Designing The Conformable Lumen Assessment Robotic Assistant

A Soft Bodied Mobile Robot for Pipeline Inspection.

Antonio Giancarlo Sanchez

A thesis presented for the degree of
Master of Science
in
Robotics Engineering

Worcester Polytechnic Institute
Worcester MA
May 2025

APPROVED:

Name: Professor Cagdas Onal, Major Thesis Advisor

Name: Professor Yunus Telliel

Name: Professor Andre Rosendo

Contents

List of Figures

List of Tables

List of Symbols

Number sets

κ	Arc Curvature
θ	Arc Angle
s	Arc Length

Other symbols

ϕ	Plane Transformation Angle
$l1$	Cable 1 Length
$l2$	Cable 2 Length
$l3$	Cable 3 Length

Physics constants

ρ	Arc Radius
c	Arc Center
o	Base Plate Origin

Abstract

Pipe infrastructure is an extremely important issue for modern day society. Pipes transport necessary materials such as clean water, oil and gas, and hazardous waste. Failure in these networks can lead to millions of dollars in damages and can endanger communities depending on the payload. In order to ensure the health of these structures, pipes need routine inspection to identify any potential failures. The inside of pipes especially need inspecting due to failures at pipe joints and internal corrosion and cracks due to transported material. In order to keep repair efforts to a minimum, non-destructive testing (NDT) is typically preferred. The purpose of this thesis is to update and test a compact wall-press pipe inspection robot with a novel design centered around using origami inspired continuum modules. The Yoshimura module allows this design to have an extensive level of flexibility similar inspection robots do not possess. A variable suspension system and front viewing camera allow it to traverse and inspect a multitude of different pipes. To determine the efficacy of the design, tests involving speed, battery life, verifying the kinematic model, and recording the time it takes to navigate different sized pipes and joints were preformed.

Acknowledgements

I would like to thank the following people and groups for their contributions to my thesis project:

- This material is based upon work partially supported by the National Science Foundation (NSF) under Grant No DGE-1922761 and Grant No NSF CMMI-1752195 and Amazon Fulfillment Technologies via the Robotics Day One Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or Amazon Fulfillment Technologies.
- The previous 2022 and 2023 MQP teams for the initial groundwork that made this current design possible
- Soft Robotics Lab member Gabby Conard and Tim Jones for guidance on design challenges and the approach to the test setups.
- Dr. Cagdas Onal for guiding me through the thesis process and his thoughtful insight.
- My parents and brother for much needed moral support throughout my time as a master's student.
- Madie Gorman for the support that made this project possible.

Chapter 1

Introduction

Mobile robotics have been used in numerous fields for a variety of reasons and in a variety of settings. Reconnaissance robots have been used to gather data in harsh or inaccessible environments such as Mars or deep sea depths or for tasks deemed too dangerous such as bomb defusals and exploring unstructured disaster areas. Payload robots have been used to move large heavy objects deemed unsuitable for human laborers to move. Swarms of these robots can be programmed to coordinate complex patterns such as those seen at Amazon processing facilities. Mobile robotics is one of the two core groups of traditional robots, meaning that the number of applications is half or even more than the totality of robots can do. In this chapter I will introduce an update of a novel pipe inspection robot and demonstrate the necessity of this robot as it applies to wider society and the problems it seeks to solve.

1.1 CLARA

CLARA, or the Compliant Lumen Assessment Robotic Assistant, is a pipe inspection robot created and then updated by two separate MQP teams under the guidance of the Soft Robotics Laboratory. The concept was to use the Yoshimura module that the lab had developed and use it as the basis of the robot's body. Traditional ground mobile robots are wheeled and have rigid bodies. The Yoshimura's design allows for multiple degrees of freedom in bending and in compression when cable actuated. Having the Yoshimura module as the base body allows for a greater range of flexibility within a pipe environment. This is especially critical at junctions such as elbows and tee joints.

The other mechanical aspects of CLARA includes a variable diameter suspension system that allows it's motor driven silicone wheels to adjust to different sizes of pipes and press against its walls for needed traction. This is a common design within pipe inspection mobile robots. Pipe networks can often shrink or expand in diameter as the material flows through it. This system allows CLARA to continue operating even when reaching those junctures and still maintain traction. Combining the Yoshimura body and this traditional wheeled design will give CLARA the speed and flexibility to effectively navigate different networks of pipes.

CLARA has two microcontrollers on it that processes the motor commands and streams visual feed respectively. The first is a Tiny Pico UM which issues simple commands to the 11 different smart motors that CLARA has. These smart motors are comprise of an N:20 micro-motor and miniature PCB board called SIMCs (Smart I2C Motor Controller). These SIMCs contain a microchip that processes these commands and preforms the necessary PID calculations to get the desired result. The second is the ESP32 Cam and lens that transmits a live feed to the operator's computer so that they can remotely pilot CLARA through the network. The robot operation is handled at the operator's computer station through an ESP-32 micro controller transmitter.

There are 4 main pieces of software and a set of firmware that controls the robot. The first 3 files dictate the robots actions via the Tiny Pico. A python file interprets the operators actions via an X-box gamepad and transmits this information to the transmitter connected to both the computer and gamepad via the USB serial bus. The operators actions are interpreted via an Arduino .ino on the transmitter and sent to the Tiny Pico via a signal packet. The packet is unpacked and informs the Tiny Pico which actions to take based off of an .ino file that is flashed onto it. The SIMC boards contain firmware which is a set of Arduino files and libraries written in C that allows it to interpret the commands from the Tiny Pico. The last file is the .ino file that sets up and allows the ESP-32 camera to operate.

1.2 Initial Broader Impacts

This section will analyze the justification for CLARA through the problem it was designed to solve and the broader impacts of said problem. CLARA was meant for the purpose of monitoring and reporting on pipe infrastructure. Pipe infrastructure delivers water, oil and gas, and other hazardous material. These networks can be found in regular commercial buildings such as houses and stores to industrial plants such as water treatment facilities and oil refineries. These networks are often vast and intricate with many different bends and junctions in order to transport the material efficiently and maintain adequate pressure. A breakdown in these networks can incur financial losses and cause physical harm to a population. Restrepo et al. [1] derived a model to predict the financial cost of pipe failures for networks transporting hazardous chemicals. Their model takes into account a variety of factors such as cause of failure, location, amount of resources lost, and any volatile reaction that might occur and the location of the occurrence in order to find a lower and upper bound of what the financial cost will be. A hypothetical scenario involving internal corrosion of an onshore pipeline cost as much as $523 thousand in product loss, property damage, and cleanup based on their model. Rifaai et al. [2] justifies the creation of their water infrastructure scoring model on stating that that breakdowns in water distribution networks (WDN) can potentially lead to $1 trillion worth of repairs and restructures. Taiwo et al.[3] expands on this by saying that water loss can be as up to 30% in most global WDNs, which creates major financial losses in terms of the resources lost and time spent fixing the issue. They mention that the money needed to properly fix these issues isn't typically available as they cite that in 2006 only $1.2 billion was spent out of the requested $6 billion.

Another aspect is the humanitarian cost that these failures incurred. The previous financial model [1] for oil and gas pipeline failures factors in possibilities of explosions and in which areas they may occur. Pipelines are typically located away from residential areas, but this fails to mention any maintenance workers or facilities that might be caught in these explosions. Beyond financial concerns, human lives could be put at risk if these types of pipes are not properly taken care of. Rifaai et al. [2] determined that the rate of deterioration and the ability to repair these damages is widening, leading to an unreliable WDN infrastructure. The ability to get potable water will start to become a challenge for the average person if this is not resolved before hand. Taiwo et al. [3] gathered the number of pipe bursts per 100 miles or kilometers depending on the country. These disruptions can create a large loss in potable water and the study even found that this was the case in the UK with a loss of 22% in water yearly. The study justifies their model further by stating that roughly 780 million people are effected by issues like this and 3.41 million die yearly due to poor potable water conditions.

1.3 Causes

Due to high cost of these failures, many studies have been conducted to find the different types of failures and their frequency. Determining which failures are the most prevalent will assist with any inspection process on the network. How and where to find any defects in the system will facilitate the implementation of any preventative measures. It's been determined by several of the aforementioned studies[1, 2, 3, 4] that factors such as pipe size, length, wall thickness, age, corrosive substances, and pressure can cause different internal failures such as fractures and pitting.

Due to CLARA's size, this section will be focusing more on issues that effect pipes networks with smaller sized pipe diameters such as WDNs and pipes found in chemical refinement plants. Large transport pipes like oil and gas pipelines will not be the main focus. Taiwo et al. [3] found that specifically circumferential cracking and corrosion pitting effect pipes less than 200 mm in diameter. These can be caused by the outside factors such as the environment the network is in such as soil acidity and the bedding around the pipe or temperature issues freezing temperatures creating frost. Internal factors such a water quality and micro-organisms can cause pitting in the inside and lead to a blow out, something all pipes are susceptible to. An elevated level of trace elements such as chloride and sulfates as well as unwanted bacteria can corrode the material from the inside. The study by Respetero et al. [1] suggests that internal corrosion is the cause of 5.9% of known pipe failures, the second largest value of the known reasons for pipe failures in their study.

The general shape and condition of the pipe is also a contributing factor. All of the aforementioned studies [1, 2, 3, 4] agree that age, length, and pipe diameter are a factor in pipe failure. Mahmoodian et al. [4] heavily studies how the age of a pipe fails due to a variety of conditions such as internal pressure, size of internal pitting that is present, and the corrosive content of the liquid. Figure 1.1 from Mahmoodian et al. [4] shows the correlation between the age of pipes and probability of failure given the correlation coefficient that is tied to the aforementioned causes. The larger the correlation coefficient

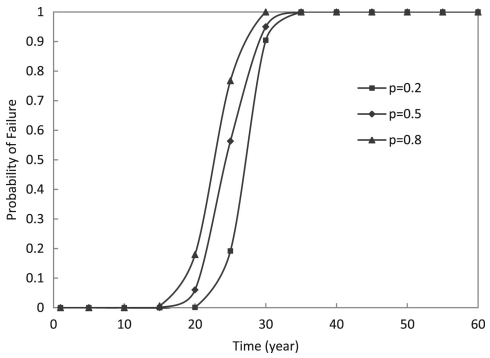


Figure 1.1: Probability of failure for different coefficients of correlation ρ (Mahmoodian et al.)

the sooner the pipe is going to fail. It draws a clear conclusion that older pipe networks are more susceptible to failure under most conditions. Rifaai et al. [2] noted that there can be 13 to 19 different failures every 100 km of pipe in a year. Their data set showed that the city of Austin, Texas had 6.5% of it's 5202.1 mile long WDN, a portion of which is pictured in Figure 1.2, failed within the time frame of 2009 to 2010. Larger stretches of pipe have more material and resources to move, meaning the probability of it failing increases. This same study identified that smaller diameter pipes with thinner walls tend to blow out as well and considered this factor heavily within their model. Taiwo et al. [3] discusses the same thing and notes that pipes with diameters diameters 300 and 150 mm fail 6 and 43 times within a year respectively, drawing a conclusion that as a pipe shrinks it is more likely to have issues. Wall thickness is directly correlated to pipe diameter and is thus the main contributing factor to this problem with smaller pipes.

Figure 1.2: Portion of Austin Texas's Water Distribution Network (Rifaai et al.)

The costs and causes of pipe failure are important to understand as it is the basis for why CLARA exists. As a pipe inspection robot, CLARA is meant as a preventative tool to identify these causes so that proper maintenance can be conducted and future loses would not have to be incurred.

Chapter 2

Related Work & Project Goals

This chapter will cover the types of ways that technicians currently inspect pipes as well as novel methods that are being actively developed at other universities. These types of inspections methods range from simple sensor packages pushed through the pipe to traditional forms of mobile robots and finally novel robots that use methods of traversal that go beyond wheel driven bodies. The pros and cons of these types of robots will discussed and how they may compare to the proposed design of CLARA. The chapter finishes with a review of the different mobile robots that the Soft Robotics Lab has created and how the new version of CLARA has taken inspiration and improved upon certain aspects of these designs.

2.1 Traditional Methods

Pipe inspection technology has been a major focus for these industries that have a stake in the health of these networks. Thus there have been a variety of different methods developed to inspect these pipes. Literary reviews[5, 6] compiled the different methods that are used to monitor various pipe networks such as oil & gas and WDNs. These reviews focused mainly on Non-Destructive-Testing (NDT). This is when a method can preserve the integrity of the system and still determine if it is being compromised by a defect. In some cases, these methods can be done during normal operation and even require it to collect the data. Many of these methods use signals ranging from acoustic, thermal, and electromagnetic frequencies. These signals are processed by the way they interact with the internal and external geometry of the pipe as well as the material itself. This is then used to determine if and where a defect in the network maybe. Visual methods such as simple CCTV or laser scanning is also employed, but are typically tied to a mobile robot that is moving through the network. Image processing and surface topography data from each respective method reveal defects directly on the pipes inner surface.

These studies [5, 6] determined that these methods can give a map out the pipe in very fine detail. For example, Carvalho et al. [7] determined that inspection using radiographic, ultrasonic, and light diffraction techniques were found as small as a millimeter in height and 20 millimeters in length. The study even automated the process by adding an artificial neural network to automatically identify and classify the defect. However these methods can also be challenging to preform due to how signals are sent and received. The same paper [7] noticed that the error in sizing the defects can be up to 29% for the manual inspectors and the automatic light diffraction method can't identify some errors. These types of signal processing is very complex, and error can easily creep upwards as it depends on how well the sensors are placed and calibrated. The methods that rely on bouncing signals off the geometry, regardless of the type of signal, have these exact issues. The range of pipe length and sizes are other factors to consider. Signal interaction may be limited and even putting the sensor on or in the pipe can be a very involved process. Covering the entirety of a network this way can be a very long and tedious process. While methods involving a visual aspect such as video feed and laser are simpler in terms of setting the process up, it still requires another system to get it to move through the network. The optical inspection system developed by Safizadeh et al. [8] can detect pit holes as small as 2 mm in diameter. It is especially effective as the tests show that the laser diode light's narrow angle makes the intensity of the defect detection greater as demonstrated by Figure 2.1. However, this system still needed a method of transport and worse it needs to be tethered for the CCD camera. Due to the improvements in technology, a wireless system would be feasible, so adding it to a mobile robot like CLARA would be extremely advantageous.

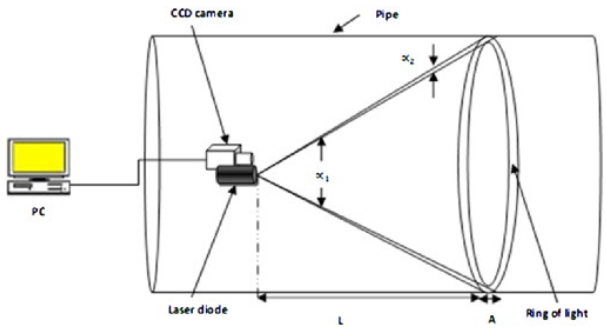


Figure 2.1: Schematic of NDT Optical System (Safizadeh et al.)

A specialized sensor package called a Pipeline Inspection Gauge (PIG) is heavily used in pipe inspection. They are a type of instrument that contain the the sensors that preform the aforementioned signal tests and are run through the pipe. Most have linkages that allow it to bend around junctions of the pipe. Its movement is typically passive and relies on the flow of liquid to traverse the network. Very few have limited controllability and most can be considered sensor packages. Ma et al. [5] compiles a list of PIGs which ones are best for which type of defect they want to identify, liquid medium the PIG is traveling in, and the general features of the pipe. These instruments can help with issues of sensor placement mentioned above, but still have the issues of passively relying on fluid mediums and the large size of the instrument itself.

2.2 Robotics

This section will discuss two types of robots. The first type is wheeled robots with rigid links. Typically in smaller pipes these robots have to rely on the geometry of the pipe to gain traction when moving. These are known as wall-press robots as there a mechanisms that push the wheels out radially to make contact with the inner walls. The other type of robot is one that uses novel movement to traverse the pipe. There are a variety of reasons to why these robots are considered novel. They range from what actuation system they are using to the exact movement pattern of their body. Both types will be analyzed for its pros and cons. The majority of these robots can be found in these literary reviews [9, 10].

2.2.1 Wheeled Robots

The way a wall-press robot works is that the suspension system holding the wheels is positioned in a way where the wheels gain traction from the curved pipe walls. The curvature of the pipe makes it so that these robots are unable to properly gain traction. A suspension mechanism needs to provide the necessary radial force towards the surface of the pipe wall to gain traction for movement along the pipe. At least 2 points of contact are needed with more providing stability. The points of contact must also be evenly spaced so that the force applied by the suspension mechanism is evenly distributed. Moving through pipe junctures are a challenge for these robots due to most wheeled robots having a rigid chassis and curves in pipes necessitate changes in the suspension system diameter as the robot moves through it. The biggest advantage of these robots is their speed and traditional control scheme for steering and propulsion.

The suspension system itself is important as the design can determine how effective the wheels gain traction. Kahnamouei et al. [10] details the different types of variable suspension systems. They typically involve a rotary system or sliding crank to position the linkage structure where the wheels make contact. This can be an issue as there are many considerations when designing a linkage system that can adjust to any changes in the pipes shape or size as well provide the necessary force for traction. Ni et al. [11] goes heavily into the calculations needed to properly create a variable suspension system. Their system is comprised of both an actuated and passive elements that allows their robot to be flexible within the pipe. A motor and series of belt and pulley systems drives a lead screw that extends and retracts a support link that pushes the cantilever the wheels are attached to. Passive springs help push the cantilever for added traction force, but also help clear small obstacles as it makes the cantilever compliant. Jatsun et al.'s [12] design employees a similar method of a belt system to actuate a screw to push out each individual wheel with the same passive spring. This paper goes into similar calculations on the required normal force to properly move.

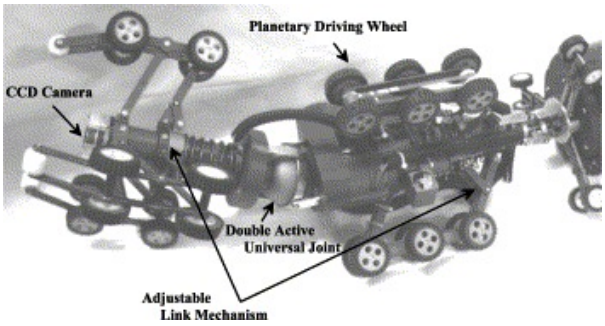


Figure 2.2: Choi et al. Steerable Pipe Inspection Robot

The biggest issue for wheel pressed robots is they are typically difficult to navigate within curved pipes. Navigating elbow junctions in particular is difficult as the size and shape of the chassis heavily factors into maneuverability. Moving through the curve also requires a change in suspension diameter. In order to account for these factors, novel changes to the rigid chassis have been designed in previous robot designs. Choi et al. [13] was one of the first instances of creating a robot that was a chain of rigid sections connected by universal joints. Figure 2.2 from the same paper details the basic structure of the robot. Furthermore, the paper goes into detailed calculations that can determine if a rigid section can clear an elbow based on its height h and width w of the section. The following equations describes a rigid section that clears an elbow if the width is significantly smaller than the height leaving one or both ends of the section outside of the curved section when moving:

$$0 < w \leq ((R + D/2)\sin(45°) - (R - D/2)) \tag{2.1}$$

$$h = 2\sqrt{2 * (D/2 + R - (R - D/2 + w) * \cos(45°))} \tag{2.2}$$

The second scenario would be when portions of both ends of the rigid structure is within the curvature of the elbow due to the larger width compared to the height.

$$((R + D/2) * \sin(45°) - (R - D/2)) < w < D \tag{2.3}$$

$$h = 2\sqrt{(D/2 + R)^2 - (R - D/2 + w)^2} \tag{2.4}$$

In this scenario, there are points in which some wheels might not be in contact with the walls, meaning that the traction of the remaining wheels and stability of the body are important to maintain movement. Figure 2.3 from Choi et al. shows the two different scenarios that the above equations describe.

Preceding designs [14, 15] have gone on to do similar concepts of segmentation and simplifying it by only having two points of contact and omni-directional wheels to help maneuver down a desired path with the simplified design. Kakogawa et al.'s [14] design has been proven to navigate extended circular pipe sections in simulation and some tee joints in practical test setups.

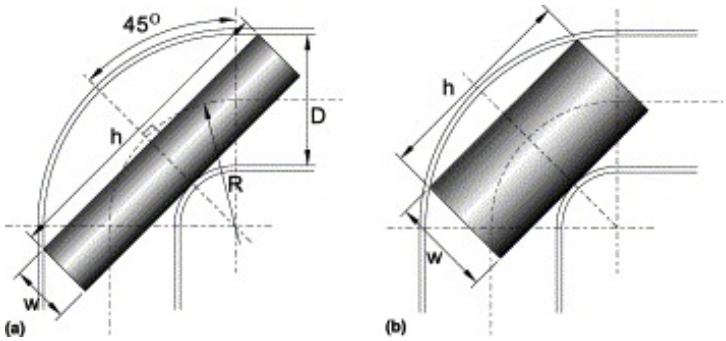


Figure 2.3: Choi et al. Rigid segment diagram

2.2.2 Novel Movement

Due the complexities of moving a traditional robot through a pipe network, many methods of novel pipe movement have been developed. Their movements often take inspiration from biological agents such as caterpillars[16] and micro organisms [17]. Typically these robots have unique methods of actuation involving pneumatics [18], linear motion [16], cable wires [17], and even legs [19]. These robots are broken down into several categories as outlined in the literary studies [9, 10]. They are inchworm, screw, and walking.

