Autonomous Crop Spraying Robot with Precision RTK based Navigation

Group 19

Name	**Index number**
R.T. Hithanadura	170227R
M.N.C. Waas	170661P
T.D.Wanigathunga	170664D
W.M.T.D.Wijesundara	170714H

Supervisor
Prof. Rohan Munasinghe

External Collaborator
Nelna Agri Development (Pvt) Ltd.

Mid Review report submitted in partial fulfillment of the requirement for the Degree of B.Sc.(Hons) in Electronic and Telecommunication Engineering

February 5, 2023

Abstract

Autonomous Crop Spraying Robot with Precision RTK based Navigation

Group Members: R.T. Hithanadura, M.N.C. Waas, T.D. Wanigathunga, W.M.T.D. Wijesundara

Supervisor: Prof. Rohan Munasinghe

Keywords: Real Time Kinematic, ROS, Computer Vision, Robotics

The agriculture industry is one that is highly resource and labour-intensive. As such, farmers are increasingly turning to technology and automation to address this issue. However, agricultural robots are far too complicated, slow, and costly to be made publicly available. As a result, the agriculture sector still lags behind in integrating modern technologies. This project details the development of a low-cost agricultural robot for spraying fertilizers to mango trees by navigating through a precise user-defined path with RTK and Machine Vision technologies.

In this report, the design, implementations so far, and testing of an autonomous agricultural robot with RTK guidance is presented. This robot is also responsible for detecting trees and spraying fertilizers appropriately and not wasting. This rover is powered by 12 V car batteries and two electric motors. Deep learning algorithms such as object detectors has been used to detect trees found in a mango field. The robot control system consists of RTK guided control of propulsion system and steering actuators, an image processing and detection system, object avoidance using sensing the surroundings and a spray control system for fertilizer applications. Multiple embedded devices such as Jetson Nano, Arduino mega, as well as an on-board computer have used to provide all control functions in an integrated fashion. Open sources software such as Mission Planner and Reach-View have been used to provide autonomous guidance of the vehicle. Object detection model like SSD-Mobilenet has been used to detect trees. Robot Operating System has been used to simulate the robot and to implement the navigation algorithms integrating with the processing units.

The entire project is segmented into three sections as follows. Robot design and navigation system, deep learning approach for tree detection and RTK positioning method for accurate lattitude and longitude readings. We are designing the robot as a proof of concept via scale down implementation as the initial stage of the project adhering to industrial norms. The experimental results up-to now demonstrate that this technology is very promising and can be further developed to provide full functionality and greater degree of accuracy.

Contents

List of Figures

List of Tables

Acronyms

GND	Ground.
GPS	Global Positioning System.
GPU	Graphical Processing Unit.
IMU	Inertial Measurement Unit.
LCD	Liquid Crystal Display.
LED	Light-emitting Diode.
PCB	Printed Circuit Board.
PoC	Proof of Concept.
ROS	Robot Operating System.
RTK	Real Time Kinematic.
SNR	Signal-to-noise ratio.
SSD	Single-shot multibox Detection.
YOLO	You Only Look Once.

1 Introduction

Autonomous robots are using in different fields to ease work in those field. Using autonomous robots in agriculture industry is becoming popular recently for different tasks such as watering, fertilizing, seeding, harvesting etc. In our project, we develop a robot for fertilizing crops which navigates using Real Time Kinematic (RTK) technology. In this chapter we will give an overview about our problem statement, primary objective, scope of the project, uniqueness and benefits of the project, potential customers and beneficiaries and analysis of alternative methods.

1.1 Problem Statement

We try to address one of the important technological down step in Sri Lanka, which is applying state of the art technological developments to the field of agriculture. In collaboration with Nelna Agri development (Pvt.) Ltd, we are trying to develop a robotic solution to one of the most common needing issue in agricultural field, that is fertilizing the crops. Our solution will make the fertilizing crops much more efficient than the traditional systems currently used in Sri lanka. We identified some of the issues that the current system is facing. Some of them are,

- Fertilizing is not done methodically but it should be done to benefit the maximum harvest
- Labour utilization will lead to these
  - Some regions may be missed
  - Fertilizer overflow
  - Difficulties in reaching
  - Timing issues
  - Labour cost

Nelna Agri development (Pvt.) Ltd, has a human operated bowser truck currently working as their main fertilizing vehicle but, we have planned to replace the bowser truck with a robot vehicle which perform all the functionalities autonomously.

1.2 Primary Objective

Our primary objective is to build a robot that sprays fertilizers by navigating autonomously in a user-defined path with high precision, using Real Time Kinematic (RTK) positioning technology and machine vision. As it moves along the way, the robot should be able to detect the trees and take optimum decisions in order to spray the fertilizers. In addition to that, this will consist of many features like collision detection and avoidance.

1.3 Scope of the Project

We have divided the scope of the project into six categories and we are planning to proceed with each step.

- Find and mark accurate Global Positioning System(GPS) locations
- Implement RTK system in the rover to get accurate positioning data
- Build an algorithm to navigate through user given path
- Detect trees/plants and fertilize them
- Real time implementation on an embedded system
- Demonstrate it in a plant field

We were advised to build a robot as the Proof of Concept (PoC) for our final year project demonstration without scaling up to the bowser truck.

1.4 Uniqueness and Benefits of the Project

Regarding the uniqueness of the project, we could find a Australian based innovative agricultural company named XAG [1] as a competitor. XAG has a product called R-150. There are some other products as well but we have selected R-150 since it has almost same features as what we are trying to achieve in this project. However, there are some improvements in our system. In XAG, there is no intelligent system to fertilize the plants. It just rotate the nozzle and fertilize everywhere. That cannot be used for a farm where there is a significant gap between trees because then there may be a huge fertilizer wastage. In our approach, we are fertilizing only if there is a tree. Then fertilizer loss is very low. And in their system, there is no method to avoid collisions with humans and objects. Since we are using a machine vision approach, our rover is capable of above issue as well. And cost-wise, it is around 60 lakhs rupees. Therefore, through the unique features that our system has, there is a possibility to market our system over XAG.

1.5 Potential Customers and Beneficiaries

This project is proposed to the Nelna agri Development (Pvt.) Ltd. Therefore, the direct beneficiary of this project is Nelna. However, since we are solving a problem related to agriculture, this project will be useful for the agriculture sector. Since we are using RTK with autonomous driving, this project can be further improved to use for surveying and road mapping too.

After completing, this project is capable of providing valuable services for the users. Some of them are;

- It can reduce the labor cost that requires for fertilizing; only one person will be required to check the autonomous driving.
- Sometimes laborers will miss certain trees or certain regions, but we can reduce those types of errors by using our robot.
- Since we are using machine vision to identify the trees, we can increase the efficiency of fertilizing by reducing fertilizer wastage.
- We can use our robot throughout the daytime. Therefore we can achieve a better working efficiency.

1.6 Analysis of Alternatives

This section contains the comparison between the methods that we are using for our project and available alternatives.

1.6.1 Positioning

Table 1.1: Comparison of alternative positioning methods

Method	**Pros**	**Cons**
Global Positioning System (GPS)	· Low cost · Fast	· Low accuracy
Real Time Kinematic (RTK)	· High Accuracy · Cellular coverage is not required	· High cost · Accuracy decreases when distance between base and rover is increased

RTK [2] is used for positioning because it gives more accurate position than GPS [3].

1.6.2 Navigation

Table 1.2: Comparison of alternative navigation methods

Method	**Pros**	**Cons**
Vision Based	· Can be used without GPS signals	· Hard to implement at night. · High impact by weather conditions. · Vehicle localization is not possible with the absence of road line.
RTK Based	· High reliable · Fast feedback	· Need GPS signals

RTK based navigation is used because vision based navigation is hard to implement compared to RTK based navigation.

1.6.3 Embedded System

Table 1.3: Comparison of alternative embedded systems

Method	**Pros**	**Cons**
Raspberry Pi	· Low power consumption · Low cost · Already available	· Relatively low performance
NVIDIA Jetson Nano	· Relatively high performance	· High cost · Power consumption

NVIDIA Jetson Nano is used for the embedded system because it has higher performance than Raspberry Pi.

In this chapter, we discussed about objective of our project and the problem that we are going to address. In addition, we explained scope, benefits, uniqueness, potential customers of our project and analysis of alternative methods.

2 Methodology

This chapter looks at the proposed methodology for building our robot and the systems that we are going to use together in order to navigate in the user defined path.

2.1 Overview

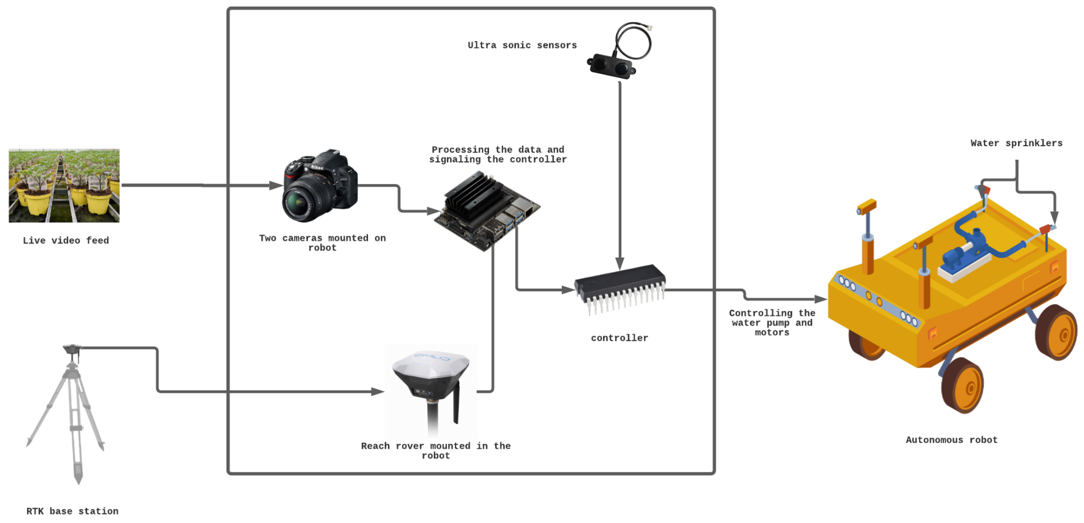

Ultra sonic sensors
Processing the data and signaling the controller
Water sprinklers
Live video feed
Two cameras mounted on robot
controller
Controlling the water pump and motors
Reach rover mounted in the robot
Autonomous robot
RTK base station

Figure 2.1: Architecture diagram

This is a high level project illustration about our project architecture. The main part of our robot is the RTK positioning system which we use to navigate the robot in centimeter level accuracy along the path given by us. We have mounted two cameras in the robot and they will feed live video into a Jetson nano and the machine vision system will run and detect the trees that we want to spray when necessary. Further explanation of each part will be presented on each section separately.

