ECE 478/578 PROJECT 1 REPORT

Fall 2018

Project:

Schrodinger Cat

Submitted by:

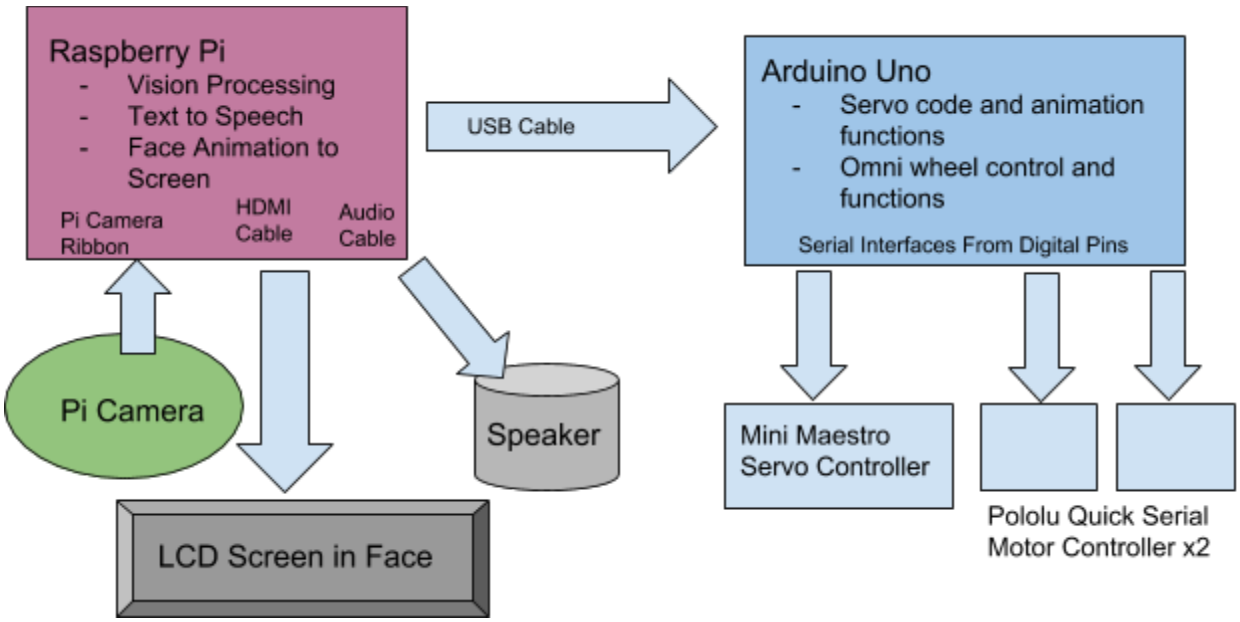
Rakhee Bhojakar
Chelsea Brooks
Erik Fox
Dhakshayini Koppad
Jamie Williams

Table of Contents

Project Summary

Schrodinger Cat (what used to be the Einstein robot) is one of the oldest robots in the lab. Schrodinger Cat is made up of a full body (head, arms, and legs controlled by servos) suspended above the ground by a base (which moves via a set of omni-wheels). The servos and motors are controlled with an Arduino Uno microcontroller. We added a Raspberry Pi to the head of the robot to add vision, speech, and a screen to the robot.

The control scheme of the robot is as follows:



Previous Work on Robot

When we received the robot, it came with all the hardware needed control the motors and servos. There were minor issues, however, which lead to redoing much of the wiring of the board, as well as replacing a few components which had died. We also removed the Bluetooth capability, as it was not needed for this Project.

The previous group used a laptop to provide most of the intelligence of the robot. We decided to change that to a Raspberry Pi to take advantage of the Pi camera and image processing abilities. We also removed the existing robot head, and replaced it with a 3D printed frame and an LCD screen showing a face animation.

Team Contribution

Text to Speech – Chelsea

Face and Object Detection - Rakhee

Servos and Arduino Code - Jamie

Omni Wheels - Dhakshayini Koppad

3D Printing/Screen Face - Erik

Text to Speech

Text to Speech is a great way to give life to your robot. Our original robot did not have text to speech capabilities, so we used a Raspberry Pi to give Schrodinger's Cat a voice. We used a program called espeak. Espeak has multiple different settings for text to speech. You can change the gender, speech speed, and spaces between words.

