Mittuniversitetet
MID SWEDEN UNIVERSITY

Environment Explorer -Enex

Navigation of a wheel-based robot

Dashnyam Ganbat, Wen Qi and Ziqin Tang

Course: Electronics MA, Specialization Project, 9 credits

Supervisor: Benny Thörnberg

Abstract

An autonomous navigation system on ROSbot 2.0 has been developed in this report. This work presents the result of one of those three subgroups that has been collaborated together for a Specialization project course at Mid Sweden University. The main purpose was to navigate to predefined waypoints, collect environmental data using sensors such as CO2, air temperature and humidity sensors. The implementation made on Rosbot 2.0 an autonomous mobile robot platform controlled by the CORE2-ROS. A A Real-Time Kinematic (RTK) Global Navigation Satellite System (GNSS) receiver has been used to get the current position data in order to make the autonomously navigating part possible. By defining the distance in two axes x and y, an angle is formed. The initial angle has been kept by correcting the angle that occurs during the movement using Proportional Integral Derivative (PID) controller. A Rplidar sensor has been used to avoid any obstacle during the movement. The robot has successfully found the way point several times with accuracy of 0-0.9 m with and without correction data but the GNSS-reciever still has low accuracy depending on the environment. It has been seen from the experiment that the sensor was weak for urban environment whereas the accuracy is almost always better in freer field such as football field. The report concludes that it is not possible to achieve 2.5cm accuracy due to factors such as buildings and weather. Anyway, it has been used simplest methods to achieve the goal namely navigate autonomously to at least one waypoint. Furthermore, one can develop this navigating system by adding methods such as Kalman filter to reduce the noise, inertial navigating system using IMU sensor.

Table of Contents

1 Introduction

To measure environmental data continuously, helps us humans to realize how the environment changes these days and how much we are affecting our environment by industry, flight etcetera. Furthermore, it can help us to give suitable feedback that we can use in order to improve our lifestyles and the ways of acting. For instance, we can consider reducing the emissions of our industry, if there is sufficient historical data that shows that can be concluded that the emissions are the affecting factor. In this project there is a wheel-based robot environmental explorer, constructed and implemented. The robot autonomously navigates to several waypoints and measures the environmental data at every single way point. The measured data in turn is sent to a web server in order to visualize the data for the end user to see the data in real time data and even the historical data.

The project is divided by three subproject Navigation, Sensor measurement and Data visualization. As it sounds the Navigation group aims to navigate to those predefined way points, Sensor measurement measures the environmental data, finally Data visualization aims to collect the data and store it for visualization. The collected data is visualized on a geographical map similar to Google Maps, where the waypoints are located according to the data from the position sensor. In this report the navigation part is presented and the communication between the other parts.

An autonomous navigation system is the heart of this project. It uses RTK-GNSS position sensor to get current position data and measures along its way to the waypoint. At the same time, it calculates the differences in angle which is controlled by the PID controller.

Objectives

- Implement an autonomously navigation system using wheel-based robot ROSbot 2.0 and RTK-GNSS position sensor.
- To test and verify the accuracy of the position sensor.

Verifiable goals

- To navigate the robot autonomously to at least one waypoint using PID.
- To make a scientific experiment on the sensor in order to know the accuracy.

Limitation

This project is limited to test the navigation part using SparkFun GPS-RTK Board-NEO-M8P-2 GNSS sensor on ROSbot 2.0.

Outlines

The outlines of this report are short described below:
Theory: In this section it presents relevant information it needs to implement the navigation system
Method: In this section it presents the method of this project, the choice of scientific experiment in order to test the accuracy of the sensor.
Result: in this section it presents the results of the chosen scientific method in form of graph and table which is the actual output from the Method section
Discussion: in this section it discusses if the result achieved the goal.

Contribution

The contribution of this project is divided into three persons Dashnyam Ganbat, Ziqin Tang and Wen Qi. Wen Qi is working with the GNSS part where he connects to the base station in order to get the correction data for the position sensor. He is also calculating the data from the GNSS sensor that becomes in the relevant unit that makes the navigating part possible. Dashnyam Ganbat works with the navigating part. He is implementing the PID controller on the differences between the start point and target point that occurs during the robot movement and controls the angular speed of the robot. Student Ziqin Tang is working with the Rplidar part that is used to avoid any obstacle that occurs during the movement to the way point.

2 Theory

In this section it summarizes the relevant background information of this project.

GNSS Receiver

GNSS (Global Navigation Satellite System) is a system that uses satellites to provide autonomous geo-spatial positioning. The system can be used for providing position, navigation or for tracking the position of something fitted with a receiver. [1]

GNSS receiver receives and processes the signal coming from GNSS satellites constellations. All GNSS systems utilize the concept of time-of-arrival (TOA) ranging to determine the position of receiver. GNSS receiver measures the transmitting time and the time interval of a GNSS signal to travel between satellite and receiver. The distance between the GNSS satellite and the receiver can be calculated by multiplying the measured time by the speed of light. [2]

In this project, we use SparkFun GPS-RTK Board-NEO-M8P-2 as GNSS receiver. Figure 1. The NEO-M8P-2 module is the top-of-the-line module for high accuracy GNSS and GPS location solutions. [3] Since the error from atmospheric effect, the TOA measurement is not precise. RTK is the perfect solution to this problem.

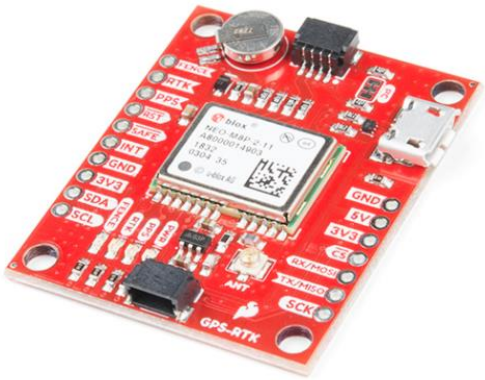

Figure 1 SparkFun GPS-RTK Board-NEO-M8P-2

Real-Time Kinematic

Real-time kinematic (RTK) positioning is a satellite navigation technique used to enhance the precision of position data derived from satellite-based positioning systems. It uses measurements of the phase of the signal's carrier wave in addition to the information content of the signal and relies on a single reference station or interpolated virtual station to provide real-time corrections, providing up to centimeter-level accuracy. [4]

The RTK system consists of a single base station receiver and multiple mobile units. During operation, the base station continuously observes GPS satellites, and the observation data and station information are sent to the rover station in real time through radio transmission equipment. The rover station receives the data link from the base station while receiving GPS satellite signals and collecting satellite data. Then processes the two sets of data by carrier phase differential processing and calculates the three-dimensional coordinates and accuracy of the rover station in real time.[5]

Use RTK technology to take advantage of the spatial correlation of the observation errors between the base station and the rover and remove most of the errors of the rover observation data by difference, so as to achieve high-precision positioning.

Base station

As said before, RTK base station acts as a reference station in an RTK setup. The core of the base station comes from the REACH M + RTK GNSS model produced by EMLID. To calculate centimeter-precise coordinates in PPK and RTK, Reach needs corrections from a base station. It could be either another Reach receiver or an NTRIP service. VRS is also supported. See fig. 2

In the project, Miun has a RTK base station and spitting out data on a fixed IP address: 108.61.171.128 on port 29000 using TCP and data format is RTCM3.

Figure 2 EMLID REACH M series

ROS

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. [6]

The architecture of a ROS system consists of five components: a ROS Master, nodes, publishers, subscribers, and topics. See figure below.

