DESIGN OF A ROBOTIC ANT TO MODEL COLLECTIVE EXCAVATION

A Thesis
Presented to
The Academic Faculty

by

Vadim Linevich

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology
May 2015

DESIGN OF A ROBOTIC ANT TO MODEL COLLECTIVE EXCAVATION

Approved by:

Professor Harvey Lipkin, Committee Chair
School of Mechanical Engineering
Georgia Institute of Technology

Professor Daniel I. Goldman, Advisor
School of Physics
Georgia Institute of Technology

Professor David L. Hu
Department of Mechanical Engineering
Georgia Institute of Technology

Date Approved: 22 April 2015

ACKNOWLEDGEMENTS

I would like to thank Professor Daniel Goldman for his support, advice, and mentorship. I would like to also thank Professor Harvey Lipkin for providing guidance with my thesis, as well as Professor David Hu for his time for the thesis reading committee. A special thanks goes to Dr. Daria Monaenkova for help and guidance with the project.

I would like to express appreciation to Mark Kingsbury, Will Savoie, Jeff Aguilar, Tingnan Zhang, Feifei Qian, and everyone else in the CRAB lab for help along the way. I give thanks to Zack Braun, Olivia Lofaro, and Charles Xiao for help with putting the robots together. I would also like to thank Apoorva Nori and Alice Lin for helping with test bed setup.

DARPA and NSF Physics of Living Systems (PoLS) supported this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Fire ants (*Solenopsis invicta*) are social insects who work together to excavate and build large underground nests. In addition to the challenges of working in narrow spaces and dark environments, the ants must also avoid creating traffic jams which could negatively affect the colony's construction efforts. Traffic jams could be created if too many ants are going to the same destination which could happen if the workload is not shared wisely. We hypothesize that a part of the ant's traffic jam avoiding strategy is to be inactive, or lazy.

A model is needed to test this hypothesis and to study which excavation work sharing methods are effective. Robots are often used as physical models to study animals in general and their behavior. Unlike animals, robots can be precisely controlled as well as equipped with sensors to systematically measure parameters of interest. This thesis documents how robotic ants were designed and built, as well as outlines how the robots will be used to study ant behavior.

Although we did not study laziness here, preliminary data showed that tunnel excavation was proportionally faster in a test bed containing multiple active digging robots. An increase in the number of robots, however, caused an increase in the amount of energy required to propagate the tunnel forward. The increase in the energy came from the fact that multiple robots were colliding and jamming.

In this thesis, Chapter I contains relevant background information and motivation. Chapter II discusses the design approaches for the robot and test bed. Chapter III covers electrical engineering design topics and Chapter IV presents mechanical design. Software elements are discussed in Chapter V. Lastly, Chapter VI contains interesting remarks and observations, as well as data, conclusions, and future work.

CHAPTER I

INTRODUCTION

1.1 Robots and Science

Robots are increasingly popular tools used to study animals and their behaviors [1]. An early example of such robot is shown in Figure 1. Figure 1 shows a robotic tortoise constructed by William Grey Walter in the 1950s [2]. Unlike animals, robots can be commanded to act or move in a desired fashion. In addition, robots can be equipped with various sensors so that parameters of interest can be measured. This allows scientists to experimentally and systematically test biological hypotheses. For example, Marvi et al. [3] used a robotic snake to study how sidewinding rattlesnakes traverse dry granular media. In another example, Mazouchova et al. [4] used a robot to discover principles of terrestrial locomotion used by animals with flippers, like sea turtles.

Research has been also done with multi-robot systems. This is often referred to as "swarm robotics". A good example of a swarm robotics project is the Kilobot Project [5]. The Kilobot Project was partially inspired by ants, and can be used to program and experiment with collective behaviors. Each robot has a programmable microcontroller, motion controller, and local communication. The robots are low cost and are simple. Nevertheless the robots can be programmed to collaborate and do sophisticated tasks which can not be accomplished by an individual robot. These robots have been used to study and develop algorithms for tasks such as self-assembly and collective transportation [5].

Swarm robotics has been also used to study animal behavior. For example, Werfel et al. [6] built robots which performed collective construction. These robots were

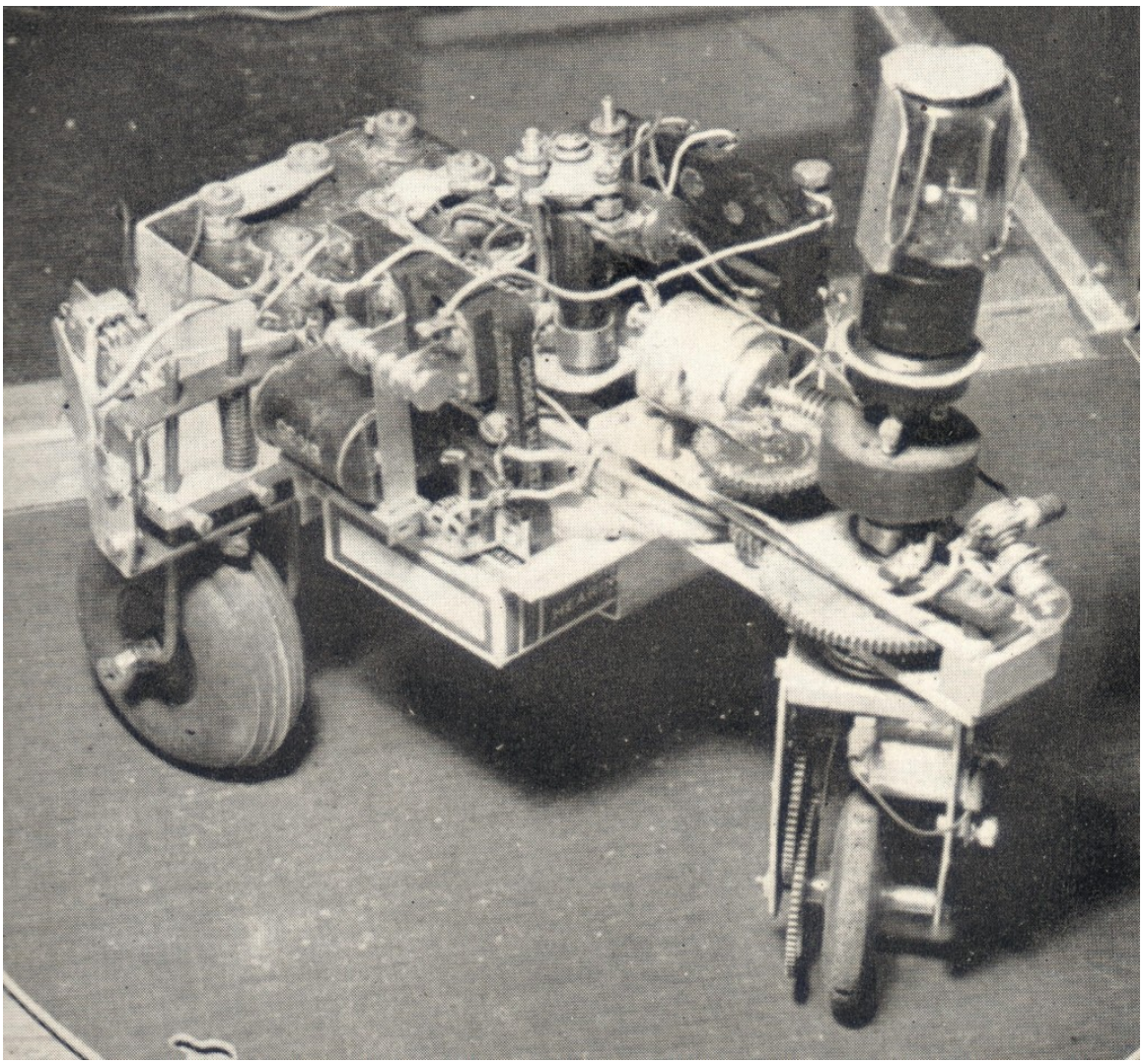


Figure 1: A robotic turtle built by William Grey Walter

inspired by termites. While working together, these robots were able to construct structures without an online centralized controller.

1.2 Digging Animals

There are many animals in nature that dig and excavate to build burrows for shelter. Rabbits, rodents, badgers, and moles are several good examples. Perhaps the most impressive nests are made by social insects like fire ants. Fire ants have bodies which have widths and lengths on the order of millimeters. Individual ants have limited cognitive abilities [7]. Yet, fire ants are known to dig nests with footprints that are larger than 10 meters in diameter [8]. The nests are large and complex because ants need room for shelter, food storage, brood care, and other social activities.

Ants benefit from working collectively. Sudd [9] conducted an experiment where he confined one ant in an artificial tunnel. The digging behavior of the single ant was observed, and the tunnel excavation and propagation was measured. A second ant was added to the tunnel. One would expect that the excavation rate to double. Sudd discovered that two ants spent less time digging than a single ant, and the tunnel propagation was comparable. Shared workload was an observed benefit. Both ants were not digging as aggressively as they did on their own. Social behaviors, such as mutual grooming were also observed.

Ants face difficult challenges while working collectively. The tunnels and spaces in which they work are small and confined. Also, navigation is a challenge since ant tunnels are shrouded in darkness. Ants must be able to find their way to their destination while also sharing the same narrow passages with other ants. Having too many active ants in the nest could create jamming and bottle neck situations which would stymie ants from getting where they need to go.

Ants must have strategies to mitigate challenges associated with collective work. We hypothesize that ants are being lazy because it is part of the strategy to optimize

the growth of the tunnel and to prevent traffic jams. Charbonneau et. al. [10] studied how ants spend their time in the colony by observing their behavior in the lab and in the artificially build field nest. The experiments showed that the ants were inactive and appear to be doing nothing for almost 80% of their time while in the nest. Inactive ants were not engaging in the activities such as building, excavation, or caring for brood. Garvish et al. [11] studied how ants of different sizes contribute to excavation. A large percentage of the population of ants was found to be inactive and not participating in digging. The same behavior was noticed while doing similar experiments in our lab.

1.3 Objective

To test a lazy ant hypothesis, a physical model is needed. Building a robotic ant will enable research to systematically study which strategies are effective in social digging. Once developed, many copies of a robot ant can be made. In addition, having a physical model could yield other interesting insights which would not be evident if a computer simulation model was used. This work will document how a robotic ant was designed and constructed, as well as some of the interesting insights and observations made in the process of building and testing.

It is of interest to study how ants work together to excavate and to construct such complex nests. Strategies used by ants certainly can be used or applied in some commercial applications. An example of this would be a swarm of robots mining minerals, or deployment of swarm robots to other planets or asteroids for resource mining or exploration purposes.

1.4 Relevant Robotic Work

This is not the first instance of using robots as a tool to study behavior of ants. SwarmLab [12] [13] built robots to better understand how robots navigate complex networks by relying on a a pheromone trail. Robots possessed photoreceptors which

detected beams of light. These beams of light were projeted on the maze to simulate pheromone trails.

MIT Artificial Intelligence Lab conducted a project titled "The Ants: A community of Microrobots" [14]. Several ant inspired robots were built. Each robot had a variety of sensors including light, bump, tilt, and food sensors. The food sensor detected if a robot's gripper stumbled upon a conductive sphere, which served as a model for food. The locomotion was achieved with a set of individually driven treads. These robots were used to develop algorithms which would enable the robots to collectively complete tasks. In that research, the robots could play various games like follow the leader, tag, and manhunt. Our application required development of a new robotic platform for several reasons. Kilobot robots require a centralized controller; however, in our application we wanted decisions to be made by individual robots. SwarmLab robots lack sensors and were developed to be used in a maze environment. Our focus was to study collective excavation where a complicated pheromone trail following was not needed. MIT's ants were the closest match, but their battery life was limited, did not have autonomous charging, and had long bump sensors which would conflict with tunnel digging. In addition, our robot was designed and constructed with readily available, off-the-shelf parts and utilized open source hardware when possible.

CHAPTER II

SYSTEM ENGINEERING

In this chapter, we will review the general approach on developing a robotic ant model, design constraints and requirements, as well as test bed design.

2.1 Design Requirements

The design of a robot satisfies the list of design constraints shown in Table 1. Each design constraint will be further discussed in this section.

Table 1: Design Constraints

Design Constraint	Brief Description
Autonomous Operation	The robot must have a sufficient amount of sensors so that it can be programmed to perform social excavation activities autonomously. The robot must make all actions and decisions based on what the robot perceives in the environment around it.
Robustness	The robot must not fail unexpectedly mechanically, electrically, or otherwise during operation.
Durability	The robot must be able to operate over an extended period of time with minimal maintenance.
Programmable Behavior	The user must be able to provide the robots with the rules which will shape how the robot reacts to the environment.
Experiment Oriented	The robot must transmit or log data of interest so that various behaviors can be assessed.
Scalability	One must be able to build additional robots with minimal costs.
Safety	The robot must have features which will ensure that the robot will not cause any harm to human operators under any circumstances.

2.1.1 Safety

Making the robot safe to operate was the number one priority. The mechanical design of the robot avoided having any sharp shapes or objects. The robot was

designed to act autonomously, which means that the robot planned and executed its actions independently. The robot could start and stop operating automatically which could produce unexpected behaviors or unanticipated movements. For this reason, it was not a good idea to have anything on the robot that could potentially cut or physically injure a user in any way, shape, or form. The circuitry was also designed with safety in mind. Each component was not expected to be loaded beyond its rating. Components were also individually tested and were not found to generate excessive heat. The circuits were hidden away from view inside of the robot. In the event of an unforeseen catastrophic failure, such as a circuit catching on fire due to a defect in a component, the failure should be contained inside of the robot.

The robot featured a mechanical emergency stop switch. The purpose of the switch was to disable the robot by cutting the power supply. The emergency switch could be used if an accident occured and immediate robot shutdown was needed. Also, the emergency switch could be used to safely put the robot in storage. The robot's battery could still be manually charged with the emergency switch engaged.

2.1.2 Autonomous Operation

As mentioned before, the core requirement for this robot was to be completely autonomous. The robot had to be able to perform various tasks such as (but not limited to): moving around the test bed, finding the excavation site, performing digging operations, retrieving and depositing excavated media into designated areas, seeking charging station and recharge its own batteries. The robot would operate independently without commands or guidance from an external source such as a centralized controller.

2.1.3 Robustness

The robot was designed to be strong enough to handle impacts and collisions. During the debugging process, the robot could perform unpredictable actions such as accelerating and ramming itself against the walls. Since there was no centralized controller, it was also expected that the robots would collide with each other. For these reasons, the robot had to be able to survive mechanical abuse and not be at risk of getting knocked out of operation in the event of rough interactions. Likewise, all electronics, cables, and wiring had to be well secured to enable robust operation.

2.1.4 Durability

Each robot was designed so that it could be used and reused for a long time (at least a year worth of use). Parts were designed to be adequately strong and robust. Electric components were expected to have a reasonably long life. Circuitry was designed with safety in mind and was not found to be a subject to overheating or have vulnerability to voltage or current spikes. Fragile parts, such as electronics, were adequately protected or shielded from possible mechanical or physical damage.

2.1.5 Scalability

The goal of this project was to construct multiple robots and have them collectively dig as a swarm. Thus, it was desired to minimize the cost associated with building each robot. It was also desired to make the processes of parts acquisition, robot assembly and manufacturing as simple and straightforward as possible. It was also desired that the robot construction from start to finish would take a reasonable amount of time.

2.2 Testbed and Experiment Design

2.2.1 Test Bed Overview

The mission was to build a simulated ant tunnel test bed and let robots dig in it for an extended period of time. Although ants dig tunnels in three dimensional space, the test bed constrained robots to dig in a quasi 2D space. The test bed was constructed on top of the flat rectangular table. A picture of a test bed is shown in Figure 2 and a schematic diagram is shown in Figure 3. Wooden planks were added along the perimeter of the table to serve as walls. The walls were needed to constrain excavation and to prevent the robots from falling of the table. The table top was covered with black construction paper so that the top surface was not as slippery. In doing so, the robot could move around the test bed. Pink tape was placed in the middle along the tunnel. This visual clue served as a simulated pheromone trail and was used to help robot navigate. The test bed was filled with a granular media (cotton) which modeled cohesive granular material like slightly wet soil, and in which the robots would dig tunnels. The test bed also had a small bay where robots could autonomously dock and recharge their batteries. The charging area was distinguished with bright colors so that the robot could find it autonomously.

The charging station design is shown in a Figure 4. Two 22 gage wires were stretched across the charging bay. Each wire was soldered to a spring on each end, and the springs were bolted with metal conductive bolts to plastic parts. The springs helped wires stretch and to be compliant when the robot ran into them. Once the robot was done charging, the springs returned the wires to an equilibrium position. The top wire carried +5V, while the bottom wire served as a ground wire. The springs were bolted to a 3D printed plastic part which was an insulator. The 3D printed plastic part was bolted to aluminum t-slot extrusions which were used to construct the charging station frame.

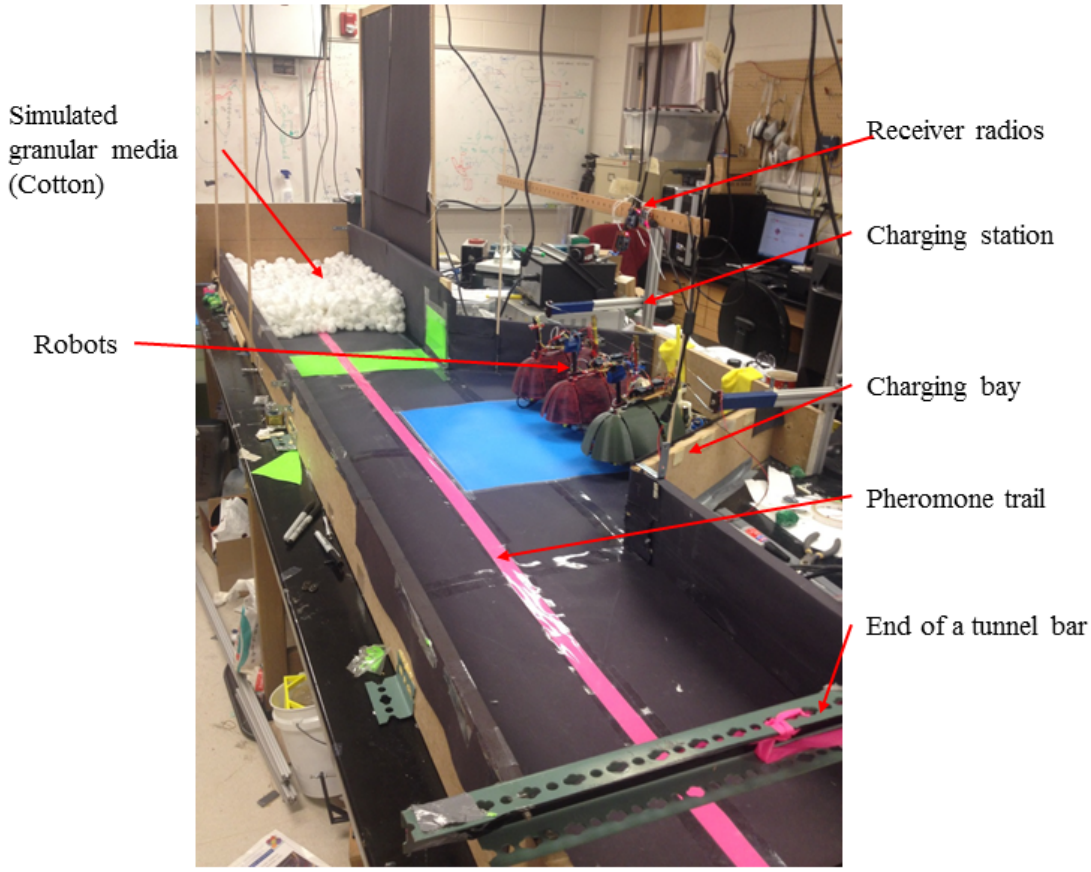


Figure 2: A picture of a tabletop experimental setup with robots

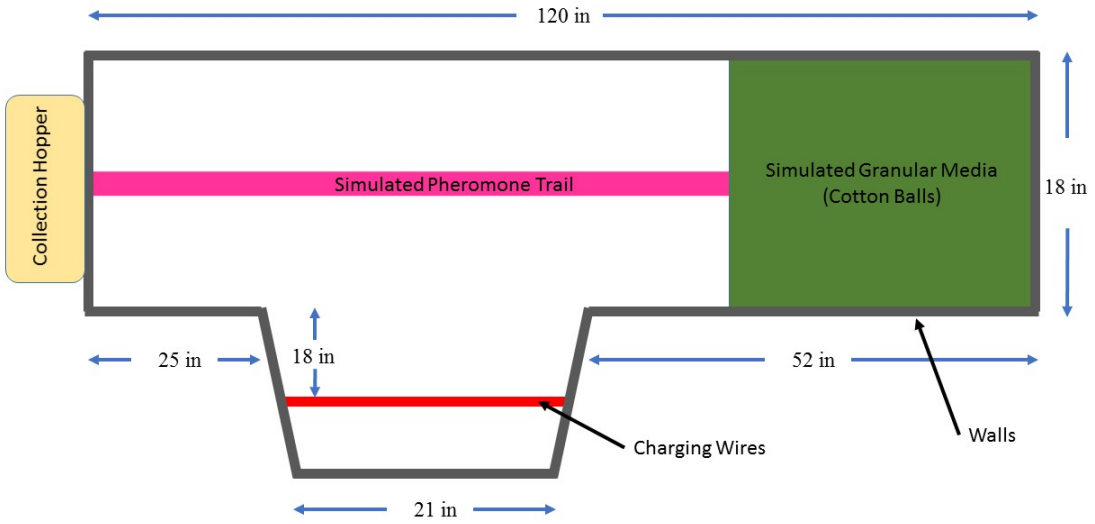


Figure 3: A diagram of a tabetop setup with dimensions

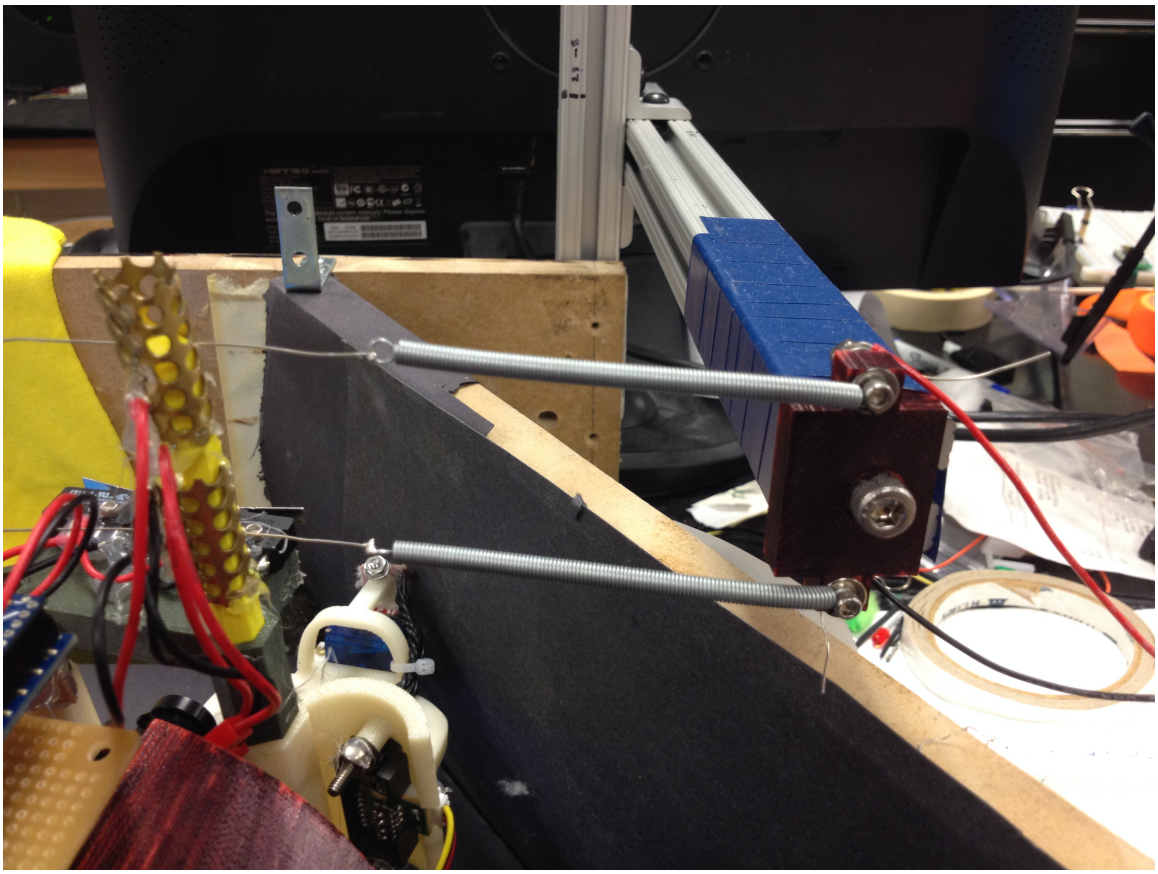


Figure 4: A picture of a robot charging at the charging station

2.2.2 Simulated Granular Media Choice

It was desired to have a material with cohesive properties so that there would be energy cost associated with excavation activity. The energy cost comes from the fact that cohesive particles are not easy to pull apart. Jumbo cotton balls was an inspired choice to serve as simulated granular media, which has cohesive properties. Cotton balls can be pulled out of pile one by one or in small bunches to form 2D tunnels. Cotton balls are also squishy which makes them an easy target for pick up with claw-style robotic manipulators.

