ระบบตรวจหาโอกาสในการเป็นสัดของโคนม
DAIRY ESTRUS DETECTION SYSTEM

โดย

นายศุภเทพ เอกอุดมชัย

นายศุภวิชญ์ ตั้งสมคิด

นายสมณะ นิชรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ระบบตรวจหาโอกาสในการเป็นสัดของโคนม

DAIRY ESTRUS DETECTION SYSTEM

โดย

นายศุภเทพ	เอกอุดมชัย	58011246
นายศุภวิชญ์	ตั้งสมคิด	58011254
นายสมณะ	นิชรัตน์	58011268

อาจารย์ที่ปรึกษา

ผศ.ดร.สิรภพ ตู้ประกาย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ผ่านการตรวจรูปเล่มแล้ว

(.....)

อาจารย์ที่ปรึกษา

22/5/62

ผ่านการตรวจชิ้นงานแล้ว

(.....)

กรรมการผู้ตรวจชิ้นงาน

22/05/62

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบตรวจหาโอกาสในการเป็นสัดของโคนม

DAIRY ESTRUS DETECTION SYSTEM

ผู้จัดทำ

1. นายศุภเทพ เอกอุดมชัย 58011246
2. นายศุภวิชญ์ ตั้งสมคิด 58011254
3. นายสมณะ นิชรัตน์ 58011268

..... อาจารย์ที่ปรึกษา

(ผศ.ดร.สิรภพ ตู้ประกาย)

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่อง “ระบบตรวจหาโอกาสในการเป็นสัดของโคนม” ไม่สามารถทำให้ลุล่วงสำเร็จไปได้ หากไม่ได้รับความช่วยเหลือ ความร่วมมือ และคำปรึกษา จากบุคคลเหล่านี้ผู้ซึ่งเป็นบุคคลที่สมควรได้รับการยกย่อง ผศ.ดร.สิรภพ ตู้ประกาย ผู้เป็นอาจารย์ที่ปรึกษาปริญญานิพนธ์นี้ ที่ชี้แนะแนวทางการวางแผน การแก้ปัญหาอันเป็นประโยชน์ในการศึกษาค้นคว้าวิจัย และให้ความช่วยเหลือทางด้านทุนทรัพย์เพื่อค้นคว้าวิจัยในครั้งนี้

ขอบพระคุณอาจารย์และเจ้าหน้าที่ประจำภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และโรงพยาบาลปศุสัตว์ คณะสัตวแพทยศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยที่ได้ประสิทธิ์ประสาทวิชา ความรู้ และอำนวยความสะดวกสถานที่ทำการวิจัย

ขอขอบคุณพระคุณ บิดา มารดา ครอบครัว และ นายเอกลักษณ์ เล็กเลิศศิริวงศ์ ผู้ให้คำปรึกษา และสนับสนุนการจัดหาอุปกรณ์ในการค้นคว้าวิจัยในครั้งนี้

นายศุภเทพ เอกอุดมชัย
นายศุภวิชญ์ ตั้งสมคิด
นายสมณะ นิชรัตน์
ผู้จัดทำ

ระบบตรวจหาโอกาสในการเป็นสัดของโคนม

DAIRY ESTRUS DETECTION SYSTEM

โดย นายศุภเทพ เอกอุดมชัย 58011246
นายศุภวิชญ์ ตั้งสมคิด 58011254
นายสมณะ นิชรัตน์ 58011268

อาจารย์ที่ปรึกษา ผศ.ดร.สิรภพ ตู้ประกาย

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เสนอการออกแบบระบบตรวจหาโอกาสในการเป็นสัดของโคนมด้วยวิธีวัดการเคลื่อนไหวของโคนม โดยแบ่งเป็น 3 ระบบ คือ ระบบเซนเซอร์วัดปริมาณการเคลื่อนไหวของโคนม ระบบการประมวลผลอัตราการเคลื่อนไหว และระบบส่งข้อมูลอัตราการเคลื่อนไหวผ่านเครือข่าย LoRaWAN ซึ่งข้อมูลอัตราการเคลื่อนไหวถูกจัดเก็บในฐานข้อมูล แสดงผลด้วยกราฟอัตราการเคลื่อนไหวเฉลี่ยต่อชั่วโมง และมีระบบการแจ้งเตือนการเป็นสัดของโคนม

ABSTRACT

This thesis proposes the design of a system for detecting estrous opportunities of dairy cows by measuring the movement of dairy cows, which is divided into 3 systems: sensor system for measuring the amount of movement of dairy cows, motion rate processing system and the movement rate data transmission system via the LoRaWAN network. Which the movement rate data is stored in the database, showing results with an average movement rate graph per hour and a system of estrus warning of dairy cows.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเทคโนโลยีเข้ามามีบทบาทในชีวิตประจำวันส่งผลให้มีความสะดวกมากขึ้น ซึ่งรวมไปถึงเทคโนโลยีที่ช่วยในการทำปศุสัตว์ในฟาร์มโคนม โดยแนวคิดของปริญญานิพนธ์นี้เกิดจากในฟาร์มโคนมปัจจุบันนั้นยังจำเป็นต้องพึ่งพาแรงงานคนเป็นหลักทั้งในด้านสุขภาพและผลผลิตของสัตว์ การตรวจการเป็นสัดของโคนมโดยใช้คนสังเกตนั้นมีความแม่นยำน้อยกว่าการใช้เทคโนโลยีเข้ามาช่วย ทางผู้จัดทำจึงเล็งเห็นประโยชน์และริเริ่มจัดทำปริญญานิพนธ์นี้ขึ้นมาเพื่อนำเทคโนโลยีเข้ามาทดแทนแรงงานคนเป็นการเพิ่มประสิทธิภาพและลดต้นทุนในการเลี้ยงโคนม โดยสร้างระบบตรวจหาโอกาสในการเป็นสัดของโคนมเพื่อทำการเก็บข้อมูล แล้วนำข้อมูลในแต่ละวันนั้น ส่งไปยังส่วนการเก็บบันทึกข้อมูล และข้อมูลบันทึกไว้นั้นจะถูกนำมาคำนวณตรวจหาโอกาสในการเป็นสัดของโคนมต่อไป

1.2 วัตถุประสงค์

1) เพื่อศึกษาและวิเคราะห์พฤติกรรมของโคนมที่จะทำให้ทราบถึงการเป็นสัดของโคนม
2) เพื่อประยุกต์ใช้ระบบตรวจหาโอกาสในการเป็นสัดของโคนม
3) เพื่อเข้าใจการใช้โปรแกรมในการออกแบบระบบตรวจหาโอกาสในการเป็นสัดของโคนม
4) เพื่อนำอุปกรณ์ Arduino MKR WAN 1300, โมดูล GY-521 และ LoRaWAN Gateway สร้างเป็นระบบตรวจหาโอกาสในการเป็นสัดของโคนม และศึกษาการเชื่อมต่อด้วยอุปกรณ์ LoRaWAN

1.3 ขอบเขตของปริญญานิพนธ์

ระบบตรวจหาโอกาสในการเป็นสัดของโคนมสามารถหาโอกาสในการเป็นสัดด้วยปริมาณการเคลื่อนไหวในแต่ละวัน และต้องสามารถส่งต่อข้อมูลผ่านเครือข่าย LoRaWAN เพื่อส่งข้อมูลไปเก็บยังส่วนการเก็บบันทึกข้อมูลที่ได้สร้างขึ้น เพื่อให้ผู้วิเคราะห์สามารถเข้าถึงข้อมูลปริมาณการเคลื่อนไหวในแต่ละวัน เพื่อทำการวิเคราะห์การเป็นสัดของโคนม

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ปริญญานิพนธ์เรื่อง “ระบบตรวจหาโอกาสในการเป็นสัดของโคนม” มีความจำเป็นที่ต้องศึกษาระบบที่เกี่ยวข้อง หลักการการทำงานของแต่ละอุปกรณ์ และการออกแบบอุปกรณ์เพื่อการประยุกต์ใช้งาน ดังนั้นปริญญานิพนธ์นี้จึงมีทฤษฎีและหลักการที่เกี่ยวข้องดังต่อไปนี้

2.1 โค

โค เป็นสัตว์เลี้ยงที่มีขนาดใหญ่ มีกีบเท้า เป็นสัตว์ที่โดดเด่นในวงศ์ย่อย Bovinae เป็นชนิดที่แพร่หลายที่สุดในสกุล Boss และถูกจำแนกเป็นกลุ่มอย่างกว้างขวางที่สุดว่า Boss primigenius โคเป็นสัตว์ลากเทียมที่ถูกเลี้ยงเป็นปศุสัตว์เพื่อเอาเนื้อ นม และผลิตภัณฑ์อื่น ๆ จากโค เช่น หนังและมูลเพื่อใช้เป็นปุ๋ยคอกหรือเชื้อเพลิง [1]

2.1.1 ประเภทของโค โคแบ่งตามประเภทของลักษณะทางผลผลิตที่เป็นประโยชน์ของประโยชน์ได้ดังนี้

1) โคเนื้อ

2) โคนม

3) โคเนื้อและนม

2.1.2 พันธุ์ของโค [2]

1) โคเนื้อ

โคพันธุ์ชาโรเลส์ (Charolais) เป็นโคสายพันธุ์ยุโรป มีแหล่งกำเนิดอยู่ทางตอนกลางของประเทศฝรั่งเศส โครงร่างเป็นรูปสี่เหลี่ยม ช่วงลำตัวยาวและแน่นทึบ กล้ามเนื้อส่วนสันหลัง ช่วงต้นขาและไหล่มีการเจริญดีมาก โคเพศผู้โตเต็มที่มีส่วนสูงเฉลี่ย 150-155 เซนติเมตร น้ำหนักประมาณ 1,100-1,200 กิโลกรัม โคเพศเมียโตเต็มที่มีส่วนสูงเฉลี่ย 140-143 เซนติเมตร น้ำหนักประมาณ 700-750 กิโลกรัม

โคพันธุ์อเมริกันบราห์มัน (American Brahman) เป็นโคเนื้อเมืองร้อนสายพันธุ์โคอินเดีย ได้รับการปรับปรุงพันธุ์จากประเทศอเมริกา ถ้าเลี้ยงเป็นโคขุนด้วยอาหารที่มีโปรตีนมากกว่า 16% จะสามารถโตได้วันละประมาณ 1 กิโลกรัมและเมื่อผสมพันธุ์กับโคพันธุ์ยุโรป เช่น ชาร์โรเลส์ ลูกผสมที่เกิดมาถ้าเลี้ยงด้วยอาหารที่มีโปรตีนมากกว่า 16% จะสามารถโตได้ถึงวันละ 1.3 กิโลกรัม เปอร์เซ็นต์ซาก 60% โคเพศผู้โตเต็มที่มีน้ำหนักประมาณ 800-1,200 กิโลกรัม โคเพศเมียมีตะโหนกขนาดเล็กกว่าโคเพศผู้ มีน้ำหนักประมาณ 550-800 กิโลกรัม

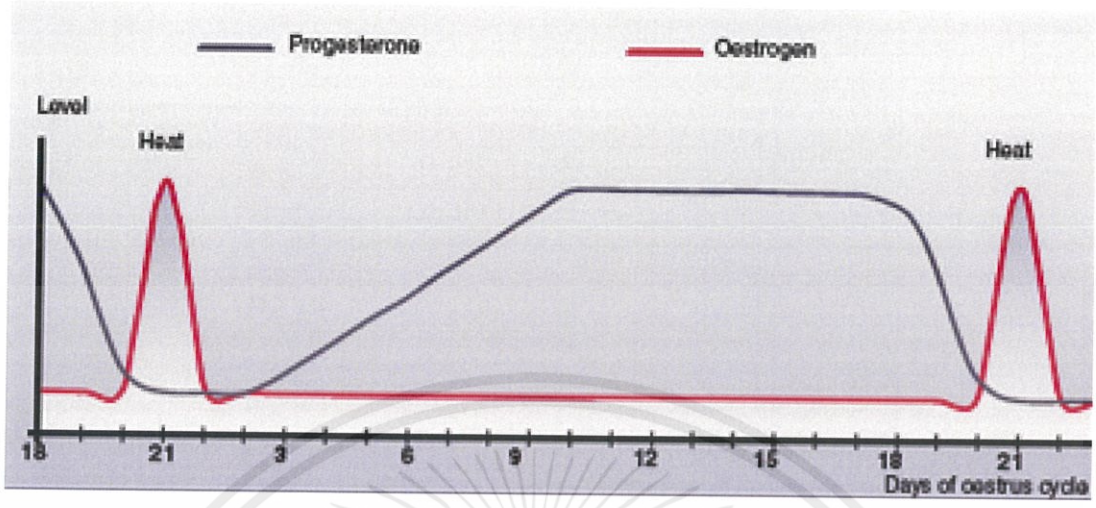
โคพันธุ์แองกัส(Angus) มีถิ่นกำเนิดอยู่ทางภาคตะวันออกเฉียงเหนือของประเทศสก๊อตแลนด์ เป็นโคขนาดกลางถึงเล็ก โคเพศผู้มีน้ำหนักประมาณ 900 กิโลกรัม โคเพศเมียมีน้ำหนักประมาณ 600 กิโลกรัม โคพันธุ์นี้จะมีสีดำตลอดตัว ไม่มีเขา ถึงวัยเจริญเร็ว มีข้อเสียคือเนื่องจากมีขนาดเล็กอัตราการเจริญเติบโตหลังหย่านมไม่ดีนัก พร้อมทั้งปรับตัวเข้ากับสภาพภูมิอากาศของประเทศไทย คือ ร้อนชื้นไม่ดี

2) โคนม

โคนมที่เลี้ยงอยู่ปัจจุบัน สามารถแบ่งออกเป็น 2 สายพันธุ์หลัก ได้แก่ โคยุโรป และโคอินเดีย ซึ่งโคทั้งสอง สายพันนี้ถือได้ว่าเป็นโคพันธุ์แท้ ในประเทศไทยได้นำเข้าโคทั้งสายพันธุ์ยุโรป และสายพันธุ์อินเดียมาเลี้ยง แต่ว่าสภาพภูมิอากาศ และปัจจัยอื่น ๆ ของเมืองไทยทำให้การเลี้ยงโคนอกไม่ประสบผลสำเร็จเท่าที่ควร จนปัจจุบันนี้ได้มีการคัดสายพันธุ์ โดยการผสมพันธุ์โคขึ้นมาใหม่เป็นโคนมไทยเอง

2.1.3 การเป็นสัด (Estrus) [3]

การเป็นสัด คือ ช่วงเวลาหนึ่งที่สัตว์เพศเมียยอมรับการผสมจากสัตว์เพศผู้แล้วมีการตกไข่ โดยพฤติกรรมการเป็นสัดอยู่ภายใต้อิทธิพลของฮอร์โมน ชนิดต่างหลายอย่างโดยฮอร์โมนที่เด่นชัดที่ส่งผลต่อพฤติกรรมกระวนกระวายคือ ฮอร์โมนเอสโตรเจน (Estrogen) ดังรูปที่ 2.1 นอกจากอาการยอมรับการผสมพันธุ์จากเพศผู้แล้ว ยังมีการเปลี่ยนแปลงทางสรีระวิทยาอื่น ๆ เช่น อวัยวะเพศบวมแดง มีเมือกไหล ร้องเสียงดัง



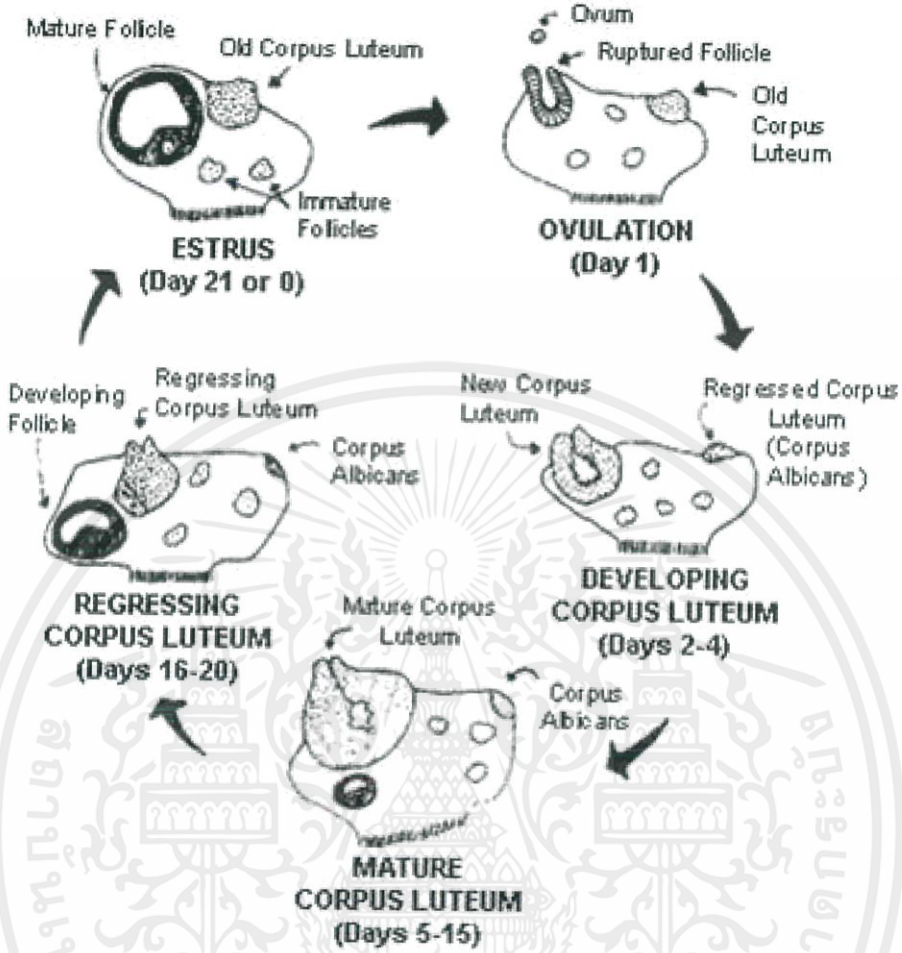
รูปที่ 2.1 ระยะการเป็นสัดของโคกับฮอร์โมนเอสโตรเจน [4]

การเป็นสัดของโคเพศเมีย จะเริ่มเป็นสัดเมื่อถึงวัยสาวหรือวัยเจริญพันธุ์ (Puberty) ในโคอายุที่ถึงวัยเจริญพันธุ์ประมาณ 7-18 เดือน (โดยเฉลี่ยประมาณ 10 เดือน) ขึ้นอยู่กับการเลี้ยงดู ความสมบูรณ์ของอาหาร ซึ่งทำให้โคเจริญพันธุ์ช้า (Delay Puberty) ปัจจัยที่มีผลมากต่อวัยเจริญพันธุ์คือ น้ำหนักตัวเมื่อเทียบกับน้ำหนักตัวที่โตเต็มที่ นอกจากนี้ยังขึ้นอยู่กับชนิดของโคแต่ละสายพันธุ์ด้วย

โดยปกติโคจะมีวงรอบการเป็นสัดเฉลี่ย 20-21 วัน (18-24 วัน) โคพันธุ์เมืองร้อน (Bos indicus) แสดงการเป็นสัดสั้น โดยเฉลี่ยแสดงอาการเป็นสัดประมาณ 11 ชั่วโมง และสัดเริ่มแสดงอาการในช่วงเย็นของวัน หลังจากการแสดงอาการเป็นสัดแล้วจะมีการตกไข่ (Ovulation) โดยประมาณ 25-26 ชั่วโมง ส่วนโคพันธุ์เมืองหนาว (Bos taurus) จะแสดงอาการเป็นสัดที่นานกว่าประมาณ 18 ชั่วโมง และมีการตกไข่ประมาณ 28-31 ชั่วโมงหลังการเป็นสัด

2.1.1.1 วงรอบการเป็นสัด

วงรอบการเป็นสัดแบ่งเป็น 4 ระยะ ตามลักษณะพฤติกรรมการเปลี่ยนแปลงอวัยวะสืบพันธุ์ การเปลี่ยนแปลงฮอร์โมน ที่มีความสอดคล้องกันดังรูปที่ 2.2 แบ่งเป็น 4 ระยะคือ



รูปที่ 2.2 วงรอบการเป็นสัดของโค [5]

1) ระยะก่อนการเป็นสัด (Proestrus)

จะมีระยะเวลาประมาณ 6-10 ชั่วโมง แม่โคจะมีพฤติกรรมเดินไป-มา ไม่อยู่นิ่ง สนใจสิ่งแวดล้อม แยกตัวออกจากฝูง ส่งเสียงร้อง เอาคางเกยบั้นท้ายตัวอื่น ขึ้นขี่โคตัวอื่น เผยอริมฝีปาก ในช่วงท้ายของระยะนี้มักจะไล่ทับโคตัวอื่น ในแม่โครีดนมพบว่าผลผลิตน้ำนมลดลงประมาณ 75% ของปกติ โคกินอาหารน้อยลง

2) ระยะเป็นสัดแท้จริง (Estrus)

ระยะที่แม่โคยืนนิ่งยอมให้ตัวอื่นขึ้นทับ ช่องคลอดบวมแดง มีเมือกใสเหนียวไหลยืดออกจากช่องคลอด มักพบว่ามีเมือกติดตามบั้นท้ายและโคนหาง และมักมีขนยุ่งเหยิง แสดงร่องรอยของการขึ้นทับ ระยะนี้ใช้เวลาประมาณ 5-30 ชั่วโมง เฉลี่ยประมาณ 15-18

ชั่วโมง โดยทั่วไปการตกไข่จะเกิดขึ้นเฉลี่ย 12 ชั่วโมง หลังสิ้นสุดการเป็นสัด (หยุดยืนนิ่ง) เมื่อทำการผสมในระยะที่เหมาะสมคือ 12 – 16 ชั่วโมง หลังการเริ่มยืนนิ่ง หลังจากได้รับการผสมแล้วมักจะพบแม่โคมีลักษณะโคนหางยกและหลังแอ่น

3) ระยะหลังเป็นสัด (Metestrus)

ระยะที่โคไม่แสดงอาการเป็นสัดให้เห็น อาจพบมีเมือกปนเลือดหรือเมือกสีชาไหลออกมา หากพบเมือกโดยโคไม่แสดงอาการเป็นสัดอาจเกิดจากโคเป็นสัดเงียบ การตกไข่จะเกิดในระยะนี้ โดยเกิดขึ้นในช่วง 24-30 ชั่วโมง ภายหลังจากเริ่มแสดงอาการเป็นสัดแล้ว หรือประมาณ 4-15 ชั่วโมง หลังจากหมดสัด ระยะนี้ฮอร์โมนเอสโตรเจน (Estrogen) จะมีปริมาณลดลงอย่างมาก การพบเลือดปนเมือกนี้แสดงถึงแม่โคตัวนี้เป็นสัดมาแล้ว ไม่ต้องทำการผสมในครั้งนี้ แต่ให้รอนับวันที่ทำการตรวจการเป็นสัดเพื่อผสมในรอบต่อไป

4) ระยะพัก (Diestrus)

ระยะเวลาประมาณ 14-16 วัน เป็นระยะที่โคเงียบสงบลง โคจะหากินตามปกติ อวัยวะเพศภายนอกชีด เนื่องจากเป็นระยะที่ คอร์ปัสลูเทียม (Corpus Luteum) เจริญและมดลูก (Uterus) เตรียมรองรับการตั้งท้อง

ในขณะที่แม่โคเป็นสัด ท่อนำไข่ (Oviduct) จะยื่นปากแตร (Fimbria) ออกไปรับไข่ที่จะแตกออกมาจากถุงไข่ (Follicle) ไข่จะตกลงมาสู่ปากแตร และเคลื่อนที่มาตามท่อนำไข่ส่วนต้น (Ampular) ซึ่งเป็นบริเวณที่ตัวอสุจิ (Sperm) เคลื่อนที่มารออยู่แล้ว ตัวอสุจิตัวแรกที่ผ่านเปลือกหุ้มไข่เข้าไปสัมผัสกับไซโตพลาสซึมของเซลล์ไข่จะเป็นอสุจิเพียงตัวเดียวเท่านั้นที่ได้รับการผสมกับไข่ เกิดการรวมตัวกันของนิวเคลียสของอสุจิกับไข่ เรียกว่าการเกิดปฏิสนธิ (Fertilization) ตัวอสุจิตัวอื่น ๆ เข้าไปผสมไข่อีกไม่ได้ เนื่องจากเมื่อมีตัวอสุจิตัวแรกที่สัมผัสกับไซโตพลาสซึมของเซลล์ไข่แล้ว ไซโตพลาสซึมของเซลล์ไข่จะปล่อยสารมาคลุมผิวไซโตพลาสซึมของเซลล์ไข่ แล้วไซโตพลาสซึมของเซลล์ไข่จะปล่อยสารมาคลุมผิวไซโตพลาสซึมทั้งหมด เป็นการป้องกันไม่ให้ตัวอสุจิอื่นเข้ามาได้อีก แต่ถ้าเป็นไข่ที่ตกมาก่อนเป็นเวลานานในกรณีที่ทำการผสมเทียมช้าเกินไป ไข่มีอายุมากเกินไปและเคลื่อนตัวลงมาเลยกว่าที่ตำแหน่งท่อนำไข่ส่วนต้น หากได้รับการผสมกับอสุจิ ส่วนใหญ่จะตาย

หลังจากมีการปฏิสนธิแล้ว เซลล์จะมีการแบ่งตัวจาก 1 เซลล์ เป็น 2 เซลล์ และแบ่งต่อไปเรื่อย ๆ จนพัฒนากลายเป็นตัวอ่อน (Fetus) ในขณะที่มีการแบ่งตัวก็จะ

มีการเคลื่อนที่ลงมาที่ปีกมดลูก การเคลื่อนของตัวอ่อนถึงปีกมดลูก ใช้เวลาประมาณ 4–5 วันจากนั้นตัวอ่อนจะค่อยเจริญขึ้น ประมาณวันที่ 35 ตัวอ่อนจะฝังตัวที่ปีกมดลูก เรียกว่า การตั้งท้อง (Pregnancy)

2.1.1.2 ลักษณะอาการของโคที่เป็นสัด

1) ยืนให้โคตัวอื่นทับ เช่น พ่อโค, ลูกโค หรือแม้แต่แม่โคในฝูงขึ้นทับ ซึ่งแสดงถึงช่วงเวลาที่เหมาะสมในการผสมพันธุ์

2) พยายามขึ้นทับตัวอื่น แต่บางครั้งโคที่ขึ้นทับตัวอื่นอาจไม่ได้อยู่ในช่วงที่เป็นสัด ซึ่งต้องสังเกตให้ดี

3) อาจสังเกตพบเมือกไหล ซึ่งจะมีลักษณะที่แตกต่างกันออกไปขึ้นกับระยะเวลา โดยเมือกที่พบ อาจจะพบได้ตั้งแต่ระยะก่อนการเป็นสัดที่แท้จริงเล็กน้อย ระยะเป็นสัด และระยะหมดสัดแล้ว หากพบเมือกใสและเหนียว มักจะเป็นระยะแรกที่แสดงอาการเป็นสัดที่แท้จริง แต่หากพบเมือกขุ่นมักเป็นช่วงที่หมดสัดใหม่ ๆ มักพบเมือกตามบั้นท้ายและโคนหาง

4) โคที่เป็นสัดมักตื่นง่าย ไม่ค่อยกินอาหาร มีกิจกรรมในการเดินมากกว่าปกติ ในโคให้นม น้ำนมจะลด ส่งเสียงร้อง

5) โคที่เป็นสัดจะโหยหาตัวผู้ และมักยืนใกล้ ๆ คอกตัวผู้

6) มักพบดินหรือโคลนติดบนหลัง ตลอดจนร่องรอยจากการขึ้นทับจากโคตัวอื่น

2.1.4 การเคลื่อนไหวของโค

การศึกษาการเคลื่อนไหวของโคจากงานวิจัยจากต่างประเทศนั้น ค้นพบการบันทึกไว้เพียงแค่การก้าวเดินของโคเท่านั้น เพราะการเคลื่อนไหวอื่น ๆ ที่นอกเหนือจากการก้าวเดินนั้นเป็นไปอย่างไร้รูปแบบ (Pattern) ไม่สามารถวัดได้โดยละเอียด แค่เพียงแสดงการเคลื่อนไหวส่วนต่าง ๆ ท่านั้น

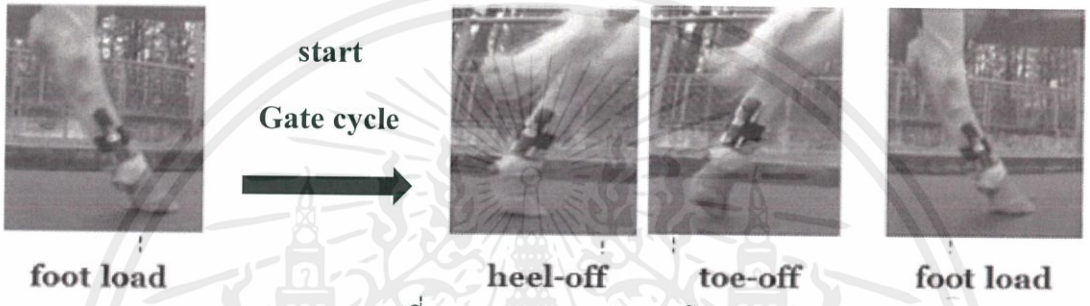
1) การก้าวเดินของโค [6]

วงรอบการเดินของโค (Gate Cycle) คือระยะในการเดินแต่ละก้าวของโค โดยจะเลือกพิจารณาเพียงขาเดียวเท่านั้นและเป็นวงรอบ โดยเริ่มที่ระยะแรกจนถึงระยะสุดท้ายจะนับเป็นหนึ่งก้าวแล้ววนกลับมาที่ระยะแรกอีกครั้ง เพื่อเริ่มการนับก้าวถัดไปโดยที่ระยะทั้งหมดถูกแบ่งเป็น 3 ระยะ ดังรูปที่ 2.3

- foot load เป็นระยะก่อนก้าว เป็นระยะที่เท้าของโคยังติดอยู่กับพื้นทั้งฝ่าเท้า และมีการทิ้งน้ำหนักลงเพื่อทรงตัว

- heel-off เป็นระยะเริ่มก้าว เป็นระยะที่โคเริ่มเปิดส้นเท้าขึ้น และทิ้งน้ำหนักไปที่เท้าส่วนหน้า

- toe-off เป็นระยะก้าว เป็นระยะที่ก่อนยกเท้าขึ้นไปจนถึงยกเท้าขึ้นเพื่อก้าว เพื่อที่จะกลับไปที่ระยะที่ 1 หรือ foot-load อีกครั้ง



