CPS342: Embedded Linux

Final Report

Web Controlled Boom Camera (WCBC)

Group Name:

Group Members	Department	Major Contribution
Mitchell Wagner	CE	3D Design/ Servo configuration/ Building of Arm/ GUI implementation
Paul Boston	CE	Servo configuration/ Building of Arm/ GUI Implementation
Jose, Da Costa Filho	CS	Website Development/ Camera control/ GUI Implementation /Gallery Structure
Charles Palsa	CS	Website Development/ Camera control/ GUI Implementation/ Website Helper
Shamiul Islam	CS	Web Help/ Pictures Building

Course Professor: Easwaran Chirakkal

05/05/2015

Table of Content

1. Introduction:

The Web Controlled Boom Camera (WCBC) was a project created in the Embedded Linux computer programming class. This project was done to create a web page that has a live feed from a camera that can be controlled to move in any direction. Our group took this to mean that a servo motor would be connected to the pi and a person could control this motor and see where it is pointed via the web page. Our project is a functioning camera arm that has 3 servo motors to control the camera direction, this enables the user to see in every direction possible and turn on any axis. These servos then communicate with the pi and are able to be controlled from the web page with a built in GUI to control the location of the servos and see the live feed of the camera.

1.1 Project Description:

This project is to design a system that will be able to be accessed through the web and the user will get live feed and be able to control the camera 360 degrees in every direction. Our system that we created is a camera arm with 3 different servo motors, these motors then connect to the pi. The pi is connected to the internet and is hosting our website that displays our GUI that will control everything.

1.2 Project Goals:

Our project goals are to have a working camera arm that will operate from the internet using a GUI to communicate with the servo motors and the camera. The servo goals are to design a rig to hold 3 motors to complete the arm. Second create a way to connect all the servos to the pi. Third, have the servos accurately move to the degrees desired and not get the wires tangled. Our goals for the camera is to hopefully display the camera feed in a high FPS (Frames Per Second) so that viewing the feed looks normal and would be able to take snapshots of a live feed and store them. We want to also make sure there would be a low delay time when accessing anything on the website. The website goals are that it will be mobile friendly and that the camera feed will display natively as well as all the controls for the servos will relay back to the servos and actually work.

2. Implementation:

The implement of the project was broken up into several different parts: website, camera, servo's and stand.

2.1 WebSite:

The web site was going to be written in different programing languages to be able to be able to be used on multiple portable devices.

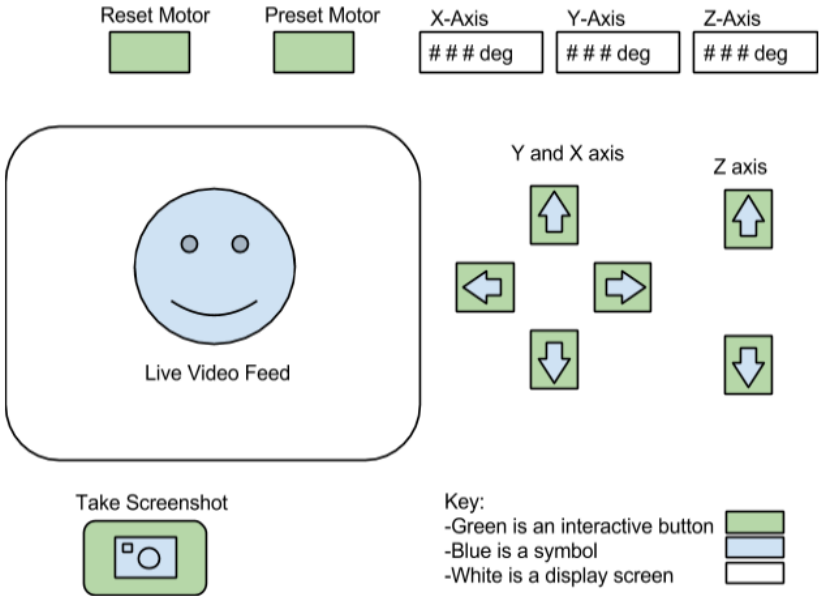


Figure 1: Preliminary Website Design

Figure 1 helps to show how the design for the end result for the website will look. This has the ability to use buttons to control the motors as well as type the location desired and display where you are. There is also a screenshot button and reset the motors to the origin location (0, 0, 0).

Since bootstrap was the desired web server to implement on we needed to layout the locations for everything. Bootstraps ability is great because of the structure to split everything into different rows and columns. This is a very useful tool because this allows a great look and feel that can be used for many different screen sizes and helps in the mobile friendly feel. The diagram of the website is represented in Figure 2.

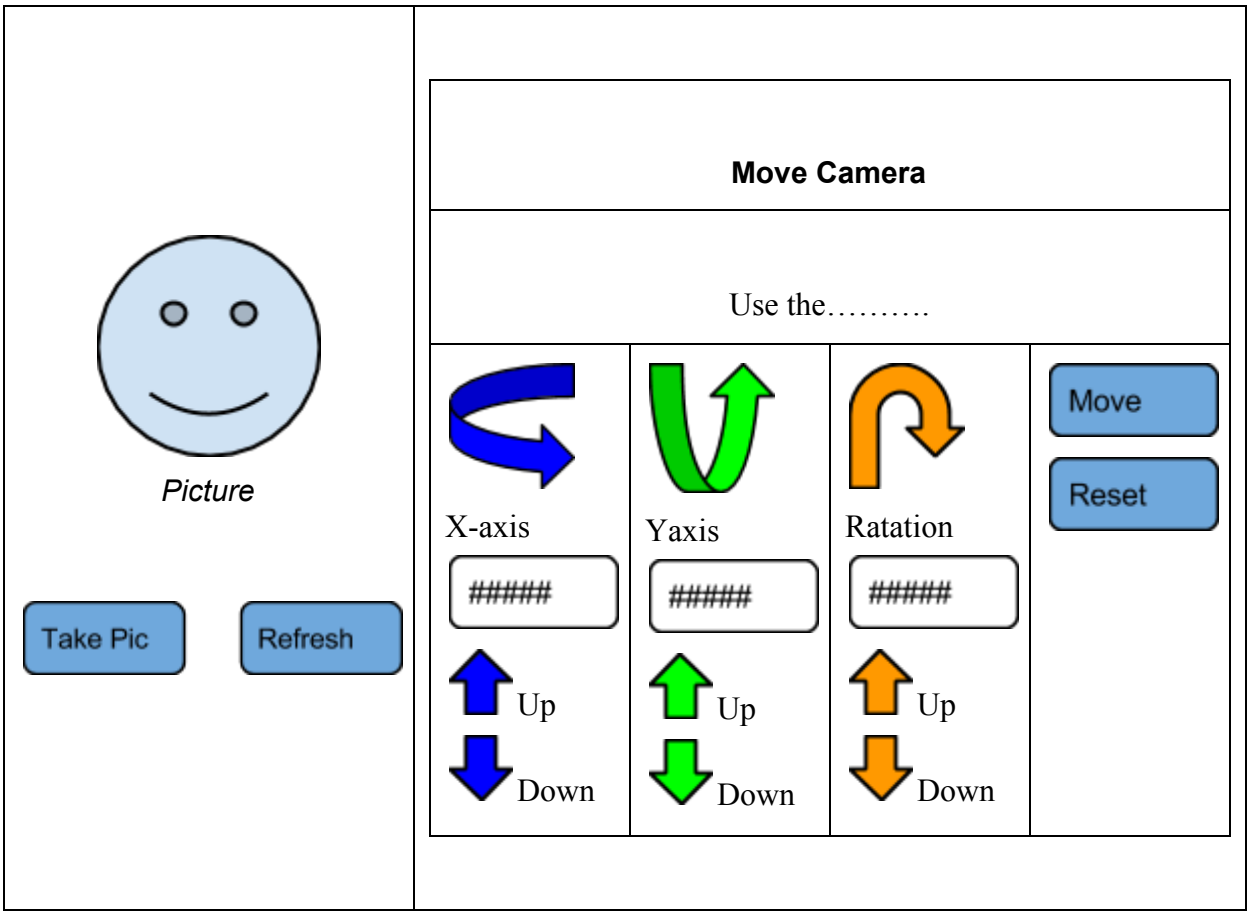


Figure 2: Bootstrap Layout

2.2 Camera:

The camera was going to be the raspberry pi camera so that the code should be able to sync up much more smoothly. We would also need a raspberry pi camera extension cable to actually make the camera reach to raspberry pi from the camera rig. This camera should sit at the end of the rig on a platform.

