# บทที่ 3
# วิธีการดำเนินงาน

ในบทนี้จะเป็นการนำความรู้จากการศึกษาค้นคว้าในบทก่อนหน้ามาประยุกต์ใช้ในการตัดสินใจเลือกใช้อุปกรณ์ต่าง ๆ เช่นโมดูลสื่อสารไร้สายเซ็นเซอร์และอุปกรณ์อื่นที่เกี่ยวข้องรวมถึงการออกแบบวงจรโดยสิ่งสำคัญที่ต้องคำนึงถึงคือประสิทธิภาพความเสถียรภาพและความสะดวกสบายในการออกแบบ ความยืดหยุ่นในการปรับใช้งาน เพื่อสร้างตลอดจนการออกแบบซอฟแวร์ที่ใช้ควบคุมการทำงานของเครื่องมือ และโปรแกรมสำหรับสื่อสารกับผู้ใช้งานเพื่อให้อุปกรณ์ทุกตัวสามารถทำงานประสานกันได้อย่างลงตัว และมีประสิทธิภาพสูงสุด ซึ่งกระบวนการต่าง ๆ

3.1 กำหนดปัญหา
3.2 การวิเคราะห์
3.3 การออกแบบ
3.4 การพัฒนา
3.5 การคำนวณ Sensor
3.6 การออกแบบและทดสอบ
3.7 การติดตั้ง
3.8 การนำไปใช้และบำรุงรักษา
3.9 สรุป

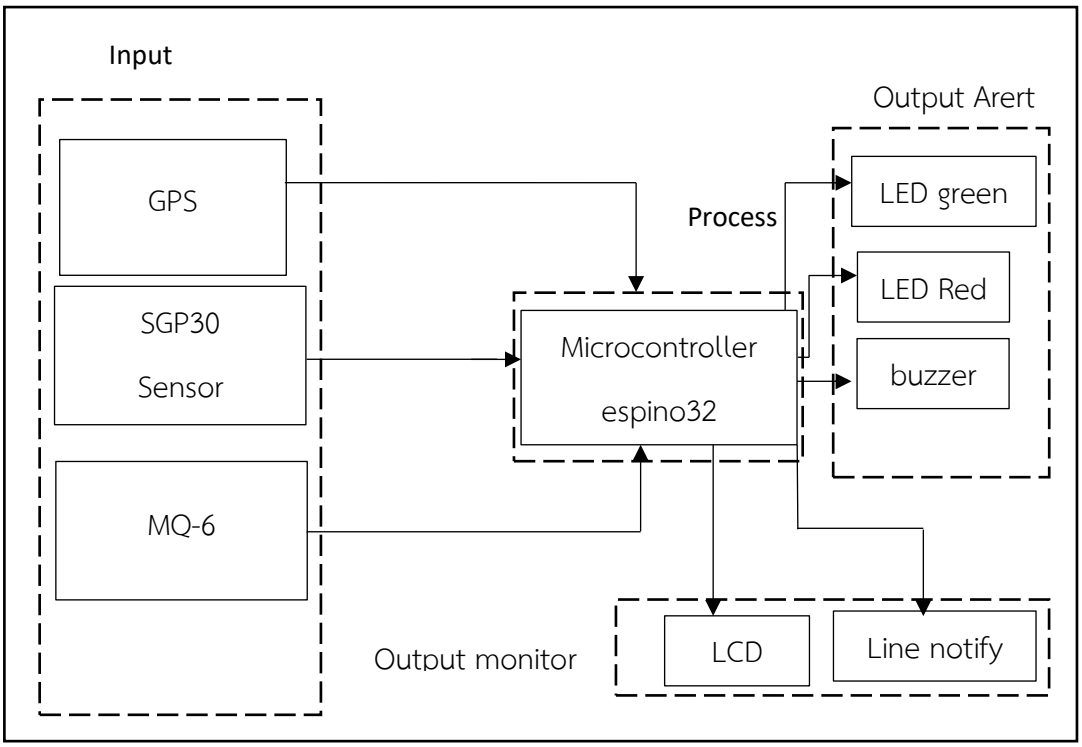
## 3.1 กำหนดปัญหา (Problem Definition)

ประเทศไทยพบปัญหาประชาชนเสียชีวิตขณะนอนหลับในรถยนต์ที่ติดเครื่องยนต์และเปิดแอร์ทุกปี ปีละประมาณ 1 –2 ราย ซึ่ง นพ.ประภาส กล่าวว่า การจอดรถติดเครื่องยนต์เปิดแอร์นอนในรถและปิดกระจกมิดชิดเป็นเรื่องที่มีอันตรายมาก เพราะเท่ากับเป็นการนอนดมก๊าซพิษในรถ โดยก๊าซพิษที่ทำให้เสียชีวิต ได้แก่ คาร์บอนมอนอกไซด์ ซึ่งเป็นก๊าซไม่มีสี ไม่มีกลิ่น อยู่ในไอเสียของรถยนต์ที่เกิดจากการเผาไหม้น้ำมัน ก๊าซสามารถไหลเวียนเข้ามาภายในตัวรถได้ทางระบบแอร์รถยนต์ซึ่งจะมีการดูดอากาศจากภายนอกและดูดเอาควันจากท่อไอเสียรถยนต์เข้ามาหมุนเวียนภายในรถด้วย ที่นอนภายในรถจึงสูดก๊าซพิษชนิดนี้เข้าไปสะสมในร่างกาย

**3.2 การวิเคราะห์ (Analysis)**

อุปกรณ์แจ้งเตือนเมื่อเกิดการล้มและจะแจ้งเตือนผ่านข้อความส่งเข้าไปในแอปพลิเคชันไลน์ ภายในตัวอุปกรณ์จะมีไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพสูงเป็นตัวควบคุมการทำงานของตัวอุปกรณ์ทั้งหมด และมีเซ็นเซอร์ต่าง ๆ ตามที่ได้ศึกษาค้นคว้ามาแล้ว ว่ามีความเหมาะสมที่จะนำมาใช้ในการวัดค่าองศา ตำแหน่ง ทำการตรวจสอบเงื่อนไขการทำงาน

3.2.1 บล็อกไดอะแกรม




**ภาพที่ 3-1** แสดงบล็อกไดอแกรม (ตัวส่ง)

จากภาพที่ 3-2 ส่วนของ Input ESPino32 ทำหน้าที่รับข้อมูลจากเซ็นเซอร์ และประมวลผลในหน้าที่รับข้อมูลจากInput เพื่อส่งคำสั่งไปยัง Output

3.2.1.1 ส่วนของ input รับค่าจาก sensor Sgp30, Gps, Mq-6

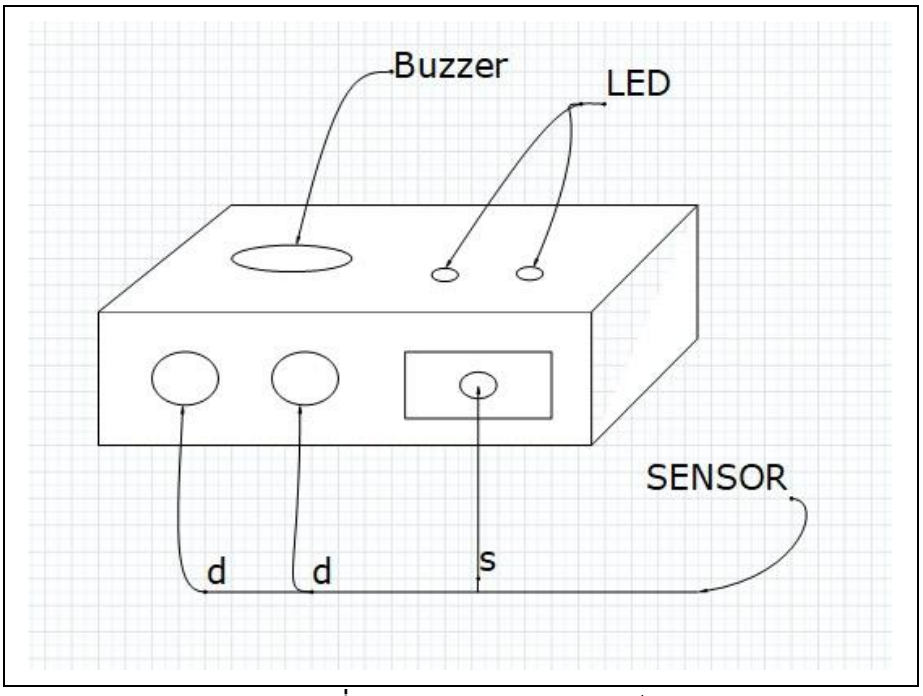
3.2.1.2 ส่วนของ Process คือ ESPino32 ทำการประมวลผล

