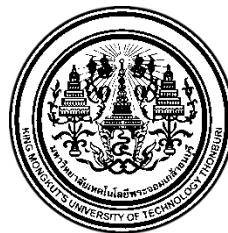


ระบบติดตามเดินทางการเคลื่อนที่ของรถยนต์ไฟฟ้า  
 โดยใช้ระบบการนำทางด้วยดาวเทียมและด้วยตัวควบคุมแบบป้อนกลับ

นายสุรยุทธ เดือนกว้าง  
 นางสาวชุติมณฑน์ เกษgar  
 นางสาวสรวิษ วนิชผล

โครงการนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต  
 สาขาวิชาวิศวกรรมระบบควบคุมและเครื่องมือวัด  
 ภาควิชาวิศวกรรมระบบควบคุมและเครื่องมือวัด คณะวิศวกรรมศาสตร์  
 มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี  
 ปีการศึกษา 2565



THE AUTONOMOUS VEHICLE: GNSS TECHNOLOGY TRACKING  
USING PID CONTROLLER

MR. SURAYUTH DUEWKWANG

MISS CHUTIMON KETKARN

MISS SORRAVEE WANICHPHOL

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING  
(CONTROL SYSTEMS AND INSTRUMENTATION ENGINEERING)

DEPARTMENT OF CONTROL SYSTEMS AND

INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI

2022

ระบบติดตามเส้นทางการเคลื่อนที่ของรถชนต์ไฟฟ้า  
โดยใช้ระบบการนำทางด้วยดาวเทียมและด้วยตัวควบคุมแบบป้อนกลับ

นายสุรยุทธ เดือนกว้าง  
นางสาวชุดิมณฑน์ เกษกการ  
นางสาวสรวิษ วนิชผล

โครงการนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต  
สาขาวิชาชีววิศวกรรมระบบควบคุมและเครื่องมือวัด  
ภาควิชาชีววิศวกรรมระบบควบคุมและเครื่องมือวัด คณะวิศวกรรมศาสตร์  
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี  
ปีการศึกษา 2565

**คณะกรรมการสอบโครงการ**

..... (รศ. ดร.เบญจมาศ พนมรัตน์รักษ์)	อาจารย์ที่ปรึกษาโครงการ กรรมการและอาจารย์ที่ปรึกษาโครงการ (ร่วม) (พศ. วุฒิชัย สิทธิอัจกร)
..... (พศ. ดร.สุศชาย บุญโട)	กรรมการ กรรมการ
..... (ดร.อิสสระพงศ์ คำนวนเครื่อ)	กรรมการ กรรมการ

ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

THE AUTONOMOUS VEHICLE: GNSS TECHNOLOGY TRACKING USING PID  
CONTROLLER

MR. SURAYUTH DUENKWANG

MISS CHUTIMON KETKARN

MISS SORRAVEE WANICHPHOL

A PROJECT IS SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF ENGINEERING  
(CONTROL SYSTEMS AND INSTRUMENTATION ENGINEERING)  
DEPARTMENT OF CONTROL SYSTEMS AND INSTRUMENTATION ENGINEERING  
FACULTY OF ENGINEERING  
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI  
2022

PROJECT COMMITTEE

..... PROJECT ADVISOR  
(ASSOC. PROF. DR.BENJAMAS PANOMRUTTANARUG)

..... PROJECT CO-ADVISOR  
(ASST.PROF.VUTTICHAI SITTIARTTAKORN)

..... COMMITTEE  
(ASST. PROF. DR.SUDCHAI BOONTO)

..... COMMITTEE  
(DR.ISSARAPONG KHUANKRUE)

หัวข้อโครงการ	ระบบติดตามเส้นทางการเคลื่อนที่ของรถยนต์ไฟฟ้าโดยใช้ระบบการนำทางด้วยดาวเทียมและด้วยตัวควบคุมแบบป้อนกลับ
หน่วยกิต	3
ผู้เขียน	นายสุรยุทธ์ เดือนกว้าง นางสาวชุดิมณฑน์ เกษgar นางสาวserviee วนิชผล
อาจารย์ที่ปรึกษา	รศ. ดร.เบญจมาศ พนรัตนรักษ์ ผศ. วุฒิชัย สิทธิอัฐกร
หลักสูตร	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมระบบควบคุมและเครื่องมือวัด
ภาควิชา	วิศวกรรมระบบควบคุมและเครื่องมือวัด
คณะ	วิศวกรรมศาสตร์
ปีการศึกษา	2565

### บทคัดย่อ

ระบบติดตามเส้นทางการเคลื่อนที่ของรถยนต์ไฟฟ้าแบบอัตโนมัติด้วยตัวควบคุมแบบป้อนกลับนี้ ถูกสร้างขึ้นเพื่อใช้ในการติดตามการเคลื่อนที่ตามเส้นทางอ้างอิงที่ได้กำหนดไว้ โดยรถยนต์ไฟฟ้าจะทำการเคลื่อนที่ เปลี่ยนแปลงมุมเลี้ยวและเปลี่ยนแปลงความเร็ว ตามเส้นทางอ้างอิงดังกล่าว ซึ่งลักษณะการทำงานจะให้ผู้ใช้งานทำการขับเคลื่อนรถยนต์ไฟฟ้าแบบมั่นคงด้วยการควบคุมพวงมาลัย และบันทึกเส้นทางอ้างอิงจากการรังวัดพิกัดด้วยดาวเทียมแบบจลน์ในรูปแบบแบบสองมิติ จากนั้นจะนำเส้นทางอ้างอิงที่ได้และตำแหน่งพิกัดปัจจุบันของรถยนต์ไฟฟ้าไปในการควบคุมพวงมาลัยและควบคุมความเร็วของรถให้สามารถเคลื่อนที่อยู่ในเส้นทางอ้างอิงได้ โดยเส้นทางอ้างอิงในโครงการนี้ จะใช้บริเวณรอบสนามฟุตบอลของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ซึ่งลักษณะของเส้นทางอ้างอิงจะประกอบไปด้วย เส้นทางอ้างอิงในลักษณะทางตรงและทางโค้งรวมระยะทาง เคลื่อนที่ทั้งสิ้น 522 เมตร

คำสำคัญ: การสำรวจรังวัดด้วยดาวเทียมแบบจลน์/ ตัวควบคุม/ มอเตอร์มูมเลี้ยว/ รถยนต์อัตโนมัติ

Project Title	The Autonomous Vehicle: GNSS Technology Tracking Using PID Controller
Project Credits	3
Candidate	Mr. Surayuth Duenkwang Miss Chutimon Ketkarn Miss Sorravee Wanichphol
Project Advisors	Assoc. Prof. Dr.Benjamas Panomruttanarug Asst. Prof. Vuttichai Sittiarttakorn
Program	Bachelor of Engineering
Field of Study	Control system and Instrumentation Engineering
Department	Control system and Instrumentation Engineering
Faculty	Engineering
Academic Year	2022

### Abstract

The tracking system of the automation electric vehicle with PID controller was created to track the reference and navigate the vehicle along that reference. The vehicle's movement consists of changing the turning angle, and the velocity follows the reference path. The working principle starts with the user recording the reference coordinates by driving the vehicle manually with the real-time kinematic satellite in two dimensions. After that, control the steering and the velocity based on the distance error between the reference path and the current vehicle's position. The reference path in this project was conducted on the road around the King Mongkut's University of Technology Thonburi soccer field, including the straight path and the curve path, with a total distance of 552 meters.

Keywords: Autonomous vehicles/ GNSS RTK/ Lateral control/ PID Controller

## กิตติกรรมประกาศ

โครงการเรื่องระบบติดตามเส้นทางการเคลื่อนที่ของรถยกไฟฟ้า โดยใช้ระบบการนำทางด้วยดาวเทียมและด้วยตัวความคุณแบบป้อนกลับ สามารถสำเร็จได้เนื่องจากได้รับคำปรึกษาจากอาจารย์ภาควิชาวิศวกรรมระบบควบคุมและเครื่องมือวัดมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรีชั้น รศ. ดร.เบญจมาศ พนรัตนรักษ์ ได้ให้คำปรึกษาระหว่างการดำเนินการสร้างโครงการ ทั้งในเรื่องทฤษฎีความรู้ต่างๆ ที่จำเป็นต้องใช้ในการดำเนินการสร้างโครงการนี้ และให้คำปรึกษาในการแก้ปัญหาต่างๆ ที่พบเจอระหว่างดำเนินงาน ดังนั้นผู้จัดทำจึงกราบขอบคุณ รศ. ดร.เบญจมาศ พนรัตนรักษ์เป็นอย่างสูง

## สารบัญ

หน้า

บทคัดย่อภาษาไทย	๑
บทคัดย่อภาษาอังกฤษ	๒
กิตติกรรมประกาศ	๓
สารบัญ	๔
รายการตาราง	๘
รายการรูปประกอบ	๙

### **บทที่**

<b>1. บทนำ</b>	<b>1</b>
1.1 ความสำคัญและที่มาของโครงการ	1
1.2 วัตถุประสงค์ของโครงการ	2
1.3 ประโยชน์และผลที่คาดว่าจะได้รับจากโครงการ	2
1.4 ขอบเขตของโครงการ	2
1.5 ขั้นตอนการดำเนินงาน	3
<b>2. ภาพรวมโครงการและทฤษฎีต่างๆที่ใช้</b>	<b>4</b>
2.1 ภาพรวมการทำงานของระบบห้องแม่	4
2.2 ภาพรวมการทำงานอาร์ดแวร์	5
2.3 ภาพรวมการทำงานซอฟต์แวร์	6
2.4 ภาพรวมและทฤษฎีต่างๆ ของระบบระบบทุ่มคำแหง	11
2.5 ภาพรวมและทฤษฎีต่างๆ ของระบบควบคุมความเร็วให้คงที่	40
2.6 ภาพรวมและทฤษฎีต่างๆ ของระบบควบคุมตำแหน่งในการเคลื่อนที่ของรถ	49
2.7 ภาพรวมของระบบ Robot Operating System	74
<b>3. การออกแบบอาร์ดแวร์และซอฟต์แวร์</b>	<b>86</b>
3.1 การออกแบบและติดตั้งอุปกรณ์	86
3.2 การออกแบบอาร์ดแวร์และซอฟต์แวร์ของระบบระบบทุ่มคำแหง	95
3.3 การออกแบบอาร์ดแวร์และซอฟต์แวร์ของระบบควบคุมความเร็ว	128

สารบัญ (ต่อ)	หน้า
3.4 การออกแบบhardtแวร์และซอฟต์แวร์ของระบบควบคุมตำแหน่งในการเคลื่อนที่	142
3.5 การออกแบบซอฟต์แวร์ของ Robot Operating System	158
<b>4. ผลการทดลอง</b>	<b>163</b>
4.1 ผลการทดลองของระบบระบุตำแหน่ง	163
4.2 ผลการทดลองของระบบควบคุมความเร็ว	192
4.3 ผลการทดลองของระบบระบุควบคุมตำแหน่งในการเคลื่อนที่	202
4.4 ผลการทดลองของ Robot Operating System	230
<b>5. สรุป</b>	<b>243</b>
5.1 สรุปผลการทดลอง	234
5.2 ปัญหาที่พบ	235
5.3 วิธีการแก้ปัญหาและคำแนะนำ	237
<b>เอกสารอ้างอิง</b>	<b>241</b>
<b>ภาคผนวก</b>	<b>243</b>
ก โภคการระบุตำแหน่งปัจจุบันของรถผ่าน Robot Operating System	244
ข โภคคำนวณความเร็วปัจจุบันจากพิกัดตำแหน่งผ่าน Robot Operating System	249
ค โภคแสดงผลการทำงานของระบบควบคุมความเร็วผ่าน Robot Operating System	255
ง โภคควบคุมความเร็วให้คงที่	260
จ โภคการควบคุมการติดตามเส้นทางอ้างอิงผ่าน Robot Operating System	267
ฉ โภคควบคุมการทำงานมอเตอร์พวงมาลัยผ่าน Robot Operating System	297
<b>ประวัติผู้จัดทำโครงการ</b>	<b>299</b>

## รายการตาราง

ตาราง	หน้า
1.1 ตารางแสดงแผนผังการดำเนินงาน (Gantt Chart) ในปีการศึกษาที่ 2565	3
2.1 ตารางแสดงรายละเอียดต่างๆของดาวเทียมในระบบ GNSS	12
2.2 ตารางแสดงค่าความคลาดเคลื่อนในการรังวัดดาวเทียมระบบ GNSS	24
2.3 ตารางแสดงรายละเอียดของค่าต่างๆใน GPS Quality Indicator	32
2.4 แสดงค่าความแตกต่างของคลาดเคลื่อนและลองจิจูด เมื่อมีการเปลี่ยนแปลง คลาดเคลื่อน 1 องศา	55
2.5 ค่าความสัมพันธ์ระหว่างความเร็วของเครื่อข่ายและความยาวสายสัญญาณ	64
4.1 ตารางแสดงลำดับเส้นทางและระยะทางรอบสนามกีฬา	181
4.2 แสดงผลการทดลองการควบคุมความเร็ว โดยการปรับค่า Digital Potentiometer	194
4.3 ตัวอย่างผลการทดลองการเคลื่อนที่ด้วยความเร็วคงที่ 5 Km/hr รอบสนาม ฟุตบอล	198
4.4 ตัวอย่างผลการทดลองการเคลื่อนที่ด้วยความเร็วคงที่ 3 Km/hr และ 1 Km/hr รอบสนามฟุตบอล	201
4.5 การทดสอบการควบคุมมอเตอร์ในรอบที่ 1	206
4.6 การทดสอบการควบคุมมอเตอร์ในรอบที่ 2	210
4.7 การทดสอบการควบคุมมอเตอร์ในรอบที่ 3	211
4.8 การทดสอบการควบคุมมอเตอร์ในรอบที่ 4	212
4.9 การทดสอบการควบคุมมอเตอร์ในรอบที่ 5	213
4.10 การทดสอบการควบคุมมอเตอร์ในรอบที่ 6	214

## รายการรูปประกอบ

รูป	หน้า
2.1 ภาพรวมของรถกอล์ฟ	4
2.2 Diagram แสดงภาพรวมการทำงานของระบบ	4
2.3 ภาพรวมของจรทั้งหมดที่มีการเขียนต่อให้รถกอล์ฟสามารถเคลื่อนที่ได้อัตโนมัติ	5
2.4 แสดงโปรแกรมภาษา ROS (Robot Operating System)	6
2.5 แสดงโปรแกรมภาษา C+ (C Programming Language)	7
2.6 แสดงโปรแกรมภาษา Python (Python Programming Language)	7
2.7 แสดงโปรแกรม Visual Studio Code	7
2.8 แสดงโปรแกรม Arduino IDE	8
2.9 แสดงโปรแกรม RFD tools	8
2.10 แสดงโปรแกรม U center	9
2.11 แสดงโปรแกรม SolidWorks	9
2.12 InnoMakerUSB2CAN Software	10
2.13 kyMotor Control Utility Software	10
2.14 ภาพรวมและลำดับการทำงานของระบบระบุตำแหน่ง	11
2.15 การเปรียบเทียบระหว่างระบบ GPS และ GNSS	11
2.16 สถาปัตยกรรมของระบบหาพิกัดตำแหน่งด้วยดาวเทียม GNSS	12
2.17 จำนวนและวงโคจรของดาวเทียมในระบบ GNSS	13
2.18 สถานีควบคุมภาคพื้นดินของดาวเทียมในระบบ GNSS ข้อมูล ณ ปี พ.ศ. 2559	14
2.19 ตัวอย่างการสถาปัตยกรรมของระบบหาพิกัดตำแหน่งด้วยดาวเทียม GNSS	14
2.20 การอ้างอิงระยะทางระหว่างโนนดอ้างอิงและตำแหน่งของวัตถุ	15
2.21 ภาพอธิบายกระบวนการ Trilateration เมื่อนำมาใช้กับการหาพิกัดแบบดาวเทียม	15
2.22 ภาพการเปรียบเทียบสัญญาณของรหัสเพื่อหาเวลาที่คลื่นเดินทางจากดาวเทียมมาถึงเครื่องรับ	16
2.23 ภาพแสดงการวัดเฟสของคลื่นส่ง	18
2.24 ภาพอธิบายการเกิด Ephemeris Error	20

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
2.25 ภาพอธิบาย GPS Clock Error	20
2.26 ภาพอธิบายการเกิด Atmospheric Error	21
2.27 ภาพอธิบายการเกิด Multipath Error	22
2.28 ภาพแสดงการเกิด Cycle Slip Error	22
2.29 ภาพอธิบายการวางแผนของดาวเทียมเพื่อให้ได้ค่า DOP ที่ดี	24
2.30 ภาพตัวอย่างเสาอากาศ Antenna ในการรับสัญญาณ GNSS	25
2.31 ภาพตัวอย่างวงจรรับสัญญาณ GNSS	26
2.32 ภาพตัวอย่างเครื่องรับสัญญาณดาวเทียม GNSS ชนิดทั่วไป	26
2.33 ภาพตัวอย่างเครื่องรับสัญญาณดาวเทียม GNSS ชนิด RTK	27
2.34 ภาพแสดงเทคนิคการรังวัดแบบสอดคล้อง	28
2.35 ภาพแสดงเทคนิคการรังวัดแบบจลน์	29
2.36 ภาพแสดงเทคนิคการรังวัดแบบ Network RTK	29
2.37 ตัวอย่างรูปแบบประโยคของโปรโตคอล NMEA	30
2.38 ตัวอย่างรูปแบบประโยคของโปรโตคอล NMEA ในโปรแกรม U-center	34
2.39 เส้นสมมติในระบบพิกัดภูมิศาสตร์	35
2.40 ตำแหน่งพิกัดของหอสังเกตการณ์ทางดาราศาสตร์ เมืองกรีนิช	35
2.41 มุมและตารางต่างๆในการวัดค่าพิกัดแบบละติจูดและลองติจูด	37
2.42 แผนที่เขตเวลาของโลก	38
2.43 UTM grid zone	39
2.44 แสดงตัวควบคุมและประมาณผลการทำงานของรถกอล์ฟ	41
2.45 แสดง Wiring Diagram ของตัวควบคุมและประมาณผลการทำงานของรถ กอล์ฟ	41
2.46 แสดงสวิตช์การทำงาน Run/Tow	42
2.47 แสดงสวิตช์เลือกการทำงานของทิศทาง (Switch Direction)	42
2.48 แสดง Pin of controller ของรถ	43
2.49 แสดง Pin ที่เชื่อมกับ Hall Sensor	43

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
2.50 แสดงตำแหน่งสายแรงดันที่ต่อเข้ามกับแบตเตอรี่	43
2.51 แสดง Block Diagram ภาพรวมการทำงานของระบบควบคุมความเร็ว	44
2.52 แสดงกลไกการออกแบบวงจรการทำงาน	45
2.53 แสดงกลไกการออกแบบวงจรการทำงาน (Diagram)	45
2.54 แสดงตัวอย่างการควบคุมกระแสแรงดันแบบ Pulse Width Modulation	45
2.55 Block Diagram การควบคุมความเร็วโมเตอร์ไฟฟ้าแบบระบบปิด	46
2.56 Block Diagram ของตัวควบคุมแบบ PID	46
2.57 แสดงตำแหน่งการหาระยะห่างระหว่างจุดสองจุดบนพื้นผิวรถกลม	47
2.58 โคลอแกรมของระบบควบคุมแบบปิด (Closed-Loop Control System)	48
2.59 แสดง PID Controller Block Diagram	50
2.60 บล็อกโคลอแกรมการนำตัวควบคุมแบบพีไอดีมาประยุกต์ใช้ในการควบคุม มุมเลี้ยว	53
2.61 แสดงการประยุกต์สูตรพีทากอรัส	54
2.62 แสดงการประยุกต์สูตรพีทากอรัสสำหรับการหาระยะทางระหว่างสองพิกัด	54
2.63 สูตรในการหาค่าความชันของเส้นตรง	56
2.64 สูตรที่มีการประยุกต์ใช้เพื่อกำหนณหาค่า Cross Track Error	56
2.65 Kinematic Bicycle Model of The Vehicle	57
2.66 Bicycle Model Block Diagram	57
2.67 โคนามิกของyanpathan	59
2.68 Product Details of Motor	60
2.69 Steering Wheel Steering Gear Motor	60
2.70 ค่าพารามิเตอร์ของมอเตอร์	61
2.71 แสดงรายละเอียดการรับส่งสัญญาณข้อมูลผ่าน CAN bus	62
2.72 กราฟความสัมพันธ์ระหว่าง Data Rate – Line Length	64
2.73 การเชื่อมต่อระหว่างส่วนควบคุมด้วย CAN bus	66
2.74 แรงดันไฟฟ้าในสาย CAN High (CANH) และ CAN Low (CANL)	66
2.75 แสดงส่วนประกอบของ CAN Node	67
2.76 แสดงโครงสร้าง Standard Frame (ที่มา; csselectronics.com)	68

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
2.77 แสดงส่วนประกอบต่างๆ ของ CAN bus data flame	68
2.78 รูปร่างสัญญาณ CAN bus โดยที่ใช้เครื่องมือวัดสัญญาณ	69
2.79 ตัวอย่างรายละเอียดแพลตฟอร์มที่เหมาะสมกับ ROS เวอร์ชัน Lunar	74
2.90 ตัวอย่างส่วนประกอบต่างๆ ที่สำคัญภายใน Package กรณีใช้ภาษา C+	83
2.91 ตัวอย่างส่วนประกอบต่างๆ ที่สำคัญภายใน Package กรณีใช้ภาษา Python	84
2.92 ตัวอย่างการเรียงลำดับไฟล์เดอร์ภายใน Workspace	84
2.93 แหล่งข้อมูลต่างๆ ของ ROS Community Level	85
3.1 แสดงโครงสร้างภาพรวมของรถ	86
3.2 แสดงโครงสร้างของรถออล์ฟจากด้านข้าง และการติดตั้งอุปกรณ์	87
3.3 แสดงโครงสร้างของรถออล์ฟจากด้านบน และการติดตั้งอุปกรณ์	87
3.4 แสดง Block Diagram ขั้นตอนการสร้างชิ้นงานอุปกรณ์สำหรับติดตั้ง มอเตอร์พวงมาลัย	88
3.5 แสดงตัวอย่างชิ้นงานอุปกรณ์ไฟล์ CAD จากการออกแบบใหม่สำหรับติดตั้ง มอเตอร์พวงมาลัย	89
3.6 ชิ้นงานอุปกรณ์ไฟล์ Drawing จากการออกแบบใหม่สำหรับติดตั้งมอเตอร์ พวงมาลัย	89
3.7 ชิ้นงาน 3D Print อะไหล่คล้องต่อกับพวงมาลัยและรถ	90
3.8 ชิ้นงานโลหะสำหรับใช้จริ่ง	90
3.9 วงจรที่เป็นตัวควบคุมการเลือกการทำงาน (Switch Controller Box)	91
3.12 แสดงศูนย์กลางการเชื่อมต่อ Hub	93
3.13 แสดงหน้าจอที่ใช้ในการแสดงผล	93
3.14 แสดงอุปกรณ์ที่ใช้ในการควบคุมการสั่งงานกับ Jetson Xavier	93
3.15 แสดงอุปกรณ์ที่ใช้ในการควบคุมการสั่งงานกับ Jetson Xavier	93
3.16 แสดงอุปกรณ์ปรับขนาดแรงดัน	94
3.17 แสดงอุปกรณ์ที่ใช้กระจายแรงดันไปยังอุปกรณ์ต่างๆ (Terminal Block Connect)	94
3.18 แสดงตัวอย่างเสารับสัญญาณของ GPS sensor	95

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
3.19 ส่วนประกอบต่างๆ ในระบบตรวจจับพิกัดและนำทางรถแบบ Real Time Kinematic - RTK	96
3.20 ตัวอย่างเสาของ Base Station	96
3.21 GNSS Antennas รุ่น ANN-MB series L1/L2 multi-band, high precision	97
3.22 บอร์ด ArduSimple SimpleRTK2B ZED-F9P	97
3.23 Radio Link RFD900A Telemetry DATA LINK	98
3.24 ตัวอย่างรถกล้องไฟฟ้าของ Rover Station	98
3.25 Antenna ArduSimple Antenna Calibrated Survey GNSS Triple band + L-band antenna (IP67)	99
3.26 ส่วนประกอบต่างๆ ในระบบตรวจจับพิกัดและนำทางรถแบบ Network RTK	99
3.27 ตัวอย่างการใช้งานโปรแกรม U-center เชื่อมต่อกับ GNSS Module	100
3.28 การคืนหาพิกัดค่า Langevin และลองติจูดที่ได้จากโปรแกรม U-center	101
3.29 การเชื่อมต่อ Board Receiver กับ Computer	101
3.30 หน้าโปรแกรมเริ่มต้นในการใช้งาน U-center	102
3.31 การเลือก Com port ในการเชื่อมต่อกับ Base Receiver	102
3.32 หน้าโปรแกรม U-center หลังจากเชื่อมต่อกับ Base Receiver	103
3.33 การเลือกใช้ Baud rate ที่มากที่สุดที่ ArduSimple RTK2B ZED-F9P รับได้	103
3.34 เริ่มการตั้งค่าโดยการเปิดหน้าต่าง Generation Configuration View	104
3.39 การเปลี่ยนค่าสั่งในการตั้งค่าเป็น UART2 สำหรับใช้ในการทดลอง	106
3.40 ตัวอย่างการเปลี่ยนค่า Baud rate ของ UART1 แบบ Manual ก่อนส่งข้อมูล	107
3.41 ตัวอย่างการเปลี่ยนค่า Baud rate ของ UART1 แบบ Manual หลังส่งข้อมูล	107
3.42 ตัวอย่างการเปลี่ยน Protocol Input ของ UART1 แบบ Manual ก่อนส่งข้อมูล	108
3.43 ตัวอย่างการเปลี่ยน Protocol Input ของ UART1 แบบ Manual หลังส่งข้อมูล	108
3.44 การ Clear Lists ใน Key Name	109
3.45 ตัวอย่างการเลือกใช้ Base Configuration file ของ UART1	109
3.46 ตัวอย่างการเลือกใช้ Base Configuration file ของ UART1 ลงใน Key Name สำเร็จ	110
3.47 ตัวอย่างการอพโหลดการตั้งค่า Base Configuration file ของ UART1 สำเร็จ	110

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
3.48 การตรวจสอบการตั้งค่าหลังจากอัพโหลด Configuration File UART1	111
3.49 การตั้งค่ารีเซ็ต Receiver เป็นค่าเริ่มต้น	111
3.50 การตรวจสอบการตั้งค่าหลังจากรีเซ็ต Receiver	112
3.51 ตัวอย่างการเลือกใช้ Base Configuration file ของ UART2	112
3.52 การตรวจสอบการตั้งค่าหลังจากอัพโหลด Configuration File UART2	113
3.53 การเปลี่ยน Baud rate ของ UART2 แบบ Manual	113
3.54 การตั้งค่า Protocol และ Baud rate แบบ Manual ผ่าน Configuration view	114
3.55 การตั้งค่า Measurement Period ของทุก Time Source	114
3.56 การกำหนดเวลาและความละเอียดในการหาพิกัดที่ต้องของ Base Receiver	115
3.57 หน้าต่าง Message View ที่แสดงสถานะการ Survey-in	115
3.58 การเลือกใช้ NTRIP เพื่อรับค่าปรับแก้จาก CORS	116
3.59 ตัวอย่างการเชื่อมต่อ NTRIP ไม่สำเร็จเนื่องจากไม่ได้เชื่อมต่อ Internet	116
3.60 การใส่ Username และ Password ที่ขอรับจากการที่คืน	117
3.61 สถานะ NTRIP Client จะเปลี่ยนเป็นสีเขียวหลังจากเชื่อมต่อสำเร็จ	117
3.62 Continuous Operation Reference Station บริเวณใกล้เคียงกับมหาวิทยาลัย	118
3.67 หน้าโปรแกรม U-center หลังจากเชื่อมต่อกับ Rover Receiver	120
3.68 การเลือกใช้ Rover Configuration file	121
3.69 รายละเอียดการตั้งค่าต่างๆภายใน Rover Configuration file	121
3.70 การเปลี่ยนค่า Baud rate ของ UART2	122
3.71 การตั้งค่า Protocol และ Baud rate แบบ Manual ผ่าน Configuration view	122
3.72 การตั้งค่า Measurement Period ของทุก Time Source	123
3.73 การบันทึกการตั้งค่าทั้งหมดใน Receiver	123
3.74 การแสดงสถานะ FIXED หลังจากที่เชื่อมต่อกับ Base Receiver	124
3.75 ตัวอย่างการแสดงสถานะ TIME ของ Base และ FIXED ของ Rover	124
3.76 การเลือกใช้ NTRIP เพื่อรับค่าปรับแก้จาก CORS	125
3.77 ตัวอย่างการเชื่อมต่อ NTRIP ไม่สำเร็จเนื่องจากไม่ได้เชื่อมต่อ Internet	125
3.78 การใส่ Username และ Password ที่ขอรับจากการที่คืน	126
3.79 สถานะ NTRIP Client จะเปลี่ยนเป็นสีเขียวหลังจากเชื่อมต่อสำเร็จ	126

## รายการรูปประกอบ (ต่อ)

รูป	หน้า	
3.80	Continuous Operation Reference Station บริเวณไก่คีบกับมหาวิทยาลัย	127
3.81	ผังงานการอ่านค่าพิกัดและส่งค่าพิกัดไปยังระบบต่างๆ	127
3.82	แสดง Circuit Diagram ระบบควบคุมความเร็วแบบ Auto	129
3.83	แสดง Circuit Diagram ควบคุมความเร็วให้คงที่ในโหมดการทำงานแบบ Auto	129
3.84	วงจรควบคุมความเร็วให้คงที่ในโหมดการทำงานแบบ Auto	130
3.85	ตัวประมวลผลที่ใช้ในการควบคุมความเร็วให้เคลื่อนที่คงที่	130
3.86	ตัวอย่างอุปกรณ์ Relay	131
3.87	ลักษณะการใช้งานของอุปกรณ์	131
3.88	ตัวอย่างอุปกรณ์อิเล็กทรอนิกส์ NDY4805C	132
3.89	ลักษณะการทำงานของอุปกรณ์อิเล็กทรอนิกส์ TLP3545	132
3.94	ผังการออกแบบซอฟต์แวร์ Function การรับค่า Lat/Long	134
3.95	ผังการออกแบบซอฟต์แวร์ Function การนับเวลา	135
3.96	ผังการออกแบบซอฟต์แวร์ Function การคำนวณระยะห่างระหว่างพิกัด	135
3.97	ผังการออกแบบซอฟต์แวร์ Function การคำนวณความเร็ว	135
3.98	ผังการออกแบบซอฟต์แวร์ Function การส่งค่าไป Node อื่น	135
3.99	แสดงผังการทำงานการควบคุมการเคลื่อนที่ของรถโดยการปรับค่า Digital Potentiometer	136
3.100	ผังการทำงานของ Function นับ Encoder1	137
3.101	ผังการทำงานของ Function นับ Encoder2	137
3.102	ผังการทำงานของ Function ควบคุมการเคลื่อนที่	138
3.103	ผังการทำงานของ Function ควบคุมการทำงานของ Digital Poten	138
3.104	แสดงผังการควบคุมความเร็วรถแบบระบบปิด	139
3.105	ผังการทำงานของ Function การรับข้อมูลจาก Node อื่น	140
3.106	ผังการทำงานของ Function การควบคุมความเร็วผ่าน PID Controller	140
3.107	ผังการทำงานของ Function การควบคุมการเคลื่อนที่ของรถ	141
3.108	Circuit Design for Steering Motor Control	142
3.109	การควบคุมมอเตอร์ด้วยการเชื่อมต่อแบบต่างๆ	143

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
3.110 การติดตั้งโปรแกรม kyMotor Control Utility	144
3.111 แสดงสถานการณ์สื่อสารสำเร็จ	144
3.112 แสดงการเชื่อมต่อระหว่างตัวควบคุมกับซอฟต์แวร์	145
3.113 แสดงวิธีการ WRITE ข้อมูลหลังจากการปรับตั้งค่าทุกรุ่น	146
3.114 แสดงการอัปโหลดไฟล์ที่มีการตั้งค่าเอาไว้ก่อนหน้า	146
3.115 คำอธิบายแต่ละพารามิเตอร์ฟังก์ชัน	147
3.116 ผังงานตรวจสอบการทำงานของมอเตอร์ในการบังคับมุ่งเลี้ยว	148
3.117 แสดงสถานการณ์สื่อสารสำเร็จ	148
3.118 แสดงการเชื่อมต่อระหว่างตัวควบคุมกับซอฟต์แวร์	149
3.119 InnoMakerUSB2CAN Software	149
3.120 แสดงการรับส่งข้อมูลผ่านโปรแกรม InnomakerUSB2CAN	150
3.121 Data frame of Control mode	151
3.122 การติดตั้ง CANtools ใน Ubuntu	152
3.123 การทดสอบการเชื่อมต่อ CAN ใน การรับส่งค่า	153
3.124 ผังงานสำหรับทดสอบการรับ-ส่งค่าจากมอเตอร์	154
3.125 ผังงานแสดงขั้นตอนการทำงานของ Lateral Control Node	155
3.126 แสดงเวอร์ชัน Ros	158
3.127 การติดตั้ง Ubuntu	159
3.128 การติดตั้ง Ubuntu สำเร็จ	159
3.129 การติดตั้ง Ros noetic	161
3.130 ผังงานการทำงานของระบบ ROS	162
4.1 การเชื่อมต่อ Base และ Rover แบบใช้สาย	163
4.2 การรังวัดหาพิกัดของ Base Station แบบ Fixed Mode	164
4.3 สถานะ FIXED ของ Rover Station	164
4.4 สถานที่ที่ใช้ในการทดสอบ	165
4.5 รูปแบบตัวอักษรที่ใช้ในการทดสอบ	165
4.6 ผลการทดสอบผลพิกัดที่ได้ที่ได้	166
4.7 การเชื่อมต่อ Base และ Rover แบบไร้สาย	167

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
4.8 รายละเอียดการตั้งค่า RF Datalink	168
4.9 ภาพแสดงสถานะ TIME ของ Base Station และสถานะ FIXED ของ Rover Station	168
4.10 รูปแบบตัวอักษรที่ใช้ในการทดสอบ	169
4.11 ผลการทดลองที่ได้	169
4.12 ผลการทดลองจากการเปลี่ยนตำแหน่งของ Base Station	170
4.13 การเชื่อมต่อ Rover ด้วยเทคนิคการวัดแบบ Network RTK	171
4.14 ภาพแสดงสถานะ FIXED ของ Rover Station	172
4.15 รูปแบบตัวอักษรที่ใช้ในการทดสอบ	172
4.16 ผลการทดลองที่ได้	173
4.17 แสดงระยะทางอ้างอิงในการรังวัด	175
4.18 อุปกรณ์และการติดตั้งในการรังวัดแบบ Real Time Kinematic	175
4.19 GPS Quality Indicator ที่ได้ในการรังวัดแบบ Real Time Kinematic	176
4.20 อุปกรณ์และการติดตั้งในการรังวัดแบบ Network RTK	176
4.21 GPS Quality Indicator ที่ได้ในการรังวัดแบบ Network RTK	177
4.22 ระยะทางอ้างอิงในการรังวัด	177
4.23 อุปกรณ์และการติดตั้งในการรังวัดแบบ Real Time Kinematic	178
4.24 GPS Quality Indicator ที่ได้ในการรังวัดแบบ Real Time Kinematic	178
4.25 อุปกรณ์และการติดตั้งในการรังวัดแบบ Network RTK	179
4.26 GPS Quality Indicator ที่ได้ในการรังวัดแบบ Network RTK	179
4.44 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางโถงที่ 4	190
4.45 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางโถงที่ 4	191
4.46 การควบคุมความเร็วการเคลื่อนที่แบบ Forward (Km/hr) แบบ Open loop	192
4.47 การควบคุมความเร็วการเคลื่อนที่แบบ Backward (Km/hr) แบบ Open loop	193
4.48 ตัวอย่างการเก็บผลการทดลอง	195
4.49 ค่า Digital Potentiometer ที่ใช้ในการควบคุม	196
4.50 ค่าความเร็วที่ใช้ในการเคลื่อนที่ (km/hr)	196

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
4.51 ค่า Error ที่ได้จากการควบคุมความเร็วแบบ PID Controller	197
4.52 ค่าความเร็วที่ใช้ในการเคลื่อนที่ (km/hr)	198
4.53 ค่าความเร็ว Setpoint บริเวณรอบสนามฟุตบอล	200
4.54 ค่า Digital Potentiometer ที่ใช้ในการควบคุม	200
4.55 ค่า Error ที่ได้จากการควบคุมความเร็วแบบ PID Controller	200
4.56 แสดงการเชื่อมต่อระหว่างตัวควบคุมกับซอฟต์แวร์	202
4.57 แสดงการรับส่งข้อมูลผ่านโปรแกรม InnomakerUSB2CAN	202
4.58 การติดตั้ง CAN tools ใน Ubuntu	203
4.59 การทดสอบการเชื่อมต่อ CAN ในการรับ-ส่งค่า	204
4.60 ผังงานสำหรับทดสอบการรับ-ส่งค่าให้มอเตอร์	205
4.61 กราฟความสัมพันธ์ระหว่างเอาร์พุตมุ่งทางทฤษฎีและมุ่งที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาในรอบที่ 1	209
4.62 กราฟแสดงค่าความคลาดเคลื่อนเทียบกับเวลาในรอบที่ 1	210
4.63 กราฟความสัมพันธ์ระหว่างเอาร์พุตมุ่งทางทฤษฎีและมุ่งที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาในรอบที่ 2	211
4.64 กราฟความสัมพันธ์ระหว่างเอาร์พุตมุ่งทางทฤษฎีและมุ่งที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาในรอบที่ 3	212
4.65 กราฟความสัมพันธ์ระหว่างเอาร์พุตมุ่งทางทฤษฎีและมุ่งที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาในรอบที่ 4	213
4.66 กราฟความสัมพันธ์ระหว่างเอาร์พุตมุ่งทางทฤษฎีและมุ่งที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาในรอบที่ 5	214
4.67 กราฟความสัมพันธ์ระหว่างเอาร์พุตมุ่งทางทฤษฎีและมุ่งที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาในรอบที่ 6	215
4.68 กราฟแสดงค่าความคลาดเคลื่อนเทียบกับเวลาของการทดสอบการควบคุมมอเตอร์ในรอบที่ 1-6	215
4.69 กราฟความสัมพันธ์เปรียบเทียบระหว่างเอาร์พุตมุ่งทางทฤษฎีและมุ่งที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาของรอบที่ 1-6	216

## รายการรูปประกอบ (ต่อ)

รูป	หน้า
4.70 การติดตามวิถีเส้นทางรอบสนามฟุตซอลมจช. โดยการติดตามวิถีการขับเคลื่อนรถกลอส์ฟอตต์โนมัติโดยใช้เทคนิคการวัดแบบ Network RTK	217
4.71 การติดตามวิถีเส้นทางรอบสนามฟุตซอลมจช. ซึ่งใช้เส้นทางอ้างอิงโดยมีผู้ขับขี่ เปรียบเทียบกับการขับเคลื่อนอัตโนมัติของยานพาหนะ โดยใช้เทคนิคการวัดแบบ RTK	218
4.72 การติดตามวิถีเส้นทางรอบสนามฟุตซอลมจช. ซึ่งใช้เส้นทางอ้างอิงโดยมีผู้ขับขี่ เปรียบเทียบกับการขับเคลื่อนอัตโนมัติของยานพาหนะ โดยใช้เทคนิคการวัดแบบ RTK	218
4.73 การติดตามวิถีเส้นทางรอบสนามฟุตซอลมจช. เปรียบเทียบเส้นทางอ้างอิงกับการขับเคลื่อนอัตโนมัติของยานพาหนะ โดยการใช้เทคนิคการวัดแบบ RTK และใช้เทคนิคการวัดแบบ Network RTK	219
4.74 ค่าความคลาดเคลื่อนจากการติดตามวิถีการขับเคลื่อนอัตโนมัติรอบสนามฟุตซอลมจช. จากการใช้เทคนิคการวัดแบบ RTK	220
4.75 ค่าความคลาดเคลื่อนจากการติดตามวิถีการขับเคลื่อนอัตโนมัติรอบสนามฟุตซอลมจช. จากการใช้เทคนิคการวัดแบบ Network RTK	220
4.76 การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถกลอส์ฟอตต์โนมัติรอบสนามฟุตซอลบริเวณถนนเส้นตรงจากทิศใต้ไปทิศเหนือ (Linear S-N)	222
4.77 การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถกลอส์ฟอตต์โนมัติรอบสนามฟุตซอลบริเวณถนนทางโค้งที่ 1 (Curve1)	223
4.81 การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถกลอส์ฟอตต์โนมัติรอบสนามฟุตซอลบริเวณถนนทางโค้งที่ 3 (Curve3)	227

4.82	การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความ คลาดเคลื่อนการขับเคลื่อนรถกอล์ฟอัตโนมัติรอบสนามฟุตซอลบริเวณถนน เส้นตรงจากทิศตะวันตกไปทิศตะวันออก (Linear W-E)	228
4.83	การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความ คลาดเคลื่อนการขับเคลื่อนรถกอล์ฟอัตโนมัติรอบสนามฟุตซอลบริเวณถนน ทางโค้งที่ 4 (Curve4)	229
4.84	ผังงานการทำงานของระบบ ROS	230
4.85	ภาพแสดงผล rqt_graph ภายใน Robot Operating System	230
4.86	ภาพแสดงการส่ง Message ของระบบทั้งหมดผ่าน Terminal	230
4.88	ภาพแสดง rqt_graph สำหรับการรับและส่งข้อมูลของระบบควบคุมความเร็ว	231
4.89	ภาพตัวอย่างการ Run Ros Master	231
4.90	ภาพตัวอย่างการ Run Velocity Calculation เพื่อส่งข้อมูลให้ Node Data speed input	231
4.91	ภาพตัวอย่างการ Run การทำงานของ Arduino	232
4.92	ภาพตัวอย่างการ Run Node Data speed input เพื่อรับข้อมูลจาก Velocity Calculation	232

## บทที่ 1 บทนำ

### 1.1 ความสำคัญและที่มาของโครงงาน

ในปัจจุบันรถยนต์เป็นยานพาหนะที่ขับเคลื่อนด้วยพลังงานต่างๆ เช่น พลังงานไฟฟ้า พลังงานกล เป็นต้น ซึ่งพลังงานไฟฟ้าจะใช้พลังงานที่เก็บอยู่ในแบตเตอรี่ในการขับเคลื่อนด้วยมอเตอร์ไฟฟ้าในการเคลื่อนที่ จึงถูกใช้อย่างกว้างขวางในอุสาหกรรมรถยนต์

ระบบการควบคุมของรถยนต์เป็นส่วนที่ช่วยเพิ่มความสามารถในการเคลื่อนที่ ซึ่งในการพัฒนารถยนต์ให้เคลื่อนที่ได้ดีขึ้น จำเป็นต้องมีระบบควบคุมความเร็วและตำแหน่งของมอเตอร์ไฟฟ้ากระแสตรงในการเคลื่อนที่ตามความเร็วหรือระยะทางที่ต้องการ ทำให้การควบคุมความเร็วและตำแหน่งของมอเตอร์ไฟฟ้ากระแสตรง มีความสนใจในโครงงานจำนวนมาก และมีวิธีการหลักที่ใช้ในการควบคุม แต่การพัฒนาระบบไฟดิจิทัลนำมาใช้อย่างกว้างขวางสำหรับการควบคุมความเร็วและตำแหน่งของมอเตอร์ไฟฟ้ากระแสตรง และระบบควบคุมของรถยนต์อีกส่วนคือ ระบบควบคุมมุมเลี้ยวของล้อรถยนต์เพื่อให้สามารถเลี้ยวตามมุมที่ต้องการ ได้ ซึ่งทั้งสองระบบจะทำงานควบคู่กันเพื่อให้รถยนต์สามารถไปลึกลุ่มหมายปลายทางที่ตั้งไว้ได้และยังสามารถเคลื่อนที่ได้โดยอัตโนมัติ

เทคโนโลยีในการติดตามตำแหน่งของรถยนต์ในอดีตถูกใช้อย่างแพร่หลาย แต่ยังขาดระบบไร้ขับขี่และแสดงตำแหน่งของรถยนต์ ซึ่งจะใช้เทคโนโลยีระบบ GPS ช่วยในการติดตามรถยนต์ แสดงตำแหน่งของรถยนต์ และในการนำทางหรือการติดตามเส้นทางการเคลื่อนที่นั้น วิธีการควบคุมด้วยตัวควบคุมมีหลักหลาวยิ่ง ทั้งระบบควบคุมแบบสัดส่วนปริพันธ์อนุพันธ์ (PID Controller) [1] และระบบด้วยควบคุมแบบคาดการณ์ขั้ลลง (MPC Controller) [2] ที่ใช้ในระบบควบคุมยานพาหนะแบบไดนามิกก็ถูกนำมาปรับใช้ในงานนี้ด้วยในระบบไดรรูบหนึ่ง สำหรับการติดตามเส้นทางการเคลื่อนที่จากตำแหน่งของยานพาหนะ ซึ่งสามารถช่วยลดการใช้ผู้ขับขี่ในการขับเคลื่อนรถยนต์ไฟฟ้าในเส้นทางเดินช้าๆ เช่นการเดินรถไฟฟ้าสำหรับการรับส่งนักศึกษาภายในมหาวิทยาลัย การเดินรถรับส่งในหมู่บ้านหรือสวนสัตว์ เป็นต้น เนื่องจากผู้ขับขี่จะไม่ต้องขับขี่รถยนต์ด้วยตนเองโดยการใช้ตัวควบคุมสำหรับเคลื่อนที่รถไฟฟ้า สีเส้นทางเดินช้าๆ ที่มีความแน่นอน และสามารถกำหนดจุดหมายปลายทางที่ต้องการให้รถยนต์สามารถเคลื่อนที่ไปได้

อย่างไรก็ตามนี่จะจาก โภคามิกของยานพาหนะมีข้อจำกัดด้านสารคดแวร์และสิ่งแวดล้อมรบกวนความเสถียรของระบบและความแม่นยำในการติดตามวิถีเส้นทางเป็นสิ่งที่ท้าทาย พารามิเตอร์การตั้งค่าในตัวควบคุมมีความเกี่ยวข้องอย่างมากกับประสิทธิภาพ

## 1.2 วัตถุประสงค์ของโครงการ

1. เพื่อศึกษาและออกแบบปรับปรุงหลักการทำงานของรถกลอฟไฟฟ้า
2. เพื่อออกแบบการระบุตำแหน่งของรถกลอฟไฟฟ้า โดยใช้การสำรวจวัดด้วยดาวเทียมแบบจลน์ในการกำหนดพิกัดและเส้นทางอ้างอิง
3. เพื่อการศึกษาการควบคุมรถยนต์ไร้คนขับตามเส้นทางที่กำหนดไว้ผ่านระบบควบคุมหุ่นยนต์ ระบบควบคุมมุมเลี้ยวและระบบควบคุมความเร็ว

## 1.3 ประโยชน์และผลที่คาดว่าจะได้รับจากโครงการ

1. สามารถออกแบบและปรับปรุงระบบการทำงานของรถกลอฟไฟฟ้า
2. สามารถออกแบบการระบุตำแหน่งของรถกลอฟไฟฟ้า โดยใช้การสำรวจวัดด้วยดาวเทียมแบบจลน์ในการกำหนดพิกัดและเส้นทางอ้างอิงได้อย่างมีประสิทธิภาพและแม่นยำยิ่งขึ้น
3. สามารถเรียนรู้และเข้าใจการควบคุมรถยนต์ไร้คนขับตามเส้นทางที่กำหนดไว้ผ่านระบบควบคุมหุ่นยนต์ ระบบควบคุมมุมเลี้ยวและระบบควบคุมความเร็ว
4. สามารถนำความรู้จากการเรียนในภาควิชาระบบควบคุมและเครื่องมือวัดมาใช้ในการออกแบบและพัฒนาระบบการทำงานของรถกลอฟ

## 1.4 ขอบเขตของโครงการ

1. สามารถทำการสำรวจเส้นทางด้วยดาวเทียมในการกำหนดพิกัดได้มีความแม่นยำในระดับเซนติเมตร หรือมีความผิดพลาดไม่เกิน 1 เมตร
2. สามารถออกแบบและปรับปรุงวงจรไฟฟ้าภายในและระบบควบคุมความเร็วของรถไฟฟ้า
3. สามารถปรับปรุงระบบควบคุมมุมเลี้ยวและทิศทางการเคลื่อนที่ได้ โดยใช้ CAN bus รวมทั้งออกแบบ Steering Part สำหรับติดตั้งมอเตอร์กับพวงมาลัย
4. สามารถควบคุมรถกลอฟไฟฟ้าแบบ Lateral Motion Control ด้วยความเร็วคงที่
5. สามารถใช้ระบบควบคุมหุ่นยนต์ควบคุมการเคลื่อนที่ของรถไฟฟ้าให้เคลื่อนที่ไปยังเส้นทางที่ต้องการได้

## 1.5 ขั้นตอนการดำเนินงาน

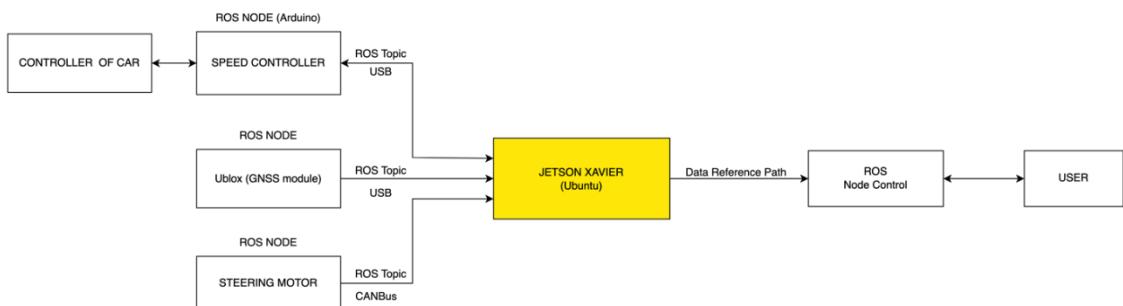
## ตารางที่ 1.1 ตารางแสดงแผนการดำเนินงาน (Gantt Chart) ในปีการศึกษา 2565

## บทที่ 2 ภาพรวมโครงงานและทฤษฎีที่ใช้

### 2.1 ภาพรวมการทำงานของระบบห้องหมด



รูปที่ 2.1 ภาพรวมของรถกอล์ฟ



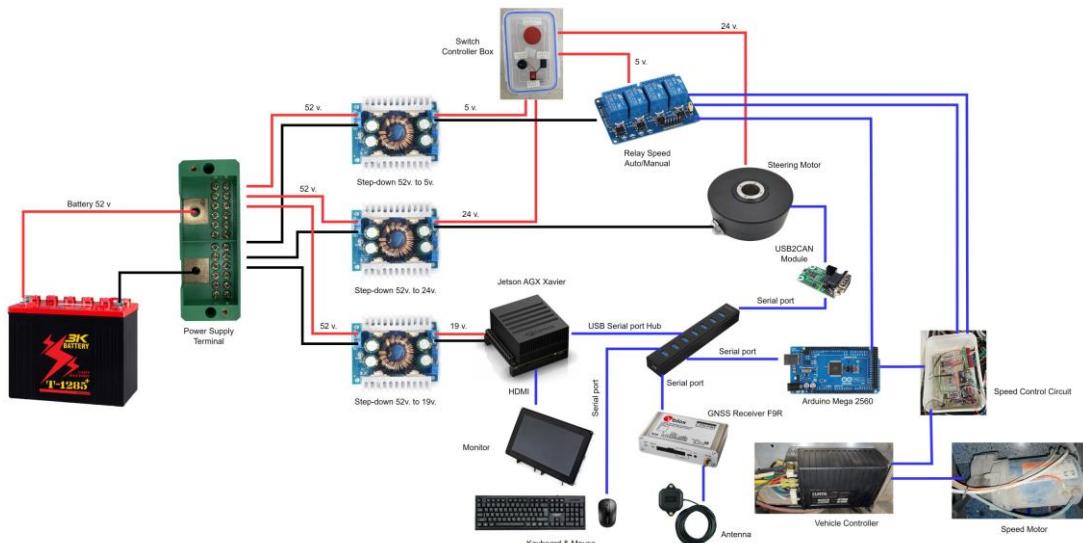
รูปที่ 2.2 Diagram ภาพรวมการทำงานของระบบ

โดยการควบคุมการทำงานของรถกอล์ฟให้เคลื่อนที่อัตโนมัติจะใช้ Jetson AGX Xavier เป็นอุปกรณ์ในการประมวลผลกลางของระบบ ซึ่งจะแบ่งการทำงานออกเป็น 3 ระบบ โดยเริ่มจากใช้ GNSS Module ใน การเก็บค่าพิกัด ซึ่งพิกัดที่จะส่งมีด้วยกัน 2 ชนิด ได้แก่ 1. พิกัดอ้างอิงในการเคลื่อนที่ของรถ 2. พิกัดของรถ

ในขณะนี้ ถัดมาในส่วนของการควบคุมความเร็วรถให้มีความเร็วที่คงที่ เริ่มจากการต่อ Controller ของรถเข้ากับ วงจร Speed Controller Box ที่ใช้ในการควบคุมการทำงานของความเร็ว ด้านในประกอบด้วยตัวประมวลผล และควบคุมการทำงานของความเร็ว (Arduino Mega 2560) ซึ่งตัวประมวลผลการทำงานของระบบจะต่อเข้ากับ Jetson Xavier ที่เป็นตัวประมวลผลกลางอีกที ส่วนสุดท้ายคือ การควบคุมมุมเลี้ยวของรถ โดยจะรับค่าพิกัด XY จาก GNSS มาเป็น อินพุตของระบบ PID Controller ซึ่งมอเตอร์ควบคุมพวงมาลัย (Steering Motor) ในการบังคับมุมเลี้ยวของรถ แล้วนำไปประมวลผลร่วมกับสมการ Kinematic Bicycle Model เพื่อบังคับทิศทางของรถและขั้นตอนมอเตอร์ให้พวงมาลัยหมุน

## 2.2 ภาพรวมการทำงานหาร์ดแวร์

การออกแบบวงจรรถกลอฟให้ขับเคลื่อนที่อัตโนมัติด้วยความเร็วคงที่ เป็นการนำระบบเดิมของรถกลอฟ มาดัดแปลง หรือนำอุปกรณ์มาติดตั้งเพิ่มเพื่อให้รถสามารถทำงานอัตโนมัติ โดยอุปกรณ์ที่นำมาติดตั้งเพิ่ม ล้วนมีความต้องการทางไฟฟ้าที่แตกต่างกัน ซึ่งสิ่งที่ต้องคำนึงถึงคือเรื่องของความปลอดภัยของตัวผู้ใช้งาน และคนรอบข้าง



รูปที่ 2.3 ภาพรวมของวงจรทั้งหมดที่มีการเชื่อมต่อให้รถกลอฟสามารถเคลื่อนที่ได้อัตโนมัติ

โดยแหล่งจ่ายแรงดันไฟฟ้าที่นำมาต่อเข้ากับอุปกรณ์ต่างๆนั้นเป็นการแบ่งแรงดันไฟฟ้ากระแสตรง 53V จากแบบเดอร์รีของรถต่อเข้ากับอุปกรณ์ลดแรงดัน (Step Down) เพื่อลดขนาดแรงดันจาก 53V เป็น 19V จ่ายให้มอเตอร์ที่ควบคุมการทำงานของมุมเลี้ยว, จาก 53V เป็น 9V เพื่อจ่ายให้กับตัวประมวลผลหลักการทำงานของระบบ (Jetson Xavier) ซึ่งเป็นอุปกรณ์ที่ควบคุมการทำงานของระบบทั้งหมดภายในรถ

สุดท้ายลดอุณหภูมิจาก 53V เป็น 5V เพื่อจ่ายเข้า向จร Switch Controller Box เป็นวงจรควบคุมการทำงานของสวิตช์เลือกโหมดการทำงานต่างๆ รวมไปถึงควบคุมการทำงานของ Relay ที่ต่อเข้ากับวงจรควบคุมการทำงานของความเร็ว

### 2.3 ภาพรวมการทำงานซอฟต์แวร์

ในส่วนซอฟต์แวร์ที่ควบคุมการทำงานของระบบจะมีโปรแกรมที่ติดตั้งอยู่บนระบบปฏิบัติการ Windows และระบบปฏิบัติการ Ubuntu หรือ ROS เพื่อติดต่อสื่อสาร หรือ รับ-ส่งข้อมูลกันภายในระบบ โดยซอฟต์แวร์ที่ใช้ในการควบคุมการทำงานของระบบทั้งหมด ได้แก่

#### 1. โปรแกรมภาษา ROS (Robot Operating System)

เป็นระบบที่สร้างขึ้นเพื่อทำให้เกิดความยืดหยุ่นในการเขียนซอฟต์แวร์ควบคุมหุ่นยนต์ซึ่งใน ROS จะรวบรวมชุดเครื่องมือและชุดคำสั่งต่างๆ ที่จำเป็นในการพัฒนาหุ่นยนต์เอาไว้ ซึ่งสิ่งต่างๆ เหล่านี้จะลดความยุ่งยากในการสร้าง และพัฒนาหุ่นยนต์ที่มีความซับซ้อน ทำให้มีประสิทธิภาพในการพัฒนาหุ่นยนต์หลายรูปแบบ



รูปที่ 2.4 แสดงโปรแกรมภาษา ROS (Robot Operating System)

#### 2. โปรแกรมภาษา C+ (C Programming Language)

เป็นภาษาคอมพิวเตอร์ที่ใช้สำหรับพัฒนา โปรแกรมทั่วไป ภาษาซีเป็นภาษาเบื้องต้นของโปรแกรมระบบเชิงคำสั่ง ( หรือเชิงกระบวนการ ) ถูกออกแบบขึ้น เพื่อใช้แปลงตัวเปลี่ยนภาษา เช่น ภาษา C แม้จะเป็นภาษา ระดับสูง แต่ก็สามารถใช้เป็นภาษาเครื่องได้ เป็นอย่างดี สามารถนำไปใช้ได้บนเครื่องทุก platform ไม่ว่าจะ เป็น Intel PC ที่วิ่ง Windows 95 หรือ Windows NT, Windows XP, Windows 7 หรือ แม้แต่ Linux ทั้งเครื่อง Macintosh และ เครื่องเวอร์กสเตชันตลอดจนเมนเฟรม เนื่องจาก มี compiler ของภาษาซี อยู่ทั่วไป



รูปที่ 2.5 แสดงโปรแกรมภาษา C+ (C Programming Language)

### 3. โปรแกรมภาษา Python (Python Programming Language)

เป็นภาษาคอมพิวเตอร์ระดับสูง โค้ดถูกออกแบบมาให้เป็นภาษาสคริปท์อ่านง่าย โดยตัดความซับซ้อนของโครงสร้างและไวยกรณ์ของภาษาออกไป ในส่วนของการแปลงชุดคำสั่งที่เราเขียนให้เป็นภาษาเครื่อง Python มีการทำงานแบบ Interpreter คือเป็นการแปลงชุดคำสั่งที่จะบรรยาย เพื่อป้อนเข้าหน่วยประมวลผลให้คอมพิวเตอร์ทำงานตามที่เราต้องการ นอกจากนี้ Python ยังสามารถนำไปใช้ในการเขียนโปรแกรมได้หลายประเภท



รูปที่ 2.6 แสดงโปรแกรมภาษา Python (Python Programming Language)

### 4. Visual Studio Code

เป็นโปรแกรมแก้ไขซอฟต์แวร์โค้ดที่พัฒนาโดยไมโครซอฟท์สำหรับ windows,linux และ macOS มีการสนับสนุนสำหรับการดีบัก การควบคุม git ในตัว github การเน้นไวยกรณ์ การเติมโค้ดอัจฉริยะ ตัวอย่าง และ code refactoring มันสามารถปรับแต่งได้หลายอย่าง ให้ผู้ใช้สามารถเปลี่ยนชื่อและเพิ่มโค้ด การตั้งค่า และติดตั้งส่วนขยาย ที่เพิ่มฟังก์ชั่นการทำงานเพิ่มเติม ซอฟต์แวร์นี้ฟรีและโอเพนซอร์ส และเผยแพร่ภายใต้ลิขสิทธิ์การใช้งาน MIT ใบอนุญาตค่อนข้างกว้างและเป็นfreeware และฟรีสำหรับการใช้ส่วนตัวหรือเพื่อการค้า



รูปที่ 2.7 แสดงโปรแกรม Visual Studio Code

## 5. Arduino IDE

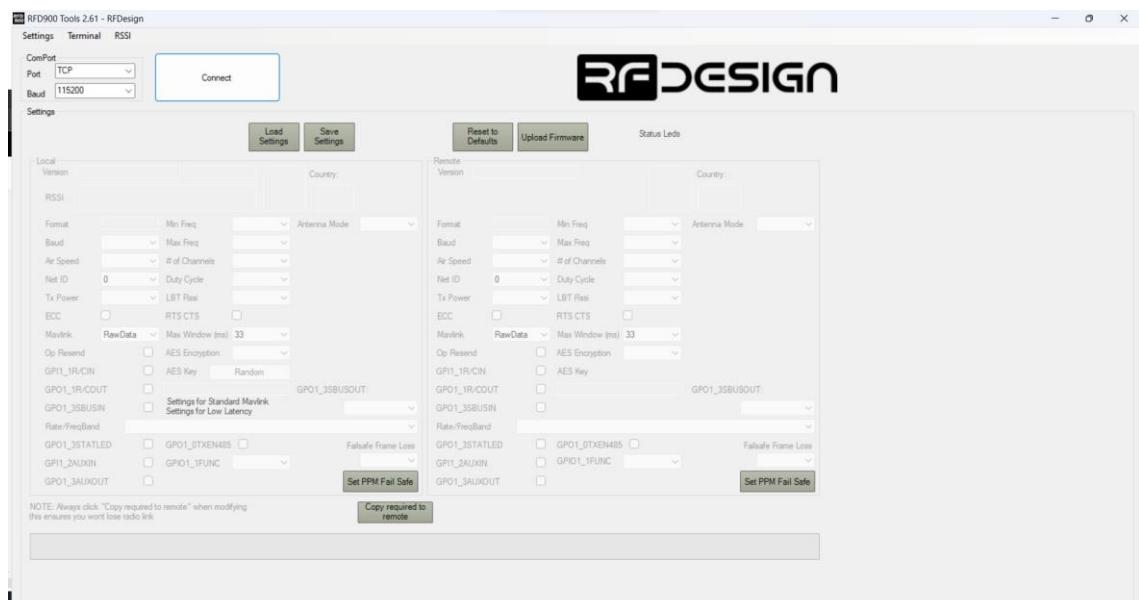
เป็นโปรแกรมที่ควบคุมการทำงานในส่วนของระบบควบคุมความเร็ว โดยใช้ภาษา C++ และ C ในการเขียนชุดคำสั่ง และอัปโหลดลงบอร์ด Arduino เพื่อใช้ในการควบคุมความเร็วให้คงที่



รูปที่ 2.8 แสดงโปรแกรม Arduino IDE

## 6. RFD tools

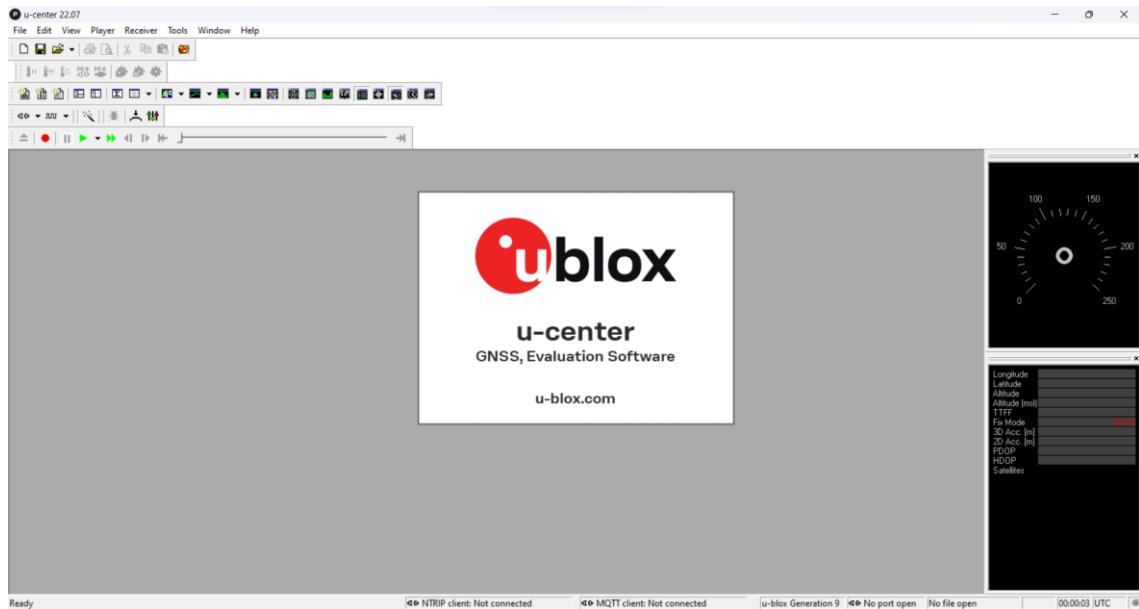
เป็นโปรแกรมตั้งค่าโมดูลที่ช่วยให้การสื่อสารของ U-blox GNSS นั้นสามารถสื่อสารได้ด้วยระบบไร้สาย เป็นซอฟต์แวร์สำหรับตั้งค่า RFD 900A DATA LINK โดย RFDesign เป็นบริษัทออกแบบ อุปกรณ์อิเล็กทรอนิกส์ที่เชี่ยวชาญด้านเสาอากาศ RF และการพัฒนาผลิตภัณฑ์แบบฝัง ได้แก่ เสาอากาศ RFID เป็นต้น



รูปที่ 2.9 แสดงโปรแกรม RFD tools

## 7. U center

เป็นโปรแกรมมาตั้งค่าโมดูล U-blox GNSS ด้วย U-center U-center เป็นซอฟต์แวร์สำหรับเรียกใช้ค่าพิกัดจาก module GNSS และแสดงข้อมูลตำแหน่งและจำนวนดาวเทียมระบุตำแหน่ง และเครื่องมืออีกหลายอย่างตามความต้องการของผู้ใช้งานเรียกเป็นหน้าต่างดูข้อมูล



รูปที่ 2.10 แสดงโปรแกรม U center

## 8. SolidWorks

เป็นโปรแกรมที่ใช้ในการออกแบบสาร์ตแวร์ ชิ้นส่วนที่ติดตั้งเพิ่มเติม ต่อเข้ากับพวงมาลัยและ Steering motor



รูปที่ 2.11 แสดงโปรแกรม SolidWork

#### 9. InnoMakerUSB2CAN Software

เป็นซอฟต์แวร์สำหรับควบคุมการสื่อสารระหว่าง USB2CAN กับอุปกรณ์ไดๆ เช่น Steering Gear Steering Wheel Motor ซึ่งสามารถส่งและรับค่าภายในซอฟต์แวร์ โดยการตั้งค่าพารามิเตอร์ให้ตรงตาม อุปกรณ์ที่ใช้สื่อสาร



รูปที่ 2.12 แสดงโปรแกรม InnoMakerUSB2CAN Software

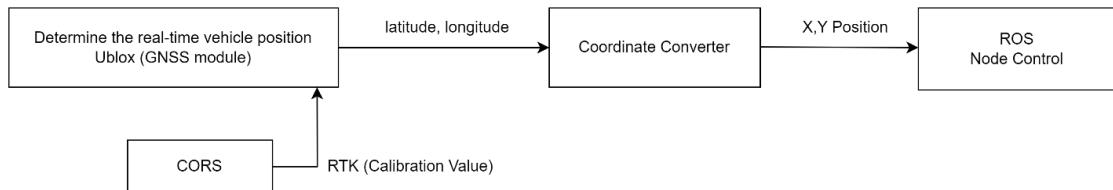
#### 10.kyMotor Control Utility Software

เป็นซอฟต์แวร์สำหรับควบคุมการการทำงานของมอเตอร์พวงมาลัย (Steering Gear Steering Wheel Motor) โดยสามารถใช้ตั้งค่าพารามิเตอร์ต่างๆ เช่น Baudrate, Gain (Kp, Ki, Kd), Protocol (RS232, CAN) เป็นต้น มักจะใช้ในการตั้งค่ามอเตอร์ในครั้งแรกก่อนการใช้งาน รวมถึงสามารถใช้ทดสอบการทำงานในการสื่อสารตามที่ได้เลือกไว้ เช่นการควบคุมให้มอเตอร์หมุนเพื่อดูว่ามีความผิดพลาดใดๆ เกิดขึ้นนอกเหนือจากนั้นหรือไม่ เช่น ค่าแรงดันอาจไม่เพียงพอหรือมากเกินไป เป็นต้น



รูปที่ 2.13 แสดงโปรแกรม kyMotor Control Utility Software

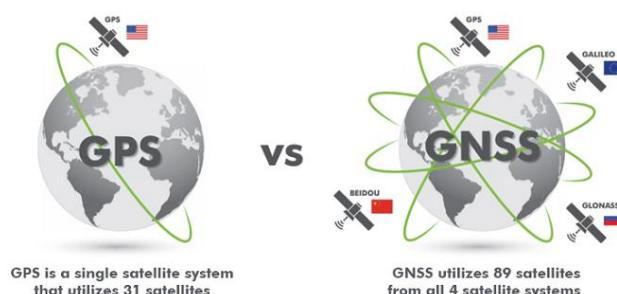
## 2.4 ภาพรวมและทฤษฎีต่างๆ ของระบบระบุตำแหน่ง



รูปที่ 2.14 ภาพรวมและลำดับการทำงานของระบบระบุตำแหน่ง

ในส่วนของการระบุตำแหน่งนั้น คือการที่รถยนต์ไฟฟ้าสามารถรับข้อมูลตำแหน่งพิกัดจากการสำรวจวัดแบบจลน์ได้แบบเรียลไทม์ เพื่อนำไปใช้ในการสร้างเส้นทางการเคลื่อนที่อ้างอิง และใช้ในการระบุตำแหน่งเพื่อควบคุมรถ ซึ่งในการที่จะทำการระบุตำแหน่งได้นั้นจะเริ่มจากการที่เชื่อมต่อของ GNSS Receiver เข้ากับสถานีอ้างอิงจากกรมที่ดินเพื่อรับค่าปรับแก้ผ่านทางอินเทอร์เน็ต ซึ่งการเชื่อมต่อในลักษณะนี้จะทำให้ค่าพิกัดที่รังวัดได้มีความละเอียดและความผิดพลาดน้อยกว่า 1 เมตร หลังจากนั้นจะนำพิกัดที่รังวัดได้ไปใช้ในการควบคุมทิศทางการเคลื่อนที่ของรถต่อไป

การตรวจจับพิกัดและนำทางรถจะใช้ระบบ Global Navigation Satellite System (GNSS) คือ เครื่องข่ายระบบนาฬิกัดตำแหน่ง โดยการรับสัญญาณระบบดาวเทียมเป็นที่รู้จักและใช้งานอย่างแพร่หลาย สามารถทำงานได้ทั้งกลางวันและกลางคืนตลอด 24 ชั่วโมง ประกอบด้วยดาวเทียมหลายระบบ ได้แก่ 1.ระบบดาวเทียม GPS ของสหรัฐอเมริกา 2.ระบบดาวเทียม GLONASS ของประเทศไทย 3.ระบบดาวเทียม Galileo ของสหภาพยุโรป 4.ระบบดาวเทียม BeiDou ของประเทศจีน 5.ระบบดาวเทียม QZSS ของประเทศญี่ปุ่น 6.ระบบดาวเทียม IRNSS หรือ NavIC ของประเทศอินเดีย



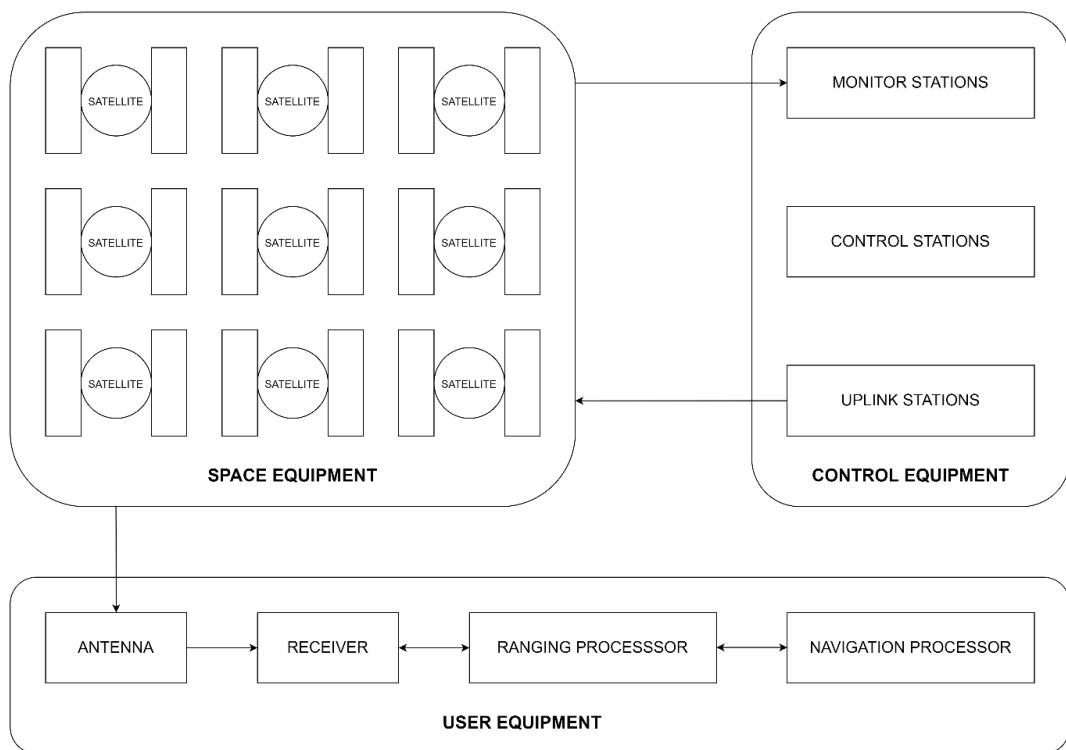
รูปที่ 2.15 การเปรียบเทียบระหว่างระบบ GPS และ GNSS

ตารางที่ 2.1 ตารางแสดงรายละเอียดต่างๆ ของดาวเทียมในระบบ GNSS

	Operator	Coverage	Altitude (km)	Satellites in Orbit
GPS	US Space Force	Global	20,180	32
GLONASS	Roscosmos	Global	19,130	24
Galileo	GSA and ESA	Global	23,222	23
BeiDou	CNSA	Global	21,528 (MEO satellite) 35,786 (GEO and IGSO satellite)	35
QZSS	JAXA	Regional	32,000 (perigee) 40,000 (apogee)	5
IRNSS/NavIC	ISRO	Global	36,000	7

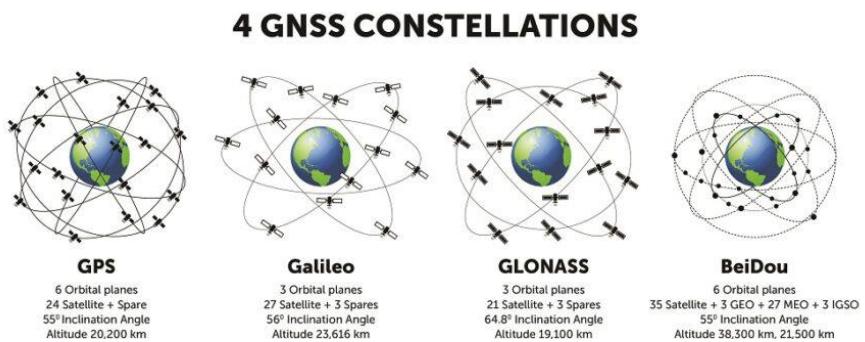
ระบบหาพิกัดตำแหน่งด้วยดาวเทียม GNSS ประกอบไปด้วย 3 ส่วนหลักดังนี้

1. Space Segment หรือ ส่วนอวกาศ
2. Control Segment หรือ ส่วนควบคุม
3. User Segment หรือ ส่วนผู้ใช้



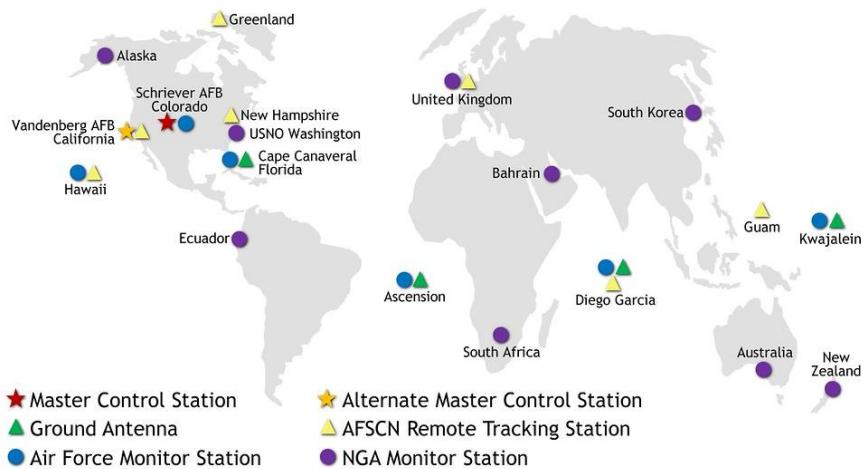
รูปที่ 2.16 สถาปัตยกรรมของระบบหาพิกัดตำแหน่งด้วยดาวเทียม GNSS

- Space Segment หรือส่วนอวกาศ ประกอบด้วย ตัวดาวเทียมและกลุ่มสัญญาณที่ส่งออกมาจากดาวเทียม โดยระบบดาวเทียม GNSS ซึ่งโครงการลูกที่ความสูงประมาณ 20,000 กิโลเมตร โดยดาวเทียม GNSS ในแต่ละระบบ สามารถรับดาวเทียมได้อย่างน้อย 4 ดวงทั่วโลกตลอด 24 ชั่วโมง และสัญญาณที่ส่งออกมาจากดาวเทียม โดยทั่วไปจะเป็นกลุ่มวิทยุที่ถูกรวมกับรหัสและข้อมูลดาวเทียม ที่เรียกว่า Modulation หรือการถ่ายทอดสัญญาณด้วยรหัสและข้อมูลจากดาวเทียม ซึ่งประกอบไปด้วย 1. Ephemeris หรือข้อมูลวงโคจรที่ถูกต้องของดาวเทียม 2. Code หรือรหัสต่างๆ 3. ข้อมูล Carrier Phase และ 4. Almanac Information หรือข้อมูลตำแหน่งตำแหน่ง โดยประมาณของดาวเทียมทั้งหมด เป็นต้น



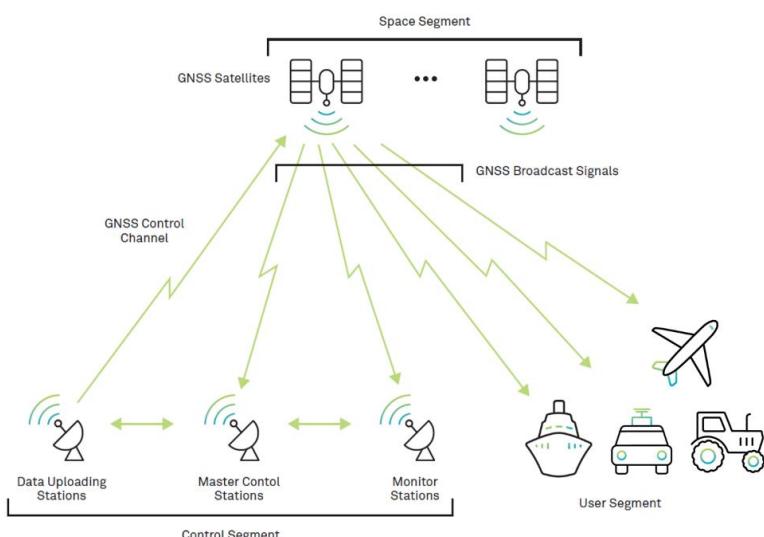
รูปที่ 2.17 จำนวนและวงโคจรของดาวเทียมในระบบ GNSS

- Control Segment หรือส่วนควบคุม ประกอบด้วยสถานีภาคพื้นดินที่จำเป็นต่อการติดตามดาวเทียม การคำนวณวงโคจรดาวเทียม การดูแลรักษาระบบ และควบคุมส่วนอวกาศ ซึ่งประกอบด้วย 3 สถานีได้แก่ 1. สถานีควบคุมหลักหรือ Master Control Stations ที่ทำการประมวลข้อมูลทั้งหมด เช่น ถ้าความคลาดเคลื่อนข้อมูลวงโคจรดาวเทียม ข้อมูลค่าแก้เวลา นาฬิกาดาวเทียม เป็นต้น 2. สถานีอัปโหลดหรือ Upload Stations จะเป็นสถานีที่คอยส่งข้อมูลที่คำนวณได้จากสถานีควบคุมหลักไปยังดาวเทียม 3. สถานีติดตามหรือ Monitor Stations ที่ทำหน้าที่คอยติดตามดาวเทียม



รูปที่ 2.18 สถานีควบคุมภาคพื้นดินของดาวเทียมในระบบ GNSS ข้อมูล ณ ปี พ.ศ. 2559

3. Use Segment หรือส่วนผู้ใช้ เป็นส่วนที่ประยุกต์ใช้งานในด้านต่างๆ ตัวเครื่องรับสัญญาณ หรือ วิธีการการประมวลผลที่อยู่ในรูปของซอฟต์แวร์ สำหรับประเภทผู้ใช้จะถูกแบ่งออกเป็นผู้ใช้ทางการทหารและพลเรือน โดยผู้ใช้พลเรือนจะไม่ได้รับอนุญาตให้สามารถเข้าถึงสัญญาณหรือ บริการของ GNSS ได้ทั้งหมด ในปัจจุบันเครื่องรับสัญญาณดาวเทียมมีความหลากหลายอย่างมาก โดยแบ่งตามชนิดของรางวัล ได้แก่ การรังวัดด้วยชูโคลเรนจ์การรับสัญญาณความถี่แบบหนึ่ง ความถี่ ส่องความถี่ หรือมากกว่านั้น โดยทั่วไปแล้วจะให้ข้อมูลสถานะกลุ่มดาวเทียม กำหนดเวลาและข้อมูลการโครงการ

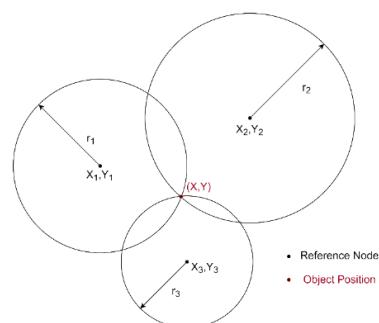


รูปที่ 2.19 ตัวอย่างการสถาปัตยกรรมของระบบหาพิกัดตำแหน่งด้วยดาวเทียม GNSS

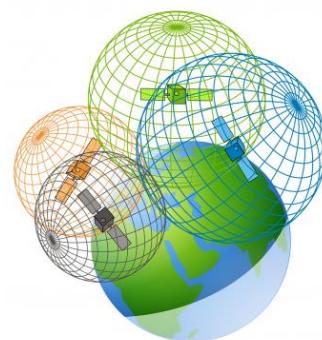
#### 2.4.1 หลักการวัดตำแหน่งด้วยดาวเทียมระบบ GNSS

ดาวเทียมจะโครงการบโลกวันละ 2 รอบในวงโคจร โดยดาวเทียมแต่ละดวงจะส่งสัญญาณและปัจจัยการโครงการเฉพาะตัวที่ช่วยให้อุปกรณ์รับสัญญาณสามารถอุดอครหัสและคำนวณตำแหน่งที่แม่นยำของดาวเทียม ดังกล่าวได้ ซึ่งตัวรับสัญญาณจะใช้ข้อมูลนี้และวิธีการสามเหลี่ยมระยะ (Trilateration) ในการคำนวณตำแหน่งที่ถูกต้องของผู้ใช้ โดยหลักการแล้วตัวรับสัญญาณจะวัดระยะห่างจากดาวเทียมแต่ละดวงโดยอ้างอิงจากระยะเวลาที่ใช้ในการรับสัญญาณที่ส่งมาได้ ด้วยค่าวัดระยะทางที่ได้จากการเที่ยวนอกระบบในบริเวณใกล้เคียง

กระบวนการ Trilateration คือ กระบวนการที่ใช้ในการหาพิกัดของโหนดของวัตถุที่ต้องการทราบตำแหน่งในระบบดาวเทียม เพื่อรับตำแหน่งของวัตถุนั้นๆ โดยจะอาศัยโหนดอ้างอิงอย่างน้อย 3 โหนด ซึ่งแต่ละโหนดอ้างอิงจะมีรัศมีครอบคลุมแพร่ออกไปในลักษณะของวงกลม ซึ่งวัตถุที่ต้องการหาจะต้องอยู่ในรัศมีครอบคลุมโหนดอ้างอิง 3 โหนด การอ้างอิงระยะทางระหว่างโหนดอ้างอิงและวัตถุแสดงดังรูปที่ 2.20



รูปที่ 2.20 การอ้างอิงระยะทางระหว่างโหนดอ้างอิงและตำแหน่งของวัตถุ



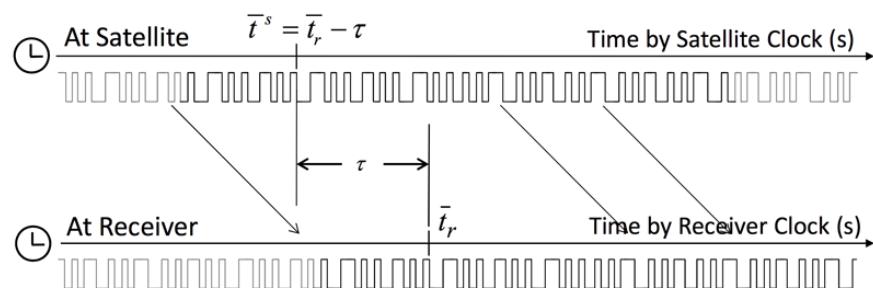
รูปที่ 2.21 ภาพอธิบายกระบวนการ Trilateration เมื่อนำมาใช้กับการหาพิกัดแบบดาวเทียม

## 2.4.2 การคำนวณหาตำแหน่งในการรังวัดดาวเทียมระบบ GNSS

ค่าที่รังวัดได้จากการรับสัญญาณดาวเทียมเพื่อนำมาใช้ประโยชน์ในการคำนวณตำแหน่งที่สำคัญมี 2 ชนิด ได้แก่ ชูโอดเรนจ์ (Pseudo range) และเฟสของคลื่นส่ง (Carrier Phase) ซึ่งมีรายละเอียดดังนี้

### 1. ชูโอดเรนจ์ (Pseudo range)

ชูโอดเรนจ์ คือ ระยะทางระหว่างดาวเทียมกับเครื่องรับสัญญาณ หากได้จากการอุดตัวจากสัญญาณที่ส่งมาจากดาวเทียมเปรียบเทียบกับรหัสที่เครื่องรับสัญญาณสร้างขึ้น โดยจะทำการเลื่อนไปมาจนกระทั่งได้รหัสที่ตรงกัน ค่าเลื่อนระหว่างรหัสทั้งสอง คือ ระยะเวลาที่คลื่นวิทยุใช้ในการเดินทางจากดาวเทียมมาจังหวะเครื่องรับสัญญาณ



รูปที่ 2.22 ภาพการเบริ่งเทียบสัญญาณของรหัสเพื่อหาเวลาที่คลื่นเดินทางจากดาวเทียมมาจังหวะเครื่องรับ

เมื่อนำความเร็วของคลื่นวิทยุคูณด้วยระยะเวลาที่ใช้ในการเดินทางระหว่างดาวเทียมมาจังหวะเครื่องรับสัญญาณ จะได้ระยะทางระหว่างดาวเทียมกับเครื่องรับสัญญาณซึ่งเรียกว่า ‘ชูโอดเรนจ์’ โดยปัจจัยที่มีผลต่อความถูกต้องของระยะเวลาดังกล่าว ได้แก่ ความเที่ยงตรงของนาฬิกาดาวเทียมและนาฬิกาเครื่องรับ ซึ่งมักมีความแตกต่างกัน รวมถึงความคลาดเคลื่อนที่เกิดจากคลื่นสัญญาณดาวเทียมที่เดินทางผ่านชั้นบรรยากาศของโลกมาจังหวะเครื่องรับสัญญาณ และปัจจัยประกอบอื่นๆ โดยมีสมการของชูโอดเรนจ์ที่ได้จากการหักและนำไปใช้เป็นระยะทาง ดังสมการที่ (2.1)

$$R = v \times t \quad (2.1)$$

โดย

$R$  คือ ระยะทางชูโอดเรนจ์ระหว่างดาวเทียมและเครื่องรับสัญญาณ (เมตร)

$v$  คือ ความเร็วของคลื่นวิทยุมีค่าเท่ากับความเร็วแสง (เมตรต่อวินาที)

$t$  คือ ระยะเวลาที่คลื่นเดินทาง (วินาที)

ในความเป็นจริงชุดโอดเรนจะมีค่าความคลาดเคลื่อนไปจากการทางวิธีที่มีความต้องรับสัญญาณมี หลายชนิด ได้แก่ ความคลาดเคลื่อนของโควตาที่ยิม ความคลาดเคลื่อนของนาฬิกาที่ยิม และ ความคลาดเคลื่อนเมื่อคลื่นทางผ่านชั้นบรรยากาศ เป็นต้น โดยมีสมการชุดโอดเรนที่ได้จากการหัสดังนี้เป็นระบบดังสมการที่ (2.2)

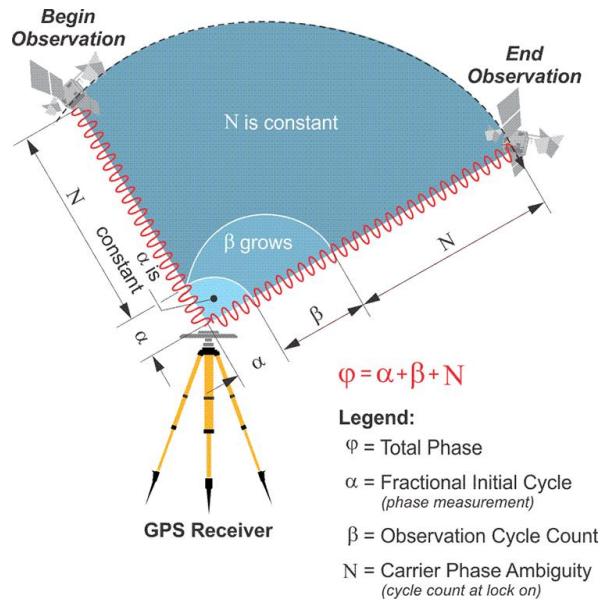
$$P = p + d_p + c(dt-dT) + d_{ion} + d_{trop} + E_{mp} + E_p \quad (2.2)$$

โดย

P	คือ	ชุดโอดเรนที่ได้รับจากการวัดรหัสของคลื่นส่ง (เมตร)
p	คือ	ระยะทางเรขาคณิตระหว่างดาวเทียมและเครื่องรับ (เมตร)
$d_p$	คือ	ความคลาดเคลื่อนเนื่องจากโควตาของดาวเทียม (เมตร)
c	คือ	ความเร็วของคลื่นส่ง (เมตร/วินาที)
dt	คือ	ความคลาดเคลื่อนของนาฬิกาที่ยิม (วินาที)
$dT$	คือ	ความคลาดเคลื่อนของนาฬิกาเครื่องรับ (วินาที)
$d_{ion}$	คือ	ความคลาดเคลื่อนเนื่องจากชั้นบรรยากาศไอโอนอฟเฟียร์(Ionosphere) ของคลื่นส่ง (เมตร)
$d_{trop}$	คือ	ความคลาดเคลื่อนเนื่องจากชั้นบรรยากาศโทรโพสฟีเยอร์(Troposphere) (เมตร)
$E_{mp}$	คือ	ความคลาดเคลื่อนเนื่องจากการเกิดคลื่นหลายวิถี (Multipath) (เมตร)
$E_p$	คือ	ความคลาดเคลื่อนของเครื่องรับสัญญาณ (เมตร)

## 2. เฟสของคลื่นส่ง (Carrier phase)

สำหรับงานที่ต้องการค่าความละเอียดถูกต้องสูงในระดับเซนติเมตร ต้องใช้ข้อมูลเฟสของคลื่นส่งใน การประมวลผล ซึ่งการวัดเฟสของคลื่นส่งในเครื่องรับเป็นการวัดเบริขบเทียบหรือค่าต่างระหว่างเฟส ของคลื่นส่งที่ดาวเทียมส่งลงมา กับเฟสของคลื่น ความถี่  $f_o$  ที่เครื่องรับสร้างขึ้นมา ดังแสดงในภาพที่ 47 โดยคลื่นส่งที่ดาวเทียมส่งลงมานั้นแยกออกเป็น 2 ส่วน คือ 1. ส่วนของจำนวนลูกคลื่นเต็มลูก เรียกว่า เลขปริศนา (Ambiguity) 2. ส่วนของคลื่นที่ไม่เต็มลูก เรียกว่า เศษของลูกคลื่น (Fractional part) การวัดเฟสของคลื่นส่งมีสมการดังสมการที่ (2.3)



รูปที่ 2.23 ภาพแสดงการวัดเฟสของคลื่นส่ง

$$\varnothing = p + d_p + c(dt-dT) + d_{ion} + d_{trop} + E_{mp} + E_p + \lambda N \quad (2.3)$$

โดย

$\varnothing$	คือ	ข้อมูลเฟสที่ได้รับจากการวัดเฟสของคลื่นส่ง (เมตร)
$p$	คือ	ระยะทางเรขาคณิตระหว่างดาวเทียมและเครื่องรับ (เมตร)
$d_p$	คือ	ความคลาดเคลื่อนเนื่องจากวงโคจรของดาวเทียม (เมตร)
$c$	คือ	ความเร็วของคลื่นส่ง (เมตร/วินาที)
$dt$	คือ	ความคลาดเคลื่อนของนาฬิกาดาวเทียม(วินาที)
$dT$	คือ	ความคลาดเคลื่อนของนาฬิกาเครื่องรับ (วินาที)
$d_{ion}$	คือ	ความคลาดเคลื่อนเนื่องจากชั้นบรรยากาศไอโอดีฟิเชียร์(Ionosphere) ของคลื่นส่ง (เมตร)
$d_{trop}$	คือ	ความคลาดเคลื่อนเนื่องจากชั้นบรรยากาศไทรโพรโพสฟีบร์(Troposphere) (เมตร)
$E_{mp}$	คือ	ความคลาดเคลื่อนเนื่องจากการเกิดคลื่นหลายวิถี (Multipath) (เมตร)
$E_p$	คือ	ความคลาดเคลื่อนของเครื่องรับสัญญาณ (เมตร)
$\lambda$	คือ	ความยาวของคลื่นส่ง (เมตร)
$N$	คือ	จำนวนเลขปริศนาของคลื่นส่ง

หลักการของการกำหนดตำแหน่งจุดเดียวความละเอียดสูง หรือ Precise Point Positioning นั้นจะเกี่ยวข้องกับการลดความคลาดเคลื่อนต่างๆ ที่ปรากฏในสมการข้างต้น ด้วยวิธีการดังนี้

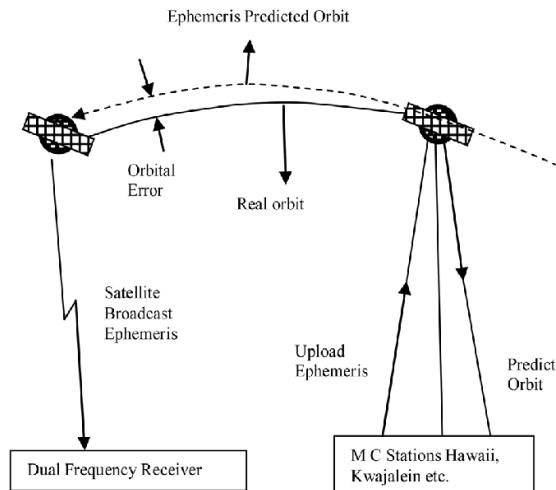
1. ความคลาดเคลื่อนของนาฬิกาดาวเทียม ( $d_t$ ) และความคลาดเคลื่อนของวงโคจรดาวเทียม ( $d_p$ ) จะใช้ค่าแก้ไขนาฬิกาดาวเทียมความละเอียดสูงและวงโคจรดาวเทียมความละเอียดสูงซึ่งมีความถูกต้องของวงโคจรดาวเทียมในระดับต่ำกว่า 5 เซนติเมตร และความถูกต้องของค่าแก้ไขนาฬิกาดาวเทียมต่ำกว่า 0.1 นาโนวินาที
2. ความคลาดเคลื่อนเนื่องจากชั้นบรรยากาศโโทรโพสเฟียร์ ( $d_{trop}$ ) จะใช้แบบจำลองความคลาดเคลื่อนในชั้นบรรยากาศโโทรโพสเฟียร์แบบ Saastamoinen Model ร่วมกับ Neill Mapping Function
3. ความคลาดเคลื่อนเนื่องจากชั้นบรรยากาศไอโอดีฟลีร์ของคลื่นสั่ง ( $d_{ion}$ ) จะใช้แบบจำลองของ Ionosphere-Free Combination (L3) จากการทดสอบกันระหว่างชั้น L1 และ L2
4. ความคลาดเคลื่อนจากคลื่นหดหายวิช (Multipath error) จะเลือกใช้จุดตั้งรับที่ไม่มีพื้นผิวสะท้อนอยู่ใกล้เคียง หรือเลือกเสาอากาศที่ออกแบบเฉพาะ

#### **2.4.3 ความคลาดเคลื่อนในการรังวัดดาวเทียมระบบ GNSS**

ความคลาดเคลื่อนในการรังวัดดาวเทียมระบบ GNSS มีข้อจำกัดอยู่หลายประการที่จะส่งผลต่อความถูกต้องของตำแหน่งที่ทำการรังวัดบนพื้นผิวโลก การศึกษาถึงปัจจัยต่างๆ ที่ส่งผลต่อการรังวัดด้วยดาวเทียมจึงมีความสำคัญมาก เพื่อที่จะนำไปประยุกต์ใช้งานให้เกิดความผิดพลาดให้น้อยที่สุด ความคลาดเคลื่อนที่ส่งผลต่อความถูกต้องของเชิงตำแหน่งโดยการรังวัดด้วยดาวเทียมมีดังนี้

##### **1. ความคลาดเคลื่อนของวงโคจรดาวเทียม (Ephemeris Error)**

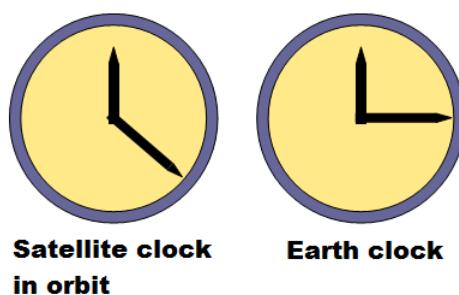
วงโคจรดาวเทียมที่ผู้ใช้รับมานั้น เป็นวงโคจรที่ได้จากการคำนวณล่วงหน้า ดังนั้น ตำแหน่งดาวเทียมจึงเป็นค่าจากการคำนวณล่วงหน้า โดยอาศัยรูปจำลองของแรงโน้มถ่วงต่างๆ ที่กระทำต่อดาวเทียม เช่น แรงดึงดูดจากดวงจันทร์ แรงดึงดูดจากดาวอาทิตย์ ลมสูริยะ ซึ่งส่งผลต่อตำแหน่งของวงโคจรของดาวเทียม รูปจำลองที่ใช้อ้างไม่สมบูรณ์ ฉะนั้นตำแหน่งของดาวเทียมที่ส่งผลกระทบมาพร้อมสัญญาณดาวเทียมนี้จึงไม่ถูกต้อง ดังนั้นส่วนควบคุมจึงจำเป็นต้องปรับแก้ให้เที่ยงตรงอยู่เสมอ



รูปที่ 2.24 ภาพอธิบายการเกิด Ephemeris Error

## 2. ความคลาดเคลื่อนของนาฬิกาบนดาวเทียม (GPS Clock Error)

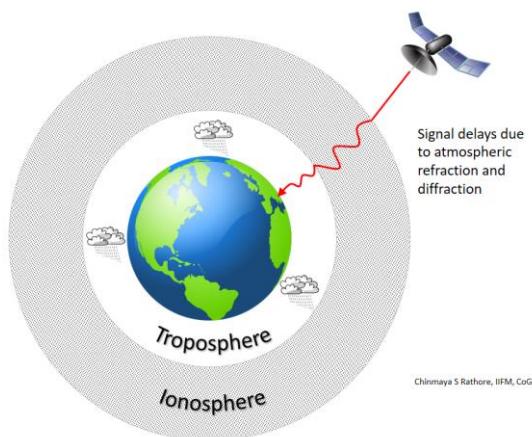
เวลา GPS คือ เวลาที่ใช้เป็นมาตรฐานอ้างอิงในระบบ GPS โดยมีสถานีควบคุมหลักเป็นผู้ดูแลรักษาเวลา GPS ให้มีความถูกต้อง โดยปกติเวลา GPS และเวลามาตรฐานสากล (Coordinated Universal Time) จะถูกรักษาให้ต่างกันไม่เกิน  $100\text{ ns}$  หรือ  $10^{-7}$  วินาที แต่ถึงแม้ว่านาฬิกาจะต้องที่เป็นตัวกำหนดเวลาบนดาวเทียมจะมีความถูกต้องสูง แต่ก็ยังคงมีความผิดพลาดอยู่ประมาณ  $3\text{ ns}$  ส่งผลให้คลื่นที่ส่งมาผิดพลาดด้วย นอกจากความคลาดเคลื่อนในส่วนของการเทียบเวลาแล้ว ยังมีความคลาดเคลื่อนที่เกิดจากความไม่เสถียรของมาตรฐานความถี่ของนาฬิกาบนดาวเทียมที่เรียกว่า ดริฟท์ (Drift)



รูปที่ 2.25 ภาพอธิบาย GPS Clock Error

### 3. ความคลาดเคลื่อนของการหักเหในชั้นบรรยากาศ (Atmospheric Errors)

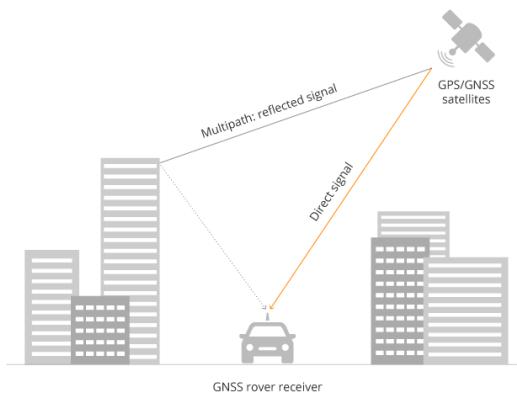
บรรยากาศของโลกถูกแบ่งออกเป็นหลายชั้น แต่ละชั้นมีคุณสมบัติที่แตกต่างกัน ชั้นบรรยากาศที่ติดกับพื้นโลก เรียกว่า โตรโพสเฟียร์ (Troposphere) เป็นชั้นที่มีปรากฏการณ์ทางธรรมชาติ เช่น ฝน เมฆ พายุ ขอบเขตของโตรโพสเฟียร์ สูงถึง 16 - 17 กม. จากพื้นผิวโลก ส่วนชั้นบรรยากาศซึ่งครอบคลุมบริเวณตั้งแต่ระดับ ความสูงราว 50 กม. ถึง 1000 กม. เรียกว่า ไอโโโนสเฟียร์ (Ionosphere) ชั้นบรรยากาศที่มีผลต่อการหักเหของคลื่นวิทยุหรือทำให้เส้นทางเคลื่อนที่เบี่ยงเบนไป คือ Troposphere และ Ionosphere



รูปที่ 2.26 ภาพอธิบายการเกิด Atmospheric Error

### 4. ความคลาดเคลื่อนเนื่องจากเกิดคลื่นสะท้อน (Multipath Errors)

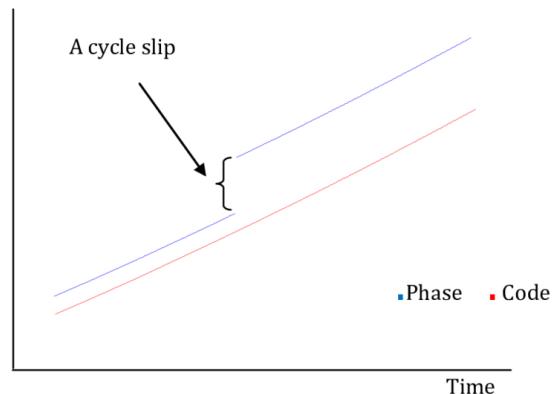
คลื่นสะท้อน เกิดจากคลื่นสัญญาณ GPS เดินทางไปกระทบวัตถุเกิดการสะท้อนของคลื่น ก่อนเข้าสู่เครื่องรับสัญญาณ โดยวัตถุสะท้อนคลื่น เช่น ผวน้ำ ผังคอนกรีต ลังผลักบารหัสและคลื่นส่อง ซึ่งผลของการเกิดคลื่นสะท้อนมีขนาดเป็น 100 เท่าของคลื่นส่อง การสะท้อนคลื่นมีผลโดยตรงต่อข้อมูลระยะทางระหว่างดาวเทียมและเครื่องรับสัญญาณผิดไปจากเดิม จึงส่งผลให้ค่าพิกัดตำแหน่งผิดพลาดไปด้วย และยังเป็นสาเหตุทำให้คลื่นหลุดได้ง่าย แต่มีการแก้ไขหลายวิธีดังนี้ เช่น เลือกจุดตั้งรับที่ไม่มีพื้นผิวสะท้อนอยู่ใกล้เคียง หรือเลือกเสาอากาศที่ออกแบบเฉพาะ เช่น เสาอากาศที่มีแผ่นกราวด์และใช้เวลาการรังสรรคให้มากยิ่งขึ้น



รูปที่ 2.27 ภาพอธิบายการเกิด Multipath Error

### 5. การเกิดคลื่นหลุด (Cycle Slip)

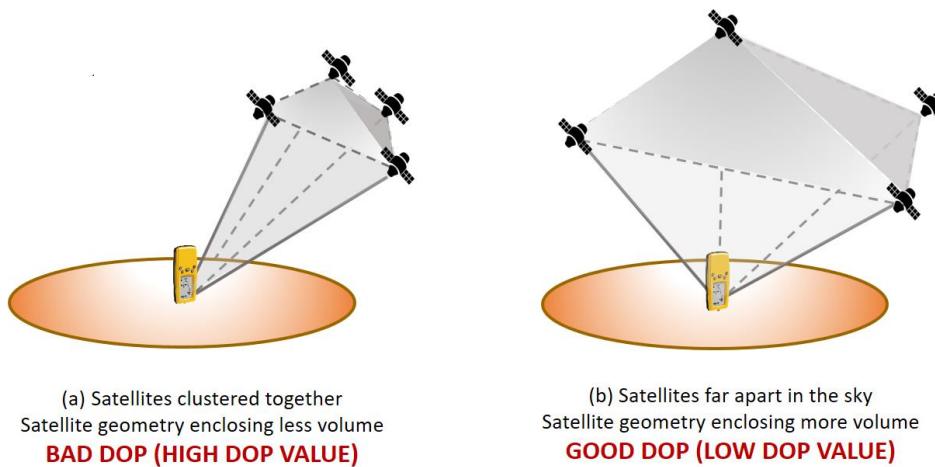
คลื่นหลุด คือ การขาดหายไปเป็นช่วงๆของคลื่น ทำให้เครื่องรับไม่สามารถบันทึกข้อมูลได้อีกต่อเนื่อง โดยมีสาเหตุหลายประการ ได้แก่ มีสิ่งกีดขวางการเดินทางของสัญญาณดาวเทียม เช่น ต้นไม้ อาคาร หรือ Noise ของสัญญาณขนาดใหญ่ เช่น สัญญาณลูก Gron กวนจากคลื่นวิทยุ หรือการเกิดคลื่นสะท้อน เป็นต้น



รูปที่ 2.28 ภาพแสดงการเกิด Cycle Slip Error

ผลกระทบจากการเกิดคลื่นหลุด คือ หลังจากที่เครื่องรับสามารถรับสัญญาณและล็อกสัญญาณได้อีก ครั้งส่วนของเฟสคลื่นส่งยังคงเหมือนเดิม แต่จำนวนเต็มรอบของลูกคลื่นจะเริ่มนับใหม่ ทำให้ในการประมวลผลต้องทำการหาค่าเลขจำนวนเต็มของลูกคลื่นใหม่ การแก้ไขการเกิดคลื่นหลุดทำได้โดยการใช้ซอฟต์แวร์ในการประมวลผลช่วยทำการแก้ไข การหลุดของสัญญาณเป็นการหายไปของสัญญาณขณะรังวัด ไม่สามารถกลับมา=rับสัญญาณของดาวเทียมที่รังวัดไปแล้ว ได้อีก ซึ่งเกิดจากการที่กลุ่มดาวเทียมขณะรังวัดมีน้อยกว่า 4 ดวง และสิ่งกีดขวางในการเดินทางของสัญญาณ เช่น อาคาร ต้นไม้ เป็นต้น

6. ความคลาดเคลื่อนเนื่องจากนาฬิกาของเครื่องรับสัญญาณดาวเทียม (Receiver Clock Errors)  
เครื่องรับสัญญาณไม่จำเป็นต้องมีนาฬิกาที่มีความถูกต้องสูงมาก ทั้งนี้ เพราะเวลาที่ถูกต้องสามารถหาได้จากสัญญาณดาวเทียมที่เป็นข้อมูลเวลา
7. ความคลาดเคลื่อนของเครื่องรับ (Receiver Error)  
ความคลาดเคลื่อนของเครื่องรับ (Noise) มีสาเหตุการเกิดหลายอย่าง เช่น ความไม่เสถียรของอสซิลเลเตอร์ จุดศูนย์กลางเฟสของเสาอากาศ การ BIAS ระหว่างช่องรับสัญญาณ Noise ในการรังวัดขึ้นอยู่กับความเร็วของสัญญาณที่ได้รับ
8. ความผิดพลาดในการวัดความสูงของเสาอากาศ (Measurement Error)  
เป็นความผิดพลาดที่ใหญ่ที่สุดที่เกิดจากผู้ปฏิบัติการรังวัดความสูงเสาอากาศในสนามมี ความสำคัญ เป็นอย่างยิ่ง เนื่องจากสัญญาณดาวเทียมที่รังวัดได้จะถูกบันทึกและคำนวณบนเสาอากาศ ดังนั้นค่าความสูงของเสาอากาศที่วัดได้จะใช้เป็นค่าในการthonค่าความสูงของเสาอากาศลงสู่หัวหมุดหลักฐาน การวัดค่าความสูงของเสาอากาศจะมีผลกระทบโดยตรงต่ความสูงของหมุดหลักฐาน ดังนั้นผู้ปฏิบัติการระลึกถึงและระมัดระวังให้มากที่สุด
9. ความคลาดเคลื่อนเนื่องจากการวางแผนด้วยดาวเทียม (Dilution of Precision: DOP)  
การหาค่าพิกัดของตำแหน่งของวัตถุ นอกจากต้องรับสัญญาณดาวเทียมให้ได้อย่างน้อย 4 ดวง การวางแผนตำแหน่งของดาวเทียมบนท้องฟ้าก็เป็นส่วนสำคัญ ค่า DOP เป็นค่าที่ได้จากการทางคณิตศาสตร์ ซึ่งบ่งบอกความไม่แน่นอนด้านตำแหน่ง ค่า DOP ที่สูงให้ค่าความไม่แน่นอนของตำแหน่งมากกว่า ค่า DOP ที่ต่ำ โดยแบ่งค่า DOP ได้ดังนี้
  1. HDOP หรือ Horizontal Dilution of Precision คือ ความไม่แน่นอนในองค์ประกอบด้านตำแหน่งทางราบ
  2. VDOP หรือ Vertical Dilution of Precision คือ ความไม่แน่นอนในองค์ประกอบด้านความสูง
  3. PDOP หรือ Position Dilution of Precision คือ ความไม่แน่นอนในองค์ประกอบด้าน ตำแหน่งในสามมิติ โดย  $PDOP = HDOP + VDOP$
  4. TDOP หรือ Time Dilution of Precision คือ ความไม่แน่นอนในองค์ประกอบด้านเวลา
  5. GDOP หรือ Geometry Dilution of Precision คือ ความไม่แน่นอนในองค์ประกอบด้านเรขาคณิต โดย  $GDOP = PDOP + TDOP$  ซึ่งถ้าแบ่งท้องฟ้าออกเป็น 4 Quadrant ดาวเทียมจะต้องอยู่เหนือนี้ หรือ เครื่องรับ สัญญาณดาวเทียมกระจายอยู่ดังแสดงในภาพที่ 52 จึงจะให้ค่า DOP ที่ดี



รูปที่ 2.29 ภาพอธิบายการวางแผนตำแหน่งของดาวเทียมเพื่อให้ได้ค่า DOP ที่ดี

การที่จะทำการรังวัดเพื่อให้ได้ค่าพิกัดตำแหน่งที่ถูกต้องที่สุด นอกจากปัจจัยต่างๆ ที่ได้กล่าวมาแล้วในข้างต้น ความพร้อมของผู้ใช้งานและการวางแผนที่เป็นระบบ จะช่วยให้การรังวัดมีประสิทธิภาพและได้ความถูกต้องมากยิ่งขึ้น

ตารางที่ 2.2 ตารางแสดงค่าความคลาดเคลื่อนในการรังวัดดาวเทียมระบบ GNSS

Segment Source	Error Description	Error value (m)
Space	GPS satellite clock stability	3.0
	GPS satellite perturbations	1.0
	Other (thermal, radiation, etc.)	0.5
Control	Ephemeris prediction	4.2
	Other (position control, etc.)	0.9
User	Ionospheric delay	5.0
	Tropospheric delay	1.5
	Receiver noise and resolution	1.5
	Multipath	2.5
System UERE	Other (interchannel bias, etc.)	0.5
	<b>Total (rss)</b>	<b>8.0</b>

