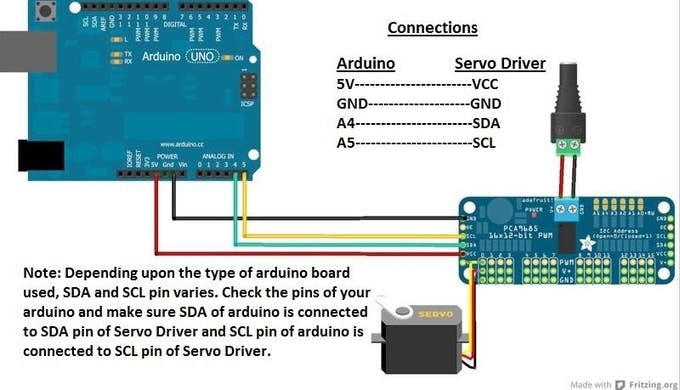
Current Parts List

* Arduino Uno
* Arduino Software
* Adafruit 16-Channel 12-bit PWM Servo Driver with I2C Interface (PCA9685)
* 5v 2.5A external Power Supply
* (4x) MG996R Servo
* DS3218mg Servo
* SainSmart S5 5-Axis Desktop DIY Robotic Arm
* 3-Pin Servo Extension Cables (4inches)
* Female-Male Wire Jumpers
* Raspberry Pi 3b+
* Monitor
* Keyboard/Mouse Combo

Arduino – PCA9685 – Servo Schematics



[Servo Motor Using Arduino & PCA9685 16 Chanel Module - Arduino Project Hub](https://create.arduino.cc/projecthub/jithinsanal1610/servo-motor-using-arduino-pca9685-16-chanel-module-d9666e)

Since I am not including any mechanical set up in this project, that will all be done on my own time. I will still include everything I use and include how I set things up. I will do my best to include all the works cited and include where I am getting my information.

NOTE: All mechanical parts of this project will be included in the journal, this will not be included in the hours counted towards the project. Only time researching and developing software is included in this project.

Day 1 –

The first thing I did was find schematics to wire the Arduino up with the PCA9685 servo driver. Arduinos website is extremely helpful when it comes to that stuff. I wired it all up, but I was having trouble getting it to work.

I downloaded the software package that Adafruit uses for Arduino to PCA9685 servo control.

“Adafruit PWM Servo Driver Library”

[Adafruit PWM Servo Driver Library - Arduino Libraries](https://www.arduinolibraries.info/libraries/adafruit-pwm-servo-driver-library#:~:text=Downloads%20%20%20%20Filename%20%20%20,%2015.21%20KiB%20%206%20more%20rows)

I followed the instructions given and read the help manuals, but the servos would not run. I found a I2C scanner program that Adafruit developed to help you find I2C components through the Arduino, I ran the software, and it was able to find my servo driver. I struggled and read a ton of help/forum pages. I finally found something that recommended I put the 0x40 that the I2C scanner found into the path when you first call the package, I did that, and it worked. The sucky part is that I tried 0x70 because 0x40 was supposed to be default, I should not have had to put that in. Once I did though, I could run the smaller test servo I had hooked up. It was a FS90R servo motor.

Day 2 –

Once I hooked up the actual servo motors the servos would hum when I tried to turn them, but they would not move. Which led me to believe my power supply was too low.

Originally, I was using a 3000mA power supply, but once I hooked up a 2.5A power supply it started working.

After reading through Adafruits sample code and going to their website to read through their documentation, I got a rather good understanding how to run the individual servos. I set up the code to run each servo to make sure it worked. After some struggle mechanically to re-align the servos so the mechanical limits were in a usable spot, I decided to find the limits using the software.

With each individual servo I found the limits by running it one way till it stopped, then move it back slower and slower until I got the servo values for the limits. I would write it down and go the other direction. Once all the limits were written down, I created global variables with the limits so I would never try and go past the physical limits. Once I did this, I was able to programmatically move the robot until I found a position where I thought was a good place to call home. I created global variables for the Home Position values, that way I could always return the robot to Home Position if need be.

Turn in Deliverable I

Day 3 –

Once I had my limits and my home positions saved, I was able to make a function that actually made the robot move to its home position servo by servo. I had to add delays in between each move so the robot would not get stuck.

I played with moving the robot to individual positions and back to home position. The problem I ran into was the robot moved too quickly and it was causing a lot of torque on the bottom joint (Servo 0). It slipped off the servo and I had to take it apart to push it back on and tighten the screw that became loose. (Maybe I need some Loctite)

After the slip I started looking into ways to slow the servos down, and there was nothing that I could do mechanically or programmatically to universally slow down the servos. The only thing that I found that worked was running the robot to a position incrementally using a FOR LOOP, it was clunky, but it would protect the servos mechanically. I created another function that accepted the servo and location to move and then made it so the servo would move incrementally to a position with delays in between each increment. This was not smooth, but it worked.

