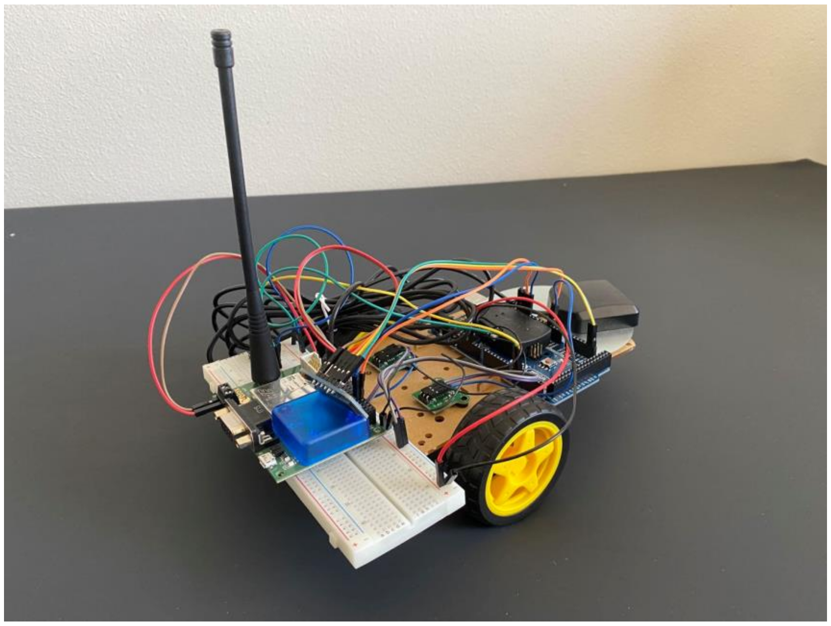


FINAL REPORT

AIRTuB Location System

Group1

Jiacong Li

Yiran Ding

Pinqi Guo

June 6 2020

Time: Feb 2020 to Jun 2020

Group Order: Group 1

Project Topic: AIRTuB Renewable Energy – Redesign of crawler position

Client:

Jos Gunsing

jos.gunsing@marometech.nl

Coach:

Edward Mouw

mouw0003@hz.nl

Group members:

Jiacong Li

00082749

1179285759@qq.com

Yiran Ding

00082781

2421475968@qq.com

Pinqi Guo

00082756

1397000967@qq.com

Contents

I Introduction **4**

1.1 Background 4

1.2 Justification of the research 4

1.3 Objective 4

1.4 The boundaries and scope of the project 4

1.5 Assignment (Aim) 5

1.6 Research questions 5

1.7 Report function and structure 5

II Theoretical framework **6**

2.1 Accuracy increasing methods of GPS 6

2.2 Auxiliary location system 6

2.3 Errors classification and analyzation 7

2.4 Indoor usage and testing 8

2.5 Reduce the parasitic effects 9

III Operationalization **10**

3.1 Methods 10

3.2 Material list 12

3.3 Project Plan 13

IV System design **15**

4.1 Working flow-chart of final prototype 15

4.2 Idea of the location system 15

4.3 Idea of the control system 17

4.4 Hardware introduction and analyzation 18

4.4.1 C94-M8P GPS system 18

4.4.2 Link Track S indoor beacons system 22

4.4.3 HC-020K rotary encoder 26

4.4.4 Adafruit 932 OLED screen 28

4.5 Code introduction and analyzation 31

4.5.1 Main function flow chart and analyzation 31

4.5.2 Explanations of partial variables, sub-function, algorithms 32

V Conclusion and recommendation **36**

5.1 Conclusion for research questions 36

5.2 Recommendation for future development 37

VI Reference **39**

VII Appendix **40**

7.1 Appendix I Test about C94-M8P GPS system 40

7.1.1 Test of dynamic accuracy and error of GPS 40

7.1.2 Test of static accuracy and error of GPS 43

7.2 Appendix II Test about LTS indoor beacon system 46

7.2.1 Indoor accuracy and error of LTS 46
7.2.2 Outdoor accuracy and error of LTS 49
7.2.3 Relationship of data deviation and distance between LTS beacons and obstacles 52
7.3 Appendix III Test about HC-020K rotary encoder 55
7.3.1 Test of accuracy and error of rotary encoder in short distance 55
7.3.2 Test of accuracy and error of rotary encoder in long distance 57
7.4 Appendix IV Final test 59
7.5 Appendix V Specification of final prototype 62
7.6 Appendix VI Transmission Protocol of GPS and LTS 64
7.7 Appendix VII Arduino codes of tests 66
7.7.1 Code for test 7.1.1 and 7.1.2 66
7.7.2 Code for test 7.2.2 70
7.7.3 Code for test 7.3.1 and 7.3.2 74

I Introduction

1.1 Background

Because of abundance of wind energy at sea and with aim of reduce carbon emission, more and more offshore wind turbines are going to be used which means more maintenance might be needed. But human labor and various force majeure such as weather can impact on maintenance efficiency. And more and more people are reluctant to take on this risky job. Meanwhile, 'Heavy machinery and maintenance personnel costs accounted for a larger proportion. The costs of human and stuff transporting are high.' So companies need an effective way which crawler robots are assigned to solve these problems instead of human activities. With these significant requirements, a more advanced navigation and position system must be designed based on the present prototype

1.2 Justification of the research

The present navigation system prototype has already had the function of recording the distance and direction at which the drone moves and display current location. But there are several disadvantages of this prototype:

(1) The accuracy is low especially the locations displayed by GPS are approximate
(2) Errors are difficult to eliminate which affect the measuring results
(3) Indoor usage and testing are difficult.

When it comes to the usage of this system on a maintenance robot in reality, the function of collecting blades erosion position data and robot locating all base on the system. So, the problem of accuracy and errors of navigation system must be solved.

1.3 Objective

The aim to do this project is to improve the abilities of the crawl location system and evaluate it results.

1.4 The boundaries and scope of the project

This project is only about how to improve the navigation system and no needs to pay attention to how to move the prototype or stick it on a surface.

1.5 Assignment (Aim)

(1) Improve the system with a site-accuracy of 5cm.
(2) Required to have a new simply prototype to place the GPS system, and a new error analysis method should be built for the complete navigation system.
(3) The parasitic effects must be considered and reduce them.
(4) Improve the system for indoor/outdoor usage

1.6 Research questions

Main problems can be divided into two categories:

(1) What can be used to improve the accuracy of the navigation and location system?
(2) What is the best method to improve the system for indoor usage and testing?

According to main problem sub problems contains:

(1) What devices are relative to location system?
(2) How many kinds of errors are there during location and what are corresponding solutions?
(3) What is parasitic effect and what can it affect?
(4) What cause the low efficiency of indoor usage and testing?
(5) What can be used to increase indoor signal strength?

1.7 Report function and structure

1. FUNCTION:
   (1) To introduce AIRTuB project and team direction and aim in this project
   (2) To show the design of final location system from hardware design and code design
   (3) To show the test experiment and conclusions which contribute to this project.

2. STRUCTURE:
   (1) Introduction
   (2) Theoretical framework
   (3) Method, material and plan
   (4) System design
   (5) Test, data and result(In the Appendix part)
   (6) Conclusion and recommendation
   (7) Reference
   (8) Appendix

II Theoretical framework

2.1 Accuracy increasing methods of GPS

In an RTK GPS system, real time carrier measurement is used to calculate the error correction value.

According to the research, the simplified carrier phase observation equation is the following:

$$\phi = \rho - I + Tr + c * (b\_Rx - b\_Sat) + [N * \lambda + \varepsilon\_\phi]$$

I is the signal path delay due to the ionosphere.
Tr is the signal path delay due to the troposphere.
b_Rx is the receiver clock offset from the reference (GPS) time.
b_Sat is the satellite clock offset from the reference (GPS) time.
c is the vacuum speed of light.
λ is the carrier nominal wavelength.
N is the ambiguity of the carrier-phase (integer number).
ε_ϕ are the measurement noise components, including multipath and other effects.
ρ is the geometrical range between the satellite and the receiver, computed as a function of the satellite (xSat, ySat, zSat) and receiver.

2.2 Auxiliary location system

To increase the accuracy, it could be useful to add other location system as assistance. For instance, 'Dead Reckoning is a kind of vehicle independent positioning technology, the basic principle of which is to calculate the instantaneous position of the vehicle by using the direction and distance sensor. This method has higher positioning accuracy in a short time' from a new type positioning instruction. Another equipment could be combined with present GPS system to apply more date details to the center control system for location. In recent years, the CP concept is the new methodology that can exploit wireless communication technology to improve vehicular positioning accuracy. 'We propose the system architecture called cooperative positioning via dead reckoning (CPDR) to improve GPS position error in VANET' However, there will be some data analysis which is disposed by software such as MATLAB and so on. There is a difficulty for operator to transfer and consolidate all data together. In general, it is a changeling to achieve the goal that is simplifying the input message and output a handled data array in a center data redaction.

2.3 Errors classification and analyzation

Error is a main element which can affect locating results significantly. And errors source can be divided into two categories with sub classifications and examples:

(1) Unsolvable error sources:
  a) Satellite clock errors & Electrical jamming
  b) Environment errors:
    i. Ionospheric errors
    ii. Obstacles impedance and reflection
(2) Solvable error sources:
  a) Receiver errors:
    i. Receiver clock errors.
  b) Circus structure:
    i. Parasitic effect
  c) Calculation method errors:
    i. Concave and convex cause placement different from actual distance.[3]

For the first category errors is beyond team's ability which is called unsolvable error sources. The satellite clock deviation is an unavoidable phenomenon but only can solved by satellite owner. Electrical jamming from governments to protect military secrets are also impossible to be solved. And the propagation through the atmosphere introduces delays into the propagation time of navigation signals, this leads to atmospheric deviations (errors). Especially the impendence and reflection of signal caused by high building also affect the transmission time.

In terms of solvable error sources, because the receiver clock may not be precisely synchronized with GPS time leads to receiver errors. And some designs of the circus can product undesirable effects such as parasitic effect and have interferences on signal to cause errors.

From the design and test report of the present prototype, the accuracy of this system is 13cm. And its calculation method is comparing the measured actual distance between two points with calculated distance according to data from GPS and the product of forward velocity and time recorded by drone respectively. But there is a visible issue if the surface where drone is roving is not a plane, convex or concave will have effects on the distance drone has actually gone which means displacement is not equal to distance. As a result, in order to improve the accuracy, the calculation method must be refined and the gradient of real routes should be taken in account.

2.4 Indoor usage and testing

In order to realize the indoor usage and testing, the signal strength should be increase. To increase signal strength, GPS signal repeater can be used. Repeater receives GPS signal from outdoor areas, magnifies and retransmits new signal to indoor areas.

Figure 2.4.1 GPS signal repeater

UWB is a carrier-free communication technology. It has the advantages of high temporal resolution, strong penetration, low power consumption, good anti-multipath effect and high security. UWB is an electromagnetic wave that travels at the same speed as the speed of light in a vacuum. By measuring the TOF (flight time) from TAG to ANCHOR, and multiplying with the speed of light, TAG can obtain the distance to ANCHOR. According to the data of the distances between ANCHOR and TAG, reference coordinates of ANCHOR, multiple groups of spherical equations can be listed, and then the coordinates of the tag can be solved by mathematical method.

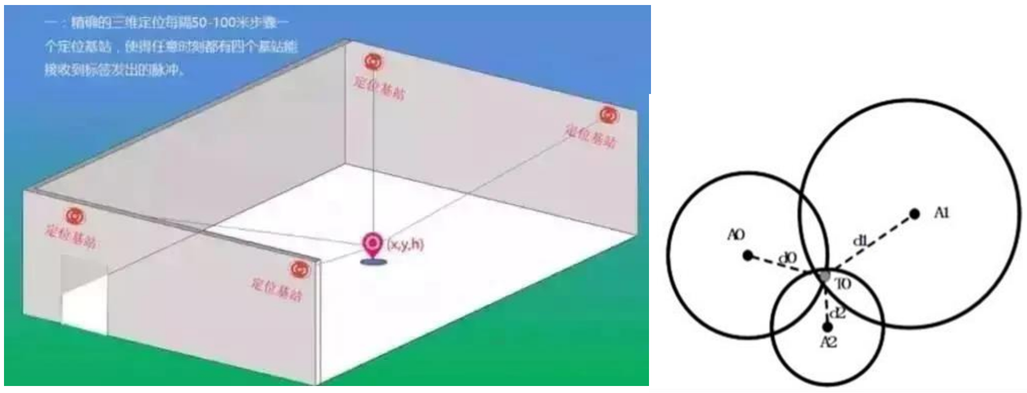

一：精确的三维定位每隔50-100米步叠一个定位基站，使得任意时刻都有四个基站能接收到标签发出的脉冲。
定位基站
定位基站
定位基站
定位基站
(x,y,h)
A0
A1
A2
d0
d1
d2
T0

Figure 2.4.2 Working principle of Indoor system

On others hands, combine an indoor location system to GPS system is equal to increase signal strength which can even reach a higher accuracy. Beacons is one of useful indoor location systems. Beacons system contains four anchors and one tag. Tag transmits signal to anchors and records time of signal transmission. Using the velocity of signal, the system can calculate distance between tag and anchors which can realize indoor location.

2.5 Reduce the parasitic effects

Except for common measuring and observational error, parasitic effect in integrated circuit can give rise to big errors. Due to the wiring structure generated by the mutual capacitance and inductance phenomenon, appear in place without originally designed capacitors and inductors which called parasitic capacitors and parasitic inductors. And in the case of high frequency signal input, this part of circuit reveals the characteristic of capacitors and inductors. This phenomenon is called parasitic effect

In the case of alternating current, especially at high frequency, the influence of Parasitic effect is great. According to the complex impedance formula, parasitic capacitance and inductance will block the movement of current greatly in AC condition.

One example of parasitic circuit:

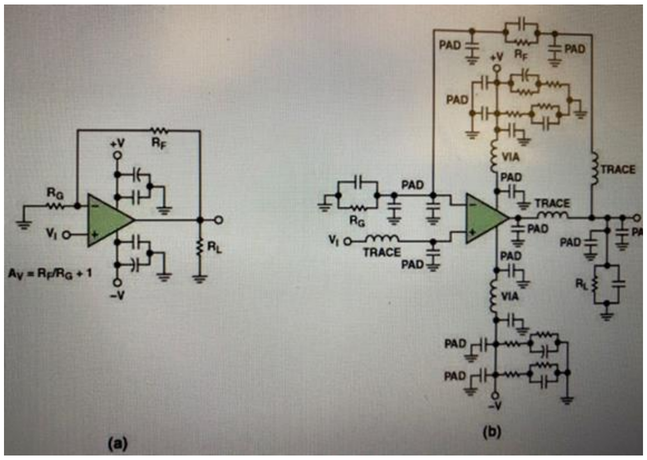


Figure 2.5.1 (a) Ideal operational amplifier (b) Having parasitic effects

There are several ways to reduce parasitic effect:

(1) Circuit board wire should be as short as possible, paying attention to its length and width to minimize parasitic effects.
(2) The distribution of the power cord should be as wide as possible.
(3) Using twisted-pair cable to reduce the parasitic effects.
(4) Using SMT (Surface-Mount Technology) components with short pins.

III Operationalization

3.1 Methods

In the whole project, we proceed it with **'Delft Design Methods'**

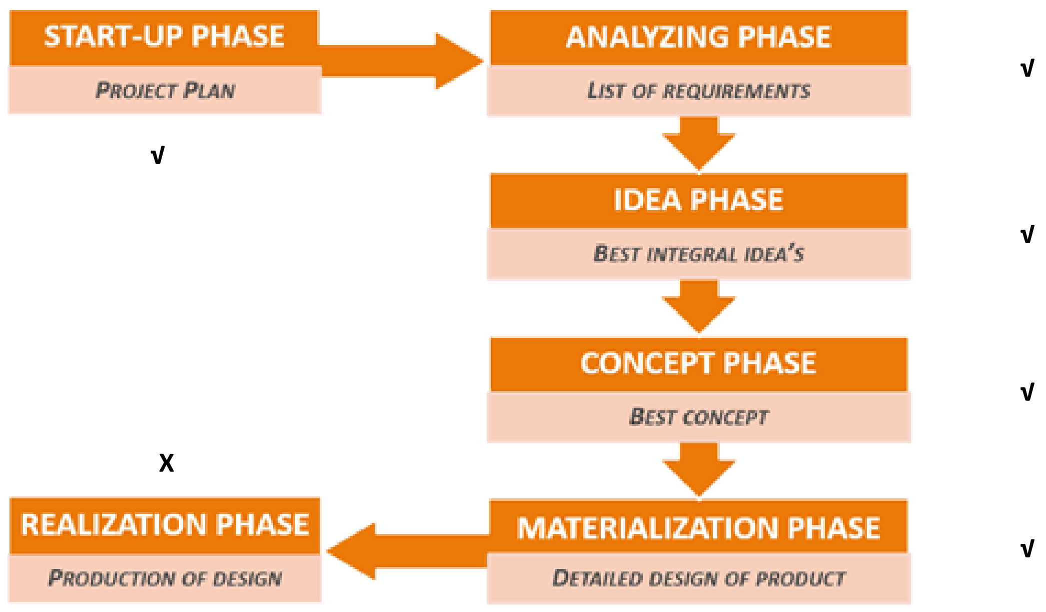


Figure 3.1.1 Delft design methods

Because the deliverables of this project are merely a prototype of our concepts rather be a real product, so the realization phase will not be covered. Materialization phase is the end of the project.

1. START-UP PHASE

In this phase, with start of the project, members need to be clear about their competence scope, aims, and ambitions of each other which is important for future direction decision. What's more, everyone exchanges opinions about what they think about the project and make an initial plan at final.

Deliverables:

(1) A report of everyone's ability, aim and ambitions.
(2) An options list of the project.
(3) An initial plan.

2. Analyzing phase

In this phase, members need to analyze clients' assignments and read report from last year to realize how the prototype works.

Deliverables:

(1) A report of assignments analyzation.

(2) Summaries after reading previous report.

3. Idea phase

In this phase, based on analyzations in analyzing phase, team starts to collect information and brainstorm to motivate member's creativity about the function of new prototype according to every assignment. An ideal model can be contributed to meet all requirement even better than clients' expectation. In order to refine this phase, members need to ignore their present knowledge storage which might limit vision. Widely collecting information of navigation and location system and thinking, trying to use the latest technology to improve the ideal model. Finally, a project proposal needs to be made which contains an initial material list.

