หน่วยที่ 5 ไลบรารี่โปรแกรมสำหรับ ARDUINO และการอ่านค่าแอนะล็อก

สาระสำคัญ

การพัฒนาโปรแกรมภาษา C/C++ ด้วย Arduino สำหรับบอร์ด Arduino ดำเนินการภายใต้การสนับสนุนของไฟล์ไลบรารีหลักที่ทาง Arduino จัดเตรียมให้ผนวกเข้ากับไฟล์ไลบรารีที่พัฒนาขึ้นมาเป็นเฉพาะสำหรับบอร์ด Arduino ทั้งนี้เพื่อช่วยลดความซับซ้อนในการเขียนโปรแกรมควบคุมลง Arduino IDE ได้บรรจุไฟล์ไลบรารีที่ช่วยให้การเขียนโปรแกรมภาษา C/C++ เพื่อให้ใช้งานไมโครคอนโทรลเลอร์ทำได้ง่ายขึ้น รวมถึงผู้เริ่มต้นใหม่ก็สามารถเรียนรู้เพื่อใช้งานไมโครคอนโทรลเลอร์ได้ โดยไม่จำเป็นต้องศึกษาเพื่อลงลึกในรายละเอียดของสถาปัตยกรรม

เนื้อหาสาระการเรียนรู้

5.1 ไลบรารี่โปรแกรมสำหรับ Arduino
5.2 ไลบรารี่เกี่ยวกับเวลา
5.3 ไลบรารี่เกี่ยวกับการอ่านค่าดิจิตอล
5.4 ไลบรารี่เกี่ยวกับการอ่านค่าแอนะล็อก
5.5 ไลบรารี่เกี่ยวกับการสื่อสารข้อมูลอนุกรม

จุดประสงค์การเรียนรู้

จุดประสงค์ทั่วไป

1. เพื่อให้มีความรู้ความเข้าใจเกี่ยวกับไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก
2. เพื่อให้สามารถนำความรู้ไปประยุกต์ใช้ในการเขียนโปรแกรมกำหนดการทำงานพื้นฐานของ Arduino
3. เพื่อให้ตระหนักถึงความสำคัญเกี่ยวกับไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก

จุดประสงค์เชิงพฤติกรรม

1. เลือกใช้งานไลบรารี่โปรแกรมสำหรับ Arduino ได้
2. เลือกใช้งานไลบรารี่เกี่ยวกับเวลาสำหรับ Arduino ได้
3. เลือกใช้งานไลบรารี่เกี่ยวกับการอ่านค่าดิจิตอลสำหรับ Arduino ได้
4. เลือกใช้งานไลบรารี่เกี่ยวกับการอ่านค่าแอนะล็อกสำหรับ Arduino ได้
5. เลือกใช้งานไลบรารี่เกี่ยวกับการสื่อสารข้อมูลอนุกรมสำหรับ Arduino ได้

แบบทดสอบก่อนเรียน หน่วยที่ 5

เรื่อง ไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก

เรื่อง ไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก **ใช้เวลา** 20 **นาที**

วิชา ไมโครคอนโทรลเลอร์เบื้องต้น **รหัสวิชา** (2127-2107)

ระดับชั้น ประกาศนียบัตรวิชาชีพ (ปวช.) **สาขาวิชา** เมคคาทรอนิกส์

****

คำชี้แจง 1. แบบทดสอบมีทั้งหมด 10 ข้อ (10 คะแนน)

2. ให้ผู้เรียนเลือกคำตอบที่ถูกที่สุดแล้วกาเครื่องหมายกากบาท (✕) ลงในกระดาษคำตอบ

1. หน่วยความจำข้อมูลอีอีพรอม ภายในตัวไมโครฯเมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีด้วยคำสั่ง
   - ก. #include< EEPROM.d>
   - ข. #include<EEPROM.o>
   - ค. #include<EEPROM. b>
   - ง. #include<EEPROM.h>
2. servo เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีด้วยคำสั่ง
   - ก. #include<servo.d>
   - ข. #include<servo.h>
   - ค. #include<servo.o>
   - ง. #include <servo.b>
3. Arduino UNO บรรจุฟังก์ชั่น Wire สำหรับติดต่อกับอุปกรณ์ผ่านบัส
   - ก. SPI
   - ข. GSM
   - ค. UART
   - ง. I2C
4. Arduino UNO บรรจุฟังก์ชั่นและคำสั่งสำหรับติดต่อกับ SPI โดยต้องใช้ขาพอร์ต
   - ก. 12 (MISO), 11 (MOSI) และ 10 (SS)
   - ข. 10 (MISO), 11 (MOSI) และ 12 (SS)
   - ค. 13 (MISO), 12 (MOSI) และ 11 (SS)
   - ง. 11 (MISO), 12 (MOSI) และ 13 (SS)

5. บัส I2C Arduino UNO ต้องใช้ขาพอร์ต
   ก. A4 (SDA) และ A5 (SCL)
   ข. A3 (SDA) และ A4 (SCL)
   ค. A2 (SDA) และ A3 (SCL)
   ง. A1 (SDA) และ A2 (SCL)
6. ฟังก์ชั่นเกี่ยวกับพอร์ตอินพุตเอาต์พุต ฟังก์ชั่น in เป็นฟังก์ชั่นอะไร
   ก. อ่านค่าสถานะลอจิกของพอร์ตที่กำหนด
   ข. กำหนดขาพอร์ตที่ต้องการอ่านค่า
   ค. อ่านค่าดิจิตอลจากพอร์ตอนุกรม
   ง. อ่านค่าดิจิตอลจากพอร์ตดิจิตอล
7. ฟังก์ชั่น out เป็นฟังก์ชั่นอะไร
   ก. กำหนดขาพอร์ตที่ต้องการอ่านค่า
   ข. กำหนดข้อมูลดิจิตอลไปยังพอร์ตที่กำหนด
   ค. อ่านค่าดิจิตอลจากพอร์ตดิจิตอล 2
   ง. อ่านค่าดิจิตอลจากพอร์ตดิจิตอล 1
8. ฟังก์ชั่นการอ่านค่าแอนะล็อกของ Arduino UNO ใช้พอร์ตอย่างไร
   ก. กำหนดช่องอินพุตแอนะล็อกที่ต้องการตรงกับขาพอร์ต A0 ถึง A6
   ข. กำหนดช่องอินพุตแอนะล็อกที่ต้องการตรงกับขาพอร์ต A1 ถึง A6
   ค. กำหนดช่องอินพุตแอนะล็อกที่ต้องการตรงกับขาพอร์ต A2 ถึง A6
   ง. กำหนดช่องอินพุตแอนะล็อกที่ต้องการตรงกับขาพอร์ต A3 ถึง A6
9. ฟังก์ชั่นเกี่ยวกับการสื่อสารข้อมูลอนุกรม เมื่อต้องการใช้งานช่อง UART ต่อสายสัญญาณอย่างไร
   ก. RxD (ขาพอร์ต A1) และ TxD (ขาพอร์ต A0)
   ข. RxD (ขาพอร์ต A0) และ TxD (ขาพอร์ต A1)
   ค. RxD (ขาพอร์ต 1) และ TxD (ขาพอร์ต 0)
   ง. RxD (ขาพอร์ต 0) และ TxD (ขาพอร์ต 1)
10. uart_available คือฟังก์ชั่นอะไร
   ก. ตัวแปรจำนวนเต็ม 64 บิต แบบไม่คิดเครื่องหมายของโมดูล UART
   ข. ปัดค่าตัวเลขให้อยู่ในช่วงที่กำหนดของโมดูล UART
   ค. ตรวจสอบการรับข้อมูลเข้ามาของโมดูล UART
   ง. ค่าต่ำสุดของช่วงที่กำหนดของโมดูล UART