#### 2.4.4 อุปกรณ์หลักในการรับ-ส่งข้อมูลกับสัญญาณดาวเทียมในระบบ GNSS

สำหรับการรับ-ส่งข้อมูล หรือการระบุตำแหน่งของการวัดในระบบ GNSS นั้นจำเป็นต้องใช้อุปกรณ์ เลพาทางในการเขื่อมต่อ ซึ่งมีอุปกรณ์ที่สำคัญอยู่ 2 ส่วนดังนี้

##### 1. เสาอากาศ หรือ Antenna

ในเรื่องของระบบคลื่นวิทยุนั้น เสาอากาศ หรือ Antenna จะทำหน้าที่ปรับรูปร่างและรวมคลื่นวิทยุให้ ส่งไปยังทิศทางที่ต้องการ เมื่อนلنกสำหรับคลื่นความถี่วิทยุหรือ RF โดยเฉพาะ คลื่นวิทยุทุกระบบ นั้น ไม่ว่าจะเป็น Wi-Fi, สัญญาณมือถือ, อุปกรณ์ไร้สาย, ต่างต้องใช้เสาอากาศบนอุปกรณ์ทั้งส่งและ รับคลื่นทั้งสิ้น เสาอากาศบนแต่ละอุปกรณ์นั้นมีการปรับจูนให้ทำงานสำหรับความถี่ที่จำเพาะ



รูปที่ 2.30 ภาพตัวอย่างเสาอากาศ Antenna ในการรับสัญญาณ GNSS

##### 2. เครื่องรับสัญญาณ หรือ Receiver

GPS หรือ GNSS Receiver มีหน้าที่หลัก คือรับสัญญาณจากดาวเทียม และนำมาแปลงเป็นพิกัดของ ตำแหน่งที่ตัวเครื่องอยู่บนพื้นโลก โดยสิ่งที่ Receiver สามารถคำนวณและให้คำตอบจะมี 3 ค่า คือ พิกัด ความเร็วในการเคลื่อนที่ และเวลา ในส่วนของฟังก์ชันอื่นๆ เช่น ตำแหน่งบนแผนที่ ระยะทาง ระหว่างสองจุดบนพื้นโลก หรือการคำนวณหาเวลาที่จะเดินทาง ไปถึงปลายทาง จะเป็นตัวเสริม ความสามารถของระบบซึ่งแล้วแต่ว่าบริษัทที่ผลิตต้องการจะเสริมบริการในส่วนใดบ้าง โดยเครื่องรับสัญญาณจะมีส่วนสำคัญ 4 ส่วน คือ

1. วงจรรับสัญญาณ GNSS ซึ่งจะกำหนดความถี่ให้ตรงกับ L1 และ L2 โดยวงจรจะทำการ Demodulate เพื่อให้ได้ Pseudo Random Code ที่ดาวเทียมส่งมา
2. Almanac เป็นข้อมูลที่บอกถึงสภาพของดาวเทียม ตำแหน่ง วงโคจรของดาวเทียมในระบบทุกดวง คร่าวๆ อุปกรณ์รับสัญญาณจีพีเอส จะรับข้อมูล Almanac จากดาวเทียมดวงใดๆ ที่สามารถรับ

สัญญาณได้ แล้วใช้ข้อมูลดังกล่าวเพื่อเลือกรับสัญญาณดาวเทียมที่สามารถใช้ในการคำนวณพิกัดตำแหน่งได้

3. Pseudo Random Code Generator ทำหน้าที่เป็นส่วนสร้าง Code ที่ตรงกับที่มีอยู่ในดาวเทียมแต่ละดวง
4. Microprocessor ทำหน้าที่ในการประมวลผลข้อมูล และคำนวณหาตำแหน่ง ความเร็ว และเวลา



รูปที่ 2.31 ภาพตัวอย่างวงจรรับสัญญาณ GNSS

เนื่องจากในปัจจุบันเทคโนโลยีการระบุตำแหน่งได้พัฒนาขึ้นมากเมื่อเทียบกับในอดีต เพื่อใช้ในลักษณะงานที่แตกต่างกันไปตามผู้ใช้งาน ดังนั้นในปัจจุบันจึงสามารถแบ่งประเภทของเครื่องรับสัญญาณได้ 2 ประเภท ดังนี้

#### 2.1 เครื่องรับสัญญาณดาวเทียม GNSS ชนิดทั่วไป หรือชนิดรัง棍

เครื่องรับสัญญาณ GNSS ชนิดนี้ ให้ความถูกต้องของพิกัดอยู่ในระดับเมตร จึงเหมาะสมกับงานติดตามตำแหน่ง (Location Tracking) เท่านั้น โดยหลักการจะใช้ Code-based (Pseudo range) ใน การคำนวณหาค่าพิกัด ความถูกต้องในเชิงพิกัดจะอยู่ในช่วง  $\pm 5$  เมตร จึงไม่เหมาะสมกับงานรังวัด และทำแผนที่ ดังนั้นเครื่องรับสัญญาณประเภทนี้จึงมีราคาถูก



รูปที่ 2.32 ภาพตัวอย่างเครื่องรับสัญญาณดาวเทียม GNSS ชนิดทั่วไป

## 2.2 เครื่องรับสัญญาณดาวเทียม GNSS ชนิด RTK หรือชนิครังวัด

เครื่องรับสัญญาณ GNSS ชนิดนี้เป็นเครื่องรับสัญญาณที่ให้ความถูกต้องของพิกัดสูงมากอยู่ในระดับเซนติเมตร โดยความถูกต้องในเชิงพิกัดจะอยู่ในช่วง  $\pm 3$  เซนติเมตร และน้อยกว่า 1 เซนติเมตรในเครื่องที่มีคุณภาพสูงหรือเทคนิคการรังแบบ Static ในโหมด RTK โดยหลักการจะใช้ Carrier phase ในการคำนวณหาค่าพิกัด ด้วยความละเอียดระดับนี้ เครื่องรับสัญญาณประเภท RTK จึงเหมาะสมกับงานทำแผนที่และงานรังวัดที่ต้องการความถูกต้องสูง ดังนั้นจึงมีราคาสูงกว่า เครื่องรับสัญญาณดาวเทียม GNSS ชนิดทั่วไป



รูปที่ 2.33 ภาพตัวอย่างเครื่องรับสัญญาณดาวเทียม GNSS ชนิด RTK

## 2.4.5 เทคนิคการวัดตำแหน่งแบบรังวัด

เนื่องจากการตรวจจับพิกัดและนำทางรถจำเป็นต้องใช้ความละเอียดและจำเป็นต้องอาศัยความถูกต้องในเชิงพิกัดที่สูง ดังนั้นจึงจำเป็นต้องเลือกใช้เครื่องรับสัญญาณดาวเทียม GNSS ชนิดรังวัด เพื่อความถูกต้องในการกำหนดค่าพิกัด โดยลักษณะหรือเทคนิคการวัดตำแหน่งแบบรังวัดที่นิยมใช้แบ่งได้ 2 วิธี ได้แก่

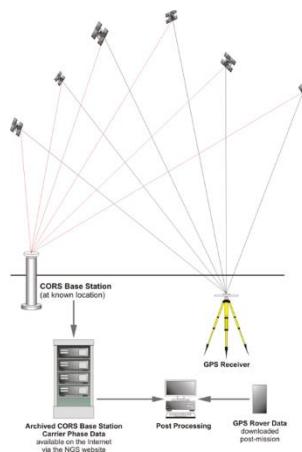
### 1. งานรังวัดแบบสถิติ (Static)

งานรังวัดแบบสถิติ เป็นงานรังวัดพื้นฐานที่ให้ค่าพิกัดที่แม่นยำสูง ซึ่งใช้เครื่องรับสัญญาณอย่างน้อย 2 เครื่อง โดยเครื่องที่ 1 จะถูกวางไว้บนหมุดที่ทราบค่าพิกัด (Base Station) ส่วนเครื่องที่ 2 จะถูกนำไปวางจุดที่ต้องการทราบพิกัด (Rover Station) ซึ่ง Base Station และ Rover Station จะต้องรับข้อมูลจากดาวเทียมก่อรากันและช่วงเวลาเดียวกันอย่างน้อย 4 ดาว งานรังวัดในลักษณะนี้จะต้องใช้ระยะเวลาที่ทำการรับสัญญาณที่เพียงพอ ซึ่งจะใช้เวลาตั้งแต่ในระดับชั่วโมงไปจนถึงวัน และต้องทำการประมวลผลด้วยซอฟแวร์ภาษาหนัง โดยปกติงานในลักษณะนี้จะใช้สำหรับงานที่ต้องการค่าความถูกต้องสูงมากๆ ในระดับมิลลิเมตรไปจนถึงเซนติเมตร ในงานสำรวจและทำแผนที่มักใช้มีต่อสื้น

ฐานเป็นโครงข่าย Network โดยระบบทางระหว่างเครื่องรับสามารถไกลได้ถึงหลายพันกิโลเมตร (ต้องใช้ซอฟต์แวร์สำหรับงานวิจัยประมวลผล)

ระยะเวลาเก็บ Log (Session Time) ของ Rover สามารถแบ่งได้ดังนี้

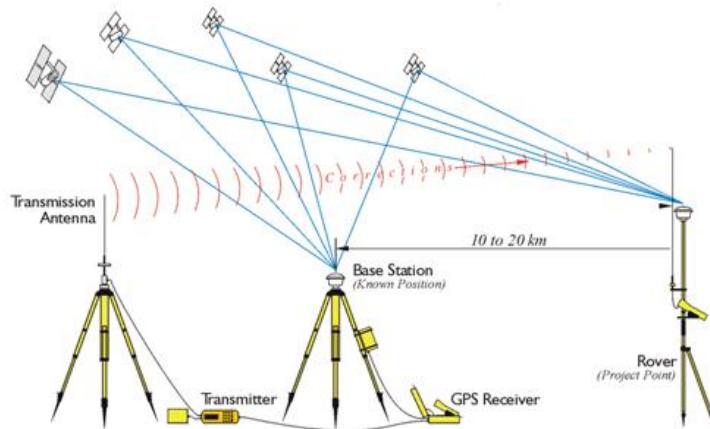
1. ใช้เวลามากกว่า 1 ชั่วโมง ความถูกต้องของพิกัดจะได้ระดับมิลลิเมตร เรียกว่าแบบ Static
2. ใช้เวลา 30-60 นาที ความถูกต้องของพิกัดจะได้ระดับ 1-3 เซนติเมตร เรียกว่าแบบ Rapid Static
3. ใช้เวลา 15-30 นาที ความถูกต้องของพิกัดจะได้รับระดับ 3-4 เซนติเมตร เรียกว่าแบบ Fast Static



รูปที่ 2.34 ภาพแสดงเทคนิคการรังวัดแบบสติ๊ก

## 2. งานรังวัดแบบจรน์ (Real Time Kinematic - RTK)

งานรังวัดแบบ RTK เป็นวิธีที่รวมเครื่องรับสัญญาณอย่างน้อย 2 เครื่อง โดยเครื่องที่หนึ่งจะถูกวางไว้บนหมุดที่ทราบค่าพิกัด (Base Station) ส่วนอีกเครื่องจะถูกนำไปวางจุดที่ต้องการทราบค่าพิกัด (Rover Station) และต้องมีอุปกรณ์สื่อสารระหว่าง Base Station และ Rover Station เช่น มือถือ วิทยุ ซึ่ง Base Station จะส่งค่าปรับแก้ไปให้ Rover Station โดยเครื่องรับที่ Base Station และ Rover Station จะต้องรับข้อมูลจากดาวเทียมกลุ่มเดียวกันและช่วงเวลาเดียวกันอย่างน้อย 5 ดวง งานรังวัดในลักษณะนี้จะใช้เวลาตั้งแต่ในระดับหลายวินาทีไปจนถึงหลายนาที โดยปกติงานในลักษณะนี้จะใช้สำหรับงานที่ต้องการค่าความถูกต้องสูงอยู่ในระดับเซนติเมตร ไปจนถึงหลายเซนติเมตร จะเห็นได้ว่างงานรังวัดแบบจรน์จะมีความคลาดเคลื่อนของพิกัดมากกว่างานรังวัดแบบสติ๊ก แต่ข้อดีของงานรังวัดในรูปแบบนี้คือ ไม่จำเป็นต้องทำการประมวลผลข้อมูลภายหลัง และจะได้พิกัดโดยตรง ซึ่งต้องแลกมาด้วยข้อเสียคือ สภาพแวดล้อมระหว่าง Base Station และ Rover Station มีผลต่อการรับ-ส่งข้อมูล หรือคุณภาพของสัญญาณมาก



รูปที่ 2.35 ภาพแสดงเทคนิคการรังวัดแบบจลน์

### 3. งานรังวัดแบบ Network RTK

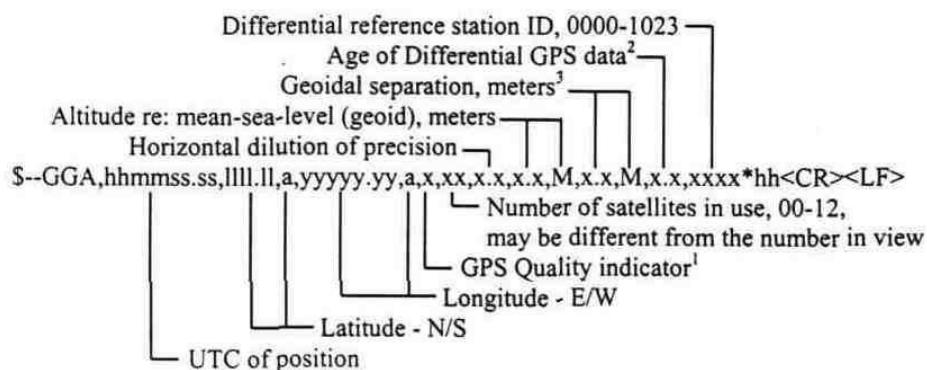
งานรังวัดในลักษณะนี้จะประยุกต์มาจากการงานรังวัดแบบจลน์ โดยจะแตกต่างกับวิธีแบบจลน์ ในเรื่องของค่าปรับแก้ที่จะส่งมาให้ Rover Station ผ่านทางระบบอินเตอร์เน็ตเท่านั้น หลักการคือผู้ให้บริการจะต้องมีการวาง Base Station ให้กระจายครอบคลุมพื้นที่การให้บริการ ตัวอย่างผู้ให้บริการในประเทศไทย ได้แก่ 1. กรมที่ดิน 2. กรมแผนที่ทหาร 3. หน่วยงานราชการอื่นๆ ซึ่งจะมีเซิฟเวอร์กลางที่รวบรวมค่าปรับแก้จากสถานีต่างๆ ที่เรียกว่า NTRIP Server โดยจะทำหน้าที่รวมรวมและกระจายค่าปรับแก้ให้กับ Rover Station ผ่านทางอินเตอร์เน็ต การทำงานของระบบจะเป็นการทำงานตลอด 24 ชั่วโมง เรียกว่าระบบ CORS (Continuous Operation Reference Station)



รูปที่ 2.36 ภาพแสดงเทคนิคการรังวัดแบบ Network RTK

## 2.4.6 มาตรฐานการสื่อสารโปรโตคอล NMEA ของการระบุตำแหน่งแบบ GPS และ GNSS

NMEA ย่อมาจาก Nation Maritime Electronics Association ซึ่งเป็นสมาคมที่มุ่งเน้นศึกษาและพัฒนา อุปกรณ์อิเล็กทรอนิกส์ เพื่อการเชื่อมต่อและทำงานร่วมกันของอุปกรณ์ โดยอุปกรณ์เหล่านี้ เมื่อเชื่อมต่อ และทำงานร่วมกันต้องสามารถเข้าใจกันได้ หรือสื่อสารโดยใช้ภาษาเดียวกัน NMEA จึงพัฒนามาตรฐาน ในการสื่อสารข้อมูลระหว่างอุปกรณ์ดังกล่าว เรียกว่า NMEA Standard



รูปที่ 2.37 ตัวอย่างรูปแบบประโยคของprotoคอล NMEA

**NMEA Standard** กี เช่นเดียวกับ protoคอลภาษาอื่นๆที่ได้มีการพัฒนาเรื่อยๆ โดยเริ่มตั้งแต่ NMEA-0180, NMEA-0182 จนถึง NMEA-0183 โดยที่การใช้งานของ NMEA-0180 และ NMEA-0182 ค่อนข้างมี ข้อจำกัดและจะเน้นทางการสื่อสารระหว่าง Loran C กับ Autopilot ดังนั้น ได้มีการพัฒนาให้สามารถใช้ งานได้อย่างกว้างขวางขึ้น โดยครอบคลุมอุปกรณ์อิเล็กทรอนิกส์ในการเดินเรือจนกลายมาเป็น NMEA-0183 ใช้อักษร ASCII และการสื่อสารข้อมูลแบบอนุกรมในการส่งข้อมูล จากอุปกรณ์ตัวหนึ่งไปยัง อุปกรณ์รับตัวหนึ่งหรือหลายๆตัว

มาตรฐาน NMEA-0183 ในมาตรฐานนี้ ตัวอักษรที่ใช้คือ ASCII Text ซึ่งสามารถพิมพ์ได้ (รวมไปถึง Carriage Return and Line Feed) NMEA-0183 นั้นส่งข้อมูลด้วยอัตรา 4800 baud ข้อมูลจะถูกส่งในรูปของ ประโยค (Sentences)

protoคอลที่สำคัญของเครื่อง GPS จะอยู่ในชุด NMEA ซึ่งเป็น protoคอลมาตรฐานของ GPS แต่ภายใน อาจมีprotoคอลอื่นประกอบอีกมากมาย แต่protoคอลที่ใช้งานหลักมีดังนี้

**GGA** – รูปแบบที่แสดงว่าข้อมูลของ GPS เพียงพอที่จะแสดงพิกัดได้สามมิติ (3D) ซึ่งดาวเทียมที่รับได้ ต้องมากถึง 4 ดวงขึ้นไป ภาษาอังกฤษเรียกว่า Fix data.

ตัวอย่าง : \$ GPGGA ,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,\*47

ความหมาย :

GGA - Global Positioning System Fix Data

123519 - ข้อมูล Fix เมื่อเวลา 12:35:19 UTC

4807.038, N - ค่าพิกัดแล็ตติจูดด้านเหนือสีน้ำเงิน 48 deg 07.038 N

01131.000, E - ค่าพิกัดลองจิจูดด้านตะวันออก 11 deg 31.000 E

1 – คุณภาพของข้อมูล Fix:

0 = ข้อมูลไม่ถูกต้อง

1 = GPS fix (SPS)

2 = DGPS fix

3 = PPS fix

4 = Real Time Kinematic

5 = Float RTK

6 = estimated (dead reckoning) (2.3 feature)

7 = Manual input mode

8 = Simulation mode

08 – จำนวนดาวเทียม GPS ที่รับได้

0.9 – ค่าความคลาดเคลื่อนการระบุตำแหน่งแนวราบ

545.4, M – ค่าความสูงเหนือระดับน้ำทะเลเป็นเมตร

46.9, M – ความสูงของจีอยด์เหนือทรง WGS84

(ช่องว่าง) – เวลาเป็นวินาทีนับจากที่ได้รับค่า fix รูปแบบ DGPS

(ช่องว่าง) – แสดงหมายเลขสถานีของ DGPS

\*47 – ค่า checksum นำหน้าด้วย \*

ในตำแหน่งของ GPS Quality Indicator จะประกอบด้วยหมายเลขต่างๆ ซึ่งหมายเลขอแต่ละชนิดจะบ่งบอกถึงสถานะการเชื่อมต่อและคุณภาพความละเอียดของพิกัดที่วัดได้ โดยจะมีรายละเอียด ดังนี้

ตารางที่ 2.3 ตารางแสดงรายละเอียดของค่าต่างๆ ใน GPS Quality Indicator

Value	Name	Description
0	Invalid	GNSS can't determinate the position (No signal)
1	GPS	Position is gathered only from 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.

GSA – รูปแบบที่แสดงรายละเอียดของข้อมูล Fix จำนวนดาวเทียมที่ใช้งานได้ รวมถึงค่าความคลาดเคลื่อน DOP (dilution of precision) ซึ่งตัวเลขน้อยๆ จะเป็นค่าที่ดีมีความถูกต้องสูง.

ตัวอย่าง : \$GPGSA,A,3,19,28,14,18,27,22,31,39,,,1.7,1.0,1.3\*35

ความหมาย :

GSA – Satellite status

A – คือ mode ของสถานะของข้อมูล fix เป็น A – Automatic, M = Manual

3 – คือตัวเลขแสดงสถานการ fix ประกอบไปด้วยค่า:

1 = ข้อมูลไม่ fix

2 = ข้อมูล fix แบบสองมิติ

3 = ข้อมูล fix แบบสามมิติ

19,28,14,18,27,22,31,39 – คือหมายเลขดาวเทียมที่รับได้ ในที่นี่รับได้ 8 ดวงและ

ตามด้วยเครื่องหมายคอมม่าว่างๆ อีก 4 ซึ่งเครื่อง GPS จะรับได้สูงสุด 12 ดวง

1.7 – ค่าความคลาดเคลื่อนในการระบุตำแหน่ง PDOP (dilution of precision)

1.0 – ค่าความคลาดเคลื่อนในการระบุตำแหน่งทางราบ (HDOP)

1.3 – ค่าความคลาดเคลื่อนในการระบุตำแหน่งทางดิ่ง (VDOP)

\*35 – ค่า checksum นำหน้าด้วย \*

**GSV** – รูปแบบที่แสดงรายละเอียดของ GPS แต่ละดวง เช่นระดับความสูง (Elevation) อะซิมัทและ SNR (Signal to Noise Ratio) ซึ่ง เทียบได้กับความแรงของสัญญาณ SNR มีค่าตั้งแต่ 0 ถึง 99 ซึ่งค่ามากเป็นค่าที่ดี ในบางขณะเครื่อง GPS อาจจะรับสัญญาณได้เต็มที่ทั้งหมด 12 ดวง การส่งข้อมูลจำนวนมากไปถ้าต้องแสดงในบรรทัดเดียว สมาคม NMEA จึงออกแบบให้รูปแบบ GSV สามารถแสดงข้อมูลดาวเทียมได้เต็มที่ ประมาณ 4 บรรทัดละ 4 ดวงเท่านั้น ดังนั้นถ้ารับสัญญาณดาวเทียมได้ทั้ง 12 ดวงจะได้รับประโยชน์อย่างมาก

ตัวอย่าง : \$GPGSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45\*75

ความหมาย :

GSV – Satellites in view

2 – จำนวนประโยชน์ข้อมูลความ (ในที่นี้รับดาวเทียมได้ 8 ดวง จึงใช้แค่ 2 บรรทัดเท่านั้น)

1 - ประโยชน์ที่ 1 จากทั้งหมด 2 ประโยชน์

08 – จำนวนดาวเทียมที่รับสัญญาณได้

01 – หมายเลขดาวเทียม GPS

40 – ระดับความสูง หน่วยเป็นองศา

083 – อะซิมัท (ทิศเหนือ 0 ทิศตะวันออก 90 ทิศใต้ 180 ทิศตะวันตก 270)

46 – SNR – ความแรงของสัญญาณ ค่าสูงเป็นค่าที่ดี

02,17,308,41 – ดาวเทียมหมายเลข 2 พร้อมข้อมูล elevation, azimuth และ SNR

12,07,344,39 – ดาวเทียมหมายเลข 12 พร้อมข้อมูล elevation, azimuth และ SNR

14,22,228,45 – ดาวเทียมหมายเลข 14 พร้อมข้อมูล elevation, azimuth และ SNR

\*75 – ค่า checksum นำหน้าด้วย \*

**RMC** – รูปแบบที่แสดงรายละเอียดของ GPS เรื่องความเร็ว (velocity) ค่าพิกัด เวลา ตลอดจนทิศทาง

ตัวอย่าง : \$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W\*6A

ความหมาย :

RMC – Recommended Minimum sentence C

123519 – ข้อมูล fix เมื่อเวลา 12:35:19 UTC

A – สถานะ A= กำลังใช้งาน หรือ V= ยกเว้น

4807.038,N – ค่าแก็ตติจูด 48 องศา 07.038 เหนือ

01131.000,E – ค่าลองจิจูด 11 องศา 31.000 ตะวันออก

022.4 – ความเร็วที่ยินยอมพื้นดิน หน่วยเป็น knot (ไมล์ทะเลต่อชั่วโมง)

#### 084.4 – มุมของทิศทางเที่ยบกับเหนือจริง

230394 – วันที่ – 23 มีนาคม 1994

003.1,W – มุมต่างระหว่างเหนือจริงกับเหนือแม่เหล็ก

\*6A – ค่า checksum นำหน้าด้วยเครื่องหมาย \*

```

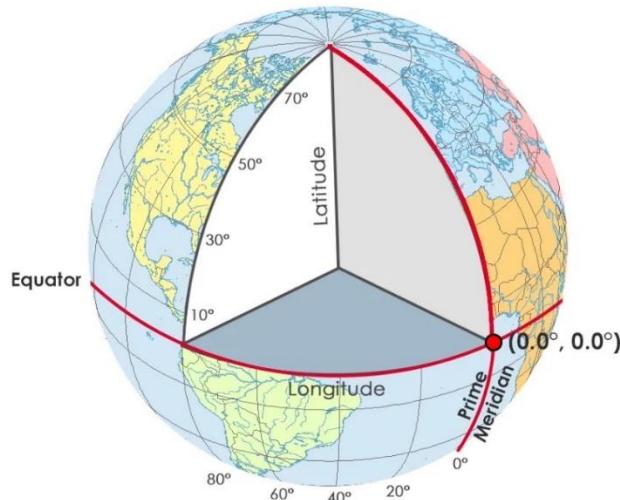
Text Console
16:25:28 $GPGSV,3,2,12,11,07,217,09,14,52,009,44,17,41,335,44,19,31,294,41,1°61
16:25:28 $GPGSV,3,3,12,30,64,220,46,40,36,257,41,41,64,233,48,50,55,115,42,1°6F
16:25:28 $GPGSV,2,1,07,01,25,033,41,03,26,087,44,06,41,220,45,07,39,171,31,6°66
16:25:28 $GPGSV,2,2,07,14,52,009,46,17,41,335,42,30,64,220,46,6°5E
16:25:28 $GPGSV,1,1,02,09,04,163,,13,08,271,,0°6E
16:25:28 $GLGSV,2,1,07,73,48,132,51,74,71,022,48,75,20,338,47,84,30,010,38,1°77
16:25:28 $GLGSV,2,2,07,85,53,285,51,86,23,236,34,,52,1°77
16:25:28 $GLGSV,2,1,05,73,48,132,50,75,20,338,45,84,30,010,38,85,53,285,47,3°7A
16:25:28 $GAGSV,1,1,03,15,04,034,,19,08,244,,30,03,125,,0°4A
16:25:28 $GNRMC,162528,50,A,1339,03909,N,10029,59298,E,0,934,89,62,101122,,R,V*2F
16:25:29 $GNVTG,89,62,T,M,0,934,N,1,729,K,D*10
16:25:29 $GNGGA,162528,50,1339,03909,N,10029,59295,E,4,12,0,69,0,9,M,-27.8,M,0.2,0000*42
16:25:29 $GNGSA,A,3,03,06,07,14,19,30,17,01,,1.57,0.69,1.41,1°02
16:25:29 $GNGSA,A,3,75,84,73,74,85,,1.57,0.69,1.41,2°0E
16:25:29 $GNGSA,A,3,27,33,26,13,21,,1.57,0.69,1.41,3°0B
16:25:29 $GNGSA,A,3,08,09,40,10,12,24,35,13,07,38,,1.57,0.69,1.41,4°05
16:25:29 $GNGSA,A,3,07,02,03,04,,1.57,0.69,1.41,220,45,07,39,171,32,1°6F
16:25:29 $GPGSV,3,1,12,01,25,033,39,03,26,087,42,06,41,220,44,07,39,171,32,1°6F
16:25:29 $GPGSV,3,2,12,11,07,217,09,14,52,009,44,17,41,335,45,19,31,294,41,1°60
16:25:29 $GPGSV,3,3,12,30,64,220,46,40,36,257,41,41,64,233,48,50,55,115,43,1°6E
16:25:29 $GPGSV,2,1,07,01,25,033,41,03,26,087,44,06,41,220,45,07,39,171,35,6°62
16:25:29 $GPGSV,2,2,07,14,52,009,46,17,41,335,42,30,64,220,46,6°5E
16:25:29 $GPGSV,1,1,02,09,04,163,,13,08,271,,0°6E
16:25:29 $GLGSV,2,1,07,73,48,132,51,74,71,022,48,75,20,338,48,84,30,010,38,1°78
16:25:29 $GLGSV,2,2,07,85,53,285,51,86,23,236,34,,52,1°77
16:25:29 $GLGSV,2,2,05,86,23,236,34,3°40
16:25:29 $GAGSV,2,1,01,70,01,148,,0°43
16:25:29 $GAGSV,2,1,07,01,14,325,39,13,54,022,51,14,27,037,51,21,52,001,50,2°7F
16:25:29 $GNRMC,162528,60,A,1339,03909,N,10029,59298,E,0,965,90,71,101122,,R,V*2F
16:25:29 $GNVTG,90,71,T,M,0,965,N,1,787,K,D*1A
  
```

รูปที่ 2.38 ตัวอย่างรูปแบบประไป็คของโปรโตคอล NMEA ในโปรแกรม U-center

#### 2.4.7 การแปลงค่าพิกัดแบบติจูดลงติจูดไปยังค่าพิกัดแบบ UTM X และ Y

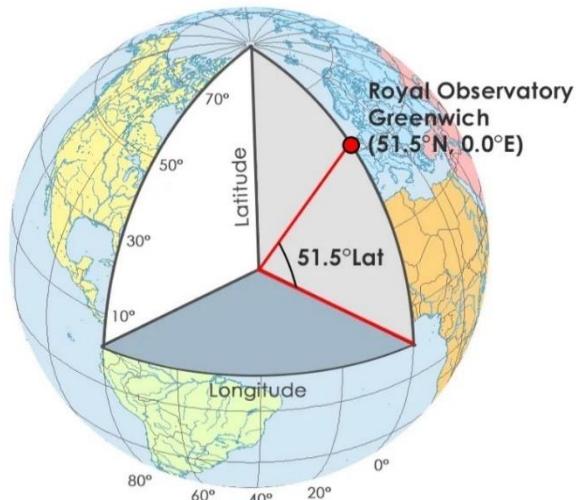
ทุกพื้นที่บนโลกนั้นมีพิกัดทางภูมิศาสตร์กำหนดไว้ พิกัดทางภูมิศาสตร์จะระบุเป็นในรูปของตัวเลข ที่ทุกคนสามารถที่จะเข้าใจถึงตำแหน่งนั้นๆ ได้จากค่าตัวเลขของพิกัดทางภูมิศาสตร์ ซึ่งการระบุพิกัดทางภูมิศาสตร์จะมีตัวเลข 2 ชุด เรียกว่า เลขละติจูด (Latitude) และเลขลองจิจูด (Longitude) (Lat/Long)

ระบบพิกัดภูมิศาสตร์ (Geographic Coordinate System : GCS) คือ ระบบอ้างอิง 3 มิติที่เก่าแก่ที่สุด ซึ่งมนุษย์นำมาใช้ในกำหนดและระบุตำแหน่งต่าง ๆ บนพื้นผิวทรงกลมของโลก โดยการอ้างอิงพิกัดที่เกิดจากค่าระยะเชิงมุมของละติจูด (Latitude) และลองจิจูด (Longitude) ซึ่งเคลื่อนออกห่างจากศูนย์กำเนิด (Origins) ที่กำหนดขึ้น



รูปที่ 2.39 เส้นสมมติในระบบพิกัดภูมิศาสตร์

สำหรับศูนย์กำเนิดของละดิจูด (Origin of Latitude) คือ เส้นสมมติในแนวราบที่ตัดผ่านศูนย์กลางของโลกพร้อมทั้งตั้งฉากไปกับแกนหมุนหรือที่เราเรียกวันว่า “เส้นศูนย์สูตร” (Equator) ขณะที่ศูนย์กำเนิดของลองจิจูด (Origin of Longitude) คือ เส้นสมมติในแนวตั้งที่ลากผ่านแกนหมุนของโลกตรงบริเวณหอดสังเกตการณ์ทางดาราศาสตร์ เมืองกรีนิช (Greenwich) ประเทศอังกฤษ ซึ่งตั้งฉากกับเส้นศูนย์สูตรหรือที่เราเรียกวันว่า “เส้นพาราเมริเดียน” (Prime Meridian) ซึ่งเป็นเส้นเมridienเริ่มแรกที่แบ่งโลกออกเป็นซีกโลกตะวันตกและซีกโลกตะวันออก



รูปที่ 2.40 ตำแหน่งพิกัดของหอดสังเกตการณ์ทางดาราศาสตร์ เมืองกรีนิช

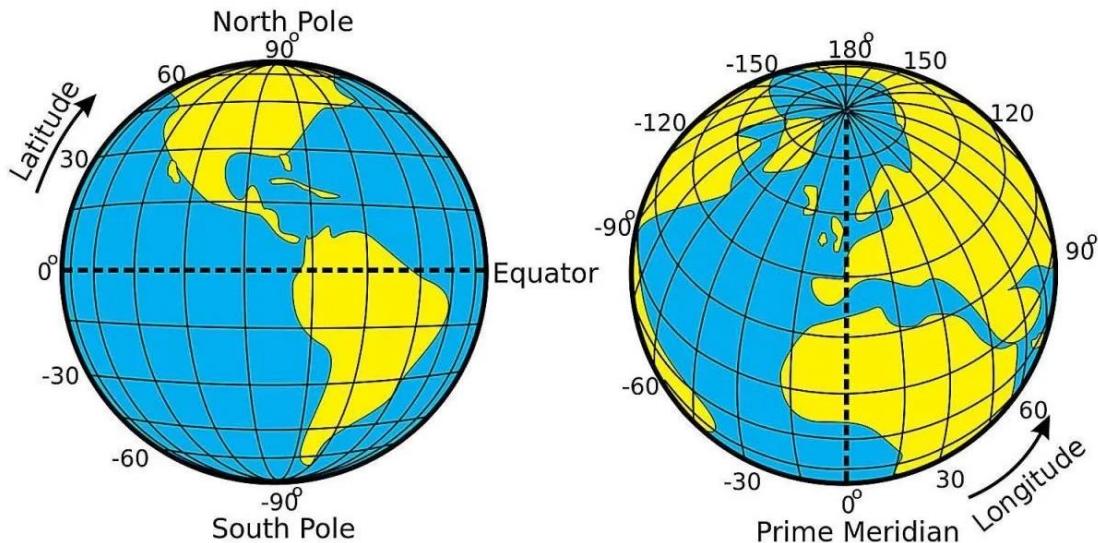
ดังนั้น ตำแหน่งที่เกิดขึ้นบนโลกตามระบบพิกัดทางภูมิศาสตร์จึงมีหน่วยเป็นองศา ลิปดา และพิลิปดา พร้อมกับกำหนดอักขระที่ระบุทิศเหนือ-ใต้ (N/S) และตะวันตก-ตะวันออก (W/E) อย่างเช่น ละติจูดที่ 13 องศา 45 ลิปดา 5 พิลิปดา เหนือ (N) หรือ ลองจิจูดที่ 100 องศา 31 ลิปดา 5 พิลิปดา ตะวันออก (E) เป็นต้น

### การระบุพิกัดบนโลก

เส้นละติจูดและลองจิจูดเป็นเส้นสมมติที่ถูกกำหนดขึ้น เพื่อนำมาใช้ในการระบุพิกัดบนพื้นผิวโลกซึ่งเป็นทรงกลม ดังนั้น เส้นสมมติที่เกิดขึ้นจึงมีลักษณะเป็นเส้นรอบวงที่เคลื่อนที่ไปรอบพื้นผิวโลก ทั้งเส้นที่垂直รอบโลกในแนวอนและเส้นที่วงศ์วงกลมในแนวตั้ง ดังนี้

เส้นละติจูด (Latitude) คือ เส้นสมมติที่วงศ์วงกลมในแนวอนตามพื้นผิวโลก แบ่งออกเป็นเส้นละติจูดที่อยู่บริเวณกึ่งกลางของโลกที่เรียกว่า “เส้นศูนย์สูตร” (Equator) และ “เส้นรุ้ง” หรือ “เส้นนานาเส้นละติจูด” (Parallels of Latitude) ที่วงศ์วงตามแนวอนบนกับเส้นศูนย์สูตรไปทางซีกโลกเหนือและซีกโลกใต้ โดยมีพิกัด (ต่าสุด) คือ 0 องศา ณ เส้นศูนย์สูตรและมีพิกัด (สูงสุด) คือ 90 องศา ณ บริเวณขั้วโลกเหนือ และใต้ นอกจากนี้ ทุก ๆ 1 องศาละติจูดหรือระยะห่างระหว่างเส้นละติจูด 1 องศาสามารถคำนวณเป็นระยะทาง ราว 111 กิโลเมตร (69 ไมล์) บนพื้นผิวโลก

เส้นลองจิจูด (Longitude) คือ เส้นสมมติที่วงศ์วงกลมในแนวตั้งตามแกนหมุนของโลกซึ่งตั้งฉากกับเส้นศูนย์สูตรหรือที่เรียกว่า “เส้นแบ่ง” หรือ “เส้นเมริเดียน” (Meridian) โดยมีเส้นสมมติหลัก คือ เส้นที่ตัดผ่านหอดสังเกตการณ์ทางดาราศาสตร์ เมืองกรีนิช (Greenwich) ประเทศอังกฤษ นับเป็นเส้นลองจิจูดที่ 0 องศา ซึ่งแบ่งโลกออกเป็นซีกโลกตะวันตกและซีกโลกตะวันออก โดยพิกัดของเส้นจากเมืองกรีนิชไปทางด้านข้าง จึงมีค่าพิกัดสูงสุดที่ 180 องศาตะวันตกหรือ 180 องศาตะวันออก นอกจากนี้ เส้นที่ 180 ซึ่งทับกันพอดียังถูกเรียกว่า “เส้นเขตวัน” (International Line) หรือเส้นแบ่งเขตวันระหว่างชาติที่เป็นเส้นของการกำหนดเขตเริ่มต้นของวันใหม่และจุดสิ้นสุดของวันเก่าอีกด้วย



รูปที่ 2.41 มุมและตารางต่างๆในการวัดค่าพิกัดแบบละติจูดและลองติจูด

### พิกัดทางภูมิศาสตร์และการกำหนดเวลามาตรฐาน

การแบ่งเขตวันและเวลาของประเทศต่าง ๆ ทั่วโลกนั้นอาศัยเส้นเมริเดียนหรือลองจิจูดในการกำหนดวันและเวลาของแต่ละพื้นที่ โดยมีเส้นพาราเมริเดียนที่ลากผ่านเมืองกรีนิช ประเทศอังกฤษ เป็นจุดอ้างอิงหลักที่ 0 องศาและจากการที่โลกหมุนรอบตัวเอง 1 รอบ (มุ่ง 360 องศา) ใช้เวลา 24 ชั่วโมง ส่งผลให้ทุก 1 ชั่วโมง โลกสามารถหมุนไปได้ 15 องศาของจิจูดหรือราว 1 องศาของจิจูดในทุก 4 นาที

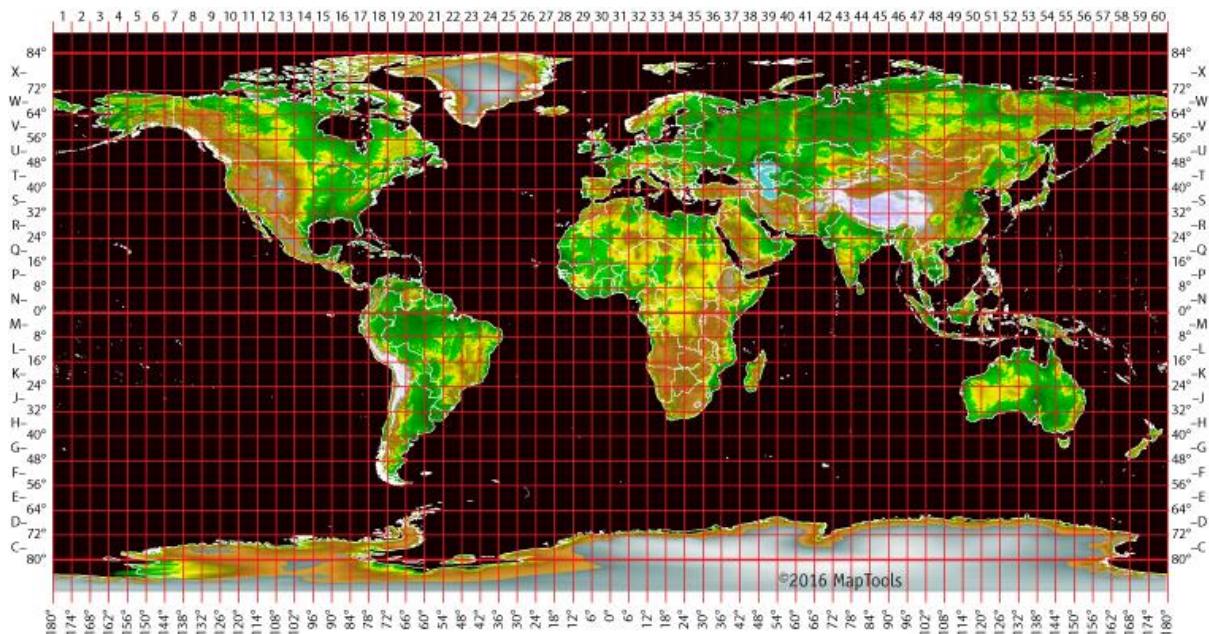
จากการกำหนดศูนย์กำหนดของลองจิจูด ณ เมืองกรีนิช เวลาท้องถิ่นของเมืองกรีนิชจึงถูกเรียกว่า “เวลาปานกลางกรีนิช” (Greenwich Mean Time : GMT) หรือเวลาสากล (Coordinated Universal Time : UTC) ตามข้อตกลงในปี ค.ศ. 1884 ส่งผลให้ประเทศไทยตั้งอยู่ทางทิศตะวันตกของเมืองกรีนิชหรือเส้นลองจิจูดที่ 0 องศา จะมีเวลาช้ากว่าเวลาของเมืองกรีนิชและในทางกลับกันประเทศไทยตั้งอยู่ทางทิศตะวันออกของเมืองกรีนิชจะมีเวลาเร็วกว่าเวลาสากล ดังนั้น หากอ้างอิงตามเส้นเมริเดียน เมืองแต่ละเมือง อย่างเช่น กรุงเทพมหานครและอุบลราชธานีจะมีการกำหนดเวลาที่แตกต่างกันออกไปเล็กน้อย ซึ่งสร้างความยุ่งยาก ทำให้เกิดการตั้ง “เวลามาตรฐาน” (Standard Time) เพื่อกำหนดให้ในบริเวณใกล้เคียงกันนั้นมีเวลาเที่ยบทั้งหมด ซึ่งการแบ่งเขตพื้นที่ตามเส้นลองจิจูดที่ลากจากขั้วโลกหนึ่งสู่ขั้วโลกได้จะทำการแบ่งเป็นเขตละ 15 องศาของจิจูดหรือเที่ยบทั้งเวลา 1 ชั่วโมง โดยอาศัยเส้นเมริเดียนกลางของเขตดังกล่าว เป็นหลัก เช่น ลองจิจูดที่ 0 องศา 15 องศา และ 30 องศา เป็นต้น โดยที่พื้นที่ซึ่งครอบคลุมอยู่ภายใน ระยะห่างไม่เกิน 7 องศา 30 ลิปดาทั้งสองข้างของเส้นเมริเดียนกลางจะนับเป็นพื้นที่ซึ่งอยู่ในเขตเวลาเดียวกัน (Time Zone) หรือมี “เวลาท้องถิ่น” (Local Time) เท่ากันนั่นเอง



รูปที่ 2.42 แผนที่เขตเวลาของโลก

ดังนั้น ประเทศที่มีขนาดใหญ่และครอบคลุมพื้นที่เป็นอาณาบริเวณกว้างจึงมีหลายเขตเวลาภายในประเทศ ของตน อย่างเช่น รัสเซียที่มีมากถึง 11 เขตเวลา หรือ สหรัฐอเมริกาที่มี 6 เขตเวลา แต่อย่างไรก็ตาม เขต เวลาบางเขตลูกกำหนดขึ้นเพื่อจ่ายต่อการสื่อสาร ดังนั้น ในบางรัฐ ประเทศ หรือเขตการปกครองอื่นๆ จะ มีการกำหนดเขตเวลาซึ่งไม่ตรงตามเส้นลองจิจูดที่ลากจากขั้วโลกเหนือลงสู่ขั้วโลกใต้อย่างสมบูรณ์

**ระบบพิกัดกริด UTM (Universal Transvers Mercator co-ordinate System)** พิกัดกริด UTM (Universal Transvers Mercator) เป็นระบบตารางกริดที่ใช้ช่วยในการกำหนดตำแหน่ง และใช้อ้างอิงในการนับกอก ตำแหน่ง ที่นิยมใช้กันแพร่ที่ในกิจการทหารของประเทศไทยฯ เกือบทั่วโลกในปัจจุบัน เพราะเป็นระบบ ตารางกริดที่มีขนาดรูปร่างเท่ากันทุกตาราง และมีวิธีการกำหนดบนกริดค่าพิกัดที่ง่ายและถูกต้องเป็นระบบกร ิดที่นำเอาเส้นโกรงแผนที่แบบ Universal Transvers Mercator Projection ของ Gauss Krugger มาใช้ ดัดแปลงการถ่ายทอด รายละเอียดของพื้นผิวโลกให้รูปทรงกระบก Mercator Projection อยู่ในตำแหน่ง Mercator Projection (แกนของรูปทรงกระบกจะทับกับแนวเส้นอิควาเตอร์และตั้งฉากกับแนวแกนของ ขั้วโลก) ประเทศไทยเราได้นำเอาเส้นโกรงแผนที่แบบ UTM นี้มาใช้กับการทำแผนที่กิจการทหาร ภายในประเทศไทยรูปถ่ายทางอากาศในปี 1953 ร่วมกับสหรัฐอเมริกา เป็นแผนที่มาตรฐาน 1:50,000 ชุด L 7018 ที่ใช้ในปัจจุบัน



รูปที่ 2.43 UTM grid zone

แผนที่ระบบพิกัดกริด ที่ใช้เส้นโกรงแผนที่แบบ UTM เป็นระบบเส้นโกรงชนิดหนึ่งที่ใช้ผิวโลกทรงกระบอกเป็นผิวแสดงเส้นเมอริเดียน (หรือเส้นลองติจูด) และเส้นละติจูดของโลกโดยใช้ทรงกระบอกตัดโลกระหว่างละติจูด 84 องศาเหนือ และ 80 องศาใต้ในลักษณะแกนรูปทรงกระบอก ทำมุกกับแกนโลก 90 องศารอบโลกแบ่งออกเป็น 60 โซนๆละ 6 องศา โซนที่ 1 อยู่ระหว่าง 180 องศา กับ 174 องศาตะวันตก และมีองค์ติจูด 177 องศาสะตะวันตก เป็น เมอริเดียนย่านกลาง (Central Meridian) มีเลขกำกับแต่ละโซนจาก 1 ถึง 60 โดยนับจากซ้าย ไปทางขวาระหว่างละติจูด 84 องศาเหนือ 80 องศาใต้แบ่งออกเป็น 2 ช่อง ๆ ละ 8 องศา ยกเว้นช่องสุดท้ายเป็น 12 องศา โดยเริ่มนับตั้งแต่ละติจูด 80 องศาใต้ขึ้นไปทางเหนือ ให้ช่องแรกเป็นอักษร C และช่องสุดท้ายเป็นอักษร X (ยกเว้น I และ O) จากการ แบ่งตามที่กล่าวแล้ว จะเห็นพื้นที่ในเขตลองติจูด 180 องศาสะตะวันตก ถึง 180 องศาสะตะวันออก และละติจูด 80 องศาใต้ถึง 84 องศาเหนือ จะถูกแบ่งออกเป็นรูปสี่เหลี่ยมผืนผ้า 1,200 รูป แต่ละรูปมีขนาดกว้างยาว 6 องศา x 8 องศา จำนวน 1,140 รูป และกว้างยาว 6 องศา x 12 องศา จำนวน 60 รูป รูปสี่เหลี่ยมนี้เรียกว่า Grid Zone Designation (GZD) การเรียกชื่อ Grid Zone Designation ประเทศไทยมีพื้นที่อยู่ระหว่างละติจูด 5 องศา 30 ถึง 20 องศา 30 ถึง 15 องศา และลองติจูดประมาณ 97 องศา 30 ถึง 105 องศา 30 ถึง 115 องศา ตะวันออก ดังนั้น ประเทศไทยจึงตกอยู่ใน GZD 47N 47P 47Q 48N 48P และ 48Q การอ่านค่าพิกัดกริดเพื่อให้พิกัดค่ากริดในโซนหนึ่งๆ มีค่าเป็นบวกเสมอ จึงกำหนดให้มีศูนย์สมมุติขึ้น 2 แห่งดังนี้

- ในบริเวณที่อยู่หนีอเส้นศูนย์สูตร: เส้นศูนย์สูตรมีระยะห่างจากศูนย์สมมุติเท่ากับ 0 เมตร และเส้นเมอริเดียนย่านกลางห่างจากศูนย์สมมุติ 500,000 เมตร ทางตะวันออก
- ในบริเวณที่อยู่ใต้เส้นศูนย์สูตร: เส้นศูนย์สูตรมีระยะห่างจากศูนย์สมมุติไปทางเหนือ 10,000,000 เมตร และเมอริเดียนย่านกลางห่างจากศูนย์สมมุติ 500,000 เมตรทางตะวันออก พิกัดภูมิศาสตร์ (Geographic Coordinate) โดยที่เราต้องอ่านค่าของละติจูดและลองจิจูดตัดกัน

## 2.5 ภาพรวมและทฤษฎีต่างๆ ของระบบควบคุมความเร็วให้คงที่

การทำงานของระบบควบคุมความเร็วให้คงที่แบ่งออกเป็น 2 โหมดการทำงาน คือระบบการทำงานแบบ Manual และ การทำงานแบบ Auto โดยผู้ใช้งานสามารถเลือกการทำงานได้ผ่านสวิตช์ควบคุมการทำงาน

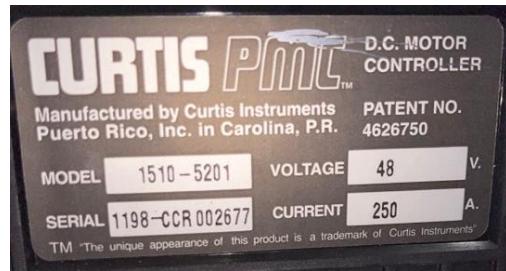
- 1. การทำงานแบบ Manual** เป็นการควบคุมความเร็วตามระบบของรถผู้ใช้งานสามารถเพิ่ม และลดความเร็วได้จากการเหยียบคันเร่งของรถ โดยตรง
- 2. การทำงานแบบ Auto** เป็นการควบคุมความเร็วให้เคลื่อนที่ด้วยความเร็วคงที่ ซึ่งจะเป็นการควบคุมความเร็วผ่านวงจร Speed Controller Box เป็นอุปกรณ์ที่ต่อเพิ่มเข้ากับ Controller ของรถ นอกเหนือไปสามารถปรับเปลี่ยนความเร็วได้โดยการเพิ่มค่าความเร็วที่เพิ่ม วงจรการควบคุมความเร็ว หรือการปรับเปลี่ยนค่าการทำงานที่ตัวประมวลผลการทำงานในส่วนของระบบควบคุมความเร็วให้คงที่ (Arduino Mega 2560)

### 2.5.1 ลักษณะการทำงานของรถกล้อไฟฟ้า

ลักษณะการทำงานของรถกล้อไฟฟ้าที่มีรูปแบบการทำงานแบบไฟฟ้า คือ เมื่อมีการเลือกสวิตช์การทำงานของรถ เป็น Run และ การ ไฟกุญแจ แรงดันจากแบตเตอรี่ 53 V จะเริ่มไฟลับเข้าตัวประมวลผลของรถเพื่อนำไปควบคุมการทำงานในส่วนอื่นๆ เช่น ตัวควบคุมทิศทางของรถ ได้แก่ การเคลื่อนที่ไปด้านหน้า (F) , การเคลื่อนที่ด้วยหลัง (B) หรือ การหยุดเคลื่อนที่ (N) แต่หากเลือกสวิตช์ Tow รถจะไม่สามารถทำงานได้แม้มีการไฟกุญแจ

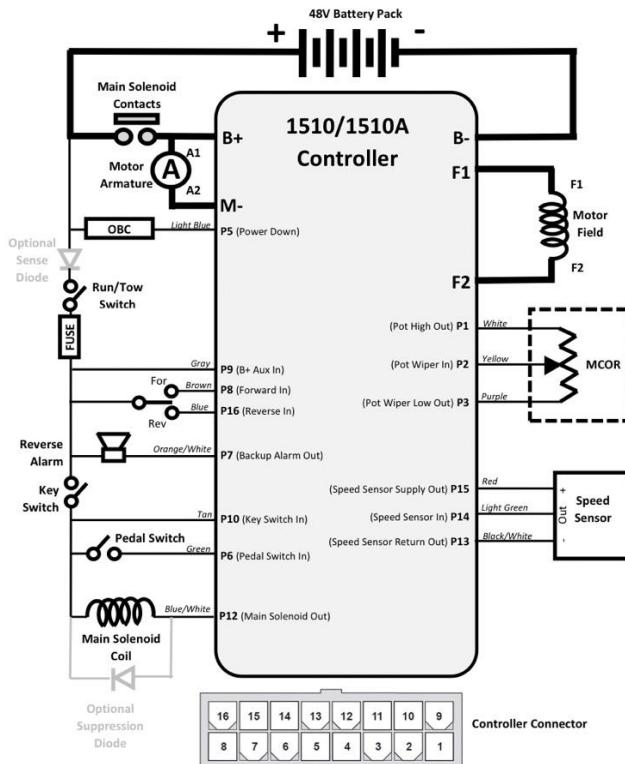
#### 2.5.1.1 ตัวประมวลผลการทำงานของรถกล้อไฟฟ้า (Controller)

ในการควบคุมการทำงานของระบบควบคุมความเร็วจะเป็นการนำงงานต่อเพิ่มเข้ากับตัวประมวลผลการทำงานของรถ (Controller) คือ Curtis PMC DC Motor Controller Model 1510-5201 Serial 1198-CCR 002677 48V 250A



รูปที่ 2.44 ตัวควบคุมและประมวลผลการทำงานของรถกอล์ฟ

#### 2.5.1.2 ลักษณะการทำงานของตัวประมวลผลการทำงานของรถกอล์ฟ



รูปที่ 2.45 Wiring Diagram ของตัวควบคุมและประมวลผลการทำงานของรถกอล์ฟ

- P1, P2, P3 (สีขาว, สีเหลือง, สีม่วง) เป็น Pin ที่ควบคุมการทำงานของความเร็ว จะใช้ในการทำงานกับ P6 โดย P1 ทำหน้าที่เป็นตัวจ่ายแรงดันให้กับ MCOR , P2 เป็นสายสัญญาณ และ P3 ทำหน้าที่เป็น GND ของ MCOR

- P6 (สีเขียวเข้ม) เป็น Pin ที่ควบคุมการทำงานของการเหยียบคันเร่ง เมื่อมีการเหยียบคันเร่ง P6 จะมีแรงดันไฟลั่งเข้าไปควบคุมตัวด้านหน้าที่อยู่ภายในทำให้ความเร็วที่ออกมากಡาต่างกันตามน้ำหนักที่ผู้ใช้งานเหยียบ
- P7 (สีส้มขาว) เป็น Pin ที่ควบคุมการทำงานของเสียงเลือก Switch Direction เป็นออยหลัง
- P8 (สีน้ำตาล) เป็น Pin ที่ควบคุมการทำงานของ Switch Direction การเคลื่อนที่ไปด้านหน้า
- P9 (สีเทา) เป็น Pin ที่จ่ายแรงดันจากแบตเตอรี่ 53 V ไปยังการทำงานอื่นๆภายใน ตัวประมวลผล
- P13, P14, P15 (สีแดง, สีเขียวอ่อน, สีดำขาว) เป็น Pin เชื่อมเข้า Hall Sensor เพื่อส่งสัญญาณไปควบคุมการทำงานของ Motor โดย P13 จะมีหน้าที่เป็น GND ของ Speed Sensor P14 มีหน้าที่เป็นสายสัญญาณ และ P15 มีหน้าที่จ่ายแรงดันเข้า Speed Sensor



รูปที่ 2.46 สวิตช์ Run/Tow



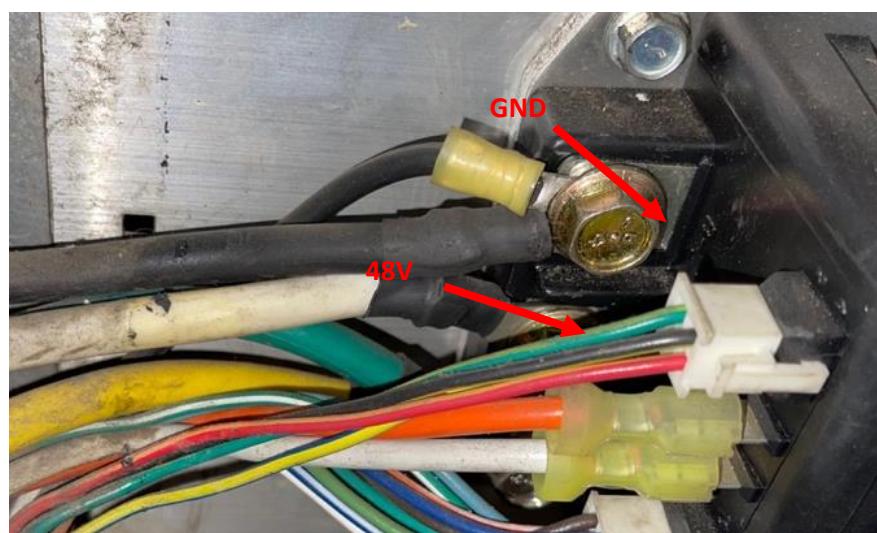
รูปที่ 2.47 สวิตช์เลือกการทำงานของทิศทาง (Switch Direction)



รูปที่ 2.48 Pin of controller



รูปที่ 2.49 Pin ที่เชื่อมกับ Hall Sensor



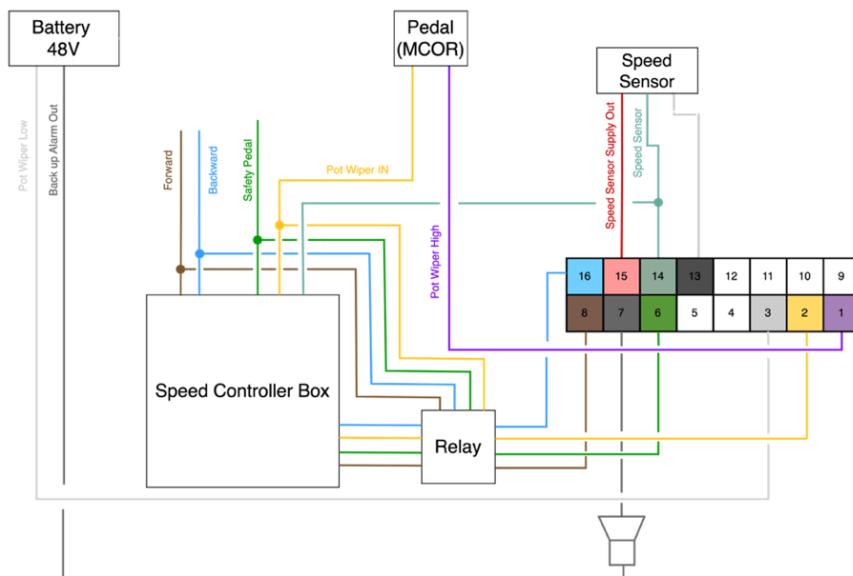
รูปที่ 2.50 แรงดันที่มาจากแบตเตอรี่

## 2.5.2 การควบคุมความเร็วในการเคลื่อนที่

โดยทั่วไปการควบคุมความเร็วของรถจะควบคุมผ่านการเหยียบคันเร่ง หรือ Throttle System (Drive -by-wire) ลักษณะการทำงาน คือ เป็นอุปกรณ์ควบคุมความเร็วจากภายในรถโดยการใช้ไฟฟ้า ด้านในอุปกรณ์ประกอบด้วยตัวถ้านทานแบบชุดลวด เมื่อเริ่มมีการเหยียบคันเร่งค่าความถ้านทานก็จะน้อยลง ทำให้มีกระแสไฟฟ้าไหลได้มากขึ้น ส่งผลให้ความเร็วเพิ่มขึ้น

### 2.5.2.1 กลไกการออกแบบทำงานของระบบควบคุมความเร็ว

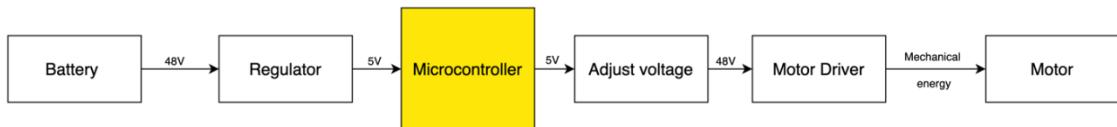
การควบคุมความเร็วในการเคลื่อนที่ คือ การควบคุมแรงดันที่จ่ายไปยัง Pin ที่ควบคุมการทำงานในส่วนของความเร็ว ดังนี้ Pin ที่นำมาใช้ในการต่อเข้ากับวงจรควบคุมความเร็ว (Speed Controller Box) ประกอบด้วย Pin ที่ควบคุมการทำงานของทิศทางการเคลื่อนที่ (Switch Direction) (P6, P8, P16), Speed Sensor (P13, P14, P15), MCOR (P1, P2, P3) รวมไปถึง Pin ที่ควบคุมการจ่ายแรงดัน 53 V ที่มาจากการแบตเตอรี่ (P9) รถกอล์ฟ



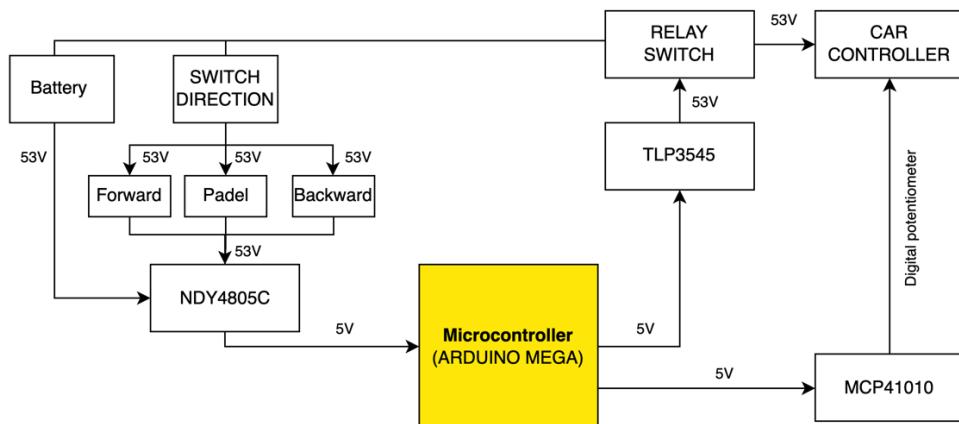
รูปที่ 2.51 Block Diagram ภาพรวมการทำงานของระบบควบคุมความเร็ว

ดังนั้นภายใน Speed Controller Box ที่ใช้ในการควบคุมความเร็วจะประกอบด้วย Microcontroller เพื่อใช้ในการควบคุมการทำงาน แต่ Microcontroller ไม่สามารถต่อเข้ากับ Controller ของรถโดยตรงได้เนื่องจากแรงดัน และกระแสจากรถมีมาเกินที่ Microcontroller จะรับได้จึงต้องมีการออกแบบวงจรเพื่อลดขนาด

แรงดัน และกระแสก่อนเข้า Microcontroller และเมื่อมีการประมวลผลเสร็จต้องมีวงจรขยายแรงดัน และกระแสให้เท่าเดิมก่อนเข้า Controller เพื่อให้แรงดัน และกระแสเพียงพอที่จะขับให้มอเตอร์เริ่มทำงาน



รูปที่ 2.52 กลไกการออกแบบวงจรการทำงาน

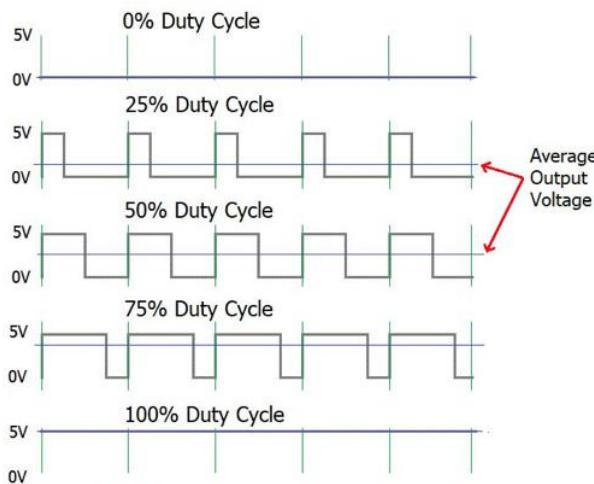


รูปที่ 2.53 กลไกการออกแบบวงจรการทำงาน

### 2.5.2.2 หลักการควบคุมความเร็วของมอเตอร์

การควบคุมความเร็วของรถผ่านมอเตอร์ โดยสามารถทำได้ 3 วิธี

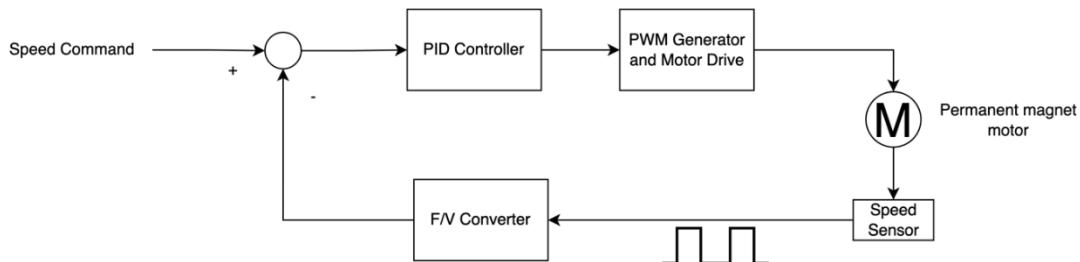
1. การควบคุมความเร็วโดยการต่อตัวต้านทานปรับค่าได้ เพื่อปรับสัญญาณที่เข้า Encoder จากนั้นนำสัญญาณที่ได้มาควบคุมความเร็วที่มอเตอร์
2. เปลี่ยนค่าระดับแรงดันที่ป้อนเข้ามอเตอร์ โดยวิธีนี้สามารถควบคุมให้มอเตอร์ทำงานจะคงที่ ส่งผลให้ความเร็วคงที่ แต่มีความเร็วต่ำ จะส่งผลให้แรงบิดต่ำ
3. การควบคุมความเร็วโดยการควบคุมการจ่ายกระแสเป็นช่วงๆ (PWM: Pulse Width Modulation) เป็นสัญญาณที่มีค่าความถี่คงที่ แต่ความกว้างของพลังเปลี่ยนแปลงได้ โดยถ้าความกว้างของลูกคลื่นมากแรงดันเฉลี่ยที่ได้จะมากตาม



รูปที่ 2.54 การควบคุมกระแสแรงดันแบบ Pulse Width Modulation

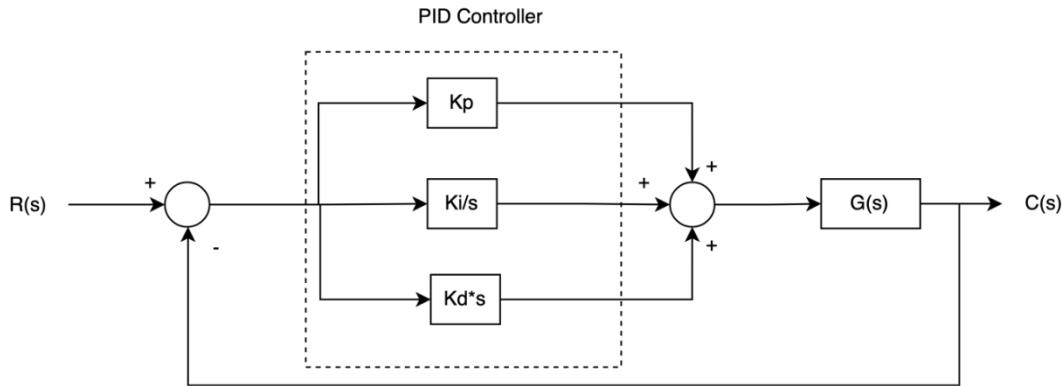
### 2.5.2.3 การควบคุมความเร็วมอเตอร์โดยใช้ PID Controller

การควบคุมความเร็วของมอเตอร์ไฟฟ้า สามารถเขียนเป็น Diagram ตามรูปที่ XX และการออกแบบตัวควบคุมการทำงานแบบ PID สำหรับการควบคุมความเร็วแบบระบบปิด (Close Loop)



รูปที่ 2.55 Block Diagram การควบคุมความเร็วมอเตอร์ไฟฟ้าแบบระบบปิด

ตัวควบคุมแบบ PID เป็นตัวควบคุมที่ทำงานโดยการนำค่าความผิดพลาดระหว่างสัญญาณอ้างอิงกับค่าปัจจุบันที่ได้จากการบวนการ (Process) มาสร้างเป็นเอต์พุตใหม่ ด้วยการขยายค่าความผิดพลาดของสัญญาณดังกล่าวด้วยค่า Gain เอต์พุตของตัวควบคุมแบบ PID ประกอบด้วย ตัวควบคุมแบบสัดส่วน ( $K_p$  : Proportional Control) ตัวควบคุมแบบปริพันธ์ ( $K_i$  : Integral Control) และตัวควบคุมแบบอนุพันธ์ ( $K_d$  : Derivative Control) ซึ่งทั้ง 3 ส่วนของตัวควบคุมจะนำค่าเอต์พุตไปคูณกับค่า Gain ของตัวควบคุมแต่ละชนิด โดยค่า Gain ดังกล่าวจะเป็นตัวกำหนดผลตอบสนองของระบบ



รูปที่ 2.56 Block Diagram ของตัวควบคุมแบบ PID

$$G_c = K_p + \frac{K_i}{s} + K_d s \quad (2.4)$$

$$U(k) = K_p e(k) + K_i \sum e(i) + K_d \Delta e(k) \quad (2.5)$$

$$(Error * K_p) + (K_i * \text{sigmaerror}) + (K_d * ((error - preerror)/dT)) \quad (2.6)$$

### คุณสมบัติของตัวควบคุม

ระบบควบคุมโดยทั่วไป สามารถแบ่งคุณสมบัติของระบบได้เป็น 4 แบบ ดังนี้

- ระบบควบคุมแบบ P (P Controller) มีคุณสมบัติในการลดค่า Rise Time ทำให้ระบบทำงานเร็วขึ้นในช่วงแรก เพิ่ม Over Shoot ทำให้ระบบแก่วงในช่วงเริ่มต้น แต่ผลของ steady State Error จะลดลง
- ระบบควบคุมแบบ PD (PD Controller) มีคุณสมบัติในการลดค่า Over Shoot ทำให้ระบบแก่วงน้อยลง ในช่วงเริ่มต้น ลด Setting Time ทำให้ระบบทำงานถึงจุดที่คงที่ Steady state เร็วขึ้น
- ระบบควบคุมแบบ PI (PI Controller) มีคุณสมบัติในการลดค่า Rise Time ทำให้ระบบทำงานเร็วขึ้นในช่วงแรก ลดค่า Steady State Error เพื่อทำให้ค่าเอ้าต์พุตเทิบเท่าค่าอินพุต
- ระบบควบคุมแบบ PID (PID Controller) เป็นการนำระบบควบคุมทั้ง 3 แบบ คือ P, PD และ PI มารวมกัน โดยสามารถกำหนดค่าอัตราการขยายทั้ง 3 แบบ คือ P, I และ D ได้อย่างอิสระ ทำให้สามารถออกแบบระบบควบคุมให้เป็นไปตามที่ต้องการได้ โดยทำการปรับค่าอัตราการขยายทั้ง 3 แบบ และนำค่ามาพิจารณาผลการตอบสนองที่ได้ เมื่อปรับค่าอัตราการขยายจนได้ผลตอบสนองตามที่ต้องการแล้วจึงนำค่าที่ได้ไปออกแบบเป็นวงจรในงานต่อไป

#### 2.5.2.4 การคำนวณหาค่าความเร็วในการเคลื่อนที่

##### การคำนวณหาค่าความเร็วของมอเตอร์

การคำนวณหาค่าความเร็วของมอเตอร์ จะอยู่ในรูปแบบ RPM (Revolutions per minute) คือจำนวนรอบ ในหนึ่งนาทีเป็นหน่วยของความเร็วในการหมุนหรือความถี่ของการหมุนรอบแกนคงที่

$$\text{RPM} = \frac{\text{จำนวนรอบ}}{60 \text{ วินาที}} \quad (2.7)$$

หากต้องการความเร็วในหน่วย Km/Hr

$$\text{Km/Hr} = 60 \times \text{RPM} \times \frac{\text{ระยะที่รถสามารถเคลื่อนที่ได้ 1 รอบ การทำงานของมอเตอร์}}{1000} \quad (2.8)$$

#### 2.5.2.5 การคำนวณหาค่าความเร็วของการเคลื่อนที่โดยใช้ระยะติดและลองจิจูด

ข้อมูลที่ได้มาจากระบบพิกัด GNSS ประกอบด้วย ละติจูด และลองจิจูด ซึ่งหาต้องการนำข้อมูลดังกล่าวไปหาเป็นค่าความเร็วในการเคลื่อนที่ของรถจะต้องเริ่มจากการหาค่าระยะทางโดยใช้ทฤษฎี Haversine Formula

##### ทฤษฎี Haversine Formula

Haversine เป็นสมการที่ใช้หาระยะห่างระหว่างจุดสองจุดบนพื้นผิวทรงกลม โดยพิจารณาจากละติจูด และลองจิจูด

$$\text{Distance (D)} = \text{average radius of the earth (R)} \times c \quad (2.9)$$



รูปที่ 2.57 การหาระยะห่างระหว่างจุดสองจุดบนพื้นผิวทรงกลม

$$\text{Distance (D)} = \text{ระยะทาง หรือ ระยะห่างระหว่างจุดสองจุด} \quad (\text{m})$$

$$R = \text{earth's radius (mean radius} = 6,371\text{km}) \quad (\text{Km})$$

$$\Delta \text{lat} = \text{lat}_2 - \text{lat}_1$$

$$\Delta \text{long} = \text{long}_2 - \text{long}_1$$

$$a = \sin^2(\Delta \text{lat}/2) + \cos(\text{lat}_1).\cos(\text{lat}_2).\sin^2(\Delta \text{long}/2)$$

$$c = 2.\text{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$D = R.c$$

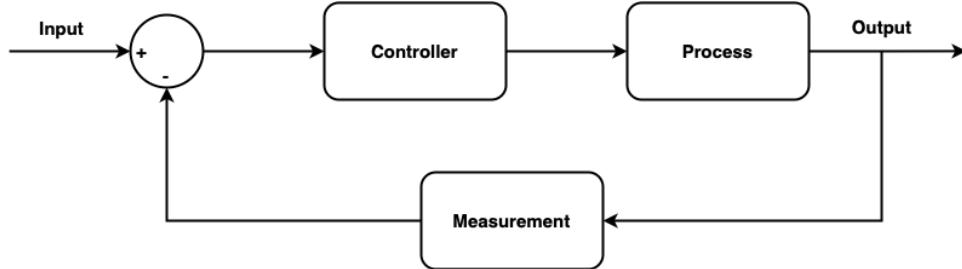
จากนั้นนำระยะทาง หรือ ระยะห่างระหว่างจุดสองจุดที่ได้มาคำนวณหาความเร็ว

$$\text{Velocity (V)} = \text{Distance (S)}/\text{Time (t)} \quad (\text{m/s})$$

$$\text{Velocity (V)} = \text{Velocity (V)} \times 3.6 \quad (\text{Km/hr})$$

## 2.6. ภาพรวมและทฤษฎีต่างๆ ของระบบควบคุมตำแหน่งในการเคลื่อนที่ของรถ

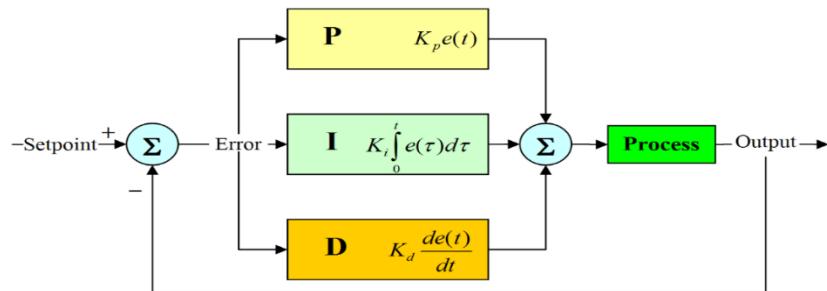
การควบคุมรถในการเคลื่อนที่นั้นจะใช้ระบบควบคุมแบบปิด (Closed-Loop Control System) เป็นระบบที่นำสัญญาณจากเอาต์พุตของระบบ ป้อนกลับมาเปรียบเทียบกับสัญญาณอินพุตที่ป้อนให้กับระบบ ซึ่งผลต่างระหว่างสัญญาณทั้งสองที่นำมาเปรียบเทียบนั้นจะเป็นค่าผิดพลาด (Error) เพื่อที่จะใช้เป็นสัญญาณป้อนเข้าตัวควบคุม (Controller) ให้ตัวควบคุมนำไปสร้างสัญญาณควบคุมเพื่อลดความผิดพลาดที่เกิดขึ้นในระบบและทำให้อาต์พุตของระบบเข้าสู่ค่าที่ต้องการ (Setpoint)



รูปที่ 2.58 ไดอะแกรมของระบบควบคุมแบบปิด (Closed-Loop Control System)

## 2.6.1 การนำตัวควบคุมระบบแบบปิดมาใช้กับการควบคุมรถถังอัตโนมัติ

### 2.6.1.1 PID Controller



รูปที่ 2.59 PID Controller Block Diagram

ตัวควบคุมแบบพีไอดี (Proportional and Derivative Controller: PID Controller) เป็นระบบควบคุมแบบป้อนกลับ ซึ่งในการคำนวณตัวควบคุมแบบพีไอดีนี้จะเป็นการนำค่าความผิดพลาด (Error) ที่เกิดขึ้นในระบบมาคำนวณ โดยตัวควบคุมจะทำหน้าที่ในการลดค่าความผิดพลาดที่เกิดขึ้นจนกระทั่งระบบมีความเสถียร โดยวิธีการคำนวณของตัวควบคุมแบบพีไอดีนี้จะขึ้นอยู่กับ 3 ตัวแปรดังนี้ ค่าสัดส่วน (Proportional: P) ปริพันธ์ (Integral: I) และอนุพันธ์ (Derivative: D) กล่าวคือ ค่าสัดส่วน (Proportional: P) หมายถึง ค่าที่กำหนดจากผลของการคำนวณในปัจจุบัน ปริพันธ์ (Integral: I) หมายถึง ค่าที่กำหนดจากผลบันทึกฐานของผลกระทบความผิดพลาดที่ผ่านไป อนุพันธ์ (Derivative: D) หมายถึง ค่าของอัตราการเปลี่ยนแปลงของค่าความผิดพลาด

$$MV(t) = P_{out} + I_{out} + D_{out} \quad (2.10)$$

โดยการควบคุมแบบพื้นๆ โอดีจะนำตัวแปรทั้ง 3 จากที่กล่าวมาข้างต้นนี้มารวมกัน ซึ่งจะเป็นไปตามสมการด้านล่างนี้

$$P_{out} = K_p e(t) \quad (2.11)$$

$$I_{out} = K_i \int_0^t e(t) d\tau \quad (2.12)$$

$$D_{out} = K_d \frac{d}{dt} e(t) \quad (2.13)$$

เมื่อนำแต่ละเทอมมารวมกันจะได้สมการตัวควบคุม PID ดังนี้

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(t) d\tau + K_d \frac{d}{dt} e(t) \quad (2.14)$$

การควบคุมประเภทนี้ใช้ในการขับเคลื่อนระบบไปในทิศทางของตำแหน่ง หรือใช้สำหรับการควบคุมอุณหภูมิและใช้ในกระบวนการทางวิทยาศาสตร์ ระบบอัตโนมัติและสารเคมีมากมาย ต่อไปจะกล่าวถึงการออกแบบคอนโทรลเลอร์ PID พร้อมโปรแกรมควบคุมที่ใช้ เช่น P, I และ D โดยจะอธิบายอย่างละเอียดดังต่อไปนี้

### **Proportional Response**

องค์ประกอบตามสัดส่วนที่ขึ้นอยู่กับความแตกต่างระหว่างจุดตั้งค่าและตัวแปรปะรุงการเท่านั้น ความแตกต่างนี้เรียกว่าเงื่อนไขข้อผิดพลาด อัตราขยายตามสัดส่วน ( $K_p$ ) กำหนดอัตราส่วนของการตอบสนอง เอาต์พุตต่อสัญญาณผิดพลาด ตัวอย่างเช่น หากจะให้ข้อผิดพลาดมีขนาด 10 การเพิ่มตามสัดส่วนเป็น 5 จะทำให้เกิดการตอบสนองตามสัดส่วนที่ 50 โดยทั่วไปการเพิ่มอัตราขยายตามสัดส่วนจะเพิ่มความเร็วของการตอบสนองของระบบควบคุม อย่างไรก็ตาม หากอัตราขยายตามสัดส่วนมากเกินไป ตัวแปรกระบวนการจะเริ่มแก่วง ถ้า  $K_p$  เพิ่มขึ้นอีก การแก่วงก็จะใหญ่ขึ้น และระบบจะไม่เสถียรและอาจแก่วงจนควบคุมไม่ได้

### **Integral Response**

ส่วนประกอบสำคัญจะรวมระยะเวลาข้อผิดพลาดในช่วงเวลาหนึ่ง ผลที่ได้คือ เม็ดค่าที่ผิดพลาดเพียงเล็กน้อยก็จะทำให้ส่วนประกอบสำคัญเพิ่มขึ้นอย่างช้าๆ การตอบสนองเชิงปริพันธ์จะเพิ่มขึ้นอย่างต่อเนื่อง

เมื่อเวลาผ่านไป เว้นแต่ข้อผิดพลาดจะเป็นศูนย์ ดังนั้นผลกระบวนการคือการขับข้อผิดพลาด Steady-State ให้เป็นศูนย์ ข้อผิดพลาด Steady-State คือความแตกต่างสุดท้ายระหว่างตัวแปรกระบวนการและจุดตั้งค่า ปรากฏการณ์ที่เรียกว่าอินทิกรัล Windup เกิดขึ้นเมื่อ แอคชันอินทิกรัลอิ่มตัวของโทรล็อร์ โดยที่ตอนโทรล็อร์ไม่ได้ขับสัญญาณผิดพลาดไปที่ศูนย์

### Derivative Response

องค์ประกอบของอนุพันธ์ทำให้เราต้องลดลง หากตัวแปรกระบวนการเพิ่มขึ้นอย่างรวดเร็ว การตอบสนองอนุพันธ์เป็นสัดส่วนกับอัตราการเปลี่ยนแปลงของตัวแปรกระบวนการ การเพิ่มพารามิเตอร์เวลา อนุพันธ์ ( $T_d$ ) จะทำให้ระบบควบคุมตอบสนองต่อการเปลี่ยนแปลงในระยะข้อผิดพลาดอย่างรุนแรงยิ่งขึ้น และจะเพิ่มความเร็วของการตอบสนองของระบบควบคุม โดยรวม ระบบควบคุมที่ใช้งานได้จริงส่วนใหญ่ใช้เวลาอนุพันธ์เพียงเล็กน้อย ( $T_d$ ) เนื่องจาก Derivative Response มีความไวสูงต่อสัญญาณรบกวนในสัญญาณแปรงฟันของกระบวนการ หากสัญญาณป้อนกลับของเซ็นเซอร์มีเสียงดังหรือถ้าอัตราลูปควบคุมช้าเกินไป อนุพันธ์ตอบสนองอาจทำให้ระบบควบคุมไม่เสถียร

#### 2.6.1.1 การใช้ PID Controller กับ Car Model

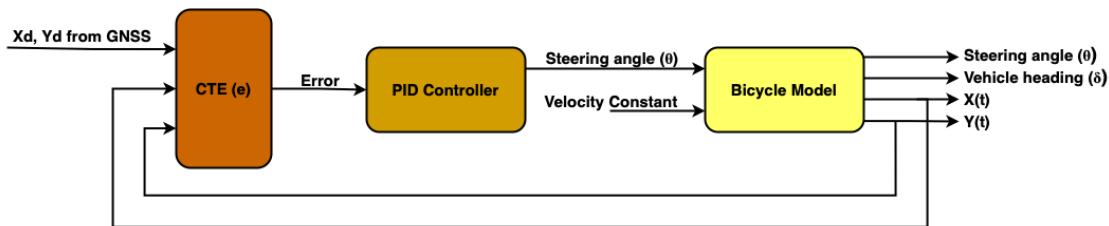
ตัวควบคุม (Controller) ใช้ตัวควบคุมแบบสัดส่วน (P-Controller) สัญญาณควบคุมจะเป็นสัดส่วนโดยตรงกับค่าสัญญาณความผิดพลาดที่เกิดจากผลต่างระหว่างค่าสัญญาณอ้างอิงกับสัญญาณเอาต์พุตของระบบที่ต้องการควบคุม จะมีคุณสมบัติของระบบควบคุมคือ ลดค่า Rise Time ( $T_r$ ) ทำให้ระบบทำงานเร็วขึ้นในช่วงแรก และเพิ่ม Over Shoot ( $M_p$ ) ทำให้ระบบแกว่งในช่วงเริ่มต้น แต่ส่งผลให้ค่า Steady State Error ลดลง สมการของตัวควบคุมแบบสัดส่วน ตัวควบคุมแบบอินทิกรัลและตัวควบคุมแบบอนุพันธ์ที่มีอินพุตเป็นสัญญาณความผิดพลาด (Error) แสดงได้ดังสมการที่ 2.15 และการแปลงลาปลาส สามารถแสดงได้ดังสมการที่ 2.16

$$u[n] = (k_p * e_i) + (k_i + \sum_{i=1}^n e_i) + (k_d * (e_i - e_{i-1})) \quad (2.15)$$

$$U(s) = \left( K_p(s) + \frac{K_I}{s} + K_D s \right) E(s) \quad (2.16)$$

เมื่อนำตัวควบคุมแบบพีไอดีมาประยุกต์ใช้กับการควบคุมมุมเลี้ยวของพวงมาลัยรถกอล์ฟ โดยค่าอินพุตที่รับเข้ามานั้นจะเป็นค่าพิกัด X-Y ที่รับมาจากเซ็นเซอร์และก็จะทำการคำนวณเพื่อหาค่าความผิดพลาด (Cross Track Error: CTE) เพื่อนำค่าความผิดพลาดไปคำนวณด้วยตัวควบคุมแบบพีไอดี หลังจากผ่าน

กระบวนการของตัวควบคุมแด๊วระบบจะส่งค่ามุ่งเลี้ยวไปยังส่วนของโนมเดครอต และเมื่อมีอาพุตออกจากโนมเดครอต ระบบจะทำการส่งค่าพิกัดที่เป็นค่าເອາດີພຸດບ້ອນກລັນມາเพื่อคำนวนให้ค่าความผิดพลาดอยู่ในช่วงที่ต้องการหรือเป็นค่าที่สามารถยอมรับได้

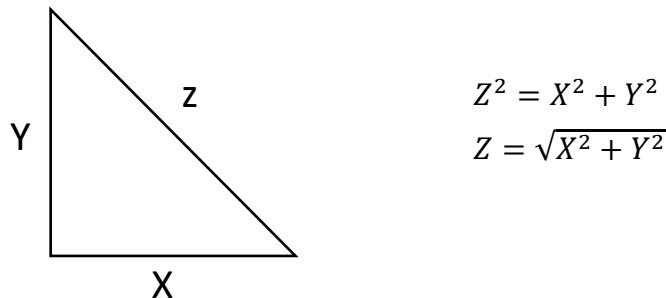


รูปที่ 2.60 บล็อกໄດ້ອະແກມการนำตัวควบคุมแบบพື້ອົດມາປະຢູກຕີໃຊ້ໃນการควบคุมມູນເລີ້ວ

จากบล็อกໄດ້ອະແກມในรูปที่ 2.60 ตัวควบคุมจะรับค่าพารามิเตอร์ທີ່ຮູ້ອືນພຸດເຂົ້າມາຄື່ອ  $X_d, Y_d$  ຜຶ້ງເປັນຄ່າພິກັດທີ່ວັດໄດ້ຈາກ GNSS Module ແລະ หลັງຈາກນັ້ນຕัวควบคุมຈະทำการคำນวนດ້ວຍອັລກອຣິຫຼິນກາຍໃນຕัวควบคุมເພື່ອສ່ວນຄ່າອາດີພຸດຂອງພວງມາລັບ (Steering angle) ທີ່ເໝາະສມກັບເສັ້ນທາງທີ່ຢານພາහນະເກລື່ອນທີ່ອອກມາ ຜຶ້ງຄ່າອາດີພຸດຂອງພວງມາລັບ (Steering angle) ຈະເປັນຄ່າອືນພຸດທີ່ສ່ວນໄປຢັງຕ້ວໂນມເດລຂອງຢານພາහນະ (Kinematic Bicycle Model) ແລະ ຈະພວກວ່າພາຣາມີເຕອຣ໌ທີ່ເປັນອືນພຸດຂອງບັດລັກ Bicycle Model ຈະມີພາຣາມີເຕອຣ໌ທີ່ເປັນຄວາມເຮົວຄົງທີ່ຮ່ວມດ້ວຍນັ້ນ ກໍເພື່ອຈະນຳໄປຢັງຕ້ວໂນມເດລຂອງ Kinematic Bicycle Model ຈະທ້າຍທີ່ສຸດແດ້ວຈະໄດ້ເອາດີພຸດອອກມາເປັນພາຣາມີເຕອຣ໌ທັງໝົດ 4 ຕ້ວ ຄື່ອ Steering angle ( $\theta$ ), Vehicle heading ( $\delta$ ),  $X(t)$  ແລະ  $Y(t)$  ເພື່ອນາໄປສ່ວນກາມອເຕອຣ໌ໃນການບັນກັນມູນເລີ້ວຂອງພວງມາລັບ (Motor Steering) ໃຫ້ເລື້ອງໄປຢັງເສັ້ນທາງຕາມທີ່ມີການກຳຫັນດ້ວຍພິກັດໄວ້ແລ້ວ

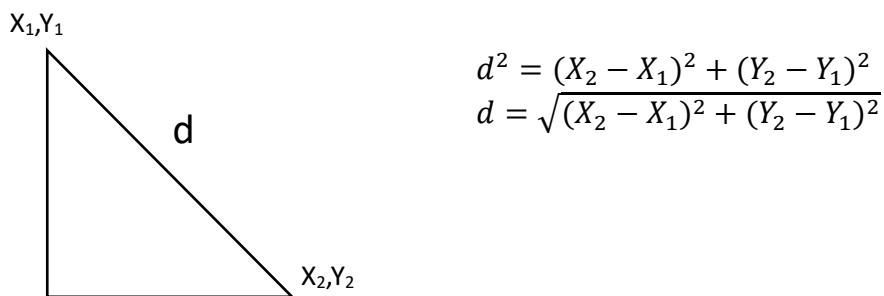
#### 2.6.1.1.2 การคำนวนระยะทางຈາກພິກັດລະຕິຈູດແລະລອງຈິຈູດ

ເມື່ອຮັບຄ່າພິກັດລະຕິຈູດແລະລອງຈິຈູດ Controller (Arduino Mega 2560) ຈະคำນົວຮະຍະທາງຈາກພິກັດລະຕິຈູດ ແລະລອງຈິຈູດ ເພື່ອນຳມາເປັນຄ່າ Setpoint ໃນການควบคຸມຮະຍະທາງກາຮັບອັນທີ່ຂອງລ້ອຫລັງ ຜຶ້ງນຳມາປະຢູກຕີໃຫ້ກັບສ່ວນກາຮັບຂອງສູງຕົກອັກອັກ ມີຮູບແບບດັ່ງຮູບທີ່ x – xx ເມື່ອກຳນົດໃຫ້ X ຄື່ອ ພິກັດລະຕິຈູດ ແລະ Y ຄື່ອ ພິກັດລອງຈິຈູດ



รูปที่ 2.61 อธิบายสูตรพีtagอรัส

จากรูปที่ 2.61 เมื่อนำสูตรพีtagอรัสมาประยุกต์ใช้กับการหาระยะทางจากพิกัดสองจุด โดยการแปลงเป็นสูตรคำนวณระยะทางดังรูปที่ 2.62



รูปที่ 2.62 แสดงการประยุกต์สูตรพีtagอรัสสำหรับการหาระยะทางระหว่างสองพิกัด

จากรูปที่ 2.62 เป็นสมการเพื่อใช้คำนวณหาระยะทางจากทฤษฎี แต่เนื่องจากสัมฐานของโลกไม่ใช่พื้นแบบราบ การคำนวณระยะทางจึงต้องมีการประยุกต์ เพื่อให้มีความถูกต้องตามระยะทางที่เป็นจริงตามความโค้งของโลก โดยการเพิ่มตัวแปรค่าความคลากรเคลื่อนเข้าไป เมื่อละติจูดมีการเปลี่ยนแปลงทุกๆ 1 องศา จึงได้สูตรใหม่ดังสมการที่ 2.17

$$d = \sqrt{(X_2 * a_2 - X_1 * a_1)^2 + (Y_2 * b_2 - Y_1 * b_1)^2} \quad (2.17)$$

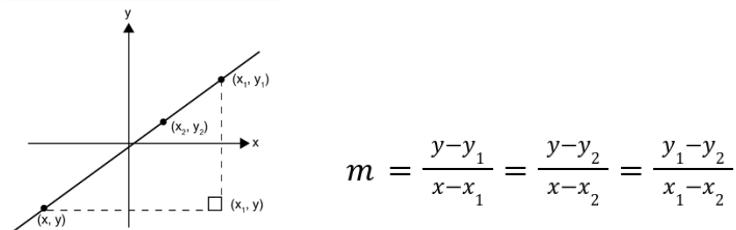
ตารางที่ 2.4 แสดงค่าความแตกต่างของระยะดูดและลงจิจูด เมื่อมีการเปลี่ยนแปลงระยะดูดทุกๆ 1 องศา

Latitude	N-S radius of curvature M (km)	Surface distance per 1 change latitude (km)	E-W radius of curvature N (km)	Surface distance per 1 change in longitude (km)
0	6335.44	110.574	6378.14	111.320
15	6339.70	110.649	6379.57	107.511
30	6351.38	110.852	6383.48	96.486
45	6367.38	111.132	688.84	78.486
60	6383.45	111.412	6394.21	55.800
75	6395.26	111.618	6396.15	28.800
90	6399.59	111.694	6399.59	0.000

จากตารางที่ 2.4 ตารางแสดงค่าความแตกต่างของระยะดูดและลงจิจูด เมื่อมีการเปลี่ยนแปลงระยะดูดทุกๆ 1 องศา และมีการนำค่าของระยะดูดที่  $15^\circ$  มาคำนวณเนื่องจากกรุงเทพมหานครตั้งอยู่ที่ละดูดที่ประมาณ  $15^\circ$  ดังนั้นจึงนำค่าการเปลี่ยนแปลงระยะทางของเส้นลงจิจูดที่เท่ากับ 110.649 กิโลเมตร และนำค่าการเปลี่ยนแปลงระยะทางของเส้นลงจิจูดที่เท่ากับ 107.551 กิโลเมตร มาใช้ในการคำนวณหาระยะทางจากพิกัดสองจุด เพื่อนำไปใช้เป็นค่า Setpoint ในการควบคุมระยะทางการเคลื่อนที่ของล้อหลัง

### 2.6.1.3 การคำนวณหาค่าความชันของเส้นตรงและสมการเส้นตรง

เมื่อระบบทำการคำนวณหาค่าระยะห่างระหว่างพิกัดสองจุดได้เรียบร้อย ระบบจะต้องทำการคำนวณหาค่าความชันเส้นตรงที่ลากจากพิกัดหนึ่งไปยังอีกพิกัดหนึ่ง โดยจะสามารถคำนวณหาค่าความชันของเส้นตรงได้ตามสูตรดังรูปที่ 2.63



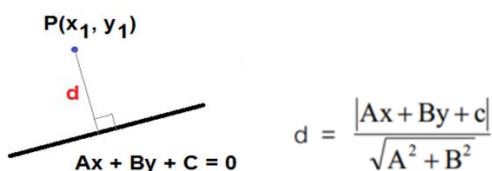
รูปที่ 2.63 สูตรในการหาค่าความชันของเส้นตรง

เมื่อทราบค่าความชันของเส้นตรงจากการคำนวณตามสูตรในรูปที่ 2.63 สามารถนำมาเขียนเป็นสมการเส้นตรงได้ โดยการนำค่าพิกัดของจุดใดๆ ที่เป็นส่วนหนึ่งของเส้นตรงมาเขียนเป็นสมการเส้นตรงได้ดังสมการที่ 2.18

$$y - y_1 = m(x - x_1) \quad (2.18)$$

### 2.6.1.4 การคำนวณหาค่าความผิดพลาดของการวิ่งของรถกอเล็ฟ

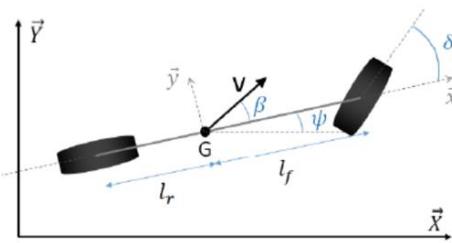
เมื่อระบบมีการคำนวณหาระยะทางระหว่างพิกัดและทราบถึงพิกัดของรถกอเล็ฟในขณะที่ติดตามเส้นทางอยู่ได้ จะทำให้ระบบสามารถที่จะคำนวณหาค่าความผิดพลาดของตำแหน่งรถที่ผิดไปจากเส้นทางที่ต้องการให้รถวิ่ง โดยสามารถคำนวณได้จากสูตรหาระยะทางจากจุดถึงเส้นตรง โดยที่เส้นตรงตั้งฉากกับเส้นที่ลากหมายจุด โดยในการคำนวณกำหนดให้จุดเป็นตำแหน่งของรถ และระยะจากจุดถึงเส้นตรงคือค่าความผิดพลาดที่รถวิ่งไป (Cross Track Error; CTE) ซึ่งสามารถคำนวณได้จากสูตรตามรูปที่ 2.64 ดังนี้



รูปที่ 2.64 สูตรที่มีการประยุกต์ใช้เพื่อคำนวณหาค่า Cross Track Error

### 2.6.1.2 Kinematic Bicycle Model

ในตัวความคุณการเดี่ยวของรถ จะใช้โมเดลแบบโมเดลจักรยาน (Kinematic Bicycle Model) ในการคำนวณ มุมเลี้ยว ซึ่งโมเดลดังกล่าวจะมีลักษณะที่คล้ายกับจักรยาน คือเป็นการคำนวณ โมเดลที่มีเพียงล้อหน้าและล้อหลัง ซึ่งในความเป็นจริงแล้วตัวรถจะมีลักษณะเป็นสี่ล้อ แต่เพื่อให้งานต่อการคำนวณจึงมีการนำโมเดลจักรยานมาพิจารณา ซึ่งโมเดลจะมีลักษณะดังรูปที่ 2.65



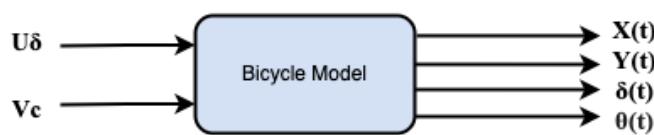
รูปที่ 2.65 Kinematic Bicycle Model of The Vehicle

$(x, y)$  คือศูนย์กลางมวลของรถ  $\psi$  คือทิศทางปัจจุบันของรถ (มุมมุ่งหน้า) และ  $v$  คือความเร็วของรถ  $l_f, l_r$  คือระยะห่างจากจุดศูนย์กลางมวลถึงเพลาหน้าและล้อหลังตามลำดับ  $\beta$  คือมุมของ  $v$  เทียบกับแกนรถ (มุมข้างของรถบนพื้นที่ CG) มุมบังคับเลี้ยวของล้อหน้าคือ  $\delta_f$  และความเร่งของรถคือ  $a$  เพื่อความง่าย เรายกตัวมันเป็นรถขับเคลื่อนล้อหน้า และจะกำหนดให้  $\delta_f$  เป็น  $\delta$

#### Bicycle Model

โมเดลของยานพาหนะที่ควบคุมได้ มีจำนวนมากและมีความซับซ้อนและความแม่นยำที่หลากหลาย ซึ่ง โมเดลของยานพาหนะที่ง่ายต่อการคำนวณและใช้กันมากที่สุด โมเดลหนึ่งคือ Bicycle Model สาเหตุที่ใช้ คำว่า Bicycle เนื่องจากล้อหน้าหันไปทางเดียว และล้อหลังหันไปทางเดียวกัน ใช้เป็นแบบล้อเดียว

คำว่า “โมเดลจักรยานจลนศาสตร์” เป็นการเรียกชื่อที่ผิด เนื่องจากไม่ใช้การสร้างแบบจำลองของจักรยาน แต่จริงแล้ว มันจะเป็นโมเดลที่ใช้ในการคุณยานพาหนะ ซึ่ง Bicycle Model จะมีข้อเสียในเรื่องของ ไม่สามารถคำนวณได้เมื่อยานพาหนะ เนื่องจากไม่ได้มีการนำโคลงของตัวรถมาพิจารณาร่วมด้วย



รูปที่ 2.66 Bicycle Model Block Diagram

ในโหมดจลนศาสตร์นี้มีวิธีการคำนวณอยู่ 2 แบบ คือ แบบเชิงเส้น (Linear Kinematic Bicycle Model) และ ไม่เป็นเชิงเส้น (Non-Linear Kinematic Bicycle Model) ซึ่งจะได้สมการเพื่อใช้ในการคำนวณดังนี้

Linear Kinematic Bicycle Model

$$\dot{x} = v \cos(\varphi)$$

$$\dot{y} = v \sin(\varphi)$$

$$\dot{\varphi} = \frac{v}{l_r}$$

$$\dot{v} = a$$

$$\beta = \tan^{-1} \left( \frac{l_r}{l_f + l_r} \tan \delta_f \right)$$

Non-Linear Kinematic Bicycle Model

$$\ddot{x} = \dot{\psi} \dot{y} + a_x$$

$$\ddot{y} = -\dot{\psi} \dot{x} + \frac{2}{m} (F_{c,f} \cos \delta_f + F_{c,r})$$

$$\ddot{\psi} = \frac{2}{l_z} (l_f F_{c,f} - l_r F_{c,r})$$

$$\dot{X} = \dot{x} \cos \psi - \dot{y} \sin \psi$$

$$\dot{Y} = \dot{x} \sin \psi - \dot{y} \cos \psi$$

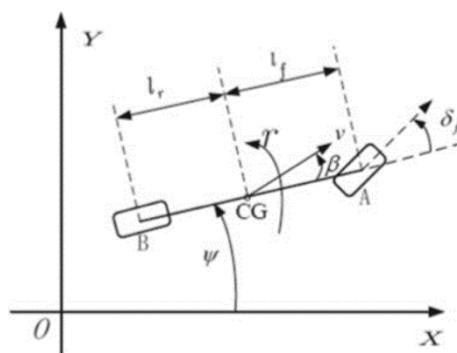
### 2.6.1.3 Tracking path

Tracking path เป็นพื้นฐานของการขับขี่แบบอัตโนมัติ ประสาทวิภาคของการติดตามเส้นทางจึงเป็นเรื่องยากที่จะกล่าวเกินจริง อัลกอริธึมสำหรับการติดตามเส้นทางเรขาคณิตเป็นวิธีที่ได้รับความนิยมมากที่สุด สำหรับยานยนต์อัตโนมัติ วิธีของสแตนเลย์ใช้เพลาหน้าเป็นจุดอ้างอิง ในขณะเดียวกันก็พิจารณาข้อผิดพลาดในการข้ามแทร็ก ซึ่งหมายถึงระยะห่างระหว่างจุดที่ใกล้ที่สุดบนเส้นทางเพลาหน้าของรถ โดยจะพิจารณาทั้งข้อผิดพลาดของหัวข้อปะปะข้อผิดพลาดในการติดตามอัลกอริธึมการติดตามเส้นทาง

#### ไอนามิกของยานพาหนะ

โฉนดยานยนต์จะประมาณค่าพารามิเตอร์ต่างๆ เพื่อให้โฉนดมีความเหมาะสมในการคำนวณดังนี้

- 1) ความเร็วของรถคงที่
- 2) แรงเวียดทานที่เกิดจากล้อเป็นศูนย์
- 3) ไม่คำนวณความต้านทานอากาศ



รูปที่ 2.67 ไอนามิกของยานพาหนะ

โฉนดรถจุนศาสตร์จากรูปข้างต้น จุดอ้างอิง CG คือจุดกึ่งกลางของแรงโน้มถ่วงของตัวรถ และค่าพิกัดระบุตำแหน่งของรถ ความเร็วของจุดพื้นที่ CG แสดงถึงความเร็วของรถ และมุมมุ่งหน้าและมุมบังคับเดียวของยานพาหนะนั้นคือมาจากมุมระหว่างแนวยาวของตัวรถ (AB) กับแนวนอนของหน้าต่างอ้างอิง กับมุมระหว่างแนวยาวของตัวรถ (AB) กับทิศทางของความเร็วของรถตามลำดับ

ภายในระบบพิกัดโลก เริ่มต้นจากตำแหน่ง CG ของรถ ทิศทางของแกน X คือได้ว่าเป็นทิศทางของรถและถือได้ว่าเป็นพิกัดตามยาวและแนวยาวของรถในขณะนั้น (*t*) ตามลำดับ จะได้สมการดังนี้

$$x_{t+1} = x_t + v_t \cos(\psi_t + \beta) \cdot dt$$

$$y_{t+1} = y_t + v_t \sin(\psi_t + \beta) \cdot dt$$

$$\psi_{t+1} = \psi_t + \frac{v_t}{l_r} \cdot \sin(\beta) \cdot dt$$

$$v_{t+1} = v_t + a \cdot dt$$

## 2.6.2 มอเตอร์ที่ใช้ในการควบคุมพวงมาลัย

ในการออกแบบระบบควบคุมมุมเลี้ยวเพื่อบังคับพวงมาลัยให้สามารถหมุนได้เองอัตโนมัติจะใช้มอเตอร์สำหรับพวงมาลัยพาหนะ คือ Steering Wheel Steering Gear Motor Model No.: KY185DD0101-10 ดังรูปที่ 2.69 โดยใช้ในการกำหนดองศาการหมุนของพวงมาลัย ซึ่งมีスペกค้างรูปที่ 2.68

Place of Origin:	Shandong, China	Brand Name:	Keya
Model Number:	KY185DD0101-10	Product name:	Auto steering wheel motor for precision agriculture
Voltage:	9-30VDC	Rated torque:	10N.m
Peak torque:	20N.m	Rated speed:	100rpm
Continuous current:	20A	Peak current:	50A
Feedback:	Incremental encoder, 48000ppr		

รูปที่ 2.68 Product Details of Motor



รูปที่ 2.69 Steering Wheel Steering Gear Motor

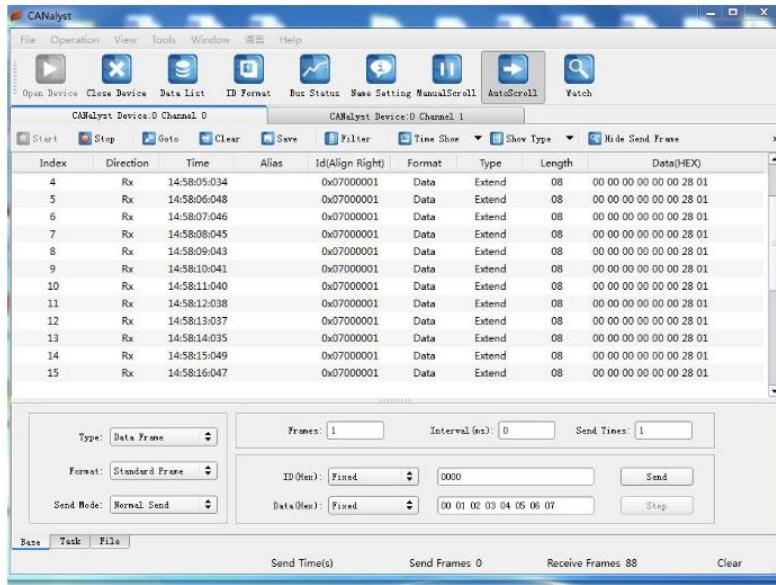
Parameter	Label	Parameter value	unit
Voltage	U	7-32	VDC
Max continuous current	I <sub>c</sub>	10	A
Max peak current	I <sub>max</sub>	15	A
PWM switching frequency	f <sub>pwm</sub>	10	kHz
Output encoder power supply	+5V <sub>out</sub>	5	VDC
	I <sub>cc</sub>	100	mA
Under voltage	V <sub>u</sub>	7 (adjustable)	V
Over voltage	V <sub>o</sub>	32(adjustable)	V
Operating temperature	Industrial grade (standard product)	-25 ~ +55	°C
	Military grade	-40 ~ +65	
Storage temperature	Industrial grade (standard product)	-35 ~ +65	°C
	Military grade	-55 ~ +85	

รูปที่ 2.70 ค่าพารามิเตอร์ของมอเตอร์

สำหรับการเชื่อมต่อการสื่อสารเพื่อสั่งงานระหว่างมอเตอร์กับ Jetson Xavier จะใช้ต่อผ่านสายแบบ CAN Bus (Controller Area Network) ทำให้สามารถส่ง-รับข้อมูลจากมอเตอร์ได้ โดยจะส่งสัญญาณเป็น Code ซึ่งอยู่ในรูปของเลขฐานสิบหก และสามารถนำค่าที่อ่านได้นั้นไปเขียนคำสั่งควบคุมค่าพารามิเตอร์และบังคับมุ่งเลี้ยวของมอเตอร์ต่อไป

#### 2.6.2.1 การสั่งงานมอเตอร์ควบคุมพวงมาลัยผ่าน KEYA ELECTRONIC

ในการควบคุมรถอัตโนมัติให้สามารถติดตามเส้นทางการเคลื่อนที่ได้ตามพิกัดอ้างอิงจีพีเอสจะต้องมีมอเตอร์ที่ใช้สำหรับการบังคับมุ่งเลี้ยวของรถ ซึ่งจะใช้มอเตอร์ยี่ห้อ KEYA เป็นมอเตอร์สำหรับควบคุมพวงมาลัยโดยเนพะ โดยในการสั่งงานมอเตอร์นั้นจะใช้การเชื่อมต่อผ่าน CAN bus โดยการเชื่อมต่อนั้นจะมีลักษณะการสื่อสารข้อมูลที่ต่างกัน ซึ่งจะมีการรับส่งข้อมูลดังรูปที่ 2.71



รูปที่ 2.71 แสดงรายละเอียดการรับส่งสัญญาณข้อมูลผ่าน CAN bus

จากรูปที่ 2.71 แสดงลักษณะการรับส่งข้อมูลของ CAN bus ซึ่งพบว่าข้อมูล (Data) ที่ส่งมานั้นจะเป็นโกล์ดชุดข้อมูลตัวเลขฐานสิบหก เพื่อกำหนด Position หรือค่ามุมเลี้ยวให้เป็นอินพุตค่าองศาการหมุนของมอเตอร์ แล้วจะไปคำนวนมุมเลี้ยวโดยการเปลี่ยนคำสั่งโปรแกรมควบคุมเพื่อส่งค่าไปยังมอเตอร์ให้ถูกต้องตามพิกัดอ้างอิงที่ได้มีการสร้างขึ้นต่อไป

### 2.6.3 การสื่อสารกับมอเตอร์เพื่อควบคุมมุมเลี้ยว Communication with hardware จุดเริ่มต้นของการสื่อสารแบบ CAN

มาตรฐานการสื่อสารแบบ CAN เป็นการสื่อสารแบบดิจิทัล ถูกพัฒนาขึ้นโดยกลุ่มบริษัท Robert Bosch จากประเทศเยอรมัน ในปี 1983 เป็นค่ายอเมริกัน Controller Area Network เพื่อใช้ในการสื่อสารภายในยานพาหนะ เพื่อทดแทนการส่งข้อมูลแบบอนาล็อก ทำให้มีการรับส่งข้อมูลได้มากขึ้น โดยใช้สายสัญญาณจำนวนน้อยลง ทำให้ได้รับความนิยมมากในกลุ่มอุตสาหกรรมยานยนต์ เช่น ระบบ ECUs (Electronic Control Units), Power train Control Module (ระบบต้นกำลัง = เครื่องยนต์), ระบบเบรก ABS, ระบบ SCS (Stability Control System) หรือ ESP (Electronic Stability Program) หรือ ระบบปรับสมดุลของช่วงล่างในสภาพถนนต่างๆ, ระบบ Steering, ระบบ Air Bag ตลอดจนระบบเกียร์อัตโนมัติ ซึ่งนับว่าระบบหรือโมดูลต่างๆ ได้ถูกนำมาติดตั้งใช้งานในรถยนต์กันมากขึ้น ดังนั้นการติดต่อสื่อสารระหว่างโมดูลต่างๆ ต้องมีความรวดเร็วตอบสนองการขับปั่นอย่างดีเยี่ยม

## การเชื่อมต่อการสื่อสารแบบ CAN

มาตรฐานการสื่อสารแบบ CAN ไม่ได้มีการกำหนดค่าจะต้องมีอุปกรณ์ภายในเครือข่ายไม่เกินกี่ตัว เนื่องจากขึ้นอยู่กับบีจัจจุลality ค้านประกอบกัน ได้แก่ รุ่นของ CAN Transceiver ที่ใช้ความยาวของสายในเครือข่ายและความเร็วของเครือข่ายที่กำหนดไว้เป็นต้น