To be able to implement the camera adafruit's website as well as many more were used to help understand how each part is used to get a live update on the website. These resources can be seen in the appendix.

2.3 Servo:

The servo motors were determine to be a 5V high torque Continuous motion 360deg motors and that it was decided that 3 motors would be needed to get full rotation of the camera. This would allow full rotation in any direction across all three axis. These motors have a signal pin that would need to be controlled by a pwm port. We figured out that since the raspberry pi

only has 1 pwm port we would need to get a pwm servo controller board to interface with the pi to control all three. Below in Figure 3 our design for the camera movement and control.

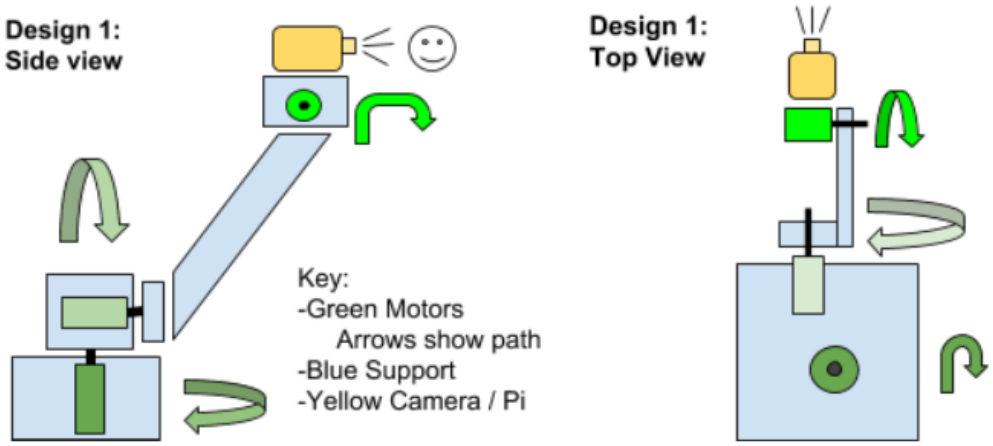


Figure 3: Motor rotation

Figure 3 helps to show how each motor will be operating the cameras movement. This initial diagram was created to help realize the amount of motors needed to get a full 360 degree rotation. As well as where they could be placed.

2.4 Stand:

The stand was decided to be printed from the 3D modeling place. The material used is a hard plastic which is durable and strong enough to support all the devices that it will interact with. We figured that we needed to create brackets that each motor can mount to and can also hold another motor. We also figured that we would need to have a base for all the components to sit on. In figures 3 and 4 are the preliminary drawings of our 3D designed pieces.

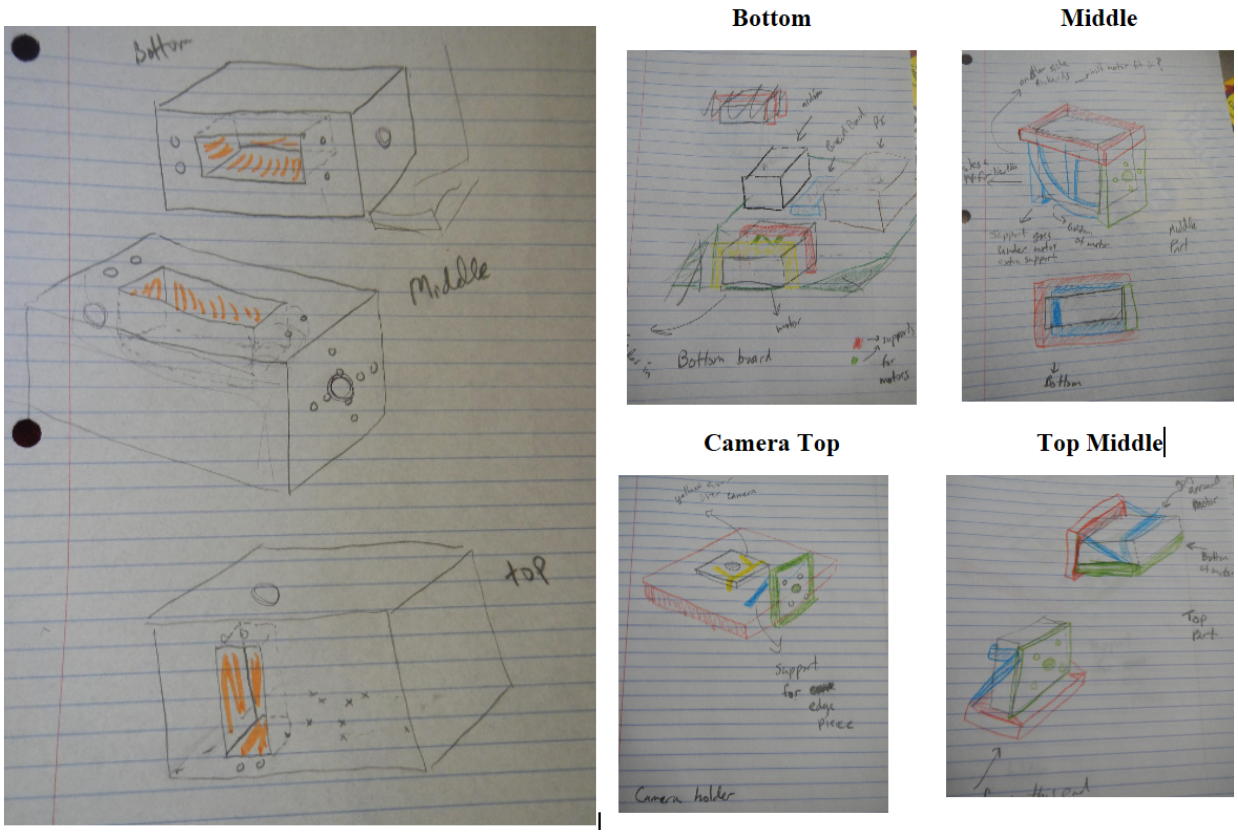


Figure 4&5: Rough drawings for a prototype of the stand

Figure 4 was a design that incorporated just printing out box holder for each of the motors although this idea should work, we were advised to think outside the box. This way to make a product that one wouldn't take up as much material as well as look more appealing. Figure 5 shows a design for a bracket look. This was broken up into all the components to help be more appealing and use less materials, but to check the functionality the design went through a prototype phase before actual printing.

3. Components:

This area of the project consisted of what areas were broken up into their different parts to be able to combine all the parts in the end to get the final product. This would go into how each area of the project would be established and what has been done in these areas in order to get closer to the final end design.

