Chimpanzee on Wheels

ECE478 Robotics 1 - Project Report

Julian Saunders
Aaron Chan

Table of Contents

Group Details

Our group consisted of the following individuals:

Aaron Chan	chanaar@pdx.edu
Julian Saunders	sjulian@pdx.edu

All up to date code and documentation can be found on our Github repository:
< https://github.com/Chana030102/PSU_WowweeChimpProject >

See the Chimp before we took it apart:
https://drive.google.com/file/d/1BsuR7ykKg1hiLC07FX7NMMB13anGvhxI/view?usp=sharing

See how our motor control code performs:
https://drive.google.com/file/d/1TsBpUk2HEiPjYVWX9bM_xnceBTUk3X9j/view?usp=sharing

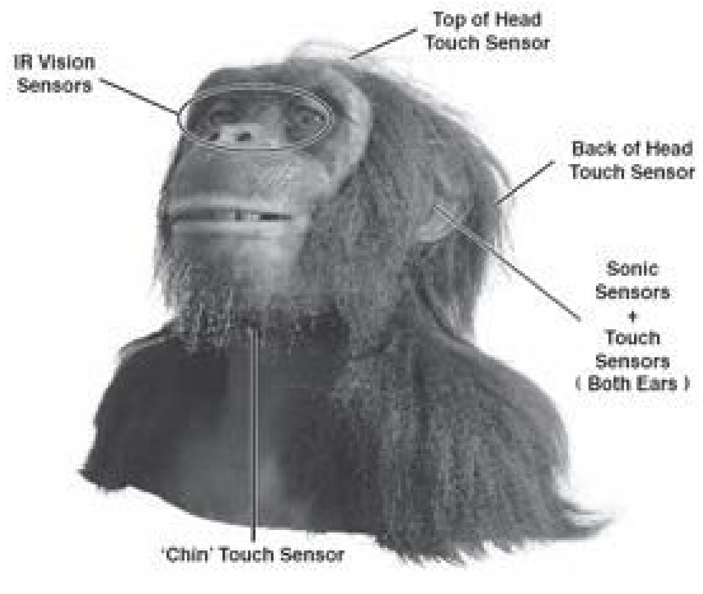
Project Introduction

The Chimpanzee on Wheels project consists of the Wowwee Chimpanzee and Turtlebot robots. The Wowwee Chimpanzee animatronic head has many motors and sensors and can react to external stimulus to perform some pre-programed animations. The Turtlebot is a robot that navigates rooms and moves around similar to commercial robot vacuums, but has a large frame and other parts integrated. The end goal was to interface both robots with one another to act as a single unit. Once combined, the unit could navigate rooms and play sounds and animations in response to certain events. Due to the lack of detailed documentation and learning curve that was presented to our team, **our main focus this term was exclusively working with the chimp head.** As such, there is no information about the turtlebot in this report.

About the Chimp

The Alive Chimpanzee is an animatronic chimp head produced by Wowwee Ltd. It has eight motors to emulate facial expressions as well as nine sensors that allow it to respond to external stimulus. Original modes included:

Alive: Reacts to sensor input, and performs autonomous actions.


Top of Head Touch Sensor
IR Vision Sensors
Back of Head Touch Sensor
Sonic Sensors + Touch Sensors (Both Ears)
'Chin' Touch Sensor

Direct Control: responds to user commands via controller.

Program: Up to 20 steps (movements and sounds) can be entered in a sequence that can be recorded and played.

Guard: Uses the vision and sound sensors to detect movement and play animations.

Demo: plays 2 or 3 random animations.

Sleep: Powers down after some time of no input.

Project Tasks

The following is the original list of tasks assigned for this project:

- Learn about existing software on Wowwee Chimpanzee
- Learn about existing software for the Turtlebot
- Attach Chimpanzee head to Turtlebot in a manner that will be easy to install and remove
- Integrate facial behaviors on the Wowwee Chimpanzee with motion/navigation on the Turtlebot
- Integrate controls from Elvis with controls on the Turtlebot. Maybe emulate the Wowwee Chimpanzee controller

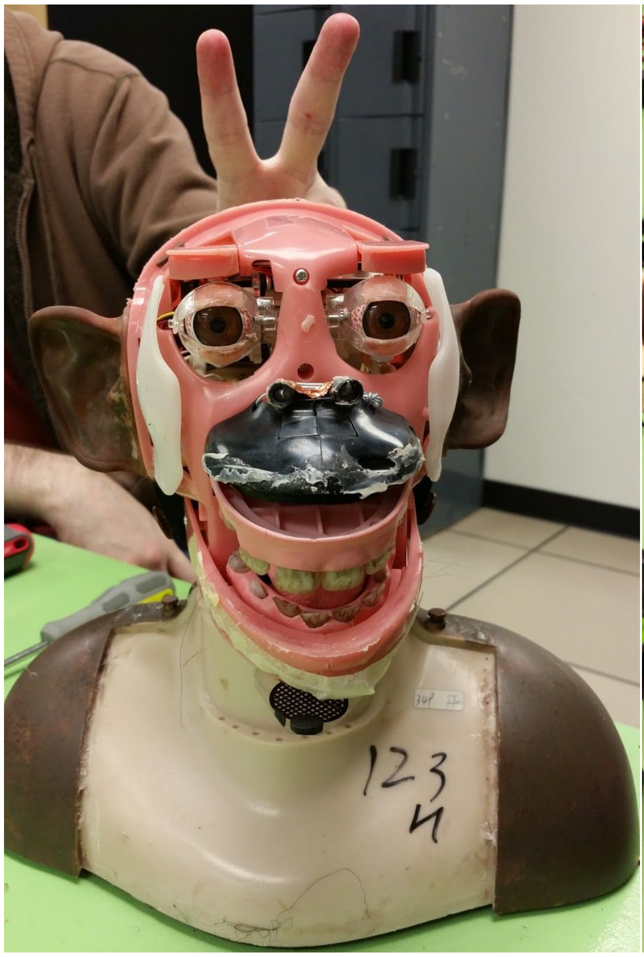
Due to complications of Assignment 1 and delaying of its deadline, only 3 weeks were left to work on this project. Below is an altered list of project tasks to adjust for the change in scheduling for the term.