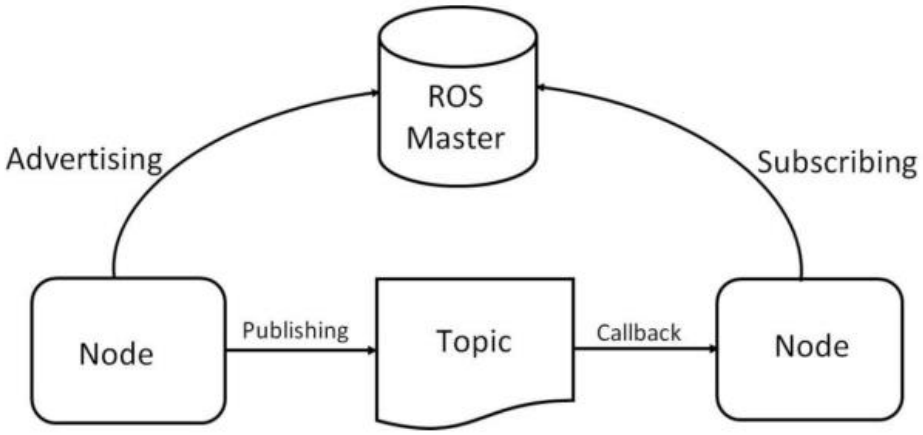

Figure 3 Typical ROS model

The ROS Master is responsible for managing names and registration services to the nodes within a ROS system. Publishers and subscribers are monitored by the ROS Master to ensure associated topics, as well as services, are provided within the robotic system. The ROS Master also enables location and communication between nodes within the robotics system. Finally, the ROS Master commonly initiates the node communication function using the **roscore** command. The roscore command is used to load the ROS Master with the essential software components to allow communication between nodes.

Node is an executable file within the ROS system to allow communication among another node.
A message that is transmitted by a node or topic within a ROS system is known as a publisher and the received one is known as a subscriber. The publishing and subscribing of a message of a specific name type is known as a topic.

Each of these software components allows a robotic system to move, sense, monitor, and process a variety of signal and imaging data.[7]

Husarion ROSbot 2.0

Figure 4 ROSbot 2.0

ROSbot 2.0 is a 4x4 drive autonomous mobile robot platform controlled by the CORE2-ROS double layers board. It is an affordable robot platform for rapid development of autonomous robots. It can be a base for custom service robots, inspection robots and robots working in swarms. [8]

The ROSbot2.0 integrates several function models：

- 4-wheels mobile platform containing DC motors with encoders and an aluminum frame
- Orbbec Astra RGBD camera
- MPU 9250 inertial sensor (accelerometer + gyro)
- RPLIDAR A2 laser scanner

In this project, we mainly use DC motors and RPLIDAR.

CORE2 - ROS

Figure 5 CORE2 - ROS

The CORE-ROS board consists of two layers boards, the upper board is "CORE2 board" and the lower one is "Tinker board". See figure 5.

(1) CORE2 board
In short, it can be defined as an elements control platform. Receiving the transitional velocity and rotational velocity calculated from the Tinker board, and control the motors make the wheels move and turn with these two parameters. The rover station board to be mentioned later is connected to the Core2 board, and all the data from the base station is transmitted to CORE2 through WiFi and then to the rover station.

(2) Tinker board (single board computer)
The Tinker board is An Arm-based Single Board Computer (SBC), 64-bits processor, offers enhanced computing performance with low power consumption. [9]
For this board, it's a small computer with Raspberry pi operating system. We mainly program this board, write all algorithms and driving methods into this board, make it finally calculate the linear speed and angular speed of the wheel, transmit them to the CORE2 board, and then operate the ROSbot to complete the navigation work.

In ROSbot, the graphic of components and connections as figure 6.

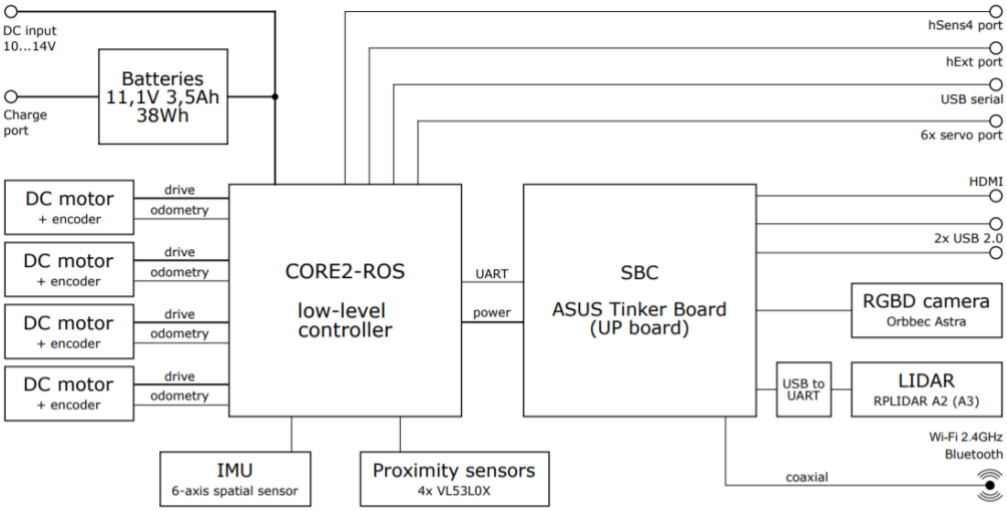

Figure 6 Graphic representation of ROSbot 2.0 components and connections

Power Supply

ROSbot is powered from an internal, rechargeable Li-Ion battery pack that contains 3 Li-Ion cells, connected in series. This type of connection is called "3S".

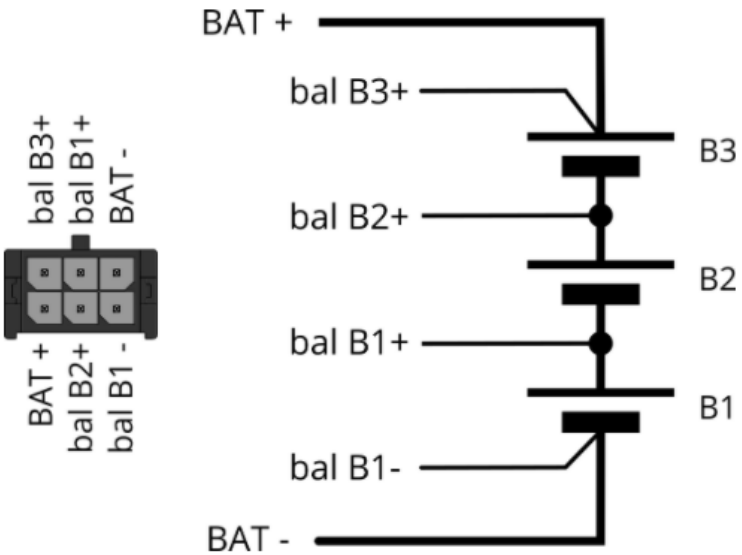

Figure 7 Charging connector and cells

In the figure 7, the BAT+ and BAT- are the power connections and the "bal Bxx" wires are used to monitor the voltage on each cell.