CHAPTER III

ELECTRICAL DESIGN

In this chapter, we will discuss circuitry design, and other topics pertaining to electrical engineering.

3.1 Microcontroller Selection

Perhaps one of the most important decisions that was made at the early project stage was selection of a microcontroller (MCU) board: a heart and a brain of each robot. Four MCUs were considered: Arduino Mega [15], Arduino Due [16], BeagleBone [17], and Raspberry Pi [18]. Intel Edison [19] is another great alternative choice, but it became available long after the inception of a project and is not included in this discussion. These MCUs were chosen because they are relatively inexpensive, readily available, and there is plenty of documentation available since all of these candidates are open source hardware.

Table 2: Microcontroller Comparison

	Arduino Mega	**Arduino Due**	**Raspberry PI Model A**	**BeagleBone**
RAM	8 KB	96 KB	512 MB	512 MB
Flash	128 KB	512 KB	256 MB	4 GB
CPU	ATmega1280	AT91SAM3X8E	ARM1176JZ-F	ARM AM335x Cortex A8
Clock Speed	16 MHz	84 MHz	700 MB	1 GHz
OS	None	None	Linux	Linux
GPIO	54 (15 PWM)	54 (12 PWM)	40	69
ADC pins	16	12	None	7
Comm	4 UART, SPI, I2C	4 UART, SPI, I2C	USB, CSI camera port, UART, I2C, SPI	USB, Ethernet, SPI, 4 UART
Cost	$39	$52	$25	$80
Size	10.2 cm x 5.4 cm x 0.8 cm	10.2 cm x 5.4 cm x 0.8 cm	8.6 cm x 5.4 cm x 1.5 cm	8.6 cm x 5.4 x 4.8 cm

From Table 2, we can see that BeagleBone and Raspberry PI are superior to Arduinos in terms of computational power and memory. These boards run Linux OS and would be an excellent choice for development of autonomous control code and software. Preference was given to Arduino family microcontrollers because robot prototyping could be accomplished faster. Arduinos have plenty of GPIO, ADC,

and communication ports (UART, I2C, SPI) making it easy to connect and evaluate performance of any sensor. Arduinos can be programmed in C or C++, and it can be argued that writing Arduino code is easier and faster than programming a board like Raspberry Pi or BeagleBone. Also, Arduinos have been around for a longer time than other boards discussed above. Beacause of that, the hardware is relatively bug free. There are also many great resources, libraries, and guides available online, as well as a huge forum community which can be taken advantage of for solving or troubleshooting any microcontroller related issues.

Of all available Arduinos, Due was chosen because it has the fastest clock and the largest program memory. Arduino Mega was a close runner-up. Arduino Due and Mega have a similar footprint and number of pins. Even though Due has more memory, not all existing libraries are compatible because Due has an ARM family chip while Mega has an AVR chip. Incompatibilities primarily arise from the fact that Dues clock runs faster. Libraries which rely on timers or SPI communication would require some meticulous modifications. Unlike Mega, Dues pins have 3.3V pin logic, even though the board requires 5V to power. For this reason, integration of Due with external circuitry and sensors is not as straightforward as with Mega. (Mega requires at least 5V to power and features 5V logic). Nevertheless, Arduino Due was ultimately chosen to be the brains of a robot.

3.2 Sensors

Our robot carries a variety of sensors that allow it to sense and interact with the environment. We will examine and discuss each sensor here.

3.2.1 Pheromone Trail Sensing

It is well known that ants leave pheromone trails to leave a path for other ants to follow [20]. To simulate this, two approaches were tested. The first (and earliest) approach involved leaving a dark line on a white floor surface and then using an

array of infrared proximity sensors, also known as line sensors. Each line sensors consists of an infrared (IR) light emitting diode (LED), and an IR detector. Each sensor emits IR light and measures the reflected portion. These sensors can be used to distinguish if the sensor is aimed at a light or dark surface. Having a linear array of these sensors allows for tracking a dark line on a white surface (or white line on dark surface). Line sensors, however, must be positioned close to the ground which is bad if the robot needs to drive over obstacles, such as loose or stray granular media. This approach has a limited sensor resolution and requires the robot to maintain a constant distance between the line sensors and the floor.

The second approach involved the use of a camera. A pheromone trail then could be either drawn with a writing utensil in advance, or by a moving robot. These lanes could be then erased and drawn again creating a changing and dynamic environment for robots to explore and navigate. Likewise, trails could be constructed using LEDs, some of which could be turned on and off. Having a camera could also enable the robot to identify objects of interest, such as digging area or charging station from a distance. Camera sensors were challenging to work with because they produced a lot of data which had to be processed in real time, as well as required knowledge and implementation of machine vision techniques.

Line sensors were evaluated and were found to serve as an adequate and simple solution to lane following. The line sensors however had tendencies to hinder the robot mechanically because they had to be positioned close to the ground which would create situations where granular media would get stuck between the ground and the sensors. Granular media could also skew the sensor readings if it was to get underneath a line sensor causing robot to lose track.

A second option to detect a pheromone trail was explored and a Pixy (CMU Cam 5) camera was found to provide a great solution. Pixy hardware performs image processing onboard on its own processor, and then sends processed data to an external

device, in our case our Arduino, via UART, SPI, or I2C connection. It does all of the image capturing and processing in real time and returns only necessary information, such as pixel coordinates of a detected pheromone trail, which is then sent to a PID controller to accomplish line following.

3.2.2 Navigation

The Pixy camera was also used to tell the robot the locations of important landmarks in the tunnel. For example, the charging station is marked with a particular color that Pixy can detect. Once the charging station marker is detected, the robot knows where to drive if charging is needed.

In addition to the camera, the Inertial Measurement Unit LSM9DS0 (IMU) was used. IMU had a triple axis accelerometer, triple axis gyroscope, and triple axis magnetometer. The magnetometer served as a compass and helped the robot to orient itself to face the simulated granular media or any other direction. The magnetometer was calibrated in advance. The gyroscope was used for feedback for turning. The robot knew if it was making turning progress by checking if there is angular acceleration.

3.2.3 Obstacle Avoidance

Fire ants are capable of detecting obstacles in front of them by using their antennas. A similar capability was sought in the robot. HC-SR04 ultrasonic sound sensor and Sharp GP2Y0A21YK infrared (IR) sensors were evaluated. An ultrasonic sensor was lower cost and required two digital pins to measure distance. The ultrasonic sensor also required a 5V power supply and logic to operate which means that a logic level shifter would be necessary to interface with a 3.3V Arduino Due logic. The infrared sensor was smaller in size, required one analog pin, and yield a faster and more accurate measurement. The infrared sensor was simpler to use and thus was chosen.

A digital proximity QRE1113 sensor was also used for obstacle detection. This

sensor works by measuring how much of the emitted infrared light is reflected back and is commonly used for line following and. This sensor was mounted on the head of the robot and was used to detect if there is anything in front of it up to 3 mm away. This sensor was primarily used to prevent the robot from hitting objects head on.

3.2.4 Grip Feedback

An analog version of a QRE1113 sensor was used to detect whether the robot was holding something in it's gripper. The sensor was positioned in a way that its produced a low voltage output if there was something gripped. The same sensor was used to trigger an excavation routine. The sensor's output changed when the robot's head touched the cotton ball pile.

3.2.5 Power Consumption

Energy consumption was an important quantity to measure. This was accomplished by having the robot periodically measure its current consumption and battery voltage. The battery current draw was sensed with an Allegro's ACS714LLCTR05B-T breakout board. This sensor is bidirectional and can be used to measure current flowing out of the battery, and into the battery when the robot was charging. It is worthwhile to point out that this sensor runs on 5V, with means that the analog output pin which can output voltage from 0 to 5V. The analog pin has a nominal output of 2.5V. The output voltage rises if the current goes one way, and drops if the current flows the other way. The output pin can be safely connected to Arduino DUE's 3.3V analog pin since the sensor output is not expected to go beyond 3.3V due to the design of power supply circuitry.

A LiPo battery with a 3.7V nominal charge was used for reasons to be discussed later. The voltage on a battery can go as high as 4.2V while charging and cannot be safely measured with Arduino's 3.3V ADC pin. A simple voltage divider was used to safely monitor the battery's voltage. A voltage divider circuit is shown in a Figure 5

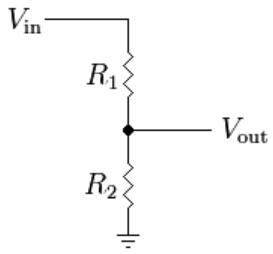


Figure 5: Voltage divider circuit schematic

where V_{in} is connected to a battery and V_{out} is connected to ADC pin. From Ohms law, the following relationship can be deduced:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} \tag{1}$$

R_1 was set to be equal to R_2 so that the voltage output of this circuit is half of the voltage input. Doing so will enable a 3.3V ADC pin to safely measure voltages as high as 6.6V which is greater than anticipated batterys 4.2V level. 3.3kΩ resistors were found to work well. If the voltage on the battery is 4.2V, from Ohms law we can compute that only about 636 μA of current was sacrificed to obtain the voltage measurement. It is obvious that increasing the values of resistors would decrease wasted current. The impedance of the pin, however, must be greater than the impedance of the resistor. Otherwise, the current will mostly go through the ADC pin and not through R_2 and the voltage divider will not work. The pins on DUE are guaranteed to have impedance between 50kΩ and 150kΩ. A 3.3kΩ resistor was found to serve well by experimentation.

3.2.6 Charging Station Detection

The same voltage divider circuit used for battery voltage measurement was also used to detect when a robot's charging electrodes were in contact with the charging station. The V_{in} was connected to robot's positive charging electrode, and V_{out} was connected to a digital pin. The digital pin on the Arduino would be driven to a high state if the robot was touching the charging station and low if it was not. This information was

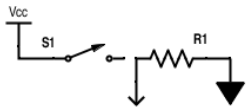


Figure 6: Contact switch circuit schematic

used to verify successful docking to the charging station. The Pixy camera was used to guide the robot to the charging station.

3.2.7 Bump and Physical Contact Sensing

Since collisions and physical contacts and interactions were expected, it was desired to have some form of sensing or detecting such events. Simple single pole single throw (SPST) switches were implemented. Six switches were mounted around the robots body: two on each side and two on the back. The robot had an outer shell which was split into six sections (see Mechanical Design chapter). Touching or pressing the shell segment triggered the switch to close, allowing contact detection. The spring inside of the switch would make the switch open if the contact was removed. The electrical schematic is show in Figure 6. The digital pin was used to read if the pin was on or off. The resistor R_1 was necessary to pull the pin to a low state when the switch was open. A resistor value of 3.3kΩ was found to work well.

It is worthwhile to mention that both the gyroscope and accelerometer could also be used to detect bumping and collisions. A spike in a reading could be observed at the moment when the bumping or collision occurs. It is hard to tell, however, if the robot is being continuously pressed or touched.

3.2.8 Deposit Area Detection

A Pixy camera and a beacon were used to signal the robot when and where to deposit an excavated payload. The robot could calculate how far it was from a deposit beacon since the beacon area was fixed and known. This approach was phased out in favor of a more robust version. Two SPST switches were mounted to the top of the robot and

used in the exact same way described in the section above. When the robot reached the end of a table, it hit an overhead horizontal bar which prevented the robot from driving off the table. When the switches closed, the robot started the deposit routine, dropping the cotton balls off the table and into a collection bin.

3.2.9 Sensor Summary

Table 3: List of Sensors

Sensor	**Function**
Pixy (CMU5) Camera	Pheromone Trail, Navigation
IMU	Navigation, Turning feedback
IR Distance Sensors (x2)	Obstacle Avoidance
Digital Proximity Sensor	Obstacle Avoidance
Analog Proximity Sensor	Grip Feedback
Current Sensor	Power Consumption Monitoring
Battery Voltage Sensor	Power Consumption Monitoring
SPST Switches	Physical Contact Sensing, Deposit Area Detection

3.3 Radios

A radio was mounted on the robot to monitor a robot's power consumption. The radio could also be used to manually control the robot although this feature was not fully integrated with the autonomous program. However, the radio was not used for peer to peer communication between robots.

It was desired to pair up a radio with another microcontroller so that the main microcontroller would not be burdened with handling communication. An XBee ZB radio was selected to enable communication between the data logging computer and a robot. XBee ZB was found to be easy to set up and serve as a robust solution.

Arduino Fio was chosen as a microcontroller since the Fio had a socket connector for XBee radios, and had an adequately small footprint. Arduino Fio also was linked to Arduino Due with UART serial. With two microcontrollers linked, Arduino Due could also send and receive data over radio. For example, Due reported operational mode changes, such as decision to terminate excavation activity and going charging. Addition of a second Arduino also expanded the robot's capabilities since the Fio had unused GPIO and ADC pins.

The robot also saved a copy of all wireless transmissions to a micro SD card. A micro SD card reader was connected to the Arduino Fio.

3.4 Motion Actuators

The robot features two servo motors for the arm control and two DC motors for locomotion.

3.4.1 Servo Motors

Servo motors chosen for arm actuation are easy to use and do not require special circuitry or a driver. The HS-55 servo was used for gripping, and the HS-A5076HB servo was used for moving the arm up and down. These servos were chosen because they fit well with the mechanical design, were relatively inexpensive, and featured an adequate torque and reasonable current draw. The HS-A5076HB servo can be freely substituted with another servo (with minor changes to the mechanical design). The same cannot be said about the HS-55 servo because it was used with an off-the-shelf gripper kit, which was designed to work for this particular servo motor.

Servo motors have 3 pin connectors: power, ground, and signal. Both servos can be commanded to move to a desired position with a pulse width modulated (PWM) signal. Servos have an onboard controller which measures the width of the signal and drives the motor to a corresponding angle. Figure 7 shows how PWM signal maps to command angle for the HS-55 servo.

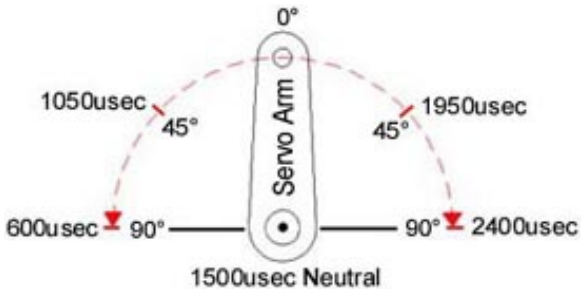


Figure 7: Gripping servo control diagram, courtesy of www.servocity.com

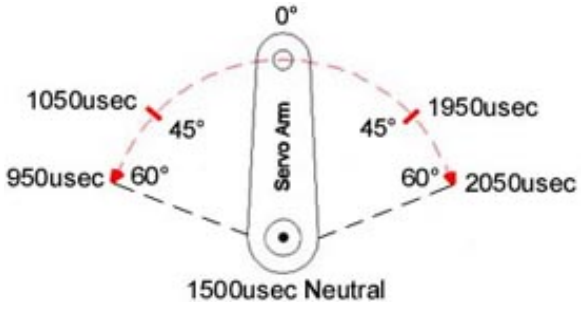


Figure 8: Gripping servo control diagram, courtesy of www.servocity.com

A PWM signal with rising edges spaced out 600μs apart will move the servo to a -90 degree position and a 2400μs pulses will move the servo to a +90 degree position. Interpolation can be used to figure out pulse width for a desired angle. Note that HS-55 servo does not rotate continuously and its movement is constraint to a 180 degree range. This servo has a 18 oz-in torque rating. Likewise, the HS-A5076HB servo is commanded in an analogous method and is constrained to 120 degree range as shown in a Figure 8. This servo has 42 oz-in of maximum torque rating.

3.4.2 Drive Motors

Simple geared brushed DC (direct current) motors were chosen to power wheeled locomotion. Brushless DC motors were not considered because they tend to be more expensive than their brushed counterparts and generally more complicated to control. Likewise, stepper motors were not considered due to complexity, cost, as well as typical high current draw.

There were several criteria for motor selections. The motors required sufficient

torque to drive the robot, draw as little current as possible, have a reasonable stall current, have an output shaft that could be coupled with a large variety of wheels, and low cost. Several different motors were tested, and ultimately a Pololu mini plastic gearmotor (Pololu part #1511) was chosen. These motors are compact and include a 120:1 reduction gearbox. The gearbox also features a built-in safety clutch which helps protect the gears from excessive loads. The output shaft is 3mm in diameter and has a D-shaped profile which easily interfaces with many wheels. At 4.5V the output shaft spins at 150 RPM and 25 oz-in torque according to the manufacturer specifications. Free running current is listed to be 130mA with 1.25A stall current.

A breakout board for TI's DRV8835 dual motor driver was used to control both drive motors. DRV8835 is a dual H-bridge that provides a bidirectional motor control. The driver can deliver 1.2A per channel continuously with peak currents up to 1.5A per channel for a few seconds if 5V are supplied to motor input and logic. This board was tested and found to run chosen motors well and safely with supply voltage varying from 3V to 5V. To control each motor, two digital pins were needed. One pin (enable) was used to set the motor direction while another pin (phase) was used for speed control with a PWM signal. In addition, a small 0.1μF ceramic capacitor was soldered to the positive and negative leads of each motor. This was done to remove some of the motor electrical noise.

3.5 Power Circuit

This subsystem was designed with the intent of being operated by a single rechargeable battery. The power circuitry handles voltage conversions and regulations to supply power to all electronics and actuators on board.

3.5.1 Battery Selection

Actuators and most of the sensors can operate on a voltage supply ranging from 3V to 5V. Only the Arduino Due and Pixy camera require 5V supply to function. For

this reason, it was decided that batteries with a nominal voltage in the 3V to 5V range would be given preference over batteries with a nominal voltage higher than 5V. It would be more efficient to step voltage up for just two devices rather than step voltage down for all sensors and actuators.

An obvious task before picking a battery was to estimate the current consumption. Most of the items had the current consumptions listed on a data sheet. For all other items, the current consumption had to be either calculated or measured. The results of the estimates are listed in a Table 4.

Table 4: Current Estimation

Item	**QNTY**	**Current in mA (per each)**	**Estimation Method**
DC Gear Motor	2	300	Measured
HS-A5076HB Servo	1	200	Data Sheet
HS-55 Servo	1	150	Data Sheet
IR Distance Sensor	2	30	Data Sheet
Analog Proximity Sensor	1	25	Data Sheet
Digital Proximity Sensor	1	25	Data Sheet
5V Step-Up Voltage regulator	2	2	Data Sheet
Current Sensor	1	13	Data Sheet
SPST Switch	7	1	Calculated
Voltage divider	1	0.6	Calculated
Dual Motor Driver Board	1	2	Data Sheet
Solid State Relay	1	12	Measured
Arduino Due	1	110	Measured
Arduino Fio + XBee	1	50	Measured
IMU	1	6.2	Data Sheet

The total sums to approximately 1.3A. We should also remember that the current draw can spike up in the event of an actuator motor stall. Drive gear motors are not expected to stall because they have a build-in clutch. Servo motors could potentially stall if the robot is involved in some form of a hard physical contact with the environment or other robots.

Rechargeable alkaline, NiMH, LiPo, and Lithium ion (Li-ion) batteries were considered to power the robot. NiMH and alkaline batteries were dismissed because they did not have a specific energy density as good as LiPo and Li-ion batteries. The desire to have the robot as small, light, and as compact as possible calls for a battery with a high energy density. Li-ion batteries were given preference over LiPo due to safety considerations. LiPo batteries have a higher energy density than Li-ion and can discharge larger currents but are more prone to failures (including explosion) if shorted, punctured, over charged, or mishandled in any way. A single cell 3.6V Li-ion battery with 3400mAh and a 2C discharge rating was chosen. In other words, chosen battery will be able to supply 3.4A of current for an hour, or supply an estimated 1.3A of current for about 156 minutes. The 2C discharge rating implies that the battery can continuously source up to 2A of current. The chosen battery also comes with a build-in safety circuit which will shut the battery if too much current is drawn, or if the battery's voltage drops too low (about 3V).

3.5.2 Circuit Design

The power circuit schematic is shown in Figure 9.

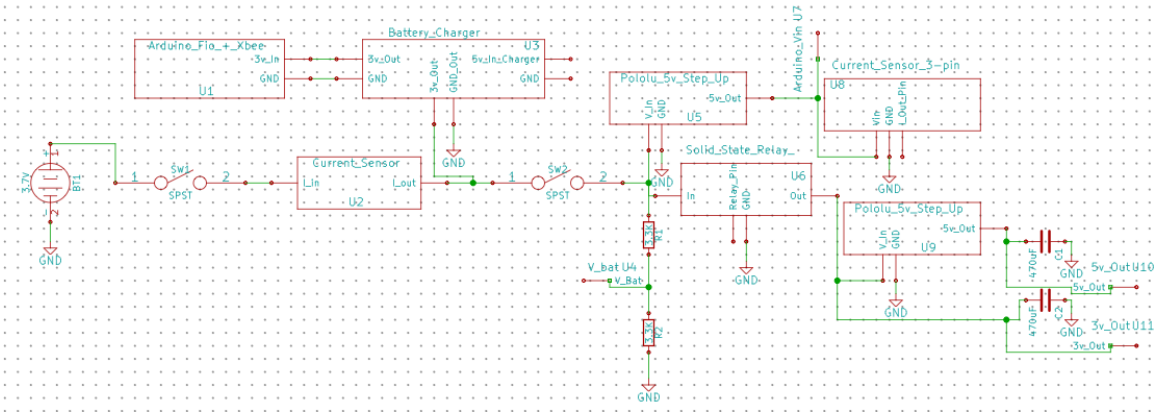


Figure 9: Power circuit schematic

The power from the battery was routed through an emergency shut off switch SW1 and then through a current sensor. Next, the power trace was connected to a

charging breakout board containing a MCP73831T chip. Arduino Fio and XBee radio were powered from the same node (the charging breakout board had a convenient JST connector). Next, the battery power was routed through another kill switch SW2. With SW1 closed and SW2 open, the robot could be manually charged with all current flowing to the battery. SW2 disconnects all loads from the battery. The node after SW2 was connected to a voltage step-up chip. The voltage step-up chip converts any voltage above 3V to a 5V output. This 5V output was used to power the current sensor, and the Arduino Due. The output of the SW2 was also connected to a solid-state relay. The robot could use this relay to put itself in a low power consumption state by cutting power to the actuators and sensors. The output from a power relay was used to power all actuators and sensors and was filtered with a 470μF capacitor. The output from a power relay was also passed to another voltage step-up chip to achieve a 5V output. This output was used to power 5V sensors, such as Pixy camera. The 5V output was also filtered with a 470μF capacitor. A picture of the assembled power circuit is shown in Figure 10.