3.1 Website:

The website was preliminary done in bootstrap, CSS and HTML a diagram can be seen at Figure 1. The website GUI and front end will be designed using the Bootstrap framework, and motor/camera controls will be interfaced using python scripts. The screenshots taken by the client will be saved in the server side. PHP scripts is used to save the images and manipulate them on the server. The gallery page makes an asynchronous http request using Ajax. Then, a php script on the server searches for all the images in the snapshots directory and sends a JSON object back to the gallery page that displays the images.

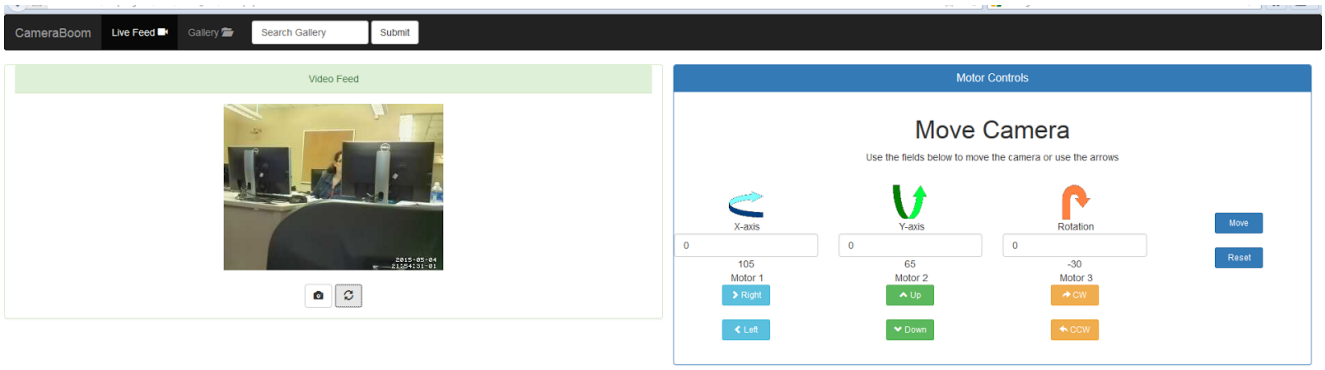


Figure 6: Website Setup

Figure 6 helps to demonstrate all the aspects of the website GUI using a bootstrap framework. Using this framework allows the system to respond to the different screen sizes changes as well as having a good base formate to locate everything in. Figure 7 represents the Control aspect of the GUI. This shows each button with clear representations of the direction the camera will move by the above picture drawing as well as the different glyphicon's taken from bootstrap to depict it as well. There is also the capability of displaying the current motor direction to help the user understand where each motor currently is located.

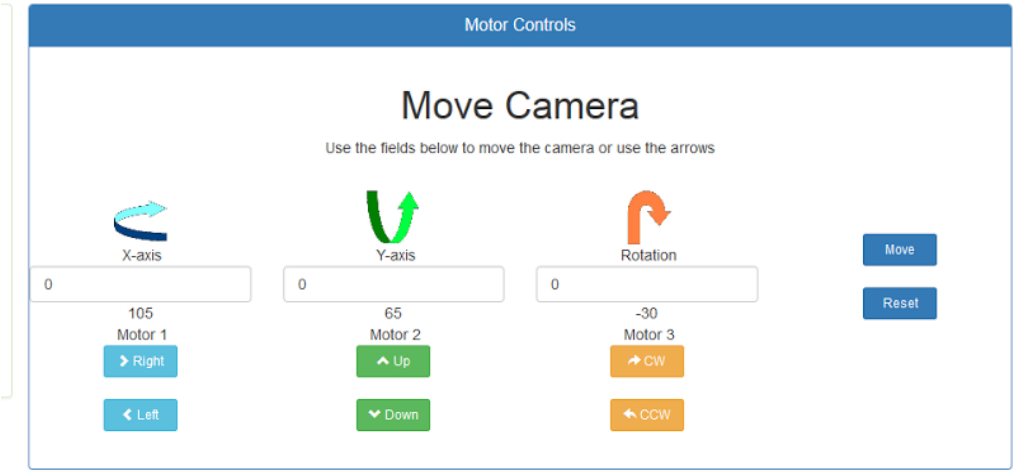


Figure 7: Control GUI

3.2 Camera:

When the implementation for the camera was started it was found to give live stream but very slow rate. This was then where it was decided to start with just having a single image show up and as we get the understanding of increasing the speed we could be able to make the camera run at a faster time this way the amount of frames shown would be more accurate. The video feed will be streamed over http via a motion-jpeg server. This works natively in Chrome, Safari, and Firefox. Internet Explorer does not natively support this feature.

For setting up the camera there was a startup file that was created to initialize that the camera is on and able to take pictures. Another command that needs to be in order to start the camera is one line of code that is initialized on the command prompt.

sudo modprobe bcm2835-v4L2 **(everything is lowercase -->L=l & not #1)**

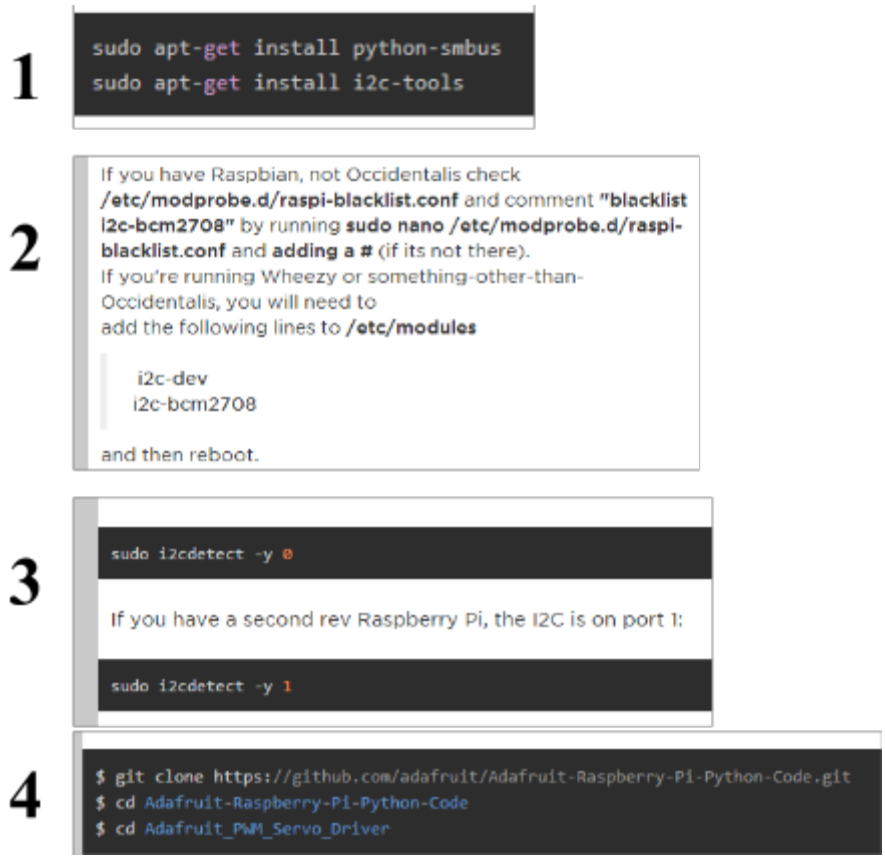
3.3 Servo:

There are several key components for the servo motors. First was the assembling of the motors with the stand. This required the stand to be constructed and measured properly. Then there was soldering of the PWM controller board, this board was important because the raspberry pi only has one PWM port and we needed at minimum 3 ports to control each motor. Then the

main component was all the things that required the coding for the motors to work in ways that the user would desire.

