**Senior Design 2 Final Report**

* Abhishek Barla
* ID # 900559822

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Project Name:**

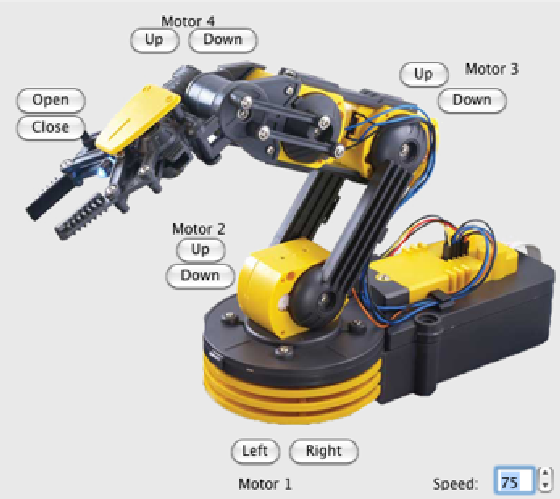
Voice Controlled Robotic Arm

**Team Members:**

This was an individual project.

**Description:**

The project consists of an OWI Robotic Arm Edge whose remote controlled is substituted by a Microcontroller which is controlled by a Laptop running Linux OS via a USB cable. The Arm consists of 4 different motors - one for shoulder rotation, one for shoulder movement (up and down), one for elbow movement (up and down) and the last one for wrist movement (open and close). The Robotic Arm accepts Speech inputs for movements of its different motors. The following picture describes the different motors. The Robotic Arm also has an LED Light in between the claws of the wrist which is used to use the Arm in dark.



**Detailed System Specifications**

**Device Setup and Configuration**

All 4 motors and the batteries of the Robotic Arm are connected to the Microcontroller using specific jacks. Firstly, the program checks if a device is connected to the USB control of the laptop and throws an error if a device is not found. The program is written in Python and *PyAudio,* a Python library to utilize the Microphone on the laptop as an audio source is installed. The program then checks if this library is installed. Once these two checks are done, this configuration is set active and the program runs to the next stages.

**Speech Recognition**

Speech Recognition Setup and Libraries

The first part of the project deals with recognizing the speech inputs. The project utilizes the microphone on the laptop to record the speech input in a buffer stream. 1024 frames of audio samples are taken into one buffer. Then the audio sampling is done. For the audio sampling, the data is sampled at 16 kHz in the 16-bit Int sampling format. The sampling of data is done using the *PyAudio* library which accepts the Speech from the microphone and stores it into a buffer sample.

Accepting Wave file Format

Instead of using the speech inputs from the microphone, further research can be done for using speech words stored in a wave file. The function for this is also written in the program code. This function uses a buffer size of 4096 as the input stream of a wave file is longer than that of a microphone input.

Differentiate between Speech and Silence

Next, to differentiate between a speech input and silence, we need to create variable for duration of quiet time and duration of speech time. In this program, the quiet time is set to 0.5 seconds and the speech time is set to 0.8 seconds. The program should also be programmed to accept speech of a certain energy. In this program the minimum threshold energy used is 100. This allows you to speak in your normal conversational tone. Any tones above or below the normal tones would not be accepted.

Listening and Recording Audio

So, the microphone is activated to listen to the speech input. The listening time and the quiet time is specified and programmed accordingly. The Audio energy is set and a function checks if the audio energy of the samples is equal to the set energy. Only samples whose energy is equal to the specified energy are recognized, everything else is neglected as silence. Each frame is filled with samples of the input and then sent to the next level, where they are converted to the FLAC format.

FLAC Conversion

The program was coded to be used in both Windows and Linux OS, therefore, before converting the samples into Free Lossless Audio Codec (FLAC) format, the program checks to see which converter to activate based upon the OS. The samples are converted to FLAC format as the Google API uses only FLAC inputs.

