Date: 9/15/2020

Goals

Long Term:

- Design and building a robotic arm which can be controlled by voice commands

Short Term:

- Discuss and finalize a working project idea
- Complete and submit a project proposal
- Begin visualizing a timeline for when each component of the project should be prioritized

Progress

Today's Progress:

Today, our first and foremost priority was deciding on a project idea. After some discussion with each other, we settled on building a voice controlled robotic arm with the ability to visually detect objects. We wanted to make sure that our project was doable, we spent a little time researching how to conduct speech recognition in Python to get an overview of what skills we would need to learn. As it turns out, there are several packages available in the Python Package Index (PyPI) which can be installed to conduct speech recognition.

We also had conversations with Mr. Levin and Mr. Pandya, about our project idea and they both seemed to be on board with it. However, as Mr. Pandya pointed out, this is a fairly ambitious project, so we broke our project down into three major parts and discussed when each component should be completed. Here was our breakdown:

1. Voice Recognition → Code
    a. We decided that voice recognition should be one of the first things that we should focus on for a couple of reasons. For one, neither of us have any experience working with speech recognition and we speculate that it could have quite a steep learning curve, so we figured that we should allow ourselves sufficient time to familiarize ourselves with the process. Additionally, as we are working remotely at the moment, it would be easier to work on the coding aspects of the project which we can collaborate on through Github and other online platforms.
2. Image Recognition → Code
    a. We decided that image recognition should also be the next thing that we should focus on for a couple of reasons. For one, Ashton does not have any experience working with openCV or image recognition and the process of learning about the topic will take some time, so we figured that we should

designate a sufficient amount of time to complete this portion of the project. Again, as we are working remotely at the moment, it would be easier to work on the coding aspects of the project which we can collaborate on through Github and other online platforms.

3. Physical Build → Mechanics
    a. We decided that we should hold off on any physical building for now for a couple of reasons. For one, as we have both been involved in the building portion of our FTC robot for the past couple of years, this is the portion that we feel most familiar and comfortable with working on, and we think that therefore it may take less time to complete than the other two aspects. This said, we plan on coming up with a design for our physical build in order to get an idea of what we will be working with as well as a list of the parts we will need to build our arm. It will be necessary to get this done early on as we will need to get our parts together before we can begin to build.

Once we decided on this general timeline for our project, we moved on to writing our project proposal which we have provided in the *Evidence of Progress* section.

Evidence of Progress:

Sites Visited for Research on Speech Recognition:

- https://realpython.com/python-speech-recognition/#picking-a-python-speech-recognition-package
- https://www.geeksforgeeks.org/python-convert-speech-to-text-and-text-to-speech/

Project Proposal:

Project Title: Voice Controlled Robotic Arm with Image Recognition

Abstract:

For our project, we will be designing and building a robotic arm which we can control through voice commands. As we have been learning about mechanical design and software development in FTC as well as our S&E classes throughout high school, we wanted to see how we could combine multiple topics to create an assistive device. We were inspired to do this project as we wanted to help people with disabilities, such as arthritis or blindness, who may have difficulty recognizing and picking up objects. In the end, our goal is to have created a robotic arm that processes speech and implements visual recognition to pick up a specific object that it has been commanded to pick up.

In order to carry out our project, we will first need to learn how to conduct speech recognition. Though speech recognition software already exists, we have decided that we want to take on the challenge of learning how to design and program our own software as this will help us to better learn how the technology works. Second, we will need to learn how to implement computer vision for object detection. We want to learn how to train a machine learning model to detect objects within images as this will be key for the arm to select the right object to pick

up. In order to do this, we will be researching how to write a machine learning algorithm as well as the implementations and uses of openCV. Lastly, we will be designing and building our physical arm. In order to do this, we will need to determine which types of components we will need to carry out our project. We will want to have a camera for the image recognition component as well as motors for the movement aspect and some sort of clamping mechanism in order to grab the desired object. After taking all of this into consideration, we will want to also decide the number of degrees of freedom the arm will need in order to reach the necessary range of motion.

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 9/17/2020

Goals

Long Term:	**Short Term:**
- Create the speech and visual recognition code that can be tested once we get hands on material	- Research the SpeechRecognition Python package - Write some simpler test programs to practice and learn how to use SpeechRecognition

Progress

Today's Progress:

Today, we focused on learning about the SpeechRecognition package available in the PyPI. As this is new material for us, we spent some time reading through some documentation (SpeechRecognition 3.8.1, SpeechRecognition library reference) and following a tutorial explaining the setup as well as how some of the basic functions are used. Through this tutorial, we both installed SpeechRecognition 3.8.1 (Ashton also had to install Python 3.8.5), and then began to write a program where we created a recognizer instance, captured data from an audio file, and used the default Google Web Speech API to process the audio data and print out the transcribed speech. As there were a few other options for APIs which we could use to process the audio data, we plan on looking into advantages of each of those to see if using one of them would be a better choice. Lastly, we created a Github repository to share our code with each other as well as make it more accessible for review by Mr. Levin and Mr. Pandya.

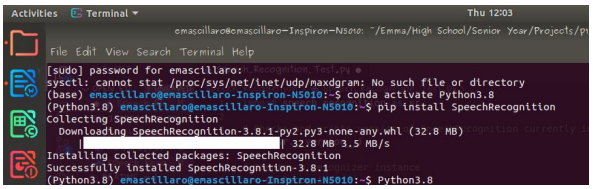
Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

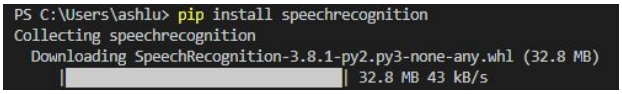
Sites Visited for Research on Speech Recognition:

- https://pypi.org/project/SpeechRecognition/
- https://github.com/Uberi/speech_recognition/blob/master/reference/library-reference.rst
- https://realpython.com/python-speech-recognition/#picking-a-python-speech-recognition-package

SpeechRecognition Install:

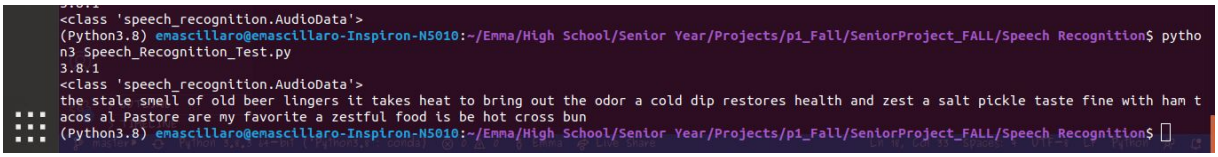


(Emma)



(Ashton)

SpeechRecognition Test Code Output:



Signatures

Emma Mascillaro

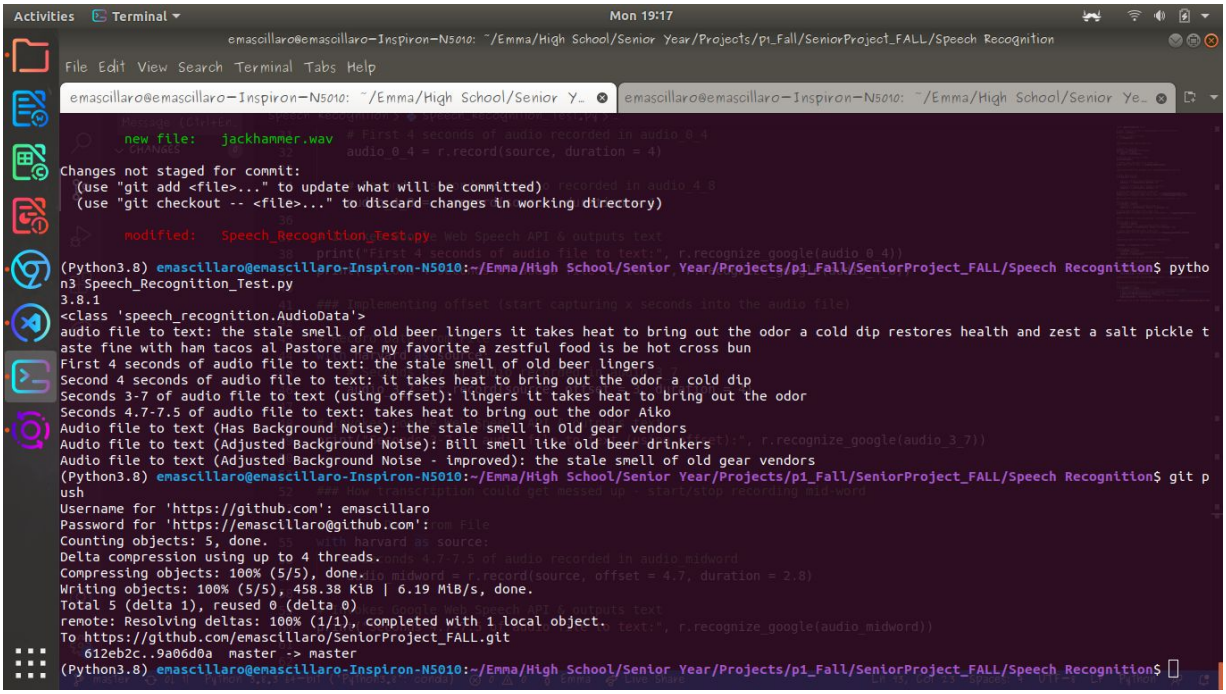
Ashton Lukyanovsky

Date: 9/18/2020

Goals

Long Term:	**Short Term:**
- Complete the speech and visual recognition code which we can test once we get our hands on material	- Continue learning about the SpeechRecognition Python package - Continue working on simpler test programs to practice and learn how to use SpeechRecognition

Progress
Today's Progress: Today, I continued to work through the Python speech recognition tutorial and completed the next two sections of it, covering audio transcriptions from portions of a file and dealing with background noise. As it turns out, these are two important issues to consider, as they have the potential to greatly reduce the accuracy of an audio transcription. For one, cutting an audio file off mid-word causes the API to try to transcribe only some of the syllables of the cut-off word, significantly increasing the chance for the transcription to be wrong. In one example I tested, I cut off recording the audio file in such a way that "akes heat" was recorded instead of "takes heat". When the Google Web Speech API was invoked on this recording, the transcription, which had previously been accurate, was "Mesquite". Excessive background noise, as I found, can also reduce the accuracy of the transcription. To visualize this, I ran my test program on an audio file with a jackhammer sound in the background, and as a result, what should've transcribed to "the stale smell of old beer lingers" instead transcribed to "the stale smell in Old gear vendors". In order to improve this transcription, I used the adjust_for_ambient_noise() function, but as you can see, this caused the first "the" to get cut out. This happened because the adjust_for_ambient_noise() function, by default, uses the first second of the recording to gauge the noise level of the audio, and, as a result, loses that second of data. To get the "the" back at the beginning of the phrase, I decreased the time frame for the function to analyze the noise level to 0.5 seconds. Though helped me get the "the" back at the beginning of the phrase, the "beer" that had been corrected previously (with the 1 second analysis period) went back to "gear" as the function had less data available to gauge the noise level. This all said, it is definitely important to keep in mind the effects of background noise on transcription, but for the scope of this project, I suspect that I will not have as many issues with background noise interference as the audio I will be using for testing will not have background noise as extreme as a loud jackhammer.
Evidence of Progress: ***Github Link:*** https://github.com/emascillaro/SeniorProject_FALL ***Speech to Text from Portions of an Audio File & With Background Noise:***



Signature

Emma Mascillaro

Date: 9/22/2020

Goals

Long Term:	**Short Term:**
- Complete the speech and visual recognition code which we can test once we get our hands on material	- Continue learning about the SpeechRecognition Python package - Continue working on simpler test programs to practice and learn how to use SpeechRecognition

Progress

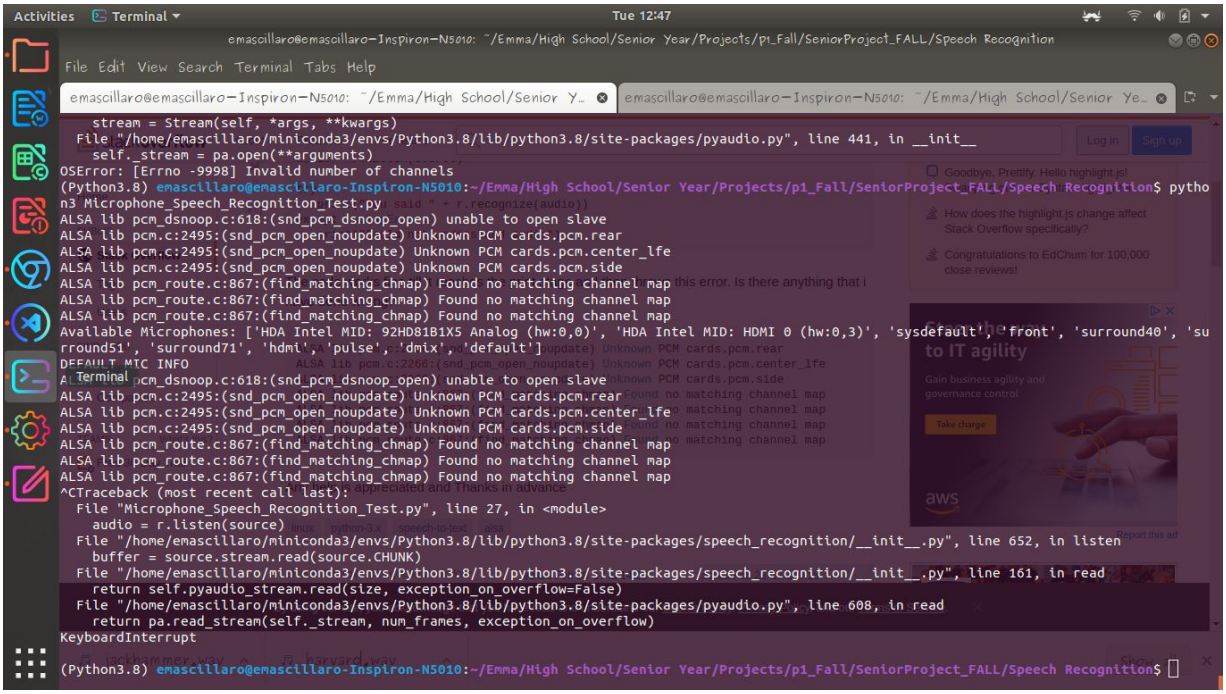
Today's Progress:

Today, we continued to work through the Python speech recognition tutorial, and worked on the next section of it covering audio transcriptions from microphone input. In order to work

with microphone input, we installed the PyAudio package which provides Python bindings for PortAudio (An open-source computer library for audio playback and recording), allowing us to use Python to play and record audio on multiple different platforms. We then were able to begin recording microphone input from our laptops. In our test program, we decided to use the default microphones on our laptops, but depending on the device that we use to run the speech recognition code for our final project, we may need to specify which microphone we will be using. On Emma's computer, functions such as listing the available microphones and setting the default microphone as the source were causing some unexpected output to come up, all beginning with "ALSA lib ...", so she spent some time looking into what those messages meant. As it turns out, these messages had nothing to do with SpeechRecognition or PyAudio. ALSA, a kernel based system that comes installed with Ubuntu, is used to connect sound hardware to a computer's operating system, and will throw these messages if an error handler is not defined in code. In reality, these messages are typically just a nuisance and shouldn't affect the functionality of the code itself. For our final product, we will need to process audio in real-time through a microphone.

