ระบบจัดการคิวสำหรับร้านอาหาร
QUEUE MANAGEMENT SYSTEM FOR RESTAURANTS

โดย
นายกิตตินันท์ สว่างพร้อง
นายชาคร แตงอ่อน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

ระบบจัดการคิวสำหรับร้านอาหาร

QUEUE MANAGEMENT SYSTEM FOR RESTAURANTS

โดย

นายกิตตินันท์ สว่างพร้อง 57010088

นายชาคร แตงอ่อน 57010309

อาจารย์ที่ปรึกษา

รศ.ดร. ยุทธพงษ์ รังสรรค์เสรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

ผ่านการตรวจรูปเล่มแล้ว

(.....)

อาจารย์ที่ปรึกษา

21/05/61

ผ่านการตรวจชิ้นงานแล้ว

(.....)

กรรมการผู้ตรวจชิ้นงาน

21/8/61

กิตติกรรมประกาศ

ปริญญานิพนธ์ครั้งนี้ประสบความสำเร็จได้ด้วยดีเนื่องจากคำแนะนำและข้อชี้แนะจากอาจารย์ที่ปรึกษา รศ.ดร.ยุทธพงษ์ รังสรรค์เสรี ที่ให้คำปรึกษา ทางคณะผู้จัดทำขอขอบพระคุณเป็นอย่างสูงและขอบคุณเพื่อนๆที่คอยให้คำปรึกษาเกี่ยวกับโครงงานนี้จนสำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ผู้จัดทำขอขอบพระคุณ บิดา มารดาและครอบครัวของคณะผู้จัดทำที่ให้การสนับสนุนช่วยเหลือผู้จัดทำ และให้กำลังใจตลอดมา คณะผู้จัดทำหวังว่าปริญญานิพนธ์เรื่องนี้จะมีประโยชน์และสามารถนำไปประยุกต์ใช้ในงานวิจัยอื่นๆและนำโครงงานนี้ไปต่อยอดให้ดีขึ้นต่อไป

นายกิตตินันท์ สว่างพร้อง

นายชาคร แตงอ่อน

ผู้จัดทำ

ปริญญานิพนธ์ปีการศึกษา 2560

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบจัดการคิวสำหรับร้านอาหาร

QUEUE MANAGEMENT SYSTEM FOR RESTAURANTS

ผู้จัดทำ

1 นายกิตตินันท์ สว่างพร้อง 57010088

2. นายชาคร แตงอ่อน 57010309

..... อาจารย์ที่ปรึกษา

(รศ.ดร. ยุทธพงษ์ รังสรรค์เสรี)

ระบบจัดการคิวสำหรับร้านอาหาร

QUEUE MANAGEMENT SYSTEM RESTAURANTS

โดย นายกิตตินันท์ สว่างพร้อง 57010088

นายชาคร แตงอ่อน 57010309

อาจารย์ที่ปรึกษา รศ.ดร. ยุทธพงษ์ รังสรรค์เสรี

บทคัดย่อ

โครงงานชิ้นนี้เป็นการจัดทำระบบจัดการคิวสำหรับร้านอาหาร โดยฝั่งส่งจะใช้ไมโครคอนโทรลเลอร์ Arduino mega ADK ซึ่งจะมีการป้อนอินพุตผ่าน Serial Monitor บอร์ด Arduino mega ADK จะส่งสัญญาณที่มีโค้ดหมายเลขเครื่องรับนั้น ผ่าน Lora module เมื่อเครื่องรับ (Arduino Nano) ได้รับโค้ดหมายเลขตรงกับสัญญาณหมายเลขที่เครื่องส่งได้ส่งมา และตรวจสอบว่าตรงกับโค้ดของเครื่องที่ตั้งไว้ ถ้าตรงกัน ไฟจะกระพริบที่ LED จากนั้น Buzzer จะส่งเสียงแจ้งเตือน และแสดงข้อความแจ้งเตือนผ่านหน้าจอ OLED

ABSTRACT

This project is the preparation of queue management system for restaurants by using Arduino microcontroller. Inputs are inserted through the serial monitor and target number will be send out by Arduino Mega ADK with LoRa Module. When Arduino Nano receives a signal by LoRa module, it will make sure codes match each other. If codes are matched, it will alarm by LED, Buzzer, and display on OLED.

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป(ต่อ)

สารบัญตาราง

หน้า

รูปที่

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันเทคโนโลยีสารสนเทศเข้ามาเป็นส่วนหนึ่งของการดำเนินชีวิตของเรามากขึ้น ในด้านของการจัดการเพื่อความสะดวกสบายและมีประสิทธิภาพมากยิ่งขึ้น ระบบจัดการคิวสำหรับร้านอาหารโดยใช้เทคโนโลยีเข้ามาจะสามารถช่วยลดปัญหาการเข้ารอคิวเพื่อรอรับอาหารจากทางร้านค้าที่มีลูกค้าเป็นจำนวนมากและอาจจะต้องรอเป็นเวลานานสำหรับร้านค้าประเภทอาหารตามสั่ง โดยทางลูกค้าของร้านไม่จำเป็นต้องมารอยืนต่อคิว เพียงแค่รอการแจ้งเตือนจากอุปกรณ์ เมื่ออาหารนั้นพร้อมเสิร์ฟร้านค้าจะแจ้งเตือนผ่านหน้าจอคอมพิวเตอร์ของทางร้านให้ลูกค้ามารับอาหาร

1.2 วัตถุประสงค์

1) เพื่อศึกษาการสร้างระบบจัดการคิวสำหรับร้านอาหาร
2) เพื่อศึกษาการเขียนโปรแกรมในการควบคุมการทำงานของ Arduino
3) เพื่อศึกษาการส่งสัญญาณแบบบรอดแคสโดยใช้คลื่นวิทยุ
4) เพื่อให้ร้านอาหารมีการจัดการอย่างมีประสิทธิภาพ
5) เพื่อสร้างเอกลักษณ์ และมูลค่าสินค้าและบริการ

1.3 ขอบเขตของปริญญานิพนธ์

1) สร้างระบบสั่งการผ่านสัญญาณคลื่นวิทยุ
2) ทำให้เครื่องรับสามารถแจ้งเตือนเมื่อได้รับสัญญาณจากเครื่องส่ง
3) สามารถออกแบบระบบโปรแกรมสั่งอาหารผ่านคอมพิวเตอร์

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 ภาษาที่ใช้เขียน Arduino

โครงสร้างโปรแกรมภาษา C++ บน Arduino จะมีลักษณะแบบเดียวกับภาษา C++ ทั่วๆไป แต่สำหรับผู้ที่ยังไม่เคยทำการเขียนโปรแกรมภาษาใดๆมาก่อน ต้องทำความเข้าใจในเรื่องต่างๆ

2.1.1 Preprocessor directives

โดยปกติแล้วเกือบทุกโปรแกรมต้องมี Preprocessor directives โดยส่วนนี้จะเป็นส่วนที่ Compiler จะมีการประมวลผลและทำตามคำสั่งก่อนที่จะมีการคอมไพล์โปรแกรม ซึ่งจะเริ่มต้นด้วยเครื่องหมาย Directive หรือเครื่องหมาย # แล้วจึงตามด้วยชื่อคำสั่งที่ต้องการเรียกใช้ หรือกำหนด โดยปกติแล้วส่วนนี้จะอยู่ในส่วนหัวของโปรแกรม และต้องอยู่นอกฟังก์ชั่นหลักใดๆทั้งหมด #include เป็นคำสั่งที่ใช้อ้างอิงไฟล์ภายนอก เพื่อเรียกใช้ฟังก์ชั่น หรือตัวแปรที่มีการสร้างหรือกำหนดไว้ในไฟล์นั้น รูปแบบการใช้งานคือ #include <ชื่อไฟล์.h> ตัวอย่าง #include <Wire.h> #include <Time.h> เป็นต้น จากตัวอย่าง จะเห็นว่าได้มีการอ้างอิงไฟล์ Wire.h และไฟล์ Time.h ซึ่งเป็น Library พื้นฐานที่มีอยู่ใน Arduino

การอ้างอิงไฟล์จากภายใน หรือการอ้างอิงไฟล์ Library ที่มีอยู่แล้วใน Arduino หรือเป็น Library ที่เราเพิ่มเข้าไปเอง จะใช้เครื่องหมาย < > ในการคร่อมชื่อไฟล์ไว้ เพื่อให้โปรแกรม Compiler เข้าใจว่าควรไปหาไฟล์เหล่านี้จากในโฟลเดอร์ Library แต่หากต้องการอ้างอิงไฟล์ที่อยู่ในโฟลเดอร์ Project จะต้องใช้เครื่องหมาย " " คร่อมแทน ซึ่ง Compiler จะ Link ไปหาไฟล์นี้โดยอ้างอิงจากไฟล์โปรแกรมที่ Compiler อยู่ เช่น #include "myFunction.h" เป็นต้น จากตัวอย่างด้านบน Complier จะ Link ไปหาไฟล์ myFunction.h ภายในโฟลเดอร์ Project ทันที หากไม่พบก็จะเกิด Error เมื่อทำการ Verify โปรแกรม #define เป็นคำสั่งที่ใช้ในการแทนข้อความที่กำหนดไว้ด้วยข้อความที่กำหนดไว้ ซึ่งการใช้คำสั่ง นี้ข้อดีคือ จะไม่มีการอ้างอิงกับตัวโปรแกรมเลย

รูปแบบคือ #define NAME VALUE ตัวอย่าง #define LEDPIN 13 เป็นต้น

จากตัวอย่าง ไม่ว่าคำว่า LEDPIN จะอยู่ส่วนใดของโค้ดโปรแกรมก็ตาม Compiler จะแทนคำว่า LEDPIN ด้วยเลข 13 ซึ่งข้อดี คือ เราไม่ต้องสร้างเป็นตัวแปรขึ้นมาเพื่อเปลืองพื้นที่ และยังช่วยให้โปรแกรมทำงานเร็วขึ้นอีกด้วยเพราะ CPU ไม่ต้องไปขอข้อมูลมาจาก RAM หลายๆต่อ

2.1.2 ส่วนของการกำหนดค่า (Global declarations)

ส่วนนี้เป็นส่วนที่ใช้ในการกำหนดชนิดตัวแปรแบบนอกฟังก์ชั่น หรือประกาศฟังก์ชั่น เพื่อให้ฟังก์ชั่นที่ประกาศสามารถกำหนด หรือเรียกใช้ได้จากทุกส่วนของโปรแกรมเช่น int pin = 13; void blink(void); เป็นต้น

2.1.3 ฟังก์ชั่น setup () และฟังก์ชั่น loop ()

ฟังก์ชั่น setup() และฟังก์ชั่น loop() เป็นคำสั่งที่ถูกบังคับให้ต้องมีในทุกโปรแกรม โดยฟังก์ชั่น setup() จะเป็นฟังก์ชั่นแรกที่ถูกเรียกใช้ นิยมใช้กำหนดค่า หรือเริ่มต้นใช้งาน Library ต่างๆ เช่น ในฟังก์ชั่น setup() จะมีคำสั่ง pinMode() เพื่อกำหนดให้ขาใดๆก็ตามเป็นดิจิตอลอินพุต หรือเอาต์พุต ส่วนฟังก์ชั่น loop() จะเป็นฟังก์ชั่นที่ทำงานหลังจากฟังก์ชั่น setup() ได้ทำงานเสร็จสิ้นไปแล้ว และมีการวนรอบแบบไม่รู้จบ เมื่อฟังก์ชั่น loop() งานครบตามคำสั่งแล้ว ฟังก์ชั่น loop() ก็จะถูกเรียกขึ้นมาใช้อีก

ตัวอย่าง Code

```
int pin = 13;
void setup () {
    pinMode (pin, OUTPUT);
}
void loop () {
  digitalWrite(pin, HIGH);
  delay (1000);
  digitalWrite(pin, LOW);
  delay (1000);
}
```

รูปที่ 2.1 โค้ดฟังก์ชัน setup และ loop [1]

จากตัวอย่าง จะเห็นว่ามีการประกาศตัวแปรแบบนอกฟังก์ชั่น ทำให้สามารถกำหนด หรือเรียกใช้จากในฟังก์ชั่นใดๆก็ตามได้ ในฟังก์ชั่น setup() ได้มีการกำหนดให้ขาที่ 13 เป็นดิจิตอลเอาต์พุต และในฟังก์ชั่น loop() มีการกำหนดให้พอร์ต 13 มีลอจิกเป็น 1 และใช้ฟังก์ชั่น delay() ในการหน่วงเวลา 1 วินาที แล้วจึงกำหนดให้พอร์ต 13 มีสถานะลอจิกเป็น 0 แล้วจึงหน่วงเวลา 1 วินาที จบฟังก์ชั่น loop() และจะเริ่มทำฟังก์ชั่น loop() ใหม่ ผลที่ได้คือไฟกระพริบบนบอร์ด Arduino Uno ในพอร์ตที่ 13 ทำงานแบบไม่รู้จบ

ในทุกๆการทำงานของฟังก์ชั่น จะต้องเริ่มด้วยการกำหนดค่าที่ส่งกลับ ตามด้วยชื่อฟังก์ชั่น แล้วตามด้วยเครื่องหมายปีกกาเปิด และจบด้วยเครื่องหมายปีกกาปิด ภายในฟังก์ชั่น หากจะเรียกฟังก์ชั่นใช้งานย่อยใดๆ จะต้องมีเครื่องหมาย ; ต่อท้ายเสมอ

* การกำหนดชนิดค่าที่ส่งกลับเป็น void หมายถึง ไม่มีการส่งค่ากลับ แต่สามารถใช้คำสั่ง return; ตรงๆได้ เพื่อให้จบการทำงานของฟังก์ชั่นก่อนจะไปถึงบรรทัดสุดท้ายของฟังก์ชั่น

2.2 คลื่นวิทยุ (Broadcast Radio) และการบรอดแคส (Broadcasting)

คลื่นวิทยุเป็นสื่อกลางที่ใช้ส่งสัญญาณไปในอากาศ โดยสามารถส่งในระยะทางได้ทั้งใกล้และไกล โดยมีตัวกระจายสัญญาณ (broadcast) ส่งไปยังตัวรับสัญญาณ และใช้คลื่นวิทยุในช่วงความถี่ต่างๆกันในการส่งข้อมูลเช่น การสื่อสารระยะไกลในการกระจายเสียงวิทยุระบบเอเอ็ม (Amplitude Modulation: AM) และเอฟเอ็ม (Frequency Modulation: FM) หรือการสื่อสารระยะใกล้โดยใช้ไวไฟ(Wi-Fi)และบลูทูธ(Bluetooth)

