A PROJECT PRESENTATION

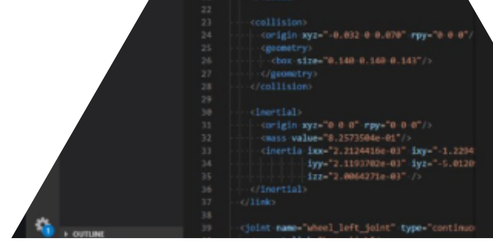
The Autonomous Vehicle: GNSS Technology Tracking Using Controller

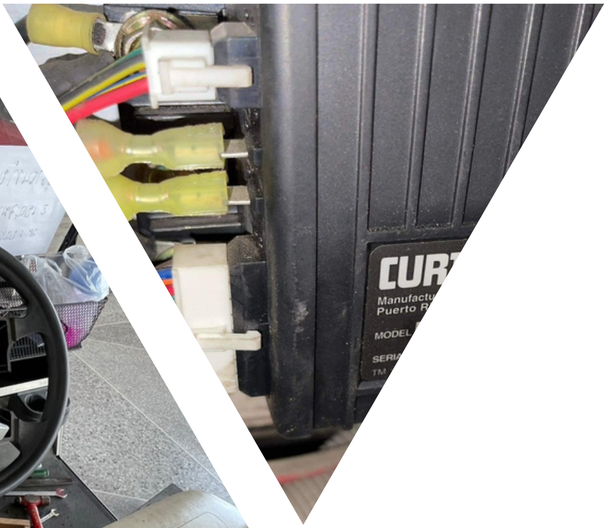
Department of Control System and Instrumentation Engineering
Faculty of Engineering

Presented by
Surayuth Duenkwang 62070501223
Chutimon Ketkarn 62070501236
Sorravee Wanichphol 62070501241

KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI

Introduction and Background

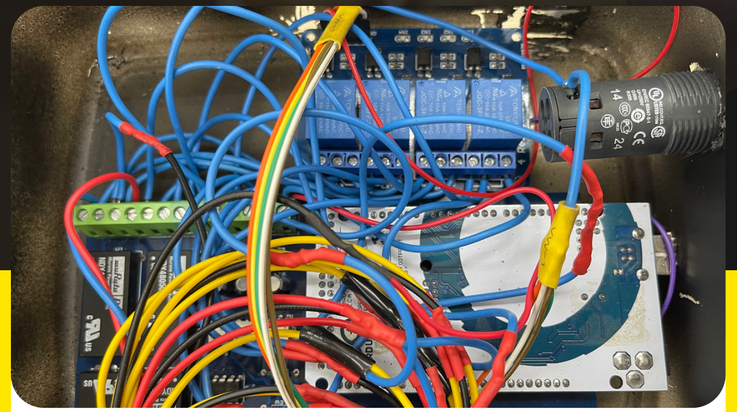




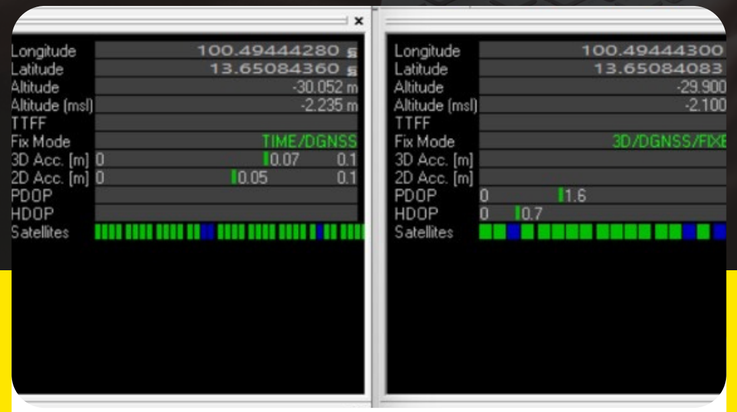
- **Eco-Friendly and save more cost**
- **Autonomous Vehicle Control System**
- **Autonomous Vehicle Tracking System**

The Autonomous Vehicle: GNSS Technology Tracking Using Controller

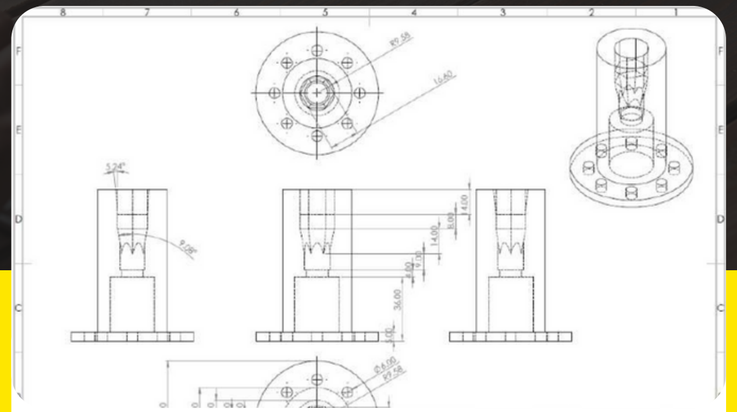
Objective



Improved and Design
Working principle



Design Navigation System
GNSS

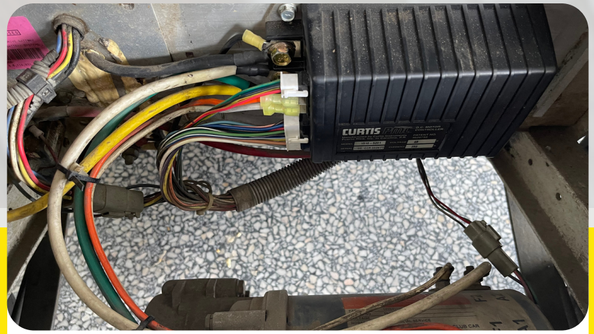


Study and Design
ROS, Speed and Steering Control

Scope of Work

Position Tracking

Accuracy in cm. (less than m.)



New Hardware design

Wiring and Speed Control

Steering part

Can bus, Design New connector and Steering Motor Control

Robot Operating System

Implement to ROS Node

Tracking and Navigation

Global Navigation Satellite System (GNSS)

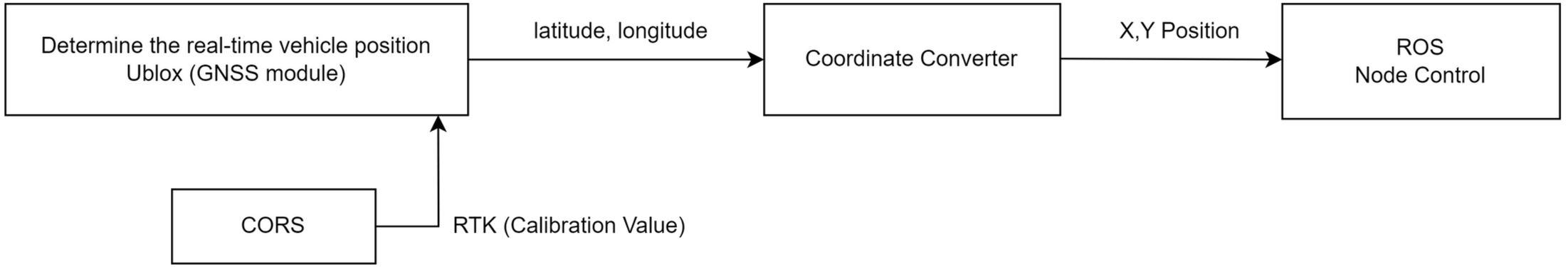
The Autonomous Vehicle: GNSS Technology Tracking Using Controller

Content

- **Measurement Quality**
- **Coordinate converter**

Block Diagram

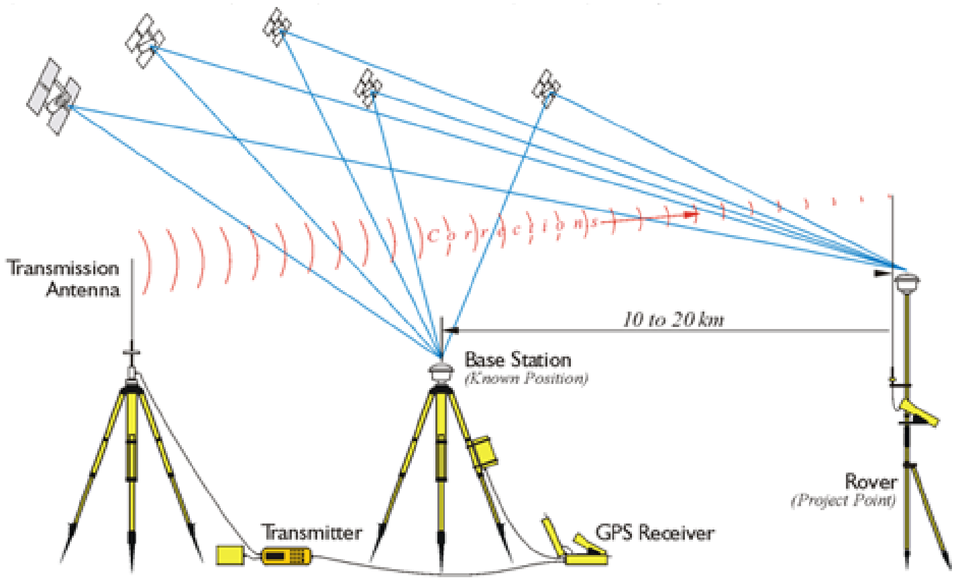
Tracking and Navigation System



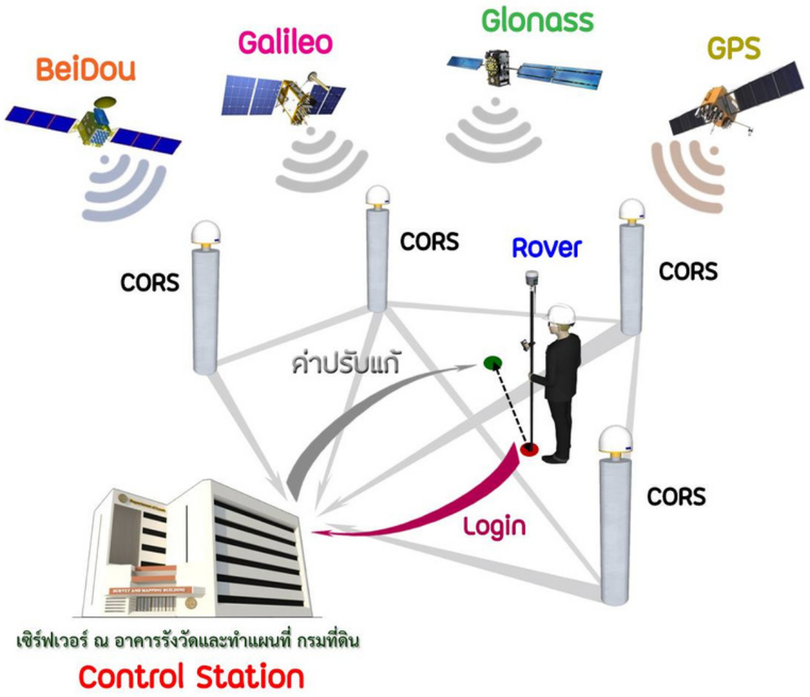
Determine the real-time vehicle position

GNSS Method Surveying

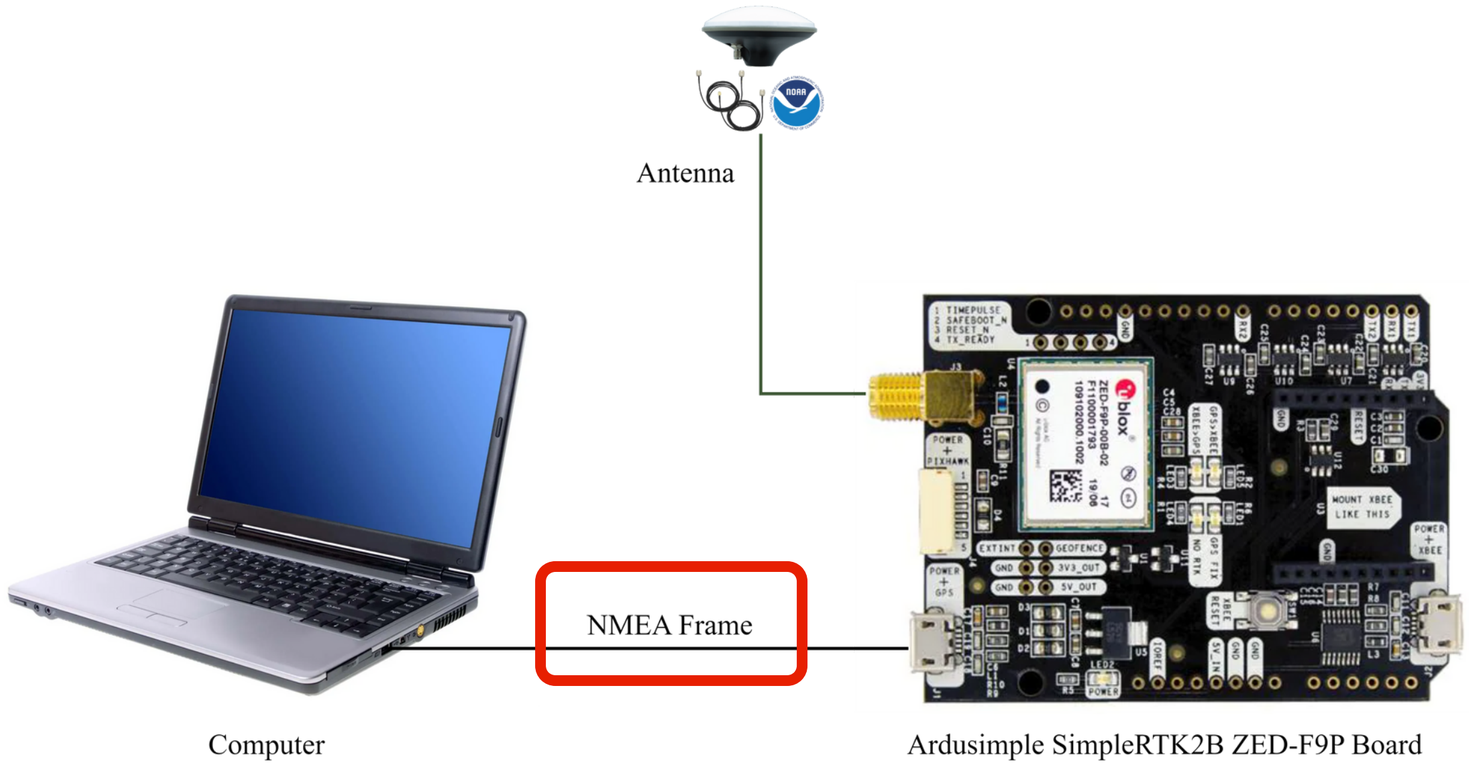
RTK (Real Time Kinematic)



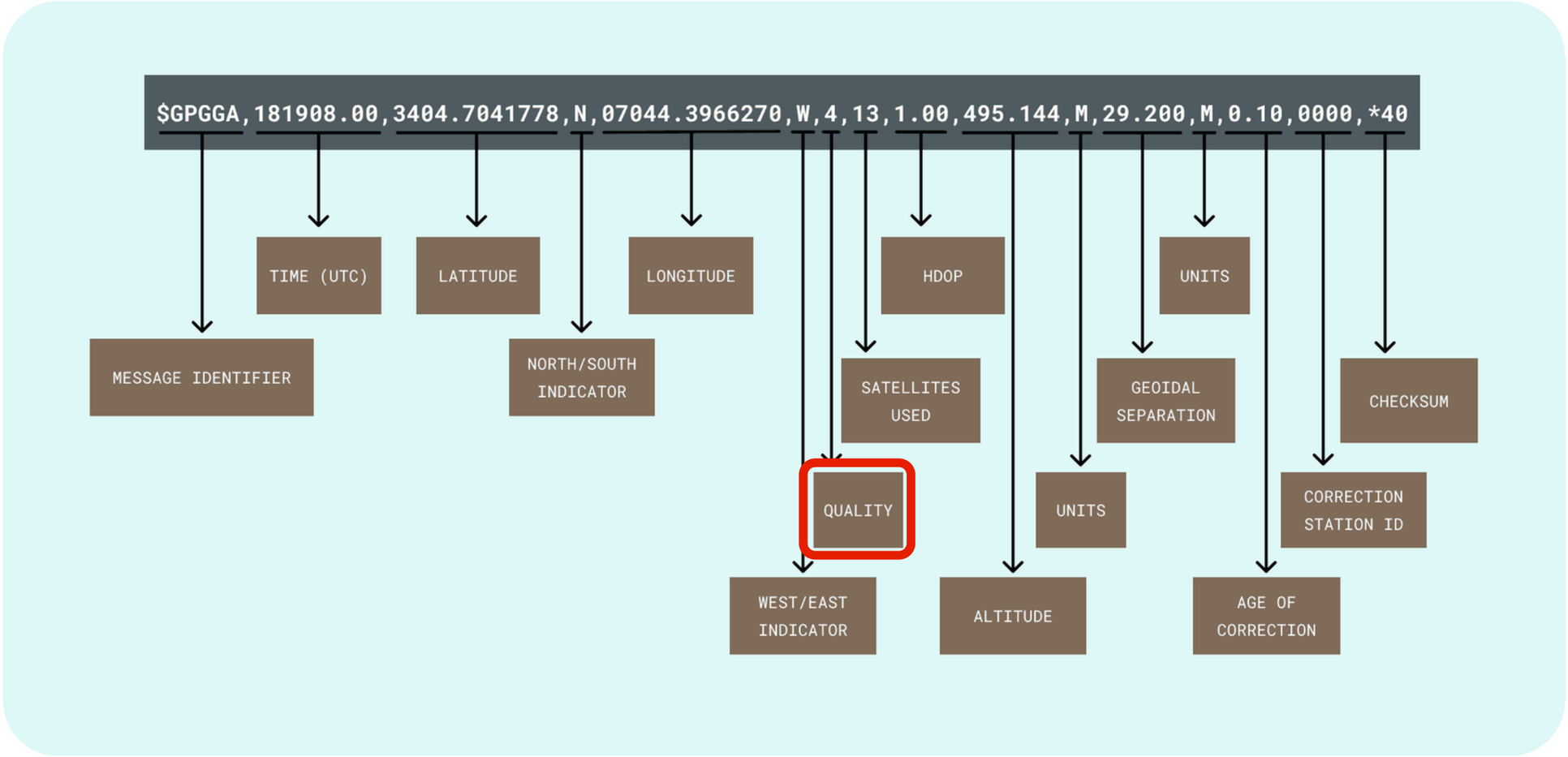
Network RTK



Measurement Quality



NMEA GGA Frame



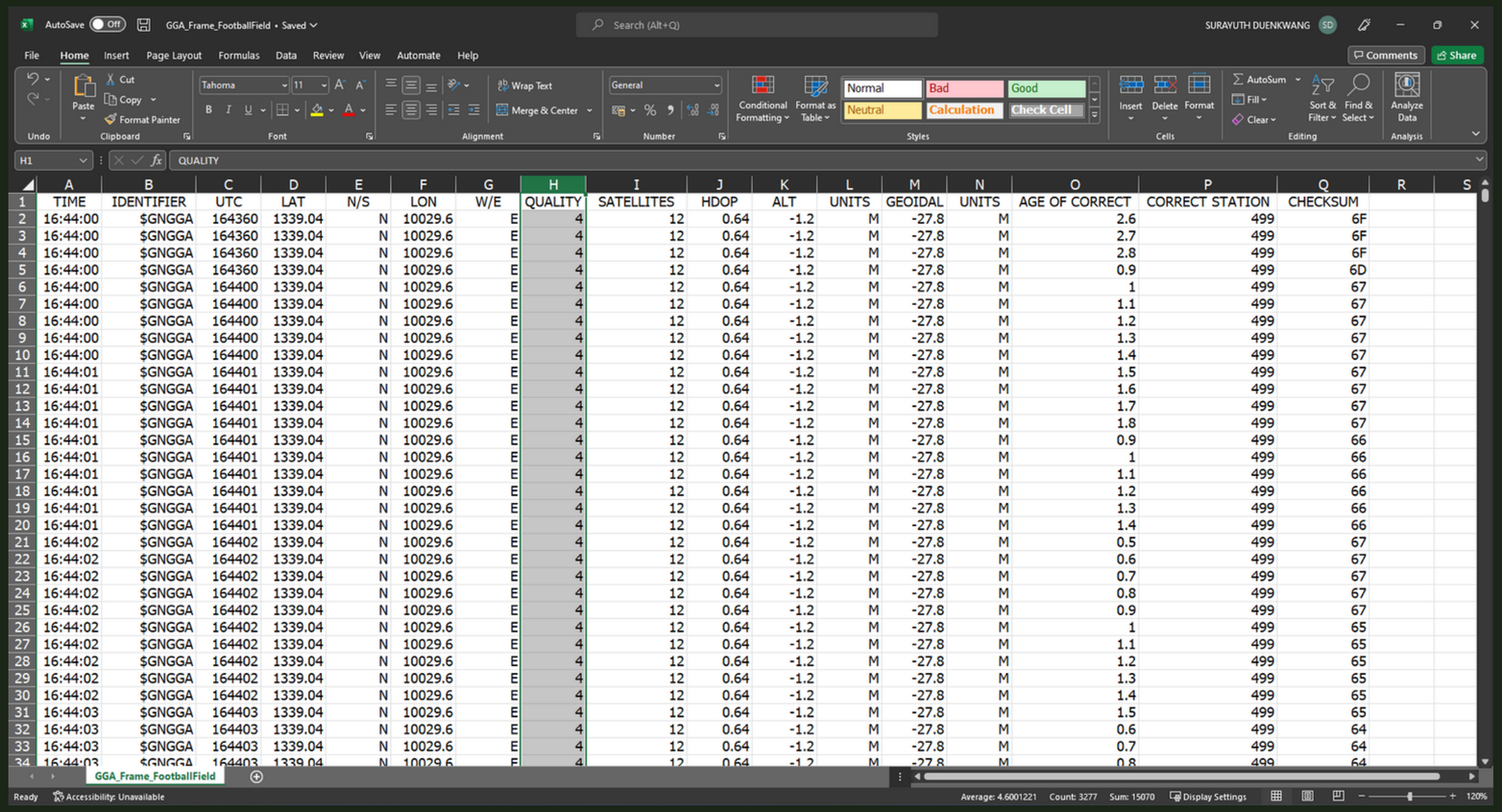
Quality Type

Value	Name	Description
0	Invalid	GNSS can't determinate the position (No signal)
1	GPS	Position is gathered only form GPS satellites.
2	DGNSS	Position is gathered from GNSS satellites. The DGNSS accuracy is in the range of 30-50 cm.
4	RTK fixed	Position is gathered and calculated between the satellites and the reference station. The RTK fixed accuracy is in the range of 1-5 cm.
5	RTK float	Very similar to the fixed RTK method. But the calculation has not been solved yet. The RTK float accuracy is in the range of 5-30 cm.

