ระบบสั่งเครื่องดื่ม โดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม
ORDER DRINKS SYSTEM WITH THE NOTIFICATION DEVICE

สุวดี เพิ่มสวัสดิ์
SUWADEE PHOEMSWAS
อลิสา กรอบพาน
ALISA KROBPAN

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมสารสนเทศ
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

ระบบสั่งเครื่องดื่ม โดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม
ORDER DRINKS SYSTEM WITH THE NOTIFICATION DEVICE

สุวดี เพิ่มสวัสดิ์
SUWADEE PHOEMSWAS

อลิสา กรอบแป้น
ALISA KROBPAN

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมสารสนเทศ
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

ORDER DRINKS SYSTEM WITH THE NOTIFICATION DEVICE

SUWADEE PHOEMSWAS

ALISA KROBPAN

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN INFORMATION ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2013

หัวข้อปริญญานิพนธ์	ระบบสั่งเครื่องดื่ม โดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม	
รายชื่อนักศึกษา	นางสาวสุวดี เพิ่มสวัสดิ์	รหัสนักศึกษา 53011798
	นางสาวอลิสา กรอบแป้น	รหัสนักศึกษา 53011901
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2556	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	อ.สรพงษ์ วชิรรัตนพรกุล	

ปริญญานิพนธ์ฉบับนี้ ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(อ.สรพงษ์ วชิรรัตนพรกุล)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	ระบบสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม	
รายชื่อนักศึกษา	นางสาวสุวดี เพิ่มสวัสดิ์	รหัสนักศึกษา 53011798
	นางสาวอลิสา กรอบแป้น	รหัสนักศึกษา 53011901
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2556	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	อ.สรพงษ์ วชิรรัตนพรกุล	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ จะกล่าวถึงระบบการสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม โดยนำไมโครคอนโทรลเลอร์ Arduino มาควบคุมการทำงานของระบบ มีจุดมุ่งหมายคือ เพื่อเพิ่มความสะดวกในการรอรับเครื่องดื่ม และแก้ไขข้อผิดพลาดในการให้บริการของร้านจำหน่ายเครื่องดื่ม รวมถึงช่วยลดต้นทุนการผลิตในการจ้างบริกรภายในร้าน สำหรับโครงงานนี้จะแบ่งการทำงานออกเป็นสองส่วน คือส่วนของอุปกรณ์รับรายการเครื่องดื่ม โดยนำไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 ทำหน้าที่เป็นส่วนประมวลผลข้อมูลที่รับมาจากปุ่มกด และแสดงผลออกทางจอแสดงผล LCD ขนาด 20 ตัวอักษร จำนวน 4 บรรทัด จากนั้นจะส่งข้อมูลผ่านอุปกรณ์รับ-ส่งสัญญาณไร้สาย nRF24L01 เพื่อส่งข้อมูลไปยังส่วนที่สอง คืออุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม โดยเมื่ออุปกรณ์ได้รับสัญญาณแล้ว ไมโครคอนโทรลเลอร์ Arduino Uno R3 ซึ่งทำหน้าที่เป็นส่วนประมวลผลข้อมูลของอุปกรณ์ส่วนนี้ จะทำการประมวลผลแล้วส่งข้อมูลไปยังจอแสดงผล LCD ขนาด 16 ตัวอักษร จำนวน 2 บรรทัด เพื่อแสดงรายละเอียดในการสั่งเครื่องดื่ม โดยเมื่อเครื่องดื่มเสร็จแล้ว อุปกรณ์รับรายการเครื่องดื่มจะส่งสัญญาณแจ้งเตือนมาอีกครั้ง ซึ่งจะแสดงผลออกทางหลอดไฟ LED และส่งเสียงแจ้งเตือน เพื่อให้ลูกค้ามารับเครื่องดื่ม

Thesis Title	**Order Drinks System with The Notification Device**	
Student	Miss Suwadee Phoemswas	Student ID. 53011798
	Miss Alisa Krobpane	Student ID. 53011901
Degree	Bachelor of Engineering	
Program	Information Engineering	
Year	2013	
Thesis Advisor	Mr. Sorapong Wachirarattanapornkul	

ABSTRACT

This thesis To talk about the drinks with notification device for a drink by the Arduino microcontroller to control the operation of a system. The aim is to easily wait for drinks and fix errors in the Services found. To reduce the production cost so as not having to hire a waiter inside the store. For this project, the work is divided into two parts. A piece of equipment by a leading drinks microcontroller Arduino Mega 2560 R3. Serves as the information processing is inherited from the keypad and display screen. An LCD display with 4 lines of 20 characters.It then sends the data through the device - nRF24L01 wireless signals to transmit the data to the second division. An alarm will sound when to get drinks when the device receives a signal. Microcontroller Arduino Uno R3 which serves as an information processing device. Processing, and then send the data to the LCD display 16 characters on 2 lines. Details in order to drink and when to drink it. The beverage equipment will alarm again. Which will show the bulb LED flashing. And emits alerts To allow customers to get the drinks.

กิตติกรรมประกาศ

โครงงานเล่มนี้สำเร็จลุล่วงได้ ด้วยความกรุณาจากอาจารย์สรพงษ์ วชิรรัตนพรกุล อาจารย์ที่ปรึกษาโครงงาน ที่ได้ให้ความรู้และช่วยแนะนำ พร้อมทั้งให้ข้อเสนอแนะ แนวคิด ตลอดจนแก้ไขข้อบกพร่องต่างๆ ด้วยความเอาใจใส่อย่างดีมาโดยตลอด จนโครงงานเล่มนี้เสร็จสมบูรณ์ ผู้ศึกษาจึงขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้ นอกจากนี้ผู้ศึกษาต้องขอขอบพระคุณนายณัฐพงศ์ อมรบัญชรเวช ที่คอยช่วยเหลือ ให้คำแนะนำ และช่วยกันแก้ไขปัญหาต่างๆ ร่วมกันมาโดยตลอดจนโครงงานเล่มนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี

ตลอดระยะเวลาในการทำโครงงานเล่มนี้ ขอกราบขอบพระคุณครอบครัวที่คอยช่วยเหลือในเรื่องต่างๆ คอยให้คำปรึกษา ให้ความห่วงใย ดูแลเอาใจใส่ อีกทั้งเป็นกำลังใจที่ดีเสมอมา และขอขอบพระคุณพี่ๆ น้องๆ และเพื่อนๆ ที่ช่วยให้คำแนะนำดีๆ ในการทำโครงงานเล่มนี้

สุดท้ายนี้ต้องขอขอบพระคุณทางสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ซึ่งเป็นแหล่งเรียนรู้และศึกษาค้นคว้าหาความรู้ อีกทั้งเป็นสถานที่ที่ใช้ทำโครงงานเล่มนี้มาโดยตลอดจนเสร็จสมบูรณ์

สุวดี เพิ่มสวัสดิ์
อลิสา กรอบแป้น

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

โดยทั่วไปแล้ว กิจการร้านจำหน่ายเครื่องดื่มยังคงเป็นระบบงานที่ใช้บุคลากรเป็นผู้ปฏิบัติงาน ซึ่งการปฏิบัติงานยังขาดความเป็นระเบียบเรียบร้อยและเกิดข้อผิดพลาด ซึ่งก่อให้เกิดปัญหาในหลาย ๆ ด้าน ยกตัวอย่างเช่น การต่อแถวบริเวณหน้าเคาน์เตอร์ (Counter) เพื่อรอรับเครื่องดื่ม ทำให้เกิดความไม่สะดวก การสื่อสารผิดพลาดในการสั่งเครื่องดื่มและเสิร์ฟ (Serve) เครื่องดื่มของพนักงาน การคำนวณราคาเครื่องดื่มผิดพลาด จำนวนพนักงานไม่เพียงพอต่อความต้องการของลูกค้าและความล่าช้าในการให้บริการ ในการดำเนินงานถ้าไม่มีการติดต่อประสานงานที่ดีอาจก่อให้เกิดปัญหาข้างต้น อาจส่งผลกระทบถึงภาพพจน์ของทางร้านและความพึงพอใจของลูกค้าผู้มาใช้บริการ ในขณะเดียวกันยังคงมีร้านจำหน่ายเครื่องดื่มเกิดขึ้นใหม่มากมาย ทำให้ลูกค้ามีทางเลือกที่หลากหลาย

เพื่อให้การดำเนินงานของกิจการร้านจำหน่ายเครื่องดื่มมีประสิทธิภาพมากยิ่งขึ้นในปัจจุบัน จึงได้มีการนำเทคโนโลยีเข้ามาช่วยในการดำเนินงาน ซึ่งสามารถช่วยให้การดำเนินงานมีระเบียบมากขึ้น เกิดข้อผิดพลาดน้อยลง มีความสะดวกรวดเร็ว ถูกต้อง และยังสามารถสร้างความพึงพอใจให้แก่ลูกค้าได้มากขึ้น เราจึงได้พัฒนาระบบสั่งเครื่องดื่มโดยมีอุปกรณ์ที่สามารถแสดงรหัสส่วนบุคคลในการรับเครื่องดื่มเพื่อป้องกันความผิดพลาดและราคารวมสุทธิของเครื่องดื่มที่สั่ง อีกทั้งอุปกรณ์นี้ยังสามารถแจ้งเตือนลูกค้าเมื่อเครื่องดื่มเสร็จแล้วด้วยหลอดไฟ LED และมีเสียงแจ้งเตือน เพื่อให้ลูกค้านำรหัสส่วนบุคคลไปรับเครื่องดื่ม

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานของบอร์ดไมโครคอนโทรลเลอร์ (Microcontroller) Arduino Mega 2560 R3 ในการรับข้อมูลและแสดงผลข้อมูลในส่วนของเคาน์เตอร์

2. เพื่อศึกษาการทำงานของบอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3 ในการรับข้อมูลและแสดงผลข้อมูลในส่วนของอุปกรณ์แจ้งเตือน

3. เพื่อศึกษาและพัฒนาระบบรับส่งข้อมูลแบบไร้สาย โดยใช้ nRF24L01

4. เพื่อศึกษาและพัฒนาการเขียนโปรแกรมโดยใช้โปรแกรม Arduino ซึ่งใช้ภาษา C++ ในการเขียน

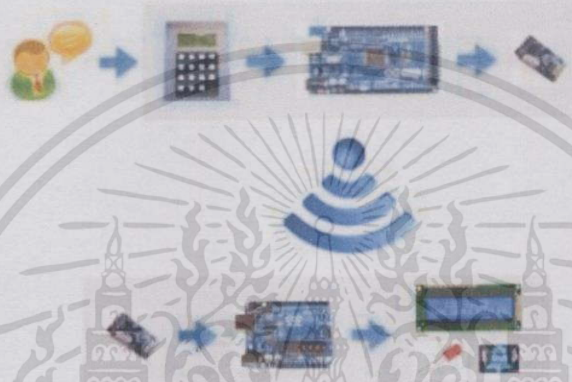
6. เพื่อพัฒนาระบบในการจำหน่ายเครื่องดื่มให้สะดวกมากขึ้น

7. เพื่อให้อุปกรณ์สามารถแจ้งเตือนลูกค้าในการรับเครื่องดื่มได้

8. มีการตรวจสอบรหัสส่วนบุคคลก่อนรับเครื่องดื่ม เพื่อป้องกันการผิดพลาด

1.3 ขอบเขตของโครงงาน

ระบบสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม แบ่งการทำงานออกเป็น 2 ส่วนหลัก ๆ คือ อุปกรณ์รับรายการเครื่องดื่มจะใช้งานอยู่ที่เคาน์เตอร์ และอุปกรณ์แจ้งเตือนในการรับเครื่องดื่มจะใช้งานโดยลูกค้า



รูปที่ 1.1 ภาพรวมการทำงานของระบบ

1.3.1 อุปกรณ์รับรายการเครื่องดื่ม

เมื่อลูกค้าสั่งเครื่องดื่มที่เคาน์เตอร์ โดยพนักงานจะใช้ปุ่มกด (Keypad) ที่เชื่อมต่อกับจอแสดงผล LCD ซึ่งเป็นอุปกรณ์รับรายการเครื่องดื่ม ภายในประกอบด้วยบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 เป็นส่วนประมวลผล และใช้ nRF24L01 แปลงสัญญาณข้อมูลเพื่อส่งออกไปยังเครื่องรับสัญญาณโดยใช้คลื่นวิทยุความถี่ 2.4 GHz โดยมีอากาศเป็นสื่อกลางในการส่ง

1.3.2 อุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

ในส่วนของเครื่องรับสัญญาณจะใช้ nRF24L01 รับข้อมูล แล้วส่งต่อไปยังบอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3 เป็นส่วนประมวลผลเพื่อใช้แสดงผลที่จอแสดงผล LCD ซึ่งจะแสดงรหัสส่วนบุคคลที่ใช้รับเครื่องดื่ม และราคาสุทธิที่ลูกค้าต้องชำระ สุดท้ายเมื่อเคาน์เตอร์ต้องการแจ้งเตือนให้ลูกค้ามารับเครื่องดื่ม ก็จะมีการสั่งการที่อุปกรณ์เครื่องรับรายการเครื่องดื่มให้แจ้งเตือนมายังอุปกรณ์แจ้งเตือนที่ลูกค้าอีกครั้งหนึ่ง เมื่ออุปกรณ์แจ้งเตือนได้รับสัญญาณแล้ว จะมีการสั่งการให้หลอดไฟ LED กระพริบและมีเสียงดังขึ้น ลูกค้าจะทราบถึงการแจ้งเตือน

แล้วกลับไปที่เคาน์เตอร์เพื่อไปรับเครื่องดื่ม โดยใช้รหัสส่วนบุคคลที่แสดงบนจอแสดงผล LCD ของอุปกรณ์เครื่องรับไปยืนยันและชำระค่าเครื่องดื่ม

1.4 ผลที่คาดว่าจะได้รับ

1. มีความรู้ความเข้าใจเกี่ยวกับการทำงานของระบบ และการสื่อสารไร้สาย
2. สามารถเขียนโปรแกรมภาษา C++ โดยใช้โปรแกรม Arduino ได้
3. ต้นแบบระบบการสั่งเครื่องดื่มมีความสะดวก รวดเร็วมากยิ่งขึ้น และมีความผิดพลาดน้อยลง
4. อุปกรณ์สามารถแจ้งเตือนลูกค้าในการรับเครื่องดื่มได้

1.5 อุปกรณ์และเทคโนโลยีที่ใช้ในโครงงาน

1.5.1 ฮาร์ดแวร์ (Hardware)

- เครื่องคอมพิวเตอร์ที่มีการเชื่อมต่อกับอินเตอร์เน็ต จำนวน 2 เครื่อง
- บอร์ดไมโครคอนโทรลเลอร์
  - Arduino Mega 2560 R3 จำนวน 1 บอร์ด
  - Arduino Uno R3 จำนวน 1 บอร์ด
- อุปกรณ์รับ-ส่งสัญญาณไร้สาย nRF24L01 จำนวน 3 อัน
- อุปกรณ์ลด Pin ในการต่อจอแสดงผล LCD จำนวน 3 อัน
- จอแสดงผล LCD
  - ขนาด 20x4 จำนวน 1 จอ
  - ขนาด 16x2 จำนวน 2 จอ
- ปุ่มกด ขนาด 4x4 จำนวน 1 แผง
- หลอดไฟ LED จำนวน 2 หลอด
- อุปกรณ์ส่งเสียง จำนวน 2 ชิ้น

1.5.2 ซอฟต์แวร์ (Software)

- โปรแกรม Arduino 1.5.4

1.6 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 ขั้นตอนการดำเนินงาน

ID	Task Name	2013							2014		
		Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar
1	Problem & Requirement										
2	Planning										
3	Learning about Hardware										
4	Learning about Arduino										
5	Analysis & Design										
6	Hardware Design										
7	Software Design										
8	Implementation										
9	Ordering Software										
10	Counter Device										
11	Notification Device										
12	Test & Debug										
13	Documentation										

1.6.1 ศึกษาค้นคว้าหาข้อมูล

ศึกษาหลักการทำงานของไมโครคอนโทรลเลอร์ประเภท Arduino ทั้งรุ่น Mega 2560 R3 และ Uno R3 โดยใช้โปรแกรม Arduino เขียนด้วยภาษา C++ รวมถึงหลักการเชื่อมต่อกับปุ่มกด, จอแสดงผล LCD และการรับ-ส่งข้อมูลแบบไร้สาย โดยใช้ nRF24L01

1.6.2 วางแผนและออกแบบ

ออกแบบโครงสร้างของอุปกรณ์รับรายการเครื่องดื่ม, อุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม, ระบบการทำงาน, การรับ-ส่งข้อมูลไร้สาย รวมถึงรูปแบบการแสดงผลออกทางจอแสดงผล LCD

1.6.3 สร้างอุปกรณ์และระบบการทำงาน

สร้างอุปกรณ์รับรายการเครื่องดื่มและอุปกรณ์แจ้งเตือน โดยใช้โปรแกรม Arduino เขียนด้วยภาษา C++ โดยให้ระบบการทำงานเป็นไปตามที่ได้ออกแบบไว้

1.6.4 การทดลอง

ทดลองการทำงานของอุปกรณ์รับรายการเครื่องดื่ม โดยมีการสั่งเครื่องดื่ม เพื่อให้มีการสร้างรหัสส่วนบุคคลเพื่อใช้รับเครื่องดื่ม ทดลองการรับ-ส่งข้อมูลไร้สาย รวมถึงทดลองระบบการแจ้งเตือนในการรับเครื่องดื่ม

1.6.5 การปรับปรุงแก้ไข

มีการปรับปรุงแก้ไขทั้งส่วนที่เป็นอุปกรณ์หรือฮาร์ดแวร์ และซอฟต์แวร์ซึ่งเป็นส่วนของระบบรับรายการเครื่องดื่ม รวมถึงการรับ-ส่งข้อมูลไร้สายเพื่อให้สามารถใช้งานได้จริง

1.6.6 ทำรูปเล่มเอกสารโครงงาน

เพื่อให้ระบบที่สร้างขึ้นสามารถใช้งานได้ตรงตามจุดประสงค์ จึงได้มีการจัดทำคู่มืออธิบายวิธีการใช้งานของอุปกรณ์รับรายการเครื่องดื่ม และอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้

ในการศึกษาเพื่อสร้างระบบสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม สำหรับโครงงานนี้ใช้ความรู้ทางด้านฮาร์ดแวร์และความรู้ทางด้านซอฟต์แวร์โดยอุปกรณ์ทางด้านฮาร์ดแวร์แบ่งเป็น 2 ส่วน คือ อุปกรณ์รับรายการเครื่องดื่มหรืออุปกรณ์ภาคส่งสัญญาณ และอุปกรณ์แจ้งเตือนหรืออุปกรณ์ภาครับสัญญาณซึ่งในการทำงานของระบบนี้มีทฤษฎีที่เกี่ยวข้องดังต่อไปนี้

2.1 ไมโครคอนโทรลเลอร์ (Microcontroller) ARDUINO

Arduino เป็นบอร์ดไมโครคอนโทรลเลอร์ตระกูล AVR โดยใช้ AVR ขนาดเล็กซึ่งเป็นตัวประมวลผลและสั่งงานเหมาะสำหรับนำไปใช้ในการศึกษาเรียนรู้ระบบไมโครคอนโทรลเลอร์และนำไปประยุกต์ใช้งานเกี่ยวกับการควบคุมอุปกรณ์ Input, Output ต่าง ๆ ได้มากมาย ทั้งในแบบที่เป็นการทำงานตัวเดียวอิสระหรือเชื่อมต่อสั่งงานร่วมกับอุปกรณ์อื่น ๆ เช่น คอมพิวเตอร์ PC ทั้งนี้ก็เนื่องมาจากว่า Arduino สนับสนุนการเชื่อมต่อกับอุปกรณ์ Input, Output ต่าง ๆ ได้มากมาย ทั้งแบบ Digital และ Analog เช่น การรับค่าจากสวิตช์ (Switch) หรืออุปกรณ์ตรวจจับ Sensor แบบต่าง ๆ รวมไปถึงการควบคุมอุปกรณ์ Output ต่าง ๆ ตั้งแต่ LED, หลอดไฟ, มอเตอร์(Motor), รีเลย์ (Relay) ฯลฯ โดยระบบฮาร์ดแวร์ของ Arduino สามารถสร้างและประกอบขึ้นใช้งานได้เอง อีกทั้ง Arduino เป็นระบบการพัฒนาไมโครคอนโทรลเลอร์แบบ Open Source รวมทั้งการเปิดเผย Source Code และตัวอย่างต่าง ๆ ไว้มากมาย

2.1.1 ภาษาซี กับ Arduino

โปรแกรมภาษาของ Arduino จะใช้ภาษา C++ ซึ่งเป็นรูปแบบของโปรแกรมภาษาซีประยุกต์แบบหนึ่ง ที่มีโครงสร้างของตัวภาษาโดยรวมใกล้เคียงกันกับภาษาซีมาตรฐาน ANSI-C อื่น ๆ เพียงแต่ได้มีการปรับปรุงรูปแบบในการเขียนโปรแกรมบางส่วนที่ผิดเพี้ยนไปจาก ANSI-C เล็กน้อย เพื่อช่วยลดความยุ่งยากในการเขียนโปรแกรมและให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้ง่ายและสะดวกมากขึ้นกว่าการเขียนภาษาซีตามแบบมาตรฐานของ ANSI-C โดยตรง

โดยประวัติความเป็นมาของภาษาซีนั้น ถูกบันทึกไว้ว่า ภาษาซี (C Programming Language) นั้นถูกพัฒนาขึ้นในปี ค.ศ.1970 โดย Dennis Ritchie แห่ง Bell Laboratories ซึ่งในช่วงแรกๆภาษาซีถูกใช้งานเฉพาะแต่ในห้องปฏิบัติการของ Bell จนกระทั่งปี 1978 Brian Kerninghan กับ Dennis Ritchie จึงได้ออกหนังสือกำหนดมาตรฐานของภาษาซี ออกมาเผยแพร่ ข้อกำหนดนี้ถูกเรียกว่า K&R C ทำให้ภาษาซีเริ่มเป็นที่รู้จักของคนทั่วไปกันมากขึ้น

จนกระทั่งปี 1980 ภาษาซี ก็ได้รับความนิยมมากขึ้นเป็นลำดับ จนได้มีการพัฒนาโปรแกรมที่ใช้ในการแปลภาษาซี (C-Compiler) ออกมาใช้งานกันอย่างแพร่หลาย ซึ่งสิ่งที่ทำให้ภาษาซีได้รับความนิยมในเวลาอันรวดเร็วก็เนื่องจากว่า ภาษาซีมีความได้เปรียบและมีความอ่อนตัวในการใช้งานที่เหนือว่าภาษาอื่น ๆ คือ ภาษาซี สามารถนำไปใช้งานบนระบบฮาร์ดแวร์ที่มีความแตกต่างกันได้หลากหลาย โดยผู้ใช้เพียงแต่เลือกใช้ตัวแปลคำสั่งการแปลภาษาซี ให้ตรงกับระบบฮาร์ดแวร์ที่ใช้งานอยู่ ส่วนเรื่องรูปแบบในการเขียนโปรแกรมจะเป็นมาตรฐานอันเดียวกัน จนในบางครั้ง อาจสามารถย้ายโปรแกรมจากระบบฮาร์ดแวร์หนึ่งไปใช้งานกับอีกระบบหนึ่งได้ ทำให้ไม่ต้องเสียเวลาในการศึกษาโปรแกรมใหม่ๆให้เสียเวลา เนื่องจากว่าภาษาซี เป็นภาษาที่มีรูปแบบการใช้งานที่ง่าย คือ มีแต่ข้อกำหนดในการใช้งาน หรือ Syntax แต่ไม่มีฟังก์ชั่นสำเร็จรูป (Built-in Function) ใด ๆ รวมอยู่ในตัวของภาษาด้วย โดยส่วนที่เป็นฟังก์ชั่นการใช้งานต่าง ๆ เช่น การดำเนินเกี่ยวกับ Input/Output การจองหน่วยความจำ (Memory Allocation) จะต้องสร้างขึ้นใช้งานเองหรืออาจใช้วิธีการเรียกใช้ฟังก์ชั่นที่ตัวแปลคำสั่งการแปลภาษาซี สร้างเตรียมไว้ให้ใช้งานในรูปแบบของ Library Function ซึ่งคำสั่งหรือฟังก์ชั่นส่วนที่เป็น Library Function นี้เอง ที่เป็นสาเหตุทำให้ภาษาซี มีความแตกต่าง ทาง American National Standard Institute (ANSI) จึงได้ทำการตั้งข้อกำหนดมาตรฐานของภาษาซีขึ้น โดยเรียกว่า “ANSI-C” เพื่อใช้เป็นข้อบังคับและคงมาตรฐานของภาษาซีไว้ไม่ให้เปลี่ยนแปลงไปจากเดิม ซึ่งส่วนของคำสั่งที่เป็นภาษาซีตามมาตรฐานของ ANSI-C จะมี 32 คำสั่ง คือ

auto, breakcase, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while

คำสั่งทั้ง 32 คำสั่งนี้ ไม่ว่าจะอยู่บนการแปลภาษาซี ตัวใดก็จะต้องมีข้อกำหนดและรูปแบบการใช้งานของคำสั่งที่เหมือนกันทั้งหมด ยกเว้น การแปลภาษาซีตัวนั้น ไม่รองรับคำสั่งของ ANSI-C ทั้งหมด ซึ่งส่วนนี้ต้องดูจากคุณสมบัติของภาษาซีที่จะใช้ว่าเป็นอย่างไร

Arduino นั้นไม่ใช่การแปลภาษาซี โดยตรง แต่ Arduino จะมีลักษณะการทำงานเช่นเดียวกันกับ Text Editor ของภาษา C++ ตัวหนึ่ง โดยจะทำงานร่วมกับ Utility บางส่วนที่ Arduino สร้างขึ้นมารองรับ โดย Arduino จะใช้รูปแบบการทำงานของ Text Editor เป็นฉากหน้าในการติดต่อสื่อสารกับผู้ใช้เท่านั้น ส่วนเบื้องหลังจริง ๆ นั้น Arduino จะไปเรียกใช้ตัวแปลภาษาซีและ Utility อื่น ๆ ที่ใช้เป็นเครื่องมือพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์อีกทีหนึ่ง

โครงสร้างการเขียนโปรแกรม ภาษาซี ของ Arduino

ภาษาซีของ Arduino จะจัดแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมออกเป็นส่วนย่อยๆ หลายๆส่วน โดยเรียกแต่ละส่วนว่า ฟังก์ชั่น และ เมื่อนำฟังก์ชั่น มารวมเข้าด้วยกัน ก็จะเรียกว่าโปรแกรมโดยโครงสร้างการเขียนโปรแกรมของ Arduino นั้น ทุกๆโปรแกรมจะต้อง

ประกอบไปด้วยฟังก์ชั่นจำนวนเท่าใดก็ได้ แต่อย่างน้อยที่สุดต้องมีฟังก์ชั่น จำนวน 2 ฟังก์ชั่น คือ setup( ) และ loop( )

โครงสร้างพื้นฐานของภาษาซีที่ใช้กับ Arduino นั้น จะประกอบไปด้วย 3 ส่วนใหญ่ๆด้วยกัน คือ

- header ในส่วนนี้จะมีหรือไม่มีก็ได้ ถ้ามีต้องกำหนดไว้ในส่วนเริ่มต้นของโปรแกรม ซึ่งส่วนของ header ได้แก่ ส่วนที่เป็น Compiler Directive ต่าง ๆ รวมไปถึงส่วนของการประกาศตัวแปร และ ค่าคงที่ต่างๆที่จะใช้ในโปรแกรม
- setup( ) ในส่วนนี้เป็นฟังก์ชั่นบังคับที่ต้องกำหนดให้มีในทุกๆโปรแกรม ถึงแม้ว่าในบางโปรแกรมจะไม่ต้องการใช้งานก็ยังจำเป็นต้องประกาศไว้ด้วยเสมอ เพียงแต่ไม่ต้องเขียนคำสั่งใดๆไว้ในระหว่างวงเล็บปีกกา { } ที่ใช้เป็นตัวกำหนดขอบเขตของฟังก์ชั่น โดยฟังก์ชั่นนี้จะใช้สำหรับบรรจุคำสั่งในส่วนที่ต้องการให้โปรแกรมทำงานเพียงรอบเดียวตอนเริ่มต้นทำงานของโปรแกรมครั้งแรกเท่านั้น ซึ่งได้แก่คำสั่งเกี่ยวกับการ setup ค่าการทำงานต่าง ๆ เช่น การกำหนดหน้าที่การใช้งานของ PinMode และการกำหนดค่า Baudrate สำหรับใช้งานพอร์ตสื่อสารอนุกรม เป็นต้น
- loop( ) เป็นส่วนฟังก์ชั่นบังคับที่ต้องกำหนดให้มีในทุกๆโปรแกรมเช่นเดียวกันกับฟังก์ชั่น Setup( ) โดยฟังก์ชั่น loop( ) นี้จะใช้บรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเป็นวงรอบซ้ำๆกันไปไม่รู้จบ ซึ่งถ้าเปรียบเทียบกับรูปแบบของ ANSI-C ส่วนนี้ก็คือ ฟังก์ชั่น main( ) นั่นเอง

จะเห็นได้ส่วนแรกซึ่งถือเป็นส่วนเริ่มต้นของโปรแกรม ซึ่งจะเรียกว่า Header โดยประกอบด้วยคำสั่ง #include ซึ่งเป็นคำสั่งพิเศษที่เรียกว่า Compiler Directive ซึ่งมันไม่ใช่คำสั่ง สำหรับสั่งงานในโปรแกรม ดังนั้นคำสั่งนี้จึงไม่ต้องมีเครื่องหมายเซมิโคลอนปิดท้ายคำสั่งเหมือนคำสั่งอื่น ๆ โดย Compiler Directive จะใช้ทำหน้าที่สำหรับบอกให้การแปลภาษาซี รับรู้เงื่อนไขในการแปลคำสั่งเท่านั้น ซึ่งในกรณีคำสั่ง #include จะใช้สำหรับบอกให้การแปลภาษาซี รับรู้ว่าในการแปลคำสั่งของโปรแกรมนี้ ไฟล์ภายนอกใดบ้างที่จำเป็นต้องใช้ร่วมในการแปลคำสั่งให้กับโปรแกรมนี้ โดยจากตัวอย่าง ข้างต้นจะเป็นการบอกให้การแปลภาษาซี ทำการผนวกไฟล์ ชื่อ “Servo.h” เข้ามา ใช้เพื่อเรียกใช้คำสั่งต่าง ๆ ที่บรรจุไว้ เข้ามาใช้งานในโปรแกรม โดยใช้รูปแบบดังนี้

```
#include <header.h>
```

โดยเมื่อพบคำสั่ง #include ตัวแปลภาษาของ Arduino จะไปค้นหาไฟล์ที่ระบุไว้ในเครื่องหมาย <> หลังคำสั่ง #include จากตำแหน่ง Directory ที่เก็บรวบรวม Library ของโปรแกรม Arduino ไว้ ซึ่งก็คือ “..\arduino-0012\hardware\libraries\” เช่น เมื่อทำการติดตั้งโปรแกรมของ Arduino ไว้ที่ Directory ที่ชื่อว่า “c:\arduino-0012” ไฟล์ภายนอกที่เป็น Library Function และ Header ต่างๆ จะถูกรวบรวมเก็บไว้ที่ “c:\arduino-0012\hardware\libraries\” เมื่อโปรแกรมพบคำสั่ง #include โปรแกรมของ Arduino จะไปค้นหาไฟล์ต่าง ๆ จากตำแหน่งของ Directory ที่ชื่อ “c:\arauino-0012\hardware\libraies\” นั้นเอง

โดยส่วนของ Header จะนับรวมไปถึง คำสั่งส่วนที่ใช้ประกาศสร้าง ตัวแปร (Variable Declaration) และค่าคงที่(Constant Declaration) รวมทั้ง ฟังก์ชั่นต่าง ๆ (Function Declaration) ด้วย ซึ่งจากตัวอย่างได้แก่ส่วนที่เป็นคำสั่ง

สำหรับส่วนที่มีความสำคัญและจำเป็นที่สุดของโปรแกรม Arduino ที่จำเป็นต้องมีและจะขาดไม่ได้ในการเขียนโปรแกรมของ Arduino คือ ฟังก์ชั่น setup( ) และ ฟังก์ชั่น loop( ) ซึ่งฟังก์ชั่นทั้ง 2 ส่วนนี้มีรูปแบบโครงสร้างที่เหมือนกัน แต่ถูกกำหนดด้วยชื่อของฟังก์ชั่นเป็นการเฉพาะ คือ setup( ) และ loop( ) โดย setup( ) จะเขียนไว้ก่อน loop( ) ซึ่งทั้ง 2 ฟังก์ชั่นนี้ มีขอบเขตเริ่มต้นและสิ้นสุด อยู่ภายใต้เครื่องหมาย { } โดยใช้รูปแบบดังนี้

```
void setup( )
{
.
คำสั่งต่างๆ ที่ต้องการเขียนไว้ภายใต้ฟังก์ชั่น setup( )
.
}
```

หน้าที่ของฟังก์ชั่น setup( ) ใน Arduino คือ ใช้ทำหน้าที่เป็นส่วนของโปรแกรมย่อยสำหรับใช้บรรจุคำสั่งต่าง ๆ ที่ใช้สำหรับกำหนดการทำงานของระบบ หรือ กำหนดคุณสมบัติการทำงานให้กับอุปกรณ์ต่าง ๆ ซึ่งคำสั่งทั้งหมดที่บรรจุไว้ภายใต้ฟังก์ชั่นของ setup( ) นี้ จะถูกเรียกขึ้นมาทำงานเพียงรอบเดียวคือตอนเริ่มต้นการทำงานของโปรแกรม (หลังการรีเซ็ตให้ MCU เริ่มต้นทำงาน) เท่านั้น โดยคำสั่งที่นิยมบรรจุไว้ในฟังก์ชั่นส่วนนี้ ได้แก่ คำสั่งสำหรับกำหนดโหมดการทำงานของ Digital Pin หรือ คำสั่งสำหรับกำหนดคุณสมบัติของพอร์ตสื่อสารอนุกรม เป็นต้น โดยใช้รูปแบบดังนี้

```
void loop( )
{
.
คำสั่งต่างๆ ที่ต้องการให้ทำงานภายใต้ฟังก์ชั่น loop( )
.
}
```

หน้าที่ของฟังก์ชั่น loop( ) ใน Arduino คือใช้ทำหน้าที่เป็นส่วนของโปรแกรมหลัก สำหรับใช้บรรจุคำสั่งควบคุมการทำงานต่างๆของโปรแกรม ที่ต้องการใช้โปรแกรมทำงาน โดยคำสั่งที่บรรจุไว้ในฟังก์ชั่นนี้ จะถูกเรียกขึ้นมาทำงานซ้ำ ๆ กันตามลำดับและเงื่อนไขที่กำหนดไว้

ในส่วนนี้โปรแกรมจะประกอบไปด้วย คำสั่งที่เขียนขึ้นเองส่วนหนึ่งและอีกส่วนหนึ่งเป็นคำสั่งจากภายนอก ที่มีการสร้างและเก็บรวบรวมเป็นไฟล์ ในรูปแบบของ Library Function

ตัวแปรใน Arduino

ตัวแปร หมายถึง กลุ่มของตัวอักษร ตัวเลข และเครื่องหมายใดๆ ที่รวมกันเป็นชื่อ เพื่อใช้กำหนดเป็นตัวแทนของค่าข้อมูลที่เราต้องการจะอ้างถึงในโปรแกรมซึ่งลักษณะของข้อมูล อาจมีทั้งแบบที่เป็นค่าซึ่งสามารถเปลี่ยนแปลงได้ (variable) หรือ อาจเป็นแบบที่มีค่าคงที่ไม่สามารถเปลี่ยนแปลงได้ (constant) ในการประกาศใช้งานตัวแปร จำเป็นต้องประกาศชนิดของตัวแปร หรือบางครั้งอาจมีการกำหนดค่าเริ่มต้นให้กับตัวแปรด้วยได้

การตั้งชื่อตัวแปรของ Arduino จะยึดหลักของ ANSI-C ทุกประการนั่นก็คือ

- ชื่อของตัวแปรต้องประกอบไปด้วยตัวอักษร ตัวเลข และ ยอมให้ใช้เครื่องหมายพิเศษอีก 2 ตัว คือ เครื่องหมาย Under Line (_) และ Dollar Sign ($) มาใช้ในการตั้งชื่อได้ โดยชื่อต้องรียงติดทั้งหมดห้ามมีการเว้นวรรค และ มีความหมายไม่เกิน 32 ตัวอักษร
- ชื่อของตัวแปรที่กำหนด จะใช้เฉพาะตัวอักษรทั้งหมด หรือ ตัวอักษรผสมกับตัวเลข หรืออาจผสมกับเครื่องหมาย _ หรือ $ ด้วยก็ได้ แต่ต้องเริ่มต้นชื่อด้วยตัวอักษรเป็นลำดับแรกเสมอ ห้ามตั้งชื่อโดยเริ่มต้นด้วยตัวเลข หรือ เครื่องหมาย
- ชื่อของตัวแปร ต้องกำหนดด้วยตัวอักษรที่เป็นตัวอักษรพิมพ์เล็กเท่านั้น และในภาษาซีจะถือว่าตัวอักษรที่เป็นตัวอักษรพิมพ์ใหญ่ และ ตัวอักษรพิมพ์เล็ก มีความหมายต่างกัน เช่น “LED” และ “led” และ “Led” ภาษาซี จะถือว่าเป็นคนละชื่อกัน
- ในการตั้งชื่อตัวแปร ห้ามนำคำสงวน (Reserve Word) มาใช้ตั้งชื่อ ซึ่งคำสงวน ได้แก่ ชื่อคำสั่ง และ ชื่อ Internal Function ต่างๆที่สร้างไว้แล้วในตัว ภาษา

คำสงวนต่างๆของ Arduino

- คำสงวนที่เป็น Constant ของ Arduino ได้แก่

HIGH, LOW, INPUT, OUTPUT, SERIAL, DISPLAY, PI, HALF_PI, TWO_PI, LSBFIRST, MSBFIRST, CHANGE, FALLING, RISING, false, true, null

- คำสงวนที่เป็น Port Variables & Constants ของ Arduino ได้แก่

DDRB, PINB, PORTB, PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7, DDRC, PINC, PORTC, PC0, PC1, PC2, PC3, PC4, PC5, PC6, PC7, DDRD, PIND, PORTD, PD0, PD1, PD2, PD3, PD4, PD5, PD6, PD7

- คำสงวนที่เป็น Datatypes ของ Arduino ได้แก่

Boolean, byte, char, class, default, do, double, int, long, private, protected, public, return, short, signed, static, switch, throw, try, unsigned, void

- คำสงวนที่เป็นคำสั่งของ Arduino ได้แก่

Abs, acos, + =, +, [ ], asin, =, atan, atan2, &, |, Boolean, byte, case, ceil, char, class, ,, //, ?:, constrain, cos, {}, --, default, delay, delayMicroseconds, /, /**, ., else, = =, exp, false, float, floor, for, <, <=, if, ++, !=, int, <<, <, <=, log, &&, !, ||, loop, max, millis, min, -, %, /*, *, new, null, (), PI, return, >>, ;, Serial, Setup, sin, sqsqrt, -=, switch, tan, this, true, void, while, begin, read, write, print, println, available digitalWrite, digitalRead, analogRead, analogWrite, attachInterrupts, detachInterrupts, beginSerial, serialWrite, serialRead, serialAvailable, printString, printInteger, printByte, printHex, printOctal, printBinary, printNewline, pulseIn, shiftOut, pinMode

ชนิดและประเภทของตัวแปร

ในภาษาซีนั้น มีการกำหนด และ จำแนก ชนิดของตัวแปร ไว้เป็น 5 ชนิดด้วยกันโดยแต่ละชนิดจะมีคุณสมบัติการใช้งานที่ต่างกัน เพื่อใช้ในการเก็บข้อมูลที่มีรูปแบบแตกต่างกัน คือ

- **char** ใช้เก็บข้อมูลที่เป็นตัวอักษร (character) ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็มได้ 256 ค่า
- **int** ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็ม (integer) ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็มได้ 65536 ค่า
- **float** ใช้เก็บข้อมูลที่เป็นเลขทศนิยมแบบ Single Precision
- **double** ใช้เก็บข้อมูลที่เป็นเลขทศนิยมแบบ Double Precision ซึ่ง สามารถเก็บค่าตัวเลขทศนิยมที่มีความละเอียดและถูกต้องของทศนิยมมากกว่าแบบ float ถึง 2 เท่า
- **void** ใช้เก็บตัวแปรที่ไม่มีค่า

ตารางที่ 2.1 คุณสมบัติของตัวแปรภาษาซี

ชนิดตัวแปร	จำนวนบิต	ค่าข้อมูลที่เก็บได้
Char	8	-128 ถึง +127
Int	16	-32768 ถึง +32767
Float	32	3.4E-38 ถึง 3.4E+38
Double	64	1.7E-308 ถึง 1.7E+308
Void	0	ไม่มีค่า

โดยในการประกาศสร้างตัวแปรขึ้นมาใช้งานของภาษาซีนั้นจะใช้รูปแบบดังนี้

type name

- **type** หมายถึง ชนิดของตัวแปร
- **name** หมายถึง ชื่อของตัวแปร

คุณสมบัติเฉพาะของตัวแปร

สำหรับตัวแปรชนิดที่ใช้เก็บค่าเลขจำนวนเต็ม (**char** และ **int**) นั้น ในภาษาซี ไม่ได้มีการจำแนกชนิดของตัวแปรเพื่อใช้เก็บค่าตัวเลขที่เป็นค่าบวกหรือค่าลบเป็นการเฉพาะ แต่ภาษาซีจะใช้วิธีการเพิ่มคำสั่งสำหรับกำหนดคุณสมบัติเฉพาะให้กับตัวแปรไว้อีก4 คำสั่ง สำหรับใช้กำหนดคุณสมบัติของตัวแปรแบบนี้ให้มีคุณสมบัติที่เฉพาะเจาะจงลงไปอีก โดยในภาษาซีจะมีคำสั่ง ที่ใช้สำหรับระบุคุณสมบัติเฉพาะของตัวแปรที่ใช้เก็บค่าเลขจำนวนเต็ม 4 คำสั่ง คือ

- **unsigned** ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรเฉพาะค่าที่เป็นบวกเท่านั้น
- **signed** ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรทั้งค่าบวกและลบ
- **short** ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรที่มีค่าน้อยกว่า int
- **long** ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรที่มีค่ามากกว่า int เป็น 2 เท่า

ตารางที่ 2.2 คุณสมบัติของตัวแปรแบบต่างๆตามมาตรฐานของ ANSI-C

ชนิดตัวแปร	จำนวนบิต	ค่าข้อมูลที่เก็บได้
Char	8	-128 ถึง +127
signed **char**	8	-128 ถึง +127
unsigned **char**	8	0 ถึง +255
Int	16	-32768 ถึง +32767
signed **int**	16	-32768 ถึง +32767
unsigned **int**	16	0 ถึง 65535
shoat **int**	16	-32768 ถึง +32767
signed short **int**	16	-32768 ถึง +32767
unsigned short **int**	16	0 ถึง 65535
long **int**	32	-2147483648 ถึง +2147483647
signed long **int**	32	-2147483648 ถึง +2147483647
unsigned long **int**	32	0 ถึง +4294967295
Float	32	3.4E-38 ถึง 3.4E+38
Double	64	1.7E-308 ถึง 1.7E+308
long **double**	80	3.4E-4932 ถึง 3.4E+4932

การกำหนดชนิดตัวแปรขึ้นมาใช้งานเอง

นอกจากตัวแปรตามมาตรฐานของ ANSI-C ทั้ง 5 ชนิดที่กล่าวไปแล้วนั้น ภาษาซียังยอมให้ผู้ใช้สามารถกำหนดชนิดตัวแปรขึ้นมาใช้งานได้เองด้วย โดยใช้คำสั่ง typedef โดยมีรูปแบบการใช้งานดังนี้

typedef type name

- type คือ ชื่อของชนิดตัวแปรที่ต้องการสร้างขึ้นใหม่
- name คือ ชื่อที่ใช้เรียกชนิตัวแปรจริง (ตัวแปรมาตรฐาน