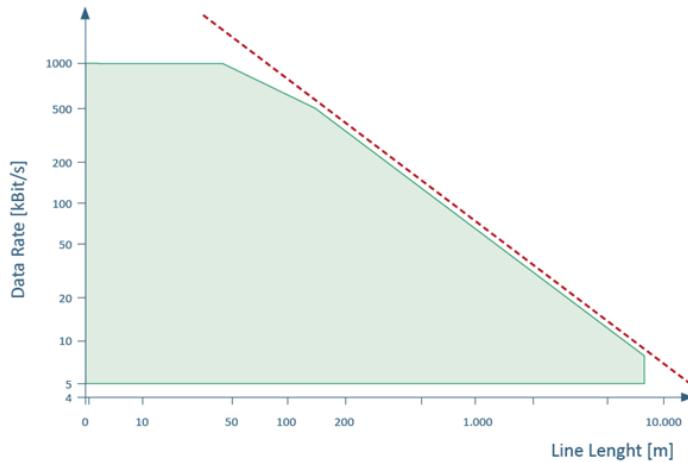
ผู้ใช้สามารถใช้สายไฟ 2 สาย แบบธรรมดานในการเชื่อมต่ออุปกรณ์สื่อสารในเครือข่าย CAN ได้ โดยสาหนึ่งจะเป็นสาย CAN High (2.5 – 3.5 VDC) และอีกสายหนึ่งจะเป็นสาย CAN Low (1.5 – 2.5 VDC) หากใช้สายคู่บิดเกลียวจะช่วยทำให้สัญญาณรบกวนแม่เหล็กไฟฟ้าน้อยลง ได้ ผู้ใช้สามารถเชื่อมต่ออุปกรณ์ได้โดยไม่ต่อสายกราวด์ เนื่องจากมีการออกแบบให้มีการเชื่อมต่อ กับส่วนชิดของคอนเนคเตอร์อยู่แล้ว เพื่อความสะดวกในการใช้งาน ทั้งนี้หากมีการนำไปเชื่อมต่อ กับอุปกรณ์อื่นที่อาจจะมีปัญหาระหว่าง Common-Mode Voltage ก็สามารถเชื่อมต่อกับสายกราวด์ของอุปกรณ์ได้

รูปแบบเครือข่ายการสื่อสารแบบ CAN ที่ได้รับความนิยมมากที่สุดแบบหนึ่ง เป็นการเชื่อมต่อสายสัญญาณหลักเป็นเส้นเดียว จากนั้นจึงเชื่อต่ออุปกรณ์ที่ต้องการจากสายสัญญาณหลัก และใช้ตัวต้านทานขนาด 120 โอห์ม ปิดที่จุดเริ่มต้นและปลายของเครือข่าย เพื่อไม่ให้เกิดการสะท้อนกลับของสัญญาณทางไฟฟ้า

ตามมาตรฐานการสื่อสารแบบ CAN นั้น อุปกรณ์ทุกตัวสามารถรับ-ส่งข้อมูลซึ่งกันและกัน ได้ มีการกำหนด ID ว่าข้อความดังกล่าวจะถูกส่งไปให้อุปกรณ์ตัวไหน หรือสามารถส่งแบบกระจาย (Broadcast) เพื่อให้อุปกรณ์ทุกตัวในเครือข่ายได้รับข้อความเดียวกันก็ได้ ซึ่งแตกต่างจากมาตรฐานการสื่อสารอุตสาหกรรมประเภท Modbus, Profinet, Profibus ฯลฯ ที่เป็นแบบ Master-Slave ในส่วนของการพัฒนาซอฟต์แวร์จะใช้เทคนิคการกรอง (Filter) ข้อความที่ไม่เกี่ยวข้องออกไป

## ความเร็ว และความยาวของสายสัญญาณ

มาตรฐานการสื่อสารแบบ CAN มีการใช้หลักการที่เรียกว่า bit-wise arbitration ซึ่งเป็นการประเมินร่วมกันระหว่างจำนวนอุปกรณ์ (Nodes) ความยาวสาย และความเร็วในการสื่อสาร (Baud Rate หรือ Bit Rate) ทำให้ผู้ใช้สามารถอ้างอิงจากรูปภาพด้านล่าง โดยมีการกำหนดว่าความเร็วสูงสุดของเครือข่ายคือ 1 Mbit/s และความยาวสายสูงสุดคือ 1,000 เมตร



รูปที่ 2.72 กราฟความสัมพันธ์ระหว่าง Data Rate – Line Length

ยกตัวอย่างการคำนวณได้ดังนี้

- หากใช้ความเร็วของเครือข่ายที่ 1 Mbit/s (หรือ 1000 kbit/s ตามรูปกราฟในรูปที่ xw) ควรใช้สายสัญญาณความยาวไม่เกิน 40 เมตร
- หากใช้สายสัญญาณความยาว 1 กิโลเมตร (หรือ 1,000 เมตร ตามรูปกราฟในรูปที่ xw) ควรใช้ความเร็วของเครือข่ายไม่เกิน 80 kbit/s หรือสามารถอ้างอิงได้ตามตารางที่ 2.5 ด้านล่าง

ตารางที่ 2.5 ค่าความสัมพันธ์ระหว่างความเร็วของเครือข่ายและความยาวสายสัญญาณ

ความเร็วของเครือข่าย (Baud Rate)	ความยาวสายสัญญาณ
1 Mbit/s	40 เมตร
500 kbit/s	110 เมตร
250 kbit/s	240 เมตร
125 kbit/s	500 เมตร
50 kbit/s	1.3 กิโลเมตร
20 kbit/s	3.3 กิโลเมตร
10 kbit/s	6.6 กิโลเมตร

โดยค่าความเร็วของเครือข่ายมาตรฐานที่นิยมใช้กันในปัจจุบันคือ 125 kbit/s CAN bus มีข้อได้เปรียบ และแตกต่างกับการสื่อสารผ่านสายแบบอื่นๆ ดังนี้

### **ทุกอุปกรณ์บนบัสคือตัวแม่ (Multi-master)**

ปกติการสื่อสารผ่านสายที่เชื่อมต่ออุปกรณ์หลายๆ ตัวเข้าด้วยกันบนบัสเดียวกัน เช่น Modbus RTU , I2C , SPI จะต้องมีตัวแม่ (Master) (พึงตัวเดียว และตัวที่เหลือเป็นตัวลูก (Slave)) โดยตัวลูกจะส่งข้อมูลได้ก็ต่อเมื่อตัวแม่ร้องขอเท่านั้น สำหรับ CAN bus อุปกรณ์ทุกตัวคือตัวแม่ การส่งข้อมูลจะส่งให้ใคร เมื่อไรก็ได้ ไม่มีใครเป็นตัวควบคุม แก้ปัญหาหากตัวแม่พังเสียหาย ระบบจะไม่สามารถใช้งานได้ทั้งหมด (บน CAN bus หากมีอุปกรณ์ใดเสียหาย อุปกรณ์นั้นๆ จะแตะตัวเองออก อุปกรณ์อื่นๆ ยังสื่อสารกันได้ปกติ)

### **ทุกอุปกรณ์ได้รับข้อมูล แต่เลือกรับข้อมูลได้ (Multi-cast)**

การส่งข้อมูลบน CAN bus คือการส่งที่ทุกอุปกรณ์ได้รับข้อมูลทั้งหมด (broadcast) ทั้งนี้หากอุปกรณ์ใดต้องการเลือกรับเฉพาะบางข้อมูล (Multi-cast) ก็สามารถทำได้ เช่นกัน

### **มีการตรวจจับความผิดพลาดและแจ้งเตือน (Error Detection and Signaling)**

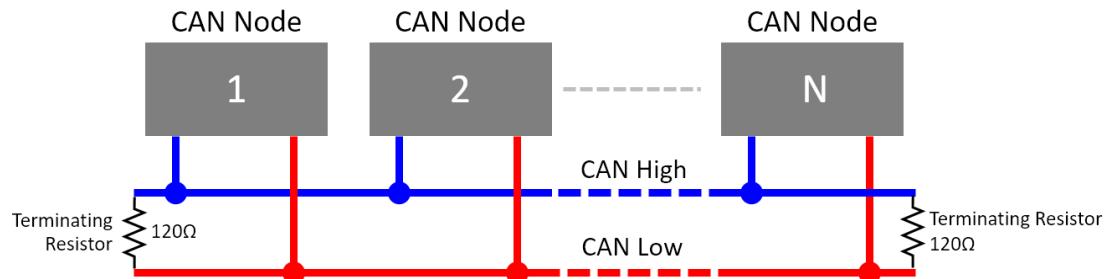
ทุกอุปกรณ์บน CAN bus จะมีการตรวจสอบข้อมูลที่วิ่งอยู่ในบัสเสนอ หากมีอุปกรณ์ใดตรวจพบความผิดพลาดของการส่งข้อมูล อุปกรณ์นั้นจะส่งข้อมูลแจ้งเตือนทันที

### **มีการจัดลำดับความสำคัญของข้อมูล (Message Priorities)**

หากเกิดเหตุการณ์ที่มีอุปกรณ์ใดๆ ส่งข้อมูลพร้อมกันขึ้น ข้อมูลที่มีความสำคัญมากกว่าจะได้รับสิทธิ์ในการส่งก่อน ส่วนข้อมูลที่มีความสำคัญน้อยกว่าจะได้โอกาสส่งใหม่ในภายหลัง

### **CAN ในฐานะ Physical Layer**

Physical Layer หรือ ชั้นกายภาพ เป็นขั้นตอนการเชื่อมต่อสาย, CAN bus ใช้สายในการเชื่อมต่อระหว่างส่วนควบคุม (ECU หรือไมโครคอนโทรลเลอร์หรือ CAN Device) ด้วยสาย 2 เส้น เชื่อมต่ออุปกรณ์ที่ต้องการสื่อสารทั้งหมดเข้าด้วยกัน (ดังรูปที่ 23) ประกอบด้วยสาย CAN High (CANH) และ CAN Low (CANL) ที่ปลายสายทั้ง 2 ด้าน ต่อตัวต้านทาน  $120 \Omega$  (เรียกว่า Terminating Resistor) เพื่อดampen overshoot for high drive lines and reduce signal noise

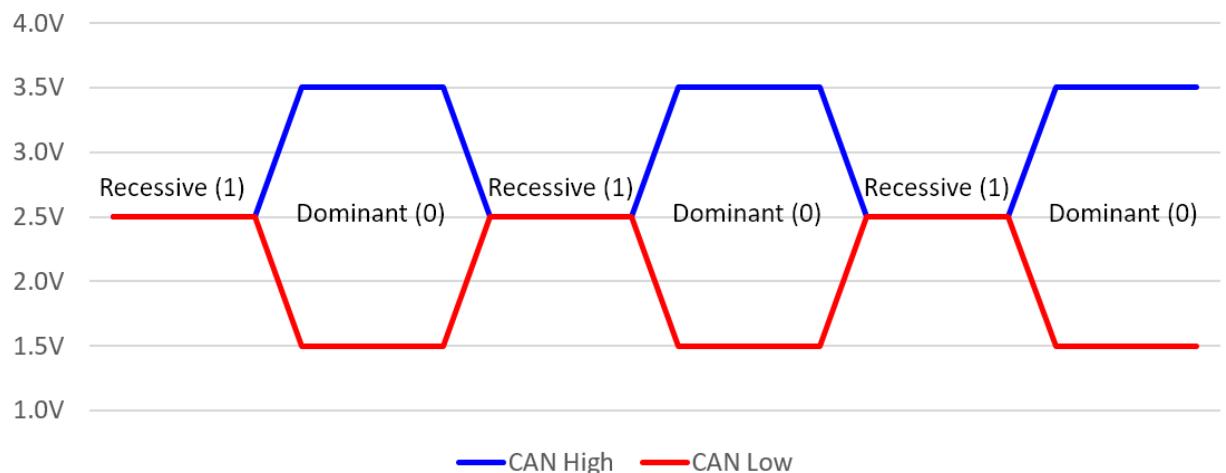


รูปที่ 2.73 การเชื่อมต่อระหว่างส่วนควบคุมด้วย CAN bus

สาย CANH และ CANL ทำงานแบบ differential wire คือใช้ความแตกต่างของแรงดันไฟฟ้าระหว่างสาย 2 เส้นในการรับ-ส่งข้อมูล โดยมีวัตถุประสงค์เพื่อลดสัญญาณรบกวน สัญญาณภายในสายประกอบด้วย 2 สถานะ คือ

1. สถานะ Dominant เกิดขึ้นเมื่อแรงดันของสาย CANH มากกว่าสาย CANL แปลงเป็นสถานะส่ง โลจิก 0
2. สถานะ Recessive เกิดขึ้นเมื่อแรงดันของสายเส้น CANH น้อยกว่าหรือเท่ากับ CANL แปลงเป็นสถานะส่ง โลจิก 1

จากรูปที่ 2.74 จะเห็นว่า ในสถานะ Dominant (ส่งโลจิก 0) แรงดันของสาย CANH มีค่าประมาณ 3.5V ส่วน CANL มีค่าประมาณ 1.5V ในสถานะ Recessive (ส่งโลจิก 1) แรงดันของสาย CANH และ CANL มีค่าเท่ากันคือ 2.5V สาย CANH และ CANL มีแรงดันแตกต่างกัน 0V



รูปที่ 2.74 แรงดันไฟฟ้าในสาย CAN High (CANH) และ CAN Low (CANL)

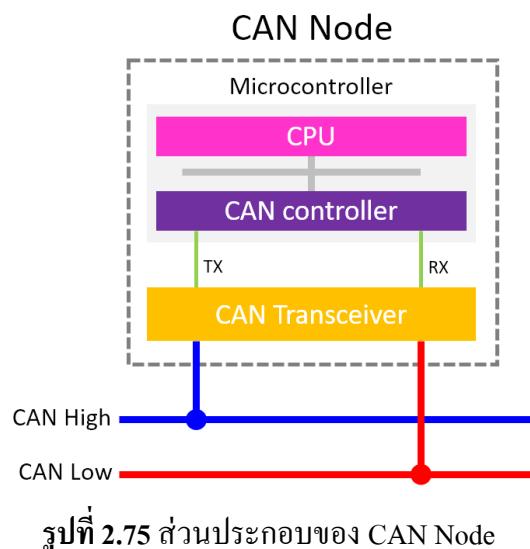
โนนด (Node) หรืออุปกรณ์ CAN (CAN Device) หรือส่วนควบคุม (ECU) ภายในประกอบด้วย 2 ส่วน คือ

### 1. CAN controller

CAN controller เป็นวงจรไฟฟ้าที่ออกแบบมาสำหรับใช้ควบคุมการรับ-ส่งข้อมูลผ่าน CAN โดยเฉพาะ มักฝังมาภายใน ไมโครคอนโทรลเลอร์ที่สเปกสูง หรือถูกออกแบบมาเพื่องานด้าน Automotive โดยเฉพาะ ไมโครคอนโทรลเลอร์ที่ฟัง CAN controller มาในตัว เช่น ESP32 ATSAME51 เป็นต้น ในไมโครคอนโทรลเลอร์ที่ไม่มี CAN controller สามารถต่อไอซี CAN controller ภายนอกเพิ่มได้ เช่น MCP2515 เป็นต้น

### 2. CAN Transceiver

CAN Transceiver ทำหน้าที่แปลงสัญญาณลอจิก 0 และ 1 แบบ TTL (ลอจิก 1 = 3.3V/5V, ลอจิก 0 = 0V) ให้เป็นสัญญาณเพื่อส่งออก CANH และ CANL โดย CAN Transceiver เป็นอุปกรณ์ที่ต้องต่อแยก ออกจากไมโครคอนโทรลเลอร์ ไอซี CAN Transceiver มีผลิตหลายบริษัท แต่ละบริษัทแต่ละรุ่น จะมีสเปกด้านความเร็วในการรับ-ส่งข้อมูล (Data Rate) ที่แตกต่างกัน และใช้แรงดันไฟเลี้ยงต่างกัน (มีรุ่นใช้ไฟเลี้ยง 5V กับรุ่นใช้ไฟเลี้ยง 3.3V) ตัวอย่าง ไอซี CAN Transceiver ได้แก่ SN65HVD232DR TJA1050 BD41041FJ-CE2 โดยส่วนใหญ่ ไอซี CAN Transceiver ตัวดังจะเป็น SOIC-8 มีขาที่ตรงกัน ทุกเบอร์ ดังนั้นจึงใช้แทนกันได้ทั้งหมด



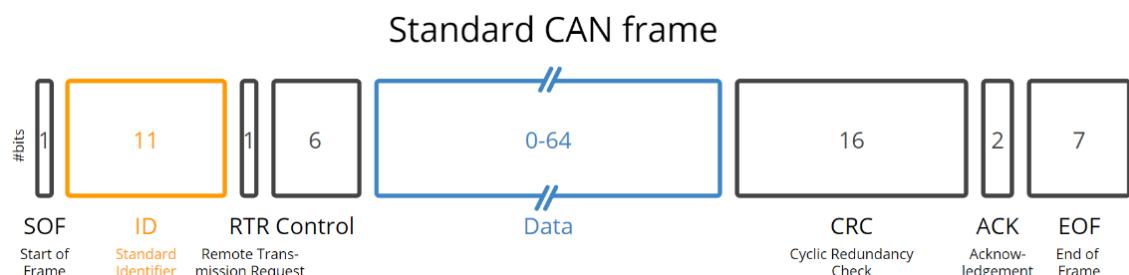
## การรับ-ส่งข้อมูลผ่าน CAN

การรับ-ส่งข้อมูลผ่าน CAN ใช้สิ่งที่เรียกว่า CAN frame และ 1 CAN frame หมายถึงการส่งข้อมูล 1 ครั้ง โดย CAN frame แบ่งเป็น 4 ชนิดดังนี้

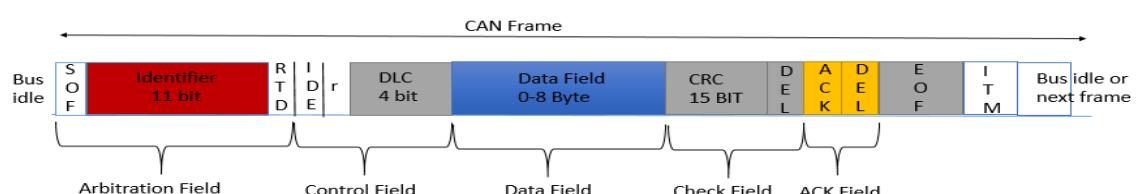
1. Data frame – ใช้ส่งข้อมูลไปยังอุปกรณ์ CAN อื่นๆ
2. Remote frame – ใช้ร้องขอข้อมูลจากอุปกรณ์ CAN อื่นๆ
3. Error frame – ใช้แจ้งทุกอุปกรณ์ CAN ว่าพบความผิดพลาดขึ้นบนบัส
4. Overload frame

การใช้งานจริงมีเพียง Data frame และ Remote frame เท่านั้นที่ผู้ใช้สามารถเขียนโปรแกรมสั่งงานได้ ส่วน Error frame และ Overload frame ตัว CAN controller จะจัดการให้อัตโนมัติ

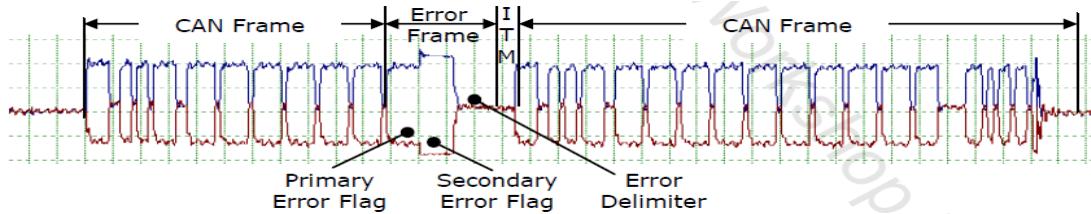
Data frame และ Remote frame มี 2 รูปแบบ คือ Standard Frame และ Extended Frame โดย Standard Frame มีโครงสร้างดังนี้



รูปที่ 2.76 โครงสร้าง Standard Frame (ที่มา; csselectronics.com)



รูปที่ 2.77 ส่วนประกอบต่างๆของ CAN bus data frame



รูปที่ 2.78 รูปร่างสัญญาณ CAN bus โดยที่ใช้เครื่องมือวัดสัญญาณ

1. SOF: Start of Frame (1 บิต) - ส่ง Dominant (ส่งลอจิก 0) ตัวที่บอกสถานะการรับส่งระหว่างผู้รับ (Receivers) และผู้ส่ง (Sender) เพื่อบอกให้ทุกอุปกรณ์บนบัสรับรู้ว่ากำลังจะมีการส่งข้อมูล
2. ID: Identifier (11 บิต สำหรับ Standard Frame) - หมายเลขเฉพาะของชุดข้อมูลนี้ โดยอาจกำหนดเป็นหมายเลขอุปกรณ์ หรือหมายเลขของข้อมูลที่ตั้งขึ้นมาเฉพาะกีตี้ได้ โดยหมายเลขนี้จะเป็นตัวกำหนดความสำคัญของข้อมูลด้วย หากมีค่าน้อย หมายถึงมีความสำคัญมาก
3. RTR: Remote Transmission Request (1 บิต) - บิตกำหนดว่าเป็น Remote frame หรือไม่ ดังนี้
  - RTR = Dominant (ลอจิก 0) หมายถึง ข้อมูลชุดนี้เป็น Data frame
  - RTR = Recessive (ลอจิก 1) หมายถึง ข้อมูลชุดนี้เป็น Remote frame
4. Control (6 บิต) ประกอบไปด้วย
  - IDE: Identifier extension bit (1 บิต) - ใช้ตรวจสอบว่าเป็น Standard Frame หรือ Extended Frame
  - Reserved (1 บิต) - สงวนบิตนี้ไว้สำหรับเฟิร์มแวร์ในอนาคต มีค่าเป็น Dominant (ลอจิก 0) เสมอ
  - DLC: Data Length Code (4 บิต) - แบ่งเป็น 2 กรณีดังนี้
    - i. ถ้า RTR = Dominant (ลอจิก 0, Data frame) ใช้บอกความยาวของข้อมูลที่ส่ง
    - ii. ถ้า RTR = Recessive (ลอจิก 1, Remote frame) ใช้บอกความยาวของข้อมูลที่ร้องขอ
5. Data (0 ถึง 64 บิต หรือ 0 ถึง 8 ไบต์) - ข้อมูลที่ต้องการส่ง
6. CRC (16 บิต) - แบ่งดังนี้
  - CRC (15 บิต)
  - CRC delimiter (1 บิต) - มีค่าเป็น Recessive (ลอจิก 1) เสมอ
7. ACK (2 บิต) - แบ่งดังนี้
  - ACK slot (1 บิต) - ฟังส่งจะปล่อยให้บัสเป็นสถานะ Recessive (ลอจิก 1) หากมีอุปกรณ์ใดๆ ในบัสได้รับ และยืนยันว่าข้อมูลที่ส่งถูกต้อง บิตนี้จะถูกดึงเป็น Dominant (ลอจิก 0)
  - ACK delimiter (1 บิต) - มีค่าเป็น Recessive (ลอจิก 1) เสมอ
8. EOF (7 บิต) - ใช้ส่งเพื่อบอกสิ้นสุด CAN frame

Extended Frame แตกต่างจาก Standard Frame ตรงที่ฟิลด์ ID มีความยาว 29 บิต และมีโครงสร้างอื่นๆ ที่แตกต่างกันเล็กน้อย

การใช้งานจริง ฟิลด์ที่กำหนดค่าได้คือ ID, RTR, DLC และ Data ส่วนฟิลด์อื่นๆ ตัว CAN controller จะจัดการให้อัตโนมัติ

### ความเร็วในการรับ-ส่งข้อมูล

CAN เป็นโปรโตคอลแบบ Asynchronous ความเร็วในการรับ-ส่งข้อมูลถูกกำหนดโดย Data Rate หรือ Baud rate ซึ่งสามารถกำหนดได้ดังนี้

- 12,500 (12.5 kbit/s)
- 16,000 (16 kbit/s)
- 20,000 (20 kbit/s)
- 25,000 (25 kbit/s)
- 50,000 (50 kbit/s)
- 100,000 (100 kbit/s)
- 125,000 (125 kbit/s) – โดยปกติใช้ค่านี้
- 250,000 (250 kbit/s)
- 500,000 (500 kbit/s)
- 800,000 (800 kbit/s)
- 1,000,000 (1 Mbit/s)

อุปกรณ์บนบัส CAN จำเป็นจะต้องกำหนด Data Rate หรือ Baud rate ให้เท่ากันทุกตัว เพื่อให้สามารถรับ-ส่งข้อมูลได้ถูกต้อง

### ความเร็วในการสื่อสารระดับต่างๆ (Speed Class of Data Bus)

1. Class A ความเร็วในการรับส่งข้อมูลน้อยกว่า 10 kbit/s (10 kbps) ระบบที่ใช้ได้แก่รถจากไฟฟ้า, เบเย่ไฟฟ้า, ระบบล็อกต่างๆ, รีโมท, ระบบไฟส่องสว่าง เป็นต้น
2. Class B ความเร็วในการสื่อสารอยู่ระหว่าง 10-125 kbps ซึ่งก็ได้แก่ Protocol ISO 9141-2 หรือ SAE J 1850 (ใช้ในรถค่าย Ford, GM และ ยูโรป + อเมริกา ทั่วไป) ระบบนี้จะมีความเร็วเพียงพอในการสื่อสารข้อมูลที่มีความซับซ้อน ได้แก่ ระบบเกียร์อัตโนมัติ, ระบบปรับอากาศ, ระบบ Immobilize, ระบบอิเล็กทรอนิกส์บริเวณแพงค์คุณต่างๆ เป็นต้น

3. Class C ทำความเร็วในการสื่อสารอยู่ที่ประมาณ 1 Mbps หรือ 1000 kbps หรือ ประมาณ 10 เท่าของ Class B แต่โดยทั่วไปจะใช้ที่ความเร็วประมาณ 500 kbps ซึ่งมีความเร็วเพียงพอสำหรับระบบ Air Bag, ABS, Stability Control, Traction Control, Power Train Control Module (ระบบต้านกำลัง หมายถึง เครื่องยนต์ + เกียร์ เป็นหลัก)
4. Class D ความเร็วในการสื่อสารประมาณ 1 Mbps ใช้ในระบบ Onboard Entertainment เช่น Video Streaming หรือ ระบบติดต่อสื่อสารกับดาวเทียม หรือ ระบบ Internet 3G ที่อนาคตจะนำมาใช้ในรถยนต์

โปรดตอกย้ำว่า CAN ได้แก่ SAE J 1939, GMLAN, OBD 2, SAE J 1587, LIN ซึ่งเป็นหน้าที่ของวิศวกรรมยานยนต์ที่จะเลือกใช้โปรดตอกย้ำให้เหมาะสมกับยานยนต์นั้นๆ ซึ่งหากรถยนต์นั้นๆ ต้องการความเร็วในการสื่อสารมากๆ การเลือกใช้ Fiber Optic แทนสายไฟธรรมดาก็อาจเป็นสิ่งที่ต้องพิจารณาต่อไป

### **การติดต่อสื่อสารระหว่างโมดูล**

โมดูลต่างๆ ในระบบ CAN จะถูกจัดเป็น Node และมี Address ที่แน่นอน เมื่อแต่ละ Node สื่อสารกันก็จะรู้ว่ามาจาก Node ไหน ด้วยการส่ง Code ของ Node นั้นๆ ไปด้วย ทำให้แต่ละ Module หรือ Node รู้จักกัน

การสื่อสารข้อมูลจะส่งสัญญาณ digital หรือ 0, 1 คือ แรงดันไฟฟ้าระดับต่ำคือ 0 และแรงดันไฟฟ้าระดับสูงคือ 1 ซึ่งจะแตกต่างกันไปแล้วแต่ผู้ผลิตรถยนต์ แต่โดยทั่วไปจะใช้ค่าระหว่าง 5-7 โวลต์

### **การเลือกรับข้อมูล (Acceptance Filter)**

เนื่องจากการรับ-ส่งข้อมูลบน CAN เป็นแบบ Broadcast ดังนั้นจึงอาจมีข้อมูลที่ไม่ต้องการเข้ามาด้วย หากมีข้อมูลที่ไม่ต้องการเข้ามากเกินไปก็อาจจะทำให้แรม หรือพื้นที่เก็บข้อมูล CAN ขาดเสื่อมได้ ดังนั้น CAN จึงมีสิ่งที่เรียกว่า Acceptance Filter เพื่อใช้กรองให้รับเฉพาะบาง CAN frame ที่มี ID ที่กำหนดไว้เท่านั้น

### **สถานะความผิดพลาดและตัวนับ (Error States and Counters)**

CAN controller ของทุกโนนด (CAN Node) มีตัวนับที่ชื่อ Transmit Error Counter (TEC) และ Receive Error Counter (REC) ทำหน้าที่นับจำนวนความผิดพลาดที่เกิดจากการรับ-ส่งข้อมูลบนบัส โดยตัวนับแต่ละตัวใช้อ้างอิงสถานะความผิดพลาด (Error States) ดังนี้

- Error Active คือ สถานะที่ตัวนับ TEC และ REC นับค่าได้น้อยกว่า 128 การรับ-ส่งข้อมูลยังเป็นไปตามปกติ แต่จะมีการส่ง Error frame ทุกรั้งที่พบความผิดพลาด
- Error Passive คือ สถานะที่ตัวนับ TEC หรือ REC นับค่าได้มากกว่าหรือเท่ากับ 128 การรับ-ส่งข้อมูลยังเป็นไปตามปกติ แต่จะมีการส่ง Error frame ทุกรั้งที่พบความผิดพลาด
- Bus-Off คือ สถานะที่ตัวนับ TEC นับค่าได้มากกว่าหรือเท่ากับ 256 ในสถานะนี้ CAN controller จะไม่รับ-ส่งข้อมูลบนบัสอีกต่อไป จนกว่าจะได้รับการรีเซ็ต

### ข้อดีของระบบ CAN

1. มีความเร็วในการสื่อสารข้อมูล ทำให้การตอบสนองของแต่ละโมดูลมีประสิทธิภาพ และจำเป็นอย่างยิ่งสำหรับบางระบบ เช่น ABS
2. รองรับ Video Streaming
3. สามารถใส่โมดูลต่างๆ เข้าไปได้มาก
4. มีสายไฟน้อยกว่าเพราระใช้ระบบ Network Cable หรือ Gateway

### ข้อเสียของระบบ CAN

1. จุดต่อของ Node ต่างๆ อาจหลวมหรือเป็นสนิม ทำให้การติดต่อสื่อสารมีปัญหา
2. หาก Node หรือ Module มีการซื้อต่องคราวค์ก็อาจทำให้ระบบทั้งหมดล้มเหลวได้
3. กรณีระดับแรงดันไฟฟ้าต่ำ การติดต่อสื่อสารอาจล้มเหลวได้
4. กรณีแบตเตอรี่ไม่เพียงพอหรือมีการถอดออก อาจทำให้ต้อง Set ค่า Module ใหม่ ติดต่อกับเครื่อข่ายได้ หรือเรียกว่าการทำ Relearn ใหม่ หรือ Re-establish
5. การถอดสายไฟหรืออุปกรณ์บางตัวไม่ทำงาน อาจทำให้ระบบบันทึกล้มเหลวได้ เช่น ระบบปรับอากาศจะไม่ทำงาน หาก Step Motor ควบคุมการผสมอากาศเย็นกับ Heater ไม่ทำงาน รถยนต์กันน้ำก็จะใช้ระบบปรับอากาศไม่ได้โดย เป็นต้น

การติดต่อสื่อสารของ Node ต่างๆ ใน Serial Bus นี้ หากมีปัญหาเกิดขึ้น ระบบ DTC หรือ MIL ก็จะแสดงขึ้นและหากใช้เครื่องสแกนจับคู่ก็จะพบໂຄດที่ขึ้นต้นด้วย U เช่น U 1026 ของ GM คือ Loss of communication between the controller and transmission controller หรือ การติดต่อระหว่างโมดูลเครื่องยนต์กับโมดูลระบบส่งกำลังล้มเหลว

### การตรวจสอบปัญหา ทำได้โดย

1. ตรวจสอบระบบกราวด์ มีการซื้อต่องกราวด์หรือไม่
2. ตรวจสอบระดับแรงดันไฟฟ้าอยู่ในมาตรฐานหรือไม่
3. ตรวจสายไฟ

หากตรวจทั้ง 3 รายการแล้วไม่พบข้อบกพร่องใดๆ แสดงว่าไม่มีคุณน้ำมีปัญหา (ส่วนมากต้องเปลี่ยนใหม่) และเนื่องจากระบบ CAN มีโมดูลต่างๆ ต่อพ่วงกันแบบเครื่อข่ายไว้เป็นจำนวนมาก ดังนั้นแต่ละโมดูลจำเป็นต้องมีระบบ Sleep Mode ในขณะที่ดับเครื่องยนต์ เพื่อป้องกันไม่ให้แบตเตอรี่หมด ยกเว้นระบบป้องกันไฟไหม้ที่ไม่มี Sleep Mode หรือ ระบบ Air Bag ที่ต้องใช้เวลาสักครู่หลังดับเครื่องยนต์ จึงจะเข้าสู่ Sleep Mode

### ความจำเป็นของเครื่อง Scan Tool

จากสถิติในปี 2007 ที่ประเทศไทยพบว่าอุบัติเหตุทางท้องถนน 182,115 ครั้ง ก็มาจากคนขับขาดความระมัดระวัง 50%, ความบกพร่องของรถยนต์ 25% และสภาพถนน ทัศนวิสัย 25% ซึ่งแน่นอนว่าเป็นสิ่งที่หลีกเลี่ยงไม่ได้ แต่ความสามารถลดลง 25% จากความบกพร่องของรถยนต์คงได้ด้วยระบบที่ทันสมัยหรือระบบ Diagnostic ที่ใช้ในรถยนต์ และสิ่งที่จำเป็นต่อมาก็คือ เครื่องมือ Scan Tool ซึ่งในอนาคตอาจจำเป็นต้องมีประจำรถ

### เครื่อง Diagnostic ประเภทต่างๆ ได้แก่

1. Diagnostic Tester สำหรับระบบตรวจสอบเครื่องยนต์, เกียร์, ABS, Air Bag, Electronic device ต่างๆ
2. Coil and Plug Tester สำหรับตรวจสอบสภาพของคอล์จูดราเบิดและหัวเทียน
3. Batter Management and Tester สำหรับตรวจสอบสภาพแบตเตอรี่, ชาร์จไฟฟ้า เป็นต้น
4. Air Condition Equipment and Tool ได้แก่ เครื่องมือเช็คระดับของระบบแอร์, อุปกรณ์ซ่อมบำรุง อื่นๆ เป็นต้น
5. Emission Analyzer อุปกรณ์ตรวจสอบการปลดปล่อยไอเสีย
6. Key Programming Tool อุปกรณ์ทำกุญแจสำรองที่มีไมโครชิพ
7. Data Recorder อุปกรณ์บันทึกการขับขี่หรือบันทึกการทำงานของรถยนต์ ซึ่งในอนาคตจะมีการนำอุปกรณ์บันทึกข้อมูลเหมือนกล้องดำเนินการร่องรอยของมนุษย์ โดยบันทึกการขับขี่ตลอดจนสภาพต่างๆ ขณะเกิดการชน ได้แก่ เวลา, ทิศทางการชน, ความเร็ว, ความรุนแรง, อัตราเร่ง - หน่วง, การเบรก เป็นต้น ซึ่งทำให้การ Claim ประกันหรือการพิจารณาของผู้พิพากษาทำได้ง่ายขึ้น

## 2.7 ภาพรวมของระบบ Robot Operating System

Robot Operating System (ROS) คือ ระบบสำหรับเขียนซอฟต์แวร์และจัดการไฟล์อย่างเป็นระเบียบเพื่อควบคุมการทำงานของหุ่นยนต์ โดยรวมรวมชุดเครื่องมือและชุดคำสั่งต่างๆ ที่จำเป็นในการพัฒนาหุ่นยนต์ ซึ่งจะช่วยลดความยุ่งยากในการสร้างและพัฒนาหุ่นยนต์ที่มีความซับซ้อน อีกทั้งยังสามารถเพิ่มประสิทธิภาพในการพัฒนาแพลตฟอร์มหุ่นยนต์ที่หลากหลาย โดย ROS จะมีหลายเวอร์ชันด้วยกัน เช่น ROS noetic, ROS melodic, ROS kinetic เป็นต้น ซึ่ง ROS แต่ละเวอร์ชันจะเหมาะสมกับลักษณะการทำงานที่แตกต่างกันและส่วนใหญ่ออกแบบมาเพื่อรองรับกับระบบปฏิบัติการ Ubuntu (Linux) ซึ่ง ROS แต่ละเวอร์ชันจะเหมาะสมกับการติดตั้งในระบบปฏิบัติการ Ubuntu ในเวอร์ชันแตกต่างกันออกไป เช่น ROS kinetic จะเหมาะสมกับระบบปฏิบัติการ Ubuntu 16.04 หรือ ROS Noetic จะเหมาะสมกับระบบปฏิบัติการ Ubuntu 20.04 เป็นต้น

The screenshot shows the ROS.org website with the navigation bar: Documentation, Browse Software, News, Download. Below the navigation, there's a link to 'lunar'. The main content area shows the title 'ROS Lunar Loggerhead' and a brief description: 'ROS Lunar Loggerhead is the eleventh ROS distribution release. It was released on May 23rd, 2017.' To the right is a cartoon illustration of an astronaut holding a flag with the ROS logo, standing on the moon with craters in the background. The text 'LUNAR•LOGGERHEAD' is written in a stylized font at the bottom of the illustration. On the left, there's a sidebar with 'Contents' and a list of links: 1. ROS Lunar Loggerhead (which is active), 1. Platforms, 2. Installation, 3. Release Planning, 4. Changes. On the right, there's a sidebar for 'ROS 2 Documentation' with links to 'Wiki', 'Distributions', 'ROS/Installation', 'ROS/Tutorials', 'RecentChanges', and 'lunar' (which is active). There are also 'Page' and 'More Actions:' sections.

**罗斯组织**

About | Support | Discussion Forum | Index | Service Status | Q&A answers.ros.org

Documentation      Browse Software      News      Download

[lunar](#)

**ROS Lunar Loggerhead**

ROS Lunar Loggerhead is the eleventh ROS distribution release. It was released on May 23rd, 2017.

**Contents**

- ROS Lunar Loggerhead
  - Platforms
  - Installation
  - Release Planning
  - Changes

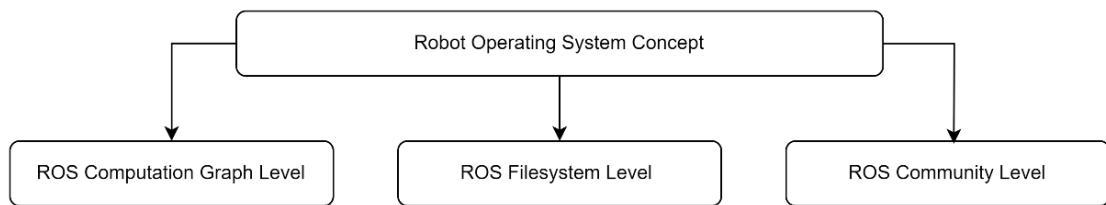
**1. Platforms**

ROS Lunar Loggerhead is primarily targeted at the Ubuntu 17.04 (Zesty) release, though other Linux systems as well as Mac OS X, Android, and Windows are supported to varying degrees. For more information on compatibility on other platforms, please see [REP 3: Target Platforms](#). It will also support Ubuntu 16.10 Yakkety, Ubuntu LTS 16.04 Xenial and Debian Stretch.

Rossi 2.79 ตัวอย่างรายละเอียดแพลตฟอร์มที่เหมาะสมกับ ROS เวอร์ชัน Lunar

โดย ROS สามารถแบ่ง Concept ของระบบออกได้เป็น 3 ส่วน คือ 1. ROS Computation Graph Level

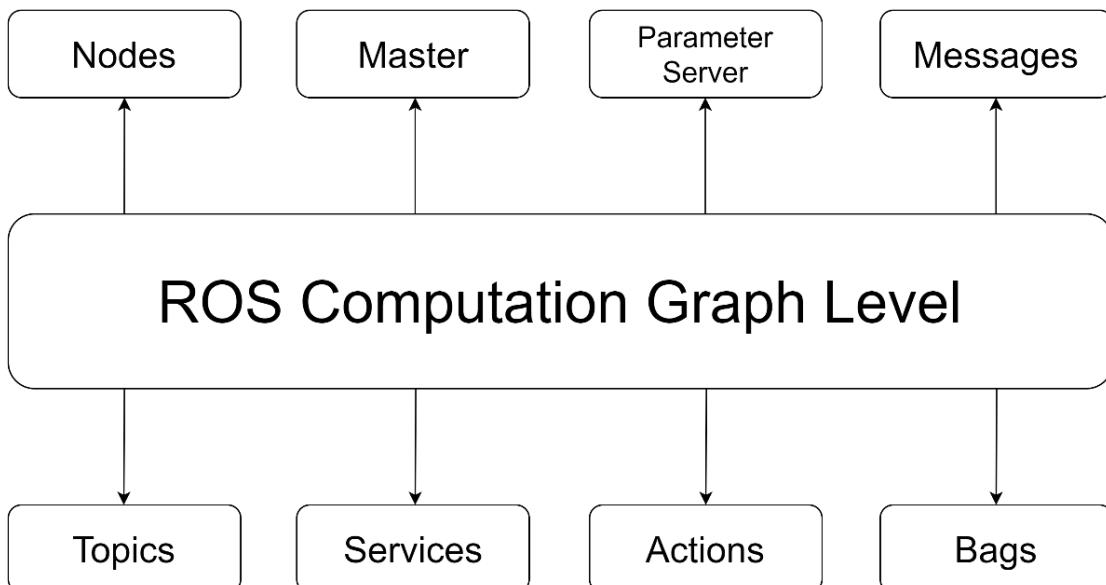
ROS Filesystem Level 3. ROS Community Level



รูปที่ 2.80 Robot Operating System Concept

### 2.7.1. ROS Computation Graph Level

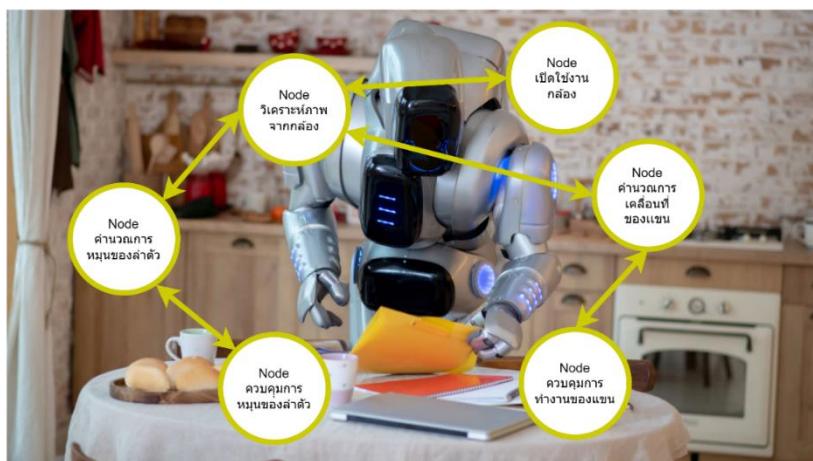
Concept ของระบบ ROS ที่จะแบ่ง Module การทำงานของหุ่นยนต์ออกเป็นส่วนๆ แล้วให้แต่ละส่วน สื่อสารข้อมูลหรือข้อมูลหากันผ่านทาง Topic ต่างๆ ในรูปแบบ Peer – to Peer โดยจะมี ROS Master เป็นส่วนที่ใช้บริหารจัดการการทำงานของระบบทั้งหมด ซึ่งส่วนประกอบของ Computation Graph Level จะมีหลายส่วนประกอบที่ควรทำความเข้าใจ เช่น Nodes, Master, Parameter Server, Messages, Topic เป็นต้น โดยแต่ละส่วนย่อมมีรายละเอียด ดังนี้



รูปที่ 2.81 องค์ประกอบส่วนต่างๆ ของ ROS Computation Graph Level

### 2.7.1.1 ROS Nodes

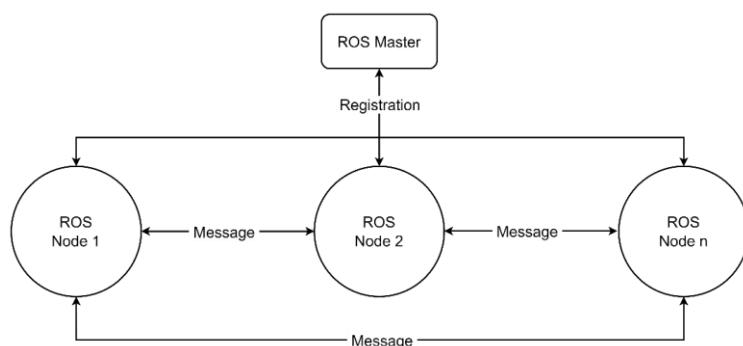
กระบวนการหรือรูปแบบที่ใช้ในการดำเนินการโปรแกรมต่างๆ ซึ่ง ROS จะถูกออกแบบให้เป็นระบบแยกส่วนในระดับที่ละเอียด ความหมายคือ จะแบ่งทั้งระบบที่ใช้ในการควบคุมหุ่นยนต์ให้เป็นแต่ละส่วนย่อยๆ เช่น ใช้ 1 Nodes ในการควบคุมมอเตอร์ล้อ, ใช้ 2 Nodes ในการวางแผนเส้นทาง, ใช้ 1 Nodes ในการรับค่าจาก Sensor ต่างๆ เป็นต้น โดย ROS Nodes จะถูกสร้างขึ้นด้วยการเขียนโปรแกรมในภาษาต่างๆผ่าน Library ของ ROS เช่น roscpp คือ การเขียน ROS Nodes ด้วยภาษา C++ หรือ rospy คือ การเขียน ROS Nodes ด้วยภาษา Python เป็นต้น ซึ่งสามารถใช้ภาษาอื่นในการเขียน ROS Nodes ได้เช่นกัน



รูปที่ 2.82 ตัวอย่าง Nodes ในการควบคุมหุ่นยนต์

### 2.7.1.2 ROS Master

ส่วนที่มีหน้าที่จัดเตรียมการ Register ชื่อ Nodes และค้นหาองค์ประกอบต่างๆใน ROS และหากไม่มี ROS Master แต่ละ Nodes ก็จะไม่สามารถค้นหาซึ่งกันและกัน เพื่อแลกเปลี่ยนข้อมูลหรือเรียกใช้



รูปที่ 2.83 ภาพอธิบายความสัมพันธ์ระหว่าง ROS Master และ ROS Nodes

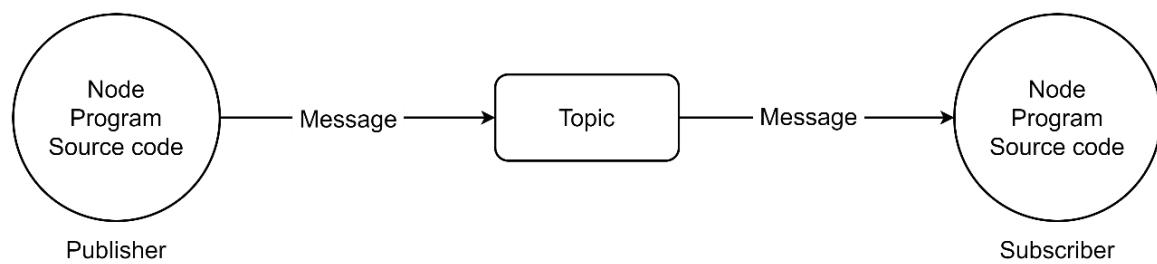
### 2.7.1.3 Parameter Server

Server กลางที่ใช้ออนุญาตในการเก็บข้อมูลโดยใช้ตำแหน่งส่วนกลาง ซึ่งในปัจจุบัน Parameter Server นั้น เป็นส่วนหนึ่งของ ROS Master

**2.7.1.4 ROS Message** ข้อความหรือตัวข้อมูลที่ Nodes ต่างๆ ใช้สื่อสารซึ่งกันและกัน โดยตัว Message นั้น จะเป็นโครงสร้างข้อมูลแบบค่ามาตรฐาน (std\_msgs) เช่น Integer, String, Float, Boolean, Array เป็นต้น อีกทั้งยังมี Message แบบพิเศษอื่นๆ ที่ใช้ในการส่งข้อมูลที่เฉพาะเจาะจง เช่น Message ที่ใช้ในการควบคุม การเคลื่อนที่ (nav\_msgs) เป็นต้น

### 2.7.1.5 ROS Topic

ชื่อหรือหัวข้อที่ใช้ในการระบุตัวตนและเนื้อหาที่เก็บ Message นั้นๆ ซึ่ง Message จะถูกส่งผ่านระบบการ ขนส่งที่มีการ Publish และ Subscribe โดย Nodes ที่ทำการส่ง Message ในหัวชื่อ Topic นั้นๆ จะเรียกว่า “Publisher” และ Nodes ที่ต้องการรับ Message ในหัวชื่อ Topic นั้นๆ จะเรียกว่า “Subscriber” โดยแต่ละ Topic ของแต่ละ Nodes สามารถกระจายการรับและส่งไปยังอีกหลายๆ Nodes ได้ ความหมายคือ Nodes หนึ่งๆ สามารถกระทำตัวเป็นหลาย Publisher หรือ Subscriber ก็ได้ ซึ่งในทั่วไปแล้ว Publisher และ Subscriber จะไม่ทราบถึงการมีอยู่ของกันและกัน หรือก็คือแยกกันผลิตหรือรับข้อมูลตามหน้าที่ของตน นั่นเอง



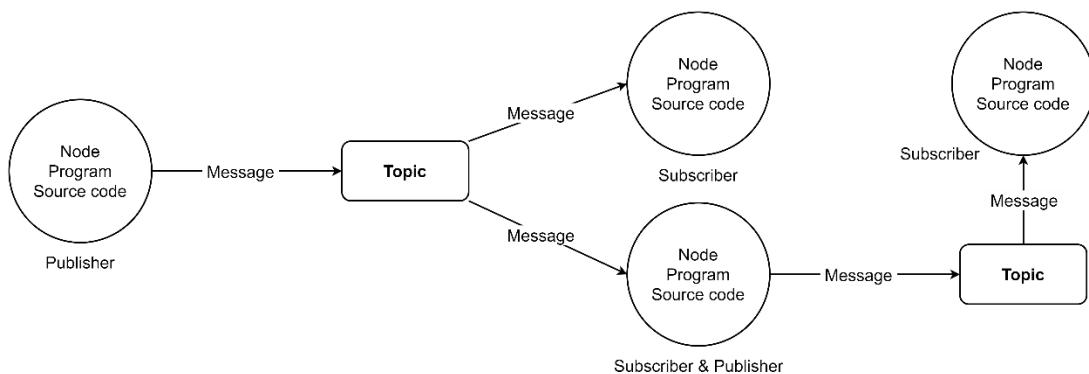
รูปที่ 2.84 ภาพอธิบายองค์ประกอบที่ใช้ในการสื่อสารระหว่าง Nodes

### 2.7.1.6 ROS Bags

รูปแบบสำหรับ Recording และ Playing Back ROS Message Data ซึ่ง ROS Bags นั้นเป็นกลไกที่สำหรับสำหรับจัดเก็บข้อมูล เช่น Sensing Data, หรือ Message ที่สำคัญต่างๆ เป็นต้น

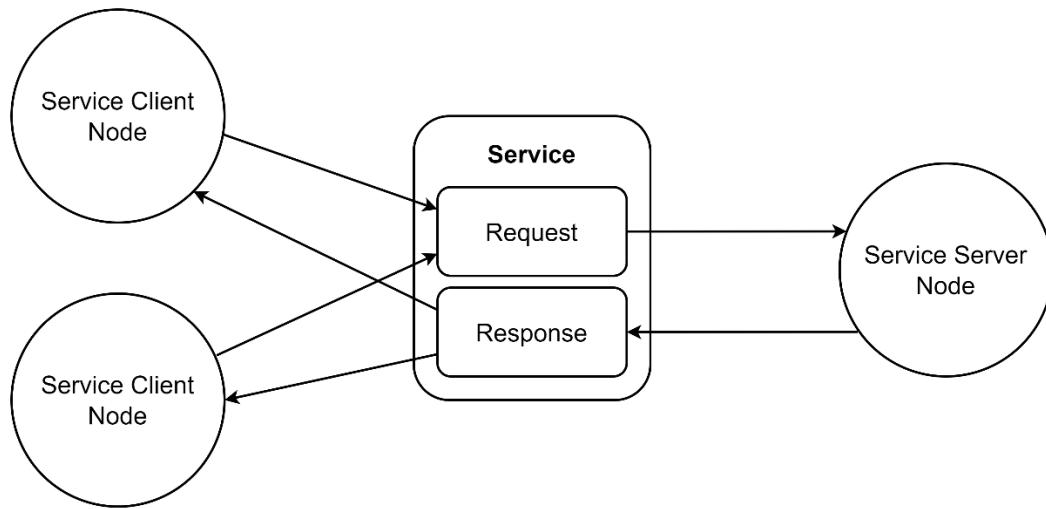
สำหรับรูปแบบของการสื่อสารหรือรูปแบบการ Messaging ในระบบ ROS สามารถแบ่งได้ 3 รูปแบบ คือ

- ROS Steam คือ การส่ง Message ระหว่าง Nodes ต่างๆ ใน ROS ซึ่งลักษณะการส่งจะเป็นการส่งข้อมูลแบบต่อเนื่องผ่าน ROS Topic โดย Nodes ที่ส่ง Message (Publisher) ก็จะส่งข้อมูลออกไปอย่างต่อเนื่อง โดยไม่สนใจว่าจะมีคนรับหรือไม่ และเช่นเดียวกับ Nodes ที่รับ Message (Subscriber) ก็จะรอฟัง Message อย่างเดียว หากนั้นมีอะไรรับข้อมูลเมื่อไหร่ ก็จะใช้งาน Receiver Callback Function



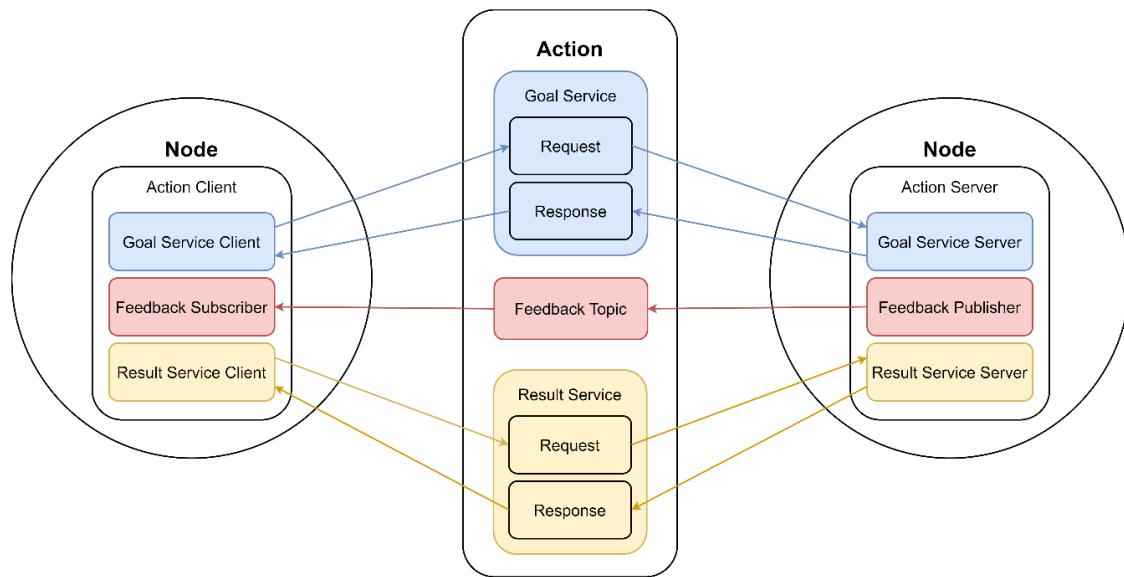
รูปที่ 2.85 ภาพอธิบายการสื่อสารข้อมูลแบบ ROS Steam

- ROS Service คือ รูปแบบแนวคิดการส่ง Message แบบ Publish และ Subscribe อีกรูปแบบหนึ่งที่พัฒนามาจากการส่ง Message แบบ Steam ซึ่งจะแตกต่างกันที่ ROS Steam จะเป็นการส่ง Message แบบทางเดียว (Publisher to Subscriber) แต่รูปแบบการสื่อสารแบบ Service จะใช้การสื่อสารแบบ Request and Response โดย Nodes ที่ทำการ Request จะเรียกว่า “Client” และ Nodes ที่ทำการ Response จะเรียกว่า “Server” ซึ่งการ Messaging รูปแบบนี้จะไม่เน้นความต่อเนื่อง แต่จะเน้นการรับส่งข้อมูลคร่าวๆ ครั้ง เช่น การส่ง Message ไปเพื่อเปิดไฟ กรณีถ้าเป็นแบบ Steam จะส่งคำสั่งขอเปิดไฟต่อเนื่อง ซึ่ง Publisher ไม่สามารถทราบได้ว่าไฟเปิดไปแล้วหรือไม่ แต่ถ้าเป็นแบบ Service ตัว Client Nodes จะทำการ Request ขอเปิดไฟเพียงแค่ครั้งเดียว หากไฟติดแล้ว Server Nodes ก็จะ Return Success หรือ Failed กลับมา แต่จะมีข้อจำกัดคือ ระหว่างที่ Client Nodes ทำการ Request ต้องหยุดการทำงานทั้งหมดเพื่อรับการ Response จาก Server Nodes เสมอและระหว่างที่ทำการรอ Response จะไม่สามารถสื่อสารกับ Nodes อื่นๆ ได้อีก แต่ถ้ามีระยะเวลา Timeout ในการรอ Response เช่นกัน



รูปที่ 2.86 ภาพอธิบายการสื่อสารข้อมูลแบบ ROS Service

3. ROS Action Library คือ รูปแบบแนวคิดการส่ง Message ที่พัฒนามาจาก ROS Service สำหรับ Task ที่ใช้เวลาประมวลผลนาน และ Client Nodes ไม่ต้องการที่หยุดการทำงานของตนเพื่อรอการ Response โดยหลักการคือ Client Nodes จะส่ง Request Goal ไปยัง Server Nodes ปลายทางและทำงานในส่วนของตนต่อ และ Server Nodes เมื่อได้รับ Request Goal มาแล้ว ก็จะทำงานตามคำสั่งที่ตนได้รับมาให้สำเร็จตามที่ตั้งไว้ โดยจะสามารถส่งข้อมูลความคืบหน้าหรือ Feedback Progress ระหว่างช่วงต่างๆ กว่าจะถึง Goal ที่รับมาได้ และเมื่อดำเนินการถึง Goal ที่ตั้งไว้ก็จะสามารถรายงานผลสุดท้ายได้ว่า Success หรือ Failed ตัวอย่างเช่น Client Nodes ทำการ Request Goal ไปยัง Server Nodes ว่าให้ทำการนับเลขตั้งแต่ 0 ไปจนถึง 1 แสน หลังจาก Server Nodes ทำการรับ Request Goal มาแล้วก็จะเริ่มนับ และสามารถรายงานผลความคืบหน้าได้ว่า ณ ตอนนี้นับเลขถึงใดแล้ว เช่น เมื่อนับถึง 1000 ก็จะ Feedback Progress 1 ครั้ง และเมื่อนับถึง 10000 ก็จะ Feedback Progress กลับไปอีก 1 ครั้ง เป็นต้น และสุดท้ายเมื่อนับถึง 1 แสนแล้วก็จะรายงานผลสุดท้ายว่า Success



รูปที่ 2.87 ภาพอธิบายการสื่อสารข้อมูลแบบ ROS Action

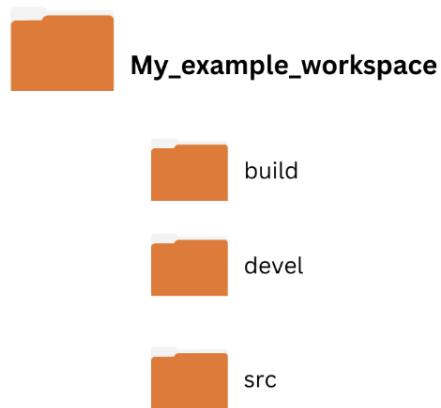
โดยจะสามารถสรุป Concept ของ ROS Computation Graph Level ได้ว่า ROS จะแบ่งการทำงานของหุ่นยนต์ออกเป็น Nodes ต่างๆ ซึ่ง Nodes ที่ว่านี้ก็คือ Source Code หรือ Program ต่างๆ ไม่ว่าจะเป็น Code ที่ใช้ในการรับค่าจาก Sensor ต่างๆ หรือ Code ที่ใช้ในการประมวลข้อมูล หรือ Code ที่ใช้ในการควบคุมการทำงานของมอเตอร์ รวมไปถึงโปรแกรมที่ใช้จำลองการทำงานของหุ่นยนต์ต่างๆ เป็นต้น และเมื่อแบ่งการทำงานของหุ่นยนต์ออกมาเป็นแต่ละ Nodes แล้ว ก็จะใช้การ Messaging ในรูปแบบต่างๆ เพื่อให้แต่ละ Nodes สามารถส่งข้อมูลหากันได้

## 2.7.2.ROS Filesystem Level

Concept ของระบบ ROS ที่จะจัดการหรือจัดเก็บไฟล์ต่างๆ ซึ่งไฟล์ที่ว่าจะเป็นโครงสร้างในการเรียกใช้งานของแต่ Nodes โดยจะมีส่วนประกอบที่สำคัญ ได้แก่ My Catkin Workspace, Package, Metapackage, Package Manifest, Repositories, Message(msg) types, Service(srv) types เป็นต้น

### 2.7.2.1 My Catkin Workspace หรือ Workspace

ไฟล์เดอร์ใหญ่ที่เก็บไฟล์เดอร์ส่วนต่างๆ ของ ROS ซึ่งก็คือ Environment ส่วนต่างๆ รวมไปถึง Source Code (Package หรือ Nodes) ที่ผู้ใช้งานพิมพ์ขึ้น โดยใน Workspace จะมีไฟล์เดอร์ย่อย 3 ไฟล์เดอร์ ดังนี้



รูปที่ 2.88 โฟลเดอร์ย่อย 3 โฟลเดอร์ใน Workspace

#### 2.7.2.2 build

โฟลเดอร์ที่ใช้ในการเก็บ Source Code ที่จะ Build หรือ Compile แล้ว จาก ROS โดยเวลาใช้งานจริงจะมีตัวที่เข้ามาเรียก Source Code ในโฟลเดอร์นี้ให้อัตโนมัติ ซึ่งไม่เป็นต้องเข้ามา Run ในโฟลเดอร์นี้เอง

#### 2.7.2.3 devel

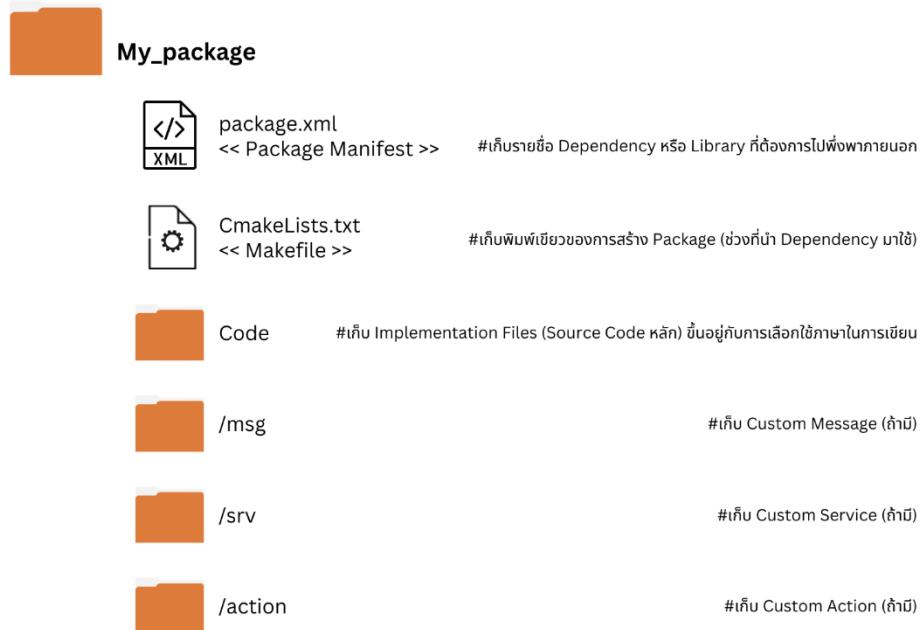
โฟลเดอร์ที่ใช้ในการเก็บ Source Code ที่ปั้งบอกว่ามีคือ Workspace ของเราและ Source Code ที่ใช้ในการเชื่อมต่อ Package ที่เราเขียนกับ Package ต่างๆ ภายนอกเข้าด้วยกัน รวมไปถึงของที่ ROS generate ให้ เช่น Custom Message

#### 2.7.2.4 src

โฟลเดอร์ที่ใช้ในการเก็บ Source Code ของ Package file ต่างๆ ซึ่งเป็นโฟลเดอร์ที่ควรสนใจมากที่สุด สำหรับการพัฒนาหุ่นยนต์

#### 2.7.2.5 Package

โฟลเดอร์ที่อยู่ภายในโฟลเดอร์ src และเป็นโฟลเดอร์หลักที่ใช้ในการบริหารจัดการและจัดระเบียบซอฟแวร์ต่างๆ ใน ROS ซึ่งอาจประกอบไปด้วย ROS Nodes, ROS dependent Library, Dataset, Configuration files และโปรแกรมอื่นที่จำเป็นสำหรับการจัดการและใช้งาน โดยใน 1 Package Folder จะประกอบไปด้วยไฟล์และโฟลเดอร์ย่อยต่างๆ ดังนี้



รูปที่ 2.89 ตัวอย่างส่วนประกอบต่างๆที่สำคัญภายใน Package

#### 2.7.2.6 Metapackage

Package ที่รวม Package อื่นๆ ที่มีหน้าที่โกลเด็คียงกันมาไว้ใน Package เพื่อการใช้งานง่าย ตัวอย่าง Navigation Metapackage ประกอบไปด้วย 10 Package เช่น AMCL(Partial Filter), DWA, EKF(Extend Kalman Filter) และ Map\_Server เป็นต้น

#### 2.7.2.7 Package Manifest หรือ Package.xml

ไฟล์ที่จะบ่งบอกว่า Package นี้รายละเอียดเป็นอย่างไร ต้องการไปพิ่งพา Source Code ภายนอกหรือไม่ เช่น ตัวช่วย Compile Code หรือ Reference Code เป็นต้น

#### 2.7.2.8 Makefile หรือ Cmakelist.txt

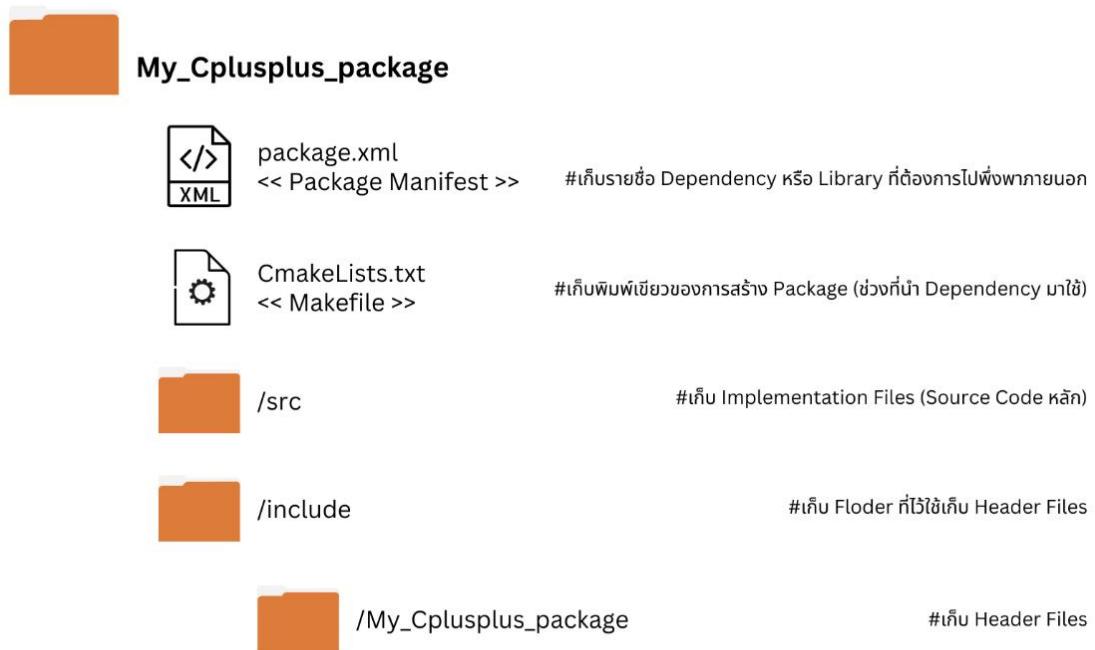
ไฟล์ที่มีความสำคัญมาก โดยลักษณะจะคล้ายกับพิมพ์เขียวของ Package นี้ ซึ่งจะเก็บรายละเอียดต่างๆ เช่น ต้องการ Run Code ใดเป็นจุดเริ่มต้นของงาน, Code ต่างๆ ที่ต้องอิงกันอย่างไร, ใช้ Code อื่นภายนอกมา อ้างอิงกับ Code ภายในไฟล์ได้บ้าง เป็นต้น

### 2.7.2.9 Source Code

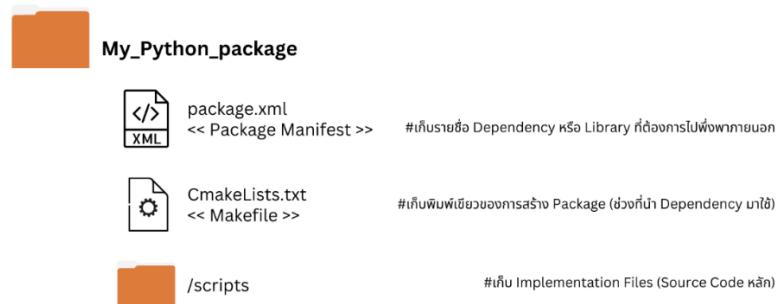
Code หรือ Nodes ของเรามาโดยชื่อของไฟล์เดอร์ส่วนนี้จะแตกต่างกันไปตามภาษาที่ใช้ในการสร้าง Nodes เช่น หากใช้ภาษา C++ จะมีชื่อว่า “/src” และมีไฟล์เดอร์ที่ใช้ในการเก็บ Header File ว่า “/include” และกรณีที่ใช้ภาษา Python จะมีชื่อว่า “/scripts”

### 2.7.2.10 Custom Folder

ไฟล์เดอร์ที่ใช้สร้าง Custom Message, Custom Service, และ Custom Action ของ ROS หรือสร้าง Message Structure, Message Prototype กรณีที่ผู้พัฒนาสร้างโครงสร้างของ Message ขึ้นมาเอง โดยไฟล์เดอร์ /msg จะอธิบายรูปแบบโครงสร้างของ Message ที่แต่ละ Nodes ที่อยู่ใน Package นั้นๆ ที่ใช้สื่อสารกัน, ไฟล์เดอร์ “/srv” จะอธิบายรูปแบบโครงสร้างการ Request/Response ของแต่ละ Nodes และไฟล์เดอร์ “/action” จะอธิบายรูปแบบโครงสร้างการ Action ของแต่ละ Nodes โดยหาก Compile ระบบ Catkin build system จะให้ ROS มาดึงโครงสร้างของ Message, Service หรือ Action จากบริเวณนี้แบบอัตโนมัติ

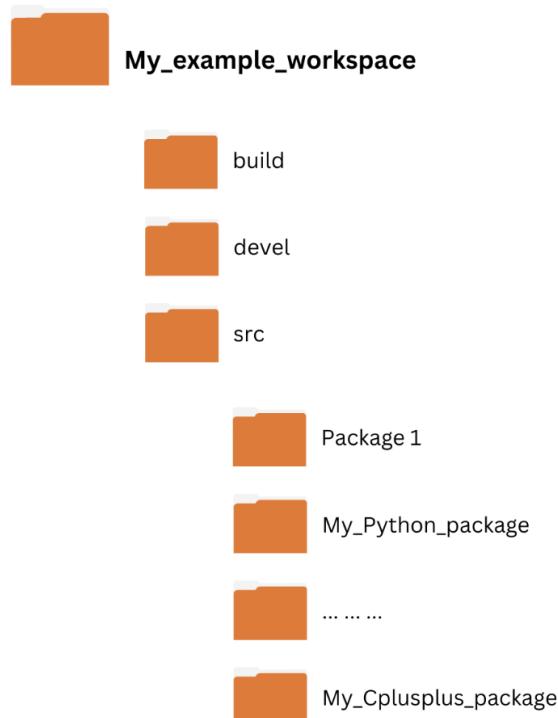


รูปที่ 2.90 ตัวอย่างส่วนประกอบต่างๆที่สำคัญภายใน Package กรณีใช้ภาษา C++



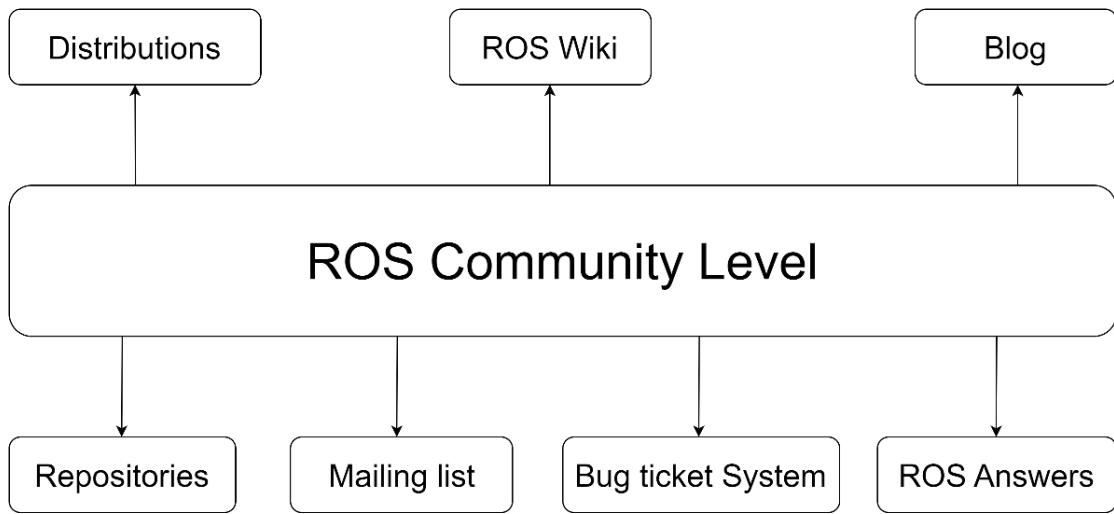
รูปที่ 2.91 ตัวอย่างส่วนประกอบต่างๆ ที่สำคัญภายใน Package กรณีใช้ภาษา Python

โดยจะสามารถสรุป Concept ของ ROS filesystem level ได้ว่าในการสร้าง ROS จะเป็นต้องมีระบบจัดการไฟล์ที่เป็นรูปแบบพื้นฐานของ ROS โดยไฟล์เดอร์ใหญ่ที่สุดจะเรียกว่า “Workspace” จะเก็บข้อมูลทั้งหมดของงาน และภายใน Workspace เราสามารถสร้าง Package ซึ่งก็คือส่วนที่ใช้ในการรวมและจัดระเบียบซอฟต์แวร์, Nodes หรือ Codes ต่างๆ ภายใน Package นั้น ซึ่งใน 1 Workspace สามารถมีหลาย Package ได้ และใน 1 Package ก็สามารถมีหลาย Nodes ได้ เช่นกัน โดย 1 Nodes มักจะทำหน้าที่ 1 หน้าที่ ขึ้นไปขึ้นอยู่กับผู้พัฒนาว่าอยากให้ Nodes นั้นๆ มีหน้าที่อย่างไร



รูปที่ 2.92 ตัวอย่างการเรียงลำดับไฟล์เดอร์ภายใน Workspace

### 2.7.3. ROS Community Level



รูปที่ 2.93 แหล่งข้อมูลต่างๆของ ROS Community Level

Concept ของระบบ ROS ที่จะเป็นแหล่งที่ใช้ในการศึกษา และเปลี่ยนความรู้ ด้านห้อง Packages ต่างๆ มาใช้งาน อีกทั้งยังใช้เป็นแหล่งเผยแพร่ Packages ที่เราออกแบบ

## บทที่ 3 การออกแบบอาร์ดแวร์และซอฟต์แวร์

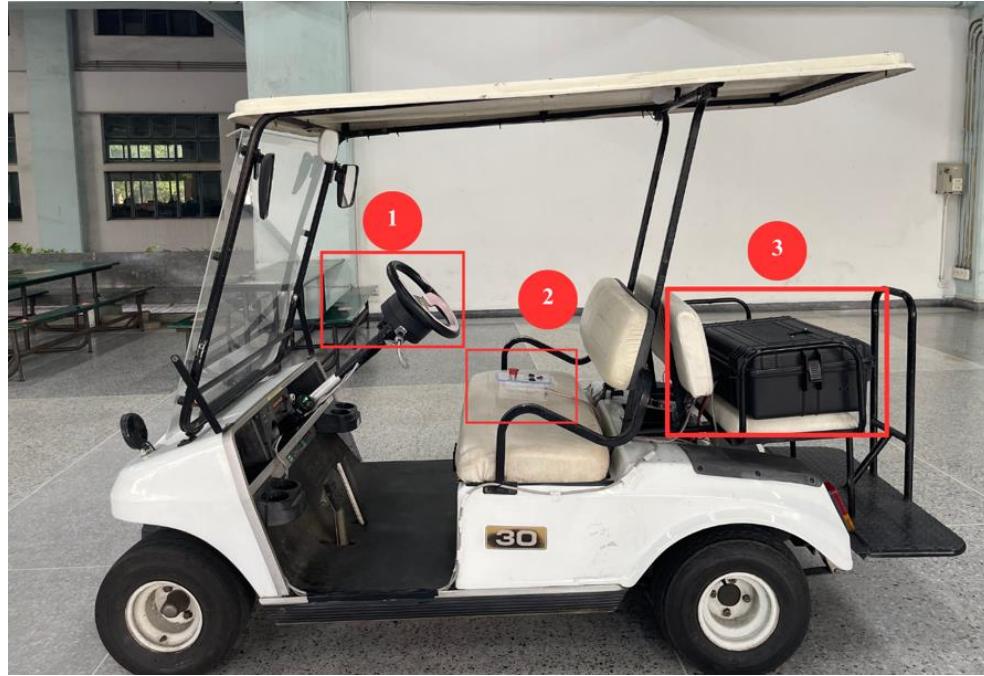
### 3.1 รายละเอียดการออกแบบอาร์ดแวร์

ในส่วนของการออกแบบอาร์ดแวร์สำหรับรถกล้องจะมีส่วนที่เป็นการออกแบบชิ้นส่วนสำหรับการติดตั้งมอเตอร์ การเชื่อมต่อสัญญาณผ่านตัวรับส่งสัญญาณวิทยุ และตัวรับค่า GNSS ด้วยโมดูล U-block F9P หรือ การออกแบบวงจรไฟฟ้าภายใน เช่น วงจรเพื่อช่วยในการป้องกันขณะตัวควบคุมพ่วงมาลัยมีปัญหา หรือมีการควบคุมที่ผิดปกติ (Safety Circuit)



รูปที่ 3.1 แสดงโครงสร้างภาพรวมของรถ

### 3.1.1 รายละเอียดอุปกรณ์ภายใน



รูปที่ 3.2 แสดงโครงสร้างของรถกอล์ฟจากด้านข้าง และการติดตั้งอุปกรณ์



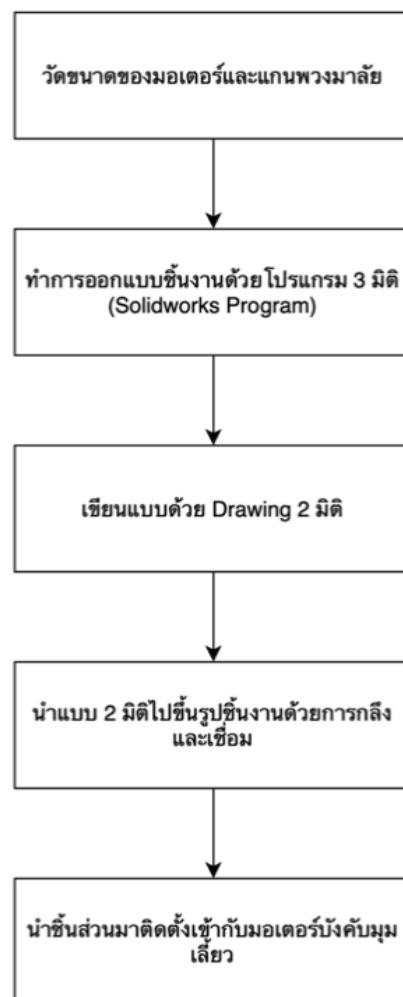
รูปที่ 3.3 แสดงโครงสร้างของรถกอล์ฟจากด้านหน้า และการติดตั้งอุปกรณ์

### 3.1.1.1 ส่วนที่ 1

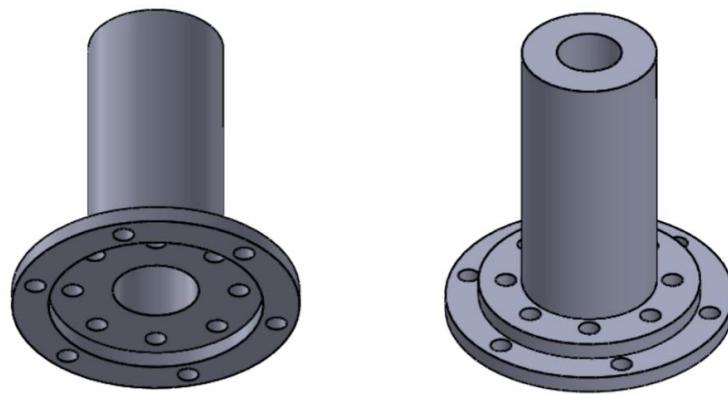
Steering motor เป็นอุปกรณ์ที่ควบคุมการทำงานเกี่ยวกับการควบคุมมุมเลี้ยว หรือควบคุมการทำงานของพวงมาลัย

#### การออกแบบชิ้นงานอุปกรณ์สำหรับติดตั้งมอเตอร์พวงมาลัย

ในส่วนของการออกแบบชิ้นงานอุปกรณ์เพื่อใช้ในการติดตั้งมอเตอร์พวงมาลัยเพื่อควบคุมมุมเลี้ยวบนนี้ ต้องทำการออกแบบใหม่จากชิ้นส่วนของงานวิจัยเดิม เนื่องจากขนาดและสัดส่วนเดิมนั้นไม่สามารถติดตั้งระหว่างมอเตอร์เข้ากับพวงมาลัยของรถก่อฟันใหม่ได้ จึงต้องสร้างชิ้นใหม่ โดยมีขั้นตอนและวิธีการสร้างชิ้นส่วนสำหรับติดตั้งมอเตอร์ขึ้น โดยทำการออกแบบแล้วนำแบบไปทำการกลึงโลหะ เพื่อให้ได้ชิ้นส่วนตามที่ต้องการในการติดตั้งร่วมกับพวงมาลัย Steering Motor และแกนพวงมาลัยดัง

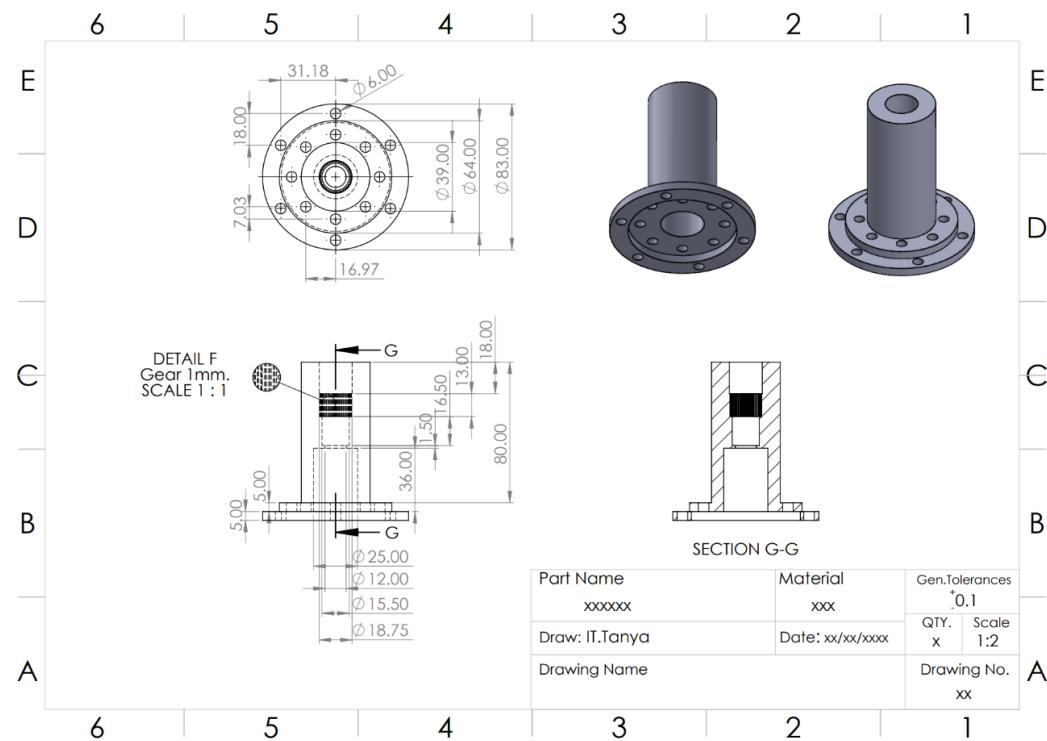


รูปที่ 3.4 แสดง Block Diagram ขั้นตอนการสร้างชิ้นงานอุปกรณ์สำหรับติดตั้งมอเตอร์พวงมาลัย



รูปที่ 3.5 แสดงตัวอย่างชิ้นงานอุปกรณ์ไฟล์ CAD จากการออกแบบใหม่สำหรับติดตั้งมอเตอร์พวงมาลัย

ชิ้นงานดังรูปที่ 3.5 นั้นออกแบบมาเพื่อทดแทนแกนพวงมาลัยเดิม ที่มีขนาดเส้นผ่านศูนย์กลางขนาดใหญ่กว่าที่ Steering Motor จะสามารถส่วนได้ไปได้พอดี โดยจะใช้ชิ้นงานนี้เป็นส่วนต่อขยายแกนพวงมาลัยเดิม ออกแบบติดตั้งเข้ากับพวงมาลัยบังคับมุมเลี้ยวของรถ ซึ่งต้องทำการปรับแก้ขนาดและออกแบบใหม่



รูปที่ 3.6 ชิ้นงานอุปกรณ์ไฟล์ Drawing จากการออกแบบใหม่สำหรับติดตั้งมอเตอร์พวงมาลัย



รูปที่ 3.7 ชิ้นงาน 3D Print ขณะทดสอบต่อกับพวงมาลัยและรถ

เมื่อนำแบบงานสามมิติมาขึ้นรูป 3D Print เพื่อนำชิ้นส่วนที่ได้ออกแบบมาทดลองต่อเข้าระหว่างรถและพวงมาลัย โดยมีข้อผิดพลาดคือออกแบบขนาดเส้นผ่านศูนย์กลางคลาดเคลื่อนไป 1 มิลลิเมตร จึงทำให้ไม่สามารถสวมลงกับแกนของรถได้จึงทำการตัดไปให้สามารถสวมชิ้นส่วนงานลงไปได้ ซึ่งขนาดในส่วนอื่นสามารถสวมใส่กับพวงมาลัยได้ลงตัว



รูปที่ 3.8 ชิ้นงาน โลหะสำหรับใช้จริ่ง

จากการทดลองนำงาน 3D Print มาทดลองรวมใส่กับพวงมาลัยรถได้พอดีแล้วจึงนำไปขึ้นรูปโลหะ ซึ่งได้ดัดแปลงให้มีความง่ายและสะดวกในการหล่อขึ้นรูปโลหะโดยการ ให้รูใส่นี้อยู่ในระนาบเดียวกันนอกนั้นมีขนาดเช่นเดิมทุกประการ

### 3.1.1.2 ส่วนที่ 2

Switch Control Box เป็นอุปกรณ์ หรือวงจรที่ติดตั้งเพื่อเลือกโหมดการทำงาน ภายในวงจรประกอบด้วยสวิตช์ที่ควบคุมการทำงานของ การเลือกโหมด Auto/Manual ของความเร็ว และ ของ Steering motor นอกจากนี้ยังประกอบด้วย สวิตช์ Reset ค่า Value ของ Digital Potentiometer ที่เป็นอุปกรณ์ในการควบคุมความเร็ว และ สวิตช์ที่ควบคุมการทำงานของ Emergency switch ซึ่งเป็นวงจรป้องกันความผิดพลาดที่อาจส่งผลให้เกิดความรุนแรง รวมไปถึงป้องกันการเกิดความเสียหายของ Steering motor หากเกิดกรณีไฟฟ้าลัดวงจร โดย Emergency Switch จะทำหน้าที่ตัดกระแสไฟฟ้าที่เข้า Steering motor ทันที



รูปที่ 3.9 วงจรที่เป็นตัวควบคุมการเลือกการทำงาน (Switch Controller Box)

### 3.1.1.3 ส่วนที่ 3

กล่องดำ หรือ กล่องที่ภายในประกอบด้วยวงจรต่างๆที่ควบคุมการทำงานของรถกอล์ฟให้เคลื่อนที่ อัตโนมัติ เช่น วงจรควบคุมความเร็ว วงจรควบคุมการทำงานของแรงดัน การออกแบบวงจรเพื่อควบคุมการทำงานของรถกอล์ฟให้เคลื่อนที่อัตโนมัติ ประกอบด้วยอุปกรณ์ไฟฟ้าหลายส่วน และแต่ละส่วนความต้องการทางไฟฟ้าที่แตกต่างกัน ทำให้ก่อนนำอุปกรณ์แต่ละตัวมาต่อ ต้องมีอุปกรณ์หรือวงจรเพื่อมาปรับแต่งแรงดันให้เข้ากับความต้องการแรงดันของอุปกรณ์



รูปที่ 3.10 วงจรควบคุมการทำงานของระบบ

นอกจากนี้น้ำยาในยังประกอบด้วยตัวประมวลผลหลักของการควบคุมการทำงานรถกอล์ฟ ซึ่งในการควบคุมการทำงานใช้ Jetson Xavier เป็นตัวประมวลผลหลักในการควบคุมการทำงาน ภายในประกอบด้วย ROS Software ซึ่งเป็นตัวประมวลผลกลางของทุกระบบเพื่อใช้ในการติดต่อสื่อสารกันระหว่างระบบต่างๆภายใน โดยจะทำการรับส่งข้อมูลผ่านอุปกรณ์ Hub เป็นอุปกรณ์ต่อพ่วงของระบบต่างๆ



รูปที่ 3.11 แสดงอุปกรณ์ที่ใช้ประมวลผลหลัก



รูปที่ 3.12 แสดงศูนย์กลางการเชื่อมต่อ Hub



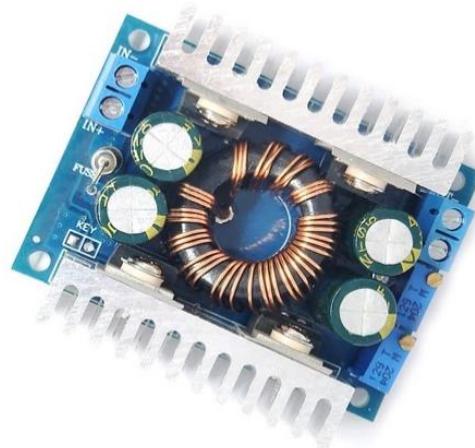
รูปที่ 3.13 แสดงหน้าจอที่ใช้ในการแสดงผล



รูปที่ 3.14 แสดงอุปกรณ์ที่ใช้ในการควบคุมการสั่งงานกับ Jetson Xavier



รูปที่ 3.15 แสดงอุปกรณ์ที่ใช้ในการควบคุมการสั่งงานกับ Jetson Xavier



รูปที่ 3.16 แสดงอุปกรณ์ปรับขนาดแรงดัน



รูปที่ 3.17 แสดงอุปกรณ์ที่ใช้กระจายแรงดันไฟยังอุปกรณ์ต่างๆ (Terminal Block Connect)

### 3.1.1.4 ส่วนที่ 4

GPS sensor (GNSS) เป็นอุปกรณ์ หรือ Sensor ที่ใช้ความคุณการทำงานของทุกรอบ ตั้งแต่การควบคุมความเร็ว การออกแบบเส้นทางของระบบการควบคุมมุ่งเลี้ยว เป็นต้น

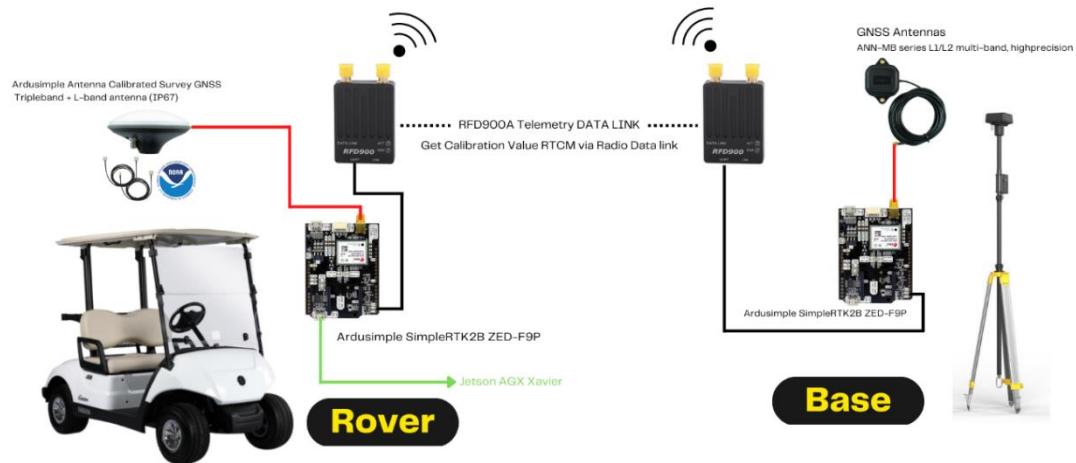


รูปที่ 3.18 แสดงตัวอย่างเสารับสัญญาณของ GPS sensor

## 3.2 การออกแบบฮาร์ดแวร์และซอฟแวร์ของระบบระบุตำแหน่ง

เนื่องจากการตรวจจับพิกัดและนำทางรถจำเป็นต้องอาศัยความถูกต้องในเชิงพิกัดที่สูง และเวลาในการส่งค่าพิกัดที่น้อย ดังนั้นงานรังวัดแบบจลน์ (Real Time Kinematic - RTK) และ งานรังวัดแบบ Network RTK จึงเหมาะสมในการนำมาประยุกต์ใช้ในการส่งค่าพิกัดมากกว่า เนื่องจากใช้เวลาในการรับ-ส่งข้อมูลน้อย และค่าความคลาดเคลื่อนที่อยู่ในระดับเซนติเมตร ซึ่งมีความละเอียดมากพอในการตรวจจับพิกัดและนำทางรถ โดยส่วนประกอบสำคัญในส่วนของตรวจจับพิกัดและนำทางรถจะประกอบไปด้วย 2 ส่วน ได้แก่ 1. ส่วนสถานี (Base Station) และ 2. ส่วนเคลื่อนที่ (Rover Station) ซึ่งเทคนิครังวัดทั้ง 2 วิธีจะแตกต่างกันที่การเลือกใช้ Base Station โดยหากเลือกใช้เทคนิครังวัดแบบ Real Time Kinematic – RTK จำเป็นต้องที่จะตั้งสถานี Base Station เอง แต่เทคนิครังวัดแบบ Network RTK ไม่จำเป็นต้องตั้งสถานี Base Station เอง โดยจะรับค่าปรับแก้ผ่านระบบอินเทอร์เน็ต ซึ่งทั้ง 2 เทคนิคให้ความละเอียดในเชิงพิกัดที่ใกล้เคียงกันในระดับเซนติเมตร แต่การส่งค่าปรับแก้จาก Base Station ด้วยเทคนิครังวัดแบบ Network RTK จะมีโอกาสสูญเสียสัญญาณได้จากเงื่อนไขทางสิ่งแวดล้อมในพื้นที่ที่ต้องการวัด เช่น มีสิ่งกีดขวางมาก มีอาคารขนาดใหญ่ ห้องฟ้ามีเมฆมาก เป็นต้น ซึ่งแต่ละเทคนิคจะมีส่วนประกอบย่อยของแต่ละเทคนิค ดังนี้

### 3.2.1. การออกแบบฮาร์ดแวร์การรังวัดแบบ Real Time Kinematic - RTK



รูปที่ 3.19 ส่วนประกอบต่างๆ ในระบบตรวจจับพิกัดและนำทางรถแบบ Real Time Kinematic – RTK

#### 1. ส่วนสถานี (Base Station) ประกอบด้วย 8 ส่วนย่อยดังนี้

- 1.1 เสาในการติดตั้ง มีหน้าที่ เป็นฐานที่ตั้งและเก็บอุปกรณ์อื่นๆ โดยเสาควรจะมีความสมดุลและแข็งแรงทนทานต่อสภาพอากาศในระดับหนึ่ง



รูปที่ 3.20 ตัวอย่างเสาของ Base Station

1.2 เสาอากาศ GNSS Antennas รุ่น ANN-MB series L1/L2 multi-band, high precision มีหน้าที่ ปรับรูปทรงและรวมคลื่นวิทยุให้ส่งไปยังทิศทางที่ต้อง



รูปที่ 3.21 GNSS Antennas รุ่น ANN-MB series L1/L2 multi-band, high precision

1.3 Receiver บอร์ด ArduSimple SimpleRTK2B ZED-F9P มีหน้าที่ รับสัญญาณจากดาวเทียม แล้วนำมาแปลงเป็นพิกัดของตำแหน่งที่ตัวเครื่องอยู่บนพื้นโลก



รูปที่ 3.22 บอร์ด ArduSimple SimpleRTK2B ZED-F9P

1.4 Radio Link RFD900A Telemetry DATA LINK มีหน้าที่ เป็นอุปกรณ์ที่ใช้ในการสื่อสารข้อมูลระดับเครือข่าย Data Link มีการทำงานจะเปรียบเสมือนผู้ตรวจสอบหรือควบคุมความผิดพลาดของข้อมูลที่จะส่งออกเป็น Package หรือ Frame กรณีหากผู้รับได้รับข้อมูลที่ถูกต้องแล้วก็จะส่งข้อมูลยืนยันกลับว่าได้รับข้อมูลแล้ว เรียกว่าสัญญาณ ACK (Acknowledge) ไปยังผู้ส่ง แต่ในกรณีที่ผู้ส่งไม่ได้รับสัญญาณ ACK หรือได้รับสัญญาณ NAK (Negative Acknowledge) กลับมา ผู้ส่งก็อาจจะทำการส่งข้อมูลไปใหม่ อีกหน้าที่หนึ่งของเลดเยอร์ชั้นนี้คือ ป้องกันไม่ให้เครื่องส่งทำการส่งข้อมูลเร็วจนเกินจีดความสามารถของเครื่องรับ ดังนี้จึงทำให้ Base Station และ Rover Station

สามารถสื่อสารระหว่างรับ-ส่งข้อมูลกันได้อย่างครบถ้วน ส่งผลให้การระบุตำแหน่งทำงานได้อย่างต่อเนื่องและลดความผิดพลาดในการสื่อสารลง ซึ่งรูปแบบในการสื่อสารยังคงใช้การสื่อสารแบบอนุกรมแบบอะซิงโกรันต์ส (Universal Asynchronous Receiver and Transmitter - UART)



รูปที่ 3.23 Radio Link RFD900A Telemetry DATA LINK

2. ส่วนเคลื่อนที่ (Rover Station) ประกอบด้วย 4 ส่วนย่อยดังนี้
  - 2.1 รถกอล์ฟไฟฟ้า มีหน้าที่ เป็นส่วนเคลื่อนที่ในการเก็บค่าพิกัด



รูปที่ 3.24 ตัวอย่างรถกอล์ฟไฟฟ้าของ Rover Station

2.2 เสาอากาศ Antenna ArduSimple Antenna Calibrated Survey GNSS Triple band + L-band antenna (IP67) มีหน้าที่ ปรับรูปร่างและรวมคลื่นวิทยุให้ส่งไปยังทิศทางที่ต้องการ



รูปที่ 3.25 Antenna ArduSimple Antenna Calibrated Survey GNSS Triple band + L-band antenna (IP67)

- 2.3 Receiver บอร์ด ArduSimple SimpleRTK2B ZED-F9P มีหน้าที่ รับสัญญาณจากดาวเทียม และนำมายแปลงเป็นพิกัดของตำแหน่งที่ตัวเครื่องอยู่บนพื้นโลก ดังรูปที่ 3.22
- 2.4 Radio Link RFD900A Telemetry DATA LINK มีหน้าที่ เป็นอุปกรณ์ที่ใช้ในการสื่อสารข้อมูลระดับเครือข่าย Data Link ดังรูปที่ 3.23

### 3.2.2. การออกแบบอาร์ดแวร์การรังวัดแบบ Network RTK



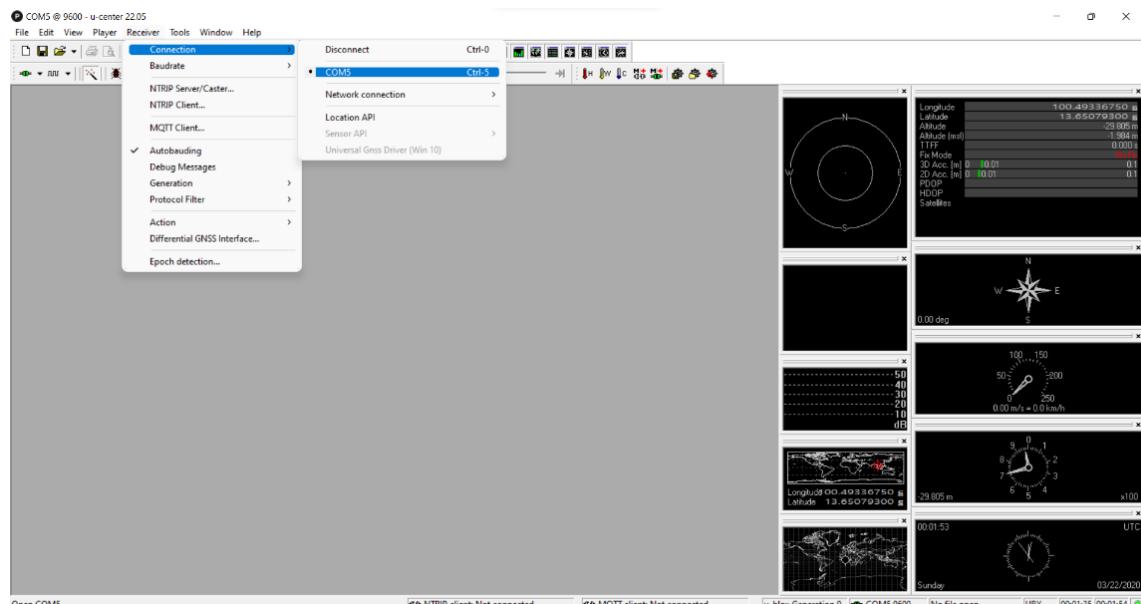
รูปที่ 3.26 ส่วนประกอบต่างๆ ในระบบตรวจจับพิกัดและนำทางรถแบบ Network RTK

## 1. ส่วนเคลื่อนที่ (Rover Station) ประกอบด้วย 4 ส่วนย่อยดังนี้

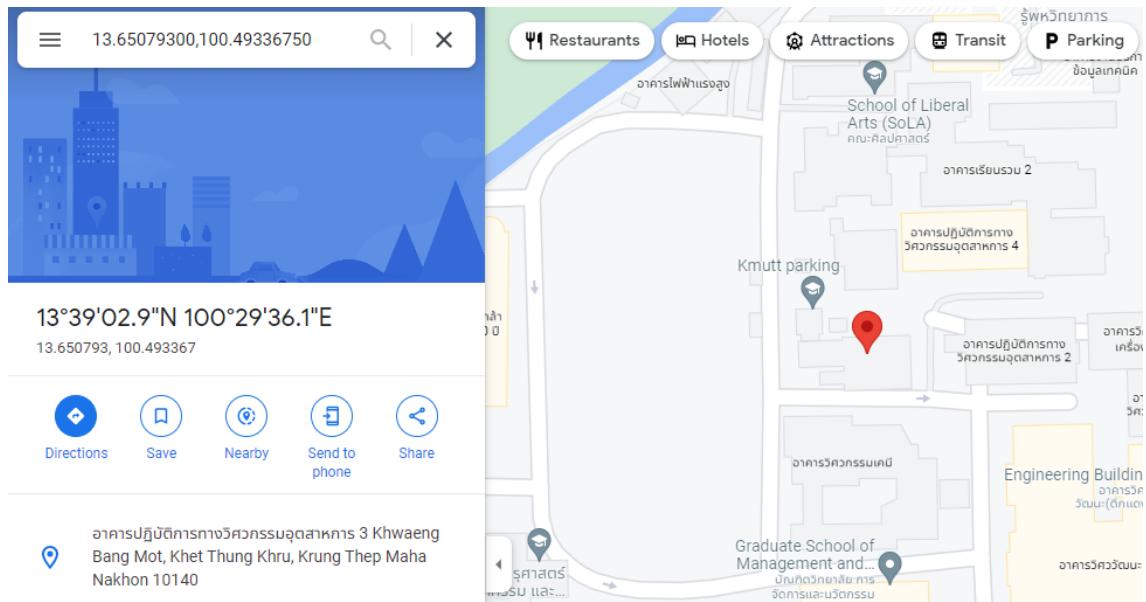
- 1.1 รถกลอสไฟฟ้า ดังรูปที่ 3.24 มีหน้าที่ เป็นส่วนเคลื่อนที่ในการเก็บค่าพิกัด
- 1.2 เสาอากาศ Antenna ArduSimple Antenna Calibrated Survey GNSS Triple band + L-band antenna (IP67) ดังรูปที่ 3.25 มีหน้าที่ ปรับรูปร่างและรวมคลื่นวิทยุให้ส่งไปยังทิศทางที่ต้องการ
- 1.3 Receiver บอร์ด ArduSimple SimpleRTK2B ZED-F9P ดังรูปที่ 3.22 มีหน้าที่ รับสัญญาณจากดาวเทียม และนำมาระบุตำแหน่งเป็นพิกัดของตำแหน่งที่ตัวเครื่องอยู่บนพื้นโลก

### 3.2.3 การออกแบบเฟร์เฟร์ที่เกี่ยวข้องในระบบตรวจจับพิกัดและนำทางระบบ GNSS RTK

**U-blox Ucenter1** คือ โปรแกรมที่ใช้ในการตั้งค่าการรับ-ส่งข้อมูลของ U-blox GNSS Module โดยหน้าที่หลักจะเป็นซอฟแวร์สำหรับการเรียกใช้ค่าพิกัดจาก GNSS Module หรือแสดงข้อมูลรายละเอียดต่างๆ เช่น จำนวนดาวเทียมระบุตำแหน่งที่เชื่อมต่ออยู่ด้วยในขณะนี้ การตั้งต่ำ Receiver Board ว่าต้องการใช้ในรูปแบบของ Base Station หรือ Rover Station เป็นต้น

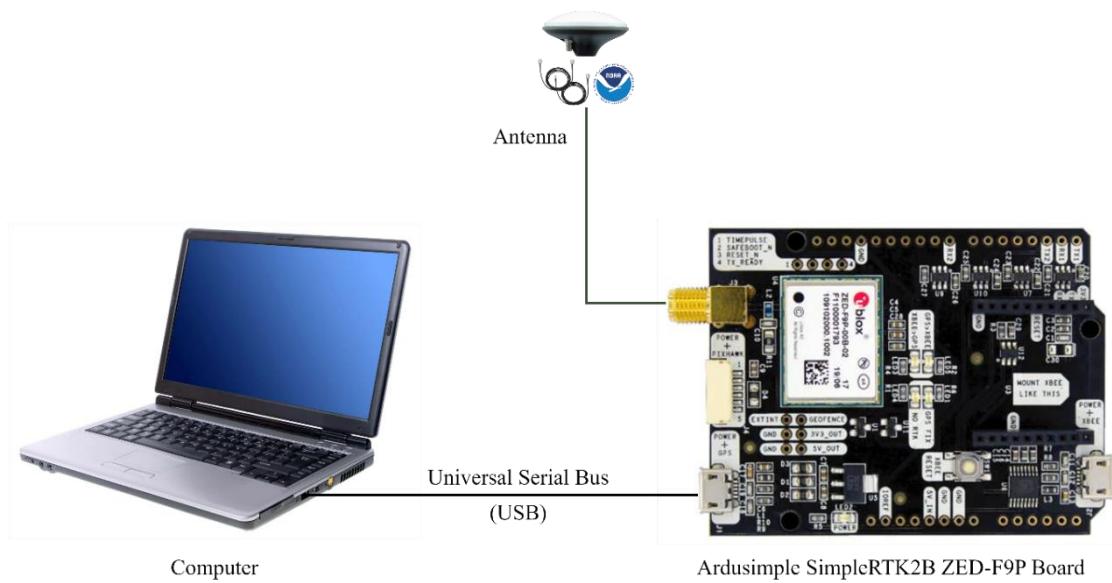


รูปที่ 3.27 ตัวอย่างการใช้งานโปรแกรม U-center เชื่อมต่อกับ GNSS Module



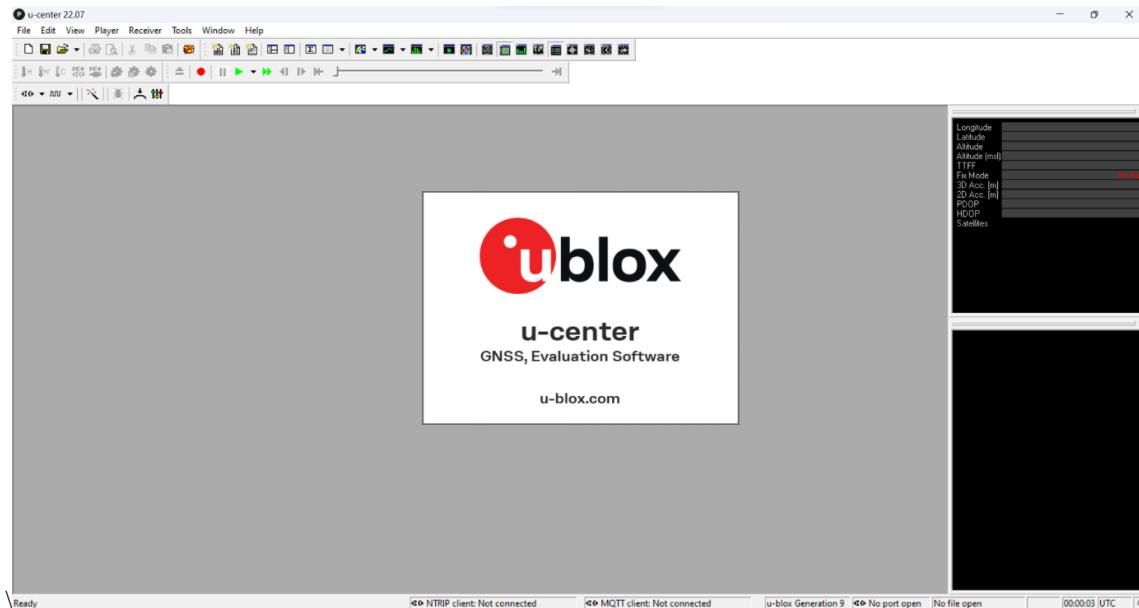
รูปที่ 3.28 การค้นหาพิกัดค่าละดิจิตและลงติดจุดที่ได้จากโปรแกรม U-center

### 3.2.3.1 ขั้นตอนการตั้งค่า Receiver Board ZED-F9P ผ่านโปรแกรม U-center

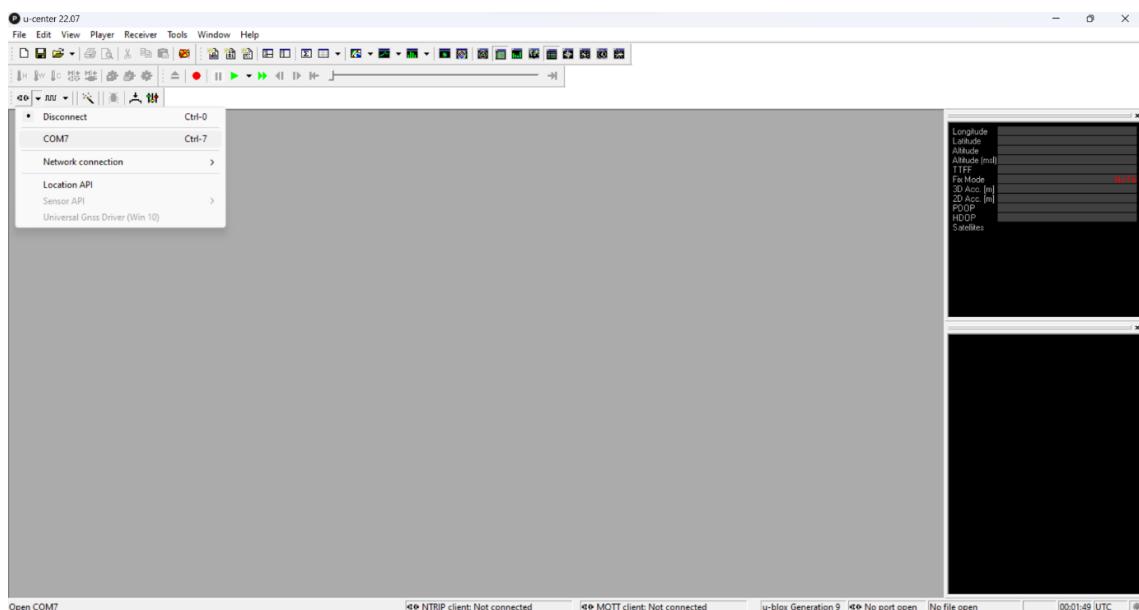


รูปที่ 3.29 การเชื่อมต่อ Board Receiver กับ Computer

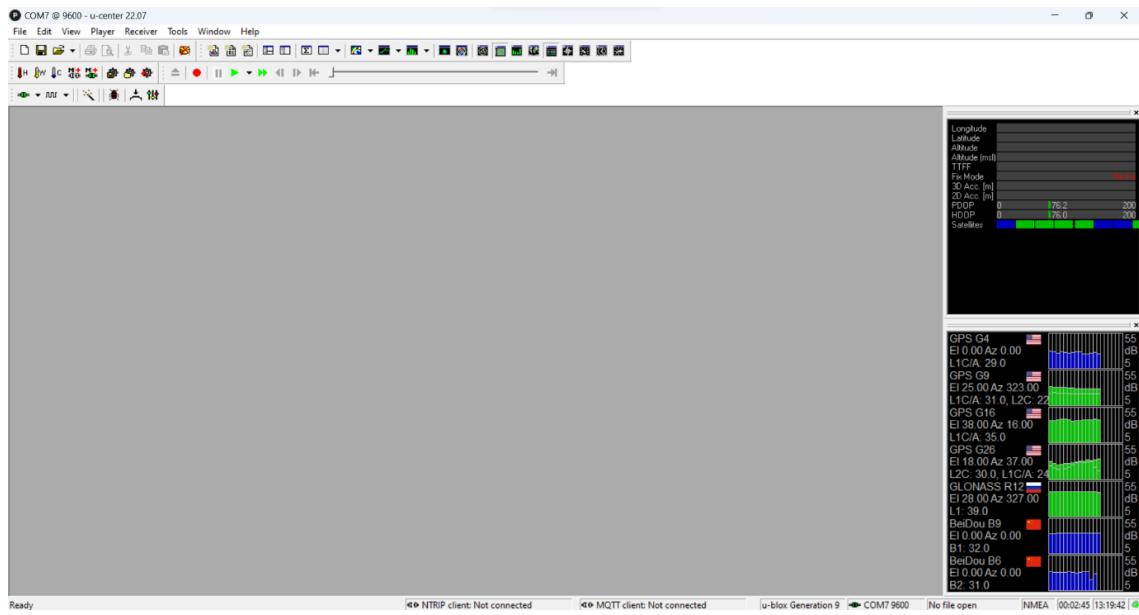
## 1. การตั้งค่า Base Station ในเทคนิคการรังวัดแบบ Real Time Kinematic