For the FLAC conversion, the FLAC converter utility tool (found online) needs to be installed on the computer, based on the Operating System. The program throws an error if the FLAC converter is not installed. The converter is usually programmed in C language and the converter is called from my program to convert into FLAC format. After the conversion,

Using the Google Speech Recognition API

Once the audio input is sampled, the program uses the Google Speech Recognition API for recognizing the speech. To use this API, firstly, we need to register in the Google Developers website and receive a key. This key needs to be sent along with the data every time we use Google’s API. There are limitations on the number of times a key can be used per day. Furthermore, Google API gives us the option to change the language and dialectic of recognition. In this program, I have used US English as the language, but most of the languages that Google uses in its search engine can be used for the API. A list of languages and their codes can be found online.

Next, the speech word is recognized using the API. The connection to the API is made through a URL link in the code and a request function is written to send and receive data from the API. Once the sent data is recognized by the API, the recognized word is sent in the form of a string, back to the program. If the speech word makes sense, then the word is used. Else if the speech input is not recognized then the program throws and error. The program checks the confidence in each prediction and re-confirms it with Google.

Listening in Background

The program listens to the speech input in the background. So, everytime you speak something, you needn’t run the program as the program keeps running in the background. This helps the speech recognition for projects in which listening in the background is needed, such as dictation. In this project, listening to the background is important as the Robotic Arm can be controlled whenever needed.

Errors in Speech Recognition

Make sure the computer has a good internet connection. The program throws an error for no internet connection. The program also throws an error when your maximum quota for the day in using the API key is reached. If Speech input is not recognized then the program throws an error.

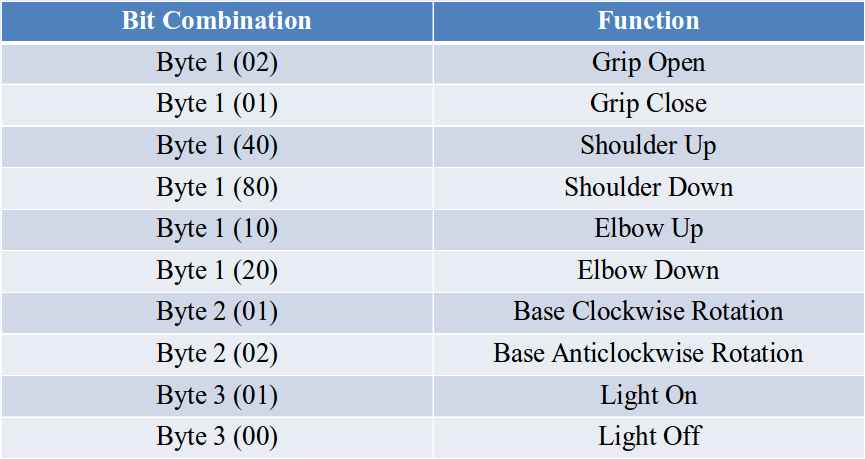
**Microcontroller Programming**

Basic Setup

The basic setup of the microcontroller includes connecting the batteries of the Robotic Arm, the motors and the ground to the Microcontroller pins. Any microcontroller such as the Raspberry Pi or an Arduino can be used. This project was performed using a Raspberry Pi but since suitable Speech recognition libraries weren’t available, the microcontroller had to be changed to a custom one.

Sending the Data

Sending the data across the USB is another challenging part of the project. For this, *pyUSB*, another Python library is installed on the computer. Since the check for a USB connected device has already been done, the next step is to send the data to the microcontroller. The program uses a byte-type data writing module to send data. This includes sending the data in packets to the microcontroller. Each packet consists of 3 bytes. The first byte defines most of the movements (Shoulder up and down, Elbow up and down, Wrist open and close). The second byte contains data about shoulder rotation and the third byte contains information about the LED Light. Each byte is a two digit number and the bytes assigned for each function used in the project are depicted in the following picture:



