ระบบนำร่องรถทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอสเอสราคาถูก
LOW-COST GNSS NAVIGATION SYSTEM FOR AGRICULTURAL VEHICLES

โดย

นายณัฏฐรงค์ นิลจันทร์

นายณัฏฐพล โพธิ์ไพจิตร

นายณัฐดนัย งามประเสริฐ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ระบบนำร่องรถทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอสเอสราคาถูก
LOW-COST GNSS NAVIGATION SYSTEM FOR AGRICULTURAL VEHICLES

โดย

นายณัฏฐณรงค์	นิลจันทร์	57010397
นายณัฏฐพล	โพธิ์ไพจิตร	57010400
นายณัฐดนัย	งามประเสริฐ	57010421

อาจารย์ที่ปรึกษา

ศ.ดร.พรชัย ทรัพย์นิธิ

ดร.เวธิต ภาคย์พิสุทธิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ผ่านการตรวจรูปเล่มแล้ว

(.....)

อาจารย์ที่ปรึกษา

21/5/61

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

ผ่านการตรวจชิ้นงานแล้ว

(.....)

กรรมการผู้ตรวจชิ้นงาน

21/5/61

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

ปริญญานิพนธ์ปีการศึกษา 2560
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ระบบนำร่องรถทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอสเอสราคาถูก
LOW-COST GNSS NAVIGATION SYSTEM FOR AGRICULTURAL VEHICLES

ผู้จัดทำ
1. นายณัฏฐณรงค์ นิลจันทร์ 57010397
2. นายณัฏฐพล โพธิ์ไพจิตร 57010400
3. นายณัฐดนัย งามประเสริฐ 57010421

..... **อาจารย์ที่ปรึกษา**
(ศ.ดร.พรชัย ทรัพย์นิธิ)

..... **อาจารย์ที่ปรึกษาร่วม**
(ดร.เวธิต ภาคย์พิสุทธิ์)

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความช่วยเหลือจากอาจารย์ที่ปรึกษาและผู้ที่เกี่ยวข้องในด้านต่างๆ ทางผู้จัดทำต้องขอขอบคุณ ศ.ดร.พรชัย ทรัพย์นิธิ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งท่านได้ให้คำแนะนำ ชี้แนะแนวคิดและวิธีการแก้ปัญหาในการทำงาน รวมถึงข้อคิดเห็นต่างๆที่เป็นประโยชน์ต่อการทำปริญญานิพนธ์เป็นอย่างยิ่ง ขอขอบคุณ ดร.เวธิต ภาคย์พิสุทธิ์ อาจารย์ที่ปรึกษาร่วมปริญญานิพนธ์ ท่านได้ให้ข้อมูลและคำแนะนำเพิ่มเติมจากอาจารย์ที่ปรึกษาหลัก ซึ่งช่วยเพิ่มเติมทักษะ ประสบการณ์ และเติมเต็มความรู้ให้กับผู้จัดทำ ขอขอบคุณ คุณสนิท เตียวฉิ้ม เจ้าหน้าที่ประจำห้องปฏิบัติการโทรคมนาคม ท่านได้ให้ความช่วยเหลือเกี่ยวกับการใช้เครื่องมือวัดทางวิศวกรรมในห้องปฏิบัติการ รวมถึงเอื้อเฟื้ออุปกรณ์ที่เป็นประโยชน์ต่อการทำปริญญานิพนธ์ ขอขอบคุณนักศึกษาปริญญาเอก คุณจิรภูมิ บุตรโท สำหรับข้อแนะนำและความช่วยเหลือทุกๆด้านตลอดการทำปริญญานิพนธ์

สุดท้ายนี้ ผู้จัดทำขอขอบคุณบิดามารดา ครอบครัว ที่ให้โอกาสทางศึกษา และขอขอบคุณผู้เกี่ยวข้องทุกท่านที่อาจมิได้กล่าวถึงข้างต้น ที่ให้การสนับสนุนและให้ความช่วยเหลือด้วยดีเสมอมา

นายณัฏฐณรงค์ นิลจันทร์
นายณัฏฐพล โพธิ์ไพจิตร
นายณัฐดนัย งามประเสริฐ
ผู้จัดทำ

ระบบนำร่องรถทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอสเอสราคาถูก
LOW-COST GNSS NAVIGATION SYSTEM FOR AGRICULTURAL VEHICLES

โดย นายณัฏฐณรงค์ นิลจันทร์ 57010397
นายณัฏฐพล โพธิ์ไพจิตร 57010400
นายณัฐดนัย งามประเสริฐ 57010421

อาจารย์ที่ปรึกษา ศ.ดร.พรชัย ทรัพย์นิธิ
อาจารย์ที่ปรึกษาร่วม ดร.เวธิต ภาคย์พิสุทธิ์

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการพัฒนาระบบนำร่องทางเกษตรกรรม รวมถึงการระบุพิกัดตำแหน่งบนพื้นโลกโดยใช้เทคโนโลยีจีเอ็นเอสเอสเพื่อทำให้เกิดความแม่นยำมากยิ่งขึ้น พร้อมทั้งแสดงผลการนำร่องผ่านทางส่วนประสานงานกราฟฟิกผู้ใช้ (GUI) เพื่ออำนวยความสะดวกให้กับผู้ขับขี่ในการบังคับการเดินรถทางเกษตรกรรมให้เดินทางไปในทิศทางที่ถูกต้อง อุปกรณ์ของระบบนำร่องที่พัฒนาขึ้นในปริญญานิพนธ์นี้ประกอบด้วย 2 ส่วน คือ อุปกรณ์สถานีฐาน และอุปกรณ์สถานีจลน์ สถานีฐานจะประกอบไปด้วยสายอากาศรับสัญญาณที่เชื่อมต่อกับอุปกรณ์ Raspberry pi และโมดูล 3G อุปกรณ์ชุดนี้ทำหน้าที่รับค่าพารามิเตอร์ต่างๆ ในการระบุพิกัดตำแหน่งจากสถานีฐานหลักที่ติดตั้งอยู่บริเวณดาดฟ้า อาคารเรียนรวม 12 ชั้น คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เพื่อใช้เป็นค่าอ้างอิงในการระบุพิกัดตำแหน่งให้กับอุปกรณ์สถานีจลน์ ซึ่งประกอบไปด้วย สายอากาศรับสัญญาณที่เชื่อมต่อกับอุปกรณ์ Raspberry pi ร่วมกับเซ็นเซอร์ IMU (Inertial Measurement Unit) เพื่อวัดความเร่งและความเอียงในทิศทางต่างๆ อุปกรณ์สถานีฐานและสถานีจลน์จะถ่ายโอนข้อมูลระหว่างกันผ่านโมดูล 3G และส่งข้อมูลมายังอุปกรณ์ Raspberry pi เพื่อประมวลผลค่าพิกัดตำแหน่งและแสดงผลพิกัดบนแผนที่ พร้อมกับเส้นทางการเดินรถทางเกษตรกรรมผ่านหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้ (GUI) ทางจอแสดงผล ทำให้ผู้ใช้งานสามารถควบคุมรถทางเกษตรกรรมให้เคลื่อนที่ไปในทิศทางที่ถูกต้องตามเส้นทางที่ได้ออกแบบไว้ ทั้งนี้ได้ทำการออกแบบวงจรควบคุมการคายประจุคงเหลือของแบตเตอรีในอุปกรณ์ภาคสนาม และตัดการคายประจุเมื่อมีระดับแรงดันต่ำกว่าแรงดันวิกฤติ เพื่อรักษาอายุการใช้งานของแบตเตอรี ผลการทดลองในเบื้องต้นทำให้ทราบว่า รูปแบบกลุ่มดาวเทียม (Constellation) และ

ความเสถียรในการรับสัญญาณจีเอ็นเอสเอสของเครื่องรับสัญญาณจีเอ็นเอสเอสมีอิทธิพลต่อความแม่นยำมากกว่าระยะ Baseline ที่ใช้ทดสอบ RTK และผลจากการจำลองด้วยโปรแกรม Matlab ในการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศในเบื้องต้นด้วย GY-85 สามารถใช้แก้ไขพิกัดของสายอากาศได้ ซึ่งมีความคลาดเคลื่อนไม่เกิน 2.7 มิลลิเมตร สำหรับการเอียงสายอากาศในมุมต่างๆ และในส่วนของส่วนประสานงานกราฟฟิกผู้ใช้ โปรแกรมที่สร้างขึ้นสามารถสร้างเส้นทางการเดินรถทางเกษตรกรรมได้โดยป้อนเพียงค่าพิกัดมุมของพื้นที่ที่สนใจจำนวน 4 มุม แล้วโปรแกรมจะสามารถคำนวณเส้นทางให้โดยอัตโนมัติ พร้อมกับแสดงลงบนแผนที่เพื่อใช้นำร่องรถทางเกษตรกรรมได้ ในส่วนของการทดสอบระบบ Auto Steering เมื่อทดสอบให้มอเตอร์ชนิดกระแสตรงที่เชื่อมเข้ากับชุดทดลองพวงมาลัยหมุนไปในทิศทางที่ต้องการ ชุดทดลองพวงมาลัยก็หมุนไปในองศาที่ต้องการเช่นกัน

ABSTRACT

This thesis develops the navigation system for Agricultural vehicles using the global navigation satellite system (GNSS) technology which to give precisioned position estimation. The Graphic User Interface (GUI) helps drivers to control agricultural vehicles on the correct path. This project is divided into 2 main parts, base station and rover station. Base station consists of antenna connected to Raspberry pi and 3G module. This kit receives the parameters obtaining the coordinates from the main base station at 12 floors classroom building, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, to estimate position of rover station more accurately. Rover station consists of antenna connected to Raspberry pi and IMU (Inertial Measurement Unit) sensor for acceleration and rotation measurement. Base and rover stations transfer data by 3G module and process the position in Raspberry pi to display it on the map with the moving path via Graphic User Interface (GUI) on the monitor. Drivers can control agricultural vehicle precisely by instruction on the monitor. Moreover, we design the discharge-controlled circuit to prevent the low voltage battery in the rover station and maintain the battery life. The results show that constellation of the GNSS satellites has more influence than the baseline on the accuracy in RTK testing. Furthermore, Matlab simulation of improving coordinates caused by the inclination of the antenna with GY-85 module with error not exceeding 2.7 mm. On the part of Graphic User Interface, the developed program can create the pathway for agricultural vehicles by entering just 4 corner points of interested area. Then it calculates the pathway automatically and displays on the map for agricultural vehicles's navigation. In auto steering experiments, when we commanded DC motor that combined with steering wheel kit

in defined direction, steering wheel kit will move in expected angles. This experiment will support automatic vehicle driving in the future.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันมีการใช้ประโยชน์จากการระบุพิกัดตำแหน่งโดยจีเอ็นเอสเอสกันอย่างแพร่หลาย ไม่ว่าจะเป็น การบอกตำแหน่งในแผนที่ การระบุตำแหน่งของสิ่งของ ใช้ในการนำทางไปยังตำแหน่งเป้าหมาย รวมไปถึงการใช้ติดตามพัสดุต่างๆ แต่ระบบจีเอ็นเอสเอสทั่วไปนั้นอาจจะใช้กับการทำงานประเภทที่ต้องการความแม่นยำในการระบุตำแหน่งสูงได้ไม่ดีเท่าที่ควร เช่น ระบบนำทางของรถทางเกษตรกรรม ทั้งการตัดหญ้าในสนามกีฬาต่างๆ ให้เป็นลวดลายแนวเส้นตรง หรือแปลงเพาะปลูกทางเกษตรกรรม การกำหนดเส้นทางการเคลื่อนของรถทางเกษตรกรรมที่ในพื้นที่ที่สนใจในแต่ละครั้งจะต้องทำการกำหนดจุดอ้างอิงของเส้นทางทุกครั้งก่อนที่จะเคลื่อนรถทางเกษตรกรรม ซึ่งถ้าหากต้องมาทำการกำหนดจุดอ้างอิงของพื้นที่ก่อนทำการตัดหญ้าทุกครั้งอาจทำให้เสียเวลา และสิ้นเปลืองทรัพยากรบุคคล อีกทั้งยังอาจก่อให้เกิดความผิดพลาดของเส้นทางการเคลื่อนที่ที่เกิดจากมนุษย์ ทำให้เส้นทางที่ต้องการตัดมีความบิดเบี้ยว ดูไม่สวยงาม

การระบุพิกัดตำแหน่งที่มีความแม่นยำสูงสามารถทำได้โดยใช้เครื่องรับสัญญาณดาวเทียมแบบ 2 ความถี่ แต่มีข้อจำกัดคือมีราคาสูง ทำให้ไม่เหมาะสมกับการใช้งานทางเกษตรกรรมในด้านของการลงทุน แต่ก็มีวิธีการระบุพิกัดตำแหน่งที่มีความแม่นยำสูงโดยใช้เครื่องรับสัญญาณแบบ 1 ความถี่มาคำนวณพิกัดตำแหน่งด้วยเทคนิค RTK (Real Time Kinematic) ซึ่งจะทำให้มีความแม่นยำใกล้เคียงกับเครื่องรับสัญญาณดาวเทียมแบบ 2 ความถี่ และมีราคาที่ต่ำกว่ามาก [1]

ในกรณีที่สายอากาศมีความเอียงเกิดขึ้นจากความไม่ราบเรียบของพื้นที่เกษตรกรรม จะทำให้เกิดความผิดพลาดในการระบุตำแหน่ง เนื่องจากการใช้เทคนิค RTK โดยทั่วไปไม่สามารถแก้ไขความเอียงด้วยตัวเองได้ จากการศึกษางานวิจัยที่เกี่ยวกับการระบุตำแหน่งด้วยเทคนิค RTK ในพื้นที่เกษตรกรรมพบว่า มีการใช้เทคนิค RTK [2] ร่วมกับเซนเซอร์ IMU (Inertial Measurement Unit) เพื่อตรวจจับความเอียงของสายอากาศแล้วแก้ไขพิกัดที่ผิดพลาดเนื่องจากความเอียงนั้น ทำให้สามารถระบุตำแหน่งรถทางเกษตรกรรมได้แม่นยำมากขึ้น แต่เซนเซอร์ที่ใช้มีราคาสูง และสมการที่ใช้ในการคำนวณมีความซับซ้อนมาก

ในโครงงานนี้จึงใช้เทคนิค RTK ในการระบุตำแหน่งร่วมกับเซนเซอร์ IMU สำหรับแก้ไขพิกัดที่ผิดพลาดที่มีราคาถูกลง และใช้สมการทางคณิตศาสตร์ที่เข้าใจง่าย รวมทั้งได้ความแม่นในระดับเซนติเมตร ทั้งนี้ เมื่อได้ค่าพิกัดตำแหน่งที่แม่นยำแล้ว จึงทำการเขียนโปรแกรมเพื่อแสดงผลพิกัดตำแหน่ง สร้างและแนะนำเส้นทางให้กับรถทางเกษตรกรรมโดยแสดงผลออกทางหน้าจอส่วนประสานงานกราฟฟิกผู้ใช้ (GUI) ทำให้สะดวกต่อการใช้งานในกลุ่มผู้ใช้งานที่หลากหลาย อีกทั้งยังได้ทำการทดลองระบบ Auto Steering เพื่อช่วยในการบังคับทิศทางของรถเกษตรกรรมให้เลี้ยวกลับมาในทิศทางของเส้นทางที่ถูกต้องในกรณีที่รถเคลื่อนที่ออกนอกเส้นทาง และรองรับระบบขับเคลื่อนอัตโนมัติในอนาคต

1.2 วัตถุประสงค์

1) เพื่อพัฒนาระบบนำร่องทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอสเอสความถี่เดียวให้มีความแม่นยำใกล้เคียงกับระบบจีเอ็นเอสเอสสองความถี่

2) ออกแบบส่วนประสานงานกราฟฟิกผู้ใช้ (GUI) เพื่ออำนวยความสะดวกในการบังคับการเดินรถทางเกษตรกรรม

3) เพื่อออกแบบวงจรสำหรับการแสดงผลค่าความจุคงเหลือของแบตเตอรี และวงจรป้องกันการคายประจุที่มากเกินไป

1.3 ขอบเขตของปริญญานิพนธ์

ออกแบบอุปกรณ์ในการนำร่องให้กับรถทางเกษตรกรรมด้วยโปรแกรม RTKRCV ที่ประมวลผลบนอุปกรณ์ Raspberry Pi เพื่อกำหนดค่าพิกัดตำแหน่ง หรือทิศทางในการเคลื่อนที่ให้กับรถทางเกษตรกรรม โดยนำอุปกรณ์ Raspberry Pi เชื่อมต่อกับอุปกรณ์ Ublox เพื่อรับสัญญาณจากสายอากาศ TW3710 M8T รวมถึงใช้เซนเซอร์ GY-85 ซึ่งเป็นเซนเซอร์วัดความเร่งและความเอียงมาประยุกต์ใช้เพื่อทำการปรับปรุงความผิดพลาดของพิกัดตำแหน่งเนื่องจากความเอียงของสายอากาศ เมื่อได้ค่าพิกัดตำแหน่งที่แม่นยำแล้วจึงทำการเขียนโปรแกรมเพื่อแสดงผลพิกัดตำแหน่ง สร้างและแนะนำเส้นทางให้กับรถทางเกษตรกรรมโดยแสดงผลออกทางหน้าจอส่วนประสานงานกราฟฟิกผู้ใช้ (GUI) หากการเคลื่อนที่ของรถทางเกษตรกรรมเกิดการผิดพลาด ระบบจะแนะนำเส้นทางให้ผู้ควบคุมรถบังคับรถให้กลับมาในทิศทางที่ถูกต้อง พร้อมออกแบบอุปกรณ์สถานีฐาน สถานีจลน์ วงจรตรวจสอบสถานะการใช้งานแบตเตอรี และวางระบบการเชื่อมต่อระหว่างอุปกรณ์แต่ละชนิดเพื่อให้ผู้ใช้งานสามารถใช้งานได้อย่างสะดวก

ในส่วนการทดลองระบบ Auto Steering จะใช้อุปกรณ์ Raspberry Pi ควบคุมมอเตอร์ชนิดกระแสตรงที่เชื่อมกับชุดทดลองพวงมาลัยรถ เมื่อมอเตอร์กระแสตรงหมุนไปตามทิศทาง หรือองศาที่ต้องการ ชุดทดลองพวงมาลัยก็จะหมุนไปในองศาที่ต้องการเช่นกัน โดยนำค่าที่ได้รับจากสถานีจลน์ มากำหนดทิศทางการเคลื่อนที่ของมอเตอร์

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

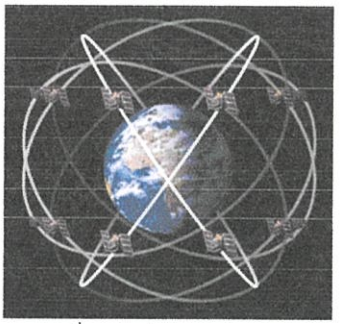
2.1 หลักการของจีเอ็นเอสเอส

จีเอ็นเอสเอส หรือ ระบบนำร่องและระบุตำแหน่งโดยใช้สัญญาณที่ถูกส่งมาจากดาวเทียมนำร่องเป็นหลัก ระบบนี้ได้พัฒนาขึ้นโดยกระทรวงกลาโหม (เฉพาะระบบจีพีเอส) แห่งสหรัฐอเมริกามาตั้งแต่ปี พ.ศ.2521 โดยอาศัยดาวเทียม ระบบคลื่นวิทยุนำร่องและรหัสที่ส่งมาจากดาวเทียม NAVSTAR จำนวน 24 ดวง โดยแบ่งวงโคจรออกเป็น 6 วงโคจร มี 4 ดวงต่อวงโคจร โดยโคจรอยู่รอบโลกวันละ 2 รอบ และอยู่เหนือพื้นโลกที่ความสูงประมาณ 20,000 กิโลเมตร ระบบจีเอ็นเอสเอสที่ใช้อยู่ในปัจจุบันได้แก่ จีพีเอส (ของสหรัฐอเมริกา) Galileo (ของสหภาพยุโรป) Glonass (ของรัสเซีย) QZSS (ของญี่ปุ่น) และBeiDou (ของจีน) เป็นต้น ในที่นี้จะกล่าวถึงองค์ประกอบการทำงาน และหลักการทำงานของจีเอ็นเอสเอส

2.1.1 องค์ประกอบการทำงานของจีเอ็นเอสเอส [1]

2.1.1.1 ส่วนอวกาศ

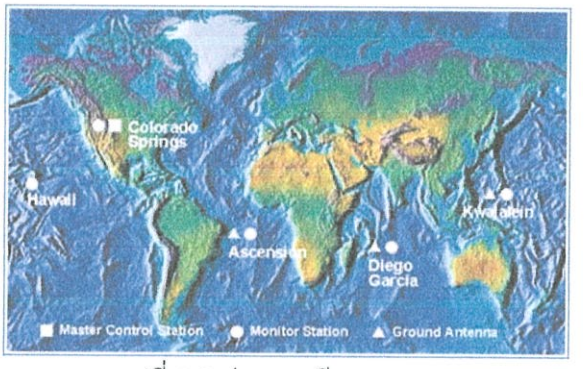
ส่วนอวกาศ (Space segment) แสดงดังรูปที่ 2.1 เป็นส่วนที่อยู่บนอวกาศ ประกอบด้วยดาวเทียมอย่างน้อย 24 ดวง แบ่งออกเป็น 6 วงโคจร วงโคจรละ 4 ดวง ทำหน้าที่ส่งสัญญาณคลื่นวิทยุจากอวกาศ มีวงโคจรสูงจากพื้นดินประมาณ 20,000 กิโลเมตรและอาจเปลี่ยนระดับความสูงได้ตามลักษณะการโคจร ในดาวเทียมจะมีนาฬิกาอะตอมมิคซึ่งมีความแม่นยำทางเวลาสูงอย่างน้อยสามเรือนใช้จับเวลาการเดินทางของสัญญาณวิทยุที่ถูกส่งไป เพื่อใช้ในระบุตำแหน่ง โดยสัญญาณที่ส่งจะเป็นสัญญาณที่มีกำลังงานต่ำและถูกส่งในแนวระดับสายตา ทำให้สัญญาณดังกล่าว สามารถทะลุผ่านชั้นบรรยากาศ เมฆ กระจก หรือพลาสติกได้ แต่ไม่สามารถทะลุผ่านโครงสร้างหนาๆ ของอาคาร และพื้นที่ที่มีภูเขาบดบังได้



รูปที่ 2.1 ส่วนอวกาศ [2]

2.1.1.2 ส่วนสถานีควบคุม

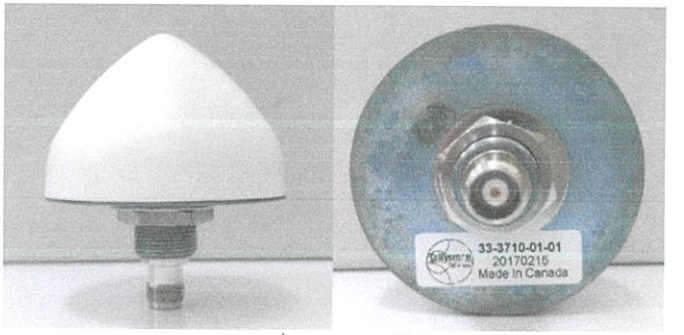
ส่วนสถานีควบคุม (Control segment) ประกอบด้วยสถานีภาคพื้นดินที่ควบคุมระบบที่กระจายอยู่ตามส่วนต่างๆ ของโลก เพื่อเฝ้าสังเกตและปรับปรุงการโคจรของดาวเทียมในระบบ (เฉพาะจีพีเอส) โดยมีสถานีควบคุมหลักตั้งอยู่ที่ฐานทัพอากาศในเมืองโคโลราโดปสริงส์ (Colorado Springs) รัฐโคโลราโดของประเทศสหรัฐอเมริกา เป็นสถานีติดตามดาวเทียม 5 แห่งดังรูปที่ 2.2



รูปที่ 2.2 ส่วนสถานีควบคุม [3]

2.1.1.3 ส่วนผู้ใช้

ส่วนผู้ใช้ (User segment) จะเป็นอุปกรณ์ภาครับ ประกอบด้วยเครื่องรับสัญญาณจีเอ็นเอสเอส ซึ่งมีหลายขนาด สามารถพกพาติดตัวหรือติดกับสิ่งของต่างๆ ได้ โดยในปริญญานิพนธ์นี้ จะใช้สายอากาศที่ทำงานในย่านความถี่ L1 ในการทดสอบของภาครับบนพื้นผิวโลก ดังรูปที่ 2.3



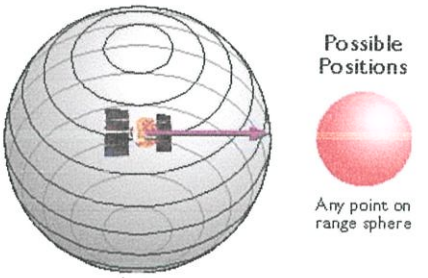
รูปที่ 2.3 ส่วนผู้ใช้

2.1.2 หลักการทำงานของจีเอ็นเอสเอส

เนื่องจากจีเอ็นเอสเอสใช้ดาวเทียมเป็นจุดอ้างอิงในการหาตำแหน่งของเครื่องรับสัญญาณของผู้ใช้งานบนผิวโลก การหาตำแหน่งของเครื่องรับดังกล่าว สามารถทำได้โดยใช้เทคนิคทางเรขาคณิตและช่วงเวลาในการเดินทางของสัญญาณที่มีความแม่นยำสูง มีเทคนิคและปัจจัยที่สำคัญต่อการทำงานของจีเอ็นเอสเอสดังต่อไปนี้

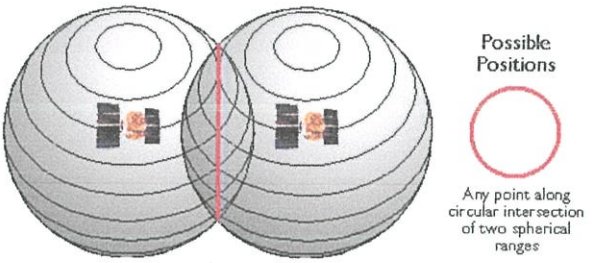
2.1.2.1 เทคนิค Trilateration

เป็นเทคนิคที่อาศัยคุณสมบัติทางเรขาคณิตของรูปวงกลม รูปสามเหลี่ยม และรูปทรงกลม เพื่อใช้วัดระยะห่างจากจุดอ้างอิงไปยังจุดที่ต้องการหาตำแหน่ง สามารถอธิบายอย่างเป็นขั้นตอนได้สังเกตจากรูปที่ 2.4



รูปที่ 2.4 มุมมองของพิกัดที่เป็นไปได้จากการใช้ดาวเทียมหนึ่งดวง [4]

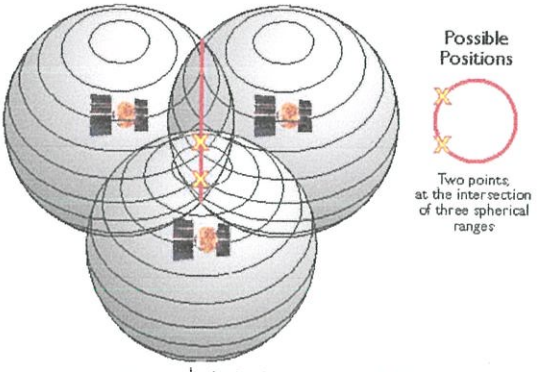
จากรูปที่ 2.4 การใช้ดาวเทียมดวงเดียวเพื่อหาตำแหน่งของเครื่องรับบนผิวโลก พบว่าตำแหน่งของเครื่องรับที่เป็นไปได้ คือจุดใดจุหนึ่งบนผิวทรงกลมสีแดง ซึ่งไม่ละเอียดมากพอ จึงต้องอาศัยดาวเทียมเพิ่มขึ้นมาอีกดวง เพื่อลดจำนวนพิกัดของเครื่องรับที่เป็นไปได้ ดังรูปที่ 2.5



รูปที่ 2.5 มุมมองของพิกัดที่เป็นไปได้จากการใช้ดาวเทียมสองดวง [5]

จากรูปที่ 2.5 หลังจากมีดาวเทียมอีกดวงมาช่วยในการหาพิกัดของเครื่องรับที่เป็นไปได้ พบว่าพิกัดของเครื่องรับจะอยู่บนเส้นรอบวงที่เกิดจากการตัดกันของทรงกลม ซึ่งมีความเจาะจงมากกว่า

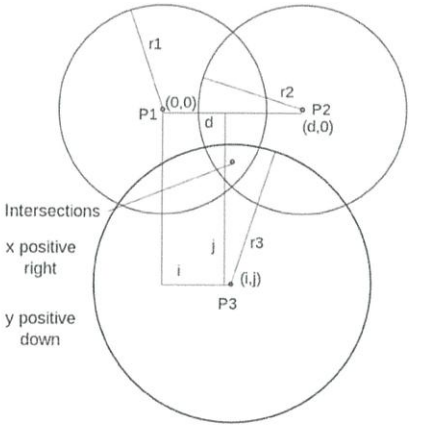
การใช้ดาวเทียมเพียงดวงเดียวขึ้นมาก เพื่อให้การหาพิกัดมีความเจาะจงมากยิ่งขึ้น จึงเพิ่มดาวเทียมเพื่อใช้หาพิกัดรวมเป็นสามดวง ดังรูปที่ 2.6



รูปที่ 2.6 มุมมองของพิกัดที่เป็นไปได้จากการใช้ดาวเทียมสามดวง [6]

จากรูปที่ 2.6 เมื่อเพิ่มดาวเทียมดวงที่สามเข้ามาแล้ว จะทำให้เกิดส่วนตัดระหว่างทรงกลมของดาวเทียมดวงที่สามกับเส้นรอบวงจากรูปที่ 2.5 ทำให้พิกัดของเครื่องรับที่เป็นไปได้ จะอยู่บนเส้นรอบวงระหว่างจุดกากบาททั้งสองจุด หากต้องการให้พิกัดเครื่องรับที่เป็นไปได้ มีความเจาะจงมากกว่านี้ สามารถทำได้โดยเพิ่มดาวเทียมเข้ามาอีกดวงได้ โดยปกติแล้ว จะใช้ดาวเทียมอย่างน้อย 4 ดวงในการหาพิกัดของเครื่องรับสัญญาณจีเอ็นเอสเอส

จากรูปที่ 2.4 ถึงรูปที่ 2.6 สามารถใช้เทคนิค Trilateration ช่วยประมาณพิกัดของเครื่องรับสัญญาณจีเอ็นเอสเอสได้ดังนี้



รูปที่ 2.7 เทคนิค Trilateration [7]

จากรูปที่ 2.7 สมมุติว่าพิกัดของดาวเทียมทั้งสามดวงเป็น P1 P2 และ P3 ตามลำดับ และทราบว่าระยะห่างของดาวเทียมแต่ละดวงกับพิกัดเครื่องรับสัญญาณจีเอ็นเอสเอส (จุดบนพื้นที่ที่วงกลมทั้งสามเหลื่อมกันอยู่) เป็น r1 r2 และ r3 ตามลำดับ จากคุณสมบัติของรูปวงกลม รูปสามเหลี่ยม และรูปทรงกลม สามารถสร้างสมการเพื่อหาพิกัดของเครื่องรับได้ดังสมการที่ 2.1 ถึงสมการที่ 2.8 [7]

$$d = \|P2 - P1\| \tag{2.1}$$

$$\hat{e}_x = \frac{P2 - P1}{\|P2 - P1\|} \tag{2.2}$$

$$i = \hat{e}_x \bullet (P3 - P1) \tag{2.3}$$

$$\hat{e}_y = \frac{P3 - P1 - i\hat{e}_x}{\|P3 - P1 - i\hat{e}_x\|} \tag{2.4}$$

$$j = \hat{e}_y \bullet (P3 - P1) \tag{2.5}$$

$$x = \frac{r_1^2 - r_2^2 + d^2}{2d} \tag{2.6}$$

$$y = \frac{r_1^2 - r_3^2 + i^2 + j^2}{2j} - \frac{i}{j}x \tag{2.7}$$

$$z = \pm\sqrt{r_1^2 - x^2 - y^2} \tag{2.8}$$

โดยที่

$P1$ เป็นพิกัดของดาวเทียมดวงที่ 1

$P2$ เป็นพิกัดของดาวเทียมดวงที่ 2

$P3$ เป็นพิกัดของดาวเทียมดวงที่ 3

d เป็นขนาดของระยะห่างเวกเตอร์ P1 และ P2

$\hat{e}_x$ เป็นเวกเตอร์หนึ่งหน่วยตามแนวแกน X

i เป็นการฉายเวกเตอร์ P13 (ซึ่งก็คือเวกเตอร์ P3 ลบด้วยเวกเตอร์ P1) ลงบนแกน X

$\hat{e}_y$ เป็นเวกเตอร์หนึ่งหน่วยตามแนวแกน Y

j เป็นการฉายเวกเตอร์ P13 (ซึ่งก็คือเวกเตอร์ P3 ลบด้วยเวกเตอร์ P1) ลงบนแกน Y

x y และ z เป็นพิกัดเครื่องรับสัญญาณจีเอ็นเอสเอส

จากสมการที่ 2.1 ถึงสมการที่ 2.8 สามารถใช้หาพิกัดเครื่องรับได้ โดยค่า x y และ z ที่ได้จะเป็นพิกัดแบบ ECEF (Earth-centered, Earth-fixed) ซึ่งจะอธิบายไว้ในหัวข้อที่ 2.2.1 เนื่องจากสัญญาณจีเอ็นเอสเอสถูกทำให้หน่วงเวลาโดยชั้นบรรยากาศไอโอโนสเฟียร์ ทำในการหาตำแหน่งของเครื่องรับ จะต้องแก้ไขค่าความผิดพลาดจากการหน่วงเวลาดังกล่าวโดยอาศัยเทคนิคตามหัวข้อที่ 2.5 ต่อไป

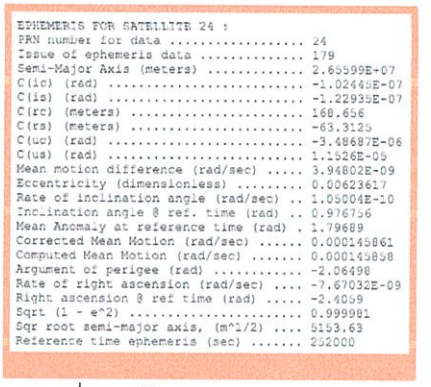
2.1.2.2 ข้อมูล Almanac และ ข้อมูล Ephemeris

Almanac เป็นข้อมูลที่เกี่ยวกับการโคจรของดาวเทียมแต่ละดวง เช่น ข้อมูลสภาพการใช้งานของดาวเทียม อายุของดาวเทียม ค่าความเยื้องการโคจรของดาวเทียม เป็นต้น ซึ่งจะถูกส่งจากดาวเทียมมาเก็บที่เครื่องรับสัญญาณจีเอ็นเอสเอส และจะปรับข้อมูลใหม่เป็นช่วงเวลาซ้ำๆ มีอายุการใช้งาน 90 วัน โดยจะใช้ในการประมาณตำแหน่งดาวเทียมและการรับสัญญาณดาวเทียมเพื่อนำไปคำนวณหาระยะห่างจากดาวเทียมถึงเครื่องรับสัญญาณบนพื้นโลกดังรูปที่ 2.8 ข้อมูล Almanac นี้สามารถดาวน์โหลดได้โดยใช้โปรแกรม RTKGET ซึ่งเป็นส่วนหนึ่งของโปรแกรม RTKLIB

```
******** Week 512 almanac for PRN-24 ********
ID:                         24
Health:                     000
Eccentricity:               0.6661891937E-002
Time of Applicability(s):   319488.0000
Orbital Inclination(rad):   0.9505615234
Rate of Right Ascen(r/s):   -0.7850758266E-008
SQRT(A)  (m 1/2):           5153.627930
Right Ascen at Week(rad):   -0.7220151424E-000
Argument of Perigee(rad):   -0.651554227
Mean Anom(rad):             -0.5415329933E-000
Af0(s):                     0.1974105835E-003
Af1(s/s):                   0.3637978807E-011 week: 512
```

รูปที่ 2.8 ข้อมูล Almanac [8]

Ephemeris เป็นข้อมูลที่ใช้ในการทำนายการโคจรของดาวเทียมซึ่งให้ความแม่นยำสูงกว่า Almanac ดาวเทียมแต่ละดวงจะมี Ephemeris เป็นของตนเองและส่งออกไปยังเครื่องรับสัญญาณจีเอ็นเอสเอสบนพื้นโลกอย่างต่อเนื่อง ประกอบไปด้วยข้อมูลเกี่ยวกับตำแหน่งและความเร็วเชิงมุมของดาวเทียม โดยจะมีการปรับข้อมูลใหม่ให้ตรงกับปัจจุบันทุก 2 ชั่วโมง แสดงได้ดังรูปที่ 2.9



```
EPHEMERIS FOR SATELLITE 24 :
PRN number for data ..................... 24
Issue of ephemeris data ................. 179
Semi-Major Axis (meters) ................ 2.65599E+07
C(ic) (rad) ............................. -1.02443E-07
C(is) (rad) ............................. -1.22935E-07
C(rc) (meters) .......................... 168.656
C(rs) (meters) .......................... -63.3125
C(uc) (rad) ............................. -3.48687E-06
C(us) (rad) ............................. 1.1526E-05
Mean motion difference (rad/sec) ..... 3.94802E-09
Eccentricity (dimensionless) ......... 0.00623617
Rate of inclination angle (rad/sec) .. 1.05004E-10
Inclination angle @ ref. time (rad) .. 0.976756
Mean Anomaly at reference time (rad) . 1.79689
Corrected Mean Motion (rad/sec) ...... 0.000145861
Computed Mean Motion (rad/sec) ....... 0.000145858
Argument of perigee (rad) ............ -2.06498
Rate of right ascension (rad/sec) .... -7.67032E-09
Right ascension @ ref time (rad) ..... -2.4059
Sqrt (1 - e^2) ....................... 0.999981
Sqr root semi-major axis, (m^1/2) .... 5153.63
Reference time ephemeris (sec) ....... 252000
```

รูปที่ 2.9 ข้อมูล Ephemeris [9]

ในการหาพิกัดของเครื่องรับสัญญาณจีเอ็นเอสเอส ที่เครื่องรับจะต้องรับข้อมูล Almanac และ Ephemeris มาคำนวณให้ได้ตำแหน่งดาวเทียมที่แม่นยำก่อน จากนั้นใช้เทคนิค Trilateration จึงสามารถหาพิกัดของตัวเครื่องรับเองได้

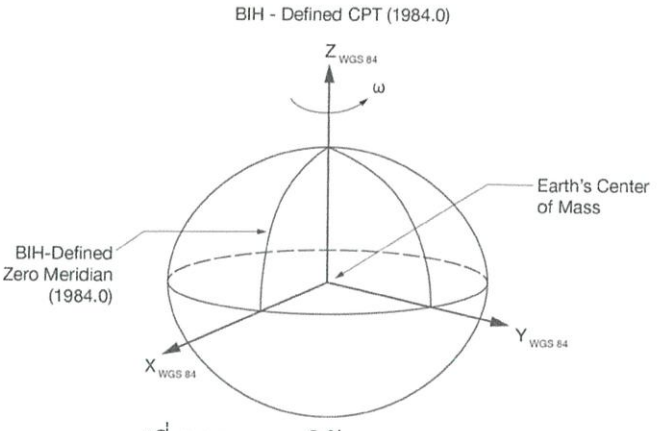
2.1.2.3 จำนวนดาวเทียม

จากการใช้เทคนิค Trilateration การใช้ดาวเทียมจำนวนมากขึ้น จะทำให้พิกัดของเครื่องรับที่เป็นไปได้อยู่ในบริเวณที่แคบลง เป็นการเพิ่มความแม่นยำของพิกัดเครื่องรับให้มากยิ่งขึ้น โดยปกติแล้ว ต้องใช้ดาวเทียมต่ำสุดเป็นจำนวน 4 ดวง (3 ดวงสำหรับ การระบุพิกัดของเครื่องรับแบบ 2 มิติ ดวงที่ 4 ช่วยระบุตำแหน่งในด้านความสูงของเครื่องรับ) เพื่อให้สามารถระบุพิกัดของเครื่องรับในรูปแบบ 3 มิติได้

2.2 ระบบพิกัดแผนที่

ระบบพิกัดในแผนที่ซึ่งถูกใช้ในปริญญานิพนธ์นี้มีด้วยกันอยู่ 2 ระบบ คือ ระบบพิกัดแบบ ECEF (Earth-centered, Earth-fixed) และระบบพิกัดแบบ Geodetic โดยแต่ละระบบจะมีรายละเอียดดังต่อไปนี้

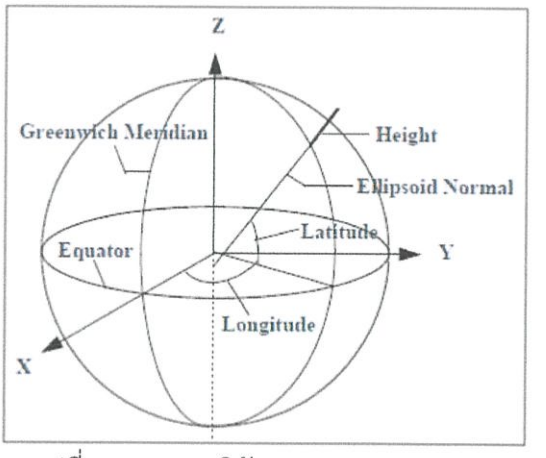
2.2.1 ระบบพิกัดแบบ ECEF



รูปที่ 2.10 ระบบพิกัดแบบ ECEF [10]

จากรูปที่ 2.10 ระบบพิกัดแบบ ECEF นี้ คล้ายกับพิกัดแบบคาทีเซียน โดยจุดกำเนิดของแกนมีตำแหน่งอยู่ที่ใจกลางโลก แกน X แกน Y และแกน Z ตั้งฉากกัน และมีหน่วยเป็นเมตร โดยที่ระนาบ XY วางตัวอยู่ในแนวตั้งฉากกับแกน Z ส่วนแกน Z วางตัวในแนวทิศตามทิศเหนือและทิศใต้ของโลก

2.2.2 ระบบพิกัดแบบ Geodetic



รูปที่ 2.11 ระบบพิกัดแบบ Geodetic [11]

เป็นระบบพิกัดที่บอกตำแหน่งเป็นค่าระยะเชิงมุมของละติจูด (Latitude) ลองจิจูด (Longitude) และความสูงในหน่วยเมตร ที่ห่างจากศูนย์กำเนิดของละติจูดและลองจิจูด โดยมีหน่วยเป็นองศาหรือเรเดียน

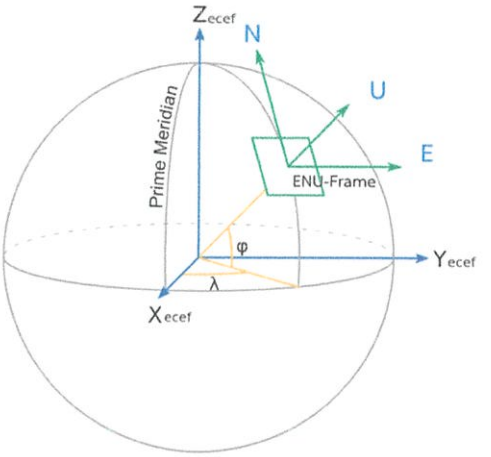
ศูนย์กำเนิดของละติจูด (Origin of latitude) กำหนดขึ้นจากแนวระดับที่ตัดผ่านศูนย์กลางของโลกและตั้งฉากกับแกนหมุน เรียกว่าเส้นระนาบศูนย์สูตรซึ่งแบ่งโลกออกเป็นซีกโลกเหนือและซีกโลกใต้ ค่าของมุมจะสิ้นสุดที่ขั้วโลกเหนือและขั้วโลกใต้ มีค่าเชิงมุม 90 องศาพอดี ดังนั้นการใช้ค่าระยะเชิงมุมของละติจูดอ้างอิงบอกตำแหน่ง จะกำกับด้วยตัวอักษรบอกทิศทางเหนือหรือใต้เสมอ เช่น ละติจูดที่ 30 องศา 20 ลิปดา 15 พิลิปดาเหนือ

ศูนย์กำเนิดของลองจิจูด (Origin of longitude) กำหนดขึ้นจากแนวระนาบที่ผ่านแกนหมุนของโลกตรงบริเวณตำแหน่งบนพื้นโลกที่ผ่านหอสังเกตการณ์ดาราศาสตร์ เมืองกรีนิช (Greenwich) ประเทศอังกฤษ เรียกศูนย์กำเนิดนี้ว่า เส้นเมริเดียนแรก (Prime meridian) เป็นเส้นที่แบ่งโลกออกเป็นซีกโลกตะวันตกและซีกโลกตะวันออก

ค่าระยะเชิงมุมของลองจิจูดเป็นค่าที่วัดมุมออกไปทางตะวันตก และตะวันออกของเส้นเมริเดียนแรก ค่าของมุมจะสิ้นสุดที่เส้นเมริเดียนตรงข้ามกับเส้นเมริเดียนแรก ซึ่งมีค่าของมุมซีกโลกละ 180 องศา การใช้ค่าอ้างอิงบอกตำแหน่งต้องบอกเป็นซีกโลกตะวันตก หรือซีกโลกตะวันออก เช่น ลองจิจูดที่ 90 องศา 20 ลิปดา 45 พิลิปดาตะวันตก ข้อดีของระบบนี้เทียบกับระบบพิกัดแบบ ECEF คือง่ายต่อการระบุพิกัดบนพื้นผิวโลกและง่ายในการคำนวณค่าแก้ให้กับพิกัดของสายอากาศ

2.2.3 ระบบพิกัดแบบ ENU [12]

ระบบพิกัดแบบ ENU (East-North-Up) เป็นระบบที่บอกตำแหน่งของวัตถุจากระยะฉายของวัตถุที่ตกลงบนแกนอ้างอิงซึ่งติดกับระนาบบนพื้นผิวโลกในหน่วยเมตร โดยมีแกนอ้างอิงทั้งหมด 3 แกน ได้แก่ แกน E (East) เป็นแกนอ้างอิงที่ชี้ไปทางทิศตะวันออก แกน N (North) เป็นแกนอ้างอิงที่ชี้ไปทางทิศเหนือ และแกน U (Up) ซึ่งเป็นแกนอ้างอิงที่มีทิศพุ่งออกจากพื้นผิวโลก แสดงได้ดังรูปที่ 2.12



รูปที่ 2.12 ระบบพิกัดแบบ ENU [12]

จากรูปที่ 2.12 ระยะขจัดตามแนวแกน E แกน N และแกน U สามารถหาได้จากการคำนวณเวกเตอร์ของสายอากาศร่วมกับมุมเอียงที่วัดด้วยเซนเซอร์ GY-85 แล้วนำไปคำนวณหาค่าแก้พิกัดที่เกิดจากความเอียงของสายอากาศ ซึ่งจะช่วยเพิ่มความแม่นยำในการระบุตำแหน่งของรถทางเกษตรกรรมบนพื้นที่เพาะปลูกจริงได้มากยิ่งขึ้น พิกัดชนิดนี้มีประโยชน์ในการคำนวณหาค่าแก้สำหรับแก้ความเอียงของสายอากาศ และคำนวณหาระยะห่างจากเส้นทางการเดินรถขณะทำการเกษตรด้วย

2.2.4 การแปลงระหว่างระบบพิกัดแบบ ECEF และระบบพิกัดแบบ Geodetic [13]

2.2.4.1 การแปลงระบบพิกัดแบบ ECEF ไปเป็นระบบพิกัดแบบ Geodetic

สมมติว่ามีพิกัดแบบ ECEF เป็น $(x,\ y,\ z)$ สามารถแปลงไปเป็นระบบพิกัดแบบ Geodetic ที่มีพิกัดเป็น $(\phi,\ \lambda,\ h)$ ได้ดังสมการที่ 2.9 ถึงสมการที่ 2.16

$$p = \sqrt{x^2 + y^2} \tag{2.9}$$

$$\theta = \tan^{-1}\left(\frac{za}{pb}\right) \tag{2.10}$$

$$e^2 = \frac{a^2 - b^2}{a^2} \tag{2.11}$$

$$e'^2 = \frac{a^2 - b^2}{b^2} \tag{2.12}$$

$$\lambda = \left(\tan^{-1}\left(\frac{y}{x}\right)\right)\left(\frac{180}{\pi}\right) \tag{2.13}$$

$$\phi = \left(\tan^{-1}\left(\frac{z+\sin^3(\theta)e'^2 b}{p-\cos^3(\theta)e^2 a}\right)\right)\left(\frac{180}{\pi}\right) \tag{2.14}$$

$$N = \frac{a}{\sqrt{1-e^2\sin^2\phi}} \tag{2.15}$$

$$h = \frac{p}{\cos(\phi)} - N \tag{2.16}$$

ตัวแปรในแต่ละสมการสามารถอธิบายได้ดังนี้

$(x,\ y,\ z)$ คือพิกัดในระบบ ECEF มีหน่วยเป็นเมตร

a เป็นค่าความยาวแกนเอกของโลก มีค่าเท่ากับ 6,378,136 เมตร

b เป็นค่าความยาวแกนโทของโลก มีค่าเท่ากับ 6,356,751 เมตร

θ ในสมการที่ 2.10 และ N ในสมการที่ 2.15 คือตัวแปรหุ่นเพื่อใช้หาค่า ϕ และ h ในสมการที่ 2.14 และสมการที่ 2.16 ต่อไป

e^2 และ e'^2 คือค่าความเยื้องศูนย์กลางของโลกยกกำลังสอง

λ คือค่าลองจิจูดในหน่วยองศา

ϕ คือค่าละติจูดในหน่วยองศา

h คือค่าความสูงในหน่วยเมตรจากผิวโลก

2.2.4.2 การแปลงระบบพิกัดแบบ Geodetic ไปเป็นระบบพิกัดแบบ ECEF

สมมติว่ามีพิกัดแบบ Geodetic เป็น $(\phi,\ \lambda,\ h)$ จะสามารถแปลงไปเป็นพิกัดแบบ ECEF ได้ตามสมการที่ 2.17 ถึงสมการที่ 2.20

$$N(\phi) = \frac{a}{\sqrt{1-e^2\sin^2(\phi)}} \tag{2.17}$$

$$x = \{N(\phi)+h\}\cos(\phi)\cos(\lambda) \tag{2.18}$$

$$y = \{N(\phi)+h\}\cos(\phi)\sin(\lambda) \tag{2.19}$$

$$z = \left\{\frac{b^2}{a^2}N(\phi)+h\right\}\sin(\phi) \tag{2.20}$$

ตัวแปรในแต่ละสมการ สามารถอธิบายได้ดังนี้

a เป็นค่าความยาวแกนเอกของโลก มีค่าเท่ากับ 6,378,136 เมตร

b เป็นค่าความยาวแกนโทของโลก มีค่าเท่ากับ 6,356,751 เมตร

e^2 คือค่าความเยื้องศูนย์กลางของโลกยกกำลังสอง

λ คือค่าลองจิจูดในหน่วยเรเดียน

ϕ คือค่าละติจูดในหน่วยเรเดียน

h คือค่าความสูงในหน่วยเมตรจากผิวโลก

$N(\phi)$ คือตัวแปรหุ่นเพื่อใช้คำนวณในสมการที่ 2.18 ถึงสมการที่ 2.20

ค่า x y และค่า z ตามสมการที่ 2.18 ถึงสมการที่ 2.20 คือพิกัดแบบ ECEF ในหน่วยเมตร

2.2.4.3 การแปลงระหว่างระบบพิกัดแบบ ENU ไปเป็นระบบพิกัดแบบ ECEF

เนื่องจากเวกเตอร์ของสายอากาศอยู่ในแบบ ENU หลังจากคำนวณร่วมกับมุมเอียงจากเซนเซอร์ GY-85 จะแปลงพิกัดแบบ ENU ที่ได้ไปเป็นพิกัดแบบ ECEF ให้ได้ค่าแก้ของพิกัดสายอากาศ สมการที่ใช้แปลงพิกัดจะอยู่ในสมการที่ 2.21

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -s(\lambda) & -c(\lambda)s(\phi) & c(\phi)c(\lambda) \\ c(\lambda) & -s(\phi)s(\lambda) & s(\lambda)c(\phi) \\ 0 & c(\phi) & s(\phi) \end{bmatrix} \begin{bmatrix} E \\ N \\ U \end{bmatrix} \tag{2.21}$$

โดยที่ $c(\lambda) = \cos(\lambda)$ $c(\phi) = \cos(\phi)$ $s(\lambda) = \sin(\lambda)$ และ $s(\phi) = \sin(\phi)$

ตัวแปรจากสมการที่ 2.21 มีความหมายดังนี้

$\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ คือ พิกัดสายอากาศในรูปแบบ ECEF จากอุปกรณ์จีเอ็นเอสเอส

λ คือ มุมลองจิจูดของเครื่องรับสัญญาณจีเอ็นเอสเอสในหน่วยองศา

ϕ คือ มุมละติจูดของเครื่องรับสัญญาณจีเอ็นเอสเอสในหน่วยองศา

$\begin{bmatrix} E & N & U \end{bmatrix}^T$ คือ พิกัดของสายอากาศในรูปแบบ ENU ในหน่วยเมตร

2.2.5 การแปลงระบบพิกัดแบบ Geodetic ไปอยู่ในระนาบ 2 มิติ

เพื่อความสะดวกและรวดเร็วในการหาระยะทางระหว่างจุดสองจุดใดๆ บนระบบพิกัดแบบ Geodetic จะต้องแปลงพิกัดแบบ Geodetic ที่มีหน่วยเป็นองศาไปเป็นระบบพิกัดแบบ 2 มิติ ที่บอกถึงความยาวที่แท้จริงระหว่างจุดปักหมุดแต่ละจุดได้ ซึ่งสมการที่ใช้ในการแปลงดังกล่าว จะเป็นไปตามสมการที่ 2.22 ถึงสมการที่ 2.24 [12]

$$x = 2\pi RX/360 \tag{2.22}$$

$$y = 2\pi RY/360 \tag{2.23}$$

ตัวแปรในแต่ละสมการ สามารถอธิบายได้ดังนี้

x คือ ค่าพิกัดแกน x

y คือ ค่าพิกัดแกน y

R คือ ค่ารัศมีโลกเท่ากับ 6,371,116 m.