3.2.1.3 ส่วนของ output arert มี LED ตัวทำการแดงสถาณะ และ ลำโพงในการส่งเสียงเตือน

3.2.1.4 ส่วนของ output monitor รับค่ามาเพื่อแสดงผลทางจอและไลน์

## 3.3 การออกแบบระบบ (System Design)

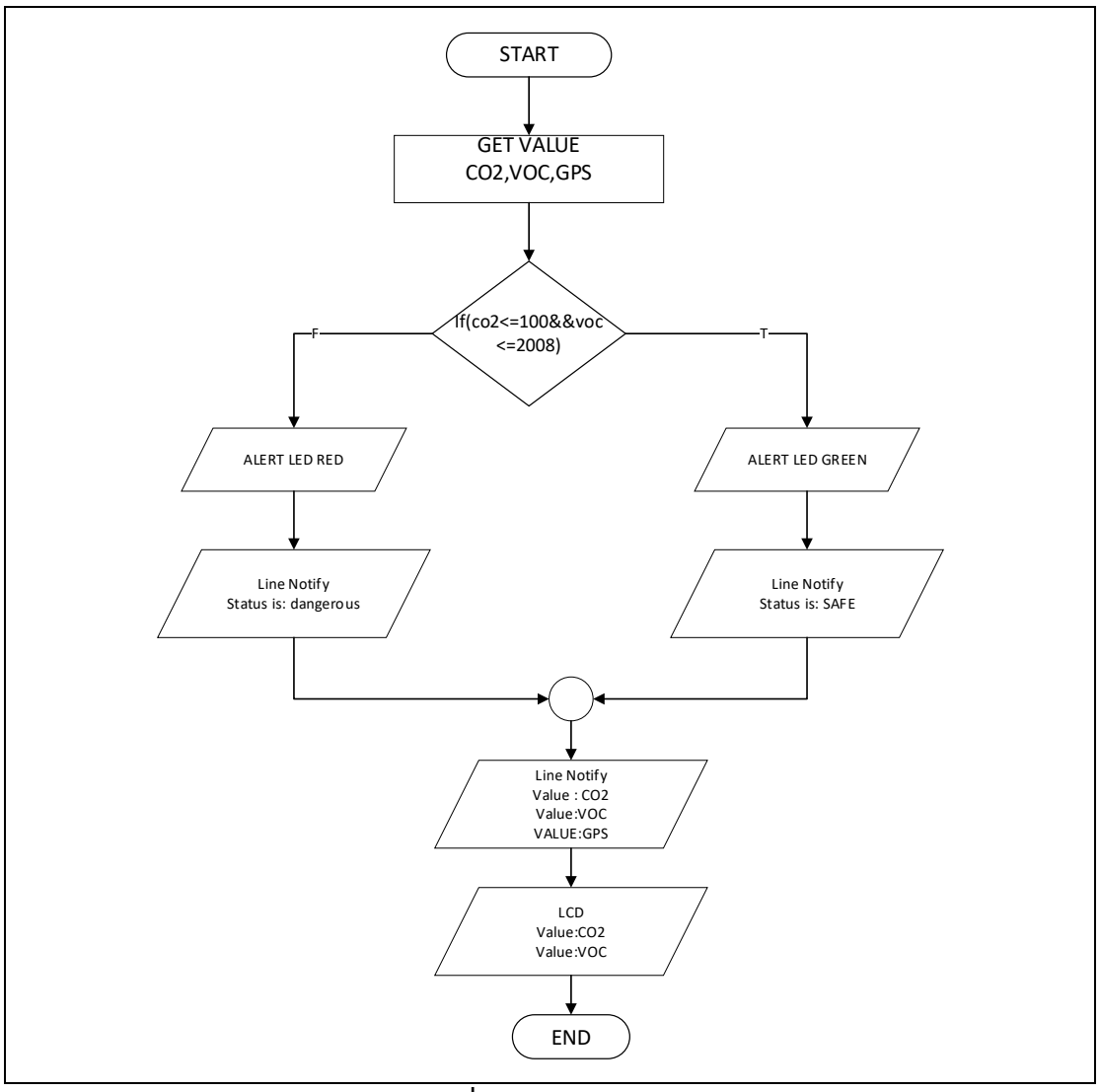
3.3.1 การออกแบบกล่องอุปกรณ์




**ภาพที่ 3-2** การออกแบบอุปกรณ์

จากภาพที่ 3-2 โครงสร้างของอุปกรณ์มีขนาดไม่ใหญ่มากเกิน 10 เซนติเมตร โดยมีการเจาะรูเพื่อนำเซ็นเซอร์โผล่ออกมาให้สามารถตรวจจับก๊าซได้มีลำโพงในตัวจำนวน 1 ตัว มีหลอดไฟ 2 ดวงบอกสถานะมีเซ็นเซอร์ทั้งหมด 3 ตัว สำหรับตรวจจับแก๊สและมีหน้าจอแสดงผล 1 จอ

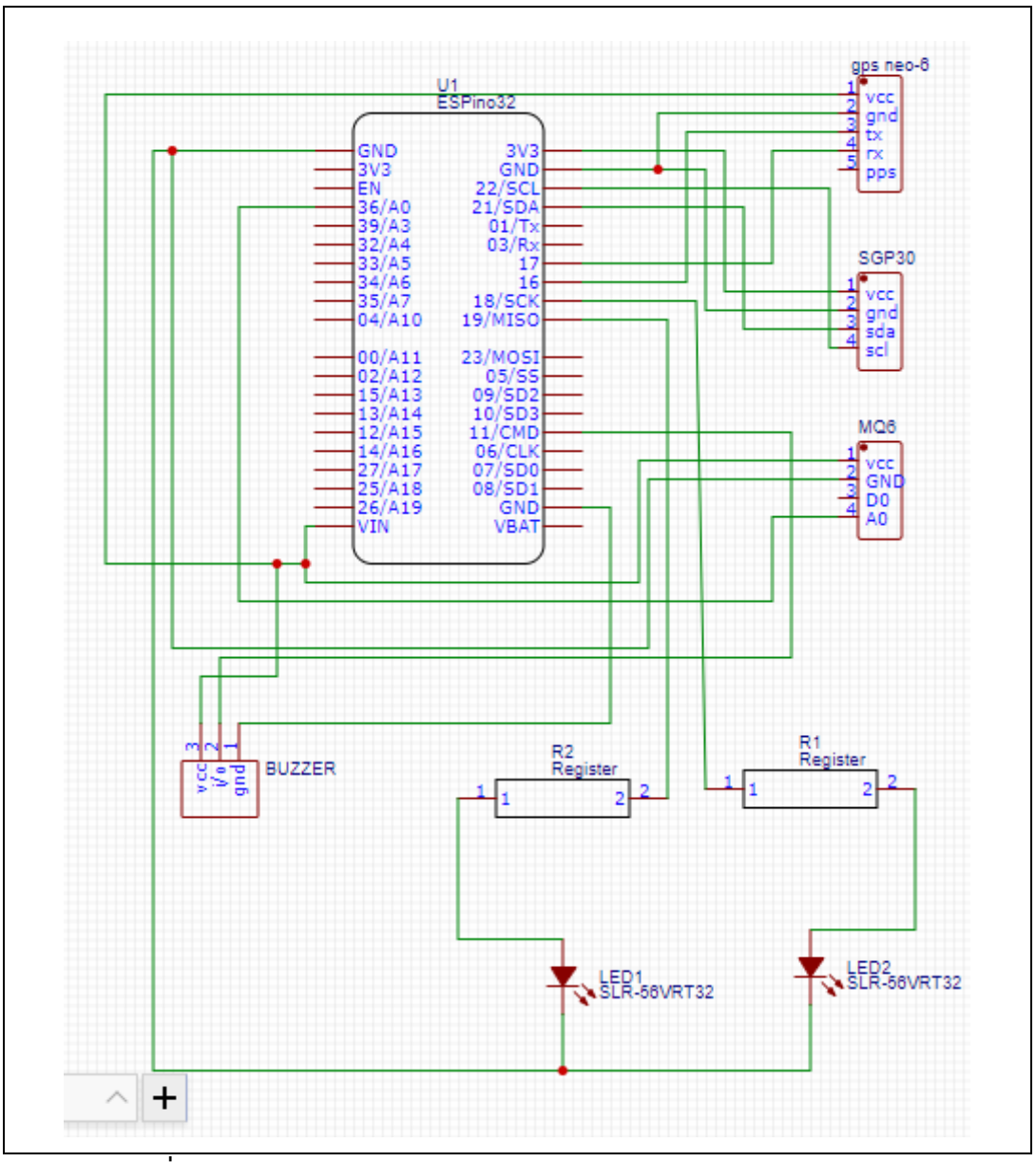
3.3.2 วิเคราะห์และออกแบบแผนผัง (Flowchart)




**ภาพที่ 3-3** การแบบระบบ

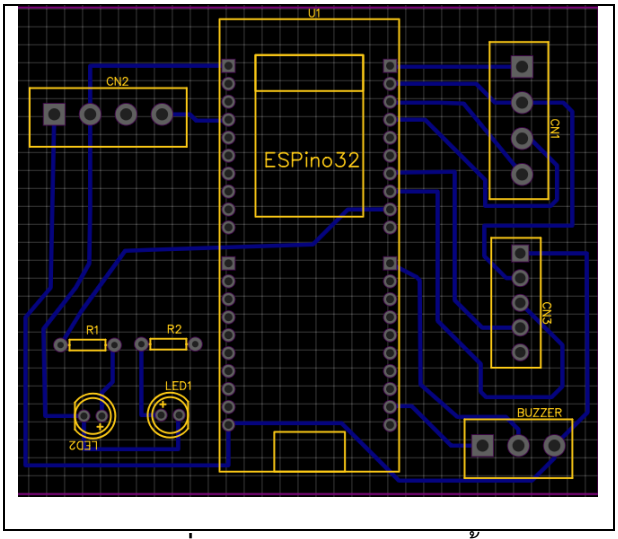
จากภาพที่ 3-3 เป็นขั้นตอนการทำงานของระบบทั้งหมด เริ่มจากจุดบนคือการเริ่มการทำงานของระบบจากนั้นระบบจะมาทำการรับค่าจาก sensor แล้วเข้าไปเช็คเงื่อนไขว่าให้ทำอะไรบ้าง ในระบบนั้นโปรแกรมจะเช็คว่า มีค่าเกินกำหนดหรือไม่ หากไม่มีจะไปทำการรอเวลาให้ครบ 1 ชม. เพื่อรอการแจ้งเตือน หากเป็นเท็จระบบจะแจ้งเตือนทันทีว่ามีแก๊สเกินกำหนด