Count The Quality Type

Example Result from Python

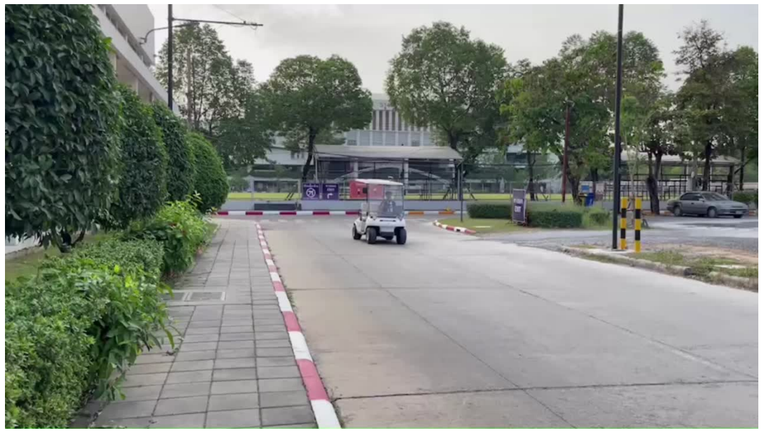
```
Sample =  3583
RTK FIXED Quality =  1727  Sample
RTK FLOAT Quality =  1803  Sample
RTK DGNSS Quality =  53  Sample
Total Quality =  3583  Sample
RTK FIXED Quality =  48.2 %
RTK float Quality =  50.32 %
RTK DGNSS Quality =  1.48 %
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	TIME	IDENTIFIER	UTC	LAT	N/S	LON	W/E	QUALITY	SATELLITES	HDOP	ALT	UNITS	GEOIDAL	UNITS	AGE OF CORRECT	CORRECT STATION	CHECKSUM		
2	16:44:00	$GNGGA	164360	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	2.6	499	6F		
3	16:44:00	$GNGGA	164360	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	2.7	499	6F		
4	16:44:00	$GNGGA	164360	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	2.8	499	6F		
5	16:44:00	$GNGGA	164360	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.9	499	6D		
6	16:44:00	$GNGGA	164400	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1	499	67		
7	16:44:00	$GNGGA	164400	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.1	499	67		
8	16:44:00	$GNGGA	164400	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.2	499	67		
9	16:44:00	$GNGGA	164400	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.3	499	67		
10	16:44:00	$GNGGA	164400	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.4	499	67		
11	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.5	499	67		
12	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.6	499	67		
13	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.7	499	67		
14	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.8	499	67		
15	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.9	499	66		
16	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1	499	66		
17	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.1	499	66		
18	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.2	499	66		
19	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.3	499	66		
20	16:44:01	$GNGGA	164401	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.4	499	66		
21	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.5	499	67		
22	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.6	499	67		
23	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.7	499	67		
24	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.8	499	67		
25	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.9	499	67		
26	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1	499	65		
27	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.1	499	65		
28	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.2	499	65		
29	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.3	499	65		
30	16:44:02	$GNGGA	164402	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.4	499	65		
31	16:44:03	$GNGGA	164403	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	1.5	499	65		
32	16:44:03	$GNGGA	164403	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.6	499	64		
33	16:44:03	$GNGGA	164403	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.7	499	64		
34	16:44:03	$GNGGA	164403	1339.04	N	10029.6	E	4	12	0.64	-1.2	M	-27.8	M	0.8	499	64		

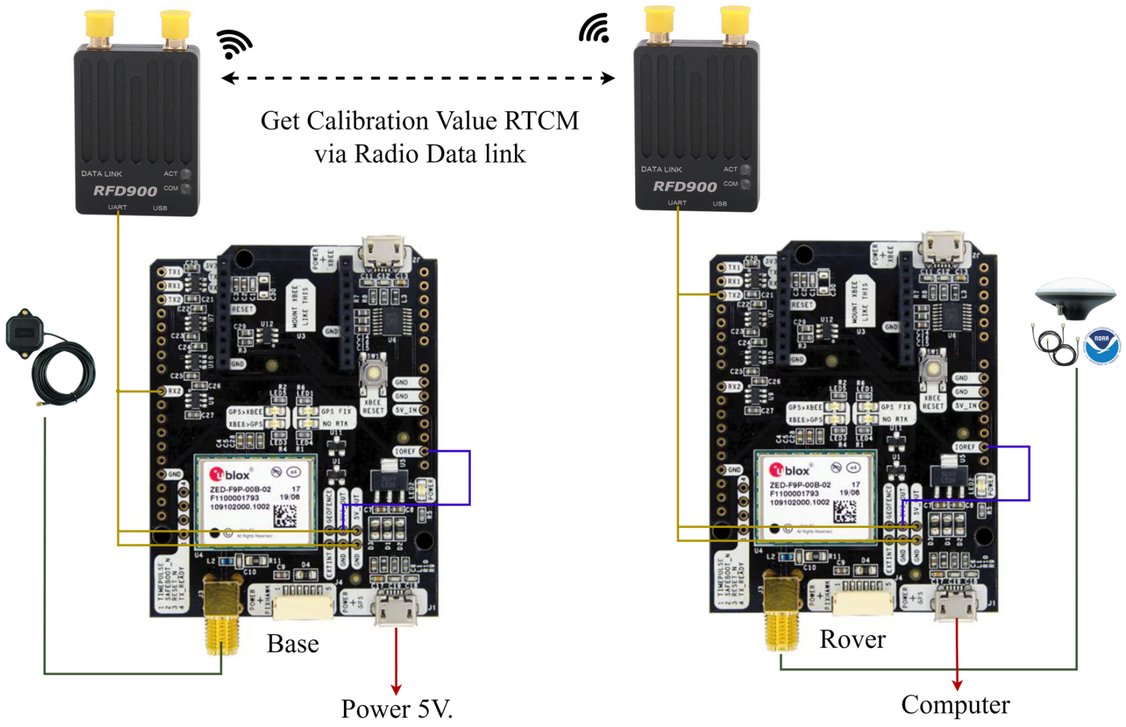
Experiment 1 :

Short Distance



Set-up

Real Time Kinematic Method (2 Boards)

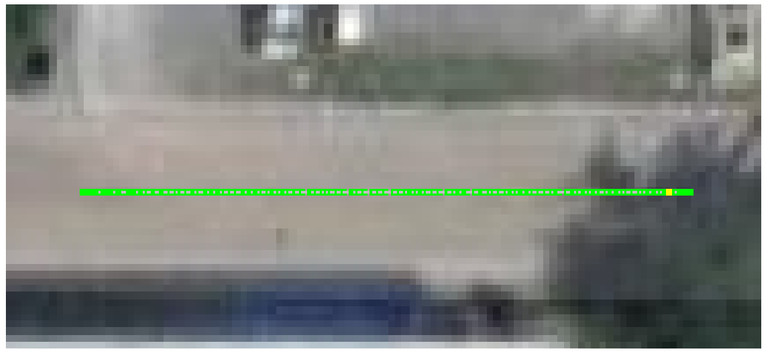


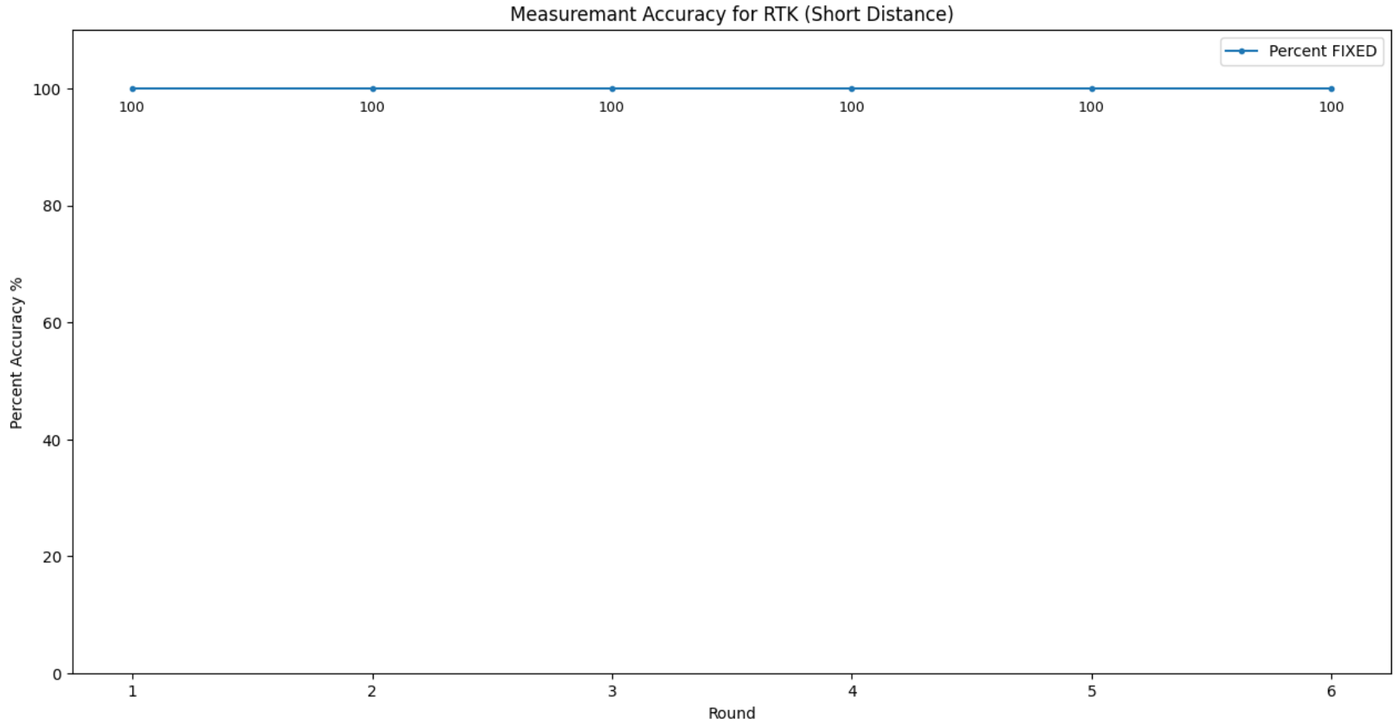
Base Station

Rover Station

Experiment 1 : Short Distance

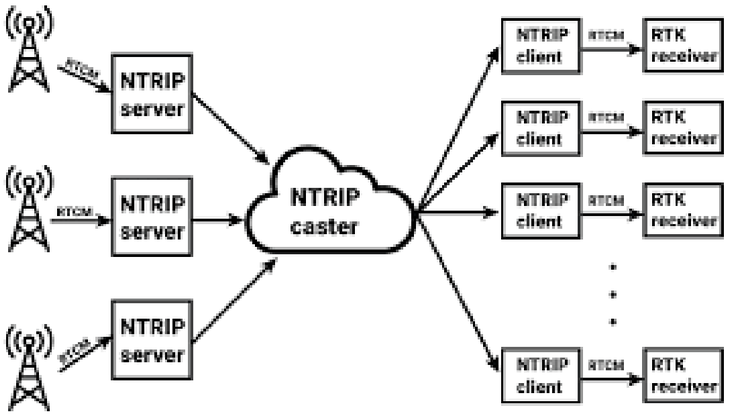
Real Time Kinematic Method (2 Boards)





Set-up

Network RTK (1 Boards)

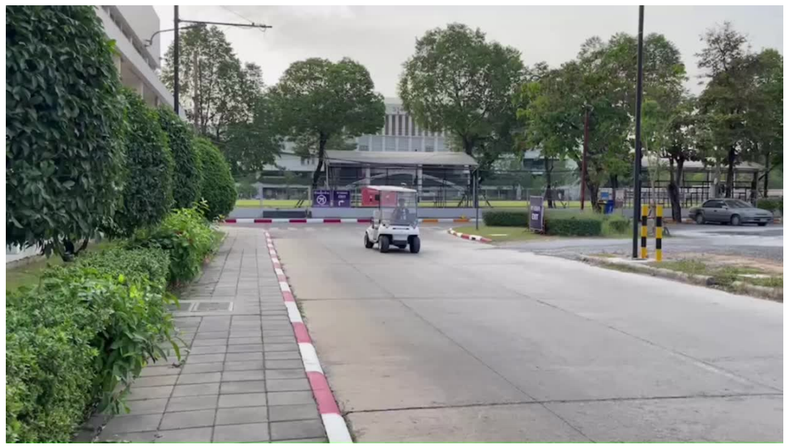


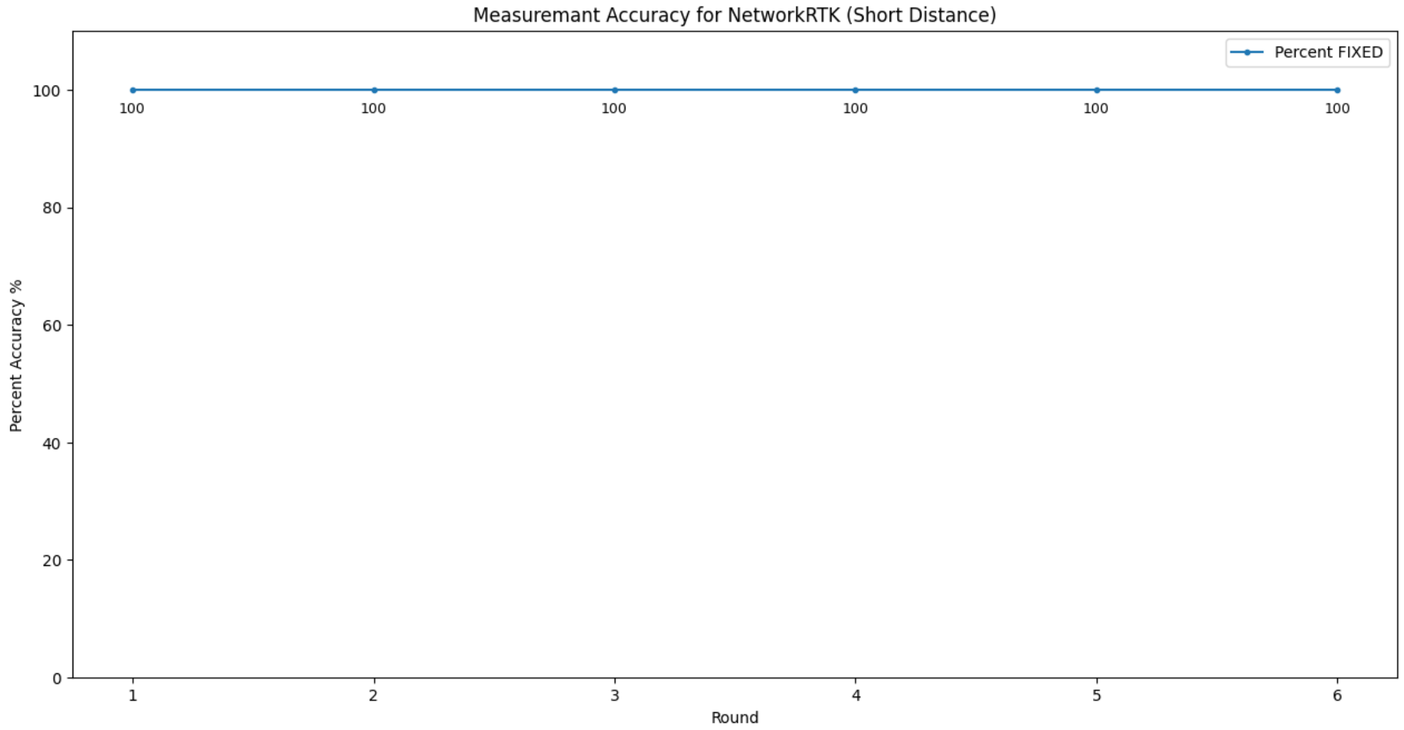
system architecture

Rover Station

Experiment 1 : Short Distance

Network RTK (1 Boards)



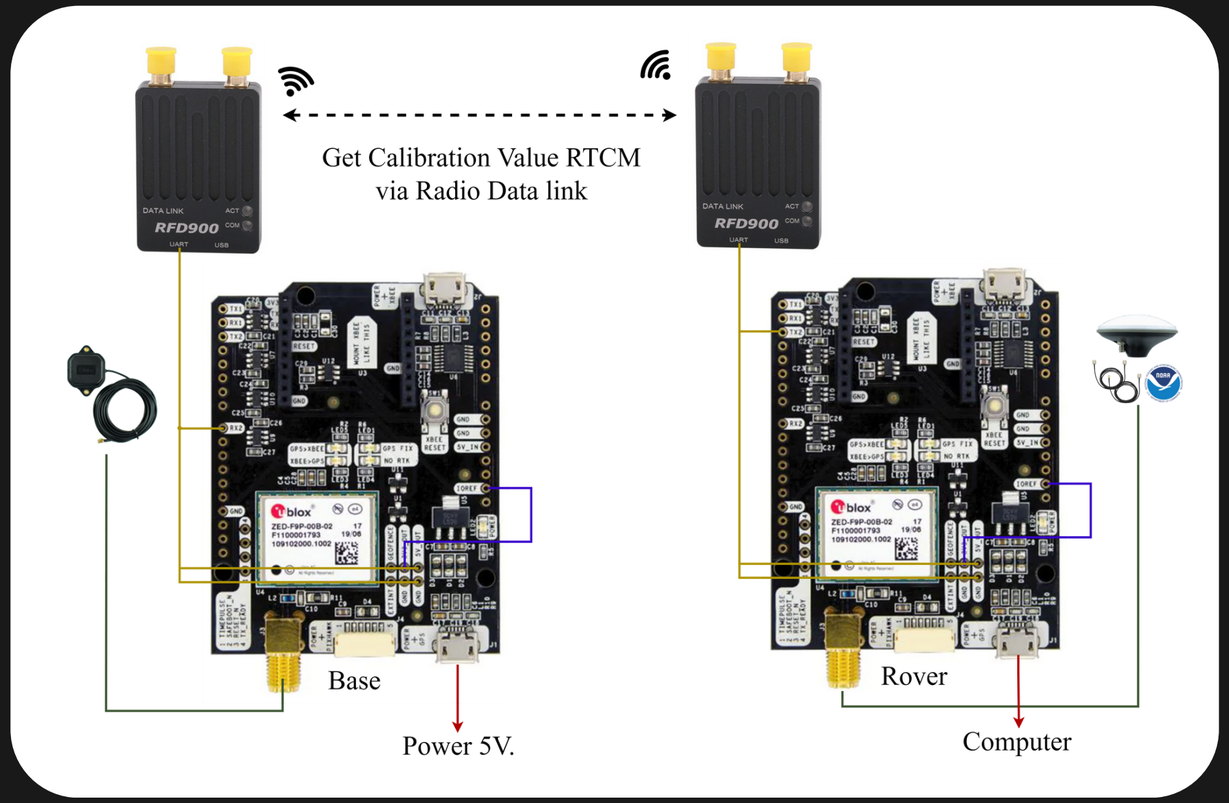


Experiment 2 :

Around KMUTT Soccer Field

Set-up

Real Time Kinematic Method (2 Boards)

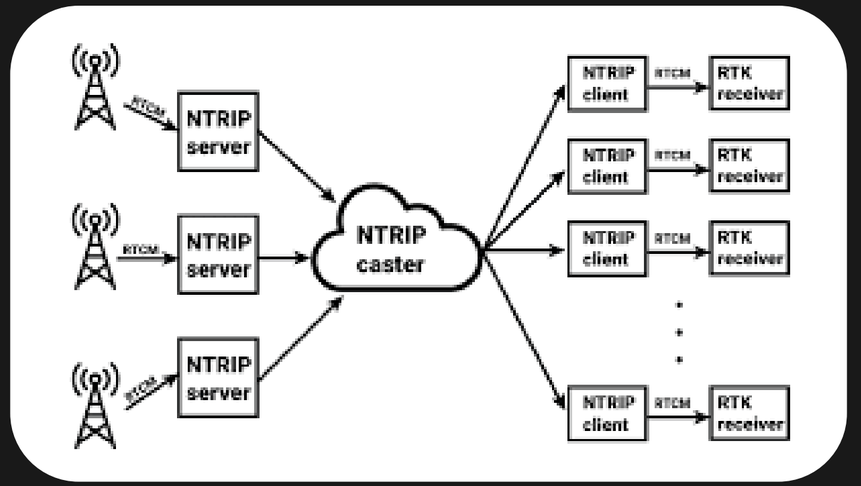


Base Station

Rover Station

Set-up

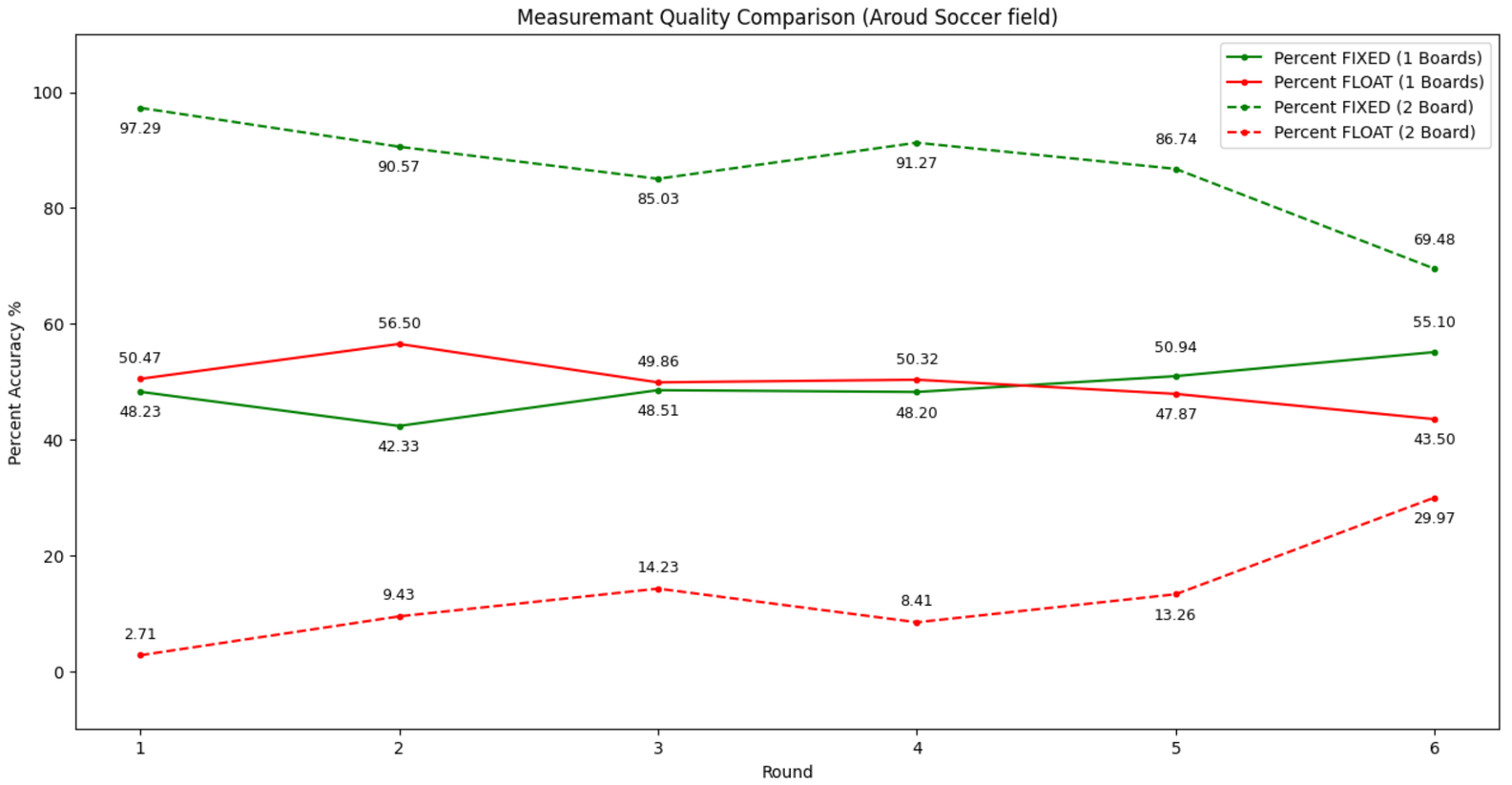
Network RTK (1 Boards)

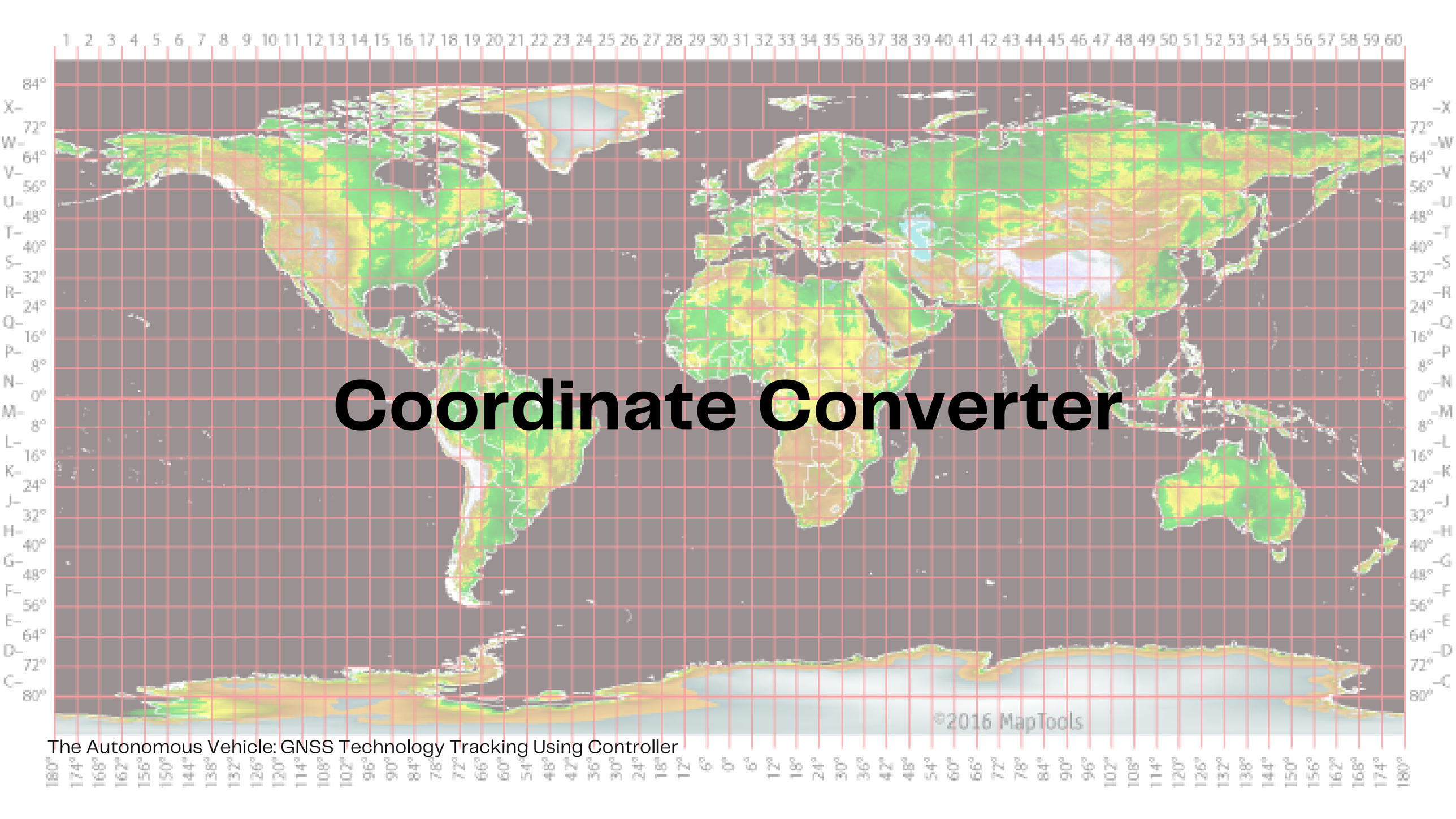


system architecture

Rover Station

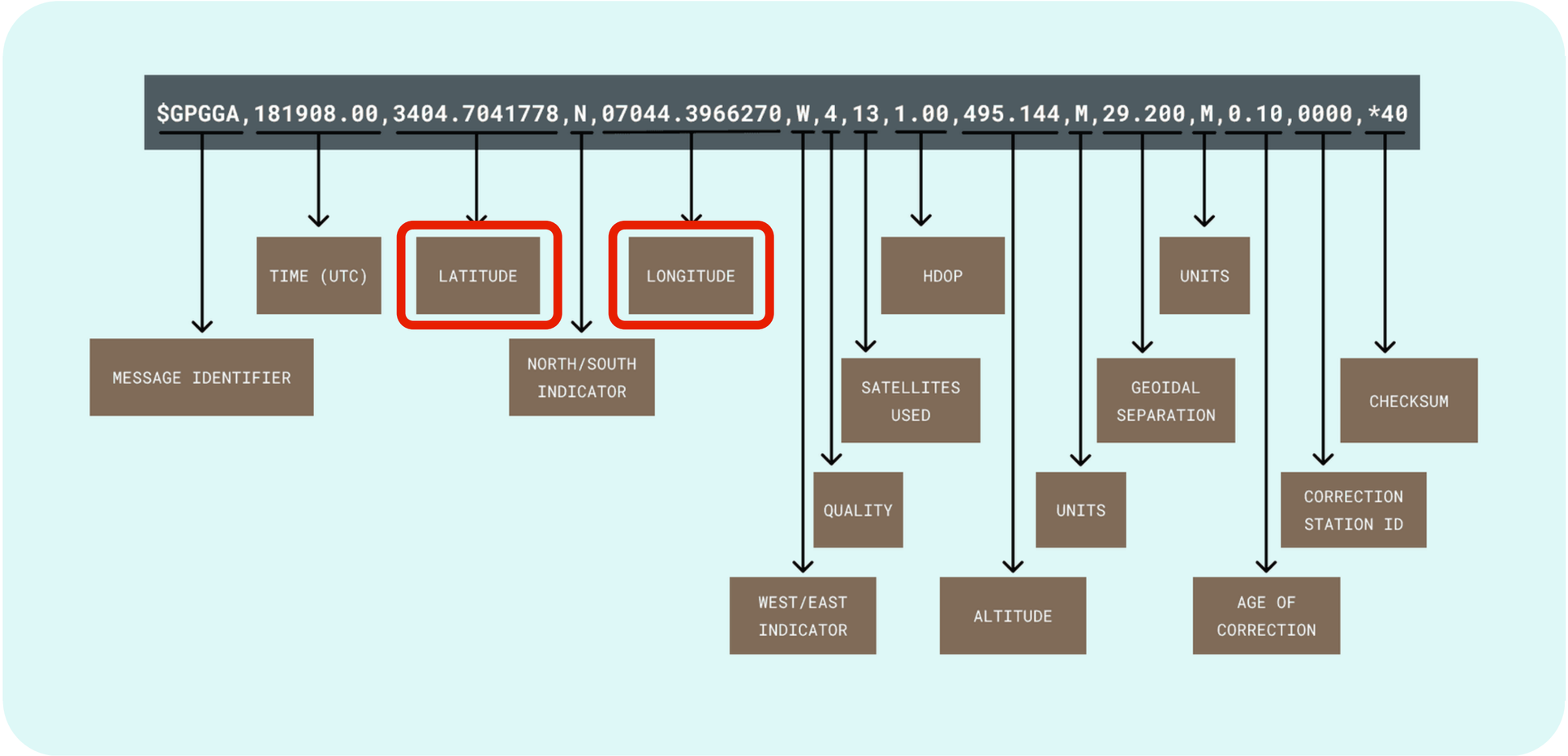
Comparison


Measuremant Quality Comparison (Aroud Soccer field)
Percent FIXED (1 Boards)
Percent FLOAT (1 Boards)
Percent FIXED (2 Board)
Percent FLOAT (2 Board)
Percent Accuracy %
Round
100
80
60
40
20
0
1
2
3
4
5
6
97.29
90.57
85.03
91.27
86.74
69.48
50.47
56.50
49.86
50.32
47.87
43.50
48.23
42.33
48.51
48.20
50.94
55.10
2.71
9.43
14.23
8.41
13.26
29.97

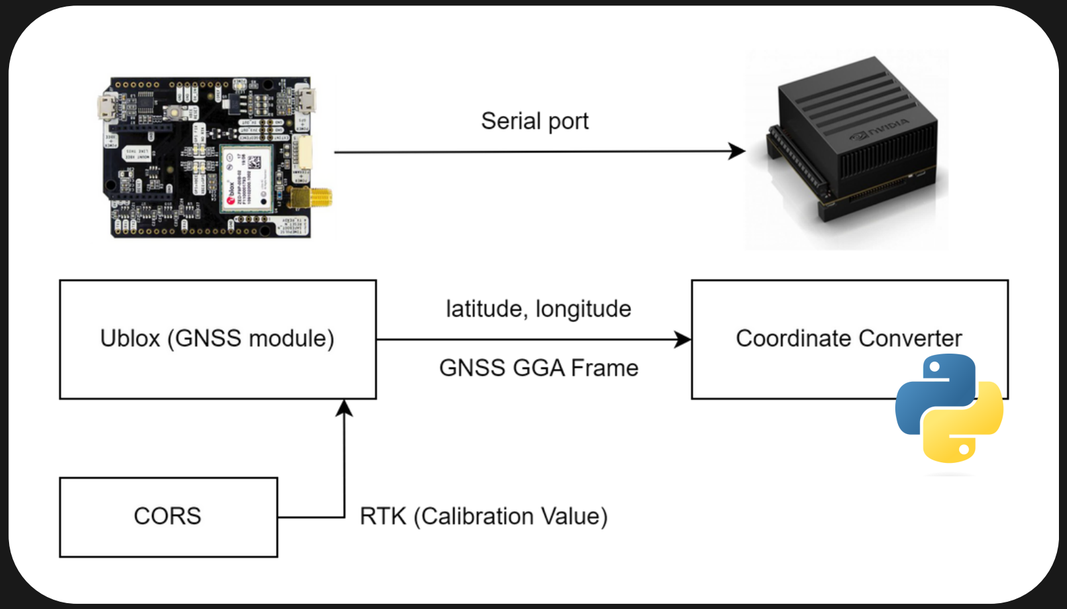


The Autonomous Vehicle: GNSS Technology Tracking Using Controller

NMEA GGA Frame



Coordinate Converter