สรุปชนิดของตัวแปรใน Arduino ที่ใช้บ่อยๆ

- boolean ใช้เก็บค่าข้อมูลเพียง 2 จำนวน คือTRUE และ FALSE
- char ใช้เก็บค่าข้อมูลขนาด 8 บิต ใช้สำหรับเก็บค่ารหัสของตัวอักษร ซึ่งสามารถกำหนดเป็นค่า หรือ เขียนตัวอักษรไว้ภายใต้เครื่องหมาย ฟันเดี่ยวก็ได้ เช่น 'A' หรือ 0x41 หรือ 65
- byte ใช้เก็บค่าข้อมูลขนาด 8 บิต ที่เป็นค่าจำนวนเต็มแบบไม่คิดเครื่องหมาย(เหมือนกันกับ unsigned char ในภาษาซี) ซึ่งสามารถเก็บค่าข้อมูลได้ 256 ค่า คือ 0-255
- int หรือ integer ใช้เก็บค่าข้อมูลขนาด 16 บิต ที่เป็นค่าจำนวนเต็ม แบบคิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 65536 ค่า คือ -32768 ถึง +32767
- unsigned int ใช้เก็บค่าข้อมูลขนาด 16 บิต ที่เป็นค่าจำนวนเต็ม แบบไม่คิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 65536 ค่า คือ 0-65536
- long ใช้เก็บค่าข้อมูลขนาด 32 บิต ที่เป็นค่าเลขจำนวนเต็มแบบคิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 4294967296ค่า คือ -2,147,483,648 ถึง 2,147,483,647
- unsigned long ใช้เก็บค่าข้อมูลขนาด 32 บิต ที่เป็นค่าเลขจำนวนเต็มแบบไม่คิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 4,294,967,296 ค่า คือ 0ถึง 4,294,967,295
- float ใช้เก็บค่าข้อมูลที่เป็นเลขทศนิยมแบบคิดเครื่องหมายขนาด 32 บิต โดยสามารถเก็บค่าได้ระหว่าง 3.4E-38 ถึง 3.4E+38 (-3.4028235e+38 ถึง 3.4028235E+38)
- double ใช้เก็บค่าข้อมูลที่เป็นเลขทศนิยมเช่นเดียวกันกับ float แต่มีค่าความละเอียดว่า float ถึง 2 เท่า สามารถเก็บค่าได้มากถึง 1.7E+308
- void เป็นตัวแปรแบบที่ไม่มีการเก็บค่าใดๆ คือ ไม่มีค่านั้นเอง
- arrays เป็นตัวแปรที่ใช้เก็บข้อมูลหลายๆค่าไว้ในตัวแปตัวเพียงชื่อเดียวแต่มีตัวเลขสำหรับชี้ตำแหน่งการเก็บข้อมูลต่างกัน โดยตัวเลยที่ใช้ทำหน้าที่เป็นตัวชี้ตำแหน่งของข้อมูล เรียกว่า Index Number โดยค่าลำดับของข้อมูลในตัวแปร array ตำแหน่งแรกจะมีค่าเป็น ศูนย์เสมอ

- **string** เป็นตัวแปรใช้เก็บข้อความ หรือ ตัวอักษรหลายๆตัว ซึ่ง string ก็คือ array ของตัวแปรแบบ char นั่นเอง
- **pointer** เป็นตัวแปรที่ไม่ได้ใช้เก็บข้อมูล แต่ใช้เก็บค่าตำแหน่งแอดเดรสของหน่วยความจำที่ใช้สร้างเป็นตัวแปรสำหรับเก็บข้อมูล ซึ่งตัวแปรแบบนี้จะใช้ทำหน้าที่เป็นตัวชี้ไปยังตำแหน่งแอดเดรสของตัวแปรอื่นๆอีกทีหนึ่ง

การแปลงค่าตัวแปรโดยการ cast

ในภาษาซีนั้น ถ้าหากว่ามีการนำค่าในตัวแปร ที่ต่างชนิดกัน มากระทำกัน จะทำได้ได้ผลลัพธ์ที่ผิดไป ดังนั้นจึงมีความจำเป็นต้อง ทำการแปลงชนิดของตัวแปรให้เป็นชนิดเดียวกันก่อน แล้วจึงนำตัวแปรนั้นกระทำกัน ซึ่งในภาษาซี จะยอมให้มีการสั่งแปลงค่าข้อมูลในตัวแปรจากประเภทหนึ่งไปเป็นอีกประเภทหนึ่งเป็นการชั่วคราว เพื่อนำไปกระทำกัน โดยที่ค่าในตัวแปรต้นแบบไม่ถูกเปลี่ยนแปลงไปด้วย ซึ่งเราเรียกวิธีการแบบนี้ว่า การ cast ซึ่งมีรูปแบบดังนี้

(type)variable

- **type** หมายถึง ชนิดของตัวแปรที่ต้องการเปลี่ยนค่า
- **variable** หมายถึง ตัวแปรที่ต้นแบบที่ต้องการนำมาเปลี่ยนค่า

ตัวแปรแบบ Arduino

ตัวแปรแบบ Array จะมีลักษณะแตกต่างจากตัวแปรแบบอื่น ๆ คือ ตัวแปรแบบอื่น ๆ นั้นเมื่อประกาศขึ้นขึ้นมาใช้งานแล้ว ตัวแปร 1 ตัวก็จะสามารถใช้เก็บข้อมูลได้ 1 ค่าเท่านั้น เมื่อต้องการเก็บข้อมูลหลายๆค่าก็จะใช้ Array เป็นตัวแปรที่ใช้เก็บข้อมูลหลายๆค่าไว้ในตัวแปรเพียงชื่อเดียว แต่มีตัวเลขสำหรับชี้ตำแหน่งการเก็บข้อมูลต่างกัน โดยตัวเลขที่ใช้ทำหน้าที่เป็นตัวชี้ตำแหน่งการเก็บข้อมูลต่างกัน โดยตัวเลขที่ใช้ทำหน้าที่เป็นตัวชี้ตำแหน่งของข้อมูลของตัวแปรแบบ Array เรียกว่า Index Number โดยค่าลำดับของข้อมูลในตัวแปร Array ตำแหน่งแรกจะมีค่าเป็นศูนย์เสมอ ตัวแปรแบบ Array สามารถกำหนดให้ใช้สำหรับเก็บข้อมูลแบบใดก็ได้ เพียงแต่มีข้อแม้ว่า ข้อมูลที่จะเก็บในตัวแปร Array นั้นต้องเป็นแบบเดียวกันทั้งหมด เช่น array ของ char, array ของ int, array ของ float array ของ double เป็นต้น โดยรูปแบบการประกาศสร้างตัวแปร array คือ

```
ชนิดตัวแปร  ชื่อตัวแปร[size];
ชนิดตัวแปร  ชื่อตัวแปร[size1,size2,size3,...,size];
```

- **ชนิดตัวแปร** หมายถึง ชนิดของข้อมูลที่จะเก็บใน Array ของตัวแปร ซึ่งใช้ได้กับตัวแปรทุกประเภทของภาษาซี เช่น char, int, float, double
- **ชื่อตัวแปร [size]** หมายถึง ชื่อของตัวแปร Array ที่ต้องการสร้างขึ้นโดยมีขนาดเท่ากับ m

- **Size** หมายถึง ขนาดของ Array ซึ่งเป็นจำนวนที่จะใช้เก็บข้อมูล โดยการเข้าถึงข้อมูลจะใช้ตัวชี้ Index ซึ่งมีค่าระหว่าง 0 ถึง size-1 ในการชี้ตำแหน่งของข้อมูลใน Array
- **Size1, size2, size3,... sizeN** หมายถึง ขนาดของ Array ในกรณีที่กำหนด Array เป็นหลายมิติ

โดยรูปแบบการจัดเก็บข้อมูลของ Array นั้นจะเก็บแบบเรียงลำดับต่อเนื่องกันไป เป็นชุดๆ เริ่มจากลำดับแรกไปจนถึงลำดับสุดท้าย ตัวอย่างเช่น

ตารางที่ 2.3 การจัดรูปแบบข้องมูลของ Array

msg[0]	msg[1]	msg[2]	msg[3]	msg[4]	msg[5]	msg[6]	msg[7]
A	B	C	D	E	F	G	H

ตารางที่ 2.4 การจัดรูปแบบข้องมูลของ Array แบบ 2 มิติ

	8 หลัก							
4 แถว	R0, C0	R0, C1	R0, C2	R0, C3	R0, C4	R0, C5	R0, C6	R0, C7
	R1, C0	R1, C1	R1, C2	R1, C3	R1, C4	R1, C5	R1, C6	R1, C7
	R2, C0	R2, C1	R2, C2	R2, C3	R2, C4	R2, C5	R2, C6	R2, C7
	R3, C0	R3, C1	R3, C2	R3, C3	R3, C4	R3, C5	R3, C6	R3, C7

ซึ่งในการอ่านข้อมูลทีละตัวจาก Array แบบ 2 มิติ จะใช้รูปแบบดังนี้

```
char buf = msg[4][0];          // อ่านค่าจาก Array msgแถวที่5 ตัวอักษรที่1 (buf = '0')
```

นากจากนี้แล้วยังสามารถใช้วิธีการอ่านตำแหน่งเริ่มต้นของ Array โดยใช้ array pointer ในการชี้ตำแหน่งเริ่มต้นข้อมูลใน Array ได้ด้วย โดยใช้รูปแบบ ดังนี้

```
char* pbuf = msg[0];          // ให้pbuf ชี้ไปที่ตำแหน่งเริ่มต้นของ Array msg[0]
Serial.println(pbuf);         // พิมพ์ค่าข้อความที่ชี้โดย pbuf
```

สำหรับการแก้ไขค่าให้กับ Array นั้นก็จะใช้การอ้างตำแหน่งแถวและหลัก ของ Array ทีละตำแหน่ง เช่นเดียวกันกับการอ่าน เช่น

```
msg[4][0] = '?';          กำหนดค่าให้ Array แถวที่5 ตัวอักษรที่1 = '?'
```

ในการกำหนดให้ Pin เป็น Outputและการสั่งให้ Output Pin เป็นลอจิก 0 หรือ 1 ซึ่งก็ต้องใช้ 1 คำสั่ง ต่อ 1Pin

```
pinMode(2,OUTPUT)
pinMode(3,OUTPUT)
digitalWrite(2,LOW)หรือ digitalWrite(2,HIGH)
digitalWrite(3/LOW) หรือ digitalWrite(3,HIGH)
```

ตัวแปรแบบ String

ตัวแปรแบบ String คือตัวแปรสำหรับใช้เก็บตัวอักษรหลายๆตัว ซึ่งในความเป็นจริงแล้วตัวแปรแบบ String ก็จัดเป็นตัวแปรแบบ Array ประเภทหนึ่ง เพียงแต่เป็น Array ของตัวอักษรแบบ char เท่านั้นเอง ซึ่งข้อแตกต่างของ String กับ Array ก็คือ String จะต้องปิดท้ายข้อมูลด้วยค่าข้อมูลที่เป็นศูนย์เสมอ เพียงแต่ว่าในตอนกำหนดค่าของ String ไม่ต้องกำหนดค่าศูนย์ด้วย โดยภาษาซีจะเติมค่าศูนย์ปิดท้ายให้เองโดยอัตโนมัติ ดังนั้นในการกำหนดขนาด Array ของ String นั้นต้องกำหนดขนาดของ Array ให้มีค่ามากกว่าจำนวนตัวอักษรที่ต้องการจัดเก็บ 1 ค่าเสมอ ซึ่งภาษาซีของ Arduino สามารถประกาศตัวแปร String ได้หลายรูปแบบ
โครงสร้างการจัดเก็บข้อมูลขนาด 8 Byte เป็นดังนี้

ตารางที่ 2.5 แสดงโครงสร้างการจัดเก็บข้อมูลของตัวแปรแบบ String

st [0]	st [1]	st [2]	st [3]	st [4]	st [5]	st [6]	st [7]
A	R	d	u	i	n	o	\0

ตัวแปรแบบ String สามารถตรวจสอบขนาดของตัวแปรได้ โดยใช้คำสั่ง sizeof( ) หรือ strlen( ); โดยถ้าใช้คำสั่ง sizeof( ) ค่าที่ได้จะนับรวม Array ที่เก็บค่าศูนย์ด้วย ถ้าต้องการนับเฉพาะขนาดที่ใช้ เก็บตัวอักษรอย่างเดียวต้องนำค่าที่ได้ลบด้วย 1 เสมอ ส่วน คำสั่ง strlen( ) จะได้ค่าความยาวของ String เฉพาะส่วนที่ใช้เก็บตัวอักษรเท่านั้น ไม่นับรวมตำแหน่งที่ใช้เก็บศูนย์ด้วย

ตัวแปรแบบ Pointer

Pointer เป็นตัวแปรที่ไม่ได้ใช้เก็บข้อมูล แต่ใช้เก็บค่าตำแหน่งแอดเดรสของหน่วยความจำที่ใช้สร้างเป็นตัวแปรสำหรับเก็บข้อมูลแทน โดยตัวแปรแบบนี้จะใช้ทำหน้าที่เป็นตัวชี้ไปยังตำแหน่งแอดเดรสของตัวแปรอื่นๆอีกทีหนึ่ง โดยมีรูปแบบการใช้งานดังนี้

ชนิดตัวแปร *ชื่อตัวแปร

- **ชนิดตัวแปร** หมายถึง ชนิดของตัวแปรที่ต้องการให้ pointer
- ***ชื่อตัวแปร** หมายถึง ชื่อของตัวแปรแบบ pointer ซึ่งกำหนดให้ต้องใช้เครื่องหมาย * นำหน้าชื่อและห้ามเว้นวรรคด้วย

โดยในการใช้งานตัวแปรแบบ pointer นั้น จะต้องเกี่ยวข้องกับเครื่องหมาย 2 อัน คือ * และ & โดยเครื่องหมาย * จะใช้ในการอ้างชื่อตัวแปรแบบ pointer ส่วนเครื่องหมาย &จะใช้ในการกำหนดค่าตำแหน่งแอดเดรสของตัวแปรให้กับ pointer

ขอบเขตของตัวแปร

ภาษาซีได้จำแนกตัวแปร ตามขอบเขตการเรียกใช้งาน ออกเป็นดังนี้

- **local** หรือ **automatic** หมายถึงตัวแปรที่ประกาศขึ้นในส่วนเริ่มต้นของโปรแกรมใน Block {} โดยตัวแปรนี้จะเรียกใช้ได้เฉพาะภายใน Block ที่ประกาศไว้เท่านั้น เมื่อโปรแกรม

จบการทำงานจาก Block หรือออกนอก Block ไป หน่วยความจำที่ใช้สร้างตัวแปรนี้จะถูกยกเลิกไป การประกาศตัวแปรแบบนี้ ควรกำหนดค่าเริ่มต้นให้กับตัวแปรด้วย
เนื่องจาก การแปลภาษาซีจะไม่กำหนดค่าเริ่มต้นให้กับตัวแปร แต่จะยึดถือเอาค่าข้อมูลเดิมที่ค้างอยู่ในความจำที่ถูกนำมาใช้สร้างตัวแปร เป็นค่าเริ่มต้นแทน

- **global หรือ External** หมายถึงตัวแปรที่สามารถใช้งานได้ร่วมกันทุกฟังก์ชั่นในโปรแกรม โดยมีขอบเขตการใช้งานตั้งแต่จุดที่เริ่มต้นประกาศตัวแปร ไปจนสิ้นสุดขอบเขตของโปรแกรม โดยตัวแปรแบบนี้จะประกาศก่อนฟังก์ชั่นหลัก main() เพื่อทำให้ทุกฟังก์ชั่นในโปรแกรมสามารถเรียกใช้งานได้
- **static** มีขอบเขตการใช้งานเฉพาะภายใน Block ที่กำหนดไว้เท่านั้น เพียงแต่มีความพิเศษตรงที่เมื่อโปรแกรมออกนอก Block ไปแล้วค่าของตัวแปรจะยังไม่หมดสภาพไป ค่าต่างๆ ยังคงอยู่และสามารถเรียกใช้งานครั้งต่อไปได้
- **volatile** เป็น keyword ใช้ประกาศเพื่อบอกให้การแปลภาษาซีรู้ว่าตัวแปรนี้ มีการเปลี่ยนแปลงค่าอยู่ตลอดเวลา และทุกๆที่ในโปรแกรม ทั้งจากโปรแกรมหลัก โปรแกรมย่อย หรือ Interrupt เพื่อป้องกันไม่ให้การแปลภาษาซีมาทำการลดขนาด Code (Optimize Code) ส่วนที่จะมากระทำกับตัวแปรที่ประกาศไว้นี้
- **const** ใช้ประกาศเพื่อบอกให้การแปลภาษาซีรู้ว่าตัวแปรที่ประกาศไว้นั้น เป็นค่าคงที่
- **PROGMEM** เป็น Keyword เพื่อบอกให้การแปลภาษาซีรู้ว่าตัวแปรที่ประกาศไว้นี้ต้องการสร้างและเก็บไว้ในหน่วยความจำที่ใช้เก็บโปรแกรมของ MCU (Program Code Memory) ตระกูล AVR ซึ่งใช้กับตัวแปรที่ต้องการสร้างไว้เพื่ออ่านออกมาใช้งานเพียงอย่างเดียว ไม่มีการเปลี่ยนแปลงค่าในขณะที่โปรแกรมทำงานอยู่ ซึ่งควรประกาศเป็น PROGMEM เป็นประหยัดหน่วยความจำ SRAM ในการนำมาใช้สร้างตัวแปร

ค่าคงที่ใน Arduino

ในภาษา Arduino มีการกำหนดค่าตัวแปรที่ (constant) ไว้จำนวนหนึ่งให้ผู้ใช้เรียกใช้เพื่ออำนวยความสะดวกและช่วยให้สามารถอ่านความหมายของคำสั่งในโปรแกรมได้ง่ายและตรงความหมายมากยิ่งขึ้น

- FALSE เป็นค่าคงที่แบบ Boolean มีค่าเป็นศูนย์หรือเท็จ
- TRUE เป็นค่าคงที่แบบ Boolean มีค่าเป็นค่าใดๆที่ไม่ใช่ศูนย์ ซึ่งอาจเป็น 1 หรือ -1 หรือค่าอื่นใดก็ได้ที่ไม่ใช่ศูนย์ จะถือเป็นจริงหรือเท็จทั้งหมด
- HIGH ใช้แทนสถานะลอจิก HIGH หรือ Logic “1”
- LOW ใช้แทนสถานะลอจิก LOW หรือ Logic “0”
- INPUT ใช้ในการกำหนดค่า สถานะ input ให้กับฟังก์ชั่น pinMode()

- OUTPUT ใช้ในการกำหนดค่า สถานะ Output ให้กับฟังก์ชั่น pinMode( )

นอกจากค่าคงที่ในส่วนที่มีการจองหรือกำหนดไว้แล้วจากการแปลภาษาซีใน Arduino ยังมีค่าคงที่อีกแบบหนึ่ง ซึ่งก็คือ ค่าคงที่แบบตัวเลขจำนวนเต็มที่ถูกกำหนดจากผู้ใช้เอง

ซึ่งได้แก่ค่าคงที่อยู่ในรูปของค่าตัวเลขจำนวนเต็มแบบต่าง ๆ แต่เนื่องจากคุณสมบัติของตัวแปรสำหรับใช้ในการเก็บค่าตัวเลขจำนวนเต็มนั้นจะมีอยู่มากมายหลายแบบ เช่น signed, unsigned, short, long ดังนั้นในการกำหนดค่าตัวเลขให้กับการแปลภาษาซีก็ต้องมีการระบุให้ Compiler ทราบด้วยว่าตัวเลขที่กำหนดไปนั้นเป็นค่าตัวเลขแบบใด โดย Arduino กำหนดให้ใช้รหัสตัวอักษรต่อท้ายตัวเลข เพื่อกำหนดค่า คุณสมบัติเฉพาะของตัวเลขตามต้องการ โดยใช้รหัสตัวอักษร U และ L ต่อท้ายตัวเลขเพื่อบงบอกถึงคุณสมบัติเฉพาะของตัวเลขในแบบที่เราต้องการ ซึ่งจะใช้ตัวอักษรตัวเล็ก หรือ ตัวใหญ่ก็ได้มีความหมายเหมือนกัน คือ

- u หรือ U ใช้บอกให้การแปลภาษาซีรู้ว่าตัวเลขที่กำหนดเป็นแบบ unsigned เช่น 33U
- l หรือ L ใช้บอกให้การแปลภาษาซีรู้ว่าตัวเลขที่กำหนดเป็นแบบ Long เช่น 100000L
- ul หรือ UL ใช้บอกให้การแปลภาษาซีรู้ว่าตัวเลขที่กำหนดเป็นแบบ unsigned long เช่น 32767UL

นิพจน์และตัวดำเนินการของ Arduino

นิพจน์ (Expression) คือ ประโยคคำสั่ง ซึ่งถือเป็นองค์ประกอบพื้นฐานที่ใช้สำหรับสร้างประโยคแบบที่เป็นเชิงซ้อนของภาษาซี โดยนิพจน์ใช้สำหรับใช้ในการสั่งให้นำเอาจำนวน 2 จำนวนมากระทำกันโดยมีเครื่องหมาย (Operator) เป็นตัวแสดงการกระทำกันของจำนวนทั้ง 2 โดยค่าของจำนวนทั้ง 2 นั้นอาจเป็น ค่าคงที่ ค่าตัวแปร หรือ ค่าข้อมูลใด ๆ ก็ได้

ตัวดำเนินการ (Operator)

คือ เครื่องหมายใด ๆ ที่ใช้ทำหน้าที่แสดงการกระทำกันของ จำนวน 2 จำนวน ซึ่งอาจเป็นการกระทำกันระหว่างข้อมูลกับข้อมูล ข้อมูลกับตัวแปรหรือตัวแปรกับตัวแปรก็ได้ โดยตัวดำเนินการที่ใช้ในภาษาซี จะมีอยู่ด้วยกันหลายประเภท เพื่อใช้ทำหน้าที่ที่แตกต่างกัน อันได้แก่

- เครื่องหมาย ทางคณิตศาสตร์ (Arithmetic operator)
- เครื่องหมาย การกำหนดค่า (Assignment operator)
- เครื่องหมาย แบบยูนารี (Unary operator)
- เครื่องหมาย เปรียบเทียบ (Comparative operator)
- เครื่องหมาย เชิงตรรก หรือ การกระทำทางโลจิก (Logical operator)
- เครื่องหมาย การกระทำแบบบิต
- เครื่องหมาย การกำหนดเงื่อนไข (Conditional Operator)

เครื่องหมายทางคณิตศาสตร์

+ ใช้แทนการบวก
- ใช้แทนการ ลบ
* ใช้แทนการ คูณ
/ ใช้แทนการการ หาร
% ใช้แทนการ mod หรือการหารแบบเอาเศษที่ได้จากการหารมาใช้

เครื่องหมายสำหรับกำหนดค่า

= ใช้ในการกำหนดค่าให้ตัวแปร
+ + การเพิ่มค่า
- - การลดค่า
+ = การเพิ่มค่าเท่ากับค่าทางขวา
- = การลดค่าเท่ากับค่าทางขวา
* = นำตัวเองคูณกับค่าทางขวา
/ = นำตัวเองหารกับค่าทางขวา
% = นำตัวเอง mod กับค่าทางขวา

ในภาษาซี สามารถกำหนดค่าให้กับตัวแปรได้อีกแบบหนึ่ง ซึ่งเรียกว่าการให้ค่าแบบยูนารี โดยใช้เครื่องหมาย ++ หรือ – ในการกำหนดค่า ตัวอย่างเช่น

x+ + เป็นการเพิ่มค่าให้ i ซึ่งมีความหมายเหมือนกับ x = x+1
x- - เป็นการลดค่าให้ i ซึ่งมีความหมายเหมือนกับ x = x-1

เครื่องหมายที่ใช้ในการเปรียบเทียบ

= = มีความหมาย หมายถึง **เท่ากับ**
! = มีความหมาย หมายถึง **ไม่เท่ากับ**
< มีความหมาย หมายถึง **น้อยกว่า**
> มีความหมาย หมายถึง **มากกว่า**
< = มีความหมาย หมายถึง **น้อยกว่าหรือเท่ากับ**
> = มีความหมาย หมายถึง **มากกว่าหรือเท่ากับ**

เครื่องหมายการกระทำทางโลจิก

&& มีความหมาย หมายถึง Logical AND ซึ่งเปรียบได้กับ **และ**
|| มีความหมาย หมายถึง Logical OR ซึ่งเปรียบได้กับ **หรือ**
! มีความหมาย หมายถึง Logic NOT ซึ่งเปรียบได้กับ **ไม่**

เครื่องหมายการกระทำแบบบิต

& มีความหมาย หมายถึง **Bitwise AND** ซึ่งเป็นการ AND ของบิต

| มีความหมาย หมายถึง **Bitwise OR** ซึ่งเป็นการ OR ของบิต

^ มีความหมาย หมายถึง **Bitwise XOR** ซึ่งเป็นการ XOR ของบิต

~ มีความหมาย หมายถึง **Bitwise NOT** ซึ่งเป็นการ NOT ของบิต

>> มีความหมาย หมายถึง **Bitwise Right** ซึ่งเป็นการเลื่อนบิตไปทางขวา

<< มีความหมาย หมายถึง **Bitwise Left** ซึ่งเป็นการเลื่อนบิตไปทางซ้าย

เครื่องหมายสัญลักษณ์อื่นๆที่ใช้

; เครื่องหมาย เซมิโคลอน (;) ใช้สำหรับบอกการสิ้นสุดประโยคคำสั่ง

{ } เครื่องหมายวงเล็บปีกกา (Curly Braces) ใช้กำหนดขอบเขตของประโยคคำสั่งหรือโปรแกรมย่อยต่างๆ

// เครื่องหมายแสดงจุดเริ่มต้นของคำอธิบายในบรรทัดคำสั่ง (comment) โดยข้อความใดๆที่ตามหลังเครื่องหมายนี้ภายในบรรทัดเดียวกัน จะถือว่าเป็นคำอธิบายและจะไม่ถูกนำไปแปลด้วย

/* */ เครื่องหมายแสดงขอบเขตของการเขียนคำอธิบาย (comment) โดยข้อความใด ๆที่อยู่ระหว่างเครื่องหมาย /* ไปจนถึงเครื่องหมาย */ โปรแกรมจะถือว่าเป็นคำอธิบายทั้งหมด และจะไม่นำมาแปลด้วย ซึ่งการเขียน คำอธิบายแบบนี้จะแตกต่างจากการใช้เครื่องหมาย // ตรงที่ การใช้เครื่องหมาย // จะมีผลเฉพาะบรรทัดนั้นบรรทัดเดียว แต่การใช้เครื่องหมาย /* */ จะสามารถเขียนคำอธิบายจำนวนกี่บรรทัดก็ได้ โดยโปรแกรมจะถือเอาเครื่องหมาย /* เป็นจุดเริ่มต้น ของคำอธิบาย และใช้ เครื่องหมาย */ เป็นจุดสิ้นสุดในการเขียนคำอธิบาย

การตรวจสอบเงื่อนไขในภาษาซีของ Arduino

ในการเขียนโปรแกรมของทุกภาษานั้น มีสิ่งหนึ่งที่จะขาดไม่ได้คือ สร้างเงื่อนไขให้กับโปรแกรมเพื่อให้ได้ผลลัพธ์การทำงานตามที่เราต้องการ ในภาษาซีของ Arduino ก็เช่นเดียวกัน ที่ได้มีการสร้างคำสั่งสำหรับตรวจสอบเงื่อนไขต่างๆ ไว้รองรับการเขียนโปรแกรม

- if
- if...else
- switch case
- while
- do...while
- break
- continue
- goto
- return

โดยคำสั่งสำหรับตรวจสอบเงื่อนไขเหล่านี้จะใช้ควบคู่กับเครื่องหมายเปรียบเทียบและเครื่องหมายทางโลจิก สำหรับใช้กำหนดเงื่อนไขการทำงานให้กับคำสั่ง ซึ่งได้แก่

= = มีความหมายว่า **เท่ากับ**

! = มีความหมายว่า **ไม่เท่ากับ**

< มีความหมายว่า **น้อยกว่า**

> มีความหมายว่า **มากกว่า**

< = มีความหมายว่า **น้อยกว่าหรือเท่ากับ**

> = มีความหมายว่า **มากกว่าหรือเท่ากับ**

&& มีความหมายว่า Logical AND เปรียบได้กับ **และ**

|| มีความหมายว่า Logical OR เปรียบได้กับ **หรือ**

! มีความหมายว่า Logical NOT เปรียบได้กับ **ไม่**

คำสั่ง if

คำสั่ง if เป็นคำสั่งสำหรับใช้ตรวจสอบเงื่อนไข เพื่อสั่งให้โปรแกรมเลือกทำงาน ตามผลลัพธ์ที่ได้จากการตรวจสอบเงื่อนไขของคำสั่ง โดยมีรูปแบบคำสั่งนี้คือ

```
If (เงื่อนไข)
{
คำสั่งที่ต้องกระทำเมื่อเงื่อนไขเป็นจริง
}
```

การทำงานของโปรแกรม เมื่อใช้การตรวจเงื่อนไขแบบนี้ คือ ถ้าเงื่อนไขเป็นจริงก็จะทำงานตามคำสั่งที่อยู่หลังเงื่อนไข แต่ถ้าเงื่อนไขเป็นเท็จก็จะข้ามคำสั่งที่อยู่หลังเงื่อนไขไป

คำสั่ง if...else แบบ 2 ทางเลือก

คำสั่ง if...else เป็นการสั่งตรวจสอบเงื่อนไขเช่นเดียวกัน if แต่ใช้สำหรับการตรวจสอบเงื่อนไขที่มีหลายเพิ่มขึ้นอีก 1 ทางเลือก โดยมีรูปแบบคำสั่งดังนี้

```
If(เงื่อนไข)
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขเป็นจริง
}
else
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขเป็นเท็จ
}
```

ซึ่งจากรูปแบบการใช้คำสั่ง if...else แบบนี้ มีความหมายเหมือนกับประโยคที่ว่า ถ้าเงื่อนไขเป็นจริง ให้ทำอย่างนี้ ไม่เช่นนั้น ให้ทำอย่างนั้น ซึ่งจะเห็นได้ว่า โปรแกรมจะมีทางเลือกในทางทำงานเพิ่มขึ้นมากกว่าการใช้คำสั่ง if อีก 1 ทางเลือก รวมเป็น 2 ทาง โดยทางเลือกแรก เป็นทางเลือกที่โปรแกรมจะทำงานเมื่อเงื่อนไขเป็นจริง ส่วนทางเลือกที่ 2 เป็นทางเลือกที่จะให้โปรแกรมทำงานเมื่อเงื่อนไขเป็นเท็จ

คำสั่ง *if...else* แบบหลายเงื่อนไข

คำสั่ง if...else แบบหลายเงื่อนไข เป็นการสั่งตรวจสอบเงื่อนไขเช่นเดียวกัน if...else แต่ใช้สำหรับการตรวจสอบเงื่อนไขที่มีเงื่อนไขมากกว่า 1 เงื่อนไข โดยมีรูปแบบคำสั่งดังนี้

```
if (เงื่อนไขที่1)
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ 1 เป็นจริง
}
else if (เงื่อนไขที่2)
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ 2 เป็นจริง
}
.
.
else if (เงื่อนไขที่n)
{
  คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขที่ n เป็นจริง
}
else
{
คำสั่งที่ต้องการให้ทำเมื่อเงื่อนไขเป็นเท็จ
}
```

ซึ่งจากรูปแบบการใช้คำสั่ง if...else แบบนี้ มีความหมายเหมือนกับประโยคที่ว่า ถ้าเงื่อนไขที่ 1 เป็นจริง ให้ทำงานที่ 1 ไม่เช่นนั้นให้ตรวจสอบเงื่อนไขที่ 2 และถ้าเงื่อนไขที่ 2 เป็นจริง ให้ทำงานที่ 2 ไม่เช่นนั้นให้ตรวจสอบเงื่อนไขที่ 3 และถ้าเงื่อนไขที่ 3 เป็นจริง ให้ทำงานที่ 3 ไม่เช่นนั้นให้ตรวจสอบเงื่อนไขที่ n และถ้าเงื่อนไขที่ n เป็นจริง ให้ทำงานที่ n

คำสั่ง for

คำสั่ง for เป็นคำสั่งสำหรับสั่งให้โปรแกรมวนรอบทำงาน โดยมีการกำหนดค่าเริ่มต้น และเงื่อนไขการสิ้นสุดที่แน่นอน โดยมีรูปแบบดังนี้

```
for (ค่าเริ่มต้น; เงื่อนไขการวนรอบ; การเพิ่มค่าหรือลดค่าตัวแปรในแต่ละรอบ)
{
คำสั่งที่ต้องการให้ทำ ถ้าเงื่อนไขการวนรอบยังไม่เสร็จ }
```

คำสั่ง switch/case

คำสั่ง swich/case ใช้สำหรับสั่งตรวจสอบเงื่อนไข เพื่อเลือกให้โปรแกรมทำงานตามเงื่อนไขที่ต้องการเพียงเงื่อนไขเดียว ซึ่งผลของการทำงานของคำสั่ง จะเหมือนกันกับ if...else แบบหลายเงื่อนไขเพียงแต่มีรูปแบบการใช้งานที่เป็นระเบียบและใช้งานง่ายกว่า ซึ่งการทำงานของคำสั่งนี้ จะเป็นการนำค่าในตัวแปรที่กำหนดให้ไปทำการเปรียบเทียบกับค่าคงที่ที่กำหนดไว้ ถ้าตรงกันก็จะให้เงื่อนไขเป็นจริงและโปรแกรมจะทำงานตามคำสั่งที่อยู่หลังเงื่อนไขการเปรียบเทียบค่านั้นๆ เมื่อตรวจสอบเสร็จก็จะไปตรวจสอบเงื่อนไขต่อไป จนถึงลำดับสุดท้าย ถ้าไม่ผลการเปรียบเทียบไม่ตรงกับเงื่อนไขใดเลย ก็จะไปทำงานตามคำสั่งที่อยู่ในส่วน default รูปแบบการใช้งานคำสั่ง

```
Switch (ตัวแปรที่จะนำมาตรวจสอบ)
{
  case ค่าคงที่1: คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่1 เป็นจริง
case ค่าคงที่2: คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่1 เป็นจริง

  case ค่าคงที่n : คำสั่งที่ต้องการให้ทำงานเมื่อการตรวจสอบค่าคงที่n เป็นจริง
default : คำสั่งที่ต้องการให้ทำงานเมื่อไม่ตรงกับเงื่อนไขใดๆ
}
```

คำสั่ง while

คำสั่ง while เป็นคำสั่งให้โปรแกรมวนรอบทำงานซ้ำๆกัน (loop) เช่นเดียวกันกับคำสั่ง for แต่มีความแตกต่างกันที่ คำสั่ง for จะมีจำนวนรอบการทำงานที่แน่นอน แต่คำสั่ง while จะทำงานในวงรอบไม่รู้จบจนกว่าเงื่อนไขจะเป็นเท็จจึงจะสิ้นสุดการทำงานในวงรอบ โดยมีรูปแบบดังนี้

```
While (เงื่อนไข)
{
คำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขยังเป็นจริงอยู่
}
```

การทำงานของคำสั่งจะเริ่มต้นด้วยการตรวจสอบเงื่อนไขในวงเล็บก่อน ถ้าพบว่าเงื่อนไขยังเป็นจริงอยู่ ก็จะข้ามไปทำงานตามคำสั่งที่อยู่หลังเงื่อนไข เสร็จแล้วก็จะกลับมาตรวจสอบเงื่อนไขใหม่ โดยการทำงานจะวนเวียนซ้ำ ๆ อยู่เช่นนี้ไปตลอด จนกว่าเงื่อนไขจะเป็นเท็จจึงจะออกจากคำสั่ง

คำสั่ง do...while

คำสั่ง do...while ใช้สำหรับสั่งให้โปรแกรมวนรอบทำงานเหมือนกับ for และ while แต่ลักษณะการทำงานจะแตกต่างกัน โดยคำสั่งนี้จะทำงานตามคำสั่งหลัง do ก่อน แล้วจึงตรวจสอบเงื่อนไข แต่ while จะตรวจสอบเงื่อนไขก่อนแล้วจึงทำงานตามคำสั่งที่อยู่หลังเงื่อนไข โดยถ้าผลการตรวจสอบเงื่อนไขยังเป็นจริงอยู่ก็จะกลับไปเริ่มต้นทำงานตามคำสั่งที่อยู่หลัง do อีก จนกว่าจะพบว่าเงื่อนไขเป็นเท็จจึงจะจบจากคำสั่ง โดยรูปแบบของคำสั่งเป็นดังนี้

```
Do
{
คำสั่งที่ต้องการให้ทำงาน
}while (เงื่อนไข)
```

คำสั่ง break

คำสั่ง break จะใช้ร่วมกับคำสั่งประเภทที่ทำงานแบบวนรอบ เช่น do, for, while เพื่อให้โปรแกรมหยุดการทำงานหรือจบการทำงานจากวงรอบโดยไม่สนใจเงื่อนไข นอกจากนี้แล้วยังใช้คำสั่ง break สำหรับสั่งให้โปรแกรมจบการทำงานของคำสั่ง switch เพื่อข้ามการตรวจสอบเงื่อนไขต่อไปในคำสั่งของ switch

คำสั่ง continue

คำสั่ง continue จะใช้สำหรับสั่งโปรแกรมข้ามการทำงานของคำสั่งที่อยู่ถัดไปจำนวน 1 คำสั่ง แต่ถ้าเขียนคำสั่งนี้ไว้ภายใต้วงรอบการทำงานของคำสั่งที่ทำงานแบบเป็นวงรอบ เช่อdo, for และ while จะเป็นการข้ามการทำงานไป 1 ระดับชั้น ซึ่งเป็นการจบการทำงานจากวงรอบโดยไม่ต้องรอตรวจสอบเงื่อนไข

คำสั่ง goto

คำสั่ง gotoเป็นคำสั่งสำหรับสั่งให้โปรแกรมกระโดไปทำงานยังตำแหน่งต่างในโปรแกรมที่อ้างถึงด้วยตำแหน่ง Label โดยไม่สนใจเงื่อนไข ซึ่งการกระโดดของโปรแกรมสามารถไปได้ทุกทิศทางทั้ง เดินหน้าและถอยหลัง โดยมีรูปแบบคำสั่งการใช้งานของคำสั่งเป็นดังนี้

กรณีการกระโดดถอยหลังกลับ

```
label:
  .
คำสั่งอื่นๆ
.
goto label;
```

กรณีการกระโดดข้ามไปข้างหน้า

```
goto label
  .
คำสั่งอื่นๆ
.
Label:
```

label หมายถึง ตำแหน่ง label ที่จะใช้เป็นจุดอ้างอิงในการให้โปรแกรมกระโดดข้ามไปทำงาน โดยการตั้งชื่อ label จะยึดหลักและข้อกำหนดเช่นเดียวกันกับการตั้งชื่อตัวแปร แต่ใช้เครื่องหมายโคลอน (:) ปิดท้ายชื่อ โดยไม่ถือว่าเครื่องหมายโคลอน เป็นส่วนหนึ่งของชื่อด้วย เป็นเพียงเครื่องหมายสำหรับบอกให้การแปลภาษาซีรับรู้เท่านั้นว่าชื่อที่ประกาศไว้นั้นเป็น Label ไม่ใช่ตัวแปร

คำสั่ง return

คำสั่ง return จะใช้สำหรับจบการทำงานจากโปรแกรมย่อย นอกจากนี้แล้วยังสามารถใช้ในการคืนค่าจากโปรแกรมย่อยกลับไปยังโปรแกรมหลักด้วย

โปรแกรมย่อย (Function)

ในการเขียนโปรแกรมนั้น เมื่อโปรแกรมมีขนาดใหญ่และมีการทำงานที่สลับซับซ้อนมากขึ้น การใช้คำสั่งจำพวกตรวจสอบเงื่อนไขต่างๆดังที่ได้กล่าวมาแล้วนั้นไม่พียงพอในการแก้ปัญหาได้ จึงมีแนวคิดในการแบ่งการทำงานของโปรแกรมออกเป็นส่วนๆโดยแต่ละส่วนก็จะถูกกำหนดให้ทำหน้าที่อย่างใดอย่างหนึ่งจนเสร็จสิ้นหน้าที่ ซึ่งส่วนของโปรแกรมที่ถูกแบ่งแยกออกออกมาเพื่อให้ทำหน้าที่อย่างใดอย่างหนึ่ง นั้นจะถูกเรียกว่า โปรแกรมย่อย ซึ่งในภาษาซีจะเรียกว่า ฟังก์ชั่น (Function)

รูปแบบของฟังก์ชั่น

```
รูปแบบการส่งค่ากลับ ชื่อฟังก์ชั่น(ค่าที่ส่งให้ฟังก์ชั่น1,..2,...n)
{
ชนิดตัวแปร ชื่อตัวแปร
```

```
.
  .
คำสั่งต่างๆ
.
  .
  return(ค่าที่ส่งคืนกลับ)
}
```

- **รูปแบบการส่งค่ากลับ** หมายถึง ชนิดของตัวแปรที่จะใช้ในการส่งผ่านค่าจากโปรแกรมย่อยกลับไปยังโปรแกรมหลักหรือโปรแกรมที่ทำหน้าที่เป็นผู้เรียกใช้โปรแกรมย่อย ถ้าไม่ต้องการส่งค่ากลับ ให้กำหนดรูปแบบการส่งค่ากลับ เป็น void แทน โดยรูปแบบการส่งค่ากลับต้องประกาศไว้หน้าชื่อของฟังก์ชั่นเสมอ
- **ชื่อฟังก์ชั่น** หมายถึง ชื่อของฟังก์ชั่นที่ต้องการตั้งชื่อ ซึ่งใช้ข้อกำหนดและหลักเกณฑ์ในการตั้งชื่อเหมือนการตั้งชื่อตัวแปร
- **ค่าที่ส่งให้ฟังก์ชั่น** หมายถึง ข้อมูลที่ต้องการส่งผ่านไปให้ฟังก์ชั่นทำงาน ซึ่งต้องกำหนดไว้ภายในวงเล็บ ( ) ที่อยู่หลังชื่อฟังก์ชั่น ซึ่งถ้าต้องการผ่านค่าให้กับฟังก์ชั่นมากกว่า 1 ค่า ให้ใช้เครื่องหมาย คอมม่า (,) เป็นตัวแบ่งแยกข้อมูลแต่ละชุดด้วย แต่ถ้าไม่ต้องการให้มีการส่งผ่านค่าไปให้กับฟังก์ชั่น ควรกำหนดค่าเป็น void แทน
- **ค่าที่ส่งคืนกลับ** หมายถึง ค่าข้อมูลที่ต้องการส่งกลับไปยังฟังก์ชั่นผู้เรียก โดยในการส่งค่าคืนกลับไปยังฟังก์ชั่นผู้เรียกนั้นจะใช้คำสั่ง return เป็นคำสั่งในการส่งค่ากลับเสมอ

ชนิดของฟังก์ชั่น

รูปแบบของฟังก์ชั่นที่ใช้ในภาษาซี จะมีอยู่ด้วยกับหลายแบบขึ้นอยู่กับการออกแบบโปรแกรม และความจำเป็นในการใช้งานของโปรแกรม ซึ่งบางโปรแกรมก็มีการส่งผลการทำงานกลับมาให้กับโปรแกรมที่เป็นฝ่ายเรียกใช้ด้วย บางโปรแกรมก็ไม่มีการส่งค่ากลับคืนมา บางโปรแกรมก็มีการส่งผ่านค่าข้อมูลไปให้กับโปรแกรมย่อยเพื่อใช้เป็นค่าการทำงานด้วย ซึ่งสามารถแบ่งออกได้เป็น 2 แบบหลัก ๆ คือ