3.3.3 แสดงการต่อวงจรระหว่างบอร์ด ESPino32 และ sensor ต่าง ๆ



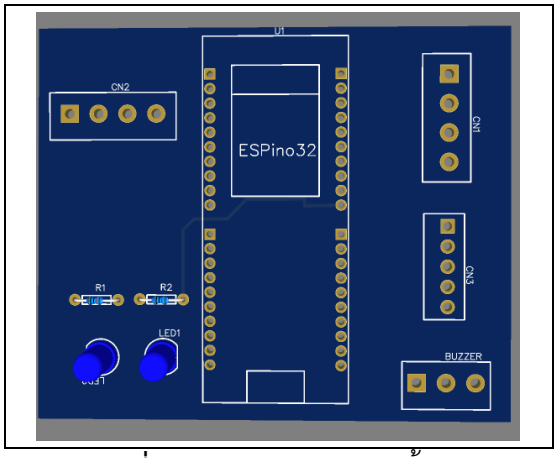

**ภาพที่ 3-4** แสดงการต่อวงจรของอุปกรณ์ระบบแจ้งเตือนมลพิษภายในรถยนต์

จากภาพที่ 3-4 เป็นการแสดงภาพการต่อวงจรของอุปกรณ์ระบบแจ้งเตือนมลพิษภายในรถยนต์ ของแต่ละอุปกรณ์



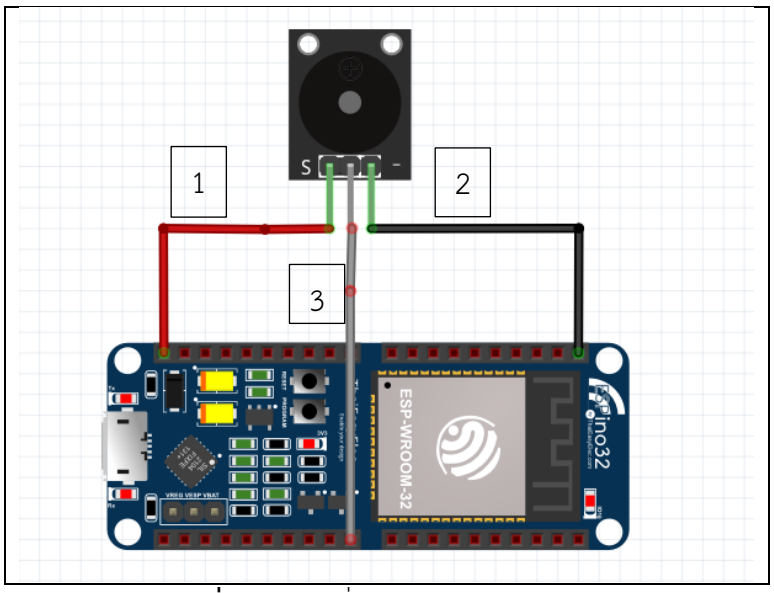

**ภาพที่ 3-5** การออกแบบลายปริ้น

จากภาพที่ 3-5 เป็นรูปแสดงลายวงจรแบบ 2 มิติโดยมีการจัดให้อุปกรณ์บอร์ดมาอยู่ตรงกลางเพื่อง่ายต่อการเชื่อมต่อกับอุปกรณ์อื่น ๆ




**ภาพที่ 3-6** แบบจำลองแผ่นปริ้น 3D

จากภาพที่ 3-6 เป็นรูปภาพแสดงแผ่นพีซีบีที่เป็นแบบ 3 มิติโดยมีการจัดให้อุปกรณ์บอร์ดมาอยู่ตรงกลางเพื่อง่ายต่อการเชื่อมต่อกับอุปกรณ์อื่น ๆ

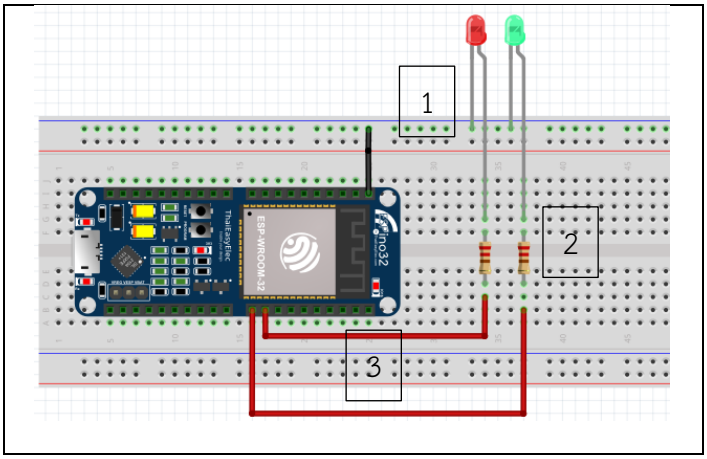



**ภาพที่ 3-7** การเชื่อมต่อ buzzer กับบอร์ด

จากภาพที่ 3-7 เป็นการเชื่อมต่อระหว่างบอร์ด ESPino32 กับ buzzer ซึ่ง buzzer มีขาทั้งหมด 3 ขาในการเชื่อมต่อคือมีขา GND ขา Vcc(5v) และ ขา output ที่เป็น digital

**ตารางที่ 3-1** ตารางการต่อวงจร Buzzer กับบอร์ด Espino32

| ลำดับ | รายละเอียด |
|---|---|
| 1 | เชื่อมต่อไฟเลี้ยงที่ 3v3 จ่ายให้กับ buzzer |
| 2 | เชื่อมต่อ GND เข้ากับขาของ buzzer |
| 3 | เชื่อมต่อ output เข้ากับขา 23 ของบอร์ด ESPino32 |

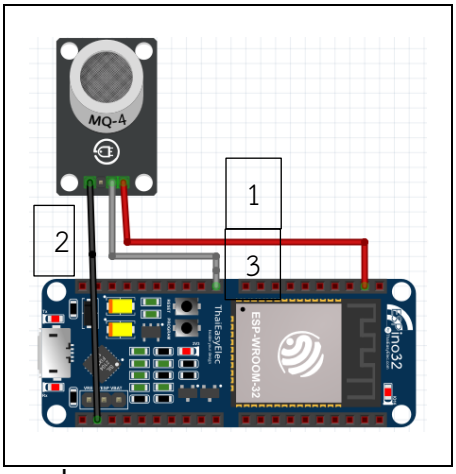



**ภาพที่ 3-8** การเชื่อมต่อ LED กับบอร์ด

จากภาพที่ 3-8 เป็นการเชื่อมต่อระหว่างไดโอดปรับค่าได้ระหว่างสีแดงและสีเขียวซึ่งแต่ละขากำหนดเป็น ขา vcc และ ขา GND

**ตารางที่ 3-2** ตารางการเชื่อต่อ led กับบอร์ด Espino32

| หมายเลข | หน้าที่ |
|---|---|
| 1 | เชื่อมต่อขาแคโทดเของไดโอดเปร่งแสงเข้ากับขา GND ของบอร์ด |
| 2 | เชื่อมต่อขาแอโนดของไดโอดเปร่งแสงเข้ากับขา resister |
| 3 | เชื่อมต่อขา output ขาที่18ของบอร์ดไปเข้ากับ ขา resister |

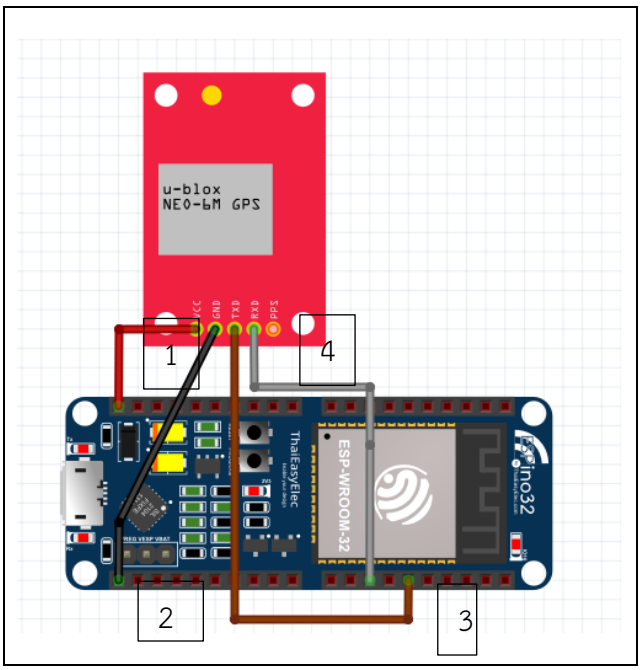



**ภาพที่ 3-9** การต่อ Sensor Mq กับบอร์ด

จากภาพที่ 3-9 เป็นการเชื่อมต่อระหว่างเซ็นเซอร์และบอร์ด Espino 32 โดยตัวเซ็นเซอร์มีทั้งหมด 4 ขา ขาที่ 1 คือ ขา Vcc ขาที่ 2 คือขา GND ขาที่ 3 คือขา A0 และขาที่ 4 คือขาของ D0

**ตารางที่ 3-3** ตารางการต่อวงจร Sensor MQ-6 กับบอร์ด Espino32

| ลำดับ | รายละเอียด |
|---|---|
| 1 | เชื่อมต่อไฟเลี้ยงที่ 3v3 จ่ายให้กับ Sensor |
| 2 | เชื่อมต่อ GND เข้ากับขาของ Sensor |
| 3 | เชื่อมต่อ I/O เข้ากับ A0 ของบอร์ด |

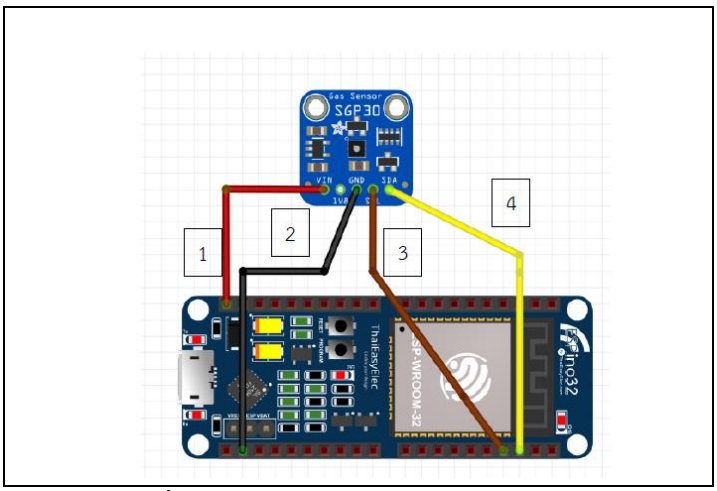



**ภาพที่ 3-10** การต่อ GPS Module กับบอร์ด

จากภาพที่ 3-10 เป็นการเชื่อมต่อระหว่าง GPS module และบอร์ด Espino 32 ตัว GPS module มี 5 ขา โดยมีขา ดังนี้ คือ 1 พินของไฟเข้าใช้ไฟ 5v 2 คือขา GND 3 คือขาของตัวรับข้อมูล Tx และสุดท้ายคือ Rx

