Oregon State University

Oregon State University Capstone
Tentacle Arm Project Document

Fall 2021 - Spring 2022

Ben Chan, Cale Hallamasek, Triet Nguyen, Jordan Porter

Table of Contents

I. Overview **7**
1.1 - Executive Summary 7
1.2 - Team Communication Protocols and Standards 7
1.3 - Gap Analysis 9
1.4 - Timeline 10
1.5 - References and File Links 12
1.5.1 - References 12
1.5.2 - File Links 12
1.6 - Revision Table 12

II. Requirements Impacts and Risks **12**
2.1 - Requirements 12
2.1.1 - The system will extend 1.0 meters in any direction 13
2.1.2 - The system will support a mass of 0.5 kilograms 13
2.1.3 - The system will be accurate 13
2.1.4 - The system will be reliable 13
2.1.5 - The system will be self-powered 13
2.1.6 - The system will be portable and easy to use 13
2.1.7 - The system will output a user-friendly 3D visual. 13
2.1.8 - The system will support input in spherical, cylindrical, and cartesian format 14
2.2 - Design Impact Statement 14
2.2.1 - Introduction 14
2.2.2 - Public Health, Safety, and Welfare Impacts 14
2.2.3 - Cultural and Social Impacts 15
2.2.4 - Environmental Impacts 16
2.2.5 - Economic Factors 17
2.2.6 - Accessibility Impacts 18
2.2.7 - Conclusion 18
2.3 - Risks 19
2.4 - References and File Links 21
2.5.1 - References 21

Citations **21**
2.5.2 - File Links 22
2.5 - Revision Table 22

III. Top-Level Architecture **24**
3.1 - Block Diagrams 24
3.2 - Block Descriptions 25
3.2.1 - Software Blocks 25

3.2.2 - Microcontroller Blocks 26
3.2.3 - Mechanical Blocks 27
3.2.4 - Power Supply Blocks 27
3.3 - Interface Definitions 27
3.4 - References and File Links 28
3.4.1 - References 28
3.4.2 - File Links 29
3.5 - Revision Table 29

IV. Block Validations 29
4.1 - Mechanical Structure and Motors Block Validation 29
4.1.1 - Description 29
4.1.2 - Design 30
4.1.3 - General Validation 34
Vertebrae Validation 34
Reach Requirement Validation 34
Load Bearing Requirement Validation 35
Pulley Validation 37
4.1.4. Interface Validation 38
4.1.5 - Verification Plan 38
4.1.6 - References 40
4.1.7 - Revision Table 40
4.2 - Microcontroller Communication Block Validation 40
4.2.1 - Description 40
4.2.2 - Design 41
4.2.3 - General Validation 42
4.2.4 Interface Validation 42
4.2.5 - Verification Plan 44
Control System Feedback Loop -> ATMega128 Microcontroller 44
ATMega128 Microcontroller -> Control System Feedback Loop 44
4.2.6 - References and File Links 45
4.2.7 - Revision Table 45
4.3 - Simulator Visualization Block Validation 45
4.3.1 - Description 45
4.3.2 - Design 46
4.3.3 - General Validation 47
4.3.4 - Interface Validation 47
4.3.5 - Verification Plan 49
Inverse kinematic controller -> display 49
User input -> inverse kinematic controller 49
4.3.6 - References and File Links 49

4.3.7 - Revision Table 49
4.4 - Control System Loop Block Validation 50
4.4.1 - Description 50
4.4.2 - Design 50
4.4.3 - General Validation 51
4.4.4 - Interface Validation 51
4.4.5 - Verification Plan 53
4.4.6. References and File Links 53
4.4.7 - Revision Table 53
4.5 - Coordinate Input Block Validation 54
4.5.1 - Description 54
4.5.2 - Design 54
4.5.3 - General Validation 56
4.5.4 - Interface Validation 56
4.5.5 - Verification Plan 57
4.5.6 - References 58
4.5.7 - Revision Table 58
4.6 - Power Supply and Distribution Block Validation 58
4.6.1 - Description 58
4.6.2 - Design 59
4.6.3 - General Validation 61
Component Selection 61
Input Interface Validation 63
4.6.4 - Interface Validation 64
4.6.5 - Verification Plan 65
4.6.6 - References 65
4.6.7 - Revision Table 66
4.7 ATMega128 and Accelerometer Block Validation 66
4.7.1 - Description 66
4.7.2 - Design 66
4.7.3 - General Validation 72
4.7.4 - Interface Validation 73
4.7.5 - Verification Plan 73
4.7.6 - References 73
4.7.7 - Revision Table 74
4.8 - Encoder Sensors Block Validation 74
4.8.1 - Description 74
4.8.2 - Design 74
4.8.3 - General Validation 75
4.8.4 - Interface Validation 76
4.8.5 - Verification Plan 77

4.8.6 - References 77
4.8.7 - Revision Table 77

V. System Verification Evidence 77
5.1 - Universal Constraints 77
5.1.1 - System may not include a breadboard 77
5.1.2 - Must contain a student designed PCB, and a custom Android/PC/Cloud application 78
5.1.3 - If an enclosure is present, the contents must be ruggedly enclosed/mounted as evaluated by the course instructor 78
5.1.4 - If present, all wire connections to PCBs and going through an enclosure (entering or leaving) must use connectors 78
5.1.5 - All power supplies in the system must be at least 65% efficient 78
5.1.6 - The system may be no more than 50% built from purchased ‘modules’ 78
5.2 - The system will extend 1.0 meters in any direction 78
5.2.1 - Requirement 78
5.2.2 - Testing Processes 79
5.2.3 - Testing Evidence 79
5.3 - The system will support a mass of 0.5 kilograms 80
5.3.1 - Requirement 80
5.3.2 - Testing Processes 81
5.3.3 - Testing Evidence 81
5.4 - The system will be accurate 81
5.4.1 - Requirement 81
5.4.2 - Testing Processes 81
5.4.3 - Testing Evidence 81
5.5 - The system will be reliable 81
5.5.1 - Requirement 81
5.5.2 - Testing Processes 81
5.5.3 - Testing Evidence 81
5.6 - The system will be self-powered 81
5.6.1 - Requirement 81
5.6.2 - Testing Processes 81
5.6.3 - Testing Evidence 81
5.7 - The system will be portable and easy to use 82
5.7.1 - Requirement 82
5.7.2 - Testing Processes 82
5.7.3 - Testing Evidence 82
5.8 - The system will output a user-friendly 3D visual. 82
5.8.1 - Requirement 82
5.8.2 - Testing Processes 82
5.8.3 - Testing Evidence 82

5.9 - The system will support input in spherical, cylindrical, and cartesian format 83
5.9.1 - Requirement 83
5.9.2 - Testing Processes 83
5.9.3 - Testing Evidence 84
5.4 - References and File Links 85
5.5 - Revision Table 85

VI. Project Closing 86
6.1 - Future Recommendations 86
6.1.1 - Technical recommendations 86
6.1.2 - Global impact recommendations 86
6.1.3 - Teamwork recommendations 86
6.2 - Project Artifact Summary with Links 86
6.3 - Presentation Materials 86
6.4 - Revision Table 86

I. Overview

1.1 - Executive Summary

The goal of this project is to develop a robotic tentacle arm that can move freely within a 3D environment with a high degree of accuracy, while maintaining the cost of the final product at under $300. Robotic arms available on the market can be roughly divided into two categories, high-level control for high cost, and low-level control for low cost. The aim of this project is to develop a highly posable robotic tentacle arm with accurate and repeatable positioning that can be inexpensively reproduced using 3D-printable models and software that will be made open source. Due to its target cost of $300, the arm will be a viable choice for less expensive ventures.

The tentacle arm will use a coordinate system in order to position itself accurately and consistently in 3D space. It will be able to move an object or attached manipulator accurately up to within 2.0 cm within its coordinate range. The project will also include a control system in the form of a user application. The user will be able to enter numerical or qualitative input into the application and the arm will move according to the specified instructions. The tentacle arm will be 3D printed and made using inexpensive components so that it can be replicated easily and at low cost.

Use cases of the finished product include in manufacturing and robotics automation, remote-operation in hazardous work environments, and in prosthesis technologies.

1.2 - Team Communication Protocols and Standards

This section includes the team communication protocols table, the team standards table, and the team individuals' contact information. Contact between group members will mainly be conducted over Discord.

Table 1.2.1 - Protocols and Standards Table

Topic	**Protocol**	**Standard**
On-time Deliverables and Team Collaboration	Team members should complete all parts of project tasks that are given to them before the set deadlines.	Complete work will include all given requirements for said task to be included in the final item.
Task Management	Team will use Google Sheets for task assignments assigned to individuals and to the group.	During team meetings, the team will review tasks to be completed and assign new tasks as needed to team members who have finished. Team members will update the sheet with their %completed to keep everyone else up to date on progress.

Completed assignments and documentation	When finishing documents, they will be uploaded to the teams shared google drive so they can be accessed and reviewed by everyone	Complete work will be put in the correct folders in the google drive so it can be accessed easily by group members.
Discord Communications	Team members are expected to check-in on the team Discord channel on a daily basis, and confirm that they've seen messages from other members.	Every member will check at least once a day and should have notifications on so they receive important information quickly. When asked questions directly, communication time should be within 24 hours.
Absences	Team members are expected to be present for scheduled meetings or classes unless specified beforehand.	All members will be present unless they have told the group they will be absent at least 3 hours beforehand.

Table 1.2.2 - Communication Analysis

Protocol	**Assessment Parameters**
Project partner role	Project partner will act as a silent partner observing progress and communicating any specifics that the project should be focusing on. Additionally if the team needs assistance they can reach out to the project partner for assistance.
Project partner profession	Project partner works on a company that mass produces mechanical arms for factories and assembly lines. He oversees the project design of said arms.
Project partner expectations	Arm should have a design that will allow it to move in 3 dimensions. The arm should be produced with materials costing less than $300 total. The partner would like access to this information as the project is being designed, and would like to know what materials will be used. They will also like to know what functions the arm will have
Project Partner knowledge level	Project partner should know all terminology related to building, designing and producing a robotic arm. He will also have lots of engineering experience and knowledge, but may not have a tremendous amount of knowledge about coding and programming arms.

Project partner communication and sharing.	Project partner prefers communication through email and wants periodic project update emails sent to them. They may share progress with their coworkers or bosses who may or may not have the same technical knowledge as them.

Table 1.2.3 - Team Member Contact Information

Member	Phone	E-mail
Ben Chan	971-295-0807	chanbe@oregonstate.edu
Jordan Porter	925-640-9233	portejor@oregonstate.edu
Cale Hallamasek	650-681-7321	hallamac@oregonstate.edu
Triet Nguyen	971-506-6589	nguytrie@oregonstate.edu
Project Partner (None)	N/A	N/A

1.3 - Gap Analysis

While mechanical arms are commonplace in many parts of industry (manufacturing, prosthetics), they are often expensive and require a high level of experience to program. Modern robotic arms can cost anywhere from $500 for a tabletop toy robotic arm, to over $50,000 for industrial grade large-scale robotics [2]. Price is also often related to possibility: robotic arms exhibit multiple points of articulation, with the three most common points of articulation being at the base, shoulder, and wrist of the robotic arm [3]. The last issue with robotic arms is control: The more degrees of freedom a robot has, the more difficult it is to control. Therefore, the targeted gap in the market would be for an arm that is inexpensive, highly articulated, and easy to program/control.

The tentacle arm will have a maximum budget of $300, which will make it affordable. This price point falls below the cost of most small to medium scale robotic arms that are freely available on the market at the moment. The software developed as well as its design will also be made open-source on the web for educational use. Materials used will be consumer-available materials - such as 3D printing and laser cutting - that will allow people to reconstruct their own tentacle arm robot using this project document, as well as the aforementioned open-source design files and code.

The tentacle arm will have many points of articulation and rotation. This will make it highly adaptable to multiple situations, and also give it an edge over traditional robotic arms that often have less points of articulation. Comparable robotic arms used in an industrial automation setting are either three-axis robotics, meaning two degrees of horizontal/rotational motion at the base and one degree of vertical motion at an elbow, or four-axis robotics, which have an additional axis of motion in a tool or manipulator attached to the end of the arm [4].

In regards to control, the tentacle arm will be developed with user friendliness in mind; an easy to use controller and GUI will allow the user to make full use of the arms movement. The user-interface will feature a 3D simulator that will allow the user to preprogram and visualize the motion path of the robot.

1.4 - Timeline

This section will include the timeline table and charts that are being used to manage the project. The timeline table contains information about the target start and end dates for each task, as well as the current percent complete state of each task.

Table 1.4.1 - Project Timeline

TASK NAME	START DATE	END DATE	MANAGER	PERCENT COMPLETE
Fall Term Documentation				
Week 4 Presentation video	10/14	10/21	Group	100%
Draft - Proj. document sec. 1	10/15	10/22	Ben	100%
Week 4 Project partner update	10/15	10/22	Cale	100%
Week 5 Presentation video	10/21	10/28	Group	100%
Draft - Proj. document sec. 2	10/22	10/29	Ben	100%
Week 6 Presentation video	10/28	11/4	Group	100%
Team Communication Evaluation	10/29	11/5	Group	100%
Week 7 Presentation video	11/4	11/11	Group	100%
Week 7 Project partner update	11/5	11/12	Cale	100%
Proj. document sec. 1-2	10/15	11/12	Ben	100%
Week 8 Presentation video	11/11	11/18	Group	100%
Draft - Proj. document sec. 3	11/12	11/19	Ben	100%
Week 10 Presentation video	11/25	12/2	Group	100%
Week 10 Project partner update	11/23	11/30	Cale	100%
Proj. document sec. 1-3	10/15	12/3	Ben	100%
Overall Timeline				
Early Documentation Phase	10/14	11/15	Group	100%
Testing	11/15	12/12	Group	60%
Modular Testing	1/3	3/12	Group	100%
System Level Testing	2/15	3/12	Group	50%
Presentation Preparation	3/12	4/30	Group	30%

Project Closing Documentation	4/30	6/12	Group	30%
Technology Transfer	6/12	6/18	Group	0%
Electrical				
Microcontroller research	11/24	1/7	Triet	100%
Design Power Supply (Component Selection)	1/27	2/4	Ben	100%
Design Power Supply (Schematic)	2/4	2/11	Ben	100%
FGPA research				
Software				
Simulator Research	11/24	1/7	Jordan	100%
Blender Plugin Research	11/24	1/7	Cale	100%
Coordinate Conversion Development	1/28	2/21	Cale	100%
Simulator Development	1/7	1/28	Jordan	90%
FPGA Programming	3/14	4/28	Ben	70%
Mechanical				
Prototype CAD	11/24	12/21	Ben	100%
Prototype 3D printing	12/17	1/21	Ben	100%
3D Model Revisions	1/14	1/28	Ben	100%

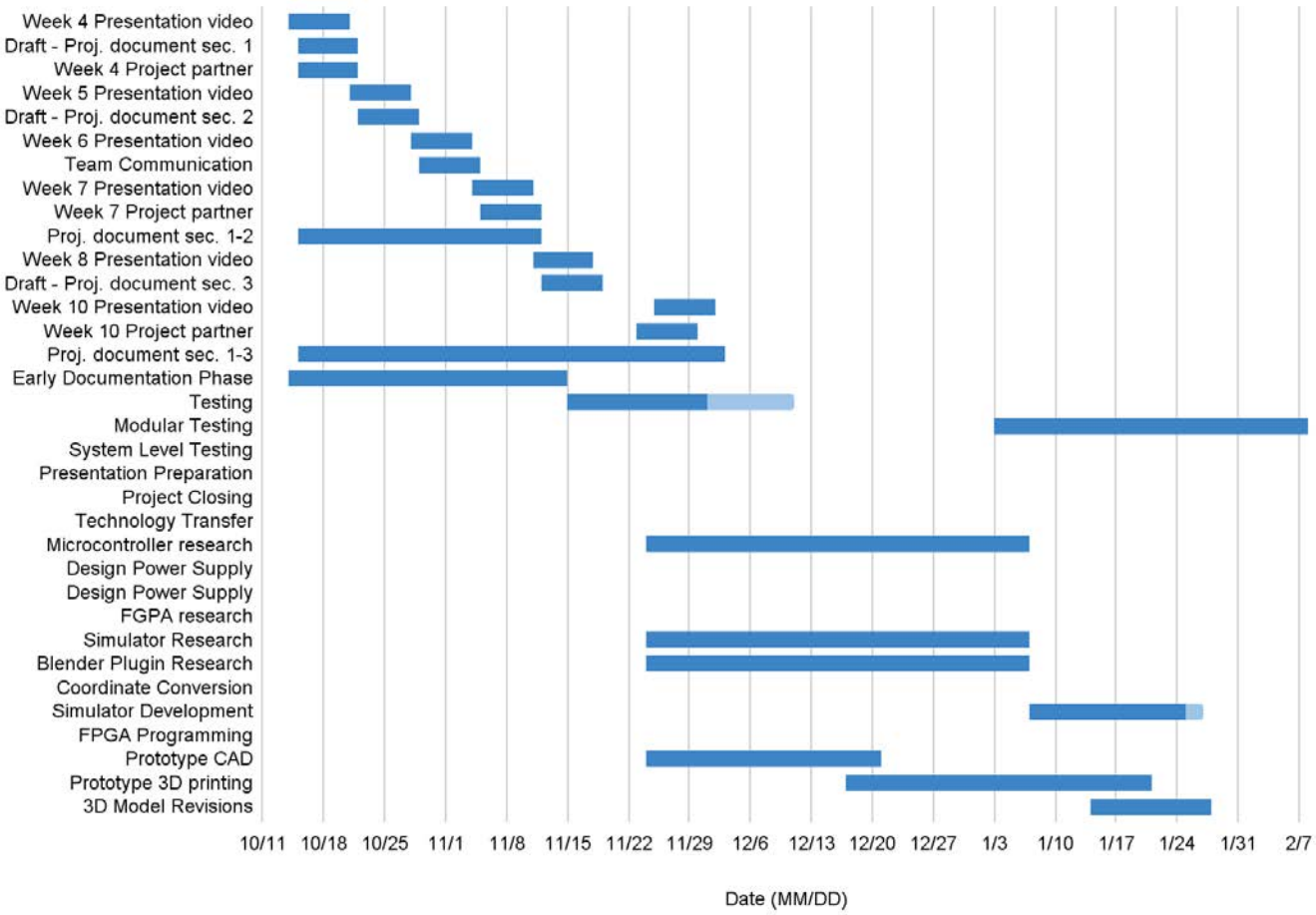


Figure 1.4.1 - Project Timeline Gantt Chart. Dark blue lines on the chart show completed progress. Light blue indicates the timeline for yet to be completed tasks. Vertical gridlines are separated by 7-day increments, aligning with Monday of each week.

1.5 - References and File Links

1.5.1 - References

[1] Heer, Scheel, *Project Documentation Requirements*, Oregon State University, Oct. 2021, https://docs.google.com/document/u/2/d/e/2PACX-1vSTGN-MsnjkEHcNywmy30H1PFOFKHb0VqDk4epHT8vFfdkYmJU_G73ZmjhTbj4UtabMG7anF15-OSYx/pub

[2] *How Much Does Robot Automation Cost?* Robots.com, https://www.robots.com/faq/how-much-does-robot-automation-cost

[3] Ismail, Mohamad & Qi, A & Voon, K & Ismail, Meera & Mustaffa, Nurfathin. (2016). DESIGN AND DEVELOPMENT OF A MECHANISM OF ROBOTIC ARM FOR LIFTING PART5. 10.13140/RG.2.1.2029.0644.

[4] *What is a Four Axis Robot?* Robots Done Right, 2021, https://robotsdoneright.com/Articles/what-is-a-four-axis-robot.html

1.6 - Revision Table

11/12/2021	Ben Chan: Revised gap analysis.
11/11/2021	Ben Chan: Changed formatting.
11/11/2021	Jordan Porter: Revised section 1.1
10/27/2021	Jordan Porter: Created Protocols and Standards Table.
10/27/2021	Ben Chan: Revising section 1 given last week's draft feedback.
10/18/2021	Ben Chan: Filling in sections 1.0 through 1.4.
10/18/2021	Cale Hallamasek: Filling in sections 1.0 through 1.4.
10/18/2021	Jordan Porter: Filling in sections 1.0 through 1.4.
10/18/2021	Ben Chan: Initial document creation. Formatting and basic content for section 1 added.

II. Requirements Impacts and Risks

2.1 - Requirements

This section goes over the project requirements. Each requirement includes a description, a failure condition, and any quantitative values that may be used to evaluate that requirement.

2.1.1 - The system will extend 1.0 meters in any direction

- **ER:** At full extension, the system will have a minimum reach of 1.0 meters. The system will be able to extend in any direction originating from it's base.
- **Verification:** The system passes if when the tentacle arm system is fully extended in any direction parallel to the ground, the center of the base of the arm is a minimum of 1.0 meters away from the end of the tip of the tentacle.

2.1.2 - The system will support a mass of 0.5 kilograms

- **ER:** At full extension, the system will support a mass of 0.5kg placed at the tip of the tentacle. The system must be stable and be able to move while the system is supporting the mass. The mass can take the form of an attachable manipulator (grabbing device) or a non-functional weight with a measured mass of 0.5kg.
- **Verification:** The system passes if it moves at full extension while supporting a load with a mass of at least 0.5kg.

2.1.3 - The system will be accurate

- **ER:** The system will be able to move to the user-specified location with an accuracy of ±2.0cm.
- **Verification:** The system passes if it is able to move to a location within 2.0cm of the user-specified coordinates.

2.1.4 - The system will be reliable

- **ER:** The system will move to the user-specified location with a success rate of 90%.
- **Verification:** The system passes if it is able to move to the user-specified location 9/10 times.

2.1.5 - The system will be self-powered

- **ER:** The system will operate continuously for a minimum of 2 hours on a fully-charged battery. Batteries will be rechargeable.
- **Verification:** The system passes if it is able to operate continuously without recharging for 2 hours.

2.1.6 - The system will be portable and easy to use

- **ER:** The system will be portable and user friendly. 9/10 users will agree that the user interface and system are portable and easy to use.
- **Verification:** The system passes if 9/10 users agree that the interface and system are easy to use.

2.1.7 - The system will output a user-friendly 3D visual.

- **ER:** The system will output a 3D simulation of the arm for motion control and user reference. Users should be able to move the real arm to a target location using only the 3D model in the simulation.
- **Verification:** The system passes if 9/10 users are able to move the arm to a location within 2.0cm of a target location using the 3D simulation.

2.1.8 - The system will support input in spherical, cylindrical, and cartesian format

- **ER:** The system will accept input coordinates in spherical, cylindrical, or cartesian (XYZ) formats. If the coordinates are not within the range of the arms motion, then the system will warn the user.
- **Verification:** The system passes if it is able to parse the three formats of input coordinates correctly, and indicate to the user if those coordinates are within the bounds of the arms motion with 95% success in 30 trials.

2.2 - Design Impact Statement

2.2.1 - Introduction

This document serves to explore the impacts that the development of a robotic tentacle arm will have on society and the environment. This project's main focus is on the development of a low-cost robotic tentacle appendage with the ability to manipulate small objects. The project aims to design and manufacture an inexpensive and easily reproducible tentacle robot driven by

a compliant cable system. The tentacle arm will have a length of approximately 3 to 4 feet. Possible applications of such a technology include automation in industry, prosthesis technologies, and manipulation of hazardous materials in a research setting. For areas of negative impact discussed in this document, plans will be made to mitigate their potential effects.

In each of the following five sections, a different area of impact of this project will be discussed. Firstly, in regards to health and safety, the use of robotics in job automation and in jobs concerning hazardous environments, as well as the application of the technology to the field of prosthesis will be discussed. Next, the negative impacts of robotic job automation on the working class in the manufacturing sector will be considered, as well as how those effects are disproportionately spread through the population. Third, this paper will bring up the effects on the environment regarding plastic waste and energy consumption. Fourth, the economic factors of the robotic tentacle project will be discussed. The document will then conclude with recommendations for going forward with this project while minimizing any negative impacts. Robotics and their applications can provide health and safety benefits. The automation of jobs in manufacturing or dealing with hazardous materials serves to increase the safety of those in dangerous environments. By removing workers from dangerous work environments or tasks, robotics can be used to improve their health and safety. Furthermore, the development of the tentacle arm robot will help improve technologies that are also used in robotic prosthetics.

2.2.2 - Public Health, Safety, and Welfare Impacts

Safety is the number one priority. One of the many possible uses of the robotic tentacle arm will be in working jobs that require a high-level of precision in dangerous work environments, such as in hands-on assembly jobs with dangerous materials, or factory work where workers are subject to extremely loud noises. According to the CDC, manufacturing jobs are disproportionately represented in terms of workplace injuries. While manufacturing accounts for only 5% of US employment, over 8% of US workplace fatalities can be attributed to manufacturing [1]. One of the goals for the development of this robotic arm is to make it reliable enough such that it can be used for remote operation in dangerous work environments. by allowing workers to control the appendage remotely, they can perform many of the same manufacturing tasks without putting themselves at risk of harm.

Another area where robotic arms could be used is in jobs that deal with hazardous materials. This includes transportation, manufacturing, and disposal of hazardous materials and waste. During a study performed by the CDC and the HSEES, nine states were monitored for chemical related incidents. Between 1999 and 2008, a total of 57,975 separate chemical incidents were reported from those nine states, during which 15,506 people were reported to have been injured [2]. 70% of injuries reported involved the volatilization or aerosolization of the hazardous material. By making the switch to remote-operated tools or robotic automation, the majority of those hazardous material incidents could have been avoided. A robotic arm capable of manipulating objects with a high degree of precision and repeatability would be able to perform those dangerous tasks instead, and therefore improve the safety of workers in environments with hazardous materials.

There is a second to robotics in job automation that must be brought up when discussing safety and public health, and this is the fact that robots are dangerous. Wolters Kluwer gave an example of an accident that occurred in 2015 that involved the death of a factory worker [3]. According to the source, “the sensor’s alarm [of a robotic, driverless forklift] was triggered by a piece of plastic wrap underneath the elevated forks of the LGV. Unfortunately, the victim did not

initiate the emergency stop, and was crushed by the forklift after it resumed its automated functions." Since the technology is so new, the OHSA has yet to have clear guidelines in place. As of early 2020, the strongest robot being used in the industry for large-scale assembly was the M-2000iA/2300 [4]. Capable of lifting 2,300 kilograms and reaching a distance of up to 4.6 meters away from its base. This means that there is a 30 foot diameter circle in which the robot is able to apply almost 23,000 newtons of force in any location. Without the proper safety precautions, this could be deadly for the unaware worker.