- ฟังก์ชั่นที่ไม่มีการส่งค่ากลับมายังโปรแกรมผู้เรียก

ฟังก์ชั่นแบบนี้จะมีรูปแบบการส่งค่ากลับ เป็น void ซึ่งเมื่อเรียกใช้ก็จะเรียกเฉพาะชื่อของฟังก์ชั่นเท่านั้น ซึ่งฟังก์ชั่นแบบนี้ ยังมีแบ่งแยกเป็น 2 แบบที่มีการส่งผ่านค่าข้อมูลมาให้กับฟังก์ชั่นและแบบที่ไม่มีการส่งผ่านค่าข้อมูลมาให้กับฟังก์ชั่น

- ฟังก์ชั่นที่มีการส่งค่ากลับคืนมายังโปรแกรมผู้เรียก

ฟังก์ชั่นแบบนี้จะต้องกำหนดรูปแบบการส่งค่ากลับให้กับฟังก์ชั่นด้วย ซึ่งในการเรียกใช้งานโปรแกรมย่อยแบบนี้ โปรแกรมที่เป็นฝ่ายเรียกใช้ต้องสร้างตัวแปร ซึ่งมีชนิดข้อมูลตรงกับรูปแบบการส่งค่า

กลับของฟังก์ชั่นด้วย ซึ่งฟังก์ชั่นแบบนี้ ยังมีแบ่งแยกเป็น 2 แบบ คือ แบบที่มีการส่งผ่านค่าข้อมูลมาให้กับฟังก์ชั่นและแบบที่ไม่มีการส่งผ่านค่าข้อมูลมาให้กับฟังก์ชั่น

- การส่งผ่านค่าระหว่างฟังก์ชั่น

สำหรับการส่งผ่านค่าข้อมูลจากฟังก์ชั่นที่เป็นฝ่ายผู้เรียก ไปให้กับฟังก์ชั่นที่เป็นฝ่ายถูกเรียกใช้ นั้นสามารถใช้ได้กับ ฟังก์ชั่นแบบที่มีการส่งค่ากลับมายังฟังก์ชั่นผู้เรียกหรือฟังก์ชั่นที่ไม่มีการส่งค่ากลับมายังฟังก์ชั่นผู้เรียกก็ได้ โดยวิธีการส่งผ่านค่าให้กับฟังก์ชั่นมี 3 แบบหลัก ๆ คือ

- การส่งผ่านค่าด้วยตัวแปร
- การส่งผ่านค่าด้วย Pointer
- การส่งผ่านค่าด้วย Array
- การส่งผ่านค่าให้ฟังก์ชั่นด้วยตัวแปร

โดยค่าข้อมูลจะถูกส่งผ่านไปให้กับโปรแกรมย่อย ผ่านทางตัวแปร num1 และ num2 ซึ่งเมื่อจบการทำงานของโปรแกรมย่อย sum ก็จะมีการคืนค่าของ num1+num2 กลับคืนมาให้กับโปรแกรมผู้เรียก โดยโปรแกรมย่อยจะใช้การผ่านค่าเป็นเลขจำนวนเต็มแบบ int ไว้รอรับค่าที่จะส่งกลับจากโปรแกรมย่อยด้วย โดยในตัวอย่างจะแสดงให้เห็นวิธีการรับค่าที่ส่งกลับมาจากโปรแกรมย่อย 2 แบบด้วยกันคือ Num0 = Sum(10,5); แบบนี้เป็นการใช้ตัวแปร num0 ในการรับค่าจากโปรแกรมย่อยของ sum โดยตอนเรียกใช้โปรแกรมย่อย sum จะมีการผ่านค่าไปให้ 2 ค่า คือ 10 กับ 5 โดย 10 จะถูกส่งผ่านไปให้ num1 และ 5 จะถูกส่งผ่านไป num2 โดยค่าที่ส่งที่กลับจากโปรแกรมย่อยด้วยคำสั่ง return นั้นจะถูกส่งมาให้กับตัวแปร num0 ของโปรแกรมผู้เรียก จากนั้นจึงสั่งพิมพ์ค่าของ num0 ที่รับกลับมาจากโปรแกรมย่อย sum อีกแบบหนึ่งใช้คำสั่ง Serial.println(Sum(8,2)); ทำการรอรับค่าที่ส่งกลับมาจากโปรแกรมย่อยโดยตรงเลย ซึ่งแบบนี้จะทำให้สามารถเขียนโปรแกรมได้สั้นกว่าแต่ได้ผลการทำงานที่เหมือนกัน

คำสั่ง หรือ ฟังก์ชั่น ของ Arduino

- กลุ่มคำสั่งสำหรับใช้งาน Digital I/O
  - pinMode(pin,mode)
  - digitalWrite(pin,value)
  - intdigitalRead(pin)
- กลุ่มคำสั่งสำหรับใช้งาน Analog I/O
  - intanalogRead(pin)
  - analogWrite(pin,value)

- กลุ่มคำสั่งเพิ่มเติมสำหรับใช้งาน I/O (Advance I/O)
    - Shiftout(dataPin,clockPin,bitOrder,value)
    - Unsigned long pulseIn(pin,value)
- กลุ่มคำสั่งสำหรับหน่วงเวลา(Time)
    - unsigned long millis()
    - delay(ms)
    - delayMicroseconds(us)
- กลุ่มคำสั่งทางคณิตศาสตร์(Math)
    - min(x,y)
    - max(x,y)
    - abs(x)
    - contrain(x,a,b)
    - map(value,fromLow,fromHigh,toLow,toHigh)
    - pow(base,exponent)
    - sqrt(x)
- กลุ่มคำสั่งการคำนวณทางตรีโกณมิติ(Trigonometry)
    - sin(rad)
    - cos(rad)
    - tan(rad)
- กลุ่มคำสั่งการสุ่มค่าจำนวน(Random Number)
    - random(seed)
    - long random(max)
    - long random(min,max)
- กลุ่มคำสั่งการสื่อสารอนุกรม(Serial communication)
    - Serial.begin(speed)
    - intSerial.available()
    - intserial.read()

- Serial.flush()
- Serial.print(data),Serial.print(data,format)
- Serial.println(data),Serial.println(data,format)

กลุ่มคำสั่งเกี่ยวกับ Digital I/O

คำสั่งในกลุ่มนี้เป็นกลุ่มคำสั่ง สำหรับใช้งาน Pin I/O ของ Arduino ในแบบของ Digital I/O ซึ่งตามปกติแล้วในการจะกำหนดหน้าที่ใช้งานขาสัญญาณของไมโครคอนโทรลเลอร์ ซึ่งนิยมเรียกกันว่า Pin I/O นั้น เราจะต้องเข้าไปกำหนดค่าให้กับรีจิสเตอร์ต่างๆในตัว MCU โดยตรง เพื่อเลือกกำหนดรูปแบบการทำงานของขาสัญญาณของ Pin I/O ต่างๆ ให้มีคุณสมบัติตามที่เราต้องการ

สำหรับ Digital I/O Pin ใน Arduino นั้น จะมีจำนวนทั้งหมด 14 Pin โดย Arduino ได้กำหนดรหัสของตัวเลขซึ่งเป็นเลขจำนวนเต็มที่มีค่าระหว่าง 0 ถึง 13 สำหรับอ้างอิง Digital I/O ทั้ง 14 Pin แต่ตามปกติแล้วจะมี Digital I/O Pin จำนวน 2 Pin ซึ่งถูกสงวนไว้สำหรับใช้เป็นพอร์ตสื่อสารอนุกรม RS232 สำหรับใช้ Upload Code ของโปรแกรมให้กับบอร์ด จึงเหลือ Digital I/O Pin สำหรับใช้งานเป็น Digital I/O จริงๆจำนวน 12 Pin คือ Digital I/O หมายเลข 2 ถึง 13 เท่านั้น โดยคำสั่งในกลุ่มนี้จะมีอยู่ด้วยกัน 3 คำสั่ง คือ

- void pinMode(pin,mode)
- void digitalWrite(pin,value)
- int digitalRead(pin)

voidpinMode(pin,mode)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับกำหนดค่าในการทำงานของขา I/O ที่เป็น Digital I/O Pin ของไมโครคอนโทรลเลอร์ตามที่ Arduino กำหนดไว้ ว่าต้องการจะกำหนดใช้งานขา Digital I/O ขาใดเพื่อใช้งานเป็น Input หรือ Output

รูปแบบของคำสั่ง

pinMode(pin,mode)

ค่าพารามิเตอร์ที่ต้องการ

- Pin หมายถึง หมายเลขรหัส Pin ของขาสัญญาณที่ทำหน้าที่เป็น Digital I/O Pin ซึ่งจะมีทั้งหมด จำนวน 14 Pin คือ 0 ถึง 13 โดยต้องกำหนดรูปแบบของตัวเลข ให้เป็นแบบจำนวนเต็มด้วย

- mode หมายถึง หน้าที่การของ Digital I/O Pin ที่ต้องการกำหนด ซึ่งสามารถกำหนดได้ 2 หน้าที่ โดยใช้รหัสข้อความเป็น INPUT กับ OUTPUT

ค่าที่คืนกลับจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

ตัวอย่างการใช้งาน

```
intledPin = 13;                    //ประกาศตัวแปร ledPin แทน Digital Pin-13
intinPin = 7;                      //ประกาศตัวแปร inPin แทน Digital Pin-7
void setup()
{
pinMode(ledPin,OUTPUT);            //กำหนดหน้าที่ของ Digital Pin-13 เป็น output
pinMode(inPin,INTPUT);             //กำหนดหน้าที่ของ Digital Pin-7 เป็น intput
}
void loop()
{
.
}
```

voiddigitalWrite(Pin,value)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับกำหนดสถานะทาง Output ให้กับ Digital I/O Pin ว่าต้องการให้มีสถานะทางลอจิกเป็น High หรือ Low ซึ่งขาสัญญาณที่จะสั่งงานด้วยคำสั่งนี้ จะต้องถูกกำหนดคุณสมบัติให้ทำหน้าที่เป็น Output เรียบร้อยแล้ว

รูปแบบของคำสั่ง

digitalWrite(pin,value)

ค่าพารามิเตอร์ที่ต้องการ

- Pin หมายถึง หมายเลขรหัส Pin ของขาสัญญาณที่ทำหน้าที่เป็น Digital I/O Pin ซึ่งจะมีทั้งหมด จำนวน 14 Pin คือ 0 ถึง 13 โดยต้องกำหนดรูปแบบของตัวเลข ให้เป็นแบบจำนวนเต็มด้วย
- value หมายถึง ค่าสถานะทาง Output ของ Digital Output Pin ที่ต้องการกำหนด ซึ่งสามารถกำหนดค่าสถานะให้กับ Pin ได้ 2 ค่า คือ HIGH และ LOW

ค่าที่คืนกลับจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

ตัวอย่างการใช้งาน

```
intledPin = 13;                    //ประกาศตัวแปร ledPin แทน Digital Pin-13
void setup()
{
pinMode(ledPin,OUTPUT);            //กำหนดหน้าที่ของ Digital Pin-13 เป็น output
}
void loop()
{
digitalWrite(ledPin,HIGH);         //ให้ Pin-13 เป็น High
delay(1000);                       //หน่วงเวลา 1000ms (1mS)
digitalWrite(ledPin,LOW);          //ให้ Pin-13 เป็น Low
delay(1000);                       //หน่วงเวลา 1000ms (1mS)
}
```

intdigitalRead(pin)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ใช้ทำหน้าที่สำหรับอ่านค่าและสถานะ Logic Input ของ Digital Input Pin ว่ามีค่าสถานะเป็น High หรือ Low ซึ่งจาสัญญาณที่จะสั่งอ่านด้วยคำสั่งนี้ จะต้องถูกกำหนดคุณสมบัติให้ทำหน้าที่เป็น Input เรียบร้อยแล้ว

รูปแบบของคำสั่ง

var = digitalRead(pin)

ค่าพารามิเตอร์ที่ต้องการ

- pin หมายถึง หมายเลขรหัส Pin ของขาสัญญาณที่ทำหน้าที่เป็น Digital Input Pin ซึ่งจะมีทั้งหมด จำนวน 14 Pin คือ 0 ถึง 13 โดยต้องกำหนดรูปแบบของตัวเลข ให้เป็นแบบจำนวนเต็ม (int) ด้วย

ค่าที่คืนกลับมาจากฟังก์ชั่น

- var ตัวแปรแบบ int สำหรับใช้รอรับค่าที่ส่งคืนกลับมาจากฟังก์ชั่นซึ่งเป็นค่าสถานะทางลอจิกของ Digital Input Pin ซึ่งมีค่าเป็น High หรือ Low

ตัวอย่างการใช้งาน

```
intinPin = 7;                    //ประกาศตัวแปร inPin แทน Digital Pin7
intval = 0;                      //ประกาศตัวแปร val เป็นแบบ int เพื่อเก็บค่า Digital Input
void setup()
{
pinMode(inPin,INPUT);            //กำหนดหน้าที่ของ Digital Pin7 เป็น Input
}
void loop()
{
val = digitalRead(inPin);        //อ่านค่า Input ของ Digital Pin7 ไว้ใน val
}
```

กลุ่มคำสั่งเกี่ยวกับ Analog I/O

- analogReference(type)
- intanalogRead(pin)
- void analogWrite(pin,value)

analogReference(type)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับเลือกชนิดของแรงดันอ้างอิงของวงจร Analog to Digital หรือ A/D เพื่อใช้งานเป็น Analog Input (analogRead) ซึ่งค่าผลลัพธ์ของ Analog Input จะมีความสัมพันธ์โดยตรงกับแรงดันอ้างอิงของ Analog Input โดยจะอ่านค่าผลลัพธ์ของ Analog Input ได้เป็นค่าสูงสุด 1023 เมื่อแรงดัน Input มีค่าเท่ากับแรงดันอ้างอิงที่กำหนดให้

รูปแบบของคำสั่ง

analogReference(type)

ค่าพารามิเตอร์ที่ต้องการ

- type หมายถึง ชนิดของค่าแรงดันอ้างอิงที่ต้องการกำหนดให้กับ MCU ซึ่ง Arduino สามารถเลือกกำหนดแรงดันอ้างอิงให้กับ Analog Input ได้ 3 แบบด้วยกัน คือ
  - DEFAULT มีค่าแรงดันอ้างอิงเท่ากับ +5
  - INTERNAL มีค่าแรงดันอ้างอิง 1.1V สำหรับ ATmega88 หรือ ATmega168 และมีค่าเป็น 2.5V สำหรับ ATmega8

- EXTERNAL มีค่าแรงดันอ้างอิงตามขนาดแรงดันที่ป้อนให้กับขา AREF จากภายนอก

ค่าที่คืนกลับจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

Unsigned long millis(void)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับอ่านค่าระยะเวลาที่โปรแกรมเริ่มต้นทำงาน โดยให้ค่าออกมาในหน่วยของ milli-Second ซึ่งจะสามารถนับระยะเวลาการทำงานของโปรแกรมได้นานสุดประมาณ 49 วัน 9 ชั่วโมง หลังจากนั้นค่าการนับของเวลาจะวนกลับไปเป็นเริ่มต้นที่ศูนย์ใหม่

รูปแบบคำสั่ง

var = millis()

ค่าพารามิเตอร์ที่ต้องการ

- ไม่มีการส่งค่าใด ๆ ให้กับฟังก์ชั่น

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือตัวแปรแบบ unsigned long สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าจำนวนหน่วยเวลาที่นับได้ตั้งแต่โปรแกรมเริ่มต้นทำงาน โดยมีหน่วยเป็น milli-Second (1/1000 mS)

ตัวอย่างการใช้งาน

```
Long time;                    //สร้างตัวแปรชื่อ time สำหรับเก็บค่าแบบ long
Void setup()
{

Serial.begin(19200);          //กำหนดค่าใช้งานพอร์ตอนุกรมด้วยค่า baud 19200
}
Void loop()
{
Serial.print("Time: ");
  Time = millis();
Serial.println(time);         //พิมพ์ค่าเวลาทั้งหมดตั้งแต่โปรแกรมเริ่มทำงาน
```

```
Delay(1000);                    //หน่วงเวลา 1 mS
}
```

Void delay(ms)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับหน่วงเวลา โดยมีหน่วยเป็น mS (milli-Second) ซึ่งสามารถกำหนดค่าการหน่วงเวลาเป็นค่าตัวเลขจำนวนเต็ม ระหว่าง 1 ถึง 4,294,967,295 mS โดยที่ ค่าเวลาของ 1 mS จะมีค่าเท่ากับ 1/1000 mS

รูปแบบคำสั่ง

delay(ms)

ค่าพารามิเตอร์ที่ต้องการ

- ms คือ ค่าเวลาที่ต้องการหน่วงเวลา มีหน่วยเป็น mS โดยค่าที่กำหนดให้กับคำสั่ง เป็นค่าแบบ unsigned long ซึ่งสามารถกำหนดค่าได้สูงสุด 4,294,967,295 แต่ต้องไม่ลืมว่าการกำหนดค่าตัวเลขของภาษาซีที่ใช้กับ Arduino ตามปกตินั้น ถ้ากำหนดค่าโดยเขียนในรูปแบบของตัวเลขปกติแล้ว โปรแกรมการแปลภาษาซี ของ Arduino จะถือว่าตัวเลขนั้นมีค่าเป็น integer ซึ่งมีค่าไม่เกิน 32767 เท่านั้น ถ้าจะกำหนดค่าการหน่วงเวลาที่มีค่ามากกว่า 32767 ต้องกำหนดรูปแบบตัวเลขให้เป็น unsigned long ด้วย โดยใช้ตัวอักษร 'UL' ตามท้ายค่าตัวเลข ตัวอย่างเช่น ถ้าต้องการกำหนดค่าหน่วงเวลาเป็น 60000ms ก็ให้ใช้รูปแบบการกำหนดค่าตัวเลขดังกล่าวให้เป็นแบบ unsigned long คือ delay(60000UL) เพื่อให้ได้ค่าที่ถูกต้อง

ค่าที่คืนกลับจากฟังก์ชัน

- ไม่มีการส่งค่ากลับ

ตัวอย่างการใช้งาน

```
intledPin = 13;          //ประกาศตัวแปรชื่อ ledPin แทน Digital pin 13
void setup()
{
pinMode(ledPin, OUTPUT);    //กำหนดให้ ledpin เป็น Digital Output
}
void loop()
{
digitalWrite(ledPin,HIGH);       //กำหนดให้ ledPin = HIGH
```

```
 delay1000);                    //หน่วงเวลา 1000ms (1 วินาที)
digitalWrite(ledPin,LOW);       //กำหนดให้ ledPin = low
delay(1000);                    //หน่วงเวลา 1000ms (1 วินาที)
}
```

Void delayMicroseconds(us)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับหน่วงเวลา โดยมีหน่วยเป็น uS (micro-Second) ซึ่งมีค่าการหน่วงเวลาเท่ากับ 1/1,000,000 mS โดยฟังก์ชั่นนี้เหมาะสำหรับใช้สั่งหน่วงเวลาระยะสั้นๆ ในหน่วยของ uS โดยสามารถกำหนดค่าสูงสุดได้ 16383 uS ซึ่งถ้าต้องการหน่วงเวลาเกิน 1000 uS ควรเปลี่ยนไปใช้คำสั่งหน่วงเวลาของ delay(ms) แทนจะเหมาะสมกว่า

รูปแบบคำสั่ง

delayMicroseconds(us)

ค่าพารามิเตอร์ที่ต้องการ

- us คือ ค่าเวลาที่ต้องการหน่วงเวลา มีหน่วยเป็น micro-Second

ค่าที่กลับคืนจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

กลุ่มคำสั่งทางคณิตศาสตร์

คำสั่งในกลุ่มนี้จะเป็นกลุ่มคำสั่งสำหรับใช้ในการคำนวณทางคณิตศาสตร์และตรีโกณมิติ ซึ่งจะใช้สำหรับตรวจสอบค่า เปรียบเทียบค่า รวมทั้งการแปลงค่าตัวแปร เพื่อนำไปใช้งานตามเงื่อนไขที่ต้องการ

- min(x,y)
- max(x,y)
- abs(x)
- contrain(x,a,b)
- map(value,fromLow,fromHigh,toLow,toHigh)
- pow(base,exponent)
- sqrt(x)
- sin(rad)
- cos(rad)

- tan(rad)

Val min(x,y)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับตรวจสอบค่าตัวเลข 2 จำนวน ว่าค่าของตัวเลขชุดใดมีค่าน้อยที่สุด ถ้าพบว่าตัวเลขชุดใดมีค่าน้อยที่สุด จะคืนค่าของตัวเลขนั้นกลับมา

รูปแบบคำสั่ง

var = min(x,y)

ค่าพารามิเตอร์ที่ต้องการ

- x คือ ค่าตัวเลขชุดแรก ที่ต้องการตรวจสอบ
- y คือ ค่าตัวเลขชุดที่ 2 ที่ต้องการตรวจสอบ

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรสำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น คือ ค่าของตัวเลขที่น้อยที่สุด จาก 2 จำนวนที่ส่งให้กับฟังก์ชั่น

ตัวอย่างการใช้งาน

```
sensVal = min(sensVal,100);          //ควบคุมให้ sensValมีค่าไม่เกิน 100
```

จากตัวอย่างข้างต้น จะเป็นเทคนิคของการประยุกต์ใช้คำสั่ง min เพื่อทำการตรวจสอบและปรับค่าในตัวแปร เพื่อไม่ให้ค่าในตัวแปรมีค่าเกินที่เรากำหนดไว้ โดยในตัวอย่างจะทำการปรับค่าในตัวแปรไม่ให้มีค่าเกิน 100 โดยถ้าพบว่าค่าของตัวแปร sensVal มีค่าน้อยกว่า 100 ก็จะนำค่านั้นไปใช้งานเลย แต่ถ้าพบว่าค่าในตัวแปรของ sensVal มีค่าเกิน 100 คำสั่งก็จะทำการปรับค่าในตัวแปรให้มีค่าเป็น 100 ดังนั้นค่าของ sensVal ก็จะมีค่าไม่เกิน 100 เสมอ

Val max(x,y)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับตรวจสอบค่าตัวเลข 2 จำนวน ว่าค่าของตัวเลขชุดใดมีค่ามากที่สุด ถ้าพบว่าตัวเลขชุดใดมีค่ามากที่สุด จะคืนค่าของตัวเลขนั้นกลับมา

รูปแบบคำสั่ง

var = max(x,y)

ค่าพารามิเตอร์ที่ต้องการ

- x คือ ค่าตัวเลขชุดแรก ที่ต้องการตรวจสอบ
- y คือ ค่าตัวเลขชุดที่ 2 ที่ต้องการตรวจสอบ

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรสำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น คือ ค่าของตัวเลขที่มีค่ามากที่สุดจาก 2 จำนวนที่ส่งกลับให้ฟังก์ชั่น

ตัวอย่างการใช้งาน

```
sensVal = max(sensVal,20);    //ควบคุมให้ sensVal มีค่าไม่ต่ำกว่า 20
```

จากตัวอย่างข้างต้น จะเป็นเทคนิคของการประยุกต์ใช้คำสั่ง max เพื่อทำการตรวจสอบและปรับค่าในตัวแปร เพื่อไม่ให้ค่าในตัวแปรมีค่าต่ำกว่าที่เรากำหนดไว้ โดยตัวอย่างจะทำการปรับค่าในตัวแปรไม่ให้มีค่าต่ำกว่า 20 โดยถ้าพบว่าค่าของตัวแปร sensVal มีค่ามากกว่า 20 ก็จะนำค่านั้นไปใช้งานเลย แต่ถ้าพบว่าค่าในตัวแปรของ sensVal มีค่าต่ำกว่า 20 คำสั่งก็จะทำการปรับค่าในตัวแปรให้มีค่าเป็น 20 ดังนั้นค่าของ sensVal ก็จะมีค่าน้อยสุดคือ 20 เสมอ

abs(x)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับคำนวณหาค่าสัมบูรณ์ของตัวแปร ซึ่งก็คือการเปลี่ยนค่าลบ ให้เป็นเลขจำนวนเต็ม

รูปแบบคำสั่ง

```
var = abs(x)
```

ค่าพารามิเตอร์ที่ต้องการ

- x คือ ค่าตัวที่ต้องการตรวจสอบ

ค่าที่คืนกลับจากฟังก์ชั่น

- varคือ ตัวแปรสำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น คือ ค่าสัมบูรณ์ของตัวเลข โดยถ้า x มีค่าเป็น 0 หรือเป็นค่าจำนวนที่เป็นบวก ค่าที่ส่งคืนมาจากฟังก์ชั่น คือ ค่าของ –x

constrain(x,a,b)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับทำหน้าที่ตรวจสอบและปรับค่าตัวเลขใน x ให้มีค่าอยู่ในย่านพิกัดที่กำหนดไว้ในย่านระหว่าง a ถึง b โดยผลการทำงานของคำสั่งจะทำให้ได้ค่าผลลัพธ์เป็นค่าตัวเลขที่มีค่าระหว่าง a ถึง b กลับคืนมา

รูปแบบคำสั่ง

```
var = constrain(x,a,b)
```

ค่าพารามิเตอร์ที่ต้องการ

- x คือ ค่าตัวเลขที่ต้องการตรวจสอบ
- a คือ ค่าพิกัดต่ำสุด
- b คือ ค่าพิกัดสูงสุด

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรสำหรับใช้รับที่ส่งคืนกลับมาจากฟังก์ชั่น คือ ค่าตัวเลขซึ่งจะมีค่าอยู่ในย่านพิกัดที่กำหนดไว้ โดยถ้าพบว่า x มีค่าอยู่ในพิกัดที่กำหนด คือ มีค่าระหว่าง a ถึง b ก็จะคืนค่าของ x กลับออกมาให้โปรแกรมหลักโดยไม่มีการปรับค่าของ x แต่อย่างใด แต่หากพบว่า x มีค่าต่ำกว่าค่าของ a คำสั่งก็จะทำการปรับค่าของ x ให้มีค่าเท่ากับ a แล้วคืนค่านั้นกลับมาให้โปรแกรมหลัก แต่ถ้าพบว่าค่าของ x มีค่ามากกว่า b ก็จะทำการปรับค่าของ x ให้มีค่าเท่ากับ b แล้วคืนค่านั้นกลับมาให้โปรแกรมหลัก

ตัวอย่างการใช้งาน

```
sensVal = constrain(senaVal,20,100);  //ควบคุมให้ sensVal มีค่าระหว่าง 20 ถึง 100
```

จากตัวอย่างข้างต้น จะเป็นเทคนิคของการประยุกต์ใช้คำสั่ง constrain เพื่อทำการตรวจสอบและปรับค่าในตัวแปร เพื่อไม่ให้ค่าในตัวแปรมีค่าอยู่ในช่วงพิกัดที่เรากำหนดไว้ โดยในตัวอย่างจะทำการปรับค่าในตัวแปรไม่ให้มีค่าระหว่าง 20 ถึง 100 โดยถ้าพบว่าค่าของตัวแปร sensVal มีค่าระหว่าง 20 ถึง 100 ก็จะนำค่านั้นไปใช้งานเลย แต่ถ้าพบว่าค่าของตัวแปร sensVal มีค่าต่ำกว่า 20 คำสั่งก็จะทำการปรับค่าในตัวแปรให้มีค่าเป็น 20 และทำนองเดียวกัน ถ้าพบว่าค่าของตัวแปร sensVal มีค่าสูงกว่า 100 ก็จะทำการปรับค่าในตัวแปรให้มีค่าเป็น 100 ดังนั้นค่าของ sensVal ก็จะมีค่าระหว่าง 20 ถึง 100 เสมอ

map(value,fromLow,fromHigh,toLow,toHigh)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้สำหรับทำหน้าที่ปรับค่าในตัวแปร value จากเดิมที่มีค่าระหว่าง fromLow ถึง fromHigh ให้มีค่าระหว่าง toLow ถึง toHigh

รูปแบบคำสั่ง

```
var = map(value,fromLow,fromHigh,toLow,toHigh)
```

ค่าพารามิเตอร์ที่ต้องการ

- value คือ ค่าตัวเลขที่ต้องการปรับค่าด้วยคำสั่ง map
- fromLow คือ ค่าพิกัดต่ำสุดที่เป็นอยู่ในปัจจุบันก่อนปรับค่า

- fromHigh คือ ค่าพิกัดสูงสุดที่เป็นอยู่ในปัจจุบันก่อนปรับค่า
- toLow คือ ค่าพิกัดที่ต่ำสุดที่ต้องการปรับค่าด้วยคำสั่ง map
- toHigh คือ ค่าพิกัดที่สูงสุดที่ต้องการปรับค่าด้วยคำสั่ง map

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรสำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าที่ทำการปรับจากคำสั่ง map เรียบร้อยแล้ว

ตัวอย่างการใช้งาน

```
/*  Map an analog value to 8 bits (0 to 255) */
void setup()
{
}
void loop()
{
intval = analogRead(0);          //อ่านค่า Analog จาก Analog-0 ไว้ในตัวแปร val
val = map(val,0,1023,0,255);  //ปรับค่าของ valจาก 0..1023 ให้เป็น 0..255
analogWrite(9,val);              //เขียนค่าในตัวแปร val ซึ่งมีค่า 0..255 ให้กับ PWM Digital-9
}
```

จากตัวอย่างเป็นการประยุกต์ใช้คำสั่ง map เพื่อปรับค่าในตัวแปร val ซึ่งอ่านมาจากขาสัญญาณของ Analog Input ซึ่งตามปกติจะมีค่าอยู่ระหว่าง 0 ถึง 1023 ให้มีค่าอยู่ระหว่าง 0 ถึง 255

pow(base,exponent)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้สำหรับทำหน้าที่ คำนวณหาค่าการยกกำลังของค่าตัวเลข

รูปแบบคำสั่ง

var = pow(base,exponent)

ค่าพารามิเตอร์ที่ต้องการ

- base คือ ค่าฐานของตัวเลขที่ต้องการคำนวณหาค่ายกกำลัง ซึ่งต้องเป็นค่าแบบ float
- exponent คือ ค่ายกกำลัง ซึ่งเป็นแบบ float

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือตัวแปรสำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าผลลัพธ์ที่ได้จากการคำนวณของคำสั่ง ซึ่งเป็นค่าที่อยู่ในรูปแบบของเลขยกกำลัง (exponential) โดยต้องใช้ตัวแปรแบบ double ในการรับค่าจากฟังก์ชั่น

ตัวอย่างการใช้งาน

```
void setup()
{
Serial.begin(19200)
}
void loop()
{
Serial.print("Value of pow(10,2) = ");
Serial.println((long)pow(10.0,2.0));
Serial.print("Value of pow(5,3) = ");
Serial.println((long)pow(5.0,.30));
While(1);
}
```

ผลลัพธ์

```
Value of pow(10,2) = 100
Value of pow(5,3) = 125
```

จะเห็นได้ว่า pow(10,2) ก็คือค่าของ 10 ยกกำลัง 2 ซึ่งก็คือ 10 คูณ 10 ซึ่งก็จะมีค่าเท่ากับ 100 ส่วนค่าของ pow(5,3) ก็คือค่าของ 5 ยกกำลัง 3 ซึ่งก็คือ 5 คูณ 5 คูณ 5 หรือก็คือ 5 คูณกัน 3 ครั้งนั่นเอง

Sqrt(x)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับคำนวณหาค่า Square Root ของตัวเองที่กำหนดให้ ซึ่งสามารถใช้กับค่าตัวแปรได้ทุกแบบ

รูปแบบคำสั่ง

```
var = sqrt(x)
```

ค่าพารามิเตอร์ที่ต้องการ

- x คือ ค่าตัวเลขที่ต้องการคำนวณหาค่า Square Root

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรแบบ double สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น

sin(rad)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับคำนวณหาค่า sin ของมุมที่กำหนดไว้ใน rad โดยจะให้ค่าผลลัพธ์เป็นค่าตัวเลขแบบ double โดยมีค่าระหว่าง -1 ถึง +1

โดยค่าของมุมที่กำหนดให้กับคำสั่งจะเป็นค่าแบบ เรเดียน (radian) ซึ่งถ้าต้องการกำหนดค่ามุมที่เป็นองศาต้องคูณด้วย (3.14159/180) เสมอ จึงจะได้ค่าที่ถูกต้อง

รูปแบบคำสั่ง

var = sin(rad)

ค่าพารามิเตอร์ที่ต้องการ

- rad คือ ค่ามุมของ radian ที่ต้องการคำนวณหาค่า sin ซึ่งต้องเป็นแบบ float

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรแบบ double สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าผลลัพธ์ของ sin โดยมีค่าระหว่าง -1 ถึง +1

cos(rad)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับคำนวณหาค่า cosine ของมุมที่กำหนดไว้ใน rad โดยจะให้ค่าผลลัพธ์เป็นค่าตัวเลขแบบ double โดยมีค่าระหว่าง -1 ถึง +1

โดยค่าของมุมที่กำหนดให้กับคำสั่งจะเป็นค่าแบบ เรเดียน (radian) ซึ่งถ้าต้องการกำหนดค่ามุมที่เป็นองศาต้องคูณด้วย (3.14159/180) เสมอ จึงจะได้ค่าที่ถูกต้อง

รูปแบบคำสั่ง

var = cos(rad)

ค่าพารามิเตอร์ที่ต้องการ

- rad คือ ค่ามุมของ radian ที่ต้องการคำนวณหาค่า sin ซึ่งต้องเป็นแบบ float

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรแบบ double สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าผลลัพธ์ของ cosin โดยมีค่าระหว่าง -1 ถึง +1

tan(rad)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับคำนวณหาค่า tangent ของมุมที่กำหนดไว้ใน rad โดยจะให้ค่าผลลัพธ์เป็นค่าตัวเลขแบบ double โดยมีค่าระหว่าง -อนันต์ถึง +อนันต์

โดยค่าของมุมที่กำหนดให้กับคำสั่งจะเป็นค่าแบบ เรเดียน (radian) ซึ่งถ้าต้องการกำหนดค่ามุมที่เป็นองศาต้องคูณด้วย (3.14159/180) เสมอ จึงจะได้ค่าที่ถูกต้อง

รูปแบบคำสั่ง

var = tan(rad)

ค่าพารามิเตอร์ที่ต้องการ

- rad คือ ค่ามุมของ radian ที่ต้องการคำนวณหาค่า tangent ซึ่งต้องเป็นแบบ float

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรแบบ double สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าผลลัพธ์ของ tangent โดยมีค่าระหว่าง -อนันต์ถึง +อนันต์

randomSeed(seed)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับกำหนดค่าเริ่มต้นให้กับคำสั่ง random() ซึ่งเป็นการสุ่มหาจำนวน ครั้งในการสุ่มข้อมูลให้กับคำสั่ง random เพื่อใช้สำหรับทำการสุ่มหาค่าจำนวน

รูปแบบคำสั่ง

randomSeed(seed)

ค่าพารามิเตอร์ที่ต้องการ

- seed คือ ค่าตัวเลขที่ต้องการนำไปทำการสุ่มหาค่า ซึ่งสามารถใช้ได้กับตัวเลขแบบ int และ long

ค่าที่คืนกลับจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

Long random(max)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับสุ่มหาค่าจำนวน โดยให้ผลลัพธ์กลับมาเป็นค่าแบบ long โดยมีค่าสูงสุดเป็นเลขจำนวนเต็ม โดยมีค่าสูงสุดเท่ากับ max-1

รูปแบบคำสั่ง

var = random(max)

ค่าพารามิเตอร์ที่ต้องการ

- max คือ ค่าจำนวนสูงสุดที่ต้องการในการสุ่มหาจำนวนของคำสั่ง random

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรแบบ long สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าจำนวนที่ได้จากการสุ่มหาโดยมีค่าสูงสุด max-1

ตัวอย่างการใช้งาน

```
longrandNumber;              //สร้างตัวแปรแบบ long สำหรับใช้เก็บค่าการสุ่มค่า
void setup()
{
Serial.begin(19200);         //กำหนดค่าให้พอร์ตอนุกรมด้วย baudrate 19200
randonSeed(analogRead(0)); //สุ่มหาค่าจำนวนครั้งในการสุ่มค่าของคำสั่ง random
 }
void loop()
{
randNumber = random(300);  //สุ่มพิมพ์ค่าตัวเลขระหว่าง 0 ถึง 299
serial,println(randNumber);
delay(50);                   //หน่วงเวลา 50 ms
}
```

Long random(min,max)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับสุ่มหาค่าจำนวน โดยให้ผลลัพธ์กลับมาเป็นค่าแบบ long โดยค่าสูงสุดเป็นเลขจำนวนเต็ม โดยมีค่าระหว่าง min ถึง max-1

รูปแบบคำสั่ง

var = random(min,max)

ค่าพารามิเตอร์ที่ต้องการ

- min คือ ค่าจำนวนเริ่มต้นที่ต้องการใช้ในการสุ่มหาค่าจำนวนของคำสั่ง random
- max คือ ค่าจำนวนสูงสุดที่ต้องการใช้ในการสุ่มหาค่าจำนวนของคำสั่ง random

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรแบบ long สำหรับใช้ระบบค่าที่ส่งกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าจำนวนที่ได้จากการสุ่มหาโดยมีค่าระหว่าง min ถึง max-1

ตัวอย่างการใช้งาน

```
longrandNumber;                    //สร้างตัวแปรแบบ long สำหรับใช้เก็บค่าการสุ่ม
void setup()
{
Serial.begin(19200);               //กำหนดค่าให้พอร์ตอนุกรมด้วย buadrate 19200
randomSeed(analogRead(0));         //สุ่มหาค่าจำนวนครั้งในการสุ่มของคำสั่ง random
}
void loop()
{
randNumber = random(10,20);        //สุ่มพิมพ์ค่าตัวเลขระหว่าง 10 ถึง 19
Serial.println(randNumber);
delay(50);                         //หน่วงเวลา 50 ms
}
```

กลุ่มคำสั่งเกี่ยวกับการสื่อสารอนุกรม

คำสั่งกลุ่มนี้ใช้สำหรับ การติดต่อสื่อสาร ระหว่าง Arduino กับอุปกรณ์ภายนอก ในรูปแบบของการสื่อสารอนุกรม ซึ่งอาจเป็นเครื่องคอมพิวเตอร์ PC หรืออุปกรณ์ใดก็ได้ โดยเมื่อต้องการรับส่งข้อมูล ด้วยการสื่อสารอนุกรม จะต้องสูญเสียขาสัญญาณไปจำนวน 2 เส้น คือ PD0(Digital-0),PD1(Digital-1) โดย

- PD0(Digital-0) ใช้ทำหน้าที่เป็นขารับสัญญาณข้อมูลอนุกรม (RXD)
- PD1(Digital-1) ใช้ทำหน้าที่เป็นขาส่งสัญญาณข้อมูลอนุกรม (TXD)

โดยเมื่อต้องการจะใช้คำสั่งในกลุ่มของการสื่อสารอนุกรมนี้แล้ว ขาสัญญาณ Digital-0, Digital-1 จะถูกสงวนไว้ใช้สำหรับการสื่อสารอนุกรมเป็นการเฉพาะ และจะไม่สามารถนำขาสัญญาณทั้ง 2 เส้นนี้ ไปใช้งานอย่างอื่นได้อีก และในโปรแกรมก็จะต้องไม่มีการใช้คำสั่งอื่นๆเช่น PinMode,DigitalWrite ที่อ้างถึงขาสัญญาณทั้ง 2 ขานี้ ในส่วนอื่น ๆ ของโปรแกรมด้วย ไม่เช่นนั้นแล้วจะทำให้การทำงานผิดพลาดได้ โดยคำสั่งในกลุ่มของการสื่อสารอนุกรมของ Arduino มีทั้งหมด 8 คำสั่ง หลักๆ คือ

- Serial.begin(speed)
- intSerial.available()
- intSerial.read()
- intSerial.flush()
- intSerial.print(data)
- intSerial.print(data,format)
- intSerial.println(data)
- intSerial.println(data,format)

Serial.begin(int speed)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับเปิดพอร์ตสื่อสารอนุกรมพร้อมทั้งกำหนดค่าความเร็วในการสื่อสาร (Baudrate) ตามที่ผู้ใช้กำหนด ซึ่งต้องการกำหนดค่าความเร็วนี้ ให้ตรงกับความเร็วของอุปกรณ์อีกฝ่ายหนึ่งที่จะทำการสื่อสารด้วย ซึ่งตามปกติก็คือ เครื่องคอมพิวเตอร์ PC โดยค่าความเร็วที่เป็นมาตรฐานที่นิยมใช้งานกันทั่วไปได้แก่ 300, 1200, 2400, 9600, 14400, 19200, 28800, 38400, 57600 และ 11520

รูปแบบคำสั่ง

Serial.begin(int speed)

ค่าพารามิเตอร์ที่ต้องการ

- speed คือ ค่าความเร็วในการสื่อสารข้อมูลของพอร์ตสื่อสารอนุกรม ซึ่งมีค่าเป็นตัวเลขจำนวนเต็ม ค่าที่แนะนำคือ 19200

intSerial.available()

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับตรวจสอบจำนวนข้อมูลที่ได้รับจากพอร์ตสื่อสารอนุกรมและเก็บรอไว้ใน Buffer มีหน่วยเป็น Byte ซึ่งก็คือ จำนวนตัวอักษรที่รับเก็บไว้ใน Buffer ขณะนั้นๆ โดย Arduino จะสร้าง Buffer สำหรับเก็บข้อมูลที่จะรับจากพอร์ตสื่อสารอนุกรม จำนวน 128 Byte โดยเมื่อเก็บข้อมูลเพิ่มเข้าไป 1 Byte ก็จะทำให้พื้นที่เก็บข้อมูลของ Buffer เหลือน้อยลง แต่เมื่อผู้ใช้สั่งอ่านค่าออกไปจาก Buffer พื้นที่ในการเก็บข้อมูลของ Buffer ก็จะเพิ่มขึ้น ถ้าผู้ใช้อ่านข้อมูลออกไปจนหมด พื้นที่ Buffer สำหรับเก็บข้อมูลก็จะว่าง (สามารถรับข้อมูลใหม่เข้ามาได้อีก 128

Byte) แต่ถ้าข้อมูลถูกรับและเก็บไว้จนเต็ม Buffer แล้ว แต่โปรแกรมไม่สามารถอ่านออกไปประมวลผลได้ทัน ก็จะทำให้ข้อมูลที่ส่งเข้ามาใหม่เสียหาย

รูปแบบคำสั่ง

var = Serial.available()

ค่าพารามิเตอร์ที่ต้องการ

- ไม่มีการส่งค่าใดๆกลับฟังก์ชั่น

ค่าที่คืนกลับจากฟังก์ชั่น

- Var คือ ตัวแปรแบบ int สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่า จำนวนข้อมูลที่รับได้จากพอร์ตสื่อสารอนุกรม ซึ่งจะถูกเก็บไว้ใน Buffer โดยสามารถรับและเก็บข้อมูลไว้ได้ 128 Byte ถ้าโปรแกรมไม่สามารถอ่านข้อมูลออกไปประมวลผลได้ทัน จะทำให้ข้อมูลที่ส่งมาเกิดการสูญหายได้ แต่ถ้าไม่มีข้อมูลเก็บไว้ใน Buffer เลย ค่าที่ส่งคืนจากฟังก์ชั่นจะมีค่าเป็น ศูนย์ เสมอ

intSerial.read()