Deliverables:

(1) An initial material list

(2) Test result reports of different parts which contain deviation analyzation

(3) A list of what can be improved by changing program code or using new components

(4) A complete project proposal

4. Concept phase

In this phase, members need to research about the present prototype and do experiment again to improve and correct ideals. Based on ideal phase, combining members' competence scope and real situation which contains real environment and budgets to narrow thinking and turn perfect ideas into concepts which can be realized. Members are going to make materials lists.

Deliverables:

(1) An integrated material list

(2) A simulated model

5. Materialization phase

Based on concept phase, real model needed to be designed according to concepts and material list. After designing and purchasing materials, testing different components respectively to check if every part can work well and conform to concepts. And then, assembling every component as an entirety, programming for the controller model and design experiments which can verify its functions. Doing outdoor and indoor experiments to check if the prototype can work well as an entire. Finally, completing the final report.

Deliverables

(1) Report for testing of components.
(2) An experiment list.
(3) An assembled prototype
(4) Report for testing of an entire prototype.
(5) Final report for the whole project.

3.2 Material list

Table 3.2.1 Material list

General-purpose device	
Jumper wires	50
Bread board	1
Adafruit 932 OLED screen	1
4pin button	5
Controller	
Arduino Mega 2560	1
Navigation and location	
C94-M8P RTK GPS system	1
LinkTrackS indoor beacons system	1
HC-020K rotary encoder	2
Power supply	
Power bank for GPS module and controller	5

3.3 Project Plan

Table 3.3.1 project plan

Phases	Activities	Deliverables	Time
START-UP PHASE	1. Search information of background 2. Identify aims and moti-vation 3. Make an initial plan	1. A report of everyone's ability, aim and ambitions. 2. An options list of the project. 3. An initial plan.	5 workdays
ANALYZATION PHASE	1. Analyze assignments 2. Read the previous report and Identify the main problem and subpro-blem	1. A report of assignments analyzation. 2. Summaries after reading previous report.	5 workdays
IDEA PHASE	1. According to analyzation of assignments, discuss the desire function of new prototype. 2. Make an initial list of what materials or components are needed 3. Test the present compass model 4. Test the present GPS model and rotary encoder 5. Test beacons for indoor usage 6. Analyze test results, if result is different from desire function result, try	1. An initial material list 2. Test result reports of different parts which contain deviation analyzation 3. A list of what can be improved by changing program code or using new components 4. A complete project proposal	27 workdays

	to find out where deviations from 7. According to the test result, find out if the present component can meet requirements and check if any components need to be replaced. 8. Analyze programming code of control model. Make sense of what the meaning of every code. 9. Fresh project proposal		
CONCEPT PHASE	1. According to the test and learning results to refresh the material list 2. Design a simulated model according to the final material list.	1. An integrated material list 2. A simulated model	6 workdays
MATERIALIZATION PHASE	1. Purchase materials 2. Test every component 3. Assemble components 4. Program controller 5. Design and perform experiments 6. Complete final report	1. Report for testing of components. 2. An experiment list. 3. An assembled prototype 4. Report for testing of an entire prototype. 5. Final report for the whole project.	52 workdays

IV System design

4.1 Working flow-chart of final prototype

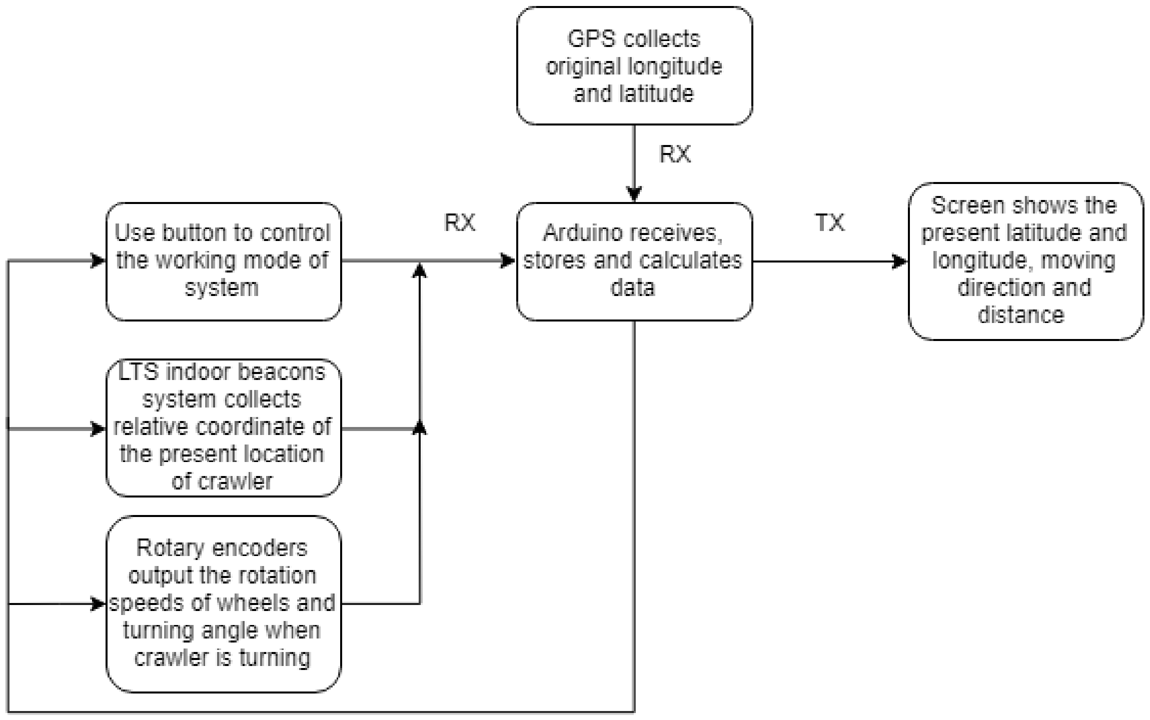

GPS collects original longitude and latitude
RX
Use button to control the working mode of system
RX
Arduino receives, stores and calculates data
TX
Screen shows the present latitude and longitude, moving direction and distance
LTS indoor beacons system collects relative coordinate of the present location of crawler
Rotary encoders output the rotation speeds of wheels and turning angle when crawler is turning

Figure 4.1.1 Working flow chart of final prototype

4.2 Idea of the location system

The new prototype bases on the prototype from last group which used GPS module as the main location component with compass and rotary contained in the system. After analyzation of the old prototype and search of GPS relative information, the results are shown in the follow list.

Table 1 Analyzation result

Components	1) GPS module 2) Rotary Encoder 3) Compass module

Accuracy	15cm
Main problem	1) The location accuracy depends on the strength of GPS signal and also number of obstacles. 2) GPS module can't work very well indoor. 3) GPS module can be affected by weather.
Conclusion	The error of location system mainly comes for the unstable working character caused by signal interference.

The analyzation results show that merely GPS can not reach the aim of 5cm accuracy of the location system. In order to improve the accuracy, team decided to find new way of location which can be less affected by environment elements and also can be used indoor. and use new location system as auxiliary system of the GPS system. Combining different system to decrease errors and improve accuracy.

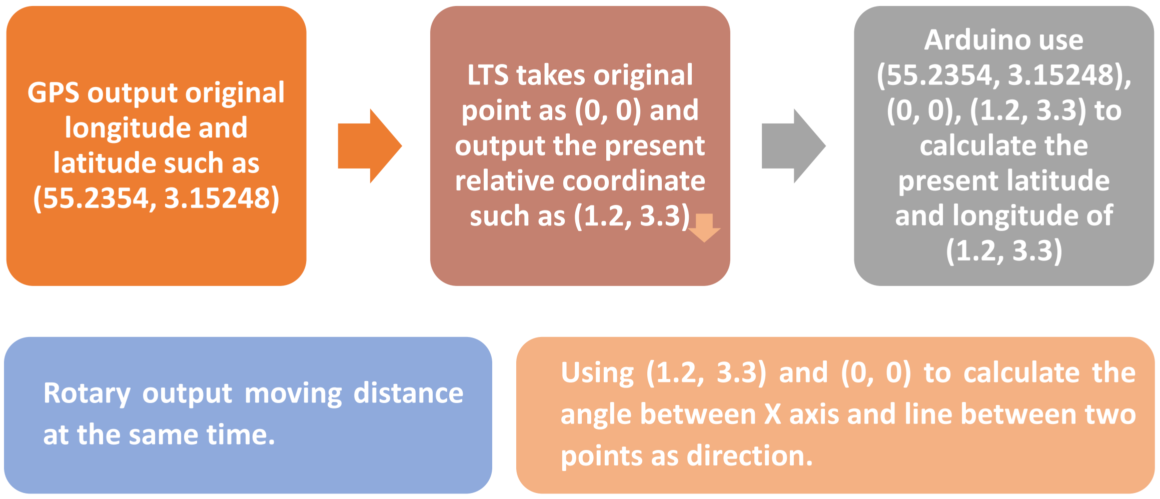
Finally, following components are used in new prototype. In the final design, team decides to replace mainly used GPS with LTS(LinkTrackS) indoor beacons system which can be less influenced by weather and some common signals such as WIFI and Bluetooth and also has a high accuracy of 5cm. Although the GPS module is still needed to provide an original latitude and longitude, the errors of system are more controllable because less usage of GPS.

Table 4.2.2 Component list of final prototype

Component	**Accuracy and error**	**Function**
C94-M8P GPS module	1cm, 5-9cm	Providing a static original latitude and longitude as a reference point of follow-up location.
LinkTrackS Indoor Beacons System	5cm, 0-7cm	1) Building its own coordinate system and output a relative coordinate. Taking the original point from GPS as (0, 0). Getting the present latitude and longitude by calculation according to the relative coordinates and original latitude and longitude. 2) Using different coordinates at different time to calculate the moving direction.

HC020K Rotary Encoder	1cm, 0-3cm	1) Providing moving distance at one moving direction. 2) Proving angle of turn when crawler is turning.

Figure 4.2.1 Location principle flow chart


GPS output original longitude and latitude such as (55.2354, 3.15248)
LTS takes original point as (0, 0) and output the present relative coordinate such as (1.2, 3.3)
Arduino use (55.2354, 3.15248), (0, 0), (1.2, 3.3) to calculate the present latitude and longitude of (1.2, 3.3)
Rotary output moving distance at the same time.
Using (1.2, 3.3) and (0, 0) to calculate the angle between X axis and line between two points as direction.

4.3 Idea of the control system

In the final design, team assumes that the movement of crawler bases on the moving control signal from a remote-control unit. The process will change working mode or calculation method depending on the moving control signal. And according to the assumption, the remote-control unit should output signal about the moving direction to tell the crawler what it should do next step. So, five buttons are used to simulate the moving remote-control signal. Every movement of crawler should according to moving control signal.

For processing data and control signal, an Arduino 2560 Mega is used to receive and send orders to every component as the processer of control system.

Table 4.3.1 Control buttons list

Button	Function
Button 1	Send go ahead order
Button 2	Send go back order
Button 3	Send turn left order

Button 4	Send turn right order
Button 5	Refresh original location and distance record.

4.4 Hardware introduction and analyzation

4.4.1 C94-M8P GPS system

1. Introduction

C94-M8P GPS system is a RTK-GPS system which contains two parts: **base station** and **rover**. Compared with normal GPS, C94-M8P uses UHF radio link technology building connection between base station and rover to which allows the Base Station to send the correction data stream to the Rover so that the Rover can output its relative position with cm-level accuracy in good environments.

Rover and base station are two same application board with same hardware structure but working in different mode. Base station is used to collect an accurate static coordinate and using UHF radio wave to locate the position of rover and transmit position data to rover. Rover can output its position according to the position of base station and distances between them. Usually, base station is fixed and rover moving with crawler.

2. Hardware structure and function

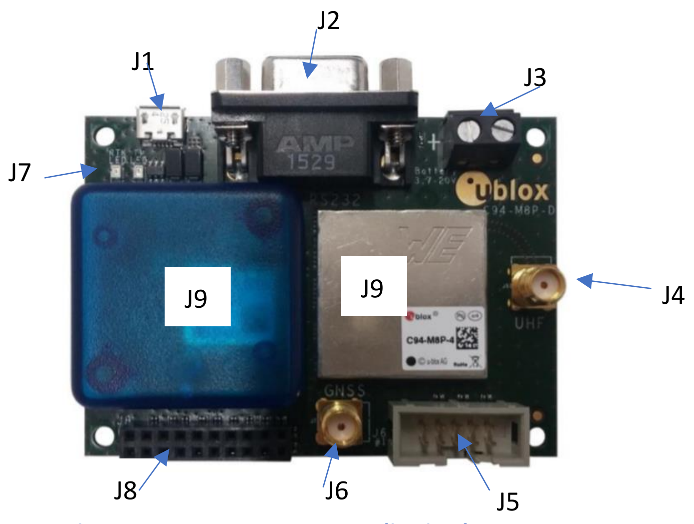
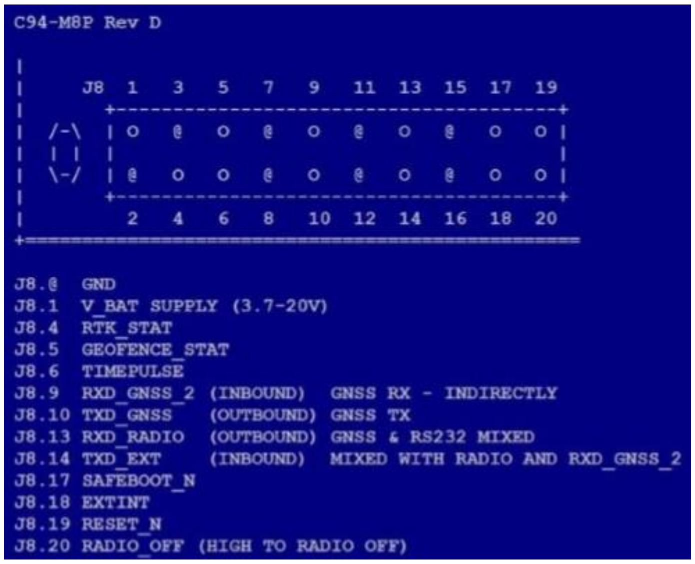


Figure 4.4.1.1 **C94-M9P GPS application board**

Table 4.4.1.1 Functions of units on GPS application board

Order	Function
J1	USB port for data transmission and power
J2	RS232 port – UART M8P/Radio
J3	External battery / DC connector
J4	UHF radio link antenna connector
J5	Debugger interface for radio module
J6	GNSS antenna connector
J7	Indicator LEDs
J8	UART debug and data transmission port
J9	Radio and GPS chips



```
C94-M8P Rev D

|
|     J8  1   3   5   7   9   11  13  15  17  19
|        +---------------------------------------+
|  /-\   | o   @   o   @   o   @   o   @   o   o |
|  | |   |                                       |
|  \-/   | @   o   o   @   o   @   o   @   o   o |
|        +---------------------------------------+
|         2   4   6   8   10  12  14  16  18  20
+==================================================

J8.@   GND
J8.1   V_BAT SUPPLY (3.7-20V)
J8.4   RTK_STAT
J8.5   GEOFENCE_STAT
J8.6   TIMEPULSE
J8.9   RXD_GNSS_2 (INBOUND)  GNSS RX - INDIRECTLY
J8.10 TXD_GNSS    (OUTBOUND) GNSS TX
J8.13 RXD_RADIO   (OUTBOUND) GNSS & RS232 MIXED
J8.14 TXD_EXT     (INBOUND)  MIXED WITH RADIO AND RXD_GNSS_2
J8.17 SAFEBOOT_N
J8.18 EXTINT
J8.19 RESET_N
J8.20 RADIO_OFF (HIGH TO RADIO OFF)
```

Figure 4.4.2 Functions of units on GPS application board

3. Input and output flow chart

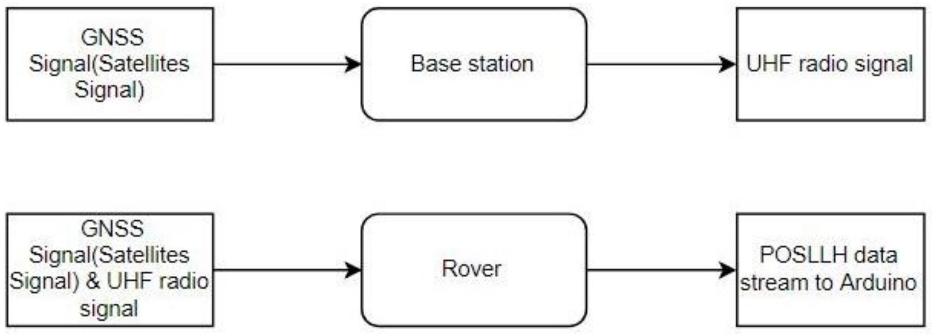

GNSS Signal(Satellites Signal)
Base station
UHF radio signal
GNSS Signal(Satellites Signal) & UHF radio signal
Rover
POSLLH data stream to Arduino

Figure 4.4.1.3 Input and output flow chart of GPS system

4. Practical usage and matters needing attention

(1) Power supply: 5V DC from DC connecter or USB port of application board.

(2) Data transmission:

a) Using UART debug and data transmission port to transmit data to Arduino and connecting PIN9, PIN10 to PIN10, PIN11 on Arduino MEGA 2560 (RX of GPS corresponds to TX on Arduino, TX of GPS corresponds to RX on Arduino).

b) UHF radio wave is used to transmit data between base station and rover. Max transmission is around 238m.

(3) Set up and date monitoring: Corresponding software u-center can be used to set up the working mode, function and accuracy of GPS before using. Location data and calibration state can also be monitored on u-center.

(4) Calibration: There are two kinds of calibration mode. Automatic calibration and fixed position calibration. Every time before using, GPS system needs to be calibrated. Fixed position calibration needs users to give base station position data if the position of it has been know. Otherwise automatic calibration needs to be used. It may take a long time to finish. In order to decrease calibration time, there are several points need to be noted:

a) Calibration should be completed in an open area which means an outdoor area without large area obstacles around.

b) Calibration is best performed in areas without WIFI signal interference.

c) Lower accuracy and less observation time of GPS which are set in u-center means less calibration time but more errors. When observation time and accuracy both conform to set parameters. The calibration is completed. For example, if observation time is set as 500s and accuracy is set as 1m, when accuracy of GPS maintains less than 1m for more than 500s, the calibration is completed.