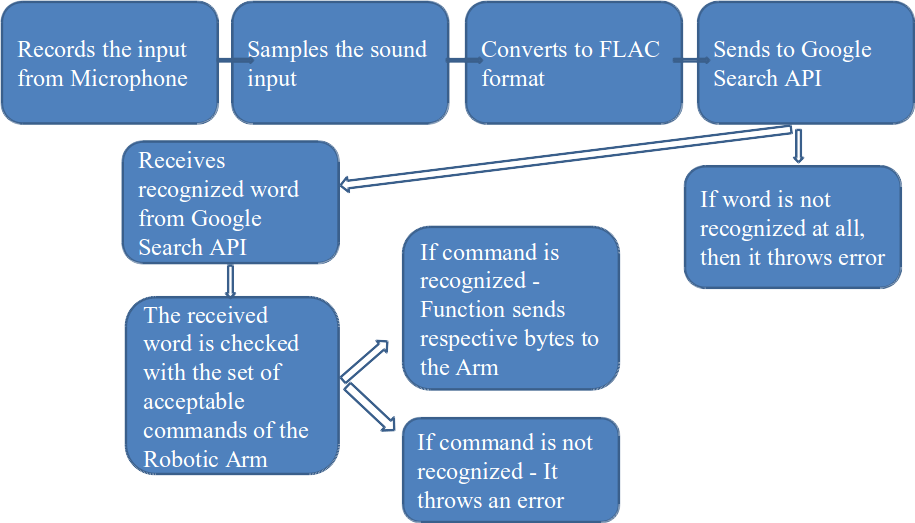
Now, that the data is defined, the next step is to transfer this data to the Microcontroller. To do this, a control (write) function is used. The Write data contains the Bit Request Type, Bit Request, Write value, Data packet and the Write Index. This function is a pre-defined function in the *pyUSB* libraries.

It can be noted that bit combination used in this project allows for performing two functions at the same time. For example *80* stands for *Shoulder Down* and *01* stands for *Grip Close*, therefore *81* would stand for *Shoulder Down and Grip Close*. Therefore, simultaneous functions of the first byte can be performed using this module.

For further research, more functions and motors can be added to the Arm and the Bit combinations can be changed accordingly. In total, for the first byte, there can be 100 possible bit combinations. More on this topic would be discussed in the *Future Research* section of this report.