2.2 Positioning Method

In this section, the method that we are going to use for the positioning of the robot will be explained.

2.2.1 Introduction

For the navigation, positioning the robot accurately in the path is very important. Therefore, we use RTK technology for more accurate positioning in autonomous navigation.

RTK is a technique that is used to improve the accuracy of traditional GPS receiver. We use two RTK receivers as the base station and the rover. We use two emlid reach M+ modules [4] for our project. Base station is stationery and placed in a known location. Rover moves freely and it is mounted on our robot. Since base station is placed in a known location, it knows its exact position. Then, base station can measure errors of GPS signals that receives from different satellites. After calculating the errors of each satellites, base station transmits error values to the rover. Using those values, rover corrects GPS locations that receives from different satellites and correct its position in centimeter level accuracy. We use 915MHz telemetry modules to communicate between the base station and the rover. After correcting its position, rover inputs its current location to the controller module. In addition, for the navigation we define the path that robot should move using mission planner software [5]. Then we can input this path to the controller. We create an algorithm such that calculate the error of position by comparing given path and given location by the RTK receiver and correct it accordingly. Then navigation will happen according to above algorithm.

2.2.2 Base Station Configuration

For RTK configuration, first we placed the base station at a known location in the university. These points were established by survey department. There are several points around the university. Figure 2.2 shows the placement of the base [6]. The base station should be placed at a place where there is clear sky view to get more accurate results. We placed a tripod on the known point accurately and then mounted the M+ module on it. This placement should be accurate. Otherwise, that error is added when calculating the error of GPS coordinates and because of that we cannot calculate the position of the rover accurately. After placing the base station at one such location, we configured settings of that M+ module [7]. This can be done by connecting to the local Wi-Fi of the device and login to the web page of the device using its IP address. It connects to the Reach Panel web app [8]. Configuration of the base is shown in Figure 2.3. First, we set the positioning mode as 'Static' to configure the M+ module as base station. After that, we input coordinates of the location of the base station and the height of the antenna manually. Correction output is sent via serial port and 57600 is selected as baud rate.

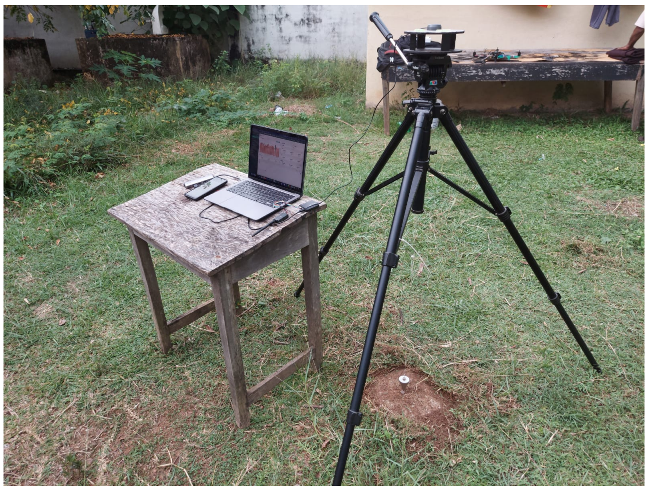
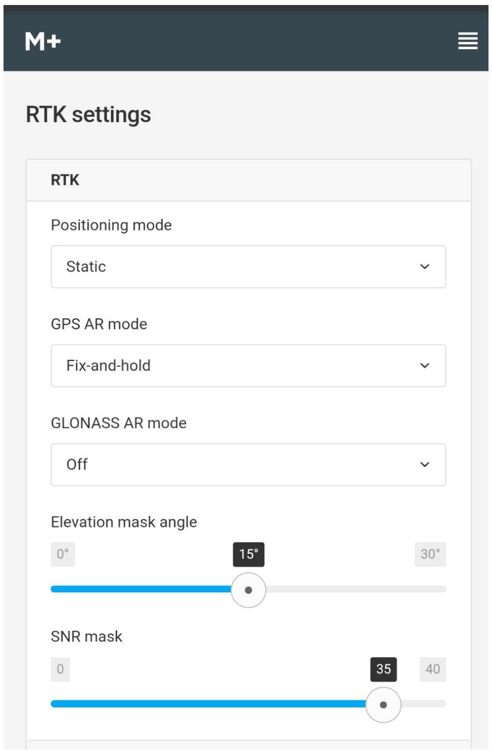


Figure 2.2: Placement of the base station


M+

RTK settings

RTK

Positioning mode: Static

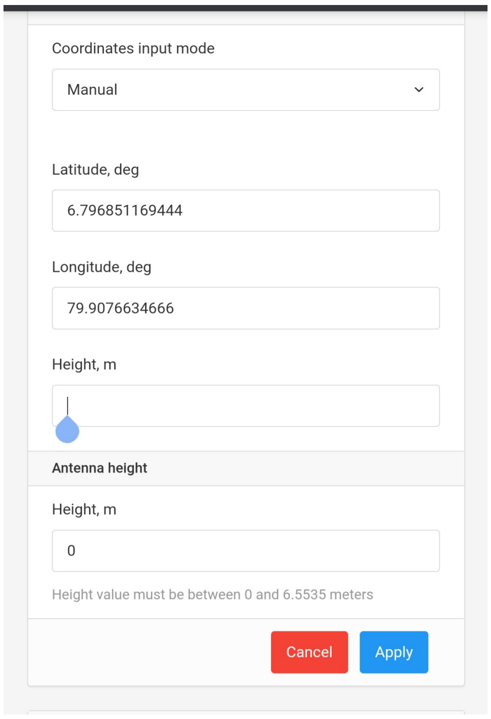
GPS AR mode: Fix-and-hold

GLONASS AR mode: Off

Elevation mask angle: 0° 15° 30°

SNR mask: 0 35 40

(a) RTK settings for base


Coordinates input mode: Manual

Latitude, deg: 6.796851169444

Longitude, deg: 79.9076634666

Height, m:

Antenna height

Height, m: 0

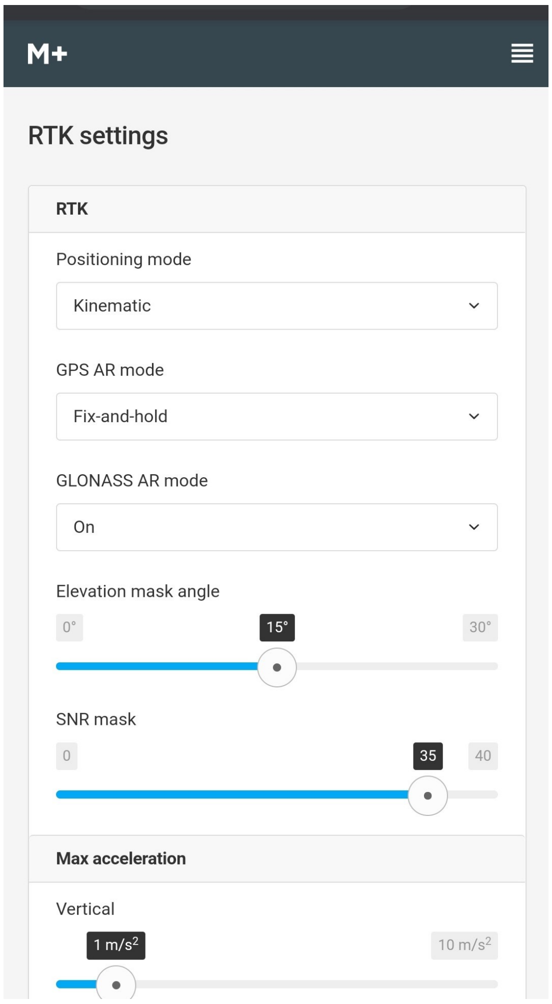
Height value must be between 0 and 6.5535 meters

(b) Inserting coordinates of the base

Figure 2.3: Configurations of base station

2.2.3 Rover Module Configuration

After that, we configured other M+ module as rover [7]. This was done by connecting to the local Wi-Fi of the device and login to the Reach Panel web app as we did when configuring the base station. Figure 2.4 shows the configurations of the rover. In this time, we set the positioning mode as 'Kinematic' to configure the M+ module as rover. Correction input is received via serial port and 57600 is selected as baud rate.


M+

RTK settings

RTK

Positioning mode: Kinematic

GPS AR mode: Fix-and-hold

GLONASS AR mode: On

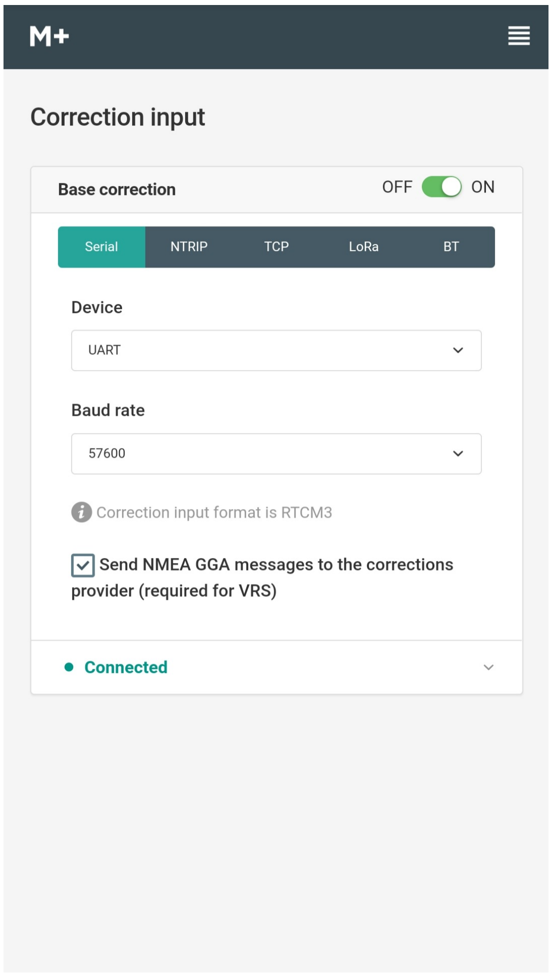
Elevation mask angle: 0° — 15° — 30°

SNR mask: 0 — 35 — 40

Max acceleration

Vertical: 1 m/s^2 — 10 m/s^2

(a) RTK settings for rover


M+

Correction input

Base correction OFF ON

Device: UART

Baud rate: 57600

Correction input format is RTCM3

[✓] Send NMEA GGA messages to the corrections provider (required for VRS)