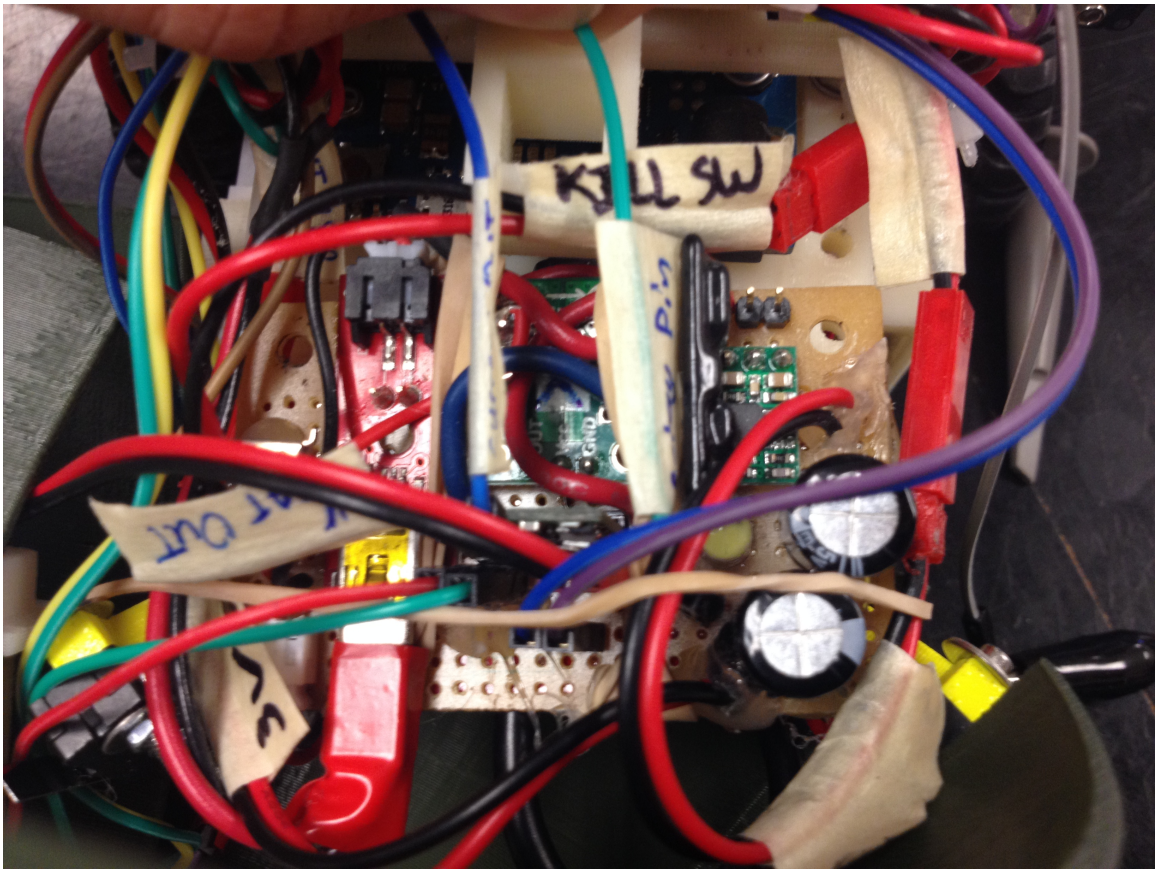


Figure 10: A picture of an assembled and installed power circuit

3.6 Other Circuits

3.6.1 Main Circuit Board

The main circuit board was connected to the 5V and the 3V power outputs from the power circuit and distributed the power to all sensors and actuators. Most sensors and actuators were standardized to a 3 pin connector (GND, Power, I/O). The main board had header pins to connect all sensors and actuators to the power and to Arduino pins. The main circuit board also had an I^2C bus and a socket for the motor driver. A picture of the assembled main board is shown in Figure 11.

Figure 11: A picture of an assembled and installed main board

3.6.2 Switch Board

This circuit board carried a circuit shown in Figure 6 for every contact switch. There were 7 contact switches in total: 6 switches were used for contact sensing, and 1 switch was used for dumping sensing. Two switches were used for dumping sensing, but they were wired up in parallel. The assembled switch board circuit is shown in Figure 12.

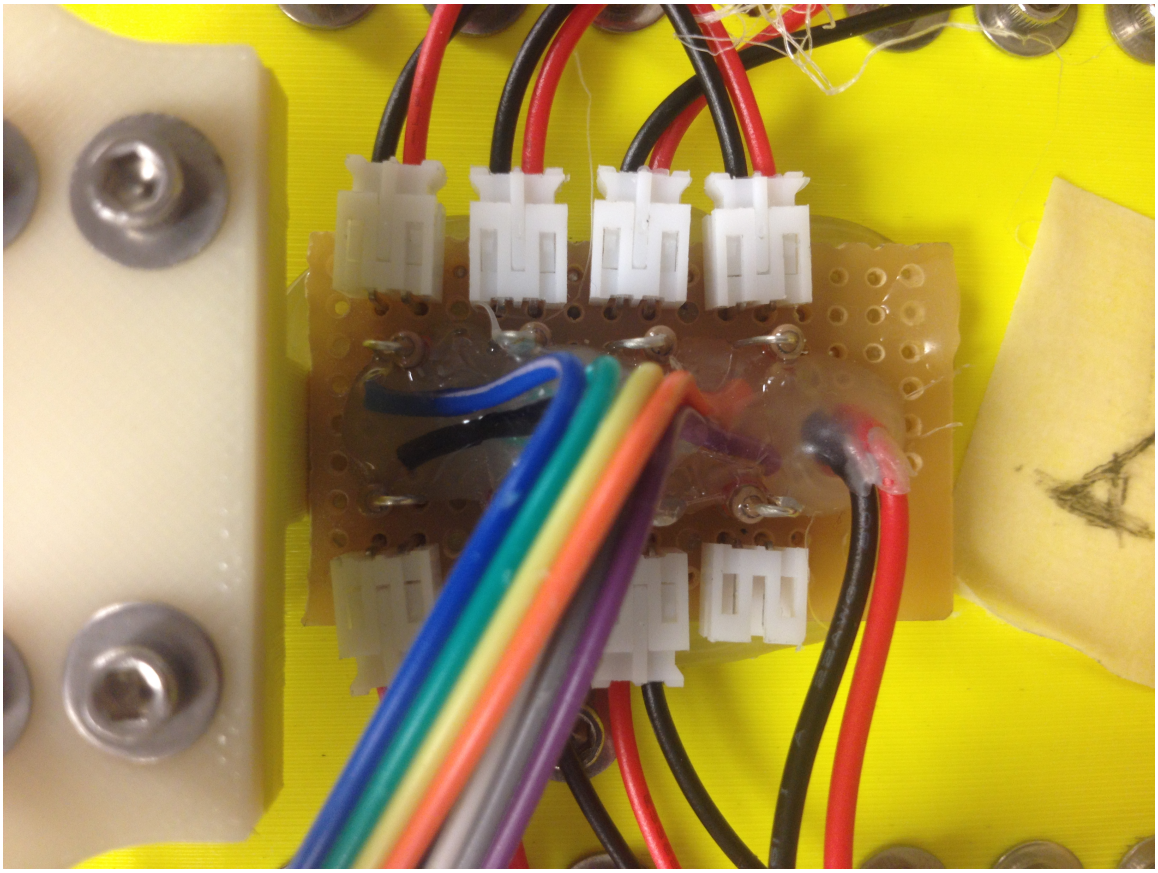


Figure 12: A picture of an assembled and installed switch board

3.6.3 Charging Detector

A picture of a circuit discussed in a Charging Station Detection subsection is shown in Figure 13. In the same figure, charging electrodes can also be seen. These electrodes were made out of a conductive brass mesh.

3.6.4 Transistor Reset

Two common 2N3904 NPN transistors were used to create an or gate circuit. The schematic of the circuit is shown in Figure 14. This circuit was used to reset the main microcontroller if the code was not running properly. The Arduino Due would be reset if either the Due or the Fio would drive a digital pin connected to a transistor gate high. A 3.3kΩ resistor was used to pull each transistor gate down to prevent

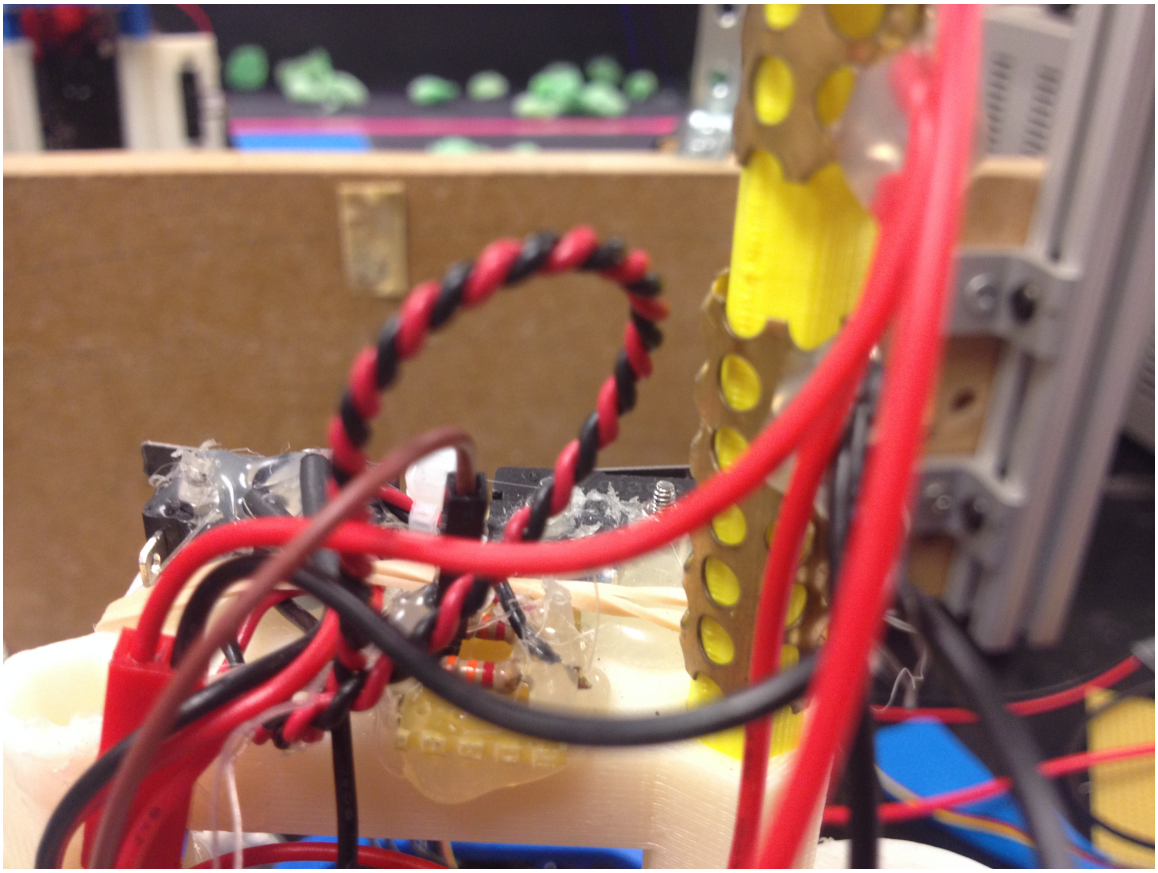


Figure 13: A picture of a charging detector board and electrodes

an accidental reset. This circuit was built as a work around to a problem discovered with Pixy camera. Sometimes on power-up, the camera would not start up properly due to an unknown bug in the library. The Arduino Due reset would fix the problem. The assembled circuit transistor reset circuit is shown in Figure 15.

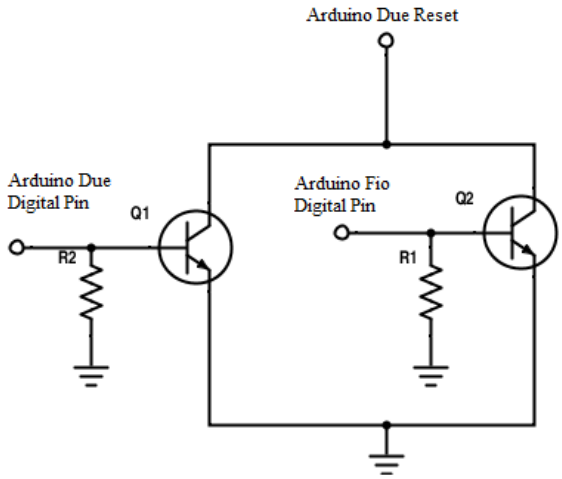


Figure 14: Reset circuit schematic

Figure 15: A picture of an assembled and installed reset circuit

3.6.5 Radio

The Arduino Fio and the XBees radio were mounted on top of the robot as shown in Figure 16. The Arduino Fio had analog pins connected to the voltage and current sensors.

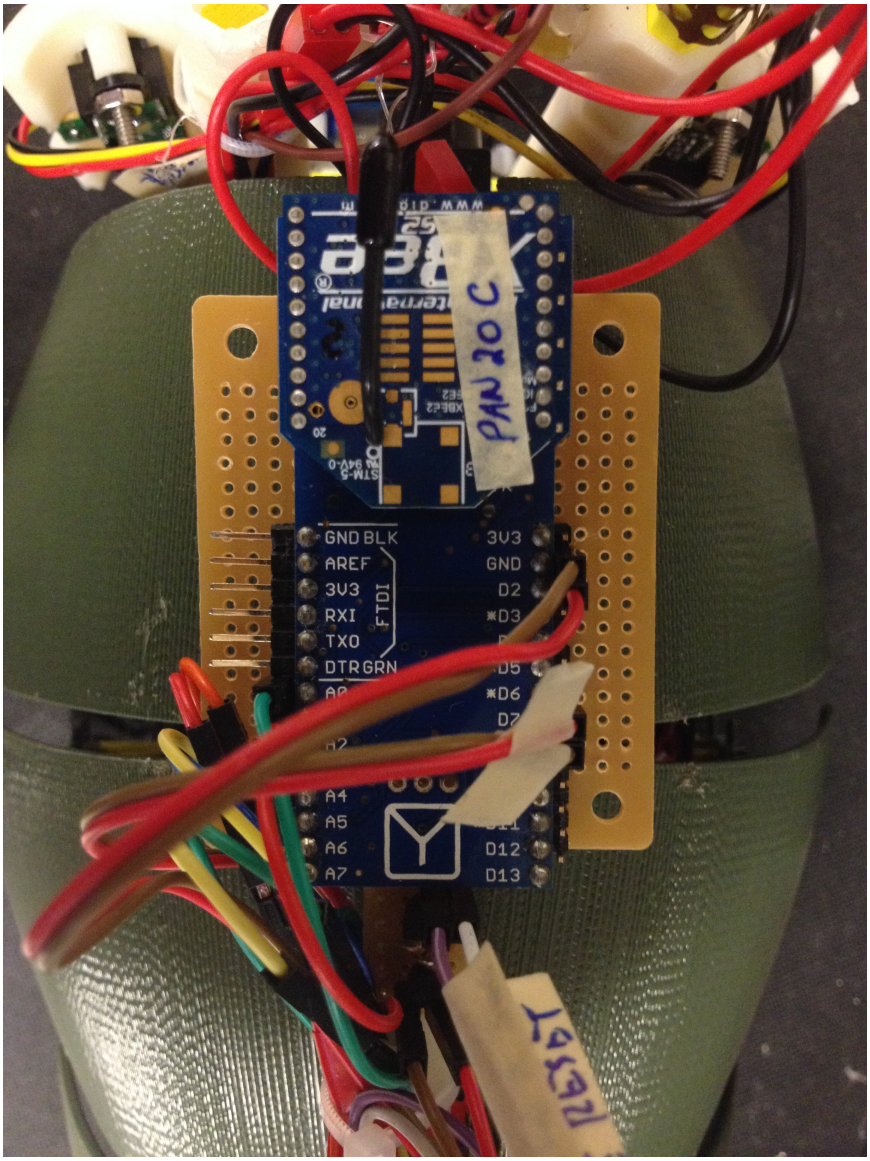


Figure 16: A picture of a Radio and XBee attached to the robot

CHAPTER IV

MECHANICAL DESIGN

In this chapter, we will go over how the mechanical design of a fire ant was developed.

4.1 Design Approach

The availability of 3D printers drove the mechanical design approach. A 3D printer is a machine which can precisely manufacture a three-dimensional solid object. In our case, the machine takes plastic filament, melts it, and then lays it down in successive layers. Melted plastic solidifies once it is in place. A part is first designed with a computer-aided design software. Next, the part is saved in a certain file format, and then imported into another software which creates a set of instructions which are uploaded to the printer to build the part. The printer can make the parts with an accuracy on the order of 300 micrometers (the actual accuracy depends on the printer and the manufacturer). The printer can also be used to make parts which would be difficult or time consuming to manufacture in a traditional machine shop environment. The plastic model material is light and adequately strong. It is advantageous to have light parts to minimize the amount of energy to run the robot (which will in turn reduce the cost and complexity of the robot). For these reasons, a 3D printer is an essential tool that was heavily utilized in the project. Also, if the project is to be released to the open source community, schematic files can be shared and anyone with access to a 3D printer would be able to make exact replicas of all parts.

It is worthwile to point out that the mechanical design presented in this work is a result of an iterative process. The body of the robot was designed to be as compact and storng as possible while having all sensors mounted in locations needed.

4.2 Prototype Overview

Figures 17, 18, and 19 show the CAD model of the robot from various angles. SolidWorks software was used to design the model. The robot was 13.5 inches long, 6 inches wide, and 9 inches tall. The height measurement was from the ground to the tip of the IMU.

Figure 17: CAD model of a robot

Figure 18: CAD model of a robot, rear view

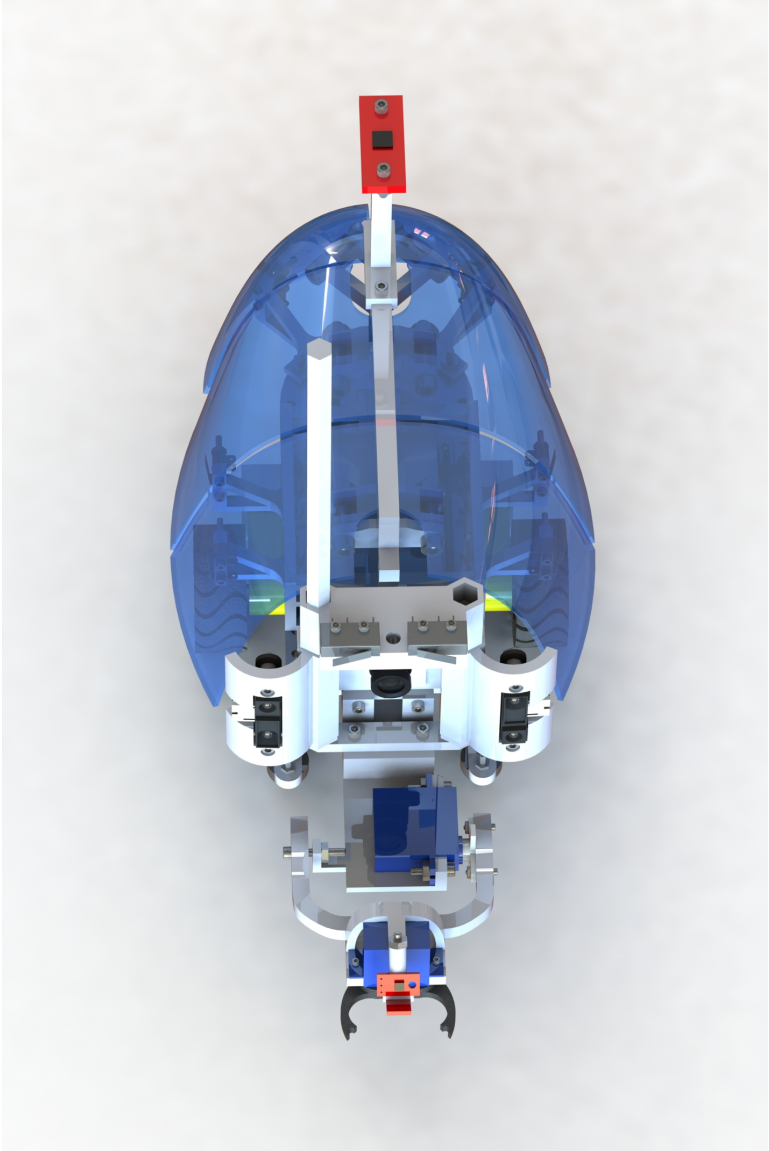


Figure 19: CAD model of a robot, overhead view

To assemble a robot, one starts with a Main Body part. Please refer to Appendix A for a list of 3D printed parts and dimensioned drawings. The top of the Main Body part was flat so that the main microcontroller, Arduino Due, can be mounted with #4 screws. There were four holes on the body matching the four hole pattern on the Arduino Due. These holes were used to secure the Arduino to the body. Along the perimeter of the body, there was an array of holes just wide enough for a #4 screw to fit through. These holes were used to mount a Circuit Board Holder part,

as well as Contact Switch Holder parts. Holes which were not used for fastening parts were used for cable management. Cable ties were snaked through these holes to secure various wires and cables. Figure 20 shows a close up exploded side view for illustration. Contact Switch Holders support contact switches which were fastened with a #2 machine screw. Figure 20 also shows how the Shell Mount part slides and snuggly fits into the Circuit Board Holder. The Shell Mount part, as the name suggests, was used to mount the shell. The shell pivoted on a #10 screw which served as an axle. The shell segments rest on contact switches and closed the switches when an external force was applied. The spring inside of the contact switch was stiff enough for the shell to rest on and returned the switch to its default state when an external force was applied and then removed. The IMU Extension part was added to the shell holder with a #4 screw. A #4 screw was also used to secure the IMU to the IMU Extension circuit. The IMU Extension part was used to raise the IMU unit away from the circuitry because the magnetometer inside of the IMU would give false readings if placed near circuitry. The main circuit board was placed on top of the Circuit Board Holder and the Power Circuit was placed on the back of the robot, right behind the Arduino so that both of these circuits were protected by the shell. All other (smaller) circuity was hot glued to the top of the Shell Mount.

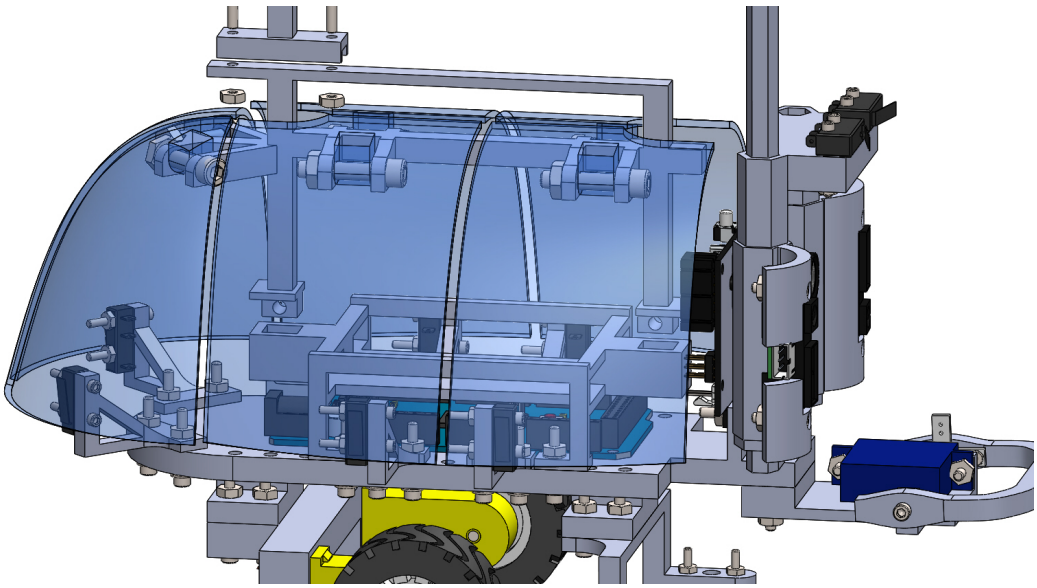


Figure 20: Parts bolted to the main body

Underneath the body, there were two points of attachments. These were used to attach a drive system allowing for flexibility and a modular design. In our case, a simple differential wheel drive locomotion system was implemented. If needed, one could swap the current drive system with another design without having the need to take the whole robot apart. Other designs could feature different wheels, treads, or even leg mechanisms provided they are driven by two motors which would be compatible with an on-board motor driver. Assembly of a drive system is illustrated in the exploded view in Figure 21.

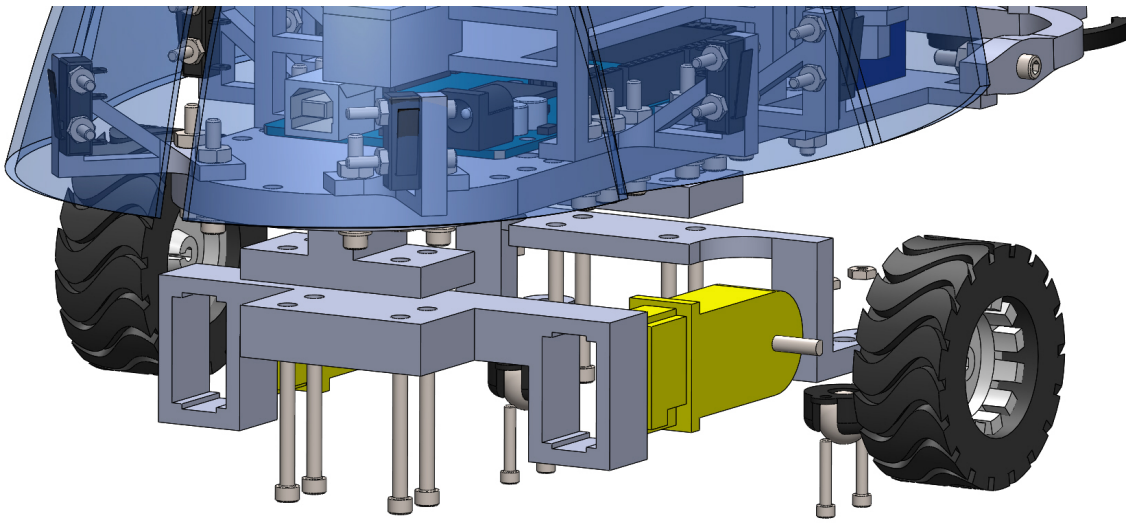


Figure 21: Drive system assembly

The drive motors were tightly pressed into the Motor Mount part. In this particular drive system, wheels that are 24mm in diameter and 19mm wide were used. These wheels were plastic and had a D-profile bore which coupled nicely with the motor's 3mm D-profile shaft. The plastic hub also had 48 outside teeth which could be used with an optional quadrature optical encoder. The plastic hub carried a rubber tire. The Motor Mount part was attached to the Main Body with #4 screws. At the front of the robot, the Caster Holder part was also attached to the body with #4 screws. There were two metal ball casters attached to the Caster Holder with #2 screws. The metal ball casters were in place for support and to make sure that the robot did not tip forward.