รูปที่ 3.30 หน้าโปรแกรมเริ่มต้นในการใช้งาน U-center

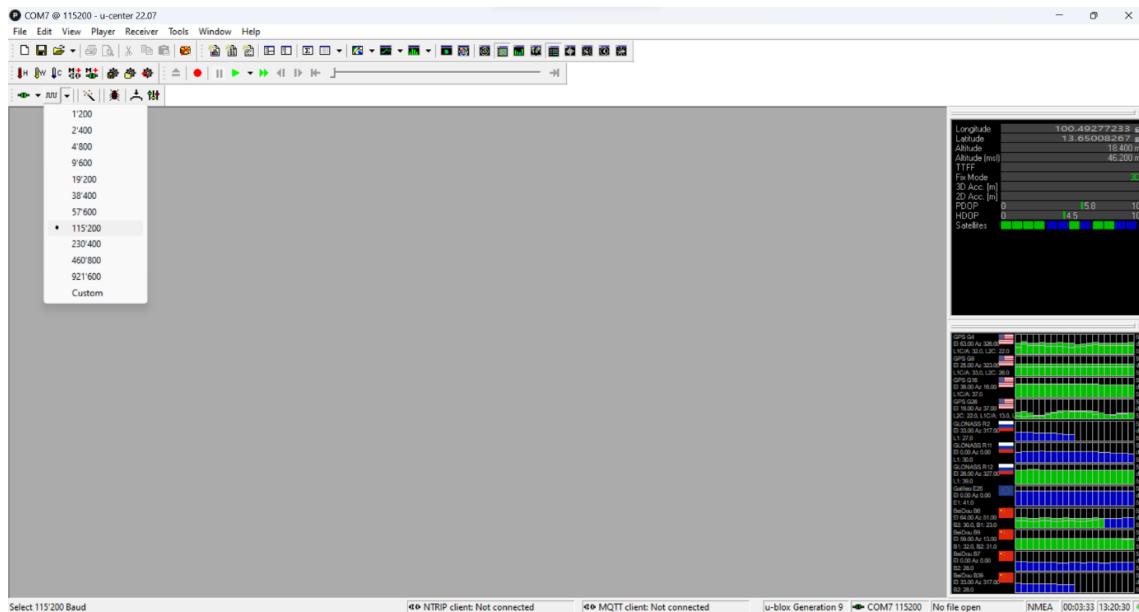


รูปที่ 3.31 การเลือก Com port ในการเชื่อมต่อ กับ Base Receiver

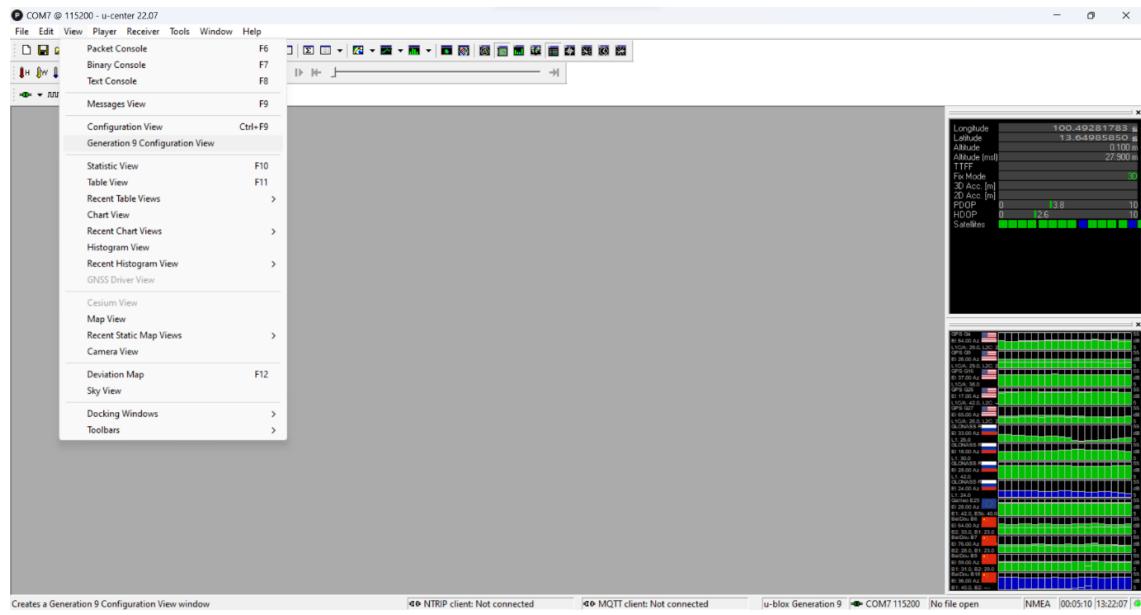


**รูปที่ 3.32** หน้าโปรแกรม U-center หลังจากเชื่อมต่อ กับ Base Receiver

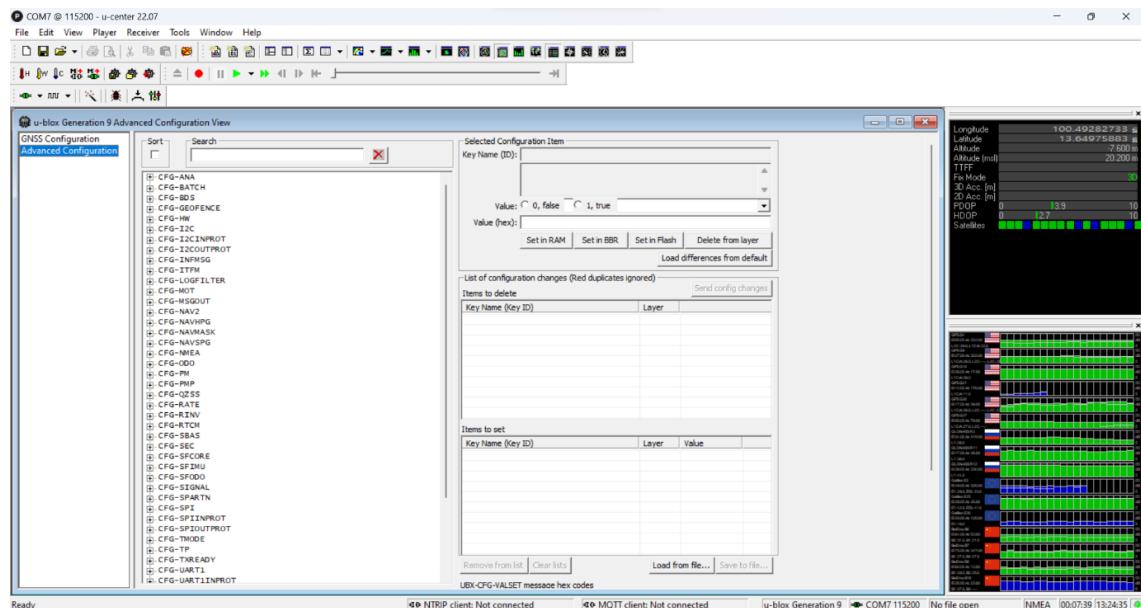
**หมายเหตุ :** การเลือกสถานที่การเชื่อมต่อที่สามารถรับคลื่นสัญญาณดาวเทียมได้ชัดเจน



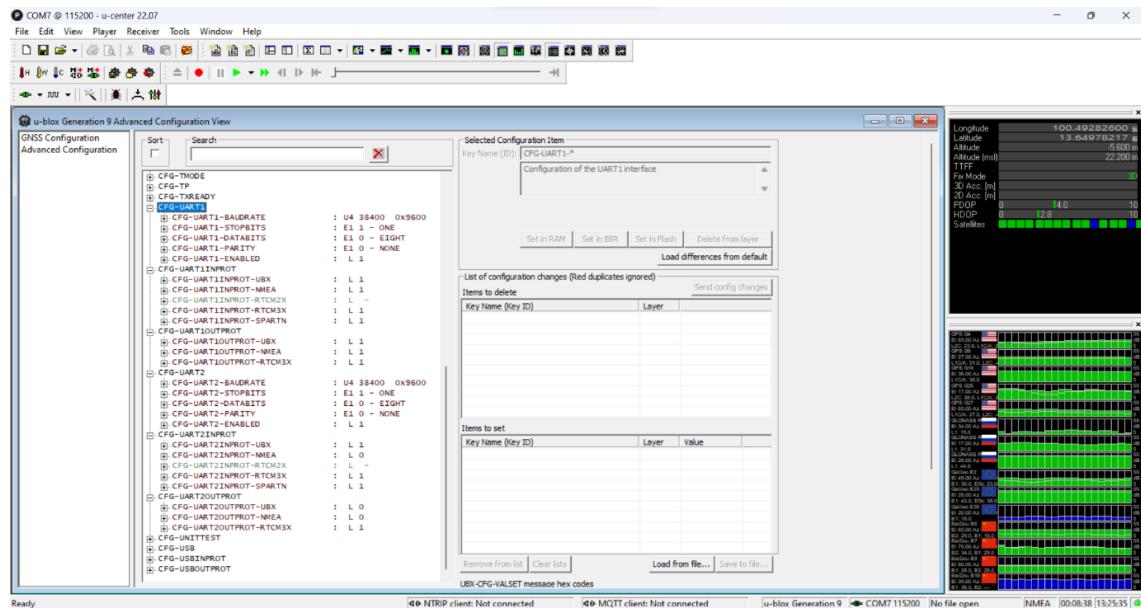
**รูปที่ 3.33** การเลือกใช้ Baud rate ที่มากที่สุดที่ Adrusimple RTK2B ZED-F9P รับได้



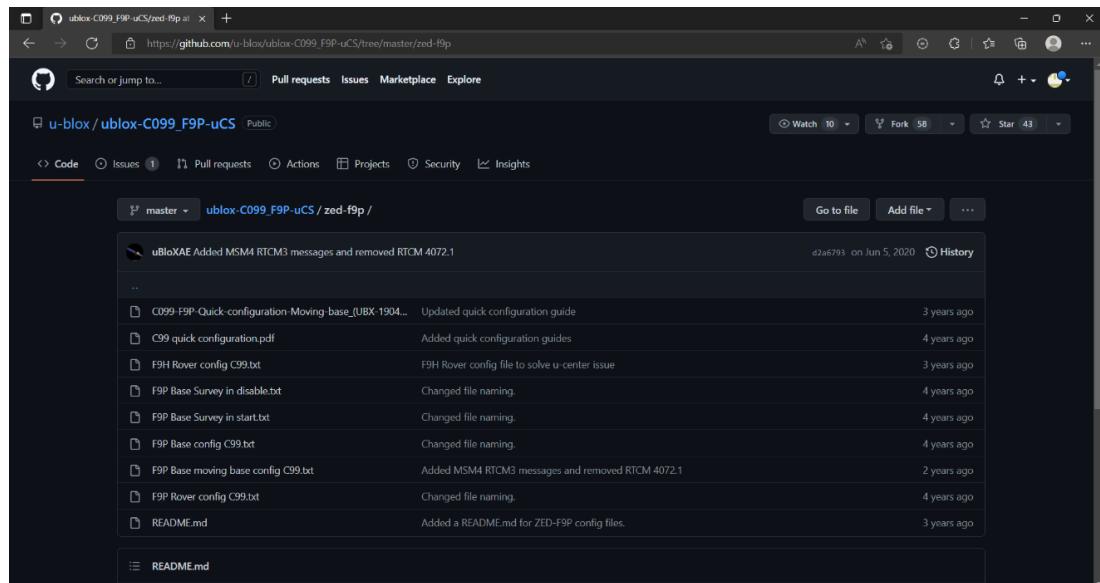
รูปที่ 3.34 เริ่มการตั้งค่าโดยการเปิดหน้าต่าง Generation Configuration View



รูปที่ 3.35 เลือกหัวข้อ Advanced Configuration

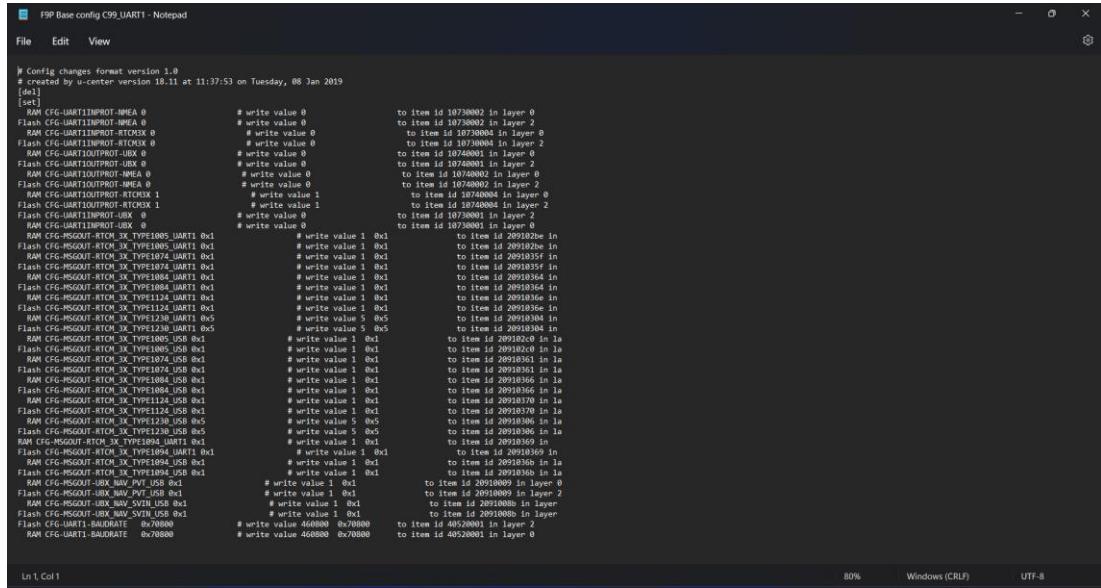


รูปที่ 3.36 เลือกหัวข้อการตั้งค่า Pin UART เพื่อตั้งค่าหัวข้อต่างๆ



รูปที่ 3.37 Website ของ U-blox สำหรับดาวน์โหลด Configuration File

สำหรับการตั้งค่า Receiver สามารถตั้งค่าได้ 2 แบบ คือ 1. การใช้ Configuration File และ 2. การตั้งค่าแบบ Manual ซึ่งการตั้งค่าทั้ง 2 แบบ ให้ผลลัพธ์ที่เหมือนกัน แต่ในการตั้งค่าการรับ-ส่งข้อมูล มีบางหัวข้อที่จำเป็นต้องตั้งค่าแบบ Manual เนื่องจากใน Configuration File ไม่มีการตั้งค่าในส่วนนี้ ซึ่งจะอธิบายในขั้นตอนถัดไป

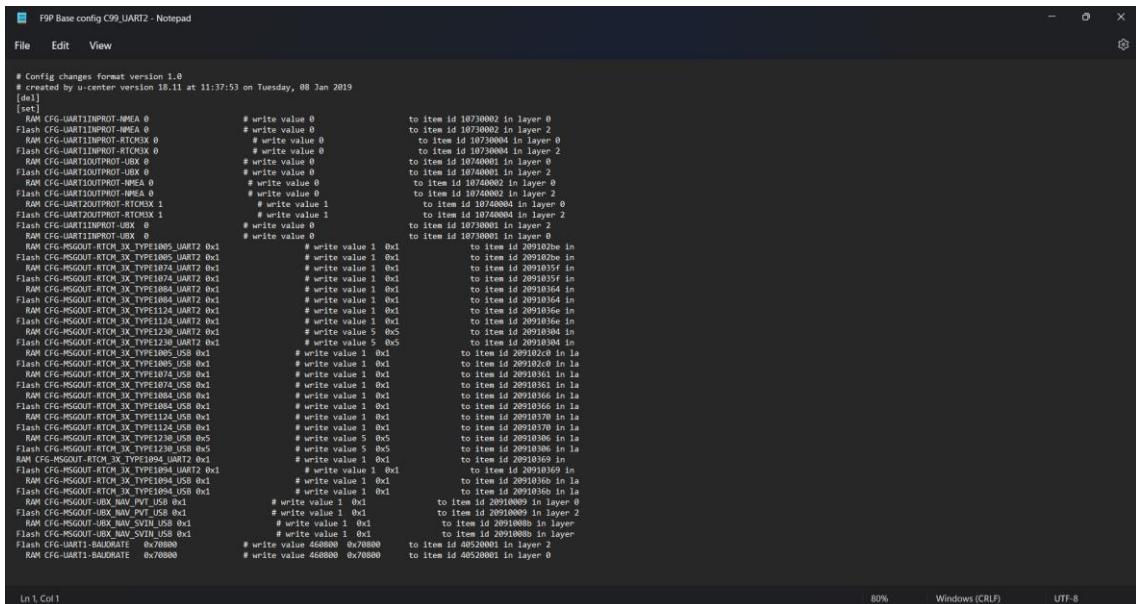


```

# Config changes format version 1.0
# created by u-center version 18.11 at 11:37:53 on Tuesday, 08 Jan 2019
[del]
[set]
RAM CFG-UART1INPROT-NMEA 0          # write value 0          to item id 10730002 in layer 0
Flash CFG-UART1INPROT-NMEA 0          # write value 0          to item id 10730002 in layer 2
RAM CFG-UART1INPROT-RTCMX 0          # write value 0          to item id 10730004 in layer 0
Flash CFG-UART1INPROT-RTCMX 0          # write value 0          to item id 10730004 in layer 2
RAM CFG-UART1OUTPROT-UBX 0           # write value 0          to item id 10740001 in layer 0
Flash CFG-UART1OUTPROT-UBX 0           # write value 0          to item id 10740001 in layer 2
RAM CFG-UART1OUTPROT-NMEA 0           # write value 0          to item id 10740003 in layer 0
Flash CFG-UART1OUTPROT-NMEA 0           # write value 0          to item id 10740003 in layer 2
RAM CFG-UART1OUTPROT-RTCMX1 0          # write value 1          to item id 10740008 in layer 0
Flash CFG-UART1OUTPROT-RTCMX1 0          # write value 1          to item id 10740008 in layer 2
RAM CFG-UART1INPROT-UBX 0           # write value 0          to item id 10750001 in layer 0
Flash CFG-UART1INPROT-UBX 0           # write value 0          to item id 10750001 in layer 2
RAM CFG-UART1INPROT-NMEA 0           # write value 0          to item id 10750002 in layer 0
Flash CFG-UART1INPROT-NMEA 0           # write value 0          to item id 10750002 in layer 2
RAM CFG-HSGOUT-RTCM_3X_TYPE10NS_UART1 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE10NS_UART1 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE10M_USB 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE10M_USB 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1074_UART1 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1074_UART1 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1084_UART1 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1084_UART1 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1086_UART1 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1086_UART1 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1124_UART1 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1124_UART1 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1126_UART1 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1126_UART1 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1128_UART1 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1128_UART1 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1194_UART1 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1194_UART1 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1196_USB 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1196_USB 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1197_USB 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1197_USB 0x1
RAM CFG-HSGOUT-UBX_NAV_PVT_USB 0x1
Flash CFG-HSGOUT-UBX_NAV_PVT_USB 0x1
RAM CFG-HSGOUT-UBX_NAV_SVHIN_USB 0x1
Flash CFG-HSGOUT-UBX_NAV_SVHIN_USB 0x1
RAM CFG-UART1-BAUDRATE 0x70000
Flash CFG-UART1-BAUDRATE 0x70000

```

รูปที่ 3.38 รายละเอียดการตั้งค่าต่างๆภายใน Base Configuration file (UART1)

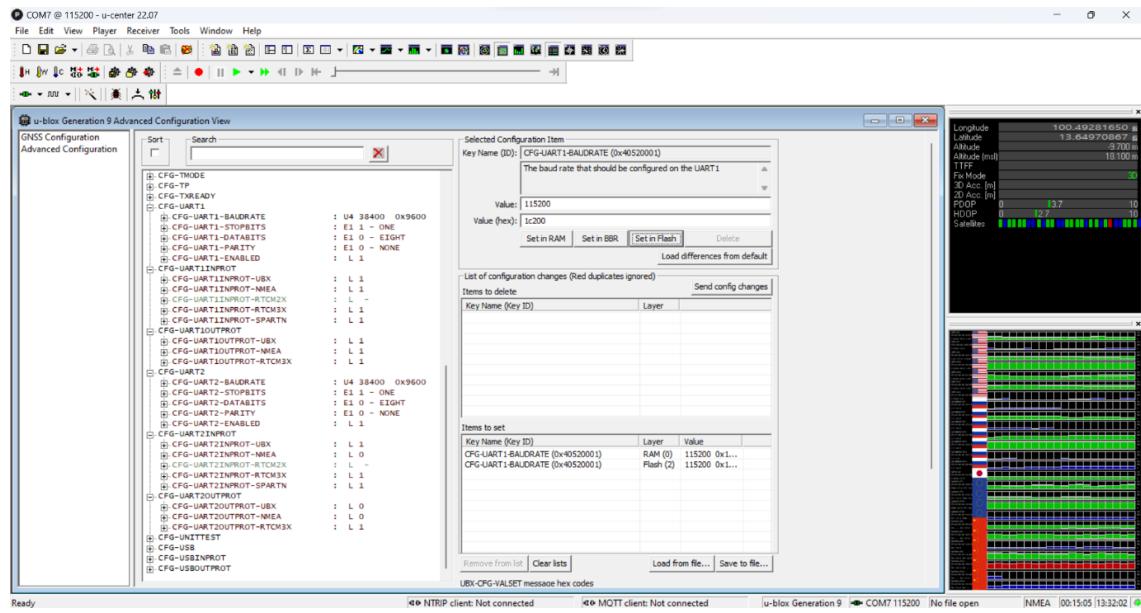


```

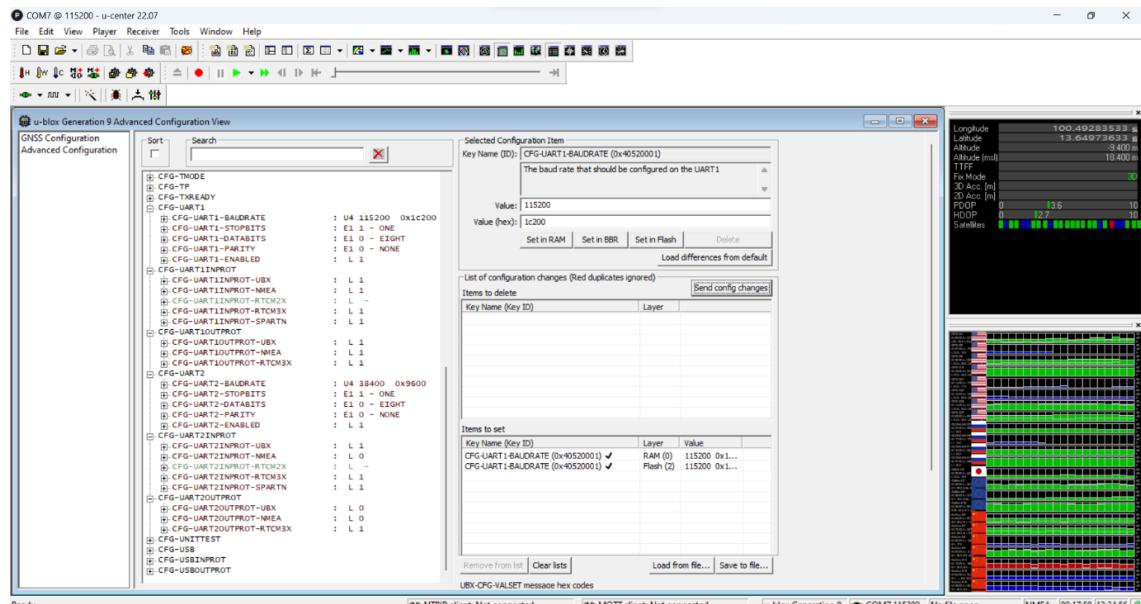
# Config changes format version 1.0
# created by u-center version 18.11 at 11:37:53 on Tuesday, 08 Jan 2019
[del]
[set]
RAM CFG-UART1INPROT-NMEA 0          # write value 0          to item id 10730001 in layer 0
Flash CFG-UART1INPROT-NMEA 0          # write value 0          to item id 10730001 in layer 2
RAM CFG-UART1INPROT-RTCMX 0          # write value 0          to item id 10730003 in layer 0
Flash CFG-UART1INPROT-RTCMX 0          # write value 0          to item id 10730003 in layer 2
RAM CFG-UART1OUTPROT-NMEA 0           # write value 0          to item id 10740001 in layer 0
Flash CFG-UART1OUTPROT-NMEA 0           # write value 0          to item id 10740001 in layer 2
RAM CFG-UART1OUTPROT-RTCMX1 0          # write value 1          to item id 10740008 in layer 0
Flash CFG-UART1OUTPROT-RTCMX1 0          # write value 1          to item id 10740008 in layer 2
RAM CFG-UART1INPROT-UBX 0           # write value 0          to item id 10750001 in layer 0
Flash CFG-UART1INPROT-UBX 0           # write value 0          to item id 10750001 in layer 2
RAM CFG-UART1INPROT-NMEA 0           # write value 0          to item id 10750002 in layer 0
Flash CFG-UART1INPROT-NMEA 0           # write value 0          to item id 10750002 in layer 2
RAM CFG-HSGOUT-RTCM_3X_TYPE10NS_UART2 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE10NS_UART2 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1074_UART2 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1074_UART2 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1084_UART2 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1084_UART2 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1086_UART2 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1086_UART2 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1124_UART2 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1124_UART2 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1126_UART2 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1126_UART2 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1128_UART2 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1128_UART2 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1194_UART2 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1194_UART2 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1196_USB 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1196_USB 0x1
RAM CFG-HSGOUT-RTCM_3X_TYPE1197_USB 0x1
Flash CFG-HSGOUT-RTCM_3X_TYPE1197_USB 0x1
RAM CFG-HSGOUT-UBX_NAV_PVT_USB 0x1
Flash CFG-HSGOUT-UBX_NAV_PVT_USB 0x1
RAM CFG-HSGOUT-UBX_NAV_SVHIN_USB 0x1
Flash CFG-HSGOUT-UBX_NAV_SVHIN_USB 0x1
RAM CFG-UART1-BAUDRATE 0x70000
Flash CFG-UART1-BAUDRATE 0x70000

```

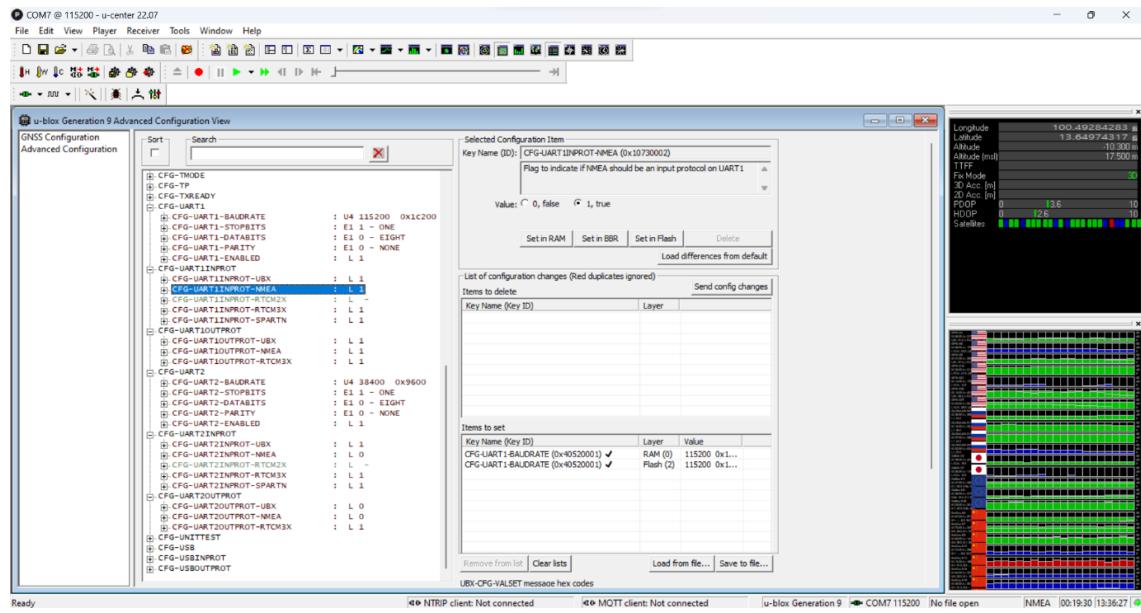
รูปที่ 3.39 การเปลี่ยนคำสั่งในการตั้งค่าเป็น UART2 สำหรับใช้ในการทดลอง



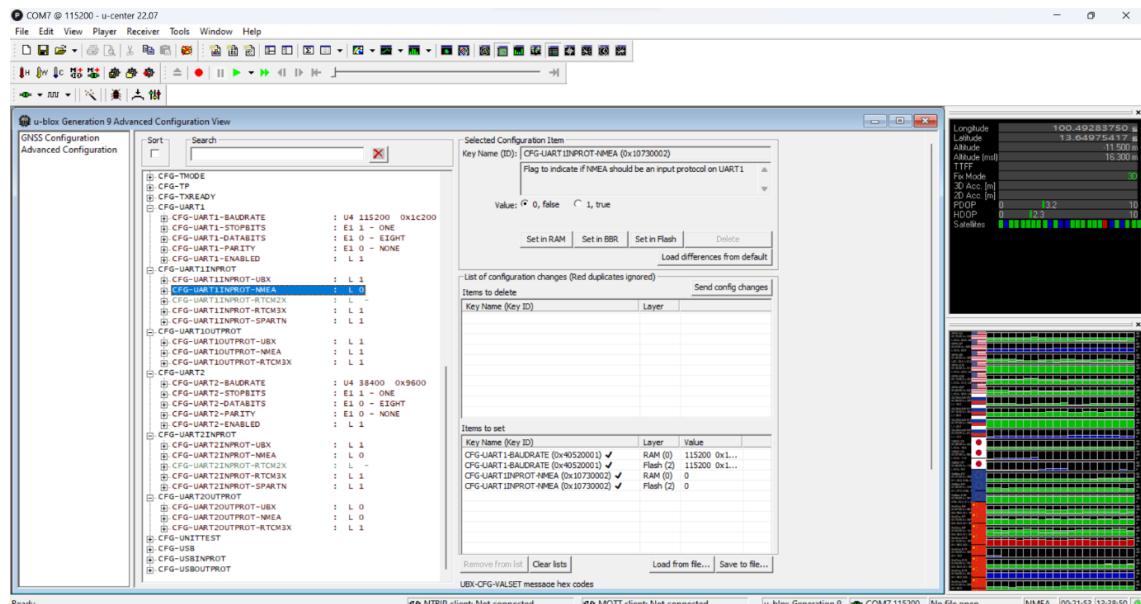
รูปที่ 3.40 ตัวอย่างการเปลี่ยนค่า Baud rate ของ UART1 แบบ Manual ก่อนส่งข้อมูล



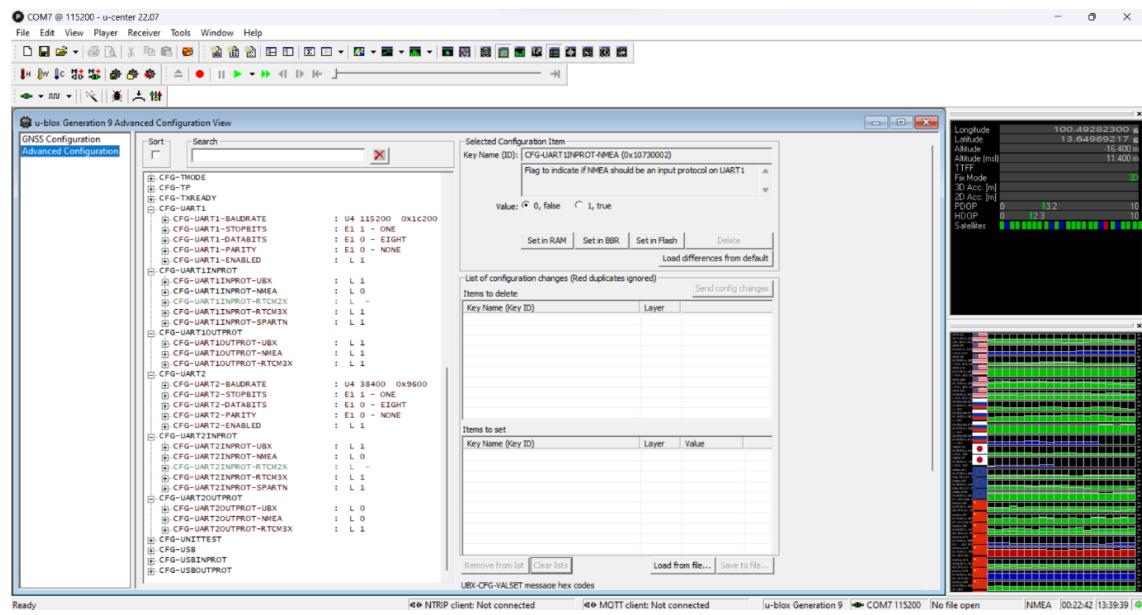
รูปที่ 3.41 ตัวอย่างการเปลี่ยนค่า Baud rate ของ UART1 แบบ Manual หลังส่งข้อมูล



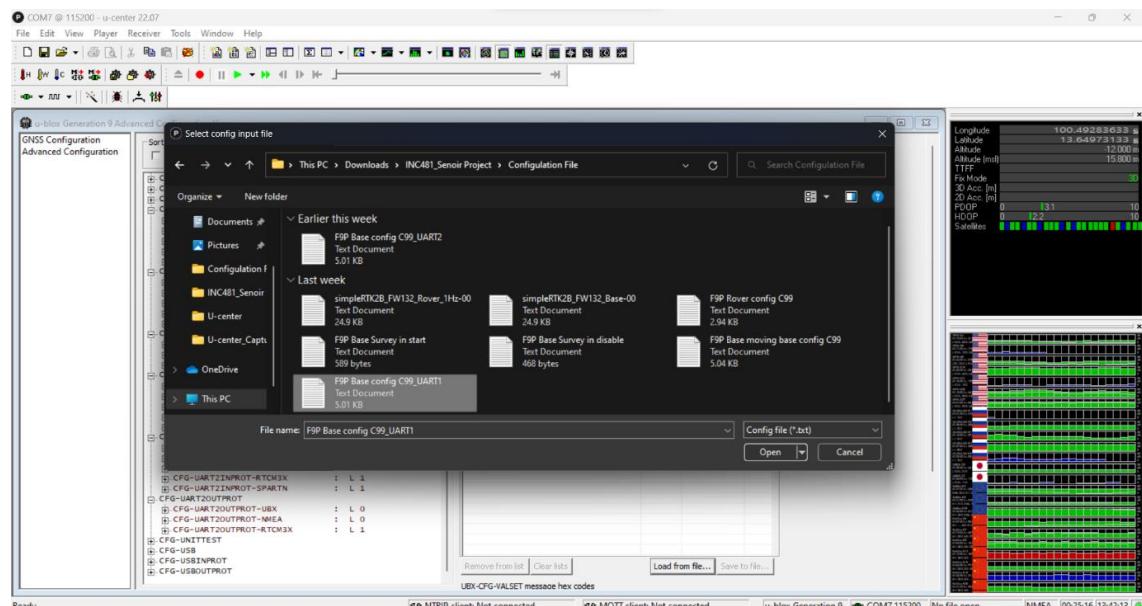
รูปที่ 3.42 ตัวอย่างการเปลี่ยน Protocol Input ของ UART1 แบบ Manual ก่อนส่งข้อมูล



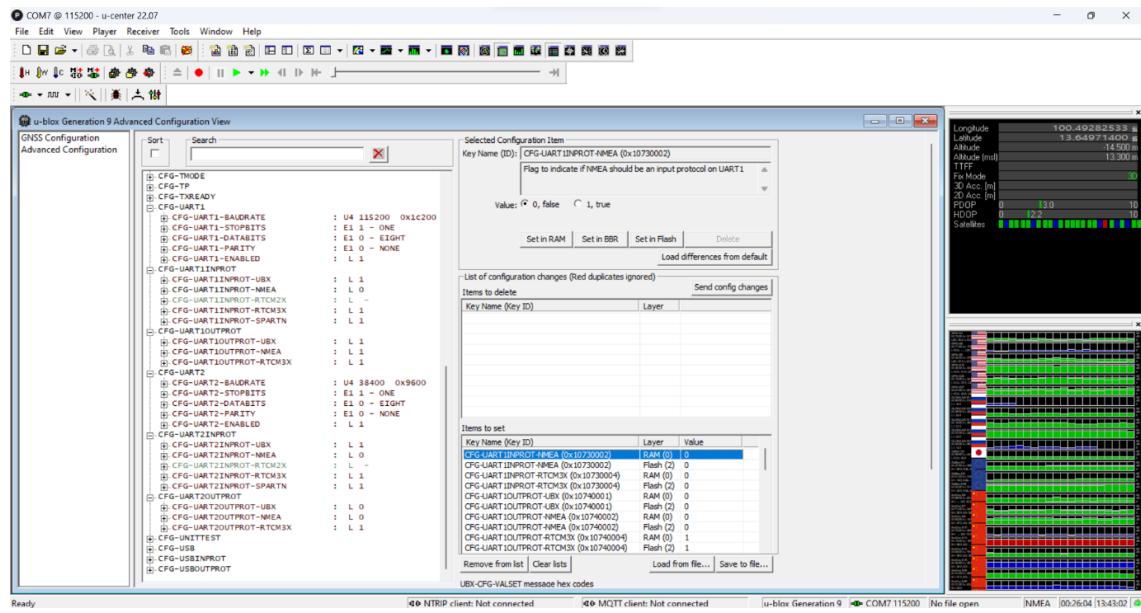
รูปที่ 3.43 ตัวอย่างการเปลี่ยน Protocol Input ของ UART1 แบบ Manual หลังส่งข้อมูล



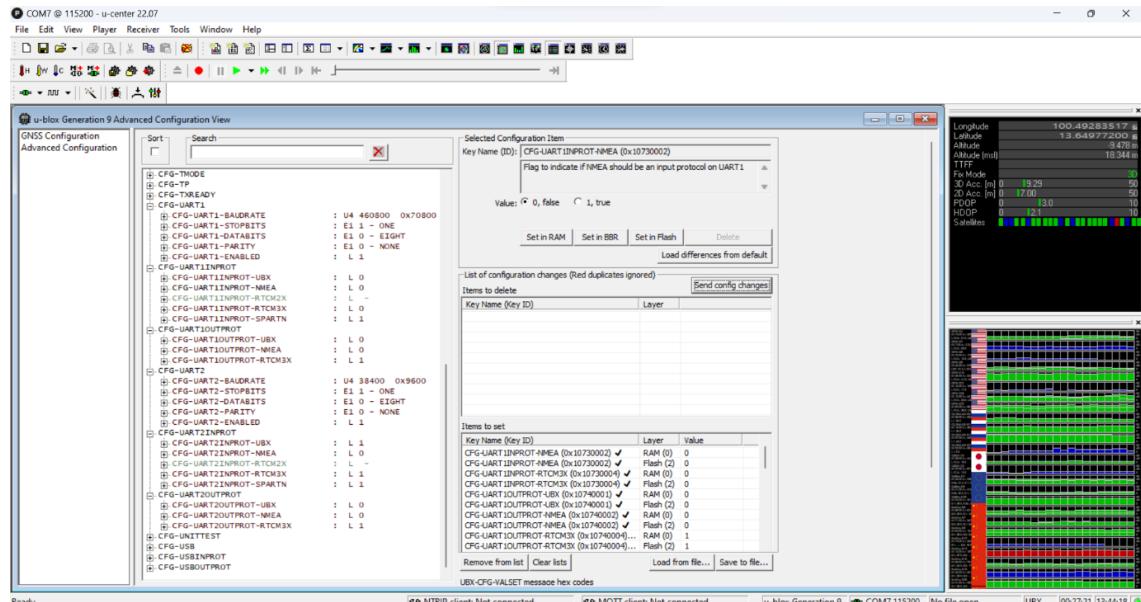
រូបថត 3.44 រាយ Clear Lists នៃ Key Name



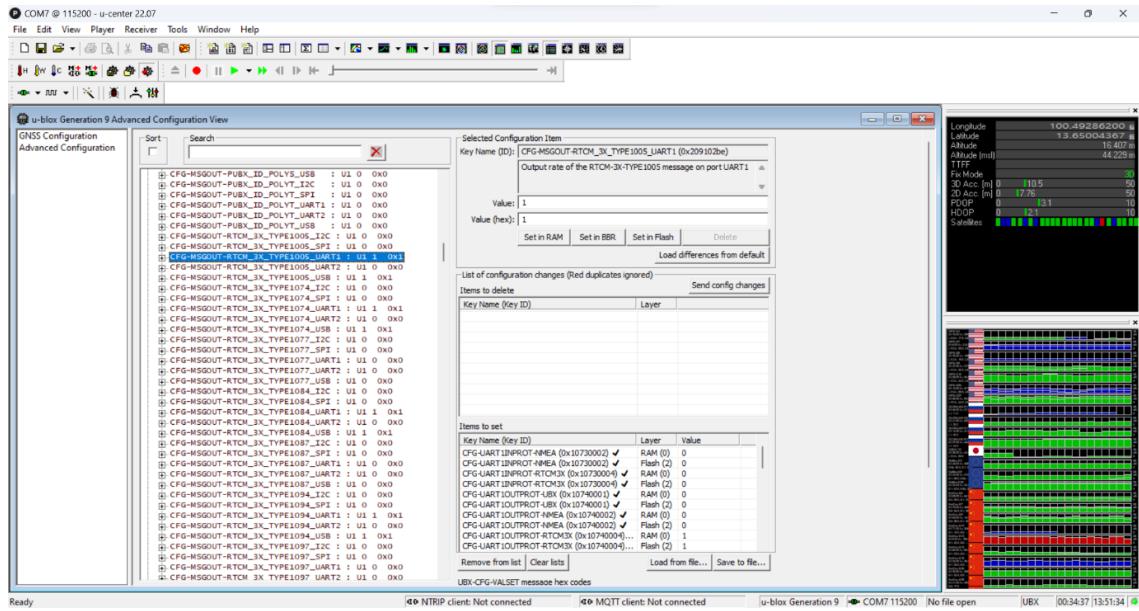
រូបថត 3.45 តាមរយៈការឡើកទិន្នន័យ Base Configuration file ទៅ UART1



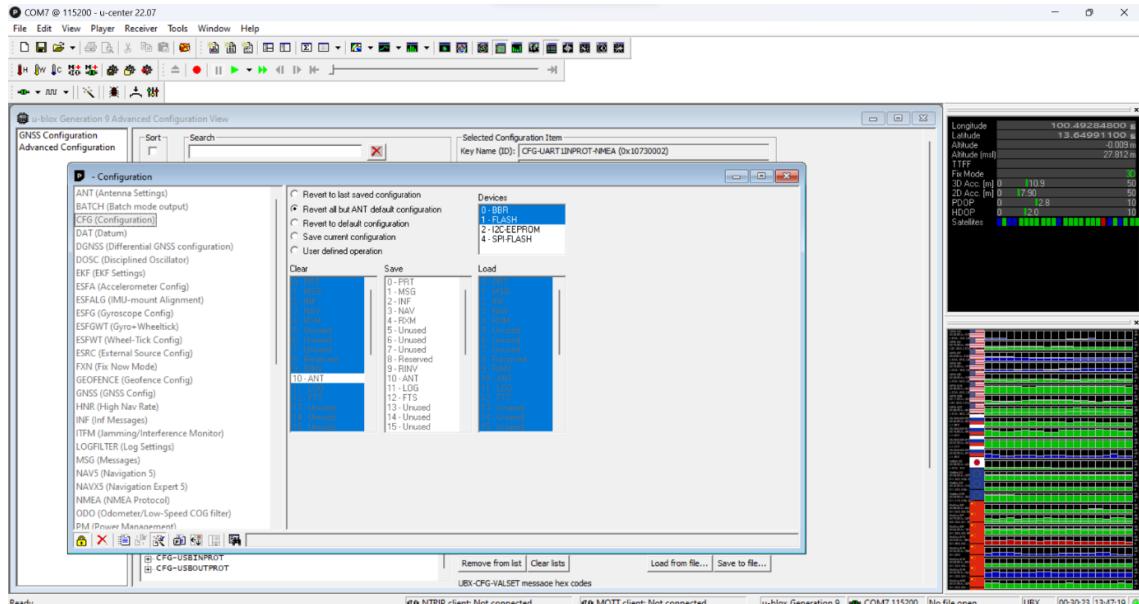
รูปที่ 3.46 ตัวอย่างการเลือกใช้ Base Configuration file ของ UART1 ลงใน Key Name สำเร็จ



รูปที่ 3.47 ตัวอย่างการอัพโหลดการตั้งค่า Base Configuration file ของ UART1 สำเร็จ

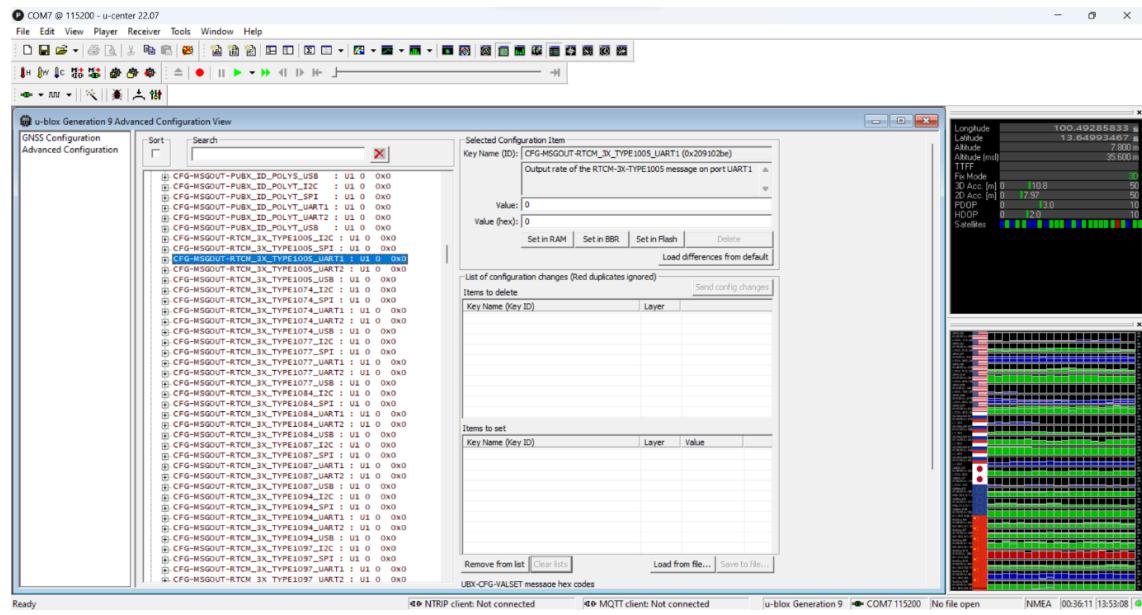


รูปที่ 3.48 การตรวจสอบการตั้งค่าหลังจากอพโหลด Configuration File UART1

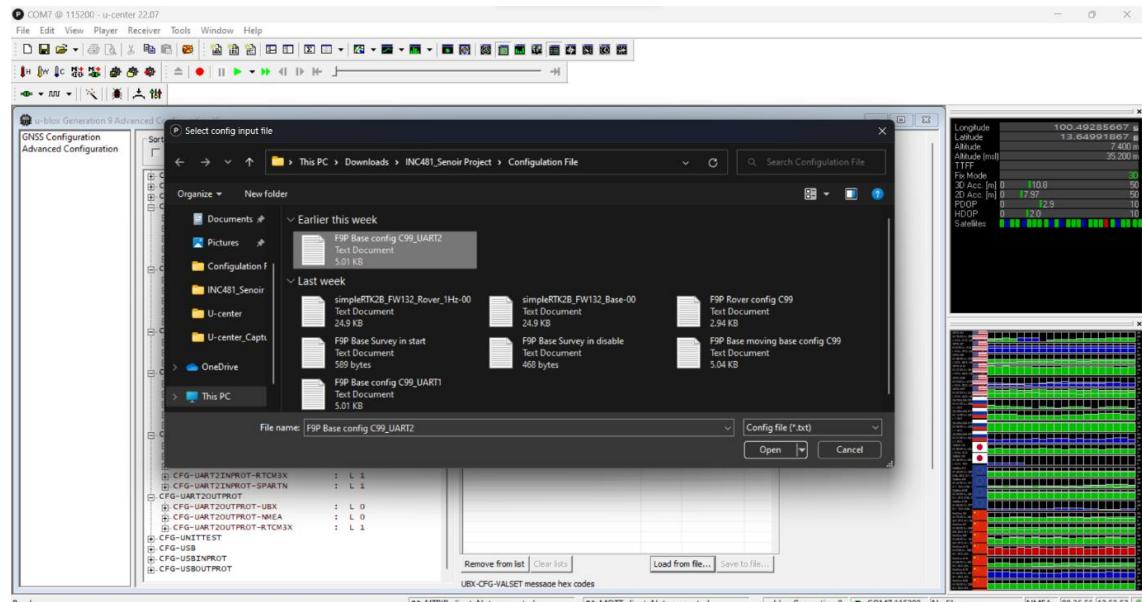


รูปที่ 3.49 การตั้งค่ารีเซ็ต Receiver เป็นค่าเริ่มต้น

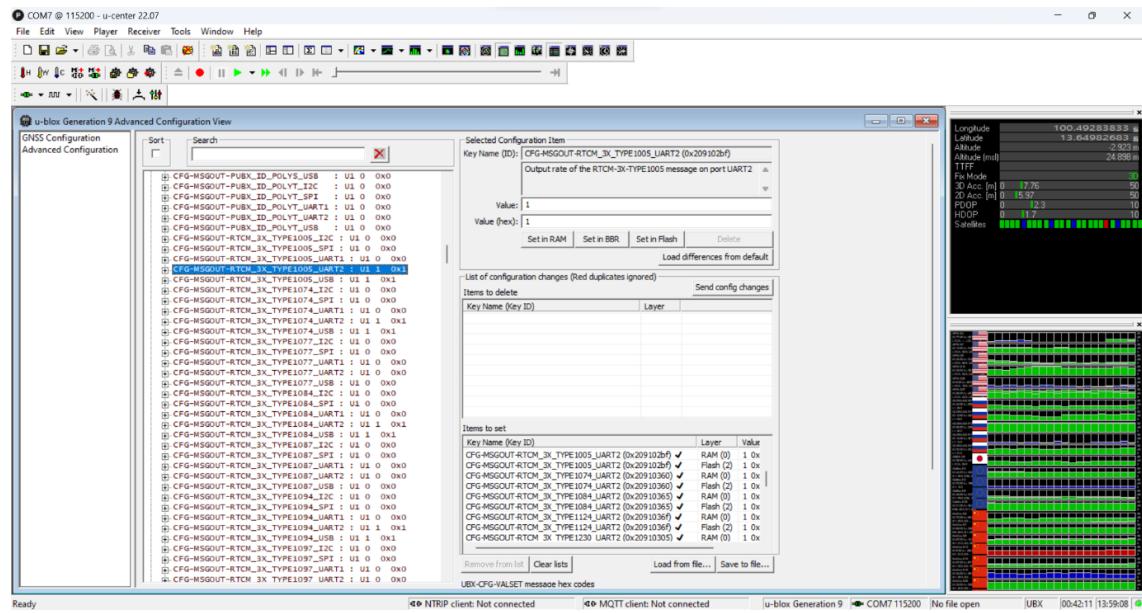
สำหรับการตั้งค่าเป็นค่าเริ่มต้นนั้นในขั้นตอนนี้จะตั้งค่าเพื่อเปลี่ยนการใช้งานจาก UART1 ไปเป็น UART2 ในการทดลองจริง



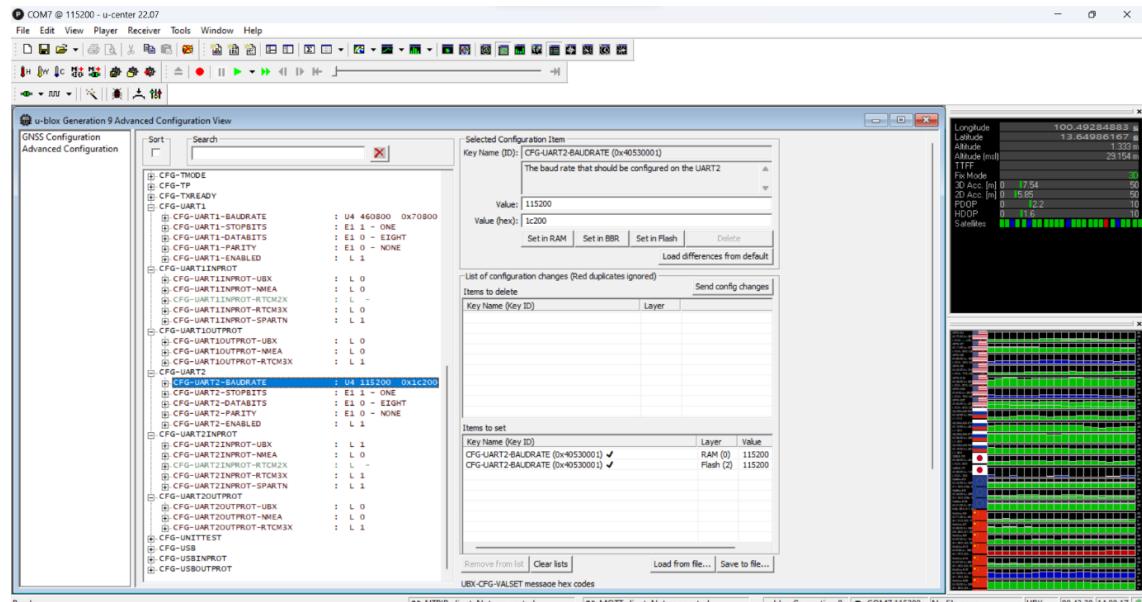
รูปที่ 3.50 การตรวจสอบการตั้งค่าหลังจากรีเซ็ต Receiver



รูปที่ 3.51 ตัวอย่างการเลือกใช้ Base Configuration file ของ UART2

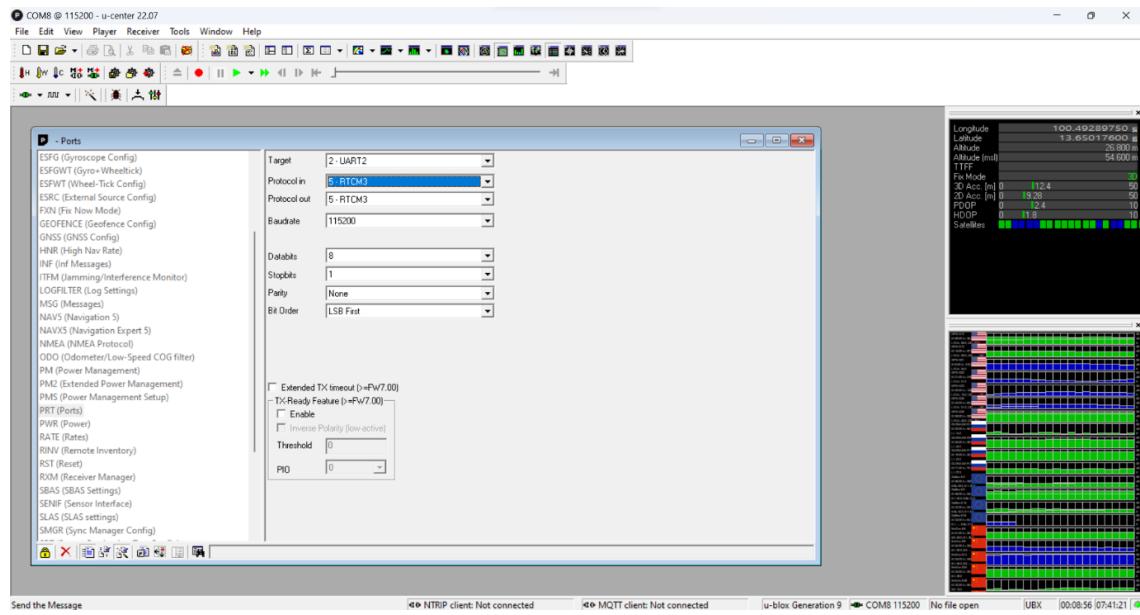


รูปที่ 3.52 การตรวจสอบการตั้งค่าหลังจากอพโหลด Configuration File UART2



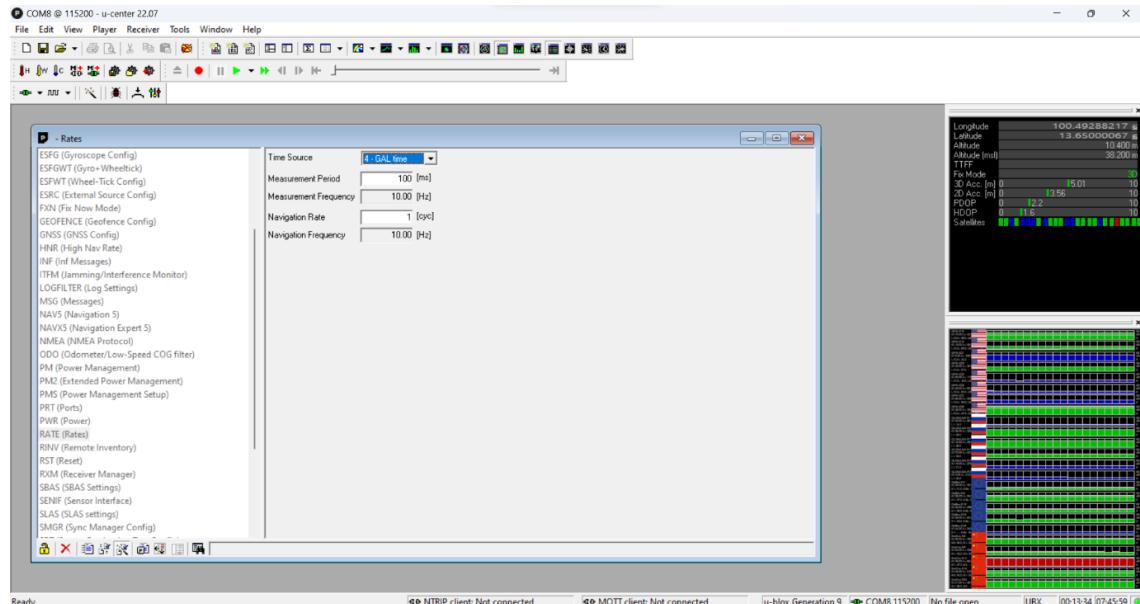
รูปที่ 3.53 การเปลี่ยน Baud rate ของ UART2 แบบ Manual

หมายเหตุ : การตั้งค่า UART Baud rate ของ Base Receiver และ Rover Receiver ควรตั้งค่าให้มีความเร็ว กัน

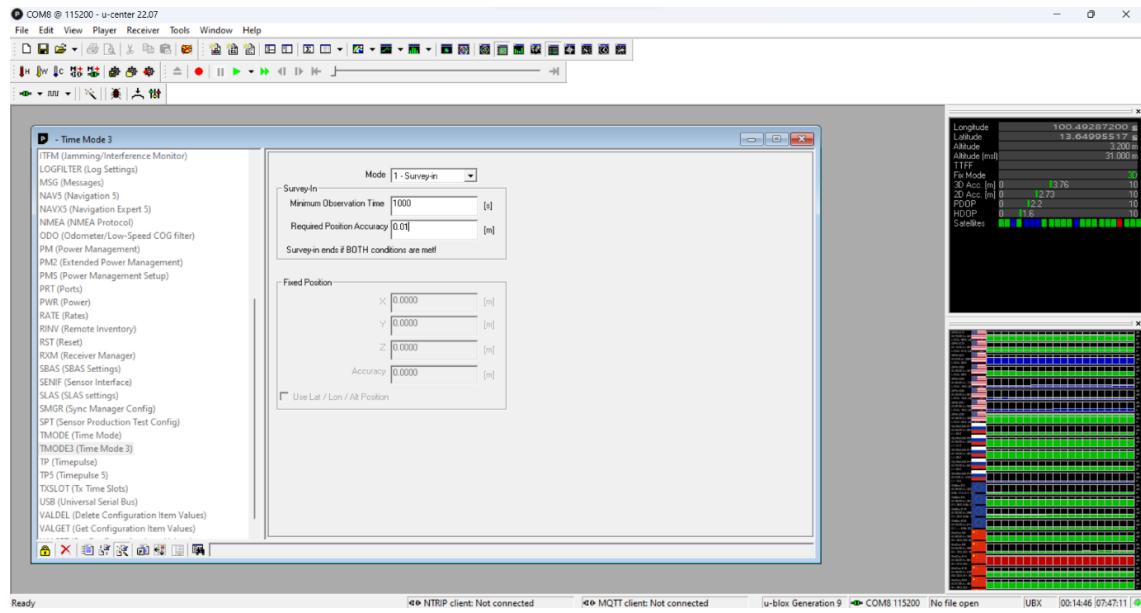


รูปที่ 3.54 การตั้งค่า Protocol และ Baud rate แบบ Manual ผ่าน Configuration view

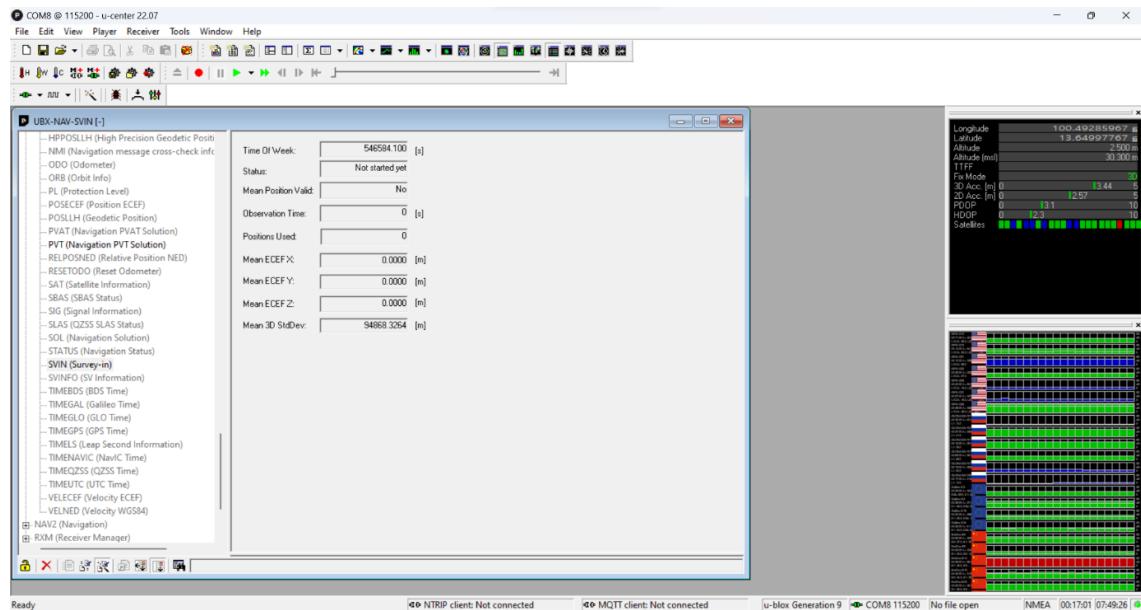
หมายเหตุ : การตั้งค่าต่างๆของ UART สามารถทำได้ทั้งผ่าน Configuration view และ Advanced Configuration ซึ่งจะให้ผลลัพธ์สุดท้ายเหมือนกัน



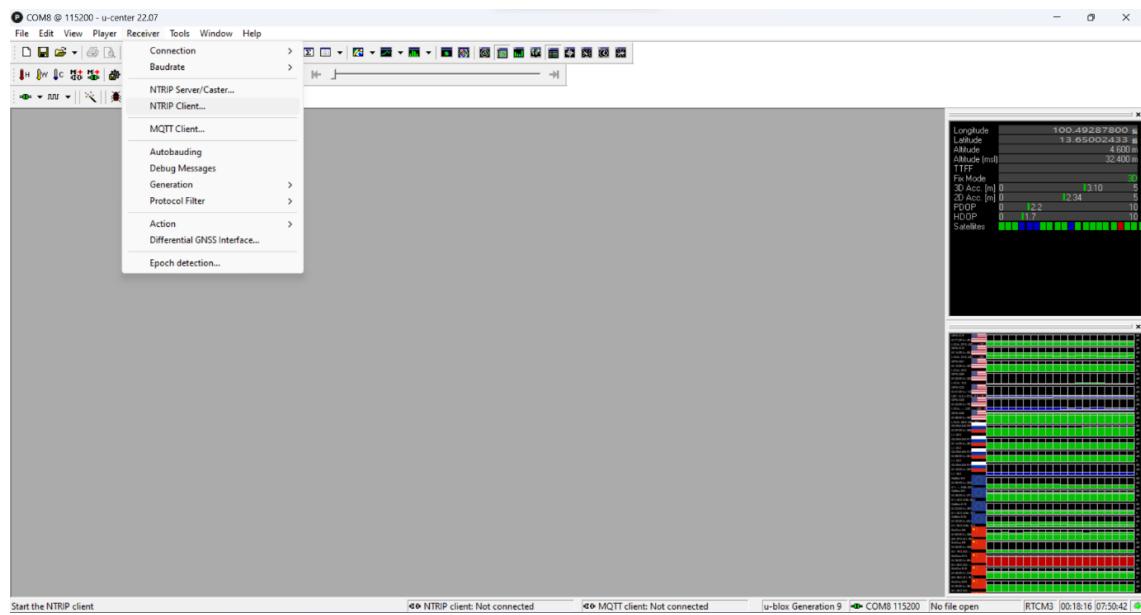
รูปที่ 3.55 การตั้งค่า Measurement Period ของทุก Time Source



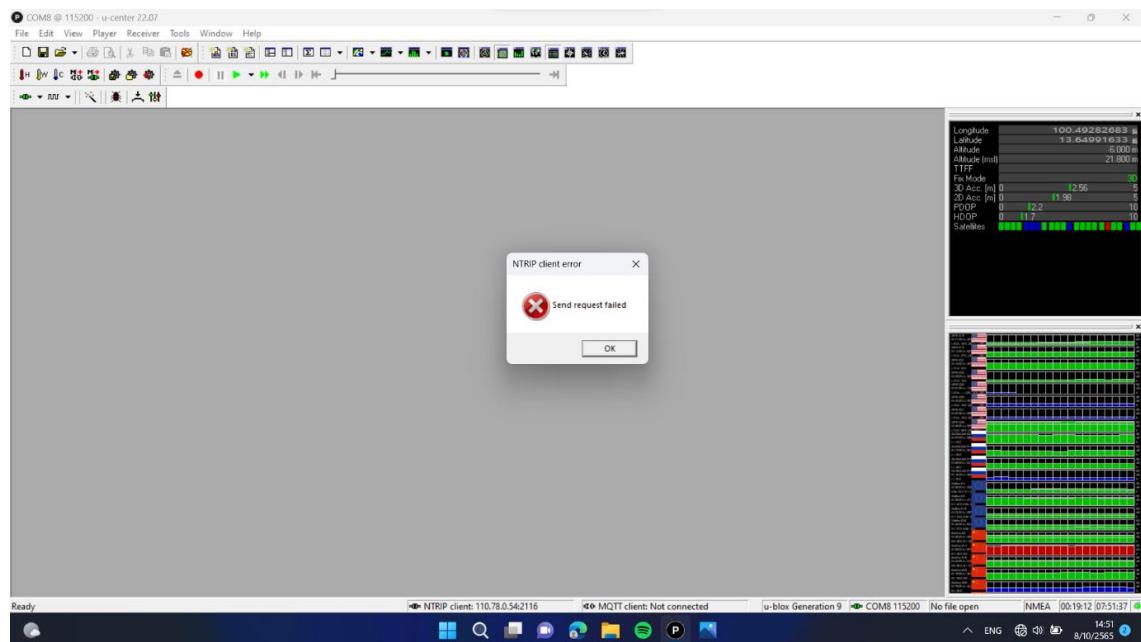
รูปที่ 3.56 การกำหนดเวลาและความละเอียดในการหาพิกัดที่ตั้งของ Base Receiver



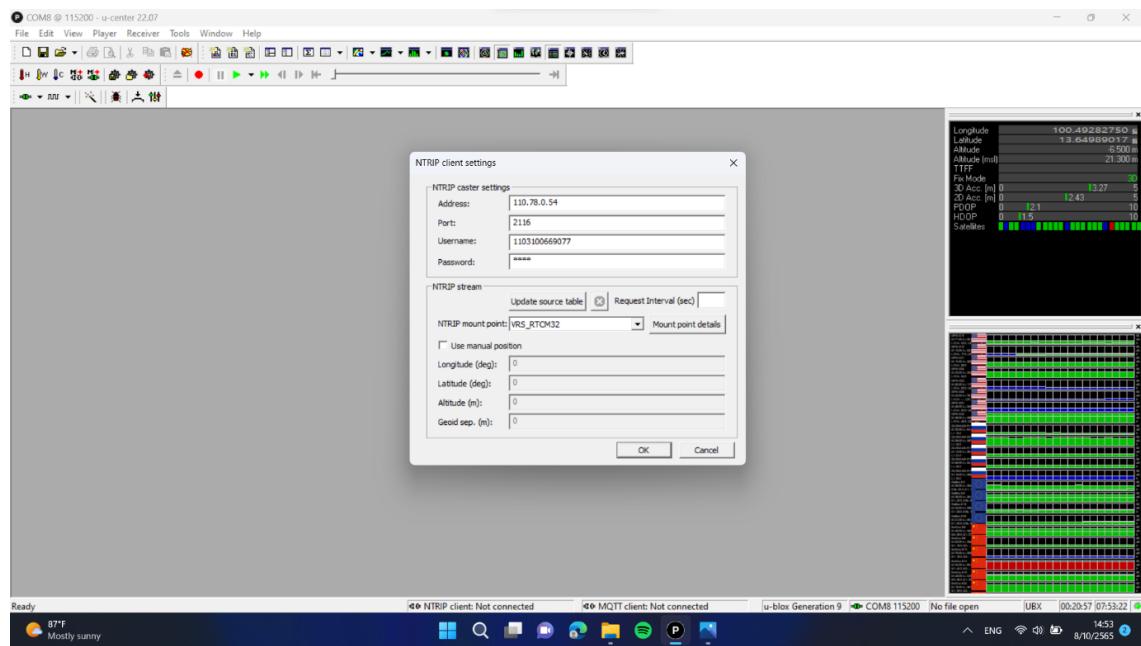
รูปที่ 3.57 หน้าต่าง Message View ที่แสดงสถานะการ Survey-in



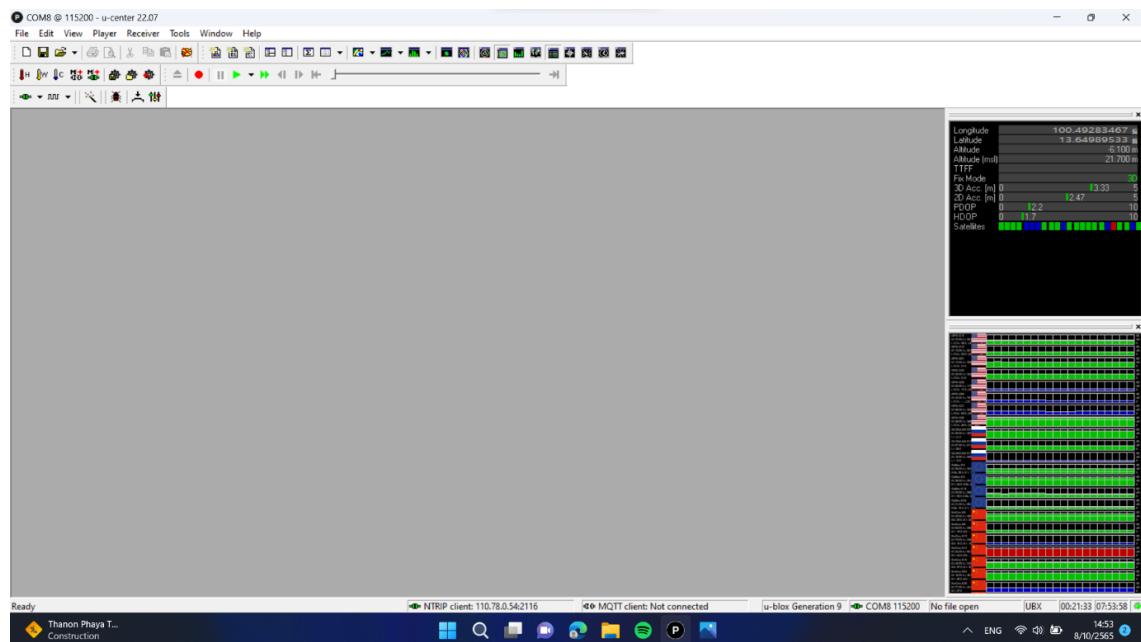
รูปที่ 3.58 การเลือกใช้ NTRIP เพื่อรับค่าปรับแก้จาก CORS



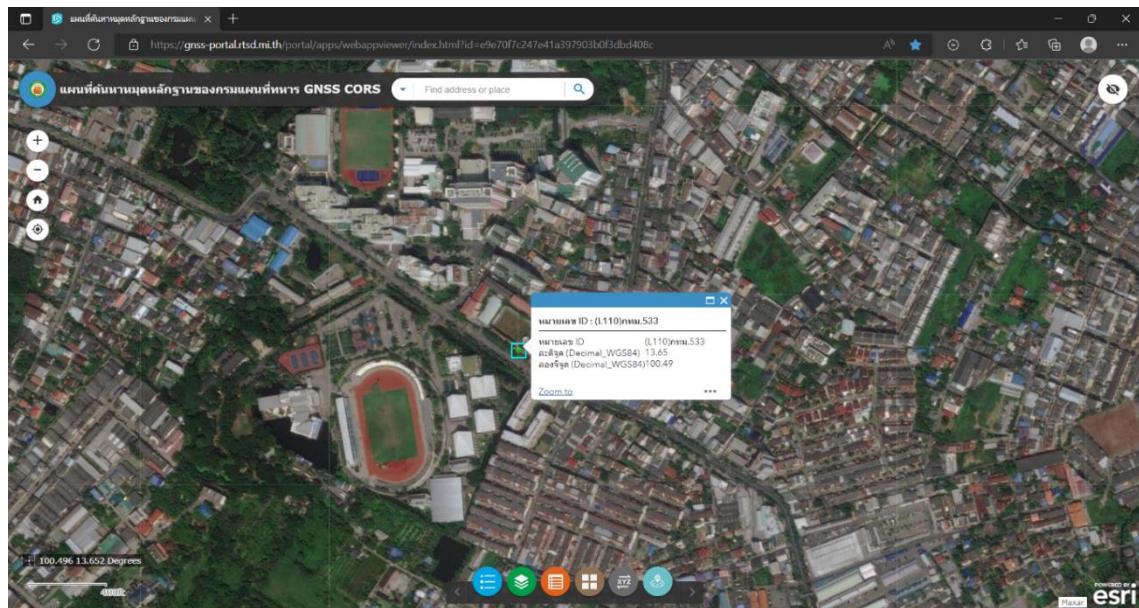
รูปที่ 3.59 ตัวอย่างการเชื่อมต่อ NTRIP ไม่สำเร็จเนื่องจากไม่ได้เชื่อมต่อ Internet



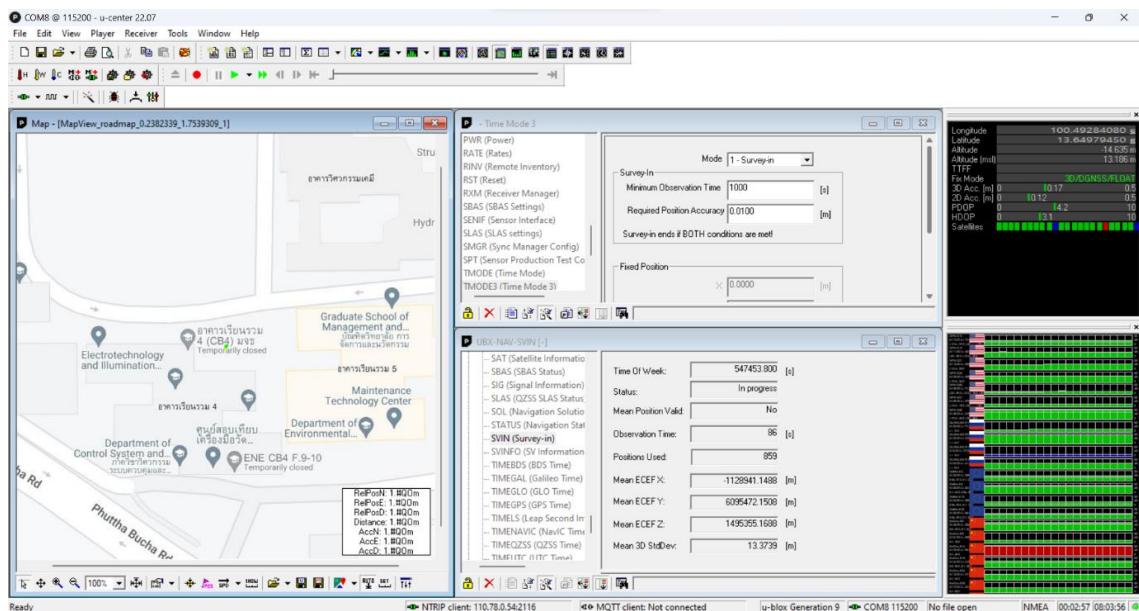
รูปที่ 3.60 การใส่ Username และ Password ที่ขอรับจากการที่ดิน



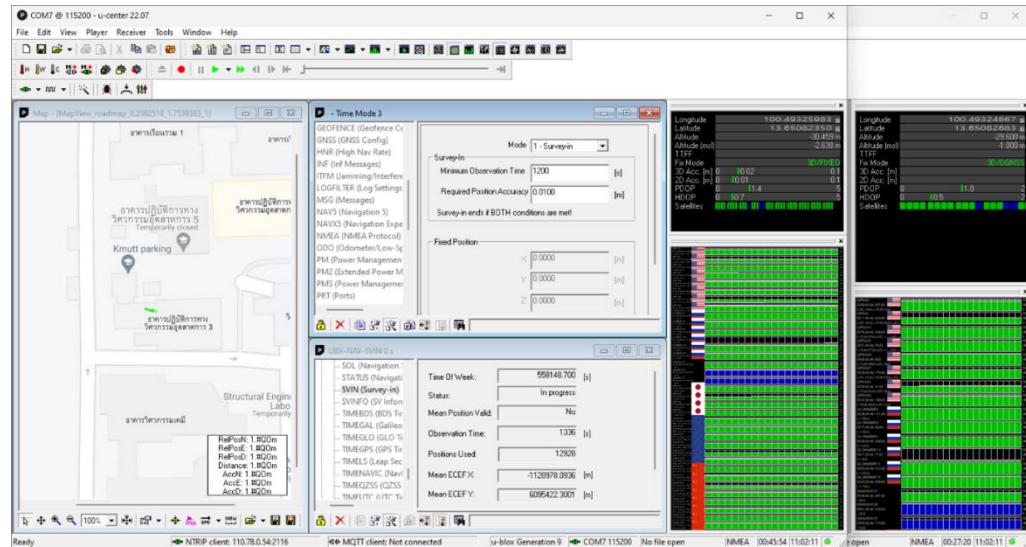
รูปที่ 3.61 สถานะ NTRIP Client จะเปลี่ยนเป็นสีเขียวหลังจากเชื่อมต่อสำเร็จ



รูปที่ 3.62 Continuous Operation Reference Station บริเวณ ไก่เคียงกับมหาวิทยาลัย

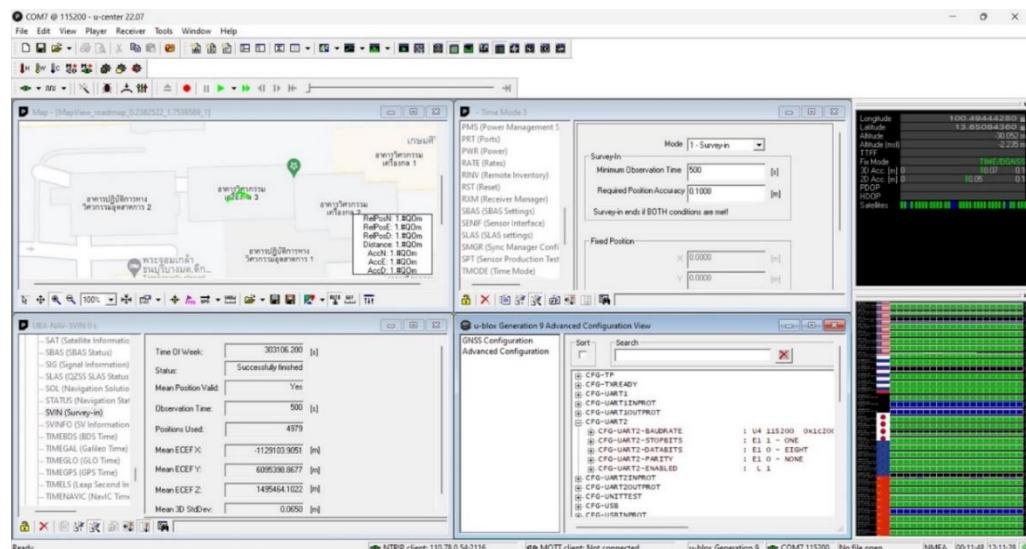


รูปที่ 3.63 หน้าโปรแกรมขณะ Survey คืนหาพิกัดในการติดตั้ง Base Station



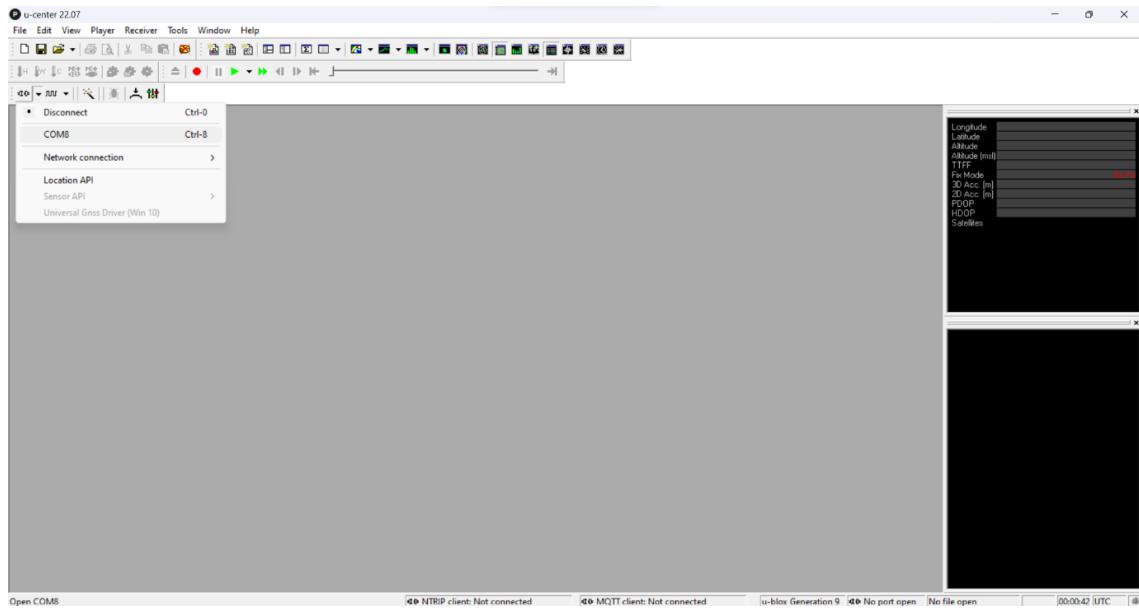
รูปที่ 3.64 ตัวอย่างการ Survey ไม่สำเร็จ

สำหรับการ Survey ในการค้นหาพิกัด สามารถกำหนดเวลาในการรังวัดและความละเอียดที่ต้องการได้ ซึ่ง ควรกำหนดเงื่อนไขในการ Survey ให้สอดคล้องกัน ความหมายคือ หากต้องการความละเอียดในการ ตรวจค่าพิกัดที่สูงจำเป็นต้องใช้เวลาในการ Survey ที่มากตามไปด้วย และที่สำคัญที่สุดควรเลื่อนสถานที่ ในการ Survey ที่ไม่ถูกไฟล์ Error ใน การรับสัญญาณจากดาวเทียม เช่น ควรเลือกสถานที่ที่ห้องฟ้าเปิด ไม่มีตึกหรือสิ่งปลูกสร้าง โดยรอบ

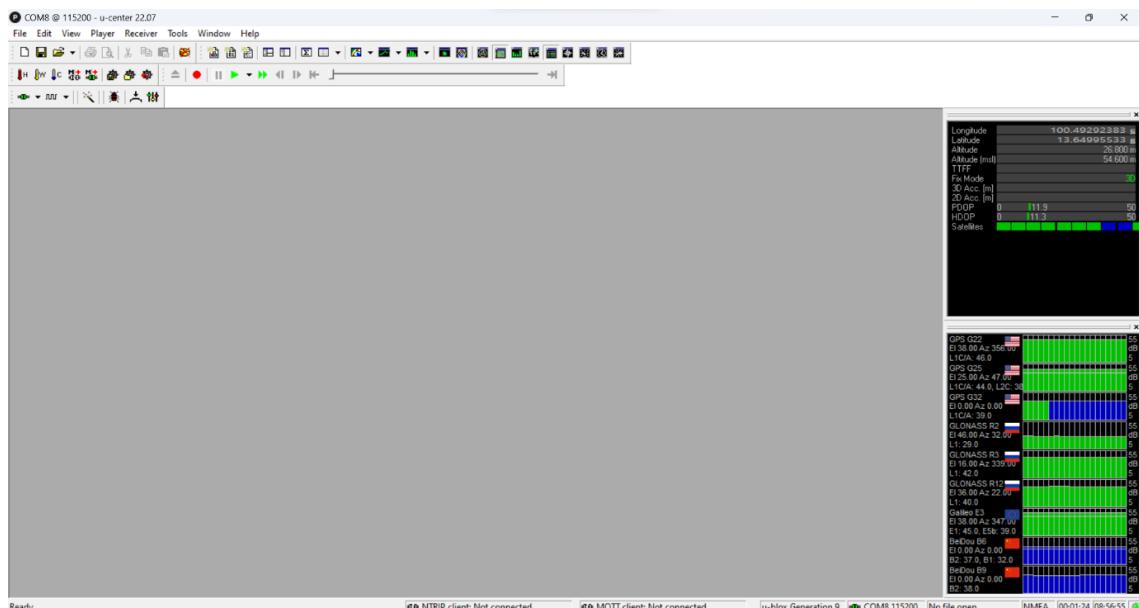


รูปที่ 3.65 ตัวอย่างการ Survey สำเร็จจะแสดงสถานะ TIME

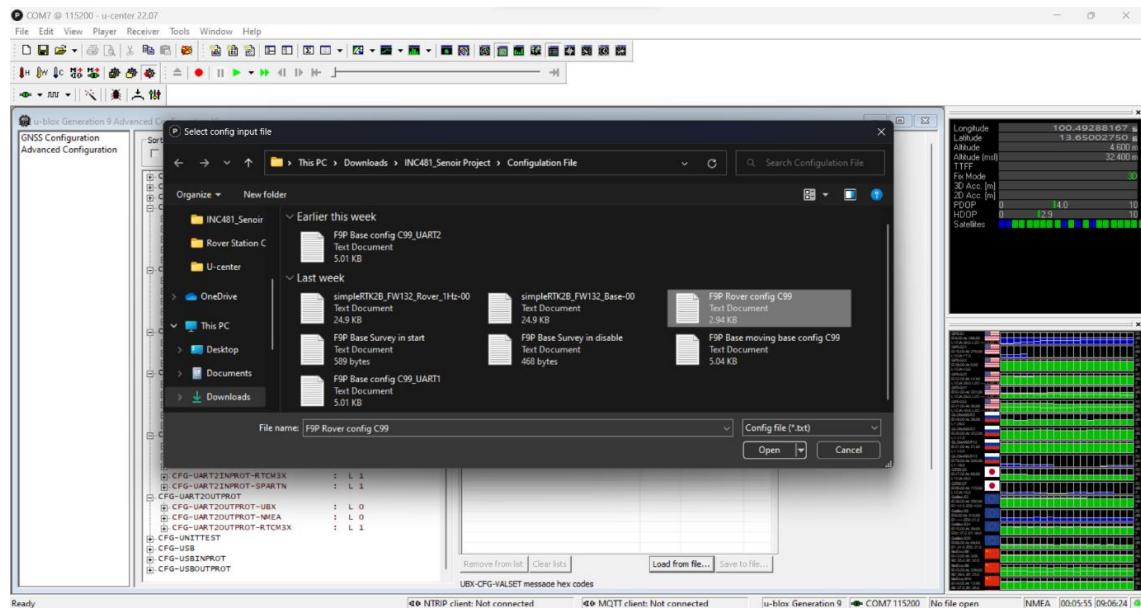
## 2. การตั้งค่า Rover Station ในเทคนิคการรังวัดแบบ Real Time Kinematic



รูปที่ 3.66 การเลือก Com port ใน การเชื่อมต่อ กับ Rover Receiver



รูปที่ 3.67 หน้าโปรแกรม U-center หลังจากเชื่อมต่อ กับ Rover Receiver



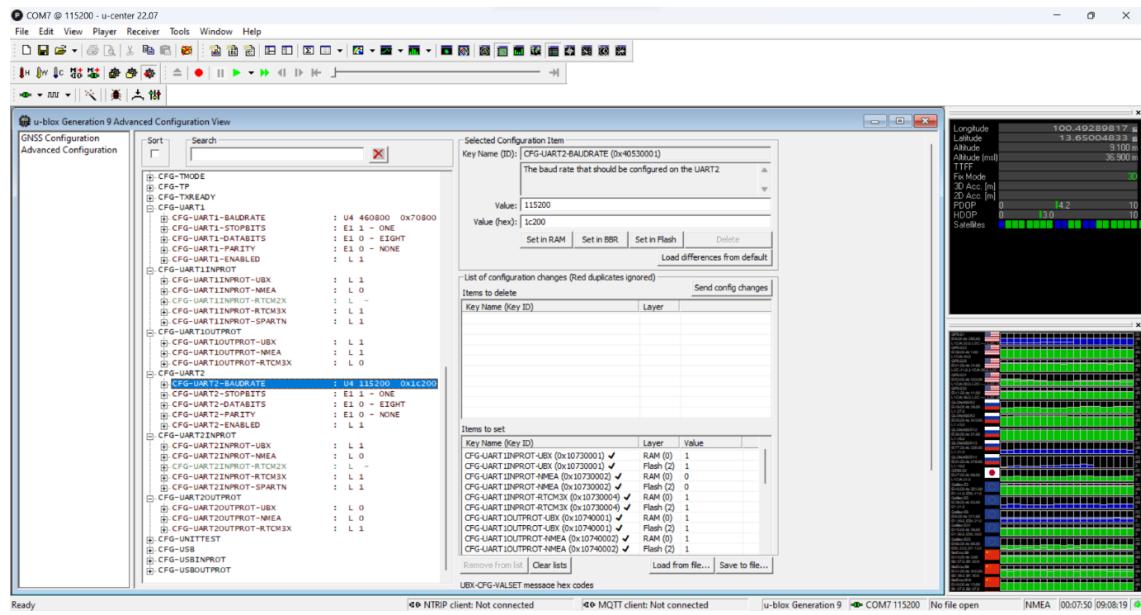
รูปที่ 3.68 การเลือกใช้ Rover Configuration file

```

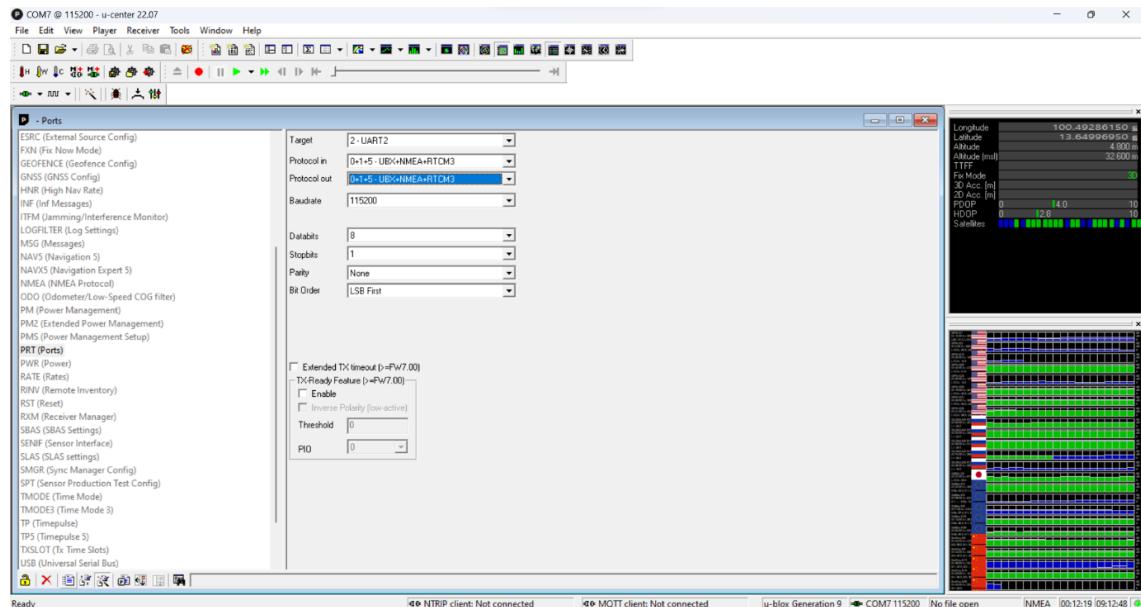
# Config changes format version 1.0
# created by u-center version 18.11 at 11:16:51 on Tuesday, 27 Nov 2018
[del]
[set]
    RAM CFG-UART1INPROT-UBX 1      # write value 1
Flash CFG-UART1INPROT-UBX 1      # write value 1
    RAM CFG-UART1INPROT-NMEA 0      # write value 0
Flash CFG-UART1INPROT-NMEA 0      # write value 0
    RAM CFG-UART1INPROT-RTCM3X 1     # write value 1
Flash CFG-UART1INPROT-RTCM3X 1     # write value 1
    RAM CFG-UART1OUTPROT-UBX 1      # write value 1
Flash CFG-UART1OUTPROT-UBX 1      # write value 1
    RAM CFG-UART1OUTPROT-NMEA 1      # write value 1
Flash CFG-UART1OUTPROT-NMEA 1      # write value 1
    RAM CFG-UART1OUTPROT-RTCM3X 0     # write value 0
Flash CFG-UART1OUTPROT-RTCM3X 0     # write value 0
    RAM CFG-USBINPROT-UBX 1      # write value 1
Flash CFG-USBINPROT-UBX 1      # write value 1
    RAM CFG-USBINPROT-NMEA 1      # write value 1
Flash CFG-USBINPROT-NMEA 1      # write value 1
    RAM CFG-USBINPROT-RTCM3X 1     # write value 1
Flash CFG-USBINPROT-RTCM3X 1     # write value 1
    RAM CFG-USBOUTPROT-UBX 1      # write value 1
Flash CFG-USBOUTPROT-UBX 1      # write value 1
    RAM CFG-USBOUTPROT-NMEA 1      # write value 1
Flash CFG-USBOUTPROT-NMEA 1      # write value 1
    RAM CFG-USBOUTPROT-RTCM3X 0     # write value 0
Flash CFG-USBOUTPROT-RTCM3X 0     # write value 0
    RAM CFG-UART1-BAUDRATE 0x70800  # write value 460800 0x70800
    RAM CFG-UART1-BAUDRATE 0x70800  # write value 460800 0x70800

```

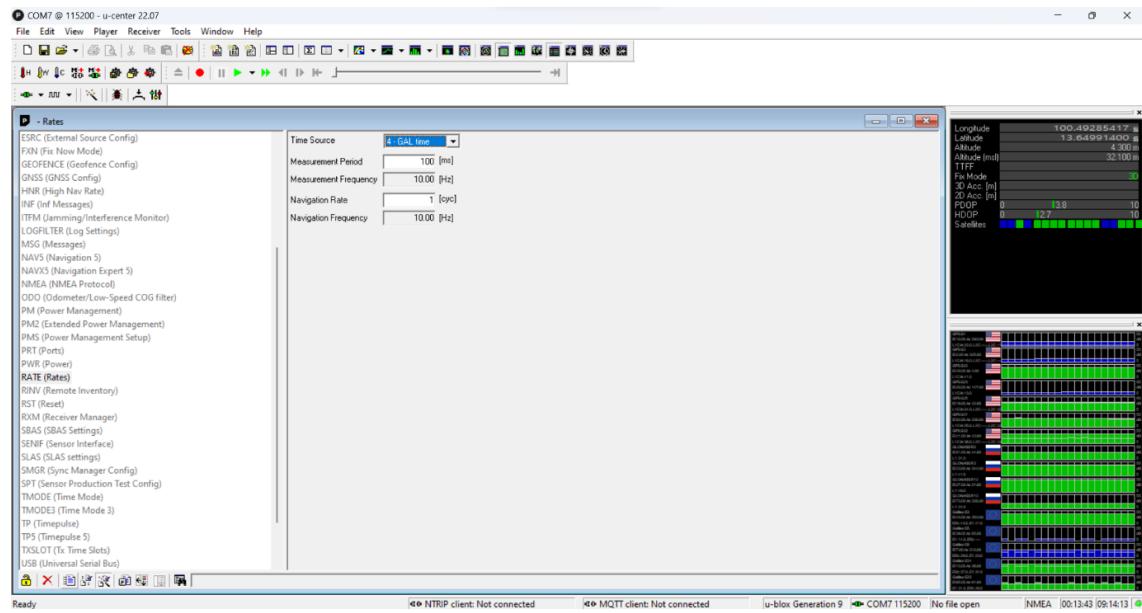
รูปที่ 3.69 รายละเอียดการตั้งค่าต่างๆ ภายใน Rover Configuration file



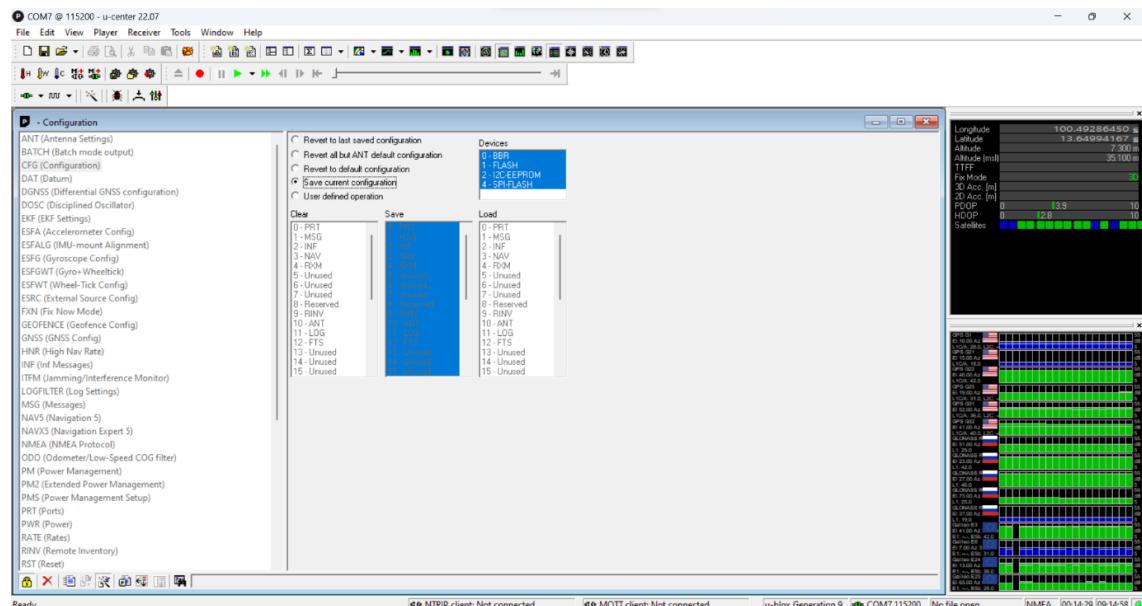
รูปที่ 3.70 การเปลี่ยนค่า Baud rate ของ UART2



รูปที่ 3.71 การตั้งค่า Protocol และ Baud rate แบบ Manual ผ่าน Configuration view

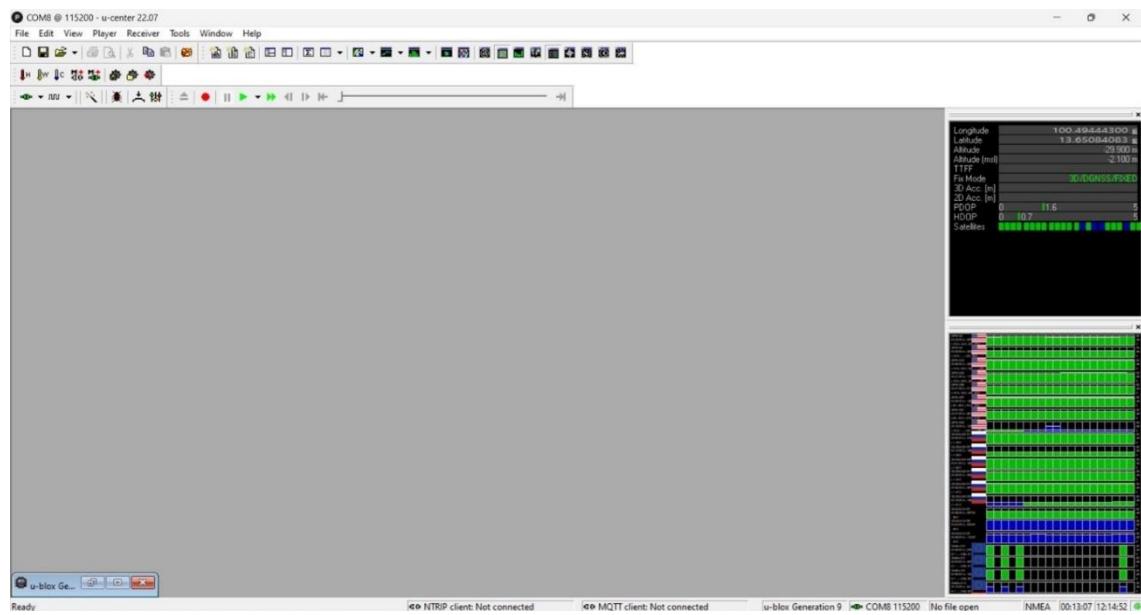


รูปที่ 3.72 การตั้งค่า Measurement Period ของทุก Time Source

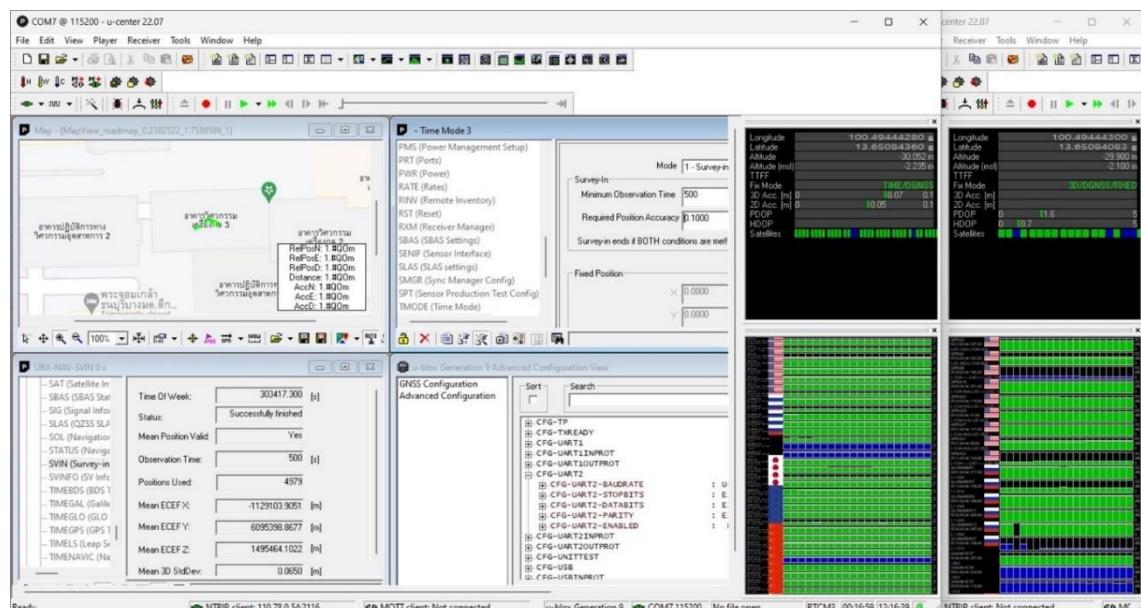


รูปที่ 3.73 การบันทึกการตั้งค่าทั้งหมดใน Receiver

หมายเหตุ : การบันทึกการตั้งค่าทุกครั้งหลังจากที่เปลี่ยนแปลงการตั้งค่าทั้ง Base Receiver และ Rover Receiver เนื่องจากหากไม่ได้บันทึกหลังจากถอด Power Supply การตั้งค่าทั้งหมดของ Receiver จะกลับไปเป็นแบบการตั้งค่าก่อนหน้า

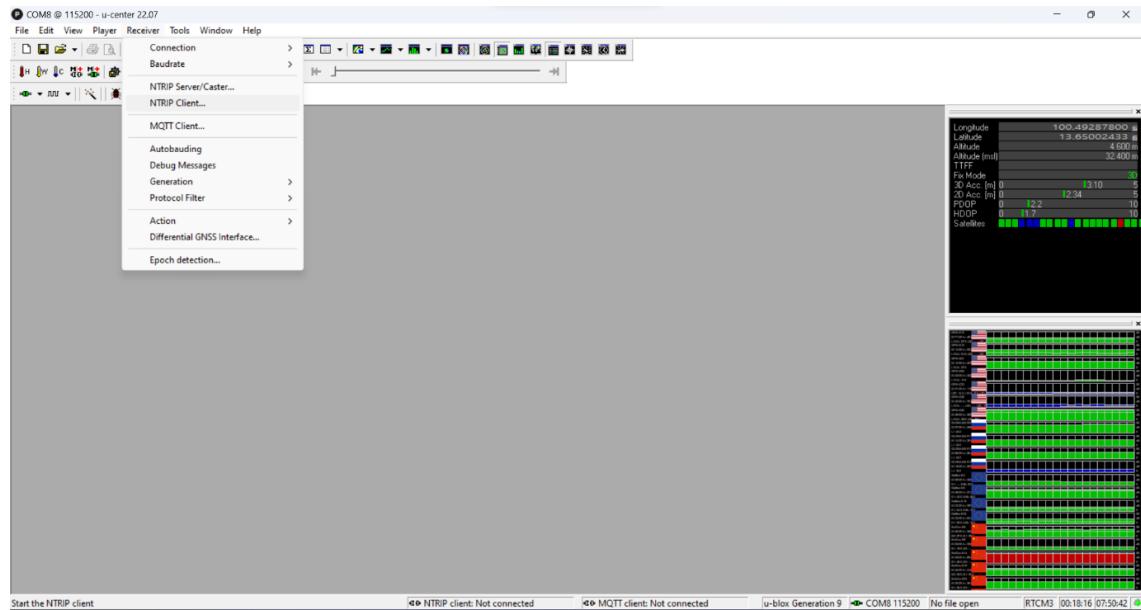


รูปที่ 3.74 การแสดงสถานะ FIXED หลังจากที่เชื่อมต่อกับ Base Receiver

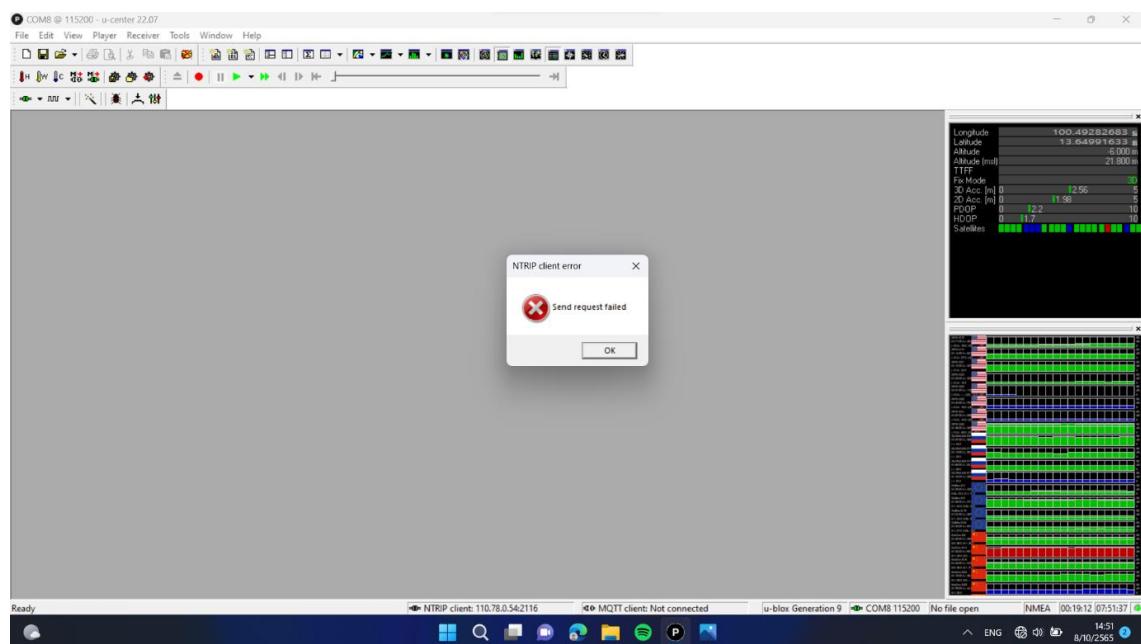


รูปที่ 3.75 ตัวอย่างการแสดงสถานะ TIME ของ Base และ FIXED ของ Rover

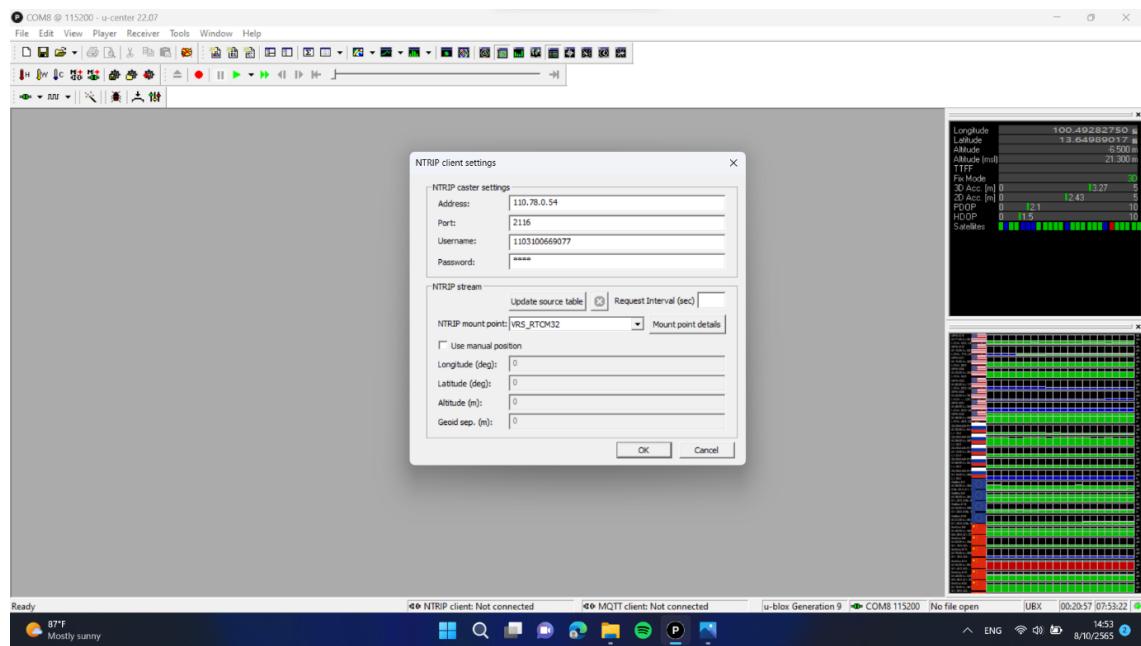
### 3. การตั้งค่า Rover Station ในเทคนิคการรังวัดแบบ Network RTK