To avoid the risk of robotics and automation related injuries, there are a few steps that can be taken. Firstly, before using the arm, a 'danger-zone' should be marked out in a circle around the tentacle arm. This would help prevent unaware persons from wandering into an area in which they may be hit. Secondly, the use of external sensors or emergency stop triggers to avoid injuring any person within the effective reach of the tentacle arm. Automation safety companies such as SICK [5] are already using features like this that immediately stop all motors, preventing the arm from hitting a user at high speed. Alternatively, the same sensors could be used to disable all motors in the system, such that if they do hit a user, the cables in the arm are slack and thus exert less force on impact. Thirdly there are also non-software related technical solutions such as using compliant materials that would cause less harm in the case of impact, such as flexible plastics or soft outer coatings [5].

Within the United States, approximately 185,000 people have one or more limbs amputated each year, and in total, the US is home to nearly 2 million amputees [6]. The development of the robotic tentacle arm project would generate research and technologies that would be beneficial to the field of robotic prosthesis, specifically in regard to lightweight robotics with a high degree of control. By helping to improve robotic prosthetic technologies, this project would be having a positive impact on the health and wellbeing of some of the amputees.

2.2.3 - Cultural and Social Impacts

While removing workers from dangerous work environments does provide a positive impact in terms of health and wellbeing, it also has a negative effect. The downside to removing workers from dangerous environments is that they are also being removed from their employment, and therefore their source of income.

It's been estimated by robotics and AI researchers that upwards of 50% of all jobs (both manufacturing and service jobs) will be automated by robotics or AI by the early 2030's [7]. This claim is supported by data from the United States Bureau of Labor Statistics; ever since the number of manufacturing jobs in the US peaked in the 1980's, jobs in the manufacturing sectors have been on the decline. As of the most recent data in 2021, the number of manufacturing jobs in the US has fallen by over 30% [8]. By developing a low-cost robotic appendage with a high degree of freedom, this project may help speed along the automation of jobs within the US and in doing so lead to unemployment for those working unskilled labor jobs. While it would be ideal for more industries to make use of remote-controlled robotics for increasing safety in the workplace, full robotic automation has greater appeal.

Another concern is that the automation of manufacturing jobs will disproportionately affect people of color and already disadvantaged communities. In 2020, the percentage of manufacturing workers who identified as black or hispanic was 27.6%, while the percentage of those in professional or technical services (doctors, researchers, accountants, etc.) who identified as black or hispanic was only 17.0% [9]. By introducing a technology that will assist in

the automation of jobs in the manufacturing sector, this project will have a disproportionate impact on historically disadvantaged communities of color.

The final social impact that arises is in regards to the manufacturing of parts - specifically the treatment of employees in factories that produce motors, microcontrollers, and other electronic parts. Foxconn is a well known Taiwanese multinational electronics manufacturer, and as of 2019 have accrued repeated violations in regards to occupational rights at its factory in China. According to BBC News [10], “the investigation found that workers put in over 100 hours of overtime a month during peak production season, violating Chinese law which says monthly overtime cannot exceed 36 hours”. There are other numerous regulations that the company has violated. With the current chip shortage and worker shortage, it is not a surprise that these manufacturers have increased the workload of their current employees to meet the high rising demands.

Luo Fuxing, a Chinese factory worker, compared Chinese factories to American prisons: “He read that American criminals had tattoos of spiders’ webs inked onto their elbows to show time spent behind bars. Mr Luo got one too, because ‘factory was just a bigger prison’” [11]. These experiences clearly show how the parts used in our tentacle arm project have significant social impacts that might go unseen if this research wasn’t conducted. These unfair labor practices not only put workers in danger, but also affect the productivity of the company. A healthy work environment can create opportunities for employees to contribute to an organization’s growth. In an industry such as the tech industry, productivity and growth are extremely significant. To mitigate this impact, companies from countries with ethical work policies will be prioritized in parts selection. Unfortunately, it is inevitable that some of the parts will be sourced from overseas companies where workers rights are not as well protected or enforced.

2.2.4 - Environmental Impacts

The majority of non-electrical components for the tentacle arm project are intended to be 3D printed or laser cut. 3D printing was selected as the main method of manufacturing because it allows the creation of high-complexity parts at a low manufacturing cost while generating very little waste. Laser cutting will be reserved for parts where greater strength is required, however; laser cutting typically creates more waste and pollutes the environment more than 3D printing.

The most used form of 3D printing available to the typical consumer is fused deposition modeling (FDM). FDM printers work by reheating a thin filament of plastic from a spool, and ejecting it at computer defined positions to create any object. According to a CleanTechnica article about whether or not 3D printing can be described as a “clean” technology, the main environmental benefits come from the type of filament used, as well as the additive manufacturing approach [12]. Additive manufacturing is where objects are built-in layers from an empty base, the opposite of subtractive manufacturing. While this is an improvement due to how complex the 3D printing process is, many printed objects may not turn out the way that they are expected to, resulting in the need to reprint parts. The downside to FDM printing is that it requires a high amount of energy to create plastic spools and later remelt them for the 3D printing process. It’s estimated that it takes approximately 0.5kWh per hour of print time for the average consumer’s FDM printer [13]. For comparison, the typical home microwave uses 0.3kWh per day (assuming 15-30 minutes of usage)[14]. For large-scale prints running an entire day continuously, a single FDM printer can easily use more than 12kWh; equivalent to running a microwave continuously for 40 hours.

The material used contributes strongly to the environmental impacts of any product; in this case, the 3D printing filament is very eco-friendly. The CleanTechnica article mentions that one of the most common 3D printing filament materials is PLA. This material is “derived from biological sources, often from plants like corn”, and that it “is not toxic to organisms like other types of plastics” [12]. While the material is biodegradable, it can take centuries to fully decompose, so it cannot be thoughtlessly discarded into natural environments. Although PLA appears to be the best option when it comes to mitigating environmental harm, we will remind the user that the plastic should still be properly discarded, and that extra parts should be printed only if necessary.

To mitigate the environmental impact of the project in regard to energy consumption, the tentacle arms motion should first be simulated to determine if the 3D modelled parts would be suitable for the required motion range. This would help avoid printing/cutting multiple part iterations in which only minor changes are being made. Secondly, the parts used should be designed as light as possible. This serves two purposes: firstly to reduce the amount of material being used in manufacturing, and secondly to reduce the total amount of mass being moved, and therefore reducing the energy expended for the system.

2.2.5 - Economic Factors

One of the primary goals for the development of the robotic tentacle arm is to keep it low cost, with a maximum development budget of $300. By keeping the cost of the development of the tentacle arm low, it allows reproduction of the technology at an even lower cost.

Modern prosthetic arms typically fall into three price ranges. Cosmetic prosthetics with no robotic functions cost up to $5000, while low-end robotic arms can cost up to $10,000. A fully equipped myoelectric capable arm can cost $20,000 to $100,000 [15]. Not only are robotic prosthetics expensive, but they are also subject to failure due to prolonged wear; a typical prosthetic limb must be replaced every three to five years [16] or more often if the user outgrows the prosthesis. By restricting the cost of the arm to $300, this project aims to develop a low-cost alternative to typical prosthetics. Another design goal for the tentacle arm project is to make the arm easily accessible and repairable; failure of one or two components shouldn’t warrant replacing the full assembly. This will also help reduce cost in the long run.

Automation technologies used in manufacturing are also expensive. For example, high precision robotics used in automobile manufacturing for welding processes start at $28,000, and can cost as much as $50,000 depending on what options or packages are selected [17]. While a $300 tentacle arm would not be able to replace a $50,000 industrial grade robot, the development of the low-cost alternative could open up future paths towards inexpensive robotics.

An additional cost that comes with any piece of technology is maintenance and repair. Proprietary parts and closed-source designs are commonplace in the industry, resulting in large amounts of e-waste, as products usually get thrown out or replaced if they stop working. Without careful consideration, our product would be similarly wasteful. Solutions include providing publicly available designs and free software, making it much easier for the user to maintain and fix the product. An ongoing movement exists called “Right To Repair”, which promotes the idea of fixing and reusing tech products to avoid e-waste [18]. By supporting this, we can allow for more affordable secondhand use of our product, as well as the freedom for customers to modify and fix the arm for no additional cost.

Producing a low-cost alternative to expensive technologies can help lower prices for those in need, but it also reduces profits for robotics research competitors. One possible negative impact that must be considered is saturating the market with low-cost robotics that reduce profits for other robotics research companies. In doing so, the low-cost technology introduced to the market would therefore lead to a reduction in research capabilities for future robotics technology.

2.2.6 - Accessibility Impacts

A versatile arm could be used to assist differently-abled people in performing tasks that may be difficult for them. Better Aging overviews the upcoming shortage of caretakers for the elderly in Japan, and discusses how robotics can be used to mitigate the labor shortage [19]. However, this would require routines to be programmed on a per-user basis, requiring a lot of work to configure the arm for each user. For those without programming experience, it can be difficult or impossible to customize a robotic device. A solution is to make the arm's programmable interface very easy to understand and use, so that scripts or routines could be easily created.

Accessibility extends to the user's understanding of the device as well. From an engineering perspective, it is an easy mistake to assume that if a device can be used by the team creating it, it will be easy for others to use as well. To avoid making this mistake, documentation should still be written in a technical manner, but the inclusion of an accessible user guide could help those with a non-technical background. Practices such as using accessible language, as noted in Google's documentation style guide, help both developers and non-developers by writing for a wider audience [20]. Further examples of these practices include avoiding ableist language, designing web documents for screen readers, and defining acronyms or abbreviations so that they are made clear to those who may be unfamiliar with them.

2.2.7 - Conclusion

The development of the robotics tentacle arm will have both positive and negative impacts on public health and safety, culture and society, the environment, and the economy. In addition, the tentacle arm system will also have accessibility impacts due to its possible applications in automation and relevance to prosthetics. In regards to future development of the tentacle arm, it would be wise to keep these impact factors in mind, especially during the design and manufacturing phase.

As with most robotics related research and development, the future use of the developed technologies in the field of automation will serve to improve the safety and health of those working in hazardous conditions or in manufacturing jobs. But similarly, by advancing automation technologies the development of the robotic tentacle arm may also lead to unemployment for those whose jobs become automated in the future. The possible automation of jobs in the manufacturing sector is particularly important as the manufacturing sector disproportionately employs people of black or hispanic identity when compared to jobs in service industries that cannot be automated. By helping to automate certain industries, this project may help further the divide between communities. One solution to these problems is to develop the robotic tentacle arm with the goal of human-operation and human-cooperation. Instead of fully automating the tentacle arm with artificial intelligence or preprogrammed maneuvers, it would instead be more effective to develop it as a remote-operation tool that workers can be trained in. As a manufacturer tool rather than a manufacturer replacement, the robotic tentacle arm would be able to assist workers in dangerous work environments, while

avoiding the issue of replacing workers in manufacturing jobs. Human and robot cooperation is dangerous, however; by maintaining proper safety protocols and designing safety features such as emergency stop and proximity sensors, the number of injuries caused by the robot can be minimized.

Another key area of impact to keep in mind is the environmental impact of the tentacle arm. During the design and manufacturing phase, it would be smart to design 3D printed or laser cut parts in such a way to minimize material waste and energy consumed, to reduce the carbon footprint of the project. Reducing the material used in the parts would also reduce the mass being moved in the system, which would save energy used in the motors. Aside from designing parts to be as light as possible, another possible solution to material waste generated from 3D printing is to determine a viable bio-compostable FDM 3D printing filament that would suit the material needs of the project. Such a material would need to be selected based on four factors: its weight, cost, durability, and printing temperature. If a biocompatible filament is not as strong or is more expensive than the ABS alternative then it may not be ideal. Alternatively, if it is just as strong and at the same price point, but uses a much higher printing temperature, then the increase in energy consumption may outweigh the reductions in material waste produced.

Future system iterations should require more impact evaluations, especially as the design progresses and use cases become better defined. The introduction of new items, such as interfaces for software, documentation, and physical components will all need to be evaluated to ensure that they are properly usable and do not have preventable negative externalities.

2.3 - Risks

This section will include the risks assessment table. The table includes columns for the risk description, category, probability, impact, indicator, and the action plan for the risk. The risks in the risk assessment table have been assigned to individual members of the group.

Table 2.3.1 - Risks Assessment Table

ID	Description	Category	Prob. (%)	Impact (L, M, H)	Performance Indicator	Party	Action plan
R1	3d printing issues/error with produced parts	Technical/ timeline	20%	M	All 3d printed parts should work correctly for arm to function optimally	Ben	Avoid 3D printing issues. Fix/retain parts that are salvageable.
R2	App to arm interface is incompatible	Technical	20%	H	Application should be able to connect to arm	Cale	Get basic communication working first. Redesign if necessary.
R3	User interface incompatible with control system	Technical	10%	H	Controller should be able to connect to the application. Arm should move when controlled.	Cale	Retain and fix the application so that it is compatible (correct output format).
R4	Production cost exceeds budget	Financial	5%	M	Production cost should be under $300, and should leave	Jordan	Avoid selecting expensive

					emergency funds to deal with risk factors.		components. Reduce additional costs via redesign.
R5	Incompatible or defective parts	Technical/ financial	10%	M	Assembly should contain all originally intended parts.	Ben	Reduce risks of purchasing incompatible parts.
R6	Order delays	Technical	30%	M	Parts arrive on time	Ben	Reduce order wait time by purchasing early or from alternative sources.
R7	Incompatible power supply	Technical	5%	H	Power supply voltage and current should be compatible with what's required for the PCB	Triet	Redesign compatible system.
R8	PCB or microcontroller failure	Technical	5%	H	Boards should exhibit the intended behavior. PCB/microcontroller should not break or overheat after prolonged use.	Triet	Redesign hardware. Replace components if needed.
R9	Broken parts during testing	Technical	20%	H	All circuit components should function together as intended	Ben	Avoid broken parts by ordering earlier and testing components. Design with repairability in mind.

2.4 - References and File Links

2.5.1 - References

Citations

[1] *Inputs: Occupational Safety and Health Risks,* Center for Disease Control, United States of America, Nov. 22, 2019, cdc.gov/niosh/programs/manuf/risks.html

[2] Duncan MA, Wu J, Neu MC, Orr MF; Centers for Disease Control and Prevention (CDC). Persons injured during acute chemical incidents—Hazardous Substances Emergency Events Surveillance, nine states, 1999-2008. MMWR Suppl. 2015 Apr 10;64(2):18-24. PMID: 25856534. Accessed at pubmed.ncbi.nlm.nih.gov/25856534/

[3] G. Gould, "The impact of Robotics on Safety and Health," *Wolters Kluwer*, 05-Nov-2019. [Online]. Available: https://www.wolterskluwer.com/en/expert-insights/the-impact-of-robotics-on-safety-and-health. [Accessed: 05-Dec-2021].

[4] Matthews, Kayla. *8 noteworthy lifting and mast robots and applications,* Robotics Business Review, April 3, 2020,

www.roboticsbusinessreview.com/manufacturing/8-noteworthy-lift-mast-robots-applications/

[5] Montaqim, Abdul. *5,000 deaths a year: A reminder of why safety systems matter*, Robotics & Automation News, May 24, 2020, https://roboticsandautomationnews.com/2020/05/25/5000-deaths-a-year-a-reminder-of-why-safety-systems-matter/32403/

[6] Ziegler-Graham K, MacKenzie EJ, Ephraim PL, Travison TG, Brookmeyer R. Estimating the Prevalence of Limb Loss in the United States: 2005 to 2050. Archives of Physical Medicine and Rehabilitation 2008;89(3):422-9. Accessed at www.amputee-coalition.org/resources/limb-loss-statistics/#1

[7] Thomas, Mike. *Will a Robot Take Your Job? Artificial Intelligence's Impact on the Future of Jobs,* Built In, July 26, 2021, builtin.com/artificial-intelligence/ai-replacing-jobs-creating-jobs

[8] *Employees, thousands, manufacturing, seasonally adjusted,* U.S. Bureau of Labor Statistics, United States of America, Nov. 2, 2021, data.bls.gov/timeseries/CES3000000001

[9] *Labor Force Statistics from the Current Population Survey,* U.S. Bureau of Labor Statistics, United States of America, 2020, bls.gov/cps/cpsaat18.htm

[10] "Foxconn broke Chinese labour laws-rights group," BBC News, 11-Jun-2018. [Online]. Available: https://www.bbc.com/news/business-44436250. [Accessed: 05-Dec-2021].

[11] Guangzhou, "How Chinese factory-workers express their views on life," The Economist, 14-Aug-2021. [Online]. Available: https://www.economist.com/china/2021/08/12/how-chinese-factory-workers-express-their-views-on-life. [Accessed: 05-Dec-2021].

[12] Jennifer Sensiba. "Is 3D Printing A Clean Technology?", *CleanTechnica*, January 25, 2021. [Online], Available: https://cleantechnica.com/2021/01/25/is-3d-printing-a-clean-technology/ [Accessed: October 24, 2021].

[13] Brooks, Mike. *How Much Power Does a 3D Printer Use?,* M3DZone, July 21, 2021, m3dzone.com/3d-printer-power-usage/

[14] Wollerton, Megan. *How to Buy a Microwave*, Cnet, Feb. 18, 2019, cnet.com/home/kitchen-and-household/microwave-buying-guide/

[15] Vandersea, Jamie. *Arm & Hand Prosthetics,* Medical Center Orthotics & Prosthetics, mcopro.com/blog/resources/arm-hand-prosthetics/

[16] Mohney, Gillian. *Health Care Costs for Boston Marathon Amputees Add Up Over Time,* ABC News, April 24, 2013, abcnews.go.com/Health/health-care-costs-boston-marathon-amputees-add-time/story?id=19035114

[17] *How Much Does Robot Automation Cost?* RobotWorx, 2017, www.robots.com/faq/how-much-does-robot-automation-cost

[18] “Right To Repair”, Oregon State Public Interest Research Group. [Online], Available: https://ospirg.org/feature/orp/right-repair [Accessed: October 24, 2021].

[19] “Robots may become caretakers for the elderly”, Better Aging, January 23, 2021. [Online], Available: https://www.betteraging.com/aging-technology/will-robots-become-caretakers-for-seniors/ [Accessed: October 28, 2021].

[20] “Writing accessible documentation”, Google, October 20, 2021. [Online], Available: https://developers.google.com/style/accessibility [Accessed: October 28, 2021].

[21] Heer, Scheel, *Project Documentation Requirements*, Oregon State University, Oct. 2021, https://docs.google.com/document/u/2/d/e/2PACX-1vSTGN-MsnjkEHcNywmy30H1PFOFKHb0VqDk4epHT8vFfdkYmJU_G73ZmjhTbj4UtabMG7anF15-OSYx/pub

2.5 - Revision Table

5/6/2022	Ben Chan: Revised and added Design Impact Assessment. Formatted relevant citations.
11/30/2021	Triet Nguyen: Revised action plan
11/26/2021	Ben Chan: Revised requirements using instructor feedback.
11/11/2021	Ben Chan: Updated risk table with more specificity. Changed the wording of some requirements.
11/11/2021	Jordan Porter: Revised section 2.1 and 2.3
10/27/2021	Ben Chan: Revising risk table and requirements.
10/26/2021	Jordan Porter: Worked on section 2.1 and 2.3
10/18/2021	Ben Chan: Formatting and basic content for section 2 added. Creation of risk table.

III. Top-Level Architecture

This section includes block diagrams and interface definition tables. Sub-blocks of the block diagram will be shown and explained in detail in the following sections.

3.1 - Block Diagrams

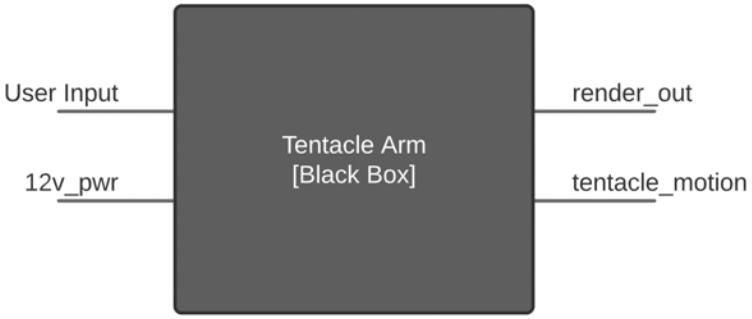


Figure 3.1.1 - Black Box Block Diagram

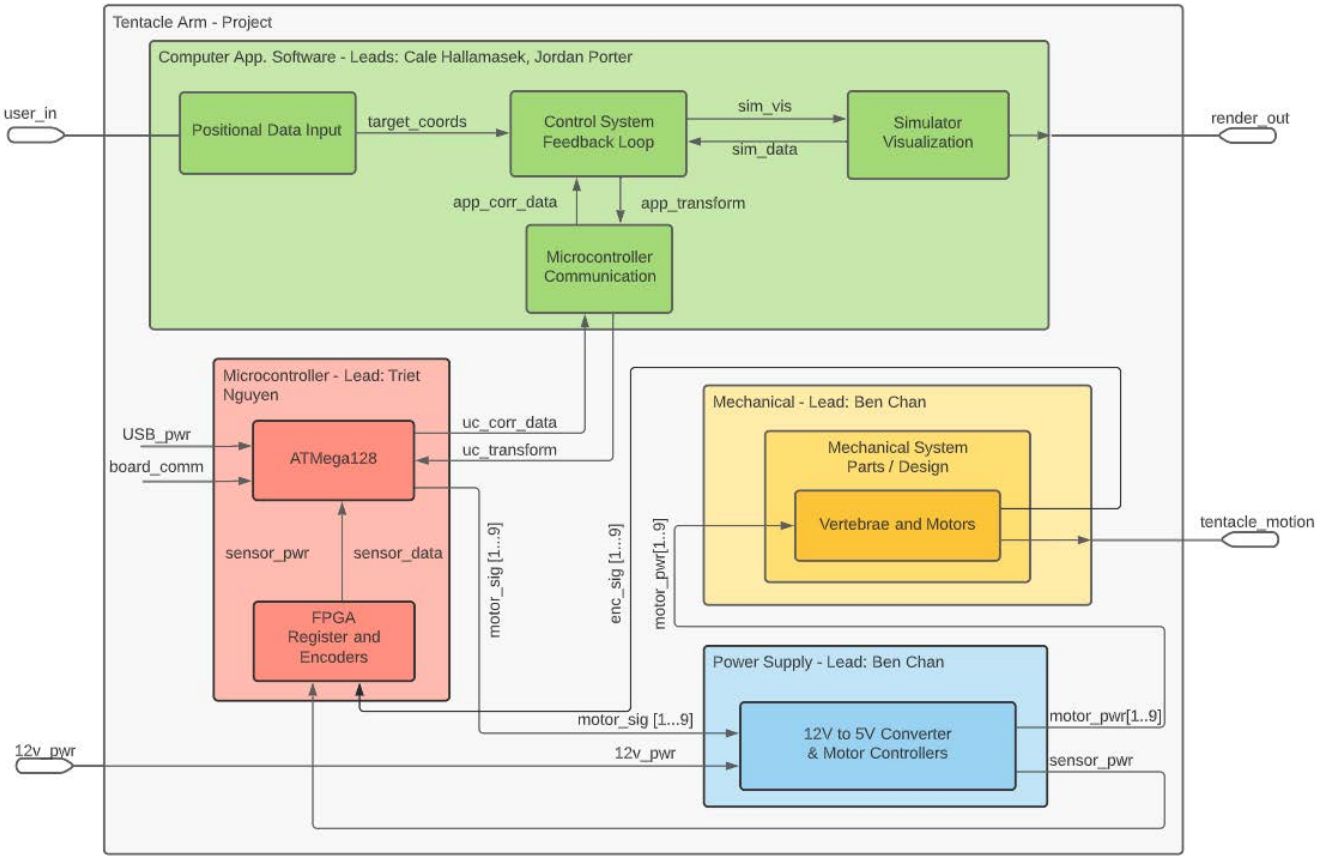


Figure 3.1.2 - Full block diagram

3.2 - Block Descriptions

3.2.1 - Software Blocks

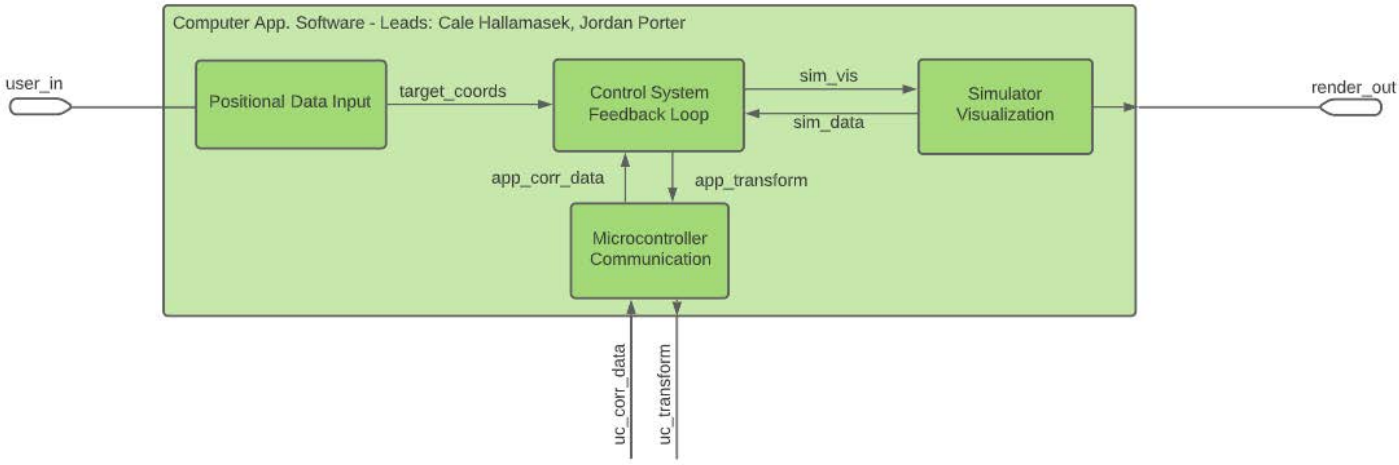


Figure 3.2.1 - Computer Application Sub-Block

Positional Data input - Lead: Cale Hallamasek

Refers to the method of coordinate input provided by the user. Coordinates are input in cartesian, spherical, or cylindrical format. The application will then convert these coordinates into normalized cartesian format coordinates that refer to the arm's end goal position. Coordinates are relative to the arm's position, and can either be specified via the visual interface or with manual input.

Control System Feedback Loop - Lead: Cale Hallamasek

This represents the most mathematically complex process, where the arm takes its current position and end position and uses inverse kinematics to solve for where each arm joint should end up. This process results in a set of instructions that will be used to drive the motors.

Microcontroller communication - Lead: Jordan Porter

The positional data obtained through the previous step will be sent to each motor; in this case, there are three tentacle sections that will receive this positional data. The signal being sent will need to be optimized for communication lag (due to wireless communication) and to avoid errors in commands that are sent.

Simulator Visualization - Lead: Jordan Porter

The simulator serves two functions: Firstly to provide the user with visual feedback of the arm's current position and orientation, and secondly to help the user plan out the arm's future movement and target locations. The simulator will run an inverse-kinematics rigged model of the arm that can be dragged into position.

