Room Mapping Robot

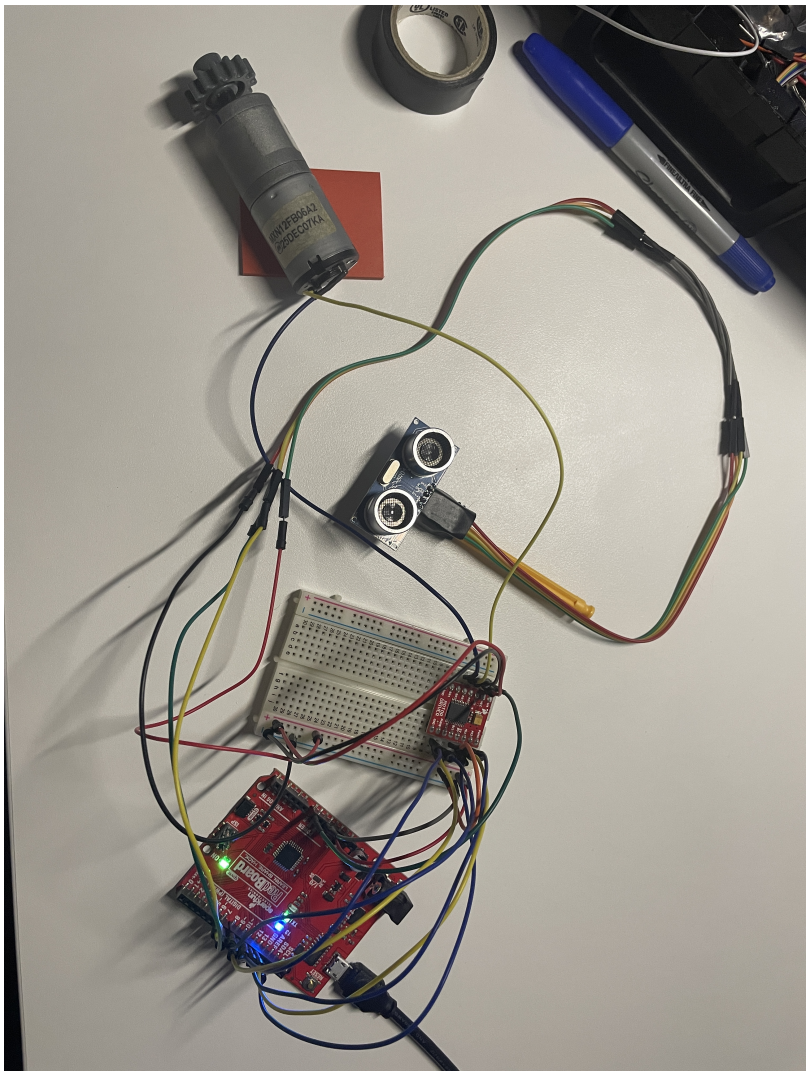
Initial Proposal

My initial proposal for this project can be found here https://landing.athabascau.ca/discussion/view/17494425/final-project-proposal-floor-area-mapping.

"The concept is to have a simple robot which acts as a room scanner. When placed on the ground, the robot spins the ultrasonic sensor 360 degrees in place, measuring the distance in front of it as it goes (may need multiple scans, or use 2 ultrasonic sensors for averaged readings, or lidar in place of ultrasound). This information would allow for an approximation of the area of floor space in the room, which would be displayed on the arduino LCD. As a possible extension on top of the area measurement, I may link the arduino to a raspberry pi in order to take that distance data and create a graphic or map of the borders of the room, and objects within the room (from the scanning perspective of the robot) which could be viewed over wifi. The robot required to complete this project may be relatively simple with the main hardware being a servo, ultrasonic sensor, LCD for viewing the calculated area, and a couple accessory LEDs for indicating status (as well as integrating the raspberry pi), though I believe the complexity of the software may make up for the static and somewhat basic hardware."