หน่วยที่ 5
ไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก

การพัฒนาโปรแกรมภาษา C/C++ ด้วย Arduino สำหรับบอร์ด Arduino ดำเนินการภายใต้การสนับสนุนของไฟล์ไลบรารีหลักที่ทาง Arduino จัดเตรียมให้ ผนวกเข้ากับไฟล์ไลบรารีที่พัฒนาขึ้นมาเป็นเฉพาะสำหรับบอร์ด Arduino ทั้งนี้เพื่อช่วยลดความซับซ้อนในการเขียนโปรแกรมควบคุมลง Arduino IDE ได้บรรจุไฟล์ไลบรารีที่ช่วยให้การเขียนโปรแกรมภาษา C/C++ เพื่อใช้งานไมโครคอนโทรลเลอร์ทำได้ง่ายขึ้น

รวมถึงผู้เริ่มต้นใหม่ก็สามารถเรียนรู้เพื่อใช้งานไมโครคอนโทรลเลอร์ได้ โดยไม่จำเป็นต้องศึกษาเพื่อลงลึกในรายละเอียดของสถาปัตยกรรมของไมโครคอนโทรลเลอร์ ไฟล์ไลบรารีที่สำคัญและใช้งานกับบอร์ด Arduino ประกอบด้วย

5.1 ไลบรารี่โปรแกรมสำหรับ Arduino

- EEPROM บรรจุไลบรารี่และคำสั่งสำหรับติดต่อกับหน่วยความจำข้อมูลอีอีพรอม ภายในตัวไมโครคอนโทรลเลอร์ เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีไว้ในตอนต้นของโปรแกรมด้วยคำสั่ง
#include <EEPROM.h>

- LiquidCrystal บรรจุไลบรารี่และคำสั่งสำหรับติดต่อกับโมดูล LCD แบบอักขระเพื่อแสดงผลข้อความและตัวเลข รองรับทั้งการติดต่อแบบ 4 บิตและ 8 บิต ใช้งานได้กับโมดูล LCD 8, 16 และ 20 ตัวอักษร 1, 2 และ 4 บรรทัด เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีไว้ในตอนต้นของโปรแกรมด้วยคำสั่ง
#include <LiquidCrystal.h>

- servo บรรจุไลบรารี่และคำสั่งสำหรับขับเซอร์โวมอเตอร์ ต้องทำงานร่วมกับเซอร์โวมอเตอร์และต้องใช้ไฟเลี้ยงแยกสำหรับเซอร์โวมอเตอร์ เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีไว้ในตอนต้นของโปรแกรมด้วยคำสั่ง
#include <servo.h>

- SoftwareSerial บรรจุไลบรารี่และคำสั่งสำหรับใช้งานขาพอร์ตของ Arduino ในการสื่อสารข้อมูลอนุกรม ไลบรารีจะถูกนำมาใช้งานเมื่อ ขาเชื่อมต่อพอร์ตอนุกรมหลัก (RxD และ TxD) ของ Arduino ถูกใช้งานไปแล้ว และมีความต้องการติดต่ออุปกรณ์ที่ต้องใช้การสื่อสารข้อมูลอนุกรม การใช้งานไลบรารีนี้ จะช่วยให้ผู้ใช้งานสามารถใช้ขาพอร์ตของ Arduino ขาอื่นที่ว่างมาทำหน้าที่เป็นขาพอร์ตสำหรับสื่อสารข้อมูลอนุกรม เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีไว้ในตอนต้นของโปรแกรมด้วยคำสั่ง #include <SoftwareSerial.h>

- SPI บรรจุไลบรารี่และคำสั่งสำหรับติดต่อกับอุปกรณ์ผ่านบัส SPI (Serial Peripheral Interface) โดยต้องใช้ขาพอร์ต 12 (MISO), 11 (MOSI) และ 10 (SS) ในการติดต่ออุปกรณ์ที่ทำงานผ่านบัสแบบ SPI ประกอบด้วยไอซีหน่วยความจำอีอีพรอมในอนุกรม 93Cxxx, ไอซีแปลงสัญญาณแอนะล็อกเป็นดิจิตอล, ไอซีแปลงสัญญาณดิจิตอลเป็นแอนะล็อก, ไอซีวัดอุณหภูมิ , ไอซีขับ LED ตัวเลข 7 ส่วน เป็นต้น เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีไว้ในตอนต้นของโปรแกรมด้วยคำสั่ง #include <SPI.h>

- Wire บรรจุไลบรารี่และคำสั่งสำหรับติดต่อกับอปุกรณ์ผ่านบัส I2C โดยต้องใช้ขาพอร์ต A4 (SDA) และ A5 (SCL) ในการติดต่ออุปกรณ์ ที่ทำงานผ่านบัส I2C มีมากมายเช่นไอซีหน่วยความจำอีอีพรอมในอนุกรม 24Cxxx , ไอซีแปลงสัญญาณแอนะล็อกเป็นดิจิตอล , ไอซีแปลงสัญญาณดิจิตอลเป็นแอนะล็อก ,ไอซีวัดอุณภูมิ , ไอซีขยายพอร์ตอินพุตเอาต์พตุ, โมดูลวิทยุ FM, ตัวตรวจจับความชื้น, ตัวตรวจจับความดันบรรยากาศ, ตัว ตัวตรวจจับความเร่งแบบ 3 แกน เป็นต้น เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีไว้ในตอนต้นของโปรแกรมด้วยคำสั่ง #include <Wire.h>

สำหรับฮาร์ดแวร์ Arduino Leonardo ซึ่งบอร์ด Arduino ก็เป็นฮาร์ดแวร์แบบหนึ่งที่เข้ากันได้มีไลบรารีพิเศษที่เพิ่มเติมขึ้นมานั่นคือ ไลบรารี USB ในการใช้งานไลบรารี่ของไลบรารี USB นี้ไม่ต้องทำการผนวกไลบรารีเพิ่มเติมแต่อย่างใด ผู้พัฒนาโปรแกรมสามารถเรียกใช้งานไลบรารี่ได้เลย ไลบรารี USB มี 2 ไลบรารีย่อยคือ

- Mouse บรรจุไลบรารี่และคำสั่งเพื่อทำให้ฮาร์ดแวร์ Arduino Leonardo หรือบอร์ด Arduino ทำงานเป็นเมาส์ USB

- Keyboard บรรจุไลบรารี่และคำสั่งเพื่อทำให้ฮาร์ดแวร์ Arduino Leonardo หรือบอร์ด Arduino ทำงานเป็นคีย์บอร์ด USB นอกเหนือไปจากไลบรารีมาตรฐานและไลบรารี USB ที่ทาง Arduino เตรียมมาให้พร้อมใช้งาน