There were two holes which were used to mount an arm assembly and the Hexagonal Pole Part. Figure 22 shows how the arm assembly was assembled and mounted to the body. The Arm Mount part was connected to the main body with two #4 screws. The same screws were used to attach the Hexagonal Pole part to the body. A servo motor which moves the arm up and down was placed on the Arm Mount and was secured with #4 screws. One one side, the Gripper Assembly was attached to the

servo horn with #0 screws. On the other side, a #4 screw was used and doubled as a pivot for the Gripper Assembly. Figure 22 provides a view of how the Pixy camera was attached to the Main Body with #4 screws.

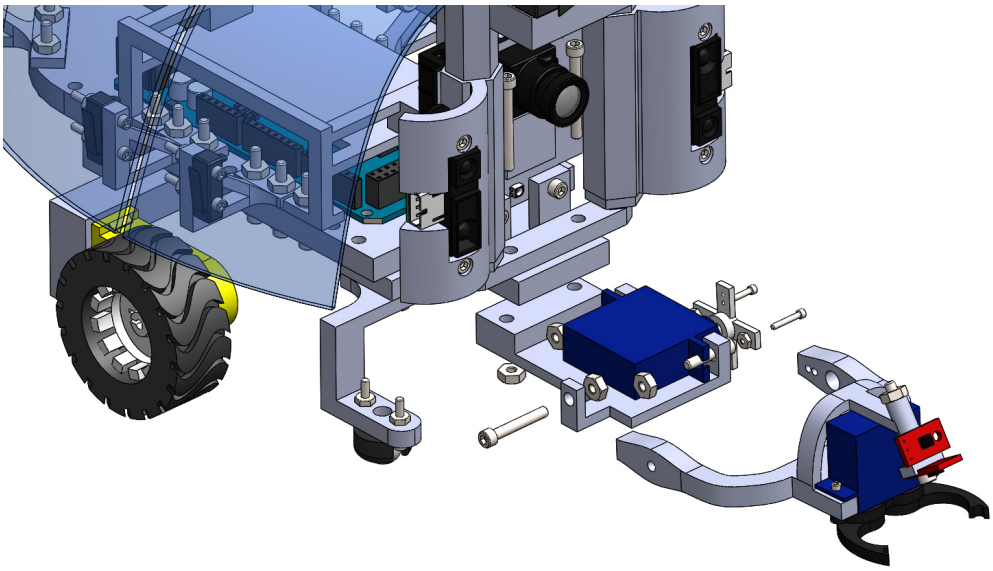


Figure 22: Arm assembly

Figure 23 shows how the Gripper Assembly was assembled. The grip servo motor slided in the Gripper Holder part. The claw mechanism was attached to the servo output shaft, and then #0 screws were used to secure the motor and the claw to the Gripper Holder. The analog proximity sensor (which was used for grip feedback) was mounted to the Gripper Holder with a #4 screw. Lastly, the digital proximity sensor (used for head bump detection) was glued to the Gripper Holder.

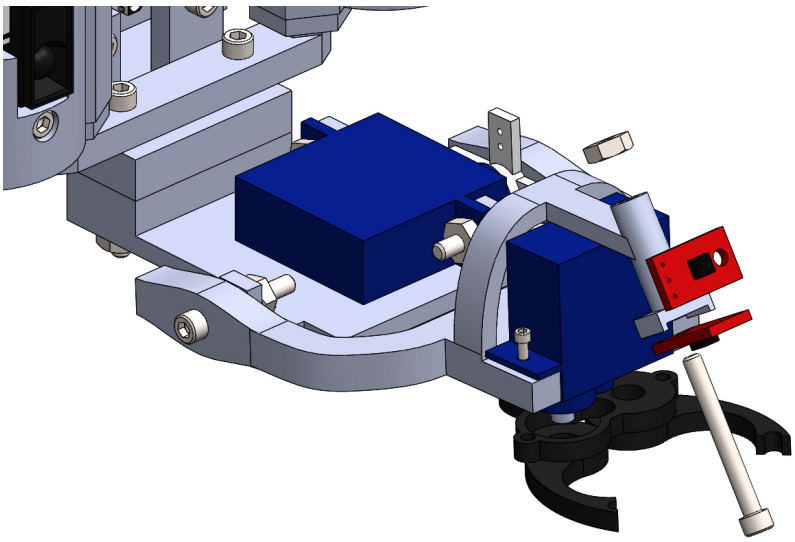


Figure 23: Gripper assembly

The last part that was attached to the main body was the Hexagonal Pole part. As mentioned earlier, it was attached to the body with the same #4 screws that were used to fasten the Arm Mount. The Hexagonal Pole part was a core of the Front Assembly which is illustrated in Figure 24. The IR Holder parts had a hexagonal bore and slid right on top of the Hexagonal Pole. The IR Holder part had a socket for the infrared sensor to fit flush. The IR Holder also doubled as a guard to cover and protect circuitry on-board. The IR Holder was secured with a #4 screw. Once in place, the Switch Bar was installed. The Switch Bar also had hexagonal bores and slid on top in a similar fashion. Two contact switches were secured on top of the Switch Bar with #2 screws. These switches were used to detect if the robot made it to the dumping site. Hot glue was used to bond parts to each other.

The Hexagonal Pole part was also used to carry the charging electrodes. A piece of conductive brass mesh was used to make the charging electrodes as shown in Figure 13. Alternatively, the charging terminals could have been made by winding stripped wire around the hexagonal pole.

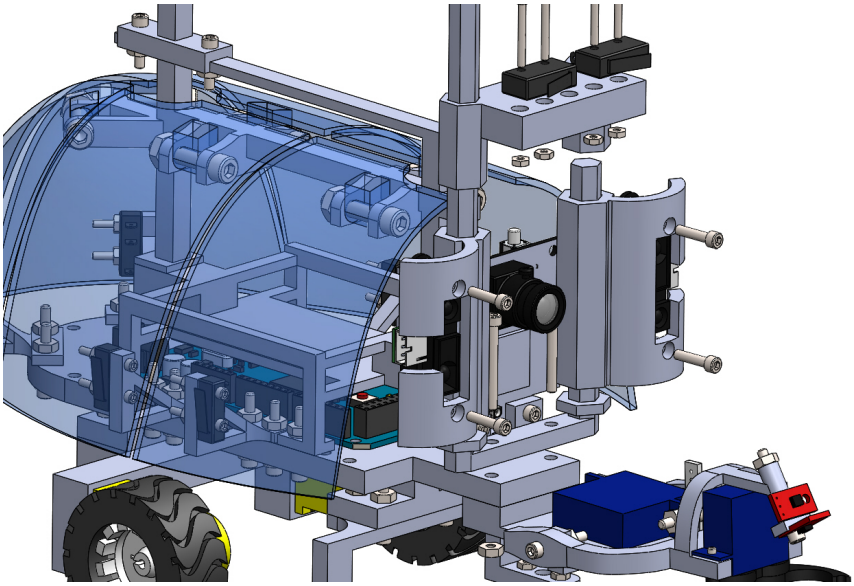


Figure 24: Front assembly

4.3 Assembled Robot

Pictures of a fully assembled robot are shown in Figures 25 - 27.

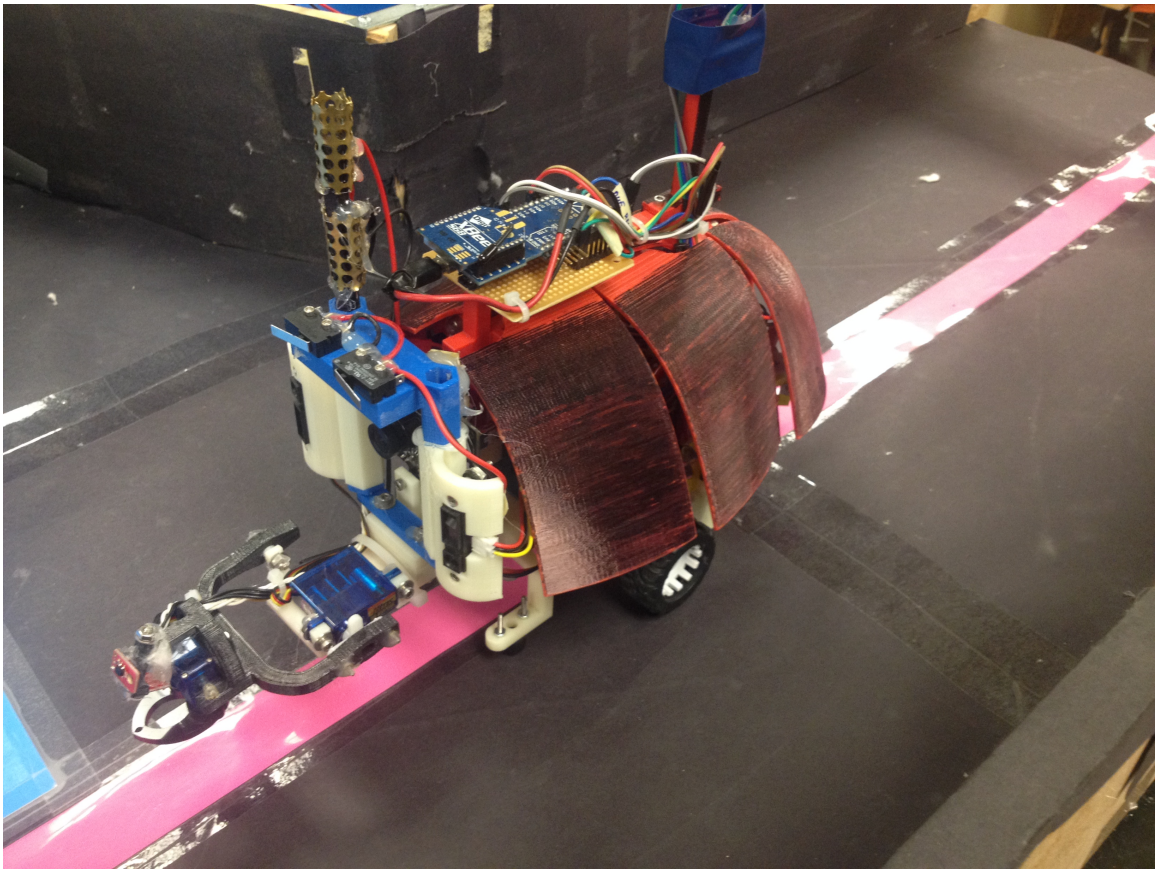


Figure 25: Assembled robot

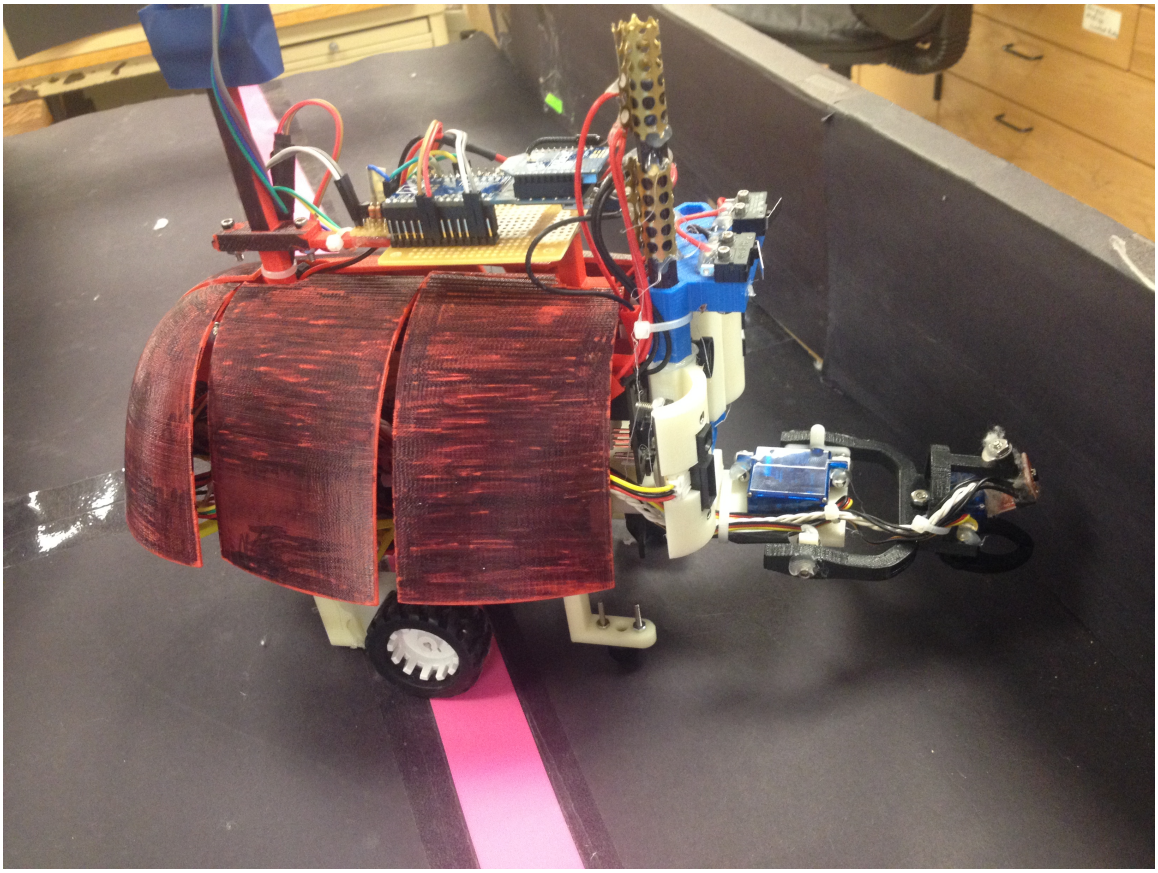


Figure 26: Assembled robot, side view

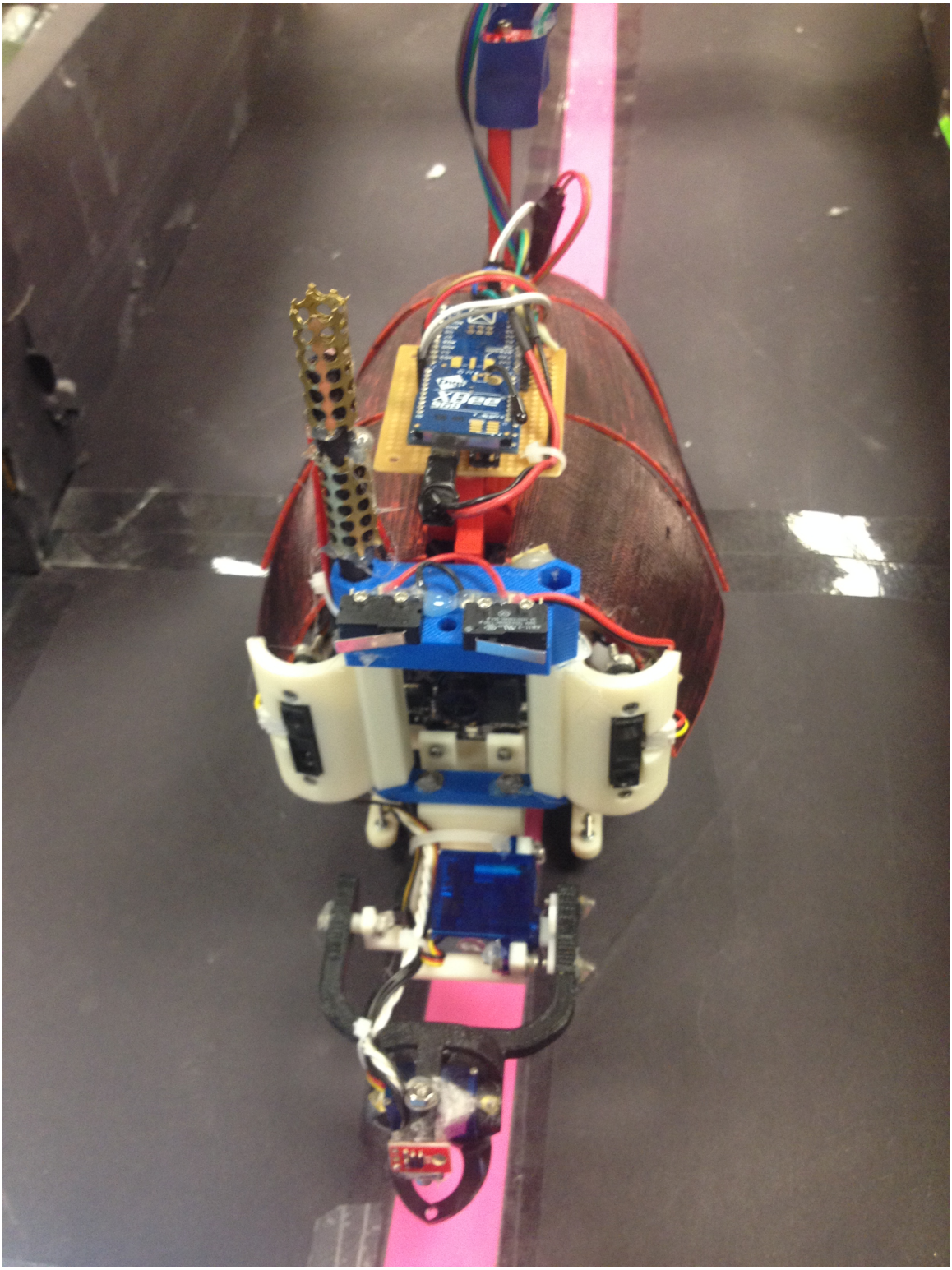


Figure 27: Assembled robot, top view

CHAPTER V

SOFTWARE OVERVIEW

In this chapter, we will discuss embedded code architecture for microcontrollers used in this project, as well as how other computer software was used for data acquisition and processing purposes.

5.1 Arduino DUE

Arduino IDE was used to write and compile the code. The code contains two important loops. The first loop is a setup() loop which runs only once as soon as the microcontroller starts. All global variables and class declarations and initiations are performed in the setup. The second loop is a main() loop which runs forever and ever. In other words, the main() loop is equivalent to while(1) loop without a break statement. Inside of the main() loop, global boolean variables are used for flow control and to select which mode of operation the robot will enter. The robot has several different behavior modes programmed and each of these modes will be discussed in the subsections below. The robot will autonomously change its mode of operation when appropriate. Simplified pseudocodes are presented in Figures 28–36.

To summarize, the robot had seven modes of operation. The "going in mode" allowed the robot to drive to the excavation area and search for cotton. The "digging mode" was triggered once the cotton was found. The robot would make several attempts to pick up cotton. In the event of success, the robot engaged the "going out mode". The "going in mode" was engaged if the robot failed to pick up cotton in several attempts. The "going out mode" allowed the robot to bring excavated payload to the dumping site. The "deposit mode" was triggered once the cotton deposit site was found. In the deposit mode, the robot disposed of the payload. Next, the "going

in mode" was enabled and the cycle was repeated. "Resting mode" could be randomly engaged after the "deposit mode" instead of the "going in mode" if the robots were allowed to be lazy. The probability of a robot engaging and remaining in a "resting mode" (lazy mode) would be taken from experiments conducted in a parallel study in our lab involving observations of fire ants and assessment of probabilities of ants becoming lazy. If the robot sensed that the battery was running low, it would switch its mode to the "going charging mode". The "going charging mode" was only allowed to be engaged when the robot was in the "going in mode" or the "resting mode". The "going charging mode" made the robot find and drive to the charging station. The "charging mode" was enabled when the robot made physical contact with the charging station, while in the "going charging mode". The robot maintained contact with a charging station until the batteries were adequately charged. The robot was programmed to find the charging station again if contact with the charging station would be lost.

5.2 Notes on Pixy Camera

As mentioned earlier in the Electrical Design chapter (Chapter III), The Pixy is a camera sensor that can detect objects of different colors. All image processing is done on board the Pixy's CPU. The user could set the Pixy to search for up to seven different color signatures. Pixy reports information to the Arduino if an object of a matching color signature is detected.

Pixy comes with a software called PixyMon. PixyMon was used to monitor what the camera was seeing and to set the color signatures. Example calibrations are shown in Figures 37 and 38.

