RAINSTORM
Rooftop Assembly Inchworm Network and Swarm Tiling Optimization for Rooftop Maintenance

A Major Qualifying Project Report submitted to the Faculty of

Worcester Polytechnic Institute

In partial fulfillment of the requirements for the Degree of Bachelor of Science

April 28, 2022

Advisors

Carlo Pinciroli, Gregory Lewin, and Markus Nemitz

Submitted By

Eli Benevedes, Dominic Ferro, Samantha Gould, and Filip Kernan

Table of Contents

Table of Figures

Abstract

Roofing is one of the most dangerous construction jobs, accounting for nearly 20% of total construction workplace fatalities in 2019 [1]. Autonomous robotic construction can increase worker safety and the overall workplace efficiency. However, these technologies are often designed for a single project and are not scalable. Therefore, we are applying an inchworm robot platform to shingle a roof with custom data shingles. Our system is a decentralized swarm of inchworm robots designed to collaboratively shingle roofs. These robots are able to communicate and collaborate by storing data within placed shingles. Overall, the use of a decentralized swarm that communicates through the environment will prevent single points of failures and increase reliability.

Introduction

Construction

Construction is one of the largest industries worldwide. At the end of 2020, Statista valued the United States construction industry at $1.36 trillion per year [1]. Over 7 million people in the United States were working in construction as of August of 2021 [2]. However, construction jobs are dangerous, with the construction industry accounting for one in five workplace deaths in 2019 [3]. Industries such as agriculture have seen their productivity increase by almost a factor of two in the last 30 years [4], [5], but there has not been a similar growth in productivity in construction during the same period [6]. This project aims to increase safety and productivity through the introduction of fully automated robotic systems.

Within construction, retrofitting existing structures is a major part of the industry. As of 2011, over 40% of residential buildings within the U.S. had been built before 1970 [7]. Since building codes and energy regulations have changed over time, many of these buildings are not compliant with modern standards. A study commissioned by the Netherlands Ministry of Foreign Affairs suggests that the retrofitting market within Boston, Massachusetts could reach up to $41 billion in the next 10 years [8]. Retrofitting buildings can increase disaster resistance, structural integrity, energy efficiency, and overall safety [9].

Over 6 million employees work in the construction field, equating to about 5% of all the workers in the U.S. [10]. Despite only making up 5% of the workers, in 2019 nearly 20% of total workplace fatalities were in the construction field [11]. Within construction itself, there is a similarly disproportionate number of injuries related to roofing when compared to other construction jobs. Fredericks et al. stated that, “Roofers are about six times higher at risk for fatal occupational injuries than the average worker” [10], with 75% of their fatal injuries coming from workers falling off of roofs. Equipping this workforce with automated systems will reduce the number of people on roofs, and will increase workplace safety in the roofing industry.

Robotics

Robotics can increase safety and productivity on the worksite, and the precision of the work done, by replacing humans with machines. In the mining industry, an industry with high workplace injuries, operator error is estimated to account for up to 88% of all accidents [12]. By replacing people in hazardous situations with autonomous robots, these accidents can be reduced. Robots also increase productivity by completing tasks faster, and reduce the overall cost of production, since robots do not incur a labor cost [13].

Multi-robot systems can increase the scale of the benefits of robotics in a similar way that a team of human workers can be more productive than one human worker. A multi-robot construction system allows for robots to work on different parts of a structure at the same time [14]. A multi-robot system can employ specialized robots, such as robots that ferry materials to other robots that construct a structure [15]. This specialization allows for a large variety of high precision jobs to be accomplished at the same time.

Swarm robotics is a subset of multi-robot systems that takes inspiration from the way large groups of animals, namely insects, behave in nature. These robots operate in a decentralized manner based on local interactions and self-organization, resulting in no one robot knowing what all other robots are doing at any given time. The decentralization and individual computations makes the swarm more reliable as the chance of all of the robots individually failing is lower than one centralized system failing. Additionally, due to the decentralization of information and decision-making, swarm robotics has the potential advantage of scaling well with the size of the problem [16]. Due to the hazardous nature of construction as well as inherent scale of projects, swarm robotics has higher potential for greater success than a singular robotic system in industrial applications.

At this time, there have not been many industrial applications of swarm robotics. Swarm construction researchers have generally developed solutions to build structures in lab environments, not for marketable industrial applications. Swarm construction researchers have generally developed solutions to build structures in lab environments, not for marketable industrial applications. The closest systems instead use swarm based behaviors while also using centralized networking and decision making [17]. More research in swarm robotics will help close this gap within the industrial robotics sector.

Problem Statement

The intent of this project is to design a system to traverse and install shingles on a flat roof. To accomplish this, the team must design a scalable decentralized algorithm to allow multiple inchworms to shingle a roof, and design a robot that is capable of the behavior required by this algorithm

This project will extend previous work done on a construction platform that used bipedal “inchworm” robots that built 3-D structures out of “smart” cubes. Instead of manipulating smart cubes, the robot will manipulate shingles that can store data to communicate with robots in the swarm. The data storage capabilities of the shingles allow the robots to store information in the environment to localize, and assign tasks without the need for a centralized system.

Related Work

Locomotion

For a swarm of robots to shingle a roof there are a few constraints on how these robots must behave. Robots must move around on an inclined roof without falling off, and route around other robots without colliding.

Dusty is a robot used to draw the layout of a building on a concrete bed within millimeters of accuracy [18]. Dusty uses wheels to move around and line up the drawing tool, displaying high accuracy in its form of locomotion. One limitation is that Dusty cannot work on uneven terrain or handle debris in the workspace. Since Dusty is not suited for an inclined workspace and the accuracy is lost when driving over rough terrain, such as roofing shingles that have already been placed, this approach would not work for shingling roofs.

More complex forms of locomotion allow for navigation over uneven terrain. The TERMES robot utilizes whegs, which are a combination between wheels and legs [19]. The capability to move over rough terrain is a benefit when moving over a roof that is under construction. However, systems with wheels and whegs still need to be designed to not tip over or slide down an incline. While whegs solve the issue of uneven terrain, they will be limited by the angle of incline based on the coefficient of friction between the wheg and the roof.

The RoboRoofer is a robot designed to take shingles *off* of a roof. The RoboRoofer company was planning to develop another manipulator to go on top of the chassis that would then also place roofing shingles [20]. The RoboRoofer uses wheels for locomotion, but required a tether that was anchored to the top of the roof to stay on the incline. Since our system is a swarm system, it must be able to move around with other robots on a roof. An anchor and tether system was effective for keeping one RoboRoofer on a roof, but it does not scale well to include multiple robots on the same roof. Increasing the number of robots makes it increasingly difficult to keep each of the tethers untangled.

The Bipedal Isotropic Lattice Locomoting Explorer (BILL-E) is an inchworm robot that was designed by the NASA Ames Research Center to be very mobile with multiple degrees of freedom. The BILL-E robot is able to move across inclined surfaces, including vertical surfaces, by gripping the structure [21]. The robot does not tip over or slip off the structure when it is on an incline, which makes it a good solution for moving on a roof. It can also move across uneven structures. This makes the bipedal inchworm structure of the BILL-E system ideal for shingling a roof.

Smart Structures

To effectively coordinate a robotic swarm, each agent in the swarm must be able to share information with other robots. Direct communication allows for decentralized robots to know where each other are and how they are affecting the environment. The larger the roof that the inchworms need to shingle, the further apart robots may be. Communication degrades over distance, and can be inconsistent causing incomplete or incorrect data to be transmitted. Additionally, the more robots there are trying to communicate over a network, the fewer messages each robot can send due to bandwidth limits. While direct robot-to-robot communication is one solution often used to coordinate a swarm, it is limited by the time it takes to transfer data, and this data may be outdated by the time it reaches a distant member of the swarm.

Another approach is to use the environment as a medium to communicate data. Allwright et al. proposed building blocks built around the idea of stigmergy, a behavior exhibited by colonies of insects like termites and wasps [22]. As worker insects move around and do work, they leave behind pheromones that indicate to future workers where to go or what tasks to perform. The Swarm Robotic Construction System (SRoCS) used blocks with embedded Light Emitting Diodes (LEDs) to simulate pheromones left behind by swarm insects [22]. The LED colors communicated information by changing color as the environment changed. The blocks were updated using Near Field Communication (NFC) as the robots interacted with them to alter the messages communicated by the LEDs. Using NFC to mimic pheromones allows for data to be stored in the environment, which workers can read and update as the environment and tasks change. LEDs are not very desirable for outdoor construction, as the light produced is often low and lighting conditions outside are subject to change. However, the application of stigmergy from this work can be applied to RAINSTORM, by allowing robots to store information on the shingles they are manipulating.

Another project with similar design constraints was the Multi-Agent Robotic Intelligent Assembly (MARIA) project at Worcester Polytechnic Institute [23]. The design of the building blocks in MARIA is similar to the SRoCS system. Unlike SRoCS, the blocks in MARIA are built to directly communicate data to each other, as each face includes both an LED and a color sensor. The data the blocks communicate includes localization data, blueprint data, and requests for blocks to be added to specific faces of the structure. Similar to SRoCS, the LEDs are not optimal for outdoor conditions. However, the kind of data communicated and the way the data is processed on each block can be adapted to robots storing data on shingles. By storing data about the state of the structure on the building blocks of the structure, robots can quickly learn about new parts of the environment without needing to directly observe the changes.

Both SRoCS and MARIA need each element of the structure to contain electronics and a power source. Powering the blocks allows for active communication between blocks, and to change their external observable state. However, on a roof, the cost of powering the shingles outweighs the benefits of these properties. Each shingle needs to be lightweight and inexpensive, which restricts the amount of hardware like batteries that can be added to each shingle. With SRoCS, the NFC technology used does not require this power source. The electronics needed only need to be installed on the robot and can communicate with inexpensive and light data tags.

Swarm Algorithms

Within the research of swarm robotics there are three relevant problems in regards to shingling a roof: task allocation, localization, and collision avoidance. The solutions to these problems will determine the effectiveness of the robotic swarm tiling algorithm.

One problem in swarm robotics is how an individual robot decides which task it should complete. Each robot should choose different tasks to ensure parallelization of the overall task. Additionally, the selection of the tasks is dependent on the order in which they need to be completed and the priority of the task towards the overall goal.

One possible solution is to use a distributed auction system in which robots bid on which task they will complete [24]. Each robot bids on the task that has the highest score for that robot based on a heuristic. Robots can bid on the same task, and the robot with the highest bid gets the task. This approach leads to the best assignment of tasks based on the heuristic, allowing for higher priority tasks to be completed first. However, this method requires a lot of communication between robots and a lot of time to converge on an assignment.

Another option is the distributed bees algorithm, which is based on how bees decide which flowers to seek out. Each robot will obtain a list of all tasks and independently assign probabilities to each task [25]. Each robot then independently decides which task to complete. While this algorithm requires less robot-to-robot communication than other methods, it assumes that multiple robots can work on a single task. When shingling a roof, it is not useful to have two robots attempting to complete the same task.

A third option is an algorithm modeled after optimal transport theory. In this algorithm, robots form groups to work on sets of tasks [26]. The robot that discovers the set of tasks is assigned as the leader. The robots then form groups around the leaders and arrange themselves to accomplish the tasks. None of these three approaches can be used alone for the task of shingling a roof, but they provide good inspiration on how to approach the problem of distributed task allocation.

Knowing where the robot is within a global coordinate system is a problem to solve in any mobile robotic system, but has unique problems and solutions in a swarm. On an empty roof there are minimal landmarks to localize a robot. In the field of swarm robotics, there are several approaches to localization with minimal landmarks, including the Agent model and methods which localize on an individual level.

In the Agent model, robots measure their local position in reference to another robot and, through communication with the other robot, transform it into a global position [27]. Each robot determines its position relative to another robot, until the algorithm reaches an anchoring robot with a known position in the global frame. Linking robots together in a chain of relative positions can lead to error propagation along this chain. While it is great for positioning robots in shapes and patterns, it is not ideal in an environment requiring high accuracy in the global frame. Another downside to this form of localization is it requires a large amount of robot-to-robot communication. This communication requires robots to stay close to each other, limiting the movement patterns of robots in the swarm. While there are algorithms that can decrease the global error [28], the time necessary to calculate the position with less error and pass along the data is too long.

Localization can also be calculated on an individual scale. Each robot in the swarm can calculate its position using a probability based algorithm, like a Bayes filter or a particle filter, and update its probability that it is in a given location based on its observations of the environment. In these filters, the environmental observations occur when interacting with other robots or any smart structures available. Updating the location of the robot when interacting with the environment would still require robot-to-robot communication, but would limit the data being passed along and the number of communications needed to stay accurate.

In an environment with multiple robots, each robot needs to be treated as a dynamic obstacle. Having several moving robots makes path planning more difficult, as it is important to avoid collisions. In a distributed system, robots must be able to communicate with or detect other robots to avoid collisions. One method of solving this problem is to navigate through areas between robots, treating the robots as dynamic obstacles and moving along the path that avoids the robots and goes towards the goal [29].

