Vision Controlled Autonomous Stakebot for Driving Stakes and Its Digital-Twin

Corey Lee Leydig

VISION CONTROLLED AUTONOMOUS STAKEBOT FOR DRIVING STAKES AND ITS DIGITAL-TWIN

by

Corey Lee Leydig

Bachelor of Science
University of South Carolina, 2021

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Mechanical Engineering

College of Engineering and Computing

University of South Carolina

2023

Accepted by:

Sourav Banerjee, Director of Thesis

Travis Knight, Reader

Nikos Vitzilaios, Reader

Cheryl L. Addy, Interim Vice Provost and Dean of the Graduate School

ACKNOWLEDGEMENTS

I would like to acknowledge Titan Farms for allowing the use of their fields for experiments and providing their knowledge on their planting process.

ABSTRACT

In this thesis a novel vision-based AI driven autonomous ‘StakeBot’ is proposed to serve the agricultural 4.0 industries of the future. In recent years supply, demand and production of vegetables that are harvested from plants with weak stems like bell pepper, tomato, eggplant has been significantly up by several fold. With growing demand and new green-house establishments across the country production of vegetables will require a tremendous number of labors which will be hard to supply soon. To overcome the issues in addition to the labor shortages, automation through robotics will be the only viable solution. Plants like bell peppers require support to avoid contamination of the pepper from touching the ground. Thus, a workforce is needed to place these stakes on several acres of land. Alternatively, as adopted in greenhouses across the world, the stakes are placed at a longer distance and rope or wires are tied between them to create continuous support, which makes the plants grow even taller. This is indeed labor-intensive work. To overcome the labor challenges in this thesis, an AI driven robotic solution is proposed that is capable of placing and removing stakes in the ground. With over 210,000 jobs, agribusiness is South Carolina’s No. 1 industry and will be immediately benefited by the proposed solution. The proposed robot is called the StakeBot. The StakeBot is an autonomous self-driving robot which meets the requirements provided by the local farmers from South Carolina. The uniqueness of the StakeBot is its self-driving capabilities which are made possible with a newly developed vision system capable of successfully driving the StakeBot. This vision system utilizes Stereo depth technology paired with the capability of modern Linux based single board computers. This grants the StakeBot the ability to “see” making it capable of avoiding obstacles and making decisions on how to proceed based on the environment. The StakeBot’s vision system is not unique to the StakeBot alone. The developed algorithms can be intergraded for any robotic system that is designed for navigating through its environment. The vision system has shown great success with an average of about 15 FPS, capable of fast response times for vehicles moving at speeds less than 7 MPH.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CPU	Central Processing Unit
DAQ	Data Acquisition System
DC	Direct Current
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GF	Glass-Filled
GPS	Global Positioning System
IoT	Internet of Things
ISO	Isometric View
lbs.	Pounds
lbs./in^3	Pounds Per Cubic Inch
LiDAR	Light Detection and Ranging
MPH	Miles Per Hour
MQTT	Message Queuing Telemetry Transport
psi.	Pounds Per Square inch
PVC	Polyvinyl Chloride
RGB	Red, Green, Blue
WIFI	Wireless Fidelity
YOLO	You Only Look Once

CHAPTER 1:

INTRODUCTION TO THE STAKEBOT

1.1 Motivation behind the StakeBot:

With the constant improvements and newly refined developments in robotics steadily rising, many different companies have been able to provide better and more improved sensors and computers being more portable and more capable than ever before. As the use of robotics becomes more susceptible for a larger variety of tasks within various industries, the desire to turn towards automation has become a great alternative to traditional means of labor. For the agricultural industry, automation has been intergraded for various means with an attempt to keep production value high and for better quality control on their products. Typically, the use of an autonomous vehicle, however, is not as prominent mostly due to the technology being relatively new and that for some farmers, it goes against some traditions by eliminating the human touch element. As true with larger agricultural companies, the issue with the evolving world is the lack of sustaining a work force not because of financial endeavors but simply due to the lack of interest within many Americans today. Farming involves very labor-intensive work, and most companies are struggling to keep a large enough work force capable of keeping up with their high demands for produce and other crops. For a company to stay successful, they must adapt and overcome these struggles and the turn to robotics is the motivation behind the development of an early-stage robot capable of reducing the need for a large labor force while not compromising on the demand of the product being supplied.

In order to understand the market, research was conducted by reaching out to some of the local farmers here in the state of South Carolina. With the help of the South Carolina Department of Agricultural, (SCDA) some of the larger agricultural companies were able to express their current struggles relating to the lack of a work force. South Carolina is home to many farmers, large and small, and the majority of these companies expressed a need for automation within their planting process. For many farmers, the purpose of a work force is to ensure the success of the planting process.

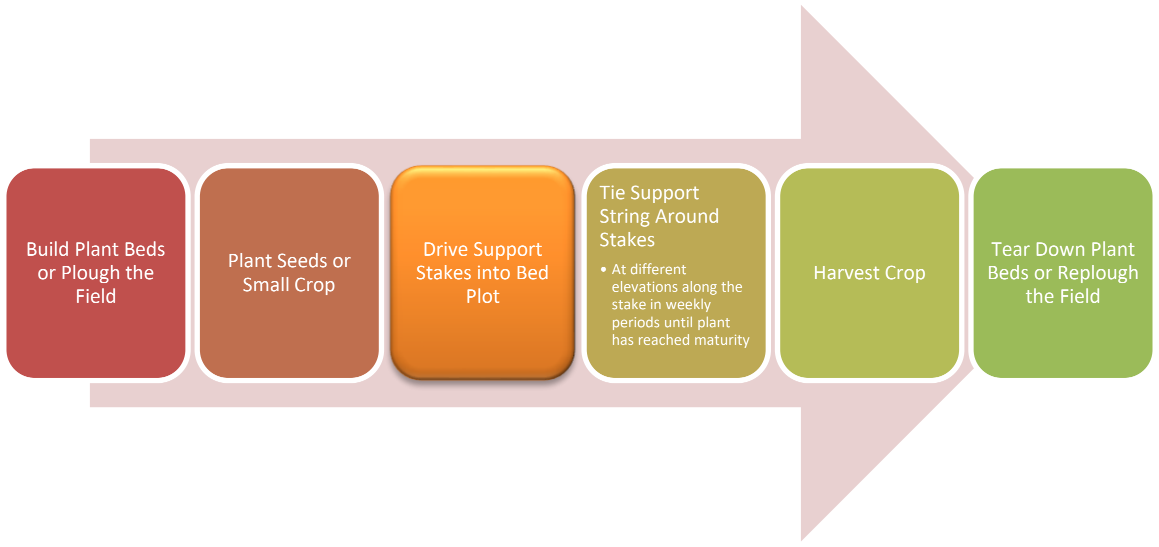

Build Plant Beds or Plough the Field
Plant Seeds or Small Crop
Drive Support Stakes into Bed Plot
Tie Support String Around Stakes
• At different elevations along the stake in weekly periods until plant has reached maturity
Harvest Crop
Tear Down Plant Beds or Replough the Field

Figure 1.1: Typical planting process for majority of crops in the agricultural industry.

Shown in Figure 1.1, the planting process is a simplified way to describe the daily tasks involving the methods of maintaining and harvesting healthy crops. This process includes many different steps that require a lot of attention from the workers. For the early stages of introducing automation into their planting process, the task that was chosen was simplified into one part of the planting process. The specific process that became the main focus was the automation of driving support stakes into the plant beds. For most plants, within their early stages of life, as they begin to grow and mature, they require extra support as the stems of the plant are not yet strong enough to support themself during these few weeks. This can cause the plant to either grow outward, and towards the aisle, where it will be in a location not suitable for healthy plant growth, or the stem could break causing the plant to die. This problem is common enough that the proper way to ensure healthy plant growth requires the use of support stakes to be driven manually by a worker in the plot next to the plant where the plant can safely grow upward and with proper support until the plant matures. For many of the farmers in South Carolina, the most common design of the bed plots is raised off the ground and constructed by other forms of machinery. For the focus of this project, the raised bed configuration was assumed and will factor into the design of the StakeBot.

After many visits with the local farmers and with the desire to automate the stake driving process, the need for preliminary research was done. Research into the market of autonomous vehicles for the agricultural industry was to ensure that a stake driving robot doesn’t already exist and to gain a better understanding of robotics in the agricultural industry.

CHAPTER 2:

PRELIMINARY RESEARCH

2.1 Early StakeBot Design Research:

Robots in agriculture are not a new concept. Robots have been aiding the process of planting, monitoring, or even harvesting for many years now. What has shown to be the upcoming innovation are the uses of neural networks (AI) and machine learning. For example, at the University of Cambridge, engineering students have developed the "Vegebot" that is capable of harvesting lettuce by using AI technologies such as deep neural networks to train a robotic arm to detect and pick lettuce. (Birrell, S, et al., 2020)

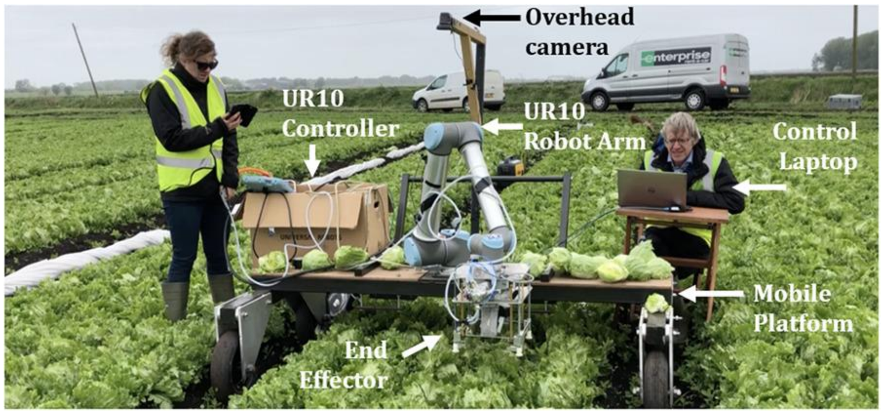


Figure 2.1: Vegebot robot created by the engineering students at the University of Cambridge for the purpose of harvesting lettuce. (Birrell, S, et al. ,2020)

Another example would be the company Harvest Croo who are a small startup company in Tampa Florida who have invented an autonomous robot capable of harvesting strawberries using a vision system that uses AI and machine learning to detect strawberries ready to be picked. (Harvest Croo Robotics)

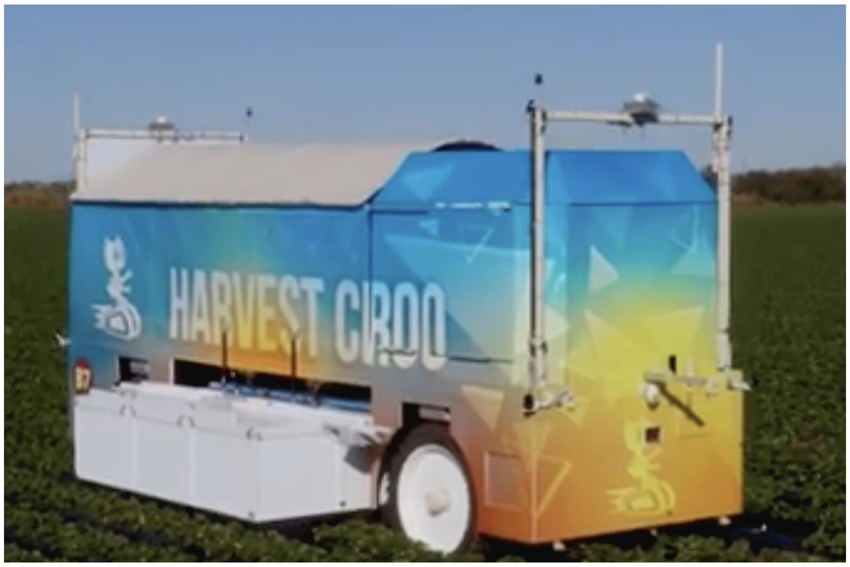


Figure 2.2: Harvest Croo's robot capable of detecting and picking strawberries ready for harvest. (Harvest Croo Robotics)

What is interesting about this product is that it also can self-navigate through the fields while detecting obstacles such as debris or workers who might be working around it. It does this by using 3D LiDAR technologies to scan the area around the robot where it can constantly monitor the activity happening around it. This product also takes advantage of deep neural networking and supervised learning to make decisions about the harvest status of each strawberry. Finally, a robotic arm picker will pick the strawberries and collect them in the bins on the sides of the robot.

One more company called Nexus who are out of Halifax, Nova Scotia, Canada have invented an autonomous robot capable for weeding the fields. (Nexus Robotics, 2020)

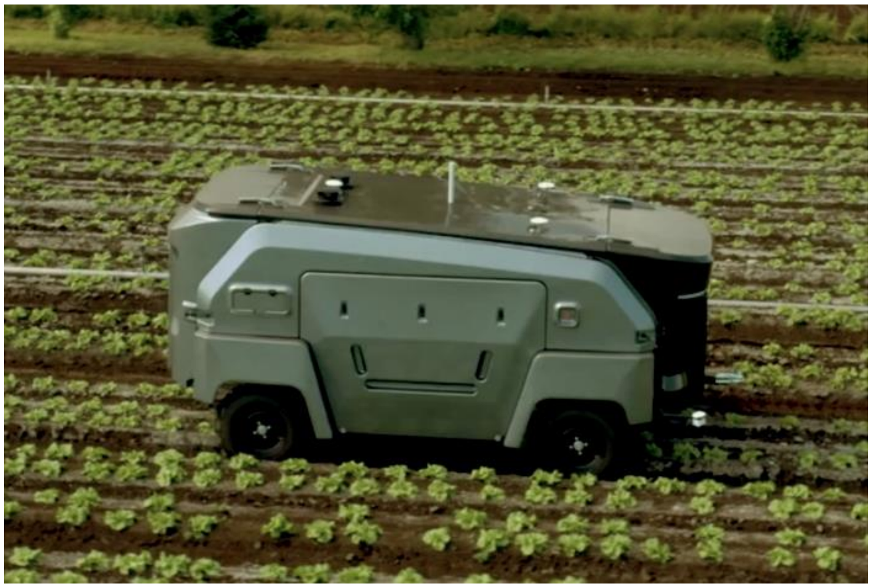


Figure 2.3: Nexus robot capable of picking weeds from the fields.
Developed by Nexus Robotic

The Nexus robot is designed to detect the difference between actual crops and unnecessary weeds that grow around the drop. Weeds are invasive to crop growth where they can take nutrients away from the plants and must be eradicated. The Nexus robot uses supervised learning and neural networking to learn the difference between weeds and the crop at all stages of growth and can remove the weeds from the fields. (Nexus Robotics, 2020) The robot also uses cameras to self-navigate through the fields without any human intervention.

After looking around the market, what is clear is that most of all the autonomous robots utilize a vision system and AI to either navigate or detect different crops, objects, etc. These technologies are not limited to any specific type of object. It is possible to train a neural network to detect any object assuming there is a large enough database for the system to learn from. For example, a highly popular detection network used in many different industries is the single stage detection neural network called YOLO.

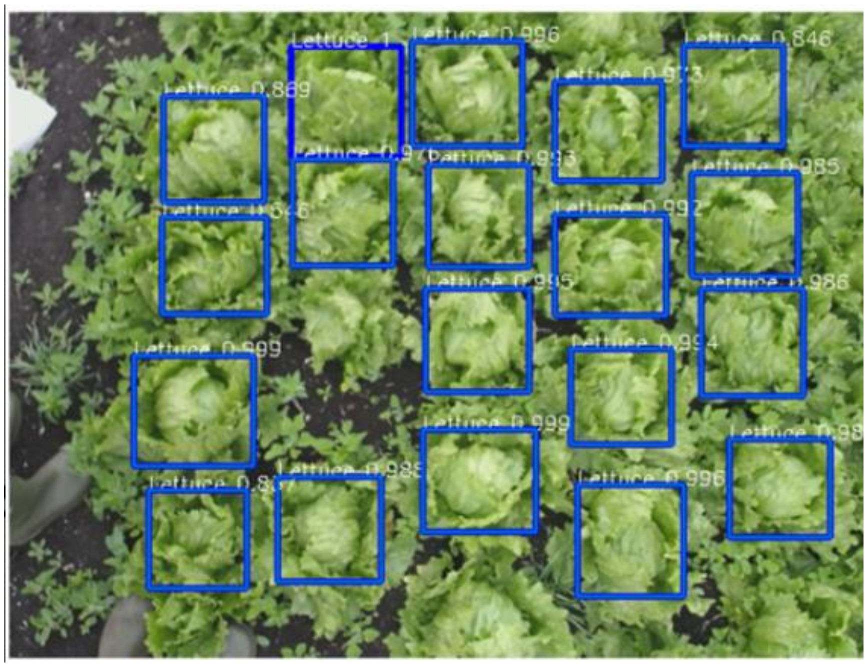


Figure 2.4 YOLO detection image. This image shows how the
YOLO algorithm detects objects, in this case, different lettuce

This network is a complex convolution neural network that extracts features from images of various objects as it attempts to learn the makeup and color of the object. (Li, Chuyi, et al., 2022) This network has been optimized by many different people in the computer science field. One of the popular networks currently is the YOLOv6 network which is an enhanced version of this network. The authors claim that it is about 1.4% more accurate with a 5% increase in time cost and runs about 21% faster than the previous adaptation of the network. (Li, Chuyi, et al., 2022)

For the StakeBot's autonomous navigation, as shown already in the current market, an AI driven robot has become the standard way of navigating through the agricultural fields. This concept was adapted, and a vision system was created to allow the StakeBot the ability for navigating by using stereo depth technology and an on-board

AI to drive the StakeBot. Later in Chapter 5, more details about the vision system will be discussed.

With a market full of autonomous, self-navigating robots already in the agricultural industry, the next bit of research was to find a robot that could satisfy the staking operations within the planning process shown in Figure 1.1. The closest product on the market is the ALMACO Stake Driver-Alley Marking System.

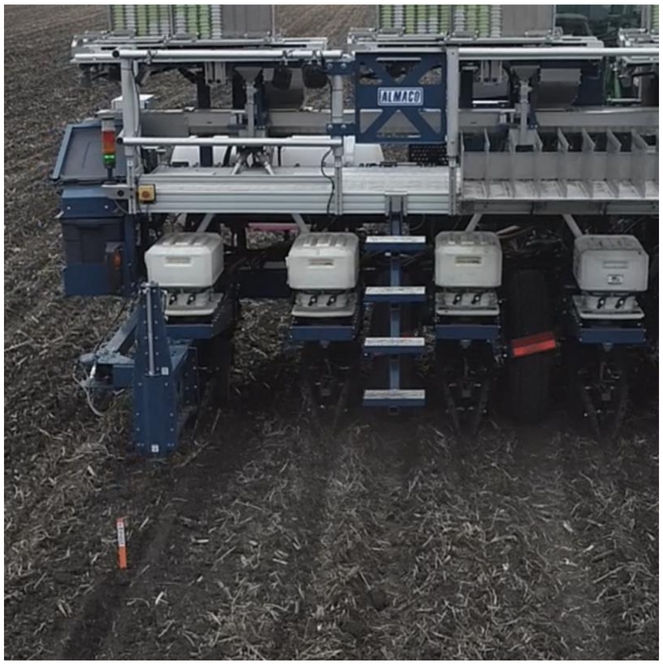


Figure 2.5 Stake Driver-Alley Marking System. Developed by ALMACO

This product can plow the fields and plant a stake all while being driven by a single tractor. (ALMACO) It is not autonomous, and it works in conjunction with plowing the fields. While this product does plant a stake into the ground, it overall does not meet the main focus for being autonomous. Another issue shows that this machine cannot work with raised bed plots which was an assumption and design constraint that was made due to the farmers already adapting this type of configuration for their planting process. Most farmers are not willing to completely change their way of operation so, having a robot that specializes in just driving stakes for their current setup is required.

The current market shows that there is not a type of robot that is capable of meeting many of the farmers' specifications for what they want the StakeBot to achieve. What the market has shown is that autonomous capabilities for an agricultural setting have been possible for some time. Next, research in the development of vision systems was done to understand the various concepts and algorithms used for robotic self-navigation.

2.2 Early Vision System Research:

For vision systems, the current technology mostly uses RGB or depth imagery to capture visual information about the environment. As there are two different ways to analysis images, each provide their own unique capabilities and have been cleverly used in different ways for providing sight to a machine/robot. Some publications focus on the capture of depth as a means for a machine/robotic vision system. Many engineers have developed new ways to process depth information that can be captured by different camera sensors. There are different techniques for measuring depth using different types of sensors. Shown in Figure 2.6 are three major types of depth capturing methods are structured light and coded light method, stereo depth method and LiDAR method. Structured light and coded light depth camera are similar technologies that use a projection of light onto a surface and compares the pattern of light to a known referenced light pattern. As the object moves closer or further away from the sensor, the deformed pattern of light compared with the projected pattern will change and a depth value can be calculated for each pixel. (RealSense, 2020) Certain drawbacks, however included light interference. If the environment is too bright or has too much light, this can interfere with the depth measurements. Typically, this type of depth sensing method is used indoors.

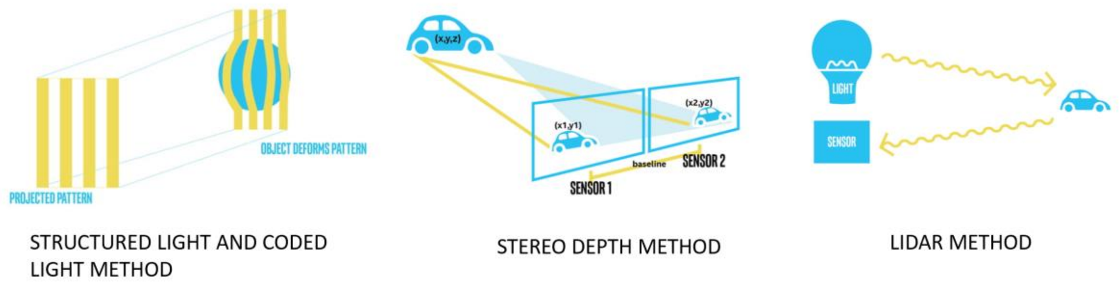


Figure 2.6: Different depth detection methods (RealSense, 2020) plants are detected with blue boxes marking their location in the image.

The stereo depth method uses two separate stereo depth cameras that compare two captures images and due to the known distance apart from the two cameras, a depth values can be calculated for each pixel. (RealSense, 2020) These cameras, unlike the structured light/coded light cameras, benefit from extra light noise. Most stereo depth cameras will include an infrared light projector that increases the accuracy of the depth measurements for low light conditions. This depth method is most suitable for outdoor and indoor environments.