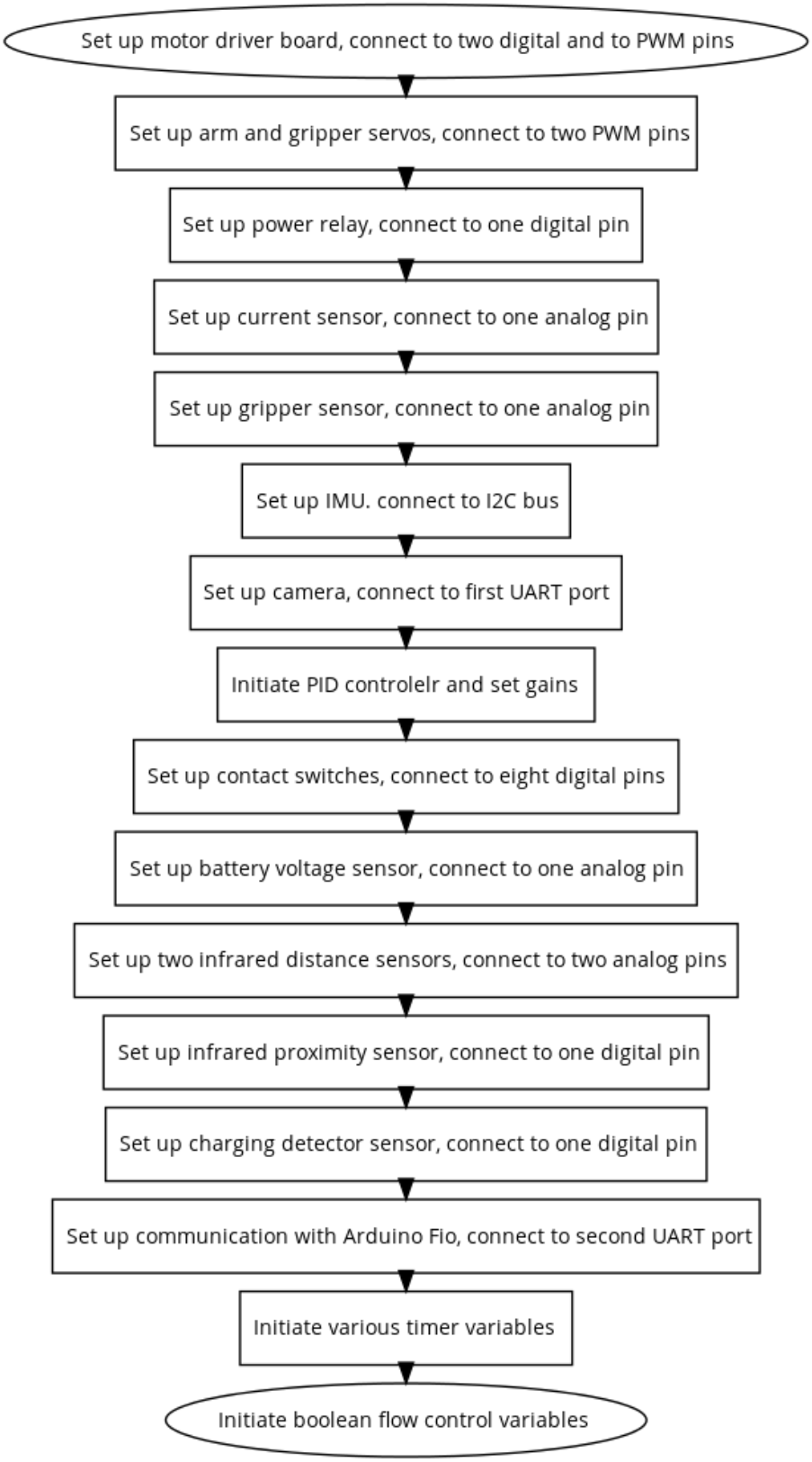


Figure 28: Setup Loop

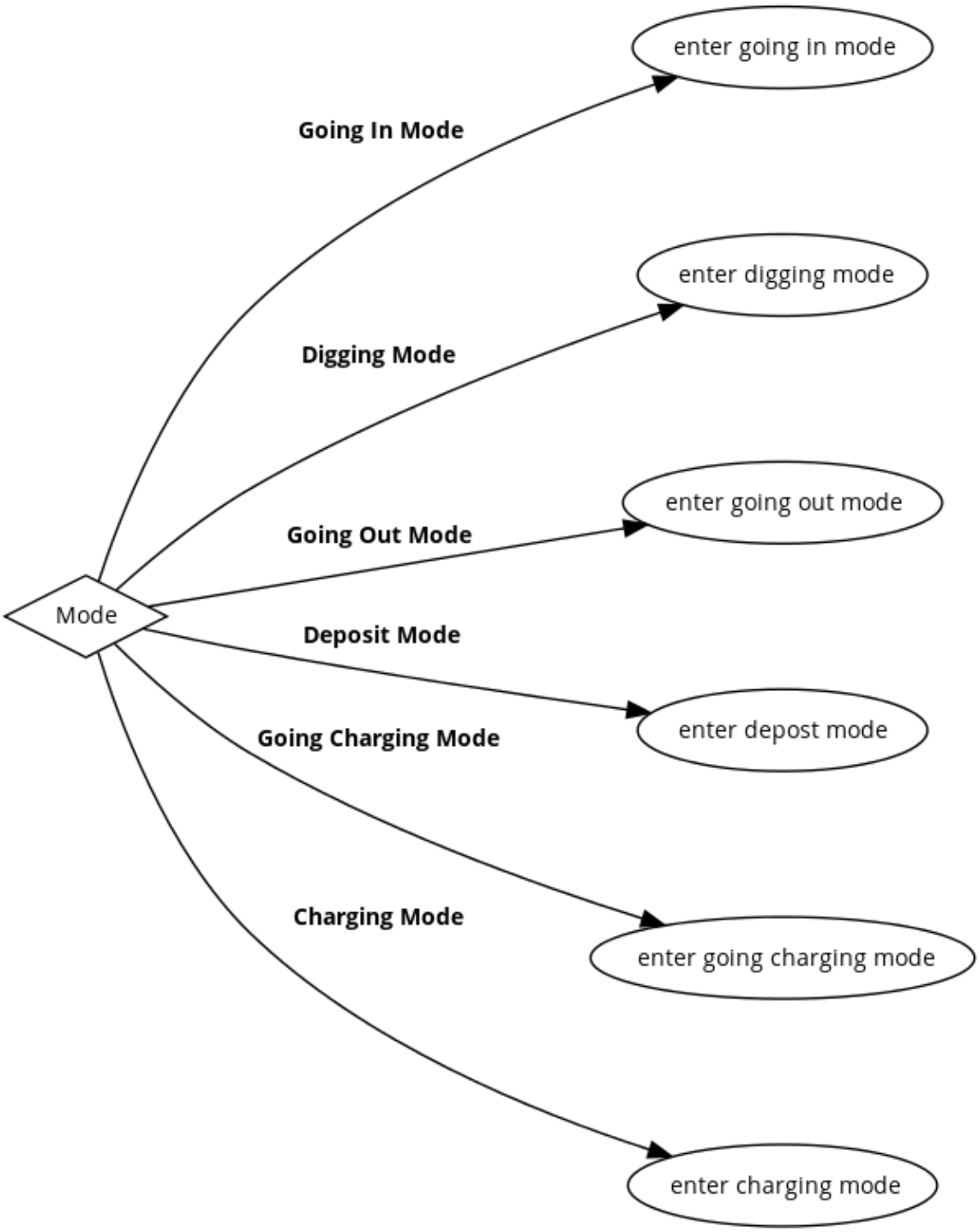


Figure 29: Main Loop

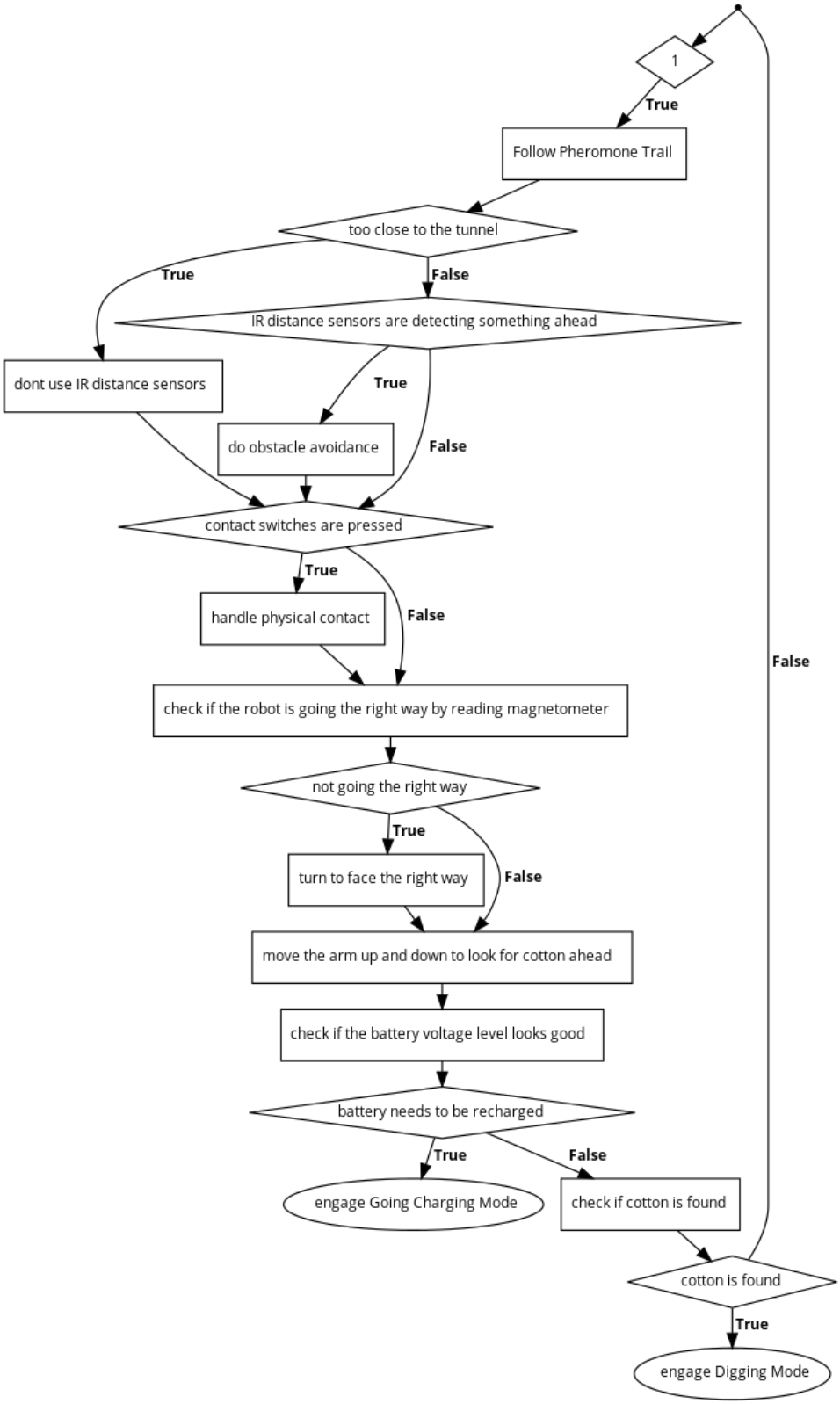


Figure 30: Going In Mode

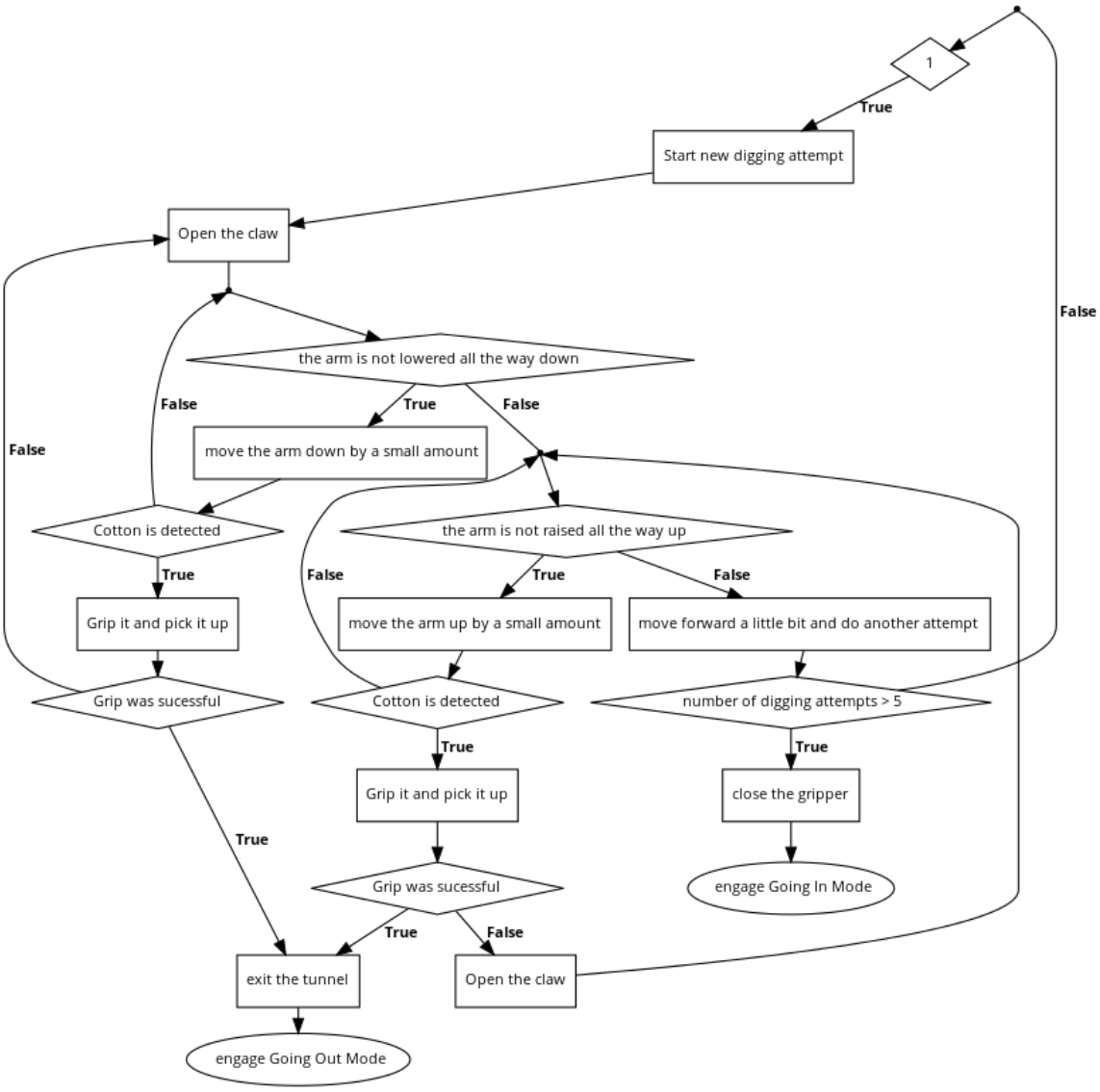


Figure 31: Digging Mode

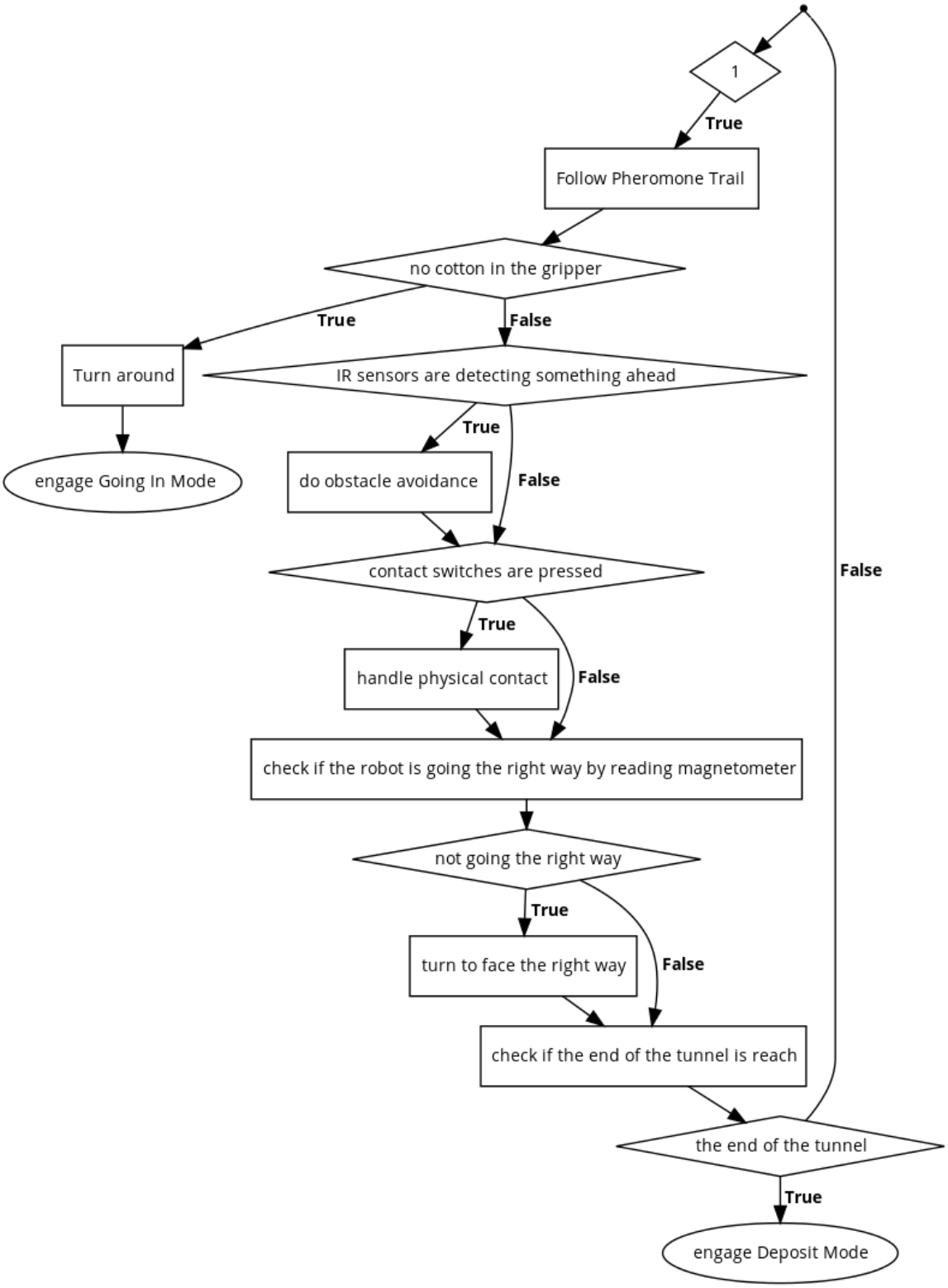


Figure 32: Going Out Mode

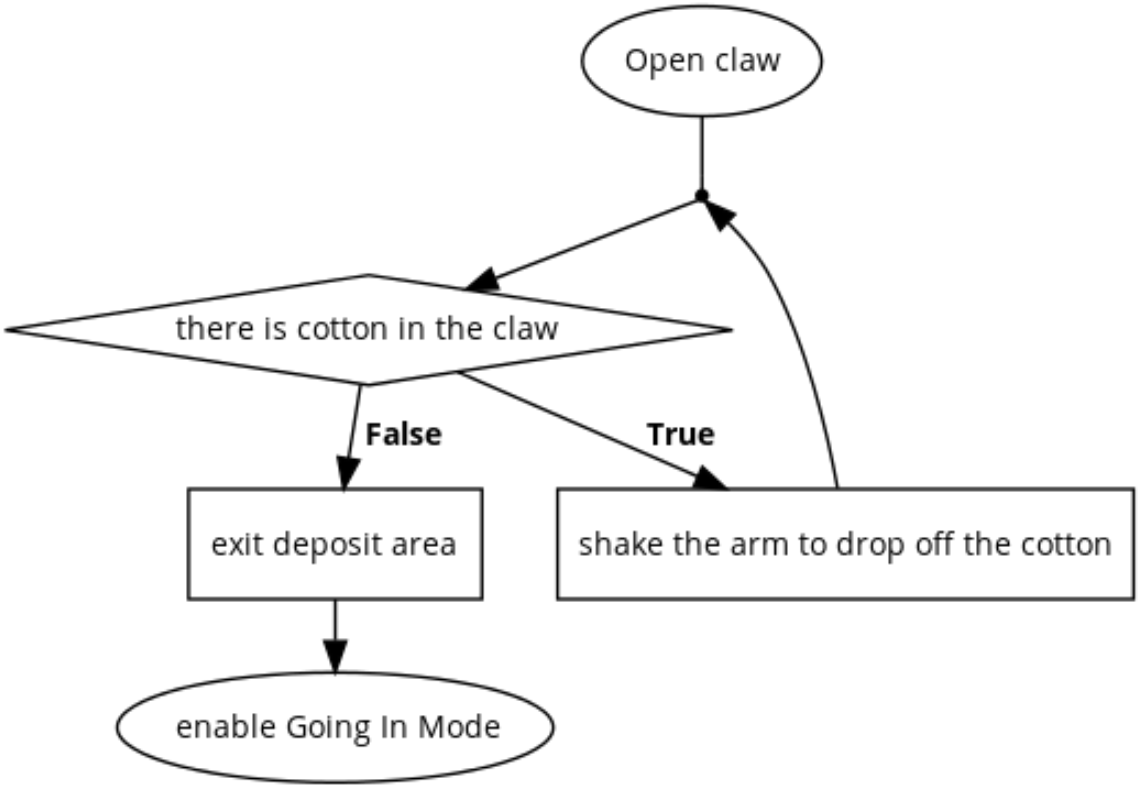


Figure 33: Deposit Mode

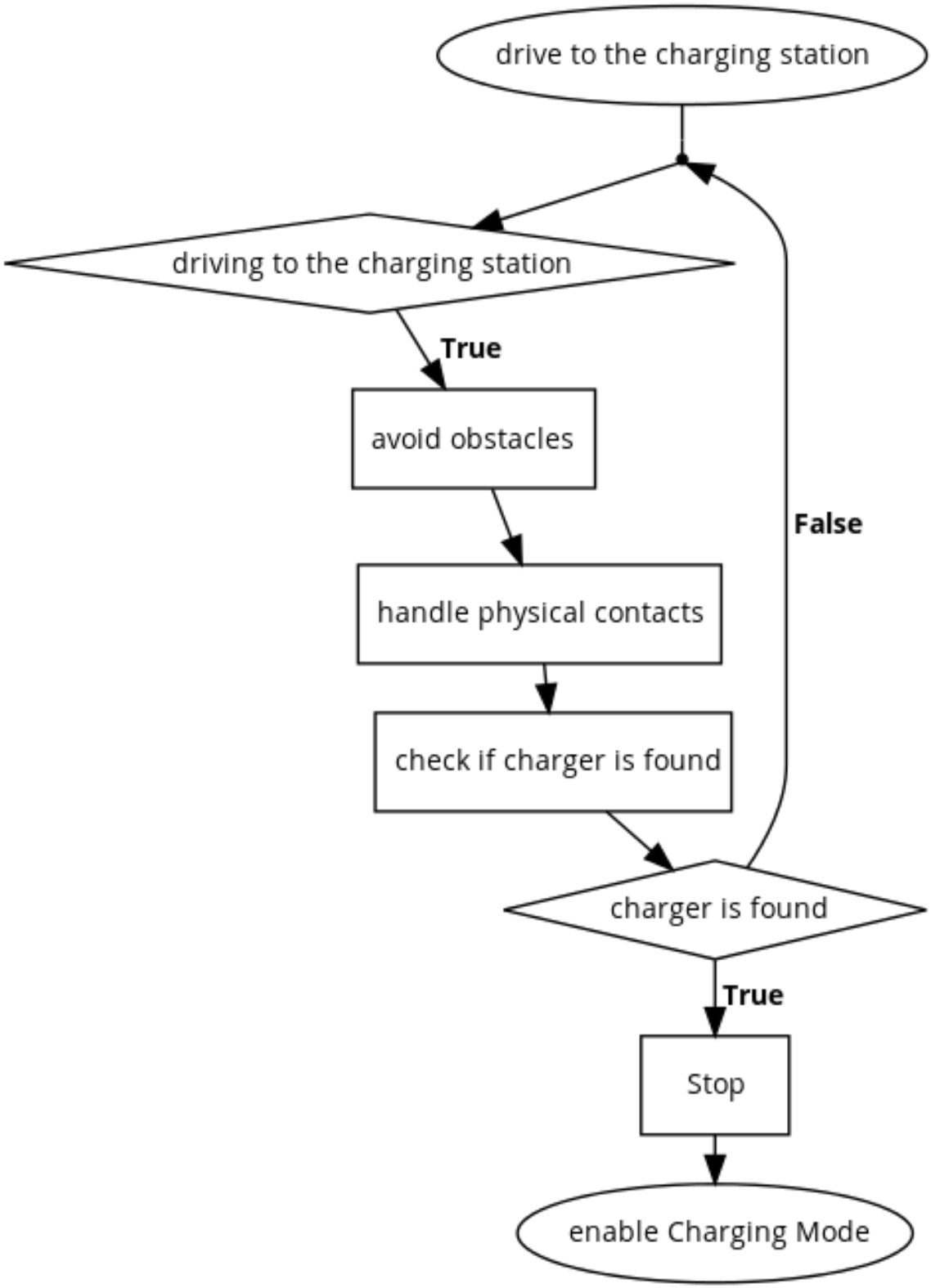


Figure 34: Going Charging Mode

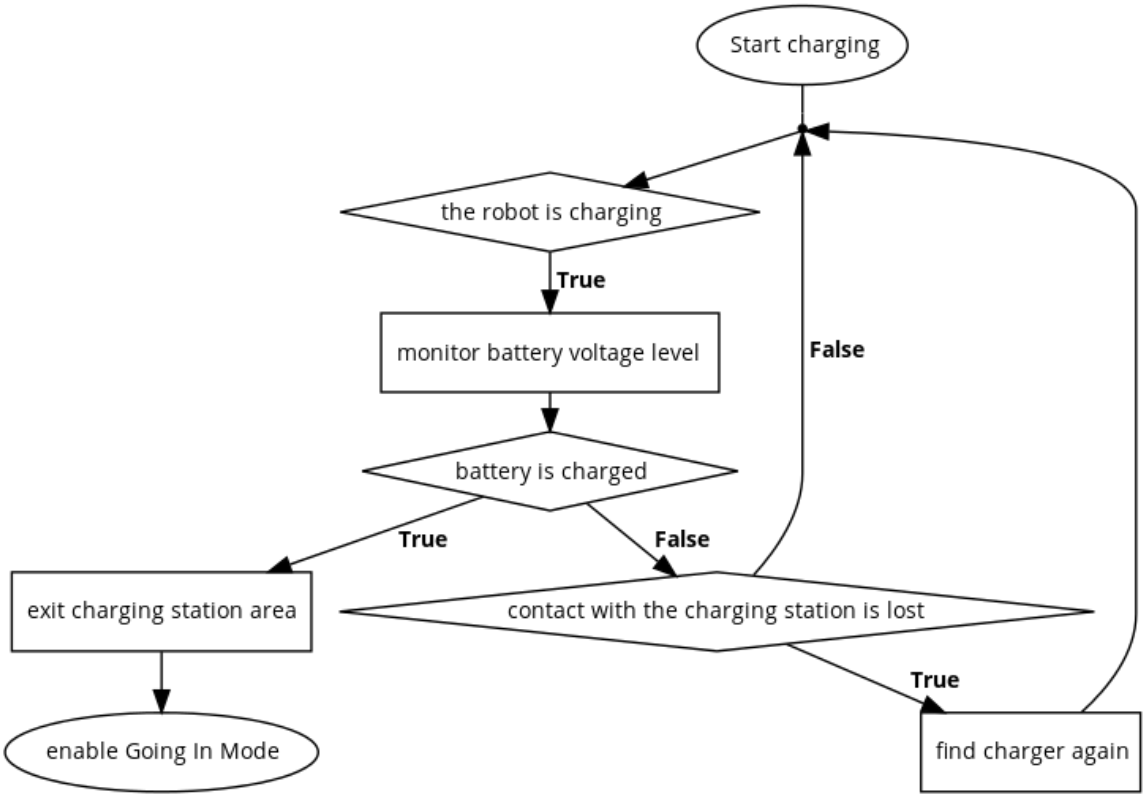


Figure 35: Charging Mode

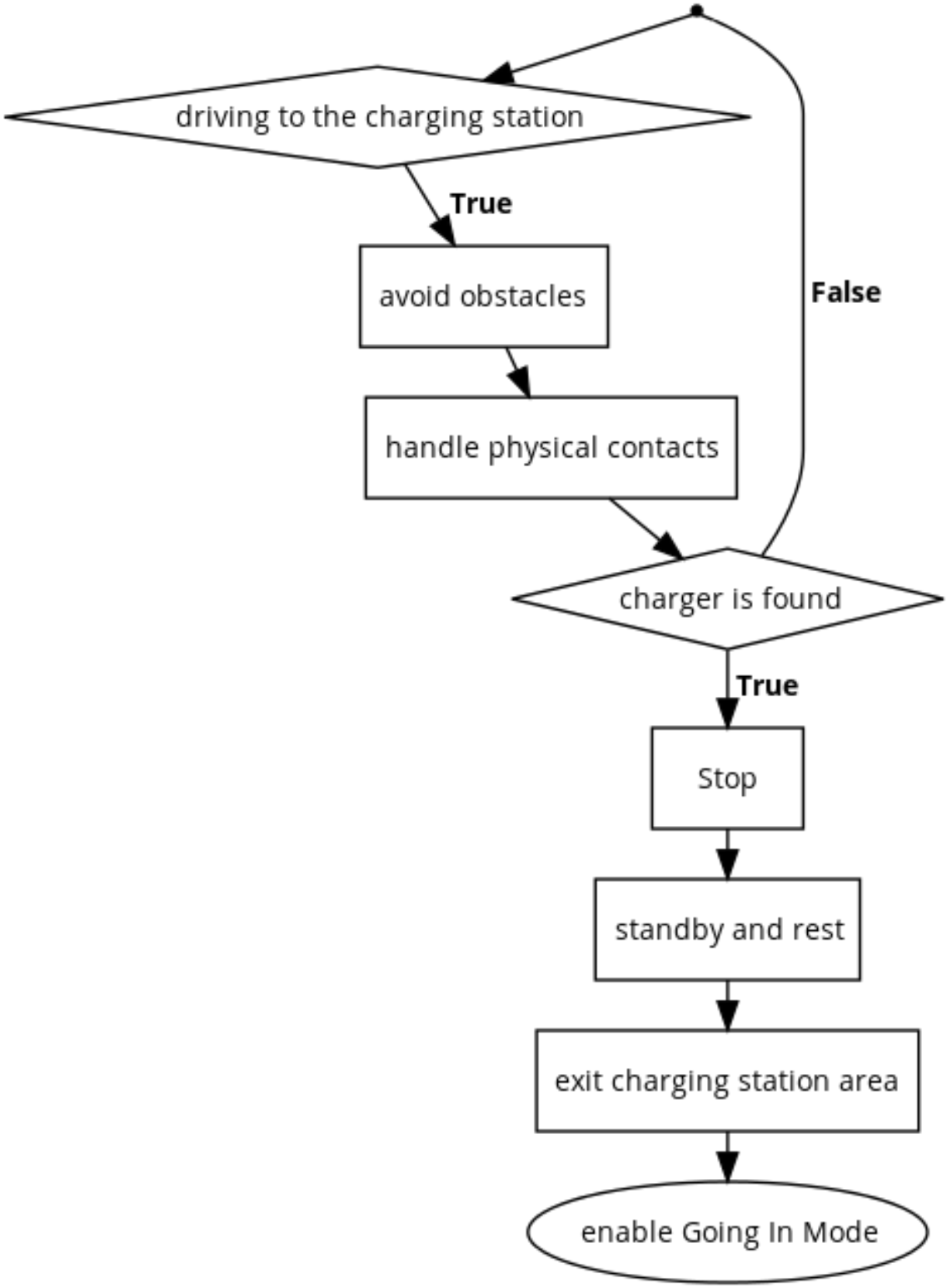


Figure 36: Resting Mode

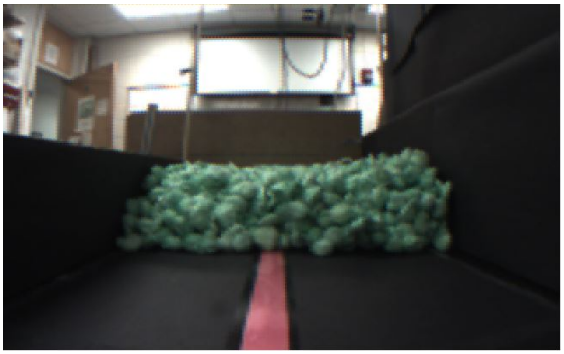


Figure 37: Raw video frame

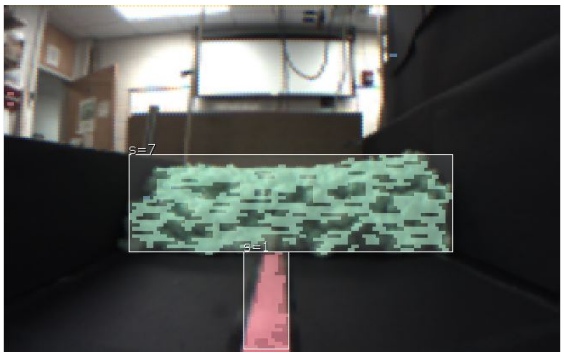


Figure 38: Processed video frame

Figure 37 shows a frame from the camera's unprocessed video feed. PixyMon software calls this "raw" video. At this instance, Pixy was mounted on the robot inside of the test bed facing cotton balls. Figure 38 shows a "cooked" (processed) video frame. The Pixy was set to look for a pink tape (pheromone trail) and green cotton balls. The processed video frame has a box drawn around pink tape, and another box around green cotton balls. The box around pink tape has "S=1" label which means that this box contains pixels matching color #1 signature. Likewise, the box around cotton balls has "S=7" label which means that the box contains pixels matching color #7 signature. Information about each box, such as label, pixel width, pixel height, and pixel centroid coordinates will be reported to the Arduino via the UART connection. Each detected group of pixels is called a "block", and Pixy can detect multiple blocks of the same signature as shown in Figures 39 and

40. There are three detected blocks in Figure 40. The first block represents pink tape. The second block represents a big pile of cotton. The third block represents a single cotton ball placed away from the big pile. In addition to the pink tape (the pheromone trail) and cotton balls (the simulated granular media), the Pixy was also configured to identify the charging area and charging station, as shown in Figures 41 and 42. Blue construction paper was used to mark the charging area (color signature #4) and bright yellow socks were used to mark the direction of a charging station.

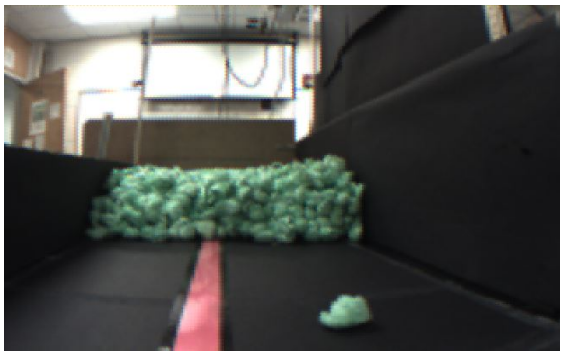


Figure 39: Raw video frame, multiple signatures

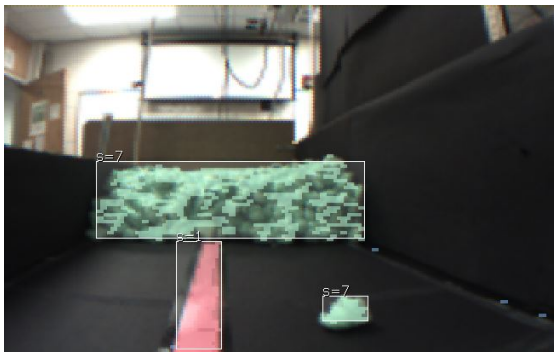


Figure 40: Cooked video frame, multiple signatures

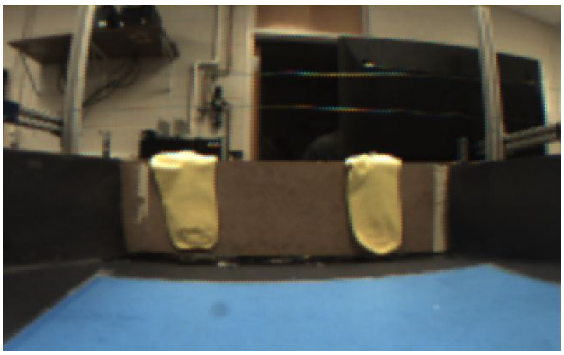


Figure 41: Raw video frame, charging area

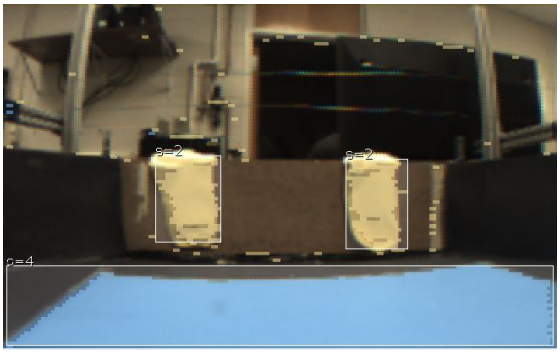


Figure 42: Cooked video frame, charging area

5.3 Arduino Fio

The code running on the Arduino Fio is fairly straightforward. Analog pins on the Arduino were used to read the voltage and current sensors. The Fio sampled each sensor 100 times, computed the average and sent it out via XBee radios every two seconds.

The Arduino Fio also had the capability to reset the Arduino Due. If the Arduino Due detects that something is not going right, it will send a reset request to Arduino Fio. Arduino Fio has a has a digital pin connected to a reset circuit which will be used in the event of a reset request.

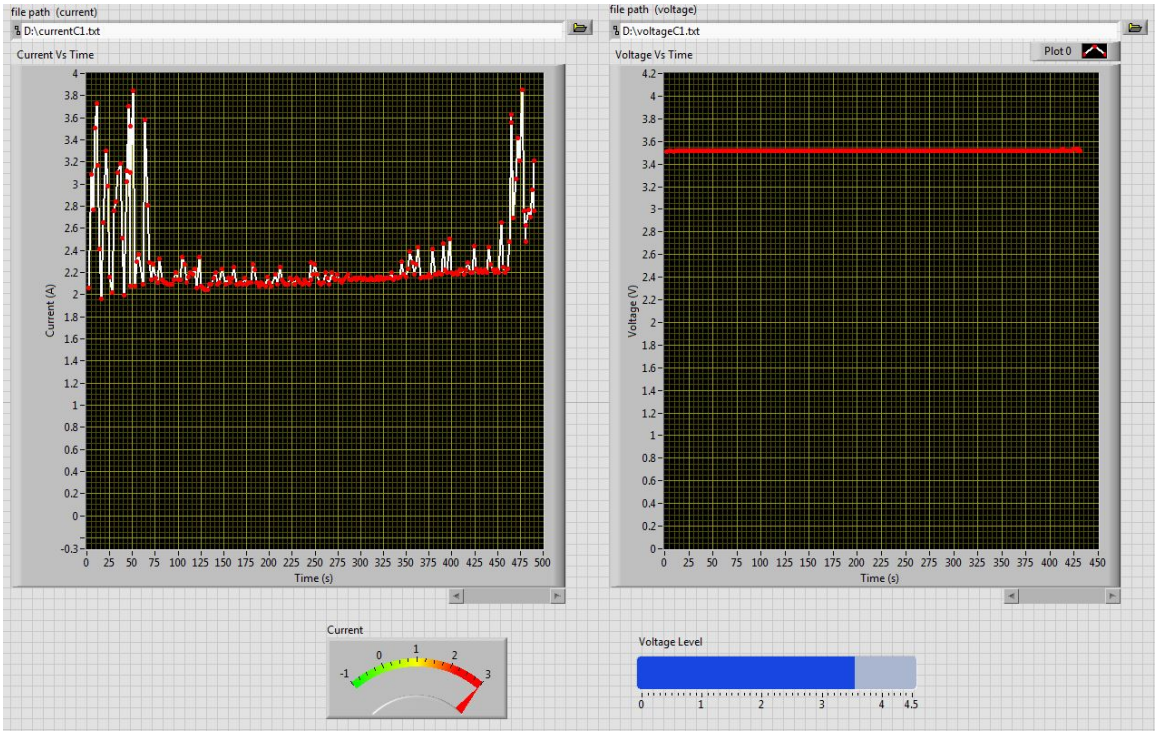


Figure 43: LabVIEW program developed for monitoring power consumption of each individual robot in real time

5.4 Digi XCTU

Digi XCTU software was used to configure XBee radio devices. Each pair of radios was set to operate on its own channel at the maximum RF level.

5.5 LabVIEW

LabVIEW was used to create a program for data aquisition. The program had two functions: video capturing and receiving wireless transmissions from the robots. LabVIEW controlled a webcam above the test bed. The camera was configured to record and save a frame every four seconds so that time lapsed videos could be generated. Receiving radios were connected to LabVIEW. LabVIEW would analyze incoming data packets, as well as sort and log data to text files. LabVIEW was also used to monitor the robot's power consumption in real time. A screen shot of a developed user interface is shown in Figure 43.

5.6 Matlab

Matlab was used to load data from text files and perform analysis.

5.7 Virtual Dub

Virtual Dub software was used to make .avi format videos from frames saved out by LabVIEW. Virtual Dub tools, such as compression and cropping were used for video processing.

CHAPTER VI

RESULTS

In this chapter, we will discuss results, observations, and insights gained from constructing and testing digging robots.

6.1 Observations

By attempting to build a robot that simulates ants, it became clear that ants are quite sophisticated organisms who manage to solve a series of problems and challenges associated with social digging of granular media. Some of the observations made while attempting to build robotic ants are shared here.

6.1.1 Digging

At the beginning of the project, the task of excavating and gripping granular media was falsely assumed to be trivial. How hard could it be to grab a few particles when there is a big pile in front of you? The first robot prototype was a differential drive robot with a similar robotic arm. The arm could be raised up and lowered down, and had a gripper claw attached at the end. Before development of autonomous software, the robot was manually controlled with a joystick. Gripping, picking up, and moving hard particles, such as plastic BB's, and 3D printed spheres was a difficult task. The spheres would slip out of the gripper most of the time. The success of picking up these spheres was sensitive to where the gripper gripped the spheres, and the amount of force the gripper applied to the spheres. One can visualize this by picking up a sphere with one hand, or a cylindrical object such as a plastic bottle or a pen, hold it only with a thumb and a middle finger, and start squeezing hard. More than likely, the object will slip out of the hand. Training the robot to grip an

object at a specified location autonomously is challenging and is probably not possible with robot's current set of sensors. This may or may not be accomplished with the development of some sophisticated machine vision algorithms alone. The problem becomes even harder when the particles that the robot encounters are different sizes, as well as not guaranteed to be spherical. Controlling the amount of force that the gripper exerts on a particle is also problematic. The servo motor which opens and closes the gripper can be commanded to a specific angle (with 1 degree resolution), and an onboard controller will apply the torque to move. Torque control could be achieved if the controller on the servo was replaced with something custom. Torque control could be possibly achieved if the current supplied to the servo motor was controlled. An even harder question would be a development of a feedback mechanism that can be used to decide if more or less torque should be applied. The sensor would need to be quite small, light, and be able to send out data fast.