บรอดแคสนั้นเป็นการส่งข้อมูลจากโหนดต้นทางหนึ่งโหนดไปยังโหนดปลายทางทุกโหนดที่ติดต่ออยู่ในลักษณะของการแพร่กระจายข้อมูล แบบ 1 ต่อ ทั้งหมด หรือเรียกว่า One-to-All ซึ่งการแพร่ข้อมูลแบบส่งไปยังโหนดทุกโหนดนั้น จะต้องมีการประมวลผลข้อมูลที่โหนดปลายทาง ส่วนโหนดที่ไม่ต้องการรับข้อมูลนั้นก็จะได้รับข้อมูลไปด้วย แต่ต้องทิ้งข้อมูลที่ได้รับมา เป็นการสูญเสียความสามารถในการประมวลผลไป อีกทั้งยังทำให้มีปริมาณข้อมูลส่งอยู่ในเครือข่ายจำนวนมากโดยเปล่าประโยชน์ และสามารถเกิดเป็นปัญหา พายุข้อมูล (Broadcast storm) ได้ การสื่อสารแบบ Broadcast นี้ปัจจุบันมีการใช้งานอยู่เฉพาะใน LAN เท่านั้น เนื่องจากเป็นการยากในการหาเส้นทาง

องค์ประกอบขั้นพื้นฐานของระบบสื่อสารโทรคมนาคมสามารถจำแนกออกเป็นส่วนประกอบได้ดังต่อไปนี้

1. ผู้ส่งข่าวสาร หรือแหล่งกำเนิดข่าวสาร (source) อาจจะเป็นสัญญาณต่าง ๆ เช่น สัญญาณภาพข้อมูล และเสียงเป็นต้น ในการติดต่อสื่อสารสมัยก่อนอาจจะใช้แสงไฟ ควันไฟ หรือท่าทางต่าง ๆ ก็นับว่าเป็นแหล่งกำเนิดข่าวสารจัดอยู่ในหมวดหมู่นี้เช่นกัน

2. ผู้รับข่าวสาร หรือจุดหมายปลายทางของข่าวสาร (sink) ซึ่งจะรับรู้จากสิ่งที่ผู้ส่งข่าวสาร หรือแหล่งกำเนิดข่าวสารส่งผ่านมาให้ตราบใดที่ การติดต่อสื่อสารบรรลุวัตถุประสงค์ ผู้รับสารหรือจุดหมายปลายทางของข่าวสารก็จะได้รับข่าวสารนั้นๆถ้าผู้รับสารหรือจุดหมายปลายทางไม่ได้รับข่าวสาร ก็แสดงว่าการสื่อสารนั้นไม่ประสบความสำเร็จ กล่าวคือไม่มีการสื่อสารเกิดขึ้นนั่นเอง

3. ช่องสัญญาณ (channel) ในที่นี้อาจจะหมายถึงสื่อกลางหรือตัวกลางที่ข่าวสารเดินทางผ่านอาจจะเป็นอากาศ สายนำสัญญาณต่าง ๆ หรือแม้กระทั่งของเหลว เช่น น้ำ น้ำมัน เป็นต้น เปรียบเสมือนเป็นสะพานที่จะให้ข่าวสารข้ามจากฝั่งหนึ่งไปยังอีกฝั่งหนึ่ง

4. การเข้ารหัส (encoding) เป็นการช่วยให้ผู้ส่งข่าวสารและผู้รับข่าวสารมีความเข้าใจตรงกันในการสื่อความหมาย จึงมีความจำเป็นต้องแปลงความหมายนี้ การเข้ารหัสจึงหมายถึงการแปลงข่าวสารให้อยู่ในรูปพลังงาน ที่พร้อมจะส่งไปในสื่อกลาง ทางผู้ส่งมีความเข้าใจต้องตรงกันระหว่างผู้ส่งและผู้รับหรือมีรหัสเดียวกัน การสื่อสารจึงเกิดขึ้นได้

5. การถอดรหัส (decoding) หมายถึงการที่ผู้รับข่าวสารแปลงพลังงานจากสื่อกลางให้กลับไปอยู่ในรูปข่าวสารที่ส่งมาจากผู้ส่งข่าวสารโดยมีความเข้าในหรือรหัสตรงกัน

6. สัญญาณรบกวน (noise) เป็นสิ่งที่มีอยู่ในธรรมชาติ มักจะลดทอนหรือรบกวนระบบ อาจจะเกิดขึ้นได้ทั้งทางด้านผู้ส่งข่าวสาร ผู้รับข่าวสาร และช่องสัญญาณ แต่ในการศึกษาขั้นพื้นฐานมักจะสมมติให้ทางด้านผู้ส่งข่าวสารและผู้รับข่าวสารไม่มีความผิดพลาด ตำแหน่งที่ใช้วิเคราะห์มักจะเป็นที่ตัวกลางหรือช่องสัญญาณ เมื่อไรที่รวมสัญญาณรบกวนด้านผู้ส่งข่าวสารและด้านผู้รับข่าวสาร ในทางปฏิบัติมักจะใช้ วงจรกรอง (filter) กรองสัญญาณแต่ต้นทาง เพื่อให้การสื่อสารมีคุณภาพดียิ่งขึ้นแล้วค่อยดำเนินการ เช่น การเข้ารหัสแหล่งข้อมูล เป็นต้น

ข่ายการสื่อสารข้อมูล หมายถึง การรับส่งข้อมูลหรือสารสนเทศจากที่หนึ่งไปยังอีกที่หนึ่ง โดยอาศัยระบบการส่งข้อมูล ทางคลื่นไฟฟ้าหรือแสง อุปกรณ์ที่ประกอบเป็นระบบการสื่อสารข้อมูล โดยทั่วไปเรียกว่า "ข่ายการสื่อสารข้อมูล (Data Communication Networks)"

องค์ประกอบพื้นฐาน

1. หน่วยส่งข้อมูล (Sending Unit)
2. ช่องทางการส่งข้อมูล (Transmission Channel)
3. หน่วยรับข้อมูล (Receiving Unit)

วัตถุประสงค์หลักของการนำการสื่อการข้อมูลมาประยุกต์ใช้ในองค์การประกอบด้วย

1. เพื่อรับข้อมูลและสารสนเทศจากแหล่งกำเนิดข้อมูล
2. เพื่อส่งและกระจายข้อมูลได้อย่างรวดเร็ว
3. เพื่อลดเวลาการทำงาน
4. เพื่อการประหยัดค่าใช้จ่ายในการส่งข่าวสาร
5. เพื่อช่วยขยายการดำเนินการองค์การ

2.3 UART

UART ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter หมายถึง อุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส (Asynchronous) ซึ่งเป็นส่วนหนึ่งในการสื่อสารอนุกรม แบบ Asynchronous แท้จริงแล้วการสื่อสารแบบอนุกรมจะแบ่งเป็น 2 แบบ คือ

1) การสื่อสารอนุกรมแบบ Synchronous เป็นรูปแบบที่ใช้วิธีส่งข้อมูล โดยใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะ การรับส่งข้อมูล การส่งข้อมูลแบบนี้ เป็นการรับส่งที่ค่อนข้างมีคุณภาพ และส่งได้ที่ความเร็วสูง มีโอกาสที่ข้อมูลจะสูญหายระหว่างการส่งน้อย ตัวอย่างการส่งข้อมูลลักษณะนี้เช่น I2C, I2S, SPI ข้อเสียของการส่งข้อมูลแบบนี้คือ ต้องใช้สายสัญญาณมาก เพราะว่าต้องส่ง Clock ไปด้วย

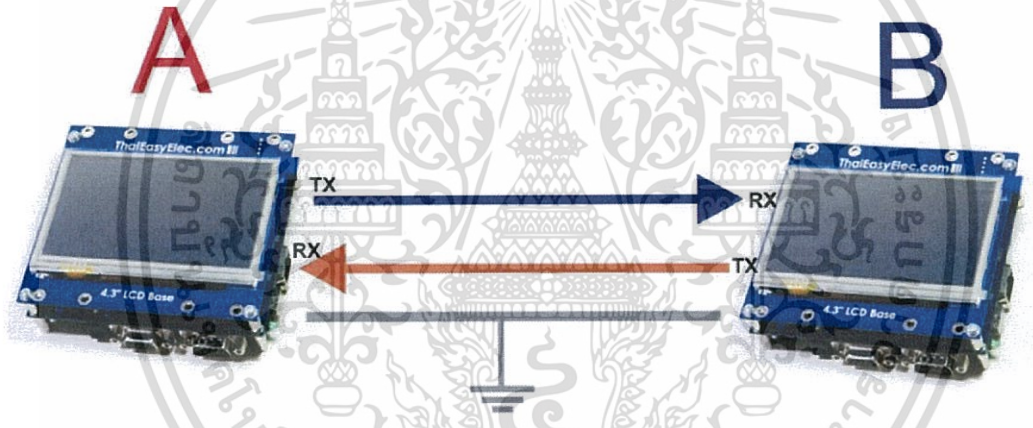
รูปที่ 2.2 การส่งบิตข้อมูลเทียบกับสัญญาณนาฬิกา [4]

2) การสื่อสารอนุกรมแบบ Asynchronous เป็นการส่งข้อมูลที่ไม่ต้องใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะการรับส่งข้อมูล แต่ใช้วิธีกำหนดรูปแบบ Format การรับส่งข้อมูลขึ้นมาแทน และ อาศัยการกำหนดความเร็วของการรับและส่งที่เท่ากันทั้งฝั่งรับและฝั่งส่ง ข้อดีของการใช้ Asynchronous คือสามารถสื่อสารแบบ Full Duplex รับและส่งได้ในเวลาเดียวกัน แต่ Asynchronous มีโอกาสที่ข้อมูลจะสูญหายขณะรับส่งข้อมูล หรือ รับส่งข้อมูลผิดพลาดได้มากกว่าแบบ Synchronous สรุปกล่าวคือ UART (Universal Asynchronous Receiver Transmitter) หมายถึง รูปแบบการส่งข้อมูล ที่ถูกกำหนดขึ้นมาเพื่อใช้รับส่งข้อมูลแบบ Asynchronous โดยมีรูปแบบดังรูป

รูปที่ 2.3 การส่งข้อมูล Asynchronous [4]

เริ่มต้นจาก Start Bit เป็น Logic 0 จากนั้นจะตามด้วย Data ที่เราส่ง แล้วจะถูกปิดด้วย STOP Bit เป็น Logic 1 สำหรับผู้เริ่มต้นศึกษา ถ้าคุณต้องการติดต่อสื่อสาร UART ระหว่างไมโครคอนโทรลเลอร์ กับ คอมพิวเตอร์ สามารถทำได้โดยใช้โปรแกรม HyperTerminal เข้าช่วยเพื่อรับและส่งข้อมูล

2.3.1 แนวคิดวิธีเชื่อมต่อระหว่างอุปกรณ์



รูปที่ 2.4 การเชื่อมต่อระหว่างอุปกรณ์ [4]

จากรูปแสดงถึงการเชื่อมต่อระหว่างบอร์ด SUN7 เพื่อส่ง Data หากันด้วย UART

1) A ส่งข้อมูล ออกไปทางขา Tx ไปยัง B ซึ่งเป็นฝั่งรับ เพราะฉะนั้น ต้องต่อสายสัญญาณจากขา Tx ของ A ไปยังขา Rx ของ B

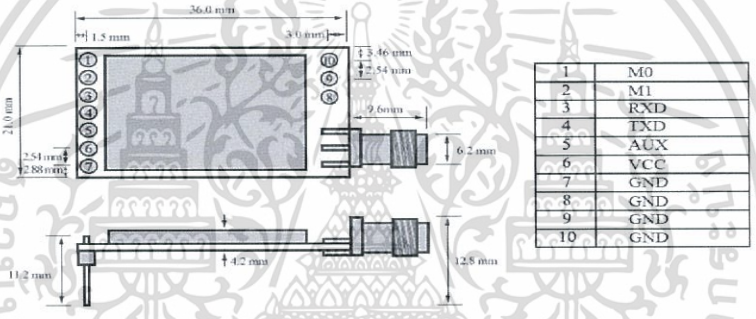
2) B ส่งข้อมูล ออกไปทางขา Tx ไปยัง A ซึ่งเป็นฝั่งรับ เพราะฉะนั้น ต้องต่อสายสัญญาณจากขา Tx ของ B ไปยังขา Rx ของ A

3) ต้องต่อ GND ของทั้ง A และ B ร่วมกันเพื่อทำให้ระดับแรงดันของทั้ง 2 บอร์ดมีจุดอ้างอิงเดียวกัน

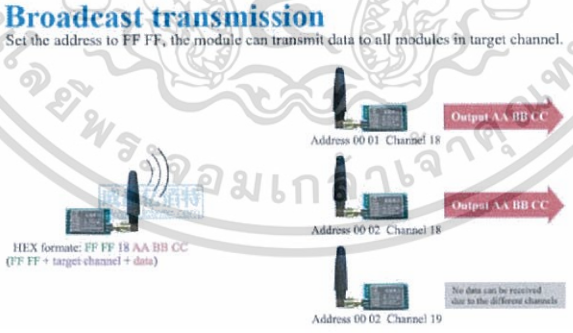
Product Description

Model	E32-TTL-100
Interface	UART
Power	20dBm
Distance	3000m
RF connector	SMA-K
Frequency	433MHz (410-441)

ตารางที่ 2.1 คุณสมบัติของ LoRa [5]



รูปที่ 2.5 พอร์ตการเชื่อมต่างๆของ LoRa Module [5]



รูปที่ 2.6 การส่งสัญญาณแบบบรอดแคส [5]

2.4 อุปกรณ์

2.4.1 บอร์ด Arduino Mega ADK

รูปที่ 2.7 Arduino Mega ADK [6]

รายละเอียดการเชื่อมต่อของบอร์ด Arduino Mega ADK

Pins	**รายละเอียดการเชื่อมต่อของบอร์ด Arduino Mega ADK**
VIN	input voltage ของบอร์ด Arduino โดยใช้แหล่งจ่ายจากภายนอก
5 V	output pin ที่ควบคุม 5 V จากบอร์ด
3.3 V	supply ที่สร้างขึ้นจาก regulator บนบอร์ด และให้กระแสได้สูงสุด 50 mA
GND	ground pin
IOREF	pin ที่ให้ voltage reference กับไมโครคอนโทรลเลอร์ เพื่อเลือกค่าแรงดันให้กับ shield ที่มาเชื่อมต่อกับบอร์ด
digital pins 54 pins	input และ output โดยจะทำงานที่แรงดัน 5 V และให้กระแสสูงสุด 40 mA
PWM	PWM 2 ถึง 13 และ 44 ถึง 46 ให้ output PWM output 8-bits
SPI	50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS) ใช้สำหรับรองรับการสื่อสารแบบ SPI โดยที่ไม่เกี่ยวข้องกันกับ ICSP header ซึ่งจะมีลักษณะคล้ายกับ Uno, Duemilanove และ Diecimila
LED 13	build-in LED ที่เชื่อมต่อกับ digital pin 13 เมื่อ pin มีค่าเป็น HIGH LED จะ

	ติด , แต่เมื่อ pin เป็น LOW LED จะดับ
TWI	20 (SDA) and 21 (SCL). รองรับการเชื่อมต่อแบบ TWI (I2C)
Reset	ใช้ในการ reset ไมโครคอนโทรลเลอร์ โดยทั่วไปจะใช้โดยการเพิ่มปุ่ม reset ไว้บน shield เพื่อป้องกันปุ่มที่อยู่บนบอร์ด

ตารางที่ 2.2 การเชื่อต่อบอร์ด Arduino Mega ADK [6]

รายละเอียด Arduino MEGA 2560 R3

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output , 4 UART TTL)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

รูปที่ 2.8 รายละเอียดของ Arduino MEGA ADK [6]