Alternatively, a robot could operate with a safety boundary and communicate that boundary with other robots [30]. Communicating where each individual robot's boundary is allows for pathing, but would still require a lot of communication between robots. The boundary method uses heading based path planning similar to the previous method.

A third option that allows for pathing and does not require constant communication is using buffered Voronoi cells [31] as seen in Figure 1. The Voronoi cells give each robot an area to work with for path planning, but the method does not require constant communication. Additionally, the Voronoi cells only need to be recalculated when one robot reaches the end of their path within the Voronoi cell.

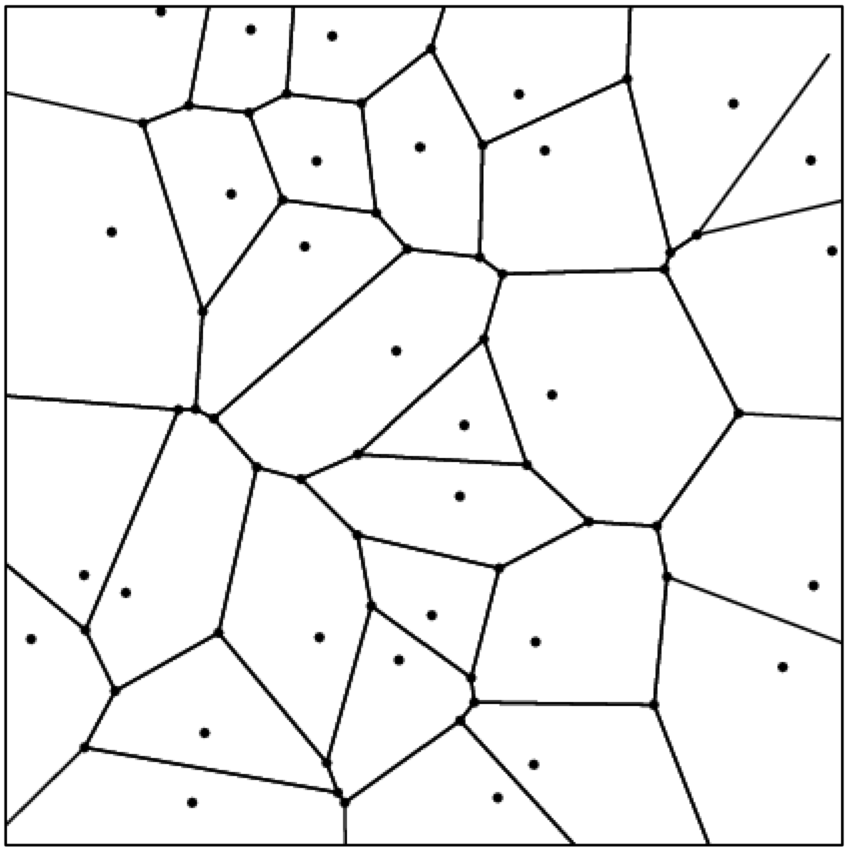

Figure 1: Illustration of Voronoi cells [32, fig 1.5].

Robotic Construction Platforms

Most robotic construction systems focus on building 3-D structures. One example of multi-agent robot construction is the TERMES system developed by a team of researchers at Harvard University. The TERMES system is capable of building structures much larger than the robots themselves [19]. Each robot can carry one block at a time, and place it in front of or on top of another block. These structures are limited in a few ways:

- The robots cannot place blocks in between two other blocks.
- The blocks are only supported by the blocks underneath them.
- The structures must be assembled in a single, linear path.

These limitations heavily restrict the order in which shingles could be placed on a roof by a robot. The required linear path would also limit the number of parallel tasks that the swarm can accomplish, decreasing the overall efficiency.

SRoCS as mentioned above uses specialized blocks to build structures, using magnets to interface between the robot and the blocks, and between adjacent blocks. The magnets allow the blocks to align with each other and the robots' end effectors precisely. Each end effector has an electropermanent magnet that can pick up and drop the blocks by switching the magnetic field on and off [22]. Unlike TERMES, the blocks can be connected to a block on any of its sides instead of just the block beneath it.

The RoboRoofer system, also mentioned above, is designed to remove and install roof shingles. This system attaches to an anchor point at the top of the roof and maneuvers around the roof to attach shingles. The manipulator grabs the shingle at the top of a preloaded stack, places the shingle, and nails the shingle to the roof [20]. This approach to shingling a roof differs from how RAINSTORM approaches shingling a roof. The RoboRoofer is one robot, while our system is a multi-agent system. While the RoboRoofer has the same application as our project, it has not proven it is an effective solution to install shingles on a roof.

Methodology

Problem Formulation

Hardware Requirements

To measure the success of the hardware of the project, the team has identified several goals. These goals are split between manipulation, locomotion, and information sharing.

The robot must be able to manipulate a shingle that is at most 0.9 kg. When attempting to grasp a shingle, the robot must succeed 80% of the time, and it must be able to release the shingle 95% of the time. The robot must be able to pick or place shingles in any location adjacent to the robot's location. Finally, the robot must be able to move between shingles in less than 10 seconds.

For the robot to succeed in shingling a roof, it also must be able to accurately traverse the roof and place new shingles, which also requires the robot to know which shingle it is on at all times. The robot must be able to use the environment to communicate information, so that it knows where it is and can communicate information to other robots.

Software Requirements

Building multiple robots is outside of the hardware scope for this project. Therefore, we plan to study the performance of multiple robots through a custom-made simulator. The simulator must be able to model the relevant state of the robot, including kinematic and dynamic properties, end effector states, and interactions with the environment. To accomplish this, the simulator must be able to model the robot stepping from one shingle to another without failure.

The purpose of the simulator is to compare and evaluate various roof shingling algorithms with different sized swarms or roof configurations. Some simulation components must run faster than real time to efficiently evaluate shingling algorithms. While a simulation is running, data must be collected regarding robot faults, task allocation across the swarm, the total time it takes the swarm to shingle the roof, the amount of time robots are idling, and path planning and execution behavior. This data will help quantify algorithm and swarm performance so that different scenarios can be objectively compared.

The algorithm that controls an individual inchworm must run independent of the decisions made by other inchworms, this will allow us to model decentralized behavior. A centralized implementation would require a central computer making decisions, or one of the robots to direct the others. Either solution creates a single point of failure. If the central system failed, all of the robots would fail. With a decentralized swarm, if a single robot fails, the remainder of the swarm would still have the ability to complete tasks.

Previous Hardware Design

This project was based off of the hardware developed by the SMAC team [15]. The robot was a 5 degree of freedom robot with two end effectors. Each link was printed using a FDM printer, using PLA plastic. The end effectors required the robot to screw into the working environment. The inchworm was able to grab onto cubes using the same screw manipulator, with four circles interfacing into the blocks or environment to ensure a known orientation.

The microcontroller running the robot was a Teensy 3.6 [33]. This was connected to AMS AS5048 magnetic encoders [34] on each of the motors driving the 5 primary joints using I^2C. A combination of Castle CCBECs [35] and MP1584 step-down converters [36] in parallel to step down voltage to the motors and microcontroller. All of these electrical components were either reused or replaced for the final RAINSTORM hardware.

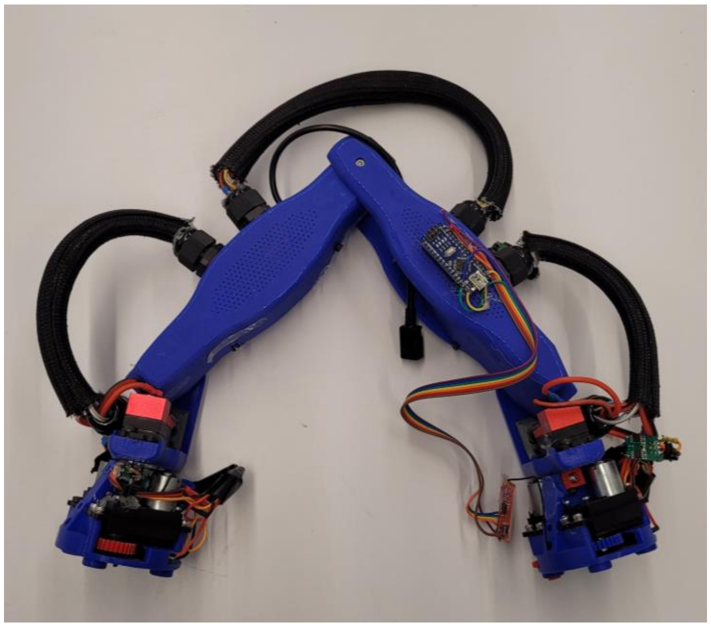
Figure 2: The SMAC hardware and electronics.

Hardware Design

When manipulating roof shingles, the design of the tile itself was considered a top priority. The roof shingles cannot have any screw holes or indentations once fully installed to form a watertight seal. Therefore, the manipulator must interface on a flat surface or grasp at the edges of the tile. The development of the manipulator occurred in parallel to the Civil Engineering MQP team's development of the roof shingle itself. To keep the end effector both simple and versatile, the team chose a magnetic interface inspired by the MARIA project [23].

An inchworm robot requires a strong, constant magnetic field as it walks across the environment to plant an end effector and support the weight of the robot. The magnetic field must then be disrupted to lift an end effector and move it to a shingle position. An electromagnet would solve the issues of a non-permanent magnetic field being required to walk, but it would also heat up and slowly lose its strength if it is activated for an extended period of time. Permanent electromagnets (PEMs), when compared to electromagnets, have inverse logic, meaning the magnetic field is on when there is no power, and it gets disrupted when current is run through the coils. Since the magnets must be on for long periods of time, permanent electromagnets draw far less current, making it a good choice for the end effector.

The design of the inchworm's links had to be altered from SMAC's original design. The chosen design is shown in Figure 3. To support more electronics, the team widened the middle links. To prevent intersections between these new links and existing hardware like encoders and motor casings, the team increased the length of links 1 and 4 and decreased the height of the end effectors. Links 1 and 4 were extended by the same amount that the end effectors were shrunk, so that kinematics stayed the same from the SMAC project.

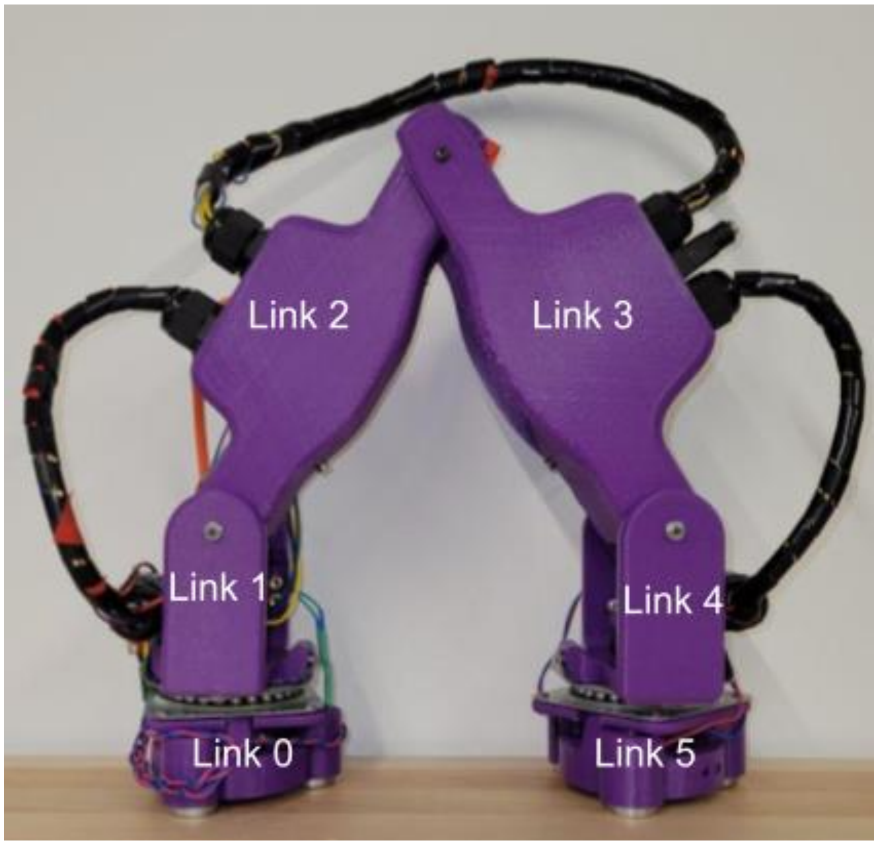

Figure 3: Robot assembly with labeled links.

Upon running some basic movement tests of the robot, the team found that the robot was not stable, as both links 1 and 4 (the “ankles”) were cantilevered at the point where they connected to the motors in link 2 and 3, respectively. Figure 4 shows the old and new ankle designs. This led the team to redesign link 1 and 4 to be supported on both sides of link 2 and 3, instead of just the mounting position on the motor. Shoulder bolts were used to create an attachment point to links 2 and 3 on the motor horn’s opposing side.