Ants were also observed to use their head to scrape the tunnels as an excavation method, and then use their body to compact excavated media and form large pellets, which are then picked up and carried out [21]. We were able to somewhat mimic this behavior by using the robotic arm as a shovel and securing a bulk of cotton balls between the body and the arm, holding it, and then moving it to the dumping site. This was done with a joystick. Even with a human mind used as a controller, it was a difficult multi-step process. Efforts were made to implement this mode of digging autonomously but were unsuccessful. Some clever feedback mechanism was needed. Ants are masters at excavating and manipulating granular media.

6.1.2 Cotton Depositing

Once the robot was smart enough to autonomously pick up cotton and move it to the end of the tunnel, a logistical problem was encountered. What do we do with excavated cotton? The robot was first programmed to dispose of its excavated payload

somewhere at the end of the simulated tunnel. The robot was set to continuously dig and move the cotton to the end of the tunnel where it was randomly dropped. It wasn't long before a huge unorganized cotton mess was formed. The cotton would cover the simulated pheromone trail which would interfere with the robot's ability to navigate. Loose cotton also inhibited locomotion, which will be discussed later. Managing excavated cotton was left out of the scope of the project for now. This is why excavated cotton is simply being dropped off the table so that it does not interfear with robot's locomotion and navigation. Ants must have a good system for dealing with excavated granular media.

6.1.3 Locomotion

Moving on granular media is a challenge that many organisms perform well. Not only can ants move on the granular media, but they can also run up and down the tunnels made from granular media. Wheels were used to move the robot around on a horizontal flat surface.

6.1.4 Navigation

The robot used a camera and an IMU to navigate. Real ant tunnels are dark, which brings the questions of how ants perceive the environment around them without light, what is the range of an ant's perception, and how do they know which direction is up? These are all interesting questions which could be studied. Our experimentation with multiple active robots suggests that counting steps may not necessarily be a reliable way for ants to navigate. We had encoders installed on a robot's drive motors. An encoder is a device which allows to measure the rotation of the wheel which can be used to measure the distance traveled. When two robots collide or bump, they displace and move each other introducing an error between the actual position and presumed position. In the case of the robot, collisions and bumping interactions caused the wheels to slip, leading to even more discrepancies. The simulated pheromone trail

(along with a PID controller for motors) allowed the robot to get back on the right track after a collision and get to the desired destination without a need to count steps. This could be a hint why pheromone trails are so important to ants.

6.1.5 Agility

First attempts to make the robot dig straight tunnels, which are only two body width in diameter, were unsuccessful. Initially, the robot would succeed in starting the tunnel and then accidentally destroy it. One example of an accidental tunnel destruction was a cotton ball getting stuck in the wheel and causing the wheel to jam. This made the robot turn in place since the other wheel would be still driven. Next, the robot's body would hit the tunnel wall, making the tunnel deform or collapse. Another example of an accidental tunnel destruction was a case where the robot rammed the tunnel walls. This could happen if the sensors were too slow to react. Likewise, the robot could run into the tunnel if the PID controller gave a sudden kick to the motors which can happen if the camera, a sensor used in a controller's feedback loop, briefly lost sight of the pheromone trail. If more than one robot makes it to an excavated tunnel, the interactions between them will more than likely result in the tunnel's destruction or collapse.

Observing the robots dig reveals that ants must be careful how they dig and maneuver inside of the tunnel. In addition, the ants must be especially careful to not destroy the walls when passing each other in the tunnel. Likewise, ants must be careful not to injure themselves or others while passing each other. Once in a while, we found a few loose bolts in a test bed after setting more than one robot to dig tunnels overnight.

6.1.6 Processing Capability

Building robot ants bolstered the fact that ants are capable of processing a lot of information. The bodies of ants must have a rich set of sensors and the means to

process all of the information.

6.2 Results

Designing and building robots that could be programmed to emulate ant's behavior was achieved. Three robots in total were constructed and labeled as a A, B, and C. The robot prototypes were tested and found to be able to operate for periods longer than 3 days without failures. The robots were observed to autonomously find the excavation site, excavate the simulated granular media, move the excavated payload to the collection area and drop the payload into the collection bin. Furthermore, the robots reported energy consumption at specified time intervals. Figure 44 shows digging activity snapshots of one robot over time. The digging activity of a robot was recorded with the overhead camera. As the time progressed, we observed that the the tunnel became longer and wider. We also observed that the test bed became messier. The number of cotton balls scattered around the test bed grew over time. The robot occasionally picked up large payloads and some cotton balls would fall off during the transportation from the excavation site to the collection site. Most of the scattered cotton balls resulted from cotton balls sticking to and jamming the drive system. This problem was not foreseen in the design stage but could be mitigated with a better drive system design and better control algorithms.

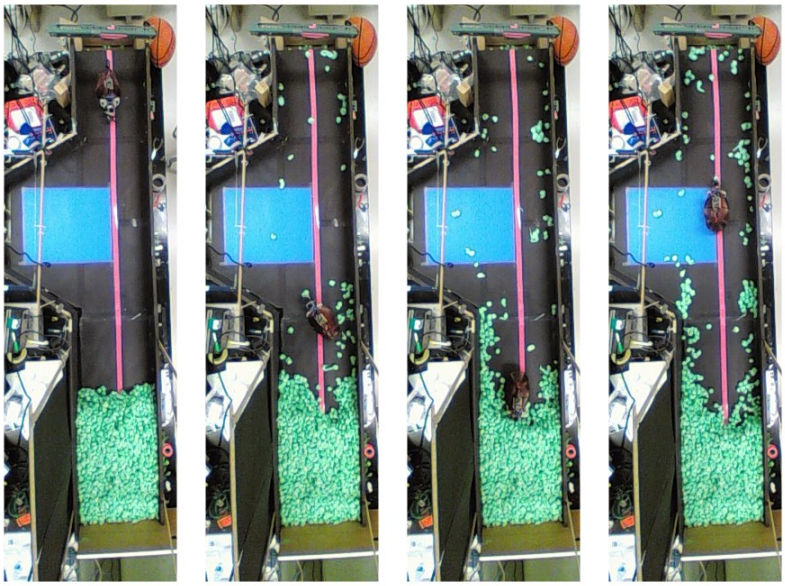


Figure 44: Snapshots of a single robot digging over time. The robot creates a tunnel by removing cotton over time

Traffic jams were not observed in a test bed with two active robots. The robots would occasionally collide but still manage to separate after the collision and move around each other. An example of a two robot collision is shown in a Figure 45. Figure 45 shows 14 snapshots recorded at two second intervals. The collision started after one robot finished digging and was in the process of turning around to leave the digging area. The robots pushed and brushed against each other during the collision and eventually got out of each other's way.

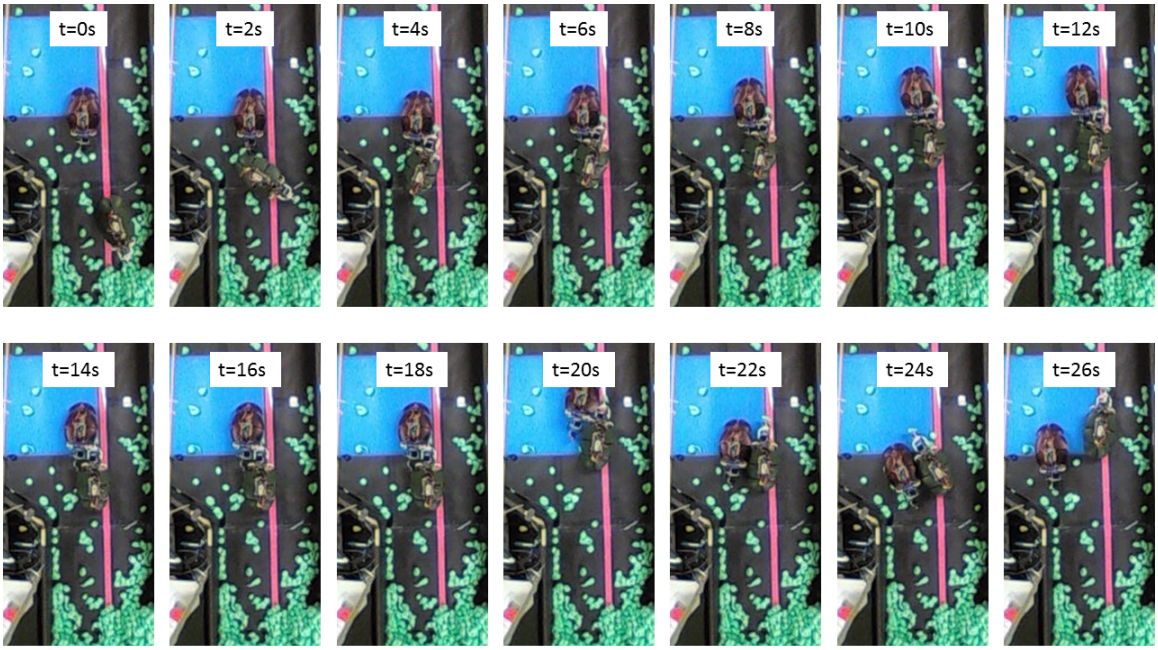


Figure 45: Video snapshots showing an example of a two robot collision. The robots spent 26 seconds to get around each other

A jamming effect was observed with three active robots. The three robots collided and it took some time for them to get past each other. An example of a three robot jam is shown in Figure 46. Figure 46 shows 33 snapshots recored at two second intervals. The jam happened after a third robot entered the digging area and was resolved after some pushing and shoving (t=64s). In the three robot experiment, the jamming occurred more frequently and lasted for longer time periods compared to the two robot experiment.