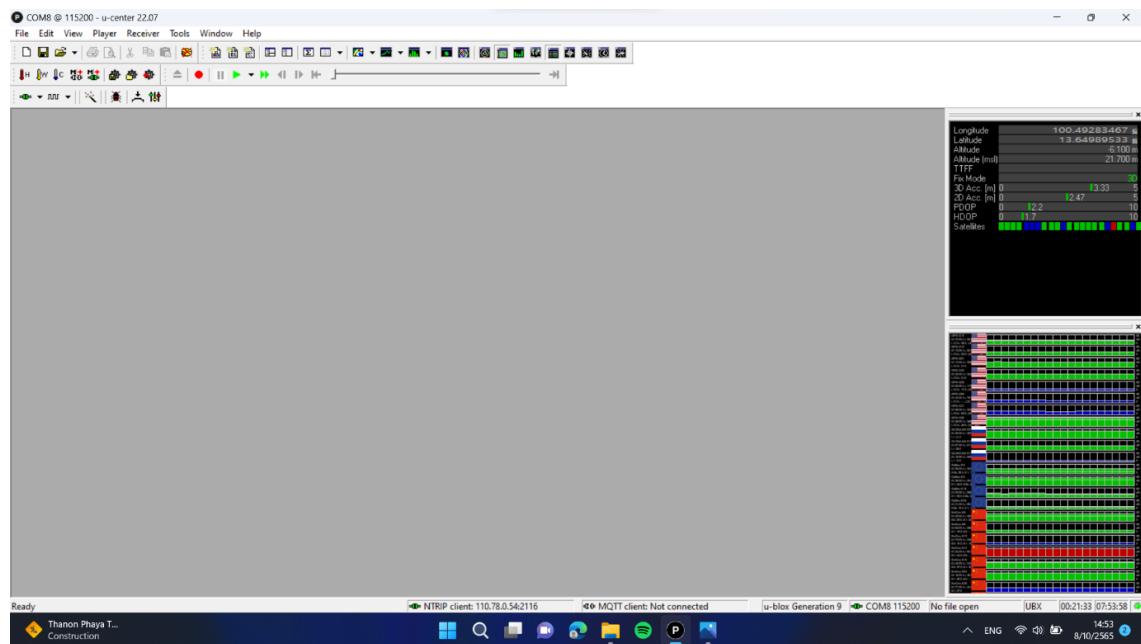
รูปที่ 3.76 การเลือกใช้ NTRIP เพื่อรับค่าปรับแก้จาก CORS



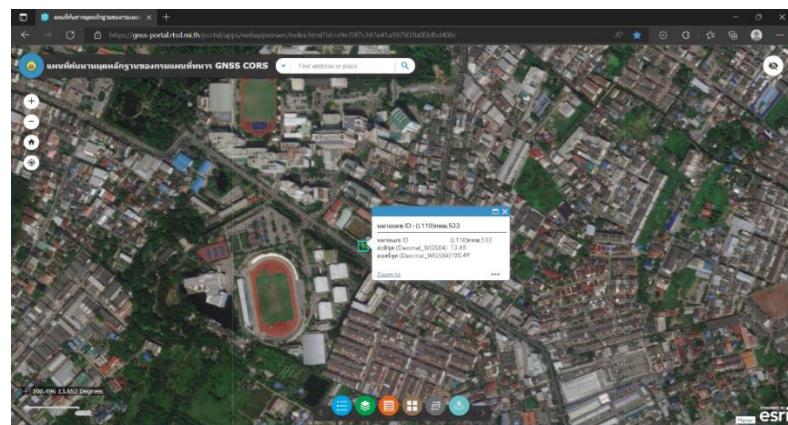
รูปที่ 3.77 ตัวอย่างการเชื่อมต่อ NTRIP ไม่สำเร็จเนื่องจากไม่ได้เชื่อมต่อ Internet



รูปที่ 3.78 การใส่ Username และ Password ที่ขอรับจากการที่ดิน



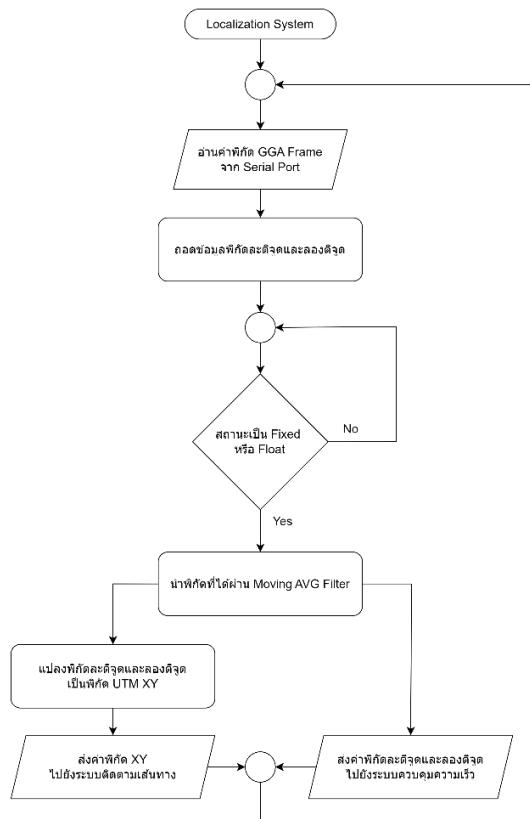
รูปที่ 3.79 สถานะ NTRIP Client จะเปลี่ยนเป็นสีเขียวหลังจากเชื่อมต่อสำเร็จ



รูปที่ 3.80 Continuous Operation Reference Station บริเวณใกล้เคียงกับมหาวิทยาลัย

### 3.2.3.2 การออกแบบซอฟต์แวร์การอ่านค่าพิกัดจาก U-blox ZED-F9P

เป็นการออกแบบการเขียนโปรแกรมด้วยภาษาไฟทอนเพื่อ อ่านค่าพิกัดและติดตาม GNSS เช่นเชอร์เพื่อนำไปประมวลผลสร้างเส้นทางอ้างอิงและระบุพิกัดตำแหน่งปัจจุบันของรถ



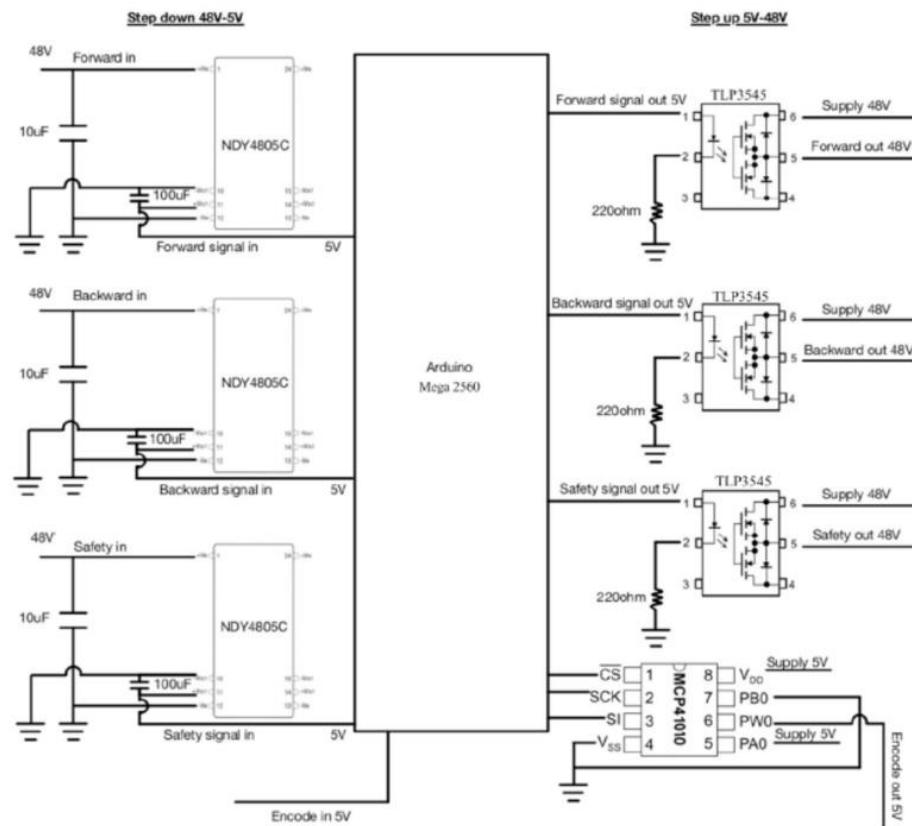
รูปที่ 3.81 ผังงานการอ่านค่าพิกัดและส่งค่าพิกัด ไปยังระบบต่างๆ

จากรูปที่ 3.81 คือผังงานที่แสดงลำดับการทำงานของซอฟต์แวร์ในการอ่านค่าพิกัดจาก GNSS เชนเซอร์ เพื่อส่งไปยังระบบอื่นๆ ผ่านในรถ เริ่มจากการอ่านค่าพิกัด NMEA GGA Frame จาก U-blox ZED-F9P ผ่าน Serial Port จากนั้นจะทำการถอดข้อมูลจาก Data Frame เพื่อนำค่าพิกัดละติจูด, ลองติจูด, เวลาปัจจุบัน และค่า GPS Quality Indicator เพื่อกำหนดเงื่อนไขการทำงานของระบบว่าหาก GPS Quality Indicator ไม่มีสถานะเป็น FIXED หรือ Float จะไม่มีค่าส่งไปยังระบบอื่นๆ เพื่อป้องกันการส่งข้อมูลพิกัดที่มีความคลาดเคลื่อนสูงไปยังระบบอื่นๆ เมื่อได้พิกัดละติจูดและลองติจูดจาก Data Frame ที่มีสถานะ FIXED หรือ Float แล้วจะนำค่าพิกัดที่ได้ไปผ่าน Moving Average Filter อีกหนึ่งครั้งเพื่อป้องกันกรณีที่ค่าพิกัดกระโดดเปลี่ยนแปลงไปอย่างพื้นพลัน จากนั้นเมื่อได้ค่าพิกัดละติจูดและลองติจูดที่มีความละเอียดและถูกต้องแล้วจะส่งค่าไปยังระบบควบคุมความเร็วของรถเพื่อนำพิกัดละติจูดและลองติจูดที่ได้ไปคำนวณหาความเร็วปัจจุบันของรถต่อไป และในขณะเดียวกันจะแปลงค่าพิกัดละติจูดและลองติจูดที่ได้ไปเป็นพิกัด UTM XY และส่งไปยังระบบติดตามเส้นทาง เพื่อติดตามตำแหน่งของรถปัจจุบันว่าอยู่ในเส้นทางอ้างอิงได้และนำไปคำนวณหาค่า CTE ต่อไป

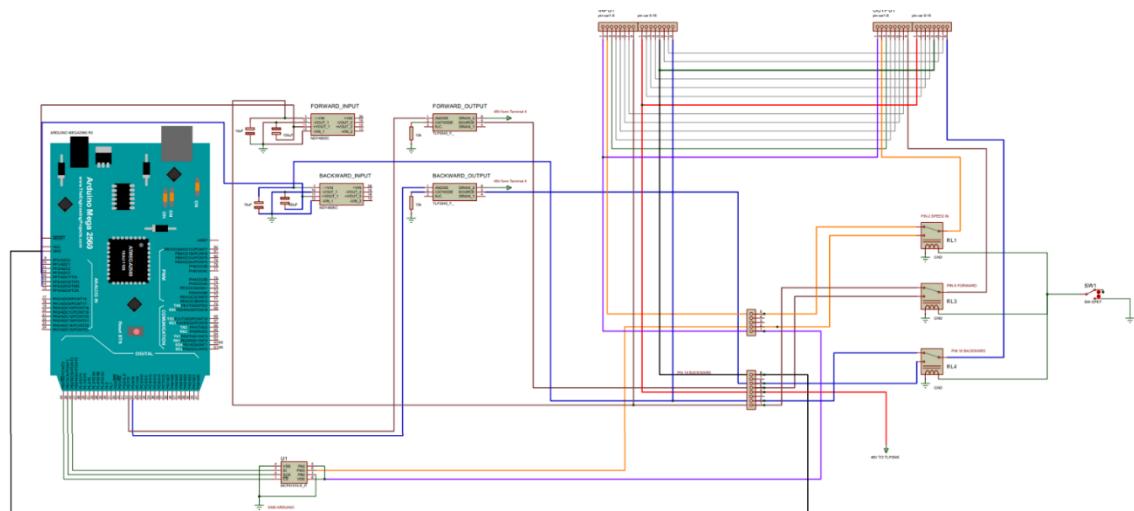
### 3.3 การออกแบบระบบควบคุมการเคลื่อนที่ที่ความเร็วคงที่

#### 3.3.1 การออกแบบฮาร์ดแวร์ของระบบควบคุมความเร็ว

ในการออกแบบฮาร์ดแวร์เพื่อควบคุมความเร็วของการเคลื่อนที่ โดยจะแบ่งการทำงานออกเป็น 2 แบบ คือ แบบแรกเป็นการทำงาน Manual mode คือ จะรับการทำงานของความเร็วรถผ่านการเหยียบคันเร่ง โดยตรง และ แบบที่สอง Auto mode คือการความคุมความเร็วผ่านวงจร Speed Controller Box ที่มี Arduino Mega 2560 เป็นตัวประมวลผลหลักของวงจร



รูปที่ 3.82 แสดง Circuit Diagram ระบบควบคุมความเร็วแบบ Auto



รูปที่ 3.83 แสดง Circuit Diagram ควบคุมความเร็วให้คงที่ในโหมดการทำงานแบบ Auto



รูปที่ 3.84 วงจรควบคุมความเร็วให้คงที่ในโหมดการทำงานแบบ Auto



รูปที่ 3.85 ตัวประมวลผลที่ใช้ในการควบคุมความเร็วให้เคลื่อนที่คงที่

นอกจากนั้นอุปกรณ์ภายในวงจรควบคุมการเคลื่อนที่ด้วยความเร็วคงที่นั้นยังประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ เช่น

## 1. Relay Switch

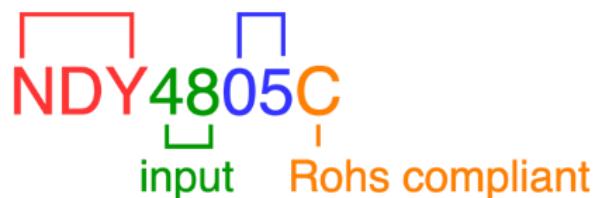
เป็นอุปกรณ์ที่ใช้ในการเปลี่ยนลักษณะการทำงานระหว่างการทำงานแบบ Auto หรือการทำงานแบบ Manual



รูปที่ 3.86 ตัวอย่างอุปกรณ์ Relay

## 2. NDY4805C

เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการปรับแต่งขนาดแรงดันที่ให้มีขนาดลดลงตามอุปกรณ์ตัวที่เลือก เช่น NDY4805C อุปกรณ์ตัวนี้จะเปลี่ยนแรงดันจาก 48 โวลต์ ที่เป็นสัญญาณขาเข้าให้เป็นสัญญาณ 5 โวลต์



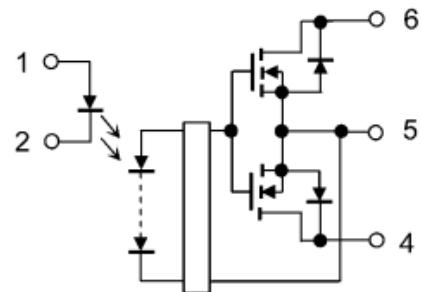
รูปที่ 3.87 ลักษณะการใช้งานของอุปกรณ์



รูปที่ 3.88 ตัวอย่างอุปกรณ์อิเล็กทรอนิกส์ NDY4805C

### 3. TLP3545

เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการปรับแต่งแรงดัน มีการทำงานแบบ Photo relay โดยจะมีหลักการทำงานคือ แบ่งการทำงานส่วนของ Control Circuit กับ ส่วนของ Power Circuit โดยส่วนของ Control Circuit จะต่อเข้ากับ Arduino Mega และเมื่อมีกระแสแรงดัน 5V จาก Arduino ให้เข้า Pin 1 Led จะทำงานส่งผลให้วงจรส่วน Power Circuit ครบวงจร ทำให้แรงดันจาก แบตเตอรี่รถกอล์ฟ 48V ที่ต่อ กับ Pin 6 สามารถให้ผ่านออกที่ Pin 5 ดังนั้นแรงดันขาออกที่ผ่าน TLP3545 ได้มีขนาด 48V สามารถนำไปใช้ในการขับเคลื่อนมอเตอร์



รูปที่ 3.89 ลักษณะการทำงานของอุปกรณ์อิเล็กทรอนิกส์ TLP3545



รูปที่ 3.90 ตัวอย่างอุปกรณ์อิเล็กทรอนิกส์ TLP3545

#### 4. MCP41010

เป็นอุปกรณ์อิเล็กทรอนิกส์ประเภท Digital Potentiometer ที่มีลักษณะคล้ายตัว้านทานปรับค่าได้ ซึ่งสามารถเขียนโปรแกรมเลือกขนาดความต้านทานผ่านบัส SPI หรือ I2C ตัวอย่างการทำงาน เช่น ใช้โปรแกรม Arduino ควบคุมแรงดันขาออกจาก MCP41010 สามารถปรับได้ตั้งแต่ 0 ถึง 255 หากปรับมากกว่า 255 ผลลัพธ์แรงดันที่ออกมีความผิดพลาด



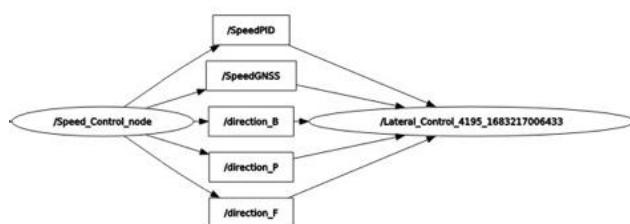
รูปที่ 3.91 ตัวอย่างอุปกรณ์อิเล็กทรอนิกส์ MCP41010

#### 3.3.2 การออกแบบซอฟต์แวร์การควบคุมการเคลื่อนอัตโนมัติตัวยความเร็วคงที่

เป็นการเขียนโปรแกรมด้วยภาษา C ผ่านโปรแกรม Arduino IDE เพื่อทำการควบคุมการเคลื่อนที่ตามความเร็วที่ต้องการ

##### 3.3.2.1 การออกแบบซอฟต์แวร์การรับ - ส่ง ข้อมูลจาก Arduino กับ Node อื่นๆ

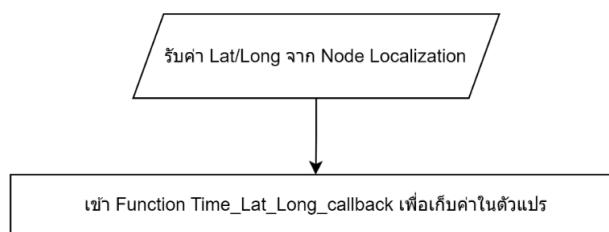
เป็นการเขียนโปรแกรมเพื่อรับ และส่งข้อมูล โดยเริ่มจากการสร้าง Node ที่ควบคุมการทำงานของ Arduino (Node Speed Control) หน้าที่ของ Node Speed Control นอกจากควบคุมการทำงานของความเร็วรถให้คงที่ ยังเป็นตัว Subscriber ค่าความเร็วที่รับมาจาก Velocity Calculation Node เพื่อนำมาแสดงผลการควบคุมความเร็วแบบ Open Loop และ นำมาระบบควบคุมความเร็วแบบ Close Loop



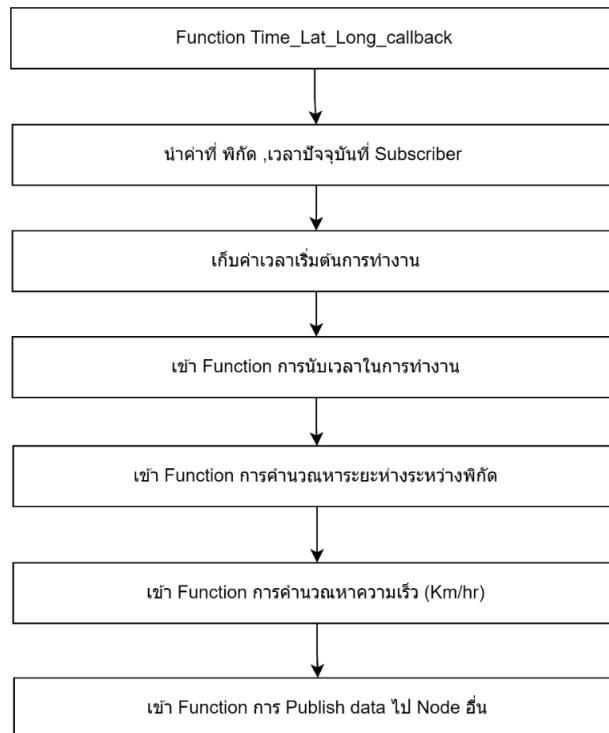
รูปที่ 3.92 ตัวอย่างการรับ-ส่งข้อมูลจาก Arduino กับ Node Lateral Control

### 3.3.2.2 การออกแบบซอฟต์แวร์การอ่านค่าความเร็วจาก Lat/Long data

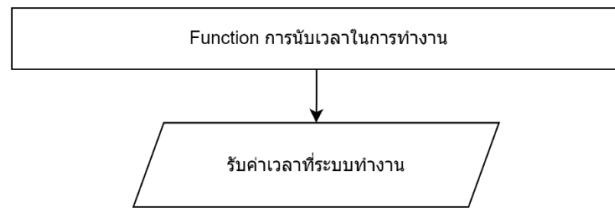
เป็นการเขียนโปรแกรมบนระบบการ Ubuntu เนื่องจากมีการรับและส่งข้อมูลจาก Node ที่อยู่ในโอดจะเป็นการเขียนโปรแกรมด้วยภาษา Python ในโปรแกรม VS CODE เพื่ออ่านค่าความเร็วจากระบบพิกัด เป็นการรับค่า Lat/Long ที่ได้จาก Node Localization ซึ่งในโปรแกรมด้านในโปรแกรมจะมี Moving average filter เป็นการกรองสัญญาณให้มีความนิ่งของสัญญาณมากขึ้น (เป็นการกรองสัญญาณแบบ Realtime) จากนั้นนำข้อมูลที่ได้เข้าสมการ Haversine เพื่อคำนวณค่าเป็นความเร็ว (km/hr)



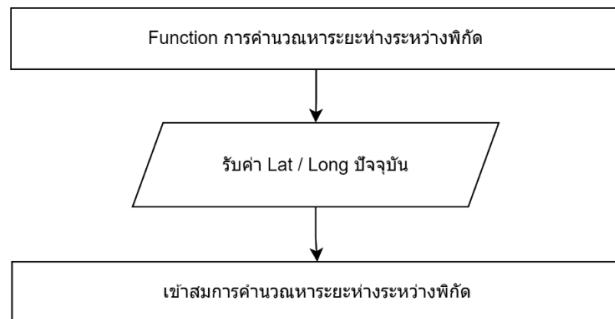
รูปที่ 3.93 ผังการออกแบบซอฟต์แวร์การคำนวณค่าความเร็วจากระบบพิกัด



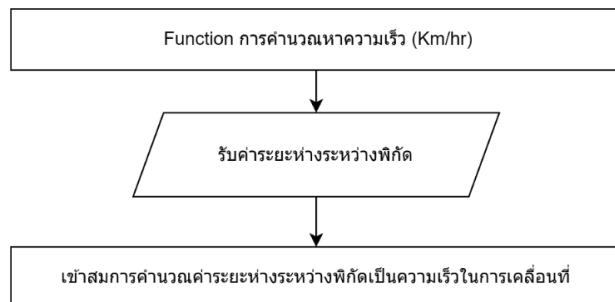
รูปที่ 3.94 ผังการออกแบบซอฟต์แวร์ Function การรับค่า Lat/Long



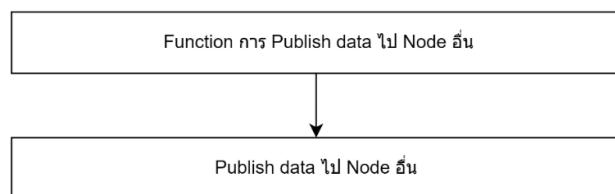
รูปที่ 3.95 ผังการออกแบบซอฟต์แวร์ Function การนับเวลา



รูปที่ 3.96 ผังการออกแบบซอฟต์แวร์ Function การคำนวณระยะห่างระหว่างพิกัด



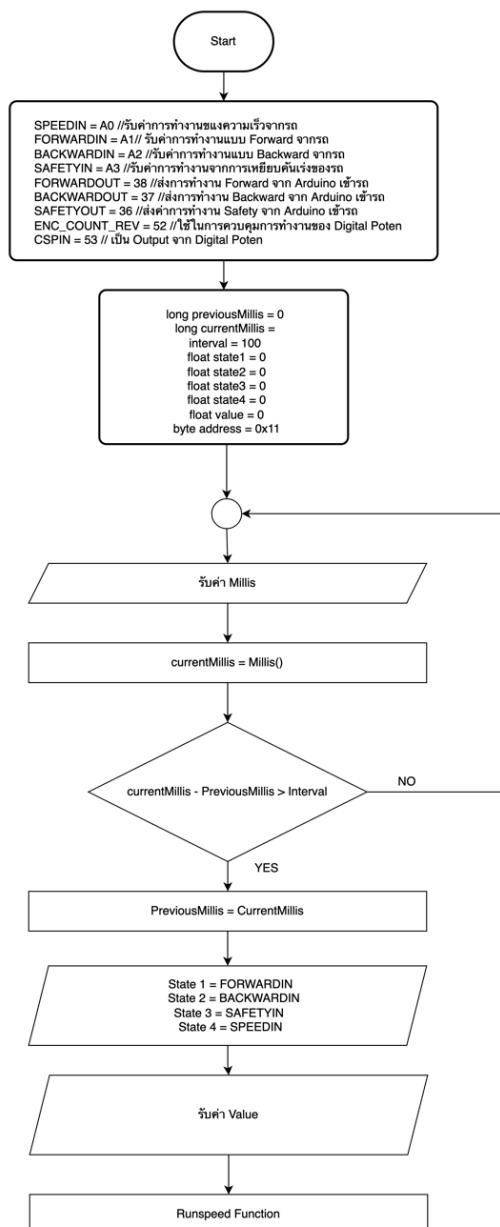
รูปที่ 3.97 ผังการออกแบบซอฟต์แวร์ Function การคำนวณความเร็ว



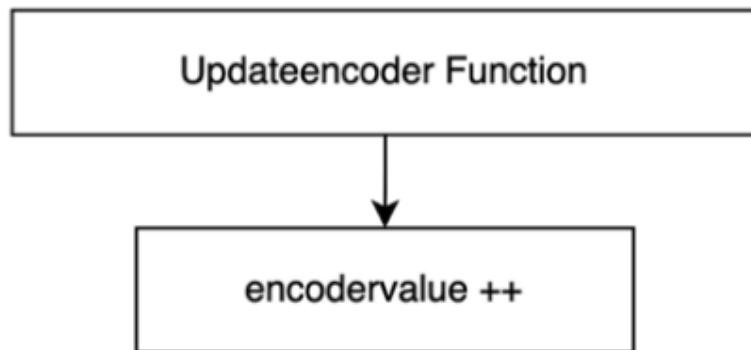
รูปที่ 3.98 ผังการออกแบบซอฟต์แวร์ Function การส่งค่าไป Node อื่น

### 3.3.2.2 ผังการควบคุมความเร็วของรถโดยใช้ Digital Potentiometer

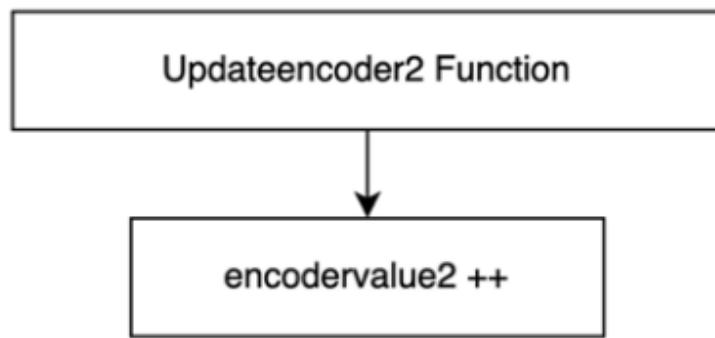
การทำงานของ Digital Potentiometer เป็นการควบคุมความเร็วแบบ Open Loop สามารถปรับค่าตั้งแต่ 0 ถึง 255 โดยจะทำงานควบคุมผ่านการเขียนโปรแกรมเพื่อควบคุมแรงดันเข้า Controller รถ Pin ที่ควบคุมการทำงานของความเร็ว นอกจากนี้ภายในระบบบันทึกมีการรับค่าจาก Node Velocity Calculation เพื่อนำความเร็วในการเคลื่อนที่มาแสดงผล



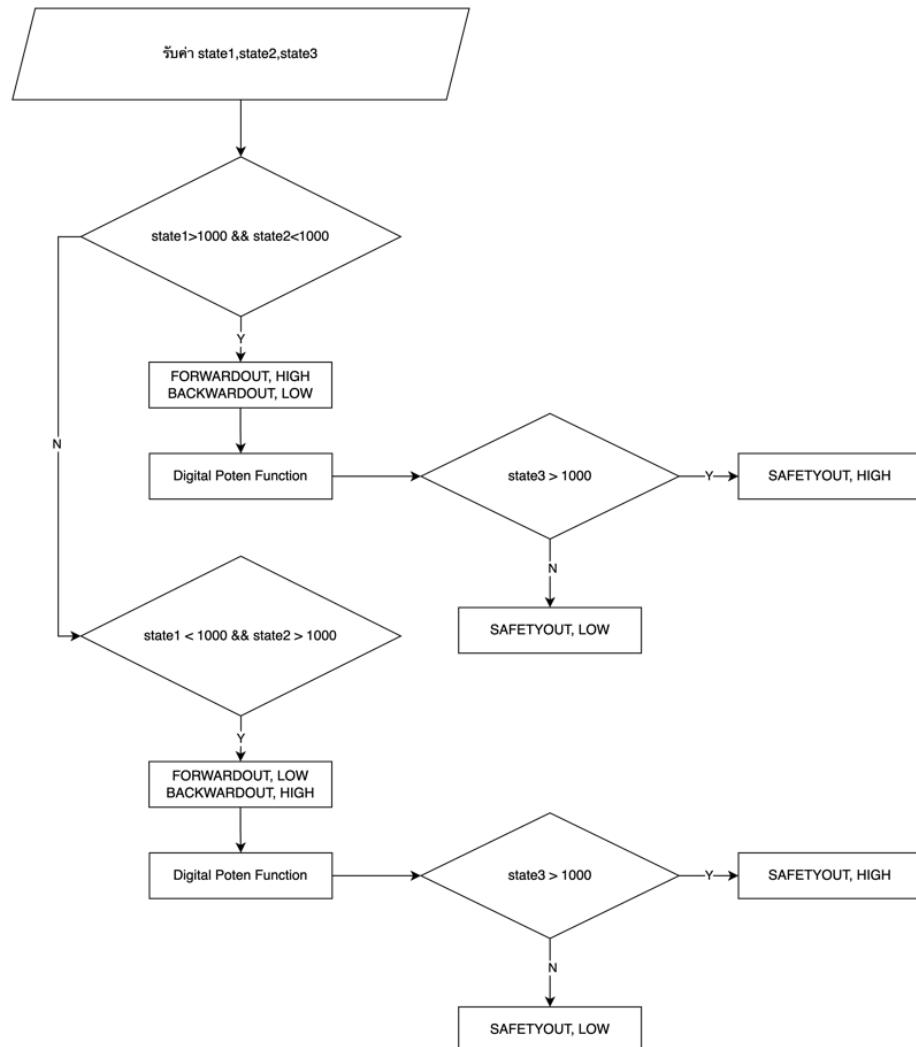
รูปที่ 3.99 แสดงผังการทำงานการควบคุมการเคลื่อนที่ของรถโดยการปรับค่า Digital Potentiometer



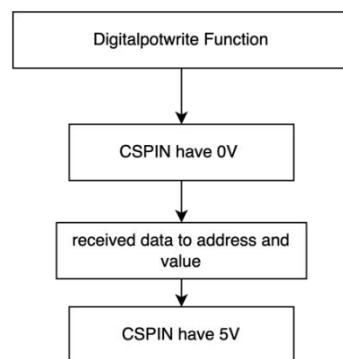
รูปที่ 3.100 ผังการทำงานของ Function นับ Encoder1



รูปที่ 3.101 ผังการทำงานของ Function นับ Encoder2



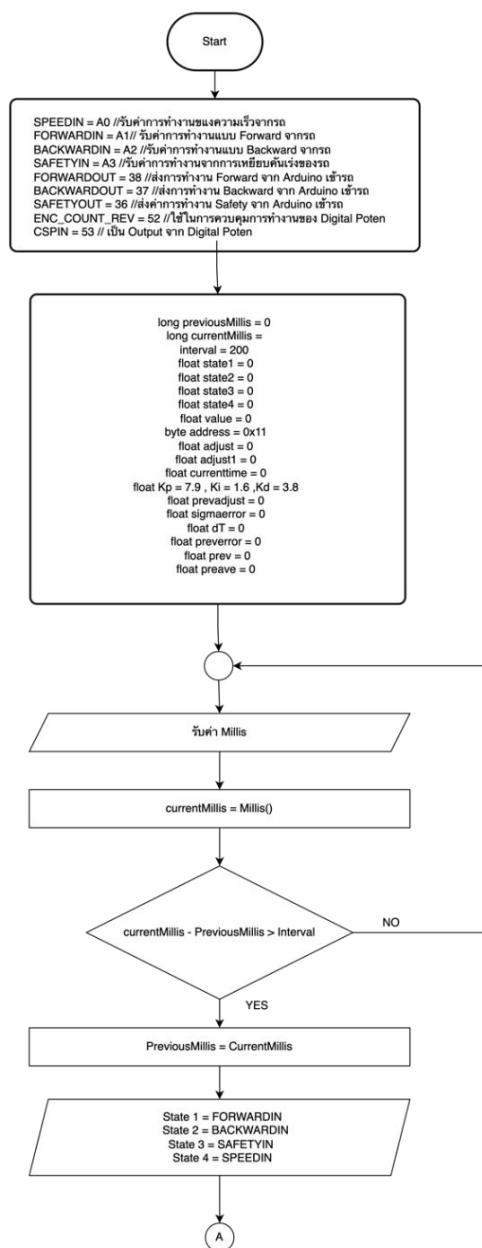
รูปที่ 3.102 ผังการทำงานของ Function ควบคุมการเคลื่อนที่

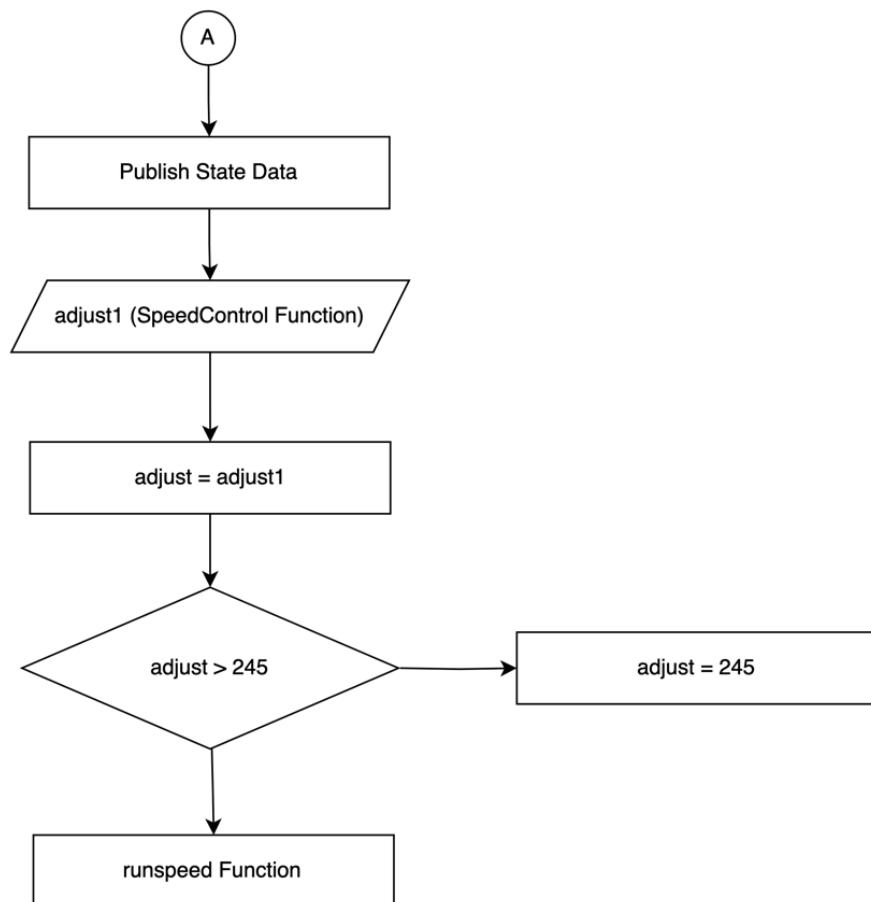


รูปที่ 3.103 ผังการทำงานของ Function ควบคุมการทำงานของ Digital Potentiometer

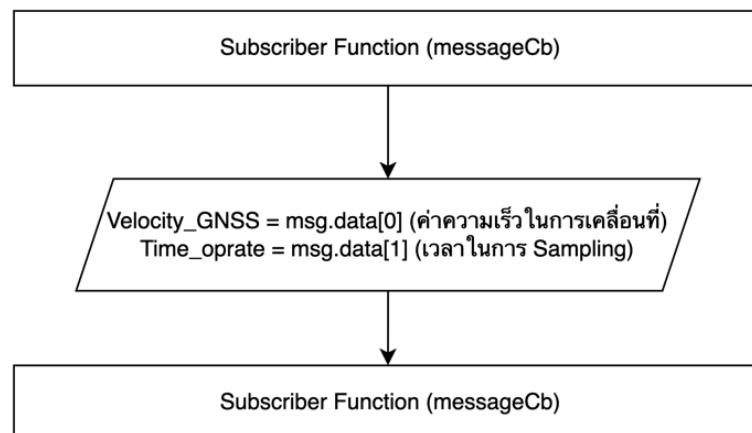
### 3.3.2.2 ผังการควบคุมความเร็วของรถแบบระบบปิด (Close Loop)

การควบคุมความเร็วรถแบบระบบปิด หรือ การควบคุมโดยใช้สมการ PID Controller ซึ่งค่าที่ออกมานั้นสามารถจะเป็นค่าที่เข้าไปควบคุมการทำงาน Digital Potentiometer อีกที เพื่อให้ได้ความเร็วคงที่ที่ต้องการ นอกจากนี้ยังมีการรับค่าความเร็วในการเคลื่อนที่มาจาก Node Velocity Calculation เพื่อนำความเร็วในการเคลื่อนที่มาแสดงผล และ เพื่อนำค่าความเร็วปัจจุบันมาคำนวณหาค่า Error เพื่อให้ในสมการ PID Controller

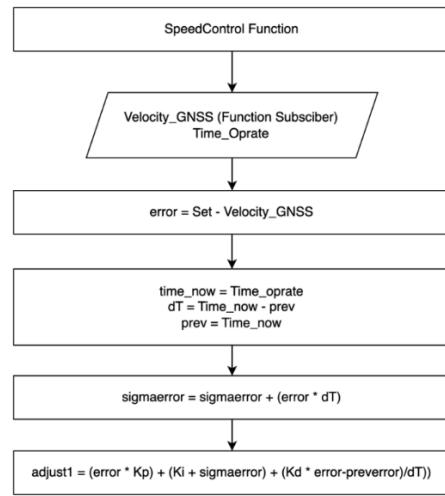




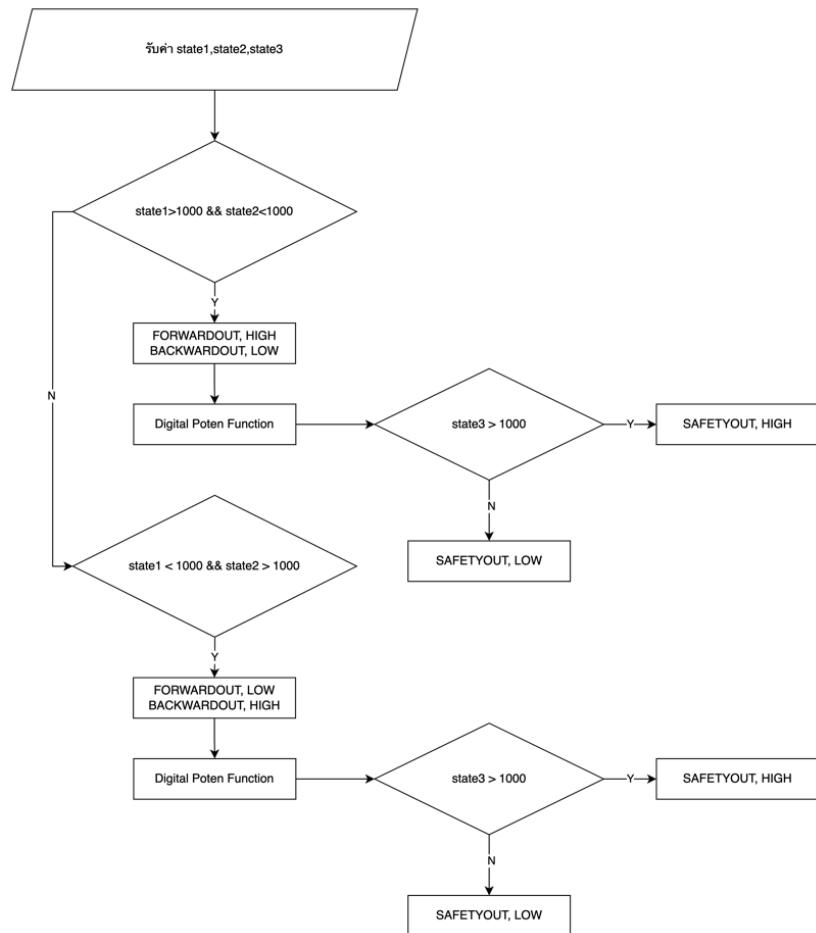
รูปที่ 3.104 ผังการควบคุมความเร็วรถแบบระบบปิด



รูปที่ 3.105 ผังการทำงานของ Function การรับข้อมูลจาก Node อื่น



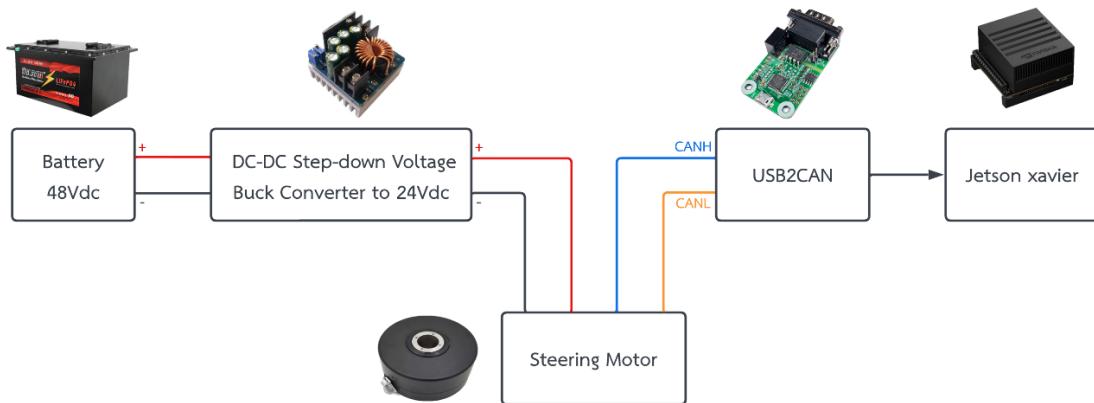
รูปที่ 3.106 ผังการทำงานของ Function การควบคุมความเร็วผ่าน PID Controller



รูปที่ 3.107 ผังการทำงานของ Function การควบคุมการเคลื่อนที่ของรถ

### 3.4 การออกแบบฮาร์ดแวร์และซอฟต์แวร์ของระบบควบคุมทิศทางและบังคับมุนได้ยิwa

#### 3.4.1 การออกแบบวงจรของมอเตอร์พวงมาลัย



รูปที่ 3.108 Circuit Design for Steering Motor Control

มอเตอร์พวงมาลัยสามารถใช้ไฟฟ้ากระแสตรงในการควบคุมได้ด้วยแต่ 7-32 โวลต์ แต่ในการควบคุมสำหรับการติดตามวิถีการขับเคลื่อนรถก็ต้องมีตัวจ่ายไฟเพียง 24 โวลต์ซึ่งเพียงพอต่อการทำงานในส่วนพวงมาลัย โดยตัวจ่ายกระแสไฟหลักจะมาจากการแปลงไฟจากตัวรถ 48 โวลต์ และแปลงไฟโดยใช้ DC-DC Step-down Voltage Buck Converter สำหรับลดแรงดันไฟฟ้าให้เหลือเพียง 24 โวลต์ เพื่อจ่ายไฟให้มอเตอร์พวงมาลัย และเชื่อมต่อผ่าน USB2CAN Module สำหรับสื่อสารข้อมูลระหว่างมอเตอร์กับ Jetson xavier

#### 3.4.2 การสั่งงานมอเตอร์ควบคุมพวงมาลัยผ่านโปรแกรม KEYA ELECTRON

สำหรับการควบคุมรถอัตโนมัติให้สามารถติดตามเส้นทางการเคลื่อนที่ได้ตามพิกัดอ้างอิงจีพีเอสนั้น จะต้องใช้มอเตอร์สำหรับบังคับมุนได้ยิwaของรถ ซึ่งในงานวิจัยนี้จะใช้มอเตอร์รีห้อ KEYA ซึ่งเป็นมอเตอร์สำหรับการควบคุมพวงมาลัยโดยเฉพาะ โดยในการสั่งงานมอเตอร์ดังกล่าวเน้นในงานวิจัยเดิมได้ใช้การเชื่อมต่อสื่อสารด้วย RS232 โดยการเชื่อมต่อดังกล่าวจะมีลักษณะการส่งข้อมูลสื่อสารที่ต่างกัน ซึ่งหากเป็นการเชื่อมต่อด้วย RS232 จะมีการส่งข้อมูลเป็น Frame Header, Enable Status, Speed/Position data, Check code โดยลักษณะการส่งข้อมูลจะมีลักษณะดังรูปที่ 3.109

ลักษณะของการส่งข้อมูลไปยังมอเตอร์เมื่อมีการเชื่อมต่อที่ต่างกัน รหัสหรือข้อมูลที่ส่งไปนั้นก็จะต่างกัน สังเกตได้จากภาพการอ่านสถานะควบคุม

- Read controller control status

The software send: ED 00 00 00 00 00 00 00

Controller feedback: ED 00 64 10 00 00

The controller control status: Electric steering motor encoder, RS232 control, speed mode

6	4	1	0
1. Incremental encoder 2. Hall 3. Magnetic encoder 4. Absolute position encoder 5. Rotary encoder 6. Steering motor incremental encoder	1. AIN 2. CAN 4. RS232 8. RC	1. Speed mode 2. Position mode 3. Torque mode	

รูปที่ 3.109 การควบคุมมอเตอร์ด้วยการเชื่อมต่อแบบต่างๆ

เมื่อทำการเชื่อมต่อมอเตอร์ให้สามารถทำงานตามที่สั่งการได้แล้ว ต้องทำการเขียนโค้ดแปลงค่า Position หรือค่ามุมเลี้ยวจากเลขฐาน 10 เป็นเลขฐาน 16 เพื่อให้ผู้ใช้งานสามารถส่งค่าเข้าสู่มอเตอร์นั้นเข้าใจมากขึ้น เมื่อเปรียบเทียบกับการส่งค่าเป็นเลขฐาน 16 โดยที่ไม่มีการแปลงเลขฐาน และหลังจากที่สามารถอินพุตค่าองศาการหมุนของมอเตอร์ได้แล้ว ก็จะเป็นขั้นตอนในส่วนของการเขียนโปรแกรมควบคุม ให้สามารถคำนวนมุมเลี้ยวและส่งค่าไปยังมอเตอร์ให้ถูกต้องตามพิกัดอ้างอิงที่ได้มีการสร้างขึ้น

#### 3.4.2.1 การตั้งค่ามอเตอร์เพื่อควบคุมพวงมาลัยในการสื่อสารผ่าน RS232

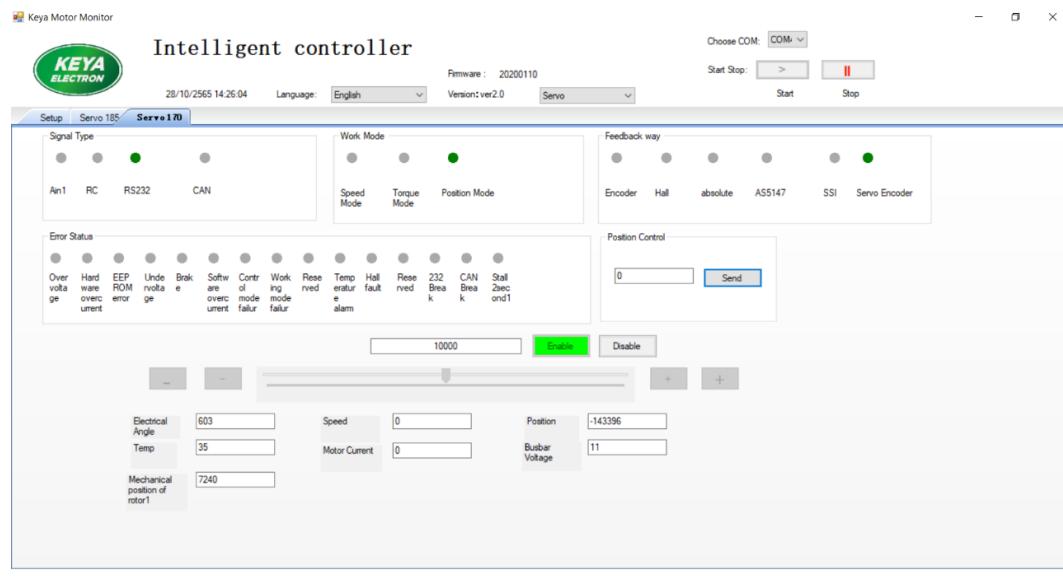
การกำหนดค่าสำหรับซอฟต์แวร์สามารถตั้งค่าพารามิเตอร์ตัวควบคุมซอฟต์แวร์ได้ โดยซอฟต์แวร์จะสื่อสารกับการควบคุมผ่าน RS232 และอัตรา Baud rate ที่ 115200 บิต โดยมีรายละเอียดขั้นตอนการทำงานและพร้อมใช้งานเรียบร้อยตามภาพ โดยเริ่มจากการติดตั้งซอฟต์แวร์ดังรูป จนกระทั่งการสั่งงานมอเตอร์ควบคุมพวงมาลัยผ่าน KEYA ELECTRON สามารถสั่งงานได้ดังรูปที่ 3.111

ซอฟต์แวร์สามารถตั้งค่าพารามิเตอร์ตัวควบคุมได้ ซอฟต์แวร์ได้รับการพัฒนาภายใต้สภาพแวดล้อม .NET ระบบ XP ต้องมี NET 4.0 ติดตั้งซอฟต์แวร์ 4.1.2 ซึ่งในขั้นตอนการติดตั้งซอฟต์แวร์เพื่อใช้งานโปรแกรม KEYA ELECTRON จะใช้ในการควบคุมและสั่งงานพวงมาลัยและทำหน้าที่เป็นตัวควบคุมพวงมาลัย จากภาพจะเห็นได้ว่าโปรแกรมดังกล่าวอยู่ในกรอบสีแดงดังรูปที่ 3.111 แสดงการติดตั้งโปรแกรม kyMotor Control Utility

名称	修改日期	类型	大小
Languages	2020/7/20 13:11	文件夹	
DevComponents.DotNetBar2.dll	2015/10/6 22:27	应用程序扩展	5,344 KB
info	2020/7/27 9:34	配置设置	1 KB
<b>kyMotor Control Utility</b>	<b>2020/7/25 15:17</b>	<b>应用程序</b>	<b>393 KB</b>
kyMotor Control Utility.exe.Config	2020/7/27 9:34	CONFIG 文件	1 KB
kyMotor Control Utility.pdb	2020/7/25 15:17	PDB 文件	284 KB
kyMotor Control Utility.vhost	2020/7/25 15:18	应用程序	12 KB
kyMotor Control Utility.vhost.exe.Co...	2020/7/20 9:20	CONFIG 文件	1 KB
kyMotor Control Utility.vhost.exe.ma...	2012/6/6 2:06	MANIFEST 文件	1 KB

รูปที่ 3.110 การติดตั้งโปรแกรม kyMotor Control Utility

ในขั้นตอนนี้จะทำการดำเนินการคลิก Start หากการสื่อสารข้อมูลสำเร็จนั้น อินเทอร์เฟซจะอ่านการควบคุมพารามิเตอร์ในขณะที่ LED ที่ด้านบนซ้ายจะกระพริบเป็นสีเขียวแสดงว่าพารามิเตอร์นั้นสื่อสารได้สำเร็จ



รูปที่ 3.111 แสดงสถานการณ์สื่อสารสำเร็จ

สำหรับขั้นตอนถัดไปจะเป็นการเปิดอินเทอร์เฟซการกำหนดค่าแล้วดำเนินการคลิกปุ่ม (Connect) ที่มุมล่างซ้าย หลังจากที่เราได้ดำเนินการป้อนค่าแต่ละอย่างเป็นไปตามรูปที่ 3.113 ระบบจะรู้แล้วเพื่อสร้างการเชื่อมต่อระหว่างซอฟต์แวร์และคอนโทรลเลอร์



รูปที่ 3.112 แสดงการเชื่อมต่อระหว่างตัวควบคุมกับซอฟต์แวร์

แรม (RAM) ในกรอบสีแดงสามารถแก้ไขได้ ส่วนด้านซ้ายของกรอบสีแดงคือตัวควบคุมพารามิเตอร์ และด้านขวาของกรอบสีแดงนั้นคือข้อมูลใน E ของรอม (ROM) ในกรณีที่การตั้งค่าต้นฉบับต้องข้อมูลทั้งสาม จะมีค่าเท่ากัน เนื่องจากข้อมูลซอฟต์แวร์ถูกสแกนอย่างต่อเนื่อง เมื่อแก้ไขข้อมูลทุกครั้งที่ได้ทำการปรับให้แก้ไขอย่างรวดเร็วหรือทันที แล้วดำเนินการคลิกปุ่ม WRITE เพื่ออธิบายให้เข้าใจมากขึ้นจะทำการยกตัวอย่างให้เห็นภาพมากขึ้น หากต้องการแก้ไขจำนวนบรรทัดจั่วเข้ารหัสจาก 48000ppr เป็น 49000ppr อีก 2 ROM data คือ 48000 ต้องดำเนินการเปลี่ยน 49000 ใน RAM และคลิกปุ่ม “Write” อย่างรวดเร็วทันที เพื่อยืนยันว่า 49000 จะไม่เปลี่ยนแปลงอีกต่อไป ขั้นตอนเดียวกันสำหรับพารามิเตอร์อื่น พารามิเตอร์หลายตัวสามารถปรับเปลี่ยนไปได้พร้อมๆ กัน ได้ โดยจะดำเนินการคลิกปุ่ม “Program” ที่ด้านล่างขวา โปรแกรมข้อมูลใน RAM เป็น E2ROM และบันทึก โดยขั้นตอนการเปลี่ยนโปรแกรมใช้เวลา 3-5 วินาที จะเห็นได้ว่าปุ่ม “Program” นั้นจะเปลี่ยนเป็นสีแดง และว่าข้อมูลกำลังถูกติดตั้งในโปรแกรม ผู้ใช้งานจะต้องรอและสังเกตข้อมูลจนกว่าจะมีการแจ้งเตือนว่า “โปรแกรมสำเร็จ” จากนั้นข้อมูลทั้งสาม (RAM, ROM, Param) ในบล็อกสีน้ำเงินจะสอดคล้องกัน ซึ่งบ่งชี้ว่าข้อมูล ROM ได้รับการตั้งค่าโปรแกรมไว้ในคอนโทรลเลอร์ ซึ่ง ณ จุดนี้ การปรับเปลี่ยนพารามิเตอร์ควบคุมเสร็จสมบูรณ์แล้ว จากนั้นให้ดำเนินการคลิกปุ่ม “ยกเลิกการเชื่อมต่อ” และคลิกที่ปุ่ม “Exit” เพื่อสิ้นสุดขั้นตอนก่อนหน้านี้ แล้วให้ทำการปิดเครื่องคอมพิวเตอร์อีกครั้ง (สิ่งที่ต้องคำนึงถึงไม่ว่า Configuration จะถูกแก้ไขหรือไม่ก็ตาม ต้องปิดเครื่องและรีเซ็ตเพื่อเริ่มต้นตามปกติทุกครั้ง)



รูปที่ 3.113 แสดงวิธีการ WRITE ข้อมูลหลังจากการปรับตั้งค่าทุกครั้ง

เมื่อตั้งค่าโปรแกรมการกำหนดค่าสำหรับควบคุมมอเตอร์ สามารถบันทึกการกำหนดค่าที่แก้ไขลงในไฟล์ จากนั้นทำการอ่านค่าจากไฟล์ เพื่อดาวน์โหลดไปยังมอเตอร์ได้ในครั้งต่อไป



รูปที่ 3.114 แสดงการอัปโหลดไฟล์ที่มีการตั้งค่าเอาไว้ก่อนหน้า

การตั้งค่าและป้อนค่าจากที่อธิบายมาทั้งหมดก่อนหน้านี้ดังรูปนี้ ค่าที่ป้อนไปนั้นพารามิเตอร์ฟังก์ชันแต่ละเลขมีคำอธิบายดังรูป

**0000** Identifier. when the system is connected, identify the software communication or serial port control. (Don't need modify)

**0001** The number of motor poles (this motor is 32 poles)

**0002** Rated motor speed (set to 80)

**0003** Motor maximum current (Don't need modify)

**0004** The number of encoder lines, the standard is 48000

**0005** Kp parameter of controller current loop PI control (typical value 0.2)  
Can be modified appropriately.

**0006** Ki parameter of controller current loop PI control (typical value 0.05)  
Can be modified appropriately.

**0007** Kp parameter of controller speed loop PI control (typical value 0.8)  
Can be modified appropriately.

**0008** Ki parameter of controller speed loop PI control (typical value 0.05)  
Can be modified appropriately.

**0009-0012** Position loop PID control parameters

**0013** Acceleration time. "5" means the acceleration time from 0rpm to rated speed is 0.5s.

**0015** Zero position compensation of magnetic encoder

**0016** Zero position compensation of rotary encoder

**0018** Controller system address, or node number of control.  
This parameter is used in the CAN, CANOpen, and EtherCAT buses.

For example: set the data to 1, then the ID in CAN bus: 0x0600000 + controller system address, it will be (0x06000001)

**0019** Control signal selection

**0020** Control mode selection, including speed control, position control  
1.Speed control,  
3.Absolute position control, refer to the CAN bus protocol  
4.Relative position control, refer to the CAN bus protocol

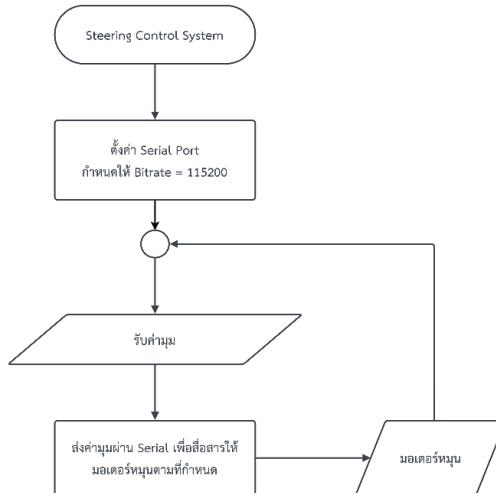
**0021** CAN bus baud rate selection (Factory setting is 250K)  
1.125k 2.250k 3.500k 4. 1M

**0022** Position sensor selection  
1. Incremental encoder  
2. Hall  
3. Magnetic encoder  
4. SSI absolute position encoder  
5. Rotary encoder  
6. Steering motor incremental encoder

Other parameters: spare

รูปที่ 3.115 คำอธิบายแต่ละพารามิเตอร์ฟังก์ชัน

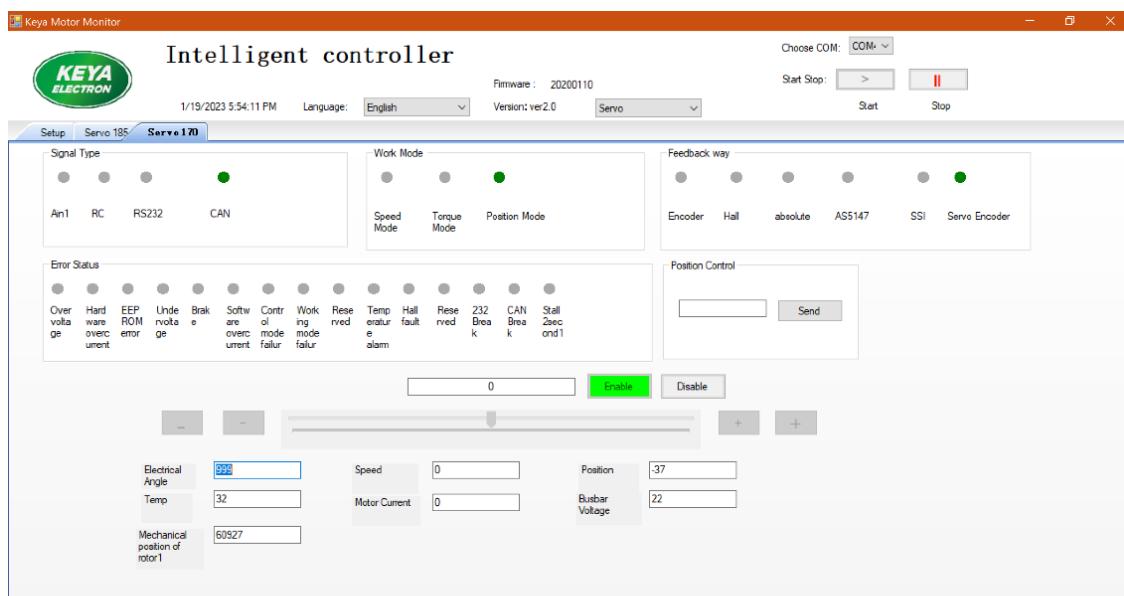
การทดสอบการทำงานของมอเตอร์ผ่าน RS232 เพื่อทดสอบการบังคับมุมเลี้ยวของมอเตอร์



รูปที่ 3.116 ผังงานตรวจสอบการทำงานของมอเตอร์ในการบังคับมุมเลี้ยว

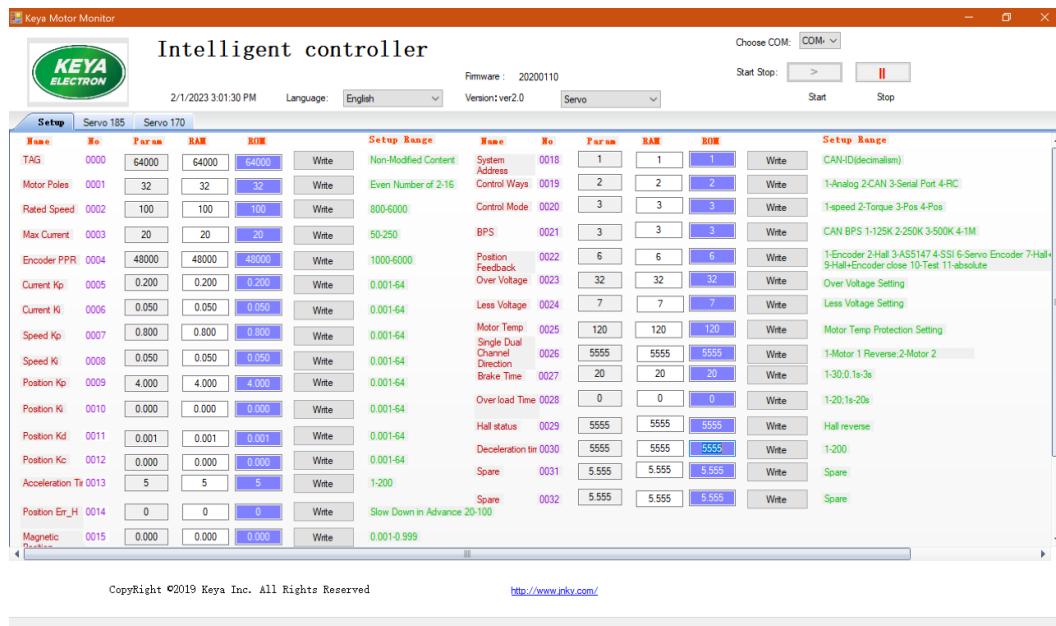
### 3.4.2.2 การตั้งค่ามอเตอร์ เพื่อควบคุมพวงมาลัยในการสื่อสารผ่าน CAN

#### 3.4.2.2.1 การตั้งค่ามอเตอร์ผ่าน KEYA ELECTRON



รูปที่ 3.117 แสดงสถานการณ์สื่อสารสำหรับ

ในขั้นตอนนี้จะทำการดำเนินการคลิก Start หากการสื่อสารข้อมูลสำเร็จนั้น อินเทอร์เฟซจะอ่านการควบคุมพารามิเตอร์ในขณะที่ LED ที่ด้านบนข้างจะกระพริบเป็นสีเขียวแสดงว่าพารามิเตอร์นั้นสื่อสารได้สำเร็จ



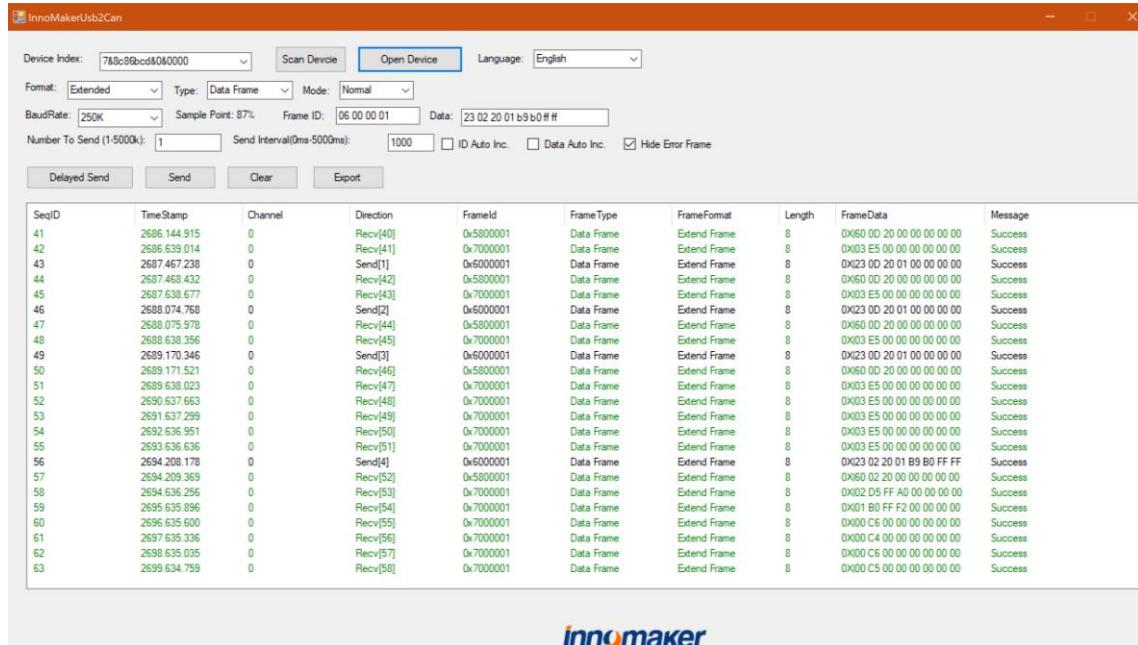
รูปที่ 3.118 แสดงการเชื่อมต่อระหว่างตัวควบคุมกับซอฟต์แวร์

ตั้งค่าพารามิเตอร์โดยเขียนที่ RAM ในช่อง No. 0018 (System address or node number of control) เป็น 1 ตั้งค่า RAM ในช่อง No. 0019 (Control Ways) เป็น 2 (CAN) ตั้งค่า RAM ในช่อง No. 0020 (BPS) เป็น 2 (250K) ตามลำดับเพื่อปรับค่าพารามิเตอร์เป็นการสื่อสารแบบ CAN

### 3.4.2.2.2 เชื่อมต่อ CANBUS เพื่อสั่งงานหรือรับค่ากับมอเตอร์พวงมาลัยผ่านโปรแกรม InnoMakerUSB2CAN



รูปที่ 3.119 InnoMakerUSB2CAN Software



รูปที่ 3.120 แสดงการรับส่งข้อมูลผ่านโปรแกรม InnomakerUSB2CAN

โดยการสั่งข้อมูลผ่านการสื่อสารแบบ CAN โดยทั่วไปนั้น ควรตั้งค่า baud rate เป็น 250Kb ตั้งค่า format เป็น extended ID ควรส่งรูปแบบข้อมูลจากต่ำไปสูง โดยอยู่ในรูปฐานสิบหก ต้องทำการ Enable ผ่าน Data: 0x23 0x0D 0x20 0x01 0x00 0x00 0x00 ผ่าน Frame ID: 0x06000001 และวิธีส่งข้อมูลที่ต้องการสื่อสารผ่าน Data: 0x23 0x02 0x20 0x01 DATA\_HI DATA\_Hh DATA\_L1 DATA\_Lh โดยในรูปที่ xx ทำการส่งข้อมูลสื่อสารเพื่อควบคุมมุมเลี้ยวของวงมาตรฐานเดียวที่ Data: 0x23 0x02 0x20 0x01 0xB9 0xB0 0xFF ซึ่งก็คือตำแหน่ง  $0xB9B0_{hex} = 47536_{dec}$  และ  $0xFFFF_{hex} = 65535_{dec}$  บอกทิศทางการหมุนคือทิศตามเข็ม (Clockwise) หากเป็นรูปแบบ CANOpen ข้อมูลจะเป็น quary mode และเป็นข้อมูล fixed heartbeat และส่งข้อมูลที่เกี่ยวข้อง มีการตรวจสอบช่วงปิดของสัญญาณ 1000 ms (คำสั่งความเร็วจะถูกส่งไปต่อเนื่องในช่วงเวลาที่ยกเว้น 1000 ms การส่งคืน Quary data เป็นข้อมูลเลขฐานสิบหก (Hexadecimal data) ซึ่งจำเป็นต้องแปลงเป็นข้อมูลเลขฐานสิบ (Decimal data)

## 1. การใช้คำสั่ง CAN Bus

หมายเหตุ 1: Controller ID เป็นตัวเลข Decimal ในกระบวนการกำหนดค่าผ่านซอฟต์แวร์ และ CAN ID เป็นเลขฐานสิบหก

ตัวอย่างที่ 1: ตั้งค่า Controller ID เป็น  $1_{dec}$  จะได้ CAN ID เป็น 06000001 (extension ID)

ตัวอย่างที่ 2: ตั้งค่า Controller ID เป็น  $112_{dec}$  หรือมีค่าเป็น  $70_{hex}$  จะได้ CAN ID เป็น 06000070

หมายเหตุ 2: ID สำหรับการส่งข้อมูล คือ 0x600000 + ID Controller (Hexadecimal)  
 ID สำหรับข้อมูลที่ส่งคืน คือ 0x0580000 + ID Controller (Hexadecimal)  
 ID สำหรับข้อมูล Heartbeat คือ 0x0700000 + ID Controller (Hexadecimal)

**การเปิดใช้งาน (Enable)** จะใช้ data เป็น 23 0d 20 01 00 00 00 00

Return ID: 0x0580000 + controller ID (hexadecimal)

Data 60 0d 20 00 00 00 00 00

**การปิดใช้งาน (Disable)** จะใช้ data เป็น 23 0c 20 01 00 00 00 00

Return ID: 0x0580000 + controller ID (hexadecimal)

Data 60 0c 20 00 00 00 00 00

**การค้นหาข้อมูลผลัด (Fault quary)** จะใช้ data เป็น 40 12 21 01 00 00 00 00

Return ID: 0x0580000 + controller ID (hexadecimal)

Data 60 12 21 01 DAT1 DAT2 00 00

## 2. คำอธิบายข้อมูล CAN bus



รูปที่ 3.121 Data frame of Control mode

DataBox\_MDL = 0x230D2001 Enable

DataBox\_MDL = 0x230C2001 Disable

DataBox\_MDL = 0x23022001 Position control

โดยต้องกำหนดให้ข้อมูลมีจำนวน 8 ตัวเลข เช่น 23 0D 20 01 00 00 00 00

ในโหมด Position Control จะสามารถหมุนไปตามเข็มและทวนเข็มได้เพียง 2 รอบครึ่ง  
 ซึ่งมีค่าคือ -25000 ถึง 25000

### ตัวอย่างการควบคุมผ่าน CAN bus สำหรับ Position Mode

กำหนดค่าให้มีค่าตั้งแต่ -50000 (0x3CB0 FFFF) ถึง 50000 (0xC350 0000) หมายความว่า หมุนตามเข็มหรือทวนเข็ม ได้ 5 รอบ โดยกำหนดการตั้งค่าในซอฟต์แวร์ kyMotor Electron คือ CAN control (No. 0019) เป็น 2 รวมทั้งตั้งค่า absolute position control (No. 0020) เป็น 3 และตั้งค่า System address (No. 0018) เป็น 1

ใช้การควบคุมผ่าน Control Command ID เป็น 0x06000001 (extended ID)

คำสั่งสำหรับส่งข้อมูลมีดังนี้

- 1) Disability 23 0C 20 01 00 00 00 00
- 2) Enable 23 0D 20 01 00 00 00 00
- 3) Position control 23 02 20 01 DATA\_L(h) DATA\_L(l) DATA\_H(h) DATA\_H(l)

ตัวอย่างที่ 1: หากต้องการให้มอเตอร์หมุน 1.8 รอบ ทิศทางเข็มนาฬิกา

- 1) ตรวจสอบให้แน่ใจว่าได้เปลี่ยนการควบคุมตำแหน่งแล้ว
- 2) เปิดใช้งาน (Enable) 23 0D 20 01 00 00 00 00
- 3) Run คำสั่งสำหรับควบคุมตำแหน่ง : 23 02 20 01 B9 B0 FF FF

ตัวอย่างที่ 2: หากต้องการหมุนไปที่มุม 76 องศา ในทิศทวนเข็มนาฬิกา จะคำนวณได้ตามสูตรดังนี้

$$(76 \times (10000 \text{ rpm}/360 \text{ deg}) = 2052_{\text{Dec}} = 0x0804_{\text{Hex}}) \quad (3.1)$$

- 1) ตรวจสอบให้แน่ใจว่าได้เปลี่ยนการควบคุมตำแหน่งแล้ว
- 2) เปิดใช้งาน (Enable) 23 0D 20 01 00 00 00 00
- 3) Run คำสั่งสำหรับควบคุมตำแหน่ง : 23 02 20 01 08 04 00 00

#### 3.4.2.2.3 ติดตั้ง Can-utils (cantools) ใน Ubuntu

```
pond@ck:~$ sudo apt-get install can-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
can-utils is already the newest version (2018.02.0-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
```

รูปที่ 3.122 การติดตั้ง CANtools ใน Ubuntu

คำสั่งสำหรับติดตั้ง can-utils คือ sudo apt -get install can-utils และทำการตั้งค่าพอร์ตโดยผ่านคำสั่ง ifconfig can0

รูปที่ 3.123 การทดสอบการเชื่อมต่อ CAN ในการรับส่งค่า

ทำการตรวจสอบ ip address ว่าพอร์ตที่เราซื้อมต่อนั้นได้ถูกซื้อมต่อแล้วหรือไม่ โดยใช้คำสั่ง: ip addr

ทำการตั้งค่า baud rate ที่ 250Kb สำหรับ can0: sudo ip link set can0 type can bitrate 250000

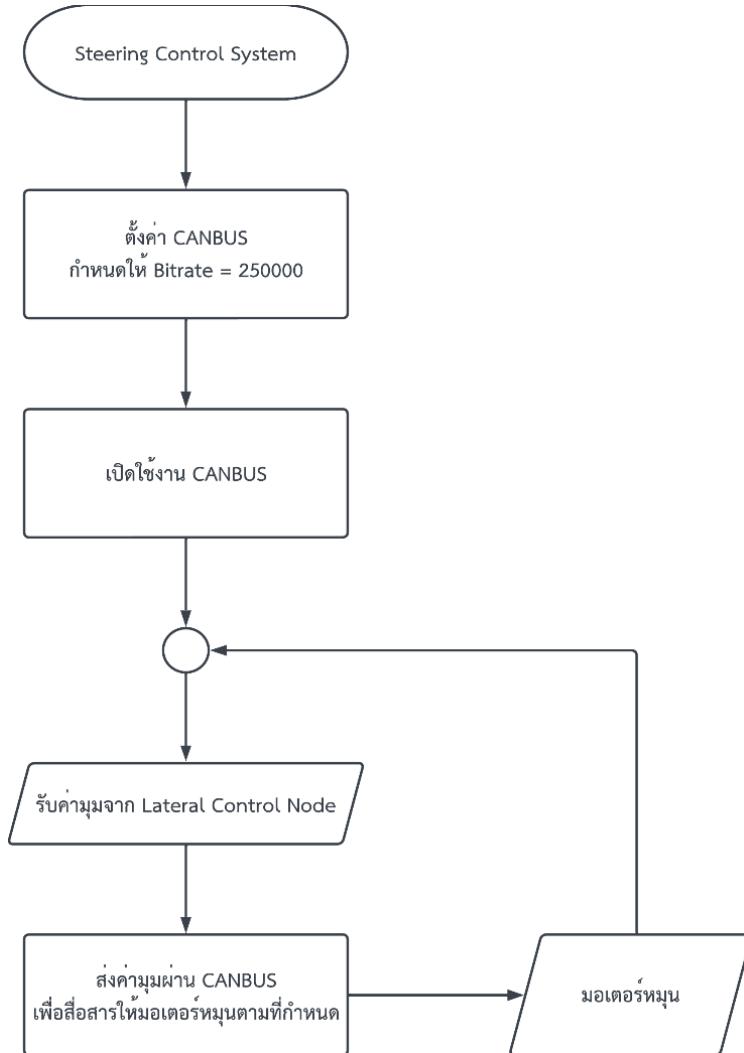
ตั้งให้ can0 ทำงาน: sudo ifconfig can0 up

ทำการตรวจสอบการรับค่าข้อมูลผ่าน คำสั่ง: candump can0

และสามารถส่งค่าข้อมูลเพื่อเปิดใช้งานผ่านคำสั่ง: cansend can0 6000001#230D200100000000

แล้วจึงสั่งให้มอเตอร์หมนตามตำแหน่งที่ต้องการตาม Data

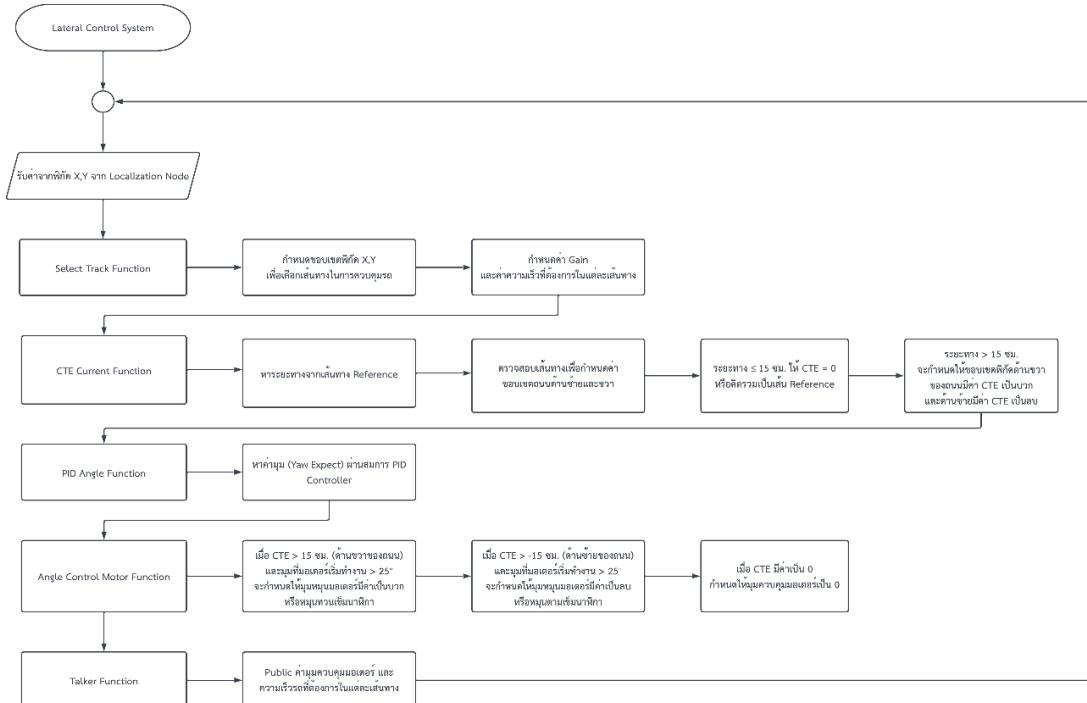
### 3.4.2.2.4 ขั้นตอนการทำงานรับ-ส่งค่าของมอเตอร์ผ่านการสื่อสารแบบ CANBUS



รูปที่ 3.124 ผังงานสำหรับทดสอบการรับ-ส่งค่าจากมอเตอร์

จากรูปที่ 3.124 ผังงานสำหรับทดสอบการรับ-ส่งค่าจากมอเตอร์ เริ่มจากตั้งค่า CANBUS โดยกำหนด Bitrate เป็น 250000 และเปิดใช้งาน CANBUS ผ่านคำสั่ง “sudo ifconfig can0 up” ใน Ubuntu หลังจากนั้นรับค่ามุมจาก Lateral Control Nodes และทำการแปลงค่ามุมตำแหน่งที่ได้รับเป็นองศา ผ่านสมการคำนวณพัลส์ก่อน และจึงแปลงเป็นเลขฐานสิบหกในการส่งค่าเพื่อบังคับมอเตอร์ให้หมุนไปยังตำแหน่งที่ต้องการ

### 3.4.3 ผังงานการทำงานของการควบคุมมุมเลี้ยวผ่าน Lateral Control Node



รูปที่ 3.125 ผังงานแสดงขั้นตอนการทำงานของ Lateral Control Node

จากรูปที่ 3.125 ผังงานแสดงขั้นตอนการทำงานของ Lateral Control Node จะเริ่มจากรับค่าพิกัด X,Y จาก Localization Node ซึ่งรับสัญญาณจาก GNSS Sensor แล้วมาตรวจสอบเส้นทางในการติดตามวิถีการเคลื่อนที่ของรถอัตโนมัติ ผ่าน Select Track Function โดยจะมีการกำหนดของเขตของถนนแต่ละเส้นในการควบคุม รวมทั้งกำหนดค่า Gain และค่าความเร็วคงที่ในแต่ละเส้นทางด้วย ถ้ามำจะเข้า CTE Current Function จะใช้หาระยะทางตั้งจากจารถถึงเส้นทางอ้างอิง ทำการกำหนดขอบเขตด้านซ้ายและขวาของถนนโดยกำหนดให้ขอบเขตพิกัดด้านขวาของถนนมีค่า CTE เป็นบวก และพิกัดด้านซ้ายของถนนมีค่า CTE เป็นลบ และวิ่งมาเข้า PID Angle Function เพื่อกำหนดค่ามุม (Yaw Expect) ผ่านสมการ PID Controller เพื่อไปควบคุมมอเตอร์พวงมาลัย โดยมุมที่ CTE > -15 ซม. (ด้านซ้ายของถนน) และมุมที่ล้อของมอเตอร์มีค่ามากกว่า 25 องศา จะกำหนดให้มุมมีค่าลบ หมายถึงให้หมุนตามเข็มนาฬิกาเพื่อเข้าสู่เส้นทางอ้างอิงหรือ CTE = 0 และถ้ามุมที่ CTE > 15 ซม. (ด้านขวาของถนน) และมุมที่ล้อของมอเตอร์มีค่ามากกว่า 25 องศา จะกำหนดให้มุมมีค่าบวก หมายถึงให้หมุนทวนเข็มนาฬิกาเพื่อเข้าสู่เส้นทางอ้างอิงหรือ CTE = 0 เช่นกัน

### 3.4.3.1 การคำนวณหาค่า Cross Track Error (CTE)

สำหรับขั้นตอนการคำนวณหาค่า CTE จะอยู่ในฟังก์ชัน CTE Current โดยจะใช้สมการ

$$\text{Cross Track Error (CTE)} = \frac{|Ax+By+C|}{\sqrt{A^2+B^2}} \quad (3.2)$$

$$A = \text{Distance 2 Points} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (3.3)$$

$$\text{Slope} = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad (3.4)$$

$$C = y_2 - (\text{Slope} \times x_2) \quad (3.5)$$

โดยจากสมการ (3.2)  $Ax+By+C = 0$  คือสมการเส้นตรงในรูปทั่วไป

เมื่อ A คือ ค่าคงที่ระหว่างทางระหว่างจุดสองจุด ซึ่งหาได้จากสมการ (3.3)

B คือค่าคงที่ -1 ซึ่งใช้เพื่ออ้างอิงตลอดทุกเส้นทาง

C คือค่าคงที่ซึ่งได้จากการคำนวณตามสมการเส้นตรงในรูปมาตรฐาน ดังสมการ (3.5)

เมื่อคำนวณหาค่า A, B และ C ได้แล้วจึงนำมาคิดรวมในสมการ (g) Cross Track Error (CTE) หรือสมการหาระยะทางระหว่างจุดตั้งจากกับเส้นตรง ซึ่งก็มาจากการถูกปฏิagoรัสในการหาสามาหริ่ยมุมจากนั้นเอง หลังจากคำนวณค่า CTE ได้แล้วจะกำหนดขอบเขตพิกัดฝั่งซ้ายและขวาของถนน โดยใช้พิกัดของถนนเป็นตัวกำหนด แล้วให้สัญลักษณ์เครื่องหมายบอกแทนถนนฝั่งขวา (+CTE) และสัญลักษณ์เครื่องหมายบอกแทนถนนฝั่งซ้าย (-CTE) ซึ่งขั้นตอนนี้จะอยู่ในฟังก์ชัน CTE Current

### 3.4.3.2 การคำนวณ PID Controller เพื่อควบคุมมุมเลี้ยว

PID controller (Proportional-Integral-Derivative controller) เป็นการควบคุมแบบวงปิดที่นำเอาค่าผิดพลาดจากระบบ (Error) คำนวณจากค่าเริ่มต้น (Setpoint) ลบด้วยค่าการตอบสนองของระบบ (Output) มาทำการซดเชยโดยมีตัวแปร P, I และ D ในการปรับค่าสัญญาณที่ป้อนให้กับระบบ โดยที่สามารถออกแบบผลการตอบสนองให้เหมาะสมกับกระบวนการที่ต้องการ การตอบสนองของตัวควบคุมจะอยู่ในรูปของการให้ตัวของตัวควบคุมจนถึงค่าความผิดพลาด ค่าการพุ่งเกิน (Overshoots) และค่าแก้ไขของระบบ (Oscillation)

สัดส่วน (Proportional term, P) คือ อัตราขยายที่เกิดจากการนำค่าพิดพลาดของระบบมาคูณด้วยค่าคงที่  $K_p$

$$P = K_p e(t) \quad (3.6)$$

โดยที่  $P$  คือ สัญญาณขาออกของเทอมสัดส่วน

$K_p$  คือ อัตราขยายสัดส่วน, ตัวแปรปรับค่าได้

$e(t)$  คือ ค่าความผิดพลาด ณ เวลา  $t$

ปริพันธ์ (Integral term, I) คือค่าที่นำผลรวมของความผิดพลาดมาคูณกับอัตราขยาย

$$I = K_i \int_0^t e(t) dt \quad (3.7)$$

โดยที่  $I$  คือ สัญญาณขาออกของเทอมปริพันธ์

$K_i$  คือ อัตราขยายปริพันธ์, ตัวแปรปรับค่าได้

$e(t)$  คือ ค่าความผิดพลาด ณ เวลา  $t$

อนุพันธ์ (Derivative term, D) คือ อัตราการเปลี่ยนแปลงของความผิดพลาดจากกระบวนการหรือความชันของความผิดพลาดของระบบทุกๆ เวลา และคูณด้วยอัตราขยายอนุพันธ์  $K_d$

$$D = K_d \frac{d}{dt} e(t) \quad (3.8)$$

โดยที่  $D$  คือ สัญญาณขาออกของเทอมอนุพันธ์

$K_d$  คือ อัตราขยายอนุพันธ์, ตัวแปรปรับค่าได้

$e(t)$  คือ ค่าความผิดพลาด ณ เวลา  $t$

สัญญาณข้อกของ การควบคุมแบบ PID เมื่อนำสัญญาณข้อกของเทอมสัดส่วนปริพันธ์ และอนุพันธ์ จะได้สัญญาณข้อกของ การควบคุมแบบ PID แทนด้วย  $u(t)$

$$u(t) = P + I + D = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{d}{dt} e(t) \quad (3.9)$$

จากสมการข้างต้น  $e(t)$  ก็คือค่า CTE ที่เกิดขึ้น และเอาต์พุต  $u(t)$  ก็คือค่ามุมเอาต์พุตที่จะถูกส่งไปควบคุม มอเตอร์พวงมาลัยในการบังคับมุมเลี้ยว (Steering Control Node) โดยกำหนดให้มุมเมื่ออยู่สี่เหลี่ยมของถนน ให้มีค่าเป็นบวก ซึ่งจะทำให้มอเตอร์หมุนตามเข็มนาฬิกา และเมื่ออยู่สี่เหลี่ยมของถนนให้มีค่าเป็นลบ จะทำให้มอเตอร์หมุนตามเข็มนาฬิกา เพื่อเข้าสู่ค่า CTE = 0 หรือ  $u_m = 0$

### 3.5 การออกแบบซอฟต์แวร์ของระบบ ROS (Robot Operating System)

เนื่องจาก Robot Operating System (ROS) คือ ระบบสำหรับเขียนซอฟต์แวร์และจัดการไฟล์อย่างเป็นระเบียบเพื่อควบคุมการทำงานของหุ่นยนต์ โดยรวมรวมชุดเครื่องมือและชุดคำสั่งต่างๆ ที่จำเป็นในการพัฒนาหุ่นยนต์ ซึ่งจะช่วยลดความยุ่งยากในการสร้างและพัฒนาหุ่นยนต์ที่มีความซับซ้อน ดังนั้นในการพัฒนาซอฟต์แวร์ภายในของรถกอล์ฟไฟฟ้า จะพัฒนาโดยมีระบบส่วนต่างๆภายในรถเป็นระบบต่างๆ ของหุ่นยนต์โดยจะมีระบบหลักๆ ด้วยกัน 3 ระบบ ได้แก่ 1. ระบบระบุตำแหน่งปั๊งจุบันของหุ่นยนต์ 2. ระบบควบคุมความเร็วการเคลื่อนที่ของหุ่นยนต์ และ 3. ระบบควบคุมทิศทางการเคลื่อนที่ของหุ่นยนต์

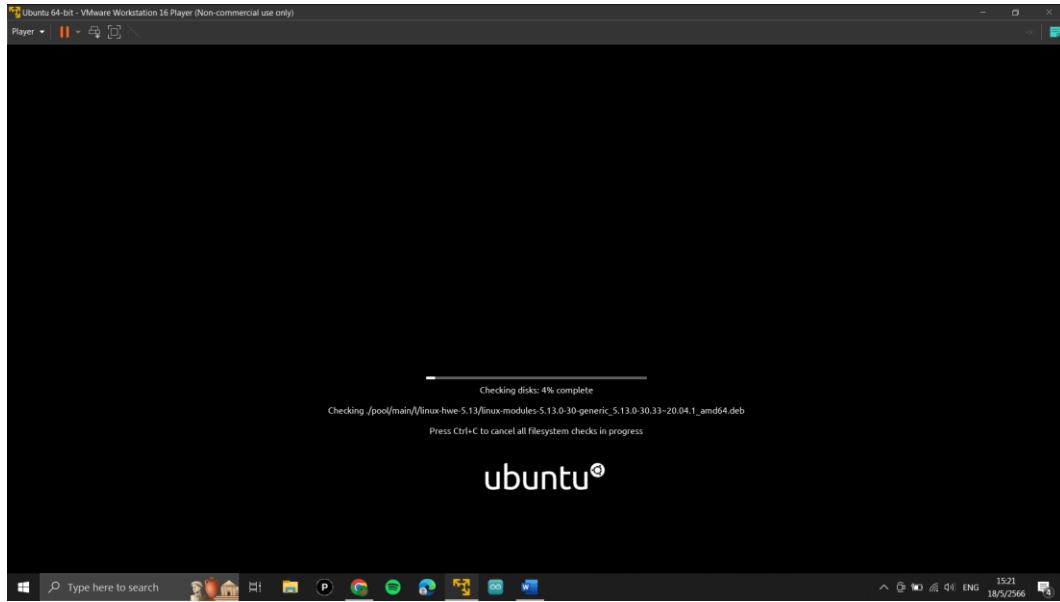
#### 3.5.1 ขั้นตอนการติดตั้ง Robot Operating System

- ตรวจสอบเวอร์ชัน Ubuntu ที่รองรับการทำงานของเวอร์ชัน ROS ที่ต้องการ

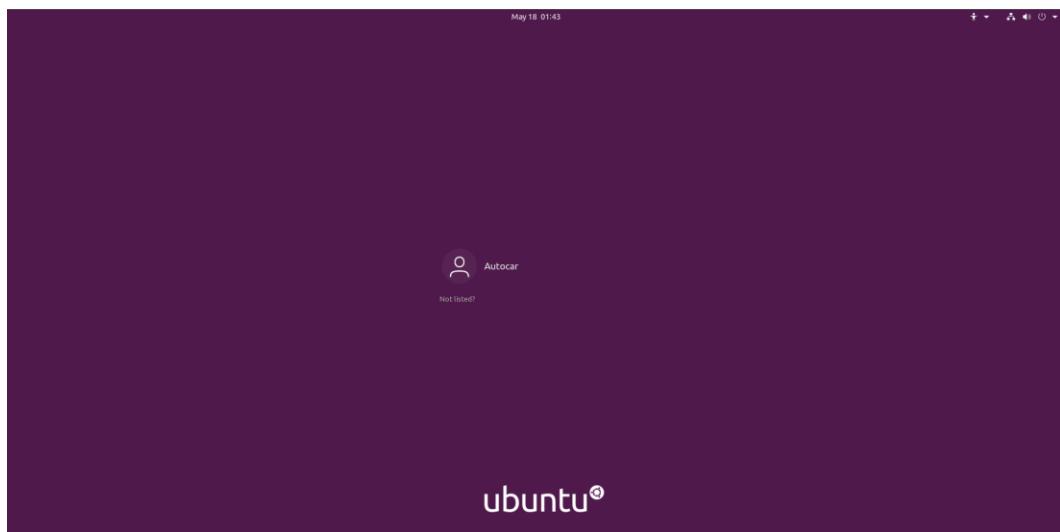
Distro	Release date	Poster	EDL date
Lunar Logperifeed	May, 2017		May, 2019
Kinetic Kame	May 23, 2016		2021-05-30
jade	May 23, 2015		2017-05-16
Indigo	July 22, 2014		2018-04-30
Hydro	September 4, 2013		2014-05-31
Groovy Galapagos	December 31, 2012		2014-07-31
Fuerte Turtle	April 23, 2012		
Electric Emys	August 30, 2011		
Diamondback	March 2, 2011		
C Turtle	August 2, 2010		
Box Turtle	March 2, 2010		

รูปที่ 3.126 แสดงเวอร์ชัน Ros

## 2. การติดตั้ง Ubuntu 20.04



รูปที่ 3.127 การติดตั้ง Ubuntu



รูปที่ 3.128 การติดตั้ง Ubuntu สีดำเริ่ม

### 3. การติดตั้ง Ros ซึ่งเป็นการติดตั้งตาม Official Wiki ของ Ros

#### 3.1 จัดเตรียม Advanced Package Tool (APT) ให้รองรับ Ros

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/roslatest.list'
```

#### 3.2 จัดเตรียม Key ในการติดตั้ง Ros

```
sudo apt install curl # if you haven't already installed curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

#### 3.3 Update Package List ก่อนที่จะเริ่มติดตั้ง Ros

```
sudo apt update
```

#### 3.4 ติดตั้ง Ros noetic

```
sudo apt install ros-noetic-desktop-full
```

#### 3.5 การเตรียม Environment bash เพื่อแสดง Command อัตโนมัติทุกครั้งที่มีการเปิด Terminal ใหม่ โดยที่ไม่ต้องใช้คำสั่ง source /opt/ros/noetic/setup.bash ทุกครั้งที่มีการเปิด Terminal ใหม่

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

### 3.6 เปิดใช้งาน rosdep เพื่อให้การติดตั้งความต้องการของ System Source ง่ายขึ้น

#### 3.6.1 การติดตั้งเครื่องมือเพื่อช่วยในการสร้าง Package Ros

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
```

#### 3.6.2 การติดตั้ง Python3 เพื่อช่วยให้ระบบทำงานได้ง่ายขึ้น

```
sudo apt install python3-rosdep
```

#### 3.6.3 สามารถเริ่มต้นการใช้ rosdep

```
sudo rosdep init
rosdep update
```

### 3.7 การติดตั้ง Ros noetic สำเร็จ

```
sorrawee@ubuntu:~$ roscore
... logging to /home/sorrawee/.ros/log/6a61bc26-f55d-11ed-875a-235b94a06af1/roslaunch-ubuntu-2690.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu:34333/
ros_comm version 1.16.0