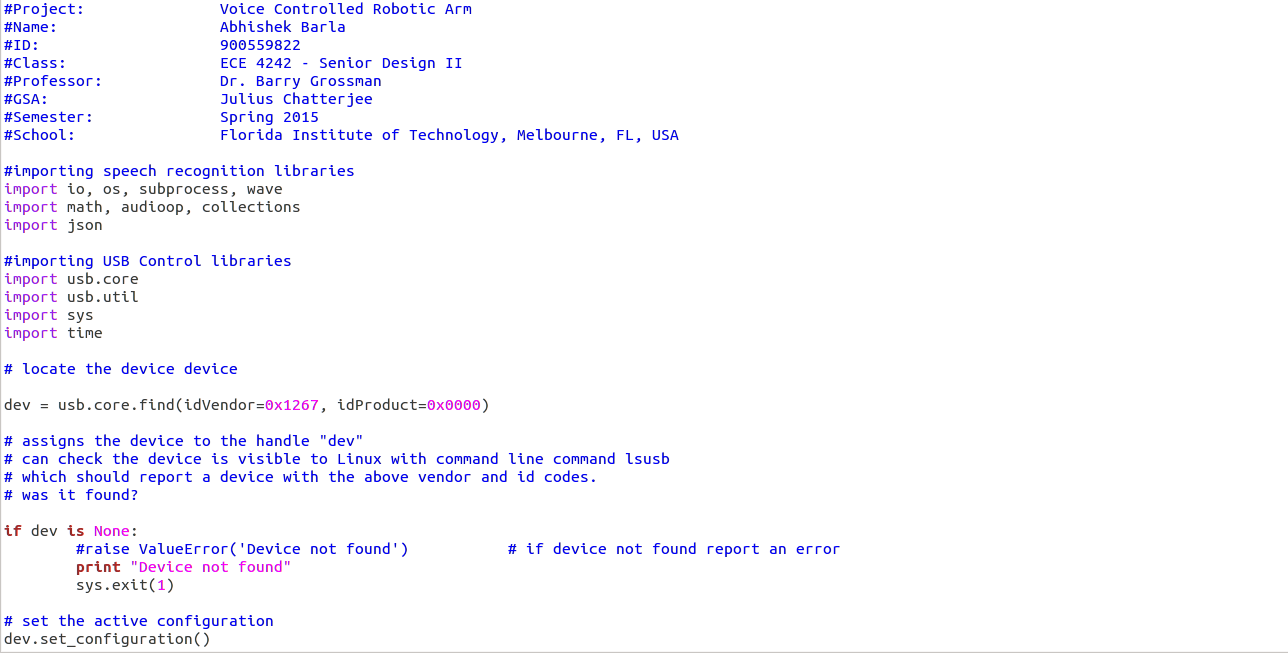
Controlling the motors

The Write function defined above activates the motors. The motors keep running until they are stopped. This involves the risk of burning the motors out. Therefore, the program sleeps (waits) for a specific time (one second in this project) and then sends a blank data packet (where all the 3 bytes are of zero value) to stop the motors. Suppose the data packet contains the code to rotate the shoulder base clockwise, then the motor would be running for one second (as specified) and the angle of movement would be the distance the rotation covers in one second. To increase the angle, the run time of the motors should be increased. It should be noted that the Robotic Arm has physical constraints so turning on the motor for too long would break the Arm.

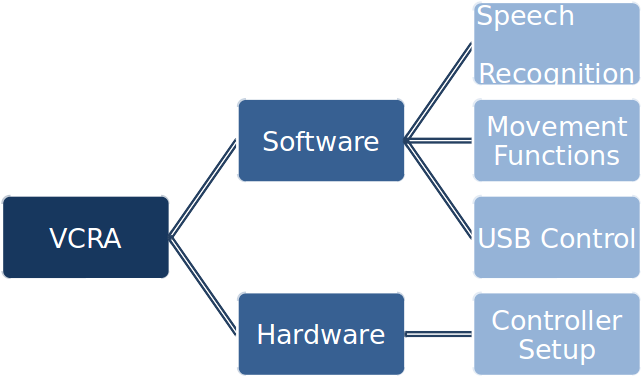
**Detailed Block Diagram:**

**Full Source Code:**

The project was coded in Python and the source code of the main program is as follows. The source code of the libraries can be found online and can be downloaded.