```
X =  661483.2739867235  Y =  1509509.5167576936
[13.649949, 100.49295966666666]
X =  661483.2738733211  Y =  1509509.5351952738
[13.649949, 100.49295966666666]
[13.649949, 100.49295966666666]
X =  661483.2738733211  Y =  1509509.5351952738
X =  661483.291904169  Y =  1509509.5353061743
```

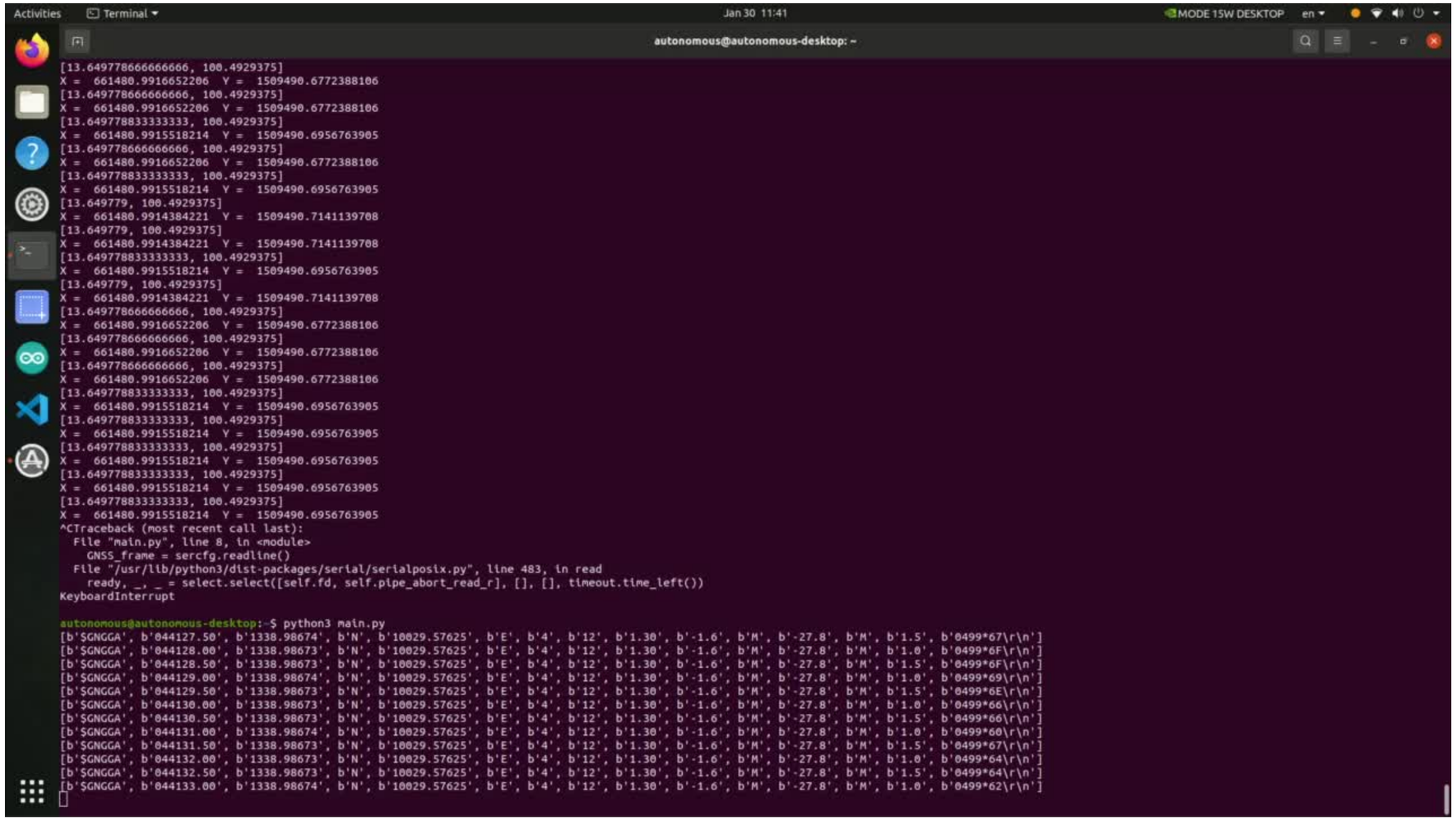
Type the latitude and longitude values to convert into **UTM** (Universal Transverse Mercator) coordinate system.

Latitude	Longitude	
13.649949	100.492960	Convert

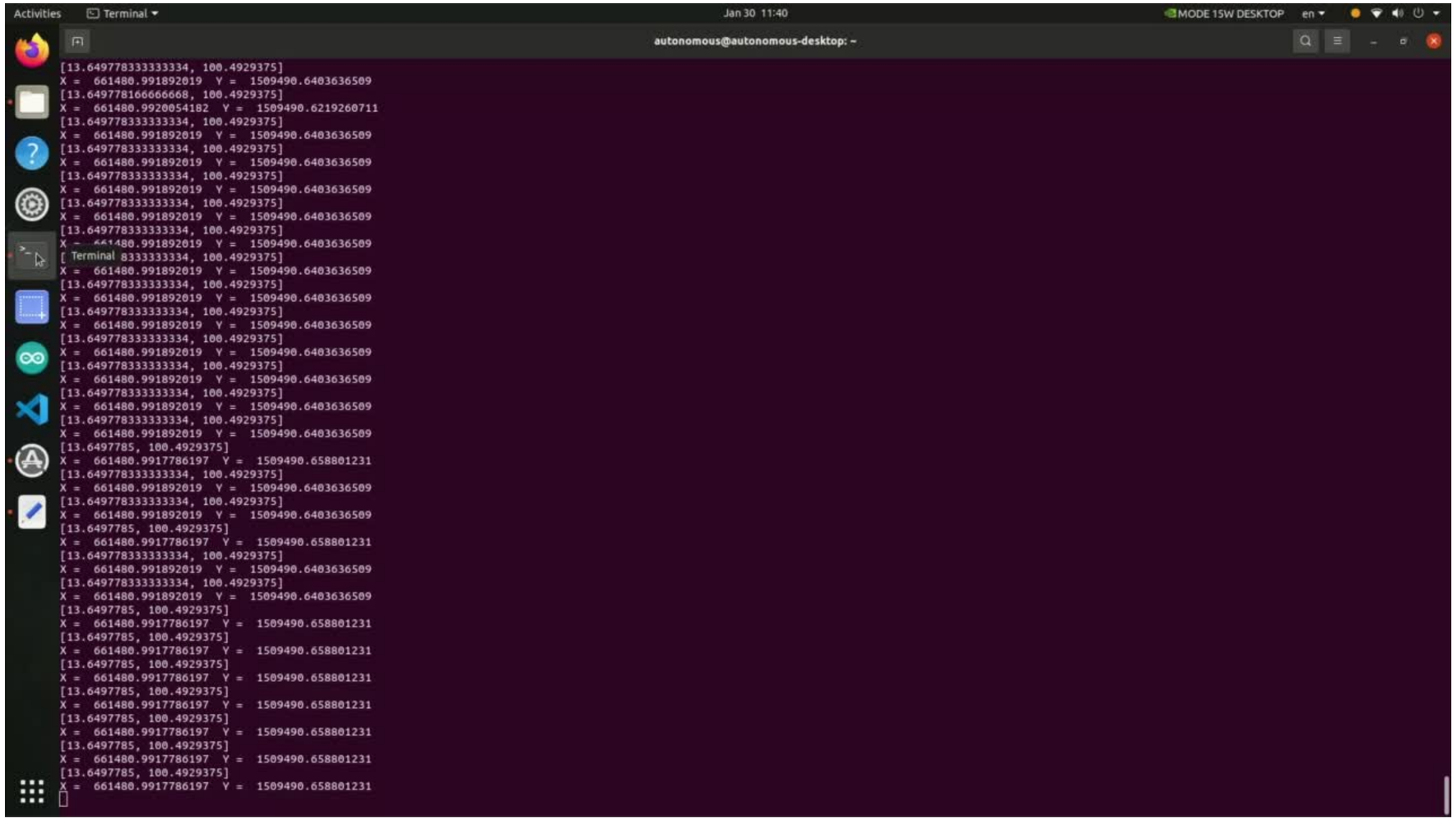
UTM Easting	UTM Northing	UTM Zone
661483.31	1509509.54	47P

The Autonomous Vehicle: GNSS Technology Tracking Using Controller

Send GGA frame to Ubuntu



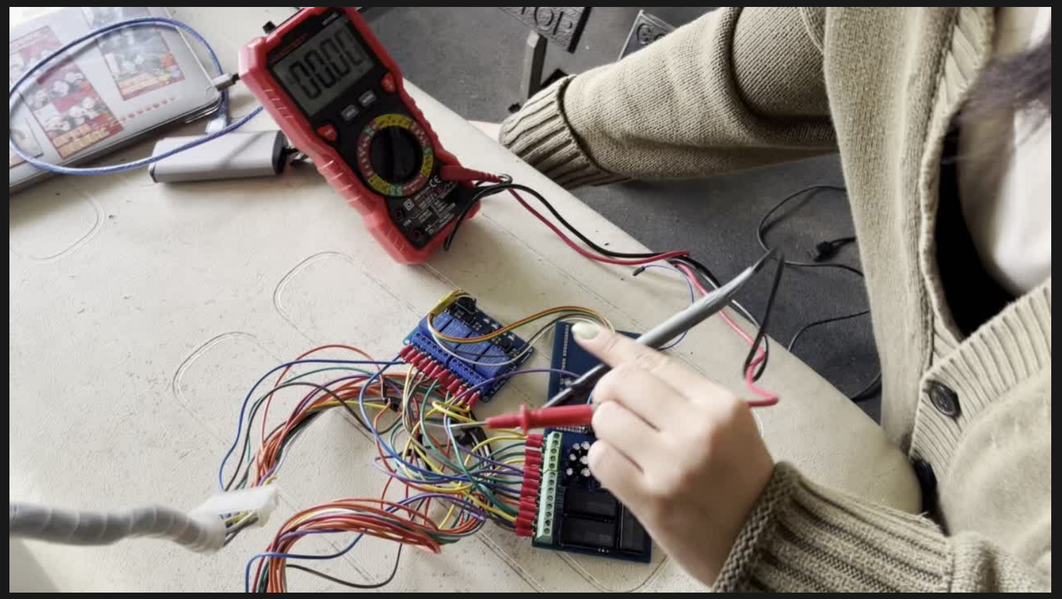
Convert Lat Lon to X Y Coordinate



SPEED CONTROL

PROBLEM

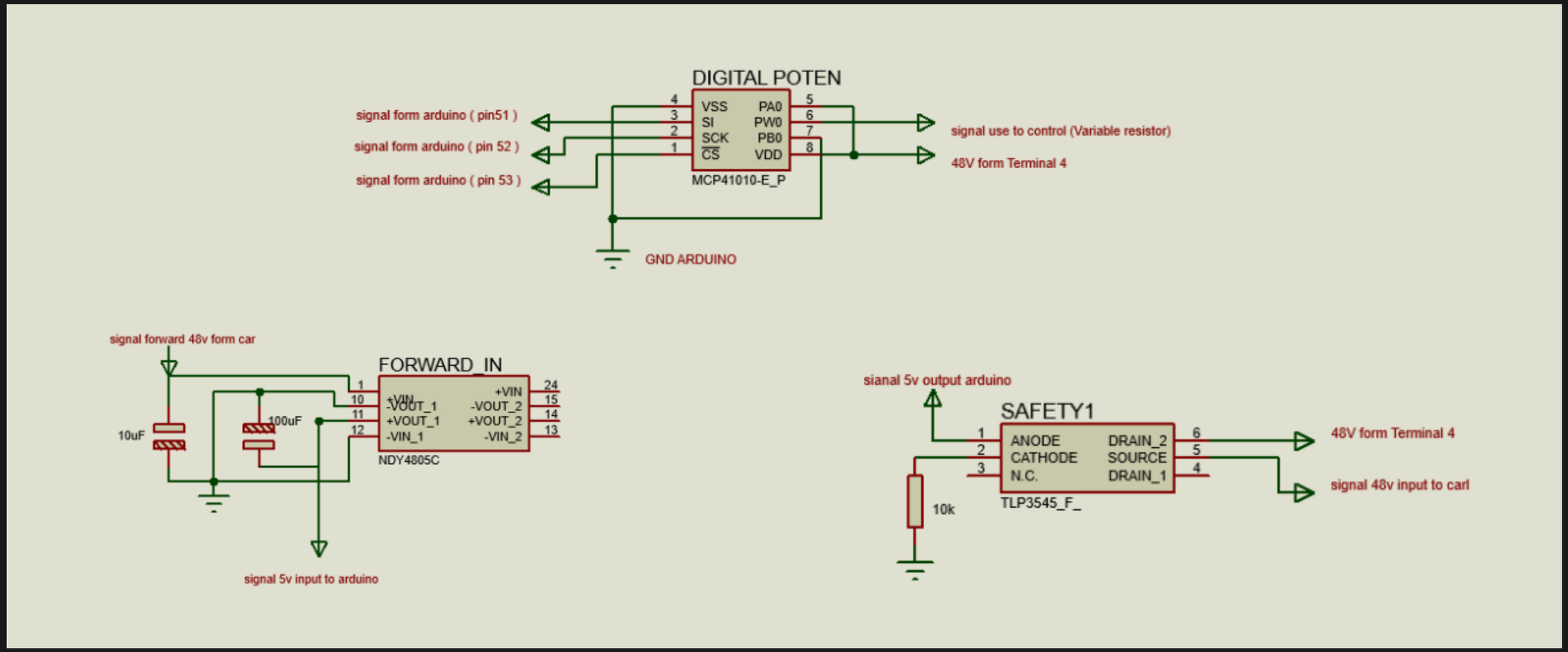
Manual Mode : CIRCUIT VEHICLE



- Have outlet voltage at terminal control forward when on switch backward
- When on switch forward have voltage outlet at terminal control backward and switch control backward is working

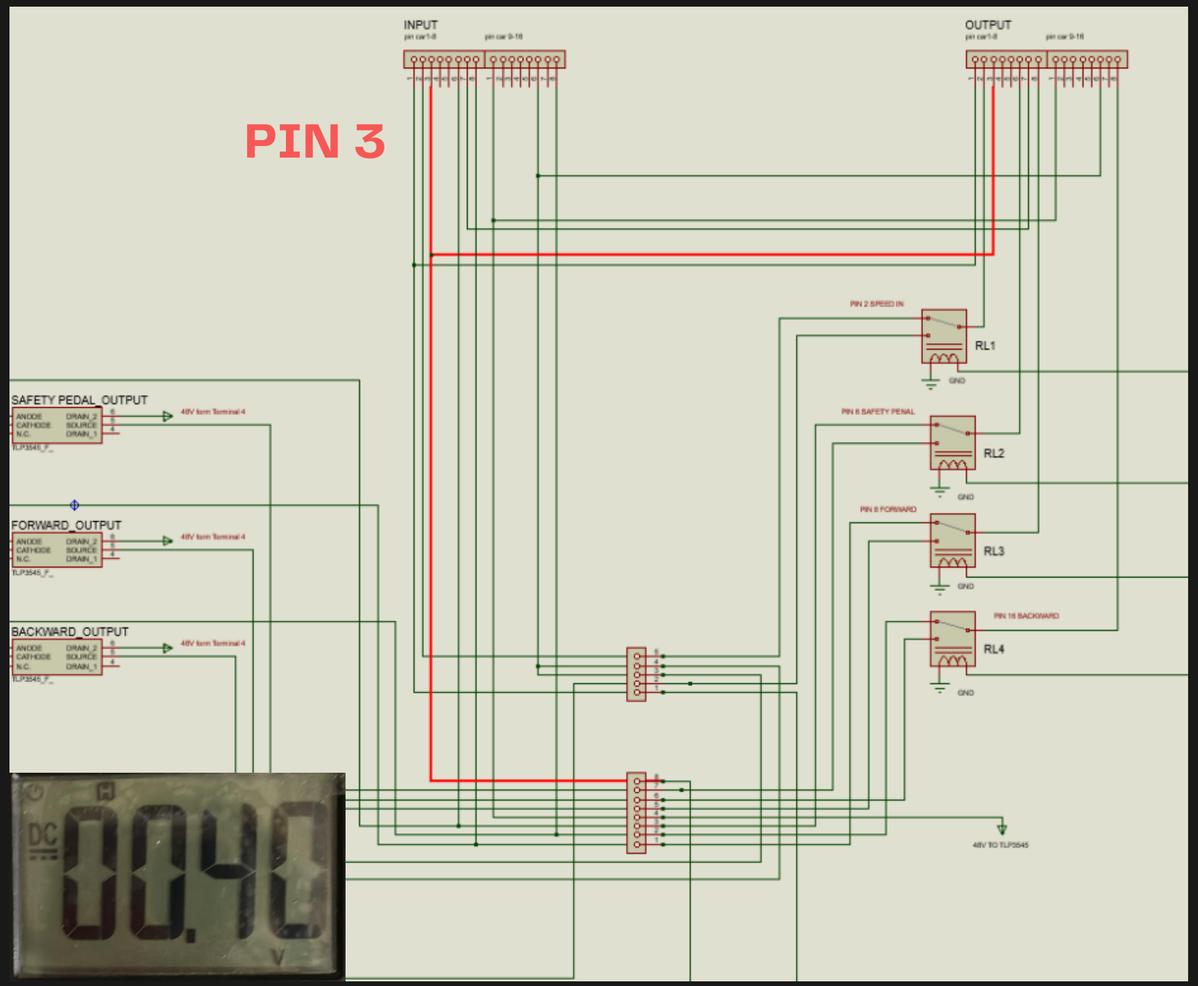
SOLVE

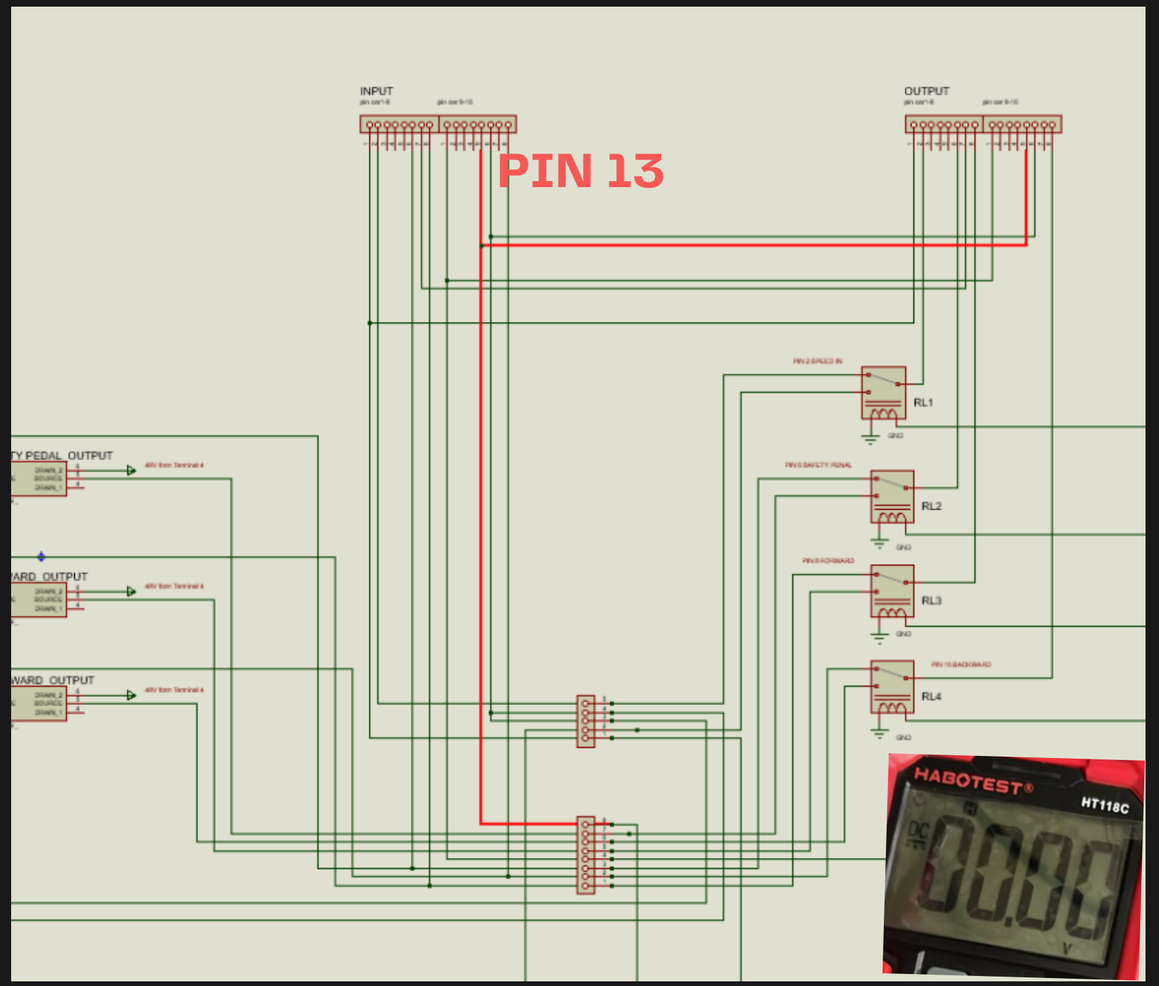
Check IC : step down ,step up , digital potentiometer

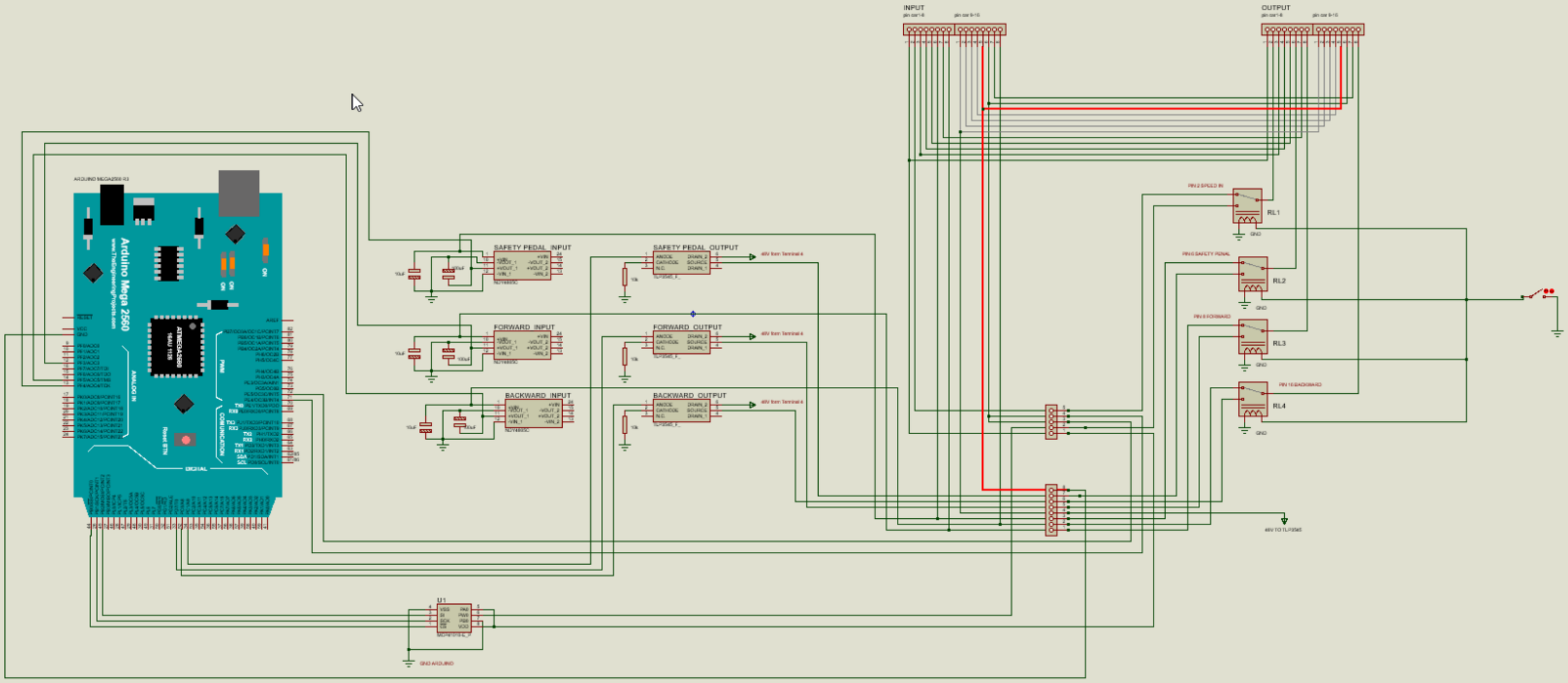


SOLVE

CHECK GND



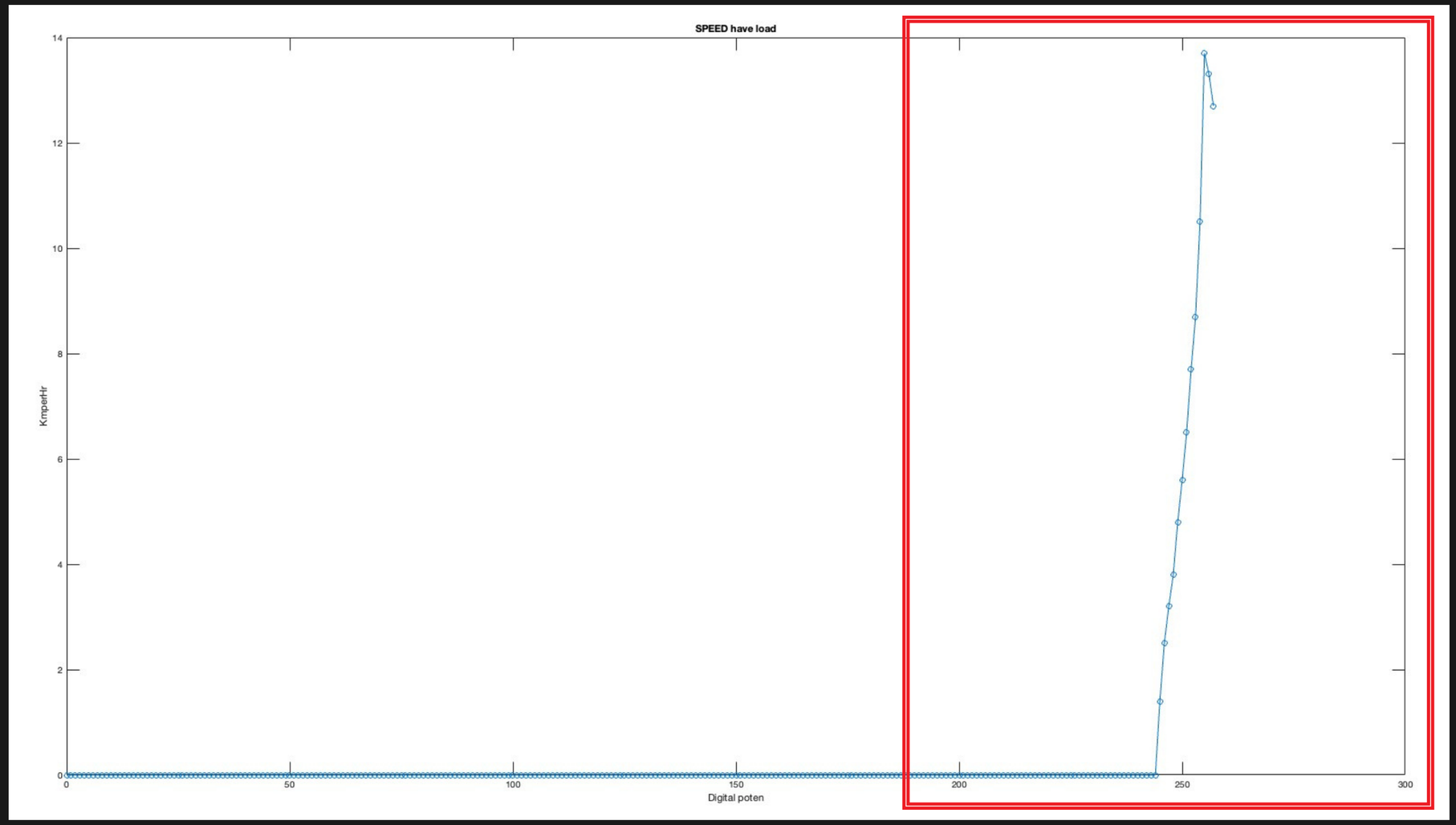




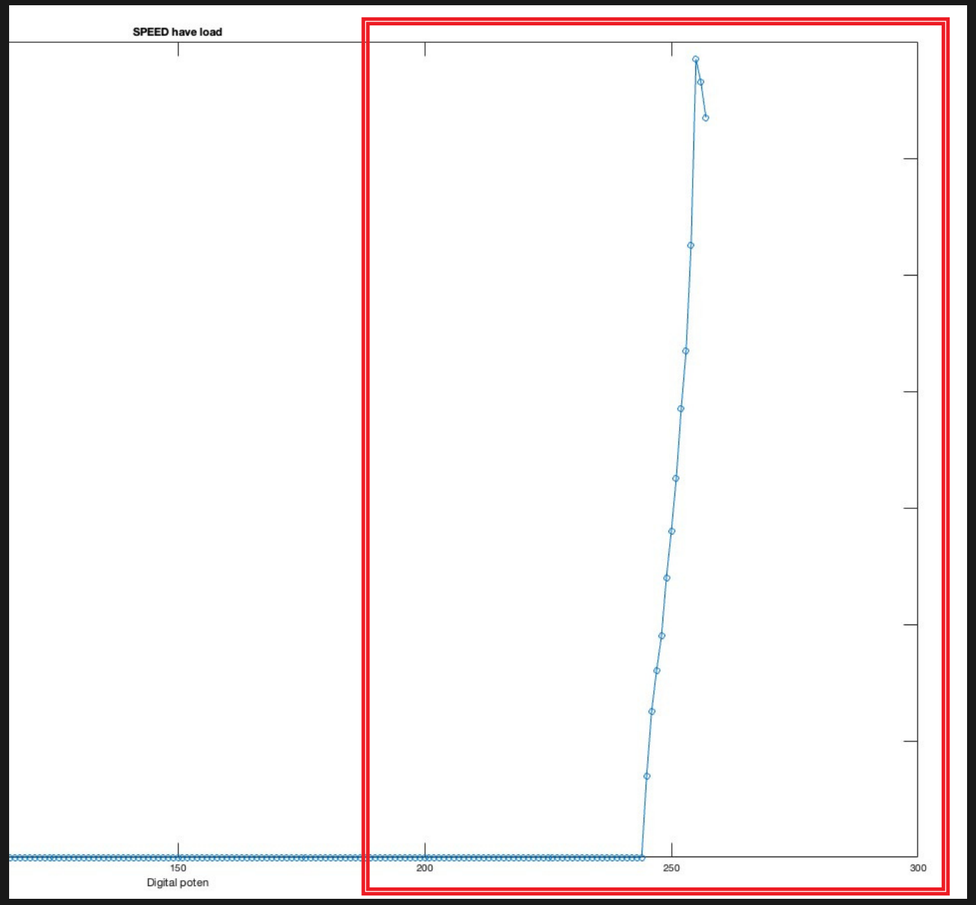
The Autonomous Vehicle: GNSS Technology Tracking Using Controller

P A R T C O N T R O L

<u>AUTO MODE</u> : Km/hr (have load)



AUTO MODE : Km/hr (have load)

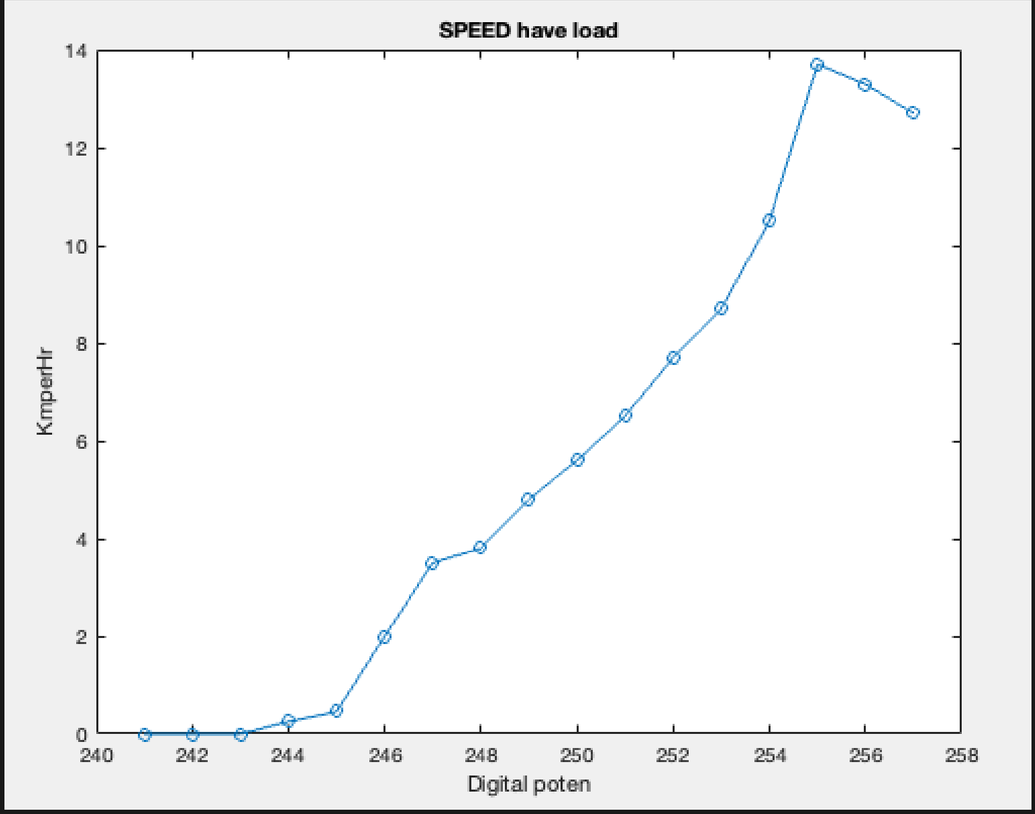


Digital Potentiometer (MCP41010)

→ Each potentiometer is made up of a variable resistor

→ Data register that determines the wiper positionan 8-bit (256 position)

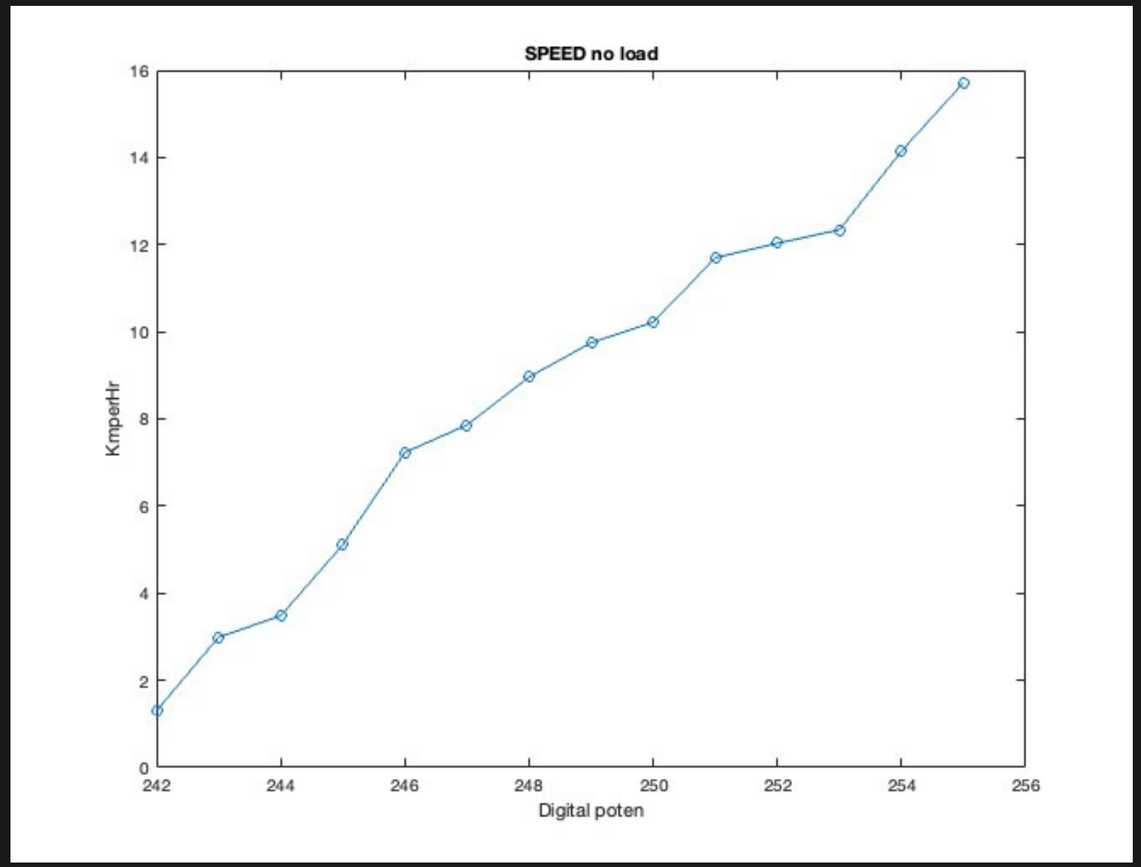
AUTO MODE : Km/hr (have load)



To adjust the value to control the speed for the wheels to work.

HAVE LOAD DIGITAL POTEN = 244

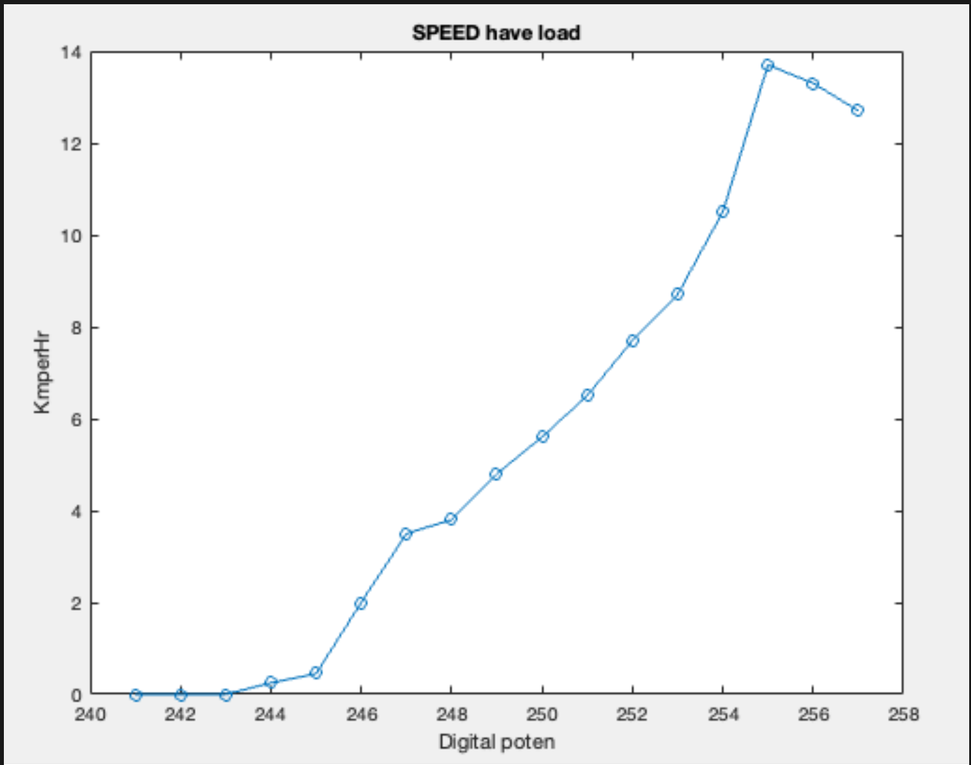
AUTO MODE : Km/hr (no load)



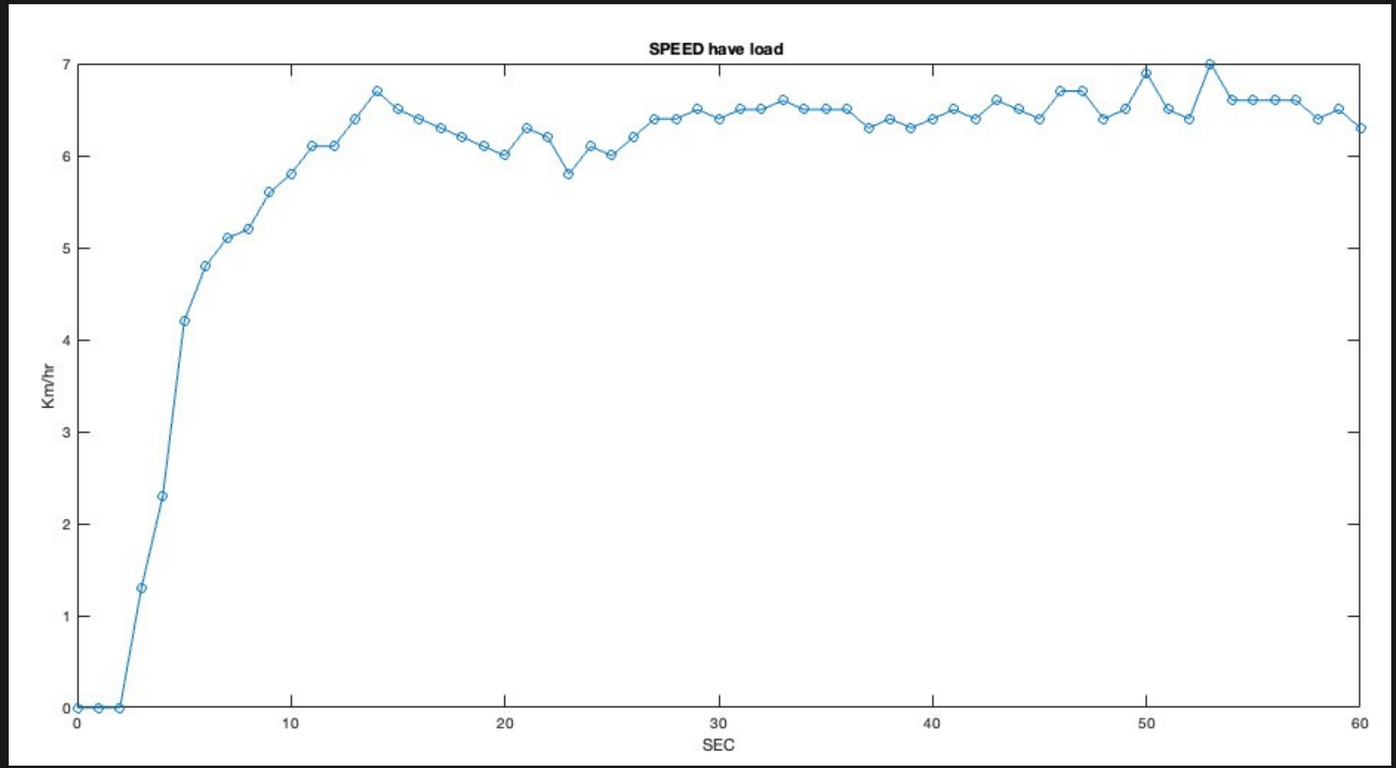
To adjust the value to control the speed for the wheels to work.

NO LOAD DIGITAL POTEN = 242

AUTO MODE : Km/hr (have load)



HOW TO POLT GRAPH



At pwm is 250 mean = 5.8 km/hr

Steering Control System

The Autonomous Vehicle: GNSS Technology Tracking Using Controller

Scope of Steering Control System

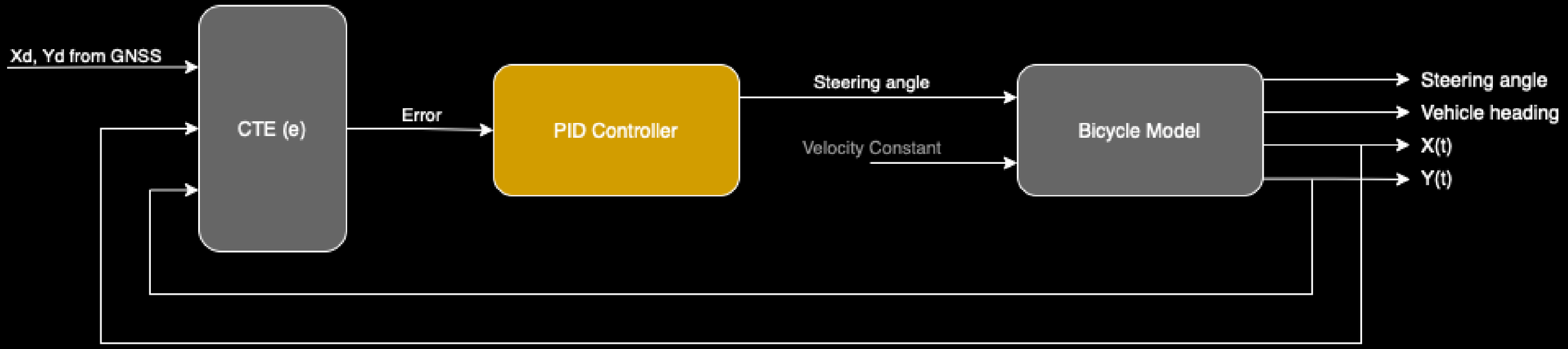
Steering wheel steering gear motor
Model No. KY15DD11-10