Finally, LiDAR cameras are a time-of-flight method that bounces light off of objects where it can measure the time it took to travel to and from the object calculating a depth value. (RealSense, 2020) These sensors are typically used to measure specific distances and do not perform well in outdoor environments. For the purposes of the

StakeBot, the stereo depth method is the most ideal method of depth capturing and has been chosen as the primary sensor configuration for navigation.

There have been different processing algorithms pertaining to navigation including one being from an institution in Mexico called trajectory planning where the goal is to try and calculate new travel points in space to avoid obstacles while considering the robots size. (Básaca-Preciado et al. 2013) Figure 2.7 illustrates how the trajectory is planned and how the algorithm takes the size of the robot into consideration.

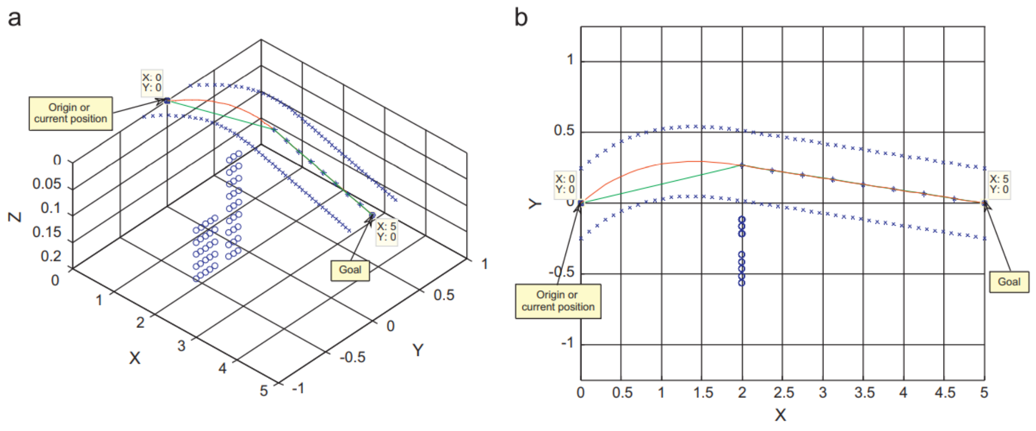


Figure 2.7: Mobile robot trajectory using trajectory planning algorithm (Básaca-Preciado et al. 2013) (a) isometric view (b) top view the blue circles are obstacles and the blue asterisks (*) show the boundary of the robot and the new navigation points. The red line shows the robot's new path.

Other approaches have been to try and estimate the velocities of objects as they approach the robot. (A. Cherubini, 2014) This allows for a better estimation of when to maneuver around certain objects, allowing for a dynamic change in response to the objects position. There are also certain algorithms that are well-known and are integrated within self-navigation including RANSAC and PEARL. RANSAC (Random Sample

Consensus) was an algorithm proposed by Fischler and Bolles that estimates general parameters in a dataset with large amounts of outliers. (Derpanis, 2010) This algorithm allows for a more accurate assumption when analyzing the imagery data. PERAL is an iteration of RANSAC that enhances the capabilities of the algorithm and has been used for crop detection during navigation. (Malavazi, 2018)

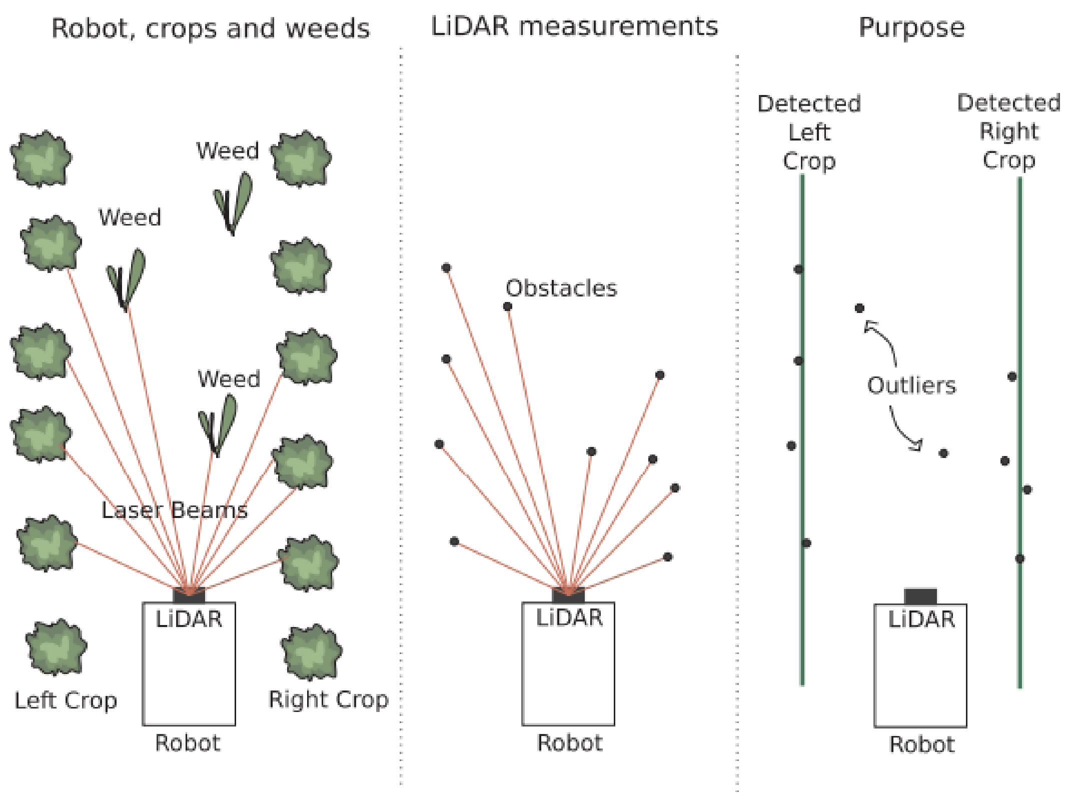


Figure 2.8: The considered line fitting problem in between bed plots (Malavazi, 2018)

Figure 2.8 shows how navigating between bed plots can be reduced to a line fitting problem that the author proposed to solve using the PERAL algorithm. This allowed for the robot to recognize certain crops with the LiDAR image while staying equidistance between the bed plots. Certain takeaways are present with this technology, being the majority of depth-based vision systems provide dynamic decision making, obstacle avoidance and, in the case of agricultural environments, crop detection.

For the RGB imagery, many different possibilities for autonomous navigation have also been explored. The majority of the research showed that many of the RGB style navigation is done by exploring different ways to extract "features" from the image. Subjecting machine learning algorithms to various datasets that describe the target environment is a fundamental way to train a robot to recognize and navigate through that environment.

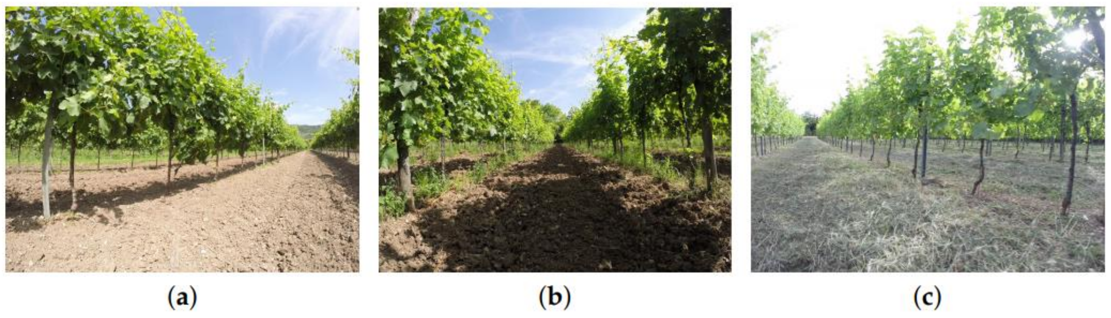


Figure 2.9: Example of three different samples of a dataset used to train a neural network for navigation presented in the paper by Diego Aghi, et. al. (a) shows an example of the left class of images which show a perspective towards the left of the vineyard. (b) shows the center class of images with the center view perspective. (c) shows the right class of images with the right view perspective. (Aghi, 2020)

For an agricultural setting, in the paper by Diego Aghi, et al. shows a demonstration of machine learning where a robot was to navigate through the vineyard using machine learning algorithms and an RGB camera. (Aghi, 2020) The type of machine learning was supervised learning and the dataset was a collection of images that were split into three classes of perspectives and is shown in Figure 2.9. This dataset of images was captured across different weather conditions and different times throughout the day which creates a more dynamic understanding of the environment for the robot to learn. (Aghi, 2020)

For more machine learning techniques, the extracted features that exist within the image are done with certain algorithms, one for these being color-index-based segmentation where the image is converted to a gray scale with high contrast that allows for certain objects in the image to appear more vibrant. (Yuhao, et al. 2023) Another feature extraction algorithm is threshold-based segmentation which compares and assigns each pixel into categories depending on a specified threshold. (Yuhao, et al. 2023) Certain methods are used widely for RGB based systems and have been shown to provide successful means of navigating through an agricultural environment. Most of these algorithms pertain to machine learning methods or other image processing algorithms that help to identify the features or objects within the image. Certain fallbacks, however include too much sunlight being an issue with RGB vision systems. The research shows that majority of RGB based vision systems use neural networks to extract features within the image and are compared to various known datasets where the system can make decision based on the discrepancies between the know images and its perceived view of the current environment.

In the next chapter, the StakeBot's design process began with a list of requirements that took note of the various advancements in the field of robotic vision systems. This list of requirements also took influence from the current products that exist in the market today.

CHAPTER 3:
STAKEBOT'S DESIGN

3.1 Introduction:

The next step involves the creation of design requirements that make the StakeBot appeal to the agricultural industry. This list of design requirements was created from the feedback collected from the local farmers and was organized into a table with target and fallback specifications. The table of design requirements, given in the next section, shows the current expectations for a robot performing the stake driving process. Many of these requirements were developed by farmers with relatively large fields and although some of the requirements are not met for the current StakeBot, the progress of developing an autonomous robot capable of stake driving is the major focus of this thesis. The current size of the StakeBot will have to be greatly increased to meet certain time constraints thus, the targeted market for the current StakeBot design is for smaller farmers with smaller fields (less than 10 acres of fields in operation).

3.2 StakeBot Design Requirements:

Table 3.1 shows a list of design requirements that became the motivation behind the development of the StakeBot. This table specifies the next challenges that the StakeBot must overcome to comply with many of the farmer's standards. The design of the StakeBot meets the majority of these specifications except for requirement 2, 4 and 5. For these requirements, a larger scaled StakeBot was proposed and is under development for phase 2, however, many of the assets of the current StakeBot's design will carry over to this larger design in the future as many of the operations are still the same.

Table 3.1: StakeBot design requirements (Entries with a * are further explained in Appendix A)

Design Matrix			
No.	**Design Requirement**	**Target Specifications**	**Fallback Specifications**
1	Robot must be able to drive both circular and square shaped stakes into bed plots	Both circular and square	Only circular stakes
*2	Robot must be able to carry enough stakes to complete a section of the field at a time before needing to be reloaded	3 rows (1,116 Stakes)	1 row (372 Stakes)
*3	Robot must be able to navigate through the rows without inflicting self-damage or damage to the fields/crops	Fully autonomous (no user control necessary)	Partially autonomous (Some user controlling necessary)
4	Robot must be "smart" enough to differentiate the difference between a bell pepper plant and the bed plot	Can detect the difference between any plant and its bed plot	Can tell the difference between only a bell pepper plant and its bed plot
5	Robot must complete a field within a timely manner	40 acers within 4 days (27,000 Stakes/day)	40 acers within 7 days (15,430 Stakes/day)
6	Robot's primary operations include both driving and removing the stakes from the bed plots	Capable of planting and removing	N/A
7	Robot must drive Stakes in a pattern like configuration where every 100 ft is dedicated to being open where employees can navigate between the rows	Every 100 feet needs to be a break between stakes of about 1 foot or one - two plants long	N/A
8	Robot must fit between bed plots	Current design is 18" wide	N/A
9	Robot must tall enough to clear the bed plots	Clearance will be 20" tall	N/A
10	Robot must communicate to the end-user necessary information about its operations, battery life and other important details regarding its use and environment.	User will have an application that will connect to the robot's internal WIFI to allow the end-user to control and receive information from the robot during operation.	N/A

There is, however, one major design component that was not known by the farmers that must be considered for development. This constraint is the force requirement of driving a stake into the bed plots. In the next section, an analysis was performed to gain a better understanding of the force requirement needed for driving a stake into a raised bed plot.

3.3 Stake Driving Experiment:

For the development of the StakeBot, one special design requirement that was not listed in Table 3.1 was the amount of force required to drive a stake into the bed plot. Many of the farmers claimed that their workers made little to no effort driving the stake however, for the design of the StakeBot, this force value is very important as it will define the strength of the materials required. With permission from a local farmer, a force experiment was done in late August of 2022 that provided a benchmark for the force required to drive both a circular and square support stake into the bed plots. Both types of stakes were used as different farmers prefer either circular or square stake. Circular stakes seemed to be the most popular amongst the different farmers, however some prefer square shaped. This preference serves no deviation from the planting process, however the targeted design requirement for the StakeBot must be compatible with both styles. The experimental setup is shown below in Figure 3.1 where a NI-9219 DAQ was used to collect force sensor data that was connected to a stake driving apparatus that was used to drive a stake into the bed plot.

With the experimental setup shown in Figure 3.1, the experiment was repeated for both square and circular stakes at the fields of one of the local farmers. The performed experimental procedures were to load a stake into the driving apparatus push downward until the stake was 8 inch in the bed plot while recording force measurements using the DAQ. This was repeated 5 times for both square and circular type stakes. A laptop running SignalExpress 2015 was used to capture the DAQ data, and the apparatus used wooden stakes to support itself upright with a level being used to ensure that the driving direction was normal to the earth's ground.

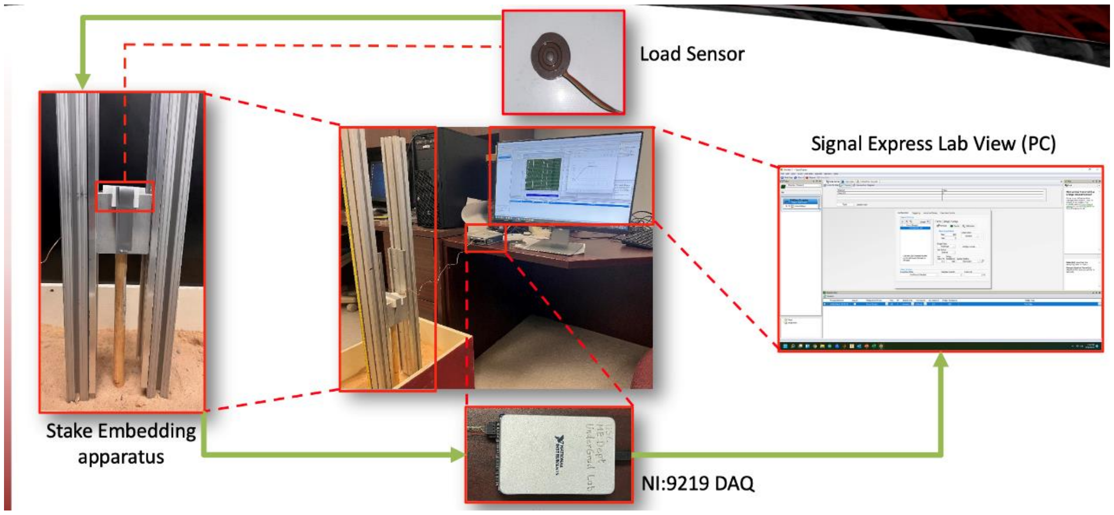


Figure 3.1: Stake driving force experimental setup. This shows the setup being testing within the lab.

The results for the Stake Driving experiment were analyzed and are shown in Table 3.2 below. The results include an averaged max force with a 15% factor of safety. The max force over all the trials was also recorded with an applied 15% factory of safety. For the design requirement, the force selected was the max force out of both stake types with the 15% factor of safety applied. 6.20 lbs. of force (Highlighted in green) were considered the required force needed to drive the stakes into the bed plots. This experimental result was partially justified by the local farmers who claim the force number seemed to be accurate to their experience, however this experiment does have a few errors that need to be addressed. First, there were some issues with the DAQ and for the square type stakes, only one force reading was captured. Second, the force experiment was only performed in one small location in the fields. This doesn't take into consideration any other potential debris that could make its way into the bed plots during its construction.

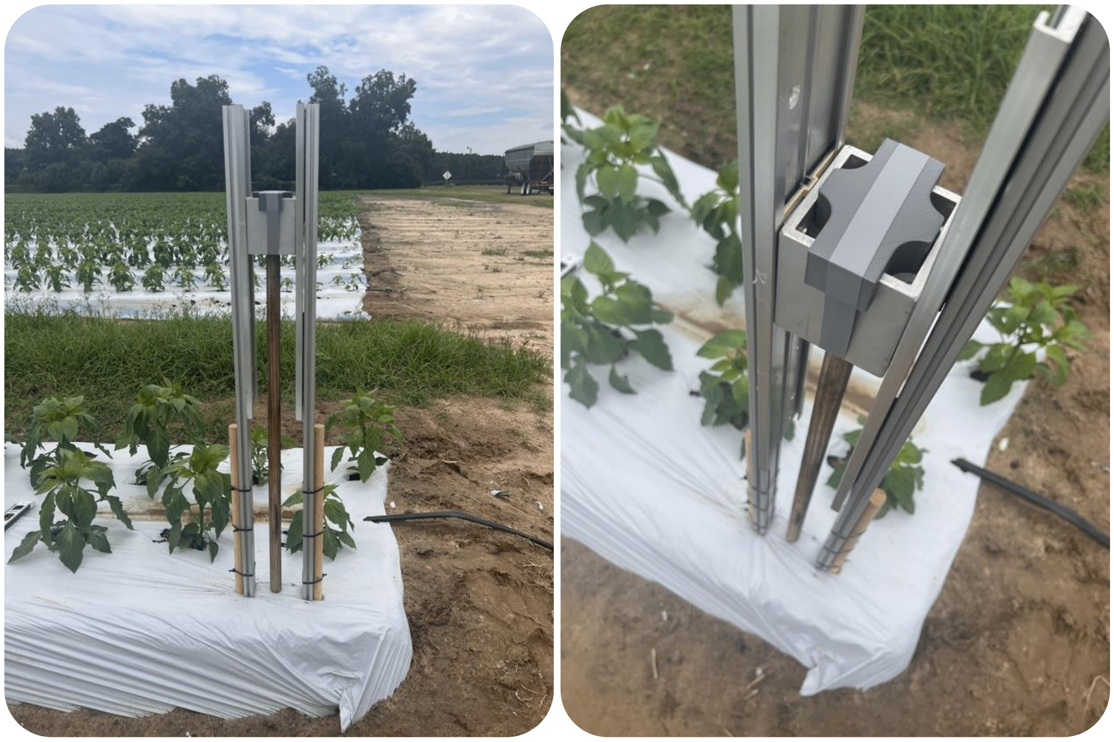


Figure 3.2: Stake driving force experimental setup in practice.

For example, harder objects such as rocks could be hidden within the soil, and this could cause the force requirement to be dramatically larger. To compensate for the extreme scenarios, the minimum force requirement was increased to 25 lbs.

Table 3.2: Stake driving experimental results.

Circular Stake Type Analysis			**Square Stake Type Analysis**		
Average Max Force	3.34	lbs.	Average Max Force	3.72	lbs.
Factor Of Safety	15%		Factor Of Safety	15%	
Average With FOS	3.84	lbs.	Average With FOS	4.28	lbs.
Max Force	5.39	lbs.	Max Force	3.72	lbs.
Factor Of Safety	15%		Factor Of Safety	15%	
Max Force With FOS	6.20	lbs.	Max Force With FOS	4.28	lbs.

With a final design requirement of 25 lbs. of force, and the design requirements shown in Table 3.1, the development of the StakeBot was able to proceed.

3.4 Proposed StakeBot Design:

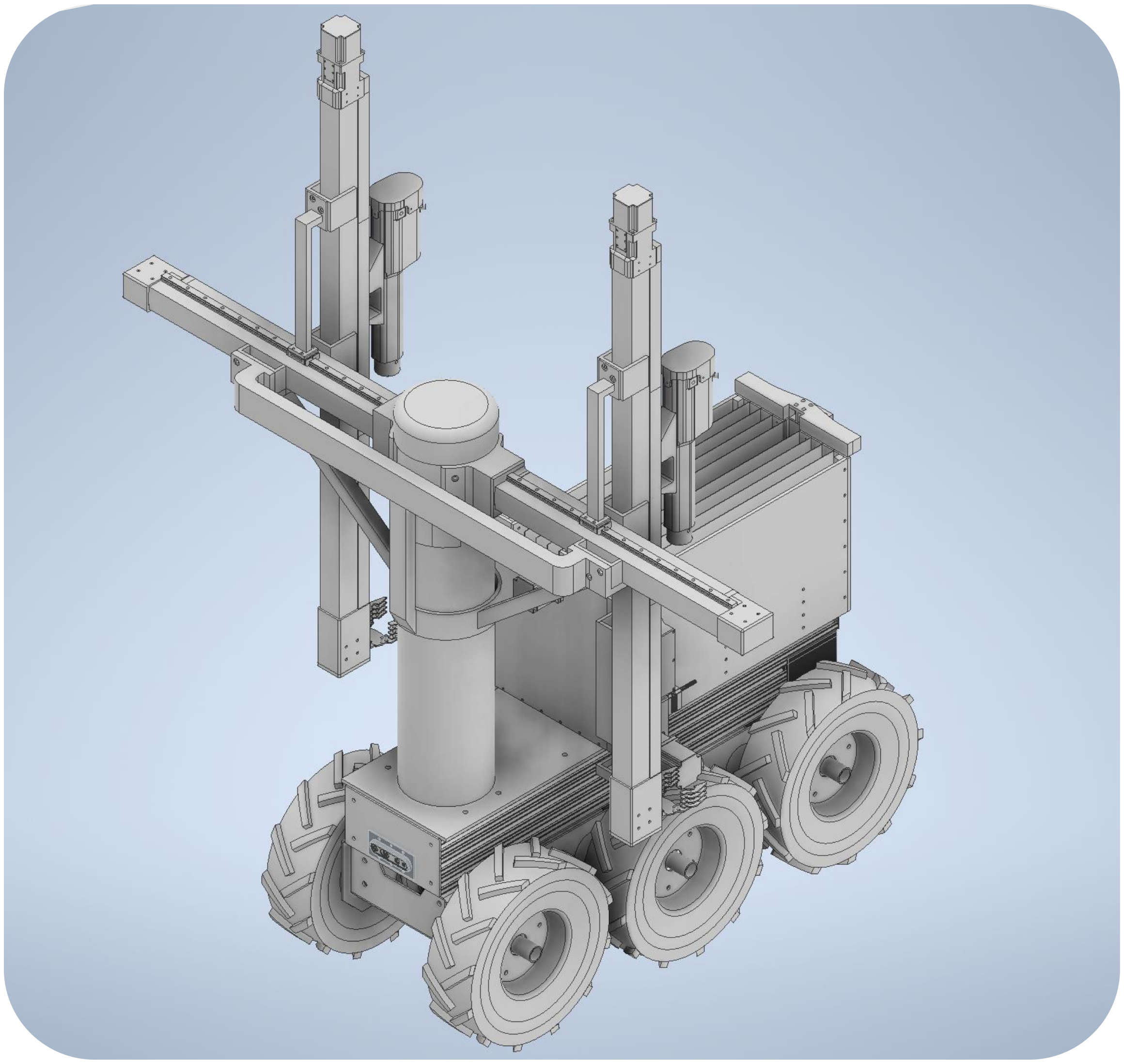


Figure 3.3: StakeBot's final design (Front ISO)

The StakeBot is a fully autonomous robotic vehicle that is capable of driving support stakes into the bed plots of any agricultural field. This robot was designed with the majority of the local farmers' planting operations being the standard. This means that this robot can navigate through the fields and is designed to work around the lifted bed plots and various types of terrain. This StakeBot includes 3 various components that are listed in detail across the next three chapters. These components are the StakeBot's Base, the main arms and the stake driving arms. There is one other component that is shown in the design and is seen on the back side of the StakeBot. Its responsibility is to hold and transport the stakes to the stake driving arms. The reason it is not listed in the next proceeding chapters is that it will not be discussed in this thesis due to its undesirable design.