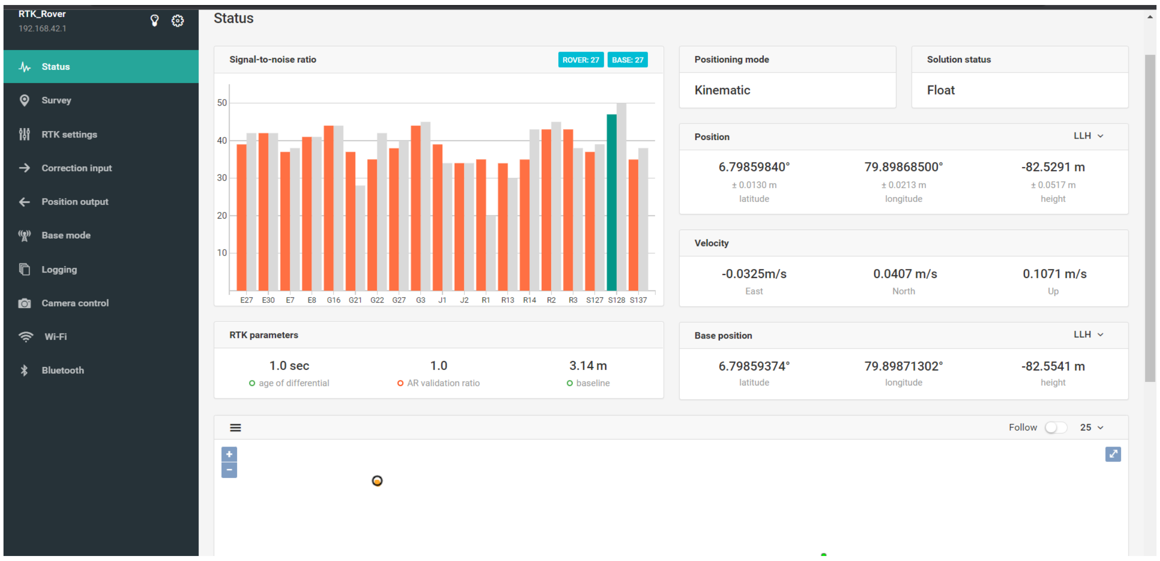
● Connected

(b) Correction input settings

Figure 2.4: Configurations of rover

2.2.4 Establishing Connection between Base Station and Rover

We kept the rover module at some distance from the base and checked the connectivity between the base and the rover. In addition, we checked whether base station transmits corrections to the rover. Figure 2.5 shows the status tab [9] of Reach Panel web app of the rover module. The orange or green bars in the graph shows the SNR of connected satellites of rover and gray bars shows the SNR of connected satellites of base. Since SNR of both base station and rover is shown in the status tab of rover module, we can decide that connection between base and rover is established.



Status

Positioning mode: Kinematic

Solution status: Float

Position (LLH)

6.79859840° ± 0.0130 m latitude | 79.89868500° ± 0.0213 m longitude | -82.5291 m ± 0.0517 m height

Velocity

-0.0325m/s East | 0.0407 m/s North | 0.1071 m/s Up

RTK parameters

1.0 sec age of differential | 1.0 AR validation ratio | 3.14 m baseline

Base position (LLH)

6.79859374° latitude | 79.89871302° longitude | -82.5541 m height

Figure 2.5: Status tab of rover

2.3 Tree Detection

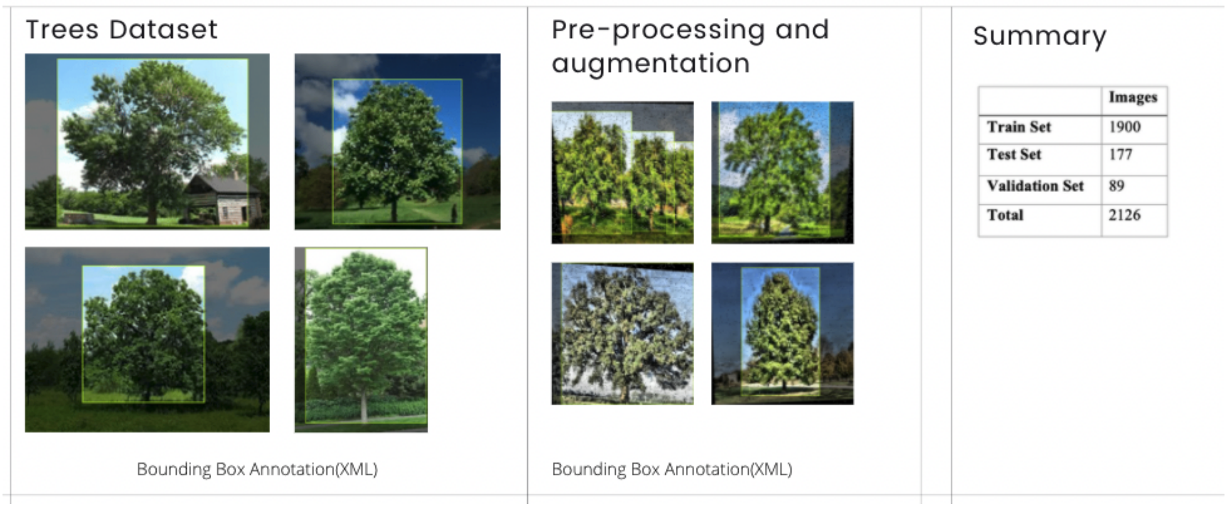
2.3.1 Introduction

Tree detection is one of the main systems in our robot that will be used to optimally take decisions on fertilizing procedure. Fertilizing can be done easily by continuously pumping the fertilizer through the nozzles. However, we can use machine vision to identify the trees. Then the robot can reduce the fertilizer wastage by fertilizing only when a plant is detected.

2.3.2 Dataset Creation

This section brings forth the process of dataset creation including the data collection and data annotation tasks along with a summary of the dataset.

There are existing datasets for training the machine vision algorithm. However, the problem with existing datasets are, they are not specifically customized to mango trees or any other similar looking trees. Therefore, we created our own dataset by scraping the internet and this dataset contains around 1200 images of trees. We applied many transformation techniques to extract features out of the images and also made the dataset much bigger having more augmented data. The extracted images are used to create train, validation and test sets. The trees found in the extracted images are annotated manually as polygons using the roboflow annotation tool. The bounding box annotations are saved as in XML format and yolo format as to manually train both Tiny-YOLO [10] and SSD-mobilenet [11] networks and evaluate the rsults in order to find the best model of our need. This variety of annotation formats will facilitate a wide range of deep learning based research in future. The summary of the dataset and some grasp of the images are shown in the Figure 2.6.


Trees Dataset
Pre-processing and augmentation
Summary
Bounding Box Annotation(XML)
Bounding Box Annotation(XML)

	Images
Train Set	1900
Test Set	177
Validation Set	89
Total	2126

Figure 2.6: Dataset statistics

Final dataset after pre-processing contains total of 2126 images having a resolution of 1080x720. The dataset is divided into the train set and the test set, which comprises 1900 images and 177 images, respectively. In this application, the rover is continuously moving. Therefore, we have to make sure that our robot can identify the trees while moving. Therefore, a clean dataset with strong results in identifying the trees is necessary in order to achieve optimum results in spraying fertilizers. Also, the robot should avoid collisions to increase the security of the workers and the robot. For that we are using the ultra sonic sensors mounted in front of the robot.

2.3.3 Tree detection

This section presents the proposed model architecture and methodology employed for the tree detection task.

Model Architecture

Figure 2.7 illustrates our tree detection and classification architecture. A state-of-the-art object detector model is used to first detect and predict the bounding box of the trees present in an image.

Single-Shot multibox Detection(SSD) [11] is a popular network architecture for realtime object detection on mobile and embedded devices that combines the SSD-300 Single-Shot MultiBox Detector with a Mobilenet backbone. The selected base model was already pre-trained on a different dataset (PASCAL VOC). Therefore, we don't need to train SSD-Mobilenet [11] from scratch, which would take much longer. Instead we'll use transfer learning to fine-tune it to detect trees with our datasets.

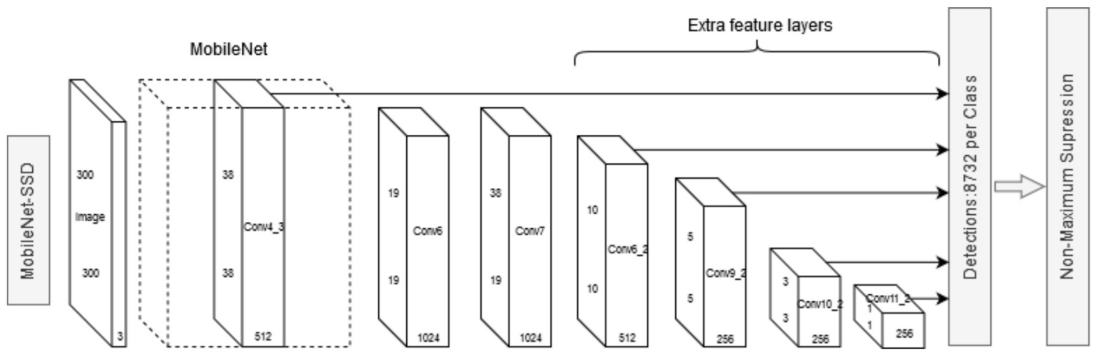


Figure 2.7: Model architecture

Detection Stage

We evaluate the performance of two state-of-the-art object detectors for object detection task. We researched and selected SSD-Mobilenet [11] and tiny-YOLO [10] to train and get a good FPS rate and accuracy with jetson nano. Darknet is used as the backbone of the tiny-YOLO [10] model while MoblieNet-v2 is used as the backbone of the SSD model. Input images are resized from 1080x720 to 300x300 and 416x416 to make compatible with the two models and to increase the inference speed. Non-maximum suppression is carried out in SSD-Mobilenet [11] model to reduce the overlapping detections to a single bounding box with an IoU (Intersection over Union) threshold of 0.45.