รูปที่ 2.3 วงรอบการเดินของโค

2) การนับการเคลื่อนไหวของโค

โดยเริ่มจากการกำหนดค่า Threshold ที่ได้มากจากสมการที่ 2.1 มีหน่วยเป็น Milli-Gs หรือ mg และค่า 0.48828125 mg/LSB คือค่า Sensitivity Scale Factor ซึ่งมาจาก Datasheet ของ MPU6050 ส่วนค่า Ax, Ay และ Az คือค่า X,Y และ Z จาก Accelerometer ซึ่งเป็นค่า Normalize โดยเมื่อค่า Threshold มากกว่าหรือเท่ากับ 4 mg และมีค่านานถึง 120 msec ให้นับค่า Activities เป็น 1 Activity

$$\left(\sqrt{\begin{array}{l}\left(A_x previous - A_x current\right)^2 + \\ \left(A_y previous - A_y current\right)^2 + \\ \left(A_z previous - A_z current\right)^2\end{array}}\right) \times 0.48828125 = \text{Threshold} \qquad (2.1)$$

จะมีการเก็บค่า Activities ที่สะสมมาจนถึง ณ. เวลานั้น ๆ ทุก 4 ชั่วโมงไว้ในตัวแปรที่ต่างกัน และนำเข้าสมการที่ 2.2 เพื่อคำนวณหาค่า Average Activities / Hour ซึ่งการคูณ 10 คือการเลื่อนตำแหน่งทศนิยม จากนั้นจึงส่งค่าที่คำนวณได้ไปเก็บบน Database (Influxdb) ผ่านระบบ LoRaWAN โดยจะส่งข้อมูลทุก 4 ชั่วโมง

$$\text{Average Activities/Hour} = \frac{N_{\text{A}}\text{current - } N_{\text{A}}4\text{hours}}{t_{\text{c}} - t_{4\text{h}}} \times 10 \qquad (2.2)$$

เมื่อ $N_{\text{A}}\text{current}$ = จำนวน Activities ณ. ปัจจุบัน

$N_{\text{A}}4\text{hours}$ = จำนวน Activities เมื่อ 4 ชั่วโมงก่อน

t_{c} = เวลาปัจจุบัน

$t_{4\text{h}}$ = เวลาเมื่อ 4 ชั่วโมงก่อน

2.2 LPWAN (Low Power Wide Area Network) [7]

เป็นการเชื่อมต่อที่มีความต้องการพลังงานต่ำ ซึ่งเกิดขึ้นภายใต้ชื่อต่าง ๆ เช่น M2M (Machine to Machine), WSN (Wireless Sensor Networking), IoT (Internet of Things) เป็นต้น ปัจจุบันความต้องการทางการตลาดที่จะมีระบบที่ให้การเชื่อมต่อแบบ ubiquitous ในทุกที่ทุกเวลา และมีประสิทธิภาพทางการใช้แบตเตอรี่ และมีความสามารถเชื่อมต่ออุปกรณ์ได้เป็นหลายล้านตัว

2.2.1 ที่มาของ IoT/LPWAN

IoT ได้เกิดขึ้นมาราวปี 1999 ในตอนนั้นมุ่งที่จะอธิบายถึงศักยภาพของการใช้ RFID ตั้งแต่นั้นมา จึงได้วิวัฒนาการให้ครอบคลุมเซนเซอร์ทุกรูปแบบที่ทำการส่งข้อมูลไปที่ cloud รวมถึงอุปกรณ์ที่สามารถสั่งให้ทำงานทางไกลได้ ซึ่งตลาดดังกล่าวไม่ใช่ตลาดใหม่ แต่มีอยู่หลายทศวรรษแล้วในชื่อ M2M

IoT และ M2M เป็นความหวังที่จะเป็นแหล่งรายได้ใหม่ของผู้ประกอบการเครือข่าย โดยเฉพาะเมื่อเดิมทีทาง Ericsson ได้ทำนายว่าในปี 2020 จะมีอุปกรณ์เชื่อมต่อกันถึง 5 หมื่นล้านตัว ในการที่จะไปถึงจุดจุดนั้น ผู้ประกอบการเครือข่ายจำเป็นต้องผสมผสานเทคโนโลยีเข้ากับบริการที่มีต้นทุนต่ำ และสามารถเข้าสู่ตลาดได้อย่างรวดเร็ว และต้องมีความเสี่ยงของโครงการที่ต่ำ โดยอุตสาหกรรม M2M จะสามารถใช้ประโยชน์จากนวัตกรรมของเทคโนโลยี Mobile ที่มีการเปลี่ยนแปลงอย่างรวดเร็ว

2.2.2 เครือข่าย IoT / LPWAN

เครือข่าย IoT เป็นเครือข่ายแบบ Wide Area ที่ใช้พลังงานต่ำ และออกแบบมาสำหรับการใช้งานแบบ M2M เครือข่ายเหล่านี้จะมีการถ่ายโอนข้อมูลที่ความเร็วต่ำเพื่อรักษาระดับการกินพลังงานที่ต่ำและยืดอายุแบตเตอรี่ เครือข่าย LPWAN จะช่วยให้เกิดการเชื่อมต่อของอุปกรณ์ที่ใช้แบนด์วิดท์น้อยกว่าที่อุปกรณ์ตามบ้านแบบมาตรฐานใช้ และต้องสามารถขยายขนาด (scalable) เพื่อรองรับการเชื่อมต่อของอุปกรณ์ที่คาดว่าจะเพิ่มขึ้นตามที่มีการคาดการณ์ไว้

วัตถุประสงค์ของเครือข่าย LPWAN ไม่ใช่อยู่ที่การให้ความเร็วสูงสุดสำหรับการใช้งานที่มีความต้องการมาก ๆ แต่จะเป็นการใช้งานที่มีความต้องการพลังงานต่ำสำหรับอุปกรณ์ขนาดเล็ก เช่น เซนเซอร์และ Smart meter อุปกรณ์เหล่านี้ทำการอ่านและส่งข้อมูลให้กับอีกระบบหนึ่งที่ทำการประมวลผลและปฏิบัติการต่อไป การใช้งานส่วนใหญ่จะอยู่ในภาคอุตสาหกรรม ในขณะที่เทคโนโลยีในการเข้าถึงเครือข่ายแบบอื่น ๆ เช่น Bluetooth, Zigbee แบะ WiFi จะเหมาะกับการใช้งานที่มีความต้องการสูง

LPWAN ทำให้อุปกรณ์สามารถเชื่อมต่อได้ในระยะทาง 24 กิโลเมตรโดยประมาณ โดยที่มีอายุของแบตเตอรี่นานถึง 10 ปี ต้นทุนที่ต่ำ การใช้พลังงานต่ำ และความปลอดภัยในระบบถือเป็น 3 สิ่งสำคัญในการสร้างเครือข่าย IoT

2.2.3 LoRaWAN ทางเลือกของ IoT

LoRaWAN เป็นโครงสร้างพื้นฐานของ LPWAN ที่เป็นแบบ open source ที่สร้างขึ้นโดย LoRa Alliance ที่ยอมให้บริษัทอื่น ๆ สามารถสร้างเครือข่าย IoT ของตนโดยอาศัยข้อกำหนดทางเทคนิคที่กำหนดไว้ เทคโนโลยีถูกออกแบบมาเพื่อให้ gateway หรือสถานีฐานสามารถครอบคลุมพื้นที่ทั้งเมืองหรือเป็นระดับหลายร้อยตารางกิโลเมตรได้ ระยะการทำงานขึ้นกับสภาพแวดล้อมและสิ่งกีดขวาง เครือข่ายของ LoRa นั้นมี link budget หรือปัจจัยที่กำหนดระยะการทำงาน ที่กว้างไกลกว่า เทคโนโลยีสื่อสารตามมาตรฐานอื่น ๆ

การสื่อสารระหว่างอุปกรณ์ปลายทางและ gateway ได้ถูกกระจายข้อมูลบนช่องสัญญาณความถี่และมีอัตราการส่งข้อมูลต่าง ๆ LoRa นั้นใช้เทคนิค Chirp Spread Spectrum ที่มีการกระจายข้อมูลไปตลอดทั้งย่านความถี่ การสื่อสารที่มีอัตราการส่งข้อมูลต่าง ๆ จึงไม่รบกวนกันและกัน ข้อดีของการใช้ spread spectrum ในการทนทานต่อสัญญาณรบกวน ทำให้มันสามารถทำงานได้ดีแม้อยู่ในสภาพที่ไม่อำนวย เช่น มีการสะท้อนของสัญญาณ มีคลื่นวิทยุอื่น ๆ เป็นต้น

LoRa สามารถทำงานได้ในระดับสัญญาณที่ต่ำกว่า noise floor ถึง 18 dB data chirp ของ LoRa จะมีการทำ forward error correction ทำให้แม้บาง symbol จะเสียหายจากสัญญาณรบกวนก็สามารถกู้คืนข้อมูลนั้นได้ นอกจากนี้จะมีสร้างกลุ่มของช่องสัญญาณเสมือน (virtual) เป็นการเพิ่มความจุของ gateway อัตราส่งข้อมูลของ LoRaWAN จะมีตั้งแต่ 0.3 ถึง 50 kbps ขึ้นกับ configuration ของ chirp

2.2.3.1 การแบ่ง Class ของ LoRaWAN

การแบ่ง Class ของ LoRaWAN ถูกแบ่งตามการรับส่งสัญญาณได้ 3 class โดยมีวิธีต่างกันคือ

1) Class A (Bidirectional end-devices) อุปกรณ์ปลายทาง ใน Class A จะสามารถทำการสื่อสารแบบ 2 ทิศทางได้ โดยการส่ง uplink ของอุปกรณ์แต่ละครั้งจะตามมาด้วย downlink receive window สั้น ๆ 2 ช่วง slot ในการส่งข้อมูลที่ตัวอุปกรณ์จัดตารางไว้จะขึ้นอยู่กับความต้องการในการสื่อสาร โดยมีการสุ่มช่วงเวลา (random time) ตามแบบโปรโตคอล ALOHA การทำงานใน class A เป็นระบบที่อุปกรณ์ปลายทางใช้พลังงานต่ำที่สุด เหมาะสำหรับการใช้งานที่ต้องการเพียงการสื่อสาร downlink จาก Server ไม่นานหลังจากอุปกรณ์ปลายทางทำการส่งข้อมูล uplink ส่วนการสื่อสาร Downlink จาก Server ในเวลาอื่น ๆ จะต้องรอจนกว่าจะมีการจัดตารางการส่ง uplink ครั้งต่อไป

2) Class B (Bidirectional end-devices with scheduled receive slots) นอกเหนือจาก random receive window ใน Class A แล้ว อุปกรณ์ Class B จะมีการเปิด receive window พิเศษตามเวลาที่จัดตารางไว้ โดยจะอาศัยสัญญาณ beacon ที่ gateway ในการ synchronize เวลา เพื่อที่ว่า Server จะได้รู้ว่าอุปกรณ์ปลายทางจะได้รับฟังข้อมูลเมื่อไร

3) Class C (Bidirectional end-devices with maximal receive slots) อุปกรณ์ปลายทางใน Class C เกือบจะมีการเปิด receive windows อย่างต่อเนื่อง จะปิดก็ต่อเมื่อมีการส่งสัญญาณ

สถาปัตยกรรมของเครือข่าย LoRaWAN มักจะมี topology เป็นในรูป star-of-star โดยที่ Gateway เป็น transparent bridge ที่ทำการถ่ายทอด (relay) ข้อมูลระหว่างอุปกรณ์ปลายทางและ Central Network Server ในส่วน backend ทั้งนี้ Gateway จะเชื่อมต่อกับ network server ผ่านการเชื่อมต่อแบบ IP มาตรฐาน ในขณะที่อุปกรณ์ปลายทางจะสื่อสารกับ Gateway ผ่านการเชื่อมต่อไร้สายแบบ single-hop การสื่อสารของจุดปลายทางทั้งหมดจะเป็นแบบ 2 ทิศทาง และจะรองรับการอัพเกรดซอฟต์แวร์แบบ multicast ทางอากาศหรือผ่านช่องทางกระจายคำสั่งปริมาณมากอื่น ๆ เพื่อลดเวลาในการสื่อสารทางอากาศ

ผู้ให้บริการต่าง ๆ จะสามารถสร้างเครือข่ายที่เป็นอิสระจากกันโดยใช้มาตรฐานเดียวกัน สถานีฐานจะมีขนาดเล็กลงและมีราคาถูกลง เช่น ในระดับ 20 เหรียญ โดยที่แต่ละสถานีจะสามารถครอบคลุมพื้นที่กว้าง ในสภาพแวดล้อมที่เป็นเมืองใหญ่ที่เต็มไปด้วยสัญญาณรบกวน อาจต้องวางสถานีฐานให้ห่างกันประมาณ 1 ไมล์ ในขณะที่ในส่วนอื่น ๆ ของประเทศที่ไม่หนาแน่น อาจวางสถานีฐานห่างกันทุก ๆ 9 ไมล์ จากการที่ LoRa ไม่ได้ควบคุมเครือข่าย เมื่อคุณเป็นเจ้าของเครือข่ายเอง จึงไม่ต้องจ่ายค่าการเชื่อมต่อให้ใคร รูปแบบนี้อาจเหมาะกับบริษัทที่จำเป็นต้อง track กลุ่มของรถบรรทุกที่เคลื่อนที่ผ่านพื้นที่เฉพาะ เช่น ผ่านเขตบางเขตเป็นต้น

มีการก่อตั้ง LoRa Alliance จากบริษัทต่าง ๆ เช่น Cisco และ IBM ที่รวมตัวกันเพื่อสร้างเครือข่าย LoRa ทั่วโลก นอกจากนี้ยังมีบริษัทอื่น ๆ ที่จะทำการสร้างเครือข่าย LoRa ที่ครอบคลุมพื้นที่สหรัฐอเมริกาเช่นกัน LoRa Alliance ได้พยายามเข้าหาผู้ประกอบการให้ใช้เครือข่าย LoRa ในการให้บริการ IoT แต่ใคร ๆ ก็สามารถซื้อสถานีฐาน LoRa ได้ด้วยราคาเพียงไม่กี่ร้อยเหรียญและตั้งเครือข่ายของตนขึ้นได้ บริษัทต่าง ๆ มองมันเป็นเหมือนเครือข่าย Private Network โดยบริษัทสามารถติดตั้งเครือข่ายของตนเองเมื่อคิดว่าตนมีโมเดลทางธุรกิจที่รองรับได้ ซึ่งเหมาะกับการวางระบบเฉพาะที่ ไม่ใช่เป็นระดับประเทศ หรือในกรณีที่ผู้ให้บริการโครงข่าย LoRa เลิกให้บริการ คุณก็ยังสามารถติดตั้งเครือข่ายของคุณเองและดำเนินธุรกิจ IoT ต่อไปได้

2.3 บอร์ด Arduino

เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่เกิดจากการนำชิปไอซีไมโครคอนโทรลเลอร์ตระกูลต่าง ๆ มาใช้ร่วมกันในภาษา C ที่มีลักษณะเฉพาะในการเขียนไลบารี่ของ Arduino ขึ้นมาเอง มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software เพราะสาเหตุนี้ทำให้ตัวบอร์ด Arduino ถูกผู้ผลิตจีนนำไปผลิตและขายออกตลาดมาในราคาที่ถูกมาก ๆ และยังถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลง เพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ด หรือโปรแกรมต่อได้อีกด้วย

การสร้างเครื่องวัดปริมาณการเคลื่อนไหวของโคนม ทำการเลือกใช้ Arduino MKR WAN 1300 เพราะเป็นบอร์ดเสริมที่มีโมดูล LoRaWAN มีความต้องการพลังงานต่ำ และมีขนาดเล็กตรงตามจุดประสงค์ที่จะนำมาสร้างเครื่องวัดปริมาณการเคลื่อนไหวของโคนมในครั้งนี้

2.3.1 Arduino MKR WAN 1300 (LoRa connectivity)

MKR WAN 1300 เป็นบอร์ดที่มีประสิทธิภาพซึ่งรวมการทำงานของ MKR Zero และ LoRa connectivity เป็นทางออกที่ดีสำหรับผู้ที่ต้องการออกแบบโครงการ IoT ด้วยประสบการณ์ที่น้อยในการใช้เครือข่ายที่มีอุปกรณ์พลังงานต่ำ ซึ่งได้ถูกออกแบบมาเพื่อนำเสนอและมอบโซลูชั่นที่มีประสิทธิภาพคุ้มค่า สำหรับนักพัฒนาที่ต้องการเพิ่ม LoRa connectivity กับระบบเครือข่ายเล็ก ๆ ที่อยู่บนพื้นฐานของ Atmel SAMD21 และโมดูล Murata CMWX1ZZABZ Lo-Ra

การออกแบบนี้รวมถึงความสามารถในการจ่ายพลังงานให้กับบอร์ดโดยใช้แบตเตอรี่ AA หรือ AAA ขนาด 1.5 โวลต์ หรือแบตเตอรี่ภายนอก 5 โวลต์ การสลับจากแหล่งจ่ายหนึ่งไปยังอีกแหล่งจ่ายหนึ่งจะกระทำโดยอัตโนมัติ กำลังการประมวลผล 32 บิตที่คล้ายคลึงกับบอร์ด MKR ZERO ซึ่งเป็นชุดอินเทอร์เฟซ I/O ที่มีการใช้งานที่หลากหลายการสื่อสารแบบ Lo-Ra ที่ใช้พลังงานต่ำ และความง่ายในการใช้งาน Arduino Software (IDE) สำหรับนักพัฒนาและนักเขียนโปรแกรม คุณสมบัติทั้งหมดนี้ทำให้บอร์ดเป็นทางเลือกที่ต้องการสำหรับโครงการ IoT ที่ใช้พลังงานจากแบตเตอรี่ต่ำและมีขนาดกะทัดรัด พอร์ต USB ถูกนำมาใช้เพื่อจ่ายกระแสไฟ (5 โวลต์) ให้กับบอร์ด Arduino MKR WAN 1300 สามารถใช้งานได้โดยมีหรือไม่มีแบตเตอรี่ที่ต่ออยู่และมีการใช้พลังงานที่จำกัด

2.3.1.1 ข้อมูลทางเทคนิคของ Arduino MKR WAN 1300

อุปกรณ์จะถูกนำมาใช้อย่างถูกวิธีจำเป็นจะต้องรู้ข้อมูลทางเทคนิคของอุปกรณ์ ซึ่ง Arduino MKR WAN 1300 ก็ข้อมูลทางเทคนิคที่จำเป็นต้องศึกษาตามตารางที่ 2.1 ดังนี้

ตารางที่ 2.1 ข้อมูลทางเทคนิคของ Arduino MKR WAN 1300

ข้อมูลทางเทคนิค	คำอธิบาย
ไมโครคอนโทรลเลอร์	SAMD21 Cortex-M0+ 32bit low power ARM MCU
แหล่งจ่ายไฟของบอร์ด (USB/VIN)	ขนาด 5 โวลต์
การรองรับแบตเตอรี่	ถ่าน AA หรือ AAA จำนวน 2 ก้อน
ไฟที่ใช้ภายในวงจร	ขนาด 3.3 โวลต์
ขา Digital I/O	จำนวน 8 ขา
ขา PWM	12 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, A3 - or 18 -, A4 -or 19)
SPI	มี 1
I2C	มี 1
ขาอินพุตแอนะล็อก	7 (ADC 8/10/12 bit)
ขาเอาท์พุตแอนะล็อก	1 (DAC 10 bit)
Interrupts ภายนอก	8 (0, 1, 4, 5, 6, 7, 8, A1 -or 16-, A2 - or 17)
กระแสไฟตรงต่อขา I/O	7 mA
Flash Memory	256 KB
SRAM	32 KB
Clock Speed	32.768 kHz (RTC), 48 MHz
กำลังส่งสายอากาศ	2dB
ย่านความถี่	433/868/915 MHz

Arduino MKR WAN 1300 ต้องใช้สายอากาศ GSM ต่อเข้ากับบอร์ดด้วยช่องต่อ UFL ขนาดเล็ก โปรดตรวจสอบว่าสามารถรับความถี่ได้ในช่วงของ LoRa (433/868/915 MHz) หรือไม่ และเพื่อให้ได้ผลลัพธ์ที่ดีที่ห้ามติดตั้งสายอากาศกับพื้นผิวโลหะ เช่น โครงรถ เป็นต้น

2.4 โมดูล GY-521 (3-axis Accelerometer/Gyro Module)

เซนเซอร์ MPU-6050 ของ InvenSense ภายในประกอบไปด้วย Accelerometer และ Gyroscope แบบ 3 แกน ภายในชิปตัวเดียวประกอบเป็น โมดูล GY-521 ซึ่งมีความแม่นยำสูง เนื่องจากภายในถูกบรรจุด้วยอุปกรณ์การแปลงแอนะล็อกเป็นดิจิทัล (Analog to Digital Converter หรือ ADC) แบบ 16 บิต สำหรับแต่ละช่องสัญญาณ (Channel) ดังนั้นโมดูลนี้สามารถทำงานทั้ง 2 อย่างในเวลาเดียวกันคือ ใช้ตรวจสอบทิศทางการเคลื่อนที่ และตรวจสอบความเร็วในการเปลี่ยนแปลงทิศทาง ตามที่กล่าวมาโมดูลจึงมีลักษณะตามรูปที่ 2.4 และมีคำอธิบายขาสัญญาณตามตารางที่ 2.2

รูปที่ 2.4 โมดูล GY-521 [8]

ตารางที่ 2.2 คำอธิบายขาสัญญาณของ โมดูล GY-521

ขาที่	ชื่อขาสัญญาณ	คำอธิบาย
1	VCC-IN	ขาป้อนไฟเลี้ยงโมดูลไปยังตัวปรับแรงดันเป็น 3.3 โวลต์
2	GND	ขากราวด์
3	SCL (Serial Clock)	ขาสัญญาณนาฬิกา บนบัส I2C
4	SDA (Serial Data)	ขาสัญญาณข้อมูล บนบัส I2C

ตารางที่ 2.2 คำอธิบายขาสัญญาณของ โมดูล GY-521 (ต่อ)

ขาที่	ชื่อขาสัญญาณ	คำอธิบาย
5	XDA (AUX_SDA)	ขาสัญญาณข้อมูล บนบัส I2C (I2C Master Mode is enabled)
6	XCL (AUX_SCL)	ขาสัญญาณนาฬิกา บนบัส I2C (I2C Master Mode is enabled)
7	AD0	ที่อยู่ของ I2C Slave LSB (AD0)
8	INT	Interrupt

2.4.1 Accelerometer sensor

Accelerometer เป็นชิปที่ประกอบอยู่ใน โมดูล GY-521 และเป็นอุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่วัดแรงความเร่ง แรงเหล่านี้อาจจะเป็นแรงคงที่ เช่น แรงโน้มถ่วงโลกที่กระทำต่อตัวบุคคล หรืออาจเป็นแรงที่ไม่คงที่ที่เกิดจากการเคลื่อนที่หรือสั่น Accelerometer โดยการวัดปริมาณของความเร่งแบบคงที่อันเนื่องมาจากแรงโน้มถ่วงโลกสามารถหามุมเอียงของอุปกรณ์ได้เมื่อเทียบกับพื้นโลก และเมื่อทราบปริมาณความเร่งที่ไม่คงที่ จะสามารถวิเคราะห์วิธีการเคลื่อนที่ของอุปกรณ์ได้

การที่จะวัดความเร่งของ Accelerometer มีหลายวิธี Accelerometer โดยส่วนใหญ่มักใช้หลักการของ Piezoelectric Effect ซึ่งเป็นหลักการของอุปกรณ์ตรวจวัดแรงกลต่าง เช่น แรงดัน ความเร่ง การสั่น แรง เครียด หรือแรงกระทำอื่น ๆ โดยเปลี่ยนพลังงานกลเหล่านี้ให้เป็นพลังงานไฟฟ้า ภายใน Accelerometer จะมีโครงสร้างภายในที่ประกอบไปด้วยผลึกโครงสร้างจุลภาค (Microstructures) ที่จะได้รับความเค้น (Stressed) ทำให้เกิดแรงดันไฟฟ้า อีกวิธีหนึ่งคือการตรวจจับการเปลี่ยนแปลงความจุ ถ้ามีโครงสร้างจุลภาคสองตัวที่อยู่ติดกันเมื่อเกิดความเร่งเคลื่อนย้ายโครงสร้างจุลภาคตัวหนึ่งจะส่งผลให้ความจุโดยรวมเปลี่ยนแปลงไป และเมื่อเพิ่มวงจรแปลงความจุไฟฟ้าเป็นแรงดันไฟฟ้าจะได้การทำงานของ Accelerometer ยังมีวิธีการอีกมากมายในการสร้าง Accelerometer รวมไปถึง Piezoresistive Effect, Hot air bubbles และด้วยวิธีการทางแสง

2.4.2 Gyroscope sensor

Gyroscope แบบ 3 แกน สามารถวัดความเร็วรอบแกนได้ 3 แกน ได้แก่ x, y และ z ซึ่ง Gyroscope บางตัวสามารถวัดได้เพียงแกนเดียวหรือ 2 แกนเท่านั้น แต่ Gyroscope แบบ 3 แกนนี้สามารถวัดความเร็วรอบแกนได้ 3 แกนภายในชิปเดียว มีขนาดที่เล็กกว่า ราคาไม่แพง และเป็นที่นิยมอีกด้วย การทำงานของ Gyroscope จะเริ่มทำงานก็ต่อเมื่อเกิดการเอียง กำลังจะเอียง หรือเกิดการเคลื่อนไหว แต่เมื่อวัตถุอยู่นิ่ง Gyroscope จะวัดค่าไม่ได้ เพราะไม่มีความเร็ว

2.4.3 ความแตกต่างระหว่าง Accelerometer และ Gyroscope sensor

เริ่มจากแรงโน้มถ่วงนั้นมีผลกับเอาท์พุตของ Accelerometer โดยตรง ดังนั้น เอาท์พุตของ Accelerometer จะไม่มีทางหยุดนิ่งแม้จะปล่อยทิ้งไว้ เอาท์พุตก็ยังมีค่าที่วิ่งขึ้น-ลง เกิดการเปลี่ยนแปลงตลอดเวลา ซึ่งต่างกันกับ Gyroscope ที่ปล่อยทิ้งไว้ค่าเอาท์พุตที่ได้ก็จะนิ่งไม่เกิดการเปลี่ยนแปลง และถ้านำเอาท์พุตของ Gyroscope มาใช้หลักการ Discrete Integral ก็จะสามารถหามุมที่เกิดขึ้นได้ เพราะเอาท์พุตของ Gyroscope เป็นความเร็วเชิงมุม นี้เป็นเหตุผลที่ทำให้เกิดการรวม Sensor สองตัวนี้ ทำให้มีทฤษฎีเกิดขึ้นมารองรับมากมาย เช่น Kalman Filter, Complementary Filter ในการใช้การผสมผสานข้อดีของทั้งสอง Sensor

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

ปริญญานิพนธ์ในส่วนนี้จะนำทฤษฎีและหลักการที่เกี่ยวข้องมาประยุกต์ใช้เพื่อออกแบบการเชื่อมต่ออุปกรณ์และการทำงานของระบบ ซึ่งเนื้อหาในบทนี้จะกล่าวถึงรายละเอียดในการออกแบบการเชื่อมต่อระบบย่อยเข้าด้วยกัน เครื่องมือที่นำมาใช้ในการทดลอง และการจัดเก็บผลการทดลองจากอุปกรณ์ต่าง ๆ

3.1 การออกแบบ

3.1.1 การออกแบบเครื่องวัดปริมาณการเคลื่อนไหวของโคนม

การศึกษาการทำงานของ Arduino MKR WAN 1300, โมดูล GY-521, แบตเตอรี่ และสายอากาศ GSM (850/900/1800/1900 MHz) แล้ว ต่อมาจะเป็นการติดตั้งอุปกรณ์เข้าด้วยกัน ดังรูปที่ 3.1 รวมไปถึงการเขียนโปรแกรมวัดปริมาณการเคลื่อนไหว ลงบน Arduino MKR WAN 1300 ตามขั้นตอนดังนี้

รูปที่ 3.1 การเชื่อมต่ออุปกรณ์เข้าด้วยกัน

1) นำขา 5V, GND, SCL และ SDA ของโมดูล GY-521 ต่อเข้ากับช่อง 5V, GND, SCL และ SDA ของ Arduino MKR WAN 1300 ตามลำดับ

2) เสียบหัวต่อแบตเตอรี่กับอุปกรณ์เข้าด้วยกัน

3) ต่อสายอากาศ GSM เข้ากับ Arduino MKR WAN 1300

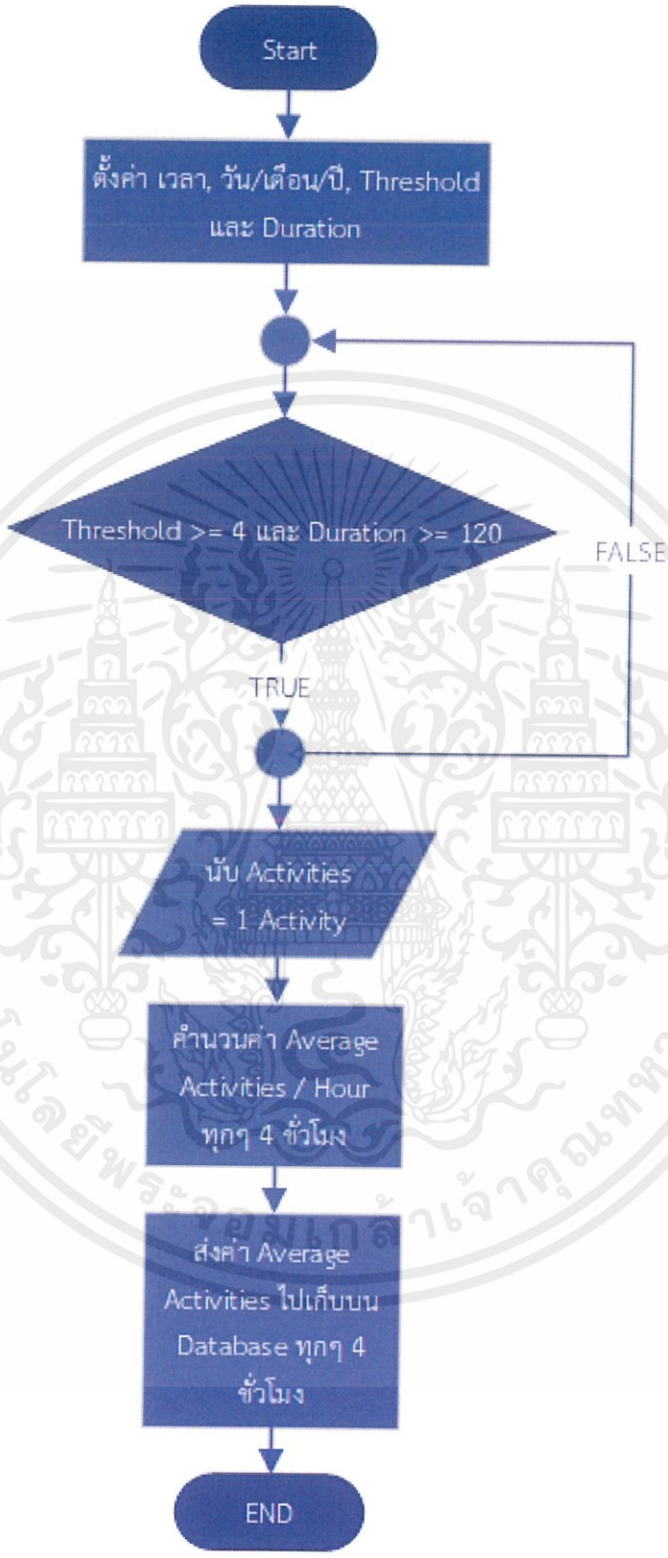
4) ติดตั้ง Arduino SAMD Boards (32-bits ARM Cortex-Mo+) by Arduino บน Arduino IDE ผ่าน Board Manager บน Arduino IDE