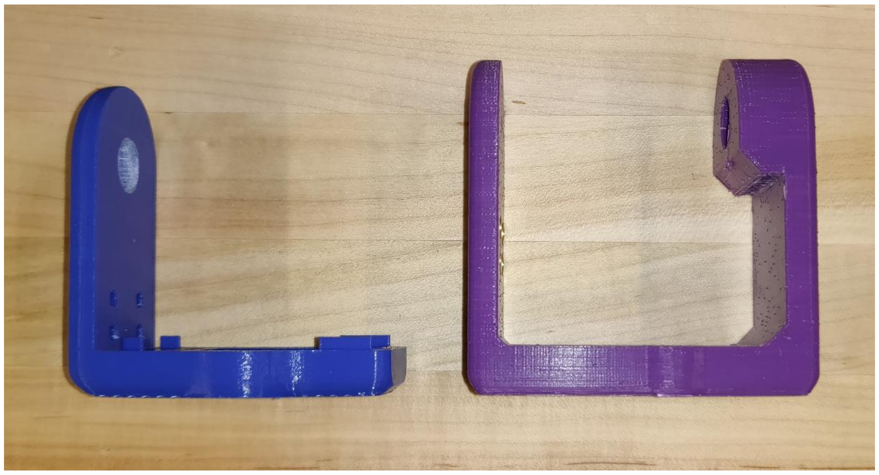

Figure 4: The old and new ankle designs, on the left and right respectively.

Electrical Repairs & Expansion

Due to the changes made to the end effector, the team found it necessary to rewire the robot. While testing the system after rewiring, multiple motors and encoders showed signs of failure. The team replaced all failing components, and evaluated and cleaned up poor electrical connections within the robot.

Due to the number of electrical faults and troubleshooting necessary to find each of the faults, the team found it necessary to expand the space within each of the links to store the electronics. With more space between each component, testing components for proper functionality became much easier. The increased space is seen in areas off of links 2 and 3 in

Figure 2 as compared to Figure 3. Even with the increased space, there were still issues presenting themselves from the encoders. After measuring the current and voltage from each of the pins the team found that increased pressure on the top of the encoders was causing the faults. To prevent future faults with this known problem, we expanded the width of each of the links to give more space above the motors and encoders.

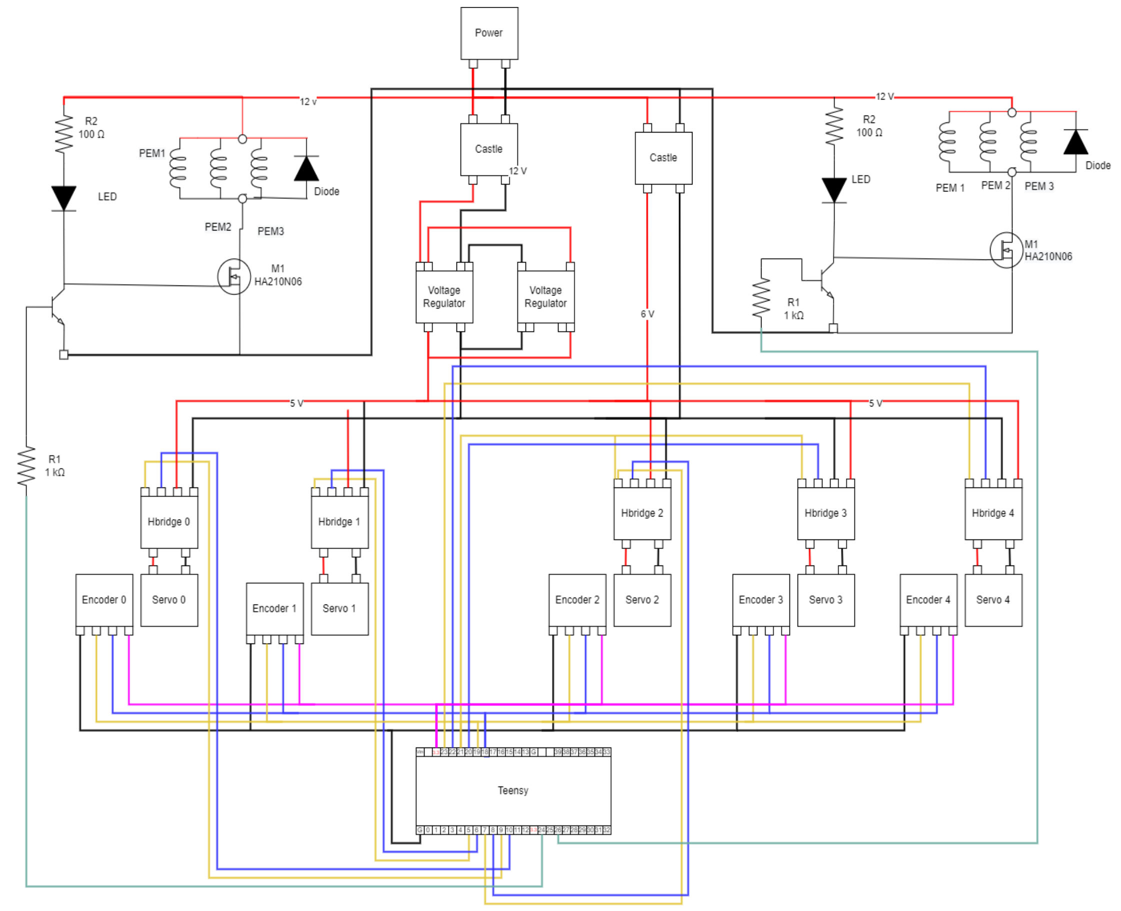

Figure 5: Diagram of electronics within the robot.

Software Design

Embedded Software

The embedded software runs on the Teensy 3.6 on the robot using Arduino embedded C++ code. The embedded software is responsible for running a PID controller for each of the motors, and reporting faults and debug statements for monitoring. The embedded software uses the `ros_serial` library [37] to publish and subscribe to ROS topics.

The inchworm control stack commands the robot through the following messages: `JointState`, `MagnetState`, `PIDConsts`, and an `Int32` message that worked as a heartbeat. The `JointState` messages update the targets of the different motors. The `MagnetState` message updates the magnet states on the robot, enabling and disabling the magnetic fields on the end effectors. The `PIDConsts` messages can be used to update the PID constants on the robot to tune the PID control on the robot without re-uploading the embedded code to the Teensy. The heartbeat message increments as it sends signals. This is used to check the connection between the inchworm control stack and the embedded software on the robot.

The embedded code publishes the current joint angles of the motors on the robot. It also publishes the current states of the magnets and the current PID values when they update. These are `JointState` messages and `PIDConsts` messages, respectively. When the embedded code receives a heartbeat message from the inchworm control stack, it publishes a heartbeat message to the inchworm control stack as an `Int32`. Additionally, the embedded software publishes debug and fault messages to the inchworm control stack.

Simulators

The simulator is split into two parts: a physics simulator and a swarm behavior simulator. The purpose of the physics simulator is to both communicate with the robot, and to show a robot swarm physically shingle a roof. The purpose of the swarm behavior simulator is to show the overall swarm behaviors, while not having to simulate the physical system.

Physics Simulator

The physics simulator is built on the Gazebo physics simulator [38]. The roof, shingles, and inchworm are fully modeled. A shingle depot is also modeled which allows robots to retrieve new shingles to place on the roof. Initially, we used MoveIt to do the inverse kinematic calculations to find the necessary poses for the robot to move to [39]. Once we had the joint angles needed, we stored the poses in a dictionary and used quintic trajectory planning to move between poses. The `ros_control` stack acts as the interface for hardware, actuating joints based on desired positions or provided joint trajectories [40].

Within the physics simulator is a magnet simulation library. This is built on a custom Gazebo plugin that simulates magnet forces and joints to attach models to each other [41]. Our team made modifications to this plugin to generalize the system slightly. We have defined magnet mounting points on each end effector, the top of the shingle, the bottom of the shingle, and many points on the roof. This allows the end effectors to connect to the shingles, and shingles to connect to the roof. The inchworm and the shingle couple this way, so the inchworms can manipulate and traverse across the environment by attaching and detaching itself from shingles. Once the inchworm places a shingle, the magnets on the bottom keep it in place, so that other shingles may be installed around it.

This simulation is also responsible for controlling the real physical robot. The `ros_control` package abstracts away hardware control, meaning that on startup it will either connect to a simulated or a real robot. For physics simulator tests, it will connect to the robot simulated by Gazebo, and command it to move around. To test real hardware, `ros_control` connects to the robot over a serial connection, and updates desired joint values to control the embedded software. Due to this abstraction, all of the high-level responsible for deciding how inchworms should complete a task is completely independent of whether a simulated robot or a real robot is being run. This allowed the team to develop robot control algorithms before the hardware was fully ready.

Swarm Behavior Simulator

The swarm behavior simulator is designed as a ROS node [42], which can then communicate with the physics simulator. This simulator controls all high-level inchworm behaviors, which are then passed to the physics simulator. Additionally, a visualization shows the current state of the roof and inchworms. This visualization allowed us to experiment with different swarm behaviors and tiling patterns, without having to simulate the physical systems.

The roof is represented as a 2D array of shingle objects. This allows the swarm behavior simulator to correctly model all values that are being stored on the rfid chips which are on the physical shingles. Because each odd shingle row is offset by half a shingle, the coordinates in the 2D list utilize a hexagonal coordinate system. The specific hexagonal coordinate system that we are using is shown in Figure 6. This system is called the *even-r hexagonal coordinate system* [43]. The origin is in the top left corner. The coordinates are column, row.

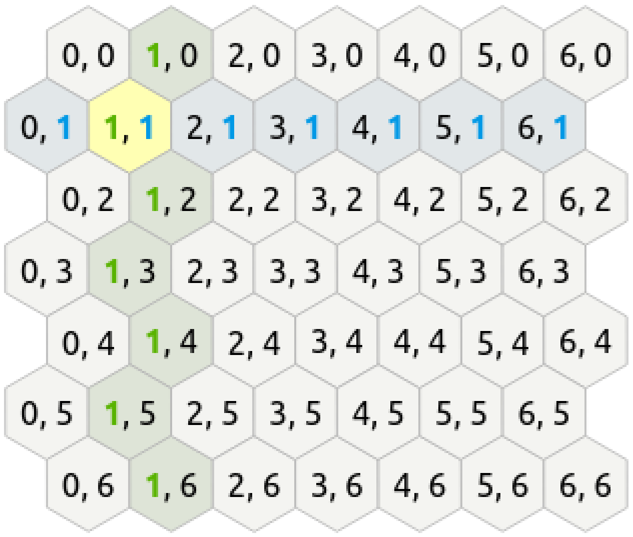

Figure 6: Even-r hexagonal coordinate system [43].

Each inchworm is capable of storing data on the shingles which the inchworms can read and write to. The roof also contains the shingle depots. These depots exist on the left and right edge of the roof and provide inchworms with new shingles. For the purpose of experimenting with many different types of shingling algorithms, the shingle depots are able to move up along the roof. The roof also holds an occupancy grid for the inchworm locations. This is used to represent the physical location of the inchworms on the roof.

The inchworms contain information relevant to where it is localized in the world, as well as a copy of the roof shingle occupancy grid and the shingle depot locations. These data points are updated when the inchworm reads data from a shingle, or it probes a shingle location. This results in the inchworm making decisions with imperfect information. The simulations of the inchworm decisions and actions take place using two separate functions. This is to enable synchronization with the physics simulator. Each inchworm that is also created with an id and an inchworm behavior. This will allow us to easily run tests with different shingling algorithms, by changing the inchworm behaviors.

When the inchworm performs operations on its copy of the roof to determine its next task, it converts all coordinates into cubic hexagonal coordinates. This allows the simulator to accurately compute distances and determine neighbors on the hexagonal grid. The coordinates are converted by projecting the hexagonal grid onto a cube grid. This results in the grid having three axes and allows for the calculating distances and calculating neighbors.

When the swarm behavior node makes calls to the inchworms, it first randomizes the order in which it will iterate through the inchworms. This not only makes the simulator more realistic, as inchworms can become out of sync, but also prevents deadlocks. The node then loops through all inchworms and gives them the chance to make individual decisions. The way the decision function is designed will allow for almost independent decisions, with the exception of collision avoidance. If an inchworm is already working on another task, it will not be allowed to make a new decision. After all inchworms have made decisions the swarm behavior node will allow for each inchworm to utilize one tick of simulated time where it is able to make changes to the environment. This is where the inchworm will be able to read information from shingles, move shingles around, pick up shingles from the shingle depot and move from shingle to shingle. If a simulation is being run along with a physics simulation, each action will take as long as the physics simulation requires, otherwise constant time values will be used to simulate the different movements.

Algorithm Design

Within designing different algorithms to shingle the roof, there are a number of constraints that we must satisfy given the roof and the current robot design. First, due to the geometry of the roof shingles, each new shingle that is placed must have two shingles below it for support. This is because shingles overlap, and a shingle cannot be installed underneath one that has already been installed. Second, when inchworms are moving shingles around, they must move them along the roof frontier. This is due to the inchworm not being able to store shingles on its body. This results in a movement pattern where an inchworm will have to move a shingle forward one place and then move itself forward. The last constraint that the behaviors must satisfy is that the last shingle to be placed on the roof must be adjacent to the shingle depot. Otherwise, there is no path to bring a shingle from the shingle depot to the final install locations.