Data Augmentation

We used data augmentation techniques such as Flip: Horizontal, Shear: ±15° Horizontal, ±15° Vertical, Noise: Up to 1% of pixels to reduce the effect of class imbalance when training models. This stage is to be optimized in the future as well.

2.3.4 Implementation Details

For the training of our detection algorithms, we use a computational platform comprising Apple Silicon M1 Chip and Apple M1 graphical processing unit(GPU) integrated graphics card offering 8 cores designed by Apple and integrated in the Apple M1 SoC. According to Apple, it is faster and more energy efficient as competing products (like the Tiger Lake Xe GPU). We use pytorch to train the object detectors.

For training the SSD-MobileNet [11] object detector model, initial learning rate of 0.01 and a momentum of 0.9, and the batch size is set to 4. Optimizations and improvements are yet to be implemented.

2.4 Navigation System

2.4.1 Introduction

Robot navigation is one of the main objectives of this project. The robot should move in a farming environment like shown in Figure 2.10. Here, we are mainly focusing on the Nelna Agri farm. Mission planner is a planning software where we use it to find the GPS locations between two specific points. First, we need to mark starting and ending points. Then, we can sample the straight line between starting and ending points as shown in the Figure 2.9 to get more points on the line. Then, we can download the data as a .txt file to be used as an input to the navigation system. The navigation system should have the capability of controlling the robot in such a way that to navigate through all the given positional data points by maintaining centimeter-level accuracy. For that to happen, the robot uses the above-mentioned RTK technology.

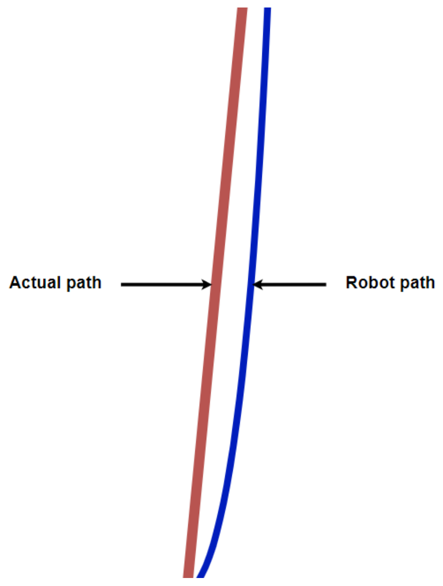
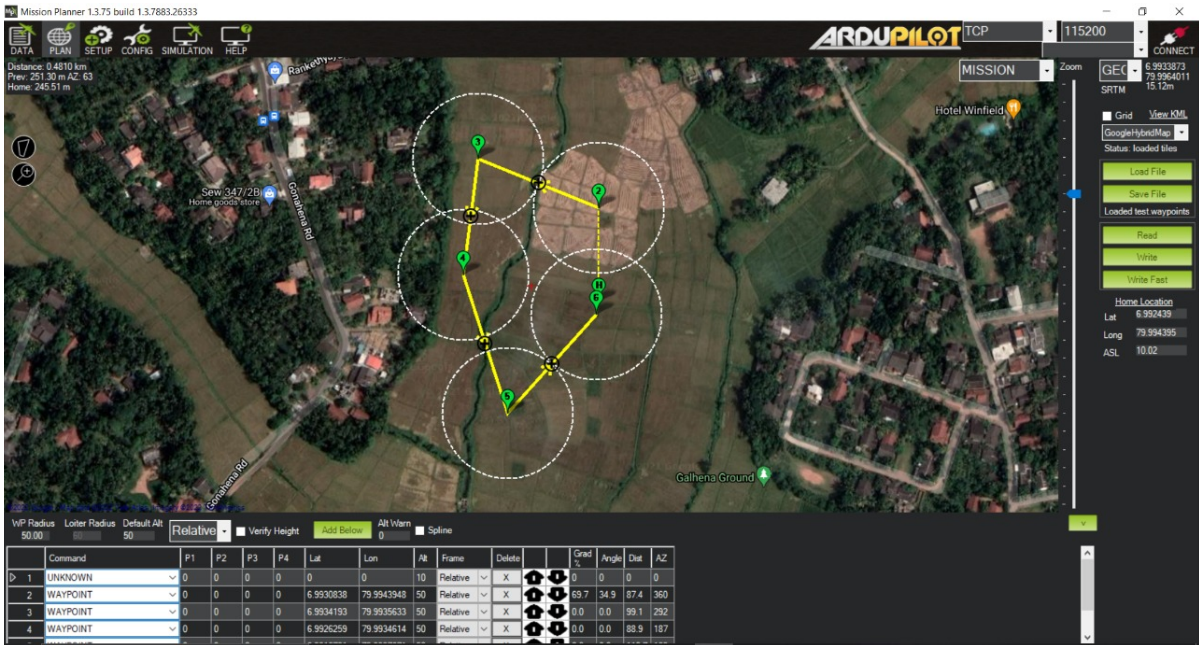


Figure 2.8: CM level accuracy



Distance: 0.4810 km
Prev: 251.30 m AZ: 63
Home: 245.51 m

Home Location
Lat 6.992439
Long 79.994395
ASL 10.02

WP Radius 50.00 | Loiter Radius 60 | Default Alt 50 | Relative | Verify Height | Add Below | Alt Warn 0 | Spline

	Command	P1	P2	P3	P4	Lat	Lon	Alt	Frame	Delete			Grad %	Angle	Dist	AZ
1	UNKNOWN	0	0	0	0	0	0	10	Relative	X			0	0	0	0
2	WAYPOINT	0	0	0	0	6.9930838	79.9943948	50	Relative	X			69.7	34.9	87.4	360
3	WAYPOINT	0	0	0	0	6.9934193	79.9935633	50	Relative	X			0.0	0.0	99.1	292
4	WAYPOINT	0	0	0	0	6.9926259	79.9934614	50	Relative	X			0.0	0.0	88.9	187

Figure 2.9: Path planning with mission planner

Figure 2.10: Farming Environment

2.4.2 Guidance logic

There are two methods that we can use to keep the robot on track while moving. First, use a reference point and keep it static until the robot reaches that point as depicted in Figure 2.11. Second, use a dynamic reference point that moves along the line while keeping the distance to the robot constant as depicted in Figure 2.12.

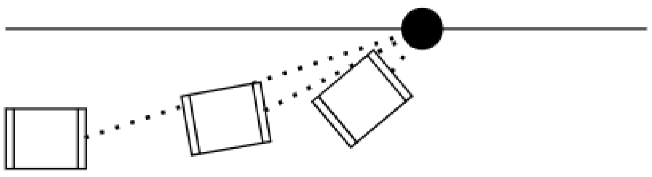


Figure 2.11: Static reference point

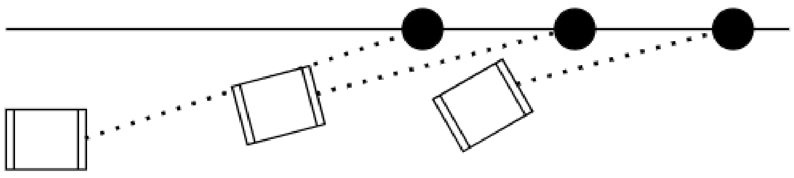


Figure 2.12: Dynamic reference point

Here, in our robot, we use the second method because according to Sanghyuk *et al.* [12]. It can take the robot onto the line and keep it on the line with less deviation. This is how it works. First, we need to find a proper distance (L1) to select a reference point away from the robot. The L1 distance is the same while the robot moves as shown in the Figure 2.13. Then we need to find the respective angles as shown in the Figure and rotate the wheels accordingly. Ultimately the robot will start to follow the correct path and the Figure 2.14 shows how the robot approaches the correct path.

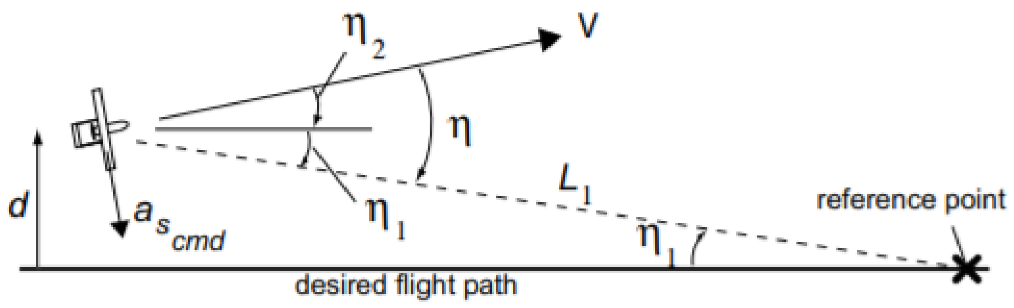


Figure 2.13: Follow straight line

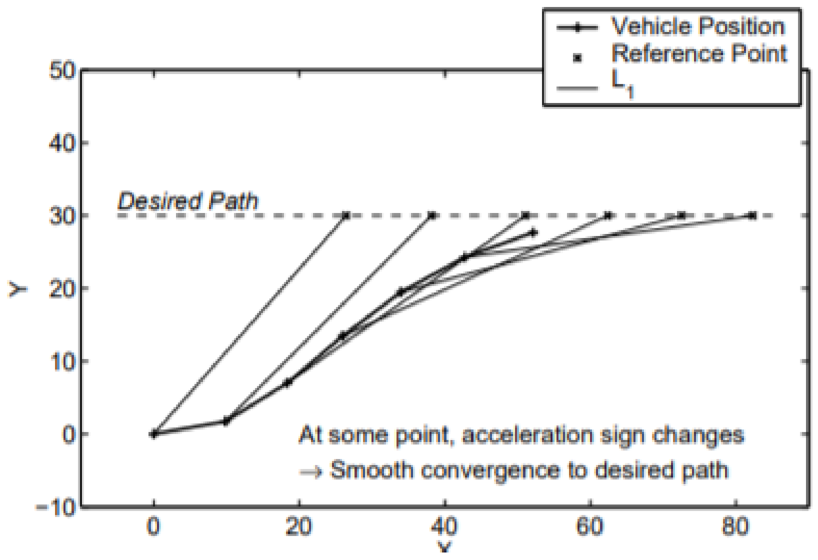


Figure 2.14: Step by step approach

2.5 Robot Controlling

2.5.1 Introduction

Robot controlling is the core functionality of the robot. Here in our robot, there are several subsystems to be controlled. Each and every sub-system is taking care of achieving an accurate outcome, but the overall accuracy depends on the intermediate system which has control of all the subsystems. Therefore, more attention should be given to the controlling system. Figure 2.15 shows the connectivity between each subsystem with central unit.