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL



Signatures

Emma Mascillaro	*Ashton Lukyanovsky*

Date: 9/24/2020

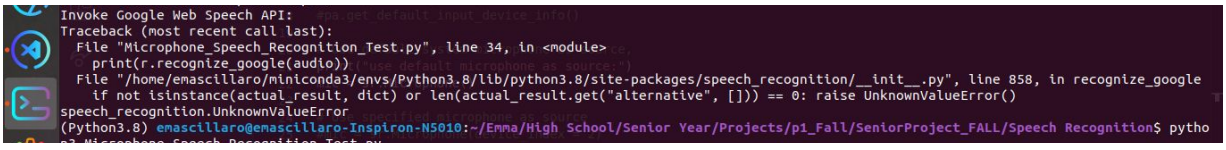
Goals

Long Term:	**Short Term:**
- Complete the speech and visual recognition code	- Continue learning about the SpeechRecognition Python package - Finish working on simpler test programs to practice and learn how to use SpeechRecognition - Begin looking into image recognition

Progress

Today's Progress:

Today, we finished working through the Python speech recognition tutorial, completing the section of it covering audio transcriptions from microphone input. In order to capture audio from the microphone, we use the r.listen(source) function with the microphone as the source. When we used this function with an audio file as the source, recording would stop at the end of the file, but with microphone input, it works a little differently. As long as the microphone is picking up sound, recording will continue, and will only stop once the microphone no longer is picking up any audio. Because of this, it is important to check the system's microphone sensitivity and set it to a point where it can recognize speech without picking up on background noise. This was an issue with Emma's computer in the beginning as her microphone was picking up on her computer's fan and therefore was getting stuck on recording audio. We also used the r.adjust_for_ambient_noise() function to filter out any background noise that may not have been accounted for in the microphone settings. This function works in the same way it did for the audio files, capturing a small portion of the file and adjusting the sound levels based on it. Lastly, in order to handle exceptions with unrecognizable audio input, we utilized a try and except block. Doing this prevented output such as the following to come up:



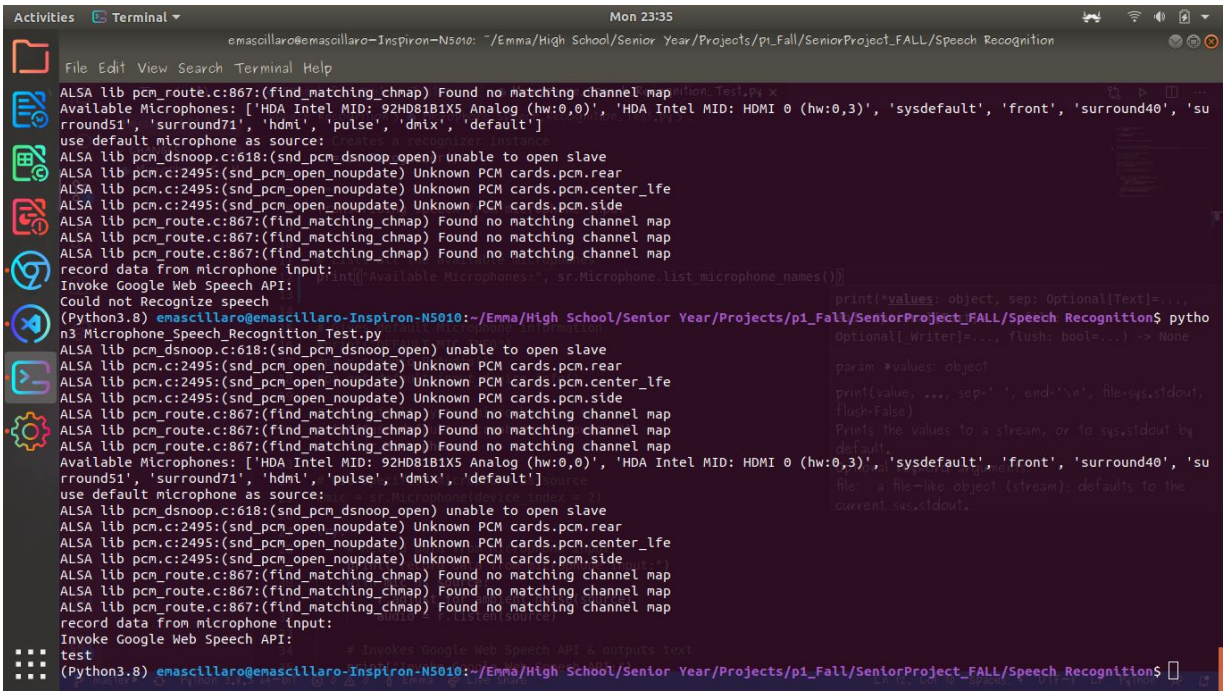
And replaced it with:

After completing work with SpeechRecognition, we moved on to brainstorming how we could go about image recognition. We decided that our two options would be to either use an image recognition API such as Google's Vision API, or to write and train our own machine learning algorithm. By using an API, we will be able to complete the image recognition portion of our project in less time and may have more accurate readings, but by writing and training our own machine learning algorithm, we will develop a better understanding of image recognition as a whole and learn more throughout the process.

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Speech to Text with Microphone input (Spoken word → "test"):



Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 10/1/2020

Goals

Long Term:

- Complete the object detection program
- Integrate our Speech Recognition and Image detection programs
- Implement said programs to function with the robotic arm

Short Term:

- Learn about Machine Learning & Neural Networks
- Write ML test programs to learn and practice object detection
- Familiarize ourselves with the mechanics of the arm and its controls

Progress

Today's Progress:

Today, we began doing background research on object detection with Python. We read through a few articles about image recognition with machine learning, using Python, and we discovered that there are three primary object detectors: R-CNN (and their variants), Single shot Detectors (SSDs), and YOLO (stands for "You Only Look Once"). Within these types of object detectors, there are two types of detector strategies, one-stage (single-stage) and two-stage, and each has its own benefits and drawbacks. To begin, the first type of detector strategy, two-stage, is what the R-CNN object detector uses. A two-stage detector strategy requires an algorithm such as Selective Search (or an equivalent algorithm) to suggest possible bounding boxes that may contain objects. These regions are then passed to a Convolutional Neural Network (CNN) for classification - this is how some of the first deep learning based object detectors worked. Two-Stage Detector Strategies for this reason tend to be more accurate, but are much slower than their Single-Stage counterpart. Single-Stage detector strategies, the strategy that YOLO uses, on the other hand, treat object detection like a regression problem, taking a given input image and simultaneously learning bounding box coordinates and corresponding class label probabilities. Therefore, these Single-State strategies tend to be much faster at the cost of some accuracy. For the scope of our senior project, we plan on using YOLO (Uses a single-stage detector strategy). We believe that this is a decent choice for us since we don't plan on having objects cluttered in the frame of the camera.

Today, we also spoke with Mr. Levin about beginning to learn about stepper motors and stepper motor drivers. He explained that the coils inside of a stepper motor are inductors, and that as we apply current to it, we get corresponding voltage. In our case, our stepper motors require 12 Volts in order to work. Our concern with this is that the current through the coil and our voltage in driving it created by our driver circuit is often larger at the start than the rate of voltage, meaning that the voltage must be dropped in order to maintain current. This is where the stepper motor driver comes in. A stepper motor driver's job is to provide a significantly

higher voltage and then to drop it in order to get current up and then maintain it, and they work by sending current through various phases in pulses to the stepper motor. In the near future, Mr. Levin suggested that we take a deeper look into stepper motors & drivers, suggesting in particular that we look at the allegro stepper driver a3967 & Big Easy Driver as well as get our information directly from the manufacturers.

Evidence of Progress:

References for Object Detection Research:

- https://towardsdatascience.com/image-recognition-with-machine-learning-on-python-image-processing-3abe6b158e9a
- https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/
- https://arxiv.org/pdf/1804.02767.pdf

Stepper Motor Driver Suggestions:

- https://www.sparkfun.com/products/12859
- https://www.google.com/search?q=allegro+stepper+driver+a3967&oq=allegro+stepper+driver+a3967&aqs=chrome..69i57j0l4.13138j0j7&sourceid=chrome&ie=UTF-8

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 10/6/2020

Goals

Long Term:

- Complete the object detection program
- Integrate our Speech Recognition and Image detection programs
- Implement said programs to function with the robotic arm

Short Term:

- Learn about Machine Learning & Neural Networks
- Use the YOLO object detector to write test programs to learn about and practice object detection
- Familiarize ourselves with the mechanics of the arm and its controls
- Learn about stepper motors and determine the extra materials needed

Progress
Today's Progress: Today, we continued researching Object Detection with Python, today focusing on getting a better understanding of YOLO. We watched this video which discussed the history of object detection, starting with facial detection (first developed in 2001) and ending with a description of YOLO (11:10) as well as a walkthrough through a sample program. From this video description, there was a link to the developer's github repository where we found further readings about YOLO object detection as well as a setup guide in order to get started with working on our own project. There were some downsides to the structure of the code in this repository, such as the fact that Darknet and Cython (Cython being a programming language aiming to be a superset of the Python programming language, designed to give C-like performance with code that is written mostly in Python with optional additional C-inspired syntax.), meaning that it could get complicated for programmers who are new to image detection, but, in particular, we liked the fact that there was a quick guide on how to train on our own dataset as well as use video/live camera feed, so we decided that we would try to go with this model.
Evidence of Progress: ***YOLO Github Repository:*** https://github.com/llSourcell/YOLO_Object_Detection ***Github Link:*** https://github.com/emascillaro/SeniorProject_FALL
Signatures
  ***Emma Mascillaro*** ***Ashton Lukyanovsky***

Date: 10/8/2020	
Goals	
Long Term: - Complete the object detection program - Integrate our Speech Recognition and Image detection programs	**Short Term:** - Learn about Machine Learning & Neural Networks

- Implement said programs to function with the robotic arm	- Use the YOLO object detector to write test programs to learn about and practice object detection - Familiarize ourselves with the mechanics of the arm and its controls - Learn about stepper motors and determine the extra materials needed

Progress

Today's Progress:

Today, we began working through the setup guide in the README file on this Github Repository. In the guide, it mentioned "weight files", which we weren't sure what they were, so we looked it up and found that Neural Network Weights control the signal (or the strength of the connection) between two neurons. In other words, a weight decides how much influence the input will have on the output, and oftentimes, these weights are contained within the hidden layers of the neural network. Once we began working through the setup directions, Emma began to run into a lot of issues with the dependencies having incompatible versions. As Emma is currently using a conda environment with Python3.8 on her laptop, the first issue she ran into was that Python 3.8 support requires TensorFlow 2.2 or later, meaning that Tensorflow 1.0 was available for installation. In order to try to resolve this, Emma created a new conda environment for Python 3, but she still ran into issues with not having the correct version of Tensorflow available in her apt. After doing some research, she read that she would have to downgrade Python again to Python 2.7 in order to be able to install Tensorflow==1.0.0, but the other dependencies (numpy and opencv3) required her to have at least Python 3. However, after more reading, she read that the issue was with her computer's CPU and therefore she was left with two options - either to downgrade to Tensorflow 1.5, or to build tensorflow from source. Again, however, since Emma can't downgrade to Tensorflow 1.5 while using Python 3, she will be working on building it from source (it looks like someone else had this issue as well and also built from source, so we will look at how they went about building from source)

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Issues with Tensorflow Installation:

Signatures

Emma Mascillaro	*Ashton Lukyanovsky*

Date: 10/9/2020

Goals

Long Term:	**Short Term:**
- Complete the object detection program - Integrate our Speech Recognition and Image detection programs - Implement said programs to function with the robotic arm	- Learn about Machine Learning & Neural Networks - Use the YOLO object detector to write test programs to learn about and practice object detection - Familiarize ourselves with the mechanics of the arm and its controls - Learn about stepper motors and determine the extra materials needed

Progress

Today's Progress:

Today, we focused on spending more time to learn about how stepper motors work. As Mr. Levin suggested, we looked at the all-about-stepper-motors guide from Adafruit Industries for our research. What we found was that Stepper motors are essentially DC motors that rotate in discrete steps. In order to create this rotation, stepper motors have multiple coils organized in groups called "phases", and by energizing the coils in each phase in a sequence, the motor will rotate in "steps". These repeated movements of small, fixed increments, appear to the eye as continuous motor rotation. Some benefits to using stepper motors include more precise positioning, better speed control, and low speed torque. Since stepper motors move in very precise and repeatable "steps", they are particularly good at being precise as opposed to other types of motors. This is why steppers are very common in 3D printers, CNC machines, plotters, and other devices that require precision. Additionally, these precise increments (the "steps") also allow for the rotational speed to be controlled much easier. Lastly, though not particularly relevant to our application, because of their incremental movement, steppers also have much higher torque at low speeds when compared to normal DC motors. On the other hand, some drawbacks to using stepper motors are that they tend to have low efficiency, limited high-speed torque, and they use an "open loop" design, meaning that they get no feedback in order to establish reference positions. However, since they tend to be accurate alone and our specific project doesn't require high efficiency or torque at high speeds, we think that a stepper motor will fit our project well.

