ระบบสถานีภาคพื้นดินของแบบจำลองดาวเทียมกระป๋อง และดาวเทียมขนาดเล็ก

Ground Station System for CanSat and CubeSat

โดย

นายธิติ เทียนประทีป

นายธีรพันธ์ ทองอ่วมใหญ่

นางสาวไปรดา รักษ์บางแหลม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ระบบสถานีภาคพื้นดินของแบบจำลองดาวเทียมกระป๋อง และดาวเทียมขนาดเล็ก

Ground Station System for CanSat and CubeSat

โดย

นายธิติ	เทียนประทีป	58010586
นายธีรพันธ์	ทองอ่วมใหญ่	58010599
นางสาวไปรดา	รักษ์บางแหลม	58010805

อาจารย์ที่ปรึกษา

ดร. พีระเมศร์ โชติกวีกิจญาดา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ผ่านการตรวจรูปเล่มแล้ว

(.....)

อาจารย์ที่ปรึกษา

22/5/62

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

ผ่านการตรวจชิ้นงานแล้ว

(.....)

กรรมการผู้ตรวจชิ้นงาน

22/5/62

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบสถานีภาคพื้นดินของแบบจำลองดาวเทียมกระป๋องและดาวเทียมขนาดเล็ก
GROUND STATION SYSTEM FOR CANSAT AND CUBESAT

ผู้จัดทำ

1. นายธิติ เทียนประทีป 58010586
2. นายธีรพันธ์ ทองอ่วมใหญ่ 58010599
3. นางสาวไปรดา รักษ์บางแหลม 58010805

..... อาจารย์ที่ปรึกษา

(ดร. พีระเมศร์ โชติกวีกิจญาดา)

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถสำเร็จลุล่วงไปด้วยดีได้เลย หากปราศจากความอนุเคราะห์จากอาจารย์ที่ปรึกษา ดร.พีรเมศร์ โชติกวีกิจญาดา ที่ให้คำแนะนำ สั่งสอน และให้ความรู้ต่าง ๆ อีกทั้งยังคอยดูแล ซื้อขนมและน้ำมาให้ตอนที่ทำปริญญานิพนธ์ฉบับนี้ ทำให้มีกำลังใจ และมีแรงผลักดันตลอดเวลาที่ทำปริญญานิพนธ์ฉบับนี้ ขอขอบพระคุณเป็นอย่างยิ่ง

ขอขอบคุณ พี่ ๆ ในห้องโปรเจคที่เป็นแรงบันดาลใจการจัดทำปริญญานิพนธ์ฉบับนี้ ทั้งคอยเสนอแนะ และให้คำปรึกษา แม้จะจบการศึกษาไปแล้ว แต่ยังให้ความช่วยเหลือกับพวกเราตลอดมา ขอบคุณพี่และน้องจากภาควิชาวิศวกรรมเครื่องกล ที่ให้คำปรึกษาในการออกแบบระบบมอเตอร์เคลื่อนที่ 2 แกน ซึ่งเป็นอุปกรณ์หลักของปริญญานิพนธ์ในครั้งนี้ ขอขอบคุณเพื่อน ๆ และอาจารย์ประจำภาควิชาวิศวกรรมโทรคมนาคมทุกท่านที่ได้ให้ความช่วยเหลือ ให้ความรู้ และอบรมสั่งสอน

สุดท้ายนี้ ขอกราบขอบพระคุณบิดามารดาของผู้จัดทำ ที่ให้การสนับสนุนตลอดมาและคอยเป็นกำลังใจให้ทั้งในการเรียนและการทำปริญญานิพนธ์ฉบับนี้จนประสบความสำเร็จ

ธิติ เทียนประทีป
ธีรพันธ์ ทองอ่วมใหญ่
ไปรดา รักษ์บางแหลม
ผู้จัดทำ

ระบบสถานีภาคพื้นดินของแบบจำลองดาวเทียมกระป๋อง
และดาวเทียมขนาดเล็ก
GROUND STATION FOR CANSAT AND CUBESAT

โดย นายธิติ เทียนประทีป 58010586
นายธีรพันธ์ ทองอ่วมใหญ่ 58010599
นางสาวไปรดา รักษ์บางแหลม 58010805

อาจารย์ที่ปรึกษา ดร. พีระเมศร์ โชติกวีกิจญาดา

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ จัดทำขึ้นเพื่อพัฒนาระบบสถานีภาคพื้นดินสำหรับแบบจำลองดาวเทียมกระป๋อง (CanSat) และแบบจำลองดาวเทียมขนาดเล็ก (CubeSat) ซึ่งสามารถแบ่งเป็น 2 ส่วนหลักๆ คือ ส่วนซอฟท์แวร์ (Software) และส่วนฮาร์ดแวร์ (Hardware)

โดยส่วนซอฟท์แวร์ คือ การเขียนโปรแกรมโดยแบ่งเป็น 3 ส่วน ได้แก่ 1. โปรแกรมในการควบคุมระบบติดตามสายอากาศ โดยจะใช้ Arduino ในการเขียนโปรแกรม 2. โปรแกรมสำหรับแสดงผลข้อมูลที่รับได้จากดาวเทียม โดยแสดงข้อมูลเป็นตำแหน่งของแบบจำลองดาวเทียมขนาดเล็ก , ภาพการเคลื่อนที่ของวัตถุ และบันทึกข้อมูลต่าง ๆ ที่รับมาจาก CanSat และ CubeSat 3. โปรแกรมสำหรับดึงข้อมูลการเคลื่อนที่ของ CubeSat เพื่อให้ระบบติดตามสายอากาศทำงานได้แบบอัตโนมัติ โดยจะดึงข้อมูลมาจาก Database ในเว็บที่มีอยู่ โดยวิธีในการดึงข้อมูลนั้น จะใช้โปรแกรม Python เพื่อดึงข้อมูลแบบ Real-Time

และส่วนฮาร์ดแวร์ สามารถแบ่งได้เป็น 2 ส่วน 1. ระบบมอเตอร์เคลื่อนที่ 2 แกน สำหรับติดตามดาวเทียม ที่ช่วยปรับทิศทางของสายอากาศให้สามารถชี้ไปยังเป้าหมายโดยอัตโนมัติ และ 2. ระบบเซ็นเซอร์ สำหรับวัดมุมก้ม มุมเงยและ ทิศทางของสายอากาศ

ABSTRACT

The purpose of this project is to develop ground station system for CanSat and CubeSat, the system consists of Software and Hardware.

The software parts are the programs which comprise (1) Program controlling the antenna for tracking system using Arduino (2) Program displaying the data that received from CanSat and CubeSat and (3) Program retrieving CubeSat motion data from website for automatic tracking antenna system by Python to make Real-Time system.

The hardware parts comprise (1) 2-Axis motor subsystem for antenna tracking which adjusts the antenna to point to the satellite automatically and (2) Sensor subsystem to measures azimuth angle, elevation angle and direction of antenna.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากในปัจจุบัน เรื่องของดาวเทียมเป็นเรื่องที่น่าสนใจในผู้คนหมู่มาก อีกทั้งยังมีการพัฒนาระบบการสื่อสารผ่านดาวเทียมอยู่ตลอด เพื่อให้มีประสิทธิภาพสูงสุดในการใช้งานในอวกาศ จึงควรค่าแก่การนำมาศึกษาและทำความเข้าใจการทำงานของระบบการสื่อสารผ่านดาวเทียม ผู้จัดทำต้องการที่จะเรียนรู้หลักการทำงานของเทคโนโลยีทางด้านอวกาศอย่างดาวเทียมนี้ โดยเริ่มจากการศึกษาอุปกรณ์แบบจำลองการทำงานของดาวเทียม ซึ่งมีชื่อว่า แบบจำลองดาวเทียมกระป๋อง (CanSat) และสร้างแบบจำลองดาวเทียมกระป๋องขึ้นมาด้วยตนเอง เพื่อลดต้นทุนในการสร้างดาวเทียม และศึกษาการเคลื่อนที่ของแบบจำลองดาวเทียมขนาดเล็ก (CubeSat) ที่มีการใช้งานจริงอยู่บนชั้นบรรยากาศ และทำการดึงข้อมูลที่ได้รับจากแบบจำลองดาวเทียมขนาดเล็ก มาศึกษาที่สถานีภาคพื้นดิน ซึ่งจะออกแบบให้มีขนาดเล็ก และใช้ระบบสถานีภาคพื้นดินอย่างคุ้มค่าที่สุด

แบบจำลองดาวเทียมกระป๋อง เป็นแบบจำลองดาวเทียมขนาดเล็กเท่ากระป๋องเครื่องดื่ม โดยจะติดตั้งอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ ไว้ภายในกระป๋องที่เป็นรูปทรงกระบอก โดยแบบจำลองดาวเทียมกระป๋องนี้จะสามารถรับส่งข้อมูลได้ผ่านอุปกรณ์ไร้สาย สู่สถานีภาคพื้นดิน และด้วยขนาดที่เล็กกว่าดาวเทียมปกติอยู่มาก ทำให้มีราคาที่ถูกกว่า และเหมาะแก่การนำมาศึกษาและทดลองเพื่อตรวจวัดค่าของสภาพอากาศ ทั้งอุณหภูมิ ความเร็วของลม ความดันอากาศ และตรวจหาตำแหน่งของแบบจำลองดาวเทียมกระป๋อง แล้วแสดงผลข้อมูลต่าง ๆ ที่หน้าจอคอมพิวเตอร์ที่สถานีภาคพื้นดิน

แบบจำลองดาวเทียมขนาดเล็ก เป็นแบบจำลองดาวเทียมที่มีรูปร่างเป็นทรงลูกบาศก์ ทางผู้จัดทำนำมาประยุกต์ใช้โดยการทำระบบติดตามแบบจำลองดาวเทียมขนาดเล็ก ซึ่งจะดึงข้อมูลของดาวเทียมแบบเรียลไทม์ตามการเคลื่อนที่จริงในขณะนั้น โดยในส่วนนี้จะใช้ระบบเดียวกันกับการตรวจหาตำแหน่งของแบบจำลองดาวเทียมกระป๋อง เพื่อการใช้งานระบบติดตามแบบจำลองดาวเทียมอย่างมีประสิทธิภาพและคุ้มค่ามากที่สุด

1.2 วัตถุประสงค์

1) ออกแบบและพัฒนาระบบสถานีภาคพื้นดิน (Ground Station System) สำหรับแบบจำลองดาวเทียมกระป๋อง (CanSat) และดาวเทียมขนาดเล็ก (CubeSat)

2) ออกแบบและสร้างระบบควบคุมการเคลื่อนที่ของสายอากาศสำหรับติดตามแบบจำลองดาวเทียมกระป๋อง และดาวเทียมขนาดเล็ก และแสดงข้อมูลผ่านหน้าจอแสดงผล

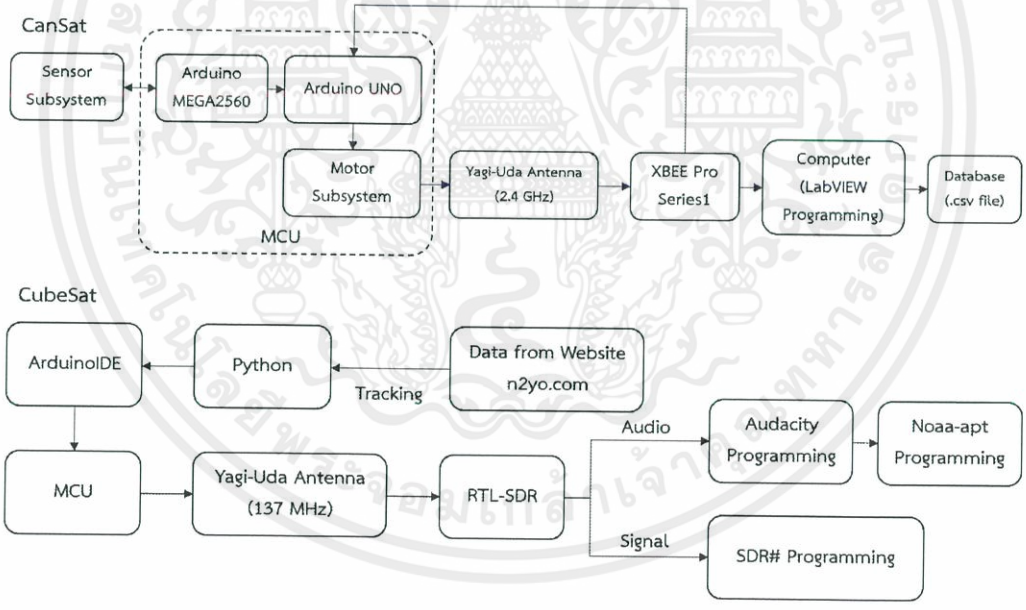
3) ออกแบบและเขียนโปรแกรมเพื่อดึงข้อมูลจากเว็บไซต์สำหรับติดตามแบบจำลองดาวเทียมขนาดเล็ก

1.3 ขอบเขตของปริญญานิพนธ์

1) สามารถออกแบบและพัฒนาระบบสถานีภาคพื้นดินสำหรับแบบจำลองดาวเทียมกระป๋องและดาวเทียมขนาดเล็กได้

2) ระบบติดตามสายอากาศสามารถใช้งานได้จริง สามารถรับน้ำหนักของสายอากาศได้ และแสดงข้อมูลผ่านหน้าจอแสดงผลได้

3) สามารถดึงข้อมูลจากเว็บไซต์สำหรับติดตามแบบจำลองดาวเทียมขนาดเล็กได้



รูปที่ 1.1 บล็อกไดอะแกรมการทำงานในส่วนสถานีภาคพื้นดิน

จากรูปที่ 1.1 จะเป็นบล็อกไดอะแกรมการทำงานในส่วนสถานีภาคพื้นดิน ซึ่งจะแบ่งออกเป็น 2 ส่วน คือส่วนของแบบจำลองดาวเทียมกระป๋อง (CanSat) และส่วนของแบบจำลองดาวเทียมขนาดเล็ก (CubeSat) ในส่วนของ CanSat จะรับข้อมูลที่ส่งมาจากภาคส่งมายังภาครับผ่าน

โมดูล XBEE Pro Series 1 ซึ่งเป็นอุปกรณ์ที่ใช้ในการสื่อสารระหว่าง CanSat กับภาครับ โดยข้อมูลจะประกอบไปด้วยความดัน อุณหภูมิ ความสูง ค่าละติจูด และลองติจูด เป็นต้น จากนั้นนำมาประมวลผลผ่าน Microcontroller Unit ซึ่งประกอบไปด้วย Arduino MEGA2560 ที่รับค่ามาจากระบบเซ็นเซอร์ตรวจจับตำแหน่งและระบบเซ็นเซอร์วัดมุมก้ม-มุมเงย และ Arduino UNO ที่เป็นตัวควบคุมระบบมอเตอร์เคลื่อนที่ 2 แกนที่ใช้ในการติดตาม CanSat ที่ใช้สายอากาศยากิ-อูดะที่มีความถี่ 2.4 GHz ตรงกับ XBEE Pro Series 1 จากนั้นจึงนำข้อมูลต่าง ๆ มาแสดงผลในหน้าจอคอมพิวเตอร์ผ่านหน้าจอแสดงผลของโปรแกรม LabVIEW และเก็บข้อมูลลงใน Microsoft Excel

ในส่วนของ CubeSat จะทำการดึงข้อมูลการเคลื่อนที่ของ CubeSat แบบ Real-Time มาจากเว็บไซต์ N2YO.com โดยใช้โปรแกรม Python จากนั้นทำการส่งข้อมูลไปยังโปรแกรม ArduinoIDE และเขียนโปรแกรมเพื่อควบคุมการติดตาม CubeSat ที่ใช้สายอากาศ Dual Band Handheld ที่มีความถี่ 144/430 MHz ผ่าน Microcontroller Unit จากนั้นจึงนำข้อมูลต่าง ๆ เข้าคอมพิวเตอร์โดยใช้ RTL-SDR และนำมาแสดงผลผ่านหน้าจอแสดงผลของโปรแกรม SDR# เพื่อสังเกตรูปสเปกตรัมของสัญญาณที่สามารถรับได้

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

CanSat [1] คือ ดาวเทียมขนาดเล็กเท่าขนาดกระป๋องเครื่องดื่ม ที่จำลองการทำงานเหมือนดาวเทียมจริง โดยติดตั้งอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ ไว้ภายในกระป๋องเครื่องดื่ม หรือวัสดุรูปทรงขนาดเท่ากระป๋องเครื่องดื่ม

การปล่อย CanSat จะใช้จรวดยิงขึ้นไปบนท้องฟ้า หรือใช้ Drone และบอลลูนปล่อยลงมาจากที่สูงในระดับหลายร้อยเมตร ตัว CanSat เมื่อถูกปล่อยออกมาแล้ว ก็จะกางร่มโดยอัตโนมัติ และระบบต่าง ๆ จะเริ่มทำงาน เช่น บันทึกภาพถ่ายหรือวิดีโอ ระบุตำแหน่งพิกัดของตัว CanSat วัดอุณหภูมิและความดันอากาศ สามารถรับส่งข้อมูลได้ทันทีผ่านผ่านอุปกรณ์ไร้สายส่งสู่อุปกรณ์รับสัญญาณภาคพื้นดิน

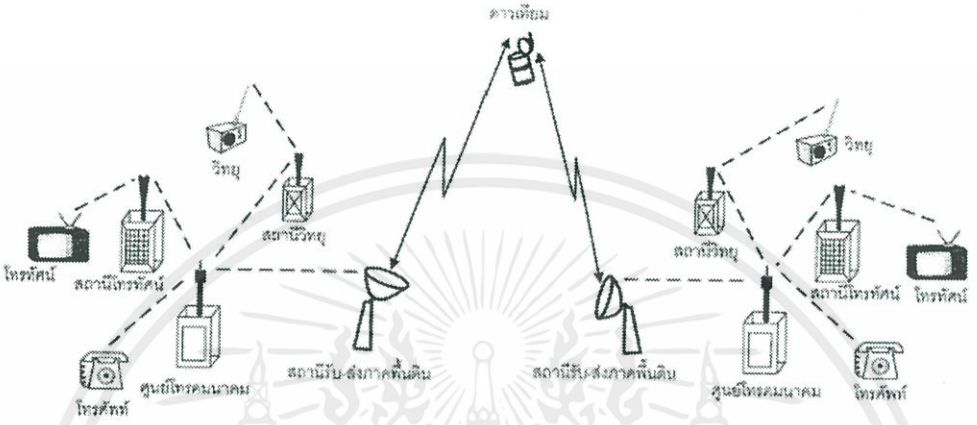
และภายใน CanSat จะมีเซ็นเซอร์ต่าง ๆ เช่นเดียวกับที่ใช้ในโทรศัพท์มือถือ Smart Phone ในปัจจุบัน เช่น ระบบ GPS, กล้องถ่ายรูป, เซ็นเซอร์วัดความเร่ง, เซ็นเซอร์วัดสนามแม่เหล็ก ซึ่งมีขนาดเล็ก และราคาถูก สามารถหาซื้อได้ตามท้องตลาดทั่วไป และสามารถหาข้อมูลการใช้งานได้ง่ายจากอินเตอร์เน็ต เพื่อนำมาพัฒนาด้วยตนเอง [2]

ระบบการสื่อสารผ่านดาวเทียมเป็นการสื่อสารที่นิยมมากในปัจจุบัน เนื่องจากเป็นการสื่อสารแบบไมโครเวฟที่สะดวกสบายต่อผู้ใช้งาน ทำให้การสื่อสารมีการพัฒนาอย่างต่อเนื่อง จนมีการใช้งานอย่างกว้างขวาง นับตั้งแต่ดาวเทียมสื่อสารดวงแรกที่ถูกส่งไปในปี พ.ศ. 2508 โดยองค์การโทรคมนาคมระหว่างประเทศ (International Telecommunications Satellite Organization) หรืออินเทลแสต (INTELSAT)

2.1 ระบบการสื่อสารผ่านดาวเทียม

ดาวเทียมสื่อสาร [3] เป็นดาวเทียมที่ใช้ในการติดต่อสื่อสารทั้งในประเทศและระหว่างประเทศ ตลอดจนการคมนาคมขนส่ง ซึ่งช่วยในการควบคุมเส้นทางและบอกตำแหน่งที่อยู่ โดยดาวเทียมจะทำหน้าที่เป็นสถานีรับส่งคลื่นวิทยุสื่อสารติดต่อกับสถานีภาคพื้นดิน ช่วยให้กิจการสื่อสารทางโทรศัพท์ โทรพิมพ์ โทรสาร และการถ่ายทอดสัญญาณโทรทัศน์ระหว่างประเทศเป็นไปอย่างทั่วถึงและรวดเร็ว สำหรับประเทศไทยจะใช้บริการของดาวเทียมอินเทลแสตและดาวเทียมปาลาปาของประเทศอินโดนีเซีย

ดาวเทียมเพื่อการสื่อสารนั้นจะทำหน้าที่เป็นสถานีทวนสัญญาณซึ่งในดาวเทียมจะติดตั้งอุปกรณ์รับส่งคลื่นวิทยุเพื่อใช้รับและถ่ายทอดสัญญาณสู่พื้นโลกโดยพลังงานไฟฟ้าที่ใช้ในตัวดาวเทียมนั้นได้มาจากเซลล์แสงอาทิตย์



รูปที่ 2.1 องค์ประกอบของระบบการสื่อสารผ่านดาวเทียม [3]

2.1.1 องค์ประกอบของระบบการสื่อสารผ่านดาวเทียม

ระบบการสื่อสารผ่านดาวเทียม [4] ประกอบไปด้วยสองส่วนหลัก คือ สถานีภาคพื้นดิน (Ground Segment) และสถานีอวกาศ (Space Segment) โดยที่สถานีภาคพื้นดินประกอบด้วยสองสถานีคือ สถานีรับและสถานีส่งภาคพื้นดินศูนย์โทรคมนาคม ซึ่งการทำงานของทั้งสองสถานีนี้มีลักษณะคล้ายกัน แสดงดังรูปที่ 2.1

2.1.1.1 สถานีภาคพื้นดิน (Ground Segment) [5]

สถานีภาคพื้นดิน มีอุปกรณ์หลักอยู่ 4 ชนิด ได้แก่

1) อุปกรณ์จานสายอากาศ (Antenna Subsystem)

มีหน้าที่ส่งสัญญาณและรับสัญญาณจากดาวเทียม ซึ่งต้องมีคุณลักษณะพื้นฐาน ได้แก่ ต้องมีอัตราขยายของทิศทางที่สูง (high Directive Gain) หรือเรียกอีกอย่างว่า มีบีมของลำคลื่นแคบ เพื่อการส่งและรับคลื่นสัญญาณ และต้องมีโลปด้านข้าง (Side Lobes) ที่ต่ำ เพื่อป้องกันการรบกวนจากสัญญาณที่สะท้อนจากพื้นดินเข้าจานรับ, ต้องมีค่าอุณหภูมิสัญญาณรบกวน (Noise Temperature) ต่ำ เพื่อให้ได้ค่า S/N ที่สูงขึ้น, ต้องมีความเที่ยงตรงสูง

(Validity) สามารถรับส่งสัญญาณได้ดี กรณีเป็นจานสายอากาศขนาดใหญ่ จะต้องมีระบบติดตาม (Tracking System) ของจานสายอากาศเพื่อควบคุมการเคลื่อนจานสายอากาศไปหาตำแหน่งที่รับสัญญาณดาวเทียมได้สูงสุด

2) อุปกรณ์เครื่องรับ-ส่งสัญญาณวิทยุ (Radio Frequency Subsystem)

มีหน้าที่รับส่งสัญญาณวิทยุที่ใช้งาน ระบบการรับและส่งของสถานีดาวเทียมภาคพื้นดินปกติจะทำงานอยู่ในย่านความถี่วิทยุต่างกัน เช่น C-Band และ Ku-Band

3) อุปกรณ์แปลงสัญญาณวิทยุ (RF/IF Subsystem)

ประกอบด้วยสถานีส่งสัญญาณและสถานีรับสัญญาณ โดยด้านสถานีส่งถูกเรียกว่า ภาคแปลงสัญญาณขาขึ้น (Up Converter Part) ซึ่งทำหน้าที่แปลงย่านความถี่ที่ได้รับมาให้เป็นความถี่ที่ใช้กับงานระบบดาวเทียม แล้วส่งสัญญาณที่แปลงความถี่ให้ภาคขยายสัญญาณ เพื่อขยายให้เป็นสัญญาณความถี่สูง จากนั้นนำส่งไปยังดาวเทียม และเช่นเดียวกันสำหรับด้านสถานีรับนั้นเรียกว่า ภาคแปลงสัญญาณขาลง (Down Converter Part) ทำหน้าที่คือแปลงสัญญาณที่ได้รับจากดาวเทียมไปเป็นความถี่ที่ใช้งาน จากนั้นส่งต่อให้ภาคแยกสัญญาณ (Demodulator) ต่อไป

4) อุปกรณ์ผสมสัญญาณและแยกสัญญาณ (Modulator/Demodulator)

เป็นชุดอุปกรณ์ที่มีหน้าที่แปลงข้อมูลที่ต้องการส่งผ่านดาวเทียมให้เป็นสัญญาณคลื่นวิทยุที่มีข้อมูลผสมอยู่ให้นำไปใช้งานได้ แปลงข้อมูลข่าวสารที่อยู่ในย่านเบสแบนด์ให้อยู่ในรูปสัญญาณ RF หรือสัญญาณคลื่นพาห์โดยการมอดูเลตสัญญาณทางด้านส่ง จากนั้นทำการแยกข้อมูลข่าวสารออกจากคลื่นพาห์ โดยการดีมอดูเลตสัญญาณทางด้านรับ เพื่อนำข้อมูลข่าวสารไปใช้งานต่อไป ซึ่งข้อมูลข่าวสารแบ่งออกเป็น 2 ลักษณะ คือ แบบแอนะลอก และแบบดิจิตอล ปัจจุบันนิยมส่งข้อมูลข่าวสารในรูปแบบสัญญาณดิจิตอลซึ่งให้ประสิทธิภาพในการรับและส่งสูงกว่าแบบสัญญาณแอนะลอก

2.1.1.2 สถานีอวกาศ (Space Segment)

สถานีอวกาศนั้น ประกอบด้วยอุปกรณ์ดังนี้

1) อุปกรณ์ขับเคลื่อนดาวเทียม (Propulsion Subsystem)

ทำหน้าที่ทำให้ดาวเทียมหมุนและรักษาตำแหน่งไว้ด้วยก๊าซหรือพลังงานความร้อนจากไฟฟ้า

2) อุปกรณ์ควบคุมดาวเทียม (Spacecraft control Subsystem)

มีหน้าที่รักษาสมดุลของดาวเทียมเพื่อไม่ให้ดาวเทียมหลุดวงโคจรออกไปในอวกาศได้

3) อุปกรณ์สื่อสาร (Electronic Communication Subsystem)

มีหน้าที่รับสัญญาณจากสถานีส่งแล้วส่งต่อไปยังสถานีรับโดยมีช่องสัญญาณรับความถี่ขาขึ้น (Transponder) จากนั้นแปลงสัญญาณเป็นสัญญาณความถี่ขาลง (Downlink Frequency) แล้วจึงส่งมายังสถานีรับภาคพื้นดินต่อไป

4) อุปกรณ์พลังงานไฟฟ้า (Electrical Power Subsystem)

มีหน้าที่แปลงพลังงานแสงอาทิตย์ให้เป็นพลังงานไฟฟ้าสำหรับอุปกรณ์สื่อสารและภาคควบคุมต่าง ๆ บนดาวเทียมนอกจากนี้ยังเก็บพลังงานไฟฟ้าไว้ในตัวเก็บประจุหรือแบตเตอรี่ (Battery) เพื่อสำรองไว้ใช้งาน

5) อุปกรณ์สายอากาศ (Antenna Subsystem)

ทำหน้าที่รับสัญญาณจากภาคพื้นดิน

6) อุปกรณ์ติดตามและควบคุม (Telemetry Tracking and Command Subsystem)

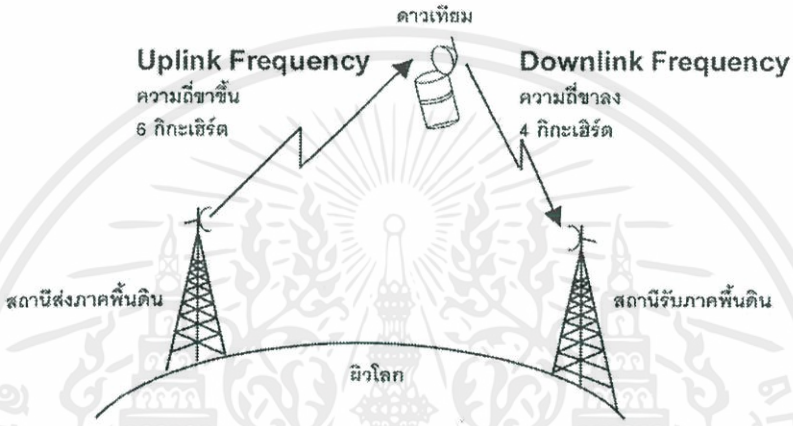
มีหน้าที่ติดตามการทำงานของดาวเทียมและควบคุมรักษาตำแหน่งของดาวเทียมให้ถูกต้องเสมอโดยอุปกรณ์การสื่อสารโทรคมนาคม

2.2 ระบบสถานีภาคพื้นดิน (Ground Station System)

ระบบสถานีดาวเทียมภาคพื้นดิน เป็นส่วนประกอบสำคัญในการสื่อสารผ่านดาวเทียมที่ทำหน้าที่ติดต่อกับดาวเทียมและเชื่อมต่อกับผู้ใช้งาน เช่น ระบบสื่อสารข้อมูลเครือข่ายคอมพิวเตอร์ ระบบโทรศัพท์ หรือระบบถ่ายทอดสัญญาณโทรทัศน์ เป็นต้น

สถานีดาวเทียมภาคพื้นดินในปัจจุบันได้รับการพัฒนาให้ใช้เทคโนโลยีในระบบดิจิตอล ทำให้มีความสามารถสูงขึ้น เช่น สามารถส่งข้อมูลได้ด้วยความเร็วสูง ในขณะที่ขนาดของอุปกรณ์ที่ใช้มีขนาดเล็กลง และสามารถทำงานในสภาพแวดล้อมทั่วไปได้

2.2.1 วิธีการทำงานของระบบสถานีภาคพื้นดิน



รูปที่ 2.2 การรับ-ส่งสัญญาณจากดาวเทียม [5]

การทำงานของระบบสถานีภาคพื้นดิน แสดงดังรูปที่ 2.2 สามารถอ้างอิงได้จากองค์ประกอบของระบบการสื่อสารผ่านดาวเทียม ในส่วนของสถานีภาคพื้นดิน

2.2.1.1 อุปกรณ์จานสายอากาศ (Antenna Subsystem)

ในส่วนนี้จะใช้สายอากาศยากิ-อูดะ เพื่อรับสัญญาณจากแบบจำลองดาวเทียมขนาดเล็ก ที่มีระยะห่างจากสถานีภาคพื้นดินในช่วง 1-1.5 กิโลเมตร

2.2.1.2 อุปกรณ์สัญญาณวิทยุ (Radio Frequency Subsystem)

ในส่วนนี้จะใช้อุปกรณ์รับสัญญาณวิทยุที่ความถี่ 2.4 GHz ซึ่งคือ XBEE Pro S1 และจะกล่าวถึงการทำงานของอุปกรณ์ชนิดนี้ในบทต่อไป