In order to be able to get the PWM servo controller working on the Raspberry Pi we needed to be able to install different function to be able to access the servo controller ports. The location for the installation that needed to be done is in the below website link.
https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/overview

Several key aspects that in the installation were to be able to install on the pi were taken from the web site. These allowed the pi to access the code samples for the movement of each motor.



The code that would operate each motor is located in The Adafruit_PWM_Servo_Driver, The 2 files for the motors are test_WriteFile.py and Location_Motor. These two files need to be to be present so that the motors will operate test_WriteFile.py is the code that can be seen in the appendix of this project. Location_Motor is just a file that will just have the current location for each motor that is grabbed both by the test_WriteFile.py and the website to display the locations.

The motors were started by assembling the entire camera arm but focusing on one motor at a time. This was difficult at first to understand how to get the motor to move just a degree motion and not freely. The problem that would occur would be that the motors continued to move freely and would be the restricted by the wires getting all tangled up and resist future movement, which was discovered when first trying the motors out. In order for the motor to move by degree this required the understanding that the amount of time that was giving for a command to be on would determine the amount of degree that it would move. There was other ways that were tried by changing the pulse length, the best that this method got was to get only 15 degrees of movement at a time. This was discovered when the entire motor would travel 360 degrees in 24 different movements. Then we discovered that by removing the time implementing and using a for loop to represent the amount of times it would cycle through gave a much more accurate degrees of rotation and would then give the user a better control over the entire camera arm.

Figure 8 helps to represent the code that will control a motor. To control each motor a value is inputted and split between 90, 180, 200, and 275 degrees. These values were chosen to be able to have the motor more accurately move between the different angles. This is seen with the representation of the red arrow. The green arrow in Figure 8 shows how the input affects the PWM by changing the range that it will travel from, inessence how long will it be on for.

```
21  if(input<0):
22   y=1
23   input=abs(input)
24  else:
25   y=0
26  if(input<=90):
27   input= int (round(input*.70))
28  if((input>90) and (input<180)):
29   input= int (round(input*.80))
30  if((input>=180) and (input<=200)):
31   input= int (round(input*.88))
32  if((input>200) and (input<=275)):
33   input= int (round(input*.97))
34  else:
35   input= int (round(input*1.02))
36
37  pwm.setPWMFreq(60)
38  x=0
39
40  if(y==0):
41   for x in range(0,input):
42    pwm.setPWM(0, 0, servoMin)
43    pwm.setPWM(0,0,0)
44
45  if(y==1):
46   x=0
47   for x in range(0, input):
48    pwm.setPWM(0,servoMax,0)
49    pwm.setPWM(0,0,0)
```

Figure 8: Motor Code

Once the first motor was understood then each motor was added one at a time checking that it would also move the correct distances.

Another construction was the ability to save the current location of the motors to a file. This was important to be able to have the motor know the location it was sent to in order to know how much farther it would be able to go before tangling the wires.

Areas that still need to be worked on is getting the more accurate way of each degree. Checking that the motors will stay between -360 and 360 degrees. Also having several numbers fixed for a certain purpose like go back to 0, 0, 0.

3.4 Stand:

We needed to create a stand that would not only stand on its own but be able to move in every direction via the motors. This was a difficult challenge, originally we had boxes that the servos would fit into to use in the arm. We soon figured out that boxes would not work because we would not be able to fit all the pieces together if we went that route. We decided to use a different approach by using brackets instead, below in Figure 9 is a prototype of our brackets before we designed them.

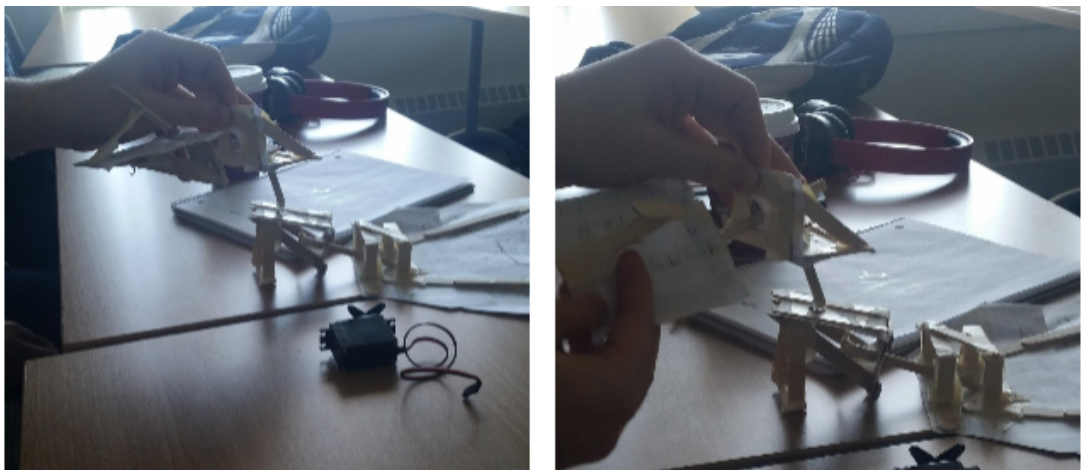


Figure 9: Prototype of Stand

Then we decided to model the parts in AutoCad Inventor since we liked the design. First we made the brackets to fit to the motors, these were the junctions between motors 1&2 and

2&3. Then we designed the mount for motor 1 which would eventually be mounted to the bottom plate which wasn't even made yet. The design for the mount for motor 1 was a simple box holder so that the motor could sit upright with a hole for the wire out the side. Motor 3 has a mount on the side which is a large flat square that our camera will be mounted to, this just ensures that the area that the camera attaches to has a wide amount of room. Finally the base was designed to be a large flat slab that has indents for all the other pieces to fit in properly. These pieces were successfully printed on the first try and installed without a problem. Below in Figure 10 is our designs for these pieces in Inventor.