SUMMARY
=====

PARAMETERS
* /rosdistro: noetic
* /rosversion: 1.16.0

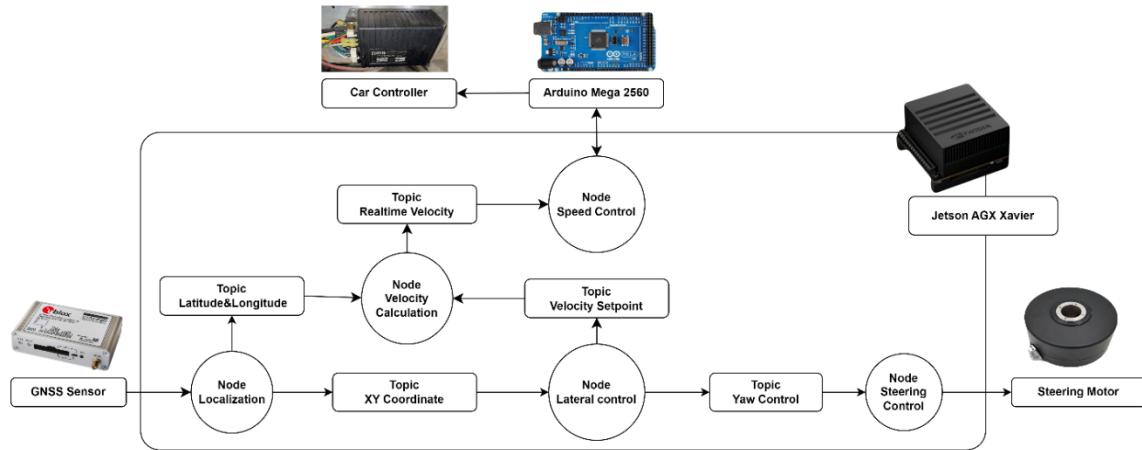
NODES

auto-starting new master
process[master]: started with pid [2704]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to 6a61bc26-f55d-11ed-875a-235b94a06af1
process[rosout-1]: started with pid [2718]
started core service [/rosout]
```

รูปที่ 3.129 การติดตั้ง Ros noetic

### 3.5.1 การออกแบบซอฟต์แวร์ของ Robot Operating System



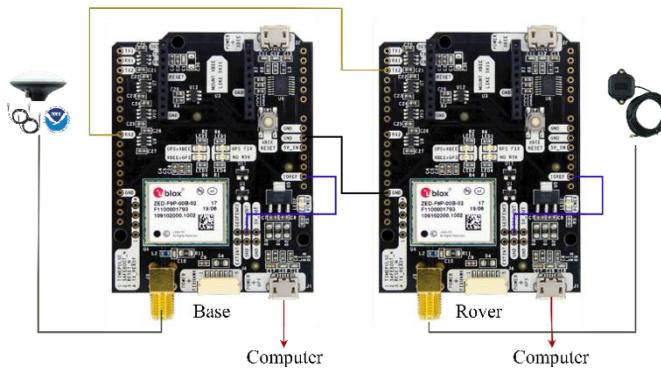
รูปที่ 3.130 ผังงานการทำงานของระบบ ROS

จากรูปที่ 3.130 คือผังงานที่แสดงลำดับการทำงานของซอฟต์แวร์ทั้งหมดผ่านการควบคุมและสื่อสารข้อมูลด้วย Robot Operating System เริ่มจากอ่านค่าพิกัดจาก GNSS Sensor เข้าไปยัง Localization Node จากนั้นภายในจะทำการถอด Data Frame และจะส่งพิกัดละติจูดและลองติจูดไปยัง Velocity Calculation Node เพื่อทำการคำนวณความเร็วของรถจากค่าพิกัดและส่งค่าไปยัง Speed Control Node ที่เชื่อมต่ออยู่กับ Arduino Mega เพื่อนำความเร็วที่คำนวณได้ไปควบคุมความเร็วของรถต่อไป และในขณะเดียวกัน Localization Node จะทำการส่งค่าพิกัด UTM XY ไปยัง Lateral Control Node เพื่อนำไปติดตามเส้นทาง และคำนวณค่า Cross Track Error และมุ่งเลี้ยวของพวงมาลัยเพื่อส่งไปยัง Steering Control Node ต่อไป อีกทั้ง Lateral Control Node ยังมีหน้าที่ในการส่งค่าความเร็วช่วงต่างๆของเส้นทางข้าง翁ที่ต้องการไปยัง Velocity Calculation Node เพื่อส่งค่าไปยัง Arduino ที่เป็นตัวประมวลผลหลักในการควบคุมความเร็วอีกด้วย

## บทที่ 4 ผลการทดลอง

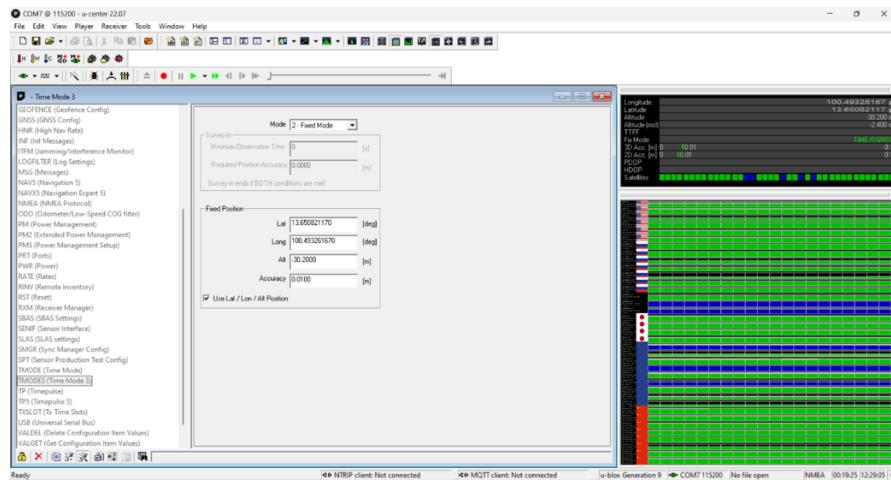
### 4.1 ผลการทดลองของระบบระบุตำแหน่ง

#### 4.1.1. ผลการทดลองการรับ-ส่งข้อมูลพิกัดผ่านการรังวัดแบบจลน์ (Real Time Kinematic – RTK) แบบใช้สาย



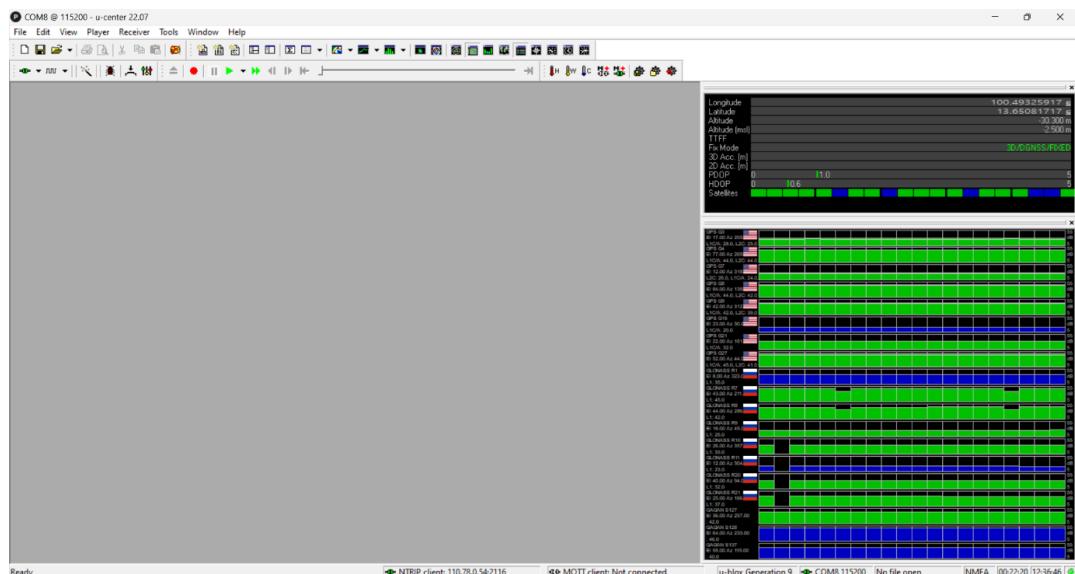
รูปที่ 4.1 การเชื่อมต่อ Base และ Rover แบบใช้สาย

เริ่มด้วยจากการเชื่อมต่อเสา Antenna เข้ากับ Receiver จากนั้นต่อสายจาก IOREF ไปยัง 3V3\_OUT เพื่อเป็นแรงดันอ้างอิงสำหรับ I/O Pin และต่อสาย GND เชื่อมต่อระหว่าง Base Receiver และ Rover Receiver สุดท้ายทำการเชื่อมต่อ Pin Rx2 ของ Base Receiver เข้ากับ Pin Tx2 ของ Rover Receiver เพื่อใช้สำหรับการส่งค่าปรับแก้ระหว่างกัน โดยการเลือกใช้ Pin ในการรับ-ส่งข้อมูลสามารถตั้งค่าได้ผ่านโปรแกรม U-center ซึ่งได้อธิบายในหัวข้อก่อนหน้านี้



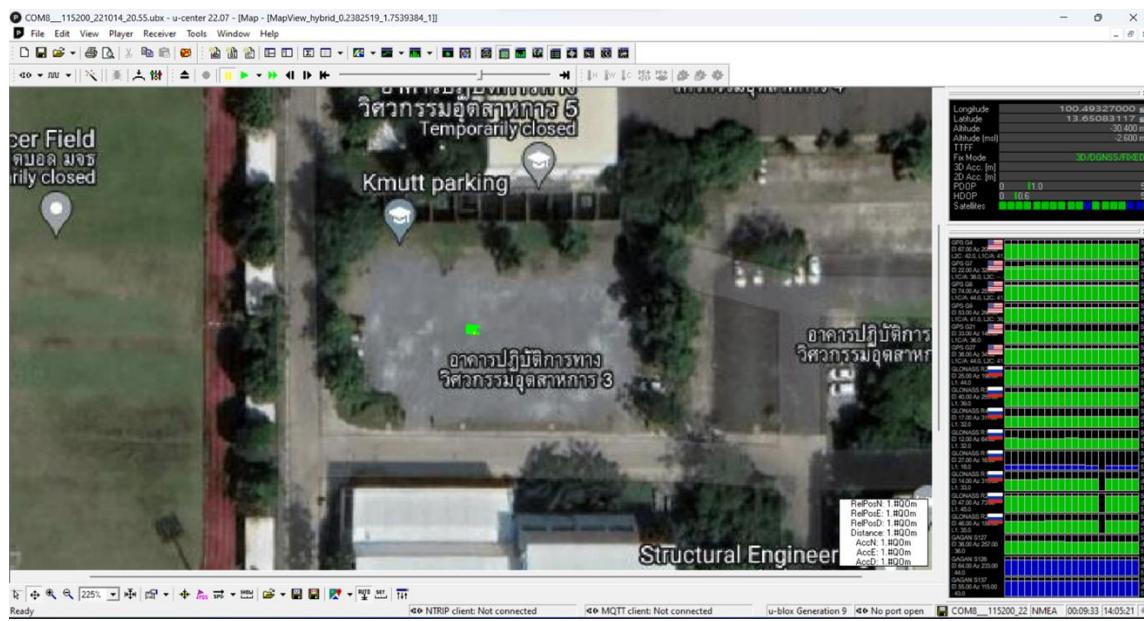
รูปที่ 4.2 การรังวัดหาพิกัดของ Base Station ในแบบ Fixed Mode

กรณีที่ใช้การรังวัดด้วยรูปแบบ RTK หากสถานะของ Base Station ขึ้นสถานะ TIME แสดงว่า Base Station พร้อมส่งค่าปรับแก้ให้กับ Rover Station แล้ว



รูปที่ 4.3 สถานะ FIXED ของ Rover Station

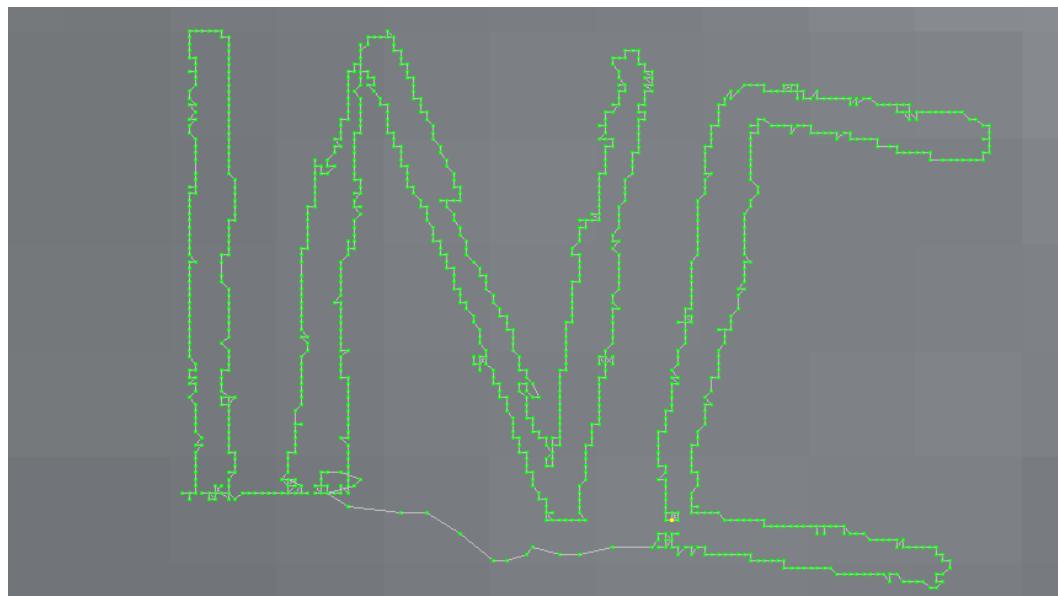
กรณีที่ใช้การรังวัดด้วยรูปแบบ RTK หากสถานะของ Rover Station ขึ้นสถานะ FIXED แสดงว่าการเชื่อมต่อระหว่าง Base Station และ Rover Station เพื่อส่งข้อมูลค่าปรับแก้ระหว่างกัน สามารถใช้งานได้



รูปที่ 4.4 สถานที่ที่ใช้ในการทดสอบ



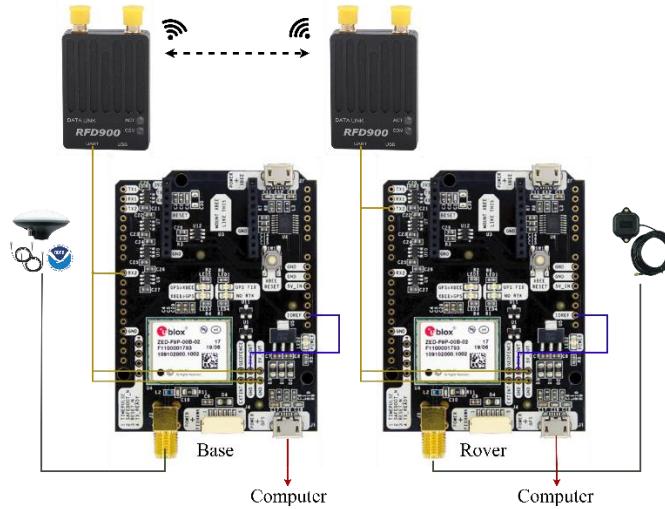
รูปที่ 4.5 รูปแบบตัวอักษรที่ใช้ในการทดสอบ



รูปที่ 4.6 ผลการทดลองที่ได้

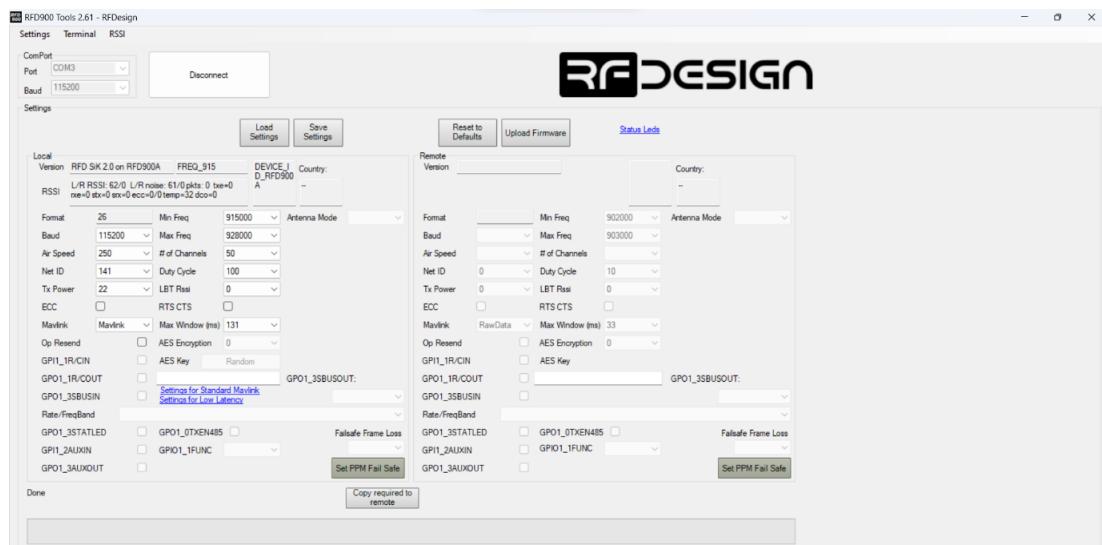
สำหรับการทดลองวัดค่าพิกัดแบบ RTK แบบใช้สาย จะต้องคำนึงถึงความละเอียดในการตรวจค่าพิกัดของ Base Station แบบ FIXED Mode ในความละเอียดอยู่ที่ 1 เซนติเมตร และรูปแบบตัวอักษรที่ใช้ในการทดสอบมีความกว้างของตัวอักษรอยู่ที่ 5 เซนติเมตร เริ่มการทดลองโดยให้ผู้ทำการทดลองทำการเคลื่อนที่เสาอากาศ Antenna ไปบริเวณรอบของตัวอักษร ผลของการทดลองพบว่าลักษณะของพิกัดที่ได้มีความใกล้เคียงกับรูปแบบของตัวอักษร แต่จะมีบางช่วงของตัวอักษรที่มีการคลาดเคลื่อนจากตัวอักษรจริงประมาณ 1-5 เซนติเมตร ทั้งนี้เกิดจากบันทึกของผู้ทดลองทำการเคลื่อนที่เสาอากาศ Antenna บริเวณลำตัวส่วนต่างๆของผู้ทดลอง ได้บดบังการรับสัญญาณบางส่วนทำให้พิกัดที่ได้จึงมีความคลาดเคลื่อน ซึ่งการลดความคลาดเคลื่อนของการตรวจวัดพิกัดสามารถทำได้โดยเคลื่อนที่เสาอากาศ Antenna แบบไม่มีส่วนใดของร่างกายบดบัง หรือการเปลี่ยนลักษณะของการทดลองโดยการติดเสาอากาศ Antenna กับอุปกรณ์อื่นที่สามารถเคลื่อนที่ได้ เช่น รถ หรือยานพาหนะอื่นๆ เป็นต้น

#### 4.1.2. ผลการทดลองการรับ-ส่งข้อมูลพิกัดผ่านการรังวัดแบบจริง (Real Time Kinematic – RTK) แบบไร้สาย



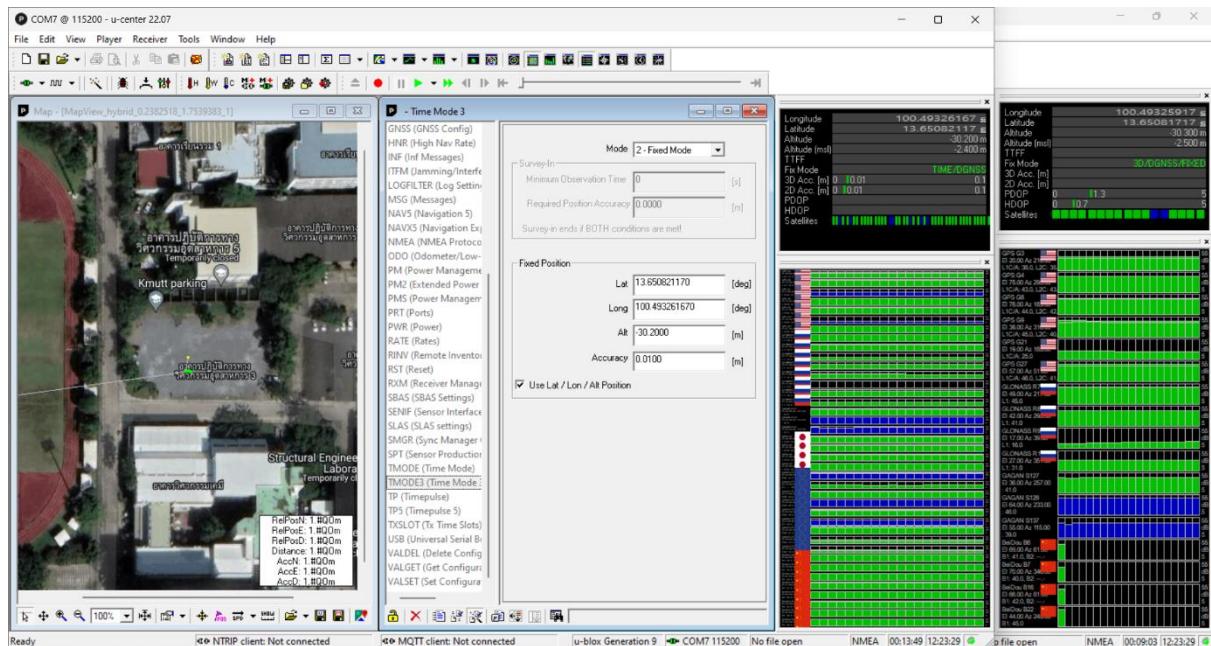
รูปที่ 4.7 การเชื่อมต่อ Base และ Rover แบบไร้สาย

สำหรับการทดลองนี้จะใช้การส่งข้อมูลค่าปรับแก้ผ่าน RF Datalink แทนการส่งข้อมูลผ่านสายไฟ ซึ่งการส่งข้อมูลผ่าน RF Datalink มีข้อดีคือสามารถส่งข้อมูลได้ระยะห่างมากที่สุดถึง 40 กิโลเมตร แต่ข้อเสียคือหากในระยะส่งข้อมูลมีสิ่งกีดขวางมาก อาจจะส่งผลให้การเชื่อมต่อระหว่าง RF Datalink ลuced การเชื่อมต่อได้



รูปที่ 4.8 รายละเอียดการตั้งค่า RF Datalink

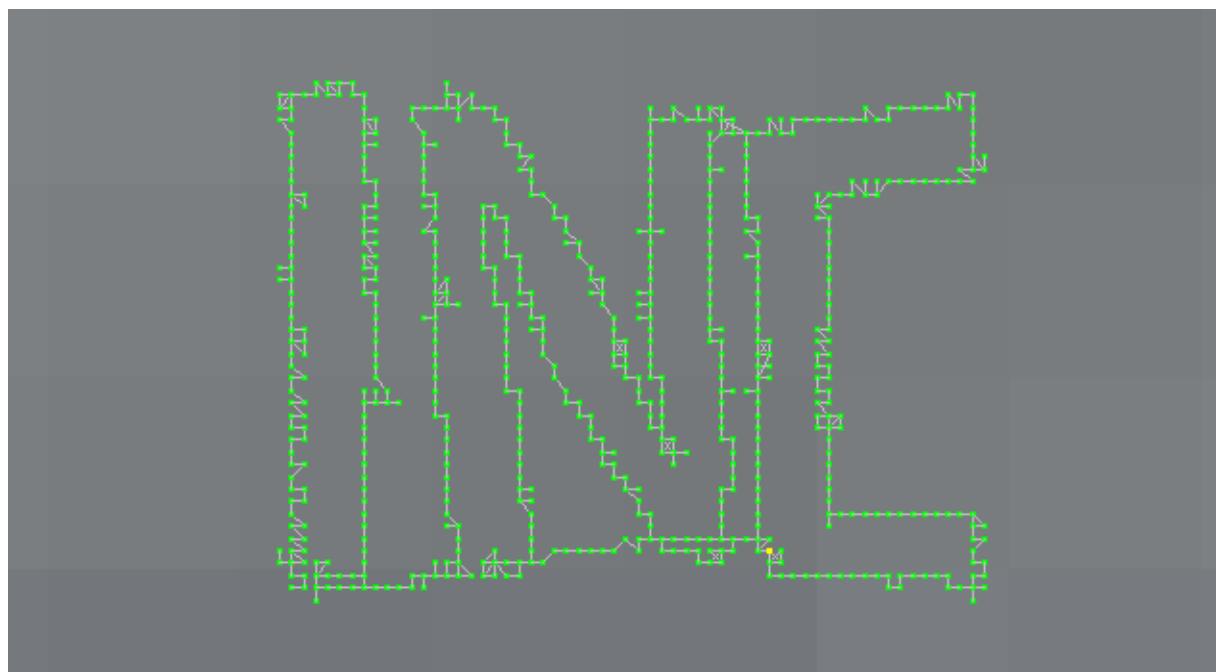
หมายเหตุ : สำหรับการตั้งค่า RF Datalink ต้องเลือก Net ID ให้ตรงกันจึงจะสามารถเชื่อมต่อกันได้



รูปที่ 4.9 ภาพแสดงสถานะ TIME ของ Base Station และสถานะ FIXED ของ Rover Station



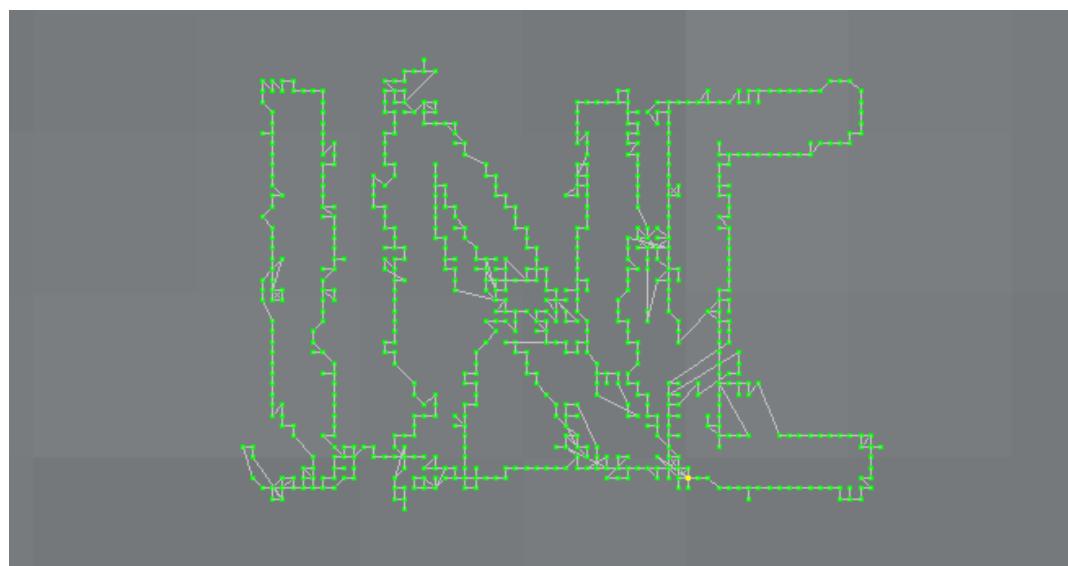
รูปที่ 4.10 รูปแบบตัวอักษรที่ใช้ในการทดสอบ



รูปที่ 4.11 ผลการทดลองที่ได้

สำหรับการทดลองวัดค่าพิกัดแบบ RTK แบบไร้สาย จะต้องคำนึงถึงความละเอียดในการตรวจค่าพิกัดของ Base Station แบบ FIXED Mode ในความละเอียดอยู่ที่ 1 เซนติเมตร แต่รูปแบบตัวอักษรที่ใช้จะมีขนาดเล็กและระยะห่างระหว่างตัวอักษรน้อยกว่าในการทดลองก่อนหน้า ผลของการทดลองพบว่าลักษณะของพิกัดที่ได้มีความใกล้เคียงกับรูปแบบของตัวอักษร แต่จะมีบางช่วงของตัวอักษรที่มีการคลาดเคลื่อนจากตัวอักษรจริงประมาณ 1-5 เซนติเมตร ทั้งนี้เกิดจากขณะที่ผู้ทดลองทำการเคลื่อนที่เสาอากาศ Antenna บริเวณลำตัว ส่วนต่างๆของผู้ทดลอง ได้บดบังการรับสัญญาณบางส่วนทำให้พิกัดที่ได้จึงมีความคลาดเคลื่อน ซึ่งการลดความคลาดเคลื่อนของการตรวจวัดพิกัดสามารถทำได้โดยเคลื่อนที่เสาอากาศ Antenna แบบไม่มีส่วนได้ของร่างกายบดบัง หรือการเปลี่ยนลักษณะของการทดลอง โดยการติดเสาอากาศ Antenna กับอุปกรณ์อื่นที่สามารถเคลื่อนที่ได้ เช่น รถ หรือขานพาหนะอื่นๆ เป็นต้น

#### ข้อผิดพลาดที่พบ

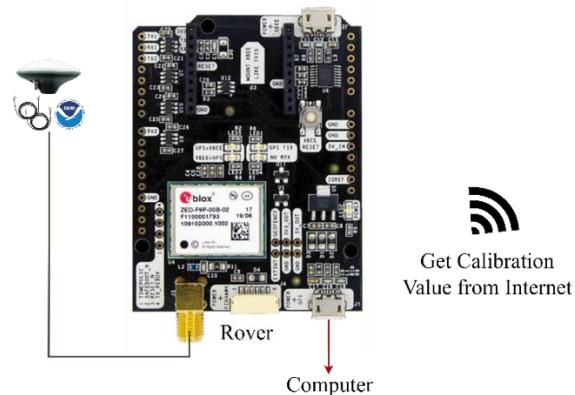


รูปที่ 4.12 ผลการทดลองจากการเปลี่ยนตำแหน่งของ Base Station

เนื่องจากการหาพิกัดของ Base Station ด้วยวิธี Survey-In หากต้องการความละเอียดที่ 1 เซนติเมตรจะใช้เวลาตรวจวัดนานประมาณ 50 นาทีขึ้นไป ดังนั้นในการทดลองครั้งนี้จึงใช้การรังวัดแบบ Fixed Mode ซึ่งจะได้ความละเอียดเท่ากัน โดยจะใส่พิกัดของ Base Station ที่ทราบค่าแน่นอน จากนั้นหลังจากใส่พิกัดเรียบร้อยสถานะ TIME จะแสดงขึ้น ซึ่งแสดงให้เห็นว่าการติดตั้ง Base Station สำเร็จเรียบร้อย แต่การ

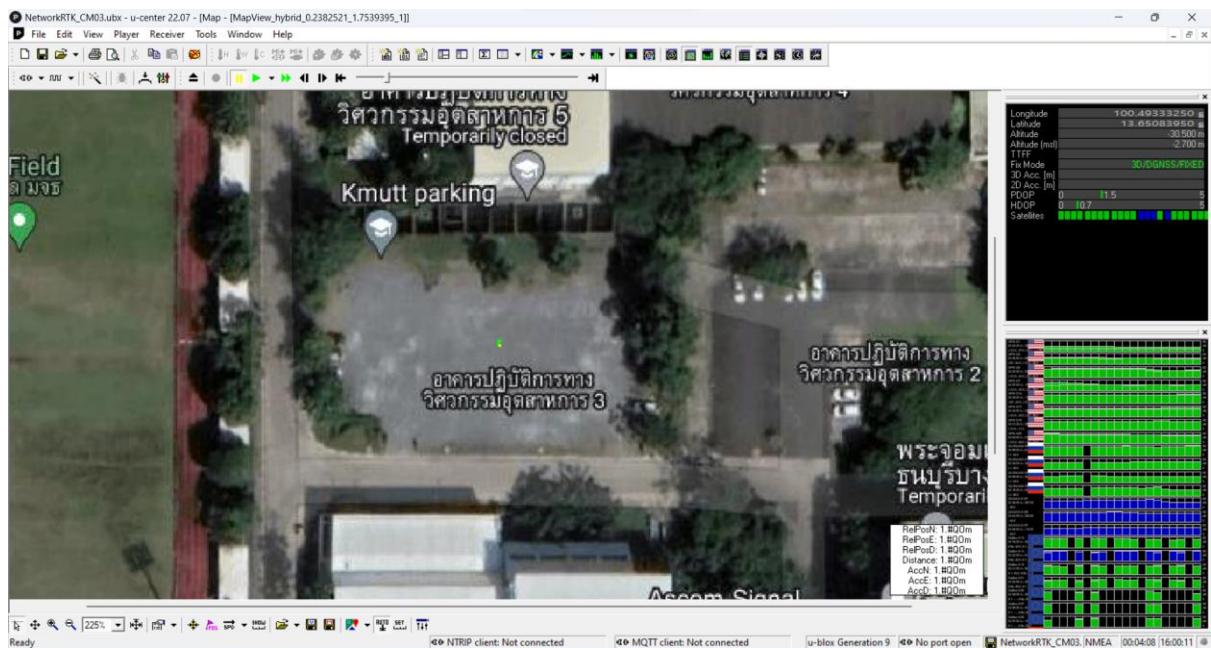
รังวัดด้วยวิธีนี้ (FIXED Mode) หาก Base Station มีการเปลี่ยนตำแหน่ง จะส่งผลให้ค่าพิกัดที่ Rover Station วัดได้เกิดความคลาดเคลื่อนดังรูปที่ A ดังนั้นหากต้องการหาพิกัดแบบ FIXED Mode จำเป็นต้องทราบพิกัดแน่นอน ณ จุดที่ต้องการติดตั้ง Base Station

#### 4.1.3. ผลการทดลองการรับ-ส่งข้อมูลพิกัดผ่านการรังวัดแบบ Network RTK



รูปที่ 4.13 การเชื่อมต่อ Rover ด้วยเทคนิคการวัดแบบ Network RTK

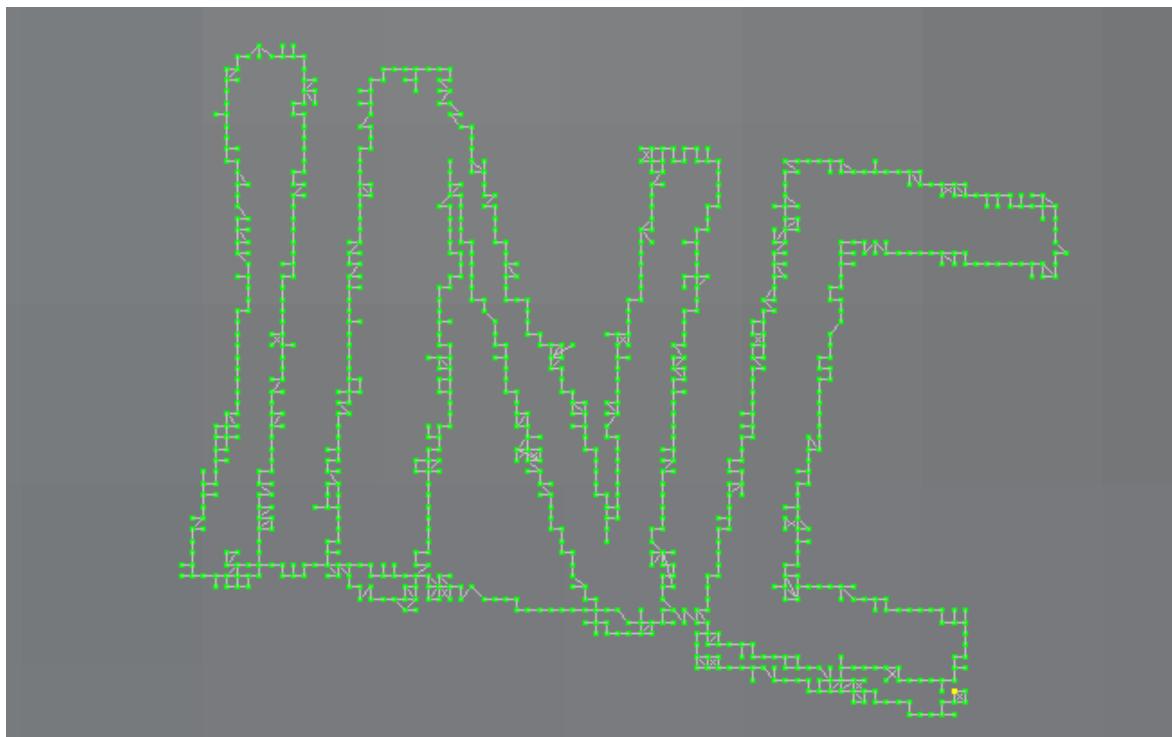
สำหรับการทดลองนี้จะทำการรับข้อมูลปรับแก้ผ่าน Internet ซึ่งทำการตั้งค่าผ่านโปรแกรม U-center โดยเลือกเมนู NTRIP Client และทำการเชื่อมต่อ CORS โดยการทดลองนี้จำเป็นต้องทำในพื้นที่โล่งหรือฟ้าเปิด เนื่องจากไม่ได้ทำการติดตั้ง Base Station เองทำให้สถานะของ Rover สามารถหลุดจากสถานะ FIXED Mode ได้ง่าย



รูปที่ 4.14 ภาพแสดงสถานะ FIXED ของ Rover Station



รูปที่ 4.15 รูปแบบตัวอักษรที่ใช้ในการทดสอบ



รูปที่ 4.16 ผลการทดลองที่ได้

สำหรับการทดลองวัดค่าพิกัดแบบ Network RTK จะทำได้โดยเชื่อมต่อ Receiver Board เข้ากับโปรแกรม U-center จากนั้น ตั้งค่าโดยการเลือกเมนู NTRIP Client และใส่ Username และ Password ที่ได้รับมาจาก กรมที่ดิน ต่อมาให้รอสักครู่เพื่อให้ Rover Board รับค่าปรับแก้จาก CORS เมื่อปรับค่าสำเร็จค่าสถานะของ Board จะแสดงเป็น FIXED Mode ซึ่งหมายความว่าความละเอียดที่วัดพิกัดได้ต่องานนี้จะอยู่ในระดับ เช่นติเมตรแล้ว ผลของการทดลองพบว่าลักษณะของพิกัดที่ได้มีความใกล้เคียงกับรูปแบบของตัวอักษร แต่จะมีบางช่วงของตัวอักษรที่มีการคลาดเคลื่อนจากตัวอักษรจริงประมาณ 1-5 เช่นติเมตร ทั้งนี้เกิดจาก ขณะที่ผู้ทดลองทำการเคลื่อนที่เสาอากาศ Antenna บริเวณลำตัวส่วนต่างๆของผู้ทดลอง ได้บดบังการรับ สัญญาณบางส่วนทำให้พิกัดที่ได้จึงมีความคลาดเคลื่อน ซึ่งการลดความคลาดเคลื่อนของการตรวจสอบพิกัด สามารถทำได้โดยเคลื่อนที่เสาอากาศ Antenna แบบไม่มีส่วนใดของร่างกายบดบัง หรือการเปลี่ยน ลักษณะของการทดลองโดยการติดเสาอากาศ Antenna กับอุปกรณ์อื่นที่สามารถเคลื่อนที่ได้ เช่น รถ หรือ ยานพาหนะอื่นๆ เป็นต้น

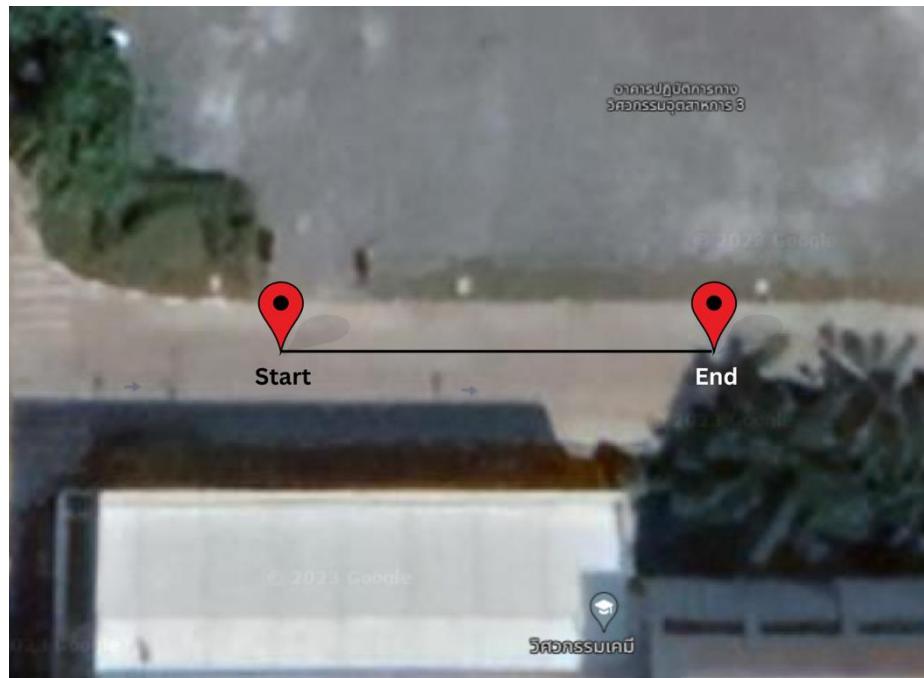
## สรุปผลการทดลองการรับ-ส่งข้อมูลพิกัดผ่านการรังวัดแบบจน

จากการทดลองทั้งหมดสามารถจำแนกวิธีการรังวัดได้ 3 วิธี ได้แก่ 1. วิธี Real Time Kinematic แบบใช้สาย 2. วิธี Real Time Kinematic แบบไร้สาย 3. วิธี Network RTK โดยทั้ง 3 วิธีสามารถวัดค่าพิกัดในระดับเซนติเมตร ได้ทั้งหมด และให้ผลการวัดในระดับที่ยอมรับได้และเพียงต่อการใช้งานต่อในการกำหนดค่าพิกัดสำหรับการพัฒนาระบบรถยนต์เคลื่อนที่แบบอัตโนมัติ แต่เมื่อพิจารณาถึงอุปกรณ์ในการวัดของทั้ง 3 วิธี จะพบว่าการรังวัดด้วยวิธี Network RTK จะใช้อุปกรณ์ในการวัดเพียง 1 ชุด ประกอบด้วย Receiver Board และเสาอากาศ Antenna ซึ่งต่างจาก 2 วิธีก่อนหน้าที่จำเป็นต้องใช้อุปกรณ์ถึง 2 ชุดในการติดตั้งเป็น Base Station และ Rover Station ดังนั้นสำหรับการเลือกใช้งานจะเลือกใช้การรังวัดแบบ Network RTK เนื่องจากใช้อุปกรณ์ที่น้อยกว่าและสามารถช่วยลดต้นทุนในการพัฒนาได้

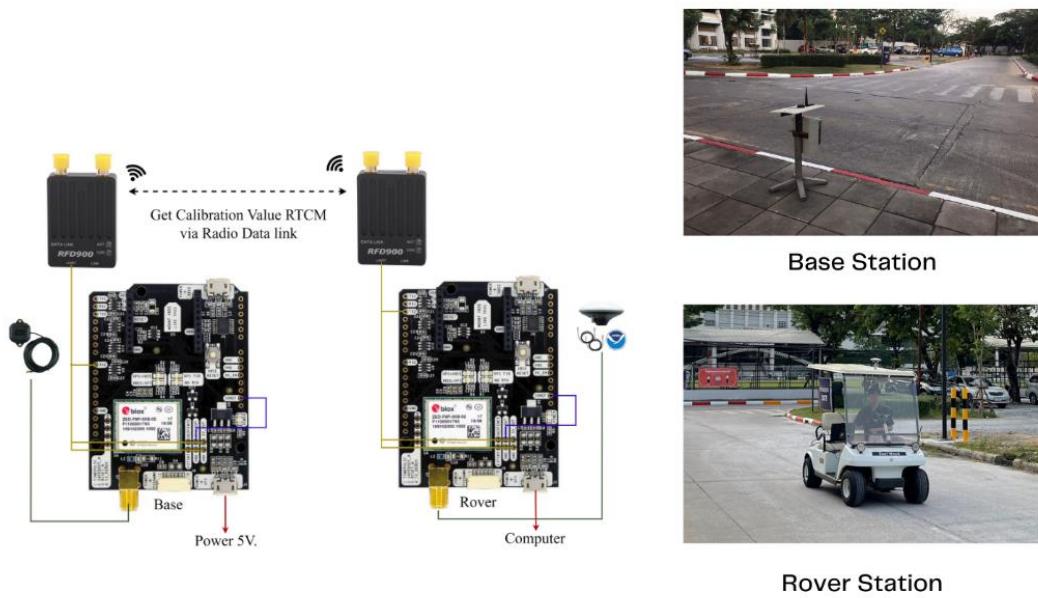
### 4.1.4. ผลการทดลองเบรี่ยนเทียบคุณภาพของพิกัดผ่านการรังวัดโดยการใช้ GPS Quality Indicator ใน NMEA GGA Protocol

สำหรับการทดลองนี้จะเป็นการทดลองเพื่อเบรี่ยนเทียบคุณภาพของพิกัดที่รังวัดได้ทั้งในเทคนิคแบบ Real Time Kinematic และ Network RTK โดยลักษณะการรังวัดจะเป็นการวัดพิกัดด้วย 2 ระยะทาง ได้แก่ 1. ระยะทางสั้น บริเวณข้างลานจอดรถ ภาควิชาวิศวกรรมเคมี ซึ่งบริเวณนี้จะเป็นบริเวณที่มีสิ่งกีดขวางน้อย และมีท้องฟ้าโปร่ง 2. ระยะทางยาว บริเวณรอบสนามกีฬา ซึ่งบริเวณนี้จะเป็นบริเวณที่มีสิ่งกีดขวางมากซึ่งมีโอกาสสูญเสียสัญญาณค่าปรับแก้ที่ส่งมาถึง Rover Station โดยรูปแบบการทดลองจะทำการเก็บค่าพิกัดทั้งหมด 6 ครั้ง และทำการนับจำนวน GPS Quality Indicator ในลำดับที่ 2, 4, 5 เนื่องจากการใช้งานจริงมีเพียงลำดับที่กล่าวในข้างต้นที่แสดงในการเก็บค่าพิกัด

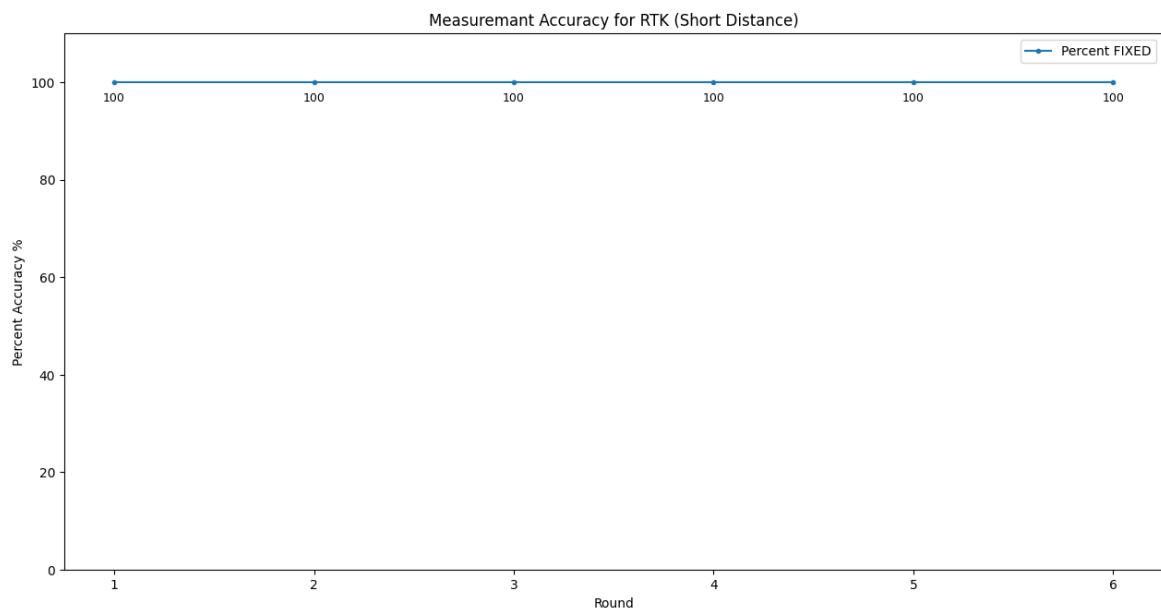
#### 4.1.4.1 การทดลองเบรี่ยนเที่ยบคุณภาพของพิกัดผ่านการรังวัดในระยะทางสั้นบริเวณข้างลานจอดรถ



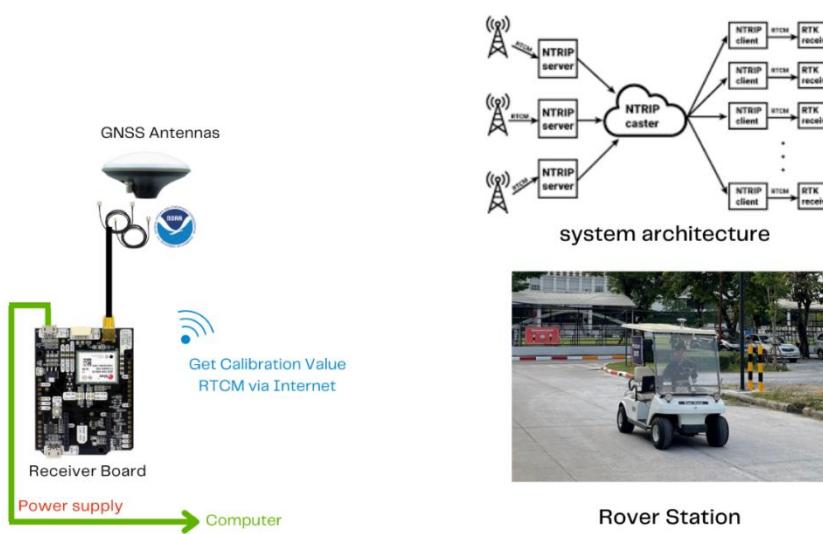
รูปที่ 4.17 ระยะทางอ้างอิงในการรังวัด



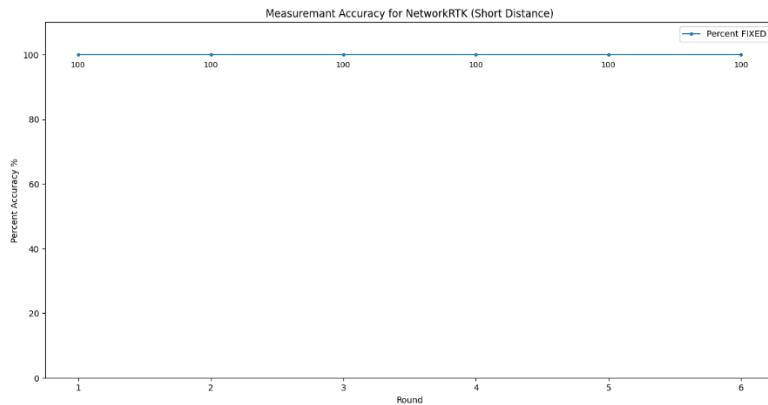
รูปที่ 4.18 อุปกรณ์และการติดตั้งในการรังวัดแบบ Real Time Kinematic



รูปที่ 4.19 GPS Quality Indicator ที่ได้ในการรังวัดแบบ Real Time Kinematic



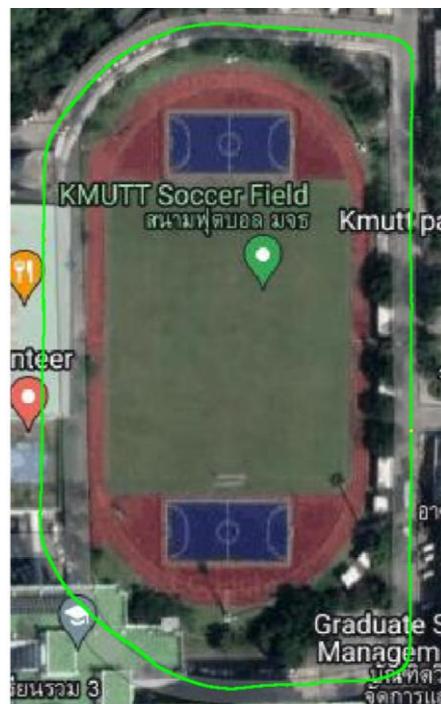
รูปที่ 4.20 อุปกรณ์และการติดตั้งในการรังวัดแบบ Network RTK



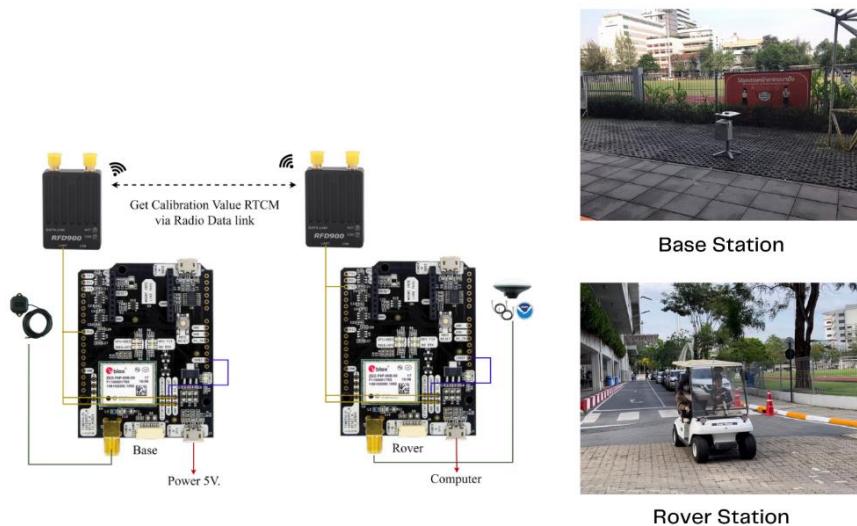
รูปที่ 4.21 GPS Quality Indicator ที่ได้ในการรังวัดแบบ Network RTK

สำหรับการทดลองนี้จะเห็นได้ว่าคุณภาพของสัญญาณที่รังวัดได้ทั้งในแบบ Real Time Kinematic และ Network RTK ในกรณีเก็บพิกัดระยะสั้น มี GPS Quality เป็นชนิด FIXED ทั้งหมด ซึ่งหมายความว่าการใช้งานในสภาพแวดล้อมที่ห้องฟ้าโปร่งและลื่นกีดขวางรอบข้างน้อยสามารถใช้การรังวัดได้ทั้ง 2 แบบ

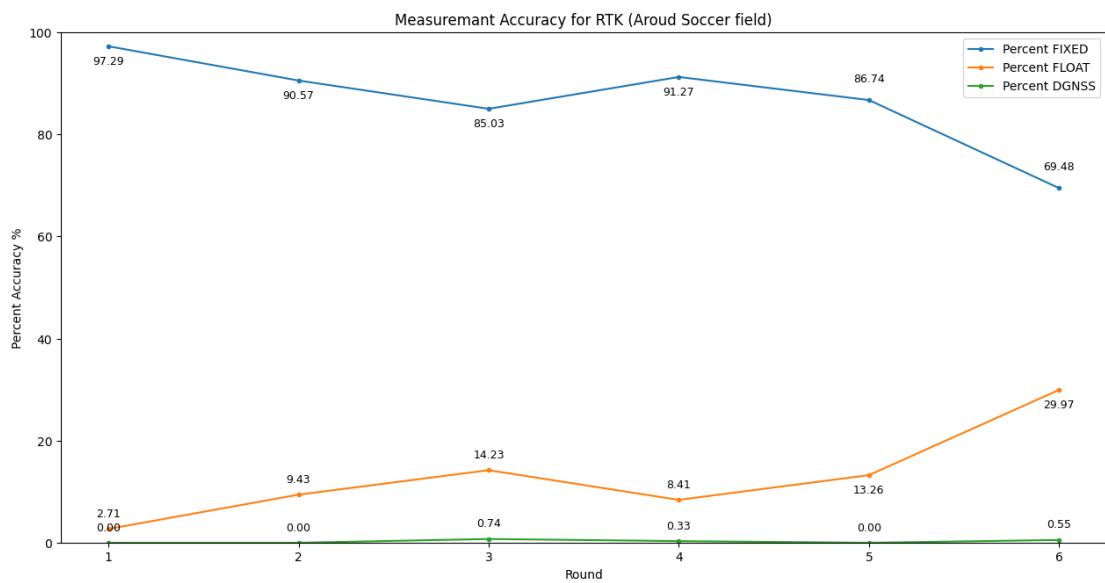
#### 4.1.4.2 การทดลองเปรียบเทียบคุณภาพของพิกัดผ่านการรังวัดในระยะทางยาวบริเวณรอบสนามกีฬา



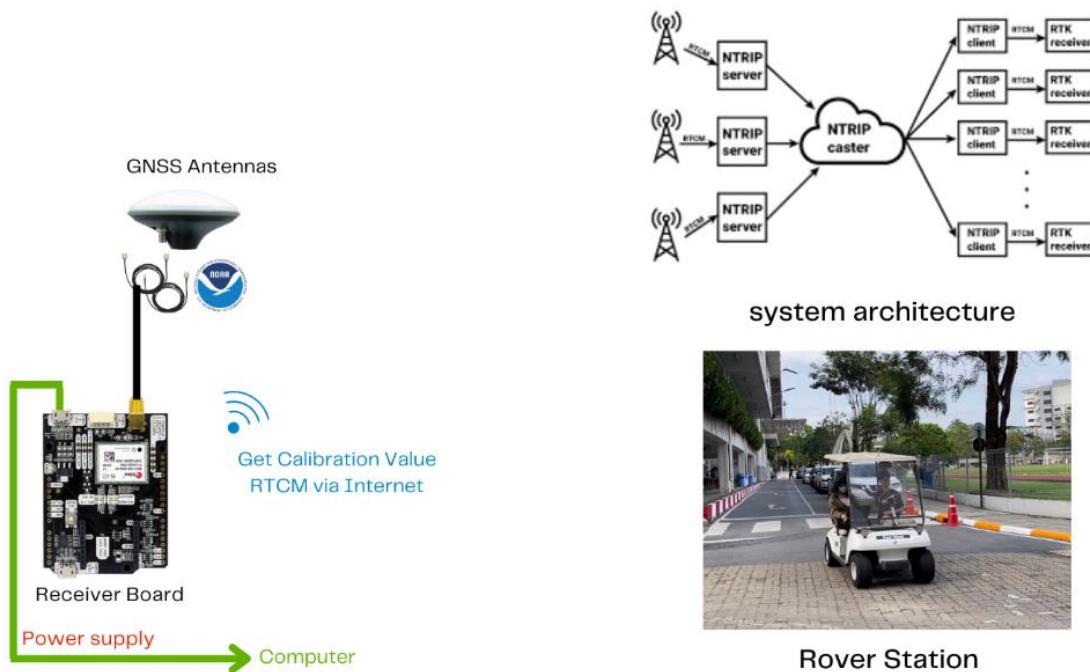
รูปที่ 4.22 ระยะทางอ้างอิงในการรังวัด



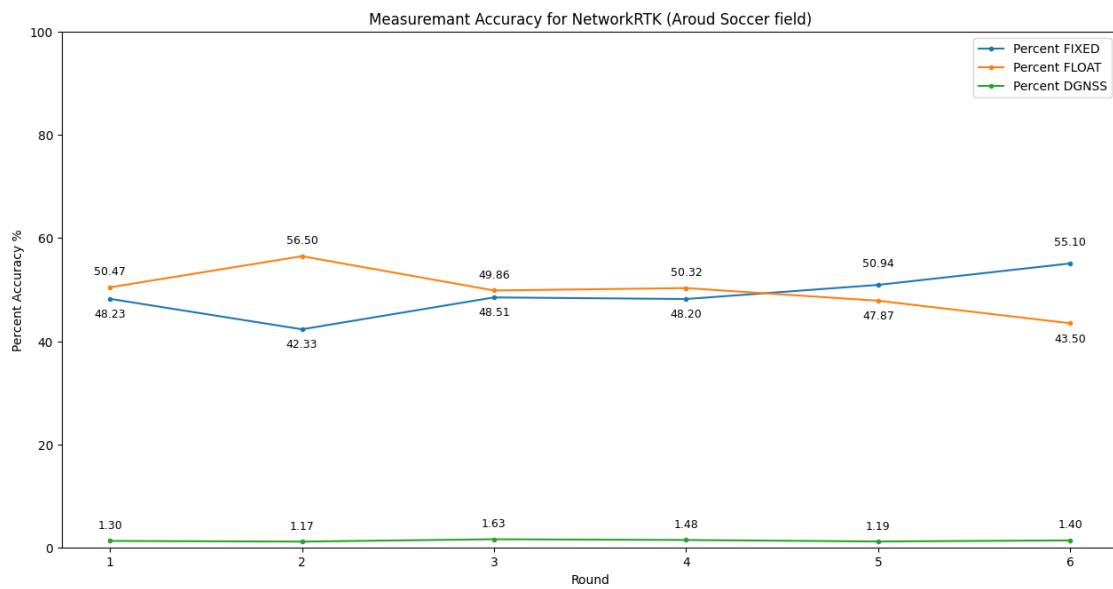
รูปที่ 4.23 อุปกรณ์และการติดตั้งในการรังวัดแบบ Real Time Kinematic



รูปที่ 4.24 GPS Quality Indicator ที่ได้ในการรังวัดแบบ Real Time Kinematic



รูปที่ 4.25 อุปกรณ์และการติดตั้งในการรังวัดแบบ Network RTK

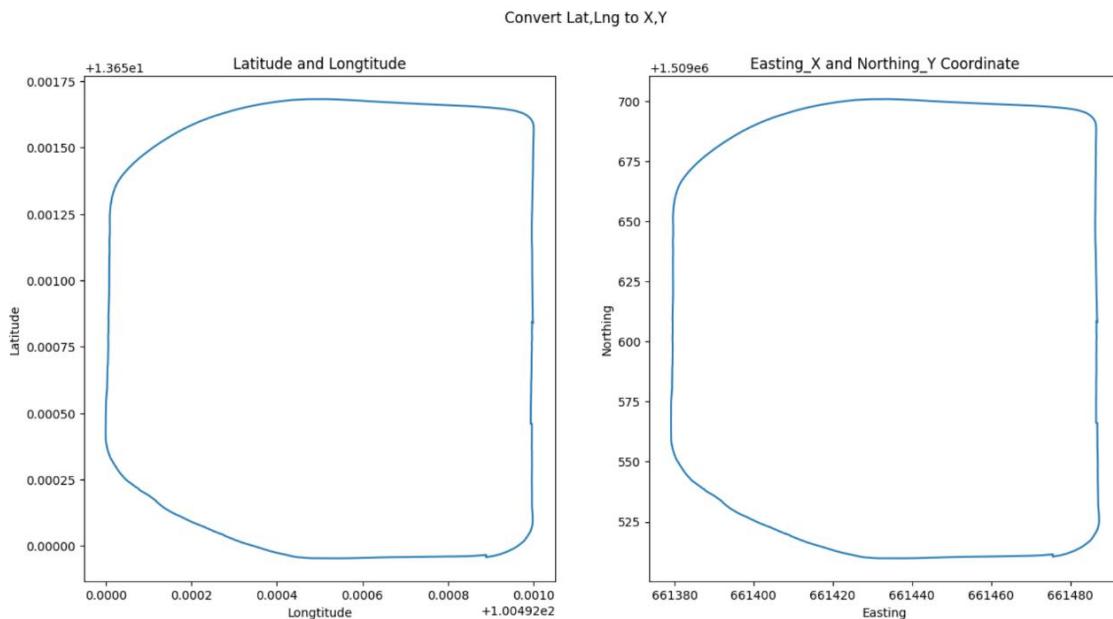


รูปที่ 4.26 GPS Quality Indicator ที่ได้ในการรังวัดแบบ Network RTK

สำหรับการทดลองนี้จะเห็นได้ว่าคุณภาพของสัญญาณที่รังวัดได้ทั้งในแบบ Real Time Kinematic และ Network RTK ในการเก็บพิกัดระยะทางมี GPS Quality ค่อนข้างแตกต่างกันพอสมควร โดยเทคนิคการรังวัดแบบ Real Time Kinematic มีเปอร์เซ็นต์ FIXED Quality เฉลี่ยอยู่ที่ 86.73 % และแบบ Network RTK อยู่ที่ 48.89 % ทั้งนี้สาเหตุเกิดมาจากการสภาพแวดล้อมบริเวณรอบสนามมีสิ่งกีดขวางมาก เช่น ตึกหรืออาคารทำให้เกิดความคลาดเคลื่อนต่อค่าพิกัดที่วัด โดยหากต้องการใช้งานในบริเวณพื้นที่ที่มีสิ่งกีดขวางมากการเลือกใช้การรังวัดแบบ Real Time Kinematic จะมีความละเอียดที่สูงมากกว่าการรังวัดแบบ Network RTK แต่การรังวัดแบบ Network RTK ความละเอียดก็ยังสามารถยอมรับได้

#### 4.1.5. ผลการทดลองในการแปลงค่าพิกัดละติจูดลงติจูดเป็นค่าพิกัดแบบ UTM X,Y ระยะทางยาวบริเวณรอบสนามกีฬา

การทดลองนี้มีจุดประสงค์เพื่อทดสอบความถูกต้องของการแปลงค่าพิกัดละติจูดลงติจูดเป็นค่าพิกัดแบบ UTM X,Y โดยจะทำการใช้โปรแกรมภาษา Python เพื่อรับค่าและจัดการข้อมูลของ GGA NMEA Frame เพื่อดึงข้อมูลละติจูดและลองติจูดใน Array ของ Data Frame มาใช้ จากนั้นจะทำการเปลี่ยนค่าละติจูดและลองติจูดที่ได้ให้อยู่ในค่าพิกัด X,Y ในระบบพิกัดแบบ UTM



รูปที่ 4.27 กราฟเปรียบเทียบระหว่างพิกัดละติจูดลงติจูดและพิกัด UTM XY บริเวณรอบสนามฟุตบอล

การทดลองนี้จะทำการเปรียบเทียบผลการแปลงค่าระหว่างค่าพิกัดคลาสติกูลองติจูดและค่าพิกัด UTM XY จากข้อมูลดิบที่มีอยู่ ที่ได้จากการวัดในการทดลองการรับ-ส่งข้อมูลพิกัดผ่านการรังวัดแบบ Network RTK จากราฟเปรียบเทียบพบว่าพิกัด UTM XY ที่แปลงได้มีความถูกต้องเมื่อเทียบกับพิกัดคลาสติกูลองติจูด

#### **4.1.6. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิงในการเคลื่อนที่ของรถ**

การทดลองนี้มีจุดประสงค์เพื่อสร้างเส้นทางอ้างอิงในการเคลื่อนที่ของรถแบบอัตโนมัติรอบสนามกีฬาจากการวัดพิกัด UTM XY โดยมีระยะทางรวมทั้งหมดประมาณ 550 เมตร ซึ่งในการวัดพิกัดจะแบ่งเส้นทางการในการวัดออกเป็น 8 เส้นทางดังนี้

**ตารางที่ 4.1 ตารางแสดงลำดับเส้นทางและระยะทางรอบสนามกีฬา**

Path Order	Name	Distance (m.)
1	Straight Path 1: South to North	169
2	Curve Path 1: North to West	19
3	Straight Path 2: East to West	39
4	Curve Path 2: West to South	79
5	Straight Path 3: North to South	96
6	Curve Path 3: South to East	78
7	Straight Path 4: West to East	38
8	Curve Path 4: East to North	34
	<b>Total</b>	<b>552</b>

ซึ่งลำดับการเดินทางรอบสนามฟุตบอลจะเริ่มจาก เส้นทางตรงที่ 1: ทิศการเดินทางจากทิศใต้สู่ทิศเหนือไปยังเส้นทางโค้งที่ 1: ทิศการโค้งจากทิศเหนือสู่ทิศตะวันตก ไปยัง เส้นทางตรงที่ 2: ทิศการเดินทางจากทิศตะวันออกไปยังทิศตะวันตก ไปยัง เส้นทางโค้งที่ 2: ทิศการโค้งจากทิศตะวันตกไปยังทิศใต้ ไปยัง

เส้นทางตรงที่ 3: ทิศการเดินทางจากทิศเหนือไปยังทิศใต้ ไปยัง เส้นทางโถงที่ 3: ทิศการโถงจากทิศใต้ไปยังทิศตะวันออก ไปยัง เส้นทางตรงที่ 4: ทิศการเดินทางจากทิศตะวันตกไปยังทิศตะวันออก ไปยัง เส้นทางโถงที่ 4: ทิศการโถงจากทิศตะวันออกไปยังทิศเหนือ จากนั้นเมื่อจบเส้นทางโถงที่ 4 จะครบรอบการเดินทางและกลับมายังจุดเริ่มต้นเส้นทางตรงที่ 1



รูปที่ 4.28 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล

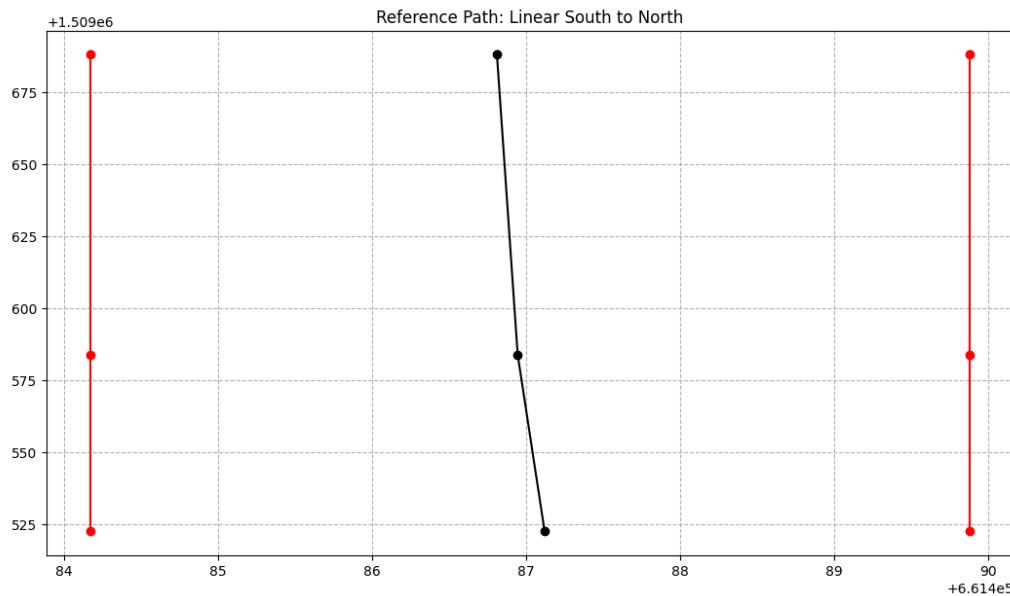
#### 4.1.6.1. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิง : เส้นทางตรงที่ 1



รูปที่ 4.29 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 1

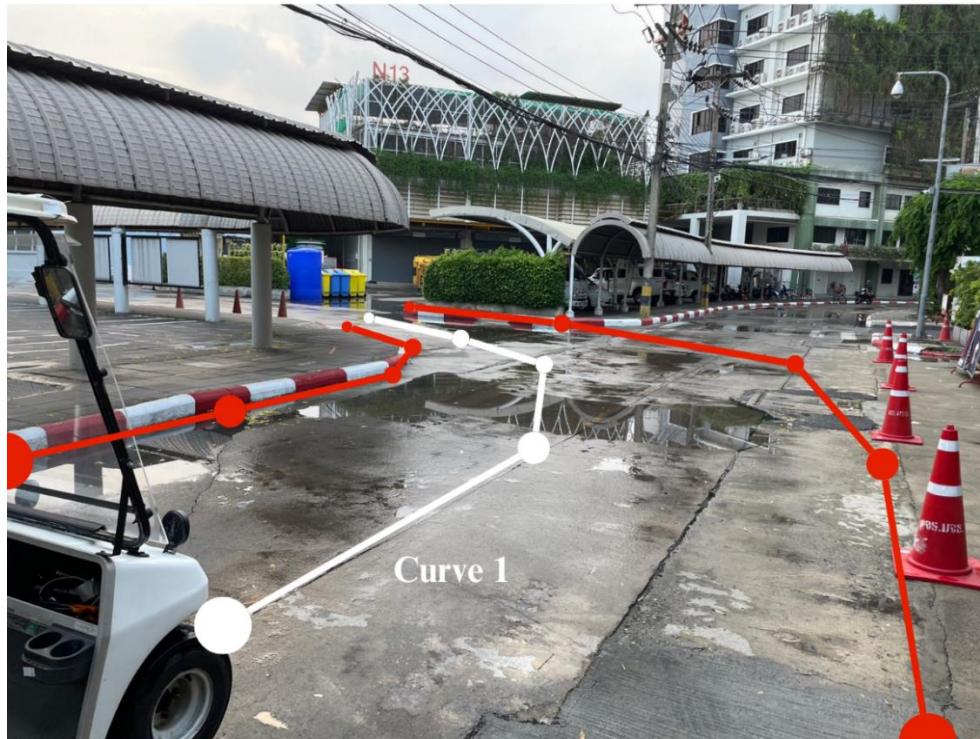


รูปที่ 4.30 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 1

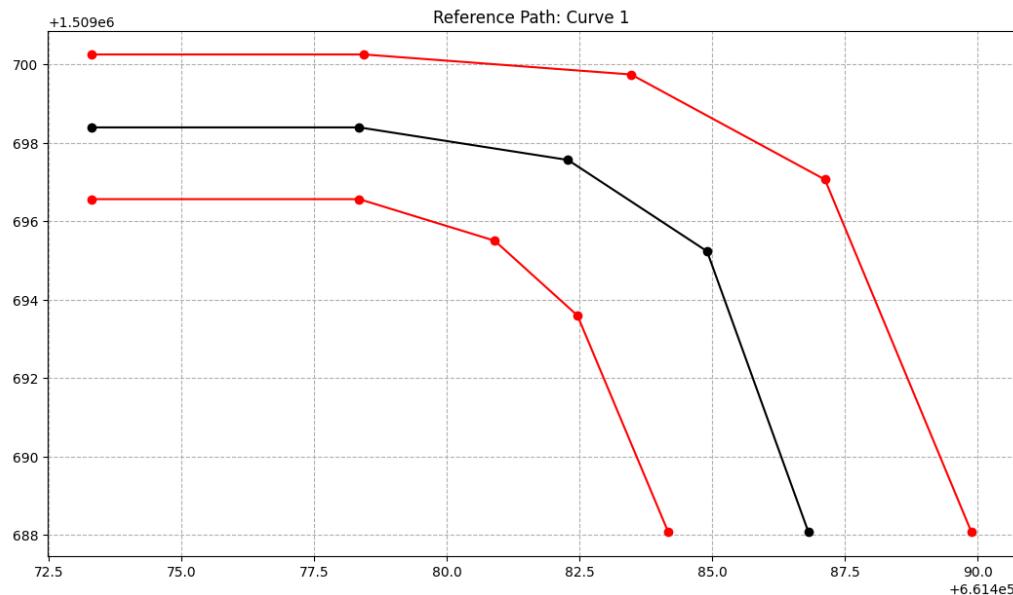


รูปที่ 4.31 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 1

#### 4.1.6.2. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิง : เส้นทางโถงที่ 1



รูปที่ 4.32 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางโถงที่ 1

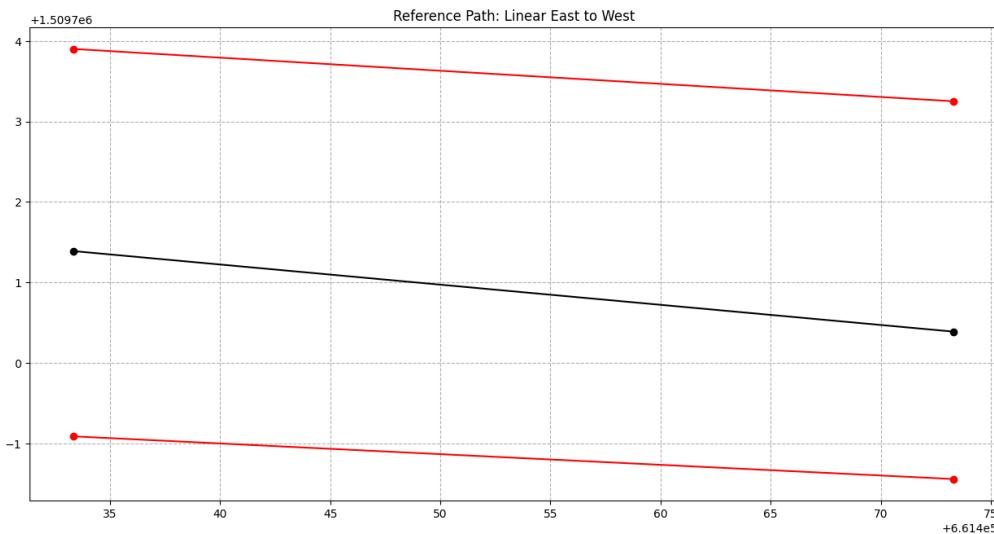


รูปที่ 4.33 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทาง โถ้งที่ 1

#### 4.1.6.3. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิง : เส้นทางตรงที่ 2



รูปที่ 4.34 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 2

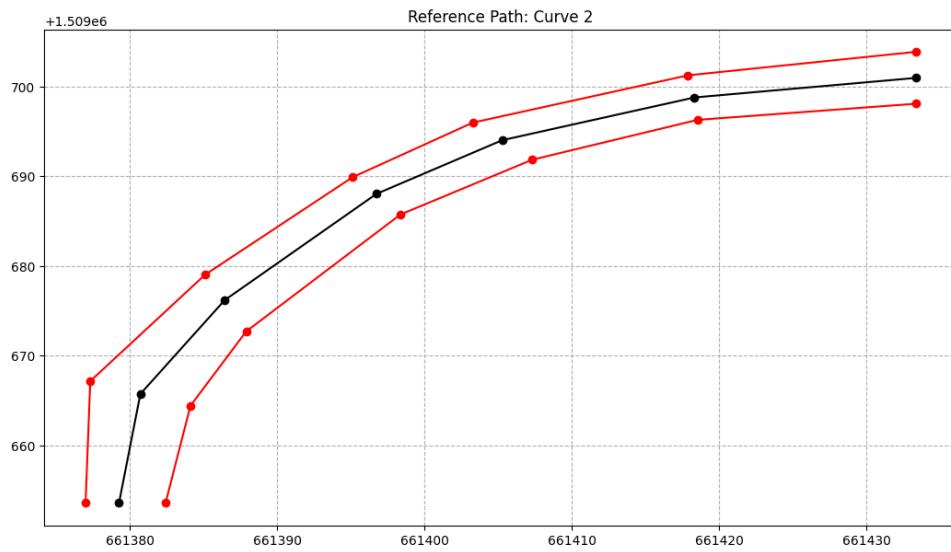


รูปที่ 4.35 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 2

#### 4.1.6.4. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิง : เส้นทางโค้งที่ 2



รูปที่ 4.36 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางโค้งที่ 2

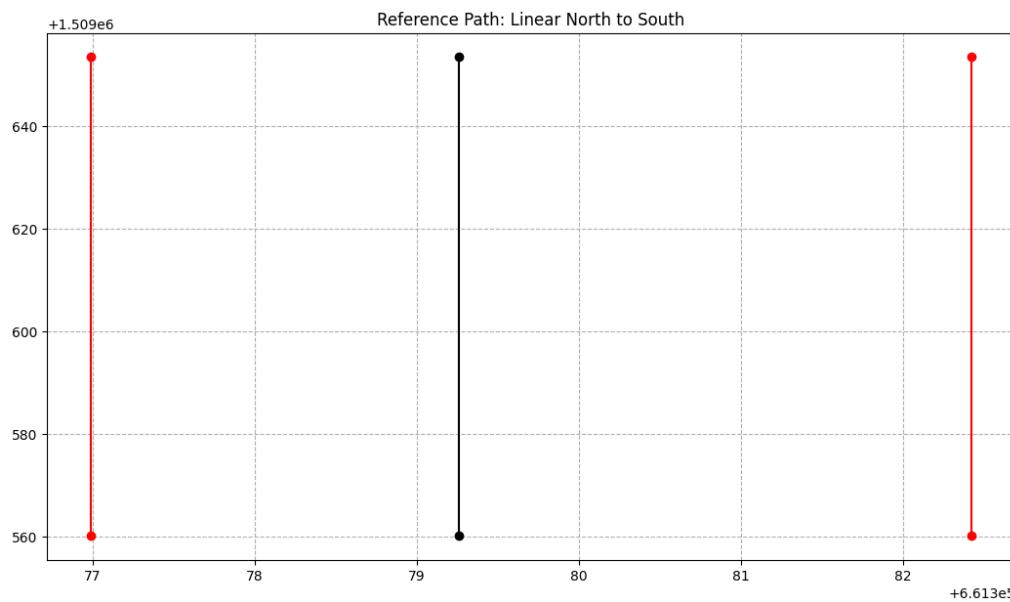


รูปที่ 4.37 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางโถงที่ 2

#### 4.1.6.5. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิง : เส้นทางตรงที่ 3

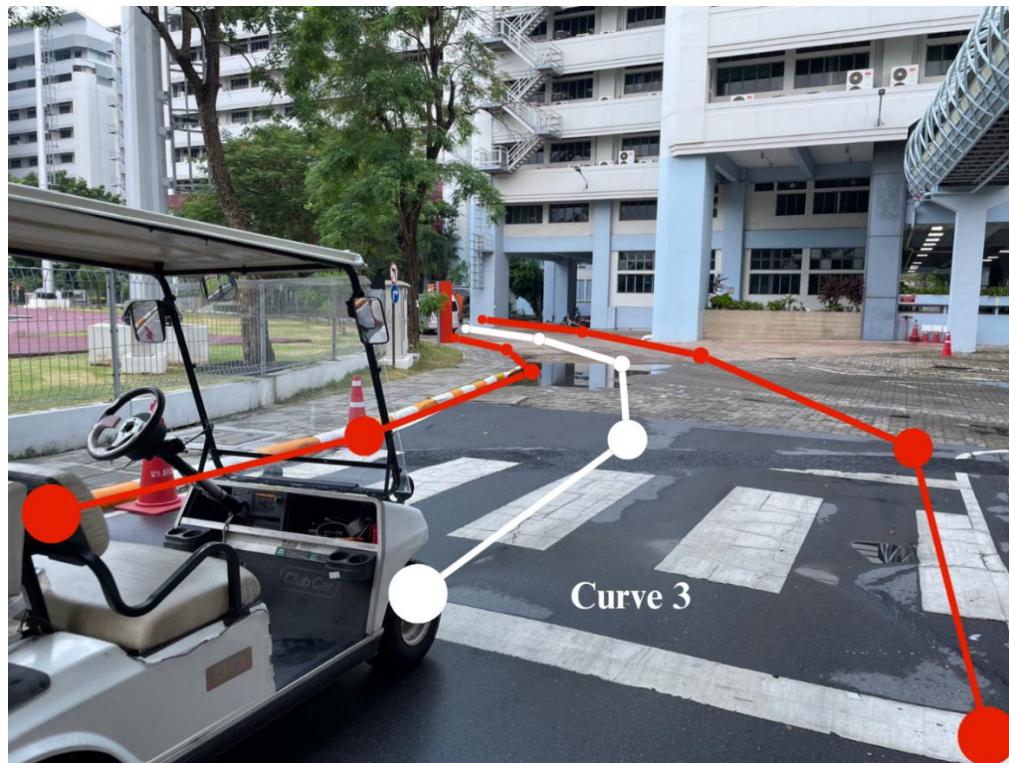


รูปที่ 4.38 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 3

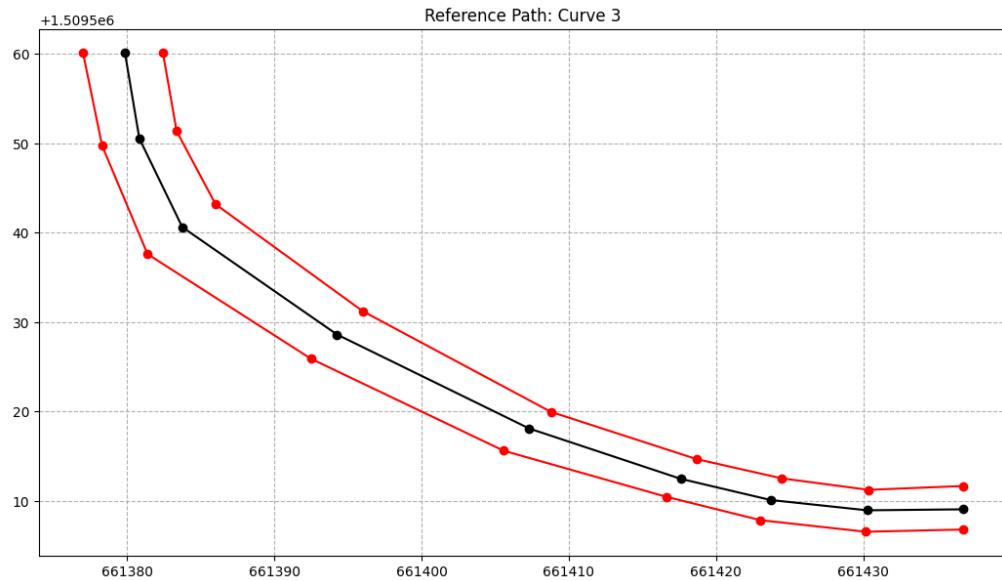


รูปที่ 4.39 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 3

#### 4.1.6.6. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิง : เส้นทางโค้งที่ 3

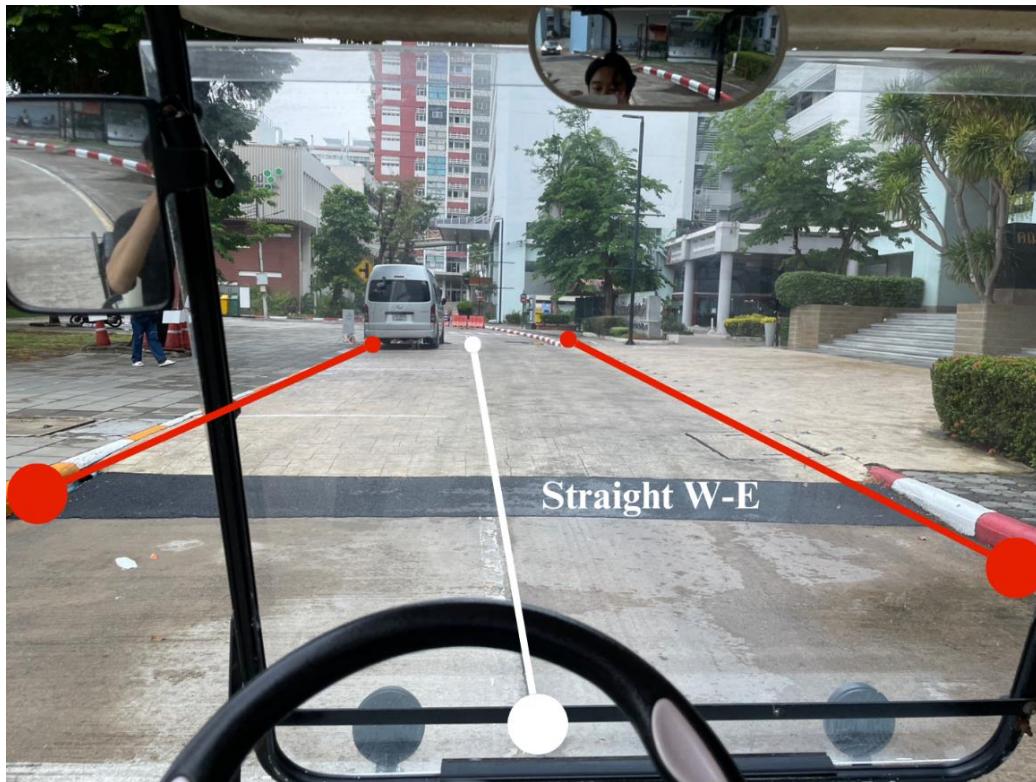


รูปที่ 4.40 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางโค้งที่ 3

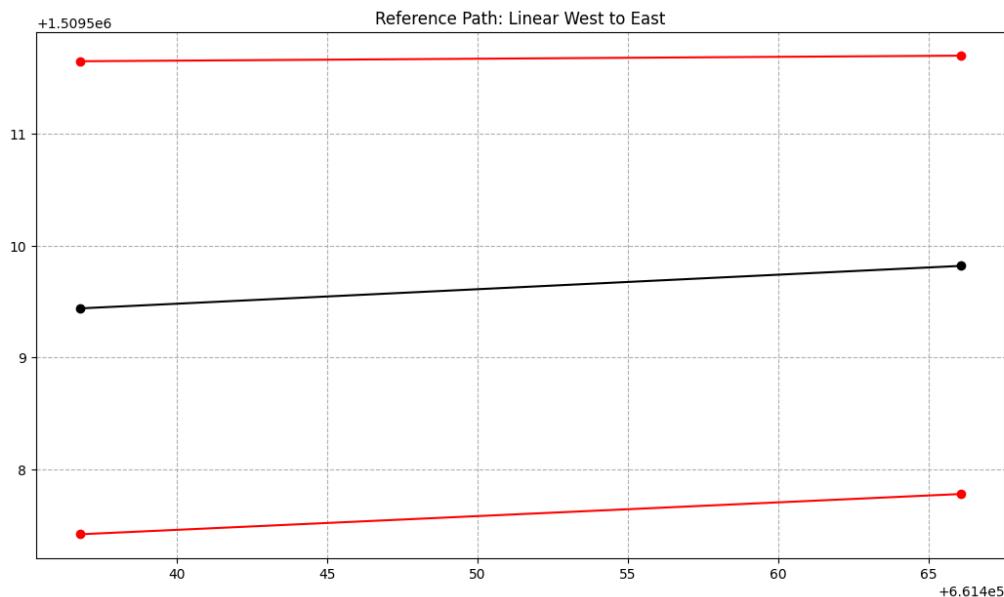


รูปที่ 4.41 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางโถงที่ 3

#### 4.1.6.7. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิง : เส้นทางตรงที่ 4



รูปที่ 4.42 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 4

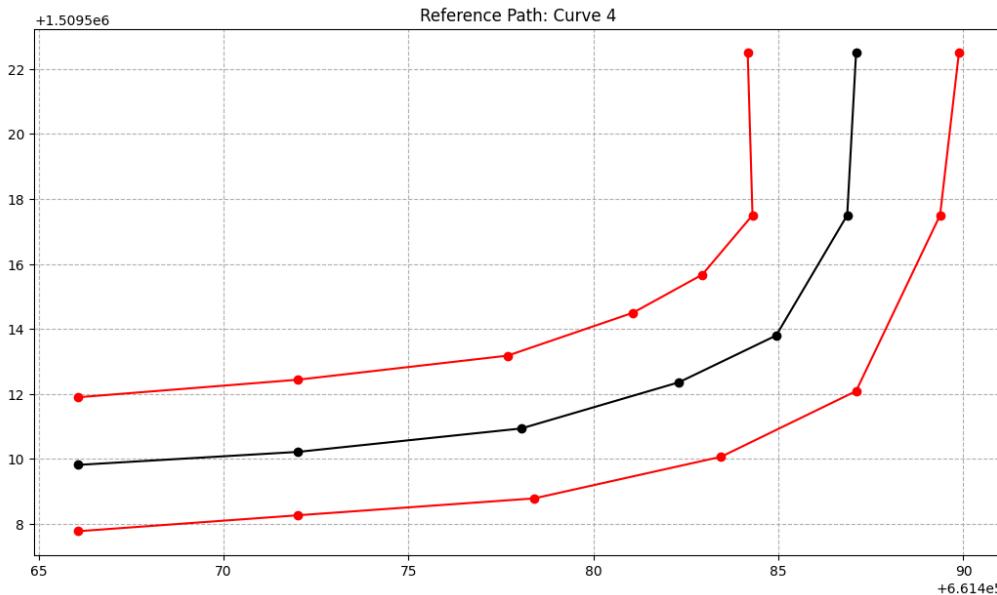


รูปที่ 4.43 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทางตรงที่ 4

#### 4.1.6.8. ผลการทดลองการวัดพิกัด UTM XY เพื่อสร้างเส้นทางอ้างอิง : เส้นทางโถงที่ 4



รูปที่ 4.44 ภาพแสดงเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทาง โถงที่ 4



รูปที่ 4.45 กราฟแสดงพิกัด XY และเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล : เส้นทาง โถงที่ 4

### สรุปผลการทดลองการวัดพิกัดเส้นทางอ้างอิงบริเวณรอบสนามฟุตบอล

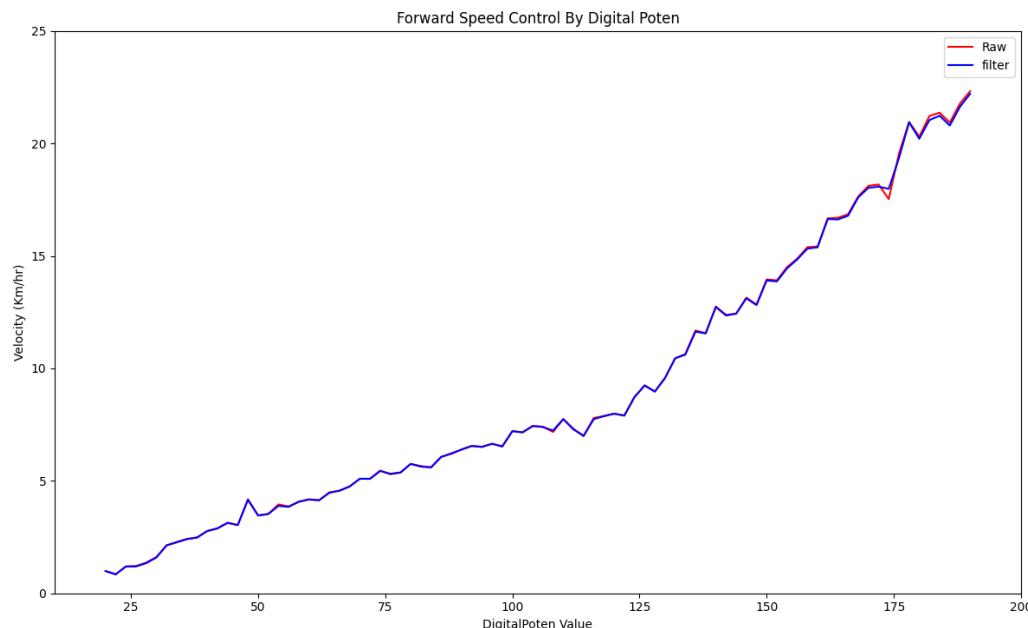
การทดลองนี้จะทำการออกแบบเส้นทางอ้างอิงในการเคลื่อนที่อัตโนมัติรอบสนามฟุตบอลจากการวัดพิกัดตำแหน่ง UTM XY ของเส้นทางจริง โดยเริ่มจากกำหนดจุดพิกัดเริ่มต้นและจุดสิ้นสุดของแต่ละเส้นทาง จากนั้นนำ GNSS เช่นเซอร์ไววัลยังจุดที่กำหนดด้วยเทคนิคการวัดแบบ Network RTK จุดละ 3 ครั้งครั้งละ 60 วินาที เพื่อความถูกต้องและหาค่าเฉลี่ยของจุดพิกัดดังกล่าว ซึ่งการออกแบบเส้นทางอ้างอิงในกรณีที่เส้นทางจริงมีลักษณะเป็นทางตรงจะทำการวัดพิกัดจุดเริ่มต้นและจุดสิ้นสุดดังที่กล่าวไว้ข้างต้น และวัดพิกัดขอบถนนฝั่งซ้ายและฝั่งขวาเพื่อเป็นเงื่อนไขในการควบคุมรถ โดยถ้าหากตำแหน่งปัจจุบันของรถ เข้าใกล้พิกัดของขอบถนนจะให้รถหยุดทำงาน และในส่วนของการออกแบบเส้นทางในกรณีที่เส้นทางจริงมีลักษณะเป็นเส้นโค้งจะทำการแบ่งเส้นโถงนั้นออกเป็นเส้นตรงต่อ กันดังที่แสดงในกราฟของเส้นทาง โถง 1-4 จากผลของการวัดพบว่าพิกัดที่ได้มีความถูกต้องและแม่นยำ โดยมีการคลาดเคลื่อนจากพิกัดจริงอยู่ไม่เกิน 10 เมตร ซึ่งสำหรับการออกแบบเส้นทางอ้างอิง ความคลาดเคลื่อนในระดับนี้สามารถยอมรับได้

## 4.2 ผลการทดลองการควบคุมความเร็วโดยการปรับค่า Digital Potentiometer (MCP41010)

ในการทำการทดลองดังกล่าวจะเป็นการควบคุมค่า Digital Potentiometer ที่มีหลักการทำงานคล้ายตัวด้านหน้าปรับค่า ที่สามารถปรับค่าได้ตั้งแต่ 0 ถึง 255 (256 ตำแหน่ง) ซึ่งในการควบคุมจะเป็นการเขียนโปรแกรมบน Arduino เพื่อควบคุมค่าความเร็ว โดยเส้นสีแดง คือ ข้อมูลที่ไม่ผ่านการกรองสัญญาณ และเส้นสีน้ำเงิน คือ ข้อมูลที่ผ่านการกรองสัญญาณ โดยหัวข้อที่ทำการทดลองและบันทึกผลประกอบด้วย

### 4.2.1 การควบคุมความเร็วการเคลื่อนที่แบบ Forward (Km/hr) แบบ Open loop

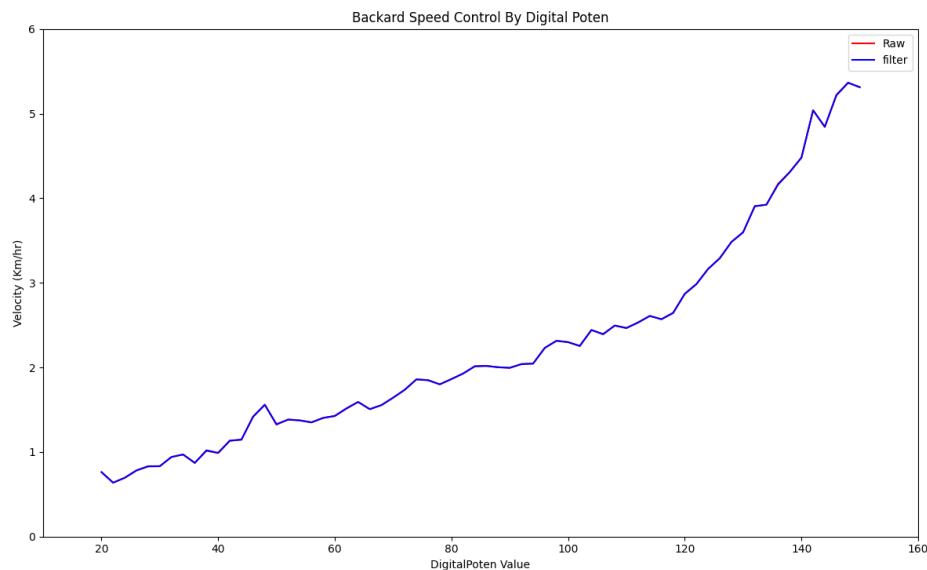
โดยรถสามารถเริ่มทำงานแบบมี Load ที่ Digital Potentiometer เท่ากับ 20 จนถึง 255 แต่ในการทดลอง 4.2.1 สามารถทำการทดลองและบันทึกผลถึง 190 เนื่องจากความเร็วมากเกิดที่จะเก็บผลการทดลอง



รูปที่ 4.46 การควบคุมความเร็วการเคลื่อนที่แบบ Forward (Km/hr) แบบ Open loop

#### 4.2.2 การควบคุมความเร็วการเคลื่อนที่แบบ Backward (Km/hr) และ Open loop

โดยรถสามารถเริ่มทำงานแบบมี Load ที่ Digital Potentiometer เท่ากับ 20 จนถึง 255 และในการทดลอง 4.2.2 สามารถทำการทดลอง และบันทึกผลถึง 150 เนื่องจากความเร็วมากเกิดที่จะเก็บผลการทดลอง

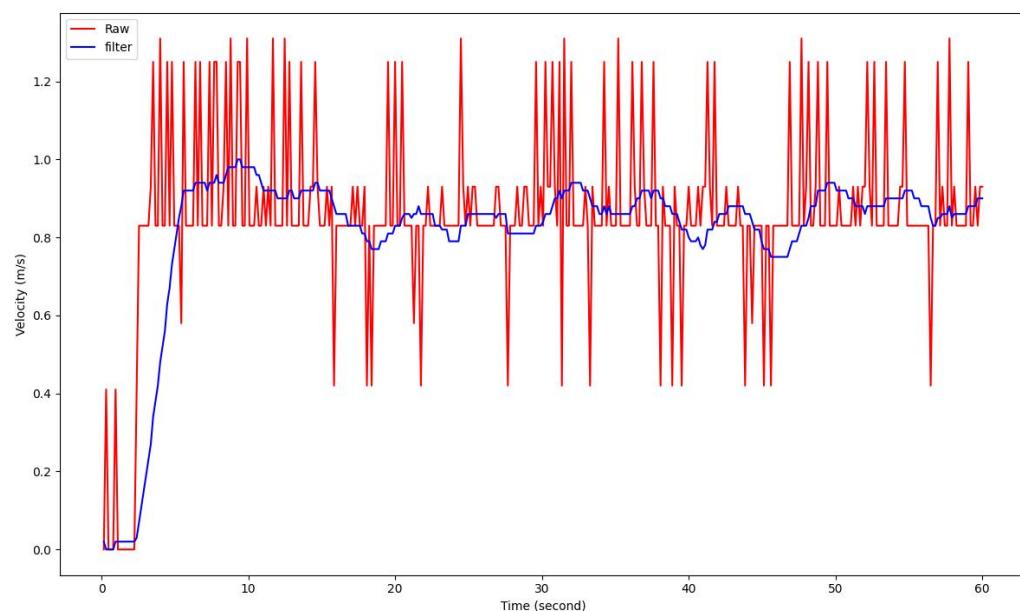


รูปที่ 4.47 การควบคุมความเร็วการเคลื่อนที่แบบ Backward (Km/hr) และ Open loop

Forward window size = 20		Backward window size = 20	
Digitalpoten	speed (km/hr)	Digitalpoten	speed (km/hr)
20	0.98	20	0.76
22	0.83	22	0.63
24	1.18	24	0.69
26	1.19	26	0.78
28	1.33	28	0.82
30	1.58	30	0.83
30	2.13	30	0.94
34	2.27	34	0.96
36	2.41	36	0.87
38	2.47	38	1.01
40	2.76	40	0.98
42	2.88	42	1.13
44	3.13	44	1.14
46	3.03	46	1.41
48	4.14	48	1.55
50	3.45	50	1.32
52	3.52	52	1.38
54	3.88	54	1.37
56	3.84	56	1.35
58	4.06	58	1.4
60	4.17	60	1.42
62	4.13	62	1.51
64	4.46	64	1.59
66	4.56	66	1.5
68	4.74	68	1.55
70	5.09	70	1.64
72	5.08	72	1.73
74	5.43	74	1.85
76	5.3	76	1.84
78	5.36	78	1.79
80	5.75	80	1.86
82	5.62	82	1.92
84	5.59	84	2.013
86	6.05	86	2.017
88	6.22	88	2
90	6.38	90	1.99
92	6.54	92	2.03
94	6.5	94	2.04
96	6.64	96	2.23
98	6.53	98	2.31
100	7.2	100	2.29
102	7.15	102	2.25
104	7.43	104	2.44
106	7.38	106	2.39
108	7.23	108	2.49
110	7.74	110	2.46
112	7.28	112	2.52
114	7	114	2.6
116	7.74	116	2.56
118	7.87	118	2.64
120	7.98	120	2.87
122	7.89	122	2.98
124	8.71	124	3.16
126	9.23	126	3.29
128	8.96	128	3.48
130	9.57	130	3.59
132	10.44	132	3.9
134	10.6	134	3.92
136	11.63	136	4.16
138	11.55	138	4.3
140	12.72	140	4.47
142	12.34	142	5.03
144	12.43	144	4.84
146	13.11	146	5.21
148	12.81	148	5.36
150	13.9	150	5.31
152	13.86		
154	14.44		
156	14.84		
158	15.32		
160	15.38		
162	16.63		
164	16.62		
166	16.78		
168	17.6		
170	18.02		
172	18.07		
174	17.98		
176	19.34		
178	20.93		
180	20.2		
182	21.03		
184	21.23		
186	20.79		
188	21.62		
190	22.2		

ตารางที่ 4.2 แสดงผลการทดลองการควบคุมความเร็ว โดยการปรับค่า Digital Potentiometer

วิธีการเก็บผลการทดลองการควบคุมความเร็วโดยการปรับค่า Digital Potentiometer เริ่มจากการกำหนดค่า Digital Potentiometer ที่จะค่าเพื่อสังเกตความแตกต่างของความเร็วที่เปลี่ยน และจะเริ่มเก็บค่าความเร็วที่เมื่อรอดเริ่มเคลื่อนที่เป็นเวลา 60 วินาที ในไฟล์ .CSV จากนั้นนำค่าที่ได้มากราฟ และหาค่าเฉลี่ย โดยค่าที่นำมาหาค่าเฉลี่ย คือเส้นกราฟสีน้ำเงิน เป็นค่าความเร็วที่ผ่านการกรองสัญญาณ เช่นตัวอย่างรูปที่ 4.48 เมื่อปรับ Digital Potentiometer ที่ 36 ที่การเคลื่อนที่แบบ Backward ค่าเฉลี่ยความเร็วที่ได้ คือ 0.87 Km/hr

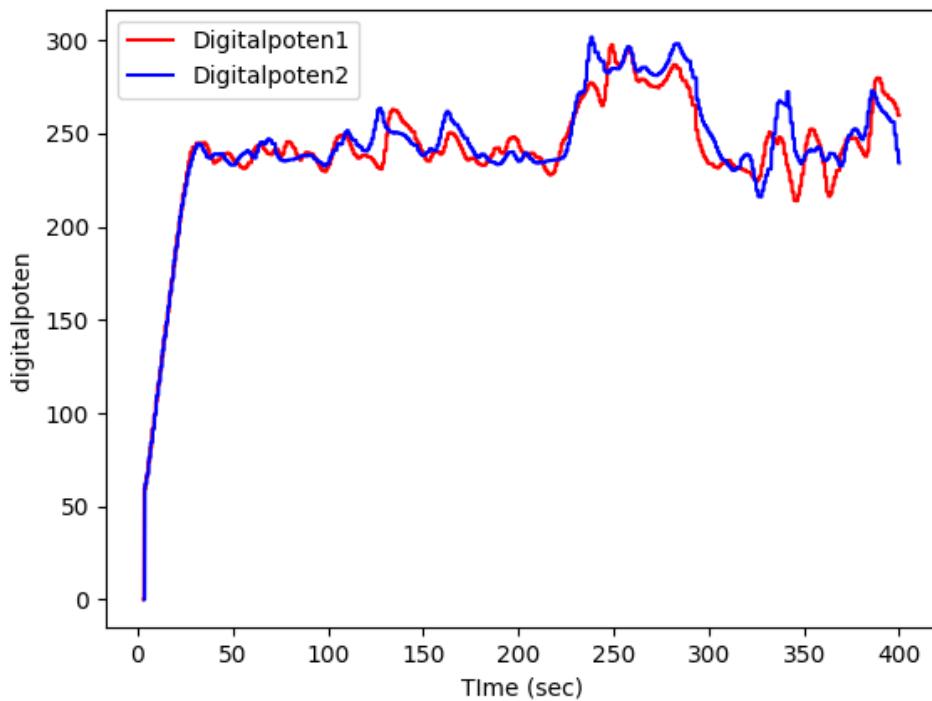


รูปที่ 4.48 ตัวอย่างการเก็บผลการทดลอง

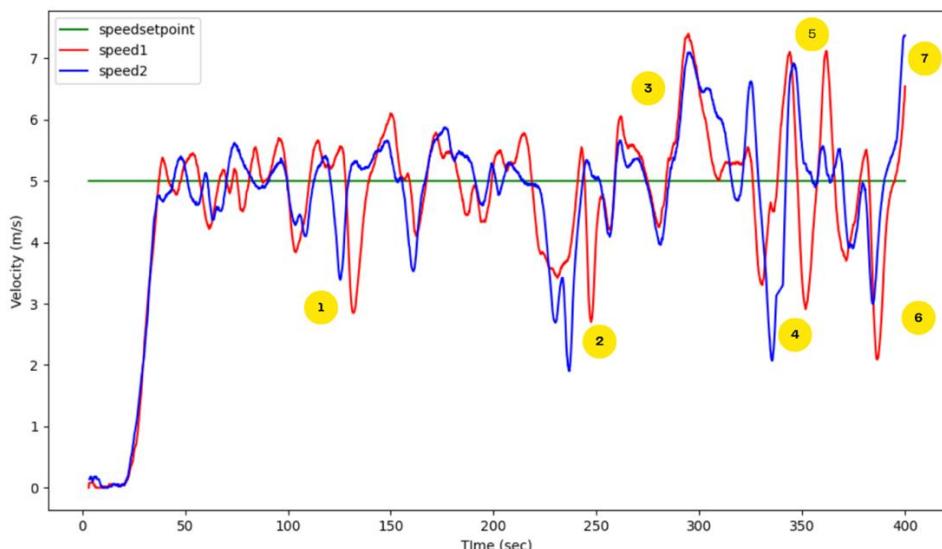
#### 4.2.3 ผลการทดลองการควบคุมความเร็วโดยการปรับค่า PID หรือการควบคุมความเร็วแบบระบบปิด

ในการทดลองดังกล่าวเป็นการปรับค่า Gain ที่เข้าสมการ PID เพื่อให้ได้ความเร็วที่ต้องการ และคงที่

#### 4.3.1 ผลการทดลองการเคลื่อนที่ด้วยความเร็ว 5 Km/hr คงที่รอบสนามฟุตบอล



รูปที่ 4.49 ค่า Digital Potentiometer ที่ใช้ในการควบคุม



รูปที่ 4.50 ค่าความเร็วที่ใช้ในการเคลื่อนที่ (km/hr)

### หมายเหตุ :

ชุดที่ 1 คือช่วงที่ขึ้นเนินบริเวณทาง โค้งเข้าโรงอาหาร (KFC)

ชุดที่ 2 คือ ช่วงที่รถขึ้นเนินหน้าโรงอาหาร (KFC)

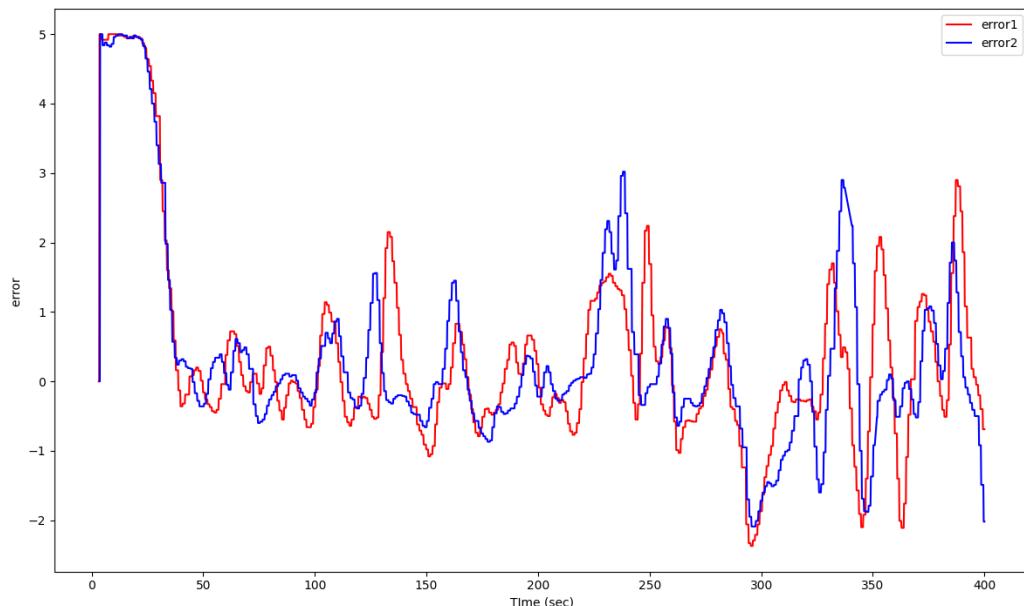
ชุดที่ 3 คือ ช่วงที่รถลงเนินหน้าโรงอาหาร (KFC)

ชุดที่ 4 คือ ช่วงที่รถขึ้นเนินบริเวณหน้าตีกครุ (Cb3)

ชุดที่ 5 คือ ช่วงที่รถลงเนินบริเวณหน้าตีกครุ (Cb3)

ชุดที่ 6 คือ ช่วงที่รถขึ้นเนินบริเวณหน้าตีกิวิศวกรรมศาสตร์ (Cb4)

ชุดที่ 7 คือ ช่วงที่รถลงเนินบริเวณหน้าตีกิวิศวกรรมศาสตร์ (Cb4)

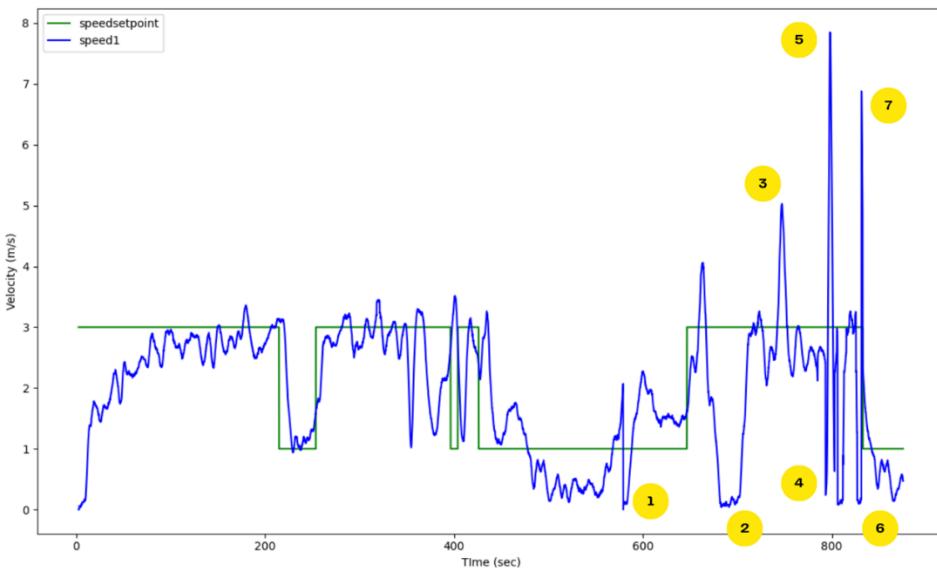


รูปที่ 4.51 ค่า Error ที่ได้จากการควบคุมความเร็วแบบ PID Controller

ตารางที่ 4.3 ตัวอย่างผลการทดลองการเคลื่อนที่ด้วยความเร็วคงที่ 5 Km/hr รอบสนามฟุตบอล

<b>Speed_time</b>	<b>Speed_velocity</b>	<b>Speed_Setpoint</b>	<b>Speed_adjust</b>	<b>Speed_error</b>
3.2	0.0	0.0	0.0	0.0
3.36	0.08	0.0	0.0	0.0
3.52	0.08	0.0	59.47999954223630	5.0
3.68	0.08	0.0	59.47999954223630	5.0
3.84	0.08	0.0	59.47999954223630	5.0
4.0	0.08	0.0	59.47999954223630	5.0
4.16	0.08	0.0	59.47999954223630	5.0
4.32	0.08	0.0	59.47999954223630	5.0
4.48	0.08	0.0	66.0964584350586	4.920000076293950
4.64	0.08	0.0	66.0964584350586	4.920000076293950
4.8	0.08	0.0	66.0964584350586	4.920000076293950
4.96	0.08	0.0	66.0964584350586	4.920000076293950
5.12	0.08	0.0	66.0964584350586	4.920000076293950
5.28	0.08	0.0	66.0964584350586	4.920000076293950
5.44	0.08	0.0	66.0964584350586	4.920000076293950
5.6	0.06	0.0	73.97024536132810	4.920000076293950
5.76	0.04	0.0	73.97024536132810	4.920000076293950
5.92	0.04	0.0	73.97024536132810	4.920000076293950
6.08	0.04	0.0	73.97024536132810	4.920000076293950
6.24	0.02	0.0	73.97024536132810	4.920000076293950
6.4	0.02	0.0	73.97024536132810	4.920000076293950
6.56	0.0	0.0	81.52735900878910	4.920000076293950
6.72	0.0	0.0	81.52735900878910	4.920000076293950
6.88	0.0	0.0	81.52735900878910	4.920000076293950
7.04	0.0	0.0	81.52735900878910	4.920000076293950
7.2	0.0	0.0	81.52735900878910	4.920000076293950
7.36	0.0	0.0	81.52735900878910	4.920000076293950
7.52	0.0	0.0	91.3827896118164	5.0
7.68	0.0	0.0	91.3827896118164	5.0
7.84	0.0	0.0	91.3827896118164	5.0
8.0	0.0	0.0	91.3827896118164	5.0
8.16	0.0	0.0	91.3827896118164	5.0
8.32	0.0	0.0	91.3827896118164	5.0

แต่เนื่องจากการควบคุมรถกอล์ฟให้เคลื่อนที่อัตโนมัติรอบสนามฟุตบอลด้วยความเร็ว 5 Km/hr ทั้งตลอดเส้นทาง มอเตอร์ควบคุมมุ่งเลี้ยวไม่สามารถทำงานได้ทันจังมีการปรับความเร็ว โดยจะแบ่งความเร็วออกเป็น 2 ช่วง คือช่วงที่เคลื่อนที่เป็นเส้นตรงจะวิ่งด้วยความเร็ว 3 Km/hr และช่วงที่มีการเลี้ยวโค้งจะเคลื่อนที่ด้วยความเร็ว 1 Km/hr เพื่อความแม่นยำในการควบคุมการทำงานของมอเตอร์ควบคุมมุ่งเลี้ยว



รูปที่ 4.52 ค่าความเร็วที่ใช้ในการเคลื่อนที่ (km/hr)

#### หมายเหตุ :

จุดที่ 1 คือช่วงที่ขึ้นเนินบริเวณทางโค้งเข้าโรงอาหาร (KFC)

จุดที่ 2 คือ ช่วงที่รถขึ้นเนินหน้าโรงอาหาร (KFC)

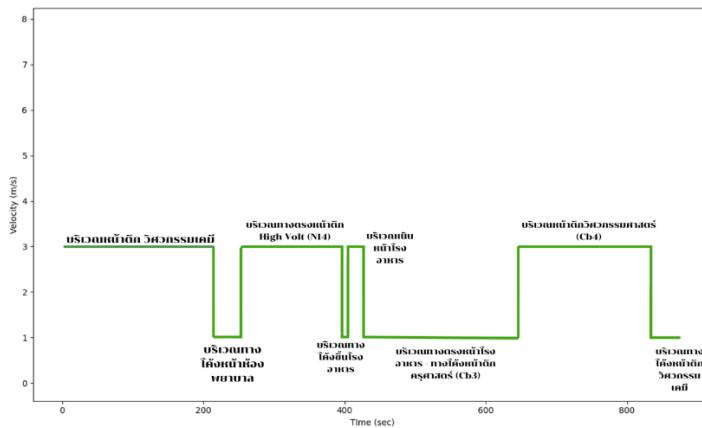
จุดที่ 3 คือ ช่วงที่รถลงเนินหน้าโรงอาหาร (KFC)

จุดที่ 4 คือ ช่วงที่รถขึ้นเนินบริเวณหน้าตึกครุ (Cb3)

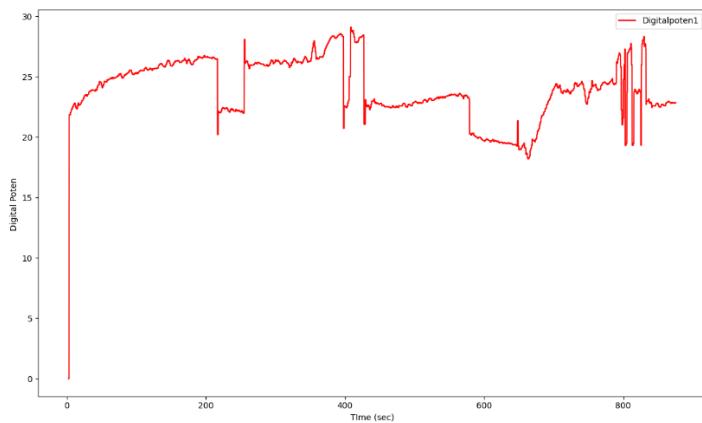
จุดที่ 5 คือ ช่วงที่รถลงเนินบริเวณหน้าตึกครุ (Cb3)

จุดที่ 6 คือ ช่วงที่รถขึ้นเนินบริเวณหน้าตึกวิศวกรรมศาสตร์ (Cb4)

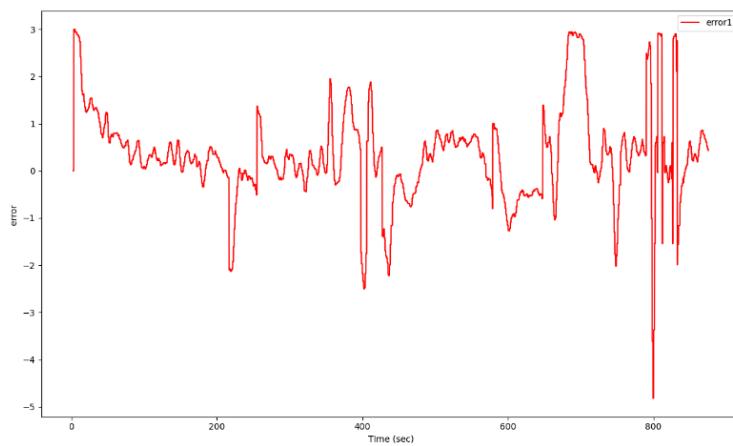
จุดที่ 7 คือ ช่วงที่รถลงเนินบริเวณหน้าตึกวิศวกรรมศาสตร์ (Cb4)



รูปที่ 4.53 ค่าความเร็ว Setpoint บริเวณรอบสนามฟุตบอล



รูปที่ 4.54 ค่า Digital Potentiometer ที่ใช้ในการควบคุม



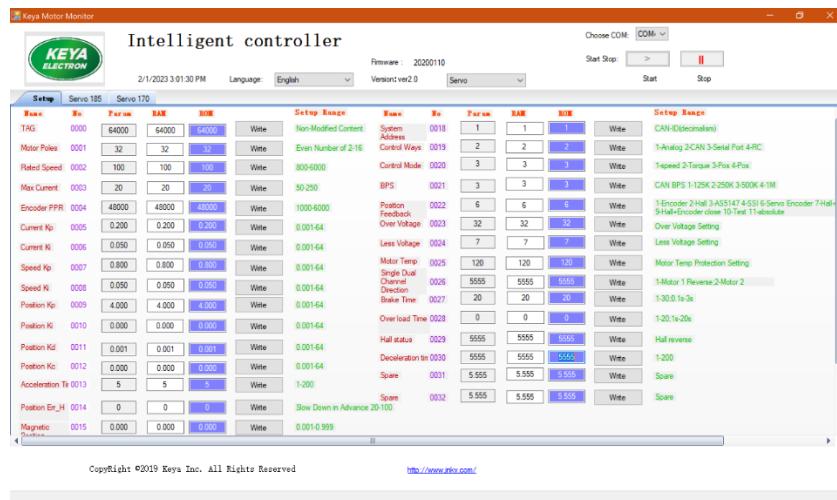
รูปที่ 4.55 ค่า Error ที่ได้จากการควบคุมความเร็วแบบ PID Controller

ตารางที่ 4.4 ตัวอย่างผลการทดลองการเคลื่อนที่ด้วยความเร็วคงที่ 3 Km/hr และ 1 Km/hr รอบสนามฟุตบอล

<b>Speed_time</b>	<b>Speed_velocity</b>	<b>Speed_Setpoint</b>	<b>Speed_adjust</b>	<b>Speed_error</b>
<b>2.56</b>	0.0	3.0	0.0	0.0
<b>2.72</b>	0.0	3.0	0.0	0.0
<b>2.88</b>	0.0	3.0	0.0	0.0
<b>3.04</b>	0.0	3.0	0.0	0.0
<b>3.2</b>	0.0	3.0	0.0	0.0
<b>3.36</b>	0.06	3.0	21.836000442504900	3.0
<b>3.52</b>	0.06	3.0	21.836000442504900	3.0
<b>3.68</b>	0.06	3.0	21.836000442504900	3.0
<b>3.84</b>	0.06	3.0	21.836000442504900	3.0
<b>4.0</b>	0.06	3.0	21.836000442504900	3.0
<b>4.16</b>	0.06	3.0	21.836000442504900	3.0
<b>4.32</b>	0.06	3.0	21.836000442504900	3.0
<b>4.48</b>	0.06	3.0	21.836000442504900	3.0
<b>4.64</b>	0.04	3.0	21.836000442504900	3.0
<b>4.8</b>	0.04	3.0	21.836000442504900	3.0
<b>4.96</b>	0.06	3.0	21.836000442504900	3.0
<b>5.12</b>	0.06	3.0	21.836000442504900	3.0
<b>5.28</b>	0.06	3.0	22.051759719848600	2.940000057220460
<b>5.44</b>	0.08	3.0	22.051759719848600	2.940000057220460
<b>5.6</b>	0.08	3.0	22.051759719848600	2.940000057220460
<b>5.76</b>	0.1	3.0	22.051759719848600	2.940000057220460
<b>5.92</b>	0.1	3.0	22.051759719848600	2.940000057220460
<b>6.08</b>	0.1	3.0	22.051759719848600	2.940000057220460
<b>6.24</b>	0.08	3.0	22.051759719848600	2.940000057220460
<b>6.4</b>	0.1	3.0	22.252880096435500	2.940000057220460
<b>6.56</b>	0.12	3.0	22.252880096435500	2.940000057220460
<b>6.72</b>	0.12	3.0	22.252880096435500	2.940000057220460
<b>6.88</b>	0.12	3.0	22.252880096435500	2.940000057220460
<b>7.04</b>	0.12	3.0	22.252880096435500	2.940000057220460
<b>7.2</b>	0.12	3.0	22.252880096435500	2.940000057220460
<b>7.36</b>	0.12	3.0	22.355281829834000	2.9000000953674300
<b>7.52</b>	0.12	3.0	22.355281829834000	2.9000000953674300
<b>7.68</b>	0.12	3.0	22.355281829834000	2.9000000953674300

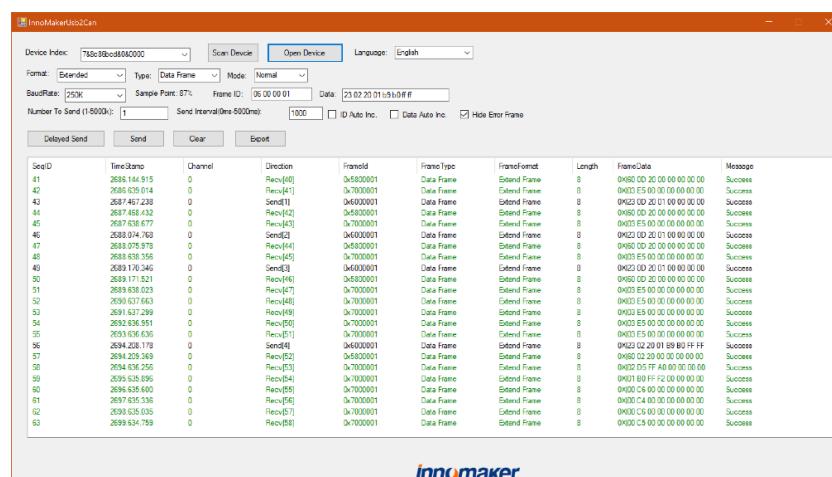
## 4.3 การทดลองและผลลัพธ์ของระบบควบคุมทิศทางและบังคับมุมเลี้ยว

### 4.3.1 การตั้งค่าการเชื่อมต่อ CANBus และ Steering Motor



รูปที่ 4.56 แสดงการเชื่อมต่อระหว่างตัวควบคุมกับซอฟต์แวร์

จากตั้งค่าพารามิเตอร์โดยเบียนที่ RAM ในช่อง No. 0018 (System address or node number of control) เป็น 1 ตั้งค่า RAM ในช่อง No. 0019 (Control Ways) เป็น 2 (CAN) ตั้งค่า RAM ในช่อง No. 0020 (BPS) เป็น 2 (250K) ตามลำดับเพื่อปรับค่าพารามิเตอร์เป็นการสื่อสารแบบ CAN แล้วทำการเชื่อมต่อ CANBUS เพื่อสั่งงานหรือรับค่ากับมอเตอร์พวงมาลัยผ่านโปรแกรม InnoMakerUSB2CAN



รูปที่ 4.57 แสดงการรับส่งข้อมูลผ่านโปรแกรม InnomakerUSB2CAN

การสั่งข้อมูลผ่านการสื่อสารแบบ CAN โดยทั่วไปนั้น การตั้งค่า baud rate เป็น 250Kb ตั้งค่า format เป็น extended ID การส่งรูปแบบข้อมูลจากตัวไปสูง โดยอยู่ในรูปฐานสิบหก ต้องทำการ Enable ผ่าน Data: 0x23 0x0D 0x20 0x01 0x00 0x00 0x00 0x00 ผ่าน Frame ID: 0x06000001 และจึงส่งข้อมูลที่ต้องการ สื่อสารผ่าน Data: 0x23 0x02 0x20 0x01 DATA\_HI DATA\_Hh DATA\_LI DATA\_Lh โดยทำการส่ง ข้อมูลสื่อสารเพื่อควบคุมมุมเลี้ยวของมอเตอร์พวงมาลัย ไปที่ Data: 0x23 0x02 0x20 0x01 0xB9 0xB0 0xFF 0xFF ซึ่งก็คือตำแหน่ง  $0xB9B0_{hex} = 47536_{dec}$  และ  $0xFFFF_{hex} = 65535_{dec}$  บอกทิศทางการหมุนคือทิศ ตามเข็ม (Clockwise) หากเป็นรูปแบบ CANOpen ข้อมูลจะเป็น quary mode และเป็นข้อมูล fixed heartbeat และส่งข้อมูลที่เกี่ยวกับ มีการตรวจสอบช่วงปีกของสัญญาณ 1000 ms (คำสั่งความเร็วจะถูก ส่งไปต่อเนื่องในช่วงเวลาที่อยกว่าหรือเท่ากับ 1000 ms การสั่งคืน Quary data เป็นข้อมูลเลขฐานสิบ (Hexadecimal data) ซึ่งจำเป็นต้องแปลงเป็นข้อมูลเลขฐานสิบ (Decimal data)

#### 4.3.2 การติดตั้ง Can-utils (cantools) ใน Ubuntu

คำสั่งสำหรับติดตั้ง can-utils คือ sudo apt-get install can-utils และทำการตั้งค่าพอร์ต โดยผ่านคำสั่ง ifconfig can0