คำสั่งนี้ ใช้ทำหน้าที่สำหรับสั่งอ่านข้อมูลจาก Buffer ของพอร์ตสื่อสารอนุกรม ออกมาประมวลผลโดยเมื่ออ่านข้อมูลออกมา 1 Byte ก็จะทำให้พื้นที่ Buffer ที่ใช้เก็บข้อมูลของพอร์ตสื่อสารอนุกรมว่างลงอีก 1 Byte เสมอ โดยค่าของข้อมูลที่อ่านได้จะเป็นข้อมูลที่ถูกเก็บรอไว้ใน Buffer เป็นลำดับแรก เมื่ออ่านออกไปแล้วข้อมูลถัดไปก็จะถูกเลื่อนตำแหน่งมารอไว้เป็นลำดับแรกแทน โดยปกติค่าของข้อมูลจะมีค่าเป็นเลขจำนวนเต็ม ระหว่าง 0-255 แต่ถ้าไม่มีข้อมูลเก็บไว้ใน Buffer ค่าข้อมูลที่อ่านได้จะมีค่าเป็น -1 แทน

รูปแบบคำสั่ง

var = Serial.read()

ค่าพารามิเตอร์ที่ต้องการ

- ไม่มีการส่าค่าใดๆให้กับฟังก์ชั่น

ค่าที่คืนกลับจากฟังก์ชั่น

- var คือ ตัวแปรแบบ int สำหรับใช้รับค่าที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่า ข้อมูลที่ถูกเก็บรอไว้ใน Buffer เป็นลำดับแรก ปกติจะมีค่าระหว่าง 0-255 และฟังก์ชั่นจะมีค่ากลับมาเป็น -1 กลับไปให้ ถ้าถูกสั่งอ่านข้อมูลในขณะที่ไม่มีข้อมูลเก็บไว้ใน Buffer เลย

Serial.flush()

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับสั่งล้างข้อมูลทั้งหมดใน Buffer ของพอร์ตสื่อสารอนุกรมให้ว่างลง เพื่อรอรับข้อมูลใหม่ที่จะส่งเข้ามาหลังจบการทำงานของคำสั่งนี้เรียบร้อย

รูปแบบคำสั่ง

Serial.flush()

ค่าพารามิเตอร์ที่ต้องการ

- ไม่มีการส่งค่าใดๆให้กับฟังก์ชั่น

ค่าที่คืนกลับจากฟังก์ชั่น

- ไม่มีการส่งค่ากลับ

Serial.print(data)

Serial.print(data,format)

Serial.println(data)

Serial.println(data,format)

หน้าที่การทำงานของคำสั่ง

คำสั่งนี้ ใช้ทำหน้าที่สำหรับสั่งให้ข้อมูลออกไปยังพอร์ตสื่อสารอนุกรม อย่างต่อเนื่องในลักษณะของการพิมพ์ โดยข้อมูลที่จะสั่งพิมพ์ด้วยคำสั่งนี้ สามารถเป็นไปได้ทั้ง ข้อมูล ตัวแปร หรือข้อความต่าง ๆ ที่กำหนดไว้ในพารามิเตอร์ภายในวงเล็บ ( ) ออกทางพอร์ตสื่อสารอนุกรม

รูปแบบคำสั่ง

Serial.print(data)

Serial.print(data,format)

Serial.println(data)

Serial.println(data,format)

ค่าพารามิเตอร์ที่ต้องการ

- data คือข้อมูลที่ต้องการสั่งพิมพ์ อาจเป็นข้อความ ตัวแปร ในรูปแบบต่างๆ ที่ต้องการการสั่งพิมพ์
- format คือ รูปแบบของการแปลงข้อมูล ซึ่งเป็นค่าของ data แล้วแปลงเป็นค่าตัวอักษร (String) ของค่านั้นๆ โดยสามารถกำหนดรูปแบบในการแปลงข้อมูลเป็นค่าตัวเลขในฐานต่างๆ คือ

- BIN เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบของเลขฐานสอง (Binary) เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็น String ของตัวเลข “1001111”
- DEC เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบของเลขฐานสิบ (Decimal) เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็น String ของตัวเลข “79”
- HEX เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบของเลขฐานสิบหก (Hexadecimal) เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็น String ของตัวเลข “4F”
- OCT เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปแบบของเลขฐานแปด (Octal) เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็น String ของตัวเลข “117”
- BYTE เป็นการแปลงค่าข้อมูลเพื่อแสดงผลในรูปของรหัส ASCII เช่น ถ้าข้อมูลมีค่า 79 จะแสดงผลเป็น String ของตัวเลข “O” ซึ่งรหัส ASCII ของตัวอักษรโอ “O” คือ 0x4F หรือ 79

ตัวอย่างการใช้คำสั่งและรูปแบบการสั่งพิมพ์

สำหรับคำสั่งที่ใช้ในการพิมพ์ค่าข้อมูลออกไปยังพอร์ตสื่อสารอนุกรมใน Arduino (Arduino-0012) จะมีอยู่ด้วยกัน 2 กลุ่ม คือกลุ่มคำสั่ง Serial.print และกลุ่มคำสั่ง Serial.println โดยคำสั่งทั้ง 2 กลุ่มนี้จะมีรูปแบบการใช้งานและให้ผลการทำงานที่เหมือนกัน ต่างกันตรงที่คำสั่งในกลุ่ม Serial.println จะมีการเติมค่ารหัสขึ้นบรรทัดใหม่ (0x0D, 0x0A) ปิดท้ายการทำงานของคำสั่งไปด้วย แต่ส่วนอื่นๆเหมือนกันหมด

- Serial.print(b) หรือ Serial.print(b,DEC)
- Serial.println(b) หรือ Serial.println(b,DEC)

เป็นรูปแบบการสั่งพิมพ์ค่าตัวแปรออกทางพอร์ตสื่อสารอนุกรม โดยคำสั่งนี้จะทำการแปลงค่าของตัวเลขในตัวแปร b ไปเป็น String ของตัวเลข แบบเลขฐานสิบ ตัวอย่างเช่น

```
int b = 65;                 //กำหนดให้ b มีค่าเป็น 65 (เลขฐานสิบ)
Serial.print(b);            //จะแสดงผลเป็น ตัวอักษรของ “65”
Serial.print(b,DEC);        //จะแสดงผลเป็น ตัวอักษรของ “65”
Serial.println(b);          //จะแสดงผลเป็น ตัวอักษรของ “65” จะจบด้วย 0x0D, 0x0A
Serial.println(b,DEC);      //จะแสดงผลเป็น ตัวอักษรของ “65” จะจบด้วย 0x0D, 0x0A
```

- Serial.print(b,HEX)

- Serial.println(b,HEX)

เป็นรูปแบบการสั่งพิมพ์ค่าตัวแปรออกทางพอร์ตสื่อสารอนุกรม โดยค่าของตัวแปรจะถูกสั่งแปลงค่าให้เป็นค่า String ของเลขฐานสิบหก (HEX) ก่อน แล้วจึงสั่งพิมพ์ค่าออกไป โดยให้ผลการพิมพ์เป็นค่าของข้อความ String หรือตัวอักษร ที่แสดงค่าตัวแปรเป็นเลขฐานสิบหก (HEX) แล้ว ตัวอย่างเช่น

```
Int b = 65;                    //กำหนดให้ b มีค่าเป็น 65 (เลขฐานสิบ)
Serial.print(b,DEC);           //จะแสดงผลเป็น ตัวอักษรของ “41”
Serial.println(b,DEC);         //จะแสดงผลเป็น ตัวอักษรของ “41” จบด้วย 0x0D, 0x0A
```

- Serial.print(b,OCT)
- Serial.println(b,OCT)

เป็นรูปแบบการสั่งพิมพ์ข้อมูลของตัวแปรออกทางพอร์ตสื่อสารอนุกรม โดยค่าของตัวแปรจะถูกสั่งแปลงค่าให้เป็นค่า String ของเลขฐานแปด (Octal) ก่อน แล้วจึงสั่งพิมพ์ค่าออกไป โดยให้ผลการพิมพ์เป็นค่าของข้อความ String หรือ ตัวอักษร ที่แสดงค่าตัวแปรเป็นเลขฐานแปด (Octal) แล้ว ตัวอย่างเช่น

```
int b = 65;                    //กำหนดให้ b มีค่าเป็น 65 (เลขฐานสิบ)
Serial.print(b,OCT);           //จะแสดงผลเป็น ตัวอักษรของ “101”
Serial.println(b,OCT);         //จะแสดงผลเป็น ตัวอักษรของ “101” จบด้วย 0x0D, 0x0A
```

ผลลัพธ์ที่ได้จะเป็นค่าของ String ของตัวเลข 65 ซึ่งถูกแปลงเป็นค่าตัวเลข ฐานแปด (Octal) แล้ว คือ “101”

- Serial.print(b,BIN)
- Serial.println(b,BIN)

เป็นรูปแบบการสั่งพิมพ์ข้อมูลของตัวแปรออกทางพอร์ตสื่อสารอนุกรม โดยค่าของตัวแปรจะถูกสั่งแปลงค่าให้เป็นค่า String ของเลขฐานสอง (Binary) ก่อน แล้วจึงสั่งพิมพ์ค่าออกไป โดยให้ผลการพิมพ์เป็นค่าของข้อความ String หรือ ตัวอักษร ที่แสดงค่าตัวแปรเป็นเลขฐานสอง (Binary) แล้ว ตัวอย่างเช่น

```
int b = 65;                    //กำหนดให้ b มีค่าเป็น 65 (เลขฐานสิบ)
Serial.print(b,BIN);           //จะแสดงผลเป็น ตัวอักษรของ “1001001”
Serial.println(b,BIN);         //จะแสดงผลเป็น ตัวอักษรของ “1001001” จบด้วย
0x0D,0x0A
```

- Serial.print(b,BYTE)
- Serial.println(b,BYTE)

เป็นรูปแบบการสั่งพิมพ์ข้อมูลของตัวแปรออกทางพอร์ตสื่อสารอนุกรม โดยค่าของตัวแปรจะถูกสั่งแปลงค่าให้เป็นค่าของรหัส ASCII ก่อน แล้วจึงสั่งพิมพ์ค่าออกไป โดยให้ผลการพิมพ์เป็นค่าของข้อความ String หรือ ตัวอักษรที่เป็นค่ารหัส ASCII ของค่าในตัวแปรนั้น ตัวอย่างเช่น

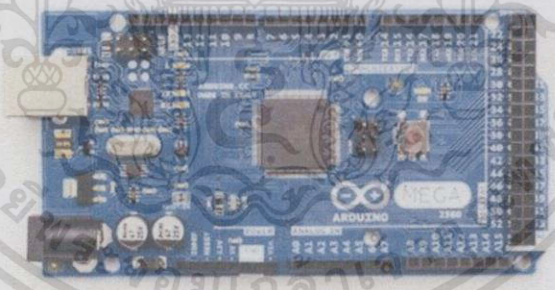
```
int b = 65;                 //กำหนดให้ b มีค่าเป็น 65 (เลขฐานสิบ)
Serial.print(b,BYTE);       //จะแสดงผลเป็น ตัวอักษรของ "A"
Serial.println(b,BYTE);     //จะแสดงผลเป็น ตัวอักษรของ "A" จบด้วย 0x0D,0x0A
```

- Serial.print("string");
- Serial.println("string");

เป็นรูปแบบการสั่งพิมพ์ข้อความออกทางพอร์ตสื่อสารอนุกรม โดยให้ผลการพิมพ์เป็นข้อความเหมือนกับที่กำหนดให้กับคำสั่ง ตัวอย่างเช่น

```
Serial.print("Hello Arduino");     //จะแสดงผลเป็นข้อความ "Hello Arduino"
Serial.println("Hello Arduino");   //จะแสดงผลเป็นข้อความ "Hello Arduino" จบด้วย
                                   //รหัส 0x0D, 0x0A (ขึ้นบรรทัดใหม่)
```

2.1.2 ไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3



รูปที่ 2.1 บอร์ด Arduino Mega 2560 R3

ตารางที่ 2.6 แสดงคุณสมบัติของบอร์ด Arduino Mega 2560 R3

Specification	Value
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
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

คุณสมบัติของบอร์ด Arduino Mega 2560 R3

กำลังไฟ

Arduino Mega 2560 R3 สามารถรับไฟผ่านการเชื่อมต่อ USB หรือมีการจ่ายไฟเลี้ยงจากแหล่งจ่ายพลังงานภายนอก (ไม่มี USB) สามารถมาใช้พลังงานทั้งจากไฟ AC-DC Adapter หรือแบตเตอรี่ ซึ่ง Adapter ที่สามารถเชื่อมต่อได้โดยการเสียบปลั๊ก มีขนาดศูนย์กลาง 2.1 มม. บอร์ดนี้สามารถทำงานได้บนแหล่งจ่ายไฟจากภายนอก ที่ 6V - 20V แต่ หากมีการใช้มากกว่า 12V ตัวควบคุมแรงดันไฟฟ้าที่อาจได้รับความร้อนสูงเกินไปและทำให้เกิดความเสียหายกับบอร์ด ช่วงที่แนะนำคือ 7V–12V กำลังไฟของแต่ละ Pin เป็นดังนี้

- Vin. แรงดันไฟที่ต่อเข้ากับบอร์ด Arduino โดยแหล่งจ่ายพลังงานภายนอก 5V จากการเชื่อมต่อ USB หรืออื่น ๆ หรือหากใช้แรงดันไฟฟ้าโดยผ่านทางช่องเสียบไฟที่สามารถใช้ได้ ก็สามารถจ่ายแรงดันไฟฟ้าได้โดยผ่าน Pin นี้
- 5V. Pin นี้ Output5V แรงดันไฟฟ้ากระแสผ่านพิน 5V หรือ 3.3V
- 3.3V. สร้างขึ้นจากตัวควบคุมบนบอร์ดใช้กระแสไฟสูงสุด 50 mA
- GND. ขา GROUND

- IOREF. Pin นี้ ระดับแรงดันไฟฟ้าจะใช้แหล่งจ่ายพลังงานที่เหมาะสมหรือใช้แรงดันไฟฟ้าเอาต์พุตสำหรับการทำงานร่วมกับ 5V หรือ 3.3V

หน่วยความจำ

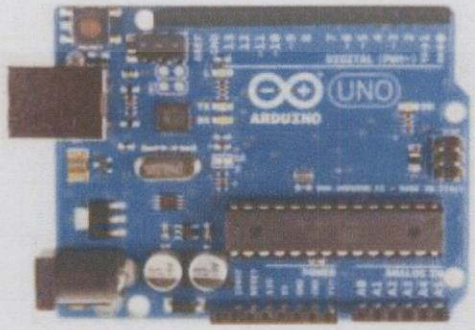
บอร์ด Arduino Mega 2560 R3 มีหน่วยความจำขนาด 256KB (โดย8KB คือใช้สำหรับ bootloader) นอกจากนี้ยังมี 2KB ของ SRAM และ 1 KB ของ EEPROM

INPUT/OUTPUT

มี 54 Pin แต่ละตัวสามารถใช้เป็นอินพุตหรือเอาต์พุต โดยสามารถใช้ฟังก์ชั่น pinMode() digitalwrite() และ digitalread() โดยจะทำงานที่ 5V ซึ่งในแต่ละ Pin สามารถรับสูงสุด 40 mA

- Serial : 0(RX) และ 1(TX), Serial 1: 19(RX) และ 18(TX), Serial 2: 17(RX) และ 16(TX), Serial 3: 15 (RX) และ 14 (TX) ใช้ในการได้รับ (RX) และส่งข้อมูล (Tx) ข้อมูลแบบอนุกรม (TTL)
- External Interrupts : 2(interrupt 0), 3(interrupt 1), 18(interrupt 5), 19 (interrupt 4), 20(interrupt 3), and 21(interrupt 2) โดย PIN เหล่านี้ใช้ในการกำหนดค่าข้อมูล
- PWM: 2-13 และ 44-46 กำหนดเป็น 8-bit PWM output
- SPI : 50(MISO), 51(MOSI), 52(SCK), 53(SS) เป็น Pin ที่เอาไว้สำหรับเชื่อมต่อ (SPI communication)
- LED: 13 มีการสร้าง LED ภายในตัว ที่มีการเชื่อต่อกับ Pin 13
- TWI: 20(SDA) and 21(SCL) ใช้สำหรับเชื่อมต่อ (TWI communication)
- AREF เป็นPIN ที่ใช้สำหรับแรงดันไฟฟ้า Input แบบ Analog
- RESET เป็น PIN ที่ใช้สำหรับ Reset บอร์ดไมโครคอนโทรลเลอร์

2.1.3 ไมโครคอนโทรลเลอร์ Arduino Uno R3



รูปที่ 2.2 บอร์ด Arduino Uno R3

ตารางที่ 2.7 แสดงคุณสมบัติของบอร์ด Arduino Uno R3

Specification	Value
Microcontroller	ATmega328
Operating	5V
Input Voltage	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash 50 mA	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

คุณสมบัติของบอร์ด Arduino Uno R3

กำลังไฟ

Arduino Uno R3 สามารถรับไฟผ่านการเชื่อมต่อ USB หรือมีการจ่ายไฟเลี้ยงจากแหล่งจ่ายพลังงานภายนอก (ไม่มี USB) สามารถมาใช้พลังงานทั้งจากไฟ AC-DC Adapter หรือแบตเตอรี่ ซึ่ง Adapter ที่สามารถเชื่อมต่อได้โดยการเสียบปลั๊ก มีขนาดศูนย์กลาง 2.1 mm. บอร์ดนี้สามารถทำงานได้บนแหล่งจ่ายไฟจากภายนอก ที่ 6V - 20V แต่ หากมีการใช้มากกว่า 12V ตัวควบคุมแรงดันไฟฟ้าที่อาจได้รับความร้อนสูงเกินไปและทำให้เกิดความเสียหายกับบอร์ด ช่วงที่แนะนำคือ 7V–12V กำลังไฟของแต่ละ Pin เป็นดังนี้

- Vin. แรงดันไฟที่ต่อเข้ากับบอร์ด Arduino โดยแหล่งจ่ายพลังงานภายนอก 5V จากการเชื่อมต่อ USB หรืออื่นๆ หรือหากใช้แรงดันไฟฟ้าโดยผ่านทางช่องเสียบไฟที่สามารถใช้ได้ ก็สามารถจ่ายแรงดันไฟฟ้าได้โดยผ่าน Pin นี้
- 5V. Pin นี้ Output 5V แรงดันไฟฟ้ากระแสผ่านพิน 5V หรือ 3.3V
- 3.3V. สร้างขึ้นจากตัวควบคุมบนบอร์ดใช้กระแสไฟสูงสุด 50 mA
- GND. ขา GROUND

- IOREF. Pin นี้ ระดับแรงดันไฟฟ้าจะใช้แหล่งจ่ายพลังงานที่เหมาะสมหรือใช้แรงดันไฟฟ้า Output สำหรับการทำงานร่วมกับ 5V หรือ 3.3V

หน่วยความจำ

บอร์ด Arduino Uno R3 มีหน่วยความจำขนาด 32KB (โดย0.5KB ใช้สำหรับ bootloader) นอกจากนี้ยังมี 2KB ของ SRAM และ 1 KB ของ EEPROM

INPUT/OUTPUT

มี 14 Pin แต่ละตัวสามารถใช้เป็น Input/Output โดยสามารถใช้ฟังก์ชั่น pinMode() digitalwrite() และ digitalread() โดยจะทำงานที่ 5V ซึ่งในแต่ละ Pin สามารถรับสูงสุด 40 mA นอกจากนี้ขาบางส่วนมีฟังก์ชันพิเศษ

- Serial: 0(RX) and 1(TX) ใช้ในการได้รับ (RX) และส่งข้อมูล (Tx) ข้อมูลแบบอนุกรม (TTL)
- External Interrupts Pin2 และ Pin3 โดย Pin เหล่านี้ใช้ในการกำหนดค่าข้อมูล
- PWM: 3, 5, 6, 9, 10 และ 11 กำหนดเป็น 8-bit PWM Output
- SPI : 10(SS), 11(MOSI), 12(MISO), 13(SCK) เป็น PIN ที่เอาไว้สำหรับเชื่อมต่อ (SPI communication)
- LED: 13 มีการสร้าง LED ภายในตัว ที่มีการเชื่อต่อกับ Pin 13
- TWI : A4 หรือ SDA pin และ A5 หรือ SCL Pin ใช้สำหรับเชื่อมต่อ (TWI communication)
- AREF เป็น Pin ที่ใช้สำหรับแรงดันไฟฟ้า Input แบบ Analog
- RESET เป็น Pin ที่ใช้สำหรับ Reset บอร์ดไมโครคอนโทรลเลอร์

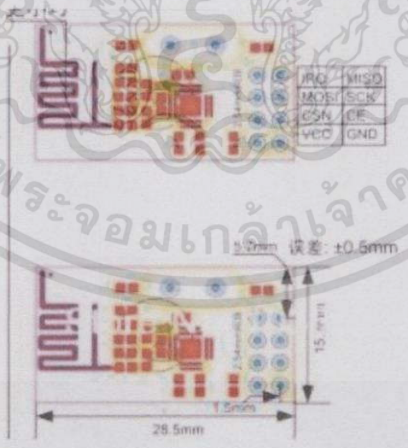
2.2 โมดูลรับ-ส่งสัญญาณไร้สาย nRF24L01

nRF24L01 (2.4GHz Wireless Transceiver Module) เป็นโมดูลรับ-ส่งสัญญาณแบบไร้สาย ซึ่งอยู่ในย่านความถี่ 2.4 GHz โดยโมดูลของ nRF24L01 ประกอบด้วยชิปตัวเดียว ที่มีความถี่ 2.4 GHz และทำหน้าที่ในรับ-ส่งสัญญาณแบบเบสแบนด์ nRF24L01 ได้รับการออกแบบเป็นพิเศษสำหรับการใช้งานแบบไร้สายที่ใช้พลังงานต่ำ โดยได้ออกแบบสำหรับใช้งานสำหรับความถี่แบบ ISM โดยอยู่ในย่านความถี่ 2.400 - 2.4835 และยังมี MCU (ไมโครคอนโทรลเลอร์) เป็นส่วนประกอบแบบ external passive อีกทั้งโมดูล nRF24L01 จะต้องมีการตั้งค่าและใช้งานผ่าน Serial Peripheral Interface (SPI) โดย Interface(SPI) ประกอบด้วย SCK, MISO, MOSI และองค์ประกอบ CSN ในโหมด Stand-by

คุณลักษณะที่สำคัญ

1. ย่านความถี่ 2.4 GHz ทั่วโลก ใช้เปิดวง ISM ซึ่งเป็นแบบใบอนุญาตฟรี
2. มีอัตราสูงสุด 2Mbps, ป้องกันการรบกวน, การปรับ GFSK มีประสิทธิภาพ, มีความเหมาะสมสำหรับการใช้งานการควบคุมอุตสาหกรรม
3. มี 125 Channel ที่จะตอบสนองการสื่อสารหลายจุดและความถี่กระโดด
4. มีการ Modulation แบบ GSFK
5. มี ซีอาร์ซี ฮาร์ดแวร์ใช้ในการตรวจสอบข้อผิดพลาด และทำการควบคุมที่อยู่ในการติดต่อสื่อสาร
6. แรงดันต่ำ 1.9 ~ 3.6V ในทำงานใช้พลังงานที่ต่ำ ภายใต้โหมด สถานะ 7Ma พลังงานลดลงเหลือเพียง 1UA
7. เสาอากาศ PCB 2.4 GHz ขนาดเล็ก ประมาณ 24mm X 24mm (ไม่รวม เสาอากาศ)
8. โมดูลซอฟต์แวร์ที่สามารถตั้งค่าที่อยู่, ที่อยู่เมื่อเครื่องจะได้รับข้อมูล SCM สามารถเชื่อมต่อโดยตรง โดยใช้ความหลากหลาของการเขียนโปรแกรมซอฟแวร์จะสะดวกมาก
9. โมดูลนี้ต้องใช้แหล่งจ่ายไฟ 3.3V, ควบคุมแรงดันไฟฟ้า, ชิปมีแรงดัน 5V, 3.3V-3.3V
10. สามารถรับ-ส่งสัญญาณ ได้ระยะทางสูงสุด 80 m

Interface circuit



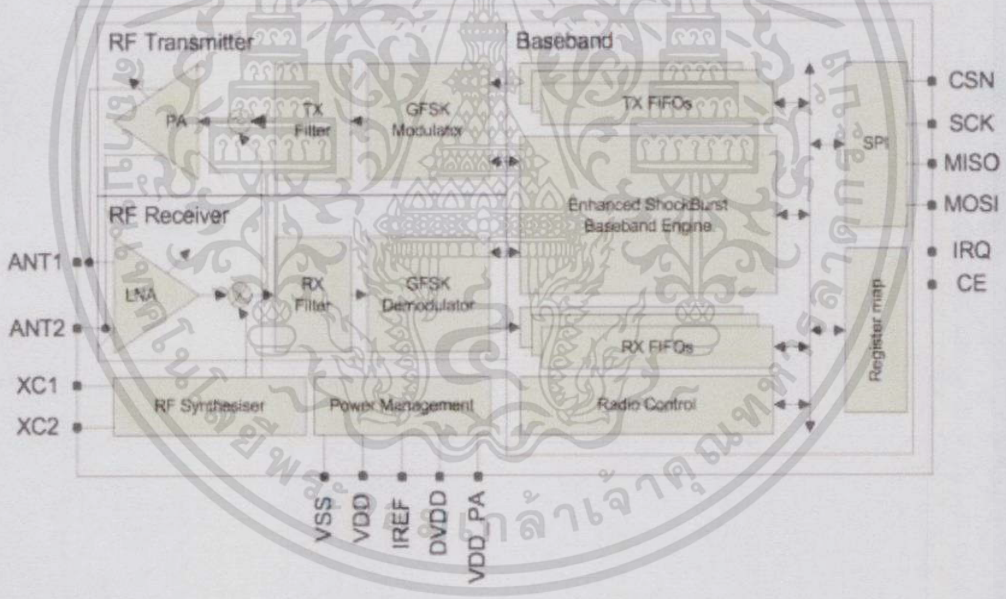
รูปที่ 2.3 ภาพแสดง Interface circuit ของโมดูล nRF24L01

Pin definition and Rating

ตารางที่ 2.8 การกำหนด PIN ของโมดูล nRF24L01

1 – GND	2 – VCC
3 – CE	4 – CSN
5 – SCK	6 – MOSI
7 – MISO	9 – IRQ

Block Diagram

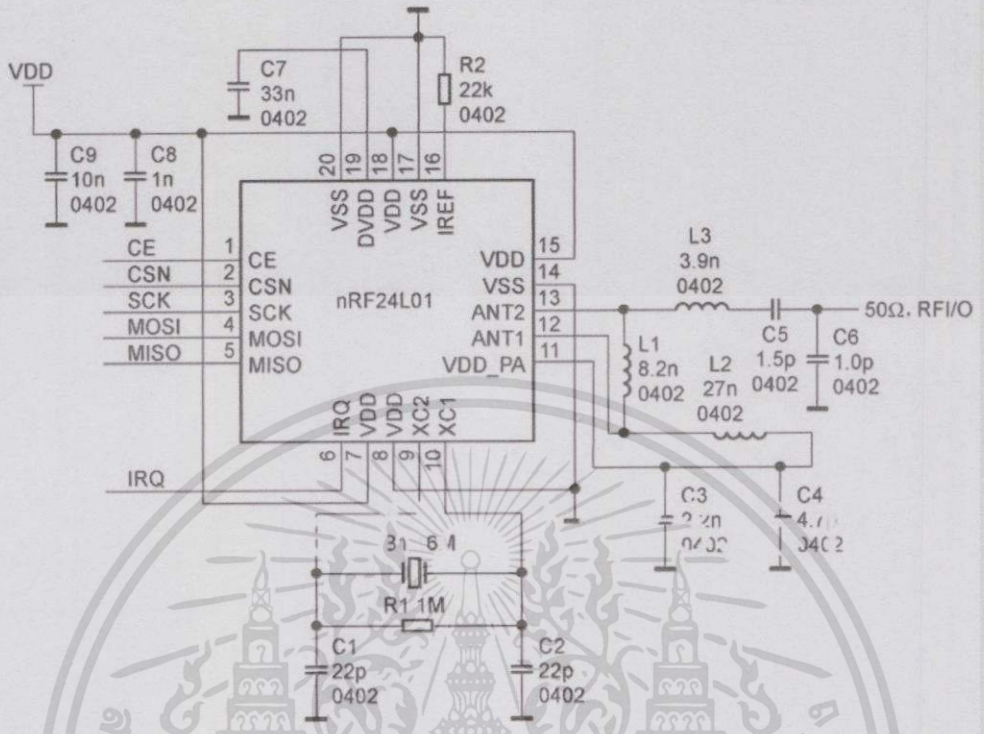


รูปที่ 2.4 ภาพ Block Diagramของโมดูล nRF24L01**Specification**

ตารางที่ 2.9 แสดงคุณสมบัติของโมดูล nRF24L01

Specification	Value
PCB Size	15mm*29mm*0.8mm
Power supply	1.9V~3.6V
Working current	13.5mA at 2Mbps / 11.3mA at 0dBm Output power
IO counts	8
Sensitivity	-85dBm at 1Mbps
Emission distance	70~100 meter at 256kbps
Data rate	256kbps / 1Mbps / 2Mbps
Commuinitcation mode	Enhanced ShockBurst TM / ShockBurst TM
Working mode	Power Down Mode / Standby Mode / RX Mode / TX Mode
Temperatures	Operating :-40°C ~ 85°C / Storage:-40°C ~ 125°C

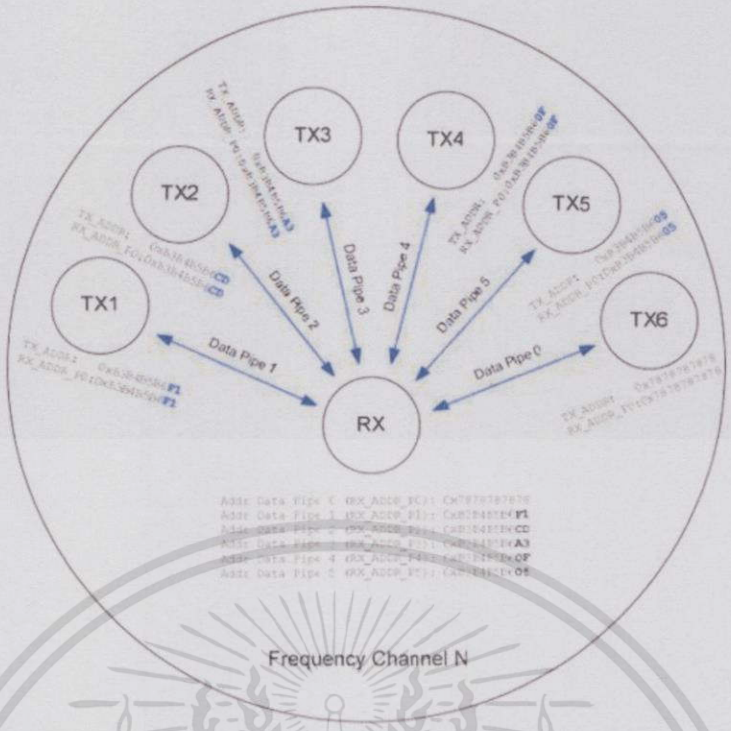
Schematic Diagram



รูปที่ 2.5 ภาพแสดง Schematic Diagram ของโมดูล nRF24L01

nRF24L01 in a star network configuration

nRF24L01 จะมีกำหนดค่าเป็น RX(PRx) เพื่อที่จะสามารถเอาไว้รับข้อมูลผ่าน 6 pipes ที่แตกต่างกัน โดยข้อมูลที่ส่งมาจะมีแอดเดรสที่ไม่ซ้ำกันแต่ใช้ช่องสัญญาณความถี่เดียวกัน ซึ่งหมายความว่า จะได้ถึง 6 ข้อมูลที่แตกต่างกัน และ nRF24L01 จะกำหนดค่าเป็น TX(PTx) ที่จะสามารถสื่อสารกับ RX(PRx) และ nRF24 L01 ที่กำหนดค่าเป็น RX(PRx) สามารถแยกแยะข้อมูลที่มีแอดเดรสข้อมูล 40 บิต ซึ่งมีได้ไม่ซ้ำกัน โดยข้อมูล pipes ในแต่ละห้องที่มีจะไม่ซ้ำกัน 8 บิตและมากที่สุด 32 บิตแอดเดรสที่สำคัญทำให้ pipes ข้อมูลทั้งหมดจะสามารถทำการรับ-ส่งข้อมูลได้อย่างเต็มประสิทธิภาพการทำงาน (shockburst™) และ nRF24L01 เมื่อรับทราบข้อมูลที่จะได้รับแล้ว ก็จะทำหน้าที่ส่ง ACK พร้อมด้วยแอดเดรสเดียวกันกับที่ได้รับมา และ TX(PTx) ที่ pipes ข้อมูล 0 จะใช้ในการรับ Acknowledge

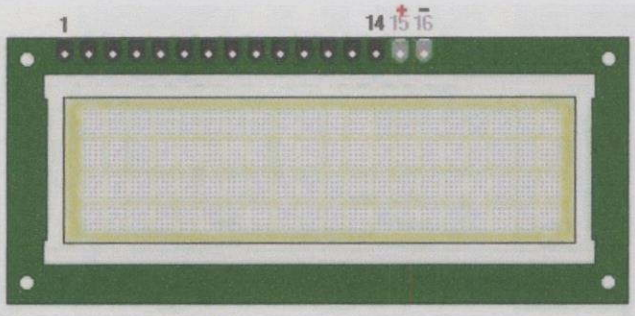


รูปที่ 2.6 ภาพแสดงการรับ-ส่งสัญญาณไร้สายของ nRF24L01 ใน star network

2.3 จอแสดงผล LCD

1. ลักษณะและตำแหน่งของขา LCD โมดูลแต่ละแบบ

รูปที่ 2.7 จอแสดงผล ขนาดLCD 16x2



รูปที่ 2.8 จอแสดงผล ขนาดLCD 20x4

2. คำสั่งควบคุมการแสดงผลของ LCD โมดูล

รายละเอียดคำสั่ง

1. เคลียร์การแสดงผล (Clear Display)

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

รูปที่ 2.9 แสดงการเคลียร์การแสดงผล (Clear Display)

RS = 0, R/W = 0

- bit0 = 1 เคลียร์การแสดงผล
- เคอร์เซอร์กลับไปอยู่ที่มุมซ้ายมือสุด

2. Home Display

7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	-

รูปที่ 2.10 แสดง Home Display

RS = 0, R/W = 0

- bit1 = 1 เคอร์เซอร์กลับไปอยู่ที่มุมซ้ายมือสุด
- ข้อมูลไม่มีการเปลี่ยนแปลง

3. โหมดการป้อนข้อมูล (Entry Mode Set)

7	6	5	4	3	2	1	0
0	0	0	0	0	1	M	S

รูปที่ 2.11 แสดงโหมดการป้อนข้อมูล (Entry Mode Set)

RS = 0, R/W = 0

- bit2 = 1
- M (Address Increase/Decrease), M = 0 ลดตำแหน่งแอดเดรส, M = 1 เพิ่มตำแหน่งแอดเดรส

- S (Shift bit) การเลื่อนข้อมูล S = 0 เคอร์เซอร์จะเลื่อนไปทางขวา S = 1 เคอร์เซอร์จะอยู่กับที่

4.การควบคุมการแสดงผล (Display/Cursor)

7	6	5	4	3	2	1	0
0	0	0	0	1	D	C	B

รูปที่ 2.12 แสดงการควบคุมการแสดงผล (Display/Cursor)

RS = 0, R/W = 0

- bit3 = 1
- D (Display ON/OFF) D = 0 OFF, D = 1 ON
- C (Cursor ON/OFF) C = 0 OFF, C = 1 ON
- B (Blinking Cursor ON/OFF) B = 0 OFF, B = 1 ON

5.การควบคุมการเลื่อนเคอร์เซอร์ (Cursor or Display Shift)

7	6	5	4	3	2	1	0
0	0	0	1	C	M	-	-

รูปที่ 2.13 แสดงการควบคุมการเลื่อนเคอร์เซอร์ (Cursor or Display Shift)

RS = 0, R/W = 0bit

- 4 = 1
- C (Cursor or Display Shift) C = 0 shift cursor, C = 1 shift display
- M (Move left/right) M = 0 left, M = 1 right

6.ฟังก์ชั่นเซ็ท (Function Set)

7	6	5	4	3	2	1	0
0	0	1	D	N	F	-	-

รูปที่ 2.14 แสดงฟังก์ชั่นเซ็ท (Function Set)

RS = 0, R/W = 0

- bit5 = 1
- D (Data bus size) D = 0 is 4-bits, D = 1 is 8-bits
- N (lines No.)N = 0 is 1-line, N = 1 is 2-lines
- F (font size) F = 0 is 5x7, F = 1 is 5x10

7.เซ็ทตำแหน่งใน CG-RAM (Set CG-RAM Address)

7	6	5	4	3	2	1	0
0	1	A5	A4	A3	A2	A1	A0

รูปที่ 2.15 แสดงการเซ็ทตำแหน่งใน CG-RAM (Set CG-RAM Address)

RS = 0, R/W = 0

- bit6 = 1
- หน่วยความจำชั่วคราว เก็บข้อมูลตัวอักษร CG-RAM (Character Generator RAM)
- A0-A5 เป็นตำแหน่งแอดเดรสใน CG-RAM

8.เซ็ทตำแหน่งใน DD-RAM (Set DD-RAM Address)

7	6	5	4	3	□	1	0
1	A6	A5	A4	A3	A2	A1	A0

รูปที่ 2.16 แสดงการเซ็ทตำแหน่งใน DD-RAM (Set DD-RAM Address)

RS = 0, R/W = 0

- bit7 = 1
- หน่วยความจำชั่วคราว เก็บข้อมูลการแสดง DD-RAM (Display Data RAM)
- A0-A6 เป็นตำแหน่งแอดเดรสใน DD-RAM ซึ่งจะถูกคัดลอกไปยัง Address Counter (AC)

DD-RAM คือหน่วยความจำที่เก็บข้อมูลการแสดงผล หากเขียนรหัส ASCII ลงในหน่วยความจำนี้ก็จะปรากฏที่จอ LCD ทันที ตำแหน่ง Address ของ LCD แต่ละแบบ

Line 1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

รูปที่ 2.17 ภาพแสดงตำแหน่ง Address ของ LCD16x2

Line 1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
Line 3	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
Line 4	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D

รูปที่ 2.18 ภาพแสดงตำแหน่ง Address ของ LCD20x4

เช่นกรณีใช้งาน LCD โมดูลแบบ 2x16 บรรทัดที่ 2 ใน column แรกจะมีค่า address = 64 (dec) หรือ 40H คำสั่งที่ใช้ใน

การเขียนข้อมูลออกไปยังโมดูล LCD บรรทัดต่าง ๆ คือ การนำ 10000000 หรือ 80H มา OR กับ address ของ DDRAM

เช่น 80H OR 40H =0C0H จะเป็นชุดคำสั่งที่ใช้เขียนไปยังโมดูล LCD บรรทัดที่ 2

9. การอ่าน BUSY Flag and Address Counter (BF and AC)

7	6	5	4	3	2	1	0
BF	A6	A5	A4	A3	A2	A1	A0

รูปที่ 2.19 แสดงการอ่าน BUSY Flag and Address Counter (BF and AC)

RS = 0, R/W = 1

- BF = bit7 เป็นตัวบอกสถานะของ LCD
- R/W = 1 กำหนดให้เป็น Read mode
- BF = 0 ว่าง, BF = 1 ไม่ว่าง
- A0-A6 = Address Counter (AC)

10. การเขียนข้อมูลใน CG or DD-RAM

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

รูปที่ 2.20 การเขียนข้อมูลใน CG or DD-RAM

RS = 1, R/W = 0

- RS = 1 กำหนดให้เป็นข้อมูล
- R/W = 0 กำหนดให้เป็น Write mode
- D0-D7 = ข้อมูลที่ต้องการเขียน

หากต้องการเขียนข้อมูลใน CG-RAM ให้เซ็ทตำแหน่ง CG-RAM ในข้อที่ 7 ก่อน

หากต้องการเขียนข้อมูลใน DD-RAM ให้เซ็ทตำแหน่ง DD-RAM ในข้อที่ 8 ก่อน

11. การอ่านข้อมูลจาก CG or DD-RAM

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

รูปที่ 2.21 แสดงการอ่านข้อมูลจาก CG or DD-RAM

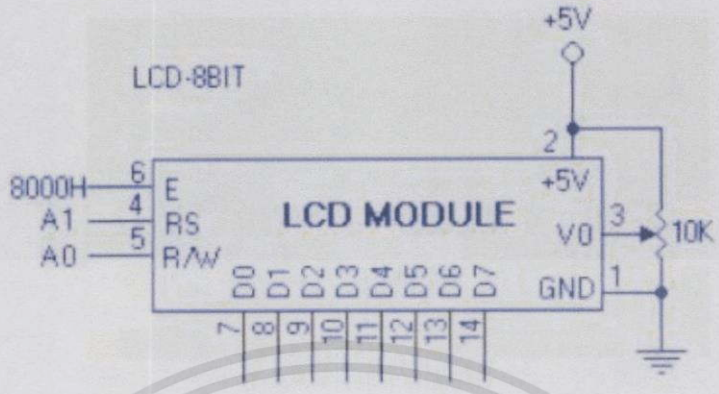
RS = 1, R/W = 0

- RS = 1 กำหนดให้เป็นข้อมูล
- R/W = 1 กำหนดให้เป็น Read mode
- D0-D7 = ข้อมูลที่อ่านได้

หากต้องการอ่านข้อมูลใน CG-RAM ให้เซ็ทตำแหน่ง CG-RAM ในข้อ 7 ก่อน

หากต้องการอ่านข้อมูลใน DD-RAM ให้เซ็ทตำแหน่ง DD-RAM ในข้อ 8 ก่อน

การเขียนโปรแกรมเพื่อควบคุม LCD



รูปที่ 2.22 โครงสร้างของ LCD Module

ขั้นตอนการเขียนโปรแกรมเพื่อใช้งาน LCD สามารถอธิบายได้ดังนี้

1. หากเป็นการเริ่มจ่ายไฟที่ระดับแรงดันถึง 4.5V ให้ LCD ให้รออย่างน้อย 15ms เพื่อให้ LCD Reset ตัวเอง (Internal Reset)

2. Set ค่าเริ่มต้นต่าง ๆ เพื่อให้ LCD เริ่มทำงานตามที่เราต้องการ

กำหนดขาควบคุม

- ให้ขา E = 1
- ให้ขา RS = 0 กำหนดเป็นคำสั่ง
- ให้ขา R/W = 0 เขียนคำสั่ง

ส่งข้อมูลค่าสั่ง 4 ครั้ง

2.1 Function Set (Instruction = 00111000B)

DL = 1 8 bit

N = 1 2 บรรทัด

F = 0 5x7 dot

2.2 Display ON/OFF (Instruction = 00001110B)

D = 1 Display ON

C = 1 Cursor ON

B = 0 Blink OFF

2.3 Entry Mode Set (Instruction = 00000110B)

M = 1 เพิ่มค่า DDRAM address ขึ้น

S = 0 No scroll

2.4 Display clear (Instruction = 00000001B)

3. Set DDRAM Address เพื่อเลือกตำแหน่งในการ Display

กำหนดขาควบคุม

- ให้ขา E = 1
- ให้ขา RS = 0 กำหนดเป็นคำสั่ง
- ให้ขา R/W = 0 เขียนคำสั่ง

เช่น

10000000B = ตำแหน่งแรก แถวที่ 1 (10000000 OR 00000000)

11000000B = ตำแหน่งแรก แถวที่ 2 (10000000 OR 01000000)

40H = 01000000B

4. เขียนตัวอักษรที่ต้องการแสดงไปยัง DDRAM

กำหนดขาควบคุม

- ให้ขา E = 1
- ให้ขา RS = 1 กำหนดเป็นข้อมูล
- ให้ขา R/W = 0 เขียนข้อมูล

หมายเหตุ หลังจากการปฏิบัติการในแต่ละขั้นตอนต้องมีการตรวจสอบว่า LCD Module พร้อมที่จะรับคำสั่งต่อไปหรือไม่