3.2.2 - Microcontroller Blocks

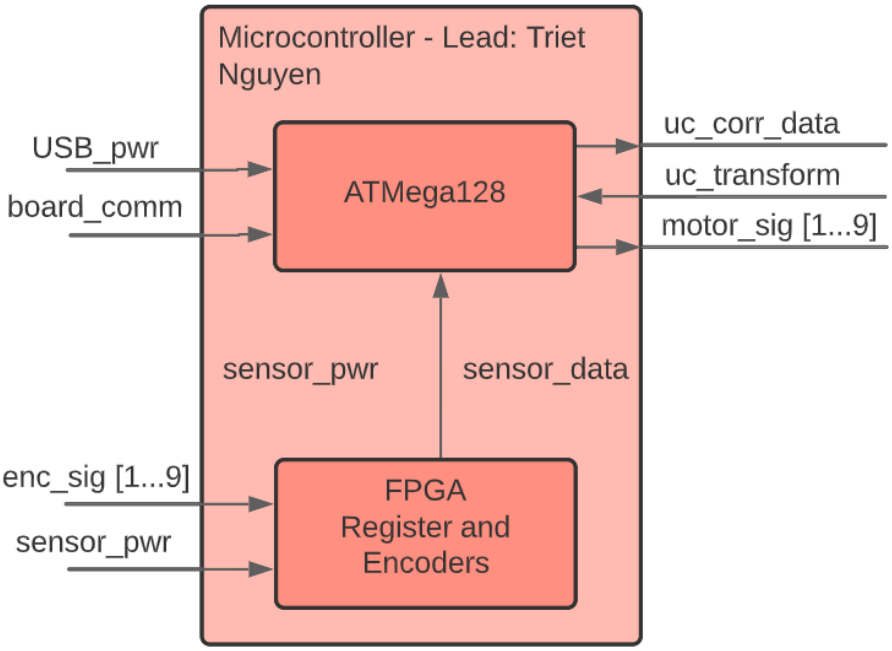


Figure 3.2.2 - Microcontroller Sub-Block

ATMega128 and Accelerometer - Lead Triet Nguyen

This block represents the ATMega128 microcontroller, and it's connected peripherals. The ATMega128 will control the motors, and also gather positional feedback data using the accelerometer. This device is powered by a 5V power source. By using its I/O ports, it can be used to interface with both the mechanical and software components of the arm. The accelerometer will be used to gauge the arm's movement progress, and help correct for potential error after movement instructions are completed. The accelerometer will poll the arm's position continuously, and compare it to the end goal to determine what adjustments will need to be made to get it closer to its target.

Communication Antenna - Lead: Triet Nguyen

The antenna is used by the ATMega128 to receive control signals from the computer application wirelessly. The ATMega128 will then parse these signals and output them into a format that is readable by the motor controllers.

3.2.3 - Mechanical Blocks

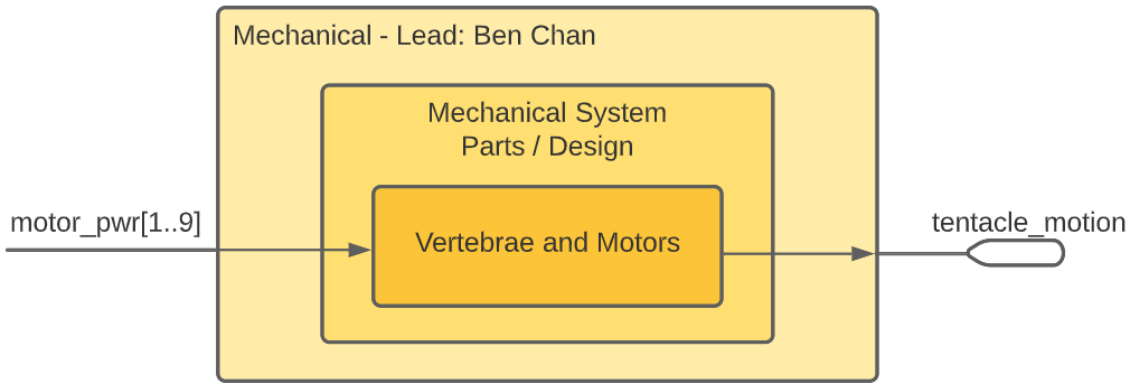


Figure 3.2.3 - Mechanical Systems Sub-Block

Mechanical System Parts / Design - Lead: Ben Chan

The mechanical portion of the arm will be made up of a total of 12 discs, forming 3 individually controllable sections of 4 discs each. The assembly will be driven by 3 motors per section, with a count of 9 main motors. Each motor will have its own motor controller. Additional motors will be used for manipulator or base motion. 3D modelled parts will be imported into the 3D simulation as well.

3.2.4 - Power Supply Blocks

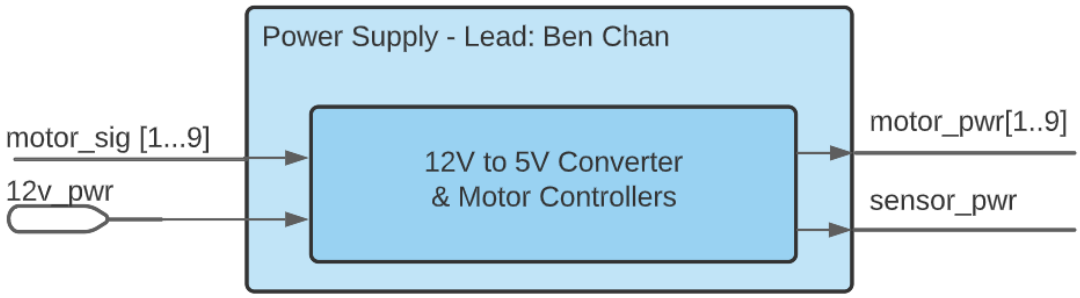


Figure 3.2.4 - Power Supply Sub-Block

Power Supply - Lead: Ben Chan

The main power source for the system will be a 12V battery bank that can be removed and recharged. This will allow the tentacle arm to be used in places without a readily accessible power outlet. The battery bank output is passed into a buck converter to reduce it to a voltage that the other blocks in the system can handle. A buck converter provides step-down DC to DC voltage. The output from the converter will be 5V, as this is the required voltage for the ATMega128 board. The buck converter power supply will pass it's output through an output filter to reduce spikes in voltage. This serves to protect the ATMega from shorts or other issues with the power supply. This block also includes the motor controllers, which will be purchased modules. These motor controllers will use the filtered 12V supply from the PCB to step up the voltage output from the microcontroller and FPGA block.

3.3 - Interface Definitions

Below is the interface definition table. The leftmost column of the table has been color coded to match the colors used for each sub-block of the block diagram. Rows with two colors indicate interfaces that are used between two sub-blocks. Rows color coded with white are interfaces that serve as either inputs or outputs of the overall system.

Table 3.3.1 - Interface Definition Table

I	O	Name	Type	Value	Description
		user_in	User Input	Coordinate or positional data. • Rectangular (X, Y, Z) • Spherical (r, θ, φ) • Cylindrical (r, θ, z) • 3D position from IK simulation	User input can take the form of input target coordinates (XYZ, spherical, cylindrical) or 3D simulation data passed to the system via the 3D simulation plugin.
		app_corr_data	Internal Variable	Parsed positional data (USB) • Accelerometer feedback. • XYZ delta from previous.	Provides correctional data from the microcontroller back to the control system application.
		app_transform	Digital Signal	List of nine motor commands. (USB)	Data sent from the application to the board on the arm. Signals will be passed along to the motor controllers to control the 9 motors.
		target_coords	Internal Variable	Converted coordinate data • Spherical (r, θ, φ) • Data from IK rigged model	System converts input data (coordinate or positional data) into standardized rectangular coordinates. Outputs to IK control system.
		sim_data	Internal Variable	IK and rigging information	IK control system data feedback from the Blender simulation.
		sim_vis	Internal Variable	IK and rigging information	IK control system data from the Blender simulation to update the render.
		uc_corr_data	Internal Variable	Parsed positional data (USART) • Accelerometer feedback. • XYZ delta from previous.	Provides correctional data from the microcontroller back to the control system application.
		uc_transform	Digital Signal	List of nine motor commands. (USART)	Data sent from the application to the board on the arm. Signals will be passed along to the motor controllers to control the 9 motors.
		render_out	GUI Output	Render Viewport	3D environment visualization within the Blender viewport.
		12v_pwr	DC Power	Volts: 12.0V	Input 12V sourced from a wall jack, or a portable battery.
		motor_sig [1..9]	DC Power, control	Volts: 0 - 5.0V	0 to 5.0V is the supported range on the motor controllers.
		sensor_pwr	DC Power	Volts: 5V	5V stepped down and filtered the supply.
		motor_pwr [1..9]	DC Power, control	Volts: 0 - 6.0V	While the motor controllers can output up to 12.0V at maximum output, the motors have a rated long-term voltage of 6.0V.

	motor_sig [1..9]	PWM signal	Array of 9 integer values in the range (0, 255)	A total of nine individual control signals that are being sent from the microcontroller to the motor controllers on board the arm.
	tentacle_motion	Mechanical motion	3D motion	The direction and curvature of the 12 discs of the tentacle arm.

3.4 - References and File Links

3.4.1 - References

[1] Heer, Scheel, *Project Documentation Requirements*, Oregon State University, Oct. 2021, docs.google.com/document/u/2/d/e/2PACX-1vSTGN-MsnjkEHcNywmy30H1PFOFKHb0VqDk4epHT8vFfdkYmJU_G73ZmjhTbj4UtabMG7anF15-OSYx/pub

[2] Heer, Cate, *Interface Definitions,* Oregon State University, Sept. 2021, https://docs.google.com/document/u/1/d/e/2PACX-1vQHvM5VaVp8gZAlKrc_7OZ8alwPbj7lUCdJBM0rMAcclGtVhzSiX5KGrePpB5ura0iWkksPuQpPyFBX/pub

3.5 - Revision Table

3/3/2022	Jordan Porter: updated with updated block diagrams
12/1/2021	Ben Chan: Simplified block diagram (previous version was too low-level, as per feedback). Revised interface definition table and block descriptions.
11/26/2021	Ben Chan: Revisions and formatting fixes.
11/18/2021	Ben Chan: Added sub-block diagrams, revised interface table.
11/17/2021	Ben Chan: Created first draft of section 3. Created initial block diagrams and tables.

IV. Block Validations

4.1 - Mechanical Structure and Motors Block Validation

Champion: Ben Chan

4.1.1 - Description

This block validation is for the mechanical structure design, and motor implementation in the robotic tentacle arm project. The mechanical system is made up of 18 vertebrae discs, forming 3 individually controllable sections of 6 discs each. The assembly is driven by 3 motors per section, with a total count of 9 main motors. Motors control the system using 9 tensioning cables, and can curl the tentacle by applying a tension force to the vertebrae. Each motor has its own motor controller.

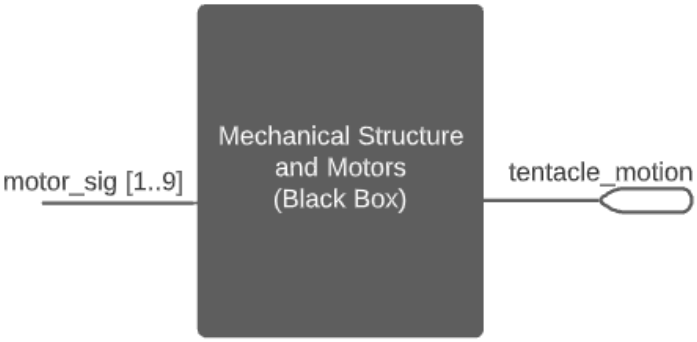


Figure 4.1.1: Mechanical systems, black box diagram.

3D modeled parts have also been imported into the 3D simulation in Blender, and rigged using inverse-kinematic relationships. The 3D models are a part of this block, while the Blender rigging and simulation will be detailed later in a separate block validation.

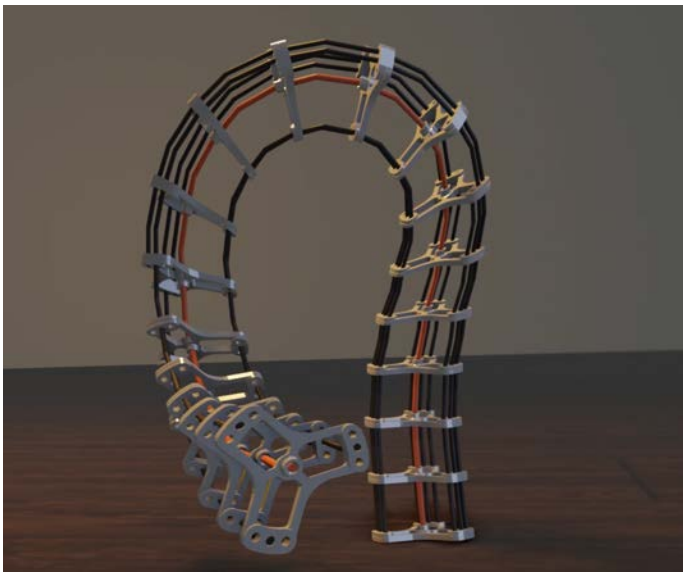


Figure 4.1.2: Tentacle model rendered in Blender.

4.1.2 - Design

This section will cover the design components of the mechanical systems of the tentacle arm. This encompasses the block diagrams (black box, and expanded), design drafts, and part drawings and dimensions.

In the following figures, tentacle positioning has been constrained to 2D in both hand drawn and computer rendered models. This is simply for clarity purposes. The final system will have full 3D motion (see rendered image in Figure 1.2 above). This first figure is the black box diagram of the mechanical systems for the tentacle arm. The only input is the motor_sig [1..9] interface that includes the motor signals for the 9 motors. The output of the mechanical system is the tentacle_motion interface, which as its name describes, is the physical motion and position of the tentacle.

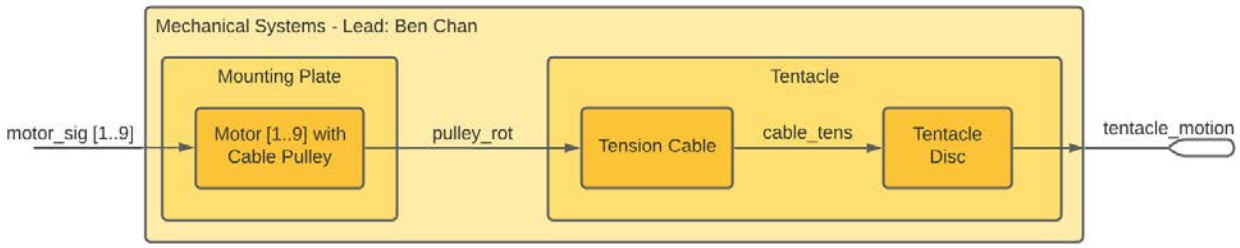


Figure 4.1.3: Mechanical systems, expanded block diagram.

This is the expanded block diagram for the mechanical systems of the tentacle arm. The same input and outputs to the system exist, however now internal functionality is displayed.

The first function of the mechanical system is contained on the mounting plate. The mounting plate provides a stable base to the tentacle, as well as provides space for the microcontroller and 9 motors to be mounted. Each of the nine motors is attached to a 3D printed pulley that will be press-fit to the motor shafts. Rotary hall-effect sensors will be mounted on the motors to keep track of revolutions. Communication with the sensors is further explained in the validations for the microcontroller and sensor blocks.

The second function is the control of cable tension via the rotational motion of the pulley, which is controlled by the motors on the mounting plate. This interface between the cable and the motors has been defined as ‘pulley_rot’. The second interface, cable_tens, is the applied tension to the connected tentacle discs. These functionalities have been split due to distinctions in what considerations must be made in regard to their design and validation.

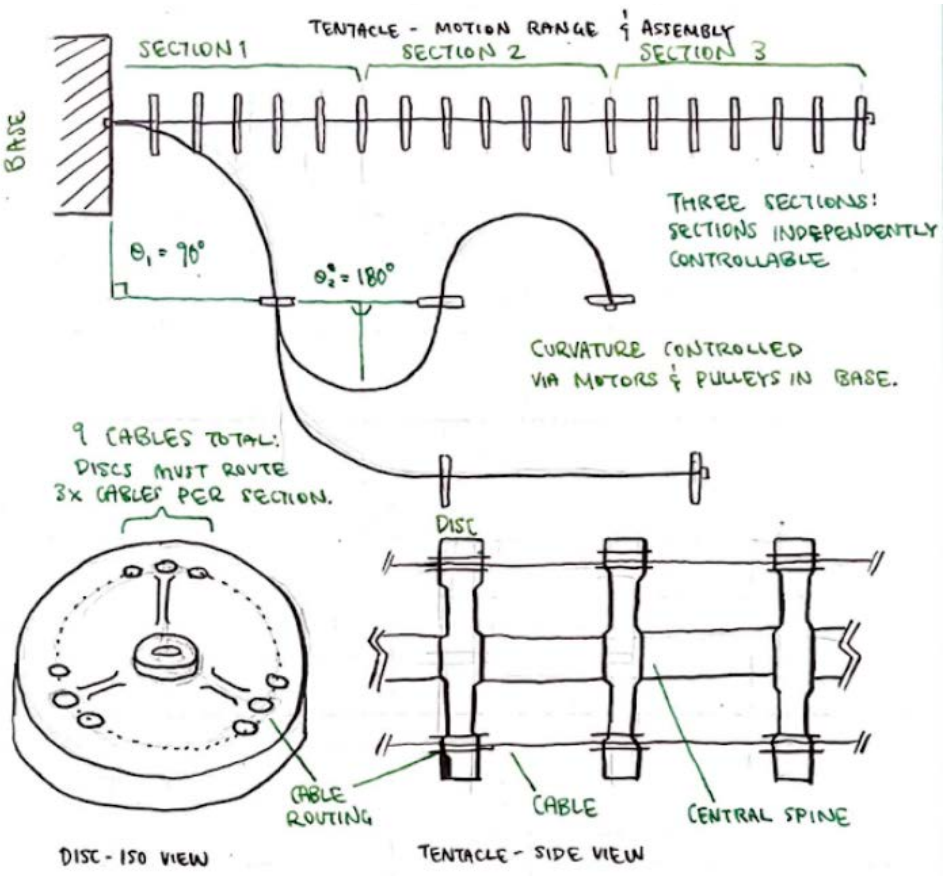


Figure 4.1.4: Early design draft with annotations.

Figure 2.2 above includes an overall side view with various tentacle positions, as well as a closer sliced view of the cables running through the tentacle. An additional isometric view of the disc was included to help clarify where cables would run through the discs. While the 3D modeled design appears different from this early draft, the key features are the same: three controllable sections, and 9 holes for cables.

In figures 2.3 through 2.5 below, the 3D modeled designs for the vertebrae disc and cable pulley are shown. The cable pulley and motor is the interface for pulley_rot, with interface type rotational motion. The vertebra disc which is put under tension by the cables running through the holes is the interface cable_tens. To prevent the tension applied by pulley_rot from compressing/retracting the tentacle, a non-compressible material is used to separate the discs. This material will be pneumatic tubing (OD 0.25”), which sits against a chamfered hole in the center of each disc. The chamfer helps keep the tubing centered, while the hole (OD 0.125”, smaller than the tubing diameter) allows a cable to run through the center of the entire tentacle, helping to keep the discs compressed together.

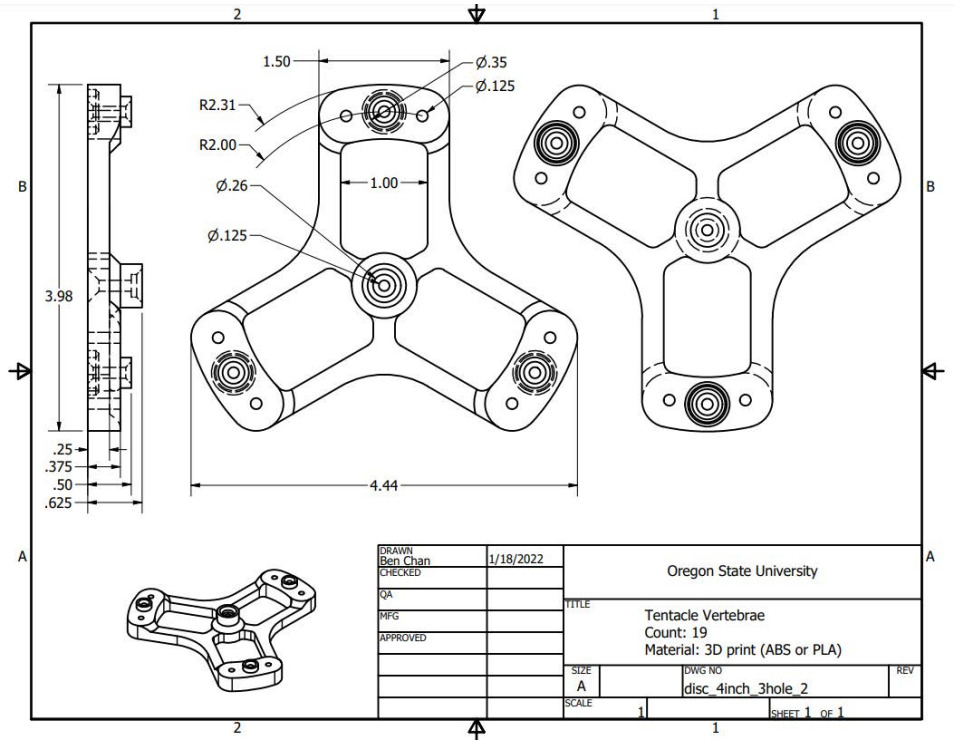


Figure 4.1.5: Vertebrae disc drawing. Only critical dimensions are labeled.

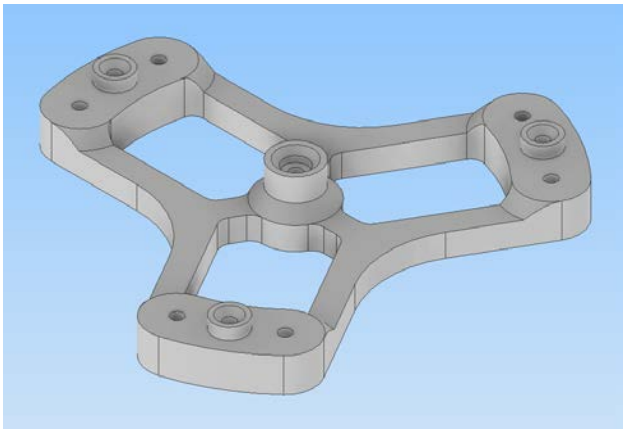


Figure 4.1.6: 3D model of vertebrae disc.

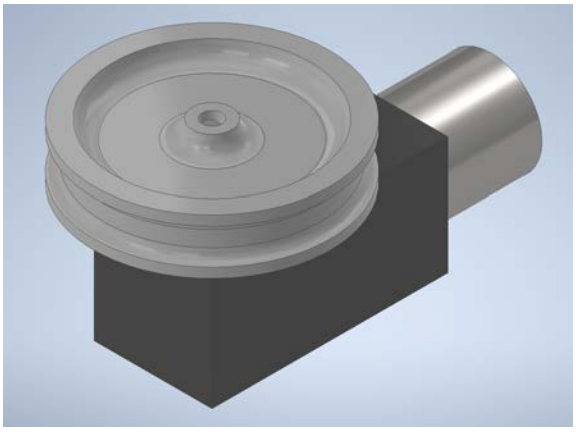
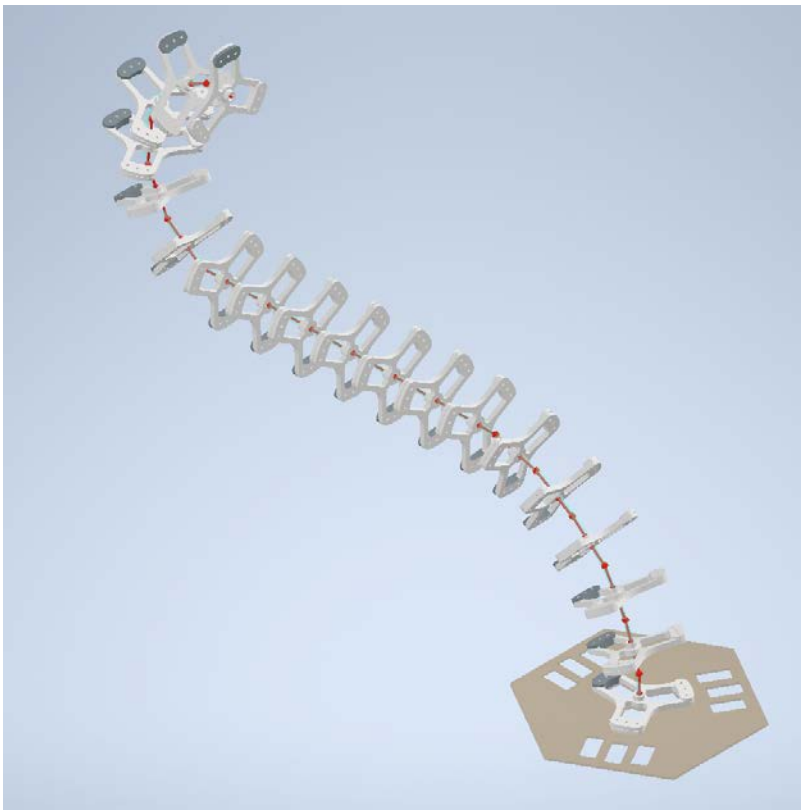


Figure 4.1.7: 3D model of cable pulley on motor.

The following are renders of the tentacle assembly. The overall system comprises 18 vertebrae discs, 9 cables (not pictured), a central cable (pictured in red), and a polycarbonate base plate for mounting the electronics and pulley motors. In the following figures, the three sections of the tentacle have been constrained at 90°, 0°, and 180°(in order from the base of the tentacle toward the tip).

Figure 4.1.8: Tentacle system side view.

Figure 4.1.9: Tentacle system isometric view, with visible base plate.



4.1.3 - General Validation

Vertebrae Validation

The design of the individual vertebrae is based on how the movement of the tentacle will be calculated. The defining values to determine the position of the tentacle are the radial distance of the tension cables from the center of the tentacle, the arclength of the tentacle section, and the angle difference between the first and final vertebrae of the section.

This means that the critical values for the vertebrae design are not the outer diameter of the disc, nor the relative position of the holes for the cables, but only the radial distance of the cables holes, and the absolute distance between the discs. With these values, and the known angle position of the target destination, the arclength of the tentacle (which is the length of the control cables) can be determined.

From figure 2.3 it can be seen that the bottom side of the vertebrae is flat, with minimal cutouts or protrusions other than through-holes. This design was a conscious choice made to improve the speed and quality of parts via 3D printing while requiring minimal post-processing. (Deburring, removal of supports, drilling through holes, etc.) Furthermore, no complex extrusions have been made in any horizontal direction so that the part can be easily mass produced via resin casting or injection molding (at large scale production). As an optional feature, small grooves have been modeled along the outer circumference to aid in drilling holes for mounting set screws if deemed necessary. In this case, the groove was suppressed before 3D printing.