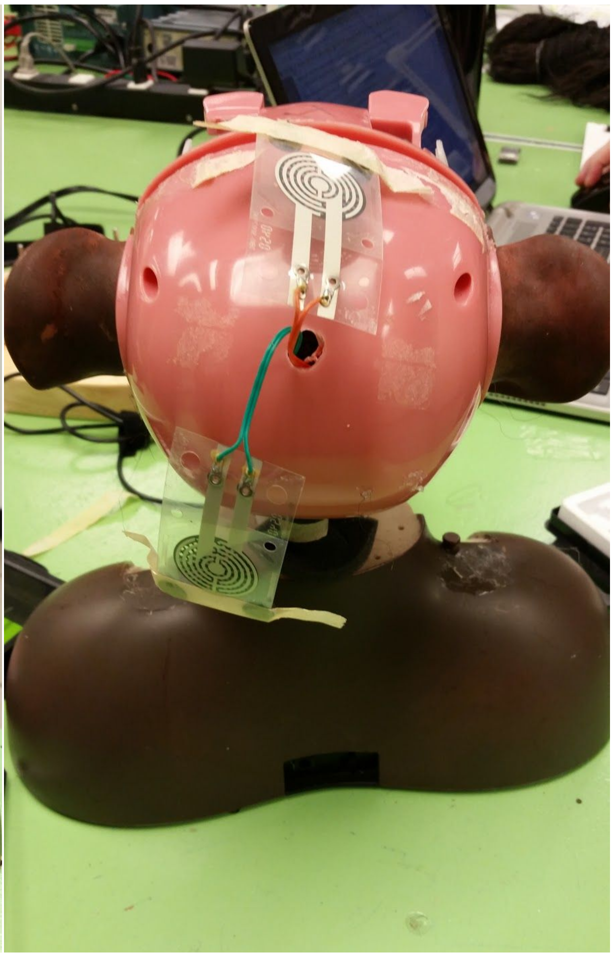
- Develop documentation of controls of motors and sensors
- Interface Chimp with Raspberry Pi 3 (RPi3)
- Code for static face expressions
- Code for sensors
- Program for Chimp to react to environment