Inchworm Type

Inchworm type robots utilize linear motion and a combination of anchoring methods to move along the pipe. Typically the steps are to anchor the backend of the robot, extend the body of the robot, anchor the front end of the robot, de-anchor the backend, retract the robot, and then anchor the back end again. This motion will be repeated ad nauseam to reach the end. Examples of this are the the robots designed by Hayashi et al. [18] and Chablat et al. [16].

The first one is an inchworm utilizing pneumatic actuation to expand and contract a silicone rubber tube they call EFPAs. Combing them into a set of 3 for the main body and adding 1 on each end in a ring shape to act as anchors as shown in Figure 2.4 The expanding and contracting the 3 tube in the body can get you different configurations to the point that it is considered a 2 DOF robot with 1 DOF axial movement. Expanding the ring shaped EFPAs allows the ends to press up against the pipe and lock itself into place. This robot has been proven to work well in small diameter pipes with the smallest they tested for being 50 mm diameter. It has also been proven to pass through elbows and tee joints as well as inclined pipes. The biggest drawback is that it requires to be tethered to a compressor and valve unit. It is also seen to be slow, averaging at 14 mm/s.

The design from Chablat et al. [16] opts for prismatic joints powered by rotatory motors. Sliding legs can expand and retract in order to dig into the inner walls. This particular design and how its motions are set up allows it to traverse variable pipe diameters ranging from 32 mm to 52 mm. Due to the clamping nature of their design, force calculations were pivotal in determining the efficacy of this robot. The normal forces of the pipe walls effect the main body differently at different points in the movement cycle. Learning this allows the user to output the most efficient level of clamping force during the movement cycle as well as during movement through different configurations of pipes. No mention of speed was in the conference paper. Both literary studies[9, 10] have determined that while they are good for small pipes, their repetitive movement cycle leads to fairly slow robots in comparison to other methods.

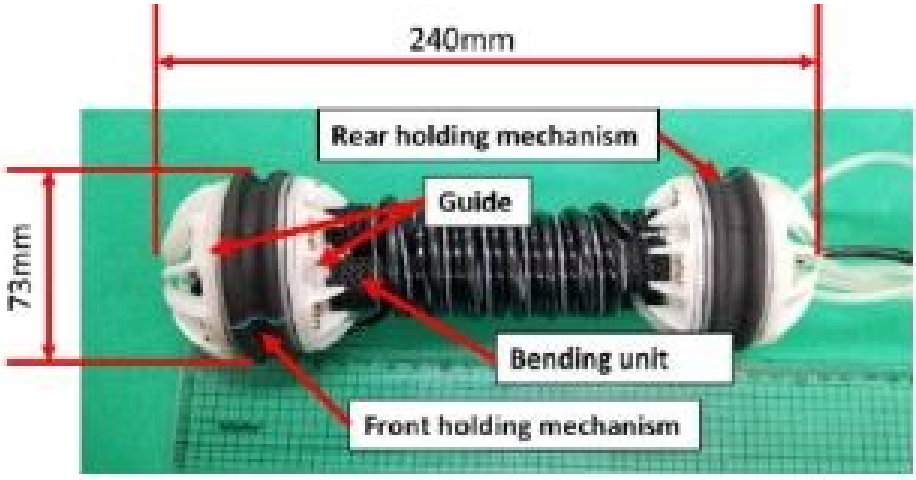


Figure 2.4: Hayashi et al. pneumatic inch worm

Screw Type

Screw type robots rely on a rotary motion to propel the robot as well as steer it. In some cases it can be the entire body such as the design from Kurata et al. [17] or part of the body can act as a rotor such as the design from Ren et al. [20]. In the case of the full body helical robot, the design was inspired by microorganisms that move using a flagellum. The helical motion is meant to roll and spin so that the motion at the contact points of the pipe walls will propel it forward. Figure 2.5 show its body which is comprised of a spring that had small solid segments in it where the cables used to pull the spring into shape were threaded through. The cables run through the segments in a helical path so that when pulled, the spring body would create a spiral shape. The pitch of the helical path was important to determine the type of motion the spring would have. The design only got as far as demonstration of motions in a pipe.

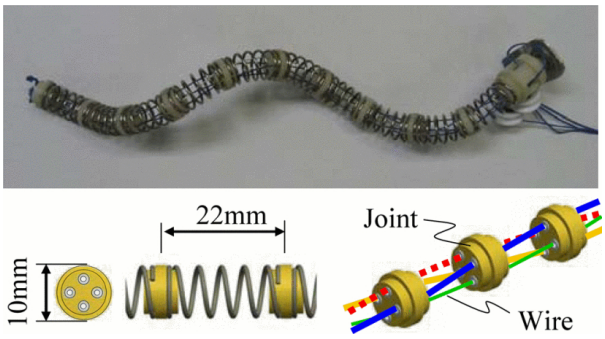


Figure 2.5: Kurata et al. bio inspired screw robot

The robot that had a rotor had a more complex structure, but has a been shown to be very effective. It is comprised of 4 main part which are the power unit, central unit, variable-pitch unit and the driving unit. The central unit keeps the robot centered with the pipe and has passive wheels that press against the pipe walls. The driving unit has passive wheels and is spun by a central driving motor in order to create propulsion. In order to have these wheels at the proper angle to create forward propulsion and turning, the variable-pitch unit has linear stepper motors that pull on struts connected to them. At certain pitch angles, the robot can also go faster. This design has been proven to be effective as they were able to go through a 135° bend within 7 seconds. Both literary studies[9, 10] believe that these types of robots excel at the traversal speed and navigation capabilities. Due to how specific they are to circular pipes, they might not do well in unstructured environments and one study believes they might even damage the pipe they are in with their high speed movements.

Legged Type

The final novel movement type is the legged robots. Typically legged robots are designed for open spaces. Designing legged robots is a very complicated process, even for basic environments such as a lightly unstructured one. Putting them into a confined space like a pipe adds another layer of complexity. In the literary studies [9, 10], most work done has only been simulation based. Both works by Savin et al. [19, 21] detail the necessary motion requirements for a robot to operate within an environment as well as navigating them. The first work by Savin et al.[19] breaks down the necessary leg trajectory for a six legged robot. This particular had 6 legs with 3 links each. Determining kinematics and pathing of each leg in this unique environment is extremely difficult as shown in Figure 2.6. His next work[21] goes further and calculates the RRT motion planning algorithm that allows the robot to navigate the pipe. In both works the tests are purely simulation. Both literary studies[9, 10] agree that a walking robot, while having good steer ability and speed when implemented well and in normal circumstances, they are too needlessly complicated for something like pipe inspection.

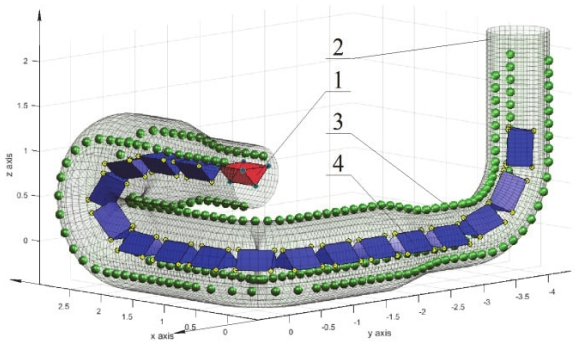


Figure 2.6: Savin et al. robot path diagram

These different novel methods have tried to shed the conventional method of wheels in favor of trying a potentially more efficient way of navigating pipes. There are strengths and weaknesses to all methods and it was important to understand them in order to make a design the maximizes these strengths and minimizes these weaknesses.

2.3 Soft Robotics Lab

The final section of this chapter focuses specifically on the work that the Soft Robotics Lab has completed. As CLARA is mainly based on the previous work from this lab, it is important to review the technology that has been previously developed. This section will first describe the Yoshimura module and then move on to the mobile robots that incorporated this module.

2.3.1 Yoshimura Module

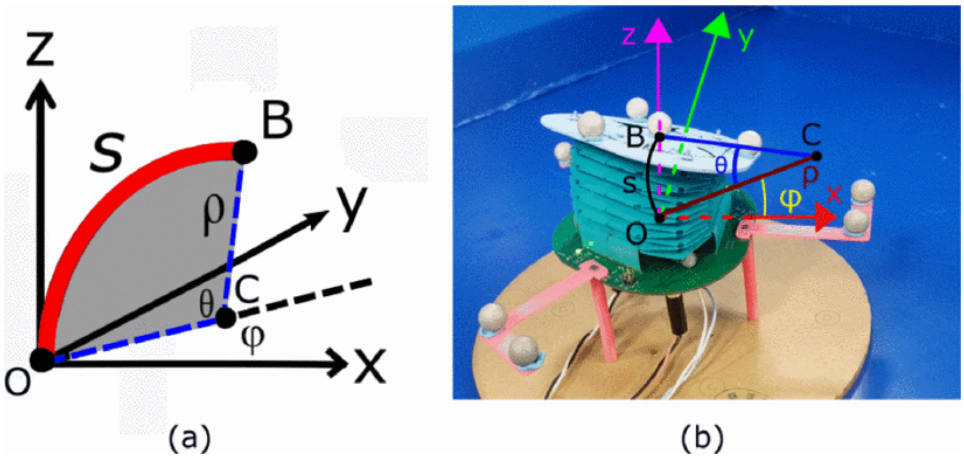


Figure 2.7: Santos et al. Yoshimura module diagram

The Yoshimura module is based on the origami pattern of the same name. The Soft Robotics Lab used this pattern by creating a module from a thin PET (Polyethylene terephthalate) sheet that has a pattern laser cut into it. From there it is folded to mirror the aforementioned origami pattern. This module can have a number of side, but traditional Yoshimura patterns are triangular in shape. At the vertices of the module are through hole in the bellows which cable is run through. One side of the module has a plate where the cables are anchored to and the other side has a plate where motors are attached to and wind up the cables. With three cables actuated like this, the module has a 2 DOF movement and can move in the axial direction as well. This movement is similar to the inchworm robot[18] and bio inspired helical robot[17] to a certain degree. Santoso et al.[22] wrote the initial paper that described the Yoshimura module. In their work, they detail the forward kinematic equations to find the distal point B and the inverse kinematics equations to determine cable length. Both can be found through these physical parameters of the module (ϕ, κ, s) and the values derived from them. Figure 2.7 shows the general diagram of the module while it is partially actuated and the kinematic parameters associated with that configuration. The distal point of the module is located on the anchor plate and is where a second module would be connected to create a continuum module. The following equation is used to determine the physical position of this point relative to the reference frame of the module:

$$[\rho\cos(\phi)(1-\cos(\theta)), \rho\sin(\phi)(1-\cos(\theta)), \rho\sin(\theta)]^T \tag{2.5}$$

where θ and ρ are found through their relationship with s and κ. The cable lengths are found from the following inverse kinematic equations:

$$l_1 = 2n\sin(\frac{\kappa s}{2n})(\frac{1}{\kappa} - d\sin(\phi)) \tag{2.6}$$

$$l_2 = 2n\sin(\frac{\kappa s}{2n})(\frac{1}{\kappa} + d\sin(\frac{\pi}{3} + \phi)) \tag{2.7}$$

$$l_3 = 2n\sin(\frac{\kappa s}{2n})(\frac{1}{\kappa} - d\cos(\frac{\pi}{6} + \phi)) \tag{2.8}$$

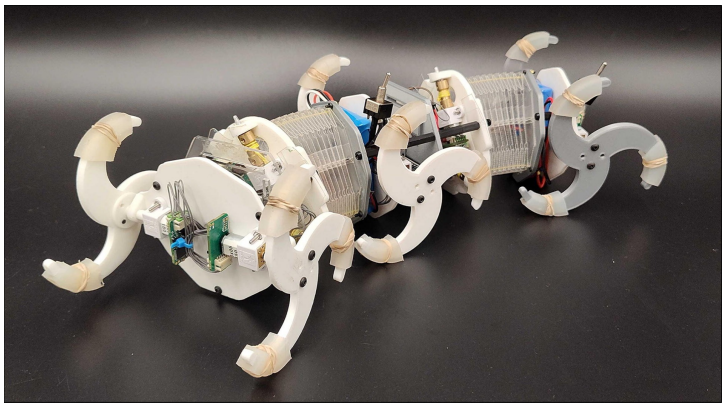
This work was the basis for all Yoshimura module work going forward including the mobile robots.

2.3.2 SRL Mobile Robots

The Yoshimura module has been used in a number of applications including continuum arms, quadrapeds, and ariel drones. Traditional wheel-based mobile robots are among those applications. The first mobile robot utilizing the Yoshimura was the tethered Salamander Bot[23] . This first attempt included a single Yoshimura module with the actuation plate containing 3 N20 micro metal gear motor that wind the cables. The other plate acted as the anchor plate and contained 6 silicone wheels with 2 at each module side. Using two different transmission systems and a single motor that was housed in the module that turned all 6 wheels in order to drive the module backward and forward. The initial test results showed it was able to navigate a simple curved maze, reach a max linear speed of 303.1 mm/s on low friction surfaces and go up inclines of up to 60°.

The natural evolution was the Lizard Bot[24]. This version had two Yoshimura modules in sequence, was untethered with its own power supply and wireless microcontroller for each module, and had motor driven whegs for obstacle traversal. Adding another module allows for more complex configurations and flexibility as they have their own set of actuation motors and operate independently of one another. Being untethered allowed it to move freely from the operator and the wheels allowed it to traverse up obstacles up to 132 mm in height. The fastest speed was on carpet at 39.2 cm/s, a much faster speed than the previous mobile robot when considering the surface. As well as the mechanical upgrades, Lizard received a motion planning algorithm called CHOMP+ that allowed the robot to easily and autonomously navigate unstructured environments by mapping out a path around observed obstacles.

The last mobile robot this section will talk about is CLARA itself. CLARA[25], originally called Continuum Locomotive Alternative for Robotic Adaptive-exploration, was conceived from an MQP project. The original designs included the original lead screw variable suspension system, mainboard that connected the Tiny Pico to the SIMC, and the SIMC designs themselves. The second team[26] improved upon the design by encasing the electronics, improving the variable suspension system by making it more stable and making the rear suspension system active instead of passive like in the original design, simplified the silicone wheels, and introduced the ESP32 camera to the system. This version was able to navigate a junction at a 45° and had a maximum bend angle of 150°.



(a) Jones et al. Lizard robot

(b) Schroeder et al. CLARA robot 2023

Figure 2.8: SRL Mobile Robots

Chapter 3

Design

This chapter will detail the mechanical, embedded system, and software design of CLARA. The first section will detail the project goals and overview of the design and the rationale behind these decisions. Next the exact details of the mechanical design will be covered including the inclusion of an additional Yoshimura module and what that entails, changes to the rigid section modules, and the change in the variable diameter suspension system. The embedded system section will detail the changes in the voltage regulator configuration and review the structure of the system and subsystems. Finally the software section will detail the changes in Arduino and python code including moving to an x-box gamepad controller and the updated SIMC firmware.

3.1 Project Goals & Overview

Considering the information in the , this section will discuss the goals for this thesis project. The original intent of the CLARA design was to combine the traditional wall press method with the flexibility of the Yoshimura module for a fast and flexible pipe inspection robot. MQP teams typically do not do rigorous testing of the designs they submit. The previous version [26] only states the maximum bending angle and that it can go through a 45° junction. This new version is meant to be more versatile in the types of junctions it can navigate by increasing its range of motion and this thesis is meant to prove the efficacy of the design and its updates.

This current version seeks to further increase its flexibility by doing two things. The first change is to add a second module to the design. The improvement from Salamander [23] to Lizard [24] showed that adding a second module had an impact on the types of configuration the robot can make and the overall maximum bending angle it has. Adding a second module to CLARA should similarly increase the flexibility. The second change would be to try to reduce the length of the rigid sections. The equations from Choi et al. [13] demonstrates that longer and wider rigid sections will have a harder time passing through elbow joints. A variable diameter suspension system changes the width of the rigid section meaning that its ability to pass through a junction is dependent on its constant length.

The next goal is to thoroughly test the capabilities of CLARA. The basic concept of using the Yoshimura module for a mobile pipe inspection robot has not been proven during the MQP process to an acceptable degree where the concept can be considered proven effective. Basic qualities such as speed, maximum traversable incline, and battery life need to be tested and determined. The final metric is to see how CLARA does in a practical test setting that involves traversing lengths of pipe and maneuvering through junctions over a prolonged period of time.

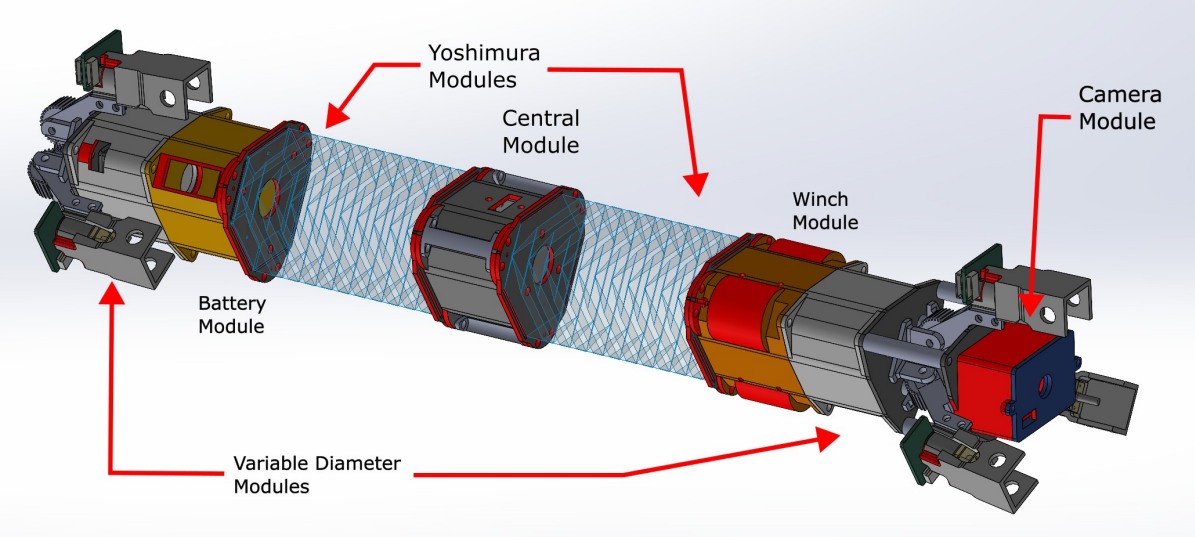


Figure 3.1: CLARA Section Diagram

Figure 3.1 displays the assembly sections that will be detailed in the upcoming sections. Additionally, Table A.1 in Appendix A contains the major components for CLARA. The rigid sections and links were made from PLA and Tough 1500 V1 for which the price is not factored in. The PCB and SIMC boards require a quote from a PCB manufacturer which will vary from manufacturer to manufacturer. Unit prices are suppliers are reflect the current time of writing. The gear ratio of the micro motor is dependent on what application the motor is being used for and range from 150:1 to 380:1. The unit price remains the same for all gear ratios at the time of writing.

3.2 Mechanical Design

The biggest update that CLARA received was in its mechanical design. The novel element introduced to this design was an added Yoshimura module and how it was implemented. The rationale behind this addition was to break up the rigid sections and add more flexibility to the robot. The second biggest change was the variable suspension system. Instead of a lead screw design, this version of CLARA had a worm gear style system with the wheels pointed forward instead of both sets of wheels pointing in opposite directions and away from the center.

3.2.1 Yoshimura Modules

The novel aspect about adding the second module is that it was actuated by the same cables that actuated the other meaning only one set of cable motors. Other platforms that utilized the Yoshimura modules had a set of cable motors for each module. The n variable in the inverse kinematic equations for the module represents the number of modules within the continuum. Due to the novel nature of this version of CLARA, this equation would no longer be considered valid. However due to the constant curvature created by the Yoshimura's mechanical properties, if each module is analyzed on its own instead of as a continuum, the each module's forward and inverse kinematic parameters can still be solved for.

Each module was created from 7 mil PET sheets that were laser cut into a specific pattern. For a triangular module, three folded parts comprise the entire module. Figure 3.2 is the template used for the Yoshimura modules on CLARA. Each part is 144.53 mm long and 61.3 mm wide with 22 creases creating a module with 11 bellows at approximately 98 mm long. Fully compressed, each module is approximately 35 mm long factoring the wires threaded inside of each module. If the triangular shape of the module were transcribed in a circle, it would be approximately 70 mm in diameter. This makes it slightly larger than the modules so that the cables can run just outside of the central module separating the two.

3.2.2 Central Module

The module that holds the Tiny Pico and mainboard PCB separates the two Yoshimura. Like every rigid module on CLARA, the central module is 3D printed from PLA material. Including the plates that are at the ends of each Yoshimura module, the height of the central module is 41 mm. A small slot on the side gives the operator access to the microcontroller for ease of flashing new software.

The lip of what would be the triangular shape's vertices are slots that the cables run through. All the rigid modules are slightly altered versions so they were originally designed for the smaller singular Yoshimura module. The cables are protected and held in place by small struts that run along the side of the module and are screwed in along with the Yoshimura modules. This keeps the cables from any potential pinching. Figure 3.3 shows the central module and the struts in the exploded view.