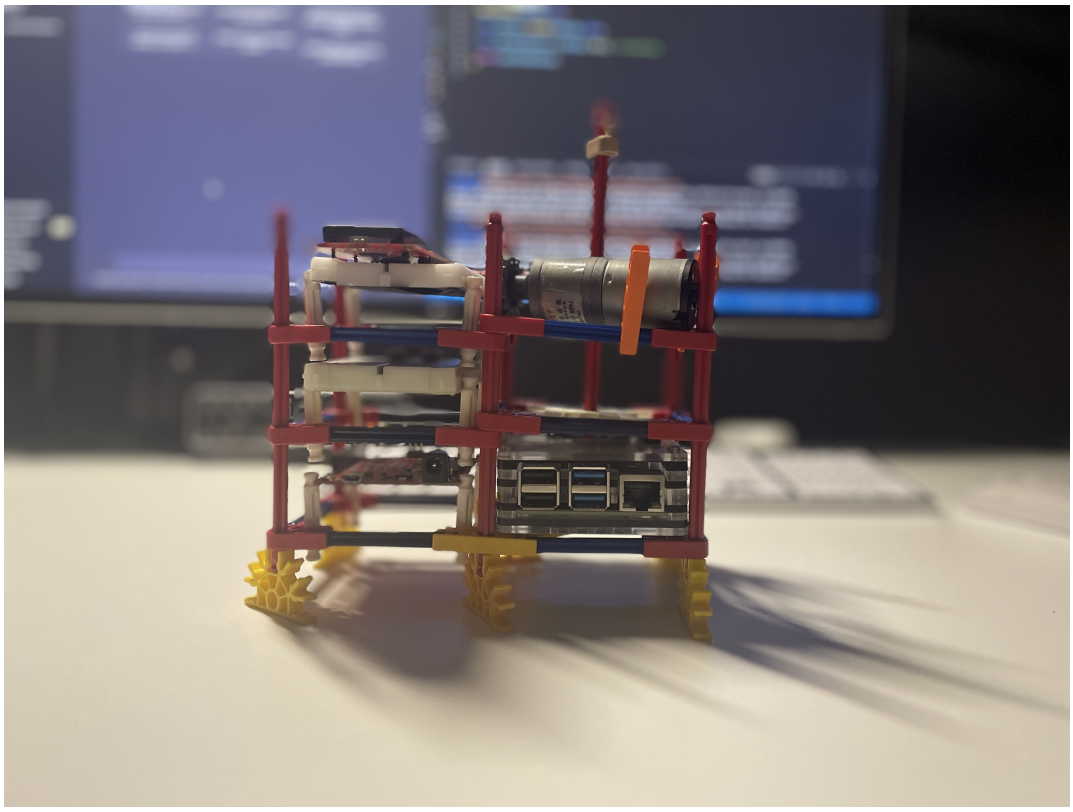
Design Decisions

While the plan originally outlined in the proposal post did change a bit, the major features remain consistent. To review some key elements, I ended up removing the LCD screen, as I didnt have enough Arduino pins to properly connect it, as well as the fact that I decided to control the robot via Raspberry Pi over wifi, so the on-robot LCD screen seemed somewhat redundant. For other hardware, I used a motor from a previous personal project, which has an internal gearbox making it extremely low rpm, in comparison to the motors which came with the course kit. The other components used include the ultrasonic sensor, motor controller, LEDs, resistors, and the breadboard which came with the SparkFun kit, as well as many jumper cables.