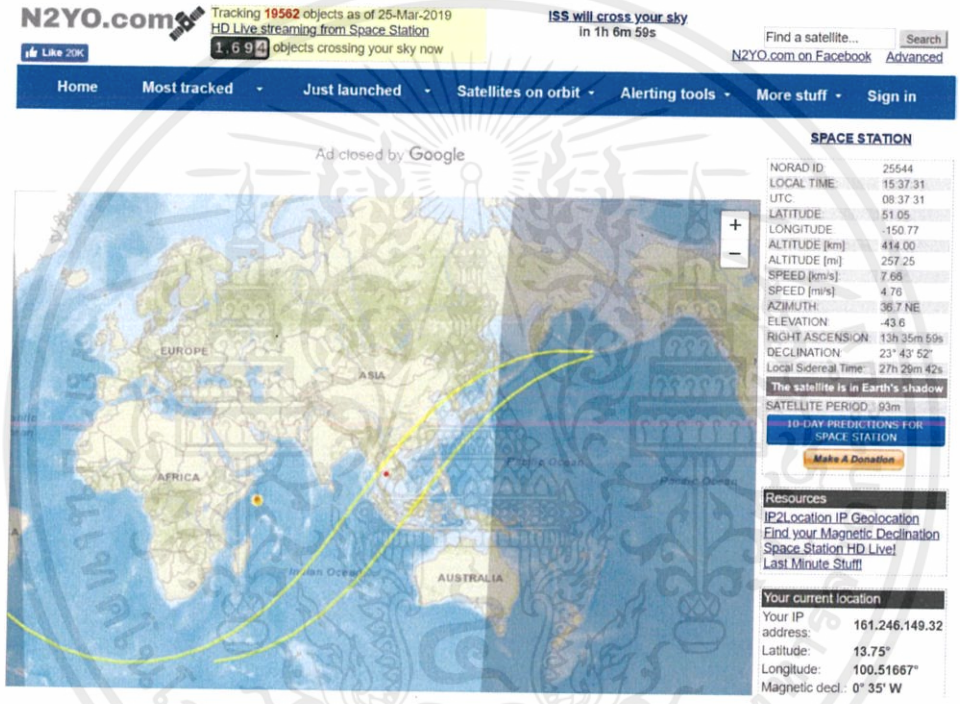
2.2.1.3 อุปกรณ์แปลงสัญญาณวิทยุ (RF/IF Subsystem)

ในส่วนนี้จะใช้อุปกรณ์ตัวเดียวกันกับการรับสัญญาณวิทยุ เนื่องจากอุปกรณ์ที่ใช้มีความสามารถในการรับข้อมูลจากดาวเทียม และประมวลผลเข้าคอมพิวเตอร์ได้เลยโดยไม่ต้องผ่านอุปกรณ์แปลงสัญญาณก่อน

2.2.1.4 อุปกรณ์ผสมสัญญาณและแยกสัญญาณ (Modulator/Demodulator)

ในส่วนนี้จะทำกระบวนการทั้งหมดในคอมพิวเตอร์ผ่านโปรแกรม XCTU และ LabVIEW แล้วจึงนำข้อมูลมาประมวลได้เลยเพื่อทำการแสดงผลในหน้าจอของโปรแกรม

2.3 แบบจำลองดาวเทียมขนาดเล็ก (CubeSat)



รูปที่ 2.3 หน้าจอของเว็บไซต์ N2YO.com

แบบจำลองดาวเทียมขนาดเล็ก [6] หรือ CubeSat เป็นดาวเทียมขนาดเล็กมาก ที่มีรูปร่างหน้าตาเป็นทรงลูกบาศก์ ที่มีความกว้างด้านละ 10 เซนติเมตร ซึ่งแบบจำลองดาวเทียมขนาดเล็กนี้ ได้ถูกผลักดันให้เป็นมาตรฐานในการสร้างดาวเทียมขนาดเล็กอื่น ๆ เพื่อทำหน้าที่แทนดาวเทียมขนาดใหญ่ในการสำรวจโลกระยะสั้น ได้แก่ สำรวจสภาพอากาศ สำรวจพื้นผิวโลก สำรวจทรัพยากรธรรมชาติ เป็นต้น โดยความสามารถของแบบจำลองดาวเทียมขนาดเล็กนี้ จะขึ้นอยู่กับอุปกรณ์หรือเซนเซอร์ต่าง ๆ ที่ติดตั้งเข้าไป

โดยในปริญญานิพนธ์ฉบับนี้ จะมีการศึกษาการทำงานของแบบจำลองดาวเทียมขนาดเล็ก จากระบบติดตามแบบจำลองดาวเทียมขนาดเล็ก เพื่อดึงเอาข้อมูลที่ได้รับมาประมวลผลผ่านคอมพิวเตอร์ และแสดงผลค่าที่ได้รับมาผ่านโปรแกรมทางคอมพิวเตอร์ โดยจะดึงข้อมูลการเคลื่อนที่ของ CubeSat ผ่านเว็บไซต์ N2YO.com ซึ่งจะมีหน้าจอของเว็บไซต์ดังรูปที่ 2.3

2.4 รายละเอียดการแข่งขัน CanSat Competition 2019

2.4.1 ภาพรวมของภารกิจ (Mission Overview)

ภารกิจของการแข่งขันในปี 2019 จะสำรวจการควบคุมการใช้งานกลไกใบพัดแบบอัตโนมัติ (auto-gyro) ของส่วนภายในของ CanSat เมื่อถูกปล่อยออกมาจากตัวโครงสร้างหลัก โดย CanSat จะประกอบไปด้วย 2 ส่วน ได้แก่ ส่วนภายใน (Science Payload) และส่วนภายนอก (Container) ซึ่งมีไว้เพื่อปกป้องส่วนภายใน ขณะที่ใช้งานอยู่กับจรวด

Cansat จะถูกปล่อยที่ระดับความสูงตั้งแต่ 670 เมตรถึง 725 เมตรเหนือจุดปล่อย ซึ่งใกล้กับยอดเขาสูง ส่วนภายนอกของ CanSat จะต้องปกป้องส่วนภายในของ CanSat จากความเสียหายต่าง ๆ ในระหว่างการปล่อยและการใช้งาน CanSat

เมื่อตัวจรวดปลด CanSat ออก CanSat จะตกลงมาด้วยการใช้ร่มชูชีพที่ความเร็ว 20 เมตรต่อวินาที และเมื่อเคลื่อนที่ลงถึงระดับความสูง 450 เมตร ส่วนภายนอกจะแยกออกจากส่วนภายใน ที่มีกลไกใบพัดอัตโนมัติ จากนั้น ความเร็วของส่วนภายในจะลดลงอยู่ที่ 10 ถึง 15 เมตรต่อวินาที เมื่อส่วนภายนอกตกอยู่ภายใต้การควบคุมโดยกลไกใบพัดอัตโนมัติ ส่วนภายนอกจะส่งข้อมูล Telemetry ซึ่งประกอบไปด้วย เซ็นเซอร์เพื่อติดตามระดับความสูง โดยใช้ความกดอากาศ, อุณหภูมิภายนอก, แรงดันไฟฟ้าของแบตเตอรี่, ตำแหน่ง GPS, มุมเอียงและมุมแหงน, และอัตราเร็วของกลไกใบพัดอัตโนมัติ เมื่อส่วนภายนอกลงสู่พื้นดิน การส่งผ่านข้อมูล Telemetry ทั้งหมดจะหยุดลง และสัญญาณ Beacon ที่ตั้งอยู่จะเปิดใช้งาน

2.4.2 ข้อบังคับของการแข่งขันในส่วนของสถานีภาคพื้นดิน (Ground Station)

ข้อบังคับของการแข่งขัน CanSat Competition 2019 ในส่วนของสถานีภาคพื้นดิน สามารถแสดงได้ดังตารางที่ 2.1

ตารางที่ 2.1 ข้อบังคับพื้นฐานในส่วนของสถานีภาคพื้นดิน

ข้อที่	ข้อกำหนด
28	สถานีภาคพื้นดินจะต้องมีคำสั่งที่ใช้ควบคุมให้ส่วนภายนอกปรับเทียบความสูงของบรรยากาศ และมุมเอียงและมุมแหงนได้ เมื่อเทียบกับจุดปล่อยจรวด
29	สถานีภาคพื้นดินจะต้องสร้างไฟล์สกุล .csv ซึ่งมีข้อมูลเซ็นเซอร์ทั้งหมด ตามที่ระบุไว้ในส่วนข้อมูล Telemetry
31	XBEE ที่ใช้สำหรับการส่ง Telemetry ต้องมีความถี่ที่ 2.4 GHz หรือ XBEE Pro ที่ความถี่ 900 MHz
32	XBEE ต้องมีการตั้งค่า NETID/PANID ตามหมายเลขของทีม
33	XBEE ห้ามใช้โหมดบรอดแคสต์
35	แต่ละทีมต้องพัฒนาสถานีภาคพื้นดินของตนเอง
36	Telemetry ทั้งหมดจะต้องแสดงแบบเรียลไทม์ขณะที่ร่อนลงสู่พื้นดิน
37	Telemetry ทั้งหมดจะต้องแสดงผลในหน่วยทางวิศวกรรม เช่น เมตร, เมตรต่อวินาที, องศาเซลเซียส เป็นต้น
39	สถานีภาคพื้นดินจะต้องมีคอมพิวเตอร์แล็ปท็อปที่สามารถใช้งานแบตเตอรี่ได้อย่างน้อย 2 ชั่วโมง, XBEE และสายอากาศ
40	สถานีภาคพื้นดินต้องเป็นแบบพกพา เพื่อให้ทีมสามารถวางตำแหน่งไว้ที่สถานีปฏิบัติงานภาคพื้นดิน โดยจะไม่มีไฟ AC อยู่ที่บริเวณที่ทำการปฏิบัติงานของสถานีภาคพื้นดิน

2.5 ทฤษฎีพื้นฐานของ Zigbee

2.5.1 คุณสมบัติของ Zigbee [7]

เป็นมาตรฐานสากลของอุปกรณ์ไร้สาย กำหนดโดย Zigbee Alliance เป็นการสื่อสารแบบไร้สายที่มีอัตราการรับ-ส่งข้อมูลต่ำ ใช้ควบคุมอุปกรณ์ต่าง ๆ แบบไร้สายซึ่งมีความเร็วไม่สูงมาก เช่น สวิตช์เปิด-ปิดไฟ ระบบควบคุมอุณหภูมิห้อง การรับค่าจากเซนเซอร์ต่าง ๆ โดย Zigbee ถูกออกแบบให้มีคุณลักษณะดังนี้

1) เป็นเครือข่ายไร้สายในระยะใกล้

2) ราคาถูก เสียค่าติดตั้งและดูแลน้อย

3) ติดตั้งง่าย สามารถประยุกต์ใช้งานได้หลากหลาย

4) สามารถรับส่งข้อมูลได้โดยเชื่อมั่นในความถูกต้องได้

5) ประหยัดพลังงาน มีพลังงานในการทำงานต่ำ

2.5.2 ลักษณะการทำงานของ Zigbee

ZigBee ได้แบ่งตามลักษณะการทำงาน 3 แบบ คือ

2.5.2.1 Coordinator

มีหน้าที่สร้างการสื่อสาร เชื่อมโยงเครือข่าย ระหว่าง End Device กับ Router หรือ Coordinator กับ Coordinator ด้วยกัน หรือ Coordinator กับ Router กำหนด address ให้กับ device ที่อยู่ในวงเครือข่าย ไม่ให้ซ้ำกัน ดูแลจัดการเรื่องการRouting เส้นทาง ซึ่งเทียบได้กับ FFD

2.5.2.2 End Device

เป็นอุปกรณ์ปลายทางสุด ซึ่งจะใช้รับสัญญาณจาก Sensor ที่ปลายทาง โดยที่ใช้พลังงานต่ำในการทำงาน เทียบได้กับ RFD หรือ FFD บางกรณี ขึ้นอยู่กับ sensor ที่ใช้

2.5.2.3 Router

มีหน้าที่ รับส่งข้อมูล ในเส้นทางต่าง ๆ ของเครือข่าย ซึ่งเทียบได้กับ FFD

2.6 ทฤษฎีพื้นฐานของ RTL-SDR Dongle

2.6.1 คุณสมบัติของ RTL-SDR Dongle [8]

หมายถึง อุปกรณ์ฮาร์ดแวร์พื้นฐานที่สร้างขึ้นเพื่อป้องกันผู้ใช้ที่ไม่ได้รับสิทธิ์ ไม่ให้สามารถคัดลอกซอฟต์แวร์นั้น ๆ ออกไปได้ โดยเฉพาะแอพพลิเคชั่นไฮเอนด์

คำว่า Dongle ส่วนใหญ่จึงถูกใช้กับฮาร์ดแวร์คีย์ แผ่นดิสก์ที่เก็บรหัสพิเศษและหมายเลขลงทะเบียน Dongle ที่พบเห็นก่อนหน้านี้จะเชื่อมต่อกับพอร์ตยูเอสบีลักษณะจะคล้าย

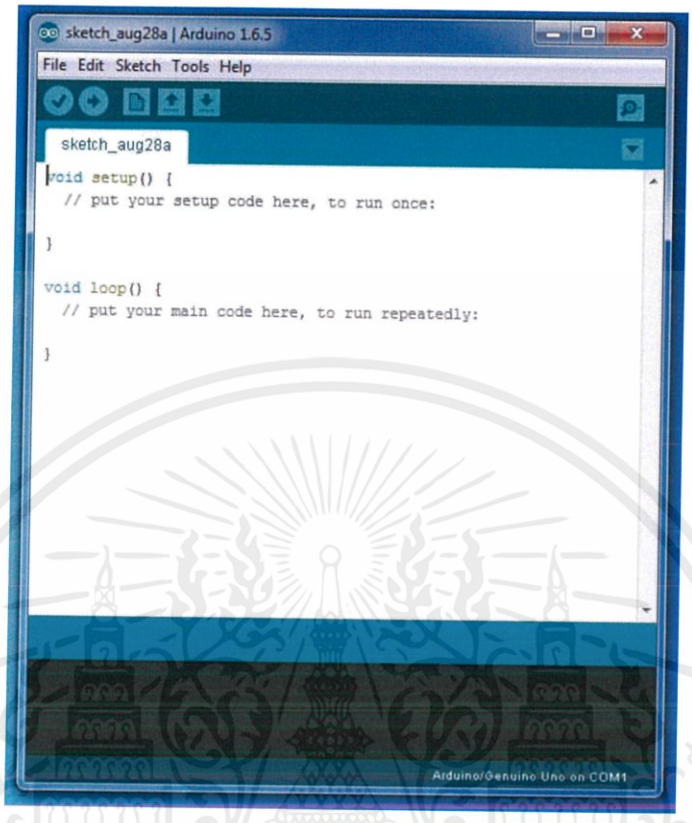
Thumb drive มาก ๆ ภายในตัวมัน จะเก็บโค้ดในการเข้าถึงการทำงานฟังก์ชันต่าง ๆ ของซอฟต์แวร์ โดยลักษณะของ Dongle สามารถแสดงได้ดังรูปที่ 2.4

รูปที่ 2.4 RTL-SDR Dongle ใน Generations ต่าง ๆ [8]

Dongle มักจะได้รับการเชื่อมต่อเข้ากับพอร์ต หรือช่องเชื่อมต่อโดยเฉพาะ เพื่อยืนยันตัวผู้ใช้ก่อนที่พวกเขาจะสามารถเข้าถึงซอฟต์แวร์โปรแกรมต่าง ๆ ได้ ซึ่ง Dongle จะถูกตรวจสอบเพื่อยืนยันตลอดเวลา ไม่เช่นนั้น ผู้ใช้จะไม่สามารถใช้ซอฟต์แวร์นั้นได้ตลอดการทำงาน

2.7 โปรแกรม ArduinoIDE

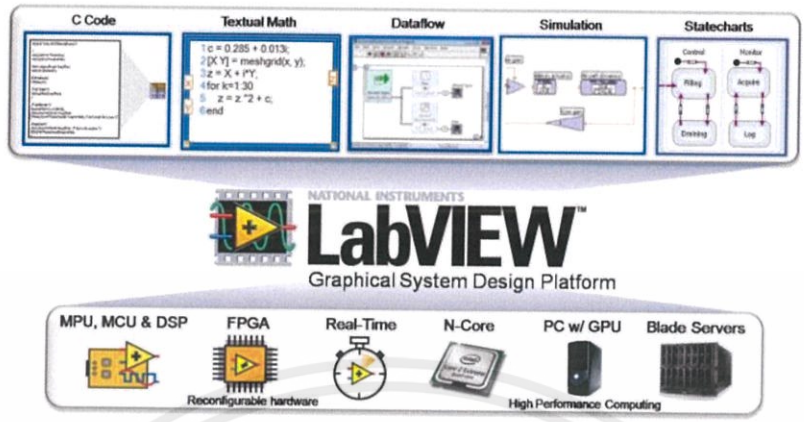
ArduinoIDE [9] เป็นซอฟต์แวร์ที่ใช้ในการพัฒนางานสำหรับบอร์ด Arduino โดยเขียนโปรแกรมและคอมไพล์ลงบอร์ด ซึ่ง IDE ย่อมาจาก Integrated Development Environment หมายถึงส่วนเสริมของระบบการพัฒนา หรือตัวช่วยต่าง ๆ ที่จะคอยช่วยเหลือ Developer หรือช่วยเหลือคนที่พัฒนา Application เพื่อเสริมให้เกิดความรวดเร็ว ถูกต้อง แม่นยำ ตรวจสอบระบบที่จัดทำได้ ทำให้การพัฒนางานต่าง ๆ เร็วมากขึ้น โดยหน้าจอของโปรแกรม ArduinoIDE จะมีลักษณะดังรูปที่ 2.5



รูปที่ 2.5 หน้าจอของโปรแกรม ArduinoIDE [10]

2.8 โปรแกรม LabVIEW

LabVIEW [11] เป็นซอฟต์แวร์ระบบวิศวกรรมที่ถูกใช้ในการทดลองการวัดและการควบคุม โดยสามารถเข้าถึงข้อมูลเชิงลึกเกี่ยวกับฮาร์ดแวร์และข้อมูลได้อย่างรวดเร็ว ซึ่งการออกแบบโปรแกรมโดยใช้โปรแกรม LabVIEW จะช่วยลดความยุ่งยากในการผสานรวมฮาร์ดแวร์สำหรับงานด้านวิศวกรรม เพื่อให้มีวิธีการที่สอดคล้องกันในการรับข้อมูลจาก NI และฮาร์ดแวร์ของบริษัทอื่น ๆ และ LabVIEW ยังช่วยลดความซับซ้อนของการเขียนโปรแกรม เพื่อให้สามารถมุ่งเน้นไปที่ปัญหาด้านวิศวกรรมเฉพาะ โดย LabVIEW มีความสามารถมากมายโดยแสดงได้ดังรูปที่ 2.6 และยังช่วยให้สามารถเห็นผลทันทีด้วยการสร้างหน้าจออินเทอร์เฟซของโปรแกรมได้อย่างสะดวก



รูปที่ 2.6 ความสามารถของโปรแกรม LabVIEW [12]

2.8.1 การทำงานของโปรแกรม LabVIEW [13]

ภาษาโปรแกรมที่ใช้ในโปรแกรม LabVIEW คือภาษาในการเขียนโปรแกรม Dataflow โดยโปรแกรมจะทำงานตามลำดับที่กำหนดโดยตัวผู้ใช้งาน ที่ออกแบบโครงสร้างของแผนภาพแบบกราฟฟิก ผู้ใช้งานจะเชื่อมต่อโหนดฟังก์ชันต่าง ๆ โดยการโยงเส้นเชื่อมกัน โปรแกรมทั้งหมดจะเริ่มทำงานได้ ก็ต่อเมื่อข้อมูลอินพุตทั้งหมดมีค่าครบถ้วนและพร้อมใช้งาน

โปรแกรม LabVIEW จะสร้างอินเทอร์เฟซของผู้ใช้เป็นแผนภาพ เรียกว่า เสมอเหมือนวัสดุ (VIs) โดยจะระบุหรือแสดงผลลัพธ์ตามที่ผู้ใช้งานกำหนดให้กับ VI ซึ่งการใช้งานโปรแกรมนี้สามารถแบ่งแยกเป็นส่วน ๆ ได้วัตถุทั้งหมดที่วางอยู่บนหน้าจอด้านหน้า จะแสดงที่ด้านหลังเป็นขั้วต่อ หมายความว่าแต่ละ VI จะสามารถทดลองได้ในโปรแกรมย่อย ก่อนจะถูกฝังเป็นโปรแกรมที่ใช้จริงในโปรแกรมใหญ่

2.8.2 เครื่องมือที่ใช้ในโปรแกรม

สำหรับ VI หนึ่ง ๆ จะประกอบด้วยส่วนประกอบ 3 ส่วน ได้แก่ Front Panel, Block Diagram, Icon และ Connector [14]

2.8.2.1 Front Panel

เป็นส่วนที่ใช้สื่อสารกันระหว่างผู้ใช้งานับโปรแกรม นิยมเรียกว่า User Interface โดยทั่วไปมีลักษณะเหมือนกับหน้าปัดของเครื่องมือการวัดต่าง ๆ ซึ่งประกอบไปด้วย

สวิตช์เปิด-ปิด, ปุ่มบิด, ปุ่มกด, และจอแสดงผล ที่ผู้ใช้งานสามารถกำหนดเองได้ เปรียบเสมือนเป็น GUI ของโปรแกรม LabVIEW

อุปกรณ์ที่อยู่บน Front Panel มีอยู่ 3 ประเภท

1) Control ทำหน้าที่รับค่าที่ผู้ใช้กำหนด โดยผู้ใช้จะพิมพ์ค่าลงไป หรือใช้เมาส์คลิก เพื่อเปลี่ยนแปลงค่าให้เป็นตามที่ต้องการ เช่น ปุ่มหมุน ปุ่มเลื่อน สวิตช์ เป็นต้น

2) Indicators ทำหน้าที่แสดงค่าต่าง ๆ เท่านั้น ผู้ใช้จะไม่สามารถแก้ไขได้ เช่น กราฟ มิเตอร์ LED เป็นต้น

3) Decorations เป็นอุปกรณ์ที่ไม่เกี่ยวข้องกับการทำงานของโปรแกรม แต่ใช้เพื่อความสวยงามและความเป็นระเบียบของหน้าจอ Front Panel

2.8.2.2 Block Diagram

ในส่วนนี้เปรียบเสมือน Source code หรือส่วนโปรแกรมหลักของ LabVIEW ซึ่งอยู่ในรูปของภาษา G ถือเป็น Executable program โดยจะสามารถทำงานได้ทันที และมีการตรวจสอบความผิดพลาดของโปรแกรมตลอดเวลา โปรแกรมจะทำงานได้เมื่อไม่มีข้อผิดพลาดในโปรแกรมเท่านั้น โดยผู้ใช้จะสามารถดูรายละเอียดของความผิดพลาดของโปแกรมได้ ทำให้การเขียนโปรแกรมง่ายขึ้นมาก

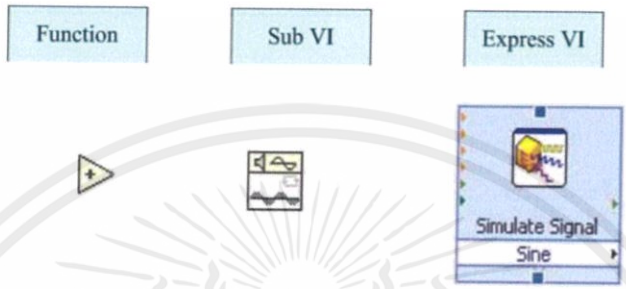
ภายใน Block Diagram ประกอบไปด้วย ฟังก์ชัน ค่าคงที่ โปรแกรมควบคุมการทำงานหรือโครงสร้าง จากนั้น แต่ละส่วนจะแสดงผลในรูปของบล็อก เล็ก ๆ และมีสาย (Wire) ที่จะใช้เชื่อมต่อบล็อกที่เหมาะสมเข้าด้วยกัน เพื่อกำหนดลักษณะการไหลของข้อมูลระหว่างบล็อก ทำให้ข้อมูลได้รับการประมวลผลตามที่ผู้ใช้ต้องการ และแสดงผลออกมาในหน้าจอแสดงผล

อุปกรณ์ที่อยู่บน Block Diagram มีอยู่ 3 ประเภท

1) Function เป็นหน้าที่พื้นฐานของเครื่องคอมพิวเตอร์ ซึ่งผู้ใช้ไม่สามารถดูรายละเอียดภายในบล็อกได้ เช่น การบวก การคูณ

2) SubVIs หรือเรียกอีกชื่อหนึ่งว่า Subprogram คือโปรแกรมย่อยที่ถูกเขียนขึ้นเพื่อนำมาใช้งานในอีกโปรแกรมหนึ่ง ผู้ใช้สามารถเปิดเข้าไปดูรายละเอียดของ Front panel และ Block Diagram ของโปรแกรมย่อยได้ เมื่อคลิกที่ Icon ของอุปกรณ์

3) Express VIs เป็น SubVIs ชนิดหนึ่ง ซึ่งเมื่อผู้ใช้เลือก Express VI มาวางบน Block Diagram จะแสดงหน้าต่าง Configuration ขึ้นมา เพื่อให้ผู้ใช้งานป้อนค่าต่าง ๆ ตามที่ต้องการ และเมื่อป้อนเสร็จ จะสร้าง code ไว้ภายในโดยอัตโนมัติ โดย Express VIs จะมี Icon ของอุปกรณ์ใหญ่ และมีพื้นหลังเป็นสีฟ้า ดังรูปที่ 2.7



รูปที่ 2.7 ตัวอย่าง Block Diagram Node [14]

2.8.2.3 Icon และ Connector

ในส่วนนี้เปรียบเสมือนโปรแกรมย่อย Subroutine ในโปรแกรมทั่วไป โดย Icon จะหมายถึง Block Diagram ตัวหนึ่งที่มีการส่งข้อมูลผ่านทาง Connector ซึ่งในโปรแกรม LabVIEW จะเรียก Subroutine นี้ว่า SubVIs ลักษณะทั่วไปของ Icon และ Connector จะแสดงดังรูปที่ 2.8 โดย Connector จะมีช่องต่อข้อมูลที่เรียกว่า Terminal แสดงให้เห็น

รูปที่ 2.8 ลักษณะทั่วไปของ Icon และ Connector [14]

2.9 ภาษา Python

Python [15] คือ ชื่อภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่ง ซึ่งถูกพัฒนาขึ้นมาโดยไม่ยึดติดกับแพลตฟอร์ม กล่าวคือสามารถรันภาษา Python ได้ทั้งบนระบบ Unix, Linux, Windows NT, Windows 2000, Windows XP ภาษา Python เป็น Open Source เหมือนกับ

PHP ทำให้ทุกคนสามารถที่นำ Python มาพัฒนาโปรแกรมได้ และความเป็น Open Source ทำให้มีคนเข้ามาช่วยกันพัฒนาให้ Python มีความสามารถสูงขึ้น และใช้งานได้ครบคุมกับทุกลักษณะงาน

2.9.1 การทำงานของภาษา Python

ในการเขียนภาษา Python จะต้องมีการศึกษาการทำงานของโปรแกรมและภาษาก่อน ซึ่งจะมีหลักการทำงาน ดังนี้

1) สนับสนุนแนวแบบคิดออปเจกต์โอเรียนเทด (OOP: Object Oriented Programming)

2) เป็น Open Source

3) Source Code ที่เขียนด้วยภาษา Python สามารถนำไปใช้บนระบบปฏิบัติการได้หลากหลาย

4) สนับสนุนเทคโนโลยี COM ของ Ms-windows

5) Python รวมมาตรฐานการอินเตอร์เฟส Tkinter ซึ่งสนับสนุนบนระบบ X windows, Ms-windows และ Macintosh โดยการใช้คำสั่ง Tkinter API ช่วยให้โปรแกรมเมอร์ไม่ต้องแก้ไขโค้ดเมื่อนำไปใช้บนระบบปฏิบัติการอื่น ๆ

6) เป็น Dynamic typing คือ สามารถเปลี่ยนชนิดข้อมูลได้ง่ายและสะดวก

7) มี Build-in Object Types คือ โครงสร้างของข้อมูลที่สามารถใช้ได้ใน Python ประกอบด้วย ลิสต์, ดิกชันนารี, สตริง ที่ง่ายต่อการใช้งานและมีประสิทธิภาพสูง

8) มีเครื่องมือต่าง ๆ มากมาย เช่น การประมวลผลเท็กซ์ไฟล์ การเรียงข้อมูล การเชื่อมต่อสตริง การตรวจสอบเงื่อนไขของข้อความ การแทนคำ เป็นต้น

9) มีโมดูลสำหรับจัดการ Regular Expression

10) มีมอดูลที่สร้างขึ้นจากนักพัฒนาสนับสนุนมากมาย ได้แก่ COM, Image, CORBA, ORBs, XML เป็นต้น

11) จัดการหน่วยความจำอย่างอัตโนมัติ สามารถจักการพื้นที่หน่วยความจำที่ไม่ต่อเนื่องให้ทำงานได้อย่างมีประสิทธิภาพ

12) อนุญาตให้ฝังชุดคำสั่งของ Python เอาไว้ภายใน Code ภาษา C/C++ ได้

13) อนุญาตให้โปรแกรมเมอร์สร้าง Dynamic Link Libray (DLL) เพื่อใช้ร่วมกับ Python

14) มีมอดูลสนับสนุนเกี่ยวกับ Network process thread regular, expression, xml, GUI และอื่น ๆ

15) ประกอบด้วยมอดูลสำหรับสร้าง Internet Script และติดต่อกับอินเทอร์เน็ตผ่าน Sockets และทำหน้าที่เป็น CGI Script และใช้งานคำสั่ง FTP, Glopher, XML และอื่น ๆ

16) สามารถประมวลผลทางด้านวิทยาศาสตร์ และวิศวกรรมศาสตร์ได้อย่างมีประสิทธิภาพ

17) มีฟังก์ชันสนับสนุนฐานข้อมูล เช่น MySQL, Sybase, , Informix, ODBC และอื่น ๆ

18) มีไลบรารีสนับสนุนด้านการสร้างภาพกราฟฟิก เช่น ทำภาพเบลอ หรือภาพชัด หรือเขียนข้อความบนภาพ ตลอดจนบันถึกไฟล์ในรูปแบบต่าง ๆ ได้อย่างสะดวกและมีประสิทธิภาพ

19) มีไลบรารีสนับสนุนด้านปัญญาประดิษฐ์

20) มีไลบรารีสำหรับสร้างเอกสาร PDF โดยไม่ต้องติดตั้ง Acrobat Writer

21) มีไลบรารีสำหรับสร้าง Shockwaves Flash (SWF) โดยไม่ต้องติดตั้ง Macromedia Flash

2.9.2 จุดเด่นของภาษา Python [16]

จุดเด่นที่ทำให้ภาษา Python ไม่เหมือนภาษาทางคอมพิวเตอร์อื่น ๆ มีดังนี้

1) ความเป็นภาษาสคริปต์

เนื่องจาก Python เป็นภาษาสคริปต์ ทำให้ใช้เวลาในการเขียนและคอมไพล์ไม่มาก ทำให้เหมาะกับงานด้านการดูแลระบบ (System administration) และมีการสนับสนุนภาษา Python โดยเป็นส่วนหนึ่งของระบบปฏิบัติการ Unix, Linux และสามารถติดตั้งให้ทำงานเป็นภาษาสคริปต์ของ Windows ผ่านระบบ en:Windows Script Host ได้

2) ไวยากรณ์ที่อ่านง่าย

ไวยากรณ์ของ Python ได้กำจัดการใช้สัญลักษณ์ที่ใช้ในการแบ่งบล็อกของโปรแกรม และใช้การย่อหน้าแทน ทำให้สามารถอ่านโปรแกรมที่เขียนได้ง่าย และมีการสนับสนุนการเขียน docstring ซึ่งเป็นข้อความสั้น ๆ ที่ใช้อธิบายการทำงานของฟังก์ชัน, คลาส และโมดูล

2.9.3 ภาษา Python ในรูปแบบต่าง ๆ

1) ซีไพทอน (CPython)

คือรูปแบบภาษา Python ดั้งเดิม โปรแกรมอินเทอร์พรีเตอร์ถูกเขียนโดยภาษาซี ซึ่งจะใช้ได้บนหลายระบบปฏิบัติการ เช่น Windows, Unix, Linux การใช้งานสามารถทำได้โดยการติดตั้งโปรแกรมอินเทอร์พรีเตอร์และแพ็คเกจที่จำเป็นต่าง ๆ

2) ไจทอน (Jython)