Evidence of Progress:

Stepper Motors Background Research:
https://cdn-learn.adafruit.com/downloads/pdf/all-about-stepper-motors.pdf

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 10/13/2020

Goals

Long Term:
- Complete the object detection program
- Integrate our Speech Recognition and Image detection programs
- Implement said programs to function with the robotic arm

Short Term:
- Learn about Machine Learning & Neural Networks
- Use the YOLO object detector to write test programs to learn about and practice object detection
- Familiarize ourselves with the mechanics of the arm and its controls
- Learn about stepper motors and determine the extra materials needed

Progress

Today's Progress:

Today, we began putting together our presentation, copying our general structure from the last presentation and removing the content that was no longer relevant for this cycle. We also continued out stepper motor research, looking into the different types of stepper motor drivers. The simplest type of stepper motor driver (The simple unipolar driver) is simply a handful of transistors which are switched on and off in sequence in order to energize the phases and step the motor. This is relatively simple and inexpensive to build, however, they only work with unipolar stepper motors. Next is a simple dual H-Bridge Driver for bipolar motors, requiring two H-Bridges to reverse the current phases. H-Bridges can be difficult to build from scratch, however, there are H-Bridge chips available (such as the L293D) which can simplify

the process. Lastly, Adafruit suggests the “Adafruit Motor Shield V2” and the “Advanced CNC Controllers” which are a step up from the basic controllers, allowing for larger capacities and lower voltage drops, ultimately driving the steppers more efficiently.

As for our object detection code, we today realized that the code in the Github repository we’ve been looking at up to this point hasn’t been maintained (the last commit was on November 16, 2017), and that is why we were running into version issues as the code was developed when those older versions were still in use.

Evidence of Progress:

Sites Visited for Research on Object Detection:
https://github.com/llSourcell/YOLO_Object_Detection

Sites Visited For Stepper Motor Research:
https://cdn-learn.adafruit.com/downloads/pdf/all-about-stepper-motors.pdf

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 10/15/2020

Goals

Long Term:	**Short Term:**
- Complete the object detection program - Integrate our Speech Recognition and Image detection programs - Implement said programs to function with the robotic arm	- Learn about Machine Learning & Neural Networks - Use the YOLO object detector to write test programs to learn about and practice object detection - Familiarize ourselves with the mechanics of the arm and its controls - Learn about stepper motors and determine the extra materials needed

Progress

Today's Progress:

Today, we focused on looking into different options for our object detection code since we realized on Tuesday that our current reference is out of date. Since last class, Emma reached out to a few people that she knew who have experience working with object detection, and they suggested that we take a look at fast.ai, classyvision, and detectron2, as well as a better overview of the YOLO algorithm found here. After looking through each one, we decided that for us, the most readable and implementable for our project, given our time constraints and lack of prior experience in object detection, is the YOLO algorithm. After reading through the page, we first looked into what a Blob object is as neither of us had heard of it before. According to the website, a blob is a 4D numpy array object (images, channels, width, height), but more specifically, a blob, or a Binary Large OBject, refers to a group of connected pixels in a binary image. The term "Large" indicates that only objects of a certain size are of interest and while the other "small" binary objects are typically noise. Essentially, Blobs allow the computer to determine whether nearby pixels are related and therefore more likely to come together to form an object in the image.

Evidence of Progress:

Resources:

- https://opencv-tutorial.readthedocs.io/en/latest/yolo/yolo.html
- https://answers.opencv.org/question/50025/what-exactly-is-a-blob-in-opencv/
- https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html
- https://www.fast.ai/
- https://classyvision.ai/
- https://ai.facebook.com/tools/detectron2/
- https://github.com/facebookresearch/detectron2/blob/master/README.md?fbclid=IwAR2DbqaawNUoobSJUAY69AYR-LhuIiMSWacN5STTKGymxDHw6YKVRYeI-9E

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 10/20/2020

Goals

Long Term:

- Complete the object detection program
- Integrate our Speech Recognition and Image detection programs
- Implement said programs to function with the robotic arm

Short Term:

- Learn about Machine Learning & Neural Networks
- Use the YOLO object detector to write test programs to learn about and practice object detection
- Familiarize ourselves with the mechanics of the arm and its controls
- Learn about stepper motors and determine the extra materials needed
- Finish putting together the presentation

Progress

Today's Progress:

Today, we focused on working on the object detection code on our own computers. Working through this tutorial on readthedocs, we learned how to load an image into Python using opencv, load class names and assign random colors, give configuration and weight files for the model and load the network, construct and display a blob from the image, create a trackbar for the bounding boxes at different confidence levels on the blob, detect objects, draw bounding boxes, and re-show the image with the bounding boxes drawn over all objects with 50% or more confidence. Once we were able to get the program working with the provided images as well as some simple images that we found online, we decided to try to see if we could detect objects in an image that we take on our own. Upon running our code with our own image, our first realization was that we were going to need to take into consideration the size of the image for visibility reasons (the image we used that was taken on Emma's phone was about 8x bigger than a reasonable size to fit on the screen). Our first instinct was to use the cv.resize() function to proportionally shrink the image, however, once we did that, we no longer could see bounding boxes on the resized image and we speculate that that has to do with the image resizing. For our final project, however, we don't believe that this will be a necessary issue to deal with as the image does not need to be displayed to fit on a screen.

Evidence of Progress:

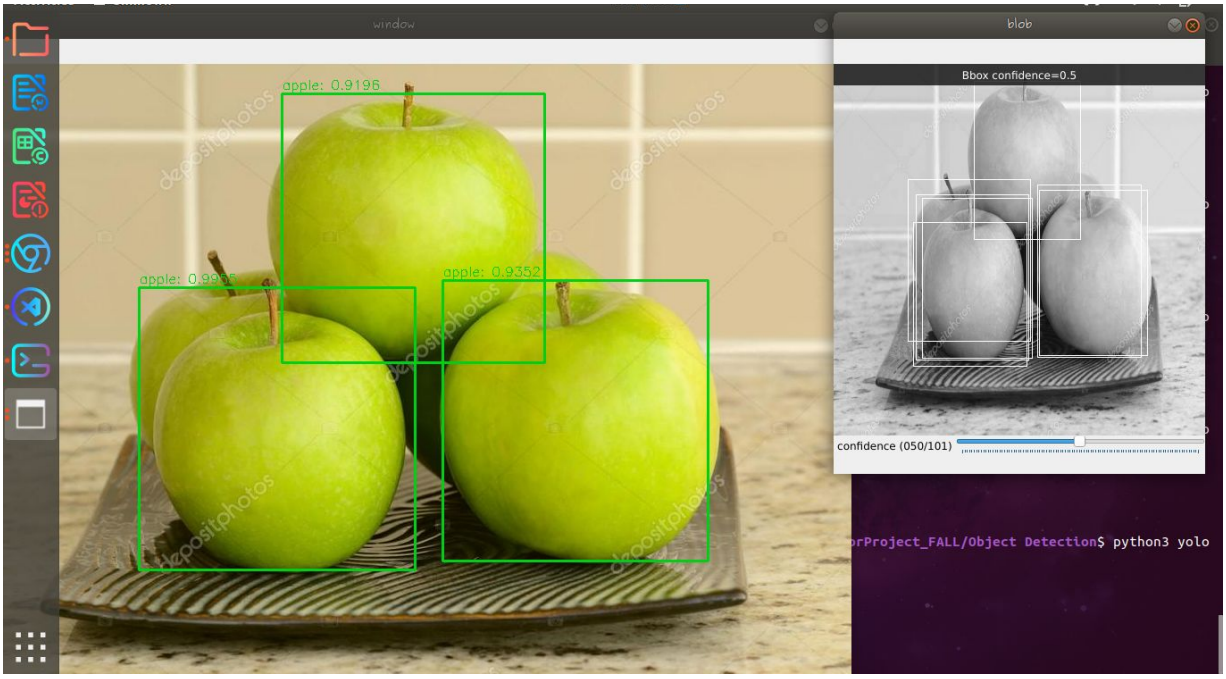
Research References:

- https://opencv-tutorial.readthedocs.io/en/latest/yolo/yolo.html
- https://www.pyimagesearch.com/2017/08/21/deep-learning-with-opencv/
- https://www.learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/

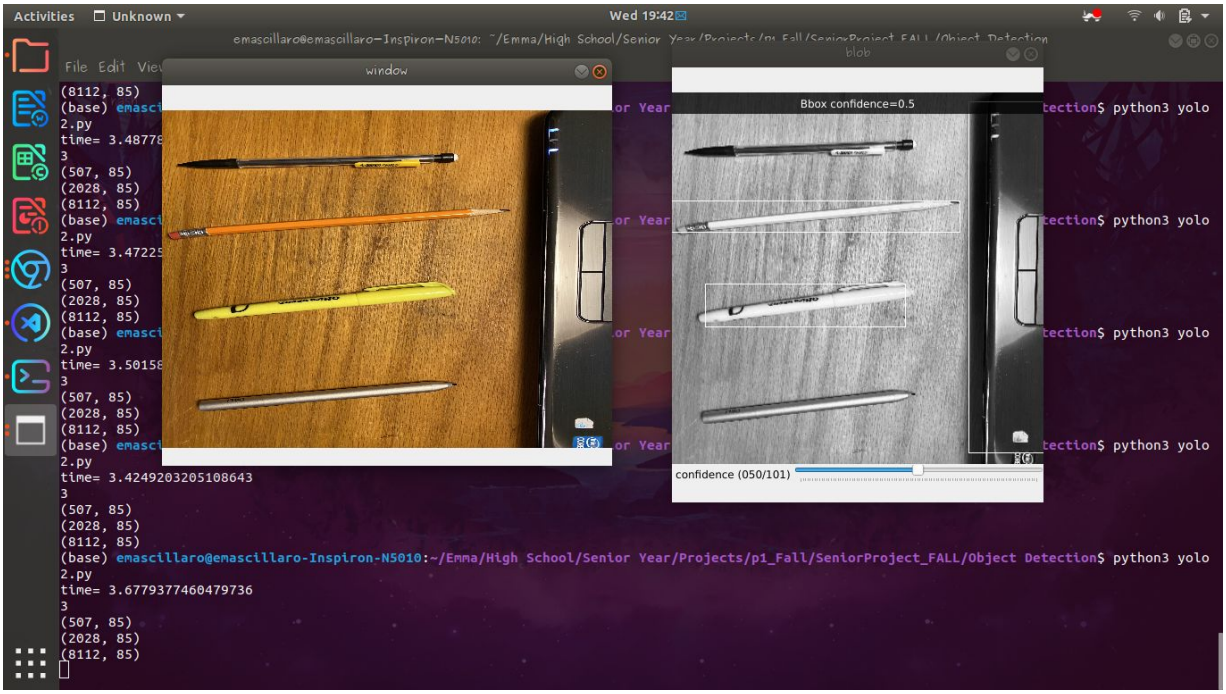
- https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/
- https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Drawing Bounding Boxes & Recognizing Object:



Difficulties with Image Resizing:



Signatures

Emma Mascillaro ***Ashton Lukyanovsky***

Date: 10/27/2020

Goals

Long Term:	**Short Term:**
- Implement said programs to function with the robotic arm - Complete code to control the movement of the robotic arm	- Present Project Progress - Create outline for upcoming cycle - Complete all code except for the code for the arm by the next presentation

Progress

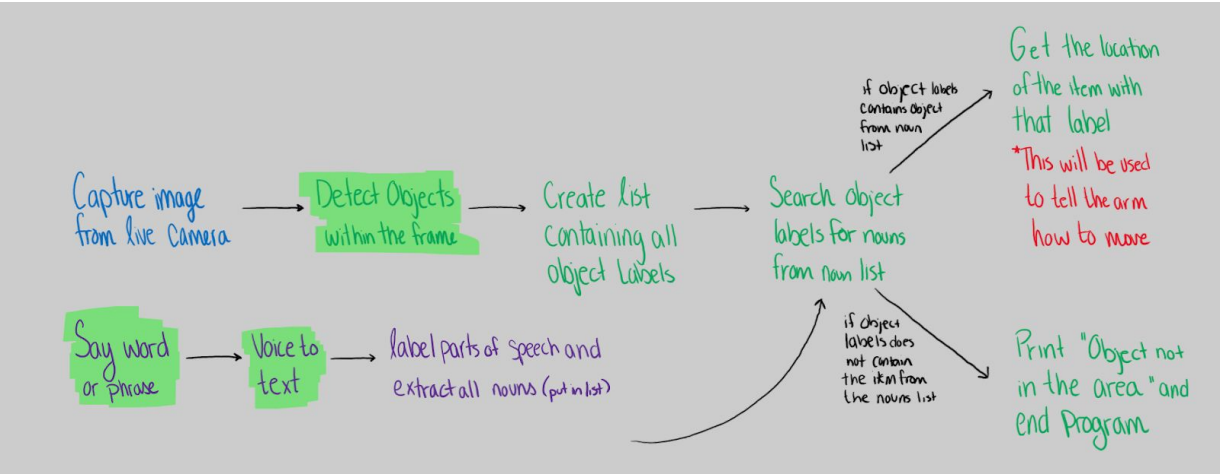
Today's Progress:

Today, after presenting our progress from the last cycle, we created an outline for the progress that we wanted to complete by the end of this next presentation cycle (ends on 11/13). We decided that our ultimate goal for November 13th was to have all of our code except for the code for the arm to be completed. As shown in the outline, this means that we would need to first write code that captures an image from a live camera for our object detection. Then, we would need to create a list of the objects that the user asked for and a list of objects in the image. With these lists, we would be able to determine whether the item the user asked for is in front of the arm. Dependent on whether the item is or is not in front of the arm, we would find the distance between the arm's claw and the object or end the program, respectively. If the item is in the frame of the camera, this distance information would be useful in order to determine the x, y, and z movements the arm would have to complete in order to approach the object. Today, we also made the decision to use a still image to determine the location of the object in relation to a live video feed. After researching stepper motors, we learned that they are very precise, and since the objects themselves are stationary, we came to the conclusion that we could accurately move the arm into the correct position without needing to use a live video feed.