(5) **Working modes:** C94-M8P GPS system has two working modes, static RTK GPS mode and moving baseline RTK GPS mode. The first one is fit for situations which have static basement and rover. The second one is fit for situations which have moving basement or rover with higher accuracy but can be easily interfered. In this project, the static RTK GPS mode is used. In final design, GPS only works when it is used to get original longitude and latitude because its low accuracy when moving. But rover is still set on the crawler for getting new original longitude and latitude when place of use is changed.

(6) **Indicator LEDs:**

a) Blue LED is blinking when there is a GNSS position fix (TIMEPULSE). If there is no GNSS fix, the LED will light up without flashing

b) Green LED indicates the RTK status. LED flashes in float mode and stays on in fix mode. The LED is off when there is no RTK fix.

(7) **Location factor:** The position of GPS module is decided by the position of GNSS antenna rather than chipset.

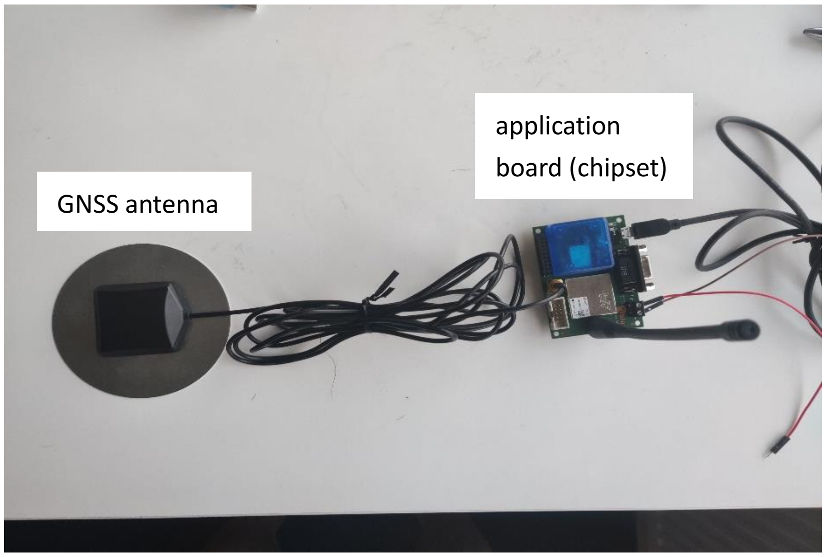


Figure4.4.1.4 GPS application board with antenna

4.4.2 Link Track S indoor beacons system

1. Introduction

LTS (Link Track S) indoor beacons system is an independent location system without using external equipment to locate. LTS has five parts, one **tag** and four **anchors**. Tags and anchors are all beacons with same hardware structure but different working modes. Tag moves with crawler and anchors are fixed. Anchors transmit date to tags by using UWB signal and tag calculates distances between itself and anchors according to data from anchors.

In the LTS system, anchors create its own coordinate system. Usually the position of A0 (anchor 0) is (0, 0). Tag can output its own coordinates according to (0, 0) and distances between itself and anchors. More anchors mean higher accuracy but At least four anchors should be used in the system to make sure cm-level accuracy. This system can only be used for small range locating which means distances between anchors and tags should be less than 80m no matter used indoor or outdoor.

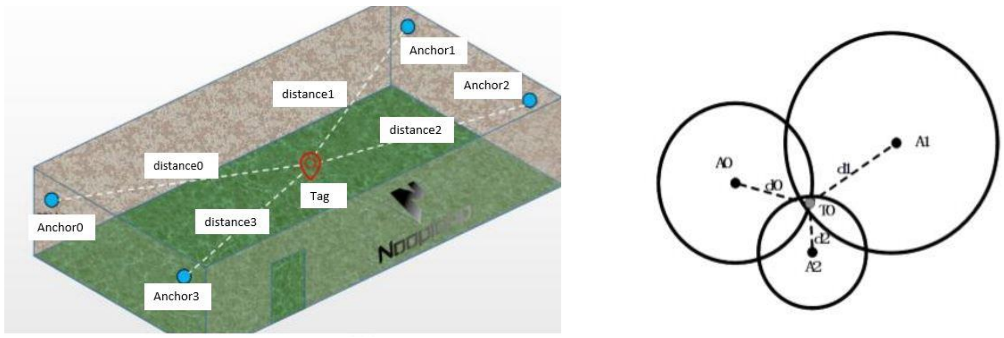


Figure 4.4.2.1 Working principle of LTS system

2. Hardware structure and function

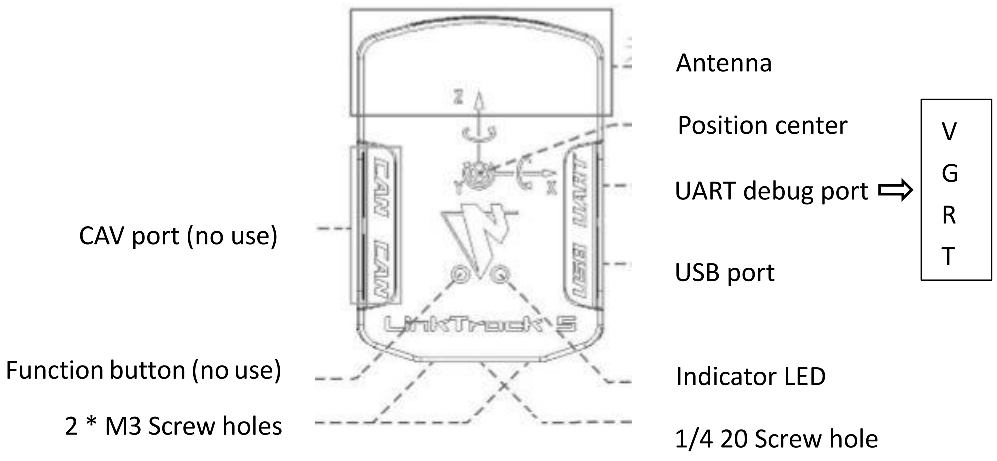


Figure 4.2.2.2 Link Track S

3. Input and output flow chart

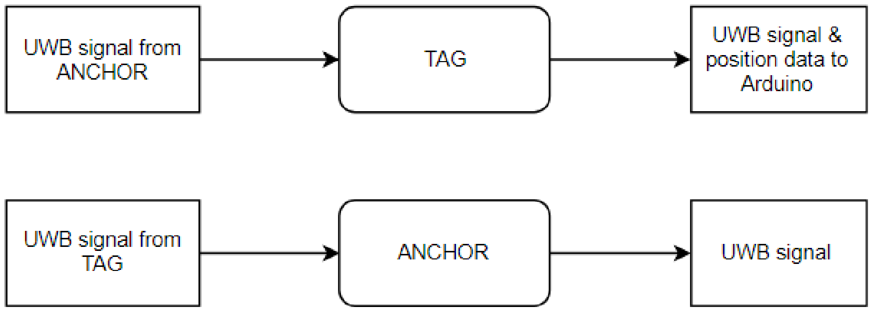


Figure 4.2.2.3 Input and output flow chart of LTS system

4. Practical usage and matters needing attention

(1) **Power supply:** 3.3V – 5V DC from UART VCC port or USB port of beacon.

(2) **Data transmission:**

a) Use debug port to transmit data to Arduino. The pin order of UART port is V(VCC), G(GND), R(RX), T(TX), which are corresponding to PIN5V, PINGND, PIN13, PIN12 on Arduino

b) UWB signal is used to transmit data between anchors and tag. The max transmission distance between beacons are around 80m.

(3) **Set up and date monitoring:** Corresponding software NAssistant can be used to set up roles, working modes, data transmission frequency of LTS beacons before using. Location data, moving positions tracks and calibration state can also be monitored on NAssistant.

(4) **Calibration:** There are two kinds of calibration modes, one key calibration and fixed position calibration mode. Fixed position calibration mode can only be used when distances between anchors have been known. One key calibration mode, anchors build connection and measure distances between each other and finally output position data of them. Normally, if the positions anchors don't change, the calibration is only needed once before the first time using. And calibration should be realized in NAssistant after beacons are placed. In order to decrease calibration time and improve accuracy, there are several points need to be noted before calibration.

a) The direction of beacons (Blue one is anchor and red one is tag, black block is the position of antenna, stars mean signal strength):

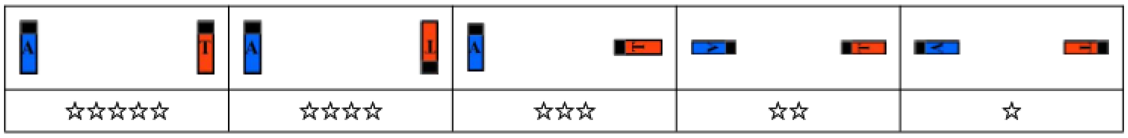


Figure 4.2.2.4 Signal strength between one anchor and tag

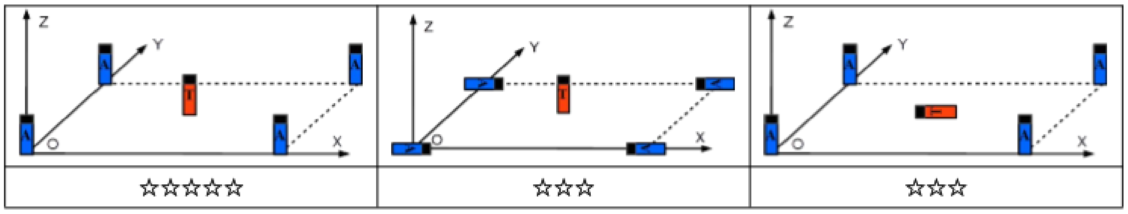


Figure 4.2.2.5 Signal strength when anchors and tag are in a same plane with different directions

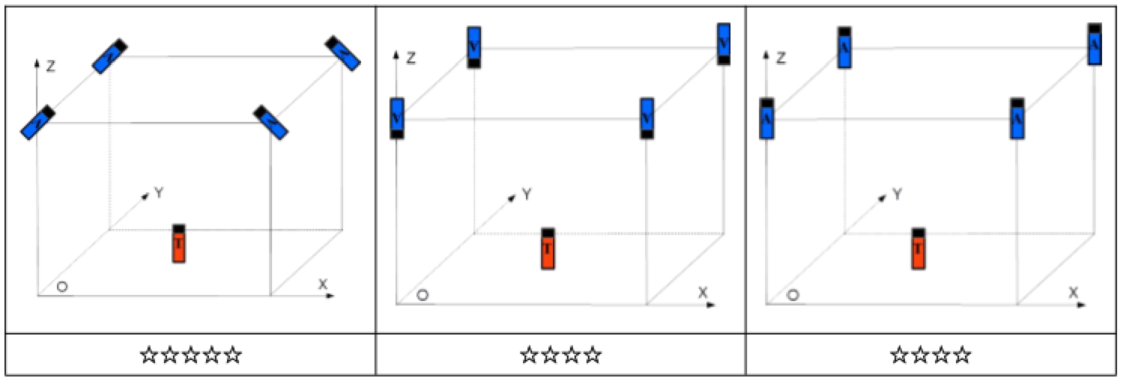


Figure 4.2.2.6 Signal strength when anchors and tag are in different planes with different directions

b) Before calibration, no obstacles between every anchor should be guaranteed.

(5) Working modes: LTS system has three working mode, LP (Local Positioning) mode, DR (Distributed Ranging) mode, DT (Data Transmitting) mode. In this project LTS working in LP mode. In LP mode, there are several working modes can be used for anchor and tag which are shown in the following list:

Table 4.4.2.1 Working modes of LTS system

Mode name	Function
Anchor	
ANCHOR_FRAME0	Output distance between tags and anchors
NODE_FRAME0	Output detail data information which is sent to tags

Mode name	**Function**
Tag	
TAG_FRAME0	Output position and posture data
NODE_FRAME2	Output position, posture and anchors information data
NODE_FRAME3	Output anchor information data

In this project, tag is used to transmit data to Arduino, so only working mode of tag is related to the output results. And TAG_FRAM0 is used because of its relative smaller data pack size compared with other modes.

(6) Indicator LED:

Table 4.4.2.2 Indicator LED states of LTS system

Blinking state	**Working state**
Green, High frequency	Beacons have been connected in the system and data is transmitting
Green, Low frequency	Beacons lose connection with other beacons
Blue and Green	Beacons are being calibrated
No light	No data transmission

(7) Location factor: During the measurement, obstacle between beacons can cause large errors, especially distances between obstacles and beacons. So it is necessary to make sure distances between obstacles and each beacons are more than 4 meters to get the best accuracy. (Test 5.2.3)

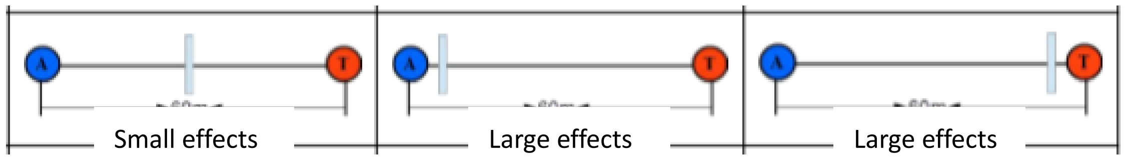


Figure 4.2.2.7 Effects of distances between obstacles and beacons

4.4.3 HC-020K rotary encoder

1. Introduction

HC-020K rotary encoder is a kind of speedometer which use photoelectrical gate to sense rotation speed of wheels. Rotary consists of chipset with photoelectrical gate and a small wheel with spokes. Wheel with spokes is fixed on the spindle of the wheel whose speed will be measured. Photoelectrical gate is put above the small wheel and let signal light pass through holes on the wheel. Spokes will prevent the light signal transmission between gate and chipset will generate signal pulses to show the wheel is rotating. Arduino calculates speeds according to number of pulses and spokes on the wheel.

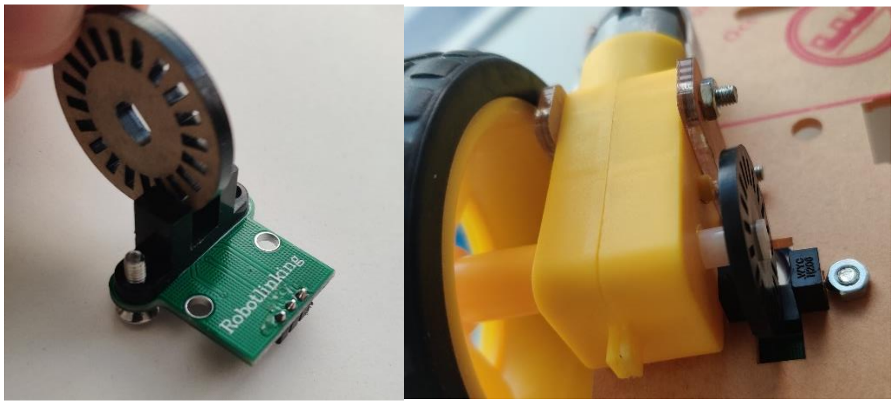


Figure 4.4.3.1 Rotary encoder 1

2. Hardware structure and function

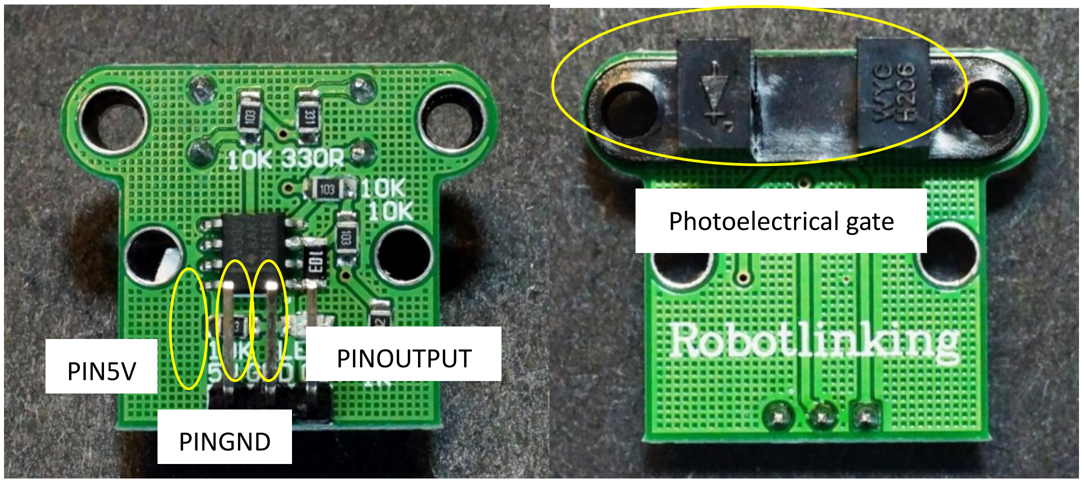


Figure 4.4.3.2 Rotary encoder 2

3. Input and output flow chart

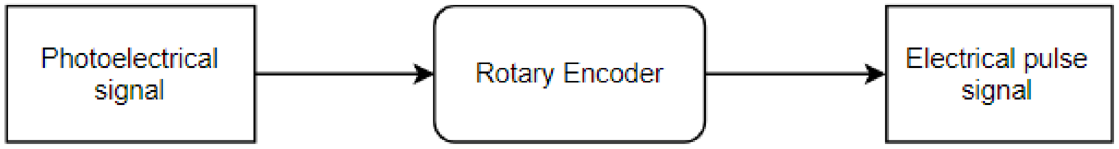


Figure 4.4.3.3 Input and output flow chart of rotary encoder

4. Practical usage and matters needing attention

(1) **Power supply:** 4.75V – 5.25V DC from PINVCC.

(2) **Data transmission:** Connecting PINOUTPUT to PIN2, PIN3 on Arduino (two rotary encoders are used), PIN5V to PIN5V, PINGND to PINGND

(3) **Calibration:** Reset is the only way to calibrate rotary encoder. Power loss and pushing reset button on Arduino can reset the rotary encoder. Also in this project, button 5 of the control system can be used to reset rotary encoder

(4) **Working modes:** Rotary encoder only has one working mode, output pulse when small wheel is rotating. So it can not distinguish the rotation direction. Control signal is needed to tell Arduino the rotation direction of wheel. Especially when using rotary encoder, velocity and baud rate control is needed to guarantee the effective data transmission and avoid packet loss.