Day 4 –

I realized that with this new way of moving the servos I would need to keep track of the current positions; this complicates the program. This means everything I set up and every position I send the robot to, it has to know exactly where it is. The reason for this is if I want to move a position incrementally from 450 to 200 then I have to increment it down, and if the starting position is 200 and I want to move to 450 I have to increment up. When I set up the program originally, I was always moving from home position, so I did not think of it, it was also late. But we are not always going to be coming from the home position.

After quite a bit of set up I made it so the program will track its current position after every move. This will complicate things, but with the function I made it should not require any additional programming, no matter where I tell it to go. I imagine this will just run in a loop over and over because I do not have any sensors to control when it moves.

Now to move onto the Raspberry Pi servers.

Day 5 –

Now that the robot is set up, I started working on getting the Web Server from the Raspberry Pi working. I wasted almost a complete day on this and got nothing done. I followed instructions online and nothing worked. The instructions would tell me to do something and the Pi would not react how it was supposed to. For example, I would download software but when I went to the command prompt to adjust the files it told me to, the files did not exist. After some research I decided that I was going to start fresh. I was using an SD card from when I first started playing with the raspberry pi. Since the IP address was locked at an IP address that was from an old project, I decided I messed something up.

Day 6 –

After clearing the SD card and downloading the newest version of Rasbian, I started from scratch. I downloaded software called Apache2, originally, I was using software called ngnix but I struggled with it and could not get it working, plus it preferred PHP and I did not want to use PHP. This software is what allows me to remote into the raspberry pi through a web server, as long as I am on the local network. I could set up the port forwarding but I decided I did not want to do that.

After setup I could not get access to the html file to edit. The raspberry pi did not give me permission to the folders, and I could not change anything. After a couple hours of research, I finally found the command prompt commands to give me access to the folders. I found it on a comment from a raspberry pi forum.

[/var/www/html permissions - Raspberry Pi Forums](https://www.raspberrypi.org/forums/viewtopic.php?p=1015166)

///

Hi,

Its is simple.

On normal situation, http daemon run as some user and group, www-data on debian (raspbian).

Standard html files are stored on /var/www/, owned by root:root, with permissive permission, all can read, but only root can write.

To ordinary user write to /var/www need to takeover it. Supposed the use is pi.

sudo chown -R pi:www-data /var/www

Also, need to set user and group permission:

sudo chmod u+rxw,g+rx-w,o-rwx /var/www

Now, /var/www can be read,write and chdir by user pi, group www-data can chdir and read. Other not have access.

sudo chmod g+s /var/www

Any new file created on /var/www belong to group www-data.

If have files on /var/www, change user and group, and allow to group www-data read.

For file chmod u+rw,g+r-xw,o-rwx

For directory chmod u+rwx,g+rx-w,o-rxw

Now, user pi can manipulate files on /var/www and httpd can read, but not write

///

Once I did those commands, I could finally change the HTML file.

Using software on the Pi called “Geany”, I was able to create an html file that mimicked the name of the html file in the folder that the Apache2 software pointed to. I read the setup code for apache2 and it looked in a specific folder for an html file called “index.html”. I could adjust their code to make it work for whatever filename I wanted, but I figured there is no point in changing code that works. I created an index.html file that pointed to a CSS and JavaScript file I created. I created a simple “Hello World” page just to test it out. I was able to remote into the web server with my laptop and my phone through the WIFI, just by typing the Pi’s IP address in my URL.



Now to work on serial communication from the Raspberry Pi to the Arduino.

Turn in Deliverable II

Day 7 –

[Controlling An Arduino From A Raspberry Pi - Woolsey Workshop](https://www.woolseyworkshop.com/2020/02/05/controlling-an-arduino-from-a-raspberry-pi/)

Working from the link above I started to try and test my code but realized I don’t want to unplug and plug in the Arduino anytime I want to test and download new code. So, I decided to download the Arduino IDE on the Raspberry Pi so I could program it directly from there and would not have to unplug and plug in the Arduino so much. I followed the instructions from the link below. It took a minute and kind of slowed down my Pi, but its better that then possibly damage my Arduino during the software development.

[Install Arduino IDE on Raspberry Pi - Raspberry Pi Spy (raspberrypi-spy.co.uk)](https://www.raspberrypi-spy.co.uk/2020/12/install-arduino-ide-on-raspberry-pi/)

After copying and pasting the code the guy provided on the top link, it was not working. I had trouble with the serial connecting at first. I kept getting an error that would say ‘Wi-Fi is currently blocked by rkfill’. The code would still run but that would always pop up. After doing some googling, I found that it was because my Wi-Fi on my Raspberry Pi was disabled. I guess it has to be enabled if I do not want that to pop up.