<u>Project Progress</u>

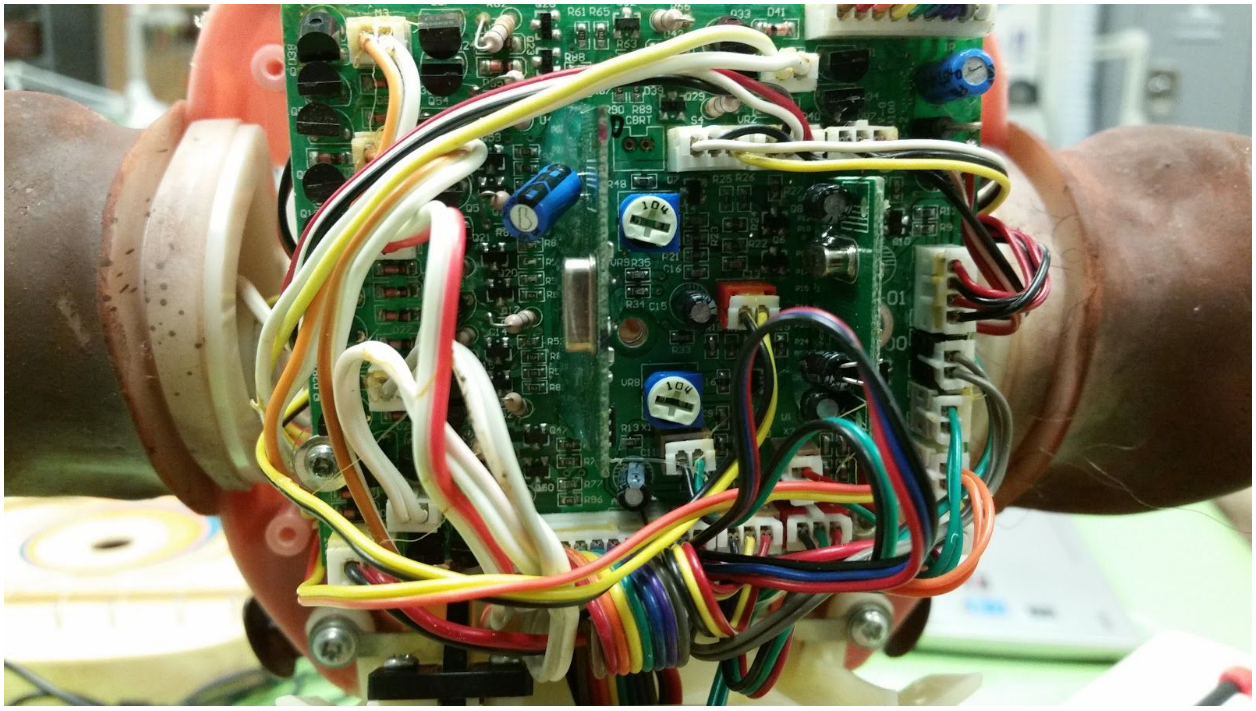
Starting the project

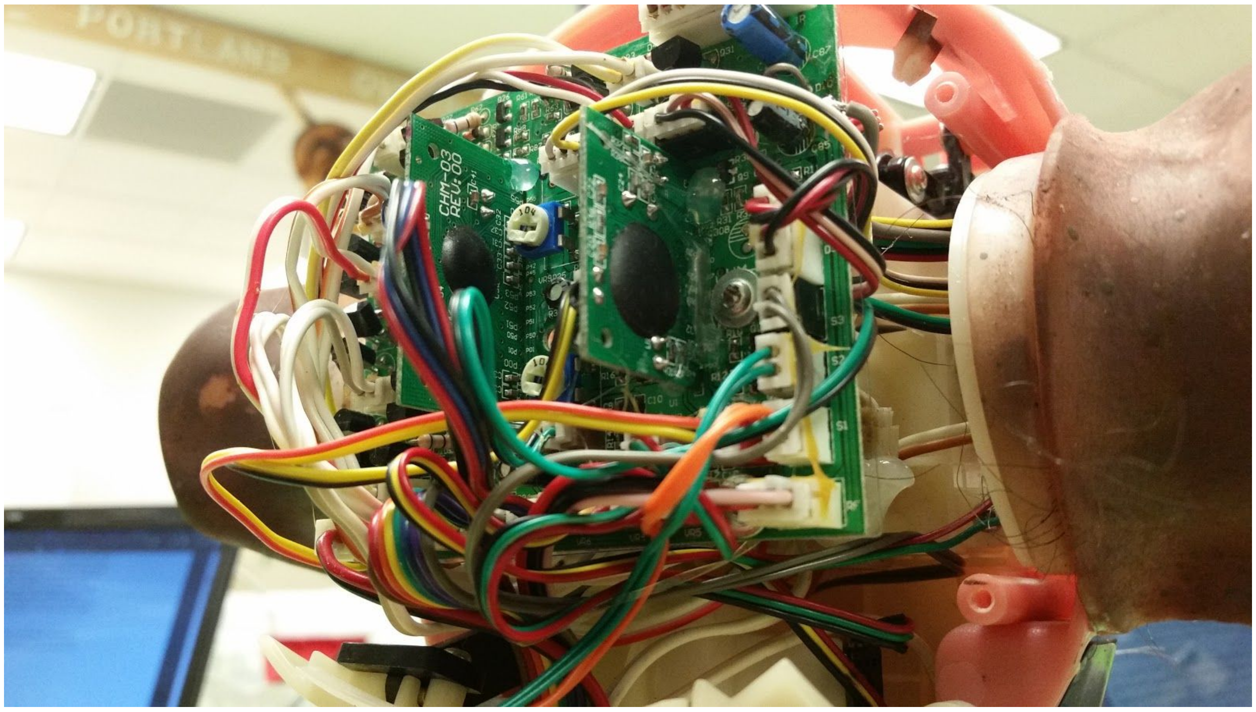
The first thing we had to do was remove the skin from the chimp so we could access the motors inside. The skin itself was glued on in several places and we had to carefully remove it with a cutting tool. Once removed, we could access the plastic casing and open the chimp up as seen in the images below. We identified touch sensor pads on the bottom of the chin (left) and the top of the head (right).



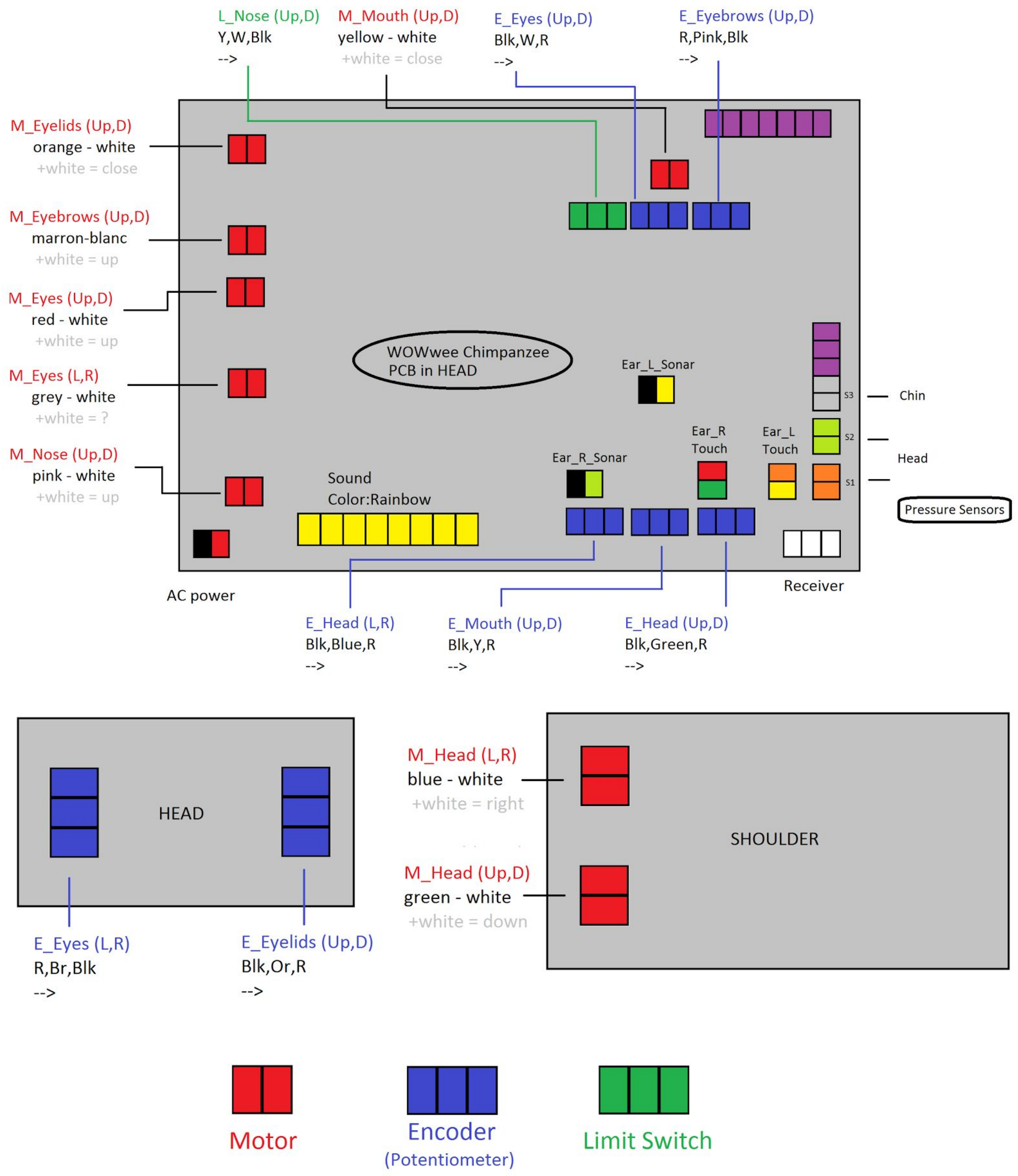


Next we unscrewed the back of the head and accessed the pcb inside (seen below). Thanks to the HackingLab diagram we were able to identify many of the pin connections on the board and label them. Some connections were not labeled so we took note of them to test and label later. Most of the connections were glued on and difficult to remove.