2.4.2 LoRa Module

LoRa เป็นเทคโนโลยีการมอดูเลชั่น หรือการเข้ารหัสสัญญาณทางไฟฟ้าในรูปแบบเฉพาะ การมอดแบบ LoRa เป็นลิขสิทธิ์เฉพาะของบริษัท Semtech ที่ผูกขาดลิขสิทธิ์การมอดแบบ LoRa และเป็นผู้เดียวที่ผลิตชิปไอซีสื่อสารไร้สายที่มีการมอดแบบ LoRa ได้ การใช้งาน LoRa สามารถใช้งานได้แบบเสรี คือหากมีฮาร์ดแวร์ ก็สามารถนำมาใช้ในการสื่อสารได้เลย โดยไม่จำเป็นต้องใช้บริการจากผู้ให้บริการต่างๆ

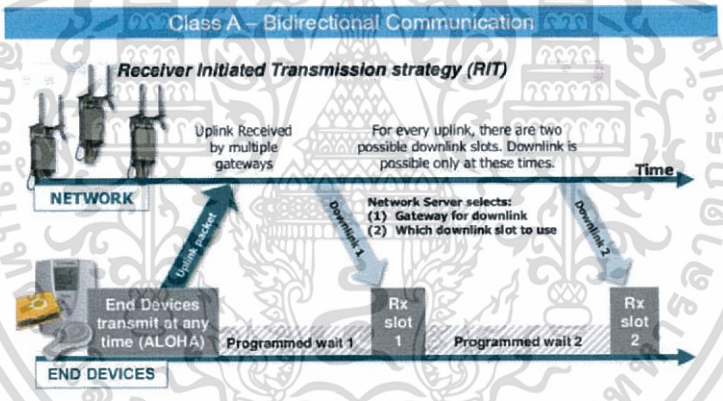
คำว่า “LoRa” ย่อมาจาก Long range มักถูกนำไปใช้เรียกอุปกรณ์ที่ใช้การมอดแบบ LoRa เนื่องจากอุปกรณ์กลุ่มนี้สามารถสื่อสารด้วยกันได้ โดยจะตั้งอยู่บนพื้นฐานของระยะห่างระหว่างตัวรับ และตัวส่ง อัตราขยายของเสาอากาศ กำลังส่ง และสัญญาณรบกวน สิ่งพื้นฐานเหล่านี้ล้วนเป็นตัวแปรที่ทำให้ระยะในการส่งข้อมูลมีมากขึ้นหรือลดลง รวมถึงความเร็วในการสื่อสาร และการตกหล่นของข้อมูล

2.4.2.1 LoRa Class

สาเหตุที่ LoRa WAN ประหยัดพลังงาน เนื่องจากตัวอุปกรณ์ จะอยู่ในสภาวะ “Sleep” เมื่อไม่จำเป็นต้องส่งข้อมูล โดย LoRa มีโหมดการทำงาน 3 คลาส (ปัจจุบันมีให้ใช้เพียง 2 คลาสคือ A และ B) ดังนี้

Class Name	Intended Usage
A (“all”)	Battery powered sensors (or actuators with no latency constraint) Most energy efficient communication class. Must be supported by all devices.
B (“beacon”)	Battery powered actuators Energy efficient communication class for latency controlled downlink. Based on slotted communication synchronized with a network beacon.
C (“continuous”)	Main powered actuators Devices witch can afford to listen continuously. No latency for downlink communication.

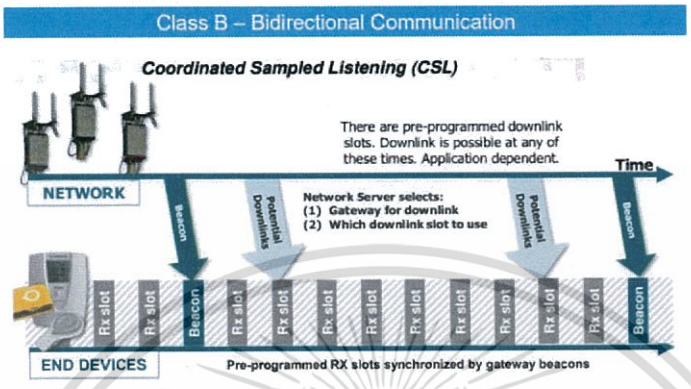
รูปที่ 2.9 โหมดประหยัดพลังงานทั้ง 3 คลาสของ LoRa [7]



รูปที่ 2.10 โหมดประหยัดพลังงานของ LoRa คลาส A [7]

Class A: เป็นวิธีที่อุปกรณ์ LoRa ส่งข้อมูลไม่บ่อย อยากจะส่งก็ส่งไปเลย (เหมือนหลักการของ ALOHA) เช่น Smoke detector ต้องการส่งข้อมูลไปหาตัวรับ Gateway (Uplink) เมื่อมีเหตุการณ์เกิดขึ้น ฝั่งรับ (Gateway) จะส่งข้อมูลรับกลับมาให้ตัวส่ง (downlink) ซึ่งก็ต้องรอช่วงเวลา ที่พอดีกับตัวอุปกรณ์ LoRa เปิดรอรับข้อมูล

ข้อดีคือประหยัดพลังงานมาก แต่ข้อเสีย คือ downlink ที่ส่งกลับมาหาตัวอุปกรณ์ LoRa อาจจะต้องรอเวลาให้พอดีกัน ซึ่งก็มีโอกาสที่ข้อมูลอาจจะสูญหาย หากจังหวะการส่งไม่พอดีกัน



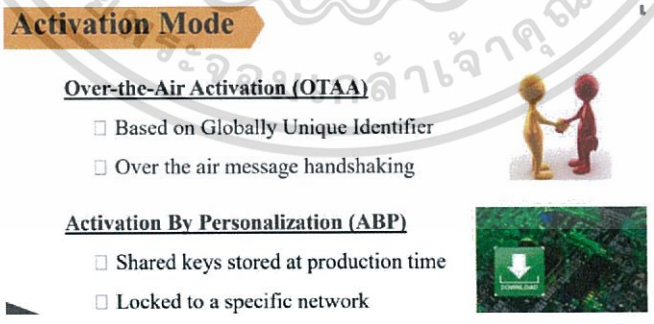
รูปที่ 2.11 โหมดประหยัดพลังงานของ LoRa คลาส B [7]

Class B: เป็นรูปแบบการส่งข้อมูลที่ได้ดีขึ้น เพราะมีการเปิด downlink จากคลาส A บ่อยขึ้น (ตามรูป) และจะกินพลังงานมากกว่าคลาส A แต่น้อยกว่าคลาส C

Class C: เปิด uplink/downlink ตลอดเวลาสำหรับงานที่ต้องการความเสถียร แต่กินพลังงานมากที่สุด

2.4.2.2 กระบวนการเชื่อมต่อ

LoRa มีโหมดการ Activate ก่อนจะเชื่อมต่อ 2 โหมด คือ OTAA กับ ABP ดังนี้



รูปที่ 2.12 โหมดการทำงานของ LoRa [7]

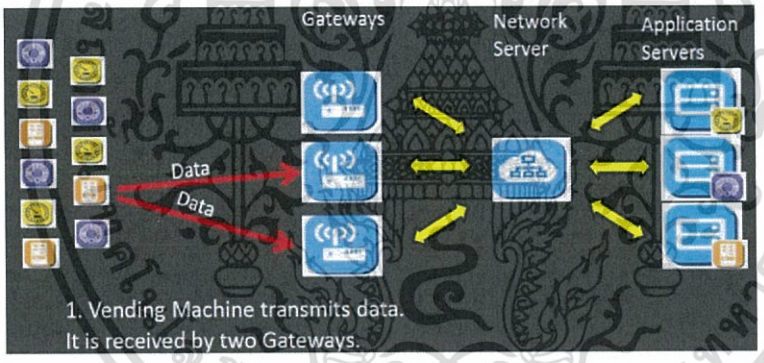
โหมด OTAA ถ้ารู้ต้นทางกับปลายทาง ใช้ devEUI, AppEUI / AppKey ส่งหากันทำ handshake ก่อน เพื่อใช้ในการตรวจสอบ ก่อนที่จะมีการเชื่อมต่อ การใช้โหมดนี้มีความยืดหยุ่นมากกว่า เพราะในกรณีที่เปลี่ยนเครือข่ายก็ยังสามารถใช้งาน Configure เดิมได้

โหมด ABP รู้ว่าต้นทางอยู่ที่ไหน ปลายทางอยู่ที่ไหน ไม่ต้องตรวจสอบมาก กำหนด dev Address, Network key และ App Session key ใช้ในการเชื่อมต่อไปเลย ไม่ต้องมีการทำ handshake มีข้อดี เมื่อเวลา start อุปกรณ์ขึ้นก็สามารถเชื่อมต่อได้เลย เร็วกว่าแบบ OTAA แต่หากเปลี่ยนเครือข่าย ก็ต้องมีการ Configure กันใหม่

2.4.2.3 LoRa WAN

อุปกรณ์ LoRa Device ส่งข้อมูลต่อไปยัง LoRa network ผ่านทาง LoRa Gateway

2.4.2.4 LoRa Gateway



รูปที่ 2.13 LoRa Gateway [7]

สำหรับกรณีที่ทำ LoRa เพียงตัวเดียว และ มี Gateway ให้เชื่อมต่อหลายตัว Gateway ก็จะได้รับข้อมูลซ้ำกัน เมื่อส่งต่อไปยัง network server แล้ว ตัว server จะนำข้อมูลทั้ง timestamp และความแรงของสัญญาณ เพื่อหา packet ที่ซ้ำกัน และทิ้งตัวที่มีความแรงของสัญญาณน้อยกว่า และเลือกที่จะส่งต่อตัวที่มีความแรงของสัญญาณที่ดีกว่าต่อไปยัง Application server ต่อไป

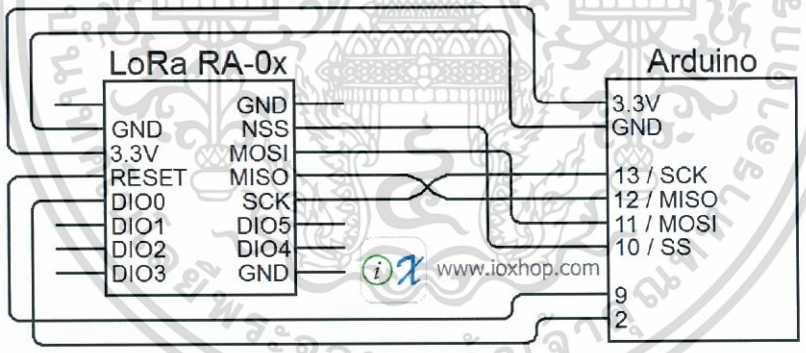
2.4.2.5 LoRa Localization

LoRa มีมาตรฐานในการระบุตำแหน่ง โดยคำนวณจากสัญญาณและระยะเวลาที่ packet รับส่งกับ gateway ซึ่งเป็นหลักการเดียวกันกับการ Localization ทั่วๆไป ยิ่งมีจำนวน gateway เยอะๆก็จะระบุตำแหน่งได้แม่นยำมากขึ้น และประหยัดกว่าการใช้ GPS มาก รวมถึงการควบคุมการ Sleep ให้ประหยัดพลังงาน

2.4.2.6 การต่อวงจรใช้งานโมดูลกลุ่ม LoRa

โมดูลกลุ่ม LoRa จะมีขาที่ใช้ต่อใช้งานจริง ๆ จำนวน 8 ขา แบ่งเป็นขาบัส SPI ที่จำเป็นต้องต่อเข้ากับไมโครคอนโทรลเลอร์ด้วยขาที่กำหนดเท่านั้นและอีก 3 ขา คือ NSS NRESET และ DIO0 สามารถต่อเข้ากับขาใดก็ได้ และต้องเรียกใช้ฟังก์ชั่นเพื่อกำหนดขาที่ต่อให้ถูกต้องด้วย

2.4.2.7 วงจรต่อใช้งานโมดูลกลุ่ม LoRa เข้ากับ Arduino Nano



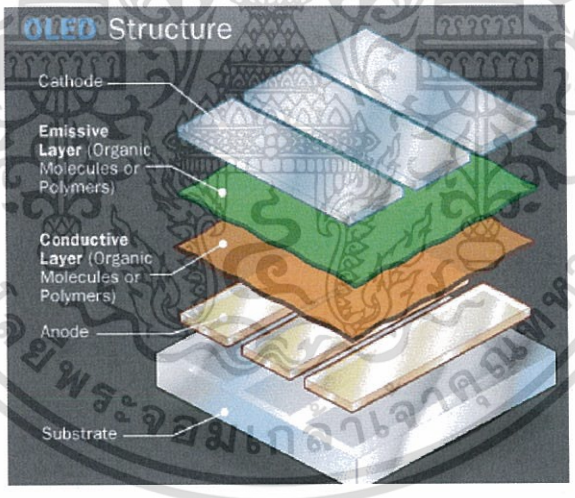
รูปที่ 2.14 วงจรการเชื่อมต่อ LoRa เข้ากับ Arduino Nano [7]

2.4.3 OLED (Organic Light Emitting Diodes)

OLED (Organic Light Emitting Diodes) คือจอภาพที่มีลักษณะคล้ายแผ่นฟิล์ม ซึ่งมีส่วนประกอบเป็นสารอินทรีย์ที่สามารถเปล่งแสงเองได้เมื่อได้รับพลังงานไฟฟ้า เรียกว่ากระบวนการอิเล็คโทรลูมิเนเซนส์ (Electroluminescence) โดยที่ไม่ต้องพึ่งพาแสง Backlight และจะไม่มีการเปล่งแสดงในบริเวณที่เป็นภาพสีดำ ส่งผลให้สีดำนั้นดำสนิท อีกทั้งยังช่วยพลังงานด้วย

นอกจากนี้ จอภาพแบบ OLED ยังมีความบางกว่า LCD รวมทั้งมีความยืดหยุ่น สามารถโค้งงอได้ เนื่องจาก OLED มีโครงสร้างที่แตกต่างจาก LCD โดยโครงสร้างของ OLED นั้นประกอบด้วยสารกึ่งตัวนำไฟฟ้าที่เป็นของแข็ง ทำจากวัสดุอินทรีย์มีทั้งแบบ Polymer และโมเลกุลขนาดเล็ก ซึ่งมีความหนาเพียง 100-500 นาโนเมตรเท่านั้น (บางกว่าเส้นผมของคน 200 เท่า) และอาจมีชั้นสารอินทรีย์เป็นองค์ประกอบ 2 ถึง 3 ชั้น

2.4.3.1 รายละเอียดโครงสร้างของ OLED



รูปที่ 2.15 รายละเอียดโครงสร้างแต่ละชั้นของ OLED [8]

1. Substrate เป็นชั้นผิวหน้าของจอภาพ อาจทำจากกระจก, ฟลอยด์โลหะ หรือพลาสติกใส ซึ่งหากทำจากฟลอยด์หรือพลาสติกใสจะทำให้ได้จอภาพที่มีความยืดหยุ่นสูง
2. Anode (ขั้วบวก) ทำด้วยวัสดุโปร่งใส (Indium Tinn Oxide ; ITO) เป็นตัวทำหน้าที่ดึงกระแสอิเล็กตรอน