PID Controller and Calculated with the Bicycle model equation

CANBUS Protocol for replace RS-232

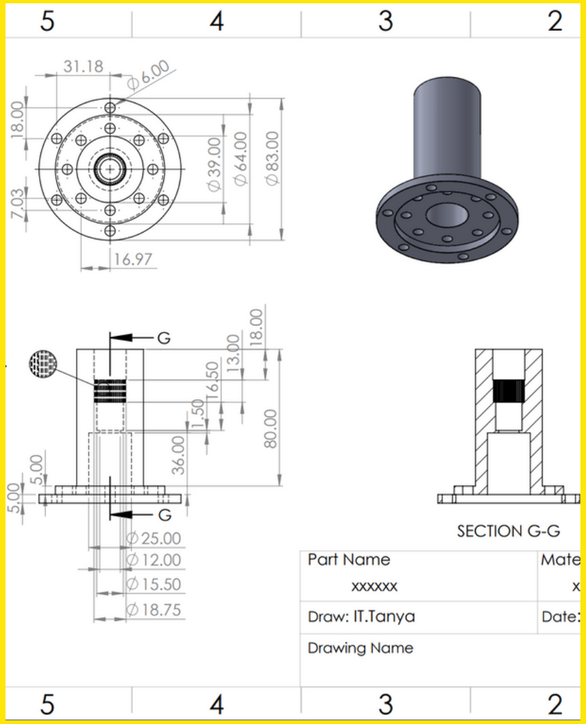
Steering Wheel

Overall Block Diagram

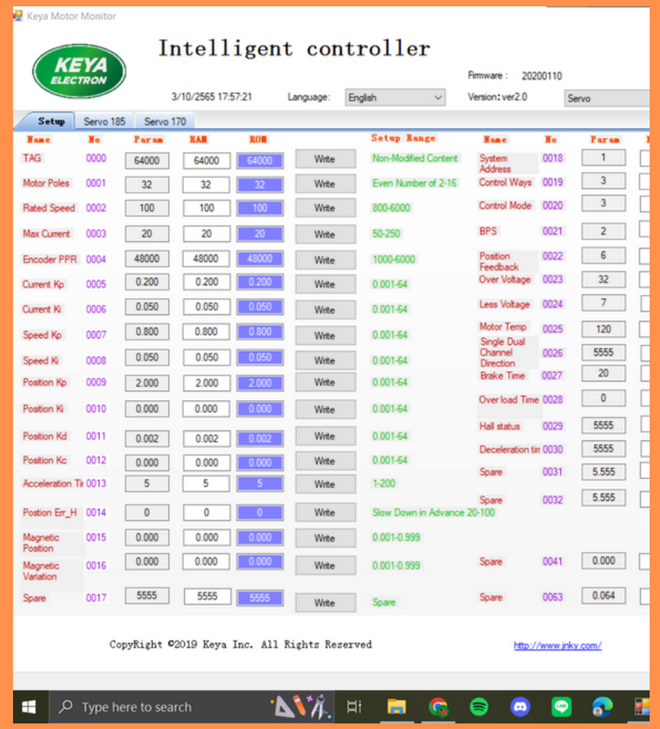


Scope of steering control system

Design Steering Part



Motor Setting



PID Controller Tuning

To control the steering angle and direction of the car's movement which will cause the error value to be close to "zero" or not cause the most error

Communicate with CAN

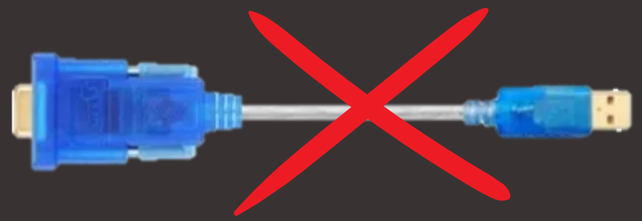
Test and Improve the system from RS232 to CAN bus to make communication more efficient and faster as well as being able to detect various system errors

Fixes & Improves

Fix and improve the operation system to be the most stable and accurate

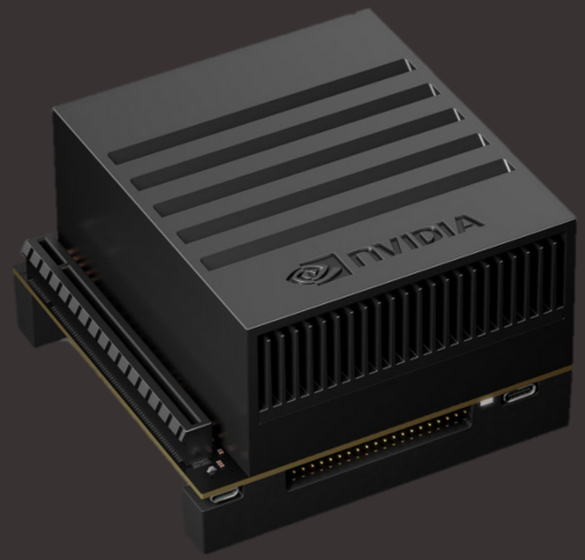
Steering System with Communicate

Steering Motor



RS-232

CAN bus



Jetson Xavier

Steering Motor Setting

New Parameters setting

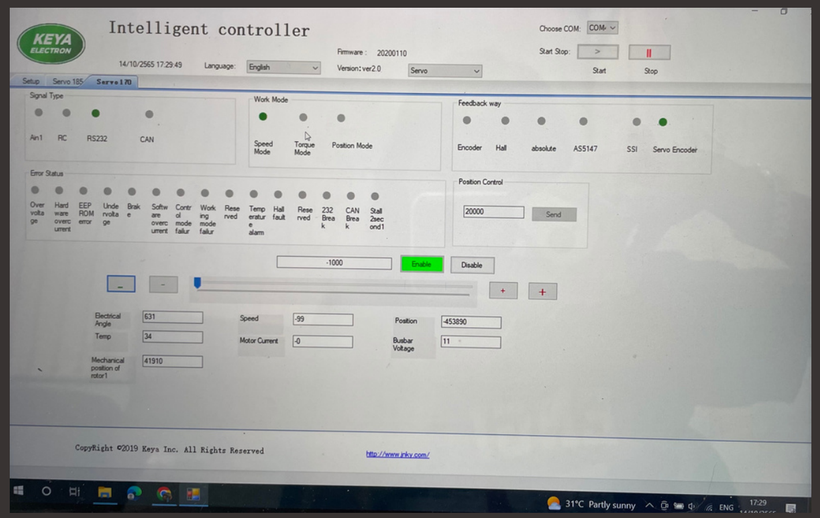
KEYA ELECTRON

Steering Motor

Keya Electron Software

Steering Motor Setting

Parameters are set in the Keya Electron Program to enable communication with the steering motor



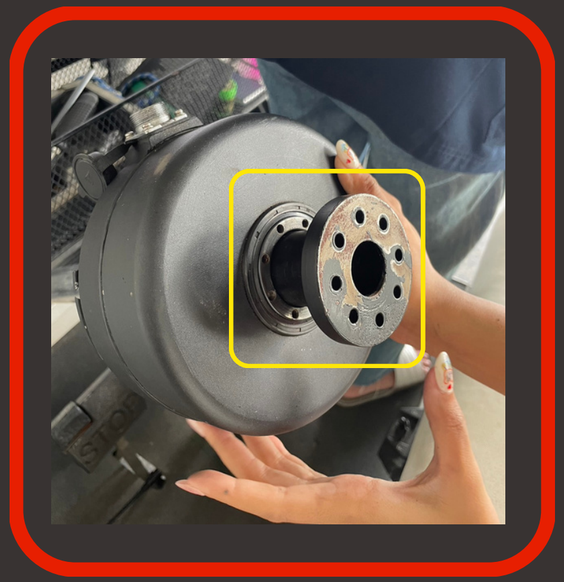
Steering Part Design

Improved and redesigned the steering part to be installed between the motor and the steering wheel of the golf cart's steering wheel for steering angles and directions.

Start by removing the steering wheel that is firmly attached to the car

Steering Part Design

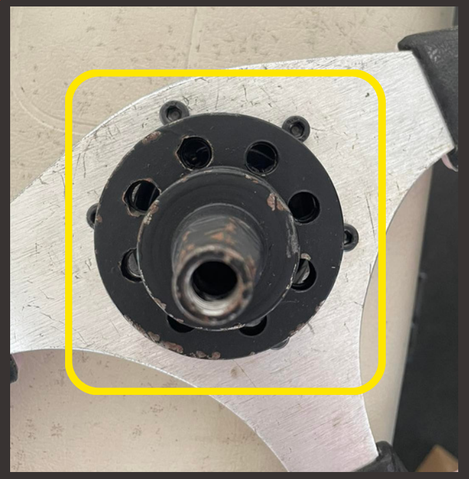
Improved and redesigned the steering part to be installed between the motor and the steering wheel of the golf cart's steering wheel for steering angles and directions.

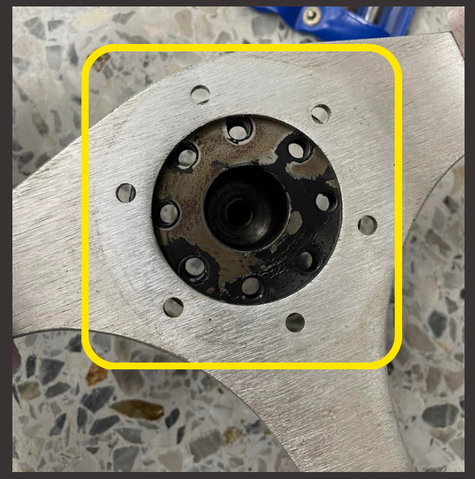


Install the steering motor to the car

Steering Part Design

Improved and redesigned the steering part to be installed between the motor and the steering wheel of the golf cart's steering wheel for steering angles and directions.

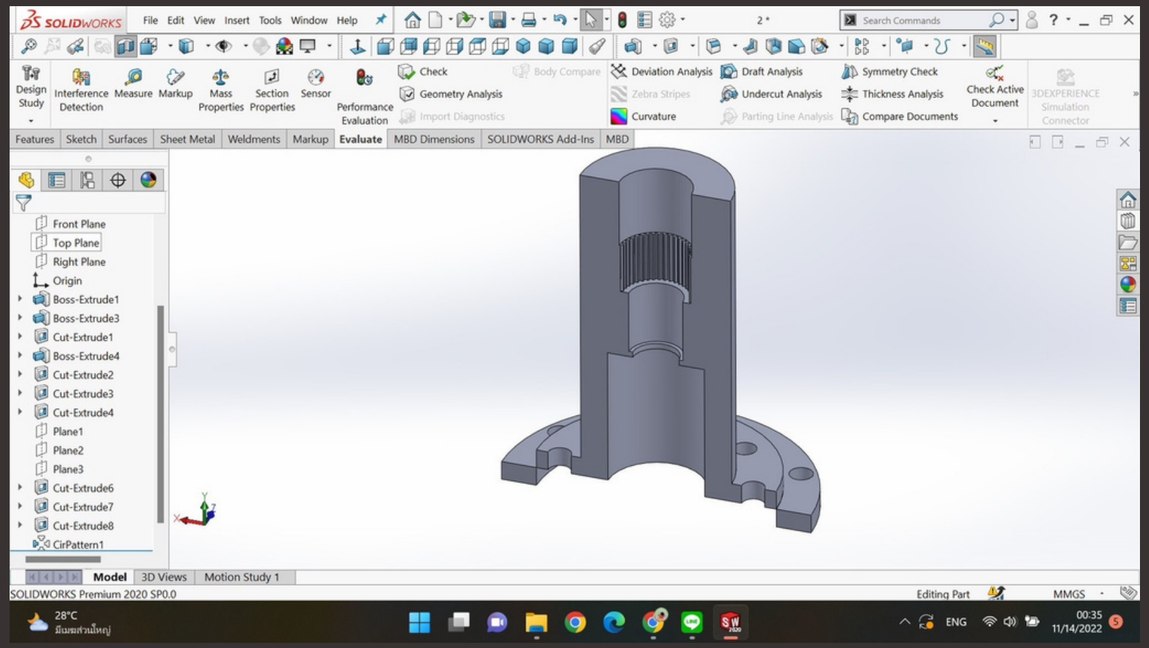


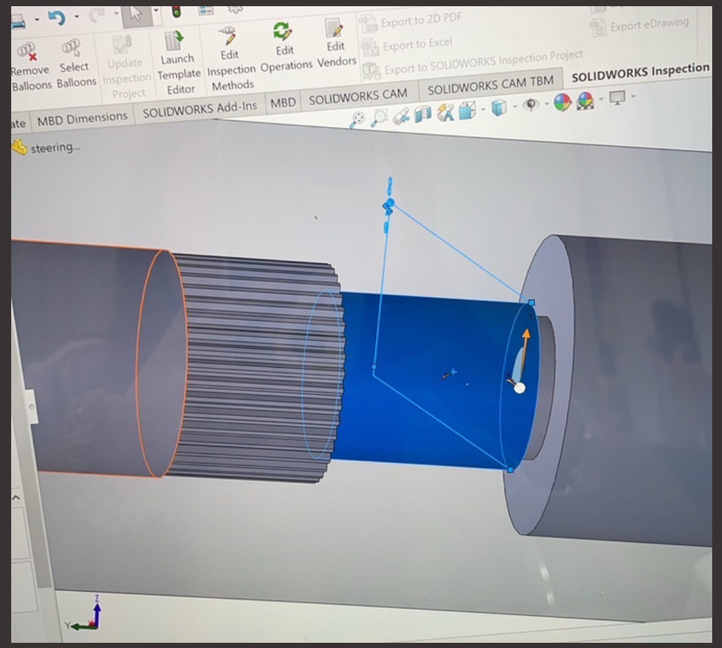


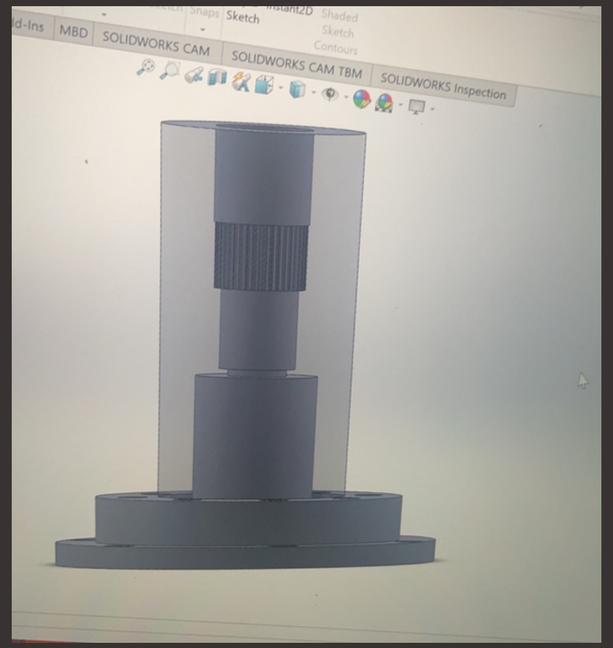
redesigned

SOLIDWORKS

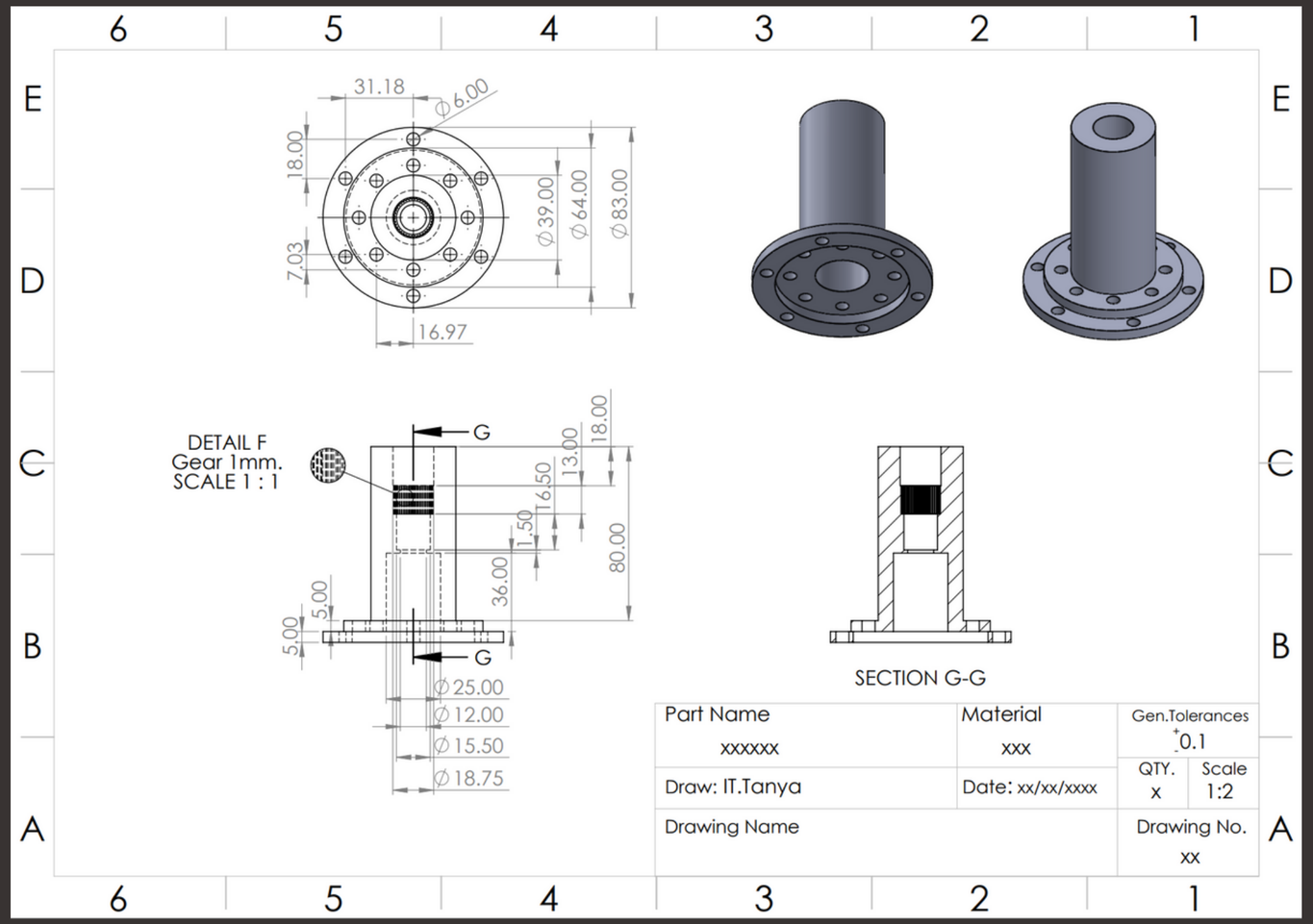
Steering Part Design

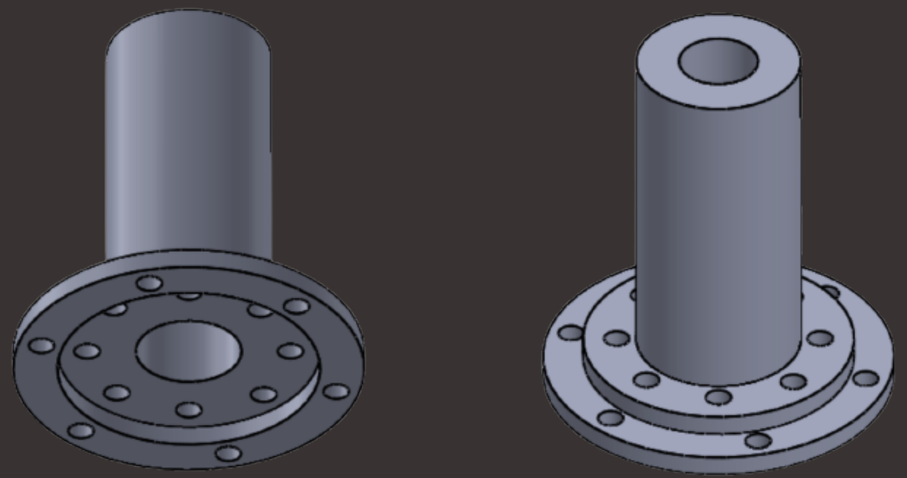






Steering Part Design

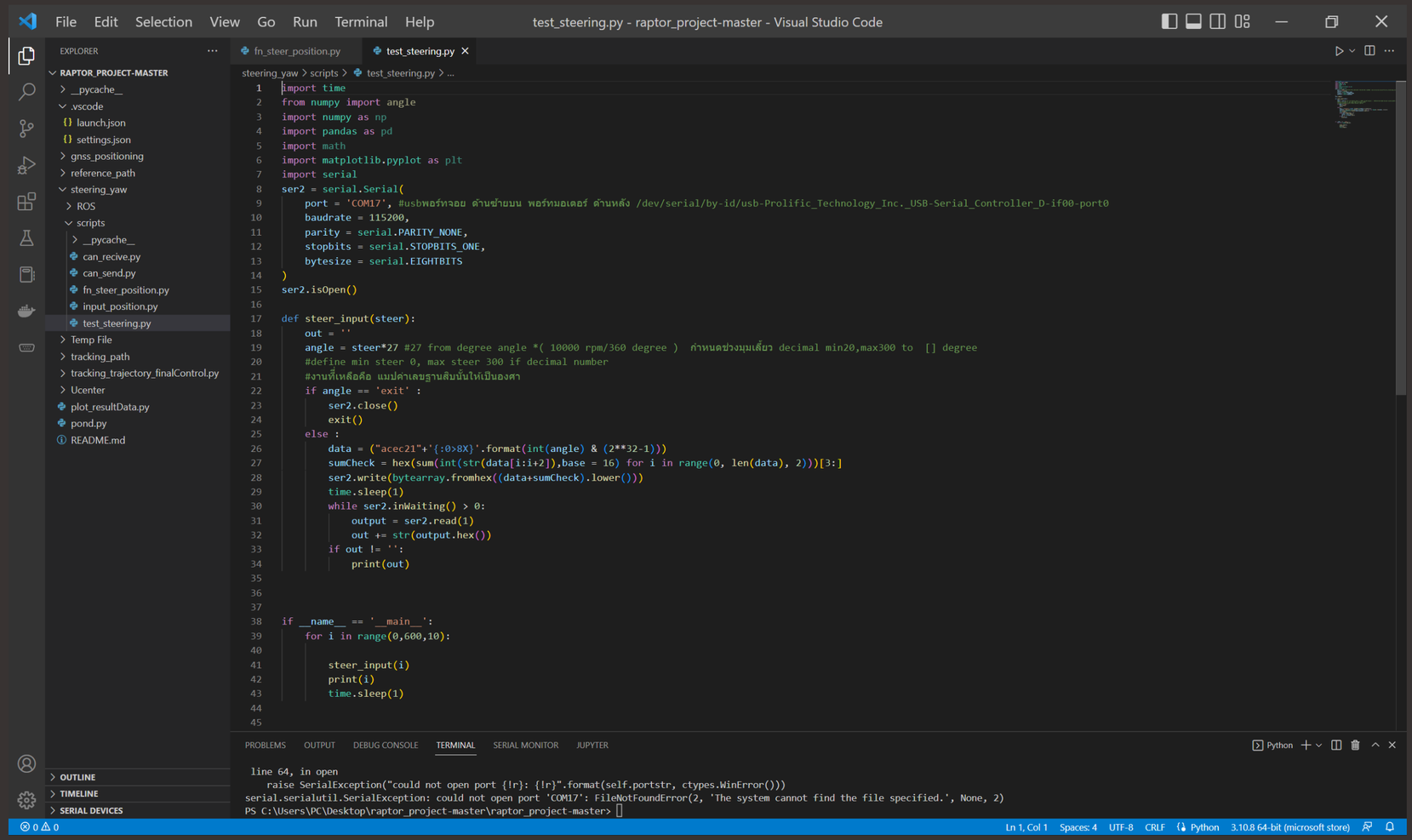




The Autonomous Vehicle: GNSS Technology Tracking Using Controller

Steering Part Design

Steering Motor Test Code



Result of Steering Motor Test Code

