the neck tube. The reduced size sat more flush with the surface and also reduced print size and time. The current robot is wearing the first of these two images. It will be switched to the second image (the updated model) when it is printed in a non-yellow material.

Appendix B: Turret Design

To give the robot a more "human" feel, one goal of the design was to create a head that can rotate independently of the body. This rotation is similar to the capabilities of an owl. Independent head rotation also allows the projection screen to be oriented normal to the projector regardless of the motion of the base. At the current stage, this functionality is not optimized as it required that the robot constantly knows where the projector is. Through the robot's light seeking code that uses images taken from the inside of the head the robot turns its head until it locates the brightest source of light. This is usually the projector given the controlled environment of the museum studio gallery, but would require recalibration for any other space.

While the first two rounds of prototypes included a head that was movable in the z direction, the final script for robot interaction did not require this functionality so it was removed to reduce weight and complexity. Proposed mechanisms for z axis motion can be found in Appendix B.

The large surface area of the projection surface "head" acts like a sail when the robot is moving. Consequently, the turning head could not be directly mounted to a servo or motor without support. The final design features a low speed, high torque DC motor coupled with a supported bearing that connects the body of the bot to the adjustable length neck. The first bearing considered was a radial ball bearing (simply because it was on hand) but it was dismissed as the weight of the turret would be acting in the wrong direction for that type of bearing.

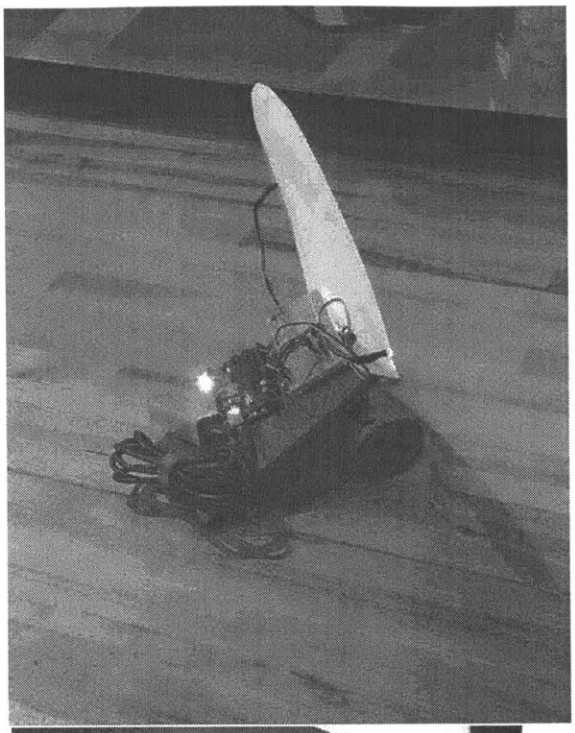
Then, a bearing for a lazy susan and a small thrust bearing were considered. Both of these bearings will require a custom piece built for the motor mount/bearing alignment/connection to the neck. The lazy susan bearing has pre-drilled mounting holes and would be easier to assemble. However, the large footprint and large amount of vertical play in the lazy susan bearing (especially when extended over the length of the neck and projection surface) made it a poor choice. A regular needle thrust bearing would be used.

	Lazy Susan Bearing	Radial Ball Bearing	Thrust Bearing	No Bearing
Vertical load (weight)	+	-	+	0
Radial load	0	+	0	0
Runout/Play	0	+	+	0
Footprint	-	0	0	0
Summary	**0**	**+**	**++**	**0**

Appendix C: Prototypes and Comments

Prototype 1: "Light Seeker"