To limit the scope of the problem, we chose a few variables to be centralized in these algorithms. The first centralized component is a form of collision avoidance. It is imperative that two inchworms do not attempt to move onto the same shingle, as that will cause direct collisions and will prevent the inchworms from being able to shingle the roof. To solve this problem we made each shingle mutually exclusive to one inchworm. This would allow an inchworm to claim a shingle location that it intended to move to and prevent other inchworms from planning to use that spot. The second factor that the team decided to centralize was the order in which shingles would be placed. The patterns used to shingle the roof are seen in Figure 7-9. This ensured that all constraints discussed above would be satisfied, and allowed for experimentation of different orders and behaviors. These centralizations allowed us to focus on the behaviors which ran on the inchworms.

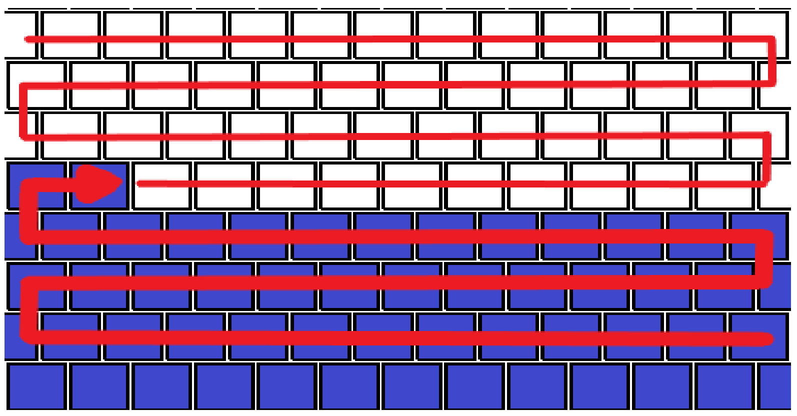

Figure 7: Pattern 0, inspired by how an ox plows a field.

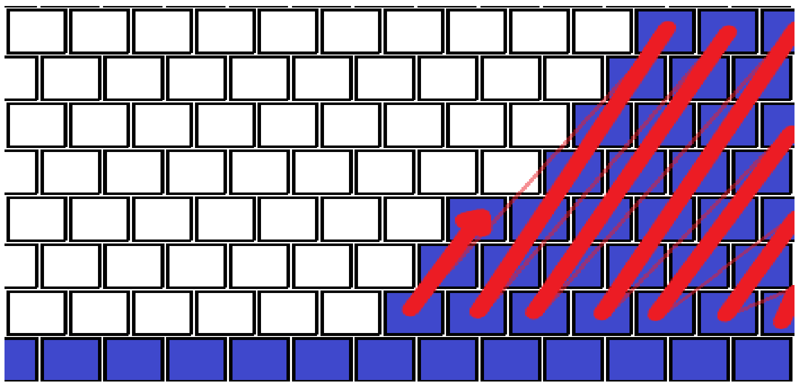
Figure 8: Pattern 1, the diagonal pattern.

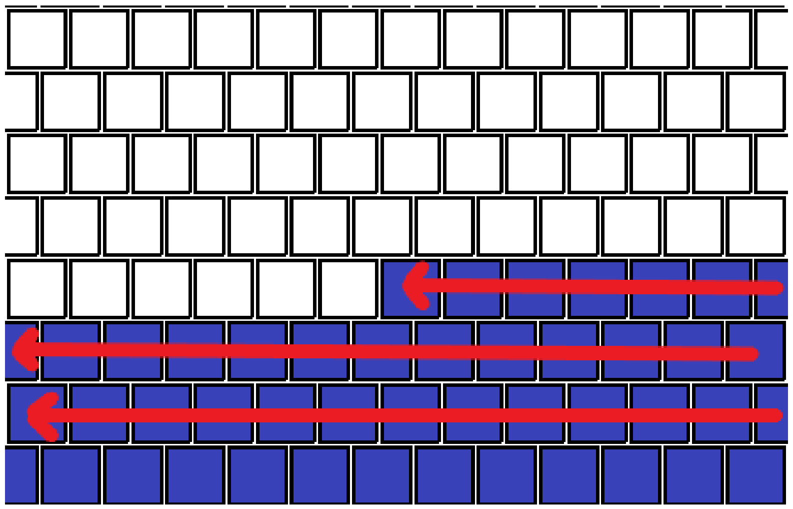
Figure 9: Pattern 2, row by row, always installing shingles at the end of the row first.

The inchworm core was designed as a behavior tree, seen in Figure 10. This tree terminates with different actions that the inchworm may make. The actions of an inchworm include: picking up a shingle from the depot, moving towards an install target, moving a shingle, installing a shingle, and exploring. The inchworms decide on which behavior they will enter based on their internal copy of the roof, and a sampling of a random uniform distribution. The inchworms use the random sampling when deciding if it should explore, when they are moving toward the shingle depot. This ensures that inchworms will receive updates about their environment, and inform other inchworms about changes to the environment.

These actions are lightweight in opposition to the move shingle and move towards target actions. These actions utilize a path planning algorithm. When an inchworm is adjacent to a shingle the inchworm enters the move shingle state and calculates a path for the shingle to follow. Since an inchworm can only move a shingle along the frontier between installed shingles and uninstalled space, the inchworm starts the path planning by calculating the frontier based on its personal map. The frontier in all patterns is only a layer of shingles, so a greedy path planning algorithm based on the distance from a given coordinate to the target position is more than adequate to find a valid path.

With a path for the shingle to be shuffled along, the inchworm calculates its next action based on the next point in the shingle's path. If the inchworm is capable of moving the shingle forward, it then performs a check to determine which end effector to use and moves the shingle. When the inchworm is not in the valid shuffling position it performs the checks described above and then uses the information in the move to target state. The inchworm continues in this pattern until it reaches the install location.

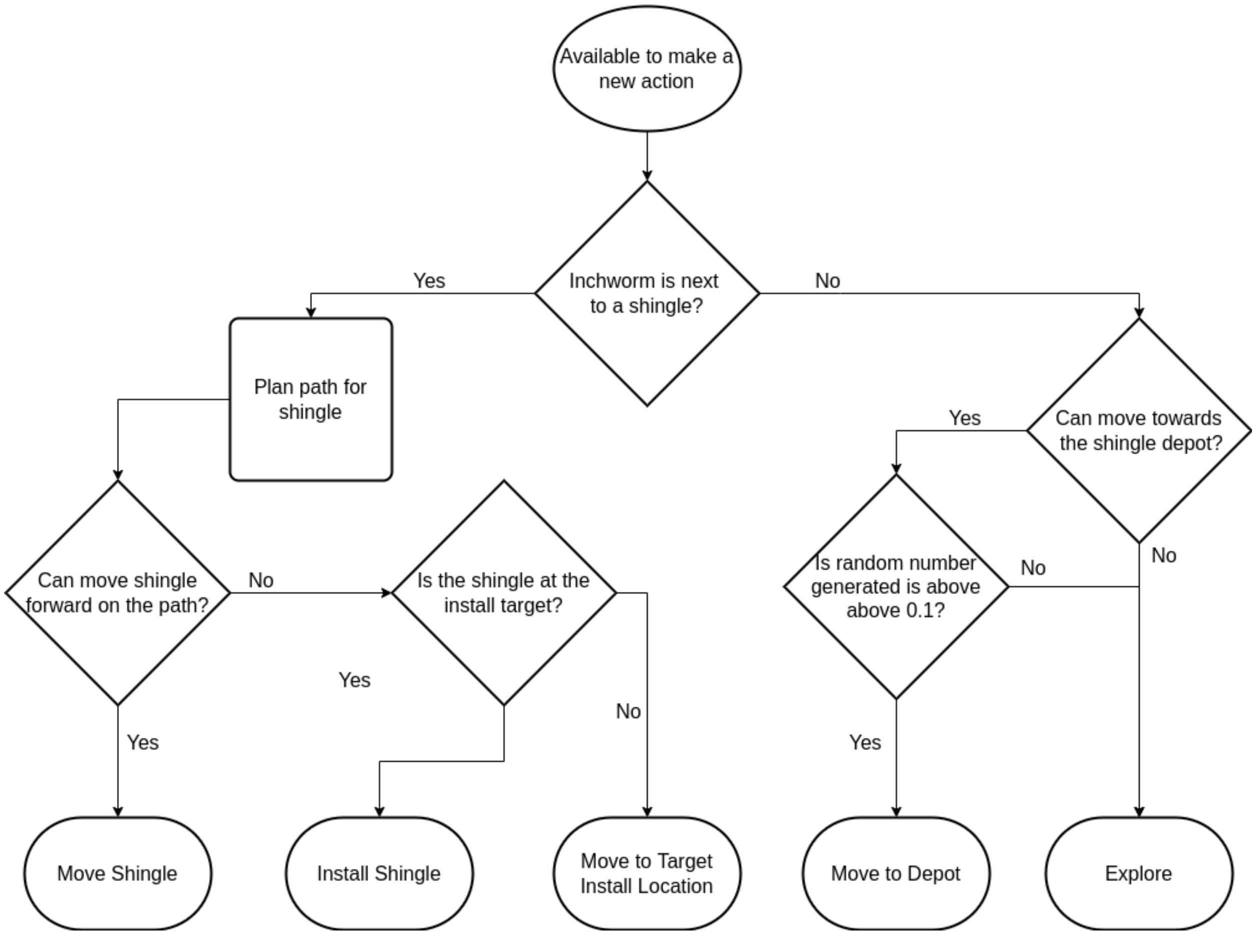
Available to make a new action
Inchworm is next to a shingle?
Yes
No
Plan path for shingle
Can move towards the shingle depot?
Yes
No
Can move shingle forward on the path?
No
Is the shingle at the install target?
Is random number generated is above above 0.1?
No
Yes
No
Yes
Yes
Move Shingle
Install Shingle
Move to Target Install Location
Move to Depot
Explore

Figure 10: Inchworm behavior tree.

The inchworms can store a few different types of information about their environment. Each shingle can store its current coordinate on the roof, as well as its current state: uninstalled, placed, and installed. This allows the inchworms to know if it is safe to use the shingle as an anchor. Additionally, each inchworm can store the state of its neighbors. This data can only be updated by another inchworm. This allows inchworms to communicate the location of placed shingles and other installed shingles. Inchworms can use this data along with the known constraints of the system to rebuild a valid roof. Another form of data that the shingles can store is the most recent target of another inchworm. When an inchworm reads a recent target from a shingle that is further in the shingle order than it currently is, it will rebuild its roof such that it will have the same target. With the combination of these two data points, over time the inchworms map will converge to the actual map.

To determine the efficiency of the algorithm and different shingling order we recorded several different data points from each run. We created a test script that would run all configurations of inchworms for a given roof configuration. For every run we collected the number of steps each inchworm takes, as well as the number of times each inchworm performs an action from a set we are interested in. This data is then recorded in a file structured in comma-separated value format (CSV) for post processing.

Experimental setup

To calculate the required strength of the Permanent Electromagnets (PEMs), the team did a static analysis at the worst-case scenario seen in Figure 11, called the “plank position”.

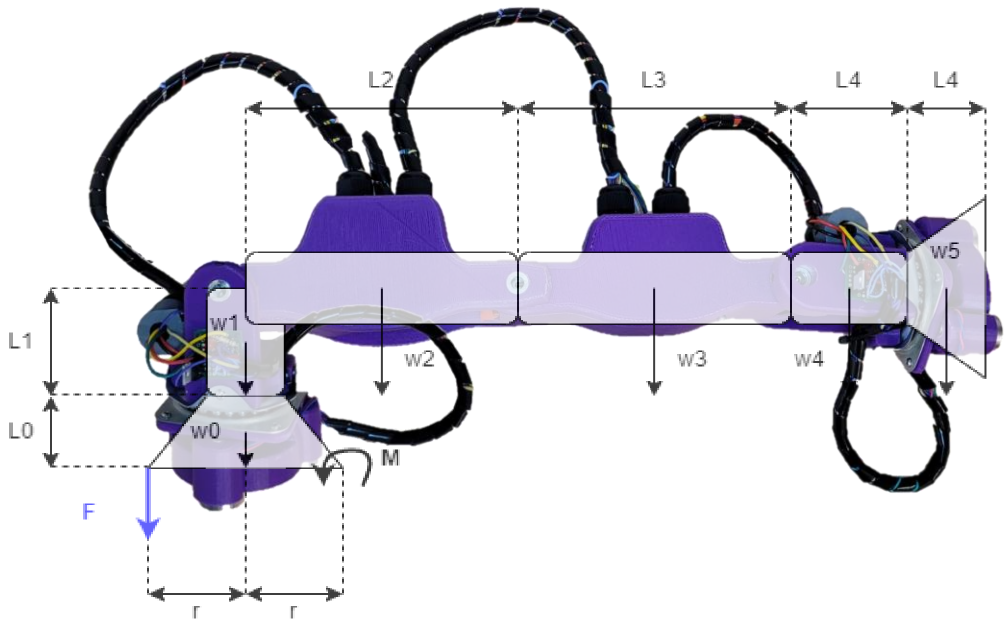

Figure 11: Free body diagram of robot in worst-case scenario.