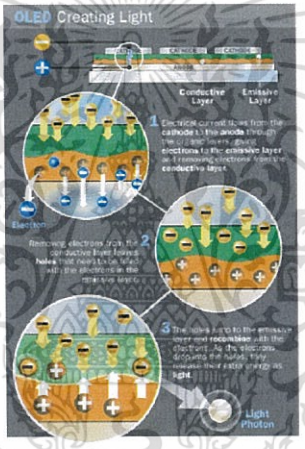
3. Organic Layer ทำจากสารประกอบอินทรีย์ หรือโพลิเมอร์ของสารอินทรีย์ โดยถูกแบ่งออกเป็น

- Conducting Layer ทำจากโมเลกุลของสารอินทรีย์ที่เป็นสี ทำหน้าที่ส่ง Hole ของอิเล็คตรอนจาก Anode

- Emissive Layer ทำจากโมเลกุลของสารอินทรีย์ที่เป็นสี ทำหน้าที่เคลื่อนย้ายอิเล็คตรอนจาก Cathode โดยชั้นนี้เป็นชั้นที่ทำให้เกิดการเปล่งแสง

4. Cathode (ขั้วลบ) อาจทำด้วยวัสดุโปร่งใสหรือไม่ก็ได้ ขึ้นอยู่กับชนิดของ OLED เป็นตัวทำหน้าที่ปล่อยกระแสอีเล็กตรอน

2.4.3.2 หลักการทำงานของกระบวนการอิเล็คโทรลูมิเนเซนส์



รูปที่ 2.16 หลักการทำงานกระบวนการอิเล็คโทรลูมิเนเซนส์ [8]

1. จอ OLED ได้รับกระแสไฟฟ้าจากแหล่งพลังงานหรือแบตเตอรี่

2. กระแสไฟฟ้าไหลจาก Cathode ผ่านชั้นสารอินทรีย์ไปยัง Anode โดย Cathode จะส่งอิเล็กตรอนให้ Emissive Layer

3. Anode ดึงอิเล็กตรอนจาก Conductive Layer ทำให้เกิด Electron Holes ขึ้น

4. ระหว่าง Emissive Layer และ Conductive Layer จะเกิดปฏิกิริยา Electron (-) รวมตัวเข้ากับ Hole (+) ขึ้น

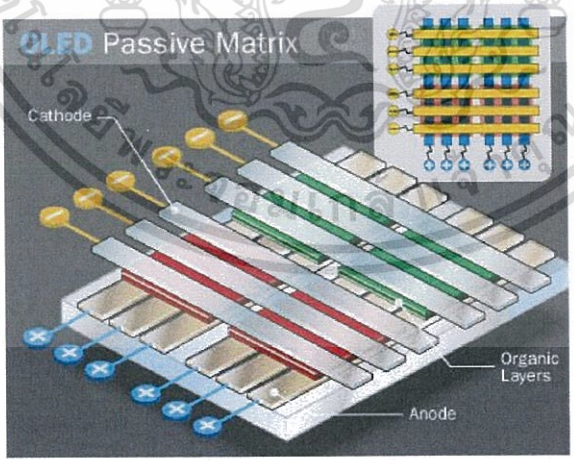
5. เนื่องจากอิเล็กตรอนมีระดับพลังงานที่สูงกว่า Holes จึงต้องลดระดับของพลังงานของอิเล็กตรอนลง ด้วยการเปลี่ยนรูปของพลังงานไปเป็นพลังงานแสงแทน

6. จอภาพ OLED เปล่งแสงจากพลังงานแสง

สำหรับสีของแสงที่ปรากฏออกมาจะขึ้นอยู่กับชนิดของโมเลกุลของสารอินทรีย์ในชั้น Emissive layer ซึ่งในจอ Full Color OLED จะมีสารอินทรีย์ทั้งหมด 3 ชนิด ได้แก่ สารอินทรีย์ที่ให้แสงสีแดง, เขียว และน้ำเงิน (RGB) โดยสารทั้ง 3 ชนิดนี้ถูกเคลือบอยู่บน OLED เพียงแผ่นเดียว เพื่อให้เกิดสีสันต่าง ๆ ส่วนความสว่างของแสงที่ปรากฏบนจอภาพจะขึ้นอยู่กับปริมาณของกระแสอิเล็กตรอน หากมีกระแสมากแสงก็จะมีความสว่างมากขึ้น ซึ่งปกติ OLED จะใช้กระแสไฟฟ้าที่ประมาณ 3-10 โวลต์

2.4.3.3 ประเภทของ OLED

PMOLED (Passive Matrix OLED) ในแต่ละชั้นจะมีลักษณะเป็นแถบแยกออกจากกัน โดยชั้นของ Cathode และ Anode จะวางในแนวขวางซึ่งกันและกัน โดยเมื่อมีกระแสไหลผ่านในแต่ละช่อง Cathode และ Anode จะทำให้ฟิล์มเกิดการเปล่งแสงบริเวณที่ขั้ว Cathode และ Anode วางตัดกัน ดังนั้นการควบคุมการเปล่งแสงของ PMOLED จึงขึ้นอยู่กับการเลือกช่องทางเดินของกระแส โดยข้อดีของ OLED ชนิดนี้คือสร้างได้ง่าย และต้องการกระแสจากวงจรภายนอก ส่งผลให้ต้องใช้พลังงานมากกว่า OLED ชนิดอื่น ๆ (แต่ก็ยังประหยัดพลังงานมากกว่า LCD) ซึ่ง PMOLED เหมาะสำหรับทำจอภาพขนาดเล็กที่มีความกว้างประมาณ 2-3 นิ้ว อย่างเช่น จอของโทรศัพท์หรืออุปกรณ์พกพาต่าง ๆ แต่ปัจจุบันนิยมหันใช้ AMOLED กันมากกว่าแล้ว

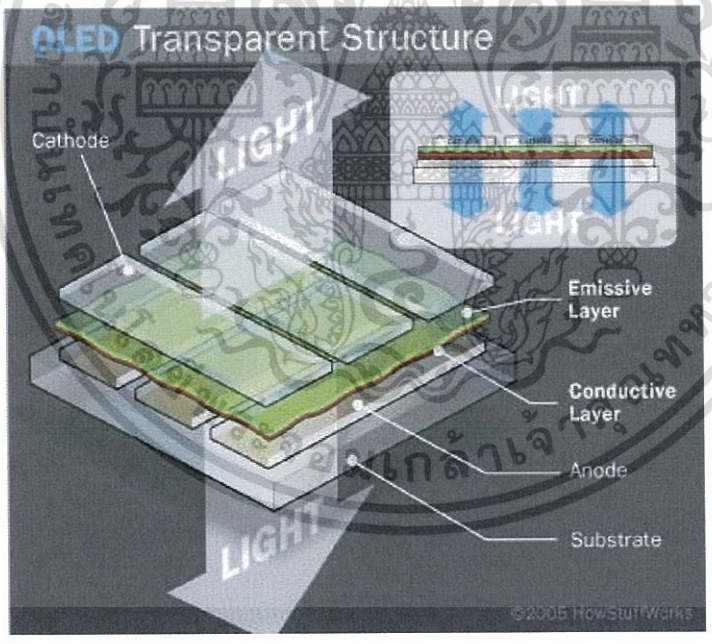


รูปที่ 2.17 โครงสร้าง OLED แบบ Passive Matrix [8]

AMOLED (Active Matrix OLED) ในแต่ละชั้นจะต่อเนื่องกันทั้งชั้น แต่ในชั้น Anode จะมีลักษณะเป็นฟิล์มบางที่เป็นวงจรในตัวเอง และควบคุมการเกิดภาพได้เอง โดย AMOLED จะใช้พลังงานน้อยกว่า PMOLED เนื่องจากลักษณะโครงสร้างที่เป็นแบบฟิล์มบาง และยังสามารถขยายให้มีขนาดใหญ่ได้ด้วย จึงทำให้ AMOLED เหมาะสำหรับทำจอภาพที่มีขนาดใหญ่ เช่น จอโทรทัศน์, จอคอมพิวเตอร์ หรือจอป้ายโฆษณาขนาดใหญ่ แต่ปัจจุบันก็นิยมใช้เป็นจอภาพของอุปกรณ์พกพาอื่นๆเช่นกัน

OLED ยังแบ่งเป็นประเภทย่อย ๆ ตามลักษณะการใช้งานได้อีก 4 ชนิด

1. Transparent OLED ประกอบด้วยชั้นที่โปร่งแสงทุกชั้น เมื่อปิดจอ แสงจากภายนอกจะสามารถผ่านจอได้ถึง 85 % และเมื่อเปิดจอกระแสไฟฟ้าก็จะถูกส่งเข้าระบบ และเปลี่ยนเป็นแสงส่องผ่านออกมาจากจอได้ทั้งสองด้าน ซึ่งสามารถสร้างจอภาพที่มองเห็นภาพได้ทั้ง 2 ด้าน โดย Transparent OLED นี้สามารถสร้างได้ทั้งแบบ PMOLED และ AMOLED



รูปที่ 2.18 โครงสร้าง OLED Transparent [8]

2. Top-emitting OLED เป็นจอแบบทึบแสงหรือสะท้อนแสง โดยจอภาพแบบนี้จะเป็นแบบ AMOLED ซึ่งถูกนำไปใช้กับ Smartcard เป็นส่วนใหญ่

3. Foldable OLED ทำด้วยวัสดุที่มีความยืดหยุ่นสูง เช่น แผ่นฟลอยด์โลหะหรือพลาสติกใส มีน้ำหนักเบา และมีความทนทานสูง เหมาะใช้สำหรับโทรศัพท์ หรืออุปกรณ์พกพาต่าง ๆ เพื่อช่วยลดปัญหาหน้าจอแตก นอกจากนี้ ยังสามารถเย็บติดกับเส้นใยผ้าต่าง ๆ อย่างเสื้อผ้าได้อีกด้วย

4. White OLED เป็น OLED ที่ให้แสงสีขาว ช่วยประหยัดพลังงานและมีคุณภาพดีกว่าแสงที่ได้จากหลอดฟลูออเรสเซนต์ ทำให้เห็นสีแท้จริง เช่นเดียวกันแสงสว่างตามธรรมชาติ และมีแนวโน้มว่าเมื่อทำให้มีขนาดใหญ่ จะสามารถใช้แทนแสงฟลูออเรสเซนต์ที่ใช้ตามบ้านและตึกต่าง ๆ ได้ ซึ่งจะช่วยประหยัดพลังงานมากกว่าการใช้หลอดไฟธรรมดา

2.4.3.4 ข้อดีและข้อเสียของ OLED

ข้อดีของ OLED	**ข้อเสียของ OLED**
บาง เบา และมีความยืดหยุ่นสูง	ฟิล์มที่ให้กำเนิดสีน้ำเงินมีอายุการใช้งานสั้นเพียง 1,000 ชั่วโมง (แต่สำหรับสีแดง และเขียว มีอายุการใช้งานที่ยาวนานถึงประมาณ 10,000-40,000 ชั่วโมง
เมื่อนำพลาสติกมาทำจอของ OLED แทนกระจก จะทำให้จอมีความยืดหยุ่นสูง	สารอินทรีย์ที่ใช้ทำ OLED จะเสียหายได้ง่ายเมื่อโดนน้ำหรือออกซิเจน
สามารถทำเป็นจอแบบโปร่งใส และมองเห็นได้จากทั้งสองด้าน	
ให้ความสว่างได้มากกว่าจอปกติ	
สีดำ ดำสนิทกว่าจอปกติ เนื่องจากไม่มีแสง Backlight เนื่องจากสามารถเปล่งแสงได้ด้วยตัวเอง	
สามารถสร้างให้เป็นขนาดใหญ่ได้ง่าย โดยมีความปลอดภัยสูง เพราะสามารถสร้างจากพลาสติก	

ตารางที่ 2.3 ข้อดีและข้อเสียของ OLED [8]

2.4.4 Passive Buzzer Module

Passive Buzzer Module เป็นอุปกรณ์ทำหน้าที่คล้ายลำโพง เพียงแต่ออกแบบมาเพื่อใช้ในการส่งสัญญาณเสียงความถี่สูงๆ บนบอร์ดมีทรานซิสเตอร์สำหรับช่วยขับมาให้แล้ว สามารถต่อใช้งานกับไมโครคอนโทรลเลอร์ได้โดยตรง การใช้งานเพียงเขียนโค้ดสร้างสัญญาณ HIGH LOW สลับกันไปมา หรือใน Arduino สามารถใช้ฟังก์ชั่น tone() ได้เลย ข้อดีของ Buzzer แบบพาสซีฟคือสามารถกำหนดความถี่เสียงที่ต้องการได้เอง

2.4.4.1 รายละเอียด Buzzer

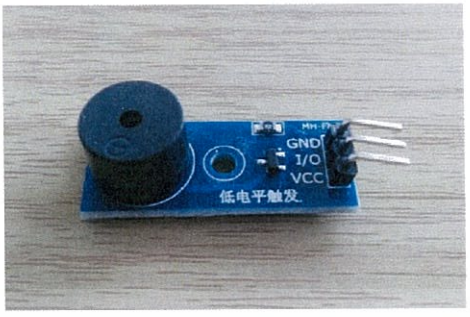
Parameter Item	Parameter Details & Description
Size	3.3 cm*1.3 cm
Supply Voltage	3.3 V to 5 V
Power Amplify	Active transistor on module
Implement	Easy to install screw

ตารางที่ 2.4 รายละเอียด Buzzer [9]

2.4.4.2 Pins Definition

Pins	Details & Description
VCC	3.3 V to 5 V
GND	Ground
I/O	I/O interface of SCM

ตารางที่ 2.5 Pins ของ Buzzer [9]



รูปที่ 2.19 อุปกรณ์ Buzzer [9]

ตัวอย่าง Code

```
int buzzer = 13;
void setup() {
pinMode(buzzer, OUTPUT);
}
void loop() {
digitalWrite(buzzer, HIGH);
delay(1000);
digitalWrite(buzzer, LOW);
delay(1000);
}
```

รูปที่ 2.20 โค้ดคำสั่งให้ Buzzer ทำงานติดดับ [9]

2.4.4.3 ผลลัพธ์

Buzzer จะมีเสียงเบา สลับ กับเสียงดัง ห่างกัน 1 วินาที

2.4.5 วงจรรักษาระดับแรงดัน

วงจรที่จะต่ออยู่ระหว่างแหล่งจ่ายไฟกระแสตรง กับ โหลด ทำหน้าที่ในการจ่ายไฟตรงให้กับโหลดและสามารถรักษาระดับแรงดันให้คงที่ได้ หรือ ยกตัวอย่างง่ายๆก็คือ สมมุติว่าเรามีแหล่งจ่ายอยู่ 12 โวลต์ แต่ว่าวงจรของเราต้องการไฟเพียง 10 โวลต์ เราก็สามารถนำตัววงจรรักษาระดับแรงดันนี้มาใช้ต่อเพื่อจ่ายให้กับวงจรของเราได้นั่นเอง