5) ติดตั้ง MKRWAN Library บน Arduino IDE ผ่าน Library Manager

6) ติดตั้ง MPU6050 Library บน Arduino IDE จาก https://github.com/jarzebski /Arduino-MPU6050

7) ติดตั้ง RTCZero Library บน Arduino IDE ผ่าน Library Manager

8) เขียนโปรแกรมวัดปริมาณการเคลื่อนไหวผ่าน Arduino IDE ลงบน Arduino MKR WAN 1300 โดยจะแสดงการทำงานของโปรแกรมวัดปริมาณการเคลื่อนไหวตามแผนภาพ Flowchart ดังรูปที่ 3.2



รูปที่ 3.2 แผนผังการทำงานของโปรแกรมวัดปริมาณการเคลื่อนไหวของโคนม

3.1.2 การออกแบบ Gateway ของ LoRaWAN [9]

การออกแบบ Gateway ของ LoRaWAN จะใช้บอร์ด Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora ที่มีการติดตั้งสายอากาศย่าน 923 MHz เอาไว้แล้ว โดยขั้นตอนออกแบบ gateway เป็นดังนี้

1) ติดตั้ง esp32 by Espressif Systems บน Arduino IDE ผ่าน Board Manager จาก https://github.com/espressif/arduino-esp32

2) ติดตั้ง Library ของ Single Channel LoRa Gateway บน Arduino IDE จาก https://github.com/things4u/ESP-1ch-Gateway-v5.0

3) ใช้ Source Code ของ Single Channel LoRa Gateway ในการออกแบบ LoRaWAN Gateway

4) เปิดไฟล์ ESP-sc-gway.ino และทำการแก้ไข Code ภายในไฟล์ตามตารางที่ 3.1 ดังนี้

ตารางที่ 3.1 การตั้งค่าไฟล์ ESP-sc-gway.ino

ลำดับที่	ตำแหน่งที่แก้ไข	แก้ไขเป็น
1	เปิดหน้า ESP-sc-gway.h ตรงหัวข้อ #define _LFREQ 868	แก้เป็น #define _LFREQ 923
2	หัวข้อ #define _SPREADING SF9	#define _SPREADING SF10
3	หัวข้อ #define _CAD 1	#define _CAD 0
4	หัวข้อ #define _PIN_OUT 1	#define _PIN_OUT 4
5	หัวข้อ #define _STRICT_1CH 0	#define _STRICT_1CH 1
6	หัวข้อ #define AP_NAME "YourName" และหัวข้อ #define AP_PASSWD "YourPassword"	ให้ใส่ SSID ของ Access point และ Password ของ Access point
7	หัวข้อ #define OLED 2	#define OLED 1

ตารางที่ 3.1 การตั้งค่าไฟล์ ESP-sc-gway.ino (ต่อ)

ลำดับที่	ตำแหน่งที่แก้ไข	แก้ไขเป็น
8	หัวข้อ #define _TTNSERVER "router.eu.thethings.network"	#define _TTNSERVER "router.as2.thethings.network" และต้องตรงกับการ Register Gateway บน The Thing Network ในหัวข้อที่ 3.1.3.1
9	หัวข้อ Gateway Ident definitions	สามารถแก้ไขตามต้องการได้
10	หัวข้อ #define NTP_TIMEZONES 2	#define NTP_TIMEZONES 7
11	หัวข้อ wpas wpa[] =	ให้แก้ในวงเล็บใน Array ที่ 2 เป็น SSID ของ Access point และ PASSWORD ของ Access point เช่น { "ECC510-2.4Ghz", "telecom510" }
12	เปิดหน้า loraModem.h	แก้ไขบรรทัดที่ 96 ถึง 111 ตามที่แสดงดังรูปที่ 3.3

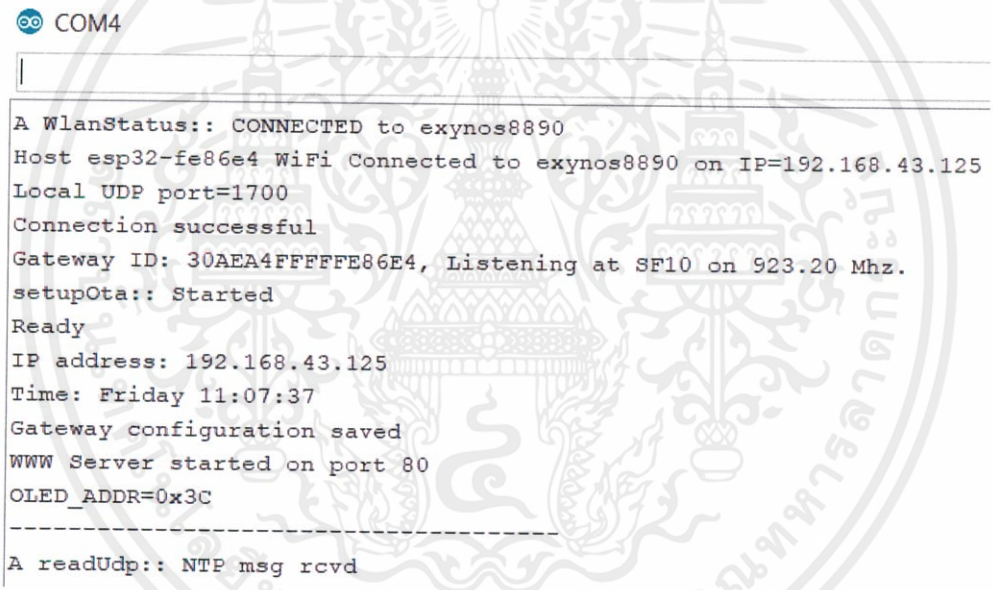
```
96 #elif _LFREQ==923
97 // US902=928
98 // AU915-928
99 int freqs [] = {
100   // Uplink
101   923200000,          // Channel 0, SF7BW125 to SF10BW125 primary
102   923400000,          // Ch 1, SF7BW125 to SF10BW125
103   923600000,          // Ch 2, SF7BW125 to SF10BW125
104   923800000,          // Ch 3, SF7BW125 to SF10BW125
105   924000000,          // Ch 4, SF7BW125 to SF10BW125
106   924200000,          // Ch 5, SF7BW125 to SF10BW125
107   924400000,          // Ch 6, SF7BW125 to SF10BW125
108   924600000,          // Ch 7, SF7BW125 to SF10BW125
109   924500000,          // Ch 8, SF8BW500
110   924800000
111   // Downlink
```

รูปที่ 3.3 Source Code ของหน้า loraModem.h

5) แล้วทำการเปิด Serial Monitor และทำการ Compile and upload ลงบอร์ด

6) บนหน้า Serial Monitor จะขึ้นดังรูปที่ 3.4 แล้วใช้ค่า IP Address ในนั้นไปเปิดบน Web Browser จะแสดงหน้าเว็บ ESP Gateway Config ขึ้นมา ดังรูปที่ 3.5

7) กดที่ Expert mode แล้ว Refresh หน้าเว็บ แล้วจึงกดปิด CAD ตรง Gateway Settings จากนั้นจำค่า Gateway ID จาก Serial Monitor หรือ ตรง System Status บนหน้าเว็บ ESP Gateway Config เพื่อใช้ในขั้นตอนการออกแบบ Network Server ของ LoRaWAN ต่อไป



รูปที่ 3.4 Serial Monitor

ESP Gateway Config

Version: V.5.3.3.H; 180825a
ESP alive since Friday 14-12-2018 11:09:11, Uptime: 0-00:01:43
Current time Friday 14-12-2018 11:10:41

Documentation | Expert Mode | Log Files

Package Statistics

Counter	C 0	C 1	C 2	Pkgs	Pkgs/hr
Packages Downlink				0	
Packages Uplink Total				0	0
Packages Uplink OK				0	
SF7 rcvd	0	0	0	0	0 %
SF8 rcvd	0	0	0	0	0 %
SF9 rcvd	0	0	0	0	0 %
SF10 rcvd	0	0	0	0	0 %
SF11 rcvd	0	0	0	0	0 %
SF12 rcvd	0	0	0	0	0 %

Message History

Time	Node	C	Freq	SF	pRSSI

Gateway Settings

Setting	Value	Set	
CAD	OFF	ON	OFF
HOP	OFF	ON	OFF
SF Setting	10	-	+
Channel	0	-	+

รูปที่ 3.5 ESP Gateway Config

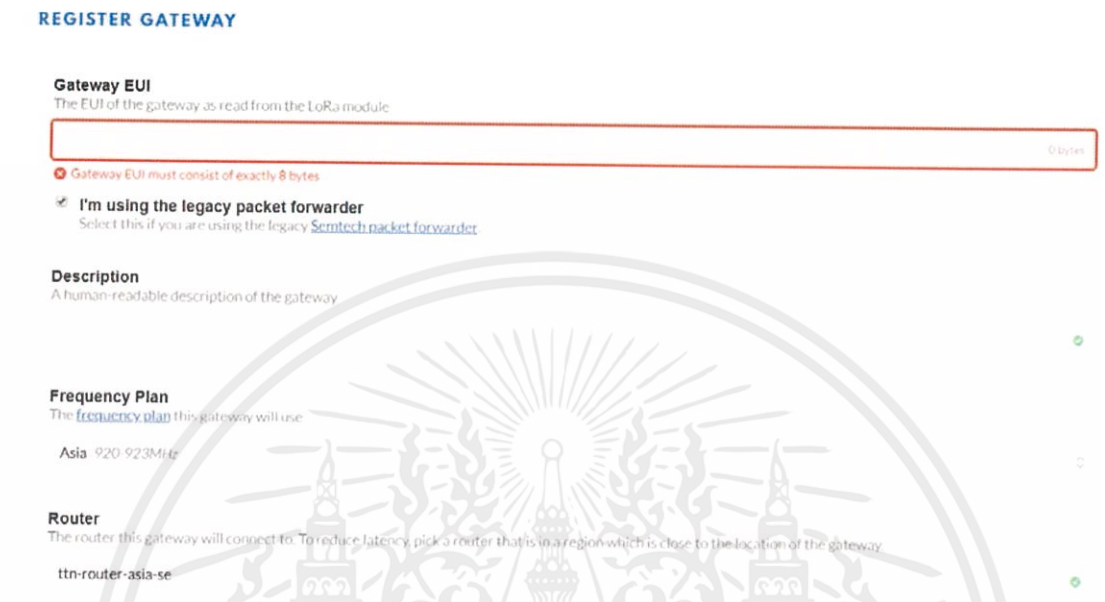
3.1.3 การออกแบบ Network Server ของ LoRaWAN [10]

การออกแบบ Network Server นั้นจะใช้ The Thing Network เป็น Network Server เพื่อเป็นตัวกลางการควบคุมการส่ง-รับข้อมูลของ Node, Gateway และ Application Server ของ LoRaWAN

3.1.3.1 การ Register Gateway บน The Thing Network สามารถ Register Gateway ได้ตามขั้นตอนดังนี้

1) สมัครสมาชิกบนเว็บไซต์ The Thing Network จาก https://console.thethingsnetwork.org/

2) เข้าไปที่หน้า Gateway แล้วทำการกด Register gateway แสดงดังรูปที่ 3.6



รูปที่ 3.6 หน้า Register Gateway บน The Thing Network

3) หน้า Register gateway ให้คลิกถูกหน้า I'm using the legacy packet forwarder แล้วใส่ Gateway EUI หรือ Gateway ID จากขั้นตอนการออกแบบ Gateway ของ LoRaWAN

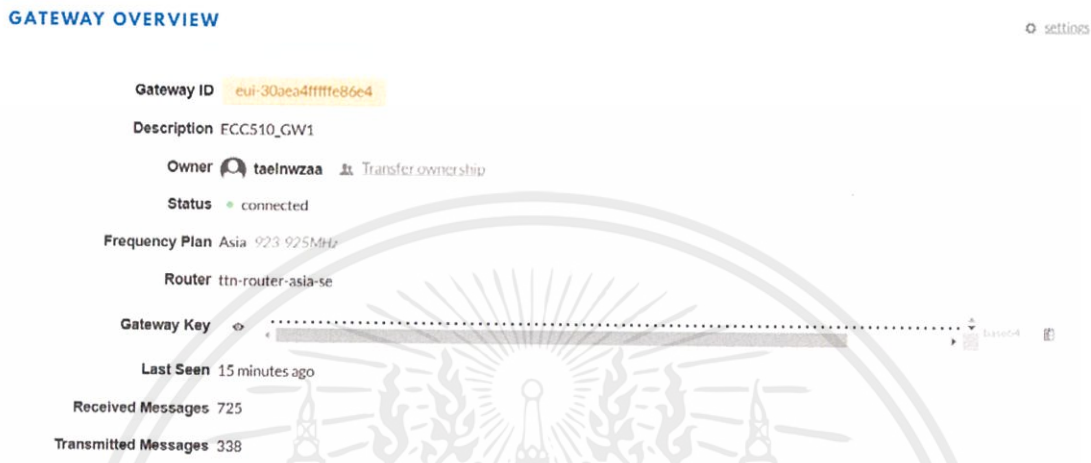
4) หัวข้อ Description ให้ใส่ชื่อ Gateway ที่อยากตั้ง แต่ในที่นี้ใช้ชื่อ ECC510_GW1

5) หัวข้อ Frequency Plan ให้เลือกตามความถี่ที่ใช้ แต่ในที่นี้ใช้ความถี่ 923 MHz จึงเลือกใช้ Asia 923-925 MHz

6) หัวข้อ Router ถ้าเลือกใช้ Asia 923-925 MHz ก็ให้ใช้ ttn-router-asia-se ซึ่งต้องเลือกให้สัมพันธ์กับการตั้งค่าตอนขั้นตอนออกแบบ Gateway ของ LoRaWAN ตรง #define _TTNSERVER "router.as2.thethings.network" ด้วย

7) หัวข้อ Location ให้ใส่ตำแหน่งที่ติดตั้ง Gateway

8) หัวข้อ Antenna Placement ให้ใส่ว่าสายอากาศของ Gateway อยู่ภายใน หรือภายนอกอาคาร ในที่นี้เลือก indoor แล้วจึงทำการกด Register Gateway ก็จะได้หน้าเว็บไซต์ดังรูปที่ 3.7



รูปที่ 3.7 หน้า Gateway Overview บน The Thing Network

3.1.3.2 การ Add Applications บน The Thing Network สามารถ Add Applications ได้ตามขั้นตอนดังนี้

1) กลับมาที่หน้า Console หลัก

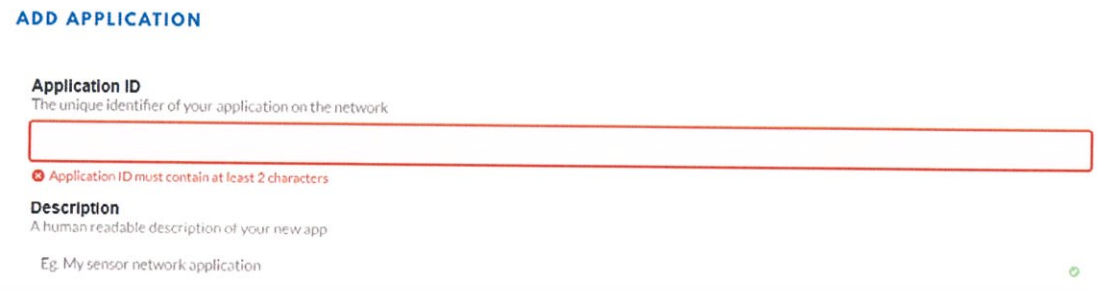
2) เข้าหน้า Applications แล้วทำการกด Add Applications

3) ดังรูปที่ 3.8 ตรงหัวข้อ Application ID ให้ใส่ ID ของ Application ซึ่งตั้งเป็นอะไรก็ได้ แต่ในที่นี้ตั้งว่า ecc_lora007

4) หัวข้อ Description ให้ใส่ชื่อของ Application ซึ่งตั้งเป็นอะไรก็ได้ แต่ในที่นี้ตั้งว่า ECC510_APP

5) หัวข้อ Application EUI ให้ The Thing Network จะสร้างให้เอง

6) หัวข้อ Handler registration ให้เลือกตามตำแหน่งที่อยู่ใกล้ที่สุด ในที่นี้เลือก ttn-handler-asia-se จากนั้นกด Add Applications แล้วจะได้หน้า Applications Overview ดังรูปที่ 3.9



รูปที่ 3.8 หน้า Add Applications บน The Thing Network

รูปที่ 3.9 หน้า Applications Overview บน The Thing Network

3.1.3.3 การ Register Device แบบ OTAA บน The Thing Network สามารถ Register Device แบบ OTAA ได้ตามขั้นตอนดังนี้

1) ดาวน์โหลดไฟล์จากhttps://github.com/gonzalocasas/arduino -mkr-wan-1300

2) เปิดไฟล์ mkrwan_01_get_deveui.ino และทำการ Compile and upload ลงบอร์ด Arduino MKR WAN 1300 จากนั้นเปิด Serial monitor แล้วเก็บค่า device EUI ที่ขึ้นบน Serial monitor เพื่อนำไปใช้ในขั้นตอนต่อไป

3) บนหน้า Applications Overview บน The Thing Network ให้กด Register Device ตรงหัวข้อ Device เพื่อเพิ่มอุปกรณ์ หรือ Node เข้าไปในระบบ

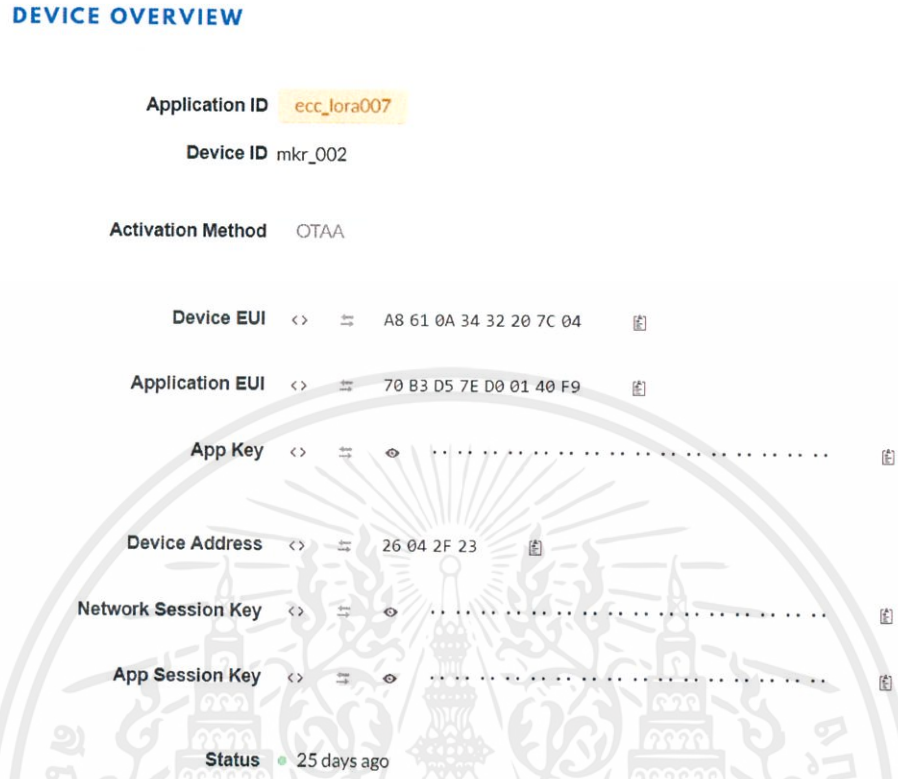
4) หน้า Register Device ดังรูปที่ 3.10 ตรงหัวข้อ Device ID ให้ใส่ ID ของ Device ซึ่งตั้งเป็นอะไรก็ได้ แต่ในที่นี้ตั้งว่า wan002

5) หัวข้อ Device EUI ให้ใส่ค่า device EUI ที่ได้จากข้อ 2

6) หัวข้อ App Key เว็บไซต์ The Thing Network จะสร้างให้

7) หัวข้อ App EUI เว็บไซต์ The Thing Network จะเอา APPLICATION EUI มาใส่ให้โดยอัตโนมัติ จากนั้นกด Register ก็จะแสดงหน้า Device Overview ดังรูปที่ 3.11

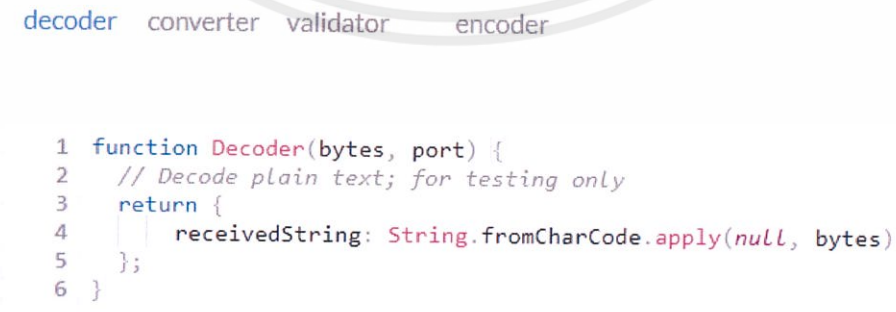
รูปที่ 3.10 หน้า Register Device บน The Thing Network



รูปที่ 3.11 หน้า Device Overview บน The Thing Network

8) กลับไปที่หน้า APPLICATION OVERVIEW ให้กดตรง Payload Formats ที่ใช้สำหรับแปลงข้อมูลที่แสดงใน Packet จากเลขฐานสิบหกให้แสดงเป็นข้อมูลตามต้องการ

9) หัวข้อ Payload Format ให้เลือก Custom และเลือก decoder ตรงด้านล่างหัวข้อ Payload Format จากนั้นใส่ Code ดังรูปที่ 3.12 แล้วกด Save Payload Function

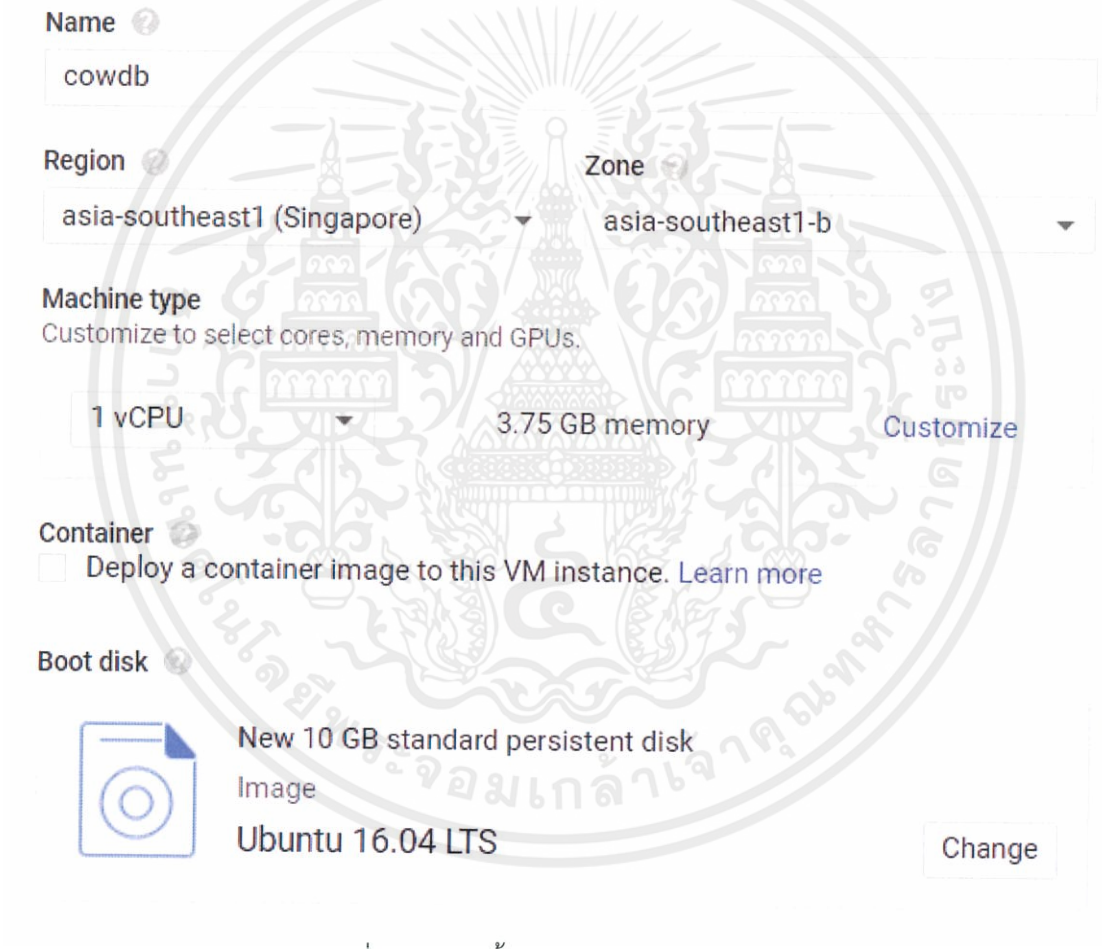


รูปที่ 3.12 Payload Format

3.1.4 การออกแบบ Node Red [11]

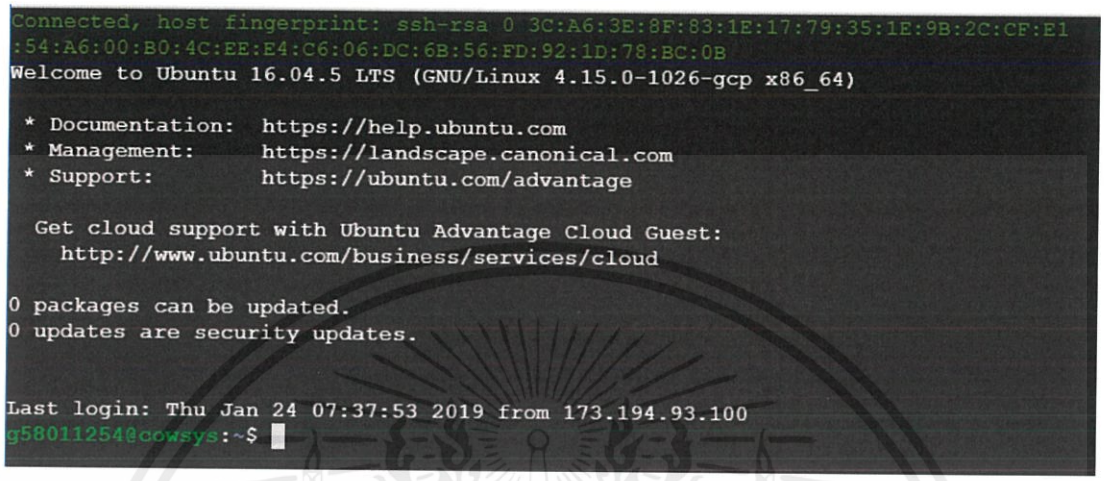
การออกแบบ Node Red ตัวกลางระหว่าง Network server และ Application Server ของ LoRaWAN จะมีการใช้งาน VM instances ของ Google cloud Platform

1) สร้างระบบ VM instances ขึ้นมาโดยเลือก Region เป็น asia-southeast1 และเลือก Machine type เป็น 1vCPU และ Boot disk เป็น Ubuntu 16.04 LTS และ Firewall ให้เลือกทั้ง Allow HTTPS traffic และ Allow HTTPS traffic และกด Create ดังรูปที่ 3.13