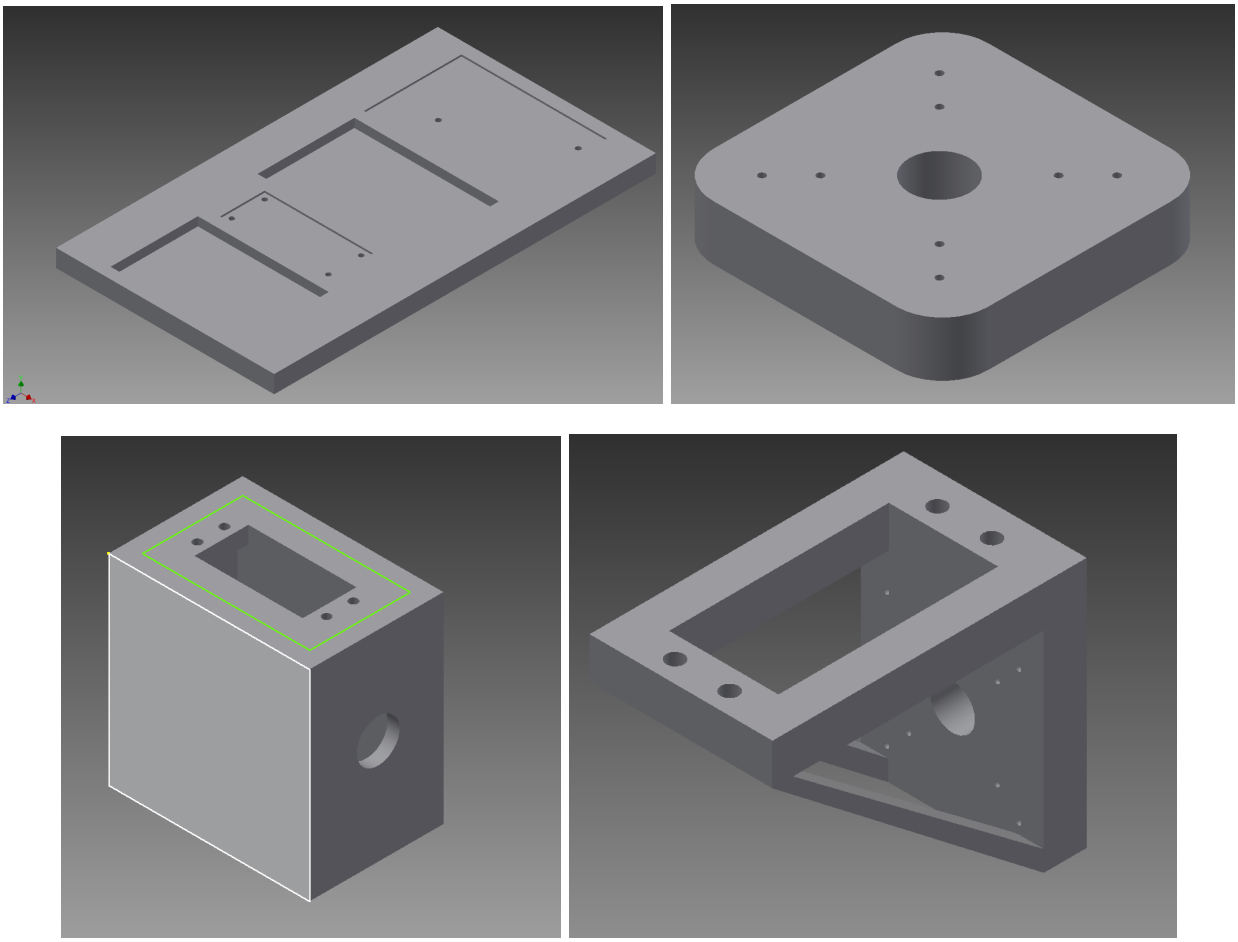


Figure 10: 3D printed parts from inventor

Each of these parts were just printed once except for the bottom right was printed twice on Figure 10. There could've been another design but it was realized that the motions were still based on the very similar design.

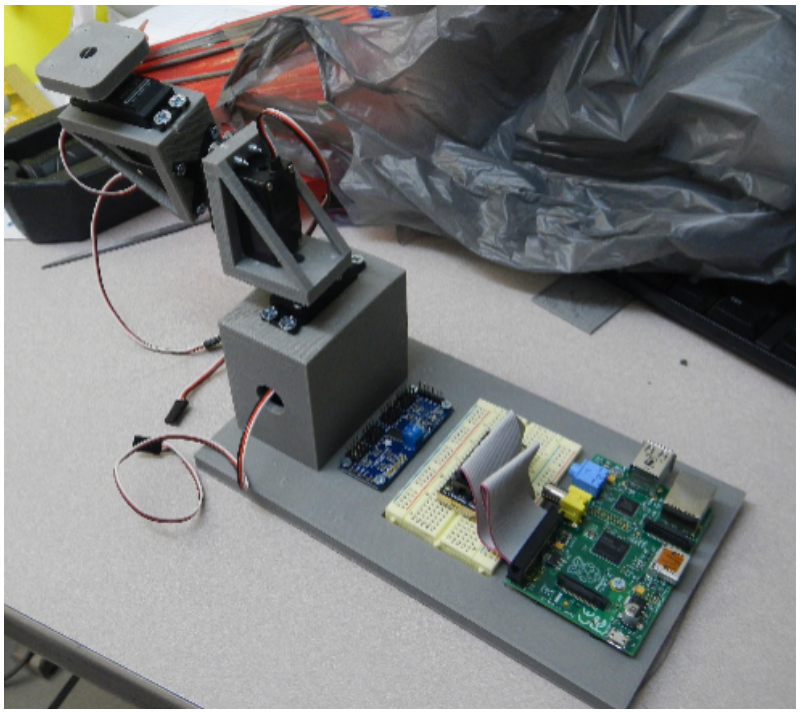


Figure 11: Complete Stand with Pi, motor and Servo attached

The assembling of the 3D modeling was constructed and it can be observed in Figure 11. This shows all the parts form Figure 10 printed and assembled. Once the parts were assembled they were tested to check and see that none of the parts would be affected by each other. Also before we decided to print a base stand we noticed that the rest of the system was a little to heavy and would easily tip over. This was when if we attached a base to it this would give enough weight to keep the entire system stable. The attaching of the base and the box that holds the first motor are secure because of the tight fit but could be glued together if there was more required movement of this system.

4. Aspects:

This area of the report helps to dictate what areas held difficulty and what areas were able to be done with relative ease. We had relative ease building the arm but figuring out what type of supports to design was tough as well as getting the servos to work correctly. Making the servos calculate distance in degrees was a hard challenge and we are still trying to perfect it. The website was a good problem to work out, the camera freed on the website was a challenge because we were having a java applet problem with the website, our frame rate is not as fast as it could be as well.

4.1 Contribution:

The team for this project consisted of 5 team members. Paul and Mitchell are both Computer Engineers, Charales and Jose are both Computer Science and Shamiul is a Computer Science minor. The teamwork was originally split up between hardware and software. Those who were computer science would focus more on the software and website aspect, with the computer engineers working on the motors and the design aspects of the project.

Software was involved creating the scripts of the website and getting the camera to display video and extract pictures. Charles worked getting the camera getting live feed setting up the initial website working on scripts to save files to the images. Jose worked on creating the website control interface and connecting its actions to the motors. Also, Jose focused on building the gallery webpage which displays all the pictures the user have taken with corresponding date and time. Shamiul worked on creating the images to help the user identify the directions each of the motors would move to help the user understand. He was the least experienced member of the group since he is a Computer Science minor, but his contributions can be seen in the Team Log. Mitchell and Paul both worked on the final user interface of the motors. Difficult areas were creating on the website the ability to display the present locations.

The hardware involved developing a software to move the motors to their desired locations and creating the physical stand that encased the entire system. Mitchell worked on the creating the 3D model and printing it out. Mitchell and Paul both worked on the motors movement, through the PWM and GPIO.

4.2 Data Log:

Team Log:
1) Date: 3/6/15 Location: OM 236 People in Attendance: Mitch, Paul, Jose, Shamiul Agenda/Goals: Create Web page outline Discuss project with easwaran Get parts list Items Discussed: Parts cost Who is assigned to what part of the project Web page functionality Action Items: Charles -- Web page design Jose -- Web page design Shamiul -- floater (help with 3d design?) Paul -- servo motors, camera control Mitch -- servo motors, camera control, 3d design
2) Date: 3/10/15 Location: OM 236 People in Attendance: Mitch, Paul, Charles Agenda/Goals: work on design of all functions Items Discussed: motor configuration, website development Action Items: Charles --continue Web Development Shamiul --learn bootstrap Mitch, Paul ----waiting on parts

3)

Date: 3/13/15

Location: OM 236

People in Attendance: Mitch, Paul, Charles

Agenda/Goals:

Website mobile device friendly

Review on what will be required

What parts people will work on

Google Drive

Break statis

4)

Date: 3/24/15

Location: OM 236

People in Attendance: Mitch, Paul, Charles , Jose, Shamiul

Agenda/Goals:

Splitting up website/camera

Hardware setup

Items Discussed:

Charles web priliminary design- http://cs.newpaltz.edu/~n02531357/ELSpring15/

Hardware Design

Camera Design

Action Items:

charles-live feed of camera and screen shots

Jose-PHP and sqlite for gallery

Shamiul-CSS and bootstrap user interface

Paul-Hardware control features (Sodeiring)