2.4.5.1 วงจรรักษาระดับแรงดัน จากซีเนอร์ไดโอด

ซีเนอร์ไดโอด คือไดโอดชนิดพิเศษที่สร้างขึ้นมา ให้มีการทำงานที่แตกต่างไปจาก ไดโอดเรียงกระแสแบบทั่วไป สรุปคือ เมื่อมีการจ่ายไบอัสให้กับตัวซีเนอร์ไดโอดจะสามารถทำงานเรียงกระแสได้เหมือนกับไดโอดปกติ แต่เมื่อมีการไบอัสกลับให้กับซีเนอร์ไดโอด จนถึงค่าแรงดันที่กำหนด ซีเนอร์ไดโอดจะนำกระแสและเกิด แรงดันตกคร่อมตัวมันเองเท่ากับค่าแรงดันที่กำหนดที่ตัวซีเนอร์

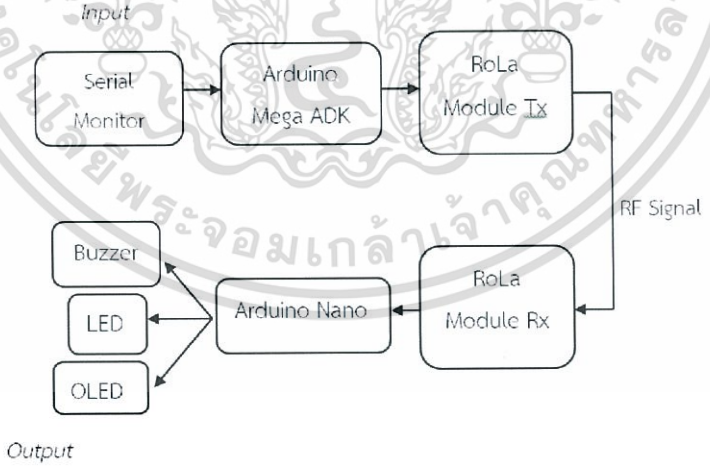
บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3. การออกแบบ

ในการออกแบบระบบจัดการคิวสำหรับร้านอาหาร ทางผู้จัดทำได้ศึกษากระบวนการทำงานของ อุปกรณ์ต่างๆที่นำมาใช้งานร่วมกัน ได้แก่ Arduino Mega ADK board, Arduino Nano 328, LoRa Module, OLED, Buzzer Module, LED Module โดยในฝั่งส่งจะใช้ไมโครคอนโทรลเลอร์ Arduino mega ADK ซึ่งจะมีการป้อนอินพุตผ่านทาง Serial Monitor จากหน้าจอสั่งการคอมพิวเตอร์ ระบุหมายเลขเครื่องรับที่ต้องการจะแจ้งเตือน บอร์ด Arduino mega ADK จะส่งสัญญาณที่มีโค้ดหมายเลขของเครื่องรับนั้นออกไปผ่าน Lora module เมื่อเครื่องรับ (Arduino Nano 328) ได้รับโค้ดหมายเลขดังกล่าว จะทำการตรวจสอบว่าตรงกับโค้ดของเครื่องที่ตั้งไว้หรือไม่ ถ้าโค้ดที่ส่งมาตรงกับโค้ดของเครื่องรับที่ตั้งไว้ บอร์ด Arduino จะส่งสัญญาณไฟกระพริบไปที่หลอดไฟ LED และ Buzzer ให้ทำการแจ้งเตือน

3.1 บล็อกไดอะแกรมของโครงงาน

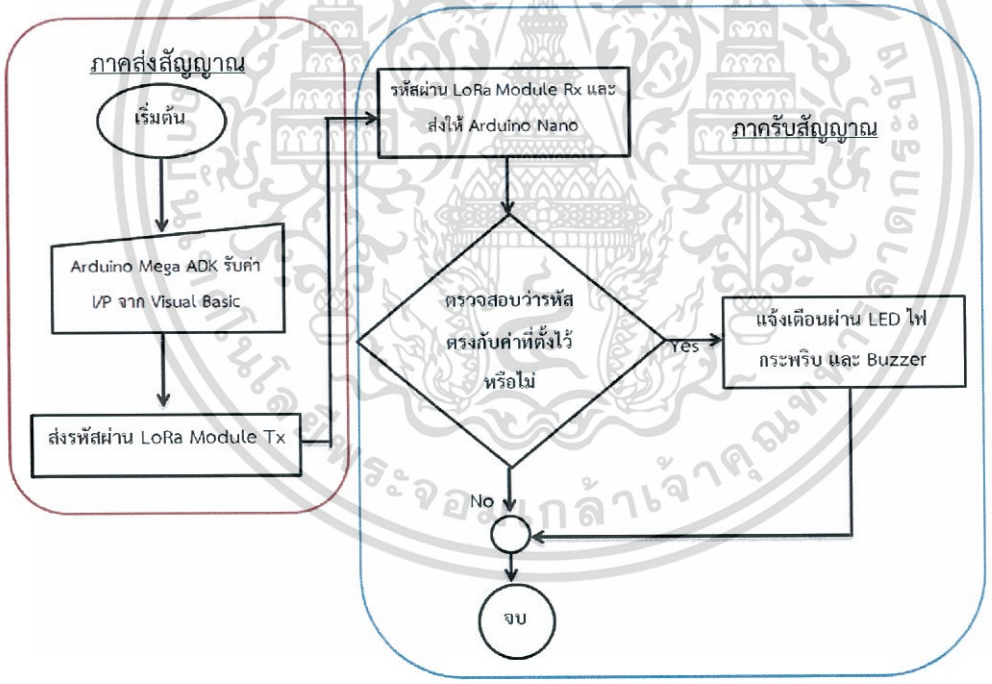


รูปที่ 3.1 บล็อกไดอะแกรมของโครงงาน

บล็อกไดอะแกรมของโครงงาน โดยสร้างระบบจัดการคิวด้วยด้วย Microcontroller ชนิด Arduino เพื่อทำระบบจัดการคิวโดยภาคส่งสัญญาณจะรับค่าผ่านทาง Serial monitor จากหน้าจอสั่งการคอมพิวเตอร์ ระบุหมายเลขเครื่องรับที่ต้องการจะแจ้งเตือน LoRa Module ภาครับสัญญาณที่ได้รับรหัสจะทำการตรวจสอบค่าว่าตรงกับค่าที่ตั้งไว้กับ LoRa Module เครื่องรับหรือไม่ ถ้าตรงกันจะแจ้งเตือนผ่าน LED Buzzer และแสดงผลข้อความผ่านหน้าจอ OLED

3.2 โฟลว์ชาร์ตการทำงานของโครงงาน

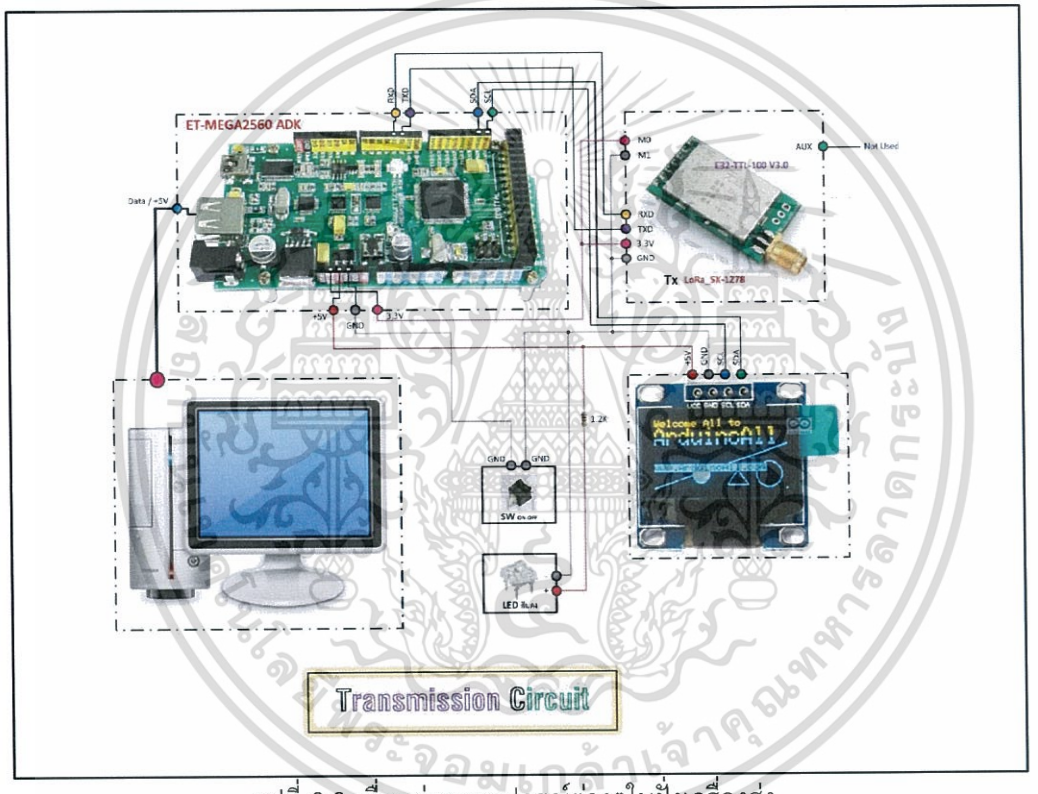
ผังงานการทำงานแสดงขั้นตอนการทำงานของวงจรโดยสรุป เพื่อให้เห็นภาพรวมของการทำงาน โดยผังงานการทำงานจะแสดงดังรูปที่ 3.2



รูปที่ 3.2 Flow Chart การทำงานระบบจัดการคิว

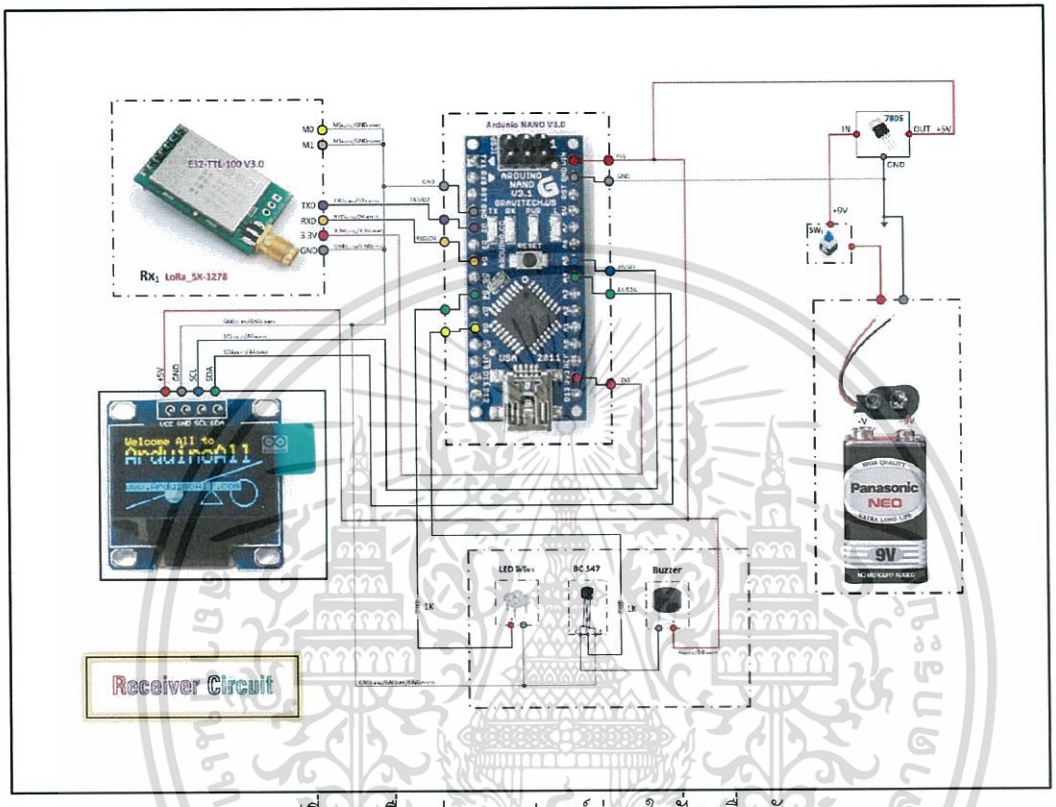
Flow Chart การทำงานระบบจัดการคิว จะเริ่มจากเมื่อเราทำการรับค่าอินพุตและส่งสัญญาณผ่าน LoRa Module ภาครับสัญญาณที่รับผ่าน LoRa Module ฝั่งรับจะทำการตัดสินใจว่าสัญญาณรหัสที่ได้รับมาตรงกับที่ตั้งค่าไว้เริ่มแรกหรือไม่ ถ้าตรงกันจะแจ้งเตือนให้ LED ไฟกระพริบและ Buzzer ทำงาน ถ้ารหัสไม่ตรงจะไม่มีการแจ้งเตือนเกิดขึ้น

3.3 การออกแบบเครื่องส่ง-รับสัญญาณ



รูปที่ 3.3 เชื่อมต่อของอุปกรณ์ต่างๆในฝั่งเครื่องส่ง

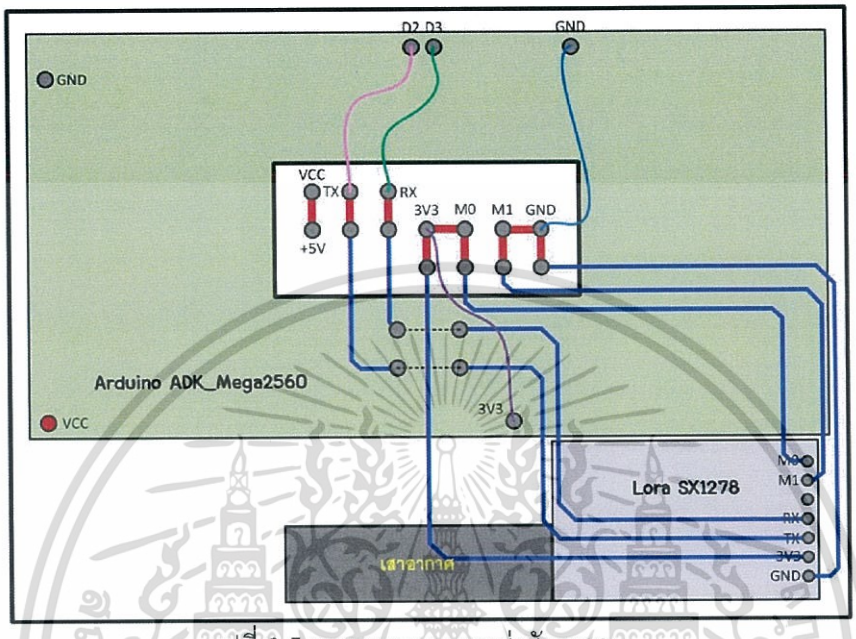
แผนภาพการเชื่อมต่อของอุปกรณ์ต่างๆในฝั่งเครื่องส่งใน รูปที่ 3.3 บอร์ด Arduino mega 2560 เชื่อมต่อและรับไฟเลี้ยงจากคอมพิวเตอร์ผ่านพอร์ต USB ส่วนอุปกรณ์ LoRa Module เชื่อมต่อกับบอร์ด Arduino Mega 2560 ผ่านพอร์ต Tx และ Rx หน้าจอ OLED Module เชื่อมต่อกับบอร์ด Arduino Mega 2560 ผ่านพอร์ต SCL และ SDA โดยอุปกรณ์ทั้งหมดในฝั่งเครื่องส่งจะทำงานผ่านการเปิดปิดของสวิตช์