Rplidar sensor

Introduction

Rplidar is a low-cost LIDAR sensor suitable for indoor robotic SLAM application. It provides a 360-degree scan field, 5.5hz/10hz rotating frequency with guaranteed 8-meter range distance, current more than 16m for Rplidar A2 and performs high speed distance measurement with more than 4K/8K samples per second. [11]

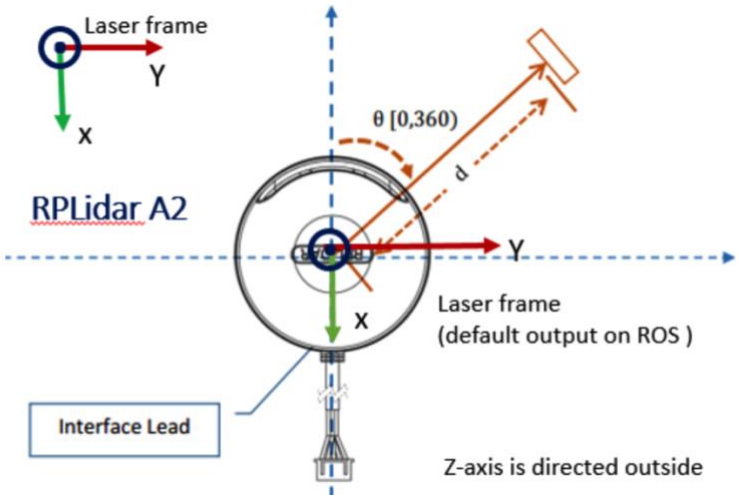

Figure 8 RPLIDAR

Principle of Laser Triangulation

As a low-cost laser radar design solution, the laser triangulation method can obtain high-precision and cost-effective application effects. The four core components of laser radar:

- Laser: Laser is the laser emitting mechanism in lidar. During work, it will light up in pulses.
- Receiver: After the laser emitted by the laser hits an obstacle, the reflected light will be converged on the receiver through the lens group through the reflection of the obstacle.
- Signal processing unit: The signal processing unit is responsible for controlling the emission of the laser and processing the signal received by the receiver. Based on this information, the distance information of the target object is calculated.
- Rotating mechanism: The above three components constitute the core part of the measurement. The rotating mechanism is responsible for rotating the above-mentioned core components at a stable speed, so as to realize the scanning of the plane and generate real-time plan information.

RPLIDAR adopts the oblique laser triangulation method. The laser emitted by the laser is incident on the surface of the object to be measured at a certain angle

to the normal of the surface of the object. The reflected (scattered) light is converged and imaged by the lens at B, and finally collected by the photosensitive unit.The figure 9 below shows it's working principle.

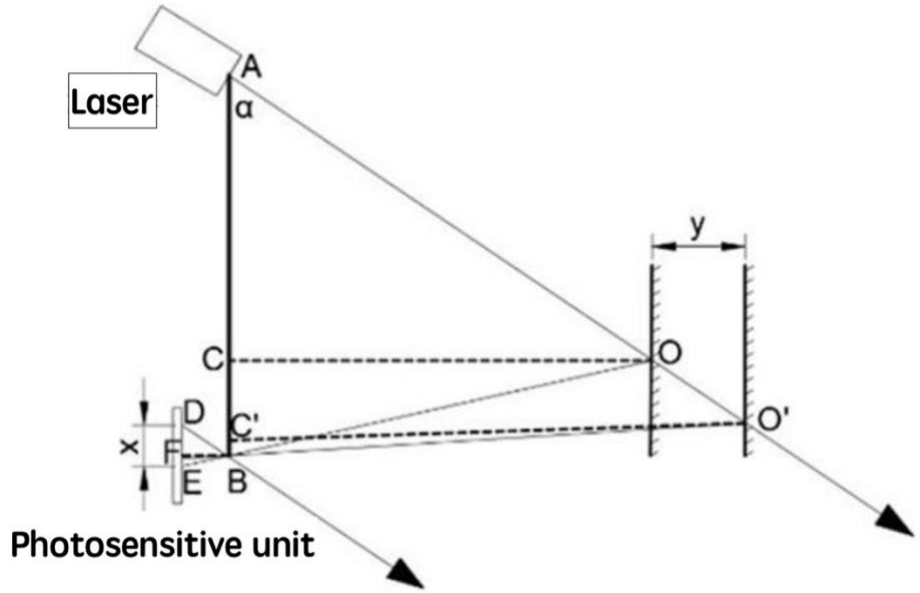

Figure 9 Laser triangulation method

It can be seen from the figure that the angle between the incident light AO and the baseline AB is α, AB is the distance between the laser center and the CCD center, BF is the focal length f of the lens, and D is the image of the reflected light on the photosensitive unit when the measured object is infinitely far away from the baseline. The extreme position. DE is the displacement of the light spot from the limit position on the photosensitive unit, denoted as x. When the optical path of the system is determined, α, AB, and f are all known parameters. From the geometric relationship in the optical path diagram, we can see that $\triangle ABO \backsim \triangle DEB$, there is a side length relationship:

$$\frac{AB}{DE} = \frac{Oc}{BF} \qquad AO = \frac{OC}{\sin\alpha}$$

Based on Silan Technology's unique laser ranging engine, it can perform ranging actions up to 8000 times per second, with a ranging radius of 8 meters and an angular resolution of up to 0,225°. During each ranging process, the RPLIDAR series lidar will emit a modulated infrared laser signal. The reflection of the laser signal after it hits the target object will be received by the RPLIDAR visual acquisition system, and then processed by the DSP embedded in the RPLIDAR. The device calculates in real time, the distance between the illuminated target object and the RPLIDAR and the current angle information will be output from the communication interface.

Rplidar Node

In the ROS system, Rplidar has a specific rplidarNode . RplidarNode is a driver for RPLIDAR. It reads RPLIDAR raw scan results using RPLIDAR's SDK and converts to ROS LaserScan messages.

- Published topics
    1. scan (sensor_msgs/LaserScan) It publishes a scan topic from the laser.
- Services
    1. stop_motor (std_srvs/Empty) Call the service to stop the motor of rplidar.
    2. start_motor (std_srvs/Empty) Call the service to start the motor of rplidar.
- Parameters
    1. serial_port (string, default: /dev/ttyUSB0) Serial port name used in our system.
    2. serial_baudrate (int, default: 115200) Serial port baud rate.
    3. frame_id (string, default: laser_frame) Frame ID for the device.
    4. inverted (bool, default: false) Indicated Whether the LIDAR is mounted inverted.
    5. angle_compensate (bool, default: false) Indicates whether the driver needs to do angle compensation.
    6. scan_mode (string, default: std::string()) The scan mode of lidar.

3 Methods

In this section it presents firstly, the scientific experiment of the GNSS-receiver in order to know the accuracy. In the scientific experiment part, it introduces the sensor part, background information, the scientific question, hypothesis of the test, prediction of the test and the test of that prediction. Secondly the method of the usage of the GNSS receiver in order to get the position data. Thirdly about the PID-controller which is used to control the angular speed of the robot and finally about the Rplidar part that is used to avoid any obstacle.

An overall flowchart of this project and more specific flow chart is presented in Appendix 1.

Scientific Experiment

It is most necessary to test and verify the accuracy and precision of the GNSS receiver, since it is used to detect whether the robot is near to the waypoint or not. Not least to measure the angular between current point and the target point. Therefore, it does several tests on the receiver to know more about the accuracy of the sensor in reality. See below for the description of the experiment:

Introduction

The GPS-RTK position sensor is used in this project in order to navigate to several waypoints. The sensor has an accuracy 0.025m with correction data and 2.5m without correction data according to its datasheet. It is suitable then to test its accuracy in reality in order to know how it works in reality.

The self-navigating robot is used for this experiment to move between several pre-defined waypoints and the waypoints are defined using the GNSS-sensor (GPS-15005 ROHS).