การเรียกใช้งานชุดคำสั่งย่อยต่างๆ เพื่อการพัฒนาโปรแกรมควบคุมสำหรับบอร์ด Arduino ผู้พัฒนาต้องผนวกไฟล์ไลบรารีหลัก Arduino.h โดย #include <Arduino.h> เพื่อประกาศให้ตัวแปลภาษาหรือคอมไพเลอร์รู้จักชุดคำสั่งย่อยต่างๆ ที่กำลังจะถูกเรียกใช้งานจากไฟล์ไลบรารี Arduino.h ไลบรารีย่อย ของไฟล์ไลบรารี Arduino.h ประกอบด้วย

- glcd บรรจุไลบรารี่และคำสั่งสำหรับแสดงผลข้อความ, ตัวเลข และสร้างภาพกราฟิกสีที่จอแสดงผลแบบกราฟิก LCD สีของแผงวงจร GLCD-XT ไลบรารี่นี้มีการการกำหนดขาใช้ งานที่เฉพาะเจาะจง

- sleep บรรจุไลบรารี่และคำสั่งสำหรับการหน่วงเวลา

- in_out บรรจุไลบรารี่และคำสั่งสำหรับอ่านค่าอินพุตดิจิตอลและส่งค่าออกทางขาพอร์ตเอาต์พุตดิจิตอล

- analog บรรจุไลบรารี่และคำสั่งสำหรับอ่านค่าจากอินพุตแอนะล็อกที่ต่อกับตัวตรวจจับ

- sound บรรจุไลบรารี่และคำสั่งสำหรับสร้างเสียงเพื่อขับออกลำโพง ฟังก์ชั่นนี้มีการกำหนดขาใช้งานที่เฉพาะเจาะจง

- motor บรรจุไลบรารี่และคำสั่งสำหรับขับมอเตอร์ไฟตรง 2 ช่อง ต้องทำงานร่วมกับวงจรขับมอเตอร์ที่ใช้ไอซี TB6612 และต้องใช้ไฟเลี้ยงแยกสำหรับมอเตอร์ไฟตรง ไลบรารี่นี้มีการกำหนดขาใช้งานที่เฉพาะเจาะจง
- servoMotor บรรจุไลบรารี่และคำสั่งสำหรับขับเซอร์โวมอเตอร์ต้องทำงานร่วมกับเซอร์โวมอเตอร์และต้องใช้ไฟเลี้ยงแยกสำหรับเซอร์โวมอเตอร์ ไลบรารี่นี้มีการกำหนดขาใช้งานที่เฉพาะเจาะจง
- serial บรรจุไลบรารี่และคำสั่งสำหรับสื่อสารข้อมูลอนุกรมผ่านทางพอร์ต USB และผ่านทางขาพอร์ต TxD และ RxD ของบอร์ด Arduino
- IRremote บรรจุไลบรารี่และคำสั่งสำหรับอ่านรหัสของปุ่มรีโมตคอนโทรลอินฟราเรด ที่ใช้ในเครื่องใช้ไฟฟ้า เมื่อต้องการใช้งานไลบรารีนี้ ต้องผนวกไฟล์ไว้ในตอนต้นของโปรแกรมหลังคำสั่ง #include <Arduino.h> ด้วยคำสั่ง #include <IRremote.h> ในการเรียนรู้เพื่อใช้งานบอร์ด Arduino จะใช้ไฟล์ไลบรารีทั้งแบบมาตรฐานและไฟล์ Arduino.h ร่วมกัน เพื่อช่วยให้การพัฒนาโปรแกรมสำหรับการใช้งานมีประสิทธิภาพสูงสุด และสามารถทำความเข้าใจได้ง่าย ทั้งนี้เพื่อประโยชน์ในการต่อยอดการเรียนรู้ของผู้ใช้งานในวงกว้าง

5.2 ไลบรารี่เกี่ยวกับเวลา

5.2.1 sleep และdelay

เป็นไลบรารี่หน่วงเวลาโดยประมาณภายในโปรแกรมในหน่วยมิลลิวินาที

รูปแบบ

```
void sleep(unsigned int ms)
void delay(unsigned int ms)
```

พารามิเตอร์

ms - กำหนดค่าเวลาที่ต้องการหน่วงในหน่วยมิลลิวินาทีมีค่า 0 ถึง 65,535

ตัวอย่างที่ 5.1

```
sleep(20); // หน่วงเวลาประมาณ 20 มิลลิวินาที
delay(1000); // หน่วงเวลาประมาณ 1 วินาที
```

5.2.2 delay_us

เป็นไลบรารี่หน่วงเวลาโดยประมาณภายในโปรแกรมในหน่วยไมโครวินาที

รูปแบบ

```
void delay_us(unsigned int us)
```

พารามิเตอร์

us - กำหนดค่าเวลาที่ต้องการหน่วงในหน่วยไมโครวินาทีมีค่า 0 ถึง 65,535

ตัวอย่างที่ 5.2

```
delay_us(100); // หน่วงเวลาประมาณ 100 ไมโครวินาที
```

5.3 ไลบรารี่เกี่ยวกับการอ่านค่าดิจิตอล

5.3.1 in

เป็นไลบรารี่อ่านค่าสถานะลอจิกของพอร์ตที่กำหนดเป็นหนึ่งในไลบรารี่ การอ่านและเขียนค่ากับพอร์ตอินพุตเอาต์พุต

รูปแบบ

```
char in(x)
```

พารามิเตอร์

x - กำหนดขาพอร์ตที่ต้องการอ่านค่า

การคืนค่า

เป็น 0 หรื อ 1

ตัวอย่างที่ 5.4

```
char x; // ประกาศตัวแปร x เพื่อเก็บค่าผลลัพธ์จาการอ่านค่าระดับสัญญาณ
x = in(2); // อ่านค่าดิจิตอลจากพอร์ตดิจิตอล 2 มาเก็บไว้ที่ตัวแปร x
```

5.3.2 out

เป็นไลบรารี่กำหนดระดับสัญญาณหรือข้อมูลดิจิตอลไปยังพอร์ตที่กำหนด

รูปแบบ

```
out(char _bit,char _dat)
```

พารามิเตอร์

bit - กำหนดขาพอร์ตที่ต้องการ

ตัวอย่างที่ 5.5

```
out(4,1); // กำหนดให้ขาพอร์ต 4/A6 เป็นเอาต์พุตดิจิตอลและมีค่าเป็น “1”
out(6,0); // กำหนดให้ขาพอร์ต 6/A7 เป็นเอาต์พุตดิจิตอลและมีค่าเป็น “0”
```

5.4 ไลบรารี่เกี่ยวกับการอ่านค่าแอนะล็อก

5.4.1 analog

เป็นไลบรารี่อ่านค่าข้อมูลแอนะล็อก และแปลงเป็นสัญญาณดิจิตอลของไมโครคอนโทรลเลอร์ ที่พอร์ต A0 ถึง A6 ซึ่งใช้ในการเชื่อมต่อกับตัวตรวจจับ ที่ให้ผลการทำงานในรูปแรงดันไฟฟ้าในย่าน 0 ถึ ง +5V