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Project Outline for this Presentation Cycle (Previously Completed Tasks Highlighted):



Signatures

Emma Mascillaro ***Ashton Lukyanovsky***

Date: 10/29/2020

Goals

Long Term:

- Implement said programs to function with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:

- Capture a still frame from a live camera
- Complete all code except for the code for the arm by the next presentation

Progress

Today's Progress:

Today, we worked on our code for capturing a still image from a live camera. In OpenCV, we can access a camera, we use the method VideoCapture() where the argument inside the parentheses determines which camera we will be using. The computer's built-in webcam is 0, and an additional camera would be 1. For today, we tested our program with 0, our webcam, but once we have a camera to mount on the arm, we would change the camera source to 1. To capture a frame from the camera feed, we used the .read() method which returns a boolean value stating if the frame was captured correctly or not and the still frame which is a numpy array. We then use the cv2.imwrite() method to save our image as a jpg file which we will use in the Object Recognition program.

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Resources for Capturing an Image From Webcam:

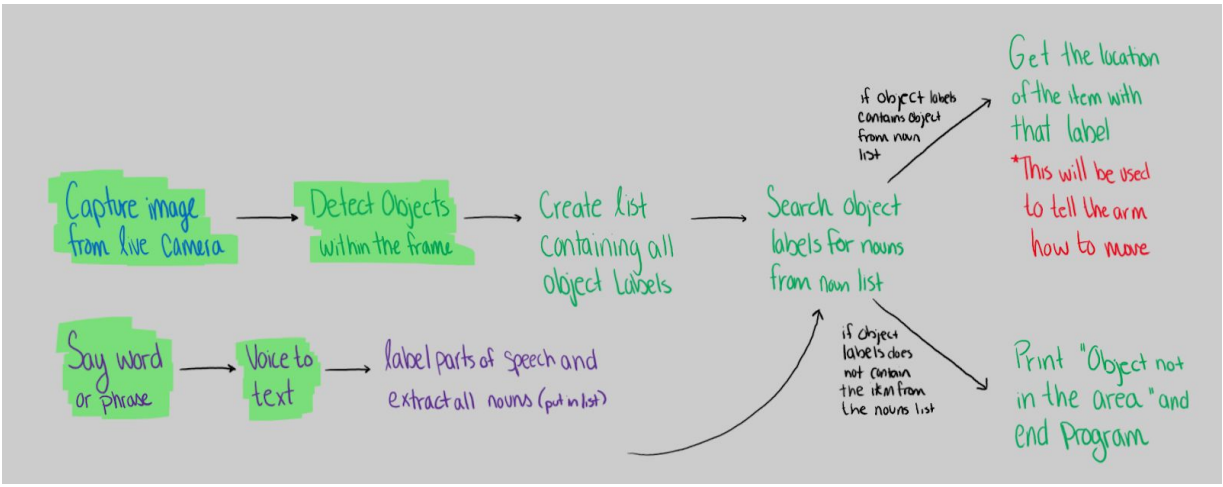
- https://www.studytonight.com/post/capture-videos-and-images-with-python-part2#
- https://stackoverflow.com/questions/11094481/capturing-a-single-image-from-my-web cam-in-java-or-python

Code to Capture an Image from a Webcam:

9 lines (8 sloc) | 217 Bytes

```
1  import cv2
2
3  videoCaptureObject = cv2.VideoCapture(0)
4  result = True
5  while(result):
6      ret, frame = videoCaptureObject.read()
7      cv2.imwrite("Capture_Image.jpg", frame)
8      result = False
9  videoCaptureObject.release()
```

Updated Outline:



Signatures

Emma Mascillaro ***Ashton Lukyanovsky***

Date: 11/3/2020

Goals	
Long Term: - Implement said programs to function with the robotic arm - Complete code to control the movement of the robotic arm	**Short Term:** - Learn how to tag words with their part of speech - Complete all code except for the code for the arm by the next presentation

Progress

Today's Progress:

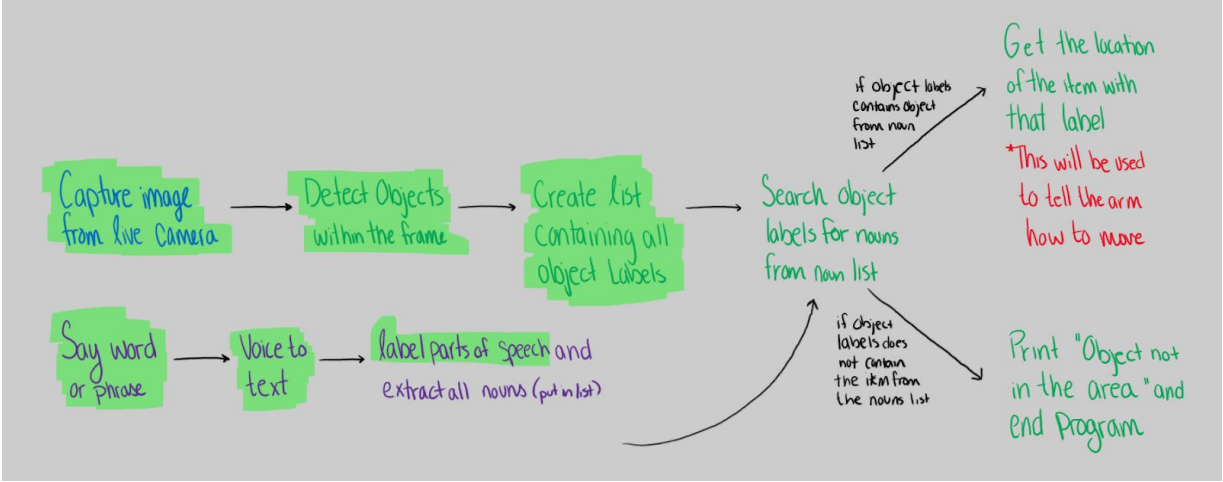
Today, we worked on learning how to detect if a spoken word refers to an object by looking for nouns in our speech input. We learned that we could do this by using Python's Natural Language Toolkit (NLTK). Once we installed the Python NLTK suite of libraries and programs for symbolic and statistical natural language processing, we began a tutorial where we learned how to tokenize text and tag each word with its part of speech. Text can be tokenized into words or sentences using word_tokenize() or sent_tokenize(), splitting up the inputted text by words or sentences into a list. From here, we learned how to tag each word by it's part of speech by iterating through each word in each sentence and using the nltk.pos_tag() method. This outputs a tuple for each word where the 0th index is the word token and the 1st index is a speech code that refers to the word's part of speech.

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Resources for NLTK:
https://pythonspot.com/category/nltk/
http://www.nltk.org/book/ch05.html

Updated Outline:



Signatures

Emma Mascillaro ***Ashton Lukyanovsky***

Date: 11/10/2020

Goals

Long Term:

- Implement said programs to function with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:

- Create a list of all nouns in the spoken input
- Learn how to call a specific variable from one Python script to another
- Complete all code except for the code for the arm by the next presentation

Progress

Today's Progress:

Today, we worked on creating a list of nouns from the spoken input and making it accessible from a separate python file. Using what we learned last time about tagging tokens with their part of speech, we took our list of tuples (word token at index 0 and part of speech at index 1) and used a for loop to search for all of the tuples that contained "NN" at index 1, indicating that the word at index 0 of that tuple is a noun. The tuples that fit that condition had their 0th index appended to another list which consisted of all of the nouns from the speech.

Something interesting that we found was that there were some instances where a word was was being used in one context in the speech, but was recognized by the NLTK with a secondary definition, causing the part of speech to be incorrect (The first time we noticed this was when the word “blue” was being used as an adjective but was tagged as a noun). In order to improve this accuracy, we will need to do further research on natural language processing, but for now, our current setup is working sufficiently in most cases. Today, we also looked into how we could access a variable from one python file in another as we need to use the noun list with some outputs from the object recognition file in order to determine if an object is in the image. We found that we can do this by placing our two files in the same folder and on the Object_Recognition.py file, we could write “from Speech_Recognition import noun_list”, allowing us to access the information stored in noun_list in Object_Recognition.py.

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Resources for NLTK:
https://pythonspot.com/category/nltk/
http://www.nltk.org/book/ch05.html

Resources for Integrating Programs:
https://datatofish.com/one-python-script-from-another/

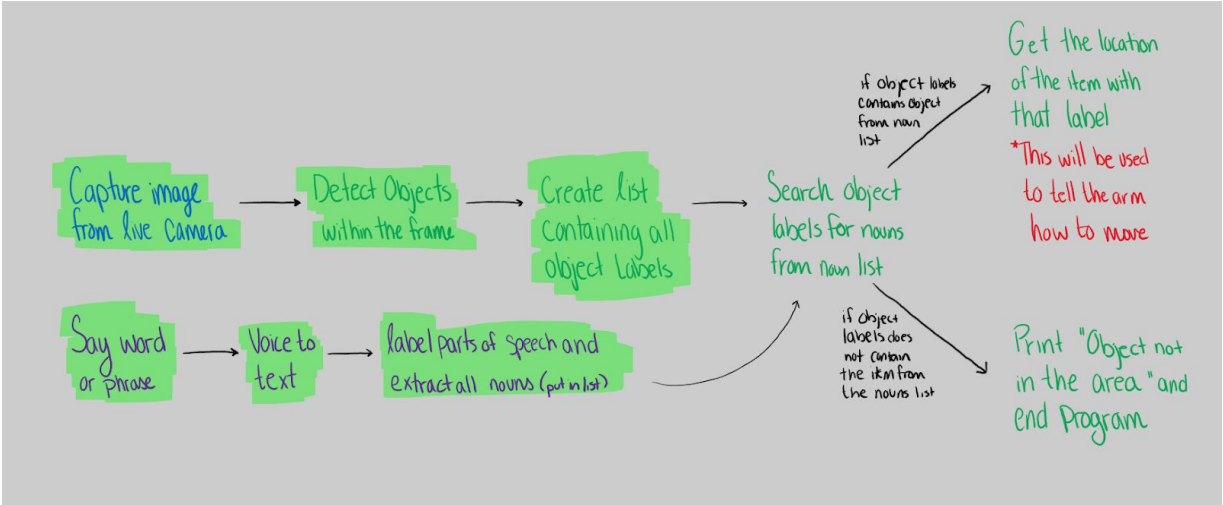
Tagging Parts of Speech:

```
35      data = []
36      noun_list = []
37      for sent in sentences:
38          data = data + nltk.pos_tag(nltk.word_tokenize(sent))
```

Creating a List of all Nouns:

```
40      for word in data:
41          if 'NN' in word[1]:
42              #print(word)
43              noun_list.append(word[0])
44
45      print("nouns:", noun_list)
```

Updated Outline:



Signatures

Emma Mascillaro ***Ashton Lukyanovsky***

Date: 11/11/2020

Goals

Long Term:

- Implement said programs to function with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:

- Create a list of all objects recognized in the image
- Determine if the object(s) requested are in the camera's field of view
- Complete all code except for the code for the arm by the next presentation

Progress

Today's Progress:

Today, the first thing that we did was to create a list of all items that were detected in the image. In order to do this, we took the text that was displayed on the banners of each of the bounding boxes (in the form of "person: 0.5215" where "person" was the object detected and "0.5215" is the confidence) and split the strings using text.rsplit(':', 1)[0] in order to only record the object detected in a new string. All of these new strings were then appended to a list of items detected in the image. The next thing we did was to iterate through the list of nouns to see how many times each noun appeared in the list of items detected in the image, using the

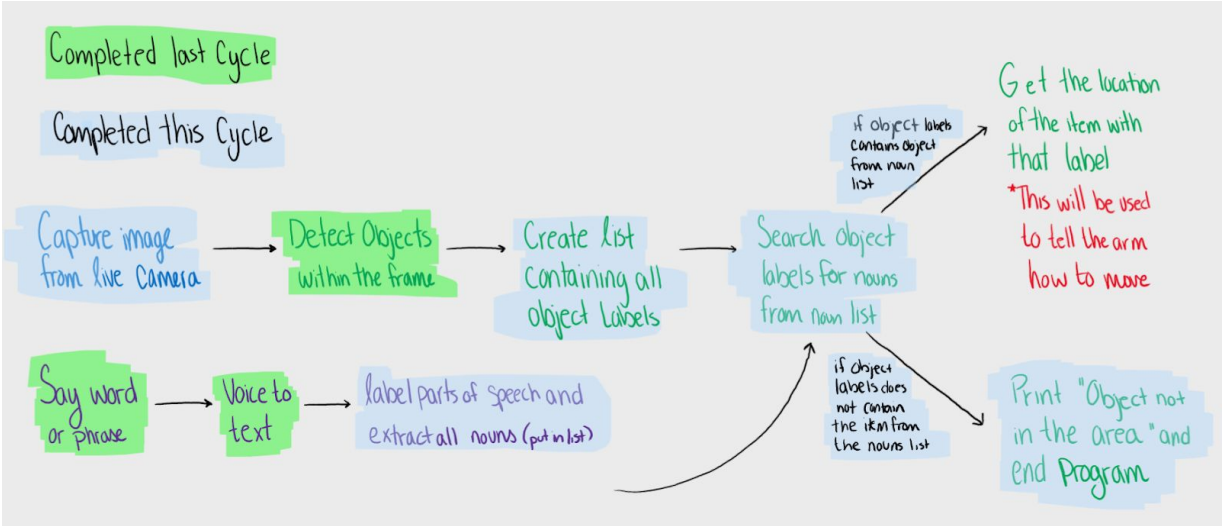
list.count() method. If the count was 0, we determined that the object was not in the area and ended the program there. If the count was 1, we determined that the object was greater than one, and here is where we would go straight to finding the location of the object (we have not gotten to finding the location yet). However, if the count was greater than one, we need to have a set of criteria to determine which object to get the location of (as of right now, we're thinking of getting the location of the first occurrence, but that is still subject to change).