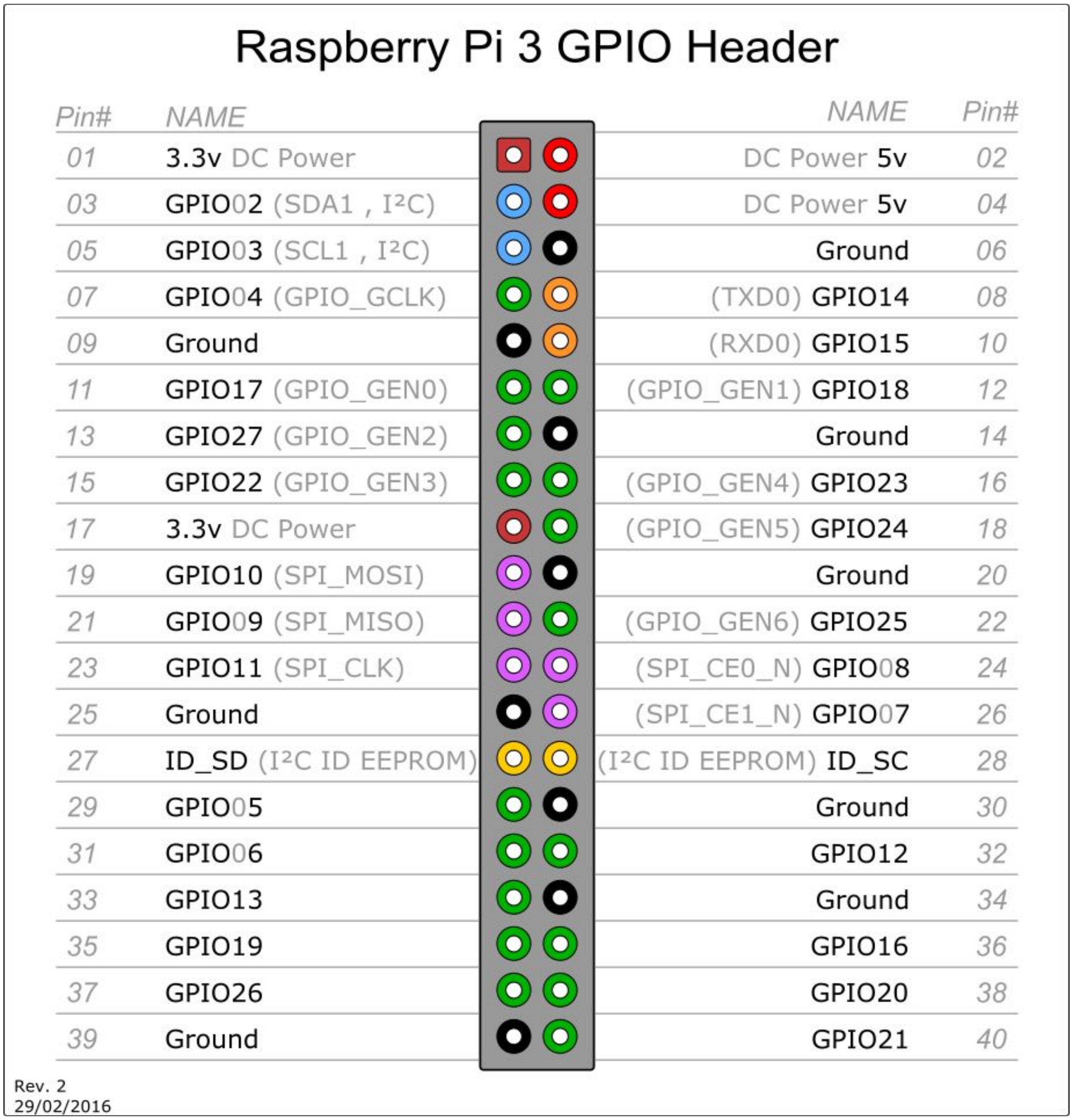


The updated diagrams below are the corresponding pin connections and their relevant motors, sensors, etc.



*Diagrams can be found in our Github repository

We decided to use the Raspberry Pi 3 Model B Vi 2 to interface with the motors and sensors since it would allow for more control of the robot.



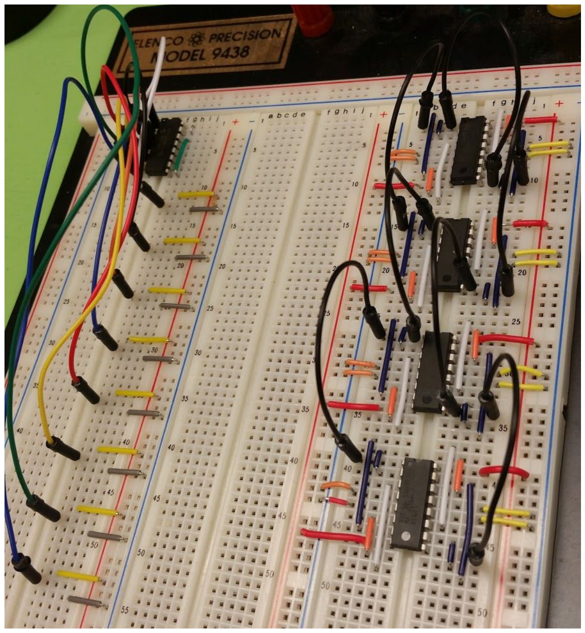
Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016

The GPIO pins number from 2 to 27, giving us a total of 25 GPIO pins to work with. Some GPIO pins overlap with pins that are for special interfacing purposes such as SPI and I2C.