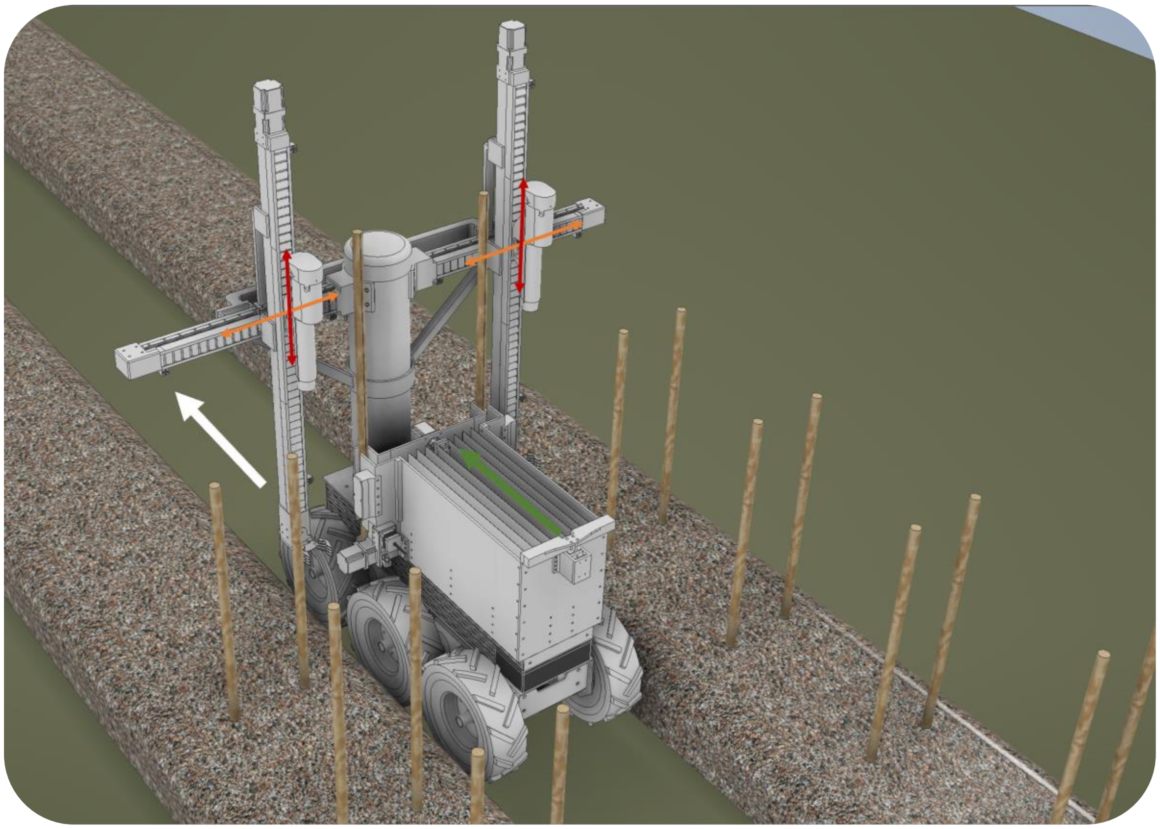


Figure 3.4: StakeBot's primary functionality. The stakes are pushed to a pickup location (green arrow), The main arms move to the pickup location then over top of the bed plot (orange arrow). The stakes are driven into the ground by the attachment (red arrow). The StakeBot moves in the direction of the white arrow.

A new design will need to be implemented for the final product, however since it is not the main focus of this thesis, and for the sake of understanding, it is important to mention its responsibility. Even though there is currently not a fleshed-out design for the stake holster, the StakeBot is designed to accompany one both physically and programmatically. Shown in Figure 3.4 is the functionality of how the StakeBot's components operate together.

The StakeBot works by first retrieving the stakes from the holster which sits on the back of the StakeBot. Then, the arms move to retrieve the stakes, the grippers will grab the stake from the holster and then the stake driving arm will lower overtop of the stake. Once the stake is grabbed, the main arm will position the stake overtop of the bed plot and then the stake driving arm will drive the stake into the ground. After the stake is properly inserted into the ground, the StakeBot will drive to the next location and the process repeats.

For ensuring that the structural integrity is met, a structural analysis and optimization needed to be performed. This optimization is important as to prevent over engineering the design. This means that the design should be built in a way that does not exceed the various structural constraints. For these arms, these constraints are as follows: beam stress, dimensional, beam buckling, lead screw stress and lead screw buckling. There are a total of 6 constraints with the dimensional being a combination of two separate dimension constraints. The two dimension constraints are set in place to ensure that the optimization algorithm keeps the C-Channel dimensions a minimal of 15% larger than the lead screw. This keeps the design physically possible as the lead screw must be able to fit inside the C-Channel. These constraints are important for the arm's design as they govern what the design is capable of. If any of these constraints are not met, then the arms could break upon operation. When performing the optimization, the objective function was to minimize the total volume of material used while still being structurally stable. The design variables are shown in Figure 3.5. They are the radius of the lead screw, and the dimension of the C-Channel leaving a total of 4 design variables.

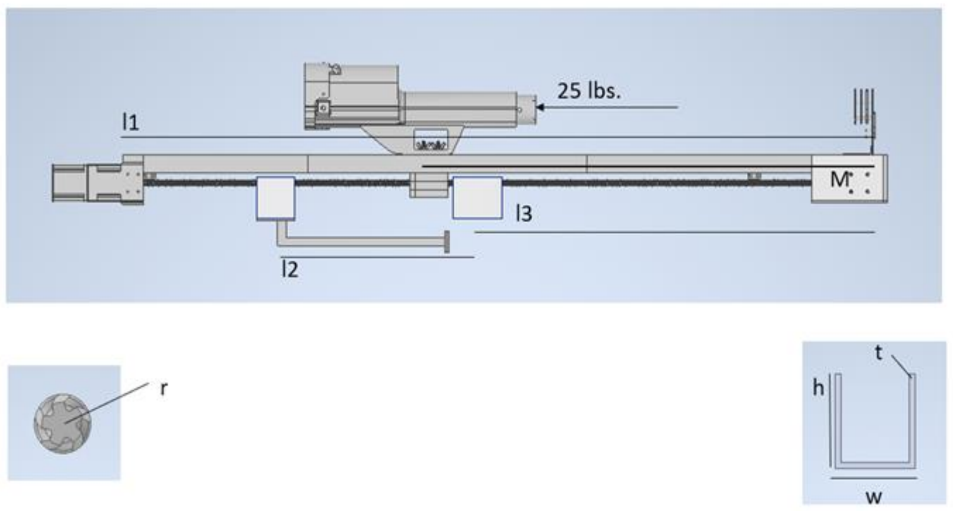


Figure 3.5: Free body diagram of the StakeBot’s stake driving arms with the design variables shown underneath. l1 is the length of the C-channel, l3 is the distance from the right side to the first attachment point, l2 is the length of the support arm, and M is the distance the attachment is from the far-right side.

For this design, the optimization constraints were calculated with the attachment under the assumption of 25 lbs. being moved across the bounds of the C-Channel. This means that for each constraint to pass, the 6 constraints were calculated with the attachment starting from the highest point to the lowest point. For the stake driving arms, the highest point is seen in Figure 3.5 at the far-left side which starts 1 inch away from the edge of the C-Channel. It then moves downward and stops 3 inches away from the far-right side (bottom) of the arm. In Figure 3.6, the constraints are shown at each point across the arm as the attachment moves from the left to the right (top to bottom). The constraint variable was considered passed if the max value didn't violate the constraints. It shows that with a lead screw diameter of 3/8 inch, a width of 1.75 inches, a height of 2 inches and a thickness of 1/8 inch, these design variables are suitable for handling the force of 25 lbs.

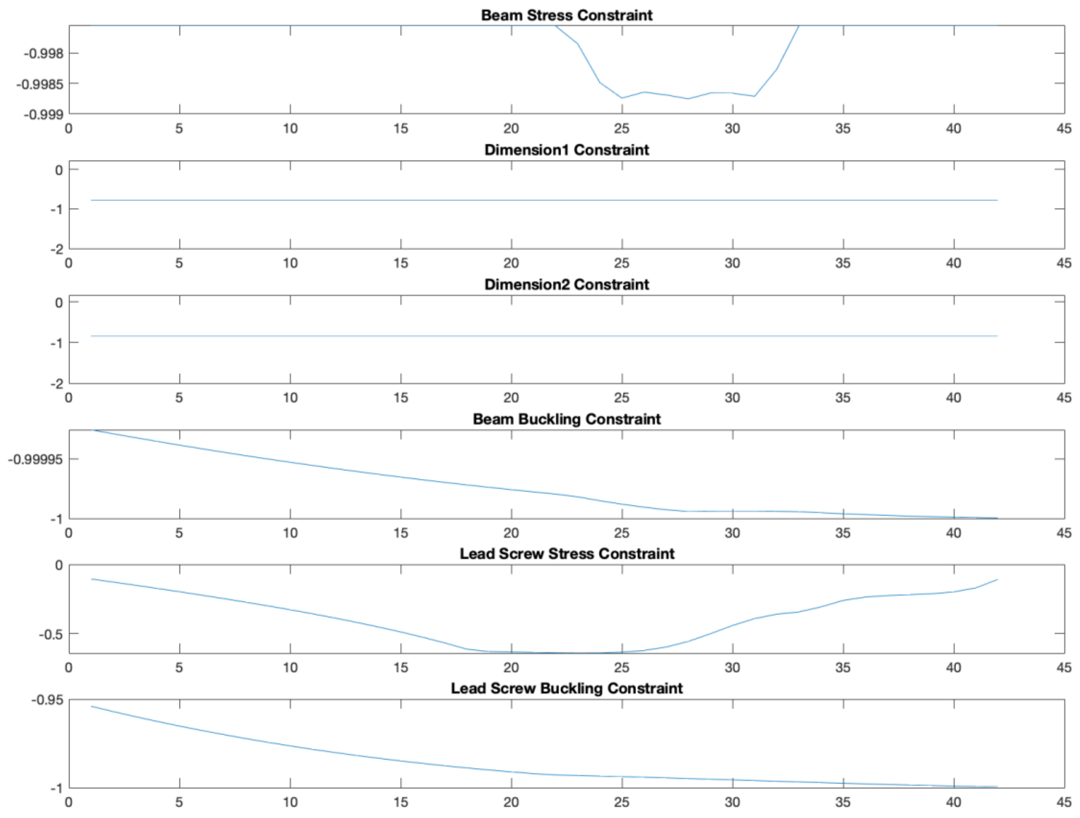


Figure 3.6: Stake driving arm constraint values across the length of the arm. These values have been normalized with positive numbers being considered a violated constraint. The horizontal axis is the length of the beam, and the vertical axis is the normalized value of the respected constraint variable.

For the stake driving arms, the maximum values of each of the 6 constraints are shown the Table 3.3. These values are the normalized values that indicate a percentage of how much more capable the design is under these conditions. This algorithm and results were calculated using MATLAB and it shows that the most concerning constraint is the lead screw stress constraint. At this constraint's max, there is only about 11% more stress the screw can handle before fracture however, this still means that the design will not fail even at this condition. Also, this condition only happens as the attachment moves closer to the bottom (shown in Figure 3.6) and this condition would be rare due to the average heigh of the stakes compared to the desired depth being driven. To add perspective, the stakes average about 3 feet tall and for majority case, they only need to be driven 8 inches in the bed plot. This would leave the attachment only moving 34 inches from the top leaving 8 inches away from the bottom bounds of the arm. This proves that the design at these dimensions is suitable.

Table 3.3: Stake driving arm normalized design constraints.

Beam Stress	**Dimension 1**	**Dimension 2**	**Beam Buckling**	**Lead Screw Stress**	**Lead Screw Buckling**
-0.9976	-0.7826	-0.8370	-0.9999	-0.1052	-0.9541
99%	78%	83%	99%	11%	95%

Given that the design is suitable for supporting the force of driving the stakes, the StakeBot's main arms also underwent the same calculations to ensure that the arms could carry the stake driving arms and support the forces during the stake driving operations. For this set of calculations, the design is the same as the stake driving arms except for both orientation and length. These arms are shorter and extend horizontally instead of vertically. This helps limit the buckling forces on the screw and the C-Channel, but the arms now require extra support. A brace was designed to compensate for this support, not just for the main arms but also the inertial forces that act on the arms from the StakeBot’s movements.

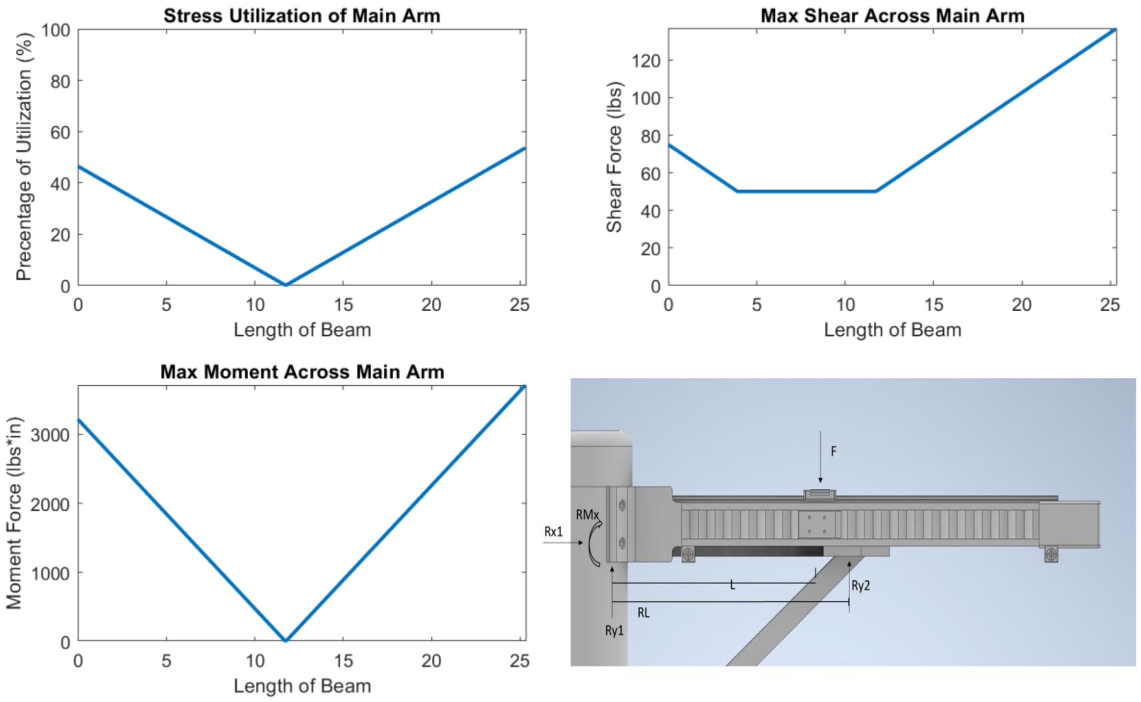


Figure 3.7: Main arm structural analysis with the free body diagram shown. With L being the distance of the arm mover, RL being the distance from the supports, Ry1, Ry2, Rx1 and RMx are reaction forces and F is the force of the stake driving arm.

The plots shown in Figure 3.7 are the calculated max shear, stress, and moment diagrams during the scenario where force (F) value is applied across the total length of the arm. The max stress happens when L is at the far-right side of the main arm bounds. It is shown in the plots in Figure 3.7 that the max stress utilization for the C-Channel is 54% showing a little over half of the arms’ structural capabilities.

It is with these analyses demonstrating that the StakeBot is structurally capable of handling 25 lbs. of driving force. The design shows promise and is unlike anything in the current market. With this design the StakeBot is capable of maneuvering safely while having the capability of driving the support stakes into the bed plots. In the next chapters, each of the StakeBot’s components are described in greater detail on their design and functionality.

CHAPTER 4:

STAKEBOT'S BASE

4.1 Introduction:

The first component of the StakeBot is its base. The base provides the StakeBot with the capability of movement and support for all other components. It also houses all major electrical components within and provides protection for the outside elements. The StakeBot's base will be shown in greater detail with discussion about the decisions made to its structural and electrical components.

4.2 Base Design:

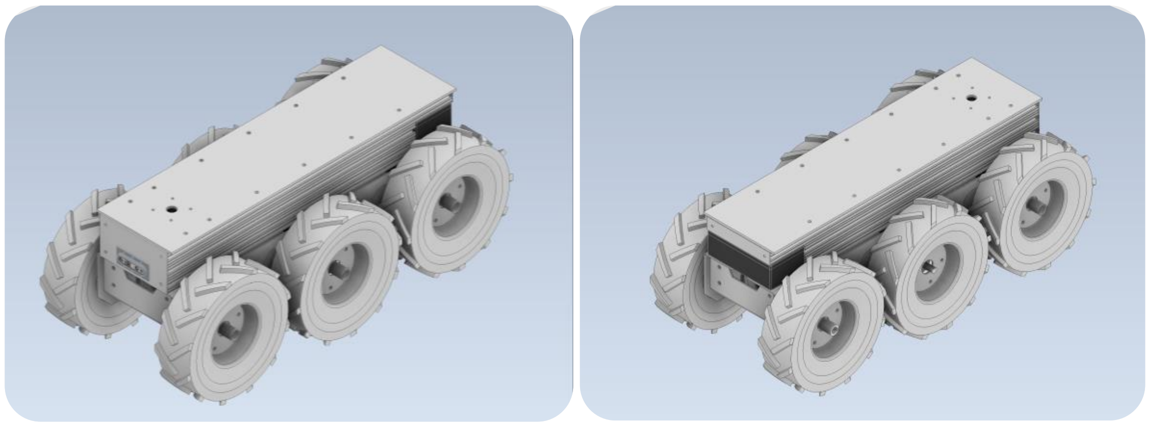


Figure 4.1: StakeBot base (Front ISO - Left), StakeBot base (Back ISO – Right)

The highlights of the StakeBot's base, as shown in Figure 4.1, are the 6 independently driven 13-inch tires, a tough aluminum build with a waterproof housing for the control system, a max speed of 7 MPH and a total weight capacity of 200 lbs. The base was specifically designed to follow the requirements shown in Chapter 3, Table 3.1 with the major design requirements being the dimensions and number of stakes to carry. The StakeBot must fit within the rows of the various plant beds whose minimum distances are 22 inches apart while also being able to transport a relatively large number of stakes. The design direction was to create a robot that fits between the rows while still being large enough to carry as many stakes as possible. During the initial visits with the local farmers, the majority of the farms with raised plant beds had a dimension of 8 inches tall which influenced the height of the StakeBot.

The final dimensions for the StakeBot's base were designed to be 3 feet long, 18 inches wide and 16 inches tall. The 18-inch width includes the tires and was designed to be as thin as possible. The 3 feet length was decided because of the desire to have a robot that can easily maneuver around the fields without being too long with respect to its width. The width to length ratio is 1:2 making the length only twice as long as the width and allowing for sensible maneuverability. The height being 16 inches was heavily dictated by the control system and other electronics that are housed in the center portion of the base's body.

The decision for the materials used to construct the base was chosen to be 6063 aluminum parts due to it being easy to fabricate and its lower weight compared to steel. The decision of materials was chosen between either steel or aluminum as these are the most used materials for fabrication. The design required a base that was robust and able to withstand various weather conditions including hot or cold weather and tough terrain. Due to the nature of the environment, the StakeBot will be operating over uneven ground, mostly dirt or rock, sometimes mud depending on the weather conditions. To address this, a 6-wheeled, independently driven 13-inch tire design was implemented. The tires are rubber, rated for 300 lbs. at 30 psi. with heavy treads that allow them to handle well over rough terrain and independently driven to allow for better handling over uneven ground. With 6 heavy duty tires, the weight capacity would rate somewhere around 1,800 lbs., however the limiting factor is the DC motors which will be further discussed in Section 4.4. The 6063-aluminum used for the body of the base has a yield strength of 23,000 psi. with a density of 0.0975 lbs./in^3. Aluminum compared to steel, which has a density of about 0.283 lbs./in^3, shows that aluminum is about 3 times lighter, this allows for significantly less stress on the tires and allows for more stakes to be carried by the base.

The StakeBot's base was designed to be the heaviest component keeping the center of gravity low. To achieve this, the heaviest components include the two batteries, and the 6 DC motors, are housed near the lowest point of the base's body. This provides better stability for the StakeBot and keeps a balance around the center of the base's body.