Question: How far the robot stops from the waypoints? What is the accuracy in reality?

Hypothesis: The robot should stop at a maximum 2.5cm from the current waypoint according to the data sheet.

Prediction: To make several tests to drive the robot to waypoint should give us suitable outcomes that can be analyzed.

Test of prediction: We make different tests such as driving along a straight line, "Urban environment and free field" with and without base station to come to a waypoint and observe the accuracy of the sensor. The outcome records also when the sensor doesn´t move, namely the sensor has the same position all the time in order to find out the accuracy. Finally, we will iterate the testing process until the outcome is enough to be analyzed.

Initial test of the sensor

Before it starts with these tests, it is most suitable to test the accuracy of the sensor that has a constant position. The test is done with and without correction data from a base station. This test aims to know the initial accuracy. The result of all tests shown in result section

GNSS

In order to realize the navigation function, we need to know the current position of the ROSbot. Through GNSS receiver, we can directly receive NMEA data which contains time, latitude, longitude
etc. To achieve high-precision positioning, we also need to enable the RTK function that we need to feed correction data from base station to rover station. The data transmission flow shows in figure 11.

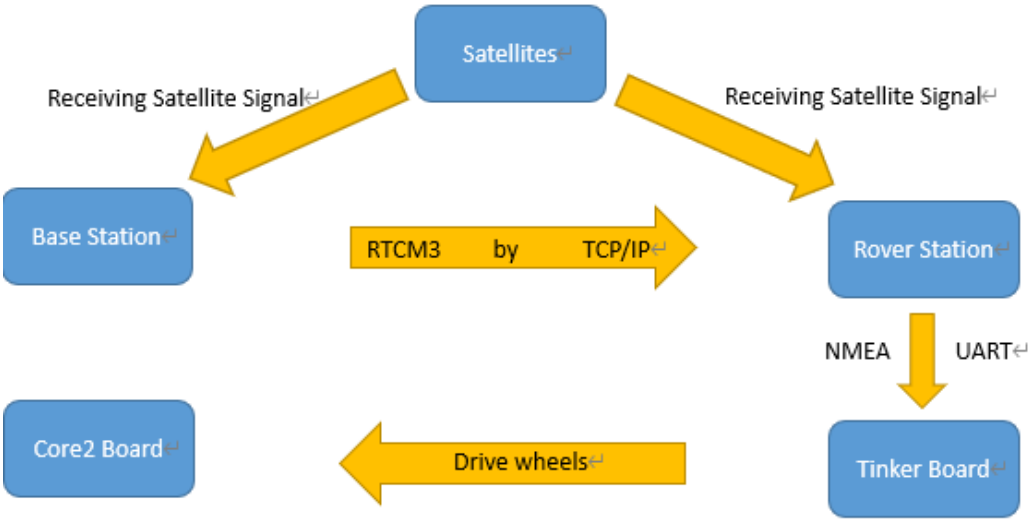

Figure 10 Data transmission flow

First, we need to configure the GNSS board. Connect the GNSS board to our computer via USB and enter U-center. In the software, we need to enable RTCM3 data, increase frequency and baud rate which can receive more data's.
Then we connect the GNSS board to ROSbot. Since the GNSS board already has RTK function, we do not need to decode RTCM3 data and combine it with NMEA data. We just need to create a socket to communicate with the base station and write the data which we received from base station to the board.
After that, we can read the serial port to get the corrected positioning data. However, NMEA data contains a lot of information, we cannot use it directly and we need to decode it. There are many sentences defined in the NMEA-0183 protocol, but the commonly used or most compatible sentences are only $GPGGA, $GPGSA, $GPGSV, $GPRMC, $GPVTG, $GPGLL, etc. Different

sentences contain different information. Our main concern is time, latitude, longitude and dilution of precision. So, we only receive GGA type data which contains all the data we need.

The GGA data like that : $GPGGA , 092204.999 , 4250.5589 , S , 14718.5084 , E , 1 , 04 , 24.4 , 12.2 , M , 19.7 , M , ,0000*1F

[0]: $GPGGA, Sentence ID, indicating that the sentence is Global Positioning System Fix Data (GGA) GPS positioning information
[1]: 092204.999, UTC time, hhmmss.sss, hour, minute, and second format
[2]: 4250.5589, Latitude, ddmm.mmmm, degree and minute format (if the leading digit is insufficient, add 0)
[3]: S, Latitude N (north latitude) or S (south latitude)
[4]: 14718.5084, Latitude, ddmm.mmmm, degree and minute format (if the leading digit is insufficient, add 0)
[5]: E, Longitude E (east longitude) or W (west longitude)
[6]: 1, GPS status, 0 = not available (FIX NOT valid), 1 = single point positioning (GPS FIX), 2 = differential positioning (DGPS), 3 = invalid PPS, 4 = real-time differential positioning (RTK FIX), 5 = RTK FLOAT, 6 = Estimating
[7]: 04, The number of satellites in use (00-12) (If the leading digit is insufficient, add 0)
[8]: 24.4, HDOP level precision factor (0.5-99.9)
[9]: 12.2, Altitude (-9999.9-99999.9)
[10]: M, Unit: M (meter)
[11]: 19.7, The height of the earth ellipsoid relative to the geoid WGS84 level division
[12]: M, WGS84 level division unit: M (meter)
[13]: Differential time (the number of seconds since receiving the differential signal, it will be empty if it is not for differential positioning)
[14]: Differential station ID number 0000-1023 (If the leading digit is insufficient, add 0, if it is not differential positioning, it will be empty)
[15]: 0000*1F, Check value

In the decoding function, we first find the beginning of the sentence through "$". Then the content between the two commas is stored in the string container, so that the NMEA data is separated. Finally, take out the data we need, store it in the array, and broadcast it through the publisher.

In the navigation node, through subscribe function, we can receive the current longitude and latitude information of ROSbot. But we need to know the distance and direction from the waypoint which means we should convert degree to meter.

Earth is not a perfect sphere but closer to an ellipsoid. But if our robot is navigating only within a tiny part of the globe Earth, then a constant scale factor between latitudes and dns and another scale between longitudes and dwe can be used. We used the conclusion of Benny Thörnberg who used the chart of Sundsvall Bay and calculated that 1° on latitude equals 111180m and 1° on longitude equals 51600m.[10]

Then we use longitude as the x-axis and latitude as the y-axis to construct a Cartesian coordinate system. Calculate the angle between the current position and the waypoint by trigonometric function. At the starting point, ROSbot will record the angle between the initial position and the target position, and constantly compare the difference between the angle of the current position and the initial angle during movement and realize navigation through the PID controller.

PID controller

Proportional-Integral-Derivative controller is used to correct the angle difference between initial angle and current angle that occurs during the movement. The result of the PID controller shall correct the heading of the robot to the target point. It implies that if the initial angle holds the same during the travel, it means that the robot is on a straight line to the target point. PID controller has three dependent values namely proportional gain, integral gain and derivative gain. The constants are selected after a few tests.

