หน่วยที่ 3 โครงสร้างโปรแกรมของ ARDUINO

สาระสำคัญ

ในการเขียนโปรแกรมสำหรับบอร์ด Arduino จะต้องเขียนโปรแกรมโดยใช้ภาษาของ Arduino (Arduino Programming Language) ซึ่งตัวภาษาของ Arduino ก็นำเอาโอเพ่นซอร์สโปรเจ็กต์ชื่อ Wiring มาพัฒนาต่อ ภาษาของ Arduino แบ่งได้เป็น 2 ส่วนหลักคือโครงสร้างภาษา (Structure) ตัวแปร ค่าคงที่และฟังก์ชั่น (Function) ภาษาของ Arduino จะอ้างอิงตามภาษา C/C++ จึงอาจกล่าวได้ว่าการเขียนโปรแกรมสำหรับ Arduino (ซึ่งก็รวมถึงบอร์ด Arduino) ก็คือการเขียนโปรแกรมภาษา C โดยเรียกใช้ฟังก์ชั่นและไลบรารีที่ทาง Arduino ได้เตรียมไว้ให้แล้ว

เนื้อหาสาระการเรียนรู้

3.1 ส่วนของฟังก์ชั่น setup()
3.2 ส่วนของฟังก์ชั่น loop()
3.3 ส่วนของตัวกระทำทางคณิตศาสตร์
3.4 ส่วนของตัวกระทำเปรียบเทียบ
3.5 ส่วนของตัวกระทำระดับบิต
3.6 ไวยากรณ์ภาษา C / C++ ของ Arduino
3.7 ตัวแปร
3.8 ขอบเขตของตัวแปร
3.9 การกำหนดค่าคงที่เลขจำนวนเต็มเป็นเลขฐานต่างๆ ของ Arduino
3.10 ค่าคงที่ (constants)
3.11 ตัวกระทำอื่นๆ ที่เกี่ยวข้องกับตัวแปร
3.12 คำสงวนของ Arduino

จุดประสงค์การเรียนรู้

จุดประสงค์ทั่วไป

1. เพื่อให้มีความรู้ความเข้าใจเกี่ยวกับลักษณะโครงสร้างโปรแกรมของ Arduino
2. เพื่อให้สามารถนำความรู้ไปประยุกต์ใช้ในการเขียนโปรแกรมกำหนดการทำงานของ Arduino
3. เพื่อให้ตระหนักถึงความสำคัญของลักษณะโครงสร้างโปรแกรมของ Arduino

จุดประสงค์เชิงพฤติกรรม

1. บอกหน้าที่การทำงานส่วนของฟังก์ชั่นได้
2. บอกหน้าที่การทำงานส่วนของฟังก์ชั่น setup() ได้
3. บอกหน้าที่การทำงานส่วนของฟังก์ชั่น loop() ได้
4. บอกหน้าที่การทำงานของตัวกระทำทางคณิตศาสตร์ได้
5. บอกหน้าที่การทำงานส่วนของตัวกระทำเปรียบเทียบได้
6. ใช้งานส่วนของตัวกระทำระดับบิตได้
7. ใช้งานไวยากรณ์ภาษา C / C++ ของ Arduinoได้
8. ใช้งานตัวแปรได้

แบบทดสอบก่อนเรียน หน่วยที่ 3
เรื่อง โครงสร้างโปรแกรมของ Arduino

เรื่อง โครงสร้างโปรแกรมของ Arduino **ใช้เวลา 20 นาที**

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

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

****

<u>คำชี้แจง</u> **1. แบบทดสอบมีทั้งหมด 10 ข้อ (10 คะแนน)**

2. ให้ผู้เรียนเลือกคำตอบที่ถูกที่สุดแล้วกาเครื่องหมายกากบาท (✕) ลงในกระดาษคำตอบ

1. ภาษาของ Arduino แบ่งได้เป็น 2 ส่วนหลักคือ
   - ก. โครงสร้างภาษา / ฟังก์ชั่น
   - ข. ตัวแปร / ไวยกรณ์
   - ค. ฟังก์ชั่น / ตัวกระทำ
   - ง. ตัวกระทำ / ตัวเปรียบเทียบ
2. โครงสร้างโปรแกรมของ Arduino แบ่งได้เป็นสองส่วนคือ
   - ก. void up() และ void loop()
   - ข. void set() และ void setup()
   - ค. void setup () และ void looping()
   - ง. void setup() และ void loop()
3. ฟังก์ชั่น loop() ซึ่งมีการทำงานแบบใด
   - ก. ทำงานแบบวนรอบต่อเนื่องตลอดเวลา
   - ข. ทำงานแบบวนรอบ 5 ครั้ง
   - ค. ทำงานแบบวนรอบ 10 ครั้ง
   - ง. ทำงานวนแบบวนรอบ 15 ครั้ง
4. คำสั่ง if...else ใช้เพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมอย่างไร
   - ก. ถ้าเงื่อนไขเป็นจริงให้ทำอะไร ถ้าเป็นเท็จให้ทำอะไร
   - ข. ไม่สามารถทำอะไรได้อีกแล้ว
   - ค. ถ้าเงื่อนไขเป็นจริงให้ทำอะไร
   - ง. ถ้าเงื่อนไขเป็นเท็จให้ทำอะไร

5. ตัวกระทำทางคณิตศาสตร์ ประกอบด้วย

ก. + (บวก),- (ลบ), * (คู ณ), / (หาร) และ % (หารเอาเศษ)

ข. + (บวก),- (ลบ), * (คู ณ), / (หาร) และ % (หารไม่เอาเศษ)

ค. * (คูณ), / (หาร) และ % (หารเอาเศษ)

ง. + (บวก),- (ลบ), และ % (หารเอาเศษ)

6. ตัวกระทำระดับบิต AND ใช้สัญญลักษณ์ใด

ก. &&

ข. AND

ค. และ

ง. &

7. ตัวกระทำระดับบิต OR ใช้สัญญลักษณ์ใด

ก. I

ข. II

ค. หรือ

ง. OR

8. คำสั่งระดับบิต Exclusive OR ใช้สัญญลักษณ์ใด

ก. ^

ข. $

ค. f

ง. ∞

9. ไวยากรณ์ภาษา C เซมิโคลอน (semicolon ;) ใช้ทำหน้าที่ใด

ก. ใช้เขียนแจ้งว่าจบคำสั่ง

ข. ใช้เขียนแจ้งว่าเริ่มคำสั่ง

ค. ใช้เขียนแจ้งว่าลบคำสั่ง

ง. ใช้เขียนแจ้งว่าเพิ่มคำสั่ง

10. ตัวแปรประเภทเลขทศนิยมคือ

ก. long ใช้พื้นที่หน่วยความจำ 2 ไบต์

ข. float ใช้พื้นที่หน่วยความจำ 2 ไบต์

ค. long ใช้พื้นที่หน่วยความจำ 4 ไบต์

ง. float ใช้พื้นที่หน่วยความจำ 4 ไบต์

หน่วยที่ 3
โครงสร้างโปรแกรมของ Arduino

ในการเขียนโปรแกรมสำหรับบอร์ด Arduino จะต้องเขียนโปรแกรมโดยใช้ภาษาของ Arduino (Arduino Programming Language) ซึ่งตัวภาษาของ Arduino ก็นำเอาโอเพ่นซอร์สโปรเจ็กต์ชื่อ Wiring มาพัฒนาต่อ ภาษาของ Arduino แบ่งได้เป็น 2 ส่วนหลักคือ

1. โครงสร้างภาษา (Structure) ตัวแปรและค่าคงที่
2. ฟังก์ชั่น (Function)

ภาษาของ Arduino จะอ้างอิงตามภาษา C/C++ จึงอาจกล่าวได้ว่าการเขียนโปรแกรมสำหรับ Arduino (ซึ่งก็รวมถึงบอร์ด Arduino) ก็คือการเขียนโปรแกรมภาษา C โดยเรียกใช้ฟังก์ชั่นและไลบรารีที่ทาง Arduino ได้เตรียมไว้ให้แล้ว ซึ่งสะดวกและทำให้ผู้ที่ไม่มีความรู้ด้านไมโครคอนโทรลเลอร์อย่างลึกซึ้งสามารถเขียนโปรแกรมสั่งงานได้ ในบทนี้จะอธิบายถึงโครงสร้างโปรแกรมของ Arduino แบ่งได้เป็นสองส่วนคือ void setup() และ void loop()

โดยฟังก์ชั่น setup() เมื่อโปรแกรมทำงานจะทำคำสั่งของฟังก์ชั่นนี้เพียงครั้งเดียว ใช้ในการกำหนดค่าเริ่มต้นของการทำงานส่วนฟังก์ชั่น loop() เป็นส่วนทำงานโปรแกรมจะทำคำสั่งในฟังกชั่นนั่นต่อเนื่องกันตลอดเวลา โดยปกติใช้กำหนดโหมดการทำงานของขาต่างๆ กำหนดการสื่อสารแบบอนุกรม ฯลฯ

ส่วนของ loop() เป็นโค้ดโปรแกรมที่ทำงาน เช่น อ่านค่าอินพุต ประมวลผล สั่งงานเอาต์พุต ฯลฯ โดยส่วนกำหนดค่าเริ่มต้น เช่นตัวแปรจะต้องเขียนที่ส่วนหัวของโปรแกรมก่อนถึงตัวฟังก์ชั่น นอกจากนั้นยังต้องคำนึงถึงตัวพิมพ์ เล็ก-ใหญ่ ของตัวแปรและชื่อฟังกชั่นั้นให้ถูกต้อง

3.1 ส่วนของฟังก์ชั่น setup()

ฟังก์ชั่นนี้จะเขียนที่ส่วนต้นของโปรแกรม ทำงานเมื่อโปรแกรมเริ่มต้นเพียงครั้งเดียว ใช้เพื่อกำหนดค่าของตัวแปรโหมดการทำงานของขาต่างๆ เริ่มต้นเรียกใช้ไลบรารี ฯลฯ

ตัวอย่างที่ 3.1

```
int buttonPin = 13;
void setup()
{
        Serial.begin(9600);
        pinMode(buttonPin, INPUT);
}
```

```
void loop()
{
        if (digitalRead(buttonPin) == HIGH)
        Serial.println('H');
        else
        Serial.println('L');
        delay(1000);
}
```

ในขณะที่โปรแกรมภาษา C มาตรฐานที่เขียนบน AVR GCC (เป็นโปรแกรมภาษา C ที่ใช้ C คอมไพเลอร์แบบ GCC สำหรับไมโครคอนโทรลเลอร์ AVR) จะเขียนได้ ดังนี้

```
int main(void)
{
        init();
        setup();
        for (;;)
        loop();
        return ;
}
```

3.2 ส่วนของฟังก์ชั่น loop()

หลังจากที่เขียนฟังก์ชั่น setup() ที่กำหนดค่าเริ่มต้นของโปรแกรมแล้ว ส่วนถัดมาคือฟังก์ชั่น loop() ซึ่งมีการทำงานตรงตามชื่อ คือจะทำงานตามฟังก์ชั่นวนต่อเนื่องตลอดเวลา ภายในฟังก์ชั่นจะมีโปรแกรมของผู้ใช้เพื่อรับค่าจากพอร์ต ประมวลผลแล้วสั่งเอาต์พุตออกขาต่างๆ เพื่อควบคุมการทำงานของบอร์ด