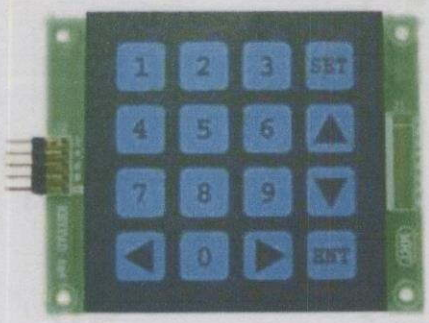
โดยการตรวจสอบ Busy Flag (Bit 7) ใน Instruction Register โดย

กำหนดขาควบคุม

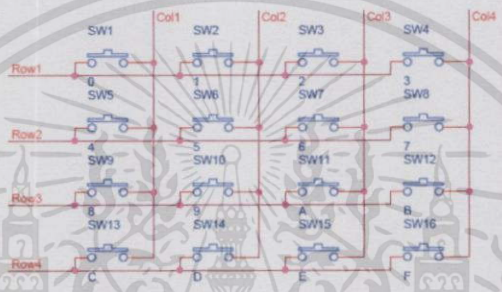
- ให้ขา E = 1
- ให้ขา RS = 0 กำหนดเป็นคำสั่ง
- ให้ขา R/W = 1 อ่านคำสั่ง

2.4 ปุ่มกด (Keypad)

การใช้งานไมโครคอนโทรลเลอร์ส่วนมากจะมีการใช้สวิตช์ เพื่อให้ไมโครคอนโทรลเลอร์ทำงานตามต้องการถ้าใช้งานสวิตช์จำนวนไม่มากนักก็สามารถต่อโดยตรงกับไมโครคอนโทรลเลอร์ได้ แต่ถ้าใช้งานสวิตช์เป็นจำนวนมากก็จะทำให้เกิดความยุ่งยากในการต่อการใช้งานดังนั้นผู้ใช้งานส่วนใหญ่จึงนำสวิตช์หลาย ๆ ตัวมาต่อกันแบบเมตริกซึ่งก็คือ ปุ่มกด นั่นเอง



รูปที่ 2.23 คีย์แพด (Keypad) ขนาด 4x4



รูปที่ 2.24 การนำสวิตช์หลายตัวมาต่อกันเป็นปุ่มกด ขนาด 4x4

ในการใช้งานปุ่มกด ส่วนมากนั้นจะต้องมีอุปกรณ์ในการแสดงผลหรือค่าข้อมูลที่คีย์บนปุ่มกดซึ่งอุปกรณ์แสดงผลส่วนใหญ่จะเป็น 7-Segment และจอแสดงผล LCD ในการเขียนโปรแกรมเพื่อใช้งานปุ่มกดโดยส่วนมากจะใช้ปุ่มกดขนาด 4x4 เป็นมาตรฐานในการอ้างอิงค่าข้อมูล ซึ่งก่อนอื่นจะต้องกำหนดให้ตำแหน่งต่างๆของแต่ละคีย์เป็นค่าตัวเลขของตำแหน่งอ้างอิงก่อน

1	2	3	SET
4	5	6	
7	8	9	
	0		ENT

(ก) ปุ่มกดขนาด 4x4

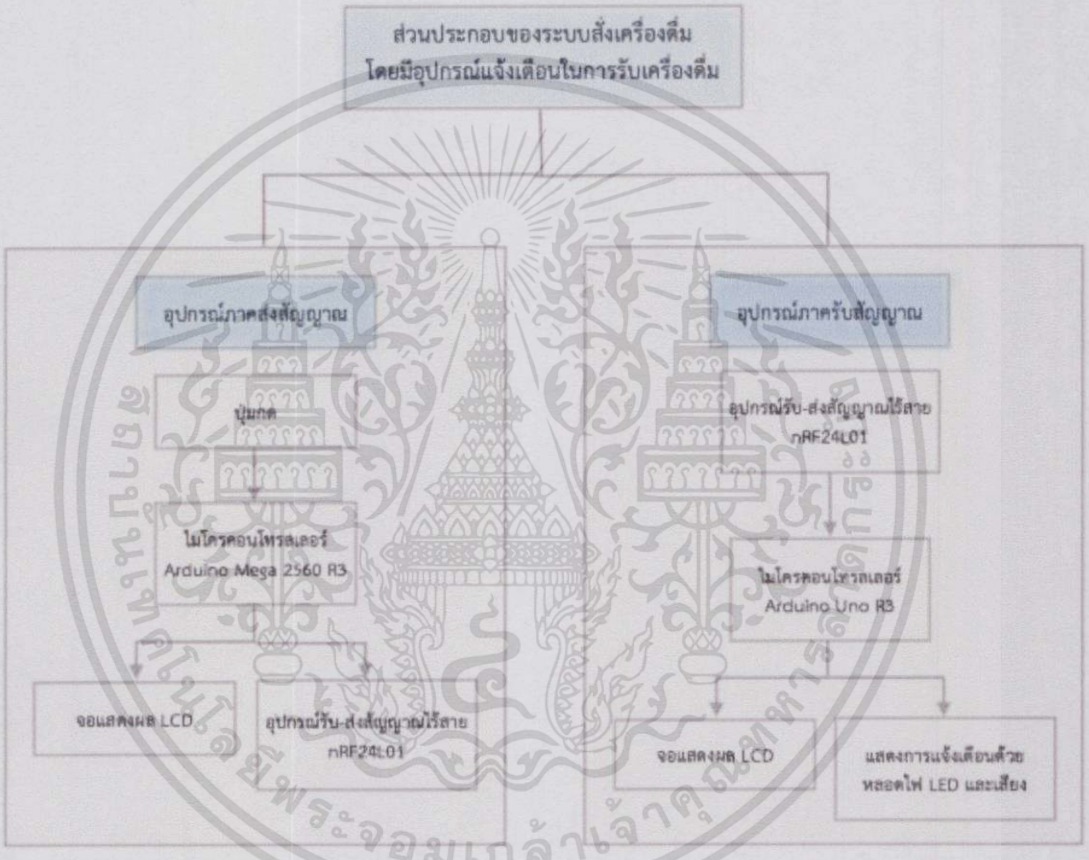
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(ข) ตำแหน่งของอาร์เรย์

รูปที่ 2.25 ความสัมพันธ์ระหว่างปุ่มกด 4x4 กับตัวเลขตำแหน่งอ้างอิง

บทที่ 3
การออกแบบและโครงสร้างของระบบ

ในหัวข้อนี้กล่าวถึงการออกแบบภาพรวมการทำงานของระบบ ซึ่งจะมีทั้งการออกแบบในส่วนฮาร์ดแวร์และซอฟต์แวร์ รวมถึงรายละเอียดแต่ละส่วนของการทำงาน และโครงสร้างของระบบ



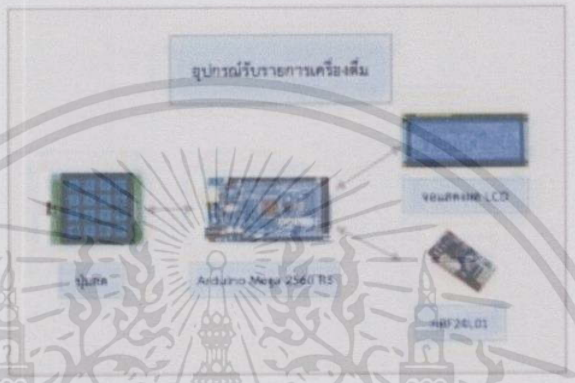
รูปที่ 3.1 ส่วนประกอบของระบบสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

3.1 การออกแบบอุปกรณ์ฮาร์ดแวร์

จากรูปที่ 3.1 ระบบสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม ประกอบไปด้วย 2 ส่วน คือ อุปกรณ์ภาคส่งสัญญาณซึ่งจะใช้งานอยู่ที่เคาน์เตอร์โดยพนักงาน และอุปกรณ์ภาครับสัญญาณจะใช้งานโดยลูกค้า รายละเอียดในแต่ละส่วนจะอยู่ในหัวข้อถัดไป

3.1.1 การออกแบบภาคส่งสัญญาณ

ภาคส่งสัญญาณหรืออุปกรณ์รับรายการเครื่องดื่มจะรับคำสั่งจากปุ่มกด แล้วส่งต่อไปยังไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 เพื่อประมวลผลคำสั่งที่ได้รับมา จากนั้นจะมีการแสดงผลออกทางจอแสดงผล LCD ขนาด 20x4 รวมถึงการส่งข้อมูลไปยังอุปกรณ์ภาครับสัญญาณจะเป็นการส่งข้อมูลแบบไร้สายในย่านความถี่ 2.4 GHz โดยใช้ nRF24L01 เป็นอุปกรณ์ในการส่งข้อมูล ซึ่งการทำงานต่าง ๆ จะเกิดขึ้นจากการกดปุ่มกดของผู้ใช้ ดังแสดงในรูปที่ 3.2



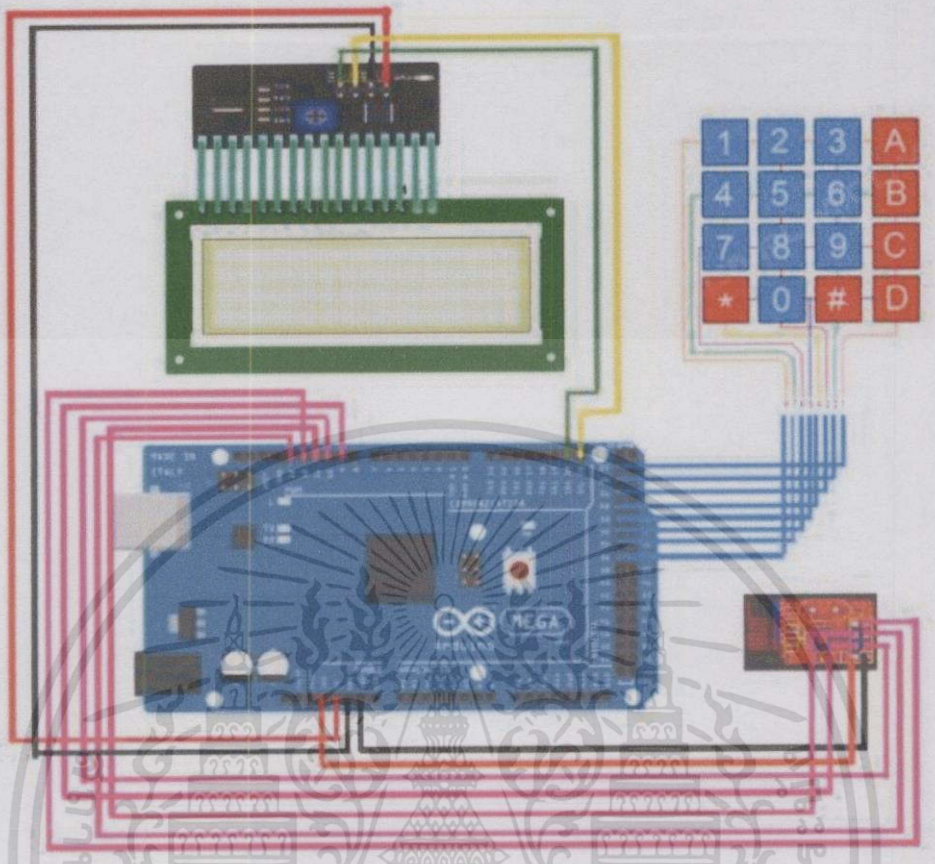
รูปที่ 3.2 โครงสร้างของอุปกรณ์รับรายการเครื่องดื่ม

วงจรภาคส่งสัญญาณ

วงจรภาคส่งสัญญาณจะรับค่าเป็นรหัสของรายการเครื่องดื่มนั้น ๆ จากนั้นจะผ่านการประมวลผล แล้วจึงแสดงผลค่าข้อมูลที่ได้ออกมา ซึ่งเป็นรายละเอียดในการสั่งเครื่องดื่ม และส่งค่าที่ได้ไปยังอุปกรณ์ภาครับสัญญาณ โดยจะมีอุปกรณ์หลัก ๆ ในการทำงาน ดังนี้

- Arduino Mega 2560 R3 ทำหน้าที่เป็นอุปกรณ์หลักของระบบในการควบคุมการทำงาน
- Keypad 4x4 ทำหน้าที่ในการรับค่าข้อมูลจากผู้ใช้
- LCD ขนาด 20x4 ทำหน้าที่ในการแสดงผลข้อมูลที่ได้
- nRF24L01 ทำหน้าที่ในการแปลงสัญญาณและส่งข้อมูลผ่าสัญญาณไร้สาย

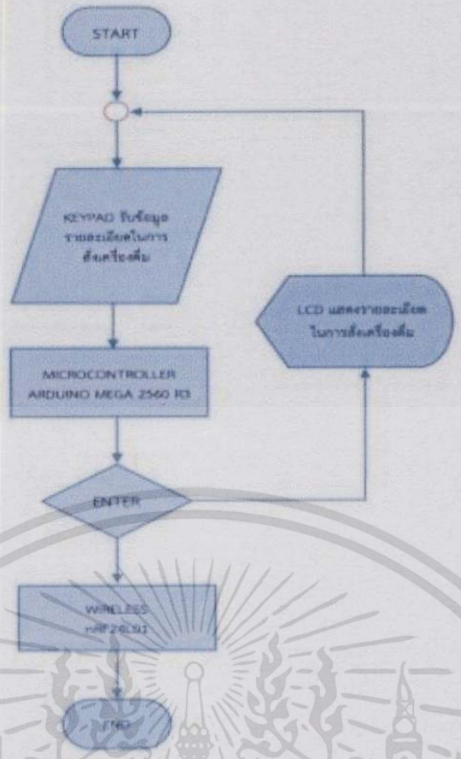
หลักการทำงาน คือ เมื่อกดปุ่มกดจะทำให้ข้อมูลที่ป้อนเข้ามาถูกส่งไปยังไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 เพื่อประมวลผลค่าที่ได้รับมา แล้วจึงแสดงผลออกทางจอแสดงผล LCD ขนาด 20x4 จากนั้นจะต้องมีการยืนยันค่าข้อมูลจากการกดปุ่มกดอีกครั้งหนึ่ง ข้อมูลจึงจะถูกส่งผ่านอุปกรณ์ส่งสัญญาณไร้สาย nRF24L01 ไปยังอุปกรณ์ภาครับสัญญาณ โดยอุปกรณ์แต่ละส่วนจะมีการเชื่อมต่อกันผ่านพอร์ตของ Arduino Mega 2560 R3 ดังในรูปที่ 3.3



รูปที่ 3.3 วงจรภาคส่งสัญญาณ

หลักการทำงานของภาคส่งสัญญาณ

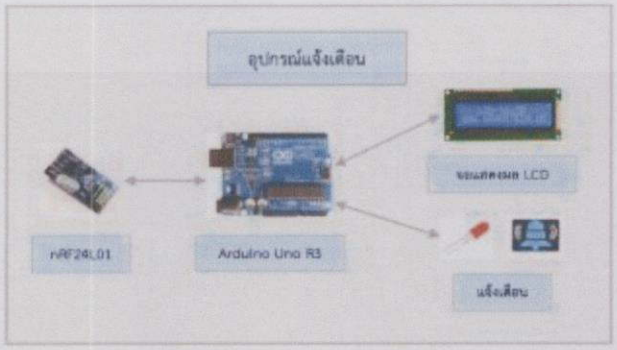
การทำงานของ Flow chart คือ เมื่อเริ่มกดปุ่มกดเพื่อทำการรับข้อมูลรายละเอียดในการสั่งเครื่องดื่ม จากนั้นข้อมูลจะถูกส่งไปยังไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 เพื่อทำการประมวลผล โดยจะทำการตรวจสอบว่ามีการกด ENTER ที่ปุ่มกด เพื่อยืนยันการสั่งรายการเครื่องดื่มแล้วหรือไม่ ถ้ายังไม่มีการกด ENTER เพื่อยืนยันการสั่งรายการเครื่องดื่ม ข้อมูลก็จะถูกส่งไปยังจอแสดงผล LCD ขนาด 20x4 เพื่อแสดงผลข้อมูลที่ได้รับมา แล้วให้ปุ่มกดรับค่าข้อมูลไปเรื่อย ๆ จนกว่าจะมีการกด ENTER เพื่อยืนยันข้อมูลในการสั่งเครื่องดื่ม แล้วจึงจะทำการส่งข้อมูลนั้นผ่านอุปกรณ์ส่งสัญญาณไร้สาย nRF24L01 ซึ่งอยู่ในย่านความถี่ 2.4 GHz ไปยังอุปกรณ์ภาครับสัญญาณต่อไป ดังแสดงในรูปที่ 3.4



รูปที่ 3.4 Flow chart แสดงขั้นตอนการทำงานของภาคส่งสัญญาณ

3.1.2 การออกแบบภาครับสัญญาณ

การรับข้อมูลของภาครับสัญญาณหรืออุปกรณ์แจ้งเตือนในการรับเครื่องดื่มจะอยู่ในรูปแบบของการรับข้อมูลแบบไร้สายโดยจะมี nRF24L01 ทำหน้าที่สำหรับรับสัญญาณข้อมูลแบบไร้สายที่ส่งมาจากภาคส่งสัญญาณซึ่งในที่นี้ nRF24L01 จะทำหน้าที่รับส่งสัญญาณไร้สายในย่านความถี่ 2.4GHz และมีการประมวลผลโดยใช้บอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3 จากนั้นจะส่งข้อมูลไปยังจอแสดงผล LCD ขนาด 16x2 เพื่อแสดงผลข้อมูลที่ได้รับมาและจะทำการแจ้งเตือนโดยการแสดงผลไฟ LED และส่งเสียงเตือน ดังแสดงในรูปที่ 3.5



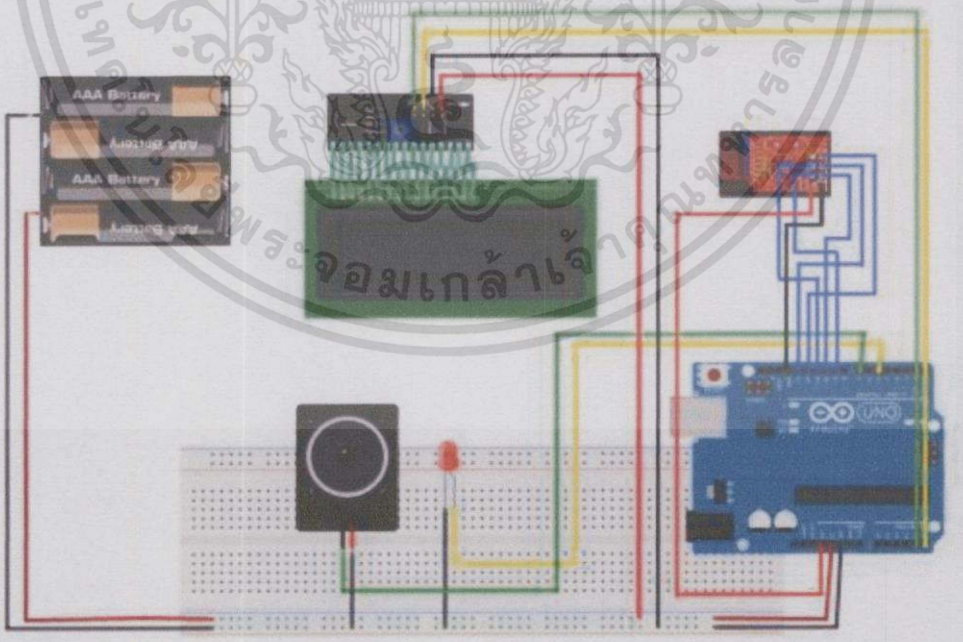
รูปที่ 3.5 โครงสร้างของอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

วงจรภาครับสัญญาณ

วงจรภาครับสัญญาณเป็นวงจรที่รับค่าข้อมูลผ่านสัญญาณไร้สายที่ส่งมาจากวงจรอุปกรณ์ภาคส่งสัญญาณ จากนั้นจะส่งต่อไปยังไมโครคอนโทรลเลอร์ Arduino Uno R3 เพื่อประมวลผลแล้วจึงแสดงผลค่าที่ได้ออกมาทางจอแสดงผล LCD ขนาด 16x2 เพื่อแสดงรายละเอียดในการสั่งเครื่องดื่ม และมีการแจ้งเตือนที่อุปกรณ์ของผู้ใช้งาน โดยจะมีอุปกรณ์หลักๆ ในการทำงาน ดังนี้

- Arduino Uno R3 ทำหน้าที่เป็นอุปกรณ์หลักของระบบในการควบคุมการทำงาน
- LCD 16x2 ทำหน้าที่ในการแสดงผลข้อมูลที่ได้รับมา
- nRF24L01 ทำหน้าที่ในการรับข้อมูลผ่านสัญญาณไร้สาย
- Speaker ทำหน้าที่ในการส่งเสียงแจ้งเตือน
- หลอดไฟ LED ทำหน้าที่ในแสดงไฟแจ้งเตือน

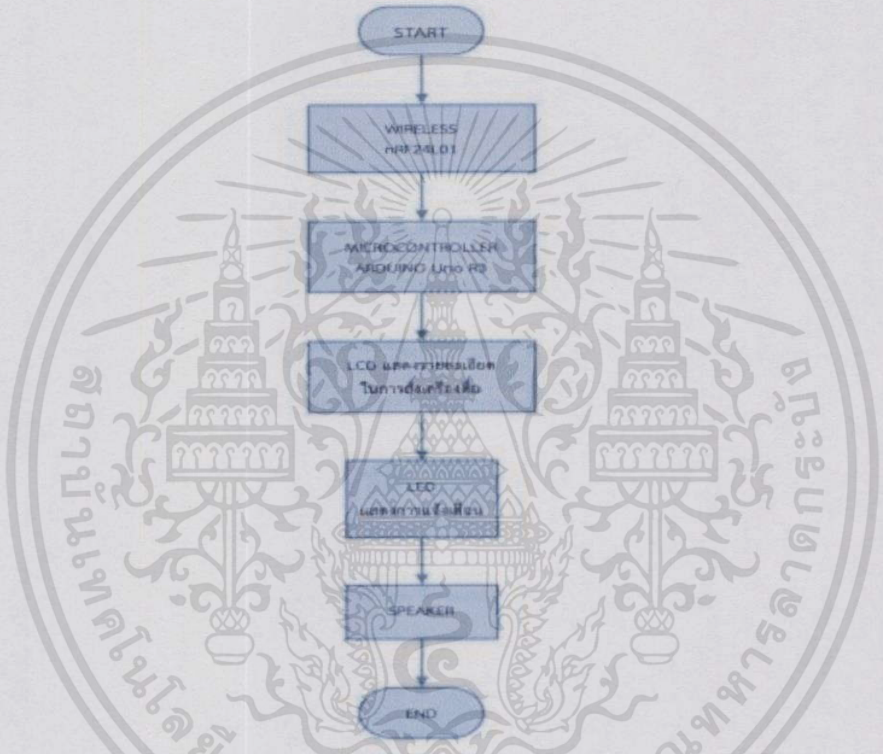
การทำงาน คือ เมื่ออุปกรณ์รับ-ส่งสัญญาณไร้สาย nRF24L01 รับค่าข้อมูลมาจากภาคส่งสัญญาณแล้วจะส่งข้อมูลต่อไปยังไมโครคอนโทรลเลอร์ Arduino Uno R3 เพื่อประมวลผล ซึ่งไมโครคอนโทรลเลอร์จะทำหน้าที่เป็นอุปกรณ์หลักของระบบในการควบคุมการทำงาน ดังนั้นจึงทำหน้าที่ควบคุมให้ข้อมูลส่งต่อไปยังจอแสดงผล LCD 16x2 เพื่อแสดงผลข้อมูลที่ได้รับมา นอกจากนั้นไมโครคอนโทรลเลอร์ยังทำการควบคุมการรับสัญญาณจากภาคส่งสัญญาณเพื่อให้เกิดการแจ้งเตือน ซึ่งการแจ้งเตือนจะแสดงผลเป็นไฟ LED และส่งเสียงเตือนโดยอุปกรณ์แต่ละส่วนจะมีการเชื่อมต่อกันผ่านพอร์ตของ Arduino Uno R3 ดังแสดงในรูปที่ 3.6



รูปที่ 3.6 วงจรภาครับสัญญาณ

หลักการทำงานของภาครับสัญญาณ

การทำงานของ Flow chart คือ เมื่ออุปกรณ์ภาครับสัญญาณรับค่าข้อมูลในรูปแบบสัญญาณไร้สายจากอุปกรณ์ภาคส่งสัญญาณแล้ว ข้อมูลที่ได้รับมาก็จะถูกส่งต่อไปยังไมโครคอนโทรลเลอร์ Arduino Uno R3 เพื่อทำการประมวลผลและส่งข้อมูลไปยังจอแสดงผล LCD ขนาด 16x2 เพื่อแสดงผลข้อมูลที่ได้รับมา จากนั้นอุปกรณ์ภาคส่งสัญญาณจะส่งข้อมูลในรูปแบบสัญญาณไร้สายเพื่อควบคุมการแจ้งเตือนมายังอุปกรณ์ภาครับสัญญาณอีกครั้งหนึ่ง โดยการแจ้งเตือนจะแสดงผลออกทางหลอดไฟ LED และส่งเสียงเตือน ดังแสดงในรูปที่ 3.7



รูปที่ 3.7 Flow chart แสดงขั้นตอนการทำงานของระบบอุปกรณ์แจ้งเตือน

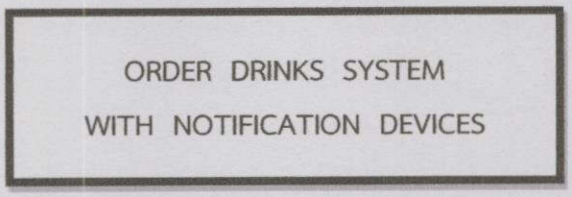
3.2 การออกแบบซอฟต์แวร์

ในส่วนของซอฟต์แวร์ได้มีการออกแบบหน้าจอแสดงผล LCD ทั้งของอุปกรณ์รับรายการเครื่องดื่ม และอุปกรณ์แจ้งเตือนในการรับเครื่องดื่มดังนี้

1. Espresso	ราคา 50 บาท
2. America-no	ราคา 50 บาท
3. Cappuccino	ราคา 50 บาท
4. Latte	ราคา 45 บาท
5. Mocha	ราคา 45 บาท

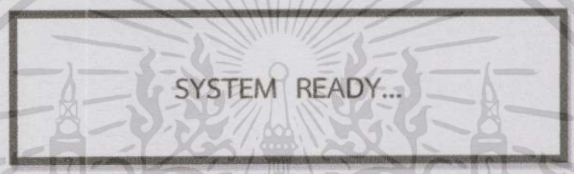
3.2.1 หน้าจอแสดงผลที่อุปกรณ์รับรายการเครื่องดื่ม

เมื่อเปิดเครื่องจะปรากฏข้อความ ORDER DRINKS SYSTEM WITH NOTIFICATION DEVICE



รูปที่ 3.8 การออกแบบแสดงข้อความเริ่มต้นการทำงานของระบบ

ต่อมาจะปรากฏข้อความ SYSTEM READY... เพื่อแสดงให้เห็นว่าระบบพร้อมใช้งาน

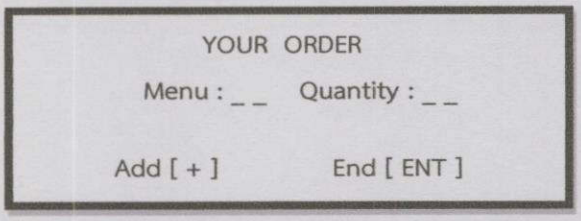


รูปที่ 3.9 การออกแบบแสดงข้อความระบบพร้อมใช้งาน

หน้าหลักของการทำงานจะมี 3 เมนูให้เลือกคือ 1.ORDER, 2.QUEUE, 3.PAY

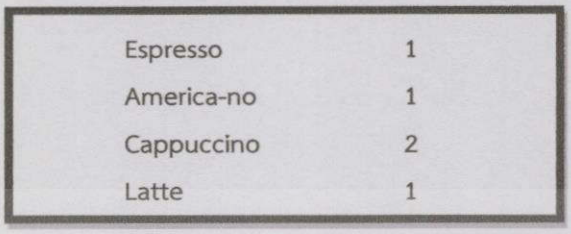
รูปที่ 3.10 การออกแบบหน้าหลักการทำงาน

เมื่อเลือกเมนูที่ 1.ORDER จะใช้สำหรับสั่งเครื่องดื่ม โดยกดตัวเลขตามลำดับเมนูที่จะสั่ง แล้วตามด้วยปุ่ม Quantity เพื่อบอกจำนวนแก้วที่จะสั่ง จากนั้นกด ENTER เพื่อยืนยันการสั่ง

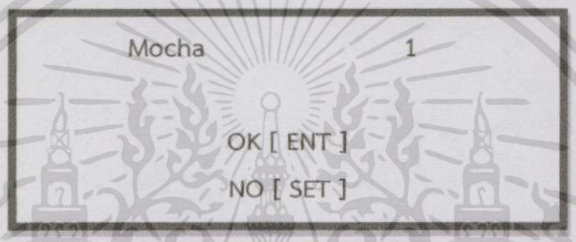


รูปที่ 3.11 การออกแบบหน้าการสั่งเครื่องดื่ม

เมื่อยืนยันการสั่งแล้วระบบจะตรวจสอบเมนูเครื่องดื่มที่สั่งอีกครั้งหนึ่ง โดยแสดงชื่อเมนู จำนวนที่สั่งให้ดูอีกครั้ง ถ้าเมนูถูกต้องตามต้องการแล้วให้กด ENTER เพื่อยืนยัน แต่ถ้าไม่ถูกต้องให้กด Home เพื่อกลับไปที่หน้าหลักของการทำงาน

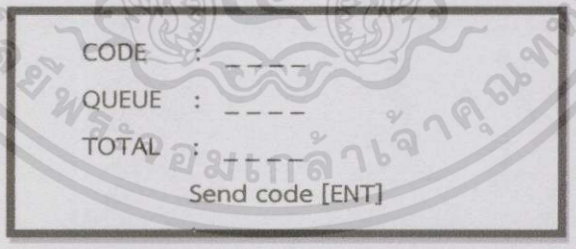


รูปที่ 3.12 การออกแบบแสดงรายละเอียดในการสั่งเครื่องดื่มเมนูที่ 1 ถึงเมนูที่ 4



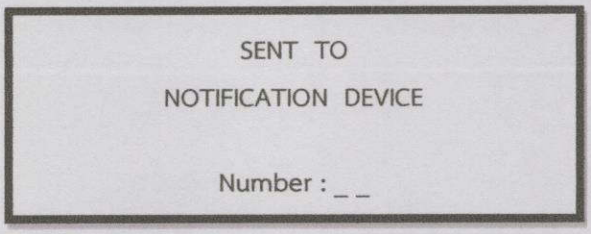
รูปที่ 3.13 การออกแบบแสดงรายละเอียดในการสั่งเครื่องดื่มเมนูที่ 5 และคำสั่ง ENT กับ SET

เมื่อยืนยันอีกครั้งแล้วระบบจะสร้างรหัสส่วนบุคคลเพื่อใช้ในการรับเครื่องดื่ม พร้อมทั้งแสดงคิว จำนวนเงินที่ต้องชำระ และปรากฏข้อความ [ENT] TO SEND CODE เพื่อส่งรหัสและจำนวนเงินไปยังอุปกรณ์แจ้งเตือน



รูปที่ 3.14 การออกแบบแสดง Code, Queue, Total

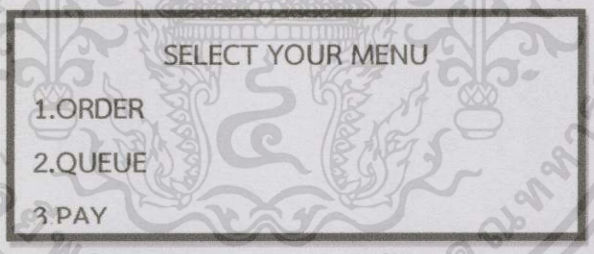
เมื่อกด ENTER แล้วจะปรากฏข้อความ Send code to notification device NUMBER เพื่อให้ใส่หมายเลขอุปกรณ์แจ้งเตือนที่ต้องการส่งข้อมูล แล้วกด ENTER



รูปที่ 3.15 การออกแบบแสดงหมายเลขอุปกรณ์แจ้งเตือนที่เราต้องการแจ้งเตือน

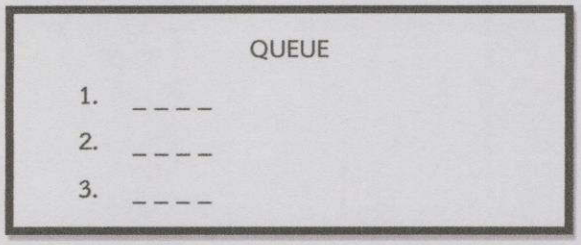
ปรากฏข้อความ SENT TO NOTIFICATION DEVICE SUCCEED แสดงว่าอุปกรณ์รับรายการเครื่องดื่มได้ส่งข้อมูลไปยังอุปกรณ์แจ้งเตือนสำเร็จแล้ว และระบบจะกลับเข้าสู่หน้าหลักโดยอัตโนมัติ

รูปที่ 3.16 การออกแบบแสดงข้อความว่าส่งสัญญาณแจ้งเตือนไปยังอุปกรณ์แจ้งเตือนเรียบร้อยแล้ว



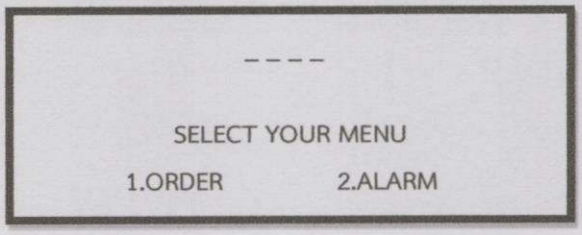
รูปที่ 3.17 การออกแบบหน้าหลักการทำงาน

เมื่อเลือกเมนูที่ 2.QUEUE จะปรากฏรหัสของการสั่งเครื่องดื่มโดยเรียงลำดับตามคิวที่สั่ง



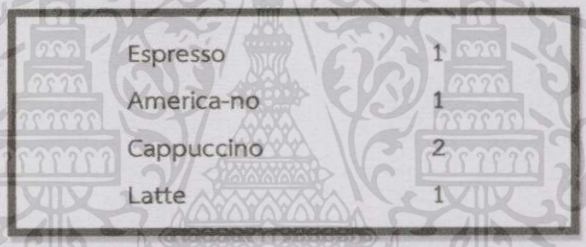
รูปที่ 3.18 การออกแบบแสดงรายละเอียดคิวในการสั่งเครื่องดื่ม

เมื่อเลือกคิวที่ 1 จะปรากฏรหัสส่วนบุคคลและปรากฏข้อความ SELECT YOUR MENU พร้อมทั้งแสดงเมนู 1.ORDER, 2.ALARM

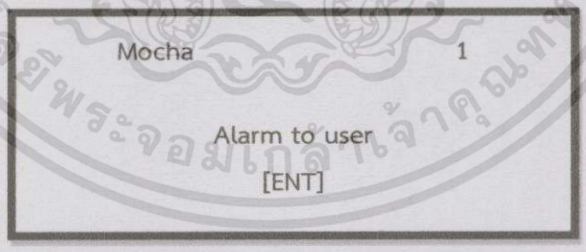


รูปที่ 3.19 การออกแบบแสดงรายละเอียดหน้าการแจ้งเตือนของแต่ละ Code

เมื่อเลือกเมนูที่ 1.ORDER จะแสดงรายการเครื่องดื่มที่สั่งอีกครั้ง เพื่อเป็นการตรวจสอบก่อนส่งสัญญาณแจ้งเตือนให้ลูกค้ามารับเครื่องดื่ม และปรากฏข้อความ ALARM TO USER PRESS [ENT]



รูปที่ 3.20 การออกแบบแสดงรายละเอียดในการสั่งเครื่องดื่มที่อยู่ในคิวเมนูที่ 1 ถึงเมนูที่ 4



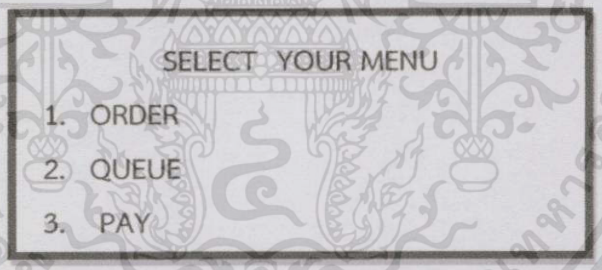
รูปที่ 3.21 การออกแบบแสดงรายละเอียดในการสั่งเครื่องดื่มที่อยู่ในคิวเมนูที่ 5 และคำสั่งแจ้งเตือน

เมื่อกด ENTER แล้วจะปรากฏข้อความ ALARM TO USER SUCCEED แสดงว่าได้ส่งสัญญาณแจ้งเตือนไปยังอุปกรณ์แจ้งเตือนที่อยู่กับลูกค้าสำเร็จแล้ว จากนั้นจะกลับเข้าสู่หน้าหลักโดยอัตโนมัติ

รูปที่ 3.22 การออกแบบแสดงการแจ้งเตือนไปที่อุปกรณ์แจ้งเตือนเรียบร้อยแร้ว

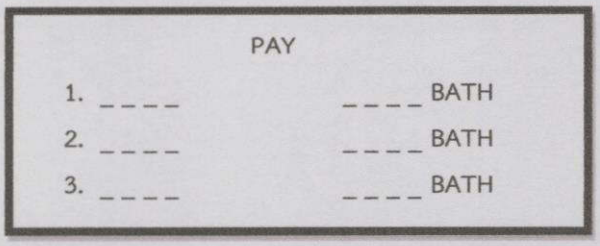
หรือถ้าเลือกเมนูที่ 2.ALARM จะปรากฏข้อความ ALARM TO USER SUCCEED แสดงว่าได้ส่งสัญญาณแจ้งเตือนสำเร็จแล้ว จากนั้นจะกลับเข้าสู่หน้าหลักโดยอัตโนมัติเช่นกัน

รูปที่ 3.23 การออกแบบแสดงการแจ้งเตือนไปที่อุปกรณ์แจ้งเตือนเรียบร้อยแล้ว



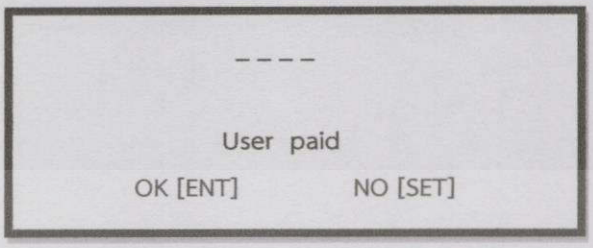
รูปที่ 3.24 การออกแบบหน้าหลักการทำงาน

เมื่อเลือกเมนูที่ 3.PAY จะแสดงรหัสและจำนวนเงินที่ต้องชำระ โดยเรียงลำดับของการจ่ายเงินตามการส่งสัญญาณแจ้งเตือนไปยังลูกค้า



รูปที่ 3.25 การออกแบบแสดงรายระเอียดในการจ่ายเงิน

เมื่อลูกค้าต้องการชำระเงิน เลือกลำดับที่ตรงกับรหัสส่วนบุคคลของลูกค้าแล้วจะแสดงรหัสนั้น พร้อมกับปรากฏข้อความ USER PAID จากนั้นกด ENTER เพื่อยืนยันว่าลูกค้าจ่ายเงินแล้วหรือกด HOME เพื่อกลับสู่หน้าหลัก



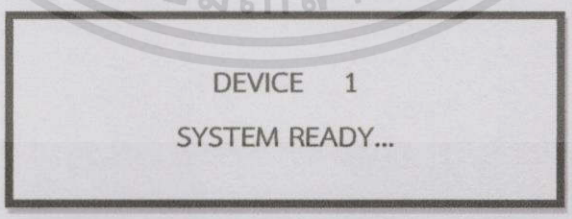
รูปที่ 3.26 การออกแบบแสดงรายละเอียดการจ่ายเงินของแต่ละ Code

เมื่อยืนยันการจ่ายเงินแล้ว จะปรากฏรหัสที่จ่ายเงินแล้วอีกครั้ง พร้อมทั้งปรากฏข้อความ DELETE SUCCEED แสดงว่าลบรหัสนี้ออกจากระบบสำเร็จแล้ว

รูปที่ 3.27 การออกแบบแสดงการจ่ายเงินเรียบร้อยแล้ว

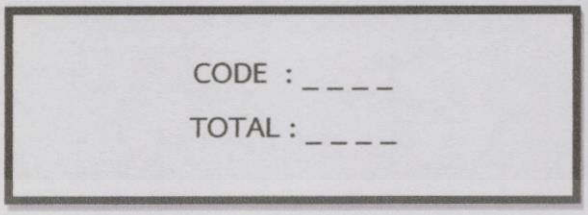
3.2.2 หน้าจอแสดงผลที่อุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

เมื่อระบบอยู่ในสถานะพร้อมใช้งานจะปรากฏข้อความ DEVICE 1 SYSTEM READY...