(5) **Indicator LEDs:** When red light is blinking, it means the state of wheel is changing.

4.4.4 Adafruit 932 OLED screen

1. Introduction

Adafruit 932 OLED screen is a 128x32 pixel monochrome OLED display. In this project, this screen is used to show the location information of the crawler which contains longitude, latitude, moving direction, moving distance on this direction, turning direction, turning angle.

2. Hardware structure and function

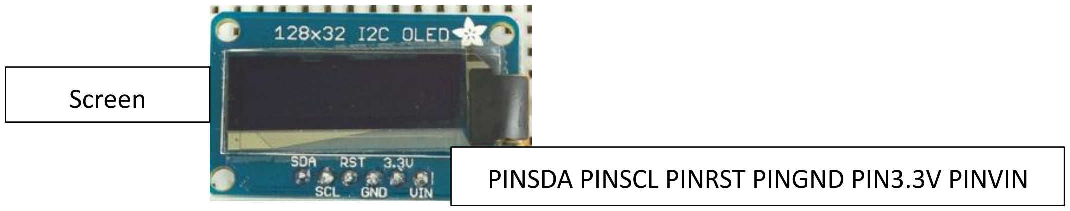


Figure 4.4.4.1 OLED screen

3. Input and output flow chart

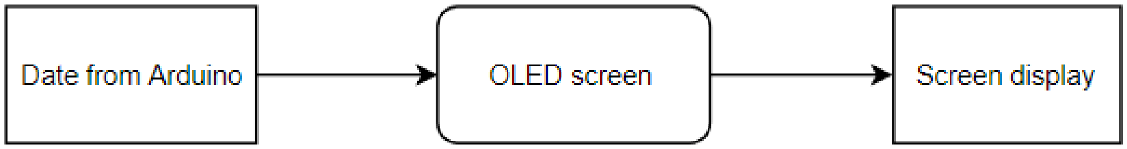


Figure 4.4.4.2 Input and output flow chart of OLED screen

4. Practical usage and matters needing attention

(1) **Power supply:** 3.3V or 5V from PIN3.3V or PINVIN

(2) **Data transmission:** Connecting PINSDA, PINSCL to PINSDA, PINSCL on Arduino, PIN3.3V or PINVIN to PIN3.3 or PIN5V, PINTGND to PINGND, PINRST to PIN4

4.4.5 Arduino MEGA 2560

1. Introduction

Arduino MEGA 2560 is the processor of location system and control system. Compared with Arduino UNO. MEGA 2560 has more pins and more interrupt port for practical usage.

2. Hardware structure and function

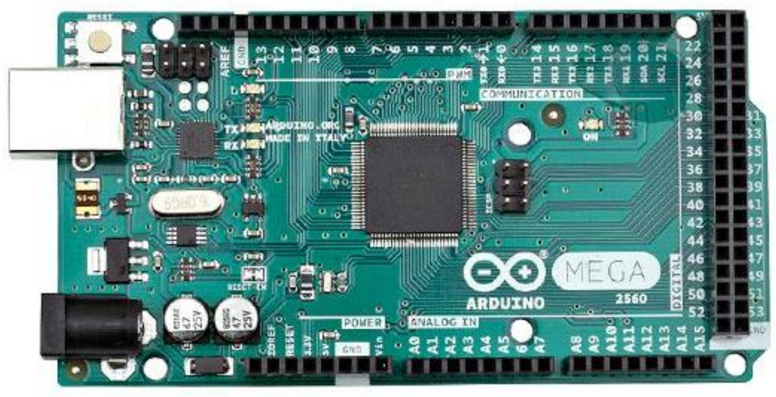


Figure 4.4.5.1 Arduino MEGA 2560

3. Practical usage and matters needing attention

(1) **Power supply:** 5V from PC or Power bank

(2) **Data transmission:**

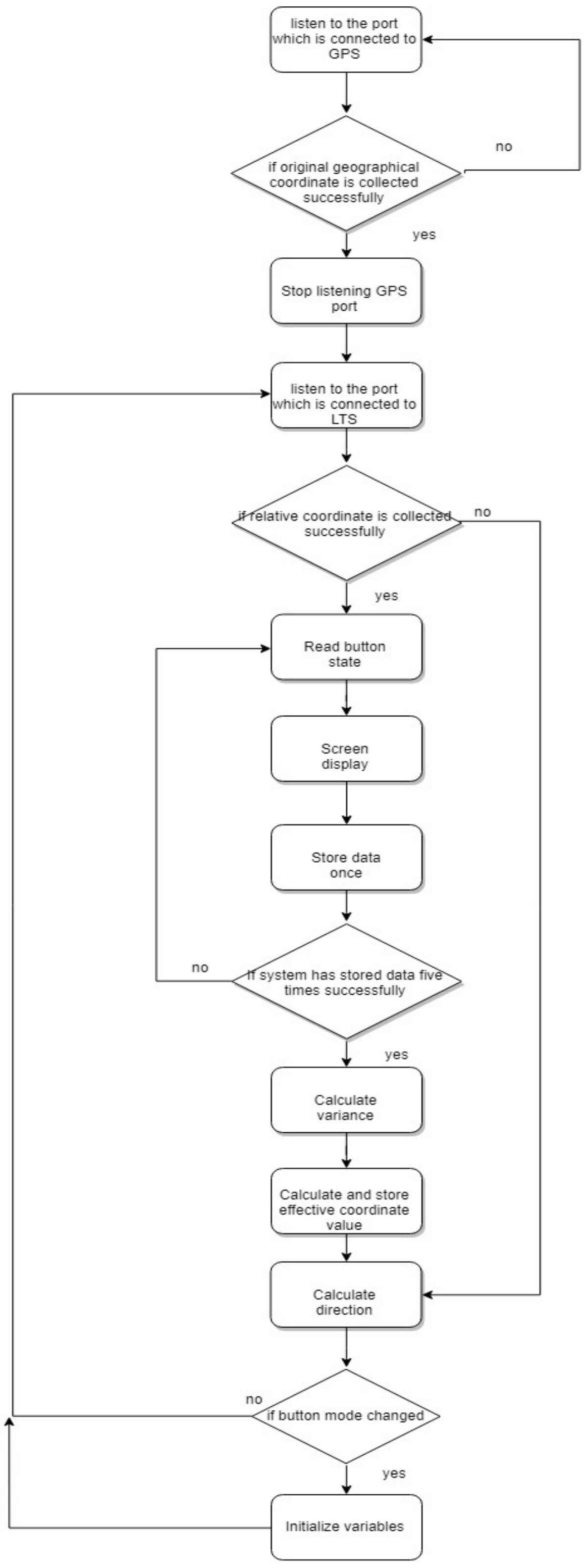
Table 4.4.5.1 Indicator LED states of LTS system

PIN Number	Function
2 (Interrupt 0)	To PINOUTPUT from rotary encoder on left wheel. Receiving pulse from it.
3 (Interrupt 1)	To PINOUTPUT from rotary encoder on right wheel. Receiving pulse from it.
4	To PINRST from OLED screen.
10	To PIN9 (RX) from GPS.
11	To PIN10 (TX) from GPS. Receiving data from it.
12	To PINTX from LTS. Receiving data from it.

13	To PINRX from LTS.
A8	To button 1.
A9	To button 2.
A10	To button 3.
A11	To button 4.
A12	To button 5.
SCL	To PINSCL from OLED screen
SDA	To PINSDA from OLED screen
5V	To PINVCC from LTS, PIN5V from rotary encoder, PINVIIN from OLED screen
GND	To PINGND from LTS, rotary encoder, OLED screen.

4.5 Code introduction and analyzation

4.5.1 Main function flow chart and analyzation


listen to the port which is connected to GPS
if original geographical coordinate is collected successfully
no
yes
Stop listening GPS port
listen to the port which is connected to LTS
if relative coordinate is collected successfully
no
yes
Read button state
Screen display
Store data once
if system has stored data five times successfully
no
yes
Calculate variance
Calculate and store effective coordinate value
Calculate direction
if button mode changed
no
yes
Initialize variables

Main function mainly contains three parts: Original GPS data collection, LTS data collection and calculation loop, mode initialization loop. In order to guarantee continuous screen display and button state monitor, these two sub-functions are set in the LTS data collection and calculation loop which is executed in every main loop. Sub-functions about calculations of every LTS related variable such as GPSlocation.curlat and GPSlocation.curlon are also contained to provide timely data update.

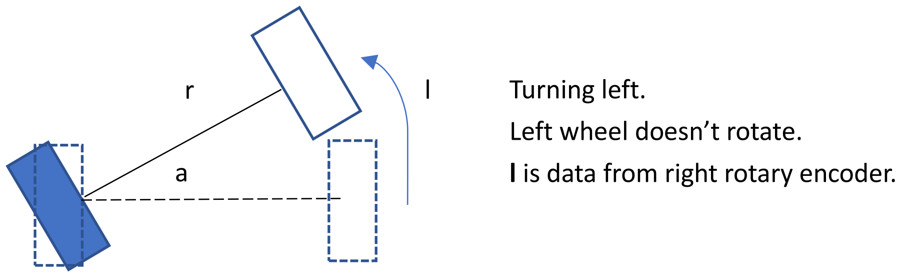
Rotary encoder related functions are not written in main loop because of the time waste of data calculation in main loop which may cause data reception delay and packet loss. So interrupt 0 and 1 of Arduino are used for left and right rotary encoder to avoid data transmission interference from main loop.

4.5.2 Explanations of partial variables, sub-function, algorithms

1. Variable and sub-function

a) **RADTODEG** is a coefficient used to convert radian into angle. This variable will be used in angle_cal() sub-function to calculate turning angle.

b) **angle_cal()** is used to calculate turning angle when crawler is turning. Formula l = a * r is used. In the final design, when crawler is turning, the wheel which corresponds to the direction of turn is set to be static (This wheel is taken as the center of rotation). So the turning radios **r** is distance between two wheels and turning distance **l** is the moving distance of another wheel.


r
l
a
Turning left.
Left wheel doesn't rotate.
l is data from right rotary encoder.

Because **a** is a radian number, so RADTODEG needed to be used to show the final result

c) **LSB_M_TO_LAT_LONG** is a coefficient used to calculate the present latitude and longitude according the original latitude and longitude and relative coordinates of original point and the present point.

d) **serialxx.listen(), serialxx.isListening(), serialxx.end()** are function which is related to the usage of Arduino soft serial. Arduino can only collect data from one port, so it is necessary to tell Arduino when to collect data from which port. Using serialxx.listen() to start collection and using serialxx.end() to stop collection. serialxx.isListening() is used to check port state. Not every port can be used as soft serial for data transmission and reception.

e) **Serial.print(), Serial.println()** is related to data monitor on Arduino IDE.

f) **pinMode(A8, INPUT_PULLUP)** is used to set port state. INPUT_PULLUP means data input mode and using pull-up resistor. A8 is connected with button. Using pull-up resistor means no more extra external resistor is needed for button. And in this state, the normal state of port is high-level. After pushing the button, the state of port will turn into low-level transiently.

g) **timesforvariance** is used count data collection time. When this variable is equal to five, Arduino will calculate the variance of five groups data. This variable is related to variable **calibration.** The final value is (timesforvariance – calibration).

h) **calibration** is a variable used to check if a group data from LTS is valid. The value of this variable is returned from coordinate_cal(). During the data transmission, there are always signal interference which cause packet loss and deviation of final result. In order to limit this problem, in coordinate_cal(), data will be checked if it is over the max measuring range. If so, this group data is invalid and coordinate_cal() will return 0. On the contrary it will return 1.

i) **coordinate_cal()** is used to decode date pack from LTS. According to the transmission protocol of working mode TAG_FRAM0. The data about relative coordinate of tag has nine bytes length with three type int24 variables. Every type int24 variables has six byes and every byte follow this rule: the order of byte from left to right corresponds to the order from small to large in actual number. Especially int24 is not a normal variable type which can be understood by Arduino. So int24 has to be convert into uint32_t with following method:

```
uint32_t coo = (uint32_t)(byte[0]<<8|byte[1]<<16|byte[2]<<24)/ 256;
```

This method aims to move every byte and build a uint32_t variable. But in practical usage, Arduino seems can’t understand this order so this code is changed into:

```
coo = (((byte[0] << 8) & 0xffff) | (byte[1] * 0x10000) | (byte[2] * 0x1000000)) / 0x100;
```

After converting int24 into uint32_t, the data is already a normal number. Finally converting this hexadecimal number into decimal number and dividing it by 1000 to get a final float result. In this design only coordinate of tag is needed. So we just need to select byte pos.x, pos.y, pos.z to calculate.At the end of this sub-function. The data will be checked if its is valid and return 1 or 0 according to the result.

j) **processGPS(), processLTS(), calGPS_checksum()** are function used to verify data transmission state. In every signal transmission protocol, the first several bytes of data pack are called header and the last few bytes called checksum. Header is used to verify if the data pack is useful or needed. Checksum is used to verify if there are problems which can affect result during transmission. The check sum is obtained by adding up all the previous bytes and taking the lowest two bits.

k) **variance_cal()** is used to judge data changing fluctuation. After several test, the conclusion of variance drawn. When the quantities of data is fixed, variance of this group of data is only related to difference of max and min value of data. Because of the aim of improving location accuracy to 5cm. The difference of max and min value is ten and variance is calculated to be around 0.002. When variance is lower than 0.002, that means this group of data is reliable without big fluctuations and errors

2. ALGORITHM

a) **Moving distance calculation:** Every time when Arduino changes (pushing different button) its working mode (receiving different demands), rotary encoders need initialization. But when crawler is moving on a same direction, no matter it goes forward or backward, the distance should be a cumulative process which means distance shouldn't be reset to zero when working mode is changing between go forward and backward. So the following algorithm is created.

Distance = dislast + disforward + disbackward;

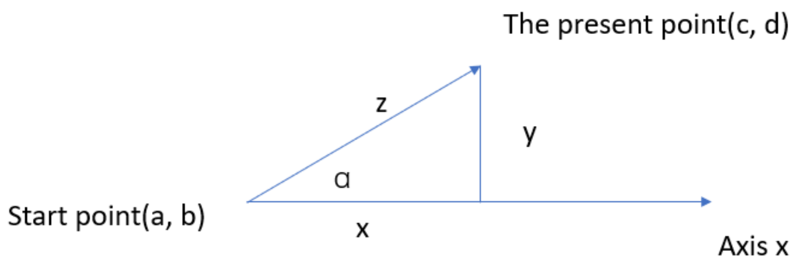
Table 4.5.2.1 Variable function

Variable name	Function
Distance	Total moving distance which will be displayed.
dislast	The moving distance before working mode changed.
disforward	Moving distance when crawler moving forward.

disbackward	Moving distance when crawler moving backward. (minus)

When mode is changed, corresponding variables will get value from rotary encoder and dislast will record moving distance before mode changed. Because of the feature of rotary encoder, it can't distinguish the moving direction of wheel. So when Arduino receives go backward order, Arduino will change the data from rotary encoder into a minus automatically.

b) **Moving direction calculation:** In final design, acquisition of moving direction information of crawler doesn't depend on compass or similar equipment but calculate by Arduino with coordinate of start point and end point (collecting from LTS). Every time when Arduino receives a go forward or backward order, it will automatically record the coordinate where it received the order as a start point. During crawler is moving forward or backward, Arduino keeps collecting coordinate from LTS and uses following algorithm to calculate moving direction:



Theoretical result

$Z = (x^2 + y^2)^{\wedge}0.5$

$\mathrm{Sin}(a) = y / z$

$\alpha = \arcsin(\alpha)$

Direction in this picture is north by east $90° - \alpha$

Figure 4.5.2.2 Direction calculation

V Conclusion and recommendation

5.1 Conclusion for research questions

1. Main question 1: What can be used to improve the accuracy of the navigation and location system

In short range location, using auxiliary location equipment whose accuracy is higher than GPS and has less signal interference from other signal is a good way to improve navigation and location accuracy. LTS indoor beacon system has higher accuracy and 10cm less error compared with GPS system which doesn't has external signal transmission with satellite. This means LTS system can be less affected by weather and other signal during long distance signal transmission and output high precision coordinate which can replace most function of GPS. What's more, rotary encoder with accuracy of 1cm and most 4cm error within 20m measurement can also improve overall accuracy during measuring distances.

2. Main question 2: What is the best method to improve the system for indoor usage and testing?

LTS indoor beacons system can improve indoor usage and testing enormously. Because of its working feature doesn't need external data transmission and also special transmission signal UWB which will not be affected by other signal strongly, so it shows good indoor working behaviors.

3. Sub question 1: What devices are relative to location system?

(1) C94-M8P GPS system (for original longitude and latitude)
(2) LTS indoor beacons system (For relative coordinate and moving direction)
(3) Rotary encoder (for moving distance)

4. Sub question 2: How many kinds of errors are there during location and what are corresponding solutions?

For GPS and LTS, they both have error from signal transmission which is related to the impedance of obstacles and interference from other signal can cause the decrease of signal strength and data deviation and finally cause errors. There are also other errors from weather, clock of satellite and also magnetic in atmosphere. But only errors from obstacles and signal interference is controllable. The best solution is using GPS and LTS in an open area. For GPS, users have to guarantee base station works in an open area without many metal obstacles to build strong connection with rover and satellite. For LTS, users have to make sure there is no obstacle within 4m of every beacon.

5. Sub question 3: What is parasitic effect and what can it affect?

Due to the wiring structure generated by the mutual capacitance and inductance phenomenon, appear in place without originally designed capacitors and inductors which called parasitic capacitors and parasitic inductors. And in the case of high frequency signal input, this part of circuit reveals the characteristic of capacitors and inductors. This phenomenon is called parasitic effect. Parasitic effect mainly affects energy consumption and has some possibility to affect the electrical level of digital signal.

6. Sub question 4: What cause the low efficiency of indoor usage and testing?

Now days, most building are built with reinforced concrete. Rebar in this kind of concrete has good Signal absorption characteristic which affects signal strength. Low signal strength can cause serious data calibration and failure of calibration.