X คือ ค่ามุมละติจูด

Y คือ ค่ามุมลองจิจูด

จากนั้นทำการหาระยะห่างระหว่างจุด 2 จุด โดยสมการหาค่าระยะห่างในระนาบ x – y แสดงดังสมการที่ 2.24

$$L = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \tag{2.24}$$

โดยที่

(x_0, y_0) คือ ค่าพิกัดของจุดเริ่มต้น

(x_i, y_i) คือ ค่าพิกัดของจุดสิ้นสุด

2.3 เครื่องรับสัญญาณจีเอ็นเอสเอสแบบความถี่เดียวยี่ห้อ U-blox NEO M8T

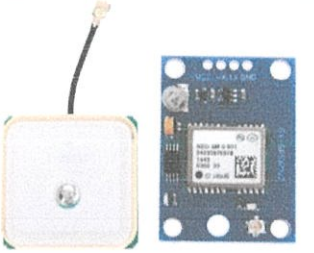
เป็นเครื่องรับสัญญาณที่รองรับการใช้งานในย่านความถี่ L1 (ประมาณ 1,575 MHz.) ซึ่งเป็นย่านกระจายสัญญาณจีเอ็นเอสเอสที่พลเรือนสามารถใช้งานได้ โดยสามารถรับสัญญาณจีเอ็นเอสเอสได้หลายระบบดาวเทียม เช่น ระบบ GPS ของอเมริกา ระบบ QZSS ของญี่ปุ่น ระบบ IRNSS ของอินเดีย

ระบบ GLONASS ของรัสเซีย ระบบ BeiDou ของจีน และระบบ SBAS ที่ใช้ในการส่งค่าปรับแก้ให้กับพิกัดที่ได้รับจากระบบจีเอ็นเอสเอส ทั้งมีระบบกรองสัญญาณรบกวนแบบแถบผ่าน ลักษณะของเครื่องรับอยู่ในรูปที่ 2.13

รูปที่ 2.13 เครื่องรับสัญญาณจีเอ็นเอสเอสแบบความถี่เดียวยี่ห้อ U-blox NEO M8T [14]

นอกจากนี้ ยังมีเครื่องรับรุ่นอื่นๆ ได้แก่ Ublox-NEO M6 และ Novatel GPStation-6 Series Receiver แสดงดังตารางที่ 2.1

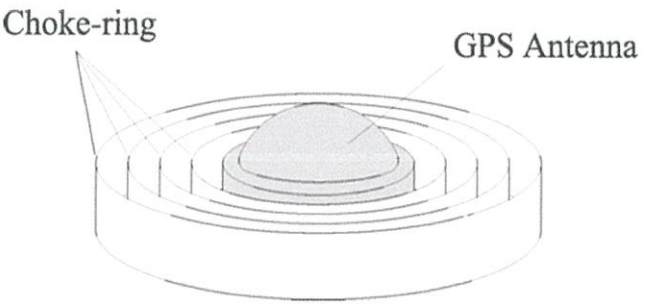
ตารางที่ 2.1 เครื่องรับสัญญาณจีเอ็นเอสเอส

ยี่ห้อเครื่องรับสัญญาณจีเอ็นเอสเอส	ภาพประกอบ
Ublox-NEO M6 • รับสัญญาณได้เฉพาะความถี่ L1 • มีความแม่นยำประมาณ 2.5 เมตร • มีสายอากาศประกอบแยกได้	

Novatel GPStation-6 Series Receiver • รับสัญญาณความถี่ L1 และ L2 ได้ • มีความแม่นยำประมาณ 8 เซนติเมตร	

2.4 สายอากาศ TW-3710

สายอากาศที่ใช้ในโครงงานนี้ รองรับการใช้งานสัญญาณจีเอ็นเอสเอสในย่านความถี่ L1 ได้หลายระบบ เช่นระบบ GPS ของสหรัฐอเมริกา ระบบ GLONASS ของรัสเซีย ระบบ QZSS ของญี่ปุ่น ระบบ Compass ของสาธารณรัฐประชาชนจีน ระบบ IRNSS ของอินเดีย และระบบ SBAS สามารถลดผลรบกวนจากสัญญาณพหุวิถีได้ดี มีอัตราขยายสัญญาณที่สูง ลักษณะของสายอากาศนี้ จะประกอบด้วยวงแหวนโลหะเป็นจำนวนหลายวงล้อมรอบสายอากาศหลัก ดังรูปที่ 2.14

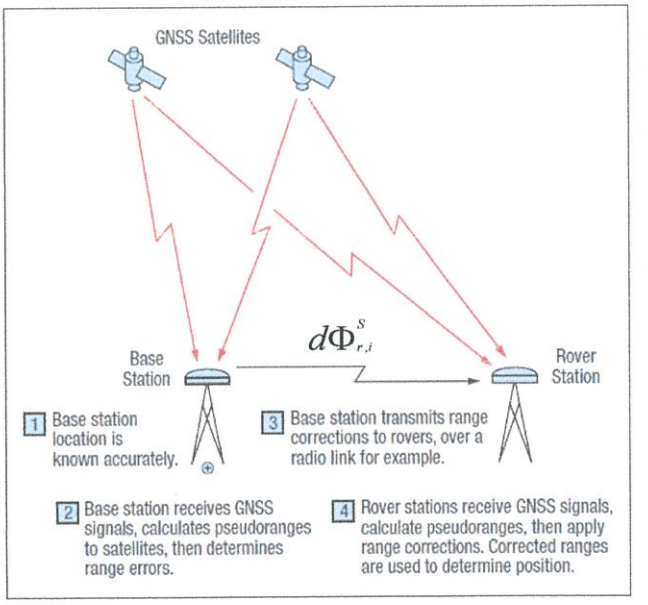


รูปที่ 2.14 สายอากาศแบบ Choke-ring [15]

จากรูปที่ 2.14 เมื่อมีสัญญาณพหุวิถีเข้ามา จะถูกเหนี่ยวนำโดยวงแหวนโลหะที่ล้อมรอบสายอากาศหลัก แล้ววิ่งลงระนาบกราวนด์ของสายอากาศ ในทางการสำรวจ มักจะมีการใช้สายอากาศรูปแบบนี้มาก เพราะสามารถลดผลรบกวนจากสัญญาณพหุวิถีได้ดี

2.5 หลักการคำนวณตำแหน่งด้วยเทคนิค RTK

Real Time Kinematic: RTK เป็นการรังวัดแบบจลน์ มีวิธีการทำงานคือ ต้องใช้เครื่องรับสัญญาณดาวเทียมจีเอ็นเอสเอสอย่างน้อยสองเครื่อง โดยเครื่องที่หนึ่งจะถูกวางไว้บนหมุดที่ทราบค่าพิกัด ในที่นี้จุดอ้างอิงที่ใช้ในปริญญานิพนธ์คือ สถานีฐาน (Base Station) ที่ติดตั้งอยู่บริเวณดาดฟ้าอาคารเรียนรวม 12 ชั้น คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อีกเครื่องจะถูกนำไปวาง ณ จุดที่ต้องการทราบค่าพิกัด (Rover) และเชื่อมต่อกันผ่านเครือข่ายวิทยุสำหรับการรับส่งข้อมูลระหว่างสถานีฐานและสถานีผู้ใช้ ข้อดีของเทคนิคนี้ คือ สามารถประมวลผลข้อมูลพร้อมกันขณะรังวัดทำให้ได้ค่าพิกัดตำแหน่งในทันที มีความถูกต้องทางตำแหน่งในระดับเซนติเมตร แต่มีข้อจำกัดคือ ความถูกต้องทางตำแหน่ง (Accuracy) และความน่าเชื่อถือของค่าพิกัด (Reliability) จะลดลงเมื่อระยะทางระหว่างสถานีฐานและสถานีผู้ใช้งานเพิ่มขึ้น ในกรณีที่สถานีฐานและสถานีจลน์อยู่ห่างไกลกันมากขึ้นและต้องการใช้เทคนิค RTK จะใช้วิธีเพิ่มความแม่นยำ โดยเชื่อมต่อสถานีจลน์เข้ากับระบบรังวัดด้วยดาวเทียมจีเอ็นเอสเอสเครือข่ายแบบจลน์ในทันที (Network-based RTK GPS) ซึ่งมีความถูกต้องทางตำแหน่งและความน่าเชื่อถือของค่าพิกัดสูงเป็นอันหนึ่งอันเดียวกันตลอดทั้งโครงข่ายจีเอ็นเอสเอส และครอบคลุมการทำงานในพื้นที่กว้างมาก โดยมีองค์ประกอบการของเทคนิคดังรูปที่ 2.15



รูปที่ 2.15 องค์ประกอบของเทคนิค RTK [16]

สมการที่ใช้อธิบายเทคนิคการทำงานของ RTK จะอยู่ในสมการที่ 2.25 ถึง 2.28

$$B_{r,i}^{S} = \phi_{r,0,i} - \phi_{0,i}^{S} + N_{r,i}^{S} \tag{2.25}$$

จากสมการที่ 2.25 ค่า $B_{r,i}^{S}$ เป็นค่าไบแอสหรือค่าจุดทำงานของเฟสคลื่นพาห์ของสัญญาณจีเอ็นเอสเอสของย่านความถี่ L_i หาได้จากผลต่างของจำนวนเฟสคลื่นพาห์ที่เครื่องรับสัญญาณจีเอ็นเอสเอส ($\phi_{r,o,i}$) กับจำนวนเฟสคลื่นพาห์ย่านความถี่ L_i ของดาวเทียม ($\phi_{0,i}^{s}$) ที่เวลา t_0 และจำนวนเฟสคลื่นพาห์ระหว่างดาวเทียมและเครื่องรับสัญญาณจีเอ็นเอสเอส ($N_{r,i}^{s}$) ค่าไบอัสนี้จะนำไปคูณกับค่าความยาวคลื่นย่านความถี่ L_i ใช้ร่วมการคำนวณหาระยะเทียมในสมการที่ 2.28 ซึ่งเป็นระยะห่างระหว่างดาวเทียมและเครื่องรับสัญญาณบนพื้นโลกต่อไป

$$d\Phi_{r,i}^{S} = -d_{r,pco,i^T} e_{r,enu}^{s} + \left(E^s d_{pco,i}^{S}\right)^T e_r^S + d_{r,pcv,i}(El) + d_{r,pcv,i}^{S}(\theta) - d_{r,disp}^{T} e_{r,enu}^{S} + \lambda_i \phi_{pw} \tag{2.26}$$

จากสมการที่ 2.26 ค่า $d\Phi_{r,i}^{s}$ เป็นค่าแก้ไขเฟสคลื่นพาห์ในย่านความถี่ L_i มีหน่วยเป็นจำนวนลูกคลื่น ซึ่งจะรวมเวคเตอร์ออฟเซตจุดกึ่งกลางเฟสของเครื่องรับสัญญาณจีเอ็นเอสเอส ณ เวลา T ($-d_{r,pco,i^T} e_{r,enu}^{s}$) เวคเตอร์ออฟเซตจุดกึ่งกลางเฟสของดาวเทียม $(E^s d_{pco,i}^{s})^T e_r^s$ ค่าความผันแปรจุดกึ่งกลางเฟสของเครื่องรับสัญญาณจีเอ็นเอสเอสในหน่วยเมตร ($d_{r,pco,i}(El)$) ค่าความแปรผันจุดกึ่งกลางเฟสของสายอากาศบนดาวเทียมในหน่วยเมตร ($d_{pcv,i}^{s}(\theta)$) เวคเตอร์การขจัดจากความโค้งของโลก ณ จุดที่เครื่องรับสัญญาณจีเอ็นเอสเอสตั้งอยู่ในหน่วยเมตร ($-d_{r,disp}^{T} e_{r,enu}^{s}$) และการเปลี่ยนแปลงของเฟสสัญญาณระหว่างสายอากาศบนดาวเทียมและสายอากาศของเครื่องรับบนผิวโลก ($\lambda_i \phi_{pw}$) ค่าแก้เหล่านี้จะถูกส่งไปแก้ไขความผิดพลาดที่สถานีจลน์ เพื่อเพิ่มความแม่นยำในการระบุตำแหน่งเครื่องรับด้วย

$$\begin{aligned}
\phi_{r,i}^{S} &= \phi_{r,i}(t_r) - \phi_i(t^S) + N_{r,i}^{s} + \varepsilon_\phi \\
&= \frac{c}{\lambda_i}\left(t_r - t^S\right) + \frac{c}{\lambda_i}\left(dt_r(t_r) - dT^S(t^S)\right) + \left(\phi_{r,0,j} - \phi_{0,j}^{S} + N_{r,i}^{S}\right) + \varepsilon_\phi \\
&= \frac{c}{\lambda_i}\left(t_r - t^S\right) + \frac{c}{\lambda_i}\left(dt_r(t_r) - dT^S(t^S)\right) + B_{r,i}^{S} + \varepsilon_\phi
\end{aligned} \tag{2.27}$$

สมการที่ 2.27 ค่า $\phi^s_{r,i}$ คือจำนวนเฟสของสัญญาณคลื่นพาห์ในย่านความถี่ L_i ระหว่างดาวเทียมและเครื่องรับสัญญาณจีเอ็นเอสเอสบนพื้นโลก โดยมีค่าเท่ากับผลรวมของจำนวนเฟสคลื่นพาห์ระหว่างดาวเทียมและเครื่องรับสัญญาณจีเอ็นเอสเอส ($\frac{c}{\lambda_i}\left(t_r - t^s\right)$) จำนวนเฟสคลื่นพาห์จากผลต่างค่าไบแอสนาฬิกาบนดาวเทียมและเครื่องรับสัญญาณจีเอ็นเอสเอส ($\frac{c}{\lambda_i}\left(dt_r(t_r) - dT^s(t^s)\right)$) ค่าไบอัสเฟสคลื่นพาห์จากสมการที่ 2.25 และความผิดพลาดในการวัดเฟสสัญญาณ (ε_ϕ) สมการที่ 2.27 นี้ เมื่อนำไปคูณกับค่าความยาวคลื่นของความถี่ L_i จะได้ระยะเทียมสำหรับคำนวณพิกัดของเครื่องรับสัญญาณต่อไป ตามสมการที่ 2.28

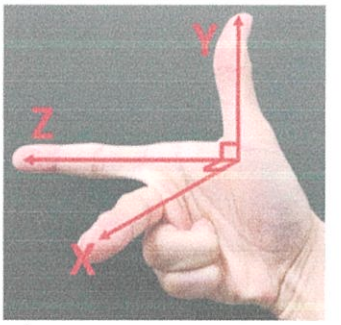
$$\begin{aligned}\Phi^s_{r,i} &= \lambda_i \phi^s_{r,j} \\ &= \rho^s_r + c\left(dt_r(t_r) - dT^s(t^s)\right) - I^s_{r,i} + T^s_r + \lambda_i B^s_{r,i} + d\Phi^s_{r,i} + \varepsilon_\Phi\end{aligned} \tag{2.28}$$

ในสมการที่ 2.28 ค่า $\Phi^s_{r,i}$ จะเป็นระยะเทียมจากย่านความถี่ L_i ในหน่วยเมตร ได้จากการนำความเร็วแสงไปคูณกับสมการที่ 2.27 โดยจะมีการบวกระยะเรขาคณิตระหว่างดาวเทียมถึงเครื่องรับสัญญาณจีเอ็นเอสเอส และลบระยะจากการหน่วงของสัญญาณในชั้นไอโอโนสเฟียร์ ($I^s_{r,i}$) ระยะจากการหน่วงของสัญญาณในชั้นโทรโปสเฟียร์ (T^s_r) และค่าแก้ไขเฟสคลื่นพาห์ในย่านความถี่ L_i ค่าในสมการที่ 2.28 นี้ จะนำไปคำนวณหาพิกัดของเครื่องรับด้วยเทคนิค Trilateration ในหัวข้อที่ 2.1.2.1 ต่อไป

จากสมการที่ 2.27 จะทำให้เราทราบจำนวนเฟสสัญญาณคลื่นพาห์ซึ่งสามารถนำไปคำนวณหาระยะเทียมระหว่างดาวเทียมถึงตัวรับสัญญาณบนโลกได้ โดยนำไปคูณกับค่าความยาวคลื่นของสัญญาณคลื่นพาห์ตามสมการที่ 2.28 ในทางปฏิบัติ การหาพิกัดของอุปกรณ์ตัวรับจะต้องต่อสายอากาศเข้ากับตัวรับสัญญาณ และขณะที่รับสัญญาณนั้น โลกก็มีการเคลื่อนที่ตลอดเวลาด้วย ดังนั้นการหาระบุตำแหน่งของตัวรับให้มีความถูกต้องมากที่สุด จะต้องรวมจุดการทำงานของเฟสคลื่นพาห์และค่าแก้เฟสสัญญาณคลื่นพาห์เข้าไปด้วย โดยในสมการที่ 2.25 จะมีตัวแปร $B^s_{r,i}$ ซึ่งเป็นค่าจุดทำงานของเฟสคลื่นพาห์ย่านความถี่ L_i และสมการที่ 2.26 มีตัวแปร $d\Phi^s_{r,i}$ เป็นค่าแก้เฟสสัญญาณคลื่นพาห์ย่านความถี่ L_i รวมอยู่ในสมการที่ 2.28 แล้ว หลังจากทราบระยะเทียมของดาวเทียม 4 ดวงขึ้นไป ก็จะสามารถหาพิกัดของเครื่องรับสัญญาณจีเอ็นเอสเอสบนโลกได้โดยใช้เทคนิค Trilateration

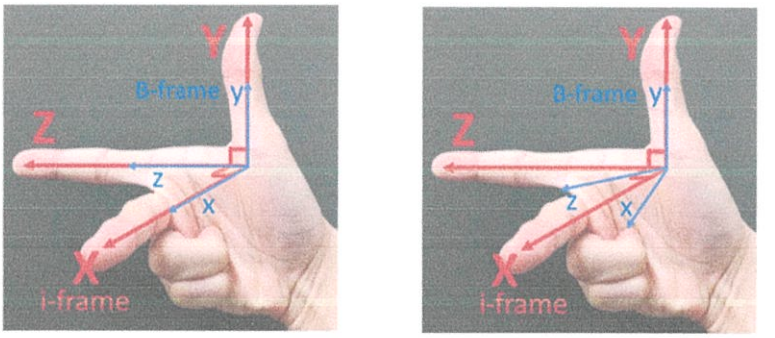
2.6 หลักการ Euler Angles และ Rotation Matrix

ในการระบุว่าวัตถุมีการวางตัว หรือมีการเคลื่อนที่ไปจากตำแหน่งเดิมอย่างไร จำต้องวัดมุมการวางตัวของวัตถุนั้นเทียบกับแกนอ้างอิง และต้องเข้าใจลำดับการหมุนของวัตถุจากจุดเดิมไปยังจุดใหม่ก่อน โดยทั่วไปมักกำหนดแกนอ้างอิงของวัตถุออกเป็นสามแกนหลัก คือ แกน X แกน Y และแกน Z โดยที่ทั้งสามแกนจะวางตัวในแนวตั้งฉากกัน และเป็นไปตามกฎมือขวา ดังเช่นรูปที่ 2.16



รูปที่ 2.16 แกนอ้างอิงแบบ X-Y-Z

จากรูปที่ 2.16 เมื่อนำไปใช้ระบุการเคลื่อนที่ของวัตถุว่าทำมุมอย่างไรกับแกนอ้างอิงของโลก(กำหนดให้เป็น I-frame) จะต้องใช้แกนอ้างอิงของวัตถุ (กำหนดให้เป็น B-frame ของ เซนเซอร์ รุ่น GY-85) มาช่วยในการระบุด้วย สมมุติว่าให้วัตถุหมุนรอบแกน Y ในทิศทวนเข็มนาฬิกา 90° จะได้การวางตัวของ B-frame ใหม่เทียบกับ I-frame ดังรูปที่ 2.17



Before rotating around Y-axis

After rotating around Y-axis for 90 deg. In counterclockwise.

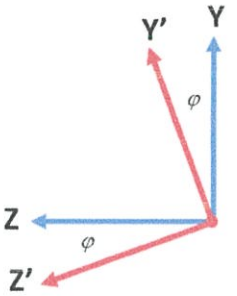
รูปที่ 2.17 การหมุน B-frame เทียบ I-frame

หลังจากกำหนดแกนอ้างอิงของวัตถุแล้ว จะเริ่มกล่าวถึง Euler Angles และ Rotation Matrix เพื่อใช้สำหรับแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศ

2.6.1 Euler Angles [17]

Euler Angles คือมุมที่เกิดจากการหมุนวัตถุบน I-frame เทียบกับ B-frame ซึ่งมีผลลัพธ์ของแต่ละลำดับการหมุนดังนี้

ขั้นตอนที่ 1 หมุน I-frame รอบแกน X ด้วยมุม Roll (φ) ดังรูปที่ 2.18



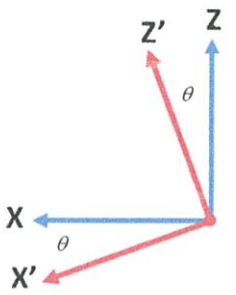
รูปที่ 2.18 การหมุน I-frame ด้วยมุม φ (แกน X พุ่งออกจากหน้ากระดาษ)

จากการหมุน I-frame ด้วยมุม φ จะได้ Rotation Matrix รอบแกน X (R_x) เป็น

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\varphi) & s(\varphi) \\ 0 & -s(\varphi) & c(\varphi) \end{bmatrix} \tag{2.29}$$

โดยที่ $c(\varphi) = \cos(\varphi)$ และ $s(\varphi) = \sin(\varphi)$

ขั้นตอนที่ 2 หมุน I-frame รอบแกน Y ด้วยมุม Pitch (θ) ดังรูปที่ 2.19



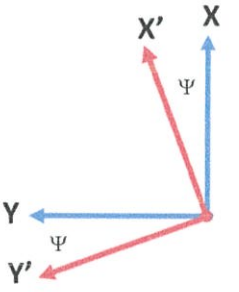
รูปที่ 2.19 การหมุน I-frame ด้วยมุม θ (แกน Y พุ่งออกจากหน้ากระดาษ)

จากการหมุน I-frame ด้วยมุม θ จะได้ Rotation Matrix รอบแกน Y (R_y) เป็น

$$R_y = \begin{bmatrix} c(\theta) & 0 & -s(\theta) \\ 0 & 1 & 0 \\ c(\theta) & 0 & s(\theta) \end{bmatrix} \tag{2.30}$$

โดยที่ $c(\theta) = \cos(\theta)$ และ $s(\theta) = \sin(\theta)$

ขั้นตอนที่ 3 หมุน I-frame รอบแกน Z ด้วยมุม Yaw (Ψ) ดังรูปที่ 2.20



รูปที่ 2.20 การหมุน I-frame ด้วยมุม Ψ (แกน Z พุ่งออกจากหน้ากระดาษ)

จากการหมุน I-frame ด้วยมุม Ψ จะได้ Rotation Matrix รอบแกน Z (R_z) เป็น

$$R_z = \begin{bmatrix} c(\Psi) & s(\Psi) & 0 \\ -s(\Psi) & c(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.31}$$

โดยที่ $c(\Psi) = \cos(\Psi)$ และ $s(\Psi) = \sin(\Psi)$

ในการอธิบายลำดับการหมุนของวัตถุด้วยวิธี Euler Angles บนระบบพิกัดแกนที่เป็น 3 มิติ จะใช้ลำดับการหมุนที่สำคัญด้วยกันอยู่ 2 แบบ Roll-Pitch-Yaw หรือ Yaw-Pitch-Roll ซึ่งในปริญญานิพนธ์นี้ เลือกใช้การหมุนแบบ Yaw-Pitch-Roll

2.6.2 Rotation Matrix [18]

จากการอธิบายใน Euler Angles แล้ว การหา Rotation Matrix ตามลำดับการหมุน Yaw-Pitch-Roll กำหนดให้ $\mathbf{R}_{\mathbf{I}}^{\mathbf{B}}$ เป็น Rotation Matrix ที่เกิดจากการหมุน I-frame เทียบกับ B-frame และมีลำดับการหมุนแบบ Yaw-Pitch-Roll เมื่อนำ Rotation Matrix ของแต่ละแกนมาเขียนรวม จะได้ $\mathbf{R}_{\mathbf{I}}^{\mathbf{B}}$ ดังสมการที่ 2.32 [16]

$$\begin{aligned} \mathbf{R}_{\mathbf{I}}^{\mathbf{B}} &= \mathbf{R}_{\mathbf{x}}(\varphi)\mathbf{R}_{\mathbf{y}}(\theta)\mathbf{R}_{\mathbf{z}}(\Psi) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\varphi) & s(\varphi) \\ 0 & -s(\varphi) & c(\varphi) \end{bmatrix} \begin{bmatrix} c(\theta) & 0 & -s(\theta) \\ 0 & 1 & 0 \\ s(\theta) & 0 & c(\theta) \end{bmatrix} \begin{bmatrix} c(\Psi) & s(\Psi) & 0 \\ -s(\Psi) & c(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c(\varphi)c(\Psi) & c(\theta)s(\Psi) & -s(\theta) \\ s(\varphi)s(\theta)c(\Psi) - c(\varphi)s(\Psi) & s(\varphi)s(\theta)s(\Psi) + c(\varphi)c(\Psi) & s(\varphi)c(\theta) \\ c(\varphi)s(\theta)c(\Psi) + s(\varphi)s(\Psi) & c(\varphi)s(\theta)s(\Psi) - s(\varphi)c(\Psi) & c(\varphi)c(\theta) \end{bmatrix} \end{aligned} \tag{2.32}$$

ในการแก้ไขพิกัดที่ผิดพลาดเนื่องจากความเอียงของสายอากาศ จะต้องใช้ Rotation Matrix ของการหมุน B-frame เทียบกับ I-frame หรือ $\mathbf{R}_{\mathbf{B}}^{\mathbf{I}}$ สามารถหาได้จากเมทริกผกผันของ $\mathbf{R}_{\mathbf{I}}^{\mathbf{B}}$ ดังสมการที่ 2.33 โดยมีลำดับการหมุนเป็น Roll-Pitch-Yaw แทน

$$\begin{aligned} \mathbf{R}_{\mathbf{B}}^{\mathbf{I}} &= (\mathbf{R}_{\mathbf{I}}^{\mathbf{B}})^{-1} \\ &= (\mathbf{R}_{\mathbf{I}}^{\mathbf{B}})^{\mathbf{T}} \end{aligned}$$

$$= \begin{bmatrix} c(\varphi)c(\Psi) & s(\varphi)s(\theta)c(\Psi) - c(\varphi)s(\Psi) & c(\varphi)s(\theta)c(\Psi) + s(\varphi)s(\Psi) \\ c(\theta)s(\Psi) & s(\varphi)s(\theta)s(\Psi) + c(\varphi)c(\Psi) & c(\varphi)s(\theta)s(\Psi) - s(\varphi)c(\Psi) \\ -s(\theta) & s(\varphi)c(\theta) & c(\varphi)c(\theta) \end{bmatrix} \quad (2.33)$$

สมการที่ 2.33 จะใช้แสดงตำแหน่งของ B-frame เทียบกับ I-frame สมมุติให้เวกเตอร์ $\begin{bmatrix} E & N & U \end{bmatrix}_{initial}^{T}$ เป็นเวกเตอร์ของสายอากาศที่มีพิกัดแบบ ENU มีทิศพุ่งจากฐานของเสาสายอากาศไปยังปลายสายอากาศสูง h เมตร สามารถเขียนได้ดังสมการที่ 2.34

$$\begin{bmatrix} E \\ N \\ U \end{bmatrix}_{initial} = \begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix} \quad (2.34)$$

จากสมการที่ 2.34 เมื่อสายอากาศเกิดความเอียงด้วยมุม Roll มุม Pitch และมุม Yaw ระยะเอียงของสายอากาศในแต่ละทิศ สามารถหาได้จากการนำเวกเตอร์ในสมการที่ 2.34 ไปคูณกับสมการที่ 2.33 จะได้การขจัดของสายอากาศในแต่ละทิศดังสมการที่ 2.35

$$\begin{bmatrix} E \\ N \\ U \end{bmatrix}_{Adj} = (-1) \bullet \mathbf{R}_{\mathbf{B}}^{\mathbf{I}} \begin{bmatrix} E \\ N \\ U \end{bmatrix}_{initial} \quad (2.35)$$

จากสมการที่ 2.35 จะได้ระยะขจัดของสายอากาศในแต่ละทิศ มีพิกัดแบบ ENU และเมื่อนำพิกัดแบบ ENU นี้ไปใช้แก้ไขความเอียงของสายอากาศ จะต้องแปลงพิกัดในสมการที่ 2.36 ไปเป็นพิกัดแบบ ECEF แล้วนำไปปรับแก้กับพิกัดแบบ ECEF (Earth-centered Earth-fixed Coordinate) ของสายอากาศดังสมการที่ 2.36

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} -s(\lambda) & -c(\lambda)s(\phi) & c(\phi)c(\lambda) \\ c(\lambda) & -s(\phi)s(\lambda) & s(\lambda)c(\phi) \\ 0 & c(\phi) & s(\phi) \end{bmatrix} \begin{bmatrix} E \\ N \\ U \end{bmatrix}_{Adj} \quad (2.36)$$

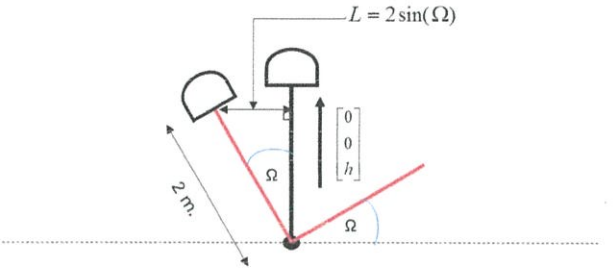
ตัวแปรจากสมการที่ 2.36 มีความหมายดังนี้

$[X' \quad Y' \quad Z']^T$ คือ พิกัดสายอากาศในรูปแบบ ECEF ที่แก้ไขความเอียงแล้ว

$[X \quad Y \quad Z]^T$ คือ พิกัดสายอากาศในรูปแบบ ECEF จากอุปกรณ์จีเอ็นเอสเอส

λ คือ มุมลองจิจูดของเครื่องรับสัญญาณจีเอ็นเอสเอสในหน่วยองศา

ϕ คือ มุมละติจูดของเครื่องรับสัญญาณจีเอ็นเอสเอสในหน่วยองศา



รูปที่ 2.21 ระยะห่างที่เกิดจากการเอียงของสายอากาศ

จากรูปที่ 2.21 จะมีสมการใช้ตรวจสอบระยะห่าง (L) มีหน่วยเป็นเมตร ดังสมการที่ 2.37

$$L = 2\sin(\Omega) \tag{2.37}$$

โดยที่ Ω คือ มุม Roll หรือมุม Pitch ในหน่วยองศา ในกรณีของมุม Yaw คือการที่สายอากาศหมุนอยู่กับที่ ไม่มีการเอียงไปข้างใดข้างหนึ่ง ดังนั้นระยะที่ได้จากการหมุนด้วยมุม Yaw ควรจะมีค่าประมาณ 0 เมตร

2.7 หลักการของ IMU เซนเซอร์ รุ่น GY-85

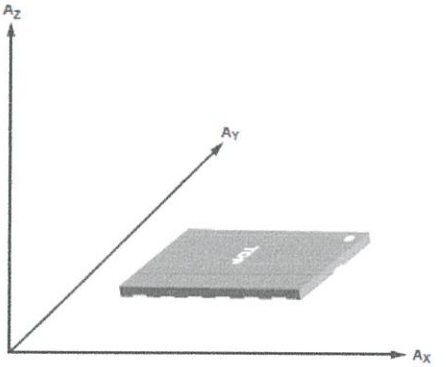
IMU (Inertial Measurement Unit) เป็นอุปกรณ์ทางไฟฟ้าที่รวมการทำงานของเครื่องวัดความเร่ง (Accelerometer) และเครื่องวัดการหมุน (Gyroscope) บางครั้งอาจรวมเครื่องวัดความเข้มสนามแม่เหล็ก (Magnetometer) เอาไว้ด้วย เพื่อใช้ในการวัดและแสดงการเคลื่อนที่ของวัตถุ ปัจจุบันมีการประยุกต์ใช้อุปกรณ์ชนิดนี้กับงานสำรวจ การบิน และใช้ทางการทหาร รายละเอียดการทำงานของแต่ละอุปกรณ์ใน เซนเซอร์ รุ่น GY-85 สามารถอธิบายได้ดังหัวข้อที่ 2.7.1 ถึง 2.7.3

2.7.1 หลักการทำงานของเครื่องวัดความเร่ง (Accelerometer)

ในปริญญานิพนธ์นี้ ใช้เครื่องวัดความเร่งเป็นรุ่น ADXL345 ดังรูปที่ 2.21 ในการวัดการเคลื่อนที่ของวัตถุ

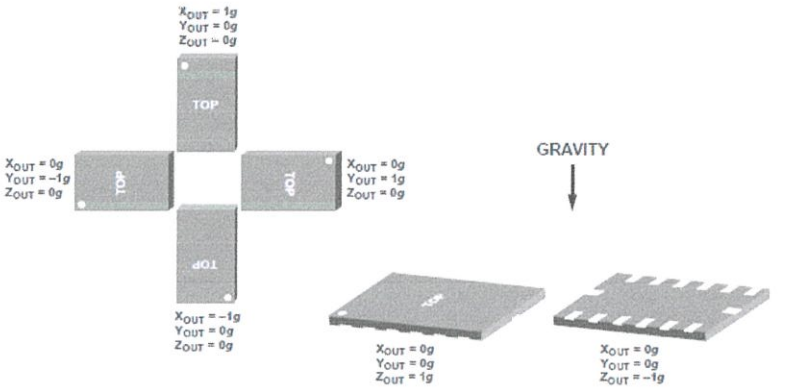
รูปที่ 2.22 เซนเซอร์ ADXL345 [19]

จากรูปที่ 2.22 ภายในเซนเซอร์ ADXL345 จะมีสปริงขนาดเล็กที่ใช้วัดแรงซึ่งมากระทำต่อสปริงและถูกแปลผลเป็นแรงดันไฟฟ้าเพื่อนำไปคำนวณหาค่าความเร่งที่เกิดขึ้น ซึ่งสามารถวัดได้ 3 แกนหลัก ได้แก่ แกน X (Ax) แกน Y (Ay) และแกน Z (Az) โดยการวางตัวของแต่ละแกนดังรูปที่ 2.23



รูปที่ 2.23 แนวการวางตัวของแกน X แกน Y และแกน Z ของเซนเซอร์ ADXL345 [20]

จากรูปที่ T12 หากนำเซนเซอร์ ADXL 345 วางตัวโดยให้แกน X แกน Y หรือแกน Z ตั้งฉากกับพื้นโลก จะได้ค่าความเร่งของแต่ละแกนดังรูปที่ 2.24

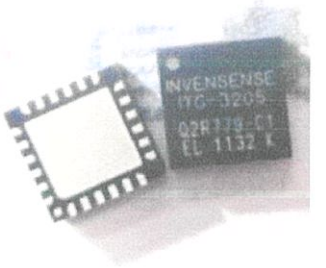


รูปที่ 2.24 ความเร่งที่วัดได้จากการวางเซนเซอร์ ADXL345 ให้แกน X แกน Y หรือแกน Z ตั้งฉากกับพื้นโลก [21]

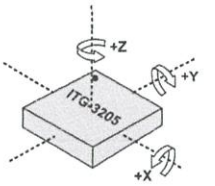
จากรูปที่ 2.24 สรุปได้ว่าค่าความเร่งที่เซนเซอร์ ADXL345 วัดได้ จะมีทิศทางตรงข้ามกับความเร่งหรือแรงที่มากระทำต่อตัวมันเอง ในที่นี้ แรงที่มากระทำจะเป็นแรงโน้มถ่วงของโลก ซึ่งมีค่าความเร่ง (g) ค่าประมาณ 9.8 m/s^2 ในการคำนวณหาทิศทางการเคลื่อนที่ของวัตถุ สามารถทำได้โดยการนำค่าความเร่งของแต่ละแกนมาคำนวณหามุมที่เอียงไปของวัตถุได้

2.7.2 หลักการทำงานของเครื่องวัดการหมุน (Gyroscope)

หน้าที่หลักของ Gyroscope คือวัดความเร็วเชิงมุมของแกน X แกน Y และแกน Z ในปริญญานิพนธ์นี้ เลือกใช้ Gyroscope รุ่น ITG-3205 ซึ่งมีลักษณะดังรูปที่ 2.25 และมีแนวการวางตัวแกน X แกน Y และแกน Z ดังรูปที่ 2.26



รูปที่ 2.25 เซนเซอร์ ITG-3205 [22]



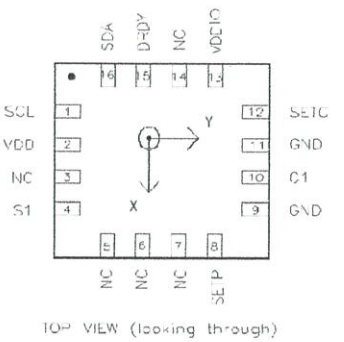
รูปที่ 2.26 แนวการวางตัวแกน X แกน Y และแกน Z ของเซนเซอร์ ITG-3205 [23]

ในการวัดการเคลื่อนที่ของวัตถุโดยใช้ Gyroscope เมื่อมีแรงมากระทำกับ Gyroscope จะทำให้สปริงภายในเกิดการสั่นสะเทือนแล้วสร้างแรงดันไฟฟ้าขึ้น จากนั้นแรงดันฟ้านี้จะถูกนำไปประมวลผลและอ่านค่าเป็นความเร็วเชิงมุมที่แต่ละเวลาออกมา เมื่อนำความเร็วเชิงมุมที่ได้ไปหาปริพันธ์ก็ จะสามารถคำนวณมุมเอียงของวัตถุได้ ทำให้ทราบการเคลื่อนที่ของวัตถุในที่สุด

2.7.3 หลักการทำงานของเครื่องวัดความเข้มสนามแม่เหล็ก (Magnetometer)

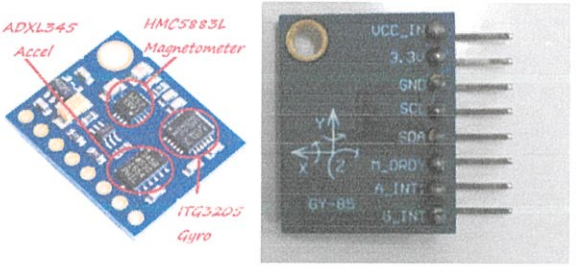
การทำงานของเซนเซอร์ Magnetometer จะใช้การตรวจจับความแรงสนามแม่เหล็กแล้วแปลผลเป็นแรงดันไฟฟ้า เพื่อนำไปแปลผลเป็นทิศทางต่อไป ในปริญญานิพนธ์นี้ เลือกใช้เซนเซอร์ Magnetometer รุ่น HMC5883L ดังรูปที่ 2.27 ในการคำนวณหาทิศทางของวัตถุว่ากำลังชี้ไปในทิศทางใด สามารถหาได้จากการเปรียบเทียบความแรงของสนามแม่เหล็กที่วัดได้ในแต่ละแกน โดยจะมีแกนหลักอยู่ 3 แกน คือ แกน X แกน Y และแกน Z ดังรูปที่ 2.28 แล้วแปลผลเป็นมุมออกมา ก็จะทราบทิศทางของวัตถุที่ชี้ไปได้

รูปที่ 2.27 เซนเซอร์ HMC5883L [24]



รูปที่ 2.28 แนวการวางตัวแกน X แกน Y และแกน Z ของเซนเซอร์ HMC5883L [25]

2.7.4 ข้อมูลเฉพาะของเซนเซอร์ รุ่น GY-85



รูปที่ 2.29 เซนเซอร์ รุ่น GY-85 [26]

จากรูปที่ 2.29 เซนเซอร์ รุ่น GY-85 ประกอบไปด้วย Accelerometer (ADXL345) Gyroscope (ITG3205) และ Compass (HMC5883L) ซึ่งสามารถซึ่งมีหลักการทำงานของแต่ละอย่างดังที่ได้อธิบายไปแล้วในหัวข้อที่ 2.7.1 ถึงหัวข้อที่ 2.7.3 ในการใช้เซนเซอร์ชนิดนี้ร่วมกับหน่วยประมวลผล จะต้องเชื่อมต่อผ่าน Bus I2C กับหน่วยประมวลผล เพื่อส่งข้อมูลให้หน่วยประมวลไปใช้งานต่อไป รายละเอียดขาของเซนเซอร์ รุ่น GY-85 ที่รับและส่งข้อมูล แสดงดังตารางที่ 2.1 และแสดงการเชื่อมต่อกับหน่วยประมวลผลได้ดังรูปที่ 2.30

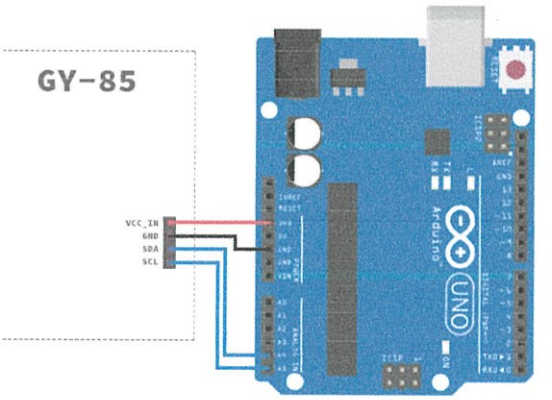
คุณสมบัติ (Features)

- ใช้ไฟเลี้ยง +3.3 ถึง +5 V
- ใช้เซนเซอร์ ADXL345 เซนเซอร์ HMC5883L และเซนเซอร์ ITG3205
- เชื่อมต่อกับหน่วยประมวลผลผ่านบัส I2C
- Accelerometer มีความละเอียดสูงสุด 16g (16 เท่าของค่าความโน้มถ่วงโลก)

- Gyroscope สามารถวัดการเปลี่ยนแปลงมุมได้สูงสุด 2000 องศาต่อวินาที
- Compass มีความแม่นยำในการวัดทิศทาง 1 องศา

ตารางที่ 2.2 ข้อมูลการต่อใช้งานขาอุปกรณ์ เซนเซอร์ รุ่น GY-85

Pin	Name	Characteristic
1	VCC_IN	ขารับไฟ +5 โวลต์ไปที่ Regulate 3.3 โวลต์
2	3.3V	ขาไฟ 3.3 โวลต์
3	GND	กราวนด์
4	SCL	ขาสัญญาณนาฬิกา บนบัส I2C
5	SDA	ขาสัญญาณข้อมูล บนบัส I2C
6	M_DRDY	Interrupt ของ HMC5883L
7	A_INT1	Interrupt ของ ADXL345B
8	G_INT	Interrupt ของ ITG3205



รูปที่ 2.30 การต่อ เซนเซอร์ รุ่น GY-85 เข้ากับหน่วยประมวลผล

2.8 ภาษาไพธอน [27]

ภาษาไพธอน (Python) ถือกำเนิดขึ้นในปี ค.ศ.1980 ถูกพัฒนาขึ้นโดย Guido van Rossum ที่ Centrum Wiskunde & Informatica (CWI) ประเทศเนเธอร์แลนด์ เป็นภาษาหนึ่งที่นิยมใช้ในการเขียนโปรแกรม ถูกพัฒนามาโดยไม่ยึดติดกลับแพลตฟอร์ม นั่นหมายความว่าภาษาไพธอนสามารถใช้งานได้ในหลายระบบปฏิบัติการ ทั้ง Unix Linux และ Windows อีกทั้งยังเป็น Open

Source ทำให้นักพัฒนาโปรแกรมสามารถใช้ภาษาไพธอนมาพัฒนาโปรแกรมหรือซอฟต์แวร์ของตัวเองได้โดยไม่เสียค่าใช้จ่ายแต่อย่างใด ภาษาไพธอนเป็นภาษาระดับสูง จึงทำให้เขียนง่าย ผู้ใช้งานไม่จำเป็นต้องมีความรู้ทางด้านคอมพิวเตอร์อย่างลึกซึ้ง เพียงแค่เข้าใจไวยากรณ์และขั้นตอนการทำงานของภาษาไพธอนก็สามารถพัฒนาโปรแกรมด้วยภาษาไพธอนได้แล้ว แต่เนื่องจากความเป็นภาษาระดับสูงที่มีข้อดีคือเขียนและพัฒนาง่ายแล้วก็จะมีข้อจำกัดอยู่ข้อหนึ่งคือจะทำงานหรือประมวลผลได้ช้าเมื่อเปรียบเทียบกับภาษาอื่น เช่น ซี หรือ ฟอร์แทรน

2.8.1 ส่วนประกอบในการทำงานของภาษาไพธอน

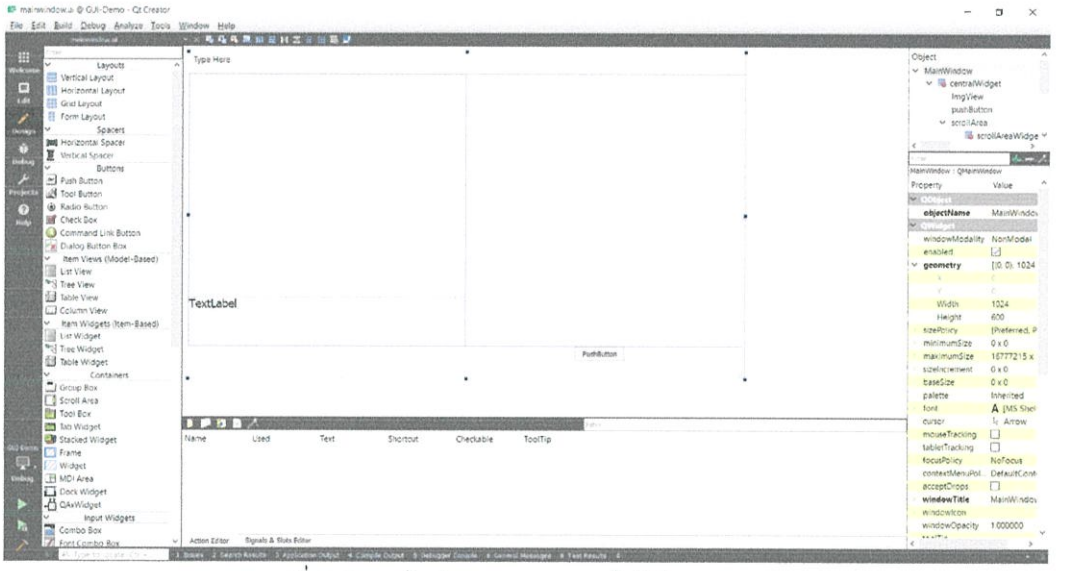
- คอมไพเลอร์ (Compiler) ใช้ตีความคำสั่งที่ผู้พัฒนาโปรแกรมเขียนขึ้นเพื่อสั่งให้คอมพิวเตอร์ทำงาน
- อีดิเตอร์ (Editor) ใช้เขียนชุดคำสั่งเพื่อให้คอมไพเลอร์อ่านแล้วสั่งคอมพิวเตอร์อีกครั้งหนึ่ง
- เชลโต้ตอบ (interactive shell) หรือ อินเทอร์พรีเตอร์ (interpreter) ใช้ป้อนคำสั่งเพื่อสั่งการคอมพิวเตอร์แบบทันที

2.8.2 ข้อดีของภาษาไพธอน

- ไพทอนเป็นภาษาสคริปต์ ทำให้ใช้เวลาในการเขียนและคอมไพล์ไม่มาก เหมาะกับงานด้านการดูแลระบบ (System administration)
- คำสั่งที่เขียนด้วยภาษาไพธอนสามารถนำไปทำงานบนระบบปฏิบัติการได้หลากหลาย
- การประมวลผลจะทำในแบบอินเทอร์พรีเตอร์ คือ จะประมวลผลไปทีละบรรทัดและปฏิบัติตามคำสั่งที่ได้รับ
- เป็นภาษาแบบ Dynamic typing คือ สามารถเปลี่ยนชนิดข้อมูลได้ง่ายและสะดวกรวดเร็ว
- มี Buil-in Object Types คือ โครงสร้างของข้อมูลที่สามารถใช้ได้ใน Python ประกอบด้วย ลิสต์ ดิกชันนารี สตริง ที่ง่ายต่อการใช้งานและมีประสิทธิภาพสูง
- มีฟังก์ชันสนับสนุนฐานข้อมูล เช่น MySQL Sybase Oracle Informix ODBC (Open Database Connectivity) และอื่นๆ
- มีไลบรารีสนับสนุนด้านปัญญาประดิษฐ์

2.9 โปรแกรม QT [28]

โปรแกรม QT เป็นโปรแกรมที่ใช้ในการออกแบบส่วนประสานงานกราฟฟิกผู้ใช้รวมถึงสร้างแอปพลิเคชันบนอุปกรณ์ต่างๆ สามารถทำงานได้หลายระบบปฏิบัติการ เช่น Windows Mac OS และ Linux เป็นต้น หรือสามารถเรียกโปรแกรมนี้ได้ว่าเป็น Cross platform นั่นหมายความว่าผู้ใช้งานไม่จำเป็นต้องทำการแก้ไขโปรแกรมเมื่อต้องการนำส่วนประสานงานกราฟฟิกผู้ใช้ที่ออกแบบได้ไปใช้งานบนอุปกรณ์อื่น เพียงแค่ลง Compiler ก็สามารถนำงานที่ออกแบบไปใช้ในอุปกรณ์ที่ใช้ระบบปฏิบัติการต่างกันได้ โดยโปรแกรม QT จะมี API และ Library ต่าง ๆ ที่เขียนด้วยภาษา C++ ทำให้สามารถพัฒนาแอพพลิเคชันแบบส่วนประสานงานกราฟฟิกผู้ใช้ ทั้งนี้ยังสนับสนุนการพัฒนาทั้ง C++ Java Python Perl Pascal และ PHP ซึ่งก็ขึ้นอยู่กับผู้พัฒนาว่าจะเลือกใช้ภาษาใดในการพัฒนา



รูปที่ 2.31 หน้าต่างการออกแบบในโปรแกรม QT

จากรูปที่ 2.31 เป็นหน้าต่างที่ใช้สำหรับออกแบบและพัฒนาส่วนประสานงานกราฟฟิกผู้ใช้ประกอบไปด้วยส่วนที่ใช้ออกแบบบล็อกการทำงาน และการเขียนคำสั่งการทำงานของหน้าต่างที่ออกแบบขึ้น หลังจากที่ผู้ใช้งานออกแบบเสร็จแล้ว สามารถบันทึกโปรแกรมแล้วนำไปใช้งานจริงได้ต่อไป