ตัวอย่างที่ 3.2

```
        int buttonPin = 13;
// setup initializes eerial and the button pin
        void setup()
                {
                Serial.begin(9600);
                pinMode(buttonPin, INPUT);
                }
// loop checks the button pin each time,
```

```
// and will send serial if it is pressed
        void loop()
                {
                if (digitalRead(buttonPin) == HIGH)
                Serial.println('H');
                else
                Serial.println('L');
                delay(1000);
                }
```

โปรแกรมทำงานวนในฟังก์ชั่น loop() ตลอดเวลา หลังจากทำงานในฟังก์ชั่น setup() จึงสรุปได้ว่าฟังก์ชั่น setup() คือส่วนต้นของโปรแกรมที่ใช้ในการประกาศ หรือตั้งค่าการทำงานในตอนเริ่มต้นทำงาน ในขณะที่ฟังก์ชั่น loop() เป็นเสมือนส่วนของโปรแกรมหลักที่ต้องวนทำงานอย่างต่อเนื่องตลอดเวลา อย่างไรก็ตามในบางโปรแกรมอาจมีเฉพาะส่วนของฟังก์ชั่น setup() และไม่มีฟังก์ชั่น loop() ก็ได้ นั่นแสดงว่าโปรแกรมนั้นๆต้องการตั้งค่าการทำงาน หรือกำหนดให้มีการทำงานเพียงครั้งหรือรอบเดียว แล้วจบการทำงานทันที

3.2.1 คำสั่ง if

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรม เช่นถ้าอินพุตมีค่ามากกว่าค่าที่กำหนดไว้ จะให้ทำอะไรโดยมีรูปแบบการเขียนดังนี้

```
        if (somevariable > 50)
        {
// do something Here
        }
```

ตัวโปรแกรมจะทดสอบว่าถ้าตัวแปร someVariable มีค่ามากกว่า 50 หรือไม่ ถ้าใช่ให้ทำอะไร ถ้าไม่ใช่ให้ข้ามการทำงานส่วนนี้ การทำงานของคำสั่งนี้จะทดสอบเงื่อนไขที่เขียนในเครื่องหมายวงเล็บ ถ้าเงื่อนไขเป็นจริงทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นเท็จข้ามการทำงานส่วนนี้ไป ส่วนของการทดสอบเงื่อนไขที่เขียนอยู่ภายในวงเล็บ จะต้องใช้ตัวกระทำเปรียบเทียบต่างๆ ดังนี้

```
x == y (x เท่ากับ y)
x != y (x ไม่เท่ากับ y)
x < y (x น้อยกว่า y)
x > y (x มากกว่า y)
x <= y (x น้อยกว่าหรือเท่ากับ y)
x >= y (x มากกว่าหรือเท่ากับ y)
```

เทคนิคสำหรับการเขียนโปรแกรมในการเปรียบเทียบตัวแปรให้ ใช้ตัวกระทำ == (เช่น if (x==10)) ห้าม

เขียนผิดเป็น = (เช่น if (x=10)) คำสั่งที่เขียนผิดในแบบที่สองนี้ ทำให้ผลการทดสอบเป็นจริงเสมอ และเมื่อผ่านคำสั่งแล้ว x มีค่าเท่ากับ 10 ทำให้การทำงานของโปรแกรมผิดเพี้ยนไปไม่เป็นตามที่กำหนดไว้ เราสามารถใช้คำสั่ง if ในคำสั่งควบคุมการแยกเส้นทางของโปรแกรมโดยใช้คำสั่ง if....else

3.2.2 คำสั่ง if...else

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมได้มากกว่าคำสั่ง if ธรรมดา โดยสามารถกำหนดได้ว่าถ้าเงื่อนไขเป็นจริงให้ทำอะไร ถ้าเป็นเท็จให้ทำอะไร เช่นถ้าค่าอินพุตแอนะล็อกที่อ่านได้น้อยกว่า 500 ให้ทำอะไร ถ้าค่ามากกว่าหรือเท่ากับ 500 ให้ทำอีกอย่าง จะเขียนคำสั่งได้ดังนี้

ตัวอย่างที่ 3.3

```
if (pinFiveInput < 500)
{
// do thing A
}
else
{
// do thing B
}
```

หลังคำสั่ง else สามารถตามด้วยคำสั่ง if สำหรับการทดสอบอื่นๆ ทำให้รูปแบบคำสั่งกลายเป็น if....else...if เป็นการทดสอบเงื่อนไขต่างๆ เมื่อเป็นจริงให้ทำตามที่ต้องการดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.4

```
if (pinFiveInput < 500)
{
// do Thing A
}
else if (pinFiveInput >= 1000)
{
// do Thing B
}
else
{
// do Thing C
}
```

หลังคำสั่ง else สามารถตามด้วยคำสั่ง if ได้ไม่จำกัดจำนวน (สามารถใช้คำสั่ง switch case แทนคำสั่ง

if...else...if สำหรับการทดสอบเงื่อนไขจำนวนมากๆ ได้) เมื่อใช้คำสั่ง if...else แล้วต้องกำหนดด้วยว่าถ้าทดสอบไม่ตรงกับเงื่อนไขใดๆเลย ให้ทำอะไรโดยให้กำหนดที่คำสั่ง else ตัวสุดท้าย

3.2.3 คำสั่ง for()

คำสั่งนี้ใช้เพื่อสั่งให้คำสั่งที่อยู่ภายในวงเล็บปีกกาหลัง for มีการทำงานซ้ำกันตามจำนวนรอบที่ต้องการ คำสั่งนี้มีประโยชน์มากสำหรับการทำงานใดๆที่ต้องทำซ้ำกัน และทราบจำานวนรอบของการทำงานที่แน่นอน ใช้คู่กับตัวแปรอะเรย์ในการเก็บสะสมค่าที่อ่านค่าได้จากขาอินพุตแอนะล็อกหลายๆขา ที่มีหมายเลขขาต่อเนื่องกัน รูปแบบของคำสั่ง for() แบ่งได้ 3 ส่วนดังนี้

```
for (initialization; condition; increment)
{
//statement(s);
}
```

เริ่มต้นด้วย initialization ใช้กำหนดค่าเริ่มต้นของตัวแปรควบคุมการวนรอบในการทำงานแต่ละรอบ จะทดสอบ condition ถ้าเงื่อนไขเป็นจริงทำตามคำสั่งในวงเล็บปีกกา แล้วมาเพิ่มหรือลดค่าตัวแปรตามที่สั่งใน increment แล้วทดสอบเงื่อนไขอีก ทำซ้ำจนกว่าเงื่อนไขเป็นเท็จ

ตัวอย่างที่ 3.5

```
for (int i=1; i <= 8; i++)
{
// statement using the value i;
}
```

คำสั่ง for ของภาษา C ยืดหยุ่นกว่าคำสั่ง for ของภาษาคอมพิวเตอร์อื่นๆ มันสามารถละเว้นบางส่วนหรือทั้งสามส่วนของคำสั่ง for ได้ อย่างไรก็ตามยังต้องมีเซมิโคลอน นอกจากนั้นยังนำคำสั่งภาษา C++ ที่มีตัวแปรที่ไม่เกี่ยวข้องมาเขียนในส่วนของ initialization condition และ increment ของคำสั่ง for ได้

3.2.4 คำสั่ง switch-case

ใช้ทดสอบเงื่อนไขเพื่อกำหนดการทำงานของโปรแกรม ถ้าตัวแปรที่ทดสอบตรงกับเงื่อนไขใดก็ให้ทำงานตามที่กำหนดไว้พารามิเตอร์

var ตัวแปรที่ต้องการทดสอบว่าตรงกับเงื่อนไขใด

default ถ้าไม่ตรงกับเงื่อนไขใดๆเลย ให้ทำคำสั่งต่อท้ายนี้

break เป็นส่วนสำคัญมากใช้เขียนต่อท้าย case ต่างๆ เมื่อพบเงื่อนไขนั้นแล้วทำตามคำสั่งต่างๆแล้ว ให้หยุดการทำงานของคำสั่ง

switch-case ถ้าลืมเขียน break เมื่อพบเงื่อนไขทำตามเงื่อนไขแล้วโปรแกรมจะทำงานตามเงื่อนไขต่อไปเรื่อยๆ จนกว่าจะพบคำสั่ง break

ตัวอย่างที่ 3.6

```
switch (var)
{
case 1:
//do something when var == 1
break;
case 2:
//do something when var == 2
break;
default:
// if nothing else matches, do the default
}
```

3.2.5 คำสั่ง while

เป็นคำสั่งวนรอบโดยจะทำคำสั่งที่เขียนในวงเล็บปีกกาอย่างต่อเนื่อง จนกว่าเงื่อนที่เขียนในวงเล็บของคำสั่ง while() จะเป็นเท็จ คำสั่งที่ให้ทำซ้ำจะต้องมีการเปลี่ยนแปลงค่าตัวแปรที่ใช้ทดสอบ โดยมีการเพิ่มค่าตัวแปรหรือมีเงื่อนไขภายนอก เช่นอ่านค่าจากตัวตรวจจับได้เรียบร้อยแล้วให้หยุดการอ่านค่า มิฉะนั้นเงื่อนไขในวงเล็บของ while() เป็นจริงตลอดเวลา ทำให้คำสั่ง while ทำงานวนรอบไปเรื่อยๆ ไม่รู้จบ รูปแบบคำสั่ ง

```
while(expression)
{
// statement(s)
}
```

พารามิเตอร์ expression เป็นคำสั่งทดสอบเงื่อนไข (ถูกหรือผิด)

ตัวอย่างที่ 3.7

```
var = 0;
while(var < 200)
{
// do something repetitive 200 times
var++; }
```

3.3 ส่วนของตัวกระทำทางคณิตศาสตร์

ประกอบด้วยตัวกระทำ 5 ตัวคือ + (บวก),- (ลบ), * (คู ณ), / (หาร) และ % (หารเอาเศษ)

3.3.1 ตัวกระทำทางคณิตศาสตร์

บวก ลบ คูณ และหาร ใช้หาค่าผลรวม ผลต่าง ผลคูณ และผลหาร ค่าของตัวถูกกระทำสองตัวโดยให้คำตอบมีประเภทตรงกับตัวถูกกระทำทั้งสองตัว เช่น 9/4 ให้คำตอบเท่ากับ 2 เนื่องจากทั้ง 9 และ 4 เป็นตัวแปรเลขจำนวนเต็ม (int) นอกจากนี้ตัวกระทำทางคณิตศาสตร์อาจทำให้เกิดโอเวอร์โฟลว์ (overflow) ถ้าผลลัพธ์ที่ได้มีขนาดใหญ่เกินกว่าจะสามารถเก็บในตัวแปรประเภทนั้น ถ้าตัวที่ถูกกระทำต่างประเภทกันผลลัพธ์ที่ได้ เช่น 9/4 = 2 หรือ 9/4.0 = 2.25