รูปแบบ

unsigned int analog(unsigned char channel)

พารามิเตอร์

channel - กำหนดช่องอินพุตแอนะล็อกที่ต้องการมีค่า 0 ถึง 6 ซึ่งตรงกับขาพอร์ต A0 ถึง A6

การคืนค่า

เป็นข้อมูลที่ได้จากการแปลงสัญญาณของโมดูลแปลงสัญญาณแอนะล็อกเป็นดิจิตอลภายในไมโครคอนโทรลเลอร์ จากช่องอินพุตที่กำหนด โดยข้อมูลมีความละเอียด 10 บิต ดังนั้นค่าที่เป็นไปได้คือ 0 ถึง 1,023

5.4.2 knob

เป็นไลบรารี่อ่านค่าข้อมูลจากตัวต้านทานปรับค่าได้ KNOB มาเสียบเข้ากับบอร์ด Arduino มีการทำงานเหมือนกับคำสั่ง analog() แต่ค่าที่อ่านได้มีค่าในช่วง 80 ถึง 1023 เนื่องจากตัวต้านทานปรับค่าได้ที่เชื่อมต่ออยู่

รูปแบบ

unsigned int knob()

การคืนค่า

ค่าที่อ่านได้จากตัวต้านทานปรับค่าได้ KNOB มีค่าระหว่าง 80 ถึง 1023

ตัวอย่างที่ 5.6

```
int val=0; // กำหนดค่าตัวแปรสำหรับเก็บค่าแอนะล็อก
val=knob(); // อ่านค่าจากตัวต้านทานปรับค่าได้ KNOB เก็บในตัวแปร val
```

5.4.3 sw_ok()

เป็นไลบรารี่ตรวจสอบสถานะสวิตช์ OK บนแผงวงจร จะใช้งานได้เมื่อนำแผงวงจรมาเสียบเข้ากับบอร์ด Arduino โดยให้สถานะ “เป็นจริง” เมื่อมีการกดสวิตช์ และ “เป็นเท็จ” เมื่อไม่มีการกดสวิตช์

รูปแบบ

unsigned char sw_ok()

การคืนค่า

1 (เป็นจริง) เมื่อมีการกดสวิตช์

0 (เป็นเท็จ) เมื่อไม่มีการกดสวิตช์

ตัวอย่างที่ 5.7

```
if(sw_ok())
{
beep(); // เมื่อกดสวิตช์ OK จะมีเสียง “ติ๊ด” ดังออกลำโพง
}
```

5.4.4 sw_ok_press()

เป็นไลบรารี่วนตรวจสอบการกดสวิตช์ OK บนแผงวงจร ต้องรอจนกระทั่งสวิตช์ถูกปล่อยหลังจากการกดสวิตช์ จึงจะผ่านไลบรารี่นี้ไปกระทำคำสั่งอื่นๆ

ตัวอย่างที่ 5.8

```
.....
sw_ok_press(); // รอจนกระทั่งเกิดกดสวิตช์ OK
.....
```

5.5 ไลบรารี่เกี่ยวกับการสื่อสารข้อมูลอนุกรม

เป็นไฟล์ไลบรารีสนับสนุนชุดคำสั่งเกี่ยวกับการรับส่งข้อมูลผ่านโมดูลสื่อสารข้อมูลอนุกรม (UART)

5.5.1 การเชื่อมต่อทางฮาร์ดแวร์

เมื่อต้องการใช้งานช่อง UART ให้ต่อสายจากจุดต่อพอร์ต USB บนบอร์ด Arduino (เป็นจุดต่อเดียวกับที่ใช้ในการอัปโหลด) เข้ากับพอร์ต USB ของคอมพิวเตอร์ เมื่อต้องการใช้งานช่อง UART ต่อสายสัญญาณเข้ากับจุดต่อ RxD (ขาพอร์ต 0) และ TxD (ขาพอร์ต 1)

5.5.2 uart

เป็นไลบรารี่สำหรับส่งข้อมูลสายอักขระออกจากโมดูล UART มีอัตราบอดเริ่มต้นที่ 4,800 บิตวินาที

รูปแบบ

```
void uart(char *p,...)
```

พารามิเตอร์

p - รับรหัสของกลุ่มข้อความที่ต้องการส่งออกจากภาคส่งของโมดูล UART โดยสามารถกำหนดรูปแบบการแทรกสัญลักษณ์พิเศษเพื่อใช้ร่วมในการแสดงผลได้ดังนี้

รหัสบังคับการทำงาน

%c หรือ %C แสดงผลตัวอักษร 1 ตัว

%d หรือ %D แสดงผลตัวเลขฐานสิบช่วงตั้งแต่ -32,768 ถึง +32,767

%l หรือ %L แสดงผลตัวเลขฐานสิบช่วงตั้งแต่ -2,147,483,648 ถึง +2,147,483,647

%f หรือ %F แสดงผลข้อมูลแบบจำนวนจริง (แสดงทศนิยม 3 หลัก)

\r กำหนดให้ข้อความชิดไปทางด้านซ้ายของบรรทัด

\n กำหนดให้ข้อความขึ้นบรรทัดใหม่

5.5.3 uart_set_baud

เป็นไลบรารี่กำหนดอัตราบอดในการสื่อสารของโมดูล UART กับคอมพิวเตอร์

รูปแบบ

void uart_set_baud(unsigned int baud)

พารามิเตอร์

baud - อัตราบอดในการสื่อสารของโมดูล UART กับคอมพิวเตอร์มีค่า 2400 ถึง 115,200

ตัวอย่างที่ 5.9

uart_set_baud(4800); // กำหนดอัตราบอดในการสื่อสารข้อมูลเป็น 4,800 บิตต่อวินาที

5.5.4 uart_available

เป็นไลบรารี่ตรวจสอบการรับข้อมูลเข้ามาของโมดูล UART เมื่อติดต่อกับคอมพิวเตอร์

รูปแบบ

unsigned char uart_available(void)

การคืนค่า

- เป็น “0” เมื่อยังไม่มีข้อมูลเข้ามา
- มากกว่า 0 เมื่อมีข้อมูลเข้ามาโดยมีค่าเท่ากับจำนวนของอักขระที่ได้รับ

ตัวอย่างที่ 5.10

char x =uart_available();

// ตรวจสอบว่ามีข้อมูลเข้ามาทางภาครับของโมดูล UART หรือไม่ถ้า x มีค่ามากกว่า 0 แสดงว่ามีข้อมูลเข้า

5.5.5 uart_getkey

เป็นไลบรารี่อ่านข้อมูลจากบัฟเฟอร์ตัวรับของโมดูล UART

รูปแบบ

char uart_getkey(void)

การคืนค่า

- เป็น “0” เมื่อไม่มีการรับอักขระใดๆ เข้ามายังวงจรภาครับของโมดูล UART
- เป็นค่าของอักขระที่รับได้ในรูปแบบของรหัสแอสกี้