Reach Requirement Validation

One of the system requirements is that the system must have a maximum horizontal reach of at least three feet (36.0”). Using the arc length equation [1], and the known fact that all three sections of the tentacle must have a constant arc length, the minimum distance between discs can be determined. Solving symbolically with *R* being the total reach of the arm, and *x* being the distance between discs, the following equation can be used.

$$R = 6x + 6x + [6x / (\pi/2)]$$

Solving the equation to isolate *x*, and plugging in *R*.

$$x = (R \times \pi) / [12(\pi + 1)]$$

$$x = 2.276" = (36" \times \pi) / [12(\pi + 1)]$$

Round up to 2.50” to comfortably exceed the requirement. Plugging in 2.50” to the original equation yields the following:

$$R = 39.55" = 12(2.50") + [6x / (\pi/2)]$$

Thus the arm has a maximum vertical reach of 45.0”, and a maximum horizontal reach of 36.0”

These values can be checked in the 3D model in Inventor Professional. The following two screenshots show the measured radial distance of the arm in the vertical and horizontal positions.

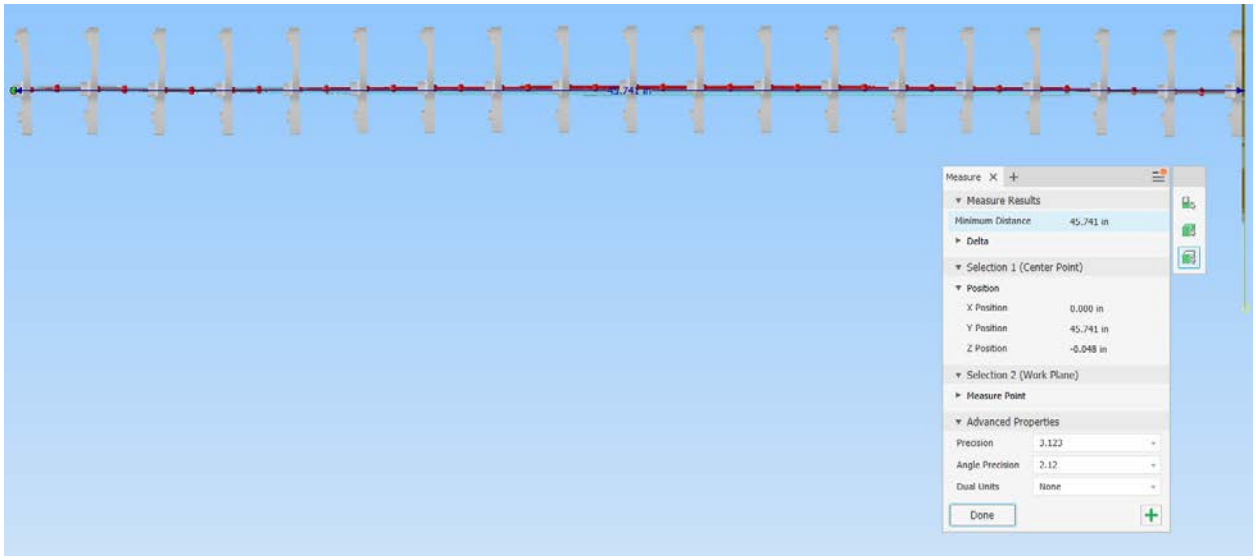


Figure 4.1.10: Vertical reach measurement in Inventor Professional

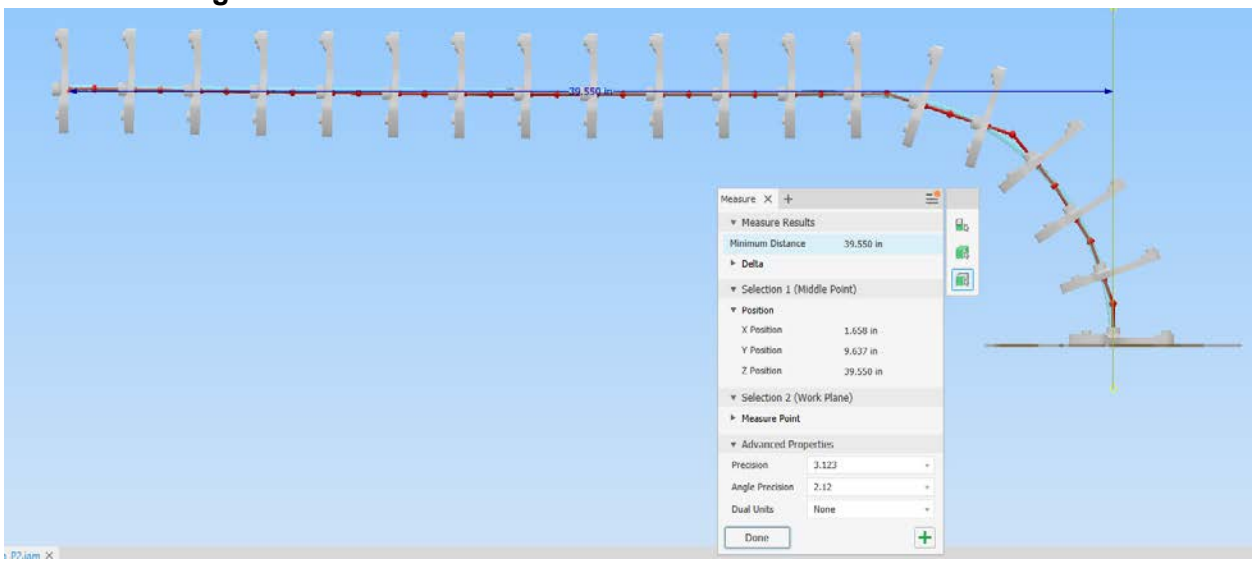


Figure 4.1.11: Horizontal reach measurement in Inventor Professional

Load Bearing Requirement Validation

Another system requirement is for the system to remain free standing while supporting a load of 0.5kg at it's tip in any position. The most extreme case would be when the tentacle is at it's furthermost horizontal position (as calculated above). To determine the physical properties required for this, a torque analysis can be done to determine the tension force applied to the load-bearing cable.

In the following diagram, distances have been converted to their metric values, and weights have been labelled with L, W_1, and W_2 (the load mass, mass of the straight section of the tentacle, and the mass of the curved section respectively).

Torque is determined using the applied force times the length of the moment arm. In this case, the forces on the right side of the arm are known, while the tension force applied to the cable is unknown.

$$\sum \tau = 0\,Nm = a \times Lg + b \times W_1 g + c \times W_2 g + d \times F_T$$

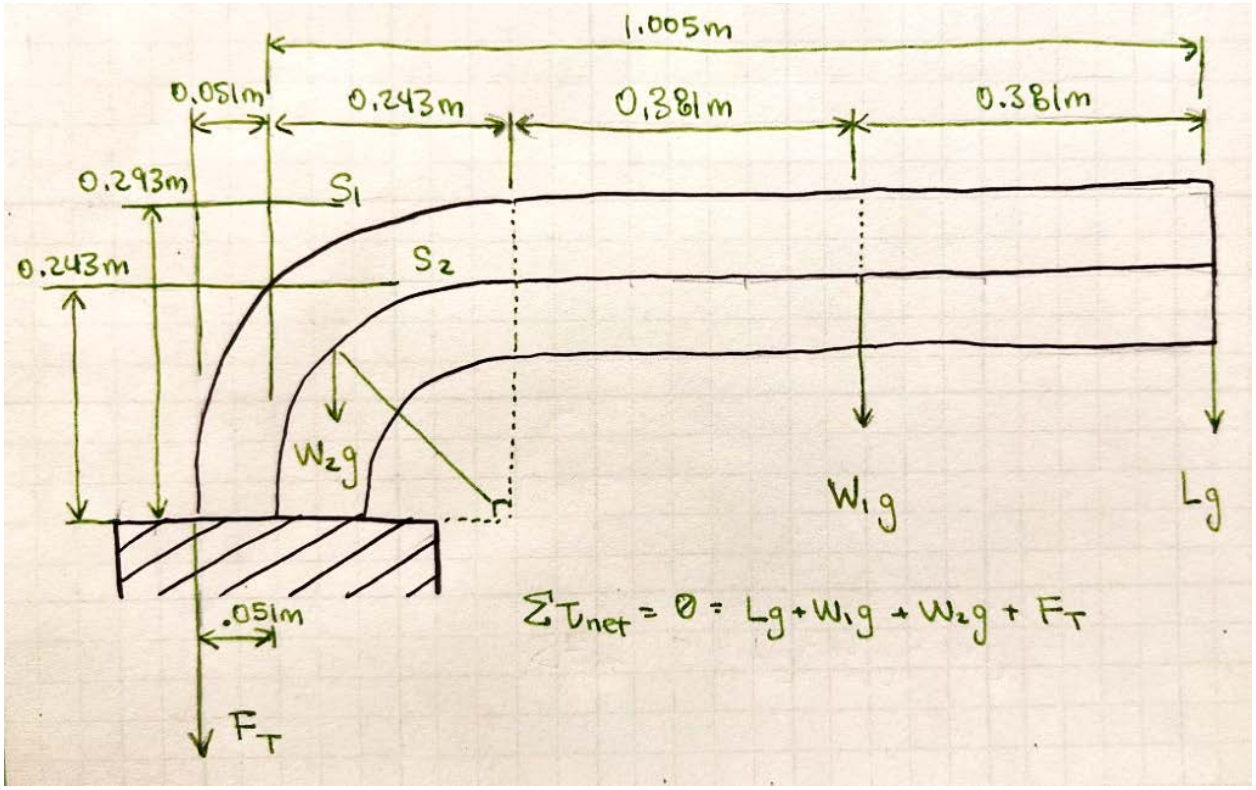


Figure 4.1.12: Torque analysis diagram for a load at the end of the tentacle.

Simplify this equation to isolate F_T, the tension force. The right side is negative, indicating that F_T acts in the opposite direction.

$$F_T = \frac{-g(a \times L + b \times W_1 + c \times W_2)}{d}$$

The values for the distances *a*, *b*, *c* and *d* are determined using the following diagram. In the case of *c*, the center of mass of the curve was estimated to be at an angle of *π/4* radians. These values are plugged into the equation to solve for the tension force. The values for W_1 and W_2 were estimated to be 0.24kg and 0.12kg respectively. (This was determined using the mass estimate of the ABS plastic part in Inventor Professional 2022, and a 3D printing infill of ~70%.)

.

$$F_T = \frac{-g(\,1.005 \times 0.5 + (0.381 + 0.243) \times 0.24 + [0.243\,(\,1 - \sqrt{2}/2)] \times 0.12)}{(0.051)}$$

$$F_T = 127N \approx 12.95kg$$

When at the most extreme position, and carrying a 0.5kg load at the end of the tentacle, the system exerts 127N of tension along the outermost cable under ideal circumstances. This is the equivalent of 12.95kg or 28.55lb. The cable that will be used for controlling the tentacle system is a nylon/fluorocarbon fishing line with a load rating of 50lb [2]. Multiple cables can be used per motor to increase the maximum capacity.

$$28.55 \,/\, (50.0) \;=\; 57.1\%\; of\; maximum\; rated\; load$$

$$28.55 \,/\, (2 \times 50.0) \;=\; 28.55\%\; of\; maximum\; rated\; load$$

Lastly, when not carrying an additional load (ie, only looking at the tentacle system with no additional forces), the tension force applied to the cable is the following:

$$F_T \;=\; \frac{-g(\;(0.381 + 0.243) \times 0.24 + [0.243\;(\,1 - \sqrt{2}/2)] \times 0.12)}{(0.051)}$$

$$F_T \;=\; 30.5N \;=\; 3.1kg$$

3.1kg is the equivalent of 6.8lb. Without the additional load, the system is well within the maximum rated load of a single cable.

Pulley Validation

The validation of the pulleys depends on the specific motors and gearboxes that are used for the system. The motors can be selected knowing that the maximum tension force experienced by the cables will be 127N. Converting this into a torque rating with a pulley radius of 1.50” (0.0381m) gives a minimum requirement of 4.84Nm of torque (0.49g/cm).

Using a larger pulley radius allows the tentacle system to move faster, but also increases the torque requirement. For example, using a pulley with a 3.0” radius results in a required torque of 9.68Nm. This increased radius would allow the tentacle system to accelerate twice as fast, as the winding speed of the pulley is proportional to its circumference, and circumference is proportional to radius: $c \;=\; 2\pi r$.

For the tentacle system, a worm-drive gearbox would be a good selection, since it cannot be back-driven by high torques. One such worm-drive gearbox available on Amazon is $14.99, making it inexpensive as well [3]. Alternatively, conventional motors with high torque ratings would be simpler and less prone to failure. The potential motor and gearbox combination on Amazon is also $14.99, and has a torque rating of 6.0 kg/cm, 1200 times more torque than calculated as necessary [4]. The downside to a conventional motor is that it can be backdriven unless it has built-in braking in which case they can only be backdriven when the system is completely powered down.

Overall, meeting the 4.84Nm torque requirement for the pulley motors is not an issue.

4.1.4. Interface Validation

Interface Property	Why does this interface have this value?	Why do you know that the design details <u>for this block</u> meet or exceed each property?
motor_sig [1..9]		
V_min = -5.0V V_max = 5.0V I_nominal = 0.5A	This is the input signal from the motor drivers. A DC voltage value between -6.0 and 6.0 volts is determined in the microcontroller and sent to the motors.	The motors are rated for continuous operation at up to 6V DC. The rated current of the motors is 0.6A, and it's stall current is 3.3A. Continuous operation at 0.5A is within the rated limits, while short spikes of 2.0A will not trigger a 'stall' condition. The motor drivers have a max continuous current output of 2.0A, and are able to output 5.0V to 35.0V as directed by the logical current input.
tentacle_motion		
F_t, nominal = 30.5N	This is the tension on the cable when in its most extreme position with no external load.	The cable chosen has been rated for 50lb of weight, which is the equivalent of 222.5N of force. 222.5N provides a safety factor of 7.3 over the expected tension force of 30.5N.
F_t, max = 127N	This is the tension exerted on the cable when in its most extreme position with a 0.5kg external load at its tip.	The 222.5N rated cable provides a safety factor of 1.75 over the expected tension force of 127.0N. Taking into account the triangular shape of the vertebrae, the tension will typically be split along two cables. In these situations, the tension force would be 63.5N per cable, and results in a safety factor of 3.5.
Reach, vert. = 40.0"	This is the total vertical length of the tentacle.	The length of the tentacle when all cables are of equal length (within each section) should be 45.0". This is because the vertebrae have a consistent spacing of 2.5". With a total of 18 vertebrae (not counting the base plate), the length is 2.5" × 18 = 45.0".
Reach, hor. = 36.0"	This is the required maximum reach of the tentacle.	The maximum radial distance of the tentacle from its origin point is 39.55". This was calculated using arc length to determine the radius of the tentacle's curvature. 39.55" exceeds the 36.0" inch requirement.

4.1.5 - Verification Plan

This section is the step-by-step verification process for the mechanical tentacle system. This includes testing the motion range of the tentacle, as well as the physical properties of the system (load-bearing capacity, reach, and accuracy of motion). This verification process is to be done after the mechanical systems have been fully manufactured and assembled.

1. Verify voltage and motor response.
    a. If not connected to to the tentacle control system:
        i. Set up a power supply and set it to 0V. Keep current below 0.6A.
        ii. Connect one of the DC motors on the base to a voltage supply.
        iii. Increase the voltage supplied. The motors should move and result in tentacle motion. Be careful not to reach the maximum or minimum travel of the motor.
            1. If the motors do not show movement, it fails.
        iv. Continue to increase the voltage supplied until reaching 5.0V. At this voltage, the tentacle should be moving faster than at the lower voltage rating. Optionally test the motor running in the opposite direction.
            1. If the motors fail before reaching 5.0V, it fails.
        v. Power off the power supply, disconnect motor, and repeat the test on a different motor.
    b. If connected to the tentacle control system (Verify control system independently first):
        i. In the control application, plug-in coordinates that align with the radial direction of the tentacle cable (0°, 120°, 240°).
        ii. Observe tentacle motion. Motion should be smooth and parallel with the targeted direction.
            1. Motion that does not align with the targeted direction indicates that motors in one of the other two directions are failing.
            2. Motion that is not smooth indicates motor or gearbox failure.

2. Determine vertical reach.
    a. Straighten the tentacle system along a flat surface, or if mounted, extend it vertically.
    b. Measure the tentacle's length from the surface of the base plate, to the top of the 18th vertebrae (19th, if including the base plate).
        i. If this distance is not greater than 40.0 inches, it fails.

3. Determine horizontal reach.
    a. Straighten the upper two sections of the tentacle (even tension along the 6 motors for sections 2 and 3).
    b. Bend the lowermost section of the tentacle to 90°.
    c. Measure the radial distance from the origin axis (where the tentacle is mounted to the base) to the tip of the tentacle. This should measure 39.6”.
        i. If this distance is *not* greater than 36.0”, then it fails.

4. Determine maximum load.
    a. With the tentacle in it’s maximum horizontal reach position, attach a 100g load to the tip of the tentacle. The tentacle should remain free-standing.
        i. If the tentacle system breaks, it fails.
        ii. If the tentacle system sags or drops such that the tip touches the ground, it fails.

b. Move the tentacle. The tentacle system should be able to move accurately even when carrying an external load. Test different positions for all three sections, as well as movement of all three sections at once.
    i. If the tentacle system no longer responds to controls, or breaks when carrying the load, it fails.
c. Increase the load by 100g and repeat until the total load reaches 500g.

4.1.6 - References

[1]. CUEMATH, *Arc Length*. Accessed Jan. 7, 2022 [Online]. Available: https://www.cuemath.com/geometry/arc-length/

[2]. OOK, Amazon*, OOK 534608, Clear Wire Supports Up to 50-Pounds*. Accessed Jan. 5, 2022. Available: https://www.amazon.com/OOK-Invisible-Hanging-Supports-50-Pounds/dp/B000FSS39M

[3]. Greartisan, Amazon, *JSX180-370 DC 12V 45RPM Worm Geared Motor 6mm Shaft*. Accessed: Jan. 5 2022. Available: https://www.amazon.com/Greartisan-Geared-Turbine-Reduction-JSX330-370/dp/B071KJ6TC3?th=1

[4]. Greartisan, Amazon, *B071KFT4P7 DC 12V 30RPM Gear Motor 37mm diameter*. Accessed: Jan. 5, 2022. Available: https://www.amazon.com/Greartisan-Electric-Reduction-Centric-Diameter/dp/B071KFT4P7?th=1

4.1.7 - Revision Table

1/21/2022	Ben Chan: Revising with instructor & peer review feedback.
1/8/2022	Revising math in section 3. Finishing sections 4 through 6.
1/6/2022	Filling in sections 1 through 5.
1/5/2022	Created initial document. Initial formatting.

4.2 - Microcontroller Communication Block Validation

Champion: Jordan Porter

4.2.1 - Description

Send Data between the control system feedback loop and the ATMega128 microcontroller. Microcontroller can communicate with a computer using USART serial data communication. The microcontroller communication block will receive data from the control system feedback loop and then send it via USART to the microcontroller. The Atmega128 microcontroller will then send back updated rotations to the microcontroller communication block via USART. The microcontroller block will then send this data back to the control system feedback loop so the

program has the current location of the arm. USART stands for universal synchronous and asynchronous receiver and transmitter [1]. The protocol is used for receiving and transmitting data bit by bit relative to clock pulses on a single wire. The microcontroller uses two pins TXD and RXD specifically for transmitting and receiving data serially and KCK as a clock pin [1]. The Microcontroller uses UDR (a 16-bit buffer register) 8-bits are for transmitting (TXB) and 8- bits are for receiving (RXD) [1].

4.2.2 - Design

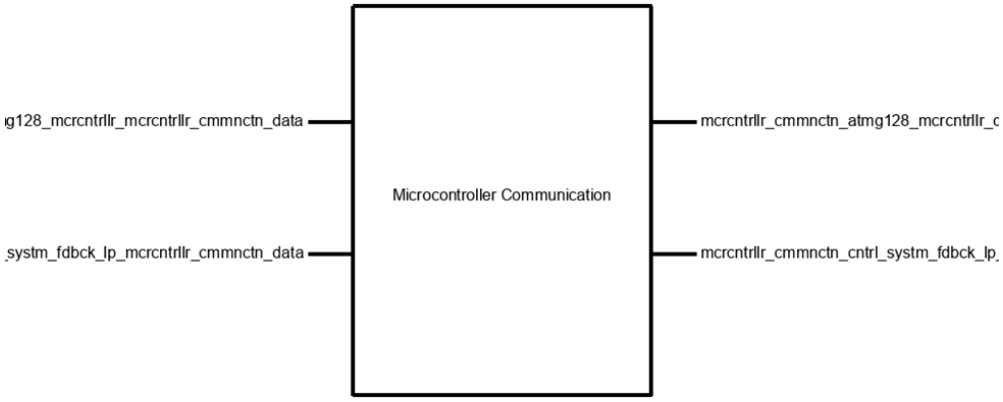


Figure 4.2.1: Block Diagram

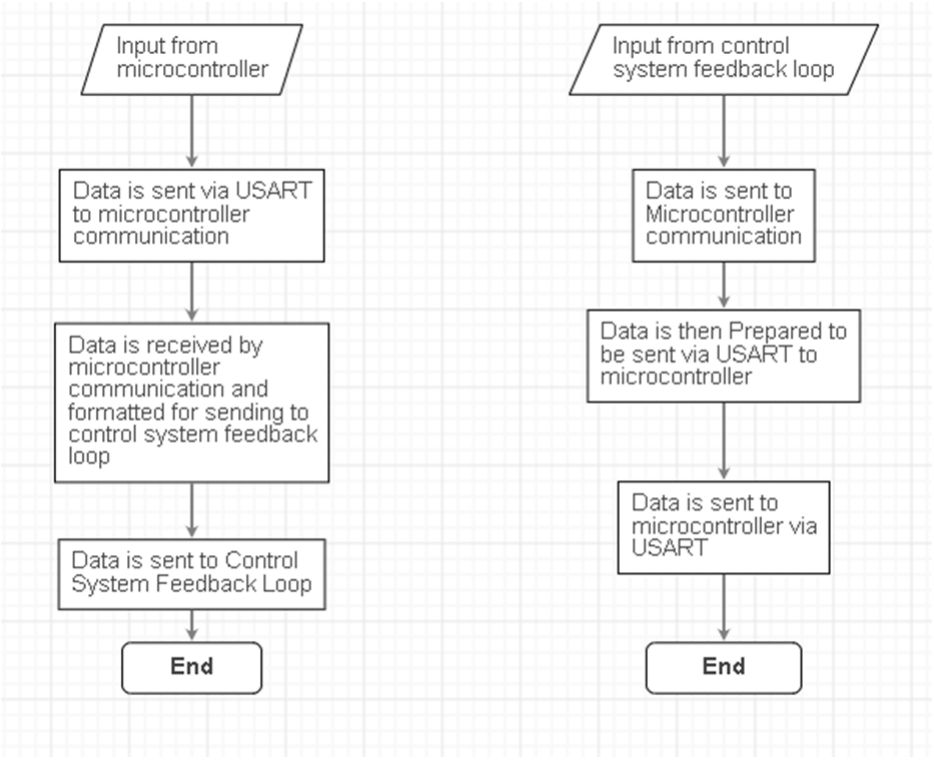


Figure 4.2.2: Flowchart of Code inputs

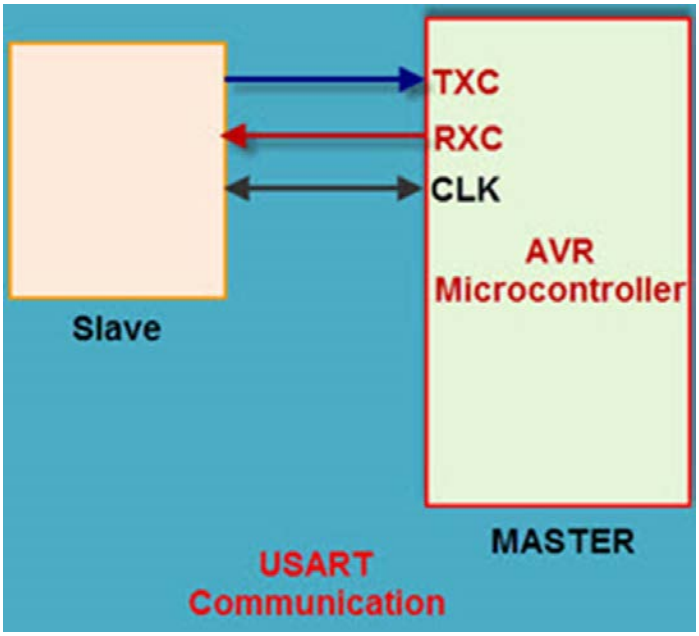


Figure 4.2.3 - USART Communication Diagram [2]

4.2.3 - General Validation

The system needs to be able to communicate from the microcontroller to the computer in order for the Tentacle arm to receive new commands and for the simulation to be updated with the arm's current location. The Atmega128 microcontroller has a built-in function for communication using USART. USART receives data from an outside source bit by bit and stores it into a register for use by the microcontroller. Additionally the microcontroller can also transmit data in the same way. By communicating this way the microcontroller will be able to send and receive data to and from the microcontroller communication code on a computer. USART is faster than traditional UART which will be needed to make sure communication between the microcontroller and computer is as quick as possible and if any error from sending data occurs it can be resent quickly.