เป็นแพลตฟอร์มภาษา Python ที่ถูกพัฒนาเพื่อใช้บน JAVA เพื่อเพิ่มอำนวยความสะดวกในการใช้ความสามารถภาษาสคริปต์ของ Python ลงในซอฟต์แวร์ JAVA อื่น ๆ การใช้งานสามารถทำได้โดยการติดตั้ง JAVA และเรียกไลบรารีของ Jython ซึ่งมาในรูปไบนารีเพื่อใช้งาน

3) ไพทอนดอตเน็ต (Python.NET)

เป็นการพัฒนาภาษา Python ให้สามารถทำงานบน .NET Framework ของ Microsoft ได้ โดยโปรแกรมที่ถูกเขียนจะถูกแปลงเป็น CLR ปัจจุบันมีโครงการที่นำภาษา Python มาใช้บน .NET Framework ของ Microsoft แล้วคือโครงการ IronPython

2.9.4 ตัวแก้ไขสำหรับ Python

ผู้ใช้สามารถใช้ตัวแก้ไขข้อความทั่วไปในการแก้ไขโปรแกรมภาษา Python นอกจากนั้นยังมี Integrated Development Environment อื่น ๆ ให้เลือกใช้อีกมากมาย ยกตัวอย่างได้ดังนี้

1) PyScripter เป็นชุดเครื่องมือสำหรับพัฒนาภาษาไพธอน บนระบบปฏิบัติการวินโดวน์ ที่ให้ผู้ใช้สามารถนำไปใช้ฟรี (open source)

2) Python IDLE มีอยู่ในชุดอินเตอร์พรีเตอร์อยู่แล้ว สามารถเลือกติดตั้งได้

3) PythonWin เป็นตัวแก้ไขในชุดของ PyWin32

4) ActivePython จาก ActiveState (ล่าสุด รุ่น 2.5.1)

5) SPE (Stani's Python Editor) เป็นตัวแก้ไขที่มาพร้อมกับตัวออกแบบยูสเซอร์อินเทอร์เฟส wxGlade และเครื่องมือสำหรับ Regular Expression มีระบบ Syntax Highlight และการจัดย่อหน้าตามวากยสัมพันธ์ของ Python ให้อัตโนมัติพัฒนาขึ้นจากภาษา Python

6) WingIDE ตัวแก้ไขที่มีระบบ Syntax Highlight และการจัดย่อหน้าตามไวยกรณ์ของ Python ให้อัตโนมัติ แต่ไม่ใช่ Freeware

7) Komodo ตัวแก้ไขที่มีระบบ Syntax Highlight, การจัดย่อหน้าตามไวยกรณ์ของ Python ให้อัตโนมัติและเติมคำอัตโนมัติ เป็นตัวแก้ไขจาก ActiveState อีกตัวหนึ่ง ไม่ใช่ Freeware

8) Pydev เป็น Python IDE สำหรับ Eclipse สามารถใช้พัฒนา Python, Jython และ Ironpython

9) PyCharm เป็น Python IDE ที่สร้างขึ้นโดยบริษัท JetBrains แบ่งออกเป็น 2 Version ได้แก่ Community Edition (ใช้งานฟรี) และ Professional Edition (เสียเงิน สามารถทดลองใช้ได้ 30 วัน) โดย Professional Edition จะเพิ่มความสามารถในการตรวจ syntax ของ Framework ที่ได้รับความนิยมที่ใช้งานร่วมกับภาษา Python เช่น Django, Flask, Google App Engine เป็นต้น

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้จะกล่าวถึงการทำงานของสถานีภาคพื้นดินเท่านั้น โดยสถานีภาคพื้นดินของระบบนี้จะทำหน้าที่รับข้อมูลต่าง ๆ ที่รับมาจากแบบจำลองดาวเทียมกระป๋องในส่วน Payload ผ่านโมดูล XBEE Pro Series1 ที่เป็นตัวกลางในสื่อสารระหว่างภาคส่งและภาครับ โดยจะใช้สายอากาศยากิ-อูดะ ซึ่งจะถูกติดตั้งอยู่ที่ระบบมอเตอร์เคลื่อนที่ 2 แกน ที่มีระบบเซ็นเซอร์ติดตั้งอยู่ เพื่อช่วยอำนวยความสะดวกในการติดตามแบบจำลองดาวเทียมกระป๋องและแบบจำลองดาวเทียมขนาดเล็ก จากนั้นข้อมูลที่รับได้จะถูกนำไปประมวลผลและแสดงผลทางคอมพิวเตอร์ ผ่านโปรแกรม LabVIEW ตามที่ได้ออกแบบไว้ และยังมีระบบติดตามแบบจำลองดาวเทียมขนาดเล็ก ซึ่งจะดึงข้อมูลต่าง ๆ แบบ Real-Time จากเว็บไซต์ N2YO.com

3.1 การออกแบบ

การออกแบบในปริญญานิพนธ์นี้ จะแสดงขั้นตอนการออกแบบของโครงสร้างในส่วนต่าง ๆ ภายในระบบสถานีภาคพื้นดิน ซึ่งแบ่งออกเป็น 5 ระบบใหญ่ ๆ ดังนี้

3.1.1 การเลือกใช้สายอากาศ

3.1.1.1 การเลือกใช้สายอากาศสำหรับติดตามแบบจำลองดาวเทียมกระป๋อง (CanSat)

ในการรับสัญญาณจาก CanSat จะต้องใช้สายอากาศที่มีความถี่ตาม CanSat Competition 2019 กำหนดไว้ ซึ่งตรงกับ XBEE ที่ใช้ นั่นคือที่ความถี่ 2.4 GHz เราจึงเลือกใช้สายอากาศ Yagi 25dBi 2.4GHz Wireless WLAN WiFi RP-SMA ซึ่งมีลักษณะดังรูปที่ 3.1 และมีคุณสมบัติดังตารางที่ 3.1

รูปที่ 3.1 สายอากาศ Yagi 25dBi 2.4GHz Wireless WLAN WiFi RP-SMA [17]

ตารางที่ 3.1 คุณสมบัติของสายอากาศ Yagi 25dBi 2.4GHz Wireless WLAN WiFi RP-SMA

คุณสมบัติ	รายละเอียด
Dimensions	50 x 7 x 3cm/ 19.7" x 2.7" x 1.2" (L*W*T)
Cable Length	150cm
Main Material	Plastic, Metal
Main Color	Black, Silver Tone
Weight	278g
Support	2.4 GHz Frequency Only
Gain	25dBi
Rated Power	30W
Input Impedance	50 Ohms * Package (1 x Wireless WLAN Wifi Antenna)

3.1.1.2 การเลือกใช้สายอากาศสำหรับติดตามแบบจำลองดาวเทียมขนาดเล็ก (CubeSat)

ในการรับสัญญาณจาก CubeSat ที่อยู่สูงจากพื้นโลกหลายร้อยกิโลเมตรนั้น จำเป็นต้องมีองค์ประกอบหลายอย่างเพื่อให้สัญญาณที่รับมานั้นมีคุณภาพที่สุด เช่น ประสิทธิภาพในการส่งสัญญาณของดาวเทียม เมฆฝน สัญญาณรบกวนต่าง ๆ แต่สิ่งที่สำคัญที่สุดในการรับสัญญาณ

คือ คุณสมบัติของสายอากาศ โดยสายอากาศที่ใช้ คือ สายอากาศ 433 MHz 12 dBi High Gain Long Range UHF Yagi ซึ่งมีลักษณะดังรูปที่ 3.2 และมีคุณสมบัติดังตารางที่ 3.2

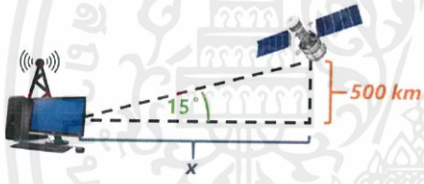
ตารางที่ 3.2 คุณสมบัติของสายอากาศ 144/430 MHz Dual Band Handheld

คุณสมบัติ	รายละเอียด
Model	AM433Y12V
Frequency Range	433 MHz
Polarization	Vertical
Gain	12 dBi
Nominal Impedance	50 Ohm
Maximum Power	50W
VSWR	≤2.0
Lightning Protection	DC Ground
Connector	N Female
Dimensions	1300mm
Weight	1.8kg
Material	Aluminum Alloy
Mounting Type	Mast Mount
Mouting Kits	2*U bolts
Mounting Mast Size	Φ30 - Φ50mm
Max. Wind Velocity	210km/h

รูปที่ 3.2 สายอากาศ 433 MHz 12 dBi High Gain Long Range UHF Yagi [18]

3.1.1.3 ความสามารถในการรับสัญญาณของสายอากาศ

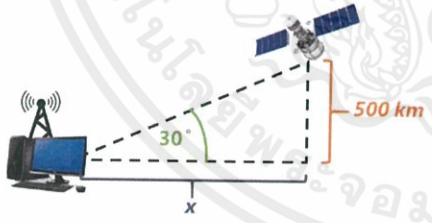
ถ้าต้องการสัญญาณที่คุณภาพสูง เราจะต้องวางสายอากาศในตำแหน่งที่เหมาะสมโดยพิจารณาทั้งในแนวแกน x ซึ่งคือมุม Azimuth และ แนวแกน y ซึ่งคือมุม Elevation ในที่นี้จะพิจารณามุม Elevation เป็นพิเศษ กำหนดให้ความสูงจาก CubeSat = 500 km



รูปที่ 3.3 CubeSat ทำมุม 15 องศากับสถานีภาคพื้นดิน

จะได้ $x = \frac{500km}{\sin(15)} = 1931.85km$

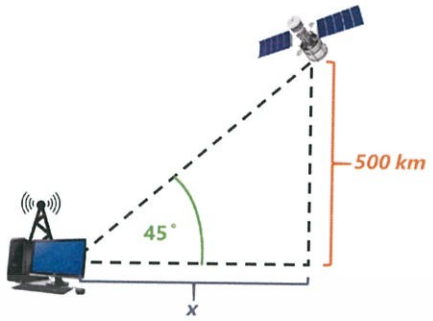
ดังนั้น ถ้าวางสายอากาศที่มุม 15 องศา จะสามารถรับสัญญาณได้ครอบคลุมในรัศมี 1932 กิโลเมตรโดยประมาณ



รูปที่ 3.4 CubeSat ทำมุม 30 องศากับสถานีภาคพื้นดิน

จะได้ $x = \frac{500km}{\sin(30)} = 1000km$

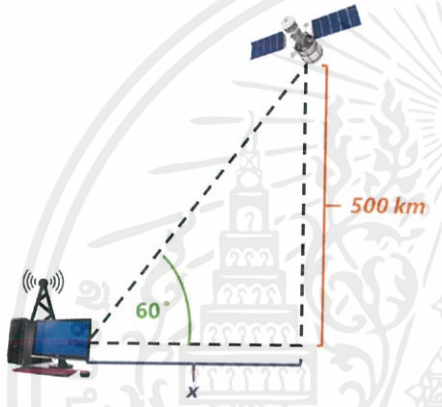
ดังนั้น ถ้าวางสายอากาศที่มุม 30 องศา จะสามารถรับสัญญาณได้ครอบคลุมในรัศมี 1000 กิโลเมตรโดยประมาณ



รูปที่ 3.5 CubeSat ทำมุม 45 องศากับสถานีภาคพื้นดิน

จะได้ $x = \frac{500km}{\sin(45)} = 707.11km$

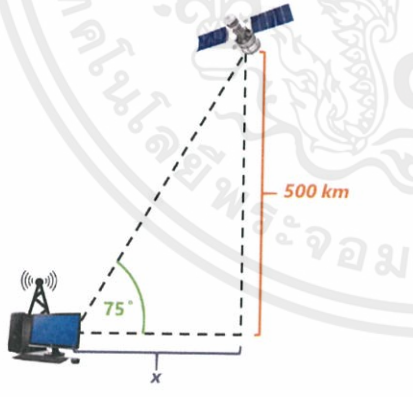
ดังนั้น ถ้าวางสายอากาศที่มุม 45 องศา จะสามารถรับสัญญาณได้ครอบคลุมในรัศมี 707.11 กิโลเมตรโดยประมาณ



รูปที่ 3.6 CubeSat ทำมุม 60 องศากับสถานีภาคพื้นดิน

จะได้ $x = \frac{500km}{\sin(60)} = 577.35km$

ดังนั้น ถ้าวางสายอากาศที่มุม 60 องศา จะสามารถรับสัญญาณได้ครอบคลุมในรัศมี 577.35 กิโลเมตรโดยประมาณ



รูปที่ 3.7 CubeSat ทำมุม 75 องศากับสถานีภาคพื้นดิน

จะได้ $x = \frac{500km}{\sin(75)} = 517.64km$

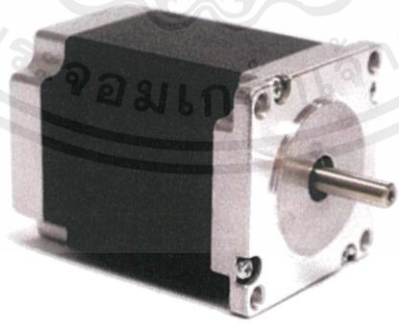
ดังนั้น ถ้าวางสายอากาศที่มุม 75 องศา จะสามารถรับสัญญาณได้ครอบคลุมในรัศมี 517.64 กิโลเมตรโดยประมาณ

จะเห็นได้ว่ายิ่งวางสายอากาศด้วยมุม Elevation ที่น้อยเท่าไร จะยิ่งรับสัญญาณได้ครอบคลุมมากขึ้น แต่ความคิดนี้เป็นเพียงหลักอุดมคติเท่านั้น เพราะในความเป็นจริงการวางมุม Elevation ไว้น้อย ๆ จะทำให้สัญญาณที่ได้รับนั้นมีคุณภาพที่แย่ลง หรือไม่มีคุณภาพเนื่องจากคุณสมบัติต่าง ๆ ของสายอากาศ ดังนั้น เมื่อพิจารณาถึงปัจจัยทั้ง 2 อย่าง คือ การรับสัญญาณได้กว้าง ครอบคลุม และ การได้สัญญาณที่มีคุณภาพ ควรจะให้มุม Elevation ที่น้อยที่สุด คือ 30 องศา เพื่อให้รับสัญญาณได้ครอบคลุมและมีคุณภาพ

3.1.2 การออกแบบระบบมอเตอร์เคลื่อนที่ 2 แกน

3.1.2.1 การออกแบบโครงสร้างของระบบมอเตอร์ด้วยโปรแกรม Fusion 360 และ Solid work

เนื่องจากต้องการให้โครงสร้างของระบบสายอากาศสำหรับติดตามแบบจำลองดาวเทียมกระป๋อง (CanSat) มีขนาดเล็ก สามารถพกพาได้ง่าย จึงออกแบบให้โครงสร้างเป็นลักษณะตัว L ที่สามารถตั้งบนขาตั้งกล้องได้ โดยตัวโครงสร้างจะใช้อุปกรณ์ให้น้อยที่สุด เพื่อให้มีน้ำหนักเบาโดยสายอากาศที่ใช้ในปริญญานิพนธ์นี้ คือ สายอากาศ Yagi 25dBi Wireless WLAN WiFi RP-SMA ซึ่งมีความถี่กลางที่ 2.4 GHz และเพื่อให้สามารถหมุนเพื่อติดตามแบบจำลองดาวเทียมกระป๋องได้ทั้งแนวแกนตั้งและแนวแกนนอน การออกแบบจึงมีมอเตอร์สำหรับหมุนแต่ละแนวแกนอย่างละ 1 ตัว ดังนั้นจึงมีมอเตอร์ที่ใช้ทั้งหมด 2 ตัว คือ Stepper Motor NEMA 23 ดังรูปที่ 3.8 ซึ่งจะนำไปใช้ในการหมุนตามแนวแกนตั้งและแกนนอน โดยจะมีโครงสร้างที่ออกแบบได้ดังรูปที่ 3.9 และมีขนาดและน้ำหนักเมื่อไม่ได้คิดรวมกับสายอากาศดังตารางที่ 3.3



รูปที่ 3.8 Stepper Motor NEMA 23 [19]

ตารางที่ 3.3 โครงสร้างของระบบมอเตอร์สำหรับแบบจำลองดาวเทียมกระป๋อง

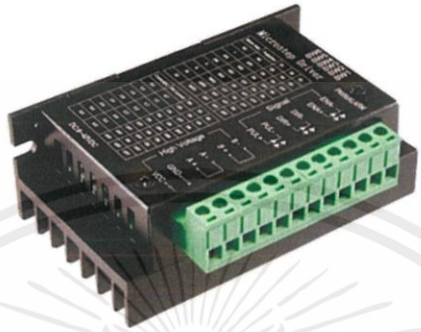
ความกว้าง	ความยาว	น้ำหนัก (โดยประมาณ)
36 เซนติเมตร	63 เซนติเมตร	3.825 กิโลกรัม

รูปที่ 3.9 โครงสร้างของระบบมอเตอร์สำหรับแบบจำลองดาวเทียมกระป๋อง

โดยโครงสร้างของระบบมอเตอร์นี้ จะประกอบไปด้วยอะลูมิเนียมโปรไฟล์ (Aluminum Profile) ดังรูปที่ 3.10 ซึ่งเป็นวัสดุคล้ายคลึงกับโลหะ แต่มีน้ำหนักเบา ทนทานต่อการสึกกร่อนมากกว่า และมีความสามารถในการรับแรงกดดันสูง

รูปที่ 3.10 อะลูมิเนียมโปรไฟล์ [20]

ในส่วนของ Stepping Motor ที่ใช้ในระบบมอเตอร์เพื่อบังคับตัวเครื่องให้หมุนไปในทิศทางที่ต้องการ เราจะใช้ TB6600 Stepper Motor Driver ดังรูปที่ 3.11 ซึ่งทำหน้าที่เชื่อมต่อกับตัวไมโครคอนโทรลเลอร์ของระบบ ซึ่งก็คือโปรแกรม Arduino IDE

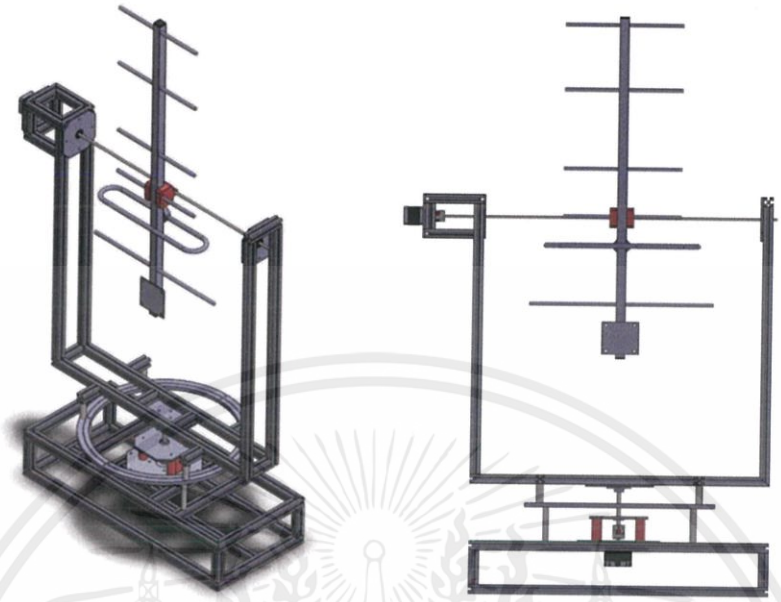


รูปที่ 3.11 TB6600 Stepper Motor Driver [21]

ส่วนระบบสายอากาศสำหรับติดตามแบบจำลองดาวเทียมขนาดเล็ก (CubeSat) จะใช้ตัวอุปกรณ์ภายในเหมือนกันกับระบบสายอากาศสำหรับติดตามแบบจำลองดาวเทียมกระป๋อง แต่ลักษณะโครงสร้างภายนอกจะไม่เหมือนกัน โดยจะมีลักษณะโครงสร้างแสดงดังรูปที่ 3.12 และมีขนาดและน้ำหนักเมื่อไม่ได้คิดรวมกับสายอากาศดังตารางที่ 3.4 และเนื่องจากต้องการให้ระบบมอเตอร์สามารถประยุกต์ใช้ได้กับหลากหลายสายอากาศ จึงออกแบบให้ตัวโครงสร้างมีความกว้างและความยาวรองรับสายอากาศได้ตั้งแต่ขนาดใหญ่จนถึงขนาดเล็ก โดยในปริญญานิพนธ์นี้จะใช้สายอากาศ 433 MHz 12 dBi High Gain Long Range UHF Yagi

ตารางที่ 3.4 โครงสร้างของระบบมอเตอร์สำหรับแบบจำลองดาวเทียมขนาดเล็ก

ความกว้าง	ความยาว	น้ำหนัก (โดยประมาณ)
77 เซนติเมตร	92 เซนติเมตร	4.70 กิโลกรัม



รูปที่ 3.12 โครงสร้างของระบบมอเตอร์สำหรับ CubeSat

3.1.2.2 การคำนวณมุมต่อ 1 Step ของระบบมอเตอร์

โดยปกติ Stepper Motor NEMA 23 จะมีค่าอัตราการหมุนของมอเตอร์อยู่ที่ 200 pulses/rev (ใช้ 200 พัลส์ต่อการหมุน 1 รอบ) หรือมุมต่อ 1 สเตปอยู่ที่ 360/200 = 1.8 องศา แต่เราต้องการความละเอียดที่มากกว่านี้ จึงเพิ่มความละเอียดโดยการใช้โหมดของ TB6600 Stepper Motor Driver เป็น OFF OFF ON OFF OFF OFF หรือก็คือ โหมดที่ใช้ Microstep 1/16 ดังตารางที่ 3.5 จะมีค่า 3200 pulses/rev (ใช้ 3200 พัลส์ต่อการหมุน 1 รอบ) สามารถคำนวณหามุมต่อ 1 สเตปได้ ดังนี้

3200 พัลส์ = 360 องศา

1 พัลส์ = 360/3200 องศา = 0.1125 องศา

ดังนั้น จะได้ว่ามุมต่อ 1 สเตป ในโหมด Microstep 1/16 คือ 0.1125 องศา

3.1.2.3 การคำนวณมุมการหมุนของระบบมอเตอร์

1) การคำนวณมุมการหมุนมอเตอร์ในแนวแกนนอน

การคำนวณนี้จะคำนวณโดยอ้างอิงจากละติจูดและลองติจูดระหว่างตำแหน่ง 2 ตำแหน่งที่มีระยะห่างกันในแนวระนาบกับพื้น โดยมีสมการในการคำนวณ ดังนี้

ตารางที่ 3.5 Microstep Driver ของ TB6600 Stepper Motor Driver

Micro Step	Pulse/rev	S1	S2	S3
NC	NC	ON	ON	ON
1	200	ON	ON	OFF
2/A	400	ON	OFF	ON
2/B	400	OFF	ON	ON
4	800	ON	OFF	OFF
8	1600	OFF	ON	OFF
16	3200	OFF	OFF	ON
32	6400	OFF	OFF	OFF
Current (A)	Pulse/rev	S4	S5	S6
0.5	0.7	ON	ON	ON
1.0	1.2	ON	OFF	ON
1.5	1.7	ON	ON	OFF
2.0	2.2	ON	OFF	OFF
2.5	2.7	OFF	ON	ON
2.8	2.9	OFF	OFF	ON
3.0	3.2	OFF	ON	OFF
3.5	4.0	OFF	OFF	OFF

$$\beta = a\tan 2(X,Y) \tag{3.1}$$

$$X = \cos\theta_b \times \sin\Delta L \tag{3.2}$$

$$Y = \cos\theta_a * \sin\theta_b - \sin\theta_a * \cos\theta_b * \cos\Delta L \tag{3.3}$$

โดย β คือมุมในแนวระนาบระหว่างตำแหน่งอ้างอิง

a คือตำแหน่งอ้างอิงที่ 1

b คือตำแหน่งอ้างอิงที่ 2

L คือลองติจูด

θ คือละติจูด

2) การคำนวณมุมการหมุนมอเตอร์ในแนวแกนตั้ง

การคำนวณนี้จะคำนวณโดยอ้างอิงจากความสูงระหว่างตำแหน่ง 2 ตำแหน่งและระยะห่างระหว่างตำแหน่ง 2 ตำแหน่งในแนวระนาบกับพื้น โดยมีสมการในการคำนวณ ดังนี้

$$dlon = lon2 - lon1 \tag{3.4}$$

$$dlat = lat2 - lat1 \tag{3.5}$$

$$a = \sin(\frac{dlat}{2})^2 + \cos(lat1) * \cos(lat2) * \sin(\frac{dlon}{2})^2 \tag{3.6}$$

$$c = 2 * a\tan 2(\sqrt{a}, \sqrt{1-a})^2 \tag{3.7}$$

$$dlon = lon2 - lon1 \tag{3.8}$$

$$d = R * c \tag{3.9}$$

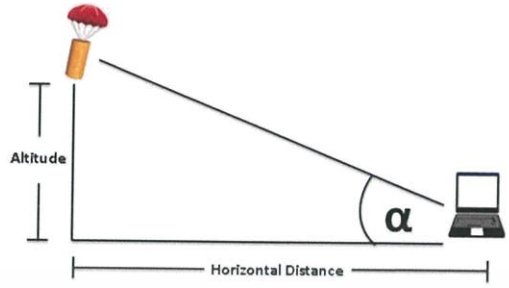
$$\tag{3.10}$$

โดย R คือรัศมีของโลก ซึ่งมีค่าเท่ากับ 6373 กิโลเมตร

d คือระยะห่างระหว่างตำแหน่งอ้างอิง 2 ตำแหน่ง

b คือตำแหน่งอ้างอิงที่ 2

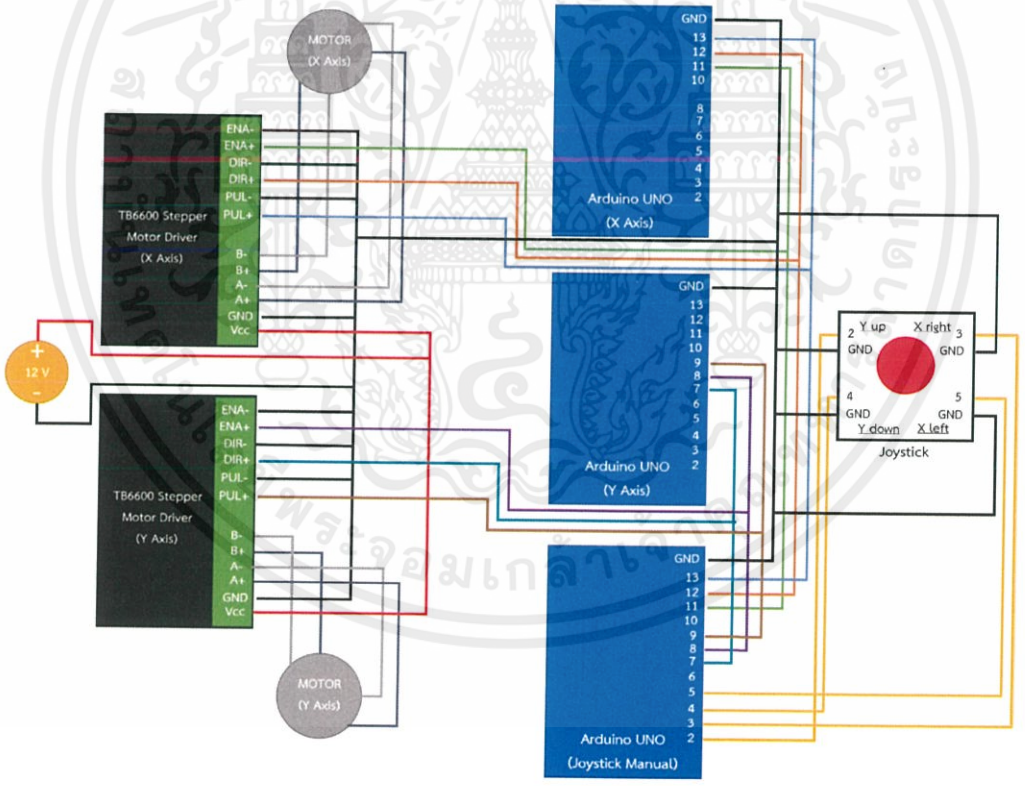
หลังจากคำนวณมุมการหมุนมอเตอร์ทั้ง 2 แกนแล้ว จะได้ค่ามุมออกมา ดังที่แสดงในรูปที่ 3.13



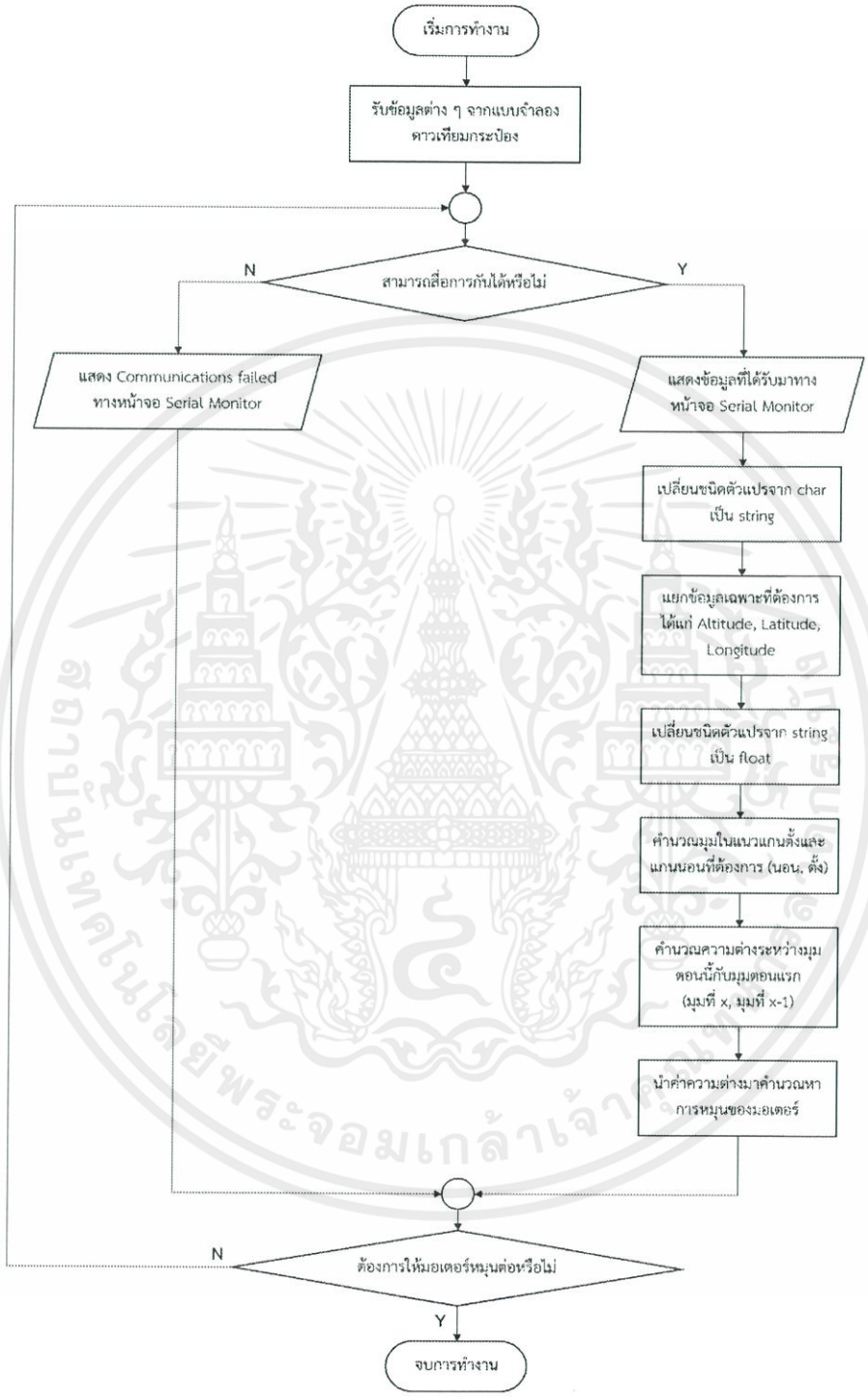
รูปที่ 3.13 การหามุมการหมุนมอเตอร์ในแนวแกนตั้ง