4.3 Base Functionality:

The StakeBot base's functionality is to provide mobility to the StakeBot. The function of the drive wheels moves in a differential configuration meaning that the left three tires and right three tires can turn at different speeds. This allows the StakeBot the ability to turn at any radius including a zero-turn radius. The StakeBot's base is controlled by either the on-board AI or the user through the StakeBot's control system. The control system is housed within the center of the StakeBot's base which provides protection from the outside elements. In the next section, the base electrical system, which is the makeup of the control system, is discussed in further detail.

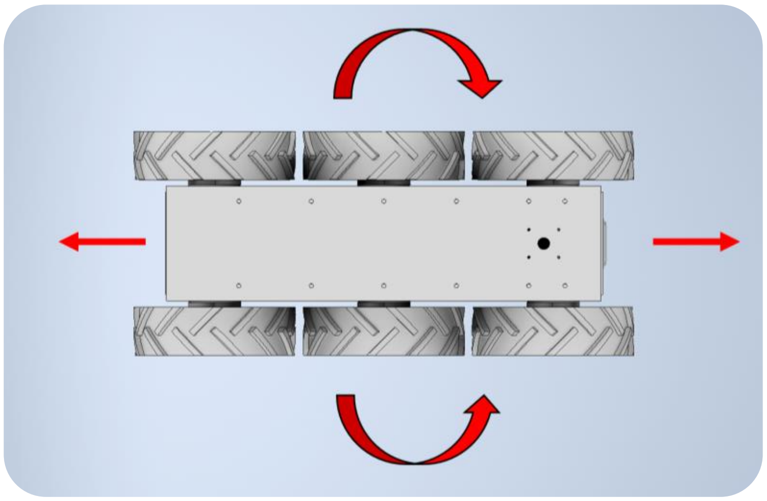


Figure 4.2: StakeBot base’s functionality with arrows showing the possible directions that the StakeBot can move.

4.4 Base Electrical System:

As mentioned, the StakeBot’s base houses all the components of the controls system. Each of the components are shown in the wiring diagram above in Figure 4.3. These components are responsible for determining how the StakeBot functions and performs its duties. These components are attached to acrylic panels that are mounted inside the StakeBot base’s body. There are two panels with the top panel being removeable for ease of access and an electrical box that houses the RoboClaw motor driver which is responsible for controlling the base’s DC motors. The electronics are surrounded by aluminum rails and panels that keep the components protected from the elements. The panels are sealed but are not completely waterproof.

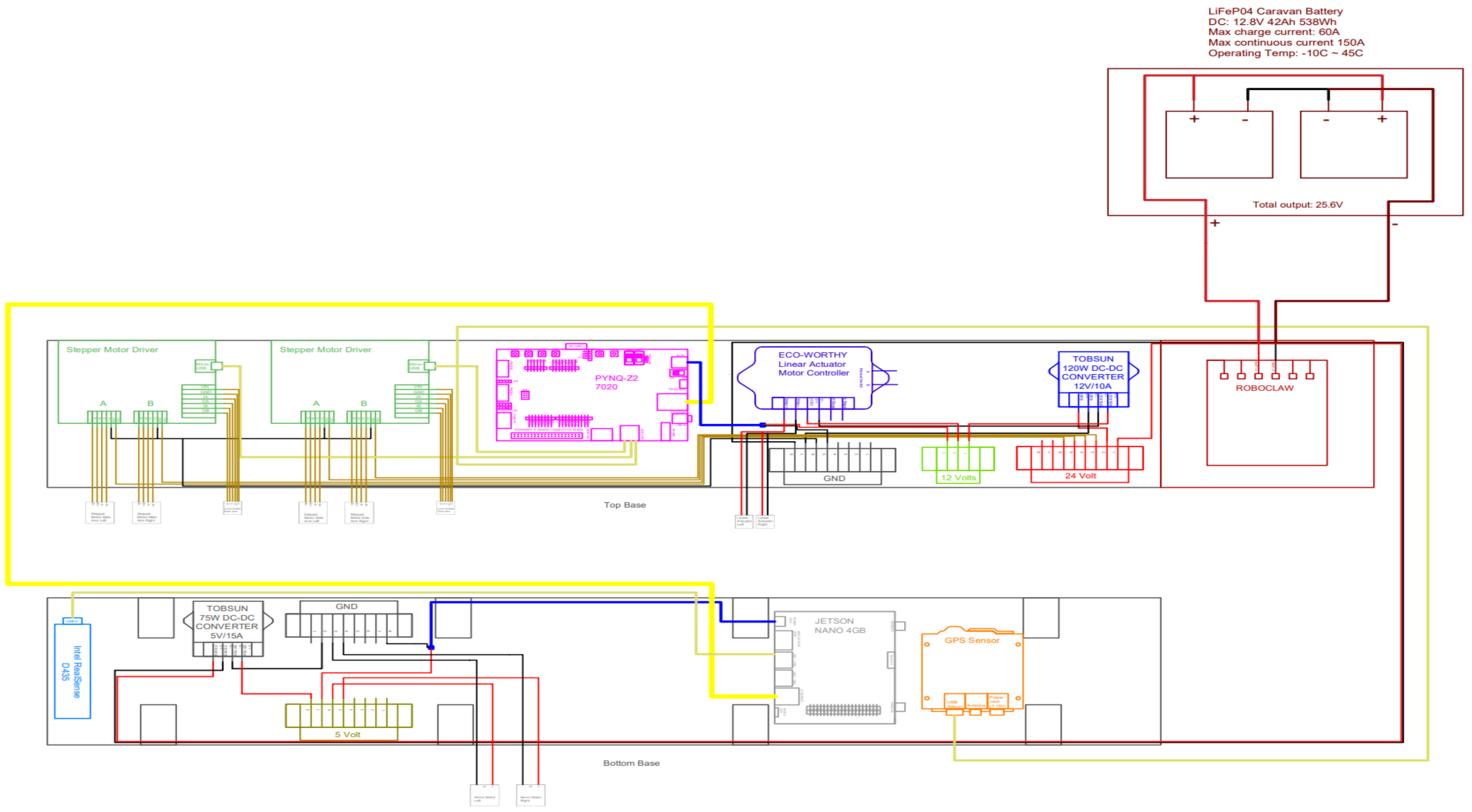


Figure 4.3 Electrical diagram of the control system inside the StakeBot’s base.

The entire controls system is made up of various components that are responsible for specific operations. The Jetson Nano and PYNQ-Z2 boards are the heart of the controls system. The Jetson Nano is responsible for operations including vision control, user to machine and machine to machine communication, and motor controls. The communication protocol is handled by the Jetson Nano due to its WIFI capabilities and uses the MQTT protocol which is a lightweight open messaging protocol developed for IoT devices. (Steve, et al., 2021) The MQTT communication protocol allows for the user to communicate instructions to the StakeBot and for the StakeBot to convey any information deemed necessary for the user such as battery life, any problems with the operations, etc. The MQTT protocol is also how the Jetson Nano communicates to the PYNQ-Z2 board where it receives instruction on how to proceed. The Jetson Nano also controls the RoboClaw motor drivers by translating commands sent to the driver into the correct power output for the DC motors. Lastly, the vision system is a complex series of algorithms that use the RealSense D435 sensor to determine objects within its path and other capabilities pertaining to the movement of the StakeBot. More on the vision system will be discussed in Chapter 7.

The PYNQ-Z2 board is responsible for controlling the various components that operate the various StakeBot subsystems who are responsible for tasks such as the stake driving operations and StakeBot navigation. The PYNQ-Z2 board is a micro-computer hosting an FPGA chip running a Linux operating system like the Jetson Nano. This board is unique in its design for machine learning and other robotic applications. The purpose of this board is to host the AI that is responsible for completing the job at hand. All the StakeBot’s operations are given instructions from this board. For example, the AI will tell the Jetson Nano to move to the next location following the GPS subsystem which provides latitude and longitude data. Once the StakeBot has reached the location it will inform the PYNQ-Z2 board who will then command a stake to be driven. The stake driving subsystem will then perform a stake driving operation and once that stake has been planted, the process repeats. To better visualize the PYNQ-Z2's operations the diagram shown in Figure 4.4 illustrates the flow of logic for the StakeBot's control system. It's best to think of the PYNQ-Z2 board as a manager, all the other components follow the instructions given to them by the manager being either instructions on what to do or asking for information handled by the respected device. The other devices will respond depending on the situation it finds itself in. For example, the Jetson Nano may be instructed to move to the next staking location, however, there appears to be a giant rock in the way which was picked up by the Jetson Nano's vision system. The Jetson Nano will respond to the PYNQ-Z2 board with the issue. The PYNQ-Z2 board will either respond with an alternative instruction or will tell the Jetson Nano to inform the user about the giant rock. For a more descriptive understanding of the controls system between the PYNQ-Z2 board and Jetson Nano will be discussed in greater detail later in Chapter 8.

The control system is what makes the StakeBot function, and it is crucial that it remains protected and unaffected from any outside forces or elements. The StakeBot's base provides adequate protection and is designed to be robust and strong enough to work in an agricultural setting. The next component is the StakeBot's main body and will be the subject for the next chapter.

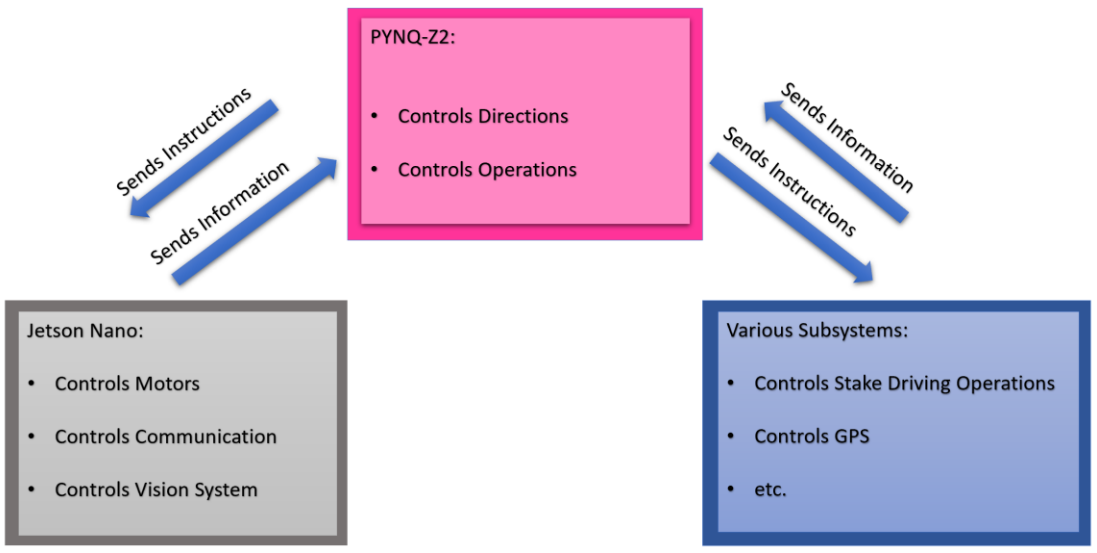


Figure 4.4 Diagram of the flow of logic between the PYNQ-Z2 board and other parts of the controls system.

CHAPTER 5:

STAKEBOT’S MAIN BODY

5.1 Introduction:

The next component to the StakeBot is the main body. The main body serves as a means of positioning the stake driving arms (discussed in the next chapter) either over the bed plots for driving or in the home position for retrieving a new stake. The details and decision for the design as well as the controls and functionality are discussed in the later sections.

5.2 Main Body Design:

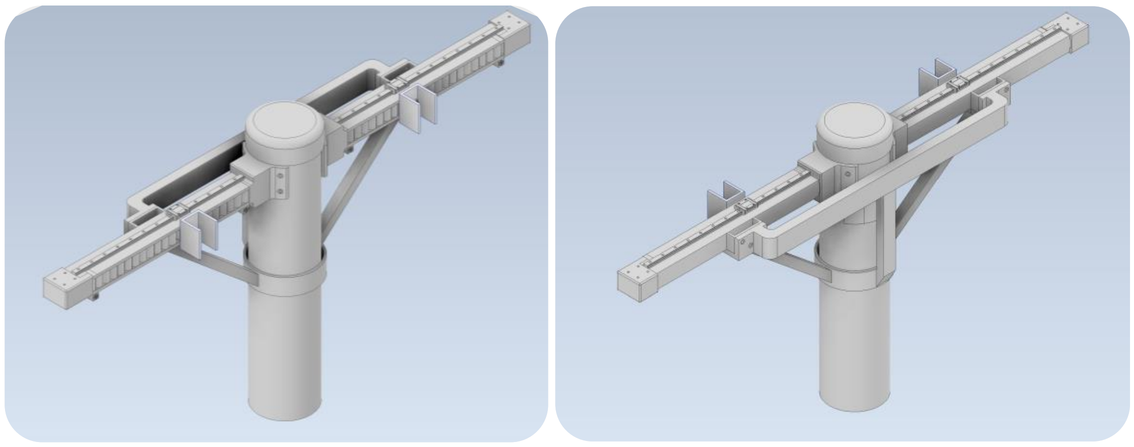


Figure 5.1: StakeBot's main body (Front ISO - Left), StakeBot's main body (Back ISO - Right)

The StakeBot’s main body includes two side arms that move outward and inward to allow the StakeBot driving arms to maneuver overtop of the plant beds allowing the stakes to be placed accordingly as shown in Chapter 3, Figure 3.4. The arms are supported by a brace that allows for support against inertial forces for when the StakeBot is in motion and are attached to the main body's cylinder. The main body's cylinder houses stepper motors that are used to control the arms and are supported by a custom plate that holds the motors in place by attaching to the inside of the cylinder. Details are shown in Figure 5.2. The main body's cylinder is made from PVC plastic and the brace is made from thin steel tubing.

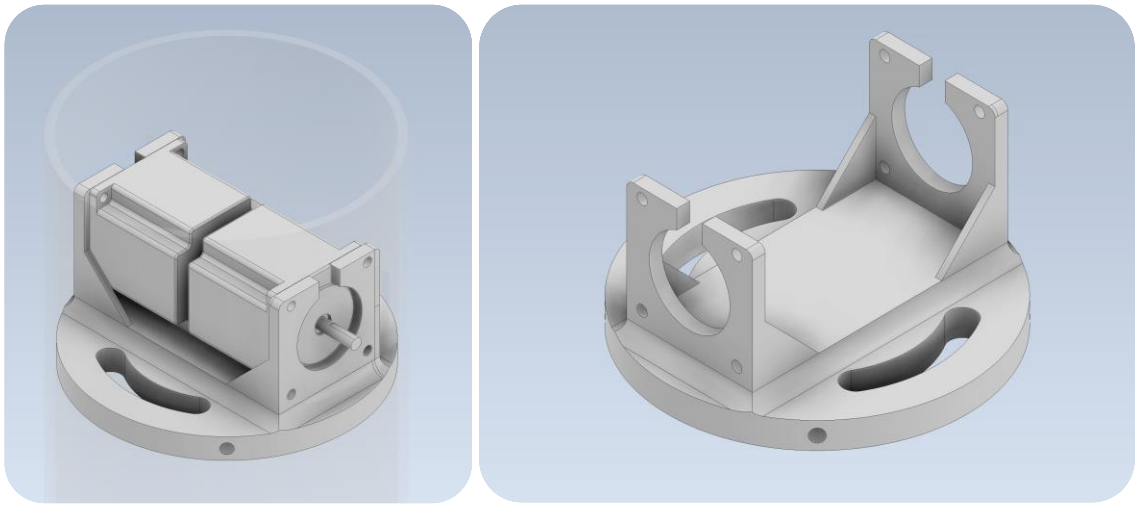


Figure 5.2: StakeBot's main body motor mount attached and assembled (Left), StakeBot's main body motor mount (Right)

The motor mount attaches with machine screws from the side wall through the cylinder, and the motors are fastened to the face plates that extrude from the base of the motor mount. Slots are cut out to allow for wires to extend down into the StakeBot's base which houses the electronics. The motor mount is made from a 3D printed nylon glass filled material.

The StakeBot's main arm includes components that make up a typical electrical linear actuator with an aluminum C-Channel housing the components and is shown in Figure 5.3. The arm allows the StakeBot's stake driving arms to move inward and outward from the main base. The arms are responsible for retrieving a stake from the retrieval point and to position the stake driving arms over top of the bed plots.

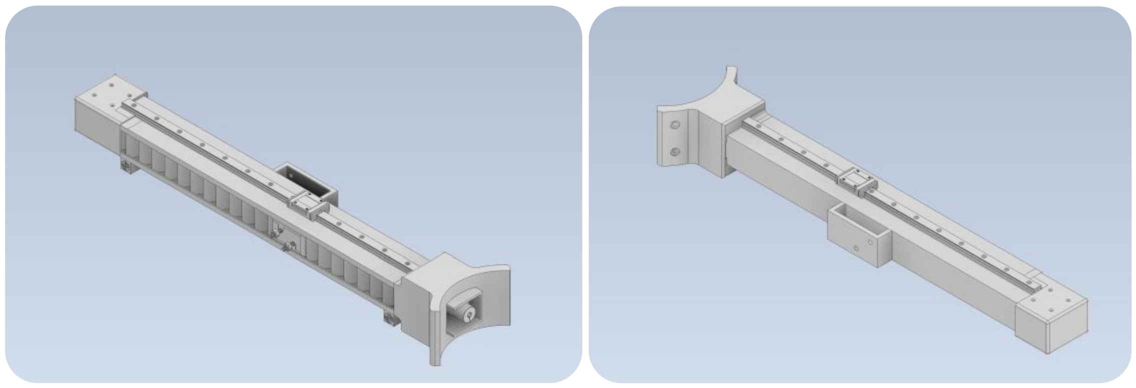


Figure 5.3: StakeBot's main arm (Front ISO - Left), StakeBot's main arm (Back ISO - Right)

The arms attach to the main body by the arm covers which are designed to wrap around the cylindrical shape of the main body. The end caps are designed to fit over the opposite end of the arms for added protection. These components were 3D printed with nylon glass-filled material.

The inner workings of the StakeBot's main arms are shown in Figure 5.4. Here the linear actuator-style arms include a lead screw, two lead screw holders housing ball bearings and the arm mover which pairs with the slider that holds the stake driving arms. These arms have a slider track and slider attached to the top of the aluminum C-Channel that moves with the arm mover. Attached to the slider will be the support arms for the stake driving arms (shown in Chapter 6) which will help to counter the moment forces that are generated by these arms during operations. The arm movers are restricted between the limit switches and the lead screw directly connects to the stepper motors by the motor coupling that are housed inside the StakeBot’s main body. The arm movers have flat bellows on either side that extend and retract with the mover and keep all the inside components protected. The arm mover and the lead screw holders are made from a 3D printed nylon glass-filled material. The lead screw is stainless steel, the slider track is a harden steel and the rest of the components are 6063-aluminum.

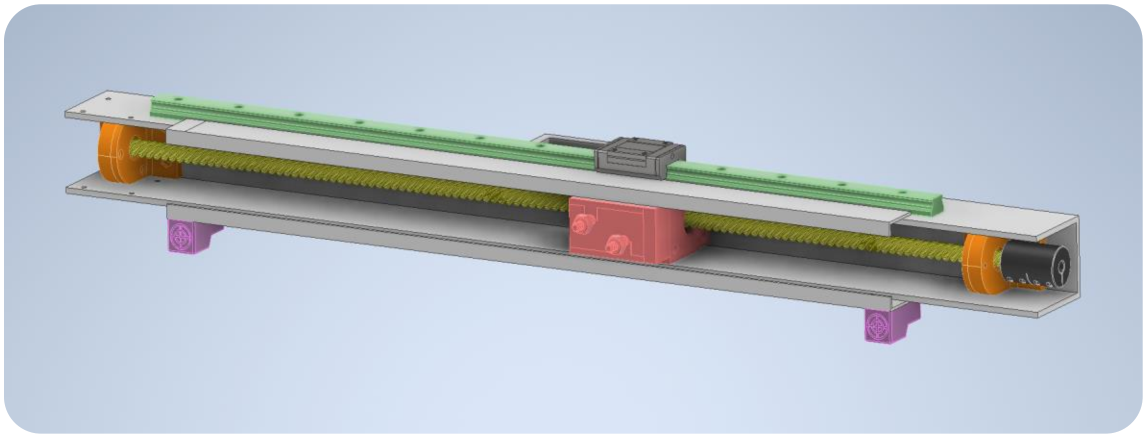


Figure 5.4: StakeBot's main arm inside view, showing the main components of the linear actuator. The main components are the arm mover (red), limit switches (purple), lead screw (yellow), lead screw holders (orange) slider track (green), slider (dark grey), and the motor coupling (black)

5.3 Main Body Functionality:

The StakeBot supports two main arms that hold the stake driving arms and is shown in Figure5.5. These arms move left to right together simultaneously which helps keep the StakeBot balanced. The StakeBot must be positioned directly between the bed plots when driving the stakes. This ensures that the stakes are driven in approximately the same location relative to their respected bed plots. The StakeBot’s AI is responsible for keeping this equal distance position as it moves down the bed plots rows. The AI does this by using the Intel RealSense sensor mounted to the front of the StakeBot’s base. (Details on how this works are discussed in Chapter 7) The purpose of the main arms is to allow the stake driving arms to retrieve a stake from the StakeBot, then to positions these arms over the bed plots where the stake driving arms can then proceed to plant the stakes.