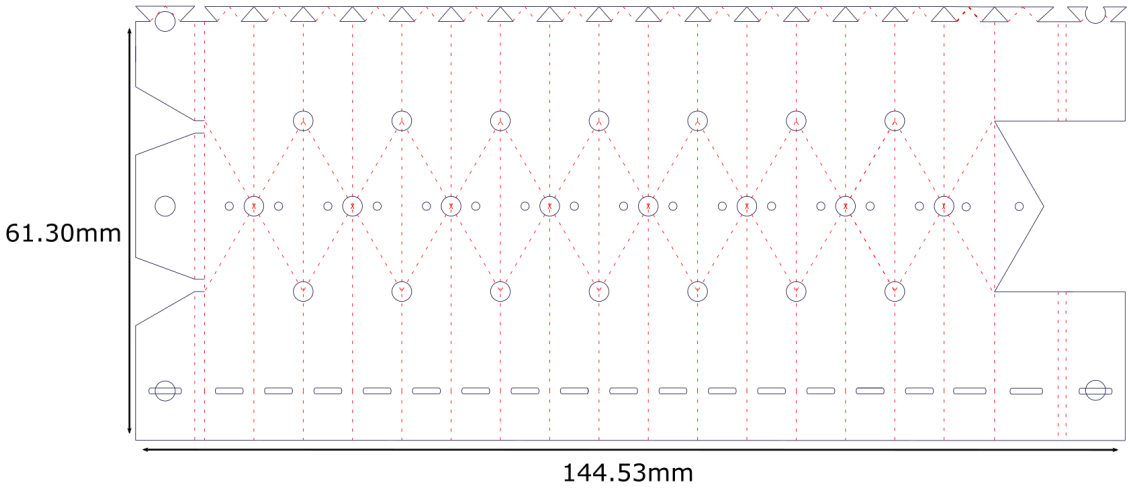


Figure 3.2: Yoshimura Module Component For Laser Cutting

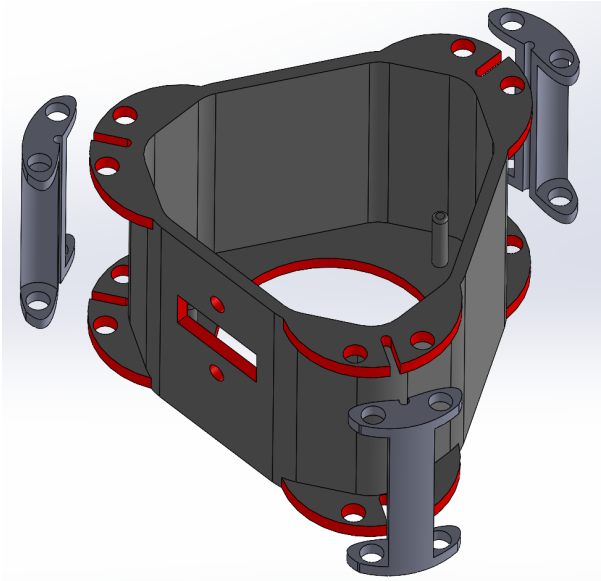
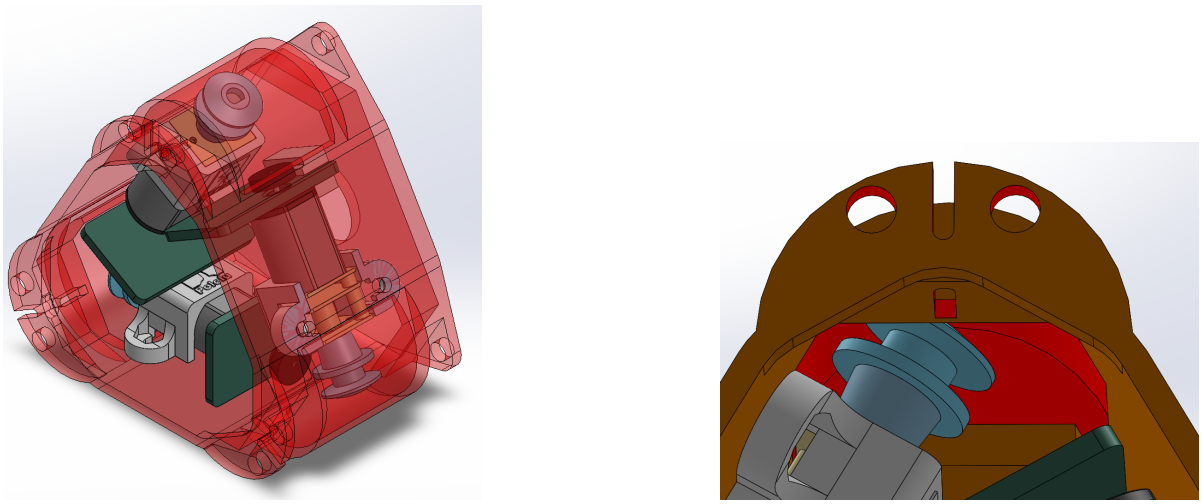


Figure 3.3: Central Module Exploded View

3.2.3 Winch Module & Pulleys

To compensate for the larger sized modules, small hollowed out sections for the winch module were added. The hollowed out sections were still small enough that they did not over overshadows the silhouette of the modules. The 3 298:1 gear ratio motors inside the module are positioned in a way where the pulleys attached to the motors poke inside these hollowed out section to ensure cable alignment with the Yoshimura module's through holes. The cables then run through small slots located on the walls and toward the edge. Figure 3.4 shows the full assembly with the composition of the motors and the inside view showing the hollowed out section and pulley piece.



(a) Transparent View (b) Inside View

Figure 3.4: Winch Module Assembly

The pulleys that are attached to the micro motors are slightly redesigned so that the ruts keep the cables from wrapping around the motor shaft instead of the pulley. Previous SRL mobile robots have had issues with the cables tangling and jamming due to this. The material was changed from resin printed material to PLA due to the fast printing nature and the fact that not much material durability is needed to be effective.

3.2.4 Battery Housing and Variable Diameter Motor Housing

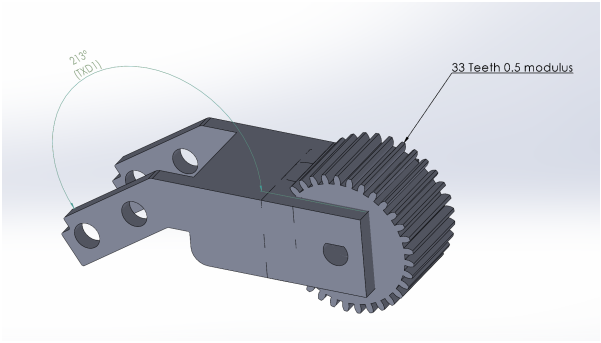
The rigid section designs that remained the most consistent from previous versions are the Battery Housing and Variable Diameter Motor Housing. Slots are put into sides to allow for JST wires and the lipo charging plugs to go through instead of a singular slot through the middle. Cable slots are added to the battery housing due to the it being the anchoring module.

3.2.5 Variable Suspension System

The previous versions of CLARA has a long lead screw that pulled and pushed the links holding the motor housing in and out. The issue with this design is that it made the rigid sections very long compared to the singular module and made the total percentage of the robot comprised mostly of rigid material. As mentioned in the related work sections, the longer a rigid section is, the more difficult it will be to pass through spaces such as elbows[13]. Reducing this rigid section length was one of the two major goals of this redesign.

Replacing the lead screw from the previous design, a smaller brass worm gear was used in it's place with a diameter of 9.7 mm and length of 20 mm as shown in figure 3.5 .The modulus for the worm gear was 0.5 and informed the design of the rigid links. This worm gear has a 3 mm d shaft through the middle which lets it secure tightly to the micro motor that drives it, which is housed in it's own simple rigid module. The gear ratio of this motor has either had a gear ratio of 298:1 or 380:1 through out the testing process. The wall press design of the robot means that most of the normal force is directed towards the worm gear motor. Higher torque motors are required to expand the wheels within a pipe and hold them in place.

(a) Brass Worm Gear



(b) Rigid Suspension Link

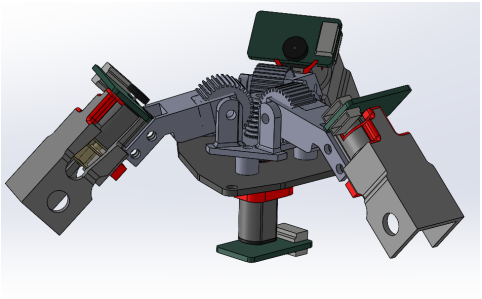
Figure 3.5: Variable Diameter Suspension System Components

Driving the worm gear turns the rigid links up and down in order to contract and expand the wheel. Figure 3.5 shows a picture of the link in Solidworks. The actual component is made from Tough 1500 V1 resin and has two parts. The basic link that has the small through holes that connect to the wheel motor housing and is angled at a 33 degree angle. This is so that the wheels can rest as close as possible to the body and still have the link avoid hitting the side of the module the wheels are pointed to. The second part of the link is the 0.5 modulus gear with a pitch diameter of 16.5 mm, a working depth of 0.25 mm, and 33 teeth. The gear is combined with the rigid link to make one complete piece that holds and moves the wheel housing when the worm gear is actuated. The link has a hole for a d-shaft through the middle of the gear which the link pivots on when driven.

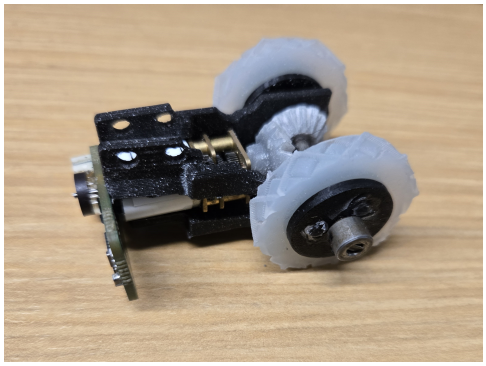
Originally both sets of wheels where pointed inwards to minimize the length of the rigid sections on the robot. This design change causes the width of the robot to increase but the overall length of the rigid sections to decrease causing a net benefit for maneuvering. However, during testing it was determined that pointing the front wheels forward allows for better control when moving through a joint. By having the points of contact with the pipe surfaces being at the front most part of the robot, it is easier to actuate the body and still have proper traction. This design change made the front rigid section have a similar length to previous versions, but is still within a reasonable range that allows it to move within the target pipes.

The links are spaced 120° apart and are located approximately 12.3 mm from the center of the plate using the middle of the gear as its point of reference. The d-shafts are held by a small plate that is situated 6 mm away from the base plate of the worm gear motor's module is located by small plastic struts. The height of the d shaft holders is 10 mm which allows the middle of the gear to hit the top of the worm gear. The gears need to properly mesh in order for the torque to fully and efficiently turn the links. The full assembly is shown in Figure 3.6

Determining the number of teeth on the link was a trial and error process. Originally the rigid links were split into two parts where the gear was sandwiched into a slot of the link. The rationale was that if one piece became worn, the other piece could still be used. Eventually I learned it was much easier fabricate and use if the two pieces were one. Originally, given the known dimensions and the chosen distance the center of the gear, the link was designed to have 30 teeth instead of 33. Testing the sub assembly showed that the gear could not mesh from that distance and was too lose. A 0.5 modulus gear at that size has very small teeth. Increasing the number of teeth slightly increases the pitch diameter by 0.5 mm so I tested out a number of gears until it was determined that 33 tooth gear has the best mesh and can operate without jamming or skipping in the worm gear. It was decided that the adjusting this part of the link was easiest as other aspects of the link such as the angle and length would have to be adjusted.



(a) Variable Diameter Suspension Assembly



(b) Wheel Housing Assembly

Figure 3.6: Variable Diameter Suspension and Wheel Housings

These links are attached to motor housing that holds the motor driven wheels. This involves a motor driving bevel gears to rotate silicone wheels made from Dragon Skin 10 NV, the same silicone from the previous CLARA design. This design is largely unchanged with the main change being the pattern on the wheels having small ruts and minor alterations to fit the SIMC better in the housing and a small added bracket to keep it in place. A assembled wheel housing without the holding bracket is shown in Figure 3.6. It is also important to note that the wheels had either a gear ratio of 150:1 or 298:1. During testing, the front wheels were shown to need more torque to properly get the first section of CLARA through a joint.

3.3 Embedded System

The embedded system is largely unchanged. The mainboard is still bare bones in that it simply connects the SDA and SCL pins of the Tiny Pico with $4.7k\Omega$ pull-up resistors and a voltage regulator that safely steps down the voltage from the battery source to acceptable levels for both the Tiny Pico and the SIMC boards. The LiPo batteries are still 7.4 voltage batteries but changed from 3 250 mAh to 2 400 mAh batteries for a slight 50 mAh increase and slightly more room in the housing. The main change was a work around for a power discrepancy with the SIMC boards.

3.3.1 Embedded System Organization

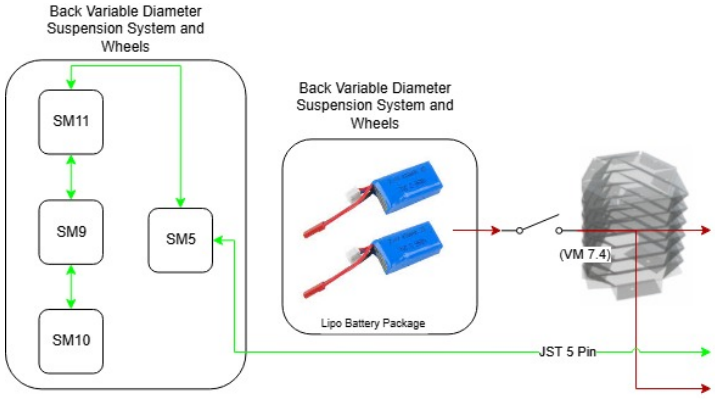


Figure 3.7: CLARA Back System Diagram

The overall structure of the system has not changed from the previous version. The central PCB shown in Figure 3.8 is located in the central module discussed in the mechanical section. The main board PCB which houses the Tiny Pico micro controller and the voltage regulator used for powering the Tiny Pico and SIMC boards. Every connecting JST and power cables come out of or through the central module. The back portion of CLARA shown in Figure 3.7 has the back motors and batteries housed in it. The structure has the JST cables running through the battery housing.

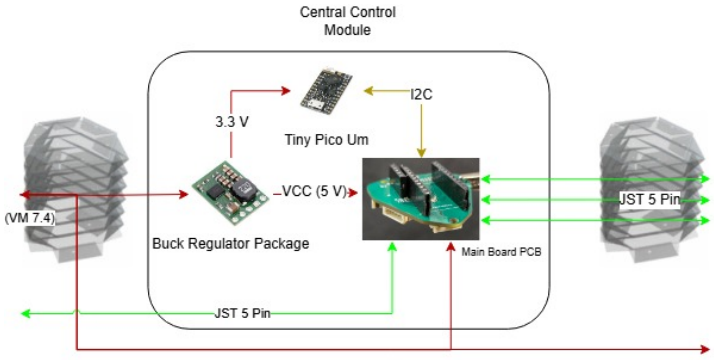


Figure 3.8: CLARA Central System Diagram

The front portion of CLARA is more complex with the majority of the motors being located in this section. Along with the JST cables, a power cable connecting directly to the LiPo batteries is run through to connect to a 5 V voltage regulator which was housed in the Variable Diameter Motor housing. This is used to power the ESP-CAM which is a system that runs in parallel to the PCB main board. Figure 3.9 shows the organization.

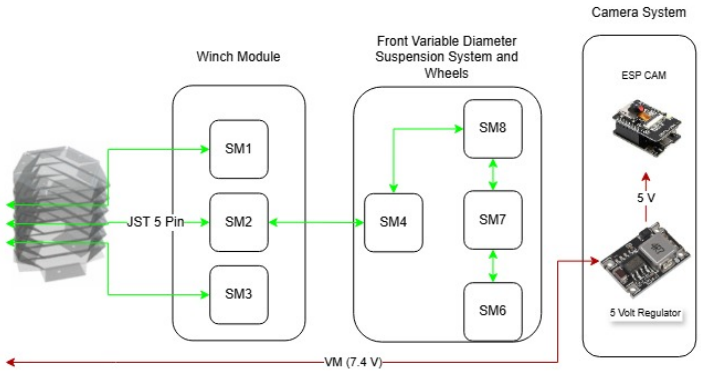
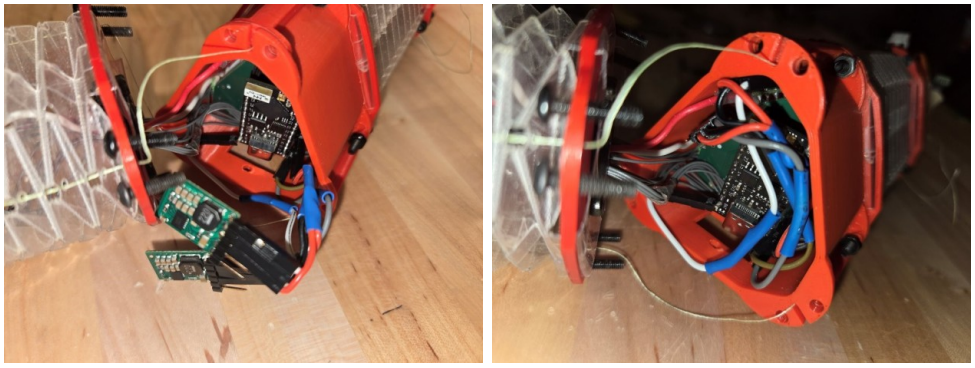


Figure 3.9: CLARA Front System Diagram

3.3.2 Voltage Regulator Work Around and new SIMC boards

Due to a change that is still yet to concretely be determined, sending a 3.3 voltage logic level to the SIMC boards made them unreliable and prone to shutting down without warning. Through experimentation, it was found that a 5 voltage logic level results in the SIMC boards becoming stable. There was however an issue with this as the voltage regulator output is for both the Tiny Pico and SIMC boards. The battery pin on the Tiny Pico board can only handle up to 4.2 volts. If a 5 V regulator was swapped in for the 3.3 V regulator, it would supply too much voltage to the Tiny Pico and damage it.

The voltage regulator that was originally on the main board was a Pololu 3.3V, 1A Step-Down Voltage Regulator D24V10F3. This was a 5 pin voltage regulator with a shared common ground pin, VIN, and VOUT pin. The 5 volt variant from the same family was chosen for the logic level of the SIMCs. The issue was using both on the same board. The solution was to connect the regulators in parallel with split jumper wires connecting the VIN and GND pins to where the regulator would rest on the main board and regular jumper wires to where their output needs to go. In the case of the 5 Volt regulator, the VOUT pin was connected to the main board as it was what connected to all 11 SIMC motors. For the Tiny Pico, the battery pin was carefully removed and replaced with one that was bent outward and not connected to the main board. This is so that the 3.3 volt regulator was able to directly connect to the Tiny Pico and the Tiny Pico was still able to rest on the board and provide the I2C signals for the SIMC boards. As shown in Figure 3.10, both regulators were carefully tucked into the control module to keep them from freely moving around.



(a) Outside Central Module (b) Inside Central Module

Figure 3.10: Central Voltage Regulators in Parallel

The potential issue with the SIMC is still yet to be determined. The only known difference between the current and previous ordered batches of SIMC boards is that the current batch uses the ATMEGA328P-AU chip instead of the ATMEGA328-AU chip. The only noticeable difference between these chips is that the ATMEGA328P-AU chip was designed to be more power efficient. Originally, the first CLARA team designed the board with this chip in mind[25] but needed to adjust their design for the ATMEGA328-AU due to a supply chain issue. These chips should have similar specifications, but it's possible that the minor differences might make them unstable with the current design. Other issues might be due to manufacturing issues or other damaged components. Time and budget constraints necessitated continued use of the current batch of SIMC boards. If future designs were to use this batch of SIMC boards, CLARA would need a redesigned main board that holds both regulators on it.

3.4 Software

The overall control scheme of CLARA remained the same but the functionality of the SIMC and gamepad were tweaked due to new firmware and outdated python libraries. Since the first CLARA redesign, the SIMC boards have been utilized in the other SRL mobile robot project, Lizard[24]. That project had the firmware updated considerably for ease of use when programming the Tiny Pico micro controller. Although this thesis did not program the firmware, this report will briefly detail the changes as they have heavily effected the arduino code of the Tiny Pico. Next the gamepad code had switched libraries from using the generic hid library which allowed 3rd party controllers to Xinput. I will also detail new commands that were implemented to give users the choice on how to operate the robot.

3.4.1 Gamepad & Transmitter Code

The operator directly controls CLARA through an x-box gamepad controller and ESP32 transmitter. Both connect to a PC workstation that the operator is stationed at. From here the commands are transmitted and the video stream from the ESP32 CAM and any motor sensor data is received. The x-box gamepad code is written in python B.1 and uses the serial terminal to transmit a 13 bit binary object that represents the button combination currently being pressed. The first bit represents a non idle game pad state, the next three bits represents gamepad direction, and the final bits represent each of the various face, shoulder, and trigger buttons. Bit wise functions create the object based on the gamepad state, encodes it using ascii encoder and sends it through the serial terminal when it is determined that the current state is new. A small delay is added after the serial write line to allow time for the serial write function to be processed.

The transmitter code B.2 is written in C and flashed via the arduino IDE to an ESP32. Changing the binary object into a character string and checks which character is a 1 or 0 to determine which command needs to be sent based on a specific combination. Certain commands take precedent over others due to an if-else . For example actuating cables using the d-pad will send a command before any other inputs can be read. This means that CLARA can not actuate its body while moving forward unless you switch inputs very quickly before the SIMC motors can be set to idle. To ensure that the stop command is sent immediately, the first check in the code is to check if the controller is idle which sets the motor speed to zero.

Once it is determined which input the user is preforming on the controller, the transmitter sends out signals via ESP-NOW to its registered peer. The only time a signal is transmitted is when a change in input is detected. The signal sent is a simple integer which the receiver will interpret.

