Agricultural Swarm Robotics with Distributed Sensing

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree in Bachelor of Science
in
Robotics Engineering and Computer Science

Authors:

Nathaniel Jefferson

Peter Raspe

Nam Tran

Advised by:

Michael J. Ciaraldi

April 27, 2017

Abstract

The goal of the project is to design and build a swarm of robots that hold a symbiotic relationship. The purpose of this robotic system is to prove a concept of using symbiotic relationships to cut down on cost and increase flexibility of implementing large tasks carried out by robots. This was achieved by having multiple configurations of robots that had different responsibilities, one of which, the driver acted as the navigation and mobility of the platform, and the other, a sensory node which collects data and regulates climate. By using one to transport the others, a distributed system can be created through deploying a field of sensing robots to monitor areas of land using various sensors. The field of nodes would then form a mesh network to relay their sectors' information to a centralized server to display the data and potentially control the irrigation. The resultant project contained several nodes, a driver, and a working server which could act as previously stated, and publish data which would then be displayed as a number of useful graphics.

Contents

List of Figures

List of Tables

Introduction

The goal of this project was to create a system of symbiotic robots which collaborate to generate data. These robots are divided into two main groups, the driver and node robots. The driver robot uses GPS and vision technology to navigate and place the node robots into place. The node robots are stationary, and use a set of simple sensors that they all share to generate data metrics for geographic areas. They communicate and share information by forming a mesh network and sending packets of data to a server which visualizes and manages the data sets. Each node robot has a unique ID number that allows it to be identified and located by using GPS coordinates of their placement that are saved at the time of deployment. The server uses this data to help with robot maintenance and to form heat maps and other data visualizations that make the sensing data easier to view. By using this technology, farming and climate mapping would be able to process information about larger areas with less infrastructure and monitor crops and soil conditions remotely. The mesh network and GPS allowed the system to be easily modular and interchangeable, in addition to keeping maintenance simple.

1. Background

The main focus of this project is to develop a working system that would be considered part of the swarm robotics domain. We planned to achieve this by creating a symbiotic pair of swarm robots, that allow for large scale-ability. The relationship is based on having one expensive platform that can complete complex tasks such as navigation and depositing payloads, and a second robot that is more easily produced that does not have to do complex tasks. This allowed for the smaller robot to gather data on simple subjects frequently and create a network with its peers to send data to the server. The smaller robot communicates data in stages through its peers to create a data stream directly back to the navigation robot, which acts as a control tower, and sends the data to the server directly.

Using sensors in farming is a concept that has been explored previously and has shown moderate success. Universities and researchers alike have been using sensors for a variety of subjects, hoping to make a difference in farming technology. One such group of students at the University of Bonn is using a laser sensor to model the growth of plants by storing 3d models of the plants at all stages of their growth. By doing this, they hope to make reasonable estimates about growth rates and differences in growing styles in different areas of the field. This practice has given them moderate success, however they abandoned this to use sensors to classify the soil. This began as a result of the researchers realizing that their comparisons were effecting the results and they needed

to study the cause of the differing growth, not individual plants. Sensor data in farming has a variety of uses, but ideally its purpose should be to simplify the farming process.

Field data is the exact form of information that we, as a team worked towards finding. Most importantly, we are working towards an approach that eliminates some of the problems researchers face with current methods. With a swarm system and using robotics, data gathering will be much easier, as we will remove the need to run multiple sets of experiments on chunks of the field. Using a field full of simple and inexpensive sensors, you can achieve the same effect. In addition, by having a movable swarm that can communicate remotely, the situation can easily be modified pending changing circumstances, and data can be transmitted to different locations for further study. A swarm approach will allow for continuous time monitoring of the field, which will provide more data for farmers and scientists alike. In addition, the simplicity of an autonomous robot which can retrieve and place robots for you makes the system unobtrusive to systems currently in place.

1.1. introduction

Research was a key part of this project as it led us from our plethora of starting ideas to the finalized idea of the project. This journey of ideas took us through sets of driving small robots similar to the previous project but working with different systems of communication, collision avoidance, and styles of robotics. Our studies brought us into communication with many robotics engineers, who gave us ideas both good, strange, and impossible for the time constraints. One such source brainstormed with us about the idea of implementing distributed sensing and the team became captivated by the idea. With this change, focused research began in full, as we had precise topics for research

as opposed to the broad variety of swarm robotics type projects.

1.2. Swarm Robotics

Swarm Robotics is a growing field that is grouped into many sub categories. Robotics Engineers are adapting to the use of swarms and beginning to adjust to the different issues that swarm robotics faces [Ernst]. As such we took into consideration these same issues which became extremely valuable as the team worked through our own project. These issues are as follows, cost of individual robots, coordination and distribution algorithms, and the lack of reliable communications. These issues were important to our discussions and decisions on the pieces and systems we would use in our robots. Price would be important as it would limit our own testing but also the future potential of the project to have expensive nodes, as producing many expensive nodes would quickly add up and hinder the system. This informed our decisions to use smaller boards and simpler systems of sensing and communication as they would be the most inexpensive both in weight of the robot and cost of the robot pieces we would have. Coordination and distribution algorithms were a limited issue in this project as we conquered those with simple decisions. Nodes are stationary, which eliminates the need for individual GPS and tracking, as they could not collide without extreme weather conditions. With our current setup of a single driver robot, collisions between drivers were also a non-issue. Distribution of the nodes for our current setup was also eliminated quickly due to the small number of nodes we needed for testing. With only a few nodes, distribution was simple as we could just place them geometrically and not worry too much about their range for our simple tests and demonstrations. The final issue of communications was a major discussion point and informed our decisions on GPS, WiFi, and the use of the ZigBee Boards we eventually settled on.

1.3. Distributed Sensing

Distributed Sensing was not strictly the focus of our project, but was still a relevant research topic, as it was the most similar subject to the subject of our project. Distributed sensing is a field of study where similarly to the goal of our project, stationary sensors are used to monitor large areas using data manipulation to extrapolate results from the sensor findings. Our project differs from this field of study due to the ability of our sensors to be moved by the driver robot, and due to the density of their network coordinators. In most sensor distribution setups, there is a significant disparity between the number of sensors and coordinators of these sensors. In our case, it was a ratio of three to one sensors to Beaglebone Black boards, and each of them managed data into small packets. Data manipulation and study was carried out by our server code entirely, which limits the true computation of the individual sensor nodes, but also creates much more valuable data. One project at Carnegie Mellon University[Kantor, George, and Lea Cox] worked through a similar idea to our using exclusively distributed sensing methods. Their goal was to manage the irrigation systems of a farm using a wireless sensor network. Their sensors were connected in groups of single farms, with a rather low density of sensors. This allows a farmer in their situation to automate their irrigation of several farming spaces using a central server. However, their data is limited as their system automatically interpret the data into generalized information, which prevents the utility of making informed decisions about individual portions of the field. Their data is more useful for hydroponic or greenhouse situations where temperature, humidity, and sunlight are much more easily managed across a whole field as it is all contained in a controlled environment.

Another discussion we found was the use of distributed sensing in farming environments which are far from ideal. One project(El-kader, Sherine, and El-Basioni) in Egypt is working to use distributed sensing to maximize the profits of a new system of farming in the area. They were developing and using a similar system of sensor nodes and mesh networks in 2012. With a few major sensors: soil salinity, weather, humidity, and temperature, they hope to optimize their ability to keep plants healthy in larger fields, even given the difficulty of cultivating plants in the region. This was shown to be not only valuable due to the high profit levels that could result from farming in the region, but also due to the expanding value and growing industry that is sensor networks. As our system resembles this system, their decisions and mathematics are key to understanding how to market our system, and to what sorts of goals we should aspire for implementation.