Mitch-Hardware (3D printed)

5)

Date: 3/31/15

Location: OM 236

People in Attendance: Mitch, Paul, Charles , Jose

Agenda/Goals:

Went over camera display and arm display
Camera goal to get live feed on website

Items Discussed:
What areas need a little more fixing
Arm needs more accurate turning

Action Items:
Write up due on Monday

6)
Date: 4/3/15
Location: OM 236
People in Attendance: Mitch, Paul, Charles , Shamiul

Agenda/Goals:
Went over camera display and arm display
Camera goal to live feed working
GUI is changing locations
Write up started good for first draft

Items Discussed:
Integrate the database and gallery
Get the camera to arm working possibly
Get arm to rotate more accurately
Getting a video of the website for 3D printing week

Action Items:
Write up can be sent through PDF
But Professor wants Write up on everyones Get Hubs

7)
Date: 4/14/15
Location: OM 236
People in Attendance: Paul, Charles , Shamiul

Agenda/Goals:
Have a presentation

Get code for motors on camera pi to be able to work for presentation
Get Apache server to work CGI for python to run from the web

Items Discussed:
Live recording doesn't sound like it is working
Presentation on Tues 4/21 "where we are at"

Action Items:
Shamiul help Joes with gallery
Getting code sent to camera pi
Preliminary presentation powerpoint

8)
Date: 4/17/15
Location: OM 236
People in Attendance: Paul, Charles , Mitchell, Jose, Shamiul

Agenda/Goals:
Have a presentation
Get code for motors on camera pi to be able to work for presentation
Get Apache server to work CGI for python to run from the web

Items Discussed:
Get picture script to work correctly
Presentation on Tues 4/21 "where we are at"

Action Items:
Shamiul help Joes with gallery
Getting code sent to camera pi
Preliminary presentation powerpoint

9)
Date: 4/24/15
Location: OM 236
People in Attendance: Paul, Charles , Mitchell, Jose, Shamiul

Agenda/Goals:
Get motor control to work on main pi
Items Discussed:

Javascript functions

Action Items:

Shamiul work on buttons

getting I2C to work correctly

10)

Date: 4/28/15

Location: OM 236

People in Attendance: Paul, Charles , Mitchell, Jose, Shamiul

Agenda/Goals:

get buttons moved around, get gallery working

Items Discussed:

webpage layout

gallery internals

Action Items:

jose work on buttons and gallery

get motors to work from web

12)

Date: 5/1/15

Location: OM 236

People in Attendance: Paul, Charles , Mitchell, Jose, Shamiul

Agenda/Goals:

finish up little things

Items Discussed:

add fixed to site

Action Items:

work on site and presentation and final report

There are many resources that were used which can be seen in the appendix of this report.

5. Future Goals:

The future goals for this project would be to get active locations of where each motor is display all the time. Also the motors still need to be more accurate if going to the correct degrees inputted as well as returning accurately returning to the origin. A future goal for the website would be to take record live video streams instead of just pictures, as well as to make the live stream fast by using a different type of server. Additionally, the motion jpeg server supports motion-tracking capability. A future goal would allow the camera head to track movement. Future applications for this program would be to be able to help the wires not tangle with each other. This camera boom could be attached to another motor device.

Another upgrade would be to add a spherical control to control all three motors at once. A diagram of this would be seen in Figure 12. To implement this the first part would be to get a scroll control in the website GUI instead of the button activation. This then could be taken into the use of a spherical button form.

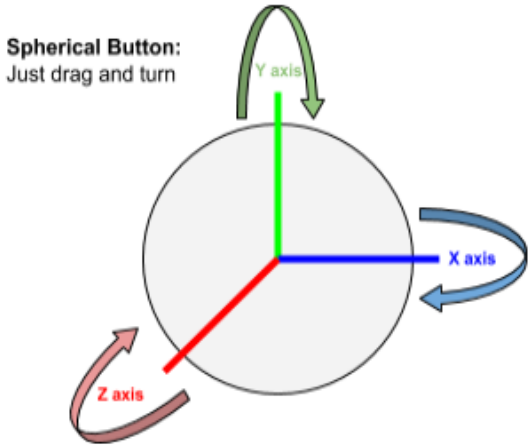


Figure 12: Future button for website

6. Conclusion

When projects are being processed it is often satisfying when they are coming smoothly along. There is often many places of difficult frustration but in the end when things work the result is satisfying. This doesn't mean that the creation for this project is over but being able to move along with results that show progression is always good.

There were two open houses that we went to for New Paltz, to display the boom camera. Here we described the progress that we made and the different parts of our project. We also demonstrated how the motors worked.

Several areas that can be improved upon are each motor having a more accurate method of creating the distance each motor will move. The motors could also move simultaneously instead of one after the other. The website can be improved by having the website refresh more often, the gallery could be organized by time and date, and the buttons could be moved to a more slider function. Another improved area would be to have the camera activate through motion sensing.

7. Appendix:

7.1 Resources:

Adafruit website for PWM:

https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/overview

Web building resources:

Motion Server

* http://lavrsen.dk/foswiki/bin/view/Motion/WebHome
* Used to serve MJPEG feed from Pi Camera

Bootstrap Framework

* www.getbootstrap.com
* html/CSS framework to implement website design
* https://github.com/blueimp/Bootstrap-Image-Gallery/blob/master/README.md

Pi Camera Python Documentation

* http://picamera.readthedocs.org/en/release-1.10/
* Documentation for the Pi Camera Module, built in functions, etc.

Apache WebServer Documentation

* http://httpd.apache.org
* WebServer on the Pi to host website. Documentation for web permissions when using PHP

PHP, JQuery/Ajax Documentation

* php.net/docs.php
* http://api.jquery.com/jQuery.ajax
* Used for various GUI-to-Server interaction with Camera and Motor

JavaScript Documentation

* w3schools.com
* DOM manipulation and feed controls

7.2 Personal Log:

Charles Palsa: Website and Camera Design

Date: March 10th, 2015

Actions:

* Built a preliminary web interface to outline the direction of the web app portion of the project.
* Utilizes bootstrap as a convenient framework.
* Organizes tasks in a visual manner

Date: March 24th, 2015

Actions:

* Assumed responsibility for camera-to-web functionality
* Assigned tasks to Jose and Shamiul
* Jose is to come up with a script that can store metadata and path data of pictures in a database.
* Shamiul is to familiarize with bootstrap and build the UI for camera and motor controls.

Date: March 24th, 2015

Actions:

* Researched methods of streaming raspberry pi video feed to web.
* upgraded pi firmware and installed official pi camera drivers
* A motion-jpeg server was the solution. Installed Motion Jpeg server/Motion tracking software.
* Used a 3rd party java applet to provide full browser support. Unfortunately the solution is not perfect.
* Began writing camera functionality scripts ie record/snapshot

Date: April 2nd, 2015

Actions:

* MJPEG is supported natively in 3/4 major browsers (Safari, FireFox, Chrome) and the java applet has been removed

* Cleaned up the interface, created buttons to activate camera control scripts.

* Video feed panel now responds correctly when video feed is offline, and displays an appropriate image.

Date: April 14th, 2015

Actions:

* Camera functionality is fully operational: Take Pic and Feed work.
* Web server permissions adjusted, PHP is used to run camera scripts
* Javascript GUI functionality implemented

Date: April 24th, 2015

Actions:

* Motor and website integration begun, web server permissions adjusted
* Feed pauses and refreshes successfully after picture.
* Website layout redesign

Date: May 1st, 2015

Actions:

* Attached camera to Camera Boom.
* Created start up scripts for camera and server initialization.