The end effector is made up of three PEMs arranged in an equilateral triangle. In the highest load scenario, pictured above, the force to counteract the moment, M, was supplied by a singular magnet. In reality, the two other magnets would also provide a small force to counter the moment. The calculations in Appendix B determine the maximum forces of the magnet. The force needed to support the robot with the magnets in the worst-case scenario without a shingle is 30.28 N and the force needed with a shingle is 31.57 N.

Based on these calculations and the voltage rails on the robot, the team chose magnets that ran on 12V and provided 45.02 N of force. The strength of the magnets exceeds the necessary force required in the worst-case scenario.

Upon arrival the team tested the PEMs to ensure that their magnetic field strength was sufficient. One ⅛" steel plate was clamped to a table. The magnet was then placed on the plate and had its own clamp attached to it to form a handle as seen in Figure 12. A spring scale measured the current force of the magnet. The team applied an upwards force, recording the maximum force recorded by the spring scale before the magnet came off of the steel. This experiment was repeated three times to create an average maximum force before removal. The experimental holding force of the magnet was found to be roughly 46.7 N, around 2.7 N greater than specified by the supplier.

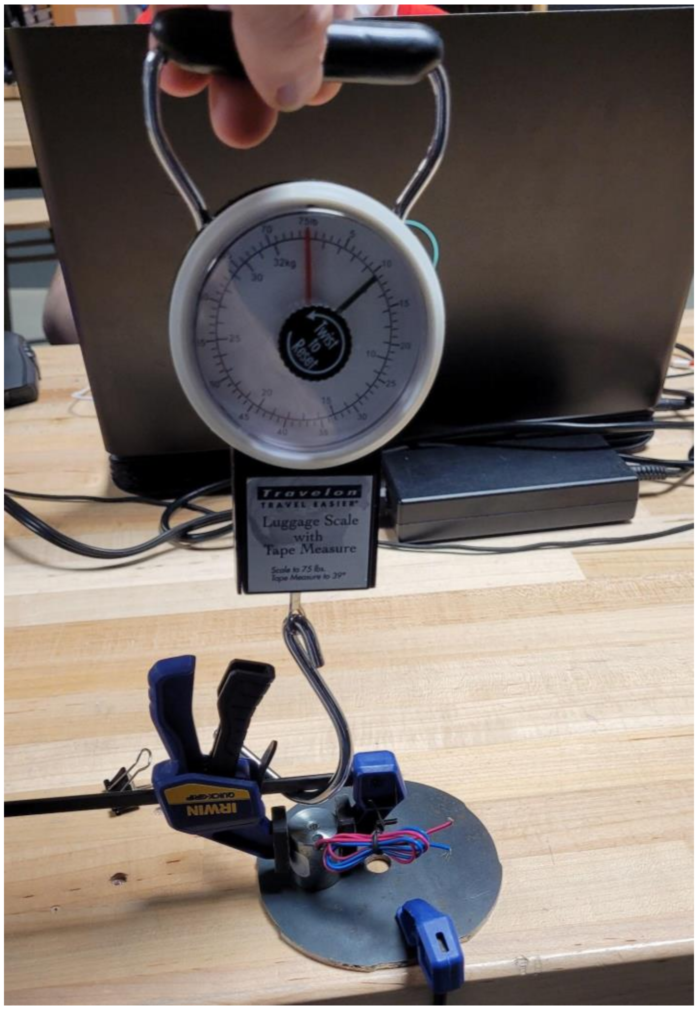

Figure 12: An experiment to evaluate the vertical strength of the PEMs.

The team also tested the PEMs shear holding force. The steel plate was fixed to the table once again, and the permanent electromagnet was placed on top. The spring scale then pulled the magnet across the plate, instead of away from the plate. The experiment was repeated three times to get an average of 18.2 N of translational holding force.

To drive these magnets, the team designed a control board. There are two main constraints that the design needed to meet. First, the Teensy must be able to control this circuit with 3.3V and very low current. Second, when the magnets were disabled, the collapsing magnetic field could not cause problems in the other electronics on the robot. Figure 13 shows the final schematic.

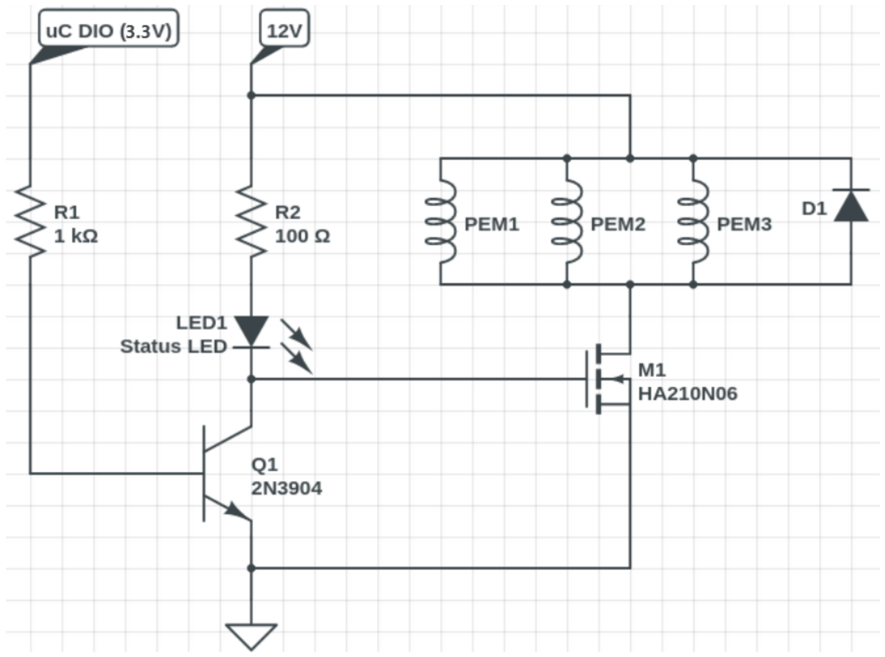

Figure 13: Magnet control circuit.

The two resistors were chosen such that the voltage applied to the transistor base and MOSFET gate were sufficient to allow current to flow through each. The LED indicates the current state of the magnetic field, where the LED being on means the field is enabled. Finally, the diode is to prevent the voltage flyback caused by the collapsing magnetic field. Figure 14 shows the voltage across the magnets when disabled without the flyback diode, while Figure 15 shows the voltage when there is a flyback diode.

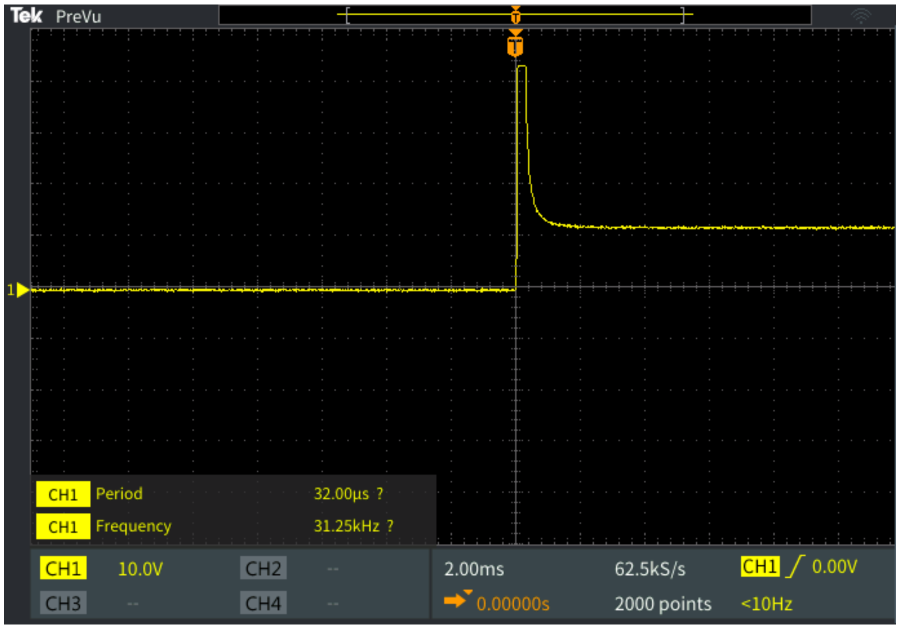

Figure 14: Voltage across magnets when disabled without a flyback diode.

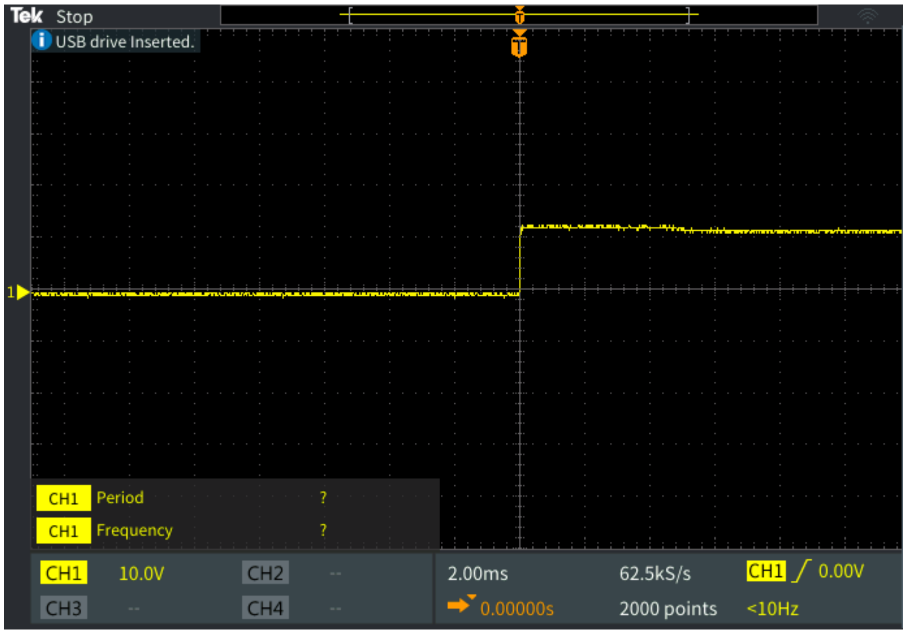

Figure 15: Voltage across magnets when disabled with a flyback diode.

Results

Swarm behavior simulator

The swarm behavior simulator is able to simulate roofs of an arbitrary width and height measured in shingles, and a number of inchworms equal to half the number of shingles across the width of the roof. The simulation was run on roofs ranging from 10 to 30 shingles in both height and width with all possible swarm sizes. This shows that the shingling algorithm is reliably able to shingle a roof.

To determine the differences between the patterns, we calculated timing estimates for all possible swarm sizes, based on the actions the inchworms take in simulation. The longest action an inchworm can take is moving, so the swarm behavior simulator counts the number of times inchworms move.We then multiplied the move count by 16 seconds, an estimate of how long the real robot takes to perform movements. Figure 16 shows that all shingling patterns benefit from an increased swarm size. The differences between pattern performance are only visible at low inchworm counts. With smaller swarm sizes, the left to right takes the least amount of time to complete. Additionally, the diagonal pattern appears to benefit the most from an increase in the size of the swarm.

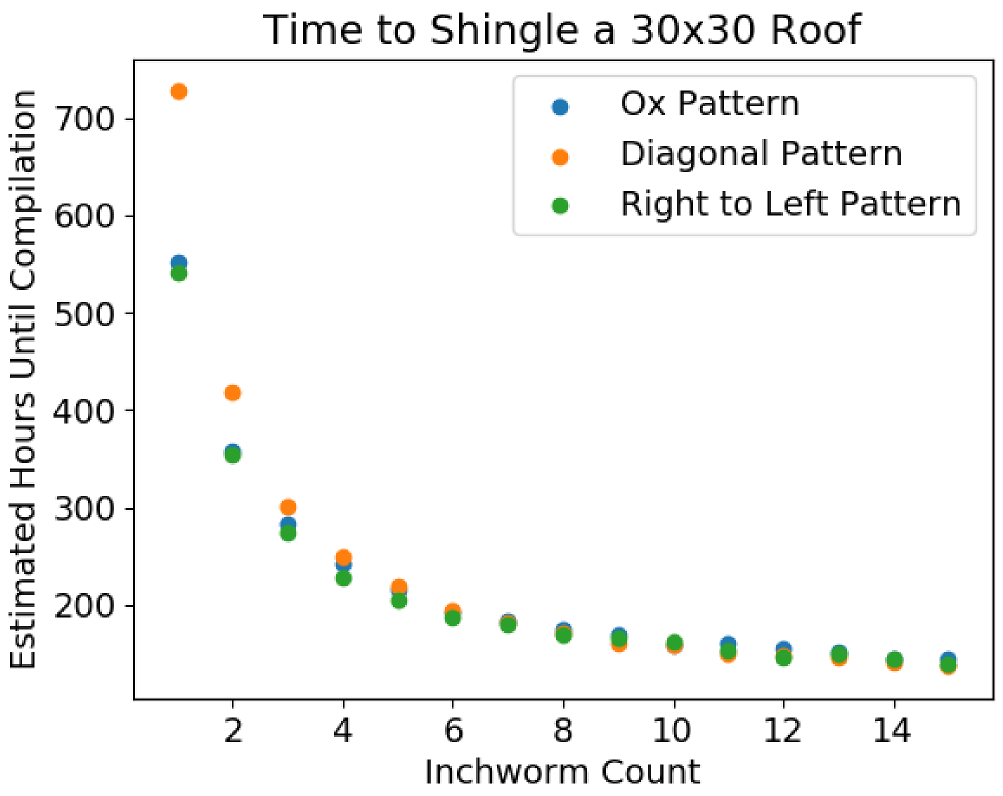

Figure 16: Timing estimates on shingling a roof with different patterns.