I was also having problems sending and receiving the commands that the guy had set up. I had replaced the LED on and off with just a print. I realized that it was causing problems because it would overwrite the data. I ended up writing some simple back and forth serial commands to get an understanding of how the commands were supposed to work. I did not know that over serial communication the data will be read from both sides. This makes me realize that what I have to set up is conditions that look for values, that way the code will get the correct values. I am not sure how I am going to make sure that each command is seen.

Day 8 –

After working on this for a few hours I learned a lot. I learned that you have to be extremely careful when you write to the serial tag. At first, I was having the Raspberry Pi constantly watch the serial, but then watching the serial from the Arduino it never had a chance to write. After tweaking with the code for awhile I was finally able to come up with a way to simulate an ON/OFF that cycled every second. The handshaking is overly complicated and hard to troubleshoot if you do it wrong. I am going to have to be really careful when I do this for the real thing.

Code written to test this was the usb\_Pi.py and the usb\_A.ino

Turn in Deliverable III

Day 9 –

Next thing I had to do was implement what I did in the last section, into the testRobot program. I was able to change a few things and change the ON/OFF to a positional movement for the robot to move too. It went pretty smoothly until I realized all my home positions were off.

After I looked through everything, I realized that when I moved everything over from the computer to the Pi, I had to redownload the Adafruit library, and I downloaded a newer version. This somehow affected my limits. It divided the limits up into a larger scale, so I had to increase all of my larger value limits. This also meant I needed to adjust all my home positions. I went through, found all my new limits, and updated the values, then I went and found my new Home Positions.

I also created a Position\_1 function that when ran it runs the robot to a saved position 1, I basically just made the position where all the servos move positive. It is a random position. I also created a controlledHome function that allows the robot to move Home at a slower speed. This is important because I do not want to move at full speed every time, I move home. The original Home function should only be running when the robot does not know its current position and/or on start up.

Day 10 –

Since this section did not take as long as I thought it would, I thought I would try to move onto the next section. Work on getting a button on the web server that can be used to call the python script. This will then allow me to move the robot from the server.

After a full day and a half of constant research and testing. I have not found anything that will let me run a python script from an html file or JavaScript file. I have tried a bunch of things like ajax, and something called flask for python. The reason I do not want to make python the server is I will have to open the terminal every time I want the server to work. I want the server to start and run whenever the Pi is on.

I also tried to set up a php file that would call the Python script from the system() command. It worked when I would run the php file by itself, but when I ran the file from the server it would not work, it would only try to download the php file or print the script on the browser.

I am where I wanted to be for Deliverable IV, I thought I could get ahead but I could not. I may have to rethink the python portion and just run the serial communication through PHP. Which means I will have to learn PHP…

Turn in Deliverable IV.

Day 11 –

Did research the entire day. Messed with the idea of using PHP, tested code and failed for hours. Googling stuff over and over, and never finding anything possible. I also read into using Node.js to do all the serial communication, but again, after hours of testing and research I found nothing that would work. I am not sure what to do at this point...

Day 12 –

Again, I googled and tested code I found online, and nothing.

Day 13 –

While sitting at my desk at work, I started finally looking into using a python server, even if I had to start the server up anytime, I started up the raspberry Pi.

As I was reading, I found a YouTube video that showed a way to attach a python flask server to a apache2 server so that the python server would run without having to start it up. I already got apache2 on the pi and have the server working with all the permissions I needed. I sent an email to myself with the link so I could follow it when I got home.

<https://www.youtube.com/watch?v=mQ5qzhAt0tw>

Once I got home, I started following the youtubers instructions, but quickly realized that some of his stuff and GitHub links did not exist. I tried to find the files and found a couple to try, specifically the activate\_this.py program. I tried a few of them and none of them worked. Something also went wrong while doing it because it corrupted my apache2 files and it would not restart or start back up anymore. I uninstalled and reinstalled apache2 and looked for another way.

Finally, I found a similar set of instructions, and it had different links.

[How to run a Flask app behind Apache + 4th Pi Giveaway - Easy Programming](https://www.easyprogramming.net/raspberrypi/pi_flask_apache.php)

It worked! I finally have a working server that can connect and run a python program. Now I just have to see if there is a way to implement the usb\_Pi.py program I wrote that would run the robot. If I can do this then I can make a way to run the robot from a server.

Also, something annoying, I have to restart apache2 anytime I want the code from piapp.py to change the server.

Run when changes are made to piapp.py …

“sudo service apache2 restart”

The way the new server is set up is actually better. It references the Python file as a separate extension. So, when I put 192.168.0.47 in the URL on my computer it will pull up the index.html file I use for apache2. If I type 192.168.0.47/piapp into the URL then it will pull up the python code. This is good because now I can keep them separate, which allows me to make the index.html my home server and the piapp the setup for the robot!