Espeak is downloaded on the Raspberry Pi using `sudo apt-get install espeak`. Once espeak is downloaded, the python script that controls the voice settings is

espeak -ven+m1 -s200 -g10 'Text' 2>/dev/null

- -ven+m1 is the gender. There are 7 different male voices and 4 different female voices.
- -s200 is the speech speed
- -g10 is the space between words.
- 2>/dev/null lets the system know that there is no more text.

One challenge we had was finding the perfect combination of gender, speed, and spacing between words. We decided that female voice at 180 speech speed and 20 pause between words was the best for this project. This worked okay, but some people found it hard to understand. Although we found the worked on fine tuning the settings, the robot did not have a very natural voice. We are looking to fix this in a later project.

The text to speech function of the robot was embedded in the vision detection python script. Using the robot's ability to talk, we could learn what the robot was seeing. This was helpful for testing and debugging the vision software. When the robot detected an object (either a human face or a 'mouse', actually a green ball) we could have it say that it saw the object, which side it noticed the object on (left or right), and then it could say when it saw the object leave its frame of vision. It would also say a sentence of greeting or describe how it wanted to interact with an object (e.g. "Ooooh. A mouse. I want to catch it.")

For our next project, we are going to use a different program for text to speech and speech to text so that we are able to have a more realistic voice. Especially since the robot will be interacting with other robots and with humans, it needs to speak loudly and clearly. We will also be adding larger speakers to the robot, so that it can be heard by an audience

Face and Object Detection

Object detection is one of the computer technologies, which connected to the image processing and computer vision and it interacts with detecting instances of an object such as human faces, building, tree, car, etc. The primary aim of face detection algorithms is to determine whether there is any face in an image or not.

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Using raspberry pi and pi camera we have detected faces present in video using haar cascade object detector. As of now we are only detecting faces present in the video, we plan to extend this capability to recognize faces robot sees.

The following is the snippet of face detection using haar cascade classifier.

```
# capture frames from the camera
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):

    cv2.imshow(window_name, image5)

    # grab the raw NumPy array representing the image
    bgr_image = frame.array
    bgr_image = imutils.resize(bgr_image, width = 300)
    gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)

    # find faces in the image
    faceRects = fd.detect(gray_image, scaleFactor = 1.1, minNeighbors = 5, minSize = (30, 30))
    bgr_clone = bgr_image.copy()

    # loop over the faces and draw a rectangle around each
    for (x, y, w, h) in faceRects:
    cv2.rectangle(bgr_clone, (x, y), (x + w, y + h), (0, 255, 0), 2)

    #print("I found %d face(s)" % (len(faceRects)))
    if len(faceRects) > 0:
        if (x < 105):

            os.system("espeak -ven+f1 -g5  'Look Theres my human on the left side meow Hi human ' 2>/dev/null")
            os.system("espeak -ven+f1 -g5  'I should go straight or else he will catch me ' 2>/dev/null")
            cv2.imshow(window_name, image2)

        elif(x > 125):
            os.system("espeak -ven+f1 -s180 -g10 'Look Theres my human on the right side meow Hi human' 2>/dev/null")
            cv2.imshow(window_name, image3)


    cv2.imshow(window_name, image5)
```

Now to detect color we need to know what is color in pixels of an image. Images are made of tiny dots of pixels each having a color and we can define those colors in terms of HSV -> Hue, Saturation, Value.