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Resource on Splitting a String at a Specific Character:
https://stackoverflow.com/questions/15851568/how-to-get-the-last-part-of-a-string-before-a-certain-character

Updated Outline:



Signatures

Emma Mascillaro ***Ashton Lukyanovsky***

Date: 11/12/2020

Goals

Long Term:	**Short Term:**

- Implement said programs to function with the robotic arm - Complete code to control the movement of the robotic arm	- Integrate the image captured into the Object Detection Code - Complete all code except for the code for the arm by the next presentation

Progress

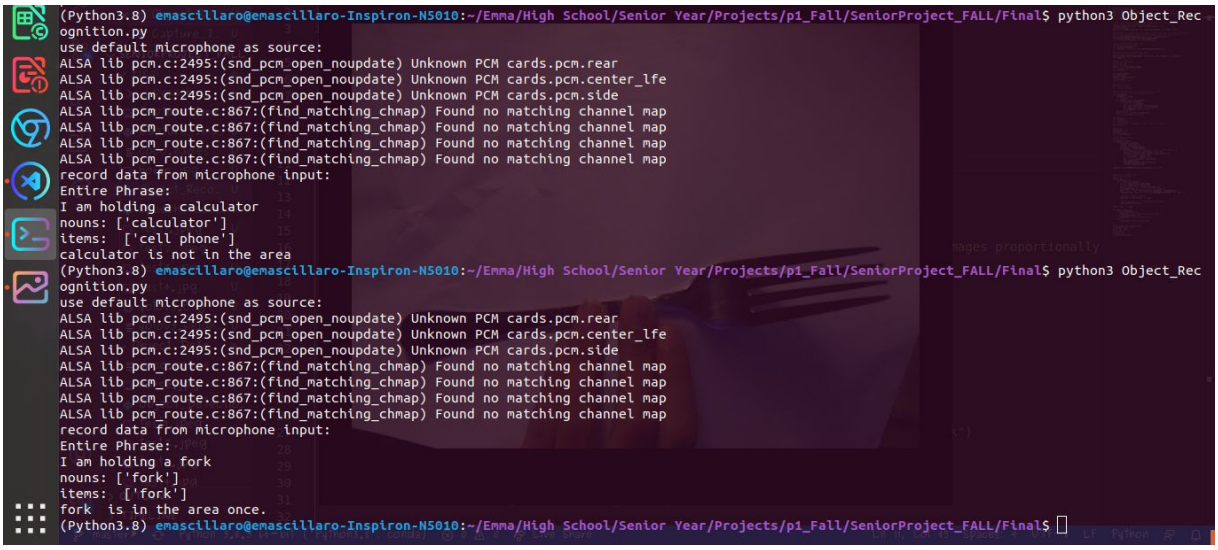
Today's Progress:

Today, we went back to our Capture_Image.py file and learned how to integrate the image that we took into our object detection program. In order to do this, we had to put the Capture_Image.py file in the same folder as Object_Recognition.py and import Capture_Image in the beginning of the file. By doing this, we run the Capture_Image.py program when the Object_Recognition.py program is run, allowing us to have a new image of the surroundings taken at runtime. We then worked on completing our presentation.

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Captured Image, Speech Recognition, and Object Detection together:



Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 11/17/2020

Goals

Long Term:

- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:

- Access the location and size of objects detected in the frame

Progress

Today's Progress:

Today, we worked on wrapping up our object detection code by finding a way to access the location and size of our desired objects in the image. We went about doing this by utilizing the dimensions of the bounding boxes which we had initially created around the objects in the frame. Even though the bounding boxes may not be the exact same sizes and shapes of the objects, they will be sufficient as the entire object is encased in the bounding box and therefore, closing the hand around the box will cause us to grab the object. Once we established these values, we printed them out in our terminal. While we were working on this, we ran into a case we had previously overlooked concerning capitalization. When running our code, we found that if we searched for "Person" in our noun list that contained "person", we found no matches as the capitalization in the words were different. We addressed this issue by adjusting all characters to be lowercase before checking if the objects were contained in the image.

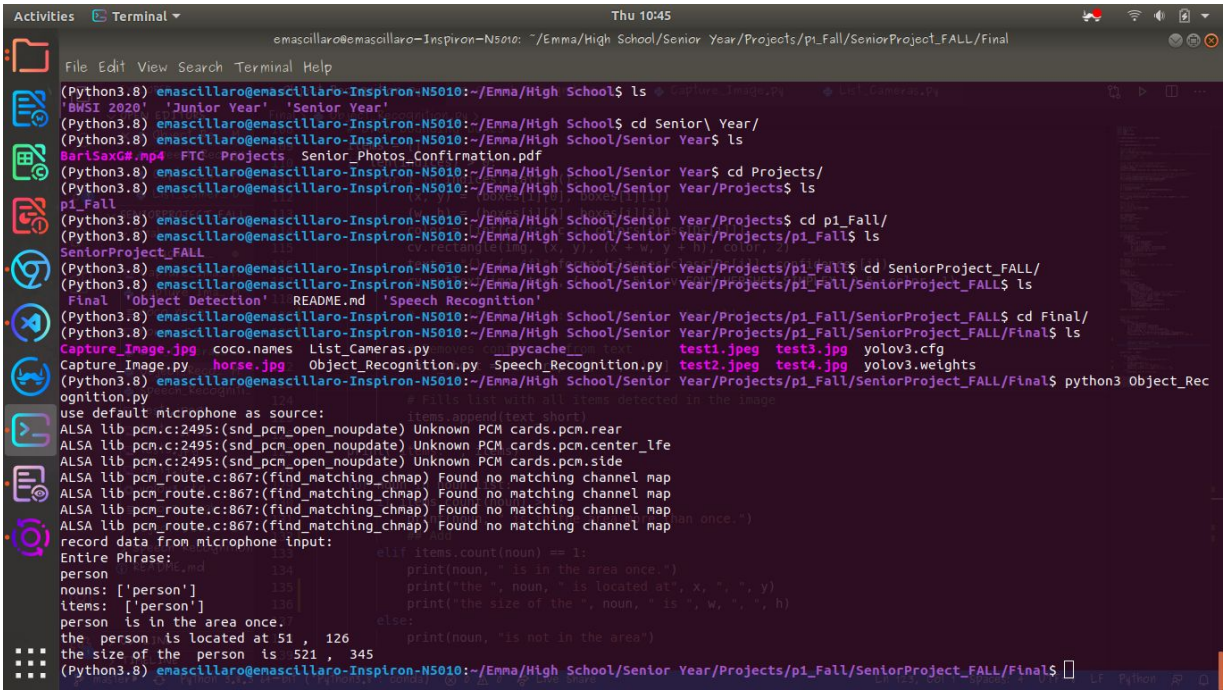
Evidence of Progress:

Research:

- https://www.geeksforgeeks.org/isupper-islower-lower-upper-python-applications/

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Fixed capitalization and printed location & size:



Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 11/19/2020

Goals

Long Term:

- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:

- Create a plan of how we could go about the motions of the arm in order to get the hand to the location of the desired object

Progress

Today's Progress:

Today, we worked on planning out a path for the arm to follow in order to get the hand to the location of our desired object. In order to get an idea of what would be in the frame of the camera after every movement, Emma temporarily attached her phone camera to the arm and took a picture after each movement. After trying a few different methods, we decided on one that seemed the most promising and forgiving for any error that we may accumulate. In our plan, we decided that our first movement would be to open the hand. From there, we would rotate the base of the arm until the x-coordinate of the object was in the center of the frame. From here, we decided to extend the shoulder of the arm until the y-coordinate was in the center of the frame. We then chose to retract the elbow of the arm to lower the z-coordinate until the object was at the top of the frame and then loop through extending the shoulder and retracting the elbow in order to gradually approach our desired object. Though this method is not particularly efficient, we chose to use it initially as we foresee it to be a viable way of getting the desired result that would be easiest for us to implement initially. However, once we are able to get this method to work, we plan on using a more logical approach such as the one described in the Python Robotics *Read the Docs*.

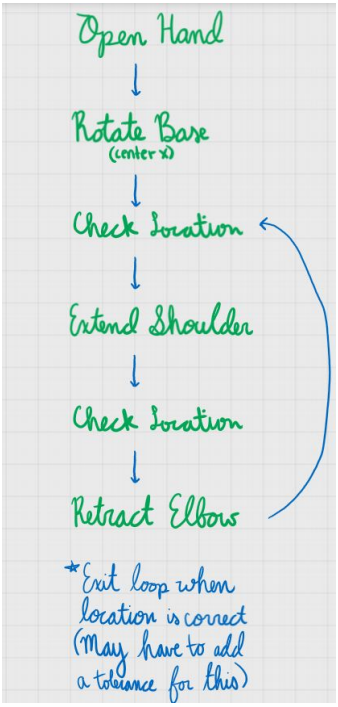
Evidence of Progress:

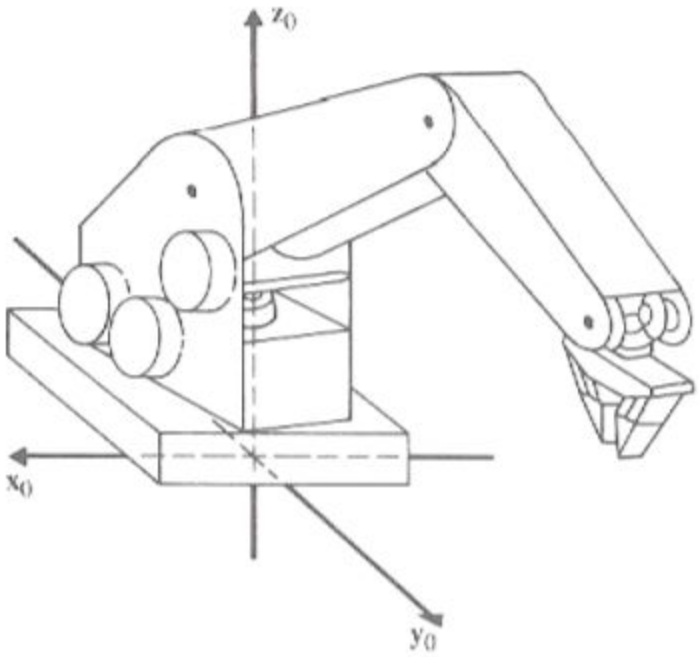
Research:

- https://stars.library.ucf.edu/cgi/viewcontent.cgi?article=6068&context=rtd
- https://pythonrobotics.readthedocs.io/en/latest/modules/arm_navigation.html#n-joint-arm-to-point-control
- https://pythonrobotics.readthedocs.io/en/latest/modules/arm_navigation.html

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Code Flowchart:





Signatures

Emma Mascillaro ***Ashton Lukyanovsky***

Date: 11/20/2020

Goals

Long Term:
- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:
- Order stepper motor drivers
- Familiarize ourselves with Arduino

Progress

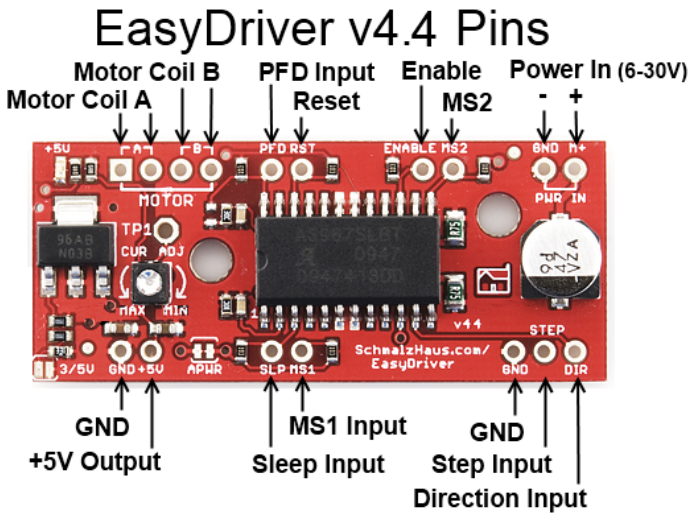
Today's Progress:

Today, our main objective was to order our stepper motor drivers so that we would have the chance to work with them as soon as possible. After watching a video of someone who has gotten a MiniMover-5 (the arm we are using) to work again after stripping the original electronics, we decided the safest route would be to use the same drivers as him since we know it will work. Because of this, we ordered five easy driver stepper motor drivers. For our project, even though there are six steppers on the arm, we only need four as we do not plan to rotate either wrist. With our drivers, we will be able to track distance travelled by counting the number of "steps" that the motors take, and with this information, we will be able to know the location of the arm in relation to the objects that need to be picked up. We also decided today that we would use two Arduino Unos as our motor controllers as Emma already had them from previous projects and they had enough digital output pins to be able to control two drivers per Arduino.