รูปแบบคำสั่ง

```
result = value1 + value2;
result = value1 - value2;
result = value1 * value2;
result = value1 / value2;
```

พารามิเตอร์

value1 : เป็นค่าของตัวแปรหรือค่าคงที่ใดๆ

value2: เป็นค่าของตัวแปรหรือค่าคงที่ ใดๆ

ตัวอย่างที่ 3.8

```
y = y + 3;
x = x - 7;
i = j * 6;
r = r / 5;
```

เทคนิคสำหรับการเขียนโปรแกรม

- เลือกขนาดของตัวแปรให้ใหญ่พอสำหรับเก็บค่าผลลัพธ์ที่มากที่สุดของการคำนวณ
- ต้องทราบว่าที่ค่าใดตัวแปรที่เก็บจะมีการวนซ้ำค่ากลับ และวนกลับอย่างไร ตัวอย่างเช่น (0 ไป 1) หรือ (0 ไป -32768)
- สำหรับการคำณวณที่ต้องการเศษส่วนให้ใช้ตัวแปรประเภท float แต่ให้ระวังผลลบ เช่นตัวแปรมีขนาดใหญ่ คำนวณได้ช้า
- ใช้ตัวกระทำ cast เช่น (int)myfloat ในการเปลี่ยนประเภทของตัวแปรชั่วคราวขณะที่โปรแกรมทำงาน

3.3.2 ตัวกระทำ % หารเอาเศษ

ใช้หาค่าเศษที่ได้ของการหารเลขจำนวนเต็ม 2 ตัว ตัวกระทำหารเอาเศษไม่สามารถใช้งานกับตัวแปรเลขทศนิยม (float)

รูปแบบคำสั่ง result = value1 % value2;

พารามิเตอร์ value1 - เป็นตัวแปรประเภท byte,char,int หรือ long

value2 - เป็นตัวแปรประเภท byte,char,int หรือ long

ผลที่ได้เศษจากการหารค่าเลขจำนวนเต็ม เป็นข้อมูลชนิดเลขจำนวนเต็ม

ตัวอย่างที่ 3.9

```
x = 7 % 5; // x now contains 2
x = 9 % 5; // x now contains 4
x = 5 % 5; // x now contains 0
x = 4 % 5; // x now contains 4
```

ตัวกระทำหารเอาเศษนี้ใช้ประโยชน์มากในงานที่ต้องการให้เหตุการณ์เกิดขึ้นด้วยช่วงเวลาที่สม่ำเสมอ หรือใช้ทำให้หน่วยความจำที่เก็บตัวแปรอะเรย์ เกิดการส่งค่ากลับ (Roll Over)

ตัวอย่างที่ 3.10

```
// check a sensor every 10 times through a loop
    void setup()
    {
    i++;
    if ((i % 10) == 0)
    {
    x = analogRead(sensPin); // read sensor every ten times through loop
    }
    }
// setup a buffer that averages the last five samples of a sensor
    int senVal[5]; // create an array for sensor data
    int i, j; // counter variables
    long average; // variable to store average

    void loop()
    {
// input sensor data into oldest memory slot
    senVal[(i++) % 5] = analogRead(sensPin);
    average = 0;
    for (j=0; j<5; j++)
    {
```

```
average += sensVal[j]; // add samples
}
average = average / 5; // divide by total
}
```

3.4 ส่วนของตัวกระทำเปรียบเทียบ

ใช้ประกอบกับคำสั่ง if() และ while() เพื่อทดสอบเงื่อนไขหรือเปรียบเทียบค่าตัวแปรต่างๆ โดยจะเขียนเป็นนิพจน์อยู่ภายในเครื่องหมาย ()

x == y (x เท่ากับ y)
x != y (x ไม่เท่ากั บ y)
x < y (x น้อยกว่า y)
x > y (x มากกว่า y)
x <= y (x น้อยกว่าหรือเท่ากับ y)
x >= y (x มากกว่าหรือเท่ากับ y)

ใช้ในการเปรียบเทียบของคำสั่ง if() มี 3 ตัวคือ &&, || และ !

3.4.1 && (ตรรกะ และ)

ให้ค่าเป็นจริงเมื่อผลการเปรียบเทียบทั้งสองข้างเป็นจริงทั้งคู่

ตัวอย่างที่ 3.11

```
if (x > 0 && x < 5)
{
    // ...
}
```

ให้ค่าเป็นจริงเมื่อ x มากกว่า 0 และน้อยกว่า 5 (มีค่า 1 ถึ ง 4)

3.4.2 || (ตรรกะ หรือ)

ให้ค่าเป็นจริงเมื่อผลการเปรียบเทียบพบว่า มีตัวแปรใดเป็นจริงหรือเป็นจริงทั้งคู่

ตัวอย่างที่ 3.12

```
if (x > 0 || y > 0)
{   // ...
```

} ให้ผลเป็นจริงเมื่อ x หรือ y มีค่ามากกว่า 0

3.4.3 ! (ใช้กลับผลเป็นตรงกันข้าม)

ให้ค่าเป็นจริงเมื่อผลการเปรียบเทียบเป็นเท็จ

ตัวอย่างที่ 3.13

```
if (!x)
{ // ...
```

} ให้ผลเป็นจริงถ้า x เป็นเท็จ (เช่น ถ้า x = 0 ให้ผลเป็นจริง)

3.4.4 ข้อควรระวัง

ระวังเรื่องการเขียนโปรแกรม ถ้าต้องการใช้ตัวกระทำตรรกะ และต้องเขียนเครื่องหมาย && ถ้าลืมเขียนเป็น & จะเป็นตัวกระทำ และระดับบิตกับตัวแปรซึ่งให้ผลที่แตกต่างเช่นกันในการใช้ตรรกะ หรือให้เขียนเป็น || (ขีดตั้งสองตัวติดกัน) ถ้าเขียนเป็น | (ขีดตั้งตัวเดียว) จะหมายถึงตัวกระทำหรือระดับบิตกับตัวแปรตัวกระทำ NOT ระดับบิต (~) จะแตกต่างจากตัวกลับผลให้เป็นตรงข้าม (!) ให้เลือกใช้ให้ถูกต้อง

ตัวอย่างที่ 3.14

```
if (a >= 10 && a <= 20){}
// true if a is between 10 and 20
```

3.5 ส่วนของตัวกระทำระดับบิต

ตัวกระทำระดับบิตจะนำบิตของตัวแปรมาประมวลผล ใช้ประโยชน์ในการแก้ปัญหาด้านการเขียนโปรแกรมได้หลากหลายตัวกระทำ ระดับของภาษา C (ซึ่งรวมถึง Arduino) มี 6 ตัวได้แก่

& (bitwise AND)

| (OR)

^ (Exclusive OR)

~ (NOT)

<< (เลื่อนบิตไปทางขวา)

>> (เลื่อนบิตไปทางซ้าย)

3.5.1 ตัวกระทำระดับบิต AND (&)

คำสั่ง AND ในระดับบิตของภาษา C เขียนได้โดยใช้ & หนึ่งตัวโดยต้องเขียนระหว่างนิพจน์ หรือตัวแปรที่เป็นเลขจำนวนเต็ม การทำงานจะนำข้อมูลแต่ละบิตของตัวแปรทั้งสองตัวมากระทำทางตรรกะและ (AND) โดยมีกฎดังนี้ ถ้าอินพุตทั้งสองตัวเป็น “1” ทั้งคู่ เอาต์พุตเป็น “1” กรณีอื่นๆเอาต์พุตเป็น “0” ดังตัวอย่างต่อไปนี้ในการดูให้คู่ของตัวกระทำตามแนวตั้ง

```
0 0 1 1 Operand1
0 1 0 1 Operand2
————————————————
0 0 0 1 Returned result
```

ใน Arduino ตัวแปรประเภท int จะมีขนาด 16 บิต ดังนั้นเมื่อใช้ตัวกระทำระดับบิต AND จะมีการกระทำ

ตรรกะและพร้อมกันกับข้อมูลทั้ง 16 บิต ดังตัวอย่างในส่วนของโปรแกรมต่อไปนี้

ตัวอย่างที่ 3.15

```
int a = 92; //   in binary: 0000000001011100
int b = 101; // in binary: 0000000001100101
int c = a & b; //   result: 0000000001000100
// or 68 in decimal.
```

ในตัวอย่างนี้จะนำข้อมูลทั้ง 16 บิตของตัวแปร a และ b มากระทำทางตรรกะ AND แล้วนำผลลัพธ์ที่ได้ทั้ง 16 บิตไปเก็บที่ตัวแปร c ซึ่งได้ค่าเป็น 01000100 ในเลขฐานสองหรือเท่ากับ 68 ฐานสิบ

นิยมใช้ตัวกระทำระดับบิต AND เพื่อเลือกข้อมูลบิตที่ต้องการ (อาจเป็นหนึ่งบิตหรือหลายบิต) จากตัวแปร int ซึ่งการเลือกเพียงบางบิตนี้จะเรียกว่า masking

3.5.2 ตัวกระทำระดับบิต OR (|)

คำสั่งระดับบิต OR ของภาษา C เขียนได้โดยใช้เครื่องหมาย | หนึ่งตัว โดยต้องเขียนระหว่างนิพจน์ หรือตัวแปรที่เป็นเลขจำนวนเต็ม การทำงานจะนำข้อมูลแต่ละบิตของตัวแปรทั้งสองตัวมากระทำทางตรรกะหรือ (OR) โดยมีกฎดังนี้ ถ้าอินพุตตัวใดตัวหนึ่งหรือทั้งสองตัวเป็น “1” เอาต์พุตเป็น “1” กรณีที่อินพุตเป็น “0” ทั้งคู่เอาต์พุตจึงจะเป็น “0” ดังตัวอย่างต่อไปนี้

```
0 0 1 1 Operand1
0 1 0 1 Operand2
————————————————
0 1 1 1 Returned result
```

ตัวอย่างที่ 3.16

ส่วนของโปรแกรมแสดงการใช้ตัวกระทำระดับบิต OR

```
int a = 92; //   in binary: 0000000001011100
int b = 101; // in binary: 0000000001100101
int c = a | b; //     result: 0000000001111101
// or 125 in decimal.
```

ตัวอย่างที่ 3.17

โปรแกรมแสดงการใช้ตัวกระทำระดับบิต AND และ OR ตัวอย่างงานที่ใช้ตัวกระทำระดับบิต AND และ OR เป็นงานที่โปรแกรมเมอร์เรียกว่า Read-Modify-Write on a port สำหรับไมโครคอนโทรลเลอร์ 8 บิต ค่าที่อ่านหรือเขียนไปยังพอร์ตมีขนาด 8 บิต ซึ่งแสดงค่าอินพุตที่ขาทั้ง 8 ขา การเขียนค่าไปยังพอร์ตจะเขียนค่าครั้งเดียวได้ทั้ง 8 บิต