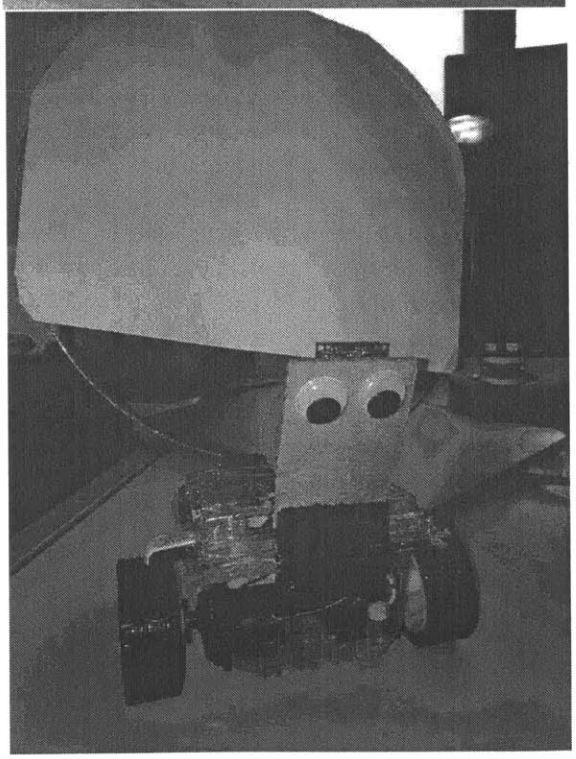
Features:
- Two photoresistors to detect light
- Ambient light calibration
- Random left/right turn if light was below the threshold
- Arduino



Prototype 2: "Blueberry"

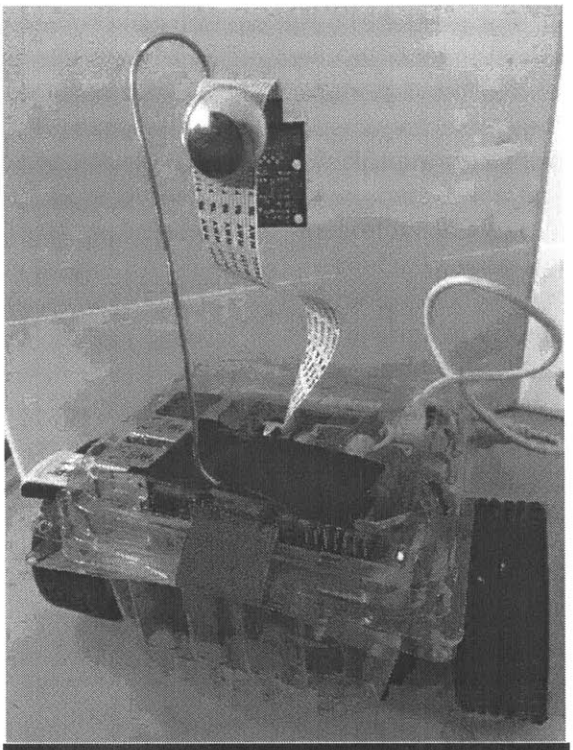
Features:
- First use of onboard "selfie" camera
- Identified circles
- Two powered wheels and a caster
- Ultra compact
- Raspberry Pi

This prototype was back when the focus of the project was essentially to create a visual programming language through projected imagery.



Prototype 3: "Blueberry Lite"
(missing projection surface in this photo)

This prototype was essentially the same as prototype 2, except that the camera was easily repositionable. Prototype 3's mobile camera enabled experiments with using a camera as a light sensor by looking into the projector directly, or looking instead at projected imagery.

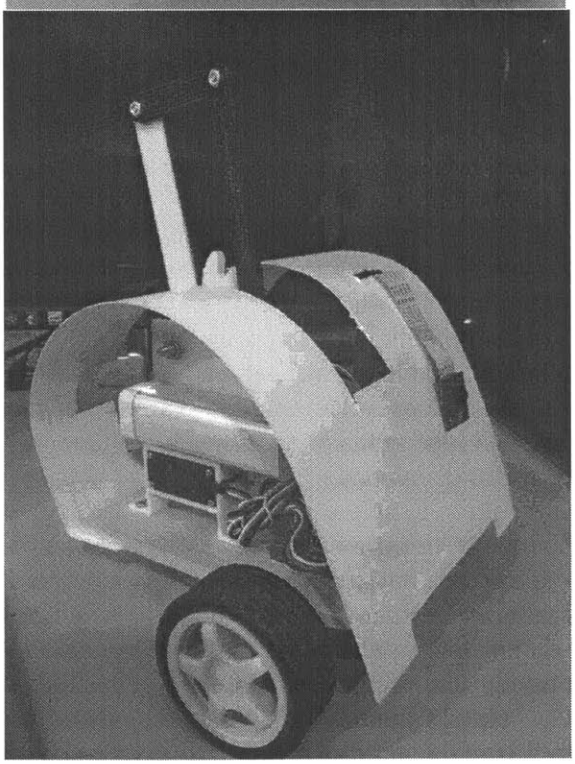


Prototype 4: "Bristol"

This prototype features a gear and servo driven four bar linkage that could raise and lower the projection surface.