Evidence of Progress:

Research:

- http://www.almostscientific.com/2011/10/23/its-alive-mini-mover-5-resurrected-with-arduino/
    - https://vimeo.com/30971710
- https://www.sparkfun.com/products/12779



-

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro	***Ashton Lukyanovsky***

Date: 11/24/2020

Goals

Long Term:	**Short Term:**
- Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Learn more about 6-wire stepper motors - Begin to build circuit connecting the motors, drivers, and motor controllers

Progress

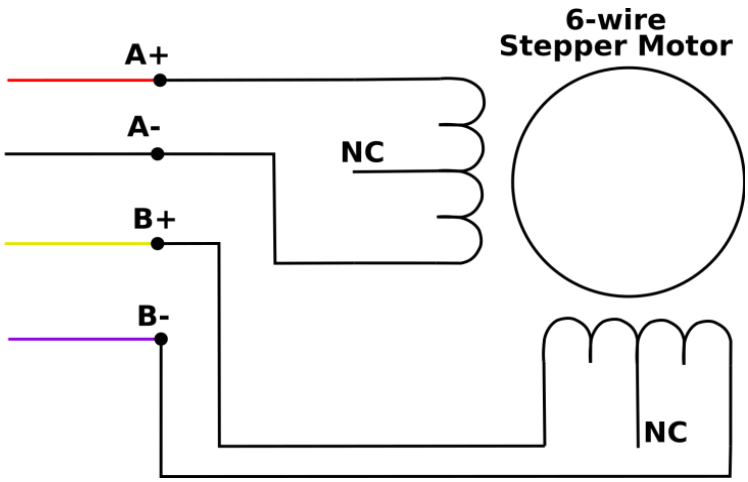
Today's Progress:

Today, my main objective was to learn about six-wire stepper motors (the steppers on our arm have 6 wires), how they worked, and how I would go about connecting them to the easy drivers once they came in. After doing some reading, I learned that on six wire stepper motors, there are two sets of 3 wires where each set is connected to one coil. In each set of wires, there are two wires that charge the coil and a third wire, a common tap, which makes these motors more commonly known as unipolar motors, suited best for applications where high torque at low speeds is necessary. Despite the fact that most stepper motor interfaces, easy drivers included, do not support six-wire motors, six-wire motors can be used in the same way as the more common four-wire motors by omitting the common tap wire. In order to determine which wire is the common tap, I measured the resistances across the stepper wire pairs and determined that all of the red wires on our motors are the common tap as the resistance between the common tap and the other wires is lower than the resistance between the two ends of the coil. Once I determined the common tap, I was ready to connect the motors to the drivers (drivers still haven't come in, so I labelled the wires with the pin they would connect to. In order to do this, I labelled one pair of wires A+ and A-, and the other pair B+ and B-. As there is no polarity on the coils, I didn't need to worry about plugging in a coil backwards on the board.

Evidence of Progress:

Research:

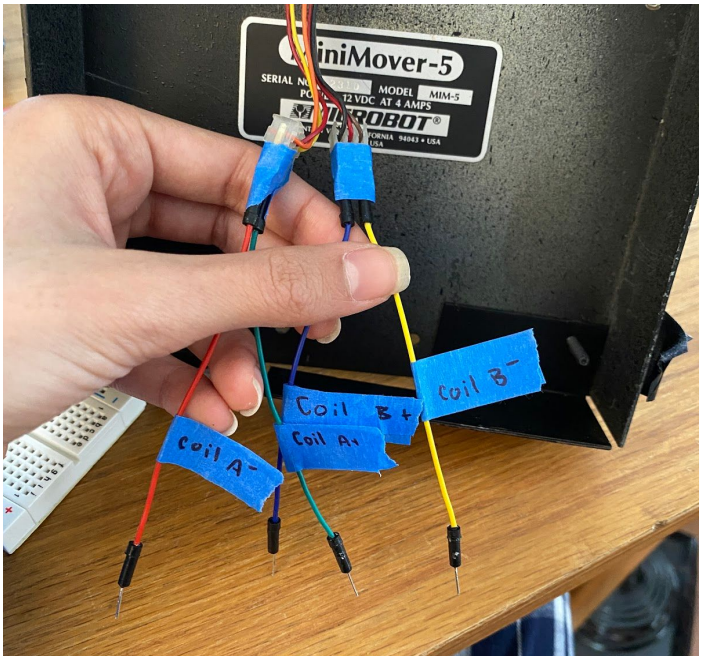
- https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000PAkPSAW
- https://www.instructables.com/Arduino-6-wire-Stepper-Motor-Tutorial/
- https://learn.sparkfun.com/tutorials/easy-driver-hook-up-guide/all



Determining the Common Tap:

- https://docs.google.com/document/d/1s2d1ec_W4-n7wR7uLjfDvybRelxaUsaxdpj48KaDIIA/edit

Labelling wires to connect to Stepper Driver:



Signatures

Emma Mascillaro

Date: 12/1/2020

Goals

Long Term:

- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:

- Finish building a circuit connecting the motors, drivers, and motor controllers

Progress

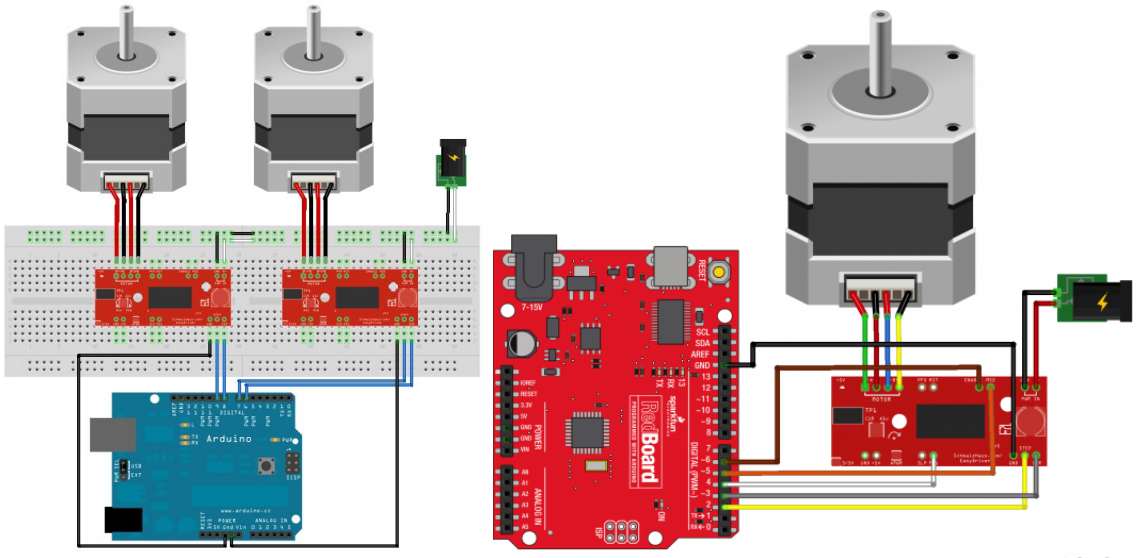
Today's Progress:

Today, we worked on building the rest of our circuit. Now that we have our motors labelled with the correct pins they will be connected to, our next objective was to connect the drivers to motor controllers and power sources. Reading further through the Sparkfun Easy Driver Hookup Guide, we used the schematic provided to tag the pins on our arduino that we will be connecting to corresponding pins on the Easy Driver (our drivers still haven't come in yet). Once this was completed, we found another image where an Easy Driver was on a breadboard and used this as a guide to find the relative pins on our breadboards where we could connect our wires to the correct driver pins once they come in and are put in place.

Evidence of Progress:

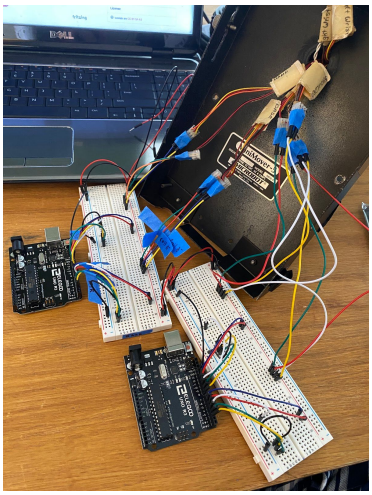
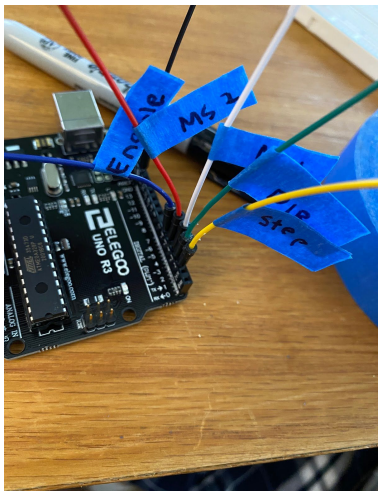
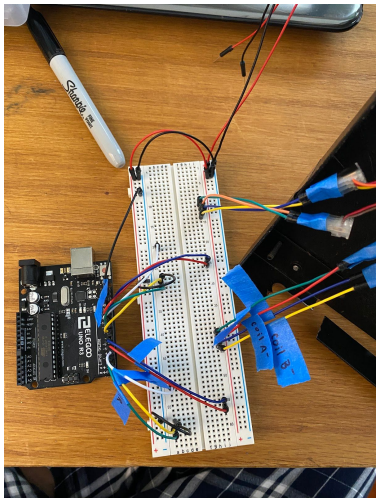
Research:

- Circuit for Arduino/Driver/Motor
  - https://learn.sparkfun.com/tutorials/easy-driver-hook-up-guide/all
  - https://learn.sparkfun.com/tutorials/redboard-vs-uno/all
  - https://www.schmalzhaus.com/EasyDriver/Examples/EasyDriverExamples.html



Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Our Circuit:

Signatures

Emma Mascillaro ***Ashton Lukyanovsky***

Date: 12/3/2020

Goals

Long Term:	**Short Term:**
- Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Learn about Arduino code as well as how to flash it onto a physical Arduino

Progress

Today's Progress:

Today, our main objective was to learn how to write Arduino code and test flashing it to our physical Arduino Uno board. The first thing we did was install the Arduino IDE on our computers. The Arduino IDE allows us to write and compile code as well as flash it onto our physical board. Installation uploading code to the boards ended up taking longer than expected on Emma's computer as she was having an issue with permission being denied, but she was able to find workarounds for the errors. Since Arduino code is very similar to C and we have some experience with C++ from freshman year, understanding the syntax for Arduino code was relatively easy, and we were able to run a couple of test programs on our physical board including a standard Blinking Light program.

Evidence of Progress:

Research:

- https://www.arduino.cc/reference/en/
- https://askubuntu.com/questions/1056314/uploading-code-to-arduino-gives-me-the-error-avrdude-ser-open-cant-open-d

Github Link:

https://github.com/emascillaro/SeniorProject_FALL

Blinking Light using Arduino Code:

- Video

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 12/8/2020

Goals

Long Term:	**Short Term:**
- Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Look at options to import values from our other python code into our stepper motor control code

Progress

Today's Progress:

Today, our main objective was to look at our options for importing values from our other python code into our stepper motor control code. In our research, we came across Firmata which is a generic protocol that allows communication between a microcontroller and software on a computer, and using firmata, any software from a computer capable of serial communication can communicate with a microcontroller through Firmata. This sounded like a good option for us because using Firmata allows us to use Python to control the motors directly, and we already know how to import values from one Python file to another. For this reason, we decided to use Firmata for our project, and began looking at pyFirmata, the Python interface for the Firmata protocol.

Evidence of Progress:

Research:

- http://www.aemn.pt/formacao/Arduino/DVD_tutorial%20-%20Arduino/arduino%20eBook%20Collection/Python%20Programming%20for%20Arduino.pdf
- https://github.com/tino/pyFirmata

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 12/10/2020

Goals

Long Term:	**Short Term:**
- Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Learn how to incorporate pyFirmata in order to control the stepper motors in the robot arm using Python

Progress

Today's Progress:

Today, our main objective was to learn how to use pyFirmata in order to control the stepper motors on our arm using Python. The first thing we did was install the pyFirmata package with pip so we could import it into our Python code. We then went into the Arduino IDE and uploaded the *StandardFirmata* sketch so that we could use that protocol to control our board. Once we ran this, we went back to our Python program and ran an equivalent test program to blink an LED. When we ran this program, the light blinked on the board in the same manner that it did with the Arduino Code we had tested a few days prior. Though there was more to the tutorial that we skimmed through, we only focused on understanding how to use pyFirmata with digital output that will be the only thing we will need to use in our project.

Evidence of Progress:

Research:

- https://realpython.com/arduino-python/

Python_Arduino_test.py > ...

```python
import pyfirmata
import time

board = pyfirmata.Arduino('/dev/ttyACM0')

while True:
    board.digital[13].write(1)
    time.sleep(1)
    board.digital[13].write(0)
    time.sleep(1)
```

Python_Arduino_test.py > ...

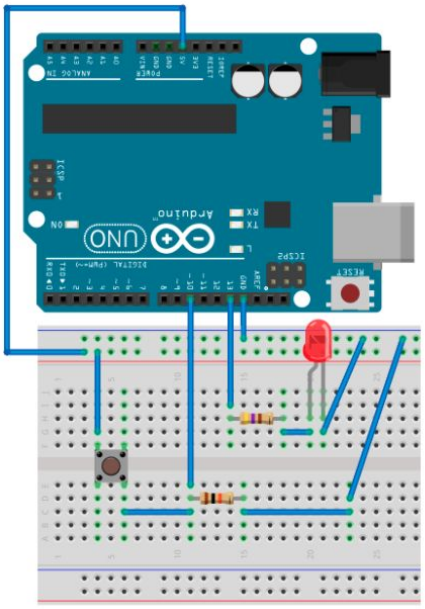
```python
import pyfirmata
import time

board = pyfirmata.Arduino('/dev/ttyACM0')

it = pyfirmata.util.Iterator(board)
it.start()

board.digital[10].mode = pyfirmata.INPUT

while True:
    sw = board.digital[10].read()
    if sw is True:
        board.digital[13].write(1)
    else:
        board.digital[13].write(0)
    time.sleep(0.1)
```



Video of Blinking LED with Python Code: Video

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures	
Emma Mascillaro	***Ashton Lukyanovsky***

Date: 12/14/2020

Goals	
Long Term: - Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Further research on the extent of our capabilities with using pyFirmata - Get a better understanding of the syntax of pyFirmata methods

Progress

Today's Progress:

Today, our main objective was to do further research on the extent of our capabilities with using pyFirmata as well as to get a better understanding of the syntax of pyFirmata methods. From the information we've read about and gathered, we have come to the understanding that the most important method in the pyFirmata library for us will be the .digital[].write() function which allows us to send a value to a digital output pin on our board, and in turn, this is how we will be able to go about controlling our motors.

Evidence of Progress:

Research:

- https://jdreyer.com/projects/Arduino/StepperMotor.php
- https://realpython.com/arduino-python/
- https://www.arduino.cc/en/Reference/Firmata
- https://github.com/tino/pyFirmata
- https://forum.arduino.cc/index.php?topic=594124.0
- https://pyfirmata.readthedocs.io/en/latest/pyfirmata.html

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 12/15/2020

Goals

Long Term:	**Short Term:**
- Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Begin writing code for controlling the robotic arm

Progress

Today's Progress:

Today, our main objective was to begin writing code to control the stepper motors on the arm. For testing purposes, we wrote a simple program where we would be able to manually control the stepper motors through a GUI, allowing us to be able to observe how the arm will move when the values sent to the pins change. This will help us once we get the drivers (they are supposed to arrive over winter break) to write our code which we will implement for our final project.