ตัวแปรชื่อ PORTD เป็นค่าที่ใช้แทนสถานะของขาดิจิตอลหมายเลข 0,1,2,3,4,5,6,7 ถ้าบิตใดมีค่าเป็น 1 ทำให้ขานั้นมีค่าลอจิกเป็น HIGH ต้องกำหนดให้ขาพอร์ตนั้นๆทำงานเป็นเอาต์พุตด้วยคำสั่ง pinMode()

ดังนั้นถ้ากำหนดค่าให้ PORTD = B00110001 ก็คือต้องการให้ขา 2, 3 และ 7 เป็น HIGH ในกรณีนี้ไม่ต้องเปลี่ยนค่าสถานะของขา 0 และ 1 ซึ่งปกติแล้วฮาร์ดแวร์ของ Arduino ใช้ในการสื่อสารแบบอนุกรม ถ้าไปเปลี่ยนค่าแล้วจะกระทบต่อการสื่อสารแบบอนุกรม

อัลกอริธึมสำหรับโปรแกรมเป็นดังนี้

- อ่านค่าจาก PORTD แล้วล้างค่าเฉพาะบิตที่ต้องการควบคุม (ใช้ตัวกระทำแบบบิต AND)
- นำค่า PORTD ที่แก้ไขจากข้างต้นมารวมกับค่าบิตที่ต้องการควบคุม (ใช้ตัวกระทำแบบบิต OR)

ซึ่งเขียนเป็นโปรแกรมได้ ดังนี้

```
int i; // counter variable
int j;
void setup()
{
DDRD = DDRD | B11111100;
// set direction bits for pins 2 to 7,
// leave 0 and 1 untouched (xx | 00 == xx)
// same as pinMode(pin,OUTPUT) for pins 2 to 7
Serial.begin(9600);
}
void loop()
{
for (i=0; i<64; i++)
{
PORTD = PORTD & B00000011;
// clear out bits 2 - 7, leave pins 0
// and 1 untouched (xx & 11 == xx)
j = (i << 2);
// shift variable up to pins 2 - 7
// to avoid pins 0 and 1
PORTD = PORTD | j;
// combine the port information with
// the new information for LED pins
Serial.println(PORTD, BIN);
// debug to show masking
```

```
delay(100);
} }
```

3.5.3 คำสั่งระดับบิต Exclusive OR (^)

เป็นโอเปอร์เตอร์พิเศษที่ไม่ค่อยได้ใช้ในภาษา C/C++ ตัวกระทำระดับบิต exclusive OR (หรือ XOR) จะเขียนโดยใช้สัญลักษณ์เครื่องหมาย ^ ตัวกระทำนี้มีการทำงานใกล้เคียงกับตัวกระทำระดับบิต OR แต่ต่างกันเมื่ออินพุตเป็น “1” ทั้งคู่จะให้เอาต์พุตเป็น “0” แสดงการทำงานได้ดังนี้

```
0  0  1  1          Operand1
0  1  0  1          Operand2

_______
0  1  1  0          Returned  result
```

หรือกล่าวได้อีกอย่างว่าตัวกระทำระดับบิต XOR จะให้เอาต์พุตเป็น “0” เมื่ออินพุตทั้งสองตัวมีค่าเหมือนกัน และให้เอาต์พุตเป็น “1” เมื่ออินพุตทั้งสองมีค่าต่างกัน

ตัวอย่างที่ 3.18

```
int x = 12;          / / binary: 1100
int y = 10;          / / binary: 1010
int z = x ^ y;       / / binary: 0110, or decimal 6
```

ตัวกระทำระดับบิต XOR จะใช้มากในการสลับค่าบางบิตของตัวตัวแปร int เช่นกลับจาก “0” เป็น “1” หรือกลับจาก”1” เป็น “0”

เมื่อใช้ตัวกระทำระดับบิต XOR ถ้าบิตของ mask เป็น “1” ทำให้บิตนั้นถูกสลับค่า ถ้า mask มีค่าเป็น “1” บิตนั้นมีค่าคงเดิม ตัวอย่างต่อไปนี้เป็นโปรแกรมแสดงการสั่งให้ขาดดิจิตอล 5(Di5) มีการกลับลอจิกตลอดเวลา

ตัวอย่างที่ 3.19

```
/ / Blink_Pin_5
/ / demo for Exclusive OR
        void setup ( )
        {
        DDRD = DDRD I B00100000;
/ / set digital pin five as OUTPUT
        Serial. begin (9600);
        }
        void loop ()
        {
        PORTD = PORTD ^ B00100000;
```

```
/ / invert bit 5 (digital pin 5),
/ / leave others untouched
        delay (100);
        }
```

3.5.4 ตัวกระทำระดับบิต NOT (~)

ตัวกระทำระดับบิต NOT จะเขียนโดยใช้สัญลักษณ์เครื่องหมาย **~** ตัวกระทำนี้จะใช้กับตัวถูกกระทำเพียงตัวเดียวที่อยู่ขวามือ โดยการสลับบิตทุกบิตให้มีค่าตรงกันข้ามคือ จาก ”0” “1” และจาก “1” เป็น “0” ดังตัวอย่าง

```
0  1          Operand1
____
1  0          ~ Operand1
int a = 103;     / / binary: 0000000001100111
int b =  ~a;     / / binary: 1111111110011000
```

เมื่อกระทำแล้วทำให้ตัวแปร b มีค่า -104 (ฐานสิบ) ซึ่งคำตอบที่ได้ติดลบ เนื่องจากบิตที่มีความสำคัญสูงสุด (บิตซ้ายมือสุด) ของตัวแปร int อันเป็นบิตแจ้งว่าตัวเลขเป็นบวกหรือลบ มีค่าเป็น “1” แสดงว่าค่าที่ได้นี้ติดลบ โดยในคอนโทรลเลอร์จะเก็บค่าตัวเลขทั้งบวกและลบ ตามระบบทูคอมพลีเมนต์ (2’s complement)

การกระทำประกาศตัวแปร int ซึ่งมีความหมายเหมือนกับการประกาศตัวแปรเป็น signed int ต้องระวังค่าของตัวแปรจะติดลบได้

3.5.5 คำสั่งเลื่อนบิตไปทางซ้าย (<<) และเลื่อนบิตไปทางขวา (>>)

ในภาษา C/C++ มีตัวกระทำเลื่อนบิตไปทางซ้าย <<

3.6 ไวยากรณ์ภาษา C / C++ ของ Arduino

3.6.1 เซมิโคลอน – semicolon ;

ใช้เขียนแจ้งว่าจบคำสั่ง

ตัวอย่าง 3.23

```
int a = 13;
```

บรรทัดคำสั่งที่ลืมเขียนปิดท้ายด้วยเซมิโคลอน จะทำให้แปลโปรแกรมไม่ผ่าน โดยตัวแปรภาษาอาจจะแจ้งให้ทราบว่า ไม่พบเครื่องหมายเซมิโคลอน หรือแจ้งเป็นการผิดพลาดอื่นๆ บางกรณีที่ตรวจสอบบรรทัดที่แจ้งว่าเกิดการผิดพลาดแล้วไม่พบที่ผิด ให้ตรวจสอบบรรทัดก่อนหน้านั้น

3.6.2 วงเล็บปีกกา – curly brace { }

เครื่องหมายวงเล็บปีกา เป็นส่วนสำคญของภาษาซี โดยมีการใช้งานต่างตำแหน่ง สร้างความสับสนให้กับผูที่เริ่มต้นวงเล็บปีกกาเปิด { จะต้องเขียนตามด้วยวงเล็บปีกกาปิด } ด้วยเสมอ หรือที่เรียกว่าวงเล็บต้องครบคู่

ในซอฟต์แวร์ Arduino IDE ที่ใช้เขียนโปรแกรมจะมีความสามารถในการตรวจสอบการควบคู่ของเครื่องหมายวงเล็บผู้ใช้งานเพียงแค่คลิกที่วงเล็บ จะแสดงวงเล็บที่เหลือ

สำหรับโปรแกรมเมอร์มือใหม่ และโปรแกรมเมอร์ที่ย้ายจากภาษา BASIC เป็นภาษา C มักจะสับสนกับการใช้เครื่องหมายวงเล็บ แท้ที่จริงแล้วเครื่องหมายปีกกาปิดนี้เทียบได้กับคำสั่ง RETURN ของ Subroutine (function) หรือแทนคำสั่ง ENDIF ในการเปรียบเทียบ และแทนคำสั่ง NEXT ของคำสั่งวนรอบ FOR

เนื่องจากมีการใช้วงเล็บปีกกาได้หลากหลาย ดังนั้นเมื่อต้องการเขียนคำสั่งที่ต้องใช้เครื่องหมายวงเล็บ เมื่อเขียนวงเล็บเปิดแล้วให้เขียนเครื่องหมายวงเล็บปิดทันที ถัดมาจึงค่อยเคาะปุ่ม Enter ในระหว่างเครื่องหมายวงเล็บเพื่อขึ้นบรรทัดใหม่ แล้วเขียนคำสั่งที่ต้องการ ถ้าทำได้ตามนี้วงเล็บจะครบคู่แน่นอน

สำหรับวงเล็บที่ไม่ครบคู่ทำให้เกิดผิดพลาดตอนคอมไฟล์โปรแกรม ถ้าเป็นโปรแกรมขนาดใหญ่จะหาที่ผิดได้ยาก ตำแหน่งที่อยู่ของเครื่องหมายวงงเล็บแต่ละตัวจะมีผลอย่างมากต่อไวยากรณ์ของภาษาคอมพิวเตอร์ การย้ายตำแหน่งวงเล็บไปเพียงหนึ่งหรือสองบรรทัดทำให้ตัวโปรแกรมทำงานผิดไป

ตำแหน่งที่ใช้วงเล็บปีกกา

ฟังก์ชั้น (function)

```
void myfunction (datatype argument)
        {
        statements (s)
        }
```

คำสั่งวนรอบ (loops)

```
while (boolean expression)
        {
        statement (s)
        }
        do
        {
        statement (s)
        }
        while (boolean expression);
        for (initialisation; termination condition; incrementing expr)
        {
        statement (s)
        }
```

คำสั่งทดสอบเงื่อนไข (condition)

```
If (boolean  expression)
{
statement (s)
}
else  if  (boolean  expression)
{
statement (s)
}
else
{
statement (s)
}
```

3.6.3 หมายเหตุบรรทัดเดียวและหลายบรรทัด / / และ / * . . . */

เป็นส่วนของโปรแกรมที่ผู้ใช้เขียนเพิ่มเติมว่าโปรแกรมทำงานอย่างไร โดยส่วนที่เป็นหมายเหตุจะไม่ถูกคอมไพล์ ไม่นำไปประมวลผล มีประโยชน์มากสำหรับการตรวจสอบโปรแกรมภายหลังหรือใช้แจ้งให้เพื่อนร่วมงานหรือบุคคลอื่นทราบว่าบรรทัดนี้ใช้ทำอะไร ตัวหมายเหตุภาษา C มี 2 ประเภทคือ

(1) หมายเหตุบรรทัดเดียว เขียนเครื่องสเลช // 2 ตัวหน้าบรรทัด

(2) หมายเหตุหลายบรรทัด เขียนเครื่องหมายสเลช / คู้กับดอกจัน * คร่อมข้อความที่เป็นหมายเหตุ เช่น

```
/*  blabla */
```