7. Sub question 5: What can be used to increase indoor signal strength?

GPS signal repeater can be used indoor to amplify GPS signal. It receives GPS signal from the antenna outdoor and retransmit through amplifying circuit to indoor space.

5.2 Recommendation for future development

1. Moving direction problem

In this design, team failed to measure moving direction of crawler. Main reason should be related to the logic problem of code. But in fact, because of the working character of LTS and the algorithm of directing measurement, the data refresh frequency is also not very high which might cause data transmission delay.

Compass is a relative better solution, but when crawler is working on the wind turbine blades, there are large electromagnetic interference which can also affect accuracy of compass.

So in a conclusion, the direction problem is still need to be solver. Maybe it is a good solution to use the longitude and latitude and relative coordinate of anchors, distance between anchors and tags to solve this problem.

2. The highest speed of rotary encoder

When crawler moves in a high speed, the frequency of pulse change of photoelectric gate on rotary encoder might higher than clock frequency of Arduino of data transmission rate of communication port. So it is necessary to push it slowly. But the highest speed that rotary encoder can measured is still needed to be researched.

3. The type of rotary encoder

The working principle of rotary encoder used in this design is generate electric pulse to show the state of wheel. This type of rotary encoder can't judge rotating direction of wheel. There is a type of rotary encoder which can generate different types of pulses and different magnitude of current to show rotating direction of wheel. Using this kind of rotary encoder can decrease the volume of codes and calculation quantities of Arduino.

4. Remote control

According to the conclusion of test 5.2.3, the distances between obstacles and beacons have effect on data transmission, when users are controlling crawler, human bodies also have effect on transmission. So remote control can be used to avoid potential effects.

What's more, it is a better way to use remote controller to send control signal to replace button. For example, when using rocker remote controller, the controller can send different strength of signal according to the direction in which the rocker is pushed and degree of being pushed to tell crawler about its moving direction and velocity. In this way, if users can measure the first direction of crawler, then crawler can calculate direction automatically.

Remote control also means using wireless method to realize data transmission. This also another question needs to be solved

5. Three-dimensional location

The prototype now can only work in two dimensions. But the blade of wind turbine is not a two dimensions horizonal plate, so three-dimensional location is needed. In fact, GPS and LTS both have function of three-dimensional location but the accuracy of this function is relatively low compared with two-dimensional location.

Except the problem of accuracy, another problem of three-dimensional location is the expression of position in a three-dimension space. Compare with two-dimensional location, coordinate and direction also distance can't show the position of crawler directly in three-dimensional location. So a spatial model for wind turbine blade is needed and drawing moving track of crawler on it to show position is a good solution.

VI Reference

[1] NistlerJ.R&Selekwa,M.F. (2011). Gravity compensation in accelerometer measurements for robot navigation on inclined surface. Procedia Computer Science,6, 413-418.

[2] SOLUTIONSPCBCADENCE. (2019.5). How parasitic Capacitance and Inductance Affect Your Signals. Source: http://resources.pcb.cadence.com/blog/2019-how-parasitic-capa-citance-and-inductance-affect-your-signals

[3] ZhengYe, Yang FuCongjie. (2012). Research and Development of Operation and Maintance For Offshore Wind Farms.

[4] 香港科技大学. (2018.3.7). What is a parasitic effect. Source: Baiduwenku: http://wenku.baidu.com/view/aa3fff20482fb4daa58d4b6d

[5] 云里雾里科技. (2019.02). Do you know about 8 kinds of indoor location methods? Source: JIanshu: https://www.jianshu.com/p/a48b17be4a68

[6]
nanshanjinniu. (2018.3.15). 遥控控制原理. Source: https://www.baidu.com/link?url=hvF407
fvJwi1eazDDmmMW7xPCMuIwPhRXaE1dV7LBhrSUy3Vd1Vst_tMbbFRzaZFbUJyV-rIwUOj04i-Mrbkp
g6MYx_vFatHSN9i_ROFW_Zg0xRmDhqDcw9r-UlfHO7O&wd=&eqid=8b6b5cc9000b612500000004
5edb6215

[7] LINK TRACK S data sheet（2020）(Nooploop, 2020).

VII Appendix

7.1 Appendix I Test about C94-M8P GPS system

7.1.1 Test of dynamic accuracy and error of GPS

1. Test aim

This test aims to measure the dynamic accuracy and error of C94-M8P GPS system when it works in static RTK GPS mode.

2. Test equipment

C94-M8P GPS system with base station and rover
Arduino mega 2560
Ruler
Laptop

Table 7.1.1.1 Equipment of test of dynamic accuracy of GPS

3. Test variables

Variable	**State**
GPS working mode	Static RTK GPS
Data transmission	Wire connected with Arduino
Data collection control	Button
Distance between base station and rover	5-10m
Position	Outdoor garage
Weather	Sunny day without clouds and heavy wind
Temperature	18.5 C

Distance to the nearest obstacles (buildings, cars, trees)	＞10m
Set calibration location accuracy	1m
Set calibration observation time	300s

Table 7.1.1.2 Variables of test of dynamic accuracy of GPS

4. Test methods

a) Use ruler to measure distance between two random points on a plate.
b) Put rover on the first point.
c) Record the longitude and latitude of that point. (Arduino automatically collect data 10 times and out an average value).
d) Repeat the same thing on the next point chosen. Make sure no pause during moving the rover and recording data.
e) Do step a) to d) 6 times
f) Processor of Arduino will automatically calculate distance according to longitude and latitude of two points with following formula
g) Compare collected data with real distance and calculate error and accuracy

Formula 1: Distance calculation

$$\text{haversin}\left(\frac{d}{R}\right) = \text{haversin}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\text{haversin}(\Delta\lambda)$$

$$\text{haversin}(\theta) = \sin^2(\theta/2) = (1 - \cos(\theta))/2$$

Haversine formula:
R: radius of earth, 6731 Km
$\Delta\lambda$: difference of two points longitude
Φ_1, Φ_2 :latitude of two points

Formula 2: Accuracy calculation

Real value – Average value = Accuracy

5. Desired results

The calculated distance between two points has an error less than 5cm and calculated accuracy of GPS is higher than 5cm.

6. DATA AND RESULT

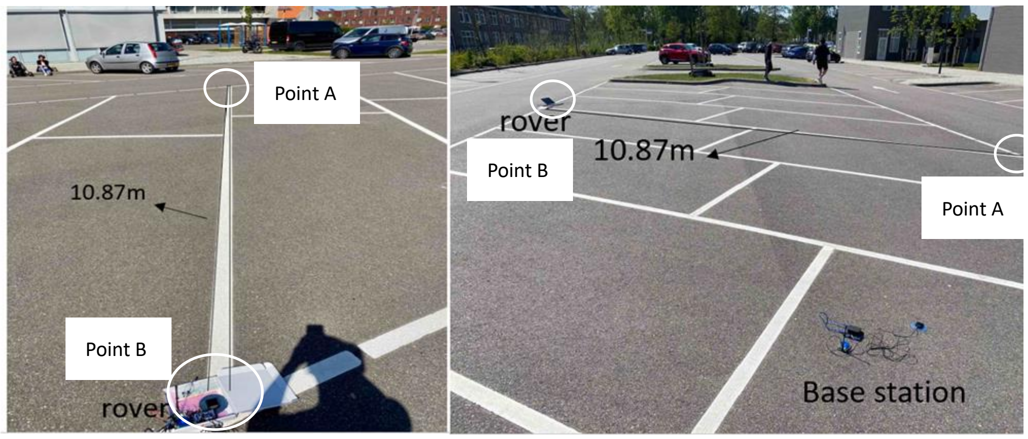
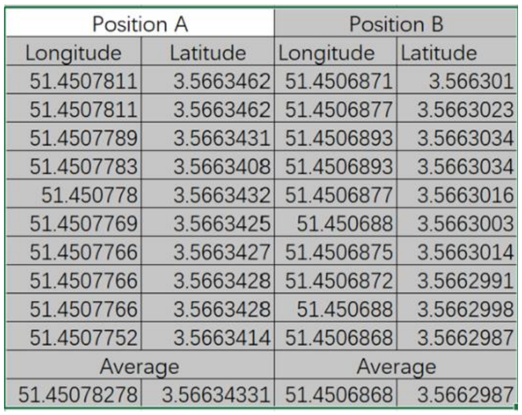


Figure 7.1.1.1 Test of dynamic accuracy of GPS

Team chose a straight line on the garage and measured the distance is 10.87m. After 6 times moving of rover and data collection, team got following data:



Position A		Position B	
Longitude	Latitude	Longitude	Latitude
51.4507811	3.5663462	51.4506871	3.566301
51.4507811	3.5663462	51.4506877	3.5663023
51.4507789	3.5663431	51.4506893	3.5663034
51.4507783	3.5663408	51.4506893	3.5663034
51.450778	3.5663432	51.4506877	3.5663016
51.4507769	3.5663425	51.450688	3.5663003
51.4507766	3.5663427	51.4506875	3.5663014
51.4507766	3.5663428	51.4506872	3.5662991
51.4507766	3.5663428	51.450688	3.5662998
51.4507752	3.5663414	51.4506868	3.5662987
Average		Average	
51.45078278	3.56634331	51.4506868	3.5662987

Figure 7.1.1.3 Results of test of dynamic accuracy of GPS 1

Distance(m)	Error(cm)	Distance(m)	Error(cm)	Average distance (cm)	Average error (cm)
10.787	8.3	10.776	9.4	10.860	11.03
10.809	6.1	10.754	12.5		
11.012	14.2	11.027	15.7		
Real distance: 10.87m				***Accuracy: 1 cm***	

Table 7.1.1.4 Results of test of dynamic accuracy of GPS 2

Picture shows 10 groups of coordinates collected by Arduino and final output value and list shows the result after six-time repeated measure. Final average error of GPS system is 11.03 cm and accuracy reaches 1cm.

7. CONCLUSION

GPS system doesn't reach the desired error range when GPS model is moving. The errors always change from 6cm to 16cm and very unstable which means GPS system can't be used on a moving crawler. The accuracy of static state GPS is still needed to be measured.

7.1.2 Test of static accuracy and error of GPS

1. TEST AIM

This test aims to measure the static accuracy and error of C94-M8P GPS system when it works in static RTK GPS mode.

2. TEST EQUIPMENT

C94-M8P GPS system with base station and rover
Arduino mega 2560
Ruler
Laptop

Table 7.1.2.1 Equipment of test of static accuracy of GPS

3. TEST VARIABLES

Variable	State
GPS working mode	Static RTK GPS
Data transmission	Wire connected with Arduino
Data collection control	Button
Distance between base station and rover	5-10m

Position	Outdoor garage
Weather	Sunny day without clouds and heavy wind
Temperature	18.5 C
Distance to the nearest obstacles (buildings, cars, trees)	$>$ 10m
Set calibration location accuracy	1m
Set calibration observation time	300s

Table 7.1.2.2 Variables of test of static accuracy of GPS

4. TEST METHODS

a) Use ruler to measure distance between two random points on a plate.
b) Put rover on the first point
c) Monitor the output data on Arduino IDE and u-center and wait about 5min until GPS data becomes stable (Stable means accuracy of GPS shown by u-center has a change less than 10cm)
d) Record the longitude and latitude of that point. (Arduino automatically collect data 10 times and out an average value).
e) Repeat the same thing on the next point chosen.
f) Do step a) to e) 6 times
g) Processor of Arduino will automatically calculate distance according to longitude and latitude of two points with following formula
h) Compare collected data with real distance and calculate error and accuracy

Formula 1: Distance calculation

$$\operatorname{haversin}\left(\frac{d}{R}\right) = \operatorname{haversin}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{haversin}(\Delta\lambda)$$

$$\operatorname{haversin}(\theta) = \sin^2(\theta/2) = (1 - \cos(\theta))/2$$

Haversine formula:
R: radius of earth, 6731 Km
$\Delta\lambda$: difference of two points longitude
ϕ_1, ϕ_2 :latitude of two points

Formula 2: Accuracy calculation

Real value – Average value = Accuracy

5. Desired results

The most difference between this and last test is a stable state of GPS. Compare with moving GPS, static should have higher accuracy and less errors. The calculated distance between two points has an error less than 5cm and calculated accuracy of GPS is higher than 5cm.

6. Data and result

Team chose the same line as test 5.1.1. After 6 times moving of rover and data collection, team got following data:

Table 7.1.2.3 Result of test of static accuracy of GPS

Distance(m)	Error(cm)	Distance(m)	Error(cm)	Average distance (cm)	Average error (cm)
10.811	5.9	10.817	5.3	10.851	4.77
10.828	4.2	10.856	1.4		
10.956	8.6	10.838	3.2		
Real distance: 10.87m				Accuracy: 2.1 cm	

In this list, some groups of errors are more than 5cm but the final average error is only 4.77cm. The accuracy of GPS is 2.1cm.

7. Conclusion

Although some errors are over the max error limit, compared with dynamic accuracy and error, static error has already had a great improvement with almost 10cm change. These results show when rover is static, GPS has almost same accuracy as dynamic rover but less error. This character decides the final function of GPS, collecting original longitude and latitude.

7.2 Appendix Ⅱ Test about LTS indoor beacon system

7.2.1 Indoor accuracy and error of LTS

1. Test aim

This test aims to measure the accuracy and error of LTS indoor beacon system when it works in LP mode and works indoor.

2. Test equipment

LTS system with four anchors and one tag
Arduino mega 2560
Ruler
Laptop

Table 7.2.1.1 Equipment of test of indoor accuracy and error of LTS

3. Test variables

Variable	**State**
LTS working mode	LP mode (Tag_Frame_0) for tag
Data transmission	Wire connected with Arduino
Data collection control	Auto
Distance between tag and anchor	0.5-8m
Position	Dormitory

Table 7.2.1.2 Variables of test of accuracy and error of LTS

4. Test methods

a) Choose five random point in the room
b) Use ruler to measure vertical distances between these points and the place where axis X and Y in coordinate system created by LTS system is.
c) Use measured distances as coordinates of these points which is called standard point.
d) Put tag at one point and record data on NAssistant.

e) Repeat the same thing on the other points.
f) Select 100 groups of data of one point from data recorded by NAssitant and calculate the average coordinate of every point.
g) Compare average coordinate with standard coordinate.
h) Analyze the track of tag and variation trend of 100 groups of data collected from NAssitant

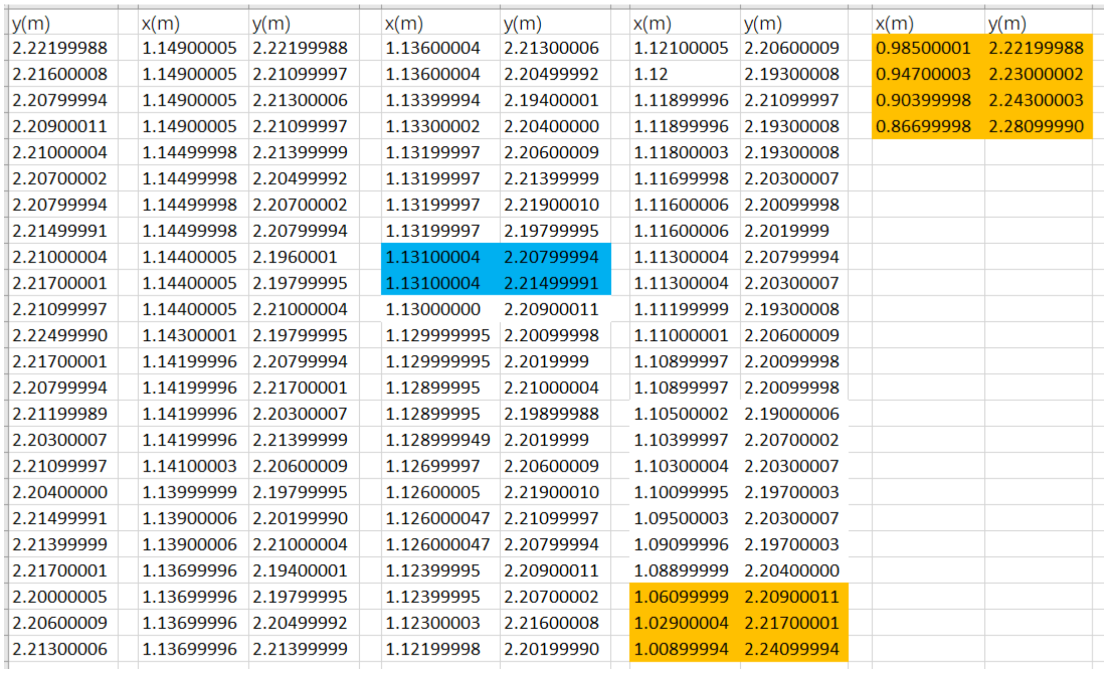
5. Desired results

The LTS can have an accuracy higher than 5cm. The error between standard and average coordinate can be less than 5cm. The fluctuation of data and track can be less than 10cm.