A short description of the working flow of the robot movement to the way point is shown below:

PID controller

- Input: current angular that calculates using the triangle,
- Incremental: Angular speed in z axis.
- Current position measures
- Distance, Delta y, Delta x calculates see fig 11.
- Starts to drive forward 3 second in order to get angle difference
- Correcting the angle with PID and drive forward at the same time
- See fig. 12 an ideal figure of PID controller

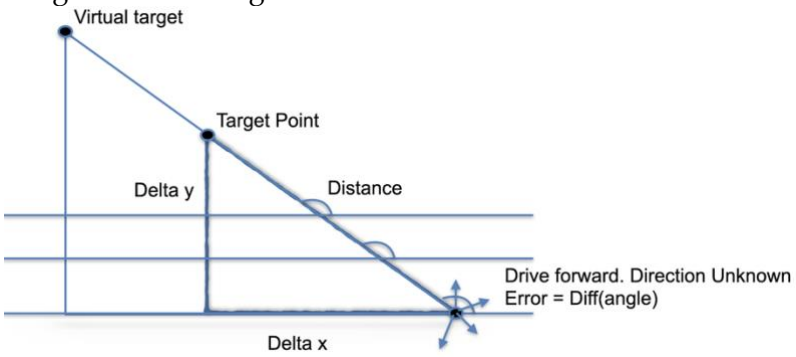

Figure 11. Illustration of angle calculation

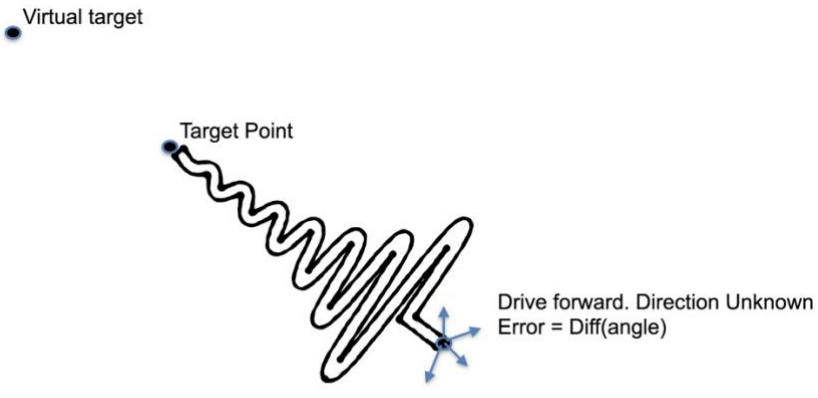

Figure 12. Illustration of PID controlled path to a waypoint

Rplidar working flow

Rplidar is used to detect the distance and angle information of the surrounding environment or obstacles. The figure 13 below shows its working flow.

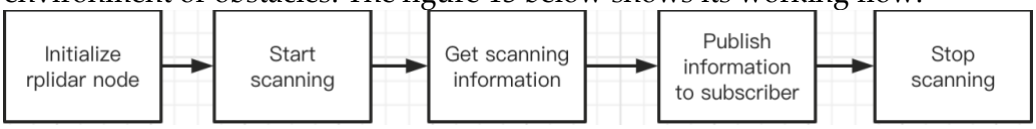

Figure 13 RPLIDAR working flow

It needs to be initialized firstly when it is starting to work. During this process, it needs to determine its serial port number, serial port baud rate, frame ID and angle compensation flag to ensure that it can be started. After initialization, it will create the driver instance, make connections with the ROS system, get rplidar device information, check rplidar device healthy and start motor to scan. During the scanning process, it will get scan time, scan duration, angle and range.

Rplidar sensor

The obstacle avoidance algorithm is used to assist in avoiding obstacles during the navigation process. This part of the content is written in the RPLIDAR subscriber. The angle and distance information received in real time by the RPLIDR is used to determine whether it is necessary to enter the obstacle avoidance. The figure 14 below shows the obstacle avoidance algorithm.

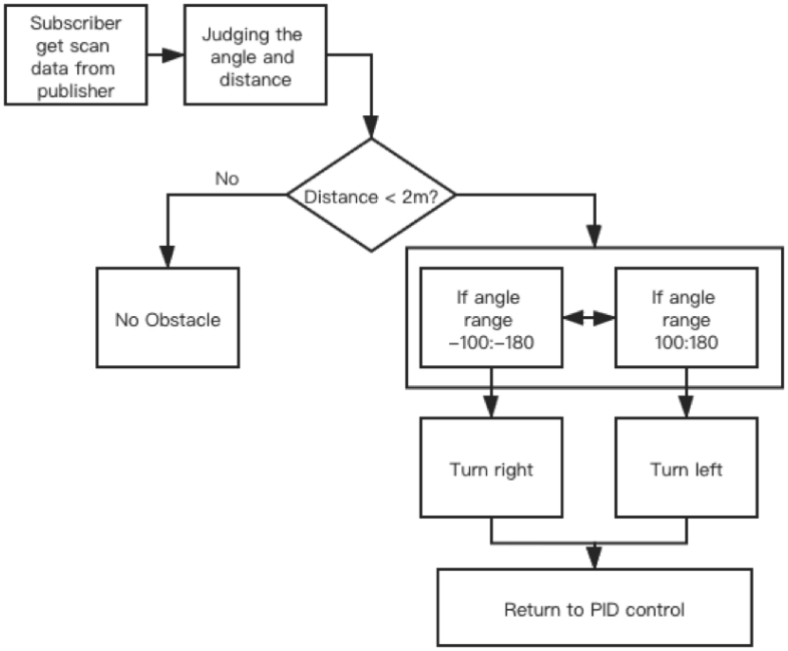

Figure 14 Obstacle avoidance working flow

The condition for entering obstacle avoidance is whether the distance of the obstacle is less than 2 meters. The reason for choosing this distance is that the

range of RPLIDAR is 8 meters and the rosbot has enough distance for steering action. If the distance is greater than 2 meters, it will not enter obstacle avoidance, otherwise it will enter obstacle avoidance. What needs to be explained here is that the obstacle will not be located directly in front of ROSBOT. In real situations, the obstacle is always located in the front left or front right. So, after judging the distance, the angle will be judged If the angle is between -100 and -180 degrees, ROSBOT will turn right at an angular velocity of 0.2. If the angle is between 100-180 degrees, ROSBOT will turn left at the same speed.

The conditions for stopping steering are distance and angle, that is, the obstacle distance is greater than two meters and between -100 degrees and 100 degrees, which means that there is no obstacle in front. After stopping the steering, the RPLIDAR subscriber will issue the value to the PID controller the start message of 1, at this time the GNSS sensor will obtain the current position information and enter the navigation node.

4 Result

Rplidar

The Information coming from the publisher will be published to a subscriber. Figure 15 below shows the data that subscribers received. Scan angle/degree is from -180 to 180, scan range is from 0.15m to 8m.

```
[ INFO] [1609860618.169103334]: [degree :-142.535172,  ranges: 6.170000]
[ INFO] [1609860618.169143001]: [degree :-141.413696,  ranges: 6.113000]
[ INFO] [1609860618.169181500]: [degree :-140.292221,  ranges: 5.999000]
[ INFO] [1609860618.169218541]: [degree :-139.170731,  ranges: 5.893000]
[ INFO] [1609860618.169254416]: [degree :-138.049255,  ranges: 5.847000]
[ INFO] [1609860618.169291748]: [degree :-136.927780,  ranges: 5.746000]
[ INFO] [1609860618.169330831]: [degree :-135.806290,  ranges: 5.664000]
[ INFO] [1609860618.169366997]: [degree :-134.684814,  ranges: 5.574000]
[ INFO] [1609860618.169402872]: [degree :-133.563339,  ranges: 5.517000]
[ INFO] [1609860618.169439330]: [degree :-132.441849,  ranges: 5.457000]
[ INFO] [1609860618.169477537]: [degree :-131.320374,  ranges: 5.380000]
[ INFO] [1609860618.169513120]: [degree :-130.198898,  ranges: 5.339000]
```

Figure 15 Scan result from subscriber

The publisher of rplidar node will publish this information to its subscriber synchronously and we are able to read and process these data information.
In order to determine the relationship between the angle of rplidar and the actual direction, we conducted three detection experiments, which are to place an object close to the front of the lidar, in the direction of the right and left side, and the experimental results we got are: When the obstacle is on the left side, the angle is from -150 to -50 degree. When the obstacle is on the right side, the angle is from 50 to 150 degrees. When the obstacle is in front of the lidar, the angle is from -125 to 125 degrees. Figure 16 below shows the three direction plots.