3.6.4 # define

เป็นคำสั่งที่ใช้งานมาก ในการกำหนดค่าคงที่ให้กับโปรแกรม ในการกำหนดค่าคงที่ไม่ได้เปลืองพื้นที่หน่วยความจำของไมโครคอนโทรลเลอร์แต่อย่างไร เมื่อถึงขั้นตอนแปลภาษา คอมไพเลอร์จะแทนที่ตัวอักษรข้อความด้วยค่าที่กำหนดไว้ ใน Arduino จะใช้คำสั่ง # define ตรงกับภาษา C

รูปแบบ

```
# define constantName value
```

อย่าลืมเครื่องหมาย #

ตัวอย่างที่ 3-24

```
# define ledpin 3
```

เป็นการกำหนดให้ตัวแปร ledpin เท่ากับค่าคงที่ 3

เทคนิคสำหรับการเขียนโปรแกรม

ท้ายคำสั่ง # define ไม่ต้องมีเครื่องหมายเซมิโคลอน ถ้าใส่เกินแล้วเวลาคอมไพล์โปรแกรมจะแจ้งว่าเกิดการผิดพลาดในบรรทัดถัดไป

3.6.5. # include

ใช้สั่งให้รวมไฟล์อื่นๆ เข้ากับไฟล์โปรแกรมหลักก่อน แล้วจึงทำการคอมไพล์โปรแกรม

รูปแบบคำสั่ง

include <file>

include “file”

ตัวอย่างที่ 3.25

include <stdio.h>

include “lcd.h”

บรรทัดแรกจะสั่งให้เรียกไฟล์ stdio.h มารวมกับไฟล์โปรแกรมหลัก โดยค้นหาไฟล์จากตำแหน่งที่เก็บไฟล์ระบบของ Arduino โดยปกติเป็นไฟล์มาตรฐานที่มาพร้อมกับ Arduino

บรรทัดที่ 2 สั่งให้รวมไฟล์ lcd.h มารวมกับไฟล์โปรแกรมหลัก โดยหาไฟล์จากตำแหน่งของไฟล์ภาษา C ปกติเป็นไฟล์ที่ผู้ใช้สร้างขึ้นเอง

ในการแก้ไขโปรแกรมใน Arduino มีข้อแนะนำว่า อย่าแก้ไขบรรทัดนั้นทันที ให้ทำบรรทัดนั้นเป็นหมายเหตุก่อนแล้วจึงแก้โปรแกรมในบรรทัดนั้น

3.7 ตัวแปร

ตัวแปรเป็นตัวอักษรหลายตัวๆ ที่กำหนดขึ้นในโปรแกรมเพื่อใช้ในการเก็บค่าข้อมูลต่างๆเช่น ค่าที่อ่านได้จากตัวตรวจจับ ที่ต่ออยู่กับขาพอร์ตแอนะล็อกของ Arduino ตัวแปรมีหลายประเภทดังนี้

3.7.1 char : ตัวแปรประเภทตัวอักขระ

เป็นตัวแปรที่มีขนาด 1 ไบต์ (8 บิต) มีไว้เพื่อเก็บค่าตัวอักษร ตัวอักษรในภาษาซีจะเขียนอยู่ในเครื่องหมายคำพูดขีดเดียวเช่น ‘A’ (สำหรับข้อความที่ประกอบจากตัวอักษรหลายตัวเขียนต่อกันจะเขียนอยู่ในเครื่องหมายคำพูดปกติเช่น “ABC”) สามารถสั่งกระทำทางคณิตศาสตร์กับตัวอักษรได้ในกรณีจะนำค่ารหัส ASCII ของตัวอักษรมาใช้เช่น ‘A’ +1 มีค่าเท่ากับ 66 เนื่องจากค่ารหัส ASCII ของตัวอักษร A เท่ากับ 65

รูปแบบคำสั่ง

charsign = ' ';

พารามิเตอร์

char var = ‘x’;

var คือชื่อของตัวแปรประเภท char ที่ต้องการ

x คือค่าที่ต้องการกำหนดให้กับตัวแปร ในที่นี้เป็นตัวอักษรหนึ่งตัว

3.7.2 byte : ตัวแปรประเภทตัวเลข 8 บิตหรือ 1 ไบต์

ตัวแปร byte ใช้เก็บค่าตัวเลขขนาด 8 บิต มีค่าได้จาก 0 - 255

ตัวอย่างที่ 3.26

```
byte b = B10010111; // “B” is the binary formatter (151 decimal)
```

3.7.3 int : ตัวแปรประเภทตัวเลขจำนวนเต็ม

ย่อจาก interger ซึ่งแปลว่าเลขจำนวนเต็ม int เป็นตัวแปรพื้นฐานสำหรับเก็บตัวเลข ตัวแปรหนึ่งตัวมีขนาด 2 ไบต์เก็บค่าได้จาก -32,768 ถึ ง 32,767 ในการเก็บค่าตัวเลขติดลบ จะใช้เทคนิคที่เรียกว่าทูคอพลีเมนต์ (2’s complement) บิตสูงสุดบางทีจะเรียกว่าเป็นบิตเครื่องหมายหรือ sign bit ถ้ามีค่าเป็น “1” แสดงว่าค่าติดลบ ใน Arduino จะจัดการกับตัวเลขค่าติดลบให้เอง ทำให้นำค่าตัวแปรไปคำนวณได้อย่างถูกต้อง อย่างไรก็ตามเมื่อนำตัวเลขค่าติดลบนี้ไปเลื่อนบิตไปทางขวา (>>) จะมีปัญหาเรื่องค่าของตัวเลขที่ผิดพลาด

รูปแบบคำสั่ง

```
int var = val;
```

พารามิเตอร์

var คือชื่อของตัวแปรประเภท int ที่ต้องการ

val คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.27

```
int ledPin = 31;
```

เทคนิคสำหรับการเขียนโปรแกรม

เมื่อตัวแปรมีค่ามากกว่าค่าสูงสุดที่เก็บได้ จะเกิดการ “ล้นกลับ” (Roll Over) ไปยังค่าต่ำสุดที่เก็บได้และเมื่อมีค่าน้อยกว่าค่าต่ำสุดที่เก็บได้ จะล้นกลับไปยังค่าสูงสุด ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.28

```
int x
x = -32,768;
x = x - 1; // x now contains 32,767
// - rolls over in neg. direction
x = 32,767;
x = x + 1; // x now contains -32,768 - rolls over
```

3.7.4 unsigned int : ตัวแปรประเภทเลขจำนวนเต็มไม่คิดเครื่องหมาย

ตัวแปรประเภทนี้คล้ายกับตัวแปร int ตรงที่ใช้หน่วยความจำ 2 ไบต์ แต่จะเก็บเลขจำนวนเต็มบวกโดยเก็บค่า 0 ถึง 65,535

รูปแบบคำสั่ง

```
unsigned int var = val;
```

พารามิเตอร์

var คือชื่อของตัวแปร int ที่ต้องการ

val คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.29

```
unsigned int ledPin = 31;
```

เทคนิคสำหรับการเขียนโปรแกรม

เมื่อตัวแปรมีค่ามากกว่าค่าสูงสุดจะล้นกลับไปค่าต่ำสุด และเมื่อมีค่าน้อยกว่าค่าต่ำสุดจะล้นกลับเป็นค่าสูงสุด ดังตัวอย่าง

ตัวอย่างที่ 3.30

```
unsigned int x
x = 0;
x = x - 1; // x now contains 65535
// - rolls over in neg direction
x = x + 1; // x now contains 0 - rolls over
```

3.7.5 long : ตัวแปรประเภทเลขจำนวนเต็ม 32 บิต

เป็นตัวแปรเก็บค่าเลขจำนวนเต็ม ที่ขยายความจุเพิ่มจากตัวแปร int โดยตัวแปร long หนึ่งตัวกินพื้นที่หน่วยความจำ 32 บิต (4 ไบต์) เก็บค่าได้จาก -2,147,483,648 ถึง 2,147,483,647

รูปแบบคำสั่ง

```
long var = val;
```

พารามิเตอร์

var คือชื่อของตัวแปร long ที่ต้องการ

val คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.31

```
long time;
void setup()
{
        Serial.begin(9600);
}
void loop()
{
        Serial.print("Time: ");
        time = millis();
```

```
        Serial.println(time); //prints time since program started
        delay(1000); // wait a second so as not to send massive amounts of data
    }
```

3.7.6 unsigned long : ตัวแปรประเภทเลขจำนวนเต็ม 32 บิต แบบไม่คิดเครื่องหมาย

เป็นตัวแปรเก็บค่าเลขจำนวนเต็มบวก ตัวแปรหนึ่งตัวกินพื้นที่หน่วยความจำ 32 บิต (4 ไบต์) เก็บค่าได้จาก 0 ถึ ง 4,294,967,295

รูปแบบคำสั่ง

```
unsigned long var = val;
```

พารามิเตอร์

var คือชื่อของตัวแปร unsigned long ที่ต้องการ

val คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.32

```
long time;
void setup()
{
Serial.begin(9600);
}
void loop()
{
Serial.print("Time: ");
time = millis();
Serial.println(time); //prints time since program started
delay(1000); // wait a second so as not to send massive amounts of data
}
```

3.7.7 float : ตัวแปรประเภทเลขทศนิยม

เป็นตัวแปรสำหรับเก็บค่าเลขทศนิยม นิยมใช้เก็บค่าสัญญาณแอนะล็อกหรือค่าที่ต่อเนื่อง ตัวแปรแบบนี้เก็บค่าได้ละเอียดกว่าตัวแปร int โดยเก็บค่าได้ในช่วง 3.4028235 x 1038 ถึ ง -3.4028235 x 1038 ตัวแปรหนึ่งตัวจะใช้พื้นที่หน่วยความจำ 32 บิต (4 ไบต์) ในการคำนวณคณิตศาสตร์กับตัวแปร float จะช้ากว่าการคำนวณของตัวแปร int ดังนั้นพยายามหลีกเลี่ยงการคำนวณกับตัวแปร float เช่นในคำสั่งวนรอบที่ทำงานด้วยความเร็วสูงสุด

สำหรับฟังก์ชันทางเวลาที่ต้องแม่นยำอย่างมาก โปรแกรมเมอร์บางคนจะทำการแปลงตัวเลขทศนิยมให้เป็นเลขจำนวนเต็มก่อน แล้วจึงคำนวณเพื่อให้ทำงานได้เร็วขึ้น จะเห็นได้ว่าการคำนวณคณิตศาสตร์ของเลข

floating point จะมีการใช้งานมากสำหรับการคำนวณค่าข้อมูลที่รับจากภายนอกเป็นตัวเลขทศนิยม ซึ่งทางผู้ศึกษาระบบไมโครคอนโทรลเลอร์มักจะมองข้ามไป

รูปแบบคำสั่ง

```
float var = val;
```

พารามิเตอร์

var คือชื่อของตัวแปร float ที่ต้องการ

val คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.33

```
float myfloat;
float sensorCalbrate = 1.117;
```