A cross piece was mounted on the top of the four bar and held the projection disk. A small mount would have also been attached to the top bar and would have held the camera so that it was fixed relative to the projection surface.

The projection disk acted like a large sail and needed additional support to avoid bending back.

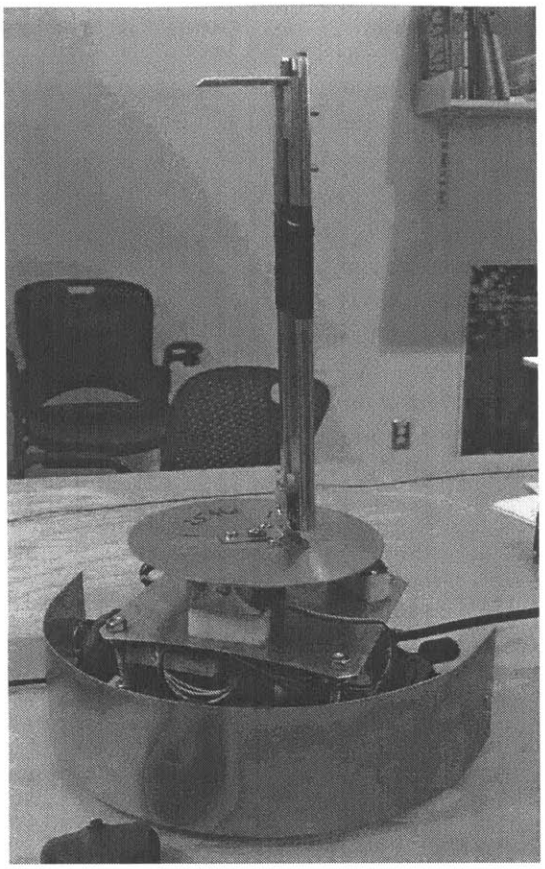


Prototype 5: "Aluminum"

Features: external controller
- 4 omni wheels
- First attempt at a turret device

This first attempt at a robot with independently turning head actually worked – the servo in the middle powered the turn and the disk was supported by the rollers. Misalignment made this very noisy.

Also, looked at having an off center neck piece, and found it contributed to noise as well



Appendix D: Motion Detector Script

This script is called by the controller upon booting and is the trigger for the entire communication sequence. Once the motion detector senses movement, it sends False back to the controller and the controller moves on to taking pictures. If there is no motion, this script loops until motion occurs. This code is modified from the Adafruit motion sensing tutorial [3].

robot_motion.py

```
def main():
    import time
    import RPi.GPIO as io

    io.setmode(io.BCM)

    pir_pin = 15

    io.setup(pir_pin, io.IN) # activate input

    while True:
        if io.input(pir_pin):
            print('PIR Alarm!')
            return False
        else:
            #print 'no alarm'
            pass
    time.sleep(0.5)

if __name__ == '__main__':
    main()
```

Appendix E: Camera Controller Script

This script follows the motion sensor script in the main body of the controller. When the motion detector is triggered, the script moves to this module, which takes a picture of the surrounding area. It then checks to see if there are any faces identified in the image. If there are not, the main script loops back to look for motion again. If there are faces, the image is cropped tight to the face. This face can then be sent to the projector's website. [4] [5] [6]

```
# controller_camera.py

import time
from SimpleCV import Camera, Image
camera = Camera()

def take_picture():
    #camera = Camera()  #Camera initialization
    #display = Display() #Display intialization

    print 'camera ready'
    time.sleep(.1)
    img = camera.getImage()
    #img.save(display)
    time.sleep(1)

    faces = img.findHaarFeatures("/home/pi/haarcascade_frontalface_alt2.xml")
    mlist=[]
    t = type(faces)
    if str(t) == "<type 'NoneType'>":
        print 'no faces'
        return False

    else:
        for f in faces:
            # print "found a face at " + str(f.coordinates())
            mlist.append(f)

        dim = faces.length()

        m = max(mlist)
        ind = mlist.index(m)

        faces[ind].draw(width = 3)
        # img.show()
        time.sleep(4)

        x_w = int(dim[ind]/2)
        y_h = x_w
```

```
        coords = faces[ind].coordinates()

        x = coords[0]
        y = coords[1]

        cropped = img[(x-x_w):(x+x_w), (y-y_h):(y+y_h)]
        cropped.show()
        print 'new face time'
        return True

if __name__ == '__main__':
    take_picture()
```