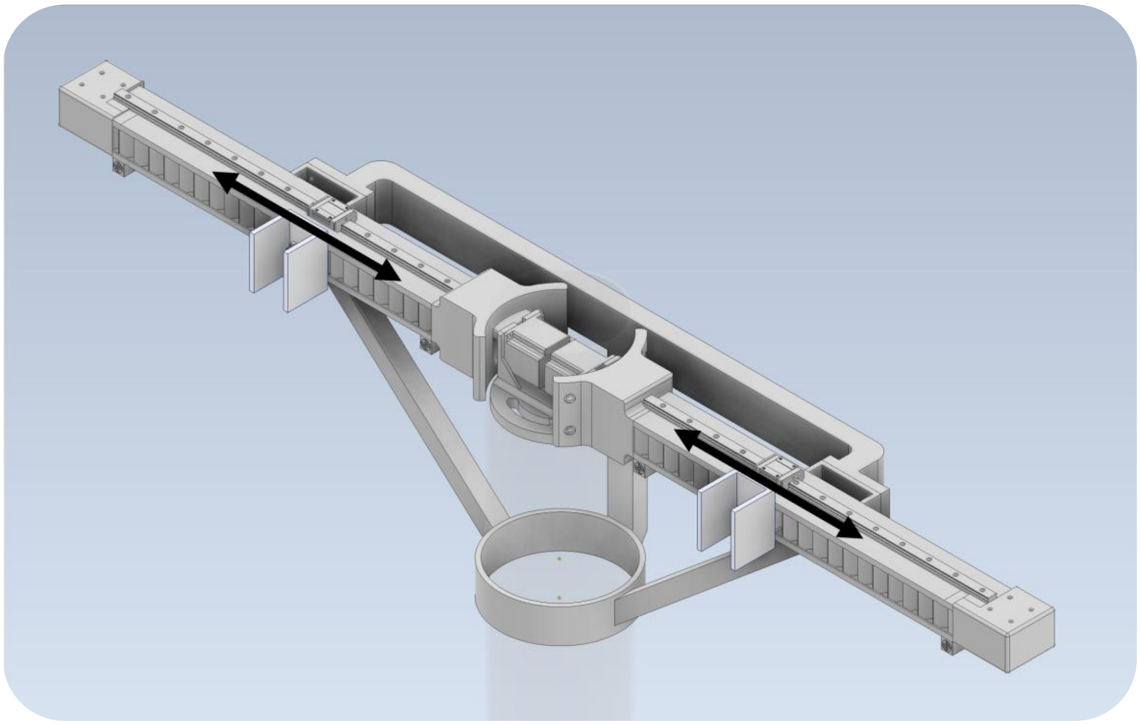


Figure 5.5: StakeBot's main arm functionality. The black arrow shows the direction in which the arm movers are allowed to traverse.

The main arms are controlled by stepper motors which allow for a precise calculated movement by the stepper motor driver. These motors ensure that the arms are always at a precise location. In addition to the stepper motor’s capability of keeping a precise location relative to the main arms, limit switches were installed that serve two major purposes. First, they allow the stepper motor driver the ability to calibrate the positioning of the movers. Second, the limit switches serve as a safety feature. In the unlikely event of a miscalculation by the stepper motor driver, if the mover attempts to move to a location outside the arms bounds, dictate by the location of the limit switch, the stepper motor driver will stop in the event the limit switch is triggered.

5.4 Main Body Electrical System:

Each of the main arms are controlled by a stepper motor that is bounded between two limit switches. The motors and switches for both arms are connected to the main arm stepper motor driver. The driver is responsible for controlling the movements of both motors independently for each arm. These movements are measured by a local coordinate system whose origin is at the limit switch closest to the StakeBot's main body cylinder. A coordinate frame of reference and the distance from that reference is stored on the driver. This is made possible due to the unique electrical capabilities of the stepper motor. Stepper motors are special DC motors that are controlled by electrical pulses provided by a host device, in this case, a microcontroller. Inside these stepper motors are two magnets whose power state alternates with each pulse. When a magnet is active the motor will rotate a set degree, this is called a step. It is possible to calculate the number of steps per distance by knowing the pitch of the lead screw. For the StakeBot's arms the lead screw's pitch is 1 inch meaning that one rotation will move the arm mover 1 inch. The number of pulses is also controlled by the driver, this is how the resolution of the distance traveled is set. For the case of the StakeBot, the number of pulses per rotation is set to 51,200. To achieve the desired resolution of 0.01 inches, the number of pulses per step is set to 512. The stepper motor driver communicates to the PYNQ-Z2 board the information on the location of the arm movers.

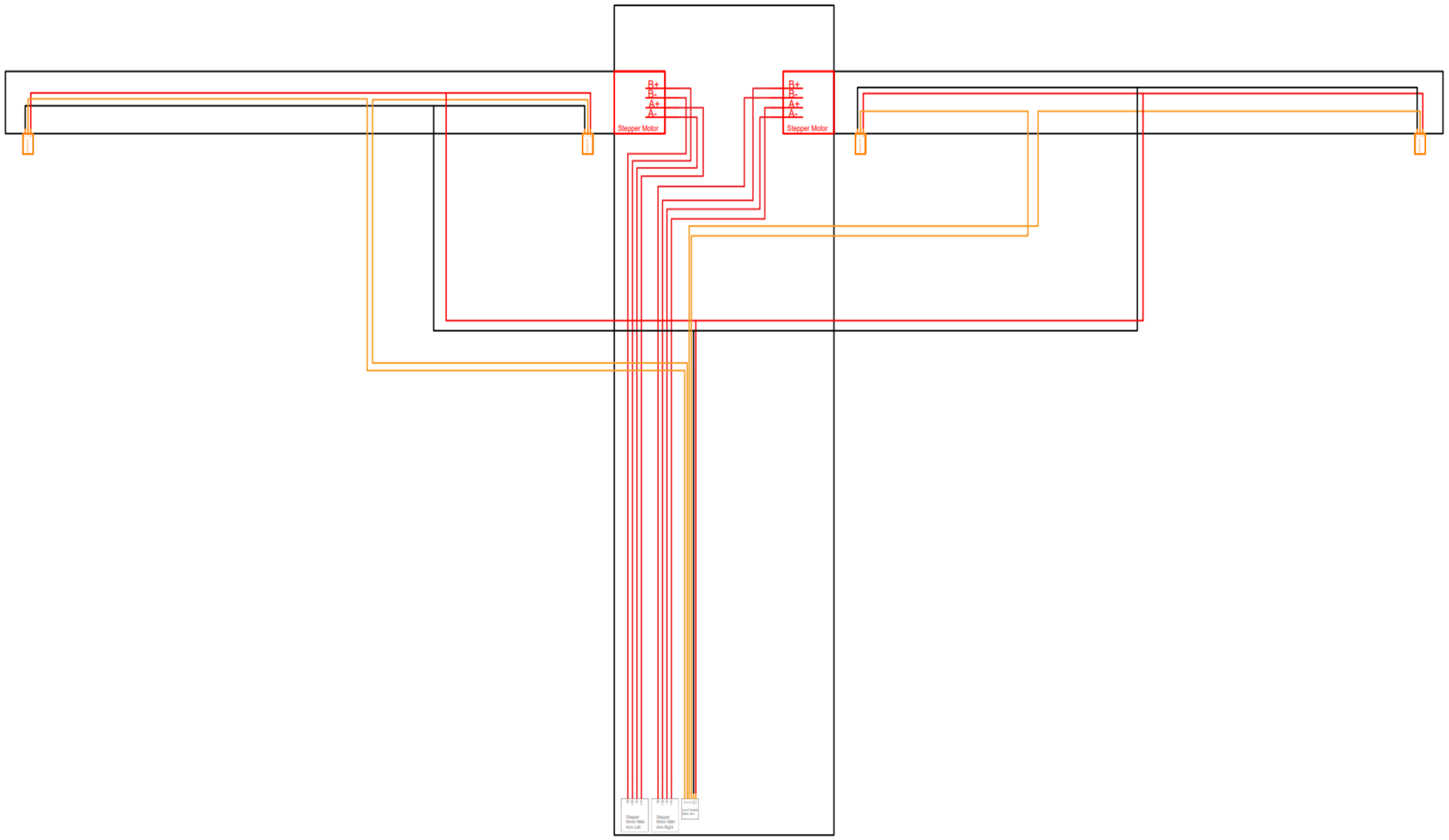


Figure 5.6: Main arm electrical diagram

They also receive instructions on which position to move to, with respect to their local coordinate system.

Shown in the wiring diagram above in Figure 5.6, the stepper motors and limit switches are wired into a three separate wire harness. These harnesses allow for the components to be easily removed in the event maintenance is required. There are only three unique plugs that are required for the stepper motor driver. The limit switches are all connected into one plug preventing the technician from installing the switches incorrectly on the driver. The motors are also labeled motor A and motor B at both the switch connections and on the motor driver to prevent any confusion when replacing the components if needed.

The StakeBot’s main arms are designed to move the stake driving arms to a desired location whose controls are handled by the stepper motor driver. The main arms are designed to handle the inertial forces and to support the stake driving arms. They attach to the StakeBot’s main body and are easily removeable if repairs are needed. Overall, the StakeBot’s main arms satisfy the requirements needed for transporting the stake driving arms to and from the StakeBot’s base allowing the stake to be driven in the correct location of the bed plots. The next component is the stake driving arm and will be the subject for the next chapter.

CHAPTER 6:

STAKEBOT'S STAKE DRIVING ARMS

6.1 Introduction:

The next component is the stake driving arms. These arms are responsible for driving the stakes into the bed plots. This is possible with the design of these arms and in the later sections, a detailed discussion on the design and control is given that shows the stake driving arm's capabilities and functionalities.

6.2 Stake Driving Arm Design:

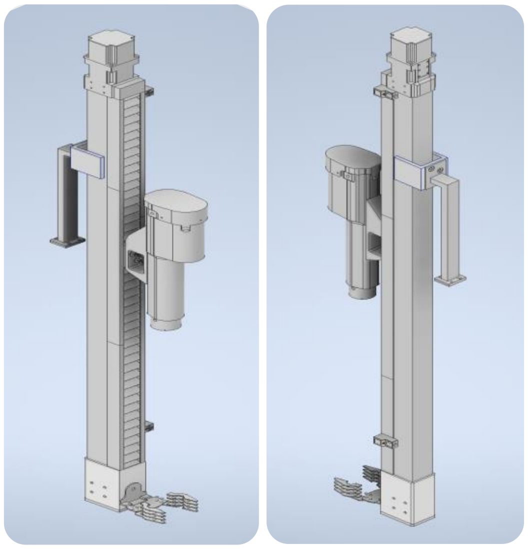


Figure 6.1: StakeBot's stake driving arm (Front ISO - Left), StakeBot's stake driving arm (Back ISO - Right)

As mentioned, the StakeBot's stake driving arms are responsible for planting the stake into the bed plots. The arm is similar in design to the main arms as both are electrical linear actuators controlled by a stepper motor. The stake driving arm attaches to the main arm movers and is supported by a steel arm attached to the back of the C-Channel. The arm has a removeable attachment on the front that the user can change depending on the type of operation needed. The arm also had an electric gripper used for grabbing and holding the stakes.

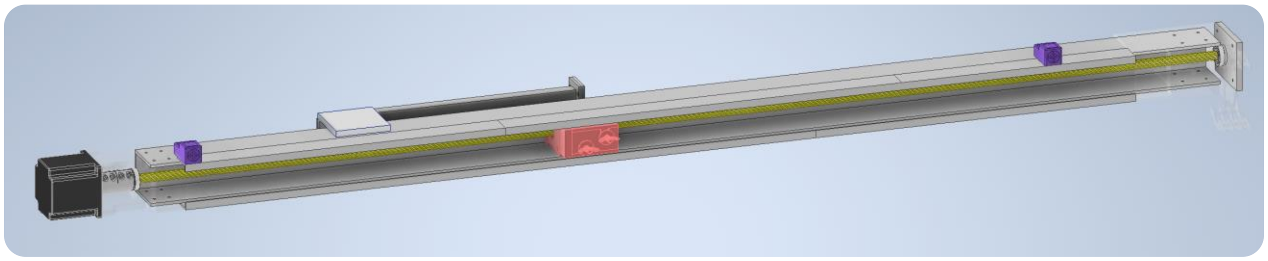


Figure 6.2: StakeBot's stake driving arm inside view, showing the main components. The main components are the lead screw (yellow), the arm mover (red), the stepper motor (black) and the limit switches (purple)

The main components are similar in design to the main arm. There is a lead screw connected to a stepper motor with the arm mover free to move between the bounds of the limit switches. The motor mount and the end cap (where the gripper attaches to) both house ball bearings that support the lead screw. The motor mount and end cap are made from a 3D nylon GF material, the lead screw is stainless steel, the arm support in the back is made from steel, and the rest of the components are aluminum.

The stake driving attachment is used for adding additional force during the stake driving operation. The attachment is an electric linear actuator design for around 25 lbs. of additional force and can hold both circular and square shaped stakes. In the event the stepper motors stall while trying to drive the stake, the linear actuator can provide additional force to drive the stake while not exceeding the arms force rating (discussed in

Chapter 3). The linear actuator is housed inside the linear actuator case that is designed to attach onto the stake driving arms and is made from a 3D nylon GF material.

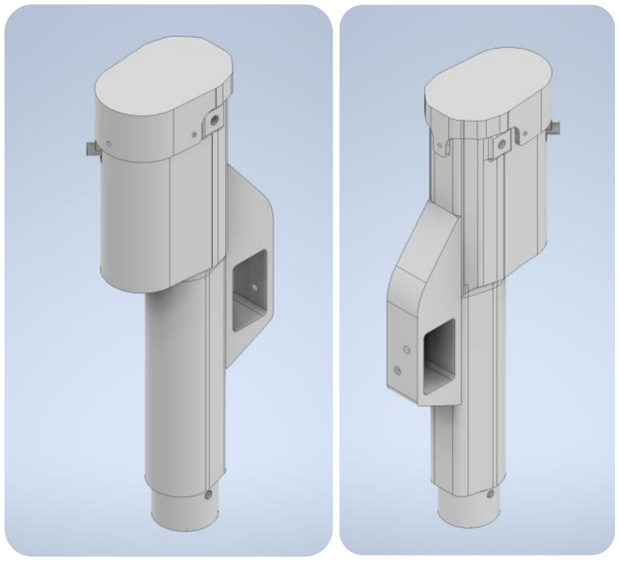


Figure 6.3: StakeBot's stake driver attachment (Front ISO - Left), StakeBot's stake driver attachment (Back ISO - Right)

These arms allow for the driving attachment to be easily replaced with different attachments that serve a different purpose. This allows for future expansion of other types of stakes manipulating operations such as a stake pulling attach for example. Currently the only attachment available is the stake driving attachment.

6.3 Stake Driving Arm Functionality:

For the StakeBot’s stake driving arms, a stake is grabbed by the gripper and the attachment lowers over top until it is firmly pressed against the stake. The attachment has a grove connected to the bottom that helps align the stake into the center of the attachment, directly underneath the linear actuator. With the gripper and the attachment over top of the stake, the main arms will then move the stake driving arms over the desired location for driving the stake. The gripper will slightly loosen its grip allowing the stake to slip through as the actuator pushes towards the ground. Then the actuator will

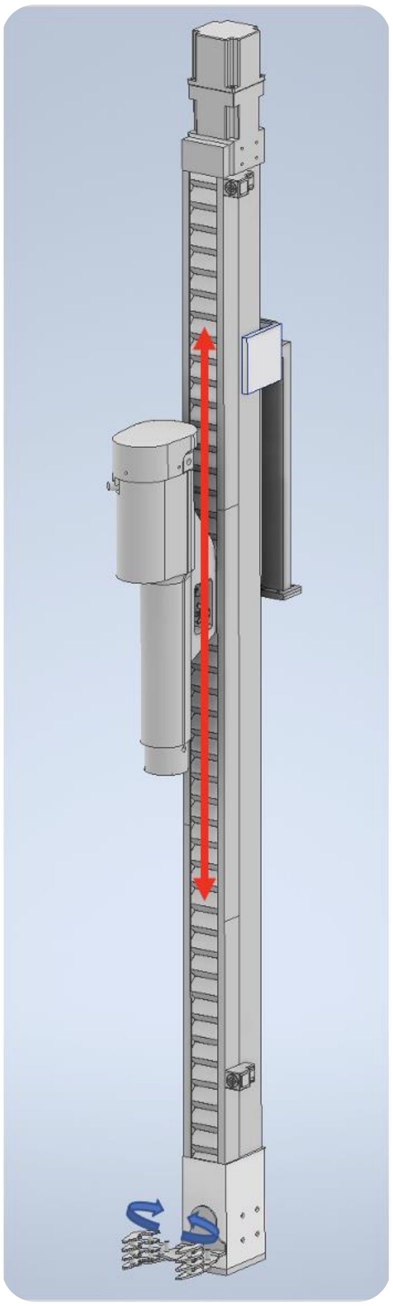


Figure 6.4: StakeBot's stake driving arm functionality. The red arrow shows the direction in which the attachment is allowed to move.

begin to drive the stake into the bed plot until it is at the desired depth specified by the user. If the stepper motor driving the arm begins to pull too much current, meaning that the force needed to plant the stake is too much for the arm’s stepper motor, the actuator will activate and will provide the extra force required to drive the stake. This current draw is monitored by the stepper motor drivers and is indicated to stop moving if the current is too high. This process keeps the stepper motors from experiencing stalls which can extend the lifetime of the arm’s stepper motors by preventing damage.

The stake driving arms functionalities are similar in design to the main arms where both use a stepper motor to move a mover back and forth. The difference here is the added functionality of the gripper and the attachment. All the moving components must work together to perform this operation. This operation is called a stake driving operation which is the action of driving the stake into the bed plot.

6.4 Stake Driving Arm Electronical System:

The StakeBot’s stake driving arms include four electrical components that need controls. Similar to the main arms the stepper motor and limit switches are controlled by a stepper motor driver. This stepper motor driver is the same type of driver as the one for the main arms, however, both are controlled by the PYNQ-Z2 board. The linear actuators are connected to an actuator driver which provides the corrected power output needed to extend and retract the actuator. The linear actuator attachment works similar to the main arms and stake driving arms where the DC motor controls the movement of a piston that is housed inside the casing and is bounded between two limit switches.

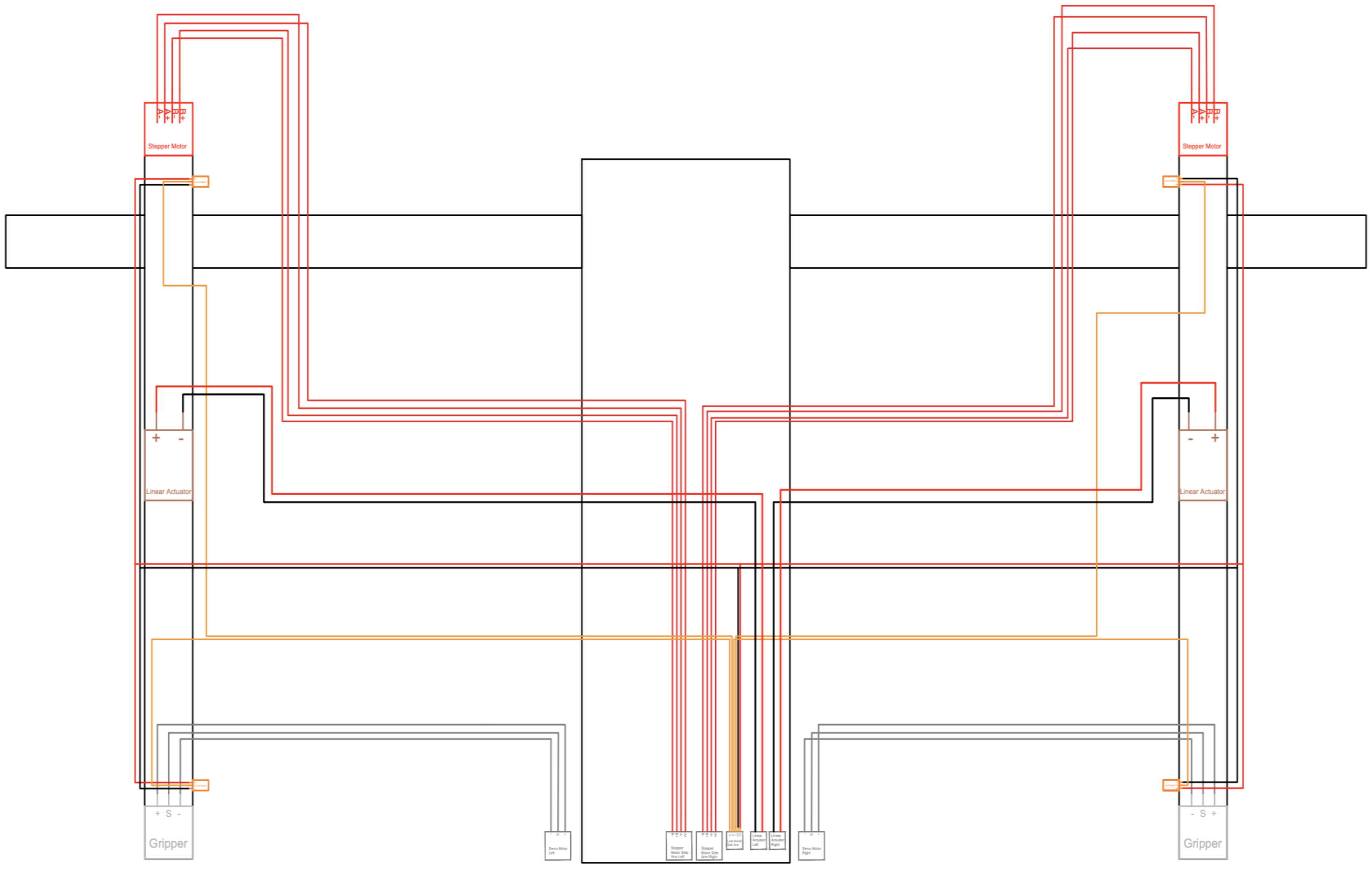


Figure 6.5: Stake driving arm electrical diagram

For the stake driving operations, the PYNQ-Z2 board runs an algorithm that will perform the driving operation. This algorithm is called stake driving operation and appropriately controls the necessary movement needed to drive the stake. This algorithm starts after retrieving a stake from the StakeBot, then it preforms the stake driving operation. The algorithm monitors the information coming for the stepper motor drivers where they will communicate the status of the movement of the stepper motors as the operation proceeds. As mentioned before, the current drawn from the stepper motors are monitored by the drivers, if a stall occurs, which happens if the current exceeds the steady state recommendation of the stepper motor, the motors will stop, the algorithm will then command the linear actuator attachment to continue driving the stake into the bed plot. After the stake has been driven into the bed plot, the algorithm ends and the PYNQ-Z2 board moves on to the next operation.

The StakeBot's stake driving arms are what make the StakeBot capable of completing its purpose. The controls systems utilize the components from the stake driving arm in a way that allows for the stakes to be driven into the bed plots. The force required to drive a stake into the bed plots were analyzed during a force experiment discussed in Chapter 3. The results showed that on average, the amount of downward force needed to drive a stake into the plant beds was about 6.20 lbs. of force. The decision to add the linear actuator attachment was to take account of all the unforeseen scenarios that could arise when driving a stake. Having a liner actuator capable of pressing with 25 lbs. of force is greatly suitable for most scenarios or possible outliners that may happen to require a significantly larger amount of force.