For color based object detection, we have converted RGB image format to HSV format . So the range of these are as follows

- Hue is mapped – >0º-359º as [0-179]
- Saturation is mapped -> 0%-100% as [0-255]
- Value is 0-255 (there is no mapping)

The following is the snippet of color object detection. Here we have decided range for different colors.

```
# convert the images from bgr to hsv
hsv_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2HSV)

lower_red1 = np.array([0, 70, 50])
higher_red1 = np.array([10, 255, 255])

lower_red2 = np.array([170, 70, 50])
higher_red2 = np.array([180, 255, 255])

mask1 = cv2.inRange(hsv_image, lower_red1, higher_red1)
mask2 = cv2.inRange(hsv_image, lower_red2, higher_red2)

# create NumPy arrays
lower_green_range0 = np.array([0, 100, 100], dtype = "uint8")
lower_green_range1 = np.array( [10, 255, 255], dtype = "uint8")

upper_green_range0 = np.array([45, 100, 100], dtype = "uint8")
upper_green_range1 = np.array([75, 255, 255], dtype = "uint8")


# find the colors within the specified boundaries and appy the mask
mask3 = cv2.inRange(hsv_image,  lower_green_range0 ,lower_green_range1)
mask4 = cv2.inRange(hsv_image,  upper_green_range0 ,upper_green_range1)

mask = mask4 | mask2

result = cv2.bitwise_and(bgr_image, bgr_image, mask = mask4)
gray = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
thresh = cv2.threshold(blurred,25 , 255, cv2.THRESH_BINARY)[1]

edged = cv2.Canny(thresh, 30, 200)
```

Challenges we encountered while doing face detection is finding algorithm to perform both detection with available raspberry pi resources. As these algorithm are computationally extensive, we could not get more frames per second as required to to real time detection.

In the future, we would like to put the face recognition and object detection in one algorithm. Also, the colors are currently hardcoded, which should be corrected.

Robot Head (Screen and 3D Printing)

The original face for the Schrodinger Cat Robot was an ESRA- Expressive System for Robotic Animation. This device did not have a flexible design, nor did it look like a cat. Additionally, it did not provide an ideal frame to place a camera. After deciding to replace the ESRA with a design of our own, the first step we took was to determine the requirements for our design. We concluded that we needed our design to be expressive, adaptable, and be able to house our camera. A 3D printed cat face frame, with a display to show facial animations, checked all of these boxes.

Our final design consists of a 3D printed cat face frame, an LCD display, a camera, a Raspberry Pi, and a series of cat face animations. The frame is designed to have a abstract look of a cat, a hole to expose a LCD display, a frame to hold the LCD display, a hole to expose

a camera, and a stand to mount the head to the robot. The cat face animations are sent to the display using OpenCV function calls.

When implementing our design, we ran into a few issues. The biggest challenge was the print of our cat face. At first the design was too large to fit on the only available printer. After resizing the design, to be as small as was possible with display we had available, it was still at the boundaries of the printer. This resulted in a few complications. The first was where to place the design on the printer software. It took multiple attempts to get the design placed correctly. The second issue was that printing surface had uneven heat. This resulted in failed prints because parts of the frame curved up and were hit by the printing head. Another challenge faced from printing resulted from the resolution of the printer. We anticipated this issue in our schematic by adding length to the dimensions, however it was not enough. To get around these issues for version 1.0, we used one of the printed faces where only the facade printed correctly. On this we created a support for the LCD with screws.

Aside from printing issues, there were some challenges faced with interfacing the Raspberry Pi with the LCD display. The first of these issues was that the Raspberry Pi wouldn't fill the LCD monitor. This issue was fixed by making some changes to the /boot/config.txt file. The second is was that the OpenCV image output function wouldn't take ownership of the display, which was necessary for the animations to look correct. We resolved this issue by including functions from the screen info library. Links to the solutions to these problems are included in the reference section.

The future changes to cat face should be to update the frame design, and to improve the animations. For the frame, each part should be printed separately, and the margin of error should increase so that the dimensions will work after printing. These changes would increase the likelihood that a print will be successful. Additionally, we should add an enclosure for the Raspberry Pi. As for the animations, we could improve them by utilizing digital animation software. This could allow for more realistic and professional looking cat faces.

Robot Body (Servo control)

The body of the robot was made up of a full body (head, arms, and legs) suspended above the ground by the base. After removing the old head at the beginning of the project, we were left with 15 servos in the body. There were four in each arm: the wrist/forearm, elbow, and two in the shoulder. Each leg had 3: the knee, and two in the hip. There was also a servo for the neck, to turn it side to side, that was left over after removing the old head.