3.1.2.4 ขั้นตอนการทำงานของระบบมอเตอร์

ภาพรวมของระบบมอเตอร์เคลื่อนที่ 2 แกนสามารถแสดงได้ดังรูปที่ 3.14 และขั้นตอนการทำงานของระบบมอเตอร์เคลื่อนที่ 2 แกนสามารถแสดงหลักการทำงานเป็นโฟล์วชาร์ตได้ดังรูปที่ 3.15



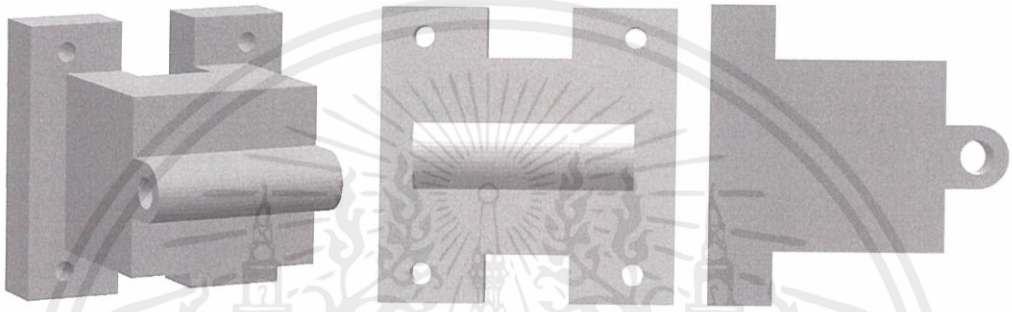
รูปที่ 3.14 ภาพรวมของระบบมอเตอร์เคลื่อนที่ 2 แกน



รูปที่ 3.15 โฟล์วชาร์ตการทำงานของระบบมอเตอร์เคลื่อนที่ 2 แกน

3.1.2.5 การออกแบบตัวต่อระหว่างขาตั้งของระบบมอเตอร์กับสายอากาศ

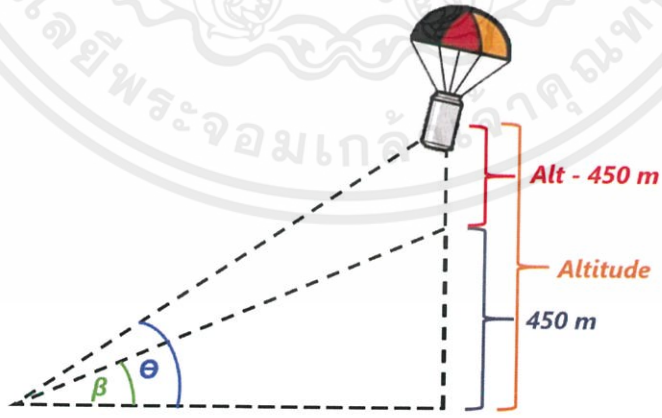
เนื่องจากสายอากาศต้องมีการติดตั้งให้ติดอยู่กับมอเตอร์เคลื่อนที่ 2 แกน เพื่อให้สามารถ Tracking ตามการเคลื่อนที่แบบจำลองดาวเทียมได้ จึงต้องทำการออกแบบตัวต่อที่จะใช้เชื่อมระหว่างขาตั้งของมอเตอร์เคลื่อนที่ 2 แกนกับสายอากาศยากิ-อูดะ ซึ่งต้องใช้ 3D Print ในการสร้าง โดยทำการออกแบบผ่านโปรแกรม Solid work ซึ่งมีโครงสร้างของอุปกรณ์ดังรูปที่ 3.16



รูปที่ 3.16 โครงสร้างของตัวต่อระหว่างขาตั้งของมอเตอร์เคลื่อนที่ 2 แกนกับสายอากาศ

3.1.2.6 การคำนวณเวลาในการขยับแต่ละ Step ของระบบมอเตอร์

จากเงื่อนไขในการแข่งขัน CanSat Competition 2019 ที่กำหนดไว้ว่า แบบจำลองดาวเทียมต้องตกลงสู่พื้นด้วยความเร็วประมาณ 20 เมตรต่อวินาที ที่ความสูงมากที่สุดประมาณ 700 เมตร และความเร็วจะลดลงเมื่อถึงระยะ 450 เมตรจากพื้น ด้วยความเร็วประมาณ 10 เมตรต่อวินาที การคำนวณองศาของมอเตอร์เคลื่อนที่ 2 แกนแสดงได้ดังรูปที่ 3.17



รูปที่ 3.17 การคำนวณองศาของมอเตอร์เคลื่อนที่ 2 แกน

ที่ระยะ Alt – 450 m มีความเร็วประมาณ 20 เมตรต่อวินาที

จาก v = s / t จะได้ 20 = (Alt – 450) / T1

ดังนั้น T1 = (Alt – 450) / 20 sec

ที่ระยะ 450 m มีความเร็วประมาณ 10 เมตรต่อวินาที

จาก v = s / t จะได้ 10 = 450 / T2

ดังนั้น T2 = 45 sec

การหา θ จะใช้ Sensor ADXL345 โดยใช้สูตรคำนวณ

$$\theta = -\ atan2\ (x, sqrt\ \ y * y + z * z\ \ * 180.0) / M\_PI; \tag{3.11}$$

เมื่อมอเตอร์เคลื่อนที่ 2 แกนใช้โหมด Microstep 16 จะได้ 3200 pulse/rev.

จะได้ 3200 Steps = 360 องศา

ดังนั้น 1 Step = 0.1125 องศา

การหาเวลาที่ใช้ในการขยับแต่ละ Step จะให้เวลาในช่วง Alt – 450 m เป็น t1 และ ช่วง 450 m เป็น t2

จะได้ $\tan\theta$ = Alt / x

ดังนั้น x = Alt / $\tan\theta$ เมื่อ x คือ ระยะแนวราบ

การหา β จะหาได้จากสมการ

$$\beta = \arctan(450 / x) \tag{3.12}$$

ระยะเวลา T1 = t1 x [(θ - β) / 0.1125]

ดังนั้น t1 = (Alt – 450 / 20) x (0.1125 / (θ - β))

ระยะเวลา T2 = t2 x [β / 0.1125]

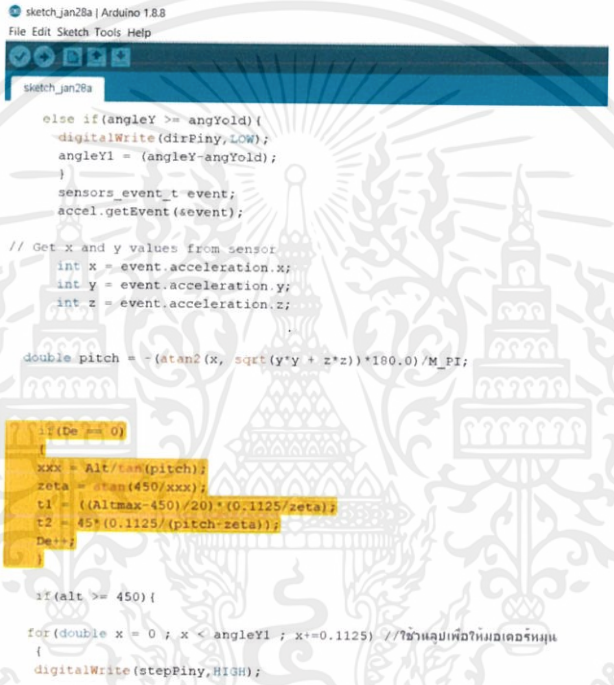
ดังนั้น t2 = 45 x (0.1125 / β)

จากการคำนวณดังกล่าว สามารถเขียน Source code และแสดงใน โปรแกรม ArduinoIDE ได้ ดังรูปที่ 3.18 โดยจะใช้ตัวแปร De เป็นตัวเช็คให้ทำงานใน loop เพียงครั้งเดียว คือครั้งที่เริ่มต้นการสื่อสารระหว่าง CanSat กับ Ground Station

เมื่อ xxx คือ ระยะในแนวราบจากสถานีภาคพื้นดินถึง CanSat

pitch คือ มุมที่รับจากเซ็นเซอร์ หรือมุมจากพื้นถึง CanSat

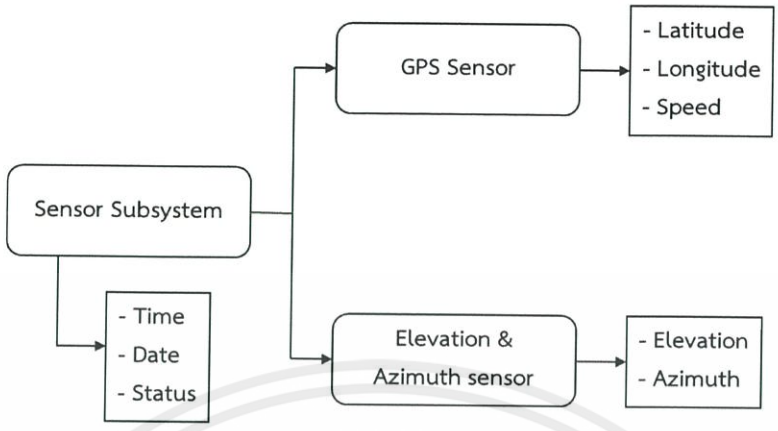
zeta คือ มุมจากพื้นถึงที่ระยะ 450 เมตร



รูปที่ 3.18 การคำนวณเวลาในการขยับแต่ละ Step ของระบบมอเตอร์ในโปรแกรม ArduinoIDE

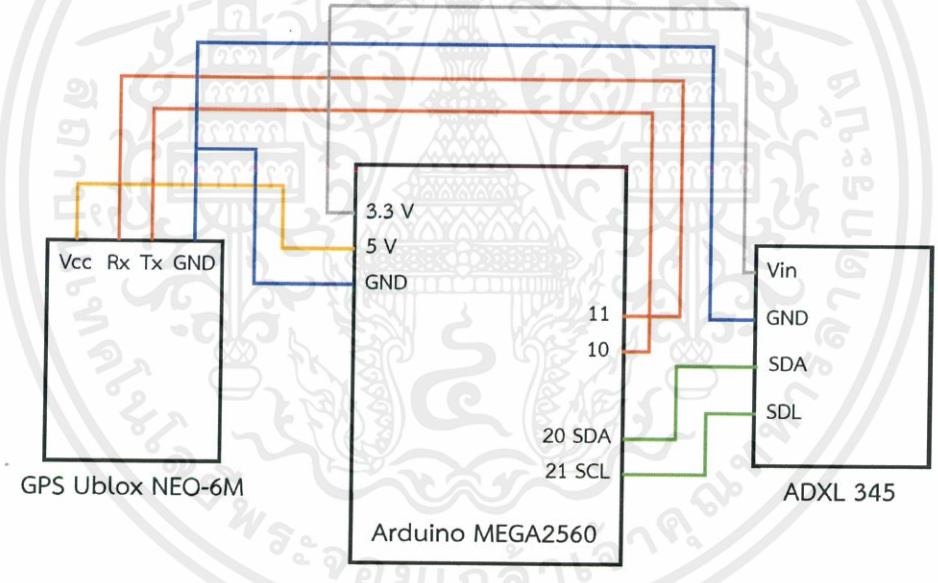
3.1.3 การออกแบบระบบเซ็นเซอร์

โครงสร้างของสถานีภาคพื้นดิน จำเป็นต้องมีระบบเซ็นเซอร์ เพื่อใช้วัดค่าต่าง ๆ ทั้งจากอุปกรณ์ของฝั่งสถานีส่งและอุปกรณ์ของฝั่งสถานีรับ โดยระบบเซ็นเซอร์ในโครงงานนี้ จะประกอบไปด้วย เซ็นเซอร์ตรวจจับตำแหน่ง (GPS Sensor) และเซ็นเซอร์วัดมุมก้ม-มุมเงย (Roll & Pitch Sensor) โดยสามารถแสดงบล็อกไดอะแกรมของระบบเซ็นเซอร์ได้ดังรูปที่ 3.19



รูปที่ 3.19 บล็อกไดอะแกรมของระบบเซ็นเซอร์

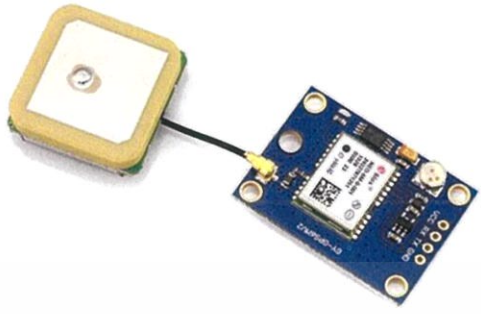
เมื่อทำการออกแบบระบบเซ็นเซอร์ทั้งหมด และรวมเซ็นเซอร์ทั้งหมดเข้าด้วยกันให้อยู่ภายในวงจรเพียงวงจรเดียว จะได้ Schemetic ของระบบเซ็นเซอร์ทั้งหมดดังรูปที่ 3.20



รูปที่ 3.20 วงจรของระบบเซ็นเซอร์ทั้งหมด

3.1.3.1 การออกแบบระบบเซ็นเซอร์ตรวจจับตำแหน่ง (GPS Sensor)

ระบบเซ็นเซอร์ตรวจจับตำแหน่งจะใช้โมดูล GPS Ublox NEO-6M ร่วมกับบอร์ด Arduino MEGA2560 ในการควบคุมการทำงานของโมดูล โดยจะเขียน Source code ของระบบผ่านโปรแกรม Arduino IDE และทำการอัปโหลดเข้าสู่บอร์ด

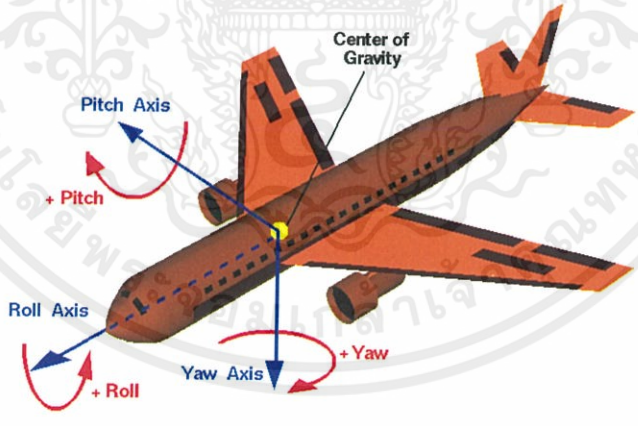


รูปที่ 3.21 โมดูล GPS Ublox NEO-6M [22]

โมดูล Ublox NEO-6M แสดงดังรูปที่ 3.21 จะสามารถรับสัญญาณ GPS ได้อย่างเดียว โดยในโครงงานนี้ จะใช้ GY-NEO6MV2 GPS module NEO6MV2 with Antenna ซึ่งจะรองรับเฉพาะไมโครคอนโทรลเลอร์แบบ UART เท่านั้น

3.1.3.2 การออกแบบระบบเซ็นเซอร์วัดมุมก้ม-มุมเงย (Roll & Pitch Sensor)

ระบบเซ็นเซอร์วัดมุมก้ม-มุมเงยจะใช้โมดูล GY-80 [23] ซึ่งมีความสามารถในการวัดค่าความเร่ง (Acceleration), ความเร็วเชิงมุม (Angular Speed), ระดับความสูง (Barometer), และสนามแม่เหล็กของโลก (Magnetometer) ได้ภายในโมดูลตัวเดียว



รูปที่ 3.22 การเคลื่อนที่ของอุปกรณ์ในมุมต่าง ๆ [23]

มุมต่าง ๆ ของอุปกรณ์แสดงดังรูปที่ 3.22 โดยเราจะนำค่าความเร่ง, ความเร็วเชิงมุม, ระดับความสูง, และสนามแม่เหล็กของโลกมาเพื่อหาทิศทางการเคลื่อนที่ และลักษณะการวางตัวของอุปกรณ์ เช่น Roll & Pitch เมื่ออุปกรณ์เอียงซ้ายขวา หรือคว่ำหงายอุปกรณ์

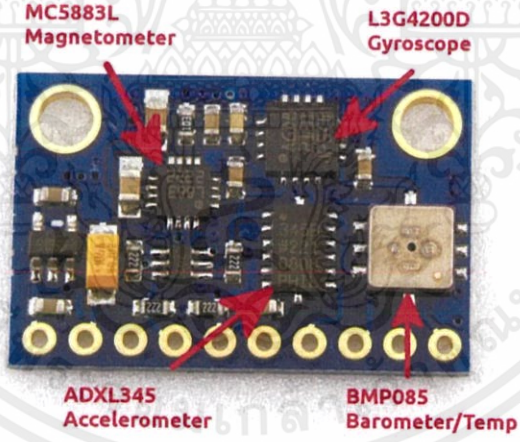
องค์ประกอบของโมดูล GY-80 สามารถแสดงได้ดังรูปที่ 3.23 ซึ่งประกอบไปด้วยอุปกรณ์ดังนี้

1) Accelerometer ADXL345 IC ทำหน้าที่วัดความเร่งออกมาให้ทั้ง 3 แกน คือ X, Y, Z

2) Gyroscope L3G4200D ทำหน้าที่วัดความเร็วเชิงมุม ทั้ง 3 แกน คือ X, Y, Z

3) BMP085 Barometer / Temp ทำหน้าที่วัดความสูงโดยการรับค่าความกดอากาศมาคำนวณ เนื่องจากการวัดความกดอากาศมาแปรความเป็นความสูงต้องมีการชดเชยทางอุณหภูมิด้วย จึงมีความสามาถในการวัดอุณหภูมิมาด้วยในตัว

4) MC5883L Magnetometer ทำหน้าที่วัดสนามแม่เหล็กของโลกเพื่อใช้ในการหาทิศทาง (ทิศเหนือใต้ออกตก) ดังนั้นจึงเรียกกันโดยทั่วไปว่า Digital Compass หรือ เข็มทิศดิจิตอล



รูปที่ 3.23 องค์ประกอบของโมดูล GY-80 [23]

ในการวัดค่ามุมก้ม-มุมเงยจะใช้ตัว Accelerometer ADXL345 IC โดยใช้กับบอร์ด Arduino MEGA2560 ต่อสายแบบ I2C โดยที่ขา A4 เชื่อมต่อกับ SDA และขา A5 เชื่อมต่อกับ SCL เราสามารถคำนวณหาค่ามุมเอียงได้จากความสัมพันธ์ของตรีโกณมิติ ดังนี้

$$Roll = \frac{a\tan 2(-fYg, fZg) \times 180.0}{M\_PI} \tag{3.11}$$

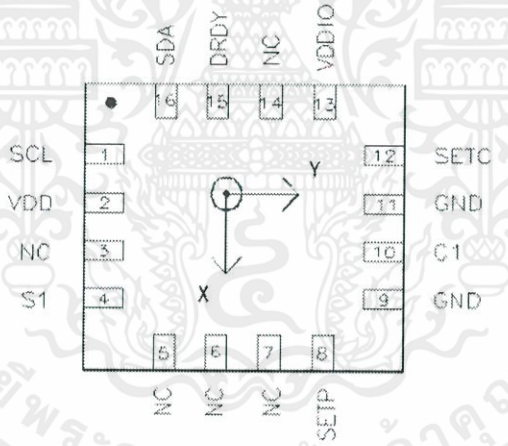
$$Pitch = \frac{a\tan 2(fXg, \sqrt{fYg \times fYg + fZg \times fZg}) \times 180.0}{M\_PI} \tag{3.12}$$

โดย Xg, Yg, Zg คือค่าความเร่งที่แกน X, Y, Z ตามลำดับ

ซึ่งสมการดังกล่าวจะถูกนำไปประมวลผลในโปรแกรม Arduino IDE จนได้ค่า fXg, fYg, fZg มาและนำมาคำนวณต่อเพื่อหาค่ามุมก้ม-มุมเงย หรือ Roll & Pitch

3.1.3.3 ระบบเซ็นเซอร์เข็มทิศ (Compass Sensor)

ระบบเซ็นเซอร์เข็มทิศ จะใช้อุปกรณ์ที่มีชื่อว่า 3-Axis Digital Compass IC HMC5883L [24] ซึ่งมีโครงสร้างของโมดูลดังรูปที่ 3.24 โดยระบบเซ็นเซอร์เข็มทิศนี้จะมีหน้าที่เป็นเครื่องมือช่วยบอกทิศทางแบบอิเล็กทรอนิกส์ แสดงค่ามุมเมื่อเทียบกับทิศเหนือเป็นตัวเลข ควบคุมการทำงานด้วยไมโครคอนโทรลเลอร์ ซึ่งจะให้ค่าทิศทางเมื่อเทียบกับทิศเหนือของโลก จึงสามารถนำไปใช้ในการพัฒนาเครื่องมือสำหรับบอก หรือกำหนดทิศทางได้ [25]



รูปที่ 3.24 โครงสร้างของโมดูล 3-Axis Digital Compass IC HMC5883L [25]

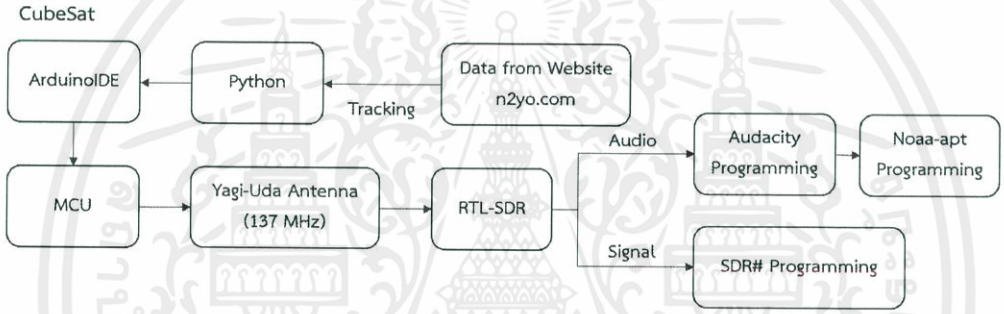
<u>รายละเอียดของโมดูล 3-Axis Digital Compass IC HMC5883L</u>

- รุ่น Gy-271
- ใช้ Chip รุ่น HMC5883L ในการวัดมุมเอียง
- รองรับแหล่งจ่ายไฟ Volt : DC 3.3 - 5V

- เชื่อมต่อไปยัง Arduino หรือ ไมโครคอนโทรลเลอร์ชนิดๆ ผ่าน I2C Interface
- ช่วงระยะการวัด ± 1.3-8 gauss

3.1.4 การออกแบบโปรแกรมสำหรับดึงข้อมูลการเคลื่อนที่ของ CubeSat

ระบบสายอากาศสำหรับติดตามแบบจำลองดาวเทียมขนาดเล็ก (CubeSat) เป็นระบบติดตามที่จะใช้มอเตอร์ควบคุมสายอากาศ ให้หมุนไปในทิศทางที่ต้องการ โดยทิศทางที่กำหนดขึ้นนั้นมาจากมุม Azimuth และมุม Elevation ซึ่งผ่านการทำงานร่วมกันของโปรแกรม Arduino และ Python โดยมีบล็อกไดอะแกรมดังรูปที่ 3.25



รูปที่ 3.25 บล็อกไดอะแกรมในส่วนของ CubeSat

3.1.4.1 การเลือก CubeSat ที่เหมาะสมต่อการรับสัญญาณ

CubeSat ที่โคจรอยู่รอบโลก จะอยู่ใน Low Earth Orbit (LEO) ที่ความสูงประมาณ 400 – 600 กิโลเมตร และมีความถี่แตกต่างกันไป การเลือก CubeSat จะต้องคำนึงถึง 2 ปัจจัย คือ การโคจรของ CubeSat และความถี่ของ CubeSat กล่าวคือ CubeSat ที่เลือกต้องมีช่วงเวลาที่โคจรผ่านน่านฟ้าของประเทศไทยและต้องมีความถี่ Downlink ตรงกับความถี่ของสายอากาศที่เลือกใช้ โดย CubeSat ที่เลือก มีทั้งหมด 3 ดวง คือ

1) NOAA 15 มีรายละเอียดของดาวเทียมดังนี้

<u>รายละเอียดของดาวเทียม NOAA 15</u>

- NORAD ID: 25338
- Int'l Code: 1998-030A

- Perigee: 804.4 km
- Apogee: 820.8 km
- Inclination: 98.8 °
- Period: 101.0 minutes
- Semi major axis: 7183 km
- RCS: 5.5361 m2 (large)
- Launch date: May 13, 1998
- Source: United States (US)
- Launch site: AIR FORCE WESTERN TEST RANGE (AFWTR)
- Downlink Frequency: 137.620 MHz

2) NOAA 18 มีรายละเอียดของดาวเทียมดังนี้

<u>รายละเอียดของดาวเทียม NOAA 18</u>

- NORAD ID: 28654
- Int'l Code: 2005-018A
- Perigee: 847.4 km
- Apogee: 869.1 km
- Inclination: 99.1 °
- Period: 102.0 minutes
- Semi major axis: 7229 km
- RCS: 5.2629 m2 (large)
- Launch date: May 20, 2005
- Source: United States (US)
- Launch site: AIR FORCE WESTERN TEST RANGE (AFWTR)
- Downlink Frequency: 137.9125 MHz / 1707.000 MHz

Visible passes | AM/PM time | UTC | Print as PDF

Start		Max altitude			End		All passes	
Date, Local time	Az	Local time	Az	El	Local time	Az	Mag	Info
26-Mar 20:08	SSE 155°	20:15	ENE 75°	50°	20:23	N 357°	+6.3	Map and details
26-Mar 21:50	SW 211°	21:56	W 260°	11°	22:02	NW 310°	+7.8	Map and details
27-Mar 08:44	N 14°	08:52	E 87°	81°	09:00	S 191°	+5.2	Map and details
27-Mar 19:56	SE 148°	20:04	ENE 74°	37°	20:11	N 2°	+6.5	Map and details
27-Mar 21:38	SSW 203°	21:44	W 260°	16°	21:51	NW 317°	+7.5	Map and details
28-Mar 08:33	NNE 19°	08:40	E 99°	60°	08:48	S 185°	+5.4	Map and details
28-Mar 19:45	SE 142°	19:52	ENE 74°	27°	19:59	N 7°	+6.7	Map and details
28-Mar 21:26	SSW 196°	21:33	W 260°	22°	21:39	NW 323°	+7.3	Map and details
29-Mar 08:21	NNE 24°	08:29	E 101°	44°	08:36	S 179°	+5.8	Map and details
29-Mar 10:03	NNW 341°	10:09	WNW 288°	14°	10:15	SW 235°	+7.2	Map and details
29-Mar 19:34	SE 135°	19:40	ENE 74°	20°	19:47	N 12°	+7.0	Map and details
29-Mar 21:14	S 189°	21:21	W 258°	30°	21:28	NW 329°	+7.0	Map and details
30-Mar 08:10	NNE 29°	08:17	E 101°	32°	08:24	S 172°	+6.3	Map and details
30-Mar 09:51	N 347°	09:57	WNW 287°	19°	10:04	SW 227°	+6.9	Map and details
30-Mar 19:23	SE 127°	19:29	ENE 71°	15°	19:35	NNE 18°	+7.2	Map and details
30-Mar 21:02	S 183°	21:09	W 258°	40°	21:16	NNW 334°	+6.7	Map and details
31-Mar 07:58	NE 34°	08:05	E 101°	24°	08:12	S 165°	+6.7	Map and details
31-Mar 09:39	N 353°	09:46	WNW 287°	26°	09:53	SW 220°	+6.6	Map and details
31-Mar 19:12	ESE 119°	19:17	ENE 71°	10°	19:23	NNE 24°	+7.5	Map and details
31-Mar 20:50	S 176°	20:57	W 258°	55°	21:05	NNW 340°	+6.5	Map and details
1-Apr 07:47	NE 40°	07:53	E 99°	18°	08:00	SSE 158°	+7.0	Map and details
1-Apr 09:27	N 358°	09:34	WNW 287°	35°	09:41	SW 213°	+6.2	Map and details
1-Apr 20:38	S 170°	20:46	W 260°	76°	20:53	NNW 345°	+6.2	Map and details
2-Apr 07:36	NE 47°	07:42	E 99°	13°	07:48	SSE 151°	+7.3	Map and details
2-Apr 09:15	N 2°	09:22	WNW 288°	47°	09:30	SSW 207°	+5.7	Map and details
2-Apr 20:26	SSE 164°	20:34	ENE 69°	80°	20:42	N 349°	+6.1	Map and details
3-Apr 09:03	N 7°	09:11	W 279°	64°	09:18	SSW 201°	+5.4	Map and details
3-Apr 20:15	SSE 158°	20:22	ENE 73°	59°	20:30	N 354°	+6.1	Map and details
4-Apr 08:51	N 12°	08:59	NW 318°	86°	09:07	S 194°	+5.2	Map and details
4-Apr 20:03	SSE 152°	20:11	ENE 74°	43°	20:18	N 359°	+6.2	Map and details
4-Apr 21:45	SSW 207°	21:51	W 261°	13°	21:57	NW 314°	+7.6	Map and details

Legend: Not visible | Marginal | Good | Excellent

รูปที่ 3.26 เวลาที่ ISS จะผ่าน ในช่วงเวลา 10 วัน ตั้งแต่ 26 มีนาคม 2562 [26]

3) NOAA 19 มีรายละเอียดของดาวเทียมดังนี้

<u>รายละเอียดของดาวเทียม NOAA 19</u>

- NORAD ID: 33591
- Int'l Code: 2009-005A
- Perigee: 849.1 km
- Apogee: 867.9 km
- Inclination: 99.2 °
- Period: 102.0 minutes
- Semi major axis: 7229 km
- RCS: 5.346 m2 (large)
- Launch date: February 6, 2009
- Source: United States (US)
- Launch site: AIR FORCE WESTERN TEST RANGE (AFWTR)
- Downlink Frequency: 137.100 MHz / 1698.000 MHz

3.1.4.2 การเลือกช่วงเวลาที่เหมาะสมต่อการรับสัญญาณของ CubeSat

เนื่องจากดาวเทียม CubeSat โคจรรอบโลก จึงอาจมีเพียงบางช่วงเวลาเท่านั้น ที่โคจรผ่านน่านฟ้าประเทศไทย โดยที่เราสามารถตรวจสอบเวลาที่ CubeSat แต่ละดวงจะผ่านได้ในเว็บไซต์ N2YO.com ดังรูปที่ 3.26

3.1.4.3 การออกแบบโปรแกรม Python

ในส่วนของโปรแกรม Python เราจะใช้เพื่อดึงข้อมูลจากเว็บไซต์ N2YO.com ผ่านระบบ API แล้วนำข้อมูลที่ได้ไปแยกให้เหลือเพียงข้อมูลที่เราต้องการ หลังจากนั้นนำข้อมูลนั้นส่งไปให้ โปรแกรม Arduino เพื่อนำไปใช้ในการเขียนโค้ดคุมการหมุนมอเตอร์ โดยมีขั้นตอนในการออกแบบโปรแกรม Python ดังนี้

1) กำหนดตัวแปร โดยตัวแปรมี 2 ประเภท คือ ตัวแปรที่ใช้ในการวนลูป และตัวแปรที่ใช้ในการเก็บค่าต่าง ๆ

2) ใช้คำสั่ง request และ get เพื่อดึงข้อมูลจากเว็บ n2yo.com (โดยปกติการดึงข้อมูลจากเว็บนั้นค่อนข้างยุ่งยาก แต่เว็บ n2yo.com มี ระบบ API เพียงแค่สมัครสมาชิกของเว็บ ก็สามารถเข้าถึงข้อมูลของ n2yo.com ได้ โดยใช้คำสั่ง request กับ get)

3) ใช้ while เพื่อวนลูปในการดึงข้อมูลจากเว็บแบบต่อเนื่อง

4) เมื่อได้ข้อมูลของดาวเทียม จะพบว่าข้อมูลของดาวเทียมนั้นเป็นข้อความยาวที่ยังใช้ประโยชน์ไม่ได้ จึงต้องเขียนคำสั่งให้มันแยกข้อมูลเป็นส่วน ๆ