2. Methodology

The team planned to have multiple phases in design and execution of the project. The team worked in a Agile development style, focusing on many pieces individually and working as a team when difficulties arose, and making larger decisions. The team started by working with simple testing and prototypes of systems to explore the most viable option for development in the project. After preliminary tests and review of specifications confirming which controllers would be suitable to use for the specific robot, design work began on the nodes. The initial task was designing the nodes: they are less manufacturing intensive and cheaper to develop; initial prototypes of its design were easy to finish quickly. Because the design was easy to construct, this allowed us to properly work out potential issues with the nodes before designing the driver. The driver's manipulation can then be built around the node allowing for their designs to better complement one another. When the physical design of the node was done, we then approached the software prototyping and creation of the server using Node.js. Using a centralized server to host data storage and visualization has the advantage of offloading the processing and storage needs of the nodes allowing for cheaper micro controllers to be used. A stationary server also operates as a central point separating communication from the operation of the placement system allowing for servicing without interrupting updates. Once individual robots had been designed and programmed, the task of configuring the nodes to operate as a mesh network became the focus. Specifically, the purpose of our mesh network was relaying data wirelessly and determining the best protocol to achieve

a mesh network. When the worker robot had its navigation functionality completed we moved towards testing the way drones were deposited and loading more into the driver autonomously. Once testing and connectivity was deemed usable we started the data collection process. We looked for a destination that the robots can be deployed on a small scale, and demonstrated a table-top mock-up for presentation purposes. Gathering successive data and completing data analysis of the robot is important to prove the potential and effectiveness of this type of design and the ability of our robot to work under differing circumstances and with different products.

2.1. Process

The goal and intentions of the project were the first tasks the team endeavored to complete. Initially, the project was focused on using echolocation technology with several small mobile robots to complete tasks. After some time cataloging the work of the previous iteration of the project, taking inventory, and some minor organizational goals, the team began to refactor the project. This started through some small initial discussions, where the team communicated a desire for a more broad project, and attempted to distance themselves, at least slightly, from the work of the previous team. This led to the change from the initial Raspberry Pi design to the Beaglebone centered design used in the this project's iterations. In a particular meeting, the current idea, of simple sensing robots working in tandem with a driving robot was discussed. Within this discussion, several ideas were discussed, which culminated in our final idea, using sensors for farm land management. Once this idea had been settled on, continued research on the topic and potential aids to the project intensified and the team began to discuss and work with the ideas discussed. A basic design was discussed, both for the nodes and driver robots, and the workspace became organized with KVM switches for the various

boards and a server computer for use with the nodes. Firstly the nodes were designed to be cube shaped, to make assembly and mounting of the boards easy. An threaded rod would run through the center like a pillar, keeping the node structurally rigid and providing simple mounting for a few key ideas. A handle to allow for easy pickup and movement of the nodes, which could be used in a variety of ways with the driver robot. Following this was the removable spike which could be used to keep the node in place on soft soil through inclement weather. The decision to move from the Raspberry Pi to the Beaglebone was settled on, and the team began purchasing and working with sensors created to work with a Beaglebone.

Use of the Beaglebone served multiple purposes. The first purpose was reduction of cost and size, as the Beaglebone and its associated cape mount were smaller by a few inches than any Raspberry Pi, in addition to being cheaper. This decision was key in allowing the node robots to become significantly smaller, a size shrink from a quite heavy six inch cube design to a lighter and more compact five inch cube. The reduction in size was made further possible due to the use of the Zigbee radio, which eliminated the use of WiFi on all nodes but the coordinator node. From that point on, the nodes were reduced to containing just a Beaglebone, the three sensors, a battery, and the Zigbee board. Coordinators had an additional USB WiFi card, creating a difference in weight of only several grams.

The driver robot was designed initially with the handle and spike that run through the center of nodes in mind. Several plans were made for a robot which could pick up and place nodes using the spike to keep their location and the handle for easy pickup. These handles mostly centered around a system for easy depositing and retrieval. However, the system was changed when a previous team's project became available to us. This team

had worked on vision tracking and picking up objects using OpenCV, an open source system for computer vision. This software allows the use of cameras to track colors and send further commands based off their pixel location in the resulting image. By properly calibrating this, you can use it as way-points or to get the driver to find nodes and get to the proper position to pick them up. This software worked with our system, as we could track using simple colors on the nodes. Building off the previous team's code and robot would save a significant amount of funds and time, as we could reuse code snippets and their designs. However, issues arose with their initial code, preventing its use, and the team completely rewrote the code for a similar purpose. The final driver robot uses their mechanism, and picks up nodes by tracking colors placed on the nodes' aforementioned handles (Figure 2.1). This was the most inexpensive solution, but also extremely effective, and had plenty of space for the addition of a GPS for the driver to publish data to the server on node location. The driver robot (Figure 2.2) became an easy addition to the project, and was the symbiote to the nodes that makes the project so compelling. With its completion in addition to the nodes and server, the project could attain the ease of use the team was seeking with the project.

2.2. Data Generation

Data management in this network happens primarily on the server, as data, once populated by the nodes, is immediately sent to the server in the form of JSON files. JSON files were chosen due to their nature as code objects. By using these files properly, the team could isolate data in a number of different ways, which would be useful for showing the different data from different sensors. The data could also be reformatted to be stored in these JSON objects sorted by date and time, object type, or which node had sent the data.