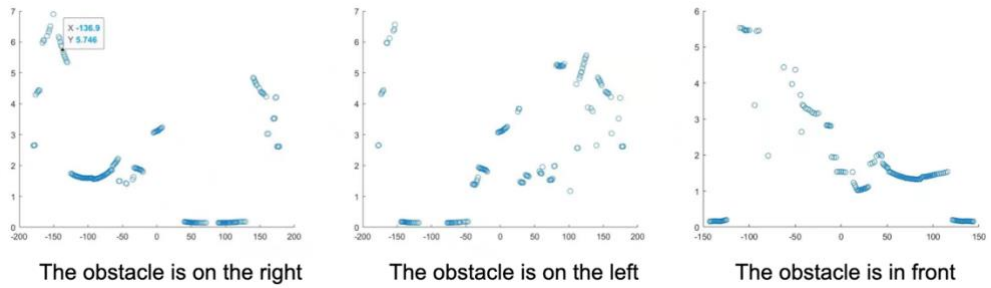

Figure 16 Relationship between the angle of RPLIDAR and the actual direction

So, we can get a conclusion that shows in the figure 17 below.

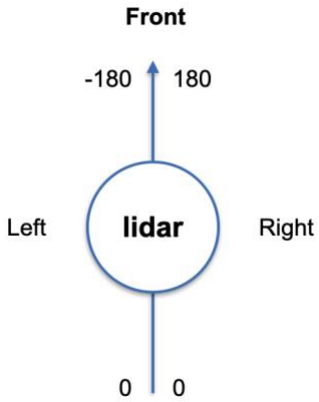

Figure 17 RPLIDAR scan angle

Scientific experiment

Result of the initial test of the sensor see figure 18 below. Firstly, the accuracy tested without correction data from the base station. Secondly the accuracy tested with correction data from the base station. An average of the accuracy is calculated for each test. See the table 1 below.

Table 1. Mean value of the accuracy. Initial test

Longitude without Base Station	Latitude without base station	Longitude with base station	Latitude with base station
0.9540 m	0.5284 m	0.1723 m	0.1408 m

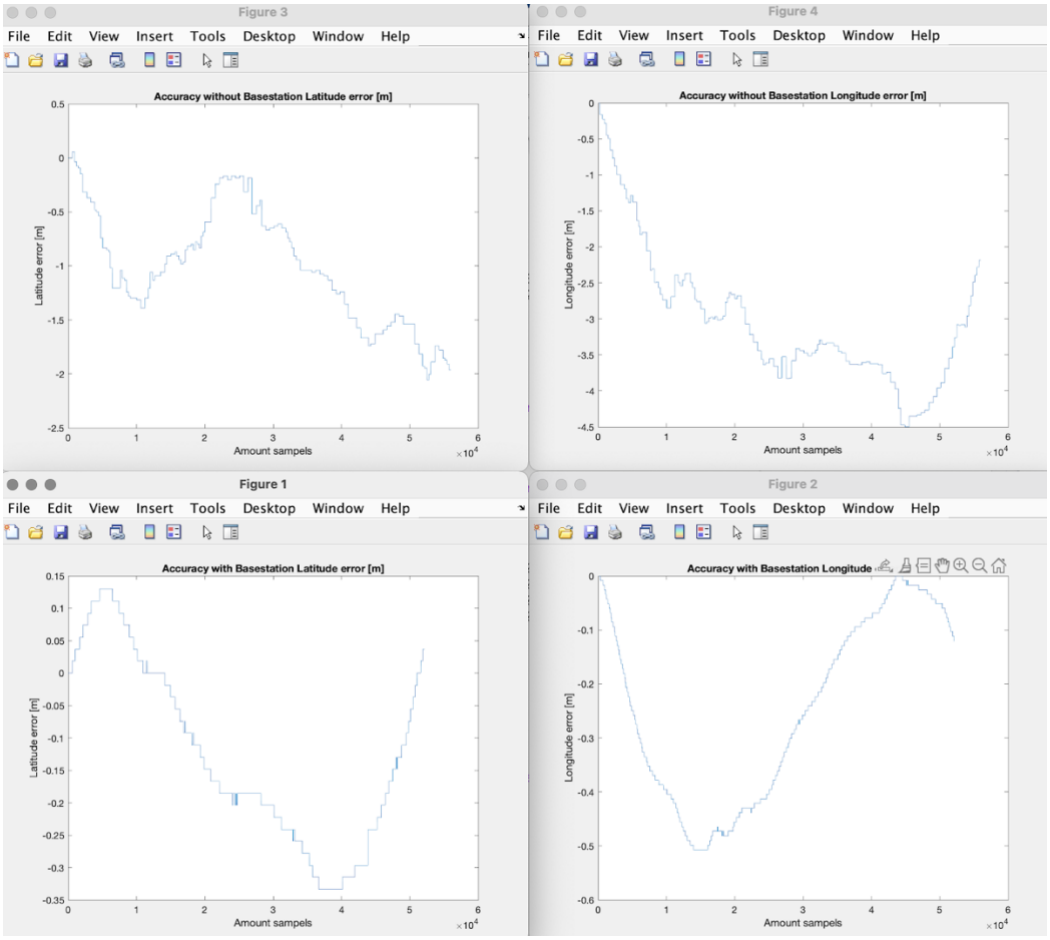

Figure 18 Output from an initial test of the sensor.

Figure 19 below shows the experiment result. There are six waypoints, three of them are the result of base station testing and the rest of them are without base station testing.

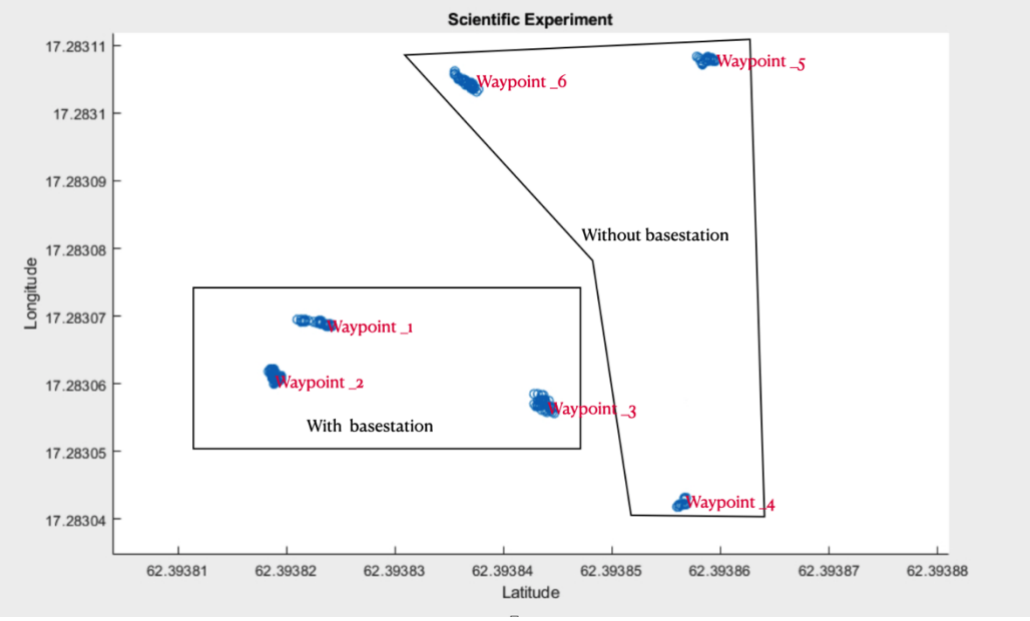

Figure 19 Scientific experiment

Table 2 Scientific experiment result. It shows latitude and longitude ranges and mean values