```
37
38  if __name__ == '__main__':
39      for i in range(0,600,10):
40
41          steer_input(i)
42          print(i)
43          time.sleep(1)
44
```

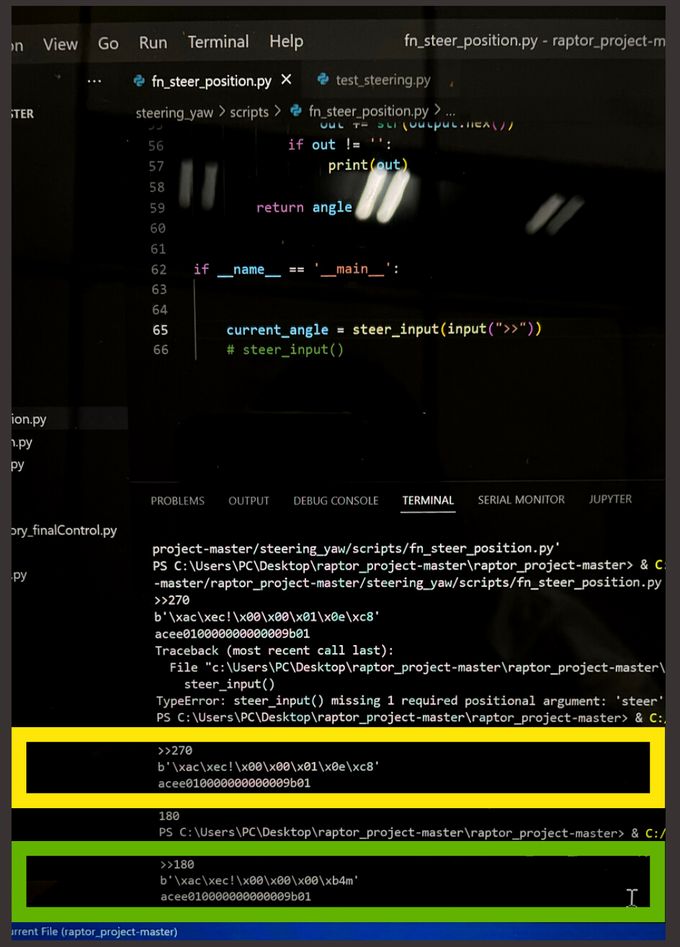
```
acee010000000000009b01
330
acee010000000000009b01
340
acee010000000000009b01
350
acee010000000000009b01
360
acee010000000000009b01
370
acee010000000000009b01
380
acee010000000000009b01
390
acee010000000000009b01
400
acee010000000000009b01
410
acee010000000000009b01
420
acee010000000000009b01
430
```

Result

Function Steering Position Code

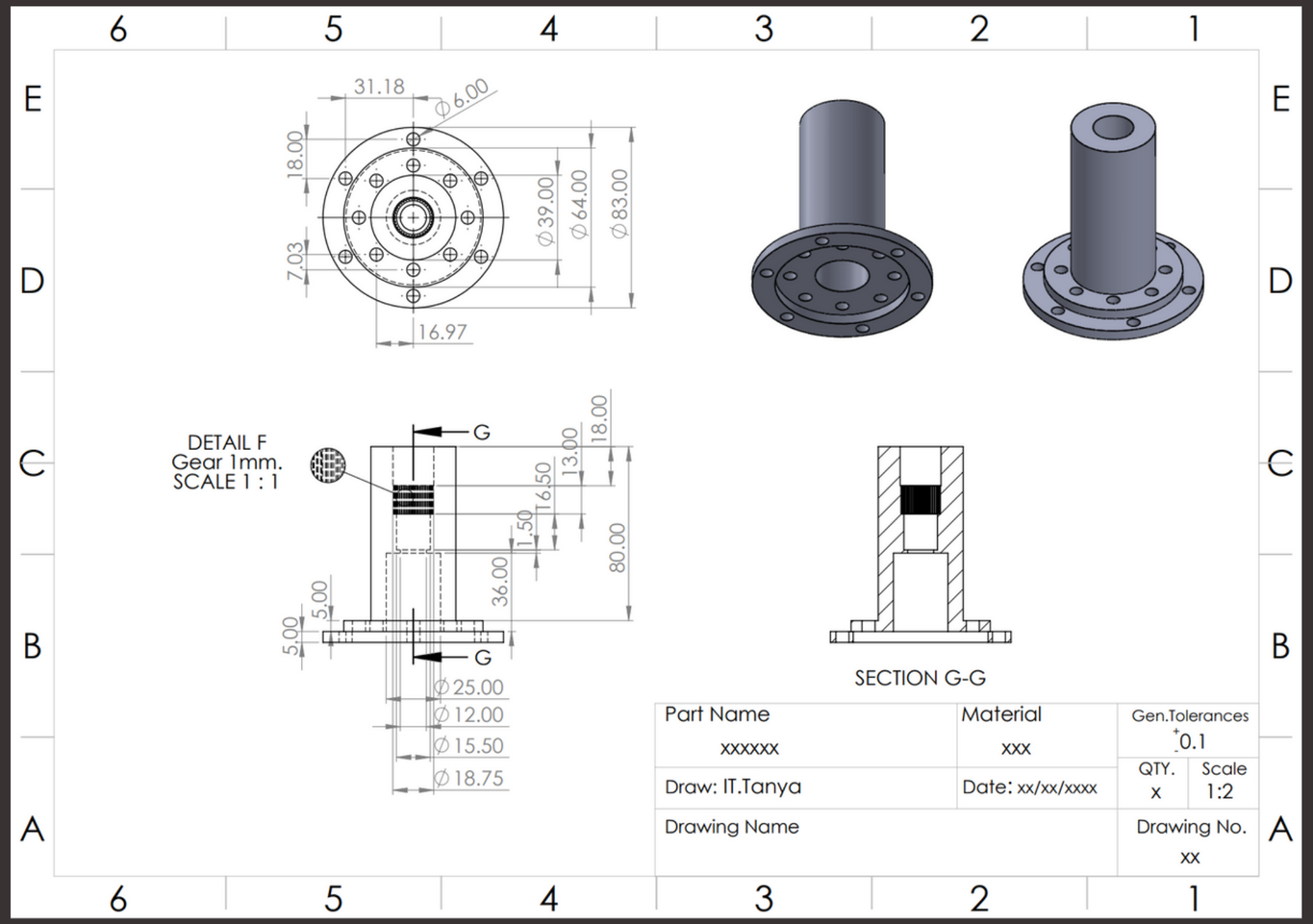
```python
# python 3.8
#run roscore,  rosrun joy joy_node ,rosrun steering_pkg test.py
import time
from numpy import angle
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import serial
ser = serial.Serial(
    port = '/dev/ttyUSB0', #usbพอร์ทจอย ด้านซ้ายบน พอร์ทมอเตอร์ ด้านหลัง /dev/serial/by-id/usb-Prolific_Technolo
    baudrate = 115200,
    parity = serial.PARITY_NONE,
    stopbits = serial.STOPBITS_ONE,
    bytesize = serial.EIGHTBITS
)
ser.isOpen()