5) การแบ่งข้อมูลจะใช้ตัว " เป็นตัวแบ่ง โดยข้อมูลที่ได้ 2 ประเภทคือ ชื่อดาวเทียม และข้อมูลต่าง ๆ ที่เหลือ เช่น Latitude, Longitude, Azimuth, Elevation เป็นต้น โดยชื่อดาวเทียมจะอยู่ใน " " แต่ข้อมูลอื่นจะตามหลัง ":

6) นับจำนวน " โดยใช้ while เก็บไว้ในตัวแปร ได้ 44 ตัว หรือ 22 คู่

7) ใช้ลูป 2 ลูปแยกข้อมูลต่าง ๆ โดยลูปที่ 1 แยกชื่อดาวเทียมอย่างเดียว ลูปที่2 แยกข้อมูลอย่างอื่น

8) เมื่อทำการแยกข้อมูลแล้ว จะได้ข้อมูลที่ต้องการ แต่ยังอยู่ในตัวแปรเดียวกัน เราจึงต้องแยกข้อมูลอีกครั้งโดยใช้ while

3.1.4.4 การรับข้อมูลและแสดงผลสัญญาณที่สามารถรับได้

ในการรับสัญญาณจะใช้ SDR เพื่อลดความถี่ให้สามารถใช้งานร่วมกับคอมได้ และจะใช้โปรแกรม SDR sharp เพื่อแสดงผลสัญญาณที่ได้รับ โดยเราสามารถนำสัญญาณที่ได้นั้นไปถอดรหัสเพื่อทำการตรวจสอบว่าสัญญาณชุดนี้ คืออะไรได้ อย่างไรก็ตามในส่วนการถอดรหัสสัญญาณนั้นยังอยู่ในขั้นตอนการศึกษาต่อไป

3.1.5 การออกแบบโปรแกรมสำหรับแสดงผลข้อมูลที่รับได้จากดาวเทียม

3.1.5.1 รูปแบบข้อมูลตามข้อบังคับของการแข่งขัน

รูปแบบของข้อมูลที่ทำการรับจากแบบจำลองดาวเทียมขนาดเล็ก จะมีรูปแบบอ้างอิงตามข้อกำหนดเป็นรูปแบบของการแข่งขัน CanSat Competition ประจำปี 2019 ซึ่งรูปแบบของข้อมูลนั้นจะแสดงดังรูปที่ 3.27

<TEAM ID>,<MISSION TIME>,<PACKET COUNT>,<ALTITUDE>, <PRESSURE>, <TEMP>,<VOLTAGE>,<GPS TIME>,<GPS LATITUDE>,<GPS LONGITUDE>,<GPS ALTITUDE>,<GPS SATS>,<PITCH>,<ROLL>,<BLADE SPIN RATE>,<SOFTWARE STATE>,<BONUS DIRECTION>

รูปที่ 3.27 รูปแบบของข้อมูลที่ทำการรับจากแบบจำลองดาวเทียมขนาดเล็ก

3.1.5.2 ขั้นตอนการแยกข้อมูล

ในการออกแบบโปรแกรมจะมีหลักการในการทำงาน คือรับข้อมูลจากฝั่งส่งที่มีการแยกค่าข้อมูลของแต่ละตัวด้วยตัวคั่นที่ได้ตั้งไว้ จากนั้นทำการแยกข้อมูลแต่ละตัวไปทำการพล็อตออกเป็นกราฟของแต่ละค่า นำค่าละติจูด, ลองจิจูด ไปทำการแสดงตำแหน่งในหน้าจอแสดงตำแหน่ง GPS โดยจะแสดงหลักการทำงานเป็นโฟล์วชาร์ต แสดงดังรูปที่ 3.28

3.1.5.3 การออกแบบหน้าต่างแสดงผลในโปรแกรม LabVIEW

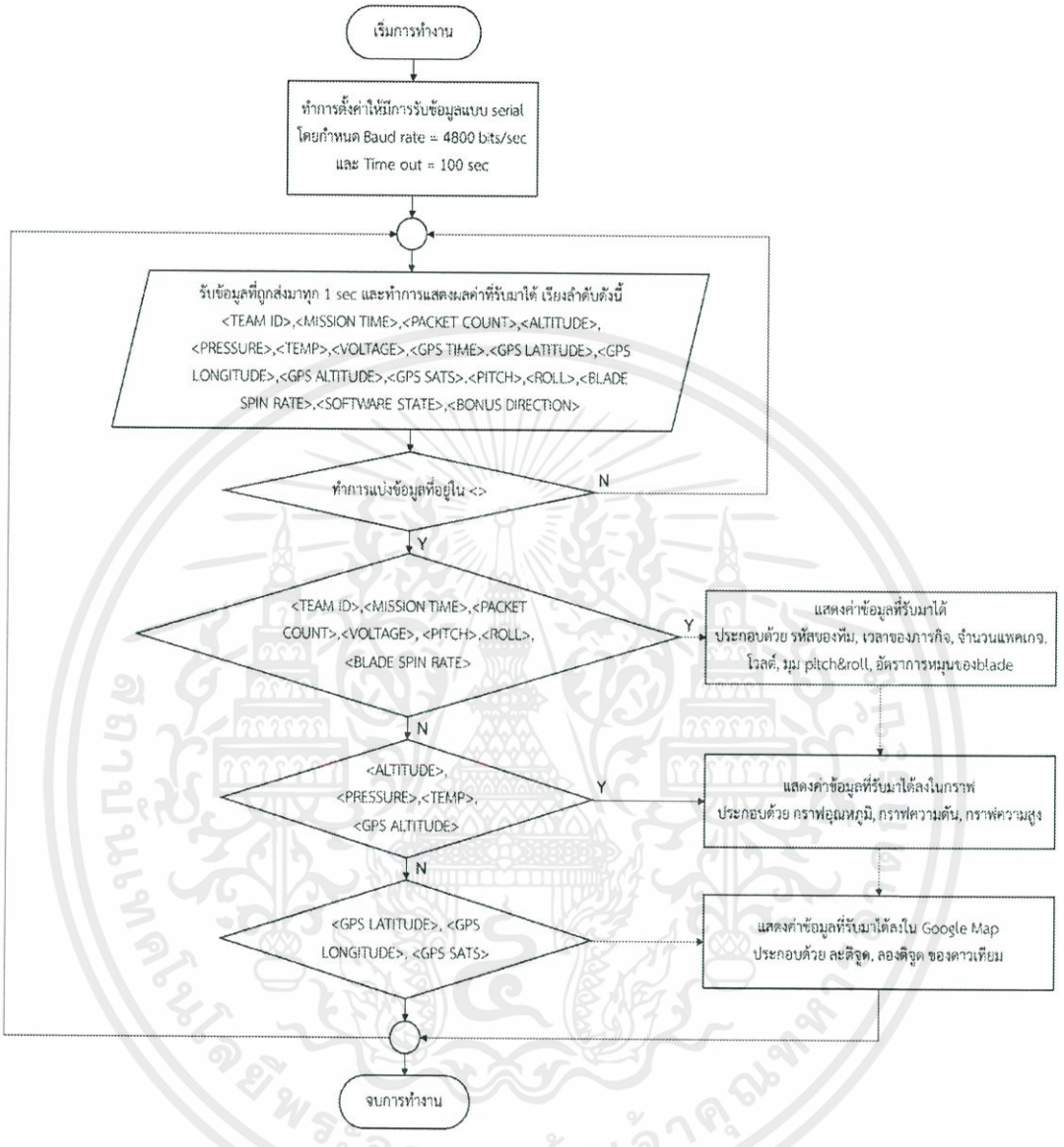
เขียนโปรแกรม LabVIEW เพื่อออกแบบหน้าต่างแสดงผล ให้สามารถรับข้อมูลจากแบบจำลองดาวเทียมขนาดเล็ก แล้วนำมาแสดงผลในหน้าจอแสดงผลของโปรแกรมที่ใช้หลักการของโฟล์วชาร์ตในรูปที่ 3.28 ในการแยกข้อมูลแต่ละส่วนไปแสดงผล โดยหน้าจอที่ทำการออกแบบเสร็จแล้วจะเป็นไปตามรูปที่ 3.29 และรูปที่ 3.30

โดยหน้าจอโปรแกรมในส่วนของ Front Panel ที่แสดงในรูปที่ 3.28 จะประกอบไปด้วย

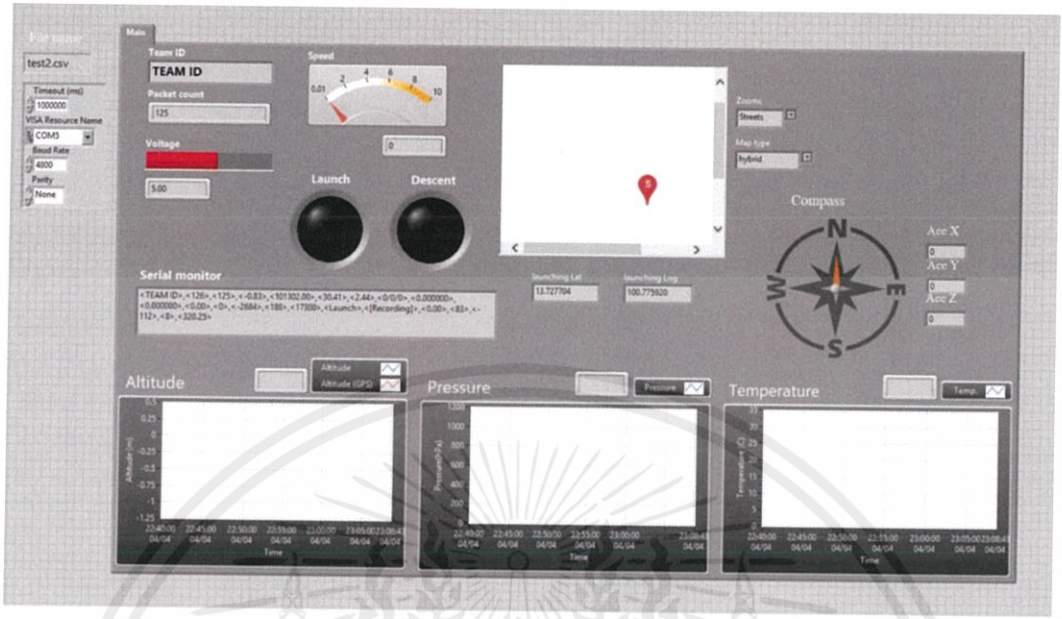
1) กล่องข้อความ ได้แก่ Team ID (หมายเลขลำดับของทีม), Packet Count (เวลาตั้งแต่แบบจำลองดาวเทียมขนาดเล็กเริ่มทำงาน), Serial Monitor (รูปแบบของข้อมูลที่รับได้จากแบบจำลองดาวเทียมขนาดเล็ก)

2) ไฟแสดงผล LED แสดงสถานะของแบบจำลองดาวเทียมขนาดเล็ก โดยมีอยู่ 2 สถานะ คือ Launch และ Descent

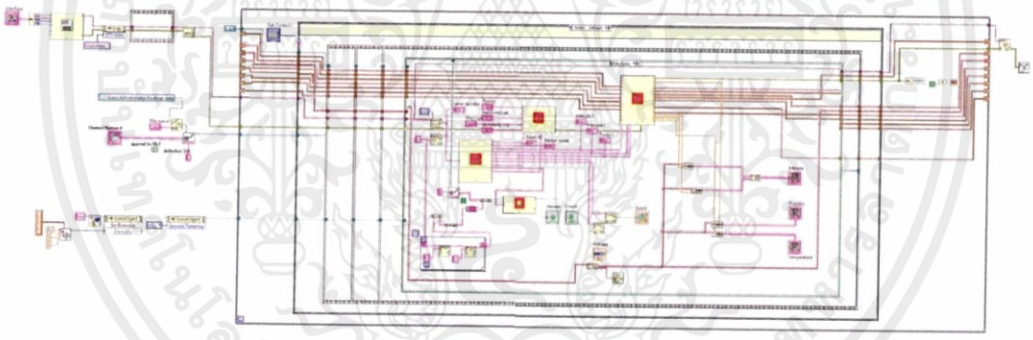
3) กราฟแสดงผล ได้แก่ Altitude Graph (กราฟความสูง), Pressure Graph (กราฟความดัน), Temperature Graph (กราฟอุณหภูมิ)



รูปที่ 3.28 โฟล์วชาร์ตการแยกข้อมูล



รูปที่ 3.29 หน้าจอโปรแกรม LabVIEW ส่วน Front Panel



รูปที่ 3.30 หน้าจอโปรแกรม LabVIEW ส่วน Block Diagram

4) หน้าจอแสดงผล GPS แสดงพิกัดของแบบจำลองดาวเทียมขนาดเล็ก โดยจะอยู่ในรูปแบบ GOOGLE MAP

5) แถบ Meter แสดงผล ได้แก่ Voltage (ค่าแรงดันที่เหลือของถ่านในแบบจำลองดาวเทียมขนาดเล็ก), Speed (ความเร็วของแบบจำลองดาวเทียมขนาดเล็ก)

3.2 เครื่องมือที่ใช้ในการทดลอง

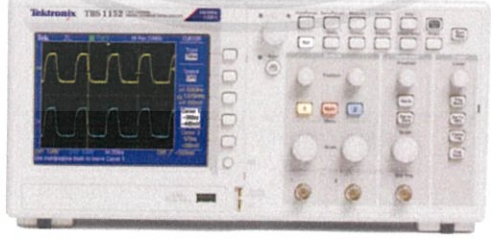
3.2.1 Spectrum Analyzer

สเปคตรัมอนาไลเซอร์ หรือ Spectrum Analyzer [27] เป็นอุปกรณ์ที่มีความสามารถในการวัดสัญญาณในย่านความถี่สูง เช่นความถี่จากเสาอากาศมักใช้ในงานสื่อสารแบบไร้สาย สเปคตรัมอนาไลเซอร์สามารถวัดและแสดงค่าในรูปแบบความถี่ (Frequency domain) และแอมพลิจูดของสัญญาณได้โดยใช้ในการวัดค่าคุณสมบัติที่จำเป็นของสายอากาศเพื่อดูประสิทธิภาพมีลักษณะแสดงในรูปที่ 3.31

รูปที่ 3.31 Spectrum Analyzer [27]

3.2.2 Oscilloscope

ออสซิลโลสโคป [28] หรือ CRO โดยย่อมาจาก Cathode Ray Oscilloscope เป็นเครื่องมือที่ใช้ในการวัดสัญญาณไฟฟ้า แล้วแสดงผลออกมาในรูปของกราฟที่เป็นค่าของแรงดันไฟฟ้าของสัญญาณนั้น ๆ อีกทั้งเรายังสามารถวัดความถี่, เฟส, คาบเวลา และแรงดันของสัญญาณได้อีกด้วย โดยในปัจจุบันแบ่งออสซิลโลสโคปออกเป็น 2 ประเภท คือ Analog Oscilloscope และ Digital Oscilloscope โดยมีลักษณะดังรูปที่ 3.32



รูปที่ 3.32 Oscilloscope [28]

3.2.3 โมดูล GPS Ublox NEO-6M

GPS Ublox NEO-6M Module เป็นโมดูลสำหรับระบุตำแหน่งต่าง ๆ โดยในที่นี้ เราสามารถรับค่าเวลา, ละติจูด, ลองติจูด และวันที่ โดยมีข้อมูลทั่วไปดังตารางที่ 3.6

ตารางที่ 3.6 ข้อมูลทั่วไปของโมดูล GPS Ublox NEO-6M

ข้อมูลทั่วไป	รายละเอียด
โมดูลจาก UBlox	รุ่น NEO-6M
ความถี่ในการอัพเดตตำแหน่ง	1 – 5 Hz
แรงดันไฟเลี้ยง	3.3 - 5 V
Baud rate	9600
Mounting Hole	3 mm
Module Size	23 mm * 30 mm
Antenna Size	12 * 12 mm
Cable	20 mm

3.2.4 โมดูล Arduino MEGA2560 3.2 Inch 320x480 TFT IPS LCD Display

เป็นหน้าจอ LCD ที่ความละเอียดอยู่ที่ 320x480 และมีทั้งหมด 262K สี โดยอุปกรณ์มีลักษณะดังรูปที่ 3.33 และมีคุณสมบัติดังตารางที่ 3.7

รูปที่ 3.33 โมดูล Arduino MEGA2560 3.2 Inch 320x480 TFT IPS LCD Display [29]

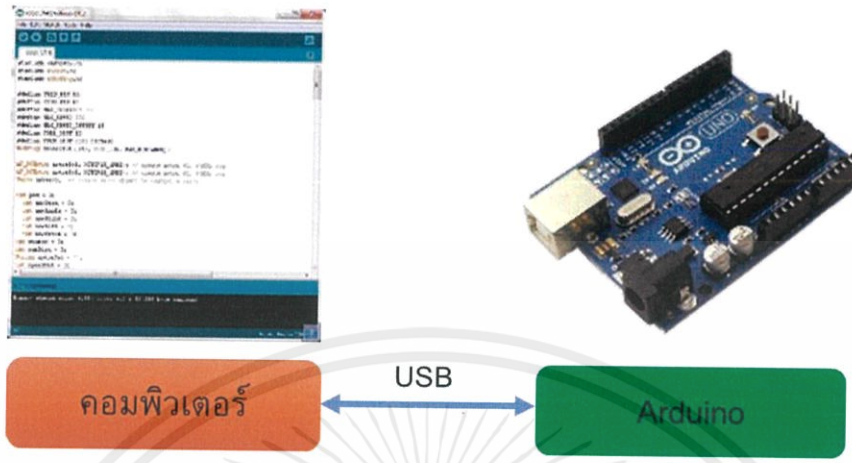
ตารางที่ 3.7 ข้อมูลทั่วไปของโมดูล Arduino MEGA2560 3.2 Inch 320x480 TFT IPS LCD Display [30]

ข้อมูลทั่วไป	รายละเอียด
Display Type:	3.2-inch a-si TFT LCD Module
Glass Type:	TFT IPS(Full-Angle)
Display Resolution:	480 X RGB X 320 Pixels
Back light:	6 chip HighLight white LEDs
Control IC:	HX8357B
Interface:	16Bit parallel interface
PCB Module size:	89.92mm X 54.25mm
LCD Area (WxHxT):	50.74mm X 78.35mm X 1.88mm
Active Area (WxH):	67.68mm X 45.12mm

3.2.5 Arduino

Arduino อ่านว่า (อาดูอิโน่ หรือ อาดุยโน่) [31] เป็นบอร์ดไมโครคอนโทรลเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัวบอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย เหมาะสำหรับผู้เริ่มต้นศึกษา ผู้ใช้งานสามารถดัดแปลง เพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ด หรือโปรแกรมต่อได้

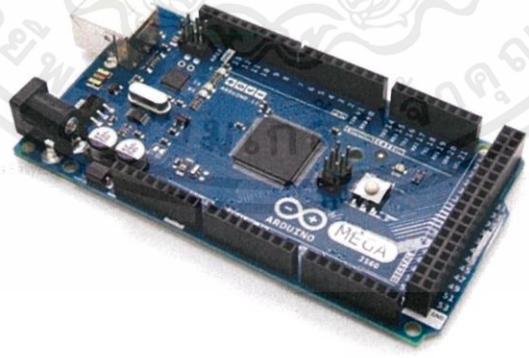
โดยรูปแบบการเขียนโปรแกรมบน Arduino สามารถแสดงได้ดังรูปที่ 3.34 ซึ่งจะเขียนโปรแกรมบนคอมพิวเตอร์ ผ่านทางโปรแกรม ArduinoIDE หลังจากที่เขียน Source code โปรแกรมเรียบร้อยแล้ว ให้ผู้ใช้งานเลือกรุ่นบอร์ด Arduino ที่ใช้และหมายเลข Com port กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม จากนั้นกดปุ่ม Upload โค้ดโปรแกรมไปยังบอร์ด Arduino ผ่านทางสาย USB เมื่ออัปโหลดเรียบร้อยแล้ว จะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที



รูปที่ 3.34 รูปแบบการเขียนโปรแกรมบน Arduino [31]

3.2.5.1 Arduino MEGA2560

Arduino Mega 2560 คือบอร์ดไมโครคอนโทรลเลอร์ที่พัฒนาจาก ATmega2560 มี 54 digital input/output โดยมี 14 ขา สามารถใช้เป็น output แบบ PWM ได้ มี analog inputs 16 ขา มี UARTs (hardware serial ports) 4 ขา ทำงานที่ความถี่ 16 MHz สามารถเชื่อมต่อกับคอมพิวเตอร์ด้วยสายเคเบิล USB หรือใช้ adaptor AC-to-DC เพื่อเริ่มต้นใช้งาน และมีปุ่ม reset สามารถต่อเข้ากับ shields ที่ออกแบบเพื่อใช้งานกับ Arduino Duemilanove หรือ Diecimila มีลักษณะดังรูปที่ 3.35 มีข้อมูลทั่วไปของ Arduino MEGA2560 ดังตารางที่ 3.8

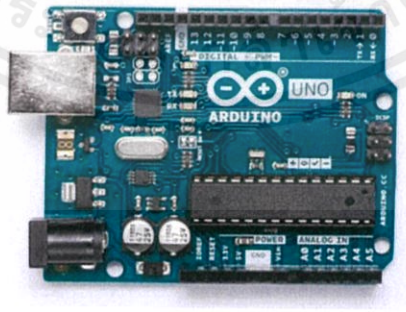


รูปที่ 3.35 Arduino MEGA2560 [32]

ตารางที่ 3.8 ข้อมูลทั่วไปของ Arduino MEGA2560

ข้อมูลทั่วไป	รายละเอียด
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

3.2.5.2 Arduino UNO R3



รูปที่ 3.36 Arduino UNO R3 [33]

Arduino Uno R3 เป็นบอร์ดไมโครคอนโทรลเลอร์แบบ Open-source ออกแบบมาให้ใช้งานได้ง่าย ใช้ชิพ ATmega328 รันที่ความถี่ 16 MHz มี Digital Input / Output 14 ขา (เป็น PWM ได้ 6 ขา) มี Analog Input 6 ขา Serial UART 1 ชุด I2C 1 ชุด SPI 1 ชุด เขียนโปรแกรมบนซอฟท์แวร์ Arduino IDE และโปรแกรมผ่านพอร์ต USB มีลักษณะดังรูปที่ 3.36 มีข้อมูลทั่วไปของ Arduino UNO R3 ดังตารางที่ 3.9

ตารางที่ 3.9 ข้อมูลทั่วไปของ Arduino UNO R3

ข้อมูลทั่วไป	รายละเอียด
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

3.2.6 XBEE Pro Series 1

รูปที่ 3.37 XBEE Pro Series 1 [34]

XBEE Pro 60mW U.FL Connection (XBEE Series 1) เป็นโมดูล รับส่งสัญญาณไร้สาย ย่านความถี่ 2.4 GHz ซึ่งเป็นย่านความถี่วิทยุ สมัครเล่น ตามมาตรฐานโปรโตคอล 802.15.4 ใช้พลังงานต่ำ จุดต่อสายอากาศแบบ U.FL. Connection มีลักษณะดังรูปที่ 3.37

<u>รายละเอียดของโมดูล XBEE Pro 60mW U.FL Connection</u>

- 3.3V @ 215mA
- 250kbps Max data rate
- 60mW output (+18dBm)
- 1-mile (1500m) range (Line-of-Sight) โดยระยะที่ทำได้จะขึ้นอยู่กับ สภาพแวดล้อมของระบบ และ สายอากาศที่ใช้ เนื่องจากความถี่ 2.4 GHz เป็นย่านความถี่สูง อัตราการลดทอนสัญญาณจะสูง และ สิ่งกีดขวางจะมีผลอย่างมากกับระยะทางที่ใช้งานได้
- Fully FCC certified
- 6 10-bit ADC input pins
- 8 digital IO pins
- 128-bit encryption
- Local or over-air configuration
- AT or API command set

3.2.7 XBEE USB Dongle

XBEE USB Dongle ทำหน้าที่เป็นตัวเชื่อมต่อระหว่าง XBEE กับ พอร์ตของคอมพิวเตอร์ โดยจะใช้อุปกรณ์ ICSH027A XBEE Mini USB Adaptors ซึ่งมีลักษณะดังรูปที่ 3.38

รูปที่ 3.38 ICSH027A XBEE Mini USB Adaptors [35]

3.2.8 Arcade Joystick

รูปที่ 3.39 Arcade Joystick [36]

Arcade Joystick มีลักษณะดังรูปที่ 3.39 ใช้ 4 microswitches to ตรวจจับตำแหน่ง on/off และควบคุมการเคลื่อนที่ทั้ง 4 ทิศ

<u>รายละเอียดของโมดูล VODOOL8 Classic Arcade Joystick [36]</u>

- วัสดุเป็น ABS และโลหะผสม
- เส้นผ่าศูนย์กลางลูก approx.33mm/1.29"
- มีสีแดง
- ขนาดแผ่นประมาณ 95*60*30 มม./3.74*2.36*1.18"
- น้ำหนักสุทธิประมาณ 206g

3.3 การจัดเก็บผลการทดลอง

3.3.1 การทดสอบการทำงานของระบบเซ็นเซอร์

ทดสอบว่าเซ็นเซอร์ของระบบสามารถใช้งานได้จริงตามความต้องการหรือไม่ ทั้งเซ็นเซอร์ระบุตำแหน่ง และเซ็นเซอร์วัดมุมก้ม-มุมเงย

3.3.2 การทดสอบการรับ-ส่งข้อมูลมายังสถานีภาคพื้นดิน

ทดสอบว่าสามารถรับและส่งข้อมูลผ่าน XBEE Pro series1 และนำค่าที่รับได้จาก Payload มาแสดงผลที่เครื่องคอมพิวเตอร์ที่สถานีภาคพื้นดินได้หรือไม่

3.3.3 การทดสอบการดึงข้อมูลการเคลื่อนที่ของ CubeSat

ทดสอบว่าการใช้ภาษา Python ร่วมกับโปรแกรม PyCharm สามารถดึงข้อมูลการเคลื่อนที่ของ CubeSat แบบ Real-Time จากเว็บไซต์ n2yo.com ได้หรือไม่

บทที่ 4

ผลการทดลอง

4.1 ผลการทดสอบการทำงานของระบบเซ็นเซอร์

4.1.1 การทดสอบการทำงานของระบบเซ็นเซอร์ตรวจจับตำแหน่ง

ทางผู้จัดทำได้ทำการทดสอบการทำงานของเซ็นเซอร์ตรวจจับตำแหน่ง โดยการขับรถไปรอบ ๆ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และเก็บผลที่ได้เป็นจำนวน 36 ค่า จะได้ค่าที่แสดงออกมาทางโปรแกรม ArduinoIDE ดังตารางที่ 4.1

ตารางที่ 4.1 ผลการทดสอบการทำงานเซ็นเซอร์ตรวจจับตำแหน่งจากโปรแกรม ArduinoIDE

ลำดับที่	เวลา	ละติจูด	ลองติจูด
1	Time: 090740.00	13.727304	100.774022
2	Time: 090750.00	13.727191	100.773567
3	Time: 090800.00	13.727128	100.773164
4	Time: 090810.00	13.727113	100.772800
5	Time: 090820.00	13.727102	100.772443
6	Time: 090830.00	13.727151	100.772106
7	Time: 090840.00	13.727088	100.771847
8	Time: 090850.00	13.726882	100.771880
9	Time: 090900.00	13.726685	100.771881
10	Time: 090910.00	13.726107	100.771909
11	Time: 090920.00	13.725752	100.771918
12	Time: 090930.00	13.725387	100.771885
13	Time: 090940.00	13.725134	100.771853
14	Time: 090950.00	13.724888	100.771887

ตารางที่ 4.1 ผลการทดสอบการทำงานเซ็นเซอร์ตรวจจับตำแหน่งจากโปรแกรม ArduinoIDE (ต่อ)

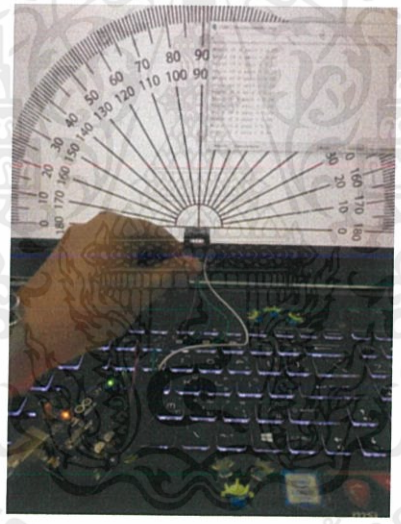
ลำดับที่	เวลา	ละติจูด	ลองติจูด
15	Time: 091000.00	13.724791	100.772189
16	Time: 091010.00	13.724756	100.772546
17	Time: 091020.00	13.724717	100.772880
18	Time: 091030.00	13.724693	100.773093
19	Time: 091040.00	13.724658	100.773536
20	Time: 091050.00	13.724677	100.773813
21	Time: 091100.00	13.724648	100.774376
22	Time: 091110.00	13.724684	100.774605
23	Time: 091120.00	13.724697	100.774881
24	Time: 091130.00	13.724682	100.775272
25	Time: 091140.00	13.724794	100.775458
26	Time: 091150.00	13.72503	100.775419
27	Time: 091200.00	13.725305	100.775227
28	Time: 091210.00	13.725623	100.775347
29	Time: 091220.00	13.725959	100.775366
30	Time: 091230.00	13.726125	100.775316
31	Time: 091240.00	13.726185	100.775109
32	Time: 091250.00	13.726447	100.774799
33	Time: 091300.00	13.726929	100.774701
34	Time: 091310.00	13.727466	100.774570
35	Time: 091320.00	13.727637	100.774652
36	Time: 091330.00	13.727520	100.775037

และเมื่อนำค่าละติจูดและลองติจูดที่ได้ มาพล็อตใน GOOGLE MAP จะได้เส้นทางเป็นวงปิด ซึ่งใกล้เคียงกับเส้นทางที่ขับรถจริง ๆ ดังรูปที่ 4.1 จึงสรุปได้ว่า เซ็นเซอร์ตรวจจับตำแหน่งมีความแม่นยำในการใช้งานจริง

รูปที่ 4.1 เส้นทางใช้ในการทดสอบการทำงานเซ็นเซอร์ตรวจจับตำแหน่งจาก GOOGLE MAP [36]

จากการทดสอบการทำงานของระบบเซ็นเซอร์ตรวจจับตำแหน่ง ซึ่งใช้โมดูล GPS Ublox NEO-6M เมื่อทำการวัดค่าสัญญาณบิตข้อมูลที่รับได้จาก Oscilloscope จะสามารถถอดรหัสข้อมูลได้ ดังรูปที่ 4.2 ค่าบิตข้อมูลที่รับได้จากโมดูล GPS Ublox NEO-6M จะอยู่ในรูปแบบการสื่อสารแบบ UART ข้อมูลที่ส่งจะอยู่ในรูปแบบรหัส ASCII 8 บิต มี Start bit และ Stop bit โดยในรูปนี้เมื่อถอดรหัสออกมาจะได้ว่า $GPRMC

รูปที่ 4.2 ค่าสัญญาณบิตข้อมูลจากโมดูล GPS Ublox NEO-6M



รูปที่ 4.3 การทดสอบการทำงานของระบบเซ็นเซอร์วัดมุมก้ม-มุมเงยในแนวแกน x ที่ 0 องศา

4.1.2 การทดสอบการทำงานของระบบเซ็นเซอร์วัดมุมก้ม-มุมเงย

ทางผู้จัดทำได้ทำการทดสอบการทำงานของเซ็นเซอร์วัดมุมก้ม-มุมเงย โดยนำอุปกรณ์เอียงไปที่มุม 0 องศา, 30 องศา, 60 องศา, 90 องศา, 120 องศา, 150 องศา และ 180 องศา ดังที่แสดงตัวอย่างการทดสอบในรูปที่ 4.3 และ 4.5 ซึ่งค่ามุมที่ได้จะแสดงในโปรแกรม ArduinoIDE ดังรูปที่ 4.4, 4.6 และ 4.7 จากนั้นนำมาสรุปผลในตาราง เพื่อหาค่าเฉลี่ยและเปอร์เซ็นต์ความ