4.2.4 Interface Validation

atmg128_mcrcntrllr_mcrcntrllr_cmmnctn_data : Input

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Datarate: Will send data at least 10 times per second	Data needs to be sent fast enough from the microcontroller so the simulation runs smoothly	Data will be sent from the microcontroller at least 10 times per second

Messages: Data will contain Rotations for each motor sent one at a time	Data will contain the current rotations for each motor	Data will be received 1 motor at a time and stored in microcontroller communication to be sent later
Protocol: Will use Usart to send data between atmega128 microcontroller and the Microcontroller communication	USART is a built in communication interface so the microcontroller can talk to outside sources.	USART will be used to communicate between the ATMega128 and the microcontroller communication code

cntrl_systm_fdbck_lp_mcrcntrllr_cmmnctn_data : Input

Datarate: Data will be sent only when a change in final position in the arm is entered by a user	The ATMega128 will need the total rotation needed for each motor to be sent at once so it can move to the final position smoothly	When new rotations are sent from the Control System Feedback loop they will immediately be stored by the microcontroller communication
Messages: Data will be sent in sets of 3 so each motor receives its own rotations	Data will contain all the rotations for every motor	Motors will be in sets of 3 that will be stored in the microcontroller communication
Messages: Data will contain the total amount of rotations for a motor	The data sent for rotations will be the total rotation for each motor	Each Motor will receive the total amount of rotations it needs to complete

mcrcntrllr_cmmnctn_atmg128_mcrcntrllr_data : Output

Datarate: When new rotations for motors are received from control system feedback loop the rotations will be sent to the microcontroller	The ATMega128 will need the total rotation needed for each motor to be sent at once so it can move to the final position smoothly	When new rotations are Stored in the microcontroller communication they will immediately be sent to the microcontroller
Messages: Data will be sent in order 1 at a time until all motors are sent	Data will contain all the rotations for every motor	Motors will be in order that microcontroller will read and will be sent to the ATMega128

Protocol: Will use Usart to send data between atmega128 microcontroller and the Microcontroller communication	USART is a built in communication interface so the microcontroller can talk to outside sources.	USART will be used to communicate between the ATMega128 and the microcontroller communication code

mcrcntrllr_cmmnctn_cntrl_systm_fdbck_lp_data : Output

Datarate: Will send data at least 10 times per second	Data needs to be sent fast enough from the microcontroller so the simulation runs smoothly	Data will be sent as soon as it is received from the ATMega128
Messages: Data will contain Rotations for each motor sent in groups of 3	Data will contain all the rotations for every motor	Motors will be set up in groups of 3 so data will be received in this way
Messages: Data will be the change in rotations from the last Data sent	Rotations will be the current change since last rotation	By sending the change in rotation the simulator will be able to update its current position

4.2.5 - Verification Plan

Control System Feedback Loop -> ATMega128 Microcontroller

1. New Total rotations will be sent for each motor from the Control System to the Microcontroller Communication Block
2. Microcontroller Communication will store all rotations
3. Microcontroller Communication will then send data over USART to the ATMega128
4. Once all data is sent the Microcontroller will stop sending data.
5. Check that the Rotations are sent and received correctly to the ATMega128

ATMega128 Microcontroller -> Control System Feedback Loop

1. Microcontroller Communication will be constantly looking for new Data Sent from the ATMega128 over USART
2. ATMega128 will send new rotations over USART
3. Microcontroller Communication will store new rotations for each motor until all motors are received
4. Check that Data is stored and received correctly
5. Microcontroller Communication will then send all rotation data for every motor to Control System Feedback Loop
6. Microcontroller will then Start Listening for new Data being sent over USART
7. Check that the Rotations are sent correctly to Control System Feedback Loop

4.2.6 - References and File Links

[1] N. *, “Ateml AVR microcontroller serial data communication (USART),” *ElProCus*, 01-Oct-2014. [Online]. Available: https://www.elprocus.com/avr-microcontroller-serial-data-communication [Accessed: 18-Feb-2022].

[2] “USART Communication .” [Online]. Available: https://www.elprocus.com/wp-content/uploads/2014/08/21.jpg.

4.2.7 - Revision Table

3/3/2022	Jordan Porter: added to project
2/18/2022	Jordan Porter: added information from feedback
2/17/2022	Jordan Porter: set up and filled out rough draft of interface validation

4.3 - Simulator Visualization Block Validation

Champion: Jordan Porter

4.3.1 - Description

This block will simulate the arm with a visual model. Simulation will move based on the arms current position. Users will be able to move the model simulation to different positions as input to change the positioning of the arm. The simulation will be made in blender and programmed using python.

4.3.2 - Design

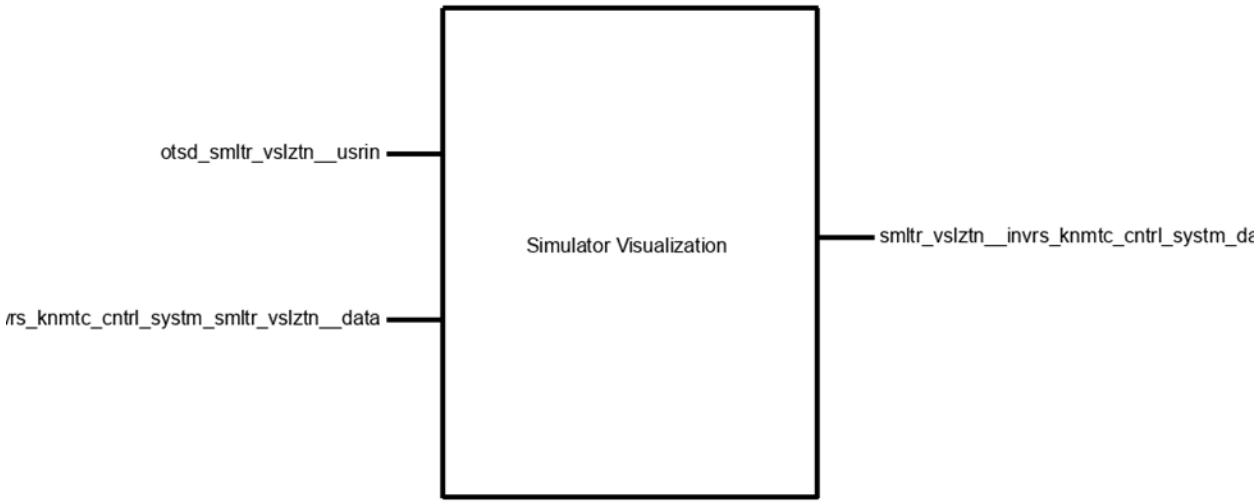


Figure 4.3.1: Diagram

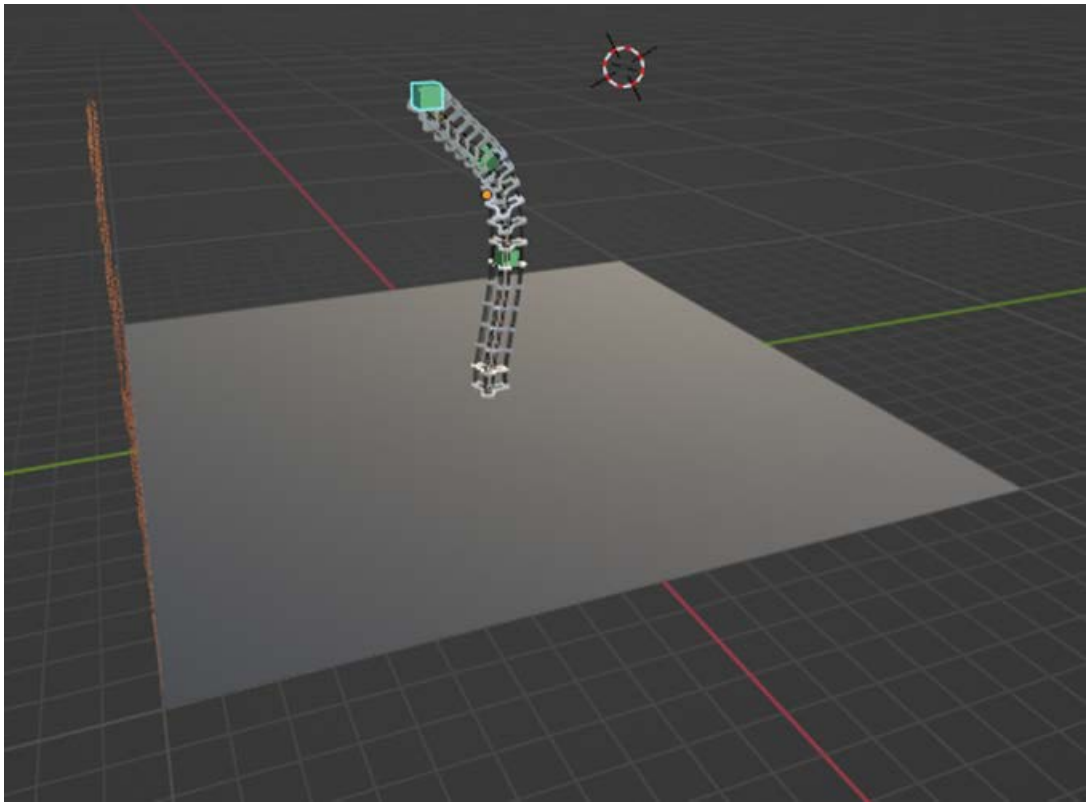


Figure 4.3.2: Image of arm side

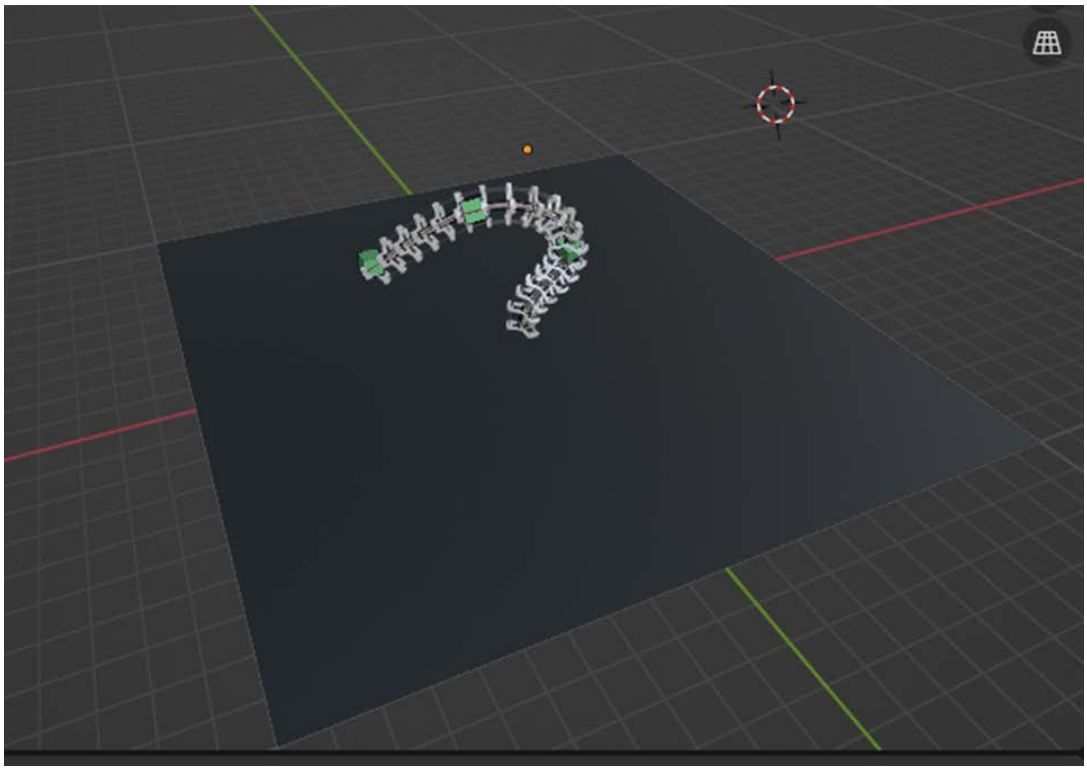


Figure 4.3.3: Image of arm top

The block will display a 2d image modeling the arm in a 3d coordinate plane. The simulation will be updated with new coordinates of the arm's current position and then display the changes by moving the arm around in the simulated environment. The user will be able to drag the arm to a different position. While the user is moving the arm the simulation will not be updated to the current coordinates of the arm to prevent the arm from moving back to its current position before the user finishes moving the arm to their desired location. Once the user has finished moving the arm into the desired position the user will hit S to send the new coordinates to the inverse kinematics control system.

4.3.3 - General Validation

The system needs a simulation in order for the user to visually move the arm into the desired location. Then coordinates will be sent from the simulation to change the physical position of the arm, which should mirror the position the simulated arm was dragged into. In order for the simulated arm to position itself properly so the user can move it to the desired location, the simulator needs the updated coordinate data of the arm's current location. In order for the changes to the arms position entered by the user to take effect the simulator will need to send new coordinates to the inverse kinematic control system to convert them to usable coordinates for the physical arm. The design of the simulator visualization block includes interfaces for user input, current coordinates from the inverse kinematics control system and new coordinates for the arm to move sent back to the inverse kinematics control system.

4.3.4 - Interface Validation

otsd_smltr_vslztn__usrin : Input

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> meet or exceed each property?**
Timing: user input will only be received after arm has finished moving from previous commands	By limiting when user input will be received there will be less issues when moving the arm	User can only change coordinates in the simulator after the arm stops moving.
Usability: User can drag arm to intended positioning	User can interact with the simulation to change the position of the arm. In this case the user should see where they drag the arm to.	User will be able to drag the arm to a new position then the arm will move to the new position. If the user can't move the arm this property fails.

invrs_knmtc_cntrl_systm_smltr_vslztn__data : Input

Messages: The control system will send updated coordinate data to the simulation using (x,y,z)	The simulation needs to move with the arms current location, which will require updated coordinates to be sent to the simulation.	Input is received by the simulation for the current coordinates of the arm in format starting with X , Y, Then Z
Datarate: control system should send updated coordinates of the arm at least 10 times a second	If the arm simulation is not updated with new coordinates fast enough then the movement of the simulation will be choppy and hard for the user to see how the arm is moving. Additionally if the arm isn't updated the user won't know what position the arm is currently in	Input is received at least 10 times every second and the arms current simulation is updated each time a new coordinate is received.

smltr_vslztn__invrs_knmtc_cntrl_systm_data : Output

Datarate: will send data as soon as user enters information and confirms	User needs to see the arm responds to its input. Sending input right away will increase time for the arm to start moving to a new location	Input is received for the current coordinates of the arm
Messages: Message should contain (x,y,z) coordinates	There needs to be consistent messages sent from simulation to the controller	Sent message will contain values for (x,y,z) in that order

4.3.5 - Verification Plan

Inverse kinematic controller -> display

1. Controller inputs will be simulated using many different points
2. Controller will send coordinates to the simulator
3. Simulator will receive the inputs and store them
4. Simulator will change the points on the arm in the simulation
5. Simulator will update the simulation display with the new coordinates
6. Check that the coordinates are the same as the ones entered
7. Repeat until all coordinates have been simulated

User input -> inverse kinematic controller

1. User will hit 'G' button to start dragging arm
2. Simulator will stop updating with input from the controller
3. User will drag the arm and the simulator will show the arm moving with the user

4. When user is finished they will Left click with mouse to stop dragging
5. User will hit ‘S’ to send coordinates or ‘G’ to re-drag object
6. Simulator will calculate the new coordinates for its current position
7. Simulator will output current coordinates to control system
8. Simulator will then switch back to receiving updated input from the controller
9. Verify that the coordinates outputted are the same as the position of the arm the user entered

4.3.6 - References and File Links

[1] Heer, Scheel, ECE44X: Block Validation(s), Oregon State University, Jan. 2022, https://docs.google.com/document/d/e/2PACX-1vQr1_5Ums-TEmYtEsfE2Che7WDHYhHQG9de33L0gSmAmbFrz3qruWEbAZlcBCEDAmjmCRK6njbntr7i/pub

[2] Porter, Jordan, Blender Simulation Code, Google Drive, Jan. 2022, https://drive.google.com/file/d/1QQLtw3lfajnqOiVgTYPJgQNQ3oLCbo8I/view?usp=sharing

4.3.7 - Revision Table

3/3/2022	Jordan Porter: added block to project document
1/20/2022	Jordan Porter: Updated information with feedback and more specific details
1/7/2022	Jordan Porter: set up and filled out rough draft of interface validation

4.4 - Control System Loop Block Validation

Champion: Cale Hallamasek

4.4.1 - Description

When moving the arm, an accelerometer will be used to identify how the arm is actually moving in physical space compared to the calculated movement instructions. If there is significant error between where the arm is and where it should be, adjustments can be made using a feedback loop to reposition the arm to a more accurate destination. This block represents that feedback loop, and will also interface with the simulator and inverse kinematics system to generate movement instructions. These will be sent to the microcontroller communication block, which will later be polled for newer accelerometer data so that the loop can continue. When a certain threshold or acceptable margin of error is reached, the loop will terminate until new input is provided.

A software PID algorithm [1] will be implemented for the control loop. Tuning can be done during testing, although a reasonable range of values can be defined beforehand.

4.4.2 - Design

Black Box Diagram

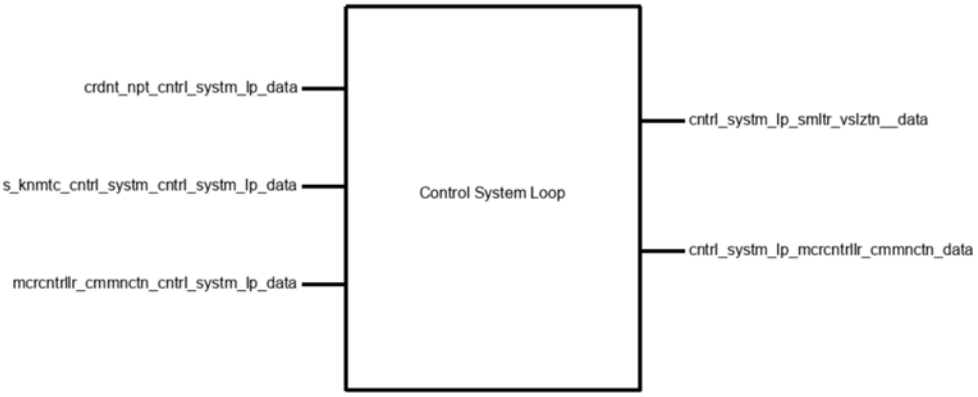


Figure 4.4.1: Block diagram

Pseudocode

```
// something will need to stop the arm if it doesn't reach its destination in time,
to prevent it from moving forever
TIMEOUT_MS = 5000
// when the arm reaches its target position within this threshold, it will cease
movement
GOAL_POSITION_THRESHOLD_MM = 6
CLEAR_LOOP = False
on_update_from_ui():
  CLEAR_LOOP = True
  set target_position from UI
control_loop():
  if CLEAR_LOOP is True, reset values and return it to False
  else:
    current_position = read_encoders()
    target_delta = compare(current_position, target_position)
    if target_delta < GOAL_POSITION_THRESHOLD_MM:
```

```
    exit loop
movement_speed = get_speed_from_delta(target_delta)
adjust target_delta through software PID algorithm
movement_set = get_inverse_kinematics_set(target_delta, movement_speed)
send_movement_to_atmega(movement_set)
```

4.4.3 - General Validation

This block's design allows it to have one main function (generating a target position to be communicated to the microcontroller) that is not overly complex. Instead of computing the inverse kinematics for each motor and handling the microcontroller communication directly, it can act as a simple feedback loop that obtains data and suggests corrections. This makes the block much easier to test, which is very important for control loops, which can be approached and tuned in many different ways.

A big part of this block's design is interfacing with various other blocks in order to compute all of the movement information. This block can be easily integrated with the UI, as both the coordinate input and calculations can run in their own threads in the same program. The block will also request calculations from the inverse kinematics and visualization, and process them accordingly. Handling microcontroller communication is the primary output of this block because it delivers the necessary control information to another part of the device.

4.4.4 - Interface Validation

crdnt_npt_cntrl_systm_lp_data : Input

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
Desired coordinates: three floating-point values (in any of three coordinate systems: spherical, cartesian, cylindrical)	The user will enter their desired arm position as coordinates in the UI, as well as their desired coordinate system. The coordinates then will be translated to the project's standard coordinate system, and sent to this block.	Having a standardized coordinate system will make calculations easier. The user input will be used as a starting point to trigger the loop, or restart it if the arm is already trying to reposition itself to somewhere specified earlier.

cntrl_systm_lp_smltr_vslztn__data : Output

Desired coordinates: three floating-point values (in	The simulator block, which is also responsible for	Updating the simulator visualization in a loop allows

simulator's coordinate system)	calculating inverse kinematics, will receive the control system's target position and update accordingly.	the user to identify where the arm is intending to move, and also allows it to calculate inverse kinematic instructions for these values that will be requested later.
Movement speed: floating point value between 0 and 1	Depending on how far the target position is from the current position, a speed value can be calculated for the arm's movement.	Getting a value that is lower when the arm is closer to the target is relatively straightforward, although it will require some fine tuning and testing.

cntrl_systm_lp_mcrcntrllr_cmmnctn_data : Output

Set of inverse kinematic instructions: list of three-value pairs: - motor ID - motor speed - motor time	These values will need to be given constraints according to the motor's specifications, but they represent the output obtained from the inverse kinematic calculations requested by this block. This output will be sent to the microcontroller, as can be seen in the following interface.	This allows the block to request inverse kinematics calculations based on the arm's current positional accuracy, and then deliver them to the microcontroller.

invrs_knmtc_cntrl_systm_cntrl_systm_lp_data : Input

Set of inverse kinematic instructions: list of three-value pairs: - motor ID - motor speed - motor time	The same values as in the previous interface, although they will first need to be used as input before they can be sent to the microcontroller.	This fulfills one of the block's purposes, which is being an intermediary between the inverse kinematics and the microcontroller.

mcrcntrllr_cmmnctn_cntrl_systm_lp_data : Input

Motor encoder data: format currently unknown	Data obtained from the motor encoders will be used for measuring the arm's	The block's ability to correct for error requires some type of input that tells it about the

	position, and making minor adjustments if the position is outside of the threshold.	state of the arm's position, which in this case is obtained through the encoders.

4.4.5 - Verification Plan

1.) Test the interface between this block and the simulator visualization by passing in coordinates, and retrieving inverse kinematics instructions
2.) Test the interface between the UI and this block by passing UI values to the control loop
3.) Create and test a PID algorithm using randomized values for "offsets"
4.) Add and test variables to the PID algorithm that are modular, so that fine tuning later is easier
5.) Determine motor movement speed, and create an equation for obtaining speed based on how far the motor is from its target
6.) Re-evaluate a threshold for arm accuracy based on past requirements, and implement it in the code

4.4.6. References and File Links

[1] J. Schaenzle, "Introduction to the PID control algorithm," *Atomic Spin*, 30-Dec-2017. [Online]. Available: https://spin.atomicobject.com/2016/06/28/intro-pid-control/. [Accessed: 05-Feb-2022].

4.4.7 - Revision Table

2/18/2022	Cale: Finalized document for submission.
2/13/2022	Cale: Added to general validation.
2/4/2022	Cale: Created document from template.

4.5 - Coordinate Input Block Validation

Champion: Cale Hallamasek

4.5.1 - Description

This block represents a graphical user interface (GUI) which will allow the user to input coordinates for the tentacle arm to move to in one of three formats (spherical, cylindrical, and cartesian). This program will attempt to connect to the arm, and provide status messages about the arm's state, which will be used for debugging and troubleshooting. If the user enters invalid values, the program will give appropriate feedback. Automatic coordinate conversion between

the three options will be performed (if possible). A scrolling text log at the bottom of the interface will provide feedback about previous commands.

For this block's implementation, a prototype is being written in Python, with the Tkinter library being used for creating the window and adding elements to the interface. Python allows for quick prototyping due to its interpreted and flexible nature. In the future, this program can be rewritten in C++ with the Qt library for a more portable, smaller application.

4.5.2 - Design

Black-box Diagram:

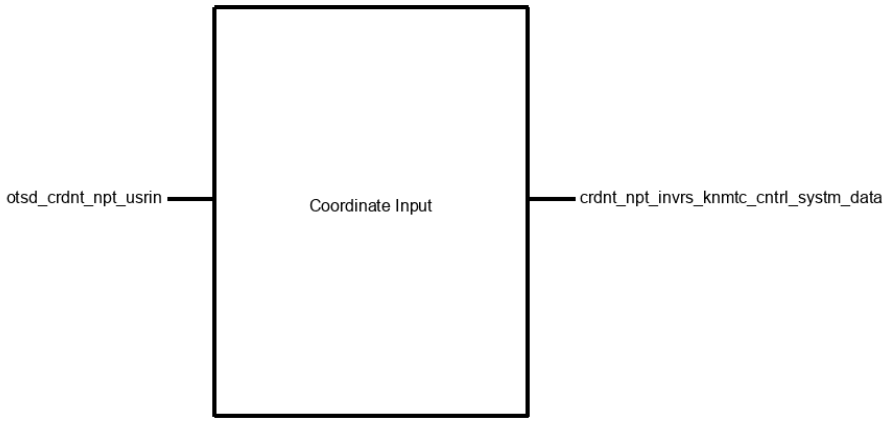


Figure 4.5.1: Black box diagram

Interface:

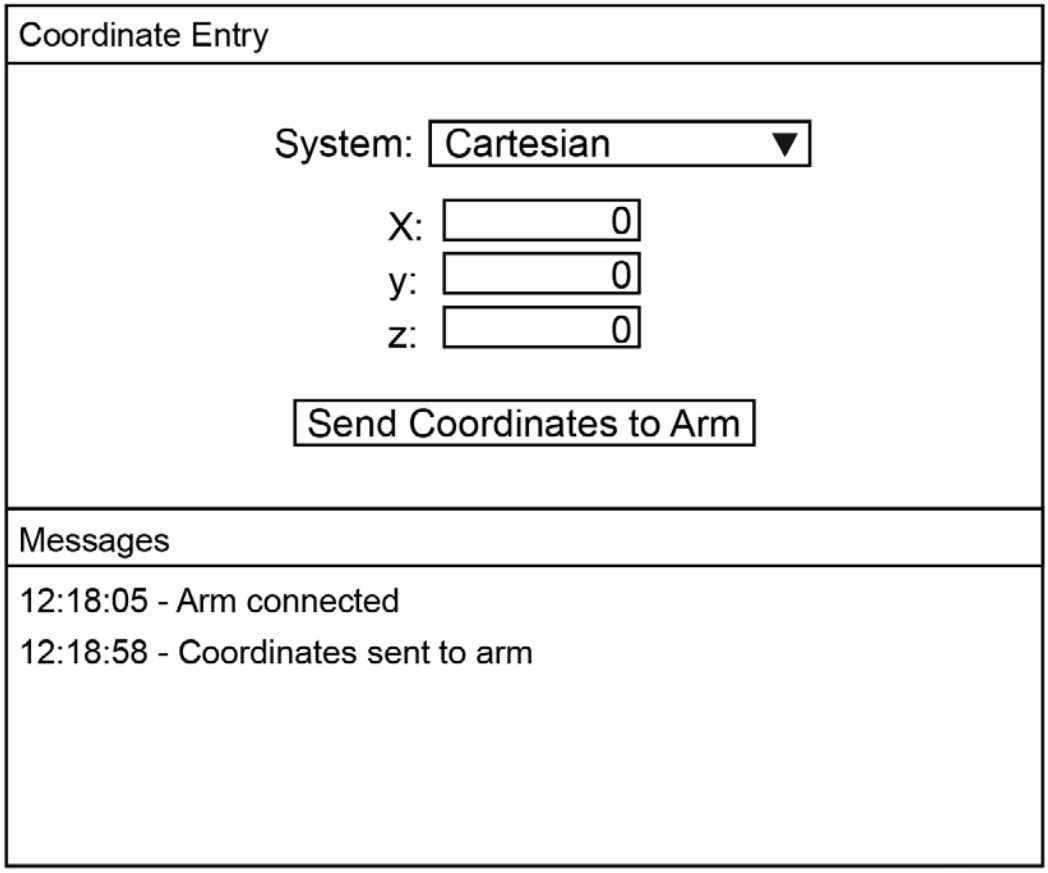


Figure 4.5.2: Screenshot of coordinate input user interface

The interface is designed to be relatively straightforward. The parts that can be interacted with are in the upper portion of the interface. The message box in the bottom section will provide feedback to the user, such as if the arm is connected or disconnected, and if instructions have been properly sent.