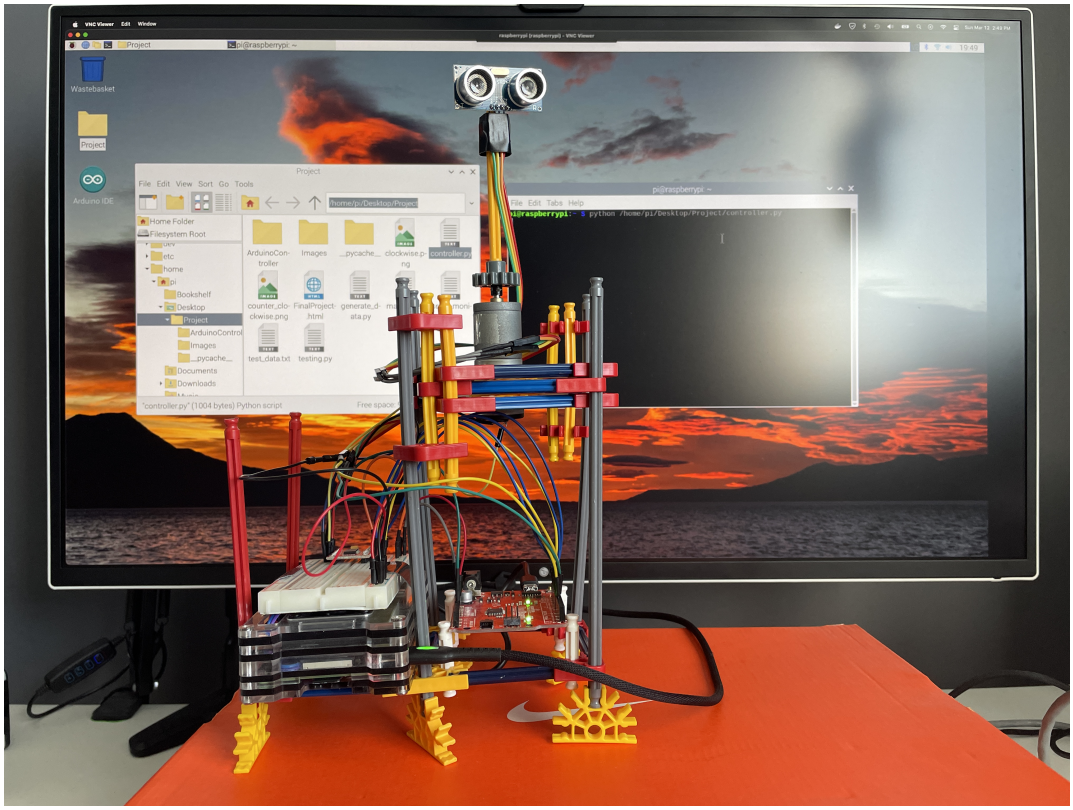


Testing hardware connections between motor, motor controller, breadboard, and arduino - before addition of LED indicators

For overall construction of the robot, I used old K'nex which I had from when I was younger, for the main frame of the project. While some components didnt fit nicely, like the motor, I was able to create friction-based holders for them. This approach for prototyping was quite useful, as the ability to connect, and rebuild elements without having to start from scratch made the process much smoother overall.