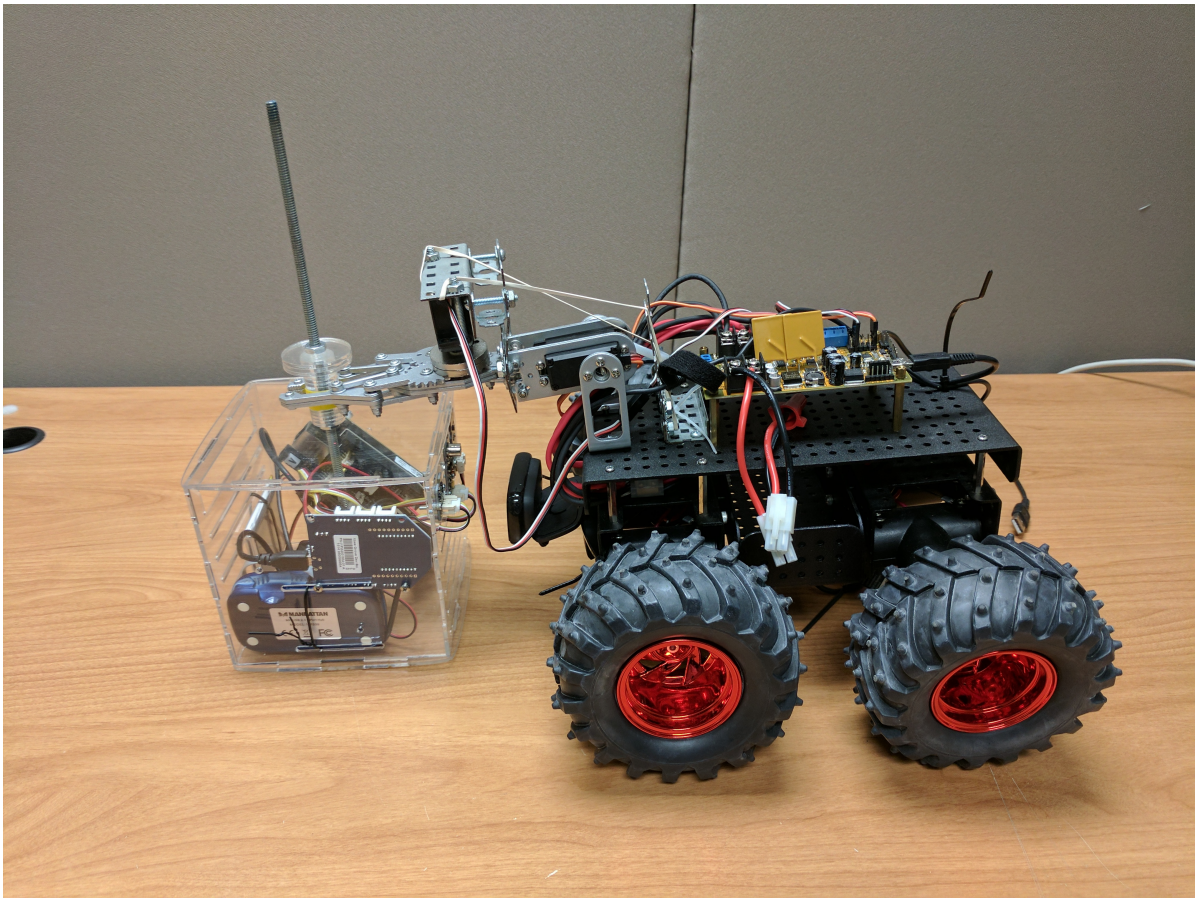

Figure 2.1.: Driver and Handle, Claw in Raised Position

To begin testing the website and work with our data science, the team developed code to generate synthetic data points. This would allow the data visualization and database to be tested even when the robots are off, or while at a distance. Initially the data was going to be sent several times a minute, but this was changed to six hour periods when the team started to get too much data on the web server which was making our data user interface run poorly. This period is easily changed but was maintained at six hours for a while to keep the website running smoothly for our times experimenting with the data.

Later data came from our running nodes and the sensors on them so that we could begin to experiment with different data visualization. Our three different sensors, temperature and humidity, air quality, and sunlight, would each generate different data points on the web server and would be populated to the JSON files. These could be visualized on the website in a number of different ways, however the team settled on two key methods. The first was a tree diagram that would allow users to access individual nodes and particular times to see the data. This is a quick and easy was to access the data with a few quick clicks and to see changes over days and months. The second method was a heat map which was displayed on top of a map of the node locations. The map of node locations was generated using Google's maps API and the stored latitude and longitude data from the GPS on the driver robot. Scaled to the proper size, a transparent heatmap would be placed in the same space to show values from the individual sensors. This would show the most recent data and would scale off of sets of data from the different sensor values, which the user could once again select to isolate individual data.

2.3. Mesh Network

Our mesh network is the efficient way our nodes communicate to the server without relying heavily on a overarching wireless Internet network. The team decided to use a Zigbee mesh network kit, as it fit our needs and would be small and cheap enough for the robots were were trying to design. Zigbee mesh networks work surrounding a coordinator, or single entity in the network which is the center of the grid. When the coordinator is removed, the network will fail unless another entity is declared the coordinator. Each entity in the network sends data through each-other to reach the coordinator, which could result in redundant data, so entities and coordinators are given unique identification numbers on initialization. As such, each system of nodes also now has a coordinator node, which is the center of the network, and it is the final node to see data before it is published on the server.

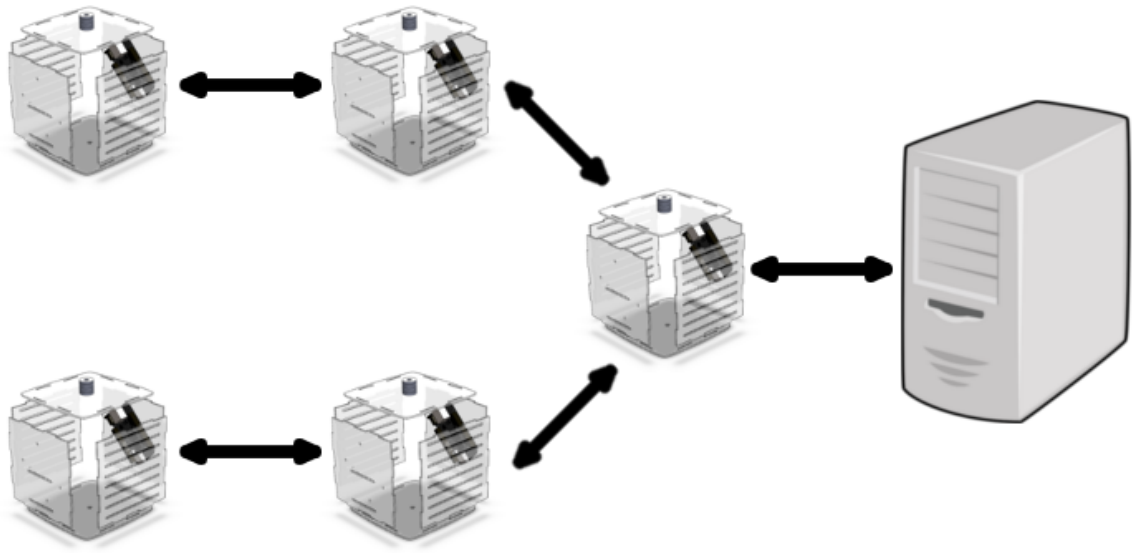

Figure 2.2.: Node Mesh Network

One issue with the Zigbee mesh network was the constant resetting of the unique

identification of the nodes, as it could cause issues with identifying individual nodes on the network and cause problems with the data. When Zigbee mesh networks are established, unique identities are given to each node or coordinator on the network. Should a node or the coordinator leave the network, it will lose the unique ID it was given and will not receive the same one the next time it connects. As such the unique ID's given by the Zigbee network are treated as separate data by our server and are stored in a temporary table in our database for later use, but deleted when the network goes down. They are also completely separate from our individual node unique identity numbers which we give to allow our server to identify the node who sends data.

3. Results

3.1. Nodes

At the finish of the project there are 3 nodes in various stages of completion. Their structure can be seen in Figure 3.1. These nodes are limited from completion in some forms by lack of availability of Grove sensors and capes, which would allow us to the use the available sensors on them. Each runs a variant of the modified Debian software the team wrote for compatibility with the software. The changes on this software are limited to Cron-Jobs, installation of some packages, libraries for the Grove sensors, Zigbee libraries and compatibility, and certificates to allow them to access the WPI WiFi network. Each has the same base software with minor changes to allow for their use as a set of nodes and coordinators (Figure 3.2). The three nodes, Node 0, Node 1, and Node 2, are all connected to the server sending data in several different ways due to their physical limitations. Node 1 is the coordinator and is discussed further in the section below. Node 0 is the first non-coordinator node to be completed, and has recently begun sending data, however it was previously sending inferred data due to the unavailability of Grove Capes over the latter course of the project. Capes for itself and Node 2, as well as additional sensors for Node 2 were ordered but had not yet shipped until very recently, when the only provider was restocked. Node 2 is in the same state as Node 0, and is also sending data, but lacks a Air Quality sensor, once again due to lack of stock from their provider. It was completed much later than Node 0 due to

updates to the design and our need to pull parts from it as replacement when the other nodes needed repairs. Regardless of its setbacks, it is also sending data and completely connected on the website.