The basic format for both the gamepad and transmitter code was created from the previous iterations of CLARA. Adjustments that were made included swapping out the generic hid library with the x-input library due to outdated versions of the pervious library and only writing to the serial bus when the controller state changes. This is so that the transmitter signal would not constantly prompting the receiver with commands. Instead CLARA continues doing the last operation it was given until told otherwise.

3.4.2 SIMC Firmware & Receiver Code

In order to consolidate the functionality of the SIMC boards so that any project may use them, firmware was created to simplify the process of creating base code using the boards. This includes moving all the communication functions from the base code into libraries. These libraries breaks down the read and write I2C functions into separate files as well as how it handles addressing for these functions. The information is written in struct and class formats to compile all the relevant values such as the PID values, address of the motor, and position and velocity functions. This makes utilizing the motors in the main function simple.

The receiver code on the Tiny Pico B.3 was also adjusted from the previous iterations of CLARA. The smart motors and sensor data structs are declared in the code setup and are separated into corresponding arrays of cable motors, front wheels, back wheels, and suspension motors. The sensor data structs are declared for sending back current and position data from the SIMCs via the ESP-NOW. The transmitter has similar data stucts declared to properly receive the signal based on the memory size of the struct. The Tiny Pico is structured event based instead of having the loop run commands in the same way the transmitter is. A simple if-else block checks for which integer is sent when the Tiny Pico receives a signal via ESP-NOW. Each if-else statement has the set RPM command for their corresponding motors. It was determined that this was the best method to use as precise position control was not needed to operate CLARA with a game pad. Certain commands take precedent due to the if-else format. The first thing the Tiny Pico checks is the command to halt all motors to reduce processing time due to the idle state being the most frequently changed to state.

Certain commands were added from the base code to try to streamline operator controls. The trigger and d-pad combinations allows more than one cable motor to be actuated simultaneously. Using only the trigger inputs will compress and decompress motors at the same time for a more consistent straight shape during operation. The same type of commands are used to expand and retract the wheels at the same rate. Table 3.1 contains the specific commands for these expanded movements. Actuating cable 1 and 3 creates an upward motion, 1 and 2 a right downward motion, and 2 and 3 a left downward motion. The original commands were left unchanged.

Table 3.1: New CLARA Commands

Bend Upward	Bend Left-Down	Bend Right-Down	Body Com-press	Body Expand	All Wheels Expand	All Wheels Retract	Current Check
RT+LT	D-pad Down+LT	D-pad Down+RT	RT	LT	RT+A	RT+Y	X+other

3.5 Differences From Previous Version

One of the aforementioned goals of these design changes is to increase flexibility and reduce the length of the continuous rigid links. Below is a table to compare the current version of CLARA with the previous version. This includes the lengths of each section, size of the Yoshimura modules, percentage of flexible material the robot is comprised of, and the size of pipes the robot is able to traverse through. The values recorded for the previous version were taken from physical measurements and information from the previous MQP report[26].

Table 3.2: CLARA Dimensions

Version	Front (mm)	Module(s) (mm)	Module Dia (mm)	Middle (mm)	Back (mm)	Total Length (mm)	Flexible Body %	Operable Pipe Dia (in)
Previous	170	130	60	N/A	202	502	25.9%	4 - 5
Current	145-166	98	70	37	100	478-499	39.3%-41.0%	5 - 7

Both robots are similar in size with the difference being the percentage of flexible material that CLARA is comprised of. The current version has 39.3% of its body made of the Yoshimura module material, which is almost a 15% increase from the previous version. Depending on how far the wheels expand, the front rigid link length can change to a minimum of 145 mm. The diameter range of the variable suspension system doubles, but loses the ability to traverse 4" pipes. Considering these values, the design changes satisfy the goal of increasing flexibility through the increasing the flexible body percentage.

Chapter 4

Testing & Results

This chapter will detail the testing process and results for evaluating the new CLARA design. These tests include the kinematics verification test, maximum bend angle & rate of change, speed test, elevation test, battery life test, and joint maneuverability and final maze test. The second goal of this thesis is verifying the efficacy of the design concept by focusing on practical testing for CLARA. Previously CLARA had minimal testing done as the focus for the MQPs had been on the design and implementation aspect of engineering. This meant that any concrete data regarding the efficacy of this design was minimal. Determining how well this version of CLARA does in a practical setting will determine if the concept of combining the wheel press model and the novel concept of using a flexible origami body could be viable option for pipe inspection work.

4.1 Kinematics Verification

The first test preformed was to verify CLARA's kinematic model. The kinematic model for the Yoshimura module has already been established[22], but due to the novel concept of using the same set of motors for two separate modules it was not clear if that same model can be applied to CLARA. The initial assumption was that both modules would have similar if not the same main kinematic values (κ, θ, ϕ) and all secondary values associated to them that are described in the model. Due to a rigid section in between the two modules, there is an expectation that friction would compress the back module. The back module is where the cables are anchored which makes it the module that experiences the full cable tension force unaffected by friction. In order to verify this a kinematics test was preformed.

4.1.1 Test Setup & Data Processing

Original kinematics tests for the Yoshimura module involved using the Motive motion capturing system and tracking the position of both the base plate and the top plate relative to each other (Satnoso et al. [22]). The same principal was used here where both modules plates were measured relative to one another and each module was analyzed separately from one another. It was determined that the plates for each module that were furthest away from the motors were to be considered the base and the ones closest were to be considered the top. Due to how CLARA is structured, the distal point would be considered towards the front of the mobile robot which are where the motors are located. The choice of which is considered the top plate and where the distal point is is arbitrary as there is no plate that is fixed in the global frame. The inherent properties of the module and how it maintains a constant curvature in any configuration means the only real difference in values would be ϕ due to it explicitly being defined by what the local frame of the module's base is. However, both modules need to have an agreed bottom plate and local frame orientation to properly compare ϕ values.

The test setup involves setting CLARA on top of a table in the middle of the testing room. The Motive motion capturing system has a series of cameras pointed toward the middle of the room. While running, the system tracks small reflective spherical markers as they move relative to an origin plate containing the same markers. These markers are 12 and 16 mm in diameter and were stuck to the vertices of each triangular plate. The vertex that corresponds to cable 2 is near the ground, so the marker was placed on the side opposite of it where the altitude point would approximately be. CLARA was put into different pose configurations and time stamps were recorded for each pose. The raw data of the marker coordinates were processed by a MATLAB code I had developed that would determine all the relevant parameters.

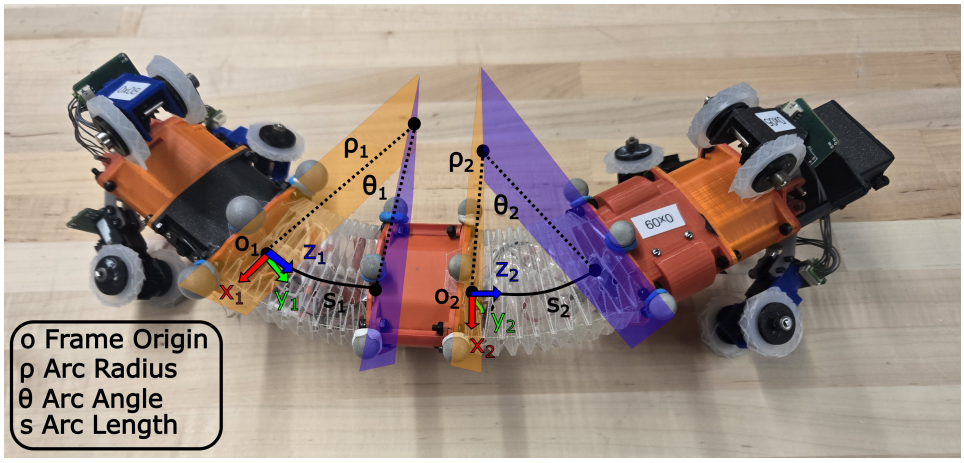


Figure 4.1: CLARA Yoshimura Kinematic Diagram

The first step of processing the data was to set the markers to where the cable through holes were. This involved finding the altitude vector for marker 2 as defined by the rigid body triangle it makes with markers 1 and 3. The coordinates for where that tracker would normally be was found by moving it the length of the altitude, 62.5 mm, along this vector direction. Once this coordinate is found, the same process is done for all the trackers to move them 5.02 mm to the through hole position. The plate is based on an equilateral triangle with the through holes being located on the altitude of each vertex. The size of the trackers were considered in all instances of moving the position of the marker coordinates. The new coordinates of the markers were used to define a plane in the global coordinate frame. This includes finding the normal vector (a, b, c) via the cross product of two vectors calculated from the markers and the d value of the plane. Finding the $-d$ value involves finding the centroid of the plate via averaging the coordinates and then preforming the dot product of this coordinate and the normal vector. The points on the module's arc are located at these centroid points so they will be used for any planar calculation going forward.

The next step was to find the plane on which the arc is located. First the intersection vector of the plate planes is found using the cross product of the plates planes' normal vectors. This vector is considered the normal vector for the arc plane. Next the $-d$ value is found by plugging in both centroid coordinates to the dot product function. Due to the imperfect method of tacking the markers onto the plate, different $-d$ values are found when plugging in the different centroid values to the dot product equation. To account for this, the d values were averaged. After defining the arc plane, the center of the arc was calculated by using all three plane equations for a systems of equation. From there the parameters for both forward and inverse kinematic models were found..

Using the center point and centroids, the radius ρ and arc angle θ were found. The ρ value was averaged similarly to the d value which means the error is compounded. The θ value was found using the two vectors found going from the center to the centroids. Next the arc length s and curvature κ were calculated. Finding the plane transformation angle ϕ is a more involved process as it requires a local coordinate frame to rotate. In this case the base plates will always be considered where the local frame is originated. A rotation transformation matrix was created by choosing the vectors going from cable 3 to 1 as the x-axis and the vector going from where marker 2 is to where it was adjusted to is the y-axis. The cross product to these vectors is the z-axis which is pointing into the module. The origin is the centroid of the plate. Figure 4.1 shows how these parameters would be represented on CLARA. It should be noted that the angle of the diagram does not show the ϕ angle, but it can be assumed that it's value is near π for this example.

Converting the center to the local frame allows the angle between the x-axis and the vector to the local center to be calculated. This value is ϕ. The coordinate value of the local center coordinate was considered in the calculations to account for direction. The next step is calculating the cable lengths of each module. Each module's inverse kinematic values are calculated using their own forward kinematic parameters. Cable 2 is found using the l_1 equation and the following cables in the clockwise direction correspond to the next 2 equations. Summing the cables and the length of the rigid section gives the total cable length.

4.1.2 Results

To process the data further, the lengths of CLARA were plotted in 3D to provide a clearer picture of the results. This includes the rigid sections, the Yoshimura arc and the cable lengths for each module. The plots were calculated parametrically using θ, arc center, plate normals, and centroids. The plotted arcs began at the bottom centroid towards the direction of the top centroid and the cable arcs where calculated by translating this equation to where the through holes are located. Markers were placed on the plot that represented the tail and nose of CLARA, and the simple and parametrically calculated plate centroids. The simple plate centroids are the coordinates calculated initially when processing the data and defining the equations for the plate planes. The parametrically calculated centroids were found when plotting the arc lengths. The difference in these points represent the absolute error in calculating the values of the whole system. The error stems from the imperfect method of tacking the markers onto the plates. This resulted in the d value for arc plane to be an averaged value which informed every value afterwards including the key kinematic values. Assuming that the true value is closer to the simply calculated centroid, the average absolute error out of 40 different pose configurations was 5.09 mm with the largest error being in a pose being 18.14 mm. A standard deviation for this error is 4.67 mm meaning the error population is relatively close to the average value.

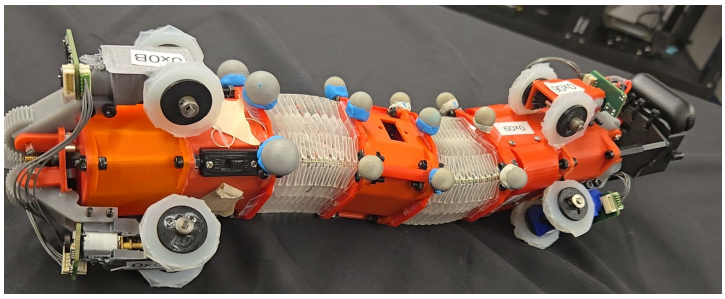
Figure 4.2 is a side by side comparison for one of the poses gathered. The results of the graphs and processed data show that there is a significant difference in their forward kinematic values, specifically their respective ϕ values. Even when looking at it relative to their base plates, the ϕ values not only have a standard deviation error of 1.90 radians but also often were in opposite directions. This created configurations that were closer to a non-monotonic spline made by 2 constant curves separated by a rigid section.

Table 4.1 shows the average difference and standard deviations when comparing the kinematic parameters of the two Yoshimura modules. The values from the back module were subtracted from the front module's values which means a positive value means that the front module's value was larger than the back's value and a negative value means it was smaller. The distribution of the 40 samples can be shown in the following Figure 4.3.

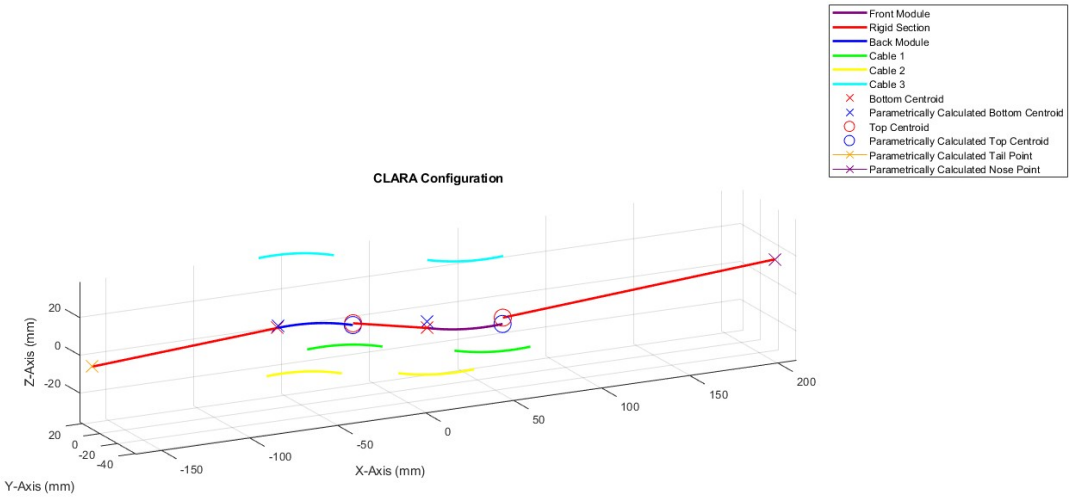
Table 4.1: Yoshimura Kinematic Statistics

	ρ	θ	s	κ	ϕ	$l1$	$l2$	$l3$
Avg Δ	-13.63 mm	0.20 rad	4.28 mm	0.0030	1.56 rad	4.72 mm	10.05 mm	4.55 mm
σ	110.77 mm	0.27 rad	5.24 mm	0.0039	1.90 rad	6.47 mm	12.30 mm	6.83 mm

Outliers were removed from certain data sets with. The ρ data set had 3 outliers removed and the s, κ, and cable 2 data sets each had one outlier removed. The distribution is skewed to the front module having larger kinematic values than the back module. This is most notable in the arc lengths for the module (s) and the cables ($l1, l2, l3$). The arc radius (ρ) and arc length ($s, l1, l2, l3$) are closer to a normal distribution.



(a) CLARA Configuration



(b) Matlab Representation Plot

Figure 4.2: CLARA Kinematic Test Sample

Due to the nature of the ϕ angle being along the unit circle, the front module tended to be further along than the back module as shown by the distribution graph. The value has most of the distribution between 0 and $\frac{\pi}{2}$ and most other values extending to 2π. Given the orientation that the base frame was chosen, it means that the front module was angled upward while the back module was angled downward more often than the reverse. This is supported by the fact that cable 2 had the largest difference out of the three cables and cable 1 and 3 having similar values. This could be due to how the samples were collected. The least notable values were the curvature and arc angle meaning the shape was relatively close to one another.

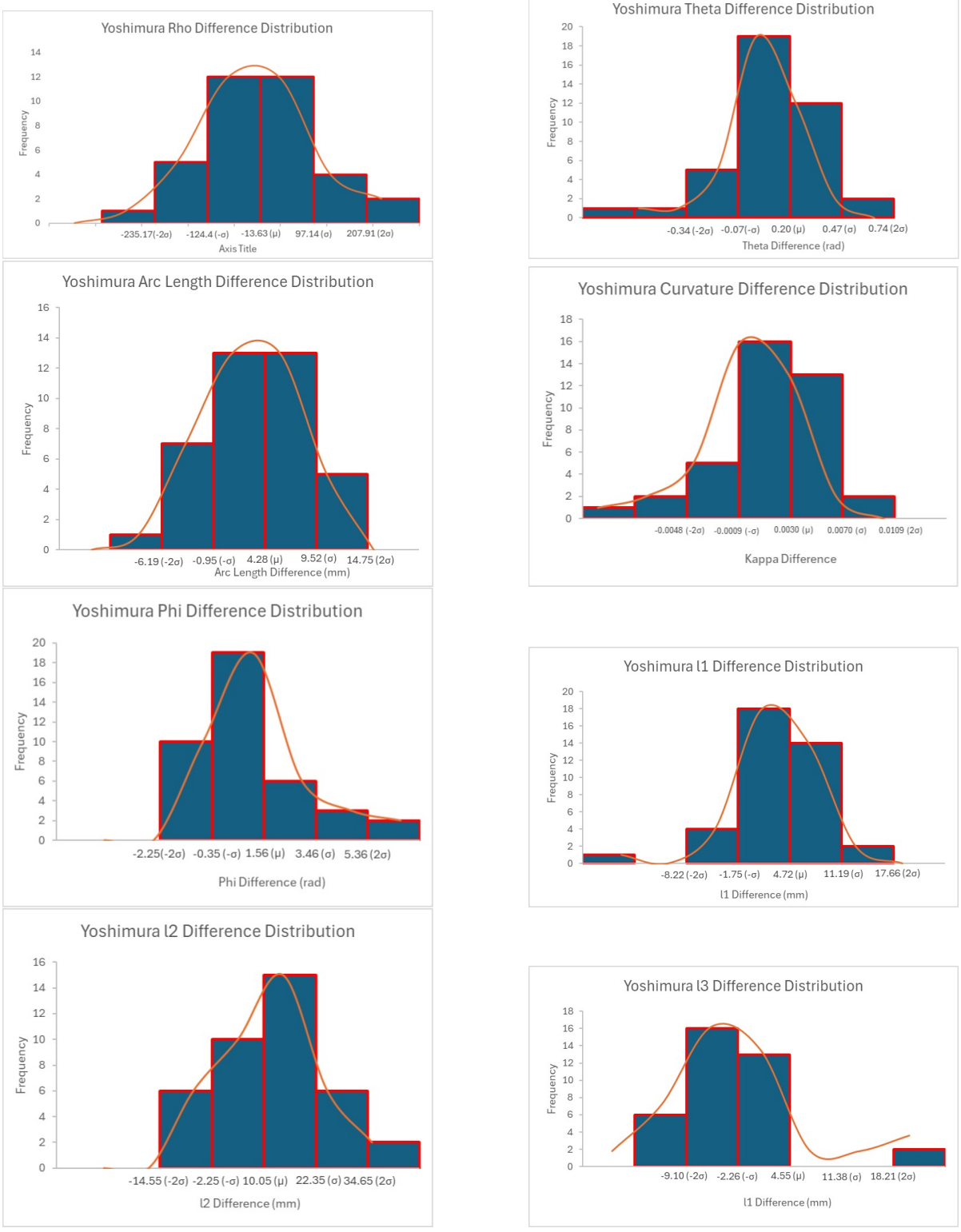


Figure 4.3: CLARA Kinematic Distribution Graphs

4.2 Max Bend & Rate of Change

This next subsection analyzes the maximum bend angle that CLARA is able to achieve as well as the rate of change in each modules bend during this action. This is information is relevant to the operator so that they know the maximum capabilities of CLARA as well as how to approach actuating the body.

4.2.1 Test Setup

The maximum bending angle was determined by setting a camera directly above CLARA recording while I actuated cable 1 until the motor stalled. The frame were this occurred was taken from the video and processed by physlet tracker program to determine the angle by using the Yoshimura plates as reference.

A separate video was recorded preforming the same action and processed via the Kinovea program to track the bending of each module. Actuating a single cable means that the module's bending arc is aligned with that single cable's bending arc and that they would exist on the same plane in 3D space. If the camera's view can be aligned with this plane, then an accurate estimation of the change in bend angle for the whole module without worrying about any change in orientation angle ϕ. Video will be taken of both the compression and release. Due to the shape of CLARA and how the wheels are attached, the body when straight will have the bending angle of cable 1 slightly angled and give a slightly inaccurate calculation of the bending angle. While it begins to compress, CLARA will naturally roll over and align itself to the camera and begin to give an accurate reading. CLARA was gently held still to keep it at this angle once rolled over and try not to influence the test. These factors are important to consider in order to properly interpret the data. The video is processed by the Kinovea program with trackers attached to the Yoshimura plates by using the heads of the screws as guiding points. 2 vectors for each module and the angle between them are calculated from this tracking data.

4.2.2 Results

Figure 4.4 pictures the maximum bending angle, which is 7° away from a full 180°. This was expected due to the limitations of a Yoshimura module's bending angle being around 90° when actuating a single cable.