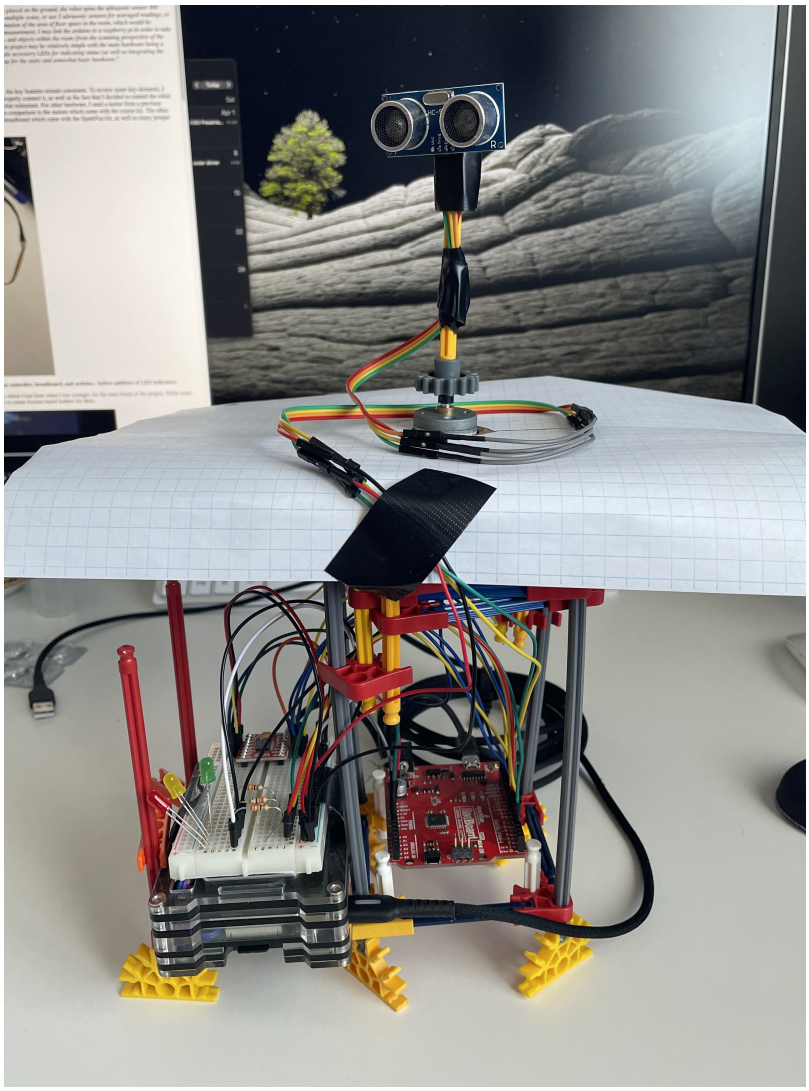
Initial design for the robot structure, featuring raspberry pi, breadboard, and arduino holders, prior to the decision to remove the LCD, and prior to mounting the motor and ultrasonic sensor



Structure of the robot as it neared completion, prior to addition of LED indicators - with raspberry pi VNC connection in the background

The aesthetics of the project sort of fell apart as I reached the final stages, as I needed a smooth surface for the wires linking the ultrasonic sensor to the arduino. The friction on the wires dragging was a major challenge, as it

made the turn degrees vary, which made results garbled. I found using a piece of paper for the wires to rest on helped them slide smoothly as the motor turned.



Final structure of the robot, featuring status LEDs (bottom left), and friction-reducing paper

After key design decisions were made, the robot description can be summarized by the paragraph below:
In order to use the Room Mapping Robot, begin by placing the robot in the room you wish to scan, then access the raspberry pi over ssh or vnc, and trigger the `controller.py` file to start the scan. The LED indicator should switch from Green to Yellow, indicating command received, and the console should also display the current status. After a few seconds, the LED will change from Yellow to Red, and the motor will begin turning, scanning the room. The Robot will complete a counter-clockwise turn, and then the python script will trigger the clockwise turn, which follows the same LED sequence, and motor instructions, except now turning the opposite direction. Once both turns are complete, the python script processes the data received from the arduino, charts the approximate room shape, and calculates the approximate floor area of the room.

General Programming Considerations

While the control of the arduino was relatively simple to implement, the processing of distance data received by the ultrasonic sensor would likely be quite challenging to complete in arduino code. This led me to consider using a Raspberry Pi as a processing device, to allow for data organization in python.

Working with reading the serial output of the arduino in python, it was tough to get the timing right, as the arduino code had to be started prior to connecting the python serial monitor. A number of different websites had

instructions about connecting an Arduino to a Raspberry Pi, so the implementation wasnt too challenging after a bit of reasearch.

For drawing the mapped distances in python, I used the Turtle package, and frequently referenced the documentation found here: https://docs.python.org/3/library/turtle.html.

For data normalization, and attempting to remove outlier readings from the ultrasonic sensor, I used the python numpy package, which has documentation here: https://numpy.org. After reading a number of resources on outlier removal, I decided to go with an approach where outliers are determined by comparing the difference between adjacent points with a threshold based on the standard deviation of the adjacent values. When an outlier is detected, it's replaced by the average of its neighbours.

Program Implementation

The plan for this robot was to implement a reactive control system, where signals are sent via the connected Raspberry Pi to the Arduino, where those signals are parsed and turned into actuator & sensor performance.