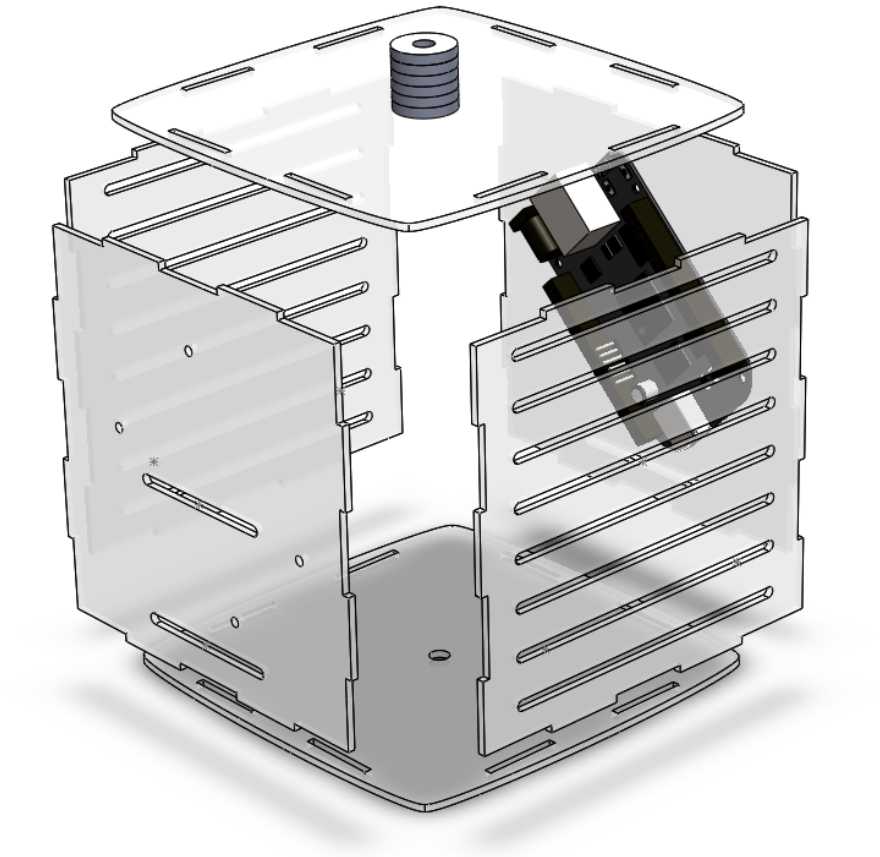

Figure 3.1.: Node Robot Enclosure

3.1.1. Coordinator

The coordinator node (Figure 3.2) was the first completed and fully implemented node, and has some minor software and hardware differences from the other nodes. For instance, its software runs to check for completed packages sent by the other nodes, as well as sending its own packages of data. Its Zigbee chip is initialized as the coordinator,

and it is thus the head of the network, as shown in Figure 2.2. It has a complete set of each sensor, Sunlight, Air Quality, and Temperature/Humidity and is enumerated as Node 1 on the server. It also has a large set of complete datasets from being initialized the longest on the server.

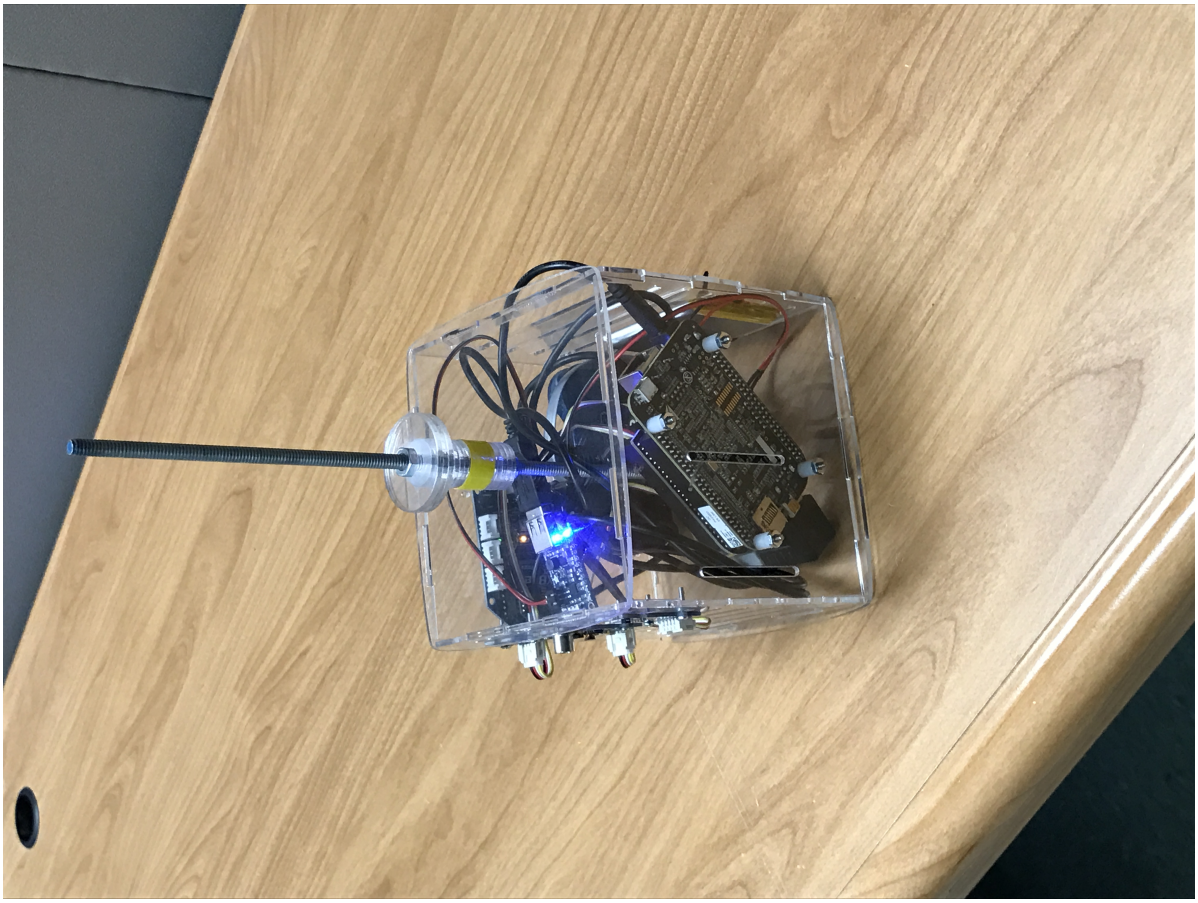

Figure 3.2.: Completed Coordinator Node

3.2. Driver

The driver robot, as mentioned earlier, was an acquisition from a previous project (Peng and Zhao), which was important for saving budget on the project. The driver robot is a Dagu Wild Thumper, using a TRex motor controller as a logic board, which complimented our goals of making the project easily purchasable and repeatable. The

previous team had written software and drivers for the robot which would allow it to use its camera to track color. Once color had been seen by the camera, the robot would navigate to a position where its claw could grab the object of that color and then return to a different color, which represented the location. With some simple testing of the arm and claw, we determined that the robot was capable of lifting and moving the nodes, and its durable and rugged design was ideal for the farm environment.