 #set param
dt = 0.1 #s
L = 1.68 #m wheel base of vehicle
  #gain controller for control steer to minimize cte
Kp = 1
Ki = 0.1
Kd = 0.5
#controller PID output is yaw_expect
'''
PID cal YAW
        yaw_expect = []
        P = Kp * CTE_t
        I = last_integral + Ki *CTE_t *(t - last_time)
        D = Kd*(CTE_t - last_CTE)
        yaw_expect = P+I+D
'''
WAYPOINTS_file = 'xy.csv'      #put file record waypoint that reference for tracking



def steer_input(steer):
        out = ''
        angle = steer
        # angle = int(angle1)
        if angle == 'exit' :
            ser.close()
            exit()
        else :
            data = ("acec21"+f'{int(angle):0>8X}')
            sumCheck = hex(sum(int(data[i:i+2],16) for i in range(0, len(data), 2)))[3:]
            print(bytes.fromhex((data+sumCheck).lower()))
            ser.write(bytes.fromhex((data+sumCheck).lower()))
            time.sleep(1)
            while ser.inWaiting() > 0:
                output = ser.read(1)
                out += str(output.hex())
            if out != '':
                 print(out)

        return angle


if __name__ == '__main__':

    # current_angle = steer_input(input(">>"))
```

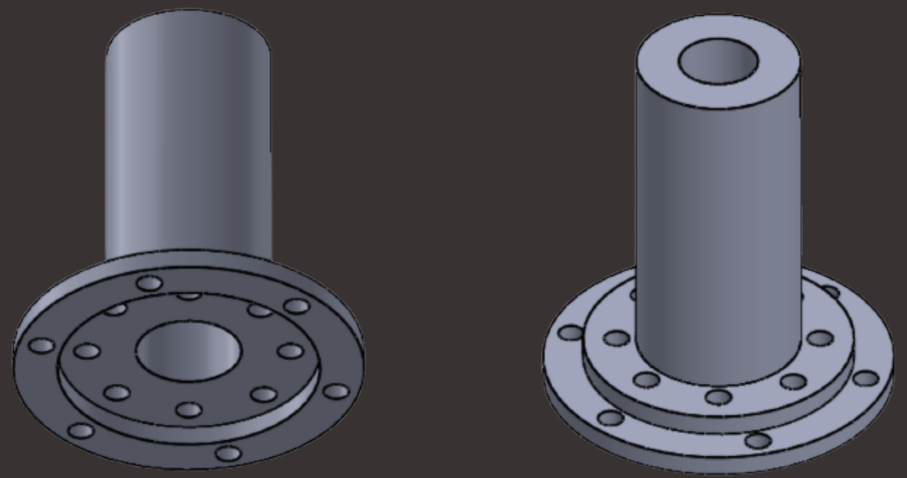
Result



sem2

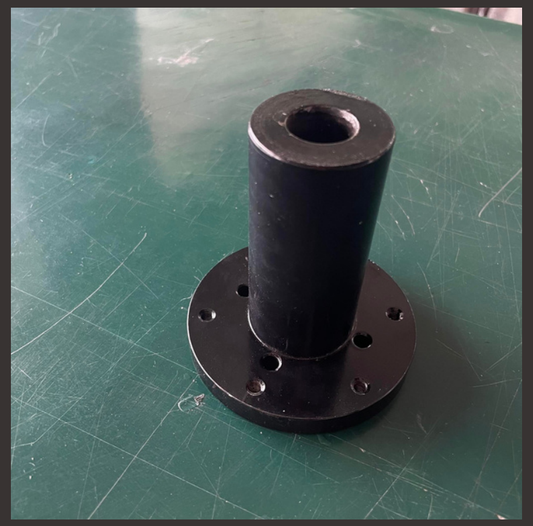
Steering Part Design





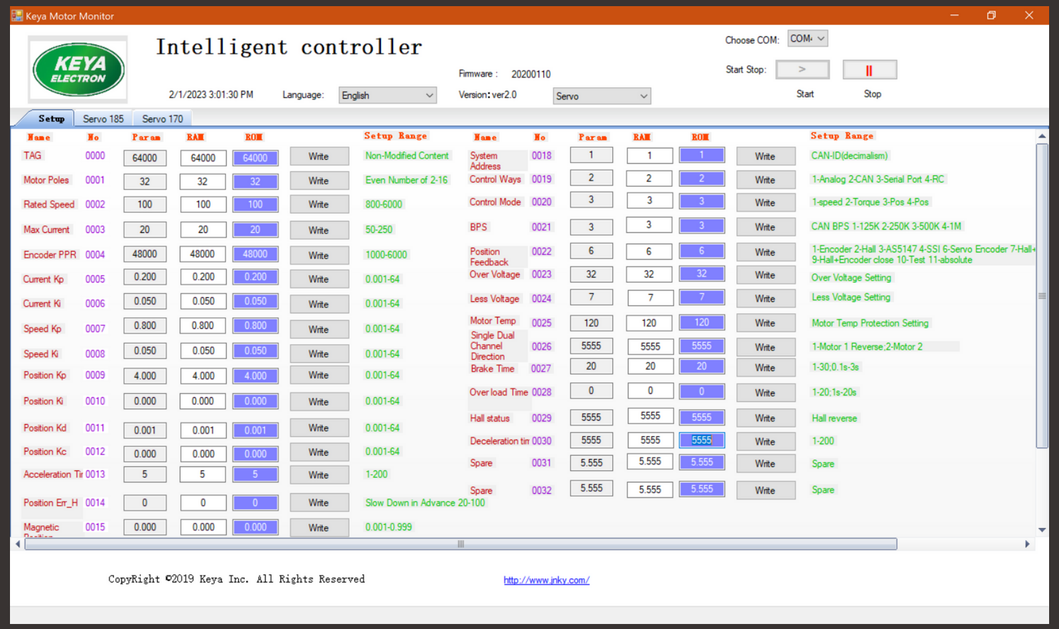
The Autonomous Vehicle: GNSS Technology Tracking Using Controller

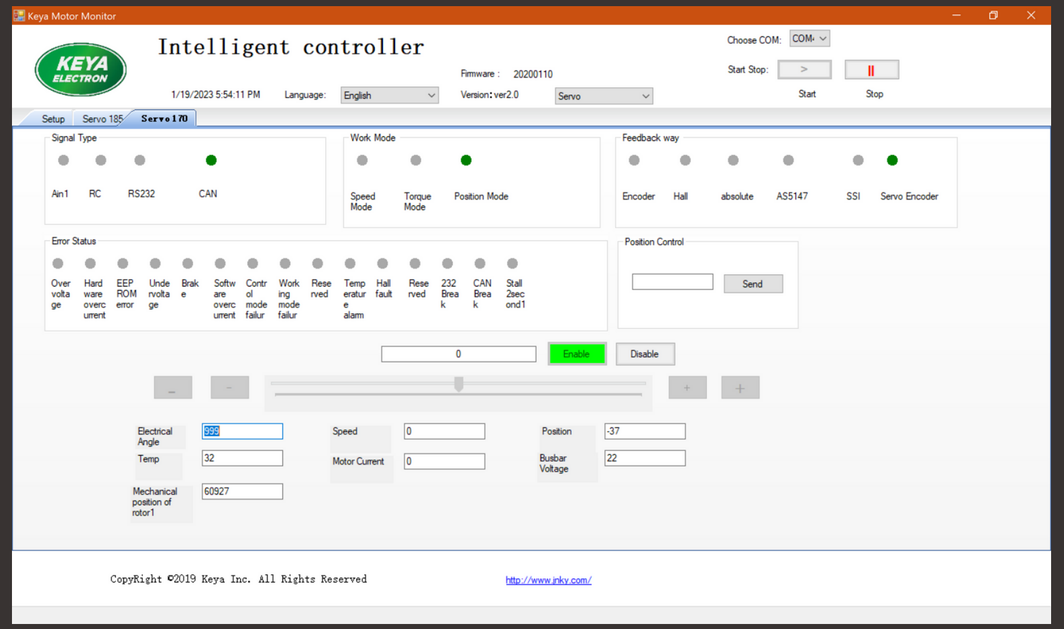
Steering Part Design



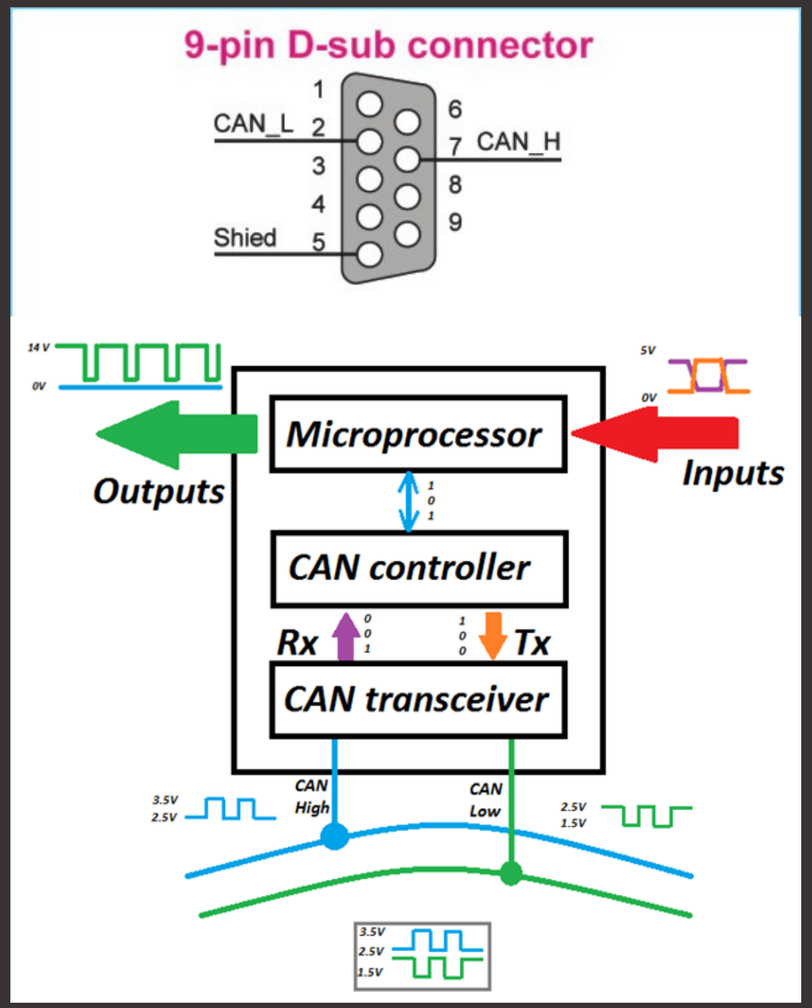
Steering Motor Setting

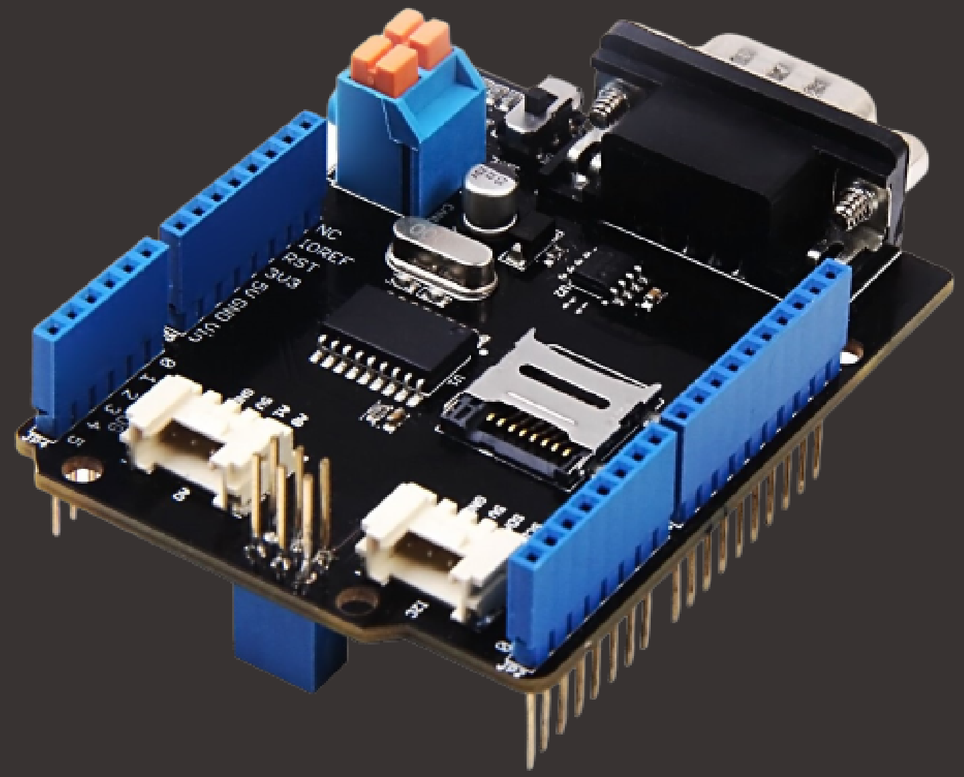
Parameters are set in the Keya Electron Program
to enable communication with the steering motor by CANBUS Protocol





Send-Receive data from CANBus Shield V2.0



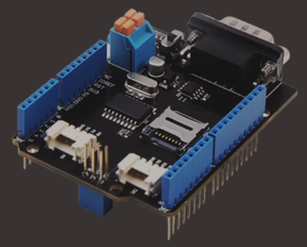


The Autonomous Vehicle: GNSS Technology Tracking Using Controller

Send-Receive data from CANBus Shield V2.0

```
#include <SPI.h>

#define CAN_2515
// #define CAN_2518FD

// Set SPI CS Pin according to your hardware

#if defined(SEEED_WIO_TERMINAL) && defined(CAN_2518FD)
// For Wio Terminal w/ MCP2518FD RPi Hat:
// Channel 0 SPI_CS Pin: BCM 8
// Channel 1 SPI_CS Pin: BCM 7
// Interupt Pin: BCM25
const int SPI_CS_PIN  = BCM8;
const int CAN_INT_PIN = BCM25;
#else

// For Arduino MCP2515 Hat:
// the cs pin of the version after v1.1 is default to D9
// v0.9b and v1.0 is default D10
const int SPI_CS_PIN = 9;
const int CAN_INT_PIN = 2;
#endif


#ifdef CAN_2518FD
#include "mcp2518fd_can.h"
mcp2518fd CAN(SPI_CS_PIN); // Set CS pin
#endif

#ifdef CAN_2515
#include "mcp2515_can.h"
mcp2515_can CAN(SPI_CS_PIN); // Set CS pin
#endif

void setup() {
    SERIAL_PORT_MONITOR.begin(115200);
    while(!Serial){};
```

```
void setup() {
    SERIAL_PORT_MONITOR.begin(115200);
    while(!Serial){};

    while (CAN_OK != CAN.begin(CAN_250KBPS)) {            // init can bus : baudrate = 500k
        SERIAL_PORT_MONITOR.println("CAN init fail, retry...");
        delay(100);
    }
    SERIAL_PORT_MONITOR.println("CAN init ok!");
}

unsigned char stmp[8] = {0, 0, 0, 0, 0, 0, 0, 0};
void loop() {

/*------------------------------------------------------------------------------*/
    unsigned char len = 0;
    unsigned char buf[8];
    unsigned long canId = CAN.getCanId();

    if (CAN_MSGAVAIL == CAN.checkReceive()) {         // check if data coming
        CAN.readMsgBuf(&len, buf);    // read data,  len: data length, buf: data buf

        //unsigned long canId = CAN.getCanId();

        SERIAL_PORT_MONITOR.println("-----------------------------");
        SERIAL_PORT_MONITOR.print("Get data from ID: 0x");
        SERIAL_PORT_MONITOR.println(canId, HEX);

        for (int i = 0; i < len; i++) { // print the data
            SERIAL_PORT_MONITOR.print(buf[i], HEX);
            SERIAL_PORT_MONITOR.print("\t");
        }
        SERIAL_PORT_MONITOR.println();
    }

/*------------------------------------------------------------------------------*/
// send data:  id = 0x00, standrad frame, data len = 8, stmp: data buf
    stmp[7] = stmp[7] + 1;
    if (stmp[7] == 100) {
        stmp[7] = 0;
        stmp[6] = stmp[6] + 1;

        if (stmp[6] == 100) {
            stmp[6] = 0;
            stmp[5] = stmp[5] + 1;

        }
    }

    CAN.sendMsgBuf(0x0580001, 0, 8, 0, stmp);
    delay(1000);                       // send data per 100ms
    SERIAL_PORT_MONITOR.println("CAN BUS sendMsgBuf ok!");
    SERIAL_PORT_MONITOR.print("ID 0x");
    SERIAL_PORT_MONITOR.println(canId, HEX);


}
```

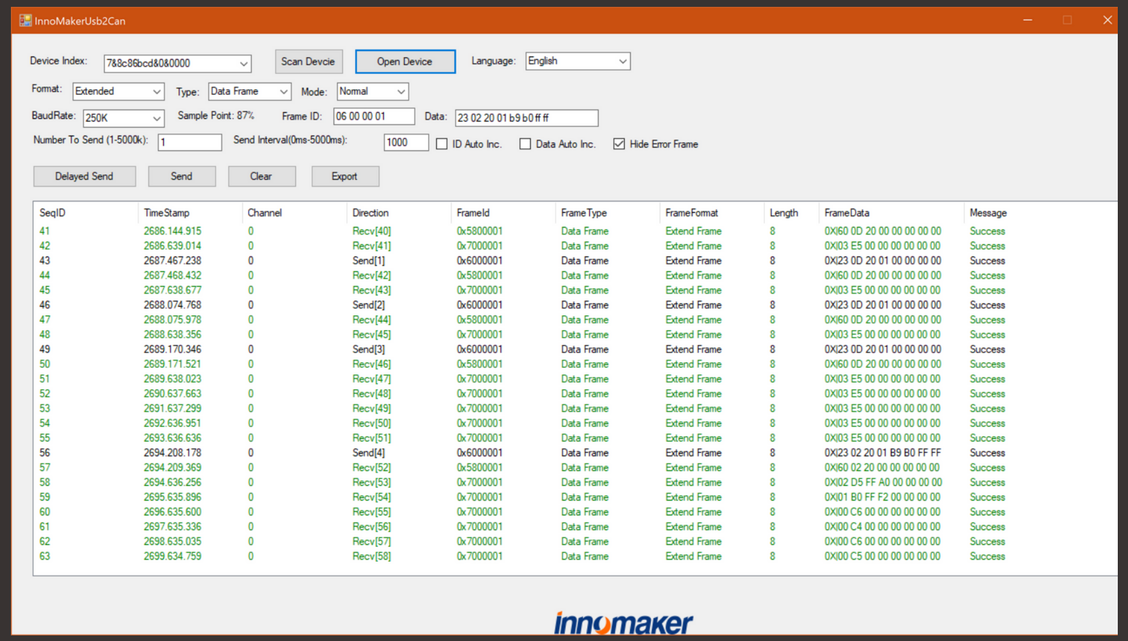


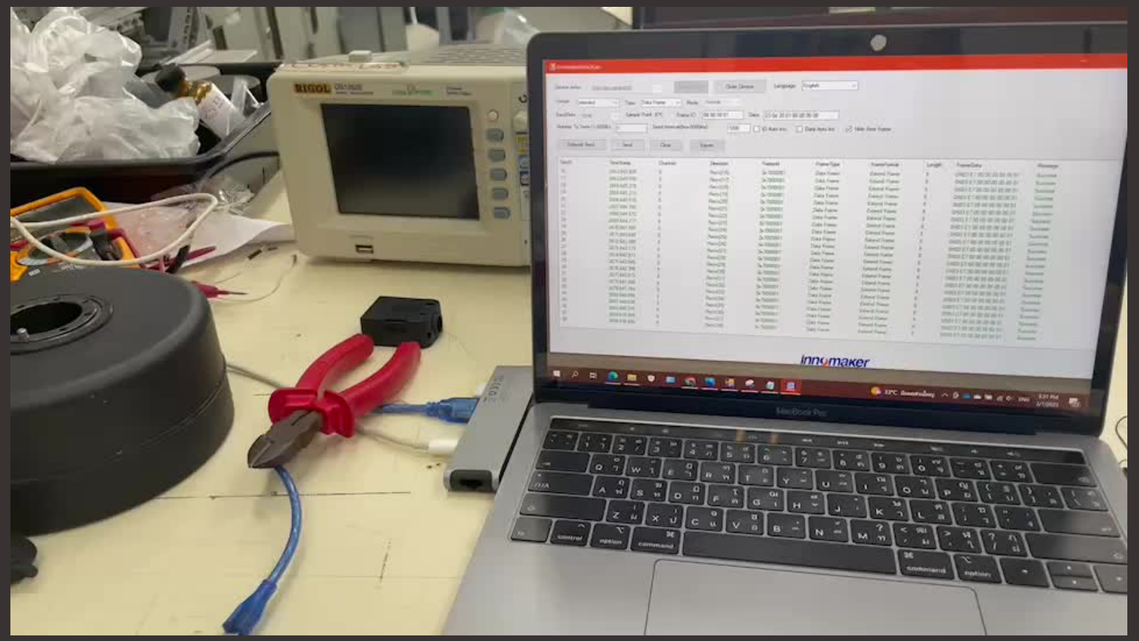
Result