Appendix F: Main Controller Script

[7] [8]

```
# controller_test.py

import socket
import sys
import robot_motion
import controller_camera

try:
   #create the AF_INET STREAM socket
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

except socket.error:
   print 'Could not create socket.'
   sys.exit()

print 'socket created'

host = '18.111.82.92'
port = 8089

try:
   remote_ip = socket.gethostbyname(host)

except socket.gaierror:
   # could not resolve
   print 'hostname could not be resolved'
   sys.exit()

print 'IP address of ' + host + ' is ' + remote_ip

#  Connect to remote server
s.connect((remote_ip, port))

print 'Socket connected to ' + host + ' on ip ' + remote_ip
while 1:
   if robot_motion.main() == False:    # runs continuously until PIR sensor is triggered
      if controller_camera.take_picture() == True:
         message = 'run'
         try:
            s.sendall(message)
            print 'message sent'
         except socket.error:
            print 'Send failed'
```

```
            sys.exit()

        ### Receive data from server
        reply = s.recv(1024)
        print reply
    else:
        print 'retry - looking for motion again'
        robot_motion.main()

### Close socket
s.close()
print 'received ' + reply
```

Appendix G: Server Script

This script must be running before the client script is initiated to start the system properly. [7] [8]

robot_server.py

```
def main():
    # Server
    import socket
    import sys
    import thread
    import robot_motion

    HOST = ''   # all available interfaces
    PORT = 8888 # arbitrary non-privledged port

    # Create a TCP socket
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print 'socket created'

    try:
        s.bind((HOST, PORT))
    except socket.error, msg:
        print 'Bind failed. Error code: ' + str(msg[0]) + ' message ' + msg[1]
        sys.exit()

    print 'Socket bind complete'

    ### listen for connections

    s.listen(10)
    print 'socket is now listening'

    def clientthread(conn):
        conn.send('welcome to the server type something and hit enter\n')
        # infinite loop
        while True:
            # Talk to client
            data = conn.recv(1024)
            if data == 'run':
                    robot_motion.main()
            reply = 'OK ...' + data
            if not data:
                break
            conn.sendall(reply)
        conn.close()
```

```
    ### accept new connections
    while 1:
        conn, addr = s.accept()
        print 'connected with ' + addr[0] + ':' + str(addr[1])

        #start mew thread
        thread.start_new_thread(clientthread ,(conn,))
    s.close()

if __name__ == "__main__":
    main()
```

Appendix H: Introspective Camera test

```
import os
import time
from SimpleCV import Camera, Image, Display
import pigpio

runcount = 0

def all_stop():
   os.system('echo "s 17 0" >/dev/pigpio')
   os.system('echo "s 18 0" >/dev/pigpio')
   os.system('echo "s 27 0" >/dev/pigpio')
   os.system('echo "s 15 0" >/dev/pigpio')

def servo_1_fast():
   os.system('echo "s 17 1700" >/dev/pigpio')

def servo_2_fast():
   os.system('echo "s 18 1700" >/dev/pigpio')

def servo_3_fast():
   os.system('echo "s 27 1700" >/dev/pigpio')

def servo_4_fast():
   os.system('echo "s 15 1700" >/dev/pigpio')

def robot_ccw():
   servo_1_fast()
   servo_2_fast()
   servo_3_fast()
   servo_4_fast()
   time.sleep(1)
   all_stop()

cam = Camera()
disp = Display()

#while True:
while runcount < 5:
   img = cam.getImage()
   img.save(disp)
   img = img.binarize()
   img.show()

   area = img.width * img.height
```

```
counter = 0

first = img.getPixel(0,0)
last = img.getPixel(img.width-1, img.height-1)

##matrix = img.getNumpy()
##matrix.shape
##flat = matrix.flatten()
x =0
y = 0
counter = 0
for x in range(0, img.width):
   for y in range(0, img.height):
      if img.getPixel(x, y) == (0.0, 0.0, 0.0):
        counter +=1

percent = float(counter)/area
print percent
if percent < .8:
   print 'move'
   robot_ccw()
runcount += 1
```