Motor Control Method



The Raspberry Pi 3's pinouts do not output a high enough current to power the motors. To resolve this problem, we used L293D H-bridges to help with powering the motors. The DC motors also had rotary encoders to tell us the position of the motor, so we used a MCP3008 AD converter to interface with them. A total of 4 L293D chips (right side of board) were needed to drive the motors and 1 MCP3008 chip (upper left of board) was needed for the motor encoders. With this additional hardware, we controlled the motors using C functions that we developed over the course of the term using the wiringPi libraries.

For motor controls alone, we need 2 GPIO pins per motor, 1 GPIO pin for H-bridge enable, 4 GPIO pins for the AD converter, and 2 for the nose limit switch. This totals to 23 GPIO pins. H-bridges are powered with the RPi3 5V pin and the ADC and encoders are powered by the 3.3V pins.

We started by taking note of which color cable moves the motors in which direction. Below is a table for what we found for the Chimp we were working on:

Motor	**Color**	**Direction**	**Motor**	**Color**	**Direction**
Head LR	Blue	Right	Head UpD	White	Up
	White	Left		Green	Down
Eyes LR	White	Right	Eyes UpD	White	Up
	Grey	Left		Red	Down
Brows	Brown	Up	Nose	White	Up
	White	Down		Light Pink	Down
Mouth	Yellow	Open	Eyelids	* Possibly broken. Unable to get it to move	
	White	Close			

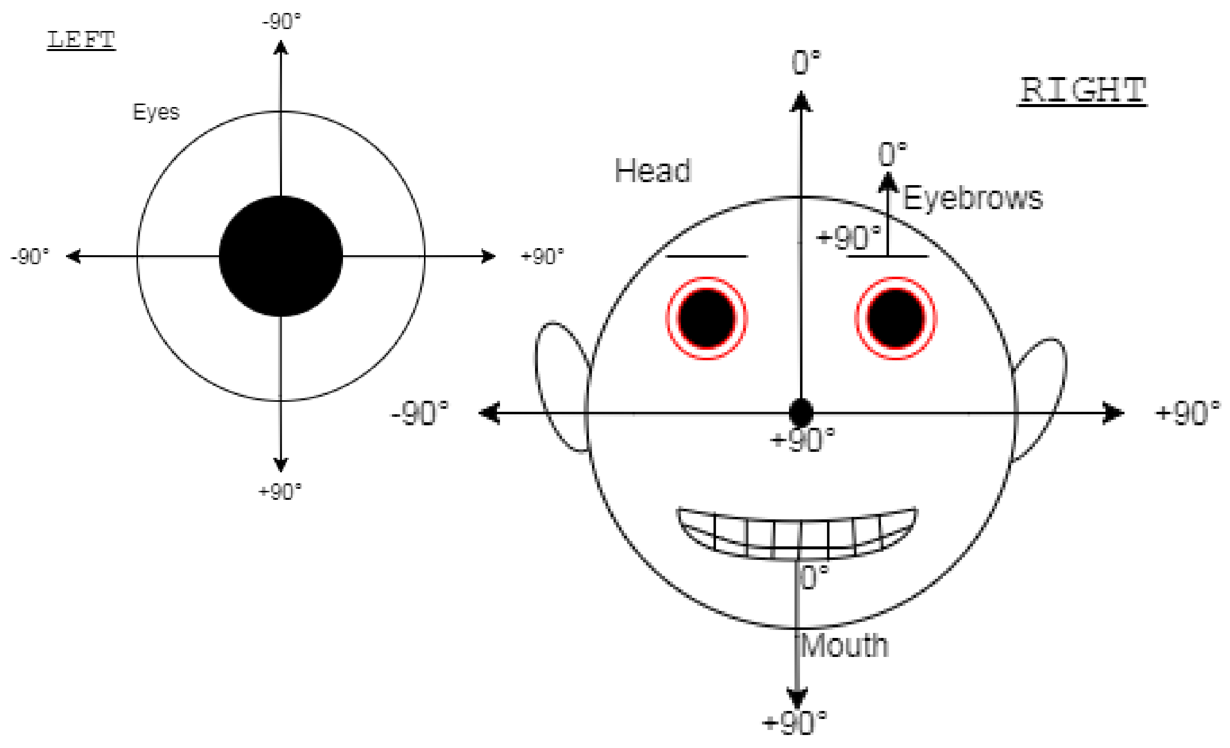
In order to know when to stop, we needed to use constantly read the encoder values and use them as reference. We found relevant values by moving the motors to the point where they can't physically move anymore and then got the encoder value for it. Every motor has two limit values or positions that the Chimp part can't move past. It is more reasonable to stop the motor BEFORE reaching that limit value, so limit values defined in our code will be in a smaller range than what's measured.

It is important to note that the encoders are powered by the RPi3's 3.3V pin. Using a different voltage than this will provide different values than what we will be noting.