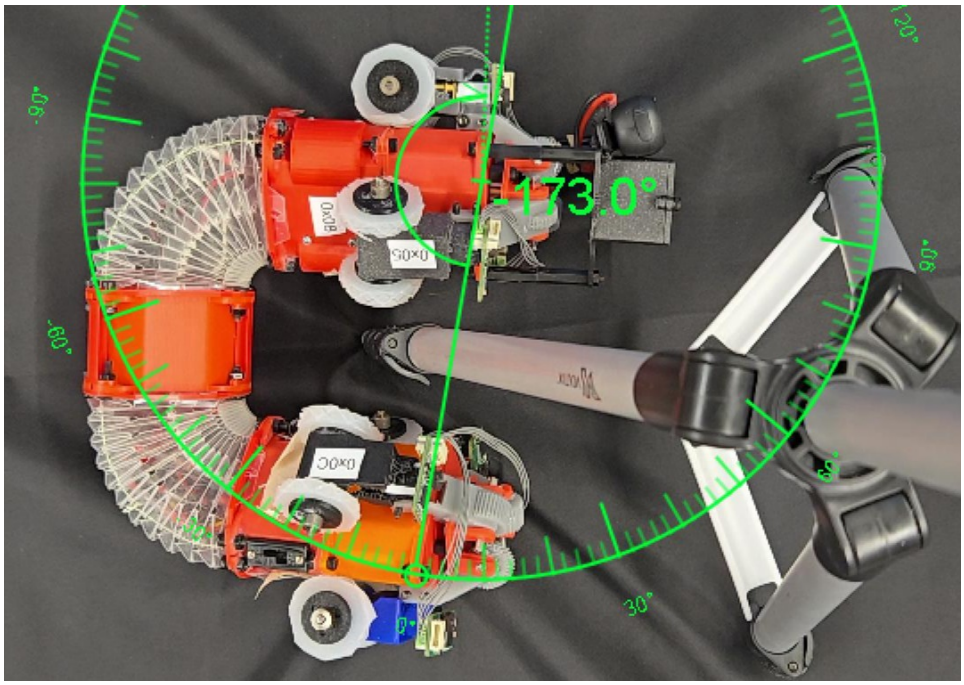
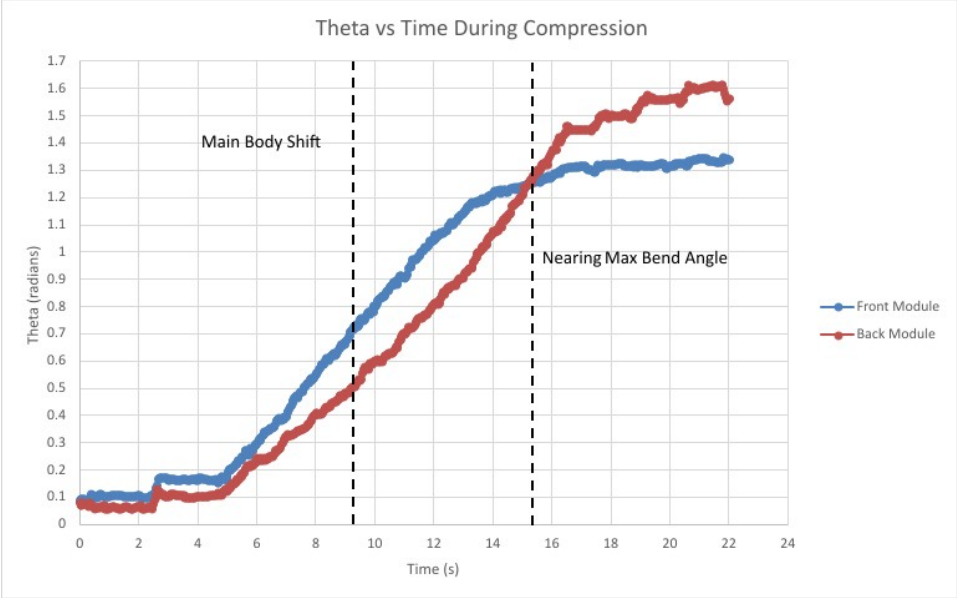


Figure 4.4: Maximum Bend Angle

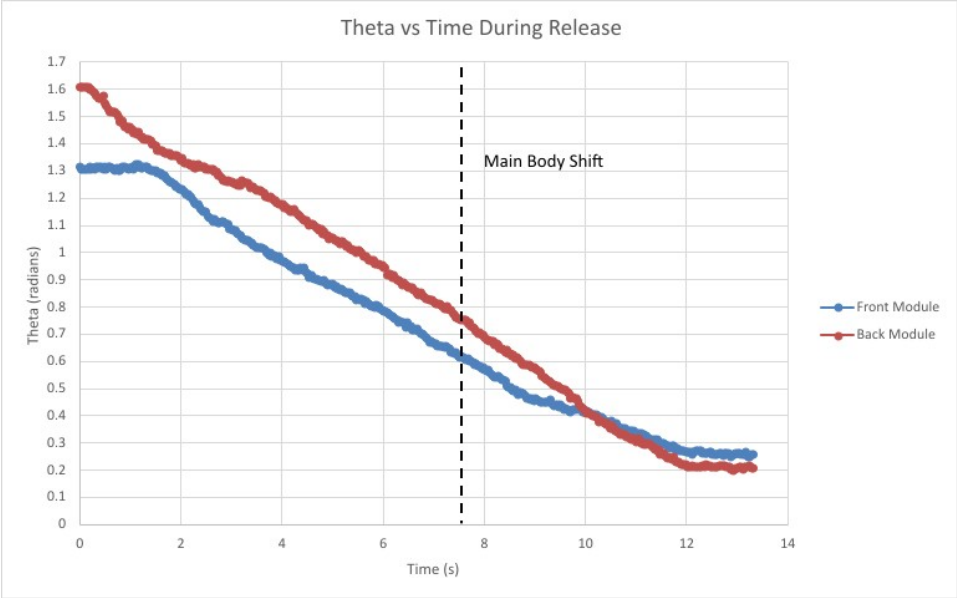
Figure 4.5 shows the rate in change of the module angles with respect to time. Both graphs have both modules in order to draw comparisons between their behavior. Notable results include the time it takes to compress and release, the rate at which both modules change, and the behavior at certain points in the actuation process.

Compressing to the max bend angle takes approximately 9 seconds longer than retraction. There are two reasons for this. The first reason is that it takes more force to compress a module than it is to release the tension. The natural spring force of the module allows the release action to be faster. The second reason is because at approximately 15.5 second I began to near the maximum bend angle and began to operate CLARA by intermittently actuating the cable. This is to avoid any potential damages to the motor or the cable. Due to this approach, the increase in bend angle happens in brief moments with static values in between. Taking this into account means that the true time difference between compression and release is likely shorter than 9 seconds.

For compression, the point between the main body rolling over and the max bend angle is when the camera is aligned with the module's arc. This is the region where the tracking is the most accurate to the true behavior of the Yoshimura modules. Both modules had approximately the same rate of change at $0.1\frac{rad}{s}$ and the front module having a larger theta value than the back module until CLARA begins to approach it's maximum bending angle. The time before the the body shifts, both modules had a compression rate of $0.75\frac{rad}{s}$. As noted previously, this is due to the body being at an angle to the camera and is not properly reflective of the true compression rate.



(a) Module Compression Graph



(b) Module Decompression Graph

Figure 4.5: Module Theta Rate of Change Graphs

The beginning of the graph shows a small peak that signifies when the cable starts to exert tension force on the module. This does not happen immediately as the cable started at a slightly lax position. As the module approaches its maximum bend angle, the front module remains static while the back module continues to compress. This may suggest that as CLARA reaches its final position the modules begin to adjust to remain at rest. Friction forces exist between the rigid module and the modules which may necessitate a larger compression force in the back module as it is the module connected to the anchor plate and experiences the full tension force directly.

For release, the back and front module have a release rate of approximately $1.25\frac{rad}{s}$ and $1\frac{rad}{s}$ respectively. This is due to the fact that the back module started in a more compressed state. The potential energy is higher in the module that is more compressed. The front module doesn't begin to release until approximately 1.5 seconds while the back module immediately starts to decompress when the cable begins to actuate. The back module has the same rate of change throughout and the front module begins to slowdown at approximately 10 seconds. Both stop decompressing at approximately 12 seconds.

4.3 Speed

The speed at which it travels is an important factor in a pipe inspection robots. As mentioned in the Related Works chapter, the some of the robots with a novel method of travel tend to be slower than traditional wheeled robots. The robots that had similar functionality to the Yoshimura module[18, 17] tend to have the slower speeds because they rely on the repetitive bodies motion to move. This test was to determine how effective combining the wheel press design with the Yoshimura module in terms of its speed.

4.3.1 Test Setup

It should be noted that these tests were done before the design decision was made to point the front wheels toward the mid-section instead of away like in the final design. These results should still be reflective of the final design. There were 2 different sets of speed tests preformed with CLARA the first set involved a 6 inch PVC dust collector hose that was stretched over the length of a table. Markers indicated the starting and stopping point of CLARA which is 1 meter in length. To fit on the table, CLARA was slightly compressed to a total length of 388 mm with the back and front modules compressed to approximately 38 mm and 47 mm respectively. This test was done before the decision to switch the wheel motors from ones with a 150:1 gear ratio to motors with 298:1 gear ratio and to run them at 60 RPM. The second set of tests involved a 5 inch PVC hose and with 298:1 gear ratio wheel motors running at 30 RPM. 30 RPM was chosen due to the belief that 60 RPM was beginning to overwork the battery and motors. The bevel gears for the second set were also switched out from being made from PLA to Tough 1500 V1 resin as it was found that PLA wore out very fast. The second test had CLARA compressed to 50 mm on both modules resulting in a total body length of 403 mm. CLARA was controlled to move 1 meter forward and then 1 meter backwards. It was assumed that CLARA will maintain a consistent speed going forward or backwards and that changing the gear ratio would not affect it.

4.3.2 Results

Table 4.2 is for 10 trails for both sets of data forwards and backwards. Both test sets show that the speed going forward and backwards is similar. Set 1 showed the forward direction was approximately 0.3 seconds faster than the backward direction and set 2 had it being approximately 0.15 seconds slower. The average speed of $81.633\frac{mm}{s}$ can be converted to $0.210\frac{BLU}{s}$ with 1 Body Length Unit (BLU) equaling to 388 mm in this instance. The second set gives an average speed of $42.59\frac{mm}{s}$ which converts to $0.106\frac{BLU}{s}$ with 1 Body Length Unit equaling 403 mm. Due to the fact that one of the main features of CLARA is to expand and contract, using Body Length Units does not accurately describe the speed at any given point. If the previous values were expressed with the fully stretched out length of 499 mm , the speeds would be $0.167\frac{BLU}{s}$ and $0.085\frac{BLU}{s}$.

Table 4.2: CLARA Speed Test: Time to Traverse 1 Meter In Pipe (s)

Trial	1	2	3	4	5	6	7	8	9	10	Avg
60 RPM FWD	11.87	12.07	12.17	12.30	12.23	12.33	12.04	12.10	12.64	12.67	12.25
60 RPM BKWD	12.40	12.13	12.80	12.83	12.90	12.74	12.00	11.60	12.17	13.74	12.53
30 RPM FWD	23.40	23.30	23.00	23.40	23.20	23.50	23.30	23.50	22.90	25.30	23.48
30 RPM BKWD	23.50	23.4	23.10	23.00	23.50	23.60	22.60	23.10	22.90	24.40	23.31

Figure 4.6 demonstrates a time lapse of CLARA during the 2nd test set going forward. As shown, the time lapse demonstrates a constant speed throughout the pipe traversal which is typical of wheel type mobile robots. A notable issue that was shown during testing was that CLARA can spiral as it goes through the pipe. This can present problems over time most of the weight of CLARA can start to offset onto a single wheel, causing undue strain in the part.

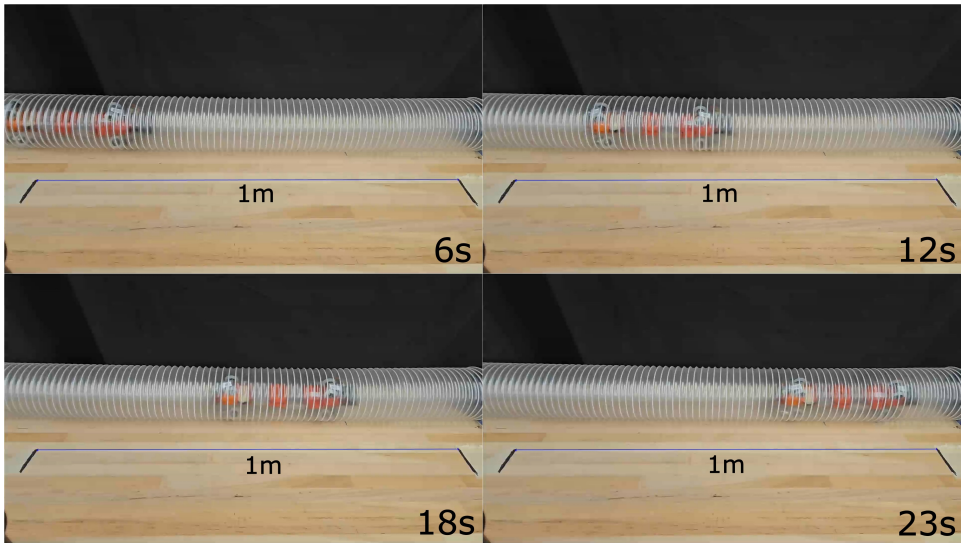


Figure 4.6: CLARA 30 RPM Speed Test Time Lapse

4.4 Elevation Test

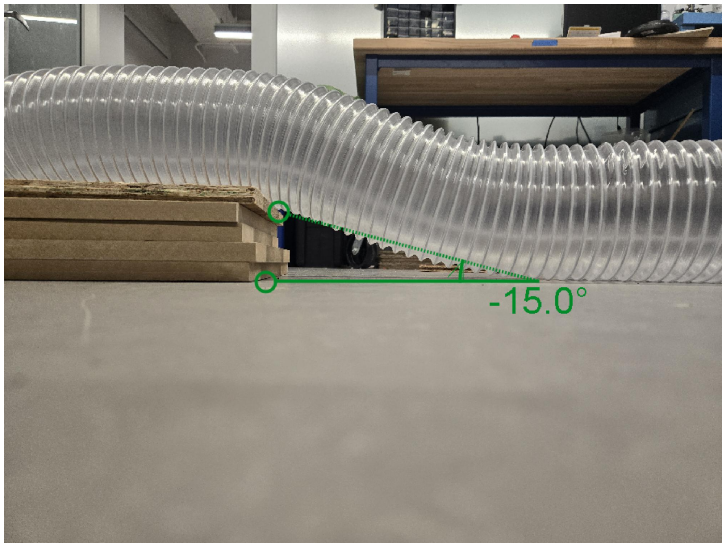
There are many pipe network configurations that contain stretches of pipe that are sloped or completely vertical. Determining CLARA's capabilities when navigating inclined pipes will give a better insight on its versatility when navigating these pipe networks. The following test was designed to determine the steepest incline CLARA can navigate.

4.4.1 Test Setup

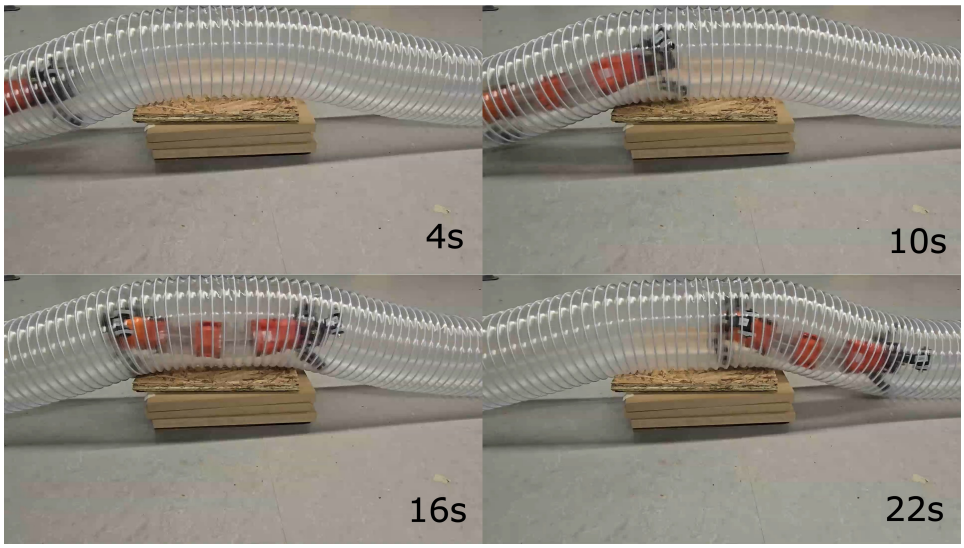
Similarly to the speed test, the 6 inch diameter PVC hose was stretched along the floor to allow CLARA to effectively navigate the pipe. 0.75" (19.05 mm) thick wooden planks were stacked under a section of the pipe as CLARA moves back and forth, increasing the number of planks each time it successfully moves up and down the incline. This process was repeated until CLARA could either no longer move up the incline or there was some sort of mechanical failure due to the strain of climbing. Video recording was taken during the test to analyze how CLARA moved through the pipe when climbing an incline.

4.4.2 Results

CLARA was able to scale up an incline of 15° when stacking 5 plates for a total of 3.75 in (95.25 mm) in height. Any increase afterwards lead to a breakdown in the suspension components. The spiraling effect detailed in the previous test is the main cause of this issue and it was noted that CLARA tended to spiral more often going up and down the different elevations. Figure 4.7 shows a time lapse of the max elevation that CLARA can handle which shows that is takes roughly half a minute to climb it.



(a) Maximum Traversable Incline



(b) Elevation Climb Time Lapse

Figure 4.7: Elevation Test

4.5 Battery Test

Pipe inspection can be preformed over an extended period of time. For this reason it is important to determine the battery life of CLARA. CLARA contains 11 motors, the Tiny Pico, and the ESP32 camera. The original hypothesis was that CLARA consumes the most power when it is moving through the pipe. This is when the most amount of wheels would be running at any given point. This would also be the action that CLARA would be preforming the most as pipe networks are mainly comprised of lengths of straight pipe.

4.5.1 Test Setup

The code was adjusted so that pressing x during the actions of driving the wheel motors, actuating the front worm gear, or actuating cable 1 gives the current values of the first front wheel, front worm gear motor, and cable 1 winch motor respectively as well as the total current values of all 11 motors via the current output function in the SIMC . Due to the daisy chain design, the total current that goes through the 11 motors at any given point is the sum of all current values. The ESP32 CAM was streaming during this test. The rationale was to emulate a pipe inspection operation which requires the camera to be streaming throughout the entire process. Parts of these tests were performed simultaneously with other tests to retain the practical element when gathering data. At random points during the action, the current values were outputted by the Tiny Pico and sent back to the ESP32 transmitter which printed out the results in the terminal line. 10 data points were sampled and averaged. Set 1 data recorded the current draw of a wheel motor during the speed test, set 2 recorded the current draw of the front variable diameter suspension system motor while the front suspension diameter was expanding and retracting in the pipe, and set 3 recorded the current draw of the motor for cable 1 when it was being wound and unwound during the joint test for the 5 inch tee joint.

An additional test was recorded that showed the current draw throughout an elevation test. The previously described test setup for the elevation test was used. During this test, the transmitter would prompt the Tiny Pico for the total current of all the motors every 250 milliseconds. The data is written out into the serial terminal. Data recording begins when the front module is at the base of the incline and ends when CLARA is on the other side of the elevation and completely level. The wheel motors will be running as the data sampling begins. This is so that the results reflect the values for an operation already in progress.

4.5.2 Results

Table 4.3 are values of the current draw on the motors for the tests. The data sheets state that the power consumption for the ESP32 CAM at 5 volts and with the flashlight on is 310 mA and for the Tiny Pico at 3.3 volts is 700 mA. This means that current draw on average would be 3.93 A at most points in operation, 4.43 A when it is adjusting the diameter of the suspension system, and 4.06 A when actuating a single cable to try and maneuver around a joint. The hypothesis that the action that would consume the most power is moving the wheels was incorrect. It did not take into consideration the energy required to keep the motors in place. This is especially important for the motors actuating the variable diameter suspension system. Even during other actions, the suspension motors were consistently drawing the most current. This is due to the force on the wheel links while they are pressed agains the pipe surface.

Table 4.3: CLARA Battery Tests

Test Set	1	2	3	4	5	6	7	8	9	10	Avg
Set 1 mA	297	280	263	275	272	271	283	272	268	259	274
Total A	3.17	3.08	2.98	2.99	2.92	2.95	2.99	2.94	2.58	2.55	2.92
Set 2 mA	379	371	368	364	361	360	360	359	357	357	363.6
Total A	3.61	3.49	3.43	3.40	3.39	3.40	3.39	3.37	3.34	3.35	3.42
Set 3 mA	302	291	286	285	284	282	281	281	276	289	285.7
Total A	3.27	3.21	3.20	3.01	3.05	3.03	2.99	2.96	2.93	2.81	3.05

The total battery life at full charge is 800 mill ampere hours. Using this value and the current draw total for set 1, the the maximum battery life would be approximately 10.84 minutes of continuous movement. Any other actions such as adjust the suspension diameter or actuating the body would decreasing this time. This value does not consider the fact that when the batteries drop below 5 volts, the SIMC and ESP32 CAM stop functioning. This duration is pretty short and can cause issues with navigating large networks. Current draw increases when preforming operations such as adjusting the suspension size and actuating cables as these actions can experience the most resistance. Large amounts of resistance can lead to the motors overheating.