โปรแกรม QT บนอุปกรณ์ Raspberry Pi

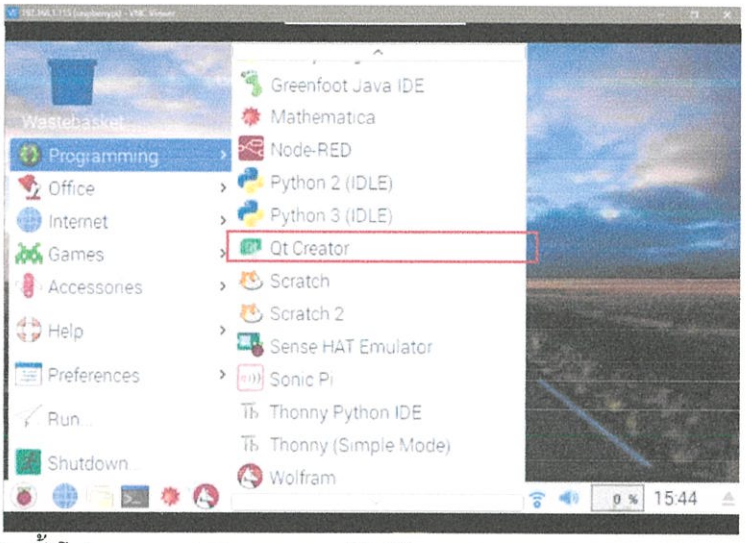
เพื่อความสะดวกในการพัฒนาโปรแกรมและการทำงานในอนาคต ผู้ใช้งานสามารถติดตั้งโปรแกรม QT บนอุปกรณ์ Raspberry Pi เพื่อรองรับการถ่ายโอนข้อมูลโปรแกรมแสดงผลส่วน

ประสานงานกราฟฟิกผู้ใช้จากระบบปฏิบัติการ Windows ไปยังระบบปฏิบัติการ Linux ซึ่งเป็นระบบปฏิบัติการที่จะนำไปใช้จริงบนอุปกรณ์ Raspberry Pi ทั้งนี้สามารถนำโปรแกรมที่ออกแบบบนระบบปฏิบัติการ Windows มาทำงานผ่าน Complier บนระบบปฏิบัติการ Linux หรือทำการแก้ไขหน้าต่างการแสดงผลเช่นเดียวกับการออกแบบในระบบปฏิบัติการณ์ Windows ได้อีกด้วย โดยขั้นตอนการติดตั้งโปรแกรม QT บนอุปกรณ์ Raspberry Pi จะติดตั้งโดยใช้ Terminal หรือหน้าต่าง Command แล้วพิมพ์คำสั่งดังนี้

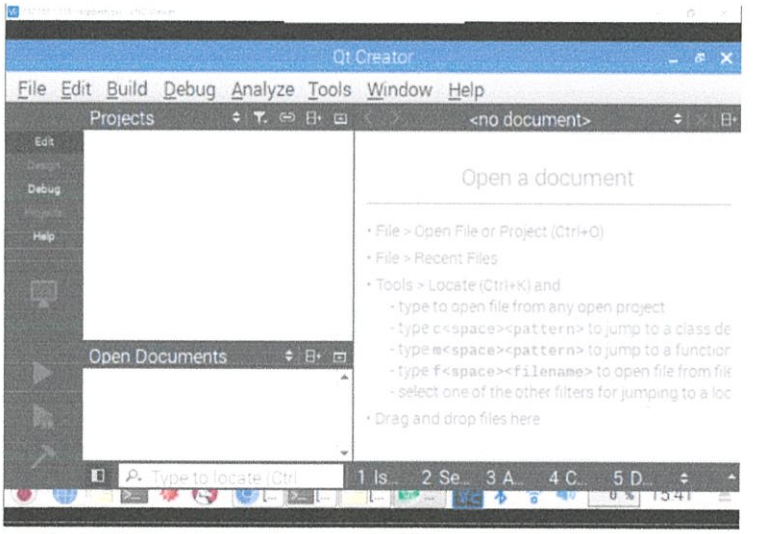
1. เชื่อมต่ออินเตอร์เน็ตกับอุปกรณ์ Raspberry Pi
2. เปิด Terminal หรือหน้าต่าง Command
3. พิมพ์คำสั่ง sudo apt-get update เพื่ออัพเดทซอฟท์แวร์จาก Repository รอจนการดำเนินการเสร็จสิ้น
4. พิมพ์คำสั่ง sudo apt-get install qt5-default เพื่อติดตั้ง Qt 5 packet
5. พิมพ์คำสั่ง sudo apt-get install qtcreator เพื่อติดตั้ง Qt Creator
6. พิมพ์คำสั่ง sudo apt-get install build-essential g++ เพื่อติดตั้ง build-essential และ g++
7. พิมพ์คำสั่ง sudo apt-get install libqt5serialport5-dev เพื่อติดตั้ง Library QSerialPort

เนื่องจาก Qt creator บางเวอร์ชันมีปัญหาเกี่ยวกับหน้า Welcome กับ Jessie ให้เปิดโปรแกรมในครั้งแรกโดย โดยใช้คำสั่ง qtcreator -noload Welcome (หากไม่มีปัญหาในหน้า Welcome ไม่ต้องใช้คำสั่งนี้เมื่อเปิดโปรแกรม)

หลังจากการติดตั้งโปรแกรม QT บนระบบปฏิบัติการ Linux แบบ Raspbian Desktop แล้วจะปรากฏชื่อโปรแกรมในแถบ Programming แสดงดังรูปที่ 2.32 และแสดงหน้าต่างของโปรแกรมดังรูปที่ 2.33



รูปที่ 2.32 การติดตั้งโปรแกรม QT บนระบบปฏิบัติการ Linux แบบ Raspbian Desktop ที่เสร็จสมบูรณ์



รูปที่ 2.33 หน้าต่างโปรแกรม QT บนระบบปฏิบัติการ Linux แบบ Raspbian Desktop

จากรูปที่ 2.33 ภายในหน้าต่างของโปรแกรม QT จะปรากฏแถบเครื่องมือต่างๆ โดยเมื่อทำการสร้างหรือนำเข้า Project แล้ว จะใช้แถบเครื่องมือ Edit ในการเขียนหรือแก้ไขชุดคำสั่งของตัวโปรแกรม ใช้แถบเครื่องมือ Design ในการออกแบบหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้ แถบคำสั่ง

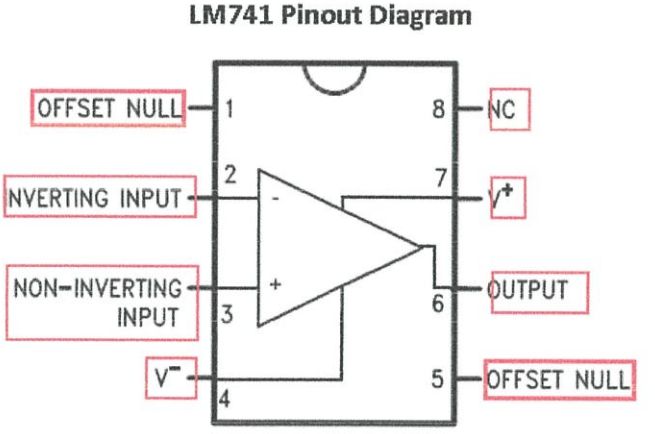
Debug สำหรับตรวจสอบความผิดพลาดของชุดคำสั่งที่เขียนว่าถูกต้องตามไวยากรณ์ของภาษานั้นๆ หรือไม่ และใช้แถบเครื่องมือ Run เพื่อทดสอบโปรแกรม โดยหลังจากติดตั้งโปรแกรม QT บนระบบปฏิบัติการ Linux แบบ Raspbian Desktop แล้ว จะสามารถทำการออกแบบและแก้ไขโปรแกรมส่วนประสานงานกราฟฟิกผู้ใช้ได้เช่นเดียวกับระบบปฏิบัติการ Windows ซึ่งจะรองรับการถ่ายโอนข้อมูลของโปรแกรมและช่วยอำนวยความสะดวกต่อการทำงานในอนาคต

2.10 หลักการวงจรตัดแบตเตอรี

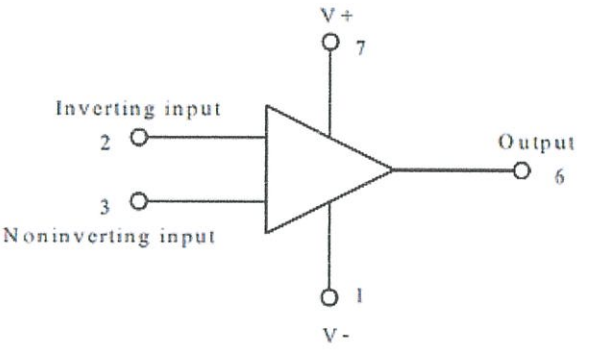
วงจรตัดแบตเตอรีนี้ จะเป็นการตัดแหล่งจ่ายไม่ป้อนให้อุปกรณ์สถานีจลน์ เพื่อถนอมแบตเตอรีไม่ให้เสื่อมสภาพเนื่องมาจากการคายประจุจนทำให้แรงดันของแบตเตอรีต่ำกว่ากำหนด จึงนำความรู้ของทางด้านการต่อวงจรอิเล็กทรอนิกส์ โดยสามารถอธิบายหลักการการทำงานได้ดังหัวข้อที่ 2.10.1 และ 2.10.2

2.10.1 หลักการ ออปแอมป์

ออปแอมป์ (OP-AMP เป็นชื่อย่อของ Operational amplifier) เป็นอุปกรณ์อิเล็กทรอนิกส์จำพวก ไอ.ซี. (Integrator Circuit) เป็นอุปกรณ์ที่รวบรวมอุปกรณ์จำพวกพาสซีฟ (Passive) และแอกตีฟ (Active) ไว้ภายในตัวเดียวกัน ซึ่งในปัจจุบันออปแอมป์ ถูกนำมาใช้งานหลากหลาย ทั้งในวงจรขยายสัญญาณ วงจรออสซิลเลเตอร์วงจรกรองความถี่สูง และความถี่ต่ำ วงจรเปรียบเทียบสัญญาณ และอื่น ๆ ซึ่งมีโครงสร้าง และสัญลักษณ์ออปแอมป์ แสดงดังรูปที่ 2.34 ถึง 2.35



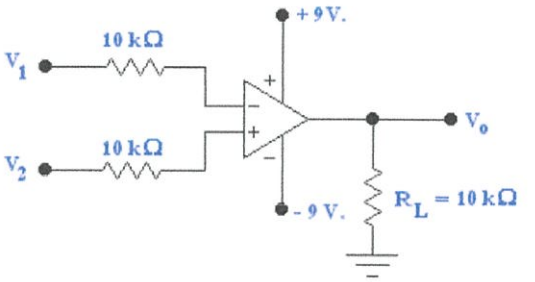
รูปที่ 2.34 โครงสร้างออปแอมป์ [29]



รูปที่ 2.35 สัญลักษณ์ออปแอมป์ [30]

รูปที่ 2.34 เป็นโครงสร้างของออปแอมป์เบอร์ 741 เป็น ไอ.ซี. 8 ขา ขาที่ 1 และขาที่ 5 จะเป็นขา OFFSET NUL ทำหน้าที่ปรับ Output offset voltage ให้เป็นศูนย์โวลต์ ขา 2 เป็นอินพุตลบ หรืออินพุตแบบกลับเฟส เรียกทับศัพท์ว่า อินเวอร์ติ้งอินพุต (Inverting input) หมายความว่าถ้าสัญญาณอินพุตเข้ามาที่ขานี้เป็นบวกมันจะถูกกลับเฟสให้เป็นลบ ถ้าสัญญาณอินพุตที่เข้ามาเป็นลบมันจะถูกกลับเฟสให้เป็นบวก สำหรับขาที่ 3 เรียกว่าอินพุตบวกหรืออินพุตแบบไม่กลับเฟส เรียกทับศัพท์ว่า นอนอินเวอร์ติ้งอินพุต (Non – inverting input) หมายความว่า สัญญาณอินพุตที่เข้ามาที่ขานี้เป็นสัญญาณแบบใดก็จะได้เอาต์พุตแบบนั้น ส่วนขาที่ 4 และ 7 จะถูกต่อเข้ากับแหล่งจ่ายไฟเลี้ยง ไอ.ซี. แบบ $\pm$ โดยขา 4 ถูกต่อกับแหล่งจ่ายไฟเลี้ยง – และขา 7 ต่อกับแหล่งจ่ายไฟเลี้ยง + สาหรับเอาต์พุตของวงจรจะอยู่ที่ขา 6 ส่วนขา 8 ไม่ได้ต่อใช้งานใด

วงจรเปรียบเทียบแรงดัน (Comparator) ลักษณะของวงจรเรียกว่าแบบลูปเปิดหรือเปิดลูป (Opened – loop) ซึ่งนั่นก็คือไม่มี การดึงสัญญาณที่ได้ทางเอาต์พุตกลับมาป้อนให้กับอินพุต โดยมีตัวอย่างแสดงการทำงานดังรูปที่ 2.36

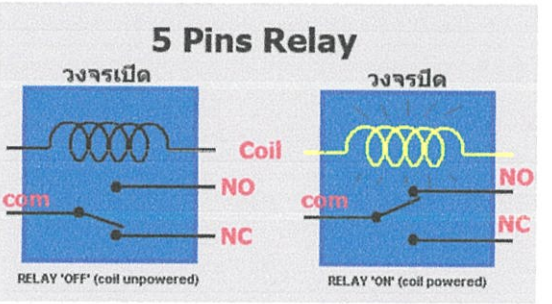


รูปที่ 2.36 ตัวอย่างวงจรเปรียบเทียบแรงดัน [31]

ในรูปที่ 2.36 มีหลักการทำงานของวงจร คือ สัญญาณด้านอินพุต V_1 และ V_2 ที่ป้อนเข้าจะมีการเปรียบเทียบค่าว่า สัญญาณใดมีค่ามากกว่า จากนั้นจึงจะส่งผลไปยังเอาต์พุตของวงจรว่ามีค่าออกมาเป็นบวกหรือลบ และจะมีค่าที่ตำแหน่งสูงสุดที่ไม่เกิน ±90% ของแหล่งจ่ายไฟเลี้ยงวงจรที่จ่ายให้กับออปแอมป์ ค่าแรงดันเอาต์พุตที่ 90% นี้จะเรียกว่า V_{sat}

2.10.2 หลักการรีเลย์

เป็นอุปกรณ์ที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานแม่เหล็ก เพื่อใช้ในการดึงดูดหน้าสัมผัสของคอนแทคให้เปลี่ยนสถานะ โดยการป้อนกระแสไฟฟ้าให้กับขดลวด เพื่อทำการปิดหรือเปิดหน้าสัมผัสคล้ายกับสวิตช์อิเล็กทรอนิกส์ เราสามารถนำไปประยุกต์ใช้กับการควบคุมวงจรต่างๆ ในงานทางอิเล็กทรอนิกส์ได้



รูปที่ 2.37 โครงสร้างของรีเลย์ [32]

จากรูปที่ 2.37 แสดงโครงสร้าง และการทำงานของรีเลย์ โดยจะแบ่งเป็น 4 ส่วนหลักๆ คือ

1. ขดลวด (Coil) เป็นส่วนที่ใช้ควบคุมการทำงานของรีเลย์ ควบคุมการเปิดหรือปิดของวงจร
2. ขา Com (Common) เป็นขาที่ต่อแหล่งจ่ายไฟ
3. ขา NO (Nomal Open) เป็นขาที่เชื่อมกับขา Com เมื่อไม่มีการจ่ายไฟให้รีเลย์
4. ขา NC (Nomal Close) ขาที่เชื่อมกับขา Com ในขณะที่มีการจ่ายไฟให้รีเลย์แล้ว โดยที่ ขา NO และ NC จะไปเชื่อมกับอีกข้างหนึ่งของวงจรภายนอก

2.11 Raspberry Pi 3

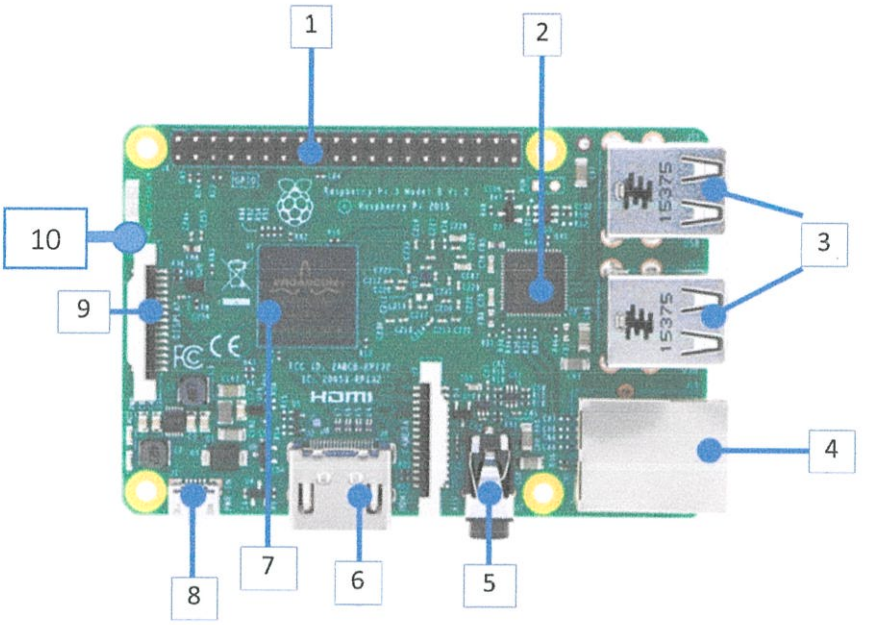
Raspberry Pi คือ บอร์ดคอมพิวเตอร์ขนาดเล็กสามารถเชื่อมต่อกับจอมอนิเตอร์ คีย์บอร์ด และเมาส์ได้ สามารถนำมาประยุกต์ใช้ในการทำปริญญานิพนธ์ด้วยอิเล็กทรอนิกส์ การเขียนโปรแกรม

หรือเป็นเครื่องคอมพิวเตอร์ตั้งโต๊ะขนาดเล็ก มีประสิทธิภาพในการทำงาน Spreadsheet Word Processing ส่งอีเมล เล่นเกมส์ หรือเล่นไฟล์วิดีโอความละเอียดสูง (High-Definition) รองรับระบบปฏิบัติการลินุกซ์ (Linux Operating System) ได้หลายระบบ เช่น Raspbian (Debian) Pidora (Fedora) และ Arch Linux เป็นต้น โดยทำการติดตั้งบน SD Card โดย Raspberry Pi ถูกออกแบบมาให้มี CPU GPU และ RAM อยู่ภายในชิปเดียวกัน มีจุดเชื่อมต่อ GPIO ให้ผู้ใช้สามารถนำไปใช้ร่วมกับอุปกรณ์อิเล็กทรอนิกส์อื่นๆ ได้ ในปัจจุบันบอร์ด Raspberry Pi มีอยู่ 2 โมเดล คือ โมเดล A และโมเดล B ซึ่งทั้ง 2 โมเดล มีคุณสมบัติทางเทคนิคที่ใกล้เคียงกันแตกต่างกันเพียงบางส่วน ในปริญญานิพนธ์ฉบับนี้จะใช้ Raspberry Pi 3 โมเดล B มีรายละเอียดดังตารางที่ 2.2 และแสดงโครงสร้างของบอร์ด Raspberry Pi ดังรูปที่ 2.38

ตารางที่ 2.3 คุณสมบัติของ Raspberry Pi 3 โมเดล B

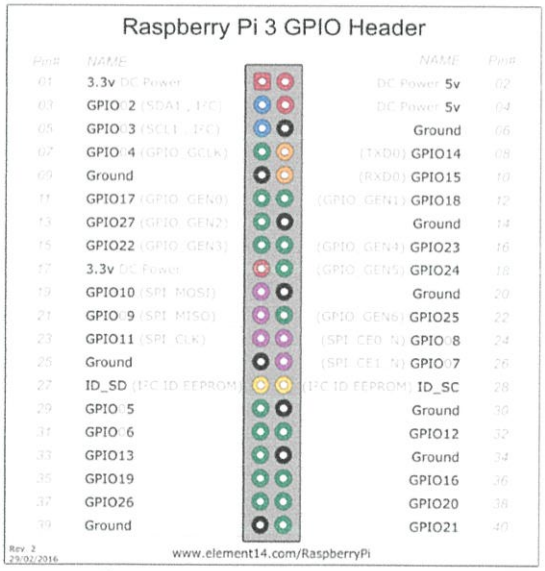
System on a chip (Soc)	Broad BCM2837
	802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
CPU	1.2GHz Quad core ARM Cortex-A53
GPU	Dual VideoCorev IV® Multimedia Co-Processor OpenGL ES 2.0 (24 GFLOPS) 1080p 30 H.264 high-profile decoder
Memory (SDRAM)	1 GB LPDDR2
USB 2.0 Ports	4 (via the build in integrated 3-port USB hub)
Video Input	Composite RCA (PAL and NTSC), HDMI (rev 1.3 & 1.4), raw LCD Panels via DSI 14 HDMI resolutions from 640x350 to 1920x1200 plus various PAL and NTSC standards (มีทั้งสองแบบ คือ แบบ RCA และแบบ HDMI)

Audio Outputs	3.5 mm jack, HDMI, and as of revision 2 boards, I^2S audio (also potentially for audio input)
Onboard storage	SD/ MMC/ SDIO card slot (3.3V card power support only)
Onboard network	10/100 Ethernet (8P8C) USB adapter on the third port of the USB hub
Low-level peripherals	8 x GPIO, UART, I^2C Bus, SPI Bus with two chip selects, I^2S audio +3.3V, +5V, Ground
Power ratings	700 mA (3.5 W)
Power source	5 Volt via Micro USB or GPIO header
Size	85.60 mm x 53. Mm (3.370 inch x 2.125 inch)
Weight	45 g. (1.6 oz.)
GPIO Connector	40-pin 2.54 mm expansion header: 2x20 strip Providing 27 GPIO pins as +3.3 V,+5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane



รูปที่ 2.38 ส่วนประกอบของบอร์ด Raspberry Pi 3 (Model B) [33]

หมายเลข 1 พอร์ต GPIO แสดงลักษณะของทุกๆ PIN แสดงดังรูปที่ 2.39



รูปที่ 2.39 พอร์ต GPIO ของ Raspberry Pi 3 Model B [34]

หมายเลข 2 ชิปควบคุม LAN (LAN Controller)

หมายเลข 3 พอร์ต USB 2.0 จำนวน 4 พอร์ต

หมายเลข 4 พอร์ต RJ-45 Ethernet LAN 10/100Mbps

หมายเลข 5 พอร์ตเชื่อมต่อสัญญาณภาพออกแบบ RCA ตัวอย่างแสดงดังรูปที่ 2.40

รูปที่ 2.40 สาย RCA [35]

หมายเลข 6 พอร์ต HDMI สำหรับเชื่อมต่อสัญญาณภาพและเสียง

หมายเลข 7 ชิพ Broadcom BCM2837

หมายเลข 8 พอร์ต Micro USB Power สำหรับเป็นไฟเลี้ยงวงจรบอร์ด Raspberry Pi

หมายเลข 9 พอร์ต DSI (Display Serial Interface) ใช้สำหรับต่อจอแสดงผล เช่น จอแสดงผลแบบ TFT Touch Screen เป็นต้น

หมายเลข 10 ช่องเสียบ SD Card อยู่บริเวณด้านล่างของบอร์ด

โปรแกรมภาษาของ Raspberry Pi จะใช้ภาษา Python เนื่องจากตัวโครงสร้างและ Syntax ของภาษาเข้าใจง่ายถูกพัฒนาขึ้นมาโดยไม่ยึดแพลตฟอร์ม สามารถเขียนโปรแกรมประมวลผลด้วยภาษา Python ได้บนหลายระบบซึ่งเป็นรูปแบบของภาษาที่ถูกสร้างมาจากภาษาซี การประมวลผลแบบอินเทอร์พรีเตอร์ จะประมวลผลไปทีละบรรทัดและปฏิบัติตามคำสั่งที่ได้รับจากบนลงล่าง อีกทั้งยังเป็น Open Source เป็น Dynamic typing คือ สามารถเปลี่ยนชนิดข้อมูลได้ง่ายและสะดวก โดยทำการจัดการหน่วยความจำอย่างอัตโนมัติ สามารถจัดการพื้นที่หน่วยความจำที่ไม่ต่อเนื่องให้ทำงานอย่างมีประสิทธิภาพ จะเห็นได้ว่าภาษา Python สามารถนำมาประยุกต์ใช้ได้กับงานทางวิศวกรรมศาสตร์ ได้เป็นอย่างดีทำให้งานที่ได้มีประสิทธิภาพ

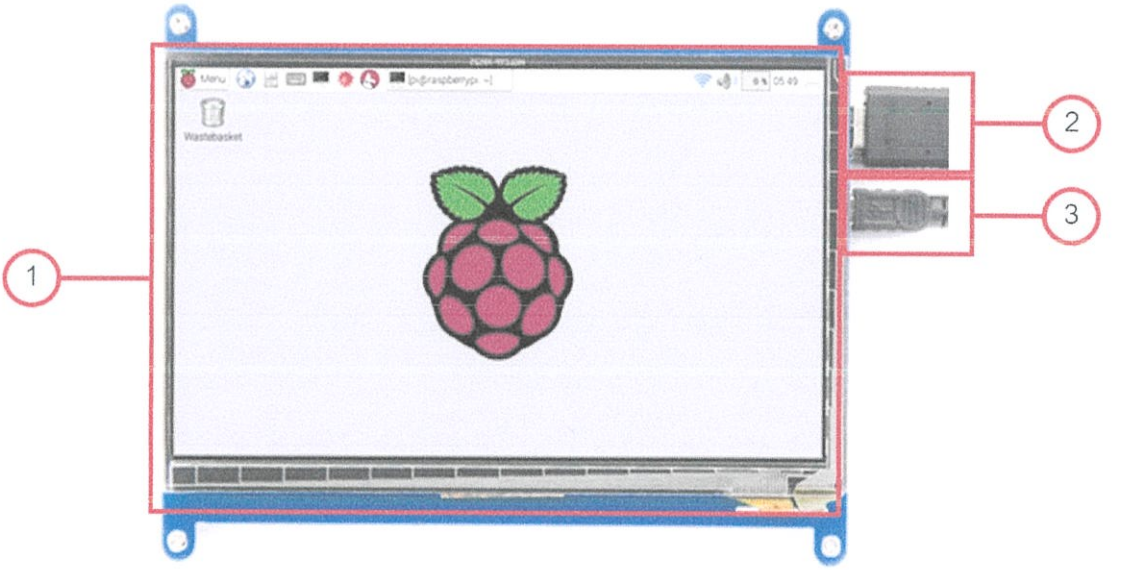
2.12 จอแสดงผล LCD แบบสัมผัส ยี่ห้อ Cheer รุ่น Cheerforbuy11-ZC659600

จอแสดงผลแบบ LCD ยี่ห้อ Cheer รุ่น Cheerforbuy11-ZC659600 เป็นอุปกรณ์การแสดงผลแบบจอสัมผัสที่รองรับการเชื่อมต่อกับอุปกรณ์ Raspberry Pi ผ่านทางพอร์ต HDMI (High-

Definition Multimedia Interface) บริโภคพลังงานไฟฟ้าขนาด 5 โวลต์ โดยเชื่อมต่อแรงดันไฟฟ้าผ่านทางพอร์ต USB (Universal Serial Bus) ของอุปกรณ์ Raspberry Pi มีคุณสมบัติในการใช้งาน คือ

- เป็นหน้าจอแสดงผลแบบสัมผัส ขนาด 7 นิ้ว (164.9 มม. x 124.27 มม.)
- ความละเอียด 1024 x 600 pixel
- ใช้ไอซี TDB ในการประมวลผล
- รองรับรูปแบบการแสดงสีแบบ RGB (Red Green Blue)
- ระบบสัมผัสรองรับการสัมผัสพร้อมกัน 5 จุด
- ใช้ Firmware เวอร์ชัน Rev2.1 ในระบบปฏิบัติการ
- ใช้วัสดุ TFT (Thin Film Transistor) ในการผลิตหน้าจอ ทำให้มีการตอบสนองและประมวลผลได้อย่างรวดเร็ว ภาพที่แสดงผลจึงมีความคมชัดและสว่างสดใส

โดยรูปภาพของหน้าจอแสดงผล ยี่ห้อ Cheer รุ่น Cheerforbuy11-ZC659600 แสดงส่วนประกอบดังภาพที่ 2.41



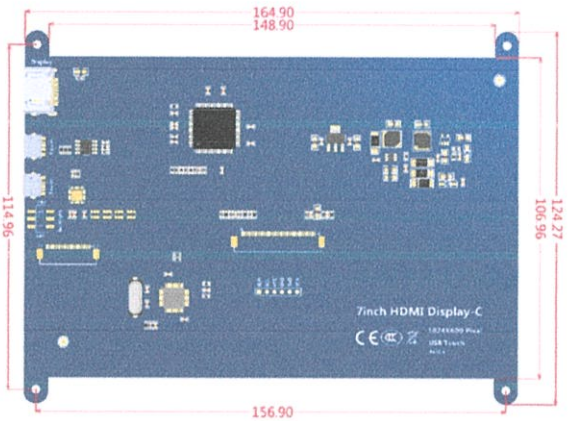
รูปที่ 2.41 ส่วนประกอบของหน้าจอแสดงผล [36]

หมายเลข 1 หน้าจอแสดงผล

หมายเลข 2 พอร์ต HDMI ที่ใช้เชื่อมต่อการแสดงผลจากอุปกรณ์ Raspberry Pi

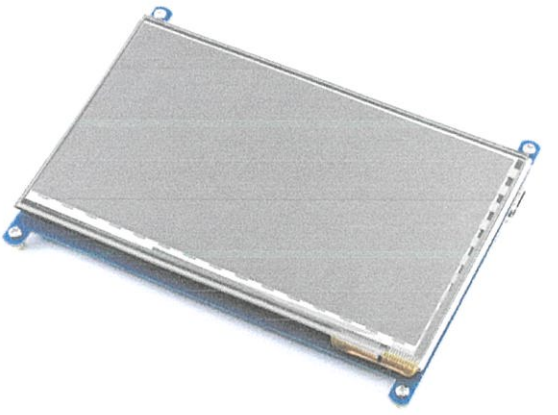
หมายเลข 3 พอร์ตกระแสไฟฟ้าขาเข้าขนาด 5 โวลต์

ในส่วนด้านหลังของหน้าจอจะทำการติดตั้งชุดอุปกรณ์ประมวลผลสำหรับการแสดงผล และปิดทับด้วยวัสดุกันกระแทกอีกหนึ่งชั้น โดยอุปกรณ์ประมวลผลสำหรับจอแสดงผลและการติดตั้งวัสดุกันกระแทกบริเวณด้านหลังของจอแสดงผล แสดงดังรูปที่ 2.42 และ 2.43 ตามลำดับ



รูปที่ 2.42 ชุดอุปกรณ์ประมวลผลของจอแสดงผล [36]

จากรูปที่ 2.42 แสดงชุดอุปกรณ์ประมวลผลของจอแสดงผล มีลักษณะเป็นแผ่นวงจรขนาดกว้าง 124.27 มม. x 164.90 มม. ประกอบไปด้วยอุปกรณ์ประมวลผลในการแสดงภาพ พอร์ตการเชื่อมต่อต่างๆ เช่น พอร์ตไฟฟ้าขาเข้า และพอร์ตการเชื่อมต่อ HDMI รวมถึงมีช่องสำหรับยึดติดหน้าจอกับอุปกรณ์อื่นๆซึ่งชุดอุปกรณ์ประมวลผลนี้จะติดตั้งอยู่บริเวณด้านหลังของหน้าจอ LCD



รูปที่ 2.43 อุปกรณ์กันกระแทกด้านหลังของจอแสดงผล [36]

จากรูปที่ 2.43 แสดงอุปกรณ์กันกระแทกซึ่งถูกติดตั้งอยู่ด้านหลังสุดของจอแสดงผล เพื่อป้องกันความเสียหายที่อาจเกิดขึ้นกับชุดอุปกรณ์ประมวลผล ดังรูปที่ 2.42 เช่น ความชื้น อุณหภูมิ หรือการกระแทก อีกทั้งยังทำให้ง่ายต่อการติดตั้งจอแสดงผลบนอุปกรณ์อื่นๆอีกด้วย

2.13 หลักการมอเตอร์ [37]

มอเตอร์ หมายถึง เครื่องมือที่เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานกล มีลักษณะคล้ายกับเครื่องกำเนิดไฟฟ้ากระแสตรง แต่จะแตกต่างกันในส่วนของการใช้งาน คือ เครื่องกำเนิดไฟฟ้าทั่วไปจะเปลี่ยนพลังงานกลเป็นพลังงานไฟฟ้า

การทำงานของมอเตอร์ เมื่อมีกระแสไหลในขดลวดตัวนำที่พันอยู่บนแกนอาร์เมเจอร์ จะเกิดเส้นแรงแม่เหล็กรอบๆ ตัวนำ แล้วเกิดปฏิกิริยากับเส้นแรงแม่เหล็กจากขั้วแม่เหล็กของมอเตอร์ ทำให้เกิดแรงผลักบนตัวนำแล้วหมุนอาร์เมเจอร์ได้ สมมุติให้ขดลวดตัวนำวางห่างจากจุดศูนย์กลางเป็นระยะ r กำหนดให้กระแสไฟฟ้าไหลเข้าขดลวดที่ปลาย A และไหลออกที่ปลาย B เมื่อกระแสไฟฟ้าไหลผ่านขดลวด ส่งผลให้ปริมาณของเส้นแรงแม่เหล็กจะมีจำนวนมากที่ด้านบนของปลาย A จึงทำให้เกิดแรง F1 กดตัวนำ A ลงด้านล่าง ขณะเดียวกันที่ปลาย B จะมีเส้นแรงแม่เหล็กปริมาณมากที่ด้านหน้าทำให้เกิดแรง F2 ดันให้ตัวนำ B เคลื่อนที่ด้านบน ทำให้อาร์เมเจอร์ของมอเตอร์เกิดการเคลื่อนที่ไปได้

2.13.1 คุณสมบัติทั่วไป

เป็นคุณสมบัติประจำตัวเกี่ยวกับลักษณะโครงสร้าง ลักษณะงาน ลักษณะของวงจรเช่น คุณสมบัติของมอเตอร์อนุกรมประกอบด้วยขดลวดสนามแม่เหล็กที่มีความต้านทานต่ำ (พันด้วยลวดทองแดงเส้นใหญ่น้อยรอบแกนขั้วแม่เหล็ก) ต่อเป็นอนุกรมกับอาร์เมเจอร์และต่อโดยตรงกับแรงดันเมน ลักษณะวงจร A1 และ A2 เป็นอาร์เมเจอร์ต่อเป็นอนุกรมกับขดลวดสนามแม่เหล็กชุดอนุกรม D1 และ D2 และต่อโดยตรงกับสายเมน L+, L- ลักษณะสนามแม่เหล็กทำให้ความเร็วสูงเมื่อโหลดลง จึงเป็นมอเตอร์ที่หมุนไม่คงที่ความเร็วเปลี่ยนแปลงไปตามโหลดจะเหมาะสมอย่างยิ่งที่จะใช้เป็นมอเตอร์ขับเคลื่อนล้อรถ

2.13.2 คุณสมบัติทางเทคนิค

เป็นคุณสมบัติประจำเครื่องกลไฟฟ้าที่ให้รายละเอียดเกี่ยวกับการทดสอบและวัดด้วยเครื่องวัดได้ด้วยวิธีทดลองในห้องปฏิบัติการทดลอง เช่น ประสิทธิภาพการหมุนขับโหลด (Speed load Curves หรือ Speed/load Characteristic) ที่แสดงให้เห็นความสัมพันธ์ระหว่างความเร็วรอบกับกระแสมอเตอร์

n คือความเร็วรอบให้อยู่บนแกน Y หรือ Ordinate และ

I_a คือกระแสอาร์เมเจอร์ให้อยู่บนแกน X หรือ abscissae)

อาจให้แสดงความสัมพันธ์ระหว่างความเร็วรอบ (n เป็น ordinate หรือ แกน Y) กับทอร์ค หรือกำลังที่หมุนขับงาน

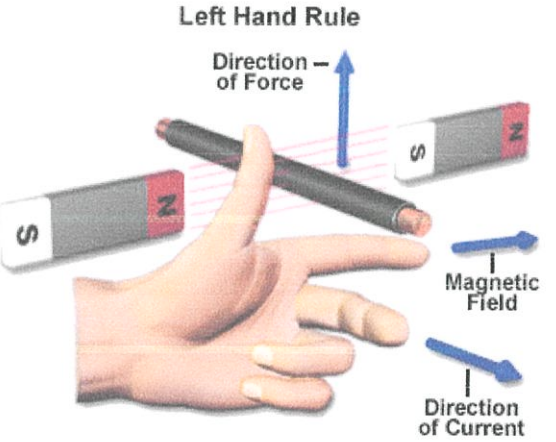
T คือทอร์ค

P คือกำลังวัตต์หรือกิโลวัตต์ ให้อยู่บนแกน x หรือ abscissae

จุดประสงค์เพื่อต้องการแสดงให้เห็นถึงความเปลี่ยนแปลงของความเร็วรอบของมอเตอร์ที่หมุนขับโหลดว่าจะมีการเปลี่ยนแปลงไปอย่างไรเมื่อโหลดเปลี่ยนแปลงไป

กฎมือซ้ายสำหรับมอเตอร์

เป็นกฎที่ใช้หาทิศทางการเคลื่อนที่ของมอเตอร์ในสนามแม่เหล็ก กฎนี้ได้นำไปใช้แบบเดียวกันกับกฎมือขวาของเครื่องกำเนิดไฟฟ้า แตกต่างกันตรงที่การหาทิศทางการเคลื่อนที่ของมอเตอร์จะใช้มือซ้ายในการอธิบายความสัมพันธ์ของทิศทางกระแส ทิศทางการเคลื่อนที่ของตัว และทิศทางของสนามแม่เหล็ก โดยมีลักษณะดังรูปที่ 2.44 คือ ใช้นิ้วหัวแม่มือ นิ้วชี้และนิ้วกลางในการอธิบาย โดยให้แต่ละนิ้ววางตัวตั้งฉากซึ่งกันและกัน นิ้วชี้ ชี้ไปตามทิศทางของสนามแม่เหล็ก (Magnetic flux = B) นิ้วกลาง ชี้ไปตามทิศทางการไหลของกระแสไฟฟ้า (Current = I) และหัวแม่มือชี้ไปตามทิศทางของการเคลื่อนที่ของตัวนำ (Force = F)



รูปที่ 2.44 กฎมือซ้ายของมอเตอร์ [37]

แรงขับที่เกิดขึ้นเกิดจากตัวนำที่มีกระแสไฟฟ้าไหลผ่านในขณะที่มันวางอยู่ในสนามแม่เหล็กจะเป็นปฏิภาคโดยตรงกับความหนาแน่นของเส้นแรงแม่เหล็ก ความยาวของตัวนำและค่ากระแสไฟฟ้าที่ไหลผ่านตัวนำแรงที่เกิดขึ้นบนตัวนำ สามารถหาได้จากสมการที่ 2.38

$$F - BIL \tag{2.38}$$

เมื่อ

F -คือแรงที่เกิดขึ้นบนตัวนำหนึ่งตัว หน่วย นิวตัน

B คือความหนาแน่นสนามแม่เหล็ก หน่วย W_b / m^2

I คือกระแสที่ไหลในตัวนำ หน่วย แอมแปร์

L คือความยาวของตัวนำ หน่วย เมตร

แรงเคลื่อนไฟฟ้าต่อต้านเกิดขึ้นเนื่องจากเมื่อขดลวดตัวนำหมุนอยู่ในสนามแม่เหล็ก จะติดกับเส้นแรงแม่เหล็กแรงเคลื่อนไฟฟ้าที่เหนี่ยวขึ้นในขดลวดแรงเคลื่อนไฟฟ้าเหนี่ยวนำที่เกิดขึ้นจะมีทิศทางขัดขวางกับแรงเคลื่อนที่ไฟฟ้าที่จ่ายให้มอเตอร์จึงเรียกว่า"แรงเคลื่อนไฟฟ้าต่อต้าน" (Back e.m.f) ซึ่งจะเกิดขึ้นในขดลวดอาร์เมเจอร์เสมอ ดังนั้นแรงเคลื่อนไฟฟ้าที่มีผลต่อการใช้งานจริงๆ ในอาร์เมเจอร์จึงมีค่าเท่ากับแรงเคลื่อนไฟฟ้าที่จ่ายให้ลบด้วยแรงเคลื่อนไฟฟ้าต้านกลับสามารถเขียนสมการได้ดังสมการที่ 2.39 และ 2.40

$$V_t = I_a + E_b \tag{2.39}$$

$$I_a R_a = V_t - E_b \tag{2.40}$$

สมการแรงเคลื่อนไฟฟ้าดังสมการที่ 2.41

$$V_t = E_b + I_a R_a \tag{2.41}$$

กำลังที่เกิดในมอเตอร์แสดงดังสมการที่ 2.42 โดยนำเอาค่า I_a คูณตลอดเพื่อหา Power

$$V_t I_a = E_b I_a + I_a{}^2 R_a \tag{2.42}$$

เมื่อ

E_b คือแรงเคลื่อนไฟฟ้าที่ต้านกลับ หน่วยโวลต์ (V)

V_t คือแรงเคลื่อนไฟฟ้าที่จ่ายให้กับมอเตอร์ หน่วยโวลต์ (V)

I_a คือกระแสที่ไหลในอาร์เมเจอร์ หน่วยแอมแปร์ (A)

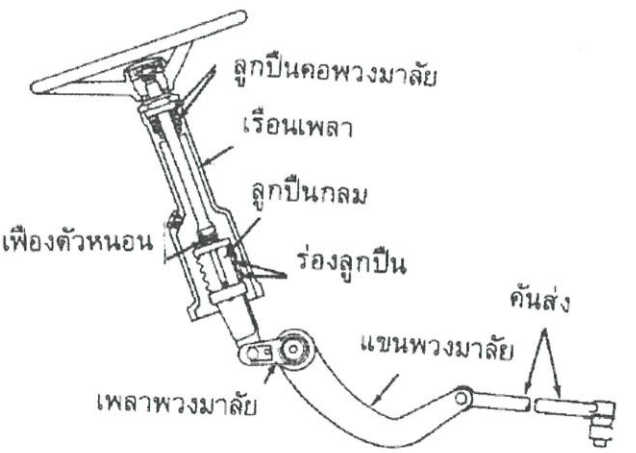
R_a คือความต้านทานของขดลวดในอาร์เมเจอร์ หน่วยโอมม์ (Ω)

2.14 หลักการบังคับเลี้ยว

ระบบบังคับเลี้ยว (Steering system) ทำหน้าที่เปลี่ยนทิศทางการเคลื่อนที่ของรถให้เคลื่อนที่ไปในทิศทางที่ต้องการ โดยล้อไม่ลื่นไถลขณะเลี้ยว และแรงกระแทกที่เกิดขึ้นระหว่างยางกับถนนที่ถ่ายทอดไปยังพวงมาลัย ระบบบังคับเลี้ยว แบ่งออกได้ 2 ระบบ คือ ระบบบังคับเลี้ยวแบบกลไก (Manual steering system) และระบบบังคับเลี้ยวแบบไฮดรอลิก (Power steering system)

2.14.1 ระบบบังคับเลี้ยวแบบกลไก

ระบบนี้ประกอบด้วยส่วนประกอบที่สำคัญ คือ พวงมาลัย เพลา พวงมาลัย ห้องเกียร์พวงมาลัย แขนพิทแมน คันส่งคันชัก คันล้อ และล้อกำลังจะถูกส่งไปยังส่วนประกอบต่าง ๆ ตามลำดับเพื่อบังคับให้ล้อเลี้ยวตามทิศทางที่ต้องการ แสดงส่วนประกอบดังรูปที่ 2.45



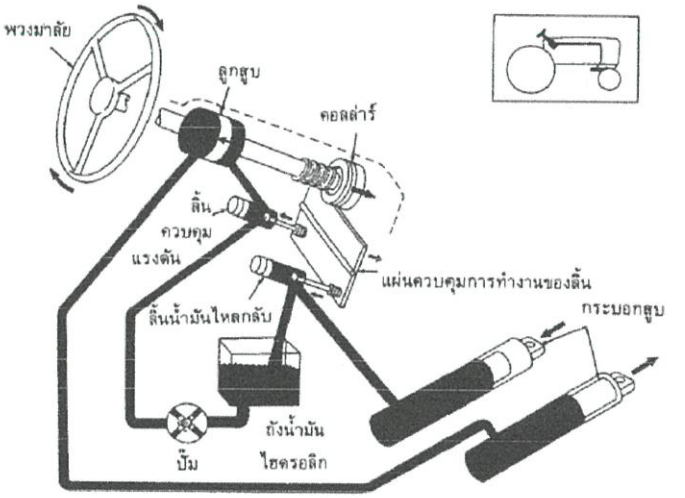
รูปที่ 2.45 ระบบบังคับเลี้ยวแบบกลไก [38]

จากรูปที่ 2.45 ผู้ใช้งานสามารถควบคุมทิศทางของรถได้โดยการเปลี่ยนทิศทางของพวงมาลัย แรงจากการบิดพวงมาลัยจะถูกขับไปยังเพลาพวงมาลัยเพื่อส่งแรงขับไปยังคันส่ง ซึ่งทำหน้าที่

ควบคุมทิศทางล้อรถ เนื่องจากการบังคับเลี้ยวแบบกลไกอาศัยการส่งกำลังผ่านชิ้นส่วนทางกลศาสตร์ที่เป็นวัตถุแข็งเกือบทั้งหมด ทำให้ต้องออกแรงบิดเพื่อควบคุมพวงมาลัยสูงพอสมควร

2.14.2 ระบบบังคับเลี้ยวแบบไฮดรอลิก

มีส่วนประกอบของระบบไฮดรอลิก และระบบบังคับเลี้ยว พวงมาลัยจะทำหน้าที่บังคับลิ้นเปลี่ยนทิศทางการไหลของน้ำมันไฮดรอลิก (Control valve) เพิ่มให้น้ำมันไปดันลูกสูบให้เคลื่อนที่ไปดันกลไกของระบบบังคับเลี้ยว ให้ล้อเลี้ยวตามทิศทางที่ต้องการได้ แสดงส่วนประกอบดังรูปที่ 2.46



รูปที่ 2.46 ระบบบังคับเลี้ยวแบบไฮดรอลิก [38]

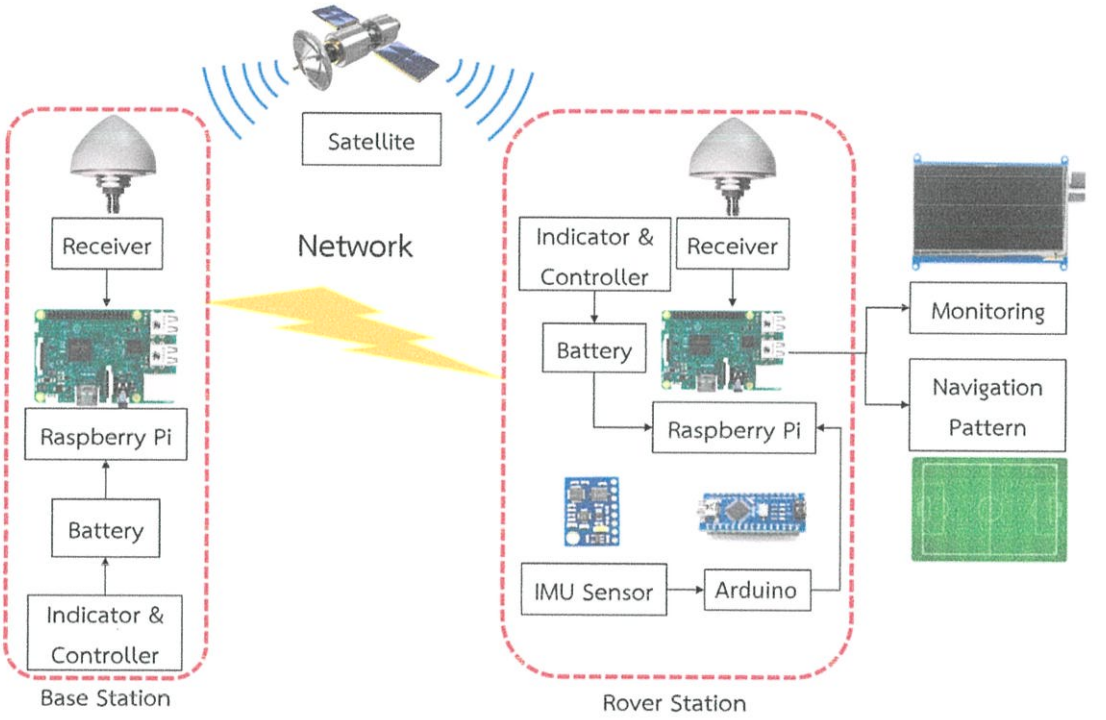
จากรูปที่ 2.46 มีขั้นตอนการทำงานที่คล้ายระบบบังคับเลี้ยวแบบกลไก แต่มีข้อแตกต่างกัน คือ ระบบบังคับเลี้ยวแบบไฮดรอลิก จะใช้แรงดันจากน้ำมันในการส่งแรงขับไปยังกระบอกสูบเพื่อควบคุมทิศทางของล้อรถ ซึ่งจะใช้แรงบิดที่พวงมาลัยน้อยกว่าแบบกลไก

จากระบบบังคับเลี้ยวที่กล่าวมา เพื่อให้การควบคุมทิศทางของรถง่ายต่อการออกแบบ ในปริญญานิพนธ์นี้จึงมุ่งการควบคุมที่พวงมาลัยเป็นหลัก โดยใช้การหมุนของมอเตอร์ไฟฟ้ากระแสตรงมาควบคุมทิศทางของพวงมาลัย ซึ่งง่ายต่อการควบคุม และไม่มีความซับซ้อนมาก

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบ



รูปที่ 3.1 บล็อกไดอะแกรมแสดงภาพรวมของปริญญานิพนธ์

จากรูปที่ 3.1 เป็นระบบการเฝ้าสังเกตและติดตามตำแหน่งของสถานีจลน์ (Rover) ให้มีความแม่นยำ เพื่อสร้างรูปแบบแนวการทำเกษตรกรรมบนแปลงเพาะปลูก (Navigation Pattern) โดยใช้เทคนิค RTK (Real Time Kinematic) มีหลักการคือสถานีฐาน (Base Station หรือ Base) และ Rover ต้องตั้งอยู่ในพื้นที่ที่สามารถมองเห็นดาวเทียมดวงเดียวกันได้ และมีการเชื่อมต่อกันผ่านโครงข่ายส่วนตัว (Network) เพื่อส่งค่าแก้จากสถานีฐานไปยังสถานีจลน์ให้ประมวลผลหาพิกัดของตนเองด้วย Raspberry Pi นอกจากนี้ใน Rover ยังมี IMU (Inertial Measurement Unit) ช่วยวัดความเอียงของสายอากาศที่ใช้รับสัญญาณจีเอ็นเอสเอส เพื่อหาระยะที่เกิดจากการเอียงดังกล่าวแล้วนำไปลดความผิดพลาดพิกัดสายอากาศขณะออกสำรวจในภูมิประเทศจริง และเพื่อให้ง่ายต่อการใช้งานจึงมีระบบการเฝ้าสังเกตแบบหน้าจอในการเฝ้าสังเกต และติดตามสถานีจลน์ ในส่วนของ

การบำรุงรักษา จะมีส่วนควบคุมการจ่ายไฟฟ้าให้กับอุปกรณ์ (Indicator & Controller) เพื่อยืดอายุการใช้งานของแบตเตอรีอีกด้วย

3.1.1 การออกแบบสถานีฐาน (Base Station)