*Limit values measured for each motor at the **unpowered position** after moving to max positions. Value used in code is arbitrarily determined.*							
Motor	**Direction**	**Value**	***in code**	**Motor**	**Direction**	**Value**	***in code**
Head LR	Right	746	760	Head UpD	Up	360	400
	Left	336	330		Down	772	740
Eyes LR	Right	627	600	Eyes UpD	Up	14	14
	Left	392	410		Down	31	30
Brows	Up	407	415	Nose	Up	* Uses limit switch.	
	Down	602	590		Down		
Mouth	Open	578	530	Eyelids	* Possibly broken. Unable to get it to move		
	Close	351	370				

With all that information, we were able to develop working motor control code. The functions currently take some value in degrees and moves the motor to that position. The degree inputs are set as the following for each motor:

Function	Degrees		
	-90	0	+90
head_UpD	▨	Face Upward	Face Forward
head_LR	Face Left	Face Forward	Face Right
eyes_UpD	Look Up	Look Forward	Look Down
eyes_LR	Look Left	Look Forward	Look Right
brows_UpD	▨	Move Up	Move Down
mouth_UpD	▨	Close Mouth	Open Mouth

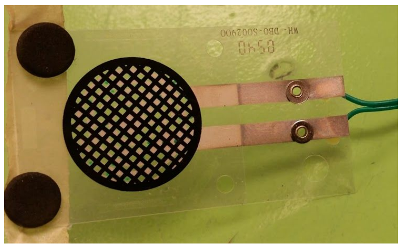


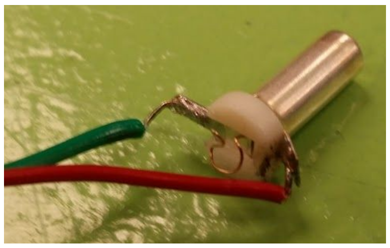
About the Sensors

We have identified that the Wowwee Chimpanzee has 4 kinds of sensors and found some possible online resources that give us a hint as to how they work:

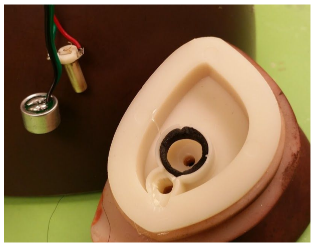
Sensor on Chimp	**Hardware**	**Link**
Touch Sensor (head and chin)	force sensitive resistor (FSR)	Link
Sonic Sensor (ear)	microphone	Link
Ear Touch Sensors	vibration switch sensors	Link
IR Vision (eyes)	*Difficulty identifying	

None of the sensors have been interfaced with the RPi3 and little testing has been done to figure out how they work. Tests haven't been very successful and more time is needed to understand them. It would be easier to buy new sensors to interface with the RPi3.


Touch Sensor


Vibration Sensor

Audio Sensor


Audio and Vibration Sensors for ear

Future Project Ideas and Suggestions

More and Better Functions

With the currently provided functions, it is possible to use a combination of them to make animations as we have done in the programs we have provided on Github. Changing the functions so that the input degrees are more consistent throughout the Chimp in what direction what degree represents would be ideal. There are many ways to tweak the C code that we have developed.

It is important to note that we mentioned that motor controls would require 23 GPIO pins, and we only have 25 GPIO pins available on the RPi3. In order to add more functionality such as sensors, GPIO expansion needs to be considered and pin definitions for motor control will need to be changed as to free the I2C pins.

Interfacing with Sensors

As we mentioned above, there was not much time left to work with sensors, but it is definitely possible. Figuring out how the old sensors work would be wonderful, but it is much easier to get new sensors that have proper documentation. There is also the possibility the old sensors are damaged as they did not seem to work well when we tested the Chimp's functionality before opening it up.

Since we are interfacing it with a RPi3, it is now more flexible and we can add other sensors that the Wowwee Chimp didn't come with. Light and temperature sensors would be good additions.

Once sensors have been interfaced, the Chimp will have more ways to interact with its surroundings and machine learning algorithms can be applied.

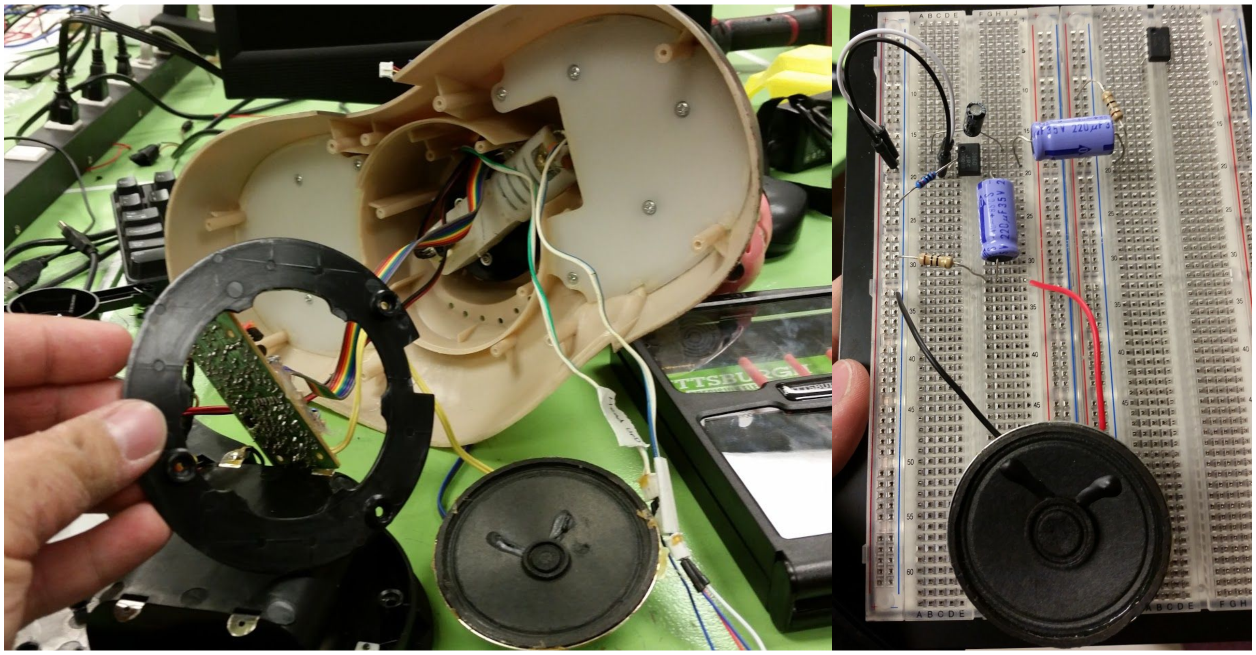
Sound

This is another feature we did not have enough time to fully implement. Once sound is implemented, it will be possible to play audio files and even have the Chimp move its mouth along to songs to make singing animations. Talking animations for human interactions would also be much easier since we are using the RPi3.