**ตารางที่ 3-4** ตารางการต่อวงจร GPS Module กับบอร์ด ESPino32

| ลำดับ | รายละเอียด |
|---|---|
| 1 | เชื่อมต่อไฟเลี้ยงที่ 5v จ่ายให้กับ Sensor |
| 2 | เชื่อมต่อ GND เข้ากับขาของ Sensor |
| 3 | เชื่อมต่อ Rx เข้ากับขา 16 ของบอร์ด |
| 4 | เชื่อมขา Tx เข้ากับขา 17 ของบอร์ด |




**ภาพที่ 3-11** การต่อ Sgp30 Sensor กับบอร์ด

จากภาพที่ 3-10 เป็นการเชื่อมต่อระหว่าง Sgp30 และบอร์ด ESPino32 ตัว Sgp30 มี 54 ขา โดยมีพิน ดังนี้ คือ 1 ขาของไฟเข้าใช้ไฟ 3v3 2 คือขา GND 3 คือขาของตัวรับข้อมูล scl และสุดท้ายคือ sda

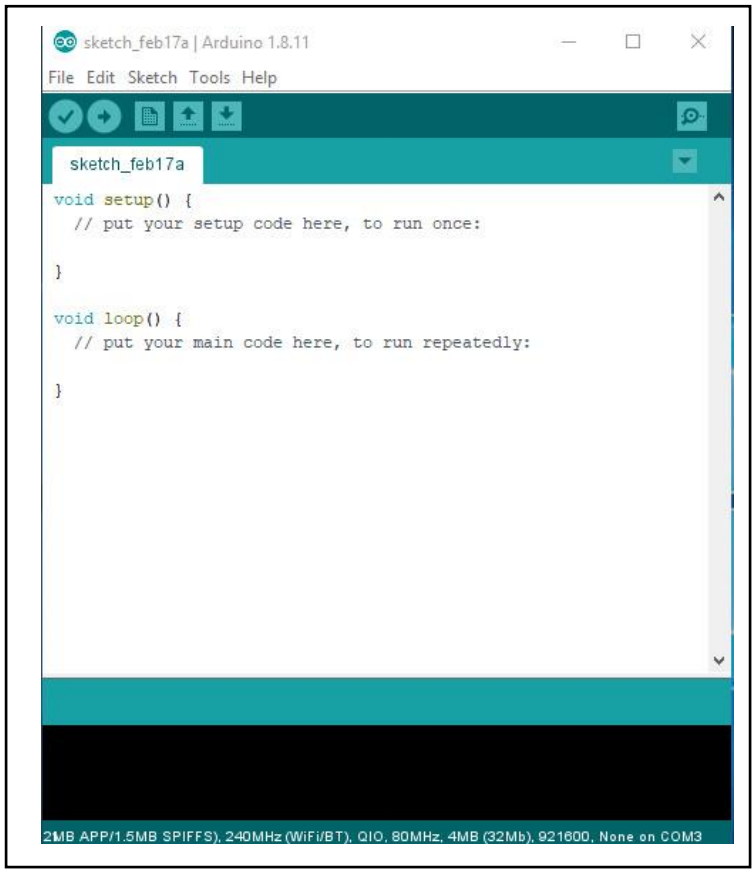
**ตารางที่ 3-5** ตารางการต่อวงจร Sgp30 Sensor กับบอร์ด ESPino32

| ลำดับ | รายละเอียด |
|---|---|
| 1 | เชื่อมต่อไฟเลี้ยงที่ 5v จ่ายให้กับ Sensor |
| 2 | เชื่อมต่อ GND เข้ากับขาของ Sensor |
| 3 | เชื่อมต่อ scl เข้ากับขา 22 ของบอร์ด |
| 4 | เชื่อมต่อ sda เข้ากับขา 21 ของบอร์ด |

## 3.4 การพัฒนา (Development)

3.4.1 Editor ในการเขียนโปรแกรม

ผู้จัดทำเลือกใช้ Arduino 1.8.11 ซึ่งเป็น Editor ใช้ในการเขียนโปรเพื่อควบคุมการทำงานของอุปกรณ์ ที่เป็น OpenSource ไม่คิดค่าใช้จ่ายในการใช้งาน




**ภาพที่ 3-12** หน้าโปรแกรม IDE ที่ใช้

จากภาพที่ 3-12 หน้าโปรแกรม IDE ที่ใช้เป็นหน้าตัวอย่างเริ่มต้นของโปรแกรม

3.4.2 Wifi Manager

WiFi Manager เป็นไลบรารี่ตัวหนึ่งที่ช่วยให้พัฒนาหรือผู้ใช้งานสามารถที่จะจัดการเรื่อง WiFi ให้กับอุปกรณ์ได้ง่ายขึ้น

ข้อดี

1. มี Captive Portal ช่วยให้สามารถระบุ ssid และ password ให้กับอุปกรณ์ได้โดยง่ายเพียงแค่เลือกเมนูและกรอกข้อมูลที่ต้องการแล้วกดบันทึก

2. สะดวกสบายไม่ต้องแก้ไขโค้ดโปรแกรมแล้วอัพโหลดใหม่ให้ยุ่งยาก

3. ช่วยดูแลเรื่องการเชื่อมต่อ Wi-Fi เมื่อมีปัญหา เช่น Access Point หายใช้งานไม่ได้ หรืออินเตอร์เน็ตหลุดตัว Wi-Fi-Manager จะทำการเปลี่ยนโหมดตัวเองให้เป็น Access Point เพื่อให้ผู้ใช้งานสามารถที่จะเชื่อมต่ออุปกรณ์เข้าไปเปลี่ยน said และ password ให้ใหม่

ข้อเสีย

Wi-Fi Manager ช่วยดูแลการเชื่อมต่อให้ได้ในระดับหนึ่งแต่การเปลี่ยนโหมดจาก Client เป็น AP นั้นจะมีการเรียก ESP. Reset() ซึ่งการเรียกบ่อย ๆ อาจจะมีผลทำให้อุปกรณ์ค้าง

ข้อควรระวัง

Wi-Fi Manager ใช้งานง่ายเพียงแค่เชื่อมต่อกับอุปกรณ์แต่ในขณะที่อุปกรณ์นั้นเปลี่ยนตัวเองให้เป็นโหมด Access Point ซึ่งผู้ใช้งานคนอื่นก็สามารถมองเห็นและเข้าถึงตัวอุปกรณ์ได้ ดังนั้นควรจะตั้ง password ของ AP ด้วยในกรณีที่ทำไปใช้งานจริง เพื่อความปลอดภัยของอุปกรณ์และมั่นใจได้ว่าจะไม่มีใครสามารถเข้าไปแก้ไขหรือสั่งงานอุปกรณ์ได้ Wi-Fi Manager เป็นไลบรารี่ตัวหนึ่งที่ช่วยให้พัฒนาหรือผู้ใช้งานสามารถที่จะจัดการเรื่อง Wi-Fi ให้กับอุปกรณ์ได้ง่ายขึ้น

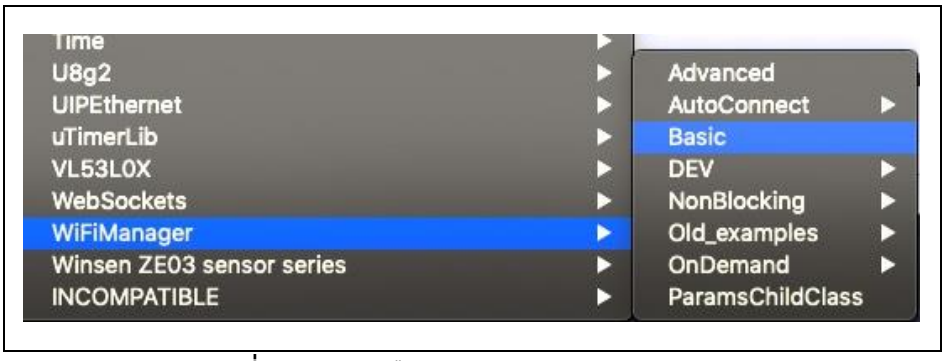
การติดตั้งการใช้งาน Wi-Fi manger

1. เปิดโปรแกรม Arduino IDE
2. เข้าไปที่ Sketch -> Include Library -> Manage Libraries
3. พิมพ์คำว่า Wi-Fi manager ในช่องFilter your search เมื่อมีค้นหาเจอแล้วให้คลิกบน Label ของWi-Fi Manager และคลิก Install