As described above, the subsystems that should be controlled are,

- Tree detection
- Fertilizing
- Navigating
- Obstacles avoidance
- RTK signalling

The controlling system should have the capability of triggering the above systems at the correct time without a due. That is where parallel processing comes into the picture. Here, as the central processing system, we use Robot Operating System (ROS). We have selected ROS because it has the potential of producing the control of the above functions parallel.

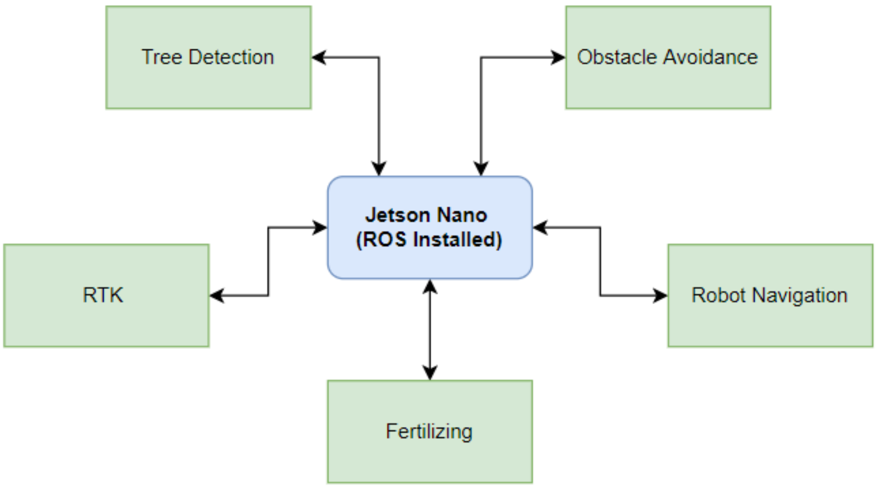


Figure 2.15: System connectivity

In our robot, there are different types of devices to be controlled. We can divide them into low-level and high-level components. Therefore, to control them separately and to make a robust controlling system we use two controllers as low-level and high-level controllers. The reason for depending on two different levels of controllers is to distribute the control of low-level and high-level components and systems. As the high-level controller, we use ROS installed Jetson nano and as the low-level controller, an Arduino board is used. As low-level components, in our robot, there are two high-power motors to drive the main wheels and four servo motors to control the fertilizing mechanism, three ultrasonic sensors to detect the surrounding, and a display. Each of them is connected with the Arduino board. There is an Inertial Measurement Unit(IMU) sensor directly connected with Jetson nano. The controllers communicate with each other as shown in Figure 2.16 serially to make the system distributed.

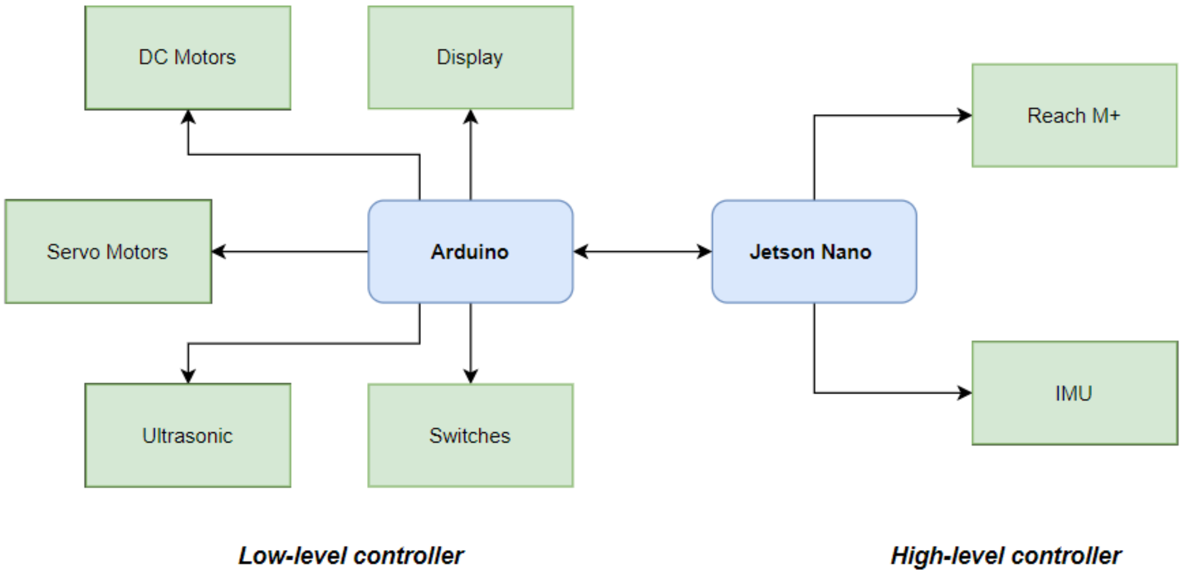


Figure 2.16: Control system

2.5.2 High-level Controller

In the robot, there are some systems that need high computing power. The tree detection system and the controlling system can be taken as two subsystems that need high computing power. The tree detection system uses a deep neural network as the backbone and it processes continuous image frames from two cameras to detect the surroundings. And, controlling system, here we use ROS, needs high computing power because it handles several nodes at a time using threads. Therefore, it is obvious that there should be such an embedded system that can handle all the data without overloading. Since Jetson nano can handle such a large amount of data, we use the Jetson Nano development kit as our high-level controller. We installed Ubuntu operating system on Jetson and ROS melodic version is installed on top of it.

2.5.3 Low-level Controller

There are different sensors that we use to capture the environment and control the robot. DC motors, servo motors, encoders, ultrasonic sensors, displays, and switches are the components that we connect directly with the low-level controller. Here, the low-level controller communicates with the high-level controller serially and exchanges the data between each other. Since we need a device that has the capability of transmitting and receiving other than component controlling, we use an Arduino mega as the low-level controller.

2.5.4 Method

As described above, we get the required path data from the .txt file. Once we set the path on the mission planner, it produces the positional data like GPS coordinates. Then, we need to convert it to Cartesian coordinates by taking the odometry frame into account. Here, the odometry frame is the frame where the origin is considered as the starting position of the robot.

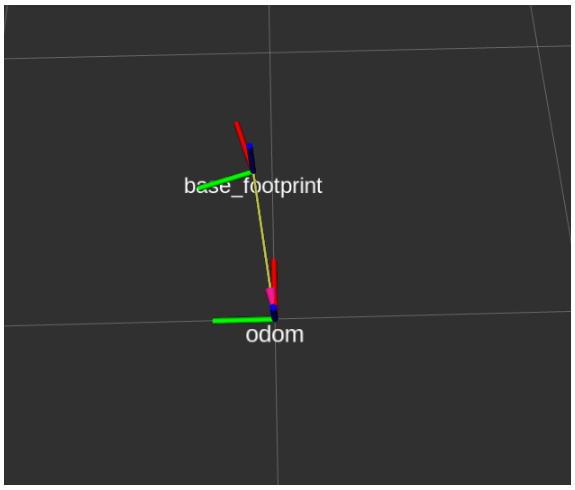


Figure 2.17: Robot frame with respect to the odometry frame

We need to have an accurate method to find the odometry (the distance traveled from the starting position) data of the robot. In our robot, we use RTK positioning data as one data source since it is accurate into centimeter level. However, there may be some situations where the RTK signals drop. Therefore we need to address those issues as well. Encoders are used as an alternative method for odometry. Here we use the sensor fusion technique using ROS. First, we need to transform RTK data to Cartesian coordinates using the navsat transform node [2.18]. Navsat transform is a ROS package that we can use as a separate node. Then, we use sensor fusion with encoder odometry, IMU, and transformed RTK data. Now, we have accurate positional data with respect to the origin of the odometry frame. Here, encoder odometry is calculated as follows. Figure 2.19 shows the robot movement from one point to another point.

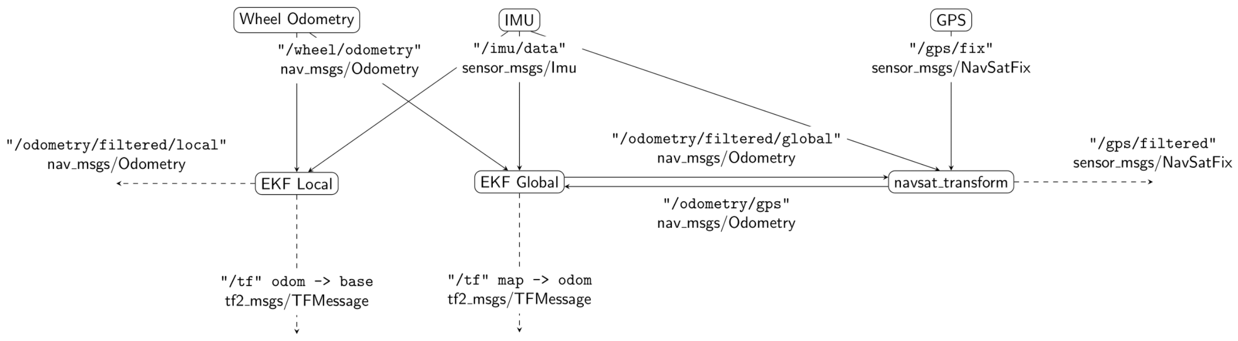

Wheel Odometry
IMU
GPS
"/wheel/odometry" nav_msgs/Odometry
"/imu/data" sensor_msgs/Imu
"/gps/fix" sensor_msgs/NavSatFix
"/odometry/filtered/local" nav_msgs/Odometry
"/odometry/filtered/global" nav_msgs/Odometry
"/gps/filtered" sensor_msgs/NavSatFix
EKF Local
EKF Global
navsat_transform
"/odometry/gps" nav_msgs/Odometry
"/tf" odom -> base tf2_msgs/TFMessage
"/tf" map -> odom tf2_msgs/TFMessage

Figure 2.18: Navsat transform node

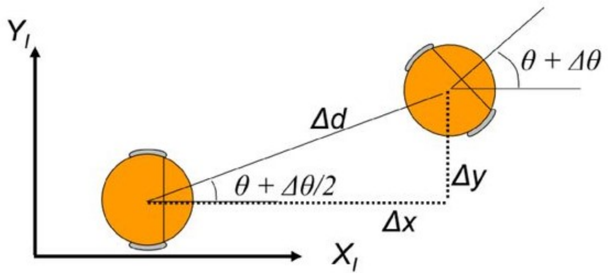


Figure 2.19: Positional difference between two time stamps

Then we need to calculate how much rotation has been occurred in two wheels when robot moves from one position to another. Then we can find the curvature of the movement using the formula, $\Delta S = R\Delta\theta$. After finding the $\Delta\theta$ we need to follow the following steps to get the Δx and Δy.

$$Average\ orientation\ is\ \theta + \frac{\Delta\theta}{2} \tag{2.1}$$

$$\Delta x = \Delta d \cos(\theta + \frac{\Delta\theta}{2}) \tag{2.2}$$

$$\Delta y = \Delta d \sin(\theta + \frac{\Delta\theta}{2}) \tag{2.3}$$

$$\Delta d \approx \Delta s \tag{2.4}$$

$$\Delta x = \Delta s \cos(\theta + \frac{\Delta\theta}{2}) \tag{2.5}$$

$$\Delta y = \Delta s \sin(\theta + \frac{\Delta\theta}{2}) \tag{2.6}$$

Then, we need to accumulate the Δx and Δy from the beginning to get the x and y from the origin.