We also collected data on the distribution of actions taken by inchworms based on the shingling pattern. The action counts were then averaged across inchworms to determine the distribution of the actions taken by the average inchworm in a given swarm size. Figures 17-19 show that the percentage of time that an average inchworm spends moving a shingle decreases and the time moving towards an install target, moving to the depot and exploring increases as the inchworm count increases across all patterns. The diagonal shingling pattern does not experience as strong of an increase in exploration as the ox plowing and right to left patterns. Furthermore, the diagonal pattern does not see a noticeable increase in time spent moving towards the shingle depot when compared with the ox plowing and right to left patterns. This increase may be due to a larger frontier between installed and empty space on the roof on the diagonal pattern, allowing for more inchworms to operate on the frontier without running into each other.

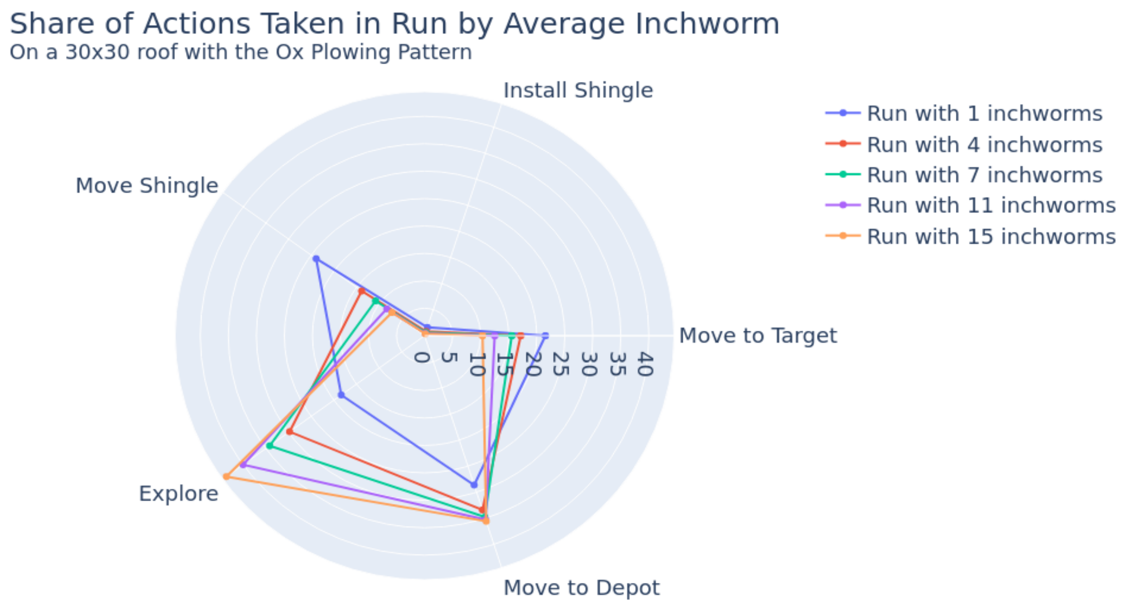

Figure 17: Share of actions taken by the average inchworm on a 30x30 roof with the ox plowing pattern.

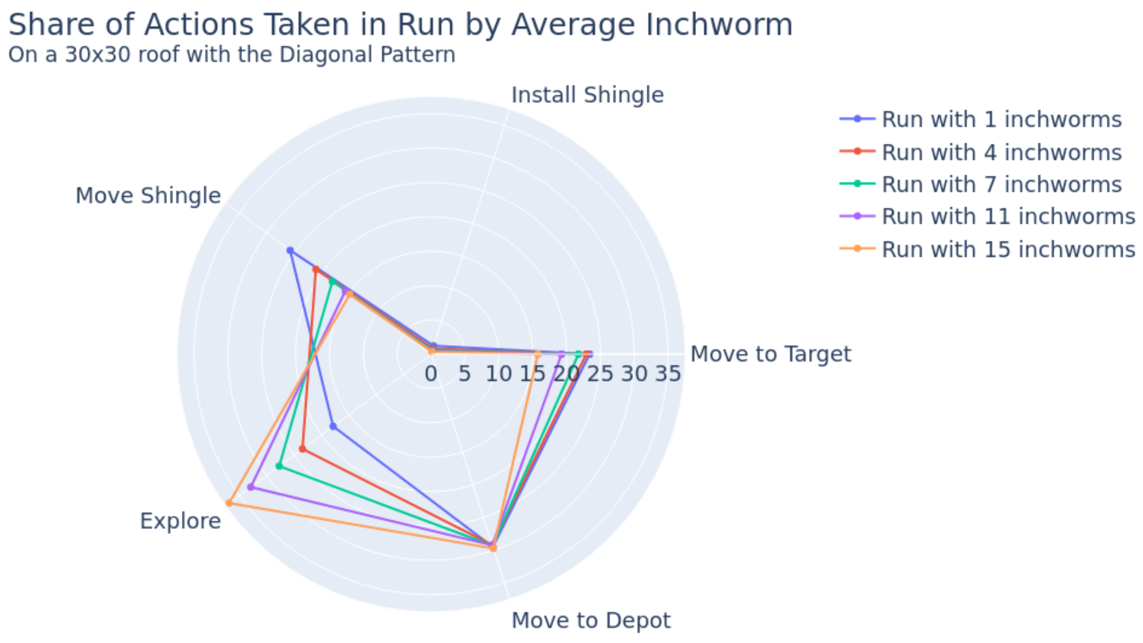

Figure 18: Share of actions taken by the average inchworm on a 30x30 roof with the diagonal pattern.

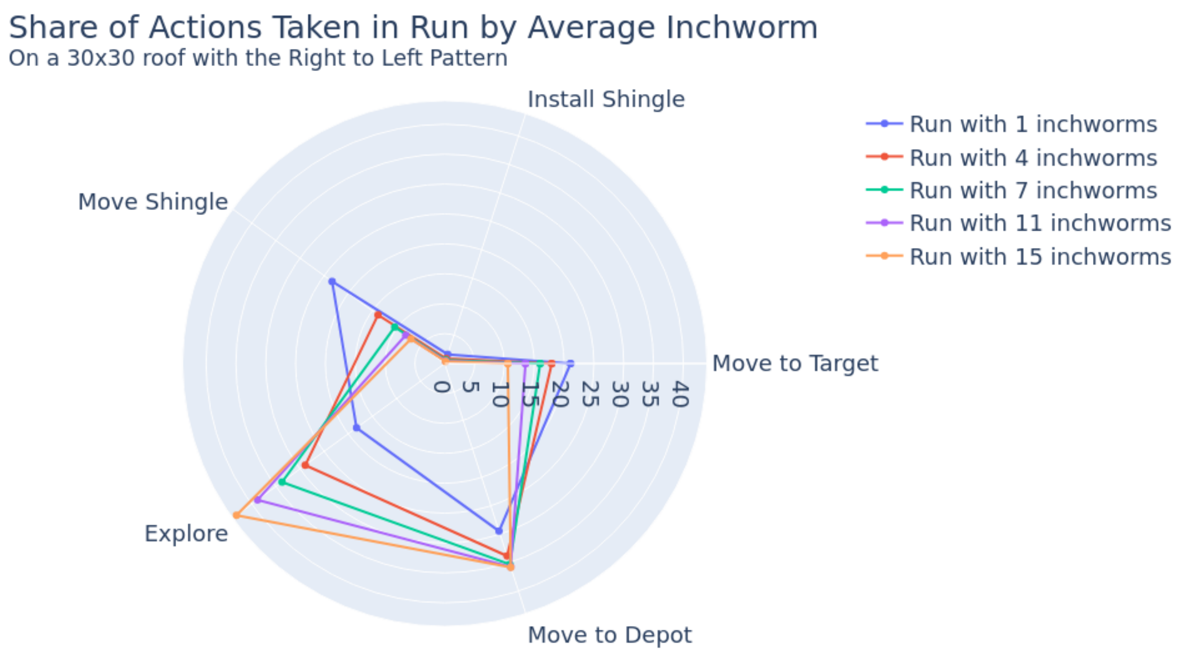

Figure 19: Share of actions taken by the average inchworm on a 30x30 roof with right to left pattern.

Physics Simulator

The physics simulator can simulate a full roof environment with multiple robots on it. The performance of the modeled robot is very similar to the real hardware. Motion times are approximately the same, and the actions an inchworm can take (moving to a neighbor and manipulating a shingle) are all implemented. Some collision meshes needed to be slightly modified to overcome imprecision within the underlying Gazebo physics simulator, but these modified meshes have no measurable impact on the overall simulator performance. The robots in this simulator can move their free end effectors to any neighbor, and switch which end effector is connected to the roof. The robot can also pick up any shingle adjacent to its current location, and move it to any free adjacent location. Figure 20 shows an inchworm moving about a simulated 5x5 roof in Gazebo.

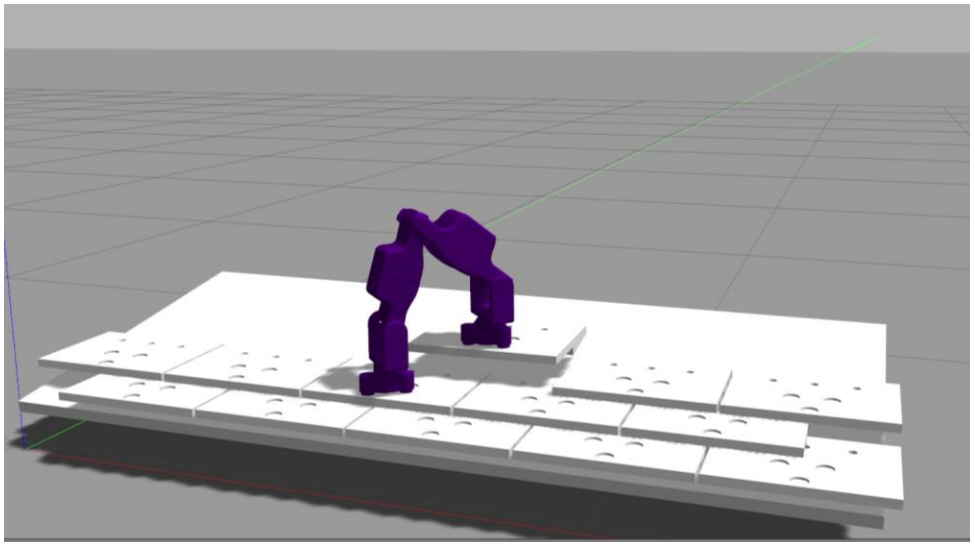

Figure 20: One inchworm shingling a 5x5 roof in the physics simulator.

During initial testing, the magnet simulator was too computationally expensive to simulate large roofs. In the original implementation, all shingles could connect to all locations on the roof, and both end effectors of all inchworms could connect to all shingles. This required many Gazebo API calls per second to update the magnet simulation state, and slowed down Gazebo considerably. After modifications to reduce the number of possible connections, the magnet simulation no longer has as large of an impact on simulator performance. However, its computation time still scales with the number of inchworms and shingles simulated.

The computational performance of the simulation is high enough to observe and evaluate simulations in close to real time. Gazebo uses the concept of Real Time Factor (RTF), which is the ratio of the speed of the simulation to real time. If the RTF is 0.5, then the simulation takes twice as long to run as the modeled robot would. The performance of the simulator is correlated with the number of inchworms present, and the size of the simulated roof. Table 1 shows Gazebo's RTF at various roof sizes and inchworm counts on a machine with an RTX 3070 graphics card and an Intel i7-10700F CPU. To find these values, the team started the simulation, and RTF was recorded once it stabilized, without commanding the inchworms to do anything.

To fully shingle a 5x5 roof with a single inchworm through predetermined joint movements, the simulator took 45 simulation minutes, or one hour in real time. However, integration tests between the physics simulation and the swarm behavior simulator take much longer, as inchworms do not have perfect information about their environment.

	1 Inchworm	2 Inchworms	3 Inchworms	4 Inchworms
5x5 Roof	0.99	0.99	▨	▨
6x6 Roof	0.96	0.94	0.91	▨
7x7 Roof	0.63	0.5	0.41	▨
8x8 Roof	0.22	0.20	0.19	0.18

Table 1: Gazebo RTFs for various inchworm counts and roof configurations. A roof must have a width of twice the inchworm count to support that many inchworms.

Embedded Software

The software running on the robot performs its job well. While the system is not responsible for any high-level logic, it must process and act upon commands quickly. In final testing of the hardware, the embedded software consistently performed without fault. While testing and debugging the hardware, the robot was capable of reporting debug and fault information about the status of its sensors and processing. However, the team disabled this functionality to conserve serial bandwidth for the `ros_serial` bridge.