Pseudocode / Application state:

```
// Constants/values (at the beginning of the program):
const enum coord_input_type = {
  SPHERICAL, CARTESIAN, CYLINDRICAL
};
// Entry point/initialization (in __main__ for Python):
  create_window()
  load_ui();
  poll_for_input();
  exit();
// Translating coordinates
  x, y, z = get_coord(0), get_coord(1), get_coord(2)
```

```
  check system type, then:
    call corresponding library function to convert units
  return (x, y, z)
// Upon receiving user input (in poll_for_input or similar method):
  coord_input_type system = get_type_from_ui();
  coords = get_and_translate_coords_to(system);
  if (not !valid_coords(coords, system))
    error_log("Please enter values within the range [ 5, -5]");
  else
    send_coords_to_inverse_kinematics();
```

4.5.3 - General Validation

This design fits the system because the arm needs some way to be controlled from the user's computer. One way that this could be done is through a command-line interface, but this is not the most user-friendly option. A cross-platform graphical interface allows the user to easily understand what data they should enter to control the arm (provided that the interface is well-labeled). The purpose of the feedback messages is to inform the user of the arm's current state, and to assist in troubleshooting, if necessary.

To improve the arm's versatility from a software standpoint, three types of coordinate systems are supported. While only one coordinate system is necessary to move the arm, multiple options allow for users familiar with other systems to operate the arm with ease. Certain applications may also be better suited to one coordinate system than the others. It is also a relatively straightforward feature, so it will not cost excess development time or resources.

This block will be relatively easy to implement because of the chosen programming language and libraries (Python and Tkinter). Because this is just a single interface and not a high-performance application (such as the simulator), Python allows the developer to create a fully-functional interface very quickly. A rewrite in C++ later would not impact functionality or be needed to progress, but would make the application smaller and more efficient with a bit of extra effort.

4.5.4 - Interface Validation

otds_crdnt_npt_usrin: Input

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Three coordinates - array of three numbers	There will be three fields where x, y, and z (or other unit type) values can be entered by the user. These need to be stored as numbers so that they can be	Keeping the input as a set of numbers ensures that the correct number of values is stored, and that they will be able to be used by the software later.

	used mathematically.	
Coordinate system - enums for Cartesian, Cylindrical, and Spherical)	Enums are the best way to represent a set of abstract states or values in programming, so having an enum to represent the chosen coordinate system makes the most sense.	Enums are supported in Python, and using a modern IDE can ensure that the coordinate system's value does not change to something invalid.
Send input to arm command - Button	This value is a button because it gives the user a way to validate and send their desired positional values to the arm without needing a programmatic solution.	Buttons can be created easily in Tkinter, the UI library, and are easy to connect to the software's backend.

crdnt_npt_invrs_knmtc_cntrl_systm_data : Output

Coordinate values - set of floating point numbers	This was chosen so that the values could be sent directly in a numeric format to the microprocessor	Our block does not output values in a way that is overly complex (JSON or other serialized format), or as a string, which would require extra computation for the microprocessor
Status messages - strings generated from microprocessor's status codes	Displaying messages about the arm's current state (calculating, moving, disconnected, etc.) was chosen in order to make troubleshooting and debugging easier	The ability to print out feedback messages is relatively straightforward, and would likely be implemented for internal purposes anyways
Log file - text file generated by program upon each run, with a timestamped filename	The program should leave a log so that troubleshooting does not require the program to remain on, and so error messages are easier to share	Our block can easily implement this by adding something to our UI message code that writes a line of text to a continually-open file. The file should be time stamped so that it is not possible to override an existing log.

4.5.5 - Verification Plan

To verify this block, the following steps should be taken:

1.) Install the latest Python interpreter

2.) In the folder with the Python script, run setup.bat (Windows) or setup.sh (macOS/Linux) for first-time setup
    a.) This will install the Tkinter UI library and create configuration files (if necessary)
3.) Execute the main file and test every UI component (buttons, text fields, dropdowns) to ensure that the application is usable and that the UI library is compatible with the OS
4.) From *otds_crdnt_npt_usrin*, obtain numeric values from the application's UI for the x, y, and z coordinates. Additionally, obtain the selected coordinate system, and convert the coordinates to the arm's native choice.
5.) Send the coordinates to the arm, and communicate with the user through *crdnt_npt_invrs_knmtc_cntrl_systm_data*, updating the status message log. Ensure that the status updates are properly corresponding to the arm's state.
6.) Close the application and ensure that everything shuts down correctly (no background processes, leftover temporary files, open connections).

4.5.6 - References

[1] G. Strang, "12.7: Cylindrical and spherical coordinates," *Mathematics LibreTexts*, 02-Jan-2021. [Online]. Available: https://math.libretexts.org/Bookshelves/Calculus/Book%3A_Calculus_(OpenStax)/12%3A_Vectors_in_Space/12.7%3A_Cylindrical_and_Spherical_Coordinates#:~:text=To%20convert%20a%20point%20from,and%20z%3D%CF%81cos%CF%86. [Accessed: 03-Mar-2022].

[2] "Tkinter - Python interface to TCL/TK¶," *tkinter - Python interface to Tcl/Tk - Python 3.10.2 documentation*. [Online]. Available: https://docs.python.org/3/library/tkinter.html. [Accessed: 03-Mar-2022].

4.5.7 - Revision Table

1/21/2022	Cale: Included more content, implemented Cale: feedback changes from peer review.
1/8/2022	Cale: Created document from template.

4.6 - Power Supply and Distribution Block Validation

Champion: Ben Chan

4.6.1 - Description

This block validation is for the power supply and power distribution in the tentacle robot system. The tentacle system contains 9 sets of motors, motor drivers, and encoders. In addition, the system will also make use of an ATMega128 microcontroller board and an FPGA. The power block supplies a constant 5V voltage at low current to each of the 9 encoders. Additionally, the power supply will filter the input 12V and supply the constant filtered 12V to the motor drivers. The microcontroller will be powered separately via USB from the computer application, and does not need to be considered in this block.

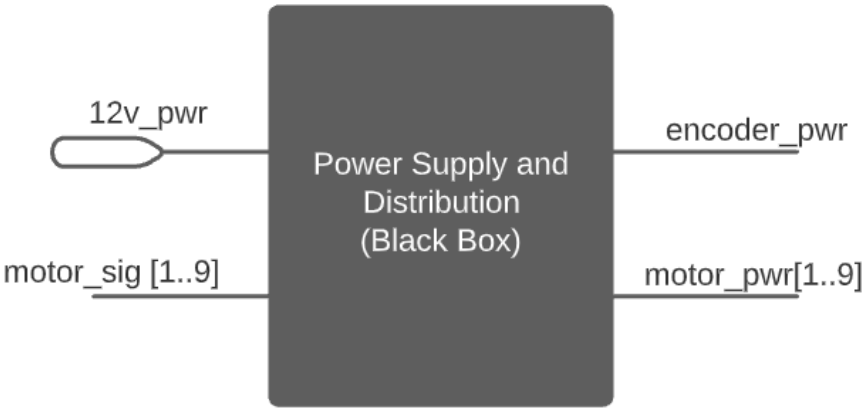


Figure 4.6.1: Mechanical systems, black box diagram.

4.6.2 - Design

This section will cover the design components of the power supply and distribution system. This encompasses the expanded block diagram, design drafts, relevant datasheet information, circuit schematics, and dimensions.

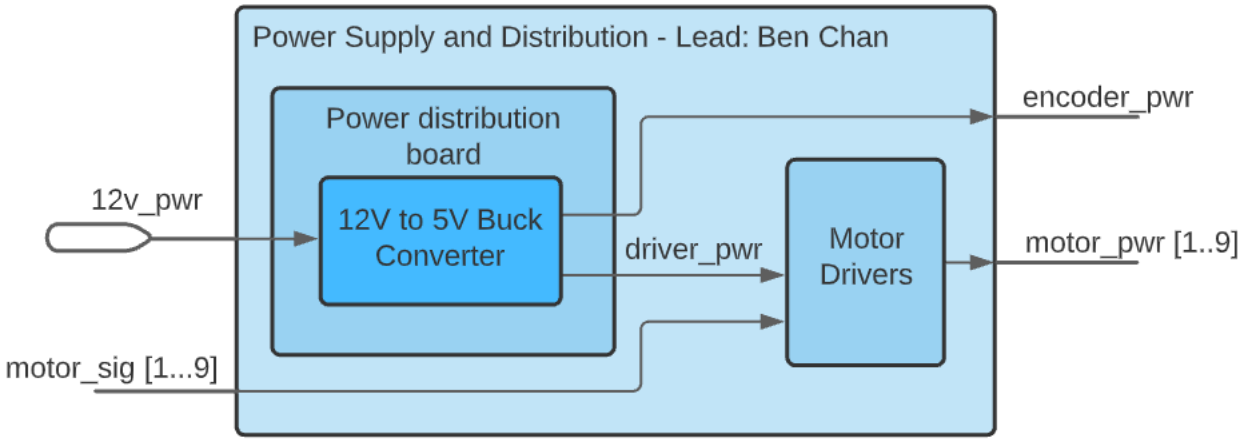


Figure 4.6.2: Power supply and distribution, expanded block diagram.

Figure 2.1 shows the expanded block diagram of the power supply system. The internal functionality has been displayed.

The first function is the power distribution board. This board consists of a DC-DC 12V to 5V buck converter, a low-pass filter, and an array of pin headers to organize and distribute the required power to each of the components in the system. The example power converter circuit included in the MC33063 datasheet is shown in figure 2.2 below.

Two pins are used for the 12V input. 9 of the pin headers included on the board output a filtered 12V; this is used for motor drivers. 10 of the pin headers are used as the buck converter's 5V output; these are used to power the encoders in the tentacle arm system, as well as the FPGA. The remaining. 19 pins serve as ground pins for each of the 5V and 12V outputs. In addition to the pin headers, a barrel jack has been included as an alternate input terminal. The barrel jack

has been hooked up such that when a jack is inserted, it cuts off the two input voltage pins. This allows the system to use alternate power sources such as batteries [1].

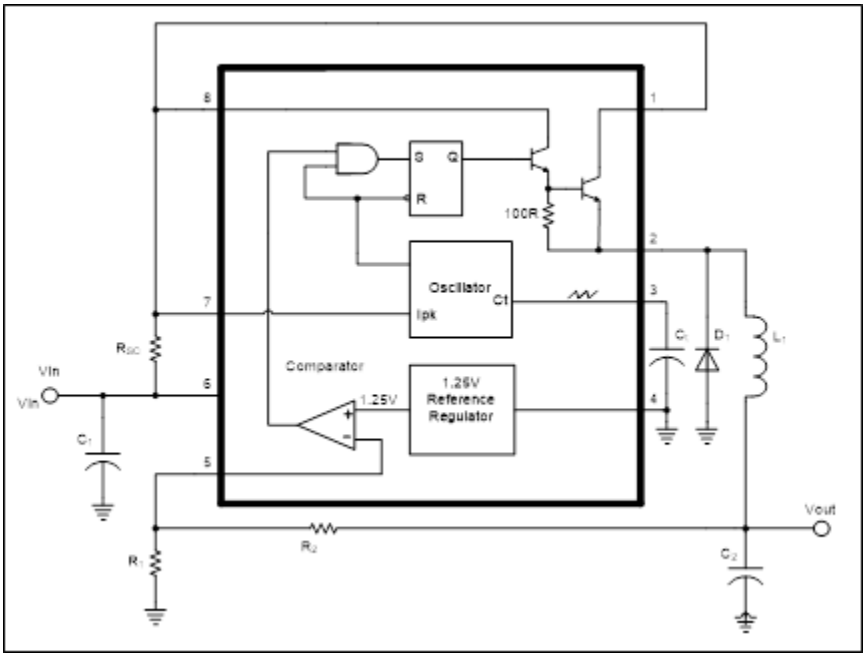


Figure 4.6.3: MC33063 Datasheet: power converter application.

The second function is the motor drivers. The 12V output of the board is used to power 9 motor drivers. Each of the drivers takes a 0-5V logical input from the ATMega128, and outputs a 0-6V output at the motors required current. The motor drivers are used since the ATMega128 has a maximum output current of 40.0mA per pin [2], and thus cannot drive the high-torque motors (under load) used in the system.

Figure 2.3 below is the schematic of the system. On the top of the figure is the buck-converter, and on the bottom is the array of pin headers and the barrel jack input.

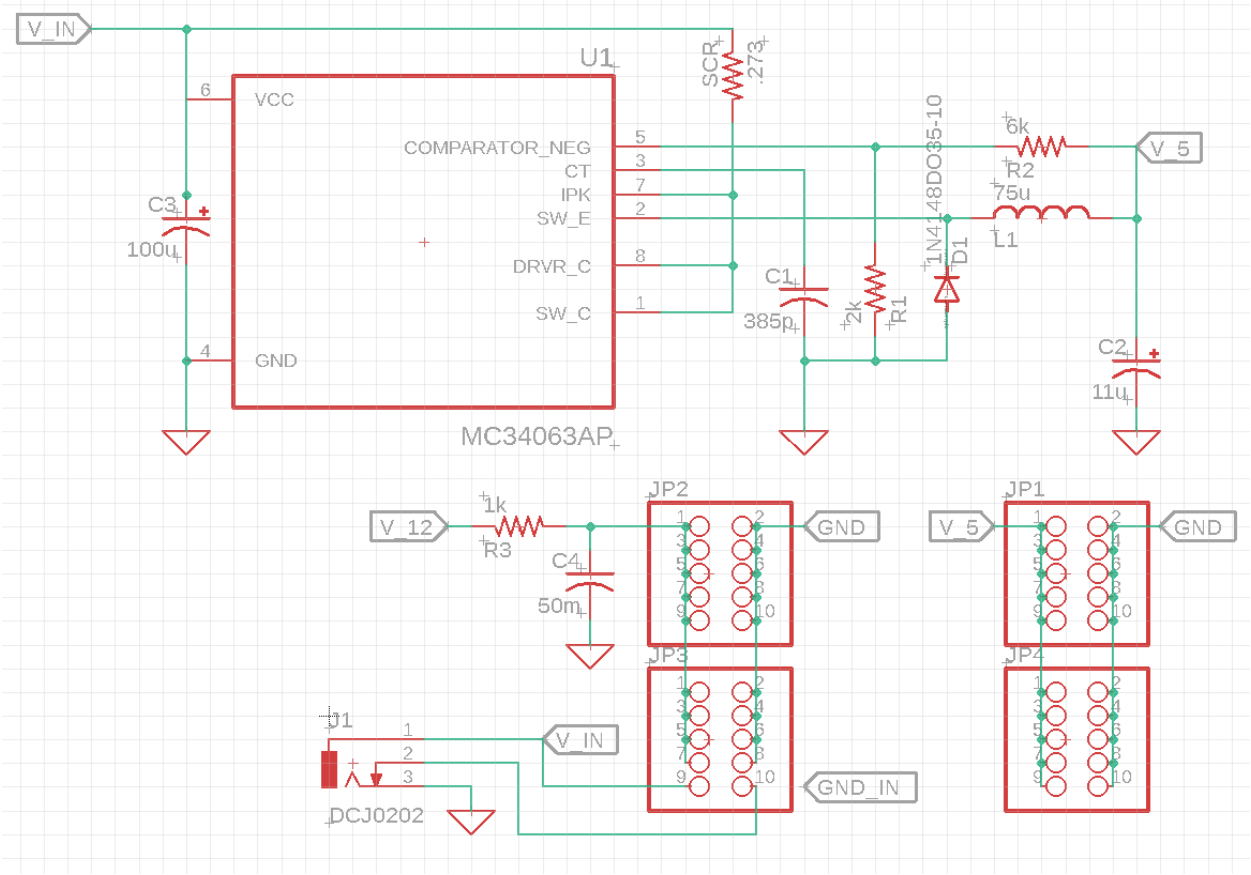


Figure 4.6.4: Autodesk Eagle schematic. 12-5 DC Buck converter and pin headers.

4.6.3 - General Validation

Component Selection

Component selection was based on the equations provided in the MC34063AP datasheet [3], the desired input and output values, and the provided buck-converter example schematic. The datasheet equations used are contained in figure 3.1 below. The desired input voltage is 12V DC, the output voltage is 5V DC with 5% voltage ripple, and the output current is 1.0A with 20% maximum current ripple.

The first component selection to be made was the diode. The 1N5819 diode was selected due to its availability and low cost. Examining the datasheet for the 1N5819 diode [4], it can be determined that at 1A instantaneous forward current, it draws a forward voltage of 0.60V.

Calculation	Step−Up	Step−Down	Voltage−Inverting
t_{on}/t_{off}	$\frac{V_{out} + V_F - V_{in(min)}}{V_{in(min)} - V_{sat}}$	$\frac{V_{out} + V_F}{V_{in(min)} - V_{sat} - V_{out}}$	$\frac{\lvert V_{out}\rvert + V_F}{V_{in} - V_{sat}}$
$(t_{on} + t_{off})$	$\frac{1}{f}$	$\frac{1}{f}$	$\frac{1}{f}$
t_{off}	$\frac{t_{on} + t_{off}}{\frac{t_{on}}{t_{off}} + 1}$	$\frac{t_{on} + t_{off}}{\frac{t_{on}}{t_{off}} + 1}$	$\frac{t_{on} + t_{off}}{\frac{t_{on}}{t_{off}} + 1}$
t_{on}	$(t_{on} + t_{off}) - t_{off}$	$(t_{on} + t_{off}) - t_{off}$	$(t_{on} + t_{off}) - t_{off}$
C_T	$4.0 \times 10^{-5}\, t_{on}$	$4.0 \times 10^{-5}\, t_{on}$	$4.0 \times 10^{-5}\, t_{on}$
$I_{pk(switch)}$	$2I_{out(max)}\left(\frac{t_{on}}{t_{off}} + 1\right)$	$2I_{out(max)}$	$2I_{out(max)}\left(\frac{t_{on}}{t_{off}} + 1\right)$
R_{sc}	$0.3/I_{pk(switch)}$	$0.3/I_{pk(switch)}$	$0.3/I_{pk(switch)}$
$L_{(min)}$	$\left(\frac{(V_{in(min)} - V_{sat})}{I_{pk(switch)}}\right) t_{on(max)}$	$\left(\frac{(V_{in(min)} - V_{sat} - V_{out})}{I_{pk(switch)}}\right) t_{on(max)}$	$\left(\frac{(V_{in(min)} - V_{sat})}{I_{pk(switch)}}\right) t_{on(max)}$
C_O	$9\,\frac{I_{out} t_{on}}{V_{ripple(pp)}}$	$\frac{I_{pk(switch)}(t_{on} + t_{off})}{8V_{ripple(pp)}}$	$9\,\frac{I_{out} t_{on}}{V_{ripple(pp)}}$

V_{sat} = Saturation voltage of the output switch.
V_F = Forward voltage drop of the output rectifier.

The following power supply characteristics must be chosen:

V_{in} − Nominal input voltage.
V_{out} − Desired output voltage, $|V_{out}| = 1.25\left(1 + \frac{R2}{R1}\right)$
I_{out} − Desired output current.
f_{min} − Minimum desired output switching frequency at the selected values of V_{in} and I_O.
$V_{ripple(pp)}$ − Desired peak−to−peak output ripple voltage. In practice, the calculated capacitor value will need to be increased due to its equivalent series resistance and board layout. The ripple voltage should be kept to a low value since it will directly affect the line and load regulation.

NOTE: For further information refer to Application Note AN920A/D and AN954/D.

Figure 4.6.5: Equation table for the MC34063AP, taken from component datasheet.

Since the inductor is in series with the output, the peak inductor current is equal to the output current. $I_{L,AVG}$ 1.0A. The ripple current, the frequency, and the duty cycle are determined using the desired values for ripple and voltage.

$$I_{peak} = I_{L,avg} + I_{ripple}/2 = 1.0 + (1.0 \times 20\%)/2 = 1.1A$$
$$f = 50kHz,\ T = 1/f = 20\mu S$$

$$D = t_{on}/t_{off} = (V_{out} + V_{f,diode})/(V_{in} - V_{sat} - V_{out})$$
$$D = (5V + 0.6V)/(12V - 1.0V - 5.0V) = 0.93$$

$$t_{off} = T / (1 + D) = 20\mu S / (1 + 0.93) = 10.36\mu S$$
$$t_{on} = T - t_{off} = 20\mu S - 10.36\mu S = 9.64\mu S$$

Next, the short circuit resistor's value was calculated.

$$R_{sc} = 0.3/I_{peak} = 0.3/1.1A = 0.273\Omega$$

Once the duty and ripple values are determined, the values for the timing capacitor, output filter capacitor, and the buck converter inductor can be decided. Note that the equations used determine the minimum capacitor and inductor values. A higher capacitance or inductance can be used to further reduce the ripple and provide a safety factor.

$$C_t = 4 \times 10^{-5} \times t_{off} = 4 \times 10^{-5} \times 10.36\mu S = 385.4pF$$
$$C_{out} = (I_{peak} \times T)/(8 \times V_{ripple}) = (1.1A \times 20\,385\mu S)/(8 \times 0.25V) = 11\mu F$$
$$L_{min} = t_{on} \times (V_{min} - V_{sat} - V_{out})/I_{peak}$$
$$L_{min} = 9.64\mu S \times (12V - 1.0V - 5.0V)/1.1A = 51.1\mu H$$

Lastly, a voltage divider is used on the MC34063AP chip's output to bring the voltage back up to the desired 5V. The equation from the datasheet can be solved for the ratio R_2/R_1. To determine the value of R_2, a resistance of 2kΩ is chosen for R_1.

$$V_{out} = 1.25(1 + R_2/R_1)$$
$$R_2/R_1 = V_{out}/1.25 - 1 = 3$$
$$R_2 = R_1 \times 3 = 2k\Omega \times 3 = 6k\Omega$$

A spreadsheet was used to determine the power dissipated by each of the selected components. Using that value, the efficiency of the system can be determined by comparing the power in and out of the system.

$$P_{in} = P_{out} + P_{loss} = 5.0W + 1.29W = 6.29W$$
$$Efficiency = P_{out}/P_{in} = 5.0W / 6.29W = 79.5\%$$

Input Interface Validation

Inputs to the system are the 12_pwr interface, and the motor_sig[1..9] interface.

12v_pwr is a 12V input from an external power supply. The chip used in the buck converter can handle a range of 3.0V to 40V before failure. Given the determined input power of 6.29W above, the input current of the converter buck converter should draw no more than 0.53A (6.29W / 12V). The motor drivers used each have a maximum current draw of 3.0A, and a rated draw of 2.0A [5]. Since each motor driver can control two motors, a total of 5 drivers are needed, with a maximum instantaneous current draw of 15.0A, and an estimated nominal of 3.0A (Assuming no more than three motors experience max torque load at a single moment).

motor_sig[1..9] is the logical signal received from the microcontroller. The motor_sig bus contains 9 signals, each of which will connect to a separate motor driver. The rated logical voltage of each driver is 5.0V, and the logical current is 0.36A [5]. The ATMega128 board is able to output a 0-0.4A current and 0-5.0V voltage per output pin, which satisfies both of the requirements for this interface.

4.6.4 - Interface Validation

Interface Property	Why does this interface have this value?	Why do you know that the design details for this block meet or exceed each property?
12v_pwr		
V_max = 13.5V	This is the peak expected voltage variation of 15%	This is the input power source to the buck converter and motor drivers. The buck converter chip is the MC34063AP, which accepts a range of 3.0 to 40.0V. The motor drivers accept a drive voltage range of 5.0 to 35.0V.
V_min = 10.5V	This is the peak expected voltage variation of -15%.	
V_nominal = 12.0V	This is the expected 12.0V input to the 12V-5V buck converter. The buck converter was designed with this value in mind.	
motor_sig [1..9]		
I_peak = 0.4A	This is the logical current maximum for the motor drivers.	The ATMega128 board is able to output a 0-0.4A current along any of its output pins.
V_max = 5.0V	This is the logical voltage maximum for the motor drivers.	The ATMega128 board is able to output 0 to 5.0V per output pin.
sensor_pwr [1..9]		
I_peak = 1.0A	The encoders have a current draw of 10mA per encoder. With 9 encoders, the power supply must be able to supply at least 0.1A.	The buck converter was designed with a 1.0A current draw in mind. The converter was also designed with a ripple of 20% (0.9A to 1.1A). In the worst case scenario of all 9 encoders drawing 10mA, the system should still have enough current.
V_max = 5.5V	The encoders are able to run on any voltage with a value of 3.5 to 20.0V. 5.0V was a relatively low power consumption value that was compatible with other voltage in the system.	The buck converter outputs 5.0V with a ripple of 5%. 4.5V and 5.5V are outside of the expected ripple range.
V_min = 4.5V		

motor_pwr [1..9]

V_min = -6.0V	This is the output signal to the motors. A DC voltage value between -6.0 and 6.0 volts is determined in the motor driver and sent to the motors.	The motors are rated for continuous operation at up to 6V DC. The rated current of the motors is 0.6A, and it's stall current is 3.3A. Continuous operation at 0.5A is within the rated limits, while short spikes of 2.0A will not trigger a 'stall' condition. The motor drivers have a max continuous current output of 2.0A, and are able to output 5.0V to 35.0V as directed by the logical current input.
V_max = 6.0V		
I_nominal = 0.5A		
I_peak = 2.0A		

4.6.5 - Verification Plan

This section is the step-by-step verification process for the power distribution system.

1. Buck Converter Verification
    a. Connect system input headers to a constant voltage power supply.
    b. Connect an oscilloscope to output headers.
    c. While measuring the output voltage and output current, vary the input voltage between 10.5V and 13.5V
        i. If the system's output voltage drops below a value of 4.5V or exceeds a value of 5.5V, it fails.
        ii. If the power supply's current draw exceeds a value of
2. Motor Driver Verification
    a. Connect system input headers to a constant voltage power supply.
    b. Connect an oscilloscope to the output headers of any single motor driver.
    c. While measuring the output voltage of the motor driver on the oscilloscope, use the 5V output of the buck converter and a potentiometer (voltage divider set-up) to vary the voltage on the logical input of the motor driver.
        i. The oscilloscope should display a change in the motor driver voltage.

    d. Connect a single motor driver to a constant current power supply.
    e. Connect an oscilloscope to the output headers of any single motor driver.
    f. While measuring the output voltage of the motor driver on the oscilloscope, use the constant current power supply to vary the logical input of the motor driver.
        i. The oscilloscope should display a change in the motor driver voltage proportional to the logical current.