Appendix I: Communication via Twitter API

For the first four prototypes, communication with the robot occurred via the Twitter API. Direct messages were sent to Twitter, and a controller could read the messages and use the information (IP address, time, etc.) to determine which commands to send back to the robot.

The first step to Raspberry Pi – Twitter communication is the setup of a Twitter App that will allow read and write access to a twitter account. Using a Twitter account that is dedicated for the project, an app was created following the instructions on apps.twitter.com. A new application was created with permissions to read, write, and access direct messages. Keys and access tokens were generated through the app creation interface and stored in a file to be referenced by a script on the Raspberry Pi.

Using the Twython module made communication via Twitter fairly simple. The following was a test to make sure it would tweet properly. The following code is based on examples from Scott Kildall's Twitterbot documentation. [9]

To send a tweet from an app/Raspberry Pi

```
from twython import Twython

tweet = 'sending a tweet from the robot controller'

aKey = "XXXXXXXXXXXXXXXXXX"
aSecret = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
aToken = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
aTokenSecret = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

api = Twython(aKey, aSecret, aToken, aTokenSecret)

api.update_status(status=tweet)
print 'tweet successful'
```

Appendix J: Robot- Controller Communication via Ad Hoc network

The robot and controller both contain a Raspberry Pi, a small computer equipped with a wireless internet connection. An ad hoc network was used as a first attempt at communication between the robot and controller. The network, "robotAdHoc", allows the robot and controller to communicate without the need for a router or other central access point. This reduced set up costs and provided a quick and effective solution.

To configure the ad-hoc network on the Raspberry Pi, the network interfaces file was edited to change the mode from infrastructure mode to ad-hoc mode. When the file was edited, static IP addresses were assigned to the controller and the robot.

The procedure for editing the interfaces file is documented on the Debian Wiki. [10]
Once the controller and robot are connected via the ad-hoc network, they can communicate using ssh to control the actions of the other body. For example, when the controller tells the robot to move, it uses ssh to access the terminal for the robot body, then sends a command to either run a python script stored on the robot computer or directly send a gpio command to make the robot move in a certain way. Current challenges with this method of communication are that the password for the ssh target must be entered after each command, which is difficult to do within the body of the controller program. A major consideration when designing the robot communication system was determining which commands would be performed from the controller computer as opposed to the robot computer.

While the ad-hoc solution is an effective method to communicate between the robot and controller, it is also an unsecure method of communication. Any device with a Wi-Fi adapter within range will be able to see the network, and could potentially access the network if it provides the correct password. Ad-hoc networks are mainly used for temporary connections for this reason. Since the robot will be exhibited in a public space, it did not seem smart to have the robot and controller constantly radiating access points. Another potential problem with the current ad-hoc network is that the maximum bandwidth of an ad-hoc network is one fifth that of other Wi-Fi networks. The communication delay is much less than the delay caused by image processing, but the more recent network solution is able to transmit commands faster, improving the overall performance of the robot.