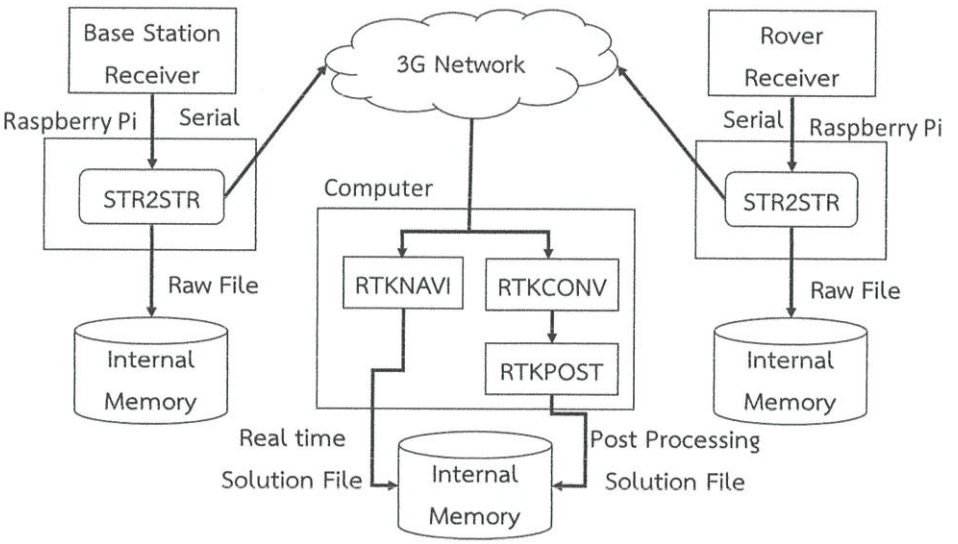
ในปริญญานิพนธ์นี้ จะมีการเก็บพิกัด GNSS เพื่อใช้ในการคำนวณตำแหน่งด้วยเทคนิค RTK โดยสถานีฐานจะรับค่าจากดาวเทียมแล้วทำการส่งค่าให้กับ สถานีจลน์ ในสถานีฐานนี้ได้ใช้ Raspberry Pi เป็นตัวประมวลผล และเก็บค่าพิกัด GNSS โดยมีการออกแบบสถานีฐาน

3.1.2 การออกแบบสถานีจลน์ (Rover)

สถานีจลน์ เป็นอุปกรณ์หลักที่ใช้ในการคำนวณตำแหน่งด้วยเทคนิค RTK โดยสถานีจลน์จะรับค่าจากดาวเทียมดวงเดียวกับสถานีฐานด้วย แต่สถานีจลน์จะต้องเคลื่อนที่ตลอดเวลา ซึ่งจะทำการเก็บค่า และนำมาเปรียบเทียบพิกัดกับสถานีฐาน ในตัวสถานีฐานจะมีการประมวลผลระยะผิดพลาดที่เกิดขึ้นเพื่อแสดงผลออกมาให้เราปรับแก้ไข

3.1.3 การคำนวณตำแหน่งด้วยเทคนิค RTK โดยใช้โปรแกรม RTKLIB

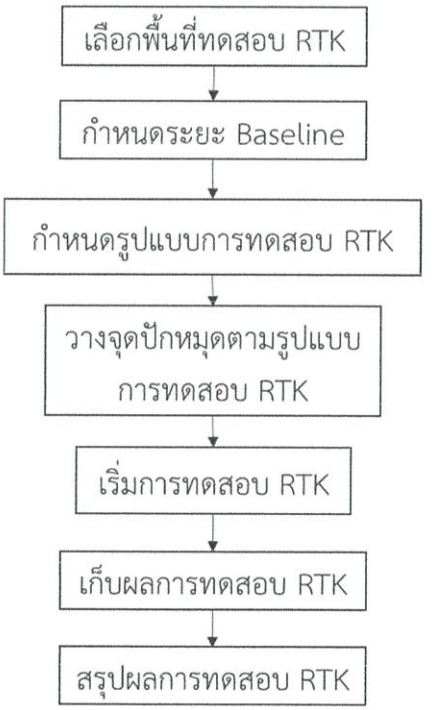
ในปริญญานิพนธ์นี้ จะใช้โปรแกรม RTKNAVI โปรแกรม STR2STR โปแกรม RTKCONV และโปรแกรม RTKPOST ซึ่งเป็นโปรแกรมย่อยในโปรแกรม RTKLIB โดยการใช้งานแต่ละโปรแกรมเพื่อการคำนวณตำแหน่งด้วยเทคนิค RTK สามารถแสดงรูปแบบการทำงานได้ดังรูปที่ 3.2



รูปที่ 3.2 การใช้งานโปรแกรม RTKNAVI STR2STR RTKCONV และ RTKPOST ในการคำนวณตำแหน่งด้วยเทคนิค RTK

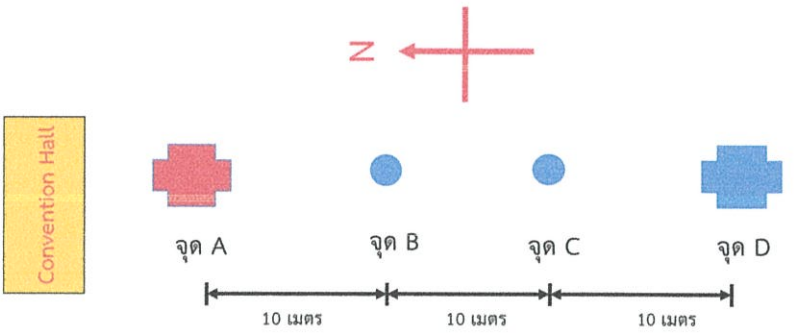
จากรูปที่ 3.2 การคำนวณตำแหน่งด้วยเทคนิค RTK จะใช้ต้องสั่งโปรแกรม STR2STR ใน Raspberry Pi ของสถานีฐานและสถานีจลน์ทำงาน เพื่อส่งข้อมูลดิบที่ได้รับจากเครื่องรับสัญญาณจีเอ็นเอสเอสไปยังเครือข่าย 3G โดยส่งไปครั้งละ 1 วินาที จากนั้นจึงให้คอมพิวเตอร์เชื่อมต่อกับเครือข่าย 3G ให้ดึงข้อมูลดังกล่าวมาประมวลผลในโปรแกรม RTKNAVI เพื่อดูผลการคำนวณตำแหน่งด้วยเทคนิค RTK ในคอมพิวเตอร์แบบเวลาจริง โดยดึงมาในอัตราที่เท่ากันกับสถานีฐานส่งมา และสามารถบันทึกผลการทดสอบไว้ในที่จัดเก็บภายในตัวเครื่องได้ หากต้องการจำลองการคำนวณตำแหน่งด้วยเทคนิค RTK ในภายหลังเพื่อวิเคราะห์ปัจจัยที่มีผลต่อความแม่นยำในการคำนวณตำแหน่งด้วยเทคนิค RTK สามารถใช้โปรแกรม RTKCONV เพื่อแปลงข้อมูลดิบจากโปรแกรม STR2STR ให้ได้รูปแบบที่เหมาะสม จากนั้นจึงเรียกใช้งานโปรแกรม RTKPOST มาประมวลผลข้อมูลที่แปลงรูปแบบแล้วไปวิเคราะห์ผลต่อไป

ขั้นตอนในการคำนวณตำแหน่งด้วยเทคนิค RTK สามารถแสดงได้ดังรูปที่ 3.3



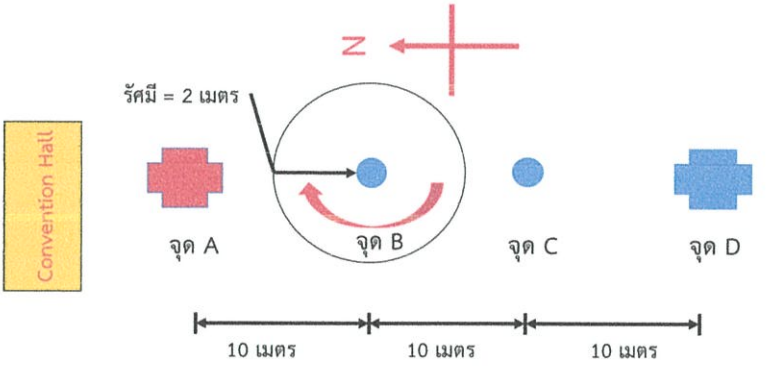
รูปที่ 3.3 ขั้นตอนการทดสอบระยะ Baseline ที่มีผลต่อความแม่นยำในการทำ RTK

รูปแบบที่ใช้คำนวณตำแหน่งด้วยเทคนิค RTK มีอยู่ด้วยกัน 2 แบบ ดังรูปที่ 3.4 และ 3.5



รูปที่ 3.4 การคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงเป็นระยะทาง 30 เมตร

จากรูปที่ 3.4 จะคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรง โดยการเดินตามแนวเส้นตรงซึ่งลากผ่านจุด A B C และ D เป็นระยะทาง 30 เมตร ซึ่งจะเดินจากจุด A ไปจุด D 1 รอบ และเดินย้อนกลับ 1 รอบ พร้อมกับเปลี่ยนระยะ Baseline (ระยะขจัดจากสถานีฐานถึงสถานีจลน์ มีหน่วยเป็นเมตร) ขณะเดินจะต้องไปหยุดที่จุดทั้ง 4 นานจุดละ 30 วินาทีเป็นอย่างน้อย เพื่อเก็บพิกัดไปเทียบกับพิกัดอ้างอิงจากการปักหมุดแล้ววิเคราะห์ผลของระยะ Baseline ที่มีต่อความแม่นยำ

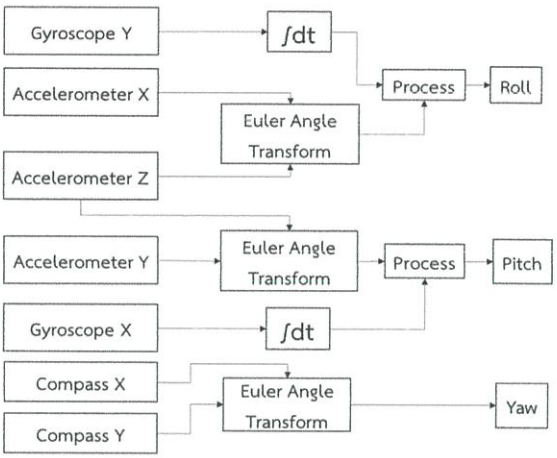


รูปที่ 3.5 การคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยรัศมี 2 เมตร

จากรูปที่ 3.5 จะคำนวณตำแหน่งด้วยเทคนิค RTK โดยการเดินเป็นวงกลมที่มีรัศมี 2 เมตร ในทิศตามเข็มนาฬิการอบจุด B อย่างน้อย 3 รอบ แล้วนำพิกัดจากการเดินไปหารัศมีในแต่ละเวลาเทียบกับรัศมีจริง เพื่อหาความแม่นยำในการคำนวณตำแหน่งด้วยเทคนิค RTK โดยจะเปลี่ยนระยะ Baseline เพื่อดูผลที่มีต่อความแม่นยำ

3.1.4 การอ่านค่ามุมที่แสดงการเคลื่อนที่ของวัตถุโดยใช้ IMU เซ็นเซอร์ รุ่น GY-85

จากหลักการทำงานของ Accelerometer Gyroscope และ Magnetometer ในหัวข้อที่ 2.2.1 ถึง 2.2.3 สามารถแสดงการหาค่ามุมที่เอียงไปของวัตถุได้ดังรูปที่ 3.6 ความละเอียดในการวัดมุมประมาณ 3° และมีอัตราการคำนวณได้ถึง 10 ค่าใน 1 วินาที



รูปที่ 3.6 การอ่านค่ามุมแสดงการเคลื่อนที่ของวัตถุโดยใช้ เซ็นเซอร์ รุ่น GY-85

จากรูปที่ 3.6 ค่าต่างๆในแผนภาพ สามารถอธิบายได้ดังนี้

Accelerometer X Accelerometer Y และ Accelerometer Z คือสัญญาณขาออกตามแนวแกน X แกน Y และแกน Z ของ Accelerometer

Gyroscope X Gyroscope Y และ Gyroscope Z คือสัญญาณขาออกตามแนวแกน X แกน Y และแกน Z ของ Gyroscope

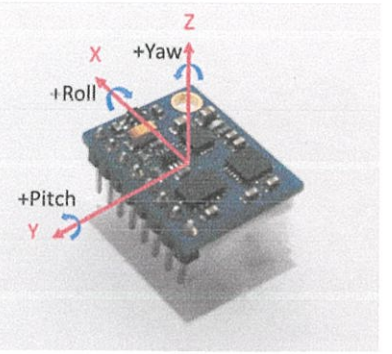
Compass X และ Compass Y คือสัญญาณขาออกตามแนวแกน X แกน Y ของ Magnetometer

ค่า Roll คือมุมที่วัดรอบแกน X ของ เซ็นเซอร์ รุ่น GY-85 ในทิศทวนเข็มนาฬิกา มีหน่วยเป็นองศา

ค่า Pitch คือมุมที่วัดรอบแกน Y ของ เซ็นเซอร์ รุ่น GY-85 ในทิศทวนเข็มนาฬิกา มีหน่วยเป็นองศา

ค่า Yaw คือมุมที่วัดรอบแกน Z ของ เซ็นเซอร์ รุ่น GY-85 ในทิศทวนเข็มนาฬิกา มีหน่วยเป็นองศา

ในส่วนของ Euler Angle Transform เพื่อหามุม Roll มุม Pitch และมุม Yaw ในหัวข้อที่ 2.6.1

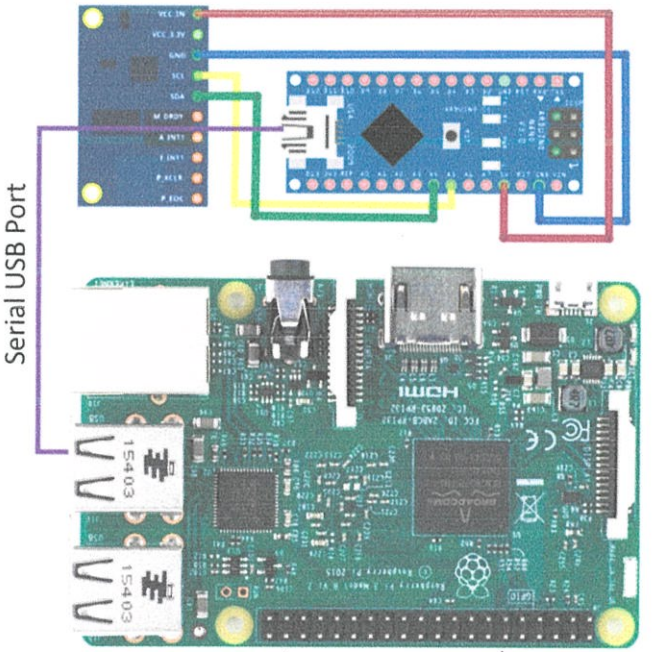


รูปที่ 3.7 แนวการวัดมุม Roll Pitch และ Yaw บน เซ็นเซอร์ รุ่น GY-85

จากรูปที่ 3.7 เป็นข้อตกลงในการทิศการวัดมุม Roll Pitch และ Yaw เพื่อเป็นแนวทางในการเลือกสัญญาณขาออกของ เซ็นเซอร์ รุ่น GY-85 มาคำนวณหามุมดังกล่าวอย่างถูกต้อง

3.1.5 การเชื่อมต่อระหว่าง Arduino Nano V3 กับ Raspberry Pi

การแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศ จะต้องใช้ค่ามุมเอียงรอบทั้งสามแกนมาคำนวณค่าแก้ให้พิกัดสายอากาศมีความถูกต้องมากยิ่งขึ้น ค่ามุมเหล่านี้เกิดจากการนำสัญญาณขาออกของ เซ็นเซอร์ รุ่น GY-85 ไปคำนวณใน Arduino Nano V3 แล้วส่งข้อมูลผ่านทางพอร์ต Serial USB ให้ Raspberry Pi ประมวลผลอีกครั้ง เพื่อลดภาระการใช้งาน Raspberry Pi ดังรูปที่ 3.8



รูปที่ 3.8 การเชื่อมต่อ Arduino Nano V3 กับ เซ็นเซอร์ รุ่น GY-85 เพื่อส่งข้อมูลไปยัง Raspberry Pi [1]

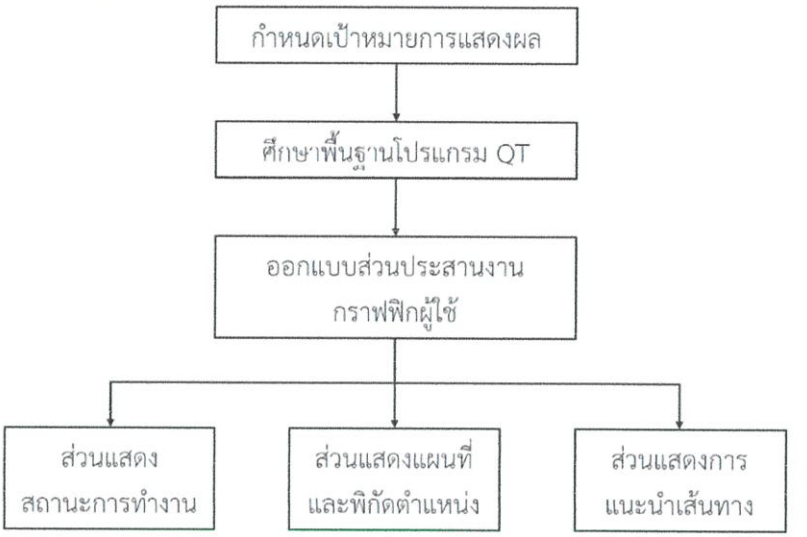
3.1.6 การเก็บพิกัดจากโปรแกรม RTKRCV ไปคำนวณใน Rotation Matrix

กรณีที่ต้องการให้ Raspberry Pi เป็นตัวประมวลผลการคำนวณตำแหน่งด้วยเทคนิค RTK จะมีโปรแกรม RTKRCV ซึ่งเป็นโปรแกรมย่อยของโปรแกรม RTKLIB ที่มีการทำงานคล้ายกันกับ RTKNAVI บนคอมพิวเตอร์ แต่โปรแกรมนี้จะทำงานได้บนระบบปฏิบัติการแบบ Linux ซึ่งเป็นระบบที่ใช้ใน Raspberry Pi มาประมวลผลการคำนวณตำแหน่งด้วยเทคนิค RTK ขณะลงสำรวจพื้นที่จริง พื้นที่อาจมีความเอียงหรือขรุขระซึ่งส่งผลให้สายอากาศที่ใช้รับสัญญาณจีเอ็นเอสเอสอยู่เกิดความเอียงและให้พิกัดที่ผิดพลาด จึงต้องนำพิกัดเหล่านั้น มาคำนวณค่าแก้ความเอียงใน Rotation Matrix ร่วมกับมุม Roll Pitch Yaw ที่วัดได้จาก Arduino โดยสามารถแสดงขั้นตอนการทำงานได้ดังรูปที่ 3.9

รูปที่ 3.9 การนำพิกัดจากโปรแกรม RTKRCV ไปคำนวณใน Rotation Matrix

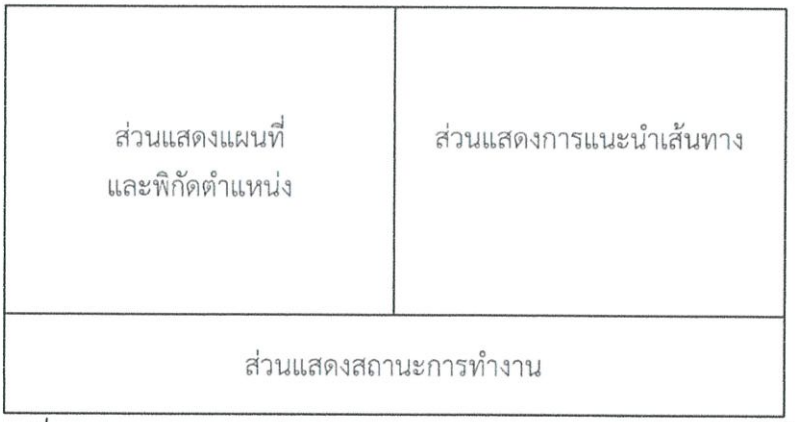
3.1.7 การออกแบบระบบเฝ้าสังเกตการณ์และติดตามทางหน้าจอ

3.1.7.1 การออกแบบหน้าต่างการใช้งานหลักของส่วนประสานงานกราฟฟิกผู้ใช้



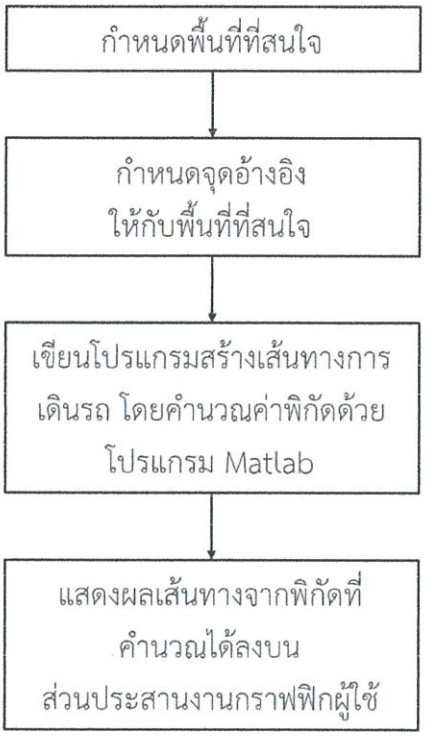
รูปที่ 3.10 ขั้นตอนการออกแบบหน้าต่างหลักส่วนประสานงานกราฟฟิกผู้ใช้

จากรูปที่ 3.10 แสดงขั้นตอนการออกแบบหน้าต่างหลักของส่วนประสานงานกราฟฟิกผู้ใช้ โดยเริ่มจากการกำหนดเป้าหมายในการแสดงผลว่าต้องการแสดงค่าหรือสถานะอะไรบ้างบนหน้าต่างประสานงานกราฟฟิกผู้ใช้ เมื่อได้สิ่งที่เป็นเป้าหมายการแสดงผลแล้วจึงทำการศึกษาโปรแกรมที่ใช้สร้างหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้ ในที่นี้ได้ทำการเลือกใช้โปรแกรม QT ซึ่งเป็นโปรแกรมที่นิยมใช้กันอย่างแพร่หลาย อีกทั้งยังเป็นโปรแกรมแบบ Cross platform ซึ่งจะรองรับการใช้งานในหลายระบบปฏิบัติการ ทั้งระบบ Windows บนคอมพิวเตอร์ หรือระบบ Linux บนอุปกรณ์ Raspberry Pi เมื่อศึกษาพื้นฐานโปรแกรม QT จนมีทักษะเบื้องต้นแล้วจึงทำการเขียนโปรแกรมสร้างหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้ ซึ่งจะประกอบด้วย 3 ส่วน คือ ส่วนแสดงสถานะการทำงานของระบบ ส่วนแสดงแผนที่และพิกัดตำแหน่ง และส่วนแสดงการแนะนำเส้นทาง โดยส่วนประกอบทั้ง 3 ส่วนดังกล่าวแสดงโดยคร่าวดังรูปที่ 3.11



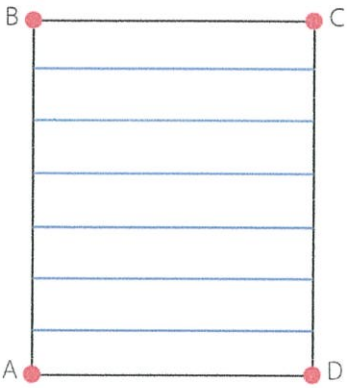
รูปที่ 3.11 การออกแบบส่วนประกอบของส่วนประสานงานกราฟฟิกผู้ใช้

3.1.7.2 การออกแบบส่วนแสดงแผนที่และพิกัดตำแหน่ง



รูปที่ 3.12 ขั้นตอนการสร้างเส้นทางการเดินรถในพื้นที่ที่สนใจ

จากรูปที่ 3.12 แสดงขั้นตอนการสร้างเส้นทางการเดินรถในพื้นที่ที่ตัดสินใจ โดยเริ่มจากการเก็บค่าพิกัดตำแหน่งบริเวณมุมของพื้นที่ที่สนใจ (ในกรณีนี้ทดลองใช้ค่าพิกัดตำแหน่งบริเวณมุมสนามทั้ง 4 มุม ของสนามกีฬาสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง) เมื่อได้ค่าตำแหน่งอ้างอิงของพื้นที่ที่สนใจแล้ว จะนำค่าพิกัดเหล่านั้นไปป้อนในโปรแกรมที่ได้ทำการเขียนไว้ด้วยโปรแกรม Matlab โดยโปรแกรมนี้จะให้ผู้ใช้งานป้อนข้อมูลตำแหน่งอ้างอิงที่ใช้ และจำนวนแนวการแบ่ง (แนวความกว้างของแต่ะเส้นทางการเดินรถ) จากนั้นโปรแกรมจะคำนวณหาจุดที่ได้จากการแบ่งพื้นที่ที่สนใจโดยอัตโนมัติ แล้วนำจุดที่ได้นั้นไปเขียนโปรแกรมแสดงเป็นเส้นทางร่วมกับแผนที่แบบ 2 มิติ ด้วยชุดคำสั่งภาษาไพธอน โดยรูปแบบเส้นทางที่แสดงจากโปรแกรมที่คำนวณได้จะแสดงโดยคร่าวดังรูปที่ 3.13

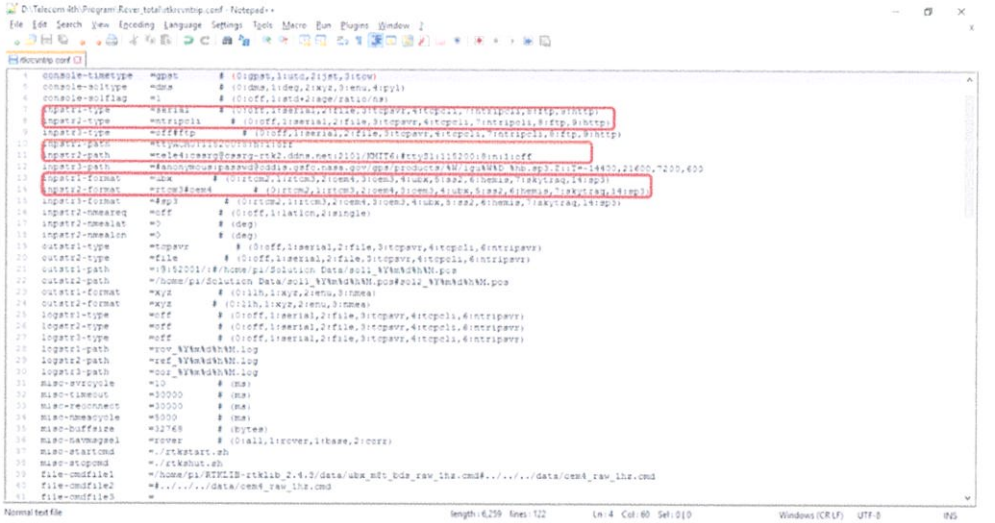


รูปที่ 3.13 รูปแบบเส้นทางที่ออกแบบ

จากรูปที่ 3.13 แสดงรูปแบบเส้นทางที่ออกแบบ โดยเส้นกรอบสีดำเป็นขอบของสนาม จุดสีแดงทั้ง 4 จุด คือพิกัดตำแหน่งอ้างอิงของพื้นที่ที่สนใจ และเส้นสีน้ำเงินแสดงเส้นทางที่ได้จากโปรแกรม

3.1.7.3 การออกแบบหน้าต่างการ Configuration ของระบบ

ในปกติแล้วการ Configuration ของระบบในแต่ละครั้ง ผู้ใช้งานจะต้องพิมพ์คำสั่งลงบน Command Line บนอุปกรณ์ Raspberry Pi เพื่อแก้ไขไฟล์ Configuration แล้วให้โปรแกรม RTKRCV ไปประมวลผลในการรับค่าพิกัดตำแหน่ง แสดงดังรูปที่ 3.14 ซึ่งก่อให้เกิดความไม่สะดวกและซับซ้อนในการใช้งาน เพื่อให้เกิดความสะดวกต่อผู้ใช้งานในหลากหลายกลุ่มเป้าหมาย ทั้งผู้ที่เป็นวิศวกรและไม่ใช่วิศวกร จึงจำเป็นต้องออกแบบและสร้างหน้าต่าง Configuration ขึ้นมา เพื่อให้ผู้ใช้งานสามารถเข้าถึงและใช้งานระบบได้อย่างรวดเร็วและง่ายต่อความเข้าใจ

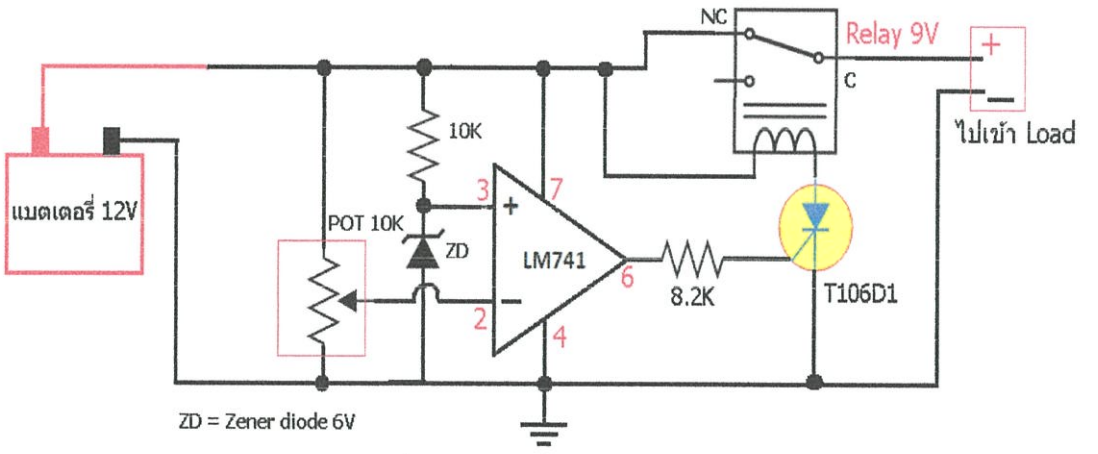


รูปที่ 3.14 ไฟล์ Config ของโปรแกรม RTKRCV

จากรูปที่ 3.14 แสดงการตั้งค่าพารามิเตอร์ต่างๆ ในการใช้งานโปรแกรม RTKRCV โดยผู้จัดทำจะทำการเลือกค่าพารามิเตอร์ที่ต้องตั้งค่าเป็นประจำ ดังตัวอย่างในกรอบสีแดงจากรูปที่ 3.14 ไปทำการสร้างหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้เพื่อแก้ไขค่าพารามิเตอร์ดังกล่าว โดยปรับรูปแบบการตั้งค่าให้เหมาะสมกับค่าพารามิเตอร์ เช่น หากการตั้งค่าต้องทำการเลือกค่าพารามิเตอร์ หน้าต่าง Configuration ในส่วนนั้นๆก็จะแสดงผลเป็นปุ่มคลิกเลือก หรือหากเป็นการกรอกข้อความ หน้าต่าง Configuration ในส่วนนั้นๆก็จะแสดงผลเป็นกล่องข้อความ เพื่อให้เหมาะสมกับค่าอินพุตที่ผู้ใช้งานจะป้อนให้กับโปรแกรม เมื่อโปรแกรมได้รับค่าอินพุตจากการป้อนโดยผู้ใช้งานแล้ว ชุดคำสั่งภายในโปรแกรมจะนำค่าอินพุตเหล่านั้นไปแก้ไขไฟล์ Config จากรูปที่ 3.14 แล้วให้โปรแกรม RTKRCV นำไปประมวลผลในการรับค่าพิกัดตำแหน่งต่อไป

3.1.8 การออกแบบวงจรควบคุมการจ่ายแบตเตอรี

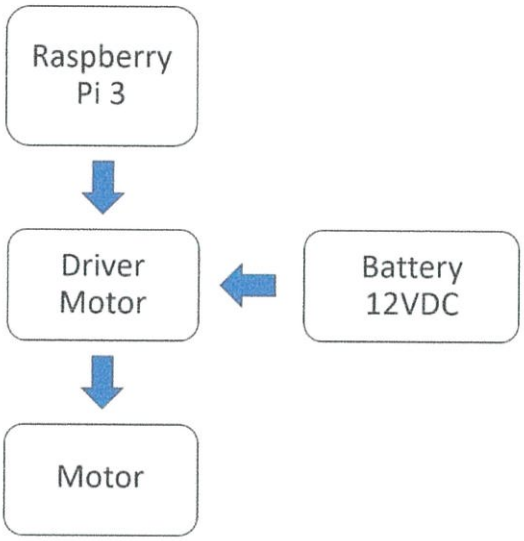
วงจรควบคุมการจ่ายแบตเตอรี เพื่อคุมปริมาณของแบตเตอรีไม่ให้มีแรงดันต่ำกว่าที่มาตรฐานกำหนด เมื่อแบตเตอรีมีแรงดันต่ำกว่า 10.8 โวลต์ จะส่งผลให้แบตเตอรี มีอายุการใช้งานได้น้อยกว่าปกติ ทำให้แบตเตอรีเสื่อมสภาพ ในปริญญานิพนธ์นี้ ได้มีการออกแบบให้ OP-AMP 741 ทำหน้าที่เป็น Comparator โดยทำการเปรียบเทียบแรงดัน ระหว่างขา อินพุตบวก และลบ ในสภาพปกติเมื่อ แบตเตอรีมีแรงดัน 12 โวลต์ มีสถานะเต็ม แรงดันเข้าที่ขา อินพุตจะมากกว่าแรงดันอ้างอิง 6 โวลต์ ที่ขา อินพุตบวก ทำให้ขา เอาต์พุต ของ OP-AMP 741 มีค่าเป็น Low SCR ไม่นำกระแส Relay จึงอยู่ในสถานะ OFF ทำให้ไฟวิ่งเข้าสู่อุปกรณ์ได้ เมื่อแบตเตอรี มีค่าต่ำลง ทำให้แรงดันของแบตเตอรีที่ผ่านการแบ่งแรงดันจาก ตัวต้านทานปรับค่าได้ (POT) เข้าที่ขาอินพุตลบ จะต่ำลงไปด้วย ทำให้แรงดันขานี้ ต่ำกว่า แรงดันที่ขาอินพุตบวก ส่งผลให้ขาเอาต์พุต มีการเปลี่ยนสถานะของ Relay ทำให้ตัดไฟออกจากอุปกรณ์ แสดงการออกแบบวงจรดังรูปที่ 3.15



รูปที่ 3.15 วงจรควบคุมแบตเตอรี

3.1.9 การออกแบบการควบคุมพวงมาลัย

การออกแบบการควบคุมพวงมาลัย เพื่อใช้ในการควบคุมรถทางเกษตรกรรมในการขับเคลื่อนอัตโนมัติ ในการควบคุมมอเตอร์ขับเคลื่อนพวงมาลัยนั้น มีการสั่งงานด้วยชุดคำสั่ง Python ที่สั่งบน Raspberry Pi แล้วทำการเชื่อมต่อผ่านอุปกรณ์ขับเคลื่อนมอเตอร์เชื่อมต่อกับแบตเตอรี และมอเตอร์ขับเคลื่อน แสดงดังรูปที่ 3.16



รูปที่ 3.16 ขั้นตอนการทำงานของมอเตอร์ขับเคลื่อน

3.2 เครื่องมือที่ใช้ในการทดลอง

โปรแกรม Matlab	
โปรแกรม RTKLIB	
โปรแกรม QT	
คอมพิวเตอร์	1 เครื่อง
Raspberry Pi	2 ชุด
Arduino Nano V.3	1 ชุด
เซ็นเซอร์ รุ่น GY-85	1 ชุด
สาย Mini USB	3 สาย
สาย Micro USB	2 สาย
สายนำสัญญาณ SMA	2 สาย
Ublox NEO-M8T Receiver	2 ชุด
TW-3710	2 ชุด
วงจรแปลงไฟ 12 เป็น 24 V	2 ชุด

วงจรแปลงไฟ 12 เป็น 5 V	2 ชุด
สวิตช์สีแดง	2 ชุด
สวิตช์สีดำ	2 ชุด
สวิตช์สีเขียว	2 ชุด
แผ่นอคริลิค 30*30 ตารางเซนติเมตร	2 แผ่น
สายไฟ	2 เมตร
Huawei E173 Aircard	1 ชุด
Huawei E303 Aircard	1 ชุด
จอแสดงผลขนาด 7 นิ้ว	1 ชุด
มอเตอร์กระแสตรง	1 ตัว

3.3 การจัดเก็บผลการทดลอง

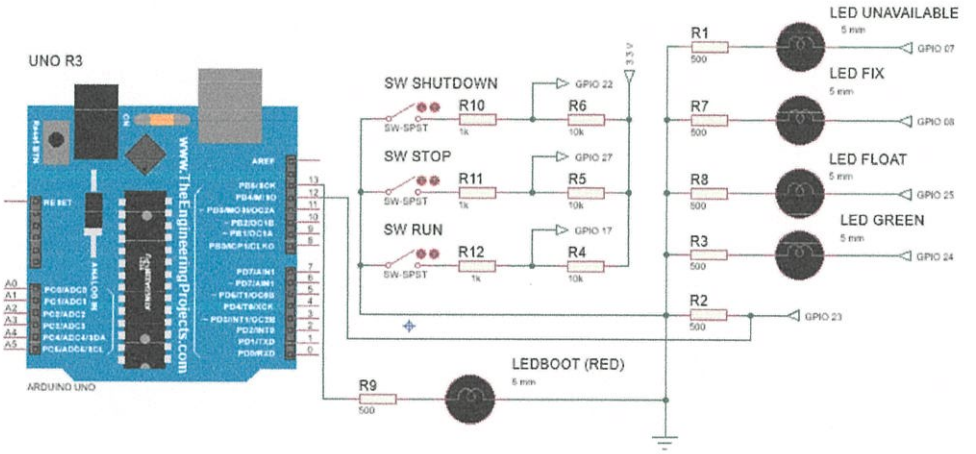
3.3.1 การออกแบบสถานีฐาน (Base Station) และสถานีจลน์ (Rover)

การออกแบบสถานีฐาน เริ่มจากการทำการเขียนโปรแกรม และวงจรเพื่อควบคุมการเปิด-ปิด อุปกรณ์ โดยมีตัวบอกสถานะการทำงานเป็นหลอด LED 2 สี และมีสวิตช์เพื่อใช้ในการกดเปิด-ปิด 3 ปุ่ม แสดงการเขียนโปรแกรมควบคุมหลอด LED แสดงสถานการณ์ทำงานของ Raspberry Pi แล้วลงโปรแกรมใน Arduino ดังนี้

```
int logicdetect = 12;
int ledboot = 13;
boolean running = false;
void setup() {
        pinMode(ledboot, OUTPUT);
        pinMode(logicdetect, INPUT);
}
void loop() {
        if(digitalRead(logicdetect)==LOW)
        {
                delay(300);
                running =! running;
                digitalWrite(ledboot, running);
        }
```

```
        else
        {
                digitalWrite(ledboot, HIGH);
        }
}
```

แสดงการต่อวงจรได้ ดังรูป 3.17 ขาที่ 12 ของบอร์ด Arduino มีหน้าที่รอรับลอจิกจากขา GPIO ที่ 23 ของ Raspberry Pi หลังจากจ่ายไฟให้ Arduino และ Raspberry Pi แล้ว หลอด LED ที่ชื่อ “LEDBOOT” จะติดแบบกะพริบเพื่อแจ้งให้ทราบว่า Raspberry Pi ยังไม่พร้อมใช้งาน เมื่อ Raspberry Pi พร้อมใช้งานจะส่งลอจิก 1 ไปยัง Arduino เพื่อให้สั่งให้ “LEDBOOT” ติดตลอด แสดงให้เห็นว่าระบบพร้อมใช้งานแล้ว เพื่อความสะดวกในการควบคุมการทำงานจึงเพิ่มปุ่มควบคุมการทำงานเพิ่มอีก 3 ปุ่ม และหลอด LED



รูปที่ 3.17 วงจรควบคุม LED

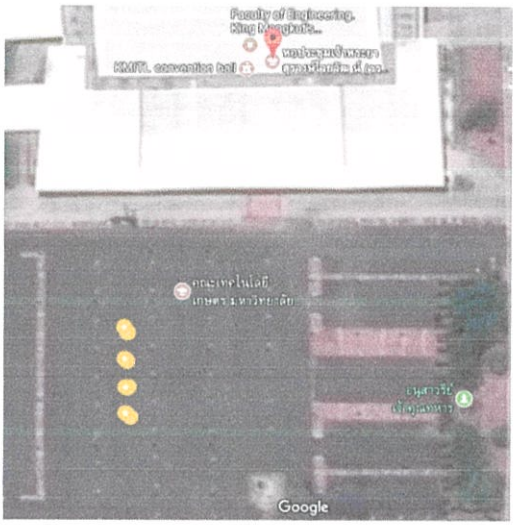
ทำการทดสอบอุปกรณ์สถานีฐาน และสถานีจลน์ โดยมีการทดสอบการรับค่าจากดาวเทียม เพื่อนำไปใช้ในการคำนวณตำแหน่งด้วยเทคนิค RTK แสดงการทดสอบสถานีฐาน และสถานีจลน์ ดังรูปที่ 3.18 และ 3.19 ตามลำดับ

รูปที่ 3.18 การทดสอบสถานีฐาน

รูปที่ 3.19 การทดสอบสถานีจลน์

3.3.2 การทดสอบระยะ Baseline ที่มีผลต่อความแม่นยำในการคำนวณตำแหน่งด้วยเทคนิค RTK โดยใช้โปรแกรม RTKLIB

1. เลือกสถานที่สำหรับการคำนวณตำแหน่งด้วยเทคนิค RTK ในที่นี้ เลือกบริเวณลานจอดรถหน้าหอประชุมเจ้าพระยาสุรวงศ์ไวยวัฒน์ ดังรูปที่ 3.20 จุดสีเหลืองในรูป คือจุดปักหมุด A B C และ D แต่ละจุดวางห่างกัน 10 เมตร และเรียงตัวอยู่ในแนวเส้นตรงเดียวกัน

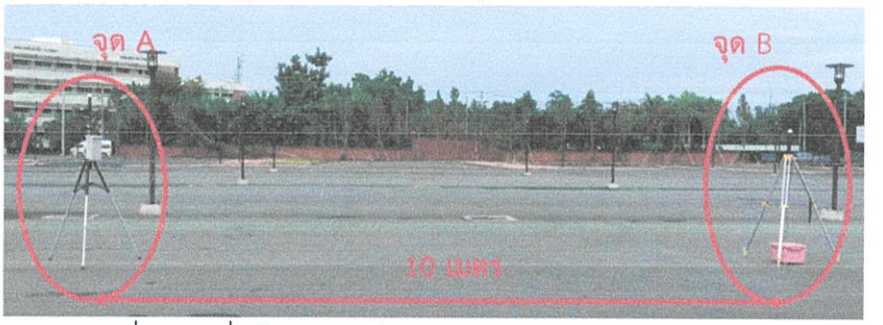


รูปที่ 3.20 ลานจอดรถหน้าหอประชุมเจ้าพระยาสุรวงศ์ไวยวัฒน์

2. ในการคำนวณตำแหน่งด้วยเทคนิค RTK จะใช้ระยะ Baseline ประมาณ 50 เมตร 808 เมตร 3516 เมตร และ 6545 เมตร

3. รูปแบบในการคำนวณตำแหน่งด้วยเทคนิค RTK มีอยู่ 2 แบบ คือแบบเส้นตรงยาว 30 เมตร และแบบวงกลมที่มีรัศมี 2 เมตรตามปรากฏในรูปที่ 3.4 และ 3.5 ในหัวข้อที่ 3.1.3

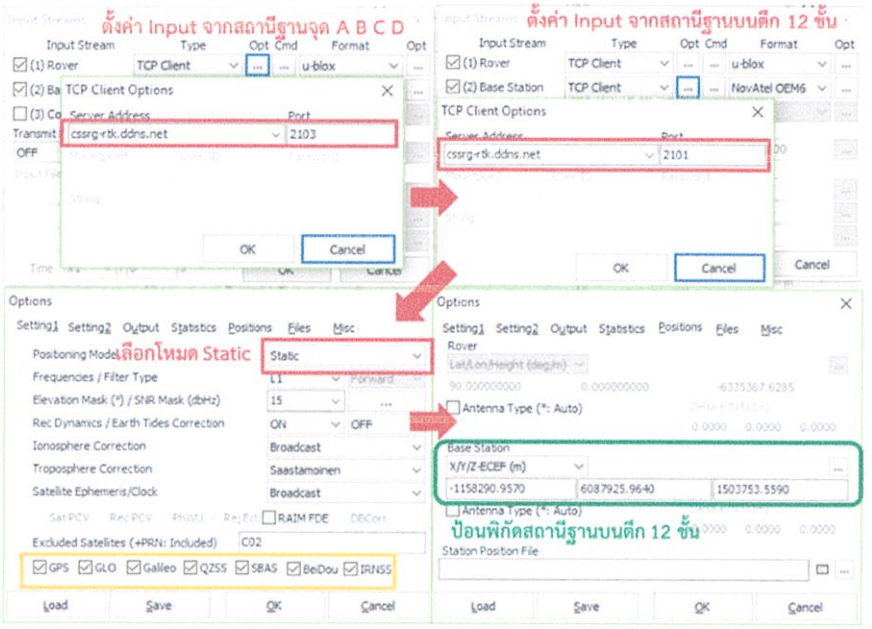
4. เริ่มวางจุดปักหมุด A B C และ D ดังรูปที่ 3.21



รูปที่ 3.21 เริ่มต้นปักหมุดโดยการปักหมุดจุด A และจุด B ก่อน

5. หาพิกัดของจุดที่ปักหมุด โดยเปิดใช้งานสถานีฐานในรูปที่ 3.20 ให้โปรแกรม STR2STR ของแต่ละสถานีฐานส่งค่าออกไปยังเครือข่าย 3G

6. เปิดโปรแกรม RTKNAVI ในคอมพิวเตอร์ แล้วตั้งค่าตามรูปที่ 3.22



รูปที่ 3.22 การตั้งค่าในโปรแกรม RTKNAVI เพื่อหาพิกัดของจุดปักหมุด A B C และ D

จากรูปที่ 3.22 เป็นการตั้งค่าโปรแกรม RTKNAVI เพื่อถ่ายหมุดจากสถานีฐานบนตึก 12 ชั้นซึ่งเป็นสถานีฐานชนิด 2 ความถี่ที่มีความแม่นยำสูงให้กับสถานีฐานที่ใช้ปักหมุด A B C และ D ซึ่งเป็นสถานีฐานชนิดความถี่เดียว โดยคอมพิวเตอร์จะรับค่าจากสถานีฐานจุดปักหมุด A B C และ D เข้าทางพอร์ต 2103 และรับค่าจากสถานีฐานชนิดสองความถี่เข้าทางพอร์ต 2101 จากนั้นโปรแกรม RTKNAVI จะประมวลผลหาพิกัดของจุดปักหมุดโดยใช้พิกัดและค่าแก้จากสถานีฐานชนิดสองความถี่เป็นจุดอ้างอิงและแก้ไขความผิดพลาดในการหาพิกัด หลังจากตั้งค่าตามรูปที่ 3.22 แล้วให้กดที่ Start ดังรูปที่ 3.23

รูปที่ 3.23 หลังการตั้งค่าโปรแกรมให้กด Start เริ่มการทำงาน

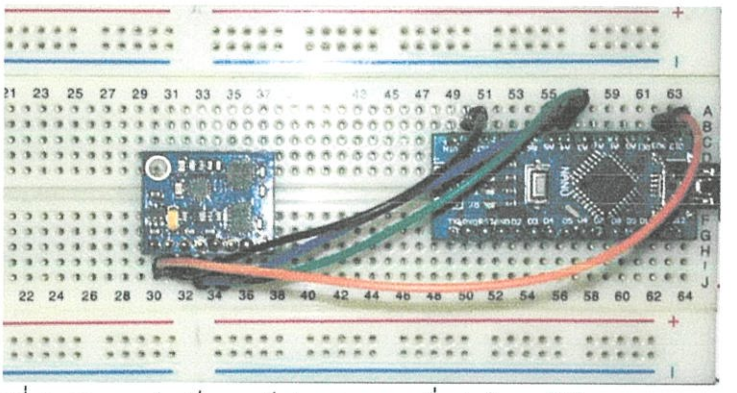
7. หลังจากได้พิกัดของจุดปักหมุดครบแล้ว จะเริ่มทำการทดสอบ โดยมีอุปกรณ์ที่ใช้ทดสอบดังรูปที่ 3.24 ซึ่งจะถูกเข็นไปตามแนววางจุดปักหมุด

รูปที่ 3.24 การติดตั้งอุปกรณ์สำหรับการคำนวณตำแหน่งด้วยเทคนิค RTK

วิธีประกอบอุปกรณ์ตามรูปที่ 3.24 ให้ติดสายอากาศกับเสา PVC จากนั้นให้ต่อสายเข้ากับ Ublox-M8T แล้วต่อเข้ากับสถานีจลน์ จากนั้นเปิดคอมพิวเตอร์เพื่อใช้งานโปรแกรม RTKNAVI แล้วเริ่มคำนวณตำแหน่งด้วยเทคนิค RTK ตามรูปแบบที่กล่าวไปแล้วในหัวข้อที่ 3.1.3

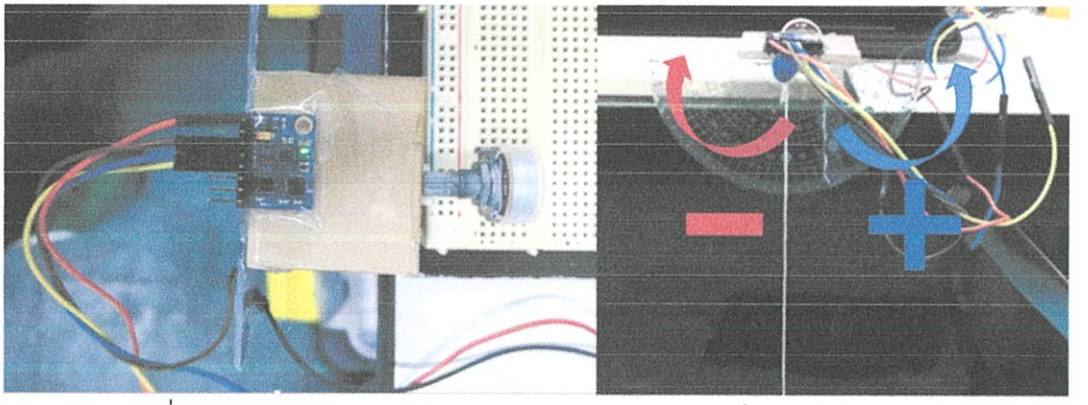
3.3.3 การอ่านค่ามุมที่แสดงการเคลื่อนที่ของวัตถุโดยใช้ IMU เซ็นเซอร์ รุ่น GY-85

ในการอ่านค่ามุม Roll Pitch และ Yaw ให้ต่อ Arduino ซึ่งเป็นหน่วยประมวลผลเข้ากับ เซ็นเซอร์ รุ่น GY-85 ดังรูปที่ 3.25



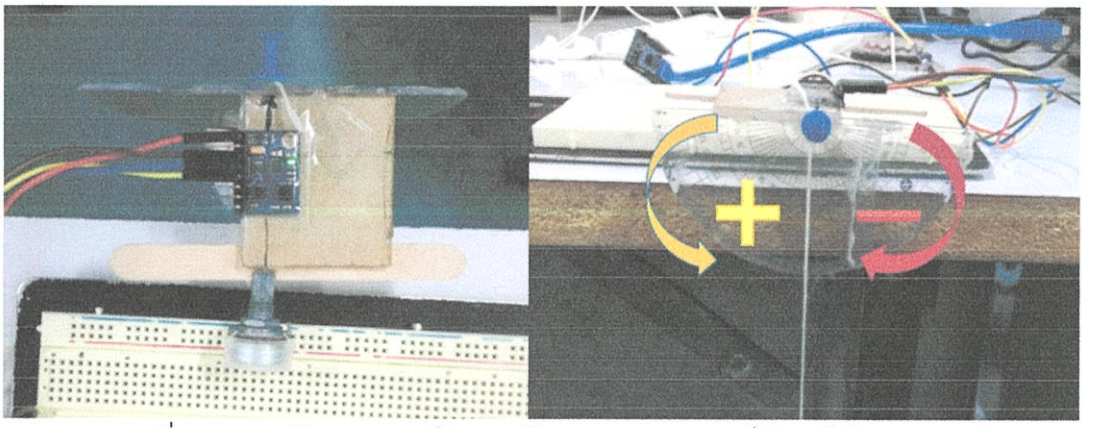
รูปที่ 3.25 การต่อเซ็นเซอร์ รุ่น GY-85 เพื่อส่งข้อมูลให้ Raspberry Pi