CHAPTER 7:

STAKEBOT'S VISION SYSTEM AND AUTONOMOUS DRIVING

7.1: Introduction to Autonomous Driving

The StakeBot is designed to be a fully autonomous vehicle capable of automatically driving support stakes into the bed plots. The term autonomous, as defined for the purposes of the StakeBot's functionality, is the capability of operating without any intervention from its user. By this definition, for the StakeBot to be autonomous, it must be able to self-navigate through its environment and self-operate as a machine capable of driving support stakes.

The desire to create a self-contained system that can operate autonomously stems from most of the local farmers lacking a labor force. Having the ability to pass the task of driving stakes to a machine is only beneficial if that machine can reduce the need for the number of workers dedicated to this task. If the StakeBot had to be manually controlled by a user, then there is no reduction in labor, just a different approach to achieve the same task. For the intended operations of the StakeBot, the idea is to assign one worker in charge of the StakeBot where they can monitor the behavior of the machine as it preforms the task of driving stakes and only intervene when a situation occurs that the StakeBot cannot overcome. These issues may be running out of stakes, bed plot row being blocked by too large of an object or running low on battery. With the StakeBot, a worker's responsibility can be greatly reduced as they would only have to monitor the

StakeBot’s performance during operation. They also could decide to split the work, having the StakeBot work one side of the field while they work another. Either way, the workload for the worker is now reduced, allowing the stake driving process to be completed faster and with less workers than traditional means.

With the advancement in technology over the past 20 years, computers have become faster and smaller, and sensors have become smaller and more reliable. For self-navigation capabilities, the StakeBot has been equipped with a vision system. This vision system allows the StakeBot to “see” and “perceive” the environment around it. Here, the concept stems from how a human uses sight as a way of navigating and perceiving different objects. This biological approach is possible by using an ordinary RGB camera for color detection and depth cameras for depth detection. For the StakeBot’s vision system, the Intel RealSense D435 camera was used as it is capable of streaming both RGB camera and depth camera images together at around 30 FPS.

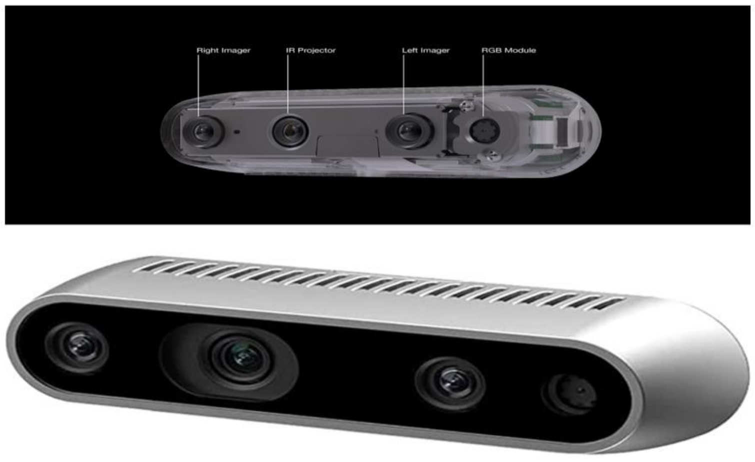


Figure 7.1: Intel RealSense D435 camera (Depth Camera D435, 2022)

The D435 camera uses stereo depth technology which is ideal for outdoor environments. As discussed in Chapter 2, the stereo depth method benefits from excessive lighting conditions and with the IR projector, this camera can reliably output depth images to the Jetson Nano in both high and low levels of light. This sensor is paired with the Jetson Nano and the images captured by the D435 sensor are further processed to aid in the understanding of the environment around the StakeBot. Finally, for complete navigation, the StakeBot also uses a GPS sensor that monitors it position offering its information to the onboard AI where it then commands the StakeBot when a specific distance has been reached.

7.2: StakeBot's Vision System

The StakeBot's vision system sets out to achieve the capabilities of "sight" that allow for self-navigation. The vision system was developed with a specific framework that would allow this system to not only work for the StakeBot, but also for any robot moving forward. This means is that the software used to analysis the incoming depth and color data from the sensor is modular and as long as the developer has the same hardware, they can use this system to develop autonomous software for their robot.

In Figure 7.2, the framework shows how a series of image processing algorithms are organized across the software that was developed for robotic vision. Here the algorithms are split into two main categories, depth data and color data. These categories help organize the type of image processing algorithms that exist within the framework. This framework allows for additional methods to be implemented that are used to define certain attributes. These attributes are the results of the type of method used for processing an image and are accessible across all parts of the computer. For example, the closest object depth method will store information about the depth of the object that is closest to the sensor within the attribute called closest object depth.

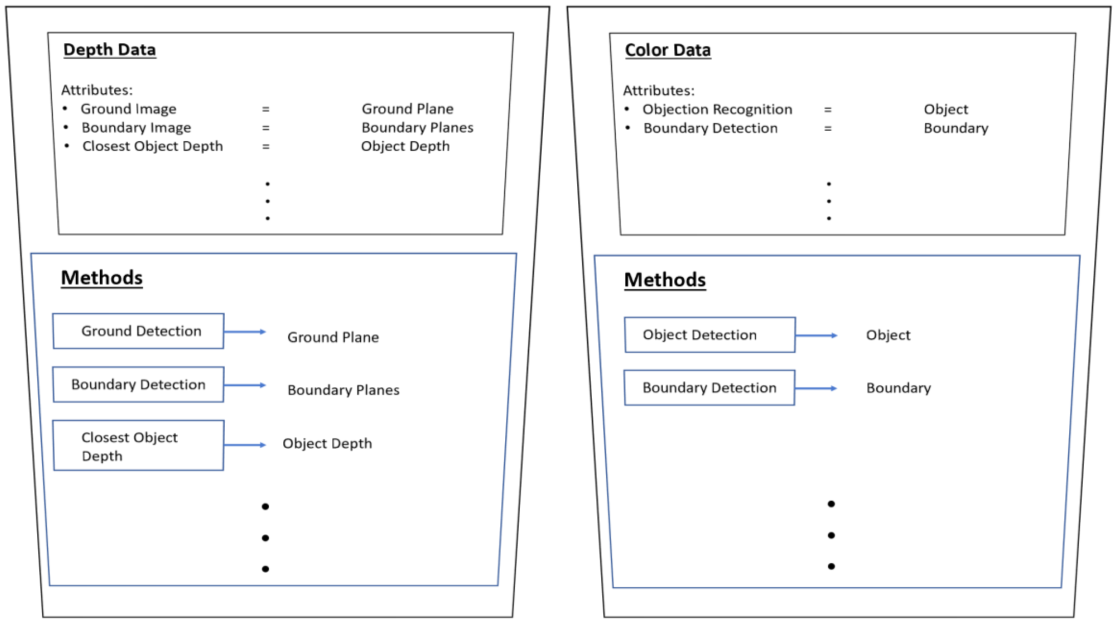


Figure 7.2: Vision system's framework at its current state.

This framework allows the developer to stack these various image processing algorithms to help shape the type of image that would allow their robot to gain a better understanding of its environment.

The order of processing algorithms for the StakeBot are shown in Figure 7.3. As depicted in Figure 7.2, these algorithms are organized in a framework and can be used in any order for both depth and color data. The StakeBot's series of image processing starts with ground detection where the ground is detected within the depth image, then that image is passed to the find boundary algorithm and so on.

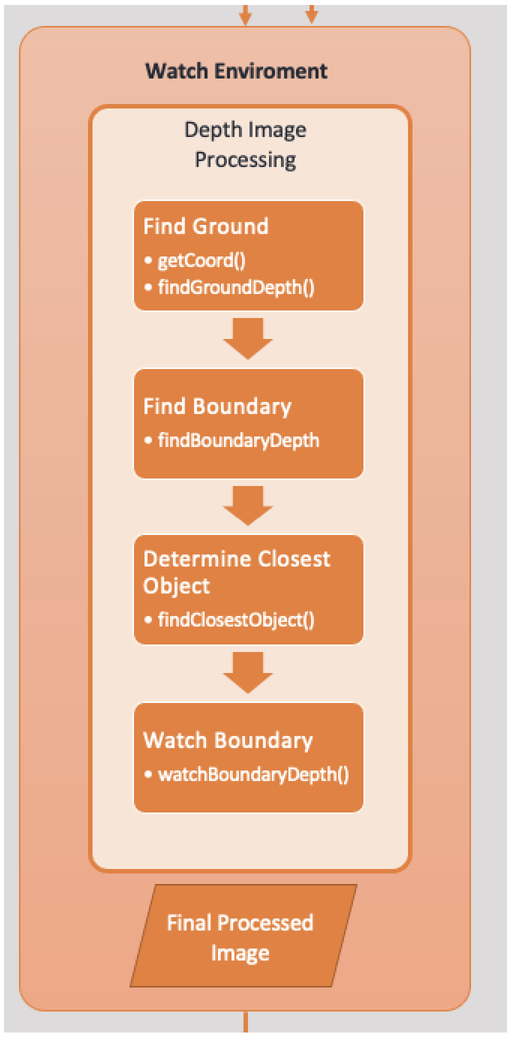


Figure 7.3: The current stack of algorithms that the StakeBot currently uses for processing the depth images.

In the next few sections, these algorithms are discussed in greater detail where the end goal is to shape the depth image and extract the necessary information needed for self-navigation.

7.2.1: Ground Detection Method

If thinking about what makes a person capable of navigating through a path, the first assessment is to look at the ground. The ground is our first clue as to whether navigation is possible. If the ground is clear of obstruction, then it is safe to proceed otherwise a different alternative path may be a better approach. It is with this ideology that will become the center of the vision system where the StakeBot will always have the ground in its view. Now, before any decision about the objects obstructing the path is made, first the distinction between the ground and the rest of the environment must be assumed before there can be any other assumptions about the path. This is the purpose of the ground detection method.

The ground detection method works by eliminating the ground from the view of the depth sensor to prevent the other methods from considering the ground as a potential object within the StakeBot's path. It does this by first creating a frame of reference which will help to better locate the position of each pixel of depth information by assigning an X, Y, Z value. This concept of ground detection and the method of distinction was inspired by authors Junji Eguchi and Koichi Ozaki, and their paper titled Extraction of drivable area using 3D range sensor. (J. Eguchi and K. Ozaki, 2016) In their paper, the authors demonstrate how they created a method called step-extraction which allows a robot to determine different elevations with respect to the ground. These different elevations are then used to determine if there is an up-step or a down-step along the path of the robot. This information allows the robot to determine if the drivable area has a cliff-like obstacle or a wall-like obstacle.

In the paper by Junji Eguchi and Koichi Ozaki, they explain their approach to develop their step-extraction method by first defining a frame of reference and then converting the depth data into a cartesian coordinate system. (J. Eguchi and K. Ozaki, 2016) Given the nature of the depth information, the X, Y and Z conversion follows the conversion formulas from a spherical coordinate system to a cartesian system. Shown below in Figure 7.4 is an illustration of the depth sensor data shown in a spherical coordinate system.

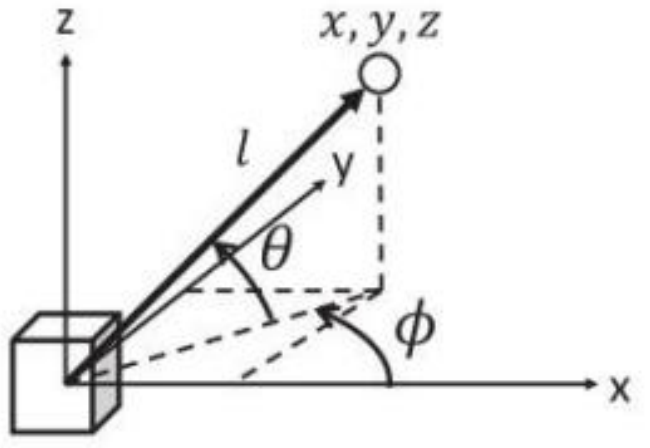


Figure 7.4: Representation of one depth value point on a spherical coordinate system. Given that the depth data is in millimeters, the units

For the conversion from spherical coordinates to cartesian coordinate, all the information needed comes directly from the depth camera. The value of l is the depth value at any pixel, the value of ϕ is the horizontal angel and θ is the vertical angle. The RealSense D435 camera has a field of view of 74 degrees in the horizontal direction and 62 degrees in the vertical direction. The orientation of the camera sensor is -5 degrees from horizontal. Given the resolution of the camera being 640 x 480, the ϕ and θ can be calculated depending on location of the depth value. With each spherical coordinate value known, the conversion to cartesian coordinates is shown in the equations below.

$$X = l\, cos(\theta)\cos(\phi) \qquad \text{Eq. 1}$$

$$Y = l\cos(\theta)\sin(\phi) \qquad \text{Eq. 2}$$

$$Z = l\sin(\theta) \qquad \text{Eq. 3}$$

These coordinates are stored in a matrix that can be accessed by any part of the software. These coordinates are calculated for each frame of the incoming depth data for the ground detection method, the Z values are analyzed to determine the ground. This analysis is done by identifying the minimal Z values within each of the depth data plus a small tolerance that can be adjusted by the developer. The values that fall below this threshold are then marked and are ignored. For demonstration purposes and for later debugging, the vision system has the ability for the developer to preview the depth data in a format that clearly conveys the depth information. This format is called color-mapping where the different depth magnitudes are given different colors that make previewing the images easy to understand. Shown below is a comparison between the same depth image that shows one image with ground detection and the other without.

The images shown in Figure 7.5 clearly represent the capabilities of the ground detection method. Notice that the ground is hard to identify in the left image as the walls blend into the ground surface. This can even make the distinction of where the objects start in relation to the ground harder to recognize. With the ground detection method, this distinction of the ground and everything else is very clear. The vision system can now better identify objects by making this distinction.

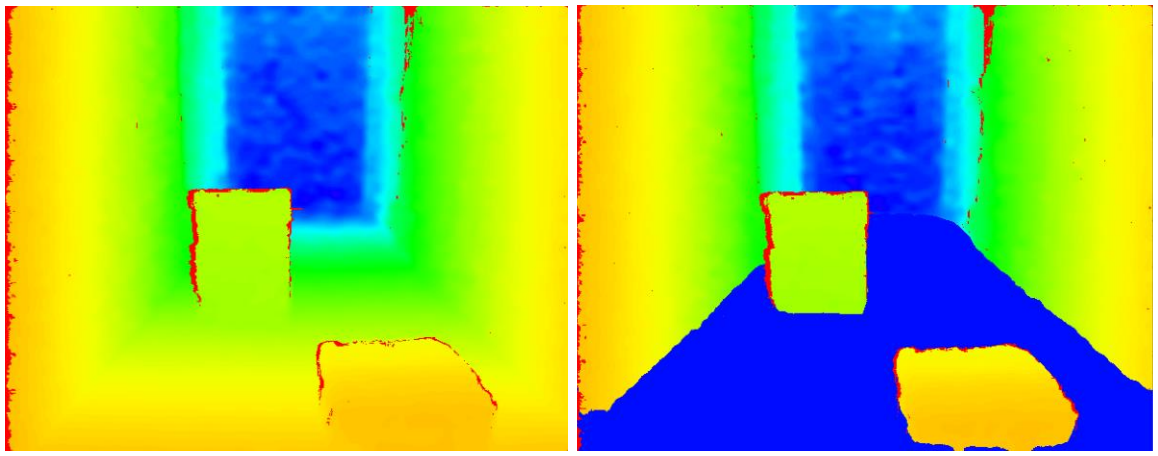


Figure 7.5: Vision system showing the color mapping of the depth values without ground detection, (Left) and with ground detection. (Right)

For self-navigation, understanding the ground defines the possible driving area for the StakeBot. Given this information, the next identification that needs to be distinguished would be the difference between the objects within the direct path of the StakeBot and the rest of the environment. For this method, the feature to extract would be the boundaries of the physical robot compared to the view of the camera.

7.2.2: Boundary Detection Method

The boundary detection method is used to better focus the other detection method on only parts of the view that are within the direct path of the StakeBot. Similar to how a human would assume whether they could physically make the journey pass certain objects. If the path is too narrow, then the path is no longer suitable to travel along. This is what the boundary detection method aims to achieve.

This method is simple as it allows the developer to define an area of focus depending on the physical constraints of the robot. This method is adjustable and depending on the robot’s size, only a certain view from the depth camera is necessary for monitoring. This method will eliminate the view outside of the focused area by defining a vertical and horizontal threshold. The vertical threshold will preserve the depth data by moving a plane on the X-axis from the center of the image out towards a distance on both sides from the center of the image and the horizontal threshold defines the distance from the top of the image down to a distance on the Z-axis. Figure 7.6 shows an illustration of how the boundary detection method defines the focused area.

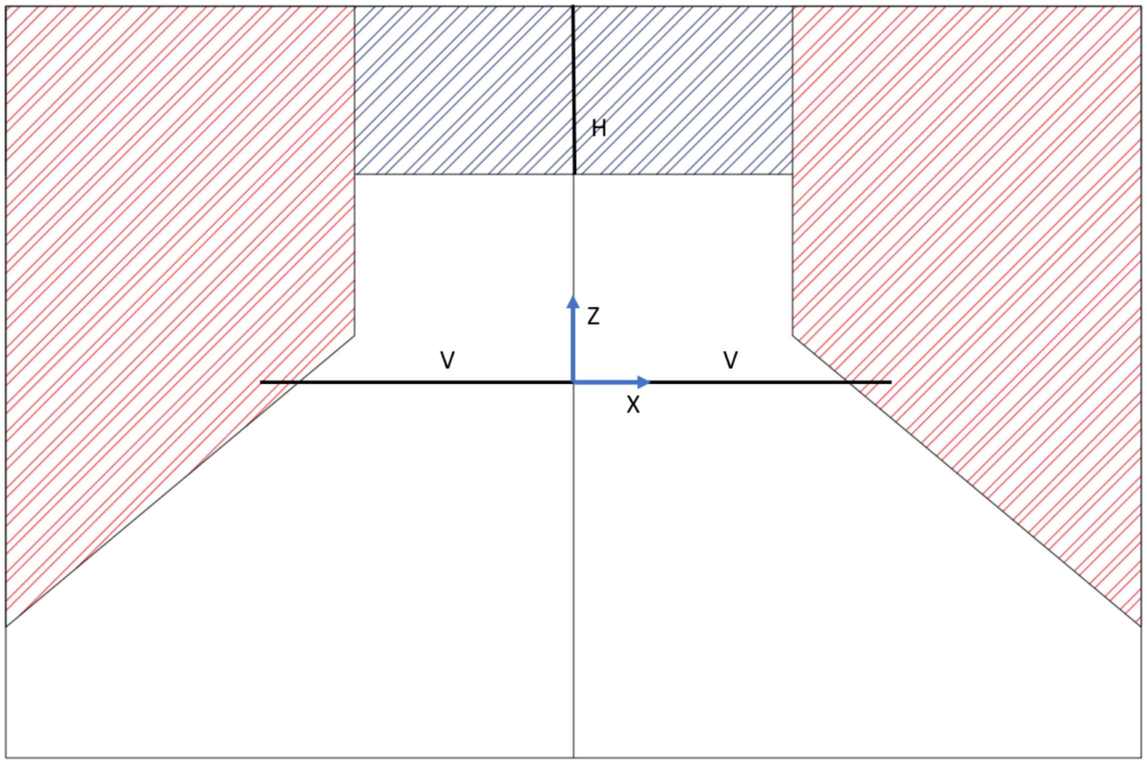


Figure 7.6: Boundary detection illustration. The V values represent a Y-plane’s distance from the center of the image to a distance on the X-axis. The H value is a Z-plane’s distance from the top of the image to a distance along the Z-axis.

For the StakeBot, the dimension of its width and height defines the area of focus for this method. With the camera being centered to the front of the StakeBot and the width being 18 inches, for the boundary detection method the V value is 228 millimeters (about 9 inches). This number was rounded up to 254 millimeters (about 10 inches), leaving about an extra half an inch beyond the physical bounds of the StakeBot on both sides. The H value was not set because the StakeBot is around 5 and a half feet tall which would put the horizontal plane beyond the measurable distance that the depth camera can measure. (Being over 18 meters) Shown below in Figure 7.7 is a depth image of the boundary detection method used for the StakeBot's vision system.

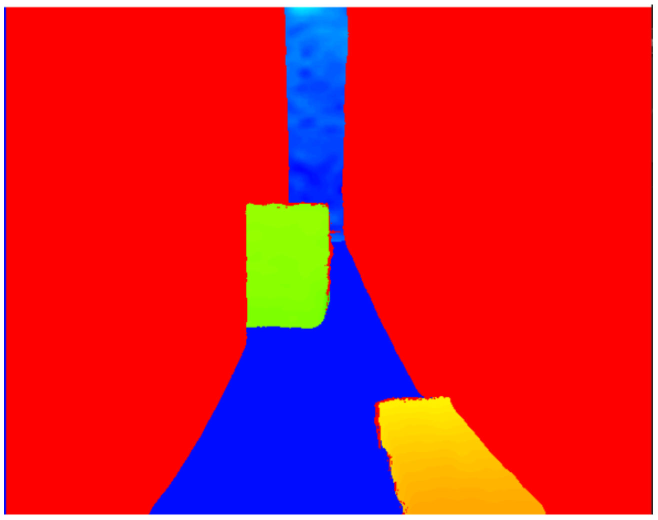


Figure 7.7: Vision system showing the color mapping of the boundary detection method.

Figure 7.7 shows how the focus of the vision system has been limited to only the area that would directly interfere with the StakeBot. Any objects within these bounds are potential obstacles that would impair the StakeBot from progressing on its path. As shown in Figure 7.7 there are two objects that are in the path of the StakeBot but only partially. This method makes finding objects within the path of the StakeBot more practical during its navigation. The final step is to now detect the closest objects within the direct path of the StakeBot.

7.2.3: Closest Object Detection Method

The final method used for self-navigation and the final step for processing the depth image is the closest object detection method. This method is the most intuitive to understand. When moving along a path, if an object is directly in the path one is traveling, then certain decisions need to be made in order to avoid the object. What is often important is understanding the distance away the object is in relation to the path being traveled. Having depth allows for decisions such as when to slow down or which direction to begin turning. All of the previous methods are designed to help increase the accuracy of object detection within the immediate path of the robot. This detection is important as it will directly influence the movement behavior of the StakeBot.

The closest object detection method simply returns the minimal depth pixel for each frame and the location of the pixel within the view. With higher resolutions these pixels usually are of a large quantity but only show a small portion of the object. Due to having many pixels sharing the same depth value, the method will return values in an array of every pixel location sharing the same minimal value. This is done for convenience to the developer where potentially a better location could be extracted from each pool of pixels. However, if the resolution is high enough, these pixels will be close enough together that only the first index is necessary to achieve a high enough accuracy to the location of the closest object. Shown in Figure 7.6 is an example of the closest object detection method being used to detect the object that is both the closest to and within the path of the StakeBot.