รูปที่ 3.4 เชื่อมต่อของอุปกรณ์ต่างๆในฝั่งเครื่องรับ

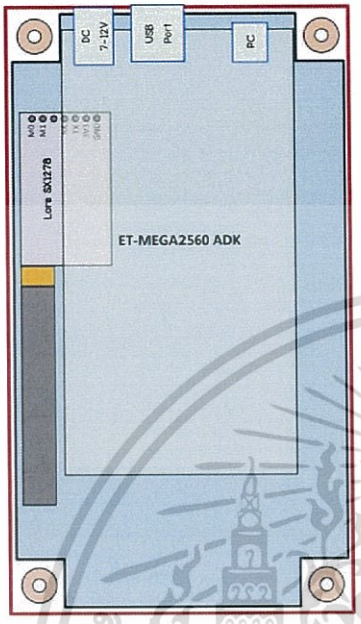
แผนภาพการเชื่อมต่อของอุปกรณ์ต่างๆในฝั่งเครื่องรับใน รูปที่ 3.4 บอร์ด Arduino Nano รับไฟเลี้ยงจากถ่าน 9 V. ส่วนอุปกรณ์ LoRa Module เชื่อมต่อกับบอร์ด Arduino Nano ผ่านพอร์ต Tx และ Rx หน้าจอ OLED Module เชื่อมต่อกับบอร์ด Arduino Mega 2560 ผ่าน Pin SCL และ SDA โดยอุปกรณ์ทั้งหมดในฝั่งเครื่องส่งจะทำงานผ่านการเปิดปิดของสวิตช์ และแจ้งเตือนด้วยเสียงผ่าน Buzzer

3.3.1 การออกแบบภาคส่งสัญญาณ

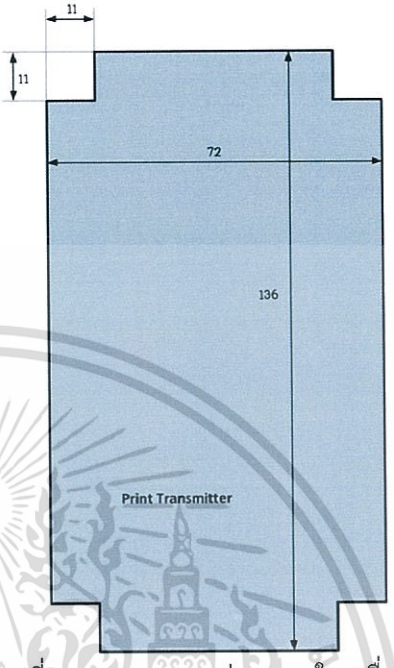


รูปที่ 3.5 การออกแบบภาคส่งสัญญาณ

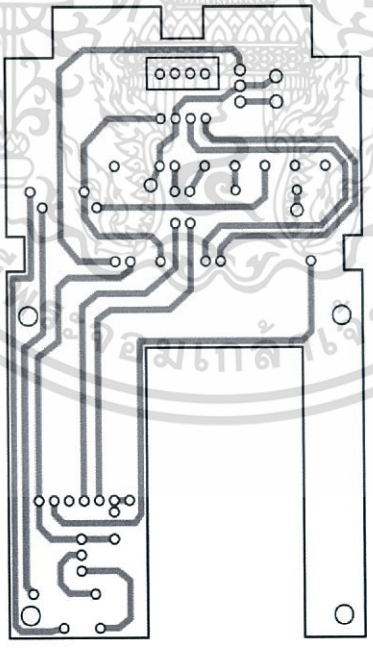
การออกแบบภาคส่งสัญญาณใน รูปที่ 3.5 Arduino Mega ADK ต่อเข้ากับ LoRa Module ที่เสียบเสาอากาศเรียบร้อยแล้ว บนบอร์ด Arduino Mega ADK จะมี 2 port คือ D2 และ D3 เชื่อมต่อกับฝั่งส่งและฝั่งรับของ LoRa จากนั้นส่งข้อมูลแบบ UART port M0 และ M1 บนบอร์ดของ LoRa ที่ทำหน้าที่ในการกำหนดโหมดในการทำงาน ซึ่งถูกกำหนดไว้ให้เป็น Wake-up Mode เพื่อเตรียมพร้อมใช้งานในการส่งสัญญาณให้ภาครับสัญญาณ



รูปที่ 3.6 การจัดวางตำแน่งของอุปกรณ์ในเครื่องส่ง

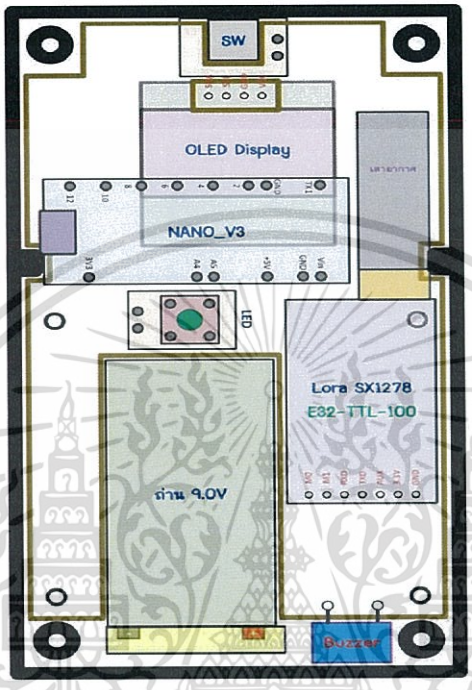


รูปที่ 3.7 ขนาดของแผ่น PCB ในเครื่องส่ง



รูปที่ 3.8 ลายวงจรแผ่น PCB เครื่องส่งสัญญาณ

3.3.2 การออกแบบภาครับสัญญาณ



รูปที่ 3.9 การออกแบบเครื่องรับสัญญาณ

การออกแบบเครื่องรับสัญญาณในรูปที่ 3.9 โดยไฟเลี้ยง 9V ที่จ่ายเข้าสู่บอร์ดจะนำไปผ่าน Voltage Regulator L7805 เพื่อควบคุมระดับแรงดันให้คงที่ จากนั้นก็จะจ่ายไฟเข้าสู่บอร์ด Arduino Nano เมื่อ LoRa Module ได้รับสัญญาณจะส่งต่อไปให้บอร์ด Arduino Nano ผ่านพอร์ต D2 และ D4 แบบ UART จากนั้นบอร์ด Arduino จะทำการตรวจสอบโค้ดที่ได้รับมาว่าตรงกับหมายเลขเครื่องหรือไม่ ถ้าตรงกันจะส่งเอาต์พุต เป็นไฟ 5V ไปที่ Transistor เพื่อให้ Transistor สร้างแรงดันไปขับ OLED, Buzzer และหลอดไฟ LED เป็นการแจ้งเตือนแก่ผู้ใช้งาน

รูปที่ 3.10 อุปกรณ์ภายในเครื่องส่งสัญญาณ

รูปที่ 3.11 อุปกรณ์ภายในเครื่องรับสัญญาณ

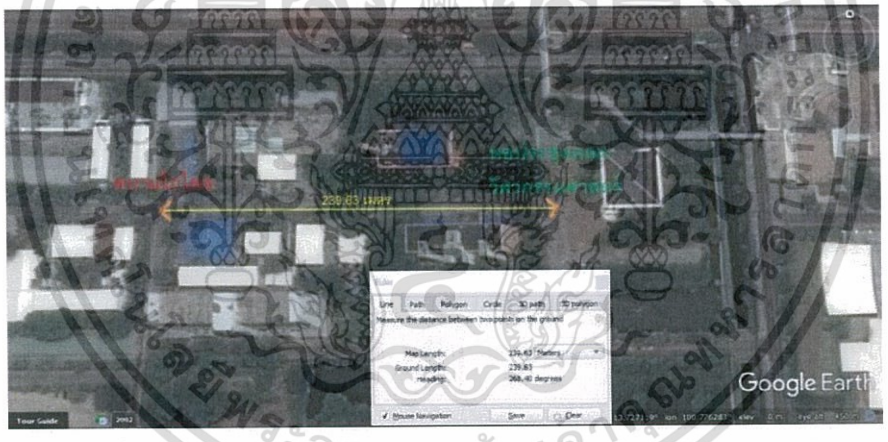
บทที่ 4

ผลการทดลอง

4.1 การจัดเก็บผลการทดลอง

การจัดเก็บผลการทดลองในขั้นตอนแรกได้ทำการทดลองการส่งและรับสัญญาณเพื่อตรวจสอบการทำงานของ LoRa Module และ Arduino Mega ADK และ Arduino Nano ว่าสามารถส่งและรับสัญญาณได้

ทดสอบการส่งสัญญาณและรับสัญญาณคลื่นวิทยุหลังจากการเขียนโปรแกรมทดสอบการทำงาน โดยการตรวจสอบความถูกต้องและความสามารถในการรับและส่งสัญญาณว่าเมื่อระยะห่างที่มากขึ้น มีความแม่นยำในการส่งที่ระยะห่างเท่าใด โดยที่จะเริ่มจากการส่งในระยะที่ไม่มีสิ่งกีดขวาง



รูปที่ 4.1 ระยะทางสนามโภไคยถึงหอประชุมคณะวิศวกรรมศาสตร์

ผลการทดสอบการส่งและรับสัญญาณ สถานที่คือสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง หน้าหอประชุมคณะวิศวกรรมศาสตร์ ผ่านโรงอาหารเอ ถึงสนามกีฬาโภไคย เป็นระยะทางทั้งสิ้น 239.63 เมตร ผลปรากฏว่า เริ่มการส่งสัญญาณที่หน้าหอประชุมคณะวิศวกรรมศาสตร์ ถึงโรงอาหารเอ มีความถูกต้องแม่นยำและดีเลย์ที่น้อยในระยะทาง 143.77 เมตร

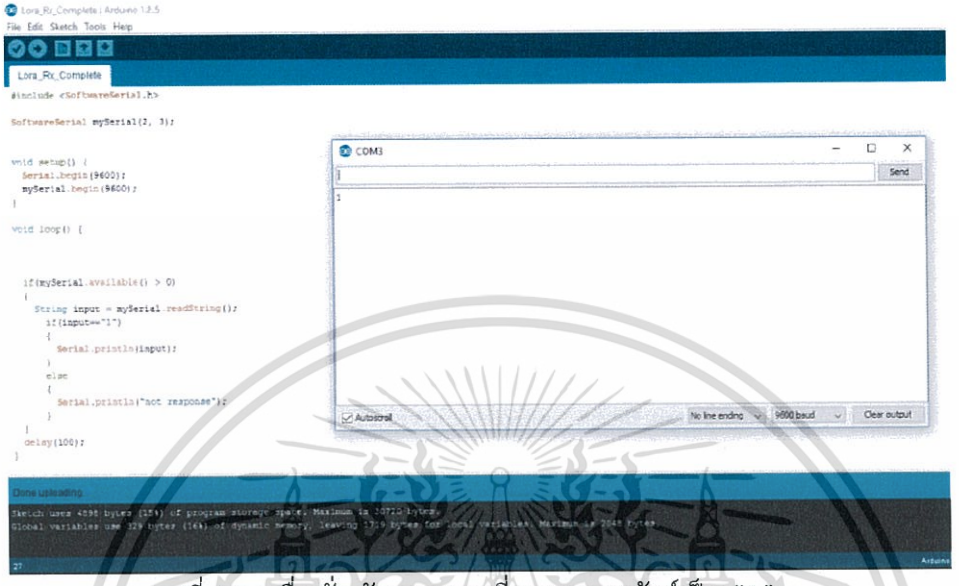
ดังรูปที่ 4.1 และจะเริ่มมีดีเลย์ที่มากขึ้น และอาจมีข้อมูลบางชุดที่สูญหายไป จนถึงระยะทาง 239.63 เมตร ดังรูปที่ 4.2 ข้อมูลที่รับมีการดีเลย์เป็นเวลานาน

รูปที่ 4.2 ระยะทางโรงอาหารเอถึงหอประชุมคณะวิศวกรรมศาสตร์

ทำการส่งข้อมูลสัญญาณคลื่นวิทยุเป็นตัวเลข และสังเกตุข้อมูลตัวเลขที่รับเข้ามา โดยส่งเริ่มที่เลข 1 เพิ่มขึ้นเรื่อยๆ ถ้าข้อมูลไม่มีสูญหายตัวเลขที่รับมาจะต่อเนื่องกัน ที่ระยะประมาณ 143.77 เมตรขึ้นไปข้อมูลเลขที่ฝั่งรับบางเลขจะสูญหายไปบางส่วนและมีดีเลย์เกิดขึ้นมาก ในขณะที่ระยะทางต่ำกว่าตัวเลขที่รับมามีดีเลย์ที่น้อยกว่าและต่อเนื่อง เกินระยะ 239.63 ข้อมูลตัวเลขที่ฝั่งรับไม่สามารถรับได้เนื่องจากกำลังส่งไม่เพียงพอ

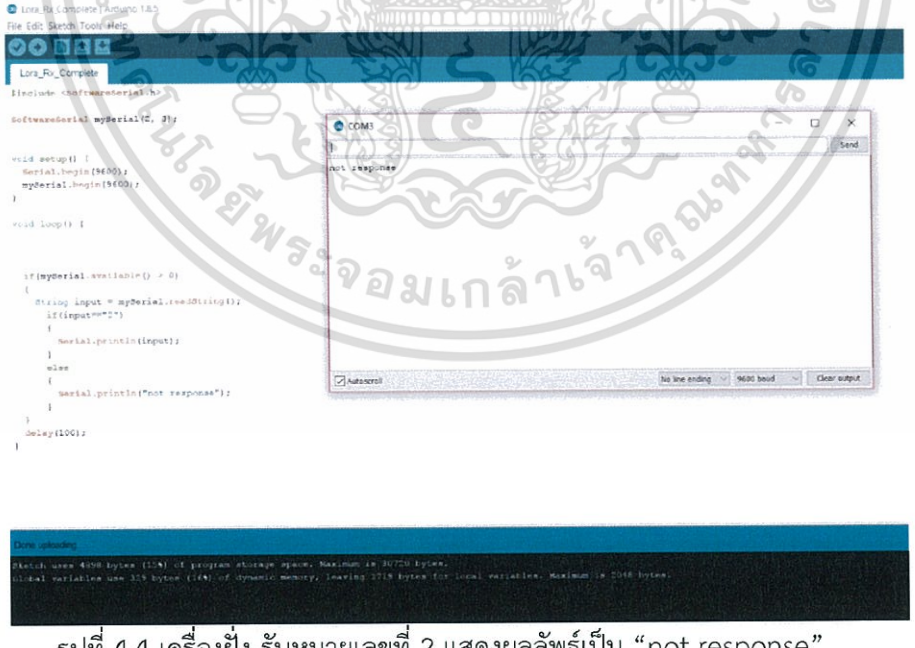
ทำการทดสอบส่งสัญญาณเครื่องส่ง 1 ตัว เครื่องรับหลายตัว โดยฝั่งส่งสัญญาณ RF จะส่งสัญญาณไปยังฝั่งรับเครื่องหมายเลข 1 และหมายเลข 2 หากต้องการส่งไปยังเครื่องฝั่งรับหมายเลขใดให้ใส่หมายเขนั้นลงหน้าจอมอนิเตอร์ ส่วนฝั่งรับที่หมายเลขตรงกับนั้นก็จะแสดงผลบนหน้าจอมอนิเตอร์เป็นเลขที่ส่งมา ส่วนเครื่องที่หมายเลขไม่ตรงก็จะแสดงผลลัพธ์เป็น not response