The first thing I implemented was the beginning of the data processing in python. I created a simple script which generated fake data (`generate_data.py`), and used that as a starting place for working with the turtle package to draw the distances and surrounding walls.

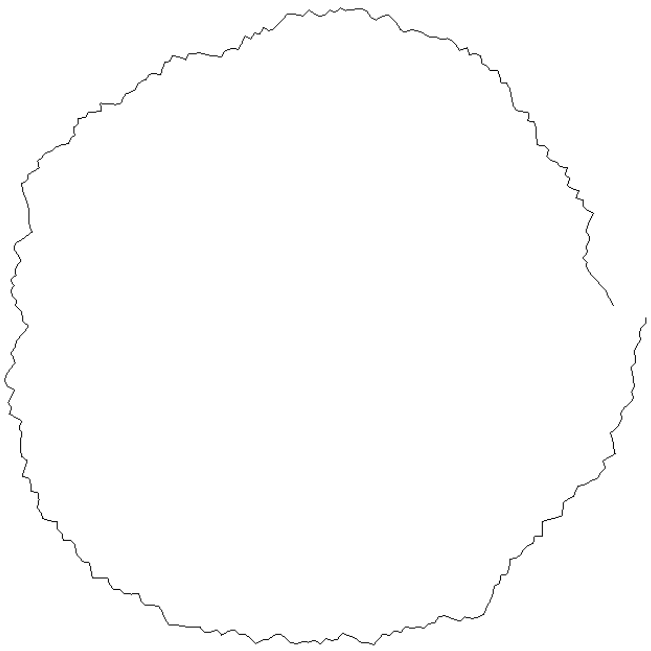


Image generated using fake data, as test for mapping code

Once I had the initial image generation complete, I moved onto the Arduino code, beginning with a simple signal-driven control loop. The serial input is checked every loop, and if theres a "1", "2", or "3" received, each of those values have actions attached to them. "1" indicates a 'Ready' state, which turns the green 'Idle' LED off, and the yellow 'Ready' LED on. "2" triggers a clockwise turn, which rotates the ultrasonic sensor 360 degrees, after swapping the LEDs to Red being active, and both green and yellow inactive. "3" is the same as "2", except for counterclockwise turn. The Arduino code is relatively straightforward, except for the trial and error elements which will be discussed later under **Testing**.

Now that the general format of the Arduino control loop has been set up, I spent some time figuring out sending values over serial, which must be encoded (but luckily theres a built-in `encode` function in python which makes this easy). Once the Raspberry Pi and Arduino were able to communicate with one another, I moved onto testing the image generation with the real ultrasound readings. The first test was just using a single direction rotation.

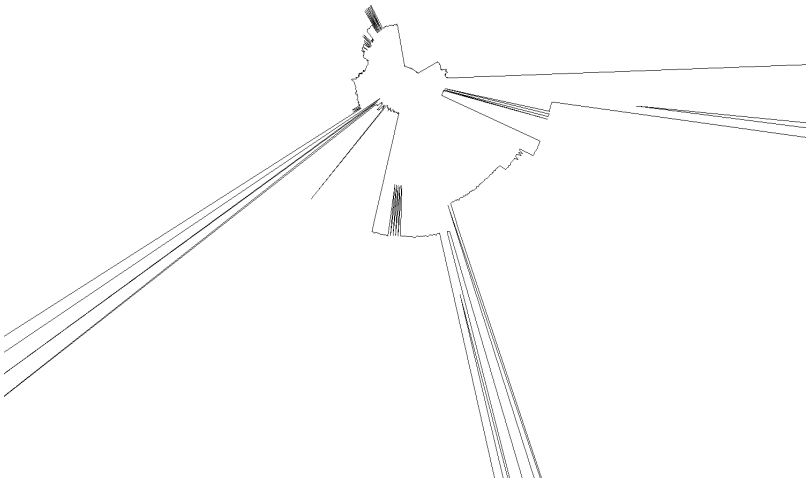


Image produced from a single clockwise turn

There is obviously a significant amount of interference and invalid data being reported by the ultrasound sensor, which was expected, so I moved onto my initial plan for increasing data consistency, which was to average the readings produced by the clockwise turn, with the reversed readings from the counter-clockwise turn. I also added a limiter to the data readings, so distances which are obviously incorrect (e.g. > ~4 meters when measuring inside my room), would be discarded in place of that designated maximum distance.