All of these servos were controlled from the same servo control board, a Pololu Mini Maestro. This board had been added previously, and supports 24 channels of servos, of which

only 15 are being currently used. The Mini Maestro is controlled via serial connection to the Arduino, using two digital pins capable of serial communication (we used pins 10 and 11). Pololu maintains an Arduino library for the board (referenced at the end of this document) that allows for easy communication and control of the servos.

There were several issues with the wiring of the servo setup. First of all, the connections between the Arduino and the Mini Maestro were not done correctly, leading to the code we received not appearing to work. We managed to get the servos to work, but we found that the poor connections of the wires between the Maestro and the servos lead to multiple servos moving when only one was activated. We re-soldered those connections to make them cleaner and to provide larger wires for the power. Even after this, we found that there were several dead servos on the torso.

After fixing that issue, we found that the mapping between the servos on the body and the Mini Maestro channels was wrong, which led to the wrong servos responding to commands and moving in ways that could damage them. We had to spend time remapping each servo on the body to the appropriate channel on the control board. And, even then, the maximums and minimums defined by the previous group were not working as they should of. So, each of those parameters had to be redefined as well. The previous groups made use of only 3 servos per arm and only 2 per leg, so there were 4 servos that we needed to map and define from scratch. All of this had to happen before we could start working on animations for the robot servos.

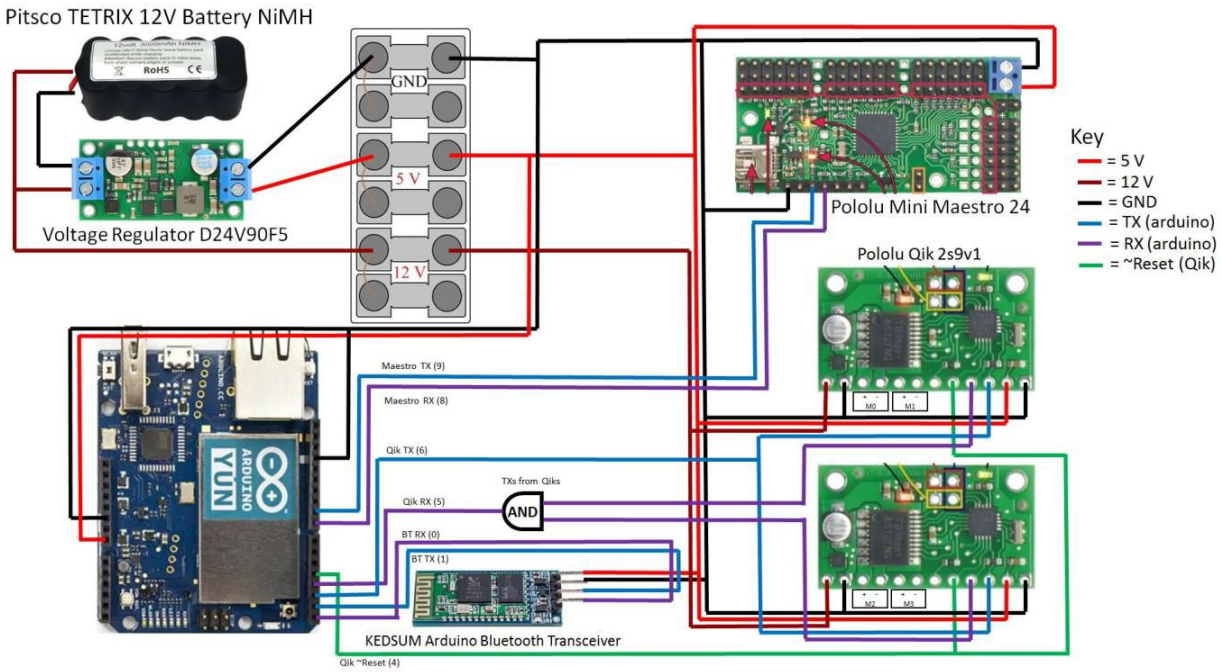
Originally, the Arduino was supposed to receive serial commands from the Raspberry Pi about what the robot was seeing. The Arduino has code to initiate certain gestures when the robot sees a “mouse” (tracked object) or recognizes a human face.