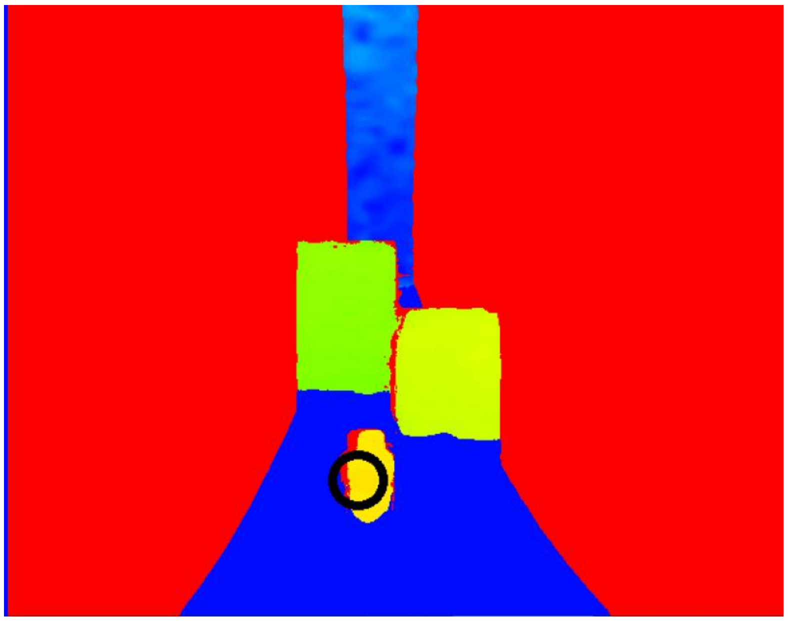


Figure 7.8: Vision system showing the color mapping view with a black circle around the object that is the closest to the depth camera. This is made possible by the closest object detection method.

At the vision system's current state, these methods are the only ones used for self-navigation. In the next section, a discussion on how the utilization of these different captured features can be used in conjunction with the motor controls to achieve self-navigation autonomously.

7.3: StakeBot's Autonomous Capabilities

The ability to achieve autonomous driving comes from the ability to mimic the human thought process. While the vision system was developed with this fundamental ideology, the driving controls further take this ideology in the form of basic artificial intelligence. If a human is driving down a street and sees a stop sign, intuitively they would begin to decelerate the car at a specific distance based on their knowledge of driving and their perception of depth. The vision system brings the perception of depth to the StakeBot but now there needs to be a level of decision making that is done based on the speed and the StakeBot's distance away from the approaching obstacle. The driving controls (later descried in Chapter 8) were developed around the ability to constantly monitor the objects' distance and speed. This allows the StakeBot to automatically adjust the speed of its motors when approaching an object. If the user was to intentionally drive the StakeBot into a wall at full speed, the StakeBot will automatically detect the wall as an obstacle and will begin to slow down as the StakeBot gets closer. If the StakeBot gets too close it will stop before it ever collides with the wall to prevent a collision. This feature is hard programmed into the driving controls allowing as a safety measure for the user or an AI while driving the StakeBot.

The driving controls pair with the vision system to create a safe driving experience but for self-navigation, an AI must be in control if the StakeBot is to drive autonomously. To achieve this, the StakeBot uses its on-board systems to make decisions on how to proceed down a path. For the StakeBot it's important to understand how the path looks and to try finding specific barriers or obstacles that could help make driving down the path easier. Fortunately for the StakeBot's case, the path is very straight and is confined by bed plots on either side. These bed plots are 8 inch in height which gives a good barrier for the StakeBot to see. The idea here is for the StakeBot to travel through these rows while staying directly centered between them. For this, an AI uses the vision system's closest object detection method to monitor the left and right bed plots to ensure they stay equal distance from each other while driving through the bed plot rows.

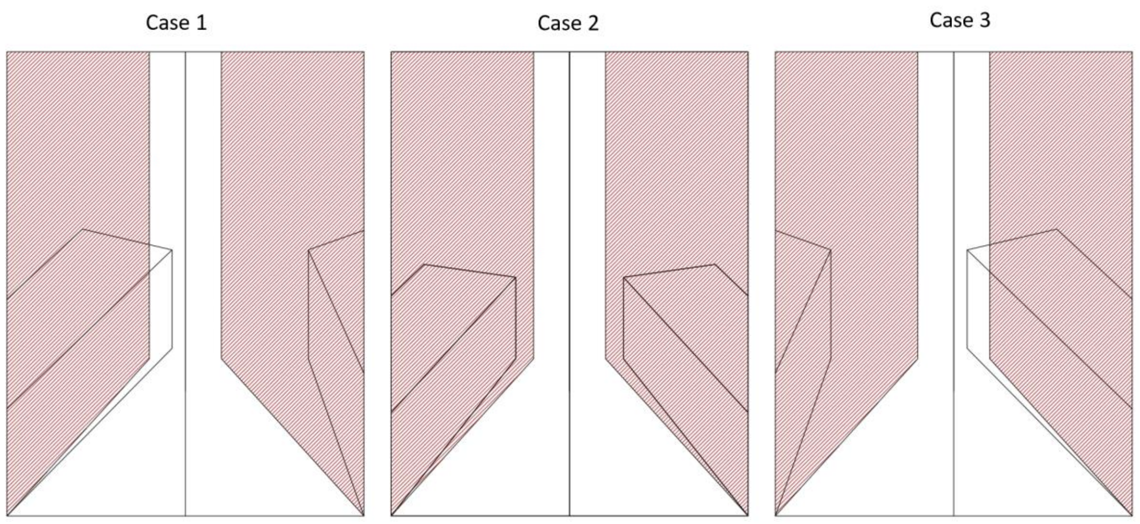


Figure 7.9: Illustration of the three different cases experienced by the StakeBot when navigating through the bed plots.

Shown in Figure 7.9 are the three different visual cases that the StakeBot sees while navigating illustrating how self-navigation through the bed plots work. By using the vision system, similar to how the closes object detection works, as the bed plots enter the focused area defined by the vision system, if an object is present on one side of the StakeBot's view of the focused area and not the other, then it will turn away from that side. If there are two objects on both sides within the focused area, then the StakeBot cannot proceed and finally if there are no objects on either side of its focused view then the StakeBot will continue to move forward. This algorithm also monitors the distance at which the object enters the focused view and the object's depth from the StakeBot. This distance will determine how fast to turn away from the object. Objects that are further away do not require a sudden adjustment to the StakeBot trajectory while objects that are closer will cause the turning speed to increase. Shown in Figure 7.10 demonstrates the vision systems capability of detecting objects from the left and right side of the center of the view. This image clearly shows the objects that are detected are objects closest to the StakeBot.

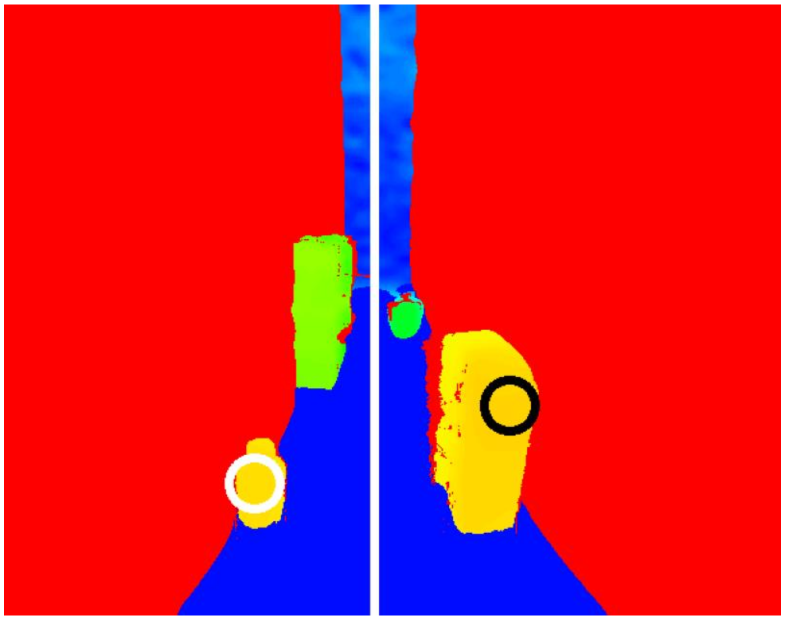


Figure 7.10: Vision system's closest object detection method used to detect objects on both sides from the center.

This algorithm allows the StakeBot to make decisions based on different objects' position with relation to the StakeBot. The StakeBot is designed to travel down the bed plot rows where it will stop at every specified distance to plant a stake. This distance is set by the user depending on the job that was created while using GPS to know how far it has traveled. The decisions are made by the PYNQ-Z2 board which gives commands to the Jetson Nano telling whether it should continue moving or stop. The algorithm that keeps the StakeBot centered between the bed plots is programmed on the Jetson Nano. In Chapter 8, further details on the control actions will be discussed. The PYNQ-Z2 must communicate when the StakeBot needs to stop and plant a stake and when it needs to continue moving forward. Certain safety algorithms are hard coded into the Nano's driving controls and if the PYNQ-Z2 board was to give a command to move forward while there are objects in the StakeBot's path then the Jetson Nano will communicate an error message that the PYNQ-Z2 board will understand and then will proceed to inform the user of the error.

The StakeBot is capable of self-navigation and can drive stakes into the ground which completes its primary purpose. With the design of the framework for the vision system more methods can be added for further feature extractions depending on the needs of the robot. This allows for expansion of the capabilities of the vision system where there is a possibility of creating a library of methods that take advantage of both color and depth cameras that allow for easy implementation across any style of robot.

7.4 StakeBot's Digital-Twin:

The development of the control system for the StakeBot was a challenge that required the use of simulation techniques. A digital-twin simulation provided a virtual world to test the algorithms to ensure a safe and successful development process. The purpose is to construct a virtual replica of the StakeBot with the real-world algorithms applied. This allows a safe and sufficient way of testing the StakeBot where potential coding bugs and other issues can be fixed before its final upload to its real-world counterpart.

With this method, the simulation tool that was used was a software by Nvidia called Issac Sim. Issac Sim is a robotic simulation tool used for developing and training robots in a virtual world with software that mimics real world physics and allows the user to program their virtual robot using the python coding language. (NVIDIA Developer)

For the case of the StakeBot, a simulation suite was created that allows for testing the different algorithms that the real-world StakeBot uses.

The first set of algorithms tested was the driving controls. Driving controls are fundamentally a major component in the control system. Within Issac Sim, the simulation suite has driving controls simulated that will load the StakeBot's base model into the virtual world where the developer can test the different driving modes. This can seen in Figure 7.11. In this simulation, the StakeBot takes in user controls through a joystick and simulates the StakeBot's movement behavior. It was with this developmental process that allowed for faster development times and for a safer means of testing. As coding bugs were found, the driving controls may have behaved in an unexpected manner which sometimes caused the StakeBot to drive out of control. Having a simulated environment ensures that no real damage was done. The efficiency of developing software in the virtual world also decreases downtime where typically the robot would have to be reset to a starting point or would have to undergo frequent updating which takes time and slows down the development process.

Issac Sim has many features that allow for further simulation and even for machine learning. For the vision system, within Issac Sim the Intel RealSense camera can be emulated where the vision system algorithms can be directly tested. Issac Sim also has the capabilities of training through machine learning algorithms where the user can run an AI while subjecting the robot through many different scenarios where it can eventually learn how to behave within its intended environment. This ideology is how the StakeBot moving forward will learn self-navigate through even more complex scenarios making the AI even more advanced.

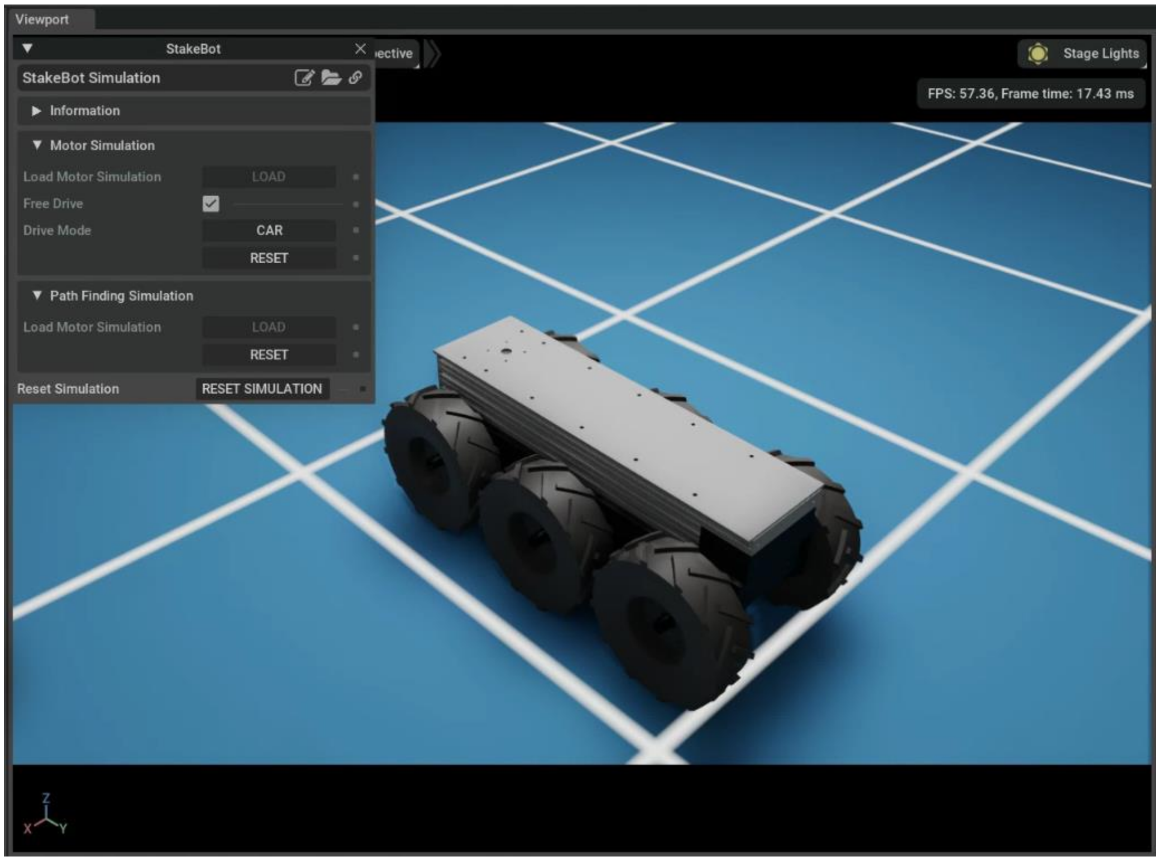


Figure 7.11: Digital-Twin of the StakeBot's base shown in Isaac Sim

The digital twin made the development of the StakeBot faster and smoother by allowing immediate code adjustments to be made. The StakeBot's driving controls and vision system were simulated using Issac Sim and will continue to be improved as the robot moves further into machine learning. Eventually, the StakeBot will be able to master the agricultural environment by making use of the vision system's algorithms paired with its driving controls where it will utilize machine learning to better navigate through its environment.

In Figure 7.12, this model shows how the entire StakeBot would look within Isaac Sim. This model, similar to the model shown in Figure 7.11, is a digital twin model programmed to act within Isaac Sim's virtual world. As future work progresses this model can be used to further test the other operations of the StakeBot and can be used for training the AI.

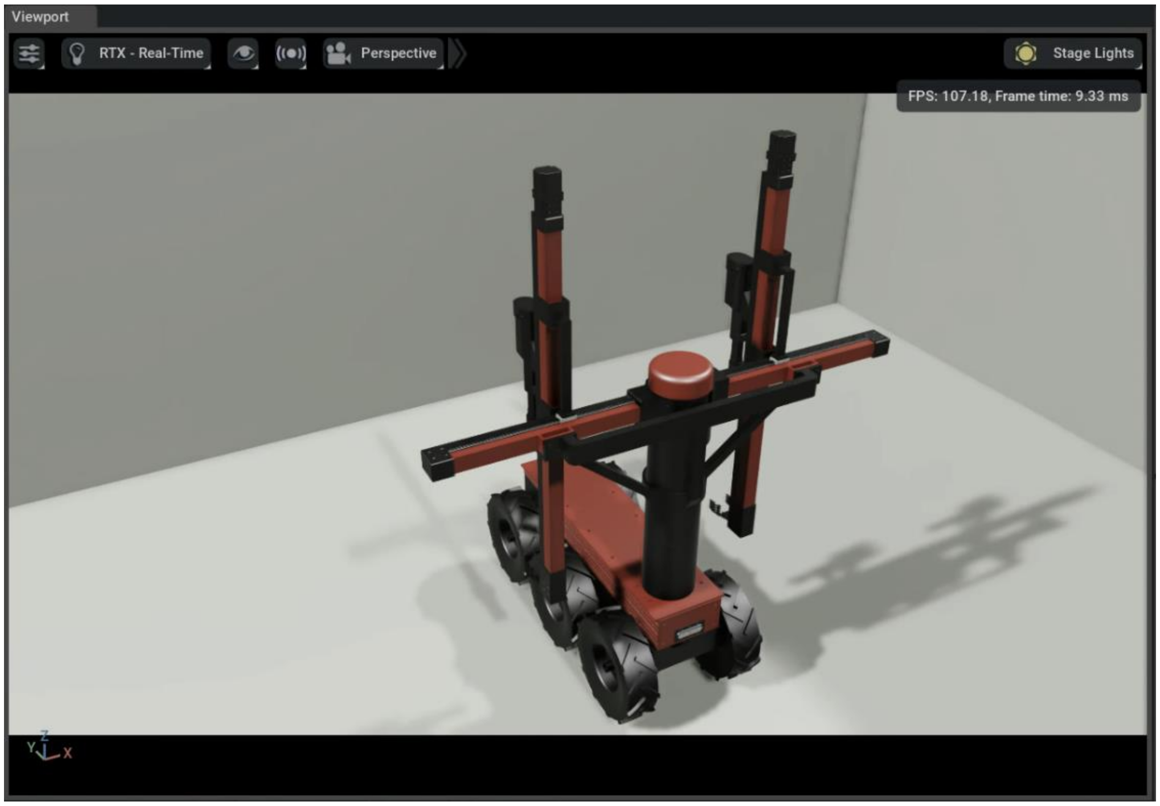


Figure 7.12: Digital-Twin of the StakeBot within Isaac Sim.

Overall, the StakeBot's digital twin acts as a tool used for further developing the StakeBot. Finally, the current StakeBot's development process will be discussed in the next chapter.

CHAPTER 8

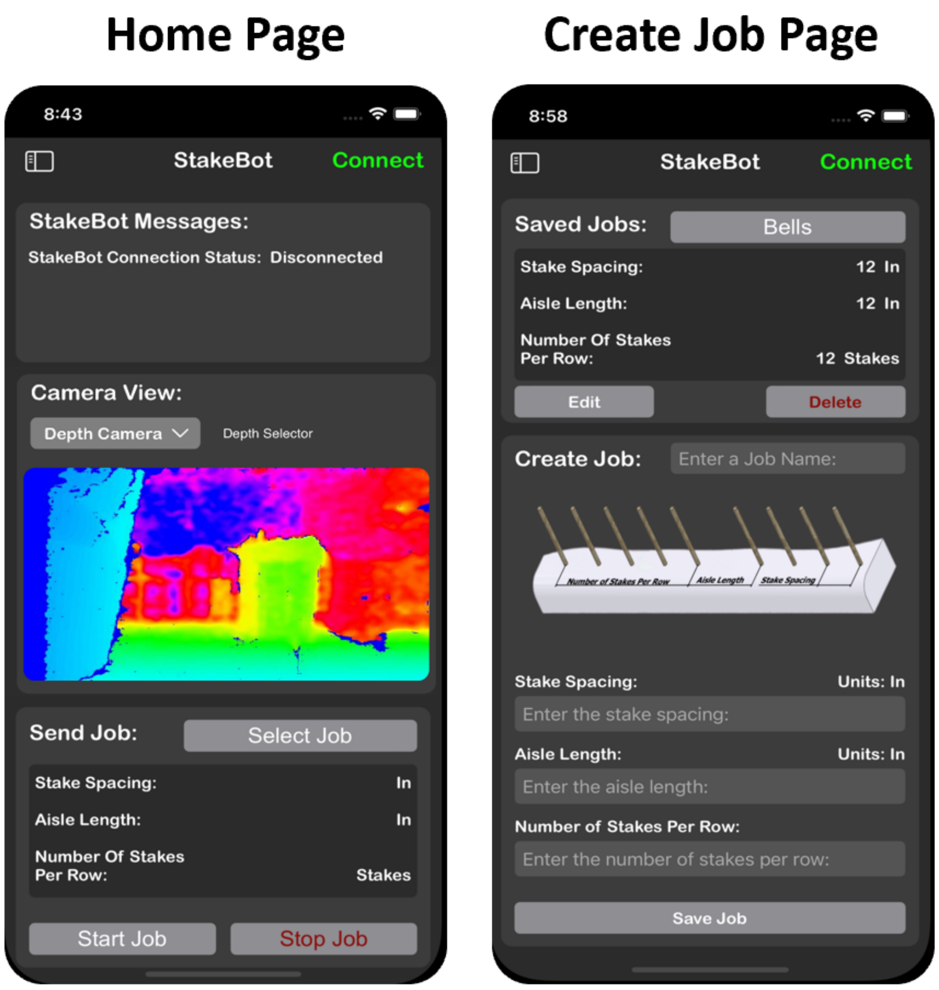
STAKEBOT'S SYSTEMS

8.1 Introduction:

The StakeBot's development utilizes all of the details discussed in the previous chapters to perform its intended operations. The StakeBot currently can perform self-navigation through bed plots using the StakeBot's autonomous capabilities. It can also be controlled by a user or by the onboard AI through the commands of the user through a mobile application. In the later sections of this chapter, the current controls flow and behaviors of the StakeBot will be discussed showing its current capabilities and functionalities.

8.2 Controlling the StakeBot:

A mobile application was created to provide the user with a means of controlling and communicating with the StakeBot via WIFI connection. The Jetson Nano hosts a network access point called StakeBot where the user can connect to with their mobile device. Shown below is the application that provides the user control over the StakeBot. The application starts with displaying a home page where the user can connect to the StakeBot and view messages regarding the status of the StakeBot. This messages block provides a way for the StakeBot to communicate to the user any issues with connection or status on the proceeding job. For example, if an obstacle was blocking the StakeBot from proceeding, this alert will appear here.