ตัวอย่างที่ 3.34

```
int x;
int y;
float z;
x = 1;
y = x / 2; // y now contains 0,
// integers can’t hold fractions
z = (float)x / 2.0;
// z now contains .5
// (you have to use 2.0, not 2)
```

ในฟังก์ชั่น serial.println() ของการส่งค่าตัวแปรไปยังพอร์ตอนุกรม จะตัดตัวเลขเศษทศนิยมออกให้เหลือเป็นเลขจำนวนเต็ม ถ้าต้องการเพิ่มความละเอียดให้นำค่าตัวแปรคูณด้วย 10 ยกกำลังต่างๆตามที่ต้องการ

3.7.8 double : ตัวแปรประเภทเลขทศนิยมความละเอียดสองเท่า

เป็นตัวแปรทศนิยมความละเอียดสองเท่า มีขนาด 8 ไบต์ ค่าสูงสุดที่เก็บได้คือ 1.7976931348623157 x 10308 ใน Arduino มีหน่วยความจำขนาดจำกัด จึงไม่นิยมใช้ตัวแปรประเภทนี้

3.7.9 string : ตัวแปรประเภทข้อความ

เป็นตัวแปรเก็บข้อความ ซึ่งในภาษา C จะนิยามเป็นอะเรย์ของตัวแปรประเภท char

ตัวอย่างที่ 3.35 ตัวอย่างการประกาศตัวแปรสตริง

```
char Str1[15];
char Str2[8] = {‘a’,‘r’,‘d’,‘u’,‘i’,‘n’,‘o’};
char Str3[8] = {‘a’,‘r’,‘d’,‘u’,‘i’,‘n’, ‘o’,’\0'};
char Str4[ ] = “arduino”;
```

```
char Str5[8] = “arduino”;
char Str6[15] = “arduino”;
```

- Str1 เป็นการประกาศตัวแปรสตริงโดยไม่ได้กำหนดค่าเริ่มต้น
- Str2 ประกาศตัวแปรสตริงพร้อมกำหนดค่าให้กับข้อความทีละตัวอักษร จากตัวอย่างคอมไพเลอร์จะเพิ่ม null character ให้เอง
- Str3 ประกาศตัวแปรสตริงพร้อมกำหนดค่าให้กับข้อความทีละตัวอักษร จากตัวอย่างเพิ่มค่า null String เอง
- Str4 ประกาศตัวแปรสตริงค์พร้อมกำหนคค่าตัวแปรในเครื่องหมายคำพูด จากตัวอย่างไม่ได้กำหนดขนาดตัวแปรคอมไพเลอร์ จะกำหนดขนาดให้เองตามจำนวนตัวอักษร + 1 สำหรับ null string
- Str5 ประกาศตัวแปรสตริงพร้อมกำหนดตัวแปรในเครื่องหมายคำพูด จากตัวอย่างต้องกำหนดขนาดตัวแปรเอง
- Str6 ประกาศตัวแปรสตริง โดยกำหนดขนาดเผื่อไว้สำหรับข้อความอื่นที่ยาวมากกว่านี้

3.7.10 การเพิ่มตัวอักษรแจ้งว่าจบข้อความ (null termination)

ในตัวแปรสตริงของภาษาซี กำหนดให้ตัวอักษรสุดท้ายเป็นตัวแจ้งการจบข้อความ (null string) ซึ่งก็คือตัวอักษร \0 ในการกำหนดขนาดของตัวแปร (ค่าในวงเล็บเหลี่ยม) จะต้องกำหนดให้เท่ากับจำนวนตัวอักษร+ 1 ดังในตัวแปร Str2 และ Str3 ในตัวอย่างที่ 3.35 ที่ข้อความ Arduino มีตัวอักษร 7 ตัว ในการประกาศตัวแปรต้องระบุเป็น [8] ในการประกาศตัวแปรสตริง ต้องเผื่อพื้นที่สำหรับเก็บตัวอักษรแจ้งว่าจบข้อความ มิฉะนั้นคอมไพเลอร์จะแจ้งเตือนว่าเกิดการผิดพลาด ในตัวอย่างที่ 3.35 ตัวแปร Str1 และ Str6 เก็บข้อความได้สูงสุด 14 ตัวอักษร

3.7.11 เครื่องหมายคำพูดขีดเดียวและสองขีด

ปกติแล้วจะกำหนดค่าตัวแปรสตริงภายในเครื่องหมายคำพูด เช่น "Abc" สำหรับตัวแปรตัวอักษร (char) จะกำหนดค่าภายในเครื่องหมายคำพูดขีดเดียว 'A'

3.7.12 ตัวแปรอะเรย์ (array)

ตัวแปรอะเรย์เป็นตัวแปรหลายตัว ที่ถูกเก็บรวมอยู่ในตัวแปรชื่อเดียวกัน โดยอ้างถึงตัวแปรแต่ละตัวด้วยหมายเลขดัชนีที่เขียนอยู่ในวงเล็บสี่เหลี่ยม ตัวแปรอะเรย์ของ Arduino จะอ้างอิงตามภาษา C ตัวแปรอะเรย์อาจจะซับซ้อน แต่ใช้แค่ตัวแปรอะเรย์อย่างง่ายจะตรงไปตรงมา

ตัวอย่างการประกาศตัวแปรอะเรย์

```
int myInts [6];
int myPins [ ] = {2, 4, 8, 3, 6};
int mySensVals [6] = {2, 4, -8, 3, 2};
char message [6] = “hello”;
```

เราสามารถประกาศตัวแปรอะเรย์ได้โดยยังไม่กำหนดค่าดังตัวแปร myInts ในตัวแปร myPins จะประกาศตัวแปรอะเรย์โดยไม่ระบุขนาด ซึ่งทำได้เมื่อประกาศตัวแปรแล้วกำหนดค่าทันที เพื่อให้คอมไพเลอร์นับว่า ตัวแปรมี

สมาชิกกี่ตัวและกำหนดค่าได้ถูกต้อง

ท้ายที่สุดสามารถประกาศตัวแปรและกำหนดขนาดของตัวแปรอะเรย์ดังตัวแปร mySersVals ในการประกาศอะเรย์ของตัวแปร char จะต้องเผื่อที่สำหรับเก็บค่าตัวอักษรแจ้งว่าจบข้อความด้วย

การใช้งานตัวแปรอะเรย์

การใช้งานตัวแปรอะเรย์ทำได้โดยการพิมพ์ชื่อตัวแปร พร้อมกับระบุค่าดัชนีภายในเครื่องหมายวงเล็บสี่เหลี่ยม ค่าดัชนีของตัวแปรอะเรย์เริ่มต้นด้วยค่า 0 ดังนั้นค่าของตัวแปร mySensVals มีค่าดังนี้

mySensVals [0] == 2, mySensVals [1] == 4, ฯลฯ

การกำหนดค่าให้กับตัวแปรอะเรย์

```
mySensVals [0] = 10;
```

การเรียกค่าสมาชิกของตัวแปรอะเรย์

```
x = mySensVals [4];
```

เทคนิคการเขียนโปรแกรมเกี่ยวกับตัวแปรอะเรย์

ในการเรียกใช้ค่าสมาชิกของตัวแปรอะเรย์ ต้องระวังอย่าอ้างถึงค่าในวงเล็บที่เกินที่กำหนด เช่นประกาศตัวแปร int x [3] ตัวแปรมี 3 ตัว คือ x [0], x [1] และ x [2] ถ้าอ้างถึง x [3] จะเป็นการอ่านค่าจากหน่วยความจำซึ่งกำหนดไว้ใช้งานอย่างอื่น ค่าที่อ่านได้จะผิดพลาด

การเขียนค่าให้กับตัวแปรอะเรย์ตัวที่เกินกว่ากำหนดไว้ อาจทำให้โปรแกรมแฮงค์ (หยุดการทำงาน) หรือทำงานผิดเพี้ยนไป

การอ่านหรือเขียนค่าเกินค่าดัชนีของตัวแปรอะเรย์นี้ ทำให้เกิดบัก (ข้อผิดพลาด) ที่ยากต่อการค้นหา

อะเรย์และคำสั่งวนรอบ for

โดยทั่วไปเราจะพบการใช้งานตัวแปรอะเรย์ภายในคำสั่ง for โดยใช้ค่าตัวแปรนับรอบคำสั่ง for เป็นค่าดัชนีของตัวแปรอะเรย์ ดังตัวอย่างการพิมพ์ค่าสมาชิกแต่ละตัวของตัวแปรอะเรย์ผ่านพอร์ตอนุกรม ให้เขียนโปรแกรมดังนี้

```
int i;
for (i = 0; I < 5; I = I + 1)
{
Serial.println (myPins [i] );
}
```

ตัวอย่างโปรแกรมสาธิตการใช้งานตัวแปรอะเรย์ที่สมบูรณ์ ดูได้ในตัวอย่างในหัวข้อ Tutorials ในเว็บไซต์ www.arduino.cc

3.8 ขอบเขตของตัวแปร

ตัวแปรในภาษ C ที่ใช้ใน Arduino จะมีคุณสมบัติที่เรียกว่า “ขอบเขตของตัวแปร” (scope) ซึ่งแตกต่างจากภาษา BASIC ซึ่งตัวแปรทุกตัวมีสถานะเท่าเทียมกันหมดคือ เป็นแบบ global

3.8.1 ตัวแปรโลคอลและโกลบอล

ตัวแปรแบบโกลบอล (global variable) เป็นตัวแปรที่ทุกฟังก์ชั่นในโปรแกรมรู้จัก โดยต้องประกาศตัวแปรนอกฟังก์ชั่น สำหรับตัวแปรแบบโลคอลหรือตัวแปรท้องถิ่น เป็นตัวแปรที่ประกาศตัวแปรอยู่ภายในเครื่องหมายวงเล็บปีกกาของฟังก์ชั่น และรู้จักเฉพาะแค่ภายในฟังก์ชั่นนั้น

เมื่อโปรแกรมเริ่มมีขนาดใหญ่และซับซ้อนมากขึ้น การใช้ตัวแปร โลคอลจะมีประโยชน์มาก เนื่องจากแน่ใจได้ว่ามีแค่ฟังก์ชั่นนั้นเท่านั้นที่สามารถใช้งานตัวแปร ช่วยป้องกันการเกิดการผิดพลาดเมื่อฟังก์ชั่นทำการแก้ไขค่าตัวแปรที่ใช้งานโดยฟังก์ชั่นอื่น

ตัวอย่างที่ 3.36

```
        int PWMval; // any function will see this variable
void setup () {
        // ...
        }
void loop () {
        int i; // “I” is only “visible” inside of “loop”
        float f; // “f” is only “visible” inside of “loop”
        }
```

3.8.2 ตัวแปรสแตติก (static)