```
13:20:37.773 -> CAN BUS sendMsgBuf ok!
13:20:37.773 -> ID 0x580001
13:20:37.773 -> -----------------------------
13:20:37.773 -> Get data from ID: 0x580001
13:20:37.773 -> 2       38      0       0       0       0       0       0
13:20:38.785 -> CAN BUS sendMsgBuf ok!
13:20:38.785 -> ID 0x580001
13:20:38.785 -> -----------------------------
13:20:38.785 -> Get data from ID: 0x580001
13:20:38.785 -> 2       38      0       0       0       0       0       0
13:20:39.796 -> CAN BUS sendMsgBuf ok!
13:20:39.796 -> ID 0x580001
13:20:39.796 -> -----------------------------
13:20:39.796 -> Get data from ID: 0x580001
13:20:39.796 -> 2       38      0       0       0       0       0       0
13:20:40.810 -> CAN BUS sendMsgBuf ok!
13:20:40.810 -> ID 0x580001
13:20:40.810 -> -----------------------------
13:20:40.810 -> Get data from ID: 0x580001
13:20:40.810 -> 2       38      0       0       0       0       0       0
13:20:41.789 -> CAN BUS sendMsgBuf ok!
13:20:41.822 -> ID 0x580001
13:20:41.822 -> -----------------------------
13:20:41.822 -> Get data from ID: 0x580001
13:20:41.822 -> 2       38      0       0       0       0       0       0
13:20:42.799 -> CAN BUS sendMsgBuf ok!
13:20:42.799 -> ID 0x580001
13:20:42.799 -> -----------------------------
13:20:42.799 -> Get data from ID: 0x580001
13:20:42.833 -> 2       38      0       0       0       0       0       0
13:20:43.811 -> CAN BUS sendMsgBuf ok!
13:20:43.811 -> ID 0x580001
13:20:43.811 -> -----------------------------
13:20:43.811 -> Get data from ID: 0x580001
13:20:43.811 -> 2       38      0       0       0       0       0       0
13:20:44.824 -> CAN BUS sendMsgBuf ok!
13:20:44.824 -> ID 0x580001
13:20:44.824 -> -----------------------------
13:20:44.824 -> Get data from ID: 0x580001
13:20:44.824 -> 2       37      0       0       0       0       0       0
13:20:45.837 -> CAN BUS sendMsgBuf ok!
13:20:45.837 -> ID 0x580001
13:20:45.837 -> -----------------------------
13:20:45.837 -> Get data from ID: 0x580001
13:20:45.837 -> 2       37      0       0       0       0       0       0
```

Send-Receive data from USB2CAN

Through InnoMakerUSB2CAN Program for Interface between The steering motor and mcu





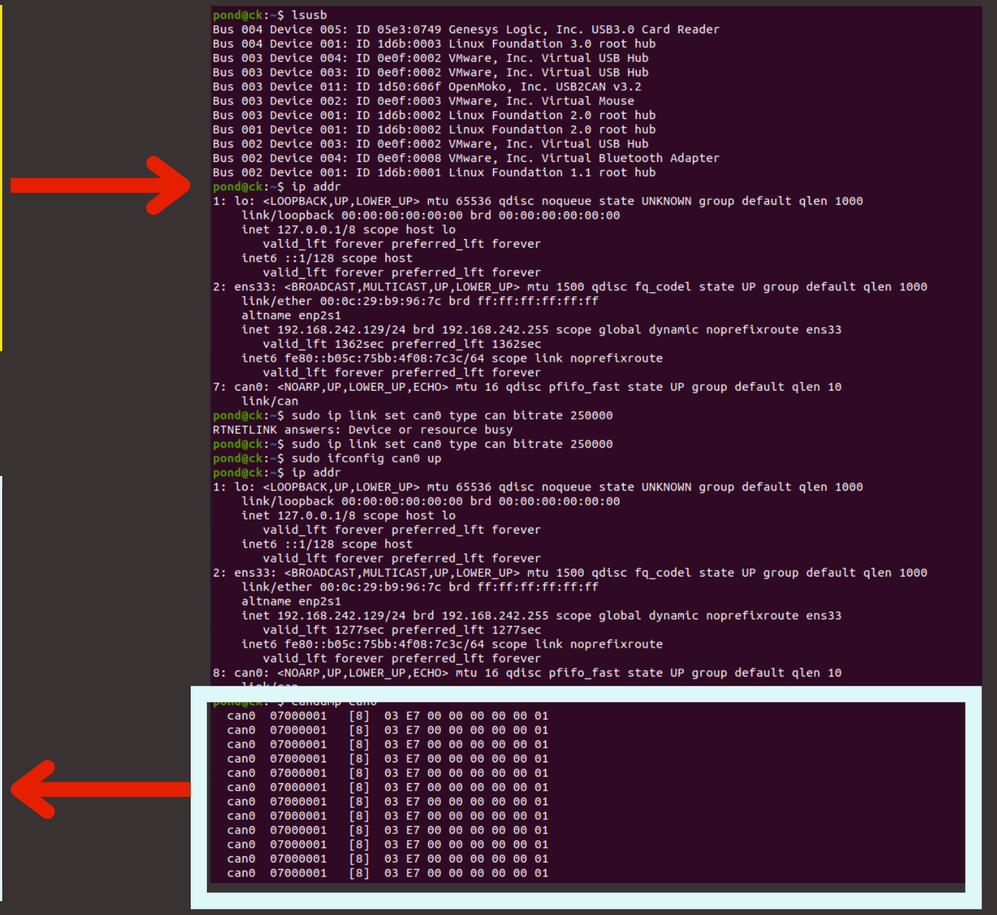
Install and Receive data by CANBUS on Ubuntu

Install can-utils

```
ond@ck:~$ sudo apt-get install can-utils
eading package lists... Done
uilding dependency tree
eading state information... Done
an-utils is already the newest version (2018.02.0-1ubuntu1).
 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
ond@ck:~$ ifconfig vcan0
can0: flags=193<UP,RUNNING,NOARP>  mtu 72
      unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

CAN Received data (candump can0)

```
ond@ck:~$ candump can0
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
 can0  07000001   [8]  03 E7 00 00 00 00 00 01
```



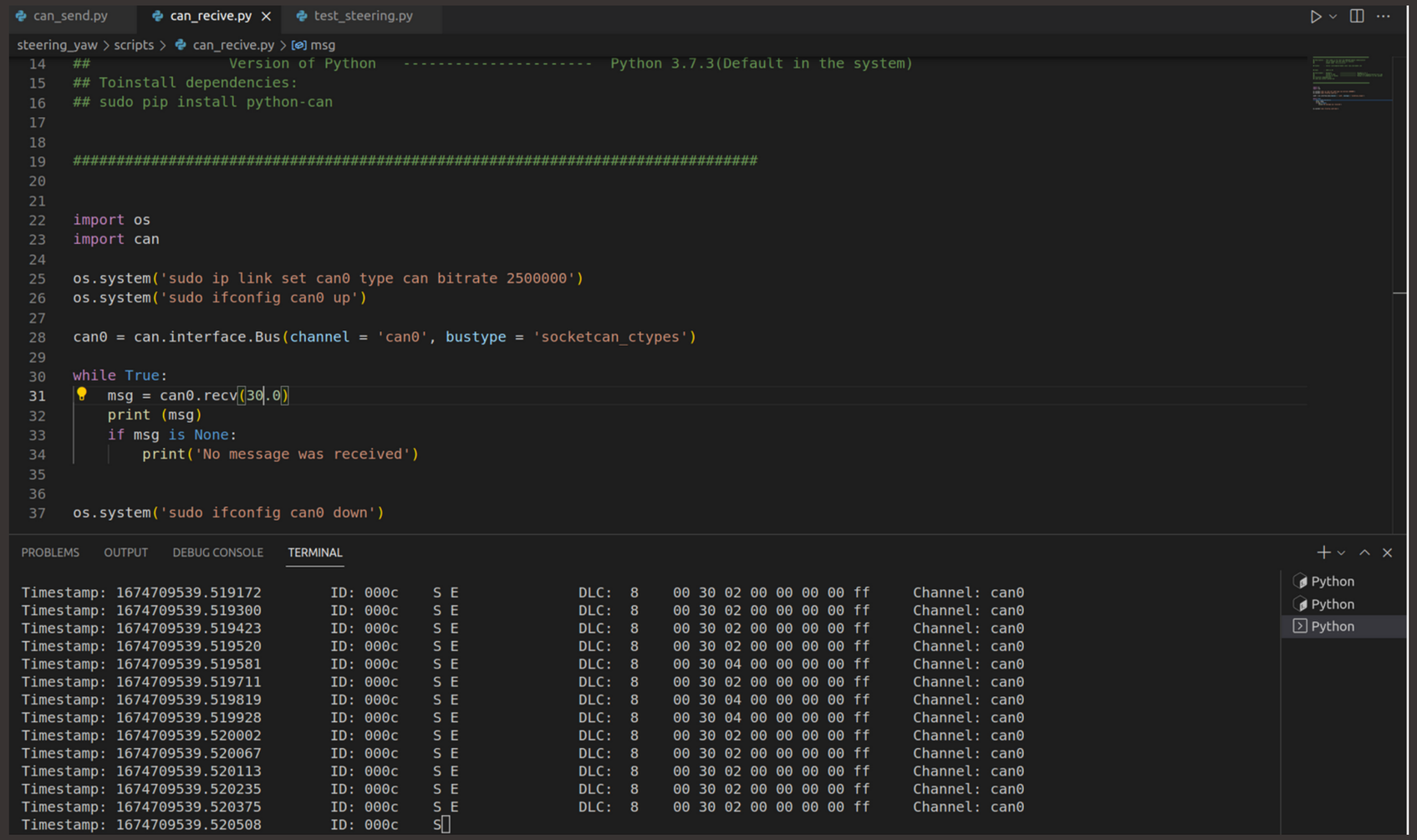
Send data code from USB2CAN

```
61
62  import os
63  import can
64  import time
65  from ast import literal_eval
66
67  #check system name, in linux will print 'posix' and in windows will print 'nt'
68  print(os.name)
69
70  os.system('sudo ifconfig can0 down')
71  os.system('sudo ip link set can0 type can bitrate 250000')
72  os.system("sudo ifconfig can0 txqueuelen 250000")
73  os.system('sudo ifconfig can0 up')
74
75  can0 = can.interface.Bus(channel = 'can0', bustype = 'socketcan')
76  send_count = 0
77  recv_count = 0
78
79  msg = can.Message(arbitration_id=0x06000001, data=[0x23,0x0d,0x20,0x01,0x00,0x00,0x00,0x00])    # Enable
80  # print("msg = ",msg )
81  can0.send(msg)
82  time.sleep(1)
83  print("Message sent on Enable state data: {}".format(msg.data[4:])) #Message sent on Enable state data bytearray(b'\x00\x00\x00\x00')
84
85
86  # msg = can.Message(arbitration_id=0x06000001, data=[0x23, 0x02, 0x20, 0x01, 0x04, 0xbf, 0x00, 0x00])
87  # print('send msg: ',msg.data)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Message sent on Enable state data: bytearray(b'\x00\x00\x00\x00')
```

Receive data code from USB2CAN



Send-Receive data code from USB2CAN

```python
import os
import can
import time

#check system name, in linux will print 'posix' and in windows will print 'nt'
print(os.name)

os.system('sudo ifconfig can0 down')
os.system('sudo ip link set can0 type can bitrate 250000')
os.system("sudo ifconfig can0 txqueuelen 250000")
os.system('sudo ifconfig can0 up')

can0 = can.interface.Bus(channel = 'can0', bustype = 'socketcan')
send_count = 0
recv_count = 0

msg = can.Message(arbitration_id=0x06000001, data=[0x23,0x0d,0x20,0x01,0x00,0x00,0x00,0x00])    # Enable
# print("msg = ",msg )
can0.send(msg)
time.sleep(1)
print("Message sent on Enable state data: {}".format(msg.data[4:])) #Message sent on Enable state data bytearray(b'\x00\x00\x00\x00')

out = ''
angle = input(">>")
angle = int(angle)*27 #27 from degree angle *( 10000 rpm/360 degree )  กำหนดช่วงมุมเลี้ยว decimal min20,max300 to  [] degree
        #define min steer 0, max steer 300 if decimal number
        #งานที่เหลือคือ แมปค่าเลขฐานสิบนั้นให้เป็นองศา
print('angle = ',angle)
```

```python
if angle == 'exit' :
    os.system('sudo ifconfig can0 down')
    print('exit')
    exit()

else :

    data = ('{:0>8X}'.format(int(angle) & (2**32-1)))       #i = 45 = 45*27 = 1215, --> data = 000004BF(hex)
    print('pulse(hex) = ',data)

    DATA_Lh = hex(sum(int(str(data[i:i+2]),base = 16) for i in range(0, len(data), 4)))
    print("DATA_Lh = ",DATA_Lh)


    DATA_Ll = hex(sum(int(str(data[i:i+2]),base = 16) for i in range(0, len(data), 3)))
    print("DATA_Ll = ",DATA_Ll)


    DATA_Hh = hex(int('{:0>2X}'.format(int(data[:2])))) #hex(int(str(data[i:i+2]),base = 16) for i in range(0, len(data), 2))
    print("DATA_Hh = ",DATA_Hh)


    DATA_Hl = hex(int('{:0>2X}'.format(int(data[2:4])))) #hex(int(str(data[i:i+2]),base = 16) for i in range(0, len(data), 2))
    print("DATA_Hl = ",DATA_Hl)

    msg = can.Message(arbitration_id=0x06000001, data=[0x23, 0x02, 0x20, 0x01, DATA_Lh, DATA_Ll, DATA_Hh, DATA_Hl])
    print('send msg: ',msg.data)
    can0.send(msg)  #can0.send(msg, timeout=0.2)`enter code here`
    time.sleep(1)
```

Result from Send-Received code

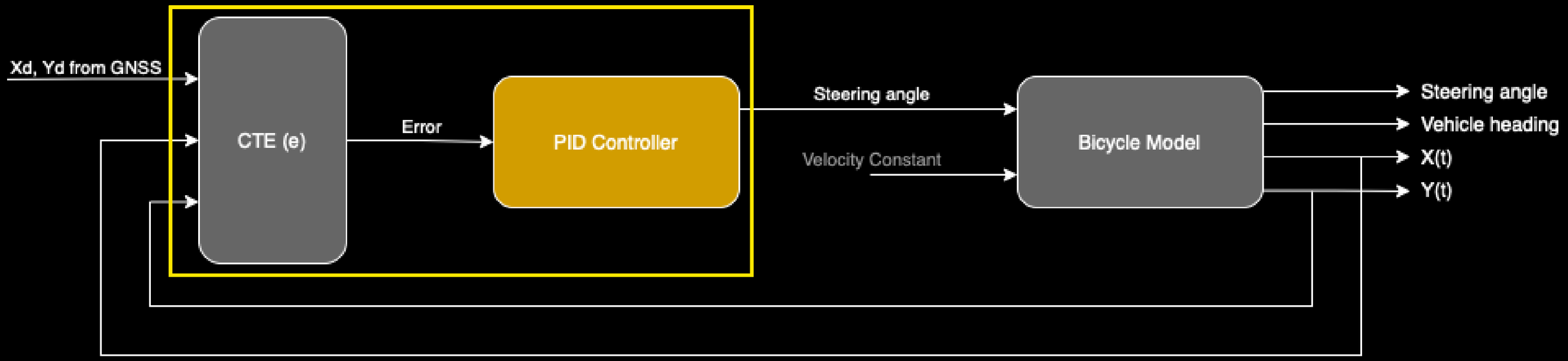
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
pond@ck:~/Desktop/raptor_project-master/raptor_project-master$ /bin/python3 /home/pond/Desktop/raptor_project-master/raptor_project-master/steering_yaw/scripts/c
an_send_recv.py
posix
[sudo] password for pond:
Message sent on Enable state data: bytearray(b'\x00\x00\x00\x00')
>>45
angle =  1215
pulse(hex) =  000004BF
DATA_Lh =  0x4
DATA_Ll =  0xbf
DATA_Hh =  0x0
DATA_Hl =  0x0
Traceback (most recent call last):
  File "/home/pond/.local/lib/python3.8/site-packages/can/message.py", line 98, in __init__
    self.data = bytearray(data)
TypeError: 'str' object cannot be interpreted as an integer