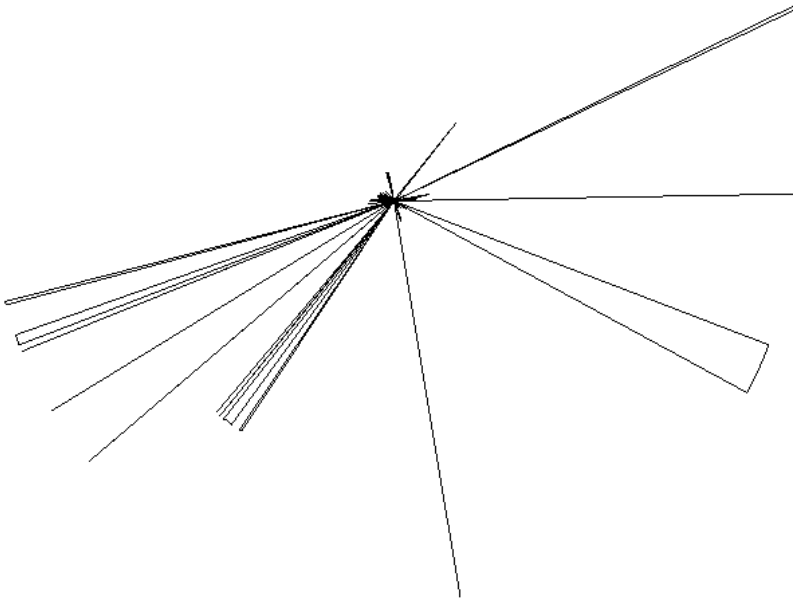
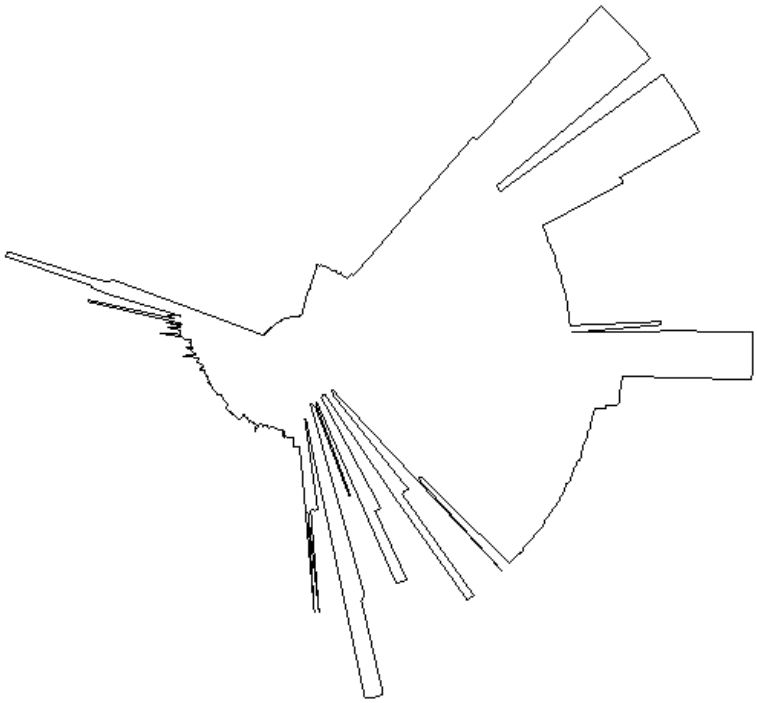


Image produced from averaging clockwise & counterclockwise rotation data

The result produced by averaging the data was a surprise to me, as I thought that would have a significant (positive) impact on the precision of the image, but clearly it made the data even less reliable. This led me to research more into outlier elimination, as that was the real goal behind my approach, and it turns out there's many different ways to handle outliers in a data set!