รูปที่ 3.13 การตั้งค่า ระบบ VM instances

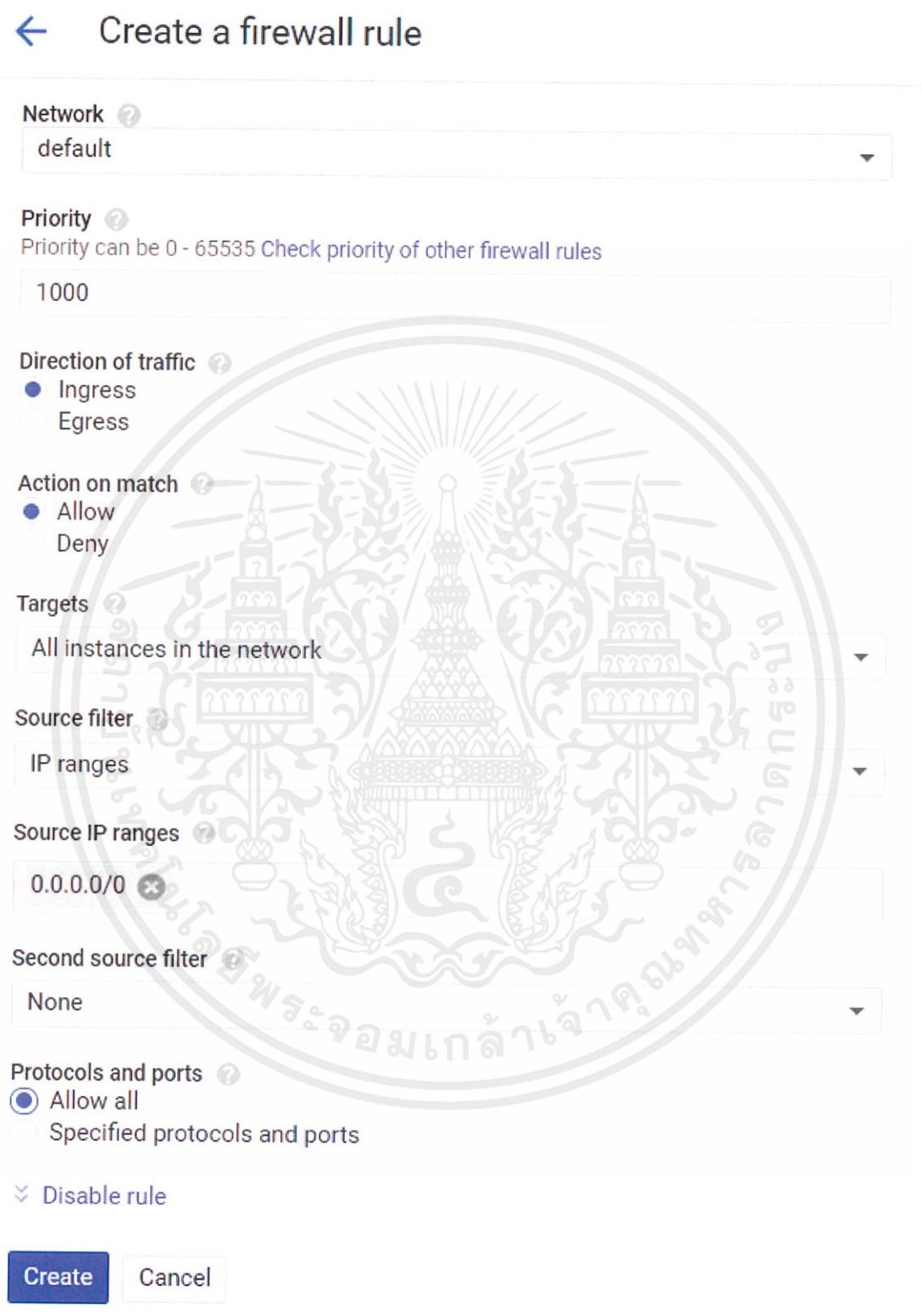
2) เมื่อกด connect เพื่อเชื่อมต่อกับ VM instances แล้วจะได้หน้าต่าง Command Line ของ Ubuntu ดังรูปที่ 3.14



รูปที่ 3.14 หน้าต่าง Command Line ของ Ubuntu

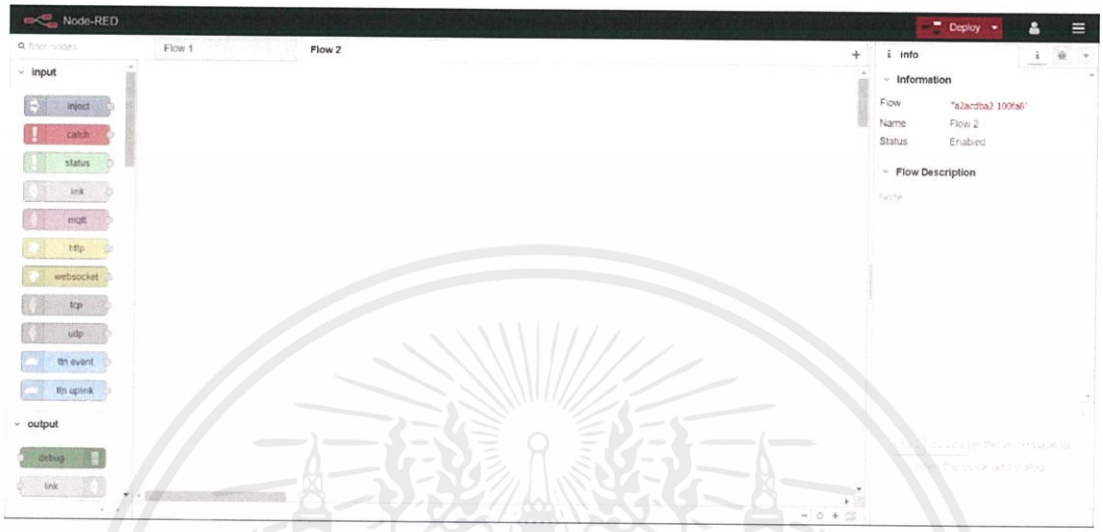
3) ใช้คำสั่งดังนี้ตามลำดับเพื่อสร้าง Node Red ในระบบปฏิบัติการ Ubuntu โดยเริ่มจากคำสั่ง sudo apt-get update, sudo apt-get upgrade เพื่ออัพเดทระบบให้เป็นปัจจุบัน, sudo apt-get install nodejs-legacy เพื่อติดตั้ง Node.js, sudo apt-get install npm เพื่อติดตั้ง Node Package Manager, sudo npm install -g --unsafe-perm node-red node-red-admin เพื่อติดตั้ง Node Red, sudo ufw allow 1880 เพื่อเปิดพอร์ต 1880 และ node-red เพื่อเปิดใช้งาน Node Red

4) ไปยังหน้าเว็บของ VM instances กดที่จุด 3 จุด ด้านข้างของ Conect เพื่อเข้า View Network Details กดที่ Firewall rules ด้านซ้ายมือของเว็บ จากนั้นกด Create Firewall Rule แล้วตั้งชื่อตามต้องการ เลือก Target เป็น All instances in the network ส่วน Source IP ranges เป็น 0.0.0.0/0 และ Protocols and ports เลือก Allow all และกด Create ดังรูปที่ 3.15



รูปที่ 3.15 เว็บสำหรับตั้งค่า Firewall

5) ทดสอบลองเข้า Node Red ด้วย http://External IP ของ VM instances:1880 ดังรูปที่ 3.16



รูปที่ 3.16 Node Red

6) การตั้งค่าให้ Node Red ทำงานอัตโนมัติบน VM instances ให้ใช้คำสั่งบน Command Line ดังนี้ sudo vi /etc/systemd/system/node-red.service จากนั้นให้ Copy และ Paste Code ต่อไปนี้

[Unit]

Description=Node-RED

After=network.target

[Service]

Type=simple

ExecStart=/usr/local/bin/node-red-pi --max-old-space-size=128 -v

Restart=on-failure

KillSignal=SIGINT

WorkingDirectory=/home/g58011254

User=g58011254

[Install]

WantedBy=multi-user.target

จากนั้นแก้ไข g58011254 ให้ตรงกับชื่อก่อน @ ที่อยู่ด้านบนของหน้าต่าง Command Line แล้วกด Esc และพิมพ์ :wq เพื่อ save จากนั้นพิมพ์คำสั่งต่อไปนี้ตามลำดับ sudo systemctl daemon-reload, sudo systemctl enable node-red.service, sudo systemctl start node-red.service และ sudo systemctl status node-red.service

7) ถ้าต้องการตั้ง Username และ Password เพื่อเข้าใช้ Node Red ทำตามนี้ ให้พิมพ์คำสั่ง node-red-admin hash-pw จากนั้นให้พิมพ์ Password ที่ต้องการ และ copy ค่า hash เอาไว้ แล้วพิมพ์คำสั่ง sudo vi ~/.node-red/settings.js จากนั้นพิมพ์ /adminAuth เพื่อหา และ uncomment โดยการเอา // ออก แล้วจึงเปลี่ยน username ด้วยการกด I เพื่อแก้ไข และเอาค่า hash ที่ได้มาก่อนหน้ามาใส่ทับ hash ที่มีอยู่แล้ว และต้อง uncomment ทั้ง section นั้นด้วย จากนั้น save โดยกด Esc และพิมพ์ :wq และรีสตาร์ต node-red ด้วยคำสั่ง sudo systemctl restart node-red.service และลองเข้า Node Red ผ่าน http://External IP ของ VM instances:1880 ใหม่

3.1.5 การออกแบบ Database Server (Influxdb) ของ LoRaWAN

Influxdb หรือ Database Server ใช้สำหรับเก็บข้อมูล โดยสามารถจัดการข้อมูลได้ผ่าน Command Line ของ Ubuntu

1) พิมพ์ตามลำดับคำสั่งดังนี้ curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add - ตามด้วย source /etc/lsb-release และ echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable" | sudo tee /etc/apt/sources.list.d/influxdb.list จากนั้นทำการติดตั้ง และสั่งทำงาน Influxdb ดังนี้ sudo apt-get update && sudo apt-get install influxdb และ sudo service influxdb start

2) สร้าง Database ด้วยการพิมพ์คำสั่ง influx ตามด้วยคำสั่ง CREATE DATABASE cowdb โดย cowdb เป็นชื่อของ DATABASE สามารถตั้งได้ตามต้องการ และคำสั่ง USE cowdb เพื่อใช้สำหรับจัดการ Database

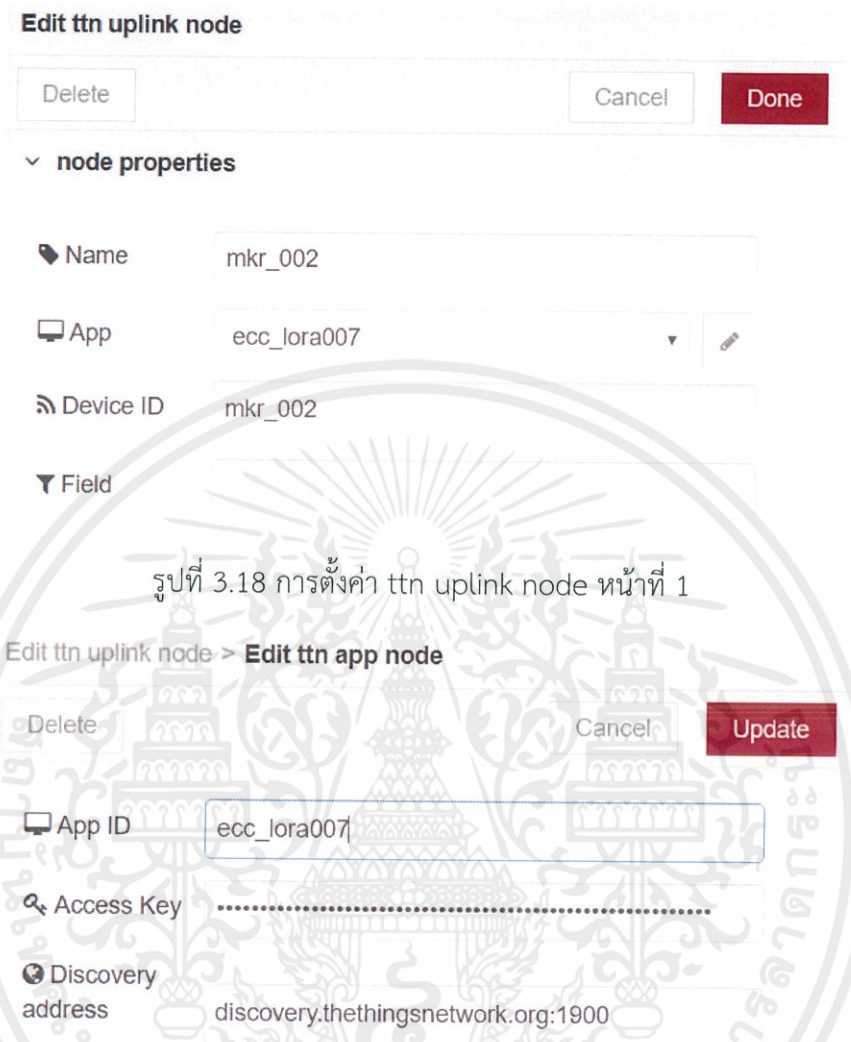
3) สามารถสร้าง Measurement หรือก็คือส่วนย่อยของ DATABASE ที่ใช้สำหรับเก็บข้อมูลได้ในหัวข้อที่ 3.15 โดยสามารถดู Measurement ได้โดยใช้คำสั่ง USE cowdb ตามด้วย show measurements และดูข้อมูลที่เก็บไว้ใน Measurement ผ่าน SELECT * FROM "2" ซึ่งเลข 2 ในที่นี้คือชื่อของ Measurement ที่สร้างมา สามรถลบข้อมูลใน Measurement ด้วยคำสั่ง DROP SERIES FROM "2" หรือ DELETE FROM "2" WHERE time = 154827159427 1597389 เพื่อลบข้อมูลที่เก็บไว้ที่เวลาตาม Timestamp ดังกล่าว

4) เข้าไปที่ Node Red แล้วกดปุ่มสัญลักษณ์สามขีด เลือก Manage palette และกดที่ Install หาคำว่า Influx จากนั้นติดตั้ง node-red-contrib-influxdb และหาคำว่า ttn จากนั้นติดตั้ง node-red-contrib-ttn

5) ด้านซ้ายมือของ Node Red ให้ลาก ttn uplink node, function node และ Influxdb node มาวางตรงที่ว่าง และเชื่อมต่อกันดังรูปที่ 3.17

รูปที่ 3.17 การเชื่อมต่อ Node ต่าง ๆ

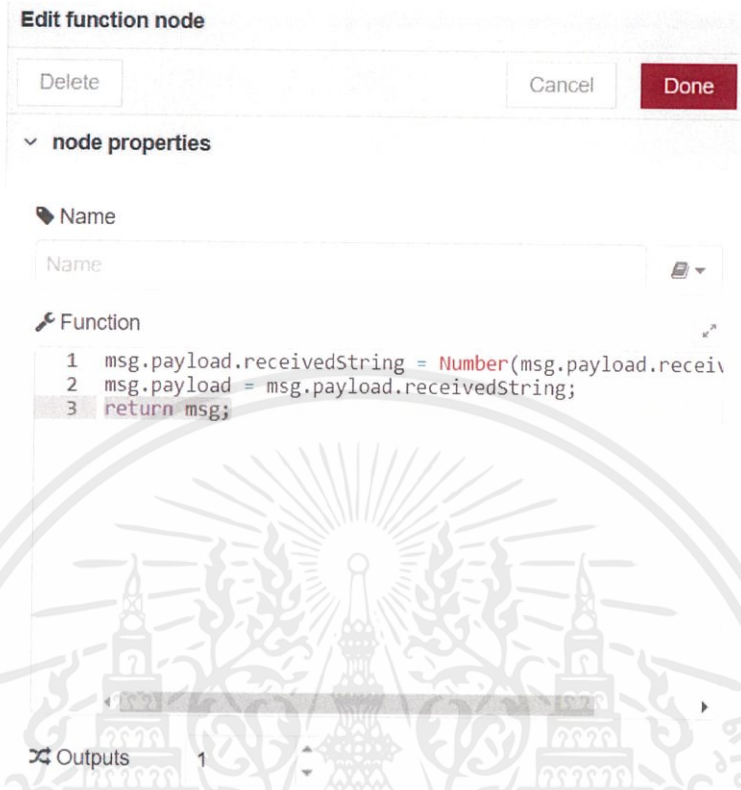
6) ดับเบิ้ลคลิกที่ ttn uplink node แล้วตั้งค่าโดยตรง Name และ Device ID ให้ใส่ Device ID ตามหน้า DEVICE OVERVIEW ของ The Thing Network Console ส่วนตรง App ให้กดที่รูปดินสอ ดังรูปที่ 3.18 ซึ่งตรง App ID ให้ใส่ Application ID ตามหน้า DEVICE OVERVIEW ของ The Thing Network Console ส่วนตรง Access Key ให้ใส่ ACCESS KEYS ตามหน้า APPLICATION OVERVIEW ของ The Thing Network Console และส่วน Discovery address ให้ใส่ discovery.thethingsnetwork.org:1900 แล้วจึงกด Update และ Done ดังรูปที่ 3.19



รูปที่ 3.18 การตั้งค่า ttn uplink node หน้าที่ 1

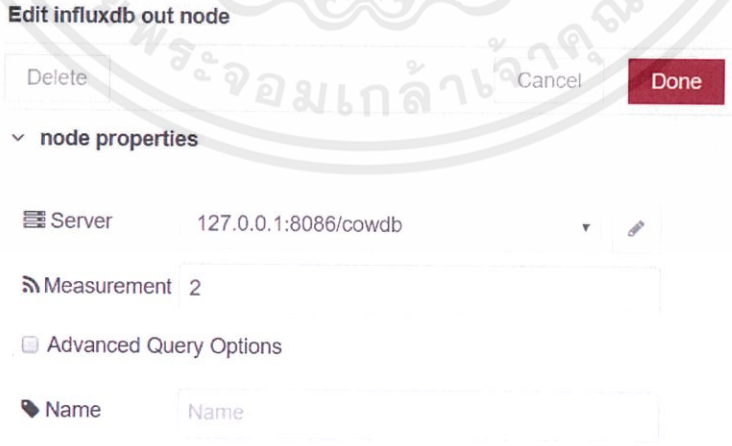
รูปที่ 3.19 การตั้งค่า ttn uplink node หน้าที่ 2

7) ดับเบิ้ลคลิกที่ function node ตรงหัวข้อ Function ให้ใส่ msg. payload.receivedString = Number(msg.payload.receivedString); ในบรรทัดที่ 1 จากนั้นใส่ msg. payload = msg.payload.receivedString; ในบรรทัดที่ 2 และใส่ return msg; ในบรรทัดที่ 3 จากนั้นกด Done ดังรูปที่ 3.20

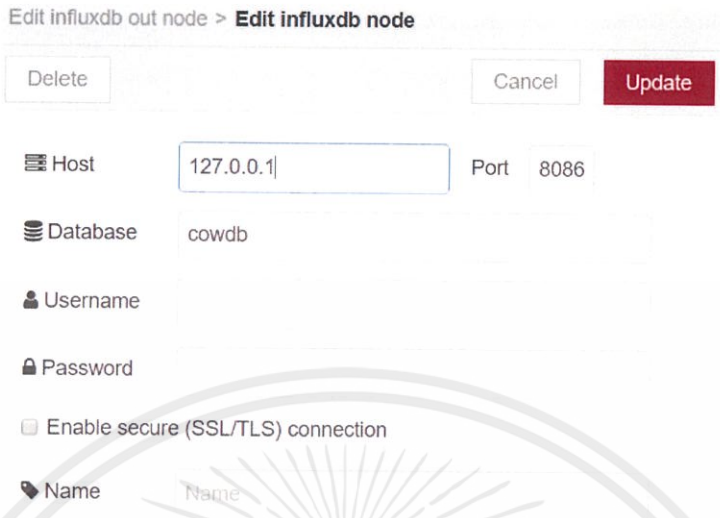


รูปที่ 3.20 การตั้งค่า function node

8) ดับเบิ้ลคลิกที่ Influxdb node ตรงหัวข้อ Server ให้กดรูปดินสอ จากนั้นตรง Host ให้ใส่ 127.0.0.1 ส่วน Port ให้ใส่ 8086 และตรง database ใส่ชื่อฐานข้อมูล ซึ่งในที่นี้คือ cowdb และกด Update ดังรูปที่ 3.21 และ 3.22 ส่วนตรงหัวข้อ Measurement ให้ใส่ชื่อส่วนย่อยของ DATABASE ที่ใช้สำหรับเก็บข้อมูล ในที่นี้คือ 2 แล้วจึงกด Done



รูปที่ 3.21 การตั้งค่า Influxdb node หน้าที่ 1



รูปที่ 3.22 การตั้งค่า Influxdb node หน้าที่ 2

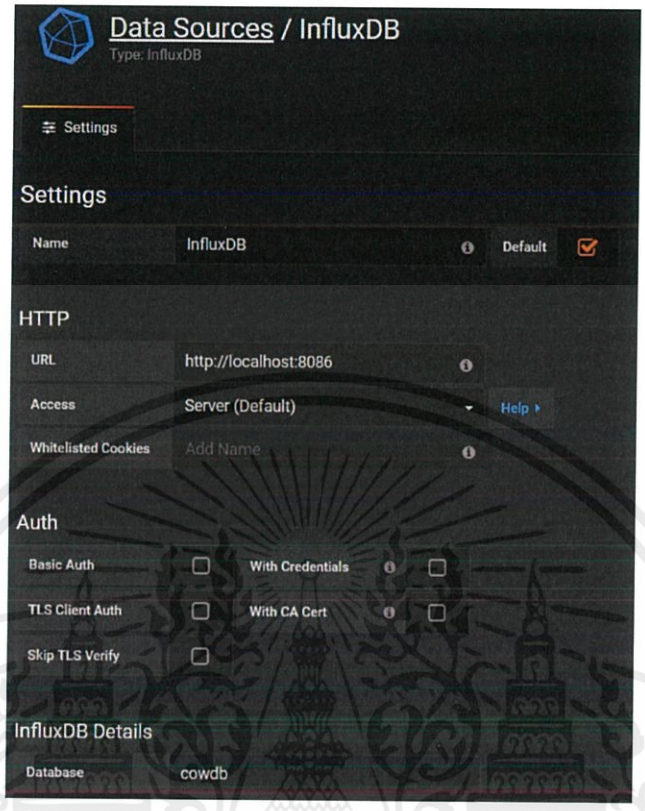
3.1.6 การออกแบบ Monitoring Server (Grafana) ของ LoRaWAN [12]

Grafana เป็นระบบที่ใช้สำหรับดูข้อมูลใน Database โดยแสดงผลในรูปแบบต่าง ๆ เช่น กราฟ หรือตาราง เป็นต้น และยังสามารถตั้งค่าให้สามารถแจ้งเตือนได้อีกด้วย

1) ไปที่หน้าต่าง Command Line ของ Ubuntu ทำการติดตั้ง Grafana ด้วยคำสั่ง wget https://dl.grafana.com/oss/release/grafana_6.0.1_amd64.deb ตามด้วยคำสั่ง sudo dpkg -i grafana_6.0.1_amd64.deb จากนั้นใช้คำสั่ง sudo service grafana-server start เพื่อเริ่มการทำงานของ Grafana ต่อมาพิมพ์คำสั่ง systemctl daemon-reload ตามด้วยคำสั่ง systemctl start grafana-server ตามด้วยคำสั่ง systemctl status grafana-server และตามด้วยคำสั่ง sudo systemctl enable grafana-server.service เพื่อให้ Grafana ทำงานอัตโนมัติบน VM instances

2) เปิด Grafana บน Web Browser ด้วย http://External IP ของ VM instances:3000 ใช้ Username : admin และ Password : admin ในการเข้าใช้

3) กด Add data source เลือก influxDB จากนั้นจะมาในส่วนของการตั้งค่า ตรงหัวข้อ Name จะใส่ InfluxDB เป็นค่า Default ส่วนตรงหัวข้อ URL จะใส่ http://localhost :8086 ส่วนหัวข้อ Access จะเลือก Server (Default) และตรงหัวข้อ Database จะใส่ชื่อ Database ของ InfluxDB ในที่นี้คือ cowdb และกด Save & Test ดังรูปที่ 3.23



รูปที่ 3.23 การสร้าง data source บน Grafana

4) กดที่เครื่องหมายบวก และสร้าง Dashboard เช่น เลือกกราฟมา ก็สามารถกด Panel Title แล้วกด Edit เพื่อแก้ไขการแสดงผลของกราฟได้ ดังรูปที่ 3.24 โดยตัวเลข 1 ที่อยู่ตรงหัวข้อ FROM คือชื่อ Measurement ส่วน Select สามารถแก้ไขรูปแบบการแสดงกราฟได้ เช่น เปลี่ยนเป็นกราฟที่เป็นค่าเฉลี่ย หรือ ค่า Moving Average เป็นต้น

รูปที่ 3.24 การตั้งค่าการแสดงผลของกราฟ

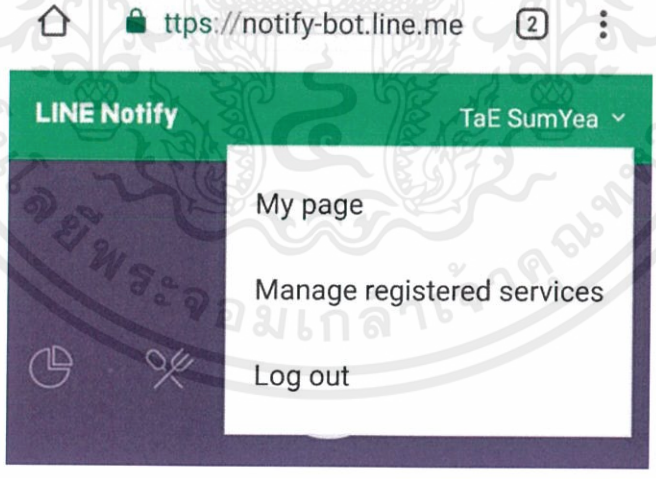
3.1.7 การออกแบบและตั้งค่าการแจ้งเตือนผ่าน Line Notify [13]

Line Notify ใช้สำหรับแจ้งเตือนเหตุการณ์ต่าง ๆ ตามที่กำหนดผ่าน Line ของผู้ใช้งาน สามารถสร้างได้ด้วยขั้นตอนดังนี้

1) เพิ่มเพื่อนที่ชื่อ LINE Notify บน Line ของผู้ใช้งาน ดังรูปที่ 3.25

รูปที่ 3.25 เพื่อนที่ชื่อ LINE Notify

2) เข้าไปที่ https://notify-bot.line.me/th/ จากนั้น login และคลิกที่ My page ดังรูปที่ 3.26

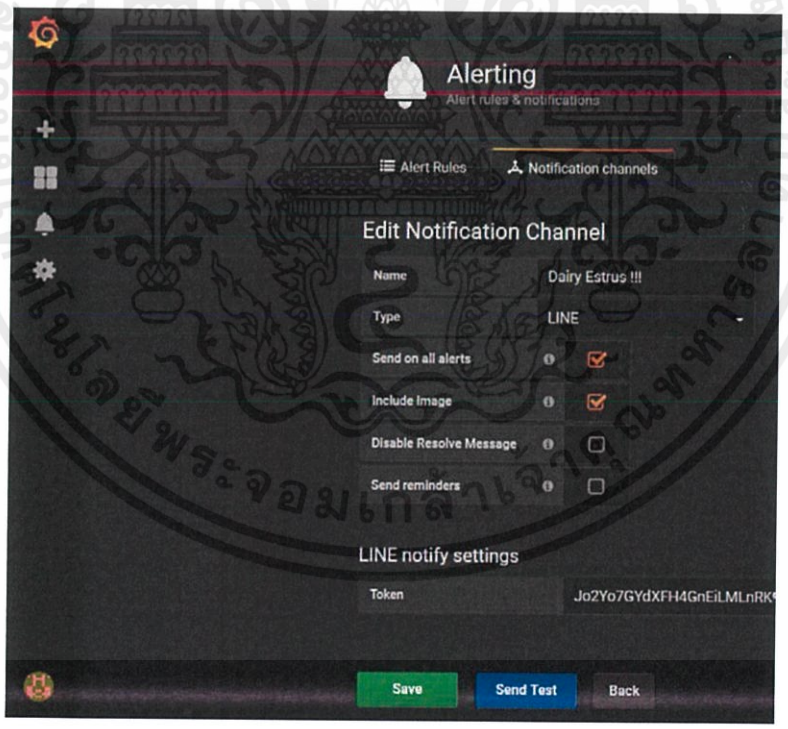


รูปที่ 3.26 การเข้าหน้าตั้งค่า Page

3) กดที่ปุ่ม Generate token ดังรูปที่ 3.27 และเลือกว่าจะส่งข้อมูลให้กับตัวเองแล้วจึงกด Generate Token และทำการ copy รหัส Token เอาไว้ก่อน

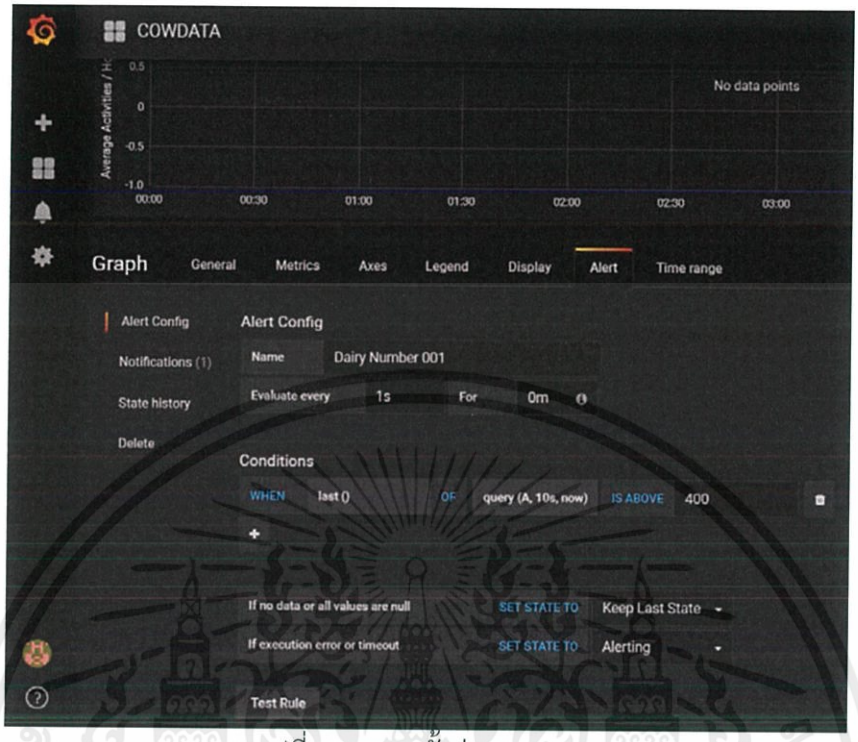
รูปที่ 3.27 Generate access token

4) เข้ามาหน้าเว็บ Grafana ด้วย http://External IP ของ VM instances:3000 กดที่รูปกระดิ่ง และเลือก Notification channels จากนั้นกดที่ ปุ่ม +New Channel และตั้งค่า ดังรูปที่ 3.28 โดย Name ให้ใส่ตามต้องการ ส่วน Type เลือก Line และนำ Token ที่คัดลอกไว้ มาวางในช่อง Token



รูปที่ 3.28 การตั้งค่า Notification channels

5) ไปที่หน้า Dashboard แล้วจะสามารถตั้งค่าต่าง ๆ ในหัวข้อใหญ่ที่ชื่อ Alert ได้ ดังรูปที่ 3.29



รูปที่ 3.29 การตั้งค่าการ Alert

6) สามารถแก้ไขข้อความที่จะให้แจ้งเตือนได้ในหน้า Notifications ด้านซ้ายมือ ดังรูปที่ 3.30