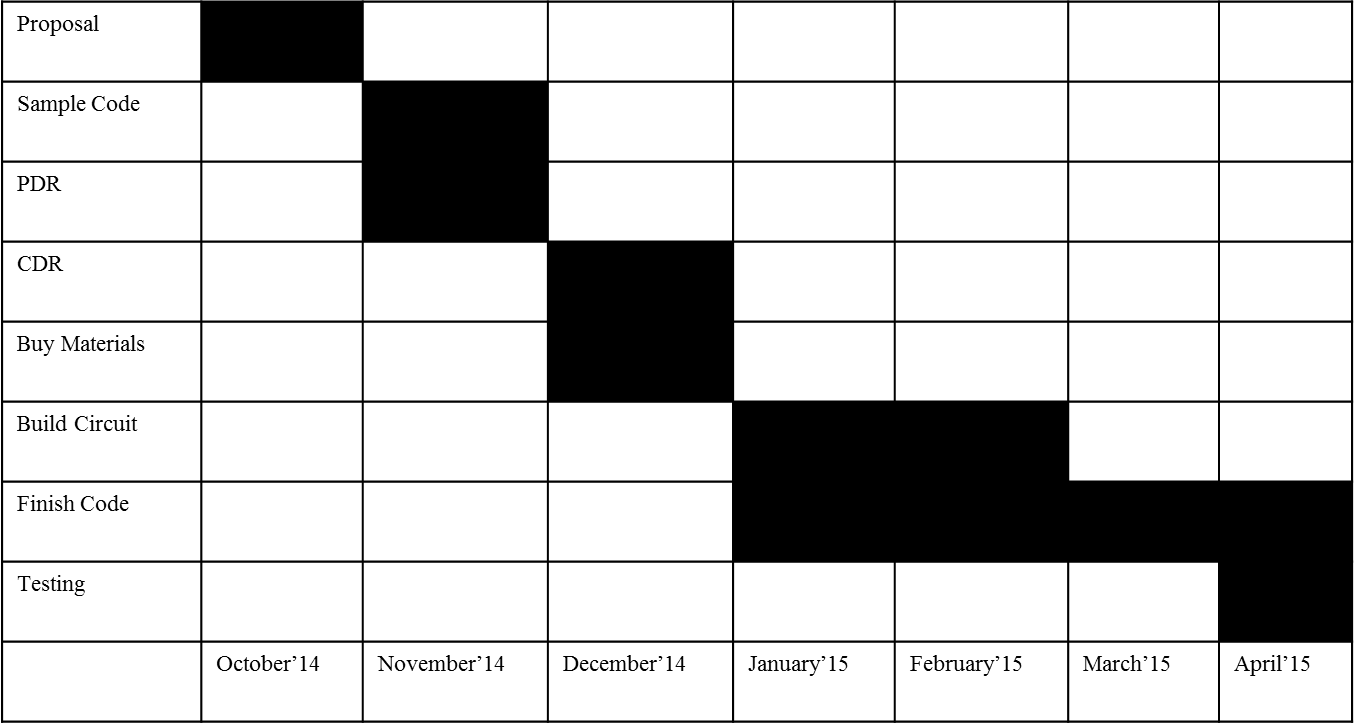


**Project Organization Chart:**



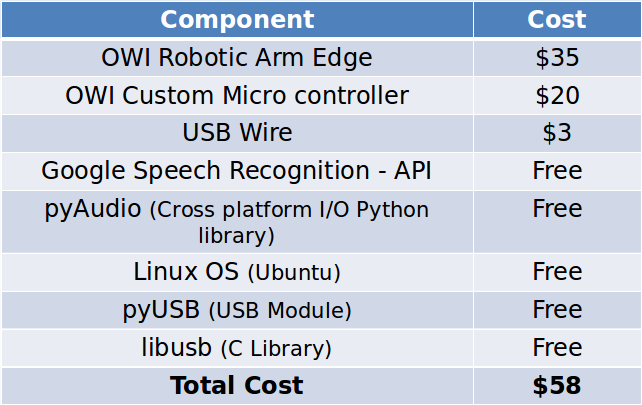
**Gantt Chart:**

The project was done in many phases due to the complexity of the project. The following Gantt Chart explains the timeline of the project.



**Final Budget:**

Since the aim of this project was to build a low cost Voice Recognizing Robotic Arm, most of the components used were open source and free. The only parts that cost money were the hardware parts. The following is the final budget table of the project.



**Other Possibilities:**

**Software**

Speech Recognition is a complex subject and it takes a lot of knowledge and programming to build a successful Speech Recognition Engine. However, due to the scope of this project, an outsourced Speech Recognition was used. But, even this Speech Engine needed to be configured and modified according to our project. But, while trying different Speech Engines, it was noticed that some Engines don’t function suitably with the rest of the program. Furthermore, the accuracy of some of the Engines in recognizing a speech input is very low. The project was tried using various speech recognition engines but none of them worked perfectly as they failed to recognize the control commands. Some of the Speech Recognition Engines and APIs used in the process are:

* Windows Speech SDKs
* Julius Speech Recognition Engine
* Google2Ubuntu Voice API from Google
* TalkTyper, an online application for Speech Recognition
* Tazti, a PC application to recognize predefined commands
* Bing Speech Recognition Control

Modifying our program in accordance to the aforementioned software and other commercially available software can be done but was out of the scope of this project. It required more attention to detail and could be done in more time or with a group of people. So, other possibilities for the software part of the project would include using these software and trying the same project out. This may or may not increase the accuracy of recognizing the input but would at the least be a new learning experience.