สรุปเนื้อหาสาระสำคัญ

ไลบรารี่โปรแกรมสำหรับ Arduino เป็นไลบรารี่เกี่ยวกับเวลา ไลบรารี่เกี่ยวกับเสียง ไลบรารี่เกี่ยวกับการอ่านค่าดิจิตอล ไลบรารี่เกี่ยวกับการอ่านค่าแอนะล็อก และไลบรารี่เกี่ยวกับการสื่อสารข้อมูลอนุกรม ที่ใช้เป็นประจำ คือไลบรารี่อ่านค่าข้อมูลแอนะล็อกและแปลงเป็นสัญญาณดิจิตอลของไมโครคอนโทรลเลอร์ ที่พอร์ต A0 ถึง A6 ซึ่งใช้ในการเชื่อมต่อกับตัวตรวจจับที่ให้ผลการทำงานในรูปแรงดันไฟฟ้าในย่าน 0 ถึ ง +5V รูปแบบ unsigned int analog(unsigned char channel) พารามิเตอร์ channel - กำหนดช่องอินพุตแอนะล็อกที่ต้องการมีค่า 0 ถึง 6 ซึ่งตรงกับขาพอร์ต A0 ถึง A6 การคืนค่าเป็นข้อมูลที่ได้จากการแปลงสัญญาณของโมดูลแปลงสัญญาณแอนะล็อกเป็นดิจิตอลภายในไมโครคอนโทรลเลอร์ จากช่องอินพุตที่กำหนด โดยข้อมูลมีความละเอียด 10 บิต ดังนั้นค่าที่เป็นไปได้คือ 0 ถึง 1,023

เรื่อง ไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก ใช้เวลา 20 นาที

****

คำชี้แจง แบบฝึกหัดมีทั้งหมด 2 ตอน ประกอบด้วยตอนที่ 1 และตอนที่ 2 (20 คะแนน)

2. แบบฝึกหัดตอนที่ 1 เป็นคำถามแบบถูก-ผิด มีทั้งหมด 20 ข้อ (10 คะแนน)

3. แบบฝึกหัดตอนที่ 2 เป็นคำถามแบบปรนัย มีทั้งหมด 10 ข้อ (10 คะแนน)

คำชี้แจง ให้ผู้เรียนกาเครื่องหมายถูก ✓ ในข้อที่คิดว่าถูก และกาเครื่องหมายผิด ✗ ในข้อที่คิดว่าผิด

คำชี้แจง ให้ผู้เรียนกาเครื่องหมายถูก ✓ ในข้อที่คิดว่าถูก และกาเครื่องหมายผิด ✗ ในข้อที่คิดว่าผิด

_____ 1. %c หรือ %C แสดงผลตัวอักษร 1 ตัว

_____ 2. %d หรื อ %D แสดงผลตัวเลขฐานสิบช่วงตั้งแต่ -32,768 ถึง +32,767

_____ 3. %l หรื อ %L แสดงผลตัวเลขฐานสิบช่วงตั้งแต่ -2,147,483,648 ถึง +2,147,483,647

_____ 4. %f หรื อ %F แสดงผลข้อมูลแบบจำนวนจริง (แสดงทศนิยม 3 หลัก)

_____ 5. \r กำหนดให้ข้อความชิดไปทางด้านซ้ายของบรรทัด

_____ 6. \n กำหนดให้ข้อความขึ้นบรรทัดใหม่

_____7. อัตราบอดในการสื่อสารของโมดูล UART กับคอมพิวเตอร์มีค่า 400 ถึง 115,200

_____8. เมื่อต้องการใช้งานช่อง UART ให้ต่อสายจากจุดต่อพอร์ต USB บนบอร์ด Arduino

_____9. การเชื่อมต่อกับตัวตรวจจับที่ให้ผลการทำงานในรูปแรงดันไฟฟ้าในย่าน 0 ถึ ง +3.3V

-------10. sound(1200,500); // กำเนิดสัญญาณเสียงความถี่ 500Hz นาน 1200 มิลลิวินาที

แบบฝึกหัดตอนที่ 2

คำชี้แจง ให้ผู้เรียนเลือกคำตอบที่ถูกที่สุดแล้วกาเครื่องหมายกากบาท (✕) ให้ครบทุกข้อ

1. LiquidCrystal เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีด้วยคำสั่ง
   - ก. #include < LiquidCrystal.o>
   - ข. #include < LiquidCrystal.h>
   - ค. #include < LiquidCrystal.b>
   - ง. #include < LiquidCrystal.d>
2. SoftwareSerial ในการสื่อสารข้อมูลอนุกรม ไลบรารีจะถูกนำมาใช้งานเมื่อ
   - ก. พอร์ตอนุกรมหลัก (RxD และ TxD) ของ Arduino ไม่ถูกใช้งาน
   - ข. พอร์ตอนุกรมหลัก (RxD และ TxD) ของ Arduino เสีย
   - ค. พอร์ตอนุกรมหลัก (RxD และ TxD) ของ Arduino ถูกใช้งาน
   - ง. พอร์ตอนุกรมหลัก (RxD และ TxD) ของ Arduino ทำงานได้ปกติ
3. ในการติดต่ออุปกรณ์ที่ทำงานผ่านบัสแบบ SPI ประกอบด้วย
   - ก. ไอซีแปลงสัญญาณแอนะล็อกเป็นดิจิตอล
   - ข. ไอซีแปลงสัญญาณดิจิตอลเป็นแอนะล็อก,
   - ค. ไอซีวัดอุณหภูมิ , ไอซีขับ LED ตัวเลข 7 ส่วน
   - ง. ถูกทุกข้อ
4. ไลบรารี่ Serial.println() มีรูปแบบอย่างไร
   - ก. Serial.print(b,FORMAT);
   - ข. Serial.println(b,FORMAT);
   - ค. initialization; condition;
   - ง. Serial.print(b,FORMAT); กับ Serial.println(b,FORMAT);
5. Arduino Leonardo สามารถเรียกใช้งานไลบรารี่ USB โดยมี 2 ไลบรารีย่อยคือ
   - ก. Mouse / Keypad
   - ข. Mouse / Keyboard
   - ค. Keypad / Keyboard
   - ง. USB 1 / USB 2

6. ไลบรารี่เกี่ยวกับเวลา มีอะไรบ้าง

ก. deboun และ delay

ข. sleep และ boun

ค. boun และ delay

ง. sleep และdelay

7. sleep และ delay กำหนดค่าเวลาที่ต้องการหน่วงในหน่วยใด

ก. มิลลิวินาที

ข. ไมโครวินาที

ค. นาโนวินาที

ง. วินาที

8. ไลบรารี่เกี่ยวกับการอ่านค่าแอนะล็อก เป็นไลบรารี่เกี่ยวกับ

ก. อ่านค่าข้อมูลแอนะล็อกแปลงเป็นสัญญาณแอนะล็อก

ข. อ่านค่าข้อมูลดิจิตอลและแปลงเป็นสัญญาณดิจิตอล

ค. อ่านค่าข้อมูลดิจิตอลและแปลงเป็นสัญญาณแอนะล็อก

ง. อ่านค่าข้อมูลแอนะล็อกและแปลงเป็นสัญญาณดิจิตอล

9. UART มีอัตราบอดเริ่มต้นที่

ก. 9600

ข. 2400

ค. 19200

ง. 4800

10. uart_available คือไลบรารี่อะไร

ก. ค่าต่ำสุดของช่วงที่กำหนด

ข. ตรวจสอบการรับข้อมูลเข้ามาของโมดูล UART

ค. ปัดค่าตัวเลขที่น้อยกว่าหรือมากกว่าให้อยู่ในช่วงที่กำหนด

ง. ตัวแปรจำนวนเต็ม 64 บิต แบบไม่คิดเครื่องหมาย

ปฏิบัติการทดลองหน่วยที่ 5
เรื่อง ไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก

****

<u>คำชี้แจง</u> ให้ผู้เรียนทุกคนทำการทดลองตามปฏิบัติการทดลองหน่วยที่ 5 เรื่อง ไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อกใช้เวลา 180 นาที (20 คะแนน)

จุดประสงค์เชิงพฤติกรรม

1. สามารถใช้ไลบรารี่ได้ถูกต้อง
2. สามารถแก้ปัญหาในการทำงานของบอร์ด Arduino Uno R3 ได้
3. สามารถต่อใช้งานและอัพโหลดโปรแกรมให้กับบอร์ด Arduino Uno R3 ได้

อุปกรณ์การทดลอง

1.	เครื่องคอมพิวเตอร์และโปรแกรม Arduino IDE 1.6.9	1	ชุด
2.	USB Cable Arduino Uno R3	1	เส้น
3.	Arduino Uno R3 Board	1	บอร์ด

ข้อควรระวัง

1. ควรระวังไม่วางบอร์ด Arduino Uno R3 หรือชีลต่างๆ บนโต๊ะโลหะหรือที่วางที่เป็นโลหะเพราะอาจเกิดการลัดวงจรของภาคจ่ายไฟได้

2. ไม่ควรต่อสายต่อวงจรในบอร์ด Arduino Uno R3 ทิ้งไว้ ควรถอดสายต่อวงจรออกให้หมด เพราะผลการทดลองอาจเกิดการผิดพลาดไม่เป็นไปตามทฤษฎีได้

3. ไม่ควรถอดสายสายโหลด USB เข้าออกตลอดเวลา เพราะอาจทำให้ภาคจ่ายไฟของบอร์ด Arduino Uno R3 เสียหายได้

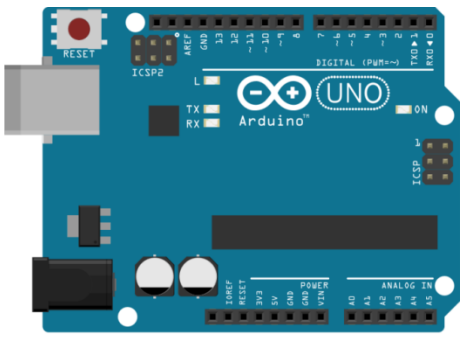
การทดลองที่ 5.1 การลบข้อมูลใน EEPROM

EEPROM Library บรรจุไลบรารี่และคำสั่งสำหรับติดต่อกับหน่วยความจำข้อมูลอีอีพรอม ภายในตัวไมโครคอนโทรลเลอร์ เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีไว้ในตอนต้นของโปรแกรมด้วยคำสั่ง

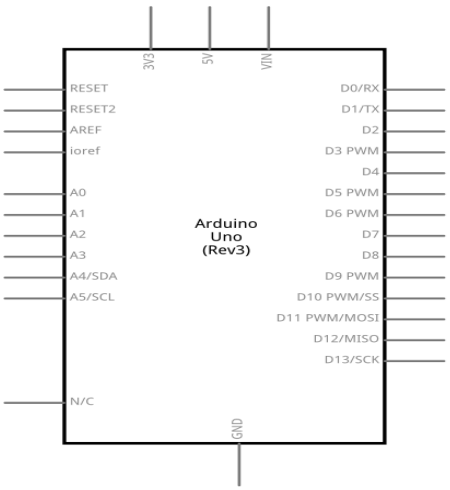
#include <EEPROM.h>

Hardware Required

- Arduino or Genuino Board



Circuit / Schematics



CODE

```
/*
 * EEPROM Clear
 *
 * Sets all of the bytes of the EEPROM to 0.
```

```
 * Please see eeprom_iteration for a more in depth
 * look at how to traverse the EEPROM.
 * This example code is in the public domain.
 */
 #include <EEPROM.h>
void setup() {
 // initialize the LED pin as an output.
  pinMode(13, OUTPUT);
 /***
   Iterate through each byte of the EEPROM storage.
   Larger AVR processors have larger EEPROM sizes, E.g:
   - Arduno Duemilanove: 512b EEPROM storage.
   - Arduino Uno:        1kb EEPROM storage.
   - Arduino Mega:       4kb EEPROM storage.
   Rather than hard-coding the length, you should use the pre-provided length function.
   This will make your code portable to all AVR processors.
 ***/
  for (int i = 0 ; i < EEPROM.length() ; i++) {
    EEPROM.write(i, 0);
  }
 // turn the LED on when we're done
  digitalWrite(13, HIGH);
}
void loop() {
 /** Empty loop. **/
}
```

ผลการทดลอง

.....

.....

.....

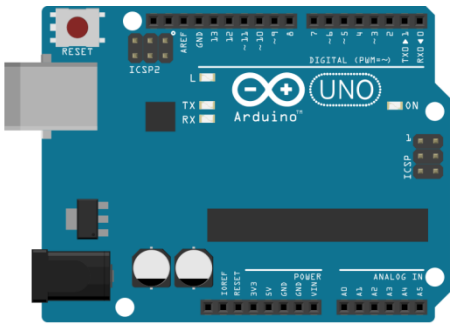
.....

.....

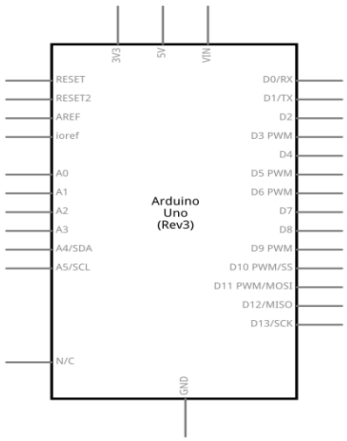
การทดลองที่ 5.2 การอ่านข้อมูลใน EEPROM

Hardware Required

- Arduino or Genuino Board



Circuit / Schematics



Code

```
/*
 * EEPROM Read
 * Reads the value of each byte of the EEPROM and prints it
 * to the computer.
 * This example code is in the public domain.
 */


#include <EEPROM.h>
```

```
// start reading from the first byte (address 0) of the EEPROM
int address = 0;
byte value;
void setup() {
  // initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
}
void loop() {
  // read a byte from the current address of the EEPROM
  value = EEPROM.read(address);
  Serial.print(address);
  Serial.print("\t");
  Serial.print(value, DEC);
  Serial.println();
  /***
    Advance to the next address, when at the end restart at the beginning.
    Larger AVR processors have larger EEPROM sizes, E.g:
    - Arduno Duemilanove: 512b EEPROM storage.
    - Arduino Uno:        1kb EEPROM storage.
    - Arduino Mega:       4kb EEPROM storage.
    Rather than hard-coding the length, you should use the pre-provided length function.
    This will make your code portable to all AVR processors.
  ***/
  address = address + 1;
  if (address == EEPROM.length()) {
    address = 0;
  }

  /***
```

```
   As the EEPROM sizes are powers of two, wrapping (preventing overflow) of an
   EEPROM address is also doable by a bitwise and of the length - 1.
   ++address &= EEPROM.length() - 1;
  ***/
  delay(500);
}
```

ผลการทดลอง

.....