ทำการใส่ค่า “1” ทางจอมอนิเตอร์ของเครื่องส่ง



รูปที่ 4.3 เครื่องฝั่ง-รับหมายเลขที่ 1 แสดงผลลัพธ์เป็น “1”

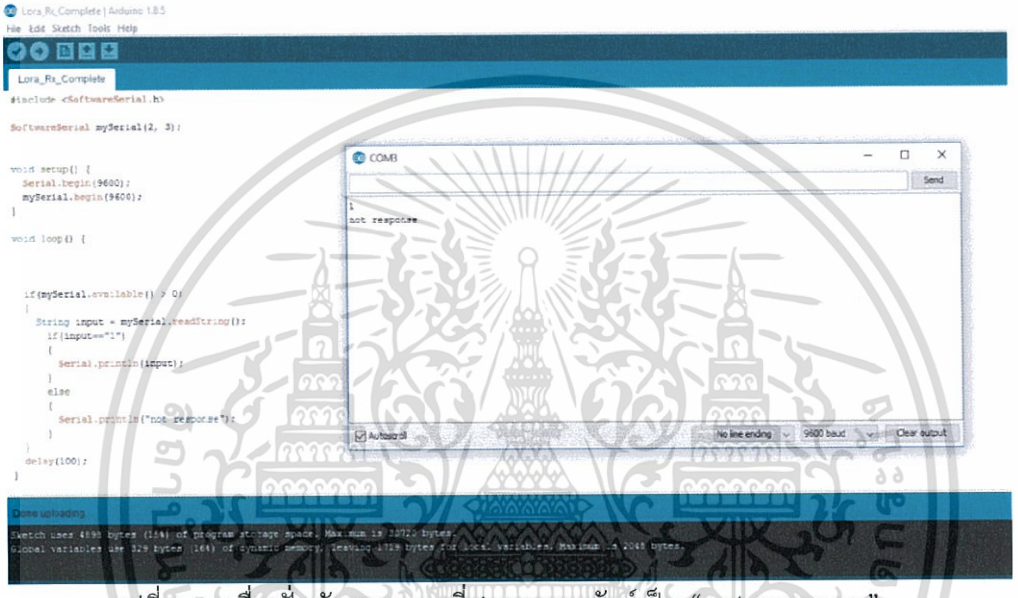
รูปที่ 4.3 แสดงหน้าจอ Serial Monitor ของโปรแกรม Arduino เครื่องรับหมายเลข 1 จะแสดงผลเป็น “1” ซึ่งตรงกับหมายเลขของฝั่งส่ง ดังรูปที่



รูปที่ 4.4 เครื่องฝั่ง-รับหมายเลขที่ 2 แสดงผลลัพธ์เป็น “not response”

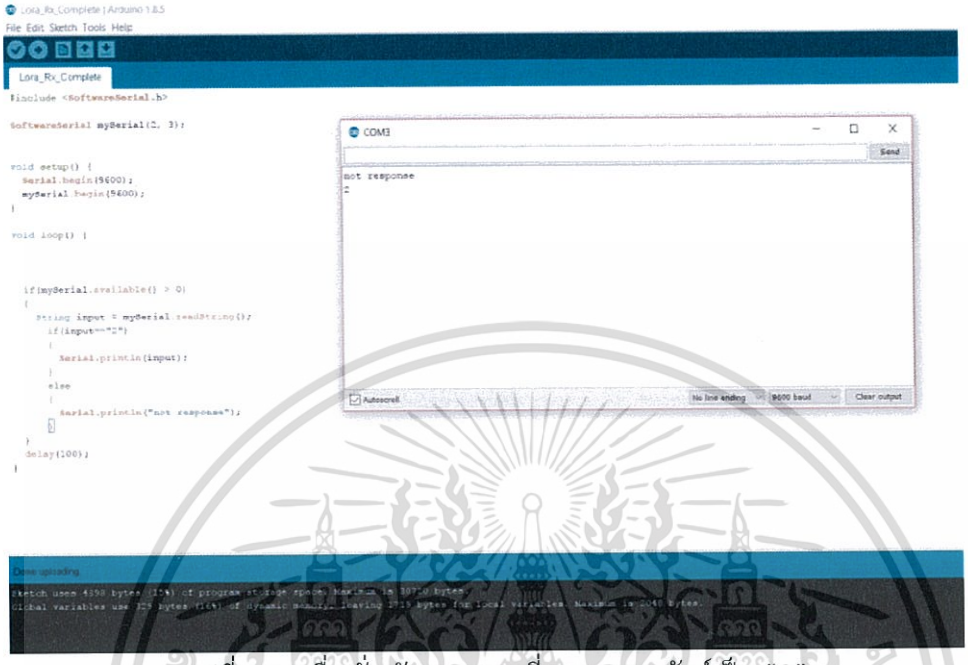
รูปที่ 4.4 แสดงหน้าจอ Serial Monitor ของโปรแกรม Arduino เครื่องรับหมายเลข 2 จะแสดงผลว่า “not response” เพราะรหัสที่ส่งมาเป็น 1 ซึ่งไม่ตรงกันกับหมายเลขฝั่งส่ง

ทดสอบเครื่องรับหมายเลข 2 เริ่มจากใส่ค่า “2” ทางจอมอนิเตอร์ของเครื่องส่งแล้วกด Enter



รูปที่ 4.5 เครื่องฝั่ง-รับหมายเลขที่ 1 แสดงผลลัพธ์เป็น “not response”

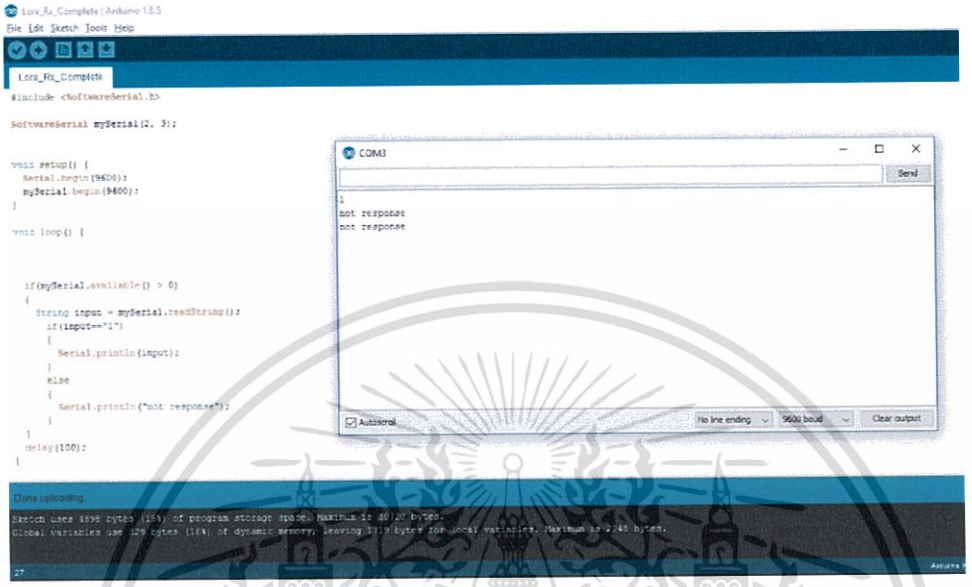
รูปที่ 4.5 แสดงหน้าจอ Serial Monitor ของโปรแกรม Arduino เครื่องรับหมายเลข 1 จะแสดงผลว่า “not response” เพราะรหัสที่ส่งมาเป็น 2 ซึ่งไม่ตรงกันกับหมายเลขของฝั่งส่ง



รูปที่ 4.6 เครื่องฝั่ง-รับหมายเลขที่ 2 แสดงผลลัพธ์เป็น “2”

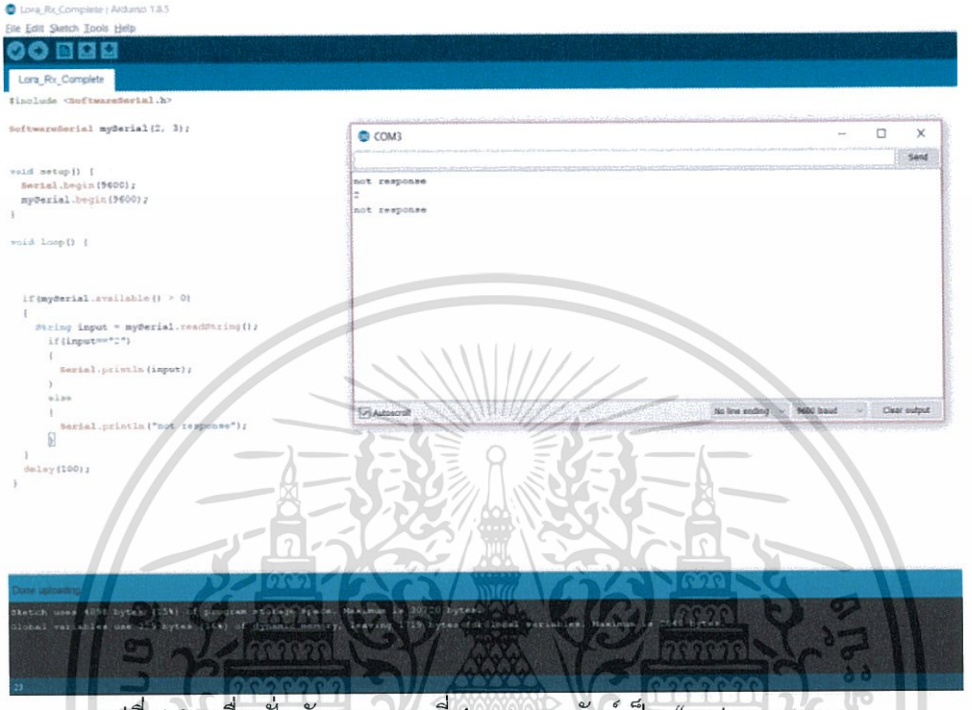
รูปที่ 4.6 แสดงหน้าจอ Serial Monitor ของโปรแกรม Arduino เครื่องรับหมายเลข 1 จะแสดงผลเป็น “1” ซึ่งหมายเลขตรงกับฝั่งส่ง

ทดสอบถ้าส่งรหัสตัวเลขอื่นที่ไม่ตรงกับเครื่องรับใดๆเลย เช่น “5” แล้วกด Enter



รูปที่ 4.7 เครื่องฝั่ง-รับหมายเลขที่ 2 แสดงผลลัพธ์เป็น “not response”

รูปที่ 4.7 แสดงหน้าจอ Serial Monitor ของโปรแกรม Arduino เครื่องรับหมายเลข 1 จะแสดงผลว่า “not response” เพราะรหัสที่ส่งมาเป็น 5 ซึ่งไม่ตรงกัน

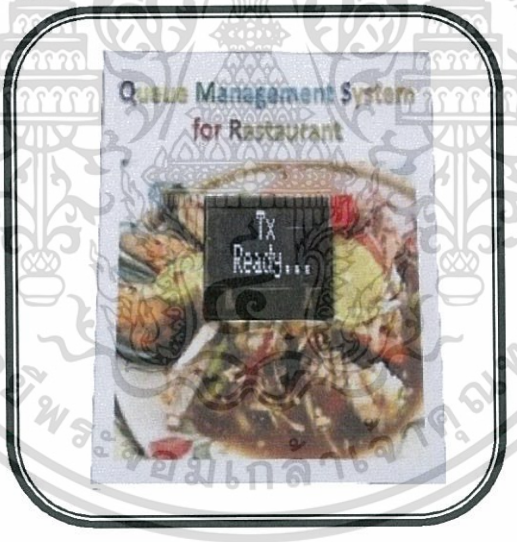


รูปที่ 4.8 เครื่องฝั่ง-รับหมายเลขที่ 1 แสดงผลลัพธ์เป็น “not response

รูปที่ 4.8 แสดงหน้าจอ Serial Monitor ของโปรแกรม Arduino เครื่องรับหมายเลข 2 จะแสดงผลว่า “not response” เพราะรหัสที่ส่งมาเป็น 5 ซึ่งไม่ตรงกันกับฝั่งส่ง

4.2 ผลการทดลอง

ผลการทดลองทำการ Run โปรแกรมเครื่องส่งแล้วทำการเปิด Serial Monitor จากนั้นใส่หมายเลขเครื่องรับ แล้วกด Enter เครื่องส่งจะทำการบรอดแคสสัญญาณออกไป เครื่องรับทุกตัว จะทำการตรวจสอบกับโค้ดที่ได้ตั้งค่าไว้ หากเครื่องรับได้รับสัญญาณหมายเลขตรงกับที่เครื่องฝั่งส่งได้ส่งมา เครื่องรับจึงจะทำการแจ้งเตือนโดยการส่งเสียงออกทาง Buzzer หลอดไฟ LED กระพริบเป็นจังหวะ และแสดงเป็นข้อความผ่านหน้าจอ OLED ในขณะที่เครื่องรับหมายเลขที่ไม่ใช่หมายเลขที่เครื่องส่งใส่อินพุต จะทำการตรวจสอบโค้ดสัญญาณที่ได้รับ และตัดสินใจว่าหมายเลขไม่ตรงกันจะไม่ทำการแจ้งเตือนเกิดขึ้น



รูปที่ 4.9 เครื่องส่งสัญญาณอยู่ในสถานะพร้อมใช้งาน

รูปที่ 4.10 เครื่องส่งสัญญาณที่ส่งค่ารหัสหมายเลข 1, 2, 3 และ 4

รูปที่ 4.10 แสดงข้อความหน้าจอ OLED ของเครื่องส่งสัญญาณ หากมีลูกค้ามาสั่งเมนูอาหาร ทางร้านจะให้เครื่องรับขนาดเล็กพกพาได้ไว้กับลูกค้า เพื่อการแจ้งเตือนลูกค้า เมื่ออาหารเสร็จพร้อมเสริฟ ทำให้ลูกค้าไม่จำเป็นต้องมาต่อเรียงคิวหน้าร้าน โดยที่เครื่องรับมี 4 เครื่อง 4 หมายเลข เมื่อทำการแจ้งเตือนฝั่งเครื่องส่งของทางร้านจะส่งสัญญาณคลื่นวิทยุ เป็นค่ารหัสหมายเลขเครื่องนั้นแสดงผลหน้าจอ OLED เป็นข้อความว่า “Sending Rx No. (หมายเลขของเครื่องรับ)”