รูปที่ 3.28 การออกแบบแสดงสถานะพร้อมใช้งาน

เมื่อลูกค้าสั่งเครื่องดื่มที่เคาน์เตอร์เรียบร้อยแล้วอุปกรณ์รับรายการเครื่องดื่มจะส่งข้อมูลแบบไร้สาย โดยจะส่งรหัสส่วนบุคคลที่ใช้ในการรับเครื่องดื่ม จำนวนเงินที่ต้องชำระและจะแสดงผลออกทางจอแสดงผล LCD ขนาด 16x2



รูปที่ 3.29 การออกแบบจอแสดงผลของอุปกรณ์แจ้งเตือนการรับเครื่องดื่ม

บทที่ 4
การทดลองและผลการทดลอง

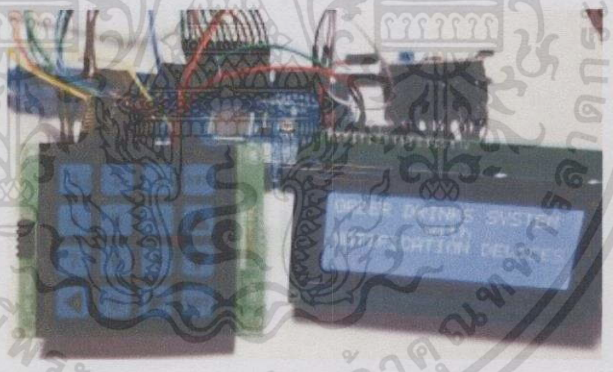
4.1 อุปกรณ์รับรายการเครื่องดื่ม

4.1.1 ฮาร์ดแวร์ของอุปกรณ์รับรายการเครื่องดื่ม

จากการทดลองต่ออุปกรณ์ฮาร์ดแวร์ของอุปกรณ์รับรายการเครื่องดื่ม ซึ่งประกอบด้วย

- จอแสดงผล LCD ขนาด 20x4
- ปุ่มกด ขนาด 4x4
- อุปกรณ์ลด Pin ในการต่อจอแสดงผล LCD
- อุปกรณ์รับ-ส่งสัญญาณไร้สาย nRF24L01

อุปกรณ์ทั้งหมดเชื่อมต่อกันผ่านพอร์ตของบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 ดังแสดงในรูปที่ 4.1 โดยรับค่าข้อมูลจากการกดปุ่มกด จากนั้นจะส่งข้อมูลต่อไปยัง Arduino Mega 2560 R3 เพื่อประมวลผล แล้วแสดงผลออกทางจอแสดงผล LCD 20x4 รวมถึงการรับค่าจากปุ่มกดเพื่อใช้ควบคุมการส่งสัญญาณไปยังอุปกรณ์แจ้งเตือน



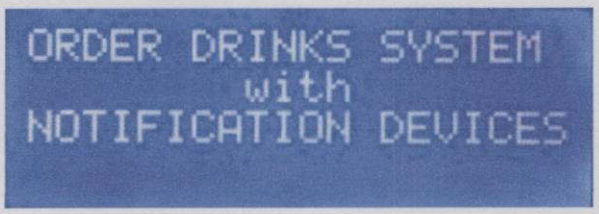
รูปที่ 4.1 การเชื่อมต่ออุปกรณ์รับรายการเครื่องดื่ม

4.1.2 ซอฟต์แวร์ของอุปกรณ์รับรายการเครื่องดื่ม

อุปกรณ์รับรายการเครื่องดื่มใช้การเขียนโปรแกรมด้วยภาษา C++ โดยเขียนผ่านโปรแกรม Arduino โดยได้กำหนดเมนูเครื่องดื่มไว้ดังนี้

1. Espresso	ราคา 50 บาท
2. America-no	ราคา 50 บาท
3. Cappuccino	ราคา 50 บาท
4. Latte	ราคา 45 บาท
5. Mocha	ราคา 45 บาท

เมื่อผู้ใช้เปิดเครื่องขึ้นมาจะปรากฏข้อความ ORDER DRINK SYSTEM with NOTIFICATION DEVICES ดังรูปที่ 4.2



รูปที่ 4.2 แสดงข้อความเริ่มต้นการทำงานของระบบ

ต่อมาจะปรากฏข้อความ SYSTEM READY... แสดงว่าระบบพร้อมใช้งานแล้ว ดังรูปที่ 4.2

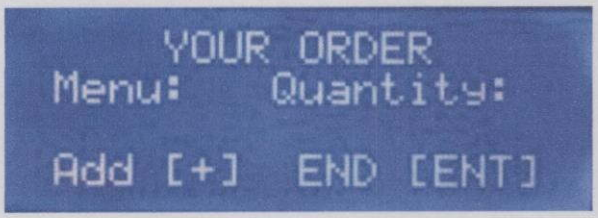
รูปที่ 4.3 แสดงข้อความระบบพร้อมใช้งาน

หน้าหลักของการทำงานจะมี 3 เมนูให้เลือก คือ

1. ORDER
2. QUEUE
3. PAY

รูปที่ 4.4 แสดงหน้าหลักการทำงาน

เมื่อเลือกเมนูที่ 1. ORDER จะใช้สำหรับสั่งเครื่องดื่มโดยการกดหมายเลขเครื่องดื่มที่จะสั่งตามเมนูของทางร้าน จากนั้นกดปุ่ม Q เพื่อบอกจำนวนแก้วที่จะสั่ง จากนั้นเลือกว่าจะสั่งเครื่องดื่มอื่นต่อหรือไม่ ถ้าจะสั่งต่อให้กด + แต่ถ้าสั่งเสร็จแล้วกดปุ่ม ENT ดังรูปที่ 4.5



รูปที่ 4.5 แสดงหน้าการสั่งเครื่องดื่ม

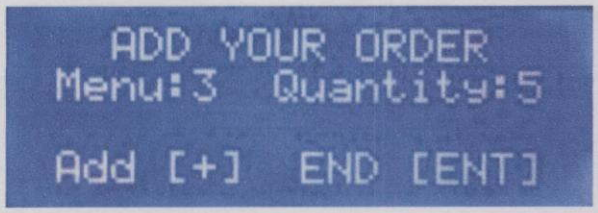
ยกตัวอย่างเช่น เลือกเมนูที่ 1. Espresso จำนวน 3 แก้ว ต้องกดปุ่ม 1 ตามด้วย Q และ 3 ตามลำดับ จากนั้นกด ENT เพื่อจบการสั่งเครื่องดื่ม ดังรูปที่ 4.6

รูปที่ 4.6 แสดงตัวอย่างการสั่งเครื่องดื่ม

แต่ถ้าต้องการสั่งเมนูเพิ่ม แทนที่จะกด ENT ต้องกด + แล้วจะปรากฏคำว่า ADD YOUR ORDER ขึ้นมา ดังรูปที่ 4.7

รูปที่ 4.7 แสดงหน้าการสั่งเครื่องดื่มเพิ่ม

จากนั้นสามารถสั่งเครื่องดื่มได้ โดยใช้หลักการสั่งเดียวกันกับในขั้นตอนที่ผ่านมา ยกตัวอย่างเช่น ต้องการสั่งเมนูที่ 3. Cappuccino จำนวน 5 แก้ว ต้องกดปุ่ม 3 ตามด้วย Q และ 5 ตามลำดับ จากนั้นกด ENT เพื่อจบการสั่งเครื่องดื่ม ดังรูปที่ 4.8



รูปที่ 4.8 แสดงตัวอย่างการสั่งเครื่องดื่มเพิ่ม

จากนั้นระบบจะแสดงรายการเครื่องดื่มที่สั่งอีกครั้งเพื่อตรวจสอบความถูกต้อง ถ้ารายการที่สั่งถูกต้องแล้วกด ENT แต่ถ้าไม่ถูกต้องกด SET แล้วจะกลับไปที่หน้าหลัก

จากตัวอย่างที่แล้ว ได้สั่งเมนูที่ 1. Espresso จำนวน 3 แก้ว และเมนูที่ 3. Cappuccino จำนวน 5 แก้ว ซึ่งระบบจะแสดงรายการเครื่องดื่มที่สั่งอีกครั้ง เมื่อถูกต้องแล้วกด ENT ดังรูปที่ 4.9 และ 4.10

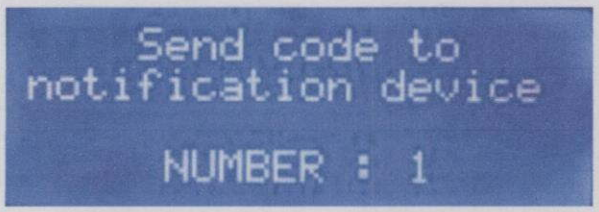
รูปที่ 4.9 แสดงรายละเอียดในการสั่งเครื่องดื่มเมนูที่ 1 ถึงเมนูที่ 4

รูปที่ 4.10 แสดงรายละเอียดในการสั่งเครื่องดื่มเมนูที่ 5 และคำสั่ง ENT กับ SET

เมื่อกด ENT เพื่อยืนยันรายการที่สั่งแล้วระบบจะสร้างรหัสส่วนบุคคลเพื่อใช้ในการรับเครื่องดื่ม แสดงคิวที่มีอยู่ในระบบ และจำนวนเงินที่ต้องชำระ จากนั้นระบบจะแจ้งว่าส่งข้อมูลกด ENT ดังรูปที่ 4.11

รูปที่ 4.11 แสดง Code, Queue, Total

เมื่อกด ENT แล้วจะปรากฏ Send code to notification device NUMBER : เพื่อใส่หมายเลขอุปกรณ์แจ้งเตือนที่ต้องการส่งข้อมูล แล้วกด ENT ดังรูปที่ 4.12



รูปที่ 4.12 แสดงหมายเลขอุปกรณ์แจ้งเตือนที่ต้องการแจ้งเตือน

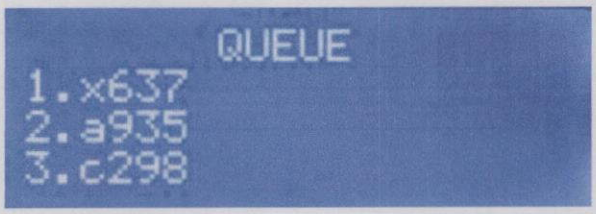
ปรากฏข้อความ SENT TO NOTIFICATION DEVICE SUCCEED แสดงว่าอุปกรณ์รับรายการเครื่องดื่มได้ส่งข้อมูลรหัสส่วนบุคคล และราคาที่ต้องชำระไปยังอุปกรณ์แจ้งเตือนเครื่องที่ 1 สำเร็จแล้ว จากนั้นจะเข้าหน้าหลักโดยอัตโนมัติพร้อมทั้งเพิ่มรหัสนี้เข้าไปใน เมนูที่ 2. QUEUE

รูปที่ 4.13 แสดงข้อความว่าส่งสัญญาณแจ้งเตือนไปยังอุปกรณ์แจ้งเตือนเรียบร้อยแล้ว

จากนั้นระบบจะกลับไปที่หน้าการทำงานหลักเพื่อรอรับการสั่งเครื่องต่อไป ดังรูปที่ 4.14

รูปที่ 4.14 แสดงหน้าการทำงานหลัก

ต่อมาเมื่อเลือกเมนูที่ 2. QUEUE จะแสดงรหัสของลูกค้าที่กำลังรอเครื่องดื่มอยู่ โดยจะเรียงลำดับตามคิวที่สั่ง ยกตัวอย่างเช่น ในระบบมีคิว 3 คิว และคิวที่ 1. x637 คือรหัสส่วนบุคคลของตัวอย่างที่ได้กล่าวมา ดังรูปที่ 4.15



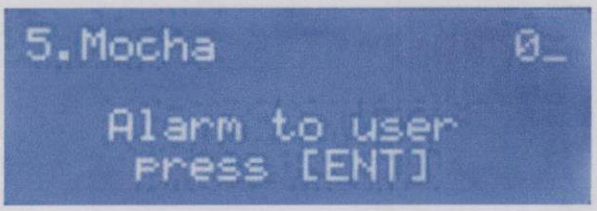
รูปที่ 4.15 แสดงรายละเอียดคิวในการสั่งเครื่องดื่ม

ถ้าเลือกคิวที่ 1. x637 จะแสดงรหัสเดิมและมีเมนูให้เลือก คือ 1. ORDER และ 2. ALARM ดังรูปที่ 4.16

รูปที่ 4.16 แสดงรายละเอียดยหน้าการแจ้งเตือนของแต่ละ Code

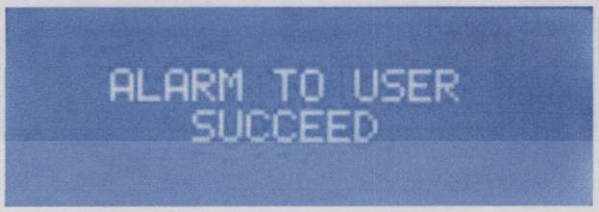
ถ้าเลือกเมนูที่ 1. ORDER จะแสดงรายการเครื่องดื่มที่สั่งอีกครั้ง จากตัวอย่างที่แล้ว ได้สั่งเมนูที่ 1. Espresso จำนวน 3 แก้ว และเมนูที่ 3. Cappuccino จำนวน 5 แก้ว ซึ่งระบบจะแสดงรายการเครื่องดื่มที่สั่งอีกครั้งเพื่อตรวจสอบความถูกต้องก่อนส่งสัญญาณแจ้งเตือนไปยังลูกค้า โดยจะปรากฏข้อความ Alarm to user press [ENT] ดังรูปที่ 4.17 และ รูปที่ 4.18

รูปที่ 4.17 แสดงรายละเอียดในการสั่งเครื่องที่อยู่ในคิวเมนูที่ 1 ถึงเมนูที่ 4



รูปที่ 4.18 แสดงรายละเอียดในการสั่งเครื่องที่อยู่ในคิวเมนูที่ 5 และคำสั่งแจ้งเตือน

เมื่อกด ENT แล้ว ระบบจะส่งสัญญาณแจ้งเตือนไปยังลูกค้า และปรากฏข้อความ ALARM TO USER SUCCEED เพื่อแจ้งลูกค้าว่าเครื่องดื่มที่สั่งเสร็จเรียบร้อยแล้ว จากนั้นจะเข้าหน้าหลัก พร้อมทั้งลบรหัสนี้ออกจากเมนูที่ 2.QUEUE และเพิ่มรหัสนี้เข้าไปในเมนูที่ 3. PAY โดยอัตโนมัติ ดังรูปที่ 4.19



รูปที่ 4.19 แสดงการแจ้งเตือนไปที่อุปกรณ์แจ้งเตือนเรียบร้อยแร้ว

แต่ถ้าเลือกเมนูที่ 2. ALARM ตั้งแต่เลือกคิวที่ 1. x637 คือ ต้องการแจ้งเตือนลูกค้าทันที โดยไม่ต้องการดูรายการที่สั่งอีกจะปรากฏข้อความ ALARM TO USER SUCCEED จากนั้นจะเข้าหน้าหลักพร้อมทั้งลบรหัสนี้ออกจากเมนูที่ 2.QUEUE และเพิ่มรหัสนี้เข้าไปในเมนูที่ 3. PAY โดยอัตโนมัติเช่นกัน ดังรูปที่ 4.1

จากนั้นระบบจะกลับไปที่หน้าการทำงานหลัก ดังรูปที่ 4.20

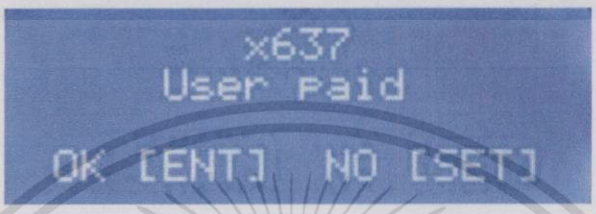
รูปที่ 4.20 แสดงหน้าการทำงานหลัก

สุดท้ายเมื่อเลือกเมนูที่ 3. PAY ใช้สำหรับชำระค่าเครื่องดื่ม โดยจะแสดงรหัส และจำนวนเงินที่ต้องชำระ ซึ่งเรียงลำดับของการจ่ายเงินตามการส่งสัญญาณแจ้งเตือนไปยังลูกค้า ดังรูปที่ 4.21 จากตัวอย่างที่กล่าวมาจะเห็นว่ารหัสส่วนบุคคลทั้งสามลำดับนี้สอดคล้องกับเมนูที่ 2. QUEUE ที่ได้ส่งสัญญาณแจ้งเตือนไปยังลูกค้า

รูปที่ 4.21 แสดงรายระเอียดในการจ่ายเงิน

เมื่อลูกค้าต้องการชำระเงินจะมาแจ้งรหัสรับเครื่องดื่ม จากนั้นกดลำดับที่ตรงกับรหัสนั้น แล้วจะแสดงรหัสนั้น พร้อมกับปรากฏข้อความ USER PAID จากนั้นกด ENT เพื่อยืนยันว่าลูกค้าจ่ายเงินแล้ว หรือกด SET เพื่อกลับสู่หน้าหลัก ดังรูปที่ 4.22

จากตัวอย่างที่กล่าวมา ถ้าลูกค้ารหัส x637 มาชำระเงินแล้ว จะสามารถยืนยันว่ารหัสนี้ได้ชำระเงินเรียบร้อยแล้วได้โดยกด 1 แล้วตามด้วย ENT จากนั้นรหัสนี้จะถูกลบออกจากระบบและกลับเข้าสู่หน้าหลักโดยอัตโนมัติ ดังรูปที่ 4.23



รูปที่ 4.22 แสดงรายละเอียดการจ่ายเงินของแต่ละ Code

รูปที่ 4.23 แสดงการจ่ายเงินเรียบร้อยแล้ว

จากนั้นระบบจะกลับไปที่หน้าการทำงานหลักเพื่อรอรับการสั่งเครื่องดื่มต่อไป ดังรูปที่ 4.24

รูปที่ 4.24 แสดงหน้าการทำงานหลัก

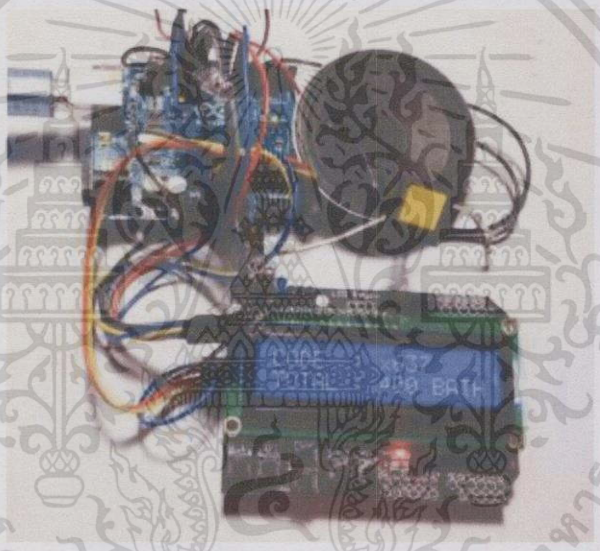
4.2 อุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

4.2.1 ฮาร์ดแวร์ของอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

จากการทดลองต่ออุปกรณ์ฮาร์ดแวร์ ของอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม ซึ่งประกอบด้วย

- จอแสดงผล LCD ขนาด 16x2
- อุปกรณ์ลด Pin ในการต่อจอแสดงผล LCD
- อุปกรณ์รับ-ส่งสัญญาณไร้สาย nFR24L01
- หลอดไฟ LED
- ลำโพง

อุปกรณ์ทั้งหมดเชื่อมต่อกันผ่านพอร์ตของบอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3 ที่ทำหน้าที่เป็นส่วนประมวลผลการทำงาน ดังแสดงในรูปที่ 4.25 โดยอุปกรณ์แจ้งเตือนจะได้รับสัญญาณผ่านอุปกรณ์รับ-ส่งสัญญาณไร้สาย nRF24L01 เพื่อแสดงรายละเอียดในการสั่งเครื่องดื่มออกทางจอแสดงผล LCD ขนาด 20x4 จากนั้นเมื่อเครื่องดื่มที่สั่งเสร็จแล้ว จะได้รับสัญญาณแจ้งเตือนโดยแสดงผลออกทางหลอดไฟ LED และเสียงแจ้งเตือน เพื่อไปรับเครื่องดื่มอีกครั้งหนึ่ง



รูปที่ 4.25 การเชื่อมต่ออุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

4.2.2 ซอฟต์แวร์ของอุปกรณ์แจ้งเตือนการรับเครื่องดื่ม

เมื่อระบบอยู่ในสถานะพร้อมใช้งานจะปรากฏข้อความ DEVICE 1 SYSTEM READY...

รูปที่ 4.26 แสดงสถานะพร้อมใช้งาน

อุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม ใช้การเขียนโปรแกรมด้วยภาษา C++ โดยเขียนผ่านโปรแกรม Arduino เมื่ออุปกรณ์แจ้งเตือนได้รับสัญญาณข้อมูลเพื่อให้แสดงผลรายละเอียดการสั่งเครื่องดื่ม โดยแสดงรหัสรับเครื่องดื่ม และจำนวนเงินที่ต้องชำระ ดังรูปที่ 4.26

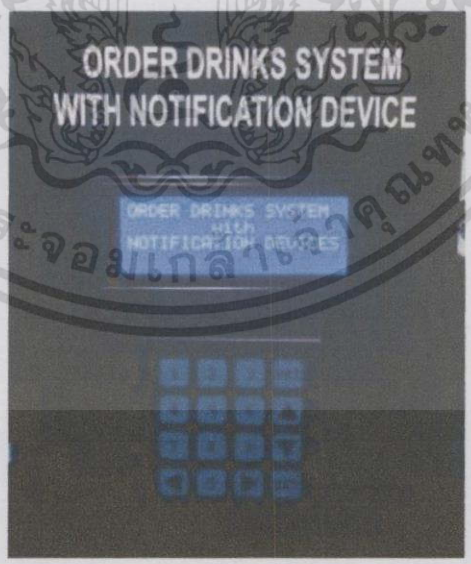
รูปที่ 4.27 จอแสดงผลของอุปกรณ์แจ้งเตือนการรับเครื่องดื่ม

บทที่ 5
สรุปผลและแนวทางการพัฒนาโครงงาน

จากการดำเนินงานโครงงานเรื่อง ระบบสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม สามารถสรุปผลการดำเนินงานและแนวทางการพัฒนาได้ดังนี้

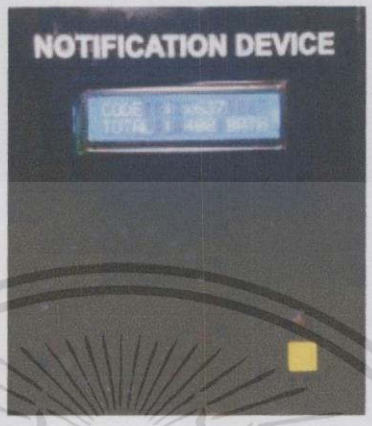
5.1 สรุปผลการทำโครงงาน

ในการดำเนินโครงงานเรื่องระบบสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม เราได้ทำการศึกษา วิเคราะห์ ออกแบบและพัฒนาระบบ ซึ่งประกอบด้วย เครื่องรับรายเครื่องดื่ม และอุปกรณ์แจ้งเตือน โดยในส่วนของฮาร์ดแวร์ เราได้ทำการจำลองการทำงานโดยออกแบบการเชื่อมต่ออุปกรณ์ต่าง ๆ ไว้บนโปรแกรม Proteus และในส่วนของซอฟต์แวร์ เราได้ใช้โปรแกรม Arduino ในการเขียนโปรแกรม ซึ่งเมื่อเราได้ทำการออกแบบและสร้างระบบทั้งหมดแล้ว ความสามารถของระบบเราที่สามารถทำได้ ก็คือ ในส่วนของอุปกรณ์รับรายการเครื่องดื่มสามารถรับรหัสข้อมูลที่ป้อนเข้ามาจากการสั่งเครื่องดื่มได้พร้อมทั้งสามารถสร้างรหัสรับเครื่องดื่มส่วนบุคคลขึ้นมา และส่งรหัสรับเครื่องดื่มส่วนบุคคลนี้พร้อมกับรายละเอียดในการสั่งเครื่องดื่มไปยังอุปกรณ์แจ้งเตือน โดยผ่านทางการส่งสัญญาณข้อมูลแบบไร้สาย ซึ่งลักษณะของอุปกรณ์รับรายการเครื่องดื่ม เป็นดังรูปที่ 5.1



รูปที่ 5.1 อุปกรณ์รับรายการเครื่องดื่ม

และในส่วนของอุปกรณ์แจ้งเตือน สามารถแสดงผลข้อมูลรายละเอียดในการสั่งเครื่องดื่มที่ส่งมาจากอุปกรณ์รับรายการเครื่องดื่มได้ พร้อมทั้งสามารถแจ้งเตือนได้ โดยการแสดงไฟและเสียงในการแจ้งเตือน ซึ่งลักษณะของอุปกรณ์แจ้งเตือน เป็นดังรูปที่ 5.2



รูปที่ 5.2 อุปกรณ์แจ้งเตือน

จากการดำเนินงานทำโครงงานครั้งนี้ ทำให้ผู้จัดทำมีประสบการณ์ในการศึกษาค้นคว้าหาความรู้ คิดวิเคราะห์ วางแผนการทำงาน และมีความสามารถในการออกแบบและพัฒนาระบบ ทั้งทางด้านซอฟต์แวร์และฮาร์ดแวร์ ทำให้เข้าใจหลักการทำงานต่าง ๆ ของระบบงานและเข้าใจการเขียนโปรแกรมเพื่อควบคุมไมโครคอนโทรลเลอร์ รวมถึงการเชื่อมต่ออุปกรณ์ต่าง ๆ เข้ากับไมโครคอนโทรลเลอร์ และยังสามารถนำความรู้ที่ได้ไปประยุกต์ใช้กับการทำงานในอนาคตต่อไปได้

5.2 ปัญหาที่เกิดขึ้นระหว่างการทำโครงงานและแนวทางการแก้ไข

1. ในการออกแบบระบบการทำงานของระบบสั่งเครื่องดื่ม ส่วนที่เป็นการส่งข้อมูลผ่านอุปกรณ์ไร้สาย Library อุปกรณ์หาได้ยาก จึงต้องทำการแก้ไข Library เอง เพื่อให้เหมาะกับระบบของเรา
2. ในการเชื่อมต่ออุปกรณ์ฮาร์ดแวร์ มีการเชื่อมต่ออุปกรณ์ฮาร์ดแวร์เข้าด้วยกันหลายตัว ทำให้เมื่อเกิดการเสียหาย จึงเป็นการยากที่จะหาสาเหตุว่าเกิดการเสียหายจากอุปกรณ์ใด ทำให้เสียเวลาในการทำงาน ดังนั้นจึงต้องใช้ความระมัดระวังเป็นอย่างมากในการเชื่อมต่ออุปกรณ์ต่าง ๆ
3. ในการเขียนโปรแกรมเพื่อออกแบบระบบสั่งเครื่องดื่ม จะอยู่ในรูปแบบที่ต้องใช้เงื่อนไขในการเขียนโปรแกรมซ้อนกันหลายเงื่อนไข เมื่อต้องการจะเปลี่ยน เพิ่มเติม หรือหากเกิดข้อผิดพลาดกับระบบจะสามารถแก้ไขโปรแกรมได้ยาก ดังนั้นในการออกแบบระบบจึงแยกฟังก์ชั่นของแต่ละส่วนการทำงานอย่างชัดเจน

บรรณานุกรม

[1] นายเอกชัย มะการ, เรียนรู้ เข้าใจ ใช้งาน ไมโครคอนโทรลเลอร์ตระกูล AVR ด้วย Arduino, บริษัท อีทีที จำกัด, 2552

[2] นายธีรวัฒน์ ประกอบผล, การประยุกต์ใช้งานไมโครคอนโทรลเลอร์, สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2540

[3] นายดอนสัน ปงผาบ, ไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน1, สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2549

[4] นายดอนสัน ปงผาบ, ไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน2, สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2549

[5] http://arduino.cc/

[6] http://arduino.cc/en/Main/arduinoBoardMega2560

[7] http://arduino.cc/en/uploads/Main/arduino-mega2560_R3-sch.pdf

[8] http://arduino.cc/en/Hacking/PinMapping2560#.Ux0dsvmSzTA

[9] http://arduino.cc/en/Main/arduinoBoardUno#.UxzIMfmSzTA

[10] http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

[11] https://lh3.googleusercontent.com/-mxyi3snCRyg/UUCO110pEzI/AAAAAAAAGyA/2P9QIUVPPZ8/s1600/ARDUINO_V2.png

[12] http://playground.arduino.cc/InterfacingWithHardware/Nrf24L01

[13] http://www.geeetech.com/Documents/RF2401AG%20Datasheet.pdf

[14] http://www.ecpe.nu.ac.th/ponpisut/mi_Lab5.pdf

[15] http://www.elecfreaks.com/wiki/index.php?title=2.4G_Wireless_nRF24L01p#Board_Schematic

[16] http://www.dfrobot.com/wiki/images/a/a7/LCDKeypad_Shield_SCH.png

[17] http://arduino-info.wikispaces.com/LCD-Blue-I2C

[18] https://www.sparkfun.com/datasheets/Wireless/Nordic/nRF24L01P_Product_Specification_1_0.pdf

[19] http://www.es.co.th/Schemetic/PDF/ABG128064A15-BIW.PDF

[20] http://tronixstuff.com/2013/12/16/arduino-tutorials-chapter-42-numeric-keypads/

[21] http://www.dfrobot.com/wiki/index.php/I2C/TWI_LCD1602_Module_(SKU:_DFR0063)

ภาคผนวก

ภาคผนวก ก.

คู่มือการใช้งาน

คู่มือการใช้งาน

การใช้งานระบบสั่งเครื่องดื่มโดยมีอุปกรณ์แจ้งเตือน จะมีอุปกรณ์ 2 ส่วน ดังนี้

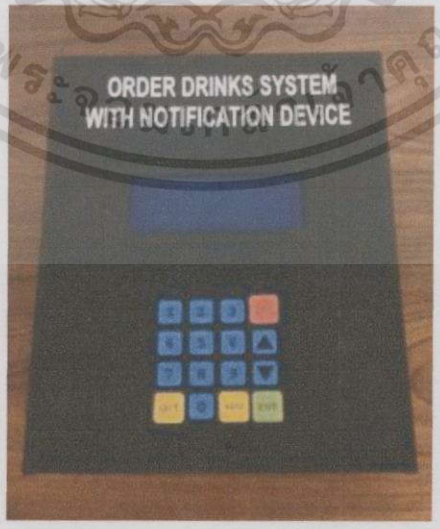
1. อุปกรณ์รับรายการเครื่องดื่มหรืออุปกรณ์ภาคส่งสัญญาณ
2. อุปกรณ์แจ้งเตือนหรืออุปกรณ์ภาครับสัญญาณ

โดยการใช้งานอุปกรณ์แต่ละส่วนต้องเปิดสวิตช์ก่อนทุกครั้งและรายละเอียดของแต่ละส่วนจะกล่าวในหัวข้อถัดไป

1. **การใช้งานอุปกรณ์รับรายการเครื่องดื่ม**

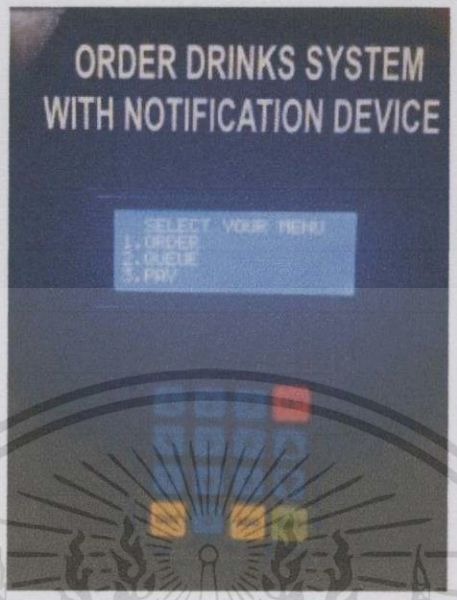
เมื่อกดสวิตช์ไฟหน้าจอจะติด และอุปกรณ์พร้อมใช้งาน

รูปที่ ก.1 สวิตช์เปิดปิดอุปกรณ์รับรายการเครื่องดื่ม



รูปที่ ก.2 อุปกรณ์รับรายการเครื่องดื่มขณะปิด

หน้าหลักของการทำงานจะมี 3 เมนูให้เลือก



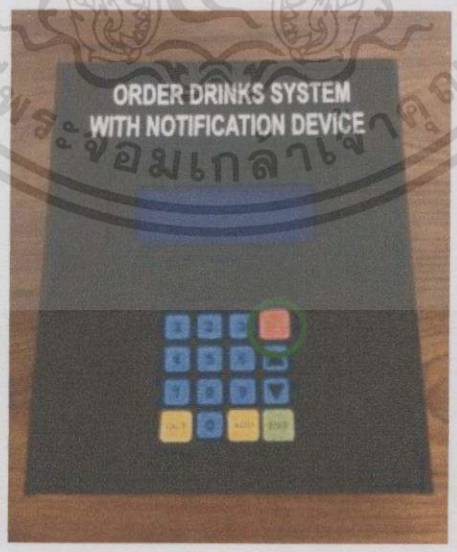
รูปที่ ก.3 หน้าหลักการทำงาน

เมนู 1. ORDER ใช้รับรายการเครื่องดื่ม และสร้างรหัสรับเครื่องดื่ม

เมนู 2. QUEUE แสดงลำดับการสั่งเครื่องดื่ม และส่งสัญญาณแจ้งเตือน

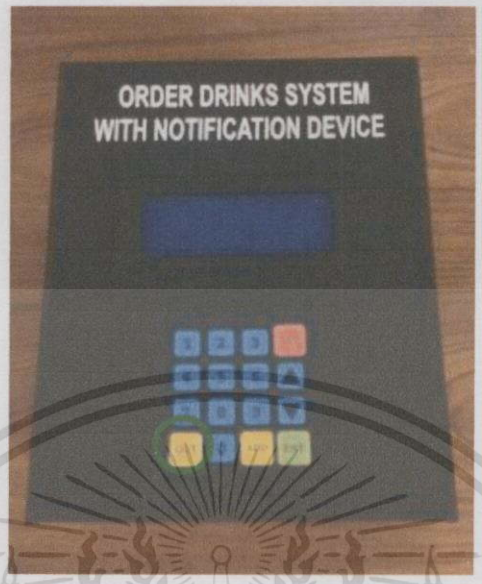
เมนู 3. PAY ใช้ลบข้อมูลออกจากระบบหลังจากลูกค้าจ่ายเงินเรียบร้อยแล้ว

ปุ่ม SET ใช้เพื่อกลับเข้าสู่หน้าหลักของการทำงาน



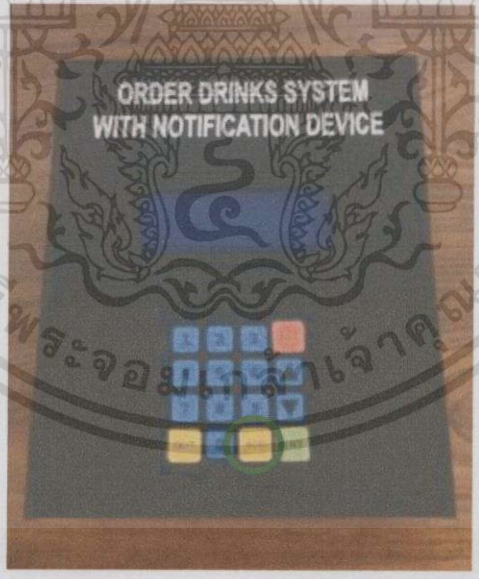
รูปที่ ก.4 ปุ่ม SET

ปุ่ม QUT ใช้เพื่อบอกจำนวนเครื่องดื่มที่จะสั่ง



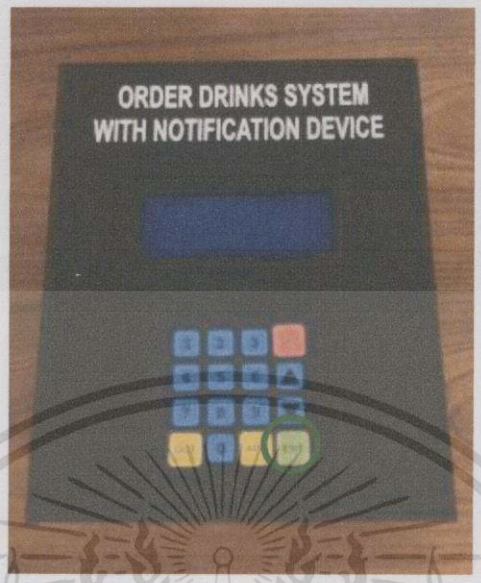
รูปที่ ก.5 ปุ่ม QUT

ปุ่ม ADD ใช้เพื่อต้องการสั่งเครื่องดื่มเพิ่ม



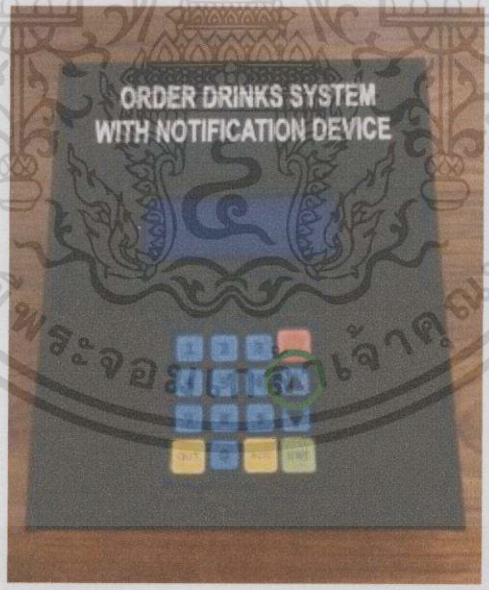
รูปที่ ก.6 ปุ่ม ADD

ปุ่ม ENT ใช้เพื่อยืนยันการทำงานต่าง ๆ ของระบบ



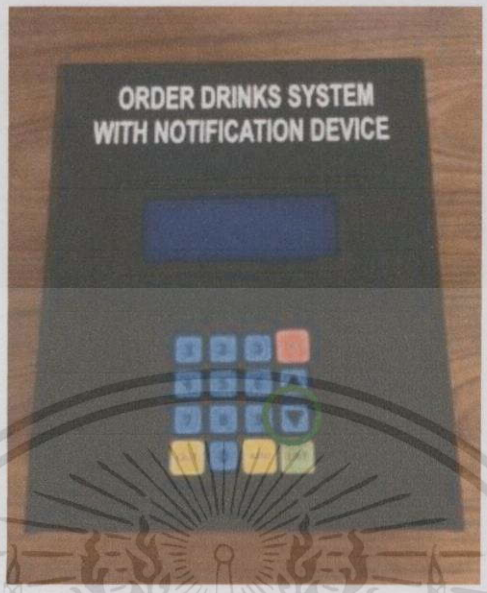
รูปที่ ก.7 ปุ่ม ENT

ปุ่ม ▲ ใช้เพื่อเลื่อนหน้าจอขึ้น



รูปที่ ก.8 ปุ่มลูกศรขึ้น

ปุ่ม ▼ ใช้เพื่อเลื่อนหน้าจอลง

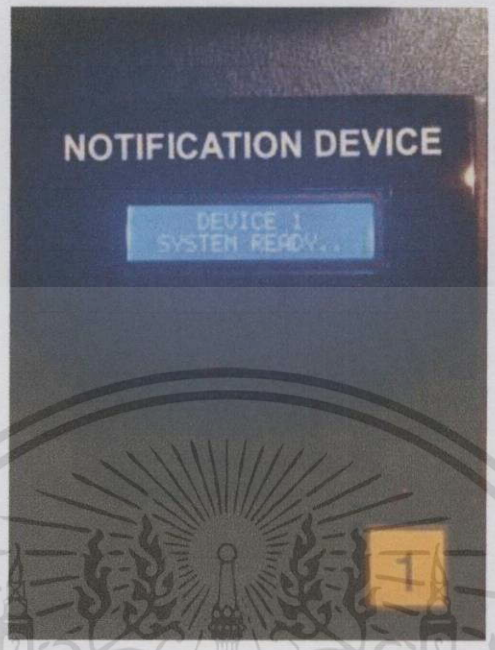


รูปที่ ก.9 ปุ่มลูกศรลง

2. **การใช้งานของอุปกรณ์แจ้งเตือน**
เมื่อกดสวิตช์ไฟหน้าจอจะติด และอุปกรณ์พร้อมใช้งาน

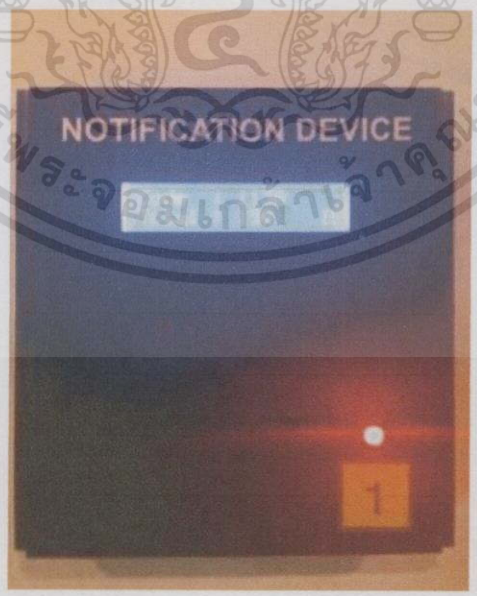
รูปที่ ก.10 สวิตช์เปิดปิดอุปกรณ์แจ้งเตือน

หลังจากนั้นอุปกรณ์จะเตรียมพร้อมเพื่อรอรับข้อมูลที่ส่งมาจากอุปกรณ์รับรายการเครื่องดื่ม



รูปที่ ก.11 อุปกรณ์แจ้งเตือนพร้อมใช้งาน

เมื่อสั่งเครื่องดื่มเสร็จอุปกรณ์แจ้งเตือนจะได้รับรหัสที่ใช้รับเครื่องดื่มและจำนวนเงินที่ต้องชำระสุดท้าย เมื่อเครื่องดื่มที่สั่งเสร็จแล้วอุปกรณ์รับรายการเครื่องดื่มจะส่งสัญญาณแจ้งเตือนมาอีกครั้งหนึ่ง จากนั้นอุปกรณ์แจ้งเตือนจะแสดงผลโดยไฟ LED กระพริบพร้อมทั้งส่งเสียงเตือน



รูปที่ ก.12 อุปกรณ์แจ้งเตือนเกิดการเตือน

ภาคผนวก ข.