Evidence of Progress:

Research:

- https://forum.arduino.cc/index.php?topic=396450.0

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 1/5/2021

Goals

Long Term:

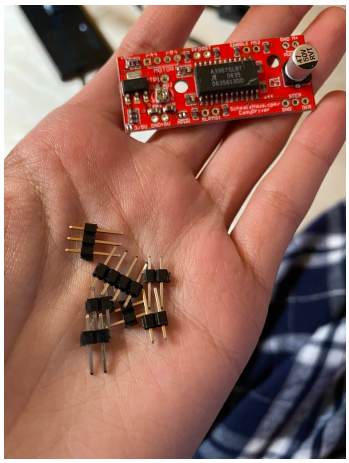
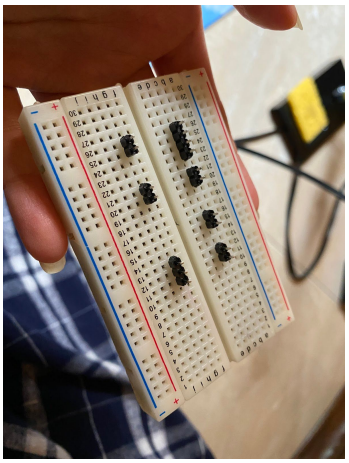
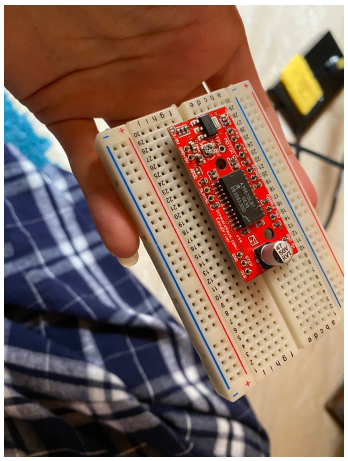
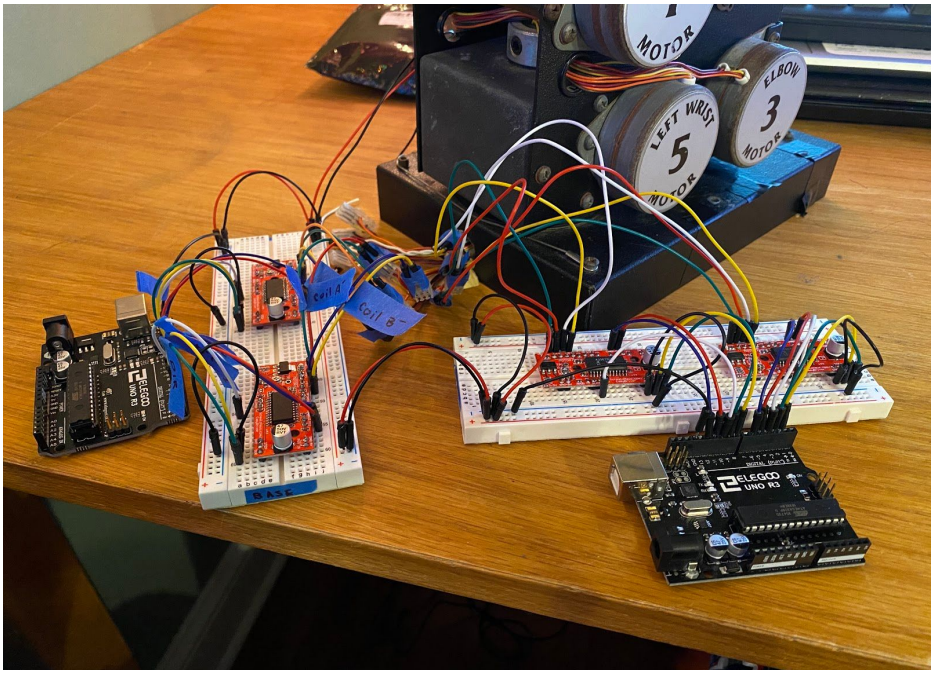
- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm
- Solder Stepper motor drivers and complete circuits

Progress

Today's Progress:

Today, our main objective was to take the stepper motor drivers (Sparkfun easy drivers) that came in and to insert them into our circuit. In order to do this, we first had to attach the pins to the driver by soldering them on. We took the set of connected pins and cut them into sections of different sizes that would align with the holes on the driver. We then put the pins into a spare breadboard in their respective locations and laid the driver on top, allowing us to solder the pins on in a more stable environment. Once the pins were soldered on, we removed the driver from the spare breadboard and inserted it into our circuit. Though our estimates were mostly accurate, we had to adjust the leads connecting to the A and B motor pins to line up with the driver.

Evidence of Progress:

Research:

- https://www.youtube.com/watch?v=6IyJpCWOtWg&t=501s

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 1/7/2021

Goals

Long Term:

- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm
- Test circuits by running code on arm

Progress

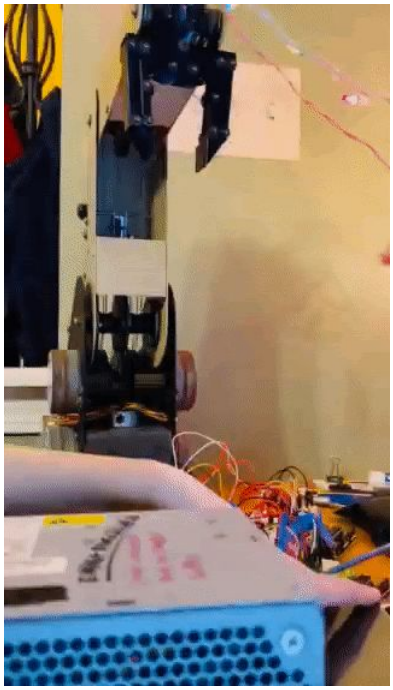
Today's Progress:

Today, our main objective was to test our circuits by running test code on the motors. We decided to use Arduino code as it would be easier for us to test with and debug. Before we were able to run code, we first had to connect our circuit to a power supply. We were given a Dell L280P-01 Desktop power supply with 24 pins, so we did some research to find a chart that decoded what the function of each pin was. Once we had this information, we connected the rails of our breadboards to 5V of power (we decided to start with a lower voltage to prevent damage to any of our materials) and ran sample test code from the Sparkfun manufacturer website. This test code used user input in order to determine the speed and direction of rotation for the stepper motor. When testing, we used option 3, *Turn at 1/8th microstep mode*. At 5V, when we ran option 3, we found that there wasn't enough torque to rotate the gears, so we decided to move to a higher voltage of 12V. This worked well for the base, however, when we tested the hand, 12V was still not producing enough of a torque to rotate all of the gears and move the hand. As the arm itself has rust and an accumulation of dust in its interior, likely due to it not being in use for a while, we speculate that this is causing there to be more friction when trying to rotate the gears, causing the steppers to need to reach a higher torque to perform their tasks. This idea of ours was reinforced by the Mini

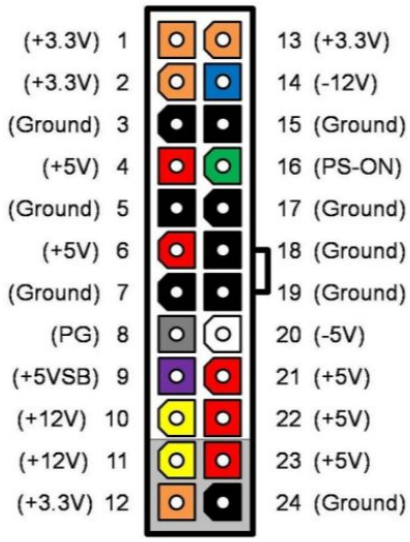
Mover 5 manual which stated that the gears should be relatively easy to manually rotate, and ours are not. In the future, we look to clean out the interior in hopes that it will make it easier to maneuver the arm.

Evidence of Progress:

Testing the Base Stepper Motor:



24 Pin Power Source Diagram:



Research:

- https://learn.sparkfun.com/tutorials/easy-driver-hook-up-guide/all#hardware-hookup
- https://makezine.com/projects/computer-power-supply-to-bench-power-supply-adapter/
- http://ww.theoldrobots.com/book21-9/MiniMover5.pdf

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 1/12/2021

Goals

Long Term:
- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:
- Present Progress
- Adjust potentiometers
- Order Materials

Progress

Today's Progress:

Today, we gave our presentation for our progress from 12/23/2020 to 1/7/2021. After all presentations were finished, we took Mr. Levin's suggestion to adjust the potentiometers on our stepper drivers (This was brought up during the Q&A period of our presentation). Using a small flathead screwdriver, we turned the knob on all four potentiometers to "max". Once this was complete, we prepared for our future project sessions by ordering new materials necessary to continue our project. We first placed an order for longer jumper wires that would extend far enough to mount the breadboards behind the base of the arm. This way, they wouldn't interfere with the space in front of the arm and appear in the frame of the camera. We also researched possible cameras and decided on the C270 HD Webcam from Logitech as it had a decent resolution and field of view while still being easy to mount on our arm.

Evidence of Progress:

Materials Ordered:
- Male to Male Jumper Wires; 40 cm
- Logitech C270 HD Webcam

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 1/14/2021

Goals

Long Term:
- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:
- Mount and test camera on the robotic arm

Progress

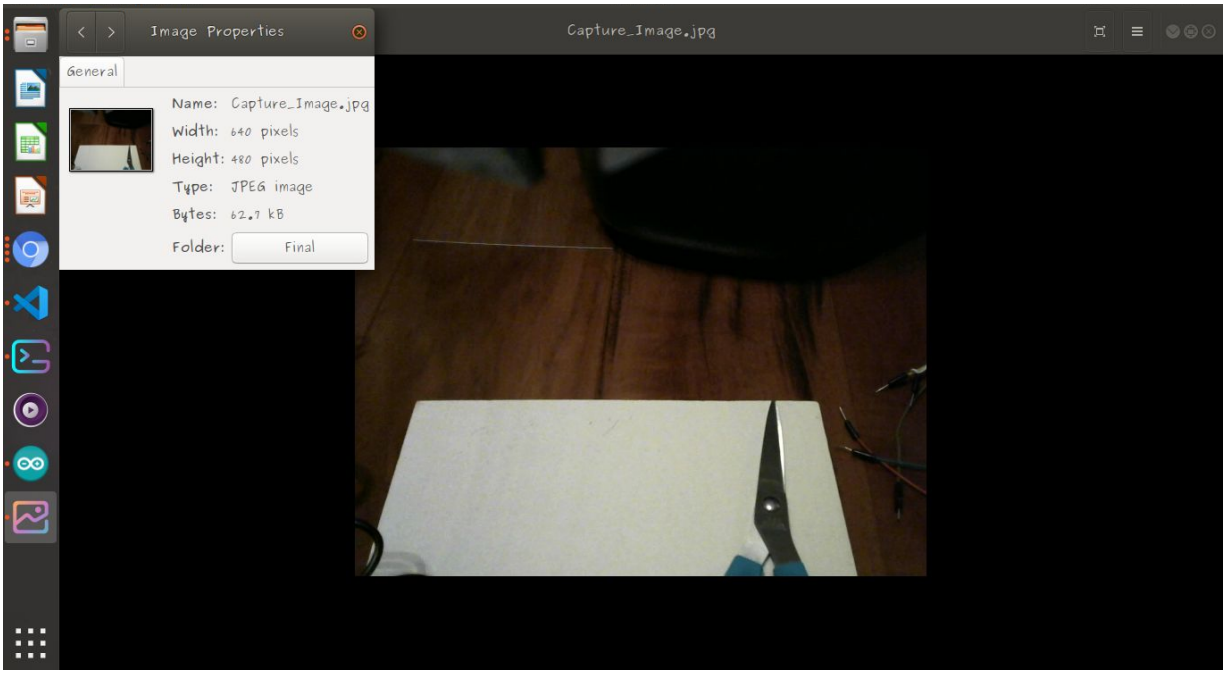
Today's Progress:

Today, our main objective was to mount and test the functionality of our camera. As for mounting the camera, we tested several locations including the base of the arm, above and below the forearm, and a "Bird's-eye view" directly above the arm (this would have to be mounted externally). Of all of these possible locations, we decided that the underside of the forearm, slightly angled downwards, would be the best location as it best captured the area where the arm had the ability to reach (When mounting at the base, much of the space directly below the camera was out of the frame, when mounting above the forearm, the hand was in the frame, and when mounting above the arm, externally, the camera did not adjust its angle with the movement of the arm). To attach the camera, we used a pair of rubber bands and an empty cardboard box (used to create extra tension in the rubber bands) as shown in the image provided below. Once we decided on the location to mount the camera, we edited the *Capture_Image* program to use the external webcam by changing the parameter of cv2.VideoCapture() to 1 (0 indicates the built-in webcam on the computer, 1+ indicates the external webcams connected). After this adjustment, we ran all of our previously written code to test the new camera and successfully were able to take a picture. From this picture, we recorded the dimensions of the frame (640, 480) which will be used later on when we write out our code for the path of the arm.

Evidence of Progress:

Camera Mounting Setup:

Image Captured from Webcam (Image Properties Displayed as well):



Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures	
	
Emma Mascillaro	***Ashton Lukyanovsky***

Date: 1/19/2021

Goals

Long Term:	**Short Term:**
- Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Figure out at a high level how to implement speech recognition & object detection code with mechanical components

Progress

Today's Progress:

Today, our main objective was to figure out how we can connect our speech recognition and object detection code with our mechanical arm controls at a high level. Initially, our plan was to use pyfirmata to write code that we could flash onto the Arduinos, but as we did more research on the firmata software, we found that it hasn't been well maintained (it appears to not have been updated since around 2013) and it has little support for stepper motors. For this reason, we did more research and brainstorming and decided to use Arduino scripts to control the stepper motors (Arduino scripts would indicate the direction and duration of the rotations that the different steppers needed to make) and compile them in the terminal using the os.system() method when each movement was needed in our *Stepper_Path.py* program. This way, all of the logic could be handled in Python and the motor rotations could be handled with Arduino code which had better support for stepper motor control.