Figure 4.8 shows the current draw for CLARA during the elvation test. It factors in the current draw for the Tiny Pico and ESP32 CAM. The camera was not streaming and the flashlight was not on so for the purpose of this test the current draw for CLARA was assumed to be 180 mA instead of 310 mA. As CLARA begins to climb the incline, the current draw from the system is at its highest with a value of 6.4 A. As CLARA begins to climb, the current draw begins to drop at a consistent curve until the front module reaches the top of the elevation where it begins to settle at 3.9 A. It continues to be at this level until the end of the test. CLARA expends the most power as the wheels start to move the incline. As the body starts to move onto the incline, the wheels begin to acclimate to the incline and the power consumption begins to reflect the same values as moving through a straight pipe.

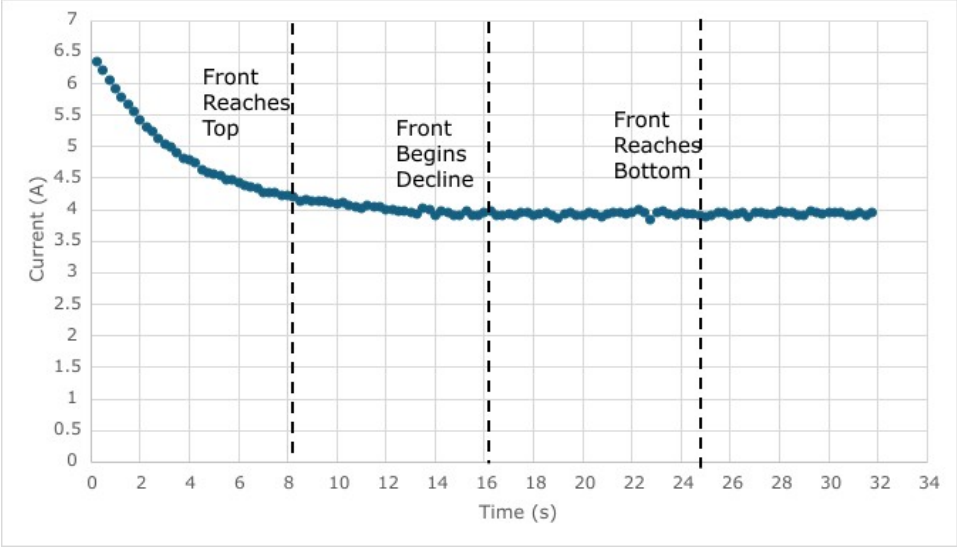


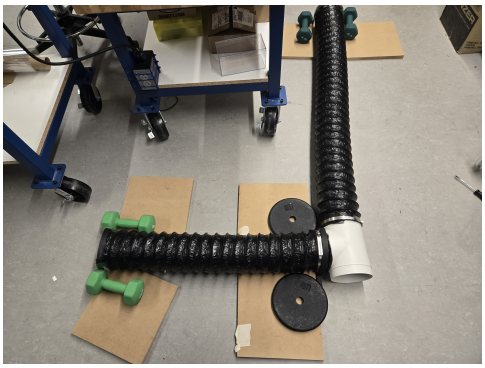
Figure 4.8: Current Draw During Elevation Test

4.6 Maze & Joint Maneuvering

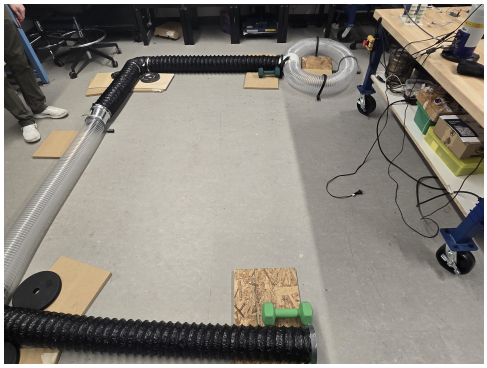
CLARA's flexibility was a key design aspect that needed to be tested within a practical setting. The final test to determine the efficacy of CLARA was to determine how well it can navigate junctions such as elbow joints and tee joints at different sizes. Due to the expected span of the suspension systems, the test was limited to using pipe and joint sizes between 5" and 6". Different joints such as elbows, tee joints, and reducer couplings would be utilized to satisfy the criteria that was set for CLARA and other pipe inspection robots.

4.6.1 Test Setup

The first part of the test was to determine how well CLARA does with each individual joint. The test setup was two lengths of pipe connected by a single joint for each joint size and type. The test for the 5 inch joints were preformed using the same clear PVC hose material for the speed and battery tests. The 6 inch joints were tested using covered PVC dryer flexible hose pipes. This was done to constrain the operator to only using the ESP32 CAM to determine how and when to maneuver through the pipe. The resolution the camera streamed at was kept to SVGA (800 x 600) as this was considered to be a good medium between power consumption and clarity. The pipes were stretched over the ground and held taut by weights and brackets. The test begins when CLARA reaches the junction.



(a) 6" Tee Test Setup Overview



(b) Maze Configuration 1 Test Overview

Figure 4.9: Successful Joint Maneuver Time Lapses

The test ends when either the operator successfully moves CLARA through the junction completely or when the operator has decided that CLARA is unable to move through the junction. For the clear PVC tests, video footage was recorded and CLARA's movements were analyzed to measure qualitative data on how it may move through the junction. For the covered PVC pipes, footage of the ESP-CAM will be recorded for one of each type of trial to analyze what the operator will be seeing during normal operations. The final test was to create maze configurations using the joints that CLARA is able to consistently maneuver and have CLARA run through this maze and see how fast the operator can finish the maze. The maze had slight changes to its configuration based on how well CLARA managed to finish the maze to scale the difficulty for the operator. Figure 4.9 shows the different test setups used.

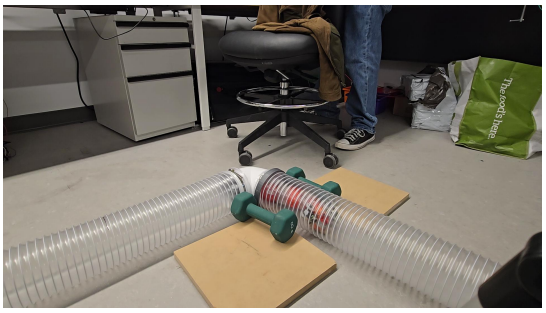
4.6.2 Results

Table 4.4: CLARA Battery Tests

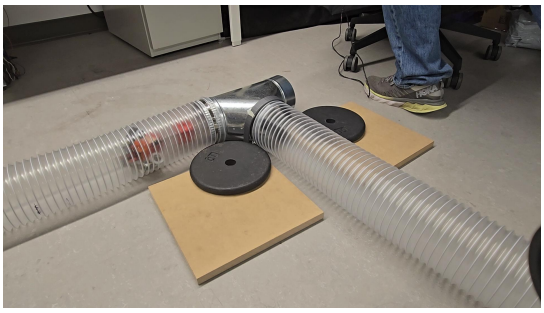
Joint	1	2	3	4	5	Avg
5" Elbow	DNF	DNF	DNF	DNF	DNF	N/A
5" Tee	DNF	490.47	291.57	DNF	223	335.013
6" Elbow	50.22	43.39	43.02	36.70	86.87	52.04
6" Tee	85.89	87.64	160.11	244.33*	161.28*	147.85

Table 4.4 illustrates the times for maneuvering the different junction where 5 trials were preformed. A successful maneuver is the time from when CLARA approaches the entrance of a joint and when the back wheels reach the exit of the joint. Videos were recorded for the 5 inch test while the operator judged the time for the 6 inch based on the ESP-CAM and observing the movement in the pipe from the outside. The average time of completion for each joint shows that CLARA was more effective maneuvering in junctions that are 6" in diameter and struggled with 5" in diameter. Figures A.1 from Appx. A shows time lapses of each type of successful trials.

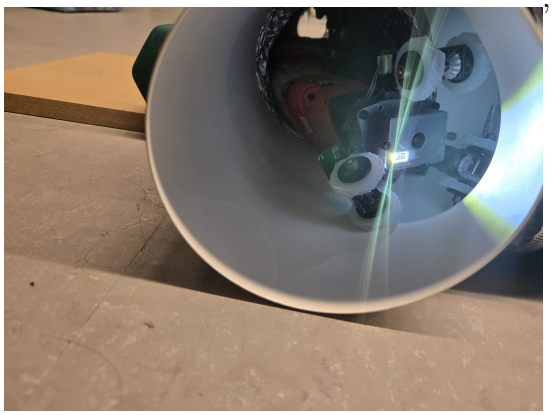
Most notably it is shown that CLARA is incapable of maneuvering in the 5 inch elbow joint. Analyzing the video shows that the back wheels would not be able to properly anchor its position in the pipe and would often cause a misalignment and go at awkward angles when actuating the cables. Due to this the front portion of CLARA would remain in the same position and not bend towards the exit CLARA was trying to move through. Figure 4.10(a) shows the misalignment issue in the back module. In both instances, the wheels could not gain any traction and would sometimes have zero contact with the inner pipe surface. Furthermore it would create strain on both the wheel motors and the worm gear motors and create intense heat that would either kill the motor or disfigure the casings and make them unusable.



(a) 5 in Elbow Joint Fail State



(b) 5 in Tee Joint Fail State

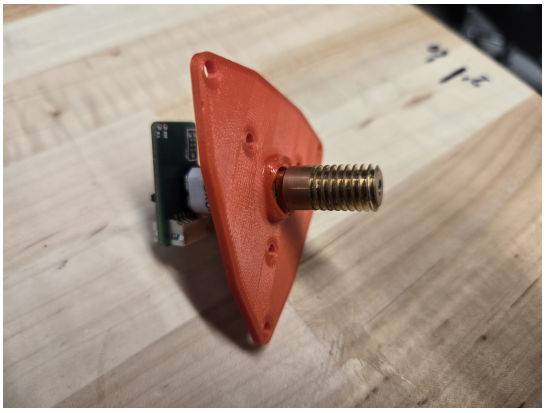


(c) Potential Fail State Inside Tee Joint

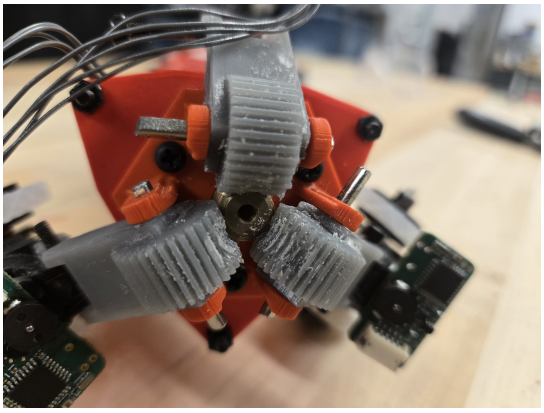
Figure 4.10: CLARA Fail States

Another notable observation was how it maneuvered through the 5 inch tee joint. CLARA is able to fully go through the joint when the two outlets are both at a 90° angle, but fails to get through the joint when only one of these outlets are at an angle. When the operator partially actuates CLARA in its preferred direction and drives forward, the body automatically complies to the shape of the joint. CLARA pushing against the inner wall causes it to do this. The inner wall is also used for traction for one of the wheels and allows more driving force to be applied. When there is no inner wall to drive into due to one of the outlets being straight ahead, CLARA can't get the necessary traction to move forward or the force it needs to automatically comply toward the outlet. This means that CLARA can only maneuver through a tee joint from a specific opening.

Figure 4.10 shows different fail state of CLARA when navigating the elbow and tee joint. Figure 4.10(c) shows a potential fail state that can happen within a tee joint. It is possible for CLARA to get stuck in this position regardless of the size of the pipe. Depending on how expanded the wheels are, the rigid links can get caught on an inner pipe edge and cause damage when trying to move forward. The operator must especially be careful how it angles CLARA and how expanded the wheels are when moving through a tee joint in this direction.



(a) Plate Damage



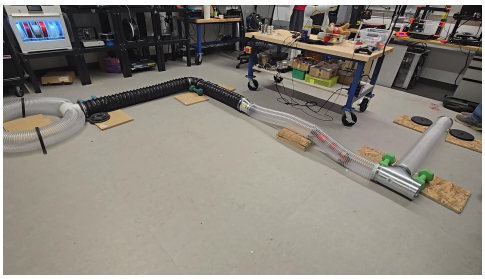
(b) Rigid Link Gear Damage

(c) Motor Housing Damage

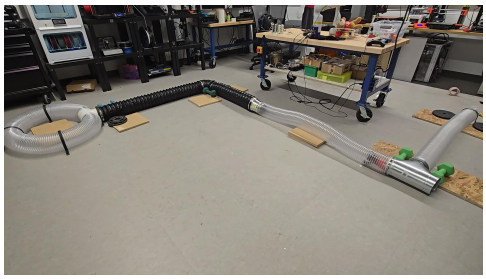
Figure 4.11: Damaged CLARA Parts

CLARA was unable to successful complete a trial within the maze. The furthest it was able to get was to the 6 inch elbow in the first configuration before the battery died. Other places it failed was during an incline and within the first tee joint as shown in Figure 4.12. CLARA might not be able to finish a maze but be able to maneuver through single joints in isolated tests seems to be due to the orientation issue for the Yoshimura modules. The misalignment issue makes it difficult for CLARA to readjust after exiting, creating an awkward orientation that creates the stress that disfigures the components. Readjustment is not possible with the current system as the kinematic model for this type of Yoshimura continuum is not fully understood and there is no current method in the software to properly track the cable lengths of each individual Yoshimura module.

Additionally, struggling at any obstacle will exacerbate that misalignment and heavily drain the battery as stress on the variable suspension motor peaks at these instances. Shortly after finishing a trial and starting maintenance, it was noted how hot these particular motors were and in most cases the motor would end up becoming unusable and needed replaced. Operational stress is the main contributor to how ineffective CLARA is in this practical setting.



(a) Fail State on Incline



(b) Fail State in Tee

Figure 4.12: Maze Fail States

While CLARA can physically be threaded through all these joints, having CLARA move through it independently would require that the Yoshimura modules bend in a way where the front module is mostly curved while the back module remains straight. If CLARA was able to properly anchor itself or if it was structured like Lizard[24] where the modules are actuated by their own set of motors, this could be achieved. However in order to keep the novel aspect of CLARA, the design focus would need to be creating a better suspension system to properly anchor CLARA in place. While the lead screw design from the previous iterations[25, 26] were more stable by virtue of their metal lead screw, threaded inserts and the extra brace to keep the lead screw from misalignment, the added length would make it much more difficult to maneuver through the junctions. Addressing the issue of the pieces wearing too quickly, fabricating the components out of stronger material would reduce wearing and extend the lifetime of the part. The current material for the rigid links is Tough 1500 V1 which has a high ultimate tensile strength of 34 MPa and Modulus of 1460 MPa. This should be acceptable for the application that CLARA is using it for, but due to the fine detail in the pieces such as the teeth of the gears and the press fit hole the motor shaft goes into they are more liable to wear down or completely break. Similarly, the PLA for the plates and casings should be durable enough to hold the motors in place despite the torque applied on them, but it is possible that heat from the motors being stressed contributed to the severe warping in the parts as shown in Figure 4.11. Due to these factors, parts were switched out more often and impeded testing.

Chapter 5

Conclusion

This chapter will reflect on the results of tests and will deliver the verdict if CLARA is able to function in a practical environment over a prolonged period of time. Recommendations will also be considered for any future work that CLARA might receive.

5.1 CLARA Efficacy

From the results of the tests preformed, it is determined that CLARA can in fact maneuver within isolated pipe environments at a decent speed but can not be used for prolonged periods of time within a practical environment. In an isolated environment, CLARA is able to move through a decent range of junctions but at a relatively slow rate compared to traversing straight pipe lengths. This can be due to operator inexperience but also the robot's misalignment with the pipe and orientation issue from CLARA's movements.

The main issue of CLARA's movements is the awkward configurations that result from the two Yoshimura module design. Due to the unknown kinematic model for this continuum configuration, judging how the inputs would effect the robot was difficult and not always apparent. While the camera view might show a straight a head view, it's possible that the back module could still be misaligned and create stress that wears the parts down. Another contributing factor is the design of the variable diameter suspension system. Even if it managed to shorten the rigid links, it was shown to be ineffective in anchoring the robot in place when the robot begins to bend. This creates the aforementioned misalignment issue.

Coupled with the fact that the only output displayed to the operator is the camera feed and that the control scheme is fairly basic, controlling CLARA ends up being challenging and awkward for the average operator. This can explain the long joint traversal times and the inputs that lead to misalignment and part damage. Although there are various issues stemming from certain mechanical and software design flaws, there are potential fixes that future projects can address.

5.2 Future Recommendations and Further Broader Impacts

There are two potential routes that future teams can explore for fixing the misalignment issue. The first method would be to change how the cables are actuated. By moving the motors to the center, a pulley system can be implemented that actuated both modules at the same rate, thus reducing the awkward orientation issue. Fixing that issue will reduce the risk of having misalignment. The second route would be to update the software so that it can either alert the operator of an awkward orientation or have a controller that easily adjust the modules to the desired shape. Doing this would require a better understanding the kinematic model of the current continuum configuration.

In either scenario it would be best to change the variable diameter suspension system to a more efficient design. Something that can apply the necessary force to stay in the pipe and not over work the motor that actuates the suspension system. Reducing the number of wheel motors via some sort of transmission system would also help with the battery life. Ideally, keeping the system shorter than the lead screw design from the previous design would still be a focus.

LiPo batteries are noted for being high density battery cells. Using these small batteries allowed CLARA to remain as small as it is, traveling in 5 inch diameter pipes. The best way to reduce current draw and increase battery duration would be to reduce the number of motors being used on CLARA. Since all 6 motors are meant to go at the same rate when actuating the wheels, reducing that number might be the best method for the next iteration. The original Salamander[23] utilized one motor to actuated the front wheels using a power transmission system. Translating that somehow to a variable diameter suspension system would be the bigger challenge for the next design.

In the introduction chapter of this paper, the initial broader impacts section discussed how pipe network failures effect broader society as a way to justify designing CLARA. After conducting the practical testing for CLARA, I have further considerations to how CLARA is controlled and what that means for an operator. In its current state, CLARA is difficult to control specifically in how it maneuvers joints even with the x-box gamepad which is a familiar method of control. Training would have to be conducted to maximize efficiency and minimize potential damage that can happen to CLARA from poor operation. The training the operators would receive should prioritize practical methods such as the maze and joint setups preformed in this thesis project. Environmental considerations would have to be factored into the training such as the type of pipes CLARA would traverse and any residual material that is in the pipes. Additionally operators would have to be trained on how to maintain and repair CLARA in these settings. This will allow inspection processes to continue even in the event of a design failure.

5.3 Final Thoughts

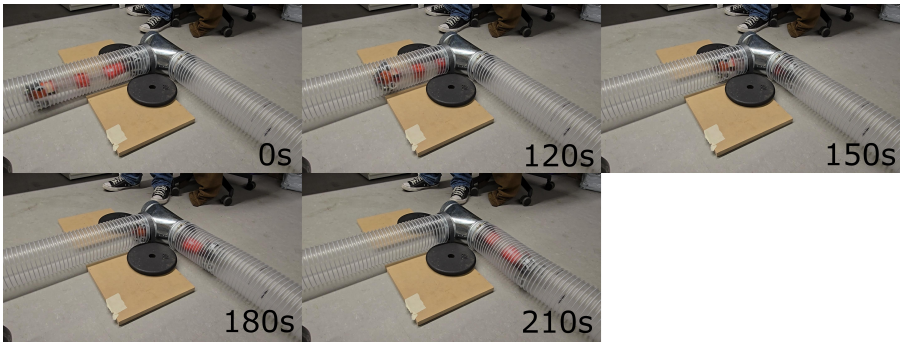
Overall the project was a success for both of its goals. This new version of CLARA is considerably more flexible and the efficacy of its design and concept were verified. Going forward the main goal of the next iteration would be to make the design efficient enough to be used for proper pipe inspection. The initial results of this thesis project showed that there is merit to further developing the design.