คลาดเคลื่อน โดยค่าของมุมที่วัดได้ในแนวแกน x แสดงได้ในตารางที่ 4.2 และค่าของมุมที่วัดได้ในแนวแกน y แสดงได้ในตารางที่ 4.3

ตารางที่ 4.2 ค่าของมุมที่วัดได้จากระบบเซ็นเซอร์วัดมุมก้ม-มุมเงยในแนวแกน x

ลำดับที่	มุมในแนวแกน x (องศา)						
	0	30	60	90	120	150	180
1	0.00	33.69	60.47	90.00	-	-	-
2	0.00	33.69	60.47	90.00	-	-	-
3	0.00	33.69	60.47	90.00	-	-	-
4	0.00	33.53	60.47	90.00	-	-	-
5	0.00	28.91	60.47	90.00	-	-	-
6	0.00	28.91	60.47	90.00	-	-	-
7	0.00	33.69	60.47	90.00	-	-	-
8	0.00	29.05	60.47	90.00	-	-	-
9	0.00	29.05	60.47	90.00	-	-	-
10	0.00	29.05	60.47	90.00	-	-	-
เฉลี่ย	0.00	31.326	60.47	90.00	-	-	-
เปอร์เซ็นต์ความคลาดเคลื่อน (%)	0	4.42	0.78333	0	-	-	-

COM12 (Arduino/Genuino Uno)

```
Angle of X axis = 33.69
Angle of X axis = 33.69
Angle of X axis = 33.69
Angle of X axis = 33.53
Angle of X axis = 28.91
Angle of X axis = 28.91
Angle of X axis = 33.69
Angle of X axis = 29.05
Angle of X axis = 29.05
Angle of X axis = 29.05
Angle of X axis = 29.05
Angle of X axis = 29.05
Angle of X axis = 29.05
```

COM12 (Arduino/Genuino Uno)

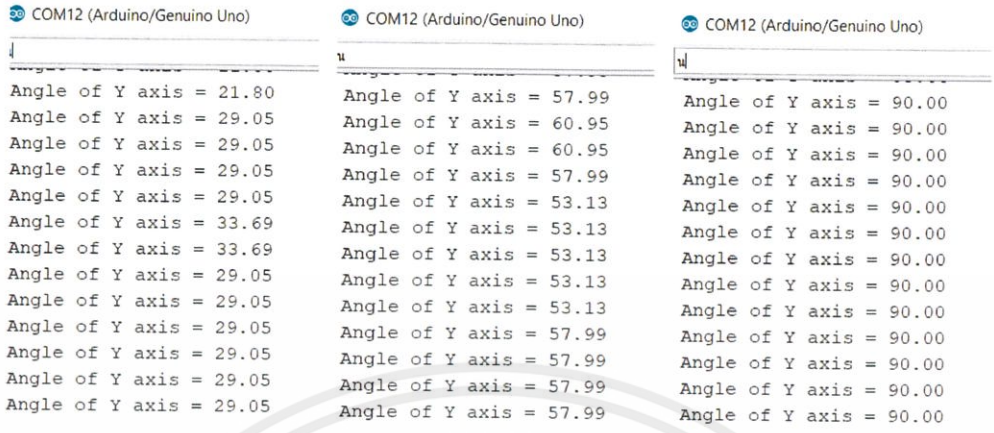
```
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
Angle of X axis = 60.47
```

COM12 (Arduino/Genuino Uno)

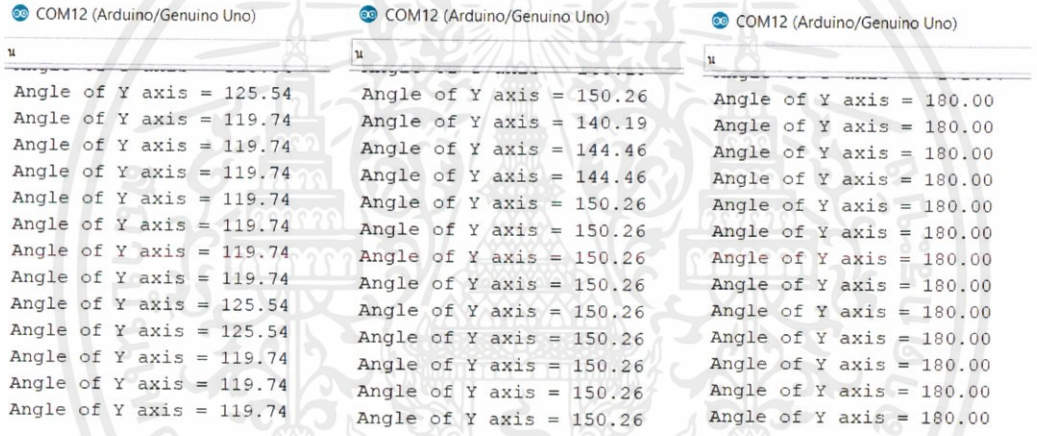
```
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
Angle of X axis = 90.00
```

รูปที่ 4.4 ค่าที่ได้จากระบบเซ็นเซอร์วัดมุมก้ม-มุมเงยในแนวแกน x ที่ 30, 60, 90 องศา

รูปที่ 4.5 การทดสอบการทำงานของระบบเซ็นเซอร์วัดมุมก้ม-มุมเงยในแนวแกน y ที่ 30 องศา



รูปที่ 4.6 ค่าที่ได้จากระบบเซ็นเซอร์วัดมุมก้ม-มุมเงยในแนวแกน y ที่ 30, 60, 90 องศา



รูปที่ 4.7 ค่าที่ได้จากระบบเซ็นเซอร์วัดมุมก้ม-มุมเงยในแนวแกน y ที่ 120, 150, 180 องศา

จากตารางที่ 4.3 สามารถคำนวณหาค่าเฉลี่ยของค่าของมุมที่วัดได้ในแนวแกน y และหาเปอร์เซ็นต์ความคลาดเคลื่อนของมุมทั้งหมด 10 ค่าได้ดังนี้

ที่มุม 0 องศา มีค่าเฉลี่ยเท่ากับ 0.00 องศา มีเปอร์เซ็นต์ความคลาดเคลื่อน 0 %

ที่มุม 30 องศา มีค่าเฉลี่ยเท่ากับ 29.25 องศา มีเปอร์เซ็นต์ความคลาดเคลื่อน -2.49 %

ที่มุม 60 องศา มีค่าเฉลี่ยเท่ากับ 56.15 องศา มีเปอร์เซ็นต์ความคลาดเคลื่อน -6.4133 %

ที่มุม 90 องศา มีค่าเฉลี่ยเท่ากับ 90.00 องศา มีเปอร์เซ็นต์ความคลาดเคลื่อน 0 %

ที่มุม 120 องศา มีค่าเฉลี่ยเท่ากับ 121.48 องศา มีเปอร์เซ็นต์ความคลาดเคลื่อน 1.23333 %

ที่มุม 150 องศา มีค่าเฉลี่ยเท่ากับ 148.09 องศา มีเปอร์เซ็นต์ความคลาดเคลื่อน -1.2713 %

ที่มุม 180 องศา มีค่าเฉลี่ยเท่ากับ 180.00 องศา มีเปอร์เซ็นต์ความคลาดเคลื่อน 0 %

ตารางที่ 4.3 ค่าของมุมที่วัดได้จากระบบเซ็นเซอร์วัดมุมก้ม-มุมเงยในแนวแกน y

ลำดับที่	มุมในแนวแกน y (องศา)						
	0	30	60	90	120	150	180
1	0.00	21.80	57.99	90.00	125.54	150.26	180.00
2	0.00	29.05	60.95	90.00	119.74	140.19	180.00
3	0.00	29.05	60.95	90.00	119.74	144.46	180.00
4	0.00	29.05	57.99	90.00	119.74	144.46	180.00
5	0.00	29.05	53.13	90.00	119.74	150.26	180.00
6	0.00	33.69	53.13	90.00	119.74	150.26	180.00
7	0.00	33.69	53.13	90.00	119.74	150.26	180.00
8	0.00	29.05	53.13	90.00	119.74	150.26	180.00
9	0.00	29.05	53.13	90.00	125.54	150.26	180.00
10	0.00	29.05	57.99	90.00	125.54	150.26	180.00
เฉลี่ย	0.00	29.25	56.15	90.00	121.48	148.09	180.00
เปอร์เซ็นต์ความคลาดเคลื่อน (%)	0	-2.49	-6.4133	0	1.23333	-1.2713	0

จากเปอร์เซ็นต์ความคลาดเคลื่อนที่คำนวณมาได้ จึงสามารถสรุปได้ว่า เซ็นเซอร์วัดมุมก้ม-มุมเงยมีความแม่นยำในการใช้งานจริง

และจากการทดสอบการทำงานของระบบเซ็นเซอร์วัดมุมก้ม-มุมเงย ซึ่งใช้โมดูล GY-80 เมื่อทำการวัดค่าสัญญาณบิตข้อมูลที่รับได้จาก Oscilloscope จะสามารถถอดรหัสข้อมูลได้ ดังรูปที่ 4.8

รูปที่ 4.8 ค่าสัญญาณบิตข้อมูลจากโมดูล GY-80

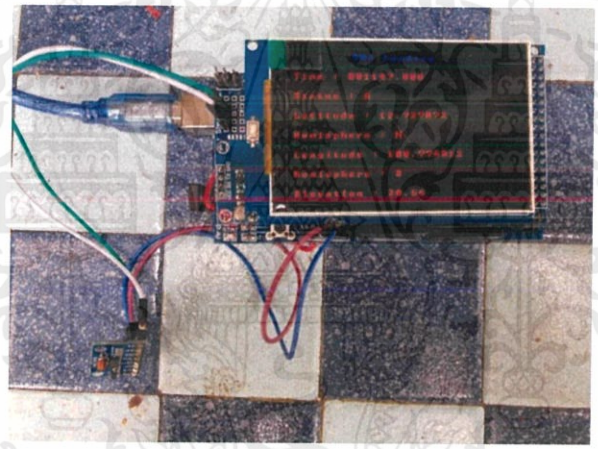
ค่าบิตข้อมูลที่รับได้จากโมดูล GY-80 จะแสดงข้อมูลออกมาเป็นบิตข้อมูลโดยเริ่มจากส่งข้อมูล SDA และ SCL จากนั้น ระหว่างที่รอสัญญาณขาขึ้นของ Clock แรก SDA จะเริ่มกำหนดค่าบิตแรก โดย Master จะส่งค่าบิตแรกไปพร้อมกับสัญญาณ Clock และไอซีที่เป็น Slave บนบัสจะเริ่มอ่านค่าในจังหวะที่ SCL เป็น H จากนั้นก็จะเป็นอย่างนี้ไปอีกเพื่อส่งค่า Address ของไอซีที่ต้องการจะติดต่อด้วย รวม 7 บิต และตามด้วยบิตที่ 8 ซึ่งจะระบุว่าจะสั่งให้ Slave รอคำสั่ง (Write ระบุโดย SDA เป็น H) หรือจะรออ่านค่าที่ส่งมาจาก Slave (Read ระบุค่าโดย SDA เป็น L) ซึ่งในที่นี้เป็น L จากนั้น บิตที่ 9 จะเป็นการตอบรับจาก Slave ที่มี Address ตรงกับที่ Master ส่งไป ถ้ามี Slave ตอบรับ (Acknowledge) โดยการดึงสัญญาณ SDA ลง L จะแปลได้ว่ามี Slave พร้อมจะสื่อสารด้วย แต่ถ้า SDA ค้างที่ H แปลว่า Master ไม่มีการตอบรับ (Not Acknowledge) ในที่นี้บิตที่ 9 เป็น ACK เมื่อ Data ครบ 8 บิตแรกแล้ว SDA จะถูกปล่อยให้เป็น H จากนั้น Master ก็จะสั่งหยุดโดยการส่งสัญญาณ

4.1.3 การทดสอบการทำงานของระบบเซ็นเซอร์ทั้งหมด

ทางผู้จัดทำได้ทำการทดสอบการทำงานของเซ็นเซอร์ทั้งหมด ประกอบไปด้วยเซ็นเซอร์ตรวจจับตำแหน่ง และเซ็นเซอร์วัดมุมก้ม-มุมเงย จะได้ค่าที่แสดงออกมาทางโปรแกรม ArduinoIDE ดังรูปที่ 4.9 และ 4.10

```
COM8
23:23:37.810 -> Time: 162338.000
23:23:37.810 -> Status: A
23:23:37.810 -> Latitude: 13.727462
23:23:37.846 -> Hemisphere: N
23:23:37.882 -> Longitude: 100.776312
23:23:37.882 -> Hemisphere: E
23:23:37.954 -> Elevation = 125.54
23:23:39.826 -> Time: 162340.000
23:23:39.826 -> Status: A
23:23:39.826 -> Latitude: 13.727468
23:23:39.862 -> Hemisphere: N
23:23:39.898 -> Longitude: 100.776313
23:23:39.898 -> Hemisphere: E
23:23:39.934 -> Elevation = 125.54
```

รูปที่ 4.9 ผลการทดสอบการทำงานของระบบเซ็นเซอร์ทั้งหมดจากโปรแกรม ArduinoIDE



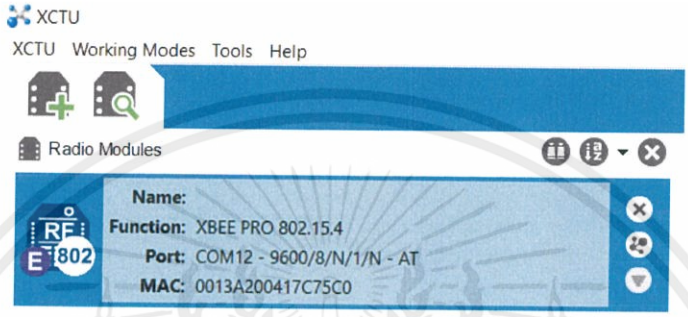
รูปที่ 4.10 ผลการทดสอบการทำงานของระบบเซ็นเซอร์ทั้งหมดจากหน้าจอมอนิเตอร์

จากการทดลองจะเห็นได้ว่า เราสามารถใช้งานระบบเซ็นเซอร์ได้จริง โดยการใช้บอร์ด Arduino MEGA2560 ร่วมกัน และสามารถแสดงผลค่าที่ได้ที่หน้าจอมอนิเตอร์ได้ตามที่ต้องการ

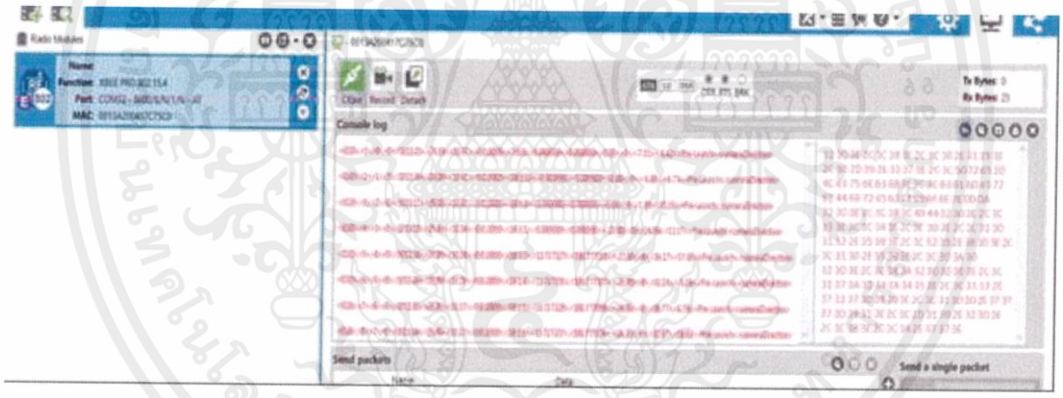
4.2 ผลการทดสอบการรับ-ส่งข้อมูล CanSat มายังสถานีภาคพื้นดิน

ผู้จัดทำได้ทำการทดสอบการรับ-ส่งข้อมูลจากฝั่งสถานีส่งมายังสถานีรับ ผ่านอุปกรณ์ Xbee Pro series1 ทั้ง 2 ตัว ซึ่งมีข้อมูลของพอร์ตแสดงดังรูปที่ 4.11 โดยอาศัยโปรแกรม XCTU

ในการทำการสื่อสารระหว่าง Xbee และ โปรแกรม ArduinoIDE ในการแสดงผลของข้อมูลที่รับได้ว่ามีข้อมูลอะไรบ้าง ซึ่งจะแสดงผลได้ดังรูปที่ 4.12 และ 4.13 ซึ่งคือผลการทดสอบการรับ-ส่งข้อมูลจากโปรแกรม XCTU และผลการทดสอบการรับ-ส่งข้อมูลจากหน้าจอ Serial Monitor ของโปรแกรม ArduinoIDE ตามลำดับ



รูปที่ 4.11 พอร์ตที่ใช้รับค่าจากโปรแกรม XCTU



รูปที่ 4.12 ผลการทดสอบการรับ-ส่งข้อมูลจากโปรแกรม XCTU

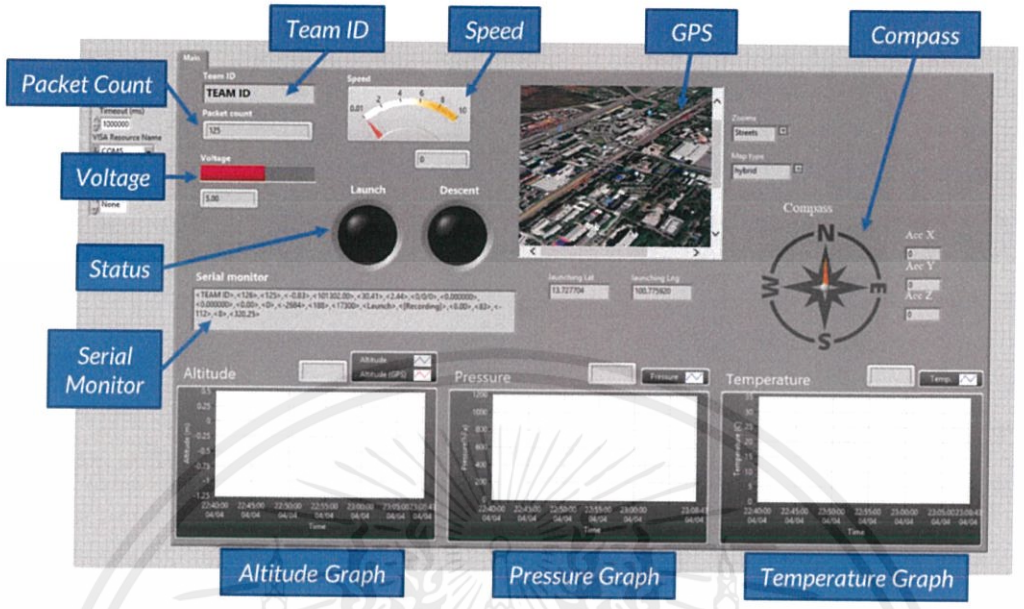
ผลการทดสอบพบว่า สถานีภาคพื้นดินและแบบจำลองดาวเทียมกระป๋อง สามารถรับส่งข้อมูลกันได้โดยไม่ติดขัด เป็นระยะทางประมาณ 450 เมตร และหลังจาก 450 เมตรจึงจะไม่สามารถรับข้อมูลได้ ซึ่งสังเกตได้จากหน้าจอโปรแกรม XCTU ไม่แสดงผลข้อมูลใด ๆ

รูปที่ 4.13 ผลการทดสอบการรับ-ส่งข้อมูลจากโปรแกรม ArduinoIDE

โดยข้อมูลที่รับมาจาก CanSat ได้นั้น จะประกอบไปด้วย <TEAM ID>, <MISSION TIME>, <PACKET COUNT>, <ALTITUDE>, <PRESSURE>, <TEMP>, <VOLTAGE>, <GPS TIME>, <GPS LATITUDE>, <GPS LONGITUDE>, <GPS ALTITUDE>, <GPS SATS>, <PITCH>, <ROLL>, <BLADE SPIN RATE>, <SOFTWARE STATE> และ <BONUS DIRECTION> จากนั้นดึงข้อมูลที่รับได้มาเก็บไว้ที่ไฟล์ .csv ที่มีรายละเอียดได้ดังรูปที่ 4.14 และสามารถแสดงข้อมูลต่าง ๆ ซึ่งมีส่วนประกอบในหน้าจอ LabVIEW ดังรูปที่ 4.15 ซึ่งเป็นรูปแบบที่บังคับเพื่อใช้ในการแข่งขัน CanSat Competition 2019

	A	B	C	D	E	F	G	H	I	J	K
1	Team ID	Packet cou	Altitude	Pressure	Temperatu	Voltage	GPS time	GPS lat	GPS lon	Pitch	Roll
2	3451	23	43.17	1004	34.6	7.53	6:31:23	13.727392	100.772867	0	0
3	3451	24	42.84	1004	34.6	7.50	6:31:24	13.727406	100.772867	0	0
4	3451	25	42.75	1004	34.5	7.50	6:31:25	13.727412	100.772867	0	0
5	3451	26	42.67	1004	34.6	7.48	6:31:26	13.727423	100.772868	0	0
6	3451	27	42.33	1004	34.5	7.48	6:31:27	13.727441	100.772868	0	0
7	3451	28	42.08	1004	34.5	7.48	6:31:28	13.727462	100.772868	0	0
8	3451	29	42.08	1004	34.5	7.45	6:31:29	13.727485	100.772868	0	0
9	3451	30	41.83	1004	34.6	7.45	6:31:30	13.727507	100.772868	0	0
10	3451	31	41.50	1004	34.6	7.45	6:31:31	13.727534	100.772869	0	0
11	3451	32	41.42	1004	34.6	7.43	6:31:32	13.727558	100.772869	0	0
12	3451	33	40.92	1004	34.5	7.43	6:31:33	13.727574	100.772869	0	0
13	3451	34	40.83	1004	34.5	7.43	6:31:34	13.727597	100.772870	0	0
14	3451	35	40.66	1004	34.6	7.43	6:31:35	13.727611	100.772870	0	0
15	3451	36	40.57	1005	34.5	7.40	6:31:36	13.727629	100.772870	0	0
16	3451	37	39.49	1005	34.5	7.40	6:31:37	13.727645	100.772870	0	0
17	3451	38	39.07	1005	34.6	7.40	6:31:38	13.727662	100.772870	0	0
18	3451	39	38.38	1005	34.6	7.40	6:31:39	13.727679	100.772871	0	0
19	3451	40	38.27	1005	34.6	7.40	6:31:40	13.727694	100.772871	0	-6.34
20	3451	41	38.25	1006	34.5	7.40	6:31:41	13.727706	100.772871	0	0
21	3451	42	38.11	1006	34.5	7.40	6:31:42	13.727718	100.772871	0	-6.34

รูปที่ 4.14 ข้อมูลที่สามารถรับมาได้จาก CanSat ในรูปแบบไฟล์ .csv



รูปที่ 4.15 ส่วนประกอบของหน้าจอแสดงผลโปรแกรม LabVIEW

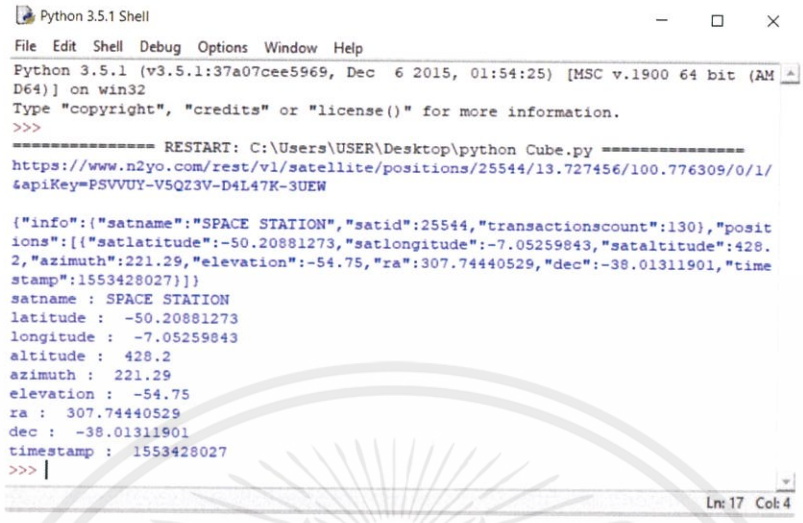
4.3 ผลการทดสอบการดึงข้อมูลการเคลื่อนที่ของ CubeSat

4.3.1 ผลการทดสอบการดึงข้อมูลจาก N2YO.com ด้วย Python

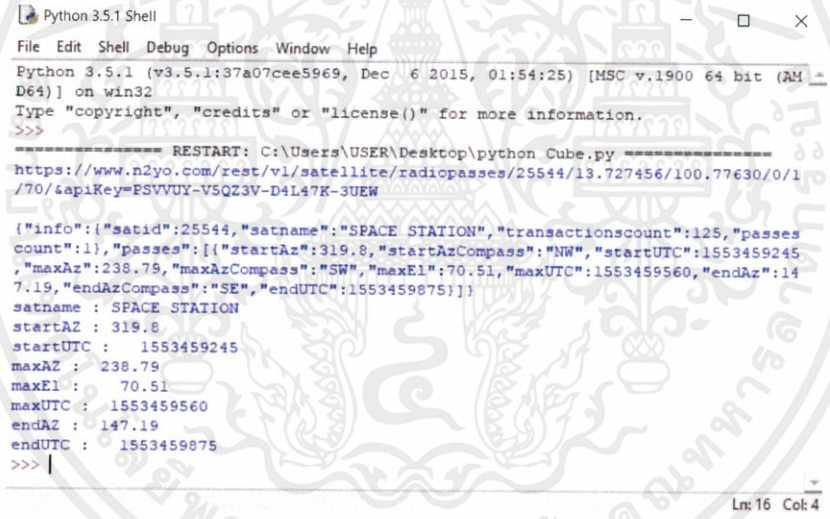
ผู้จัดทำได้ทำการทดสอบการดึงข้อมูลการเคลื่อนที่ของ CubeSat โดยใช้ภาษา Python ผ่านโปรแกรมที่มีชื่อว่า PyCharm ในการประมวลผลข้อมูล ข้อมูลที่ดึงจากเว็บ N2YO.com นั้นจะดึงจากในส่วนที่เว็บอนุญาตให้เข้าถึงผ่านระบบ API แบบ URL

ข้อมูลที่ต้องการจะประกอบไปด้วยข้อมูล 2 ชุด คือ ข้อมูลปัจจุบันของ CubeSat โดยส่วนนี้ต้องการการกำหนดค่าละติจูดและลองติจูดของสถานที่รับสัญญาณ และกำหนดเวลาที่ต้องการจะดึงข้อมูล ในที่นี้จะใช้เวลา 1 วินาที ผลการดึงข้อมูลแสดงได้ในรูปที่ 4.16

ข้อมูลที่ต้องการอีกชุด คือ ข้อมูลในอนาคตของ CubeSat เป็นข้อมูลที่บ่งบอกว่าช่วงที่ CubeSat ผ่านน่านฟ้าของเรานั้น มีค่าต่าง ๆ เป็นอย่างไร ต้องการการกำหนดค่าละติจูดและลองติจูดของสถานที่รับสัญญาณ กำหนดเวลาที่ต้องการดึงข้อมูล ในที่นี้ใช้เวลา 1 วัน และกำหนดค่ามุม Elevation ต่ำสุดที่ต้องการ ในที่นี้ใช้ 30 องศา ผลการดึงข้อมูลแสดงได้ในรูปที่ 4.17



รูปที่ 4.16 ตัวอย่างข้อมูลปัจจุบันของ CubeSat



รูปที่ 4.17 ตัวอย่างข้อมูลอนาคตของ CubeSat

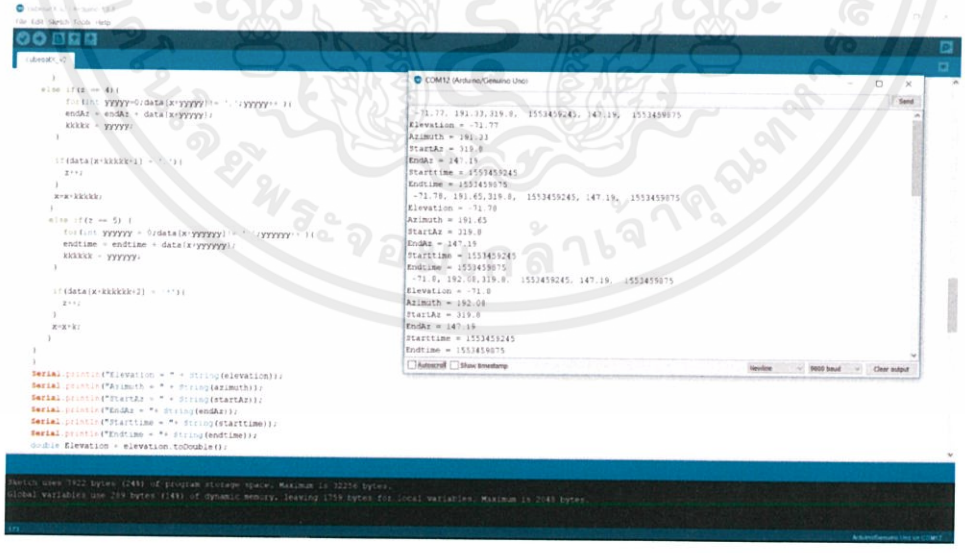
4.3.2 ผลการทดสอบการส่งข้อมูลจาก Python ไป Arduino

เมื่อสามารถดึงข้อมูลจาก Python ได้ ขั้นตอนถัดมาคือ การส่งข้อมูลจาก Python ไปยัง Arduino โดยเราสามารถทำได้โดยใช้ library ของ Python ที่ชื่อ PySerial มาช่วย โดยในส่วนนี้จะใช้ Arduino 2 ตัวในการรับข้อมูลและส่งข้อมูลเนื่องจากพอร์ตไม่เพียงพอ จากนั้นจึงแสดงผลที่ได้ไปยัง Serial monitor โดยหน้าจอฝั่งโปรแกรม Python แสดงได้ดังรูปที่ 4.18 และหน้าจอฝั่งโปรแกรม ArduinoIDE แสดงได้ดังรูปที่ 4.19

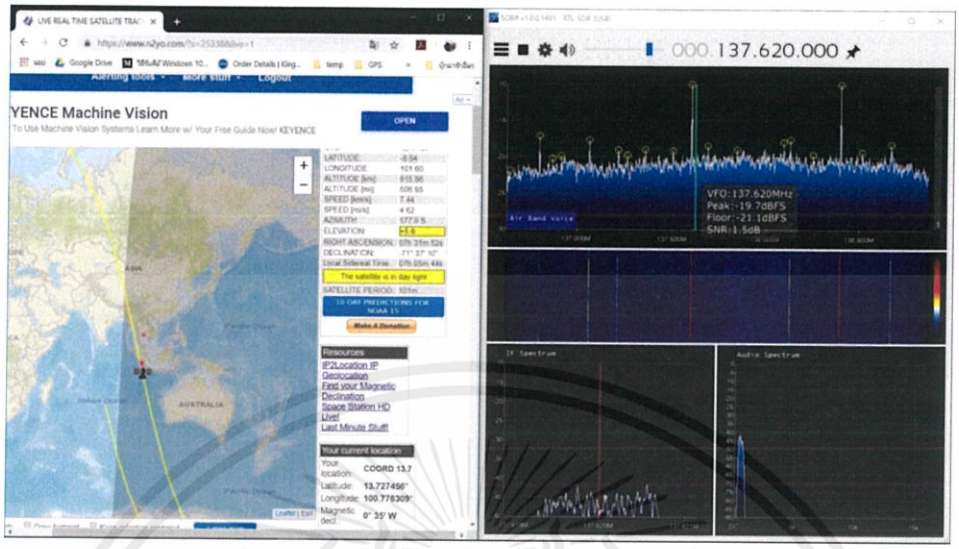
4.3.3 ผลการทดสอบการทำงานของสายอากาศ

ก่อนที่จะทำการใช้ระบบติดตามแบบจำลองดาวเทียมขนาดเล็กจริง จะต้องทำการทดสอบสายอากาศที่ใช้ก่อน ว่ามีประสิทธิภาพในการรับสัญญาณจาก CubeSat ภายใต้เงื่อนไขและข้อจำกัดที่กำหนดขึ้นมากน้อยเพียงใด โดยจะใช้ RTL-SDR เป็นตัวแปลงสัญญาณเข้าคอมพิวเตอร์ และใช้โปรแกรม SDR# แสดงผลรูปสัญญาณ โดยได้ผลลัพธ์ดังรูปที่ 4.20