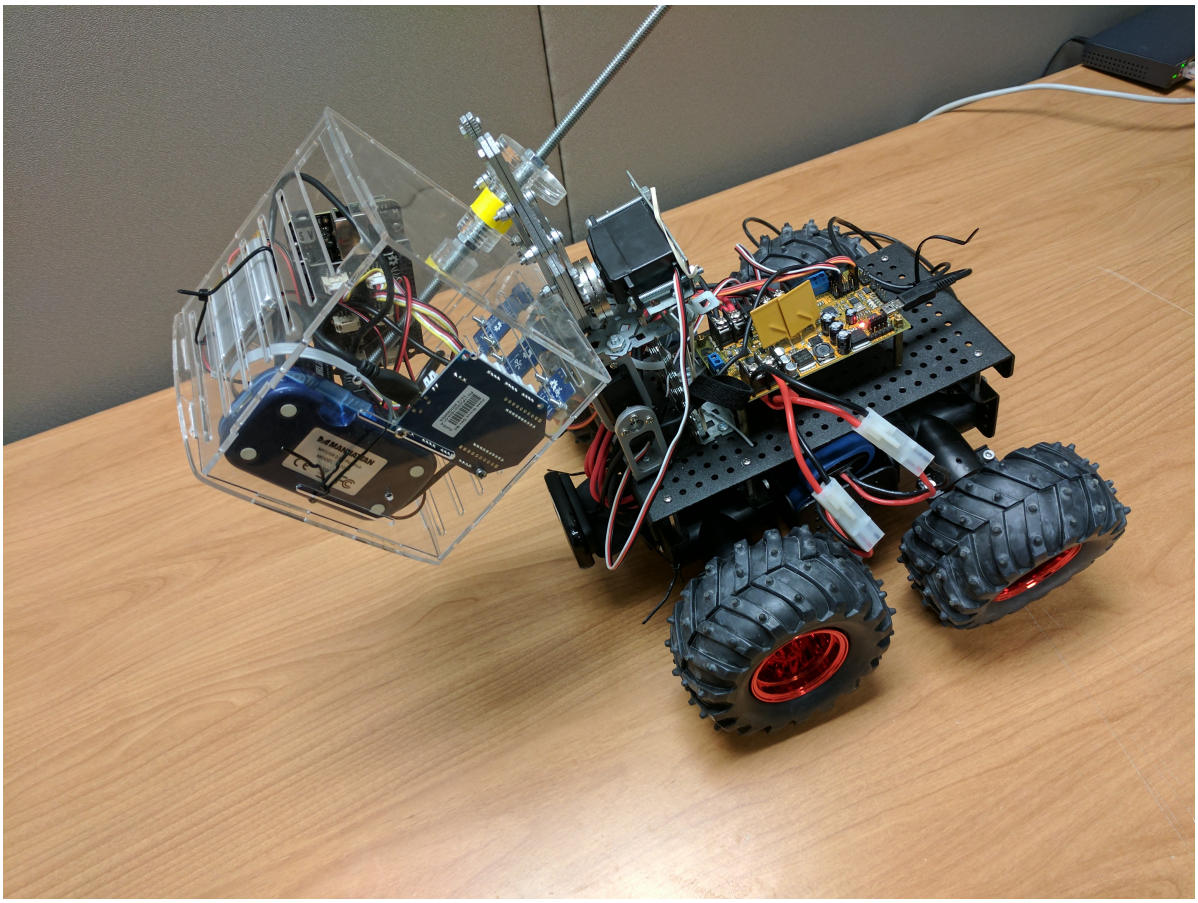

Figure 3.3.: Driver With Practice Handle, Claw Extended

3.2.1. Image Tracking

Although the hardware provided a good fit for this application, there was one major issue when re-purposing the Driver robot to pick up the nodes. Due to the implementation of the computer vision library the previous MQP team used; there were multiple obsta-

cles that arose. The main issue was the use of off-board computing to process the vision from the camera, which would then feed the output back to the driver's control board. The computing was done using a laptop that needed to follow behind the robot using long tether cables. To eliminate the need for tethering the driver robot, a Raspberry Pi was substituted for the trailing laptop. The change of hardware from a Windows laptop to a Raspberry Pi Running Linux led to the discovery that the OpenCV library version that was previously implemented was not supported on the new operating system. This library also required dependencies on specific Java libraries which were not upgraded to work with newer versions of the OpenCV library. This problem required a complete rewrite of the program controlling the vision tracking.

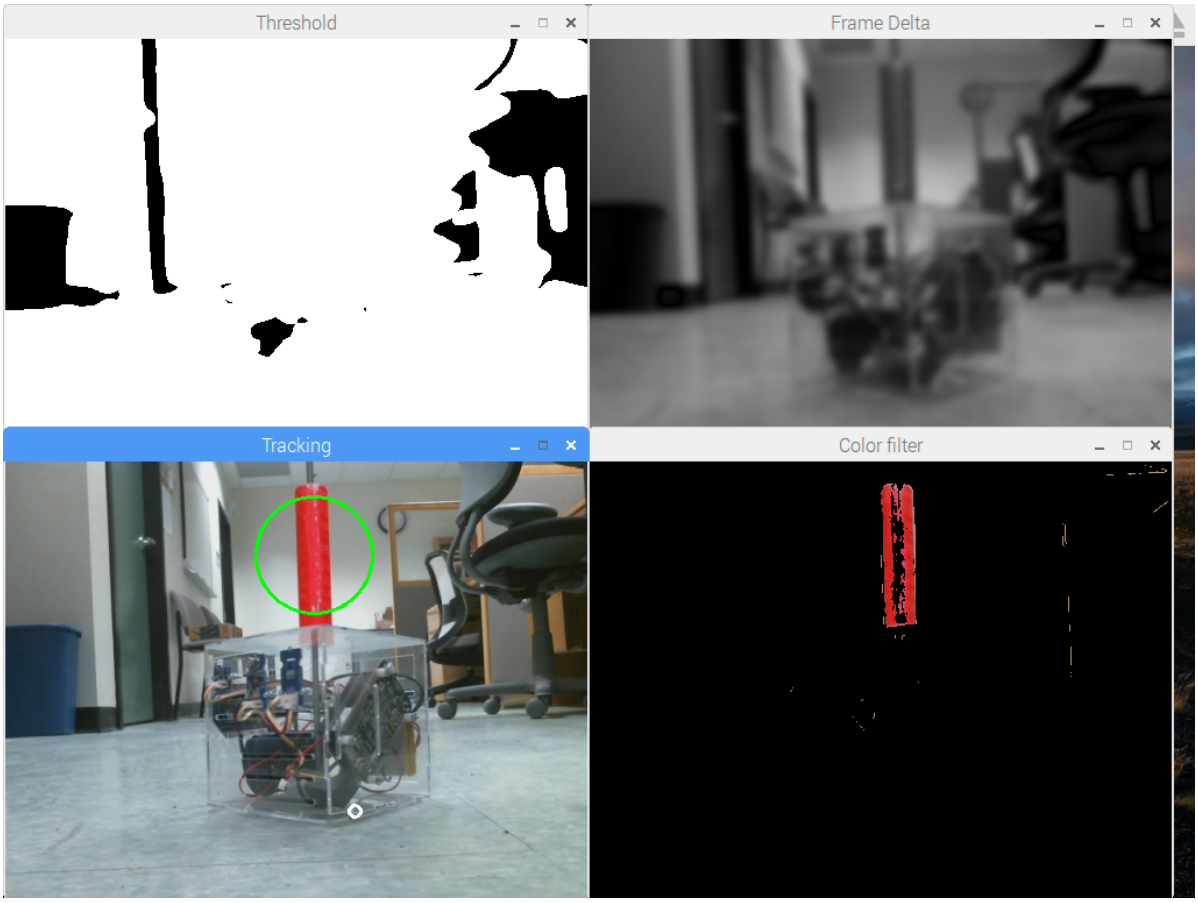

Figure 3.4.: Computer Vision Filters

To remedy this, new software was written in Python to work with newer versions of OpenCV on the newly installed Raspberry Pi. The Raspberry Pi was able to take the live image buffer from the Logitech webcam that was attached to the driver and apply different filters as seen in Figure 3.4. These filters allowed for masking the image to only search for the specific color marking the handle of the nodes. After the image was masked the pixels that passed over the threshold were then used to approximate the center of the handle. After the center of the handle was located the position in the field of view of the camera was used to determine the pathing and distance to the node from the front of the driver.