StakeBot Messages:
StakeBot Connection Status: Disconnected

Stake Spacing:	In
Aisle Length:	In
Number Of Stakes Per Row:	Stakes

Saved Jobs: Bells

Stake Spacing:	12 In
Aisle Length:	12 In
Number Of Stakes Per Row:	12 Stakes

Stake Spacing: Units: In
Aisle Length: Units: In
Number of Stakes Per Row:

Figure 8.1: StakeBot's mobile application. Showing the home page and the create job page.

The home page also allows the user access to the on-board camera through the camera view block. Here the user has the ability to see both the RBG camera view and depth camera view that the StakeBot sees. Finally, the last block is the send job block. This block is how the user can begin the StakeBot's operations where after the user selects a job preset that they create in the create job page, after the user presses the start job button, the StakeBot will begin to operate autonomously and will proceed to navigate through the bed plots driving stakes. The next page is the create job page where the user has the ability to create different jobs for the StakeBot. These jobs allow the user to create presets for different fields that have different staking parameters such as stake spacing, aisle length and number of stakes per row. As described in the design requirements in Chapter 3, the ability to create different patterns is done through the app. The stake spacing is the distance between each stake, the number of stakes per row and aisle length describe this pattern of how many stakes are places before a break is set. A break or aisle is created for the farmers to have a space to cross over the bed plots. This spacing and number of stakes between each spacing can be set through the app.

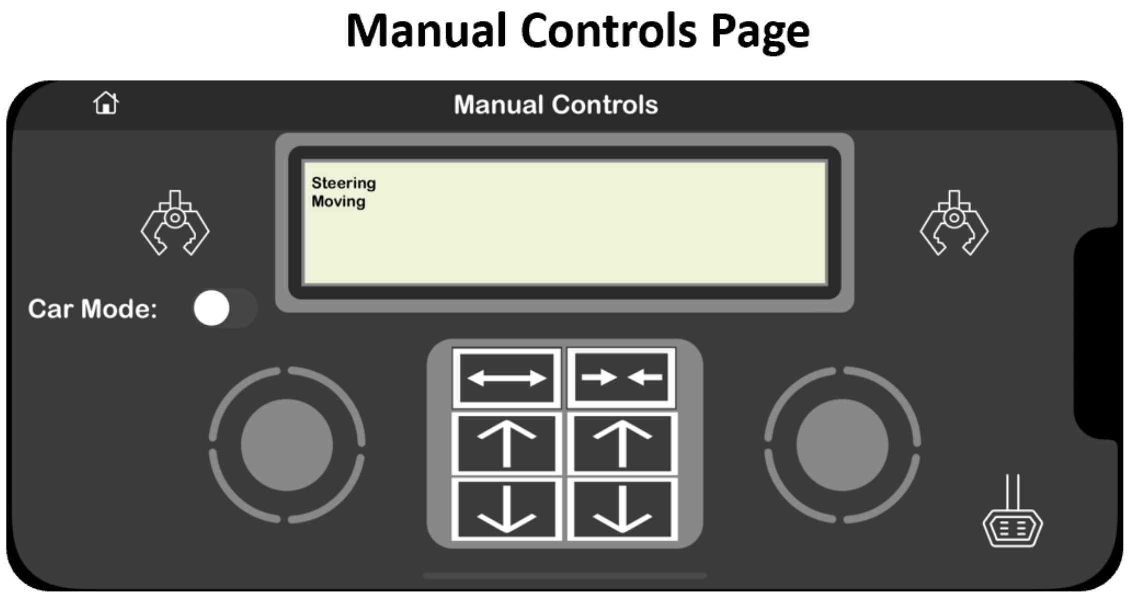


Figure 8.2: Manual page of the StakeBot's mobile app

The last page of the StakeBot's mobile app is the manual controls page. Here, the user has the ability to manually drive the StakeBot where they can move the StakeBot to any location on the field. The app provides two different driving modes, joystick control, and status window and buttons for controlling the grippers and arms.

8.3 StakeBot's Control Flow:

The StakeBot's controls are split between the Jetson Nano and the PYNQ-Z2 boards. For the Jetson Nano, the software is illustrated through a flow diagram shown in Figure 8.3 and the PYNQ-Z2 flow diagram is shown in Figure 8.4.

The Jetson Nano is split into 5 threads that run simultaneously, providing separate sections of controls but all communicate information to each other. The first thread to discuss is the communications thread which is responsible for keeping communications between other devices open and then storing these messages into corresponding python dictionaries for other threads to view. The next thread is the thread manager which is responsible for providing the switch from manual to autonomous modes depending on instruction sent from the application or the on-board AI. This thread also is responsible for monitoring the battery levels of the StakeBot by measuring the batteries voltage output. Another thread is responsible for the differential driving. Here this thread works together with the vision system, as mentioned in chapter 7, where it heavily influences the driving decision made by the differential drive thread. The logic inside this thread states that if an object is within 600 and 400 millimeters, then the StakeBot's movement will decrease until the object is less than 400 millimeters away, then it will stop otherwise it will follow the instructions given by the other threads. These instructions on how to move come from either the autonomous thread or the communications thread as messages from the app. The autonomous thread runs the logic behind the StakeBot's self-navigation algorithm. Once the StakeBot is in autonomous mode, which can be set by the user of the on-board AI, the StakeBot will follow the procedures mentioned in Chapter 7 for self-navigation. Finally, the last thread is the vision systems thread where the camera sensor images are processed and stored in attributes for other thread the view as described in Chapter 7. These threads begin when the Jetson Nano first boots up and will continue to run indefinitely until the system shuts down.

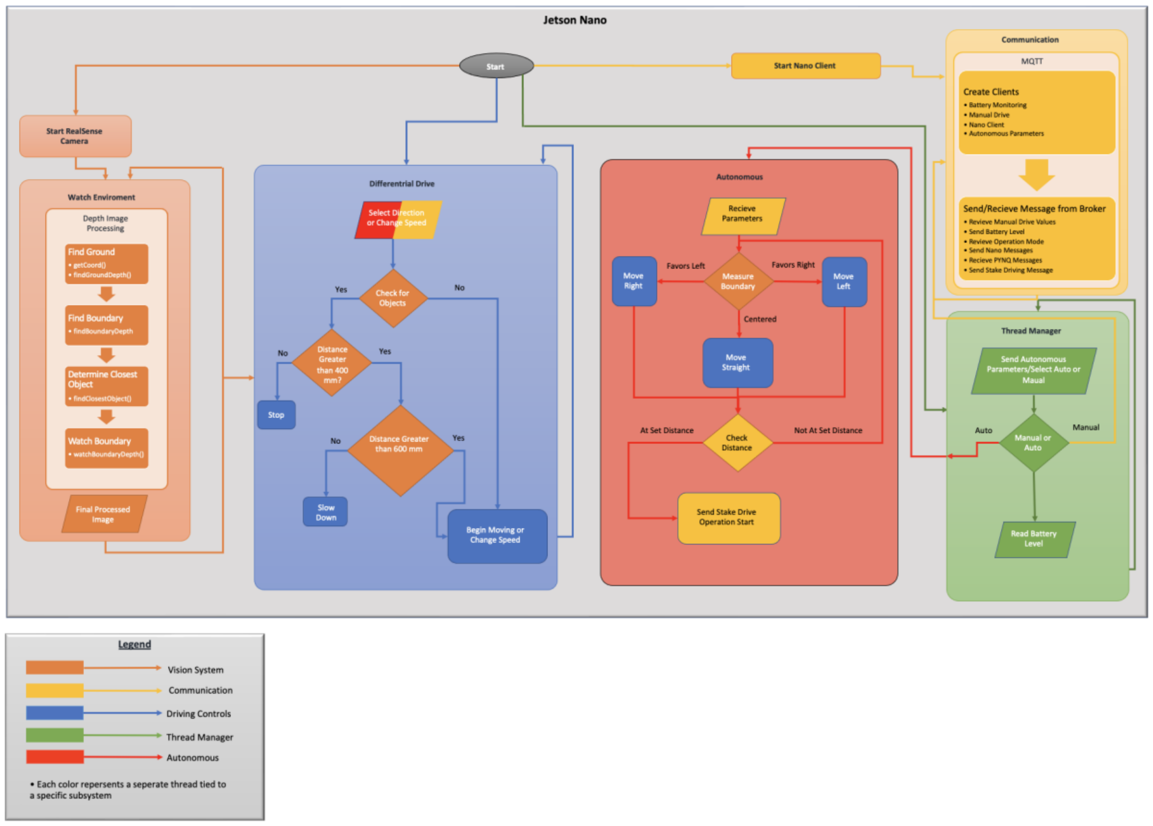


Figure 8.3: Jetson Nano processing flow diagram

The PYNQ-Z2 board continuously runs three separate threads that, like the Jetson Nano, are responsible for sperate sections of controls but all communicating information to each other. The first thread to discuss is the communications thread, similar to the Jetson Nano, which is responsible for handling communications between the different devices where the messages are stored in python dictionaries for other threads to access.

Another thread is the operations control where its responsibilities are to collect information from the various on-board subsystems, including the GPS sensor, and will give instructions to the Jetson Nano on how to proceed.

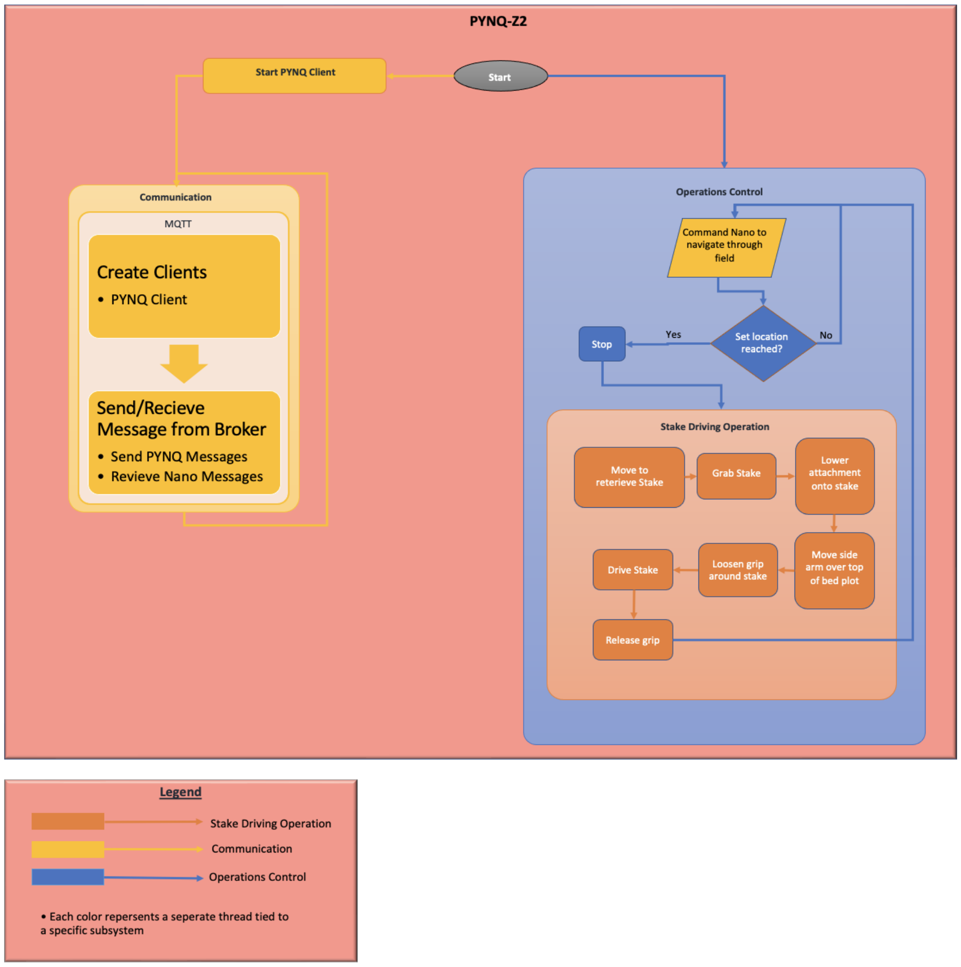


Figure 8.4: PYNQ-Z2 processing flow diagram

Depending on the parameters sent by the user, coming from the start job block in the app, the PYNQ-Z2 will begin to command the Jetson Nano to move the StakeBot to the next staking location, it will then command the StakeBot to stop if the StakeBot has reached the next staking location. The final thread is the stake driving operations which are responsible for controlling the subsystems that handle driving the stakes. This includes arm movements and gripper movements and, depending on the parameters set by the user, the StakeBot will position and drive the stake into the bed plot, then retract back for another stake. The PYNQ-Z2 board is the on-board AI and currently behaves as a low-level AI system capable of instructing the StakeBot to move and drive stakes and to communicate any issue that the StakeBot might experience to the user through the mobile application.

8.4 StakeBot's Future:

The StakeBot currently has the ability for a low-level AI that can control the navigation of the StakeBot through a row of bed plots. This is made possible with the vision system that is capable of monitoring obstacles in real time, preventing the StakeBot from collision while attempting to move forward through bed plots. The success of the StakeBot's operations can come from the current on-board systems but only in a very controlled case. The next level of progress will come from machine learning and advanced AI development where the StakeBot's on-board AI can make moving through the bed plots more possible as the more particle scenarios are taught. Currently the development of the StakeBot is shown below in Figure 8.5 where everything but the stake holster has been developed and implemented.

Shown in Figure 8.6 is the pseudo-bed plots that were created for testing the StakeBot's self-navigation capabilities. These bed plots were built in the lab that fit the size and dimensions of a typical bed plot. So far, the StakeBot has been able to self-navigate through the set of pseudo-bed plots only after being positioned directly in front of them.

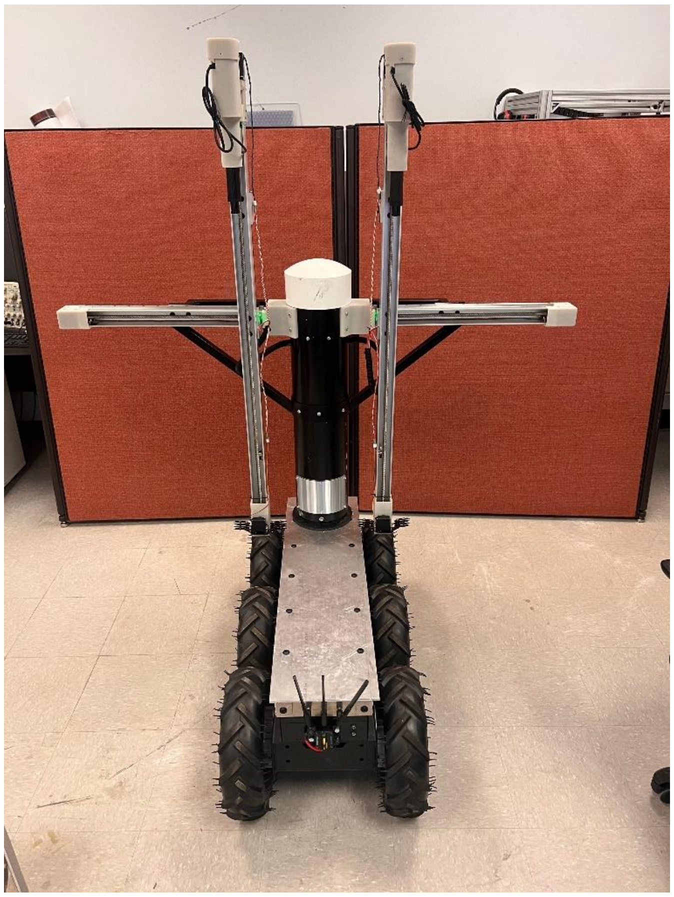


Figure 8.5: Current StakeBot development

Even though the StakeBot was able to self-navigate through this in housed mockup of a bed plot row, the next set of testing is to try and attempt a real scenario down a set of actual bed plots in an agricultural setting. However, before this can be attempted, advancements on the on-board AI must be developed. This can be made possible by using Nvidia Isaac Sim software to subject the virtual StakeBot down various simulated bed plots for the AI to learn in the virtual environment first. The success of the StakeBot will only come from advanced AI however, the current algorithms provide a decent starting point for further development. Many of the subsystems have been developed for control taken from an AI and much of the framework has been developed. Overall, the

StakeBot has shown to be capable of autonomous navigation but further development and machine learning implementation will improve the AI’s overall success rate and will improve the StakeBot’s capabilities.

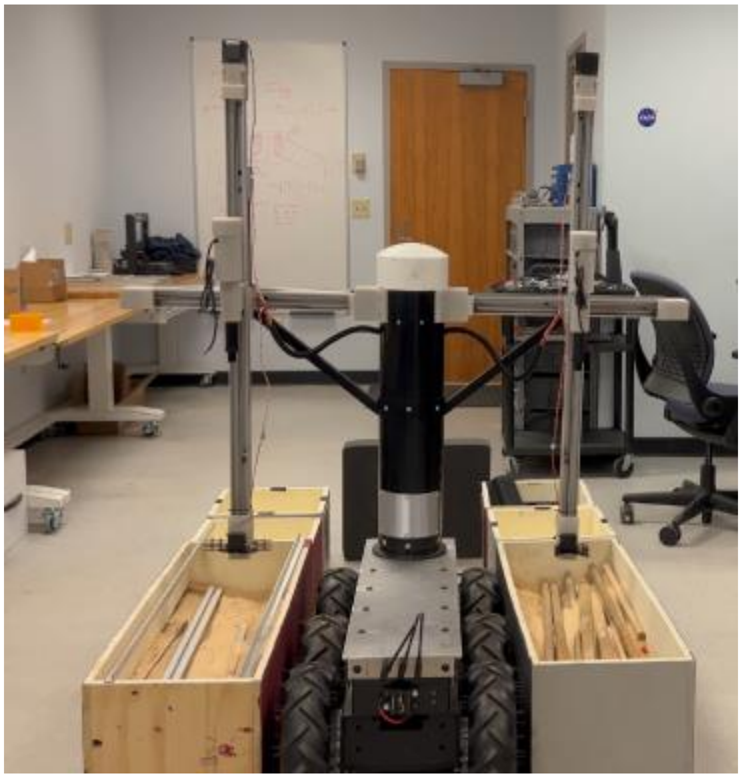


Figure 8.6 Current StakeBot navigating through pseudo-bed plots.

REFERENCES

A. Cherubini, F. Spindler and F. Chaumette, "Autonomous Visual Navigation and Laser-Based Moving Obstacle Avoidance," in IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 5, pp. 2101-2110, Oct. 2014, doi: 10.1109/TITS.2014.2308977.

Aghi, Diego, et al. ‘Local Motion Planner for Autonomous Navigation in Vineyards with a RGB-D Camera-Based Algorithm and Deep Learning Synergy’. Machines, vol. 8, no. 2, 2020, https://doi.org10.3390/machines8020027.

Básaca-Preciado, L. C., Sergiyenko, O. Y., Rodríguez-Quinonez, J. C., García, X., Tyrsa, V. V., Rivas-Lopez, M., … Starostenko, O. (2014). Optical 3D laser measurement system for navigation of autonomous mobile robot. Optics and Lasers in Engineering, 54, 159169. doi:10.1016/j.optlaseng.2013.08.005

Birrell, S, Hughes, J, Cai, JY, Iida, F. A field-tested robotic harvesting system for iceberg lettuce. J. Field Robotics. 2020; 37: 225– 245. https://doi.org/10.1002/rob.21888

“Depth Camera D435.” *Intel® RealSense™ Depth and Tracking Cameras*, 5 Dec. 2022, Accessed Jan. 2022, https://www.intelrealsense.com/depth-camera-d435/.

Derpanis, Konstantinos G. *Overview of the RANSAC Algorithm - Electrical Engineering and Computer* ... 13 May 2010, Accessed Aug. 2022, http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf.

“Harvest Croo Robotics.” *Harvest CROO Robotics*, Accessed Aug. 2022, https://www.harvestcroorobotics.com/.

“Isaac Sim.” *NVIDIA Developer*, 23 Feb. 2023, Accessed March 2023, https://developer.nvidia.com/isaac-sim.

J. Eguchi and K. Ozaki: Extraction method of travelable area by using 3D-laser scanner – Development of autonomous mobile robot for urban area--, Transactions of the Society of Instrumental and Control Engineer, Vol. 52, No. 3, pp. 152-159, 2016.

Li, Chuyi, et al. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. arXiv, 2022, https://doi.org10.48550/ARXIV.2209.02976.

Malavazi, Flavio B. P., et al. ‘LiDAR-Only Based Navigation Algorithm for an Autonomous Agricultural Robot’. Computers and Electronics in Agriculture, vol. 154, 2018, pp. 71–79, https://doi.org10.1016/j.compag.2018.08.034.

“Nexus Robotics at Expo Champs.” *Nexus Robotics*, 2020, Accessed Aug. 2022, https://nexusrobotics.ca/.

RealSense, Intel. “Beginner's Guide to Depth (Updated).” *Intel® RealSense™ Depth and Tracking Cameras*, 8 May 2020, Accessed Dec. 2022, https://www.intelrealsense.com/beginners-guide-to-depth/.

“Stake Driver System.” *ALMACO*, Accessed Aug. 2022, https://www.almaco.com/store/c152/optional-advantages/p1532/stake-driver-system/.

Steve, et al. “How Mqtt Works -Beginners Guide.” |, 12 Feb. 2021, Accessed July 2022, http://www.stevesinternet-guide.com/mqtt-works/.

Yamashita, Satoshi, et al. ‘Autonomous Traveling Control of Agricultural Mobile Robot Using Depth Camera in Greenhouse’. *Journal of Signal Processing*, vol. 23, no. 4, 2019, pp. 201–204, https://doi.org10.2299/jsp.23.201.

Yuhao Bai, et al. "Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review". Computers and Electronics in Agriculture 205. (2023): 107584.

APPENDIX A:

DESIGN REQUIREMENTS FURTHER EXPLAINED

Entry 2: This calculation was done by assuming 1200 (foot row length) / 8712 (foot per acers) * 2700 (stakes per acer) formulation was given by Jason on August 22, 2022

Entry 3: For target specification, the robot will be able to drive between and around the bed plots, provided there are no obstacles or uneven ground to traverse. For fallback specifications, the robot will be able to self-navigate only between the bed plots and would require operator assistance to turn the robot around to lineup with the next set of rows.