รูปที่ 4.18 ผลการส่งข้อมูลจาก Python ไป Arduino (ฝั่งโปรแกรม Python)



รูปที่ 4.19 ผลการส่งข้อมูลจาก Python ไป Arduino (ฝั่งโปรแกรม ArduinoIDE)



รูปที่ 4.20 ผลการทดสอบการทำงานของสายอากาศ

4.3.4 ผลการทดสอบการรับสัญญาณของ CubeSat

ผู้จัดทำได้ทำการทดสอบการรับสัญญาณของ CubeSat ส่งมายังสถานีรับ เป็นจำนวน 5 ดวง ซึ่งผลการทดสอบการรับสัญญาณมีดังนี้

4.3.4.1 ผลการทดสอบการรับสัญญาณ NOAA 15

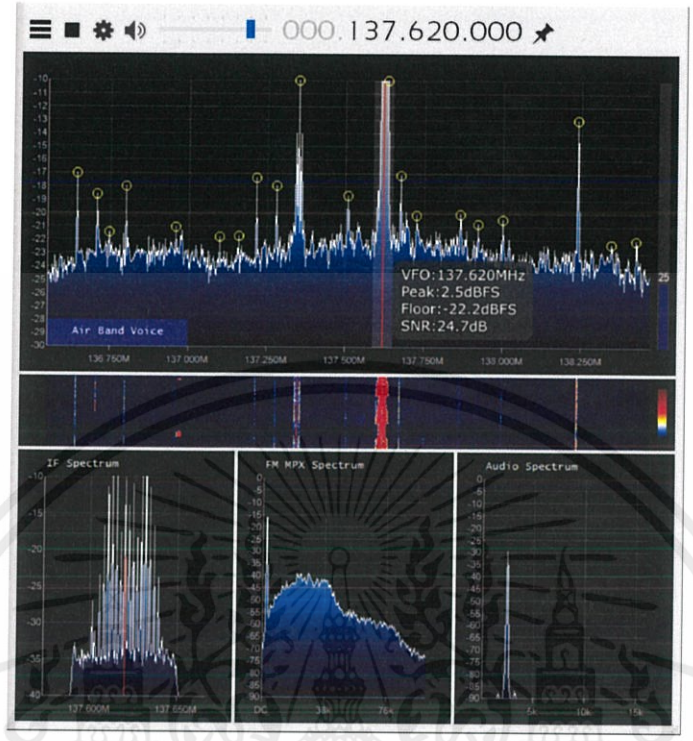
ผู้จัดทำได้ทำการทดสอบการรับสัญญาณ NOAA 15 เมื่อวันที่ 25 มีนาคม 2562 เวลา 19.10 ถึง 19.25 น. เป็นเวลาทั้งหมด 15 นาที หรือ 900 วินาที ทุก ๆ 30 วินาที

ตารางที่ 4.4 ผลการทดสอบการรับสัญญาณ NOAA 15

เวลา (วินาที)	มุม Elevation (องศา)	SNR (dB)
30	+0.0	4.9
60	+5.6	5.1
90	+13.0	5.3
120	+17.4	7.2
150	+22.3	10.6
180	+25.6	12.2
210	+29.5	15.4
240	+36.7	17.6

ตารางที่ 4.4 ผลการทดสอบการรับสัญญาณ NOAA 15 (ต่อ)

เวลา (วินาที)	มุม Elevation (องศา)	SNR (dB)
270	+40.2	20.8
300	+43.1	21.6
330	+49.5	23.4
360	+52.5	24.7
390	+48.6	22.8
420	+47.2	21.6
450	+44.9	19.2
480	+42.7	18.3
510	+40.1	16.7
540	+39.5	15.4
570	+33.6	15.3
600	+31.3	13.9
630	+24.1	12.1
660	+19.2	11.6
690	+15.4	11.2
720	+14.1	10.6
750	+11.3	11.0
780	+10.5	9.4
810	+8.6	7.5
840	+5.0	5.9
870	+0.8	5.6
900	+0.0	4.5



รูปที่ 4.21 ผลการทดสอบการรับสัญญาณ NOAA 15 จากโปรแกรม SDR#

จากตารางที่ 4.4 และรูปที่ 4.21 ซึ่งแสดงผลการทดสอบการรับสัญญาณ NOAA 15 จะเห็นได้ว่าที่มุม Elevation เท่ากับ +52.5 องศา จะมีค่า SNR สูงสุด คือ 24.7 dB

4.3.4.2 ผลการทดสอบการรับสัญญาณ NOAA 18

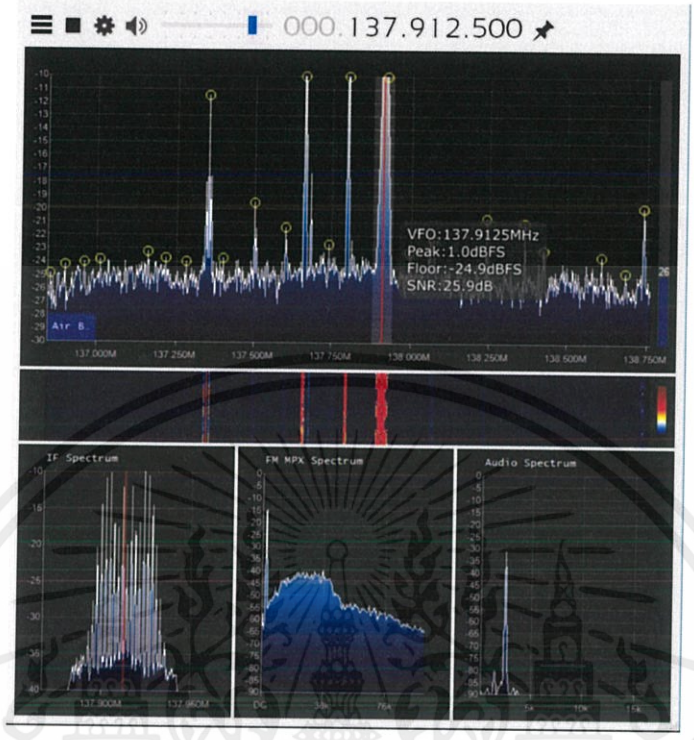
ผู้จัดทำได้ทำการทดสอบการรับสัญญาณ NOAA 18 เมื่อวันที่ 25 มีนาคม 2562 เวลา 20.19 ถึง 20.34 น. เป็นเวลาทั้งหมด 15 นาที หรือ 900 วินาที ทุก ๆ 30 วินาที

ตารางที่ 4.5 ผลการทดสอบการรับสัญญาณ NOAA 18

เวลา (วินาที)	มุม Elevation (องศา)	SNR (dB)
30	+0.0	5.1
60	+0.1	5.6
90	+0.7	9.5
120	+1.5	11.6

ตารางที่ 4.5 ผลการทดสอบการรับสัญญาณ NOAA 18 (ต่อ)

เวลา (วินาที)	มุม Elevation (องศา)	SNR (dB)
150	+2.1	13.0
180	+3.4	15.8
210	+7.6	17.6
240	+9.2	18.5
270	+10.6	20.3
300	+11.8	21.6
330	+12.9	23.2
360	+14.1	24.7
390	+14.6	25.9
420	+14.3	25.4
450	+12.6	24.6
480	+11.7	22.3
510	+10.6	21.6
540	+10.0	20.0
570	+9.2	18.5
600	+8.1	16.3
630	+7.6	14.2
660	+5.2	12.9
690	+4.9	11.3
720	+4.3	10.3
750	+3.5	8.6
780	+3.2	7.5
810	+2.9	5.3
840	+2.1	5.1
870	+1.2	4.9
900	+0.0	5.0



รูปที่ 4.22 ผลการทดสอบการรับสัญญาณ NOAA 18 จากโปรแกรม SDR#

จากตารางที่ 4.5 และรูปที่ 4.22 ซึ่งแสดงผลการทดสอบการรับสัญญาณ NOAA 18 จะเห็นได้ว่าที่มุม Elevation เท่ากับ +14.6 องศา จะมีค่า SNR สูงสุด คือ 25.9 dB

4.3.4.3 ผลการทดสอบการรับสัญญาณ NOAA 19

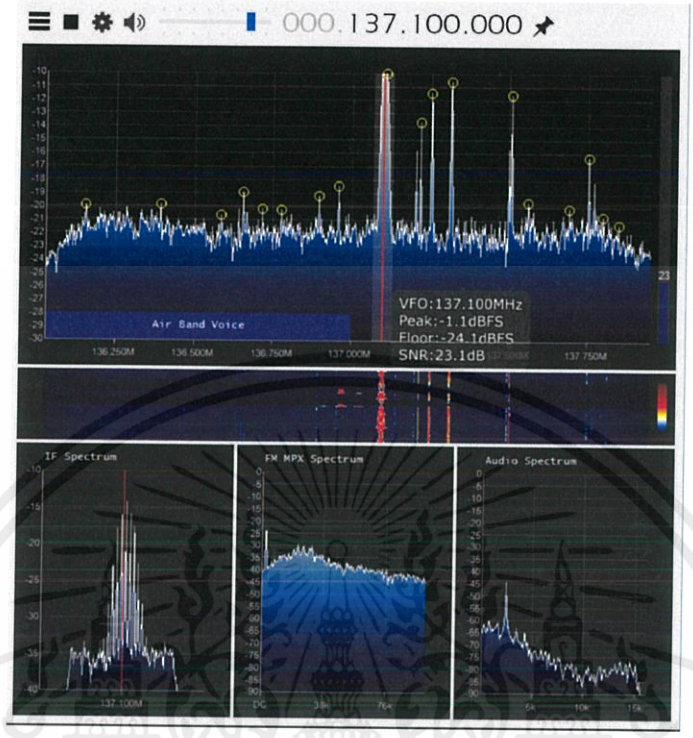
ผู้จัดทำได้ทำการทดสอบการรับสัญญาณ NOAA 19 เมื่อวันที่ 25 มีนาคม 2562 เวลา 16.24 ถึง 16.39 น. เป็นเวลาทั้งหมด 15 นาที หรือ 900 วินาที ทุก ๆ 30 วินาที

ตารางที่ 4.6 ผลการทดสอบการรับสัญญาณ NOAA 19

เวลา (วินาที)	มุม Elevation (องศา)	SNR (dB)
30	+0.0	5.1
60	+0.9	5.5
90	+1.6	6.8
120	+2.3	7.5

ตารางที่ 4.6 ผลการทดสอบการรับสัญญาณ NOAA 19 (ต่อ)

เวลา (วินาที)	มุม Elevation (องศา)	SNR (dB)
150	+3.6	9.4
180	+4.9	11.6
210	+6.0	12.9
240	+7.5	14.3
270	+8.6	16.5
300	+9.1	18.1
330	+10.3	20.6
360	+11.9	23.0
390	+13.6	23.1
420	+13.2	22.8
450	+12.3	22.3
480	+11.6	21.6
510	+10.8	20.4
540	+9.9	18.6
570	+9.1	15.6
600	+8.2	14.5
630	+7.5	13.1
660	+6.4	11.0
690	+5.6	9.2
720	+4.2	8.6
750	+2.8	7.4
780	+1.9	5.3
810	+1.3	5.5
840	+0.0	5.2



รูปที่ 4.23 ผลการทดสอบการรับสัญญาณ NOAA 19 จากโปรแกรม SDR#

จากตารางที่ 4.6 และรูปที่ 4.23 ซึ่งแสดงผลการทดสอบการรับสัญญาณ NOAA 19 จะเห็นได้ว่าที่มุม Elevation เท่ากับ +13.6 องศา จะมีค่า SNR สูงสุด คือ 23.1 dB

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ระบบสถานีภาคพื้นดินของแบบจำลองดาวเทียมกระป๋องและดาวเทียมขนาดเล็ก (Ground Station System for CanSat and CubeSat) ที่ได้พัฒนาขึ้นในปริญญานิพนธ์ฉบับนี้ สามารถแบ่งเป็น 2 ส่วนหลักๆ คือ ส่วน Software และ ส่วน Hardware

ส่วน Software คือ การเขียนโปรแกรมโดยสามารถสรุปผลการดำเนินงานได้ดังนี้

1. สามารถออกแบบและเขียนโปรแกรมในการควบคุมระบบติดตามสายอากาศได้

2. สามารถเก็บค่าข้อมูลต่าง ๆ ที่รับมาได้จากแบบจำลองดาวเทียมกระป๋องและแบบจำลองดาวเทียมขนาดเล็ก โดยใช้ระบบเซ็นเซอร์ของสถานีภาคพื้นดิน ได้แก่ ความสูง ความดัน อุณหภูมิ ตำแหน่ง GPS เป็นต้น

3. สามารถดึงข้อมูลการเคลื่อนที่ของแบบจำลองดาวเทียมขนาดเล็กจากเว็บไซต์ N2YO.com แบบ Real-Time ได้

4. สามารถออกแบบหน้าจอแสดงผลในโปรแกรม LabVIEW ได้

ส่วน Hardware สามารถสรุปผลการดำเนินงานได้ดังนี้

1. สามารถออกแบบระบบมอเตอร์เคลื่อนที่ 2 แกน สำหรับติดตามดาวเทียม ที่ช่วยปรับทิศทางของสายอากาศให้สามารถชี้ไปยังเป้าหมายโดยอัตโนมัติได้

2. ระบบเซ็นเซอร์สามารถตรวจจับตำแหน่งของอุปกรณ์จากเซ็นเซอร์ตรวจจับตำแหน่ง และวัดมุมก้ม-มุมเงยจากเซ็นเซอร์วัดมุมก้ม-มุมเงยได้

5.2 ข้อเสนอแนะ

- ควรออกแบบให้ระบบติดตามสายอากาศสามารถทำงานแบบอัตโนมัติได้อย่างมีเสถียรภาพมากกว่านี้

- สายอากาศที่ใช้มีกำลังขยายไม่เพียงพอต่อความต้องการ ทำให้ผลการทดสอบการรับ-ส่งข้อมูลออกมาไม่ดีเท่าที่ควร

- การทำงานของสายอากาศยังมีปัญหา เนื่องจากเกิดปัญหาในการติดตั้งสายอากาศกับระบบมอเตอร์เคลื่อนที่ 2 แกน

บรรณานุกรม

[1] ดาราศาสตร์ศึกษา (Astronomy Education). “CanSat คืออะไร? CanSat ดาวเทียมกระป๋อง การปล่อยแคนแซท การทำงานของ CanSat ประโยชน์ CanSat | ดาราศาสตร์ศึกษา (Astronomy Education).” http://www.astroeducation.com/what-is-cansat./.

[2] UNISEC THAILAND. “แคนแซท (CanSat).” http://www.astroeducation.com/wp-content/uploads/2017/06/cansat-lesa.jpg.

[3] วารุณี ลักษณจันทร์ - GoToKnow. “การสื่อสารผ่านดาวเทียม (Satellite Communication).” https://www.gotoknow.org/posts/560608.

[4] ศิริพร ครูบา. “เทคโนโลยีการสื่อสารและอินเทอร์เน็ต: การสื่อสารผ่านดาวเทียม (Satellite Communication).” http://siripornk.blogspot.com/2010/08/satellite-communication.html.

[5] มหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ. “สถานีดาวเทียมภาคพื้นดิน (Ground segment).” http://lms.rmutsb.ac.th/elearning/claroline/backends/download.php?url=L%2BC4hOC4o%2BC4seC5ieC4h%2BC4l%2BC4teC5iF8xM1%2FguKrguJbguLLguJnguLXguJTguLLguKfguYDguJfguLXguKLguKHguKDguLLgulTguJ7guLfguYnguJnguJTguLTguJkucGRm&cidReset=true&cidReq=1962301.

[6] Geo-Informatics. “ดาวเทียมจิ๋ว CubeSat และ จอสัมผัส GeoTouch.” https://gi4u.wordpress.com/2013/07/01/ดาวเทียมจิ๋ว-cubesat-และ-จอสัมผ/.

[7] ThaiEasyElec.com. “Zigbee and Xbee BASIC ตอน Zigbee คืออะไร.” https://www.thaieasyelec.com/article-wiki/basic-electronics/what-is-zigbee.html.

[8] Com-School-MaxZa. “Dongle คืออะไร.” http://com_school.blogspot.com/2011/03/dongle.html.

[9] วิทยาลัยเทคนิคสระบุรี. “เอกสารประกอบการสอนวิชาไมโครคอนโทรลเลอร์.” http://www.sbt.ac.th/new/sites/default/files/TNP_Unit_2.pdf.

[10] โรบอทสยาม อุปกรณ์หุ่นยนต์ Arduino : Inspired by LnwShop.com. "การติดตั้งโปรแกรม Arduino (IDE)." http://www.robotsiam.com/article/2/การติดตั้งโปรแกรม-arduino-ide-และ-การติดตั้งไดร์เวอร์.

[11] aramboy2525. "LabVIEW คืออะไร? ประโยชน์และการใช้งาน LabView." https://www.lcdtvthailand.com/webboard/index.php?topic=392759.0.

[12] electricaltechnology. "What-is-LabVIEW-and-How-to-make-basic-Electrical-projects-in-LabVIEW." http://www.electricaltechnology.org/wp-content/uploads/2015/10/What-is-LabVIEW-and-How-to-make-basic-Electrical-Projects-in-LabVIEW.jpg.

[13] MAHOSOT. "LABVIEW คือ ?? การประยุกต์ใช้งานด้าน Automation และ Instrument." http://keil-cvi.com/labview-คือ/.

[14] มหาวิทยาลัยสยาม. "ทฤษฎีและหลักการ." http://www.research-system.siam.edu/images/coop/DESIGN_AND_CONSTRUCTION_OF_ELECTRICAL_MEASUREMENT_USING_LABVIEW___PROGRAM/ch2.pdf.

[15] mindphp.com. "Python คืออะไร ไพธอน คือภาษา สำหรับเขียนโปรแกรมคอมพิวเตอร์ ภาษาหนึ่ง." https://www.mindphp.com/คู่มือ/73-คืออะไร/2417-python-คืออะไร.html.

[16] Mandy Sladden. "Python Class Room." https://sites.google.com/site/pythonclassroom/.

[17] ebay. "New 25dBi 2.4GHz Wireless WLAN WiFi Antenna RP-SMA for Modem PCI Card Route I5S2." https://www.ebay.com/itm/New-25dBi-2-4GHz-Wireless-WLAN-WiFi-Antenna-RP-SMA-for-Modem-PCI-Card-Route-I5S2-/112655849661.

[18] ALL MAX ELECTRONICS LIMITED. "433 MHz 12 dBi High Gain Long Range UHF Yagi Antenna." http://www.allmaxantenna.com/433-mhz-12-dbi-high-gain-long-range-uhf-yagi-antenna/.

[19] DIYElectronics. "NEMA 23 Stepper Motor | 3A , 30kg-cm | DIYElectronics." https://www.diyelectronics.co.za/store/stepper-motors/478-nema-23-57bygh115-003.html.

[20] Cloudfront.net. "Clear Extrusion 20x20." https://dzevsq2emy08i.cloudfront.net/paperclip/technology_image_uploaded_images/24217/large/Clear%20Extrusion%2020x20.jpg?1377705802.

[21] Core Electronics. "tb6600-stepper-motor-driver 1000x1000." https://core-electronics.com.au/media/catalog/product/t/b/tb6600-stepper-motor-driver.jpg.

[22] elec2you. "GY-NEO6MV2 Ublox NEO-6M GPS Module." http://www.elec2you.com/product/156/gy-neo6mv2-ublox-neo-6m-gps-module.

[23] Arduitronics. "Inertial Measurement Unit - GY-80 Module for Arduino." https://www.arduitronics.com/article/54/inertial-measurement-unit-gy-80-module-for-arduino-part-1-adxl345.

[24] Honeywell. "3-Axis Digital Compass IC HMC5883L." https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf.

[25] Dept.Computer Engineering, Chiang Mai University. "Digital Compass | Micro/Embedded Systems 2556." https://mcu56.learninginventions.org/?page_id=266.

[26] N2YO.com. "Satellite Predictions Passes for SPACE STATION 1998-067A NORADD 25544." https://n2yo.com/passes/?s=25544.

[27] RIGOL Innovation or nothing. "Spectrum Analyzers | Rigol Technologies - Innovation or nothing." http://www.rigol.eu/products/spectrum-analyzers/.

[28] บริษัท เลกะ คอร์ปอเรชั่น จำกัด. "Oscilloscope ออสซิโลสโคป คืออะไร | เครื่องวัด ความรู้ | LEGATOOL." https://legatool.com/wp/5682/.

[29] ARDUINOTHAI. "8h320x480." https://cq.lnwfile.com/_/cq/_raw/y2/ri/8h.jpg.

[30] zeedless.com. "ARDUINO MEGA2560 3.2 Inch 320 X 480 TFT IPS LCD Display Module." http://www.zeedless.com/shop/index.php?route=product/product&product_id=121.

[31] ThaiEasyElec.com. "บทความ ตอนที่1 แนะนำเพื่อนใหม่ที่ชื่อ Arduino." https://www.thaieasyelec.com/article-wiki/basic-electronics/บทความ-arduino-คืออะไร-เริ่มต้นใช้งาน-arduino.html.

[32] ThaiEasyElec.com. "Arduino Mega 2560 (บอร์ดแท้ 100%)." https://www.thaieasyelec.com/products/development-boards/arduino/official-boards-made-in-italy/arduino-mega-2560-detail.html.

[33] ThaiEasyElec.com. "Arduino Uno R3 (บอร์ดแท้ 100%)." https://www.thaieasyelec.com/products/development-boards/arduino/official-boards-made-in-italy/arduino-uno-r3-detail.html.

[34] ThaiEasyElec.com. "Xbee Series1: XBee Pro 60mW U.FL Connection." https://thaieasyelec.com/products/wireless-modules/zigbee-802-15-4/xbee-series-1/xbee-pro-60mw-u-fl-connection-detail.html.

[35] AliExpress. "ICSH027A Xbeeอะแดปเตอร์มินิMiniture USB Xbeeโล่ ใน ICSH027A Xbee อะแดปเตอร์มินิMiniture USB Xbeeโล่." https://th.aliexpress.com/item/ICSH027A-Xbee-Miniture-USB-Xbee/32681807313.html.

[36] Pairada R. "gps new." https://www.google.co.th/maps/@13.7261518,100.7730677,17.65z/data=!4m3!11m2!2s1VHXzklmQqSkwwY2NLi5IffwnxHA!3e3?hl=th&authuser=0.

ภาคผนวก

Source code ทั้งหมดจะประกอบไปด้วย Code ที่ใช้ในโปรแกรม ArduinoIDE กับบอร์ด Arduino UNO และ Arduino MEGA2560 ซึ่งใช้ควบคุมระบบเซ็นเซอร์ทั้งหมด, ระบบมอเตอร์เคลื่อนที่ 2 แกนของทั้ง CanSat และ CubeSat และระบบควบคุม Joystick เพื่อกำหนดการเคลื่อนที่ของมอเตอร์ และ Code ที่ใช้ในโปรแกรม Python เพื่อดึงข้อมูลการเคลื่อนที่ของ CubeSat แบบ Real-Time ซึ่งมีรายละเอียดของ Code ดังนี้

1. ระบบเซ็นเซอร์ทั้งหมด

ใช้ Arduino MEGA2560 กับระบบเซ็นเซอร์ทั้งหมด ได้แก่ ระบบเซ็นเซอร์ตรวจจับตำแหน่ง และระบบเซ็นเซอร์วัดมุมก้ม-มุมเงย

```
// ++++++++++++++++++++++ GPS ++++++++++++ //
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>

// ++++++++++++++++++++++ Temp ++++++++++ //
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2
String Temp,temp;

// ++++++++++++++++++++++ ADXL345 ++++++++++ //
#include <Wire.h>
///#include <ADXL345.h> // roll pitch old version
#include <Adafruit_ADXL345_U.h>
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

// ++++++++++++++++++++++ TFT LCD ++++++++++ //
#include <UTFT.h>

// ++++++++++++++++++++++ Notice Fn'  ++++++++ //
SoftwareSerial GPSModule(10, 11); // TX, RX GPS
OneWire oneWire(ONE_WIRE_BUS);  // Temp
DallasTemperature sensors(&oneWire); // Temp
// TFT LCD
// Declare which fonts we will be using
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];
UTFT myGLCD(CTE32HR,38,39,40,41);   // Remember to change the model
//parameter to suit your display module!

// ++++++++++++++++++++++ variable ++++++++++ //
// -----GPS-------- //
int updates;
int failedUpdates;
int pos;

int stringplace = 0;
int de = 0;
double pitch, roll, fXg, fYg, fZg;
```

```
String timeUp ;
String nmea[15];
String labels[12] {"Time: ", "Status: ", "Latitude: ", "Hemisphere:
", "Longitude: ", "Hemisphere: ", "Speed: ", "Track Angle: ", "Date:
"};
// -----ADXL345-------- //
#define DEVICE (0x53)    //ADXL345 device address
#define TO_READ (6)      //num of bytes we are going to read each time
(two bytes for each axis)
byte buff[TO_READ] ;     //6 bytes buffer for saving data read from
the device
char str[512];           //string buffer to transform data before
sending it to the serial port
String Pitch,Roll;
/**************************************/

// +++++++++++++++++ Progrqam ++++++++++++ //
// ----------------- wire ----------------//
void writeTo(int device, byte address, byte val)
{
  Wire.beginTransmission(device); //start transmission to device
  Wire.write(address);        // send register address
  Wire.write(val);        // send value to write
  Wire.endTransmission(); //end transmission
}

// --------------- sub GPS ------------- //
    String ConvertLat() {
          String posneg = "";
          if (nmea[3] == "S") {
          posneg = "-";
          }
        String latfirst;
        float latsecond;
        for (int i = 0; i < nmea[2].length(); i++) {
          if (nmea[2].substring(i, i + 1) == ".") {
            latfirst = nmea[2].substring(0, i - 2);
            latsecond = nmea[2].substring(i - 2).toFloat();
          }
        }
        latsecond = latsecond / 60;
        String CalcLat = "";

        char charVal[9];
        dtostrf(latsecond, 4, 6, charVal);
        for (int i = 0; i < sizeof(charVal); i++) {
          CalcLat += charVal[i];
        }
        latfirst += CalcLat.substring(1);
        latfirst = posneg += latfirst;
        return latfirst;
    }

    String ConvertLng() {
        String posneg = "";
        if (nmea[5] == "W") {
            posneg = "-";
        }
```

```
        String lngfirst;
        float lngsecond;
        for (int i = 0; i < nmea[4].length(); i++) {
            if (nmea[4].substring(i, i + 1) == ".") {
                lngfirst = nmea[4].substring(0, i - 2);
                //Serial.println(lngfirst);
                lngsecond = nmea[4].substring(i - 2).toFloat();
                //Serial.println(lngsecond);
            }
        }
        lngsecond = lngsecond / 60;
        String CalcLng = "";
        char charVal[9];
        dtostrf(lngsecond, 4, 6, charVal);
        for (int i = 0; i < sizeof(charVal); i++) {
            CalcLng += charVal[i];
        }
        lngfirst += CalcLng.substring(1);
        lngfirst = posneg += lngfirst;
        return lngfirst;
    }

// ---------------- main ----------------- //
void setup() {
     Serial.begin(9600); //กำหนด serial 9600
    // GPS
     GPSModule.begin(9600);
    // Temp
     Serial.println("99% LOADING ...");
     Serial.println("by kazo oooooo ");
     sensors.begin();
    // ADXL345
      Wire.begin();
      Serial.begin(9600);
      delay(100);

      if (!accel.begin())
      {
            // Serial.println("Could not find a valid ADXL345
sensor, check wiring!");
            delay(500);
       }

      // Set measurement range
      // +/-  2G: ADXL345_RANGE_2G
      // +/-  4G: ADXL345_RANGE_4G
      // +/-  8G: ADXL345_RANGE_8G
      // +/- 16G: ADXL345_RANGE_16G
      accel.setRange(ADXL345_RANGE_16_G);
      accel.setDataRate(ADXL345_DATARATE_25_HZ);
    // TFT LCD
     myGLCD.InitLCD();
     myGLCD.clrScr();
}

void loop() {
    //GPS
```

```
        Serial.flush();
        GPSModule.flush();
        while (GPSModule.available() > 0)
         {
            GPSModule.read();
         }
        if (GPSModule.find("$GPRMC,")) {
           String tempMsg = GPSModule.readStringUntil('\n');
           for (int i = 0; i < tempMsg.length(); i++) {
               if (tempMsg.substring(i, i + 1) == ",") {
                   nmea[pos] = tempMsg.substring(stringplace, i);
                   stringplace = i + 1;
                   pos++;
                   }
               if (i == tempMsg.length() - 1) {
                   nmea[pos] = tempMsg.substring(stringplace, i);
                   }
           }
           updates++;
           nmea[2] = ConvertLat();
           nmea[4] = ConvertLng();
           for (int i = 0; i < 9; i++) {
              if (i==7) {
                 de++;
                 de=0;
              }
              else {
                 Serial.print(labels[i]);
                 Serial.print(nmea[i]);
                 Serial.println("");
                if (i==8) {
                 Serial.print("");
                 }
              }
           }

        }
        else {
            failedUpdates++;
        }
        stringplace = 0;
        pos = 0;

      // Temp
        sensors.requestTemperatures(); //Send the command to get
temperature readings
        Serial.print("Temperature : ");
        Serial.print(sensors.getTempCByIndex(0)); //  sensor 0
        temp = sensors.getTempCByIndex(0);
        Serial.print("\n");
        delay(1000);

      // Roll Pitch

        sensors_event_t event;
        accel.getEvent(&event);

        // Get x and y values from sensor
```

```
    int x = event.acceleration.x;
    int y = event.acceleration.y;
    int z = event.acceleration.z;


    // Calculate Pitch & Roll
    float pitch = -(atan2(x, sqrt(y*y + z*z))*180.0)/M_PI;
    float roll  = (atan2(y, z)*180.0)/M_PI;


    Serial.print(" Pitch = ");
    Serial.println(pitch);
    Serial.print(" Roll = ");
    Serial.println(roll);


    //we send the x y z values as a string to the serial port
    nmea[0] = "Time : "+nmea[0];
    nmea[1] = "Status : "+nmea[1];
    nmea[2] = "Latitude : "+nmea[2];
    nmea[3] = "Hemisphere : "+nmea[3];
    nmea[4] = "Longitude : "+nmea[4];
    nmea[5] = "Hemisphere : "+nmea[5];
    nmea[6] = "Speed : "+nmea[6];
    nmea[7] = "Date : "+nmea[7];
    Roll = "Elevation : "+ String(roll);
    Pitch = "Pitch : "+ String(pitch);
    Temp = "Temp : "+temp;
  //TFT LCD
     myGLCD.setColor(0, 100, 249);
     myGLCD.setBackColor(VGA_BLACK);
     myGLCD.fillScr(0,0,0);
     myGLCD.setFont(BigFont);
     myGLCD.print("99% Loading ...", CENTER, 10);
     myGLCD.setColor(255, 0, 0);
     myGLCD.print(nmea[0], LEFT, 50);
     myGLCD.print(nmea[1], LEFT, 90);
     myGLCD.print(nmea[2], LEFT, 130);
     myGLCD.print(nmea[3], LEFT, 170);
     myGLCD.print(nmea[4], LEFT, 210);
     myGLCD.print(nmea[5], LEFT, 250);
     //myGLCD.print(nmea[6], LEFT, 189);
     //myGLCD.print(nmea[7], LEFT, 231);
     //myGLCD.print(Temp, LEFT, 216);
     myGLCD.print(Roll, LEFT, 290);
     //myGLCD.print(Pitch, LEFT, 297);

}
```