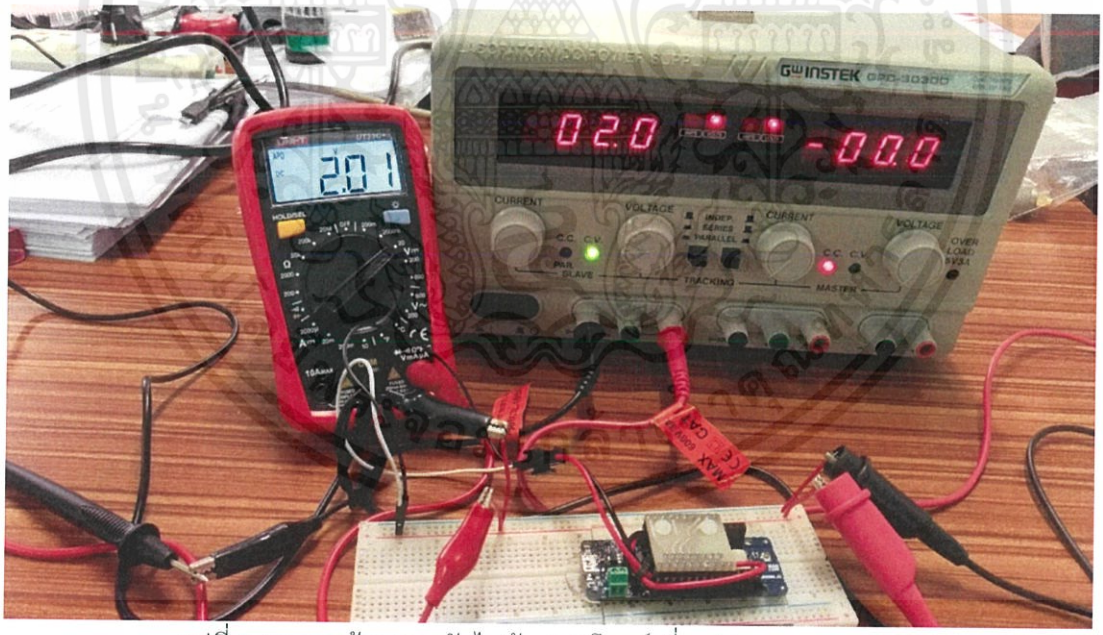
รูปที่ 3.30 การตั้งค่าข้อความที่แสดงขณะ Alert

3.1.8 การออกแบบแบตเตอรี่

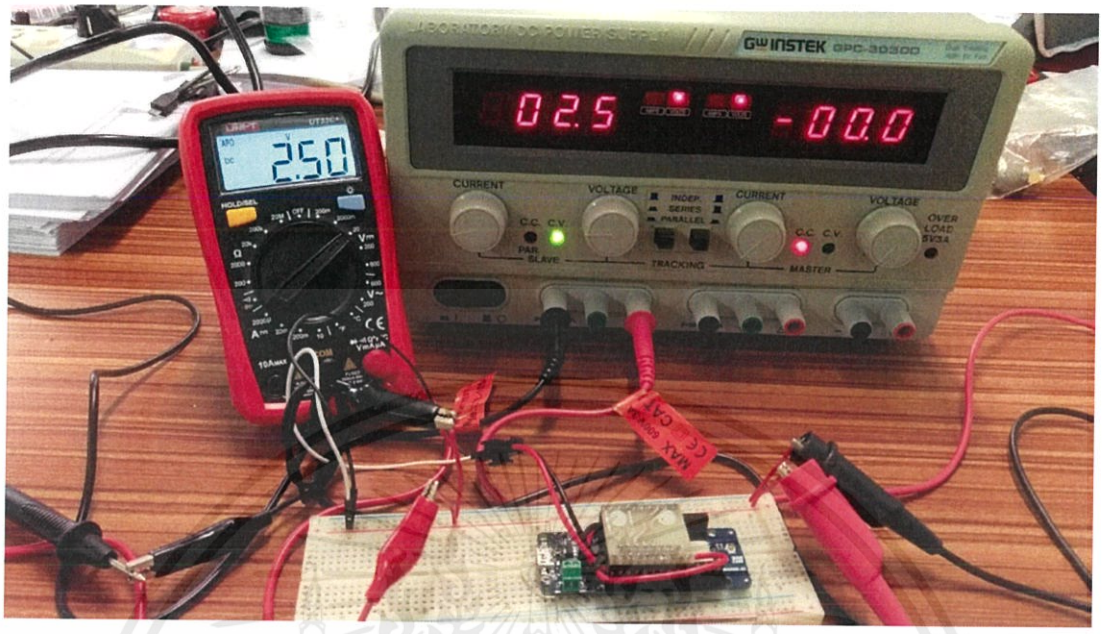
จากการศึกษาบอร์ด Arduino MKR WAN 1300 ทางผู้ผลิตได้แจ้งคุณสมบัติของบอร์ดที่เกี่ยวข้องกับการใช้พลังงานของบอร์ดว่า มีการใช้แรงดันเพื่อที่จะให้บอร์ดเริ่มทำงานอยู่ในช่วง 3.3 – 5.0 โวลต์ โดยสามารถรับได้สูงสุดถึง 6 โวลต์ และการใช้กระแสไฟฟ้าน้อยกว่า 20 มิลลิแอมแปร์ ตามเอกสารของผู้ผลิต โดยยังไม่มีเซนเซอร์ที่เชื่อมต่อเข้ากับบอร์ด ดังนั้นเพื่อที่จะทำให้ทราบถึงการใช้พลังงานโดยรวมทั้งหมดของอุปกรณ์ และเพื่อที่จะทำการออกแบบแบตเตอรี่ให้มีการจ่ายพลังงานที่เพียงพอให้กับอุปกรณ์นั้น จึงต้องทำการทดสอบการใช้พลังงานของอุปกรณ์ โดยมีการทดสอบความต้องการแรงดันไฟฟ้าและกระแสไฟฟ้า โดยมีรายละเอียดดังต่อไปนี้

3.1.8.1 ทดสอบความต้องการแรงดันไฟฟ้า

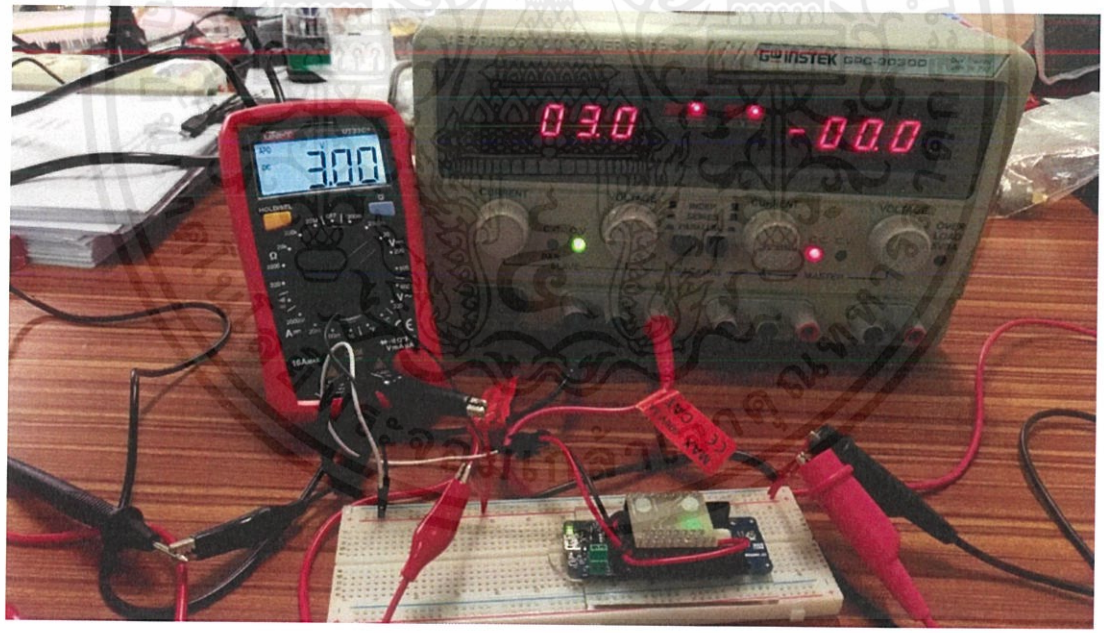
จากการทดสอบเพื่อหาความต้องการแรงดันไฟฟ้าโดยการป้อนแรงดันเข้าที่ขา Vin และ Ground ด้วยแหล่งจ่ายไฟฟ้ากระแสตรง (DC Power Supply) ที่มีค่าโดยประมาณเป็น 2.0, 2.5, 3.0, 3.3, 3.7, 4.0 และ 5.0 โวลต์ ดังรูปที่ 3.31 – 3.37 ตามลำดับ พบว่าอุปกรณ์ จะเริ่มทำงานที่ 3.0 โวลต์ เป็นต้นไป สังเกตได้จากไฟสีเขียวที่สว่างอยู่บนตัวอุปกรณ์



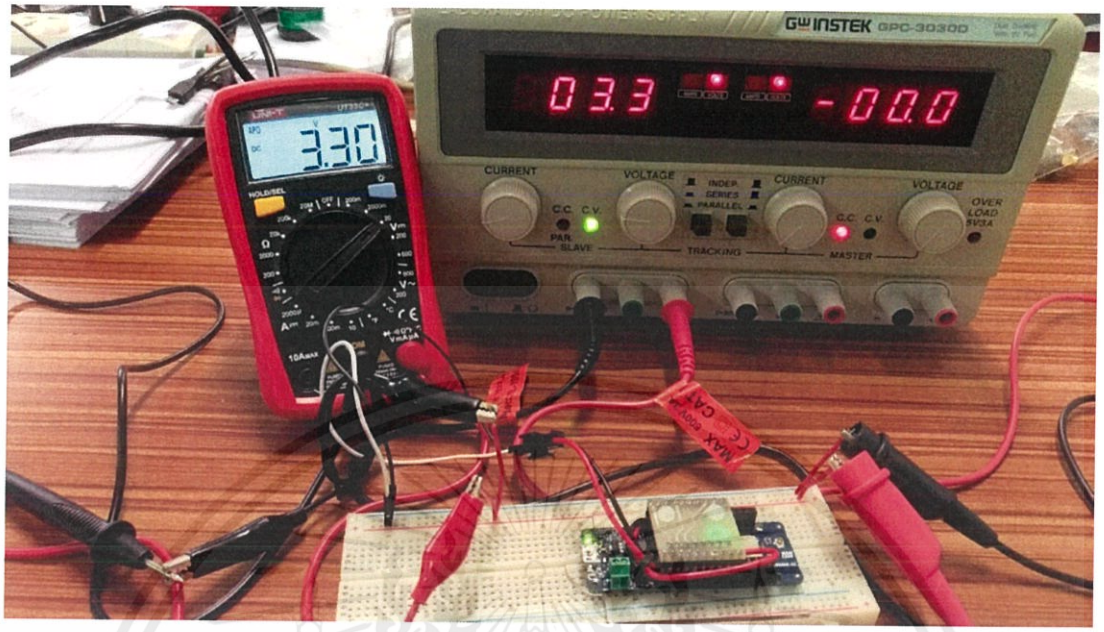
รูปที่ 3.31 การป้อนแรงดันไฟฟ้า 2.0 โวลต์ ที่ขา Vin และ Ground



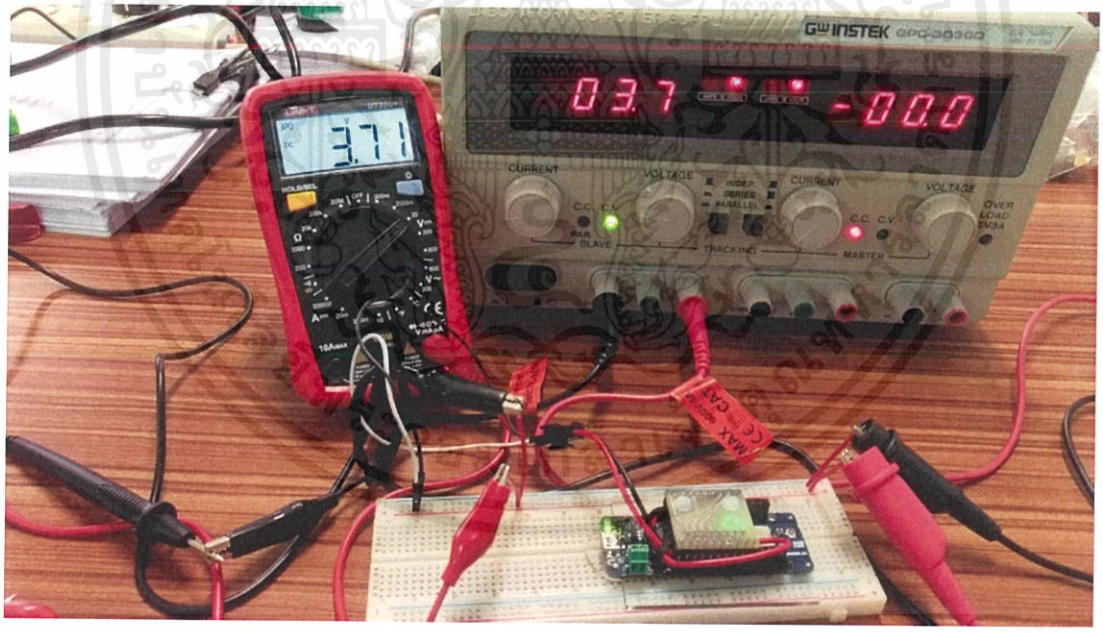
รูปที่ 3.32 การป้อนแรงดันไฟฟ้า 2.5 โวลต์ ที่ขา Vin และ Ground



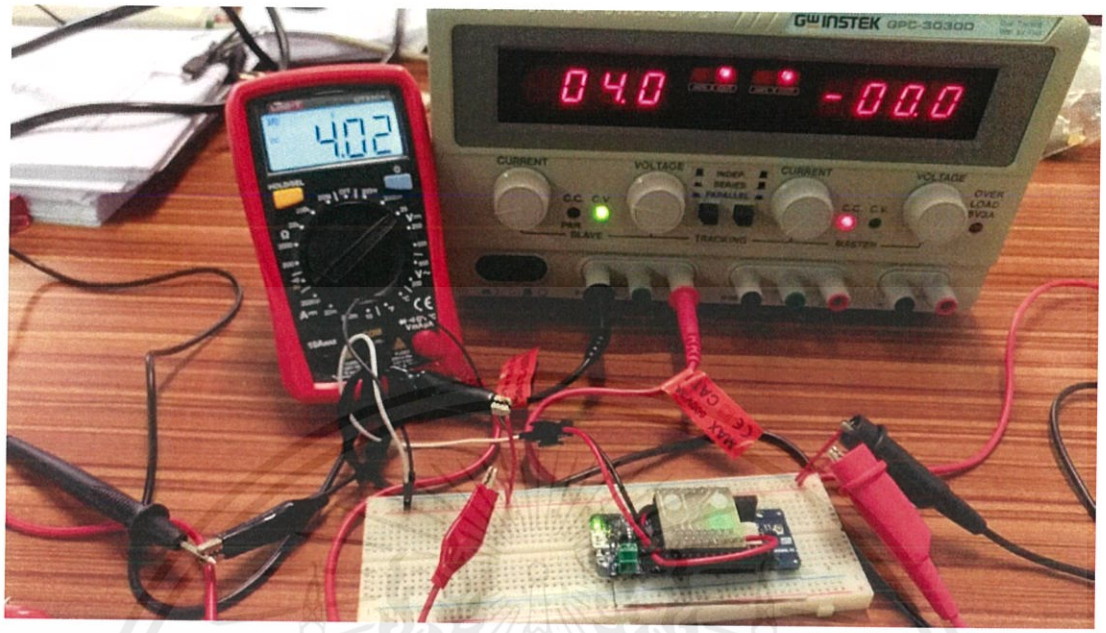
รูปที่ 3.33 การป้อนแรงดันไฟฟ้า 3.0 โวลต์ ที่ขา Vin และ Ground



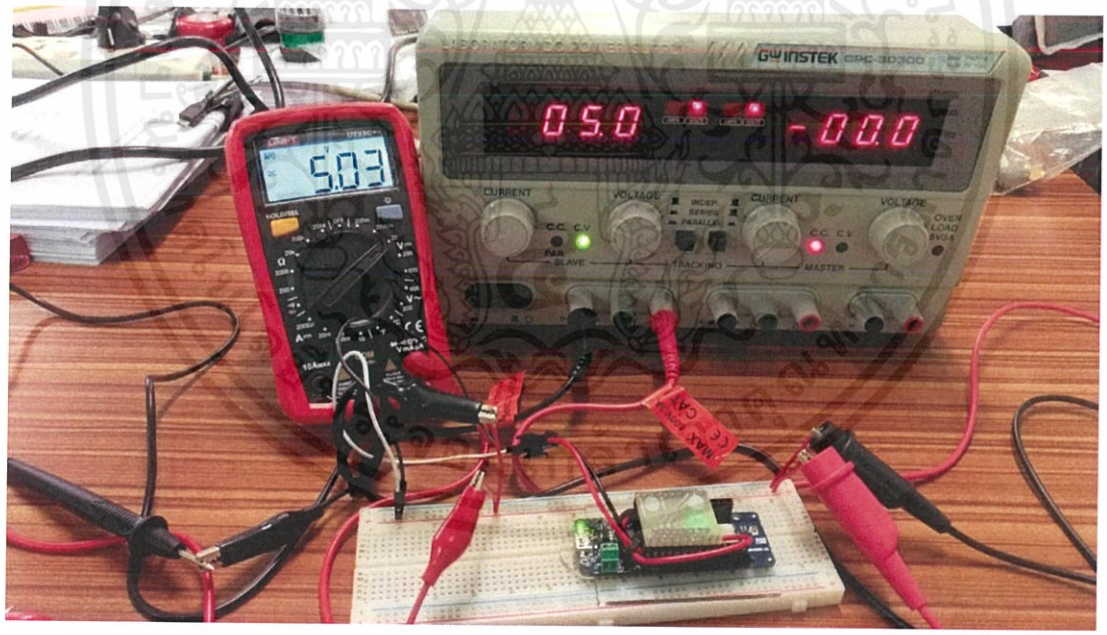
รูปที่ 3.34 การป้อนแรงดันไฟฟ้า 3.3 โวลต์ ที่ขา Vin และ Ground



รูปที่ 3.35 การป้อนแรงดันไฟฟ้า 3.7 โวลต์ ที่ขา Vin และ Ground

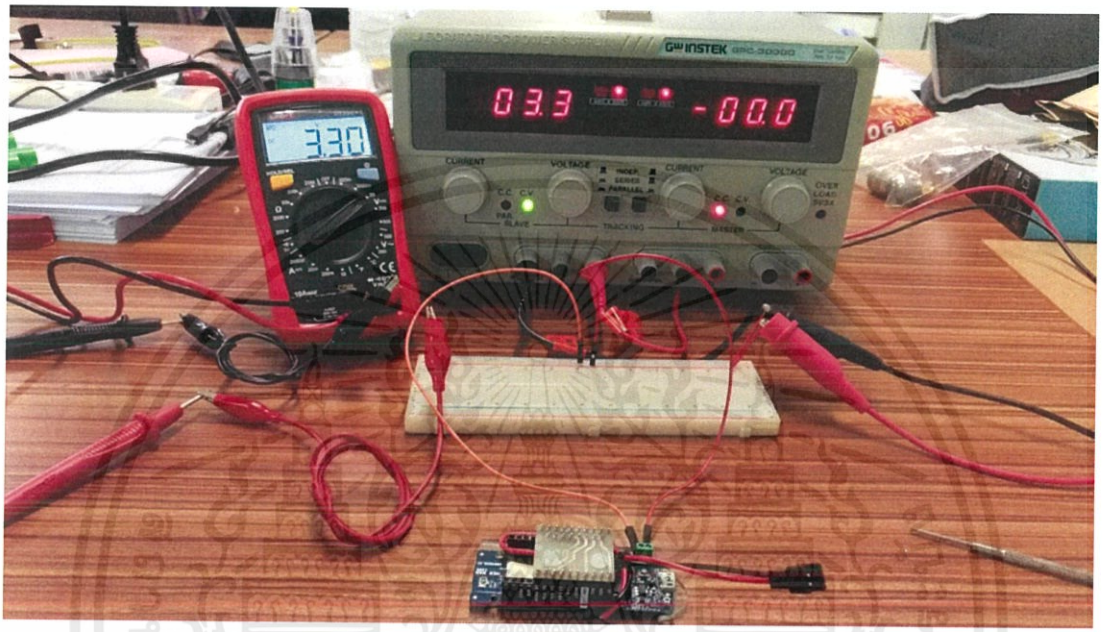


รูปที่ 3.36 การป้อนแรงดันไฟฟ้า 4.0 โวลต์ ที่ขา Vin และ Ground

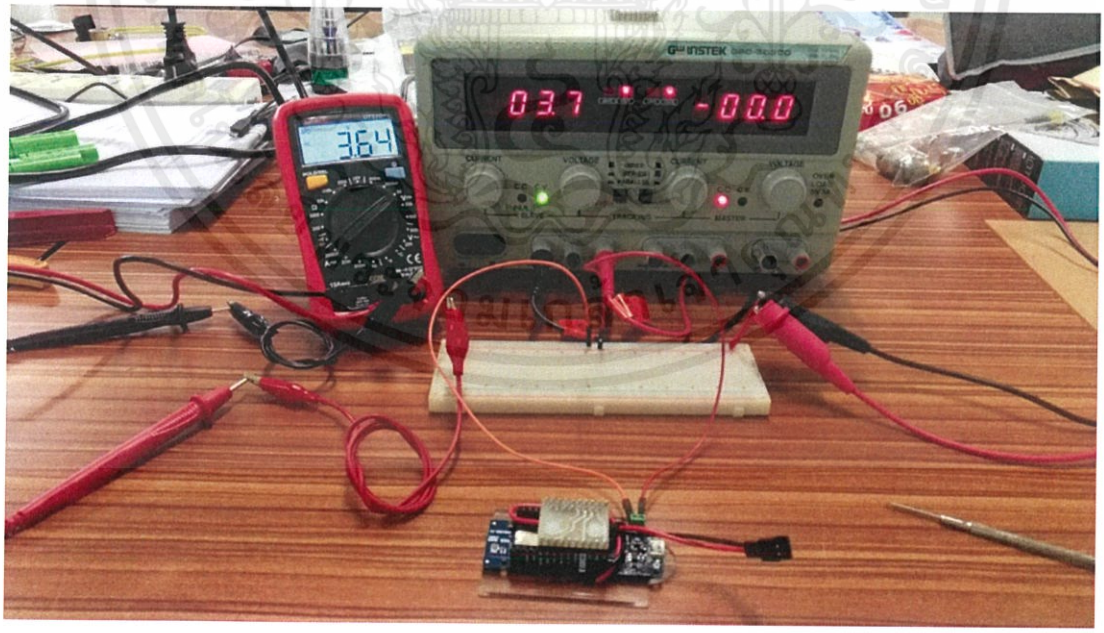


รูปที่ 3.37 การป้อนแรงดันไฟฟ้า 5.0 โวลต์ ที่ขา Vin และ Ground

จากการทดสอบเพื่อหาความต้องการแรงดันไฟฟ้าโดยการป้อนแรงดันเข้าที่เทอร์มินอลบล็อกสีเขียวโดยการป้อนแรงดันด้วยแหล่งจ่ายไฟฟ้ากระแสตรงที่ 3.3 และ 3.7 โวลต์ พบว่าอุปกรณ์ไม่ทำงานตามเอกสารของผู้ผลิตที่ได้กล่าวไว้ ดังรูปที่ 3.38 และ 3.39 ตามลำดับ



รูปที่ 3.38 การป้อนแรงดันไฟฟ้า 3.3 โวลต์ ที่เทอร์มินอลบล็อกสีเขียว



รูปที่ 3.39 การป้อนแรงดันไฟฟ้า 3.7 โวลต์ ที่เทอร์มินอลบล็อกสีเขียว

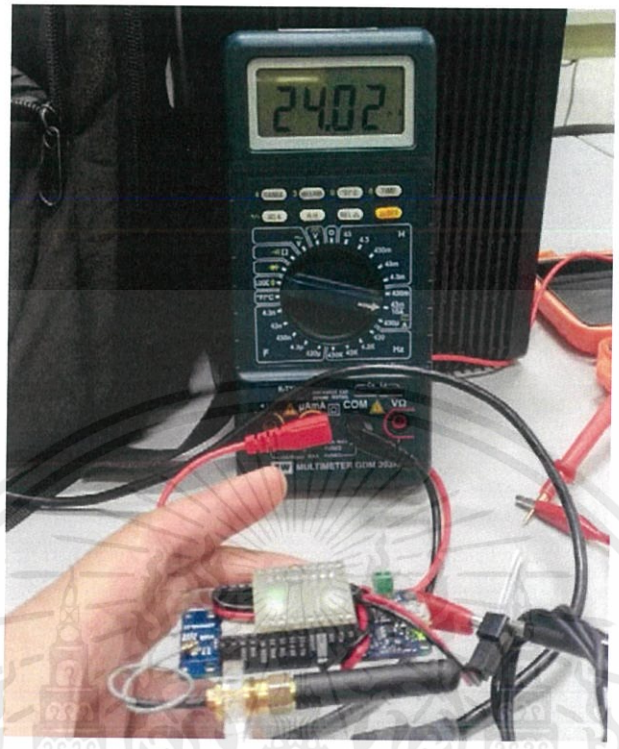
3.1.8.2 ทดสอบความต้องการกระแสไฟฟ้า

จากการทดสอบเพื่อหาความต้องการกระแสไฟฟ้าโดยการป้อนแรงดันไฟฟ้าเข้าที่ขา Vin และ Ground ด้วยแหล่งจ่ายไฟฟ้ากระแสตรงที่มีค่าโดยประมาณเป็น 4.0 โวลต์ พบว่ากระแสไฟฟ้าที่อุปกรณ์ต้องการโดยประมาณนั้นอยู่ในช่วง 24.0 – 33.0 มิลลิแอมแปร์ ซึ่งการทดสอบได้แบ่งโดยละเอียดออกเป็น 2 ช่วง คือ ช่วงการทำงานของเซนเซอร์ และช่วงการส่งข้อมูล

1) ช่วงการทำงานของเซนเซอร์

ทดสอบการวัดความต้องการกระแสไฟฟ้าของอุปกรณ์ทุก ๆ 1 นาที จำนวน 5 ครั้ง รวมเป็น 5 นาที ผลที่ได้นั้นแสดงดังรูปที่ 3.40 – 3.44 ผลลัพธ์โดยรวมแล้วตัวอุปกรณ์จะมีความต้องการกระแสไฟฟ้าไม่เกิน 24.2 มิลลิแอมแปร์

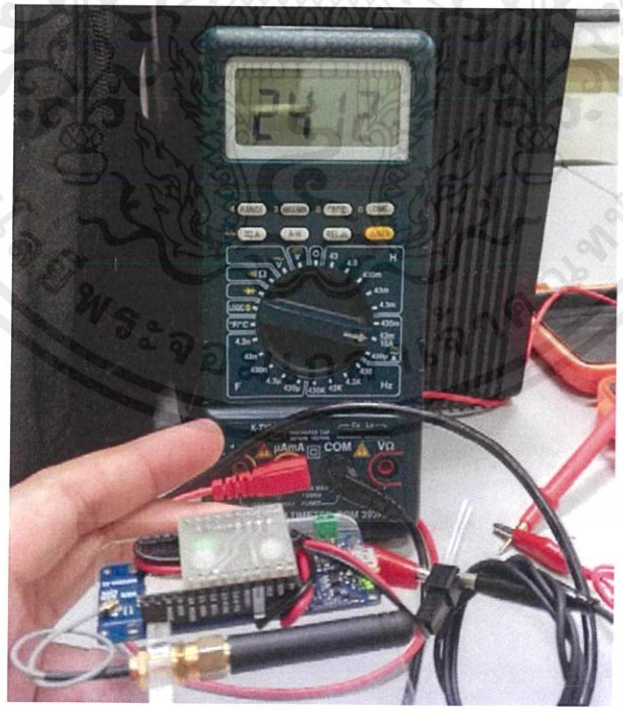
รูปที่ 3.40 ความต้องการกระแสไฟฟ้าในช่วงการทำงานของเซนเซอร์นาทีที่ 1



รูปที่ 3.41 ความต้องการกระแสไฟฟ้าในช่วงการทำงานของเซนเซอร์นาทีที่ 2

รูปที่ 3.42 ความต้องการกระแสไฟฟ้าในช่วงการทำงานของเซนเซอร์นาทีที่ 3

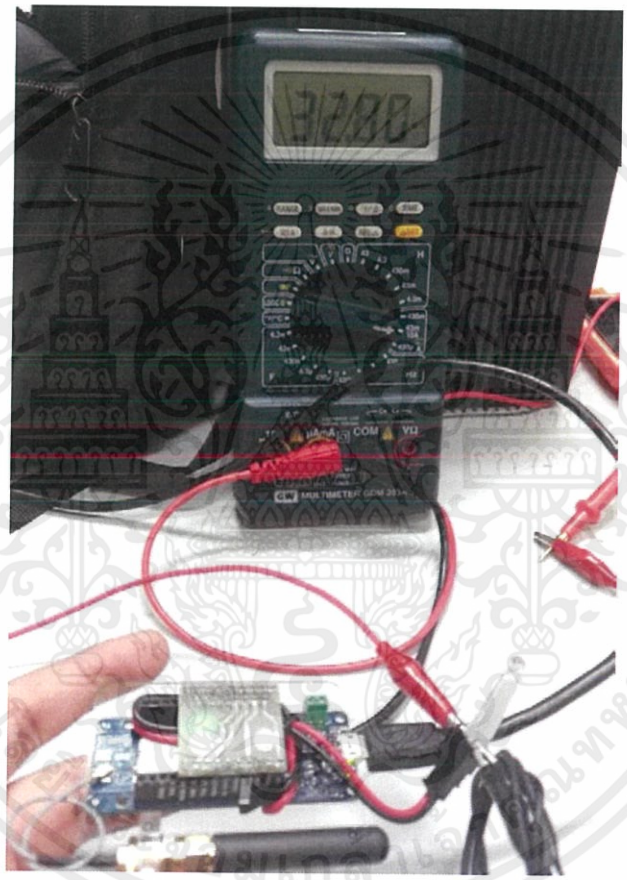
รูปที่ 3.43 ความต้องการกระแสไฟฟ้าในช่วงการทำงานของเซนเซอร์นาทีที่ 4