We discussed about the proposed methodology for our project in this chapter. We have mentioned about the methods that we use for positioning of the robot, navigation, robot controlling and tree detection.

3 Design

This chapter describes the designing part of our project. It is divided into two sections. Those are robot design and Printed Circuit Board(PCB) design. Process that we followed in the designing of the robot and the PCB will be discussed here.

3.1 Introduction

We are using an existing wheelchair platform to build our robot. The existing platform is 78 cm long and has a width of 60 cm and a height of 50 cm. The platform itself weighs 28 kg. Since our robot contains lots of components, we have to do some modifications to the existing platform. The solid-works design of the existing platform is shown in Figure 3.1.

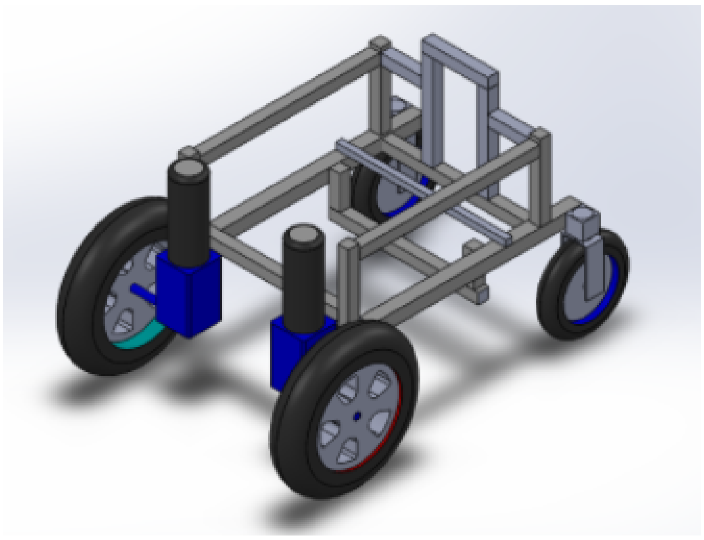


Figure 3.1: Existing wheelchair platform

3.2 Robot design

We are building the robot on an existing platform. That is the backbone or the structure of the robot. We have three main layers on this platform. The first layer is located at the bottom of the robot and it contains the batteries and fertilizer tank.

We are using two 12 V batteries which have a total of 30kg to power the system. Two batteries are located at the bottom of the robot to lower the center of gravity for greater stabilization. Two batteries are isolated from other components by considering the safety. They are connected in series to supply a 24 V input power.

After placing the batteries, we have 29cm x 14cm x 24cm space for the fertilizer tank. We have planned to locate a tank that can store 9.7 Liters of fertilizer that weighs 10 kg.

Therefore we can fertilize 45 trees by using 200 ml per tree from a single tank. This tank and the motors that use for pumping are located at the bottom back of the robot. That way we can isolate the fertilizer from electronic components.

Therefore we have divided this bottom layer into two chambers. One chamber contains two 12V batteries and the other one contains the tank that is required to store the fertilizer. These two chambers are isolated by a wall to protect batteries from the fertilizer if there was any leakage. Filling the fertilizer tank, we have designed a suitable hole in the robot body. Two batteries can be charged separately by using the ports located at the robot body.

The next layer of the robot body contains the brain or the Central PCB. This PCB has a high-level robot controller, which is a Jetson nano, a low-level robot controlling microcontroller, which is Arduino mega, all the motor controllers, such as water motor controller and two main motor controllers to control the motors. The 24 V input power is supplied from the layer below the PCB layer. PCBs are located above the battery chamber. Since the PCB is located at a top level, we can easily access the PCB and other components connected to the PCB. This layer is also isolated from all the layers to protect the electronics.

The third layer is the layer that contains the components that are interacting with the environment and the user. It contains two cameras on camera holders, two nozzles in two-link robot arms, a reach M+ module and its antenna, Liquid Crystal Display(LCD), Light Emitting Diode(LED) indicators, buzzer, and all the switches to interact with the user. The two nozzles are located at the further back end of the robot in order to protect other components from fertilizer. Further, some components have provided an extra shield to protect them, such as two cameras and a reach M+ module. Figures 3.2 and 3.3 show the modified robot platform.

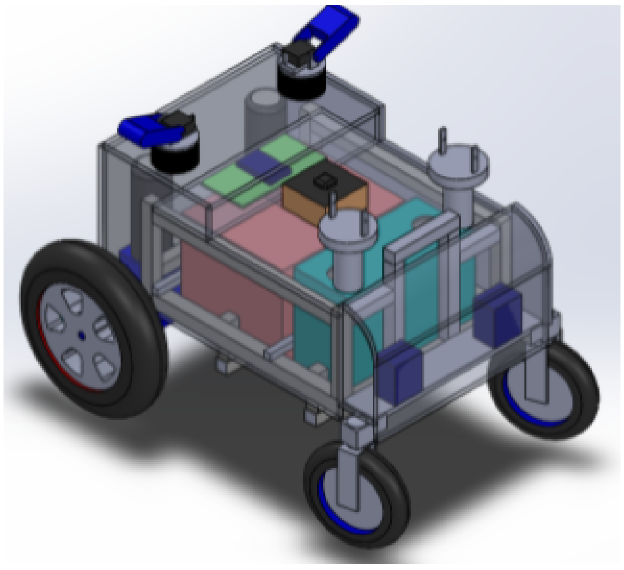


Figure 3.2: Modified platform

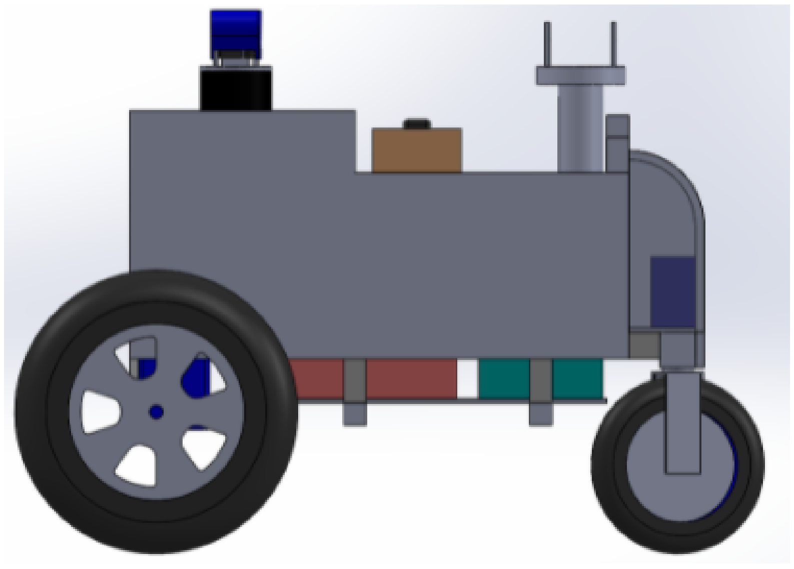


Figure 3.3: Side view

3.2.1 Mounting cameras

Two cameras are located at the front of the robot to capture the trees. They are placed at a fixed angle to capture the important areas in the surrounding. Two cameras are on the camera holding platform to avoid capturing robot parts. We are covering two cameras to provide protection from fertilizer. Two fertilizer nozzles are located at the furthest possible location to the two cameras. Figure 3.4 shows the camera setup.

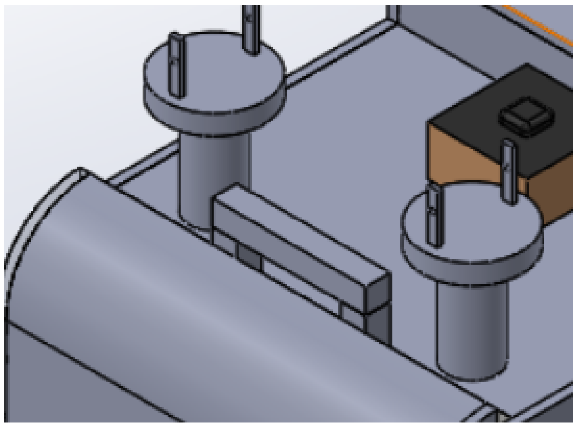


Figure 3.4: Camera mounting setup

3.2.2 Placing components of navigation system

We are using Reach M+ to correctly identify the location of the robot. We have placed the Reach M+ module, GPS antenna, and telemetry module at the center of the robot on the top floor. That way we are able to find the correct position of the center of the rover. It also helps to maximize the Signal to Noise Ratio(SNR) too. We are providing a shield to protect those modules from fertilizer as well. Three ultrasonic sensors are also located at the front and two sides to avoid a collision. The gyroscope is located at the center of the robot as well. The component placement of the navigation system is shown in the Figure 3.5.