จากรูปที่ 3.25 เป็นการต่อ Arduino เข้ากับ เซ็นเซอร์ รุ่น GY-85 ลงบนบอร์ด โดยมีแผนภาพการต่อดังแสดงในไปแล้วในรูปที่ 2.17 จากนั้นอัพโหลดโปรแกรม Total_Sensor.ino แล้วทำการการวัดมุม Roll Pitch และ Yaw ตามรูปที่ 3.26 ถึง 3.28



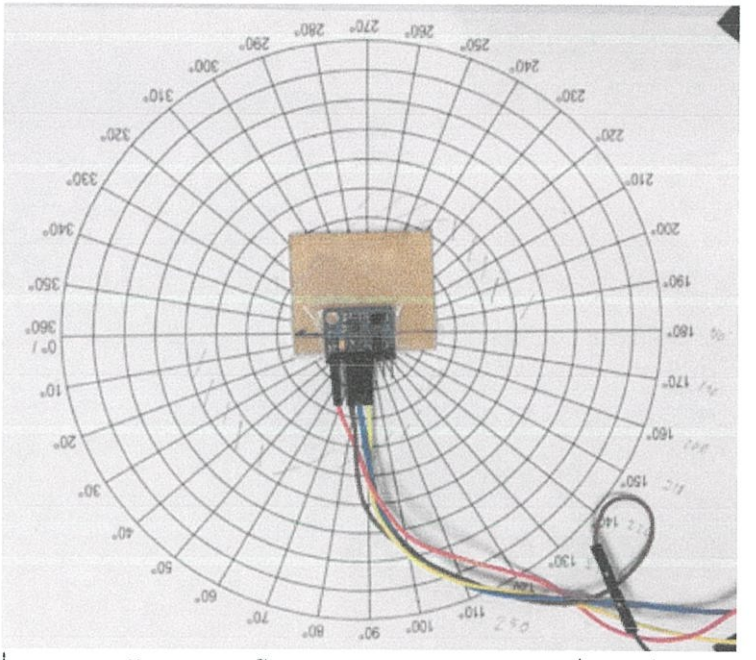
รูปที่ 3.26 การวัดมุม Roll โดยการวัดรอบแกน X ของเซ็นเซอร์ รุ่น GY-85

จากรูปที่ 3.26 ในการวัดมุม Roll จะวัดเทียบกับมุมไม้ครึ่งวงกลมโดยใช้ด้ายที่ถูกถ่วงน้ำหนักในการเล็งมุม วิธีการวัด ให้ปรับมุมการวาง เซ็นเซอร์ รุ่น GY-85 จนได้มุมที่ต้องการ จากนั้นจึงทำการบันทึกค่ามุม Roll ที่อ่านได้ แล้วหาค่าความผิดพลาดในการอ่านค่ามุม



รูปที่ 3.27 การวัดมุม Pitch โดยการวัดรอบแกน Y ของเซ็นเซอร์ รุ่น GY-85

จากรูปที่ 3.27 ในการวัดมุม Pitch จะวัดเทียบกับมุมไม้ครึ่งวงกลมโดยใช้ด้ายที่ถูกถ่วงน้ำหนักในการเล็งมุม วิธีการวัด ให้ปรับมุมการวางเซ็นเซอร์ รุ่น GY-85 จนได้มุมที่ต้องการ จากนั้นจึงทำการบันทึกค่ามุม Pitch ที่อ่านได้ แล้วหาค่าความผิดพลาดในการอ่านค่ามุม Pitch

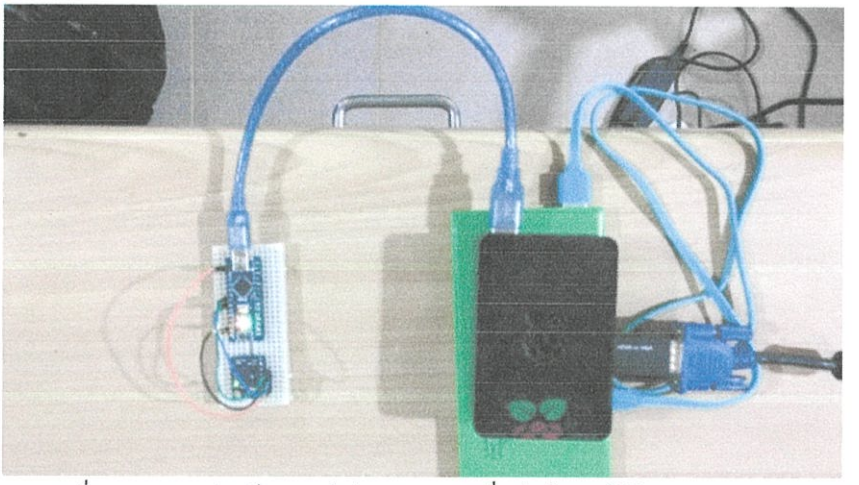


รูปที่ 3.28 การวัดมุม Yaw โดยการวัดรอบแกน Z ของเซ็นเซอร์ รุ่น GY-85

จากรูปที่ 3.28 ในการวัดมุม Yaw เป็นการวัดทิศ โดยใช้กระดาษวัดทิศวางบนโต๊ะราบ แล้ววัดให้มุม 0° ของแผ่นกระดาษชี้ไปทิศเหนือจริง โดยการวัดด้วยเข็มทิศแม่เหล็กจริงก่อน จากนั้นจึงปรับมุม Yaw ในทิศตามเข็มนาฬิกาไปทีละ 10° จนถึงมุม 350° แล้วบันทึกค่าจนครบแล้วหาค่าความผิดพลาดในการวัดมุม Yaw

3.3.4 การเชื่อมต่อระหว่าง Arduino Nano V3 กับ Raspberry Pi

1. ต่ออุปกรณ์ตามรูปที่ 3.29



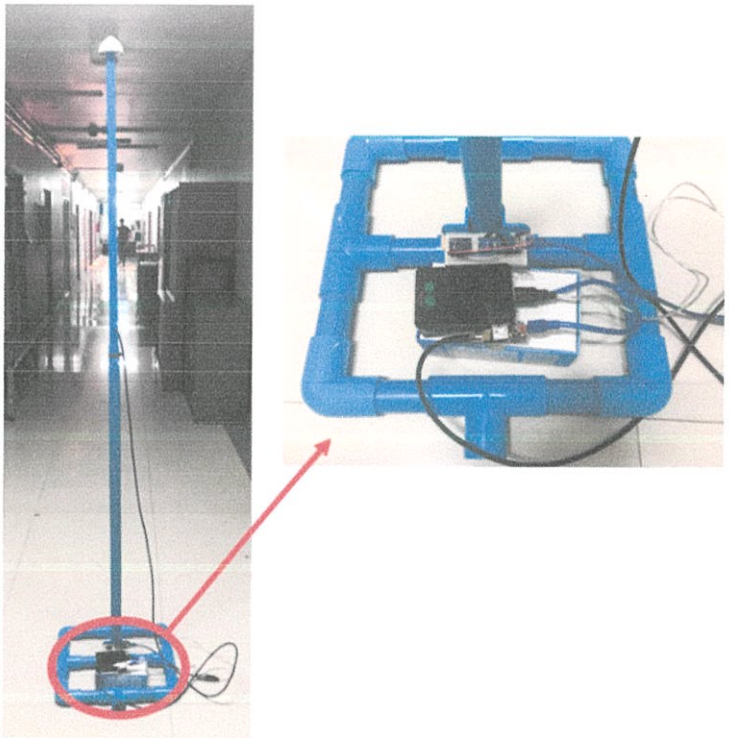
รูปที่ 3.29 การต่อเซ็นเซอร์ รุ่น GY-85 เพื่อส่งข้อมูลให้ Raspberry Pi

2. พิมพ์คำสั่ง “python serial_test.py” เพื่อเรียกใช้งานโปรแกรม serial_test.py ในโปรแกรม serial_test.py จะมีคำสั่งที่ใช้เชื่อมต่อระหว่าง Raspberry Pi กับ Arduino ผ่านทางพอร์ต Serial ที่ใช้ส่งค่ามุม Roll Pitch และ Yaw เข้าสู่ Raspberry Pi

3. ตรวจสอบผลการทำงานของโปรแกรม

3.3.5 การเก็บพิกัดจากโปรแกรม RTKRCV ไปคำนวณใน Rotation Matrix

1. ต่ออุปกรณ์ทั้งหมดตามรูปที่ 3.30



รูปที่ 3.30 การต่ออุปกรณ์เพื่อเก็บพิกัดจากโปรแกรม RTKRCV ไปคำนวณใน Rotation Matrix

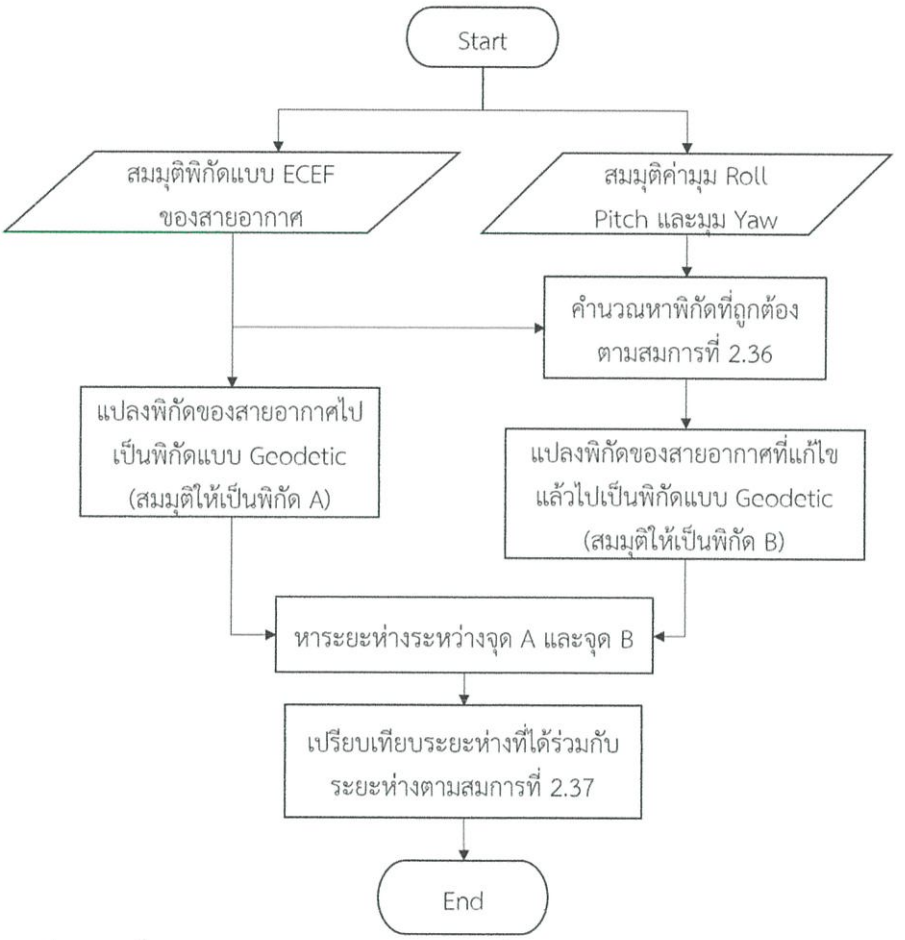
2. ล็อกอินเพื่อเข้าใช้งาน Raspberry Pi

3. พิมพ์คำสั่ง “python newedit.py” เพื่อใช้งานโปรแกรม newedit.py

4. หลังจากทำตามข้อ 3. โปรแกรม RTKRCV จะเริ่มหาค่าพิกัด พร้อมกับรับค่ามุมเอียงจากเซ็นเซอร์ GY-85 มาคำนวณร่วมกันใน Rotation Matrix ที่เขียนไว้ในโปรแกรม newedit.py จะได้ค่าพิกัดของสายอากาศที่แก้ไขแล้วแบบ ECEF และระยะเอียงของสายอากาศในหน่วยเมตร

3.3.6 การจำลองการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศในโปรแกรม MatLab

ในการจำลองการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศ จะนำสมการที่ 2.36 มาจำลองการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศตามวิธีการในรูปที่ 3.31



รูปที่ 3.31 ขั้นตอนการจำลองการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศ

กระบวนการตามรูปที่ 3.31 จะถูกเขียนเป็นโปรแกรม newedit.py ที่ให้เอาต์พุตเป็นพิกัดของสายอากาศที่แก้ไขแล้ว และระยะเอียงของสายอากาศ หลังจากที่โปรแกรมสามารถคำนวณหาเอาต์พุตได้ จะถูกนำไปแสดงผลผ่านระบบเฝ้าสังเกตการณ์และติดตามทางหน้าจอ เพื่อให้ผู้ใช้งาน หรือระบบควบคุมพวงมาลัย เป็นผู้ปรับตำแหน่งของรถให้ตรงตามเส้นทางบนพื้นที่ทางเกษตรกรรมได้อย่างถูกต้อง

3.3.7 การออกแบบระบบเฝ้าสังเกตการณ์และติดตามทางหน้าจอ

3.3.7.1 หน้าต่างการใช้งานหลักของส่วนประสานงานกราฟฟิกผู้ใช้

ในส่วนนี้จะทำการทดสอบการ เปิด-ปิด โปรแกรม ทดสอบความครบถ้วนของการแสดงผลตามขอบเขตที่ได้ออกแบบ ทั้งอัตราส่วน ขนาดอักษร รวมถึงทดสอบชุดคำสั่งที่เขียนในโปรแกรม QT ว่ามีข้อผิดพลาด และยืดหยุ่นในการใช้งานมากน้อยเพียงใด

3.3.7.2 ส่วนแสดงแผนที่และพิกัดตำแหน่ง

ในส่วนนี้ผู้ใช้งานต้องเลือกพื้นที่ที่สนใจก่อน (ในที่นี้เลือกพื้นที่ที่สนใจเป็นสนามกีฬาสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง) แล้วจึงทำตามขั้นตอนต่อไปนี้

1. ป้อนค่าพิกัดตำแหน่งอ้างอิง (มุมทั้ง 4 มุมของสนามกีฬา) ในโปรแกรมที่เขียนขึ้นจากโปรแกรม Matlab

2. ป้อนตัวเลขจำนวนเส้นทางในสนาม (หรือระยะห่างระหว่างเส้นเดินรถแต่ละเส้น)

3. กดปุ่ม Run ในโปรแกรม Matlab จะได้กราฟแสดงจุดของเส้นทางโดยคร่าว และได้ค่าพิกัดตำแหน่งของเส้นทางแต่ละเส้น

4. นำค่าพิกัดตำแหน่งที่ได้จากข้อ 3. ไปใช้ในชุดคำสั่งภาษาไพธอนเพื่อแสดงเส้นทางการเดินรถร่วมกับแผนที่แบบ 2 มิติ

จากขั้นตอนที่กล่าวไปแล้ว สามารถแสดงการเขียนโปรแกรมแสดงแผนที่และตำแหน่งของรถทางเกษตรกรรมบนแผนที่ แล้วแสดงออกผ่านทางส่วนประสานงานกราฟฟิกผู้ใช้ โดยขั้นตอนการทำงานแสดงดังรูปที่ 3.32

รูปที่ 3.32 ขั้นตอนการออกแบบส่วนแสดงผลพิกัดตำแหน่งของรถทางเกษตรกรรมลงบนแผนที่

จากรูปที่ 3.32 แสดงขั้นตอนการออกแบบส่วนแสดงผลพิกัดตำแหน่งของรถทางเกษตรกรรมออกทางส่วนประสานงานกราฟฟิกผู้ใช้ ซึ่งเริ่มทำการทดลองโดยการเขียนคำสั่งด้วยโปรแกรม QT เพื่อรับค่าพิกัดตำแหน่งที่ส่งค่าออกมาจาก TCP Port ของอุปกรณ์ Raspberry Pi จากส่วนงานของนายณัฏฐณรงค์ นิลจันทร์ โดยค่าที่สามารถรับได้จาก TCP Port ของอุปกรณ์ Raspberry Pi แสดงดังรูปที่ 3.8

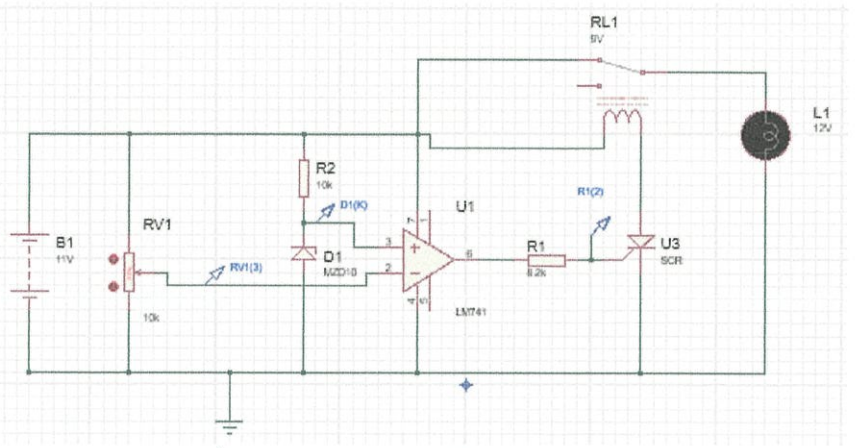
3.3.7.3 ส่วนการตั้งค่า Configuration สำหรับโปรแกรม RTKRCV

ในส่วนนี้ จะทำการทดสอบการพิมพ์หรือคลิกเลือกค่าพารามิเตอร์ที่ต้องการปรับแก้ลงในหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้ในส่วนการ Configuration จากนั้นจึงทำการตรวจสอบค่าพารามิเตอร์ที่ถูกปรับแก้ภายในไฟล์ Config ว่าตรงกับที่ป้อนค่าผ่านหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้หรือไม่ สุดท้ายจึงนำไฟล์ Config ที่ได้ไปทดสอบการรับค่าพิกัดตำแหน่งด้วยโปรแกรม RTKRCV โดยการทดสอบในส่วนนี้สามารถแสดงขั้นตอนในรูปแบบไดอะแกรมดังรูปที่ 3.33

รูปที่ 3.33 ขั้นตอนการทดสอบหน้าต่างการตั้งค่า Configuration สำหรับโปรแกรม RTKRCV

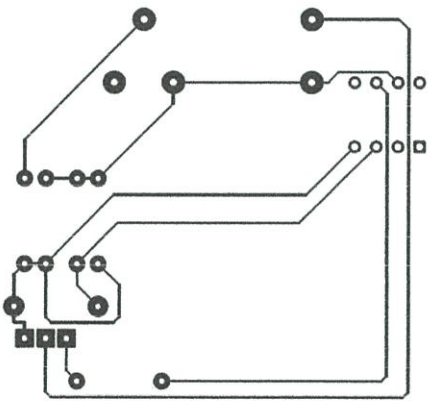
3.3.8 วงจรควบคุมการจ่ายแบตเตอรี

วงจรควบคุมการจ่ายแบตเตอรี ได้มีการออกแบบให้ OP-AMP 741 ทำหน้าที่เป็น Comparator โดยทำการเปรียบเทียบแรงดัน ระหว่างขา อินพุตบวก และลบ ในสภาพปกติเมื่อ แบตเตอรีมีแรงดัน 12 โวลต์ มีสถานะเต็ม แรงดันเข้าที่ขา อินพุตจะมากกว่าแรงดันอ้างอิง 6 โวลต์ ที่ขา อินพุตบวก ทำให้ขา เอาต์พุต ของ OP-AMP 741 มีค่าเป็น Low SCR ไม่นำกระแส Relay จึงอยู่ในสถานะไม่ทำงาน ทำให้ไฟวิ่งเข้าสู่อุปกรณ์ได้ เมื่อแบตเตอรี มีค่าต่ำลง ทำให้แรงดัน ของแบตเตอรีที่ผ่านการแบ่งแรงดันจากตัวต้านทานปรับค่าได้ (POT) เข้าที่ขาอินพุตลบ จะต่ำลงไปด้วย ทำให้แรงดันขานี้ ต่ำกว่า แรงดันที่ขาอินพุตบวก ส่งผลทำให้ขาเอาต์พุต มีการเปลี่ยนสถานะของ Relay ทำให้ตัดไฟออกจากอุปกรณ์ แสดงการออกแบบวงจรได้จำลองบนโปรแกรม Proteus ดังรูปที่ 3.34



รูปที่ 3.34 จำลองวงจรบนโปรแกรม Proteus

จากรูปที่ 3.34 หลังจากทำการจำลองวงจรบนโปรแกรม Proteus แล้วทำการออกแบบวงจรเพื่อนำลงแผ่น PCB ประกอบกับอุปกรณ์ต่างๆ ในการใช้งานจริง แสดงการออกแบบลายวงจรดังรูปที่ 3.35



รูปที่ 3.35 ลายวงจรการควบคุมแบตเตอรี

3.3.9 การออกแบบการควบคุมพวงมาลัย

ในส่วนนี้ จะออกแบบการควบคุมพวงมาลัยโดยใช้การควบคุมมอเตอร์ผ่าน Raspberry Pi โดยใช้ชุดคำสั่งภาษา Python ในการควบคุม จะทำการหมุนมอเตอร์ตามเข็มนาฬิกาและทวนเข็มนาฬิกา เพื่อให้พวงมาลัยหมุนเลี้ยวซ้าย เลี้ยวขวา ตามลำดับ ตัวอย่างแสดงทิศทางของชุดคำสั่งตามรูปที่ 3.36 และ 3.37

```
elif x=10:
    print('counter-clockwise')
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(16,GPIO.HIGH)
    print(time.ctime())
    time.sleep(5)
```

รูปที่ 3.36 คำสั่งควบคุมการหมุนตามเข็มนาฬิกา

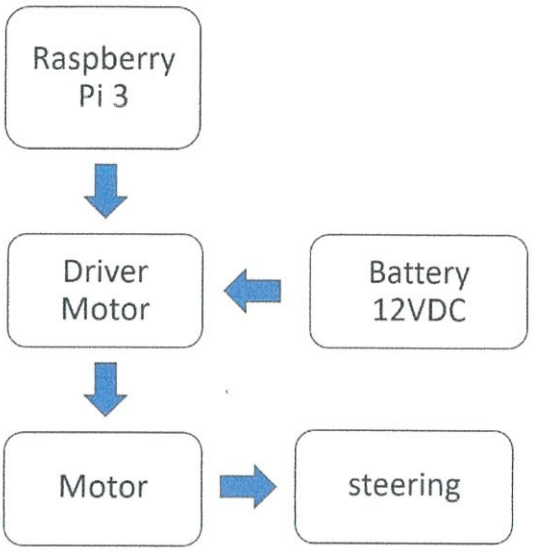
จากรูปที่ 3.36 ลำดับลอจิกของขาที่ 5 เป็นลอจิกสูง และขาที่ 7 เป็นลอจิกต่ำ ทำให้กระแสไฟที่ไหลผ่านมอเตอร์ผ่านตัวมอเตอร์ตามการควบคุมของขาที่ 5 ขาที่ 7 ไม่ได้ควบคุมการหมุนของมอเตอร์ในกรณีนี้ ในทิศที่ทำให้มอเตอร์หมุนในทิศตามเข็มนาฬิกา ส่วนขาที่ 16 เป็นขาที่ทำให้มอเตอร์เลือกใช้ลอจิกจาก Raspberry Pi ในการควบคุมมอเตอร์ จึงต้องตั้งค่าเป็นลอจิกสูงเสมอ

```
elif x=-10:
    print('Clockwise')
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH)
    print(time.ctime())
    time.sleep(1)  # One second delay
```

รูปที่ 3.37 คำสั่งควบคุมการหมุนทวนเข็มนาฬิกา

จากรูปที่ 3.37 ลำดับลอจิกของขาที่ 5 เป็นลอจิกต่ำ และขาที่ 7 เป็นลอจิกสูง ทำให้กระแสไฟที่ไหลผ่านมอเตอร์ผ่านตัวมอเตอร์ตามการควบคุมของขาที่ 7 ขาที่ 5 ไม่ได้ควบคุมการหมุนของมอเตอร์ในกรณีนี้ ในทิศที่ทำให้มอเตอร์หมุนในทิศทวนเข็มนาฬิกา ส่วนขาที่ 16 เป็นขาที่ทำให้มอเตอร์เลือกใช้ลอจิกจาก Raspberry Pi ในการควบคุมมอเตอร์ จึงต้องตั้งค่าเป็นลอจิกสูงเสมอ

จากรูปที่ 3.36 และ 3.37 แสดงคำสั่งควบคุมพวงมาลัยจาก อุปกรณ์ Raspberry Pi โดยการสั่งการด้วยโปรแกรม Python ในการสร้างชุดคำสั่ง กำหนดการหมุนของมอเตอร์ มีการป้อนกระแสไฟส่งผ่านทางพอร์ต GPIO ของ Raspberry Pi เชื่อมต่ออุปกรณ์ขับเคลื่อนมอเตอร์ ก่อนเชื่อมต่อกับมอเตอร์ แสดงการเชื่อมต่ออุปกรณ์ควบคุมพวงมาลัยดังรูปที่ 3.38



รูปที่ 3.38 การเชื่อมต่ออุปกรณ์ขับเคลื่อนกับพวงมาลัย

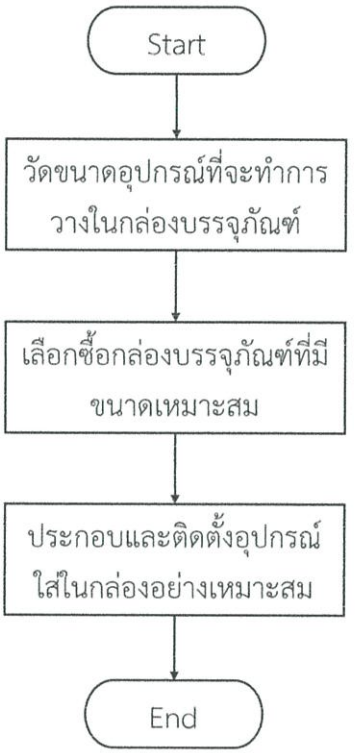
จากรูปที่ 3.38 ผู้ใช้งานสามารถป้อนมุมการหมุนของพวงมาลัยลงในโปรแกรมคำนวณการหมุน จากนั้น Raspberry Pi จะสร้างสัญญาณกระแสไฟฟ้าไปยังตัวขับมอเตอร์ให้ควบคุมการจ่ายไฟฟ้าจากแบตเตอรี เพื่อขับเคลื่อนมอเตอร์ให้หมุนพวงมาลัยจนได้มุมตามต้องการ

บทที่ 4

ผลการทดลอง

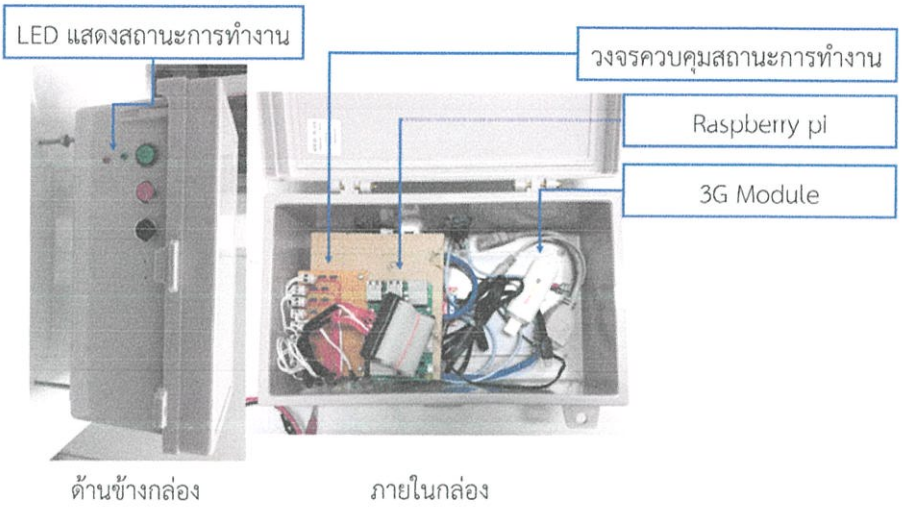
4.1 ผลการทดสอบการออกแบบอุปกรณ์สถานีฐานและสถานีจลน์

การคำนวณตำแหน่งด้วยเทคนิค RTK ต้องมีอุปกรณ์สถานีฐาน และสถานีจลน์ ที่มีความทนทานต่อสภาพอากาศ และสะดวกต่อการใช้งาน สามารถแสดงแผนภาพการประกอบอุปกรณ์สถานีฐาน และสถานีจลน์ได้ดังรูปที่ 4.1



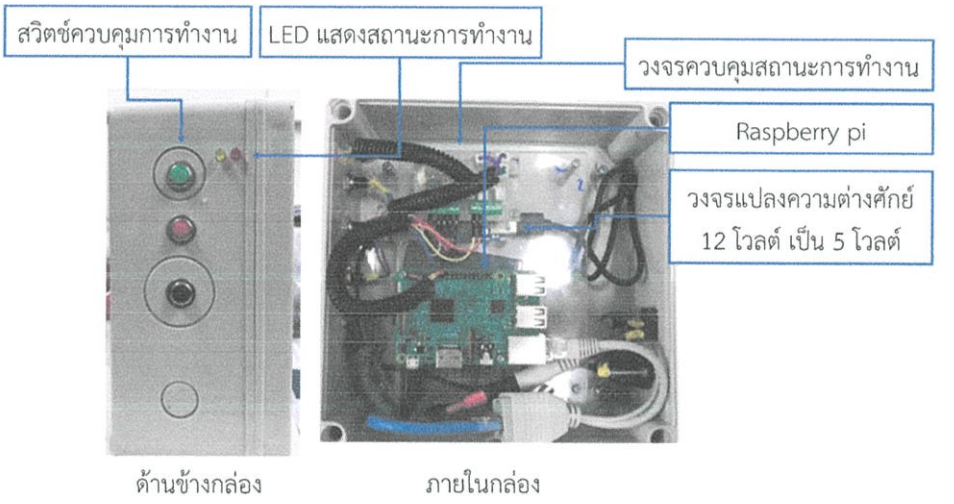
รูปที่ 4.1 แผนภาพการประกอบสถานีฐาน และสถานีจลน์

เมื่อทำการประกอบอุปกรณ์สถานีฐาน และสถานีจลน์เสร็จสิ้นแล้ว จะนำไปคำนวณตำแหน่งด้วยเทคนิค RTK ในหัวข้อที่ 4.3 โดยแสดงอุปกรณ์สถานีฐาน และสถานีจลน์ที่ประกอบเสร็จสิ้นดังรูปที่ 4.2 และ 4.3



รูปที่ 4.2 อุปกรณ์สถานีฐาน

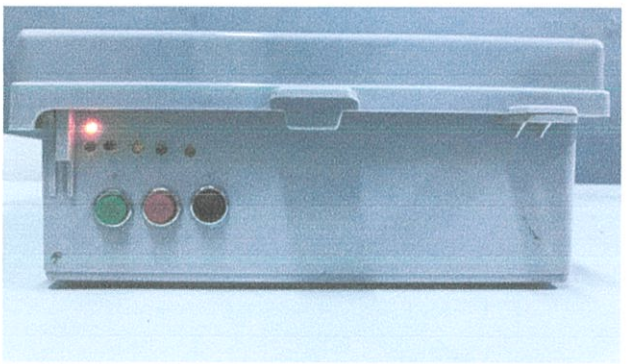
จากรูปที่ 4.2 ภายในอุปกรณ์สถานีฐาน ประกอบไปด้วย Raspberry Pi และ U-blox NEO M8T ซึ่งเป็นส่วนที่ใช้ระบุพิกัดสถานีฐาน โดย U-blox จะเชื่อมต่อกับสายอากาศ TW-3710 ที่ใช้รับสัญญาณจีเอ็นเอสเอส โมดูล 3G Air-card ใช้สำหรับเชื่อมต่อเครือข่าย 3G เพื่อส่งค่าแก้ไปประมวลผลในคอมพิวเตอร์ นอกจากนี้มีวงจรสวิตช์สำหรับเปิด-ปิด หรือสั่งใช้งานอุปกรณ์ โดยภายนอกจะมีหลอด LED สีแดงและสีเหลือง เมื่ออุปกรณ์ถูกเปิดใช้งาน หลอด LED สีแดงจะสว่าง หากกดสวิตช์สีเขียว โปรแกรม STR2STR จะเริ่มการทำงาน โดยหลอด LED สีเหลืองจะสว่าง เมื่อกดปุ่มสีแดง โปรแกรม STR2STR จะหยุดทำงาน โดยหลอด LED สีเหลืองจะดับ หากกดที่สวิตช์สีดำจะเป็นการปิดการใช้งานสถานีฐาน



รูปที่ 4.3 อุปกรณ์สถานีจลน์

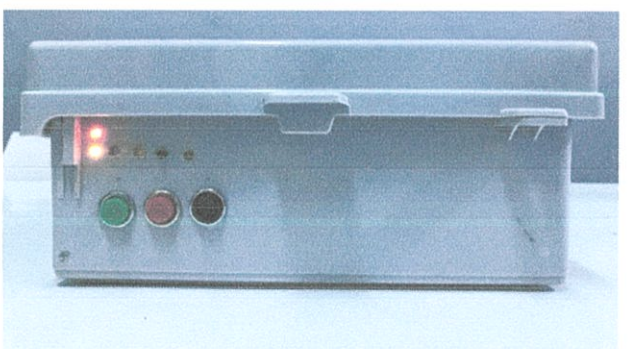
จากรูปที่ 4.3 ภายในอุปกรณ์สถานีจลน์จะมีลักษณะอุปกรณ์และการทำงานโดยรวมคล้ายกันกับสถานีฐาน แต่ในสถานีจลน์จะนำแบตเตอรีเอาไว้ภายนอก เพื่อเพิ่มพื้นที่ให้กับ Arduino สำหรับอ่านค่า Roll Pitch และ Yaw จากเซนเซอร์ GY-85 แล้วส่งไปยัง Raspberry Pi ให้แก้ไขพิกัดที่เกิดจากการเอียงของสายอากาศได้

จากรูปที่ 4.2 และ 4.3 จะมีวงจรสำหรับควบคุมการทำงานอยู่ภายใน เพื่อใช้สั่งการทำงาน การหยุดการทำงานของโปรแกรม จนถึงการปิดระบบการทำงานของ Raspberry Pi โดยออกแบบใช้ปุ่มในการสั่งการ และมีหลอด LED 3 สี สำหรับแสดงสถานะการทำงาน ดังรูปที่ 4.4 ถึง 4.7



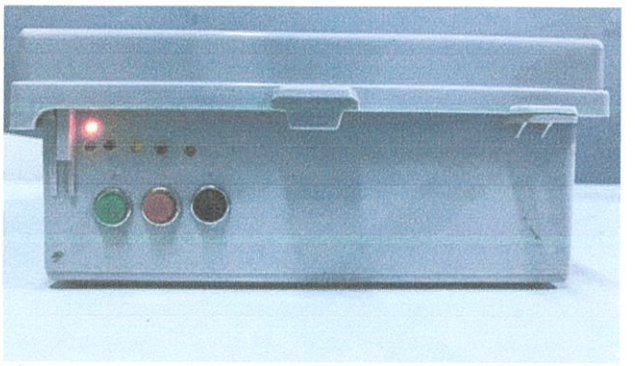
รูปที่ 4.4 สถานะการเปิดอุปกรณ์

จากรูปที่ 4.4 เมื่อนำวงจรสำหรับควบคุมการทำงานเข้ากับ Raspberry Pi หลังจากจ่ายไฟแล้ว Raspberry Pi จะทำงานตามโปรแกรมที่กำหนดไว้ โดยจะส่งไฟเลี้ยงไปยังหลอด LED สีแดงติด เพื่อบอกให้ทราบว่า Raspberry Pi พร้อมที่จะทำงานแล้ว เพื่อเริ่มการทำงานของโปรแกรม RTKRCV จะต้องทำตามขั้นตอนดังรูปที่ 4.5



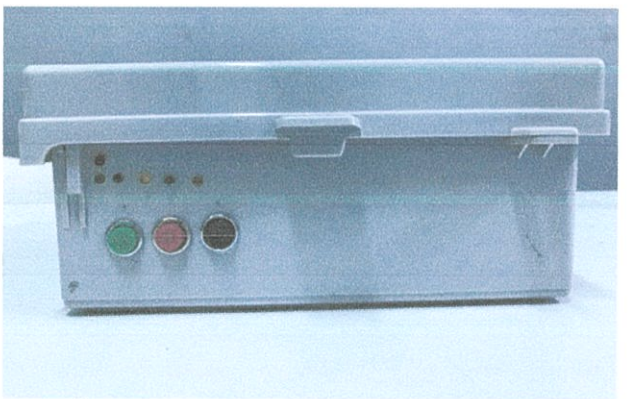
รูปที่ 4.5 สถานะแสดงการทำงานของโปรแกรม RTKRCV

จากรูปที่ 4.5 ทำการกดปุ่มสีเขียวบนอุปกรณ์ จะเป็นการสั่งการให้โปรแกรม RTKRCV ทำงาน เพื่อให้ผู้ใช้งานทราบ จึงตั้งเงื่อนไขว่าหากมีการทำงานของโปรแกรม RTKRCV จะสั่งให้หลอด LED สีเหลืองติด และหากต้องการหยุดการทำงานของโปรแกรม ให้ทำตามรูปที่ 4.6



รูปที่ 4.6 สถานะแสดงการหยุดทำงานของโปรแกรม RTKRCV

จากรูปที่ 4.6 ทำการกดปุ่มสีแดงบนอุปกรณ์ จะทำให้หยุดการทำงานของโปรแกรม RTKRCV และเพื่อให้ผู้ใช้งานสามารถทราบสถานการณ์ทำงานได้ หลังจากโปรแกรมหยุดการทำงาน จะสั่งให้หลอด LED สีเหลืองดับ หากทำการทดสอบเสร็จแล้วต้องการปิดตัวอุปกรณ์ ให้ทำตามรูปที่ 4.7



รูปที่ 4.7 แสดงสถานะหยุดการทำงานของอุปกรณ์

จากรูปที่ 4.7 ทำการกดปุ่มสีดำบนอุปกรณ์ จะเป็นการหยุดการทำงานของ Raspberry Pi โดยสังเกตได้จากหลอดไฟ LED สีแดงจะดับลง แล้ว Raspberry Pi จะปิดตัวลง และหยุดการทำงานทันที

4.2 ผลการทดสอบการคำนวณตำแหน่งด้วยเทคนิค RTK โดยใช้โปรแกรม RTKLIB

ค่าความคลาดเคลื่อนในการระบุพิกัดตำแหน่งจากการปักหมุดจุด A B C และ D ด้วยเทคนิค RTK โดยใช้สถานีฐานชนิดสองความถี่เป็นจุดอ้างอิงในการปักหมุด แสดงดังตารางที่ 4.1

ตารางที่ 4.1 พิกัดของจุดปักหมุด A B C และ D

Marker	ECEF Position (m.)			95% Error (mm.)	STDV (mm.)
A	-1158983.0314,	6087796.5553,	1503511.9346	6.7	0.97
B	-1158983.6901,	6087798.8499,	1503502.2306	1.75	0.93
C	-1158984.2619,	6087801.1681,	1503492.5065	2.22	0.72
D	-1158984.8431,	6087803.4609,	1503482.8178	3.98	5.66

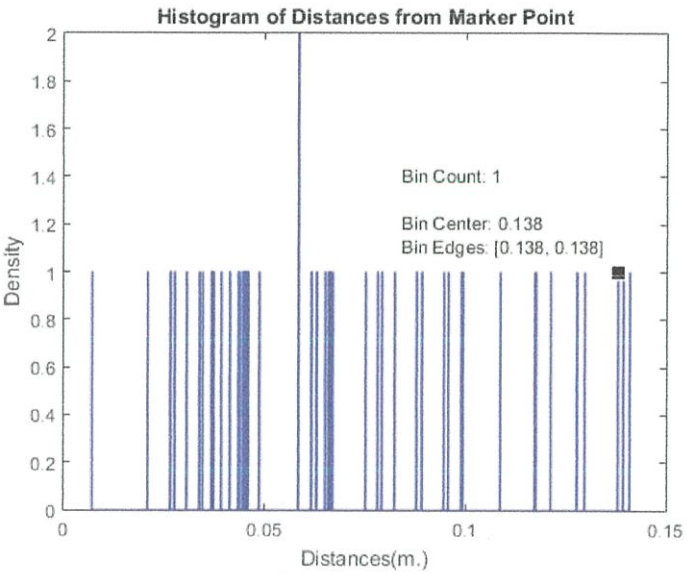
ค่า 95% Error คือระยะคลาดเคลื่อนของพิกัดโดยรวมกับพิกัดอ้างอิง ค่า STDV คือค่าความเบี่ยงเบนมาตรฐานที่บ่งบอกว่าพิกัดโดยรวมมีการกระจายห่างไปจากพิกัดอ้างอิงเท่าใด หลังจากการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงเป็นระยะ 30 เมตร และทำ RTK แนววงกลมรัศมี 2 เมตรด้วยสถานีฐานชนิดความถี่เดียวที่สร้างขึ้น จะได้ผลการคำนวณตำแหน่งด้วยเทคนิค RTK ในแต่ละระยะ Baseline ดังนี้

ตารางที่ 4.2 การคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงด้วยสถานีฐานชนิดความถี่เดียวด้วยระยะ Baseline 50 เมตร (ระยะ Baseline ไม่เกิน 57.76 เมตรที่จุด D)

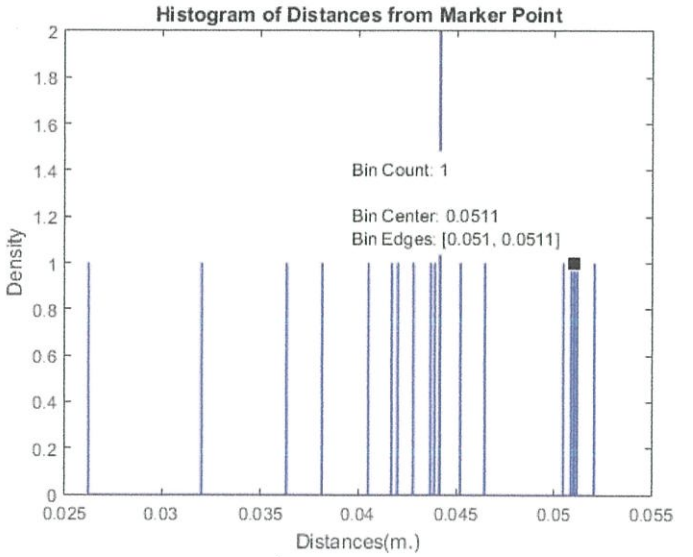
RTK Testing with Baseline 50+ m. (Single Frequency)			
Marker	95% Error (cm.)	STDV (cm.)	%Fixed Solution
A	13.82	3.51	40.93
B	5.12	0.68	
C	8.98	2.53	
D	11.24	3.72	
95% Error total (cm.)	12.11		
STDV total (cm.)	3.28		

จากตารางที่ 4.2 การคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรง ทำการเลื่อนสายอากาศโดยเดินถือตามแนวเส้นตรงที่ลากเชื่อมจุดปักหมุดทั้งหมดเข้าด้วยกัน โดยจะเดินไปหยุดที่จุดปักหมุดนานจุดละ 30 วินาทีเป็นอย่างน้อย เพื่อเก็บค่าพิกัดตรงจุดปักหมุด แล้วนำไปหาระยะห่างกับจุดปักหมุดจากการใช้สถานีฐานชนิดสองความถี่ เนื่องจากตัวเสาที่ติดสายอากาศถูกถือ

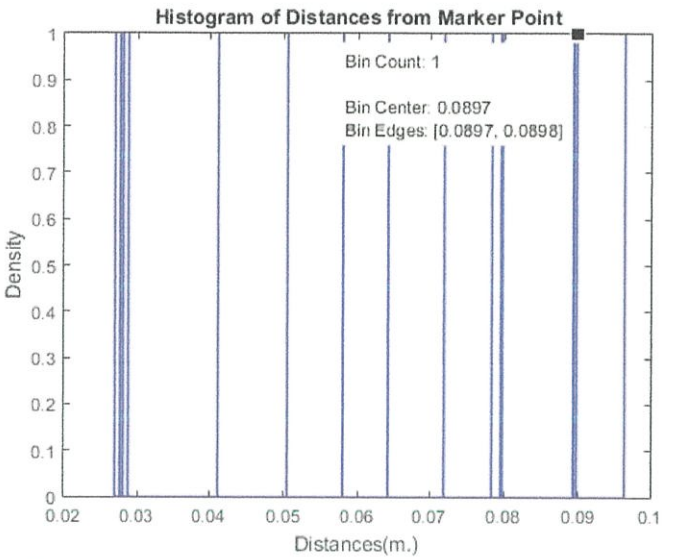
ด้วยมือ ไม่ได้ติดตั้งกับโครงสร้างที่แข็งแรง ทำให้มีความผิดพลาดจากตัวมนุษย์ในการทดสอบได้ ส่งผลให้ในการทดสอบนี้มีค่าความผิดพลาดที่แต่ละจุดปักหมุดมีค่าค่อนข้างสูง โดยมีพิกัดที่เป็น Fixed Solution (ค่าที่บ่งบอกถึงความน่าเชื่อถือของพิกัดที่คำนวณได้ โดยทั่วไปมักมีค่าไม่ต่ำกว่า 3) เพียงร้อยละ 40.93 ของพิกัดทั้งหมด ค่าความคลาดเคลื่อนทั้งการทดลองนี้มีค่าประมาณ 12.11 เซนติเมตร ความคลาดเคลื่อนแต่ละจุดปักหมุดในตารางที่ 4.2 สามารถหาได้จากฮิสโตแกรมในรูปที่ 4.8 ถึง 4.11 และมีระยะคลาดเคลื่อนทั้งการทดสอบดังรูปที่ 4.12



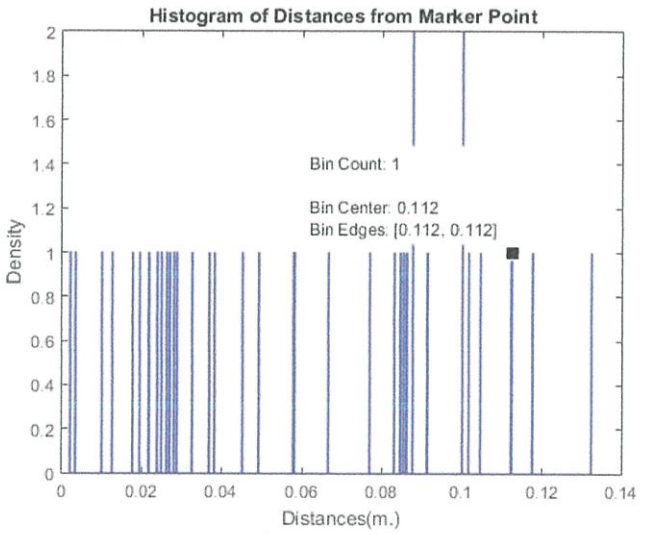
รูปที่ 4.8 ฮิสโตแกรมความคลาดเคลื่อนจุด A (ระยะ Baseline เท่ากับ 49.91 เมตร)



รูปที่ 4.9 ฮิสโตแกรมความคลาดเคลื่อนจุด B (ระยะ Baseline เท่ากับ 50.65 เมตร)

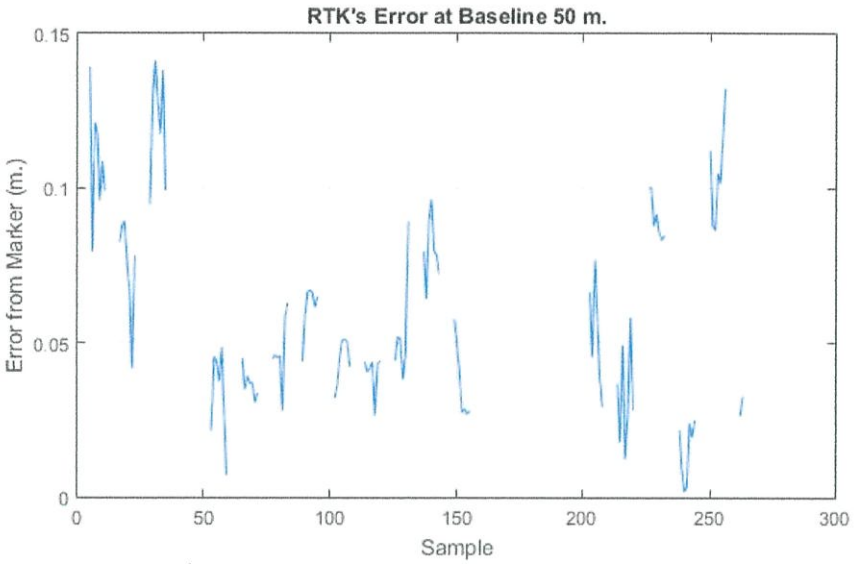


รูปที่ 4.10 ฮิสโตแกรมความคลาดเคลื่อนจุด C (ระยะ Baseline เท่ากับ 53.40 เมตร)



รูปที่ 4.11 ฮิสโตแกรมความคลาดเคลื่อนจุด D (ระยะ Baseline เท่ากับ 57.76 เมตร)

จากรูปที่ 4.4 ถึง 4.7 เป็นฮิสโตแกรมสำหรับหาค่า 95% Error ในตารางที่ 4.2 จากฮิสโตแกรมดังกล่าว ค่าความคลาดเคลื่อนในการระบุพิกัดตำแหน่งด้วยเทคนิค RTK ที่ใช้สถานีฐานชนิด 1 ความถี่ ระยะ Baseline 50 เมตร ในการปักหมุดแต่ละจุดจะมีค่าไม่เกิน 14 เซนติเมตร



รูปที่ 4.12 ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ด้วยระยะ Baseline 50 เมตร

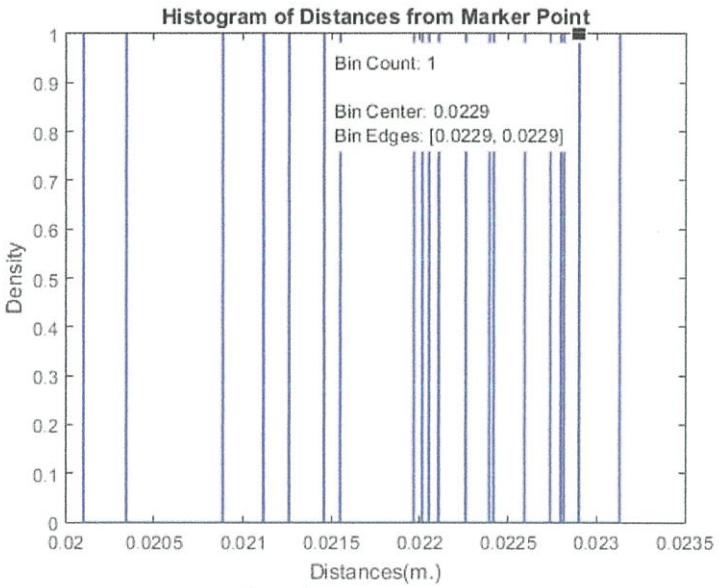
จากรูปที่ 4.12 เส้นกราฟในรูปมีลักษณะขาดหายไป เนื่องจากผู้จัดทำเขียนโปรแกรมให้นำพิกัดที่มี Fixed Solution ไม่ต่ำกว่า 3 มาหาความคลาดเคลื่อน ซึ่งขณะทดสอบค่าพิกัดส่วนมากที่บันทึกได้มีจำนวนพิกัดที่เป็น Fixed Solution ค่อนข้างน้อย จึงทำให้พิกัดโดยรวมที่เก็บได้จากการทดลองมีความแม่นยำน้อย

เมื่อเปลี่ยนระยะ Baseline ในการทำ RTK เป็น 808 เมตร โดยใช้สถานีฐานชนิดความถี่เดียว จะได้ผลการทดสอบดังตารางที่ 4.3