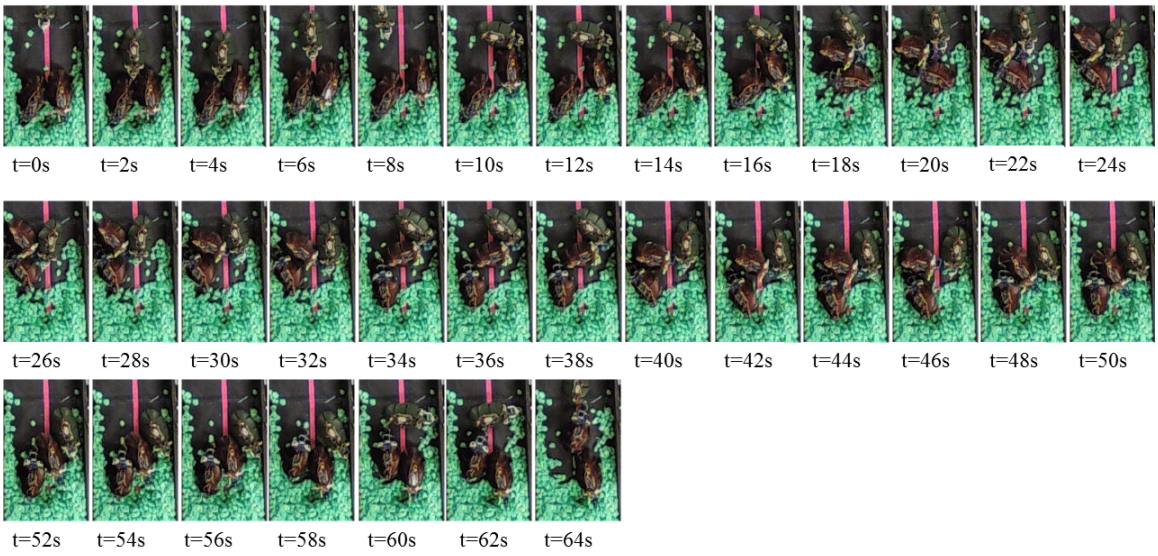


Figure 46: Video snapshots showing an example of a three robot collision. The robots spent 64 seconds to get around each other

The power consumption over time was comparable for all robots participating in the one and two robot experiments. This was the case since none of the robots have been subject to significant traffic jams. The power consumption of robots A, B, and C robots involved in the first two experiments is illustrated in Figures 47–49. There was no significant variation in current draw, battery voltage decay, and power for all robots. This suggests that the robots were consistent and that one robot was not better than the other performance-wise.

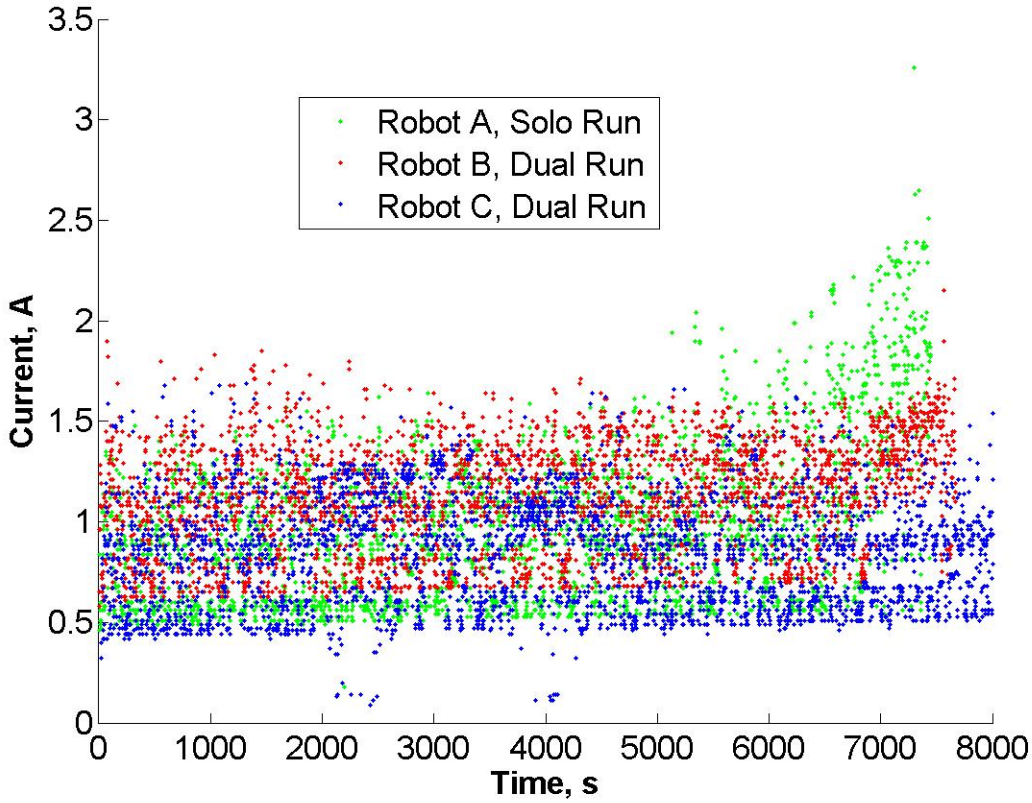


Figure 47: Current consumption vs time before charging

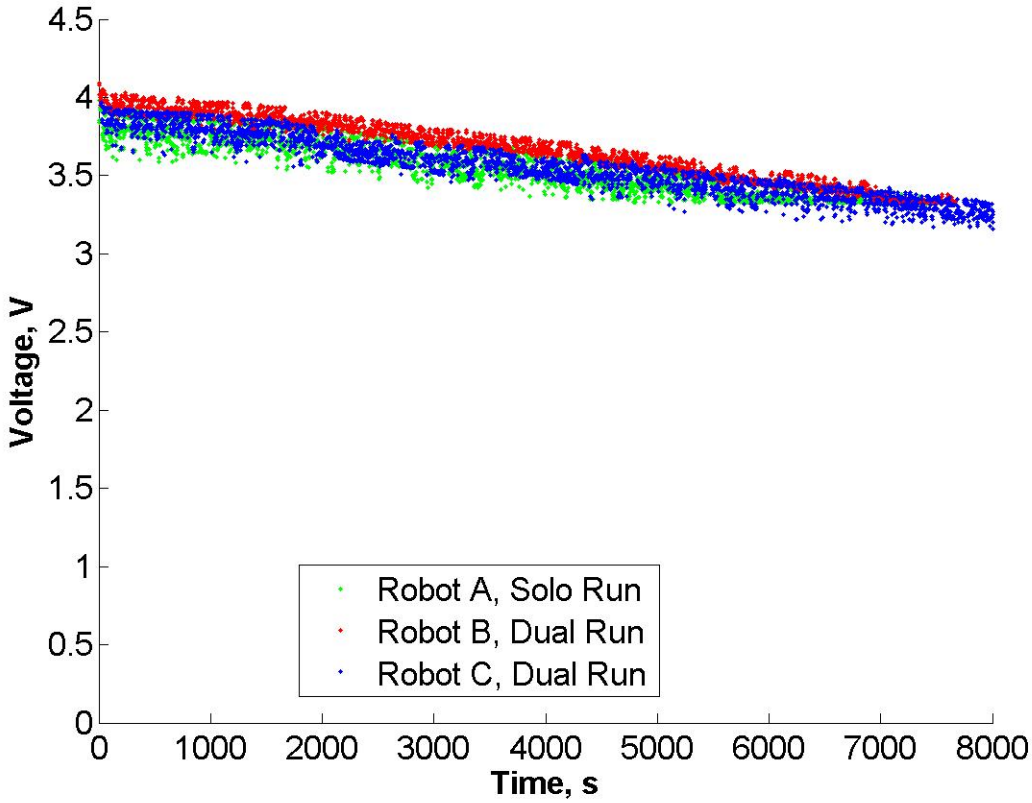


Figure 48: Battery voltage vs time before charging

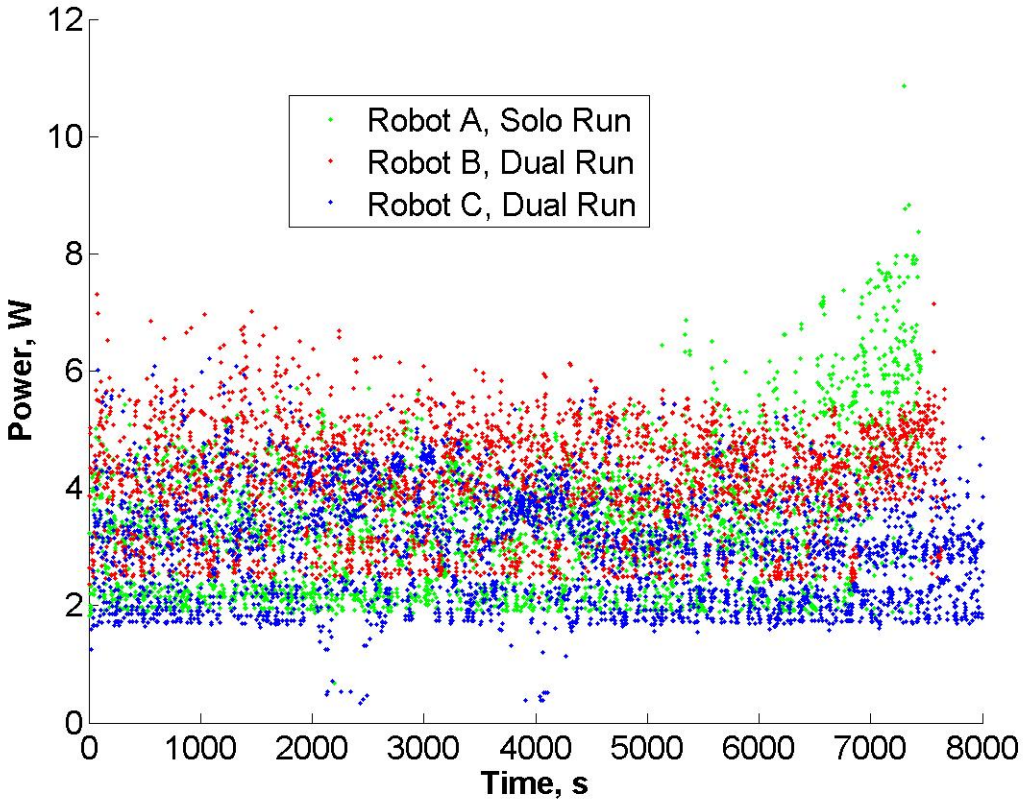


Figure 49: Power consumption vs time before charging

6.2.1 Preliminary Data

Three experiments were conducted. In the first experiment, a single robot was placed in the test bed and was allowed to be active for one activity cycle. An activity cycle starts when the robot enters the test bed with a fully charged battery and ends when the robot drives to the charging station. The robot's power consumption was logged and its activity was recorded with an overhead camera. In the second experiment, two robots were allowed to be active for one activity cycle. In the third experiment, three robots were used.

Figure 50 shows that the robots in all three experiments were individually consuming approximately the same amount of energy over time. This is not surprising since all robots were found to draw approximately the same amount of power over

time. Consequentially, a system of two robots proportionally draws more energy than a system of robots. Likewise, a system of three robots draws more energy than a system of one robot. This result is illustrated in Figure 51.

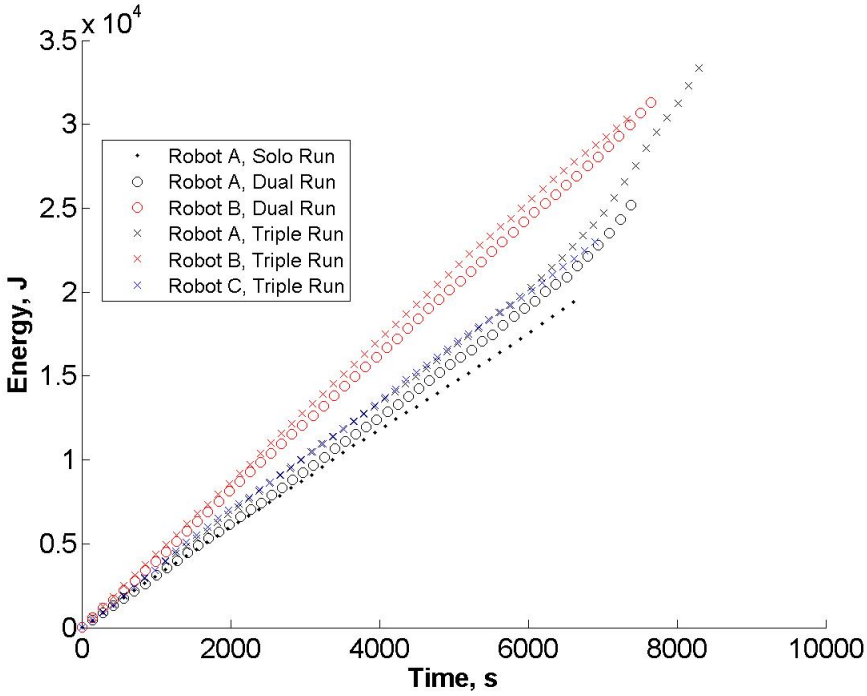


Figure 50: Energy vs time, per robot. The plot shows that the robots approximately draw the same amount of energy over time

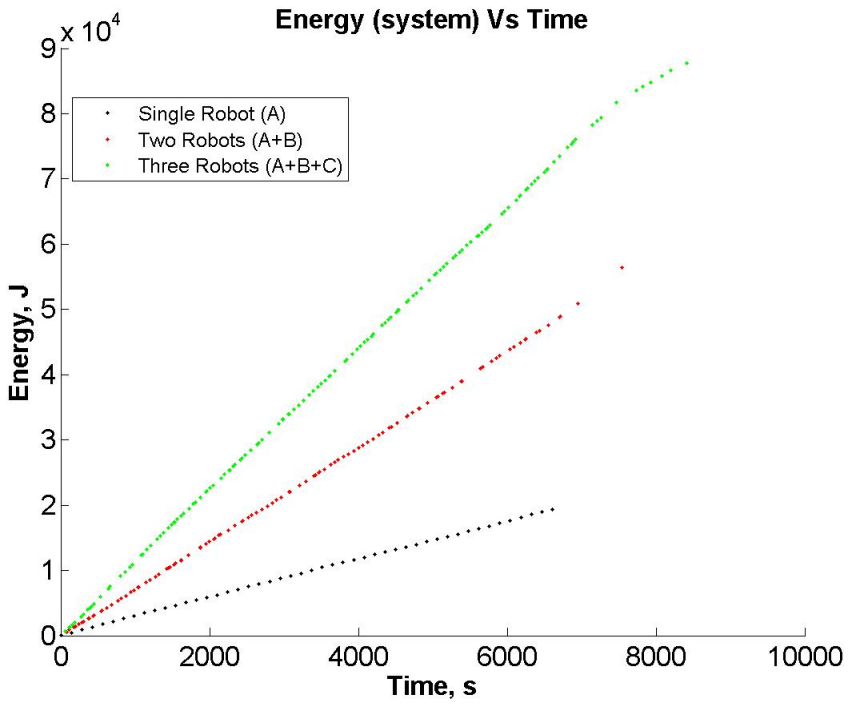


Figure 51: Energy vs time, per system. The plot shows that adding robots to the system proportionally increases the amount of energy drawn by the system over time

To evaluate the efficiency of a one, two, and three robot system, a number of cotton deposits will be considered. The amount of deposits made by the robots could be roughly correlated to the excavation and tunnel propagation rates. A deposit is defined as an instance where a robot in a system delivers some cotton to the collection bin. Each successful deposit added from one to five cotton balls into the collection bin. Figure 52 shows the number of deposits made by the one, two, and three robot systems vs time. The figure shows that the speed of tunnel propagation is proportional to the number of robots in a system. Two robots propagate the tunnel faster than one robot and three robots propagate the tunnel faster than two robots.

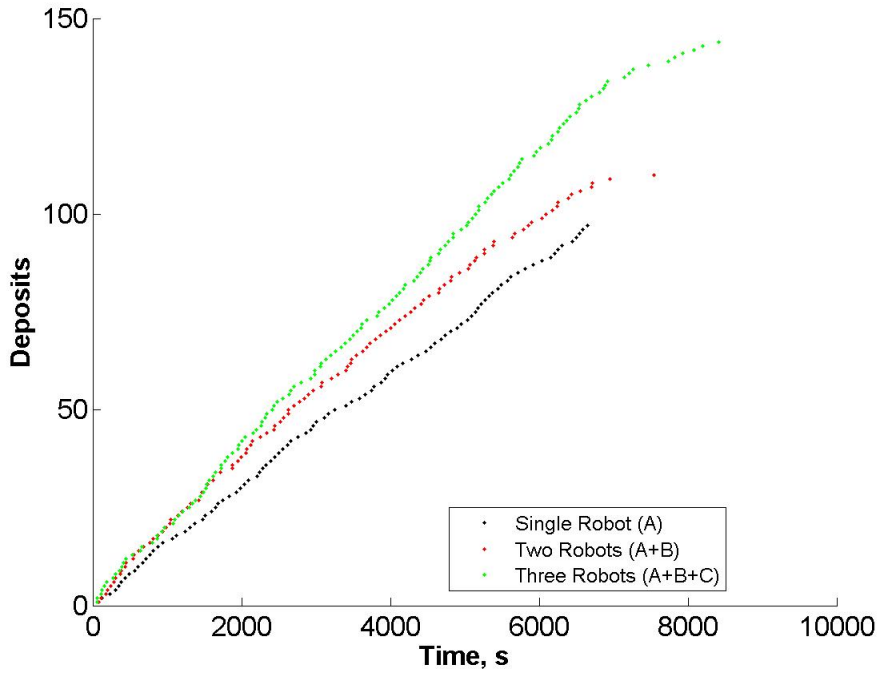


Figure 52: Deposits vs time. The plot shows that adding robots to the system increases the excavation rate

The main preliminary result is illustrated in Figure 53 which is a plot of energy vs the amount of deposits. The plot shows that increasing the number of robots in a system causes an increase in the energy cost to propagate the tunnel forward. The two robot system spent more energy than a one robot system to make the same number of deposits. Likewise, the three robot system spent more energy than a two robot system. The increase in the energy cost comes from the fact that multiple robots collide into each other and create traffic jams.

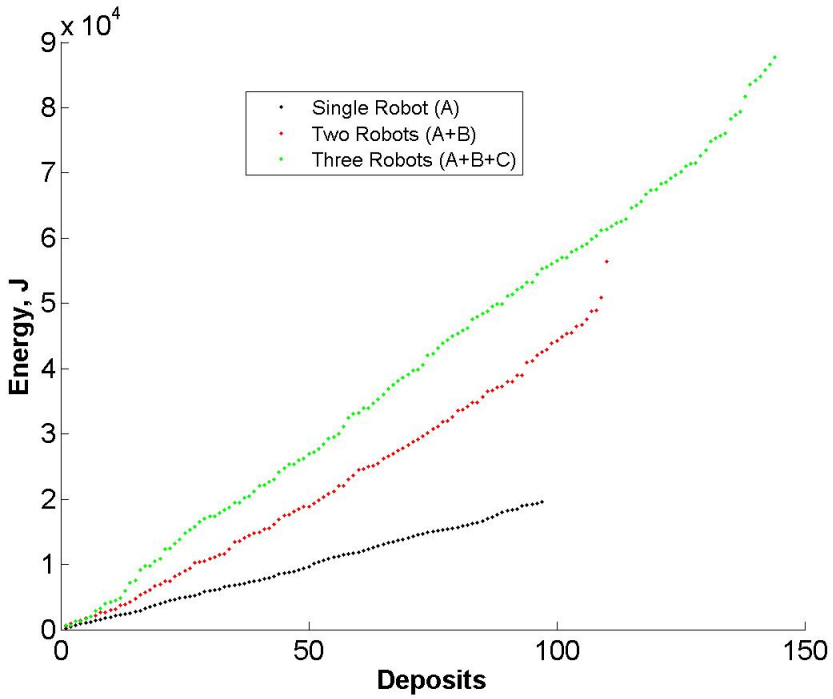


Figure 53: Energy per deposit. The plot shows that adding robots to the system increases the energy cost to complete the same amount of deposits

The rate of change of the energy with respect to time dE/dt, as well as the excavation rate dL/dt was computed. The excavation rate was considered to be a rate at which the number of deposits changes over time. The energy rate of change was obtained by fitting a $y = mx$ line through the data in Figure 51. The slope of the best fit line is the desired rate of change of energy. The excavation rate was obtained in a similar fashion by fitting a best fit line through the data in Figure 52. The results are shown in Figure 54. Figure 54 shows that adding a second robot to the system speeds up excavation by approximately 21%, and the energy consumption of the system goes up by approximately 144%. Adding an additional bumps up excavation up to 32% at the energy consumption to 275% compared to a one robot system. If we normalize the energy rate of the system by a number of the robots of the system, we also notice that robots spend about 25% more energy when digging

collectively comparing to digging alone.

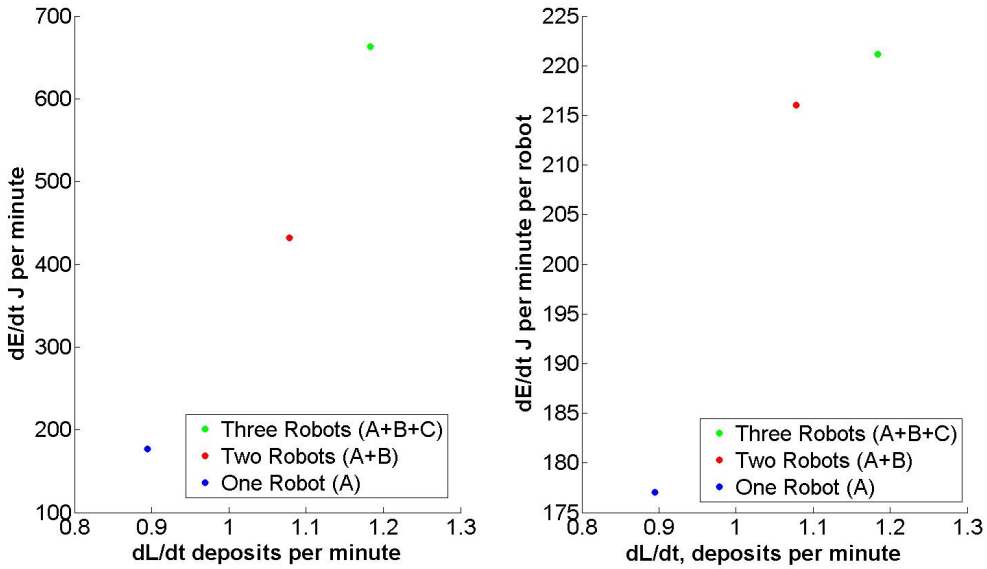


Figure 54: Energy and excavation rates of change. The plot shows that adding robots to the system increases both the excavation and the energy consumption rates

6.3 Conclusion and Future Work

This thesis achieved a design of a robotic ant which met all the set design constraints. There robots were constructed and tested in a tabletop test bed. In summary, we found that the robots have a fixed energy consumption rate. Most of the energy consumption comes from the fact that the robots are switched on and are running. Collisions do not cause a significant increase in the energy consumption. We also found that the excavation rate increased as we increased the number of robots. This was a benefit of a collective robotic excavation. The cost per deposit, however, also increased as the number of robots was increased. The jamming effect in the multi robot systems caused individual robots to take longer times to make the deposit. Thus, the robots spend more energy to make the deposits because the robots spend more time to make the deposit trips.

More experiments will be conducted with the one, two, and three robot systems.

The robot's behavior will be changed and the performance of the one, two, and three robot systems will be re-evaluated. The robots will be programmed to engage in the digging activity according to a specified probability (which would be obtained from the experiments with real ants). This scheme will force some of the robots to be lazy in a multi robot system. In doing so, we hope to observe a reduction in the the number of traffic jams and thus an increase in the excavation efficiency. The dimensions of the test bed could also be varied. For example, the width of the test bed could be reduced making it harder for the robots to avoid contact and jamming.

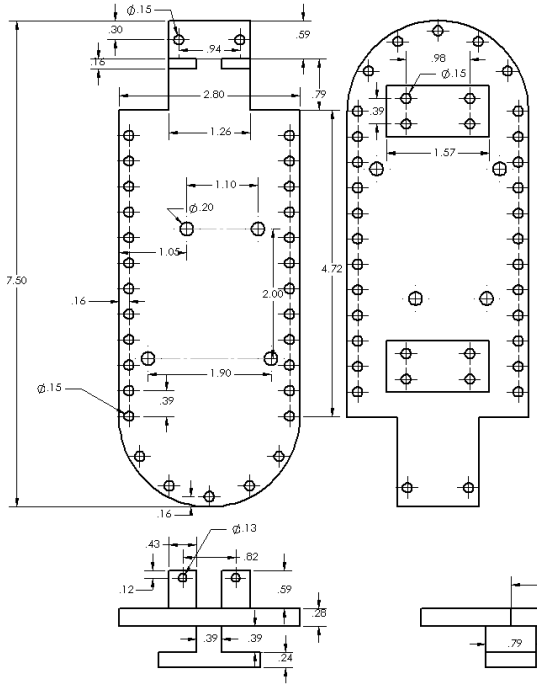
The robots could be improved in several ways. The Arduino Due could be replaced with a Linux based board, like a Raspberry Pi or BeagleBone to increase computational power. Doing so will pave the way for development of more complicated control algorithms making the robot more intelligent and perhaps robust. The Arduino Due still has plenty of unused RAM but additional RAM and flash storage and faster CPU would be useful in the future.