6. Data and result

standard coordinate	average coordinate	error	average error
(2.05, 1.40)	(2.11, 1.36)	(0.06, 0.04)	(4.4, 1.9)
(1.13, 2.25)	(1.14, 2.25)	(0.01, 0.00)	
(2.50, 2.30)	(2.44, 2.31)	(0.06, 0.01)	
(0.50, 3.10)	(0.53, 3.09)	(0.03, 0.01)	
(3.10, 0.50)	(3.10, 0.55)	(0.00, 0.05)	
(3.70, 4.00)	(3.69, 4.01)	(0.01, 0.01)	
(1.00, 4.60,)	(1.07, 4.58)	(0.07, 0.02)	
(0.50, 0.50)	(0.46, 0.51)	(0.04, 0.01)	
(4.20, 5.60)	(4.35, 5.63)	(0.15, 0.03)	
(4.70, 3.70)	(4.71, 3.71)	(0.01, 0.01)	

Table 7.2.1.3 Result of test of indoor accuracy and error of LTS 1



y(m)	x(m)	y(m)	x(m)	y(m)	x(m)	y(m)	x(m)	y(m)
2.22199988	1.14900005	2.22199988	1.13600004	2.21300006	1.12100005	2.20600009	0.98500001	2.22199988
2.21600008	1.14900005	2.21099997	1.13600004	2.20499992	1.12	2.19300008	0.94700003	2.23000002
2.20799994	1.14900005	2.21300006	1.13399994	2.19400001	1.11899996	2.21099997	0.90399998	2.24300003
2.20900011	1.14900005	2.21099997	1.13300002	2.20400000	1.11899996	2.19300008	0.86699998	2.28099990
2.21000004	1.14499998	2.21399999	1.13199997	2.20600009	1.11800003	2.19300008		
2.20700002	1.14499998	2.20499992	1.13199997	2.21399999	1.11699998	2.20300007		
2.20799994	1.14499998	2.20700002	1.13199997	2.21900010	1.11600006	2.20099998		
2.21499991	1.14499998	2.20799994	1.13199997	2.19799995	1.11600006	2.2019999		
2.21000004	1.14400005	2.1960001	1.13100004	2.20799994	1.11300004	2.20799994		
2.21700001	1.14400005	2.19799995	1.13100004	2.21499991	1.11300004	2.20300007		
2.21099997	1.14400005	2.21000004	1.13000000	2.20900011	1.11199999	2.19300008		
2.22499990	1.14300001	2.19799995	1.129999995	2.20099998	1.11000001	2.20600009		
2.21700001	1.14199996	2.20799994	1.129999995	2.2019999	1.10899997	2.20099998		
2.20799994	1.14199996	2.21700001	1.12899995	2.21000004	1.10899997	2.20099998		
2.21199989	1.14199996	2.20300007	1.12899995	2.19899988	1.10500002	2.19000006		
2.20300007	1.14199996	2.21399999	1.128999949	2.2019999	1.10399997	2.20700002		
2.21099997	1.14100003	2.20600009	1.12699997	2.20600009	1.10300004	2.20300007		
2.20400000	1.13999999	2.19799995	1.12600005	2.21900010	1.10099995	2.19700003		
2.21499991	1.13900006	2.20199990	1.126000047	2.21099997	1.09500003	2.20300007		
2.21399999	1.13900006	2.21000004	1.126000047	2.20799994	1.09099996	2.19700003		
2.21700001	1.13699996	2.19400001	1.12399995	2.20900011	1.08899999	2.20400000		
2.20000005	1.13699996	2.19799995	1.12399995	2.20700002	1.06099999	2.20900011		
2.20600009	1.13699996	2.20499992	1.12300003	2.21600008	1.02900004	2.21700001		
2.21300006	1.13699996	2.21399999	1.12199998	2.20199990	1.00899994	2.24099994		

Table 7.2.1.4 Result of test of indoor accuracy and error of LTS 2

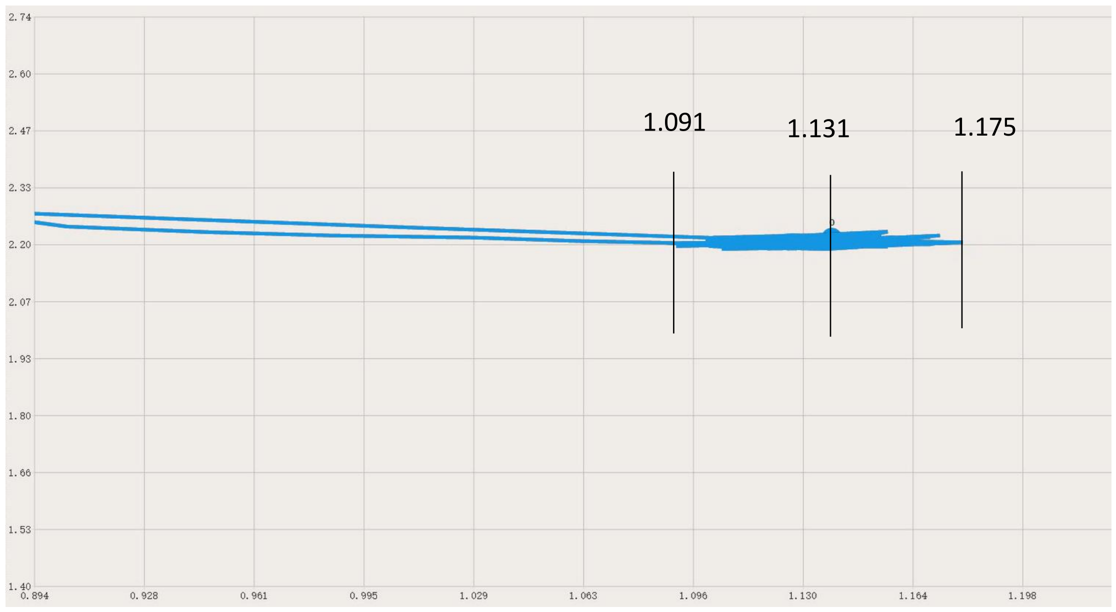


Figure 7.2.1.1 Result of test of indoor accuracy and error of LTS

The first list shows standard coordinate, average coordinate calculated with 100 groups of data and also error of every point and average error.

This second list shows 100 groups of data collected at the second we chose. Data in forms with blue background is the mid-value of these data. Data in forms with yellow background is data which has error more than 5cm.

This picture shows the track of tag when it is put at the second point. Except the biggest fluctuation, the max value is 1.175 and min value is 1.091. (Not corresponding to the second list)

7. Conclusion

According to list1, although some average coordinates have errors more than 5cm but average error is less than 5cm. Especially error of Y is less than error of X.

According to list2, in 100 groups of data, there are only 7 groups are out of limit which means most data from LTS system can meet the request of control error less than 5cm.

According to the final picture, the changing trend of tag can be directly observed. Max and min values all have error less than 5cm compared with mid-value. The biggest fluctuation is the error when quantities, distance of obstacles between tag and anchors change quickly.

In a conclusion, although the error of LTS can be controlled very well, but an algorithm is still needed to let Arduino learn to judge the fluctuation of data and eliminate invalid and unstable data to improve

7.2.2 Outdoor accuracy and error of LTS

1. Test aim

This test aims to measure the accuracy and error of LTS indoor beacon system when it works in LP mode and works outdoor.

2. Test equipment

LTS system with four anchors and one tag
Arduino mega 2560
Adafruit 932 OLED screen
Ruler

Table 7.2.2.1 Equipment of test of outdoor accuracy and error of LTS

3. Test variables

Variable	**State**
LTS working mode	LP mode (Tag_Frame_0) for tag
Data transmission	Connect tag with laptop by wire
Data collection control	Auto
Distance between tag and anchor	0.5-35m
Position	Outdoor garage
Weather	Sunny day without clouds and heavy wind
Temperature	20.3 C

Table 7.2.2.2 Variables of test of outdoor accuracy and error of LTS

4. Test methods

a) Choose five random point on the garage.

b) Use ruler to measure vertical distances between these points and the place where axis X and Y in coordinate system created by LTS system is.

c) Use measured distances as coordinates of these points which is called standard point.

d) Put tag at one point and read data from OLED screen. Arduino will receive data 10 times and calculate average coordinate and process data with variance algorithm automatically. Record 5 groups of data at this point.

e) Repeat the same thing on the other points.

f) Compare average coordinate with standard coordinate.

5. Desired results

The LTS can have an accuracy higher than 5cm. The error between standard and average coordinate can be less than 5cm. Less data fluctuation appears.

6. Data and result

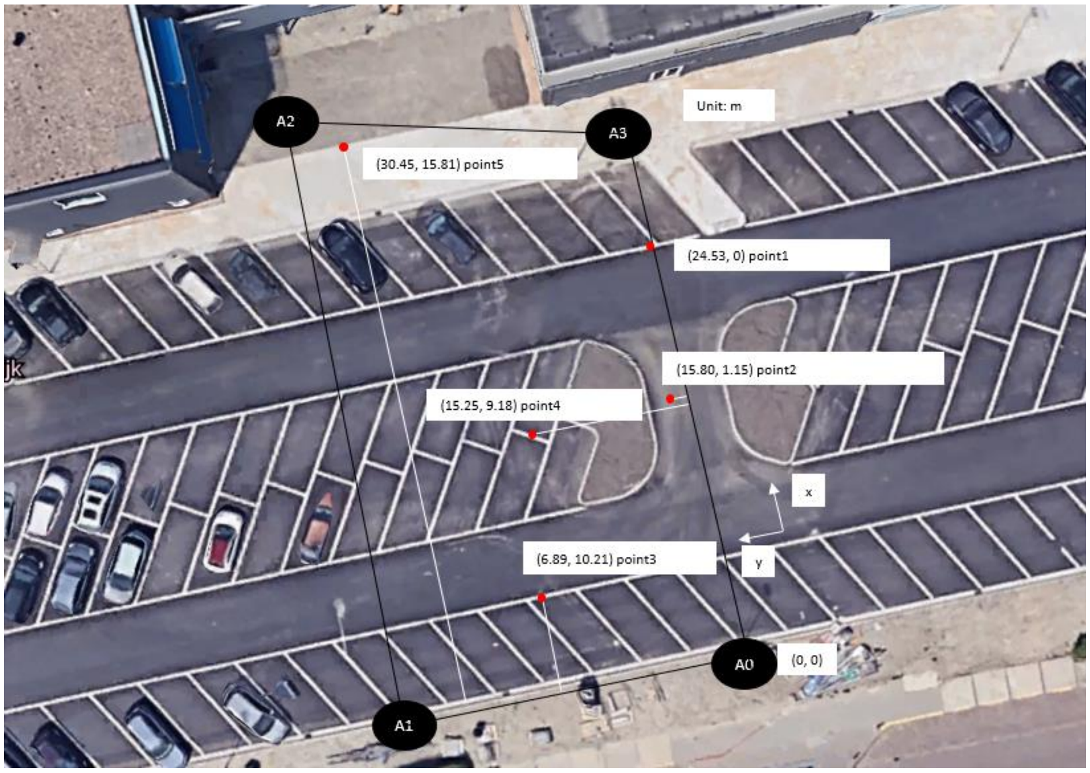
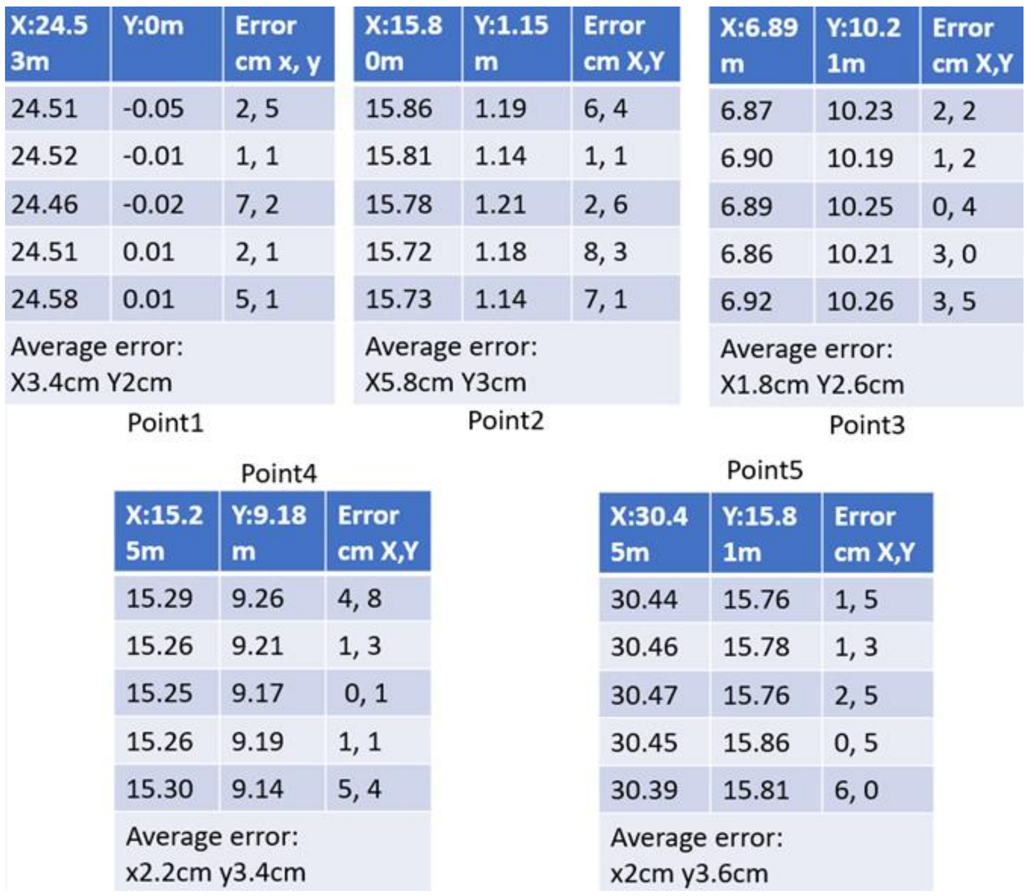


Figure 7.2.2.1 Result of test of outdoor accuracy and error of LTS

This picture shows the test position. And background picture is from google map. There was no car during the test was being done.



X:24.5 3m	Y:0m	Error cm x, y
24.51	-0.05	2, 5
24.52	-0.01	1, 1
24.46	-0.02	7, 2
24.51	0.01	2, 1
24.58	0.01	5, 1
Average error: X3.4cm Y2cm		

Point1

X:15.8 0m	Y:1.15 m	Error cm X,Y
15.86	1.19	6, 4
15.81	1.14	1, 1
15.78	1.21	2, 6
15.72	1.18	8, 3
15.73	1.14	7, 1
Average error: X5.8cm Y3cm		

Point2

X:6.89 m	Y:10.2 1m	Error cm X,Y
6.87	10.23	2, 2
6.90	10.19	1, 2
6.89	10.25	0, 4
6.86	10.21	3, 0
6.92	10.26	3, 5
Average error: X1.8cm Y2.6cm		

Point3

Point4

X:15.2 5m	Y:9.18 m	Error cm X,Y
15.29	9.26	4, 8
15.26	9.21	1, 3
15.25	9.17	0, 1
15.26	9.19	1, 1
15.30	9.14	5, 4
Average error: x2.2cm y3.4cm		

Point5

X:30.4 5m	Y:15.8 1m	Error cm X,Y
30.44	15.76	1, 5
30.46	15.78	1, 3
30.47	15.76	2, 5
30.45	15.86	0, 5
30.39	15.81	6, 0
Average error: x2cm y3.6cm		

Table 7.2.2.3 Result of test of outdoor accuracy and error of LTS

Except the error of second point has a little over than 5cm, errors of other points are all less than 5cm. The result of direction is not show here. Team failed to measure the moving direction. The moving distance always shown 180, -180, 360, 0 degree.

7. Conclusion

All data shows, when indoor system is working outdoor, it can control error under 5cm. This means this system can be used for outdoor small range location.

When it comes to the data of second point, during the test, there are some tree next to the ponit2. These trees can be obstacles which effect signal transmission and cause data deviation. A test of relationship between data deviation and distance between beacons and obstacles is needed.

The failure of measuring moving direction should be related to final code which is need to be checked

7.2.3 Relationship of data deviation and distance between LTS beacons and obstacles

1. Test aim

This test aims to measure the Relationship of data deviation and distance between LTS beacons and obstacles.

2. Test equipment

LTS system with four anchors and one tag
Laptop

Table 7.2.3.1 Equipment of test of relationship between beacons and obstacles

3. Test variables

Variable	State
LTS working mode	LP mode (Tag_Frame_0) for tag
Data transmission	Connect tag with laptop by wire
Data collection control	Auto
Distance between tag and anchor	0.5-30m
Position	Outdoor garage
Weather	Sunny day without clouds and heavy wind
Temperature	20.3 C

Table 7.2.3.1 Variables of test of relationship between beacons and obstacles

4. Test methods

a) Choose a random point on the garage to place the tag
b) Put obstacles between one anchor and tag. A 1.5 * 1.5m board is used in this test
c) Change distance between anchor and obstacles from 1m to 5m
d) Record the tag track graph on NAssitant every 1m.
e) Analyze tag track graphs and conclude the relationship.

5. Desired results

The effect to data deviation from distance between obstacles and beacons can be decreased as distance becomes longer.

6. Data and result

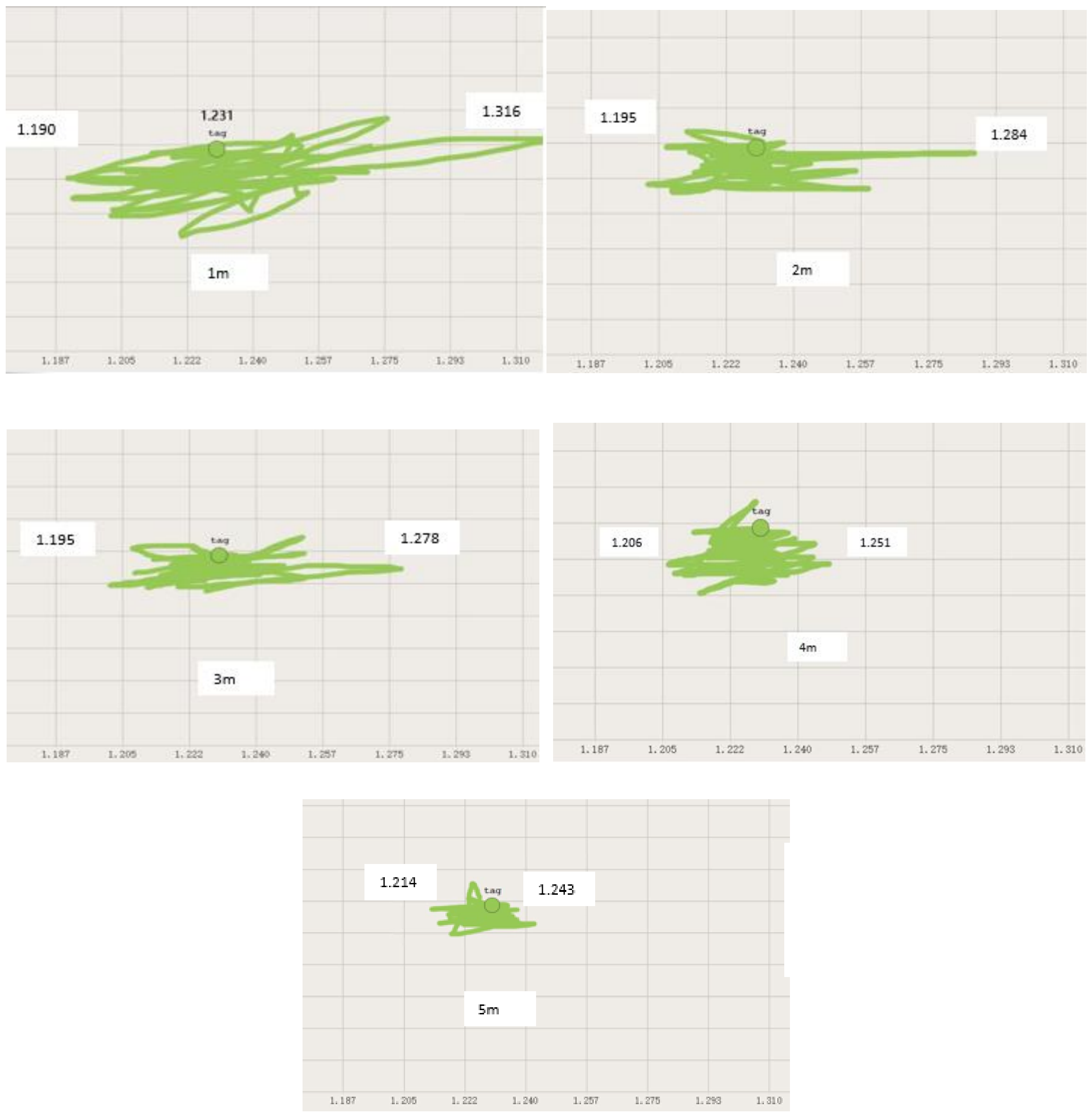


Figure 7.2.3.1 Results of test of relationship between beacons and obstacles

7. Conclusion

From five pictures, as the distance between anchor and obstacle becomes longer, the max and min value of data change become smaller and bigger. The differences between mid-value and max, min value are less which means data deviation decreases.