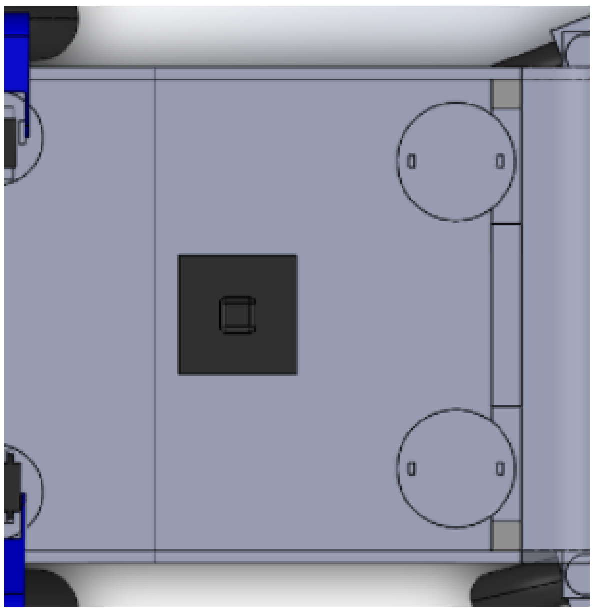


Figure 3.5: Components of navigation system

3.2.3 Fertilizing mechanism

Two water motors are connected with two nozzles which are located at each side of the robot to fertilize the identified trees. Two 2-link robot arms are used to control the direction of fertilizer by controlling the base of the robot arm. We are using two revolute joints to rotate the nozzles in two directions. Two 6V servo motors are located at each joint of the 2-link robot arm for that. Those robot arms help to fertilize the trees efficiently while reducing fertilizer wastage. Figures 3.6 and 3.7 shows the Fertilizing mechanism.

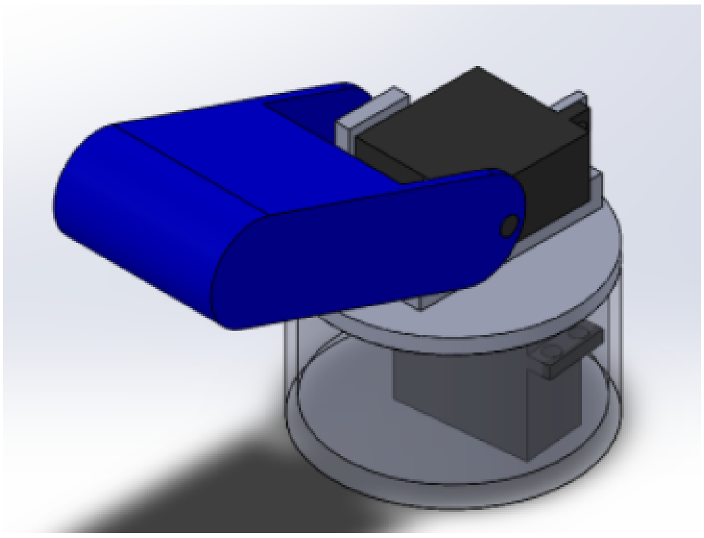


Figure 3.6: Spraying mechanism

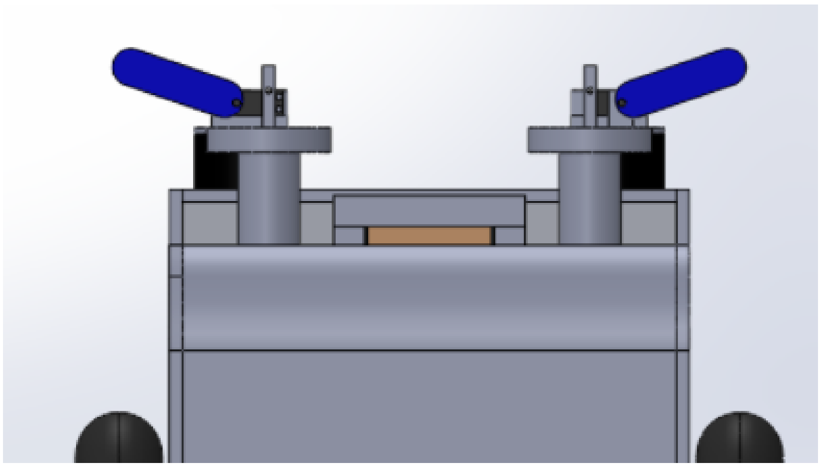


Figure 3.7: Front view of spraying mechanism

3.3 PCB design

We use Altium designer to design the Schematic and the PCB layout. The PCB contains Jetson nano, Arduino mega, and three motor controllers as main parts. The 24 V power supply is connected to the five parallel voltage regulators. Those regulators are capable of delivering 24V, 12V, 6V, and 5V regulated outputs for the use of various components in the robot. All the components are powered by this PCB.

Arduino mega is communicating with the Jetson nano. Arduino mega is connected to motor controllers, servo motors, LED indicators, buzzer, and LCD display. We have included 6V, 5V, and Ground(GND) power ports for further use. The gyroscope is also connected to the PCB.

The low-level controller is interacting with the user and taking inputs and giving indications by communicating with the high-level controller. Figure 3.8 and Figure 3.9 shows the schematic design and the PCB respectively.

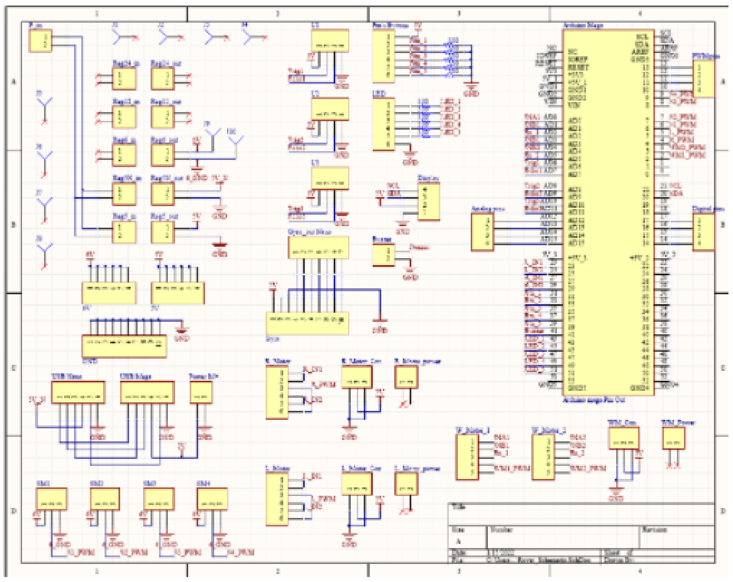


Figure 3.8: Schematic of the PCB

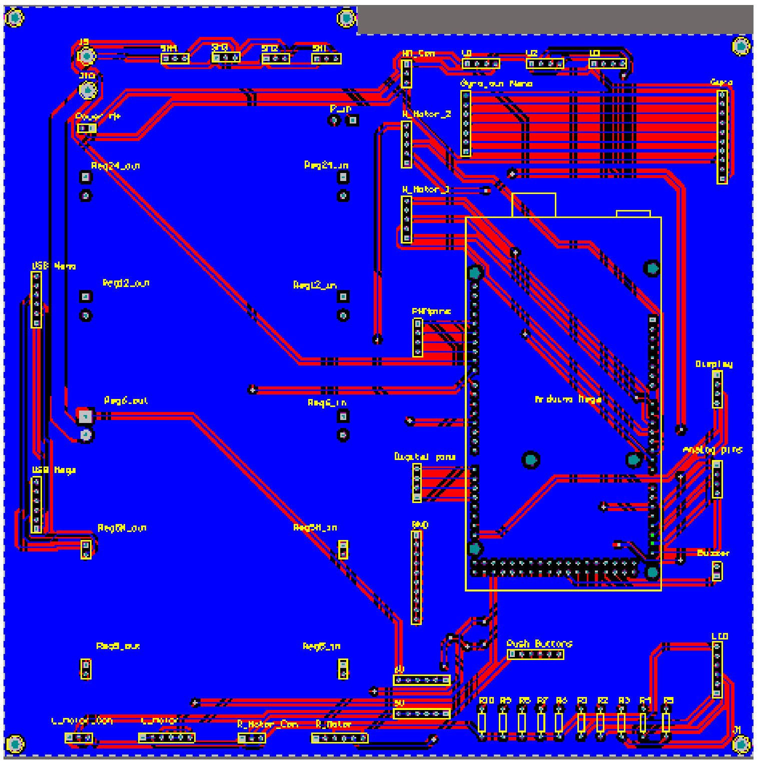


Figure 3.9: PCB design

In this chapter, we described about the process that we followed in the designing of the robot and the PCB. In addition, we provided the images of the robot design and PCB design in this chapter.

4 Results

In this chapter we discuss the results that we obtain so far during our progress in the project. Results are discussed in term of different parts in our project. Those are tree detection, positioning system and navigation system.

4.1 Tree Detection

Figure 4.1 shows the evaluated results of the trained model on some of the images. It can be observed that SSD-MobileNet [11] model performs better both in terms of speed and accuracy. Careful usage of data augmentation techniques might be able to increase the accuracy values in these results up to a certain extent. Note that these results are the initial results of the trained model and these are yet to be further improved by tuning hyper parameters and augmentation techniques.