I also need to clean up the folder for my deliverables, it is a mess. Cleaned up for the most part.

With this Python Flask server, I still cannot find a way to get a button to work. I have spent a couple hours getting it all set up to call an external html file, but when I set a button up it always gives me an error. I am not sure how to have a button just be a button, not a link to another page. I am getting extremely frustrated.

I did make some progress though; I have gotten it to where the website will call the command to the robot. The only problem is it is not done by a button, and I can only run the code once. So when I refresh the page it does it again. So, with my code …

@app.route(‘/robot/auto’)

Def auto():

Val = ser.readline()

Val = val.decode()

Val = val.strip()

If val == “Home”:

Ser.write(b’1’)

Return “<h1>Position 1</h1>”

Elif val == “Pos1”:

Ser.write(b’0’)

Return “<h1>Home</h1>”

Time.sleep(1)

Every time I refresh the page ‘192.168.0.47/piapp/robot/auto’ the robot will change positions, switching between pos1 and home. This is all based on my Arduino code TestRobot.ino.

Calling it a day …

Day 14 –

1:00am … I woke up because I could not sleep, I had a thought. The HTML file would say page not found anytime I clicked the button, because my url was pointing towards /robot/gui, because that is what matched my python code. But the link was actually /piapp/robot/gui, the html wasn’t adding the link onto the folder piapp like the python link was, so it could not find it. Once I added the piapp to the url it worked perfectly.

<form action=”/**piapp/robot/gui**” method=”POST”>

<input type=”submit” name=”submit” value=”Home”>

<input type=”submit” name=”submit” value=”Pos1”>

</form>

I ended up changine “gui” to “pos”, made more sense.

I also found out that the external CSS file and JavaScript files do not show up when you are on an external device. When I open the html file on the raspberry pi it works fine, but when I do it from my phone or from my computer none of the CSS decorators or JavaScript stuff shows up. It works fine though if I put the scripts or style/CSS code embedded into the html file. I have no use for JavaScript currently anyway. I am sure there is a way to get it to work.

This took me WAY longer then expected. Now all the pieces are hooked together though, I just have to clean up the code, and add any extra code to make it work the way I wanted to.



Turn in Deliverable V !!!

Day 15 –

Today I decided to clean up the server web browsers. Right now, they are just white pages with a couple buttons. Since there are technically only 3 web browsers possible to go to then I will just set up 3 for now, but I want to be able to expand if needed.

Since I am not the best web developer, I am going to pull from my final project that I did in JavaScript. I want to have a strip that has all the pages that you can go to. I want to set it up so you can switch between pages seamlessly, but I also want to be able to access all pages from each page. Just like I did with my previous final project.

Since I have to use embedded CSS and JavaScript in the python server html files, I will just set up one CSS file and then copy it into the other two html files that link to the python server.

I set up the main server from apache2 to where it looks nice, I had trouble organizing a few things on the menu, but I was able to figure it out. I also added in some images of the raspberry pi symbol to make sure it is known that it comes from a raspberry pi.

I copied the CSS in the other two files, and it does not look right. I ended up copying all the CSS code into the html file, but that did not work. Still looks off, but I don’t want to switch everything back, so I am not using an external CSS file anymore.

I figured it out. I had double copied the buttons and so everything was shifted for some reason. It also looks terrible on a phone, so I am trying to adjust the sizes and alignment to look good on a phone.

I finally just decided the buttons are going to just move downward for each button. So, they are all stacked on top of each other. Only two right now but it is set up so you can have as many as you want.

Day16 –

Now that all the connections are done, and the websites are set up I need to add more positions on the webpage, the python code, and the Arduino code.

It was fairly easy. I just copied and pasted the code, which is great because it’ll be easier to add more at any time. Now I have to adjust the positions to meaningful positions, not just random ones.

I decided to pick up a tissue paper and have it drop it in my hand. Easy but a task that can show how repeatable it can be.

It took awhile to get all the positions right, but they are all set up. Now I need to add a button and function that will have it go through the positions automatically.

It was amazingly simple. I made a function that just calls the positions in the order needed and BAM, done. Honestly, I am happy with the way I set it up, it was a lot of work, but the process can be changed, and positions can be added so easy.

I put the auto button at the top and separated it from the positional buttons.

It runs smoothly and runs from my phone. I took a video of it and have attached it to the folder.

This project was awesome! It was incredibly fun, and I learned a lot. I can honestly say this certificate was worth every penny. I did not use JavaScript, but its all set up to be easily added. I figured I will leave the main page and the main python pages blank until I can figure out what I want to do with them. I will also attach screenshots.

Thanks!

Turn in the last Deliverable.