7.3 Code:

Web html Code:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
  <script src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.js"></script>
  <meta charset="utf-8">
  <link href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.css"
rel="stylesheet">
  <!-- Always force latest IE rendering engine (even in intranet) & Chrome Frame
       Remove this if you use the .htaccess -->
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Embedded Linux 2015</title>
  <meta name="description" content="">
  <meta name="author" content="Charles">
  <meta name="viewport" content="width=device-width; initial-scale=1.0">
  <!-- Replace favicon.ico & apple-touch-icon.png in the root of your domain and delete these
references -->
  <link rel="shortcut icon" href="/favicon.ico">
  <link rel="apple-touch-icon" href="/apple-touch-icon.png">
 <style media="screen" type="text/css">
 .gly-flip-horizontal{
  filter:progid:DXImageTransform.Microsoft.BasicImage(rotation=0,mirror=1);
  -webkit-transform: scale(-1,1);
  -moz-transform: scale(-1,1);
  -ms-transform: scale(-1,1);
  -o-transform:scale(-1,1);
  transform: scale(-1,1);
 }
 </style>
</head>
<body>
 <?php
      $f=fopen("Location_Motor","r");
      echo fgets($f);
      fclose($f);
      ?>
  <div class="container-fluid"> <!-- Site-inner container -->
   <div id="top-nav" class="row"> <!-- Site-Nav container-->
       <header>
   <nav class="navbar navbar-inverse">
       <div class="container-fluid">
       <!-- Brand and toggle get grouped for better mobile display -->
       <div class="navbar-header">
       <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
data-target="#bs-example-navbar-collapse-1">
       <span class="sr-only">Toggle navigation</span>
       <span class="icon-bar"></span>
       <span class="icon-bar"></span>
       <span class="icon-bar"></span>
       </button>
       <a class="navbar-brand" href="#">CameraBoom</a>
       </div>

       <!-- Collect the nav links, forms, and other content for toggling -->
       <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
       <ul class="nav navbar-nav">
```

```
        <li class="active"><a href="index.html">Live Feed <span class="glyphicon
glyphicon-facetime-video"></span><span class="sr-only">(current)</span></a></li>
        <li><a href="gallery/gallery.html">Gallery <span class="glyphicon
glyphicon-folder-open"></a></li>
        </ul>
        <form class="navbar-form navbar-left" role="search">
        <div class="form-group">
               <input type="text" class="form-control" placeholder="Search Gallery">
        </div>
        <button type="submit" class="btn btn-default">Submit</button>
        </form>
        </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
   </nav>
        </header>
 </div>

 <div id='feed-panel' class='row'>
  <div id="feed-status" class="panel panel-success  col-md-6" style="text-align: center;">
        <div class="panel-heading" style="text-align: center;">Video Feed</div>
        <div class="panel-body" style="text-align: center;">
        <img id="feed" onerror="displayOffline()" src="http:/137.140.114.52:8081">
        <br><div id='timer'></div><br>
        <button class="btn btn-default" onclick="takePic()"><span class="glyphicon
glyphicon-camera"></span></button>
        <button class="btn btn-default" onclick="refreshFeed()"><span class="glyphicon
glyphicon-refresh"></span></button>
        <br>
              </div>
  </div>
 <div id='control-panel' class='col-md-6'>
  <div id="motor_controls">
   <div class="panel panel-primary" style="text-align: center;">
        <div class="panel-heading" style="text-align: center;"><h4 class="panel-title">Motor
Controls</h4></div>
        <div class="panel-body">
    <div class="row">
        <h1>Move Camera</h1>
        <p>Use the fields below to move the camera or use the arrows</p>
        <br>
        <div class="row">
        <div class="col-sm-3">
      <img src="./xaxis.jpg" height="50" width="50">
      <br>
       X-axis
      <br>
      <input type="text" size="5" class="form-control" id="motor1" placeholder="0">
      <br>
       Motor 1
      <br>
      <div class="btn btn-sm btn-info" style="height:30px; width:70px;"
onclick="moveXUp()"><span class="glyphicon glyphicon-chevron-right"></span> Right</div>
      <br>
      <br>
      <div class="btn btn-sm btn-info" style="height:30px; width:70px;"
onclick="moveXDown()"><span class="glyphicon glyphicon-chevron-left"></span> Left</div>
      <br>
      <br>
       </div>
       <div class="col-sm-3">
      <img src="./yaxis.jpg" height="50" width="50">
```

```
      <br>
       Y-axis
      <br>
      <input type="text" size="5" class="form-control" id="motor2" placeholder="0">
      <br>
       Motor 2
      <br>
      <div class="btn btn-sm btn-success" style="height:30px; width:70px;"
onclick="moveYUp()"><span class="glyphicon glyphicon-chevron-up"></span>  Up</div>
                                          <br>
      <br>
      <div class="btn btn-sm btn-success" style="height:30px; width:70px;"
onclick="moveYDown()"><span class="glyphicon glyphicon-chevron-down"></span> Down</div>
      <br>
      <br>
       </div>
       <div class="col-sm-3">
      <img src="./zaxis.jpg" height="50" width="50">
      <br>
       Rotation
      <br>
      <input type="text" size="5" class="form-control" id="motor3" placeholder="0">
      <br>
       Motor 3
      <br>
      <div class="btn btn-sm btn-warning" style="height:30px; width:70px;"
onclick="moveZUp()"><span class="glyphicon glyphicon-share-alt"></span> CW</div>
                                    <br>
      <br>
      <div class="btn btn-sm btn-warning" style="height:30px; width:70px;"
onclick="moveZDown()"><i class="glyphicon glyphicon-share-alt gly-flip-horizontal"></i>
CCW</div>
                                    <br>
      <br>
       </div>
     <div class="col-sm-3">
      <br>
      <br>
      <div class="btn btn-sm btn-primary" style="height:30px; width:70px;"
onclick="moveCamera()">Move</div>
      <br>
      <br>
                    <div class="btn btn-sm btn-primary" style="height:30px; width:70px;"
onclick="resetMotors()">Reset</div>
     </div>
       </div>
    </div>
       </div>
       </div>
       </div>
       </div>
 </div>

 <div class='row footer'>
       <footer>
       <p>&copy; Copyright  by Charles</p>
       </footer>
 </div>
  </div>
<script type="text/javascript">
var count = 10;
```

```
var counter;
function moveXUp()
{
 $.ajax({ url: './control_camera.php',
        data: {direction: 'xup'},
        type: 'post'
    });
}
function moveXDown()
{
 $.ajax({ url: './control_camera.php',
        data: {direction: 'xdown'},
    type: 'post'
   });
}
function moveYDown()
{
   $.ajax({ url: './control_camera.php',
        data: {direction: 'ydown'},
    type: 'post'
   });
}
function moveYUp()
{
   $.ajax({ url: './control_camera.php',
        data: {direction: 'yup'},
    type: 'post'
   });
}
function moveZDown()
{
 $.ajax({ url: './control_camera.php',
        data: {direction: 'zdown'},
    type: 'post'
   });
}
function moveZUp()
{
 $.ajax({ url: './control_camera.php',
        data: {direction: 'zup'},
    type: 'post'
   });
}
function resetMotors()
{
 $.ajax({ url: './move_camera.php',
    data: {
     motor1: 0,
          motor2: 0,
          motor3: 0
    },
    type: 'post'
   });
}
function moveCamera()
{
 $.ajax({ url: './move_camera.php',
        data: {
         motor1: document.getElementById("motor1").value,
          motor2: document.getElementById("motor2").value,
          motor3: document.getElementById("motor3").value
```