3.2.2. Controlling the Driver Robot

The software that was pre-existing on the TRex controller on the driver accepted characters over a USB serial connection to control the drive train and claw manipulator. These characters were easy to send from the Raspberry Pi and provided motions consistent enough to successfully ascertain the node when the driver was positioned 10+ feet away. The integration of a Raspberry Pi onto the driver robot also allowed for capes and hats to be used in conjunction with it. This additional interface gave room to mount a GPS chip onto the driver so that it could log the drop-off location of the nodes when they were to be deployed. Finally, the claw was raised to an ideal position so as to better move the nodes, and the camera was moved so that the claw would not block the vision. With these modifications created, the driver would now work with a simple mentality: drop the robots at given GPS locations, which would be stored in on the server with the nodes unique id for identification. Then to retrieve the nodes, the driver would return to the specific GPS coordinates to navigate to the approximate location of the node then use the camera to find and retrieve the node. This would allow users to control the location of nodes to within five meters, and would also limit their control

over the driver enough to make the system feel quick and seamless.

3.3. Server

The server became the most flexible portion of the system. The use of a computer science department donated desktop functioned as the main source of storage and interface for the coordinator to communicate with. The surface level of the server was to provide an interface to store data and display the information in a comprehensive manner. To continue with the theme of using the Node.js platform, the dedicated web server was running in a Rest configuration, servicing post requests from the nodes to receive the sensory data. The second function of the server was to display the data in a legible manner. This was initially accomplished by creating a tree that organized all of the readings by date, sensor, and time.

3.3.1. Data Persistence

Because the logging of the data was connected to a time stamp and on a standardized interval, it was necessary to have the data be persistent to maintain a working dataset. This was especially useful due to constant additions and restarts to the server. The finalized decision for the method of storing the data was multiple text files in JSON format. This was chosen over more longterm storage solutions such as databases because it would allow the data to be easily readable and present straightforward debugging. This plain text storage also allowed for testing and constructing data visualizations by generating mock data by hand.

3.3.2. Data Visualization

Key to the ease of use of the server data for intended users, the data visualization of the project is a constant work in progress, and should perpetually receive updates for improvement. The data was fashioned into a variety of formats, and is mostly using several libraries for easily creating data visualizations. The main functionality of the front-end the server provides is for displaying the data in formats that make sense to get a general understanding of the information that has been collected. The website that was created had a dashboard theme with 5 tabs each presenting information using different charts. The first tab, Home, includes simple graphs of the data as it is produced in real-time. These graphs are coded using the D3.js library and Socket.IO to update and pass information as it is produced.

The second tab, Node, allows the user to select a specific node and see a timestamped tree that expands into every reading the sensor has taken. This file interprets data from the persistent JSON file in the package and develops the chart based upon it. As the server is constantly updating the JSON file, this updates the tree whenever the data is changed. Users can use the drop down menu on the Node tab to choose individual nodes, and then by clicking through the circles on the tree(Figure 3.5), can be further limited to individual dates/times and the various data fields within each section. This was chosen for its quick and easy use to find data points that happened at a specific time as well as allowing for troubleshooting whether data was being transmitted from a node.

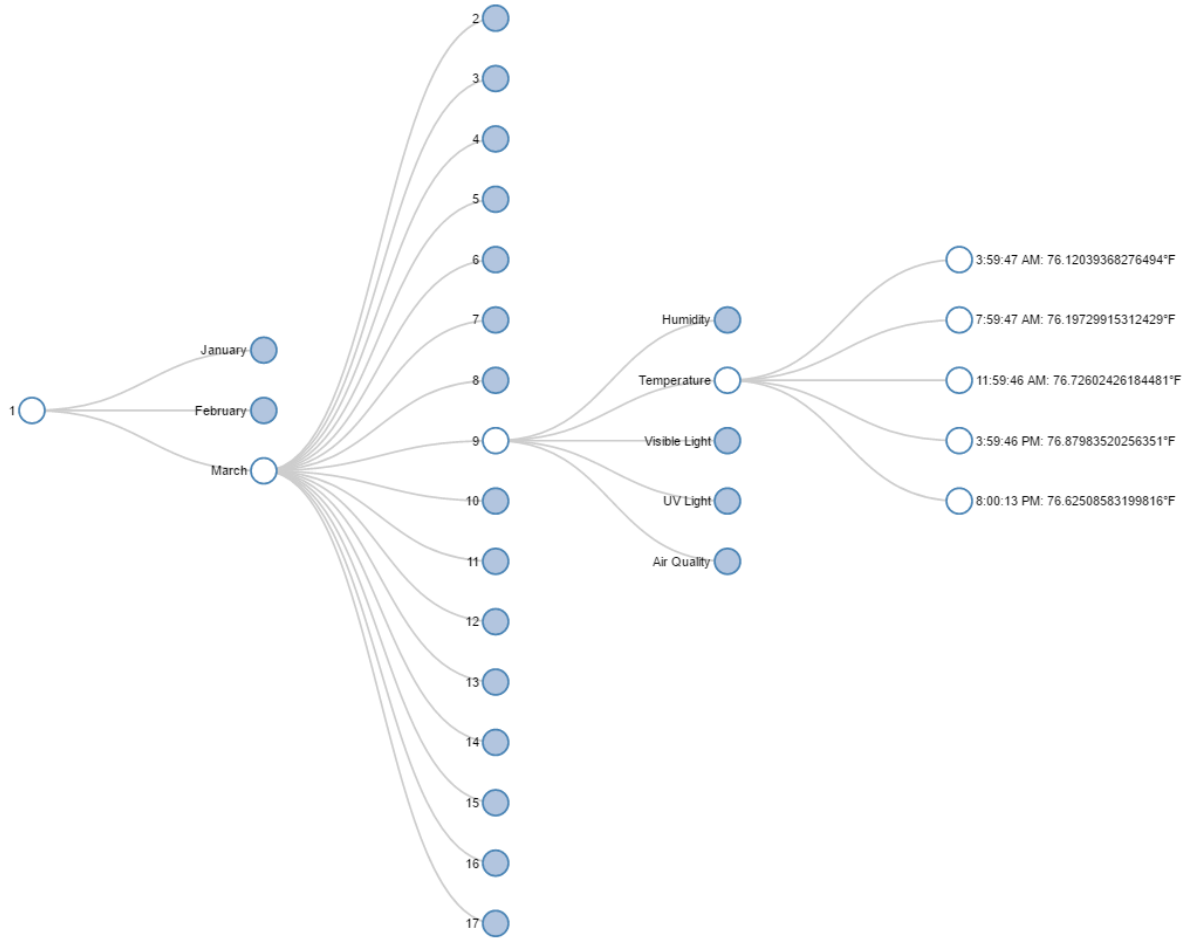

Figure 3.5.: Website Node Tree Data

The network tab was not completed, due to its designation as unnecessary until more nodes have been completed. For future iterations, a more complex map showing coordinators and nodes and their respective connections was planned. With the addition of this feature, the team believed it would be easier for users to understand issues that arise within the network. This visualization would also help users see which nodes are struggling to connect and find them to repair their issues.