	Latitude error ranges (m)	Latitude error mean value (m)	Longitude error ranges (m)	Longitude error mean value (m)
1	-0.20- 0.60	0.48	0.25 - 0.40	0.335
2	0.05 - 0.35	0.232	0.25 - 0.60	0.446
3	0 - 0.35	0.23	0.25 - 0.65	0.52
4	0 - 0.45	0.26	0.50 - 0.80	0.645
5	0 - 0.60	0.328	0.02 - 0.22	0.146
6	0 - 0.90	0.52	0.30 - 0.90	0.66

Figures below show the latitude and longitude errors of six waypoints.

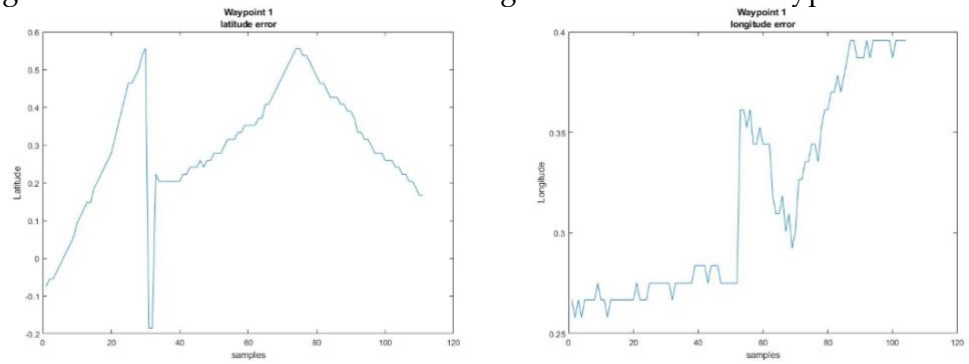

Figure 20 Scientific experiment result of waypoint 1

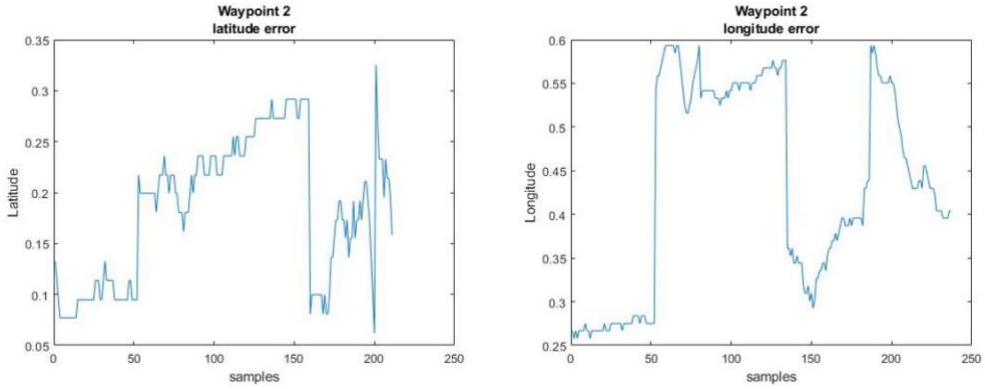

Figure 21 Scientific experiment result of waypoint 2

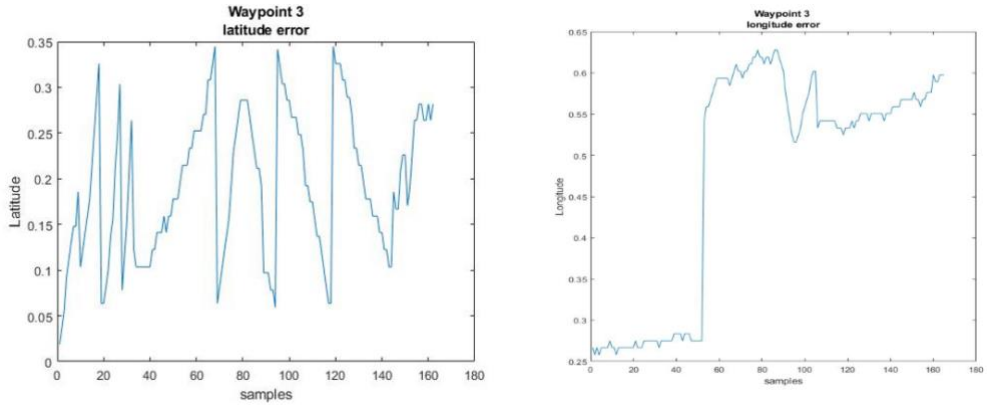

Figure 22 Scientific experiment result of waypoint 3

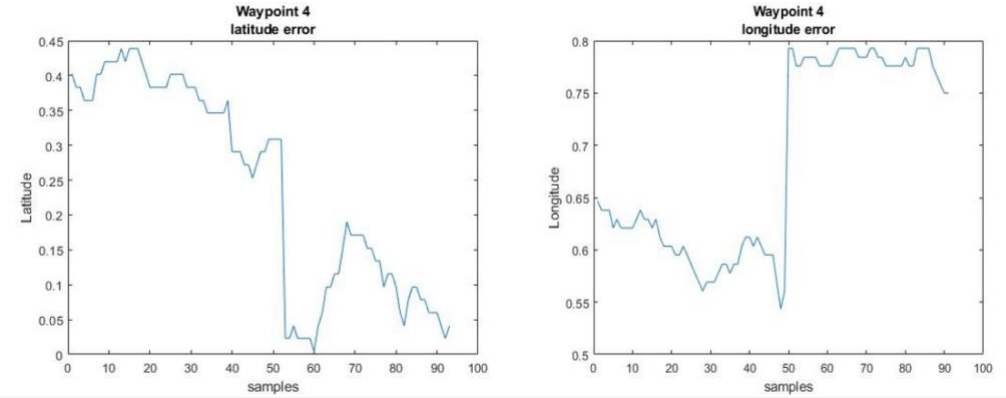

Figure 23 Scientific experiment result of waypoint 4

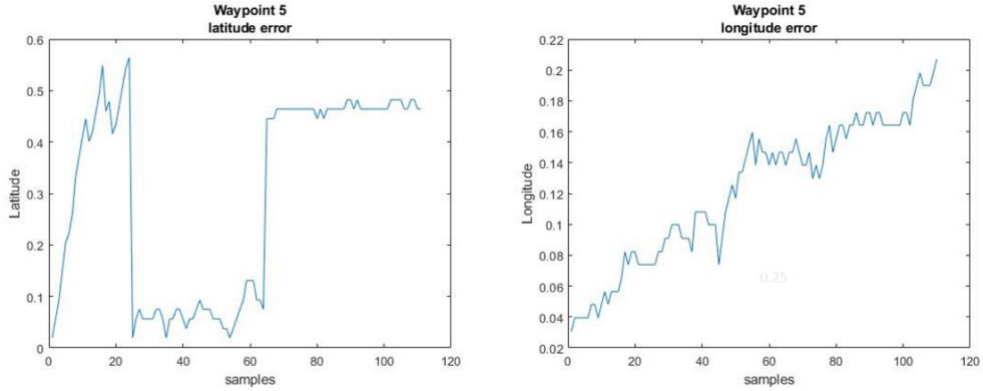

Figure 24 Scientific experiment result of waypoint 5

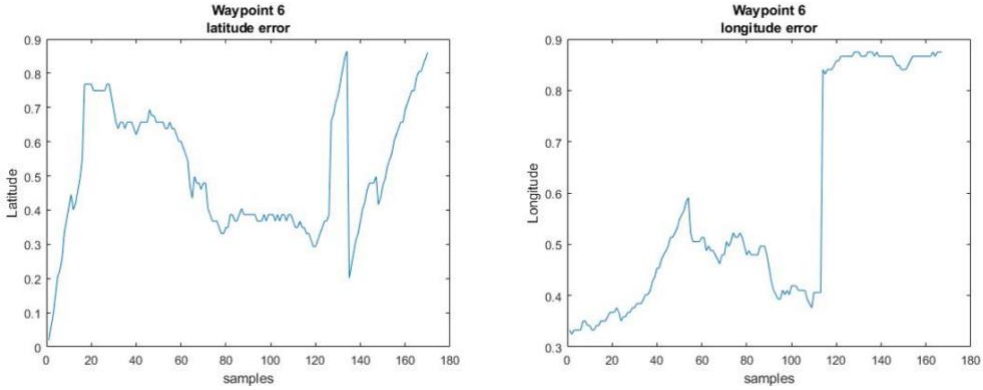

Figure 25 Scientific experiment result of waypoint 6

5 Discussion

In this project, we encountered the following problems：

- When using RTK, the positioning data will not be calibrated immediately, and it takes a while to get accurate positioning data. So, in the beginning, ROSbot's navigation is not accurate. There may be situations that the PID controller cannot control
- Due to factors such as buildings and weather, the rover cannot enter the RTK FIX mode, but can only enter the RTK FLOAT mode. Therefore, the nominal 2.5cm accuracy cannot be achieved
- Due to time constraints, the PID controller cannot be adjusted to the most suitable state, and ROSbot cannot navigate between very close target points (for example, 5m)

We considered the following solutions：

- After receiving the GNSS data at the starting point, the ROSbot will not start to move immediately, but waits for a period of time, and compares the obtained GNSS data, and starts navigation when the error is less than the threshold.
- Navigate as close as possible to the base station and in an open location. Due to the use of TCP for communication, the network needs to be as smooth as possible.
- Use MPU9250 inertial sensor to read the attitude angle of ROSbot. And yaw value will be used as a reference value to avoid missing the target point because the GNSS data is not updated in time or the accuracy is not enough.