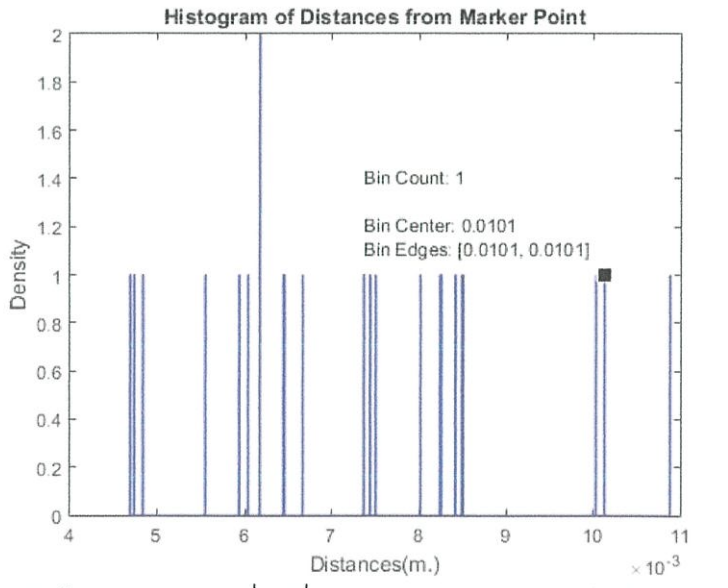
ตารางที่ 4.3 การคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงด้วยสถานีฐานชนิดความถี่เดียวด้วยระยะ Baseline 808 เมตร (ระยะ Baseline ไม่เกิน 822.97 เมตรที่จุด D)

RTK Testing with Baseline 808 m. (Single Frequency)			
Marker	95% Error (cm.)	STDV (cm.)	%Fixed Solution
A	2.29	0.09	42.56
B	1.01	0.17	
C	4.72	0.17	
D	3.29	0.21	
95% Error total (cm.)	4.49		
STDV total (cm.)	1.24		

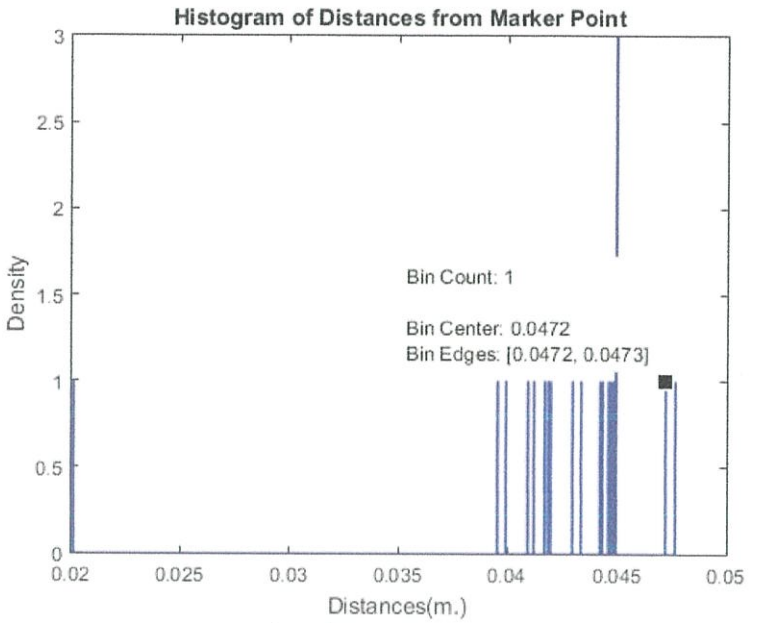
จากตารางที่ 4.3 นักศึกษาได้เปลี่ยนจากการถือเสาสายอากาศเป็นการติดตั้งเสาสายอากาศบนฐานรถเข็น ทำให้การเคลื่อนย้ายเสามีความเสถียรมากขึ้น ส่งผลให้มีความคลาดเคลื่อนของทั้งการทดสอบถดถอยเหลือ 4.49 เซนติเมตร และมีร้อยละของ Fixed Solution เป็น 42.56 โดยความคลาดเคลื่อนของแต่ละจุดในตารางที่ 4.3 แสดงดังรูปที่ 4.13 ถึง 4.16 และมีความคลาดเคลื่อนทั้งการทดลองดังรูปที่ 4.17



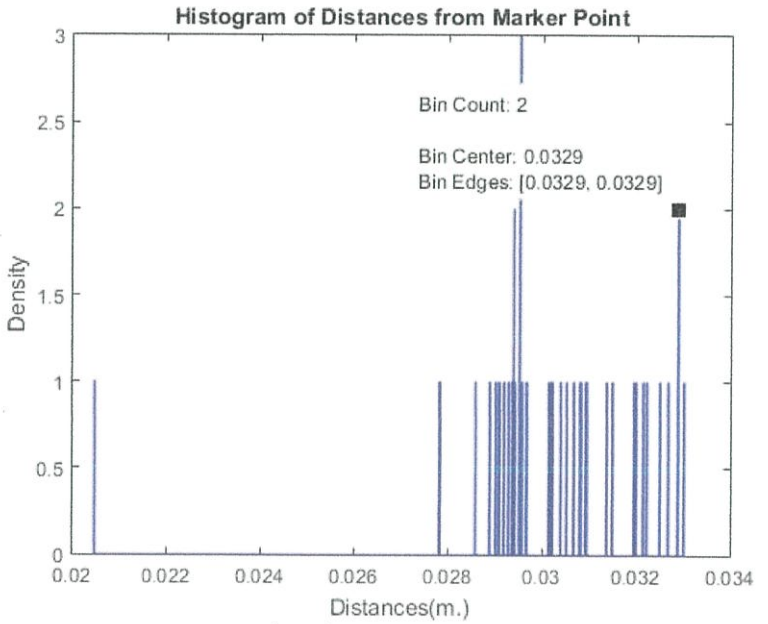
รูปที่ 4.13 ฮิสโตแกรมความเคลื่อนที่จุด A (ระยะ Baseline เท่ากับ 808.13 เมตร)



รูปที่ 4.14 ฮิสโตแกรมความเคลื่อนที่จุด B (ระยะ Baseline เท่ากับ 813.03 เมตร)

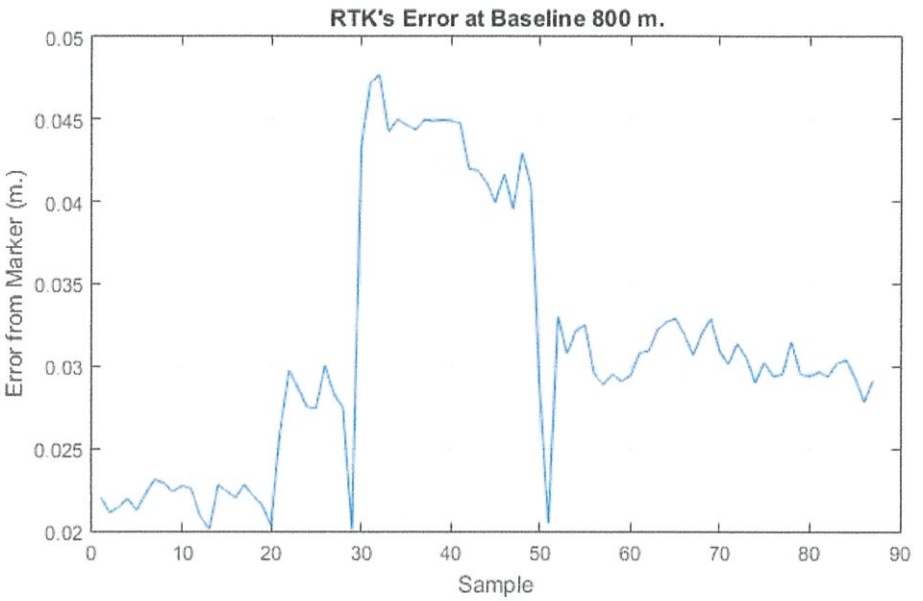


รูปที่ 4.15 ฮิสโตแกรมความเคลื่อนที่จุด C (ระยะ Baseline เท่ากับ 817.96 เมตร)



รูปที่ 4.16 ฮิสโตแกรมความเคลื่อนที่จุด D (ระยะ Baseline เท่ากับ 822.97 เมตร)

จากรูปที่ 4.13 ถึง 4.16 เป็นฮิสโตแกรมสำหรับหาค่า 95% Error ในตารางที่ 4.3 จากฮิสโตแกรมดังกล่าว ค่าความคลาดเคลื่อนในการระบุพิกัดตำแหน่งด้วยเทคนิค RTK ที่ใช้สถานีฐานชนิด 1 ความถี่ ระยะ Baseline 808 เมตร ในการปักหมุดแต่ละจุดจะมีค่าไม่เกิน 5 เซนติเมตร



รูปที่ 4.17 ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ด้วยระยะ Baseline 808 เมตร

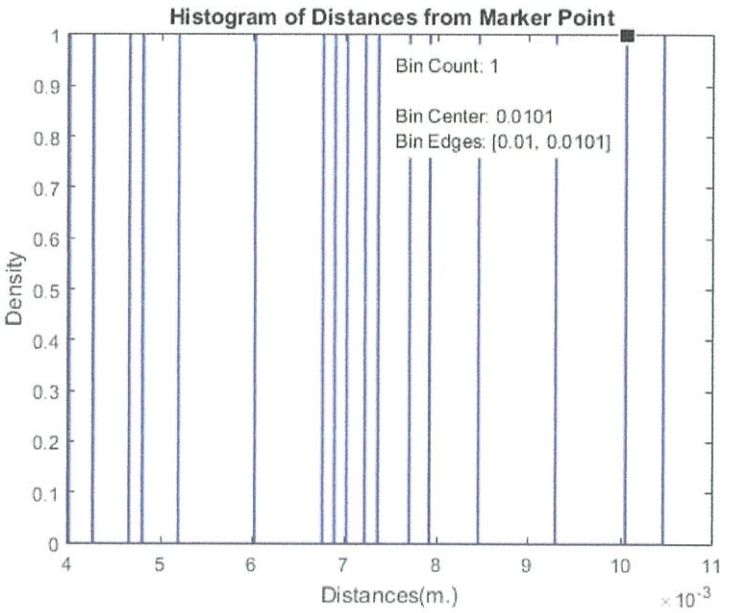
จากรูปที่ 4.17 หลังจากปรับปรุงอุปกรณ์การทดสอบ ความคลาดเคลื่อนในการทดสอบลดลง และมีค่าร้อยละ Fixed Solution เพิ่มขึ้นเป็นร้อยละ 42.56 และมีค่าความคลาดเคลื่อนของทั้งการทดสอบนี้ลดลงอีกเช่นกัน

เมื่อเปลี่ยนระยะ Baseline เป็น 3.5 กิโลเมตร ในการทดลองนี้ได้ปรับปรุงการติดตั้งเสาเข้ากับรถเข็นใหม่ให้เล็งต้นเสาได้ตรงกับจุดปักหมุดมากขึ้น และมีผลการทดสอบดังตารางที่ 4.4

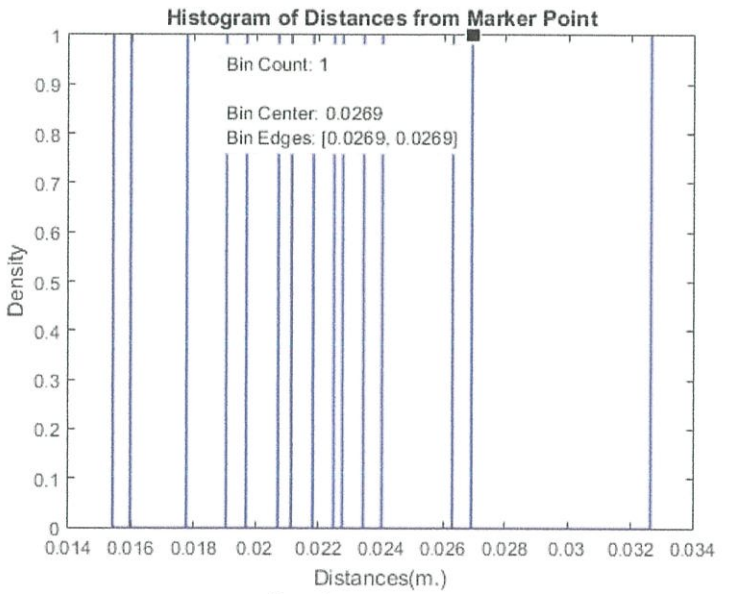
ตารางที่ 4.4 การคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงด้วยสถานีฐานชนิดความถี่เดียวด้วยระยะ Baseline 3.5 กิโลเมตร (ระยะ Baseline ไม่เกิน 3.523 กิโลเมตรที่จุด D)

RTK Testing with Baseline 3.5 km. (Single Frequency)			
Marker	95% Error (cm.)	STDV (cm.)	%Fixed Solution
A	1.01	0.2	29.8
B	2.69	0.45	
C	3.13	0.5	
D	3.82	0.22	
95% Error total (cm.)	3.78		
STDV total (cm.)	1.15		

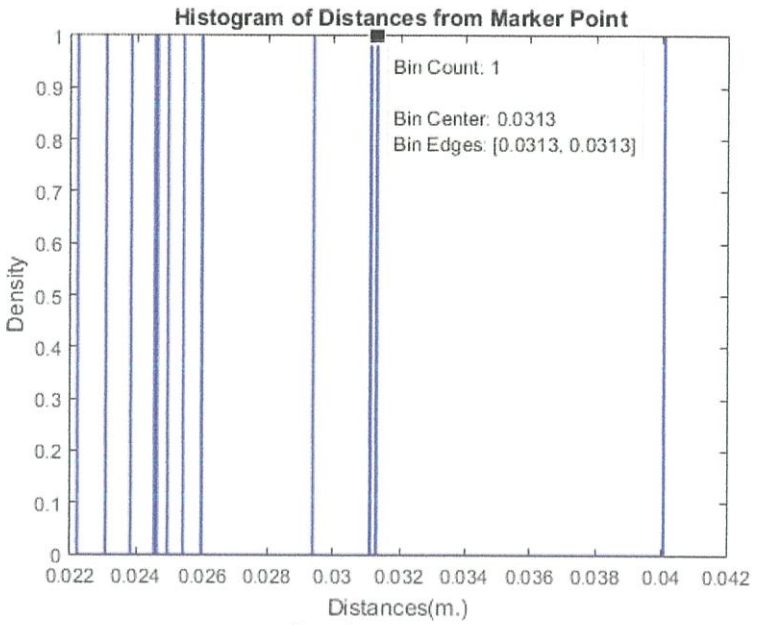
จากตารางที่ 4.4 หลังจากปรับปรุงการวางเสาใหม่แล้ว พบว่าความคลาดเคลื่อนโดยรวมของการทดสอบเป็น 3.78 เซนติเมตร แต่เนื่องจากระยะ Baseline ที่ไกลขึ้น ทำให้มีร้อยละ Fixed Solution ลดลง ความคลาดเคลื่อนของจุดปักหมุดแต่ละจุดจากตารางที่ 4.4 สามารถหาได้จากรูปที่ 4.18 ถึง 4.21 และมีความคลาดเคลื่อนตลอดทั้งการทดสอบเป็นดังรูปที่ 4.22



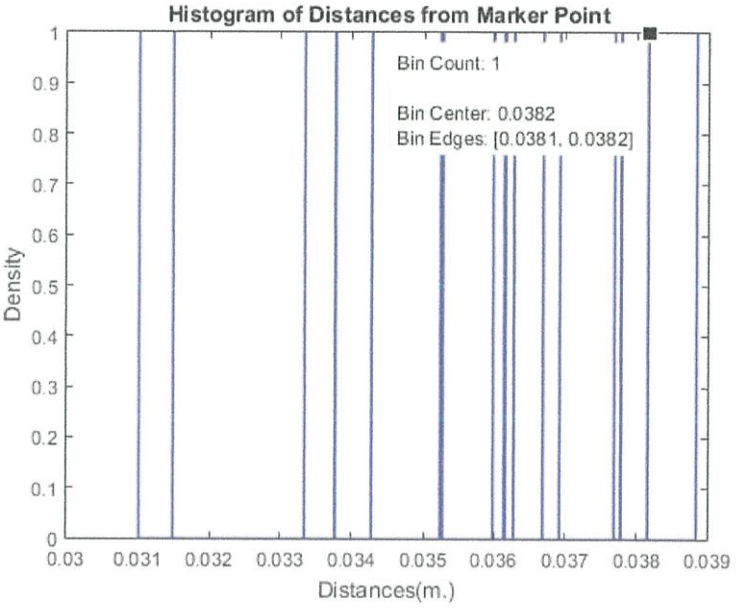
รูปที่ 4.18 ฮิสโตแกรมความคลาดเคลื่อนที่จุด A (ระยะ Baseline เท่ากับ 3.513 กิโลเมตร)



รูปที่ 4.19 ฮิสโตแกรมความคลาดเคลื่อนที่จุด B (ระยะ Baseline เท่ากับ 3.516 กิโลเมตร)

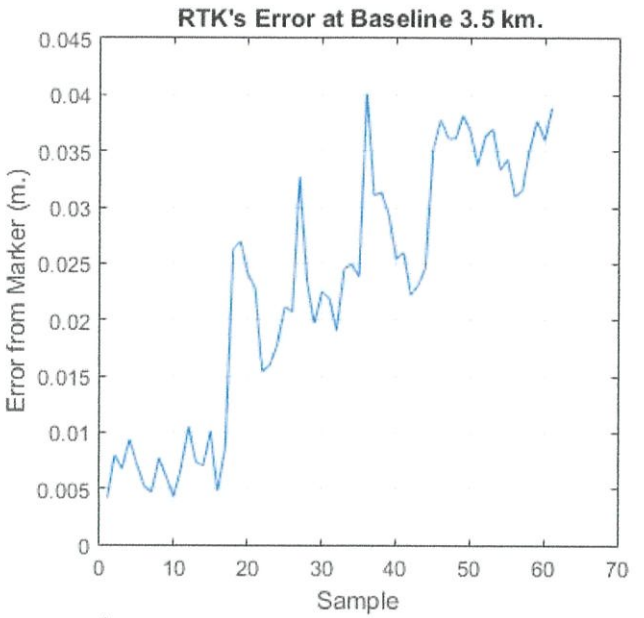


รูปที่ 4.20 ฮิสโตแกรมความคลาดเคลื่อนที่จุด C(ระยะ Baseline เท่ากับ 3.519 กิโลเมตร)



รูปที่ 4.21 ฮิสโตแกรมความคลาดเคลื่อนที่จุด D (ระยะ Baseline เท่ากับ 3.523 กิโลเมตร)

จากรูปที่ 4.18 ถึง 4.21 เป็นฮิสโตแกรมสำหรับหาค่า 95% Error ในตารางที่ 4.4 จากฮิสโตแกรมดังกล่าว ค่าความคลาดเคลื่อนในการระบุพิกัดตำแหน่งด้วยเทคนิค RTK ที่ใช้สถานีฐานชนิด 1 ความถี่ ระยะ Baseline 3.50 กิโลเมตร ในการปักหมุดแต่ละจุดจะมีค่าไม่เกิน 4 เซนติเมตร



รูปที่ 4.22 ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ด้วยระยะ Baseline 3.5 กิโลเมตร

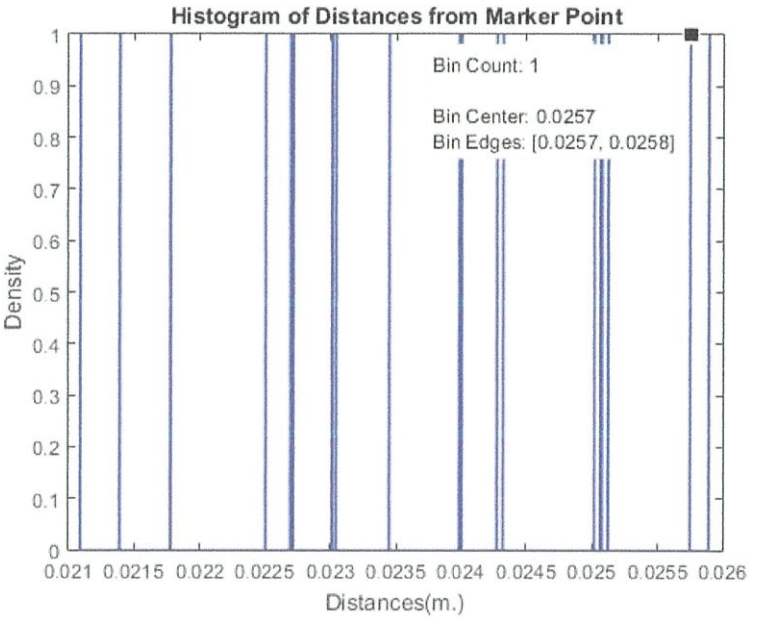
จากรูปที่ 4.22 ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK มีค่าไม่เกิน 4 เซนติเมตร แม้ว่าร้อยละของ Fixed Solution จะลดลงแต่ก็ถูกทดแทนด้วยการติดตั้งเสาที่ง่ายต่อการเล็งจุดปักหมุดมากขึ้น

ต่อมาจึงทำการคำนวณตำแหน่งด้วยเทคนิค RTK ด้วยระยะ Baseline 6.5 กิโลเมตร ได้ผลการทดลองดังตารางที่ 4.5

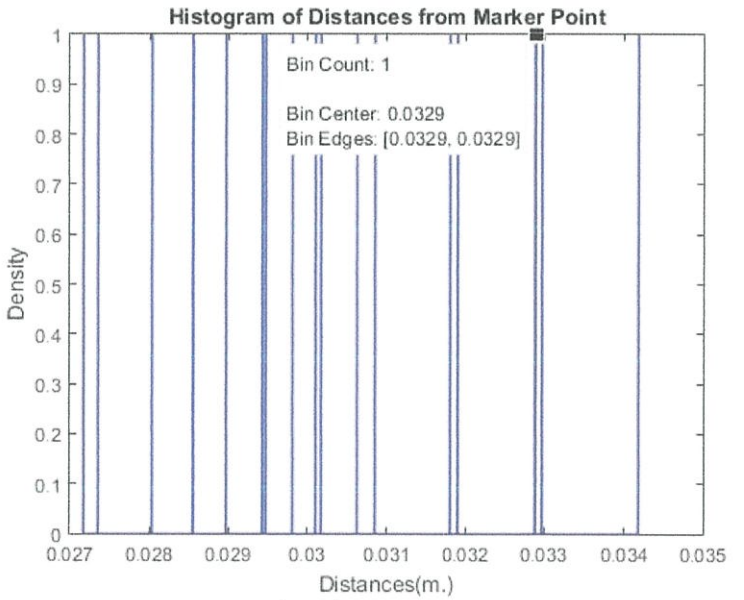
ตารางที่ 4.5 การคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงด้วยสถานีฐานชนิดความถี่เดียวด้วยระยะ Baseline 6.5 กิโลเมตร (ระยะ Baseline ไม่เกิน กิโลเมตรที่จุด D)

RTK Testing with Baseline 6.5 km. (Single Frequency)			
Marker	95% Error (cm.)	STDV (cm.)	%Fixed Solution
A	2.58	0.14	22.08
B	3.3	0.2	
C	3.97	0.15	
D	3.63	0.21	
95% Error total (cm.)	3.89		
STDV total (cm.)	0.47		

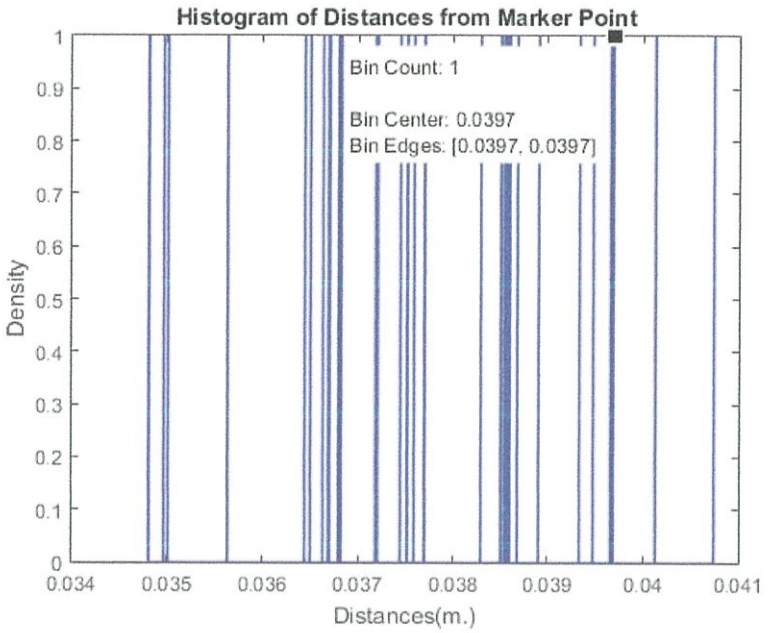
จากตารางที่ 4.5 ผลจากการทดสอบ พบว่ามีร้อยละของ Fixed Solution ลดลง โดยมีค่าเป็นร้อยละ 22.80 และมีความคลาดเคลื่อนของทั้งการทดลองเพิ่มขึ้นเป็น 3.89 เซนติเมตร ค่าความคลาดเคลื่อนของแต่ละจุดปักหมุดจากตารางที่ 4.5 สามารถหาได้จากฮิสโตแกรมในรูปที่ 4.23 ถึง 4.26 และมีความคลาดเคลื่อนตลอดทั้งการทดลองดังรูปที่ 4.27



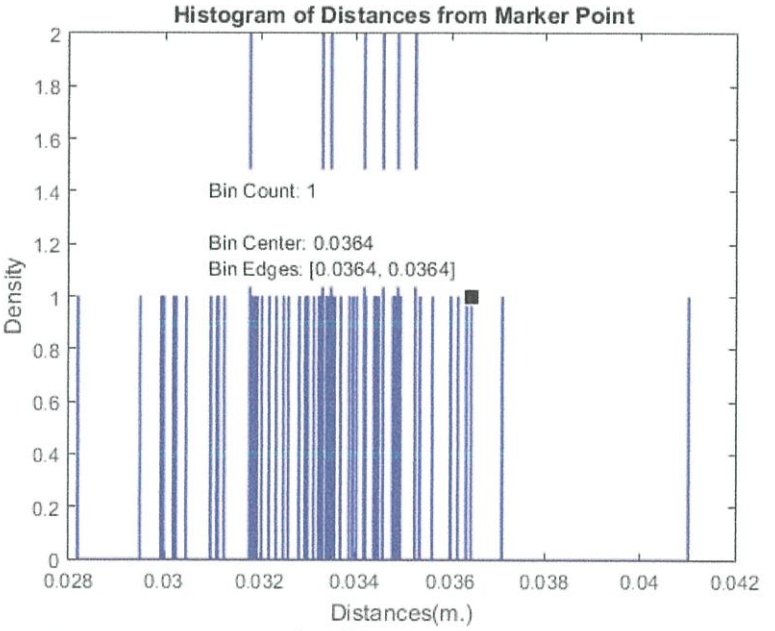
รูปที่ 4.23 ฮิสโตแกรมความคลาดเคลื่อนที่จุด A (ระยะ Baseline เท่ากับ 6.580 กิโลเมตร)



รูปที่ 4.24 ฮิสโตแกรมความคลาดเคลื่อนที่จุด B (ระยะ Baseline เท่ากับ 6.589 กิโลเมตร)

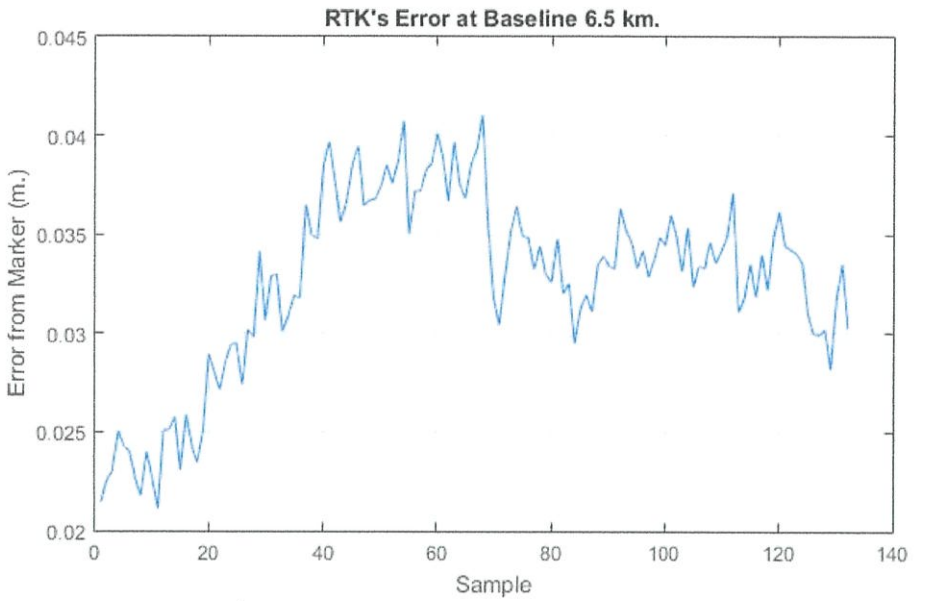


รูปที่ 4.25 ฮิสโตแกรมความคลาดเคลื่อนที่จุด C (ระยะ Baseline เท่ากับ 6.599 กิโลเมตร)



รูปที่ 4.26 ฮิสโตแกรมความคลาดเคลื่อนที่จุด D (ระยะ Baseline เท่ากับ 6.610 กิโลเมตร)

จากรูปที่ 4.23 ถึง 4.26 เป็นฮิสโตแกรมสำหรับหาค่า 95% Error ในตารางที่ 4.5 จากฮิสโตแกรมดังกล่าว ค่าความคลาดเคลื่อนในการระบุพิกัดตำแหน่งด้วยเทคนิค RTK ที่ใช้สถานีฐานชนิด 1 ความถี่ ระยะ Baseline 6.50 กิโลเมตร ในการปักหมุดแต่ละจุดจะมีค่าไม่เกิน 4 เซนติเมตร



รูปที่ 4.27 ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ด้วยระยะ Baseline 6.5 กิโลเมตร

จากรูปที่ 4.27 เนื่องจากระยะ Baseline ที่ไกลขึ้น จะส่งผลโดยตรงต่อความแม่นยำในการทำ RTK ทำให้ร้อยละของ Fixed Solution ลดลง และความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK เพิ่มขึ้น

เมื่อนำความคลาดเคลื่อนของการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงที่แต่ละระยะ Baseline ด้วยการใช้สถานีฐานชนิดความถี่เดียวมาแสดงเทียบกัน จะได้ผลดังตารางที่ 4.6

ตารางที่ 4.6 ความคลาดเคลื่อนของการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงด้วยสถานีฐานชนิดความถี่เดียว ด้วยสถานีฐานชนิดความถี่เดียว ด้วยระยะ Baseline 50 เมตร 808 เมตร 3.5 กิโลเมตร และ 6.5 กิโลเมตร

RTK's Testing in Straight Line 30 m. (Single Frequency)			
Baseline	95% Error (cm.)	STDV (cm.)	%Fixed Solution
50 m.	12.11	3.28	40.93
808 m.	4.49	1.24	42.56
3.5 km.	3.78	1.15	29.8
6.5 km.	3.89	0.47	22.08

จากตารางที่ 4.6 พบว่าเมื่อระยะ Baseline ไกลขึ้น ความแม่นยำจากเทคนิค RTK ในแนวเส้นตรงโดยรวมจะสูงขึ้น ร้อยละของ Fixed Solution มีแนวโน้มลดลง โดยทั่วไปแล้วค่าความคลาดเคลื่อนจะเพิ่มขึ้นตามระยะ Baseline แต่เนื่องจากการทดสอบที่ระยะ Baseline 50 m. และระยะ Baseline 808 m.อยู่ในขั้นตอนการปรับปรุงการวางเสาที่ใช้ติดตั้งสายอากาศ ทำให้ค่า 95% Error ในครึ่งแรกมีแนวโน้มไม่สัมพันธ์กับระยะ Baseline ขณะที่ผลการทดสอบด้วยระยะ Baseline 3.5 กิโลเมตรและ 6.5 กิโลเมตร มีแนวโน้มของค่า 95% Error เพิ่มขึ้นตามระยะ Baseline ในกรณีที่นำพิกัดซึ่งไม่ใช่ Fixed Solution มาคำนวณ พบว่าความแม่นยำในการระบุตำแหน่งที่ระยะ Baseline 50 เมตร 808 เมตร 3.5 กิโลเมตร และ6.5 กิโลเมตร มีค่าเท่ากับ. 15.02 เซนติเมตร 8.03 เซนติเมตร 16.82 เซนติเมตร และ 4.10 เซนติเมตรตามลำดับ ซึ่งมีความแม่นยำน้อยกว่าการใช้พิกัดที่เป็น Fixed Solution มาคำนวณ

จากผลลัพธ์ที่ได้ ระยะ Baseline ที่เหมาะสมต่อการใช้งาน ควรมีค่าไม่เกิน 6.5 กิโลเมตร เนื่องจากความแม่นยำในการระบุตำแหน่งด้วยเทคนิค RTK เริ่มมีค่าลดลง และใช้เวลาในการหาพิกัดที่เป็น Fixed Solution ยาวนาน และอาจความไม่สะดวกในการใช้งานในกรณีที่มีเวลาอยู่อย่างจำกัดได้

หลังจากคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงแล้ว จะคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมที่มีรัศมี 2 เมตรรอบจุด B เพื่อวิเคราะห์แนวโน้มของความคลาดเคลื่อนกับระยะ Baseline ขณะที่สถานีจลน์กำลังเคลื่อนที่ โดยจะแบ่งออกเป็น 3 การทดสอบ ได้แก่ การทดสอบที่ระยะ Baseline 808 เมตร 3.5 กิโลเมตร และ 6.5 กิโลเมตร ตามลำดับ

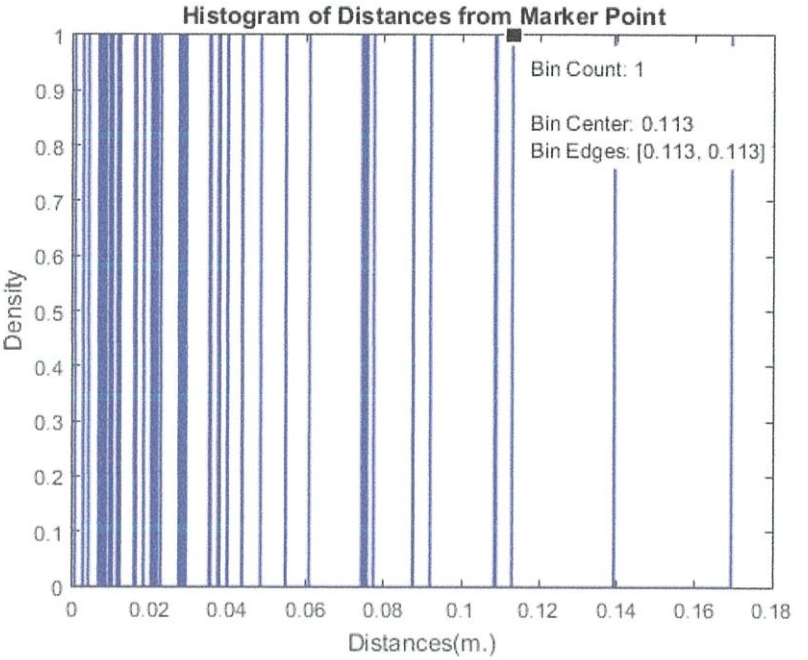
ในกรณีของการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยรัศมี 2 เมตร ด้วยระยะ Baseline 808 เมตร 3.5 กิโลเมตร และ 6.5 กิโลเมตร จะเป็นดังตารางที่ 4.7

ตารางที่ 4.7 ความคลาดเคลื่อนของการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมรัศมี 2 เมตร ด้วยสถานีฐานชนิดความถี่เดียว ด้วยระยะ Baseline 808 เมตร 3.5 กิโลเมตร และ 6.5 กิโลเมตร

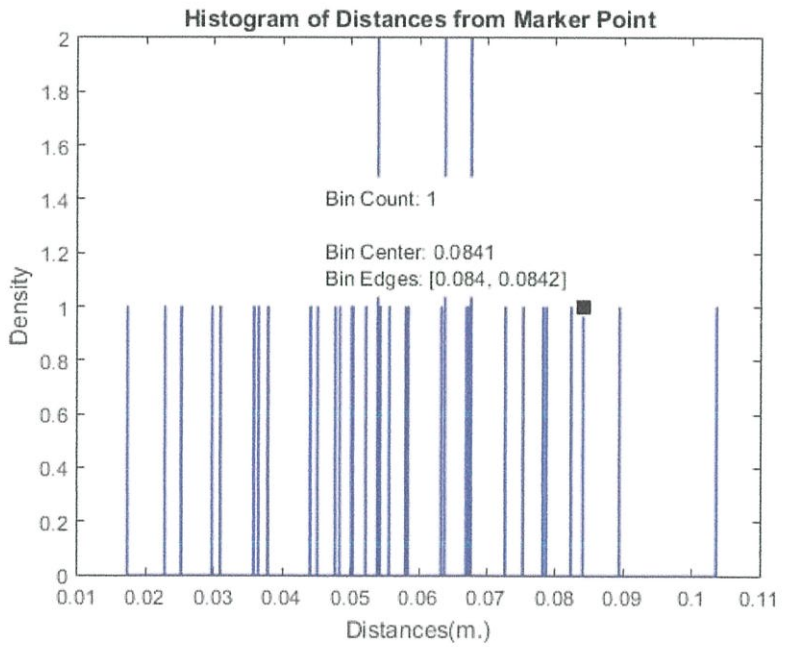
RTK's Testing in Circle Line (Center at B)			
Baseline	95% Error (cm.)	STDV (cm.)	%Fixed Solution
808 m.	11.28	4.03	33.72
3.5 km.	8.4	1.98	25.47
6.5 km.	6.67	1.94	20

จากตารางที่ 4.7 การคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมที่มีรัศมี 2 เมตร ด้วยระยะ Baseline มากขึ้น จะได้ค่าร้อยละของ Fixed Solution ที่ลดลง เนื่องจากสถานีฐานและสถานีจลน์อยู่ห่างกันมากขึ้น ในขณะระยะคลาดเคลื่อนมีแนวโน้มลดลงเมื่อระยะ Baseline เพิ่มขึ้น

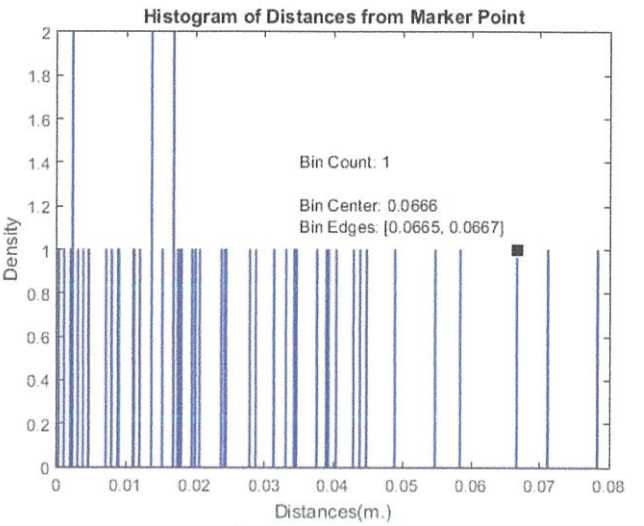
ค่า 95% Error ของการทดสอบในแต่ละ Baseline สามารถหาได้จากค่าที่ตำแหน่ง Bin Center ที่ทำเครื่องหมายไว้บนฮิสโตแกรมความคลาดเคลื่อนดังรูปที่ 4.28 ถึง 4.30



รูปที่ 4.28 ฮิสโตแกรมความคลาดเคลื่อนรัศมีรอบจุด B ด้วยระยะ Baseline 808 เมตร



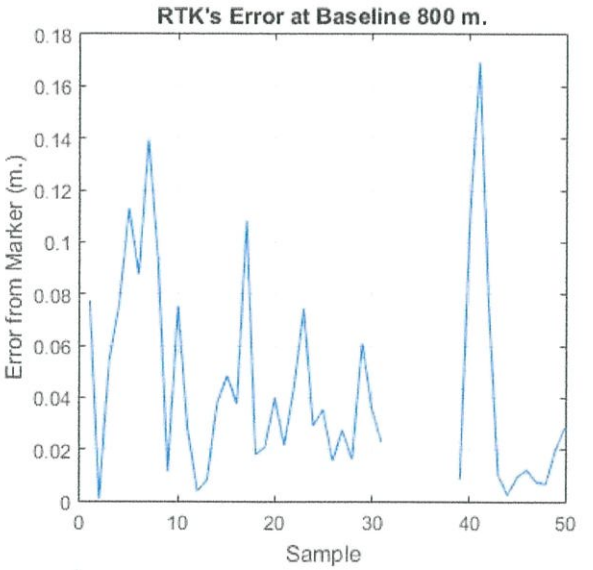
รูปที่ 4.29 ฮิสโตแกรมความคลาดเคลื่อนรัศมีรอบจุด B ด้วยระยะ Baseline 3.5 กิโลเมตร



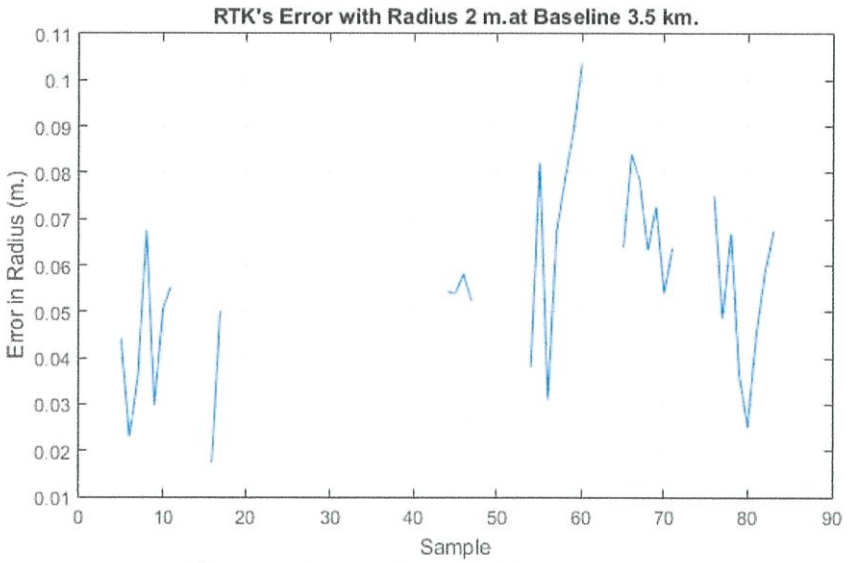
รูปที่ 4.30 ฮิสโตแกรมความคลาดเคลื่อนรัศมีรอบจุด B ด้วยระยะ Baseline 6.5 กิโลเมตร

จากรูปที่ 4.28 ถึง 4.30 แกนนอน (Distances) เป็นระยะคลาดเคลื่อนจากการทดสอบในหน่วยเมตร ค่าความคลาดเคลื่อนในการระบุพิกัดตำแหน่งด้วยเทคนิค RTK ในแนววงกลมรัศมี 2 เมตร ที่ใช้สถานีฐานชนิด 1 ความถี่ ระยะ Baseline 808 เมตร 3.50 กิโลเมตร 6.50 กิโลเมตร มีค่าไม่เกิน 12 เซนติเมตร

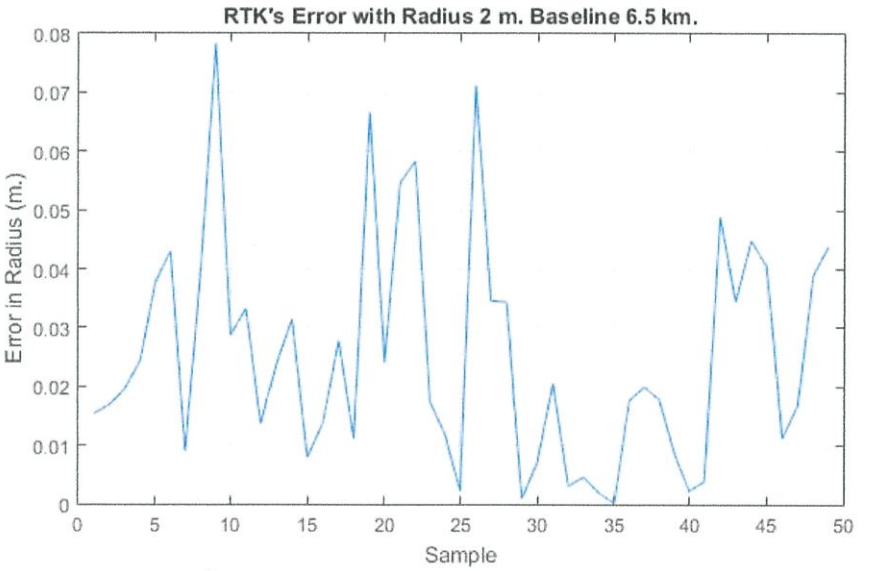
ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมทุกระยะ Baseline ตลอดการทดสอบ จะเป็นดังรูปที่ 4.31 ถึง 4.33



รูปที่ 4.31 ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยระยะ Baseline 808 เมตร



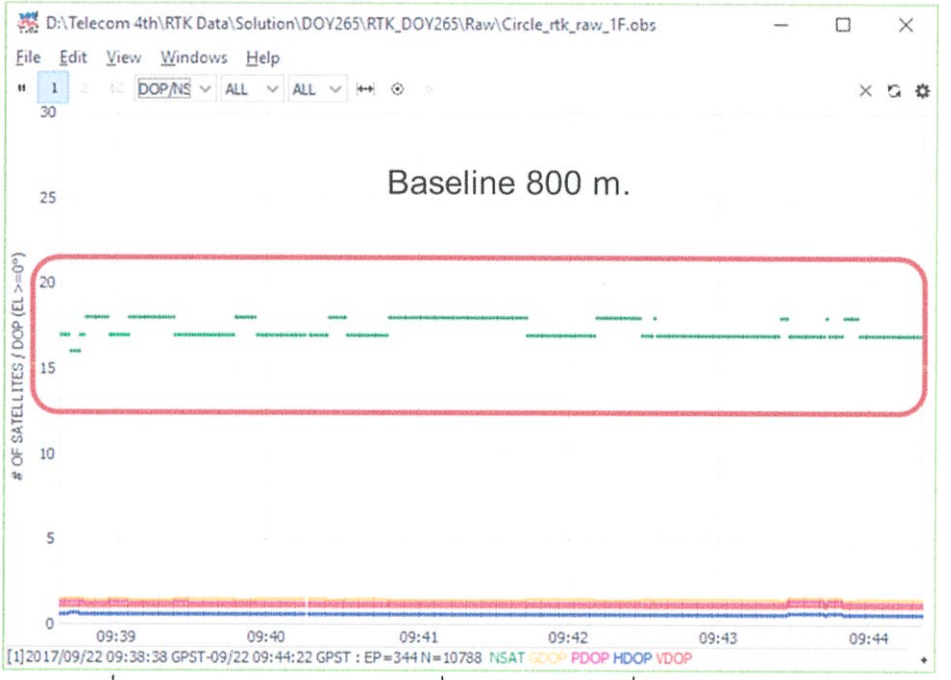
รูปที่ 4.32 ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยระยะ Baseline 3.5 กิโลเมตร



รูปที่ 4.33 ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยระยะ Baseline 6.5 กิโลเมตร

จากรูปที่ 4.31 ถึง 4.33 ที่ระยะ Baseline 808 เมตร มีร้อยละ Fixed Solution มาก จึงทำให้กราฟความคลาดเคลื่อนมีความต่อเนื่องเกือบทั้งกราฟ แต่มีระยะคลาดเคลื่อนค่อนข้างสูง ที่ระยะ Baseline 3.5 กิโลเมตร มีร้อยละ Fixed Solution ลดลง ทำให้กราฟระยะคลาดเคลื่อนขาดตอนมากกว่าระยะ Baseline 808 เมตร และในระยะ Baseline 6.5 กิโลเมตร การกระจายของ

ระยะคลาดเคลื่อนน้อยที่สุดจากทุกระยะ Baseline จึงทำให้มีความแม่นยำมากที่สุด จากการทดสอบนี้ ระยะ Baseline มีผลต่อระยะคลาดเคลื่อนน้อยมาก ปัจจัยที่มีผลต่อระยะคลาดเคลื่อนในการทดสอบ สามารถอธิบายได้ด้วยรูปที่ 4.34 ถึง 4.36

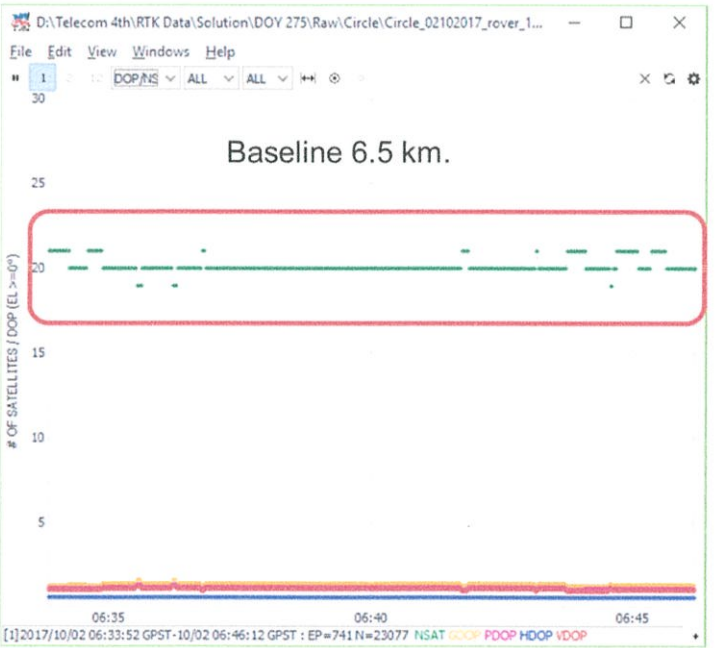


รูปที่ 4.34 รูปแบบกลุ่มดาวเทียมที่สถานีจลน์รับได้ที่ Baseline 808 เมตร

จากรูปที่ 4.34 ขณะคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยระยะ Baseline 808 เมตร จำนวนดาวเทียมที่สถานีจลน์รับได้อยู่ในช่วง 16 ถึง 18 ดวง สังเกตได้จากกรอบสีแดง แต่มีความไม่เสถียร เนื่องจากจำนวนดาวเทียมที่สถานีจลน์รับได้มีการเปลี่ยนแปลงไปตามเวลาค่อนข้างมาก

รูปที่ 4.35 รูปแบบกลุ่มดาวเทียมที่สถานีจลน์รับได้ที่ Baseline 3.5 กิโลเมตร

จากรูปที่ 4.35 ขณะคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยระยะ Baseline 3.5 กิโลเมตร จำนวนดาวเทียมที่สถานีจลน์รับได้อยู่ในช่วง 16 ถึง 18 ดวง สังเกตได้จากกรอบสีแดง แม้มีระยะ Baseline ที่ไกลกว่า แต่มีความเสถียรเนื่องจากการเปลี่ยนแปลงของจำนวนดาวเทียมที่สถานีจลน์สามารถรับได้น้อยกว่า จึงทำให้มีความแม่นยำมากกว่า



รูปที่ 4.36 รูปแบบกลุ่มดาวเทียมที่สถานีจลน์รับได้ที่ Baseline 6.5 กิโลเมตร

จากรูปที่ 4.36 ขณะคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยระยะ Baseline 6.5 กิโลเมตร จำนวนดาวเทียมที่สถานีจลน์รับได้อยู่ในช่วง 19 ถึง 21 ดวง สังเกตได้จากกรอบสีแดง ซึ่งมากกว่าการทดสอบก่อนหน้า และมีความเสถียรในการรับสัญญาณดาวเทียมมากกว่า เนื่องจากการเปลี่ยนแปลงของจำนวนดาวเทียมที่สถานีจลน์สามารถมองเห็นได้ในแต่ละเวลาน้อยกว่าการทดสอบที่แล้ว จึงทำให้มีความแม่นยำมากที่สุด

เพื่อตรวจสอบความแม่นยำในการทำ RTK ของสถานีฐานชนิดความถี่เดียว จะนำผลการทดสอบไปเปรียบเทียบกับผลการคำนวณตำแหน่งด้วยเทคนิค RTK ด้วยสถานีฐานชนิดสองความถี่ เพื่อวิเคราะห์ความแม่นยำการใช้สถานีฐานทั้งสองสำหรับการคำนวณตำแหน่งด้วยเทคนิค RTK ทั้งในแนวเส้นตรง และในแนววงกลมได้ผลดังตารางที่ 4.8 และตารางที่ 4.9

ตารางที่ 4.8 การทำ RTK ในแนวเส้นตรงของสถานีฐานชนิดความถี่เดียวและชนิดสองความถี่ ด้วยระยะ Baseline 50 เมตร 808 เมตร 3.5 กิโลเมตร และ 6.5 กิโลเมตร