Evidence of Progress:

Pyfirmata Research:

- https://scruss.com/blog/2012/10/28/servo-control-from-pyfirmata-arduino/
- https://github.com/tino/pyFirmata
- https://forum.arduino.cc/index.php?topic=594124.0
- https://pyfirmata.readthedocs.io/en/latest/pyfirmata.html

Using Python to Execute Shell Commands:

- https://stackabuse.com/executing-shell-commands-with-python/

Github Link:

https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 1/21/2021

Goals

Long Term:

- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:

- Determine the programs needed to be written in Arduino code
- Adapt example code to create necessary programs
- Test Arduino code on the arm

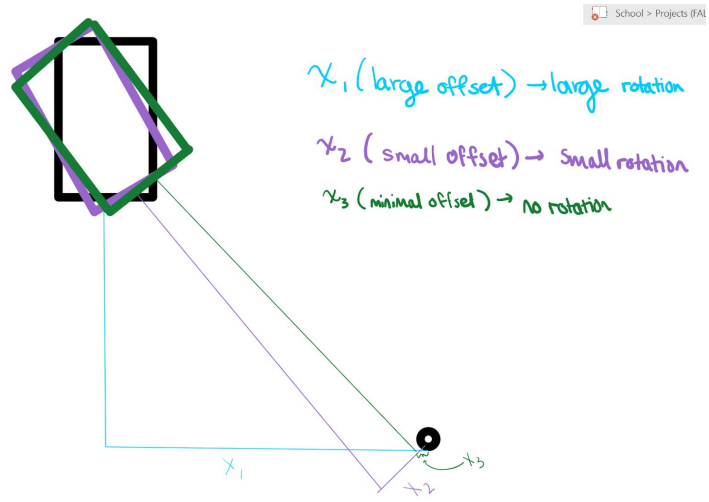
Progress

Today's Progress:

Today, our main objective was to figure out which programs we needed to write in Arduino code and to adapt example code to fit our requirements. As we decided on during our last project period, all of our stepper motor rotations would be controlled with Arduino code. This includes opening and closing the hand, rotating the base, and extending and retracting the elbow and shoulder. For this reason, we decided that we would need a Hand_Open program and a Hand_Close program to control the hand by opening and closing it by a set amount. For our base rotation, we decided upon a path as shown below (under *Base Rotations Plan*) where the number of steps in the rotation would depend on the offset distance between the object's x coordinate in the image and the center of the image taken. For this reason, we created 10 Arduino scripts for 5 different step durations for turning left and turning right, controlling the base rotation. Since the base and the shoulder used the same pins, we were able to reuse one of the intermediate motions for the base, *BaseShoulder_3_Right.ino*, to also control the extension of the shoulder. Lastly, we reused the same intermediate motion for the base/shoulder, *BaseShoulder_3_Right.ino*, with updated pins in order to write our code to lift the elbow. Extracting and adjusting portions of code from the Sparkfun website which we used to test on our arm during previous project sessions, we were able to create and successfully test our own modified scripts on the arm.

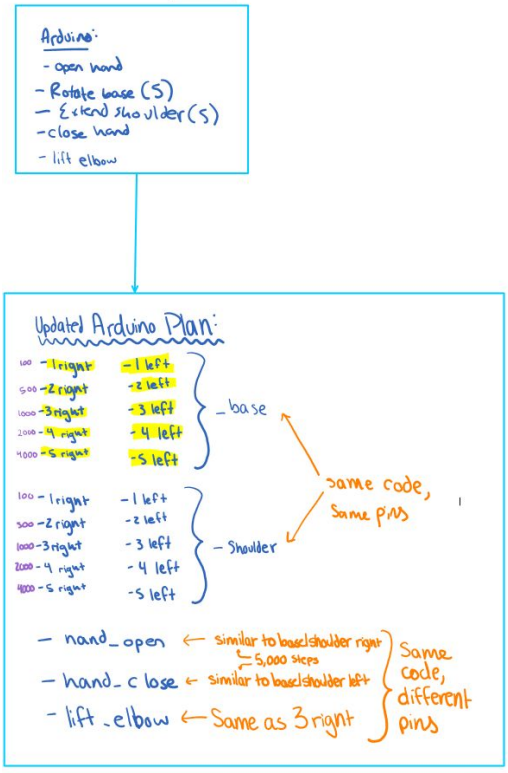
Evidence of Progress:

Base Rotations Plan:

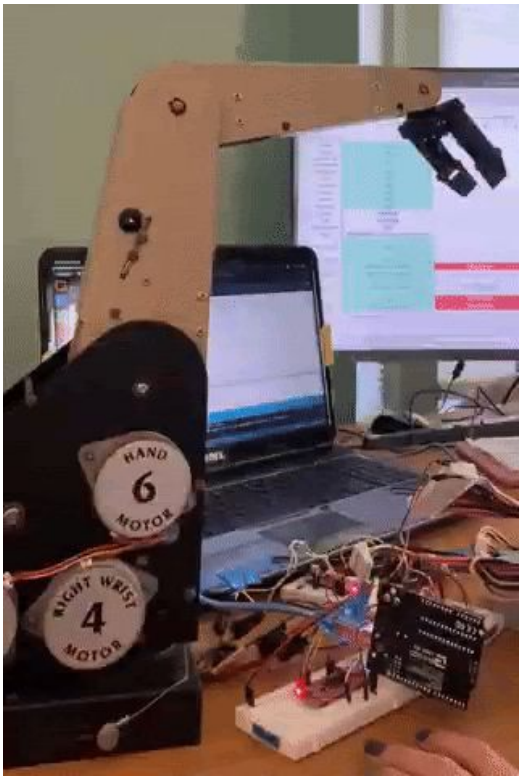


Programs Overview:

Necessary Arduino Programs:



Retracting Shoulder Script Test:



Research:

- https://learn.sparkfun.com/tutorials/easy-driver-hook-up-guide/all

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures	
Emma Mascillaro	***Ashton Lukyanovsky***

Date: 1/22/2021

Goals

Long Term:	**Short Term:**
- Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Replace jumper wires - Learn how to compile Arduino code in the terminal

Progress

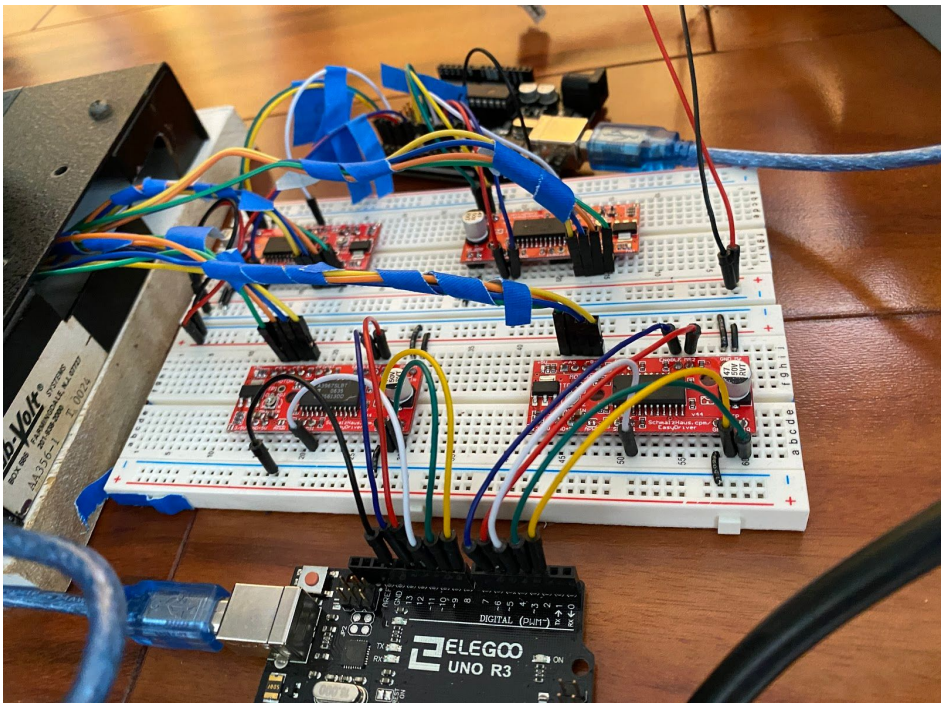
Today's Progress:

Today, our primary objective was to replace our existing jumper wires with wires fitted to the board for all breadboard to breadboard connections and longer jumper wires for the arm's stepper motor wires to motor driver connections. For all breadboard to breadboard connections, we decided to replace our jumper wires with wires fitted to the board as the abundance of jumper wires made the boards more illegible when the connections were typically only spanned from the rails to adjacent pins. For the wires connecting the drivers to the stepper motor leads in the arm, we needed to replace those with longer wires (40 cm) so that the breadboards could be mounted behind the base, staying out of the camera's frame when capturing images.

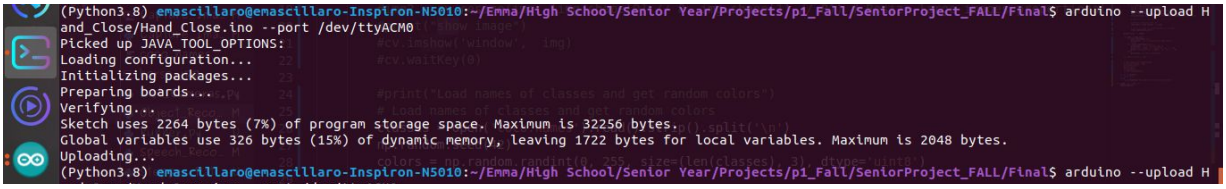
Once we replaced the necessary jumper wires, we moved on to look into how we could compile Arduino code in the terminal as opposed to compiling it in the Arduino IDE. After doing some research, we found the terminal command, "arduino --upload sketch/sketch.ino --port /dev/ttyACM*" which allowed us to compile directly in the command line. By being able to compile our Arduino code directly in the command line, we are now able to use the Python command, os.system(), to compile our Arduino code directly from our Python code whenever necessary.

Evidence of Progress:

Replaced Jumper Wires with Breadboards Behind the Arm:



Compiling Arduino Code in the Command Line:



Research:

- https://arduino.stackexchange.com/questions/15893/how-to-compile-upload-and-monitor-via-the-linux-command-line

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 1/26/2021

Goals

Long Term:	**Short Term:**
- Implement all programs to interact with the robotic arm - Complete code to control the movement of the robotic arm	- Write Stepper_Path.py program to integrate speech recognition and object detection with the arm's mechanical movements

Progress

Today's Progress:

Today, our main goal was to write our code for the path of the arm which would integrate our object detection, speech recognition, and stepper control programs. Our first step in our path was to open the hand, so we opened the Stepper_Path program by compiling the Hand_Open.ino Arduino script. Our next movement in the path was to adjust the base to align with our object in x, so we began to write out our first method to go about that. However, our first method had an excess of if statements and didn't have any tolerance, so we commented out that method and started again. In our second attempt, we had five if/elif statements to determine how large of a turn was necessary followed by an if/else where we determined the direction of the turn and used an f-string in the os.system method to compile

the necessary Arduino code. Once the base rotation code was done, we decided that for the sake of simplicity given the time we have left to complete the project, we would simplify the path and make the assumption that once the x coordinate was aligned, simply extending the shoulder until the arm reached the floor would get us to the y and z location of the desired object as well. For this reason, we were able to follow our base rotation by compiling our Arduino program to extend the shoulder, followed by our Arduino program to close the hand, and lastly our Arduino program to lift the elbow once the object was in the hand.

Evidence of Progress:

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky

Date: 1/28/2021

Goals

Long Term:
- Implement all programs to interact with the robotic arm
- Complete code to control the movement of the robotic arm

Short Term:
- Create functions within code
- Create a runner program
- Test code
- Write an abstract describing our project's process and accomplishments

Progress

Today's Progress:

Today, our main objectives were to adjust our code by creating functions within our programs, create a runner to compile all of our code at once, test our code's functionality, and write our abstract. By creating functions within all of our programs, we were able to better integrate our programs as we could better access and compile specific tasks when necessary rather than recompiling entire scripts. By doing this, we also made it easier for ourselves to create a runner program where we could call the functions in all of our programs in the correct order, resulting in our code running cohesively with only one command. Once we finished making

our code adjustments, we began to test our code and started to have some promising results, but unfortunately, after a few tests, we had to stop running code on the arm as the pulley wire for the base got lodged between the pulley wheel and frame, causing the rest of the wire to have excessive slack. Since we didn't want to damage anything permanently, we decided to stop testing at that point, though we were able to see the hand close and the base adjust with the desired object prior to the incident with the pulley wire. After testing, we moved on to drafting our abstract.

Evidence of Progress:

Abstract:
Combining speech recognition, object detection, and the mechanical movement of a robotic arm, the goal of this project was to create an assistant that would help people with disabilities to pick up and interact with objects. In our project, we created a prototype for code and circuits that could be used to achieve this task. In order to do so, the project was split into three main components: speech recognition, object detection, and the mechanical movement of the arm. For the speech recognition component of the project, a pre established library and natural language processing was used to transcribe voice to text and determine the objects of interest that the user requests. The object detection component of the project used an external camera to take a picture of the area in the arm's range of motion. From there, machine learning and computer vision were used to determine the locations, sizes, and names of the objects in the surrounding area. Lastly, in the mechanical movement of the arm component, circuits were built to connect Arduino microcontrollers to stepper motors, utilizing stepper motor drivers which allowed for ease of control. Speech recognition and object detection code was utilized here with an algorithm for path control to instruct the arm on how to approach the object of interest. In this project, the arm successfully approached and grasped the objects of interest, and can be used as a starting point to research more efficient prototypes.

Github Link:
https://github.com/emascillaro/SeniorProject_FALL

Signatures

Emma Mascillaro

Ashton Lukyanovsky