6 Conclusion

Finally, the robot is somehow autonomously navigated to at least one waypoint as we´ve seen on demonstration. It has shown that the PID controller was possible solution to correct the angle that occurs during the movement. However, it has been seen from the experiment that the GNSS-sensor was quite sensitive regarding the environment. From the figure 19 we can clearly see that results without base station are further with each other than the result with base station.

In the case of connecting with a base station experiment. Waypoints 1 to waypoints 3. Waypoint 1 and 2 are in the free field, they are obviously closer to each other geographically, according to calculation the straight-line distance between these two points is 1.22 m. Waypoint 1 is more precise in longitude and waypoint 2 is more precise in latitude. For waypoint 3, it is in an urban environment, we can see that it has larger longitude error than waypoint 1 and 2, besides, the distance between it and waypoint 1 is more than 2 meters.

In the case of without base station experiment. Waypoint 4 to waypoint 6. It is possible for us to find out that they are far away from each other. Waypoint 4 and 5 are in the free field and waypoint 6 is in an urban environment.

And from the table 2, we can see that the latitude and longitude errors of waypoints are the largest among these six waypoints.

In summary we can get a conclusion that the best scientific result goes to connect with the base station and also in a free field.

References

[1]https://en.wikipedia.org/wiki/Satellite_navigation
[2]https://en.wikipedia.org/wiki/Time_of_arrival
[3]https://www.sparkfun.com/products/15005
[4]https://en.wikipedia.org/wiki/Real-time_kinematic
[5]https://zh.wikipedia.org/wiki/%E5%AE%9E%E6%97%B6%E5%8A%A8%E6%80%81%E6%8A%80%E6%9C%AF
[6]http://wiki.ros.org/ROS/Introduction
[7]https://www.designnews.com/gadget-freak/ros-101-intro-robot-operating-system
[8]https://husarion.com/manuals/rosbot/
[9]https://tinker-board.asus.com/product/tinker-board-2.html
[10]http://apachepersonal.miun.se/~bentho/sp/download/gnssHowTo.pdf
[11]http://wiki.ros.org/rplidar
[12] http://georgepavlides.info/research/LaserScanningAndTriangulation.php

Appendix 1

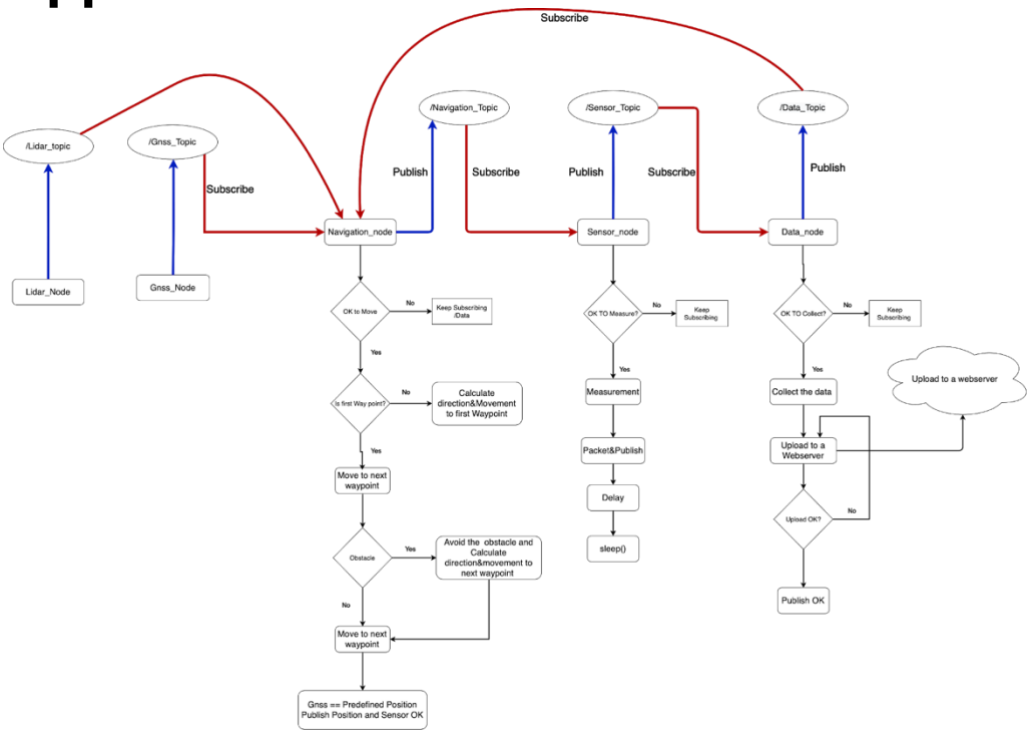

Figure 26. An overall description of the navigation system