```
pong@ck:~$ sudo apt-get install can-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
can-utils is already the newest version (2018.02.0-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
```

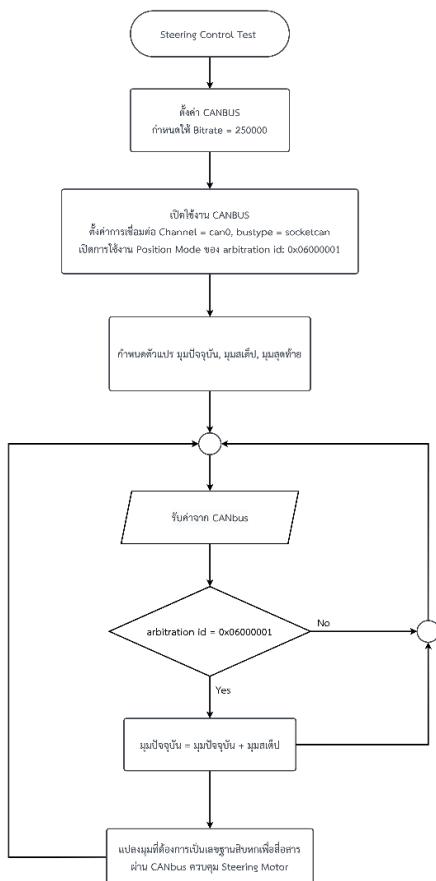
รูปที่ 4.58 การติดตั้ง CANtools ใน Ubuntu



### 4.3.3 การทดสอบการทำงานของมอเตอร์ผ่านการสื่อสารแบบ CAN

#### 4.3.3.1 การทดสอบการรับ-ส่งค่าให้มอเตอร์ผ่านการสื่อสารข้อมูล CAN

เป็นการเขียนคำสั่งเพื่อสั่งให้มอเตอร์ทำงานหรือ Enable ก่อน และจึงสามารถกำหนดตำแหน่งในการหมุนให้มอเตอร์ต่อไป โดยจะส่งไปยัง arbitration id 0x06000001 และจึงกำหนด data ที่ใช้ในการสั่งมอเตอร์ให้ทำงานตามโหมดการทำงานใดๆ โดยในที่นี้จะควบคุมการทำงานในโหมด Position Control ก็จะใช้ Data ตามที่ datasheet กำหนด คือ data: 0x23, 0x0d, 0x20, 0x01, 0x00, 0x00, 0x00, 0x00 เพื่อสั่งงานให้ Enable หรือเป็นการล้างข้อมูลเพื่อเปิดใช้งานก่อน โดยขั้นตอนการทำงานสำหรับขั้นตอนการแปลงมุมที่ต้องการให้เป็นเลขฐานสิบหกนั้น จะทำการกำหนดตัวแปรเพื่อใช้ในการแปลงค่ามุมตำแหน่งที่ได้รับเป็นองศาแล้วจะผ่านสมการคำนวนพัลส์ก่อน และจึงแปลงเป็นข้อมูลเลขฐานสิบหกในการส่งค่าเพื่อบังคับมอเตอร์ให้หมุนไปยังตำแหน่งที่ต้องการซึ่งแสดงดังรูปที่ 4.60 ผังงานสำหรับทดสอบการรับ-ส่งค่าให้มอเตอร์



รูปที่ 4.60 ผังงานสำหรับทดสอบการรับ-ส่งค่าให้มอเตอร์

### 4.3.4 การทดลองในการส่งและรับค่าตอบกลับจาก CAN ผ่านการควบคุมมอเตอร์พวงมาลัย มุ่งเลี้ยว

จากการทดลองการสื่อสารผ่าน CAN กับการควบคุมมอเตอร์พวงมาลัย พบว่ามอเตอร์นั้นมีข้อจำกัดการสื่อสารคือมุ่งทางไฟฟ้าจะมีค่าตั้งแต่ 0 ถึง 1000 โดยมีค่าต่ำสุดที่ -25000 และค่าสูงสุดที่ 25000 ดังนี้จากตารางที่ 4.5 จะเห็นว่ามีการนับจำนวนรอบเมื่อค่ามุ่งที่ตอบกลับมาไม่ค่าน้อยกว่าค่ามุ่งก่อนหน้า ซึ่งหมายถึงจะต้องบวกค่า 1000 เข้าไปในทุกๆ จำนวนรอบนั้นเอง แล้วจึงนำค่าที่ผ่านการคำนวณรอบนั้นมาหาผลต่างหรือในคอลัมน์ delta out1 จึงสามารถทราบค่าผลต่างเฉลี่ยอยู่ที่ 155.52 จึงนำไปหาค่าເອົາດີພຸດ มุ่งทางไฟฟ้าอุอกมาในคอลัมน์ angle output1 actual และเมื่อแปลงค่ากลับมาเป็นมุ่งที่เกิดขึ้นจริงจะได้ดังคอลัมน์ angle out actual ซึ่งจะเห็นว่าเกิดความคลาดเคลื่อนของมุ่งประมาณค่าเฉลี่ยที่ 124-125 องศา จากในคอลัมน์ angle difference และสามารถหาค่า angle error ได้ดังคอลัมน์ angle error และได้ทำการพล็อตกราฟความสัมพันธ์ระหว่างເອົາດີພຸດมุ่งทางທຽບແລະເອົາດີພຸດມุ่งที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลา และกราฟแสดงค่าความคลาดเคลื่อนเทียบกับเวลาดังรูปที่ 4.61 และ 4.62 ตามลำดับ

ตารางที่ 4.5 การทดสอบการควบคุมมอเตอร์ในรอบที่ 1

Steering Motor Control Test Round 1 (0° - 400°)											
Time (s)	Angle input (deg)	Angle output (hex)	Angle output (dec)	Round No.	Electrical Angle output	Electrical Angle output	Actual Angle output	Actual Angle output (deg)	Angle output theory	Angle difference (deg)	Error Angle
0	0.000	01aa	426.000	0.000	426.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	0225	549.000	0.000	549.000	44.280	44.280	2.847	155.529	155.529	0.715
2	20.000	03af	943.000	0.000	943.000	141.840	186.120	11.967	311.059	155.529	0.402
3	30.000	0178	376.000	1.000	1376.000	155.880	342.000	21.989	466.588	124.588	0.267
4	40.000	0328	808.000	1.000	1808.000	155.520	497.520	31.989	622.118	124.598	0.200
5	50.000	00ef	239.000	2.000	2339.000	155.160	652.680	41.965	777.647	124.967	0.161
6	60.000	029f	671.000	2.000	2571.000	155.520	808.200	51.964	933.177	124.977	0.134
7	70.000	0068	104.000	3.000	3104.000	155.880	964.080	61.987	1088.706	124.626	0.114
8	80.000	0218	536.000	3.000	3536.000	155.520	1119.600	71.986	1244.236	124.636	0.100
9	90.000	03c7	967.000	3.000	3967.000	155.160	1274.760	81.963	1399.765	125.005	0.089
10	100.000	0190	400.000	4.000	4400.000	155.880	1430.640	91.985	1555.295	124.655	0.080
11	110.000	033f	831.000	4.000	4831.000	155.160	1585.800	101.961	1710.824	125.024	0.073
12	120.000	0107	263.000	5.000	5263.000	155.520	1741.320	111.961	1866.354	125.034	0.067
13	130.000	0268	696.000	5.000	5696.000	155.880	1897.200	121.983	2021.883	124.683	0.062
14	140.000	0080	128.000	6.000	6128.000	155.520	2052.720	131.983	2177.413	124.693	0.057
15	150.000	0231	561.000	6.000	6561.000	155.880	2208.600	142.005	2332.942	124.342	0.053
16	160.000	03e1	993.000	6.000	6993.000	155.520	2364.120	152.005	2488.472	124.352	0.050
17	170.000	01a7	423.000	7.000	7423.000	154.800	2518.920	161.958	2644.001	125.081	0.047
18	180.000	0357	855.000	7.000	7855.000	155.520	2674.440	171.957	2799.531	125.091	0.045
19	190.000	0120	288.000	8.000	8288.000	155.880	2830.320	181.980	2955.060	124.740	0.042
20	200.000	02d0	720.000	8.000	8720.000	155.520	2985.840	191.979	3110.589	124.749	0.040
21	210.000	0097	151.000	9.000	9151.000	155.160	3141.000	201.955	3266.119	125.119	0.038
22	220.000	0248	584.000	9.000	9584.000	155.880	3296.880	211.978	3421.648	124.768	0.036
23	230.000	0011	17.000	10.000	10017.000	155.880	3452.760	222.000	3577.178	124.418	0.035
24	240.000	01c0	448.000	10.000	10448.000	155.160	3607.920	231.977	3732.707	124.787	0.033
25	250.000	03f6	879.000	10.000	10879.000	155.160	3765.080	241.953	3888.237	125.157	0.032
26	260.000	0138	312.000	11.000	11312.000	155.880	3918.960	251.975	4043.766	124.806	0.031
27	270.000	02e8	744.000	11.000	11744.000	155.520	4074.480	261.975	4199.296	124.816	0.030
28	280.000	0061	177.000	12.000	12177.000	155.880	4230.360	271.997	4354.825	124.465	0.029
29	290.000	025e	606.000	12.000	12606.000	154.440	4384.800	281.927	4510.355	125.555	0.028
30	300.000	0026	38.000	13.000	13038.000	155.520	4540.320	291.927	4665.884	125.564	0.027
31	310.000	01d8	472.000	13.000	13472.000	156.240	4696.560	301.972	4821.414	124.854	0.026
32	320.000	0386	902.000	13.000	13902.000	154.800	4851.360	311.925	4976.943	125.583	0.025
33	330.000	014d	333.000	14.000	14333.000	155.160	5006.520	321.902	5132.473	125.593	0.025
34	340.000	0300	768.000	14.000	14768.000	156.600	5163.120	331.971	5288.002	124.882	0.024
35	350.000	00c6	198.000	15.000	15198.000	154.800	5317.920	341.924	5443.532	125.612	0.023
36	360.000	0278	632.000	15.000	15632.000	156.240	5474.160	351.969	5599.061	124.501	0.022
37	370.000	0040	64.000	16.000	16064.000	155.520	5629.680	361.969	5754.591	124.911	0.022
38	380.000	01ef	495.000	16.000	16495.000	155.160	5784.840	371.945	5910.120	125.280	0.021
39	390.000	03a0	928.000	16.000	16928.000	155.880	5940.720	381.967	6065.649	124.929	0.021
40	400.000	0168	360.000	17.000	17360.000	155.520	6096.240	391.967	6221.179	124.939	0.020

จากตารางที่ 4.5 เป็นการสั่งงานและรับค่าในรอบที่ 1 จากการใช้ CANBUS ในการสื่อสารกับมอเตอร์พวงมาลัย โดยสั่งให้หมุนพวงมาลัยเริ่มต้นที่ละ 10 องศา เพิ่มขึ้นทีละ 10 องศา จนถึงหมุน 400 องศา และทำการคำนวณค่า เปรียบเทียบค่าที่เกิดขึ้นจริงกับค่าทางทฤษฎีขึ้น โดยจะอธิบายแต่ละคอลัมน์ได้ดังนี้

1. Time (s) เป็นเวลาที่เปลี่ยนแปลงไปทุกๆ 1 วินาที โดยเริ่มจากวินาทีที่ 0 จนถึงวินาทีที่ 40
2. Angle input (deg) เป็นมุมอินพุตที่ป้อนให้ระบบสั่งงานมอเตอร์พวงมาลัย โดยจะเริ่มตั้งแต่ 0 องศา เพิ่มขึ้นทีละ 10 องศา ไปจนถึงหมุน 400 องศา
3. Angle output (hex) เป็นค่ามุมआดั๊ฟต์ที่รับจาก CANBUS โดยแสดงเป็นเลขฐานสิบหก
4. Angle output (dec) เป็นค่ามุมआดั๊ฟต์ที่แปลงจากเลขฐานสิบหกเป็นเลขฐานสิบ ซึ่งได้จากค่าที่รับจาก CANBUS
5. Round No. เป็นจำนวนรอบของมอเตอร์ เนื่องจากข้อจำกัดของมอเตอร์ในโหมดการทำงานของ Heartbeat return คือ จะส่ง Receive data ผ่าน Return ID: 0x07000001 โดยสามารถควบคุมมุมทางไฟฟ้าตั้งแต่ 0 – 1000 ซึ่งหมายความว่าเมื่อรอบการหมุนของมุมมอเตอร์เกิน 1000 จะนับเป็นรอบที่ N+1 โดยที่ N คือจำนวนรอบ 0, 1, 2, 3, ..., N ที่มอเตอร์หมุนผ่านมุมทางไฟฟ้าที่ 1000 ไปแล้ว โดยจะเก็บค่า 1000 ไว้ โดยการคูณกับจำนวนรอบ ได้ดังสัมการ

$$[(1000 \times N) + \text{Angle output (dec)}] \quad (4.1)$$

จากตัวอย่างในตารางที่ 4.5 คอลัมน์ Angle output (dec) ณ วินาทีที่ 2 และ 3 พบว่าค่า Angle output (dec) มีค่า 943 และ 346 ตามลำดับ จะสามารถนับค่ามุม 943 เป็นรอบที่ 0 เนื่องจากยังไม่ผ่านค่ามุมทางไฟฟ้าที่ 1000 ดังนั้นค่ามุมทางไฟฟ้าจะเป็น  $(1000 \times N) + 943 = (1000 \times 0) + 943 = 943$  และที่ค่ามุม 346 นั้นจะถูกนำไปรอบที่ 1 หรือมีการเก็บค่า 1000 ไว้ ดังนั้นจะได้มุมทางไฟฟ้าเป็น  $(1000 \times N) + 346 = (1000 \times 1) + 346 = 1346$  ดังคอลัมน์ Electrical Angle output

6. Electrical Angle output คือค่ามุมทางไฟฟ้าที่แปลงออกมาหลังจากนับค่า Round No. รวมเข้าไปด้วย ดังตัวอย่างการคำนวณในข้อที่ 5) ซึ่งจะนำผลรวมของ  $[(1000 \times N) + \text{Angle output (dec)}]$  เข้าด้วยกันแล้วจะได้ผลลัพธ์ของมุมทางไฟฟ้าตามหลักการของมอเตอร์พวงมาลัยรุ่นนี้
7.  $\Delta$  Electrical Angle output คือผลต่างของ Electrical Angle output จะพบว่าในช่วงเริ่มต้นหรือวินาทีที่ 0 - 3 นั้นจะมีค่าผลต่างที่ไม่คงที่ เมื่อเทียบกับวินาทีที่ 4 เป็นต้นไปนั้นมุมทางไฟฟ้าจะอยู่ที่ค่าประมาณ 155 เป็นต้น

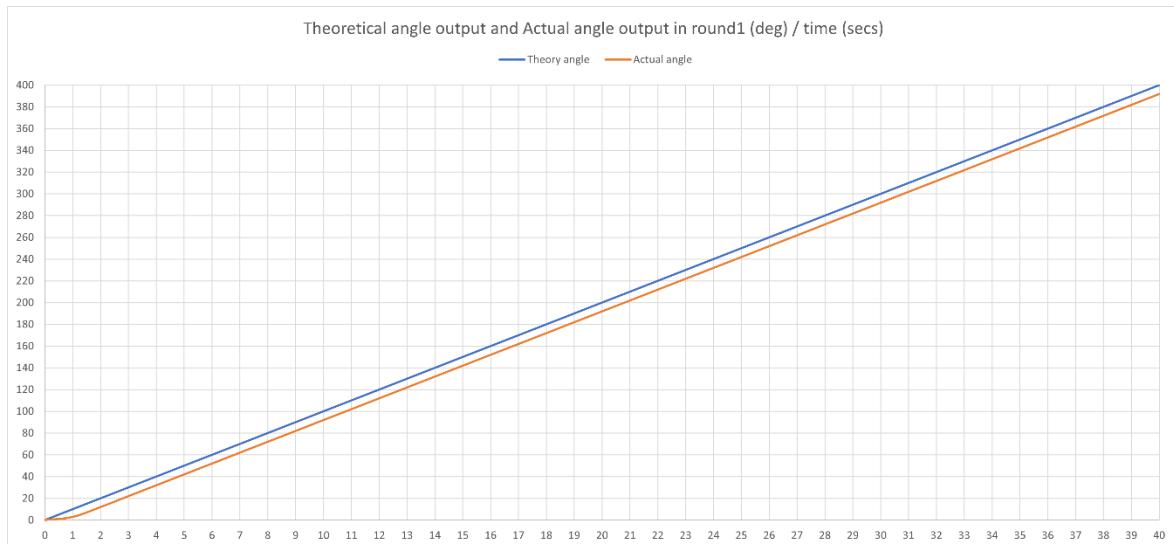
8. Actual Angle output คือผลรวมจากผลต่างมุมทางไฟฟ้าจากคลัมบ์  $\Delta$  Electrical Angle output ตัวอย่างเช่น ที่วินาทีที่ 0 จะได้ผลต่างมุมทางไฟฟ้าเป็น 0 ดังนั้นจะได้ Actual Angle output = 0 + 0 = 0 เนื่องจากเป็นค่าเริ่มต้นซึ่งยังไม่มีค่าผลต่างมุมทางไฟฟ้าก่อนหน้า และในวินาทีที่ 1 จะเกิดผลต่างมุมทางไฟฟ้าเป็น 44.280 ดังนั้นจะได้ค่า Actual Angle output = 44.280 + 0 = 44.280 เนื่องจากผลต่างมุมทางไฟฟ้าก่อนหน้ามีค่าเป็น 0 และตัวอย่างในวินาทีที่ 2 จะแสดงให้เห็นได้ชัดเจนยิ่งขึ้น เนื่องจากผลต่างมุมทางไฟฟ้ามีค่าเป็น 141.840 ดังนั้นค่า Actual Angle output = 141.480 + 44.280 = 186.120 โดยค่า 44.280 นั้นมากจากค่าผลต่างมุมทางไฟฟ้าก่อนหน้าหรือในวินาทีที่ 1 นั่นเอง
9. Actual Angle output (deg) คือค่าผลลัพธ์มุมในหน่วยองศา โดยคำนวณได้จากการนำค่า Actual Angle output / Average of Angle output per degree โดยค่า Average of Angle output หาได้จากการนำค่า  $\Delta$  Electrical Angle output ตั้งแต่วินาทีที่ 3 เป็นต้นไปนั้นมาหารเฉลี่ยเนื่องจากค่าผลต่างจะมีค่าประมาณเฉลี่ยอยู่ที่ 155 โดยเมื่อนำมาเฉลี่ยแล้วจะได้ค่าออกมาคือ  $155.5294737/10^\circ$  ดังนั้นใน 1 องศา จะมีค่าเป็น 15.55295 ตัวอย่างการคำนวณเช่น ในวินาทีที่ 1 Actual Angle output มีค่าเป็น 44.280 ดังนั้น Actual Angle output (deg) =  $44.280/15.55295 = 2.847^\circ$  เป็นต้น
10. Angle output theory คือค่ามุมผลลัพธ์ทางไฟฟ้าที่ได้จากการคำนวณทางทฤษฎีโดยอ้างอิงจากหลักการตาม Datasheet ของมอเตอร์พวงมาลัย ซึ่งจะนำมาเปรียบเทียบกับ Actual Angle Output ที่เกิดขึ้นจริงจากการสั่งการมอเตอร์พวงมาลัย โดยคำนวณได้จากการนำค่า Angle input (deg) x Average of Angle output per degree (โดยที่ Average of Angle output per degree = 15.55295) ตัวอย่างเช่น ในวินาทีที่ 1: Angle input (deg) มีค่าคือ  $10^\circ$  ดังนั้นจะได้ Angle output theory =  $10^\circ \times 15.55295 = 155.529^\circ$  หรือในวินาทีที่ 2: Angle input (deg) มีค่าคือ  $20^\circ$  ดังนั้นจะได้ Angle output theory =  $20^\circ \times 15.55295 = 311.059^\circ$  เป็นต้น
11. Angle difference (deg) คือผลต่างของ Angle output theory จะพบว่าในช่วงเริ่มต้นหรือวินาทีที่ 0 - 3 นั้นจะมีค่าผลต่างที่ไม่คงที่ที่ค่า 155.529 เมื่อเทียบกับวินาทีที่ 4 เป็นต้นไปนั้น มุมทางไฟฟ้าจะอยู่ที่ค่าประมาณ  $124.9247/10^\circ$  หรือมีค่าเท่ากับ  $12.4924/1^\circ$  เป็นต้น
12. Error Angle คือความคลาดเคลื่อนที่เปรียบเทียบระหว่างค่าผลลัพธ์มุมทางทฤษฎีกับค่าผลลัพธ์มุมที่เกิดขึ้นจริงจากมอเตอร์ ซึ่งคำนวณได้จาก

$$\text{Error Angle} = \left| \frac{\text{Angle output theory} - \text{Actual Angle output}}{\text{Angle output theory}} \right| \quad (4.2)$$

ตัวอย่างในวินาทีที่ 1 ค่า Angle output theory = 155.529, Actual Angle output = 44.280

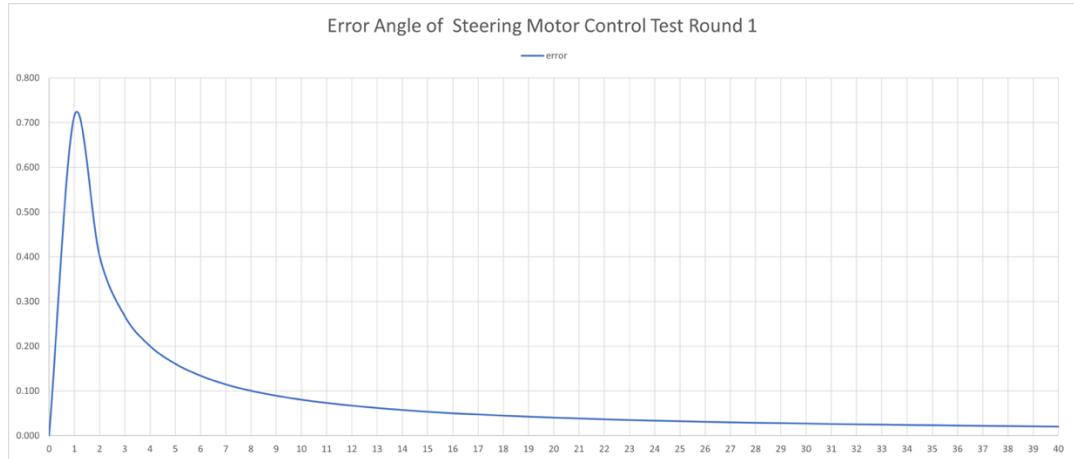
$$\text{Error Angle} = \left| \frac{155.529 - 44.280}{155.529} \right| = 0.715 \quad (4.3)$$

ดังนั้นค่าความคลาดเคลื่อนที่เกิดขึ้นคือ 0.715 ซึ่งมีค่าเข้าใกล้ 1 หรือมีความคลาดเคลื่อนสูงกว่าในวินาที ถัดไปจะพบว่าค่าความคลาดเคลื่อนจะค่อยๆ ลดลงนั่นเอง



รูปที่ 4.61 กราฟความสัมพันธ์ระหว่างเอาร์พุตมุมทางทฤษฎีและมุมที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาในรอบที่ 1

จากรูปที่ 4.61 เป็นกราฟความสัมพันธ์ระหว่างเอาร์พุตมุมทางทฤษฎีและมุมที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลา จะสังเกตุได้ว่าช่วงเริ่มต้นที่วินาทีที่ 0 ถึง 1 เอาร์พุตมุมที่เกิดขึ้นจริงจะมีค่าลดลงและจะมีค่าคงที่โดยจะมีค่ามุนน้อยกว่าเมื่อเทียบกับเอาร์พุตมุมทางทฤษฎี

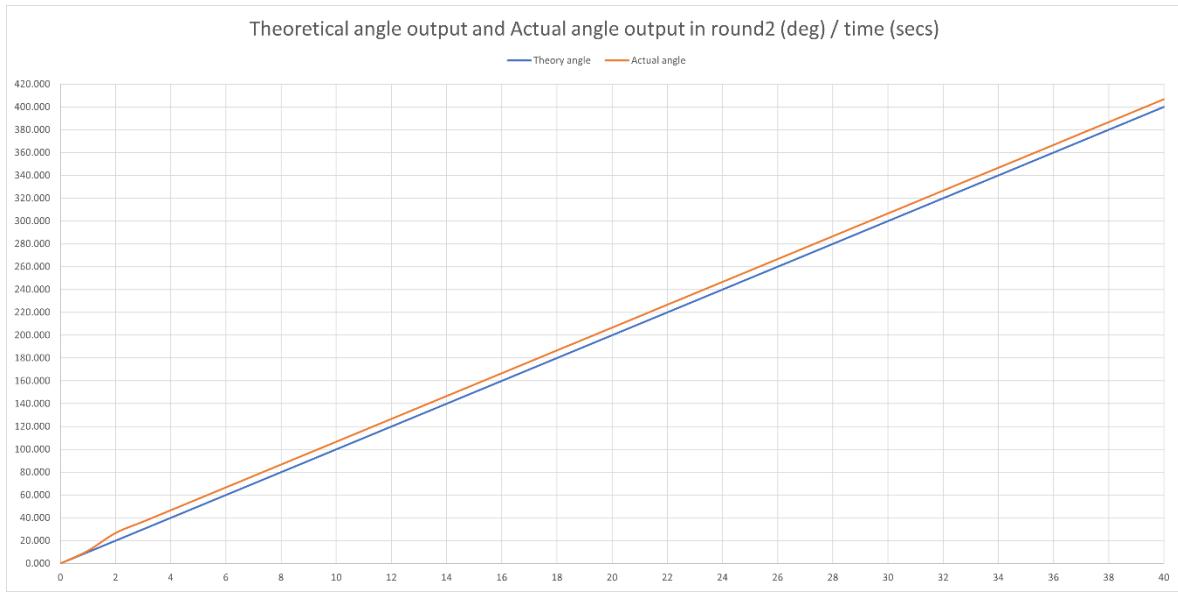


รูปที่ 4.62 กราฟแสดงค่าความคลาดเคลื่อนเทียบกับเวลาในรอบที่ 1

จากราฟในรูปที่ 4.62 เป็นกราฟแสดงค่าความคลาดเคลื่อนที่เกิดขึ้นจากເອົາຕີພຸດມູນທີ່ເກີດຂຶ້ນຈິງກັນນູນທາງທາງຄູນຄູ່ເທືບກັນເວລາ ຈະພວ່ານຳໃນຫຼວງເຮົ້າໃນວິນາທີ່ 0 - 1 ນັ້ນຄ່າຈະຄລາດເຄລື່ອນສູງທີ່ສຸດແລ້ວໜີ້ຈາກນັ້ນຈະກ່ອຍາ ລດຄງຈນເຂົ້າໄກສັກ່າ 0 ຕ່ອງໄປຈະແສດງຂໍ້ມູນການທົດສອບການຄວບຄຸມມອເຕອຣີໃນรอบທີ່ 2 - 6

ตารางที่ 4.6 การທົດສອບການຄວບຄຸມມອເຕອຣີໃນรอบທີ່ 2

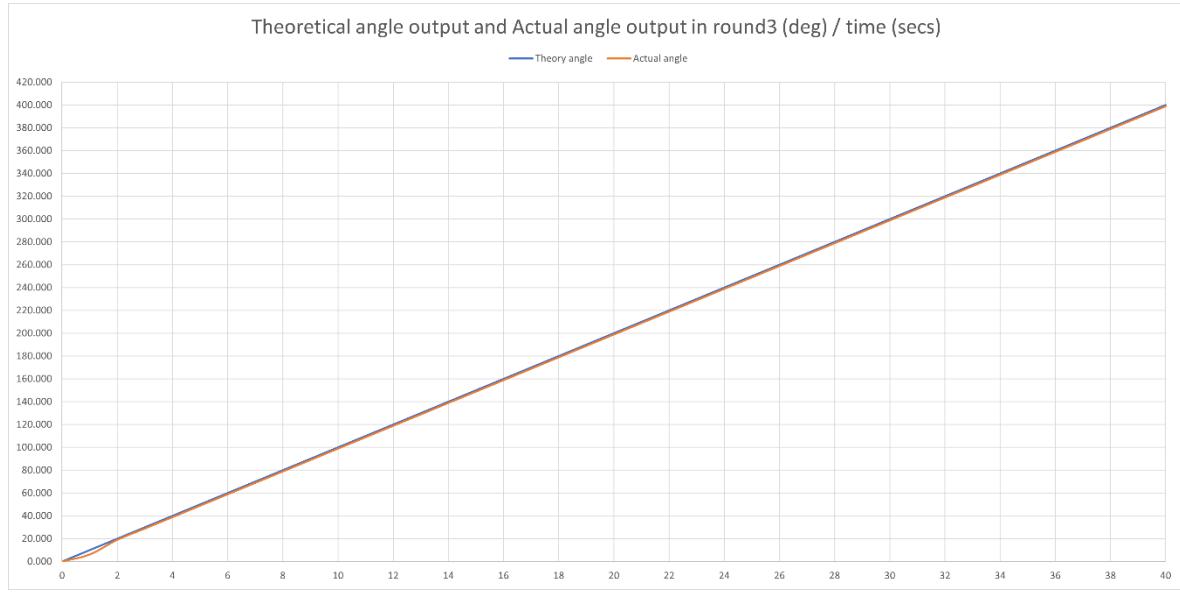
Steering Motor Control Test Round 2 (0° - 400°)											
Time (s)	Angle input (deg)	Angle output (hex)	Angle output (dec)	Round No.	Electrical Angle output	Digital Angle output	Actual Angle output	Actual Angle output (deg)	Angle output theory	Angle difference (deg)	Error Angle
0	0.000	0316	790.000	0.000	790.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	0111	273.000	1.000	1273.000	173.880	173.880	11.180	155.530	155.530	0.118
2	20.000	03b0	944.000	1.000	1944.000	241.560	415.440	25.711	311.059	155.530	0.336
3	30.000	0176	374.000	2.000	2374.000	154.800	570.240	36.664	466.589	-103.652	0.222
4	40.000	0326	806.000	2.000	2806.000	155.520	725.760	46.664	622.118	-103.642	0.167
5	50.000	00ee	238.000	3.000	3238.000	155.520	881.280	56.663	777.648	-103.633	0.133
6	60.000	029e	670.000	3.000	3670.000	155.520	1036.800	66.663	933.177	-103.623	0.111
7	70.000	0066	102.000	4.000	4102.000	155.520	1192.320	76.662	1088.707	-103.614	0.095
8	80.000	0217	535.000	4.000	4535.000	155.520	1348.200	86.685	1244.236	-103.564	0.084
9	90.000	03c7	967.000	4.000	4967.000	155.520	1503.720	96.684	1399.766	-103.555	0.074
10	100.000	018e	398.000	5.000	5398.000	155.520	1658.880	106.660	1555.295	-103.585	0.067
11	110.000	033e	830.000	5.000	5830.000	155.520	1814.400	116.660	1710.825	-103.576	0.061
12	120.000	0106	262.000	6.000	6262.000	155.520	1969.920	126.659	1866.354	-103.566	0.055
13	130.000	02b7	695.000	6.000	6695.000	155.520	2125.800	136.681	2021.884	-103.517	0.051
14	140.000	007f	127.000	7.000	7127.000	155.520	2281.320	146.681	2177.413	-103.507	0.048
15	150.000	022e	558.000	7.000	7558.000	155.520	2436.480	156.657	2332.943	-103.538	0.044
16	160.000	03de	990.000	7.000	7990.000	155.520	2592.000	166.656	2488.472	-103.528	0.042
17	170.000	01a6	422.000	8.000	8422.000	155.520	2747.520	176.656	2644.002	-103.519	0.039
18	180.000	0356	854.000	8.000	8854.000	155.520	2903.040	186.655	2799.531	-103.509	0.037
19	190.000	011d	285.000	9.000	9285.000	155.520	3058.200	196.632	2955.061	-103.439	0.035
20	200.000	02cd	717.000	9.000	9717.000	155.520	3213.720	206.631	3110.590	-103.430	0.033
21	210.000	0095	149.000	10.000	10149.000	155.520	3369.240	216.630	3266.120	-103.421	0.032
22	220.000	0247	583.000	10.000	10583.000	155.520	3525.480	226.676	3421.649	-103.431	0.030
23	230.000	000e	14.000	11.000	11014.000	155.520	3680.640	236.652	3577.179	-103.461	0.029
24	240.000	01be	446.000	11.000	11446.000	155.520	3836.160	246.652	3732.708	-103.452	0.028
25	250.000	036e	878.000	11.000	11878.000	155.520	3991.680	256.651	3888.238	-103.442	0.027
26	260.000	0136	310.000	12.000	12310.000	155.520	4147.200	266.650	4043.767	-103.433	0.026
27	270.000	02e7	743.000	12.000	12743.000	155.520	4303.080	276.673	4199.297	-103.784	0.025
28	280.000	00ae	174.000	13.000	13174.000	155.520	4458.240	286.649	4354.826	-103.414	0.024
29	290.000	025e	606.000	13.000	13606.000	155.520	4613.760	296.649	4510.356	-103.404	0.023
30	300.000	0026	38.000	14.000	14038.000	155.520	4769.280	306.648	4665.885	-103.395	0.022
31	310.000	01d4	468.000	14.000	14468.000	154.800	4924.080	316.601	4821.415	-102.665	0.021
32	320.000	0386	902.000	14.000	14902.000	156.240	5080.320	326.647	4976.944	-103.376	0.021
33	330.000	014f	335.000	15.000	15335.000	155.880	5236.200	336.669	5132.474	-103.726	0.020
34	340.000	02fd	765.000	15.000	15765.000	154.800	5391.000	346.622	5288.003	-102.997	0.019
35	350.000	00c4	196.000	16.000	16196.000	155.160	5546.160	356.599	5443.533	-102.628	0.019
36	360.000	0274	628.000	16.000	16528.000	155.520	5701.680	366.598	5599.062	-102.618	0.018
37	370.000	003e	62.000	17.000	17062.000	155.240	5857.920	376.644	5754.592	-103.328	0.018
38	380.000	01ed	493.000	17.000	17493.000	155.160	6013.080	386.620	5910.121	-102.959	0.017
39	390.000	039e	926.000	17.000	17926.000	155.880	6162.960	396.642	6065.651	-103.309	0.017
40	400.000	0166	358.000	18.000	18358.000	155.520	6324.480	406.642	6221.180	-103.300	0.017



**รูปที่ 4.63** กราฟความสัมพันธ์ระหว่างເອົາຫຼຸດມູນຖາງທຸຍ້ມີແລ້ມູນທີ່ເກີດບັນຈິງຈາກນອເຕອຣ໌ເຖິງກັນເວລາໃນຮອບທີ່ 2

**ตารางที่ 4.7** การทดสอบการควบคุมນອເຕອຣ໌ໃນຮອບທີ່ 3

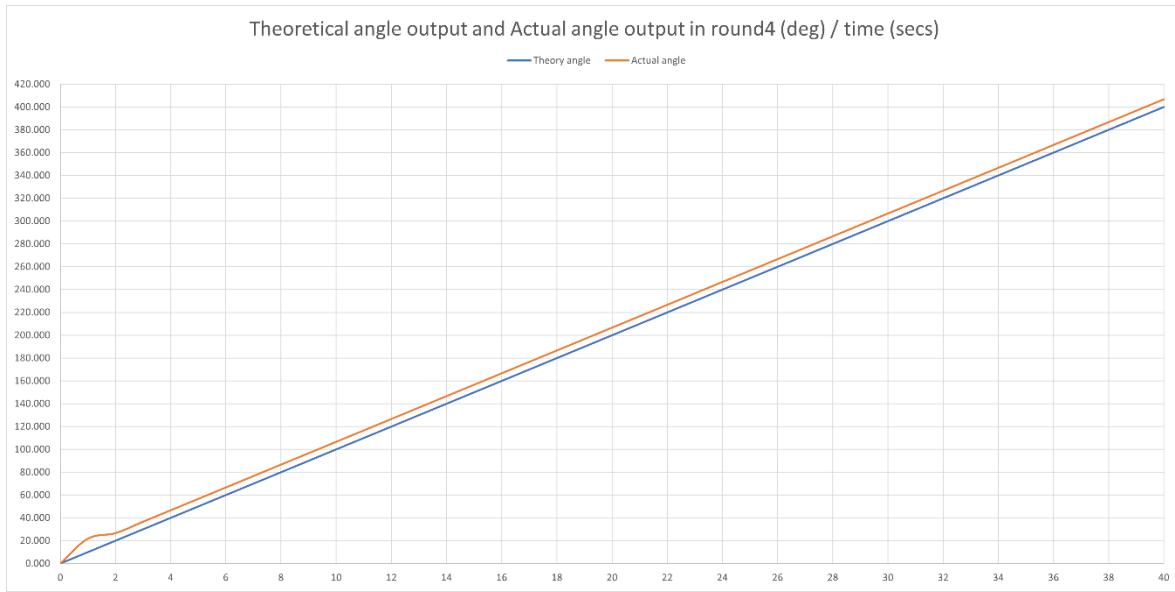
Steering Motor Control Test Round 3 (0° - 400°)											
Time (s)	Angle input (deg)	Angle output (hex)	Angle output (dec)	Round No.	Electrical Angle output	Digital Angle output	Actual Angle output	Actual Angle output (deg)	Angle output theory	Angle difference (deg)	Error Angle
0	0.000	0081	129.000	0.000	129.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	018b	395.000	0.000	395.000	95.760	95.760	95.760	155.530	155.530	0.384
2	20.000	03b6	950.000	0.000	950.000	199.800	295.560	199.003	311.059	311.059	0.050
3	30.000	0179	377.000	1.000	1377.000	153.720	449.280	28.887	466.589	17.309	0.037
4	40.000	0328	808.000	1.000	1808.000	155.160	604.440	38.863	622.118	17.678	0.028
5	50.000	00f0	240.000	2.000	240.000	155.520	759.960	48.863	777.648	17.688	0.023
6	60.000	02a1	673.000	2.000	2673.000	155.880	915.840	58.885	933.177	17.337	0.019
7	70.000	0069	105.000	3.000	3105.000	155.520	1071.360	68.885	1088.707	17.347	0.016
8	80.000	021a	538.000	3.000	3538.000	155.880	1227.720	78.907	1244.236	16.996	0.014
9	90.000	03ca	970.000	3.000	3970.000	155.520	1382.760	88.907	1399.766	17.006	0.012
10	100.000	0192	402.000	4.000	4402.000	155.520	1538.280	98.906	1555.295	17.015	0.011
11	110.000	0341	833.000	4.000	4833.000	155.160	1693.440	108.882	1710.825	17.385	0.010
12	120.000	0109	265.000	5.000	5265.000	155.520	1848.960	118.882	1866.354	17.394	0.009
13	130.000	0269	697.000	5.000	5697.000	155.520	2004.480	128.881	2021.884	17.404	0.009
14	140.000	0083	131.000	6.000	6131.000	156.240	2160.720	138.927	2177.413	16.693	0.008
15	150.000	0230	560.000	6.000	6560.000	154.440	2315.160	148.857	2332.943	17.782	0.008
16	160.000	03e1	993.000	6.000	6993.000	155.880	2471.040	158.879	2488.472	17.432	0.007
17	170.000	01a9	425.000	7.000	7425.000	155.520	2526.560	168.879	2644.002	17.441	0.007
18	180.000	035a	858.000	7.000	7858.000	155.880	2782.440	178.901	2799.531	17.091	0.006
19	190.000	0122	290.000	8.000	8290.000	155.520	2937.960	188.900	2955.061	17.101	0.006
20	200.000	02d2	722.000	8.000	8722.000	155.520	3093.480	198.900	3110.590	17.110	0.006
21	210.000	009a	154.000	9.000	9154.000	155.520	3249.000	208.899	3266.120	17.119	0.005
22	220.000	024a	586.000	9.000	9586.000	155.520	3404.520	218.899	3421.649	17.129	0.005
23	230.000	0011	17.000	10.000	10017.000	155.160	3559.680	228.875	3577.179	17.499	0.005
24	240.000	01c2	450.000	10.000	10450.000	155.880	3715.560	238.897	3732.708	17.148	0.005
25	250.000	0371	881.000	10.000	10881.000	155.160	3870.720	248.874	3888.238	17.517	0.005
26	260.000	0139	313.000	11.000	11313.000	155.520	4026.240	258.873	4043.767	17.527	0.004
27	270.000	02ea	746.000	11.000	11746.000	155.880	4182.120	268.896	4199.297	17.176	0.004
28	280.000	00b1	177.000	12.000	12177.000	155.160	4337.280	278.872	4354.826	17.546	0.004
29	290.000	0262	610.000	12.000	12610.000	155.880	4493.160	288.894	4510.356	17.195	0.004
30	300.000	0028	40.000	13.000	13040.000	154.800	4647.960	298.847	4665.885	17.925	0.004
31	310.000	0149	473.000	13.000	13473.000	155.880	4803.840	308.870	4821.415	17.574	0.004
32	320.000	0389	905.000	13.000	13905.000	155.520	4959.360	318.869	4976.944	17.584	0.004
33	330.000	0150	336.000	14.000	14336.000	155.160	5114.520	328.846	5132.474	17.953	0.003
34	340.000	0300	768.000	14.000	14768.000	155.520	5270.040	338.845	5288.003	17.963	0.003
35	350.000	00c9	201.000	15.000	15201.000	155.880	5425.920	348.868	5443.533	17.612	0.003
36	360.000	027a	634.000	15.000	15634.000	155.880	5581.800	358.890	5599.062	17.262	0.003
37	370.000	0041	65.000	16.000	16065.000	155.160	5736.960	368.866	5754.592	17.631	0.003
38	380.000	01f2	498.000	16.000	16498.000	155.880	5892.840	378.889	5910.121	17.281	0.003
39	390.000	03a1	929.000	16.000	16929.000	155.160	6048.000	388.865	6065.651	17.650	0.003
40	400.000	0169	361.000	17.000	17361.000	155.520	6203.520	398.865	6221.180	17.660	0.003



**รูปที่ 4.64** กราฟความสัมพันธ์ระหว่างເອົາຫຼຸດມູນທາງທາງໝາຍຸແລະມູນທີ່ເກີດບັນຈິງຈາກນອເຕອຣ໌ທີ່ຢືນກັນເວລາໃນรอบທີ່ 3

**ตารางที่ 4.8** การทดสอบการควบคุมນອເຕອຣ໌ໃນรอบທີ່ 4

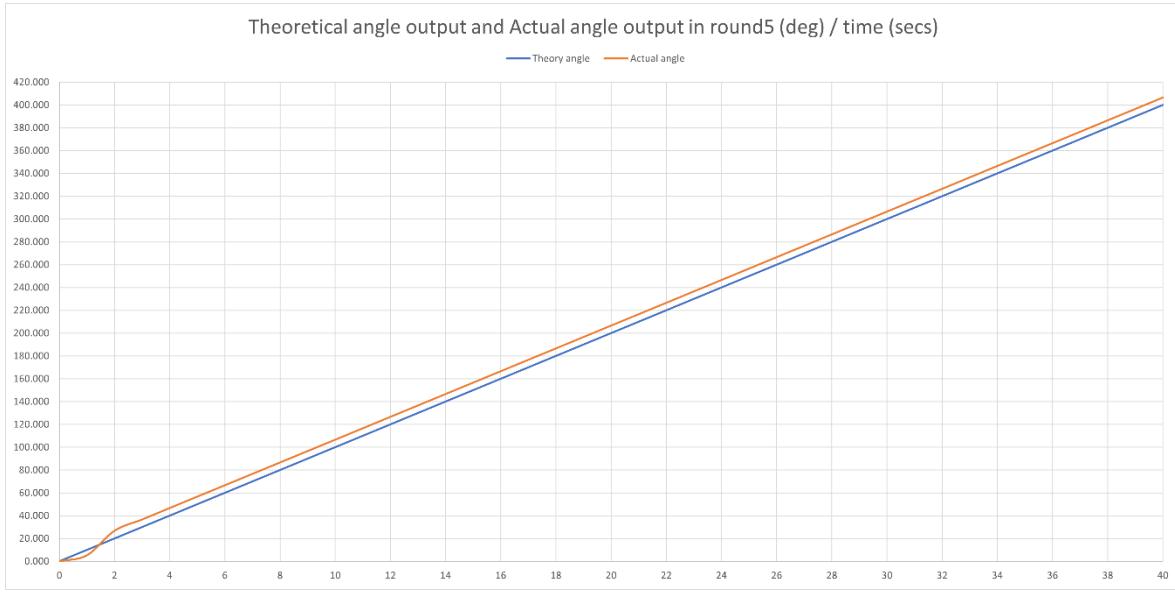
Steering Motor Control Test Round 4 (0° - 400°)											
Time (s)	Angle input (deg)	Angle output (hex)	Angle output (dec)	Round No.	Electrical Angle output	Delta electrical Angle output	Actual Angle output	Actual Angle output (deg)	Angle output theory	Angle difference (deg)	Error Angle
0	0.000	031a	794.000	0.000	794.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	02e0	736.000	1.000	1736.000	339.120	21.804	155.530	155.530	1.180	
2	20.000	03b1	945.000	1.000	1945.000	75.240	26.642	311.059	155.530	0.332	
3	30.000	0179	377.000	2.000	2377.000	155.520	56.880	36.641	466.589	-103.292	0.221
4	40.000	0329	809.000	2.000	2809.000	155.520	725.400	46.641	622.118	-103.282	0.166
5	50.000	00f1	241.000	3.000	3241.000	155.520	880.920	56.640	777.648	-103.273	0.133
6	60.000	02a1	673.000	3.000	3673.000	155.520	1036.440	66.639	933.177	-103.263	0.111
7	70.000	0069	105.000	4.000	4105.000	155.520	1191.960	76.639	1088.707	-103.254	0.095
8	80.000	021a	538.000	4.000	4538.000	155.520	1347.840	86.661	1244.236	-103.604	0.083
9	90.000	03c9	969.000	4.000	4969.000	155.520	1503.000	96.638	1399.766	-103.235	0.074
10	100.000	0191	401.000	5.000	5401.000	155.520	1658.520	106.637	1555.295	-103.225	0.066
11	110.000	0340	832.000	5.000	5832.000	155.160	1813.680	116.613	1710.825	-102.856	0.060
12	120.000	0109	265.000	6.000	6265.000	155.880	1969.560	126.636	1866.354	-103.206	0.055
13	130.000	0269	697.000	6.000	6697.000	155.520	2125.080	136.635	2021.884	-103.197	0.051
14	140.000	0081	129.000	7.000	7129.000	155.520	2280.600	146.635	2177.413	-103.187	0.047
15	150.000	0230	560.000	7.000	7560.000	155.160	2435.760	156.611	2332.943	-102.818	0.044
16	160.000	03e1	993.000	7.000	7993.000	155.880	2591.640	166.633	2488.472	-103.168	0.041
17	170.000	01e9	425.000	8.000	8425.000	155.520	2747.160	176.633	2644.002	-103.159	0.039
18	180.000	0358	856.000	8.000	8886.000	155.160	2902.320	186.609	2799.531	-102.789	0.037
19	190.000	0121	289.000	9.000	9289.000	155.880	3058.200	196.632	2955.061	-103.140	0.035
20	200.000	02d1	721.000	9.000	9721.000	155.520	3213.770	206.631	3110.590	-103.130	0.033
21	210.000	0098	152.000	10.000	10152.000	155.160	3368.880	216.607	3266.120	-102.761	0.031
22	220.000	0249	585.000	10.000	10585.000	155.880	3524.760	226.630	3421.649	-103.111	0.030
23	230.000	0011	17.000	11.000	11017.000	155.520	3680.280	236.629	3577.179	-103.102	0.029
24	240.000	01c4	452.000	11.000	11452.000	156.600	3836.880	246.689	3732.708	-104.172	0.028
25	250.000	0370	889.000	11.000	11889.000	154.080	3999.960	256.605	3888.238	-102.723	0.026
26	260.000	013a	314.000	12.000	12314.000	156.240	4147.200	266.650	4043.767	-103.433	0.026
27	270.000	02e9	745.000	12.000	12745.000	155.160	4302.360	276.627	4199.297	-103.064	0.025
28	280.000	00b1	177.000	13.000	13177.000	155.520	4457.880	286.626	4354.826	-103.054	0.024
29	290.000	0262	610.000	13.000	13610.000	155.880	4613.760	296.649	4510.356	-103.405	0.023
30	300.000	002a	42.000	14.000	14042.000	155.520	4769.280	306.648	4665.885	-103.395	0.022
31	310.000	01d8	472.000	14.000	14472.000	154.800	4924.080	316.601	4821.415	-102.665	0.021
32	320.000	0388	904.000	14.000	14904.000	155.520	5079.600	326.600	4976.944	-102.656	0.021
33	330.000	0151	337.000	15.000	15337.000	155.880	5235.480	336.623	5132.474	-103.006	0.020
34	340.000	0301	769.000	15.000	15769.000	155.520	5391.000	346.622	5288.003	-102.997	0.019
35	350.000	00c9	201.000	16.000	16201.000	155.520	5546.520	356.622	5443.533	-102.987	0.019
36	360.000	027b	635.000	16.000	16635.000	156.240	5702.760	366.667	5599.062	-103.698	0.019
37	370.000	0042	66.000	17.000	17066.000	155.160	5857.920	376.644	5754.592	-103.328	0.018
38	380.000	01f2	498.000	17.000	17498.000	155.520	6013.440	386.643	5910.121	-103.319	0.017
39	390.000	03a3	931.000	17.000	17931.000	155.880	6169.320	396.666	6065.651	-103.669	0.017
40	400.000	016a	362.000	18.000	18362.000	155.160	6324.480	406.642	6221.180	-103.300	0.017



รูปที่ 4.65 กราฟความสัมพันธ์ระหว่างເອົາຫຼຸດມູນທາງທາງໝາຍງືແລະມູນທີ່ເກີດບັນຈິງຈາກນອເຕອຣ໌ເຖິງກັນເວລາໃນຮອບທີ່ 4

ตารางที่ 4.9 การทดสอบการควบคุมນອເຕອຣ໌ໃນຮອບທີ່ 5

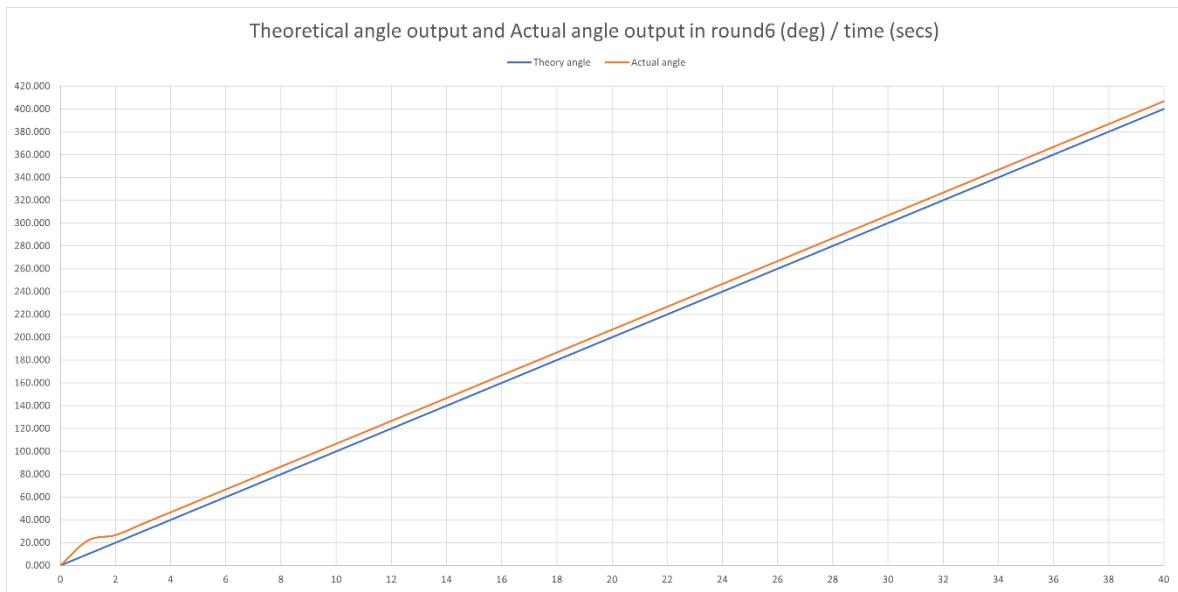
Steering Motor Control Test Round 5 (0° - 400°)											
Time (s)	Angle input (deg)	Angle output (hex)	Angle output (dec)	Round No.	Electrical Angle output	Electrical Angle output	Actual Angle output	Actual Angle output (deg)	Angle output theory	Angle difference (deg)	Error Angle
0	0.000	0319	793.000	0.000	793.000	0.000	793.000	0.000	0.000	0.000	0.000
1	10.000	0014	20.000	1.000	1020.000	81.720	81.720	5.254	155.530	155.530	0.475
2	20.000	0384	948.000	1.000	1948.000	334.080	415.800	26.734	311.059	155.530	0.337
3	30.000	0177	375.000	2.000	2375.000	153.720	569.520	36.618	466.589	-102.932	0.221
4	40.000	0323	810.000	2.000	2810.000	156.600	726.120	46.687	622.118	-104.002	0.167
5	50.000	0092	242.000	3.000	3242.000	155.520	881.640	56.686	777.648	-103.993	0.134
6	60.000	0241	673.000	3.000	3673.000	155.160	1036.800	66.663	933.177	-103.623	0.111
7	70.000	0069	105.000	4.000	4105.000	155.520	1192.320	76.662	1088.707	-103.614	0.095
8	80.000	0218	536.000	4.000	4536.000	155.160	1347.480	86.638	1244.236	-103.244	0.083
9	90.000	0383	968.000	4.000	4968.000	155.520	1503.000	96.638	1399.766	-103.235	0.074
10	100.000	0187	399.000	5.000	5399.000	155.160	1658.160	106.614	1555.295	-102.865	0.066
11	110.000	0340	832.000	5.000	5832.000	155.880	1814.040	116.636	1710.825	-103.216	0.060
12	120.000	0109	265.000	6.000	6265.000	155.880	1969.920	126.659	1866.354	-103.566	0.055
13	130.000	0268	696.000	6.000	6696.000	155.160	2125.080	136.635	2021.884	-103.197	0.051
14	140.000	0081	129.000	7.000	7129.000	155.880	2280.960	146.658	2177.413	-103.547	0.048
15	150.000	0231	561.000	7.000	7561.000	155.520	2435.480	156.657	2332.943	-103.538	0.044
16	160.000	0361	993.000	7.000	7993.000	155.520	2592.000	166.656	2488.472	-103.528	0.042
17	170.000	01aa	426.000	8.000	8426.000	155.880	2747.880	176.679	2644.002	-103.879	0.039
18	180.000	0359	857.000	8.000	8857.000	155.160	2903.040	186.655	2799.531	-103.509	0.037
19	190.000	0121	289.000	9.000	9289.000	155.520	3058.560	196.655	2955.061	-103.500	0.035
20	200.000	02d2	722.000	9.000	9722.000	155.880	3214.440	206.677	3110.590	-103.850	0.033
21	210.000	0099	153.000	10.000	10153.000	155.160	3369.600	216.653	3266.120	-103.481	0.032
22	220.000	0249	585.000	10.000	10585.000	155.520	3525.120	226.653	3421.649	-103.471	0.030
23	230.000	0011	17.000	11.000	11017.000	155.520	3680.640	236.652	3577.179	-103.462	0.029
24	240.000	01c1	449.000	11.000	11449.000	155.520	3835.160	246.652	3732.708	-103.452	0.028
25	250.000	0371	881.000	11.000	11881.000	155.520	3991.680	256.651	3888.238	-103.443	0.027
26	260.000	0139	313.000	12.000	12313.000	155.520	4147.200	266.650	4043.767	-103.433	0.026
27	270.000	02e9	745.000	12.000	12745.000	155.520	4302.720	276.650	4199.297	-103.424	0.025
28	280.000	0081	177.000	13.000	13177.000	155.520	4458.340	286.649	4354.826	-103.414	0.024
29	290.000	0261	609.000	13.000	13609.000	155.520	4613.760	296.649	4510.356	-103.405	0.023
30	300.000	0029	41.000	14.000	14041.000	155.520	4769.280	306.648	4665.885	-103.395	0.022
31	310.000	01d9	473.000	14.000	14473.000	155.520	4924.800	316.647	4821.415	-103.386	0.021
32	320.000	0388	904.000	14.000	14904.000	155.520	5079.960	326.624	4976.944	-103.016	0.021
33	330.000	0151	337.000	15.000	15337.000	155.880	5235.840	336.646	5132.474	-103.367	0.020
34	340.000	0301	769.000	15.000	15769.000	155.520	5391.360	346.645	5288.003	-103.357	0.020
35	350.000	00c9	201.000	16.000	16201.000	155.520	5546.880	356.645	5443.533	-103.348	0.019
36	360.000	0279	633.000	16.000	16633.000	155.520	5702.400	366.644	5599.062	-103.338	0.018
37	370.000	0040	64.000	17.000	17064.000	155.520	5857.560	376.621	5754.592	-102.969	0.018
38	380.000	01f2	498.000	17.000	17498.000	156.240	6013.800	386.666	5910.121	-103.679	0.018
39	390.000	03a1	929.000	17.000	17929.000	155.160	6168.960	396.642	6065.651	-103.310	0.017
40	400.000	016a	362.000	18.000	18362.000	155.880	6324.840	406.665	6221.180	-103.660	0.017



**รูปที่ 4.66 กราฟความสัมพันธ์ระหว่างເອົາຫຼຸດມູນທາງທາງໝາຍງືແລະມູນທີ່ເກີດບັນຈິງຈາກນອເຕອຣ໌ເຖິງກັນເວລາໃນຮອບທີ່ 5**

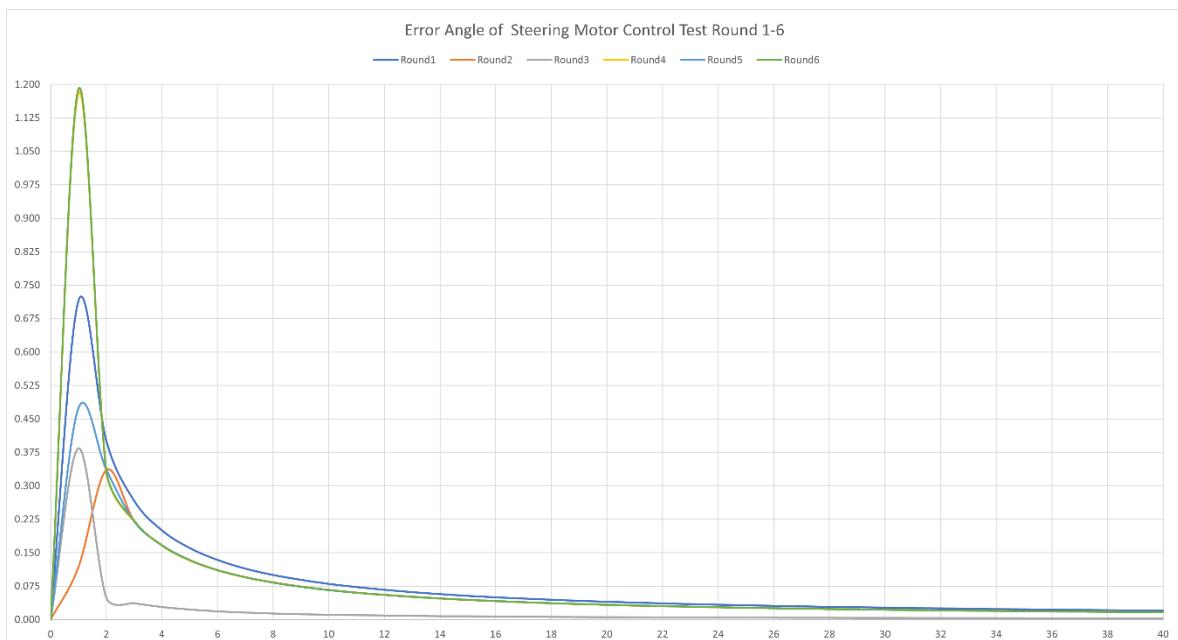
**ตารางที่ 4.10 การทดสอบการควบคุมນອເຕອຣ໌ໃນຮອບທີ່ 6**

Steering Motor Control Test Round 6 (0° - 400°)											
Time (s)	Angle input (deg)	Angle output (hex)	Angle output (dec)	Round No.	Electrical Angle output	Delta Electrical Angle output	Actual Angle output	Actual Angle output (deg)	Angle output theory	Angle difference (deg)	Error Angle
0	0.000	0319	793.000	0	793.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	02e3	739.000	1	1730.000	340.560	340.560	21.897	155.530	155.530	1.190
2	20.000	03b7	946.000	1	1946.000	74.520	415.080	26.688	311.059	155.530	0.334
3	30.000	0179	377.000	2	2377.000	155.160	570.240	36.664	466.589	-103.652	0.222
4	40.000	0329	809.000	2	2809.000	155.520	725.760	46.664	622.118	-103.642	0.167
5	50.000	00f1	241.000	3	3241.000	155.520	881.280	56.663	777.648	-103.633	0.133
6	60.000	02a1	673.000	3	3673.000	155.520	1036.800	66.663	933.177	-103.623	0.111
7	70.000	0069	105.000	4	4105.000	155.520	1192.320	76.662	1088.707	-103.614	0.095
8	80.000	0219	537.000	4	4537.000	155.520	1347.840	86.661	1244.236	-103.604	0.083
9	90.000	03c8	968.000	4	4968.000	155.160	1503.000	96.638	1399.766	-103.235	0.074
10	100.000	0192	402.000	5	5402.000	156.240	1659.240	106.683	1555.295	-103.545	0.067
11	110.000	0340	832.000	5	5832.000	154.800	1814.040	116.636	1710.825	-103.216	0.060
12	120.000	010a	266.000	6	6266.000	156.240	1970.280	126.682	1866.354	-103.926	0.056
13	130.000	02b9	697.000	6	6697.000	155.160	2125.440	136.658	2021.884	-103.557	0.051
14	140.000	0080	128.000	7	7128.000	155.160	2280.600	146.635	2177.413	-103.187	0.047
15	150.000	0230	560.000	7	7560.000	155.520	2436.120	156.634	2332.943	-103.178	0.044
16	160.000	03e3	995.000	7	7995.000	156.600	2592.720	166.703	2488.472	-104.248	0.042
17	170.000	01a8	424.000	8	8424.000	154.440	2747.160	176.633	2644.002	-103.159	0.039
18	180.000	0358	856.000	8	8856.000	155.520	2902.680	186.632	2799.531	-103.149	0.037
19	190.000	0121	289.000	9	9289.000	155.880	3058.560	196.655	2955.061	-103.500	0.035
20	200.000	02d1	721.000	9	9721.000	155.520	3214.080	206.654	3110.590	-103.490	0.033
21	210.000	0098	152.000	10	10152.000	155.160	3369.240	216.630	3266.120	-103.121	0.032
22	220.000	024a	586.000	10	10586.000	156.240	3525.480	226.676	3421.649	-103.831	0.030
23	230.000	0012	18.000	11	11018.000	155.520	3681.000	236.675	3577.179	-103.822	0.029
24	240.000	01c0	448.000	11	11448.000	154.800	3835.800	246.628	3732.708	-103.092	0.028
25	250.000	0371	881.000	11	11881.000	155.880	3991.680	256.651	3888.238	-103.443	0.027
26	260.000	0139	313.000	12	12313.000	155.520	4147.200	266.650	4043.767	-103.433	0.026
27	270.000	02e9	745.000	12	12745.000	155.520	4302.720	276.650	4199.297	-103.424	0.025
28	280.000	00b1	177.000	13	13177.000	155.520	4458.240	286.649	4354.826	-103.414	0.024
29	290.000	0261	609.000	13	13609.000	155.520	4613.760	296.649	4510.356	-103.405	0.023
30	300.000	002a	42.000	14	14042.000	155.880	4769.640	306.671	4665.885	-103.755	0.022
31	310.000	01d8	472.000	14	14472.000	154.800	4924.440	316.624	4821.415	-103.026	0.021
32	320.000	0388	904.000	14	14904.000	155.520	5079.960	326.624	4976.944	-103.016	0.021
33	330.000	0151	337.000	15	15337.000	155.880	5235.840	336.646	5132.474	-103.367	0.020
34	340.000	0301	769.000	15	15769.000	155.520	5391.360	346.645	5288.093	-103.357	0.020
35	350.000	00c9	201.000	16	16201.000	155.520	5546.880	356.645	5443.533	-103.348	0.019
36	360.000	027a	634.000	16	16634.000	155.880	5702.760	366.667	5599.062	-103.698	0.019
37	370.000	0042	66.000	17	17066.000	155.520	5858.280	376.667	5754.592	-103.689	0.018
38	380.000	01f1	497.000	17	17497.000	155.160	6013.440	386.643	5910.121	-103.319	0.017
39	390.000	03a2	930.000	17	17930.000	155.880	6169.320	396.666	6065.651	-103.670	0.017
40	400.000	016a	362.000	18	18362.000	155.520	6324.840	406.665	6221.180	-103.660	0.017



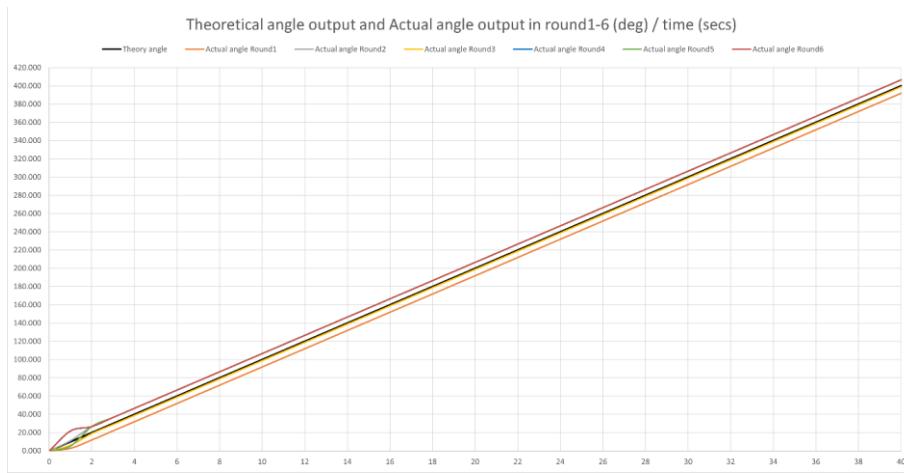
รูปที่ 4.67 กราฟความสัมพันธ์ระหว่างเวลาต่อร์พุตมุมทางทฤษฎีและมุมที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาในรอบที่ 6

สามารถเปรียบเทียบการทดสอบการควบคุมมอเตอร์ทั้ง 6 รอบได้ดังรูปที่ 4.68 และ 4.69



รูปที่ 4.68 กราฟแสดงค่าความคลาดเคลื่อนเทียบกับเวลาของ การทดสอบการควบคุมมอเตอร์ ในรอบที่ 1-6

จากรูปที่ 4.68 แสดงการเปรียบเทียบค่าความคลาดเคลื่อนจากการควบคุมมอเตอร์ในการรับ - ส่งค่าของมอเตอร์จำนวน 6 รอบ จะสังเกตเห็นว่าในรอบที่ 2 จะเกิดความคลาดเคลื่อนน้อยที่สุดและค่อยๆ ลดลง แต่จะพบว่าใช้เวลามากกว่าการทดลองในทุกรอบยกเว้นรอบที่ 1 และจะพบว่าในรอบที่ 3 นั้นจะมีค่าความผิดพลาดน้อยที่สุดรวมทั้งใช้เวลาเข้าสู่ค่า 0 น้อยที่สุด ซึ่งจากการทดลองรับส่งค่าในแต่ละรอบนั้นจะเห็นว่าไม่สามารถควบคุมการแก่วงของมอเตอร์ในช่วงเริ่มต้นสั่งงานได้ และจะเกิดค่าความคลาดเคลื่อนในช่วงเริ่มต้นสูงและค่อยๆ ลดลงจนค่าเข้าใกล้ 0



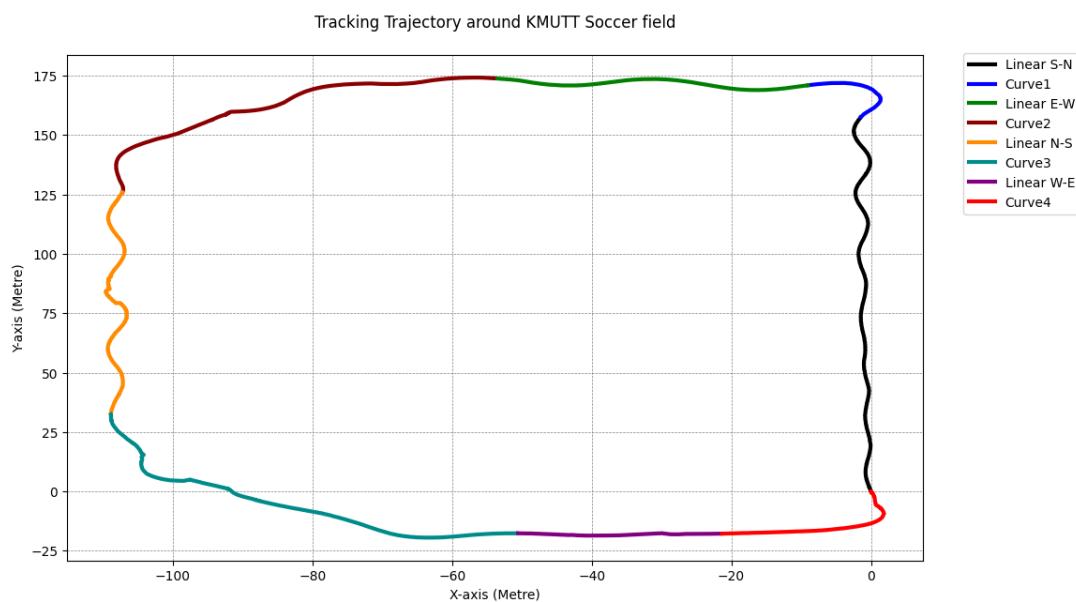
รูปที่ 4.69 กราฟความสัมพันธ์เปรียบเทียบระหว่างเอกสารพุฒน์ทางทฤษฎีและมุมที่เกิดขึ้นจริงจากมอเตอร์เทียบกับเวลาของรอบที่ 1-6

จากรูปที่ 4.69 เป็นการเปรียบเทียบค่า Theoretical angle output และ Actual angle output จากการส่งค่ามุม  $0^\circ - 400^\circ$  จำนวน 6 รอบเทียบกับเวลา ซึ่งจะพบว่าในรอบที่ 3 (เส้นสีเหลือง) มีค่าเข้าใกล้ค่าทางทฤษฎีมากที่สุด และในรอบอื่นๆ นั้นก็จะเกิดค่าความคลาดเคลื่อนมากขึ้น

#### 4.3.5 ผลการทดสอบการทำงานของมอเตอร์พวงมาลัยผ่าน Lateral Control Node รอบสนามฟุตบอล

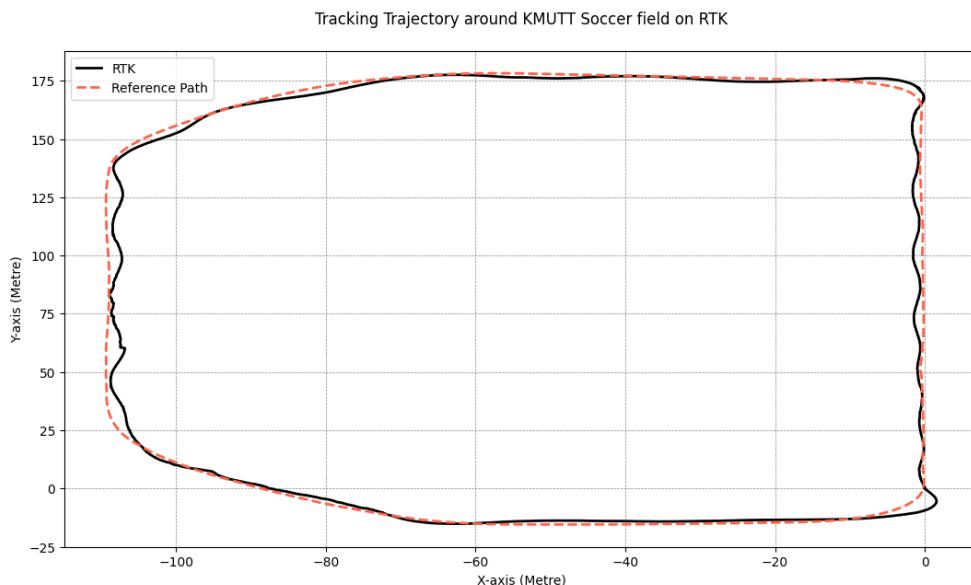
การทดลองการทำงานของมอเตอร์พวงมาลัยผ่าน Lateral Control Node ควบคุมโดยการเลือกเส้นทางในการติดตามเส้นทาง โดยจะมีการกำหนดค่าเกณฑ์ ( $K_p, K_i, K_d$ ) ในแต่ละเส้นทางแตกต่างกันเนื่องจากถนนรอบสนามฟุตบอลนั้น มีทิศทางเส้นตรงและทางโค้ง ซึ่งมีการเปลี่ยนแปลงทิศทางเกิดขึ้น จึงต้องมีการปรับจูนค่าเกณฑ์ในแต่ละเส้นทางให้มีความเหมาะสม เพื่อให้รถสามารถขับเคลื่อนอัตโนมัติผ่านถนนเส้นนั้น ได้อย่างต่อเนื่อง รวมทั้งการกำหนดค่ามุมบวกและมุมลบในการควบคุมให้มอเตอร์พวงมาลัยหมุน

โดยถ้ามุมเป็นบวกจะส่งผลให้มอเตอร์หมุนทวนเข็มนาฬิกาและหากมุมเป็นลบจะส่งผลให้มอเตอร์หมุนตามเข็มนาฬิกา ดังนั้นจึงมีการกำหนดเงื่อนไขในการแบ่งขอบเขตของถนนประกอบด้วยถนนฝั่งซ้ายให้แสดงค่าระยะทางตั้งจากที่รถที่ห่างจากเส้นทางอ้างอิงเป็นค่าติดลบ (-CTE) ส่วนถนนฝั่งขวาให้แสดงค่าระยะทางตั้งจากที่รถที่ห่างจากเส้นทางอ้างอิงเป็นค่าบวก (+CTE) และเส้นทางอ้างอิงตรงกลางหรือค่าระยะทางตั้งจากจากรถถึงเส้นทางอ้างอิงเป็นศูนย์ ( $CTE = 0$ ) เพื่อใช้สำหรับควบคุมให้รถเคลื่อนที่ตามตลอดเส้นทาง ซึ่งเส้นทางที่ใช้ควบคุมนั้นประกอบด้วยทางเส้นตรงทิศใต้ไปทิศเหนือ (Linear-SN), ทางเส้นตรงทิศตะวันออกไปทิศตะวันตก (Linear-EW), ทางเส้นตรงทิศตะวันตกไปทิศตะวันออก (Linear-WE), ทางเส้นตรงทิศเหนือไปทิศใต้ (Linear-NS), ทางโก้งระหว่างทางเส้นตรงทิศใต้ไปเหนือและทางเส้นตรงทิศตะวันออกไปทิศตะวันตก (Curve1), ทางโก้งระหว่างทางเส้นตรงทิศเหนือไปทิศใต้และทิศตะวันตกไปตะวันออก (Curve2), ทางโก้งระหว่างทางเส้นตรงทิศเหนือไปทิศใต้และทิศตะวันตกไปตะวันออก (Curve3) และ ทางโก้งระหว่างทางเส้นทิศตะวันตกไปตะวันออกกับทางเส้นตรงทิศใต้ไปเหนือ (Curve4) ดังรูปที่ 4.70 การติดตามวิถีเส้นทางรอบสนามฟุตบอลจช. โดยการติดตามวิถีการขับเคลื่อนรถกล้องอัตโนมัติโดยใช้เทคนิคการวัดแบบ Network RTK

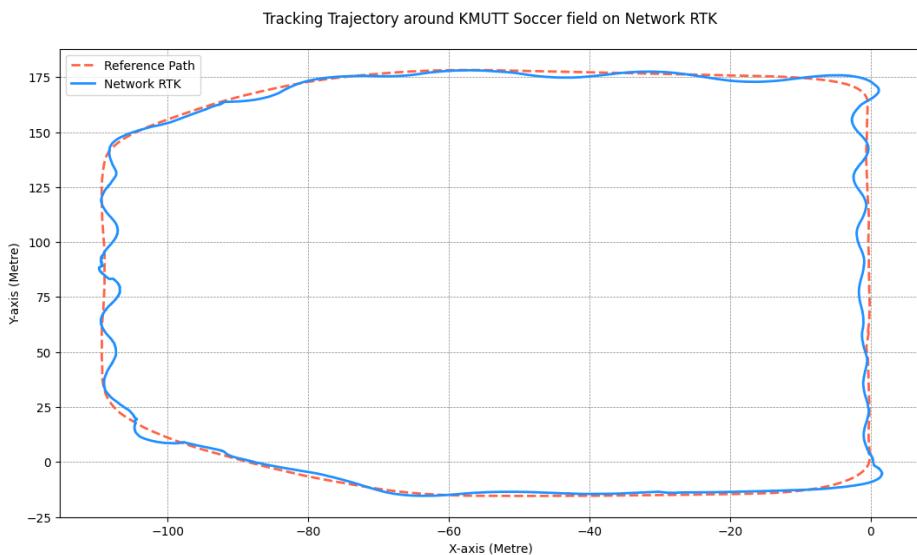


รูปที่ 4.70 การติดตามวิถีเส้นทางรอบสนามฟุตบอลจช. โดยการติดตามวิถีการขับเคลื่อนรถกล้องอัตโนมัติโดยใช้เทคนิคการวัดแบบ Network RTK

การติดตามวิถีเส้นทางนั้นได้ใช้เส้นทางอ้างอิง (Reference Path) ในการติดตามเส้นทาง โดยมีผู้ขับขี่yanpath เปรียบเทียบกับการขับเคลื่อนอัตโนมัติรอบสนามฟุตซอลมจช. ผ่านการใช้เทคนิคการวัดแบบ RTK แสดงดังรูปที่ 4.71 และผ่านการใช้เทคนิคการวัดแบบ Network RTK แสดงดังรูปที่ 4.72

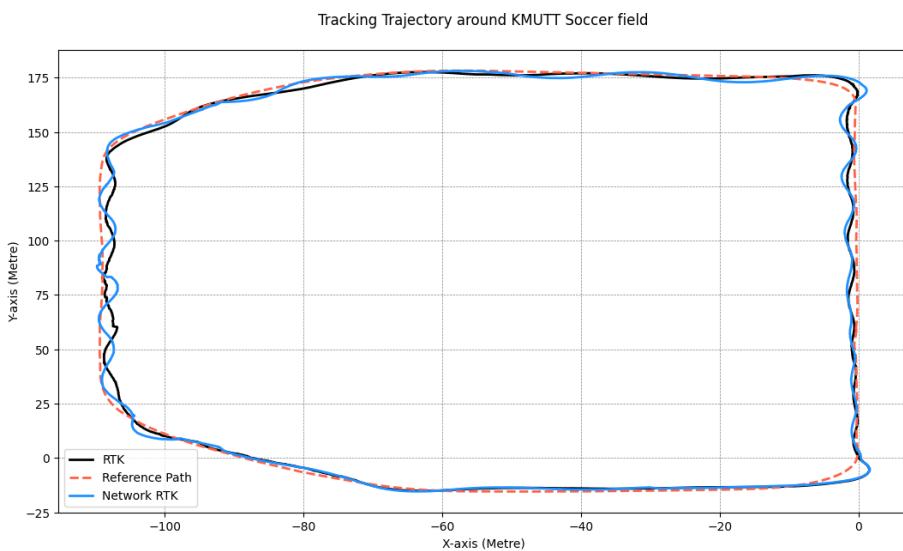


รูปที่ 4.71 การติดตามวิถีเส้นทางรอบสนามฟุตซอลมจช. ซึ่งใช้เส้นทางอ้างอิงโดยมีผู้ขับขี่เปรียบเทียบ กับการขับเคลื่อนอัตโนมัติของyanpath โดยใช้เทคนิคการวัดแบบ RTK



รูปที่ 4.72 การติดตามวิถีเส้นทางรอบสนามฟุตซอลมจช. ซึ่งใช้เส้นทางอ้างอิงโดยมีผู้ขับขี่เปรียบเทียบ กับการขับเคลื่อนอัตโนมัติของyanpath โดยใช้เทคนิคการวัดแบบ Network RTK

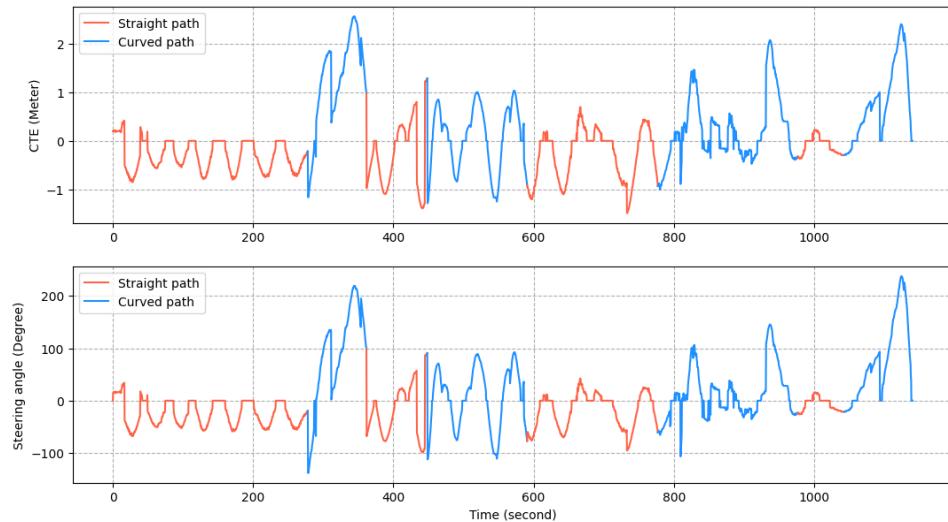
ในรูปที่ 4.71 จะแสดงผลของพิกัด XY จากการควบคุมรถด้วย Lateral Tracking Control รอบสนามฟุตซอล โดยใช้เทคนิคการวัดแบบ RTK และ ในรูปที่ 4.72 จะแสดงผลพิกัดของการใช้เทคนิคการวัดแบบ Network RTK เนื่องจากใช้เพียง GNSS Sensor ในการควบคุมรถ จึงมีโอกาสสูญเสียน้ำเสียได้ง่าย หากสภาพอากาศหรือสภาพแวดล้อมโดยรอบ แต่ก็มีความละเอียดเพียงพอต่อการควบคุมรถให้สามารถเคลื่อนที่รอบเส้นทางอ้างอิงได้ และจะเห็นได้ว่าสำหรับ Straight path N-S การควบคุมรถให้วิ่งตรงค่อนข้างยาก เนื่องจากบริเวณนั้น มีอาคารขนาดใหญ่ตั้งอยู่รอบเส้นทาง ทำให้พิกัดของรถเกิดความคลาดเคลื่อนเพิ่มขึ้นเล็กน้อย



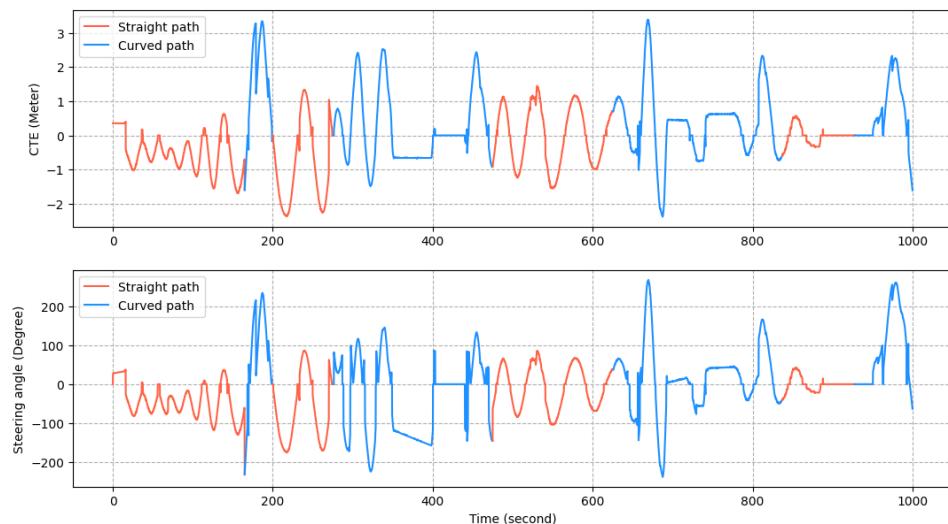
รูปที่ 4.73 การติดตามวิถีเส้นทางรอบสนามฟุตซอลมหาช. เปรียบเทียบเส้นทางอ้างอิงกับการขับเคลื่อนอัตโนมัติของyanพาหนะโดยการใช้เทคนิคการวัดแบบ RTK และ Network RTK

ในรูปที่ 4.73 จะแสดงการเปรียบเทียบพิกัด XY ด้วยการใช้เทคนิคการวัดแบบ RTK และ Network RTK จะเห็นได้ว่าพิกัดที่ได้มีความแตกต่างกันเล็กน้อย แต่เนื่องจากใช้ความเร็วรถน้อยอยู่ที่ค่าไม่เกิน 5 กิโลเมตร/ชั่วโมง และความละเอียดของทั้ง 2 เทคนิคนั้นมีค่าความละเอียดอยู่ในระดับเซนติเมตรทั้งคู่ ถึงแม้ในเทคนิค RTK จะมีความละเอียดมากกว่าเทคนิค Network RTK ก็ตาม แต่เมื่อนำมาใช้ในการนำทางรถผลต่างระหว่างความละเอียดนี้จะไม่ส่งผลกระทบมากนัก ทำให้เมื่อใช้งานจริงจึงไม่รู้สึกถึงความแตกต่าง

จากการขับเคลื่อนยานพาหนะรถกอล์ฟอัตโนมัติรอบสนามฟุตซอลมาตรฐาน. ได้ทำการบันทึกค่าความคลาดเคลื่อนที่เกิดขึ้นจากการติดตามวิถีการขับเคลื่อนรถอัตโนมัติ โดยการใช้เทคนิคการวัดแบบ RTK ดังรูปที่ 4.74 และเทคนิคการวัดแบบ Network RTK ดังรูปที่ 4.75



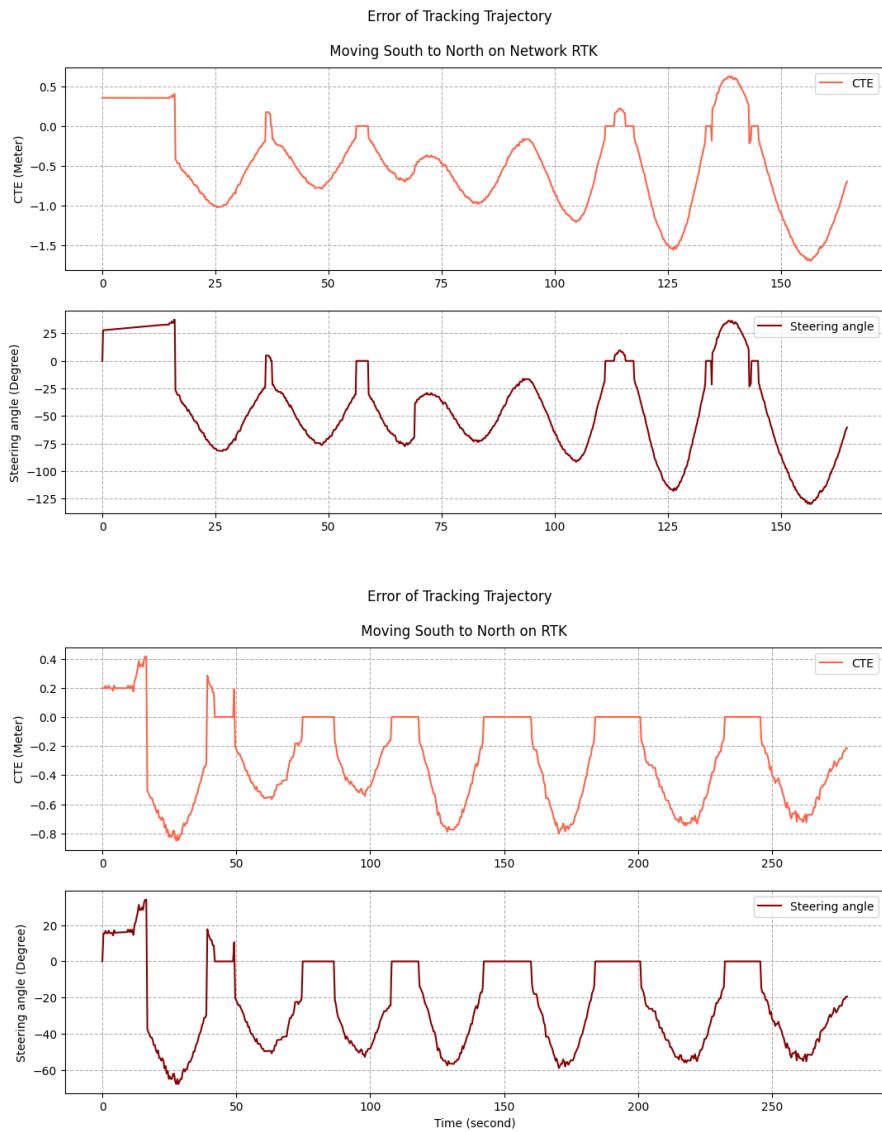
รูปที่ 4.74 ค่าความคลาดเคลื่อนจากการติดตามวิถีการขับเคลื่อนอัตโนมัติรอบสนามฟุตซอลมาตรฐาน. จากการใช้เทคนิคการวัดแบบ RTK



รูปที่ 4.75 ค่าความคลาดเคลื่อนจากการติดตามวิถีการขับเคลื่อนอัตโนมัติรอบสนามฟุตซอลมาตรฐาน. จากการใช้เทคนิคการวัดแบบ Network RTK

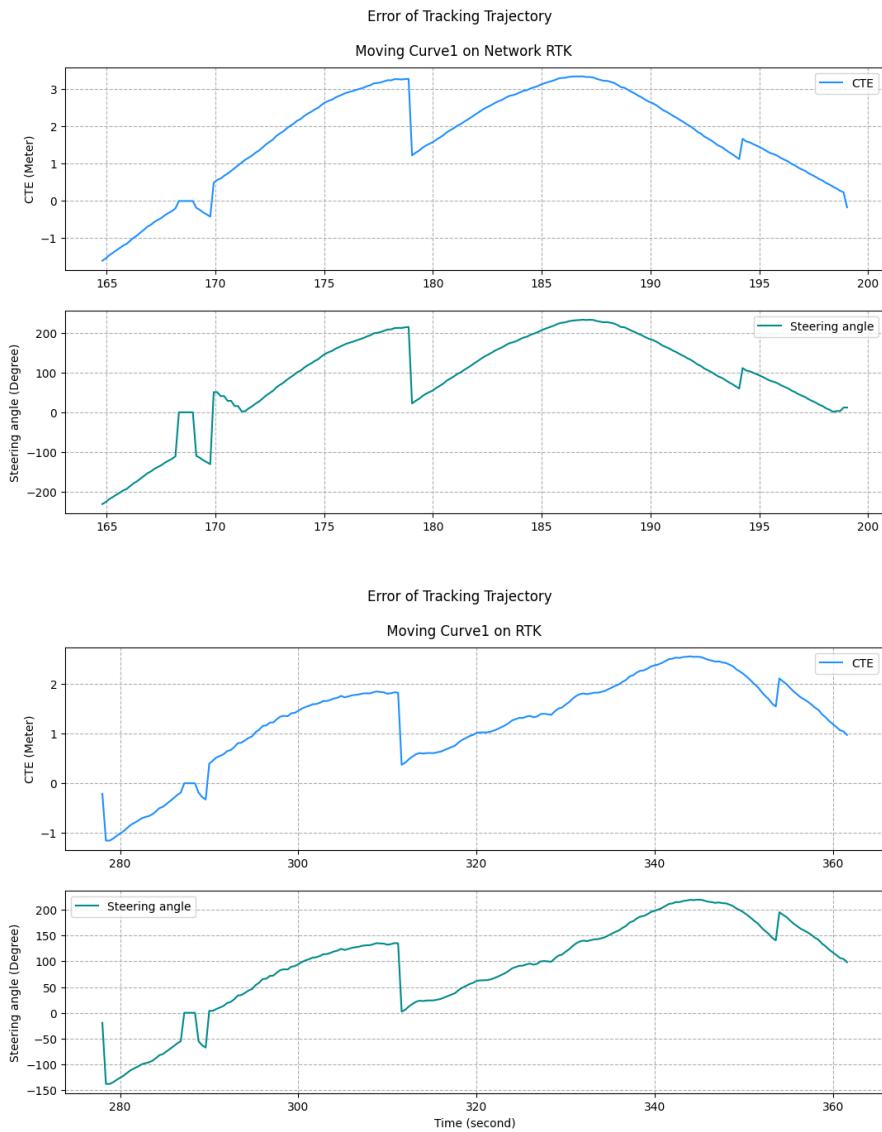
จากรูปที่ 4.74 และ 4.75 เมื่อเปรียบเทียบค่า CTE (Error Cross Track) จะพบว่าการใช้เทคนิคการวัดแบบ RTK นั้นส่งผลให้ค่า CTE หรือค่าความคลาดเคลื่อนมีช่วงค่าอยู่ระหว่าง -1 ถึง 3 เมตร หรือหมายถึงเกิดค่าความคลาดเคลื่อนมากสุดไม่เกิน 3 เมตร ซึ่งเกิดช่วงค่าที่น้อยกว่าการใช้เทคนิคการวัดแบบ Network RTK ซึ่งเกิดค่าความคลาดเคลื่อนอยู่ในช่วง -2 ถึง 3.5 เมตร หรือหมายถึงเกิดค่าความคลาดเคลื่อนมากสุดไม่เกิน 3.5 เมตร และเมื่อเปรียบเทียบมุมหมุนของพวงมาลัย (Steering angle) ก็จะสามารถเห็นถึงความแตกต่างได้อ่ายหักเจนคือจากการใช้เทคนิคการวัดแบบ RTK ค่ามุมด้านลบจะอยู่ในช่วงไม่เกิน -150 องศา ส่วนมุมด้านบวกจะมีค่าในช่วงไม่เกิน 250 องศา และจากการใช้เทคนิคการวัดแบบ Network RTK ค่ามุมด้านลบจะอยู่ในช่วงไม่เกิน -250 องศา ส่วนมุมด้านบวกจะมีค่าในช่วงไม่เกิน 300 องศา ซึ่งส่งผลต่อ มุมหมุนที่ไปสั่งมอเตอร์พวงมาลัย เนื่องจากมอเตอร์นั้นมีค่ามุมจำกัดในการหมุน คือถ้าหากเกินค่ามุมประมาณ -300 หรือ 300 องศา จะทำให้มอเตอร์หยุดทำงานและส่งผลให้มอเตอร์พวงมาลัยตัดวงจรไม่ จ่ายไฟ หรือมอเตอร์ถูกปิดใช้งานเนื่องจากต้องดึงกระแสไฟมาใช้มากขึ้นเมื่อกระทำการกับโหลดมากขึ้น โดยที่โหลดนั้นคือแรงเสียดทานที่เกิดขึ้นระหว่างที่ล้อรถกระทำกับพื้นผิวนน ส่งผลให้กระแสไฟอาจไม่ เพียงพอ จึงต้องทำการสั่งเปิดใช้งานมอเตอร์ใหม่หลังจากที่มอเตอร์นั้นถูกตัดวงจรไป ดังนั้นจากราฟแสดงค่าความคลาดเคลื่อนจากการติดตามวิถีการขับเคลื่อนอัตโนมัติจากการใช้เทคนิคการวัดแบบ RTK นั้นส่งผลต่อการควบคุมรถให้มั่นคงหรือมีความเสถียรมากกว่า รวมทั้งส่งผลทำให้เกิด Overshoot หรือ เกิดการแกว่งที่น้อยกว่าเมื่อเทียบกับการใช้เทคนิคการวัดแบบ Network RTK ใน การขับเคลื่อนอัตโนมัติ

ในส่วนถัดไปรูปที่ 4.76 – 4.84 จะแสดงการใช้เทคนิคการวัดแบบ Network RTK และ RTK และแสดงค่าความคลาดเคลื่อนจากการติดตามวิถีการขับเคลื่อนรถกล้องอัตโนมัติรอบสนามฟุตบอลในแต่ละเส้นทาง ประกอบด้วยทางเส้นตรงทิศใต้ไปทิศเหนือ (Linear-SN), ทางเส้นตรงทิศตะวันออกไปทิศตะวันตก (Linear-EW), ทางเส้นตรงทิศตะวันตกไปทิศตะวันออก (Linear-WE), ทางเส้นตรงทิศเหนือไปทิศใต้ (Linear-NS), ทางโค้งระหว่างทางเส้นตรงทิศใต้ไปเหนือและทางเส้นตรงทิศตะวันออกไปทิศตะวันตก (Curve1), ทางโค้งระหว่างทางเส้นตรงทิศตะวันออกไปทิศตะวันตกและทางเส้นตรงทิศเหนือไปทิศใต้ (Curve2), ทางโค้งระหว่างทางเส้นตรงทิศเหนือไปทิศใต้และทิศตะวันตกไปตะวันออก (Curve3) และ ทางโค้งระหว่างทางเส้นทิศตะวันตกไปตะวันออกกับทางเส้นตรงทิศใต้ไปเหนือ (Curve4)



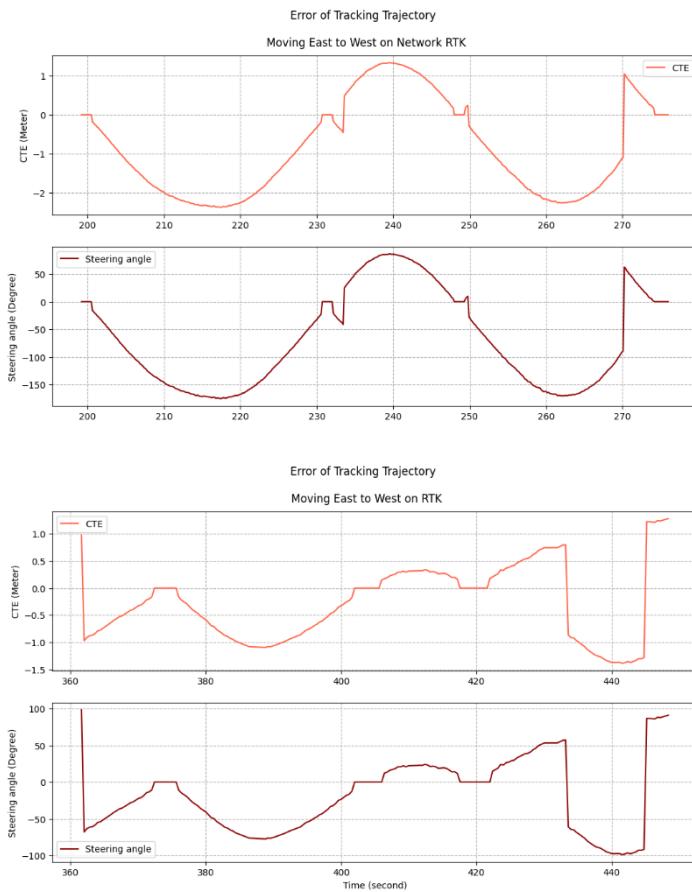
**รูปที่ 4.76** การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถ กอล์ฟอัตโนมัติรอบสนามฟุตซอลบริเวณถนนเส้นตรงจากทิศใต้ไปทิศเหนือ (Linear S-N)

จากรูปที่ 4.76 ในเส้นทางเส้นตรงจากทิศใต้ถึงทิศเหนือ (Linear S-N) ซึ่งเป็นเส้นทางเริ่มต้นการเคลื่อนที่ของรถ จะสามารถสังเกตเห็นได้ชัดเจนว่าการใช้เทคนิคการวัดแบบ RTK นั้นจะส่งผลต่อการขับเคลื่อนรถ กอล์ฟอัตโนมัติ โดยค่ามุ่งและค่าระยะทางตั้งฉากจากจราจรจะลิ่วเส้นทางข้างอยู่นั้นมีค่าเป็น 0 และเข้าใกล้ 0 มากกว่าการใช้เทคนิคการวัดแบบ Network RTK เนื่องมาจาก การรับสัญญาณพิกัด XY ที่มีค่าความแม่นยำที่สูงขึ้น จึงทำให้ค่า CTE และค่า Steering angle นั้นมีค่าที่แม่นยำเพิ่มขึ้น เช่นกัน รวมทั้งการใช้เทคนิคการวัดแบบ Network RTK หรือ RTK นั้นใช้ค่าเกณฑ์ในการควบคุม PID Controller เพื่อใช้ในการคำนวณมุมหมุนของมอเตอร์พวงมาลัยมีค่าเท่ากัน



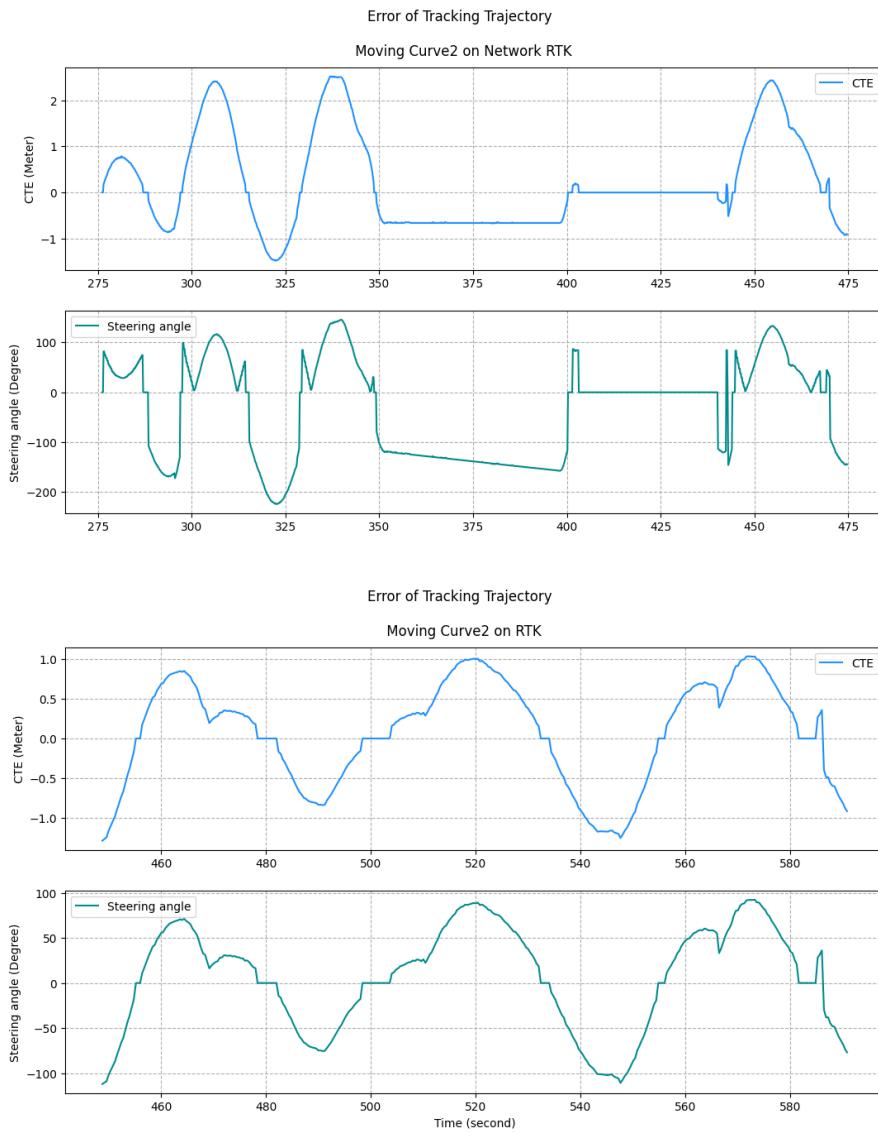
**รูปที่ 4.77** การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถ กอล์ฟอัตโนมัติรอบสนามฟุตซอลบริเวณถนนทางโค้งที่ 1 (Curve1)

จากรูปที่ 4.77 ในเส้นทางเส้นโค้งที่ 1 (Curve1) ในช่วงนี้ผลของการขับเคลื่อนรถกอล์ฟอัตโนมัตินี้มีค่า มุมและระยะทางตั้งจากจารถถึงเส้นทางอ้างอิงไม่ต่างกันมากนัก เนื่องจากทางเส้นโค้งนี้จะมีการ วางแผนเส้นทางที่แตกต่างจากทางเส้นตรงคือใช้เส้นความชันเป็นตัวกำหนดของเขตของถนนฝั่งชายและ ขวาก่อนการใช้พิกัด XY รวมทั้งต้องมีการเก็บเส้นทางเป็นเส้นตรงต่อ กันประมาณ 3 เส้น เนื่องจากพิกัด XY นี้มีการเปลี่ยนแปลงทั้ง 2 แกน จึงต้องใช้เส้นตรงต่อ กันเพื่อเป็นเส้นทางอ้างอิง จึงเป็นผลทำให้ค่า ความคลาดเคลื่อนจากการขับเคลื่อนอัตโนมัติในช่วงโค้งนี้ค่าไม่ต่างกันมากในระยะเวลาใดๆ ก็ตาม



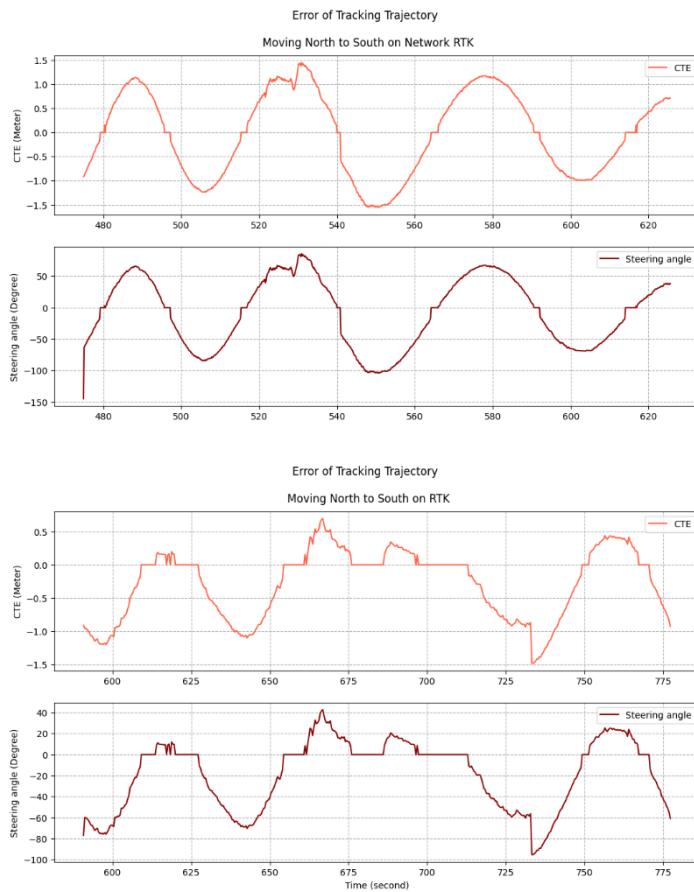
**รูปที่ 4.78** การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถ กอัลฟอัต โนมัติ robust สำหรับสถานที่ชิดซ้ายบนถนนเส้นตรงจากทิศตะวันออกไปทิศตะวันตก (Linear E-W)

จากรูปที่ 4.78 ในเส้นทางเส้นตรงจากทิศตะวันออกไปทิศตะวันตก (Linear E-W) จะสามารถสังเกตุเห็น ว่าการใช้เทคนิคการวัดแบบ RTK ใน การขับเคลื่อนรถกอัลฟอัต โนมัติไม่ต่างจากการใช้เทคนิคการวัดแบบ Network RTK มากนัก โดยค่ามุมและค่าระยะทางตั้งจากจารรถถึงเส้นทางอ้างอิงนั้นมีค่าอยู่ในช่วงที่เทียบเท่ากันคือแบบ Network RTK นั้นเกิดค่า CTE ด้านลบ (ฝั่งซ้ายของถนน) ในช่วงไม่เกิน -3 เมตร และเกิดค่า CTE ด้านบวก (ฝั่งขวาของถนน) ในช่วงไม่เกิน 1.5 เมตร สำหรับมุมในการควบคุมมอเตอร์พวงมาลัยนั้นจะอยู่ในช่วงไม่เกิน -200 ถึง 100 องศา และแบบ RTK นั้นเกิดค่า CTE ด้านลบ (ฝั่งซ้ายของถนน) ในช่วงไม่เกิน -1.5 เมตร และเกิดค่า CTE ด้านบวก (ฝั่งขวาของถนน) ในช่วงไม่เกิน 1.5 เมตร สำหรับมุมในการควบคุมมอเตอร์พวงมาลัยนั้นจะอยู่ในช่วงไม่เกิน -100 ถึง 100 องศา ดังนั้นจะพบว่า เทคนิคการวัดพิกัดทั้ง 2 แบบนี้ไม่ค่อยส่งผลกระแทบท่อการขับเคลื่อนรถกอัต โนมัติมากนักในเส้นทาง Linear E-W



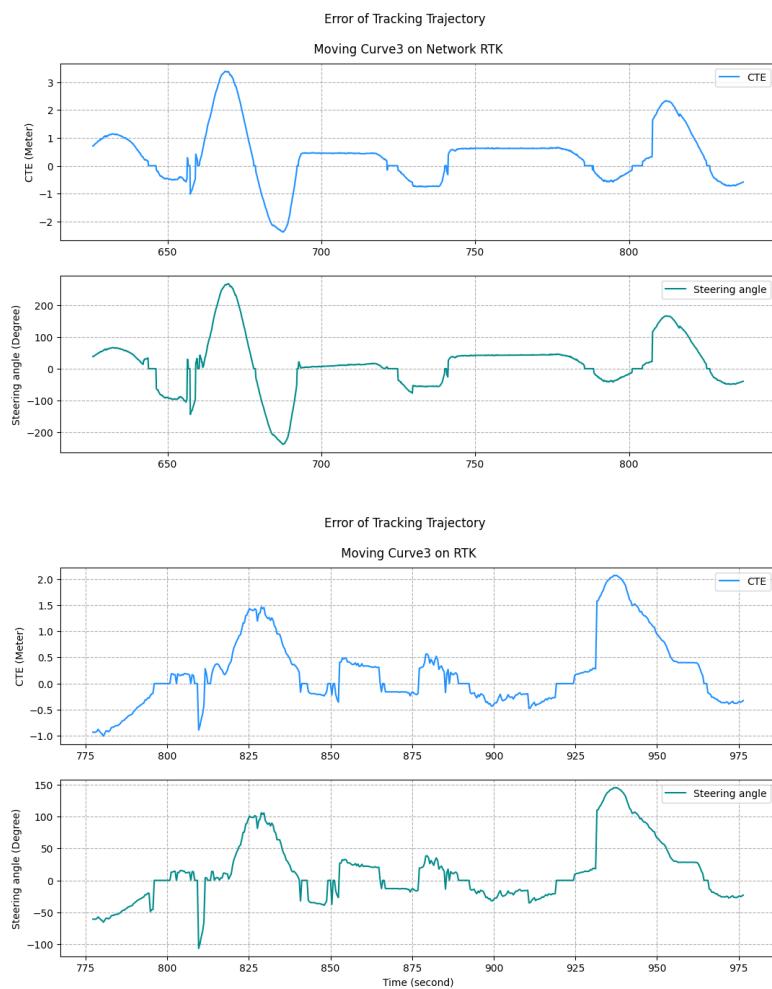
รูปที่ 4.79 การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถ กอัลฟอต์โนมัติ robust สำหรับฟุตซอลบริเวณถนนทางโค้งที่ 2 (Curve2)

จากรูปที่ 4.79 ในเส้นทางเส้นโค้งที่ 2 (Curve2) ในช่วงนี้ผลของการขับเคลื่อนรถกอัลฟอต์โนมัตินี้มีค่า มุมและระยะทางตั้งจากจารถถึงเส้นทางอ้างอิงไม่ต่างกันมากนัก เนื่องจากทางเส้นโค้งนี้จะมีการ วางแผนเส้นทางที่แตกต่างจากทางเส้นตรงดังที่อธิบายในรูปที่ 4.77 โดยมีการเก็บเส้นทางเป็นเส้นตรงต่อ กันประมาณ 6 เส้น จึงเป็นผลทำให้ค่าความคลาดเคลื่อนจากการขับเคลื่อนอัตโนมัติในช่วงโค้งมีค่าไม่ ต่างกันมากในระยะเวลาใดๆ ก็ตาม



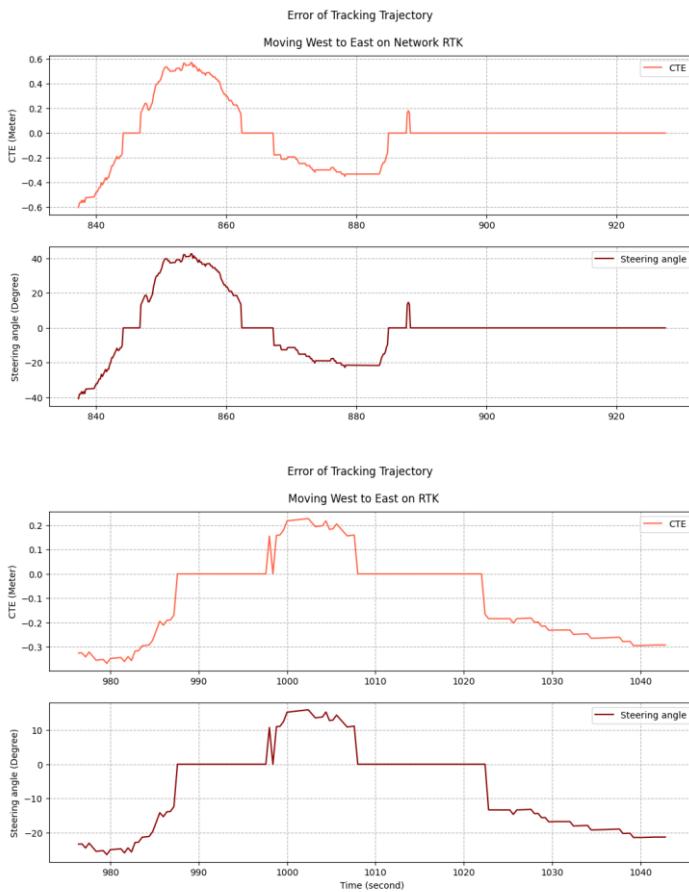
**รูปที่ 4.80** การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถ กอัลฟอตโนมัติรอบสนามฟุตซอลบริเวณถนนเส้นตรงจากทิศเหนือไปทิศใต้ (Linear N-S)

จากรูปที่ 4.80 ในเส้นทางเส้นตรงจากทิศเหนือไปทิศใต้ (Linear N-S) จะสามารถสังเกตเห็นว่าการใช้เทคนิคการวัดแบบ RTK ในการขับเคลื่อนรถกอัลฟอตโนมัติไม่ต่างจากการใช้เทคนิคการวัดแบบ Network RTK มากนัก โดยค่ามุมและค่าระยะทางตั้งจากจารรถถึงเส้นทางอ้างอิงนั้นมีค่าอยู่ในช่วงที่เทียบเท่ากันคือแบบ Network RTK นั้นเกิดค่า CTE ด้านลบ (ฝั่งซ้ายของถนน) ในช่วงไม่เกิน -1.5 เมตร และเกิดค่า CTE ด้านบวก (ฝั่งขวาของถนน) ในช่วงไม่เกิน 1.5 เมตร สำหรับมุมในการควบคุมมอเตอร์พวงมาลัยนั้นจะอยู่ในช่วงไม่เกิน -150 ถึง 100 องศา และแบบ RTK นั้นเกิดค่า CTE ด้านลบ (ฝั่งซ้ายของถนน) ในช่วงไม่เกิน -1.5 เมตร และเกิดค่า CTE ด้านบวก (ฝั่งขวาของถนน) ในช่วงไม่เกิน 0.75 เมตร สำหรับมุมในการควบคุมมอเตอร์พวงมาลัยนั้นจะอยู่ในช่วงไม่เกิน -100 ถึง 50 องศา และเข้าใกล้ค่าศูนย์เป็นช่วงๆ ดังนั้นจะพบว่าเทคนิคการวัดพิกัดทั้ง 2 แบบนั้นไม่ค่อยส่งผลกระทบต่อการขับเคลื่อนรถ อัตโนมัติมากนักในเส้นทาง Linear N-S



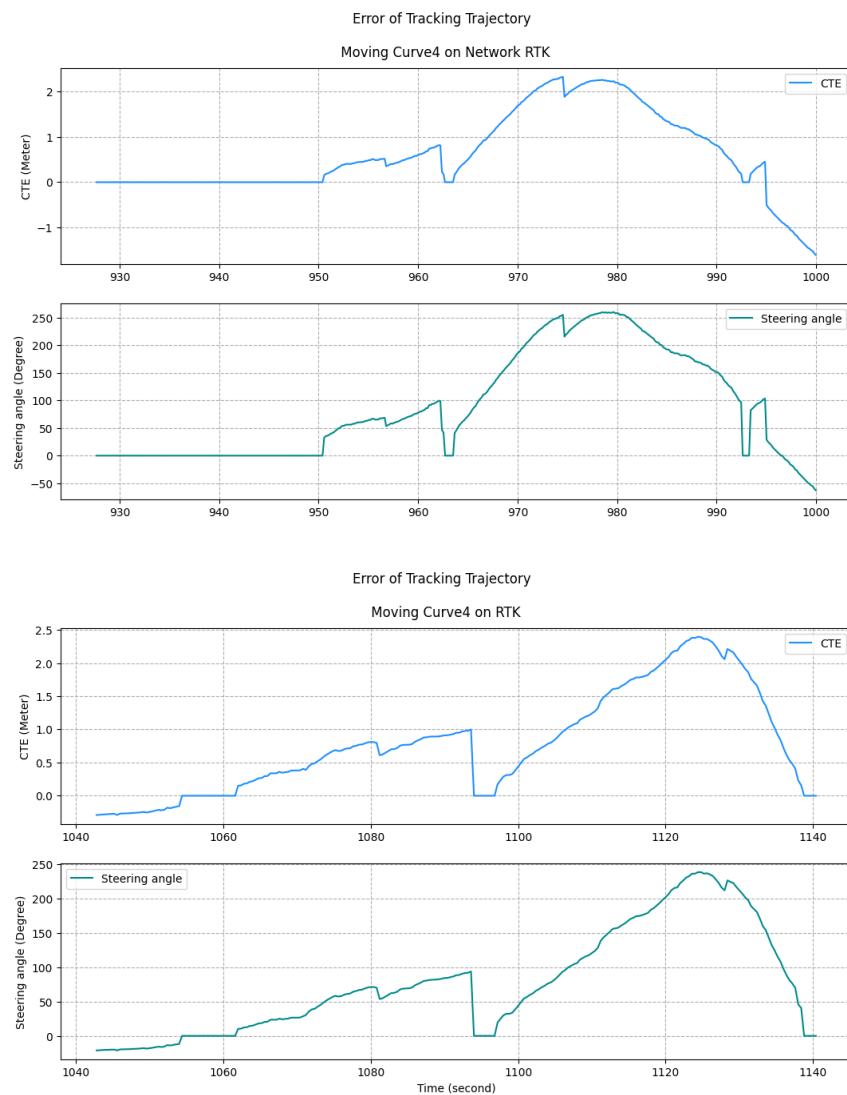
รูปที่ 4.81 การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถ กอัลฟอตโนมติ robust สำหรับเส้นทางฟุตซอลบริเวณถนนทางโถงที่ 3 (Curve3)

จากรูปที่ 4.82 ในเส้นทางเส้นโถงที่ 3 (Curve3) ในช่วงนี้ผลของการขับเคลื่อนรถกอัลฟอตโนมตินี้ใช้ระยะเวลาเท่าๆ กัน ค่ามุ่งและระยะทางตั้งแต่จากจารถึงเส้นทางอ้างอิงมีความแตกต่างกันเนื่องจากสถานีรับ-ส่ง สัญญาณตั้งอยู่บน บริเวณหน้าอาคารพระจอมเกล้าราชนครินทร์ 190 ปี เนื่องจากทางเส้นโถงนี้จะมีอาคารขนาดใหญ่ติดกันอย่างต่อเนื่อง ส่งผลทำให้สัญญาณหายเป็นช่วงๆ และค่าการรับสัญญาณนั้นอาจมีความคลาดเคลื่อนเกิดขึ้นค่อนข้างมากจากสภาพแวดล้อม และเมื่อตั้งสถานีรับ-ส่งสัญญาณจึงทำให้สัญญาณบริเวณนั้นสามารถรับค่าพิกัดได้แม่นยำมากขึ้นกว่าการใช้เทคนิคการวัดแบบ Network RTK โดยช่วงโถงนี้จะเก็บเส้นทางเป็นเส้นตรงต่อ กันประมาณ 8 เส้น และพบว่าค่า CTE และค่ามุ่งควบคุมพวงมาลัยนั้นมีค่าแตกต่างกันซึ่งไม่สูงมากนัก โดยการใช้เทคนิคการวัดแบบ RTK จะมีความละเอียดมากกว่าและส่งผลให้การขับเคลื่อนรถกอัลฟอตโนมติมีความแม่นยำมากขึ้น



รูปที่ 4.83 การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถ กอัลฟอัต โนมัติ robust สำหรับเส้นทางฟุตซอลบริเวณถนนเส้นตรงจากทิศตะวันตกไปทิศตะวันออก (Linear W-E)

จากรูป 4.83 ในเส้นทางเส้นตรงจากทิศตะวันตกไปทิศตะวันออก (Linear W-E) จะสามารถสังเกตุเห็นว่า การใช้เทคนิคการวัดแบบ RTK ใน การขับเคลื่อนรถ กอัลฟอัต โนมัติไม่ต่างจากการใช้เทคนิคการวัดแบบ Network RTK มากนัก โดยค่ามุมและค่าระยะทางตั้งจากจารรถถึงเส้นทางอ้างอิงนั้นมีค่าอยู่ในช่วงที่เทียบเท่ากันคือแบบ Network RTK นั้นเกิดค่า CTE ด้านลบ (ผ่านซ้ายของถนน) ในช่วงไม่เกิน -0.6 เมตร และเกิดค่า CTE ด้านบวก (ผ่านขวาของถนน) ในช่วงไม่เกิน 0.6 เมตร สำหรับมุมในการควบคุมมอเตอร์ พวงมาลัยนั้นจะอยู่ในช่วงไม่เกิน -40 ถึง 40 องศา และแบบ RTK นั้นเกิดค่า CTE ด้านลบ (ผ่านซ้ายของถนน) ในช่วงไม่เกิน -0.4 เมตร และเกิดค่า CTE ด้านบวก (ผ่านขวาของถนน) ในช่วงไม่เกิน 0.25 เมตร สำหรับมุมในการควบคุมมอเตอร์พวงมาลัยนั้นจะอยู่ในช่วงไม่เกิน -30 ถึง 20 องศา และเข้าใกล้ค่าสูงยิ่งเป็นช่วงๆ ดังนั้นจะพบว่าเทคนิคการวัดพิกัดทั้ง 2 แบบนั้นไม่ค่อยส่งผลกระทบต่อการขับเคลื่อนรถ อัตโนมัติมากนักในเส้นทาง Linear W-E

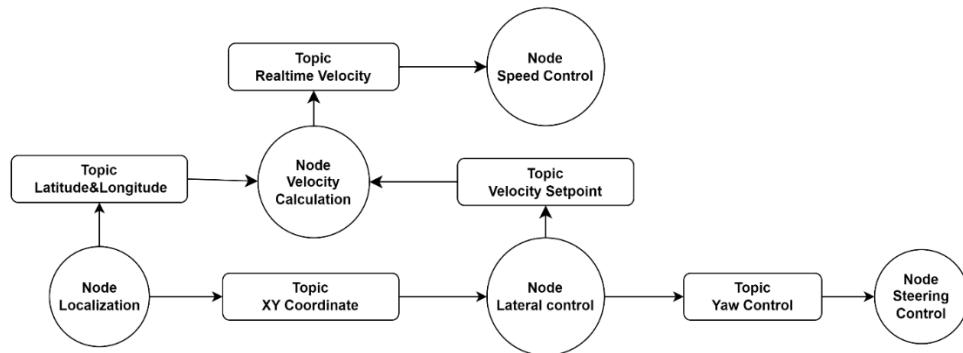


รูปที่ 4.84 การใช้เทคนิคการวัดแบบ Network RTK และ RTK แสดงค่าความคลาดเคลื่อนการขับเคลื่อนรถ กอล์ฟอัตโนมัติรอบสนามฟุตซอลบริเวณถนนทางโถงที่ 4 (Curve4)

จากรูปที่ 4.84 ในเส้นทางเส้นโถงที่ 4 (Curve4) ในช่วงนี้เป็นโถงสุดท้ายที่จะวนกลับมาถึงจุดเริ่มต้น จะพบว่าค่ามุมและระยะทางตั้งจากจารถึงเส้นทางอ้างอิงมีค่าสูงมากในระยะเวลาช่วงหลัง เพราะต้องพยายามเลี้ยวกลับมาอย่างจุดกึ่งกลางของถนนหรือเส้นทางอ้างอิงที่  $CTE = 0$  ซึ่งแสดงดังกราฟจากการใช้เทคนิคการวัดแบบ RTK และพบว่าค่ามุมและ CTE จากการใช้เทคนิคการวัดทั้ง 2 แบบมีค่าไม่ต่างกันมาก เนื่องจากทางเส้นโถงนี้จะมีการวางแผนเส้นทางที่แตกต่างจากทางเส้นตรงดังที่อธิบายในรูปที่ 4.70 โดยมีการเก็บเส้นทางเป็นเส้นตรงต่อ กันประมาณ 6 เส้น จึงเป็นผลทำให้ค่าความคลาดเคลื่อนจากการขับเคลื่อนอัตโนมัติในช่วงโถงมีค่าไม่ต่างกันมากในระยะเวลาใดๆ ก็ตาม

## 4.4 การทดลองและผลลัพธ์ของระบบ Robot Operating System (ROS)

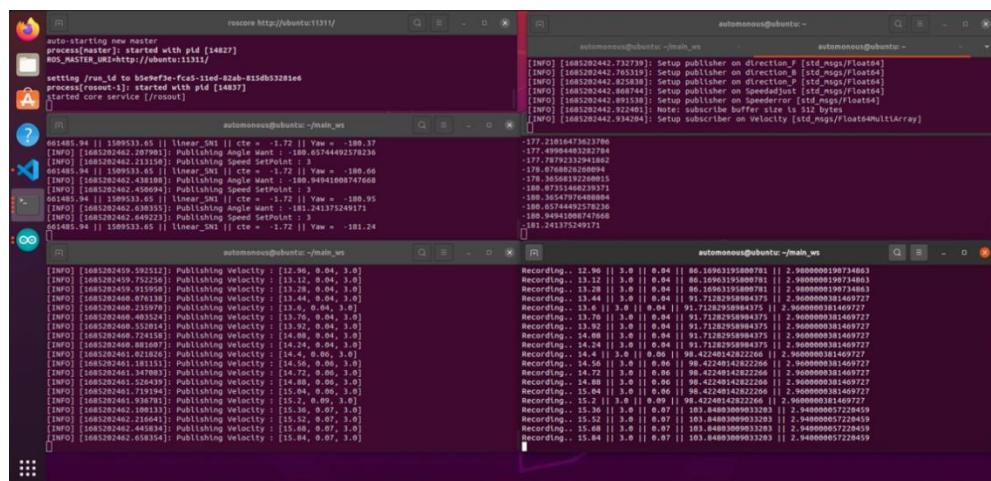
การทดลองนี้มีจุดประสงค์เพื่อทดสอบความถูกต้องในการสื่อสารข้อมูลและความคุณภาพของระบบต่างๆ ผ่าน Robot Operating System ตามที่ได้ออกแบบไว้



รูปที่ 4.85 ผังงานการทำงานของระบบ ROS

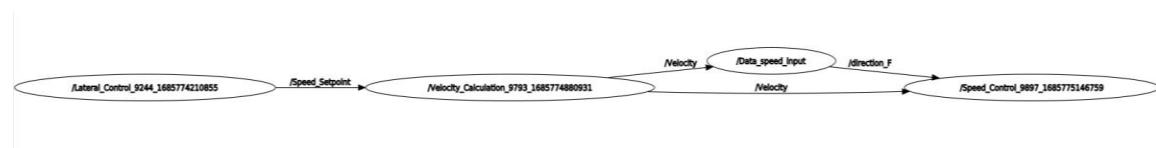


รูปที่ 4.86 ภาพแสดงผล rqt\_graph ภายใน Robot Operating System

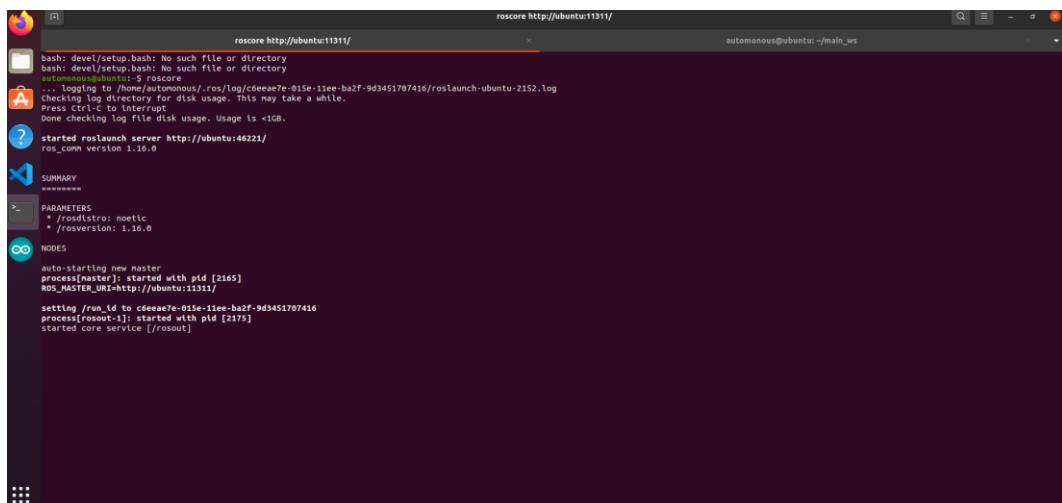


รูปที่ 4.87 ภาพแสดงการส่ง Message ของระบบทั้งหมดผ่าน Terminal

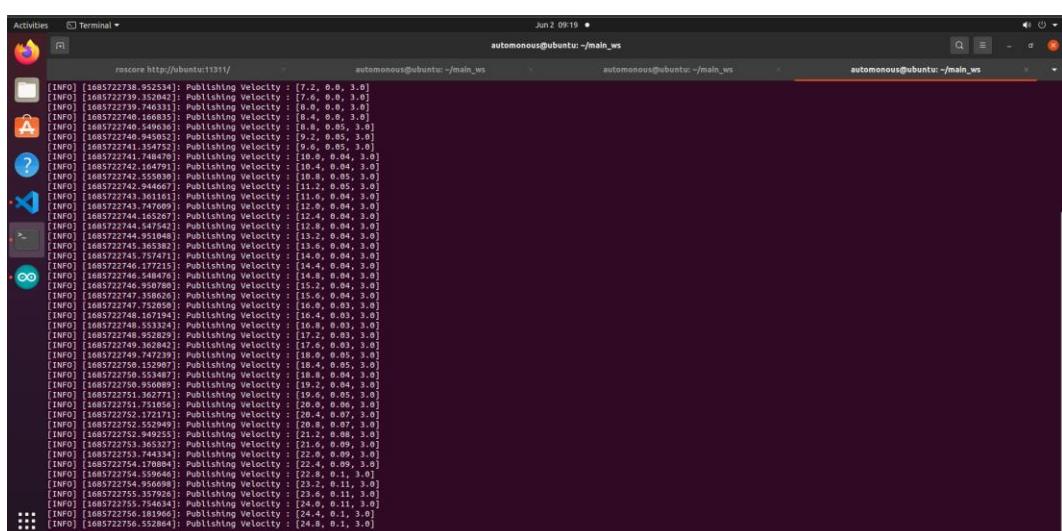
#### 4.4.1 ตัวอย่างผลการทดลองของ Robot Operating System ในการส่งข้อมูลไปยังระบบควบคุมความเร็ว



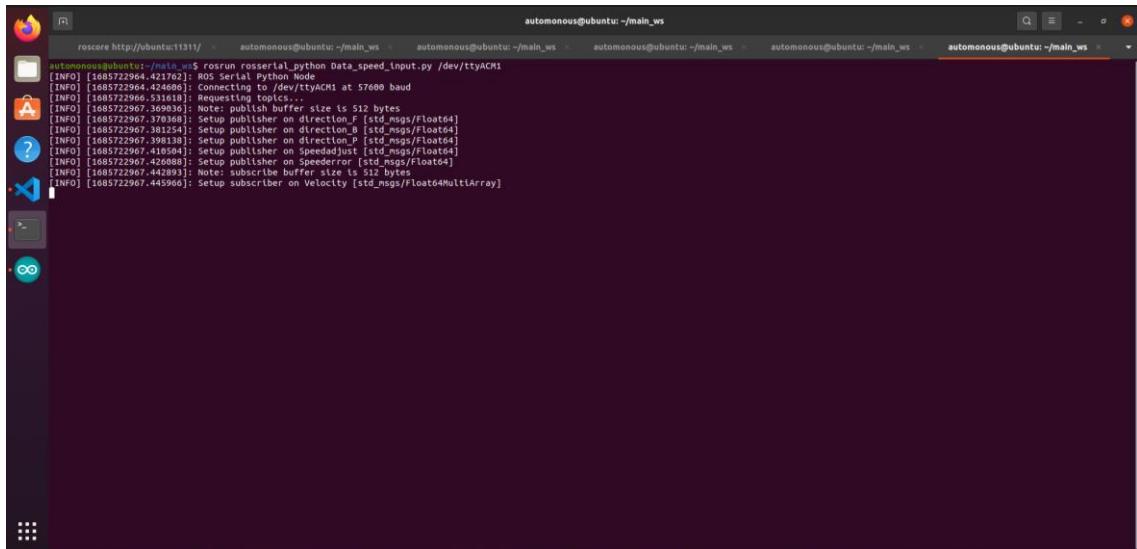
รูปที่ 4.88 ภาพแสดง rqt\_graph สำหรับการรับและส่งข้อมูลของระบบควบคุมความเร็ว



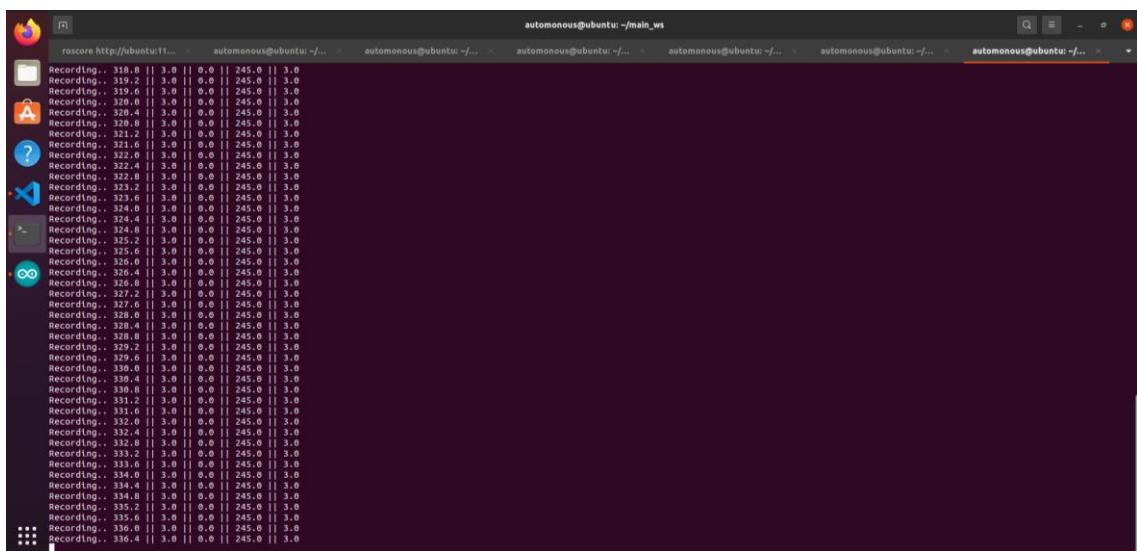
รูปที่ 4.89 ภาพตัวอย่างการ Run Ros Master



รูปที่ 4.90 ภาพตัวอย่างการ Run Velocity Calculation เพื่อส่งข้อมูลให้ Node Data speed input



รูปที่ 4.91 ภาพตัวอย่างการ Run การทำงานของ Arduino



รูปที่ 4.92 ภาพตัวอย่างการ Run Node Data speed input เพื่อรับข้อมูลจาก Velocity Calculation

ในการควบคุมการเคลื่อนที่ของรถให้คงที่ จะเป็นการรับ-ส่งข้อมูลระหว่าง Node Velocity Calculation, Node Speed Control และ Node Data speed input (Arduino) โดย Velocity Calculation จะทำหน้าที่เป็นตัวรับและตัวส่งข้อมูลพร้อมกัน โดยเมื่อ Velocity Calculation ทำหน้าที่เป็นตัวรับข้อมูลจะทำการรับค่า Lat/Long มาจาก Node Localization เพื่อใช้ในการคำนวณเป็นค่าความเร็ว (Km/hr) และเมื่อทำหน้าที่เป็นตัวส่งข้อมูลจะทำการส่งค่าความเร็วที่ได้จากการคำนวณเข้า Node Data speed input เพื่อในการคำนวณหา Error ในสมการ PID ในขณะเดียวกัน Node Data speed input ก็จะทำหน้าที่เป็นตัวส่งข้อมูลไปยัง Node Speed Control ซึ่งเป็น Node ที่รับค่าการทำงานมาจาก Arduino เพื่อมาแสดงผล เช่น การแสดงผลค่าความเร็วของรถ, แสดงค่าที่ได้จากการ PID Controller หรือ ค่า Digital potentiometer เป็นต้น

### **สรุปผลการทดลองการรับ-ส่งข้อมูลผ่าน Robot Operating System**

จากการทดลองการรับ-สื่อข้อมูลผ่าน Robot Operating System พบว่าการสื่อสารทั้งหมดในระบบ ROS สามารถใช้งานได้อย่างถูกต้องและความเร็วในการส่งข้อมูลของทุก Node จะขึ้นอยู่กับความเร็วในการ Sampling ค่าของ GNSS Sensor ตามที่ได้กำหนดไว้ ซึ่งในการทำงานจริงจะใช้ความเร็วอยู่ที่ 7 เฮิร์ตซ์ เพียงพอต่อการควบคุมการทำงานทุกระบบท้ายในรถ

## บทที่ 5 สรุป

### 5.1 สรุปผลการทดลอง

สำหรับระบบระบุตำแหน่งจะเห็นได้ว่าคุณภาพของสัญญาณที่รังวัดได้ทั้งในแบบ Real Time Kinematic และ Network RTK ในการเก็บพิกัดระยะยาวมี GPS Quality การทดลองเบริกเทียบคุณภาพของพิกัดผ่านการรังวัดในระยะทางยาวบริเวณรอบสนามกีฬา ค่อนข้างแตกต่างกันพอสมควร โดยเทคนิครังวัดแบบ Real Time Kinematic มีเปอร์เซ็น FIXED Quality เนลี่ยอยู่ที่ 86.73 % และแบบ Network RTK อยู่ที่ 48.89 % ทั้งนี้สาเหตุเกิดมาจากการสภาพแวดล้อมบริเวณรอบสนามมีสิ่งกีดขวางมาก เช่น ตึกหรืออาคาร ทำให้เกิดความคลาดเคลื่อนต่อค่าพิกัดที่วัด โดยหากต้องการใช้งานในบริเวณพื้นที่ที่มีสิ่งกีดขวางมากการเลือกใช้การรังวัดแบบ Real Time Kinematic จะมีความละเอียดที่สูงมากกว่าการรังวัดแบบ Network RTK แต่การรังวัดแบบ Network RTK ความละเอียดก็ยังสามารถยอมรับได้ และการรังวัดแบบ Network RTK จะใช้อุปกรณ์ในการวัดเพียง 1 ชุด ประกอบด้วย Receiver Board และ เสาอากาศ Antenna ซึ่งต่างจากวิธีการรังวัดแบบ Real Time Kinematic ที่จำเป็นต้องใช้อุปกรณ์ถึง 2 ชุดในการติดตั้งเป็น Base Station และ Rover Station ดังนั้นสำหรับการเลือกใช้งานจะเลือกใช้การรังวัดแบบ Network RTK เนื่องจากใช้อุปกรณ์ที่น้อยกว่าและสามารถช่วยลดต้นทุนในการพัฒนาได้

สำหรับระบบควบคุมความเร็วให้คงที่ด้วย PID Controller จะให้ได้ว่าในโครงงานก่อนหน้ากำหนดความเร็วของรถอยู่ที่ 5 กิโลเมตรต่อชั่วโมงรอบสนามฟุตบอล ในกรณีที่ผู้ใช้งานควบคุมพวงมาลัยแบบแม่นวลด การเลือกใช้ความเร็วที่ 5 กิโลเมตรต่อชั่วโมงนั้นสามารถใช้งานได้ตลอดเส้นทางรอบสนามฟุตบอล แต่เมื่อนำระบบควบคุมความเร็วมารวมเข้ากับระบบควบคุมมุมเลี้ยวแบบอัตโนมัติแล้ว ในขณะที่รถเคลื่อนที่อยู่ในบริเวณทางโค้ง หากใช้ความเร็วที่ 5 กิโลเมตรต่อชั่วโมงเช่นเดิม จะมีโอกาสทำให้รถเคลื่อนที่หลุดจากทางโค้งอ้างอิงได้ เนื่องจากประสิทธิภาพของระบบควบคุมพวงมาลัยนั้นยังมีการตอบสนองที่ช้าอยู่ ดังนั้นจึงต้องมีการเปลี่ยนแปลงค่าความเร็วตามลักษณะเส้นทางจริง โดยค่าของความเร็วรอบสนามฟุตบอลจะถูกแบ่งเป็น 3 ช่วงดังนี้ ช่วงที่ 1 คือช่วงทางโค้ง ในช่วงนี้จะเลือกใช้ความเร็วที่ 1 กิโลเมตรต่อชั่วโมง เพื่อบริโภคกันรถเคลื่อนที่หลุดออกจากเส้นโค้งอ้างอิงตามที่ได้กำหนดไว้ในข้างต้น ช่วงที่ 2 คือช่วงทางตรงและช่วงเส้นทางโค้งที่ถนนมีลักษณะเป็นเนิน โดยในช่วงนี้จะเลือกใช้

ความเร็วที่ 3 กิโลเมตรต่อชั่วโมง และในช่วงสุดท้าย คือ ช่วงที่เส้นทางมีความคลาดเคลื่อนของพิกัดไม่คงที่ ได้แก่ เส้นทางบริเวณหน้าโรงอาหาร โดยในช่วงนี้จะเลือกใช้ความเร็วที่ 1 กิโลเมตรต่อชั่วโมง เพื่อความปลอดภัยในการใช้งานในการนี้ที่พิกัดขณะทดลองเกิดการแก่วงหรือคลาดเคลื่อน

สำหรับระบบควบคุมทิศทางและบังคับมุ่งเลี้ยว ได้ดำเนินการในการเขียนโค้ดเพื่อควบคุมมุ่งเลี้ยวในการหมุนของมอเตอร์พวงมาลัยผ่าน Steering Control Node ได้สำเร็จ โดยสามารถส่งสัญญาณดิจิตาร์เป็นมุมองศาในเลขฐานสิบผ่าน Lateral Control Node จากนั้นจะถูกแปลงเป็นเลขฐานสิบหกและคำนวนค่าให้ถูกต้องตามข้อจำกัดของมอเตอร์พวงมาลัย และภายในระบบ Lateral Control Node จะประกอบด้วยส่วนของ PID Controller ซึ่งเป็นตัวควบคุมการเปลี่ยนแปลงมุมสำหรับควบคุมมอเตอร์พวงมาลัยที่จะเกิดขึ้นโดยจะต้องนำพิกัดปัจจุบันของตัวรถที่ได้จากระบบระบุตำแหน่งและเส้นทางอ้างอิงที่ได้บันทึกไว้ มาคำนวนหาค่า Cross Track Error (CTE) รวมทั้งคำนวนมุ่งเลี้ยวของรถด้วยสมการ PID เพื่อบังคับมุ่งเลี้ยวของพวงมาลัย และ ทำการปรับปรุงค่า Gain ของระบบ โดยการทดลองใช้รถกล้อฟริ่งกับเส้นทางอ้างอิงหลายครั้ง เพื่อปรับแก้และเลือกใช้ค่า Gain ที่เหมาะสมกับการเดินทางแต่ละช่วง

## 5.2 ปัญหาที่พบ

### 5.2.1 ระบบระบุตำแหน่ง

- สำหรับการรังวัดในรูปแบบ Real Time Kinematic มีความยุ่งยากในการติดตั้ง Base Station เนื่องจาก การติดตั้ง Base Station จำเป็นต้องเลือกสถานที่ติดตั้งที่ห้องฟ้าโปร่ง สิ่งกีดขวางน้อย และถ้าหากมีการเคลื่อนย้าย Base Station จะใช้เวลาในการค้นหาพิกัดที่แน่นอนค่อนข้างนานพอสมควร
- ตำแหน่งพิกัดที่รังวัดได้มีความคลาดเคลื่อนหรือมีการแก่วง ในวันที่สภาพอากาศปิดหรือในช่วงที่ฝนตก

### 5.2.2 ระบบควบคุมความเร็วให้หักที่

1. ปัญหาเกี่ยวกับตัววงจรเนื่องจากน้ำหนักรของเดิมมาใช้ ทำให้มีอุปกรณ์บางตัวเสียหายบางส่วน เช่น สายไฟที่เชื่อมจากวงจรเข้า Controller รถกอล์ฟ
2. ปัญหาด้านการควบคุมความเร็วผ่านวงจรไม่สามารถใช้รูปแบบวงจรเก่าได้แม้การทำงานของรถกอล์ฟ จะเหมือนกัน เช่น เมื่อนำวงจรเดิมไปต่อเข้ากับรถกอล์ฟพบแรงดันไฟลอกอกมาทุกช่องที่ Terminal ควบคุมการทำงาน เป็นต้น
3. ไม่สามารถวัดความเร็วจากการอ่านค่าที่ Encoder ได้ เนื่องจากมีสัญญาณรบกวนมากเกินไปทำให้ตัวประมวลผลไม่สามารถนับสัญญาณจริงๆของ Encoder ได้
4. การคำนวณความเร็วจาก Lat/Long หากอยู่ในจุดที่มีตึก หรือ จุดอับสัญญาณค่าที่ได้จากการคำนวณจะมีความผิดพลาดสูง
5. ความเร็วเดิมของรถตัด หรือข้าลง

### 5.2.3 ระบบควบคุมทิศทางและบังคับมุ่งเลี้ยว

1. ค่ามุนที่ตอบสนองกลับมาจากความเร็วเดิมของรถกอล์ฟ ที่มีค่าที่ไม่คงที่อย่างสม่ำเสมอ โดยในช่วงเริ่มต้นสั่งงานจะเกิดการแกว่งสูง ทำให้ความคลาดเคลื่อนสูงมากและค่อยๆ ลดลงจนค่าความคลาดเคลื่อนเข้าใกล้ 0 ในระยะเวลาที่เท่ากันและการควบคุมมุนในการเลี้ยวชั่วคราวเป็นจำนวน 6 ครั้ง พบว่าค่ามุนที่ตอบสนองจากมอเตอร์กลับมาจริงๆ นั้นไม่ตรงตามการคำนวณทางทฤษฎี จึงไม่สามารถนำค่ามุนที่ตอบสนองจากมอเตอร์มาคิดแบบ Closed loop ได้
2. ในส่วนของการควบคุมมุนเลี้ยวจะได้รับผลกระทบจากตำแหน่งพิกัดที่อาจเกิดความคลาดเคลื่อนขึ้นเนื่องจากสภาพแวดล้อมต่างๆ เช่น สภาพภูมิอากาศ หรือการสะท้อนของอาคารขนาดใหญ่ ที่มีผลทำให้พิกัดนั้นเกิดความคลาดเคลื่อนขึ้น จึงทำให้ข้อมูลของถนนที่ได้ระบุนั้นเกิดความคลาดเคลื่อนตามมาตั้งนั้นจึงกระทบไปปัจจัยการคำนวณหาค่า CTE, PID และ Steering Angle สำหรับบังคับมุนเลี้ยวด้วย
3. สภาพการใช้งานของรถกอล์ฟที่ใช้ในงานวิจัยนี้มีผลต่อการควบคุมต่างๆ เช่นในการทดสอบรถยานแบบ ทำให้ล้อกระทำการกับพื้นถนนมากขึ้น จึงเกิดแรงเสียดทานเพิ่มขึ้น มีผลต่อการปรับค่าเกณฑ์ในการควบคุมมุนมุนเลี้ยวให้สามารถเปลี่ยนแปลงได้เหมาะสม รวมทั้งเมื่อเบนเตอร์ของรถอ่อนก็มีผลต่อการดึงกระแสไฟฟ้ามาใช้สำหรับมอเตอร์พวงมาลัย

### 5.3 วิธีการแก้ปัญหาและคำแนะนำ

#### 5.3.1 ระบบระบุตำแหน่ง

1. ติดตั้ง Base Station ในสถานที่ที่สามารถติดตั้งได้อย่างถาวรเพื่อลดเวลาในการค้นหาพิกัดที่แน่นอน
2. ในการณ์ที่พิกัดที่วัดได้เกิดการแก่วงจะนำ Moving Average Filter มาใช้ในการกรองตำแหน่งพิกัดที่ได้ให้มีความต่อเนื่องมากขึ้น
3. ในการณ์ที่พิกัดที่วัดได้ในขณะนี้เกิดการคลาดเคลื่อน ให้ลองขับรถเคลื่อนที่รอบสนามฟุตบอลจำนวนสองถึงสามครั้ง จากนั้นลองอ่านค่าพิกัดหลังจากนั้นว่ามีความแม่นยำขึ้นหรือไม่ ถ้าหากยังมีความคลาดเคลื่อนอยู่ก็อาจจะต้องทำการทดสอบในช่วงเวลาที่สภาพอากาศดีขึ้น

#### 5.3.2 ระบบควบคุมความเร็วให้คงที่

1. ทำการศึกษารูปแบบวงจรใหม่ และทำการต่อสายใหม่เพื่อให้มีความเป็นระบบมากขึ้น หรือถูกใช้งานง่ายยิ่งขึ้น รวมไปถึงทำการตรวจสอบการทำงานของแต่ละอุปกรณ์ว่าปัญหาเกิดที่อุปกรณ์ตัวไหน
2. เปลี่ยน GND เนื่องจากเดิมเป็น GND ที่มาจากการเหยียบคันเร่งทำให้มีโอกาสที่ค่าจะผิดพลาดได้ ซึ่ง GND ใหม่ที่นำมาใช้มาจาก Speed Sensor ของรถที่มีความแม่นยำมากกว่า
3. เปลี่ยนวิธีการอ่านค่าความเร็วจากเดิมเป็นการอ่านค่าจาก Encoder เป็นการนำค่า Lat/Long ที่ได้จากระบบวัดพิกัดเป็นตัวคำนวณหากความเร็วในการเคลื่อนที่แทน
4. ลดความเร็วลงเพื่อเพิ่มจำนวนในการ Sampling ข้อมูลให้มีความละเอียดมากขึ้น
5. เริ่มตรวจสอบสายที่ต่อเข้า Controller ,แบนตเตอร์ หรือบิร์มาณ์ก้าลั่นที่อยู่ในแบนตเตอร์ เนื่องจากถ้าหากกลั่นน้อบส่งผลให้กระแสที่ออกจากแบนตเตอร์นั้นองลงตาม

#### 5.3.3 ระบบควบคุมทิศทางและบังคับมุ่งเลี้ยว

1. เนื่องจากมุ่งที่ตอบสนองกลับมาจากมอเตอร์นั้นมีค่าที่ไม่คงที่อย่างสม่ำเสมอ จึงไม่ได้คำนึงถึงการทำ Closed-loop หรือการคำนวณผ่าน PID Controller อิกครั้ง ดังนั้นในส่วนของ Steering Control Node จึงควบคุมแบบ Open-loop
2. สภาพแวดล้อมในการทดลองดำเนินการมีผลอย่างมากในการทำงาน จึงสามารถแก้ไขสถานการณ์ได้โดย หากสภาพภูมิอากาศมีฝนฟ้าคะนอง เมฆทึบ ฟ้าไม่โปร่ง จะไม่สามารถทำการทดลองสำหรับการขับเคลื่อนรถอัตโนมัติได้ หรือในการณ์ที่ขับเคลื่อนรถอัตโนมัติผ่านอาคารขนาดใหญ่ ซึ่งส่งผล

คลาดเคลื่อนต่อสัญญาณในการรับค่าพิกัด ทำให้ตำแหน่งในการเคลื่อนที่ของรถนั้นเกิดความคลาดเคลื่อนขึ้น จึงแก้ไขได้ด้วยการขับเคลื่อนรถด้วยความเร็วต่ำในที่นี่จะใช้ความเร็วประมาณ 1 กิโลเมตรต่อชั่วโมง หรือสามารถใช้เทคนิคการวัดพิกัดแบบ RTK ใน การรับตำแหน่งที่แม่นยำเพิ่มขึ้นแทนได้ในกรณีนี้

3. การแก้ไขสภาพการใช้งานของรถก่อตัวการตรวจสอบแบบเดอร์ร์ น้ำกลั่น หรือลมยางอย่างสม่ำเสมอ ในกรณีที่ยางแบบนี้มีความตึงลมยางให้มีค่าแรงดันอยู่ที่ 25-30 psi หากจอดรถทิ้งไว้เฉยๆ ไม่ได้ใช้งาน ควรขับรถบ่ายๆ ที่ยางจะต้องรับน้ำหนักรถ สัปดาห์ละครั้งเพื่อเป็นการรักษาสภาพยางให้คงทนได้ไว้อยู่เสมอ และไม่ควรจอดไว้ในพื้นที่ที่อุณหภูมิสูงจนเกินไป จะส่งผลให้ยางแตกลายง่ายได้ และควรตรวจสอบน้ำกลั่นอย่างสม่ำเสมอ

## บรรณานุกรม

- [1] Tilbury, D. and Messner, B., 2562, “**PID Controller Design**”, [Online], Available: <https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID> [7 กันยายน 2565]
- [2] Sergio, L. and Felix, F., 2565, “**Basics of model predictive control**”, [Online], Available: [https://www.do-mpc.com/en/latest/theory\\_mpc.html](https://www.do-mpc.com/en/latest/theory_mpc.html) [7 กันยายน 2565]
- [3] Areerob, P. and Panomruttanarug, B., 2566, “Iterative Learning Control Based Lateral Tracking for Autonomous Vehicles”, **Journal of Intelligent & Robotic Systems**, Vol. 1, No.1, pp.5-8
- [4] Boonruam, N., Ketnoi, T. and Kumar Ojha, A., 2564, “**The Autonomous Vehicle: Trajectory Tracking Using PID Controller**”, Bachelor of Engineering Program, Department of Control systems and Instrumentation Faculty of Engineering King Mongkut’s University of Technology Thonburi, pp.42-45,49-52
- [5] Samadhi, M., Swapna, R.T. and Veda, C.M., 2558, “Position Control of DC Motor Using PID Controller”, **International Journal of Scientific Engineering and Applied Science**, Vol. 1, No. 3, pp. 436-442.
- [6] Panomruttanarug, B., Thurnim, L., Kornwong, A., Promdum, S. and Tangwongsan, P., 2565, “**Practical Speed Control for an Autonomous Golf Cart**”, pp.302-304
- [7] Falch, M., 2564, “**CAN Bus Explained**”, [Online], Available: <https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial> [9 กันยายน 2565].

- [8] Jinankeya Electron Science and Technology Co., Ltd., 2554, “**KY170C Electric Steering Motor User Manual (V1.5)**”, [Online], Available: [https://mailkmattacth-my.sharepoint.com/:b/g/personal/surayuth\\_inc\\_mail\\_kmutt\\_ac\\_th/EXE4PM4JET5Dl20mRJgyEpAB9cLBkggThKH2jTn3iZwGcA?e=IMGFmg](https://mailkmattacth-my.sharepoint.com/:b/g/personal/surayuth_inc_mail_kmutt_ac_th/EXE4PM4JET5Dl20mRJgyEpAB9cLBkggThKH2jTn3iZwGcA?e=IMGFmg) [9 กันยายน 2565].
- [9] Ken Robotsiam. 2561, “**Robot Operating System**”, [Online], Available: <https://rosthai.blogspot.com/2018/09/introduction-ros.html> [7 กันยายน 2565].
- [10] Sontaya, R., 2564, “**Analysis and Evaluation of GNSS Signal Quality of Plate Tectonics Measuring Station**”, [Online], Available: <https://earthquake.tmd.go.th/documents/file/seismo-doc-1601617083.pdf> [9 กันยายน 2565].
- [11] ศูนย์ข้อมูลค่าอ้างอิงพิกัดแบบต่อเนื่องแห่งชาติ, 2565, “**ข้อมูลเกี่ยวกับศูนย์ข้อมูลค่าอ้างอิงพิกัดแบบต่อเนื่องแห่งชาติ**”, [Online], Available: <https://ncdc.in.th/portal/apps/sites/#/ncdc> [7 กันยายน 2565].
- [12] ArduSimple, 2561, “**How to connect simpleRTK2B series receiver to a NTRIP correction service via PC**”, [Online], Available: <https://www.ardusimple.com/simplertk2b-performance-in-standalone-mode-and-ntrip-corrections/> [9 กันยายน 2565].
- [13] Dutch, S., 2563, “Converting UTM to Latitude and Longitude (Or Vice Versa)”, **Natural and Applied Sciences**, [Online], Available: [https://stevedutch.net/usefuldata/utmformulas.htm?fbclid=IwAR2ckyuNaentd9UVFc\\_ewmPRTL1wOsUKkU\\_HZ7KkU6yBMs-zS4\\_LYel1WV4](https://stevedutch.net/usefuldata/utmformulas.htm?fbclid=IwAR2ckyuNaentd9UVFc_ewmPRTL1wOsUKkU_HZ7KkU6yBMs-zS4_LYel1WV4) [9 กันยายน 2565].
- [14] John, M., 2565, “**Lat/Long to UTM conversion failing for dataframe**”, [Online], Available: <https://stackoverflow.com/questions/70308792/lat-long-to-utm-conversion-failing-for-dataframe> [9 กันยายน 2565].

- [15] marXact B.V., 2564, “**What is the difference between RTK, FIX, and RTK Float?**”, [Online], Available: <https://support.marxact.com/article/85-what-is-the-difference-between-rtk-fix-and-rtk-float> [10 กันยายน 2565].
- [16] Hexagon, 2565, “**GPS fix data and undulation**”, [Online], Available: <https://docs.novatel.com/OEM7/Content/Logs/GPGGA.htm> [10 กันยายน 2565].
- [17] กำจัด ใจตรง, สงกรานต์ ภารกุล และ ณรงค์ฤทธิ์ อี้มเจริญพรสกุล, 2564, “การควบคุมความเร็ว รอบของเซอร์โวมอเตอร์ไฟฟ้ากระแสตรงด้วยตัวควบคุมแบบพีไอดี โดยใช้ ไมโครคอนโทรลเลอร์ระดับ PIC”, วารสารวิทยาศาสตร์และเทคโนโลยี มหาสารคาม, ปีที่ 5, ฉบับที่ 1.
- [18] ชนะชัย คงชนะวิจิตร, 2562, การศึกษาการควบคุมความเร็วรอบมอเตอร์กระแสตรงโดยใช้ตัว ควบคุมพีไอ, วิทยานิพนธ์ปริญญาวิศวกรรมศาสตร์มหบันฑิต สาขาวิชา วิศวกรรมไฟฟ้า คณะ วิศวกรรมศาสตร์ มหาวิทยาลัยสยาม.
- [19] Karr, D., 2565, “**How to calculate or query the great circle distance between points of latitude and longitude using the haversine formula**”, [Online], Available: <https://th.martech.zone/calculate-great-circle-distance/> [25 มกราคม 2565].
- [20] Chris V., 2565, “**Calculate distance, bearing and more between Latitude/Longitude points**”, [Online], Available: <https://www.movable-type.co.uk/scripts/latlong.html> [28 มกราคม 2565].
- [21] Sleat, A., 2554, “**ROS: Publishing and Subscribing to Arrays**”, [Online], Available: <https://alex sleat.co.uk/2011/07/02/ros-publishing-and-subscribing-to-arrays/> [4 มกราคม 2565].
- [22] Thomas, M., 2557, “**rosserial/Overview/Initialization**”, [Online], Available: <http://wiki.ros.org/roserial/Overview/Initialization> [4 มกราคม 2565].

- [23] Woodall, W., 2558, “**rosCPP/Overview/Initialization and Shutdown**”, [Online], Available: <http://wiki.ros.org/roscpp/Overview/Initialization%20and%20Shutdown> [4 มกราคม 2565].
- [24] Bhargave Sanjay, S., Jadhav Vijay, M. and Patil Sunil, A., 2561, “Automatic Vehicle Speed Control System”, **International Journal of Innovative Science and Research Technology**, Vol. 3, No. 1, pp. 535-538.
- [25] Bhadani et al., 2565, “Strym: A Python Package for Real-time CAN Data Logging, Analysis and Visualization to Work with USB-CAN Interface”, **Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities Workshop (DI-CPS)**, Vol. 2, No. 1.
- [26] สนชยา นงนุช, 2563, “**CAN bus ตอนที่ 1 ทฤษฎี CAN**”, [Online], Available: <https://www.artronshop.co.th/article/101/can-bus-%E0%A1%AA-%E0%A1%AA-can> [10 ตุลาคม 2565].
- [27] Smith Maloy, G., 2564, “**What Is CAN Bus (Controller Area Network) and How It Compares to Other Vehicle Bus Networks**”, [Online], Available: <https://dewesoft.com/blog/what-is-can-bus> [10 ตุลาคม 2565].
- [28] Falch M., 2565, “**CAN Bus Explained - A Simple Intro**”, [Online], Available: <https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial> [10 ตุลาคม 2565].
- [29] Choudhary, M., 2562, “**How GNSS Works**”, [Online], Available: <https://www.geospatialworld.net/blogs/how-gnss-works/> [10 ตุลาคม 2565].
- [30] Jittagorn, P., 2554, “**การคำนวณระยะห่าง ของตำแหน่ง GPS 2 จุด**”, [Online], Available: <https://na5cent.blogspot.com/2011/07/gps-2-javascript.html> [10 ตุลาคม 2565].
- [31] Ding, Y., 2555, “**Simple Understanding of Kinematic Bicycle Model**”, [Online], Available: <https://dingyan89.medium.com/simple-understanding-of-kinematic-bicycle-model-81cac6420357> [10 ตุลาคม 2565].

## ภาคผนวก ก

โค้ดการระบุตำแหน่งปัจจุบันของรถผ่าน Robot Operating System