**ภาพที่ 3-13** Library Wi-Fi manager [15]

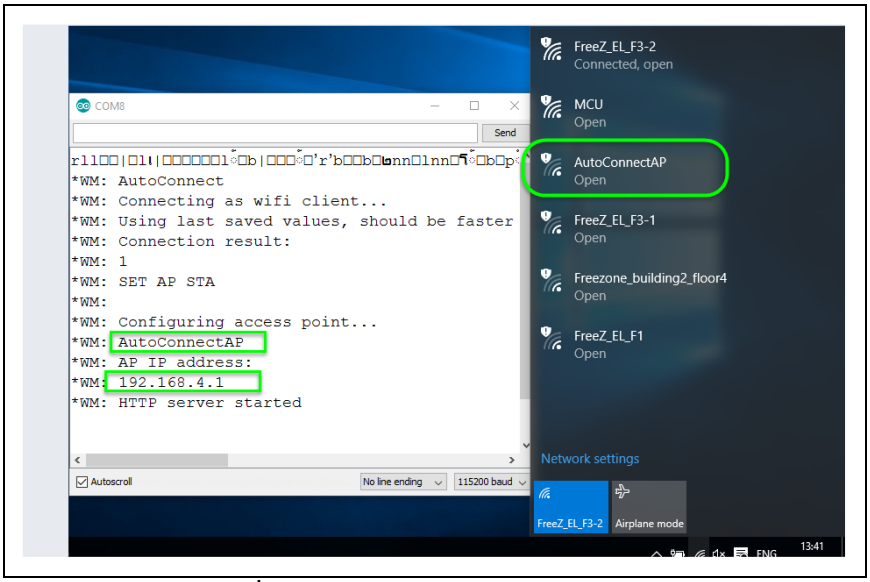
จากภาพที่ 3-13 หน้าแสดงการติดตั้ง Library Wi-Fi manager ที่สามารถติดตั้งได้เลยภายในโปรแกรม หรือ สามารถลงด้วยไฟล์ .zip ก็ได้

4. เมื่อทำการติดตั้งเรียบร้อยให้เข้าไปที่ Example แล้วเลือก Wi-Fi manager เพื่อทำการดูตัวอย่างตามภาพที่ 3-14

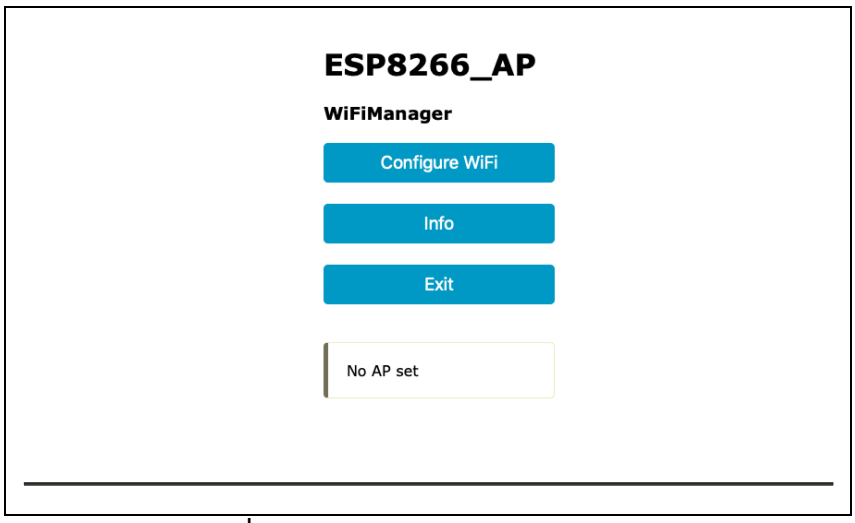



**ภาพที่ 3-14** การเลือกตัวอย่าง Wifi manager [15]

5. หลังจาก upload เสร็จแล้วดูที่ Serial Monitor โดยคลิกที่รูปแว่นขยาย หรือเข้าไปที่ Tools -> SerialMonitor เมื่ออุปกรณ์เข้าสู่โหมด AP จะได้ output ดังภาพที่ 3-14

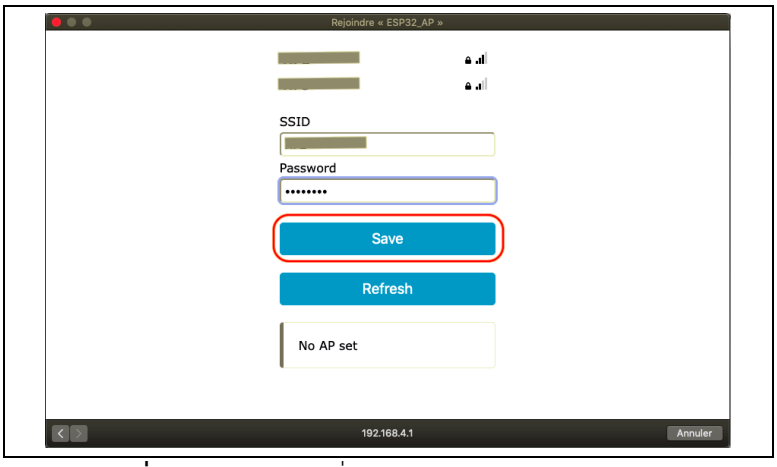



**ภาพที่ 3-15** การอัพโหลดเสร็จสมบูรณ์ [15]




**ภาพที่ 3-16** หน้าตัวอย่างการ CONFIG [15]

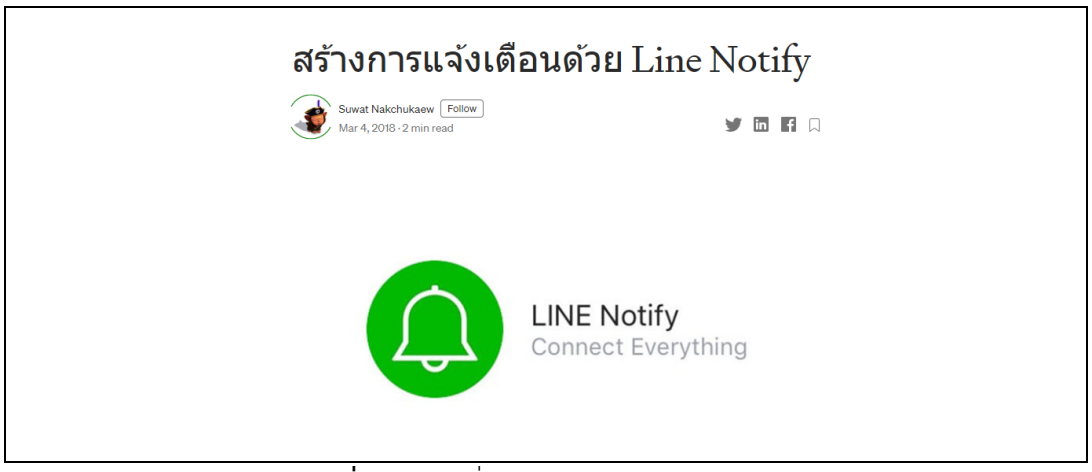
จากภาพที่ 3-13 หน้าตัวอย่างการ CONFIG ส่วนนี้คือให้ผู้ใช้ได้เข้ามา config ค่าต่าง ๆ ภายในระบบเช่น Wifi และการเพิ่มหรือแก้ไข Line Token เมื่อกดเข้า configure Wifi จะแสดงผลดังภาพที่ 3-17




**ภาพที่ 3-17** แสดงผลเมื่อเลือก config ในหน้าเมนูแรก [15]

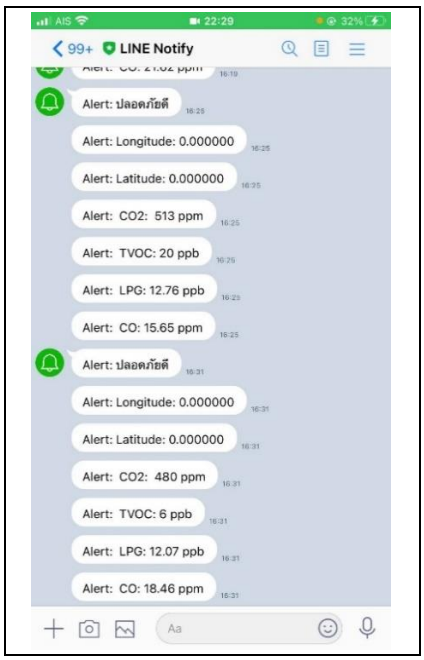
3.4.3 API Line notify

เป็น API ของ Line ที่ใช้ในการแจ้งเตือนโดยที่เราเพียงแค่ไปนำ line token จากเว็บของไลน์มาใส่ในส่วนของโค้ดเท่านี้ก็สามารถเอามาใช้ได้แล้ว




**ภาพที่ 3-18** ฟังชั่นใช้งาน API Line notify [5]

จากภาพที่ 3-13 Api Line notify ส่วนนี้ผู้จัดทำโปรเจ็คได้เอา API ของไลน์มาใช้เพื่อที่จะใช้ในการแจ้งเตือนค่าของแก๊ส และ พิกัด ได้รับทราบ การใช้ไลน์ในการแจ้งเตือนแบ่งเป็น 2 หน้าที่หลัก ๆ คือ 1. ใช้แจ้งเตือนในภาวะที่มีค่าแก๊สในระดับที่ปกติจะแจ้งเตือนทุก ๆ 1 ชั่วโมง 2. ใช้ในหากแก๊สเกินกำหนดจะแจ้งเตือนในทันทีตัวอย่างดังรูปที่ 3-14