4.6.6 - References

[1]. Macmade, DC Barrel + Battery schematic. https://electronics.stackexchange.com/questions/193847/dc-barrel-battery-schematic (Accessed Feb. 4, 2022)

[2]. Atmel, 8-bit Atmel Microcontroller with 128KBytes In-System Programmable Flash, https://ww1.microchip.com/downloads/en/DeviceDoc/doc2467.pdf (Accessed Feb. 4, 2022)

[3]. Onsemi, Inverting Regulator - Buck, Boost, Switching, 1.5A. https://www.onsemi.com/pdf/datasheet/mc34063a-d.pdf (Accessed Feb. 3, 2022)

[4]. Vishay, 1N5817, 1N5818, 1N5819 Schottky Barrier Plastic Rectifier. https://www.vishay.com/docs/88525/1n5817.pdf (Accessed Feb. 3)

[5]. Aideepen, L298N Dual H Bridge DC Stepper Motor Drive Controller Board Module. https://www.amazon.com/Aideepen-Driver-H-Bridge-Replace-Stepper/dp/B07L892P54?th=1 (Accessed Feb. 3, 2022)

4.6.7 - Revision Table

2/16/2022	Peer review feedback changes. Citation formatting
2/4/2022	Citation formatting, and final review of draft.
2/1/2022	Created initial document. Initial formatting.

4.7 ATMega128 and Accelerometer Block Validation

Champion: Triet Nguyen

4.7.1 - Description

This block includes the schematic and code design for the ATMega128 microcontroller that is capable of controlling the tentacle arm. The ATMega128 will be powered by a 3.3V USB connection to a laptop. This microcontroller will communicate through I/O ports with nine DC motors mounted on the tentacle arm to control its movement. The board will receive coordinate inputs to move the motors to their required positions. In return, these motors will also send signals back to the microcontroller through their built-in encoders, the microcontroller will determine which motor to communicate with using its on-board SPI interrupts. The positions of these motors will be constantly updated to a computer to verify the accuracy of the arm.

4.7.2 - Design

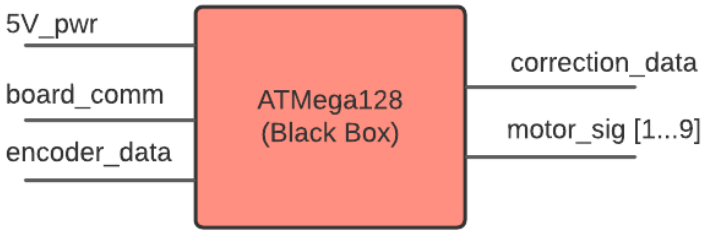


Figure 4.7.1: Black box design

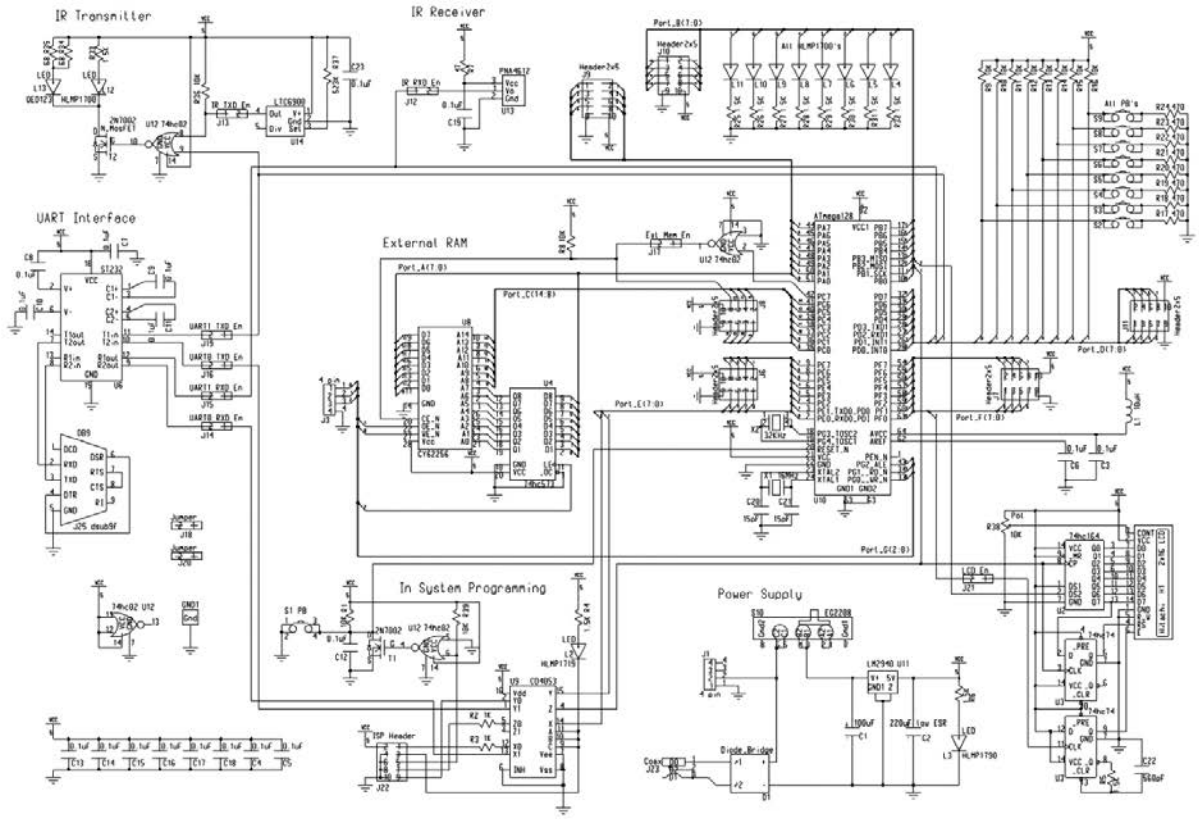


Figure 4.7.2: AVR board schematic

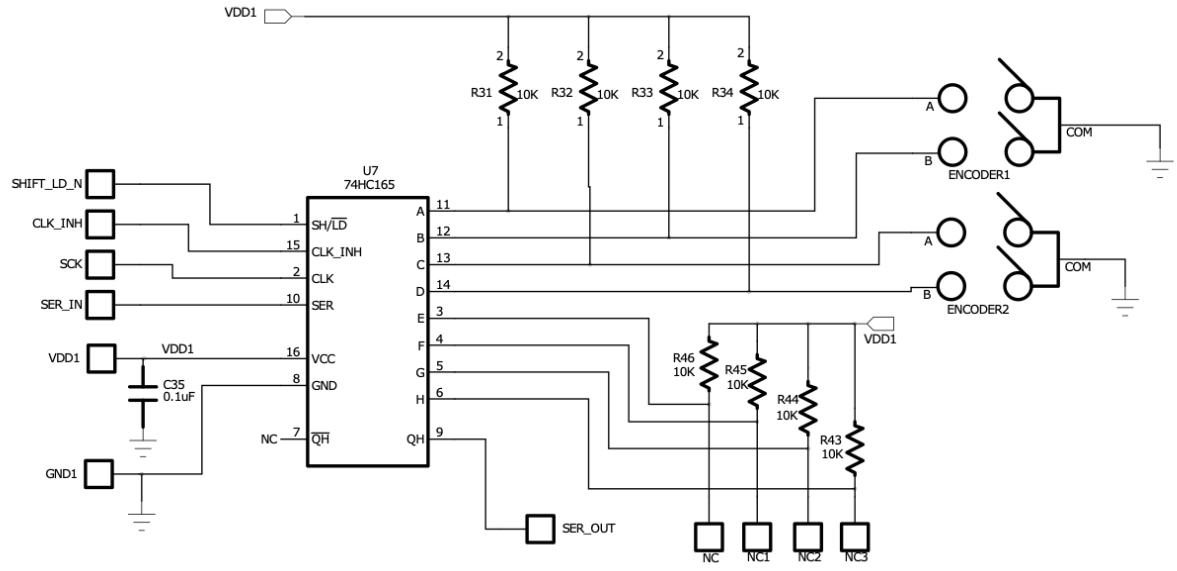


Figure 4.7.3: Encoders

Pseudocode:

```
// Tentacle Arm
```

```
#define DP    0b01111111    // PM

#define SNOOZE_TIME 10

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <string.h>
#include <stdlib.h>
#include "uart_functions.h"

uint8_t  count  = 0x00;
uint16_t sum    = 0x0000;

//holds data to be sent to the segments. logic zero turns segment on
uint8_t segment_data[5];

// Encoder status
uint8_t old_A[2];
uint8_t old_B[2];

// Display
uint8_t minutes = 0; // Clock minutes
uint8_t hours   = 0; // Clock hours

//decimal to 7-segment LED display encodings, logic "0" turns on segment
uint8_t dec_to_7seg[12] = {
 ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, COLON, DP
};

/*********************************************************************/
//                            spi_init
//Initializes the SPI port on the mega128. Does not do any further
//external device specific initializations.  Sets up SPI to be:
//master mode, clock=clk/2, cycle half phase, low polarity, MSB first
//interrupts disabled, poll SPIF bit in SPSR to check xmit completion
/*********************************************************************/
void spi_init(void){//MOSI
 DDRB  |=    (1<<PB0)  | (1<<PB2) | (1<<PB1); //Turn on SS, MOSI, SCLK (SRCLK)
 SPCR  |=    (1<<MSTR) | (1<<SPE);            //enable SPI, master mode
 //SPSR  |=    (1<<SPI2X);                     // double speed operation
}//spi_init (Bar Graph)

//*********************************************************************************
//                            spi_read
//  Reads the SPI port.
//*********************************************************************************
uint8_t spi_read(void){
 SPDR = 0x00;                      //"dummy" write
 while(bit_is_clear(SPSR,SPIF)){}  //wait till 8 clock cycles are done
```

```
 return (SPDR);                    //return incoming data from SPDR
}

/*********************************************************************/
//                              tcnt0_init
//Initialize timer/counter0 (TCNT0). TCNT0 is running in async mode
//with external 32khz crystal.  Runs in normal mode with no prescaling.
//Interrupt occurs at overflow 0xFF.
//
void tcnt0_init(void){
 //timer counter 0 setup, running off i/o clock
 TIMSK |= (1<<TOIE0);                       //enable interrupts
 ASSR  |= (1<<AS0);                         //use external oscillator
 TCCR0 |= (0<<CS02) | ((0<<CS01) |1<<CS00);  //normal mode, no prescaler
}//tcnt0_init

//*****************************************************************************
//                                  segment_sum
//takes a 16-bit binary input value and places the appropriate equivalent 4 digit
//BCD segment code in the array segment_data for display.
//array is loaded at exit as:  |digit3|digit2|colon|digit1|digit0|
void segsum(uint16_t hours, uint16_t minutes) {
 //determine how many digits there are
 //break up decimal sum into 4 digit-segments
 for(int i = 0; i < sizeof(segment_data); i++){
   if(i < 2){
       segment_data[i] = dec_to_7seg[minutes % 10];
       minutes /= 10;
   }
   else{
       segment_data[i] = dec_to_7seg[hours % 10];
       hours /= 10;
   }
 }
}//segment_sum

//*********************************************************************
//                        Read encoder
//*********************************************************************/
void encoder_chk(uint8_t h, uint8_t m){
   uint8_t new_A[2], new_B[2];

   PORTE |= 0x80;                    // SH/LD low  (bit 6)
   PORTE &= ~(0x40);                 // CLK_INH high (bit 7)

   _delay_ms(1);

   PORTE |= 0x40;                    // CLK_INH low
   PORTE &= ~(0x80);                 // SH/LD high

   uint8_t data = spi_read();        // Read current encoder state
```

```
//Read new state of encoders (See which one is spinning)
new_B[0] = ((data & 0x01) == 0) ? 0 : 1;
new_A[0] = ((data & 0x02) == 0) ? 0 : 1;
new_B[1] = ((data & 0x04) == 0) ? 0 : 1;
new_A[1] = ((data & 0x08) == 0) ? 0 : 1;

//sum = new_A[0];
int counter = 0;
// int return_val = -1; // default return value , no change
for(int i = 0; i < 2; i++){
    if ((new_A[i] != old_A[i]) || (new_B[i] != old_B[i])){ //if change occurred
    if ((new_A[i] == 0) && (new_B[i] == 0)) {
        if (old_A[i] == 1){
        if(i == 1){h++;}
        else      {m++;} // sum += count;
        }
        else {
        if(i == 1){h--;}
        else      {m--;} // sum -= count;
        }
    }
    else if ((new_A[i] == 0) && (new_B[i] == 1)) {
        if (old_A[i] == 0){
        if(i == 1){h++;}
        else      {m++;}// sum+=count;
        }
        else {
        if(i == 1){h--;}
        else      {m--;}// sum-=count;
        }
    }
    else if ((new_A[i] == 1) && (new_B[i] == 1)) {//detent position
        if (old_A[i] == 0){ if(counter ==  3){sum += count;}} //one direction
        else              { if(counter == -3){sum -= count;}} //or the other
        counter = 0; //count is always reset in detent position
    }
    else if ((new_A[i] == 1) && (new_B[i] == 0)) {
        if (old_A[i] == 1) {
        if(i == 1){h++;}
        else      {m++;}// sum+=count;
        }
        else {
        if(i == 1){h--;}
        else      {m--;}// sum-=count;
        }
    }
    } //if change occurred
    old_A[i] = new_A[i]; //save what are now old values
    old_B[i] = new_B[i];
}
hours = h;
minutes = m;// 1/32768         = 30.517578uS
```

```
}

/*************************************************************************/
//                          timer/counter0 ISR
//When the TCNT0 overflow interrupt occurs, the count_7ms variable is
//incremented. Every 7680 interrupts the minutes counter is incremented.
//TCNT0 interrupts come at 7.8125ms internals.

//(1/32768)*256    = 7.8125ms
//(1/32768)*256*64 = 500mS
/*************************************************************************/
ISR(TIMER0_OVF_vect){

   uint8_t tempA = PORTA;
   uint8_t tempB = PORTB;
   uint8_t DDRA_temp = DDRA;
   uint8_t DDRB_temp = DDRB;

   encoder_chk(hours, minutes);
   segsum(hours,minutes);

   PORTA = tempA;
   PORTB = tempB;
   DDRA = DDRA_temp;
   DDRB = DDRB_temp;
}

//***********************************************************************************
//              MAIN
//***********************************************************************************
int main()
{
 tcnt0_init();
 spi_init();
 sei();

 //set port bits 4-7 B as outputs
 DDRB  |= 0xF0; //Set to all outputs, change back to 4-7
 DDRE  |= 0xFF; //Set PORTE to output
 PORTE |= 0x00; //COM_LVL On
 PORTE |= 0x40; //SH/LD On

 DDRC  |= 0xFF;
 // DDRD  |= 0xFF; //Speaker on
 // DDRD  &= ~(0x20);

 while(1){

   PORTB = 0x00;

   /*********************************************
   //              7 Seg Display Output
```

```
**********************************************/
DDRA = 0xFF;
```

4.7.3 - General Validation

For this block validation, I will test the input from the encoders and display the encoder data on a pre-built LED display. The overall purpose of this block validation is to test for encoder input. In the design of the system, we will be using motors to control the tentacle arm. These motors have built-in encoders, which will function as sensors that will record the rotation status of the motors. The ATmega128 board will receive these inputs to verify the accuracy of the motors' positions. There are many reasons why this method of validation will be beneficial for the system as a whole. The first reason why I volunteered to be responsible for this block is because of prior engineering knowledge from working with the ATmega128 microcontroller and encoders in past projects. Since this project uses a microcontroller as a driver to receive inputs from the motors' encoders, past experience will come in handy.

When it comes down to efficiency, we want to prioritize and use what is already available to us to save money as well as time. For this reason, I am reusing my ATmega128 microcontroller and encoders to verify my design. In the design impact assessment we did from fall term, I mentioned how chip manufacturers are producing their products ethically inappropriate, along with that there are also many impacts of how manufacturing and delivering our electronic parts can potentially affect the environment. By using existing parts, I believe we will reduce our environmental footprint. When looking at the code, all I/O ports are initialized.

The encoder function is the most important function of this code. It will use the SPI interrupt to read the current encoder state, and output the changes being made. Timer/counter 0 (TCNT0) runs in normal mode to detect encoder input as well as outputting the changes to the LED display. The validation method of using the 7-segment LED display is efficient because it shows the recognition of the encoders while our motors are still being delivered. Since each motor will have its own binary input and will also take 5V of power similar to each segment on the LED display, we can use the LED display segments to verify that the right voltage and input to be supplied once the motors are connected.

4.7.4 - Interface Validation

Interface Property	**Why is this interface of this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
atmega128_spi_input: Input		
spi_register:	ATmega128 SPI interrupt	I expect the SPI interrupt to be able to register data from the encoders. The AVR board will be powered by a computer to turn on those interrupts. SPI interrupts will verify the direction that the encoders turn (left/right).

enc_new_data:	Encoder data will be transmitted into the SPI interrupt port of the AVR board	I expect the encoders to be connected by two connections, gnd and power; their signals will be connected to the AVR board to read their inputs. The newly recorded data will be recorded and displayed on an LED display.
enc_data: Output		
enc_new_data	Encoder data will be displayed on an LED display.	This process will allow the user to verify which direction the encoder is turning (left/right).
LED_display	V_max: 5V	The LED display is used to verify that the correct voltage is sent out to the motors which are also powered by a 5V power supply.

4.7.5 - Verification Plan

1. Plug AVR board into a computer
2. Follow encoder connections in code file
3. Verify LED segments turn on
4. Turn encoders left/right
5. Verify encoder's right turn is valid if number increases
6. Verify encoder's left turn is valid if number decreases

4.7.6 - References

[1] "8-bit Atmel microcontroller with 128kbytes in-system ..." [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/doc2467.pdf. [Accessed: 18-Jan-2022].

4.7.7 - Revision Table

01/20/2022	Triet Nguyen: General validation update
01/19/2022	Triet Nguyen: Interface validation update
01/06/2022	Triet Nguyen: General updates

4.8 - Encoder Sensors Block Validation

Champion: Triet Nguyen

4.8.1 - Description

This block involves the encoder sensors that come with the motors that we use to power our tentacle arm. The general idea of why we decide to use motors with encoder sensors is to keep track of the rotation of the motors. The coordinate data will be computed on a software on the

computer to find how many rotations each motor needs to spin, this number will be sent to the microcontroller and in return, the built-in encoder sensors will update the number of rotations it they have rotated back to the microcontroller, once the required rotation is accomplished, the microcontroller will verify with the computer and the new rotation instruction will be sent. This microcontroller will use its interrupts and ports to communicate the information with the encoders. Since the sensors are built into the motors, they are expected to share the power with the motors. This power supply will come from my partner's power supply block. This power is expected to be around 5V.

4.8.2 - Design

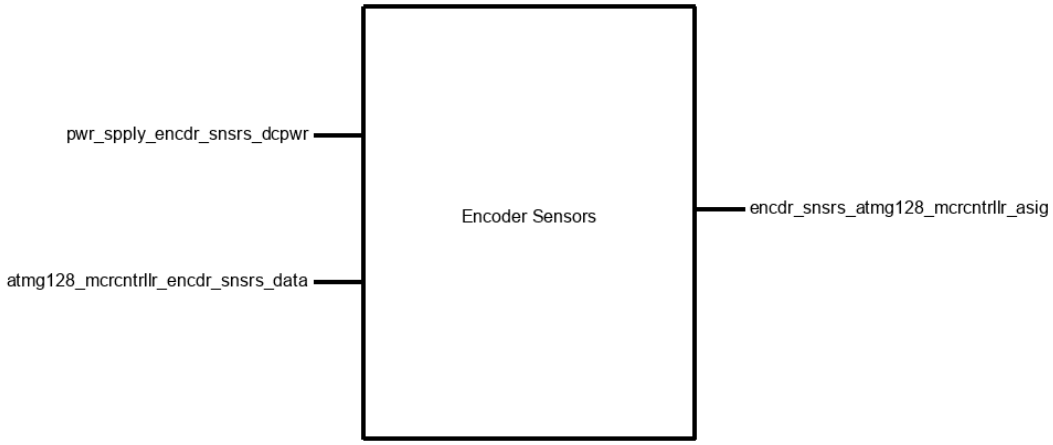


Figure 4.8.1: Black box diagram

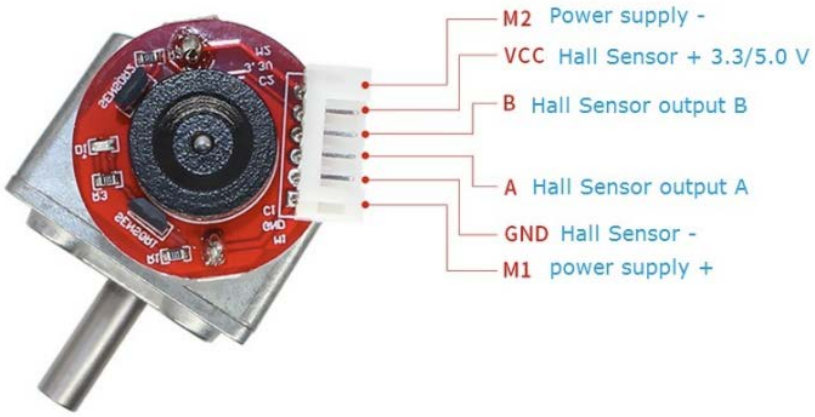


Figure 4.8.2: Motor schematic

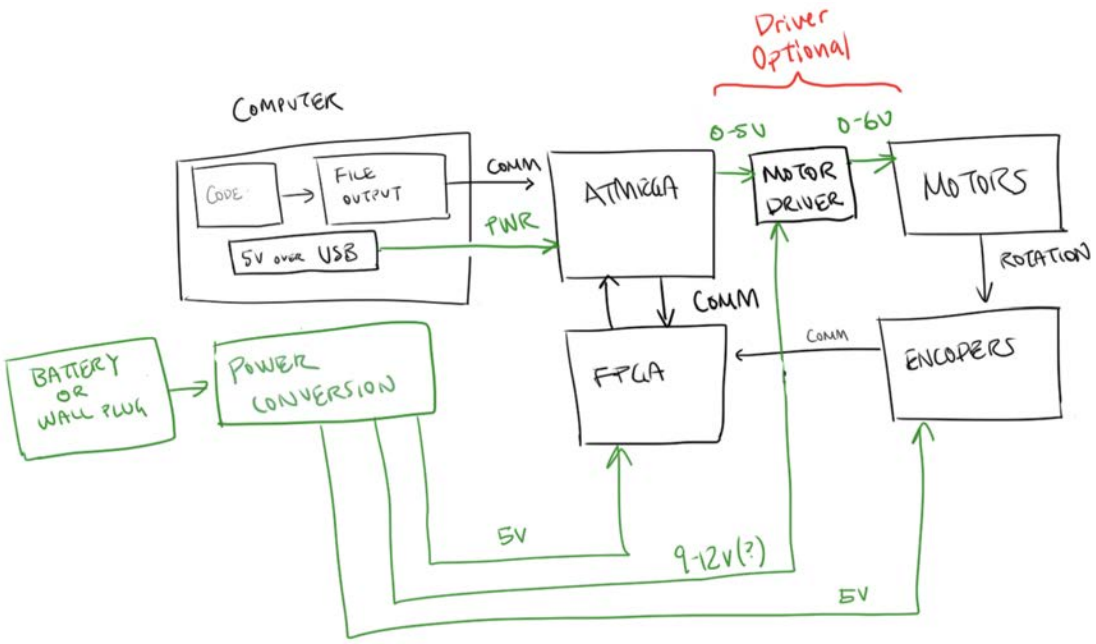


Figure 4.8.3: System Design

4.8.3 - General Validation

For this block validation, I will test the encoder data and verify that the encoders spin up to the required revolution. The overall purpose of this block validation is to test how the motors' encoders and the ATMega128 interact with each other to control the tentacle arm. The ATMega128 board will receive the encoder sensors' data input from a main computer through its USART. The data will contain instructions to either spin the motors left or right, and which motor the instructions will be directed to. Since the encoders are built into the motors, it will share the power supply input with the motors. The appropriate values are listed in the Interface Validation section. This design will ensure the motors move to their required positions. The encoder sensors will be giving feedback data to the microcontroller to verify the required amount of rotations is met. This data will verify that the system is working correctly. The ATMega128 will be designed to record a revolution at every 16 single edge counts done by the encoders.

4.8.4 - Interface Validation

Interface Property	**Why is this interface of this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
pwr_spply_encdr_snsrs_dcpwr: Input		
• **Ipeak:** 1A	Maximum current going to motors	This current is the maximum current that can be drawn by the Hall sensor.

• **Vmax:** 5.5V	Maximum voltage to power motors	Motors will need this maximum voltage in order to rotate to its full torque capacity
• **Vmin:** 4.5V	Minimum voltage to power motors	Motors will need this minimum voltage in order to rotate to its minimum torque capacity.
atmg128_mcrcntrllr_encdrs_snsrs_data: Output		
• **Messages: Left Rotation**	Instruction for encoders to spin left	This instruction comes from the software block that instructs the encoders to spin left.
• **Messages: Right Rotation**	Instruction for encoders to spin right	This instruction comes from the software block that instructs the encoders to spin right.
• **Messages: Motor**	Instruction for motors	This instruction instructs which motor to spin
encdr_snsrs_atmg128_mcrcntrllr_asig: Input		
• **Vmax: 5V**	Maximum voltage output of Hall sensor	The Hall sensor requires input voltage, this voltage matches VCC
• **Vmin: 3.3V**	Minimum voltage output of Hall sensor	The Hall sensor requires input voltage, this voltage matches VCC
• **Left Rotation: 16 counts**	Controlling encoder to spin left	Using just a single edge of one channel results in 16 counts per revolution of the motor shaft, so the frequency of the A output in the above oscilloscope capture is 16 times the motor rotation frequency.
• **Right Rotation: 16 counts**	Controlling encoder to spin right	Using just a single edge of one channel results in 16 counts per revolution of the motor shaft, so the frequency of the A output in the above oscilloscope capture is 16 times the motor rotation frequency.

4.8.5 - Verification Plan

1. Connect motors to ATMega128 board
2. Connect encoder sensors to ATMega128 board
3. Power up ATMega128 board
4. Send revolution data to motors

5. Verify motors' rotation

4.8.6 - References

[1] "Model:16sg-030PA-en voltage:6vdc - cdn.robotshop.com." [Online]. Available: https://cdn.robotshop.com/rbm/a00a7635-653b-4220-aac9-b0c23c5c5e2c/8/8bb18634-8d93-4cc9-a271-1218b33f7358/9965da24_16sg-030pa-en-6v.pdf. [Accessed: 04-Mar-2022].