Baseline	RTK's Testing in Straight Line 30 m.					
	Single Frequency			Dual Frequency		
	95%Error (cm.)	STDV (cm.)	%Fixed Solution	95%Error (cm.)	STDV (cm.)	%Fixed Solution
50 m.	12.11	3.28	40.93	11.58	2.82	65.03
808 m.	4.49	1.24	42.56	2.74	0.36	87.73
3.5 km.	3.78	1.15	29.8	3.18	5.21	33.42
6.5 km.	3.89	0.47	22.08	4.08	1.32	12.55

จากตารางที่ 4.8 ค่าความคลาดเคลื่อนสูงสุดจากการใช้สถานีฐานชนิดความถี่เดียว มีค่ามากกว่าค่าความคลาดเคลื่อนสูงสุดจากการใช้สถานีฐานชนิดสองความถี่ แต่ความคลาดเคลื่อนโดยรวมทุกระยะ Baseline ของทั้งสองมีค่าแตกต่างกันไม่มาก แนวโน้มร้อยละของ Fixed Solution เมื่อเทียบกับระยะ Baseline ของทั้งสอง มีแนวโน้มลดลงเมื่อมีระยะ Baseline มากขึ้น

เมื่อพิจารณาผลการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมด้วยการใช้สถานีฐานชนิดความถี่เดียวและชนิดสองความถี่ จะได้ผลตามตารางที่ 4.9

ตารางที่ 4.9 การทำ RTK ในแนววงกลมของสถานีฐานชนิดความถี่เดียวและชนิดสองความถี่ ด้วยระยะ Baseline 808 เมตร 3.5 กิโลเมตร และ 6.5 กิโลเมตร

Baseline	RTK's Testing in Circle Line Radius 2 m. (Center at B)					
	Single Frequency			Dual Frequency		
	95%Error (cm.)	STDV (cm.)	%Fixed Solution	95%Error (cm.)	STDV (cm.)	%Fixed Solution
808 m.	11.28	4.03	33.72	8.27	2.55	85.39
3.5 km.	8.4	1.98	25.47	8.35	1.66	19.57
6.5 km.	6.67	1.94	20	5.3	1.55	97.03

จากตารางที่ 4.9 ขณะทำการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลม ที่ระยะ Baseline 808 เมตร จะมีรูปแบบและความเสถียรของกลุ่มดาวเทียมน้อยกว่าที่ระยะ Baseline 3.5 กิโลเมตร และที่ 6.5 กิโลเมตร ดังที่อธิบายในรูปที่ 4.30 ถึง 4.32 จากการทดสอบนี้ ทำให้ทราบว่ารูปแบบและความเสถียรของกลุ่มดาวเทียมที่สถานีจลน์มองเห็น มีผลต่อความคลาดเคลื่อนในการคำนวณตำแหน่งด้วยเทคนิค RTK มากกว่าระยะ Baseline แต่ความคลาดเคลื่อนการคำนวณตำแหน่งด้วยเทคนิค RTK ด้วยสถานีฐานทั้งสองชนิดโดยรวม จะมีค่าแตกต่างกันสูงสุดประมาณ 3 เซนติเมตร หมายความว่าในการคำนวณตำแหน่งด้วยเทคนิค RTK หากต้องการให้มีความแม่นยำในการทดสอบไม่เกิน 3 เซนติเมตร จะสามารถใช้งานสถานีฐานชนิดความถี่เดียวแทนการใช้สถานีฐานชนิดสองความถี่ได้

4.3 ผลการทดสอบการอ่านค่ามุมที่แสดงการเคลื่อนที่ของวัตถุโดยใช้เซนเซอร์ IMU รุ่น GY-85

การวัดความแม่นยำในการวัดมุม Roll Pitch และ Yaw จะใช้การบอกร้อยละความแตกต่างระหว่างค่าที่วัดได้กับค่าที่กำหนด สามารถคำนวณได้จากสมการที่ 4.1

$$\text{ร้อยละความแตกต่างในการวัดมุม} = \frac{2(a-b)}{(a+b)} \times 100 \qquad (4.1)$$

โดยที่ a = มุมที่กำหนดในหน่วยองศา

b = มุมที่อุปกรณ์เซนเซอร์ รุ่น GY-85 วัดได้ในหน่วยองศา

ผลการวัดมุม Roll ด้วยเซนเซอร์ รุ่น GY-85 จากมุม -15° ถึงมุม 15° โดยเปลี่ยนมุมไปครั้งละ 5° จะเป็นดังตารางที่ 4.13

ตารางที่ 4.10 ผลการวัดมุม Roll ด้วยเซนเซอร์ รุ่น GY-85

ค่า Roll ที่กำหนด	ค่า Roll ที่วัดได้จริง	ร้อยละความแตกต่างในการวัด
-15	-15.17	-1.13
-10	-10.06	-0.60
-5	-4.95	1.01
0	0.08	-200.00
5	5.18	-3.54
10	10.1	-1.00
15	15.17	-1.13

จากตารางที่ 4.10 ค่าร้อยละความแตกต่างของมุม Roll จากการวัดด้วยอุปกรณ์เซนเซอร์ รุ่น GY-85 กับมุม Roll ที่กำหนดนั้นค่อนข้างน้อย โดยมีค่าต่ำสุดที่ร้อยละ -0.60 แต่ที่มุม 0 องศา จะพบว่าค่า a ตามสมการที่ 4.1 จะเท่ากับ 0 ส่งผลให้การคำนวณมีร้อยละความแตกต่างในการวัดที่มุมดังกล่าว มีค่าเท่ากับร้อยละ -200 เสมอ

ผลการวัดมุม Pitch ด้วยเซนเซอร์ รุ่น GY-85 จากมุม -15° ถึงมุม 15° โดยเปลี่ยนมุมไปครั้งละ 5° จะเป็นดังตารางที่ 4.11

ตารางที่ 4.11 ผลการวัดมุม Pitch ตามมุมที่กำหนด

ค่า Pitch ที่กำหนด	ค่า Pitch ที่วัดได้จริง	ร้อยละความแตกต่างในการวัด
-15	-15.32	-2.11
-10	-10.14	-1.39
-5	-5.21	-4.11
0	0.3	-200.00
5	5.19	-3.73
10	10.06	-0.60
15	14.86	0.94

จากตารางที่ 4.11 ค่าความแตกต่างระหว่างมุม Pitch ที่วัดได้จากอุปกรณ์เซนเซอร์ รุ่น GY-85 และมุม Pitch ที่กำหนด มีค่าค่อนข้างน้อย โดยมีร้อยละความแตกต่างต่ำสุดเท่ากับร้อยละ -0.60 และที่มุม 0 องศา มีค่าร้อยละความแตกต่างในการวัดเท่ากับร้อยละ -200 สามารถอธิบายได้จากสมการที่ 4.1

จากผลการทดลองวัดมุม Roll และมุม Pitch ค่าร้อยละความแตกต่างเกิดขึ้นได้จากหลายสาเหตุ ได้แก่ ความฝืดของตัวปรับองศาการวัด ไม้วัดมุมมีเส้นเล็งมุมที่เล็ก รวมทั้งความ

ละเอียดในการแปลงสัญญาณแอนาล็อกเป็นดิจิตอลบนตัวอุปกรณ์เซนเซอร์ รุ่น GY-85 มีผลต่อการคำนวณหามุม Roll และมุม Pitch ด้วย

ในการวัดมุม Yaw จะเริ่มวัดจากมุม 0° ในทิศตามเข็มครั้งละ 10° ไปจนถึงมุม 350° มีผลการวัดดังตารางที่ 4.12

ตารางที่ 4.12 ผลการวัดมุม Yaw ด้วย GY-85

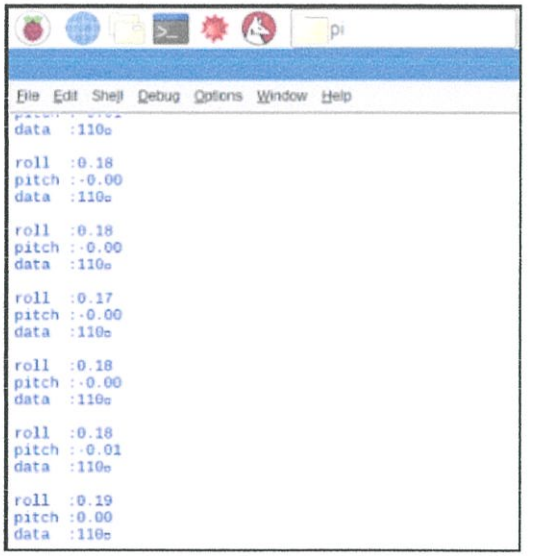
ค่า Yaw ที่กำหนด	ค่า Yaw ที่วัดได้จริง	ร้อยละความแตกต่างในการวัด
0	0	-
10	10	0.00
20	20	0.00
30	31	-3.28
40	40	0.00
50	51	-1.98
60	61	-1.65
70	71	-1.42
80	81	-1.24
90	91	-1.10
100	101	-1.00
110	111	-0.90
120	121	-0.83
130	131	-0.77
140	141	-0.71
150	150	0.00
160	160	0.00
170	170	0.00
180	180	0.00
190	190	0.00
200	201	-0.50
210	211	-0.48
220	220	0.00
230	232	-0.87
240	242	-0.83
250	252	-0.80

260	263	-1.15
270	272	-0.74
280	283	-1.07
290	293	-1.03
300	302	-0.66
310	312	-0.64
320	322	-0.62
330	332	-0.60
340	342	-0.59
350	351	-0.29

ผลจากการวัดมุม Yaw ตามตารางที่ 4.12 พบว่ามีค่าร้อยละความแตกต่างมีค่าไม่เกินร้อยละ 3.28 หรือมีความแม่นยำไม่เกิน 3° ซึ่งเพียงพอต่อการใช้งานสำหรับระบุทิศทางบนพื้นที่เพาะปลูกทางการเกษตร

4.4 ผลการทดสอบการเชื่อมต่อระหว่าง Arduino Nano V3 กับ Raspberry Pi

ผลจากการทำงานของโปรแกรม serial_test.py จะได้ผลดังรูปที่ 4.37



รูปที่ 4.37 ผลการทำงานของโปรแกรม serial_test.py

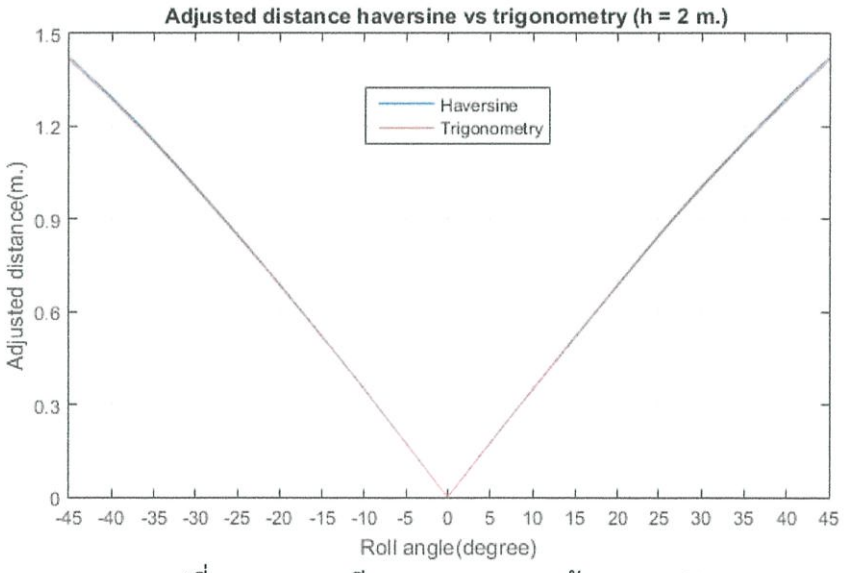
จากรูปที่ 4.37 ค่า roll คือค่า Roll ในหน่วยองศา ค่า pitch คือค่า Pitch ในหน่วยองศา และค่า data คือค่า Yaw ในหน่วยองศา โดยเป็นค่าที่ Arduino คำนวณแล้วส่งเข้าสู่

Raspberry Pi จากนั้นจึงถูกแสดงค่าขึ้นด้วยภาษาไพธอน ข้อมูลที่ส่งมาถูกส่งด้วยความเร็วสูง จึงต้องเขียนโปรแกรมเพื่อรอรับค่าแล้วนำไปใช้คำนวณแก้พิกัดที่เกิดจากความเอียงของสายอากาศต่อไป

4.5 ผลการทดสอบการจำลองการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศในโปรแกรม Matlab

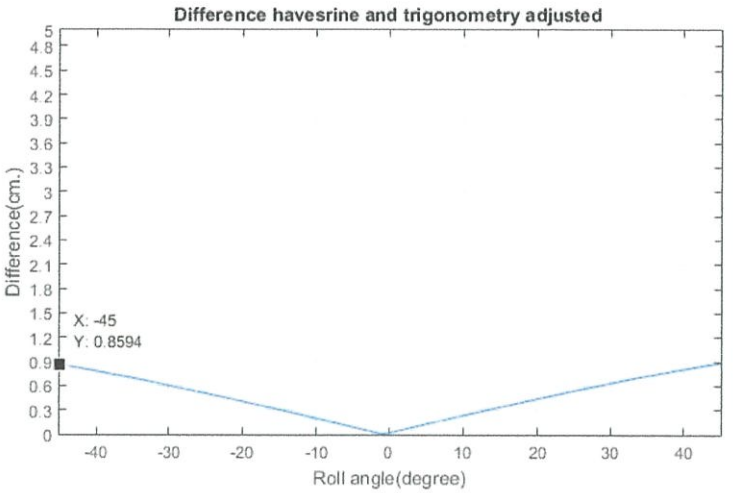
ในการจัดเก็บผลการทดสอบ จะใช้โปรแกรม Matlab เพื่อจำลอง และบันทึกผลการแก้ไขพิกัดของสายอากาศ โดยกำหนดให้ความสูงของสายอากาศจากพื้นดินเท่ากับ 2 เมตร (h = 2 m.) และกำหนดพิกัดสายอากาศที่ปลายเสาสำหรับติดตั้งสายอากาศเป็น [13.725704257792572°, 100.7788757714733° , 2 m.] จากนั้นสมมุติสายอากาศเอียงด้วยมุม Roll (แกนหมุนของมุม Roll ชี้ไปทางทิศเหนือ) มุม Pitch (แกนหมุนของมุม Pitch ชี้ไปทางทิศตะวันออก) และมุม Yaw (แกนหมุนของมุม Yaw ชี้ในทิศพุ่งขึ้นจากผิวโลก) เพื่อดูระยะห่างที่เกิดจากการเอียงของสายอากาศ จากนั้นนำไปหาผลต่างกับระยะเอียงตามสมการที่ 2.44 หลังจากเอียงสายอากาศในแต่ละมุมแล้วจะได้ผลการทดสอบดังนี้

4.5.1 ผลการจำลองเอียงสายอากาศด้วยมุม Roll จากมุม -45° จนถึง 45°



รูปที่ 4.38 ระยะเอียงของสายอากาศด้วยมุม Roll

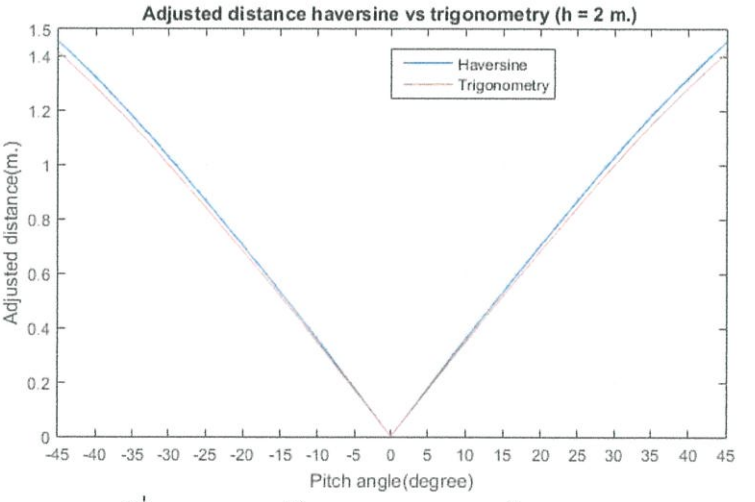
จากรูปที่ 4.38 พบว่าระยะเอียงของสายอากาศจากสมการที่ 2.36 (เส้น Haversine) จะมีระยะห่างระหว่างพิกัดของสายอากาศตั้งตรงกับสายอากาศที่ถูกเอียง) ใกล้เคียงกับระยะเอียงตามสมการที่ 2.37 (Trigonometry) และสามารถหาความแตกต่างของระยะได้ดังรูปที่ 4.39



รูปที่ 4.39 ผลต่างของระยะเอียงของอากาศด้วยมุม Roll เทียบกับระยะเอียงตามสมการที่ 2.37

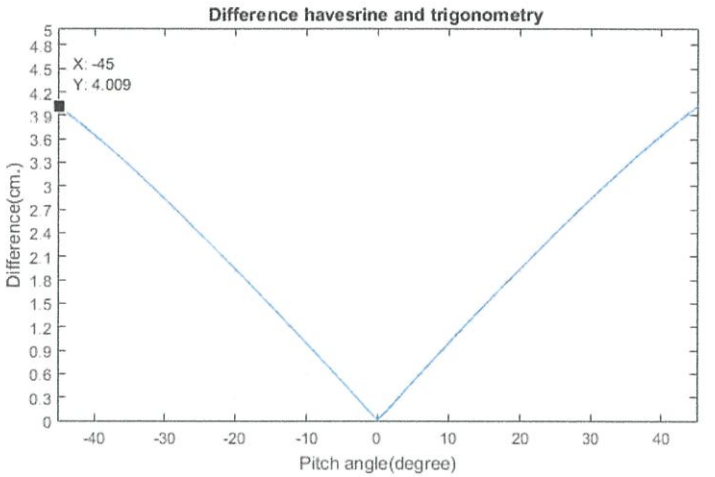
จากรูปที่ 4.39 พบว่าระยะเอียงของสายอากาศ มีความใกล้เคียงกับระยะเอียงในทางทฤษฎี โดยมีค่าต่างกันประมาณ 8.6 มิลลิเมตร สรุปได้ว่าสมการทางคณิตศาสตร์สามารถแก้ไขพิกัดของสายอากาศที่เอียงด้วยมุม Roll ได้อย่างถูกต้อง

4.5.2 ผลการจำลองเอียงสายอากาศด้วยมุม Pitch จากมุม -45° จนถึง 45°



รูปที่ 4.40 ระยะเอียงของสายอากาศด้วยมุม Pitch

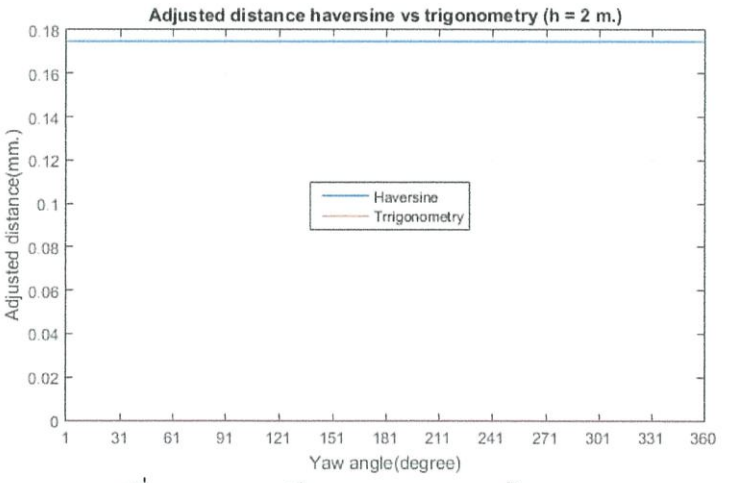
จากรูปที่ 4.40 พบว่าระยะเอียงของสายอากาศ (เส้น Haversine) มีความใกล้เคียงกับระยะเอียงตามสมการที่ 2.37 (เส้น Trigonometry) จากนั้นนำไปหาความแตกต่างด้วยกันจะได้ผลดังรูปที่ 4.41



รูปที่ 4.41 ผลต่างของระยะเอียงของอากาศด้วยมุม Pitch เทียบกับระยะเอียงตามสมการที่ 2.37

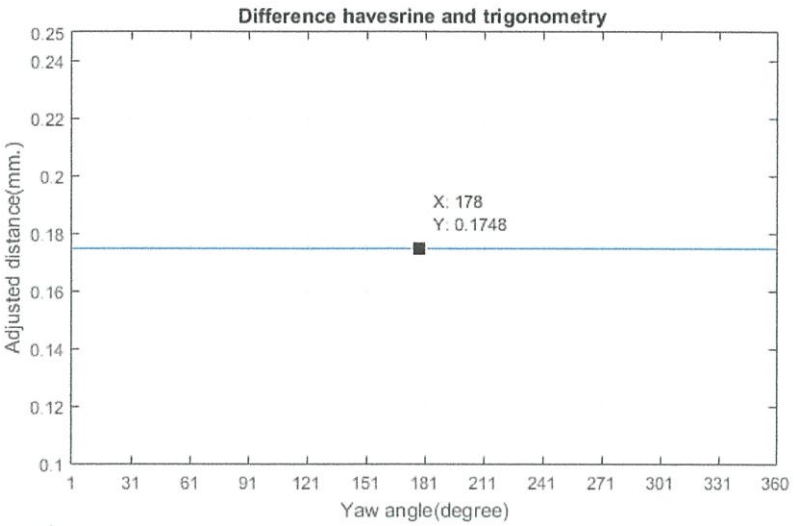
จากรูปที่ 4.41 พบว่าระยะเอียงของสายอากาศ มีความใกล้เคียงกับระยะเอียงในทางทฤษฎี โดยมีค่าประมาณ 4.01 เซนติเมตร สรุปได้ว่าสมการทางคณิตศาสตร์สามารถแก้ไขพิกัดของสายอากาศที่เอียงด้วยมุม Pitch ได้

4.5.3 ผลการจำลองเอียงสายอากาศด้วยมุม Yaw จากมุม 1° จนถึง 360°



รูปที่ 4.42 ระยะเอียงของสายอากาศด้วยมุม Yaw

จากรูปที่ 4.42 พบว่าระยะเอียงของสายอากาศด้วยมุม Yaw (เส้น Haversine) มีความแตกต่างกับระยะเอียงตามสมการที่ 2.37 (เส้น Trigonometry) ไม่เกิน 0.18 มิลลิเมตร สามารถแสดงได้ดังรูปที่ 4.43



รูปที่ 4.43 ผลต่างของระยะเอียงของอากาศด้วยมุม Yaw เทียบกับค่าศูนย์

จากรูปที่ 4.43 พบว่าระยะเอียงที่ได้จากการเอียงสายอากาศ มีความใกล้เคียงศูนย์ โดยมีค่าประมาณ 0.17 มิลลิเมตร สรุปได้ว่าสมการที่ 2.36 สามารถแก้ไขพิกัดของสายอากาศที่เอียงด้วยมุม Yaw ได้

เมื่อนำผลต่างระยะเอียงของสายอากาศที่ได้จากการใช้สมการที่ 2.36 เทียบกับระยะเอียงตามสมการที่ 2.37 (กำหนดให้เป็น Delta1) มาแสดง จะได้ผลดังตารางที่ 4.13

ตารางที่ 4.13 ผลต่างระยะเอียงที่ได้จากสมการที่ 2.36 กับระยะเอียงตามสมการที่ 2.37

ผลต่างระยะเอียงสูงสุด(m.)	Delta1
Roll[-45,45]	0.0086
Pitch[-45,45]	0.0401
Yaw[1,360]	0.0002

จากตารางที่ 4.13 สมการที่ 2.36 มีความคลาดเคลื่อนในการแก้ไขพิกัดสูงสุดประมาณ 4.01 เซนติเมตร สรุปได้ว่าผลการจำลองใช้สมการที่ 2.43 สามารถแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศได้ค่อนข้างถูกต้อง

4.6 ผลการทดสอบการเก็บพิกัดจากโปรแกรม RTKRCV ไปคำนวณใน Rotation Matrix

หลังจากโปรแกรม newedit.py เริ่มทำงานแล้ว จะเริ่มเก็บค่าพิกัดที่คำนวณได้จากโปรแกรม RTKRCV แล้วนำไปคำนวณหาค่าแก้ร่วมกับมุมเอียงจากเซนเซอร์ GY-85 จะได้ผลดังรูปที่ 4.44

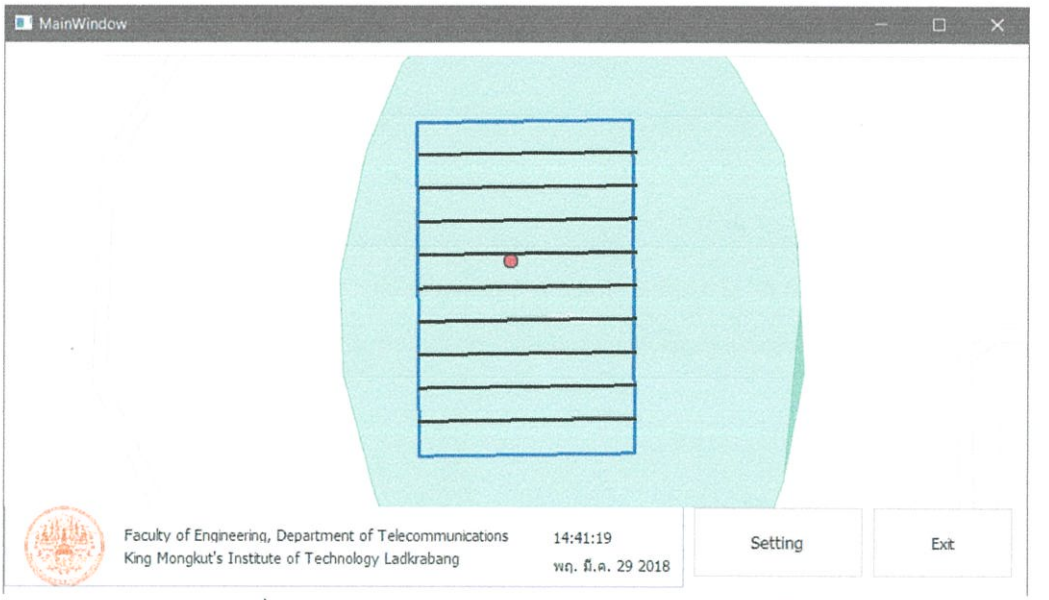
รูปที่ 4.44 ผลการทำงานของโปรแกรม newedit.py

จากรูปที่ 4.44 โปรแกรม newedit.py จะให้เอาต์พุตอยู่ 2 ข้อมูล ได้แก่ พิกัดแบบ ECEF ที่แก้ไขแล้ว (ในกรอบสีเขียว) ซึ่งเป็นพิกัดที่แท้จริงของรถทางเกษตรกรรม และระยะที่เกิดจากการเอียงของสายอากาศในหน่วยเมตร (ในกรอบสีแดง) ซึ่งจะนำไปเปรียบเทียบกับระยะเอียงทางตรีโกณมิติต่อไป

4.7 ผลการทดสอบระบบเฝ้าสังเกตการณ์และติดตามทางหน้าจอ

4.7.1 ผลการทดสอบหน้าต่างการใช้งานหลักของส่วนประสานงานกราฟฟิกผู้ใช้

หน้าต่างการใช้งานหลักของส่วนประสานงานกราฟฟิกผู้ใช้ตามที่ได้ทำการออกแบบจะประกอบไปด้วย 3 ส่วน คือ ส่วนแสดงสถานะการทำงาน ส่วนแสดงแผนที่และพิกัดตำแหน่ง และส่วนแสดงการแนะนำเส้นทาง โดยทำการสร้างส่วนประสานงานกราฟฟิกผู้ใช้ด้วยโปรแกรม QT หน้าต่างการใช้งานหลักที่ได้แสดงดังในรูปที่ 4.45

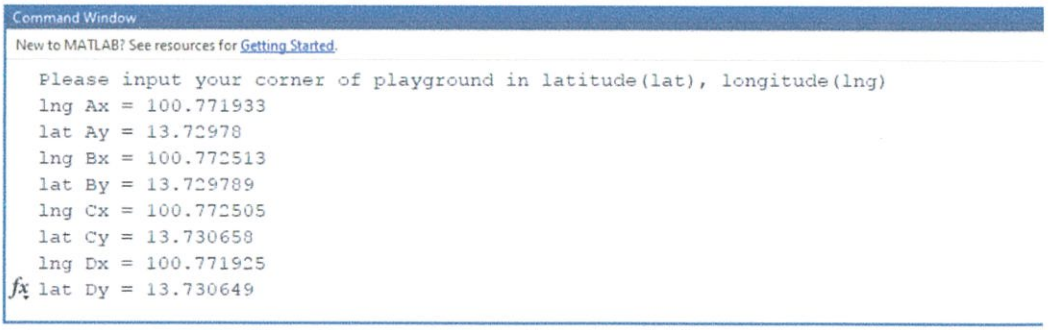


รูปที่ 4.45 หน้าต่างหลักส่วนประสานงานกราฟฟิกผู้ใช้

จากรูปที่ 4.45 แสดงหน้าต่างหลักส่วนประสานงานกราฟฟิกผู้ใช้ โดยผู้จัดทำได้ทำการกำหนดพื้นที่ที่สนใจเป็นบริเวณสนามกีฬา สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยมีจุดวงกลมสีแดง แสดงตำแหน่งของรถ ณ ปัจจุบัน เส้นตรงสีดำแสดงเส้นทางการเดินรถ และเส้นขอบสีน้ำเงินแสดงขอบเขตของพื้นที่ที่สนใจ มีการแสดงวันและเวลาในการทดลอง รวมถึงมีปุ่มกดเพื่อตั้งค่า (Configuration) ดังจะกล่าวในหัวข้อ 4.7.3

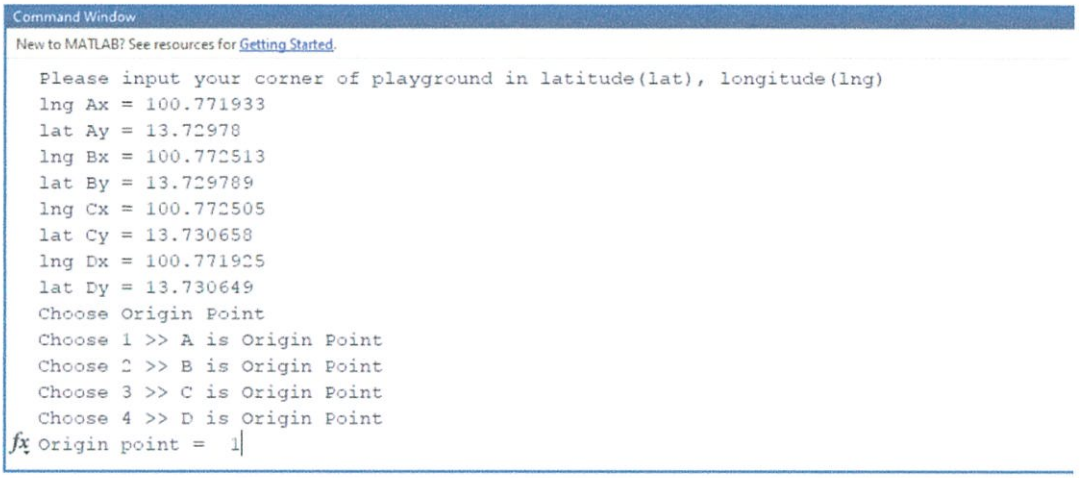
4.7.2 ผลการทดสอบส่วนแสดงแผนที่และพิกัดตำแหน่ง

เมื่อได้พื้นที่ที่สนใจจะทำการสร้างเส้นทางได้แล้ว ต้องทำการป้อนค่าพิกัดตำแหน่งมุมของพื้นที่ทั้ง 4 มุม ในโปรแกรมการคำนวณเส้นทางที่เขียนขึ้นด้วยโปรแกรม Matlab โดยเมื่อกดปุ่ม Run ในโปรแกรม Matlab ที่ทำการเขียนขึ้นแล้ว โปรแกรมจะให้ผู้ใช้งานป้อนค่าพิกัดตำแหน่งของมุมสนามทั้ง 4 มุม ดังแสดงในรูปที่ 4.46 (ลำดับจุด A - D เรียงลำดับตามรูปที่ 3.13)



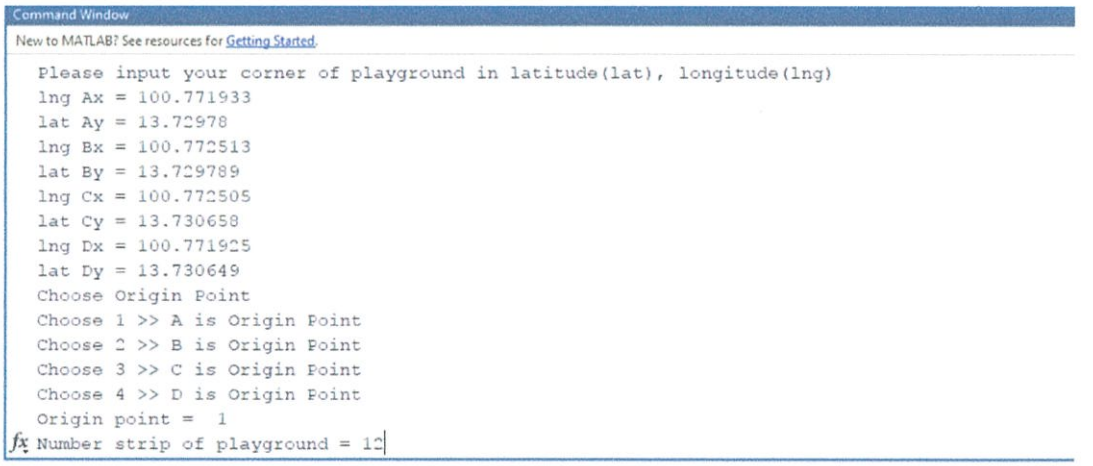
รูปที่ 4.46 การป้อนค่าพิกัดตำแหน่งของมุมสนามทั้ง 4 มุม

เมื่อทำการป้อนค่าพิกัดตำแหน่งของมุมสนามทั้ง 4 มุมดังรูปที่ 4.46 แล้ว โปรแกรมจะให้ผู้ใช้งานทำการเลือกจุดอ้างอิงที่จะใช้ในการคำนวณ ซึ่งจะขึ้นอยู่กับรูปแบบของเส้นทางที่จะมีทั้งรูปแบบเส้นแนวตั้งหรือเส้นแนวนอน ในที่นี้เลือกจุด A เป็นจุดอ้างอิงในการคำนวณ และแบ่งแนวเส้นทางในทิศทาง เหนือ - ใต้ การเลือกจุดอ้างอิงในการคำนวณแสดงดังรูปที่ 4.47



รูปที่ 4.47 การเลือกจุดอ้างอิงในการคำนวณ

เมื่อเลือกจุดอ้างอิงในการคำนวณดังรูปที่ 4.47 แล้ว โปรแกรมจะให้ผู้ใช้งานทำการป้อนจำนวนเส้นทาง (จำนวนลายเส้น) ในพื้นที่ที่สนใจ แสดงดังรูปที่ 4.48

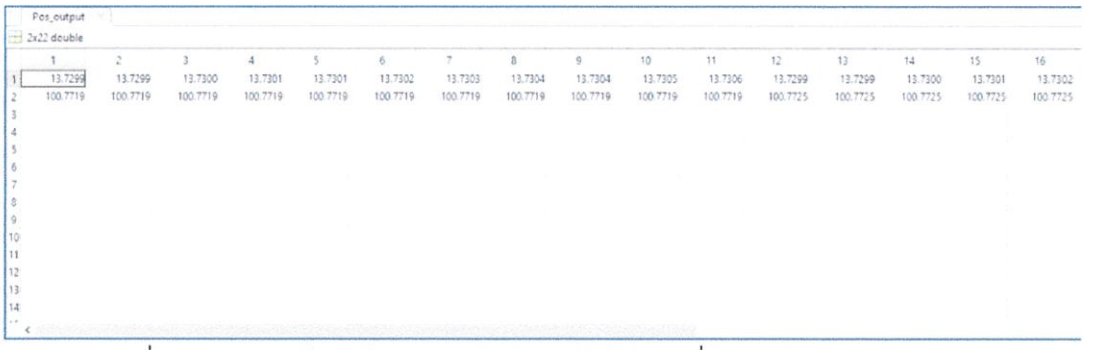


รูปที่ 4.48 การป้อนจำนวนเส้นทางให้กับโปรแกรม

เมื่อทำการป้อนจำนวนเส้นทางให้กับโปรแกรมดังรูปที่ 4.48 แล้ว จึงกดปุ่ม Enter จะได้จุดของเส้นทางที่ได้จากการคำนวณดังรูปที่ 4.49

รูปที่ 4.49 จุดของเส้นทางที่ได้จากการคำนวณ

จากรูปที่ 4.49 แสดงจุดของเส้นทางที่ได้จากการคำนวณด้วยโปรแกรม Matlab โดยจุดที่เป็นเครื่องหมาย + แสดงมุมทั้ง 4 จุดของสนาม และเครื่องหมาย o แสดงจุดของเส้นทาง โดยจะนำจุดพิกัดตำแหน่งที่ได้เป็นเอาต์พุตที่เก็บค่าในรูปแบบของเมทริกซ์ดังแสดงในรูปที่ 4.50

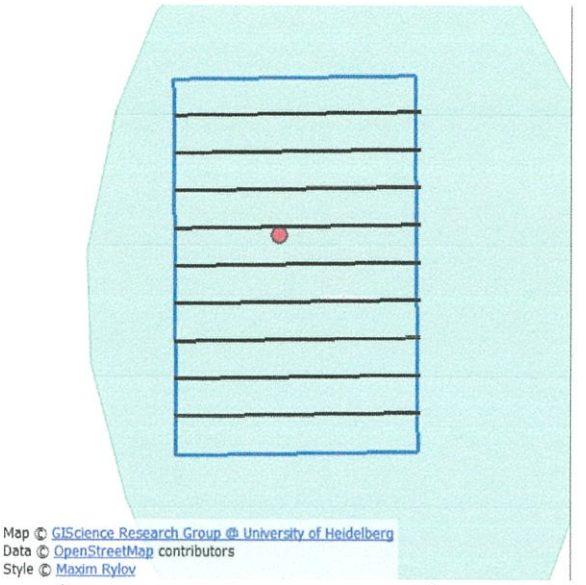

Pos_output

2x22 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	13.7299	13.7299	13.7300	13.7301	13.7301	13.7302	13.7303	13.7304	13.7304	13.7305	13.7306	13.7299	13.7299	13.7300	13.7301	13.7302
2	100.7719	100.7719	100.7719	100.7719	100.7719	100.7719	100.7719	100.7719	100.7719	100.7719	100.7719	100.7725	100.7725	100.7725	100.7725	100.7725

รูปที่ 4.50 เมทริกซ์เอาต์พุตของพิกัดตำแหน่งเส้นทางที่ได้จากโปรแกรม Matlab

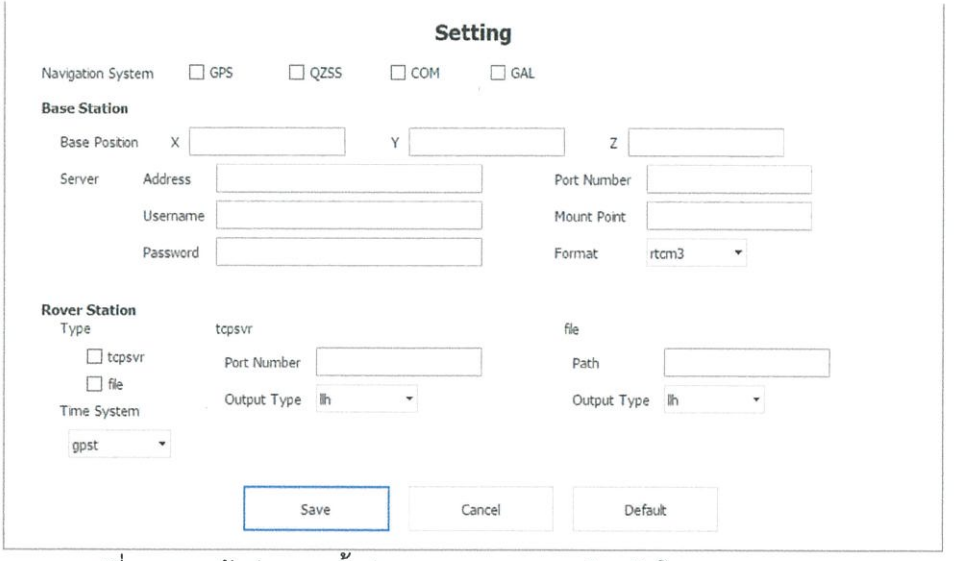
เมื่อได้พิกัดตำแหน่งของเส้นทางดังรูปที่ 4.50 แล้ว จึงไปทำการเขียนโปรแกรมแสดงเส้นทางร่วมกับแผนที่แบบ 2 มิติลงบนส่วนประสานงานกราฟฟิกผู้ใช้ ดังแสดงในรูปที่ 4.51



รูปที่ 4.51 การแสดงเส้นทางบนแผนที่แบบ 2 มิติ

จากรูปที่ 4.51 แสดงเส้นทางการเดินรถของรถทางเกษตรกรรม โดยเส้นทางในภาพสร้างมาจากการนำพิกัดตำแหน่งของจุดต้น - ปลาย ของเส้นทางแต่ละเส้นที่คำนวณได้จากรูปที่ 4.46 ถึงรูปที่ 4.50 มาใช้เป็นพิกัดอ้างอิงในการสร้างเส้นทางในรูปแบบเส้นตรงแบบจุดต่อจุดซ้อนทับบนแผนที่แบบ 2 มิติ ด้วยชุดคำสั่ง QML ภายในโปรแกรม QT

4.7.3 ผลการทดสอบส่วนการตั้งค่า Configuration สำหรับโปรแกรม RTKRCV

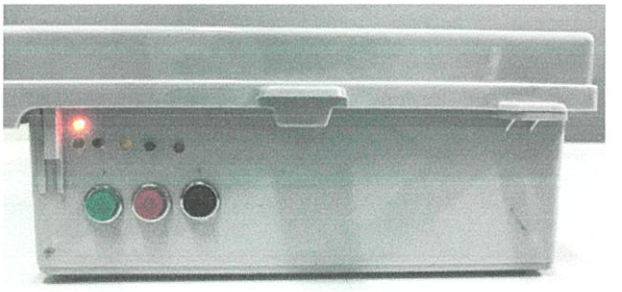


รูปที่ 4.52 หน้าต่างการตั้งค่า Configuration สำหรับโปรแกรม RTKRCV

จากรูปที่ 4.52 แสดงหน้าต่างการ Configuration โดยผู้ใช้งานจะทำการป้อนค่าพารามิเตอร์ต่างๆ ตามการทดลองที่ต้องการ หรือสามารถกดปุ่ม Default เพื่อเรียกใช้งานค่าเริ่มต้นก็ได้ การ Configuration ภายในหน้าต่างนี้ ผู้ใช้งานสามารถเลือกระบบดาวเทียมที่ต้องการใช้ในการทดลอง ซึ่งมีให้เลือกถึง 4 ระบบ คือ 1. GPS 2. QZSS (Quasi – Zenith Satellite System) 3. COM (BeiDuo) 4. GAL (Galileo) จากนั้นในส่วนของสถานีฐานผู้ใช้งานจะต้องป้อนค่าพิกัดตำแหน่งของ Base Station ในระบบ ECEF จากนั้นจะกำหนดการตั้งค่าขาออกผ่านทาง Server ซึ่งจะต้องป้อนค่า Address Username Password Port No. และ Mount Point และสุดท้ายจึงทำการเลือกรูปแบบว่าจะเป็น Rtcm3 สำหรับการทดลองแบบ Static หรือ Ubx สำหรับการทดลองแบบ Dynamic และสุดท้ายในส่วนของสถานีจลน์ ผู้ใช้งานสามารถเลือกประเภทของเอาต์พุตได้ว่าต้องการให้ส่งค่าออกทาง TCP Server หรือบันทึกเก็บเป็นไฟล์ หรือเลือกทั้ง 2 แบบ ถ้ามีการคลิกเลือกเอาต์พุตแบบ TCP Server ผู้ใช้งานต้องทำการป้อนเลขพอร์ตที่ต้องการ และชนิดของค่าพิกัด โดยมีให้เลือก 3 แบบ คือ LLH ECEF และ ENU และถ้ามีการคลิกเลือกประเภทของเอาต์พุตในรูปแบบของไฟล์ ผู้ใช้งานต้องทำการป้อน File Path ที่ต้องการ และเลือกชนิดของพิกัดเช่นเดียวกับเอาต์พุตแบบ TCP Server เมื่อทำการ Config ผ่านหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้ที่ออกแบบขึ้น ระบบก็จะสามารถใช้งานได้โดยไม่ต้องทำการ Config ผ่าน Command Line ซึ่งมีความซับซ้อน และยุ่งยาก และเมื่อนำไฟล์ Config ที่ได้ไปใช้งานในโปรแกรม RTKRCV ก็สามารถใช้ในการประมวลผลการรับค่าพิกัดตำแหน่งได้

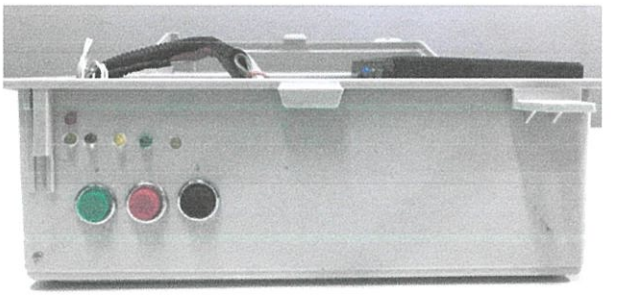
4.8 ผลการทดสอบวงจรควบคุมการจ่ายแบตเตอรี

ทำการทดสอบโดยการนำวงจรที่ทำการออกแบบเชื่อมเข้ากับอุปกรณ์สถานีจลน์ เริ่มจากการต่อแบตเตอรี 12 โวลต์ เข้าอุปกรณ์สถานีจลน์ จะเห็นได้ว่าที่ OP-AMP 741 แรงดันที่ขา 2 มีแรงดันมากกว่า แรงดันที่ขา 3 จะทำให้อุปกรณ์สถานีจลน์ แสดงสถานการณ์ทำงานของตัวอุปกรณ์ด้วยหลอด LED แสดงดังรูปที่ 4.53



รูปที่ 4.53 อุปกรณ์สถานีจลน์มีสถานะการทำงาน

เมื่อแบตเตอรีมีแรงดันที่ต่ำลงที่ประมาณ 11 โวลต์ อุปกรณ์ OP-AMP 741 ของวงจรควบคุมแบตเตอรี มีแรงดันที่ขา 2 น้อยกว่า แรงดันที่ขา 3 ทำให้ตัวรีเลย์ทำงานเปลี่ยนสถานะทำให้ตัดกระแส ออกจากอุปกรณ์สถานีฐาน แสดงดังรูปที่ 4.54

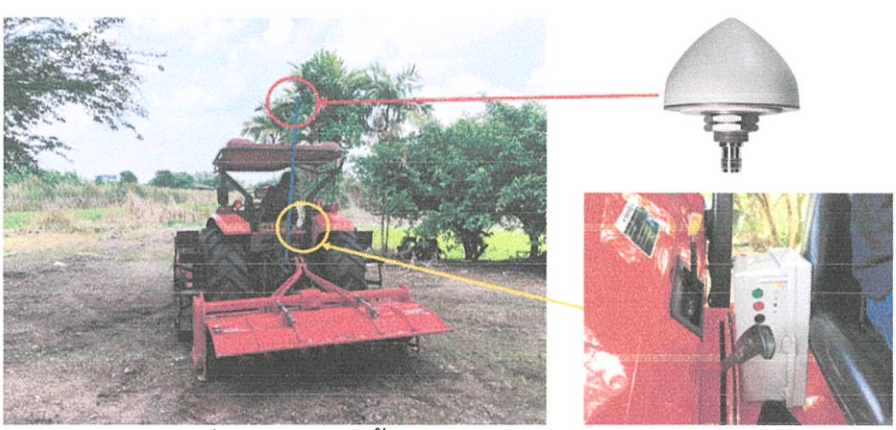


รูปที่ 4.54 อุปกรณ์สถานีจลน์มีสถานะหยุดทำงาน

ผลการทดสอบแสดงให้เห็นว่า วงจรควบคุมการจ่ายแบตเตอรี มีการทำงานของระบบในวงจร เป็นไปตามที่ต้องการ โดยมี OP-AMP 741 เป็นส่วนเปรียบเทียบแรงดัน และมี ตัวรีเลย์ ทำหน้าที่เป็นสวิตซ์ ที่คอยตัดการจ่ายแรงดันให้กับอุปกรณ์

4.9 ผลการทดสอบอุปกรณ์บนรถทางเกษตรกรรม

เมื่อได้ผลการทดลองจากหัวข้อที่ 4.1 ถึงหัวข้อที่ 4.9 แล้ว จึงนำอุปกรณ์ที่ออกแบบขึ้นไปทดสอบบนรถทางเกษตรกรรมที่ถูกใช้งานจริง เพื่อทดสอบความแม่นยำในการระบุค่าพิกัดตำแหน่ง ลักษณะการเอียงของสายอากาศ การซับแรงกระแทกของตัวรถและอุปกรณ์ที่ถูกติดตั้ง รวมถึงเสถียรภาพการทำงานของอุปกรณ์โดยรวม โดยตำแหน่งของการติดตั้งอุปกรณ์ต่างๆบนรถทางเกษตรกรรม แสดงดังรูปที่ 4.55

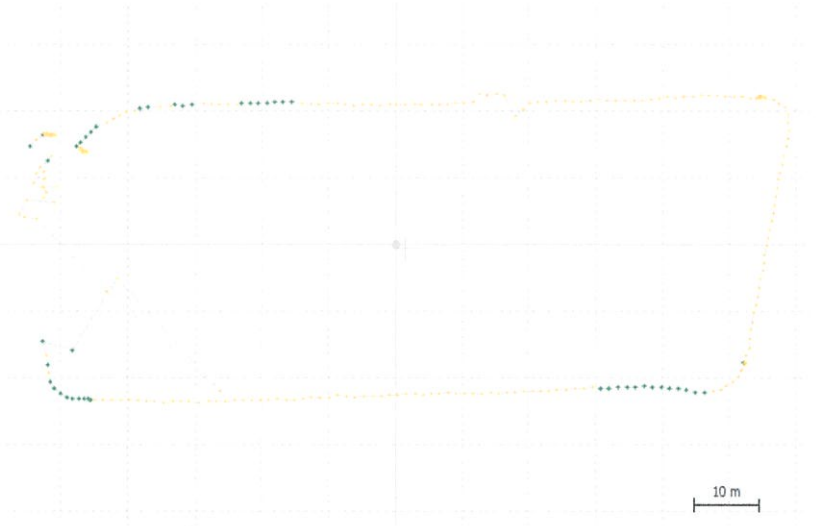


รูปที่ 4.55 การติดตั้งอุปกรณ์บนรถทางเกษตรกรรม

จากรูปที่ 4.55 ทำการติดตั้งสายอากาศไว้บริเวณหลังคารถ เชื่อมต่อกับอุปกรณ์ Ublox M8T เข้ามายังกล่องอุปกรณ์หลัก ซึ่งมีรายละเอียดแสดงดังรูปที่ 4.3 และถูกติดตั้งไว้บริเวณเบาะนั่งของคนขับ เมื่อติดตั้งอุปกรณ์ต่างๆ เสร็จเรียบร้อยแล้วจะทำการเคลื่อนที่รถทางเกษตรกรรมไปบนเส้นทางแปลงเกษตรกรรมจำลองรูปสี่เหลี่ยมที่มีขอบของพื้นที่เป็นเป็นเส้นตรง และตรวจสอบว่าผลการทดสอบการระบุค่าพิกัดตำแหน่งเป็นไปตามเส้นทางที่ได้ออกแบบไว้หรือไม่ โดยตัวอย่างของเส้นทางที่ใช้ทดสอบแสดงดังรูปที่ 4.56

รูปที่ 4.56 เส้นทางการทดสอบของรถทางเกษตรกรรม

เมื่อทำการทดสอบตามรูปที่ 4.55 และรูปที่ 4.56 เรียบร้อยแล้วจึงนำค่าพิกัดตำแหน่งที่บันทึกไว้มาแสดงบนโปรแกรม RTKPLOT เพื่อดูผลการทดสอบการรับค่าพิกัดตำแหน่งตามเส้นทาง แสดงดังรูปที่ 4.57