**ภาพที่ 3-19** แสดงการแจ้งเตือน [5]

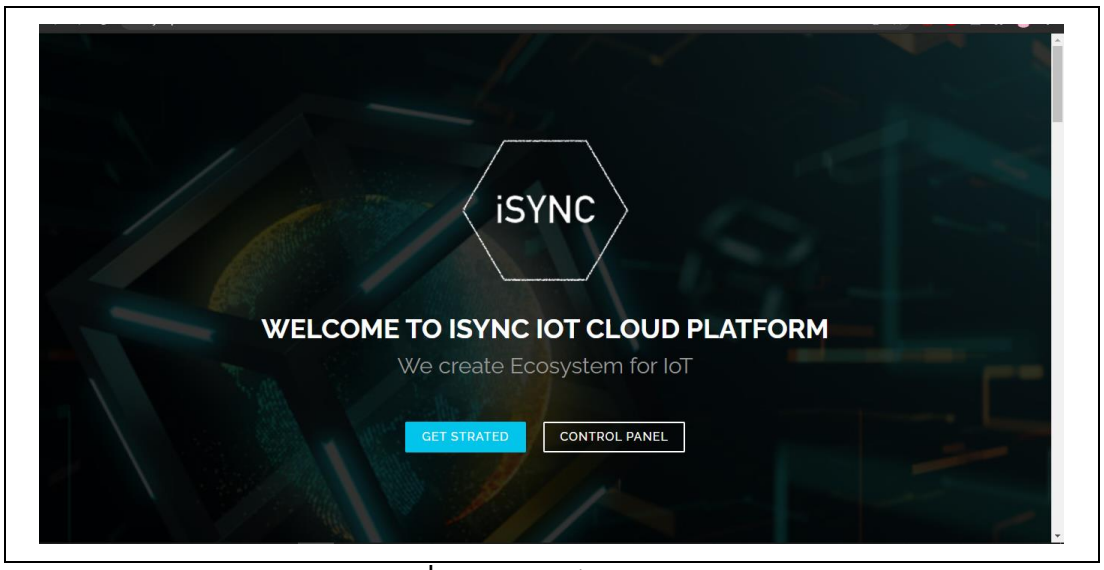
จากภาพที่ 3-14 การแดสงการแจ้งเตือนเที่ค่าส่งมามีรายการดังนี้ คือ 1.ค่าพิกัดที่ระบุตำแหน่งในรูปตัวอย่างเป็นการแจ้งเตือนจากที่อับสัญญาณพิกัดที่ได้เลยเป็น 0.0000, 0.00000 2. เป็นการส่งค่า Tvoc ที่มีจำนวน 58 ppb ในเวลานั้น 3. แจ้งเตือนค่า Lpg มีจำนวน 32.51 ppb ในขณะนั้น 3. คือค่า Co มีจำนวน 48.87 ppm ในขณะนั้น

3.4.4 Line Chatbot API

Line ChatBot คือบัญชีไลน์ที่สามารถตอบโต้กับผู้ใช้ได้อัตโนมัติ โดยที่เราไม่ต้องไปแตะต้องอะไรเลย นอกจากกลับบ้านมาเปิดดูสถิติเพื่อนำไปใช้ประโยชน์ สำหรับพ่อค้าแม่ค้าคุณสามารถสร้าง Line ChatBot ขึ้นมาเพื่อตอบคำถามต่าง ๆ ของลูกค้า เช่น ราคาสินค้า ประเภทสินค้า และอื่น ๆ โดยที่เราไม่ต้องมานั่งตอบให้เมื่อยมือ หรือหน่วยงานต่าง ๆ สามารถสร้างขึ้นมาเพื่อให้บริการด้านข้อมูลข่าวสารได้ เช่น ตอบคำถามด้านสภาพอากาศ เพียงให้ผู้ใช้แชร์ตำแหน่งเข้าไปในห้องแชท หรือแจ้งรายชื่อผู้สมัครสมาชิกผู้แทนราษฎรได้ เพียงผู้ใช้พิมพ์รหัสไปรษณีย์ของตนเอง เป็นต้น

ขั้นตอนการสร้าง line chat

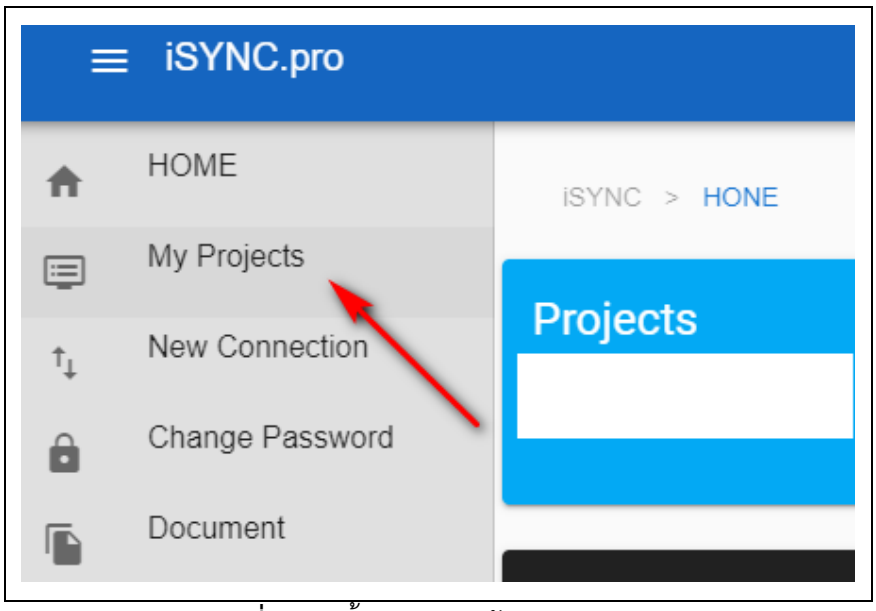
1. เข้าไปที่เว็บ ISYNC




**ภาพที่ 3-20** หน้าเว็บ ISYNC [16]

จากภาพที่ 3-14 เป็นหน้าเว็บไซต์ของ เว็บ ISYNC เข้ามาหน้านี้เพื่อทำการเลือกเมนูที่จะทำงานต่อไป จากเมนูที่มีให้ทำการเลือก CONTROL PANEL เพื่อไปยังหน้าต่อไป

2. จากนั้นให้ทำการสมัครสมาชิกเพื่อเข้าไปใช้บริการ
3. หลังจากนั้นให้ทำการเลือกไปที่ My Project ที่อยู่ทางด้านซ้ายมือของเว็บ
4. หลังจากนั้นทำการเลือกไปที่โปรเจ็ค กดสร้างโปรเจ็ค

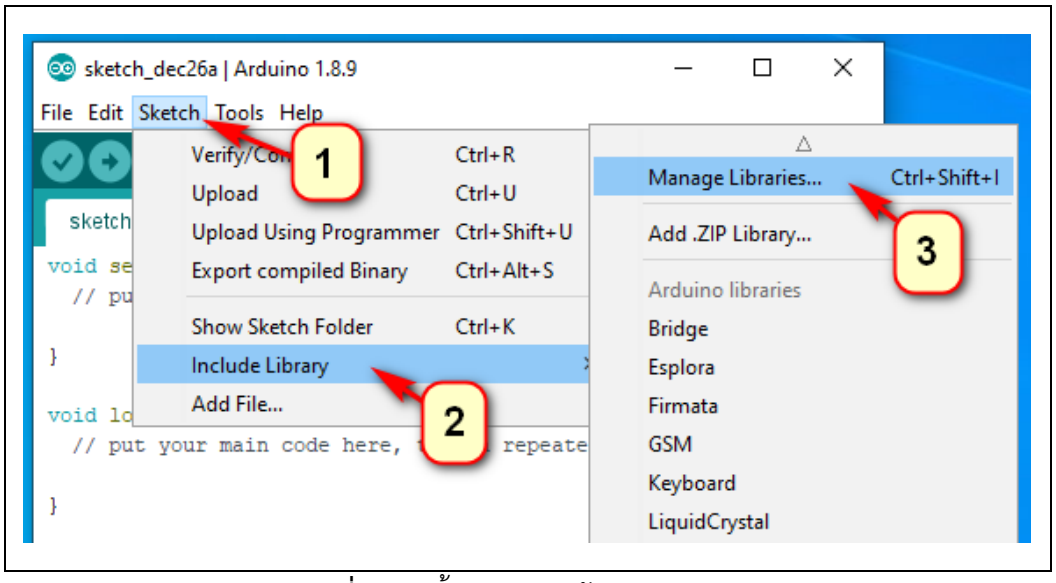



**ภาพที่ 3-21** ขั้นตอนการสร้าง project [16]

จากภาพที่ 3-21 เป็นขั้นตอนหลังจากที่เราทำการสมัครและกดเข้ามาในเมนู My Project แล้ว

5. หลังให้สร้าง Key ในการสร้างนั้นมีข้อควรระวังคือห้ามให้คนอื่นเห็นข้อมูลที่สื่อสารกันอยู่ อาจจะไม่ปลอดภัยเพราะมีคนแอบเข้ามาดูได้ และให้ copy มาด้วย

6. จากนั้นให้เข้าไปในโปรแกรม Arduino Ide เพื่อทำการส่งค่าเมื่อเข้าไปใน Arduino IDE แล้วให้ทำการติดตั้ง ไลบรารี่ให้เรียบร้อย จากนั้นเลือก Example เลือก ESP32 เลือก Isync mqtt

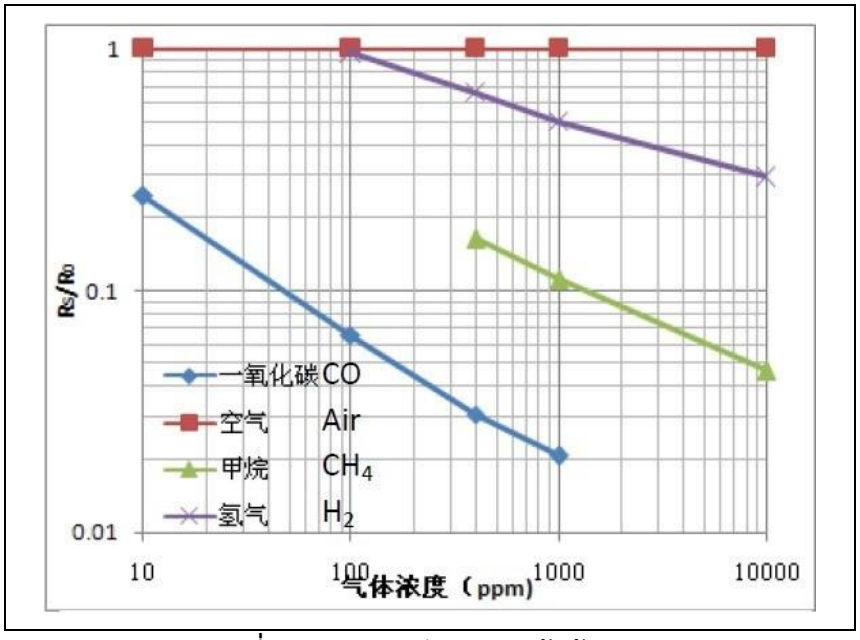



**ภาพที่ 3-22** ขั้นตอนการสร้าง example [16]

## 3.5 สูตรการคำนวณ Sensor

3.5.1 สูตรการคำนวณ MQ-7

เซ็นเซอร์คาร์บอนมอนอกไซด์ MQ-7 ได้รับการออกแบบมาโดยเฉพาะให้มีความไวต่อก๊าซคาร์บอนมอนอกไซด์ (CO) ซึ่งปล่อยออกมาจากยานพาหนะโรงงาน ฯลฯ เนื่องจากก๊าซนี้ถือว่าเป็นพิษต่อมนุษย์ในระดับหนึ่งความเข้มข้นของ CO จึงถูกใช้เพื่อกำหนด มลพิษทางอากาศในพื้นที่ที่กำหนด