The fourth tab on the website is the Maps tab, and is designed to show heat-maps overlain on a map of node locations. This tab uses a combination of two libraries for its

purposes, and has two scripts which produce images which overlay on each-other, with heat-maps transparent on top of the map. This visualization uses Google's maps API and has details for some calculations for distance to pixel density for heat-map use. Due to issues with this segment of code, the Maps tab is not currently running and is limited. The following layer is a heat-map, which uses a Node.JS module to generate the chart. The link can be found in the appendices and has information about installation and documentation. The heat-map uses calculations to convert the latitude and longitude values of each node into pixel density on the server to properly give x and y coordinates for the heat-map script to populate. Each point given an x and y value would then parse a value from the JSON data from individual nodes and a particular data type to create an intensity value for its point. This would allow the image to show the different values of the different nodes simultaneously on the map. This output image, called blob.PNG is placed into a transparent pane on top of the Google map image. An example heat-map image with a single data point is shown in Figure 3.6.

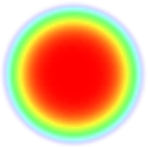

Figure 3.6.: Example populated heatmap

The final tab is the About tab. This tab contains a description of the project and explains what the data tabs are showing for those unacquainted with its use. The about tab is simple in construction, but needs to be updated with changes to the project and server website.

3.4. Communications

A large portion of this project was formatting multiple forms of communication to work in sync to pass information from one node across multiple others, eventually relaying the information to the server. The nodes all poses a ZigBee radio which possesses the ability for medium to long range communication based on the antenna being within line of sight of one another. The protocol for communication between the ZigBee radios comes with the product after minor configurations. Each of the standard nodes is configured as a router, with the ability to pass the information specifically generated by itself or allow other access points to relay information through it without affecting the nodes operation. The network is built into the radio interface, because of this the ZigBee is the only part of the node that needs to be on and powered for a node to act as a router. Thus the Beaglebone is able to go into low power mode when it is not in use and conserve on battery life.

The node can talk to the ZigBee unit by multiple forms of information based on configuration of the radios. There is an option to send checksummed hexadecimal packets, or plain text information. Again for ease of troubleshooting the plain text route was chosen with the option to convert to the checksummed container after more testing had been conducted. The plain text is passed along through sending characters through a serial connection from the Beaglebone to the ZigBee radio. The router configured nodes pass the information towards the coordinator. Upon the coordinator receiving the information, it waits until transmission is done and then formulates a request to the server to log the information.

The coordinator passes the information from the mesh network to the server through a WiFi connection. The coordinator node is the only node in the chain that cannot support being turned off when not in use. The coordinator constantly listens for serial communication to be received from the ZigBee radio. Upon the data coming in, the

coordinator processes the data into the same format request to submit to the server. Because the server is supporting a Node.JS web server, the method chosen to publish data using the WiFi connection was HTTP Post requests. This was a standard way of passing information and allowed for the server to properly respond and update based on new information being received.

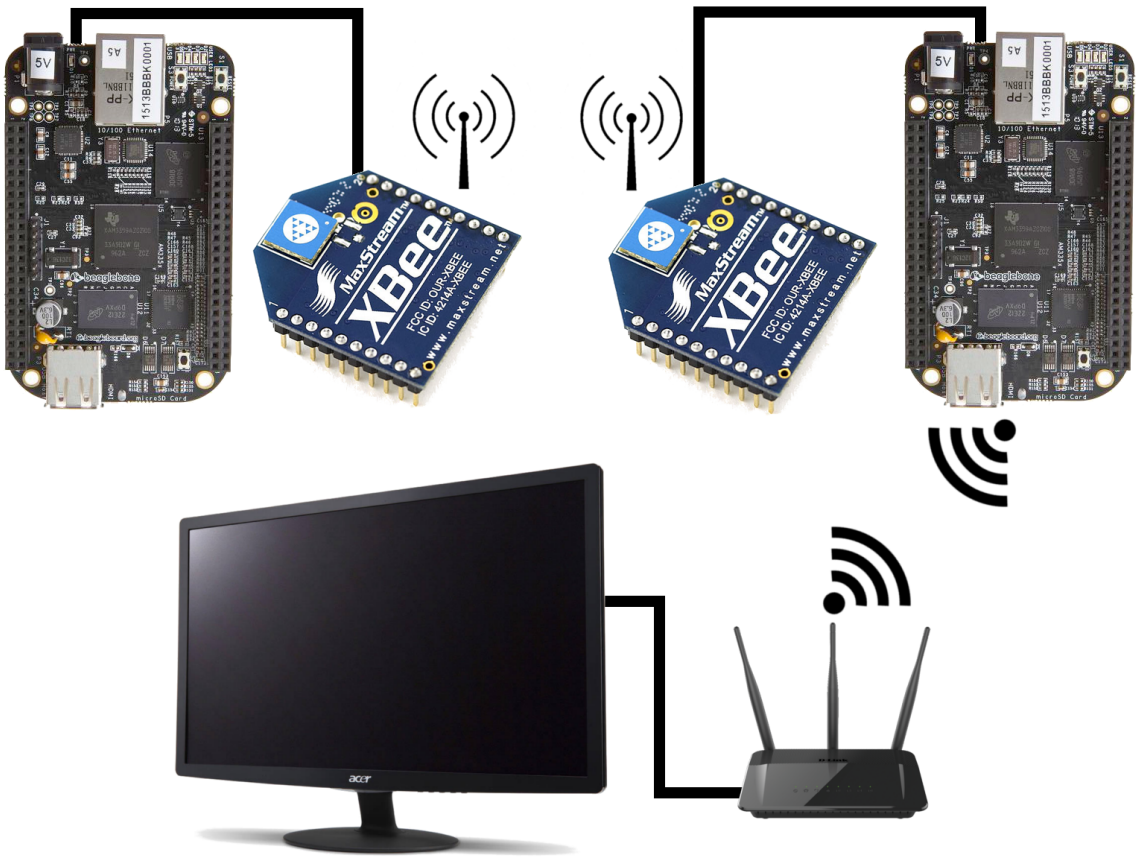

Figure 3.7.: Node Robot Enclosure

4. Discussion

4.1. Unfinished Projects

Though work on the project's base components has completed, the team continues to iterate and improve on several projects which were unable to be finished in the time frame. The first of these is the unfinished nodes, which are being further completed as the needed parts are shipped and is limited by their availability. This will be finished shortly, and has been difficult to overcome as key components such as Grove Capes for the BeagleBones were unavailable for the longest periods of time. In addition to the previously stated issue, the team continues to iterate on the website to add more useful data visualizations, complete the incomplete tabs, such as the Network and Maps tabs. This is not limited by physical issues, but can constantly iterate to further forms of completion as larger amounts of data and nodes are processed. Lastly, the Driver robot is in a state requiring repair, as constant use lead to a short circuit of part of the board. This needs analysis and then repair of the damaged parts. In addition, the driver robot is lacking in the fine tuning of the software. The driver needs more testing, which can optimize its algorithms and allow it to function properly even under less than ideal circumstances. All of these projects are key to the future success of the project, creation of more nodes, completion of the nodes which require work, repair and improvement of the driver, and constant update of the server website will give the future teams platforms with which to expand the project to levels which will allow it to move into use in its

desired environment.

4.2. Future Work

With the direction the team took the project, there are several further areas of intensive study which could benefit the completed product heavily. These are as follows: sensor layout and efficiency, weatherproofing and design, data and crop based analysis, and cost analysis and product trimming. With further research into these fields, this project could advance from its completed state to a marketable state, which could provide extreme value for the projects.