As the data processing and normalization was near completion, I added the approximate area calculation code, which sums the area of each slice of the scan circle, using the lengths of 2 sides, and the known angle of 0.5 degrees between measurements. This estimation assumes a flat surface perpendicular to the ultrasonic sensor's distance readings. The estimation is also only as accurate as the data it receives, so the inconsistent ultrasound readings made the calculated area vary to a fairly wide margin.

After all the code was complete and data was properly aligned and normalized, the wall map generation created images such as the one below, with no clear start/end of the map which demonstrates the consistency of the data. Though while the start/end align properly, theres obviously some interference and odd data with the way the image is presented.



Generated image using final code process, normalized data & bidirectional scan

Testing

Overall I tested the final robot in a few different environments, and it performed fairly consistently between them, so the final testing of the robot was mainly just made up of an overview of the products produced (e.g. scan image, and approximate area calculation).

Software Testing

As mentioned in **Program Implementation** I did some initial testing of the `turtle` python package by generating some mock data. This was a relatively simple script, but was extremely valuable when it came to learning about producing images in python.

Hardware Testing

My intial proposal plan was to use a servo to orient the ultrasonic sensor, though that was quickly replaced with a low-rpm motor, when I remembered that the servo in the SparkFun kit only turned 180 degrees, and I was looking for a complete 360.

Timing the 360 degree turn of the motor with 720 distance readings at equal intervals took plenty of trial and error, as a motor obviously lacks the position/state knowledge of a servo. Precise alignment for a 360 degree turn in both directions took many adjustments of motor speed, and control loop frequencies. This motor control was also impacted by the friction of the wires attached to the ultrasonic sensor. I ended up adding 1 point of overlap (approximately 0.5 degrees of rotation), with 721 points being measured during each rotation of the ultrasonic sensor, in order for the complete scan to be drawn properly.

It took lots of trial and error working with the motor and ultrasonic sensor, as the wires made performing a complete turn quite challenging. The motor torque is relatively low, though the main element that was negatively affected by the friction of the wires, was the connection between the post holding the ultrasonic sensor, and the

motor, as it slipped within itself when it encountered too much resistance. I wanted to avoid glueing it together though, as that would make future adjustments challenging, and I was also cautious about getting liquid glue near the motor spindle.

Process Documentation

As shown above, there are many images of the robot throughout its construction available on my Github (linked below with source code), as well as a number of images produced by the program throughout the data processing stages.

A video displaying the robot functionality can be found here: https://youtu.be/6DuLbHUCw0E

Outcome

Overall, given the hardware and time constraints of this project, I'm satisfied that it met the original design goals as both the key elements, approximating room area, and creating a visual map of the walls, were completed. While the results produced by the robot vary somewhat, and the readings arent incredibly precise, given the relatively basic and inexpensive hardware, I believe it serves as a solid proof of concept. This project was a great learning experience, and hands-on opportunity to deal with integrating hardware and software, expecially considering the uncertainty and unreliable nature of the physical components.

In the inital proposal, I noted that the ultrasonic sensor is likely not the best choice for this use-case, so had there been a budget for the project, using lidar or some combination of sensors likely wouldve produced more accurate results than the current implementation which uses (the extremenly inexpensive SparkFun) ultrasound.

It's also worth considering using a 360 degree servo, or a stacked system with 2, 180 degree servos, since those appear to be more common. The constant motion and imprecision of using a motor likely had a negative impact on the ultrasonic readings, though it's difficult to know how much of an impact it would have without trying it!

Source Code

Source code for the project, including both Arduino code and Python code which handles data processing and representation can be found in this GitHub repository: https://github.com/jaredwebber/comp444-final-project

Collaboration Regarding Project

The only reply I received on my project proposal discussion post was from the Tutor & Course Coordinator, which can be found here https://landing.athabascau.ca/discussion/view/17494425/final-project-proposal-floor-area-mapping

I also made a blog post with the content of this report, which can be found here https://landing.athabascau.ca/blog/view/17900048/final-project-room-mapping-robot