2. ระบบมอเตอร์เคลื่อนที่ 2 แกนสำหรับติดตาม CanSat

ระบบมอเตอร์นี้จะใช้ร่วมกับสายอากาศยากิ-อูดะ โดยมี Arduino UNO เป็นตัวควบคุมการทำงานของมอเตอร์ ซึ่งจะทำงานร่วมกันทั้ง 2 แกน คือ แกน X และ แกน Y

1) การเคลื่อนที่ในแนวแกน X

```
//กำหนด pin ของมอเตอร์แกน X
#define stepPinx  11
#define dirPinx  12
#define enPinx 13

char incomingdata; //กำหนดตัวแปรในการรับข้อมูลจาก cansat
String data; //ใช้เปลี่ยนรูปแบบข้อมูลจาก char เป็น string
String latcan; //ใช้รับข้อมูล latitude
String loncan; //ใช้รับข้อมูล longitude
int z = 0; //ใช้เช็คตำแหน่ง <

double pi = 3.1452; //ค่า pi
double Longnd = 100.776309; //ค่า longitude ของดึกภาค
double Latgnd = 13.727528; //ค่า latitude ของดึกภาค
int R = 6373; //ค่ารัศมีโลก
int calibrate = 0; //ระดับน้ำทะเล

double Latcan; //ข้อมูล latitude
double Loncan; //ข้อมูล longitude
double dlon; //ระยะห่าง longitude ของ cansat กับ ground station
double dlat; //ระยะห่าง latitude ของ cansat กับ ground station

//ตัวแปรที่ใช้ในการคำนวณ angleX
double X1;
double Y1;

double angleX; //ค่ามุมในแนวแกนตั้ง
double angleX1;
double angXold = 0;

void setup() {
  //เซตโหมดให้มอเตอร์ทำงาน
  pinMode(stepPinx,OUTPUT);
  pinMode(dirPinx,OUTPUT);
  pinMode(enPinx,OUTPUT);
  digitalWrite(enPinx,LOW);

  Serial.begin(9600); //เริ่มการเชื่อมต่อ xbee กับ cansat

}

void loop() {

   if(Serial.available()==0){
    String data = Serial.readStringUntil("\n"); //รับข้อมูลจาก xbee
    Serial.print(data);

    String alt; //ใช้รับข้อมูลความสูง
    String latcan; //ใช้รับข้อมูล latitude
    String loncan; //ใช้รับข้อมูล longitude

    //แยกข้อมูลที่ได้รับ alt lat lon
    for(int x = 0;x < data.length();x++){
```

```
  if(data[x]=='<'){
    z++;
     if(z==14){
  for(int y = x+1;data[y]!='>';y++){
  double latcan = latcan + data[y];
    }
  }
    else if(z==15){
  for(int y = x+1;data[y]!='>';y++){
  loncan = loncan + data[y];
    }
  }
  else if(z==4){
  for(int y = x+1;data[y]!='>';y++){
  alt = alt + data[y];
    }
  }
  }
}
 z = 0;
    //แปลง lat lon เป็น float
    Latcan = (latcan.toDouble())*pi/180;
    Loncan = (loncan.toDouble())*pi/180;
    //คำนวณระยะห่าง latitude longitude
    dlon = Loncan-Longnd;
    dlat = Latcan-Latgnd;
    //คำนวณมุมในแนวแกนนอน
    X1 = cos(Latcan)*sin(dlon);
    Y1 = cos(Latgnd)*sin(Latcan)-sin(Latgnd)*cos(Latcan)*cos((dlon));
    angleX = (atan2(X1,Y1)*180)/pi; //มุมในแนวแกนนอนในเวลานั้นๆ


     if(angleX < angXold){ //ใช้เช็คว่าต้องหมุนซ้ายหรือหมุนขวา
       digitalWrite(dirPinx,HIGH);
       angleX1 = (angXold-angleX);
     }

    else if(angleX >= angXold){
      digitalWrite(dirPinx,LOW);
      angleX1 = (angleX-angXold);
      }

   for(float x = 0 ; x < angleX1 ; x+=0.1125) //ใช้วนลูปเพื่อให้มอเตอร์หมุน
    {
    digitalWrite(stepPinx,HIGH);
    delayMicroseconds(60000);
    digitalWrite(stepPinx,LOW);
    delayMicroseconds(60000);

      }
      Serial.println(angleX);
      angXold = angleX; //ใช้เก็บค่าตัวแปร

     }

     else
     {
```

```
      Serial.println("Communication failed"); //ถ้าไม่สามารถสื่อสารกับ xbee ใน
cansat ได้ โชว์ Communication failed
      }
      }
```

2) การเคลื่อนที่ในแนวแกน Y

```
//กำหนด library ที่จำเป็น
#include <Wire.h>
#include <Adafruit_ADXL345_U.h>

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345); //
setting พื้นฐานของเซนเซอร์

//กำหนด pin ของมอเตอร์แกน Y
#define stepPiny  9
#define dirPiny  7
#define enPiny 8

String data; //กำหนดตัวแปรในการรับข้อมูลจาก cansat
int z = 0; //

double pi = 3.1452; //ค่า pi
double Longnd = 100.776309; //ค่า longitude ของตึกภาค
double Latgnd = 13.727528; //ค่า latitude ของตึกภาค
int R = 6373; //ค่ารัศมีโลก
int calibrate = 0; //ระดับน้ำทะเล
double Latcan; //ข้อมูล latitude

double Loncan; //ข้อมูล longitude
double dlon; //ระยะห่าง longitude ของ cansat กับ ground station
double dlat; //ระยะห่าง latitude ของ cansat กับ ground station

//ตัวแปรที่ใช้ในการคำนวณ angleY
double a;
double c;
double d;
double Alt; //ข้อมูลความสูง

double angleY; //ค่ามุมในแนวแกนตั้ง
double angleY1; //ค่าความต่างมุมในปัจจุบันกับมุมที่เพิ่งผ่านมา
double angYold = 0;  //เก็บค่าตัวแปร angelY

double xxx; //ระยะในแนวแกน x จากสถานีภาคพื้นดินถึง cansat
double zeta; //มุมจากพื้นถึง cansat
double t1; //delay ในช่วงที่สูงจากพื้นมากกว่า 450 m
double t2; //delay ในช่วงที่สูงจากพื้นต่ำกว่า 450 m
int De = 0;//ใช้ในการรันรอบแรก


void setup() {
  //เซตโหมดให้มอเตอร์ทำงาน
  pinMode(stepPiny,OUTPUT);
```

```
  pinMode(dirPiny,OUTPUT);
  pinMode(enPiny,OUTPUT);
  digitalWrite(enPiny,LOW);

  Serial.begin(9600); // เริ่มการเชื่อมต่อ xbee กับ cansat
  if (!accel.begin())
  {
    delay(500);
  }
  Wire.begin();//เริ่มต้นการทำงานของเซนเซอร์
  //setting พื้นฐานของเซนเซอร์
  accel.setRange(ADXL345_RANGE_16_G);
  accel.setDataRate(ADXL345_DATARATE_25_HZ);

}


void loop() {

    if(Serial.available()==0){
    String data = Serial.readStringUntil("\n"); //รับข้อมูลจาก xbee
    Serial.print(data);

    String alt; //ใช้รับข้อมูลความสูง
    String latcan; //ใช้รับข้อมูล latitude
    String loncan; //ใช้รับข้อมูล longitude

    //แยกข้อมูลที่ได้รับ alt lat lon
    for(int x = 0;x < data.length();x++){
  if(data[x]=='<'){
    z++;
     if(z==14){
  for(int y = x+1;data[y]!='>';y++){
  double latcan = latcan + data[y];
    }
  }
    else if(z==15){
  for(int y = x+1;data[y]!='>';y++){
  loncan = loncan + data[y];
    }
  }
  else if(z==4){
  for(int y = x+1;data[y]!='>';y++){
  alt = alt + data[y];
    }
  }
  }
}
    z = 0; //รีเซ็ตข้อมูลที่แยกแล้ว
    //แปลง alt lat lon เป็น float
    Alt = alt.toDouble();
    Latcan = (latcan.toDouble())*pi/180;
    Loncan = (loncan.toDouble())*pi/180;
    //คำนวณระยะห่าง latitude longitude
    dlon = Loncan-Longnd;
    dlat = Latcan-Latgnd;
    //คำนวณมุมในแนวแกนตั้ง
```

```
    a = (sin(dlat/2))*(sin(dlat/2)) + cos(Latgnd) * cos(Latcan) *
(sin(dlon/2))*(sin(dlon/2));
    c = 2 * atan2( sqrt(a), sqrt(1-a) );
    d = R*c*1000;
    Alt = Alt - calibrate;
    angleY = (atan(Alt/d)*180)/pi; //มุมในแนวแกนดั่งในเวลานั้นๆ

    Serial.println(angleY);

     if(angleY < angYold){ //ใช้เช็คว่าต้องหมุนซ้ายหรือหมุนขวา
       digitalWrite(dirPiny,HIGH);
       angleY1 = (angYold-angleY);
     }

    else if(angleY >= angYold){
      digitalWrite(dirPiny,LOW);
      angleY1 = (angleY-angYold);
      }
      sensors_event_t event;
      accel.getEvent(&event);

// เก็บค่า x y z ที่ได้จากเซนเซอร์ไว้ในตัวแปร
      int xx = event.acceleration.xx;
      int yy = event.acceleration.yy;
      int zz = event.acceleration.zz;


  double roll  = (atan2(yy, zz)*180.0)/M_PI;//สูตรคำนวณหามุม roll ของเซนเซอร์

    //ต้องการให้รันแค่รอบแรกรอบเดียว

    if(De == 0)
    {
    //การคำนวณหา delay ที่เหมาะสมในแต่ละช่วงของการปล่อย cansat
    Altmax = Alt;
    xxx = 700/tan(roll*(3.14159265/180));
    zeta = atan((450/xxx))*(180/3.14159265) ;
    t1 = ((((Altmax-450)/20)/(roll-zeta/0.1125))/2)*1000000;
    t2 = ((45/(zeta/0.1125))/2)*1000000;
    De++;
    }

    //หมุนมอเตอร์ช่วงที่ cansat อยู่สูงจากพื้นมากกว่า 450 m
    if(alt >= 450){

   for(double x = 0 ; x < angleY1 ; x+=0.1125) //ใช้วนลูปเพื่อให้มอเตอร์หมุน
    {
    digitalWrite(stepPiny,HIGH);
    delayMicroseconds(t1);
    digitalWrite(stepPiny,LOW);
    delayMicroseconds(t1);
    }
      }
      else
      {
    for(double x = 0 ; x < angleY1 ; x+=0.1125) //ใช้วนลูปเพื่อให้มอเตอร์หมุน
    {
```

```
    digitalWrite(stepPiny,HIGH);
    delayMicroseconds(t2);
    digitalWrite(stepPiny,LOW);
    delayMicroseconds(t2);
    }
      }

      angYold = angleY;

     }
     else
     {
      Serial.println("Communication failed"); //ถ้าไม่สามารถสื่อสารกับ xbee ใน
cansat ได้ โชว์ Communication failed
     }
     }
```

3. ระบบมอเตอร์เคลื่อนที่ 2 แกนสำหรับติดตาม CubeSat

ระบบมอเตอร์นี้จะใช้ร่วมกับสายอากาศ Dual Band Handheld โดยมี Arduino UNO เป็นตัวควบคุมการทำงานของมอเตอร์ ซึ่งจะทำงานร่วมกันทั้ง 2 แกน คือ แกน X และ แกน Y

1) การเคลื่อนที่ในแนวแกน X

```
//กำหนด pin ของมอเตอร์แกน X
#define stepPinx  11
#define dirPinx  12
#define enPinx 13


double Azimuth; //ค่ามุมในแนวแกนตั้ง
double Azimuth1;
double Azimuthold = 0;

double Elevation;

double StartAz;
double EndAz;

double Starttime;
double Endtime;
double Time;

double Az;
double Delay;

int de = 0;

void setup() {
  //เซตโหมดให้มอเตอร์ทำงาน
  pinMode(stepPinx,OUTPUT);
  pinMode(dirPinx,OUTPUT);
```

```
  pinMode(enPinx,OUTPUT);
  digitalWrite(enPinx,LOW);


  Serial.begin(9600); // เริ่มการเชื่อมต่อ xbee กับ cansat


}
void loop() {
  // put your main code here, to run repeatedly:\String elevation;
    String elevation;
    String azimuth;
    String startAz;
    String endAz;
    String starttime;
    String endtime;
if(Serial.available()>0){
    String data = Serial.readStringUntil("*"); //รับข้อมูลจาก xbee
    Serial.println(data);
int z = 0;
    int k = 0;
    int kk = 0;
    int kkk = 0;
    int kkkk = 0;
    int kkkkk = 0;
    int kkkkkk = 0;
    for(int x = 0; x<data.length();x++)
    {
      if(data[x]!=' ' && data[x]!= ','){

       if(z == 0){
         for(int y=0;data[x+y]!= ',';y++ ){
         elevation = elevation + data[x+y];
         k = y;

       }

       if(data[x+k+1] = ','){
         z++;
       }
       x=x+k;
       //Serial.println(data[10]);
      }


      else if(z == 1){
       //Serial.println(x);
        for(int yy = 0;data[x+yy]!= ',' ;yy++ ){
         azimuth = azimuth + data[x+yy];
         kk = yy;
        // Serial.println(kk);
       }

       if(data[x+kk+1] = ','){
         z++;
       }
       x=x+kk;
```

```
      /*Serial.println(z);
      Serial.print(elevation);
      Serial.print(azimuth);*/
      //Serial.println(kk);
     }
    else if(z == 2){
      //Serial.println(x);
        for(int yyy=0;data[x+yyy]!= ',';yyy++ ){
        startAz = startAz + data[x+yyy];
        kkk = yyy;
      }

      if(data[x+kkk+1] = ','){
        z++;
      }
      x=x+kkk;
     }
    else if(z == 3){
        for(int yyyy=0;data[x+yyyy]!= ',';yyyy++ ){
       starttime = starttime  + data[x+yyyy];
         kkkk = yyyy;
      }

      if(data[x+kkkk+1] = ','){
        z++;
      }
      x=x+kkkk;
     }
  else if(z == 4){
      for(int yyyyy=0;data[x+yyyyy]!= ',';yyyyy++ ){
      endAz = endAz + data[x+yyyyy];
      kkkkk = yyyyy;
      }

      if(data[x+kkkkk+1] = ','){
        z++;
      }
      x=x+kkkkk;
     }
     else if(z == 5) {
        for(int yyyyyy = 0;data[x+yyyyyy]!= ' ';yyyyyy++ ){
        endtime = endtime + data[x+yyyyyy];
        kkkkkk = yyyyyy;
      }

      if(data[x+kkkkkk+2] = '*'){
        z++;
      }
      x=x+k;
     }
}
}
Serial.println("Elevation = " + String(elevation));
Serial.println("Azimuth = " + String(azimuth));
Serial.println("StartAz = " + String(startAz));
Serial.println("EndAz = "+ String(endAz));
Serial.println("Starttime = "+ String(starttime));
Serial.println("Endtime = "+ String(endtime));
```

```
 double Elevation = elevation.toDouble();
 double Azimuth = azimuth.toDouble();
 double StartAz =  startAz.toDouble();
 double EndAz =  endAz.toDouble();
 double Starttime = starttime.toDouble();
 double Endtime = endtime.toDouble();
 /*Serial.println(Elevation);
 Serial.println(Azimuth);
 Serial.println(StartAz);
 Serial.println(EndAz);
 Serial.println(starttime);
 Serial.println(endtime);
 Serial.println(Starttime);
 Serial.println(Endtime);*/



  if(Azimuth < Azimuthold){ //ใช้เช็คว่าต้องหมุนซ้ายหรือหมุนขวา
    digitalWrite(dirPinx,HIGH);
    Azimuth1 = (Azimuthold-Azimuth);
  }

 else if(Azimuth >= Azimuthold){
  digitalWrite(dirPinx,LOW);
  Azimuth1 = (Azimuth-Azimuthold);
  }

 Time = Endtime - Starttime;
 //Serial.println(Time);
 if(StartAz > EndAz){
  Az = StartAz-EndAz;
 }
 else{
  Az = EndAz-StartAz;
 }
  Delay = (1/((Az/0.1125)/Time)/2)*(1000000);

if(Azimuth == StartAz && de == 0){

for(double x = 0 ; x < Azimuth1 ; x+=0.1125) //ตอนมอเตอร์เริ่มหมุน
 {
 digitalWrite(stepPinx,HIGH);
 delayMicroseconds(1000);
 digitalWrite(stepPinx,LOW);
 delayMicroseconds(1000);
 }
 de++;
}
else if(de == 1){

 for(double x = 0 ; x < Azimuth1 ; x+=0.1125) //ตอนที่ต้องการความละอียด
 {
 digitalWrite(stepPinx,HIGH);
 delayMicroseconds(Delay);
 digitalWrite(stepPinx,LOW);
 delayMicroseconds(Delay);
 }
 if(Azimuth == EndAz){
```

```
        de == 2;
        }
      }
        Azimuthold = Azimuth;

       }
       /*else
       {
        Serial.println("Communication failed"); //ถ้าไม่สามารถสื่อสารกับ xbee ใน
cansat ได้ โชว์ Communication failed
      }*/

}
```

2) การเคลื่อนที่ในแนวแกน Y

```
//กำหนด pin ของมอเตอร์แกน Y
#define stepPiny  9
#define dirPiny  7
#define enPiny 8

double Elevation; //ค่ามุมในแนวแกนตั้ง
double Elevation1; //ค่าความต่างมุมในปัจจุบันกับมุมที่เพิ่งผ่านมา
double Elevationold = 0; //เก็บค่าตัวแปร Elevation

void setup() {
  //เซตโหมดให้มอเตอร์ทำงาน
  pinMode(stepPiny,OUTPUT);
  pinMode(dirPiny,OUTPUT);
  pinMode(enPiny,OUTPUT);
  digitalWrite(enPiny,LOW);

  Serial.begin(9600); // เริ่มการเชื่อมต่อ xbee กับ cansat
}

void loop() {

    if(Serial.available()==0){
    String data = Serial.readStringUntil("\n"); //รับข้อมูลจาก xbee
    Serial.print(data);

    //รับข้อมูลจาก python
    String elevation;
    String MaxEl;

    //แปลงข้อมูลจาก string เป็น python
    Elevation = elevation.toDouble();
    MaxEl =  MaxEl.toDouble();
    Serial.println(Elevation);

     if(Elevation < Elevationold){ //ใช้เช็คว่าต้องหมุนซ้ายหรือหมุนขวา
       digitalWrite(dirPiny,HIGH);
       Elevation1 = (Elevationold-Elevation);
     }
```

```
    else if(Elevation >= Elevationold){
      digitalWrite(dirPiny,LOW);
      Elevation1 = (Elevation-Elevationold);
      }

   for(double x = 0 ; x < Elevation1 ; x+=0.1125) //ใช้วนลูปเพื่อให้มอเตอร์หมุน
    {
    digitalWrite(stepPiny,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPiny,LOW);
    delayMicroseconds(1000);
    }
      Elevationold = Elevation;//ใช้เก็บค่าตัวแปร
     }
     }
```

4. ระบบควบคุม Joystick เพื่อควบคุมการเคลื่อนที่ของมอเตอร์

```
#define stepPin1  9 //กำหนดขา pul แกน y
#define dirPin1  7 //กำหนดขา dir แกน y
#define enPin1 8 //กำหนดขา en แกน y
#define stepPin2  11 //กำหนดขา pul แกน x
#define dirPin2  12 //กำหนดขา dir แกน x
#define enPin2 13 //กำหนดขา en แกน x

void setup() {
pinMode(2, INPUT_PULLUP); //กำหนดขา joystick แกน y ด้านทวนเข็มนาฬิกา หมุนลง
pinMode(3, INPUT_PULLUP); //กำหนดขา joystick แกน x ด้านทวนเข็มนาฬิกา หมุนขวา
pinMode(4, INPUT_PULLUP); //กำหนดขา joystick แกน y ด้านตามเข็มนาฬิกา หมุนขึ้น
pinMode(5, INPUT_PULLUP); //กำหนดขา joystick แกน x ด้านตามเข็มนาฬิกา หมุนซ้าย

//กำหนดสถานะขาต่างๆที่ใช้เป็น OUTPUT
pinMode(stepPin1,OUTPUT);
pinMode(dirPin1,OUTPUT);
pinMode(enPin1,OUTPUT);
pinMode(stepPin2,OUTPUT);
pinMode(dirPin2,OUTPUT);
pinMode(enPin2,OUTPUT);

//กำหนดขา en ให้เป็น LOW
digitalWrite(enPin1,LOW);
digitalWrite(enPin2,LOW);
}

 void loop() {

// ใช้buttonStateมารับสถานะที่ขาต่างๆ
int buttonState2 = digitalRead(2);
int buttonState3 = digitalRead(3);
int buttonState4 = digitalRead(4);
int buttonState5 = digitalRead(5);
```

```
//แกน y ด้านทวนเข็มนาฬิกา หมุนลง
if (buttonState2 == LOW) {
    digitalWrite(dirPin1,HIGH);
    //ลูปเพื่อให้ให้มอเตอร์หมุน
    for(int x =0 ;x<=0; x++) {
    digitalWrite(stepPin1,HIGH);
    delayMicroseconds(1500);
    digitalWrite(stepPin1,LOW);
    delayMicroseconds(1500);
    //คำสั่งออกจากลูปเมื่อเปลี่ยนทิศที่หมุน
    if (buttonState2 == HIGH)
    {
      x = -1;
    }
    }
}
//แกน y ด้านตามเข็มนาฬิกา หมุนขึ้น
else if (buttonState4 == LOW) {
    digitalWrite(dirPin1,LOW);
    //ลูปเพื่อให้ให้มอเตอร์หมุน
    for( int x=0 ;x<=0; x++) {
    digitalWrite(stepPin1,HIGH);
    delayMicroseconds(1500);
    digitalWrite(stepPin1,LOW);
    delayMicroseconds(1500);
    //คำสั่งออกจากลูปเมื่อเปลี่ยนทิศที่หมุน
    if (buttonState4 == HIGH)
    {
      x = -1;
    }
    }
}
//แกน x ด้านทวนเข็มนาฬิกา หมุนขวา
else if (buttonState3 == LOW) {
    digitalWrite(dirPin2,HIGH);
    //ลูปเพื่อให้ให้มอเตอร์หมุน
    for(int x = 0;x<=0; x++) {
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(1500);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(1500);
    //คำสั่งออกจากลูปเมื่อเปลี่ยนทิศที่หมุน
    if (buttonState3 == HIGH)
    {
      x = -1;
    }
    }
}
//แกน x ด้านตามเข็มนาฬิกา หมุนซ้าย
else if (buttonState5 == LOW) {
    //ลูปเพื่อให้ให้มอเตอร์หมุน
    digitalWrite(dirPin2,LOW);
    for(int x = 0;x<=0; x++) {
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(1500);
```

```
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(1500);
    //คำสั่งออกจากลูปเมื่อเปลี่ยนทิศที่หมุน
    if (buttonState5 == HIGH)
    {
      x = -1;
    }
    }
}
}
```

5. การดึงข้อมูลการเคลื่อนที่ของ CubeSat แบบ Real-Time

เราได้ทำการดึงข้อมูลต่าง ๆ ของ CubeSat จากเว็บไซต์ N2YO.com ซึ่งจะต้องมี Code Arduino เป็นตัวกลางก่อนจะเขียนภาษา Python โดย Code มีดังนี้

```
#include<SoftwareSerial.h>
void setup() {

  Serial.begin(9600); //เริ่มต้นการสื่อสารด้วย baudrate 9600

}

void loop() {
 //ใช้รับข้อมูลจาก python แล้วส่งข้อมูลผ่าน port TX RX ไปยัง arduino อีกรอบ
  if(Serial.available() > 0){
    String data = Serial.readStringUntil("*"); //เก็บข้อมูลตั้งแต่แรกจนถึง *
    Serial.print(data);
  }
}
```

จากนั้นจึงทำการดึงข้อมูลโดยใช้โปรแกรม Python ซึ่งมี Code ดังนี้

```
 # กำหนด library ที่จำเป็น
import serial
import requests
y=1;
send = serial.Serial('COM15',9600)# กำหนด com และ baudrate ให้สัมพันธ์กับใน arduino
while y<=1: # ใช้วนลูปแบบไม่หยุด
    # ตัวแปรต่างๆที่ใช้ในการเก็บข้อมูล
    satname = '';
    satid = '';
    transactioncount = '';
    latitude = '';
    longtitude = '';
    altitude = '';
    azimuth = '';
    elevation = '';
    ra = '';
```

```
    dec = '';
    timestamp = '';
    infosat = '';
    bigdata = '';
    passes = '';
    startAZ = '';
    endAZ = '';
    startUTC = '';
    maxAZ = '';
    maxEl = '';
    maxUTC = '';
    endUTC = '';
    app = '';
    sa = [];
    # url ที่มีข้อมูลที่ต้องการ
    url2 =
"https://www.n2yo.com/rest/v1/satellite/radiopasses/28654/13.727456/100.77630/0/1/30/&a
piKey=PSVVUY-V5QZ3V-D4L47K-3UEW"
    # ดึงข้อมูลที่มีอยู่ใน url ด้วย request
    data2 = requests.get(url2)

    # เก็บข้อมูล data ใน sa
    aa = 1;
    for n in data2.text:
        while aa >= 1:
            if data2.text[aa] == '"':
                sa.append(aa);
                aa = aa + 1;
            aa = aa + 1;
            if aa == len(data2.text):
                aa = 0;
    aa = 1;
    bb = 4;
    # แยกข้อมูลโดยแยกส่วนแรกซึ่งคือ satname ก่อน
    while aa >= 1:
        if data2.text[int(sa[bb]) + aa] == '"':
            aa = 0;
        else:
            satname = satname + data2.text[int(sa[bb]) + aa]
            aa = aa + 1;
    aa = 1;
    # แยกข้อมูลเฉพาะส่วนที่ใช้รวมไว้ใน infosat
    while bb < len(sa):
        while aa >= 1:
            if data2.text[int(sa[bb + 1]) + aa + 1] == '[' or
data2.text[int(sa[bb + 1]) + aa + 1] == '"' or data2.text[
                int(sa[bb + 1]) + aa + 1] == '}':
                aa = 0;
            else:

                infosat = infosat + data2.text[int(sa[bb + 1]) + aa + 1]
```

```
                aa = aa + 1;

        infosat = infosat + ' '
        bb = bb + 2;
        aa = 1;
    infosat = infosat + ','

    gg = 0;
    ss = 1;
    kk = 0;

    # แยกข้อมูลใน infosat ให้ตรงตามที่เราต้องการ
    while ss <= 10:
        while gg >= 0:
            if infosat[gg] == ',' or (infosat[gg] == infosat[gg + 1]==
infosat[gg + 2]==' ' and ss!=6)or gg == len(infosat)  :
                kk = gg;
                gg = -1;
            elif ss == 1:
                transactioncount  = transactioncount + infosat[gg]
                gg = gg + 1;
            elif ss == 2:
                passes  = passes  + infosat[gg]
                gg = gg + 1;
            elif ss == 3:
                startAZ = startAZ + infosat[gg]
                gg = gg + 1;
            elif ss == 4:
                startUTC = startUTC + infosat[gg]
                gg = gg + 1;
            elif ss == 5:
                maxAZ = maxAZ + infosat[gg]
                gg = gg + 1;
            elif ss == 6:
                maxEl = maxEl + infosat[gg]
                gg = gg + 1;
            elif ss == 7:
                maxUTC = maxUTC + infosat[gg]
                gg = gg + 1;
            elif ss == 8:
                endAZ = endAZ + infosat[gg]
                gg = gg + 1;
            else:
                endUTC = endUTC + infosat[gg]
                gg = gg + 1;
        ss = ss + 1;
        gg = kk + 1;

    # นำข้อมูลที่แยกได้แล้วมาแยกอีกทีเพื่อให้สมบูรณ์ยิ่งขึ้น
    passes = passes + '*'
    ee = 0;
```

```
    while ee>=0:
        if ee < len(passes)-1:
            if passes[ee] != ' ' and passes[ee+1]!=' ' :
                startAZ = startAZ + passes[ee]
                ee +=1

            elif ee == len(passes):
                ee = -1;
            else:
                ee+=1
        else:
            ee = -1;

    # สร้างตัวแปรใหม่เพื่อให้ข้อมูลไม่ทับกัน
    satname = '';
    infosat = '';
    sa = [];
    # url ที่มีข้อมูลที่ต้องการ
    url =
"https://www.n2yo.com/rest/v1/satellite/positions/28654/13.727456/100.776309/0/1/&api
Key=PSVVUY-V5QZ3V-D4L47K-3UEW"
    data = requests.get(url)# ดึงข้อมูลด้วยคำสั่ง request

    # เก็บ data ไว้ในตัวแปร sa
    a = 1;
    for n in data.text:
        while a >= 1:
            if data.text[a] == '"':
                sa.append(a);
                a = a + 1;
            a = a + 1;
            if a == len(data.text):
                a = 0;
    a = 1;
    b = 4;
    # แยกข้อมูลโดยแยกส่วนแรกก่อน คือ satname
    while a >= 1:
        if data.text[int(sa[b]) + a] == '"':
            a = 0;
        else:
            satname = satname + data.text[int(sa[b]) + a]
            a = a + 1;
    a = 1;
    # แยกข้อมูลส่วนที่ใช้ไว้ใน infosat
    while b < len(sa)-1:
        while a >= 1:
            if data.text[int(sa[b + 1])+a + 1]=='[' or
data.text[int(sa[b + 1])+a + 1]=='"' or data.text[int(sa[b + 1])+a + 1]==
'}':
                a = 0;
            else:
```

```
                infosat = infosat + data.text[int(sa[b + 1])+a + 1]
                a = a + 1;

        infosat = infosat + ' '
        b = b + 2;
        a = 1;
    infosat = infosat + ' '

    g = 0;
    s = 1;
    k = 0;
    # แยกข้อมูลใน infosat ให้ตรงตามที่เราต้องการ
    while s <= 10:
        while g >= 0:
            if infosat[g] == ',' or infosat[g] == infosat[g + 1]==' '
or g == len(infosat):
                k = g;
                g = -1;
            elif s == 1:
                satid = satid + infosat[g]
                g = g + 1;
            elif s == 2:
                transactioncount = transactioncount + infosat[g]
                g = g + 1;
            elif s == 3:
                latitude = latitude + infosat[g]
                g = g + 1;
            elif s == 4:
                longtitude = longtitude + infosat[g]
                g = g + 1;
            elif s == 5:
                altitude = altitude + infosat[g]
                g = g + 1;
            elif s == 6:
                azimuth = azimuth + infosat[g]
                g = g + 1;
            elif s == 7:
                elevation = elevation + infosat[g]
                g = g + 1;
            elif s == 8:
                ra = ra + infosat[g]
                g = g + 1;
            elif s == 9:
                dec = dec + infosat[g]
                g = g + 1;
            else:
                timestamp = timestamp + infosat[g]
                g = g + 1;
        s = s + 1;
        g = k + 1;
    # เก็บตัวแปรที่จำเป็นต้องใช้ในการหมุนไว้ใน bigdata
```

```
    bigdata =elevation +','+ azimuth +','+ startAZ+',' + startUTC+','
+ endAZ +','+ endUTC
    print(url)
    print(data.text)

    print("satname : %s" % satname)

    print("latitude : %s" % latitude)
    print("longitude : %s" % longtitude)
    print("altitude : %s" % altitude)
    print("azimuth : %s" % azimuth)
    print("elevation : %s" % elevation)
    print("ra : %s" % ra)
    print("dec : %s" % dec)
    print("timestamp : %s" % timestamp)

    print(url2)
    print(data2.text)

    print("startAZ : %s" % startAZ)
    print("startUTC : %s" % startUTC)
    print("maxAZ : %s" % maxAZ)
    print(" maxEl : %s" %  maxEl)
    print("maxUTC : %s" % maxUTC)
    print("endAZ : %s" % endAZ)
    print("endUTC : %s" % endUTC)

    send.write(bigdata.encode()) #ส่งข้อมูลไปยัง arduino
```