4.2.1. Sensor Layout and Sensor Efficiency

One major topic which was discussed but which could be further optimized is our use of sensors. Sensor range and potential value was considered, but limited by budget and size constraints. With more powerful sensors, and or research on their value and optimal use, each node could maximize its potential. This could further improve our software and change the driver robot software in order to place nodes in ranges which cover the most ground based on the sensor ranges. Sensors could also be placed in low power modes based on their needs, which could conserve power, make for more frequent results and allow for more sensors to be used. One major discussion was the use of solar panels, which could double as solar sensors with minimal changes. Simply attaching a voltmeter and an ammeter would give valuable information about the amount of sunlight plants are getting, and they could doubly be used to recharge the batteries of the nodes. Other senors of this type could be used for a long use version of the nodes, even potentially saving and storing so much energy that nodes could be used indeterminately, of course assuming conditions do not drastically change. Sensors also produce a variety of data which could be further implemented into the server visualizations and could make for

more powerful data analysis, which could be valuable in countless ways.

4.2.2. Weatherproofing and Design

One of the initial major discussions when working on the node design was on the subject of weatherproofing. Concerns arose regarding wind and softness of soil leading to nodes being easily misplaced, which could make it much harder to retrieve them, and could break parts of the nodes. In addition, rain and other treacherous weather could damage pieces of the nodes very easily without making the nodes water-resistant.The team decided that this was outside of initial project scope, however remnants of this discussion could lead to some significant follow up work. Weatherproof nodes would enable extended testing in farm and other environments without fear, and could allow for some interesting changes to the data collected and how nodes are used. With nodes that stay in position full time, solar energy and using the Beaglebone chips in constant sleep mode could make for significant power conservation and could allow the nodes to never need to change batteries or go in for a recharge. This bring up separate concerns, as the Zigbee radios would need some inherent understanding of which nodes are coordinators, so that as they will now reset semi-frequently, the mesh network could fail without its coordinator. In addition to weatherproofing the node case, designs also contained a potential spike and placement mechanism for keeping nodes placed in areas of high wind and soft soils. The spike was an initial system design, but other systems may serve to correct the same issue and could be valuable to the redesign and use of the nodes. Lastly, weatherproofing the driver is an additional project that could allow the driver robot to work in situations of less ideal weather, although extreme cases will probably always delay its use. This would prevent the driver system from running into issues with scheduling due to weather, which is a key concern should this project be commercialized or completely automated.

4.2.3. Data and Crop Analysis

Distributed Sensing is a deep field which the team was unfamiliar with at the start of the project, but the team has developed some understanding through the work of the project and some research. Distributed Sensing has been applied to farming before, and there is a wealth of information to be discovered from further study. One particularly valuable subject would be the study of how to properly analyze the data and layout of nodes. Nodes cover a particular radius with their sensors and could be more valuable should their layout be optimized. If their are covering the maximized area of space, increasing numbers of data points could increase the difficulty of interpreting and working with the data, which distributed sensing could improve through its algorithms and methods. In addition, different sets of crops have different needs and densities with with they can be planted. With a better understanding of this and study of the data, the nodes could be used in lesser or greater density depending on the needs of the plants. It could also improve the value of the nodes by changing the amount of sensors to match only sensors that the crops require. This could reduce the cost of individual nodes for different fields of situations, and change firmware for different nodes so that they have different understanding of their sensors and use. Study of this subject would require heavy research and testing, in addition to a far larger system than could be completed with the current budget, but would help the product provide a far larger and more valuable service.

4.2.4. Cost Analysis and Product Trimming

Shown in the appendices is a table containing our purchases and materials for the nodes we constructed and worked with. While not everything there is used in the final project, the materials will certainly be useful to future teams or for future iterations of the project. One major change that future teams should make at seeing our finalized

node robots and driver is to minimize the costs further. For example, use of additional right-angle connectors for the ports could shrink the nodes further. A big change that was discussed but impossible to achieve was use of the Raspberry Pi Zero board in replacement of the BeagleBone Black, as it minimizes space and cost quite effectively at $10. Working with this board would require additional wiring for the Grove sensors and changes to design, however due to the potential to make nodes cost even less, would be beneficial to the project overall. It lacks some of the useful features that were key to the decision to use the BeagleBone, such as extra breakout pins, but with some breakout boards could be a reduction of size and cost, at the expense of addition time requirements. Processes and discussions such as these could trim the product down to cheaper and smaller pieces, which could contribute to greater sensor density, ease of use, and potential to create far more nodes for future testing.

5. Conclusion

As a team, we feel this project was a successful and eventful process, which culminated in a compelling project which has huge potential value. Its current state is far from perfect, with many changes which could be made to benefit or improve it. Yet its development and decisions required heavy thought and will certainly be further discussed by other groups in similar projects or that continue our work. Finished nodes, website and driver robot will undoubtedly change, excess pieces and unused materials may be used later on, however decisions made and the project designed by this team will certainly stay as a valuable contribution for WPI and for society.

6. Citations

Ernst , Jason. "4 Challenges Holding Back Swarm Robotics." Robotics Trends. 2017 Robotics Trends A Division of EH Publishing, Inc. d.b.a EH Media. All Rights Reserved., 28 Sept. 2015. Web. 17 Feb. 2017.

Kantor, George, and John Lea-Cox. "CMU Presentation: Distributed Sensing for Crops." Specialty Crops Workshop. N.p., n.d. Web. 19 Feb. 2017.

Rashvand, Habib F., and Alcaraz Calero Jose M. Distributed sensor systems practice and applications. Chichester, West Sussex: Wiley, 2012. Print.

Abd El-kader, Sherine M., and Basha M. Mohammed El-Basioni. "Precision farming solution in Egypt using the wireless sensor network technology." Precision farming solution in Egypt using the wireless sensor network technology. N.p., n.d. Web. 16 Jan. 2017.

http://wiki.seeed.cc/Sensor/ Team, Seeedstudio. "Sensor Introduction." Seeed Wiki. N.p., n.d. Web. 20 Mar. 2017.

S. (2013, April 11). Substack/node-heatmap. Retrieved February 11, 2017, from https://github.com/substack/node-heatmap

Ren, Peng, and Tianci Zhao. "Intelligent Vision-Driven Robot For Sample Detection And Return." Worcester Polytechnic Institute. Michael Ciaraldi, 30 Apr. 2014. Web. 10 Jan. 2017.

"MESH." Xbee - MESH. N.p., n.d. Web. 27 Feb. 2017.

"Basic motion detection and tracking with Python and OpenCV." PyImageSearch. N.p., 03 Feb. 2017. Web. 27 Apr. 2017.

Bostock, MIke. "Collapsible Tree." Popular Blocks. N.p., n.d. Web. 27 Apr. 2017.

Appendices

A. Materials

A.0.1. Purchases

Item	Quantity	Price per unit	Total Price
3/16 Acrylic Sheet	10	$20.00	$200.00
BeagleBone Black	3	$40.00	$120.00
Server	1	$0.00	$0.00
Mini-Hdmi to HDMI cable	3	$5.29	$15.87
KVM	1	$22.99	$22.99
Grove Cape for BeagleBone Black	3	$20.00	$60.00
Grove Air Quality Sensor	2	$8.00	$16.00
Grove Sunlight Sensor	3	$9.00	$9.00
Grove Temperature and Humidity Sensor	3	$7.00	$14.00
Raspberry Pi GPS Hat	1	$80.00	$80.00
Vex Smart Charger V2	1	$16.99	$16.99
Grove Connector Wires	2	$2.75	$5.50
Zigbee Mesh Network Kit	1	$89.00	$89.00

Figure A.1.: Bill of materials