```
#!/usr/bin/env python3

import serial
import pynmea2
import utm
import csv
import rospy
from std_msgs.msg import Float64MultiArray

class main(object):

    def __init__(self):

        # GNSS Function
        self.start = 0
        self.raw_lat = 0.0
        self.raw_lng = 0.0
        self.time_hr = 0.0
        self.time_min = 0.0
        self.time_sec = 0.0
        self.quality = 0
        self.time_start = 0.0
        self.time_operate = 0.0

        #StreamingMovingAverage Funtion
        self.window_size = 1
        self.lat_buffer = []
        self.sum_lat = 0.0
        self.latitude = 0.0
        self.lng_buffer = []
        self.sum_lng = 0.0
        self.longitude = 0.0

        #UTMconvert Function
        self.utm_frame = []
        self.x_east = 0.0
        self.y_north = 0.0
        self.utm_position = []

        # CreateCSV Function
        self.value = []
        self.name = 'Data Column Name'
        self.column = 0
        self.csv_name = 0
        self.Data = []
```

```

        self.Rt_xy_pub =
rospy.Publisher('Realtime_XY',Float64MultiArray,queue_size=10)
        self.Time_lat_lon_Pub =
rospy.Publisher('Time_Lat_Lon',Float64MultiArray,queue_size=10)

        while True:
            self.GNSS()

def GNSS(self):
    GNSS_frame = sercfg.readline()
    GNSS_buffer = GNSS_frame.split(b",")

    if GNSS_buffer[0] == b"$GNGGA":
        # print(GNSS_buffer)
        newmsg=pynmea2.parse(GNSS_frame.decode("utf-8"))
        self.raw_lat = newmsg.latitude
        self.raw_lng = newmsg.longitude
        self.time_hr = newmsg.timestamp.hour
        self.time_min = newmsg.timestamp.minute
        self.time_sec = newmsg.timestamp.second +
newmsg.timestamp.microsecond/1000000
        self.quality = newmsg.gps_qual

        if (self.quality == 4)or(self.quality == 5):
            self.StreamingMovingAverage()
            self.UTMconvert()
            if (self.latitude > 0.0)&(self.longitude > 0.0)&(self.x_east
> 0.0)&(self.y_north > 0.0):

                if self.start == 0:
                    self.time_start = (self.time_min*60) + self.time_sec
                    self.start = 1

                if self.start == 1:
                    self.time_operate = round((self.time_min*60) +
self.time_sec - self.time_start,2)
                    self.Talker()
                    # self.CreateCSV()

def StreamingMovingAverage(self):

    self.lat_buffer.append(self.raw_lat)
    self.sum_lat = self.sum_lat + self.raw_lat
    if len(self.lat_buffer) > self.window_size:
        self.sum_lat = self.sum_lat - self.lat_buffer.pop(0)

```

```

        self.latitude = float(self.sum_lat)/len(self.lat_buffer)

        self.lng_buffer.append(self.raw_lng)
        self.sum_lng = self.sum_lng + self.raw_lng
        if len(self.lng_buffer) > self.window_size:
            self.sum_lng = self.sum_lng - self.lng_buffer.pop(0)
            self.longitude =
float(self.sum_lng)/len(self.lng_buffer)

def UTMconvert(self):

    self.utm_frame = utm.from_latlon(self.latitude,self.longitude)
    self.x_east = self.utm_frame[0]
    self.y_north = self.utm_frame[1]
    self.utm_position = self.x_east,self.y_north

def Talker(self):

    rospy.loginfo("Localization Nodes")
    Realtime_xy = Float64MultiArray()
    Realtime_xy.data = [self.time_operate ,self.x_east,self.y_north]
    self.Rt_xy_pub.publish(Realtime_xy)
    rospy.loginfo('Publishing RT XY :
%s',Realtime_xy.data)

    Time_lat_lon = Float64MultiArray()
    Time_lat_lon.data = [self.time_hr ,self.time_min ,self.time_sec
, self.latitude ,self.longitude ,self.raw_lat ,self.raw_lng]
    self.Time_lat_lon_Pub.publish(Time_lat_lon)
    rospy.loginfo('Pubilshing Times Lat/Lon : %s',Time_lat_lon.data)

def CreateCSV(self):

    self.value = self.time_operate , self.x_east , self.y_north ,
self.latitude , self.longitude
    self.name =
'time_operate,self.x_east,self.y_north,self.latitude,self.longitude'

    if self.time_operate <= 40 :

        name = 'start_3.csv'      #change name csv

        if self.csv_name == 0:

            for i in range(len(self.name.split(','))):
```

```
        self.Data.append(self.name.split(',')[self.'][i])
# print(self.Data)

    with open(name, 'a',newline='') as f:
        writer = csv.writer(f,delimiter=",")
        writer.writerow(self.Data)
        self.csv_name = 1

if self.csv_name == 1:

    self.Data = self.value
# print(self.Data)

    with open(name, 'a',newline='') as f:
        writer = csv.writer(f,delimiter=",")
        writer.writerow(self.Data)
        self.csv_name = 1

if self.time_operate > 40 :
    print("Stop...")

if __name__ == '__main__':
    port = "/dev/ttyUSB2"

    sercfg = serial.Serial(port ,baudrate = 115200)

    rospy.init_node('Localization',anonymous=True)
    try:
        cls = main()
    except rospy.ROSInterruptException:
        pass
```

## **ภาคผนวก ฯ**

โภคดีคำนวนความเร็วปั๊มน้ำจากพิกัดตำแหน่งผ่าน Robot Operating System

```
#!/usr/bin/env python3
import rospy
from std_msgs.msg import Float64MultiArray,Float64
from math import radians, cos, sin, asin, sqrt, atan2, degrees
import csv

class main(object):

    def __init__(self):

        #Speed_Setpoint_callback Function
        self.speed_sp = 0.0

        #Time_Lat_Lon_callback Function
        self.data = None
        self.time_hr = 0.0
        self.time_min = 0.0
        self.time_sec = 0.0
        self.lat = 0.0
        self.lon = 0.0
        self.raw_lat = 0.0
        self.raw_lon = 0.0
        self.prev_data = None
        self.time_start = 0.0

        #Time_Counter Function
        self.time_operate = 0.0

        #Haversine_raw Function
        self.now_lat = 0.0
        self.now_lon = 0.0
        self.prev_lat = 0.0
        self.prev_lon = 0.0
        self.dlat = 0.0
        self.dlon = 0.0
        self.distance_km_raw = 0.0
        self.distance_m_raw = 0.0

        #Movingfilter_raw Function
        self.raw_speed_buffer = []
        self.sum_raw_speed = 0.0
        self.window_size = 20
        self.raw_speed_filter = 0.0

        #Calculate Function
```

```

    self.prev_sec = 0.0
    self.now_sec = 0.0
    self.raw_speedms = 0.0
    self.raw_speed = 0.0

    # CreateCSV Function
    self.value = ()
    self.name = 'Data Column Name'
    self.csv_name = 0
    self.Data = []

    self.Velocity_Pub =
rospy.Publisher('Velocity',Float64MultiArray,queue_size=10)
    rospy.Subscriber("Time_Lat_Lon",Float64MultiArray,self.Time_Lat_Lon_
callback)
    rospy.Subscriber("Speed_Setpoint",Float64,self.Speed_Setpoint_callba
ck)
    rospy.spin()

def Speed_Setpoint_callback(self,msg):
    self.speed_sp = msg.data

def Time_Lat_Lon_callback(self,msg):      #For Get Message from
Localization Nodes

    self.data = msg.data
    self.time_hr = msg.data[0]
    self.time_min = msg.data[1]
    self.time_sec = msg.data[2]
    self.lat = msg.data[3]
    self.lon = msg.data[4]
    self.raw_lat = msg.data[5]
    self.raw_lon = msg.data[6]

    if self.prev_data is None:

        self.time_start = (self.data[1]*60) + self.data[2]

    if self.prev_data is not None:

        self.Time_Counter()
        self.Haversine_raw()
        self.Calculate()
        self.Talker()
        # self.CreateCSV()

```

```

    self.prev_data = msg.data

def Time_Counter(self):                      #For Count Time Operate

    self.time_operate = round(((self.data[1]*60) + self.data[2]) -
self.time_start,2)

def Haversine_raw(self):                     #For Raw Latitude/Longitude

    self.now_lat = self.data[5]
    self.now_lon = self.data[6]

    self.prev_lat = self.prev_data[5]
    self.prev_lon = self.prev_data[6]

    self.now_lat, self.now_lon, self.prev_lat, self.prev_lon =
map(radians,[self.now_lat, self.now_lon, self.prev_lat, self.prev_lon])

    self.dlat = self.prev_lat - self.now_lat
    self.dlon = self.prev_lon - self.now_lon

    a = sin(self.dlat/2)**2 + cos(self.now_lat) * cos(self.prev_lat) *
sin(self.dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # Radius of earth in kilometers. Use 3956 for miles.
Determines return value units.

    self.distance_km_raw = c*r
    self.distance_m_raw = self.distance_km_raw*1000

def Movingfilter_raw(self):                  #For Filter Speed after
Calculate Function

    self.raw_speed_buffer.append(self.raw_speed)
    self.sum_raw_speed = self.sum_raw_speed + self.raw_speed

    if len(self.raw_speed_buffer) > self.window_size:
        self.sum_raw_speed = self.sum_raw_speed -
self.raw_speed_buffer.pop(0)
        self.raw_speed_filter =
float(self.sum_raw_speed)/len(self.raw_speed_buffer)
        self.raw_speed_filter = round(self.raw_speed_filter,2)

def Calculate(self):                        #For Calculate Speed

```

```

    self.prev_sec = self.prev_data[2]
    self.now_sec = self.data[2]

    if self.now_sec > self.prev_sec:

        self.raw_speedms = self.distance_m_raw / (self.now_sec -
self.prev_sec)                                #Speed from Raw lat/lon
        self.raw_speed = round(self.raw_speedms*3.6 ,2)
        self.Movingfilter_raw()

def Talker(self):                               #For Publish Velocity Data

    Velocity = Float64MultiArray()
    Velocity.data = [self.time_operate ,self.raw_speed_filter
, self.speed_sp]
    self.Velocity_Pub.publish(Velocity)
    rospy.loginfo('Publishing Velocity : %s',Velocity.data)

def CreateCSV(self):

    self.value = self.time_operate , self.raw_speed ,
self.raw_speed_filter
    self.name = 'time_operate, self.raw_speed, self.raw_speed_filter'

    if self.time_operate <= 40 :

        print("Recording..",self.time_operate , "Sec. || R =
",self.raw_speed , "|| RF =",self.raw_speed_filter)

        name = 'test_f_100.csv'      #change name csv

        if self.csv_name == 0:

            for i in range(len(self.name.split(','))):
                self.Data.append(self.name.split(', self.')[i])
            # print(self.Data)

            with open(name, 'a',newline='') as f:
                writer = csv.writer(f,delimiter=",")
                writer.writerow(self.Data)
            self.csv_name = 1

        if self.csv_name == 1:

```

```
    self.Data = self.value
    # print(self.Data)

    with open(name, 'a',newline='') as f:
        writer = csv.writer(f,delimiter=",")
        writer.writerow(self.Data)
        self.csv_name = 1

    if self.time_operate > 40 :
        print("Stop...")

if __name__ == '__main__':
    rospy.init_node('Velocity_Calculation',anonymous=True)
    try:
        my_subs = main()
    except rospy.ROSInterruptException:
        pass
```

## **ภาคผนวก ค**

โค้ดแสดงผลการทำงานของระบบควบคุมความเร็วผ่าน Robot Operating System

```
#!/usr/bin/env python3
import rospy #Import python for
Ros
from std_msgs.msg import Float64 ,Float64MultiArray #Import Standard
Message type
import csv
import time
class main (object):

    def __init__(self):

        #Direction_F_callback Function
        self.forward = 0.0

        #Direction_B_callback Function
        self.backward = 0.0

        #Direction_P_callback Function
        self.padel = 0.0

        #velocity_GNSS_callback Function
        self.Speed_time = 0.0
        self.Speed_velocity = 0.0
        self.Speed_Setpoint = 0.0

        #Speederror_callback Function
        self.Speed_error = 0.0

        #Speederror_callback Function
        self.Speed_adjust = 0.0

        #CreateCSV Function
        self.value = ()
        self.name = 'Data Column Name'
        self.csv_name = 0
        self.Data = []

        rospy.Subscriber("direction_F",Float64,self.Direction_F_callback)
        rospy.Subscriber("direction_B",Float64,self.Direction_B_callback)
        rospy.Subscriber("direction_P",Float64,self.Direction_P_callback)
        rospy.Subscriber("Speedadjust",Float64,self.Speedadjust_callback)
        rospy.Subscriber("Speederror",Float64,self.Speederror_callback)
        rospy.Subscriber("Velocity",Float64MultiArray,self.velocity_GNSS_cal
lback)
```

```

        rospy.spin()

def Direction_F_callback(self,msg):
    if msg.data >= 1000:
        self.forward = 1
    if msg.data < 1000:
        self.forward = 0

def Direction_B_callback(self,msg):
    if msg.data >= 1000:
        self.backward = 1
    if msg.data < 1000:
        self.backward = 0

def Direction_P_callback(self,msg):
    if msg.data >= 1000:
        self.padel = 1
    if msg.data <=1000:
        self.padel= 0

def velocity_GNSS_callback(self,msg):
    self.Speed_time = msg.data[0]
    self.Speed_velocity = msg.data[1]
    self.Speed_Setpoint = msg.data[2]

    if self.Speed_time > 0.0:
        # self.print()
        self.CreateCSV()

def Speederror_callback(self,msg):
    self.Speed_error = msg.data

def Speedadjust_callback(self,msg):
    self.Speed_adjust = msg.data

def print(self):
    print("Recording..", self.Speed_time
,"||",self.Speed_Setpoint,"||",self.Speed_velocity,"||",self.Speed_adjust,"|"
|",self.Speed_error)

```

```

def CreateCSV(self):

    self.value = self.Speed_time ,self.Speed_velocity
    ,self.Speed_Setpoint ,self.Speed_adjust ,self.Speed_error

    self.name =
    'Speed_time,self.Speed_velocity,self.Speed_Setpoint,self.Speed_adjust,self.S
    peed_error'

    if self.Speed_time <= 1000 :

        print("Recording..", self.Speed_time
        ,|||,self.Speed_Setpoint,|||,self.Speed_velocity,|||,self.Speed_adjust,|||
        ,self.Speed_error)

        name = 'speed_fullway_1.csv'      #change name csv

        if self.csv_name == 0:

            for i in range(len(self.name.split(','))):
                self.Data.append(self.name.split(',')[self.'][i])
            # print(self.Data)

            with open(name, 'a',newline='') as f:
                writer = csv.writer(f,delimiter=",")
                writer.writerow(self.Data)
                self.csv_name = 1

        if self.csv_name == 1:

            self.Data = self.value
            # print(self.Data)

            with open(name, 'a',newline='') as f:
                writer = csv.writer(f,delimiter=",")
                writer.writerow(self.Data)
                self.csv_name = 1

    if self.Speed_time > 1000 :
        print("Stop...")

if __name__ == '__main__':
    rospy.init_node('Speed_Control',anonymous=True)      #Create ROS Node
    "Speed Control"
    try:

```

```
my_sub = main()           #When This Code Run Do The Talker Function

except rospy.ROSInterruptException:
    pass
#When This Code get Interrupt or Ctrl+c Stop Do The Function
```

**ภาคผนวก ง**

โค๊ดควบคุมความเร็วให้คงที่

```
#include <ros.h>
#include <std_msgs/Int64.h>
#include <std_msgs/String.h>
#include <std_msgs/Float64.h>
#include <std_msgs/Float64MultiArray.h>

//-----LIBRARY-----//
#include <SPI.h>
#include <digitalWriteFast.h>
//-----PIN--ARDUINO-----//
#define FORWARDIN      A1
#define BACKWARDIN     A2
#define SAFTYIN        A3
#define SPEEDIN        A0

#define FORWARDOUT     38
#define BACKWARDOUT    37
#define SAFTYOUT        36

#define CONTROLSPEED   53

//-----PARAMETER--TIME-----//
long previousMillis = 0;
long currentMillis = 0;

int interval = 200;

//-----PARAMETER--FUNCTION-----//
int value;
byte address = 0x11;
//-----PARAMETER-PID-----//

long currenttime = 0;

float Kp = 0.0;

float Ki = 0.0;

float Kd = 0.0;

float derivative = 0;
float prevadjust;
float sigmaerror=0, dT, prerror=0;
float prev=0;
float preave=0;
```

```

float adjust1;
//-----//  

//-----ROS--TOPIC-----//  

ros::NodeHandle nh;  

std_msgs::Float64 FORWARD, BACKWARD, PADEL, SPEED_adjust, SPEED_error;  

ros::Publisher direc1("direction_F",&FORWARD);  

ros::Publisher direc2("direction_B",&BACKWARD);  

ros::Publisher direc3("direction_P",&PADEL);  

ros::Publisher Speedadjust("Speedadjust",&SPEED_adjust);  

ros::Publisher Speederror("Speederror",&SPEED_error);  

//-----ROS---SUBSCRIBER-----//  

void messageCb( const std_msgs::Float64MultiArray& msg)  

{  

    float Time_oprate = msg.data[0];  

    float velocity_GNSS = msg.data[1]; // sub data  

    float Speed_direction = msg.data[2];  

    if (Speed_direction == 1.0)  

    {  

        Kp = 0.2;  

        Ki = 0.02;  

        Kd = 1.0;  

        SpeedControl(velocity_GNSS,Speed_direction ,Time_oprate,Kp,Ki,Kd);  

    }  

    if (Speed_direction == 3.0)  

    {  

        Kp = 0.5;  

        Ki = 0.05;  

        Kd = 1.0;  

        SpeedControl(velocity_GNSS,Speed_direction ,Time_oprate,Kp,Ki,Kd);  

    }  

}  

ros::Subscriber<std_msgs::Float64MultiArray> subspeed("Velocity",  

&messageCb);  

void setup()  

{

```

```

//----CREATE--ROS--TOPIC-----//
nh.initNode();
nh.advertise(direc1);           //ROS PUBLISH
nh.advertise(direc2);           //ROS PUBLISH
nh.advertise(direc3);           //ROS PUBLISH

nh.advertise(Speedadjust);      //ROS PUBLISH
nh.advertise(Speederror);       //ROS PUBLISH

nh.subscribe(subspeed);          //ROS SUBSCRIBER

//-----PINMODE-----//
pinMode (FORWARDIN, INPUT);
pinMode (BACKWARDIN, INPUT);
pinMode (SAFTYIN, INPUT);

pinMode (FORWARDOUT, OUTPUT);
pinMode (BACKWARDOUT, OUTPUT);
pinMode (SAFTYOUT, OUTPUT);

pinMode (53,OUTPUT);

SPI.begin();
Serial.begin(57600);

}

void loop()
{

//-----publish--direction--forward --backward--padel----- //

    float state1 = analogRead(FORWARDIN);
    float state2 = analogRead(BACKWARDIN);
    float state3 = analogRead(SAFTYIN);
    float state4 = analogRead(SPEEDIN);
    FORWARD.data = state1;
    direc1.publish(&FORWARD);
    BACKWARD.data = state2;
    direc2.publish(&BACKWARD);
    PADEL.data = state3;
    direc3.publish(&PADEL);
    nh.spinOnce();
    delay(1000);
}

```

```

//-----//  

float adjust = adjust1;  

runSpeed( state1, state2, state3, adjust ); //call runspeed function  

close loop  

}  

void runSpeed (float state1,float state2,float state3,float value)  

{  

if ((state1 > 1000) & (state2 < 1000))                                //Forward  

{  

    Serial.print(" || Forward...");  

    digitalWrite(FORWARDOUT,HIGH);  

    digitalWrite(BACKWARDOUT,LOW);  

    digitalPotWrite(value);  

}  

if ((state1 < 1000) & (state2 > 1000))                                //Backwad  

{  

    Serial.print(" || Backward...");  

    digitalWrite(FORWARDOUT,LOW);  

    digitalWrite(BACKWARDOUT,HIGH);  

    digitalPotWrite(value);  

}  

if ((state1 < 1000) & (state2 < 1000))                                //None  

{  

    value = 0;  

    Serial.print(" || None...");  

    digitalWrite(FORWARDOUT,LOW);  

    digitalWrite(BACKWARDOUT,LOW);  

    digitalPotWrite(value);  

}  

if ((state1 > 1000) & (state2 > 1000))                                //ERROR  

{  

    value = 0;  

    Serial.print(" || ERROR...");  

    digitalWrite(FORWARDOUT,LOW);
}

```

```

digitalWrite(BACKWARDOUT,LOW);
digitalPotWrite(value);

}

}

float SpeedControl(float velocity_GNSS,float set,float Time_ostrate,float
Kp,float Ki,float Kd)
{
    float time_now = Time_ostrate;
    dT = time_now - prev;
    prev = time_now;
    float error = set-velocity_GNSS ;
    float Uo = 20;
    if (set == 1.0)
    {
        sigmaerror = sigmaerror + (error*dT);

        float adjust1km = Uo+(error*Kp)+(Ki*sigmaerror)+(Kd*((error-
preerror)));
        adjust1 = adjust1km;

        if (adjust1 > 25 )
        {
            adjust1 = 20;
        }
        SPEED_Adjust.data = adjust1km; // and pud
        Speedadjust.publish(&SPEED_Adjust);
    }
    if (set == 3.0)
    {
        sigmaerror = sigmaerror + (error*dT);
//        float adjust3km = (error*Kp)+(Ki*sigmaerror)+(Kd*((error-
preerror)/dT));
        float adjust3km = Uo+(error*Kp)+(Ki*sigmaerror)+(Kd*((error-
preerror)));
        adjust1 = adjust3km;

        if (adjust1 > 30 )
        {
            adjust1 = 25;
        }
        SPEED_Adjust.data = adjust3km; // and pud
        Speedadjust.publish(&SPEED_Adjust);
    }
}

```

```
    }

SPEED_error.data = error; // and pud
Speederror.publish(&SPEED_error);

SPEED_adjust.data = adjust1; // and pud
Speedadjust.publish(&SPEED_adjust);

    prerror = error;
    prevadjust = adjust1;
    return adjust1;

}

int digitalPotWrite(float value)
{
    Serial.print(" || Value = ");
    Serial.println(value);
    digitalWrite(53, LOW);
    SPI.transfer(address);
    SPI.transfer(value);
    digitalWrite(53, HIGH);
}
```

## **ภาคผนวก จ**

โภคการควบคุมการติดตามเส้นทางอ้างอิงผ่าน Robot Operating System

```
#!/usr/bin/env python3
import rospy
from std_msgs.msg import Float64MultiArray,Float64
import csv
import math
import can
import os
import time

class main(object):

    def __init__(self):

        # CreateCSV Function
        self.value = []
        self.name = 'Data Column Name'
        self.column = 0
        self.csv_name = 0
        self.Data = []

        #Realtime XY callback Function
        self.time = 0.0
        self.x_east = 0.0
        self.y_north = 0.0
        self.xy = 0.0
        self.start = 0
        self.time_start = 0.0
        self.time_operate = 0.0

        #Select_Track Function
        self.Track = None
        self.linear = 0
        self.curve = 0

        self.reference = 0.0
        self.max_right = 0.0
        self.max_left = 0.0

        self.speed = 0
        self.a,self.b,self.c      = 1,1,1
        self.aL,self.bL,self.cL   = 0,0,0
        self.aR,self.bR,self.cR   = 0,0,0
```

```

        #linear S-N
self.SN1_start = 1509522.50
self.SN1_stop = 1509583.749
self.SN1_maxright = 661489.88
self.SN1_maxleft = 661484.173410818
self.SN1_ref = 661487.31726504
self.aSN1,self.bSN1,self.cSN1 = -78.53077, -1, 53456668.62524

self.SN2_start = 1509583.7491
self.SN2_stop = 1509685.06490994
self.SN2_maxright = 661489.88
self.SN2_maxleft = 661484.173410818
self.SN2_ref = 661486.8775
self.aSN2,self.bSN2,self.cSN2 = -272.2586713752698, -1,
181605145.8698626

        #curve1 E-W
self.curve1_s1_start = 1509685.065
self.curve1_s1_stop = 1509694.17
self.curve1_s1_maxright = 661488.735
self.curve1_s1_maxleft = 661482.46
self.aL1,self.bL1,self.cL1 = -2.0882352939222786, -1,
2891024.6175178257
self.a1,self.b1,self.c1 = -3.7748691098637175, -1, 4006714.14565129
self.aR1,self.bR1,self.cR1 = -4.341614906864185, -1,
4381619.400890658

self.curve1_s2_start = 661489.72
self.curve1_s2_stop = 661480.88
self.curve1_s2_maxright = 1509700.31
self.curve1_s2_maxleft = 1509694.17
self.aL2,self.bL2,self.cL2 = -0.5649350650366756, -1,
1883388.2652620187
self.a2,self.b2,self.c2 = -0.6870229007823532, -1,
1964150.5048217247
self.aR2,self.bR2,self.cR2 = -0.7342465753199211, -1,
1995391.7124782377

self.curve1_s3_start = 661480.90
self.curve1_s3_stop = 661478.34
self.curve1_s3_maxright = 1509700.31
self.curve1_s3_maxleft = 1509694.17
self.aL3,self.bL3,self.cL3 = -0.8039215686310316, -1,
2041473.2727474666

```

```

    self.a3,self.b3,self.c3 = -0.3460559796060786, -1 ,
1738606.9283974625
    self.aR3,self.bR3,self.cR3 = -0.10139165010069193, -1,
1576768.6405376315

        #linear E-W
    self.EW_start = 661478.34
    self.EW_stop   = 661433.35
    self.EW_maxright = 1509703.9
    self.EW_maxleft = 1509696.5
    self.EW_ref = 1509700
    self.aEW,self.bEW,self.cEW = -0.0666666666666667, -1,
1553796.9466666665

        #curve 2
    self.curve2_s1_start = 661433.35
    self.curve2_s1_stop = 661418.32
    self.curve2_s1_maxright = 1509703.9
    self.curve2_s1_maxleft = 1509699.09
    self.aL4,self.bL4,self.cL4 = 0.07692307693214072 , -1 ,
1458819.6015324665
    self.a4,self.b4,self.c4 = 0.1064537591388774, -1, 1439289.3234726791
    self.aR4,self.bR4,self.cR4 = 0.16936005170547253, -1,
1397683.513644276

        self.curve2_s2_start = 661418.32
        self.curve2_s2_stop = 661405.27
        self.curve2_s2_maxright = 1509703.27
        self.curve2_s2_maxleft = 1509689.86
        self.aL5,self.bL5,self.cL5 = 0.542297417618975, -1,
1151012.3892156614
        self.a5,self.b5,self.c5 = 0.4942528735622956, -1, 1182791.8847132542
        self.aR5,self.bR5,self.cR5 = 0.45174537988360325, -1,
1210908.8085375926

        self.curve2_s3_start = 661405.27
        self.curve2_s3_stop = 661396.73
        self.curve2_s3_maxright = 1509696.68
        self.curve2_s3_maxleft = 1509684
        self.aL6,self.bL6,self.cL6 = 0.5973741794515571, -1,
1114584.2168792302
        self.a6,self.b6,self.c6 = 0.6194379391117303, -1, 1099993.8226335626
        self.aR6,self.bR6,self.cR6 = 0.5845588235293593, -1,
1123065.5626103287

```

```

    self.curve2_s4_start = 661396.73
    self.curve2_s4_stop = 661387.65
    self.curve2_s4_maxright = 1509691
    self.curve2_s4_maxleft = 150966.5
    self.aL7,self.bL7,self.cL7 = 1.243577545178, -1, 687186.4998019538
    self.a7,self.b7,self.c7 = 1.1019417475893303, -1, 780867.3814939316
    self.aR7,self.bR7,self.cR7 = 1.086913086900911, -1,
790810.9093287324

    self.curve2_s5_start = 661387.65
    self.curve2_s5_stop = 661376.99
    self.curve2_s5_maxright = 1509682.03
    self.curve2_s5_maxleft = 1509666.5
    self.aL8,self.bL8,self.cL8 = 2.519774011317643, -1, -
156874.08187644905
    self.a8,self.b8,self.c8 = 1.918356643326645, -1, 240901.64820340672
    self.aR8,self.bR8,self.cR8 = 1.5237483954085216, -1,
501894.5451278954

    self.curve2_s6_start = 1509666.5
    self.curve2_s6_stop = 1509653.56
    self.curve2_s6_maxright = 661376.99
    self.curve2_s6_maxleft = 661387.65
    self.aL9,self.bL9,self.cL9 = 6.420118343324884, -1, -
2736499.8465946023
    self.a9,self.b9,self.c9 = 8.391034482958728, -1, -4040002.6169737265
    self.aR9,self.bR9,self.cR9 = 42.4999999090505, -1, -
26598868.50898481

    #linear N-S
    self.NS_start = 1509653.566
    self.NS_stop = 1509560.129
    self.NS_maxright = 661375.26
    self.NS_maxleft = 661383.26
    self.NS_ref = 661379.26
    self.aNS,self.bNS,self.cNS = 7786.41667453521,-1, -
5148264891.408258

    #Curve3
    self.curve3_s1_start = 1509560.13
    self.curve3_s1_stop = 1509550.53
    self.curve3_s1_maxright = 661376.99
    self.curve3_s1_maxleft = 661384.5
    self.aL10,self.bL10,self.cL10 = -9.391025640947507, -1,
7720619.394691913

```

```

    self.a10,self.b10,self.c10 = -6.130268199177321, -1,
5563992.33839182
    self.aR10,self.bR10,self.cR10 = -7.999999999641799, -1,
6800576.049763095

    self.curve3_s2_start = 1509550.53
    self.curve3_s2_stop = 1509542.1
    self.curve3_s2_maxright = 661378.29
    self.curve3_s2_maxleft = 661388.00
    self.aL11,self.bL11,self.cL11 = -3.090015128686663, -1,
3553235.915901557
    self.a11,self.b11,self.c11 = -4.058365758626549, -1,
4193675.803300349
    self.aR11,self.bR11,self.cR11 = -3.9306545689721237, -1,
4109199.3274074704

    self.curve3_s3_start = 1509542.1
    self.curve3_s3_stop = 1509529.5
    self.curve3_s3_maxright = 661379.5
    self.curve3_s3_maxleft = 661398.2
    self.aL12,self.bL12,self.cL12 = -1.2935779816383115, -1,
2365097.5369638363
    self.a12,self.b12,self.c12 = -1.0589318600381945, -1,
2209900.0433710665
    self.aR12,self.bR12,self.cR12 = -1.0545749279777836, -1,
2207013.8169610477

    self.curve3_s4_start = 1509529.5
    self.curve3_s4_stop = 1509519.2
    self.curve3_s4_maxright = 661387.00
    self.curve3_s4_maxleft = 661412.4
    self.aL13,self.bL13,self.cL13 = -0.7512877115516731, -1,
2006429.024524765
    self.a13,self.b13,self.c13 = -0.748349229653909, -1,
2004482.477485025
    self.aR13,self.bR13,self.cR13 = -0.6805157593141369, -1,
1959613.8957318598

    self.curve3_s5_start = 1509519.2
    self.curve3_s5_stop = 1509510.45
    self.curve3_s5_maxright = 661398.4
    self.curve3_s5_maxleft = 661417.61
    self.aL14,self.bL14,self.cL14 = -0.6045258620781848, -1,
1909359.3577216151

```

```

    self.a14,self.b14,self.c14 = -0.6115107913632912, -1,
1913976.4561127168
    self.aR14,self.bR14,self.cR14 = -0.5840707964522756, -1,
1895824.576189464

    self.curve3_s6_start = 661417.61
    self.curve3_s6_stop = 661423.72
    self.curve3_s6_maxright = 1509507.83
    self.curve3_s6_maxleft = 1509516.66
    self.aL15,self.bl15,self.cL15 = -0.37177280549719216, -1,
1755412.1382718496
    self.a15,self.b15,self.c15 = -0.3878887070187147, -1,
1766068.8715423085
    self.aR15,self.bR15,self.cR15 = -0.4087363494330117, -1,
1779855.460625758

    self.curve3_s7_start = 661423.72
    self.curve3_s7_stop = 661430.28
    self.curve3_s7_maxright = 1509506.53
    self.curve3_s7_maxleft = 1509512.5
    self.aL16,self.bl16,self.cL16 = -0.21768707483451122, -1,
1653496.0624520085
    self.a16,self.b16,self.c16 = -0.1753048780685827, -1,
1625460.8845862686
    self.aR16,self.bR16,self.cR16 = -0.1818181818241025, -1,
1629766.560913007

    self.curve3_s8_start = 661430.28
    self.curve3_s8_stop = 661436.79
    self.curve3_s8_maxright = 1509506.53
    self.curve3_s8_maxleft = 1509512.5
    self.aL17,self.bl17,self.cL17 = 0.0669781931358256, -1,
1465209.8089322394
    self.a17,self.b17,self.c17 = 0.07834101382620332, -1,
1457691.8112894504
    self.aR17,self.bR17,self.cR17 = 0.13444108759791248, -1,
1420583.1385751278

    #linear W-E
    self.WE_start = 661436.79
    self.WE_stop = 661466.07
    self.WE_maxright = 1509506.53
    self.WE_maxleft = 1509512.5
    self.WE_ref = 1509509.63

```

```

    self.aWE,self.bWE,self.cWE = 0.012978142080676918, -1,
1500925.219361993

        #Curve4
        self.curve4_s1_start = 661466.07
        self.curve4_s1_stop = 661472.01
        self.curve4_s1_maxright = 1509507.42
        self.curve4_s1_maxleft = 1509512.44
        self.aL18,self.bl18,self.cL18 = 0.12457912457628709, -1,
1427106.836062483
        self.a18,self.b18,self.c18 = 0.06734006732370224, -1,
1464966.6503138554
        self.aR18,self.bR18,self.cR18 = 0.08249158248917392, -1,
1454942.3971228052

        self.curve4_s2_start = 661472.01
        self.curve4_s2_stop = 661478.05
        self.curve4_s2_maxright = 1509508.27
        self.curve4_s2_maxleft = 1509513.50
        self.aL19,self.bl19,self.cL19 = 0.13051146384218998, -1,
1423182.7596842642
        self.a19,self.b19,self.c19 = 0.11920529800788404, -1,
1430659.251924076
        self.aR19,self.bR19,self.cR19 = 0.08137715180242404, -1,
1455679.5618291756

        self.curve4_s3_start = 661478.05
        self.curve4_s3_stop = 661481.50
        self.curve4_s3_maxright = 1509508.79
        self.curve4_s3_maxleft = 1509515.1
        self.aL20,self.bl20,self.cL20 = 0.44837758113998843, -1,
1212921.4178635087
        self.a20,self.b20,self.c20 = 0.33333333336977006, -1,
1289018.2566425644
        self.aR20,self.bR20,self.cR20 = 0.2534653465436919, -1,
1341846.938112833

        self.curve4_s4_start = 661481.5
        self.curve4_s4_stop = 661483.9
        self.curve4_s4_maxright = 1509509.5
        self.curve4_s4_maxleft = 1509517.5
        self.aL21,self.bl21,self.cL21 = 0.8590604026415655, -1,
941262.5056660265
        self.a21,self.b21,self.c21 = 0.5475285171122858, -1,
1147331.9317096907

```

```

    self.aR21, self.bR21, self.cR21 = 0.44818652848068885, -1,
1213042.0988970706

    self.curve4_s5_start = 661483.9
    self.curve4_s5_stop = 661489.37
    self.curve4_s5_maxright = 1509509.5
    self.curve4_s5_maxleft = 1509517.5
    self.aL22, self.bl22, self.cL22 = 1.1343283582504489, -1,
759177.5537038959
    self.a22, self.b22, self.c22 = 1.927083333267016, -1, 234777.196918868
    self.aR22, self.bR22, self.cR22 = 2.766990291320843, -1, -
320817.1646019409

    self.curve4_s6_start = 1509517.50
    self.curve4_s6_stop = 1509522.50
    self.curve4_s6_maxright = 661489.88
    self.curve4_s6_maxleft = 661483.9
    self.aL23, self.bl23, self.cL23 = 18.518518517240985, -1, -
10740184.351006784
    self.a23, self.b23, self.c23 = 6.172839505746995, -1, -
2573734.721940532
    self.aR23, self.bR23, self.cR23 = 9.80392156844842, -1, -
4975672.401842357

#cte_current Function
self.acceprtable_error = 0
self.distance = 0.0
self.error_track = 0.0
self.cte = 0.0

#pid angle function
self.cte_prev = 0.0
self.sum_error_cte = 0.0
self.error_cte_prev = 0.0
self.yaw_prev = 0.0
self.yaw_expect = 0.0
self.kp = 0.0
self.ki = 0.0
self.kd = 0.0
self.P = 0.0
self.I = 0.0
self.D = 0.0

#angle controlMotor function
self.yaw_control = 0.0

```

```

    self.yaw = 0.0

    #Control_sent_can function
    self.angle = 0.0
    self.elec_angle_dec = 0.0
    self.elec_angle_hex = 0.0
    self.DATA_Hh = 0.0
    self.DATA_Hl = 0.0
    self.DATA_Lh = 0.0
    self.DATA_Ll = 0.0
    self.msg_sent = None

    self.Steering_Angle_want =
    rospy.Publisher('Steering_Angle_Want',Float64,queue_size=10)
    self.CTE_Pub =
    rospy.Publisher('CTE',Float64,queue_size=10)
    self.Speed_Pub =
    rospy.Publisher('Speed_Setpoint',Float64,queue_size=10)

    rospy.Subscriber("Realtime_XY",Float64MultiArray,self.Realtime_XY_ca
llback)
    rospy.spin()

def Realtime_XY_callback(self,msg):

    self.time = msg.data[0]
    self.x_east = msg.data[1]
    self.y_north = msg.data[2]
    self.xy = msg.data

    if self.start == 0:
        self.time_start = self.time
        self.start = 1

    if self.start == 1:

        self.time_operate = round(self.time - self.time_start,2)

        self.Select_Track()

        self.cte_current()

        self.pid_angle()

        self.angle_controlMotor()

```

```

        self.Talker()

        self.CreateCSV()

        print(round(self.x_east,2),'||'
,round(self.y_north,2),'||' ,self.Track,'|| cte = ' ,round(self.cte,2),'||'
Yaw = ' ,round(self.yaw_control,2))

def Select_Track(self):

    #linear SN1
    if (self.x_east <= self.SN1_maxright)&(self.x_east >=
self.SN1_maxleft) & (self.y_north >= self.SN1_start)&(self.y_north <=
self.SN1_stop):

        self.Track = 'linear_SN1'
        self.linear = 1
        self.curve = 0
        self.speed = 3

        self.a = self.aSN1
        self.b = self.bSN1
        self.c = self.cSN1


        self.reference = self.SN1_ref
        self.max_right = self.SN1_maxright
        self.max_left = self.SN1_maxleft

        self.kp = 48
        self.ki = 0.17
        self.kd = 30

    #linear SN2
    if (self.x_east <= self.SN2_maxright)&(self.x_east >=
self.SN2_maxleft) & (self.y_north >= self.SN2_start)&(self.y_north <=
self.SN2_stop):

        self.Track = 'linear_SN2'
        self.linear = 1
        self.curve = 0
        self.speed = 3

        self.a = self.aSN2

```

```

    self.b = self.bSN2
    self.c = self.cSN2

    self.reference = self.SN2_ref
    self.max_right = self.SN2_maxright
    self.max_left = self.SN2_maxleft

    self.kp = 45
    self.ki = 0.02
    self.kd = 26

    #Curve1_set1
    if (self.y_north >= self.curve1_s1_start) & (self.y_north <=
self.curve1_s1_stop ) & (self.x_east <=
self.curve1_s1_maxright)&(self.x_east >= self.curve1_s1_maxleft):

        self.Track = 'Curve1_Set1'
        self.linear = 0
        self.curve = 1

        self.speed = 1
        self.a = self.a1
        self.b = self.b1
        self.c = self.c1

        self.aL = self.aL1
        self.bL = self.bL1
        self.cL = self.cL1

        self.aR = self.aR1
        self.bR = self.bR1
        self.cR = self.cR1

        self.kp = 50
        self.ki = 0.17
        self.kd = 38

    #Curve1_set2
    if (self.y_north >= self.curve1_s2_maxleft)&(self.y_north <=
self.curve1_s2_maxright) & (self.x_east <=
self.curve1_s2_start)&(self.x_east >= self.curve1_s2_stop):

        self.Track = 'Curve1_Set2'
        self.linear = 0
        self.curve = 1

```

```
    self.speed = 1
    self.a = self.a2
    self.b = self.b2
    self.c = self.c2

    self.aL = self.aL2
    self.bL = self.bL2
    self.cL = self.cL2

    self.aR = self.aR2
    self.bR = self.bR2
    self.cR = self.cR2

    self.kp = 50
    self.ki = 0.2
    self.kd = 38

    #Curve1_set3
    if (self.x_east < self.curve1_s3_start)&(self.x_east >
self.curve1_s3_stop) & (self.y_north >=
self.curve1_s3_maxleft)&(self.y_north <= self.curve1_s3_maxright):

        self.Track = 'curve1_set3'
        self.linear = 0
        self.curve = 1

        self.speed = 1
        self.a = self.a3
        self.b = self.b3
        self.c = self.c3

        self.aL = self.aL3
        self.bL = self.bL3
        self.cL = self.cL3

        self.aR = self.aR3
        self.bR = self.bR3
        self.cR = self.cR3

        self.kp = 52
        self.ki = 0.2
        self.kd = 38
```

```

#linear EW
if (self.x_east <= self.EW_start)&(self.x_east >= self.EW_stop) &
(self.y_north <= self.EW_maxright)&(self.y_north >= self.EW_maxleft):

    self.Track = 'linear_EW'
    self.linear = 2    #for linear E-W and W-E
    self.curve = 0
    self.speed = 3

    self.a = self.aEW
    self.b = self.bEW
    self.c = self.cEW

    self.reference = self.EW_ref
    self.max_right = self.EW_maxright
    self.max_left = self.EW_maxleft

    self.kp = 45
    self.ki = 0.02
    self.kd = 26

#Curve2_set1
if (self.x_east < self.curve2_s1_start)&(self.x_east >
self.curve2_s1_stop) & (self.y_north <
self.curve2_s1_maxright)&(self.y_north > self.curve2_s1_maxleft):

    self.Track = 'Curve2_set1'
    self.linear = 0
    self.curve = 1
    self.speed = 3
    self.a = self.a4
    self.b = self.b4
    self.c = self.c4

    self.aL = self.aL4
    self.bL = self.bL4
    self.cL = self.cL4

    self.aR = self.aR4
    self.bR = self.bR4
    self.cR = self.cR4

    self.kp = 48
    self.ki = 0.17
    self.kd = 38

```

```
#Curve2_set2
    if (self.x_east < self.curve2_s2_start)&(self.x_east >
self.curve2_s2_stop) & (self.y_north <
self.curve2_s2_maxright)&(self.y_north > self.curve2_s2_maxleft):

        self.Track = 'Curve2_set2'
        self.linear = 0
        self.curve = 1

        self.speed = 3
        self.a = self.a5
        self.b = self.b5
        self.c = self.c5

        self.aL = self.aL5
        self.bL = self.bL5
        self.cL = self.cL5

        self.aR = self.aR5
        self.bR = self.bR5
        self.cR = self.cR5

        self.kp = 50
        self.ki = 0.2
        self.kd = 38

#Curve2_set3
    if (self.x_east < self.curve2_s3_start)&(self.x_east >
self.curve2_s3_stop) & (self.y_north <
self.curve2_s3_maxright)&(self.y_north > self.curve2_s3_maxleft):

        self.Track = 'Curve2_set3'
        self.linear = 0
        self.curve = 1

        self.speed = 3
        self.a = self.a6
        self.b = self.b6
        self.c = self.c6

        self.aL = self.aL6
        self.bL = self.bL6
        self.cL = self.cL6

        self.aR = self.aR6
```

```
    self.bR = self.bR6
    self.cR = self.cR6

    self.kp = 52
    self.ki = 0.2
    self.kd = 38

    #Curve2_set4
    if (self.x_east < self.curve2_s4_start)&(self.x_east >
self.curve2_s4_stop) & (self.y_north <
self.curve2_s4_maxright)&(self.y_north > self.curve2_s4_maxleft):

        self.Track = 'Curve2_set4'
        self.linear = 0
        self.curve = 1

        self.speed = 3
        self.a = self.a7
        self.b = self.b7
        self.c = self.c7

        self.aL = self.aL7
        self.bL = self.bL7
        self.cL = self.cL7

        self.aR = self.aR7
        self.bR = self.bR7
        self.cR = self.cR7

        self.kp = 50
        self.ki = 0.2
        self.kd = 38

    #Curve2_set5
    if (self.x_east < self.curve2_s5_start)&(self.x_east >
self.curve2_s5_stop) & (self.y_north <
self.curve2_s5_maxright)&(self.y_north > self.curve2_s5_maxleft):

        self.Track = 'Curve2_set5'
        self.linear = 0
        self.curve = 1

        self.speed = 1
        self.a = self.a8
        self.b = self.b8
```

```

    self.c = self.c8

    self.aL = self.aL8
    self.bL = self.bL8
    self.cL = self.cL8

    self.aR = self.aR8
    self.bR = self.bR8
    self.cR = self.cR8

    self.kp = 50
    self.ki = 0.2
    self.kd = 38

#Curve2_set6
if (self.x_east < self.curve2_s6_maxleft)&(self.x_east >
self.curve2_s6_maxright) & (self.y_north <
self.curve2_s6_start)&(self.y_north > self.curve2_s6_stop):

    self.Track = 'Curve2_set6'
    self.linear = 0
    self.curve = 1

    self.speed = 3
    self.a = self.a9
    self.b = self.b9
    self.c = self.c9

    self.aL = self.aL9
    self.bL = self.bL9
    self.cL = self.cL9

    self.aR = self.aR9
    self.bR = self.bR9
    self.cR = self.cR9

    self.kp = 50
    self.ki = 0.2
    self.kd = 38

#linear_NS
if (self.x_east < self.NS_maxleft)&(self.x_east > self.NS_maxright)
& (self.y_north < self.NS_start)&(self.y_north > self.NS_stop):

    self.Track = 'linear_NS'

```

```
    self.linear = 3
    self.curve = 0

    self.speed = 1
    self.a = self.aNS
    self.b = self.bNS
    self.c = self.cNS

    self.reference = self.NS_ref
    self.max_right = self.NS_maxright
    self.max_left = self.NS_maxleft

    self.kp = 40
    self.ki = 0.02
    self.kd = 23

#Curve3_set1
if (self.y_north < self.curve3_s1_start)&(self.y_north >
self.curve3_s1_stop) & (self.x_east > self.curve3_s1_maxright)&(self.x_east
< self.curve3_s1_maxleft):

    self.Track = 'Curve3_set1'
    self.linear = 0
    self.curve = 1

    self.speed = 3
    self.a = self.a10
    self.b = self.b10
    self.c = self.c10

    self.aL = self.aL10
    self.bL = self.bL10
    self.cL = self.cL10

    self.aR = self.aR10
    self.bR = self.bR10
    self.cR = self.cR10

    self.kp = 40
    self.ki = 0.02
    self.kd = 23
```

```
#Curve3_set2
if (self.y_north < self.curve3_s2_start)&(self.y_north >
self.curve3_s2_stop) & (self.x_east > self.curve3_s2_maxright)&(self.x_east
< self.curve3_s2_maxleft):

    self.Track = 'Curve3_set2'
    self.linear = 0
    self.curve = 1

    self.speed = 3
    self.a = self.a11
    self.b = self.b11
    self.c = self.c11

    self.aL = self.aL11
    self.bL = self.bL11
    self.cL = self.cL11

    self.aR = self.aR11
    self.bR = self.bR11
    self.cR = self.cR11

    self.kp = 50
    self.ki = 0.2
    self.kd = 38

#Curve3_set3
if (self.y_north < self.curve3_s3_start)&(self.y_north >
self.curve3_s3_stop) & (self.x_east > self.curve3_s3_maxright)&(self.x_east
< self.curve3_s3_maxleft):

    self.Track = 'Curve3_set3'
    self.linear = 0
    self.curve = 1

    self.speed = 3
    self.a = self.a12
    self.b = self.b12
    self.c = self.c12

    self.aL = self.aL12
    self.bL = self.bL12
    self.cL = self.cL12

    self.aR = self.aR12
```

```

    self.bR = self.bR12
    self.cR = self.cR12

    self.kp = 50
    self.ki = 0.2
    self.kd = 38

    #Curve3_set4
    if (self.y_north < self.curve3_s4_start)&(self.y_north >
    self.curve3_s4_stop) & (self.x_east > self.curve3_s4_maxright)&(self.x_east
    < self.curve3_s4_maxleft):

        self.Track = 'Curve3_set4'
        self.linear = 0
        self.curve = 1

        self.speed = 3
        self.a = self.a13
        self.b = self.b13
        self.c = self.c13

        self.aL = self.aL13
        self.bL = self.bL13
        self.cL = self.cL13

        self.aR = self.aR13
        self.bR = self.bR13
        self.cR = self.cR13

        self.kp = 45
        self.ki = 0.02
        self.kd = 26

    #Curve3_set5
    if (self.y_north > self.curve3_s5_start)&(self.y_north <
    self.curve3_s5_stop) & (self.x_east > self.curve3_s5_maxright)&(self.x_east
    < self.curve3_s5_maxleft):

        self.Track = 'Curve3_set5'
        self.linear = 0
        self.curve = 1

        self.speed = 3
        self.a = self.a14
        self.b = self.b14

```

```

    self.c = self.c14

    self.aL = self.aL14
    self.bL = self.bL14
    self.cL = self.cL14

    self.aR = self.aR14
    self.bR = self.bR14
    self.cR = self.cR14

    self.kp = 45
    self.ki = 0.02
    self.kd = 26

#Curve3_set6
    if (self.x_east > self.curve3_s6_start)&(self.x_east <
self.curve3_s6_stop) & (self.y_north < self.curve3_s6_maxleft)&(self.y_north
> self.curve3_s6_maxright):

        self.Track = 'Curve3_set6'
        self.linear = 0
        self.curve = 1

        self.speed = 3
        self.a = self.a15
        self.b = self.b15
        self.c = self.c15

        self.aL = self.aL15
        self.bL = self.bL15
        self.cL = self.cL15

        self.aR = self.aR15
        self.bR = self.bR15
        self.cR = self.cR15

        self.kp = 45
        self.ki = 0.02
        self.kd = 26

#Curve3_set7
    if (self.x_east > self.curve3_s7_start)&(self.x_east <
self.curve3_s7_stop) & (self.y_north < self.curve3_s7_maxleft)&(self.y_north
> self.curve3_s7_maxright):

```

```
    self.Track = 'Curve3_set7'
    self.linear = 0
    self.curve = 1

    self.speed = 3
    self.a = self.a16
    self.b = self.b16
    self.c = self.c16

    self.aL = self.aL16
    self.bL = self.bL16
    self.cL = self.cL16

    self.aR = self.aR16
    self.bR = self.bR16
    self.cR = self.cR16

    self.kp = 45
    self.ki = 0.02
    self.kd = 26

#Curve3_set8
    if (self.x_east > self.curve3_s8_start)&(self.x_east <
self.curve3_s8_stop) & (self.y_north < self.curve3_s8_maxleft)&(self.y_north
> self.curve3_s8_maxright):

        self.Track = 'Curve3_set8'
        self.linear = 0
        self.curve = 1
        self.speed = 3
        self.a = self.a17
        self.b = self.b17
        self.c = self.c17

        self.aL = self.aL17
        self.bL = self.bL17
        self.cL = self.cL17

        self.aR = self.aR17
        self.bR = self.bR17
        self.cR = self.cR17

        self.kp = 45
        self.ki = 0.02
        self.kd = 26
```

```

#linear_WE
if (self.x_east >= self.WE_start)&(self.x_east <= self.WE_stop) &
(self.y_north <= self.WE_maxleft)&(self.y_north >= self.WE_maxright):

    self.Track = 'linear_WE'
    self.linear = 4
    self.curve = 0

    self.speed = 3
    self.a = self.aWE
    self.b = self.bWE
    self.c = self.cWE

    self.reference = self.WE_ref
    self.max_right = self.WE_maxright
    self.max_left = self.WE_maxleft

    self.kp = 45
    self.ki = 0.02
    self.kd = 26

#Curve4_set1
if (self.x_east > self.curve4_s1_start)&(self.x_east <
self.curve4_s1_stop) & (self.y_north < self.curve4_s1_maxleft)&(self.y_north
> self.curve4_s1_maxright):

    self.Track = 'Curve4_set1'
    self.linear = 0
    self.curve = 1

    self.speed = 1
    self.a = self.a18
    self.b = self.b18
    self.c = self.c18

    self.aL = self.aL18
    self.bL = self.bL18
    self.cL = self.cL18

    self.aR = self.aR18
    self.bR = self.bR18
    self.cR = self.cR18

    self.kp = 45

```

```
    self.ki = 0.02
    self.kd = 26

    #Curve4_set2
    if (self.x_east > self.curve4_s2_start)&(self.x_east <
self.curve4_s2_stop) & (self.y_north < self.curve4_s2_maxleft)&(self.y_north
> self.curve4_s2_maxright):

        self.Track = 'Curve4_set2'
        self.linear = 0
        self.curve = 1

        self.speed = 1
        self.a = self.a19
        self.b = self.b19
        self.c = self.c19

        self.aL = self.aL19
        self.bL = self.bL19
        self.cL = self.cL19

        self.aR = self.aR19
        self.bR = self.bR19
        self.cR = self.cR19

        self.kp = 45
        self.ki = 0.02
        self.kd = 26

    #Curve4_set3
    if (self.x_east > self.curve4_s3_start)&(self.x_east <
self.curve4_s3_stop) & (self.y_north < self.curve4_s3_maxleft)&(self.y_north
> self.curve4_s3_maxright):

        self.Track = 'Curve4_set3'
        self.linear = 0
        self.curve = 1

        self.speed = 1
        self.a = self.a20
        self.b = self.b20
        self.c = self.c20

        self.aL = self.aL20
        self.bL = self.bL20
```

```

    self.cL = self.cL20

    self.aR = self.aR20
    self.bR = self.bR20
    self.cR = self.cR20

    self.kp = 52
    self.ki = 0.2
    self.kd = 38

#Curve4_set4
    if (self.x_east > self.curve4_s4_start)&(self.x_east <
self.curve4_s4_stop) & (self.y_north < self.curve4_s4_maxleft)&(self.y_north
> self.curve4_s4_maxright):

        self.Track = 'Curve4_set4'
        self.linear = 0
        self.curve = 1

        self.speed = 1
        self.a = self.a21
        self.b = self.b21
        self.c = self.c21

        self.aL = self.aL21
        self.bL = self.bL21
        self.cL = self.cL21

        self.aR = self.aR21
        self.bR = self.bR21
        self.cR = self.cR21

        self.kp = 52
        self.ki = 0.2
        self.kd = 38

#Curve4_set5
    if (self.x_east > self.curve4_s5_start)&(self.x_east <
self.curve4_s5_stop) & (self.y_north < self.curve4_s5_maxleft)&(self.y_north
> self.curve4_s5_maxright):

        self.Track = 'Curve4_set5'
        self.linear = 0
        self.curve = 1

```

```
    self.speed = 1
    self.a = self.a22
    self.b = self.b22
    self.c = self.c22

    self.aL = self.aL22
    self.bL = self.bL22
    self.cL = self.cL22

    self.aR = self.aR22
    self.bR = self.bR22
    self.cR = self.cR22

    self.kp = 52
    self.ki = 0.2
    self.kd = 38

#Curve4_set6
if (self.y_north > self.curve4_s6_start)&(self.y_north <
self.curve4_s6_stop) & (self.x_east > self.curve4_s6_maxleft)&(self.x_east <
self.curve4_s6_maxright):

    self.Track = 'Curve4_set6'
    self.linear = 0
    self.curve = 1

    self.speed = 1
    self.a = self.a23
    self.b = self.b23
    self.c = self.c23

    self.aL = self.aL23
    self.bL = self.bL23
    self.cL = self.cL23

    self.aR = self.aR23
    self.bR = self.bR23
    self.cR = self.cR23

    self.kp = 52
    self.ki = 0.2
    self.kd = 38
```

```

def cte_current(self):

    self.distance = abs(( self.a * self.x_east ) + ( self.b * 
self.y_north ) + self.c) / math.sqrt( self.a**2 + self.b**2 )

    if (self.linear == 1)&(self.curve == 0): #for linear S-N

        if(self.distance <= 0.15):
            self.cte = self.acceprtable_error

        if(self.distance > 0.15):
            if (self.x_east >= self.reference) & (self.x_east <=
self.max_right):
                self.cte = self.distance

            if (self.x_east < self.reference) & (self.x_east >=
self.max_left):
                self.cte = (-1)*self.distance

    if (self.linear == 2)&(self.curve == 0): #for linear E-W

        if(self.distance <= 0.15):
            self.cte = self.acceprtable_error

        if(self.distance > 0.15):
            if (self.y_north >= self.reference) & (self.y_north <=
self.max_right):
                self.cte = self.distance

            if (self.y_north < self.reference) & (self.y_north >=
self.max_left):
                self.cte = (-1)*self.distance

    if (self.linear == 3)&(self.curve == 0): #for linear N-S

        if(self.distance <= 0.15):
            self.cte = self.acceprtable_error

        if(self.distance > 0.15):
            if (self.x_east <= self.reference) & (self.x_east >=
self.max_right):
                self.cte = self.distance

            if (self.x_east > self.reference) & (self.x_east <=
self.max_left):
                self.cte = (-1)*self.distance

```

```

        self.cte = (-1)*self.distance

    if (self.linear == 4)&(self.curve == 0): #for linear W-E

        if(self.distance <= 0.15):
            self.cte = self.acceprtable_error

        if(self.distance > 0.15):
            if (self.y_north < self.reference) & (self.y_north >=
self.max_right):
                self.cte = self.distance

            if (self.y_north >= self.reference) & (self.y_north <=
self.max_left):
                self.cte = (-
1)*self.distance

    if (self.linear == 0)&(self.curve == 1): #for curve

        self.distance_L = abs(( self.aL * self.x_east ) + ( self.bL
* self.y_north ) + self.cL) / math.sqrt( self.aL**2 + self.bL**2 )
        self.distance_R = abs(( self.aR * self.x_east ) + ( self.bR
* self.y_north ) + self.cR) / math.sqrt( self.aR**2 + self.bR**2 )

        if(self.distance <= 0.15):
            self.cte = self.acceprtable_error

        if(self.distance > 0.15):
            if (self.distance_L >= self.distance_R):
                self.cte = self.distance

            if (self.distance_R > self.distance_L):
                self.cte = (-1)*self.distance

def pid_angle(self):

    self.yaw_prev = self.yaw_expect
    self.sum_error_cte += self.cte

    self.P = self.kp * self.cte
    self.I = self.ki * self.sum_error_cte
    self.D = self.kd * ((self.cte - self.cte_prev))

    self.yaw_expect = (-1)*(self.P + self.I + self.D)

```

```

def angle_controlMotor(self):

    if self.cte > 0.15 :
        self.yaw = abs(self.yaw_control - self.yaw_prev)

        if self.yaw >= 0.25:
            self.yaw_control = abs(self.yaw_expect)

    if self.cte < -0.15 :
        self.yaw = abs(self.yaw_control - self.yaw_prev)

        if self.yaw >= 0.25:
            self.yaw_control = -1*(self.yaw_expect)

    if self.cte == 0 :
        self.yaw_control = 0

def Talker(self):

    Angle_want = Float64()
    Angle_want.data = self.yaw_control

    self.Steering_Angle_want.publish(Angle_want)
    rospy.loginfo('Publishing Angle Want : %s',Angle_want.data)

    Speed_SP = Float64()
    Speed_SP.data = self.speed
    self.Speed_Pub.publish(Speed_SP)
    rospy.loginfo('Publishing Speed SetPoint : %s',Speed_SP.data)

def CreateCSV(self):

    self.value = self.time_operate , self.Track , self.x_east ,
    self.y_north , self.cte , self.yaw_control
    self.name =
    'time_operate,self.Track,self.x_east,self.y_north,self.cte,self.yaw_control'

    if self.time_operate <= 1000 :

        name = 'fullway_4.csv'      #change name csv

        if self.csv_name == 0:

            for i in range(len(self.name.split(','))):
```

```
        self.Data.append(self.name.split(',')[self.'][i])
# print(self.Data)

    with open(name, 'a',newline='') as f:
        writer = csv.writer(f,delimiter=",")
        writer.writerow(self.Data)
        self.csv_name = 1

if self.csv_name == 1:

    self.Data = self.value
# print(self.Data)

    with open(name, 'a',newline='') as f:
        writer = csv.writer(f,delimiter=",")
        writer.writerow(self.Data)
        self.csv_name = 1

if self.time_operate > 1000 :
    print("Stop...")

if __name__ == '__main__':
    rospy.init_node('Lateral_Control',anonymous=True)

try:
    my_subs = main()
except rospy.ROSInterruptException:
    pass
```

## ภาคผนวก ฉ

โค๊ดควบคุมการทำงานมอเตอร์พวงมาลัยผ่าน Robot Operating System

```

#!/usr/bin/env python3
import rospy
from std_msgs.msg import Float64MultiArray,Float64
import os
import time
import can

class main(object):

    def __init__(self):

        #Steering_Angle_Want_callback function
        self.yaw_control = 0.0

        #Control_sent_can function
        self.angle = 0.0
        self.elec_angle_dec = 0.0
        self.elec_angle_hex = 0.0
        self.DATA_Hh = 0.0
        self.DATA_Hl = 0.0
        self.DATA_Lh = 0.0
        self.DATA_Ll = 0.0
        self.msg_sent = None

        rospy.Subscriber("Steering_Angle_Want",Float64,self.Steering_Angle_Want_callback)
        rospy.spin()

    def Steering_Angle_Want_callback(self,msg):

        self.yaw_control = msg.data

        self.angle = self.yaw_control

        self.elec_angle_dec = self.angle*27

        self.elec_angle_hex = ('{:0>8X}'.format(int(self.elec_angle_dec) &
(2**32-1)))

        self.DATA_Hh = ((int(self.elec_angle_hex[0:2],16)))
        self.DATA_Hl = ((int(self.elec_angle_hex[2:4],16)))
        self.DATA_Lh = ((int(self.elec_angle_hex[4:6],16)))
        self.DATA_Ll = ((int(self.elec_angle_hex[6:8],16)))

```

```
    self.msg_sent = can.Message(arbitration_id=0x06000001,
data=[0x23,0x02,0x20,0x01,(self.DATA_Lh),(self.DATA_Ll),(self.DATA_Hh),(self
.DATA_Hl)])
        can0.send(self.msg_sent)
        print(self.yaw_control)

if __name__ == '__main__':

    rospy.init_node('Steering_Control',anonymous=True)

    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')

    print(os.name)

    can0 = can.interface.Bus(channel = 'can0', bustype = 'socketcan')
    msg_init = can.Message( arbitration_id=0x06000001, data=
[0x23,0x0d,0x20,0x01,0x00,0x00,0x00,0x00])

    can0.send(msg_init)
    print("Message sent on Enable state data: {}".format(msg_init))

try:
    my_subs = main()
except rospy.ROSInterruptException:
    pass
```

## ประวัติผู้จัดทำโครงการ

**ชื่อ - สกุล**

วัน เดือน ปีเกิด

นาย สุรยุทธ เดือนกร่าง

24 พฤษภาคม 2543

### ประวัติการศึกษา

ระดับมัธยมศึกษา

ประถมมัธยมศึกษาตอนต้น

โรงเรียนวัดสุทธิวราราม จังหวัดกรุงเทพมหานคร

ปีการศึกษา 2558

ประถมมัธยมศึกษาตอนปลาย

โรงเรียนวัดสุทธิวราราม จังหวัดกรุงเทพมหานคร

ปีการศึกษา 2561

ระดับปริญญาตรี

วิศวกรรมศาสตรบัณฑิต สาขาวิชาศิวกรรมระบบควบคุมและเครื่องมือวัด

มหาวิทยาลัยพระจอมเกล้าธนบุรี ปีการศึกษา 2565

## ประวัติผู้จัดทำโครงการ

**ชื่อ - สกุล**

วัน เดือน ปีเกิด

นางสาวสรวิษ์ วนิชผล

14 ธันวาคม 2543

### ประวัติการศึกษา

ระดับมัธยมศึกษา

ประถมมัธยมศึกษาตอนต้น

โรงเรียนนวนิตรากนูหิส สาขาวิชา พุทธศาสนา

ปีการศึกษา 2558

ประถมมัธยมศึกษาตอนปลาย

โรงเรียนนวนิตรากนูหิส สาขาวิชา พุทธศาสนา

ปีการศึกษา 2561

ระดับปริญญาตรี

วิศวกรรมศาสตรบัณฑิต สาขาวิชา วิศวกรรมระบบควบคุมและ

เครื่องมือวัด

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ปีการศึกษา 2565

## ประวัติผู้จัดทำโครงการ

**ชื่อ - สกุล**

วัน เดือน ปีเกิด

นางสาวชุติมนทน์ เกยการ

30 มิถุนายน 2543

### ประวัติการศึกษา

ระดับมัธยมศึกษา

ประโภค�ัธยมศึกษาตอนต้น

โรงเรียนวิทยาศาสตร์จุฬาภรณราชวิทยาลัย บุรีรัมย์  
ปีการศึกษา 2558

ประโภค�ัธยมศึกษาตอนปลาย

โรงเรียนอัสสัมชัญศึกษา จังหวัดกรุงเทพมหานคร  
ปีการศึกษา 2561

ระดับปริญญาตรี

วิศวกรรมศาสตรบัณฑิต สาขาวิชาศิวกรรมระบบควบคุมและ  
เครื่องมือวัด

มหาวิทยาเทคโนโลยีพระจอมเกล้าธนบุรี ปีการศึกษา 2565