We did manage to identify key hardware in the Chimp for sound though. The Chimp has a 3 inch 8 Ohm, 1 Watt passive speaker and a dual low voltage power amplifier (part number UTC2822D).

We have tested getting sound from the audio jack on the RPi3 and successfully hooked that up to a LM386 audio amp feeding into a 8 Ohm 0.5 Watt passive speaker. The sound volume is extremely low and more research is needed to try to get more volume.

Original speaker (left) and attempt to wire one on board (right)



Development Process Section

This section is to reference resources that we used that aided us in the development of the project. There is also a design log that we wrote in as we worked on the project throughout the term. It includes plans we had for the project, problems we ran into, and any solutions we came up with.

Useful Links

Our Github Repo for the Chimp: https://github.com/Chana030102/PSU_WowweeChimpProject
HackingLab: http://www.hackinglab.org/chimp/index_chimp.html

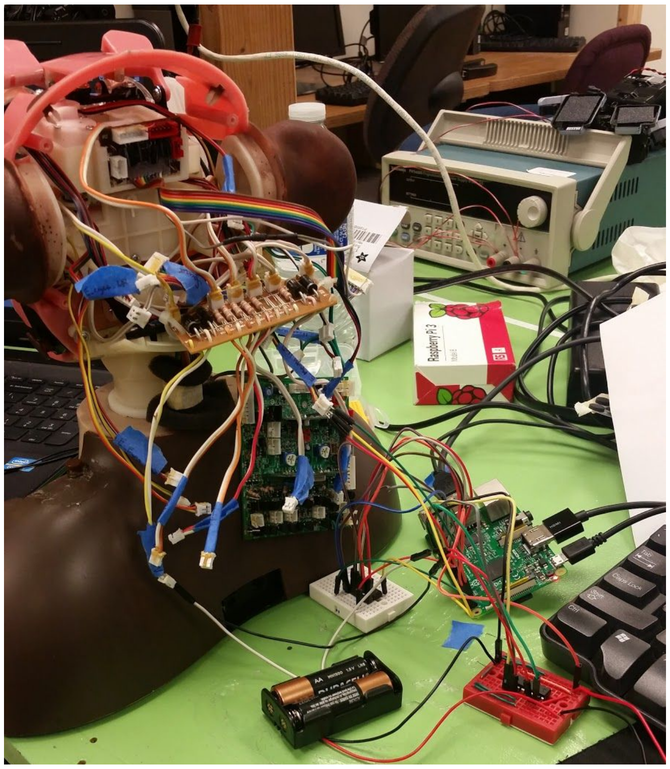
Audio Amplifiers:
http://www.instructables.com/id/LM386-Audio-Amplifier/
http://www.rason.org/Projects/icamps/icamps.htm

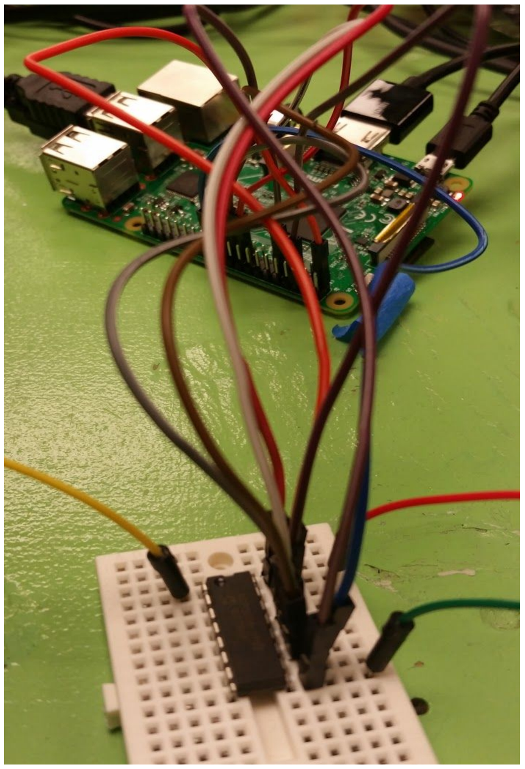
Good resource for how to hook 3.5mm jack to speaker:
https://www.youtube.com/watch?v=AqUZb2vBjis

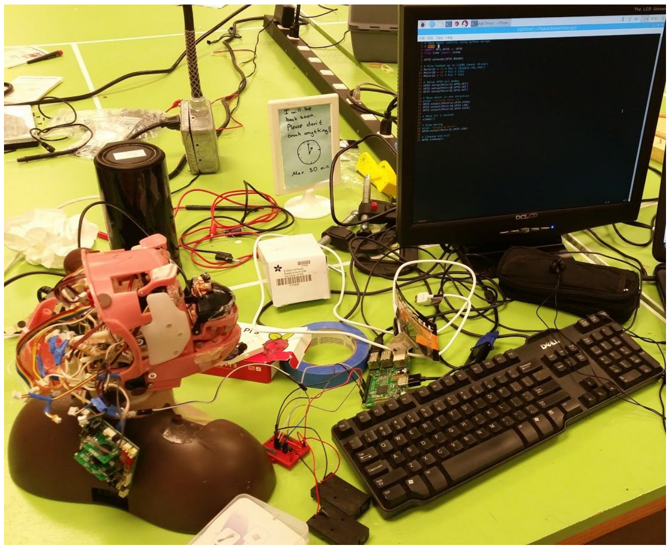
GPIO expansion:
https://learn.adafruit.com/mcp230xx-gpio-expander-on-the-raspberry-pi/hooking-it-all-up

References for Animation movements:
https://www.youtube.com/watch?v=2r-wv9eAyQI
https://www.youtube.com/watch?v=XAlvmr0OIv4
https://www.youtube.com/watch?v=-vvt9OBli6A

Misc Pictures







Design Log

2017-10-25 (Week 5)