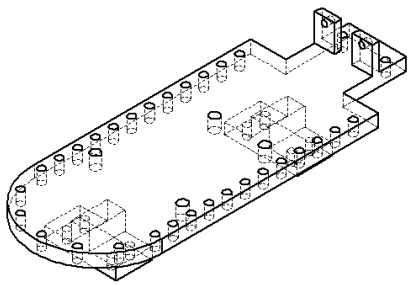
Improvements to the locomotion could be made. The cotton balls made the wheels jam occasionally. Several different wheels of different diameters and widths were tested. All wheels tested were found to be prone to jamming. This jamming problem could be addressed by installing stronger drive motors and redesigning power circuitry so that an adequate amount of current could be supplied to the motors. Alternatively, the wheels could be replaced with a platform involving legs.

Automation elements could be added to the test bed. For example, a system that would clean the test bed and repack the simulated granular media could be added. Programming robots to clean the test bed would be an ideal solution. Likewise, weight scales could be installed to monitor the amount of excavated simulated granular media over time. This would allow for a better measurement of excavation efficiency.

APPENDIX A

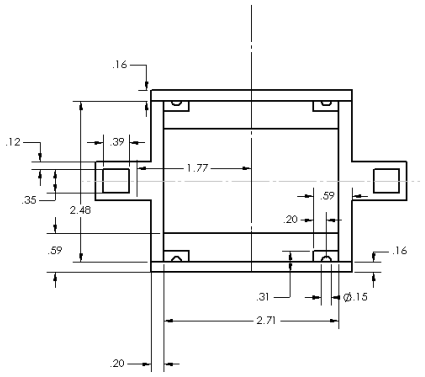
PARTS DRAWINGS

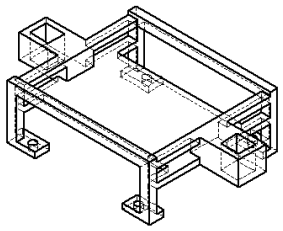


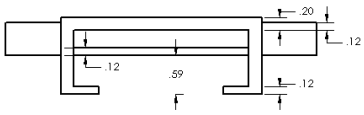


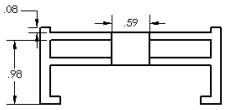
Main Body

all unites are in inches



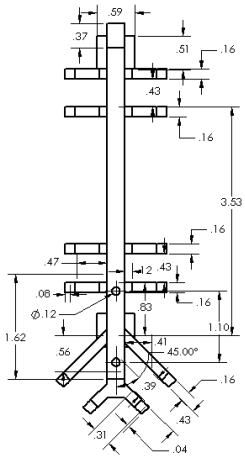


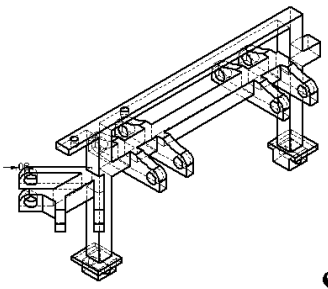




Circuit Board Holder

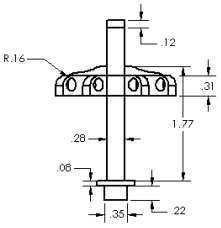
all unites are in inches

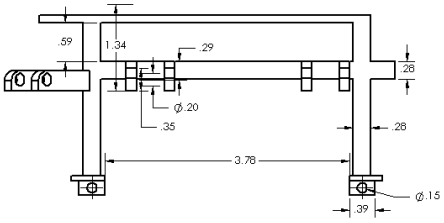


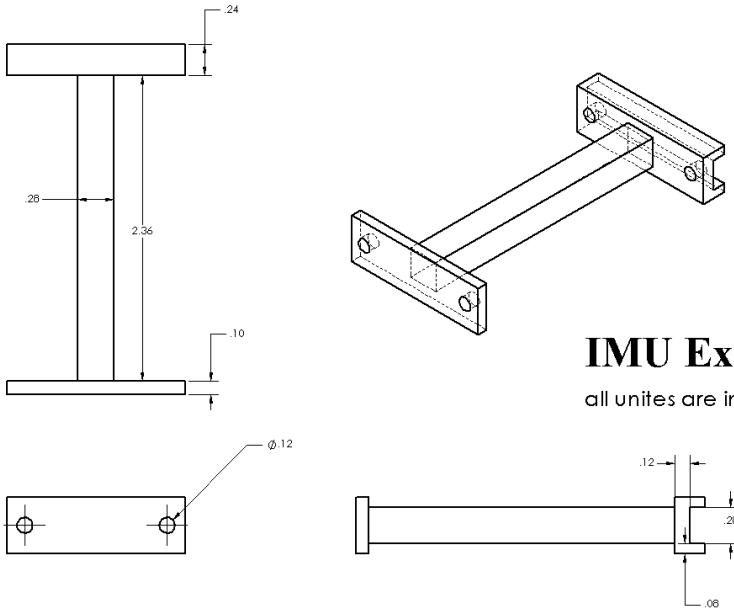


Shell Mount

all unites are in inches

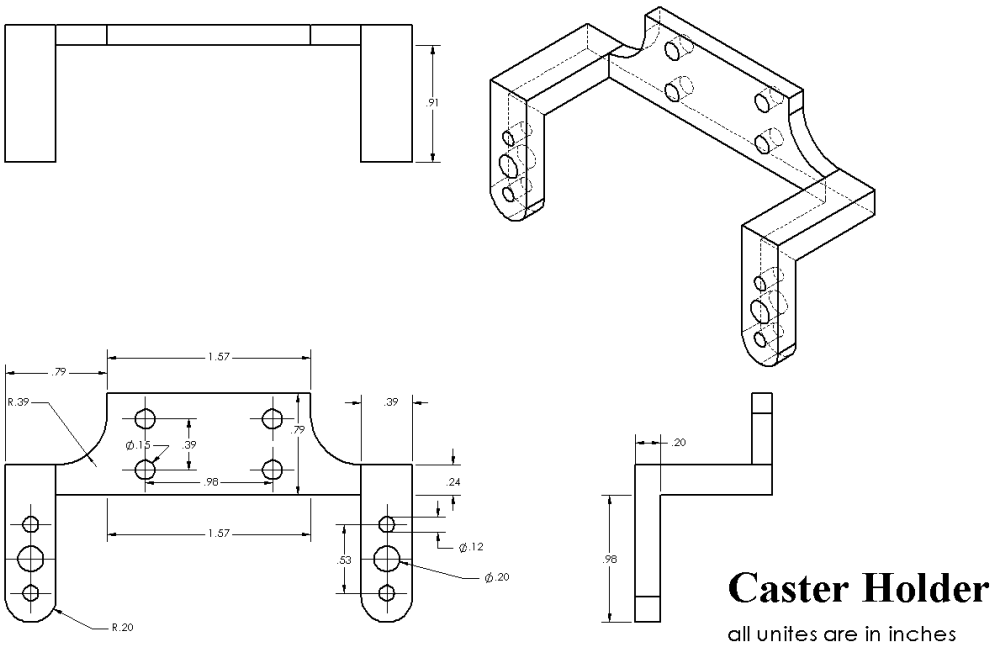






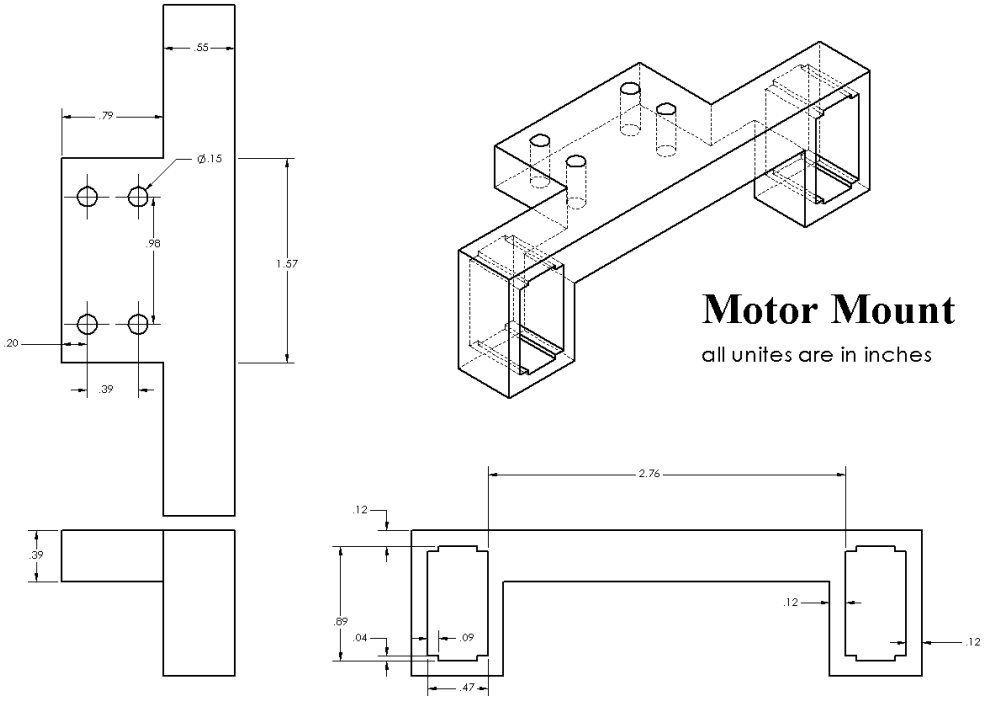
IMU Extension

all unites are in inches



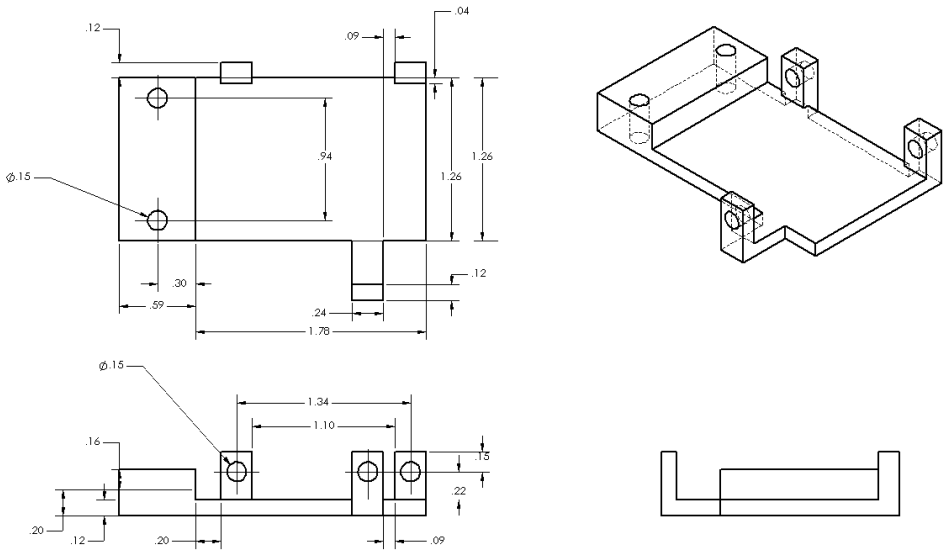
Caster Holder

all unites are in inches



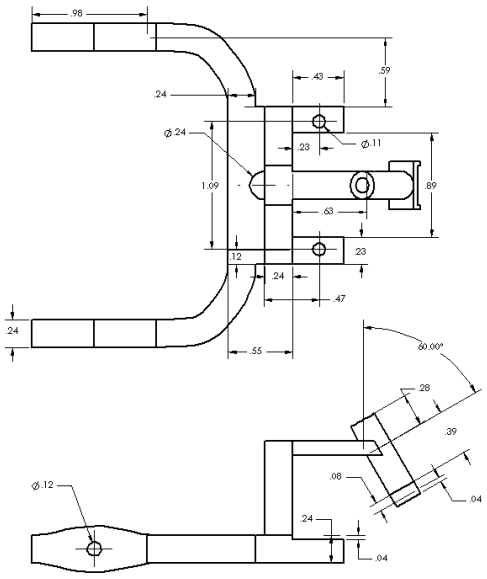
Motor Mount

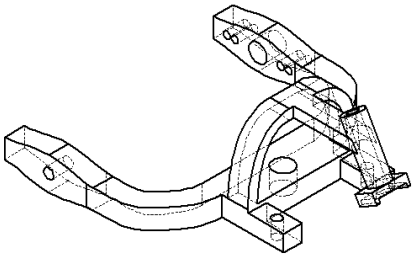
all unites are in inches



Arm Mount

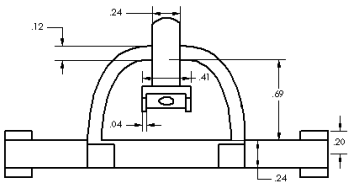
all unites are in inches

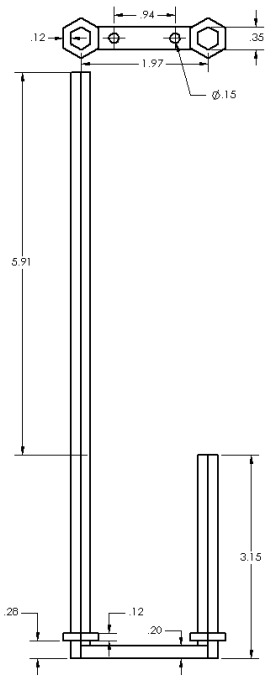


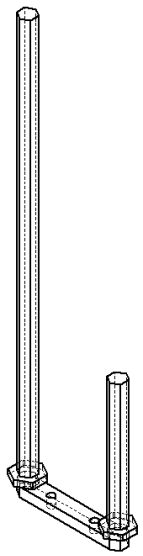


Gripper Holder

all unites are in inches

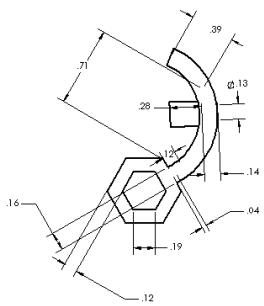


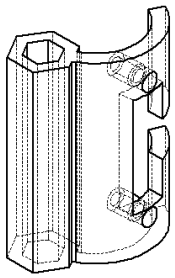


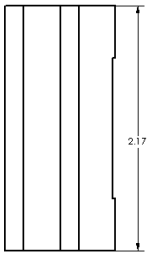


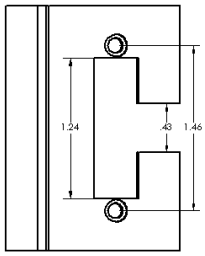
Hexagonal Pole

all unites are in inches



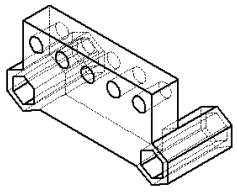


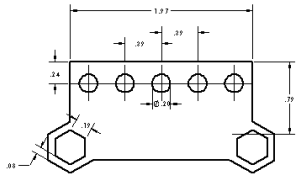


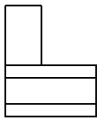


IR Holder

all unites are in inches

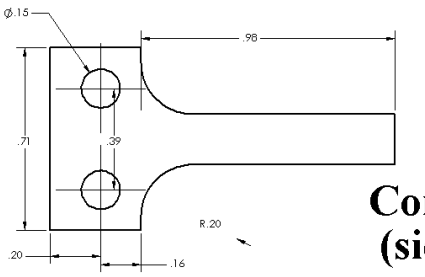


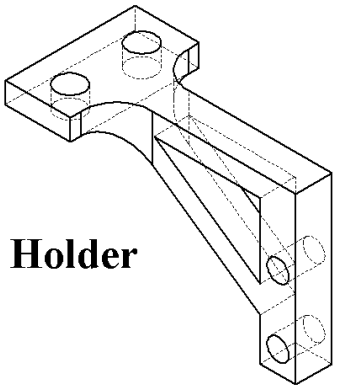




Switch Bar

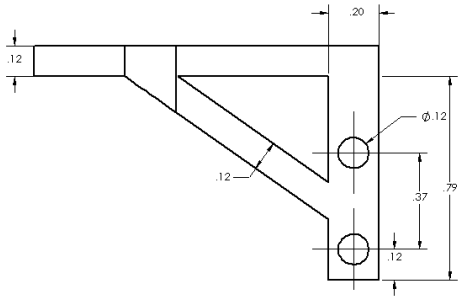
all unites are in inches

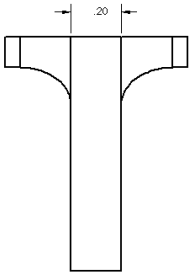


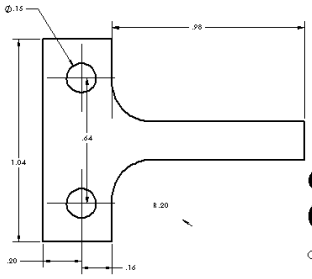


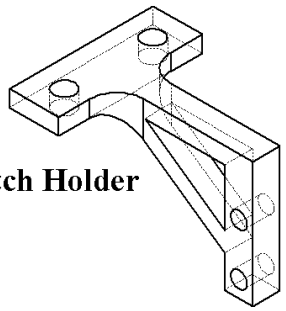
Contact Switch Holder (sides)

all unites are in inches



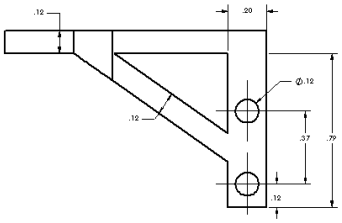


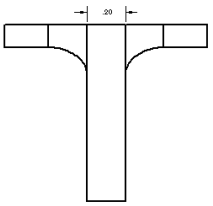


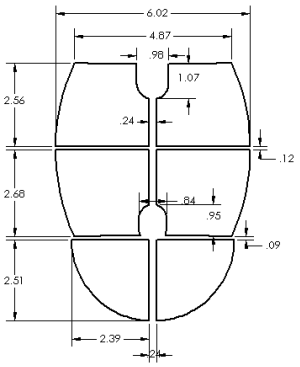


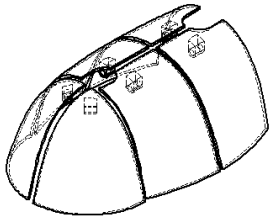
Contact Switch Holder (back)

all unites are in inches



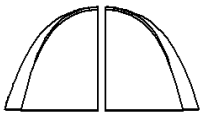


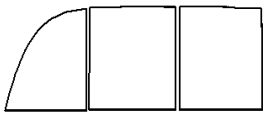




Outer Shell

all unites are in inches





APPENDIX B

MINIMUM PARTS LIST

Item	Vendor	Part #	Qnty	Price
Pixy Camera	Amazon	–	1	$ 69.00
IR Distance Sensors	Sparkfun	SEN-00242	2	$ 13.95
Analog Proximity Sensor	Sparkfun	ROB-09453	1	$ 2.95
Digital Proximity Sensor	Sparkfun	ROB-09454	1	$ 2.95
9 DOF IMU	Sparkfun	SEN-12636	1	$ 29.95
Mini Plastic Gearmotor, D-Shaft	Pololu	1511	2	$ 5.49
HS-55 Servo Motor	Servocity	31055S	1	$ 9.99
HS-A5076HB Servo Motor	Servocity	35076S	1	$ 23.99
Micro Gripper Kit	Servocity	637104	1	$ 6.99
3.3k Ohm Resistors (100 pack)	Amazon	–	1	$ 4.49
0.1μF Capacitors	Sparkfun	COM-08375	2	$ 0.25
470μF Capacitors (10 pack)	Amazon	–	2	$ 4.34
5V Step-Up Voltage Regulator	Pololu	2115	2	$ 3.95
Current Sensor	Pololu	1185	1	$ 9.95
Micro Contact Switches	Pololu	1402	8	$ 0.75
3400mAH 3.6V Li-On Battery, 2 Pack	Amazon	–	1	$ 26.95
LiPo Charger	Sparkfun	PRT-10401	1	$ 7.95
Dual Motor Driver Board	Pololu	2135	1	$ 4.49
Solid State Relay	DigiKey	DMO063	1	$ 21.87
Toggle Switch	Sparkfun	COM-09276	2	$ 1.95
Xbee Radios ZB Series 2 - 2mW	Adafruit	968	2	$ 22.95
Xbee Explorer	Sparkfun	WRL-11812	1	$ 24.95
Micro SD Card Reader	Sparkfun	BOB-1294	1	$ 9.95
Micro SD Card	Amazon	–	1	$ 4.95
Arduino FIO	Sparkfun	DEV-10116	1	$ 24.95
Arduino DUE	Adafruit	1076	1	$ 49.95
Small Wheels	Sparkfun	ROB-08899	1	$ 6.95
Metal Ball Caster 3/8”	Sparkfun	ROB-08909	2	$ 2.95
FTDI Basic Breakout, 3.3V	Sparkfun	DEV-09873	1	$ 14.95

The grand total is $475.83. Additional miscellaneous items such as perforated boards, hook up wires, connectors, cable ties, and fasteners may be useful.

REFERENCES

[1] "Animal Robotics, http://www.popsci.com/taxonomy/term/201054," (accessed March 27, 2015).

[2] "Archive for the "W. Grey Walter" category, http://cyberneticzoo.com/category/cyberneticanimals/grey-walter-cyberneticanimals/," (accessed March 28, 2015).

[3] Hamidreza Marvi, Chaohui Gong, Nick Gravish, Henry Astley, Matthew Travers, Ross L. Hatton, Joseph R. Mendelson III, Howie Choset, David L. Hu, Daniel I. Goldman , "Sidewinding with minimal slip: Snake and robot ascent of sandy slopes," *Science*, vol. 346, pp. 224–229, 2014.

[4] Nicole Mazouchova, Paul B Umbanhowar, and Daniel I. Goldman , "Flipper-driven terrestrial locomotion of a sea turtle-inspired robot," *Bioinspiration & Biomimetics*, vol. 8, 2013.

[5] "The Kilobot Project, http://www.eecs.harvard.edu/ssr/projects/progSA/kilobot.html," (accessed March 24, 2015).

[6] Justin Werfel, Kirstin Petersen, Radhika Nagpal, "Designing Collective Behavior in a Termite-Inspired Robot Construction Team," *Science*, vol. 343, pp. 754–758, 2014.

[7] Theraulaz, G., Bonabeau, E., and Deneubourg, J.-L., "The origin of nest complexity in social insects," *Complexity*, vol. 3, no. 6.

[8] GP Markin, J. O"Neal, J. Diller, "Foraging tunnels of the red imported fire ant, Solenopsis invicta (Hymenoptera: Formicidae)," *Journal of the Kansas Entomological Society*, vol. 48, pp. 83–89, 1975.

[9] Sudd, J. H., "The absence of social enhancement of digging in pairs of ants (Formica Lemani Bondroit) ," *Animal Behavior*, vol. 20, pp. 813–819, 1972.

[10] D. Charbonneau, N. Hillis, A. D., ""Lazy" in nature: ant colony time budgets show high "inactivity" in the field as well as in the lab", journal =,"

[11] Nick Gravish, Mateo Garcia, Nicole Mazouchova, Laura Levy, Paul B. Umbanhowar, Michael A. D. Goodisman, Daniel I. Goldman, "Effects of worker size on the dynamics of fire ant tunnel," *Journal of the royal society*, vol. 9, pp. 3312–3322, 2012.

[12] "Sugar cube-sized robotic ants mimic real foraging behavior, http://www.smithsonianmag.com/science-nature/sugar-cube-sized-robotic-ants-mimic-real-foraging-behavior-11201586/?no-ist," (accessed March 24, 2015).

[13] "Swarm Lab Robotics, http://www.theswarmlab.com/research/swarm-robotics," (accessed March 24, 2015).

[14] "The Ants: A Community of Microrobots, http://www.ai.mit.edu/projects/ants/index.html," (accessed March 24, 2015).

[15] "Arduino Mega, http://arduino.cc/en/Main/arduinoBoardMega2560," (accessed March 23, 2015).

[16] "Arduino Due, http://arduino.cc/en/Main/arduinoBoardDue," (accessed March 23, 2015).

[17] "BeagleBone, http://beagleboard.org/bone," (accessed March 23, 2015).

[18] "Raspberry Pi, http://www.raspberrypi.org," (accessed March 23, 2015).

[19] "Intel Edison, http://www.intel.com/content/www/us/en/do-it-yourself/edison.html," (accessed March 23, 2015).

[20] Tschinkel, W. R., *The Fire Ants*. Belknap Press, 2013.

[21] Daria Monaenkova, Nick Gravish, Greggory Rodriguez, Rachel Kutner, Michael A. D. Goodisman and Daniel I. Goldman, "Behavioral and mechanical determinants of collective subsurface nest excavation," *The Journal of Experimental Biology*, 2015 (in press).