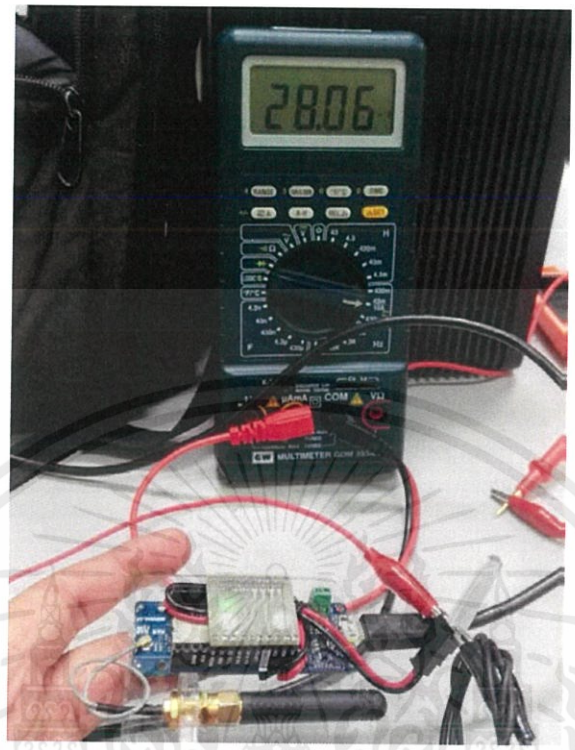
รูปที่ 3.44 ความต้องการกระแสไฟฟ้าในช่วงการทำงานของเซนเซอร์นาทีที่ 5

2) ช่วงการส่งข้อมูล

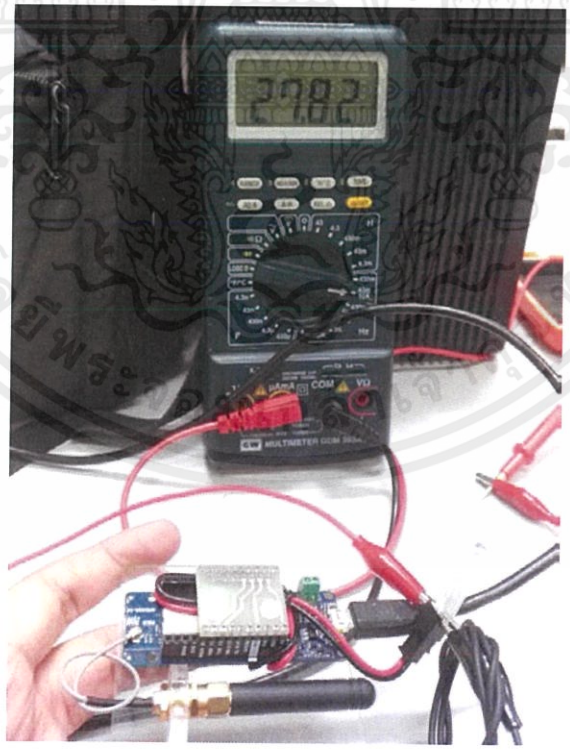
ทดสอบการวัดความต้องการกระแสไฟฟ้าของอุปกรณ์ในช่วงการส่งข้อมูลจำนวน 3 ครั้ง ผลที่ได้นั้นแสดงดังรูปที่ 3.45 – 3.47 กระแสที่เกิดขึ้นนั้นเกิดขึ้นเพียงเสี้ยววินาที โดยรวมแล้วตัวอุปกรณ์จะมีความต้องการกระแสไฟฟ้าในช่วงการส่งข้อมูลประมาณ 27.0 – 33.0 มิลลิแอมแปร์



รูปที่ 3.45 ความต้องการกระแสไฟฟ้าในช่วงการส่งข้อมูลครั้งที่ 1



รูปที่ 3.46 ความต้องการกระแสไฟฟ้าในช่วงการส่งข้อมูลครั้งที่ 2



รูปที่ 3.47 ความต้องการกระแสไฟฟ้าในช่วงการส่งข้อมูลครั้งที่ 3

การทดสอบในหัวข้อ 3.1.5 นั้น ทำให้ได้ข้อมูลในการเลือกแบตเตอรี่ที่มีความเหมาะสมต่อความต้องการของอุปกรณ์ แบตเตอรี่ที่เลือกใช้ควรมีแรงดันไฟฟ้ามากกว่า 3.0 โวลต์ แต่ไม่เกิน 6 โวลต์ ส่วนความจุของแบตเตอรี่ที่เลือกใช้สามารถคำนวณหาได้จากความต้องการกระแสไฟฟ้าของอุปกรณ์ ซึ่งอุปกรณ์นั้นมีความต้องการกระแสไฟฟ้าประมาณ 24.0 มิลลิแอมแปร์

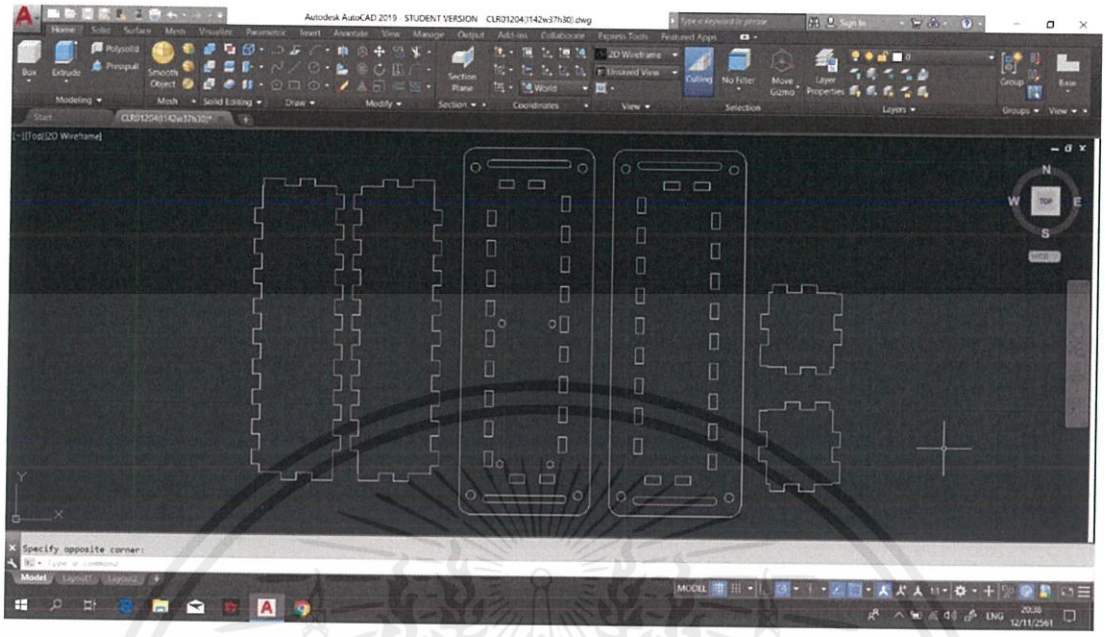
แบตเตอรี่ที่เลือกใช้จึงเป็นลิเทียมโพลิเมอร์แบตเตอรี่ (Lithium Polymer Battery) ที่มีความจุ 10,000 มิลลิแอมแปร์ชั่วโมง และแรงดันไฟฟ้า 3.7 โวลต์ จากนั้นคำนวณหาว่าอุปกรณ์จะทำงานได้เป็นระยะเวลานานเท่าใด โดยการนำค่าความจุของแบตเตอรี่มาส่วนด้วยความต้องการกระแสไฟฟ้าของอุปกรณ์จะได้ระยะเวลาออกมาเป็น 416 ชั่วโมง 16 นาที แปลงเป็นวันได้ 17 วัน 8 ชั่วโมง 16 นาที เป็นไปได้ตามทฤษฎีเท่านั้น ดังนั้นจะให้ทำการนำมาชาร์จทุก ๆ 15 วัน เพื่อป้องกันการหมดประจุของแบตเตอรี่ก่อนเวลา และเป็นการดูแลรักษาแบตเตอรี่ไม่ให้หมดอายุการใช้งานก่อนเวลาอันควร

3.1.9 การออกแบบกล่องบรรจุเครื่องวัดปริมาณการเคลื่อนไหวของโคนม

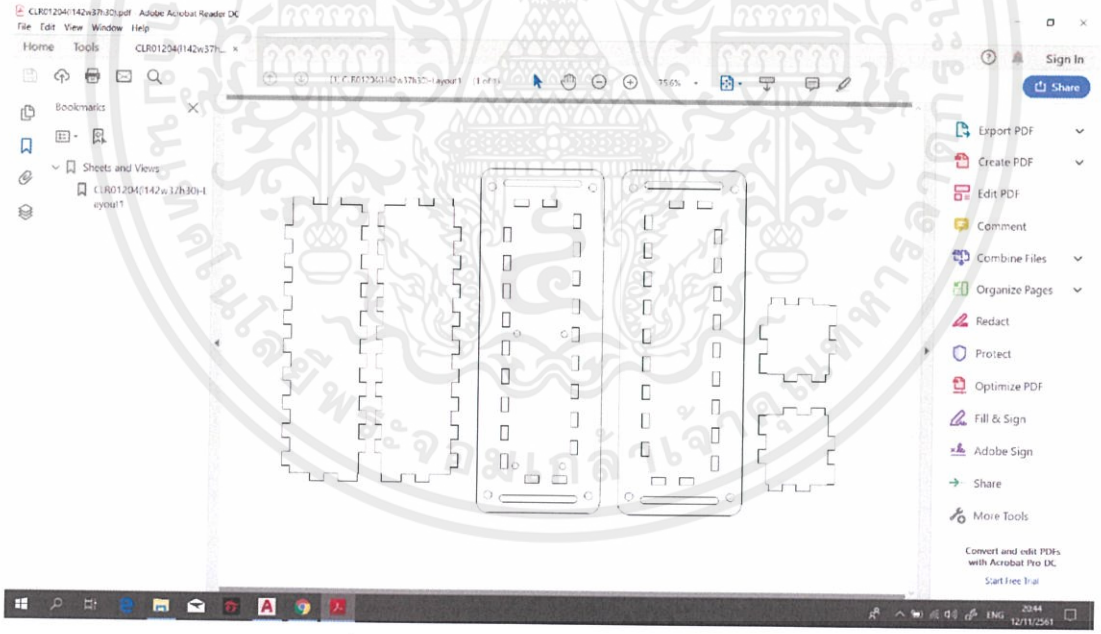
เมื่อสร้างเครื่องวัดปริมาณการเคลื่อนไหวของโคนมด้วยการประกอบอุปกรณ์ทุกส่วนเข้าด้วยกัน จากนั้นจะเป็นการนำอุปกรณ์ทั้งหมดไปติดกับโคนม เพื่อไม่ให้เกิดความเสียหายต่ออุปกรณ์จึงจำเป็นต้องมีเครื่องป้องกันอุปกรณ์ ผู้จัดทำจึงออกแบบในลักษณะกล่องและมีสายคล้องเหมือนกับปลอกคอ ซึ่งการออกแบบนั้นก็ได้มีการคิดและพัฒนามาตลอดเพื่อให้มีความเหมาะสมและประหยัดค่าใช้จ่ายมากที่สุด

1) การออกแบบครั้งที่ 1

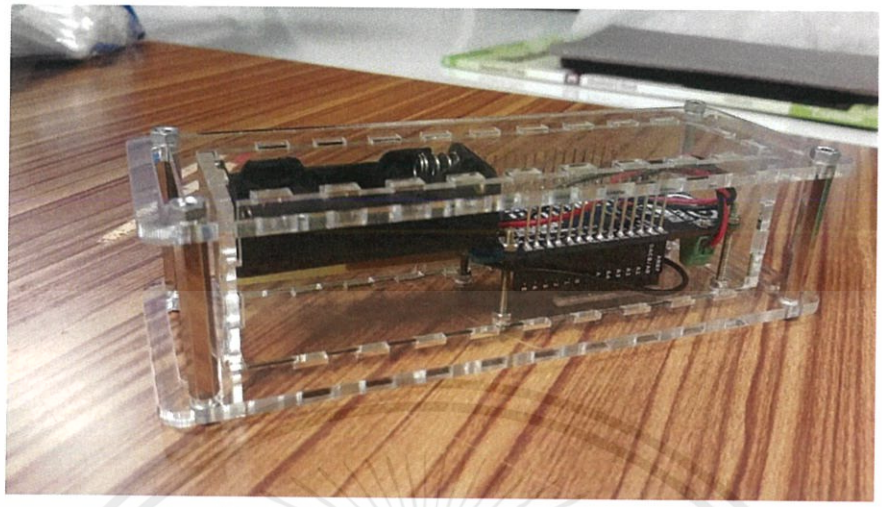
ครั้งแรกนั้นได้ใช้วัสดุในการสร้างกล่องบรรจุเครื่องวัดปริมาณการเคลื่อนไหวของโคนมเป็นแผ่นอะคริลิค และออกแบบด้วยโปรแกรม AutoCAD2019 ดังรูปที่ 3.48 เมื่อออกแบบเสร็จสิ้นจะทำการบันทึกไฟล์ให้อยู่ในนามสกุล .pdf ดังรูปที่ 3.49 และนำไฟล์นี้ไปทำการใช้เครื่องเลเซอร์แกะสลักเพื่อทำการตัดแผ่นอะคริลิคให้ได้ตามที่ออกแบบไว้ จากนั้นนำอุปกรณ์มาประกอบด้วยสกรู และบรรจุอุปกรณ์ลงในกล่องเป็นอันเสร็จเรียบร้อยดังรูปที่ 3.50



รูปที่ 3.48 โปรแกรม AutoCAD2019



รูปที่ 3.49 ไฟล์ที่ถูกบันทึกเป็นนามสกุล .pdf



รูปที่ 3.50 กล่องที่บรรจุอุปกรณ์เรียบร้อยแล้ว

ผลการทดสอบเมื่อนำไปทดสอบกับโคนมพบว่า กล่องนั้นไม่สามารถกันความชื้นได้ เพราะพบความชื้นจากตัวโคนมเข้าไปภายในกล่อง จึงต้องทำการแก้ไขกล่องให้สามารถกันน้ำและความชื้นได้

2) การออกแบบครั้งที่ 2

จากผลการทดสอบครั้งแรกที่พบพบความชื้นจากตัวโคนมเข้าไปภายในกล่องทำให้เกิดการปรับปรุงแก้ไขโดยทำการออกแบบด้วยโปรแกรม AutoCAD2019 และใช้วัสดุเป็นอะคริลิคดังเดิมแต่การประกอบนั้นจะใช้น้ำยาประสานอะคริลิคทำหน้าที่ยึดแผ่นอะคริลิคแต่ละส่วนเข้าด้วยกันและทำการยาแนวด้วยซิลิโคนอีกครั้งเพื่อกันความชื้นและกันน้ำ ดังรูปที่ 3.51

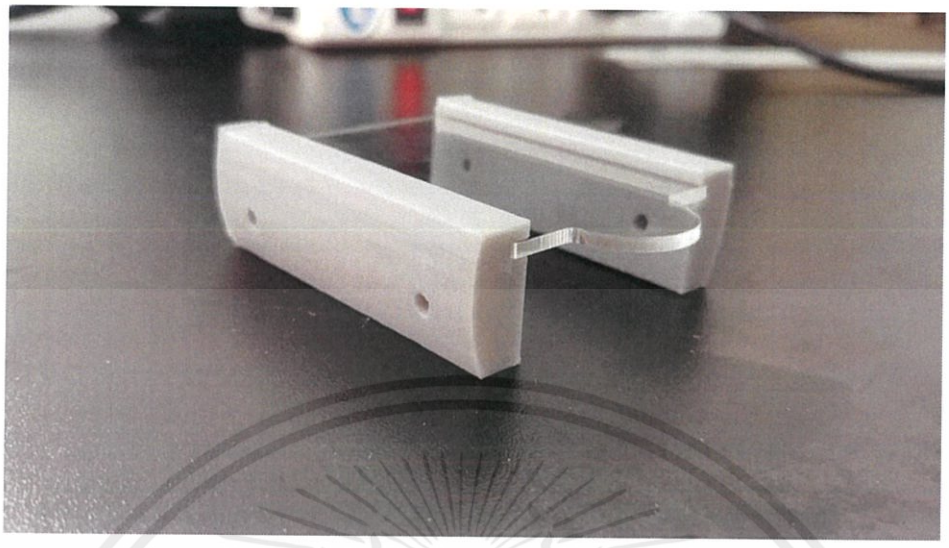
รูปที่ 3.51 การออกแบบครั้งที่ 2

จากผลการทดสอบครั้งที่สองเมื่อทดสอบในเรื่องของการกันน้ำ สามารถกันน้ำได้จริง แต่เมื่อทดสอบกล่องบรรจุอุปกรณ์กับโคนมในสภาพแวดล้อมจริงพบปัญหาเรื่องความร้อนจากอุปกรณ์ภายใน และความร้อนจากภายนอกที่ไม่สามารถป้องกันได้มีผลทำให้ตัวอุปกรณ์รีสตาร์ทตัวเอง อีกทั้งยังมีเรื่องความแข็งแรงของกล่องเนื่องจากวัสดุที่ใช้ทำกล่องเป็นอะคริลิคจึงทำให้กล่องมีความแข็งแต่เปราะไม่สามารถรับแรงกระแทกได้มากจนทำให้หูที่คล้องสายตัวกล่องบรรจุนั้นแตกหัก ซึ่งเป็นอันตรายหากโคนมกินเข้าไป

3) การออกแบบครั้งที่ 3

จากผลการทดสอบครั้งที่สองพบปัญหากล่องเกิดการกระแทกเสียหายจึงนำมาคิดปรับปรุงกล่องอีกครั้ง ดังนั้นเพื่อเสริมความยืดหยุ่นไม่ให้เกิดการแตกหัก วัสดุที่เลือกนำมาใช้ใส่อุปกรณ์ใหม่จึงเป็นกระปุกใส่ยาของสัตว์ที่เหลือใช้ภายในบ้านที่ทำมาจากพลาสติก ดังรูปที่ 3.52 โดยออกแบบอุปกรณ์เสริมจาก 3D Printer และแผ่นอะคริลิคจากเครื่องเลเซอร์แกะสลักเพื่อจัดระเบียบให้กับอุปกรณ์ภายในกล่อง ดังรูปที่ 3.53

รูปที่ 3.52 กระปุกใส่ยาของสัตว์ที่ทำเป็นกล่องบรรจุอุปกรณ์



รูปที่ 3.53 อุปกรณ์เสริมเพื่อจัดระเบียบให้กับอุปกรณ์ภายในกล่อง

4) การออกแบบครั้งสุดท้าย

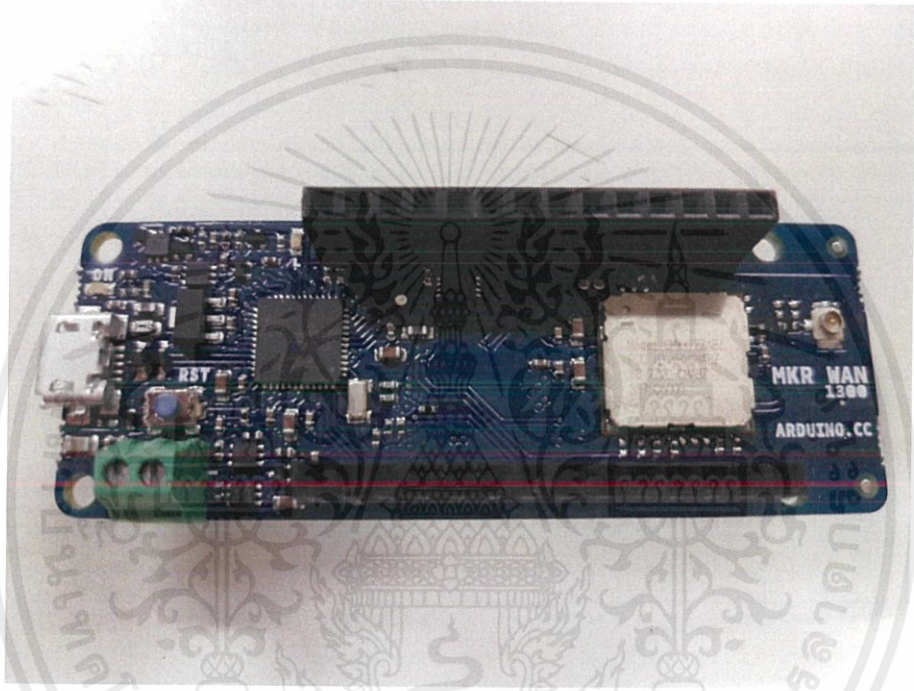
จากผลการทดสอบในการออกแบบครั้งที่สามนั้น พบปัญหาที่เกิดขึ้นคือ หูที่ใช้คล้องสายสำหรับคล้องคอโคนมที่ตัวกล่องนั้นมีความยืดหยุ่นมากเกินไปทำให้กล่องหลุดออกจากสาย และแก้ปัญหาระยะเวลาในการทำงานของอุปกรณ์ที่ขึ้นอยู่กับแบตเตอรี่ จึงทำการเพิ่มแบตเตอรี่ทำให้ต้องขยายขนาดกล่องอีกครั้ง ดังนั้นกล่องใหม่ในครั้งนี้จึงตัดสินใจที่จะเลือกซื้อกล่องสำหรับใส่หูฟังซึ่งมีคุณสมบัติกันน้ำ กันกระแทกได้เป็นอย่างดี ดังรูปที่ 3.54 ทำให้เหมาะสมกับการทดลองนี้มากยิ่งขึ้น

รูปที่ 3.54 กล่องใส่หูฟังที่นำมาประยุกต์ใช้

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 Arduino MKR WAN 1300

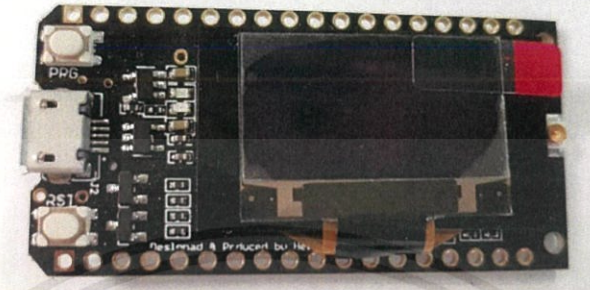
บอร์ด Arduino รุ่น MKR WAN 1300 ใช้เป็นเครื่องวัดปริมาณการเคลื่อนไหวของโคนม และเป็น Node ของ LoRaWAN โดยต้องเชื่อมเข้ากับโมดูล GY-521 และสายอากาศ GSM ดังรูปที่ 3.55



รูปที่ 3.55 บอร์ด Arduino รุ่น MKR WAN 1300

3.2.2 Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora

บอร์ด Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora ใช้เป็น Gateway ของ LoRaWAN โดยต้องเชื่อมต่อกับสายอากาศย่าน 923 MHz ซึ่งตัวบอร์ดจะมี OLED ในตัว ใช้ ESP32 เป็น Microcontroller ใช้ SX1276 เป็น LoRa Modem และสามารถเชื่อมต่อ Wi-Fi และ Bluetooth ได้ ดังรูปที่ 3.56



รูปที่ 3.56 บอร์ด Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora

3.2.3 สาย USB A to micro USB

สาย USB A to micro USB ใช้สำหรับเชื่อมต่อบอร์ด Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora และ Arduino MKR WAN 1300 เข้ากับคอมพิวเตอร์ เพื่อ Upload โปรแกรมลงบอร์ด หรือดูผลลัพธ์ทาง Serial Monitor ดังรูปที่ 3.57

รูปที่ 3.57 สาย USB A to micro USB

3.2.4 Dipole Antenna หรือ GSM Antenna

Dipole Antenna หรือ GSM Antenna (850/900/1800/1900 MHz) ใช้กับบอร์ด Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora หรือ Arduino รุ่น MKR WAN 1300 ดังรูปที่ 3.58

รูปที่ 3.58 Dipole Antenna หรือ GSM Antenna

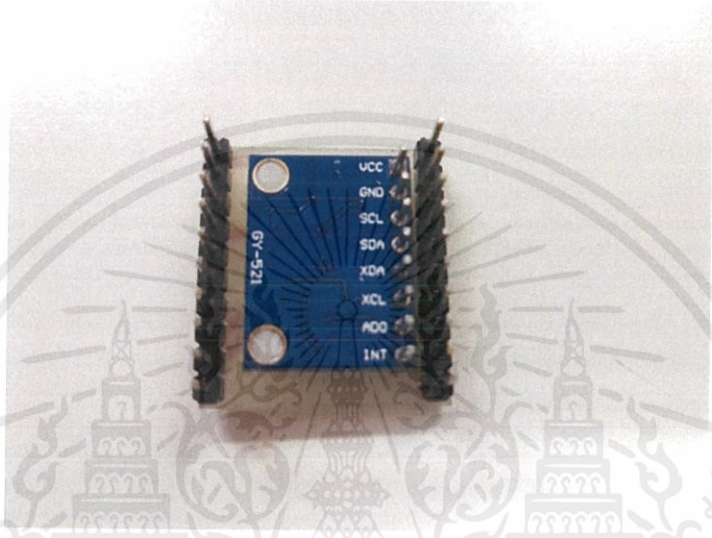
3.2.5 923 MHz helical antenna

923 MHz helical antenna ใช้กับบอร์ด Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora หรือ Arduino MKR WAN 1300 ดังรูปที่ 3.59

รูปที่ 3.59 923 MHz helical antenna

3.2.6 โมดูล GY-521

เป็นโมดูลที่มีทั้ง Accelerometer และ Gyroscope ในตัว และใช้ MPU-6050 ดังรูปที่ 3.60 ใช้ร่วมกับบอร์ด Arduino MKR WAN 1300 สำหรับวัดปริมาณการเคลื่อนไหวของโคนม



รูปที่ 3.60 โมดูล GY-521

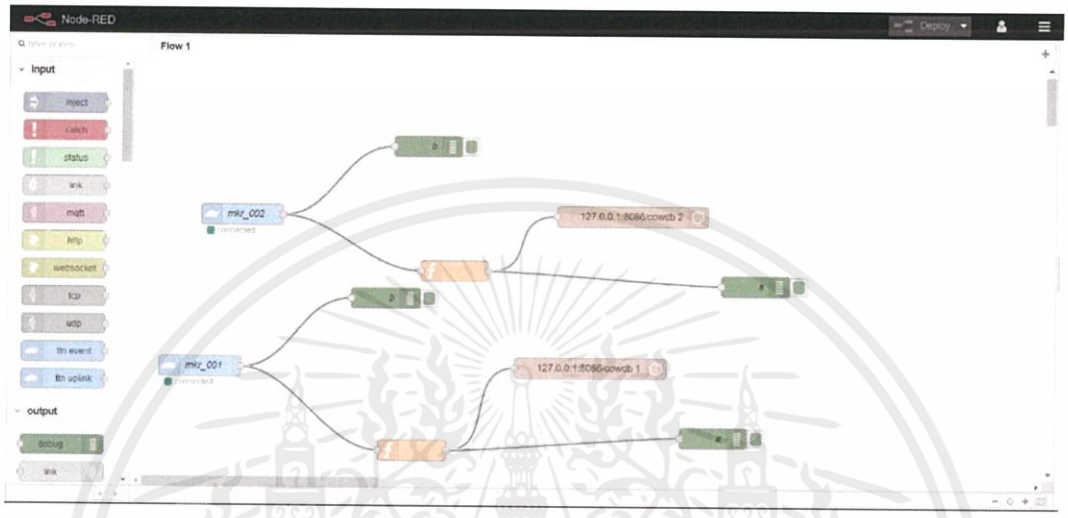
3.2.7 The Thing Network

The Thing Network เป็น Network Server เพื่อเป็นตัวกลางการควบคุมการรับ-ส่งข้อมูลของ Node, Gateway และ Application Server ของ LoRaWAN ดังรูปที่ 3.61

รูปที่ 3.61 The Thing Network

3.2.8 Node Red

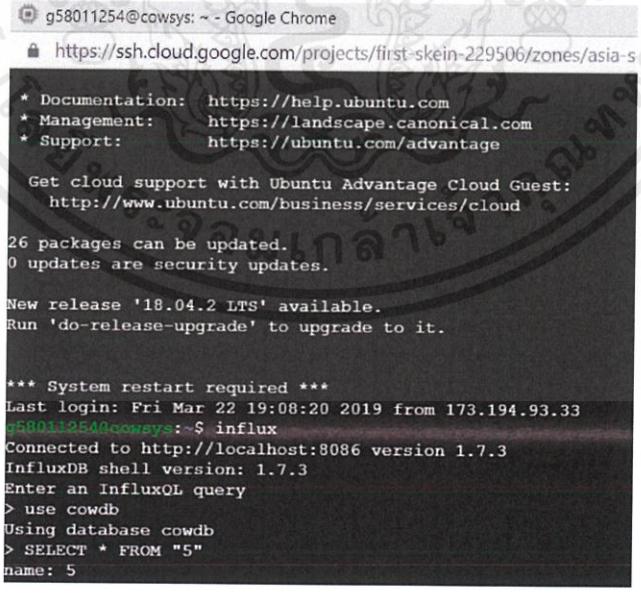
Node Red เป็นตัวกลางระหว่าง Network server และ Application Server ของ LoRaWAN ดังรูปที่ 3.62



รูปที่ 3.62 Node Red

3.2.9 InfluxDB (Database Server)

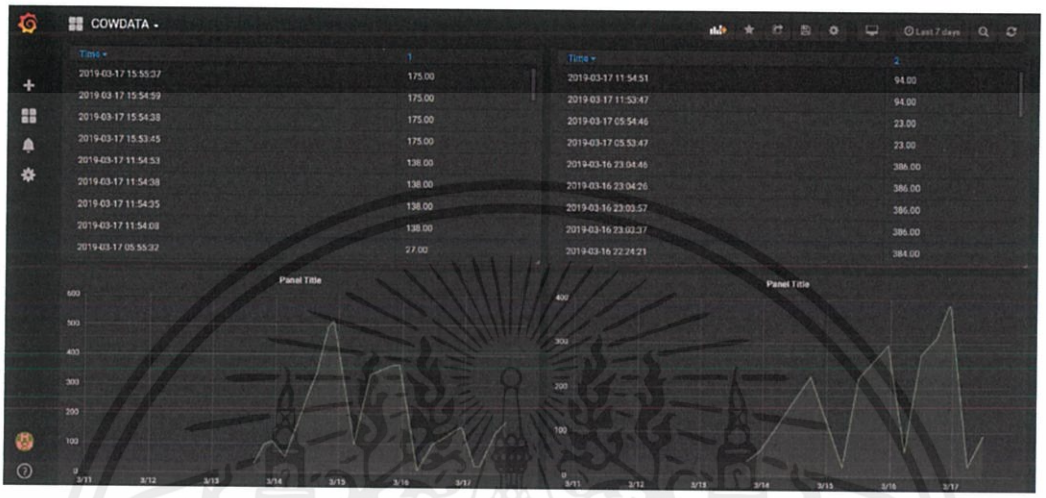
Influxdb หรือ Database Server ใช้สำหรับเก็บข้อมูล โดยสามารถจัดการข้อมูลได้ผ่าน Command Line ของ Ubuntu ดังรูปที่ 3.63



รูปที่ 3.63 InfluxDB (Database Server)