**ภาพที่ 3-23** กราฟเทียบความเข้มข้นของ CO [17]

จากภาพที่ 3-23 คือกราฟของ Rs / R0 เทียบกับความเข้มข้นของก๊าซในหน่วย ppm
Rs คือความต้านทานของเซ็นเซอร์ในก๊าซเป้าหมาย
R0 คือความต้านทานในอากาศบริสุทธิ์

จะใช้กราฟนี้ในภายหลังเมื่อเราสร้างโค้ดมีสองวิธีในการอ่านเอาต์พุตจาก MQ-7 หนึ่งคือผ่านพิน DOUT ซึ่งให้ค่าสูงเมื่อถึงเกณฑ์ความเข้มข้นและต่ำเป็นอย่างอื่น เกณฑ์สามารถเปลี่ยนแปลงได้โดยการปรับทริมเมอร์บนบอร์ดเบรกเอาต์ซึ่งเป็น Rp ในแผนผังในขณะเดียวกันขา AOUT จะให้แรงดันไฟฟ้าที่แตกต่างกันซึ่งแสดงถึงความเข้มข้นของ CO เราสามารถแปลงการอ่านแรงดันไฟฟ้าเป็น ppm ได้หากเราดูเส้นโค้งลักษณะพิเศษด้านบนซึ่งเป็นพล็อตบันทึกการทำงานเราสนใจเฉพาะเส้นสีน้ำเงินบนพล็อตซึ่งให้ความเข้มข้นของ CO ฟังก์ชันของเส้นในพล็อตดังกล่าวได้รับเป็น

$$\boldsymbol{F(x) = \ F(0)\ (\frac{x}{x_0})^{\frac{log(F_{1/F_0})}{log(x_{1/x_0})}}}$$

**ภาพที่ 3-24** ฟังก์ชันของเส้นในพล็อต [17]

F1 = 0.25 , x1 = 10

F0 = 0.065, x0 = 100

$$F(x) = 0.065(\frac{x}{100})^{\frac{\log(0.25/0.065)}{\log(10/100)}} = 0.0065x^{-0.585}$$

or the relationship between concentration in ppm and RS/R0 is now

$$\frac{Rs}{R_0} = 0.00065ppm^{-0.585}$$

Solving for ppm

$$ppm = (1.538.46\frac{Rs}{R_0})^{-1.709}$$

**ภาพที่ 3-25** ฟังก์ชันของเส้นในพล็อต [17]

จากภาพที่ 3-25 สูตรของการคำนวณคือ F(x) ให้มีค่าเท่ากับ 0.065 คูณ x แล้วหารด้วย 100 ยกกำลังด้วยค่าของ log หาร log จากนั้นหาค่าความสัมพันธ์ความเข้มข้นของแก๊ส Rs/R0 คือ เอาค่าของ RS มาหารด้วย R0 จะเท่ากับ 0.00065 ppm ยกกำลังด้วย -0.585 หลังจากนั้นให้เอาค่าที่ได้มาตั้งสูตรการคำนวณดังภาพที่ 3-25

โค้ดการคำนวณ

```
float RS_gas = 0;
float ratio = 0;
float sensorValue = 0;
float sensor_volt = 0;
float R0 = 7200.0;

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

void loop() {
   sensorValue = analogRead(A0);
   sensor_volt = sensorValue/1024*5.0;
   RS_gas = (5.0-sensor_volt)/sensor_volt;
   ratio = RS_gas/R0; //Replace R0 with the value found using the sketch a
   float x = 1538.46 * ratio;
   float ppm = pow(x,-1.709);
   Serial.print("PPM: ");
   Serial.println(ppm);
   delay(1000);
}
```

**ภาพที่ 3-26** เป็นโค้ดการคำนวณค่าจากเซ็นเซอร์ [17]

จากภาพที่ 3-26 เป็นโค้ดการคำนวณค่าต่าง ๆ ที่ได้มา เป็นการนำค่าที่พอลตไว้มาใช้คำนวณเพื่อทำการหาค่า CO โค้ดแต่ละบรรทัดมีความหมายดังนี้

บรรทัดที่ 1 ประกาศตัวแปร RS_gas ให้มีค่าเท่ากับ 0

บรรทัดที่ 2 ประกาศตัวแปรชื่อว่า ratio ให้มีค่าเท่ากับ 0

บรรทัดที่ 3 ประกาศตัวแปรชื่อ SensorValue ให้มีค่าเท่ากับ 0

บรรทัดที่ 4 ประกาศตัวแปรชื่อ Sensor_volt ให้มีค่าเท่ากับ 0

บรรทัดที่ 5 ประกาศตัวแปรชื่อ R0 ให้เก็บค่าคงที่ 7200.0 ไว้ในรูปแบบชนิดทศนิยม

บรรทัดที่ 11 เข้าสู่ฟังชั่นการทำงานหลัก

บรรทัดที่ 12 กำหนดให้ตัวแปร SensorValue เท่ากับ ค่าที่บอร์ดรับมาจากขา A0

บรรทัดที่ 13 ให้ตัวแปร Sensor_volt เท่ากับ SensorValue หารด้วย 1024 คูณ 5

บรรทัดที่ 14 ตัวแปร Rs_gas เท่ากับ 0.5 ลบด้วยค่าของ Sensor_volt

บรรทัดที่ 15 ตัวแปร raito เก็บค่า Rs_gas ที่ทำการหารกับ ตัวแปร R0

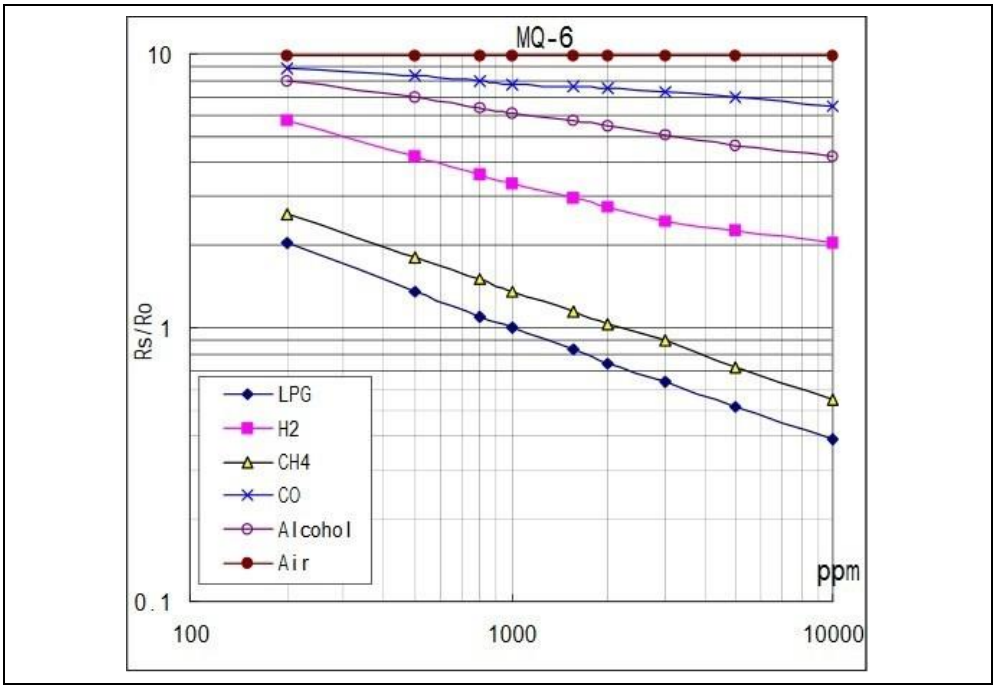
บรรทัดที่ 16 สร้างตัวแปรมาเพิ่มหนึ่งที่ชื่อว่า x นำมาเก็บค่าผลคูณของ ratio คูณ 1538

บรรทัดที่ 18 สร้างตัวแปรมาเพิ่มชื่อ ppm มาเก็บค่าของยกกำลัง x – 1.709

บรรทัดที่ 19 และ 20 คือการแสดงผลทางหน้าจอของ ppm

3.5.2 สูตรการคำนวณ MQ-6

MQ-6 เป็นอุปกรณ์เซมิคอนดักเตอร์สำหรับตรวจจับระดับของโพรเพนและบิวเทนในอากาศ เนื่องจากก๊าซปิโตรเลียมเหลว (LPG) ประกอบด้วยก๊าซทั้งสองนี้จึงสามารถใช้ MQ-6 เป็นเซ็นเซอร์ก๊าซหุงต้มได้ แผ่นข้อมูลแสดงกราฟการเข้าสู่ระบบเข้าสู่ระบบของความต้านทานของ MQ-6 คณะกรรมการที่มี 1,000 ppm ของแอลพีจี (R0) เพื่อต้านทานเมื่อก๊าซอื่น ๆ ที่มีอยู่ (RS)