คู่มือการติดตั้งไดรเวอร์และการลงโปรแกรม

การติดตั้งไดรเวอร์และการลงโปรแกรม

ในการเขียนโปรแกรมจะใช้โปรแกรม Arduino และส่วนของซอฟต์แวร์ ได้แบ่งการทำงานออกเป็น 2 ส่วน คือ อุปกรณ์รับรายการเครื่องดื่มและอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม ซึ่งการลงโปรแกรมจะต้องลงโปรแกรม ดังนี้

- โปรแกรม Arduino
- โปรแกรมติดตั้งอุปกรณ์รับรายการเครื่องดื่ม
- โปรแกรมติดตั้งอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

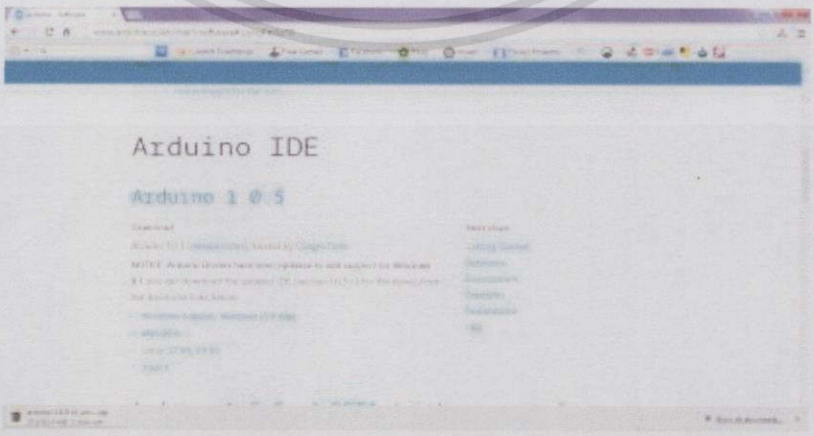
ซึ่งขั้นตอนในการลงโปรแกรม เป็นดังต่อไปนี้

1. การลงโปรแกรม Arduino

ในที่นี้เราใช้โปรแกรม Arduino IDE version 1.0.5 โดยสามารถดาวน์โหลดโปรแกรมได้จาก www.arduino.cc/en/main/software#.Uylo6vmSzTA ดังรูปที่ ข.1

รูปที่ ข.1 รูปหน้า Web page การดาวน์โหลดโปรแกรม

ทำการดาวน์โหลดโปรแกรม โดยเลือกให้เหมาะสมกับคอมพิวเตอร์ที่ใช้ ดังรูปที่ ข.2

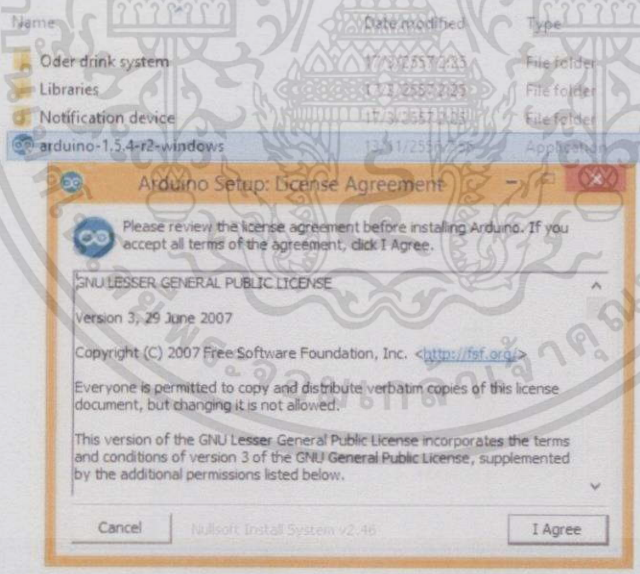


รูปที่ ข.2 รูปการดาวน์โหลดโปรแกรม

เมื่อดาวน์โหลดโปรแกรมเสร็จก็จะได้ไฟล์โปรแกรม Arduino ดังรูปที่ ข.3

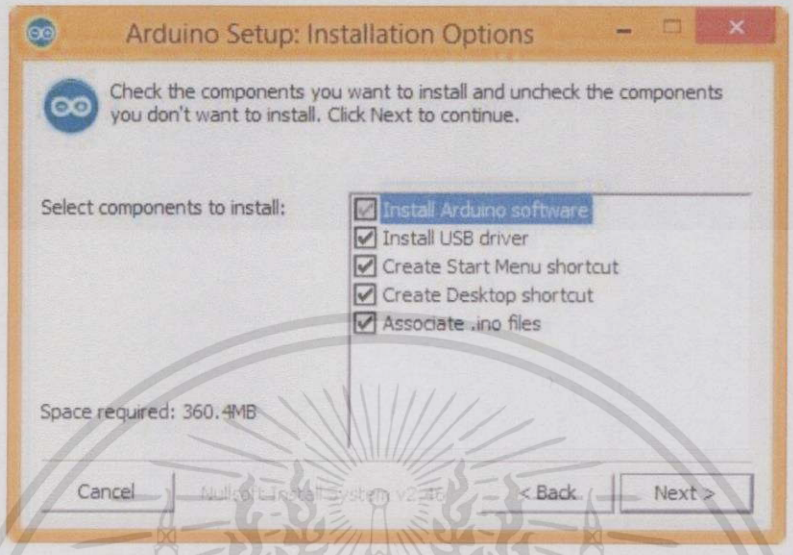
รูปที่ ข.3 แสดงไฟล์โปรแกรม Arduino

เข้าไปที่ไฟล์โปรแกรม Arduino ก็จะปรากฏหน้าต่าง Arduino Setup: License Agreement ดังรูปที่ ข.4 จากนั้นเลือก I Agree



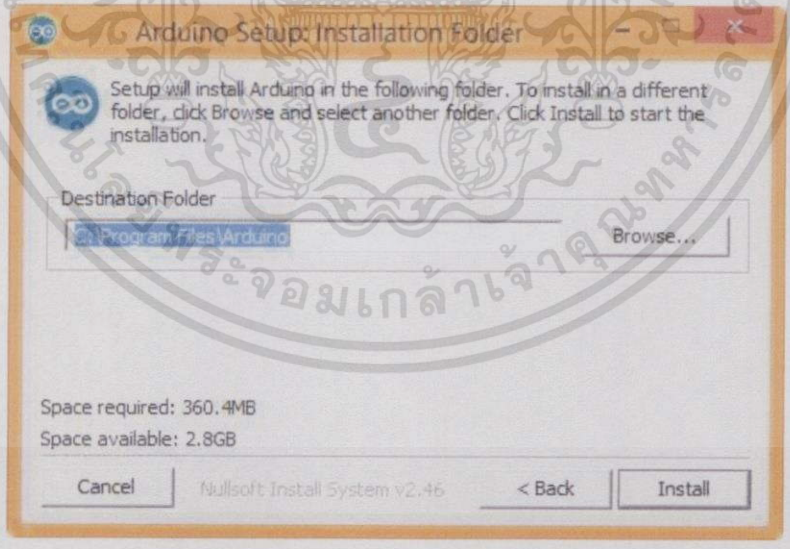
รูปที่ ข.4 ภาพแสดงหน้าต่าง Arduino Setup: License Agreement

โปรแกรมจะแสดงหน้าต่าง Arduino Setup: Installation Options ดังรูปที่ ข.5 จากนั้นเลือก Next>



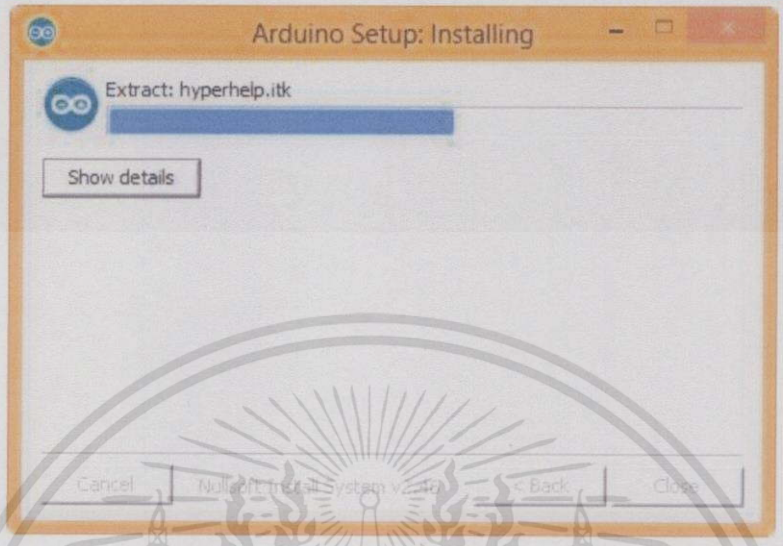
รูปที่ ข.5 ภาพแสดงหน้าต่าง Arduino Setup: Installation Options

โปรแกรมจะแสดงหน้าต่าง Arduino Setup: Installation Folder ดังรูปที่ ข.6 เพื่อให้เลือกโฟลเดอร์ที่เราต้องการเก็บไฟล์โปรแกรม Arduino จากนั้นเลือก Install



รูปที่ ข.6 ภาพแสดงหน้าต่าง Arduino Setup: Installation Folder

โปรแกรมกำลัง Install โดยจะแสดงหน้าต่าง Arduino Setup: Installing ดังรูปที่ ข.7 จากนั้นทำการรอจนกว่าโปรแกรมจะ Install เสร็จ



รูปที่ ข.7 ภาพแสดงหน้าต่าง Arduino Setup: Installing

เมื่อโปรแกรม Install เสร็จ จะปรากฏหน้าต่าง Arduino Setup: Completed ดังรูปที่ ข.8 จากนั้นเลือก Close เพื่อปิดหน้าต่าง Arduino Setup ซึ่งการลงโปรแกรม Arduino ก็จะเสร็จสมบูรณ์

รูปที่ ข.8 ภาพแสดงหน้าต่าง Arduino Setup: Completed

จากนั้นก็จะปรากฏโปรแกรม Arduino ที่หน้า Desktop ดังรูปที่ ข.9

รูปที่ ข.9 รูปโปรแกรม Arduino

เมื่อเข้าไปในโปรแกรมจะปรากฏหน้าสำหรับเขียนโปรแกรม Arduino ดังรูปที่ ข.10

รูปที่ ข.10 ภาพแสดงหน้าสำหรับเขียนโปรแกรม Arduino

2. การลงโปรแกรมติดตั้งอุปกรณ์รับรายการเครื่องดื่ม

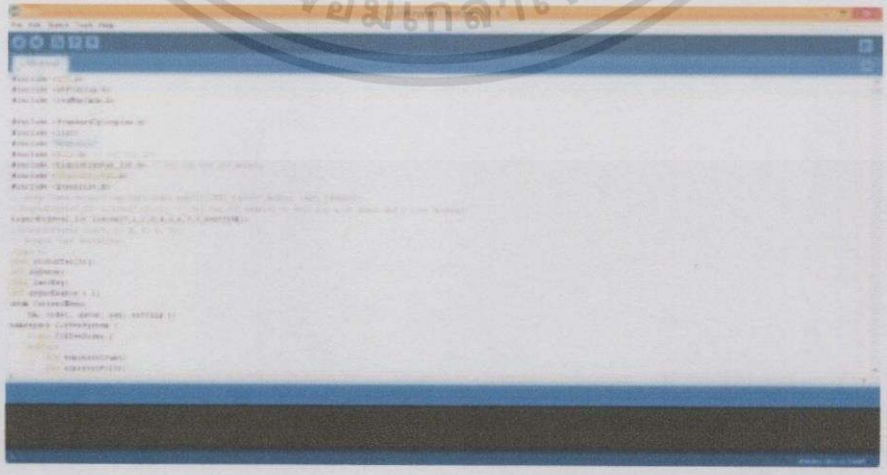
ทำการเชื่อมต่ออุปกรณ์รับรายการเครื่องดื่มกับพอร์ตของเครื่องคอมพิวเตอร์ ดังรูปที่ ข.11

รูปที่ ข.11 การเชื่อมต่ออุปกรณ์รับรายการเครื่องดื่มเข้ากับเครื่องคอมพิวเตอร์

เลือกโปรแกรมของระบบรับรายการเครื่องดื่ม ที่ต้องการจะติดตั้งกับอุปกรณ์รับรายการเครื่องดื่ม ดังรูปที่ ข.12

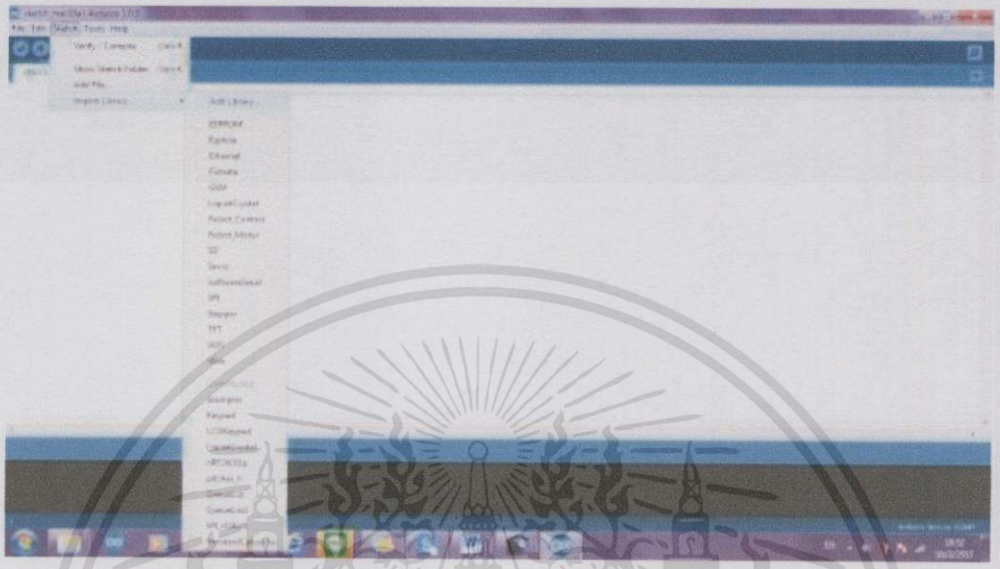
รูปที่ ข.12 โฟลเดอร์โปรแกรมระบบรับรายการเครื่องดื่ม

เมื่อทำการเลือกไฟล์โปรแกรมรับรายการเครื่องดื่ม จะปรากฏหน้าโปรแกรม Arduino ของระบบรับรายการเครื่องดื่ม ดังรูปที่ ข.13



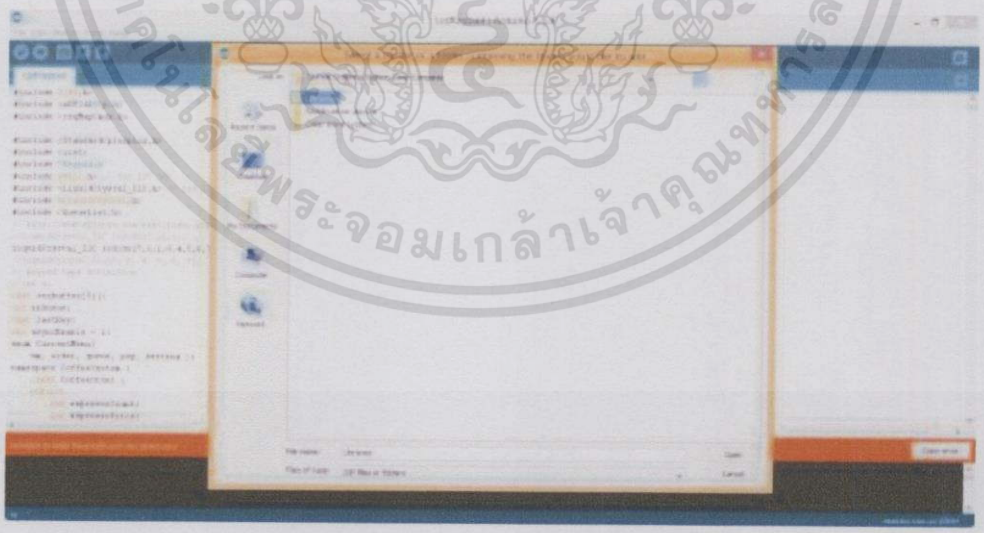
รูปที่ ข.13 ภาพแสดงโปรแกรม Arduino ของระบบรับรายการเครื่องดื่ม

เนื่องจากในโปรแกรม Arduino อาจไม่มี Library ที่ต้องการใช้เกี่ยวกับระบบ ดังนั้นจึงต้องเพิ่ม Library เข้ามาในโปรแกรม ซึ่งทำโดยการ เลือก sketch จากนั้นไปที่ Import Library และเลือก Add Library ดังรูปที่ ข.14



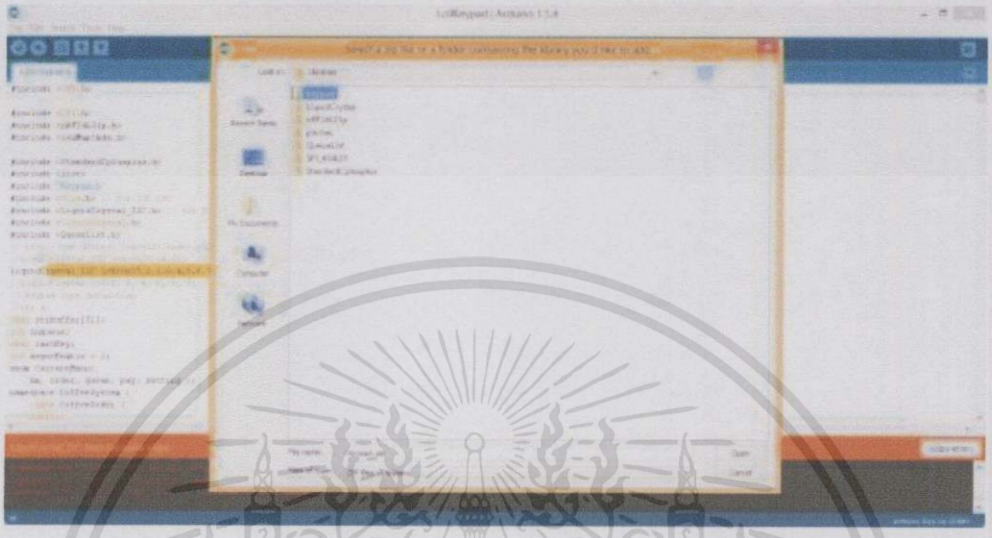
รูปที่ ข.14 ภาพแสดงขั้นตอนการเพิ่ม Library

จากนั้นทำการเลือกโฟลเดอร์ที่เก็บ Library ไว้ ซึ่งในที่นี้ Library อยู่ในโฟลเดอร์ชื่อว่า Libraries ดังรูปที่ ข.15



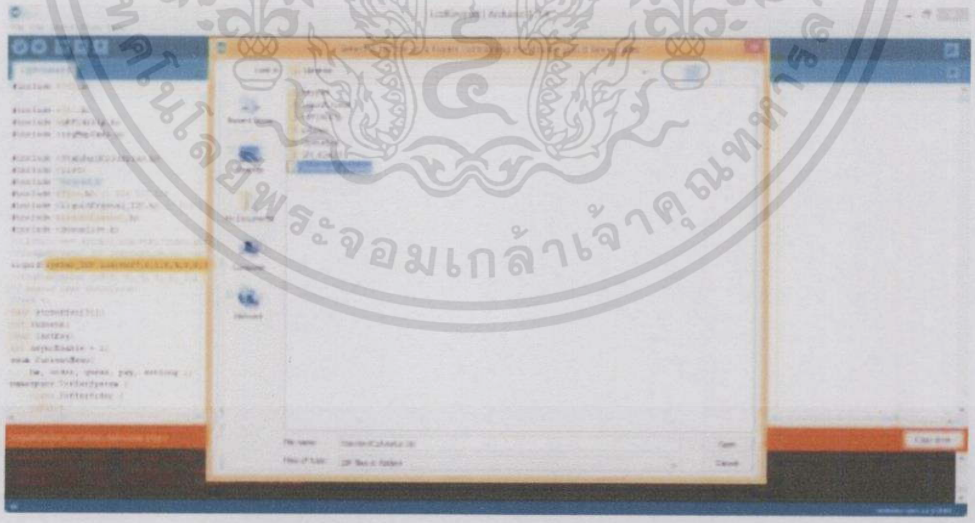
รูปที่ ข.15 ภาพแสดงโฟลเดอร์ Libraries

เมื่อเข้าไปในโฟลเดอร์ Libraries จะปรากฏไฟล์ของ Library ต่าง ๆ ที่ต้องการใช้สำหรับระบบ ดังรูปที่ ข.16



รูปที่ ข.16 ภาพไฟล์ Library

ทำการเลือกไฟล์ Library ที่เราต้องการ แล้วกด Open ดังรูปที่ ข.17 จากนั้นทำเช่นนี้ไปเรื่อย ๆ จนครบทุกไฟล์ Library ที่เราต้องการสำหรับระบบ



รูปที่ ข.17 ภาพแสดงการเพิ่ม Library

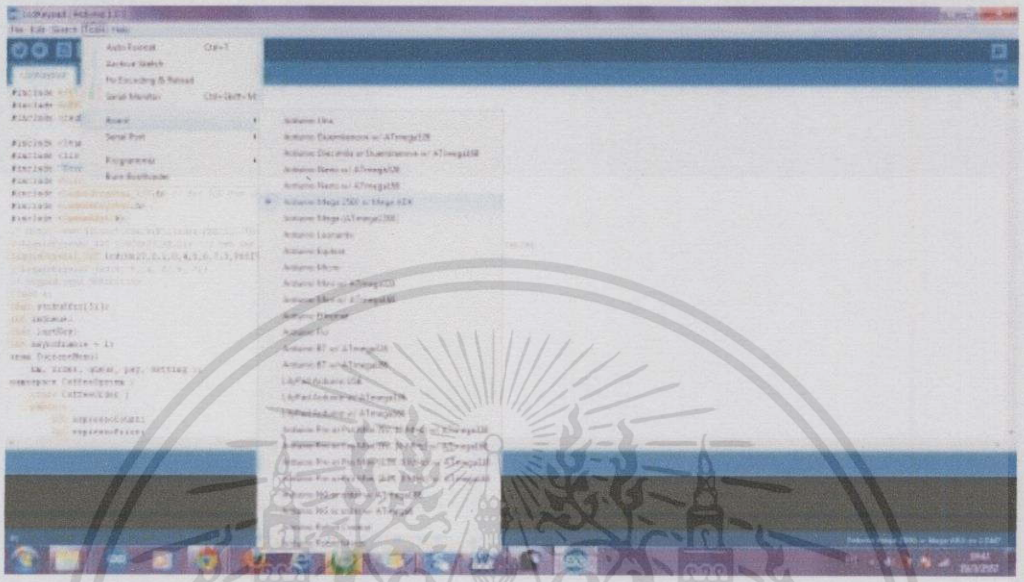
จากนั้นเพื่อตรวจสอบว่าโปรแกรมถูกต้องและพร้อมใช้งานหรือไม่ ให้ทำการ Compile โปรแกรม โดยกดที่เครื่องหมายถูก ดังรูปที่ ข.18

รูปที่ ข.18 ภาพแสดงการ Compile โปรแกรมสำหรับระบบรับรายการเครื่องดื่ม

เมื่อการ Compile โปรแกรมเสร็จสิ้น จะขึ้นว่า Done Compiling ดังรูปที่ ข.19 ซึ่งแสดงว่าโปรแกรมถูกต้องไม่มีข้อผิดพลาดอะไร พร้อมที่จะลงโปรแกรมติดตั้งเพื่อใช้งานต่อไป

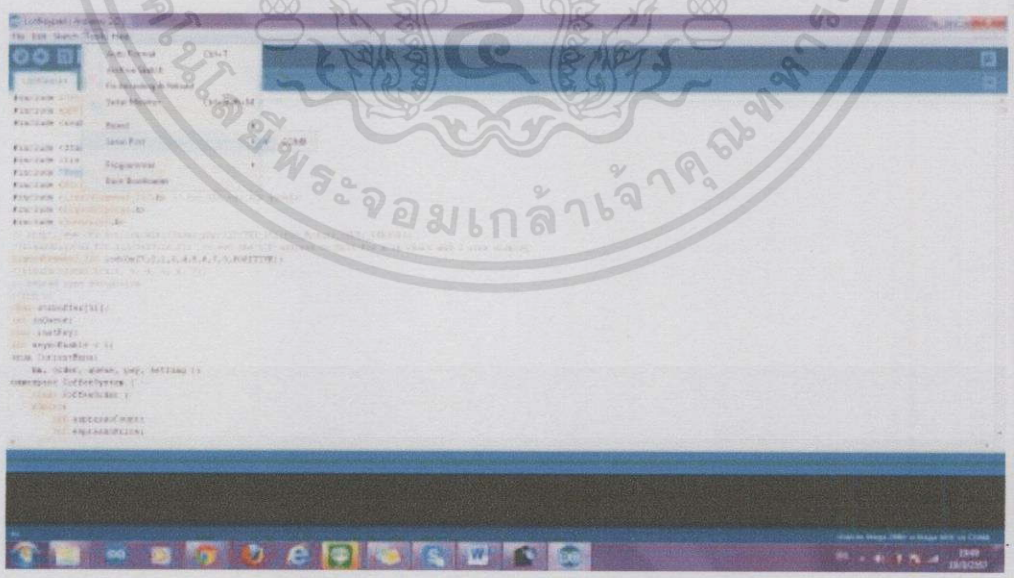
รูปที่ ข.19 ภาพแสดงการ Compile โปรแกรมสำหรับระบบรับรายการเครื่องดื่มเสร็จสิ้น

จากนั้นทำการ Upload โปรแกรม ลงอุปกรณ์รับรายการเครื่องดื่ม โดยเลือก Tools และเลือกบอร์ดไมโครคอนโทรลเลอร์ที่ใช้ ซึ่งในที่นี้อุปกรณ์รับรายการเครื่องดื่มใช้ไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 ดังรูปที่ ข.20



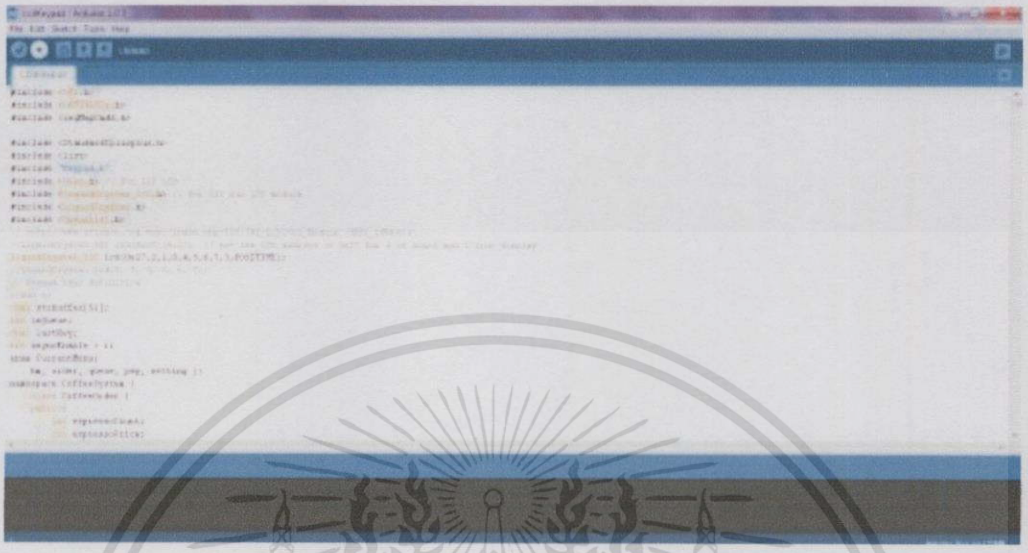
รูปที่ ข.20 ภาพแสดงการเลือกบอร์ด Arduino Mega 2560 R3 เพื่อ Upload โปรแกรม

ต่อไปทำการเลือกพอร์ตที่บอร์ดไมโครคอนโทรลเลอร์ Arduino เชื่อมต่ออยู่ ดังรูปที่ ข.21 เพื่อที่จะทำการ Upload โปรแกรม



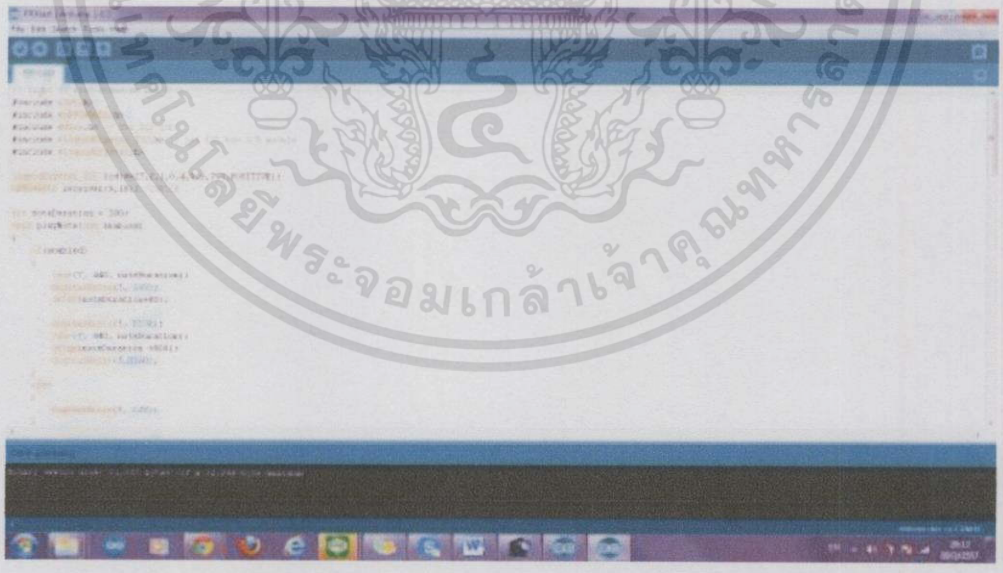
รูปที่ ข.21 ภาพแสดงการเลือกพอร์ตที่บอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 เชื่อมต่ออยู่

จากนั้นทำการ Upload โปรแกรม ลงบนอุปกรณ์รับรายการเครื่องดื่ม โดยกดที่เครื่องหมายลูกศร ดังรูปที่ ข.22



รูปที่ ข.22 ภาพแสดงการ Upload โปรแกรมลงบนอุปกรณ์รับรายการเครื่องดื่ม

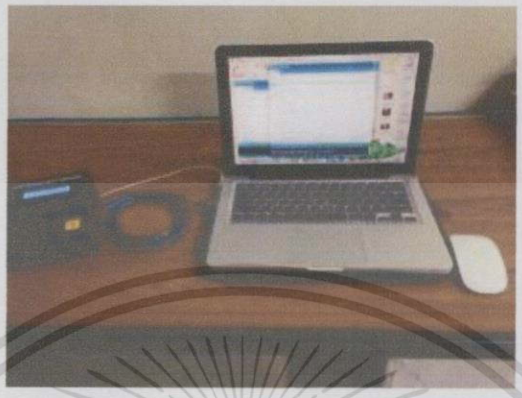
เมื่อโปรแกรม Upload เสร็จสิ้น จะขึ้นว่า Done Uploading ดังรูปที่ ข.23 ซึ่งแสดงว่าโปรแกรม Upload ลงอุปกรณ์รับรายการเครื่องดื่มเสร็จเรียบร้อยแล้ว สามารถนำไปใช้งานได้เลย



รูปที่ ข.23 ภาพแสดงการ Upload โปรแกรมลงบนอุปกรณ์รับรายการเครื่องดื่มเสร็จสิ้น

3. การลงโปรแกรมติดตั้งอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

ทำการเชื่อมต่ออุปกรณ์แจ้งเตือนในการรับเครื่องดื่มกับพอร์ตของเครื่องคอมพิวเตอร์ ดังรูปที่ ข.24

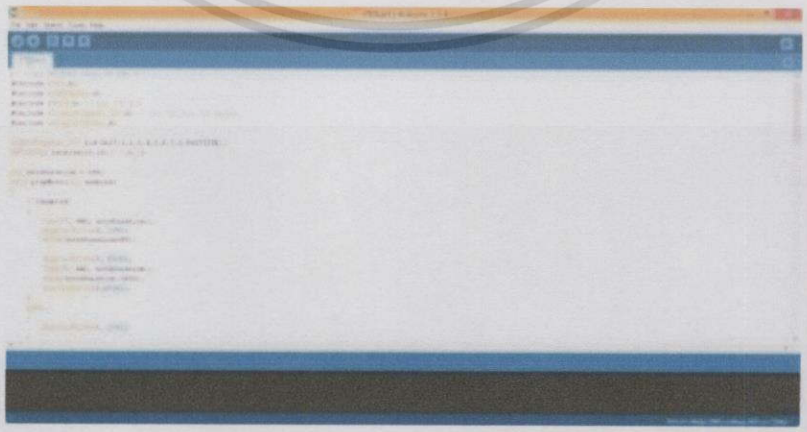


รูปที่ ข.24 การเชื่อมต่ออุปกรณ์แจ้งเตือนในการรับเครื่องดื่มเข้ากับเครื่องคอมพิวเตอร์

เลือกโปรแกรมของระบบแจ้งเตือน ที่ต้องการจะติดตั้งกับอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม ดังรูปที่ ข.25

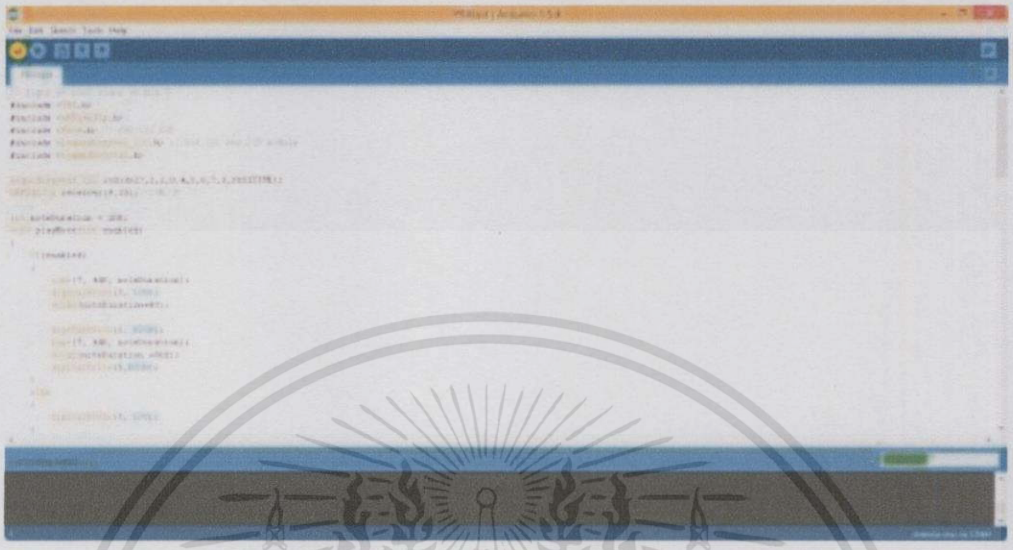
รูปที่ ข.25 โฟลเดอร์โปรแกรมระบบแจ้งเตือน

เมื่อทำการเลือกไฟล์โปรแกรมแจ้งเตือนในการรับเครื่องดื่ม จะปรากฏหน้าโปรแกรม Arduino ของระบบแจ้งเตือนในการเครื่องดื่ม ดังรูปที่ ข.26



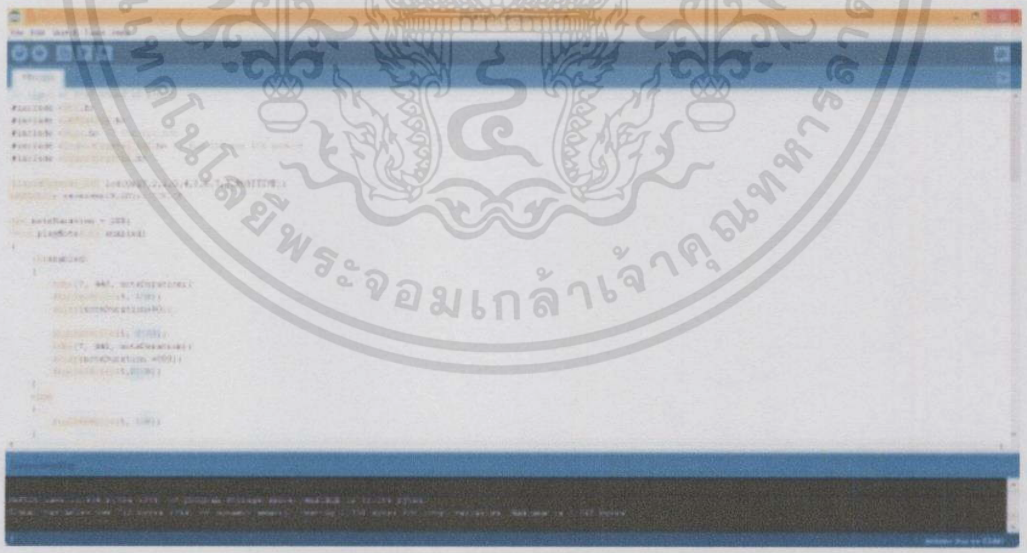
รูปที่ ข.26 ภาพแสดงโปรแกรม Arduino ของระบบแจ้งเตือนในการรับเครื่องดื่ม

จากนั้นเพื่อตรวจสอบว่าโปรแกรมถูกต้องและพร้อมใช้งานหรือไม่ ให้ทำการ Compile โปรแกรม โดยกดที่เครื่องหมายถูก ดังรูปที่ ข.27



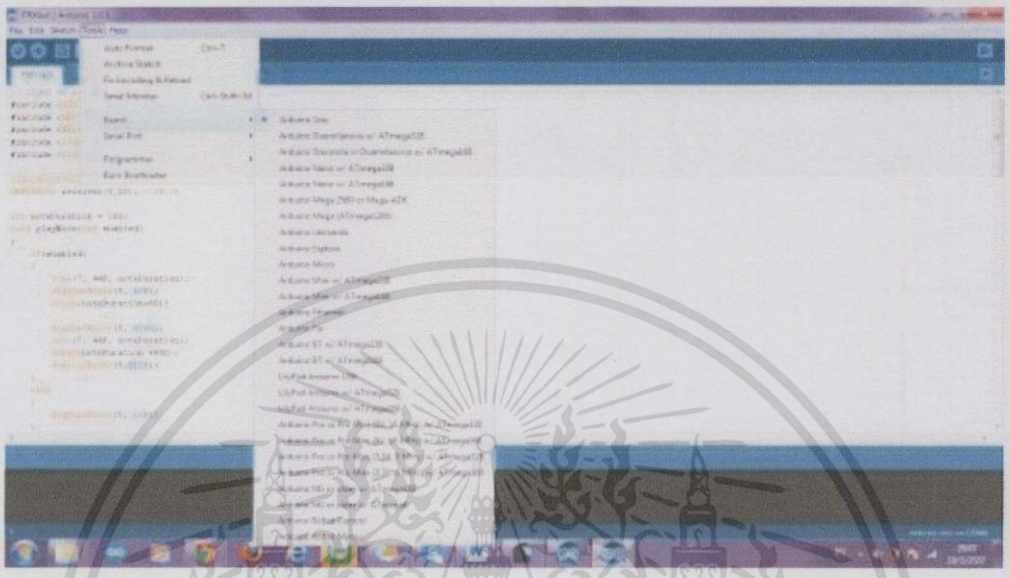
รูปที่ ข.27 ภาพแสดงการ Compile โปรแกรมสำหรับระบบแจ้งเตือนในการรับเครื่องดื่ม

เมื่อการ Compile โปรแกรมเสร็จสิ้น จะขึ้นว่า Done Compiling ดังรูปที่ ข.28 ซึ่งแสดงว่าโปรแกรมถูกต้องไม่มีข้อผิดพลาดอะไร พร้อมที่จะลงโปรแกรมติดตั้งเพื่อใช้งานต่อไป



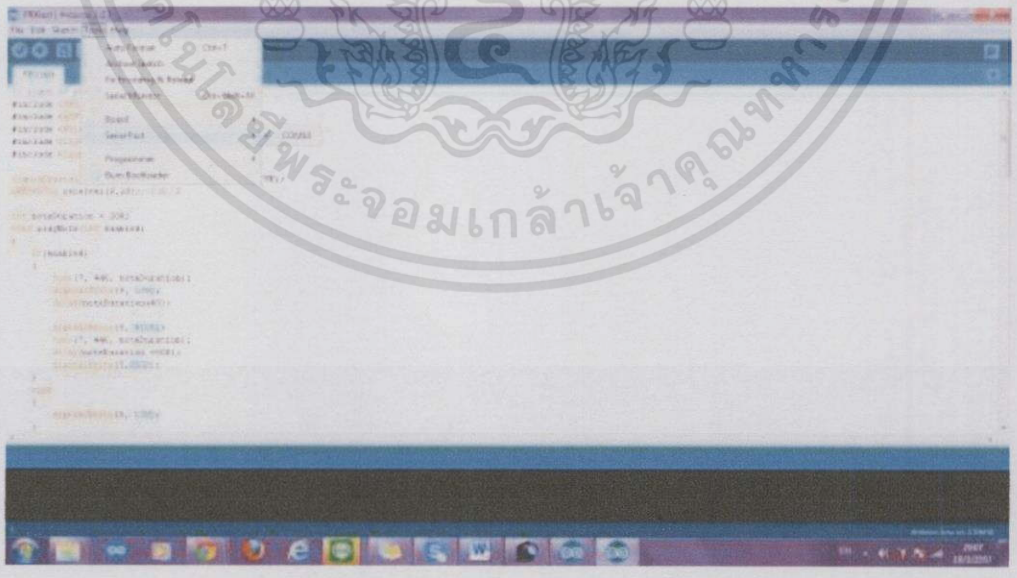
รูปที่ ข.28 ภาพแสดงการ Compile โปรแกรมสำหรับระบบแจ้งเตือนในการรับเครื่องดื่มเสร็จสิ้น

จากนั้นทำการ Upload โปรแกรม ลงอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม โดยเลือก Tools และเลือกบอร์ดไมโครคอนโทรลเลอร์ที่ใช้ ซึ่งในที่นี้อุปกรณ์แจ้งเตือนในการรับเครื่องดื่มใช้ไมโครคอนโทรลเลอร์ Arduino Uno R3 ดังรูปที่ ข.29



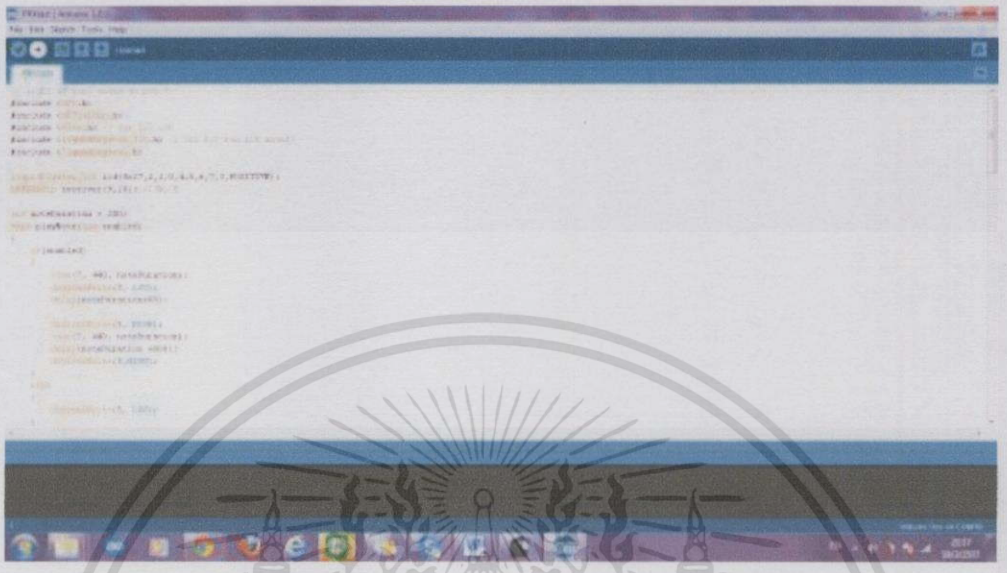
รูปที่ ข.29 ภาพแสดงการเลือกบอร์ด Arduino Uno R3 เพื่อ Upload โปรแกรม

ต่อไปทำการเลือกพอร์ตที่บอร์ดไมโครคอนโทรลเลอร์ Arduino เชื่อมต่ออยู่ ดังรูปที่ ข.30 เพื่อที่จะทำการ Upload โปรแกรม



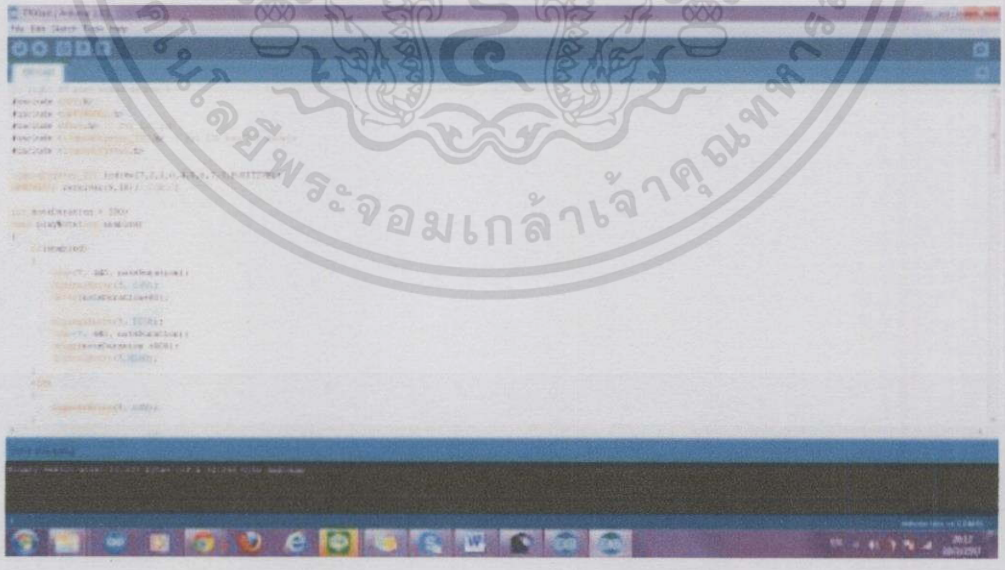
รูปที่ ข.30 ภาพแสดงการเลือกพอร์ตที่บอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3 เชื่อมต่ออยู่

จากนั้นทำการ Upload โปรแกรม ลงบนอุปกรณ์รับรายการเครื่องดื่ม โดยกดที่เครื่องหมายลูกศร ดังรูปที่ ข.31



รูปที่ ข.31 ภาพแสดงการ Upload โปรแกรมลงบนอุปกรณ์แจ้งเตือนในการรับเครื่องดื่ม

เมื่อโปรแกรม Upload เสร็จสิ้น จะขึ้นว่า Done Uploading ดังรูปที่ ข.32 ซึ่งแสดงว่าโปรแกรม Upload ลงอุปกรณ์แจ้งเตือนในการรับเครื่องดื่มเสร็จเรียบร้อยแล้ว สามารถนำไปใช้งานได้เลย

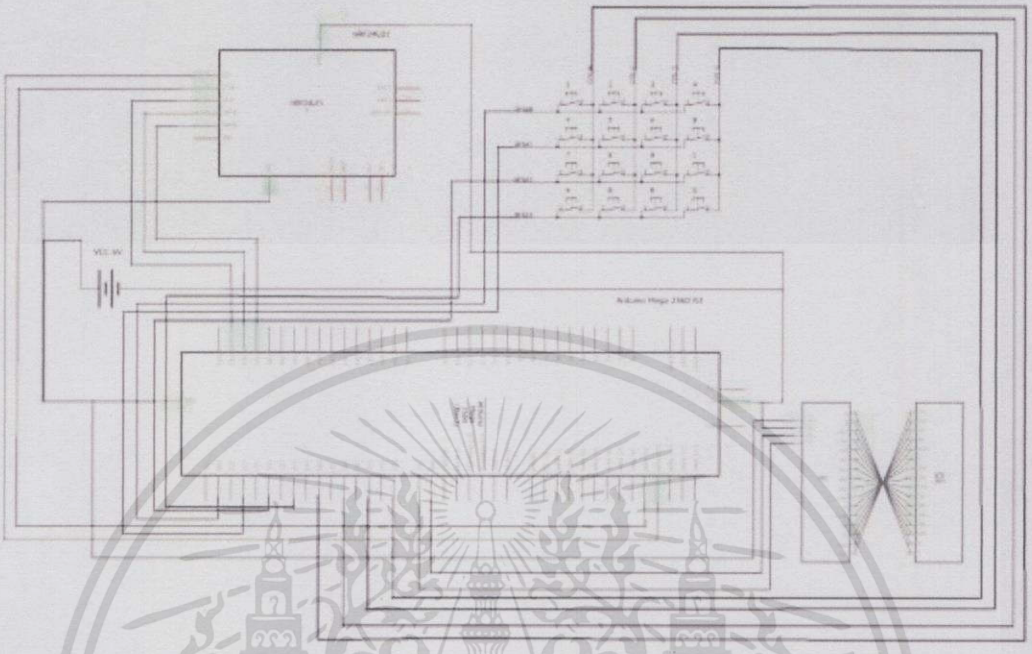


รูปที่ ข.32 ภาพแสดงการ Upload โปรแกรมลงบนอุปกรณ์แจ้งเตือนในการรับเครื่องดื่มเสร็จสิ้น

ภาคผนวก ค.

วงจรอิเล็กทรอนิกส์

วงจรอุปกรณ์ภาคส่งสัญญาณ



รูปที่ ค.1 ภาพวงจรอุปกรณ์ภาคส่งสัญญาณ

จากรูปที่ ค.1 วงจรภาคส่งสัญญาณมีอุปกรณ์ ดังนี้ บอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3, Battery, จอแสดงผล LCD ขนาด 20x4, ปุ่มกด และอุปกรณ์รับ-ส่งสัญญาณไร้สาย nRF24L01 ซึ่งการเชื่อมต่อวงจรมีรายละเอียด ดังต่อไปนี้

ส่วนที่ 1 เชื่อมต่อ Battery 6V กับบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3

ส่วนที่ 2 เชื่อมต่ออุปกรณ์ลด Pin I2C กับจอแสดงผล LCD ขนาด 20x4 โดย

Pin 1 ขา Gnd ต่อกับ Pin 1 ขา Gnd ของ LCD

Pin 2 ขาแรงดันไฟเลี้ยง Vcc (+4.5V +5.5V) สำหรับ ต่อกับ Pin 2 Vcc ของ LCD

Pin 3 นำไปต่อกับวงจรปรับระดับแรงดันได้ ต่อกับ Pin 3 VEE ของ LCD

Pin 4 ต่อกับ Pin 4 RS หรือ Register ของ LCD

Pin 5 ต่อกับ Pin 5 RW หรือ Read not Write ของ LCD

Pin 6 ต่อกับ Pin 6 EN หรือ Enable ของ LCD

Pin 7 ต่อกับ Pin 7 (DB0 หรือ Data Bit 0) ของ LCD ซึ่งไม่ได้ใช้งาน

Pin 8 ต่อกับ Pin 8 (DB1 หรือ Data Bit 1) ของ LCD ซึ่งไม่ได้ใช้งาน

Pin 9 ต่อกับ Pin 9 (DB2 หรือ Data Bit 2) ของ LCD ซึ่งไม่ได้ใช้งาน

Pin 10 ต่อกับ Pin 10 (DB3 หรือ Data Bit 3) ของ LCD ซึ่งไม่ได้ใช้งาน

Pin 11 ต่อกับ Pin 11 (DB4 หรือ Data Bit 4) ของ LCD

Pin 12 ต่อกับ Pin 12 (DB5 หรือ Data Bit 5) ของ LCD

Pin 13 ต่อกับ Pin 13 (DB6 หรือ Data Bit 6) ของ LCD

Pin 14 ต่อกับ Pin 14 (DB7 หรือ Data Bit 7) ของ LCD

Pin 15 LED+ เป็นขา Vcc สำหรับ LED backlight ต่อผ่าน +5V ผ่าน Jumper

Pin 16 LED- เป็นขา Gnd สำหรับ LED backlight ควบคุมเปิด-ปิด

ส่วนที่ 3 เชื่อมต่ออุปกรณ์ลด Pin I2C กับบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3 โดย

Pin Gnd ต่อกับ Gnd ของ Arduino Mega 2560 R3

Pin Vcc +5V (+4.5 +5.5) ต่อกับ Vcc +5V ของ Arduino Mega 2560 R3

Pin SDA serial data ของ I2C ต่อกับ Pin 20 SDA ของ Arduino Mega 2560 R3

Pin SCL serial clock ของ I2C ต่อกับ Pin 21 SCL ของ Arduino Mega 2560 R3

ส่วนที่ 4 เชื่อมต่อปุ่มกดกับบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3

Pin 1 ของปุ่มกด ต่อกับ Pin 22 ของ Arduino Mega 2560 R3

Pin 2 ของปุ่มกด ต่อกับ Pin 24 ของ Arduino Mega 2560 R3

Pin 3 ของปุ่มกด ต่อกับ Pin 26 ของ Arduino Mega 2560 R3

Pin 4 ของปุ่มกด ต่อกับ Pin 28 ของ Arduino Mega 2560 R3

Pin 5 ของปุ่มกด ต่อกับ Pin 30 ของ Arduino Mega 2560 R3

Pin 6 ของปุ่มกด ต่อกับ Pin 32 ของ Arduino Mega 2560 R3

Pin 7 ของปุ่มกด ต่อกับ Pin 34 ของ Arduino Mega 2560 R3

Pin 8 ของปุ่มกด ต่อกับ Pin 36 ของ Arduino Mega 2560 R3

ส่วนที่ 5 เชื่อมต่อโมดูล nRF24L01 กับบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3

Pin 1 Gnd ต่อกับ Gnd ของ Arduino Mega 2560 R3

Pin 2 Vcc 3.3V ต่อกับ Vcc 3.3V ของ Arduino Mega 2560 R3

Pin 3 CE ต่อกับ Pin 10 ของ Arduino Mega 2560 R3

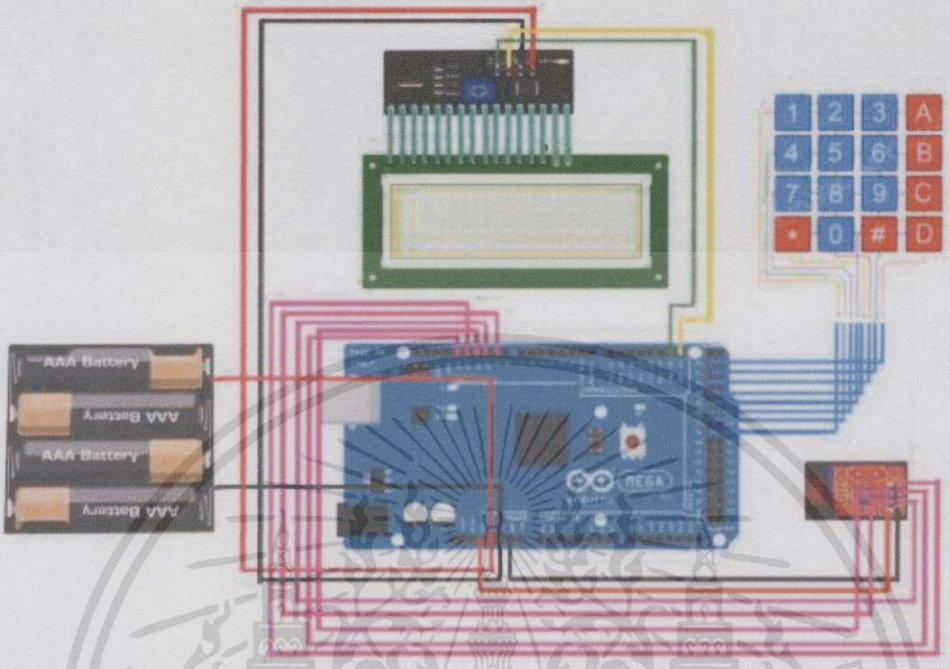
Pin 4 CSN ต่อกับ Pin 9 ของ Arduino Mega 2560 R3

Pin 5 SCK ต่อกับ Pin 52 SCK ของ Arduino Mega 2560 R3

Pin 6 MOSI ต่อกับ Pin 51 MOSI ของ Arduino Mega 2560 R3

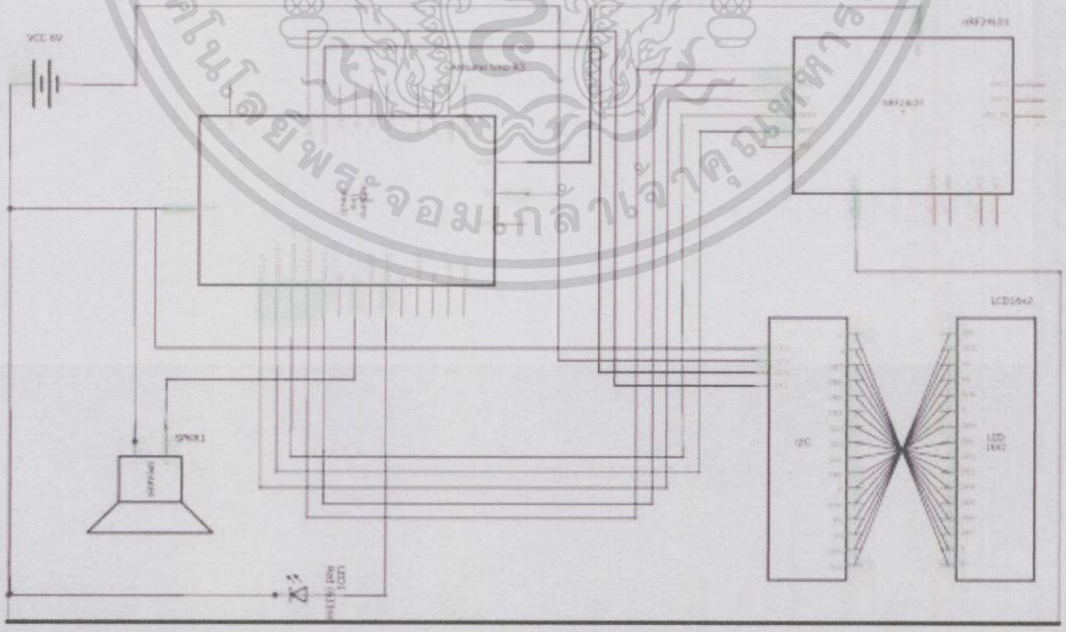
Pin 7 MISO ต่อกับ Pin 50 MISO ของ Arduino Mega 2560 R3

แผนภาพอุปกรณ์ภาคส่งสัญญาณ



รูปที่ ค.2 แผนภาพอุปกรณ์ภาคส่งสัญญาณ

วงจรอุปกรณ์ภาครับสัญญาณ



รูปที่ ค.3 ภาพวงจรอุปกรณ์ภาครับสัญญาณ

จากรูปที่ ค.3 วงจรภาครับสัญญาณมีอุปกรณ์ ดังนี้ บอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3, Battery, จอแสดงผล LCD ขนาด 16x2, อุปกรณ์รับ-ส่งสัญญาณไร้สาย nRF24L01, หลอดไฟ LED, ลำโพง ซึ่งการเชื่อมต่อวงจรมีรายละเอียด ดังต่อไปนี้

ส่วนที่ 1 เชื่อมต่อ Battery 6V กับบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3

ส่วนที่ 2 เชื่อมต่ออุปกรณ์ลด Pin I2C กับ จอแสดงผล LCD ขนาด 16x2 โดย

Pin 1 ขา Gnd ต่อกับ Pin 1 ขา Gnd ของ LCD

Pin 2 ขาแรงดันไฟเลี้ยง Vcc (+4.5V +5.5V) สำหรับ ต่อกับ Pin 2 Vcc ของ LCD

Pin 3 นำไปต่อกับวงจรปรับระดับแรงดันได้ ต่อกับ Pin 3 VEE ของ LCD

Pin 4 ต่อกับ Pin 4 RS หรือ Register ของ LCD

Pin 5 ต่อกับ Pin 5 RW หรือ Read not Write ของ LCD

Pin 6 ต่อกับ Pin 6 EN หรือ Enable ของ LCD

Pin 7 ต่อกับ Pin 7 (DB0 หรือ Data Bit 0) ของ LCD ซึ่งไม่ได้ใช้งาน

Pin 8 ต่อกับ Pin 8 (DB1 หรือ Data Bit 1) ของ LCD ซึ่งไม่ได้ใช้งาน

Pin 9 ต่อกับ Pin 9 (DB2 หรือ Data Bit 2) ของ LCD ซึ่งไม่ได้ใช้งาน

Pin 10 ต่อกับ Pin 10 (DB3 หรือ Data Bit 3) ของ LCD ซึ่งไม่ได้ใช้งาน

Pin 11 ต่อกับ Pin 11 (DB4 หรือ Data Bit 4) ของ LCD

Pin 12 ต่อกับ Pin 12 (DB5 หรือ Data Bit 5) ของ LCD

Pin 13 ต่อกับ Pin 13 (DB6 หรือ Data Bit 6) ของ LCD

Pin 14 ต่อกับ Pin 14 (DB7 หรือ Data Bit 7) ของ LCD

Pin 15 LED+ เป็นขา Vcc สำหรับ LED backlight ต่อผ่าน +5V ผ่าน Jumper

Pin 16 LED- เป็นขา Gnd สำหรับ LED backlight ควบคุมเปิด-ปิด

ส่วนที่ 3 เชื่อมต่ออุปกรณ์ลด Pin I2C กับบอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3 โดย

Pin Gnd ต่อกับ Gnd ของ Arduino Mega 2560 R3

Pin Vcc +5V (+4.5 +5.5) ต่อกับ Vcc +5V ของ Arduino Uno R3

Pin SDA serial data ของ I2C ต่อกับ Pin A4 SDA ของ Uno R3

Pin SCL serial clock ของ I2C ต่อกับ Pin A5 SCL ของ Arduino Uno R3

ส่วนที่ 4 เชื่อมต่อโมดูล nRF24L01 กับบอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3

Pin 1 Gnd ต่อกับ Gnd ของ Arduino Uno R3

Pin 2 Vcc 3.3V ต่อกับ Vcc 3.3V ของ Arduino Uno R3

Pin 3 CE ต่อกับ Pin 10 ของ Arduino Uno R3

Pin 4 CSN ต่อกับ Pin 9 ของ Arduino Uno R3

Pin 5 SCK ต่อกับ Pin 13 SCK ของ Arduino Uno R3

Pin 6 MOSI ต่อกับ Pin 11 MOSI ของ Arduino Uno R3

Pin 7 MISO ต่อกับ Pin 12 MISO ของ Arduino Uno R3

ส่วนที่ 5 เชื่อมต่อหลอดไฟ LED กับบอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3

Pin Gnd ต่อกับ Gnd ของ Arduino Uno R3

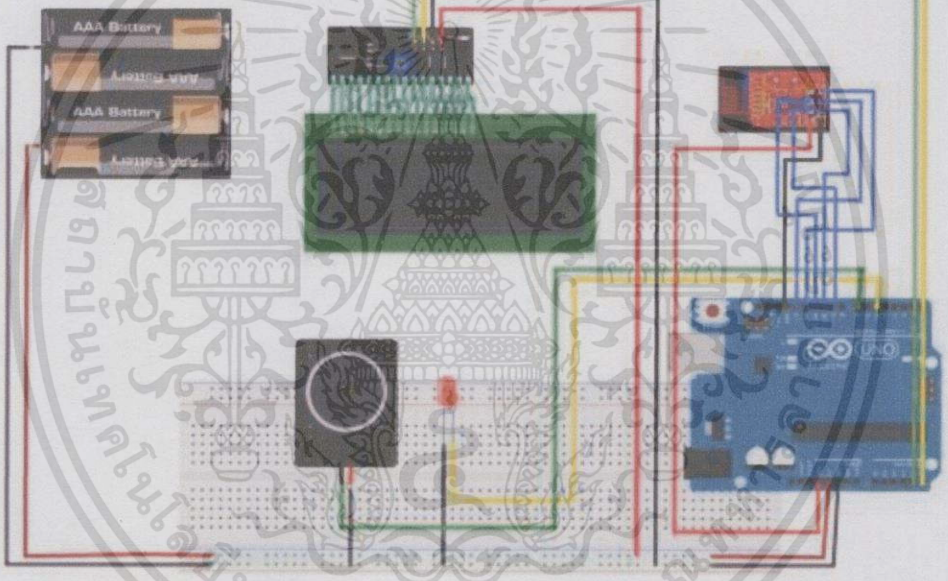
Pin Vcc ต่อกับ Pin 5 ของ Arduino Uno R3

ส่วนที่ 6 เชื่อมต่อลำโพงกับบอร์ดไมโครคอนโทรลเลอร์ Arduino Uno R3

Pin Gnd ต่อกับ Gnd ของ Arduino Uno R3

Pin Vcc ต่อกับ Pin 7 ของ Arduino Uno R3

แผนภาพอุปกรณ์ภาครับสัญญาณ



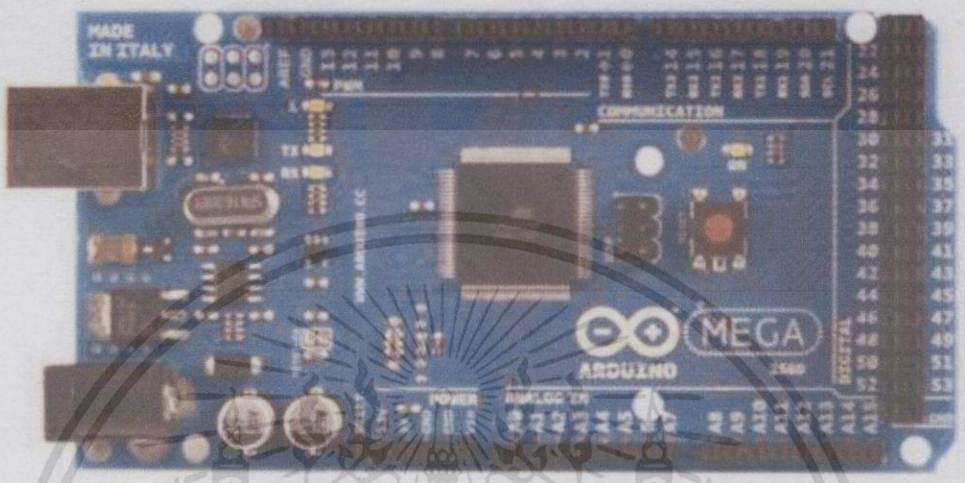
รูปที่ ค.4 แผนภาพอุปกรณ์ภาครับสัญญาณ

ภาคผนวก ง.

รายละเอียดของอุปกรณ์

ไมโครคอนโทรลเลอร์ Arduino Mega 2560 R3

Arduino MEGA 2560



CE

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

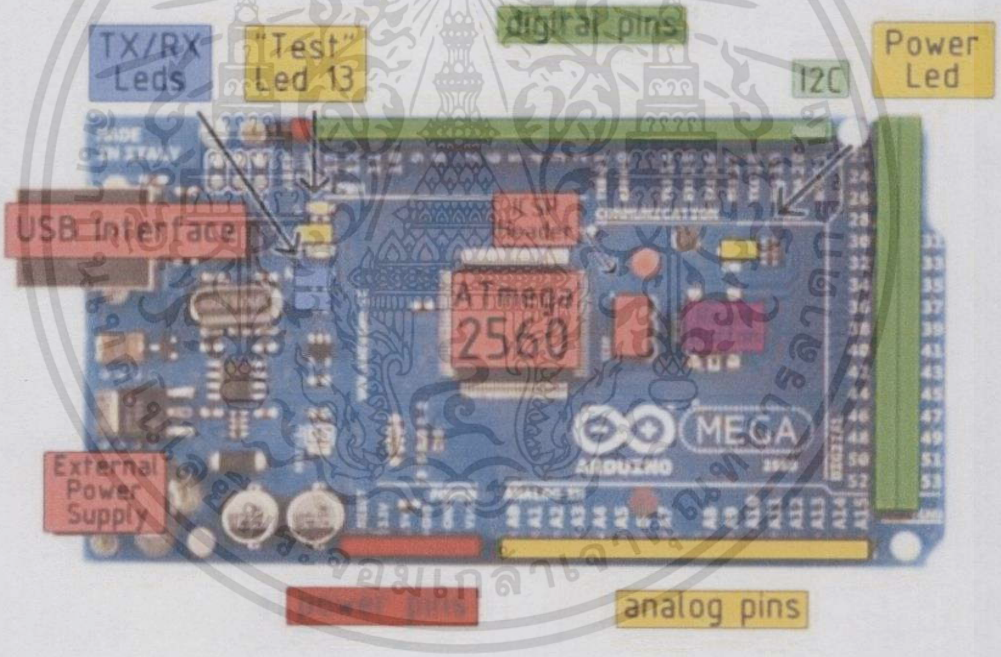
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

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)
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

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

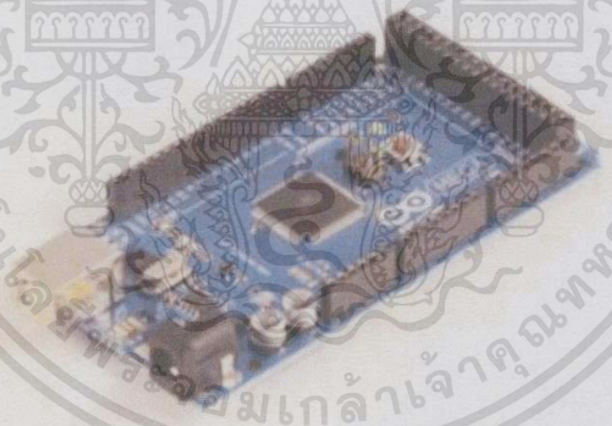
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.



Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line, see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I²C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install Windows Install Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

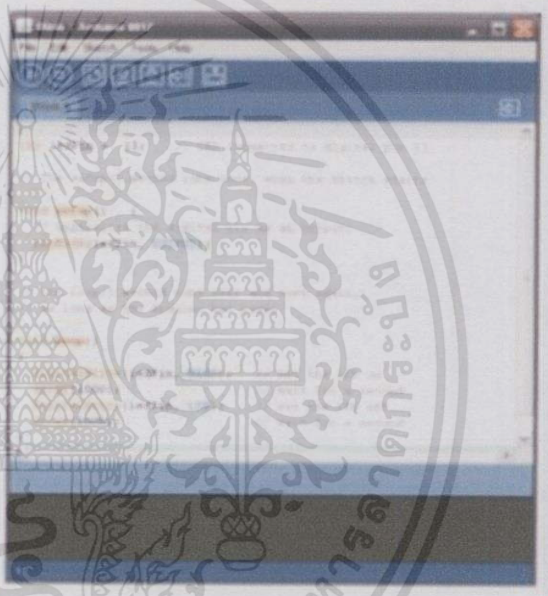
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your skecth you'll see something very close to the screenshot on the right.

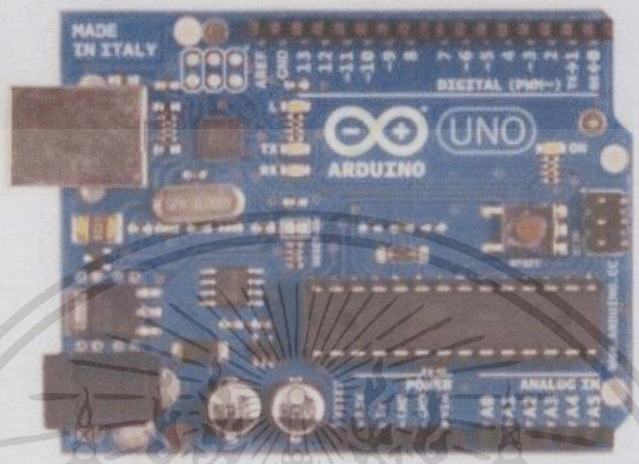
In **Tools>Board** select MEGA

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.



ไมโครคอนโทรลเลอร์ Arduino Uno R3

Arduino UNO



CE

Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Index

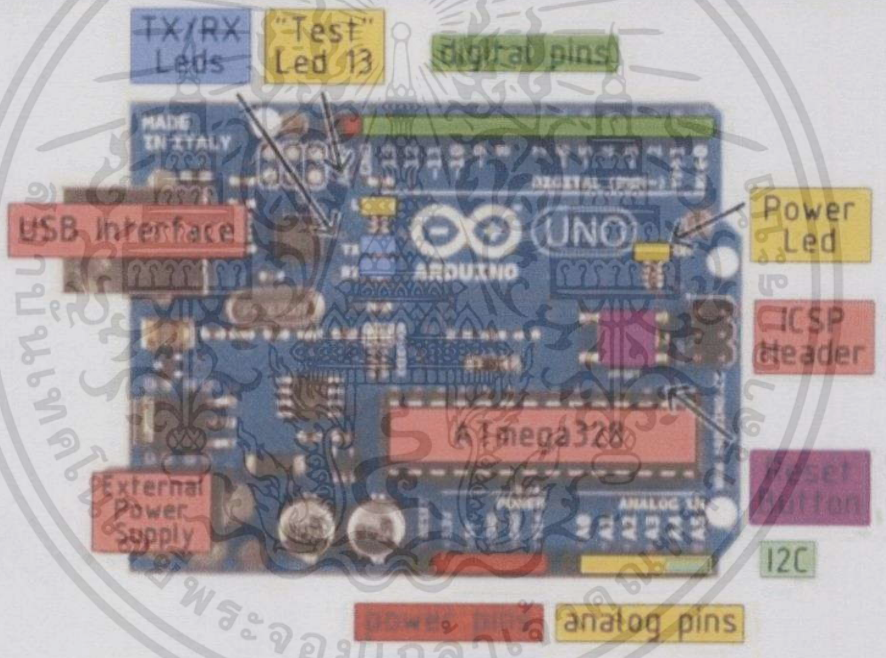
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0.5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required.

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line, see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

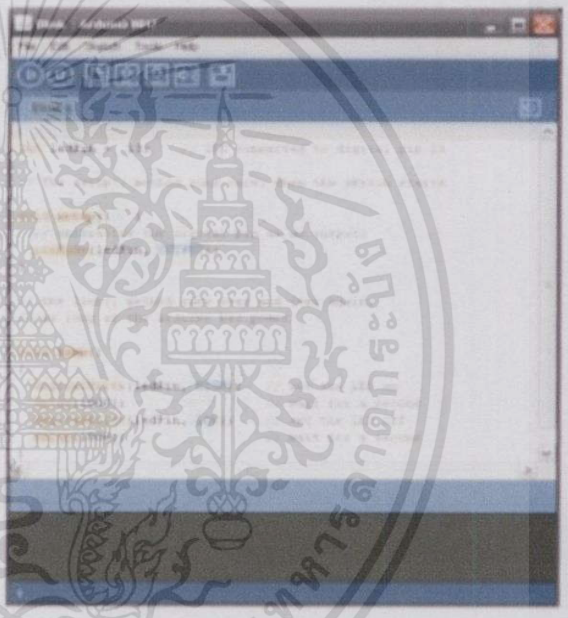
Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink



Once you have your skecth you'll see something very close to the screenshot on the right.

In **Tools>Board** select

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.

nRF24L01+ Product Specification

NORDIC SEMICONDUCTOR

1 Introduction

The nRF24L01+ is a single chip 2.4GHz transceiver with an embedded baseband protocol engine (Enhanced ShockBurst™), suitable for ultra low power wireless applications. The nRF24L01+ is designed for operation in the world wide ISM frequency band at 2.400 - 2.4835GHz.

To design a radio system with the nRF24L01+, you simply need an MCU (microcontroller) and a few external passive components.

You can operate and configure the nRF24L01+ through a Serial Peripheral Interface (SPI). The register map, which is accessible through the SPI, contains all configuration registers in the nRF24L01+ and is accessible in all operation modes of the chip.

The embedded baseband protocol engine (Enhanced ShockBurst™) is based on packet communication and supports various modes from manual operation to advanced autonomous protocol operation. Internal FIFOs ensure a smooth data flow between the radio front end and the system's MCU. Enhanced ShockBurst™ reduces system cost by handling all the high speed link layer operations.

The radio front end uses GFSK modulation. It has user configurable parameters like frequency channel, output power and air data rate. nRF24L01+ supports an air data rate of 250 kbps, 1 Mbps and 2Mbps. The high air data rate combined with two power saving modes make the nRF24L01+ very suitable for ultra low power designs.

nRF24L01+ is drop-in compatible with nRF24L01 and on-air compatible with nRF2401A, nRF2402, nRF24E1 and nRF24E2. Intermodulation and wideband blocking values in nRF24L01+ are much improved in comparison to the nRF24L01 and the addition of internal filtering to nRF24L01+ has improved the margins for meeting RF regulatory standards.

Internal voltage regulators ensure a high Power Supply Rejection Ratio (PSRR) and a wide power supply range.

1.1 Features

Features of the nRF24L01+ include:

- Radio
  - Worldwide 2.4GHz ISM band operation
  - 126 RF channels
  - Common RX and TX interface
  - GFSK modulation
  - 250kbps, 1 and 2Mbps air data rate
  - 1MHz non-overlapping channel spacing at 1Mbps
  - 2MHz non-overlapping channel spacing at 2Mbps
- Transmitter
  - Programmable output power: 0, -6, -12 or -18dBm
  - 11.3mA at 0dBm output power
- Receiver
  - Fast AGC for improved dynamic range
  - Integrated channel filters
  - 13.5mA at 2Mbps
  - -82dBm sensitivity at 2Mbps
  - -85dBm sensitivity at 1Mbps
  - -94dBm sensitivity at 250kbps
- RF Synthesizer
  - Fully integrated synthesizer
  - No external loop filter, VCO varactor diode or resonator
  - Accepts low cost ±60ppm 16MHz crystal
- Enhanced ShockBurst™
  - 1 to 32 bytes dynamic payload length
  - Automatic packet handling
  - Auto packet transaction handling
  - 6 data pipe MultiCeiver™ for 1:6 star networks
- Power Management
  - Integrated voltage regulator
  - 1.9 to 3.6V supply range
  - Idle modes with fast start-up times for advanced power management
  - 26µA Standby-I mode, 900nA power down mode
  - Max 1.5ms start-up from power down mode
  - Max 130us start-up from standby-I mode
- Host Interface
  - 4-pin hardware SPI
  - Max 10Mbps
  - 3 separate 32 bytes TX and RX FIFOs
  - 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

1.2 Block diagram

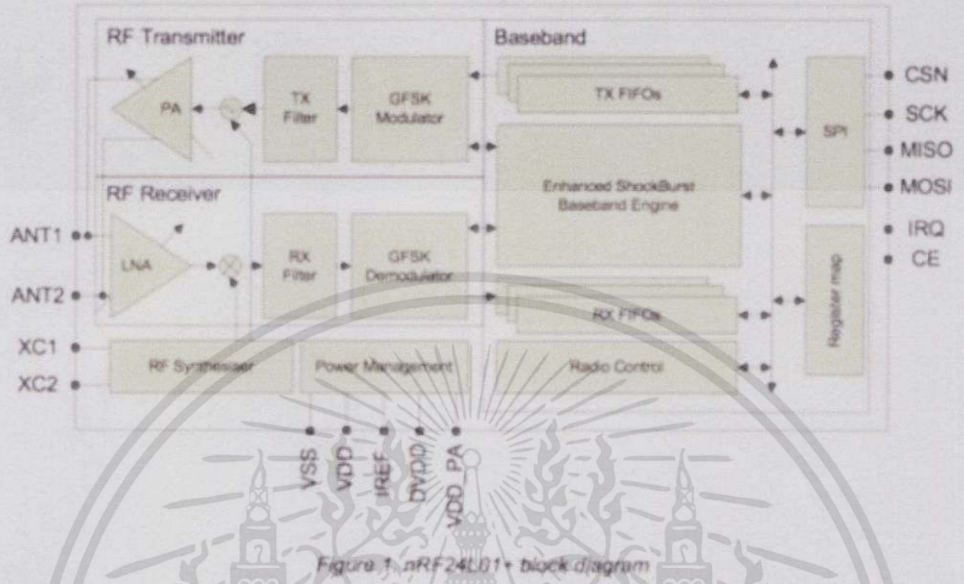


Figure 1. nRF24L01+ block diagram

2 Pin Information

2.1 Pin assignment

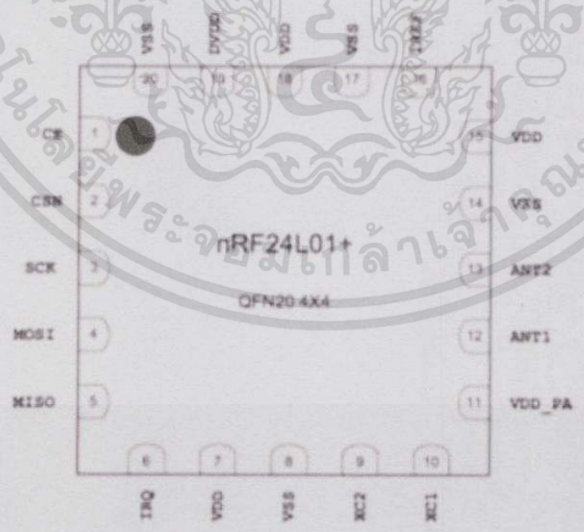


Figure 2. nRF24L01+ pin assignment (top view) for the QFN20 4x4 package

2.2 Pin functions

Pin	Name	Pin function	Description
1	CE	Digital Input	Chip Enable Activates RX or TX mode
2	CSN	Digital Input	SPI Chip Select
3	SCK	Digital Input	SPI Clock
4	MOSI	Digital Input	SPI Slave Data Input
5	MISO	Digital Output	SPI Slave Data Output, with tri-state option
6	IRQ	Digital Output	Maskable interrupt pin. Active low
7	VDD	Power	Power Supply (+1.9V - +3.6V DC)
8	VSS	Power	Ground (0V)
9	XC2	Analog Output	Crystal Pin 2
10	XC1	Analog Input	Crystal Pin 1
11	VDD_PA	Power Output	Power Supply Output (+1.8V) for the internal nRF24L01+ Power Amplifier. Must be connected to ANT1 and ANT2 as shown in Figure 32.
12	ANT1	RF	Antenna interface 1
13	ANT2	RF	Antenna interface 2
14	VSS	Power	Ground (0V)
15	VDD	Power	Power Supply (+1.9V - +3.6V DC)
16	IREF	Analog Input	Reference current. Connect a 22kΩ resistor to ground. See Figure 32.
17	VSS	Power	Ground (0V)
18	VDD	Power	Power Supply (+1.9V - +3.6V DC)
19	DVDD	Power Output	Internal digital supply output for de-coupling purposes. See Figure 32.
20	VSS	Power	Ground (0V)

Table 1. nRF24L01+ pin function

3 Absolute maximum ratings

Note: Exceeding one or more of the limiting values may cause permanent damage to nRF24L01+.

Operating conditions	Minimum	Maximum	Units
Supply voltages			
VDD	-0.3	3.6	V
VSS		0	V
Input voltage			
V_I	-0.3	5.25	V
Output voltage			
V_O	VSS to VDD	VSS to VDD	
Total Power Dissipation			
P_D (T_A=85°C)		60	mW
Temperatures			
Operating Temperature	-40	+85	°C
Storage Temperature	-40	+125	°C

Table 2. Absolute maximum ratings

4 Operating conditions

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
VDD	Supply voltage		1.9	3.0	3.6	V
VDD	Supply voltage if input signals >3.6V		2.7	3.0	3.3	V
TEMP	Operating Temperature		-40	+27	+85	°C

Table 3. Operating conditions

6 Radio Control

This chapter describes the nRF24L01+ radio transceiver's operating modes and the parameters used to control the radio.

The nRF24L01+ has a built-in state machine that controls the transitions between the chip's operating modes. The state machine takes input from user defined register values and internal signals.

6.1 Operational Modes

You can configure the nRF24L01+ in power down, standby, RX or TX mode. This section describes these modes in detail.

6.1.1 State diagram

The state diagram in Figure 4. shows the operating modes and how they function. There are three types of distinct states highlighted in the state diagram:

- **Recommended operating mode:** is a recommended state used during normal operation.
- **Possible operating mode:** is a possible operating state, but is not used during normal operation.
- **Transition state:** is a time limited state used during start up of the oscillator and settling of the PLL.

When the VDD reaches 1.9V or higher nRF24L01+ enters the Power on reset state where it remains in reset until entering the Power Down mode.

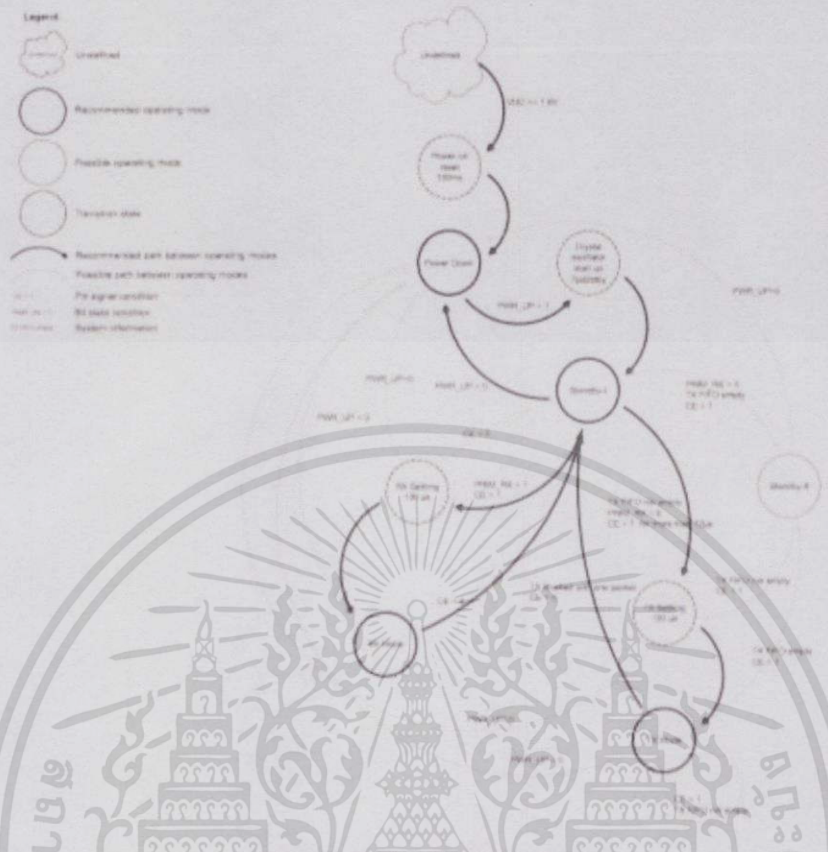


Figure 4. Radio control state diagram

6.1.2 Power Down Mode

In power down mode nRF24L01+ is disabled using minimal current consumption. All register values available are maintained and the SPI is kept active, enabling change of configuration and the uploading/downloading of data registers. For start up times see Table 16. on page 24. Power down mode is entered by setting the PWR_UP bit in the CONFIG register low.

6.1.3 Standby Modes

6.1.3.1 Standby-I mode

By setting the PWR_UP bit in the CONFIG register to 1, the device enters standby-I mode. Standby-I mode is used to minimize average current consumption while maintaining short start up times. In this mode only part of the crystal oscillator is active. Change to active modes only happens if CE is set high and when CE is set low, the nRF24L01 returns to standby-I mode from both the TX and RX modes.

6.1.3.2 Standby-II mode

In standby-II mode extra clock buffers are active and more current is used compared to standby-I mode. nRF24L01+ enters standby-II mode if CE is held high on a PTX device with an empty TX FIFO. If a new packet is uploaded to the TX FIFO, the PLL immediately starts and the packet is transmitted after the normal PLL settling delay (130µs).

Register values are maintained and the SPI can be activated during both standby modes. For start up times see Table 16. on page 24.

6.1.4 RX mode

The RX mode is an active mode where the nRF24L01+ radio is used as a receiver. To enter this mode, the nRF24L01+ must have the PWR_UP bit, PRIM_RX bit and the CE pin set high.

In RX mode the receiver demodulates the signals from the RF channel, constantly presenting the demodulated data to the baseband protocol engine. The baseband protocol engine constantly searches for a valid packet. If a valid packet is found (by a matching address and a valid CRC) the payload of the packet is presented in a vacant slot in the RX FIFOs. If the RX FIFOs are full, the received packet is discarded.

The nRF24L01+ remains in RX mode until the MCU configures it to standby-I mode or power down mode. However, if the automatic protocol features (Enhanced ShockBurst™) in the baseband protocol engine are enabled, the nRF24L01+ can enter other modes in order to execute the protocol.

In RX mode a Received Power Detector (RPD) signal is available. The RPD is a signal that is set high when a RF signal higher than -64 dBm is detected inside the receiving frequency channel. The internal RPD signal is filtered before presented to the RPD register. The RF signal must be present for at least 40µs before the RPD is set high. How to use the RPD is described in Section 6.4 on page 25.

6.1.5 TX mode

The TX mode is an active mode for transmitting packets. To enter this mode, the nRF24L01+ must have the PWR_UP bit set high, PRIM_RX bit set low, a payload in the TX FIFO and a high pulse on the CE for more than 10µs.

The nRF24L01+ stays in TX mode until it finishes transmitting a packet. If CE = 0, nRF24L01+ returns to standby-I mode. If CE = 1, the status of the TX FIFO determines the next action. If the TX FIFO is not empty the nRF24L01+ remains in TX mode and transmits the next packet. If the TX FIFO is empty the nRF24L01+ goes into standby-II mode. The nRF24L01+ transmitter PLL operates in open loop when in TX mode. It is important never to keep the nRF24L01+ in TX mode for more than 4ms at a time. If the Enhanced ShockBurst™ features are enabled, nRF24L01+ is never in TX mode longer than 4ms.

6.1.6 Operational modes configuration

The following table (Table 15.) describes how to configure the operational modes.

Mode	PWR_UP register	PRIM_RX register	CE input pin	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFOs. Will empty all levels in TX FIFOs[a].
TX mode	1	0	Minimum 10µs high pulse	Data in TX FIFOs.Will empty one level in TX FIFOs[b].
Standby-II	1	0	1	TX FIFO empty.
Standby-I	1	-	0	No ongoing packet transmission.
Power Down	0	-	-	-

a. If CE is held high all TX FIFOs are emptied and all necessary ACK and possible retransmits are carried out. The transmission continues as long as the TX FIFO is refilled. If the TX FIFO is empty when the CE is still high, nRF24L01+ enters standby-II mode. In this mode the transmission of a packet is started as soon as the CSN is set high after an upload (UL) of a packet to TX FIFO.

b. This operating mode pulses the CE high for at least 10µs. This allows one packet to be transmitted. This is the normal operating mode. After the packet is transmitted, the nRF24L01+ enters standby-I mode.

Table 15. nRF24L01+ main modes

6.1.7 Timing Information

The timing information in this section relates to the transitions between modes and the timing for the CE pin. The transition from TX mode to RX mode or vice versa is the same as the transition from the standby modes to TX mode or RX mode (max. 130µs), as described in Table 16.

Name	nRF24L01+	Notes	Max.	Min.	Comments
Tpd2stby	Power Down → Standby mode	a	150µs		With external clock
			1.5ms		External crystal, Ls < 30mH
			3ms		External crystal, Ls = 60mH
			4.5ms		External crystal, Ls = 90mH
Tstby2a	Standby modes → TX/RX mode		130µs		
Thce	Minimum CE high			10µs	
Tpece2csn	Delay from CE positive edge to CSN low			4µs	

a. See Table 11. on page 19 for crystal specifications.

Table 16. Operational timing of nRF24L01+

For nRF24L01+ to go from power down mode to TX or RX mode it must first pass through stand-by mode. There must be a delay of Tpd2stby (see Table 16.) after the nRF24L01+ leaves power down mode before the CE is set high.

Note: If VDD is turned off the register value is lost and you must configure nRF24L01+ before entering the TX or RX modes.