Integration

Figure 21 shows how the different simulators communicate. The swarm behavior simulator commands the movements and actions for the physics simulator in the Gazebo simulation. The physics simulator creates an action server for each inchworm, which the swarm behavior simulator can control with action clients and goals [44]. The physics simulator does not provide feedback as to the success of these actions, except to indicate when the action is complete. There are several possible ways to walk the inchworm across the roof, but multiple inchworms were only able to avoid collision when lifting their end effectors directly above the planted foot. Shingling a 6x6 roof with 2 inchworms by walking this way in simulation takes roughly 7 hours of sim time and 10 hours of real time. Due to RTF and duration constraints, the team only tested two inchworms on a 6x6 roof in simulation to demonstrate functionality, as shown in Figure 22.

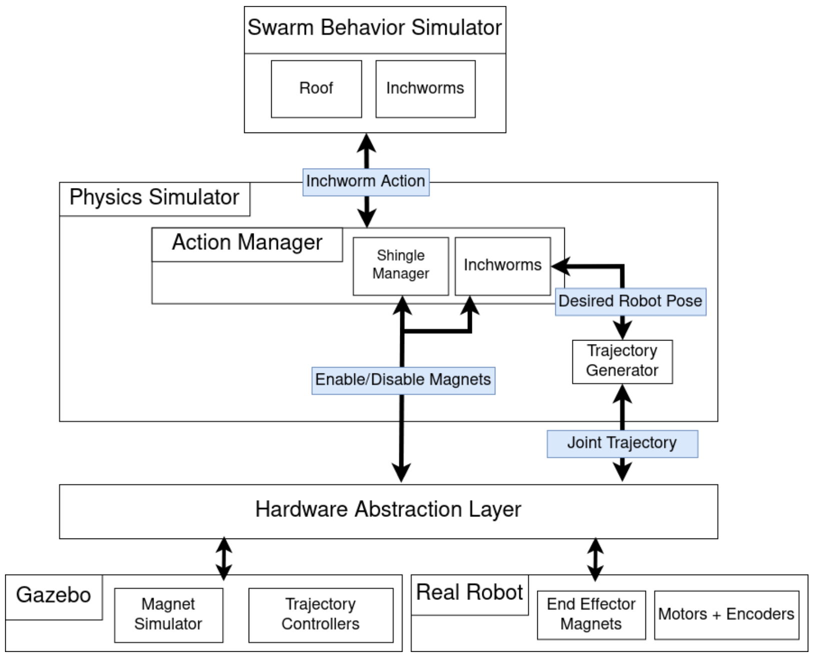

Figure 21: Simulator communication pipeline.

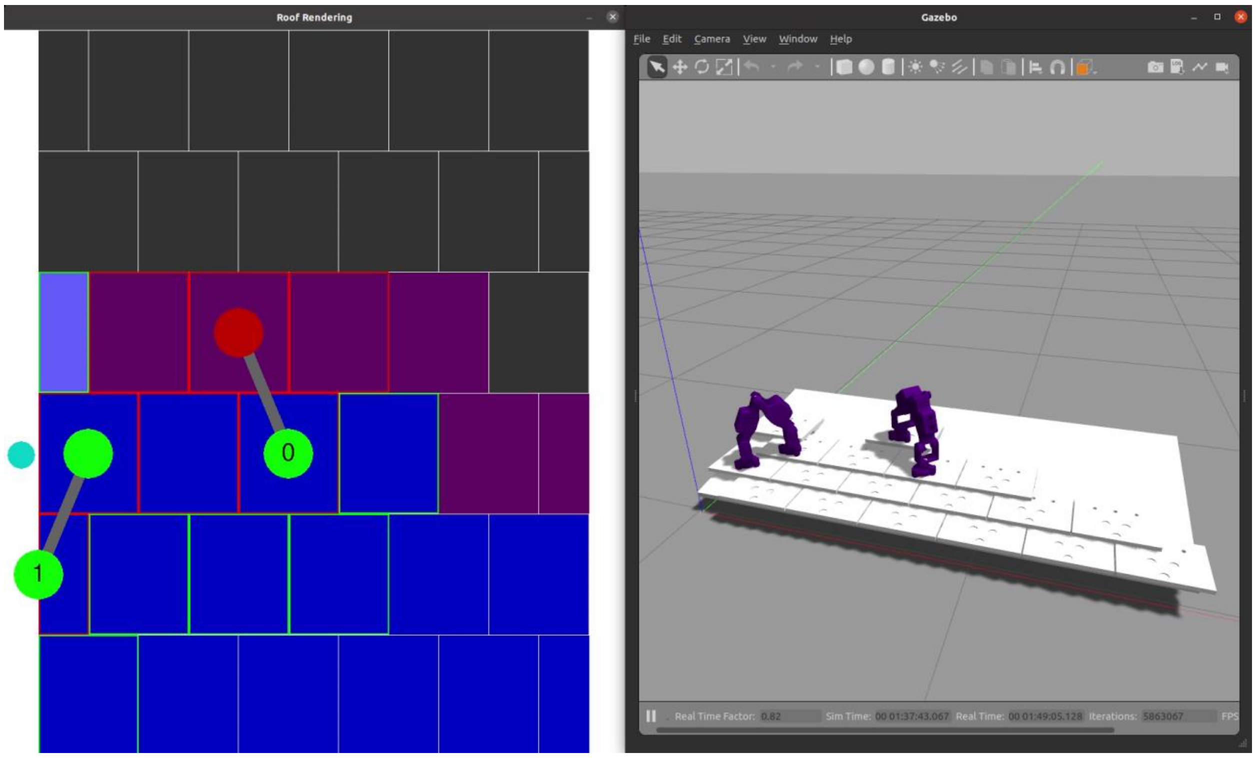

Figure 22: Swarm Behavior Simulator communicating with the Physics Simulator.

Hardware

The custom circuit designed to control the PEMs demonstrated full functionality. In all of the tests run the circuit did not fail to enable or disable the magnets when intended. The PEMs in combination with this circuit fulfill the hardware requirements outlined for picking up and dropping a shingle.

We were able to experimentally demonstrate the inchworm robot walking across shingles and picking up and placing shingles on a model roof. Figure 23 shows a robot walking across the roof. It starts with one end effector planted on the shingle to the left, then it moves the other end effector on top of the shingle to its right. After planting that end effector on the new shingle, it disables the magnets on the old shingle and spins around to plant the old end effector on the next shingle to the right. During several tests the robot was able to transfer its base end effector from one shingle to another in under 10 seconds.

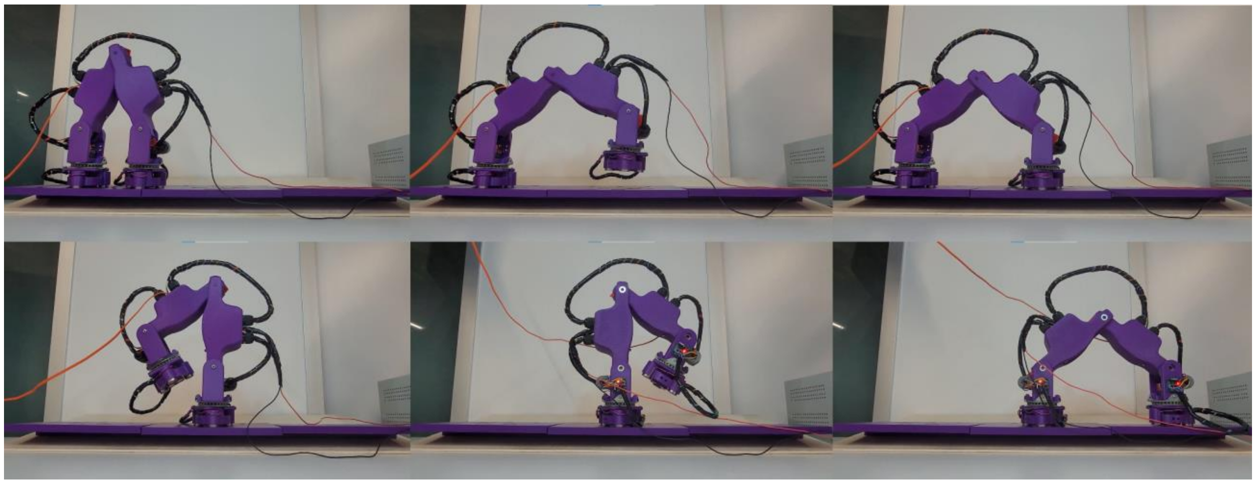

Figure 23: Inchworm walking across a shingle.

In Figure 24, we demonstrate the robot picking up, carrying, and then placing a shingle on a model roof. The robot moves its end effector above a shingle, disables its magnets, then places its end effector on the shingle. After re-enabling the magnets, the robot connects to the shingle, lifts up the shingle, then moves the shingle into the next position. After the shingle is in the new position, the robot disables its magnets to release the shingle. The robot was able to reach both horizontally adjacent shingles when demonstrating walking, and all the vertically adjacent shingles when demonstrating placing. Through these two hardware demonstrations, the robot can perform the atomic actions needed to shingle a roof.

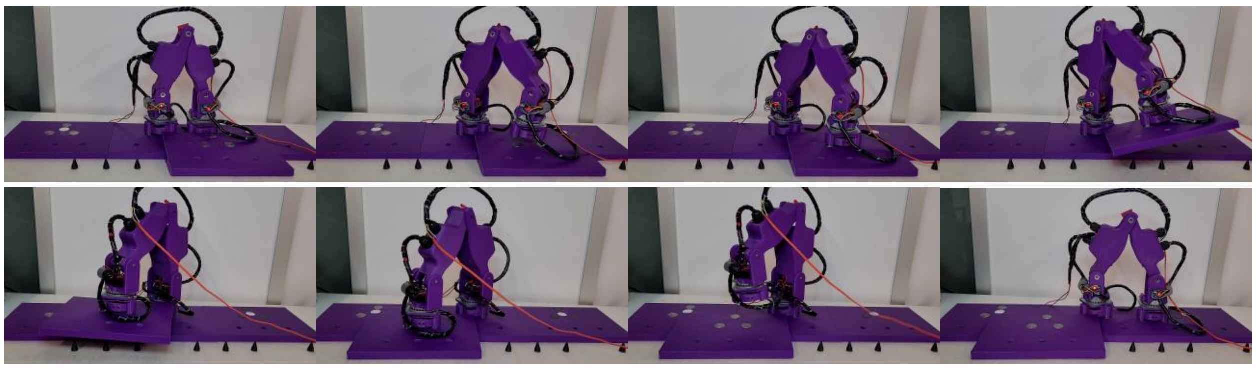

Figure 24: Inchworm picking up and placing a shingle.

In both simulators, inchworms can move to any adjacent shingle, and can move a shingle from one neighboring location to another. Because the physical inchworm can perform these two kinds of actions, given more time and a larger roof, the inchworm would be able to shingle a roof with these actions alone.

Conclusion

The team successfully modified the existing inchworm design created by the SMAC team to manipulate and traverse shingles. The compilation of these actions show that the robot is able to complete the actions necessary to shingle a roof. Additionally, the team designed a novel distributed shingling algorithm that runs on a swarm of inchworms to shingle roofs. The algorithm was experimentally shown to shingle roofs of various sizes and with different shingling patterns in simulation. The action pipeline allows for the shingling algorithm to connect to the inchworm control stack running within the physics simulator. This allowed the swarm behavior simulator to connect to the Gazebo physics simulator, demonstrating that modeled robots can complete the actions needed to shingle a roof. Because the robot can also replicate these actions, we have shown that a real inchworm robot could shingle a roof.

Lessons Learned

We learned several lessons while working on this project, the biggest being that hardware is more difficult than simulation. Electrical noise was present on the voltage lines, causing noise to propagate throughout the system, including the I^2C bus. This noise caused faults in the embedded software, preventing the encoders from functioning until we resolved the issue. We recommend that future teams use communication protocols that are more noise resilient, such as RS-485 [45] or CAN [46]. On top of the electrical issues, there were additional mechanical issues that would have benefited from more thorough verification of the functionality of legacy hardware components.

We also encountered significant issues in software that were caused by poor initial assumptions. Initially, the physics simulator was only built to run one robot. This required us to refactor significant portions of the software stack once we wanted to simulate multiple robots. We advise teams to carefully consider and document the assumptions made about the system to anticipate and resolve these problems more rapidly.

Future Work

Integration

Currently the swarm behavior simulation feeds actions to the physics simulation without any feedback. We suggest implementing feedback from the physics simulation to allow for recovery behavior. As mentioned above in the integration results, the inchworms have a tendency to collide when moving in the optimal walking movement. These collisions can knock robots off of the roof. The benefit of a distributed swarm is that the rest of the swarm can continue to work if any of the individual robots fail. Without feedback the swarm behavior simulator hangs, waiting for a response or completion of an action from an inchworm the simulator is not aware has fallen off the roof.