Furthermore, the USB control function and the write module used in this program can be modified and replaced with other write module available on the internet. Some may be easier to use but would require more programming. However, implementing the same project with those functions would be a new learning experience.

**Hardware**

The Hardware part of this project was troublesome. The project was initially done using the Raspberry Pi Microcontroller as it was a low cost option. Controlling the Robotic Arm using the *Pi* was successfully implemented but the Speech Recognition API was not compatible with the Microcontroller. Initially, the Speech Recognition software was to be embedded in the Micro controller so that the project would be laptop free and the Speech controls could be sent to the motor controllers internally as both of them would be on the same microcontroller. But, due to not finding an appropriate low cost and effective Speech Recognition Engine that could be embedded onto the Raspberry Pi Microcontroller, the idea of the Pi was scrapped.

Instead, a OWI custom Microcontroller was used which was connected to a laptop (where the Speech Recognition program ran) via a USB. However, the same project could be implemented using a Raspberry Pi and the Speech Recognition software could be embedded onto the Pi. This would require a team of people and more time as it would need more programming.

**Future Research:**

This project can be taken forward in many ways both individually and with a team of members. Some of the ways in which it can be taken forward are as follows.

**Measurement of Degrees of Movement**

In this project, the motors were switched on for a specific amount of time and then switched off. So, the degree of movement in this case was what the motor could do in that amount of time. But, a more controlled movement would be appealing. For this, a circuit can be designed that controls the motors and the degrees of movement. This would be appealing as it would enable us to move the arm based on distance instead of time.

**Detecting Physical Constraints**

Another major problem faced in this project is not being able to detect the physical constraints. The Arm keeps moving when you command it to move the Shoulder. But, the Shoulder movement has a physical restriction until its base. This cannot be detected in my program and thus when the Shoulder is asked to move more when it reaches its constraint, there is a risk of breaking the arm. The most dangerous risk is in rotating the shoulder. Since the rotation is based on the specified time, a rotation more than the physical constraint could result in breaking the connection wires. Therefore, it is necessary to detect the physical thresholds of movements and this could be an interesting research.

**Differentiating between an Egg and a Metal**

As discussed, the motors run based on time, therefore it is difficult to be able to control the movement of the parts of the Arm precisely. One such part is the claw. The claws are important to the Arm as they help in holding an object. The pressure with which they hold an object cannot be controlled by this program. For example, if the Arm were to hold an egg, it would run the motors for a specified amount of time, but what if the Egg’s diameter is less than the movement of the claws. This would result in the breakage of the Egg. However, if a metal were to be used and the motors ran for more than needed, it would not result in damage to the metal. Therefore, the pressure that the claws apply on an object should be controlled. This would help to differentiate between an Egg and a metal (different pressures to be applied). For this the motors should be controlled based on distance and not on time. Furthermore, a pressure sensor should be used in an analog circuit to measure the pressure applied on object and some kind of module should be made to differentiate between objects. This can be a very interesting research for this project. This might also pave way for commercial use of the project.

**Designing an Application to Control the Arm**

To use this project, one had to speak into the microphone and had to be close to the laptop as the USB cable extends to a limited distance. But, to be able to increases the usage of the Arm, the Speech recognition and the Microcontroller Circuit should be far away and the signals should be sent wirelessly. This can be done if an iPhone application could be developed. You could speak into your phone and the app would recognize your input using your phone’s inbuilt microphone and the Google Voice API and could send a signal to the Microcontroller via Bluetooth or other Communication modules that the Microcontroller is capable of handling such as WiFi. This could be instrumental in using the Arm from a distance. This research would also be very interesting and would pave way to commercially use this project.