For the next project, the goal is to replace all of the dead servos, and work on better animations. The number of servos allows for excellent emotion and range of movement but adds a large amount of complexity. If the control board supports it, changing the speed of the servo movements might create more fluidity in the gestures. This will need to be experimented with later. Additionally, new animations will need to be created to match the new behaviors of the robot.

If we can get the current servos working well, it would be nice to add addition servos to the robot to give it new features. At the moment, the ears are just a part of the face plate. If we had time, it would be nice to have more realistic cat ears that could move or rotate to show emotion. Also, another servo (or multiple servos) could be added to create a tail for the robot, which could also bring more emotion and expression to the robot.

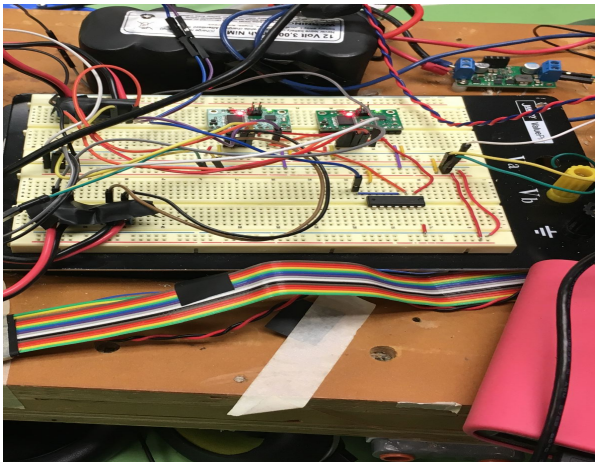
Robot Base (Electronics Board and Omni-Wheels)

Robot Electronics: The Robot base houses the Electronics of the board. The Schematics Blocks are shown below (although the Bluetooth capability was removed).



We have a 12V Battery that powers all the electronics and the motors. The wheels of the robot use 12V to run. We have a voltage regulator to step down the DC voltage to 5V that is used to power the rest of the components on the board.

Challenges: The NAND gate IC was damaged. Had to rewire the entire breadboard of connections between the Arduino, Pololus and the motors and replace the IC.



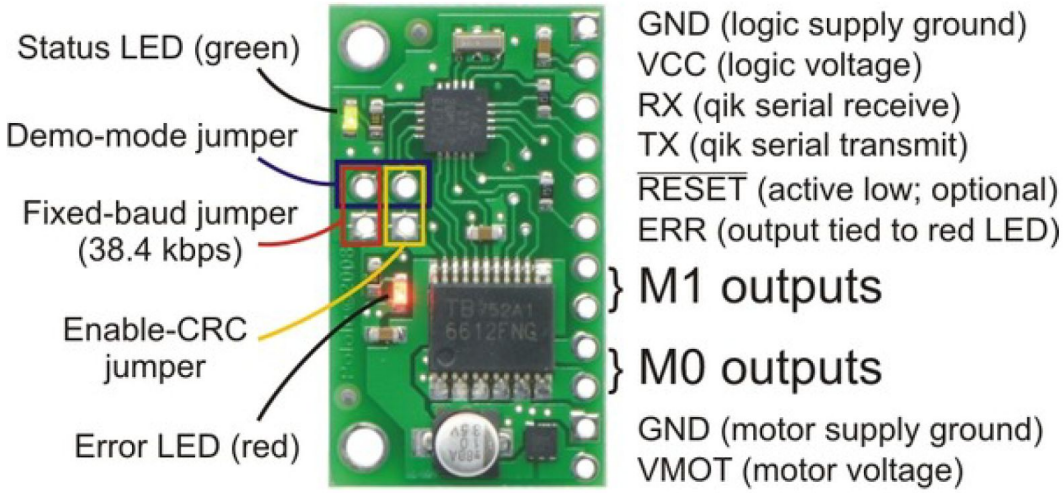
Next project: Transfer all the electronics on the breadboard to more reliable solderable general purpose PCBs.