เป็นคำสงวน (Keyword) ที่ใช้ตอนประกาศตัวแปรที่มีขอบเขตใช้งานแค่ภายในฟังก์ชั่นเท่านั้น โดยต่างจากตัวแปรโลคอลตรงที่ตัวแปรแบบโลคอลจะถูกสร้างและลบทิ้งทุกครั้งที่เรียกใช้ฟังก์ชั่น สำหรับตัวแปร สแตติกเมื่อจบการทำงานของฟังก์ชั่นค่าตัวแปรจะยังคงอยู่ (ไม่ถูกลบทิ้ง) เป็นการรักษาค่าตัวแปรไว้ระหว่างการเรียกใช้งานฟังก์ชั่น ตัวแปรที่ประกาศเป็น static จะถูกสร้างและกำหนดค่าในครั้งแรกที่เรียกใช้ฟังก์ชั่น

3.9 การกำหนดค่าคงที่เลขจำนวนเต็มเป็นเลขฐานต่างๆ ของ Arduino

ค่าคงที่เลขจำนวนเต็มก็คือตัวเลขที่เขียนโปรแกรมของ Arduino โดยตรงเช่น 123 โดยปกติแล้วตัวเลขเหล่านี้จะเป็นเลขฐานสิบ (decimal) ถ้าต้องการกำหนดเป็นเลขฐานอื่นจะต้องใช้เครื่องหมายพิเศษระบุ เช่น

ฐาน	ตัวอย่าง
10 (decimal)	123
2 (binary)	B1111011
8 (octal)	0173
16 (hexadecimal)	0x7B

Decimal ก็คือเลขฐานสิบ ซึ่งเราใช้ในชีวิตประจำวัน

ตัวอย่าง 101== 101 decimal ((1 * 2^2) + (0 * 2^1) + 1)

Binary เป็นเลขฐานสองตัว ตัวเลขแต่ละหลักเป็นได้แค่ 0 หรือ 1

ตัวอย่าง B101 == 5 decimal ((1 * 2^2) + (0 * 2^1) + 1)

เลขฐานสองจะใช้งานได้ไม่เกิน 8 บิต (ไม่เกิน 1 ไบต์) มีค่าจาก 0 (B0) ถึง 255 (B11111111)

ถ้าต้องการป้อนเลขฐานสองขนาด 16 บิต (ตัวแปรประเภท int) จะต้องป้อนค่าสองขั้นตอนดังนี้

myInt = (B11001100 * 256) +B10101010; // B11001100 is the high byte

Octal เป็นเลขฐานแปด ตัวเลขแต่ละหลักมีค่าจาก 0 ถึง 7 เท่านั้น

ตัวอย่าง 0101 == 65 decimal ((1 * 8^2) + (0 * 8^1) + 1)

ข้อควรระวังในการกำหนดค่าคงที่ อย่าใส่เลข 0 นำหน้า มิฉะนั้นตัวคอมไพเลอร์จะแปลความหมายผิดไปว่าตัวเลขเป็นเลขฐาน 8

Hexadecimal (hex) เป็นเลขฐานสิบหก ตัวเลขแต่ละหลักมีค่าจาก 0 ถึง 9 และตัวอักษร A คือ 10, B คือ 11 ไปจนถึง F ซึ่งเท่ากับ 15

ตัวอย่าง 0x101 == 257 decimal ((1 * 16^2) + (0 * 16^1) + 1)

3.10 ค่าคงที่ (constants)

ค่าคงที่เป็น กลุ่มตัวอักษรหรือข้อความที่ได้กำหนดค่าไว้ล่วงหน้าแล้ว ตัวคอมไพเลอร์ของ Arduinoจะรู้จักกับค่าคงที่เหล่านี้แล้ว ไม่จำเป็นต้องประกาศหรือกำหนดค่าคงที่

3.10.1 HIGH,LOW : ใช้กำหนดค่าทางตรรกะ

ในการอ่านหรือเขียนค่าให้กับขาที่เป็นดิจิตอล ค่าที่เป็นได้มี 2 ค่าคือ HIGH หรือ LOW เท่านั้น

HIGH เป็นการกำหนดค่าให้ขาดิจิตอลนั้นมีแรงดันเท่ากับ +5V ในการอ่านค่า ถ้าอ่านได้ +3V หรือมากกว่าไมโครคอนโทรลเลอร์จะอ่านค่าได้เป็น HIGH ค่าคงที่ของ HIGH ก็คือ “1” หรือเทียบเป็นตรรกะคือจริง (TRUE)

LOW เป็นการกำหนดค่าให้ขาดิจิตอลนั้นมีแรงดันเท่ากับ 0V ในการอ่านค่า ถ้าอ่านได้ +2V หรือน้อยกว่าไมโครคอนโทรลเลอร์จะอ่านค่าได้เป็น LOW ค่าคงที่ของ LOW ก็คือ “0” หรือเทียบเป็นตรรกะคือเท็จ (FLSE)

3.10.2 INPUT, OUTPUT : กำหนดทิศทางของขาพอร์ตดิจิตอล

ขาของพอร์ตดิจิตอลทำหน้าที่ได้ 2 อย่างคือ เป็นอินพุต (INPUT) หรือเอาต์พุต (OUTRUT) ซึ่งค่าคงที่นี้ก็ระบุไว้ชัดเจน

3.11 ตัวกระทำอื่นๆ ที่เกี่ยวข้องกับตัวแปร

3.11.1 cast : การเปลี่ยนประเภทตัวแปรชั่วคราว

Cast เป็นตัวกระทำที่ใช้สั่งให้เปลี่ยนประเภทของตัวแปรไปเป็นประเภทอื่น และบังคับให้คำนวณค่าตัวแปรเป็นประเภทใหม่

รูปแบบคำสั่ง

(type) variable

เมื่อ Type เป็นประเภทของตัวแปรใดๆ (เช่น int, float, long)

Variable เป็นตัวแปรหรือค่าคงที่ใดๆ

ตัวอย่างที่ 3.37

```
int i;
float f;
f = 3. 6;
i = (int) f; // now i is 3
```

ในการเปลี่ยนประเภทตัวแปรจาก float เป็น int ค่าที่ได้จะถูกตัดเศษออก ดังนั้น (int)3.2 และ (int)3.7 มีค่าเท่ากันคือ 3

3.11.2 sizeof : แจ้งขนาดของตัวแปร

ใช้แจ้งบอกจำนวนไบต์ของตัวแปรที่ต้องการทราบค่า ซึ่งเป็นทั้งตัวแปรปกติและตัวแปรอะเรย์

รูปแบบคำสั่ง

เขียนได้ทั้งสองแบบดังนี้

```
sizeof (variable)
sizeof (variable)
```

เมื่อ Variable คือตัวแปรปกติหรือตัวแปรอะเรย์ (int, float, long) ที่ต้องการทราบขนาด

ตัวอย่าง 3.38

ตัวกระทำ sizeof มีประโยชน์อย่างมากในการจัดการกับตัวแปรอะเรย์ (รวมถึงตัวแปรสตริง) ตัวอย่างต่อไปนี้จะพิมพ์ข้อความออกทางพอร์ตอนุกรมครั้งละหนึ่งตัวอักษร ให้ทดลองเปลี่ยนข้อความ

```
        char myStr [ ] =”this is a test”;
        int i;
void setup ( )
        {
        Serial . begin(9600);
        {
void loop ( )
        {
         for (i = 0; I < sizeof (myStr) – 1; i++
        {
        Serial .print (i, DEC);
        Serial .print (“ = “ );
        Serial .println (myStr[i],  BYTE);
        }
        }
```

3.12 คำสงวนของ Arduino

คำสงวนคือ ค่าคงที่ ตัวแปร และฟังก์ชั่นที่กำหนดไว้เป็นส่วนหนึ่งของภาษา C ของ Arduino ห้ามนำคำเหล่านี้ไปตั้งชื่อตัวแปรแสดงได้ดังนี้

# Constants	# Port Constants	# Datatypes	# other	# other
HIGH	DDRB	boolean	+=	abs
LOW	PINB	byte	+[	acos
INPUT	PORTB	char	]	analogRead
OUTPUT	DDRC	class	=&&	analogWrite
SERIAL	PINC	default	=	attachInterrupts
DISPLAY	PORTC	do	||	asin
PI	DDRD	double	=	atan
HALF_ PI	PIND	int	,/	atan2
TWO_PI	PORTD	long	/	available
LSBFIRST		private	?:	begin
MSBFIRST		protected	<<	bit
CHANGE		public	<<	bitRead
FALLING		return	=	bitWrite
RISING		shot	log	bitSet
false		signed	&&	bitClear
true		static	!|	boolean
null		switch	|	byte
		throw	^^	case
		Try	=	ceil
		Unsigned	++	char
		Void	!=	char
		while	-	class
			%	abs
			/	acos

			*	constrain
			{	cos
			}	default
			/ /	delay
			**	delayMicroseconds
			.	detachInterrupts
			-	digitalWrite
			=	digitalRead
			==	else
			<<	exp
			=	false
			()	find
			>>	findUntil
			;	float
				floor
				for
				HALF_PI
				if
				int
				log
				loop
				map
				max
				micros
				millis
				min
				new
				noInterrupts
				noTone
				null
				parseInt
				parseFloat

				pinMode print println pulseIn radians this tone true write # USB Keyboard Mouse read press release releaseAll readBytes readBytesUntil return round serial serial1 serial2 serial3 setTimeout Setup shiftIn shiftOut sin sq sqrt tan

สรุปเนื้อหาสาระสำคัญ

โปรแกรมทำงานวนในฟังก์ชั่น loop() ตลอดเวลา หลังจากทำงานในฟังก์ชั่น setup() จึงสรุปได้ว่า ฟังก์ชั่น setup() คือส่วนต้นของโปรแกรมที่ใช้ในการประกาศ หรือตั้งค่าการทำงานในตอนเริ่มต้นทำงาน ในขณะที่ฟังก์ชั่น loop() เป็นเสมือนส่วนของโปรแกรมหลักที่ต้องวนทำงานอย่างต่อเนื่องตลอดเวลาอย่างไรก็ตามในบางโปรแกรมอาจมีเฉพาะส่วนของฟังก์ชั่น setup() และไม่มีฟังก์ชั่น loop() ก็ได้แสดงว่าโปรแกรมนั้นๆ ต้องการตั้งค่าการทำงานหรือกำหนดให้มีการทำงานเพียงครั้งหรือรอบเดียว แล้วจบการทำงานทันที เทคนิคสำหรับการเขียนโปรแกรม เลือกขนาดของตัวแปรให้ใหญ่พอสำหรับเก็บค่าผลลัพธ์ที่มากที่สุดของการคำนวณ ต้องทราบว่าที่ค่าใด ตัวแปรที่เก็บจะมีการวนซ้ำค่ากลับ และวนกลับอย่างไร สำหรับการคำณวณที่ต้องการเศษส่วนให้ใช้ตัวแปรประเภท float แต่ให้ระวังผลลบ เช่นตัวแปรมีขนาดใหญ่ คำนวณได้ช้า ใช้ตัวกระทำ cast เช่น (int)myfloat ในการเปลี่ยนประเภทของตัวแปรชั่วคราวขณะที่โปรแกรมทำงาน

.