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/home/pond/Desktop/raptor_project-master/raptor_project-master/steering_yaw/scripts/can_send_recv.py", line 135, in <module>
    msg = can.Message(arbitration_id=0x06000001, data=[0x23, 0x02, 0x20, 0x01, DATA_Lh, 0x00, 0x00, 0x00])
  File "/home/pond/.local/lib/python3.8/site-packages/can/message.py", line 101, in __init__
    raise TypeError(err) from error
TypeError: Couldn't create message from [35, 2, 32, 1, '0x4', 0, 0, 0] (<class 'list'>)
pond@ck:~/Desktop/raptor_project-master/raptor_project-master$
```

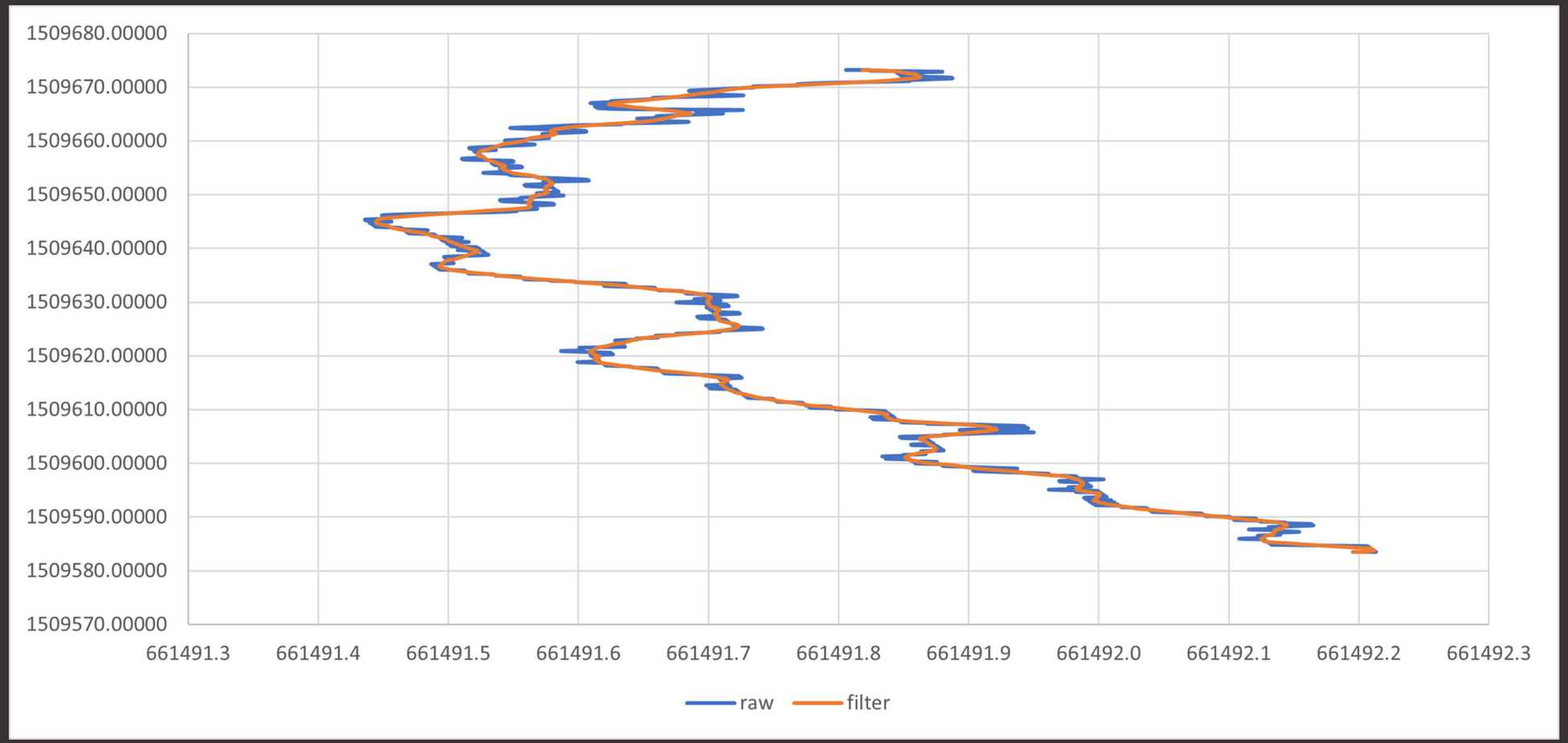
Problem

Steering Wheel

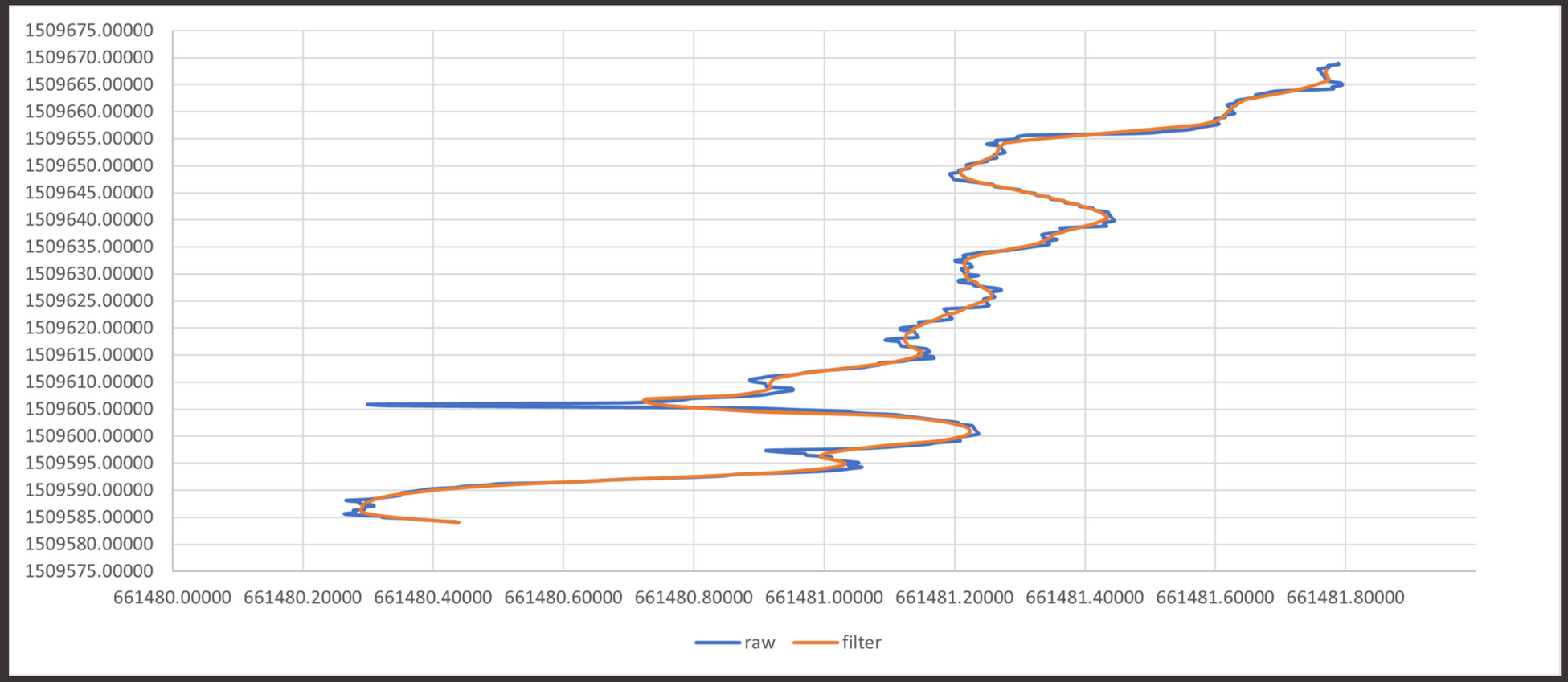
Overall Block Diagram



Result from Send-Received code



Result from Send-Received code



Result from Send-Received code

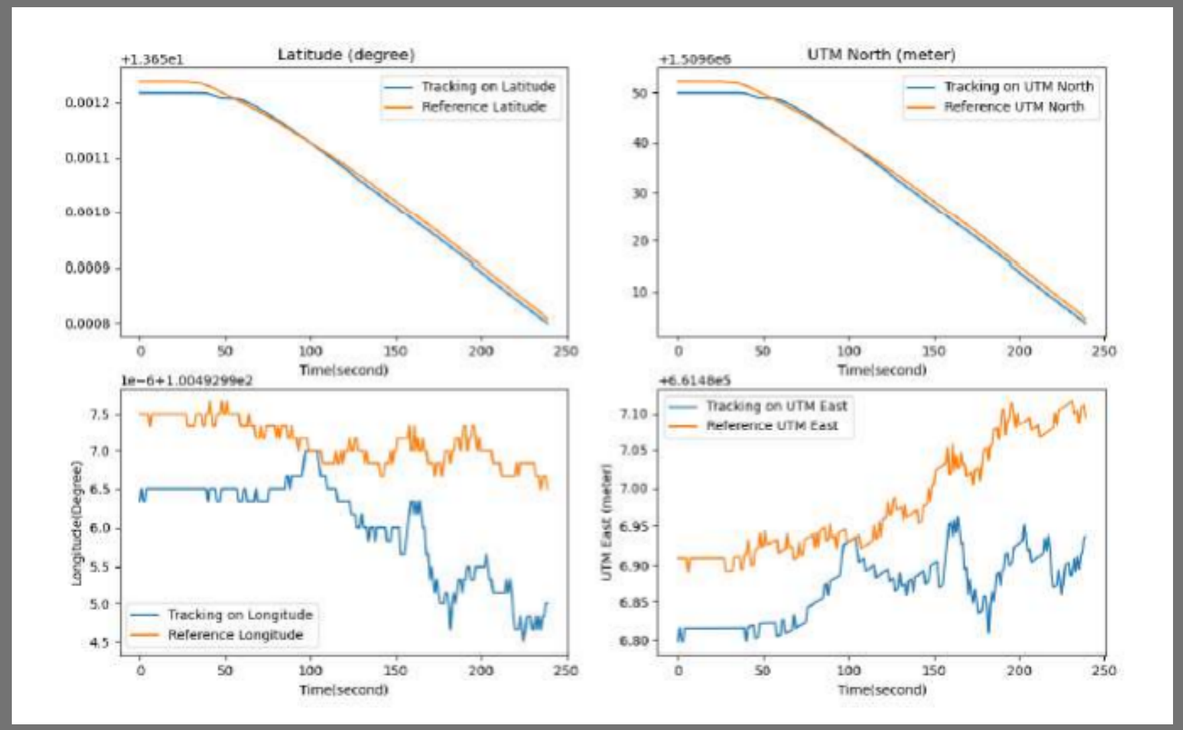
PID Controller

The General form of the PID control algorithm equation

$$u(t) = K_p e(t) + K_i \int_0^t e(t)d\tau + K_d \frac{de(t)}{dt}$$

PID Controller Tuning

PID tuning to reduce steering errors and steering angles



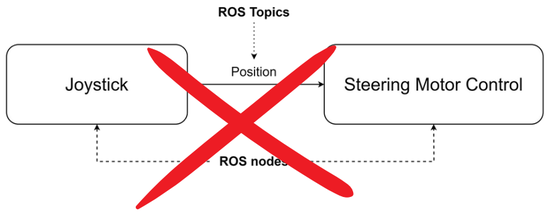
The graph compares the actual path of travel with the reference line starting from the center point

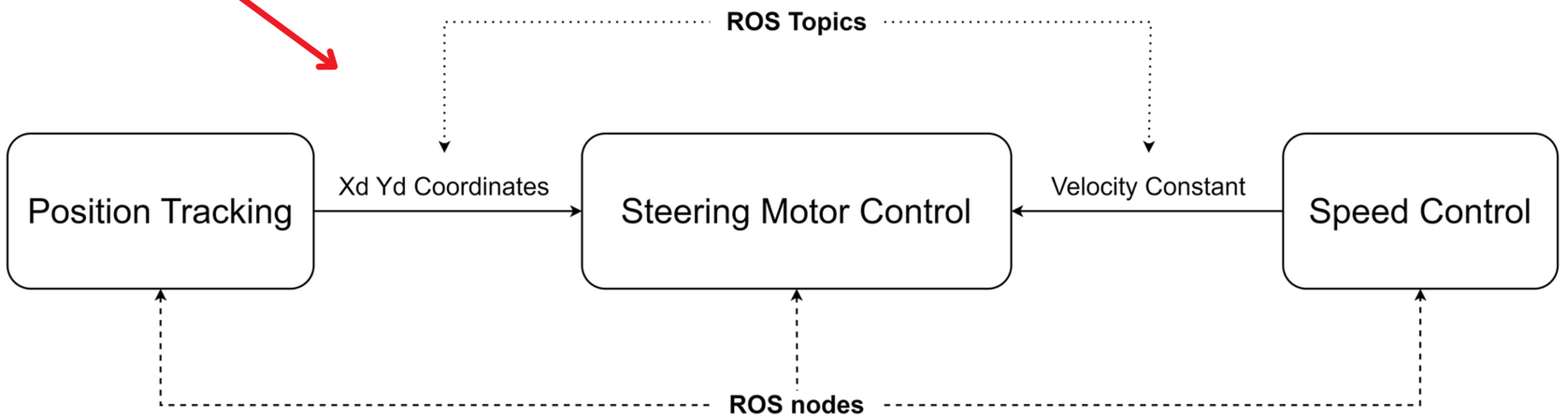
(Based on past reserch)

Robot Operating System

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications

Improvement





The Autonomous Vehicle: GNSS Technology Tracking Using Controller

Summary Of Different and Improvement

Last year

Tracking and Navigation

- 2 Antenna/Receiver Board and 2 RF Data link to sent calibration value from base to rover

Control System

Speed :

- Arduino Mega Board to control speed at 5km/hr

Steering :

- RS-232 Communication
- PID Controller and Bicycle Model

Robot Operating System

- for coding steering control with joystick

Current year

Tracking and Navigation

- use 1 Antenna/Receiver at rover and get calibration value from Internet for short Distance
- use 2 Antenna/Receiver for Around Soccer Feild

Control System

Speed :

- Arduino Mega Board to control speed at 5km/hr
- New Hardware Design

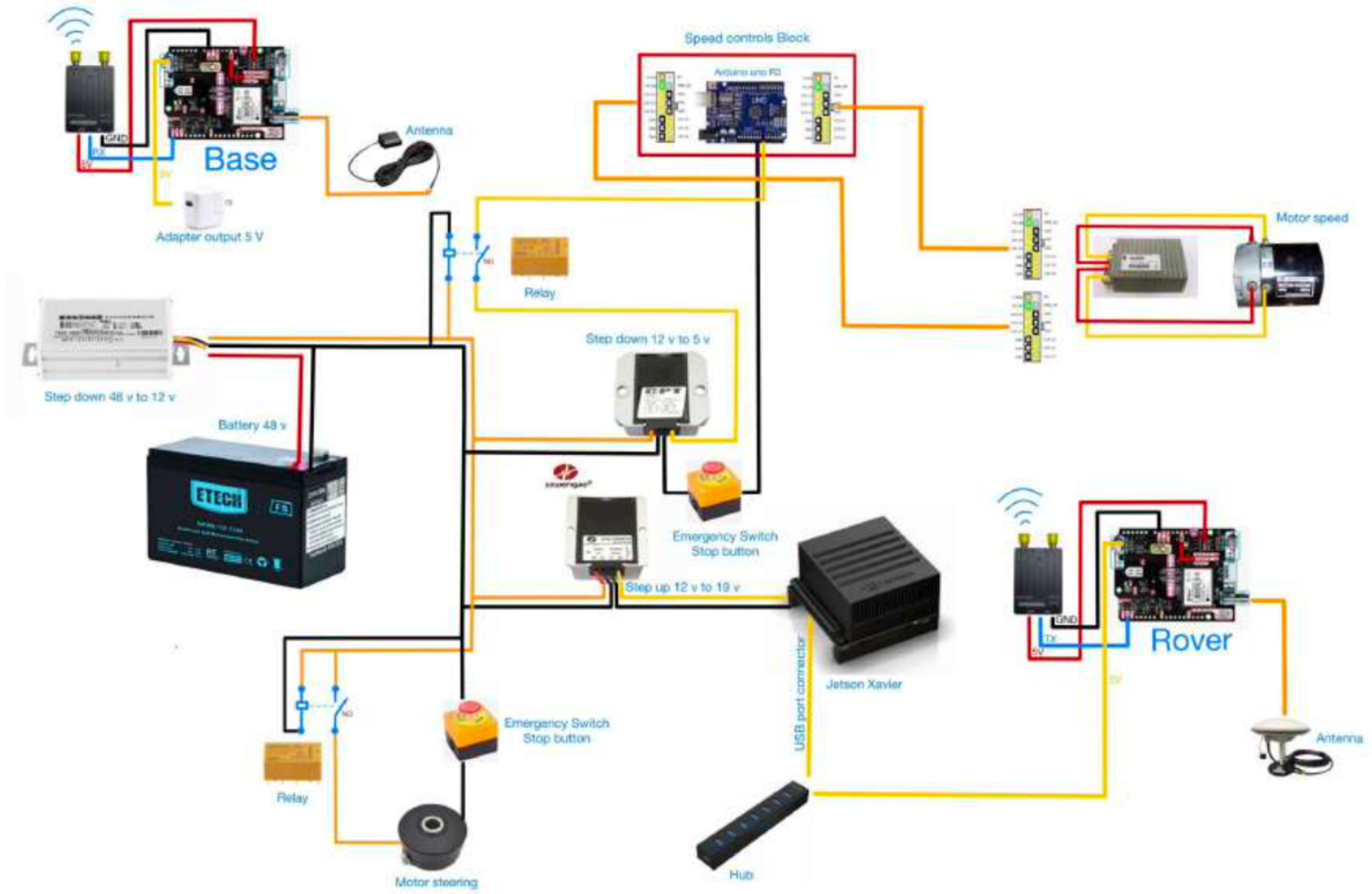
Steering :

- Can bus Communication
- Design new Steering Connector
- PID Controller tuning

Robot Operating System

- for coding and Implement all systems to ROS Node

Hardware Block Diagram



The Autonomous Vehicle: GNSS Technology Tracking Using Controller

SOFTWARE

SOLIDWORKS

ใช้ในการออกแบบในส่วนของ Steering part

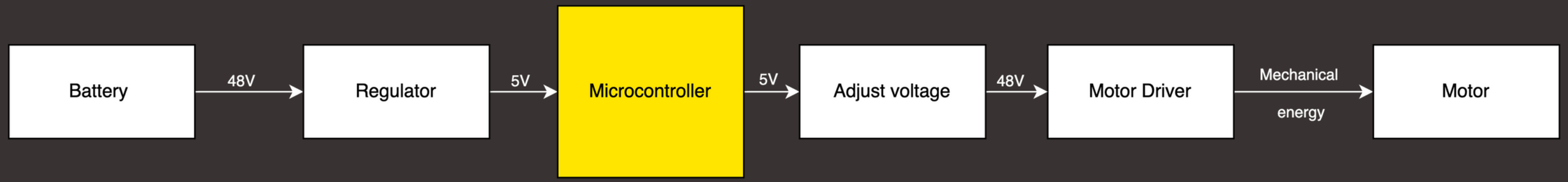
Program KEYA Electron
ใช้ในการควบคุมการทำงานของ Steering Motor

Program Ublox u-center
ใช้ในการ Set-up บอร์ด และรับค่าตำแหน่ง

SPEED CONTROL

Control Speed

การควบคุมความเร็วสามารถควบคุมผ่านการปรับแรงดันที่เข้ามาให้เป็นช่วงๆ หรือ การควบคุม PWM แต่ในการควบคุมความเร็วของรถไฟฟ้านี้ไม่สามารถทำได้โดยตรงเพราะระบบไม่รู้จักว่า PWM คืออะไร ดังนั้นจึงใช้วิธีการปรับค่า ตัวต้านทานและสร้างวงจรเพื่อมาควบคุมความเร็ว



Digital Potential

เป็นการนำอุปกรณ์ที่สามารถปรับค่าความต้านทานได้โดยการเขียนโปรแกรมควบคุม เช่นการใช้โปรแกรม Arduino

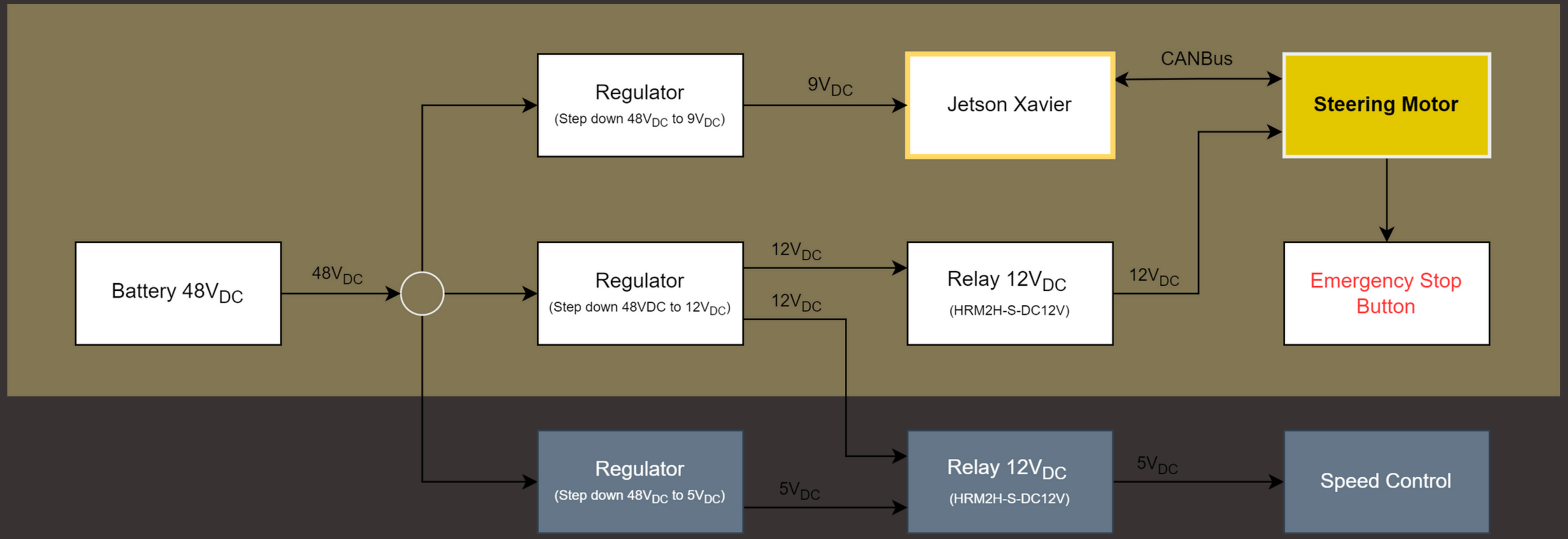
Regulator

ใช้ในการปรับลดแรงดันที่มาจาก Controller ของรถไฟฟ้า ก่อนที่จะเข้า Microcontroll เนื่องจากแรงดันที่เข้ามามีขนาด 48V แต่ microcontroller สามารถรองรับการทำงานได้ 7-12V

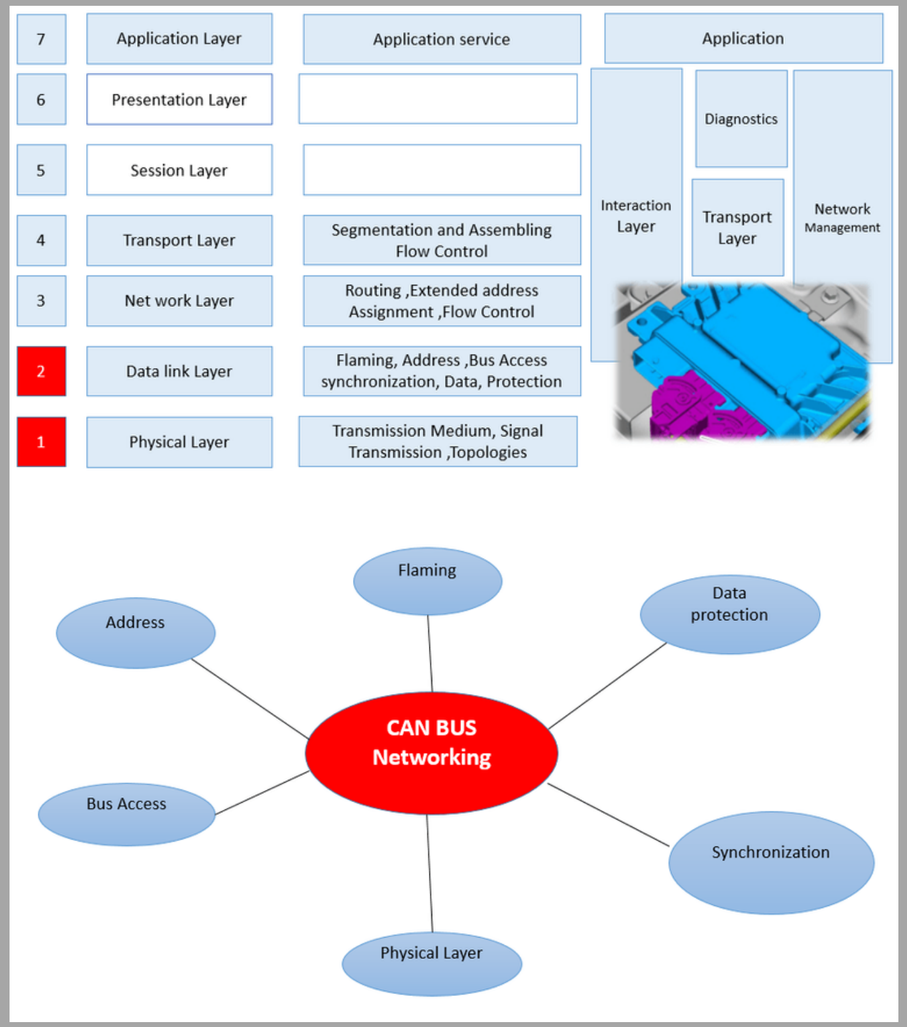
Adjust voltage

เป็นการปรับแต่งแรงดันให้มีขนาดเพิ่มจาก 5V เป็น 48V เพื่อให้ในการขับ Motor

STEERING MOTOR HARDWARE BLOCK DIAGRAM

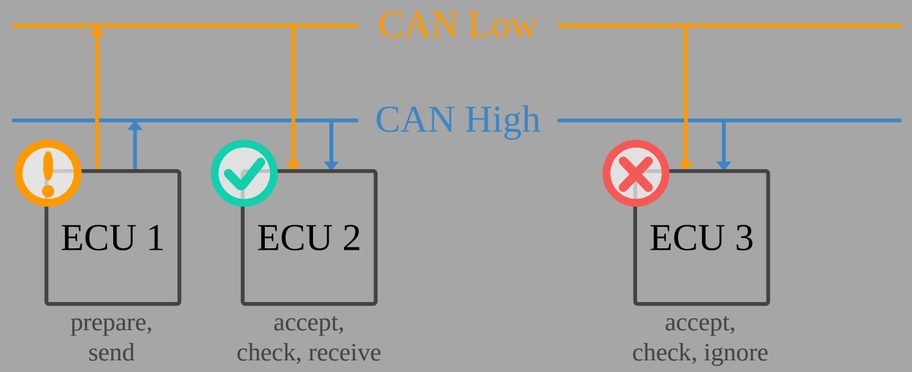


COMMUNICATE WITH CAN PROTOCOL



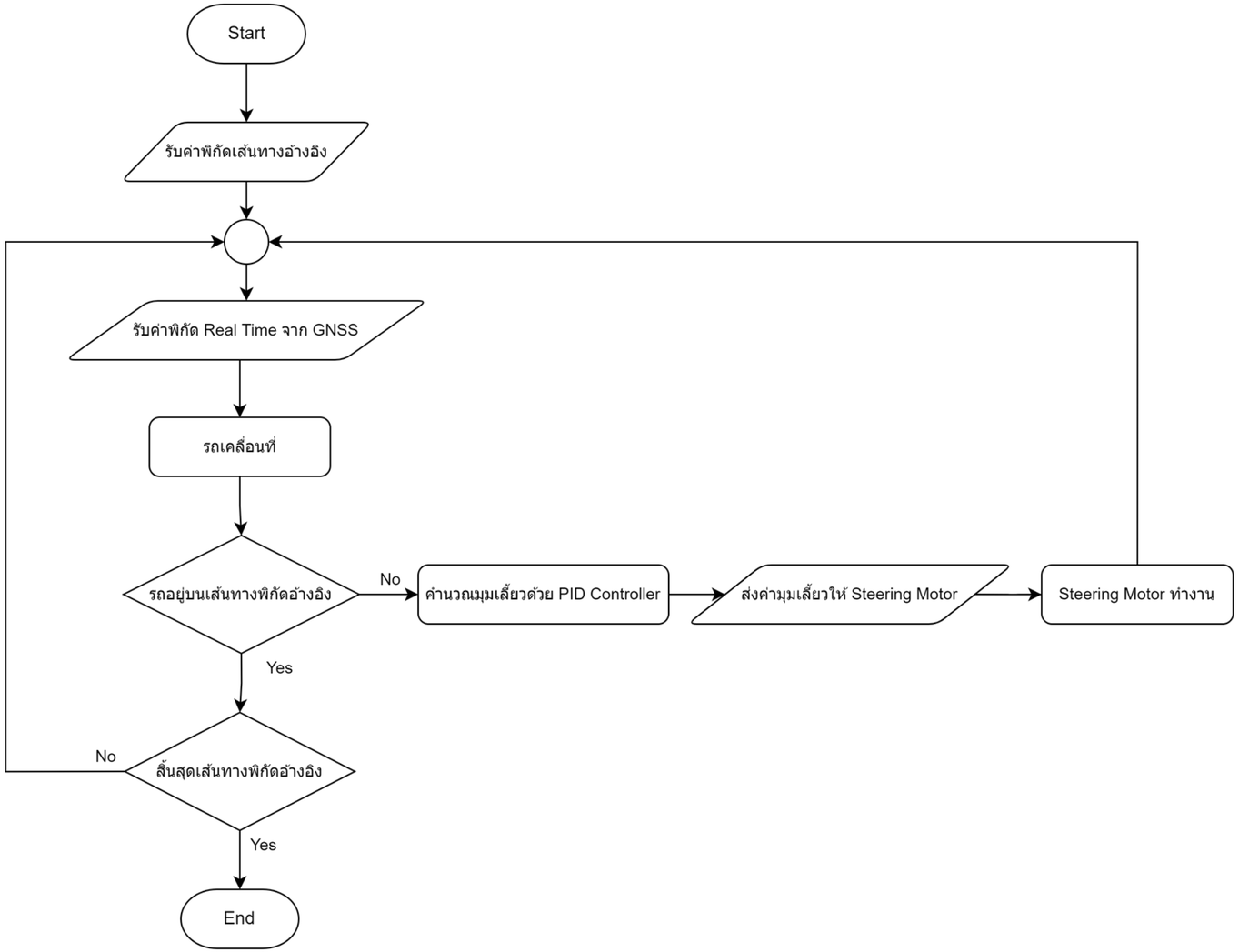
CAN Bus specification requirement OSI Layer 1 and 2

- สื่อสารแบบอนุกรมด้วยข้อมูลดิจิตอล
- ต้นทุนต่ำ
- ง่ายและสะดวกต่อการเชื่อมต่อ
- ข้อมูลมีความน่าเชื่อถือสูง
- ความเร็วสูงและแทบไม่มีข้อผิดพลาดในการส่งข้อมูลผ่านบัสที่มีความสอดคล้องของข้อมูลสูง
- เป็นระบบการสื่อสารบัสมาตรฐาน



The Autonomous Vehicle: GNSS Technology Tracking Using Controller

Main Flowchart



แผนดำเนินงาน	ตารางดำเนินงาน (เดือน)																							
	มิ.ย. - ก.ค.				ส.ค. - ก.ย.				ต.ค. - พ.ย.				ธ.ค - ม.ค				ก.พ. - มี.ค.				เม.ย. - พ.ค.			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
ศึกษาข้อมูลที่เกี่ยวข้องกับงานวิจัย																								
ศึกษาการเขียนโปรแกรมพื้นฐานภาษา Python และ C++																								
ศึกษาการทำงานและการใช้งานของระบบปฏิบัติการ Ubuntu																								
ศึกษา Middleware สำหรับควบคุมหุ่นยนต์																								
ศึกษาทฤษฎีการระบุตำแหน่งด้วยระบบ GNSS																								
ศึกษาการทำงานของ Speed controller Hardware																								
ศึกษาการทำงานของ Steering System																								
ศึกษาระบบการสื่อสารข้อมูล CAN bus																								
ทดสอบการระบุตำแหน่งด้วยเซนเซอร์ผ่านโปรแกรม U-center																								
ทดสอบและเปรียบเทียบผลการระบุตำแหน่งระหว่างแบบ RTK และ Network RTK																								
ศึกษาวงจรและทดสอบการทำงานของมอเตอร์ผ่านการสื่อสารข้อมูล RS-232																								
ออกแบบและวัดค่าวงจร Speed control box																								
ออกแบบการ Wiring สายไฟ																								
ปรับปรุง ออกแบบ Steering Part																								
ศึกษาและทดสอบการทำงานของ Software Speed control box																								
ทดสอบการส่งค่าตำแหน่งไปยัง Middleware																								
ทดสอบระบบควบคุมมุมเลี้ยวด้วย PID Controller ผ่านการ Simulate																								
ศึกษาและทดสอบการเชื่อมการทำงานของ CAN bus กับอุปกรณ์อื่น																								
ศึกษาการรับ-ส่งข้อมูลของแต่ละส่วน																								
ทดสอบการทำงานของ Middleware ROS																								
ทดสอบและแก้ไขการทำงานรวมของระบบ																								

Block Diagram