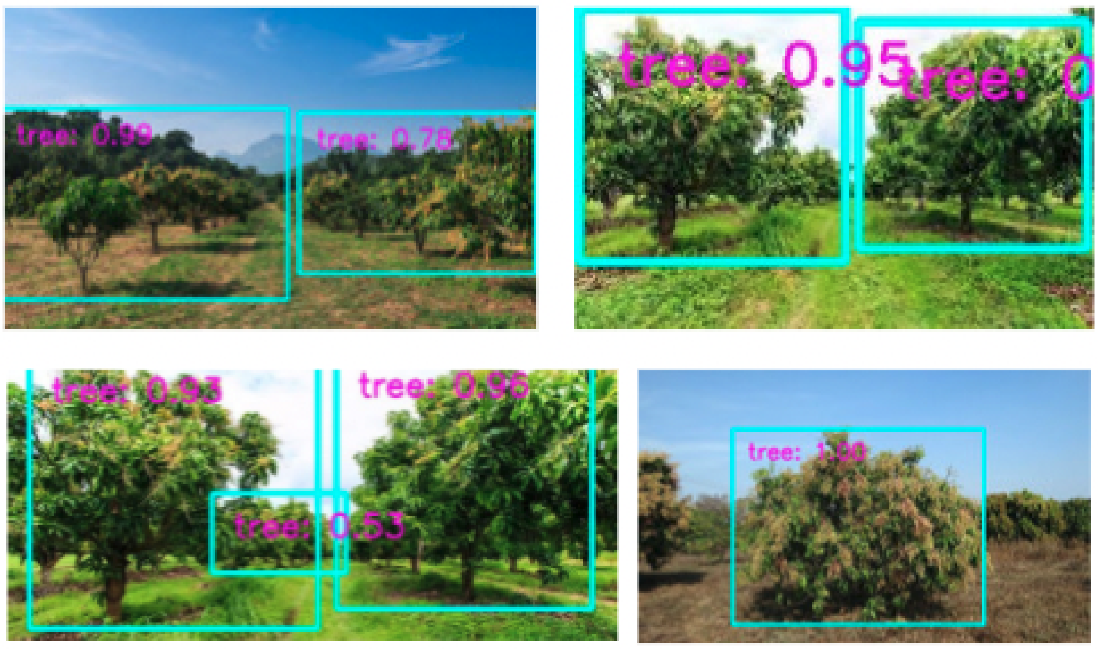


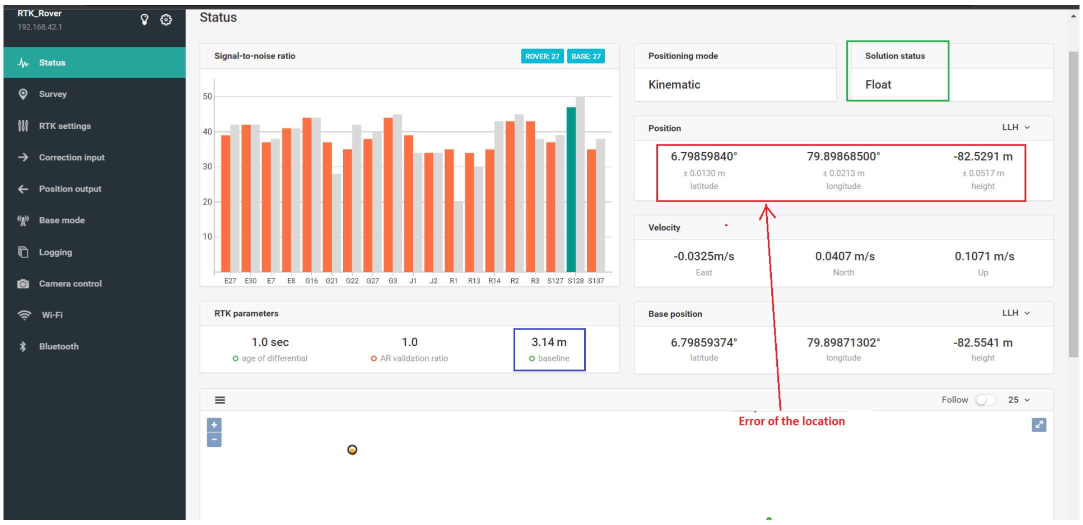
Figure 4.1: Initial detection results

4.2 Positioning System

We use RTK technology for the positioning of our robot accurately in the path in the navigation. Since we are still building over robot, we could not check whether robot is positioned correctly when navigating. Instead of that, we checked the connectivity between the base and the rover M+ modules and checked whether the base transmits corrections to the receiver. In addition, we checked whether rover module calculates its position correctly.

Figure 4.2 shows the status tab of Reach Panel web app of the rover module. The orange or green bars in the graph shows the SNR of connected satellites of rover and gray bars shows the SNR of connected satellites of base. Since SNR of both base station and rover is shown in the status tab of rover module, we can confirm that the connection between base and rover modules is established. Values inside the red box shows calculated location of the rover module and the error of the location of the rover. We could get a lower error around 5cm. Therefore, it is clear that the rover module received corrections from the base station.

After that, we checked whether baseline is shown correctly. Baseline is the distance between the base station and the rover. For that, we kept the rover module about 3m away from the base. Then, Reach Panel web app showed baseline as 3.14m. This is shown inside the blue box in Figure 4.2. Therefore, we could get a value closer to the correct value. Correctness of this value and the location depend on the signal strength that receives from the satellites. Signal strength can be differed due to various reasons such as weather conditions, obstacles around the RTK receivers that could block the clear sky view like buildings, trees, cars, humans, laptops etc. We got solution status as 'Float'. It is shown inside the green box in Figure 4.2. If this solution status is set to 'Fix', we can get more accurate result. However, it depends on the signal strength of satellites. If SNR of a satellite is over 45, it is shown in green color. Otherwise, it is shown in orange color. If we can achieve as many satellites signals in "green zone" as possible, it will set solution status as 'Fix' and it gives more accurate results [9].


RTK_Rover
192.168.42.1

Status

Signal-to-noise ratio — ROVER: 27 BASE: 27

Positioning mode	Solution status
Kinematic	Float

Position — LLH

latitude	longitude	height
6.79859840° ± 0.0130 m	79.89868500° ± 0.0213 m	-82.5291 m ± 0.0517 m

Velocity

East	North	Up
-0.0325m/s	0.0407 m/s	0.1071 m/s

RTK parameters

age of differential	AR validation ratio	baseline
1.0 sec	1.0	3.14 m

Base position — LLH

latitude	longitude	height
6.79859374°	79.89871302°	-82.5541 m

Figure 4.2: Results of RTK

In addition of above, we checked whether when we move the rover module, the moving path will show correctly in the Reach panel web app of the rover module. Since we are still developing our robot, we checked that by moving the rover M+ module manually. Figure 4.3 shows the resulting path when we moved the rover module. It shows that the path is displayed correctly.

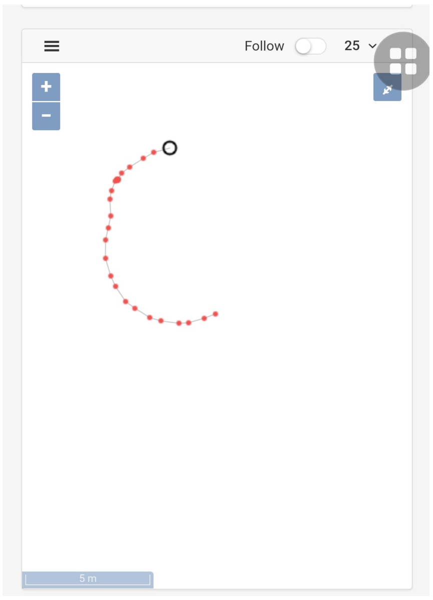


Figure 4.3: Path of rover module

4.3 Navigation system

Figure 4.4 shows how the above navigation algorithm works in the gazebo simulation environment. The gazebo provides GPS coordinates to the respective ROS node to integrate it with the encoder and IMU values to get accurate positional data. The path between starting and ending positions is shown in the yellow color line. Green lines depict how the reference point moves along the line.

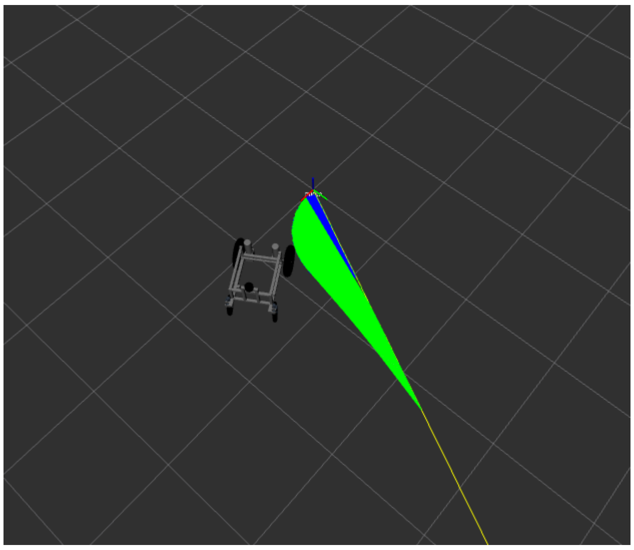


Figure 4.4: Result in gazebo environment

We discussed in this chapter about the results that we have obtained so far in our project. These were discussed under different areas of our project such as tree detection, positioning system and navigation system.

5 Conclusion

In this robot, what we are trying to achieve is to replace the existing fertilizing system in Nelna farm with an automated system. Ultimately, it will increase the annual mango harvest as well. In this prototype version, we are trying to imitate what we can achieve in the production level output. We are carrying a 10 liters fertilizer tank with the robot and it can cover up to 45 trees with 200 milliliters per tree. Since we can change the fertilizer volume through our system, this can be used for potted plants which need less amount of fertilizer as well.

This is the first time in Sri Lanka that RTK technology is used for industrial agricultural robots. Having that technology in our robot, we can guarantee that the plants or trees around the path that the robot is moving won't be damaged. As mentioned above, we have taken some precautions for the RTK signal drops as well.

The tree detection system will ensure that the robot will add fertilizer only after detecting a nearby tree. We cannot see this feature in most of the existing fertilizing robots. In addition to that, the robot is continuously scanning the environment for any obstacles to avoid any kind of crash.

After considering all the above-mentioned technologies that we use, we can conclude that this approach will suit more than enough for fertilizing. We have now tested each and every feature separately in the simulation environment and we need to combine them into the physical robot platform and test as a whole.

Bibliography

[1] "XAG R150," https://www.xagaustralia.com.au/r150, [Online].

[2] J. W. Brian Donahue and R. Berg, "Guidelines for rtk/rtn gnss surveying in canada," in *Earth Sciences Sector, General Information Product 100-E*, 2013.

[3] "Advantages and Disadvantages of GPS," https://www.geeksforgeeks.org/advantages-and-disadvantages-of-gps/, [Online].

[4] "Getting Started with Reach M2/M+," https://emlid.com/support/reach-m/, [Online].

[5] "Mission Planner," https://ardupilot.org/planner/, [Online].

[6] "Placing the Base," https://docs.emlid.com/reach/tutorials/basics/placing-the-base, [Online].

[7] "Base and Rover Setup," https://docs.emlid.com/reach/quickstart/base-rover-setup/, [Online].

[8] "Reach Panel," https://docs.emlid.com/reachrs/reach-panel/index, [Online].

[9] "Status Tab of Reach Panel," https://docs.emlid.com/reachrs/reach-panel/status, [Online].

[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[12] S. Park, J. Deyst, and J. How, "A new nonlinear guidance logic for trajectory tracking," in *AIAA guidance, navigation, and control conference and exhibit*, 2004, p. 4900.