Especially when distance is more than 4m, there is no obvious fluctuation appears. Because of the working principle of LTS system. During measurement, there are only data transmissions between anchors and tag. So effect is same when distance between obstacles and tag. Users need to check if there are obstacles within 4m of every beacon during usage.

7.3 Appendix III Test about HC-020K rotary encoder

7.3.1 Test of accuracy and error of rotary encoder in short distance

1. Test aim

This test aims to measure the accuracy and error of LTS indoor beacon system when it works in LP mode and works outdoor.

2. Test equipment

HC-020K rotary encoder
Arduino mega 2560
Car kit with wheels
Ruler
Laptop

Table 7.3.1.1 Equipment of test of accuracy and error of rotary encoder in short distance

3. Test variables

Variable	State
Data transmission	Wire connected with Arduino
Data collection control	Auto
Position	Dormitory

Table 7.3.1.2 Variables of test of accuracy and error of rotary encoder in short distance

4. Test methods

a) Fix a 15cm ruler on the table
b) Align axle of wheel at the zero line of ruler.
c) Push car kit slowly and smoothly until axle aligns at 15cm line of ruler.
d) Monitor data on ArduinoIDE and record data every 15cm
e) Reset rotary encoder and do repeat test for 5 times.
f) Calculate average errors.

5. Desired results

Rotary encoder can output moving distances with errors less than 5cm.

6. Data and result

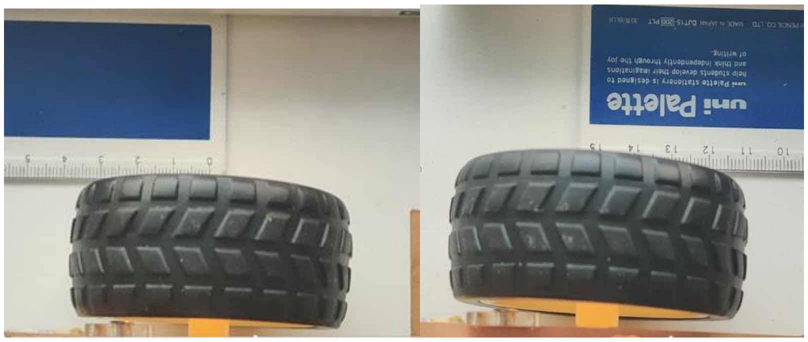


Figure 7.3.1.1 Results of test of accuracy and error of rotary encoder in short distance

Times	Distances	Error	Average error
1	14.86cm	0.14cm	0.14cm
2	14.86cm	0.14cm	
3	14.86cm	0.14cm	
4	14.86cm	0.14cm	
5	14.86cm	0.14cm	

Table 7.3.1.3 Results of test of accuracy and error of rotary encoder in short distance

The list shows distances measured by rotary encoder are all same. And average error is very small.

7. Conclusion

When moving distance is very short, rotary encoder can output very accurate results with small errors which can be ignored.

7.3.2 Test of accuracy and error of rotary encoder in long distance

1. Test aim

This test aims to measure the accuracy and error of LTS indoor beacon system when it works in LP mode and works outdoor.

2. Test equipment

HC-020K rotary encoder
Arduino mega 2560
Car kit with wheels
Ruler
Laptop

Table 7.3.1.1 Equipment of test of accuracy and error of rotary encoder in long distance

3. Test variables

Variable	State
Data transmission	Wire connected with Arduino
Data collection control	Auto
Position	Outdoor garage

Table 7.3.2.2 Variables of test of accuracy and error of rotary encoder in long distance

4. Test methods

a) Choose a random line on a plat and measured length.
b) Put wheels of car kit at the start of line.
c) Push car kit slowly and smoothly until the end of line.
d) Monitor data on ArduinoIDE and record data.
e) Keep push same distance on same direction without reset.
f) Repeat j) and k) total 5 times.
g) Calculate errors of every length of movement

5. Desired results

Rotary encoder can still output moving distances with errors less than 5cm.

6. Data and Result

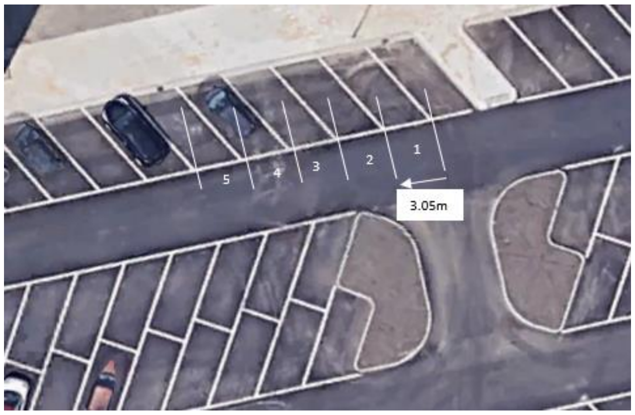


Figure 7.3.2.1 Results of test of accuracy and error of rotary encoder in long distance

	dis1	dis2	dis3	dis4	dis5
Standard	3.05m	6.10m	9.15m	12.20m	15.25m
Measured	3.04m	6.10m	9.16m	12.22m	15.28m
Error	1cm	0cm	1cm	2cm	3cm

Table 7.3.2.3 Results of test of accuracy and error of rotary encoder in long distance

The list shows when rotary encoder measures long distance, errors are much bigger than measuring short distance but still less than 5cm.

7. Conclusion

When moving distance is increasing, errors from rotary encoder are still increasing. Rotary encoder is reliable for the location system when moving distance is less than 15.25m.

7.4 Appendix IV Final test

1. Test aim

This test aims to measure the dynamic accuracy and error of C94-M8P GPS system when it works in static RTK GPS mode.

2. Test equipment

HC-020K rotary encoder
Car kit with wheels
LTS system with four anchors and one tag
Adafruit 932 OLED screen
Arduino mega 2560
Ruler

Table 7.4.1 Equipment of final test

Because the longitude and latitude of every measured point is calculated rather than measured and this test mainly focuses on accuracy of combination of LTS and rotary encoder, so GPS were not used in this test.

3. Test variables

Variable	**State**
LTS working mode	LP mode (Tag_Frame_0) for tag
Data transmission	Wire connected with Arduino
Data collection control	Button
Distance between tag and anchor	0.5-8m
Position	Outdoor garage

Table 7.4.2 Variables of final test

4. Test methods

a) Choose random 5 points on a plate
b) Use ruler to measure vertical distances between these points and the place where axis X and Y in coordinate system created by LTS system is.
c) Use measured distances as coordinates of these points which is called standard point.
d) Use ruler to measure distance between every point.
e) Put prototype at the first point.
f) Read and record the coordinate of this point from screen
g) Push button1 and push prototype to the next point smoothly and slowly.
h) Read and record moving distance and direction from screen.
i) Read and record the coordinate of this point from screen
j) Press button5 to reset.
k) Press button3 or 4 to send turn left or right order. Then push prototype turning. Make sure the wheel on the side which direction the protype is turning shouldn't move or rotate.
l) After turning press button5 to reset.
m) Repeat step g) to i)

5. Desired results

The final prototype has accuracy higher than 5cm and errors less than 5cm

6. Data and result

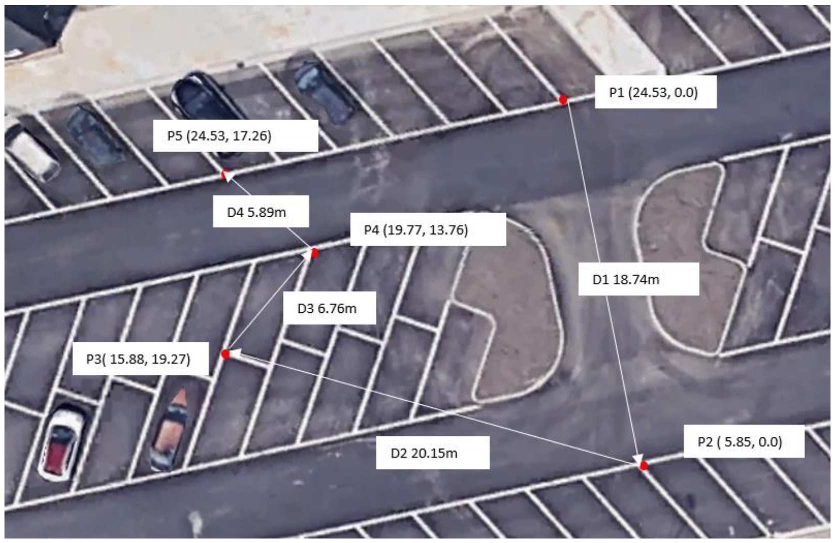


Figure 7.4.1 Results of final test

x	y
24.53	0
24.54	0.02
1cm	2cm

Point1

5.85	0
5.79	-0.03
6cm	3cm

Point2

15.88	19.27
15.94	19.26
6cm	1cm

Point3

19.77	13.76
19.77	13.74
0cm	2cm

Point4

24.53	17.26
24.58	17.27
5cm	1cm

Point5

18.74
18.77
3cm

Distance1

20.15
20.19
4cm

Distance2

6.76
6.77
1cm

Distance3

5.89
5.89
0cm

Distance4

Table 7.4.3 Results of final test

7. Conclusion

According to the final result, although some errors of some points are over 5cm, most data is reliable. Especially there is only once data calculation at every point. Final prototype shows nice accuracy when there are not many obstacles next to the measured points.

7.5 Appendix V Specification of final prototype

1. Specification

Highest Accuracy	1 cm
Average Location Error on X, Y	3.6cm, 1.8cm
Average Distance Error in 20m	4cm
Max Signal Transmission Distance of LTS	80m between every beacon
Max Signal Transmission Distance of GPS	238m between base station and rover
Max Location Area	640m²
Working Voltage	5v

2. Component List

Crawler-mounted Component	
C94-M8P GPS module	1 (Rover)
LinkTrackS beacon	1 (Tag)
HC-020K Rotary Encoder	2
Adafruit 932 OLED screen	1
Arduino MEGA 2560	1
Button	4
Bread board	1

External Component	
C94-M8P GPS module	1 (Base station)
LinkTrackS beacon	4 (Anchor)

3. Operation Instruction

STEPS:

a) For first time use, please calibrate GPS and LTS system according to the instruction of each system.

b) After calibration, place prototype on a plate.

c) Press reset button on Arduino to reset the whole system. Arduino will collect original longitude and latitude information automatically.

d) After seeing longitude and latitude information on screen (change from 0 to a real number). Press button 1 to 4 to send go forward, backward, turn left and right order (direction order) to Arduino.

e) Then push the prototype to the direction which is corresponding to direction order and read data from screen. Don't change moving direction before sending a next order.

f) After the end of movement, before changing moving direction, press button 5 to reset data and press corresponding direction button to send order.

g) If users want to change original start point (original longitude and latitude).

NOTICE:

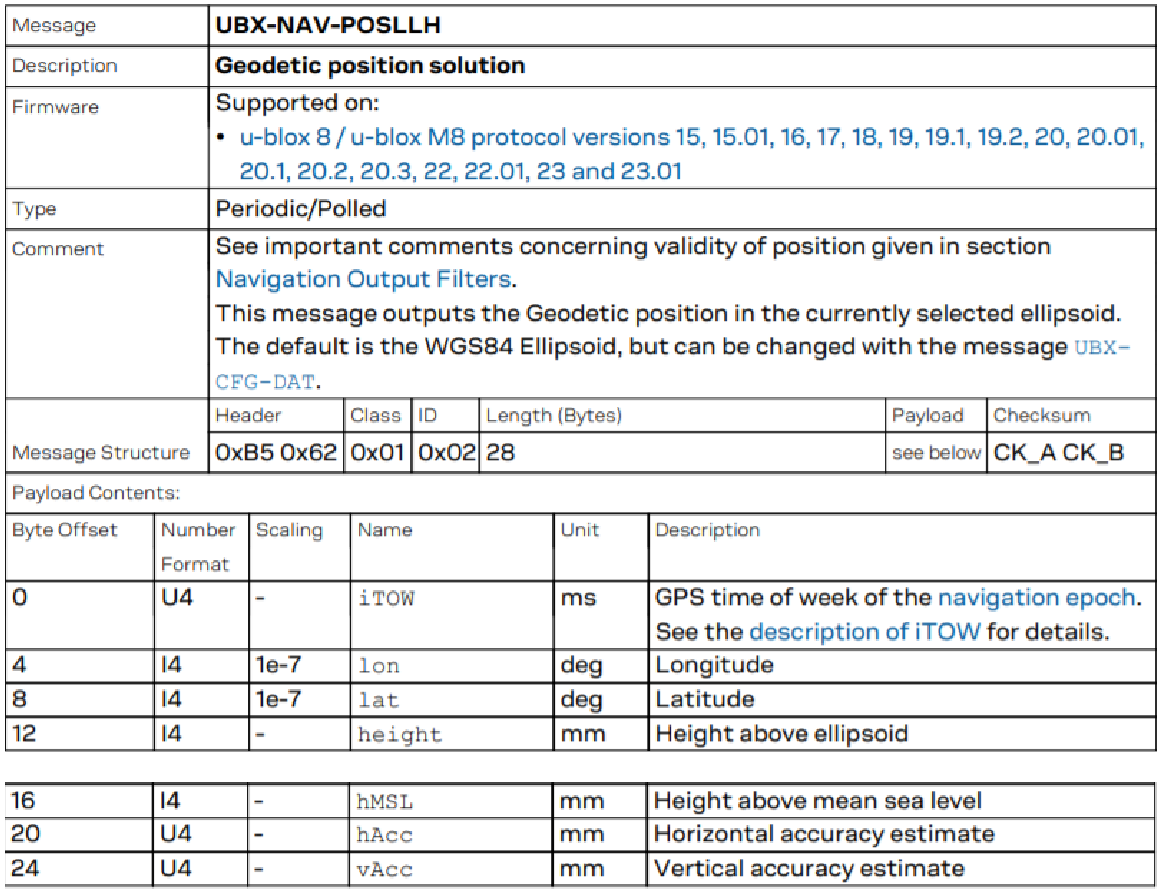
a) In this design, the use of button is to simulate control signal from remote controller and send order to Arduino. So the function of button is non-negligible. Don't forget to press button when moving direction of crawler is changed to guarantee measurement accuracy.

b) For every first-time calibration at new use sites, users have to make sure base station of GPS is placed in an open area to get best signal strength of GNSS signal. This can decrease calibration time obviously. GPS module will calibrate automatically after every restart (from power off to power on). If place of base station changes during usage, user don't need to worry. Except in situation of collecting original longitude and latitude, the movement of base station will not affect final result. Users can change calibration standard in u-center.

c) For LTS system, user also only needs to calibrate once on NAssistant at every usage site and make sure the static state of anchors during usage. If anchors are moved, users need to recalibrate the LTS system on NAssistant. Calibration will not take much time.