```
        },
        type: 'post'
    });
}
function displayOffline(){ //If the feed img tag returns an error, display this image
 var feed_status = document.getElementById("feed-status");
 feed_status.className = "panel panel-danger col-md-6";
 var feed_source = document.getElementById("feed");
 feed_source.src = "http://cs.newpaltz.edu/~n02531357/ELSpring15/unplugged.png";
}
function displayWorking(){ //during image capture, display "working" to the user to show that
a picture is being taken
 var feed_status = document.getElementById("feed-status");
 feed_status.className = "panel panel-warning col-md-6";
 var feed_source = document.getElementById("feed");
 feed_source.src = "./loading_spinner.gif";
}
function refreshFeed(){ //Refreshes the feed by changing the source back to the server
location
 var feed_status = document.getElementById("feed-status");
 feed_status.className = "panel panel-success col-md-6";
 var feed_source = document.getElementById("feed");
 feed_source.src = "http://137.140.114.52:8081";
}
function startTimer(){ //Timer until auto refresh occurs
 count--;
 var timerText = document.getElementById("timer");
 timerText.innerHTML="Hold still! Refresh in " + count;
 if(count <=0){
  clearInterval(counter); //stop running startTimer method
  count=10;
  timerText.innerHTML="";
  refreshFeed();
  return;
 }
}
function takePic(){//display notification to user that a picture is being captured
 displayWorking();
 $.post("./take_pic.php");//runs PHP script that runs camera function
 counter = setInterval(startTimer, 1000); //Runs the startTimer method every 1000ms
 return false; //allows function to post without refreshing page
}

</script>
</body>
</html>
```

Motor Code:

```
#!/usr/bin/python
from Adafruit_PWM_Servo_Driver import PWM
import time
import sys
#import math
# ===========================================================================
# Example Code
# ===========================================================================
# Initialise the PWM device using the default address
pwm = PWM(0x40)
# Note if you'd like more debug output you can instead run:
#pwm = PWM(0x40, debug=True)
servoMin = 150  # Min pulse length out of 4096
servoMax = 150  # Max pulse length out of 4096
# Start is the program to move motors of a arm board in any axis (X, Y, Z)
# input --> Motor 1 (X-axis)
# input-->Motor 2 (Y-axis)
# input-->Motor 3 (Z-axis)
# get input--> Move X-axis-->Move Y-axis-->Move Z-axis
def Start(input, input1, input2):
#Motor 1 X-axis
 if(input<0): #take absolute value if less then 0
  y=1
  input=abs(input)
 else:
  y=0
 if(input<=90):#if the destination to move is less than 90degrees
  if(y==1):#going around in the clockwise direction may need to be higher
   input=int(round(input*.8))
  else:
   input= int (round(input*.71))
 if((input>90) and (input<180)):#destination to move < 180 degrees
  input= int (round(input*.82))
 if((input>=180) and (input<=200)):
  input= int (round(input*.9))
 if((input>200) and (input<=275)):
  input= int (round(input*.99))
 else:
  input= int (round(input*1.05))
 pwm.setPWMFreq(60)                        # Set frequency to 60 Hz
 x=0
 if(y==0):
  for x in range(0,input):
   pwm.setPWM(0, 0, servoMin)
   pwm.setPWM(0,0,0)
 if(y==1):
  x=0
  for x in range(0, input):
   pwm.setPWM(0,servoMax,0)
   pwm.setPWM(0,0,0)
#-------------------------------------------------------------------------------------
#Motor 2 Y-axis
 if(input1<0):
  y=1
  input1=abs(input1)
 else:
  y=0
 if(input1<=90):
  input1= int (round(input1*.70))
```

```
 if((input>90) and (input1<180)):
  input1= int (round(input1*.80))
 if((input1>=180) and (input1<=200)):
  input1= int (round(input1*.88))
 if((input1>200) and (input1<=275)):
  input1= int (round(input1*.97))
 else:
  input1= int (round(input1*1.02))
 pwm.setPWMFreq(60)                        # Set frequency to 60 Hz
 x=0
 if(y==0):
  for x in range(0,input1):
   pwm.setPWM(1, 0, servoMin)
   pwm.setPWM(1,0,0)
 if(y==1):
  x=0
  for x in range(0, input1):
   pwm.setPWM(1,servoMax,0)
   pwm.setPWM(1,0,0)
#-------------------------------------------------------------------------------------
#Motor 3 Z-axis
 if(input2<0):
  y=1
  input2=abs(input2)
 else:
  y=0
 if(input2<=90):
  input2= int (round(input2*.71))
 if((input2>90) and (input2<180)):
  input2= int (round(input2*.81))
 if((input2>=180) and (input2<=200)):
  input2= int (round(input2*.88))
 if((input2>200) and (input2<=275)):
  input2= int (round(input2*.97))
 else:
  input2= int (round(input2*1.02))
 pwm.setPWMFreq(60)                        # Set frequency to 60 Hz
 x=0
 if(y==0):
  for x in range(0,input2):
   pwm.setPWM(2, 0, servoMin)
   pwm.setPWM(2,0,0)
 if(y==1):
  x=0
  for x in range(0, input2):
   pwm.setPWM(2,servoMax,0)
   pwm.setPWM(2,0,0)

#-------------------------------------------------------------------------------------
#Main method that is initially called when running
if __name__ == '__main__':
#The location of the motors are allways saved in a Locaction_Motor file
#The file has only 3 numbers for each motors location
 loc= open('./Adafruit_PWM_Servo_Driver/Location_Motor','r')
#Grab each present motor's location
 loc1=loc.readline()   #Motor 1 X-axis
 loc2=loc.readline()   #Motor 2 Y-axis
 loc3=loc.readline()   #Motor 3 Z-axis

#input the desired inputs that are sent to the program (where you want to go)
 input=int (sys.argv[1])
```

```
 input1=int (sys.argv[2])
 input2=int (sys.argv[3])
 #copy the inputs values
 I1=input
 I2=input1
 I3=input2
 loc1 = int (float(loc1))
 loc2 = int (float(loc2))
 loc3 = int (float(loc3))

#If the user inputs 555 to any of the inputs that Reserved number will make the motor not move
#this is done by saving the current location the the input value of the motor
 if(input==555):
  input=loc1
  I1=loc1
 if(input1==555):
  input1=loc2
  I2=loc2
 if(input2==555):
  input2=loc3
  I3=loc3
 I1=input
 I2=input1
 I3=input2
#The location that you want to move needs to be subtracted to check if you are out of the
range
 input=input-loc1
 input1=input1-loc2
 input2=input2-loc3

 #Close the location of the file you opened Location_Motor
 loc.close()
#The checking of each motors location to be between 360 and -360 degrees
 if(((I1<=360) and (I2<=360) and (I3<=360)) and ((I1>=-360) and (I2>=-360) and (I3>=-360)) ):
  #If the location to move is okay then save the new location in the file Location_Motor
  loc = open('./Adafruit_PWM_Servo_Driver/Location_Motor','w')
  loc.write(str(I1)+'\n')
  loc.write(str(I2)+'\n')
  loc.write(str(I3)+'\n')
  loc.close()
  #Send only the distance that is needed to get to the desired location
  Start(input, input1, input2)
 else:
 # loc=open('Location_Motor', 'w')
 # loc.write(str(loc1)+'\n')
 # loc.write(str(loc2)+'\n')
 # loc.write(str(loc3)+'\n')
 # loc.close()
  print "Bad Input....Please Enter Again"
```