**ภาพที่ 3-27** กราฟแสดงข้อมูลรับเข้าจากเซ็นเซอร์ [18]

จากภาพที่ 3-27 ในขณะที่สามารถอ่านก๊าซอื่น ๆ ได้ แต่ MQ-6 มีความไวต่อ LPG มากที่สุดตามกราฟ เรายังเห็นว่า RS / R0 เท่ากับค่าคงที่ 10 ในอากาศ ข้อเท็จจริงนี้มีประโยชน์เมื่อเราต้องคำนวณความเข้มข้นของก๊าซหุงต้มจริง (เป็น PPM) ในภายหลัง

สูตรการคำนวณของเซ็นเซอร์ MQ-6

$$F(x) = F_0 \left(\frac{x}{x_0}\right)^{\frac{\log(F_{1/F_0})}{\log(x_{1/x_0})}}$$

Here, we substitute F0 = 1, x0 = 1000 and F1 = 0.2, x1 = 10000.

$$F(x) = (1)\left(\frac{x}{1000}\right)^{\frac{\log\left(\frac{0.2}{1}\right)}{\log\left(\frac{10000}{1000}\right)}} = 0.001x^{-0.699}$$

The relationship between RS/R0 and ppm is therefore

$$\frac{Rs}{R_0} = 0.001\ ppm^{-0.690}$$

$$ppm = \left(100\frac{Rs}{R_0}\right)^{-1.431}$$

**ภาพที่ 3-28** สูตรการคำนวณ MQ-6 [18]

จากภาพที่ 3-28 เราสามารถกำหนดความเข้มข้นของ LPG ใน PPM ได้โดยใช้อัตราส่วน RS / R0 การกำหนดอัตราส่วน RS / R0 สามารถทำได้หากเรากำหนด R0 ของอุปกรณ์ก่อน การเรียกคืน R0 คือความต้านทานของอุปกรณ์เมื่อมีก๊าซ LPG อยู่ในอากาศ 1,000 ppm นี่เป็นเรื่องเดียวกับอากาศบริสุทธิ์ ในความเป็นจริง RS เท่านั้นที่จะเปลี่ยนสำหรับก๊าซที่แตกต่างกัน สังเกตว่า RS / R0 เท่ากับ 10 ในอากาศตามกราฟเราใช้ภาพร่างนี้เพื่อกำหนด R0 ของอุปกรณ์

```
float sensor_volt;
float RS_gas; // Get value of RS in a GAS
float R0 = 15000; //example value of R0. Replace with your own
float ratio; // Get ratio RS_GAS/RS_air
float LPG_PPM;

void setup() {
    Serial.begin(9600);
}
void loop() {
    int sensorValue = analogRead(A0);
    sensor_volt=(float)sensorValue/1024*5.0;
    RS_gas = (5.0-sensor_volt)/sensor_volt;
    ratio = RS_gas/R0;
    x = 1000*ratio
    LPG_PPM = pow(x,-1.431)//LPG PPM
    Serial.print("LPG PPM = ");
    Serial.println(LPG_PPM);
    Serial.print("\n\n");
    delay(1000);
}
```

**ภาพที่ 3-29** โค้ดการทำงาน MQ-6 [18]

จากภาพที่ 3-29 เป็นโค้ดการคำนวณค่าต่าง ๆ ที่ได้มาเป็นการนำค่าที่พอลตไว้มาใช้คำนวณเพื่อทำการหาค่าก๊าซปิโตรเลียมเหลว หรือ LPG โดยโค้ดมีการทำงานดังนี้

บรรทัดที่ 1 สร้างตัวแปรมารับค่าโดยใช้ชื่อว่า Sensor_Volt

บรรทัดที่ 2 สร้างตัวแปร Rs_gas

บรรทัดที่ 3 สร้างตัวแปร R0 ขึ้นมาแล้วเก็บค่า 15000 ไว้
บรรทัดที่ 4 สร้างตัวแปรชื่อ ratio
บรรทัดที่ 5 สร้างตัวแปรชื่อ LPG สำหรับเก็บค่าการคำรวณ
บรรทัดที่ 10 ฟังชั่นการทำงานหลักของโปรแกรม
บรรทัดที่ 11 SensorVolt เก็บค่าที่ได้จากขา A0 ไว้
บรรทัดที่ 12 ให้ตัวแปร Sensor_Volt เท่ากับ SensorValue หารด้วย 1024 คูณ 5
บรรทัดที่ 13 ตัวแปร Rs_gas เท่ากับ 0.5 ลบด้วยค่าของ Sensor_volt
บรรทัดที่ 14 ตัวแปร raito เก็บค่า Rs_gas ที่ทำการหารกับ ตัวแปร R0
บรรทัดที่ 15 ตัวแปร x เก็บค่า 1000 คูณ ratio ไว้
บรรทัดที่ 16 สร้างตัวแปร LPG_PPM ไว้เก็บค่าที่ได้จากการคำนวณ
บรรทัดที่ 17 แสดงผลผ่านหน้าจอ

### 3.6 การออกแบบการทดสอบระบบ (Testing)

ในส่วนนี้เป็นส่วนของการออกแบบการทดสอบการทำงานอุปกรณ์ต่าง ๆ ของระบบแจ้งเตือนมลพิษในรถยนต์ส่วนบุคคล

**ตารางที่ 3-6** ตารางออกแบบการทดสอบของ buzzer

| ลำดับ | ฟังก์ชันการทำงาน | การทดสอบ | |
|---|---|---|---|
| | | ผ่าน | ไม่ผ่าน |
| ทดสอบการทำงานของ buzzer | | | |
| 1 | สามารถแจ้งเตือนเมื่อค่า gas เกินกำหนด | | |

**ตารางที่ 3-7** ตารางออกแบบการทดสอบไฟสถานะ LED ทำงาน

| ลำดับ | ฟังก์ชันการทำงาน | การทดสอบ | |
|---|---|---|---|
| | | ผ่าน | ไม่ผ่าน |
| ทดสอบการทำงานของไฟสถานะ LED ทำงาน | | | |
| 1 | ไฟ LED สีแดง และ สีเขียว ทำงาน | | |

**ตารางที่ 3-8** ตารางออกแบบการทดสอบของ Sensor สามารถรับค่า VOC ได้

| ลำดับ | ฟังก์ชันการทำงาน | การทดสอบ | |
|---|---|---|---|
| | | ผ่าน | ไม่ผ่าน |
| ทดสอบการทำงาน Sensor สามารถรับค่า VOC ได้ | | | |
| 1 | Sensor Sgp30 ส่งค่า VOC แสดงผล LCD ได้ | | |

**ตารางที่ 3-9** ตารางออกแบบการทดสอบของ Sensor สามารถรับค่า $CO_2$ ได้

| ลำดับ | ฟังก์ชันการทำงาน | การทดสอบ | |
|---|---|---|---|
| | | ผ่าน | ไม่ผ่าน |
| ทดสอบการทำงาน Sensor สามาถรับค่า $CO_2$ ได้ | | | |
| 1 | Sensor Sgp30 ส่งค่า $CO_2$ แสดงผล LCD ได้ | | |

**ตารางที่ 3-10** ตารางออกแบบการทดสอบของ GPS Module

| ลำดับ | ฟังก์ชันการทำงาน | การทดสอบ | |
|---|---|---|---|
| | | ผ่าน | ไม่ผ่าน |
| ทดสอบการทำงานของ GPS Module | | | |
| 1 | สามารถส่งค่า Latitude | | |
| 2 | สามารถส่งค่า longitude | | |

**ตารางที่ 3-11** ตารางออกแบบการทดสอบของ sensor MQ-6

| ลำดับ | ฟังก์ชันการทำงาน | การทดสอบ | |
|---|---|---|---|
| | | ผ่าน | ไม่ผ่าน |
| ทดสอบ sensor MQ-6 | | | |
| 1 | สามารถรับค่า LPG ได้ | | |

**ตารางที่ 3-12** ตารางออกแบบการทดสอบจอ LCD

| ลำดับ | ฟังก์ชันการทำงาน | การทดสอบ | |
|---|---|---|---|
| | | ผ่าน | ไม่ผ่าน |
| การแสดงค่าผ่าน LCD | | | |
| 1 | แสดงค่า $CO_2$ ทาง LCD | | |
| 2 | แสดงค่า TVOC ทาง LCD | | |
| 3 | แสดงค่า LPG ทาง LCD | | |

**ตารางที่ 3-13** ตารางการออกแบบการทดสอบสามารถแจ้งเตือนผ่าน line notify

| ลำดับ | ฟังก์ชันการทำงาน | การทดสอบ | |
|---|---|---|---|
| | | ผ่าน | ไม่ผ่าน |
| line notify | | | |
| 1 | สามารถแจ้งเตือนผ่าน line | | |

### 3.7 การติดตั้ง (Implementation)

ติดตั้ง (Implementation) การติดตั้งอุปกรณ์ระบบแจ้งเตือนมลพิษในรถยนต์ส่วนบุคคล ผู้จัดทำโครงการพิเศษจะทำการติดตั้งการพัฒนาอุปกรณ์เป็นการพัฒนา เพื่อทำการทดสอบต่อไป

### 3.8 การนำไปใช้และการบำรุงรักษา (Operation and Support)

เมื่อได้ทำการติดตั้งอุปกรณ์ระบบแจ้งเตือนมลพิษในรถยนต์ส่วนบุคคบนพื้นที่จริง ผู้จัดทำโครงการพิเศษจะทำการปรับปรุงตามข้อเสนอแนะกับรถยนต์ที่ติดตั้งต่อไป

### 3.9 สรุป

จากผลการดำเนินงานที่ผ่านมาระบบแจ้งเตือนมลพิษในรถยนต์ส่วนบุคค มีการพัฒนาใช้เทคโนโลยีและอุปกรณ์สำหรับป้องกันและแจ้งเตือนมลพิษ จะสรุปผลการดำเนินงานทั้งหมดในบทถัดไป