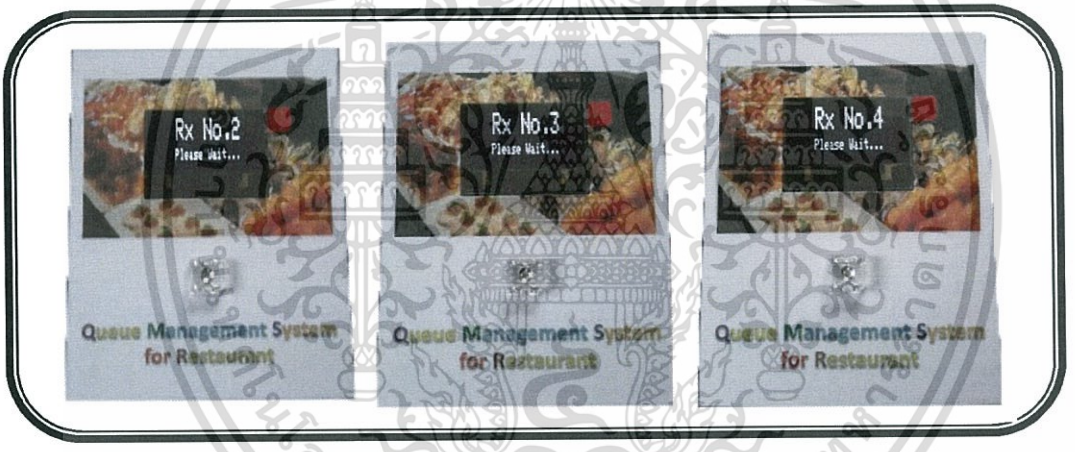
รูปที่ 4.11 เครื่องรับหมายเลข 1, 2, 3 และ 4 ที่ยังไม่ได้ใช้งานจองคิว

รูปที่ 4.11 แสดงข้อความหน้าจอ OLED ของเครื่องรับสัญญาณที่ยังไม่ได้ให้ลูกค้า เพื่อทำการรับการจองคิว จะแสดงข้อความขึ้นผ่านหน้าจอ OLED ว่า “Rx No. (หมายเลขของเครื่องรับ) Ready”

รูปที่ 4.12 เครื่องรับหมายเลข 1, 2, 3 และ 4 จองคิวแล้ว รอรับการแจ้งเตือน

รูปที่ 4.12 แสดงข้อความหน้าจอ OLED ของเครื่องรับสัญญาณ เมื่อทำการสั่งเมนูอาหารเรียบร้อยแล้ว จะได้เครื่องรับสัญญาณขนาดเล็กเพื่อรอรับการแจ้งเตือนว่าอาหารเสร็จ พร้อมเสริฟ ปรากฏเป็นข้อความผ่านหน้าจอ OLED ว่า “Rx No. (หมายเลขของเครื่องรับ) Please wait...”

รูปที่ 4.13 เครื่องรับที่ได้รับสัญญาณตรงกับเครื่องส่ง



รูปที่ 4.14 เครื่องรับที่ได้รับสัญญาณไม่ตรงกับเครื่องส่ง

รูปที่ 4.13 เมื่ออาหารทำเสร็จพร้อมเสริฟทางร้านจะป้อนอินพุตของหมายเลขเครื่องรับผ่านทางหน้าจอคอมพิวเตอร์เพื่อทำการแจ้งเตือนเครื่องรับสัญญาณหมายเลขนั้น เครื่องรับที่ 1 ได้รับสัญญาณตรงกับเครื่องส่งจะแสดงผล หลอดไฟ LED กระพริบ Buzzer ส่งเสียงแจ้งเตือน และแจ้งเตือนข้อความผ่าน OLED ว่า “Order is Completed” ส่วนเครื่องรับหมายเลขอื่นที่ได้รับสัญญาณไม่ตรงกับเครื่องส่งก็ยังคงขึ้นสถานะ “Please Wait” อาหารยังไม่พร้อมเสริฟ ดังรูปที่ 4.14

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

จากการทำปริญญานิพนธ์เรื่องระบบจัดการคิวสำหรับร้านอาหารโดยใช้ Microcontroller ชนิด Arduino Mega ADK เป็นภาคส่งคลื่นสัญญาณวิทยุ และ Arduino Nano เป็นภาครับคลื่นสัญญาณวิทยุ โดยสามารถส่งรหัสหมายเลขผ่านหน้าจอคอมพิวเตอร์ สัญญาณจะผ่าน LoRa Module ภาคส่ง โดยมี Arduino Mega ADK เป็นตัวควบคุมการส่งสัญญาณ และสัญญาณที่ได้ได้รับจาก LoRa Module และ Arduino Nano ทางภาครับจะทำการตรวจสอบสัญญาณคลื่นวิทยุที่ได้รับมาว่ามีหมายเลขเลขตรงกับทางภาครับหรือไม่ ถ้าหากตรงกันจะทำการแจ้งเตือนป็นข้อความผ่าน OLED จากนั้น หลอดไฟกระพริบ LED จะทำงาน และ Buzzer Module จะส่งเสียงแจ้งเตือนเป็นจังหวะ และถ้าหากหมายเลขทางภาครับที่ได้ไม่ตรงกันจะไม่มีการแจ้งเตือนเกิดขึ้น ซึ่งปริญญานิพนธ์นี้ประสบความสำเร็จตามวัตถุประสงค์และขอบเขตและสามารถนำไปใช้ในชีวิตประจำวันและสามารถนำไปดัดแปลงแก้ไขให้มี Option เพิ่มขึ้นอีกได้

5.2 ข้อเสนอแนะ

1) การส่งข้อมูลเมื่อเพิ่มคำสั่งลงใน Serial Monitor ควรเว้นระยะการกดส่งในแต่ละครั้ง เพื่อป้องกันข้อมูลถูกรวมกันเป็นคำสั่งเดียวกัน ทำให้การทำงานเกิดข้อมูลผิดพลาด

2) LoRa Module ที่ใช้มีกำลังส่งเพียง 100 mW ซึ่งอาจไม่เพียงพอในการส่งข้อมูลระยะไกลๆ และอาจมีความผิดพลาดเกิดขึ้นกับข้อมูลที่ส่งไป หรือข้อมูลสูญหาย จึงควรเลือกใช้ Transmitter ที่มีกำลังส่งมากขึ้น

บรรณานุกรม

[1] Arduino, https://www.ioxhop.com/article/5/arduino

[2] Arduino, https://www.arduino.cc/en/Reference/SoftwareSerial.

[3] Broadcasting, http://intimenetwork.blogspot.com/2016/05/broadcast-radio.html

[4] UART, http://www.thaieasyelec.com/article-wiki/review-product-article/การใช้งาน-character-lcd

[5] LoRa Module, http://www.satorshop.com/product/562/sx1278-lora-long-range-3000m-uart-433mhz-rf-tx-rx-wireless-transmitter-and-receiver-transceiver-modu

[6] Arduino Mega ADK, https://store.arduino.cc/usa/arduino-mega-adk-rev3

[7] LoRa Module, https://werapun.com/lora-wan-workshop-f6e90efdd576

[8] OLED, https://men.kapook.com/view54324.html

[9] Buzzer Module, https://www.ioxhop.com/product/229/passive-buzzer-module

[10] Voltage Regulator Circuit, https://phukphan.blogspot.com/2017/02/voltage-regulator.html

ภาคผนวก

โค้ดโปรแกรมอาดูโน่

โค้ดภาคส่งสัญญาณ

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
Adafruit_SSD1306 display(-1);

#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);
void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);

display.begin(SSD1306_SWITCHCAPVCC, 0x3c);
display.clearDisplay();
display.setTextColor(WHITE);
display.setCursor(50,0);
display.setTextSize(2);
display.println("Tx");
display.setTextSize(2);
display.setCursor(22,16);
display.println("Ready...");
display.display();

}

void loop() {
if(Serial.available() > 0)
```

```
{
  String input = Serial.readString();
  mySerial.print(input);

    if(input=="1")
  {
    Serial.println(input);
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(WHITE);
    display.setCursor(25,0);
    display.println("Sending");
    display.setTextSize(1);
    display.setCursor(45,20);
    display.println("Rx No.1");
    display.setTextColor(BLACK, WHITE);
    display.display();
    }


  else if(input=="2")
  {
    Serial.println(input);
     display.clearDisplay();
     display.setTextSize(2);
     display.setTextColor(WHITE);
     display.setCursor(25,0);
     display.println("Sending");
```

```
    display.setTextSize(1);
    display.setCursor(45,20);
    display.println("Rx No.2");
    display.setTextColor(BLACK, WHITE);
    display.display();
}
else if(input=="3")
{
   Serial.println(input);
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(WHITE);
    display.setCursor(25,0);
    display.println("Sending");
    display.setTextSize(1);
    display.setCursor(45,20);
    display.println("Rx No.3");
    display.setTextColor(BLACK, WHITE);
    display.display();
}

 else if(input=="4")
 {
    Serial.println(input);
     display.clearDisplay();
     display.setTextSize(2);
     display.setTextColor(WHITE);
     display.setCursor(25,0);
     display.println("Sending");
```

```
      display.setTextSize(1);
      display.setCursor(45,20);
      display.println("Rx No.4");
      display.setTextColor(BLACK, WHITE);
      display.display();
    }
    else if(input=="5")
    {
      display.clearDisplay();
      display.setTextColor(WHITE);
      display.setCursor(50,0);
      display.setTextSize(2);
      display.println("Tx");
      display.setTextSize(2);
      display.setCursor(22,16);
      display.println("Ready...");
      display.display();

    }

    else
    {
      Serial.println("not found");
    }
  }
  delay(100);
}
```

โค้ดภาครับสัญญาณ

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
Adafruit_SSD1306 display(-1);

#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 4);
int output1 = 6;
int output2 = 8;
int val1 = 0;

void setup()
{
Serial.begin(9600);
mySerial.begin(9600);
pinMode(output1,OUTPUT);
pinMode(output2,OUTPUT);

 display.begin(SSD1306_SWITCHCAPVCC, 0x3c);
 display.clearDisplay();
 display.setTextColor(WHITE);
 display.setCursor(25,0);
 display.setTextSize(2);
 display.println("Rx No.1");
 display.setTextSize(2);
 display.setCursor(35,17);
```

```
display.println("Ready");
display.display();

}

void loop()
{
  if(mySerial.available() > 0){
    String input = mySerial.readString();
    if(input=="1")
    {
       Serial.println(input);
       display.clearDisplay();
       display.setTextSize(1);
       display.setTextColor(WHITE);
       display.setCursor(30,0);
       display.println("Order is");
       display.setTextSize(2);
       display.setCursor(5,20);
       display.println("Completed");
       display.setTextColor(BLACK, WHITE);
       display.display();
       for(int p=0; p<=2; p++){
       for(int i=0; i<=9; i++){
      digitalWrite(output1,HIGH);
      digitalWrite(output2,HIGH);
      delay(200);
      digitalWrite(output1,LOW);
      digitalWrite(output2,LOW);
```

```
      delay(200);
       }
       delay(2000);
       }
    }
    else
    {
      Serial.println("not found");
    }
  }
  delay(20);

}

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
Adafruit_SSD1306 display(-1);

#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 4);
int output1 = 6;
int output2 = 8;
int val1 = 0;

void setup()
{
Serial.begin(9600);
```

```
mySerial.begin(9600);
pinMode(output1,OUTPUT);
pinMode(output2,OUTPUT);

display.begin(SSD1306_SWITCHCAPVCC, 0x3c);
display.clearDisplay();
display.setTextColor(WHITE);
display.setCursor(25,0);
display.setTextSize(2);
display.println("Rx No.2");
display.setTextSize(2);
display.setCursor(35,17);
display.println("Ready");
display.display();

}

void loop()
{
  if(mySerial.available() > 0){
    String input = mySerial.readString();
    if(input=="2")
    {
       Serial.println(input);
       display.clearDisplay();
       display.setTextSize(1);
       display.setTextColor(WHITE);
       display.setCursor(30,0);
       display.println("Order is");
```

```
      display.setTextSize(2);
      display.setCursor(5,20);
      display.println("Completed");
      display.setTextColor(BLACK, WHITE);
      display.display();
      for(int p=0; p<=2; p++){
      for(int i=0; i<=9; i++){
     digitalWrite(output1,HIGH);
     digitalWrite(output2,HIGH);
      delay(200);
      digitalWrite(output1,LOW);
      digitalWrite(output2,LOW);
      delay(200);
       }
       delay(2000);
       }
    }
    else
    {
      Serial.println("not found");
    }
  }
  delay(20);

}

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
```

```
#include <Adafruit_SSD1306.h>
Adafruit_SSD1306 display(-1);

#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 4);
int output1 = 6;
int output2 = 8;
int val1 = 0;

void setup()
{
Serial.begin(9600);
mySerial.begin(9600);
pinMode(output1,OUTPUT);
pinMode(output2,OUTPUT);

display.begin(SSD1306_SWITCHCAPVCC, 0x3c);
display.clearDisplay();
display.setTextColor(WHITE);
display.setCursor(25,0);
display.setTextSize(2);
display.println("Rx No.3");
display.setTextSize(2);
display.setCursor(35,17);
display.println("Ready");
display.display();

}
```

```
void loop()
{
  if(mySerial.available() > 0){
    String input = mySerial.readString();
    if(input=="3")
    {
       Serial.println(input);
       display.clearDisplay();
       display.setTextSize(1);
       display.setTextColor(WHITE);
       display.setCursor(30,0);
       display.println("Order is");
       display.setTextSize(2);
       display.setCursor(5,20);
       display.println("Completed");
       display.setTextColor(BLACK, WHITE);
       display.display();
       for(int p=0; p<=2; p++){
       for(int i=0; i<=9; i++){
      digitalWrite(output1,HIGH);
      digitalWrite(output2,HIGH);
      delay(200);
      digitalWrite(output1,LOW);
      digitalWrite(output2,LOW);
      delay(200);
       }
       delay(2000);
       }
    }
```

```
    else
    {
      Serial.println("not found");
    }
  }
  delay(20);

}

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
Adafruit_SSD1306 display(-1);

#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 4);
int output1 = 6;
int output2 = 8;
int val1 = 0;

void setup()
{
Serial.begin(9600);
mySerial.begin(9600);
pinMode(output1,OUTPUT);
pinMode(output2,OUTPUT);

display.begin(SSD1306_SWITCHCAPVCC, 0x3c);
```

```
display.clearDisplay();
display.setTextColor(WHITE);
display.setCursor(25,0);
display.setTextSize(2);
display.println("Rx No.4");
display.setTextSize(2);
display.setCursor(35,17);
display.println("Ready");
display.display();

}

void loop()
{
  if(mySerial.available() > 0){
    String input = mySerial.readString();
    if(input=="4")
    {
       Serial.println(input);
       display.clearDisplay();
       display.setTextSize(1);
       display.setTextColor(WHITE);
       display.setCursor(30,0);
       display.println("Order is");
       display.setTextSize(2);
       display.setCursor(5,20);
       display.println("Completed");
       display.setTextColor(BLACK, WHITE);
       display.display();
```

```
      for(int p=0; p<=2; p++){
      for(int i=0; i<=9; i++){
     digitalWrite(output1,HIGH);
     digitalWrite(output2,HIGH);
     delay(200);
     digitalWrite(output1,LOW);
     digitalWrite(output2,LOW);
     delay(200);
      }
       delay(2000);
      }
   }
   else
   {
     Serial.println("not found");
   }
 }
 delay(20);

}
```