.....

.....

.....

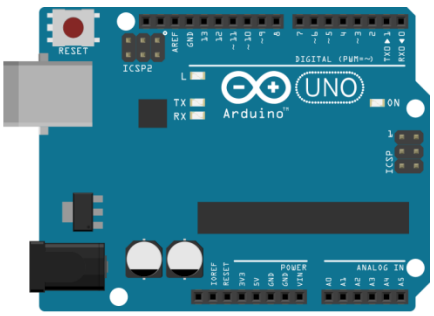
.....

.....

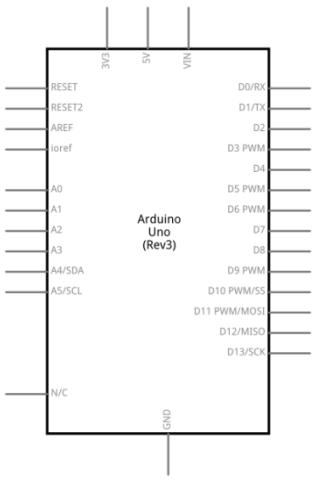
การทดลองที่ 5.3 การเขียนข้อมูลลง EEPROM

Hardware Required

- Arduino or Genuino Board



Circuit / Schematics



Code

```
/*
 * EEPROM Write
 * Stores values read from analog input 0 into the EEPROM.
 * These values will stay in the EEPROM when the board is
 * turned off and may be retrieved later by another sketch.
 */
#include <EEPROM.h>
/** the current address in the EEPROM (i.e. which byte we're going to write to next) **/
int addr = 0;
void setup() {
  /** Empty setup. **/
}
void loop() {
  /***
    Need to divide by 4 because analog inputs range from
    0 to 1023 and each byte of the EEPROM can only hold a
    value from 0 to 255.
  ***/
```

```
  int val = analogRead(0) / 4;

  /***
    Write the value to the appropriate byte of the EEPROM.
    these values will remain there when the board is
    turned off.
  ***/
  EEPROM.write(addr, val);
  /***
    Advance to the next address, when at the end restart at the beginning.
    Larger AVR processors have larger EEPROM sizes, E.g:
    - Arduno Duemilanove: 512b EEPROM storage.
    - Arduino Uno:        1kb EEPROM storage.
    - Arduino Mega:       4kb EEPROM storage.
    Rather than hard-coding the length, you should use the pre-provided length function.
    This will make your code portable to all AVR processors.
  ***/
  addr = addr + 1;
  if (addr == EEPROM.length()) {
    addr = 0;
  }
  /***
    As the EEPROM sizes are powers of two, wrapping (preventing overflow) of an
    EEPROM address is also doable by a bitwise and of the length - 1.
    ++addr &= EEPROM.length() - 1;
  ***/
  delay(100);
}
```

ผลการทดลอง

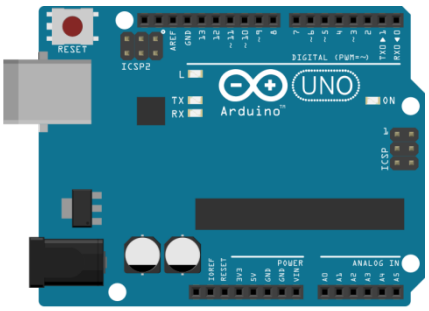
.....

.....

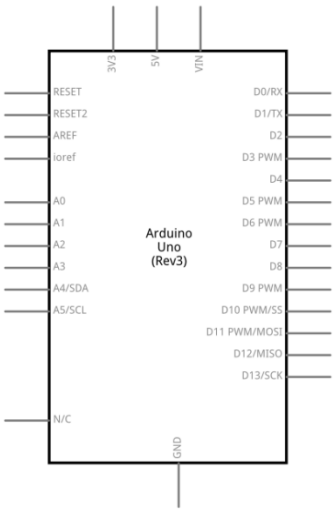
การทดลองที่ 5.4 Software Serial

Hardware Required

- Arduino or Genuino Board



Circuit / Schematics



Code

```
/*
 Software serial multple serial test
Receives from the hardware serial, sends to software serial.
Receives from software serial, sends to hardware serial.
The circuit:
* RX is digital pin 10 (connect to TX of other device)
* TX is digital pin 11 (connect to RX of other device)
Note:
```

```
 Not all pins on the Mega and Mega 2560 support change interrupts,
 so only the following can be used for RX:
 10, 11, 12, 13, 50, 51, 52, 53, 62, 63, 64, 65, 66, 67, 68, 69
 Not all pins on the Leonardo support change interrupts,
 so only the following can be used for RX:
 8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).
 created back in the mists of time
 modified 25 May 2012
 by Tom Igoe
 based on Mikal Hart's example
 This example code is in the public domain.
 */
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(57600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }


  Serial.println("Goodnight moon!");
  // set the data rate for the SoftwareSerial port
  mySerial.begin(4800);
  mySerial.println("Hello, world?");
}

void loop() { // run over and over
  if (mySerial.available()) {
    Serial.write(mySerial.read());
```

```
  }
  if (Serial.available()) {
    mySerial.write(Serial.read());
  }
}
```

ผลการทดลอง

.....

.....

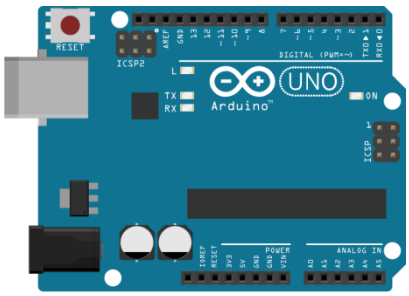
.....

.....