Appendix K: BOM and Part Information

Mechanical Components

BOM

Name	Part Number	Quantity
Projection Surface	Oval Dome (custom)	1
Neck		1
Shell	Custom	1
Neck to Head Joint	Custom	1
Neck to Tower Joint	Custom	1
Tower "shelves"	Custom (laser cut)	2
Battery shelf	Custom	1
Frame pieces		8
Standoffs	276-195	16
Omni wheels		4
Shaft collars	276-2010	8
Square shafts	276-2011	4
Nylon spacers		8

Projection Surface: Oval Dome
http://www.eztopsworldwide.com/OD8x10.htm

Instructions:
1. Using 240 or 320 grit sandpaper, sand both surfaces of the dome to remove shine
2. Cut off flange if present
3. Drill hole for mounting to neck

Neck: Polycarbonate Tubing
http://www.amazon.com/gp/product/B000OM9IO8/ref=oh_aui_detailpage_o00_s01?ie=UTF8&psc=1

Instructions:
1. Cut off one foot of tubing
2. Remove sharp edges
3. Sand with fine sandpaper so that it feels similar in texture to the sanded oval dome

Shaft Collars
http://www.vexrobotics.com/276-2010.html

Shafts
http://www.vexrobotics.com/shafts.html

Omni Directional Wheels
http://www.vexrobotics.com/wiki/Omni_Directional_Wheel

Electronic Components

BOM

Name	Part Number	Quantity
Raspberry Pi 2	DEV-13297	2
Webcam (Controller)		1
Mini SD Card		2
WIFI adapter		2
Webcam (Robot)		1
Motion Sensor	2760347	1
Momentary Switch		1
Motors	276-2181	4
Motor Drivers	276-2193	4
Distance Sensors	HC-SR04	
Power Supply		2
Board		As needed
Jumpers/Wire		As needed
Heat Shrink		As needed
Electrical Tape		As needed

Raspberry Pi 2
https://www.sparkfun.com/products/13297

Motion Sensor
http://www.radioshack.com/radioshack-passive-infrared-sensor/2760347.html#.VTmhTCFViko

M3 x 26mm Standoffs
http://www.amazon.com/Female-Screw-Stand-off-Spacer-Length/dp/B00AO436FI/ref=sr_1_4?ie=UTF8&qid=1431566708&sr=8-4&keywords=standoffs+m3

References

[1] M. M. Studio and O. Piene, Artists, *Light Ballet Robots.* [Art]. Miller Yzerski Gallery, 2015.

[2] "MIT Museum Studio," MIT Museum Studio, 2015. [Online]. Available: http://mitmuseumstudio.mit.edu/. [Accessed 3 April 2015].

[3] S. Monk, "Adafruit's Raspberry Pi Lesson 12: Sensing Movement," 13 2 2013. [Online]. Available: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-12-sensing-movement/hardware. [Accessed 10 March 2015].

[4] Sight Machine, "SimpleCV Image Arithmetic," 2013. [Online]. Available: http://tutorial.simplecv.org/en/latest/examples/image-math.html. [Accessed 14 March 2015].

[5] University of Cambridge: The Computer Laboratory, "Physical Computing with Raspberry Pi: Blob Detection," [Online]. Available: https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/blob_detection/. [Accessed 13 March 2015].

[6] C. Carbajal, "Computer Vision using SimpleCV and Raspberry PI," [Online]. Available: http://www.researchgate.net/publictopics.PublicPostFileLoader.html?id=54aada77d039b1666d8b45a5&key=5077bcd7-30c9-4be9-a926-939b7f23f85e. [Accessed 13 2 2015].

[7] H. John, "Raspberry Pi StackExchange: How to receive incoming data from TCP socket," 20 December 2014. [Online]. Available: http://raspberrypi.stackexchange.com/questions/26158/how-to-receive-incoming-data-from-tcp-socket. [Accessed 22 April 2015].

[8] BinaryTides, "Python Sockets - Network Programming Tutorial," 22 July 2012. [Online]. Available: http://www.binarytides.com/python-socket-programming-tutorial/. [Accessed 22 April 2015].

[9] S. Kildall, "Raspberry Pi Twitterbot: Instructables," 2014. [Online]. Available: http://www.instructables.com/id/Raspberry-Pi-Twitterbot/. [Accessed 24 February 2015].

[10] JPierreGiraud, "Debian Wiki: Wi-Fi Ad Hoc," 22 2 2012. [Online]. Available: https://wiki.debian.org/WiFi/AdHoc. [Accessed 30 3 2015].

[11] SF., "Stack Overflow: Pi to Pi Communication," 4 December 2013. [Online]. Available: http://raspberrypi.stackexchange.com/questions/12061/pi-to-pi-communication. [Accessed 30 4 2015].