3.2.10 Grafana (Monitoring Server)

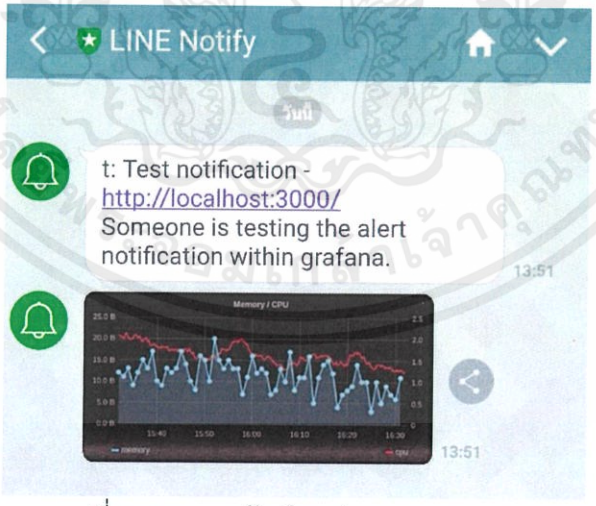
Grafana เป็นระบบที่ใช้สำหรับดูข้อมูลใน Database โดยแสดงผลในรูปแบบต่าง ๆ เช่น กราฟ หรือตาราง เป็นต้น และยังสามารถตั้งค่าให้สามารถแจ้งเตือนได้อีกด้วย ดังรูปที่ 3.64



รูปที่ 3.64 Grafana (Monitoring Server)

3.2.11 Line Notify

Line Notify เป็นระบบที่ใช้สำหรับแจ้งเตือนเหตุการณ์ต่าง ๆ ตามที่กำหนดผ่านทาง Line ของผู้ใช้งาน ดังรูปที่ 3.65



รูปที่ 3.65 การแจ้งเตือนผ่าน Line Notify

3.3 การจัดเก็บผลการทดลอง

3.3.1 ทดลองโปรแกรมวัดปริมาณการเคลื่อนไหวของโคนมครั้งที่ 1

จะเป็นการทดสอบการทำงานของโปรแกรมวัดปริมาณการเคลื่อนไหวบน Arduino MKR WAN 1300 ด้วยการทดลองกับคนก่อน

3.3.2 ทดลองการเชื่อมต่อ LoRaWAN และเก็บปริมาณการเคลื่อนไหวของโคนมบน InfluxDB

จะเป็นการทดสอบการเชื่อมต่อ LoRaWAN ระหว่างตัววัดปริมาณการเคลื่อนไหว (Arduino MKR WAN 1300) หรือ Node กับ Gateway (Wemos TTGO LORA32 868 / 915 Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora) โดยมี The Thing Network เป็นตัวกลางรับ-ส่งข้อมูล (Network Server) และนำข้อมูลปริมาณการเคลื่อนไหวของโคนมไปเก็บไว้บน InfluxDB (Database Server)

3.3.3 ทดลองโปรแกรมวัดปริมาณการเคลื่อนไหวของโคนมครั้งที่ 2

จะเป็นการทดสอบการทำงานของโปรแกรมวัดปริมาณการเคลื่อนไหวบนเครื่องวัดปริมาณการเคลื่อนไหวของโคนม ด้วยการทดลองกับโคนมที่โรงพยาบาลปศุสัตว์

3.3.4 เก็บปริมาณการเคลื่อนไหวของโคนมในแต่ละวัน

จะเป็นการเก็บปริมาณการเคลื่อนไหวในแต่ละวัน และส่งผ่านระบบ LoRaWAN เพื่อขึ้นไปเก็บบน Database สำหรับในการตรวจสอบว่าโคนมเป็นสัตหรือไม่ต่อไป

บทที่ 4

ผลการทดลอง

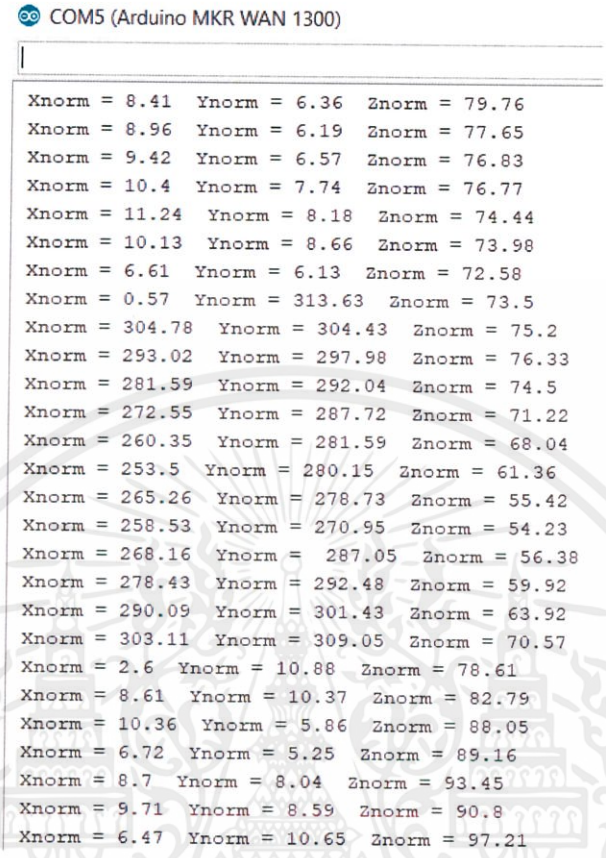
4.1 ผลการทดลองเครื่องวัดปริมาณการเคลื่อนไหวของโคนมด้วยการทดสอบกับคน

เป็นการทดสอบโปรแกรมวัดปริมาณการเคลื่อนไหวของโคนมด้วยการทดสอบกับคนก่อน เพื่อให้ทราบว่าโปรแกรมสามารถทำงานได้อย่างถูกต้อง และเป็นไปตามที่ต้องการ ซึ่งมีค่าพารามิเตอร์ 2 ตัวที่ต้องปรับเปลี่ยนให้เหมาะสมกับก้าวเดินของคนคือค่า Threshold (ค่าสำหรับการตรวจจับการเคลื่อนไหว) และค่า Duration (ค่าสำหรับการตรวจจับระยะเวลาในการเคลื่อนไหว ซึ่งขึ้นอยู่กับค่า Threshold ด้วย) และจากผลการทดสอบ ถ้าตั้งค่า Threshold เท่ากับ 4 และค่า Duration เท่ากับ 40 ms ก็จะสามารถนับก้าวเดินของคนได้อย่างมีประสิทธิภาพ ดังตาราง ที่ 4.1 เมื่อทดลองเดิน, เดินเร็ว และวิ่งเป็นจำนวนอย่างละ 100 ก้าว

ตารางที่ 4.1 ทดสอบจำนวนก้าวเดินจริงของคนเทียบกับจำนวนก้าวเดินที่ได้จากเครื่องวัดปริมาณการเคลื่อนไหวของโคนมที่ปรับค่าให้เหมาะสมกับคน

พฤติกรรม	จำนวนก้าวที่นับได้จริง (ก้าว)	จำนวนก้าวที่นับได้จากเครื่อง (ก้าว)	ค่าความคลาดเคลื่อนสัมพัทธ์
เดิน	100	100	0%
เดินเร็ว	100	100	0%
วิ่ง	100	103	3%

จากรูปที่ 4.1 เมื่อนำค่า Xnorm, Ynorm และ Znorm จาก Serial monitor มาคำนวณด้วยสมการที่ 2.1 ก็จะได้ดังตารางที่ 4.2 โดยค่าลำดับที่ 8 ถึง 21 คือขณะกำลังก้าวเดินหนึ่งก้าว ซึ่งก็คือช่วงที่มีค่ามากกว่าหรือเท่ากับ 4 และถ้ามีช่วงที่ค่ามากกว่าหรือเท่ากับ 4 นานถึง 50 ms ก็จะนับเป็นหนึ่งก้าว



รูปที่ 4.1 ค่า Xnorm, Ynorm, Znorm จาก Serial monitor

ตารางที่ 4.2 ผลการคำนวณค่า Xnorm, Ynorm และ Znorm จาก Serial monitor ด้วยสมการเพื่อนับการเคลื่อนไหว

ลำดับที่	Xnorm	Ynorm	Znorm	ค่าที่ได้จากสมการ
1	8.41	6.36	79.76	-
2	8.96	6.19	77.65	1.06
3	9.42	6.57	76.83	0.49
4	10.4	7.74	76.77	0.74
5	11.24	8.18	74.44	1.22
6	10.13	8.66	73.98	0.63
7	6.61	6.13	72.58	2.22

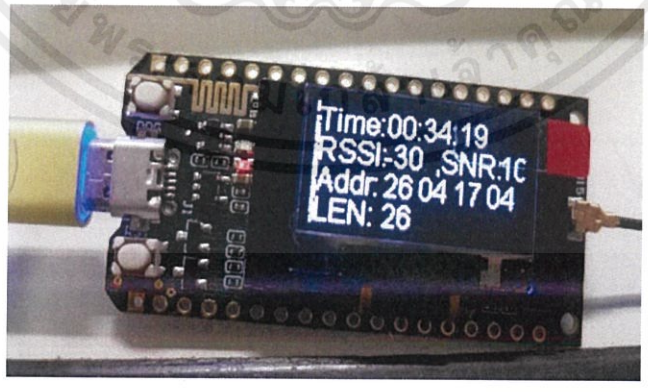
ตารางที่ 4.2 ผลการคำนวณค่า Xnorm, Ynorm และ Znorm จาก Serial monitor ด้วยสมการเพื่อนับการเคลื่อนไหว (ต่อ)

ลำดับที่	Xnorm	Ynorm	Znorm	ค่าที่ได้จากสมการ
8	0.57	313.63	73.5	150.17
9	304.78	304.43	75.2	148.61
10	293.02	297.98	76.33	6.57
11	281.59	292.04	74.5	6.35
12	272.55	287.72	71.22	5.14
13	260.35	281.59	68.04	6.84
14	253.5	280.15	61.36	4.72
15	265.26	278.73	55.42	6.47
16	258.53	270.95	54.23	5.05
17	268.16	287.05	56.38	9.22
18	278.43	292.48	59.92	5.92
19	290.09	301.43	63.92	7.43
20	303.11	309.05	70.57	8.05
21	2.6	10.88	78.61	206.74
22	8.61	10.37	82.79	3.58
23	10.36	5.86	88.05	3.48
24	6.72	5.25	89.16	1.88
25	8.7	8.04	93.45	2.67
26	9.71	8.59	90.8	1.41
27	6.47	10.65	97.21	3.64

4.2 ผลการทดลองการเชื่อมต่อ LoRaWAN และนำข้อมูลเข้าไปเก็บบน Database Server (InfluxDB)

จะเป็นการทดสอบการเชื่อมต่อ LoRaWAN ระหว่างเครื่องวัดปริมาณการเคลื่อนไหวของโคนม (Arduino MKR WAN 1300) หรือ Node กับ Gateway (Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 Oled-display Bluetooth WIFI Lora) โดยมี The Thing Network เป็นตัวกลางในการรับ-ส่งข้อมูล (Network Server) และนำข้อมูลปริมาณการเคลื่อนไหวของโคนมเก็บไว้บน InfluxDB (Database Server)

เมื่อตัววัดปริมาณการเคลื่อนไหวของโคนมส่งข้อมูลปริมาณการเคลื่อนไหวไปยัง Gateway ตัวหน้าจอของ Gateway ก็จะแสดงเวลาที่ได้รับข้อมูล, ค่า RSSI, ค่า SNR, Address ของผู้ส่ง และความยาวของข้อมูล ดังรูปที่ 4.2 และเมื่อเปิดหน้าเว็บ ESP Gateway Config ก็จะสังเกตเห็นว่ามีข้อมูลส่งผ่าน Gateway มา โดยดูจากหัวข้อ Message History จะได้ดังรูปที่ 4.3 และเมื่อเปิดหน้า APPLICATION DATA บนเว็บไซต์ The Thing Network โดยกดที่ Data ในหน้า Application Overview จะเห็นได้ว่ามีข้อมูลที่ The Thing Network ได้รับจาก Gateway และเตรียมทำการส่งต่อไปยัง Node Red และ Node Red จะส่งข้อมูลไป Database Server ให้ ซึ่งแสดงดังรูปที่ 4.4 โดยถ้ามีการเขียน Code การแปลงข้อมูลใน Payload Format ไว้ก่อนแล้ว ข้อมูลที่ถูกแปลงจะแสดงอยู่ข้าง ๆ ข้อมูลแบบเลขฐานสิบหก และเมื่อเปิด Dashboard ของ Grafana จะแสดงค่า ปริมาณการเคลื่อนไหวของโคนมที่ถูกเก็บไว้ใน Database Server (InfluxDB) เรียบร้อยดังรูปที่ 4.5



รูปที่ 4.2 หน้าจอแสดงข้อมูลต่าง ๆ ของ Wemos TTGO LORA32 868 / 915Mhz SX1276 ESP32 เมื่อ Node ส่งข้อมูลมาถึง Gateway

Package Statistics

Counter	C 0	C 1	C 2	Pkgs	Pkgs/hr
Packages Downlink				1	
Packages Uplink Total				2	20
Packages Uplink OK				2	
SF7 recvd	0	0	0	0	0 %
SF8 recvd	0	0	0	0	0 %
SF9 recvd	0	0	0	0	0 %
SF10 recvd	2	0	0	2	100 %
SF11 recvd	0	0	0	0	0 %
SF12 recvd	0	0	0	0	0 %

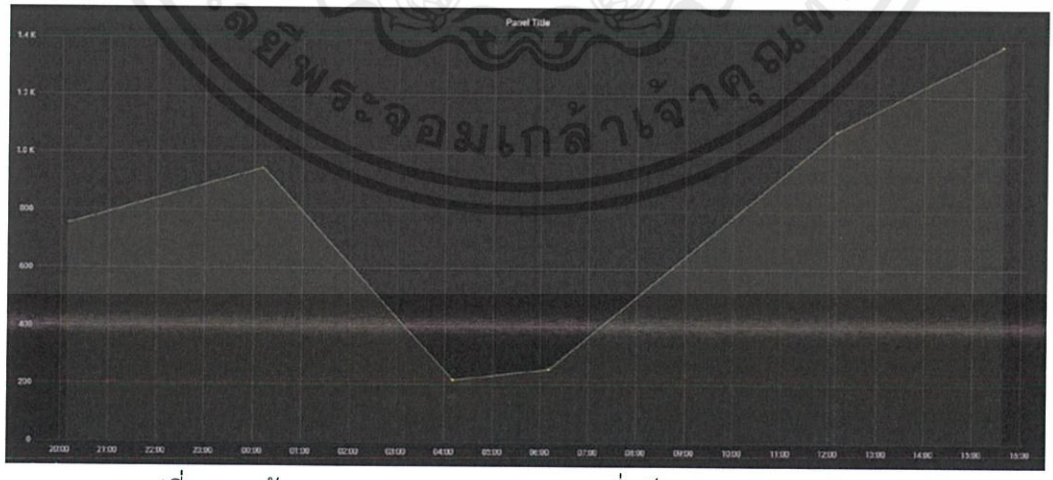
Message History

Time	Node	C	Freq	SF	pRSSI
Monday 19-11-2018 18:47:25	26 04 2b 6c	0	923200000	10	-106
Monday 19-11-2018 18:42:57	d0 01 40 f9	0	923200000	10	-44

รูปที่ 4.3 หน้าเว็บ ESP Gateway Config เมื่อ Node ส่งข้อมูลมาถึง Gateway

time	counter	port		
18:06:17	6	2	confirmed	dev id: mkr_001 payload: 31 34 39 receivedString: "149"
18:06:02		0		dev id: mkr_001
18:06:02	5	2	confirmed	dev id: mkr_001 payload: 31 34 35 receivedString: "145"
18:05:44		0		dev id: mkr_001
18:05:44	4	2	confirmed	dev id: mkr_001 payload: 31 33 34 receivedString: "134"
18:05:33		0		dev id: mkr_001
18:05:32	3	2	confirmed	dev id: mkr_001 payload: 31 32 34 receivedString: "124"
18:04:39		0		dev id: mkr_001
18:04:39	1	2	confirmed	dev id: mkr_001 payload: 34 38 receivedString: "48"
18:03:31		0		dev id: mkr_001
18:03:31	0	2	confirmed	dev id: mkr_001 payload: 34 34 receivedString: "44"
18:02:57				dev id: mkr_001 dev addr: 26 04 21 ED app eui: 70 B3 D5 7E D0 01 40 F9 dev eui: A8 61 0A 34 32 20 7I

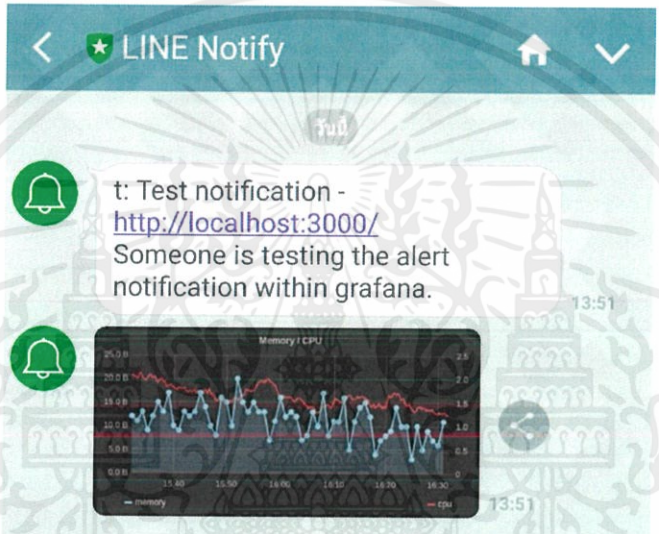
รูปที่ 4.4 หน้า APPLICATION DATA บนเว็บไซต์ The Thing Network เมื่อ Gateway ส่งต่อข้อมูลไปยัง The Thing Network



รูปที่ 4.5 หน้า Dashboard ของ Grafana ซึ่งเป็น Monitoring Server

4.3 ผลการทดลองการทดสอบระบบ Line Notify

เมื่อตั้งค่าระบบ Line Notify บน Grafana (Mornitoring Server) ให้สามารถแจ้งเตือนผ่านทาง Line ของผู้ใช้งานได้ ก็ได้ทำการกดปุ่ม Test Send บน Grafana (Mornitoring Server) เพื่อทำการทดสอบว่า Grafana (Mornitoring Server) จะส่งข้อความแจ้งเตือนมายัง Line ของผู้ใช้งานหรือไม่ ซึ่งผลก็คือ Grafana (Mornitoring Server) สามารถส่งข้อความแจ้งเตือนมาบน Line ของผู้ใช้งานได้ ดังรูปที่ 4.6

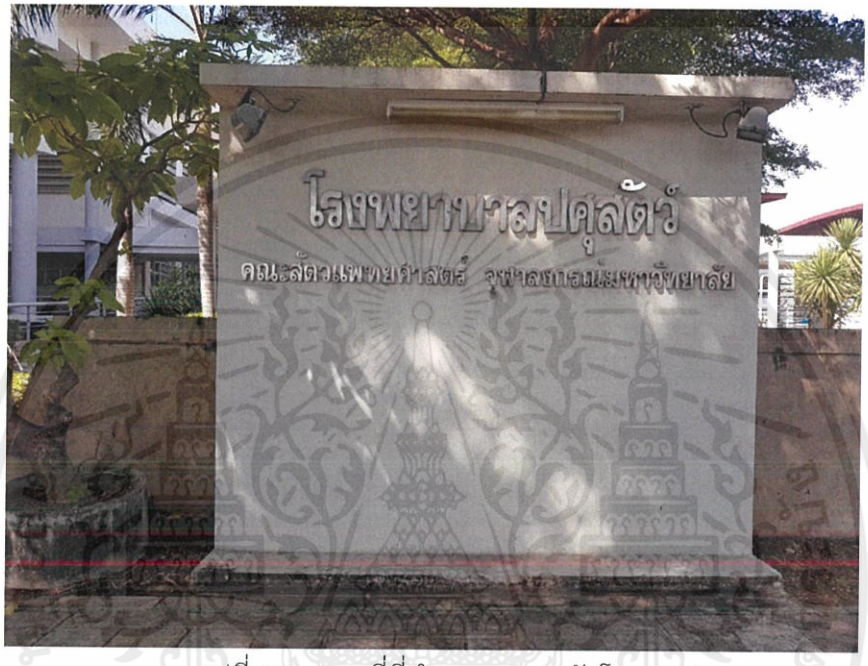


รูปที่ 4.6 การแจ้งเตือนของระบบผ่าน Line ผู้ใช้งาน

4.4 ผลการทดลองระบบวัดปริมาณการเคลื่อนไหวของโคนม

เป็นการทดสอบระบบวัดปริมาณการเคลื่อนไหวของโคนม ซึ่งทดสอบที่โรงพยาบาลปศุสัตว์ ดังรูปที่ 4.7 และดังรูปที่ 4.8 โดยการทดสอบนั้นเป็นไปได้ยาก เพราะโคนมมีการแสดงอาการขัดขืนต่าง ๆ ซึ่งทำให้การติดตั้งเครื่องวัดปริมาณการเคลื่อนไหวของโคนมที่บริเวณคอของโคนมทำได้ยาก และไม่สามารถเสียบสาย USB กับบอร์ด Arduino ขณะอุปกรณ์ติดอยู่ที่คอได้ จึงทำให้การใช้โปรแกรมเพื่อหาค่า Threshold และ Duration ที่เหมาะสมเป็นไปได้ยากลำบาก ดังนั้นจึงได้เลือกค่า Threshold และ Duration มาค่าหนึ่ง โดยเริ่มจากค่าที่ใช้นับก้าวเดินของคนก่อน แล้วปล่อยให้โคนมเริ่มเคลื่อนไหวต่าง ๆ ซึ่งค่า Activities ก็คือค่าการเคลื่อนไหวทุกรูปแบบของโคนม ไม่ว่าจะเดิน วิ่ง ส่ายคอ การโดนโคนมอีกตัวเดินชน การชนโคนมตัวอื่น ๆ และการขึ้นทับ

ตัวอื่น ๆ ซึ่งจะสังเกตพฤติกรรมในลักษณะนี้แล้วนับเป็น 1 Activity ต่อการกระทำใด ๆ 1 ครั้ง แล้วทำการส่งค่า Activities ของโคนมที่วัดได้ขึ้นไปบน Database Server และเทียบดูกับผลการทดสอบกับค่า Threshold และ Duration อื่น ๆ จึงได้ข้อสรุปว่า Threshold เท่ากับ 4 และ Duration เท่ากับ 120 เหมาะสมที่สุด

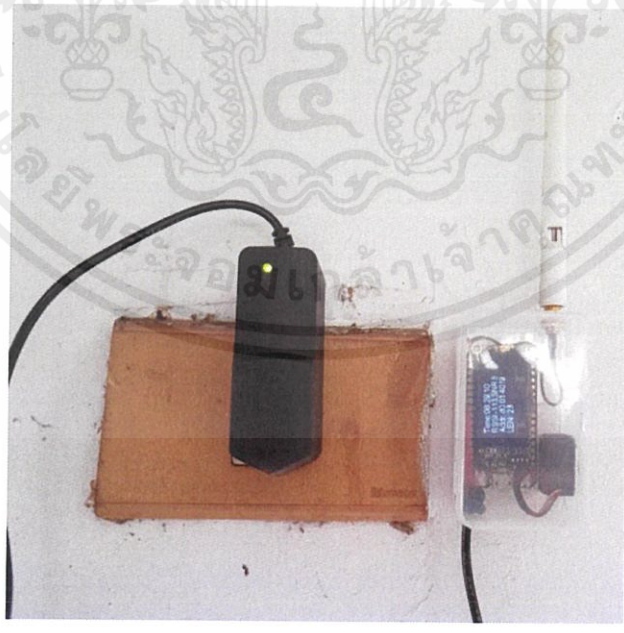


รูปที่ 4.7 สถานที่ที่ทำการทดสอบกับโคนม

รูปที่ 4.8 เตรียมนำอุปกรณ์ไปติดตั้งกับโคนม

4.5 ผลการเก็บค่า Activities ของโคนม เพื่อหาโอกาสในการเป็นสัดของโคนม

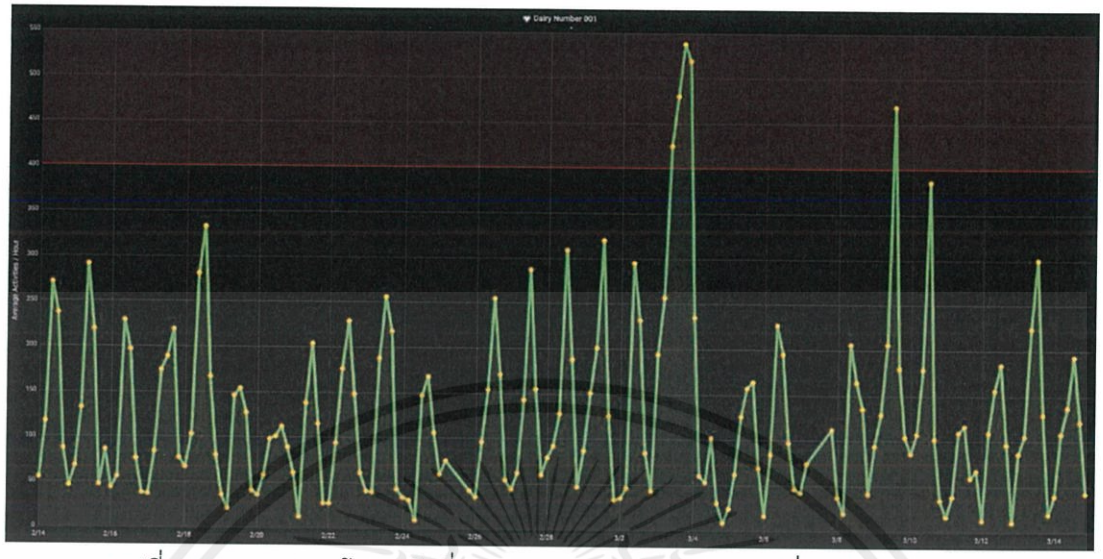
ทำการติดตั้งระบบวัดปริมาณการเคลื่อนไหวของโคนมที่โรงพยาบาลปศุสัตว์ ซึ่งรวมถึงการติดตั้ง LoRaWAN Gateway ดังรูปที่ 4.9 และติดตั้งเครื่องวัดปริมาณการเคลื่อนไหวของโคนมไว้ที่คอของโคนม จำนวน 2 ตัว ดังรูปที่ 4.10 รวมถึงมีการติดตั้งระบบวัดปริมาณการเคลื่อนไหวของโคนมที่เป็นระบบที่ใช้งานจริงในเชิงพาณิชย์กับโคนมที่ใช้ในการทดสอบอีกด้วย เพื่อเปรียบเทียบข้อมูลกัน โดยระบบวัดปริมาณการเคลื่อนไหวของโคนมจะเก็บค่า Activities สะสมของโคนมทุก 4 ชั่วโมงในหน่วย Average Activities / Hour และส่งข้อมูลผ่าน LoRaWAN ขึ้นไปเก็บบน Database Server (InFluxDB) โดยจากการสังเกตด้วยการติดตามดูพฤติกรรมของโคนมและดูค่า Activities บน Grafana (Mornitoring Server) ซึ่งโคนมทั้ง 2 ตัวส่วนใหญ่โดยปกติจะมีการเคลื่อนไหว หรือ Activites เพิ่มขึ้นช่วงเวลา 00.00 น. ถึง 8.00 น. และสูงที่สุดสุด ช่วงเวลา 8.00 น. ถึง 12.00 น. และจะมีการเคลื่อนไหว หรือ Activites ลดลงที่ช่วงเวลา 12.00 น. ถึง 20.00 น. และการเคลื่อนไหว หรือ Activites น้อยที่สุดช่วง 20.00 น. ถึง 00.00 น. โดยเมื่อนำข้อมูลที่ได้นั้นมาแสดงในรูปของกราฟที่ลากเส้นจากจุดถึงจุด ซึ่งจุดก็คือข้อมูลที่มาจากใน Database Server (InfluxDB) จะมีแกน X เป็นวัน, เดือน, ปี และเวลา ส่วนแกน Y เป็นค่า Activities ของวัวในหน่วย Average Activities / Hour



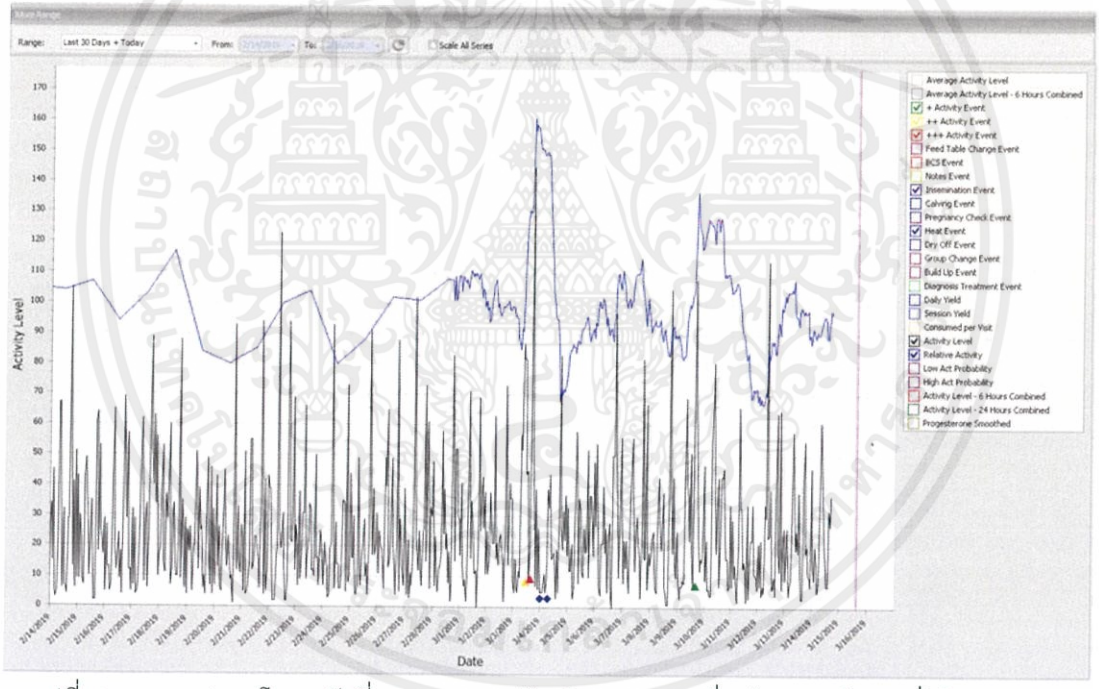
รูปที่ 4.9 LoRaWAN Gateway ที่ติดตั้งไว้บนผนังของโรงพยาบาลปศุสัตว์