4.8.7 - Revision Table

02/16/2022	Triet Nguyen: Update general validation
02/15/2022	Triet Nguyen: Update description, general validation, added figure 3
02/03/2022	Triet Nguyen: Update design, interface validation, and verification plan
02/02/2022	Triet Nguyen: Document Created

V. System Verification Evidence

This section will go over the system verification information and evidence. The first part of this section (5.1) will be about the universal constraints - rules by which all OSU 2022 Capstone projects had to abide by. The second part of this section (5.2) will go over each of the individual requirements that were set for the system.

5.1 - Universal Constraints

5.1.1 - System may not include a breadboard

To satisfy this constraint, a custom PCB was developed to serve as both a DC-DC buck converter, and as a distribution board to supply power to each of the devices connected to the tentacle. Breadboards and protoboards were used to test and design the power supply PCB, but were not used in the final product.

5.1.2 - Must contain a student designed PCB, and a custom Android/PC/Cloud application

The project contains a student-designed PCB that contains a DC-DC buck converter, and a distribution board to supply power to each of the motor drivers and encoders connected to the tentacle.

The application used to control the arm runs on any PC that can sufficiently run Blender (works on Windows, macOS, and Linux). There is an interface written in Python used to enter coordinates, which interfaces with a Blender plugin that is also written in Python that updates the arm in the simulation.

5.1.3 - If an enclosure is present, the contents must be ruggedly enclosed/mounted as evaluated by the course instructor

No enclosure was required for this system, however; all electrical components (inclusive of motors, sensors, and motor drivers) have been securely mounted to the polycarbonate base plate attached to the base of the tentacle.

5.1.4 - If present, all wire connections to PCBs and going through an enclosure (entering or leaving) must use connectors

Wire connections to the PCB have been designed to attach via pin headers. In addition, the proprietary cables that are used to connect the encoders and DC motors have been spliced with pin header compatible cables. All wires in the system can be removed.

5.1.5 - All power supplies in the system must be at least 65% efficient

As discussed in section 4.6, the power supply was designed with a predicted power efficiency of 79.5%. Using a spreadsheet from Texas Instruments,estimated power dissipated in each of the selected components was determined, and used to calculate this power efficiency value.

5.1.6 - The system may be no more than 50% built from purchased ‘modules’

In terms of software, no paid programs or code was used. Everything used in the tentacle system was either free and open source, or self-developed.

In terms of purchased electronic modules, a total of 9 motors with encoders, and 5 motor drivers were purchased. An ATMega128 development board was also used. The power supply PCB and the tentacle arm were self designed and manufactured.

5.2 - The system will extend 1.0 meters in any direction

5.2.1 - Requirement

At full extension, the system will have a minimum reach of 1.0 meters. The system will be able to extend in any direction originating from its base.

5.2.2 - Testing Processes

The system passes if when the tentacle arm system is fully extended in any direction parallel to the ground, the center of the base of the arm is a minimum of 1.0 meters away from the end of the tip of the tentacle.

5.2.3 - Testing Evidence

Following the verification procedure outlined in section 4.1.5 for determining the maximum reach of the tentacle, the mechanical tentacle system was set in the vertical and horizontal positions by tensioning the control cables. The following figures show the measured radial lengths of the tentacle in these two positions. Note that the red-markers on the tape-measure each indicate a distance of 12.0”.

Figure 5.2.1: Vertical reach, full length photo.

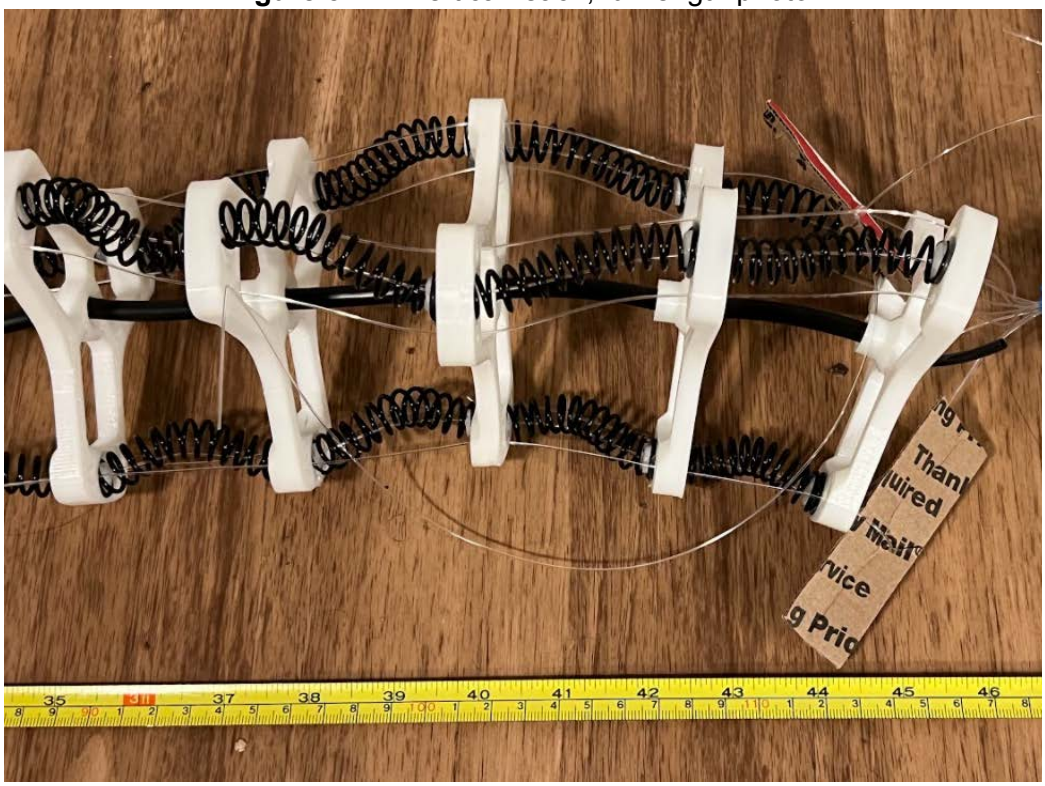


Figure 5.2.2: Vertical reach, close-up photo.

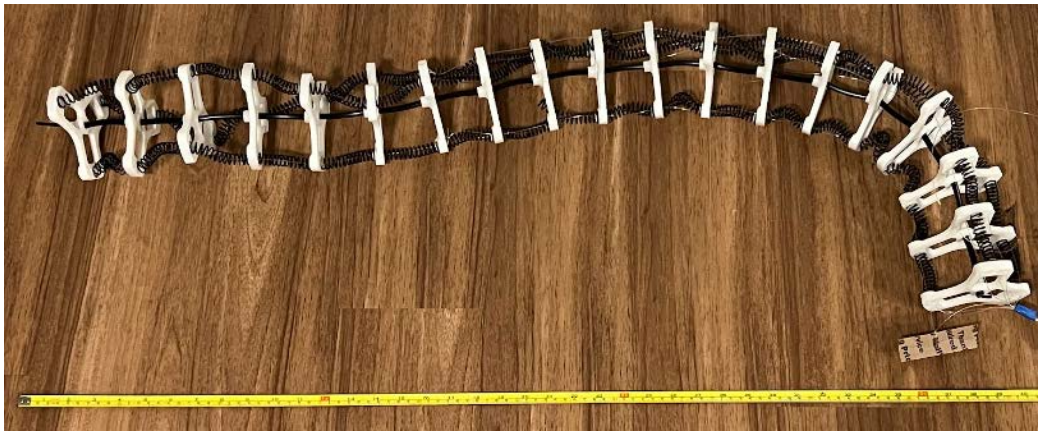


Figure 5.2.3: Horizontal reach, full-length photo.

Figure 5.2.4: Horizontal reach, close-up photo.

5.3 - The system will support a mass of 0.5 kilograms

5.3.1 - Requirement

At full extension, the system will support a mass of 0.5kg placed at the tip of the tentacle. The system must be stable and be able to move while the system is supporting the mass. The mass can take the form of an attachable manipulator (grabbing device) or a non-functional weight with a measured mass of 0.5kg.

5.3.2 - Testing Processes

The system passes if it moves at full extension while supporting a load with a mass of at least 0.5kg.

5.3.3 - Testing Evidence

5.4 - The system will be accurate

5.4.1 - Requirement

The system will be able to move to the user-specified location with an accuracy of ±2.0cm.

5.4.2 - Testing Processes

The system passes if it is able to move to a location within 2.0cm of the user-specified coordinates.

5.4.3 - Testing Evidence

5.5 - The system will be reliable

5.5.1 - Requirement

The system will move to the user-specified location with a success rate of 90%.

5.5.2 - Testing Processes

The system passes if it is able to move to the user-specified location 9/10 times.

5.5.3 - Testing Evidence

5.6 - The system will be self-powered

5.6.1 - Requirement

The system will operate continuously for a minimum of 2 hours on a fully-charged battery. Batteries will be rechargeable.

5.6.2 - Testing Processes

The system passes if it is able to operate continuously without recharging for 2 hours.

5.6.3 - Testing Evidence

5.7 - The system will be portable and easy to use

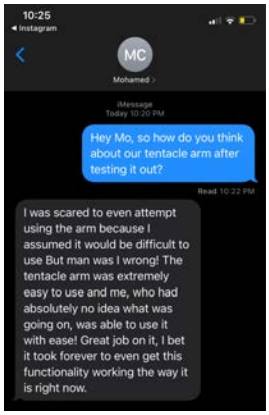
5.7.1 - Requirement

The system will be portable and user friendly. 9/10 users will agree that the user interface and system are portable and easy to use.

5.7.2 - Testing Processes

The system passes if 9/10 users agree that the interface and system are easy to use.

5.7.3 - Testing Evidence



5.8 - The system will output a user-friendly 3D visual.

5.8.1 - Requirement

The system will output a 3D simulation of the arm for motion control and user reference. Users should be able to move the real arm to a target location using only the 3D model in the simulation.

5.8.2 - Testing Processes

The system passes if 9/10 users are able to move the arm to a location within 2.0cm of a target location using the 3D simulation.

5.8.3 - Testing Evidence

The following is a screenshot of the Blender simulation visual output. In the Blender environment, control explanations have been included on how to control the tentacle arm.

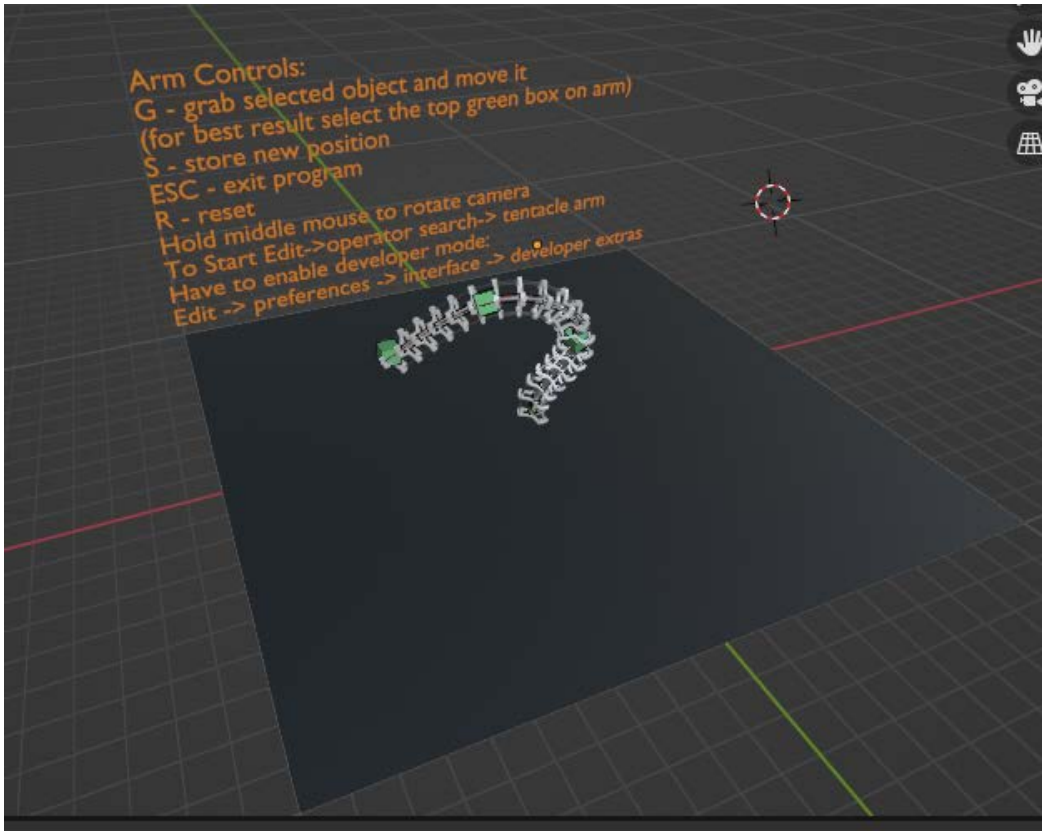


Figure 5.8.1: Screenshot of Blender simulation.

5.9 - The system will support input in spherical, cylindrical, and cartesian format

5.9.1 - Requirement

The system will accept input coordinates in spherical, cylindrical, or cartesian (XYZ) formats. If the coordinates are not within the range of the arms motion, then the system will warn the user.

5.9.2 - Testing Processes

The system passes if it is able to parse the three formats of input coordinates correctly, and indicate to the user if those coordinates are within the bounds of the arms motion. Values to be tested:

x	**y**	**z**	**Expected result**
1	2		Invalid (z is empty)
2	4	-5	Invalid (z is negative)
38	0	4	Invalid (x > 36”)

r	**ϴ**	**z**	**Expected result**
-8	1	2	Invalid (r < 0)
2	7	23	Invalid (theta must be between 0 and 2 pi)
1	0	1	Valid

ρ	**ϴ**	**φ**	**Expected result**
3	3.3	2	Invalid (theta must be between 0 and pi)
1	2	3	Valid

5.9.3 - Testing Evidence

The user is presented with a window to input coordinates. Changing the coordinate system dropdown will automatically convert the coordinates to the new coordinate system (provided that they've been entered). If the coordinates are invalid, a message will be output in the message log.

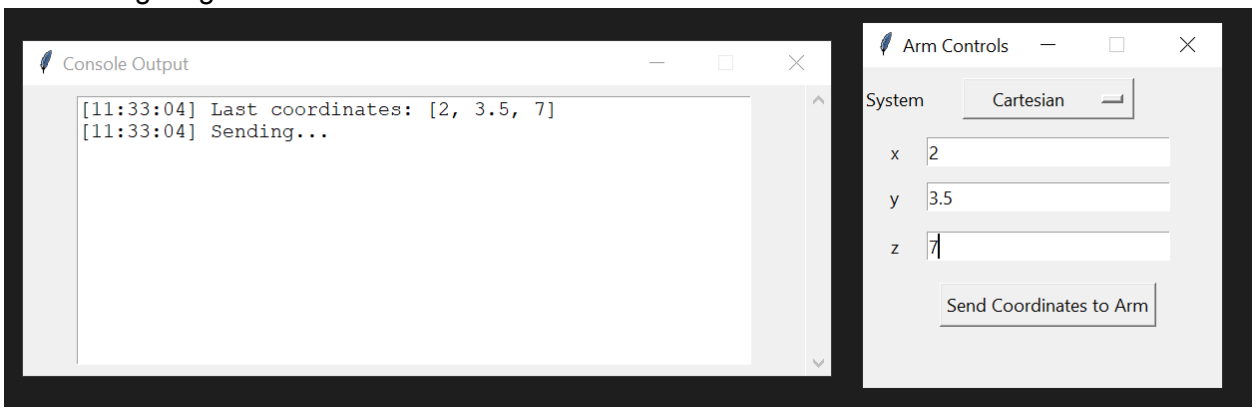


Figure 5.9.1: Coordinate entry window, with some cartesian coordinates entered

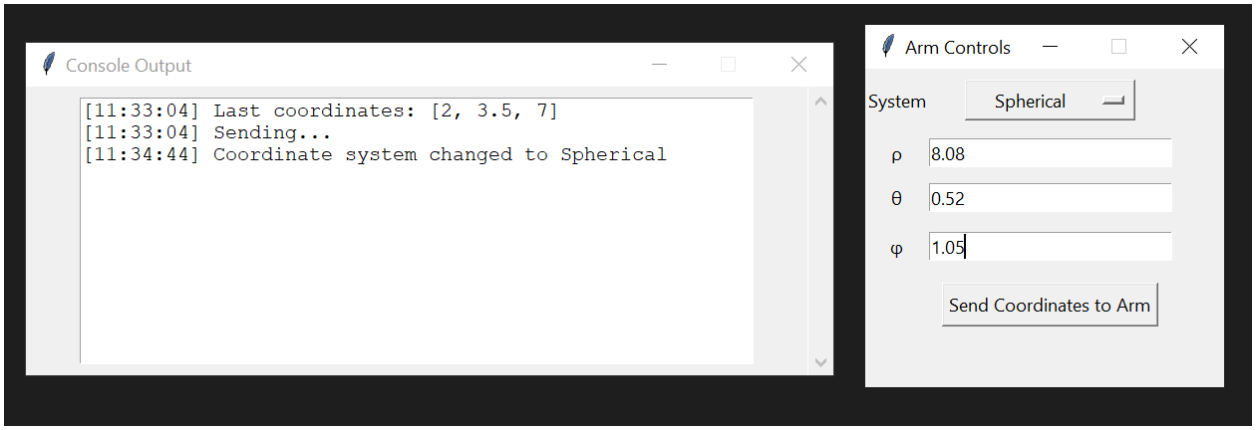
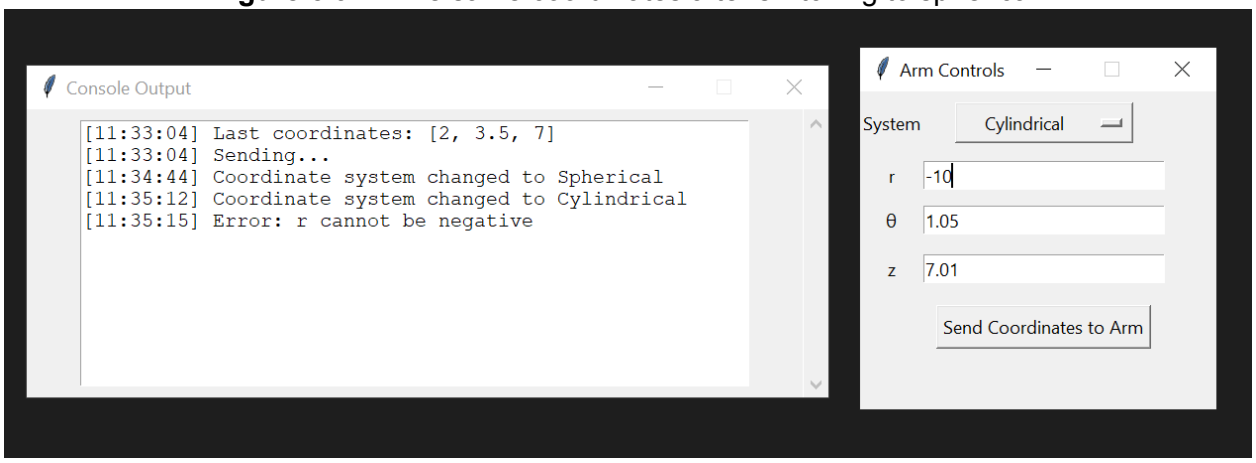


Figure 5.9.2: The same coordinates after switching to spherical



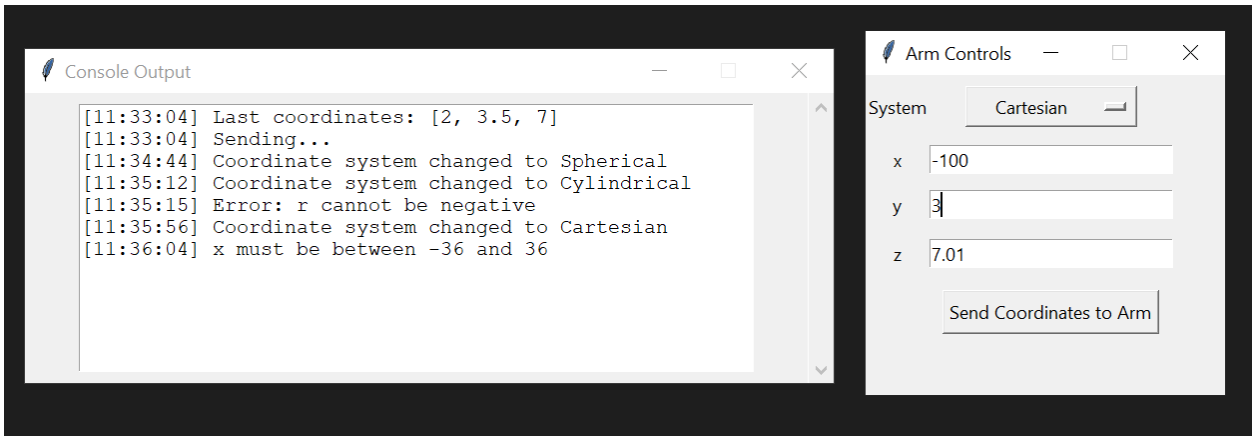


Figure 5.9.3: Errors printed in the log if the coordinate values are invalid

5.4 - References and File Links

5.5 - Revision Table

3/14/2022	Ben Chan, Cale Hallamasek: Testing evidence for R2, R9

3/3/2022	Ben Chan, Jordan Porter, Cale Hallamasek: Created initial section

VI. Project Closing

6.1 - Future Recommendations

6.1.1 - Technical recommendations

The system uses a USB-B to UART cable for the communication between the computer and the ATMega128. The cable doesn't have any labels on its wires; therefore, it posed a problem for the team to figure out the functionality of each wire during the process of setting up the communication line. Based on this problem, a recommendation for the future is to label the wires to help accelerate the testing process.

Something that can help software troubleshooting in particular is taking note of what electronic devices your group members have at their disposal. It can be good to have a collection of different computers for software testing, preferably with different operating systems. Documenting what versions of software your group is using (applications, libraries, dependencies), and defining a consistent code style can make integrating software blocks a lot easier.

The coordinate entry interface is written in Python, as the language allows for rapid prototyping and runs on many different operating systems. If the user has Blender installed for the arm visualization, Python is automatically bundled in the installation; however, if the user does not have Blender, they might have to install Python manually. Installing Python just for the coordinate entry interface takes up more space than necessary; it is likely to use over 100 MB. The interface could be re-written in a compiled language, such as C++, in order to use less space and system resources.

The 3D modeled and manufactured parts for the mechanical tentacle segments were designed with modularity and printing convenience in mind. As a result, these parts could also be easily manufactured on a CNC, as there is very little complexity that could not be machined with a simple milling machine or 3-axis CNC. If future group's plan to re-use the same 3D models, then this gives them the freedom of manufacturing the parts in different materials (wood, aluminum, etc.) or at a larger scale than could be achieved on a small consumer-grade 3D printer.

Additionally, due to cost restraints, materials that were sourced for the development of the 3D mechanical system were not ideal for the given application. One issue that was encountered had to do with the chosen springs, and how they interfaced with the 3D printed tentacle segments. Solutions to the spring issues involved decreasing the space between tentacle segments (and therefore the spring lengths), increasing the strength of the springs (the spring constant, k), and increasing the radius of the springs.

6.1.2 - Global impact recommendations

The materials used to build our physical arm were mostly plastic since the joints of the arm were 3D printed. The use of plastic has been the major cause of ocean pollution in recent

years. We decided to 3D print our physical system to save time and money on purchasing more expensive materials. By doing this, we admit we are contributing a small part to the rising issue of marine pollution and we hope future teams can do better. A recommendation we have for the material used in a future system is either wood or aluminum since these materials are cheap, light, easy to work with, and sturdy. Some issues that could arise with using these materials is that wood or metal manufacturing processes are typically subtractive, and would thus generate more waste.

Another impact to be aware of is that due to the cost and time restraints of this project, many components were sourced from overseas companies, specifically in regard to electronic components such as resistors/capacitors/inductors. Additionally, the PCB machining company chosen for the power supply board is also a Chinese company. The specific issues with sourcing from these overseas companies is more explicitly outlined in the Design Impact Assessment in section 2.2.

6.1.3 - Teamwork recommendations

Teamwork plays a big role since this project includes working interchangeably between different blocks. The team needed to meet many times to discuss and to work on the communication between our devices. The communication between the main computer and the microcontroller uses USART which isn't as efficient in decoding information as was originally expected, therefore it took the team a long time to figure out how information was being sent and received. Looking back at the experience as a team, we recommend future groups to collaborate more to develop better and more efficient communication protocols.

Specific advice for future groups to work more efficiently are to meet in-person as often as possible, and to check in on team members progress to both A) ensure that your blocks interface correctly, and B) help one another with the problem solving progress. While our group made an effort to meet together in-person when possible, the consistency and frequency of those meetings could've been greater. Unfortunately due to world events (specifically Covid-19 and as a result of that, online courses) in-person meetings were difficult to schedule.

6.2 - Project Artifact Summary with Links

Documentation:

ATMega128 User Manual:
https://ww1.microchip.com/downloads/en/DeviceDoc/doc2467.pdf

Tkinter (Python GUI library):
https://docs.python.org/3/library/tkinter.html

Software:

Python 3.10.4 Release:
https://www.python.org/downloads/release/python-3104/

Blender, installation:
https://www.blender.org/download/

Blender, add-on documentation:
https://docs.blender.org/manual/en/latest/editors/preferences/addons.html

Project repository: https://github.com/mangosaver/OSUCapstoneArm

Hardware:

3D part model - Tentacle segment v2 STL:
https://drive.google.com/file/d/1GngXeKhra0QlLi_Q6oG50bkjbQzriB2w/view?usp=sharing

3D part schematic - Tentacle segment v1:
https://drive.google.com/file/d/1Ol0qvrTrmCEtNq8yoThlJrNkuyL3HSpe/view?usp=sharing

3D part schematic - Tentacle segment v2:
https://drive.google.com/file/d/1NCF5LiGbf1Z7Ftos2sPW5bNW1jnsgaQp/view?usp=sharing

Buck Converter PCB - Top layer:
https://drive.google.com/file/d/1znSDSESc2elGaJGDSapxr766CZgGNvyu/view?usp=sharing

Buck Converter PCB - Bottom layer:
https://drive.google.com/file/d/1S116rQdohbOpnVSQU2i5t9s1NQExNqk5/view?usp=sharing

Buck Converter PCB - No planes:
https://drive.google.com/file/d/1O9gxm4I-9XDsGr8nZZtYHYCEKOsF_QnI/view?usp=sharing

Buck Converter PCB - Schematic (Autodesk Eagle):
https://drive.google.com/file/d/1PbvEtNjrtqu0pOp3Ozhru-ZWeG1D_qlS/view?usp=sharing

6.3 - Presentation Materials

6.4 - Revision Table

5/6/2022	Ben Chan: Added hardware artifacts.
5/6/2022	Cale Hallamasek: Added content to recommendations sections