Robot Wheels: The Robot uses 4 Ohmi Wheels. Omni wheels or poly wheels, similar to Mecanum wheels, are wheels with small discs around the circumference which are perpendicular to the turning direction. The effect is that the wheel can be driven with full force, but will also slide laterally with great ease.

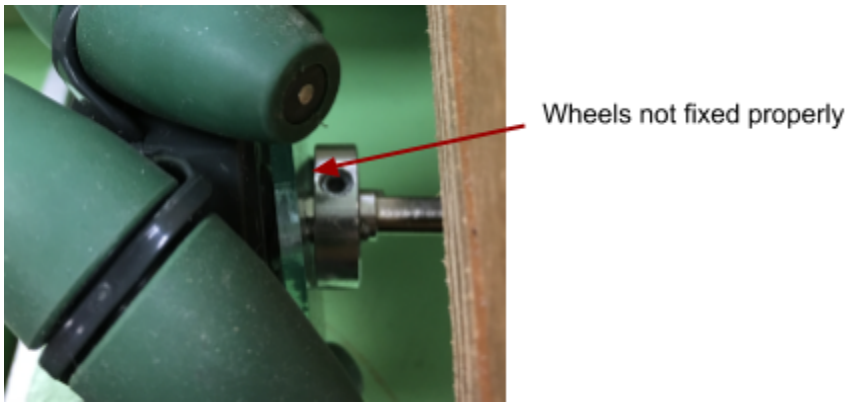
The robot wheels are controlled using two qik 2s9v1 to control the motors. It has the following features:

1. High-frequency PWM to eliminate switching-induced motor shaft hum or whine
2. A robust, high-speed communication protocol with user-configurable error condition response
3. Visible LEDs and a demo mode to help troubleshoot problematic installations
4. Reverse power protection on the motor supply (not on the logic supply).

Connecting the Qik:



Challenges: One of the Wheels was wobbling, had to remove and fix it again. The acrylic sheet holding the motor and the wheel was damaged.



Next project: Get the lateral left and right movement of the robot. Controlling the angles of the Robot.

Reach: Adding additional IR sensors around the robot so we can achieve obstacle detection around the body of the robot.

Media Links

Videos of the robot can be found on the project GitHub page at:
https://github.com/cbrooks1/ECE578_Schrodinger_Cat/tree/master/Project1/Videos

Code Reference

All of the code for this project (for both the Raspberry Pi and the Arduino) can be found on the projects GitHub:

https://github.com/cbrooks1/ECE578_Schrodinger_Cat/tree/master/Project1

Index of Materials for Project 1

Raspberry Pi 3 - Used for text to speech, screen, and face/object detection - $35
Raspberry Pi Camera
XPT2046 - LCD Touch Screen : $35

References

Text to Speech
https://www.dexterindustries.com/howto/make-your-raspberry-pi-speak/

Vision and Object Detection
https://thecodacus.com/opencv-object-tracking-colour-detection-python/
https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/
https://www.pyimagesearch.com/2016/02/15/determining-object-color-with-opencv/
https://gist.github.com/ronekko/dc3747211543165108b11073f929b85e
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html

Face Detection and Color Object Detection
https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html
https://www.superdatascience.com/opencv-face-detection/
https://www.blog.pythonlibrary.org/2018/08/15/face-detection-using-python-and-opencv/
https://www.geeksforgeeks.org/detection-specific-colorblue-using-opencv-python/
https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/

LCD Display/Face
https://gist.github.com/ronekko/dc3747211543165108b11073f929b85e
https://raspberrypi.stackexchange.com/questions/53931/why-is-my-raspberry-pi-3-display-not-filling-the-screen/54074

Servo Control
Details on the Pololu Mini Maestro: https://www.pololu.com/product/1356/resources
Arduino Library for the Mini Maestro: https://github.com/pololu/maestro-arduino

Board Hardware

Details on the Pololu Qiks - https://www.pololu.com/product/1110

Arduino Library for the Pololu Qiks - https://github.com/pololu/qik-arduino