รูปที่ 4.57 ผลการทดสอบบนรถทางเกษตรกรรม

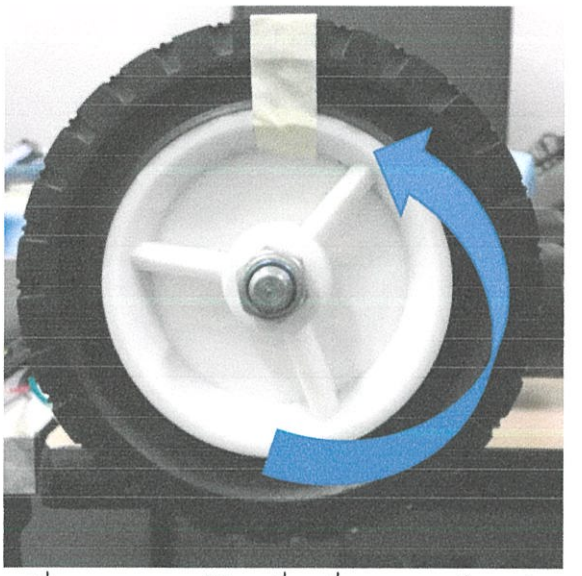
จากรูปที่ 4.57 แสดงผลการทดสอบบนเส้นทางในแปลงเกษตรกรรม ค่าพิกัดตำแหน่งที่ได้มีความใกล้เคียงกับเส้นทางรูปสี่เหลี่ยมที่มีขอบของพื้นที่เป็นเส้นตรง โดยช่วงจุดสีเขียวแสดงสถานะ Fixed Solution ซึ่งให้ความแม่นยำในการระบุพิกัดตำแหน่งมาก ช่วงจุดสีเหลืองแสดงสถานะ Float Solution ซึ่งมีความแม่นยำในการระบุพิกัดตำแหน่งน้อยกว่าช่วงจุดสีเขียว ซึ่งอาจเกิดจากการมีวัตถุบดบังการรับสัญญาณของสายอากาศ หรือจำนวนดาวเทียมที่รับสัญญาณได้มีจำนวนน้อยเกินไป และจะมีในบางช่วงของเส้นทางที่ไม่ปรากฏจุดพิกัดตำแหน่ง (ขอบขอบพื้นที่ทางด้านซ้าย) ซึ่งมีสาเหตุเนื่องจากพื้นที่ในช่วงนั้นมีต้นไม้สูงบดบังจำนวนมาก ทำให้สายอากาศไม่สามารถรับสัญญาณดาวเทียมได้

4.10 ผลการทดสอบการควบคุมพวงมาลัย

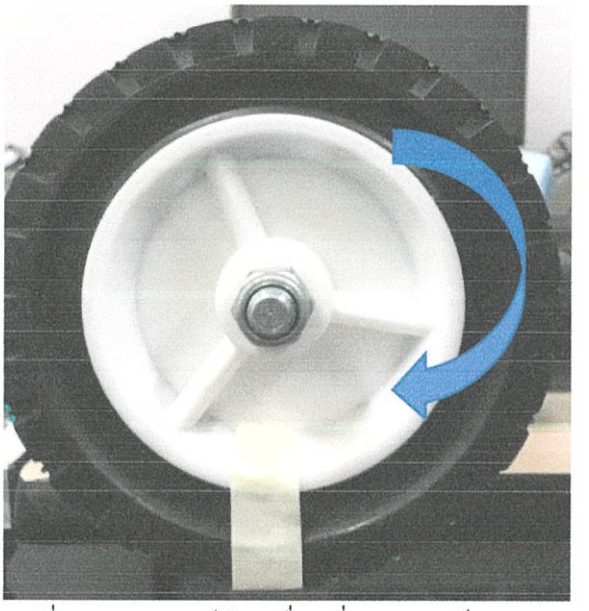
ก่อนสั่งการทำงานโปรแกรมในหัวข้อที่ 3.3.9 ตำแหน่งของมอเตอร์จะเป็นตามรูปที่ 4.58

รูปที่ 4.58 กำหนดให้จุดอ้างอิงของมอเตอร์ขับเคลื่อนอยู่ทางขวามือ

จากรูปที่ 4.58 แสดงจุดอ้างอิงของตัวมอเตอร์ขับเคลื่อน จากจุดอ้างอิงถ้าต้องการขับเคลื่อนพวงมาลัยไปทางขวาจะสั่งการหมุนของมอเตอร์ไปในทิศทางทวนเข็มนาฬิกา จะแสดงดังรูปที่ 4.59 จากจุดอ้างอิงถ้าต้องการหมุนพวงมาลัยไปทางซ้ายจะสั่งการหมุนของมอเตอร์ไปในทิศตามเข็มนาฬิกาดังรูปที่ 4.60



รูปที่ 4.59 มอเตอร์ขับเคลื่อนที่หมุนทวนเข็มนาฬิกา



รูปที่ 4.60 มอเตอร์ขับเคลื่อนที่หมุนตามเข็มนาฬิกา

จากรูปที่ 4.58 ถึง 4.60 จะนำมอเตอร์ควบคุมพวงมาลัยที่ทดสอบ มาทำการติดตั้งเข้ากับอุปกรณ์นำร่อง สถานีจลน์ เพื่อใช้ในการปรับแก้ไขการหมุนมอเตอร์ตามพิกัดที่มีการคำนวณตำแหน่งด้วยเทคนิค RTK ที่มีการรับค่าจากสถานีฐานเป็นจุดอ้างอิง เมื่อรถเกิดการวิ่งผิดออกนอกเส้นทาง ตัวมอเตอร์จะมีการทำงานหมุนให้พวงมาลัยกลับมาอยู่ตามเส้นทางที่กำหนดไว้

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

จากการทำปริญญานิพนธ์ฉบับนี้ สามารถแบ่งการสรุปผลออกได้เป็น 5 ส่วนหลัก คือ ส่วนการออกแบบสถานีฐาน และสถานีจลน์ ส่วนที่สองคือส่วนวงจรควบคุมการจ่ายแบตเตอรี่ ส่วนที่สามคือส่วนการประสิทธิภาพพัฒนาความแม่นยำในการระบุพิกัดตำแหน่ง ส่วนที่สี่คือระบบเฝ้าสังเกตการณ์และติดตามทางหน้าจอ และส่วนสุดท้ายคือการทดลองระบบ Auto Steering โดยสามารถสรุปผลได้ดังนี้

5.1.1 ส่วนการออกแบบอุปกรณ์สถานีฐาน และสถานีจลน์

การออกแบบอุปกรณ์สถานีฐาน และสถานีจลน์ มีการออกแบบในการวางอุปกรณ์ที่ใช้งานลงในบรรจุภัณฑ์ กำหนดคำสั่งการเปิด-ปิดตัวอุปกรณ์ ด้วย Raspberry Pi ด้วยภาษาไพธอน เพื่อใช้ในการคำนวณตำแหน่งด้วยเทคนิค RTK ในภาคสนาม ได้ตัวอุปกรณ์ที่ใช้งานสะดวก และทนทาน เป็นไปตามการออกแบบที่ต้องการ

5.1.2 ส่วนวงจรควบคุมการจ่ายแบตเตอรี่

วงจรควบคุมแบตเตอรี่ ทดสอบโดยทำการเชื่อมต่อแบตเตอรี่เข้ากับวงจรบนแผ่น PCB (Printed circuit board) ผลการทดสอบแหล่งจ่ายที่มีค่าแรงดันต่างกัน เพื่อแสดงให้เห็นถึงการเปลี่ยนแปลงของปริมาณแรงดันในแบตเตอรี่ จะส่งผลให้มีการถนอมแบตเตอรี่ ไม่ให้มีการเสื่อมสภาพเร็วเกินไป ผลการทดสอบวงจรควบคุมการจ่ายแบตเตอรี่ เป็นไปตามแบบที่ต้องการ มีการคุมการจ่ายแรงดัน จากแบตเตอรี่เข้าสู่อุปกรณ์

5.1.3 ส่วนของการพัฒนาความแม่นยำในการระบุตำแหน่ง

ในการพัฒนาความแม่นยำของการระบุตำแหน่ง จะใช้การคำนวณตำแหน่งด้วยเทคนิค RTK และการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศด้วยเซนเซอร์ GY-85 นี้ จะอยู่ในผลการทดลองจากบทที่ 4 ตั้งแต่หัวข้อที่ 4.2 ถึง 4.6 ซึ่งสามารถสรุปผลได้ดังนี้

5.1.3.1 การคำนวณตำแหน่งด้วยเทคนิค RTK ในระยะ Baseline 50 เมตร 808 เมตร 3.5 กิโลเมตร และ 6.5 กิโลเมตร

รูปแบบที่ใช้ในการคำนวณตำแหน่งด้วยเทคนิค RTK โดยใช้สถานีฐานชนิดความถี่เดียว มีทั้งรูปแบบเส้นตรง 30 เมตร และแบบวงกลมที่มีรัศมี 2 เมตรรอบจุด B ตามระยะ Baseline ที่กำหนดขึ้น ซึ่งแต่ละรูปแบบสามารถสรุปผลการทดสอบได้ดังนี้

จากผลการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนวเส้นตรงเป็นระยะทาง 30 เมตร จากตารางที่ 4.6 สามารถสรุปได้ว่า ความคลาดเคลื่อนโดยรวมของทุกการทดสอบ มีแนวโน้มเพิ่มขึ้น

ตามระยะ Baseline แต่เนื่องจากที่ระยะ Baseline 50 เมตร และ 808 เมตร อยู่ในขั้นปรับปรุงการวางเสาสำหรับติดตั้งสายอากาศ จึงทำให้มีความคลาดเคลื่อนสูงกว่าระยะ Baseline 3.5 กิโลเมตร และ 6.5 กิโลเมตร และค่าร้อยละของ Fix Solution มีแนวโน้มผกผันกับระยะ Baseline

จากผลการคำนวณตำแหน่งด้วยเทคนิค RTK ในแนววงกลมที่มีรัศมี 2 เมตร ในตารางที่ 4.9 สามารถสรุปได้ว่า ระยะ Baseline ส่งผลกระทบต่อความคลาดเคลื่อนในการคำนวณตำแหน่งด้วยเทคนิค RTK น้อยมาก ปัจจัยที่มีผลต่อความแม่นยำในการทดสอบมากที่สุด คือ รูปแบบของกลุ่มดาวเทียม จำนวนดาวเทียม และความเสถียรของดาวเทียมที่สถานีจลน์มองเห็นได้

ดังนั้นในการใช้งานจริง ควรติดตั้งเสาของสายอากาศให้มีความมั่นคง และสูงกว่าหลังคารถ เพื่อให้สายอากาศสามารถรับสัญญาณได้อย่างเต็มที่ ระยะ Baseline ที่เหมาะสมในการตั้งสถานีฐานควรมีค่าไม่เกิน 6.5 กิโลเมตรโดยประมาณ เพื่อความรวดเร็วในการหาพิกัดที่เป็น Fixed Solution ให้กับสถานีจลน์ และสามารถเริ่มการทดสอบได้รวดเร็วยิ่งขึ้น

5.1.3.2 การจำลองการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศ

จากการผลจำลองการแก้ไขพิกัดที่ผิดพลาดเนื่องจากความเอียงของสายอากาศในโปรแกรม Matlab ตามตารางที่ 4.12 สามารถสรุปได้ว่าสมการที่ 2.36 สามารถใช้แก้ไขความผิดพลาดพิกัดได้ค่อนข้างแม่นยำ เนื่องจากระยะเอียงที่ได้จากสมการแก้ไขพิกัดสายอากาศมีค่าต่างจากระยะเอียงตามสมการที่ 2.37 ซึ่งเป็นระยะเอียงทางทฤษฎี โดยมีค่าไม่เกิน 4 เซนติเมตร ซึ่งเป็นค่าที่น้อยมาก

5.1.3.3 การแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศด้วยเซนเซอร์ GY-85

จากผลการทดสอบในบทที่ 4 หัวข้อที่ 4.3 สรุปได้ว่ามุม Roll Pitch และ Yaw ที่คำนวณได้จากเซนเซอร์ GY-85 มีความแม่นยำอย่างน้อย 3° จากผลการทดสอบในหัวข้อที่ 4.6 สรุปได้ว่าสามารถนำค่ามุมดังกล่าว ไปคำนวณค่าแก้ความผิดพลาดพิกัดร่วมกับค่าพิกัดจากโปรแกรม RTKRCV พร้อมกับแสดงระยะเอียงของสายอากาศ โดยมีความผิดพลาดไม่เกิน 4 เซนติเมตร

5.1.4 ส่วนของระบบเฝ้าสังเกตการณ์และติดตามทางหน้าจอ

เมื่อทำการทดสอบหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้แล้ว หน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้ได้ถูกแบ่งพื้นที่ออกเป็น 3 ส่วนหลัก คือ ส่วนแสดงสถานะการทำงาน ส่วนแสดงแผนที่และพิกัดตำแหน่ง และส่วนการตั้งค่าโปรแกรม ในส่วนแสดงสถานะการทำงานสามารถแสดงผลเวลาปัจจุบัน ข้อมูลผู้จัดทำ ปุ่มกดตั้งค่าและปุ่มปิดโปรแกรม ซึ่งสามารถทำงานได้ตรงตามที่ได้ออกแบบ

ส่วนการแสดงแผนที่และพิกัดตำแหน่ง ชุดคำสั่งคำนวณเส้นทางด้วยโปรแกรม Matlab ที่จัดทำขึ้น สามารถทำการคำนวณเส้นทางได้โดยอัตโนมัติ โดยทำการป้อนเพียงจุดพิกัดมุมของพื้นที่ที่สนใจจำนวน 4 มุม และเมื่อนำจุดของเส้นทางที่ได้จากการคำนวณด้วยโปรแกรม

Matlab ไปสร้างเส้นทางลงบนแผนที่แบบ 2 มิติ เส้นทางที่ได้เป็นไปตามที่ออกแบบไว้ และเมื่อแสดงแผนที่บนส่วนประสานงานกราฟฟิกผู้ใช้ สามารถแสดงพิกัดตำแหน่งที่รับค่าจาก Raspberry Pi ได้อย่างถูกต้อง

ส่วนการตั้งค่าโปรแกรม เมื่อทำการป้อนค่าพารามิเตอร์ต่างๆ ที่ต้องการตั้งค่าลงในส่วนประสานงานกราฟฟิกผู้ใช้ โปรแกรมที่เขียนขึ้นสามารถนำค่าอินพุตดังกล่าวไปแก้ไขไฟล์ Config ได้อย่างถูกต้อง และใช้ในการประมวลผลการรับค่าพิกัดตำแหน่งด้วยโปรแกรม RTKRCV ได้

5.1.5 ส่วนของการทดสอบอุปกรณ์บนรถทางเกษตรกรรม

จากการทดสอบอุปกรณ์บนรถทางเกษตรกรรม ขณะทำการทดสอบ การะบุตำแหน่งของรถทางเกษตรกรรม จะมีความแม่นยำสูงเมื่อสาอากาศอยู่กลางที่โล่งแจ้ง และจะมีความแม่นยำลดลงขณะรถเคลื่อนที่เข้าสู่บริเวณที่มีต้นไม้สูงบดบัง ดังนั้นในการใช้งานจริง สายอากาศควรติดตั้งเข้ากับตัวรถโดยให้มีความสูงอยู่เหนือกว่าพืชพรรณในบริเวณพื้นที่ที่ทำการทดสอบด้วย

5.1.6 ส่วนของการทดลองระบบ Auto Steering

เมื่อทำการต่อมอเตอร์ในการควบคุมพวงมาลัย แล้วควบคุมผ่านอุปกรณ์ Raspberry Pi โดยรับคำสั่งควบคุมการหมุนจากผู้ใช้งาน มอเตอร์สามารถหมุนไปในทิศทางที่ผู้ใช้งานกำหนดอย่างถูกต้อง

เมื่อนำทั้งระบบมาเชื่อมต่อเพื่อทำงานร่วมกัน ผู้ใช้งานสามารถตั้งค่าตัวแปรสำคัญในการทดสอบผ่านทางหน้าจอกราฟิกประสานงานผู้ใช้ได้อย่างสะดวก ตัวอุปกรณ์ถูกออกแบบให้ง่ายต่อการติดตั้งบนตัวรถ และมีความแม่นยำในการติดตามตำแหน่งของรถทางการเกษตรในระดับเซนติเมตร

5.2 ข้อเสนอแนะ

1) การสร้างจุดปักหมุดสำหรับการคำนวณตำแหน่งด้วยเทคนิค RTK ควรใช้ลูกดิ่งในการปักหมุด และร่างเส้นเชื่อมระหว่างจุดปักหมุดโดยใช้เครื่องตีเส้น

2) ตำแหน่งของต้นเสาที่ใช้ติดตั้งสายอากาศควรติดตั้งให้ง่ายต่อการเล็งจุดปักหมุดบนพื้นที่ทดสอบด้วย

3) ในการคำนวณระยะห่างระหว่างจุดด้วยวิธีลัดโดยใช้พิกัดแบบ Geodetic มาคำนวณ ควรใช้พิกัดที่มีความละเอียดระดับทศนิยมสูงมาก และไม่ควรนำพิกัดแบบ Geodetic มาหาระยะห่างกันโดยใช้วิธีพีทาโกรัสโดยตรง

4) ในการออกแบบหน้าต่างส่วนประสานงานกราฟฟิกผู้ใช้ในส่วนการแสดงแผนที่ หากต้องการแผนที่ที่ขยายได้ถึงระดับเซนติเมตร ควรใช้แผนที่ความละเอียดสูงหรือทำการสำรวจและสร้างแผนที่ขึ้นเอง เนื่องจากแผนที่ที่เปิดให้ใช้งานแบบ Open Source มีความละเอียดไม่เพียงพอ

บรรณานุกรม

บทที่ 1

[1] Robert Odolinski, Peter J. G. Teunissen. *"Low- cost, high- precision, single-frequency GPS–BDS RTK positioning."* GPS Solut. 2017 : 1315 - 1330

[2] Akira Mizushima, Kazunobu Ishii, Noboru Noguchi, Yousuke Matsuo, Renfu Lu. *"Development of a low-cost attitude sensor for agricultural vehicles."* Computers and Electronics in Agriculture. 2011 : 198 - 204

บทที่ 2

[1] Bhatta B. *"Global Navigation Satellite Systems."* Insights into GPS, GLONASS, Galileo, Compass, and Others, BS Publications, Hyderabad. 2010

[2] "ส่วนอวกาศ"
https://www.e-education.psu.edu/natureofgeoinfo/sites/www.e-education.psu.edu.natureofgeoinfo/files/image/gps_const.jpg

[3] "ส่วนสถานีควบคุม"
https://www.e-education.psu.edu/natureofgeoinfo/sites/www.e-education.psu.edu.natureofgeoinfo/files/image/gps_control_segment.jpg

[4] "ส่วนผู้ใช้"
https://www.e-education.psu.edu/natureofgeoinfo/sites/www.e-education.psu.edu.natureofgeoinfo/files/image/gps_ranging_2.jpg

[4] "มุมมองของพิกัดที่เป็นไปได้จากการใช้ดาวเทียมหนึ่งดวง".
https://www.e-education.psu.edu/natureofgeoinfo/sites/www.e-education.psu.edu.natureofgeoinfo/files/image/gps_ranging_1.jpg

[5] "มุมมองของพิกัดที่เป็นไปได้จากการใช้ดาวเทียมสองดวง"
https://www.e-education.psu.edu/natureofgeoinfo/sites/www.e-education.psu.edu.natureofgeoinfo/files/image/gps_ranging_2.jpg

[6] "มุมมองของพิกัดที่เป็นไปได้จากการใช้ดาวเทียมสามดวง"
https://www.e-education.psu.edu/natureofgeoinfo/sites/www.e-education.psu.edu.natureofgeoinfo/files/image/gps_ranging_3.jpg

[7] "เทคนิค Trilateration"
https://en.wikipedia.org/wiki/Trilateration#/media/File:3spheres.svg

[8] "ข้อมูล Almanac"
https://en.wikipedia.org/wiki/Trilateration
[9] "ข้อมูล Ephemeris"
https://www.linkedin.com/pulse/you-know-how-gps-functions-its-method-locate-ashish-chawla
[10] "ระบบพิกัดแบบ ECEF"
http://docs.novatel.com/OEM7/Content/Resources/Images/WGS84_ECEF_Coordinate_744x474.png
[11] "ระบบพิกัดแบบ Geodetic"
https://landsat.usgs.gov/sites/default/files/images/Handbook61.jpg
[12] "ระบบพิกัดแบบ ENU"
https://commons.wikimedia.org/wiki/File:ECEF_ENU_Longitude_Latitude_relationships.svg
[13] Pornchai Supnithi. "*Coordinate Systems, Satellite Orbits.*" Introduce to Aeronautical Communication and Navigation system. Bangkok : King Mongkut's Institute of Technology Ladkrabang, 2017
[14] "เครื่องรับสัญญาณจีเอ็นเอสเอสแบบความถี่เดียวยี่ห้อ U-blox NEO M8T"
http://www.csgshop.com/img/p/205-470-thickbox.jpg
[15] "สายอากาศแบบ Choke-ring"
http://ascelibrary.org/cms/attachment/83639/1778706/figure6.gif
[16] "องค์ประกอบของเทคนิค RTK"
https://www.e-education.psu.edu/natureofgeoinfo/sites/www.e-education.psu.edu.natureofgeoinfo/files/image/gps_ranging_3.jpg
[17] "Euler Angles"
http://library.rtna.ac.th/web/RTNA_Journal/y.5c.1/4_2.pdf
[18] "Rotation Matrix"
http://library.rtna.ac.th/web/RTNA_Journal/y.5c.1/4.pdf
[19] "ADXL345"
https://cdn.sparkfun.com/r/140-140/assets/parts/2/4/0/8/09045-03-L.jpg
[20] "แนวการวางตัวของแกน X แกน Y และแกน Z ของ ADXL345"
https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf
[21] "ความเร่งที่วัดได้จากการวาง ADXL345 ให้แกน X แกน Y หรือแกน Z ตั้งฉากกับพื้นโลก"
https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf

[22] "ITG-3205"
http://www.datasheetspdf.com/datasheet/download.php?id=731954
[23] "แนวการวางตัวแกน X แกน Y และแกน Z ของ ITG 3205"
http://www.datasheetspdf.com/datasheet/download.php?id=731954
[24] "HMC5883L"
https://cdn.sparkfun.com//assets/parts/5/0/2/3/10494-01b.jpg
[25] "แนวการวางตัวแกน X แกน Y และแกน Z ของ HMC5883L"
http://robokits.download/datasheets/HMC5883L.pdf
[26] "เซนเซอร์ รุ่น GY-85"
https://medium.com/jungletronics/jaythree-balancing-car-project-part-3-5-3b3117047ac6
[27] "ภาษาไพธอน"
https://th.wikipedia.org/wiki/%E0%B8%A0%E0%B8%B2%E0%B8%A9%E0%B8%B2%E0%B9%84%E0%B8%9E%E0%B8%97%E0%B8%AD%E0%B8%99
[28] "โปรแกรม QT"
http://www.thaieasyelec.com/article-wiki/embedded-electronics-application/what-is-qt.html
[29] "โครงสร้างออปแอมป์"
http: / / www. learningaboutelectronics. com/ Articles/ Op- amp- offset- null-terminals
[30] "สัญลักษณ์ออป์แอมป์"
http://karnn2012.wixsite.com/device2/op-amp
[31] "ตัวอย่างวงจรเปรียบเทียบแรงดัน"
http://academic.udru.ac.th/~banyat/EL00202/05chapter5.pdf
[32] "โครงสร้างของรีเลย์"
http://commandronestore.com/products/bq001.php
[33] "ส่วนประกอบของบอร์ด Raspberry Pi 3 (Model B)"
https://shop.pimoroni.com/products/raspberry-pi-3
[34] "พอร์ต GPIO ของ Raspberry Pi 3 Model B"
https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout
[35] "สาย RCA"
https://th.aliexpress.com/popular/rca-yellow-cable.html, n.d.

[36] "จอแสดงผล ขนาด 7 นิ้ว"
https://www.lazada.co.th/products/cheer-7-inch-hdmi-lcd-screen-module-for raspberry display ultra clear for raspberry-pie-i11516502-s14358916.html
[37] "หลักการมอเตอร์"
http://tisade.blogspot.com/
[38] "ระบบควบคุมพวงมาลัย"
http://www.pcat.ac.th/_files_school/00000831/data/00000831_1_20141103-221329.pdf

บทที่ 3
[1] "การเชื่อมต่อ Arduino Nano V3 กับ เซนเซอร์ รุ่น GY-85 เพื่อส่งข้อมูลไปยัง Raspberry Pi"
https://shop.pimoroni.com/products/raspberry-pi-3

ภาคผนวก ก

โปรแกรมที่ใช้ในการอ่านค่ามุมแสดงการเคลื่อนที่ของวัตถุโดยใช้ เซนเซอร์ รุ่น GY-85

```
#include <Wire.h> //I2C Arduino Library
#include <math.h>
const int hmc5883Address =0x1E; //0011110b, address of HMC5883
const byte hmc5883ModeRegister =0x02;
const byte hmcContinuousMode =0x00;
const byte hmcDataOutputXMSBAddress =0x03;
const byte hmcDataOutputYMSBAddress =0x07;
const byte hmcDataOutputZMSBAddress =0x05;
//Gyro and Accel Variable
int Gyro_output[3], Accel_output[3];
float dt =0.02;
float Gyro_cal_x, Gyro_cal_y, Gyro_cal_z, Accel_cal_x, Accel_cal_y,
Accel_cal_z;
float Gyro_pitch =0; //Initialize value
float Accel_pitch =0; //Initial Estimate
float Predicted_pitch =0; //Initial Estimate
float Gyro_roll =0; //Initialize value
float Accel_roll =0; //Initialize value
float Predicted_roll =0; //Output of Kalman filter
float Q =0.1; //Prediction Estimate Initial Guess
float R =5; //Prediction Estimate Initial Guess
float P00 =0.1; //Prediction Estimate Initial Guess
float P11 =0.1; //Prediction Estimate Initial Guess
float P01 =0.1; //Prediction Estimate Initial Guess
float Kk0, Kk1;
float Comp_pitch;
float Comp_roll;
unsigned long timer;
unsigned long time;
//Function to write in Gyro and Accel
void writeTo(byte device, byte toAddress, byte val){
Wire.beginTransmission(device);
Wire.write(toAddress);
Wire.write(val);
Wire.endTransmission();}
//Read Accel and Gyro
void readFrom(byte device, byte fromAddress, int num, byte result[]){
Wire.beginTransmission(device);
Wire.write(fromAddress);
Wire.endTransmission();
Wire.requestFrom((int)device, num);
int i =0;
while(Wire.available()){
result[i]=Wire.read();
i++;
}
}
void getGyroscopeReadings(int Gyro_output[]){
byte buffer[6];
readFrom(0x68,0x1D,6,buffer);
Gyro_output[0]=((int)buffer[0]<< 8 )| buffer[1];
Gyro_output[1]=((int)buffer[2]<< 8 )| buffer[3];
Gyro_output[2]=((int)buffer[4]<< 8 )| buffer[5];
```

```
}
void getAccelerometerReadings(int Accel_output[]){
byte buffer[6];
readFrom(0x53,0x32,6,buffer);
Accel_output[0]=((int)buffer[1]<< 8 )| buffer[0];
Accel_output[1]=((int)buffer[3]<< 8 )| buffer[2];
Accel_output[2]=((int)buffer[5]<< 8 )| buffer[4];
}
//
void setup(){
int Gyro_cal_x_sample =0;
int Gyro_cal_y_sample =0;
int Gyro_cal_z_sample =0;
int Accel_cal_x_sample =0;
int Accel_cal_y_sample =0;
int Accel_cal_z_sample =0;
int i;
delay(5);
Serial.begin(9600);
Wire.begin();
Wire.beginTransmission(hmc5883Address); //open communication with HMC5883
Wire.write(hmc5883ModeRegister); //select mode register
Wire.write(hmcContinuousMode); //continuous measurement mode
Wire.endTransmission();
writeTo(0x53,0x31,0x09); //Set accelerometer to 11bit, +/-4g
writeTo(0x53,0x2D,0x08); //Set accelerometer to measure mode
writeTo(0x68,0x16,0x1A); //Set gyro +/-2000deg/sec and 100Hz Low Pass Filter
writeTo(0x68,0x15,0x09); //Set gyro 100Hz sample rate
delay(100);
for (i =0; i < 100; i +=1){
getGyroscopeReadings(Gyro_output);
getAccelerometerReadings(Accel_output);
Gyro_cal_x_sample +=Gyro_output[0];
Gyro_cal_y_sample +=Gyro_output[1];
Gyro_cal_z_sample +=Gyro_output[2];
Accel_cal_x_sample +=Accel_output[0];
Accel_cal_y_sample +=Accel_output[1];
Accel_cal_z_sample +=Accel_output[2];
delay(50);
}
Gyro_cal_x =Gyro_cal_x_sample /100;
Gyro_cal_y =Gyro_cal_y_sample /100;
Gyro_cal_z =Gyro_cal_z_sample /100;
Accel_cal_x =Accel_cal_x_sample /100;
Accel_cal_y =Accel_cal_y_sample /100;
Accel_cal_z =(Accel_cal_z_sample /100)-256;
}
void loop(){
//compass
int x,y,z;
Wire.beginTransmission(hmc5883Address);
Wire.write(hmcDataOutputXMSBAddress); //select register 3, X MSB register
```

```
Wire.endTransmission();
Wire.requestFrom(hmc5883Address, 6);
if(6<=Wire.available()){
x =Wire.read()<<8; //X msb
x |=Wire.read(); //X lsb
z =Wire.read()<<8; //Z msb
z |=Wire.read(); //Z lsb
y =Wire.read()<<8; //Y msb
y |=Wire.read(); //Y lsb
}
 x =1.014380531*x-139.98;
 y =y+93.5;
int azimuth =atan2(y , x)/M_PI *180;
if(azimuth<0)
 { azimuth =azimuth+360;}
//roll pitch
timer =millis();
getGyroscopeReadings(Gyro_output);
getAccelerometerReadings(Accel_output);
Accel_pitch =atan2((Accel_output[0]-Accel_cal_x)/256, (Accel_output[2]-
Accel_cal_z)/256)*180 /PI;
Gyro_pitch =Gyro_pitch +((Gyro_output[1]-Gyro_cal_y)/14.375)*dt;
Predicted_pitch =Predicted_pitch +((Gyro_output[1]-Gyro_cal_y)/14.375)*dt;
Accel_roll =atan2((Accel_output[1]-Accel_cal_y)/256, (Accel_output[2]-
Accel_cal_z)/256)*180 /PI;
Gyro_roll =Gyro_roll -((Gyro_output[0]-Gyro_cal_x)/14.375)*dt;
Predicted_roll =Predicted_roll -((Gyro_output[0]-Gyro_cal_x)/14.375)*dt;
P00 +=dt *(2 *P01 +dt *P11);
P01 +=dt *P11;
P00 +=dt*Q;
P11 +=dt *Q;
Kk0 =P00 /(P00 +R);
Kk1 =P01 /(P01 +R);
Predicted_pitch +=(Accel_pitch -Predicted_pitch)*Kk0;
Predicted_roll +=(Accel_roll -Predicted_roll)*Kk0;
P00 *=(1 -Kk0);
P01 *=(1 -Kk1);
P11 -=Kk1 *P01;
time=millis();
Serial.print(Predicted_roll);
/*Serial.print("\t");
Serial.print(Predicted_pitch);
Serial.print("\t");
Serial.print("\t");
Serial.print(azimuth);*/
Serial.print("\n");
timer =millis()-timer;
timer =(dt *1000)-timer;
delay(timer);
}
```

ภาคผนวก ข

ชุดคำสั่งที่ใช้ในการเชื่อมต่อระหว่าง Arduino Nano V3 กับ Raspberry Pi ด้วยภาษาไพธอน

โปรแกรม serial_test.py เป็นชุดคำสั่งที่ใช้ในการเชื่อมต่อระหว่าง Arduino Nano V3 กับ Raspberry Pi ด้วยภาษาไพธอน มีรายละเอียดดังนี้

```
import serial,sys,time
try:
   ser = serial.Serial('/dev/ttyUSB1', 9600)
except serial.serialutil.SerialException:
   ser = serial.Serial('/dev/ttyUSB0', 9600)
def savesolution(data):
    f = open('tilt.txt','a+')
    f.write(data + "\n")
    return
tilt = []
A = []
while True:
        tilt=ser.readline()
        A = tilt.split(",")
        try:
            R = float(A[0])
            P = float(A[1])
            Y = float(A[2])
        except ValueError:
            A = []
            tilt=ser.readline()
            A = tilt.split(",")
            R = float(A[0])
            P = float(A[1])
            Y = float(A[2])
        print(R,P,Y)
        print("\n")
        t = time.clock()
        print("\n")
        data = str([R,P,Y,t])
        savesolution(data)
```

การเชื่อมต่อพอร์ต USB ของเซนเซอร์ รุ่น GY-85

นำค่ามุม Roll Pitch และ Yaw มาแสดงออกทางหน้าจอ

ภาคผนวก ค

ชุดคำสั่งการเก็บพิกัดจากโปรแกรม RTKRCV ไปคำนวณใน Rotation Matrix ด้วยภาษาไพธอน

ชุดคำสั่งที่ใช้ส่งค่าพิกัดจากโปรแกรม RTKRCV ไปยัง Raspberry Pi จะอยู่ในโปรแกรม newedit.py สามารถแสดงการเขียนได้ดังนี้

```
import getpass
import os
import warnings, socket, select, string, sys, time, serial, binascii,
serial.tools.list_ports, decimal
import numpy as np
import pickle, encodings
h = 2#input('Antenna High(m.) = ')
os.system("/home/pi/rtklib2.4.3b26/app/rtkrcv/gcc/telnetgps.sh")
time.sleep(5)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostbyname(socket.gethostname())
port = 52001
s.connect((host,port))
sd = []
myData = []
try:
   ser = serial.Serial('/dev/ttyUSB1', 9600)
except serial.serialutil.SerialException:
   ser = serial.Serial('/dev/ttyUSB0', 9600)
def savesolution(data):
    f = open('RTK.txt','a+')
    f.write(data + '\n')
    return
while True:
# Split data for ECEF Coordinate
        data = s.recv(1024)#defualt is 1024
        word = data.split()
        length_word = len(word)
        #ECEF = word[2:5]
        X = float(word[2])
        Y = float(word[3])
        Z = float(word[4])
        q = int(word[5])
        ns = int(word[6])
# Input Roll Pitch Yaw from GY-85
        if len(data) != 0:
           myData = ser.readline()
           sd = myData.split(",")
           try:
              r = float(sd[0])
              p = float(sd[1])
              y = float(sd[2])
           except ValueError:
              sd = []
              myData = ser.readline()
              sd = myData.split(",")
              r = float(sd[0])
              p = float(sd[1])
              y = float(sd[2])
           Roll = r
           Pitch = p
           Yaw = y
           h = float(h)
#Use Rotation Matrix
        a = 6378137
        b = 6356752.3142
        P = np.sqrt(np.power(X,2)+np.power(Y,2))
```

```
        e2 = 0.00669437999014
        e2p = 0.00673949674228
        theta = np.array(np.arctan((Z*a)/(P*b)))
        Lat = np.arctan((Z+e2p*b*np.power(np.sin(theta),3))/(P-
e2*a*np.power(np.cos(theta),3)))
        Lng = np.arctan2(Y,X)
        enu = np.matrix([[0],[0],[h]])
        ecef = np.array([[X],[Y],[Z]])
        r = np.pi*np.array(r)/180
        p = np.pi*np.array(p)/180
        y = np.pi*np.array(y)/180
        cr = np.cos(r)
        cp = np.cos(p)
        cy = np.cos(y)
        sr = np.sin(r)
        sp = np.sin(p)
        sy = np.sin(y)
        clat = np.cos(Lat)
        clng = np.cos(Lng)
        slat = np.sin(Lat)
        slng = np.sin(Lng)
        rot = np.array([[cy*cp,sr*sp*cy-
cr*sy,cr*sp*cy+sr*sy],[cp*sy,sr*sp*sy+cr*cy,cr*sp*sy-sr*cy],[-
sp,sr*cp,cr*cp]])
        conv2ecef = np.array(np.transpose([[-slng,clng,0],[-slat*clng,-
slat*slng,clat],[clat*clng,clat*slng,slat]]))
        enup = np.matmul(rot,enu)
        ecefp = ecef - np.matmul(conv2ecef,enup)

#Use convert efef to lla
        thetaA = np.arctan((Z*a)/(P*b))
        latA = np.arctan((Z+e2p*b*np.power(np.sin(thetaA),3))/(P-
e2*a*np.power(np.cos(thetaA),3)))
        lngA = np.arctan2(Y,X)
        LatA = (latA*180)/np.pi
        LngA = (lngA*180)/np.pi
        P2 = np.sqrt(np.power(ecefp[0],2)+np.power(ecefp[1],2))
        thetaB = np.arctan((ecefp[2]*a)/(P2*b))
        latB =
np.arctan((ecefp[2]+e2p*b*np.power(np.sin(thetaB),3))/(P2-
e2*a*np.power(np.cos(thetaB),3)))
        lngB = np.arctan2(ecefp[1],ecefp[0])
        LatB = (latB*180)/np.pi
        LngB = (lngB*180)/np.pi
        A = [LatA,LngA]
        B = [LatB,LngB]
#Find Error from tilting
        lat1 = (np.pi*A[0])/180
        lat2 = (np.pi*B[0])/180
        lng1 = (np.pi*A[1])/180
        lng2 = (np.pi*B[1])/180
        dlat = lat2-lat1#np.array([lat2-lat1])
        dlng = lng2-lng1#np.array([lng2-lng1])
        S = np.power(np.sin(dlat/2),2)+
np.cos(lat1)*np.cos(lat2)*np.power(np.sin(dlng/2),2)
        c = 2*np.arcsin(np.sqrt(S))
        r = 6371230
        dist = c*r
        Distance = dist
#Display Position before and after correction
        print("Input position = " + str(np.array([X,Y,Z])))
```

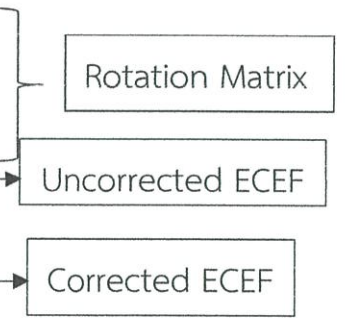
```
        print("Tilt = " + str(Roll) + "," + str(Pitch) + "," + str(Yaw))
        print("After improved = " + str(ecefp))
        print("Error(m.) " + str(Distance))
# Savefile to text
        DataOut =
(X,Y,Z,Roll,Pitch,Yaw,ecefp[0],ecefp[1],ecefp[2],Distance)
        #np.savetxt('CompensateData.txt', DataOut, fmt=('%5.3f',
'%5.3f','%5.3f', '%5.3f', '%5.3f',
'%5.3f','%5.3f','%5.3f','%5.3f','%5.3f'))
        sd = []
        savesolution(str(DataOut))
```

ภาคผนวก ง

ชุดคำสั่งการจำลองการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศในโปรแกรม MatLab

โปรแกรมสำหรับจำลองการแก้ไขพิกัดที่เกิดจากความเอียงของสายอากาศ สามารถแสดงได้ดังนี้

```
clear
clc
tic
latA = 13.725704257792572;  %input('lat0 = ');
lngA = 100.7788757714733;
A = [latA,lngA];
h = input('Antenna High from ground(m.) = ');
ENU0 = [0;0;h]%input('lng0 = ');
r = input('Roll in Degrees = ');
p = input('Pitch in Degrees = ');
yaw = input('Yaw in Degrees = ');
cr = cosd(r);
sr = sind(r);
sp = sind(p);
cp = cosd(p);
sy = sind(yaw);
cy = cosd(yaw);
m11 = cp*cy; m12 = cp*sy; m13 = -sp;
m21 = sr*sp*cy-cr*sy; m22 = sr*sp*sy-cr*cy; m23 = sr*cp;
m31 = cr*sp*cy+sr*sy; m32 = cr*sp*sy-sr*cy; m33 = cr*cp;
rot = [m11 m12 m13; m21 m22 m23; m31 m32 m33]'
ENU1 = rot*ENU0
ENUnew = -1*(ENU1);
xyz0 = lla2ecef([latA lngA h])
x0 = xyz0(1,1); y0 = xyz0(1,2);z0 = xyz0(1,3);
[x,y,z]=ENUtoECEF(xyz0,ENUnew);
xyz1 = [x,y,z]
B = ecef2lla([xyz1(1,1) xyz1(1,2) xyz1(1,3)],'WGS84');
L = haverside(latA,lngA,B(1,1),B(1,2));
refdis = h*sqrt((sind(r))^2+(sind(p))^2);
error = refdis - L;
fprintf('latA = %f, lngA = %f, hA = %f \n',latA,lngA,h);
fprintf('Roll = %d, Pitch = %d, Yaw = %d \n',r,p,yaw);
fprintf('latB = %f, lngB = %f, hB = %f \n',B(1,1),B(1,2),B(1,3));
fprintf('Distance from A to B point via arc length = %f meters
\n',L);
fprintf('Pethagorian Distance from A to B point = %f meters
\n',pethadis);
fprintf('Distance from A to B point Ref = %f meters \n',refdis);
fprintf('Error of arc length - %f meters \n',error);
quiver3(0,0,0,ENU0(1,1),ENU0(2,1),ENU0(3,1),'-');grid;hold on;
quiver3(0,0,0,ENU1(1,1),ENU1(2,1),ENU1(3,1));grid
toc
```



ภาคผนวก จ

ชุดคำสั่งคำนวณเส้นทางจากจุดอ้างอิงมุมสนาม 4 จุด ด้วยโปรแกรม Matlab

```
K =(pi/180)*6371116 ;%difined constant to mapped lat lng to x,y plane
disp('Please input your corner of playground in latitude(lat),
longitude(lng)')
Ax =input('lng Ax =');
Ay =input('lat Ay =');
Bx =input('lng Bx =');
By =input('lat By =');
Cx =input('lng Cx =');
Cy =input('lat Cy =');
Dx =input('lng Dx =');
Dy =input('lat Dy =');
A =[Ax Ay];
B =[Bx By];
C =[Cx Cy];
D =[Dx Dy];
disp('Choose Origin Point')
disp('Choose 1 >> A is Origin Point')
disp('Choose 2 >> B is Origin Point')
disp('Choose 3 >> C is Origin Point')
disp('Choose 4 >> D is Origin Point')
```

ปัจจุบันสามารถใช้งานได้เพียง Case 1 เนื่องจากอยู่ในช่วงการทดสอบ

```
Origin =input('Origin point =');
switch Origin
          case 1
                a =[0 0];
                b =K*(B-A);
                c =K*(C-A);
                d =K*(D-A);
                n =input('Number strip of playground =');
                l =d -a; L =sqrt((l(1,1))^2+(l(1,2))^2);
                P1 =a; P2 =b; P3 =c;
                P21 =P2-P1; P31 =P3-P1;
                LP21 =sqrt((P21(1,1))^2+(P21(1,2))^2);
                LP31 =sqrt((P31(1,1))^2+(P31(1,2))^2);
                D =LP21;
                cd =c-d; CD =sqrt((cd(1,1))^2+(cd(1,2))^2);
                ex =P21/LP21; I =dot(ex,P31);
                eyupper =P31-I*(ex);
                  eylower =sqrt((eyupper(1,1))^2+(eyupper(1,2))^2);
                ey =eyupper/eylower;
                J =dot(ey,P31);
          for m =1:n-1
               r1(m,1)=m*(L/n);
               r2(m,1)=sqrt(((r1(m,1)).^2)+(D^2));
               r3(m,1)=sqrt(((L-r1(m,1))^2)+(CD)^2);
               x(m,1)=(((r1(m,1))^2)-((r2(m,1))^2)+(D^2))/(2*D);
               y(m,1)=(((r1(m,1))^2)-((r3(m,1))^2)+((x(m,1)-I).^2)+(J^2)-((x(m,1)).^2))/(2*J);
               X(m,1)=(x(m,1))/K;
               Y(m,1)=(y(m,1))/K;
      %find opposite of marker
          end
```

```
        M =[x Y];
        for i =1:n-1
                M_1(:,i)=A *[M(i,1)M(i,2)]; M_2(:,i)=B*[M(i,1)M(i,2)];
        end
        figure(1)
        geoshow([Ay Dy By Cy], [Ax Dx Bx Cx], 'DisplayType', 'Point',
'Marker', '+', 'Color', 'red');
        grid on;hold on;
        M_lon =[M_1(1,:)M_2(1,:)];
        M_lat =[M_1(2,:)M_2(2,:)];
        Pos_output =[M_lat;M_lon];
        figure(1)
        geoshow(M_lat(1,:),M_lon(1,:),'DisplayType', 'Point', 'Marker', 'o')
        title('Position');
        xlabel('Longtitude');ylabel('Latitude');
end
```

ภาคผนวก ฉ

ชุดคำสั่งแสดงเส้นทางร่วมกับแผนที่ด้วยภาษาไพธอน

```
import pygmaps
import math

mymap =pygmaps.maps(13.730221, 100.772189, 19)

# create a list of locations
path =[(13.730649, 100.771925), #NW
  (13.729780, 100.771933), #SW
  (13.729789, 100.772513), #SE
  (13.730658, 100.772505), #NE
  (13.730649, 100.771925)]

pos1 =[(13.7298705658597,100.771933000000),
  (13.7298795658597, 100.772513000000)]
pos2 =[(13.7299574712723,100.771933000000),
  (13.7299664712723,100.772513000000)]
pos3 =[(13.7300443766848,100.771933000000),
  (13.7300533766848,100.772513000000)]
pos4 =[(13.7301312820974,100.771933000000),
  (13.7301402820974,100.772513000000)]
pos5 =[(13.7302181875100,100.771933000000),
  (13.7302271875100,100.772513000000)]
pos6 =[(13.7303050929226,100.771933000000),
  (13.7303140929226,100.772513000000)]
pos7 =[(13.7303919983351,100.771933000000),
  (13.7304009983351,100.772513000000)]
pos8 =[(13.7304789037477,100.771933000000),
  (13.7304879037477,100.772513000000)]
pos9 =[(13.7305658091603,100.771933000000),
  (13.7305748091603,100.772513000000)]

for point in path:
 mymap.addpoint(point[0], point[1])
mymap.addpath(path, "#ff0000")
mymap.addpath(pos1, "#0026ff")
mymap.addpath(pos2, "#0026ff")
mymap.addpath(pos3, "#0026ff")
mymap.addpath(pos4, "#0026ff")
mymap.addpath(pos5, "#0026ff")
mymap.addpath(pos6, "#0026ff")
mymap.addpath(pos7, "#0026ff")
mymap.addpath(pos8, "#0026ff")
mymap.addpath(pos9, "#0026ff")
mymap.draw('./line.html')
```

ชุดพิกัดอ้างอิง
(มุมสนามทั้ง 4 มุม)

ชุดพิกัดที่ได้
จากการคำนวณ

ชุดคำสั่ง
การสร้างเส้นทาง

ภาคผนวก ช

ชุดคำสั่งควบคุมการทำงานของหลอด LED แสดงสถานะการทำงานของ Raspberry Pi

```
int logicdetect =12;
int ledboot =13;
boolean running =false;
void setup(){
pinMode(ledboot, OUTPUT);
pinMode(logicdetect, INPUT);
}
void loop(){
if(digitalRead(logicdetect)==LOW)
{
     delay(300);
running =!running;
digitalWrite(ledboot, running);
}
else
{
digitalWrite(ledboot, HIGH);
}
}
```

เงื่อนไขการทำงานของหลอด LED

ภาคผนวก ซ

ชุดคำสั่งควบคุมการทำงานของมอเตอร์เพื่อควบคุมพวงมาลัย

โปรแกรมที่ใช้ควบคุมการทำงานของมอเตอร์เพื่อการควบคุมพวงมาลัยด้วยภาษาไพธอน สามารถแสดงได้ดังนี้

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(5,GPIO.OUT) # motor input A
GPIO.setup(7,GPIO.OUT) # motor input B
GPIO.setup(16,GPIO.OUT) #En1
GPIO.setup(18,GPIO.OUT) #En2
while True:
    print('Direction: \n 0 to 90:counter-clockwise \n -90 to
0:Clockwise')
    x = int(input("Please enter number of direction : "))

    if x==0:                #1.1s = 360degree
        print('offmotor')
        GPIO.output(5,GPIO.LOW)
        GPIO.output(7,GPIO.LOW)
        GPIO.output(16,GPIO.HIGH)
        GPIO.output(18, GPIO.HIGH)

    elif x==10:
        print('counter-clockwise')
        GPIO.output(5,GPIO.HIGH)
        GPIO.output(7,GPIO.LOW)
        GPIO.output(16,GPIO.HIGH)
        GPIO.output(18,GPIO.HIGH)
        print(time.ctime())
        time.sleep(0.076)
        GPIO.output(5, GPIO.LOW)
        GPIO.output(7, GPIO.HIGH)
        time.sleep(0.076)
        GPIO.output(5,0)
        GPIO.output(7,0)
    elif x==20:
        print('counter-clockwise')
        GPIO.output(5,GPIO.HIGH)
        GPIO.output(7,GPIO.LOW)
        GPIO.output(16,GPIO.HIGH)
        GPIO.output(18,GPIO.HIGH)
        print(time.ctime())
        time.sleep(0.152)
        GPIO.output(5, GPIO.LOW)
        GPIO.output(7, GPIO.HIGH)
        time.sleep(0.152)
        GPIO.output(5,0)
        GPIO.output(7,0)
    elif x==30:
        print('counter-clockwise')
        GPIO.output(5,GPIO.HIGH)
        GPIO.output(7,GPIO.LOW)
        GPIO.output(16,GPIO.HIGH)
        GPIO.output(18,GPIO.HIGH)
        print(time.ctime())
        time.sleep(0.228)
        GPIO.output(5, GPIO.LOW)
        GPIO.output(7, GPIO.HIGH)
        time.sleep(0.228)
        GPIO.output(5,0)
```

```
    GPIO.output(7,0)
elif x==40:
    print('counter-clockwise')
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.304)
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    time.sleep(0.304)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==50:
    print('counter-clockwise')
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.38)
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    time.sleep(0.38)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==60:
    print('counter-clockwise')
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.456)
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    time.sleep(0.456)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==70:
    print('counter-clockwise')
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.532)
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    time.sleep(0.532)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==80:
    print('counter-clockwise')
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.608)
```

```
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    time.sleep(0.608)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==90:
    print('counter-clockwise')
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.684)
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    time.sleep(0.684)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==360:
    print('counter-clockwise')
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(1.1)
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    time.sleep(1.1)
    GPIO.output(5,0)
    GPIO.output(7,0)

elif x==-10:
    print('Clockwise')
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.076)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.076)
    GPIO.output(5,0)
    GPIO.output(7,0)

elif x==-20:
    print('Clockwise')
    GPIO.output(5,GPIO.LOW)
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.152)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.152)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==-30:
    print('Clockwise')
    GPIO.output(5, GPIO.LOW)
```

```
    GPIO.output(7, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.228)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.228)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==-40:
    print('Clockwise')
    GPIO.output(5,GPIO.LOW)
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.304)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.304)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==-50:
    print('Clockwise')
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.38)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.38)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==-60:
    print('Clockwise')
    GPIO.output(5,GPIO.LOW)
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.456)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.456)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==-70:
    print('Clockwise')
    GPIO.output(5, GPIO.LOW)
    GPIO.output(7, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.532)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.532)
    GPIO.output(5,0)
    GPIO.output(7,0)
```

```
elif x==-80:
    print('Clockwise')
    GPIO.output(5,GPIO.LOW)
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.608)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.608)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==-90:
    print('Clockwise')
    GPIO.output(5,GPIO.LOW)
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(0.684)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(0.684)
    GPIO.output(5,0)
    GPIO.output(7,0)
elif x==-360:
    print('Clockwise')
    GPIO.output(5,GPIO.LOW)
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(1.1)
    GPIO.output(5,GPIO.HIGH)
    GPIO.output(7,GPIO.LOW)
    time.sleep(1.1)
    GPIO.output(5,0)
    GPIO.output(7,0)
else :
    print('off motor')
    GPIO.output(5,0)
    GPIO.output(7,0)
    GPIO.output(16,GPIO.HIGH)
    GPIO.output(18,GPIO.HIGH)
    print(time.ctime())
    time.sleep(1)  # One second delay
```