Electrical Work

There are a few changes that should be made to the electronics within the robot. First, we recommend that PCBs should be developed for some components. This would allow for the wires within the center links to take up less space. This would not only facilitate debugging of electrical issues, but also would enable expansion of the sensing and communication abilities of the robot. We would also suggest the addition of a button on the magnetic control circuit across the mosfet to temporarily disable the magnets when the robot is powered.

Robot to Robot Communication

The robot does not support any ability to communicate with other robots in its environment. This has been implemented in the swarm behavior simulator to experiment with different distributed algorithms, but this capability does not exist on the physical robot. We suggest that a future team implement limited robot to robot communication to allow for some real-time communication between nearby inchworms. As part of this, a future team may want to build a second inchworm to validate that features like collision avoidance works between two robots in a decentralized manner.

Allowing a Robot to Store a Shingle

One major area of work which would allow for more experiments with shingling algorithms would be to create a method for the robot to store a shingle on itself and be able to move about at the same time. By allowing for the robot to move across the roof without needing to stop to shuffle a shingle about, this would remove one of the major constraints on the shingling algorithms explored above. New behaviors could then be defined and potentially allow for different robots to have different roles in the swarm.

Installing Shingles

The current end effector has no way of installing the newly placed shingles. This would require the development of a fairly complex mechanical device that would be able to install each shingle once it is in the correct location. This mechanism would then have to be integrated into the end effector, alongside the current magnets and control circuits.

Acknowledgments

We would like to thank Professor Torbjorn Stansen Bergstrom for providing crucial advice in the fabrication of the metal disks embedded in the first iteration of the roof shingles. We would like to thank our colleague James Mitchell for his help with cutting out the metal disks for the second iteration of the roof shingles with the plasma cutter. We would also like to thank Kimberly Coudrey, William Crist, and Hannah Hirsch, the members of the civil, architectural, and environmental engineering MQP team we worked in conjunction with to develop the roof shingle.

Appendices

Appendix A: Magnet Force-Torque Calculations

Worst-case position of the robot

$$M = -(r * w0) - (r * w1) + ((L2/2 - r) * w2) + ((L3/2 + L2 - r) * w3) + ((L4/2 + L3 + L2 - r) * w4) + ((L5/2 + L4 + L3 + L2 - r) * w5)$$

$$F = M/(2 * r)$$

F = 30.28 N

Worst case with worst case shingle:

$$M = -(r * w0) - (r * w1) + ((L2/2 - r) * w2) + ((L3/2 + L2 - r) * w3) + ((L4/2 + L3 + L2 - r) * w4) + ((L5/2 + L4 + L3 + L2 - r) * w5) + ((Ls/2 + L5 + L4 + L3 + L2 - r) * ws)$$

$F = M/(2 * r)$

F = 31.57 N

Where: r = radius (6.9 cm), L0 and L5 are the length of the foot (5 cm), L1 and L4 are the length of the ankle (5.5 cm), L2 and L3 are the length of the beam (16.35 cm), w_i is the weight of link i, Ls is the length of the shingle (2.9 cm), and ws is the weight of the shingle (0.215 kg).

Appendix B: Software GitHub Links

Simulator code: https://github.com/Ronoman/inchworm_ros
Embedded code: https://github.com/ssgould/inchworm_embedded

Bibliography

[1] Statista Research Department, "Topic: U.S. construction industry," *Statista*. https://www.statista.com/topics/974/construction/ (accessed Sep. 15, 2021).

[2] Bureau of Labor Statistics, "Employment, Hours, and Earnings from the Current Employment Statistics survey (National)," *Bureau of Labor Statistics*. https://data.bls.gov/timeseries/CES2000000001 (accessed Sep. 15, 2021).

[3] Occupational Safety and Health Administration, "Commonly Used Statistics." https://www.osha.gov/data/commonstats (accessed Sep. 15, 2021).

[4] E. Njuk and Resource and Rural Economics, "A Look at Agricultural Productivity Growth in the United States, 1948-2017," *U.S. Department of Agriculture*. https://www.usda.gov/media/blog/2020/03/05/look-agricultural-productivity-growth-united-states-1948-2017 (accessed Sep. 15, 2021).

[5] S. Francis, "Global agriculture robot market expected to quadruple in size by 2026," *Robotics & Automation News*, Jun. 14, 2021. https://roboticsandautomationnews.com/2021/06/14/global-agriculture-robot-market-expected-to-quadruple-in-size-by-2026/43841/ (accessed Apr. 26, 2022).

[6] "Measuring productivity growth in construction," *U.S. Bureau of Labor Statistics*, Jan. 2018. https://www.bls.gov/opub/mlr/2018/article/measuring-productivity-growth-in-construction.htm (accessed Apr. 22, 2022).

[7] M. Sarkar, "How American Homes Vary By the Year They Were Built," U.S. Census Bureau, Washington, DC 20233, Demographic Directorate 2011–18, Jun. 2011.

[8] Netherlands Enterprise Agency RVO and CADMUS, "Deep Energy Retrofits Market in the Greater Boston Area," Government of Netherlands, Oct. 2020.

[9] "Retrofitting old buildings to make them earthquake safe | News Center," *Georgia Tech News Center*, Oct. 20, 2014. http://news.gatech.edu/news/2014/10/20/retrofitting-old-buildings-make-them-earthquake-safe (accessed Oct. 29, 2021).

[10] T. K. Fredericks, O. Abudayyeh, S. D. Choi, M. Wiersma, and M. Charles, "Occupational Injuries and Fatalities in the Roofing Contracting Industry," *J. Constr. Eng. Manag.*, vol. 131, no. 11, pp. 1233–1240, Nov. 2005, doi: 10.1061/(ASCE)0733-9364(2005)131:11(1233).

[11] U.S. Bureau of Labor Statistics, "Number and rate of fatal work injuries, by industry sector," *Graphs for Economic News Releases*. https://www.bls.gov/charts/census-of-fatal-occupational-injuries/number-and-rate-of-fatal-work-injuries-by-industry.htm (accessed Sep. 20, 2021).

[12] P. Lever, "Automation and Robotics," in *SME mining engineering handbook*, P. Darling, Ed. Littleton, CO, United States: Society for Mining, Metallurgy, and Exploration, 2011, pp. 805–824.

[13] R. Bloss, "Collaborative robots are rapidly providing major improvements in productivity, safety, programing ease, portability and cost while addressing many new applications," *Ind. Robot Int. J.*, vol. 43, no. 5, pp. 463–468, Jan. 2016, doi: 10.1108/IR-05-2016-0148.

[14] B. Khoshnevis, "Automated construction by contour crafting—related robotics and information technologies," *Autom. Constr.*, vol. 13, no. 1, pp. 5–19, Jan. 2004, doi: 10.1016/j.autcon.2003.08.012.

[15] C. Wagner *et al.*, "SMAC: Symbiotic Multi-Agent Construction," *IEEE Robot. Autom.*

Lett., vol. 6, pp. 3200–3207, Apr. 2021, doi: 10.1109/LRA.2021.3062812.

[16] I. Navarro and F. Matía, "An Introduction to Swarm Robotics," *ISRN Robot.*, vol. 2013, p. e608164, Sep. 2012, doi: 10.5402/2013/608164.

[17] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm Robotic Behaviors and Current Applications," *Front. Robot. AI*, vol. 7, p. 36, 2020, doi: 10.3389/frobt.2020.00036.

[18] "Dusty Robotics FieldPrinter mobile robot automates building layout," *The Robot Report*, Dec. 16, 2019. https://www.therobotreport.com/dusty-robotics-fieldprinter-automates-building-layout/ (accessed Sep. 08, 2021).

[19] K. Petersen, R. Nagpal, and J. Werfel, "TERMES: An Autonomous Robotic System for Three-Dimensional Collective Construction," p. 9.

[20] USPTO.report, "Automatic roof shingle removal and installation system," *USPTO.report*. https://uspto.report/patent/grant/9,945,128 (accessed Sep. 28, 2021).

[21] B. Jenett and K. Cheung, "BILL-E: Robotic Platform for Locomotion and Manipulation of Lightweight Space Structures," presented at the 25th AIAA/AHS Adaptive Structures Conference, Grapevine, Texas, Jan. 2017. doi: 10.2514/6.2017-1876.

[22] M. Allwright, N. Bhalla, H. El-faham, A. Antoun, C. Pinciroli, and M. Dorigo, "SRoCS: Leveraging Stigmergy on a Multi-robot Construction Platform for Unknown Environments," in *Swarm Intelligence*, vol. 8667, M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, and T. Stützle, Eds. Cham: Springer International Publishing, 2014, pp. 158–169. doi: 10.1007/978-3-319-09952-1_14.

[23] P. A. Clark, P. G. Lewin, and P. C. Pinciroli, "MARIA: Multi-Agent Robotic Intelligent Assembly." [Online]. Available: https://digitalwpi.wpi.edu/pdfviewer/m039k804v

[24] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *2008 47th IEEE Conference on Decision and Control*, Dec. 2008, pp. 1212–1217. doi: 10.1109/CDC.2008.4739098.

[25] A. Jevtic, Á. Gutierrez, D. Andina, and M. Jamshidi, "Distributed Bees Algorithm for Task Allocation in Swarm of Robots," *IEEE Syst. J.*, vol. 6, no. 2, pp. 296–304, Jun. 2012, doi: 10.1109/JSYST.2011.2167820.

[26] Q. Wang and X. Mao, "Dynamic Task Allocation Method of Swarm Robots Based on Optimal Mass Transport Theory," *Symmetry*, vol. 12, no. 10, p. 1682, Oct. 2020, doi: 10.3390/sym12101682.

[27] H. Wang and M. Rubenstein, "Shape Formation in Homogeneous Swarms Using Local Task Swapping," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 597–612, Jun. 2020, doi: 10.1109/TRO.2020.2967656.

[28] H. Wang and M. Rubenstein, "Decentralized Localization in Homogeneous Swarms Considering Real-World Non-Idealities," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6765–6772, Oct. 2021, doi: 10.1109/LRA.2021.3095032.

[29] U. Kumar, A. Banerjee, and R. Kala, "Collision avoiding decentralized sorting of robotic swarm," *Appl. Intell.*, vol. 50, no. 4, pp. 1316–1326, Apr. 2020, doi: 10.1007/s10489-019-01602-5.

[30] L. Wang, A. D. Ames, and M. Egerstedt, "Safety Barrier Certificates for Collisions-Free Multirobot Systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 661–674, Jun. 2017, doi: 10.1109/TRO.2017.2659727.

[31] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, On-line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 1047–1054, Apr. 2017, doi: 10.1109/LRA.2017.2656241.

[32] “Figure 1.5: Voronoi cells on a 2D surface. Illustrated is the capture...,” *ResearchGate*. https://www.researchgate.net/figure/Voronoi-cells-on-a-2D-surface-Illustrated-is-the-capture-zone-areas-for-randomly_fig3_267400965 (accessed Oct. 28, 2021).
[33] “Teensy® 3.6.” https://www.pjrc.com/store/teensy36.html (accessed Apr. 25, 2022).
[34] “AMS AS5048 Magnetic Rotary Encoder.” Jan. 29, 2018. [Online]. Available: https://ams.com/documents/20143/36005/AS5048_DS000298_4-00.pdf
[35] “CCBEC 10A PEAK 25V MAX INPUT SBEC.” https://www.castlecreations.com/en/cc-bec-010-0004-00 (accessed Apr. 28, 2022).
[36] “MP1584.pdf.” Accessed: Apr. 25, 2022. [Online]. Available: https://www.makerfabs.com/desfile/files/MP1584.pdf
[37] “rosserial - ROS Wiki.” Accessed: Apr. 26, 2022. [Online]. Available: http://wiki.ros.org/rosserial
[38] “Gazebo.” http://gazebosim.org/ (accessed Mar. 25, 2022).
[39] “Source Code & API | MoveIt.” https://moveit.ros.org/documentation/source-code-api/ (accessed Mar. 25, 2022).
[40] S. Chitta *et al.*, “ros_control: A generic and simple control framework for ROS,” *J. Open Source Softw.*, vol. 2, no. 20, p. 456, Dec. 2017, doi: 10.21105/joss.00456.
[41] J. Bohren, C. Paxton, R. Howarth, G. D. Hager, and L. L. Whitcomb, “Semi-autonomous telerobotic assembly over high-latency networks,” in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2016, pp. 149–156. doi: 10.1109/HRI.2016.7451746.
[42] Morgan Quigley *et al.*, *ROS*. [Online]. Available: https://github.com/ros/ros
[43] Amit Patel, “Red Blob Games: Hexagonal Grids.” https://www.redblobgames.com/grids/hexagons/ (accessed Mar. 25, 2022).
[44] “actionlib - ROS Wiki.” Accessed: Apr. 26, 2022. [Online]. Available: http://wiki.ros.org/actionlib
[45] T. Kugelstadt, “The RS-485 Design Guide.” Texas Instruments, May 2021. Accessed: Apr. 28, 2022. [Online]. Available: https://www.ti.com/lit/an/slla272d/slla272d.pdf?ts=1651060327565
[46] S. Corrigan, “Introduction to the Controller Area Network (CAN).” Texas Instruments, May 2016. Accessed: Feb. 28, 2022. [Online]. Available: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf