



เอกสารประกอบการบรรยาย

ตันติกร โนนศรี

วิทยาการ

สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยราชภัฏเลย

ปรับปรุง 2566

## สารบัญ

หน้า

1. Microcontroller and Arduino .....	1
1.1. ไมโครคอนโทรลเลอร์ (Microcontroller).....	1
1.2. Arduino .....	2
1.3. Layout & Pinout Arduino Board (Model: Arduino UNO R3).....	7
2. Basic Flowchart.....	7
2.1. วิธีเขียนผังงาน (Flowchart) ที่ดี.....	7
2.2. สัญลักษณ์ตามมาตรฐานของสถาบัน ANSI ที่ใช้บ่อย .....	8
2.3. รูปแบบของผังงาน.....	9
3. การใช้งาน Arduino IDE .....	12
4. หลักการเขียนโปรแกรมบน Arduino IDE.....	14
4.1. ส่วนของตัวประมวลผลก่อน (Preprocessor Directives).....	14
4.2. ส่วนของการกำหนดค่า (Global Declarations).....	15
4.3. ส่วนของโครงสร้าง (Structure) .....	15
4.4. ฟังก์ชัน (Functions) และฟังก์ชันที่สร้างขึ้นมาเอง (Users-Defined Function).....	15
4.5. ส่วนของการอธิบายโปรแกรม (Program Comments).....	16
5. พื้นฐานภาษา C สำหรับ Arduino.....	16
5.1. ชนิดข้อมูลและการประกาศตัวแปร .....	16
5.2. ค่าคงที่ (Constants).....	18
5.3. ตัวดำเนินการ (Operation).....	19
5.4. ตัวควบคุมทิศทางในการเขียนโปรแกรม.....	23
6. Basic Output.....	29

6.1. Digital Output .....	29
6.2. Analog Output (LED) .....	30
6.3. Serial Monitor .....	30
7. Basic Input .....	30
7.1. Digital Input.....	30
7.2. Analog Input .....	31
8. Temperature and Humidity Sensor (DHT11).....	33
9. Control Switch (Relay Switch).....	35
10. LCD Output (16x2) .....	36
11. Distance Sensor (Ultrasonic).....	39
12. Alarm Control (Buzzer) .....	40
เอกสารอ้างอิง .....	41

## สารบัญตาราง

หน้า

ตารางที่ 1 แสดงการเปรียบเทียบคุณสมบัติบอร์ด Arduino ในแต่ละรุ่น.....	6
ตารางที่ 2 สัญลักษณ์ตามมาตรฐานของสถาบัน ANSI ที่ใช้บ่อย.....	8
ตารางที่ 3 รูปแบบของผังงาน แบบเรียงลำดับ.....	9
ตารางที่ 4 รูปแบบของผังงาน แบบมีเงื่อนไข.....	10
ตารางที่ 5 รูปแบบของผังงาน แบบทำซ้ำ.....	11
ตารางที่ 6 แสดงชนิดของข้อมูล.....	16
ตารางที่ 7 ตัวอย่างตารางข้อมูลอาเรย์.....	18
ตารางที่ 8 ค่าคงที่เริ่มต้น.....	19
ตารางที่ 9 เครื่องหมายการคำนวณทางคณิตศาสตร์.....	19
ตารางที่ 10 เครื่องหมายการคำนวณทางคณิตศาสตร์ประเภทลดรูป.....	20
ตารางที่ 11 เครื่องหมายการดำเนินการเปรียบเทียบ.....	21
ตารางที่ 12 เครื่องหมายทางตรรกศาสตร์.....	22
ตารางที่ 13 นิพจน์ทางคณิตศาสตร์.....	22
ตารางที่ 14 นิพจน์ทางตรรกศาสตร์.....	22
ตารางที่ 15 ลำดับความสำคัญเครื่องหมาย.....	23

## สารบัญภาพประกอบ

หน้า

ภาพประกอบที่ 1 บอร์ด Arduino Uno R3 .....	3
ภาพประกอบที่ 2 บอร์ด Arduino Uno SMD.....	3
ภาพประกอบที่ 3 บอร์ด Arduino Mega 2560 R3 .....	3
ภาพประกอบที่ 4 บอร์ด Arduino UNO WiFi Rev2 .....	4
ภาพประกอบที่ 5 บอร์ด Arduino Leonardo .....	4
ภาพประกอบที่ 6 บอร์ด Arduino Leonardo .....	4
ภาพประกอบที่ 7 บอร์ด Arduino Pro Mini 328 3.3V .....	5
ภาพประกอบที่ 8 บอร์ด Arduino Pro Mini 328 5V .....	5
ภาพประกอบที่ 9 บอร์ด Arduino Ethernet with PoE module Arduino Leonardo ETH with PoE ....	5
ภาพประกอบที่ 10 บอร์ด Arduino Due .....	6
ภาพประกอบที่ 11 แสดง 2.3. Layout & Pin out Arduino Board ของ Arduino UNO R3.....	7
ภาพประกอบที่ 12 รูปแบบการเชื่อมต่อระหว่างคอมพิวเตอร์กับ Arduino.....	12
ภาพประกอบที่ 13 เลือกบอร์ด Arduino ที่ต้องการ upload .....	12
ภาพประกอบที่ 14 เลือกหมายเลข Comport ของบอร์ด.....	13
ภาพประกอบที่ 15 กด Verify เพื่อตรวจสอบความถูกต้อง .....	13
ภาพประกอบที่ 16 กด Upload เพื่อส่งโปรแกรมไปยัง Arduino และแสดงผลของการ Upload .....	14
ภาพประกอบที่ 17 LED Pinout.....	29
ภาพประกอบที่ 18 รูปแบบการต่อวงจร LED .....	29
ภาพประกอบที่ 19 Button Pinout .....	30
ภาพประกอบที่ 20 รูปแบบการต่อวงจร Button Pulldown .....	30
ภาพประกอบที่ 21 รูปแบบการต่อวงจร Button Pullup.....	31
ภาพประกอบที่ 22 การต่อวงจร Button โดยไม่ต้องต่อ R (Pullup) .....	31
ภาพประกอบที่ 23 Variable Resistor Pinout .....	32
ภาพประกอบที่ 24 การต่อวงจร Variable Resistor .....	32
ภาพประกอบที่ 25 LDR Pinout .....	32
ภาพประกอบที่ 26 การต่อวงจร LDR.....	33

ภาพประกอบที่ 27 DHT11 Pinout .....	33
ภาพประกอบที่ 28 การต่อวงจร DHT11 .....	34
ภาพประกอบที่ 29 Relay Pinout .....	35
ภาพประกอบที่ 30 รูปแบบของ Relay (ก) Normally Open Mode (Active High) (ข) Normally Closed Mode (Active Low) .....	35
ภาพประกอบที่ 31 การต่อวงจร Relay .....	35
ภาพประกอบที่ 32 LCD 16x2 Pinout .....	36
ภาพประกอบที่ 33 การต่อวงจร LCD 16x2 .....	36
ภาพประกอบที่ 34 LCD 16x2 (I2C) Pinout .....	37
ภาพประกอบที่ 35 I2C Address .....	37
ภาพประกอบที่ 36 การต่อวงจร LCD 16x2 (I2C) .....	38
ภาพประกอบที่ 37 Ultrasonic Sensor Pinout.....	39
ภาพประกอบที่ 38 การต่อวงจร Ultrasonic Sensor .....	39
ภาพประกอบที่ 39 Buzzer Pinout .....	40
ภาพประกอบที่ 40 การต่อวงจร Buzzer .....	40

## สารบัญตัวอย่าง

หน้า

ตัวอย่างที่ 1 เปิด LED (Digital) .....	29
ตัวอย่างที่ 2 ไฟกระพริบ .....	29
ตัวอย่างที่ 3 เปิด LED (Analog).....	30
ตัวอย่างที่ 4 Serial Monitor .....	30
ตัวอย่างที่ 5 การอ่านค่าจาก Button.....	31
ตัวอย่างที่ 6 การอ่านค่าจาก Button โดยไม่ต้องต่อ R (Pullup).....	31
ตัวอย่างที่ 7 การอ่านค่าจาก Variable Resistor .....	32
ตัวอย่างที่ 8 การอ่านค่าจาก LDR.....	33
ตัวอย่างที่ 9 การอ่านค่าจาก DHT11 .....	34
ตัวอย่างที่ 10 การสั่งงาน Relay .....	35
ตัวอย่างที่ 11 การแสดงผลทาง LCD.....	36
ตัวอย่างที่ 12 การแสดงผลทาง LCD (I2C) .....	38
ตัวอย่างที่ 13 การอ่านค่าจาก Ultrasonic Sensor .....	39
ตัวอย่างที่ 14 การสั่งงาน Buzzer .....	40

## 1. Microcontroller and Arduino

### 1.1. ไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์ คือส่วนสำคัญที่ทำหน้าที่สำหรับการประมวลผล และการควบคุมให้ทำงานตามที่ต้องการ ซึ่งการเลือกใช้ไมโครคอนโทรลเลอร์ก็ควรจะต้องเหมาะสมกับงานประยุกต์ที่จะนำไปใช้ สาเหตุที่นิยมใช้ไมโครคอนโทรลเลอร์มากกว่าการใช้งานไมโครโพรเซสเซอร์ (Microprocessor) ก็เนื่องมาจากว่าไมโครคอนโทรลเลอร์นั้นมีส่วนประกอบอื่นที่จำเป็นต่อการทำงานที่ไม่ซับซ้อนมากนัก มีส่วนของมาตรฐานการเชื่อมต่อที่หลากหลาย เช่น ADC UART I2C เป็นต้น ในส่วนของหน่วยความจำภายในของไมโครคอนโทรลเลอร์มีทั้งแบบ ROM (Read Only Memory) RAM (Random Access Memory) หรือ EEPROM (Erasable Programmable Read Only Memory) และอาจจะมีส่วนจัดการพลังงาน (Power Management Unit) อีกด้วย (วรรณรัช สันติอมรทัต และ สุกุณา เจริญปัญญาศักดิ์, 2559) ดังนั้นการออกแบบในส่วนของการทำงานหากต้องการให้สามารถประมวลผลงานที่ซับซ้อนทางด้านการประมวลผลภาพ (Image Processing) หรือการประมวลผลทางปัญญาประดิษฐ์ จึงควรเลือกใช้ไมโครโพรเซสเซอร์ที่มีความสามารถประมวลผลที่มีความซับซ้อนมากกว่า และจะต้องพิจารณาการใช้พลังงานที่เพิ่มขึ้นอีกด้วยเช่นกัน

ตระกูลของไมโครคอนโทรลเลอร์ มีหลากหลายรุ่น หลากหลายยี่ห้อ หลากหลายผู้ผลิต แต่ละรุ่นอาจจะเลือกใช้ CPU ที่แตกต่างกันไปตามคุณสมบัติ หรือตามต้นทุนการผลิต หรือตามเทคโนโลยีที่ค้นพบ และตามแบบฉบับสถาปัตยกรรมที่มีการจดสิทธิบัตร ได้แก่

- 1.1.1. ไมโครคอนโทรลเลอร์ ตระกูล PIC บริษัทผู้ผลิต Microchip
- 1.1.2. ไมโครคอนโทรลเลอร์ ตระกูล ACS51 บริษัทผู้ผลิต Atmel, Phillips
- 1.1.3. ไมโครคอนโทรลเลอร์ ตระกูล AVR บริษัทผู้ผลิต Atmel
- 1.1.4. ไมโครคอนโทรลเลอร์ ตระกูล ARM บริษัทผู้ผลิต Atmel, Phillips, Analog Device, Samsung, STMicroelectronics
- 1.1.5. ไมโครคอนโทรลเลอร์ ตระกูล Basic Stamp บริษัทผู้ผลิต Parallax
- 1.1.6. ไมโครคอนโทรลเลอร์ ตระกูล PCOS บริษัทผู้ผลิต Cypress
- 1.1.7. ไมโครคอนโทรลเลอร์ ตระกูล MSC บริษัทผู้ผลิต Texas Instruments
- 1.1.8. ไมโครคอนโทรลเลอร์ ตระกูล 68HC บริษัทผู้ผลิต Motorola
- 1.1.9. ไมโครคอนโทรลเลอร์ ตระกูล H8 บริษัทผู้ผลิต Renesas
- 1.1.10. ไมโครคอนโทรลเลอร์ ตระกูล Rabbit บริษัทผู้ผลิต Rabbit Semiconductor
- 1.1.11. ไมโครคอนโทรลเลอร์ ตระกูล Z80 บริษัทผู้ผลิต Zilog



## 1.2. Arduino

Arduino เป็นภาษาอิตาลี อ่านว่า “อาดูอีโน” จนเพี้ยนมาเป็น “อาร์ดูโน” ซึ่ง Arduino คือ ไมโครคอนโทรลเลอร์ตระกูล AVR แบบ Open Source ทั้ง Hardware และ Software สำหรับต้นแบบโปรแกรมการทำงานของวงจร โดยเปิดกว้างให้คนที่สนใจสามารถพัฒนาและประยุกต์ใช้งานได้อย่างเสรี ไม่มีลิขสิทธิ์ จึงทำให้บอร์ด Arduino เป็นที่รู้จักและนิยมนำมาใช้งานกันอย่างแพร่หลาย และข้อดีอีกอย่างคือเป็นบอร์ดขนาดเล็กสามารถพกพาสะดวก ส่วนประกอบหลักของ Arduino คือไมโครคอนโทรลเลอร์ นำมาประกอบรวมกันกับอุปกรณ์อิเล็กทรอนิกส์อื่น ๆ และถูกพัฒนาได้ง่าย ทั้งนี้บอร์ด Arduino มีหลายรุ่นให้เลือกใช้

ความง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่าง ๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิกส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด หรือเพื่อความสะดวกสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่าง ๆ เช่น Arduino XBee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino GPRS Shield เป็นต้น มาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้เลย (บทความ Arduino คืออะไร ตอนที่ 1 แนะนำเพื่อนใหม่ที่ชื่อ Arduino, 2017)

จุดเด่นที่ทำให้บอร์ด Arduino เป็นที่นิยม เนื่องจากง่ายต่อการพัฒนา มีรูปแบบคำสั่งพื้นฐานไม่ซับซ้อนเหมาะสำหรับผู้เริ่มต้น มี Arduino Community เป็นกลุ่มคนที่ร่วมกันพัฒนาที่เข้มแข็ง และการเป็น Open Source ทำให้ผู้ใช้งานสามารถนำบอร์ดไปต่อยอดใช้งานได้หลายด้าน ราคาไม่แพง และสามารถ Cross Platform สามารถพัฒนาโปรแกรมบนระบบปฏิบัติการใดก็ได้กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม จากนั้นกดปุ่ม Upload โค้ด โปรแกรมไปยังบอร์ด Arduino ผ่านทางสาย USB เมื่ออัปโหลดเรียบร้อยแล้ว จะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที

### 1.2.1. Arduino รุ่นต่าง ๆ

1.2.1.1. Arduino Uno R3 เป็นบอร์ด Arduino ที่ได้รับความนิยมมากที่สุด เนื่องจากราคาไม่แพง ส่วนใหญ่โปรเจกต์และ Library ต่าง ๆ ที่พัฒนาขึ้นมา Support จะอ้างอิงกับบอร์ดนี้เป็นหลัก และข้อดีอีกอย่างคือ กรณีที่ MCU เสีย ผู้ใช้งานสามารถซื้อมาเปลี่ยนเองได้ง่าย



ภาพประกอบที่ 1 บอร์ด Arduino Uno R3

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

1.2.1.2. Arduino Uno SMD เป็นบอร์ดที่มีคุณสมบัติและการทำงานเหมือนกับบอร์ด Arduino UNO R3 ทุกประการ แต่จะแตกต่างกับที่ Package ของ MCU ซึ่งบอร์ดนี้จะมี MCU ที่เป็น Package SMD (Arduino UNO R3 มี MCU ที่เป็น Package DIP)



ภาพประกอบที่ 2 บอร์ด Arduino Uno SMD

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

1.2.1.3. Arduino Mega 2560 R3 เป็นบอร์ด Arduino ที่ออกแบบมาสำหรับงานที่ต้องใช้ I/O มากกว่า Arduino Uno R3 เช่น งานที่ต้องการรับสัญญาณจาก Sensor หรือควบคุมมอเตอร์ Servo หลายๆ ตัว ทำให้ Pin I/O ของบอร์ด Arduino Uno R3 ไม่สามารถรองรับได้ ทั้งนี้บอร์ด Mega 2560 R3 ยังมีความหน่วยความจำแบบ Flash มากกว่า Arduino Uno R3 ทำให้สามารถเขียนโค้ดโปรแกรมเข้าไปได้มากกว่า ในความเร็วของ MCU ที่เท่ากัน



ภาพประกอบที่ 3 บอร์ด Arduino Mega 2560 R3

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

1.2.1.4. Arduino UNO WiFi Rev2 ใช้ชิพ ATmega4809 ซึ่งมี Flash memory 48KB, SRAM 6KB และยังประหยัดพลังงานมากกว่าตัวเวอร์ชันเก่า อีกทั้งตัวบอร์ด Arduino Uno Wifi Rev2 ยังมาพร้อม

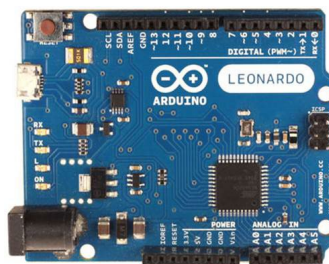
กับชิพ ESP32 u-Blox NiNA-W13 Wifi ที่ทำให้ตัวบอร์ดรองรับใช้งานร่วมกับ Wifi และมีฟังก์ชันการใช้งานที่หลากหลายมากขึ้น



ภาพประกอบที่ 4 บอร์ด Arduino UNO WiFi Rev2

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

1.2.1.5. Arduino Leonardo การทำงานจะคล้ายกับบอร์ด Arduino Uno R3 แต่มีการเปลี่ยน MCU ตัวใหม่เป็น ATmega32U4 ซึ่งมีโมดูลพอร์ต USB มาด้วยบนชิป (แตกต่างจากบอร์ด Arduino UNO R3 หรือ Arduino Mega 2560 ที่ต้องใช้ชิป ATmega16U2 ร่วมกับ Atmega328 ในการเชื่อมต่อกับพอร์ต USB)



ภาพประกอบที่ 5 บอร์ด Arduino Leonardo

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

1.2.1.6. Arduino MKR WIFI 1010 เป็นบอร์ดในตระกูล MKR ของ Arduino ปรับปรุงจากบอร์ด MKR 1000 WIFI ประกอบด้วย 3 ส่วน ได้แก่ (1) ชิพหลัก SAMD21 ARM Cortex-M0+ 32-bit Low Power MCU 48 MHz SRAM 32 KB และ Flash Memory 256 KB มี Digital I/O 8 ขา Analog Input 7 ขา Analog Output 1 ขา UART 1 ชุด SPI 1 ชุด I2C 1 ชุด รองรับ External Interrupt 8 ขา รองรับ PWM 12 ขา แรงดันขา I/O ทำงานที่ 3.3 โวลต์เท่านั้น (2) ชิพ ESP32 U-BLOX NINA-W10 Series Low Power 2.4 GHz 802.11 bgn (3) ชิพ ECC508 Crypto Authentication



ภาพประกอบที่ 6 บอร์ด Arduino Leonardo

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

1.2.1.7. Arduino Pro Mini 328 3.3V เป็นบอร์ด Arduino ขนาดเล็ก ที่ใช้ MCU เบอร์ ATmega328 ซึ่งจะคล้ายกับบอร์ด Arduino Mini05 แต่บนบอร์ดจะมี Regulator 3.3 V ชุดเดียวเท่านั้น ระดับแรงดันไฟฟ้า I/O คือ 3.3V



ภาพประกอบที่ 7 บอร์ด Arduino Pro Mini 328 3.3V

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

1.2.1.8. Arduino Pro Mini 328 5V เป็นบอร์ด Arduino ขนาดเล็ก ที่ใช้ MCU เบอร์ ATmega328 เช่นเดียวกับบอร์ด Arduino Mini 05 แต่บนบอร์ดจะมี Regulator 5V ชุดเดียวเท่านั้น ระดับแรงดันไฟฟ้า I/O คือ 5V



ภาพประกอบที่ 8 บอร์ด Arduino Pro Mini 328 5V

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

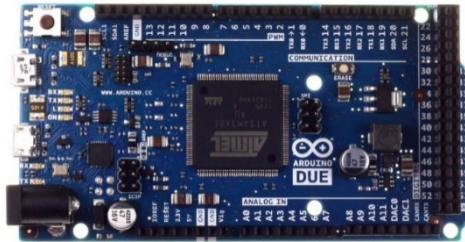
1.2.1.9. Arduino Ethernet with PoE module เป็นบอร์ดจาก Arduino.org ที่มี ATmega32U4 พร้อมกับโมดูล Ethernet ในตัว เหมาะสำหรับผู้ที่ต้องการใช้งานควบคุมผ่าน Network (TCP/IP)



ภาพประกอบที่ 9 บอร์ด Arduino Ethernet with PoE module Arduino Leonardo ETH with PoE

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

1.2.1.10. Arduino Due เป็นบอร์ด Arduino ที่เปลี่ยนชิป MCU ใหม่ ซึ่งจากเดิมเป็นตระกูล AVR เปลี่ยนเป็นเบอร์ AT91SAM3X8E (ตระกูลARM Cortex-M3) แทน ทำให้การประมวลผลเร็วขึ้น แต่ยังคงรูปแบบโค้ดโปรแกรมของ Arduino ที่ง่ายอยู่ ข้อควรระวัง: เนื่องจาก MCU เป็นคนละเบอร์กับ Arduino Uno R3 อาจจะทำให้บอร์ด Shield บางตัวหรือ Library ใช้ร่วมกันกับบอร์ด Arduino Leonardo ไม่ได้ ผู้ใช้งานจำเป็นต้องตรวจสอบก่อนใช้งาน



ภาพประกอบที่ 10 บอร์ด Arduino Due

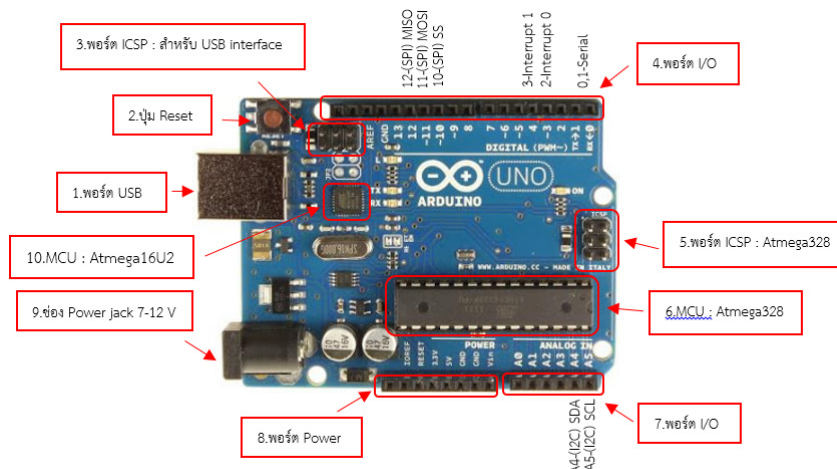
ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

ตารางที่ 1 แสดงการเปรียบเทียบคุณสมบัติบอร์ด Arduino ในแต่ละรุ่น

	Processor					Input / Output							Power				Connectivity				
	Family	SRAM	FLASH	EEPROM	Clock	Digital I/O	Analog In	ADC Bits	PWM	UART	Analog Out	DAC Bits	VCC	Vin Range	5V	3V3	USB-Serial	I2C	Ethernet	USB-Host	SD Card
Arduino UNO R3	ATmega328	2k	32k	1k	16MHz	14	6	10	6	1	N/A	N/A	5V	7-12V	Yes	Yes	ATmega16U2	1	No	No	No
Arduino UNO SMD	ATmega328	2k	32k	1k	16MHz	14	6	10	6	1	N/A	N/A	5V	7-12V	Yes	Yes	ATmega16U2	1	No	No	No
Arduino Mega 2560 R3	ATmega2560	8k	256k	4k	16MHz	54	16	10	14	4	N/A	N/A	5V	7-18V	Yes	Yes	ATmega16U2	1	No	No	No
Arduino Mega ADK	ATmega2560	8k	256k	4k	16MHz	54	16	10	14	4	N/A	N/A	5V	7-18V	Yes	Yes	ATmega16U2	1	MAX3421E	No	No
Arduino Leonardo	ATmega32U4	2.5k	32k	1k	16MHz	25	12	10	7	1	N/A	N/A	5V	7-12V	Yes	Yes	Built-in	1	No	No	No
Arduino Mini 05	ATmega328	2k	32k	1k	16MHz	14	6	10	6	1	N/A	N/A	5V	7V-9V	Yes	No	N/A	1	No	No	No
Arduino Pro Mini 328 - 3.3V	ATmega328	2k	32k	1k	8MHz	14	6	10	6	1	N/A	N/A	3.3V	5V-12V	No	Yes	N/A	1	No	No	No
Arduino Pro Mini 328 - 5V	ATmega328	2k	32k	1k	16MHz	14	6	10	6	1	N/A	N/A	5V	7V-12V	Yes	No	N/A	1	No	No	No
Arduino Ethernet with PoE module	ATmega328	2k	32k	1k	16MHz	9	6	10	4	1	N/A	N/A	5V	6-18V	Yes	Yes	N/A	1	No	No	No
Arduino Ethernet without PoE module	ATmega328	2k	32k	1k	16MHz	9	6	10	4	1	N/A	N/A	5V	6-18V	Yes	Yes	N/A	1	No	No	No
Arduino DUE	SAM3X8E	96kb	512k	N/A	84MHz	70	12	12	12	4	2	12	3.3V	7-12V	No	VC C	Built-in	2	No	Yes	No

ที่มา : (บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน, 2016)

### 1.3. Layout & Pinout Arduino Board (Model: Arduino UNO R3)



ภาพประกอบที่ 11 แสดง 2.3. Layout & Pin out Arduino Board ของ Arduino UNO R3

ที่มา : (บทความ Arduino คืออะไร ตอนที่1 แนะนำเพื่อนใหม่ที่ชื่อ Arduino, 2017)

1.3.1. USB Port : ใช้สำหรับต่อกับ Computer เพื่ออัปโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด

1.3.2. Reset Button : เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่

1.3.3. ICSP Port ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com Port บน Atmega16U2

1.3.4. I/O Port : Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin 0, 1 เป็นขา Tx,Rx Serial และ Pin 3, 5, 6, 9, 10, 11 เป็นขา PWM (Pulse Width Modulation)

1.3.5. ICSP Port : Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader

1.3.6. MCU : Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino

1.3.7. I/O Port : Analog I/O ตั้งแต่ขา A0-A5

1.3.8. Power Port: ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, Vin

1.3.9. Power Jack: รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V

1.3.10. MCU ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U2

## 2. Basic Flowchart

### 2.1. วิธีเขียนผังงาน (Flowchart) ที่ดี

วิธีเขียนผังงานที่ดีนั้นมีข้อควรปฏิบัติดังนี้ (สุรศักดิ์ วีรรวงศ์, ม.ป.พ.)

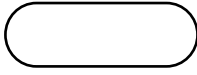
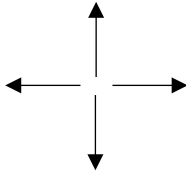

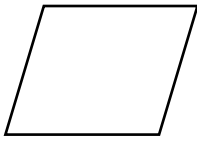

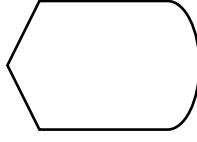
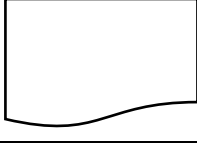
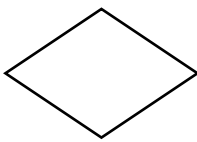
2.1.1. ใช้สัญลักษณ์ตามมาตรฐานของสถาบัน ANSI

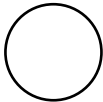
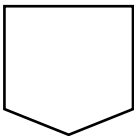
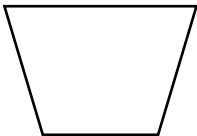


- 2.1.2. ข้อความที่ใช้ในสัญลักษณ์ควรจะเป็นข้อความสั้นๆ ที่อ่านเข้าใจและชัดเจน
- 2.1.3. ขนาดของสัญลักษณ์ไม่ควรเล็กหรือใหญ่เกินไป
- 2.1.4. เขียนขั้นตอนจากบนลงล่าง / จากซ้ายไปขวา โดยเชื่อมขั้นตอนด้วยลูกศรกับทิศทาง
- 2.1.5. เขียนผังงานให้จบภายในหน้าเดียวกัน

## 2.2. สัญลักษณ์ตามมาตรฐานของสถาบัน ANSI ที่ใช้บ่อย

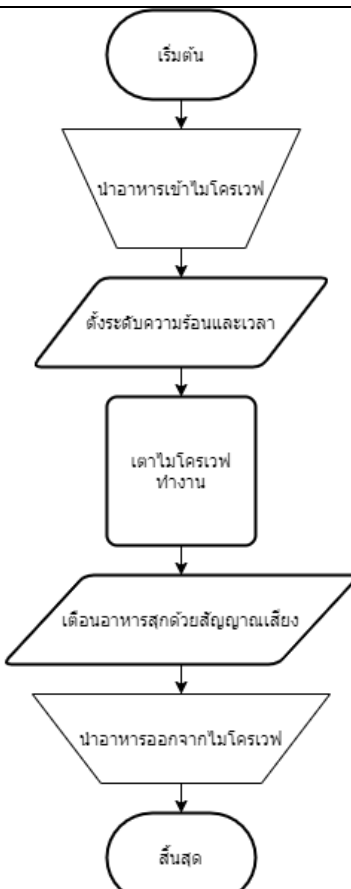
ตารางที่ 2 สัญลักษณ์ตามมาตรฐานของสถาบัน ANSI ที่ใช้บ่อย

สัญลักษณ์	ชื่อที่เรียก	ความหมาย
	Terminator	จุดเริ่มต้น และจุดสิ้นสุดของการทำงาน
	Flow line/Direction	เส้นแสดงทิศทางการทำงาน ต้องมีหัวลูกศรเดียวเท่านั้น
	Process	การปฏิบัติงาน / ประมวลผล หรือกำหนดค่าข้อมูลให้กับตัวแปร
	Input/Output	รับ/แสดงผลข้อมูล ในกรณีที่ไม่ระบุอุปกรณ์
	Keyboard	รับ/อ่านข้อมูลที่รับเข้ามาจากคีย์บอร์ด
	Monitor/Display	แสดงรายละเอียดข้อมูล หรือผลลัพธ์ทางจอภาพ
	Printer	แสดงรายละเอียดข้อมูล หรือผลลัพธ์ทางเครื่องพิมพ์
	Decision	การเปรียบเทียบเพื่อให้ตัดสินใจเลือก โดยจะมีเส้นออกจากสัญลักษณ์นี้เพื่อชี้ทิศทางไปยังการทำงานตามเงื่อนไขที่เป็นจริง และเส้นที่ชี้ไปยังการทำงานตามเงื่อนไขที่เป็นเท็จ

สัญลักษณ์	ชื่อที่เรียก	ความหมาย
	In -Paper Connector	จุดเชื่อมต่อ ภายในหน้าเดียวกัน
	Between-page connector	จุดเชื่อมต่อไปยังหน้าอื่น
	Manual Operator	กระบวนการที่ทำโดยคน

### 2.3. รูปแบบของผังงาน

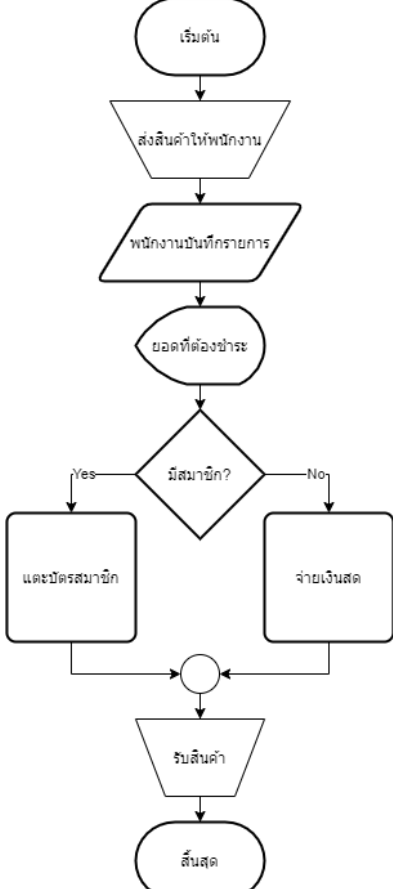
2.3.1. แบบเรียงลำดับ (Sequence) เป็นรูปแบบผังงานที่ง่ายที่สุด ไม่ซับซ้อนและไม่มีการเปรียบเทียบเงื่อนไขใดๆ โดยแสดงขั้นตอนการทำงานไปตามลำดับตั้งแต่ต้นจนสิ้นสุดกระบวนการ ดังตัวอย่าง การอุ่นอาหาร ตารางที่ 3 รูปแบบของผังงาน แบบเรียงลำดับ

ขั้นตอนการทำงาน (Algorithm)	ผังงาน
<ul style="list-style-type: none"> <li>● นำอาหารเข้าเตาไมโครเวฟ</li> <li>● ตั้งระดับความร้อนและระยะเวลาที่ต้องการอุ่น</li> <li>● เตาไมโครเวฟเริ่มกระบวนการอุ่นอาหาร</li> <li>● เตาไมโครเวฟส่งสัญญาณเสียงเตือนอาหารสุก</li> <li>● นำอาหารที่อุ่นสุกแล้วออกจากเตาไมโครเวฟ</li> </ul>	 <pre> graph TD     Start([เริ่มต้น]) --&gt; A[นำอาหารเข้าไมโครเวฟ]     A --&gt; B[/ตั้งระดับความร้อนและเวลา/]     B --&gt; C[เตาไมโครเวฟทำงาน]     C --&gt; D[/เตือนอาหารสุกด้วยสัญญาณเสียง/]     D --&gt; E[นำอาหารออกจากไมโครเวฟ]     E --&gt; End([สิ้นสุด]) </pre>



2.3.2. แบบมีเงื่อนไข (Decision หรือ Selection) เป็นรูปแบบของผังงานที่มีเงื่อนไขให้เลือกตัดสินใจ โดยเตรียมขั้นตอนการทำงานไว้รองรับสำหรับเงื่อนไขนั้นๆ ดังตัวอย่าง การชำระค่าสินค้าที่ 7-11

ตารางที่ 4 รูปแบบของผังงาน แบบมีเงื่อนไข

ขั้นตอนการทำงาน (Algorithm)	ผังงาน
<ul style="list-style-type: none"> <li>● นำสินค้าให้พนักงานคิดเงิน</li> <li>● แสดงจำนวนเงินที่ต้องชำระบนหน้าจอของเครื่องคิดเงิน</li> <li>● ชำระเงิน</li> </ul>	 <pre> graph TD     Start([เริ่มต้น]) --&gt; Send[/ส่งสินค้าให้พนักงาน/]     Send --&gt; Record[/พนักงานบันทึกรายการ/]     Record --&gt; Amount([ยอดที่ต้องชำระ])     Amount --&gt; Decision{มีสมาชิก?}     Decision -- Yes --&gt; TapCard[แตะบัตรสมาชิก]     Decision -- No --&gt; PayCash[จ่ายเงินสด]     TapCard --&gt; Merge(( ))     PayCash --&gt; Merge     Merge --&gt; Receive[/รับสินค้า/]     Receive --&gt; End([สิ้นสุด]) </pre>

2.3.3. แบบทำซ้ำ (Repeat หรือ Loop) เป็นรูปแบบผังงานที่มีขั้นตอนการทำงานซ้ำ ๆ โดยมีเงื่อนไขเป็นตัวควบคุมเช่นเมื่อตรวจสอบแล้วพบว่าเงื่อนไขเป็นจริง จึงทำงานขั้นตอนนั้นซ้ำ ๆ ซึ่งจะหาภายใต้เงื่อนไขที่เป็นจริงเท่านั้น ดังตัวอย่าง การซื้อบัตรโดยสารรถไฟฟ้า BTS ผ่านเครื่องจำหน่ายบัตร

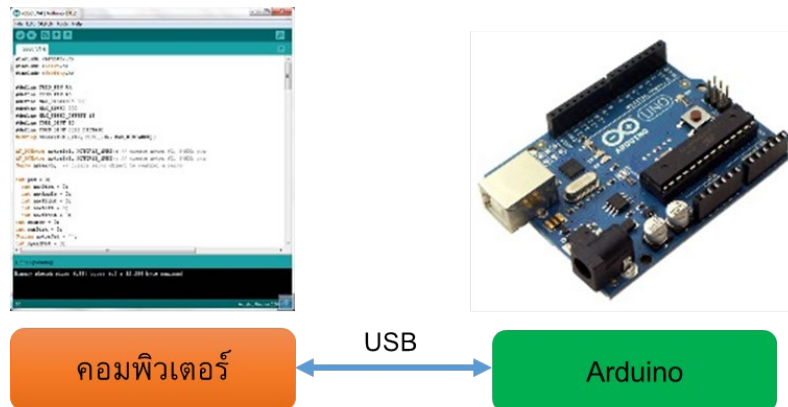
ตารางที่ 5 รูปแบบของผังงาน แบบทำซ้ำ

ขั้นตอนการทำงาน (Algorithm)	ผังงาน
<ul style="list-style-type: none"> <li>• ดูหมายเลขสถานีที่ต้องการ</li> <li>• กดหมายเลขสถานีปลายทางที่ต้องการ</li> <li>• หน้าจอของเครื่องจำหน่ายจะแสดงจำนวนเงินที่ต้องชำระ</li> <li>• หยอดเหรียญ 5,10 บาทลงในช่องรับเหรียญ ตามจำนวนเงินที่แสดงบนหน้าจอ</li> <li>• ตรวจสอบว่าเหรียญที่หยอดลงไปในนั้นเป็นเหรียญ 5,10 5,10 บาทหรือไม่ ถ้าใช่ ให้ทำข้อ 6 ถ้าไม่ใช่ ให้กลับไปทำข้อ 4</li> <li>• ตรวจสอบว่าหยอดเหรียญครบตามจำนวนเงินที่ต้องชำระหรือไม่ถ้าครบให้ทำข้อ 7 ถ้าไม่ครบ ให้กลับไปทำข้อ 4</li> <li>• รับบัตรโดยสารรถไฟฟ้า BTS จากช่องรับบัตรโดยสาร</li> </ul>	<pre> graph TD     Start([เริ่มต้น]) --&gt; Look[ดูหมายเลขสถานี]     Look --&gt; Press[/กดหมายเลขสถานี/]     Press --&gt; Display([แสดงยอดเงินที่ต้องชำระ])     Display --&gt; Connector(( ))     Connector --&gt; Insert[/หยอดเหรียญ/]     Insert --&gt; Receive[/รับเหรียญ/]     Receive --&gt; Coin{เหรียญ 5,10 ?}     Coin -- Yes --&gt; Paid{ครบตามยอดเงิน?}     Coin -- NO --&gt; Connector     Paid -- Yes --&gt; Ticket[บัตรโดยสาร]     Paid -- NO --&gt; Connector     Ticket --&gt; ReceiveTicket[/รับบัตรโดยสาร/]     ReceiveTicket --&gt; End([สิ้นสุด])   </pre>

### 3. การใช้งาน Arduino IDE

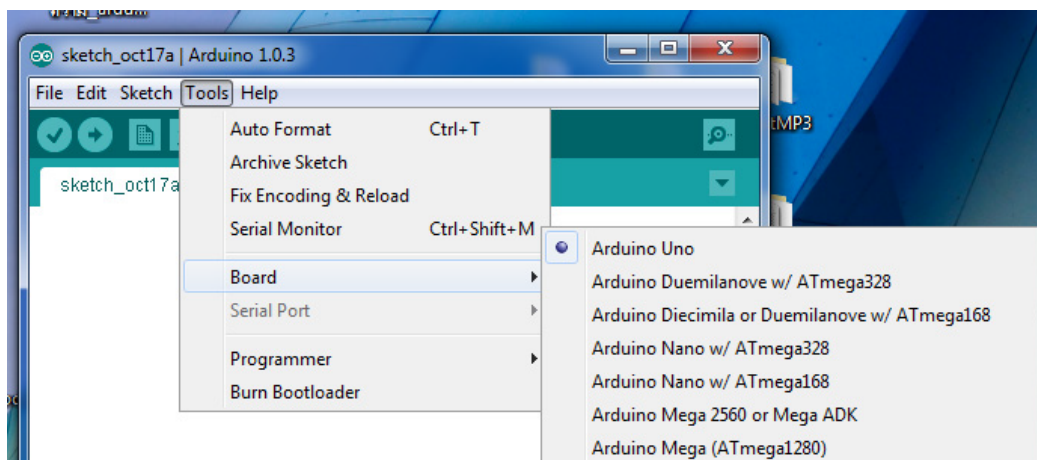
3.1. เขียนโปรแกรมบนคอมพิวเตอร์ ผ่านทางโปรแกรม Arduino IDE ซึ่งสามารถดาวน์โหลดได้จาก [Arduino.cc/en/main/software](https://www.arduino.cc/en/main/software)

3.2. หลังจากเขียนโค้ดโปรแกรมเรียบร้อยแล้ว ให้ผู้ใช้งานเลือกบอร์ด Arduino ที่ใช้และหมายเลข Com port



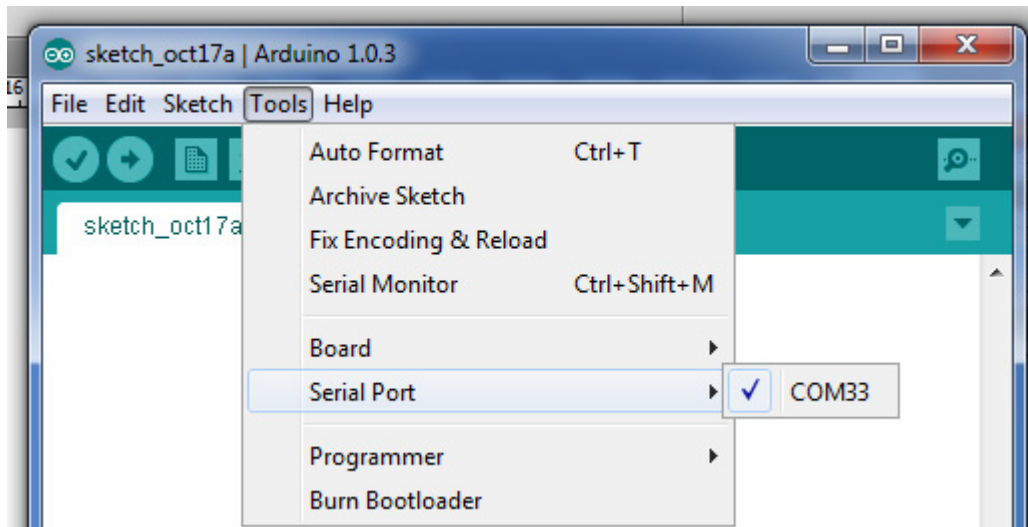
ภาพประกอบที่ 12 รูปแบบการเชื่อมต่อระหว่างคอมพิวเตอร์กับ Arduino

ที่มา : (บทความ Arduino คืออะไร ตอนที่1 แนะนำเพื่อนใหม่ที่ชื่อ Arduino, 2017)



ภาพประกอบที่ 13 เลือกบอร์ด Arduino ที่ต้องการ upload

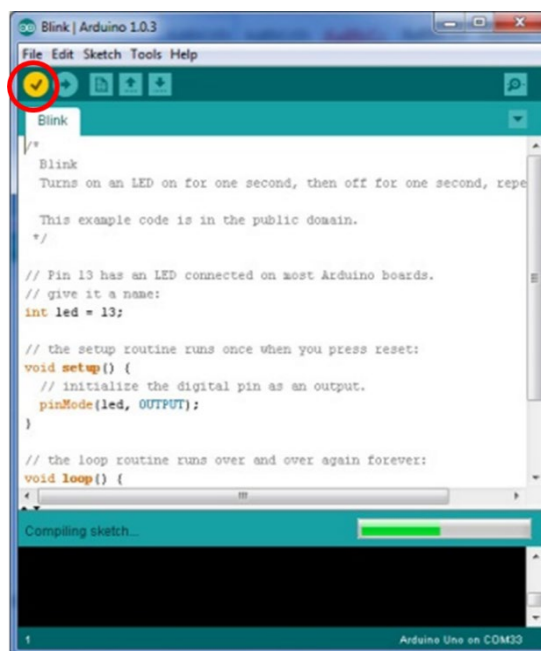
ที่มา : (บทความ Arduino คืออะไร ตอนที่1 แนะนำเพื่อนใหม่ที่ชื่อ Arduino, 2017)



ภาพประกอบที่ 14 เลือกหมายเลข Comport ของบอร์ด

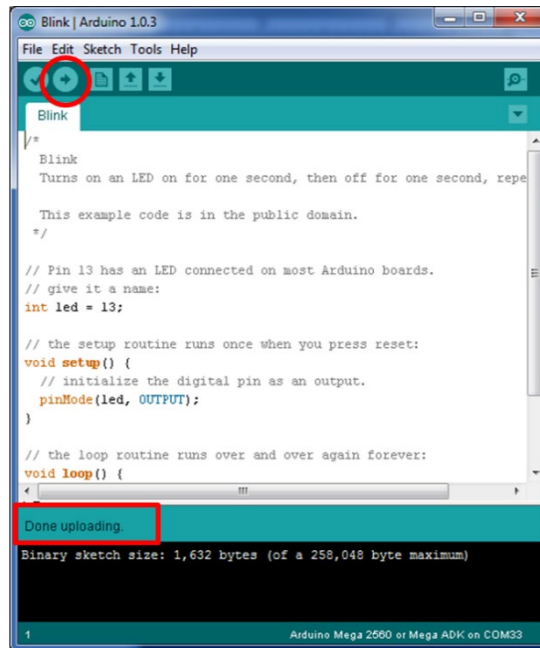
ที่มา : (บทความ Arduino คืออะไร ตอนที่1 แนะนำเพื่อนใหม่ที่ชื่อ Arduino, 2017)

3.3. กดปุ่ม Verify เพื่อตรวจสอบความถูกต้องและ Compile โค้ดโปรแกรม จากนั้นกดปุ่ม Upload โค้ดโปรแกรมไปยังบอร์ด Arduino ผ่านทางสาย USB เมื่ออัปโหลดเรียบร้อยแล้ว จะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที



ภาพประกอบที่ 15 กด Verify เพื่อตรวจสอบความถูกต้อง

ที่มา : (บทความ Arduino คืออะไร ตอนที่1 แนะนำเพื่อนใหม่ที่ชื่อ Arduino, 2017)



ภาพประกอบที่ 16 กด Upload เพื่อส่งโปรแกรมไปยัง Arduino และแสดงผลของการ Upload  
ที่มา : (บทความ Arduino คืออะไร ตอนที่1 แนะนำเพื่อนใหม่ที่ชื่อ Arduino, 2017)

#### 4. หลักการเขียนโปรแกรมบน Arduino IDE

ในการเขียนโปรแกรมบน Arduino IDE จะใช้ภาษา C++ ในการพัฒนา แต่จะไม่เหมือนภาษาซี (C Language) ทั้งหมด มีโครงสร้างของตัวภาษาโดยรวมใกล้เคียงกันกับภาษาซีมาตรฐาน (ANSI-C) จะมีความแตกต่างอยู่บ้าง แต่คำสั่งต่าง ๆ ฟังก์ชัน (Function) ในภาษา C ยังสามารถนำมาใช้ใน Arduino IDE ได้ ดังนั้นเราจึงควรทราบหลักการเขียนโปรแกรมบน Arduino IDE ก่อนเริ่มเขียนโปรแกรม ซึ่งหลักการเขียนโปรแกรมบน Arduino IDE นั้น มี 5 ส่วนที่สำคัญคือ

##### 4.1. ส่วนของตัวประมวลผลก่อน (Preprocessor Directives)

จะอยู่ในบริเวณส่วนหัวของโปรแกรม (Header Program) คือสิ่งที่ตัวคอมไพเลอร์ (Compiler) จะต้องประมวลผล หรือ อ่านไฟล์ส่วนนี้ก่อนว่าจะให้ทำอะไร มีสิ่งใดที่จะต้องดำเนินการ ก่อนที่จะให้เข้าไปทำงานในส่วนของโปรแกรม (มานพ ปักซี่, 2562) จุดสังเกตคือเครื่องหมาย # นำหน้า จะมีอยู่ 2 ชนิดคือ

4.1.1. #include ใน Arduino IDE มีไว้สำหรับดึงไฟล์ไลบรารี (Library) ต่าง ๆ ที่มีใน Arduino IDE หรืออาจจะเป็นไลบรารีที่เราเขียนขึ้นมาเองหรือติดตั้งเพิ่มเติม วิธีการใช้มีอยู่ 2 แบบคือ

4.1.1.1. หากดึงไฟล์ไลบรารีมาตรฐานที่มีอยู่ใน Arduino IDE จะใช้เครื่องหมาย <ชื่อไฟล์> เช่น #include <Wire.h>

4.1.1.2. หากดึงไฟล์ไลบรารีที่เราเขียนขึ้นมาเองและอยู่ในโฟลเดอร์ (Folder) เดียวกันกับไฟล์โปรแกรม หรือติดตั้งเพิ่มเติม จะใช้เครื่องหมาย “ชื่อไฟล์” เช่น #include “mylib.h”

4.1.2. #define ใน Arduino IDE คือการนิยามหรือการกำหนดค่า ชื่อ หรือตัวอักษรใด ๆ เพื่อใช้อ้างอิงถึงสิ่งหนึ่ง หากเปรียบเทียบให้เห็นในชีวิตประจำวันอาจจะหมายถึงชื่อเล่น เช่น ชื่อ-สกุล ตันติกร โนนศรี และมีชื่อเล่นคือ เอก หากจะเขียนในรูปแบบ Arduino IDE จะได้เป็น #define เอก “ตันติกร โนนศรี” เวลาเรียกใช้งานก็สามารถเรียก เอก แทน ตันติกร โนนศรี ได้เลย หรืออาจจะกำหนดเป็นสมการทางคณิตศาสตร์ก็ได้ #define A 1+2 เวลาเรียก A ก็จะหมายถึง 1+2 เป็นต้น ประโยชน์ในการใช้ #define ก็เพื่อลดการเขียนโปรแกรมลง และทำให้ขนาดของไฟล์โปรแกรมมีขนาดเล็กลง

## 4.2. ส่วนของการกำหนดค่า (Global Declarations)

จะอยู่ในบริเวณเดียวกันหรือต่อจากส่วนของตัวประมวลผลก่อน จึงไม่ใช่คำสั่งที่ส่งผลต่อการทำงานของโมดูล (Module) เป็นการสร้างความเข้าใจให้กับคอมไพเลอร์ ซึ่งอาจจะเขียนรวมไปถึงคำสั่งส่วนที่ใช้ประกาศสร้าง (นพ มหิษานนท์, 2561)

4.2.1. ตัวแปร (Variable Declaration) โดยส่วนนี้เป็นส่วนที่ใช้ในการประกาศตัวแปรแบบนอกฟังก์ชัน เพื่อให้สามารถเรียกใช้ได้จากทุกส่วนของตัวโปรแกรม

4.2.2. ค่าคงที่ (Constant Declaration) โดยส่วนนี้เป็นส่วนที่ใช้ในการประกาศค่าคงที่แบบนอกฟังก์ชัน เพื่อให้สามารถเรียกใช้ได้จากทุกส่วนของตัวโปรแกรม เช่นเดียวกันกับตัวแปร

4.2.3. ฟังก์ชันต่าง ๆ (Function Declaration) โดยส่วนนี้เป็นส่วนที่ใช้ในการประกาศฟังก์ชัน เพื่อให้สามารถเรียกใช้ได้จากทุกส่วนของตัวโปรแกรม เช่นเดียวกันกับตัวแปรและค่าคงที่

## 4.3. ส่วนของโครงสร้าง (Structure)

ใน Arduino IDE จะมีโครงสร้างหลักคือฟังก์ชัน (Function) หลักอยู่ 2 ฟังก์ชันคือ

4.3.1. ฟังก์ชัน setup มีไว้สำหรับการตั้งสถานะของพอร์ต (Port) หรือพิน (Pin) หรือขาที่จะใช้ เนื่องจากไมโครคอนโทรเลอร์ในการทำงานต่าง ๆ สามารถทำงานได้ทั้งเป็นส่วนรับสัญญาณเข้ามา และส่วนส่งสัญญาณออกไปก็ได้ นอกจากการตั้งสถานะของพอร์ตแล้ว ยังเป็นส่วนที่ใช้ในการตั้งค่าความเร็วในการรับเข้าและส่งสัญญาณออก เพื่อนำไปเชื่อมต่อกับสิ่งอื่น ๆ เช่น เครื่องคอมพิวเตอร์ บลูทูธ (Bluetooth) WiFi หรือโมดูลสื่อสารต่าง ๆ เป็นต้น ในการทำงานของฟังก์ชัน setup โปรแกรมจะทำงานเพียงรอบเดียว ต่อจากการทำงานในส่วนของตัวประมวลผลก่อน

4.3.2. ฟังก์ชัน loop มีไว้เพื่อให้โปรแกรมทำงานซ้ำต่อเนื่อง โดยไม่หยุดจนกว่าจะมีการหยุดจ่ายไฟฟ้า หรือรีเซ็ต (Reset) การทำงานของบอร์ด (Board) เพื่อเริ่มการทำงานใหม่

## 4.4. ฟังก์ชัน (Functions) และฟังก์ชันที่สร้างขึ้นเอง (Users-Defined Function)

สำหรับการใช้งานฟังก์ชัน ในการสร้างฟังก์ชันขึ้นมา คำสั่งต่างๆที่อยู่ภายในฟังก์ชัน ต้องอยู่ภายใต้เครื่องหมายปีกกาเปิด { และปีกกาปิด } เท่านั้น ภายใต้เครื่องหมาย {} เราสามารถนำฟังก์ชันหรือคำสั่งใดๆก็ได้มาใส่ไว้ แต่จะต้องคั่นแต่ละคำสั่งด้วยเครื่องหมายอัฒภาค (Semicolon) “;” โดยจะนำคำสั่งทั้งหมดไว้บรรทัดเดียวกันเลย หรือแยกบรรทัดกันก็ได้เพื่อความสวยงามของโค้ด

#### 4.5. ส่วนของการอธิบายโปรแกรม (Program Comments)

ส่วนอธิบายโปรแกรม หรือการคอมเมนต์โปรแกรมเป็นส่วนที่สำคัญอย่างมากที่จะช่วยให้ผู้ที่ไม่ได้เขียนโปรแกรม หรือเป็นผู้เขียนโปรแกรมเข้าใจโปรแกรมได้ง่ายขึ้นโดยอ่านจากคอมเมนต์ แทนการทำความเข้าใจโปรแกรมโดยอ่านแต่ละฟังก์ชัน ส่วนอธิบายโปรแกรม หรือส่วนคอมเมนต์นี้ จะไม่มีผลใดๆกับขนาดของโปรแกรมหลังคอมไพล์ เนื่องจากส่วนนี้จะถูกตัดทิ้งทั้งหมดเนื่องจากไม่ได้ถูกนำไปใช้งาน มีผลเพียงแค่ไฟล์โค้ดโปรแกรมจะใหญ่ขึ้นมา หากมีการคอมเมนต์โค้ดเยอะๆ แต่ขนาดก็จะเพิ่มขึ้นตามตัวอักษร ดังนั้นการคอมเมนต์โค้ดจึงไม่คิดพื้นที่มากนัก แต่ผู้เขียนแนะนำให้คอมเมนต์โค้ดให้สั้น และกระชับ เพื่อให้เกิดความรวดเร็วในการทำควมเข้าใจ และไม่ยาวจนต้องเลื่อนสกอ์บาร์ไปทางขวาเพื่ออ่านคอมเมนต์เพิ่มเติมอีก การคอมเมนต์โค้ดมีอยู่ 2 รูปแบบ

4.5.1. แบบที่ 1 เปิดด้วย /\* และปิดด้วย \*/ เป็นการคอมเมนต์โค้ดแบบข้ามบรรทัด คือตราบใดที่ยังไม่มี \*/ ตรงส่วนนั้นจะเป็นคอมเมนต์ทั้งหมด

4.5.2. แบบที่ 2 เป็นการคอมเมนต์บรรทัดเดียว คือเปิดด้วยเครื่องหมาย // และปิดด้วยการขึ้นบรรทัดใหม่

### 5. พื้นฐานภาษา C สำหรับ Arduino

#### 5.1. ชนิดข้อมูลและการประกาศตัวแปร

##### 5.1.1. ชนิดของข้อมูล (เจ้าของร้าน, 2558)

ตารางที่ 6 แสดงชนิดของข้อมูล

ชนิดข้อมูล	การเก็บข้อมูล	ขนาด
boolean	จริง (True) หรือ เท็จ (False)	1 บิต
char	ตัวเลข หรือตัวอักษร	1 ไบต์ ใส่ค่าได้ตั้งแต่ -128 ถึง 127
unsigned char	ตัวเลข หรือตัวอักษร	1 ไบต์ ใส่ค่าได้ตั้งแต่ 0 ถึง 255
byte	ไบต์	1 ไบต์ ใส่ค่าได้ตั้งแต่ 0 ถึง 255
int	ตัวเลขจำนวนเต็ม	2 ไบต์ ใส่ค่าได้ตั้งแต่ -32,768 ถึง 32,767
unsigned int	ตัวเลขจำนวนเต็ม (ไม่นับจำนวนเต็มลบ)	2 ไบต์ ใส่ค่าได้ตั้งแต่ 0 ถึง 65,535 ( $(2^{16}) - 1$ )
long	ตัวเลขจำนวนเต็มที่มีความยาว	4 ไบต์ ใส่ค่าได้ตั้งแต่ -2,147,483,648 ถึง 2,147,483,647
unsigned long	ตัวเลขจำนวนเต็มที่มีความยาว (ไม่นับจำนวนเต็มลบ)	4 ไบต์ ใส่ค่าได้ตั้งแต่ 0 ถึง 4,294,967,295 ( $2^{32} - 1$ )

ชนิดข้อมูล	การเก็บข้อมูล	ขนาด
float	ตัวเลขทศนิยมใช้ในการคำนวณ	4 ไบต์ ใส่ค่าได้ตั้งแต่ - 3.4028235E+38 ถึง 3.4028235E+38 มีทศนิยมได้ 6 ถึง 7 ตำแหน่ง
double (เฉพาะบอร์ด Arduino Due)	ตัวเลขทศนิยมที่มีความยาวและต้องการความแม่นยำ	8 ไบต์ ใช้ในการคำนวณที่ต้องการประสิทธิภาพสูง
String	ข้อความ	ไม่ระบุ

### 5.1.2. การประกาศตัวแปร

การประกาศตัวแปรจะเหมือนกับภาษา C โดยปกติ คือ

TYPE KEY;

โดย TYPE เป็นชนิดของข้อมูล ส่วน KEY เป็นชื่อตัวแปร การประกาศตัวแปรข้างต้นคือการประกาศตัวแปรแบบไม่กำหนดค่า ดังนั้นค่าปกติที่อ่านจากตัวแปรจะเป็น 0

TYPE KEY = VAL;

จากตัวอย่างด้านบน เป็นลักษณะของการประกาศตัวแปรแบบกำหนดค่า เมื่ออ่านค่าของตัวแปรออกมา จะได้เป็นค่าที่กำหนดไว้ตอนประกาศ

int i;

int a = 10, b = 20;

จากตัวอย่าง จะเห็นว่าเราสามารถกำหนดชนิดของข้อมูลให้ทีเดียวหลายๆตัวแปรก็ได้ โดยใช้เครื่องหมาย , คั่นไว้

int i;

int a = 10, b = 20;

i = a + b;

จากตัวอย่างด้านบน เราสามารถกำหนดค่าให้กับตัวแปรเมื่อไรก็ได้ โดยใช้เครื่องหมายเท่ากับ = เป็นตัวเชื่อม ชื่อตัวแปรจะอยู่ทางซ้าย และจะกำหนดค่าเป็นอะไร ให้อยู่ทางขวา ค่าที่อยู่ทางขวาจะถูกนำมาใส่ในค่าที่อยู่ทางซ้ายเสมอ

int i = 10, a;

a = i;

จากตัวอย่างด้านบน จะเห็นว่า a ไม่ได้กำหนดค่าไว้ตอนประกาศ ทำให้ค่าที่อ่านได้จาก a คือ 0 แต่บรรทัดถัดมา มีการกำหนดค่าให้ a เท่ากับ i ซึ่งตอนประกาศ i ได้ประกาศไว้ว่าค่าเท่ากับ 10 เมื่อนำมาใส่ a ค่าที่อ่านได้จาก a จึงเป็น 10 ด้วยเช่นกัน



```
boolean is = false;
```

```
is = !is;
```

จากตัวอย่างด้านบน มีการประกาศตัวแปร boolean ซึ่งเป็นตัวแปรทางลอจิก มีค่าเป็น True (1) หรือ False (0) ได้เท่านั้น ในบรรทัดแรกได้ประกาศว่าตัวแปร is เป็นตัวแปรชนิด boolean และมีค่าเป็น false หรือลอจิก 0 บรรทัดต่อมา ได้มีการกำหนดให้ is เท่ากับ !is การที่เครื่องหมายนิเสธไปอยู่หน้าตัวแปร หมายถึงการกลับเป็นค่าตรงข้าม จากบรรทัดแรก ตัวแปร is มีค่าเป็น false เมื่อเจอ !is ค่าจึงถูกกลับเป็น true และถูกนำไปใช้ในตัวแปร is ทำให้สุดท้ายแล้วตัวแปร is มีค่าเป็น true

```
String text = "IoT Loie";
```

จากตัวอย่างด้านบน ได้มีการประกาศตัวแปรชื่อ text เป็นชนิด String เมื่ออ่านค่าที่ได้จากข้อความจึงได้ค่าออกมาเป็น "IoT Loie"

### 5.1.3. ตัวแปรอาร์เรย์ (Array)

อาร์เรย์เปรียบเหมือนกับการสร้างตารางขึ้นมา 1 ตาราง ซึ่งชื่อของตารางนั้นเปรียบได้กับชื่อของตัวแปร การเพิ่มข้อมูลของตารางลงไป เปรียบเหมือนกับการเพิ่มข้อมูลลงในอาร์เรย์ และการจะดึงข้อมูลตารางออกมาได้ เราก็จำเป็นต้องรู้ว่าข้อมูลที่เราต้องการนั้นอยู่ลำดับที่เท่าไร ก็เปรียบได้กับเราต้องรู้ว่าเราเพิ่มข้อมูลอาร์เรย์ไปในลำดับที่เท่าไร

ตารางที่ 7 ตัวอย่างตารางข้อมูลอาร์เรย์

ลำดับร้าน	ราคาสินค้า Arduino Uno R3
1	280
2	320
3	320
4	380

จากตัวอย่างตารางข้างต้น จะสามารถประกาศตัวแปรอาร์เรย์คือ `int price[4] = {280, 320, 320, 380};` และเวลาอ้างอิงจะต้องกำหนด `price[x]` โดยที่ `x` จะเป็นลำดับเริ่มต้นจาก 0 ในลำดับที่ 1 เช่น `price[1]` จะได้ค่าคือ 320 เป็นต้น

## 5.2. ค่าคงที่ (Constants)

ค่าคงที่ คือค่าที่ถูกกำหนดไว้แล้วไม่สามารถเปลี่ยนได้ ซึ่งในการเขียนโปรแกรมใน Arduino IDE นั้นสามารถใช้งานได้ 2 รูปแบบ ดังนี้

5.2.1. ค่าคงที่เริ่มต้น เป็นค่าคงที่ที่ถูกกำหนดมาแล้วใน Arduino IDE สามารถเรียกใช้ได้ทันที โดยที่ไม่จำเป็นต้องมาประกาศใหม่

ตารางที่ 8 ค่าคงที่เริ่มต้น

ตัวแปร	คำอธิบาย
HIGH	เป็นค่าสูงสุดของสถานะดิจิทัล มีค่าเป็น 1 หรือมีสัญญาณไฟฟ้า
LOW	เป็นค่าต่ำสุดของสถานะดิจิทัล มีค่าเป็น 0 หรือไม่มีสัญญาณไฟฟ้า
INPUT	เป็นค่าการทำงานของขา (Pin) ที่มีลักษณะการทำงานเป็นขารับสัญญาณไฟฟ้าเข้ามา
OUTPUT	เป็นค่าการทำงานของขา (Pin) ที่มีลักษณะการทำงานเป็นขาส่งสัญญาณไฟฟ้าออกไป
INPUT_PULLUP	เป็นค่าการทำงานของขา (Pin) ที่มีลักษณะการทำงานเป็นขารับสัญญาณไฟฟ้าเข้ามา และเป็นการเปิดการใช้งาน Resistant Pull Up ภายในตัว IC
LED_BUILTIN	เป็นขาที่ทำการเชื่อมต่อกับหลอด LED ที่ติดตั้งมากับตัวบอร์ด ซึ่งบอร์ดแต่ละค่ายจะถูกกำหนดมาต่างกัน
true	เป็นค่าทางตรรกะ ที่มีค่าเป็นจริง หรือมีค่าเป็นค่า 1
false	เป็นค่าทางตรรกะ ที่มีค่าเป็นเท็จ หรือมีค่าเป็นค่า 0

5.2.2. ค่าคงที่แบบกำหนดเอง เป็นค่าคงที่ที่ต้องการ สามารถประกาศได้ตามรูปแบบดังนี้

const ประเภทตัวแปร ชื่อตัวแปร

ตัวอย่างเช่น

`const int num=100; //เป็นประกาศค่าคงที่ประเภท int ใช้ชื่อตัวแปร num ซึ่งกำหนดค่าให้เท่ากับ 100 ซึ่งค่านี้จะไม่สามารถเปลี่ยนแปลงในโปรแกรมได้`

### 5.3. ตัวดำเนินการ (Operation)

การดำเนินการในการเขียนโปรแกรมด้วย Arduino IDE มีอยู่ 3 ประเภท คือ การคำนวณทางคณิตศาสตร์ การเปรียบเทียบ และการดำเนินการทางตรรกศาสตร์ ซึ่งการดำเนินการแต่ละประเภทจะมีเครื่องหมายที่ต้องใช้เพื่อเขียนคำสั่งสำหรับการดำเนินการประเภทนั้นๆ ดังรายละเอียด

5.3.1. เครื่องหมายการคำนวณทางคณิตศาสตร์ การดำเนินการพื้นฐานที่สุดทั้งในชีวิตประจำวันและในการเขียนโปรแกรมคือ การคำนวณทางคณิตศาสตร์ โดยเครื่องหมายที่ใช้สำหรับการคำนวณทางคณิตศาสตร์สรุปดังนี้

ตารางที่ 9 เครื่องหมายการคำนวณทางคณิตศาสตร์

เครื่องหมาย	ความหมาย	ตัวอย่าง
+	บวก	3+2 การบวก เลข 3 บวกกับ 2 ได้ผลลัพธ์คือ 5
-	ลบ	3-2 การลบ เลข 3 ลบกับ 2 ได้ผลลัพธ์คือ 1

เครื่องหมาย	ความหมาย	ตัวอย่าง
*	คูณ	2*3 การคูณ เลข 3 บวกกับ 2 ได้ผลลัพธ์คือ 6
/	หาร	15/2 การหาร เลข 15 หารกับ 2 ได้ผลลัพธ์คือ 7
%	หารเอาเศษ	15%2 การหารเอาเศษ เลข 15 หารกับ 2 ได้ผลลัพธ์คือ 1
++	เพิ่มค่าขึ้น 1 โดย	b=a++; จะมีความหมายเทียบเท่ากับ 2 บรรทัดต่อไปนี้ b=a; a=a+1; หรือ b=++a; จะมีความหมายเทียบเท่ากับ 2 บรรทัดต่อไปนี้ a=a+1; b=a;
--	ลดค่า 1 โดย	b=a--; จะมีความหมายเทียบเท่ากับ 2 บรรทัดต่อไปนี้ b=a; a=a-1; หรือ b=--a; จะมีความหมายเทียบเท่ากับ 2 บรรทัดต่อไปนี้ a=a-1; b=a;

5.3.2. เครื่องหมายการคำนวณทางคณิตศาสตร์ประเภทลดรูป นอกจากเครื่องหมายคำนวณตามปกติ และเครื่องหมายคำนวณแบบเพิ่ม/ ลดค่าแล้ว การคำนวณทางคณิตศาสตร์ในภาษาซียังมีเครื่องหมายที่เรียกว่า ลดรูปได้อีกด้วยดังแสดงรายละเอียดในตารางต่อไปนี้

ตารางที่ 10 เครื่องหมายการคำนวณทางคณิตศาสตร์ประเภทลดรูป

เครื่องหมาย	การดำเนินการ	ตัวอย่าง	ความหมาย
+=	บวกแบบลดรูป	$z += x$	บวกค่าในตัวแปร $z$ กับค่าของตัวแปร $x$ ผลลัพธ์ นำไปเก็บไว้ในตัวแปร $z$ มีค่าเหมือนกับ $z = z + x$

เครื่องหมาย	การดำเนินการ	ตัวอย่าง	ความหมาย
$-=$	ลบแบบลดรูป	$z -= x$	ลบค่าในตัวแปร $z$ ด้วยค่าของตัวแปร $x$ ผลลัพธ์นำไปเก็บไว้ในตัวแปร $z$ มีค่าเหมือนกับ $z = z - x$
$*=$	คูณแบบลดรูป	$z *= x$	คูณค่าในตัวแปร $z$ กับค่าของตัวแปร $x$ ผลลัพธ์นำไปเก็บไว้ในตัวแปร $z$ มีค่าเหมือนกับ $z = z * x$
$/=$	หารแบบลดรูป	$z /= x$	หารค่าในตัวแปร $z$ ด้วยค่าของตัวแปร $x$ ผลลัพธ์นำไปเก็บไว้ในตัวแปร $z$ มีค่าเหมือนกับ $z = z / x$
$\%=$	หารเอาเศษแบบลดรูป	$z \% = x$	หารค่าในตัวแปร $z$ กับค่าของตัวแปร $x$ ผลลัพธ์คือเศษที่ได้จากการหารนำไปเก็บไว้ในตัวแปร $z$ มีค่าเหมือนกับ $z = z \% x$

5.3.3. ส่วนใหญ่แล้วการดำเนินการเปรียบเทียบจะทำงานร่วมกับการดำเนินการอื่นๆ เช่น เปรียบเทียบผลจากการคำนวณทางคณิตศาสตร์ หรือเปรียบเทียบเพื่อกำหนดเงื่อนไขร่วมกับคำสั่งอื่น เช่น คำสั่ง if หรือ while เป็นต้น ทำให้ในหลายๆครั้ง เราเห็นภาพของการดำเนินการเปรียบเทียบไม่ชัดเจนทั้งที่การเปรียบเทียบเป็นการดำเนินการสำคัญไม่น้อยไปกว่าการดำเนินการในรูปแบบอื่นๆ เครื่องหมายที่ใช้ในการเปรียบเทียบแสดงดังตารางต่อไปนี้

ตารางที่ 11 เครื่องหมายการดำเนินการเปรียบเทียบ

เครื่องหมาย	การดำเนินการ	ตัวอย่าง	ความหมาย
$==$	เท่ากับ	$x == y$	ถ้าค่าของตัวแปร $x$ เท่ากับค่าของตัวแปร $y$ ผลลัพธ์จะออกมาเป็นจริง
$!=$	ไม่เท่ากับ	$x != y$	ถ้าค่าของตัวแปร $x$ ไม่เท่ากับค่าของตัวแปร $y$ ผลลัพธ์จะออกมาเป็นจริง
$<$	น้อยกว่า	$x < y$	ถ้าค่าของตัวแปร $x$ น้อยกว่าค่าของตัวแปร $y$ ผลลัพธ์จะออกมาเป็นจริง
$<=$	น้อยกว่าหรือเท่ากับ	$x <= y$	ถ้าค่าของตัวแปร $x$ น้อยกว่าหรือเท่ากับตัวแปร $y$ ผลลัพธ์จะออกมาเป็นจริง
$>$	มากกว่า	$x > y$	ถ้าค่าของตัวแปร $x$ มากกว่าค่าของตัวแปร $y$ ผลลัพธ์จะออกมาเป็นจริง

$\geq$	มากกว่าหรือเท่ากับ	$x \geq y$	ถ้าค่าของตัวแปร $x$ มากกว่าหรือเท่ากับค่าของตัวแปร $y$ ผลลัพธ์จะออกมาเป็นจริง
--------	--------------------	------------	---

5.3.4. เครื่องหมายทางตรรกศาสตร์ เครื่องหมายทางตรรกศาสตร์เป็นการหาผลลัพธ์โดยใช้หลักการทางตรรกศาสตร์ ส่วนใหญ่มักใช้ในระบบคอมพิวเตอร์หรือระบบเกี่ยวกับวงจรไฟฟ้า โดยเครื่องหมายทางตรรกศาสตร์ที่ใช้ในการเขียนโปรแกรมภาษาซี มีอยู่ 3 ชนิด คือ && (and), || (or) และ ! (not) ผลลัพธ์จากการดำเนินการทางตรรกศาสตร์โดยใช้เครื่องหมายทั้ง 3 ชนิด แสดงดังตารางต่อไปนี้โดยกำหนดให้ T แทนค่าที่เป็นจริง และ F แทนค่าที่เป็นเท็จ

ตารางที่ 12 เครื่องหมายทางตรรกศาสตร์

x	y	$x \& y$	$x    y$	$!x$
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

5.3.5. นิพจน์ คือ การนำข้อมูลหรือตัวแปรในภาษาซี มาดำเนินการโดยใช้เครื่องหมายทางคณิตศาสตร์ ตรรกศาสตร์ หรือเครื่องหมายเปรียบเทียบที่มีความหมายในภาษาซีเป็นตัวสั่งงาน

5.3.5.1. นิพจน์ทางคณิตศาสตร์ การเขียนนิพจน์ทางคณิตศาสตร์ในภาษาซีจะเหมือนกับการเขียนนิพจน์ทางคณิตศาสตร์เพียงแต่เปลี่ยนมาใช้เครื่องหมายทางคณิตศาสตร์ของภาษา C แทน

ตารางที่ 13 นิพจน์ทางคณิตศาสตร์

นิพจน์ทางคณิตศาสตร์ตามปกติ	นิพจน์ทางคณิตศาสตร์ในภาษาซี
$x + y - z$	$x + y - z$
$2xy + 4z$	$2 * x * y + 4 * z$
$x^2 + 2x + 1$	$x * x + 2 * x + 1$
$2r$	$2 * r$
$b^2$	$b * b$

5.3.5.2. นิพจน์ทางตรรกศาสตร์ การเขียนนิพจน์ทางตรรกศาสตร์ในภาษาซีคือ การเขียนนิพจน์โดยใช้เครื่องหมายการดำเนินการทางตรรกศาสตร์ในภาษาซี (&&, ||, !) เป็นตัวสั่งงาน ตัวอย่างนิพจน์ทางตรรกศาสตร์พร้อมทั้งผลลัพธ์จากการดำเนินการ แสดงดังตารางต่อไปนี้โดยกำหนดให้ตัวแปร  $a = 25$ ,  $b = 124$  และ  $c = 0$

ตารางที่ 14 นิพจน์ทางตรรกศาสตร์

นิพจน์ทางตรรกศาสตร์	การดำเนินการ	ผลลัพธ์
$c \& (a \leq b)$	$F \& F$	F

$(b > c) \parallel (c \leq a)$	$F \parallel F$	F
$(a > b) \&\& (c \leq a)$	$T \&\& T$	T
$!(c) \parallel (a == b)$	$T \parallel F$	T
$(a + 5) > (b - 100)$	$30 > -224$	T

5.3.6. ลำดับความสำคัญเครื่องหมาย ส่วนใหญ่นิพจน์ที่เขียนขึ้นในโปรแกรมมักมีความซับซ้อน มีการดำเนินการหลายอย่างปะปนอยู่ภายในนิพจน์เดียวกัน ยกตัวอย่างเช่น  $a/b + 15 * c$  หรือ  $(a - b) * 10 / c \&\& d + 5$  ซึ่งผลลัพธ์จะออกมาเป็นอย่างไรนั้น ต้องพิจารณาจากลำดับความสำคัญก่อนหลังของเครื่องหมายที่ภาษาซีกำหนดไว้ ดังตารางต่อไปนี้

ตารางที่ 15 ลำดับความสำคัญเครื่องหมาย

ลำดับความสำคัญ	เครื่องหมาย
1	( )
2	! , ++, -- , (typecast)
3	*, / , %
4	+, -
5	< , <= , > , >=
6	== , !=
7	&&
8	
9	*= , /= , %= , += , -=

#### 5.4. ตัวควบคุมทิศทางการเขียนโปรแกรม

5.4.1. คำสั่งตัดสินใจ เป็นกระบวนการตรวจสอบเพื่อให้โปรแกรมทำงานตามขั้นตอนที่ได้กำหนดไว้ตามเงื่อนไข (Condition) โดยจะมีการนำเอาค่าของตัวแปร หรือนิพจน์ต่าง ๆ มาเปรียบเทียบกับตรรกศาสตร์ว่าผลลัพธ์ที่เปรียบเทียบนั้น ว่ามีค่าเป็นจริงหรือเท็จ หากมีค่าเป็นจริงก็จะทำงานตามคำสั่งหรือชุดคำสั่งของเงื่อนไขนั้นๆ หากเป็นเท็จก็จะหยุดการทำงานแล้วไปทำงานตามคำสั่งต่อไปของโปรแกรม คำสั่งที่ใช้กำหนดเงื่อนไข ได้แก่ คำสั่ง if ประเภทต่าง ๆ และคำสั่ง Switch-case เป็นต้น

5.4.1.1. If เป็นคำสั่งที่กำหนดการเลือกกระทำตามเงื่อนไข เมื่อเงื่อนไขเป็นจริง (True) ก็จะทำตามคำสั่ง หรือ ชุดคำสั่ง (Statements) แต่หากเงื่อนไขเป็นเท็จ (False) ก็จะไม่มีการทำงานใด ๆ

รูปแบบคำสั่ง

if (condition) statement ;

หรือการนำเอาเครื่องหมาย { } มาช่วยในการทำงานแบบหลายคำสั่ง

if (condition){

```

Statement1;
Statement2;
StatementN;
}

```

5.4.1.2. if-else เป็นคำสั่งที่กำหนดการเลือกกระทำตามเงื่อนไข เมื่อเงื่อนไขเป็นจริง (True) ก็ทำตามคำสั่งหลังเงื่อนไข หากเป็นเท็จก็จะทำตามคำสั่งหลัง else ตามคำสั่งที่กำหนดไว้

รูปแบบคำสั่ง

```

if (condition) statement;
else
    statement
หรือการนำเอาเครื่องหมาย { } มาช่วยในการทำงานแบบหลายคำสั่ง
if (condition){
    statement1;
    statement2;
    statementN;
} else {
    statement1;
    statement2;
    statementN;
}

```

5.4.1.3. if-else-if เป็นคำสั่งที่กำหนดการเลือกกระทำตามเงื่อนไขที่ 1 เมื่อเงื่อนไขเป็นจริง (True) ก็ทำตามคำสั่งหลังเงื่อนไข หากเป็นเท็จก็จะทำตามคำสั่งหลัง else และจะมีการตรวจสอบเงื่อนไขที่ 2 และเงื่อนไขต่อ ๆ ไปอีก ถ้าเงื่อนไขเป็นจริงก็จะทำงานตามเงื่อนไขที่ได้กำหนดไว้

รูปแบบคำสั่ง

```

if (condition-1)
    statement;
else if (condition-2)
    statement;
else if (condition-N)
    statement;
หรือการนำเอาเครื่องหมาย { } มาช่วยในการทำงานแบบหลายคำสั่ง

```

```

if (condition-1) {
    statement;
    statement;
} else if (condition-2) {
    statement;
    statement;
} else if (condition-N) {
    statement;
    statement;
} else {
    statement;
    statement;
}

```

5.4.1.4. Nested if เป็นการเขียนคำสั่ง if ซ้อน if เพื่อเป็นการตรวจสอบเงื่อนไขแบบซับซ้อนมากขึ้น โดยจะมีการตรวจสอบเงื่อนไขที่ 1 ก่อน ถ้าเป็นจริงก็จะตรวจสอบเงื่อนไขที่ 2 ต่อไป เช่น ตรวจสอบว่าเป็นนักศึกษาชาย และมีอายุมากกว่า 20 ปี เป็นต้น

รูปแบบคำสั่ง

```

if (condition-1) {
    if (condition-2) {
        if (condition-N)
            statement;
    }
}

```

หรือการนำเอาเครื่องหมาย { } มาช่วยในการทำงานแบบหลายคำสั่ง

```

if (condition-1) {
    if (condition-2) {
        if (condition-N) {
            statement1;
            statement2;
            statementN;
        }
    }
}

```



```

    }
}

```

5.4.1.5. switch-case จะใช้ในกรณีที่มีหลายทางเลือก แต่ใช้ค่าของตัวแปรเพียงตัวเดียวมาตรวจสอบกับค่าคงที่ถ้าตรวจสอบแล้วมีค่าตรงกับค่าคงที่ใดก็จะไปทำงานส่วนการทำงานของค่าคงที่นั้น และหากค่าที่นำมาตรวจสอบไม่ตรงกับ ค่าคงที่ใดๆ ก็ทำงานในส่วนของ default

รูปแบบคำสั่ง

```

switch(variable) {
case constant1:
    statement;
    break;
case constant2:
    statement;
    break;
case constant3:
    statement;
    break;
case constant4:
    statement;
    break;
default:
    statement;
}

```

หรือการทำงานในแต่ละเงื่อนไขแบบหลายคำสั่ง สามารถเขียนชุดคำสั่ง (statement) เพิ่มได้ แล้วจบด้วยคำสั่ง break

```

switch(variable) {
case constant1:
    statement;
    statement;
    break;
case constant2:
    statement;
}

```

```

        statement;
        break;
    case constant3:
        statement;
        statement;
        break;
    case constant4:
        statement;
        statement;
        break;
    default:
        statement;
        statement;
}

```

5.4.2. คำสั่งทำซ้ำ ในการวนรอบการทำงานในโปรแกรมคอมพิวเตอร์ของแต่ละภาษาที่ถูกเขียนขึ้นมา นั้นจะต้องอาศัยฟังก์ชันในการวนรอบ ซึ่งแต่ละภาษานั้นก็จะมีคำสั่งหรือชุดคำสั่งที่คล้ายกัน ในภาษาซีนั้นจะใช้ คำสั่งในการวนรอบอยู่ 3 คำสั่งคือ (รุ่งทิวา เสาร์สิงห์, 2549)

5.4.2.1. for เป็นคำสั่งที่ใช้ในการวนรอบ เมื่อต้องการวนรอบตามจำนวนรอบที่ต้องการ เช่น ต้องการวน 10 รอบ เป็นต้น มักจะใช้กับการเขียนโปรแกรมที่รู้จำนวนรอบที่แน่นอน คำสั่ง for มีรูปแบบการทำงานและรูปแบบคำสั่ง

```

รูปแบบคำสั่ง
for (initial; continue; change)
    statement;

```

หรือหากต้องการให้การวนรอบหนึ่งรอบสามารถทำงานได้หลายคำสั่งพร้อม ๆ กันในแต่ละรอบ สามารถที่จะใช้เครื่องหมาย { } เข้ามากำหนดส่วนของการทำงานเกี่ยวกับคำสั่ง หรือstatement หลายๆ ชุดได้

```

for (initial; condition; step) {
    statement1;
    statement2;
    statement3;
    statementN;
}

```

```
}
```

5.4.2.2. while เป็นคำสั่งที่ใช้ควบคุมการวนรอบของโปรแกรม โดยจะมีเงื่อนไข (Condition) เป็นตัวตรวจสอบว่า หากเงื่อนไขที่กำหนดไว้เป็นจริง ก็จะทำงานตามคำสั่ง หรือชุดคำสั่งที่กำหนดไว้แล้วก็จะวนไปตรวจสอบเงื่อนไขที่กำหนดไว้อีกครั้งหากเงื่อนไขยังคงเป็นจริงก็จะทำงานในรอบต่อไปอีก หากเป็นเท็จก็จะออกจากการวนรอบของ while เพื่อทำงานยังคำสั่งต่อไป

รูปแบบคำสั่ง

```
while (condition)
```

```
statement;
```

หรือหากต้องการให้การวนรอบแต่ละครั้ง สามารถทำงานได้หลายชุดคำสั่ง ก็จะนำเอาเครื่องหมาย { } มาเขียนไว้เพื่อเป็นการกำหนดขอบเขตของการทำงานทั้งหมด

```
while (condition) {
```

```
statement1;
```

```
statement2;
```

```
statement3;
```

```
statementN;
```

```
}
```

5.4.2.3. do...while นั้นมีการทำงานคล้ายกับคำสั่ง while ซึ่งจะมีการตรวจสอบเงื่อนไขการทำงาน หากเป็นจริงก็จะทำงานตามคำสั่ง หรือชุดคำสั่งที่ได้กำหนดไว้ แต่คำสั่ง do while นั้นจะทำงานก่อน 1 รอบ แล้วจึงมีการตรวจสอบเงื่อนไข หากเป็นจริง ก็จะทำงานต่อในรอบต่อไป หากเป็นเท็จก็จะออกจากการทำงานในรอบการทำงานนั้น แต่อย่างไรก็เกิดการทำงานแล้ว 1 รอบ

รูปแบบคำสั่ง

```
do
```

```
statement;
```

```
while (condition);
```

หรือหากต้องการให้แต่ละการวนรอบนั้น ทำงานมากกว่าหนึ่งคำสั่งก็สามารถทำได้โดยการใช้เครื่องหมาย { } ครอบชุดคำสั่ง (statement) ที่ต้องการดั่งรูปแบบ

```
do {
```

```
statement1;
```

```
statement2;
```

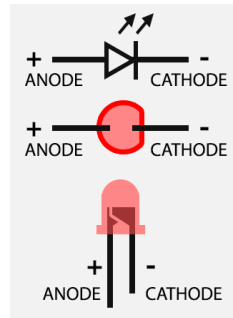
```
statement3;
```

```
statementN;
```

```
} while (condition);
```

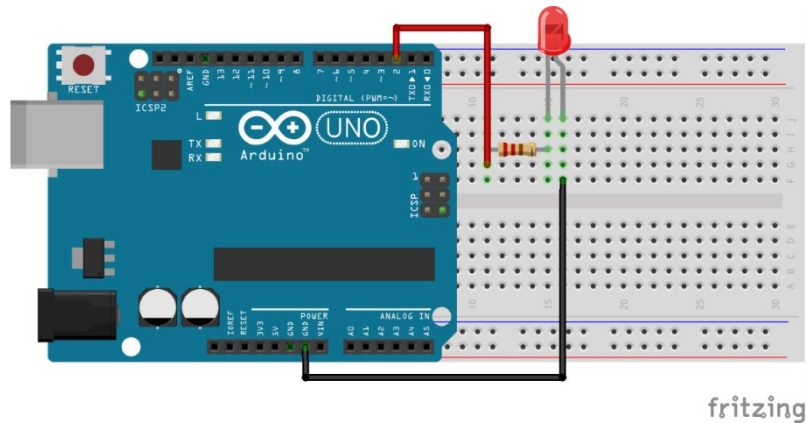
## 6. Basic Output

### 6.1. Digital Output



ภาพประกอบที่ 17 LED Pinout

ที่มา : (Arduino – Getting started with your first project, 2014)



ภาพประกอบที่ 18 รูปแบบการต่อวงจร LED

ตัวอย่างที่ 1 เปิด LED (Digital)

```
1. void setup()
2. {
3.     pinMode(2, OUTPUT);
4.     digitalWrite(2, HIGH);
5. }
6.
7. void loop()
8. {
9.
10. }
```

ตัวอย่างที่ 2 ไฟกระพริบ

```
1. void setup()
2. {
3.     pinMode(2, OUTPUT);
4. }
5.
6. void loop()
7. {
8.     digitalWrite(2, HIGH);
```

```

9.     delay(1000);
10.    digitalWrite(2, LOW);
11.    delay(1000);
12. }

```

## 6.2. Analog Output (LED)

ตัวอย่างที่ 3 เปิด LED (Analog)

```

1. void setup()
2. {
3.     pinMode(2, OUTPUT);
4. }
5.
6. void loop()
7. {
8.     analogWrite(2, 255);
9. }

```

## 6.3. Serial Monitor

ตัวอย่างที่ 4 Serial Monitor

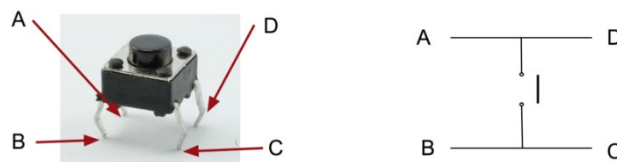
```

1. void setup()
2. {
3.     Serial.begin(9600);
4. }
5.
6. void loop()
7. {
8.     Serial.print("Hello World\t");
9.     Serial.print("Hello World")
10. }

```

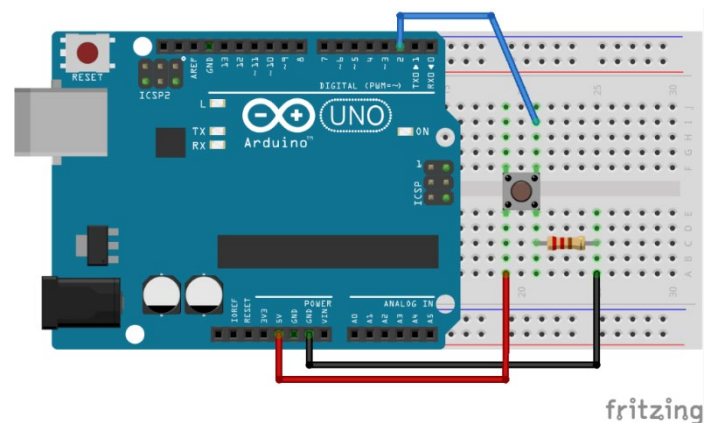
## 7. Basic Input

### 7.1. Digital Input

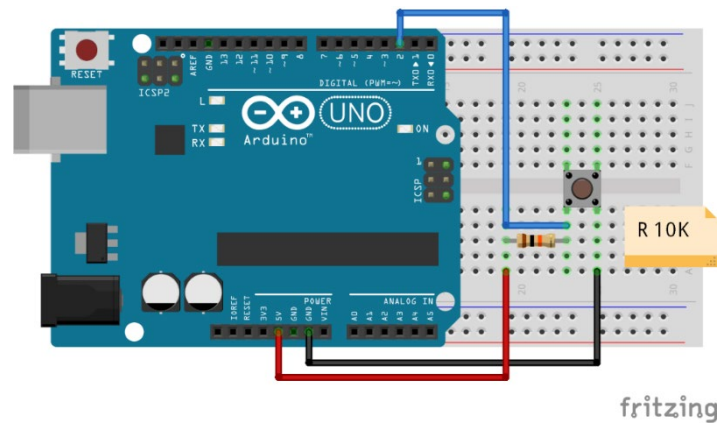


ภาพประกอบที่ 19 Button Pinout

ที่มา : (Arduino lesson – Button, 2017)



ภาพประกอบที่ 20 รูปแบบการต่อวงจร Button Pulldown



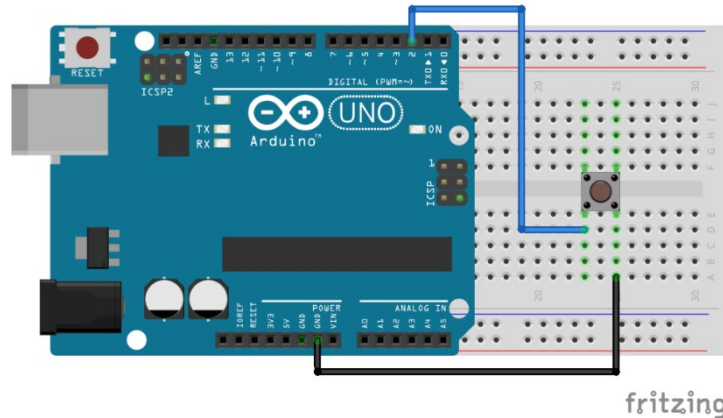
ภาพประกอบที่ 21 รูปแบบการต่อวงจร Button Pullup

ตัวอย่างที่ 5 การอ่านค่าจาก Button

```

1. void setup()
2. {
3.   Serial.begin(9600);
4.   pinMode(2, INPUT);
5. }
6.
7. void loop()
8. {
9.   unsigned int i = digitalRead(2);
10.  Serial.println(i);
11. }

```



ภาพประกอบที่ 22 การต่อวงจร Button โดยไม่ต้องต่อ R (Pullup)

ตัวอย่างที่ 6 การอ่านค่าจาก Button โดยไม่ต้องต่อ R (Pullup)

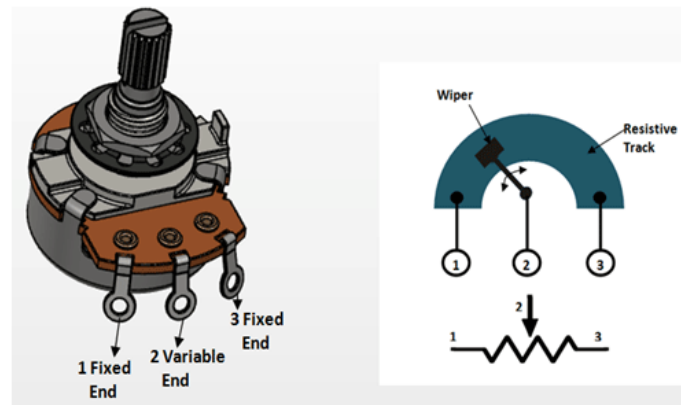
```

1. void setup()
2. {
3.   Serial.begin(9600);
4.   pinMode(2, INPUT_PULLUP);
5. }
6.
7. void loop()
8. {
9.   unsigned int i = digitalRead(2);
10.  Serial.println(i);
11. }

```

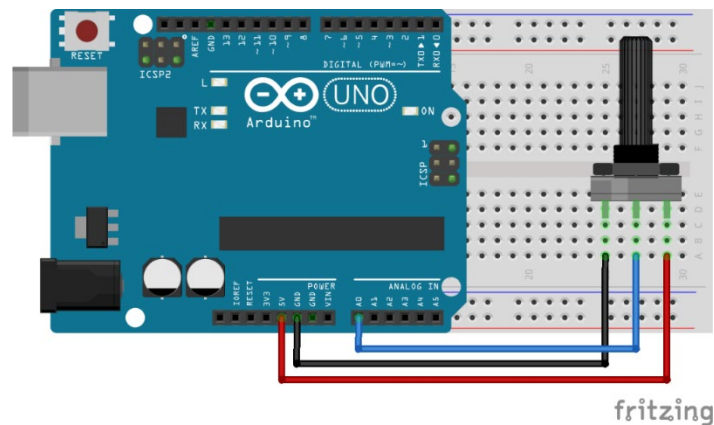
## 7.2. Analog Input

### 7.2.1. Variable Resistor, Potentiometer



ภาพประกอบที่ 23 Variable Resistor Pinout

ที่มา : (Potentiometer, 2017)



ภาพประกอบที่ 24 การต่อวงจร Variable Resistor

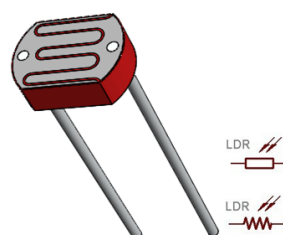
ตัวอย่างที่ 7 การอ่านค่าจาก Variable Resistor

```

1. void setup()
2. {
3.     Serial.begin(9600);
4. }
5.
6. void loop()
7. {
8.     unsigned int i = analogRead(A0);
9.     Serial.println(i);
10. }

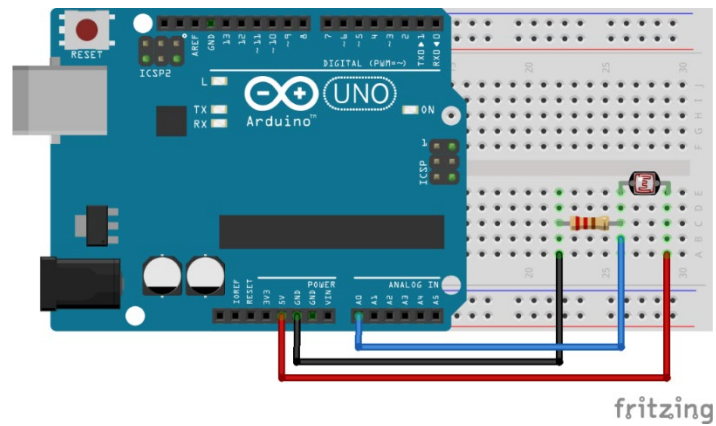
```

### 7.2.2. Light Decreasing Resistance : LDR, Photoresistor



ภาพประกอบที่ 25 LDR Pinout

ที่มา : (LDR, 2017)



ภาพประกอบที่ 26 การต่อวงจร LDR

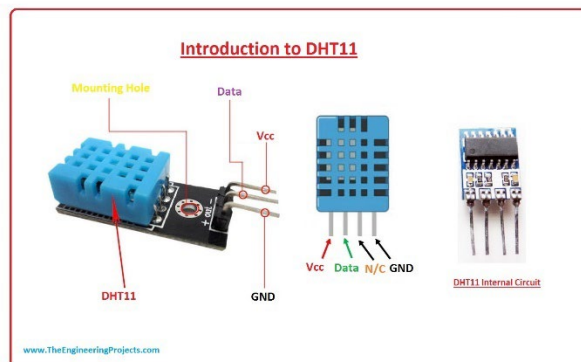
ตัวอย่างที่ 8 การอ่านค่าจาก LDR

```

1. void setup()
2. {
3.     Serial.begin(9600);
4. }
5.
6. void loop()
7. {
8.     unsigned int i = analogRead(A0);
9.     Serial.println(i);
10. }

```

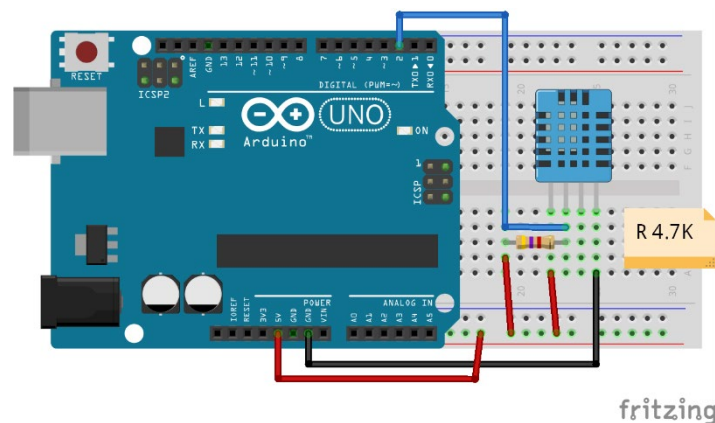
## 8. Temperature and Humidity Sensor (DHT11)



ภาพประกอบที่ 27 DHT11 Pinout

ที่มา : (Nasir, 2019)





ภาพประกอบที่ 28 การต่อวงจร DHT11

ก่อนที่จะดำเนินการเขียนโปรแกรมจะต้องทำการติดตั้ง Library สำหรับการใช้งาน DHT11 ก่อน โดยการไปที่เมนู Sketch -> Include Library -> Manage Libraries... หรือกด Ctrl + Shift + I จากนั้นค้นหาด้วยข้อความ “DHT sensor library” เมื่อพบแล้วคลิกเพื่อเลือก Library และกด Install รอจนกว่าจะติดตั้งเสร็จแล้วจึงกดปุ่ม Close

ตัวอย่างที่ 9 การอ่านค่าจาก DHT11

```

1. #include "DHT.h"
2.
3. #define DHTPIN 2    // Digital pin connected to the DHT sensor
4. #define DHTTYPE DHT11 // DHT 11
5. // #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
6. // #define DHTTYPE DHT21 // DHT 21 (AM2301)
7.
8. DHT dht(DHTPIN, DHTTYPE);
9.
10. void setup() {
11.   Serial.begin(9600);
12.   dht.begin();
13. }
14.
15. void loop() {
16.   delay(2000); // Reading temperature or humidity takes about 250 milliseconds!
17.   float h = dht.readHumidity(); // Read temperature as Celsius (the default)
18.   float t = dht.readTemperature(); // Read temperature as Fahrenheit (isFahrenheit
   = true)
19.   float f = dht.readTemperature(true);
20.   if (isnan(h) || isnan(t) || isnan(f)) { // Check if any reads failed and exit ea
   rly (to try again).
21.     Serial.println(F("Failed to read from DHT sensor!"));
22.     return;
23.   }
24.   float hif = dht.computeHeatIndex(f, h); // Compute heat index in Fahrenheit (the
   default)
25.   float hic = dht.computeHeatIndex(t, h, false); // Compute heat index in Celsius
   (isFahreheit = false)
26.
27.   Serial.print(F("Humidity: "));
28.   Serial.print(h);
29.   Serial.print(F("% Temperature: "));
30.   Serial.print(t);
31.   Serial.print(F("°C "));
32.   Serial.print(f);

```

```

33. Serial.print(F("°F Heat index: "));
34. Serial.print(hic);
35. Serial.print(F("°C "));
36. Serial.print(hif);
37. Serial.println(F("°F"));
38. }

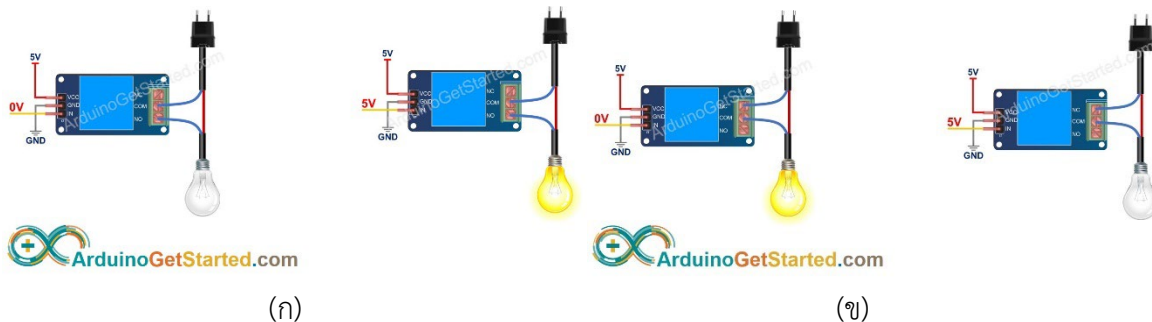
```

## 9. Control Switch (Relay Switch)



ภาพประกอบที่ 29 Relay Pinout

ที่มา : (Arduino - Relay, n.d.)

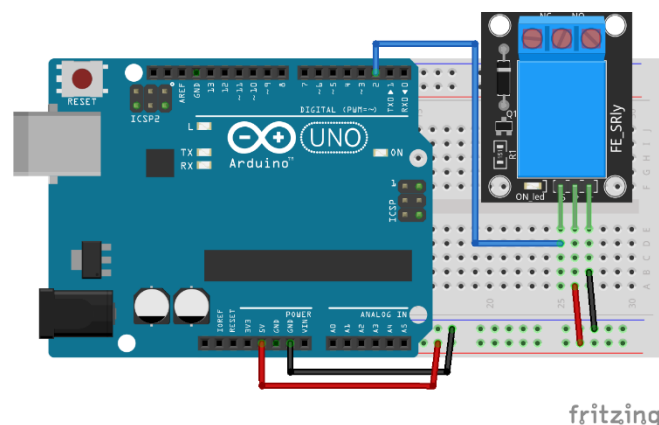


(ก)

(ข)

ภาพประกอบที่ 30 รูปแบบของ Relay (ก) Normally Open Mode (Active High) (ข) Normally Closed Mode (Active Low)

ที่มา : (Arduino - Relay, n.d.)



ภาพประกอบที่ 31 การต่อวงจร Relay

ตัวอย่างที่ 10 การสั่งงาน Relay

```

1. const int RELAY_PIN = 2;
2. void setup() {

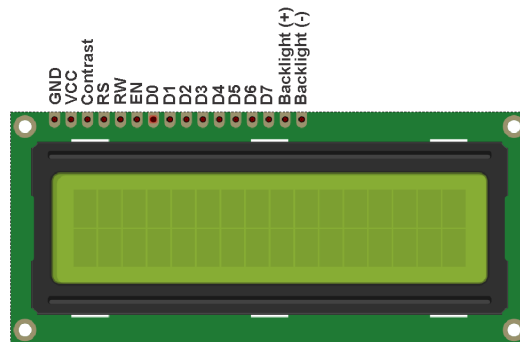
```

```

3.   pinMode(RELAY_PIN, OUTPUT);
4. }
5.
6. void loop() {
7.   digitalWrite(RELAY_PIN, HIGH);
8.   delay(500);
9.   digitalWrite(RELAY_PIN, LOW);
10.  delay(500);
11. }

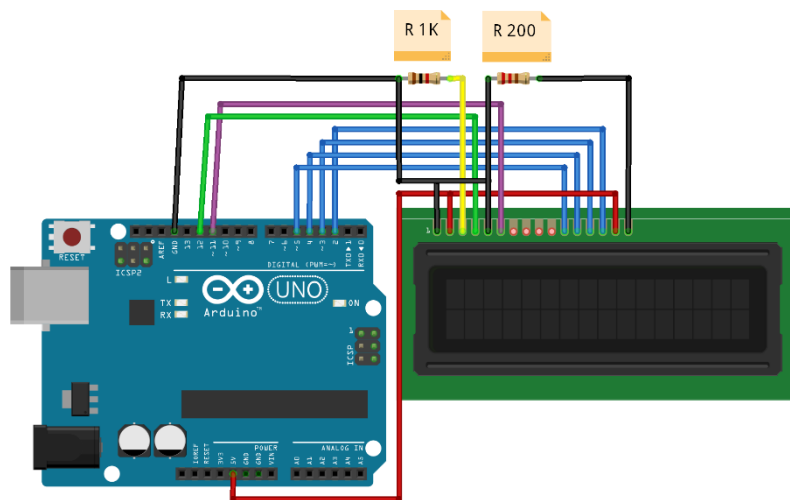
```

## 10. LCD Output (16x2)



ภาพประกอบที่ 32 LCD 16x2 Pinout

ที่มา : (16x2 LCD Display interface with Arduino, 2016)



fritzing

## ภาพประกอบที่ 33 การต่อวงจร LCD 16x2

ก่อนที่จะดำเนินการเขียนโปรแกรมจะต้องทำการติดตั้ง Library สำหรับการใช้งาน DHT11 ก่อน โดยการไปที่เมนู Sketch -> Include Library -> Manage Libraries... หรือกด Ctrl + Shift + I จากนั้นค้นหาคำว่า “LiquidCrystal” เมื่อพบแล้วคลิกเพื่อเลือก Library และกด Install รอจนกว่าจะติดตั้งเสร็จแล้วจึงกดปุ่ม Close

ตัวอย่างที่ 11 การแสดงผลทาง LCD

```

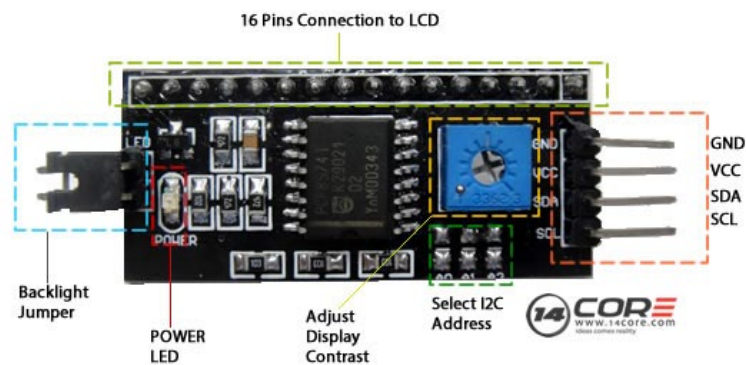
1. #include <LiquidCrystal.h>
2.
3. LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
4.

```

```

5. byte smiley[8] = {
6.     0b00000,
7.     0b00000,
8.     0b01010,
9.     0b00000,
10.    0b00000,
11.    0b10001,
12.    0b01110,
13.    0b00000
14. };
15.
16. void setup() {
17.     lcd.begin(16, 2); // set up the LCD's number of columns and rows:
18.     lcd.createChar(1, smiley); // create a new character
19.     lcd.write(1); //Print custom character smiley
20.     lcd.print("hello, world!");
21.     lcd.write(1);
22. }
23.
24. void loop() {
25.     lcd.setCursor(0, 1); // set the cursor to column 0, line 1
26.     lcd.print("Loop Display");
27. }

```



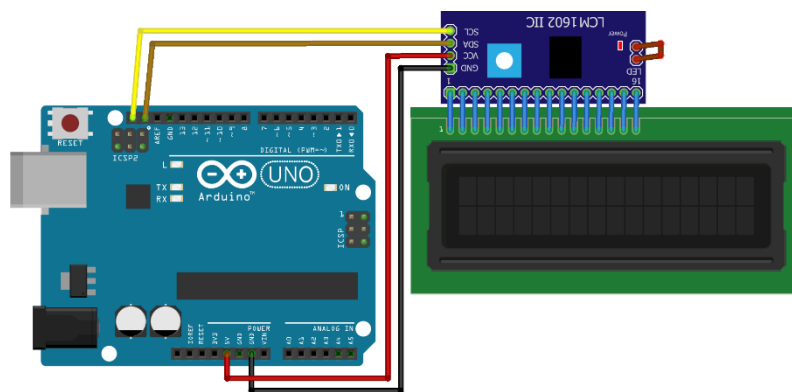
ภาพประกอบที่ 34 LCD 16x2 (I2C) Pinout

ที่มา : (14CORE, n.d.)

I2C ADDRESS	A0	A1	A2
0X20	0	0	0
0X21	1	0	0
0X22	0	1	0
0X23	1	1	0

ภาพประกอบที่ 35 I2C Address

ที่มา : (14CORE, n.d.)



fritzing

ภาพประกอบที่ 36 การต่อวงจร LCD 16x2 (I2C)

ก่อนที่จะดำเนินการเขียนโปรแกรมจะต้องทำการติดตั้ง Library สำหรับการใช้งาน DHT11 ก่อน โดยการไปที่เมนู Sketch -> Include Library -> Manage Libraries... หรือกด Ctrl + Shift + I จากนั้นค้นหาด้วยข้อความ “LiquidCrystal I2C” เมื่อพบแล้วคลิกเพื่อเลือก Library และกด Install รอจนกว่าจะติดตั้งเสร็จแล้วจึงกดปุ่ม Close

ตัวอย่างที่ 12 การแสดงผลทาง LCD (I2C)

```

1. #include <LiquidCrystal_I2C.h>
2.
3. LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a 16 chars and 2 line display
4.
5. void setup()
6. {
7.   lcd.init();
8.   lcd.backlight();
9.
10.  lcd.setCursor(0,0); // set the cursor to column 0, line 1
11.  lcd.print("Hello, world!");
12.  lcd.setCursor(0,1);
13.  lcd.print("Show Screen");
14. }
15.
16.
17. void loop()
18. {
19. }

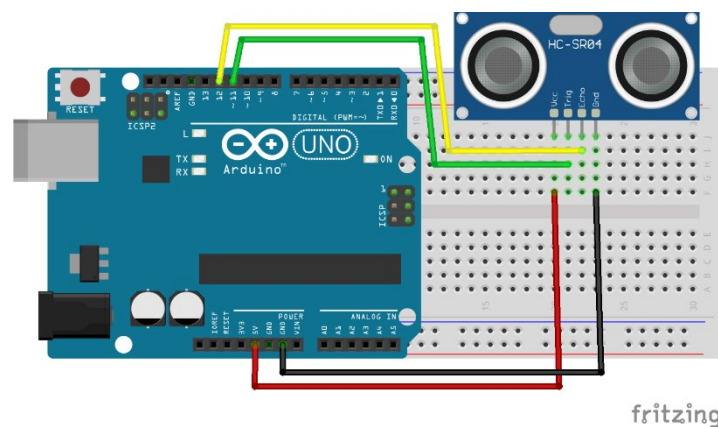
```

## 11. Distance Sensor (Ultrasonic)



ภาพประกอบที่ 37 Ultrasonic Sensor Pinout

ที่มา : (Aqeel, 2018)



ภาพประกอบที่ 38 การต่อวงจร Ultrasonic Sensor

ตัวอย่างที่ 13 การอ่านค่าจาก Ultrasonic Sensor

```

1. const int trigPin = 11;
2. const int echoPin = 12;
3.
4. long duration;
5. int distanceCM, distanceINCH;
6.
7. void setup() {
8.   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
9.   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
10.  Serial.begin(9600);
11. }
12.
13. void loop() {
14.
15.   digitalWrite(trigPin, LOW); // Clears the trigPin
16.   delayMicroseconds(2);
17.
18.   digitalWrite(trigPin, HIGH); // Sets the trigPin on HIGH state for 10 micro seconds
19.
20.   delayMicroseconds(10);
21.   digitalWrite(trigPin, LOW);
22.
23.   duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in microseconds
24.   // Calculating the distance
25.   distanceCM= duration*0.034/2;

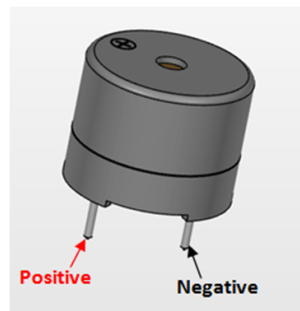
```

```

26. distanceINCH= duration*0.0133/2;
27.
28. Serial.print("Distance CM: ");
29. Serial.println(distanceCM);
30. Serial.print("Distance INCH: ");
31. Serial.println(distanceINCH);
32. }

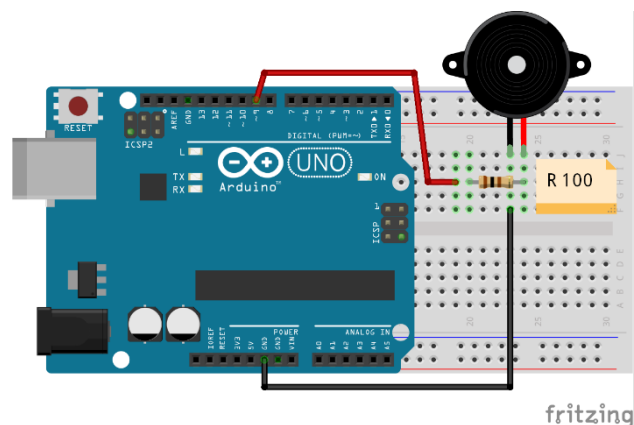
```

## 12. Alarm Control (Buzzer)



ภาพประกอบที่ 39 Buzzer Pinout

ที่มา : (Active Passive Buzzer, 2017)



ภาพประกอบที่ 40 การต่อวงจร Buzzer

ตัวอย่างที่ 14 การสั่งงาน Buzzer

```

1.  const int buzzer = 9;
2.
3.  void setup(){
4.    pinMode(buzzer, OUTPUT);
5.  }
6.
7.  void loop(){
8.    tone(buzzer, 1000); // Send 1KHz sound signal...Resonant Frequency: ~2300 Hz
9.    delay(1000);
10.   noTone(buzzer);    // Stop sound...
11.   delay(1000);
12. }

```

## เอกสารอ้างอิง

- เจ้าของร้าน. (2558, พฤษภาคม 18). *Arduino ตอนที่ 6 ตัวแปร และอาเรย์*. Retrieved from IOXhop: <https://www.ioxhop.com/article/7/arduino-ตอนที่6-ตัวแปร-และอาเรย์>
- นพ มหิษานนท์. (2561). *Arduino Startup สนุกสุดเหวี่ยงกับเซ็นเซอร์*. นนทบุรี: สำนักพิมพ์ คอร์ฟังก์ชั่น.
- บทความ Arduino คืออะไร ตอนที่1 แนะนำเพื่อนใหม่ที่ซื้อ Arduino. (2017, Mar 11). Retrieved from ThaiEasyElec: <https://www.thaieasyelec.com/article-wiki/latest-blogs/what-is-arduino-ch1.html>
- บทความ Arduino คืออะไร? ตอนที่2 มาทำความรู้จักกับ Arduino รุ่นต่างๆกัน. (2016, Sep 23). Retrieved from ThaiEasyElec: <https://www.thaieasyelec.com/article-wiki/basic-electronics/what-is-arduino-ch2-introduce-series-of-arduino.html>
- มานพ ปักชี. (2562). *Arduino ขั้นพื้นฐาน สำหรับผู้เริ่มต้น 1*. กรุงเทพฯ: สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย.
- รุ่งทิวา เสาร์สิงห์. (2549). *คู่มือเรียนรู้ภาษาซีด้วยตนเอง*. กรุงเทพฯ: ซีเอ็ดดูเคชั่น.
- วรรณรัช สันติอมรทัต, และ สกฤณา เจริญปัญญาศักดิ์. (7 ตุลาคม 2559). *Introduction to Wireless Sensor Networks*. เข้าถึงได้จาก ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ: <https://www.nectec.or.th/sectionFileDownload/3204>
- สุรศักดิ์ วีระวรวงศ์. (ม.ป.พ.). *KM - IT*. Retrieved from คณะจิตวิทยา จุฬาลงกรณ์มหาวิทยาลัย: <http://www.psy.chula.ac.th/psy/file/kmit/การใช้โปรแกรมเขียน Flowchart.pdf>
- 14CORE. (n.d.). Retrieved from Wiring I2C module on 16x2 LCD with SCL/SDA: <https://www.14core.com/wiring-i2c-module-on-16x2-lcd-with-sclsd/>
- 16x2 LCD Display interface with Arduino. (2016, May 15). Retrieved from circuits4you: <https://circuits4you.com/2016/05/15/how-to-lcd-display-arduino-uno/>
- Active Passive Buzzer. (2017, September 25). Retrieved from Components101: <https://components101.com/buzzer-pinout-working-datasheet>
- Aqeel, A. (2018, October 5). *Introduction to HC-SR04 (Ultrasonic Sensor)*. Retrieved from THE ENGINEERING PROJECTS: <https://www.theengineeringprojects.com/2018/10/introduction-to-hc-sr04-ultrasonic-sensor.html>



*Arduino – Getting started with your first project.* (2014, JAN 1). Retrieved from Tweaking4All.com - Computer Tips, Tricks, ... for everyone ...:  
<https://www.tweaking4all.com/hardware/arduino/first-arduino-project/>

*Arduino - Relay.* (n.d.). Retrieved from Arduino Get Started:  
<https://arduinogetstarted.com/tutorials/arduino-relay>

*Arduino lesson – Button.* (2017, July 02). Retrieved from Osoyoo.com:  
<https://osoyoo.com/2017/07/02/arduino-lesson-button/>

*LDR.* (2017, October 30). Retrieved from Components101: <https://components101.com/ldr-datasheet>

Nasir, S. Z. (2019, March 5). *Introduction to DHT11.* Retrieved from THE ENGINEERING PROJECTS: <https://www.theengineeringprojects.com/2019/03/introduction-to-dht11.html>

*Potentiometer.* (2017, September 29). Retrieved from Components101:  
<https://components101.com/potentiometer>