7.6 Appendix VI Transmission Protocol of GPS and LTS

1. GPS Transmission Protocol



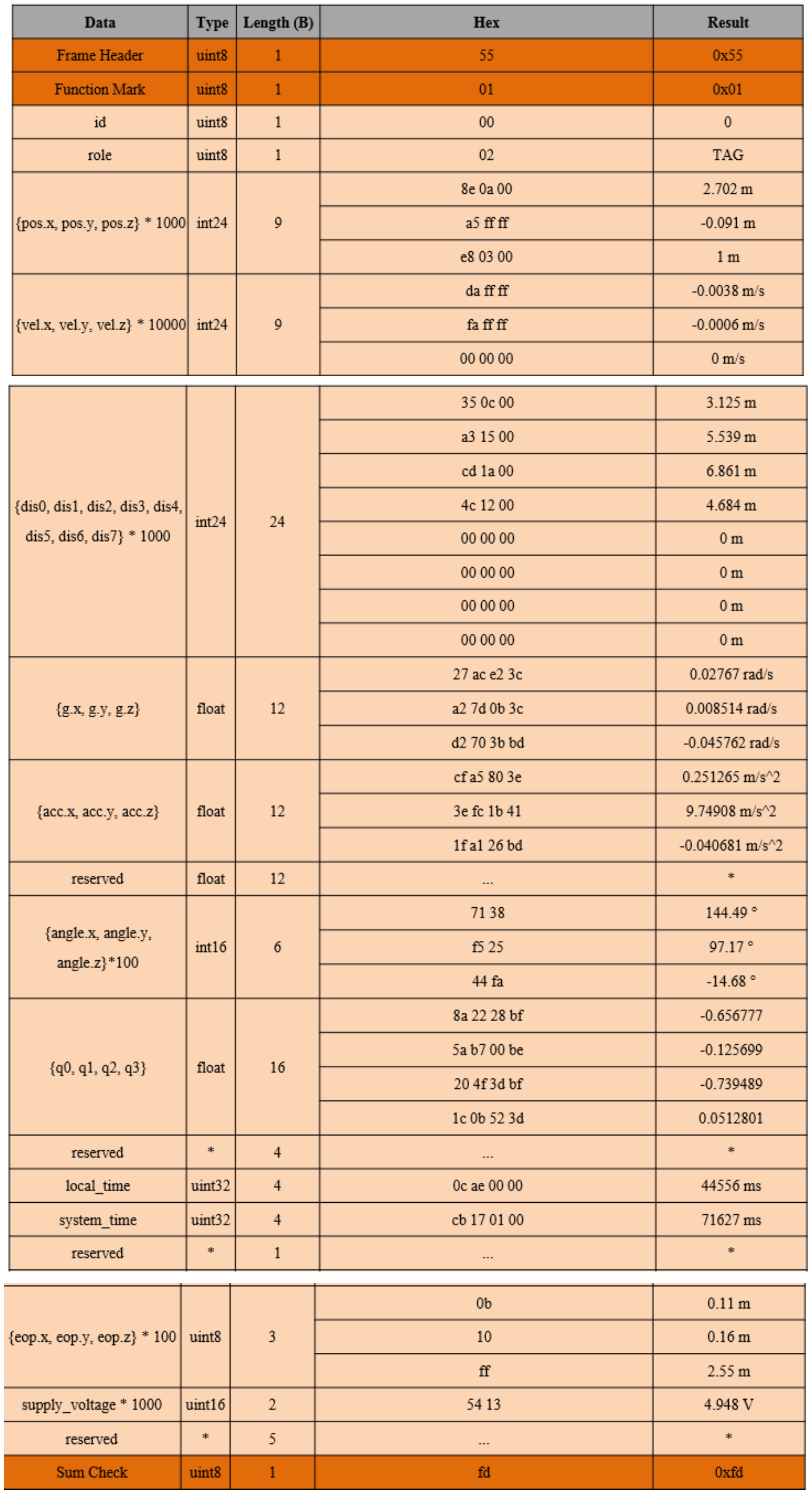
Message	**UBX-NAV-POSLLH**					
Description	**Geodetic position solution**					
Firmware	Supported on: • u-blox 8 / u-blox M8 protocol versions 15, 15.01, 16, 17, 18, 19, 19.1, 19.2, 20, 20.01, 20.1, 20.2, 20.3, 22, 22.01, 23 and 23.01					
Type	Periodic/Polled					
Comment	See important comments concerning validity of position given in section Navigation Output Filters. This message outputs the Geodetic position in the currently selected ellipsoid. The default is the WGS84 Ellipsoid, but can be changed with the message UBX-CFG-DAT.					
	Header	Class	ID	Length (Bytes)	Payload	Checksum
Message Structure	0xB5 0x62	0x01	0x02	28	see below	CK_A CK_B

Payload Contents:

Byte Offset	Number Format	Scaling	Name	Unit	Description
0	U4	-	iTOW	ms	GPS time of week of the navigation epoch. See the description of iTOW for details.
4	I4	1e-7	lon	deg	Longitude
8	I4	1e-7	lat	deg	Latitude
12	I4	-	height	mm	Height above ellipsoid
16	I4	-	hMSL	mm	Height above mean sea level
20	U4	-	hAcc	mm	Horizontal accuracy estimate
24	U4	-	vAcc	mm	Vertical accuracy estimate

Table7.6.1 GPS Transmission Protocol

2. LTS Transmission Protocol



Data	Type	Length (B)	Hex	Result
Frame Header	uint8	1	55	0x55
Function Mark	uint8	1	01	0x01
id	uint8	1	00	0
role	uint8	1	02	TAG
{pos.x, pos.y, pos.z} * 1000	int24	9	8e 0a 00	2.702 m
			a5 ff ff	-0.091 m
			e8 03 00	1 m
{vel.x, vel.y, vel.z} * 10000	int24	9	da ff ff	-0.0038 m/s
			fa ff ff	-0.0006 m/s
			00 00 00	0 m/s
{dis0, dis1, dis2, dis3, dis4, dis5, dis6, dis7} * 1000	int24	24	35 0c 00	3.125 m
			a3 15 00	5.539 m
			cd 1a 00	6.861 m
			4c 12 00	4.684 m
			00 00 00	0 m
			00 00 00	0 m
			00 00 00	0 m
			00 00 00	0 m
{g.x, g.y, g.z}	float	12	27 ac e2 3c	0.02767 rad/s
			a2 7d 0b 3c	0.008514 rad/s
			d2 70 3b bd	-0.045762 rad/s
{acc.x, acc.y, acc.z}	float	12	cf a5 80 3e	0.251265 m/s^2
			3e fc 1b 41	9.74908 m/s^2
			1f a1 26 bd	-0.040681 m/s^2
reserved	float	12	...	*
{angle.x, angle.y, angle.z}*100	int16	6	71 38	144.49 °
			f5 25	97.17 °
			44 fa	-14.68 °
{q0, q1, q2, q3}	float	16	8a 22 28 bf	-0.656777
			5a b7 00 be	-0.125699
			20 4f 3d bf	-0.739489
			1c 0b 52 3d	0.0512801
reserved	*	4	...	*
local_time	uint32	4	0c ae 00 00	44556 ms
system_time	uint32	4	cb 17 01 00	71627 ms
reserved	*	1	...	*
{eop.x, eop.y, eop.z} * 100	uint8	3	0b	0.11 m
			10	0.16 m
			ff	2.55 m
supply_voltage * 1000	uint16	2	54 13	4.948 V
reserved	*	5	...	*
Sum Check	uint8	1	fd	0xfd

Figure 7.6.2 LTS Transmission Protocol

7.7 Appendix Ⅶ Arduino codes of tests

7.7.1 Code for test 7.1.1 and 7.1.2

```
#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#ifndef DEGTORAD
#define DEGTORAD 0.0174532925199432957f
#define RADTODEG 57.295779513082320876f
#endif

SoftwareSerial serial = SoftwareSerial(10,11);


const unsigned char UBX_HEADER[] = { 0xB5, 0x62 };
float z;
float originlat[10];
float originlon[10];
float currentlat[10];
float currentlon[10];
int switchstate1 = 0;
int switchstate2 = 0;
struct NAV_POSLLH {
  unsigned char cls;
  unsigned char id;
  unsigned short len;
  unsigned long iTOW;
  long lon;
  long lat;
  long height;
  long hMSL;
  unsigned long hAcc;
  unsigned long vAcc;
};

NAV_POSLLH posllh;

struct geoloc {
    double lat;
    double lon;
    geoloc(double x = 0, double y = 0, int32_t a = 0) {

         lat = x;
         lon = y;
     }
};
```

```
geoloc originLocation;
geoloc currentLocation;


void calcChecksum(unsigned char* CK)
{
  memset(CK, 0, 2);
  for (int i = 0; i < (int)sizeof(NAV_POSLLH); i++)
  {
    CK[0] += ((unsigned char*)(&posllh))[i];
    CK[1] += CK[0];
  }
}

bool processGPS() {
  static int fpos = 0;
  static unsigned char checksum[2];
  const int payloadSize = sizeof(NAV_POSLLH);

  while ( serial.available() ) {
    byte c = serial.read();
    if ( fpos < 2 )
    {
      if ( c == UBX_HEADER[fpos] )
        fpos++;
      else
        fpos = 0;
    }
    else
    {
      if ( (fpos-2) < payloadSize )
        ((unsigned char*)(&posllh))[fpos-2] = c;

      fpos++;

      if ( fpos == (payloadSize+2) )
      {
        calcChecksum(checksum);
      }
      else if ( fpos == (payloadSize+3) )
      {
        if ( c != checksum[0] )
          fpos = 0;
      }
      else if ( fpos == (payloadSize+4) )
      {
        fpos = 0;
        if ( c == checksum[1] )
        {
          delay(2000);
          return true;
        }
        }
        else if ( fpos > (payloadSize+4) )
        {
          fpos = 0;
        }
      }
    }
    return false;
  }


  void setup()
  {
    Serial.begin(19200);
    serial.begin(19200);
    pinMode(6,INPUT);
    pinMode(7,INPUT);
  }

  void loop()
  {

      // Read the status of the switches
      switchstate1 = digitalRead(6);
      switchstate2 = digitalRead(7);


      if (processGPS())
        if(switchstate1 == HIGH)
        {
          delay(1000);
          if(switchstate1 == HIGH)
          {
            {

              originLocation.lat = 0;
              originLocation.lon = 0;
              for(int i=0;i<10;i++)
              {
                processGPS();
                delay(600);
                originlat[i] = posllh.lat * 0.0000001;
                originLocation.lat = originLocation.lat + originlat[i];
                originlon[i] = posllh.lon * 0.0000001;
                originLocation.lon = originLocation.lon + originlon[i];
                Serial.print(posllh.lat,10);
                Serial.println();
                Serial.print(posllh.lon,10);
                Serial.println();

              }
            originLocation.lat = originLocation.lat / 10;
            originLocation.lon = originLocation.lon / 10;
            Serial.print(" lat/lon: "); Serial.print(originLocation.lat,10);
            Serial.print(","); Serial.print(originLocation.lon,10);
          }
        }
      }

      if(switchstate2 == HIGH)
    {
       delay(1000);
        if(switchstate2 == HIGH)
        {
          currentLocation.lat = 0;
          currentLocation.lon = 0;

          for(int j=0;j<10;j++)
          {
            processGPS();
            delay(600);
            currentlat[j] = posllh.lat * 0.0000001;
            currentLocation.lat = currentLocation.lat + currentlat[j];
            currentlon[j] = posllh.lon * 0.0000001;
            currentLocation.lon = currentLocation.lon + currentlon[j];
            Serial.print(posllh.lat,10);
            Serial.println();
            Serial.print(posllh.lon,10);
            Serial.println();
          }
        }
          currentLocation.lat = currentLocation.lat / 10;
          currentLocation.lon = currentLocation.lon / 10;

          z = geoDistance(currentLocation, originLocation); //meters
          Serial.println(" lat2/lon2: "); Serial.print(posllh.lat);
          Serial.print(","); Serial.print(posllh.lon);
          Serial.print(" D: ");      Serial.print(z, 10);
          Serial.println();
    }
}

float geoDistance(struct geoloc &a, struct geoloc &b)   // returns meters
{
    const double R = 6371000; // km
    double p1 = a.lat * DEGTORAD;
    double p2 = b.lat * DEGTORAD;
    double dp = (b.lat-a.lat) * DEGTORAD;
    double dl = (b.lon-a.lon) * DEGTORAD;
    double x = sin(dp/2) * sin(dp/2) + cos(p1) * cos(p2) * sin(dl/2) * sin(dl/2);
    double y = 2 * atan2(sqrt(x), sqrt(1-x));
    return y*R;
}
```

7.7.2 Code for test 7.2.2

```
#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "math.h"

SoftwareSerial serialLTS = SoftwareSerial(12,13);

#define OLED_RESET 4
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

typedef struct
{
  uint8_t BYTE[3];
}int24_t;

struct TAG_FRAME0
{
  int24_t pos[3];
  int24_t vel[3];
  int24_t dis[8];
  float g[3];
  float acc[3];
  float res1[3];

  int16_t ang[3];
  float q[4];
  uint32_t res2;

  uint32_t local_time;
  uint32_t system_time;
  uint8_t res3;

  uint8_t eop[3];
  uint16_t voltage;
  uint8_t res4[5];
};
TAG_FRAME0 tf0;
const unsigned char LTS_HEADER[] = { 0x55, 0x01, 0x00, 0x02};
float coordinatex[11], coordinatey[11], coordinate[2], distance;

void setup()
{
  Serial.begin(115200);
  serialLTS.begin(115200);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C))// Address 0x3C for 128x32
  {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  display.clearDisplay();
  display.display();
}

void loop()
{
    static int timesforvariance = 0;

    if (processLTS())
    {
      screen_display();
      int calibration = 0 ;
      calibration = coordinate_cal(timesforvariance);

      timesforvariance++;
      timesforvariance = timesforvariance - calibration;

      if (timesforvariance == 5)
      {
        variance_cal();
        Serial.print("Coo:");Serial.print(coordinate[0]);Serial.print(", ");
        Serial.print(coordinate[1]);Serial.println();
        timesforvariance = 0;
       /*Serial.print("lx");Serial.print(lx);Serial.print(" ");
       Serial.print("ly");Serial.print(ly);Serial.println();*/
       /*Serial.print(direction_angle);Serial.println();*/
      }

    }
}

bool processLTS()
{
  static int fpos = 0;
  const int payloadSize = sizeof(tf0);
  while ( serialLTS.available() )
  {
    byte c = serialLTS.read();
    if ( fpos < 4 )
    {
      if ( c == LTS_HEADER[fpos] )
      {
        fpos++;
      }
      else
        fpos = 0;
    }
    else
    {
      if ( fpos >= 4 )
      {
        ((unsigned char*)(&tf0))[fpos-4] = c;
        /*Serial.print("data");Serial.print(fpos-3);Serial.print(": ");
        Serial.print(((unsigned char*)(&tf0))[fpos-4], HEX);Serial.println();*/
        fpos++;
        if ( fpos - 4 >= payloadSize )
        {
          fpos = 0;
          /*Serial.print(tf0.pos[0].BYTE[0], HEX);Serial.print(" ");
          Serial.print(tf0.pos[0].BYTE[1], HEX);Serial.print(" ");
          Serial.print(tf0.pos[0].BYTE[2], HEX);Serial.print(" ");*/
          return true;
        }
      }
    }
    return false;
  }
}

int coordinate_cal(int a)
{
  uint8_t cx[3], cy[3], cz[3];
  int32_t cxx, cyy, czz;
  float px, py, pz;
  for(int i = 0; i < 3; i++)
  {
    cx[i] = tf0.pos[0].BYTE[i];
    cy[i] = tf0.pos[1].BYTE[i];
    cz[i] = tf0.pos[2].BYTE[i];
  }

  cxx = (((cx[0] << 8) & 0xffff) | (cx[1] * 0x10000) | (cx[2] * 0x1000000)) / 0x100;
  cyy = (((cy[0] << 8) & 0xffff) | (cy[1] * 0x10000) | (cy[2] * 0x1000000)) / 0x100;
  czz = (((cz[0] << 8) & 0xffff) | (cz[1] * 0x10000) | (cz[2] * 0x1000000)) / 0x100;

  px = cxx / 1000.0f;
  py = cyy / 1000.0f;
  pz = czz / 1000.0f;

  Serial.print(cx[0], HEX);Serial.print(" ");Serial.print(cx[1], HEX);
  Serial.print(" ");Serial.print(cx[2], HEX);Serial.print(" ");Serial.println();
  Serial.print(cy[0], HEX);Serial.print(" ");Serial.print(cy[1], HEX);
  Serial.print(" ");Serial.print(cy[2], HEX);Serial.print(" ");Serial.println();
  Serial.print(cz[0], HEX);Serial.print(" ");Serial.print(cz[1], HEX);
  Serial.print(" ");Serial.print(cz[2], HEX);Serial.print(" ");Serial.println();
  Serial.print(px);Serial.print(", ");Serial.print(py);Serial.print(", ");
  Serial.print(pz);Serial.println();

  if (px > 80 || py > 80 || px < -80 || py < -80)
  {
    return 1;
  }
  else
  {
    coordinatex[a] = px;
    coordinatey[a] = py;
    return 0;
  }
}

void variance_cal()
{
  float coo[2];
  float variancex = 0;
  float variancey = 0;
  float sumx = 0;
  float sumy = 0;

  for (int a = 0; a < 5 ; a++)
  {
    sumx += coordinatex[a];
    sumy += coordinatey[a];
  }
  coo[0] = sumx / 5;
  coo[1] = sumy / 5;

  for (int j = 0; j < 5; j++)
  {
    variancex += pow(coordinatex[j] - coo[0], 2) / 5;
    variancey += pow(coordinatey[j] - coo[1], 2) / 5;
  }
  /*Serial.print("v: ");Serial.print(variancex, 4);Serial.print(", ");
  Serial.print(variancey, 4);Serial.println();*/
  if (variancex < 0.002 && variancey < 0.002)
  {
    coordinate[0] = coo[0];
    coordinate[1] = coo[1];
  }
  coordinate[0] = coo[0];
  coordinate[1] = coo[1];
}

void screen_display()
{
    display.print("lat: "); display.print(coordinate[0]); display.print( '\n');
    display.print("lon: "); display.print(coordinate[1]); display.print('\n');
}
```

7.7.3 Code for test 7.3.1 and 7.3.2

```
#define pinA 2
#define pinB 3

int StateLEFT, LastStateLEFT, StateRIGHT, LastStateRIGHT;

void setup()
{
   pinMode (2, INPUT);
   pinMode (3, INPUT);
   Serial.begin (115200);

   StateLEFT = digitalRead(pinA);
   StateRIGHT = digitalRead(pinA);
   attachInterrupt(0, distance_calLEFT, CHANGE);
   attachInterrupt(1, distance_calRIGHT, CHANGE);

 }

void loop()
{

}

void distance_calLEFT()
{
  static long counter = 0;
  float meter = 0;
  float moving_distance = 0;
  float distick = 10.4772115;
  float dismov = 0;

  StateLEFT = digitalRead(pinA);

  if (StateLEFT != LastStateLEFT)
  {
    counter ++;
  }
  else
  {
    counter = counter;
  }

  dismov = counter * distick /10 /2;

  /*Serial.print("Ticks= ");Serial.print(counter);Serial.print(" | ");
  Serial.print("Distance moved= ");Serial.print(dismov);Serial.println(" cm");*/

  LastStateLEFT = StateLEFT;
}

void distance_calRIGHT()
{
  static long counter = 0;
  float meter = 0;
  float moving_distance = 0;
  float distick = 10.4772115;
  float dismov = 0;

  StateRIGHT = digitalRead(pinB);

  if (StateRIGHT != LastStateRIGHT)
  {
    counter ++;
  }
  else
  {
    counter = counter;
  }

  dismov = counter * distick /10 /2;

  Serial.print("Ticksr= ");Serial.print(counter);Serial.print(" | ");
  Serial.print("Distance movedright= ");Serial.print(dismov);Serial.println(" cm");

  LastStateRIGHT = StateRIGHT;
}
```