Appendix A

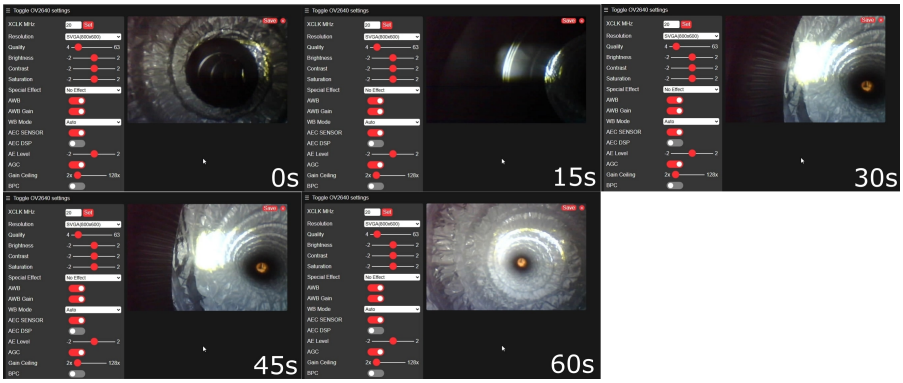
Tables and Figures

Table A.1: Bill of Materials

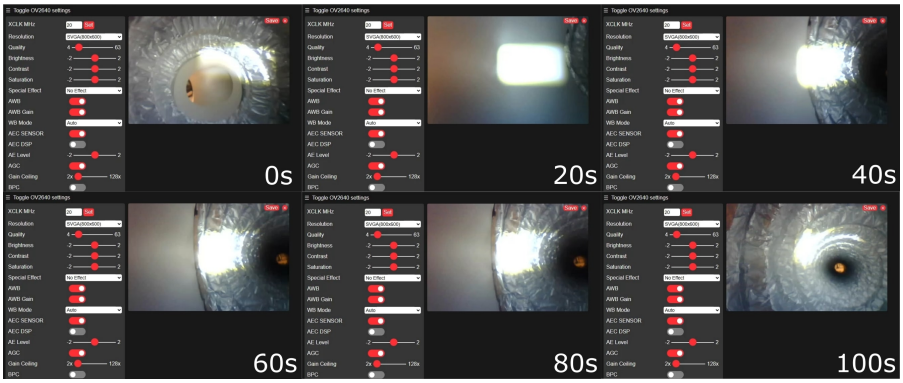
Part Name	Supplier	Quantity	Unit Price
ESP32 Development Board	Amazon	1	$8.99
X-Box Gamepad Controller	Amazon	1	$29.99
UM Tiny Pico Development Board	Adafruit	1	$20.00
ESP32 CAM Development Board	Amazon	1	$13.99
MP1584EN Mini 5 Volt Buck Regulator	Amazon	1	$1.99
Pololu 5V, 1A Step-Down Voltage Regulator D24V10F5	Pololu	1	$11.95
Pololu 3.3V, 1A Step-Down Voltage Regulator D24V10F3	Pololu	1	$11.95
Pololu Micro Metal Gearmotor HP 6V with Extended Motor Shaft	Pololu	11	$23.87
4" 5 Position Socket to Socket ZR Series JST Connector	DigiKey	7	$1.37
8" 5 Position Socket to Socket ZR Series JST Connector	DigiKey	4	$1.65
400 mAh 2s 7.4V Lipo Battery	Amazon	2	$12.00
JST compatible On/Off Power Switch	Amazon	1	$4.00
		Total	$405.62



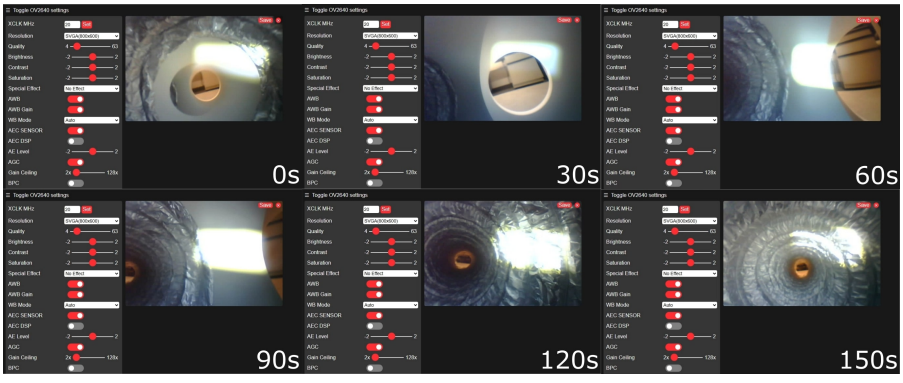
(a) 5 in Tee Joint Time Lapse



(b) 6 in Elbow Joint Time Lapse



(c) 6 in Tee Joint A Time Lapse



(d) 6 in Tee Joint B Time Lapse

Figure A.1: Successful Joint Maneuver Time Lapses

Appendix B

Software

Listing B.1: Gamepad Code

```
import hid
import XInput
import serial
import time
import math
import yaml

# LOAD SEMI-OPTIONAL CONSTANTS
# load constants from config.yaml file
try:
    with open('config.yaml', 'r') as ymlfile:
        cfg = yaml.load(ymlfile, Loader=yaml.Loader)
except FileNotFoundError:
    print('config.yaml file not found, fix that please')
    cfg = {}
    quit()

TIME_DELAY = cfg.get('time delay')
DESIRED_BAUD = cfg.get('desired_baud')
HARDSET_PICO = cfg.get('hardset_pico')



def is_idle(state):
        active = 0b0
        button = XInput.get_button_values(state)
        for presses in button.values():
            if presses:
                active |= 0b1
        triggers = XInput.get_trigger_values(state)
        if triggers[0] != 0 or triggers[1] != 0:
            active |= 0b1
```

```
        return active

def construct_button_bits(state):
    """
    Constructs the button bits from the gamepad state
    :param gpst: Gamepad State
    :return button_st: 10 Bits Representing the Button State
    """
    buttons = XInput.get_button_values(state)
    # Blank Button

    button_st = 0b00000000

    # Regular Buttons
    if buttons.get("A"):
        if __debug__: print("A")
        button_st |= 0b0000001
    if buttons.get("B"):
        if __debug__: print("B")
        button_st |= 0b0000010
    if buttons.get("X"):
        if __debug__: print("X")
        button_st |= 0b0000100
    if buttons.get("Y"):
        if __debug__: print("Y")
        button_st |= 0b0001000



    if buttons.get("DPAD_RIGHT") and buttons.get("DPAD_UP"):
        if __debug__: print("D-Pad Up Right")
        button_st |= 0b0010000
    elif buttons.get("DPAD_DOWN") and buttons.get("DPAD_RIGHT"):
        if __debug__: print("D-Pad Down Right")
        button_st |= 0b0110000
    elif buttons.get("DPAD_DOWN") and buttons.get("DPAD_LEFT"):
        if __debug__: print("D-Pad Down Left")
        button_st |= 0b1010000
    elif buttons.get("DPAD_UP") and buttons.get("DPAD_LEFT"):
        if __debug__: print("D-Pad Up Left")
        button_st |= 0b1110000
    elif buttons.get("DPAD_UP"):
        if __debug__: print("D-Pad Up")
        button_st |= 0b0000000
    elif buttons.get("DPAD_RIGHT"):
        if __debug__: print("D-Pad Right")
        button_st |= 0b0100000
```

```
    elif buttons.get("DPAD_DOWN"):
        if __debug__: print("D-Pad Down")
        button_st |= 0b1000000
    elif buttons.get("DPAD_LEFT"):
        if __debug__: print("D-Pad Left")
        button_st |= 0b1100000

    # Rear Buttons
    button_st = button_st << 2
    if buttons.get("LEFT_SHOULDER"):
        if __debug__: print("LB")
        button_st |= 0b000000001
    if buttons.get("RIGHT_SHOULDER"):
        if __debug__: print("RB")
        button_st |= 0b000000010

    return button_st



def construct_trigs(state):
    """
    Constructs the sticky bits from the gamepad state
    :param gpst: Gamepad state
    :return stick_st: 4 Bytes Representing the Sticky State
    """


    LT_RT = XInput.get_trigger_values(state)
    if LT_RT[0] > 0.5 and LT_RT[1] > 0.5:
        triggers = 0b00100000
        if __debug__: print("LT_RT")
    elif (LT_RT[1] > 0.5):
        triggers = 0b01000000
        if __debug__: print("RT")
    elif (LT_RT[0] > 0.5):
        triggers = 0b10000000
        if __debug__: print("LT")
    else:
        triggers = 0b00000000

    triggers = triggers << 24

    return triggers


def main():
```

```
# Open the Serial Port
ser = serial.Serial(HARDSET_PICO, DESIRED_BAUD,timeout=0)

if __debug__: print(f"Found serial at {ser.name}" + f" with baud
    {ser.baudrate}")
#ser.open()

print("--- Serial Port Opened, Gamepad Connected, Ready for Action ---")

# Save Previous State
prev_state = None

# Main Loop
while True:

    if XInput.get_connected()[0]:



        # Construct Button Bits
        state = XInput.get_state(0)
        button_state = construct_button_bits(state)

        # Construct Sticky
        trig_state = construct_trigs(state)

        active = is_idle(state)

        # Print sizes of the variables


        # Combined Sticky and Button Bits format(enable, '01b')
        curr_state = format(active, '01b') + format(button_state, '09b') +
            format(trig_state, '032b') + '\n'



        # Write to Serial
        if curr_state is not prev_state:
            combo = curr_state.encode('ascii')
            byr = ser.write(combo)
        # Assign the
        prev_state = curr_state

        #if __debug__: print(f"Bytes Written: {byr}")
```

```
            time.sleep(0.15) #v3


            # Print from the serial port buffer, allowing for multiple lines of
                debug
            while ser.in_waiting:
                #if __debug__: print("Writing in ser")
                #while True:
                print(ser.readline())



            time.sleep(0.15)

if __name__ == '__main__':
    main()
```

Listing B.2: Transmitter Code

```
#include <esp_now.h>
#include <WiFi.h>
#include <sensor_data.h>

/**********************************************************************************************

                                                VARIABLES + DEFINITIONS


 **********************************************************************************************
//ESP-NOW Variables
//uint8_t broadcastAddress1[] = {0x4C, 0x75, 0x25, 0xCA, 0x9F, 0x3C};
    //replace with the MAC Address of your ESP
uint8_t broadcastAddress1[] = {0xD4, 0xD4, 0xDA, 0xAA, 0x2E, 0xD0}; // v3
//uint8_t broadcastAddress1[] = {0xC8, 0xC9, 0xA3, 0xCF, 0xAE, 0xE8}; //
    Sender 1
//uint8_t broadcastAddress1[] = {0xC8, 0xF0, 0x9E, 0xA2, 0x5F, 0xA0}; //
    Sender 2


typedef struct data_struct_rec { //data struct to receive wifi commands from
    the external ESP - must match data struct sent from mainboard
  String wifiData;
} data_struct_rec;
data_struct_rec test;
/*
typedef struct data_struct { //data struct to receive from the mainboard
    containing address, current sensor, and encoder data
  // data types must match data struct that mainboard is expecting
  int smdAddress;
  int currentData;
  int encoderData;
} data_struct;

 //create instance of sending struct to be populated with data
data_struct receiveData; //create instance of receiving struct
*/
//SensorData sensor_values;
typedef struct MotorGroup {
  //Front Wheels
  SensorData M1;
  SensorData M2;
  SensorData M3;
  //Back Wheels
  SensorData M4;
  SensorData M5;
```

```
  SensorData M6;
  //Cables
  SensorData M7;
  SensorData M8;
  SensorData M9;
  //Screws
  SensorData M10;
  SensorData M11;
} MotorGroup;

MotorGroup All_Motors;

typedef struct CurrentGroup {
  SensorData Wheel;
  SensorData Screw;
  SensorData Cable;
  int16_t Total_Current;
} CurrentGroup;

CurrentGroup Battery_Test;

typedef struct F_struct{
  float v1;
  float v2;
  float v3;
} F_struct;

F_struct f_values;
/*
int countA = 0;
int countB = 0;
bool countC = true;
*/
bool change = true;
bool mode = true;
bool input_change = true;
bool only_up = false;
String deviceBData = ""; //variable for inputting commands into the serial
    monitor and storing the input

String newData = "";
String oldData = "0";

char deviceBdata;
char input[43];
```

```
/**********************************************************************************************

                                                    SETUP + LOOP


 **********************************************************************************************/

void setup() {
  Serial.begin(9600);
  delay(100);

  //initialize device as wifi station
  WiFi.mode(WIFI_STA);

  //initialize ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  //register callback function to be called when a message is sent
  esp_now_register_send_cb(OnDataSent);

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);

  // register peer
  esp_now_peer_info_t peerInfo;
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // register first peer
  memset(&peerInfo, 0, sizeof(peerInfo));
  memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }
}

void loop() {
  if (Serial.available() > 0) {
    if (Serial.peek() != '\n') //if we press enter in the serial monitor and
        sent data
    {
      deviceBData += (char) Serial.read(); //add read string into a data cache
```

```
    }
    else { //end of aline
      Serial.read();
      //now need to interpret deviceBData
      //Serial.print("You said: ");
      newData = interpretData(deviceBData);
      //Serial.println(newData);
      // Changes Being Here
      if(newData.toInt() == oldData.toInt()){
        input_change = false;
        //Serial.println("Not New");
      } else {
        input_change = true;
        oldData = newData;
        //Serial.println("New");
      }
      if(input_change){
        //sends the actual data
        test.wifiData = newData; //convert data to an integer
        //send the message - first argument is mac address, if you pass 0 then
            it sends the same message to all registered peers
        esp_err_t result = esp_now_send(0, (uint8_t *) &test,
            sizeof(data_struct_rec));
        if (result == ESP_OK) {
          //Serial.println("Sent with success");
        }
        else {
          //Serial.println("Error sending the data");
        }
      }
      //Changes End Here
      deviceBData = "";
    }
  }else{
    //Serial.println("Oopsie! No Serial Available");
  }

  delay(5);
}


/*****************************************************************************************

                                        ESP-NOW WIFI HELPER FUNCTIONS

 *****************************************************************************************
```

```
/**
    Callback function to be executed when WiFi data is sent to mainboard
    prints if message was successfully delivered to know if board received
        message
    @param mac_addr - the mac address of the board that data is being sent to
    @param status - the status of the transaction - success or fail
*/
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  char macStr[18];
  //Serial.print("Packet to: ");
  // Copies the sender mac address to a string
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
          mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
              mac_addr[5]);
  // Serial.print(macStr);
  //Changes Begin Here
  if(status == ESP_NOW_SEND_SUCCESS && change){
    Serial.println("Send Status: Delivery Successful");
    change = false;
  } else if (status != ESP_NOW_SEND_SUCCESS && !change){
    Serial.println("Sent Status: Delivery Failed");
    change = true;
  } // Changes End Here
  //Serial.print(" send status:\t");
  //Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
      "Delivery Fail");
}


/**
   Callback function to be executed when WiFi data is received from mainboard
   Prints received smart motor driver address, current sensor and encoder data
   @param mac - the mac address of the board sending the data
   @param incomingData - the data to be copied into the myData variable - the
       instance of the receiving struct
   @param len the number of bytes received
*/
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {



  if(len == sizeof(MotorGroup)){
    memcpy(&All_Motors, incomingData,sizeof(MotorGroup));
    Serial.println("Front Wheels Current:");
    Serial.println(All_Motors.M1.current);
    Serial.println(All_Motors.M2.current);
```

```
    Serial.println(All_Motors.M3.current);
    Serial.println("Back Wheels Current:");
    Serial.println(All_Motors.M4.current);
    Serial.println(All_Motors.M5.current);
    Serial.println(All_Motors.M6.current);
    Serial.println("Cables Current");
    Serial.println(All_Motors.M7.current);
    Serial.println(All_Motors.M8.current);
    Serial.println(All_Motors.M9.current);
    Serial.println("Front Lead Screw Currents");
    Serial.println(All_Motors.M10.current);
    Serial.println(All_Motors.M11.current);
  } else if(len == sizeof(F_struct) /*&& mode*/){
    memcpy(&f_values, incomingData,sizeof(F_struct));
    Serial.println("Cable lengths:");
    Serial.println(f_values.v1);
    Serial.println(f_values.v2);
    Serial.println(f_values.v3);
  } else if(len == sizeof(CurrentGroup)){
    memcpy(&Battery_Test, incomingData,sizeof(Battery_Test));
    Serial.println("Current of One Wheel");
    Serial.println(Battery_Test.Wheel.current);
    Serial.println("Current of One Screw");
    Serial.println(Battery_Test.Screw.current);
    Serial.println("Current of One Cable");
    Serial.println(Battery_Test.Cable.current);
    Serial.println("Total Current Draw");
    Serial.println(Battery_Test.Total_Current);
  }

}



/*********************************************************************************************

                                                CONTROLLER FEEDBACK FUNCTIONS


 *********************************************************************************************


/**
 * Function to interpret input from Logitech gamepad from registers and return
     a command as a String to send to the mainboard
 * @param data - String output from controller
 */
```

```
String interpretData(String data) {
  data.toCharArray(input, 43);
  //Serial.println(data);

  //A button is index 0
  only_up = true;

  // Idle so Brake
  for(int i = 1; i <= 12; i++){
    if(data[i] == '1'){
      only_up = false;
      break;
    }
  }
  //make D pad into integer 0-7
  String dpad = "";
  int dpadval = 0;
  for (int i = 1; i <= 3; i++) {
    dpadval *= 2;
    if (data[i] == '1'){
      dpadval++;
    }
  }
  //Brakes
  if(data[0] == '0' || only_up) {
    //Serial.println("Stop");
    return "0";
  }
  // Forward & Backward
  else if (data[8] == '1' && data[5] == '0') { //right bumper forward
    return "8";
  }else if (data[9] == '1'&& data[5] == '0') { //left bumper backward
    return "9";
  }else if (data[8] == '1' && data[5] == '1'){ //forward battery test
    return "31";
  }else if (data[9] == '1' && data[5] == '1'){ // Back
    return "32";
  }

  // Single Cable Control
  else if (dpadval == 2 && data[6] == '0' && data[5] == '0') { //cable 1 down
      d-pad right
    return "12";
  }else if (dpadval == 2 && data[6] == '0' && data[5] == '1') {
    return "33";
  }else if (dpadval == 4 && data[6] == '0') { //cable 2 down d-pad down
    return "14";
```

```
}else if (dpadval == 6 && data[6] == '0') { //cable 3 down d-pad left
  return "16";
}else if (data[6] == '1' && dpadval == 2 && data[5] == '0') { //B button
    with dpad right cable 1 extend
  return "18";
}else if (dpadval == 2 && data[6] == '1' && data[5] == '1') {
  return "34";
}else if (data[6] == '1' && dpadval == 4) { //B button w dpad down cable 2
    extend
  return "19";
}else if (data[6] == '1' && dpadval == 6) { //B button w dpad left cable 3
    extend
  return "20";
}

// Front Screw Control
else if (data[7] == '1' && data[6] == '0' && data[11] == '0' && data[5] ==
    '0') { //A button NO B - front lead screw down
  return "4";
} else if (data[4] == '1' && data[6] == '0' && data[11] == '0' && data[5] ==
    '0') { //y button NO B - front lead screw up
  return "1";
} else if (data[7] == '1' && data[6] == '0' && data[11] == '0' && data[5] ==
    '1') { //A button NO B - front lead screw up and check
  return "35";
} else if (data[4] == '1' && data[6] == '0' && data[11] == '0' && data[5] ==
    '1') { //y button NO B - front lead screw up and check
  return "36";
}  // Rear Screw Control
else if (data[7] == '1' && data[6] == '1') { //A button WITH B - rear lead
    screw down
  return "28";
}  else if (data[4] == '1' && data[6] == '1') { //y button WITH B - rear lead
    screw up
  return "27";
}

//Both Screw Controls data[11] Left trigger 10 Right 11 Both 12
else if (data[7] == '1' && data[6] == '0' && data[11] == '1'){
  return "29";
  //expand
} else if(data[4] == '1' && data[6] == '0' && data[11] == '1'){
  return "30";
  //retract
}
```

```
//check buttons
else if (data[5] == '1') { //x button - send data
  return "2";
}

//Two Cables
// Down and Right Reverse (0x09 & 0x08 currently)
else if(dpadval == 4 && data[6] == '1' && data[11] == '1'){
  //send for down right reverse
  return "22";
} else if(dpadval == 4 && data[6] == '0' && data[11] == '1'){
  //send for down right actuate
  return "21";
}
// Down Left (0x08 & 0x0F currently)
else if(dpadval == 4 && data[6] == '1' && data[10] == '1'){
  //send for down left reverse
  return "24";
} else if (dpadval == 4 && data[6] == '0' && data[10] == '1'){
  //send for down left actuate
  return "23";
}
// Right Left AKA Up (0x0F & 0x09 currently) right trigger ?
else if (data[6] == '1' && data[12] == '1'){
  //send for up reverse
  return "26";
} else if (data[6] == '0' && data[12] == '1'){
  //send for up
  return "25";
}


//All Cable Control
else if(data[10] == '1'){
  // send vaule for reverse all
  return "17";
}
else if(data[11] == '1'){
  // send value for all cables
  return "15";
}
// When a nothing button is being pushed.
else if(dpadval == 0){
  return "0";
}

else {
```

```
  return "0";
}
```