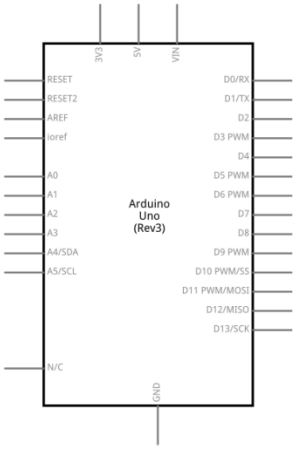
การทดลองที่ 5.5 Two Port Receive

Hardware Required

- Arduino or Genuino Board



Circuit / Schematics



Code

```
/*
 Software serial multple serial test
Receives from the two software serial ports,
sends to the hardware serial port.
In order to listen on a software port, you call port.listen().
When using two software serial ports, you have to switch ports
by listen()ing on each one in turn. Pick a logical time to switch
ports, like the end of an expected transmission, or when the
buffer is empty. This example switches ports when there is nothing
more to read from a port
The circuit:
Two devices which communicate serially are needed.
* First serial device's TX attached to digital pin 10(RX), RX to pin 11(TX)
* Second serial device's TX attached to digital pin 8(RX), RX to pin 9(TX)
Note:
Not all pins on the Mega and Mega 2560 support change interrupts,
so only the following can be used for RX:
10, 11, 12, 13, 50, 51, 52, 53, 62, 63, 64, 65, 66, 67, 68, 69
Not all pins on the Leonardo support change interrupts,
so only the following can be used for RX:
8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).
created 18 Apr. 2011
modified 19 March 2016
by Tom Igoe
based on Mikal Hart's twoPortRXExample

This example code is in the public domain.
*/
#include <SoftwareSerial.h>
// software serial #1: RX = digital pin 10, TX = digital pin 11
SoftwareSerial portOne(10, 11);
```

```
// software serial #2: RX = digital pin 8, TX = digital pin 9
// on the Mega, use other pins instead, since 8 and 9 don't work on the Mega
SoftwareSerial portTwo(8, 9);
void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  // Start each software serial port
  portOne.begin(9600);
  portTwo.begin(9600);
}
void loop() {
  // By default, the last intialized port is listening.
  // when you want to listen on a port, explicitly select it:
  portOne.listen();
  Serial.println("Data from port one:");
  // while there is data coming in, read it
  // and send to the hardware serial port:
  while (portOne.available() > 0) {
    char inByte = portOne.read();
    Serial.write(inByte);
  }
  // blank line to separate data from the two ports:
  Serial.println();
  // Now listen on the second port
  portTwo.listen();
  // while there is data coming in, read it
  // and send to the hardware serial port:
  Serial.println("Data from port two:");
  while (portTwo.available() > 0) {
```

```
    char inByte = portTwo.read();
    Serial.write(inByte);
  }
  // blank line to separate data from the two ports:
  Serial.println();
}
```

ผลการทดลอง

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ปัญหาอุปสรรคหรือข้อเสนอแนะ

.....

.....

ตารางการประเมินผลคะแนนภาคปฏิบัติ

หัวข้อการพิจารณาภาคปฏิบัติ	ระดับคะแนน
การทดลองที่ 5.1 EEPROM Clear	4 คะแนน
การทดลองที่ 5.2 EEPROM Read	4 คะแนน
การทดลองที่ 5.3 EEPROM Write	4 คะแนน
การทดลองที่ 5.4 Software Serial	4 คะแนน
การทดลองที่ 5.5 Two Port Receive	4 คะแนน
รวมคะแนนภาคปฏิบัติ	.....คะแนน

แบบทดสอบหลังเรียน หน่วยที่ 5

เรื่อง ไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก

เรื่อง ไลบรารี่โปรแกรมสำหรับ Arduino และการอ่านค่าแอนะล็อก **ใช้เวลา** 20 **นาที**

วิชา ไมโครคอนโทรลเลอร์เบื้องต้น **รหัสวิชา** (2127-2107)

ระดับชั้น ประกาศนียบัตรวิชาชีพ (ปวช.) **สาขาวิชา** เมคคาทรอนิกส์

****

คำชี้แจง 1. แบบทดสอบมีทั้งหมด 10 ข้อ (10 คะแนน)

2. ให้ผู้เรียนเลือกคำตอบที่ถูกที่สุดแล้วกาเครื่องหมายกากบาท (✕) ลงในกระดาษคำตอบ

1. หน่วยความจำข้อมูลอีอีพรอม ภายในตัวไมโครฯเมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีด้วยคำสั่ง
   - ก. #include<EEPROM.o>
   - ข. #include< EEPROM.d>
   - ค. #include<EEPROM.h>
   - ง. #include<EEPROM. b>
2. servo เมื่อต้องการใช้งานต้องผนวกไฟล์ไลบรารีด้วยคำสั่ง
   - ก. #include<servo.h>
   - ข. #include<servo.d>
   - ค. #include <servo.b>
   - ง. #include<servo.o>
3. Arduino UNO บรรจุฟังก์ชั่น Wire สำหรับติดต่อกับอุปกรณ์ผ่านบัส
   - ก. GSM
   - ข. SPI
   - ค. I2C
   - ง. UART
4. Arduino UNO บรรจุฟังก์ชั่นและคำสั่งสำหรับติดต่อกับ SPI โดยต้องใช้ขาพอร์ต
   - ก. 10 (MISO), 11 (MOSI) และ 12 (SS)
   - ข. 12 (MISO), 11 (MOSI) และ 10 (SS)
   - ค. 11 (MISO), 12 (MOSI) และ 13 (SS)
   - ง. 13 (MISO), 12 (MOSI) และ 11 (SS)

5. บัส I2C Arduino UNO ต้องใช้ขาพอร์ต
   ก. A1 (SDA) และ A2 (SCL)
   ข. A2 (SDA) และ A3 (SCL)
   ค. A3 (SDA) และ A4 (SCL)
   ง. A4 (SDA) และ A5 (SCL)
6. ฟังก์ชั่นเกี่ยวกับพอร์ตอินพุตเอาต์พุต ฟังก์ชั่น in เป็นฟังก์ชั่นอะไร
   ก. กำหนดขาพอร์ตที่ต้องการอ่านค่า
   ข. อ่านค่าสถานะลอจิกของพอร์ตที่กำหนด
   ค. อ่านค่าดิจิตอลจากพอร์ตดิจิตอล
   ง. อ่านค่าดิจิตอลจากพอร์ตอนุกรม
7. ฟังก์ชั่น out เป็นฟังก์ชั่นอะไร
   ก. กำหนดข้อมูลดิจิตอลไปยังพอร์ตที่กำหนด
   ข. กำหนดขาพอร์ตที่ต้องการอ่านค่า
   ค. อ่านค่าดิจิตอลจากพอร์ตดิจิตอล 1
   ง. อ่านค่าดิจิตอลจากพอร์ตดิจิตอล 2
8. ฟังก์ชั่นการอ่านค่าแอนะล็อกของ Arduino UNO ใช้พอร์ตอย่างไร
   ก. กำหนดช่องอินพุตแอนะล็อกที่ต้องการตรงกับขาพอร์ต A3 ถึง A6
   ข. กำหนดช่องอินพุตแอนะล็อกที่ต้องการตรงกับขาพอร์ต A2 ถึง A6
   ค. กำหนดช่องอินพุตแอนะล็อกที่ต้องการตรงกับขาพอร์ต A1 ถึง A6
   ง. กำหนดช่องอินพุตแอนะล็อกที่ต้องการตรงกับขาพอร์ต A0 ถึง A6
9. ฟังก์ชั่นเกี่ยวกับการสื่อสารข้อมูลอนุกรม เมื่อต้องการใช้งานช่อง UART ต่อสายสัญญาณอย่างไร
   ก. RxD (ขาพอร์ต 0) และ TxD (ขาพอร์ต 1)
   ข. RxD (ขาพอร์ต 1) และ TxD (ขาพอร์ต 0)
   ค. RxD (ขาพอร์ต A0) และ TxD (ขาพอร์ต A1)
   ง. RxD (ขาพอร์ต A1) และ TxD (ขาพอร์ต A0)
10. uart_available คือฟังก์ชั่นอะไร
   ก. ค่าต่ำสุดของช่วงที่กำหนดของโมดูล UART
   ข. ตรวจสอบการรับข้อมูลเข้ามาของโมดูล UART
   ค. ปัดค่าตัวเลขให้อยู่ในช่วงที่กำหนดของโมดูล UART
   ง. ตัวแปรจำนวนเต็ม 64 บิต แบบไม่คิดเครื่องหมายของโมดูล UART