รูปที่ 4.10 โคนมที่ติดเครื่องวัดปริมาณการเคลื่อนไหวของโคนม

โดยจะเห็นว่ากราฟของโคนมตัวที่ 1 จากระบบวัดปริมาณการเคลื่อนไหวของโคนม กับกราฟของโคนมตัวที่ 1 จากระบบวัดปริมาณการเคลื่อนไหวของโคนมที่ใช้งานจริงใน เชิงพาณิชย์ ดังรูปที่ 4.11 และ ดังรูปที่ 4.12 ตามลำดับ มีรูปแบบ และลักษณะกราฟต่างกัน เพราะใช้การคำนวณ และอัลกอลิทึมต่างกัน แต่สิ่งที่เหมือนกันคือสามารถตรวจจับการเป็นสัดของโคนมได้เหมือนกัน และมีความผิดพลาดในการตรวจจับการเป็นสัดของโคนมเหมือนกันด้วย โดยจากรูปที่ 4.12 กราฟจากระบบวัดปริมาณการเคลื่อนไหวของโคนมที่ใช้งานจริงในเชิงพาณิชย์ จะแสดงจุดสามเหลี่ยมสีแดง และเหลืองใต้กราฟคือระบบแจ้งเตือนว่าโคนมกำลังเป็นสัด ส่วนจุดสี่เหลี่ยมสีน้ำเงิน 2 จุด ใต้กราฟ คือสัตวแพทย์ได้ทำการมาร์คเอาไว้ว่า โคนมเป็นสัดวันที่ 03/03/2019 จริง ส่วนจุดสามเหลี่ยมสีเขียว คือระบบแจ้งเตือนว่าโคนมกำลังเป็นสัด แต่ไม่มีจุดสี่เหลี่ยมสีน้ำเงิน 2 จุด ใต้กราฟ ซึ่งก็คือวันที่ 09/03/2019 โคนมไม่ได้เป็นสัดจริง ๆ ดังนั้นเป็นความผิดพลาดของระบบ และจากรูปที่ 4.11 กราฟจากระบบวัดปริมาณการเคลื่อนไหวของโคนม เมื่อค่า Activities ของโคนมถึงขีดเส้นสีแดง หรือมีค่ามากกว่าหรือเท่ากับ 400 Average Activities / Hour นั้นจะแสดงว่าโคนมกำลังมีอาการเป็นสัด ซึ่งตรงกับช่วงวันที่ 03/03/2019 ส่วนวันที่ 09/03/2019 นั้นเป็นความผิดพลาดของระบบเหมือนกันกับระบบวัดปริมาณการเคลื่อนไหวของโคนมที่ใช้งานจริงในเชิงพาณิชย์



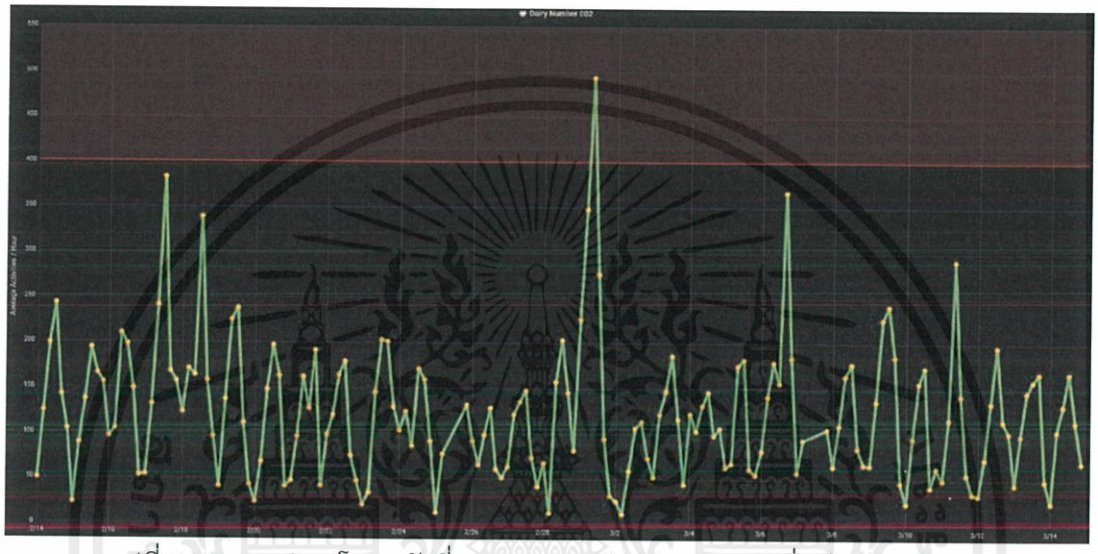
รูปที่ 4.11 กราฟของโคนมตัวที่ 1 จากระบบวัดปริมาณการเคลื่อนไหวของโคนม



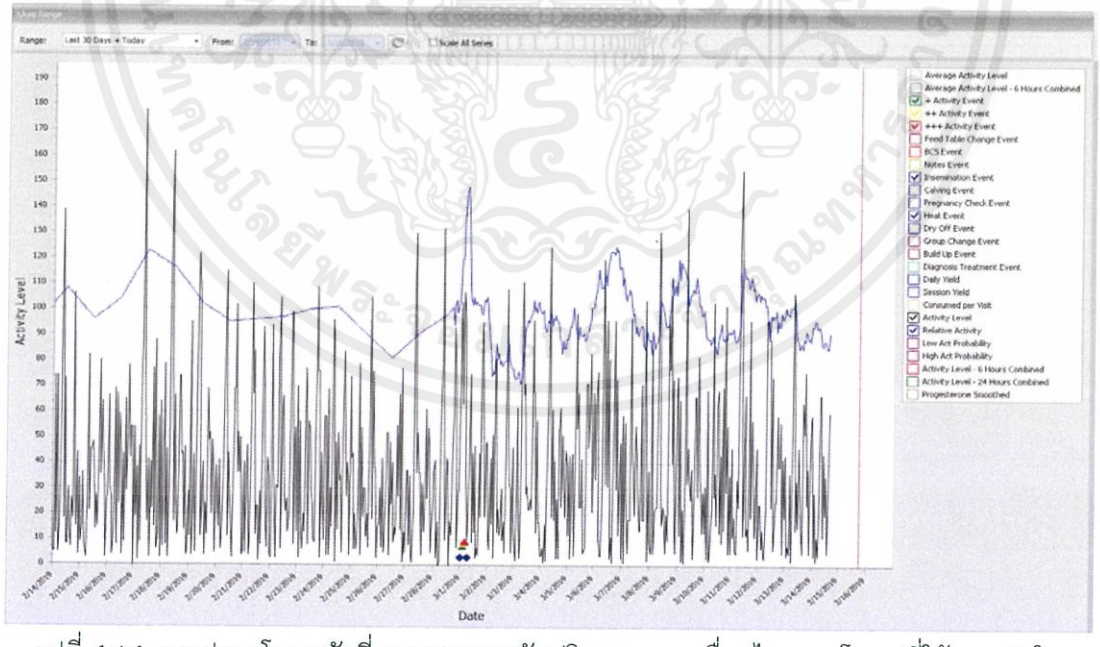
รูปที่ 4.12 กราฟของโคนมตัวที่ 1 จากระบบวัดปริมาณการเคลื่อนไหวของโคนมที่ใช้งานจริงในเชิงพาณิชย์

ส่วนกราฟของโคนมตัวที่ 2 จากระบบวัดปริมาณการเคลื่อนไหวของโคนมกับกราฟของโคนมตัวที่ 2 จากระบบวัดปริมาณการเคลื่อนไหวของโคนมที่ใช้งานจริงในเชิงพาณิชย์ ดังรูปที่ 4.13 และ ดังรูปที่ 4.14 ตามลำดับ สามารถตรวจจับการเป็นสัดของโคนมได้เหมือนกัน โดยไม่มีความผิดพลาด โดยจากรูปที่ 4.13 ในกราฟจากระบบวัดปริมาณการเคลื่อนไหวของโคนม ที่ใช้งาน

จริงในเชิงพาณิชย์แสดงจุดสามเหลี่ยมสีแดง เขียวและเหลืองใต้กราฟคือระบบแจ้งเตือนว่าโคนมกำลังเป็นสัด ส่วนจุดสี่เหลี่ยมสีน้ำเงิน 2 จุด ใต้กราฟ คือ สัตวแพทย์ได้ทำการมาร์คเอาไว้ว่า โคนมเป็นสัดวันที่ 01/03/2019 จริง และจากรูปที่ 4.14 เมื่อค่า Activities ของโคนมถึงขีดเส้นสีแดงหรือมีค่ามากกว่าหรือเท่ากับ 400 Average Activities / Hour นั้นจะแสดงว่าโคนมกำลังมีอาการเป็นสัด ซึ่งตรงกับช่วงวันที่ 01/03/2019

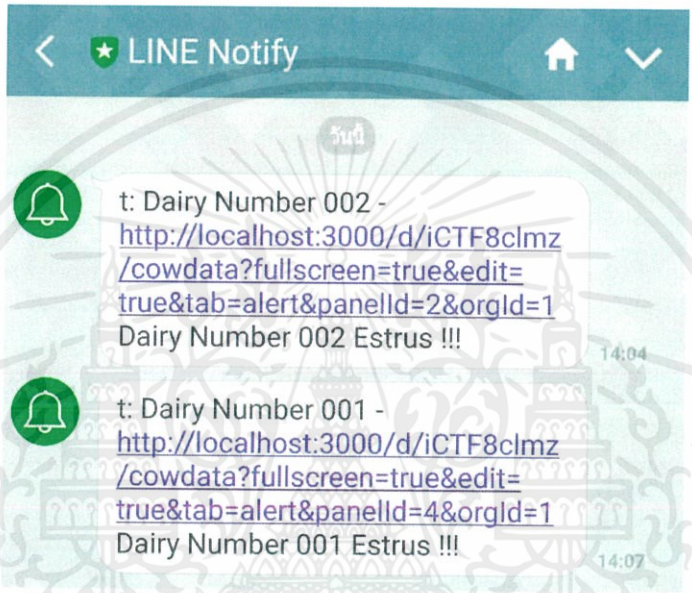


รูปที่ 4.13 กราฟของโคนมตัวที่ 2 จากระบบวัดปริมาณการเคลื่อนไหวของโคนม



รูปที่ 4.14 กราฟของโคนมตัวที่ 2 จากระบบวัดปริมาณการเคลื่อนไหวของโคนมที่ใช้งานจริงในเชิงพาณิชย์

ดังนั้นแล้วค่า Activities ของโคนมที่ถึงขีดเส้นสีแดง หรือมีค่ามากกว่าหรือเท่ากับ 400 Average Activities / Hour ที่จะแสดงว่าโคนมกำลังมีอาการเป็นสัดนั้น จะเรียกว่าค่า Estrus Threshold ซึ่งค่านี้จะแม่นยำขึ้น หรือแม่นยำน้อยลงจะขึ้นกับสถิติข้อมูลในการเป็นสัดของโคนม ยิ่งมีข้อมูลมากค่า Estrus Threshold ก็จะถูกเปลี่ยนแปลงไปด้วย และสามารถนำมาประยุกต์ใช้กับระบบการแจ้งเตือนเมื่อโคนมกำลังเป็นสัดผ่านทาง Line ของผู้ใช้งานดังรูปที่ 4.15



รูปที่ 4.15 การแจ้งเตือนผ่านทาง Line

ระบบในการตรวจจับการเป็นสัดของโคนมที่ใช้เพียงการวิเคราะห์แค่ปริมาณการเคลื่อนไหวของโคนม อาจยังไม่เพียงพอที่จะทำให้ระบบมีความแม่นยำมาก โดยถ้าจะให้มีความแม่นยำยิ่งขึ้น ควรใช้ร่วมกับการวิเคราะห์อื่น ๆ นอกเหนือจากการวิเคราะห์ปริมาณการเคลื่อนไหวของโคนม หรือค่า Activities เช่น วิเคราะห์เรื่องการเคี้ยวเอื้อง วิเคราะห์เรื่องปริมาณน้ำนม และวิเคราะห์เรื่องปริมาณของฮอร์โมน เป็นต้น

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์นี้มีจุดประสงค์เพื่อลดความไม่แม่นยำในการระบุโคนมที่อาจจะเป็นสัด สำหรับปศุสัตว์ผู้เลี้ยงโคนมเป็นจำนวนมาก จึงออกแบบระบบวัดปริมาณการเคลื่อนไหวของโคนม เพื่อช่วยในการวิเคราะห์การเป็นสัดในโคนม โดยการนำอุปกรณ์วัดปริมาณการเคลื่อนไหวของโคนม ทำเป็นปลอกคอสวมใส่ที่คอของ เพื่อตรวจจับปริมาณการเคลื่อนไหวในแต่ละวันของโคนม และส่งข้อมูลปริมาณการเคลื่อนไหวของโคนมนั้นผ่านเครือข่าย LoRaWAN และอินเทอร์เน็ต ไปเก็บบันทึกไว้ที่ส่วนบันทึกข้อมูลและนำข้อมูลไปใช้ในการวิเคราะห์หาความน่าจะเป็นในการเป็นสัดของโคนมต่อไป

5.2 ข้อเสนอแนะ

ระบบในการตรวจจับการเป็นสัดของโคนมที่ใช้เพียงการวิเคราะห์แค่ปริมาณการเคลื่อนไหวของโคนม อาจยังไม่เพียงพอที่จะทำให้ระบบมีความแม่นยำมาก โดยถ้าจะให้มีความแม่นยำยิ่งขึ้น ควรใช้ร่วมกับการวิเคราะห์อื่น ๆ นอกเหนือจากการวิเคราะห์ปริมาณการเคลื่อนไหวของโคนม หรือค่า Activities เช่น วิเคราะห์เรื่องการเคี้ยวเอื้อง วิเคราะห์เรื่องปริมาณน้ำนม และวิเคราะห์เรื่องปริมาณของฮอร์โมน เป็นต้น

เนื่องจากระบบนี้จะถูกส่งต่อการใช้งานไปยังกลุ่มผู้ที่ประกอบอาชีพปศุสัตว์ซึ่งอาจจะไม่มีความชำนาญเกี่ยวกับการใช้เทคโนโลยี ดังนั้นหน้าต่างการใช้งานของระบบควรมีการปรับให้มีความง่ายต่อการใช้งานมากยิ่งขึ้น

บรรณานุกรม

[1] Belinda Ary. "Breeds of Beef Cattle." https://cattle-today.com/.

[2] Pearl Wilson. "Pedigree Dairy Breeds." http://www.thecattlesite.com/breeds/dairy/.

[3] เนติยะ ยอดเณร. "วงรอบการเป็นสัด การกระตุ้นการเป็นสัดในโค และการเตรียมตัวรับ." http://www.thailivestock.com/knowledge/.

[4] DeLaval. "Cow comfort: 6) Reproduction." http://www.milkproduction.com/Library/Scientific-articles/Housing/Cow-comfort-6/.

[5] Cristiano Cortes. "Estrous cycle." http://www.groupe-esa.com/ladmec/bricks_modules/brick03/co/ZBO_Brick03_4.html.

[6] M. Alsaaod,*1 M. Luternauer,* T. Hausegger,† R. Kredel,† and A. Steiner*. "The cow pedogram—Analysis of the gait cycle variables allows the detection of lameness and foot pathologies." Journal of Dairy Science Vol. 100. 2 (February 2017) : 1-10.

[7] พรชัย ลีลาพรชัย. "LoRa vs LTE vs Sigfox ทางเลือกของ IoT." http://www.semi-journal.com/lora-vs-lte-vs-sigfox-ทางเลือกของ-iot/

[8] Michael Schoeffler. "Tutorial: How to use the GY-521 module (MPU-6050 breakout board) with the Arduino Uno." https://www.mschoeffler.de/2017/10/05/.

[9] platenspeler. "ESP-1ch-Gateway-v5.0." https://github.com/things4u/ESP-1ch-Gateway-v5.0.

[10] gonzalocasas. "Getting started with the Arduino MKR WAN 1300." https://github.com/gonzalocasas/arduino-mkr-wan-1300.

[11] Ashvin Suthar. "Node RED—Getting started (read live tweets)." https://blog.usejournal.com/node-red-getting-started-read-live-tweets-c6fde71a7857.

[12] Grafana Labs. "Alert Notifications." http://docs.grafana.org/alerting/notifications/.

[13] Suwat Nakchukaew. "สร้างการแจ้งเตือนด้วย Line Notify." https://engineering.thinknet.co.th/สร้างการแจ้งเตือนด้วย-line-notify-670f9b20ac27.

[14] S. Reith, H. Brandt, S. Hoy. "Simultaneous analysis of activity and rumination time, based on collar-mounted sensor technology, of dairy cows over the peri-estrus period." Livestock Science. 170 (October 2014) : 219–227

[15] Qingchi Zeng, Biao Zhou, Changqiang Jing, Nammoon Kim and Youngok Kim. "A Novel Step Counting Algorithm Based on Acceleration and Gravity Sensors of a Smart-Phone." International Journal of Smart Home. 4 (2015) : 221-224.

[16] Ahmad Abadleh, Eshraq Al-Hawari, Esra'a Alkafaween and Hamad Al-Sawalqah. "Step Detection Algorithm For Accurate Distance Estimation Using Dynamic Step Length." IT Department, Mutah University.

ภาคผนวก

Source Code ของโปรแกรมวัดปริมาณการเคลื่อนไหวของโคนม

และ arduino_secrets.h

Source Code ของโปรแกรมวัดปริมาณการเคลื่อนไหวของโคนม

```
#include <Wire.h>
#include <RTCZero.h>
#include <MPU6050.h>
#include <MKRWAN.h>
#include "arduino_secrets.h"

MPU6050 mpu;
int steps = 0;
int h1 = 0;
int h2 = 0;
int h3 = 0;
int h4 = 0;
int h5 = 0;
int h6 = 0;
int activity = 0;

// Select your region (AS923, AU915, EU868, KR920, IN865, US915, US915_HYBRID)
_lora_band region = AS923;

LoRaModem modem(Serial1);

String appEui = SECRET_APP_EUI;
String appKey = SECRET_APP_KEY;
/* Create an rtc object */
RTCZero rtc;

/* Change these values to set the current initial time */
const byte seconds = 00;
const byte minutes = 35;
const byte hours = 14;
```

```
/* Change these values to set the current initial date */
const byte day = 13;
const byte month = 03;
const byte year = 19;

void setup()
{
  Serial.begin(115200);
  delay(20000);
  rtc.begin(); // initialize RTC

  // Set the time
  rtc.setHours(hours);
  rtc.setMinutes(minutes);
  rtc.setSeconds(seconds);

  // Set the date
  rtc.setDay(day);
  rtc.setMonth(month);
  rtc.setYear(year);

  Serial.println("Initialization MPU6050");

  while (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_16G))
  {
    Serial.println("Can not find MPU6050 - check connection!");
    delay(500);
  }

  // Additional delay for powering the accelerometer 3ms
```

```
  mpu.setAccelPowerOnDelay(MPU6050_DELAY_3MS);

  // Turn off hardware interrupts for selected events
  mpu.setIntFreeFallEnabled(false);
  mpu.setIntZeroMotionEnabled(false);
  mpu.setIntMotionEnabled(false);

  // Set the bandwidth filter
  mpu.setDHPFMode(MPU6050_DHPF_5HZ);

  // Set the limits of motion detection to 4mg
  // and a minimum duration of 120ms
  mpu.setMotionDetectionThreshold(4);
  mpu.setMotionDetectionDuration(120);
}

void loop()
{
  Vector rawAccel = mpu.readRawAccel();
  Activites act = mpu.readActivites();

  // If motion is detected
  if (act.isActivity)
  {
    steps = steps + 1;

    // Print date...
    print2digits(rtc.getDay());
    Serial.print("/");
    print2digits(rtc.getMonth());
    Serial.print("/");
```

```
    print2digits(rtc.getYear());
    Serial.print(" ");

    // ...and time
    print2digits(rtc.getHours());
    Serial.print(":");
    print2digits(rtc.getMinutes());
    Serial.print(":");
    print2digits(rtc.getSeconds());
    Serial.print(" ");
    Serial.print("steps=");
    Serial.println(steps);
  }
  delay(50);

  //time to activation LoRaWAN at 23.55
  if (rtc.getHours() == 23 && rtc.getSeconds() == 00 && rtc.getMinutes() == 55)
  {
   // change this to your regional band (eg. US915, AS923, ...)
   if (!modem.begin(AS923)) {
     Serial.println("Failed to start module");
   };
   Serial.print("Your module version is: ");
   Serial.println(modem.version());
   Serial.print("Your device EUI is: ");
   Serial.println(modem.deviceEUI());

   int connected = modem.joinOTAA(appEui, appKey);
   if (!connected) {
     Serial.println("Something went wrong; are you indoor? Move near a window and
retry");
```

```
    int connected = modem.joinOTAA(appEui, appKey);
  }

  // Set poll interval to 60 secs.
  modem.minPollInterval(60);
  // NOTE: independently by this setting the modem will
  // not allow to send more than one message every 2 minutes,
  // this is enforced by firmware and can not be changed.
}
delay(50);

//set time to sent Average Activities / Hour at 00.00
if (rtc.getHours() == 00 && rtc.getSeconds() == 00 && rtc.getMinutes() == 00)
{
  sendsteps1();
}
delay(50);

//reset Activities at 00.01
if (rtc.getHours() == 00 && rtc.getSeconds() == 00 && rtc.getMinutes() == 01)
{
  steps = 0;
}
delay(50);

//time to activation LoRaWAN at 3.55
if (rtc.getHours() == 03 && rtc.getSeconds() == 00 && rtc.getMinutes() == 55)
{
  // change this to your regional band (eg. US915, AS923, ...)
  if (!modem.begin(AS923)) {
    Serial.println("Failed to start module");
```

```
  };
  Serial.print("Your module version is: ");
  Serial.println(modem.version());
  Serial.print("Your device EUI is: ");
  Serial.println(modem.deviceEUI());

  int connected = modem.joinOTAA(appEui, appKey);
  if (!connected) {
    Serial.println("Something went wrong; are you indoor? Move near a window and
retry");
    int connected = modem.joinOTAA(appEui, appKey);
  }

  // Set poll interval to 60 secs.
  modem.minPollInterval(60);
  // NOTE: independently by this setting the modem will
  // not allow to send more than one message every 2 minutes,
  // this is enforced by firmware and can not be changed.
}
delay(50);

//set time to sent Average Activities / Hour at 4.00
if (rtc.getHours() == 04 && rtc.getSeconds() == 00 && rtc.getMinutes() == 00)
{
  sendsteps2();
}
delay(50);

//time to activation LoRaWAN at 7.55
if (rtc.getHours() == 07 && rtc.getSeconds() == 00 && rtc.getMinutes() == 55)
{
```

```
    // change this to your regional band (eg. US915, AS923, ...)
    if (!modem.begin(AS923)) {
      Serial.println("Failed to start module");
    };
    Serial.print("Your module version is: ");
    Serial.println(modem.version());
    Serial.print("Your device EUI is: ");
    Serial.println(modem.deviceEUI());

    int connected = modem.joinOTAA(appEui, appKey);
    if (!connected) {
      Serial.println("Something went wrong; are you indoor? Move near a window and
retry");
      int connected = modem.joinOTAA(appEui, appKey);
    }

    // Set poll interval to 60 secs.
    modem.minPollInterval(60);
    // NOTE: independently by this setting the modem will
    // not allow to send more than one message every 2 minutes,
    // this is enforced by firmware and can not be changed.
  }
  delay(50);

  //set time to sent Average Activities / Hour at 8.00
  if (rtc.getHours() == 8 && rtc.getSeconds() == 00 && rtc.getMinutes() == 00)
  {
    sendsteps3();
  }
  delay(50);
```

```
  //time to activation LoRaWAN at 11.55
  if (rtc.getHours() == 11 && rtc.getSeconds() == 00 && rtc.getMinutes() == 55)
  {
    // change this to your regional band (eg. US915, AS923, ...)
    if (!modem.begin(AS923)) {
      Serial.println("Failed to start module");
    };
    Serial.print("Your module version is: ");
    Serial.println(modem.version());
    Serial.print("Your device EUI is: ");
    Serial.println(modem.deviceEUI());

    int connected = modem.joinOTAA(appEui, appKey);
    if (!connected) {
      Serial.println("Something went wrong; are you indoor? Move near a window and
retry");
      int connected = modem.joinOTAA(appEui, appKey);
    }

    // Set poll interval to 60 secs.
    modem.minPollInterval(60);
    // NOTE: independently by this setting the modem will
    // not allow to send more than one message every 2 minutes,
    // this is enforced by firmware and can not be changed.
  }
  delay(50);

  //set time to sent Average Activities / Hour at 12.00
  if (rtc.getHours() == 12 && rtc.getSeconds() == 00 && rtc.getMinutes() == 00)
  {
    sendsteps4();
```

```
  }
  delay(50);

  //time to activation LoRaWAN at 15.55
  if (rtc.getHours() == 15 && rtc.getSeconds() == 00 && rtc.getMinutes() == 55)
  {
    // change this to your regional band (eg. US915, AS923, ...)
    if (!modem.begin(AS923)) {
      Serial.println("Failed to start module");
    };
    Serial.print("Your module version is: ");
    Serial.println(modem.version());
    Serial.print("Your device EUI is: ");
    Serial.println(modem.deviceEUI());

    int connected = modem.joinOTAA(appEui, appKey);
    if (!connected) {
      Serial.println("Something went wrong; are you indoor? Move near a window and
retry");
      int connected = modem.joinOTAA(appEui, appKey);
    }

    // Set poll interval to 60 secs.
    modem.minPollInterval(60);
    // NOTE: independently by this setting the modem will
    // not allow to send more than one message every 2 minutes,
    // this is enforced by firmware and can not be changed.
  }
  delay(50);

  //set time to sent Average Activities / Hour at 16.00
```

```
  if (rtc.getHours() == 16 && rtc.getSeconds() == 00 && rtc.getMinutes() == 00)
  {
    sendsteps5();
  }
  delay(50);

  //time to activation LoRaWAN at 19.55
  if (rtc.getHours() == 19 && rtc.getSeconds() == 00 && rtc.getMinutes() == 55)
  {
    // change this to your regional band (eg. US915, AS923, ...)
    if (!modem.begin(AS923)) {
      Serial.println("Failed to start module");
    };
    Serial.print("Your module version is: ");
    Serial.println(modem.version());
    Serial.print("Your device EUI is: ");
    Serial.println(modem.deviceEUI());

    int connected = modem.joinOTAA(appEui, appKey);
    if (!connected) {
      Serial.println("Something went wrong; are you indoor? Move near a window and
retry");
      int connected = modem.joinOTAA(appEui, appKey);
    }

    // Set poll interval to 60 secs.
    modem.minPollInterval(60);
    // NOTE: independently by this setting the modem will
    // not allow to send more than one message every 2 minutes,
    // this is enforced by firmware and can not be changed.
  }
```

```
  delay(50);

  //set time to sent Average Activities / Hour at 20.00
  if (rtc.getHours() == 20 && rtc.getSeconds() == 00 && rtc.getMinutes() == 00)
  {
    sendsteps6();
  }
  delay(50);

}

void print2digits(int number) {
  if (number < 10) {
    Serial.print("0"); // print a 0 before if the number is < than 10
  }
  Serial.print(number);
}

void sendsteps1()
{
  h1 = steps;
  activity = abs(((h1 - h6) * 10) / 4);
  int err;
  modem.beginPacket();
  modem.print(activity);
  err = modem.endPacket(true);
  if (err > 0) {
    Serial.println("Message sent correctly!");
  } else {
    Serial.println("Error sending message :(");
```

```
    Serial.println("(you may send a limited amount of messages per minute, depending
 on the signal strength");
    Serial.println("it may vary from 1 message every couple of seconds to 1 message
 every minute)");
   }
   delay(5000);
}

void sendsteps2()
{
   h2 = steps;
   activity = abs(((h2 - 0) * 10) / 4);
   int err;
   modem.beginPacket();
   modem.print(activity);
   err = modem.endPacket(true);
   if (err > 0) {
     Serial.println("Message sent correctly!");
   } else {
     Serial.println("Error sending message :(");
     Serial.println("(you may send a limited amount of messages per minute, depending
on the signal strength");
     Serial.println("it may vary from 1 message every couple of seconds to 1 message
every minute)");
   }
   delay(5000);
}

void sendsteps3()
{
   h3 = steps;
```

```
  activity = abs(((h3 - h2) * 10) / 4);
  int err;
  modem.beginPacket();
  modem.print(activity);
  err = modem.endPacket(true);
  if (err > 0) {
    Serial.println("Message sent correctly!");
  } else {
    Serial.println("Error sending message :(");
    Serial.println("(you may send a limited amount of messages per minute, depending
on the signal strength");
    Serial.println("it may vary from 1 message every couple of seconds to 1 message
every minute)");
  }
  delay(5000);
}

void sendsteps4()
{
  h4 = steps;
  activity = abs(((h4 - h3) * 10) / 4);
  int err;
  modem.beginPacket();
  modem.print(activity);
  err = modem.endPacket(true);
  if (err > 0) {
    Serial.println("Message sent correctly!");
  } else {
    Serial.println("Error sending message :(");
    Serial.println("(you may send a limited amount of messages per minute, depending
on the signal strength");
```

```
    Serial.println("it may vary from 1 message every couple of seconds to 1 message
 every minute)");
  }
  delay(5000);
}

void sendsteps5()
{
  h5 = steps;
  activity = abs(((h5 - h4) * 10) / 4);
  int err;
  modem.beginPacket();
  modem.print(activity);
  err = modem.endPacket(true);
  if (err > 0) {
    Serial.println("Message sent correctly!");
  } else {
    Serial.println("Error sending message :(");
    Serial.println("(you may send a limited amount of messages per minute, depending
on the signal strength");
    Serial.println("it may vary from 1 message every couple of seconds to 1 message
every minute)");
  }
  delay(5000);
}

void sendsteps6()
{
  h6 = steps;
  activity = abs(((h6 - h5) * 10) / 4);
  int err;
```

```
  modem.beginPacket();
  modem.print(activity);
  err = modem.endPacket(true);
  if (err > 0) {
    Serial.println("Message sent correctly!");
  } else {
    Serial.println("Error sending message :(");
    Serial.println("(you may send a limited amount of messages per minute, depending
on the signal strength");
    Serial.println("it may vary from 1 message every couple of seconds to 1 message
every minute)");
  }
  delay(5000);
}
```

Source Code ของ arduino_secrets.h

```
// Replace with keys obtained from TheThingsNetwork console
#define SECRET_APP_EUI "70B3D57ED00140F9"
#define SECRET_APP_KEY "E327BAD9F4C251F10AB8D217C4C7A2F1"
```