Listing B.3: Reciever Code

```
#include <Wire.h> //include Wire.h library
#include <esp_now.h> //ESP-Wifi comms
#include <WiFi.h>
#include <math.h>
#include <time.h>
#include <smartmotor.h>
#include <simc_write.h>
#include <header.h>
#include <sensor_data.h>
#include <S_OP.h>
//#include <Motoron.h>

/*****************************************************************************************

                                                VARIABLES + DEFINITIONS


 *****************************************************************************************

//ESP-NOW Variables
//uint8_t broadcastAddress1[] = {0xC8, 0xC9, 0xA3, 0xCF, 0xAE, 0xE8}; //
    Sender 1
uint8_t broadcastAddress1[] = {0xC8, 0xF0, 0x9E, 0xA2, 0x5F, 0xA0}; // Sender 2
/*
typedef struct data_struct { //data struct to send to the receiver/external
    ESP containing address, current sensor, and encoder data
  // data types must match data struct sent from external ESP
  int smdAddress;
  int currentData;
  int encoderData;
} data_struct;
*/
typedef struct data_struct_rec { //data struct to receive wifi commands from
    the external ESP - must match data struct sent from external ESP
  String wifiData;
} data_struct_rec;

typedef struct MotorGroup {
  //Front Wheels
  SensorData M1;
  SensorData M2;
  SensorData M3;
  //Back Wheels
  SensorData M4;
  SensorData M5;
  SensorData M6;
```

```
  //Cables
  SensorData M7;
  SensorData M8;
  SensorData M9;
  //Screws
  SensorData M10;
  SensorData M11;
} MotorGroup;

typedef struct CurrentGroup {
  SensorData Wheel;
  SensorData Screw;
  SensorData Cable;
  int16_t Total_Current;
} CurrentGroup;

MotorGroup All_Motors;
CurrentGroup Battery_Test;

//data_struct test; //create instance of sending struct to be populated with
    data
//SensorData sensor_values;
data_struct_rec myData; //create instance of receiving struct



//WHEEL ADDRESSES
uint8_t ADDR_DriveFront1 = 0x07;
uint8_t ADDR_DriveFront2 = 0x06;
uint8_t ADDR_DriveFront3 = 0x05;
SmartMotor F_Wheels[3] =
    {SmartMotor(ADDR_DriveFront1),SmartMotor(ADDR_DriveFront2),SmartMotor(ADDR_DriveFront3)};

uint8_t ADDR_DriveRear1 = 0x0A;
uint8_t ADDR_DriveRear2 = 0x0C;
uint8_t ADDR_DriveRear3 = 0x0B;
SmartMotor B_Wheels[3] =
    {SmartMotor(ADDR_DriveRear1),SmartMotor(ADDR_DriveRear2),SmartMotor(ADDR_DriveRear3)};

uint8_t ADDR_Cable1 = 0x09;
uint8_t ADDR_Cable2 = 0x08;
uint8_t ADDR_Cable3 = 0x0F;
SmartMotor Cables[3] =
    {SmartMotor(ADDR_Cable1),SmartMotor(ADDR_Cable2),SmartMotor(ADDR_Cable3)};


uint8_t ADDR_LeadScrewFront = 0x0D;
```

```
SmartMotor LS_Front(ADDR_LeadScrewFront);

uint8_t ADDR_LeadScrewRear = 0x0E; //110
SmartMotor LS_Rear(ADDR_LeadScrewRear);

SmartMotor Lead_Screws[2] = {LS_Front, LS_Rear};


//data variables from smart motor drivers
int count = 0; //encoder count
char current = 0; //current reading
const int vRef = 3.3; //reference logic level voltage
const int senseResistor = 0.5; //current sense resistor in Ohms
byte enc1, enc2; //encoder variables
const int encTicksPerRev = 12; //encoder ticks per revolution
const float motorGearRatio = 298.0; //gear ratio for cables and worm gear



//cable length and encoder count variables
float l1 = 0;
float l2 = 0;
float l3 = 0;
int32_t c1, c2, c3;

const float drumdiameter = 5.5; //Diameter of the winch drums for the cable
    motors - mm
const float origlength = 235; //original length of the Yoshimura module
    including endplates - mm

const float r = 96 ; //length of module without endplate - mm (I think this is
    starting, but I'm always starting at slack?)
const float L0 = 37 ;//shortest length of module in mm - Front module can
    compress to this - might not be accurate.

float s, theta, phi; //arc length, bending angle, bending directions - from
    soft robotics lab paper
typedef struct F_struct{
  float v1;
  float v2;
  float v3;
} F_struct;

F_struct f_values;


/*****************************************************************************
```

```
                                                    SETUP + LOOP

 ********************************************************************************************

void setup()
{
  Wire.begin(); // I2C communication begin
  Serial.begin(9600); // The baudrate of Serial monitor is set in 9600 - lower
      baudrates work best with motor driver modules
  while (!Serial); // Waiting for Serial Monitor to initialize
  Serial.println("\nI2C Scanner");

  //Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  //Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);

  //run once on startup to verify SAMIs connected
  findDevices();

  //register callback function to be called when a message is sent
  esp_now_register_send_cb(OnDataSent);

  // register peer
  esp_now_peer_info_t peerInfo;
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // register first peer
  memset(&peerInfo, 0, sizeof(peerInfo));
  memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }
}
```

```
void loop() {
  //nothing really.... this is all event based
}


/*********************************************************************************************

                                                    I2C FUNCTIONS


 *********************************************************************************************

/**
   Finds all available I2C devices on the current bus - can be used for
      troubleshooting
*/
void findDevices() {
  byte error, address; //variable for error and I2C address
  int nDevices; //number of devices found on I2C bus

  Serial.println("Scanning...");

  nDevices = 0;
  for (address = 1; address < 127; address++)
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address); //begin transmission with each possible
        address
    error = Wire.endTransmission(); //returns 0 for success, 1,2,3,4 for other
        errors


    //if we receive a successful transaction then print out the device's
        address in hex format
    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.print(address, HEX);
      Serial.println(" !");
      nDevices++;
    }
    else if (error == 4)
    {
```

```
      Serial.print("Unknown error at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.println(address, HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");
}

/**
   Reads encoder & current data from a singular motor driver board, stores
      them, and sends them to the external ESP for debugging
   @param address - the hexadecimal I2C address of the motor driver you are
      requesting data from
   @param numBytes - the number of bytes you are requesting from the motor
      driver board over I2C
*/

void requestDataMotors(){
  //Wire.requestFrom(address, numBytes, true);//create a request from an
     individual motor driver board for given number of bytes
    All_Motors.M1.current = F_Wheels[0].get_current();
    All_Motors.M2.current = F_Wheels[1].get_current();
    All_Motors.M3.current = F_Wheels[2].get_current();

    All_Motors.M4.current = B_Wheels[0].get_current();
    All_Motors.M5.current = B_Wheels[1].get_current();
    All_Motors.M6.current = B_Wheels[2].get_current();

    All_Motors.M7.current = Cables[0].get_current();
    All_Motors.M8.current = Cables[1].get_current();
    All_Motors.M9.current = Cables[2].get_current();

    All_Motors.M10.current = Lead_Screws[0].get_current();
    All_Motors.M11.current = Lead_Screws[1].get_current();

    esp_err_t result = esp_now_send(0, (uint8_t *) &All_Motors,
       sizeof(MotorGroup));
    if(result == ESP_OK){
      Serial.println("Test C Pass");
    } else {
      Serial.println("Test C Fail");
    }
```

```
}

void BatteryTestPacket(){
  Battery_Test.Wheel.current = F_Wheels[0].get_current();
  Battery_Test.Screw.current = Lead_Screws[0].get_current();
  Battery_Test.Cable.current = Cables[0].get_current();
  int16_t TC = 0;
  for(int i = 0;i<=2;i++){
    TC += F_Wheels[i].get_current();
    TC += B_Wheels[i].get_current();
    TC += Cables[i].get_current();
  }
  for(int j=0;j<=1;j++){
    TC += Lead_Screws[j].get_current();
  }
  Battery_Test.Total_Current = TC;
  esp_err_t result = esp_now_send(0, (uint8_t *) &Battery_Test,
      sizeof(Battery_Test));
    if(result == ESP_OK){
      Serial.println("Test A Pass");
    } else {
      Serial.println("Test A Fail");
    }
}

void requestDataCables(){
  l1 = calcCablelen(Cables[0].get_position());
  l2 = calcCablelen(Cables[1].get_position());
  l3 = calcCablelen(Cables[2].get_position());

  f_values.v1 = l1;
  f_values.v2 = l2;
  f_values.v3 = l3;

  esp_err_t result_cable_len = esp_now_send(0, (uint8_t *) &f_values,
      sizeof(F_struct));
    if (result_cable_len == ESP_OK){
        Serial.println("Test B Pass");
      } else {
        Serial.println("Test B Fail");
    }

}


/*****************************************************************************
```

```
                                                    ESP-NOW WIFI HELPER FUNCTIONS

 ********************************************************************************************

/**
    Callback function to be executed when WiFi data is sent to external ESP
    prints if message was successfully delivered to know if board received
        message
    @param mac_addr - the mac address of the board that data is being sent to
    @param status - the status of the transaction - success or fail
*/
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  char macStr[18];
  //Serial.print("Packet to: ");
  // Copies the sender mac address to a string
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
          mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
              mac_addr[5]);
  // Serial.print(macStr);
  // Serial.print(" send status:\t");
  //Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
      "Delivery Fail");
}


/**
   Callback function to be executed when WiFi data is received from external
       ESP
   Handles controller command data and commands motor driver boards to move or
       send data based on input
   @param mac - the mac address of the board sending the data
   @param incomingData - the data to be copied into the myData variable - the
       instance of the receiving struct
   @param len the number of bytes received
*/
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData)); //copy content of
      incomingdata variable into mydata variable

  // convert data into an integer to figure out what command was sent from the
      controller
  int commanddata = myData.wifiData.toInt();

  /*********************
     Controller Responses
```

```
 **********************/

/**
   button pad - drive the lead screw and request data
*/
// Screw Controls
if (commanddata == 1) { //Y button - lead screw retract
  Serial.println("lead up");
  LS_Front.set_rpm(30); /// fix
}
else if (commanddata == 4) { //A button - lead screw expand
  Serial.println("lead down");
  LS_Front.set_rpm(-30); /// fix
}

else if (commanddata == 28) { // rear expand
  Serial.println("rear lead screw up");
  LS_Rear.set_rpm(30);
}
else if (commanddata == 27) { //rear retract
  Serial.println("rear lead screw down");
  LS_Rear.set_rpm(-30);
}

else if(commanddata == 29){ // Both Expand
  LS_Front.set_rpm(-30);
  LS_Rear.set_rpm(30);
}

else if(commanddata == 30){// Both Retract
  LS_Front.set_rpm(30);
  LS_Rear.set_rpm(-30);
}


else if (commanddata == 2) { //X button - send data back from all motors
  requestDataMotors();
  requestDataCables();
  //invCableKin(l1,l2,l3);
}


/**
   D-Pad - control the cable motors
*/
```

```
else if (commanddata == 12) { //dpad = 2 - cable 1 down
  Serial.println("cable 1 down");
  Cables[0].set_rpm(30);
  Cables[1].set_rpm(0);
  Cables[2].set_rpm(0);
}
else if (commanddata == 14) { //dpad = 4 - cable 2 down
  Serial.println("cable 2 down ");
  Cables[0].set_rpm(0);
  Cables[1].set_rpm(30);
  Cables[2].set_rpm(0);
}
else if (commanddata == 16) { //dpad = 6 - cable 3 down
  Serial.println("cable 3 down");
  Cables[0].set_rpm(0);
  Cables[1].set_rpm(0);
  Cables[2].set_rpm(30);
}
else if (commanddata == 18) { //reverse cable 1
  Serial.println("cable 1 up");
  Cables[0].set_rpm(-30);
  Cables[1].set_rpm(0);
  Cables[2].set_rpm(0);
}
else if (commanddata == 19) { //reverse cable 2
  Serial.println("cable 2 up");
  Cables[0].set_rpm(0);
  Cables[1].set_rpm(-30);
  Cables[2].set_rpm(0);
}
else if (commanddata == 20) { //reverse cable 3
  Serial.println("cable 3 up");
  Cables[0].set_rpm(0);
  Cables[1].set_rpm(0);
  Cables[2].set_rpm(-30);
}

//Dual Cable Control
else if(commanddata == 21){ //Down right (1 and 2)
  Cables[0].set_rpm(30);
  Cables[1].set_rpm(30);
  Cables[2].set_rpm(0);
} else if(commanddata == 22){ //Down right reverse (1 and 2)
  Cables[0].set_rpm(-30);
  Cables[1].set_rpm(-30);
  Cables[2].set_rpm(0);
} else if(commanddata == 23){ //Down left (2 and 3)
```

```
  Cables[0].set_rpm(0);
  Cables[1].set_rpm(30);
  Cables[2].set_rpm(30);
} else if(commanddata == 24){ // Down left reverse (2 and 3)
  Cables[0].set_rpm(0);
  Cables[1].set_rpm(-30);
  Cables[2].set_rpm(-30);
} else if(commanddata == 25){ // Up (1 and 3)
  Cables[0].set_rpm(30);
  Cables[1].set_rpm(0);
  Cables[2].set_rpm(30);
} else if(commanddata == 26){ // Up reverse (1 and 3)
  Cables[0].set_rpm(-30);
  Cables[1].set_rpm(0);
  Cables[2].set_rpm(-30);
}

//All Cables
else if(commanddata == 15){ // All cables actuate
  for(int i = 0; i <= 2; i++){
    Cables[i].set_rpm(30);
  }
} else if(commanddata == 17){
  for(int i = 0; i <= 2; i++){
    Cables[i].set_rpm(-30);
  }
}

/**
   wheel driving - drive wheels and lead screw based on current sensing
*/

else if (commanddata == 8) { //right bumper - drive forward fast
  Serial.println("drive forward");
  for(int i = 0; i <= 2; i++){
    F_Wheels[i].set_rpm(-30);
    B_Wheels[i].set_rpm(30);
  }

}
else if (commanddata == 9) { //left bumper - drive backward fast

  for(int i = 0; i <= 2; i++){
    F_Wheels[i].set_rpm(30);
    B_Wheels[i].set_rpm(-30);
  }
  Serial.println("drive backward");
```

```
  }

  else if (commanddata == 31 || commanddata == 32 || commanddata == 33 ||
      commanddata == 34 || commanddata == 35 || commanddata == 36){
    BatteryTestPacket();
  }

  /*
  else if(commanddata == 7){
    Homing();
  }*/
  //brake motors
  else if (commanddata == 0) { //brake all motors/do nothing
    for(int i = 0; i <= 2; i++){
      Cables[i].set_rpm(0);
      F_Wheels[i].set_rpm(0);
      B_Wheels[i].set_rpm(0);
    }
   LS_Front.set_rpm(0);
   LS_Rear.set_rpm(0);
  }

}


/*************************************************************************************

                                                SMART MOTOR DRIVER DRIVE
                                                    FUNCTIONS


 *************************************************************************************

/**
   Calculates the length of all cables using encoder counts
   @param enccounts - the integer amount of quadrature encoder ticks
   @return float cablelen - the length of the cable
*/

float calcCablelen(int32_t enccounts) {
  //Serial.println(enccounts);
  int32_t rotations1 = abs(enccounts) / encTicksPerRev; //calculate number of
      rotations of the motor shaft from the encoder wheel
  float rotations = rotations1 / motorGearRatio; //calculate actual number of
      rotations from motor gearbox
  //Serial.println(rotations);
  float deltacablelen = rotations * M_PI * drumdiameter; //calculate the
```

```
    change in cable length from circumference of drum diameter and the amount
    of rotations
  float cablelen = origlength - deltacablelen; //calculate final cable length
  return cablelen;
}

/*
   Calculates the inverse kinematics (bending angle - theta, arc length - S,
      bending direction - phi) of the cables given the cable lengths
   @param l1 - float length of cable 1
   @param l2 - float length of cable 2
   @param l3 - float length of cable 3
*/
void invCableKin (float l1, float l2, float l3) {
  // calc S, theta, and phi


  s = (3 * L0 + l1 + l2 + l3) / 3;

  theta = 2 * sqrt((3 * (l1 * l1) - l1 * l2 - l1 * l3 - l2 * l3) / (3 * r));

  phi = atan((sqrt(3) * (l3 - l2)) / (l2 + l3 - 2 * l1));

  f_values.v1 = s;
  //f_values.v1 = 4;
  Serial.println(f_values.v1);
  f_values.v2 = theta;
  //f_values.v2 = 5;
  Serial.println(f_values.v2);
  f_values.v3 = phi;
  //f_values.v3 = 6;
  Serial.println(f_values.v3);
  Serial.println(" ");
  esp_err_t result_FK = esp_now_send(0, (uint8_t *) &f_values,
     sizeof(F_struct));
    if (result_FK == ESP_OK){
       Serial.println("Sent FK");
     } else {
       Serial.println("Fail FK");
    }


}
```

Bibliography

[1] C. E. Restrepo, J. S. Simonoff, and R. Zimmerman, "Causes, cost consequences, and risk implications of accidents in us hazardous liquid pipeline infrastructure," *International Journal of Critical Infrastructure Protection*, vol. 2, no. 1, pp. 38–50, 2009.

[2] T. M. Rifaai, A. A. Abokifa, and L. Sela, "Integrated approach for pipe failure prediction and condition scoring in water infrastructure systems," *Reliability Engineering System Safety*, vol. 220, p. 108271, 2022.

[3] R. Taiwo, I. A. Shaban, and T. Zayed, "Development of sustainable water infrastructure: A proper understanding of water pipe failure," *Journal of Cleaner Production*, vol. 398, p. 136653, 2023.

[4] M. Mahmoodian and C. Q. Li, "Failure assessment and safe life prediction of corroded oil and gas pipelines," *Journal of Petroleum Science and Engineering*, vol. 151, pp. 434–438, 2017.

[5] Q. Ma, G. Tian, Y. Zeng, R. Li, H. Song, Z. Wang, B. Gao, and K. Zeng, "Pipeline in-line inspection method, instrumentation and data management," *Sensors*, vol. 21, no. 11, 2021.

[6] J. Latif, M. Z. Shakir, N. Edwards, M. Jaszczykowski, N. Ramzan, and V. Edwards, "Review on condition monitoring techniques for water pipelines," *Measurement*, vol. 193, p. 110895, 2022.

[7] A. Carvalho, J. Rebello, M. Souza, L. Sagrilo, and S. Soares, "Reliability of non-destructive test techniques in the inspection of pipelines used in the oil industry," *International Journal of Pressure Vessels and Piping*, vol. 85, no. 11, pp. 745–751, 2008.

[8] M. Safizadeh and T. Azizzadeh, "Corrosion detection of internal pipeline using ndt optical inspection system," *NDT E International*, vol. 52, pp. 144–148, 2012.

[9] A. Verma, A. Kaiwart, N. D. Dubey, F. Naseer, and S. Pradhan, "A review on various types of in-pipe inspection robot," *Materials Today: Proceedings*, vol. 50, pp. 1425–1434, 2022. 2nd International Conference on Functional Material, Manufacturing and Performances (ICFMMP-2021).

[10] J. T. Kahnamouei and M. Moallem, "A comprehensive review of in-pipe robots," *Ocean Engineering*, vol. 277, p. 114260, 2023.

[11] J. Ni, M. Wang, L. Du, S. Bao, Z. Hu, and J. Yuan, "Design of an active suspension mechanism for obstacle traversal of in-pipe robots," in *2024 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1403–1408, 2024.

[12] S. Jatsun and A. Malchikov, "Adaptive suspension system position-force control of wheeled wall-pressed in-pipe climbing robot," in *Synergetic Cooperation between Robots and Humans* (E. S. E. Youssef, M. O. Tokhi, M. F. Silva, and L. M. Rincon, eds.), (Cham), pp. 101–111, Springer Nature Switzerland, 2024.

[13] H. Choi and S. Ryew, "Robotic system with active steering capability for internal inspection of urban gas pipelines," *Mechatronics*, vol. 12, no. 5, pp. 713–736, 2002.

[14] A. Kakogawa and S. M. and, "Design of a multilink-articulated wheeled pipeline inspection robot using only passive elastic joints," *Advanced Robotics*, vol. 32, no. 1, pp. 37–50, 2018.

[15] E. Dertien, M. M. Foumashi, K. Pulles, and S. Stramigioli, "Design of a robot for in-pipe inspection using omnidirectional wheels and active stabilization," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5121–5126, 2014.

[16] D. Chablat, S. Venkateswaran, and F. Boyer, "Mechanical design optimization of a piping inspection robot," *Procedia CIRP*, vol. 70, pp. 307–312, 2018. 28th CIRP Design Conference 2018, 23-25 May 2018, Nantes, France.

[17] M. Kurata, T. Takayama, and T. Omata, "Helical rotation in-pipe mobile robot," in *2010 3rd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 313–318, 2010.

[18] K. Hayashi, T. Akagi, S. Dohta, W. Kobayashi, T. Shinohara, K. Kusunose, and M. A. A. Sani, "Improvement of pipe holding mechanism and inchworm type flexible pipe inspection robot," *International Journal of Mechanical Engineering and Robotics Research*, 06 2020.

[19] S. Savin, S. Jatsun, and L. Vorochaeva, "Trajectory generation for a walking in-pipe robot moving through spatially curved pipes," vol. 113, 2017. Cited by: 15; All Open Access, Gold Open Access, Green Open Access.

[20] T. Ren, Q. Liu, y. Chen, and S. Ji, "Variable pitch helical drive in-pipe robot," *International Journal of Robotics and Automation*, vol. 31, 01 2016.

[21] S. Savin, "Rrt-based motion planning for in-pipe walking robots," in *2018 Dynamics of Systems, Mechanisms and Machines (Dynamics)*, pp. 1–6, 2018.

[22] J. Santoso, E. H. Skorina, M. Luo, R. Yan, and C. D. Onal, "Design and analysis of an origami continuum manipulation module with torsional strength," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2098–2104, 2017.

[23] Y. Sun, Y. Jiang, H. Yang, L.-C. Walter, J. Santoso, E. H. Skorina, and C. Onal, "Salamanderbot: A soft-rigid composite continuum mobile robot to traverse complex environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2953–2959, 2020.

[24] T. V. Jones, G. G. Conard, A. G. Sanchez, Y. Sun, and C. D. Onal, "Lizard: A novel origami continuum mobile robot for complex and unstructured environments," *Robotics Reports*, vol. 3, no. 1, pp. 1–11, 2025.

[25] B. Katz and K. Wheeler, "Continuum locomotive alternative for robotic adaptive-exploration (clara)," tech. rep., Worcester Polytechnic Institute, 100 Institute Road, Worcester MA 01609-2280 USA, April 2022.

[26] B. "Schroeder and D. Pignone, ""continuum locomotive alternative for robotic adaptive-exploration (clara)"," tech. rep., "Worcester Polytechnic Institute", "100 Institute Road, Worcester MA 01609-2280 USA", "April" "2023".