- "Skinned" the chimpanzee to access to the head and shoulder's insides.
- Observed what boards are in the head and shoulders.
- Identified cables for motors and sensors with assistance of Hack Lab's outlines
- Made a list of things to do/consider for next session:
    - Remove battery compartments in chimpanzee
    - Remove receiver (missing remote anyway)
    - Difficulty removing screws to remove PCB board
    - Unidentified board with lots of resistors in head
- Bring the following items:
    - Development board that may be compatible w/ this project
    - Velcro strips
    - Masking tape
- **MINI-GOAL:** figure out how to move motors and communicate w/ sensors

2017-10-26

- Successfully took PCB board out and replaced screws for easier removal
- Labeled motors and encoders (everything from diagram from hacking lab)
- Successfully moved motors with power supply, each line moves in different direction
- Identified other wires (sound, separate for upper motors, sensors)
- Consider using RPi3 for compatibility with possible future projects
- NOTE: currently have L293D

<u>To do:</u>

- Higher resolution and better diagram of board mappings
- Figure out how to write and control from a development board (Nucleo, RPi3)
- Understand motor drivers (L293D)
- Possibly figure out sensors on chimpanzee

Plans for next meeting:

- Attempt controlling motors with development board (RPi3)
- Figure out how to communicate w/ sensors and encoders

2017-11-3 (Week 6)

- Took apart more glue to get access to more wires from sensors inside head
- Identified more sensors (ears, chin)
- Setup microSD for RPi3. Changed settings so that we can work with it. Also updated and installed Git and Vim.
- Trouble with setting static ip for direct ethernet connection

2017-11-8 (Week 7)

- Searched for tutorials to get a general idea of what signals to send to a motor driver to drive DC motors, and also some way to interpret analog signals using the RPi3
- Found a tutorial for DC motor control with motor driver and successfully implemented it
- Found a tutorial for ADC but was not able to receive a signal from encoder. Accidently applied 5.5V which may have been the problem. Next time try 3.3V first, then 5V.

2017-11-9

- Successfully connected 3.3V to encoder and got readings. Using hardware SPI for RPi3
- Started recording values for positions of motors. Some relabeling was done for wires. Eyelid motor does not seem to be reacting to power.
- Somewhat successful test python script written to open and close mouth with testing values of motor position. Needed some delays in between opening and closing before testing and changing signals.
- One L239D only controls two motors, so a total of 4 L239Ds will be necessary to control all 6 motors. Large numbers of pins will be taken over, so we need to consider using a multiplexer of some sort to lower the number of pins being used on the direct RPi3
- One MCP3008 can take 8 channels and will cover all 8 motor encoders.

2017-11-10

- Tried controlling multiple motors using “limit” values from encoders, but failed. Maybe due to not connecting battery pack for power, but was not able to determine because we had taken it apart already.
- Searched for C libraries we could use instead of Python
- Extremely messy wiring method and difficult to work with and re-create every time we go into the lab. For the time being, we will complete other tasks for the project while we wait for an order of female pin connectors to arrive. These will take advantage of the male pin heads already used in the Chimp’s wiring.

Due to the little time left for the project, we evaluated how much we could get done and that we could probably only figure out how to control the functions of the robot:

- Determine Motor Controls

- Investigate and play around with how the smooth movement controls are
- Make a few scripts to demonstrate emotional face expressions
- Determine how to interface with sensors

Things to do while we wait for connectors:
- Schematic for wiring of Motors and ADC
- Search for C libraries (or develop C library for basic needs)
- Make a C program to test with for next time we meet
- Update documents and Github repo
- Find documentation or user manual for RPi3
- Find documentation for processor on RPi3

2017-11-16 (Week 8)
- Applied new labels to cables, this time with two different colors to different motors from encoders. Also, used better ink that shouldn't fade as fast as previous pen ink.
- Started making extensions for easier wiring in the future to speed up setup time for coding.

2017-11-17
Finished making extension cables with connector heads, variations of 2 pin and 3 pin.

2017-11-18
- Continued searching for C libraries to use.
- Found that extension cables are having trouble providing signals to ADC. Possibility that they are too long. Will need to experiment with them another time.

2017-11-19 (Week 9)
- NOTE: extensions are poorly made and do not send signals properly even when shortened. Switched over to using purchased jumper cables. EyesLR was thought to be broken but is working with jumper cables.
- Successful test program that uses encoder value to test motor position and switch directions.
- Motor position limit values need to be modified, or a function to dynamically find 'limit values' are needed.
- C libraries from Gabriel Perez-Cerezo weren't fully compatible with all versions of pi.
- Developed RPi3 version of Gabriel Perez-Cerezo's GPIO library, but after testing it was very unstable. GPIO library causes segmentation faults when using gpio_setup but not all the time. Unable to determine what causes it. The MCP3008 library also does not work

and actually messes pin settings. After using it, the working Python version does not work either.
- Gave up on developing libraries and started using a known working library (wiringPi).

2017-11-21
- wiringPi versions work and reads MCP3008. Started developing libraries for Chimp motor controls
- Almost burned out motor driver by constantly writing high to it. Need to make sure to write low when you don't need it or when done using.

2017-11-29 (Week 10)
- For some reason, library for chimp is writing high to everything. When setting it to low, it keeps it low and doesn't write high when programmed to.
- Discovered that setting pins to output was missing from setup. This has now been added and it works successfully! Now to work on making static face expressions and finalizing the library.

2017-12-07 (Finals Week)
- Successfully connected multiple motors and moved them all. There is difficulty moving more than one motor at a time, but after looking at the videos of the Chimp's movement, it does not move multiple motors at once at any given time.
- A problem we are running into is that either voltage or current output is not high enough on the motor drivers, resulting in slower movements than the original Chimp.

2017-12-08
- Tested ear sensors based off datasheets that seem to be for these sensors. Difficulty getting expected signals based off the datasheets. There is a need to look at how circuitry is wired up for to the sensors in the Chimp for further research. May also consider improved or new options to replace them.
- IR sensors have not been figured out and will need to be researched more.
- Made small animations and recorded them for the deliverables. The programs for these animations will need to be renamed and also commented before turning in the report.