เรื่อง โครงสร้างโปรแกรมของ Arduino 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

_____ 1. #define เป็นคำสั่งที่ใช้งานมาก ในการกำหนดค่าคงที่ให้กับโปรแกรม

_____ 2. ท้ายคำสั่ง # define ต้องมีเครื่องหมายเซมิโคลอน

_____ 3. #include ใช้สั่งให้รวมไฟล์อื่นๆ เข้ากับไฟล์โปรแกรมหลัก แล้วจึงทำการคอมไพล์โปรแกรม

_____ 4. char : ตัวแปรประเภทตัวอักขระมีขนาด 1 ไบต์ (8 บิต) มีไว้เพื่อเก็บค่าตัวอักษร

_____ 5. byte : ตัวแปรประเภทตัวเลข 8 บิตหรือ 1 ไบต์ ใช้เก็บค่าตัวเลขขนาด 8 บิต มีค่าได้จาก 0 - 255

_____ 6. int : ตัวแปรประเภทตัวเลขจำนวนเต็ม มีขนาด 1 ไบต์ เก็บค่าได้ จาก -32,768 ถึ ง 32,767

_____ 7. unsigned int : ตัวแปรประเภทเลขจำนวนเต็มไม่คิดเครื่องหมาย เก็บค่า 0 ถึง 65,535

_____ 8. long : ตัวแปรประเภทเลขจำนวนเต็ม (4 ไบต์) เก็บค่าได้จาก -2,147,483,648 ถึง 2,147,483,647

_____ 9. unsigned long : ตัวแปรประเภทเลขจำนวนเต็ม 16 บิต แบบไม่คิดเครื่องหมาย เก็บค่าได้ จาก 0 ถึง 4,294,967,295

_____10. string : ตัวแปรประเภทข้อความ เป็นตัวแปรเก็บข้อความ ซึ่งในภาษาซี จะนิยามเป็นอะเรย์ของตัวแปรประเภท char

คำชี้แจง ให้ผู้เรียนเลือกคำตอบที่ถูกที่สุดแล้วกาเครื่องหมายกากบาท (✕) ให้ครบทุกข้อ

1. การเขียนโปรแกรมสำหรับ Arduino จะอ้างอิงตามภาษา
   - ก. C/C++
   - ข. English / Thai
   - ค. Assembly Language
   - ง. Java Language
2. ฟังก์ชั่น setup() เมื่อโปรแกรมเริ่มต้นทำงานจะทำคำสั่งนี้กี่ครั้ง
   - ก. ทำคำสั่งของฟังก์ชั่นนี้ 1 ครั้ง
   - ข. ทำคำสั่งของฟังก์ชั่นนี้ 2 ครั้ง
   - ค. ทำคำสั่งของฟังก์ชั่นนี้ 3 ครั้ง
   - ง. ทำคำสั่งของฟังก์ชั่นนี้ 4 ครั้ง
3. คำสั่ง if ใช้เพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมอย่างไร
   - ก. ถ้าเงื่อนไขเป็นเท็จทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นจริงข้ามการทำงาน
   - ข. ถ้าเงื่อนไขเป็นจริงทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นเท็จข้ามการทำงาน
   - ค. ถ้าเงื่อนไขเป็นจริงทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นจริงข้ามการทำงาน
   - ง. ถ้าเงื่อนไขเป็นเท็จทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นเท็จข้ามการทำงาน
4. คำสั่ง for() แบ่งได้ 3 ส่วนอะไรบ้าง
   - ก. initialition; condition; increment
   - ข. initialization; contion; increment
   - ค. initialization; condition; crement
   - ง. initialization; condition; increment
5. ผลที่ได้จากการหารเอาเศษของ x = 5 % 5
   - ก. 1
   - ข. 0
   - ค. 0.1
   - ง. 0.5

6. ตัวกระทำเปรียบเทียบใช้สำหรับประกอบกับคำสั่งใด

ก. if() และ while()

ข. switch() และ case()

ค. for() และ while()

ง. for() และ if()

7. ตรรกะที่ใช้ในการเปรียบเทียบของคำสั่ง if() คือ

ก. && และ !

ข. || และ !

ค. &&, ||

ง. &&, || และ !

8. ตัวกระทำระดับบิต NOT ใช้สัญญลักษณ์ใด

ก. ~

ข. α

ค. β

ง. €

9. หมายเหตุบรรทัดเดียวและหลายบรรทัดคือข้อใด

ก. / / และ / * . . . */

ข. / / และ * / . . . /*

ค. / / และ */ . . . */

ง. / / และ / * . . . /*

10. ตัวแปร unsigned long เป็นตัวแปรประเภทใด

ก. ตัวแปรจำนวนเต็ม 8 บิต แบบคิดเครื่องหมาย

ข. ตัวแปรจำนวนเต็ม 16 บิต แบบคิดเครื่องหมาย

ค. ตัวแปรจำนวนเต็ม 32 บิต แบบไม่คิดเครื่องหมาย

ง. ตัวแปรจำนวนเต็ม 64 บิต แบบไม่คิดเครื่องหมาย

ปฏิบัติการทดลองหน่วยที่ 3
เรื่อง โครงสร้างโปรแกรมของ Arduino

****

<u>คำชี้แจง</u> ให้ผู้เรียนทุกคนทำการทดลองตามปฏิบัติการทดลองหน่วยที่ 3 เรื่อง โครงสร้างโปรแกรมของ Arduino โดยใช้เวลา 180 นาที (20 คะแนน)

จุดประสงค์เชิงพฤติกรรม

1. สามารถเขียนโครงสร้างโปรแกรมของ Arduino ได้ถูกต้อง
2. สามารถแก้ปัญหาในการทำงานของบอร์ด Arduino Uno R3 ได้
3. สามารถต่อใช้งานและอัพโหลดโปรแกรมของบอร์ด 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
6.	Hook-up Wires	10	เส้น
7.	Breadboard	1	แผง
8.	LED	6	ตัว
9.	Momentary Button or Pushbutton Switch	2	ตัว
10.	10K Ohm Resistor	2	ตัว
11.	220 Ohm Resistor	6	ตัว
12.	Potentiometer or Variable Resistor	1	ตัว
13.	Photoresistor or Another Analog Sensor	1	ตัว

ข้อควรระวัง

1. ควรระวังไม่วางบอร์ด Arduino Uno R3 หรือชีลต่างๆ บนโต๊ะโลหะหรือที่วางที่เป็นโลหะเพราะอาจเกิดการลัดวงจรของภาคจ่ายไฟได้

2. ไม่ควรต่อสายต่อวงจรในบอร์ด Arduino Uno R3 ทิ้งไว้ ควรถอดสายต่อวงจรออกให้หมด เพราะผลการทดลองอาจเกิดการผิดพลาดไม่เป็นไปตามทฤษฎีได้

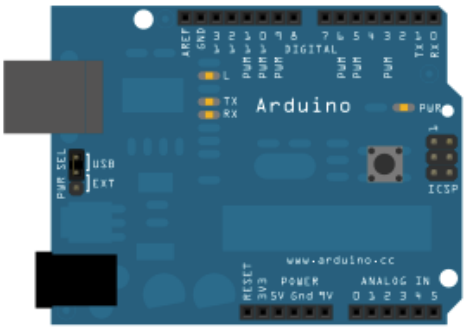
3. ไม่ควรถอดสายสายโหลด USB เข้าออกตลอดเวลา เพราะอาจทำให้ภาคจ่ายไฟของบอร์ด Arduino Uno R3 เสียหายได้

การทดลองที่ 3.1 ฟังก์ชั่น setup()

ฟังก์ชั่นนี้จะเขียนที่ส่วนต้นของโปรแกรม ทำงานเมื่อโปรแกรมเริ่มต้นเพียงครั้งเดียวใช้เพื่อกำหนดค่าของตัวแปรโหมดการทำงานของขาต่างๆ เริ่มต้นเรียกใช้ ไลบรารี ฯลฯ

Hardware Required

1. Arduino Uno Board



Code

```
void setup()
        {
        Serial.begin(9600);
        }
void loop()
        {
        int sensorValue = analogRead(A0);
        float voltage = sensorValue * (5.0 / 1023.0);
        Serial.println(voltage);
        }
// จากนั้นให้นักเรียนปรับเปลี่ยนค่า  Serial.begin แล้วสังเกตุผลที่ได้
```

ผลการทดลอง

.....

.....

.....

.....

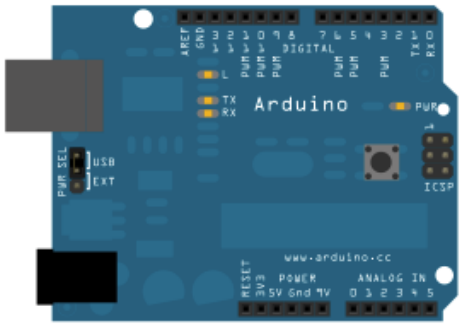
.....

การทดลองที่ 3.2 ฟังก์ชั่น loop()

หลังจากที่เขียนฟังก์ชั่น setup() ที่กำหนดค่าเริ่มต้นของโปรแกรมแล้ว ส่วนถัดมาคือฟังก์ชั่น loop() ซึ่งมีการทำงานตรงตามชื่อ คือจะทำงานตามฟังก์ชั่นนั่น วนต่อเนื่องตลอดเวลา ภายในฟังก์ชั่นนั่นจะมีโปรแกรมของผู้ใช้เพื่อรับค่าจากพอร์ต ประมวลผลแล้วสั่งเอาต์พุตออกขาต่างๆ เพื่อควบคุมการทำงานของบอร์ด

Hardware Required

1. Arduino Uno Board



Code

```
const int buttonPin = 13
void setup(){
        Serial.begin(9600);
        pinMode(buttonPin, INPUT);}
void loop(){
if (digitalRead(buttonPin) == HIGH)
        Serial.write('H');
else
        Serial.write('L');
        delay(1000);}
// จากนั้นให้นักเรียนเปลี่ยนค่า Serial.write('H');เป็น Serial.write('L');แล้วสังเกตุผลที่ได้
```

ผลการทดลอง

.....

.....

.....

.....

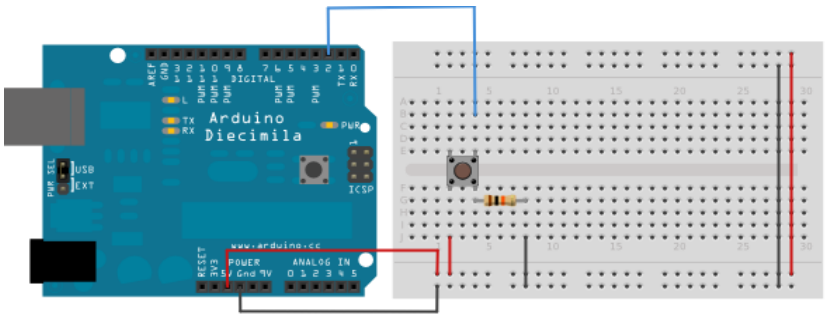
การทดลองที่ 3.3 คำสั่ง if

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรม โดยสามารถกำหนดได้ ว่าถ้าเงื่อนไขเป็นจริงให้ทำอะไร ถ้าเป็นเท็จให้ทำอะไร

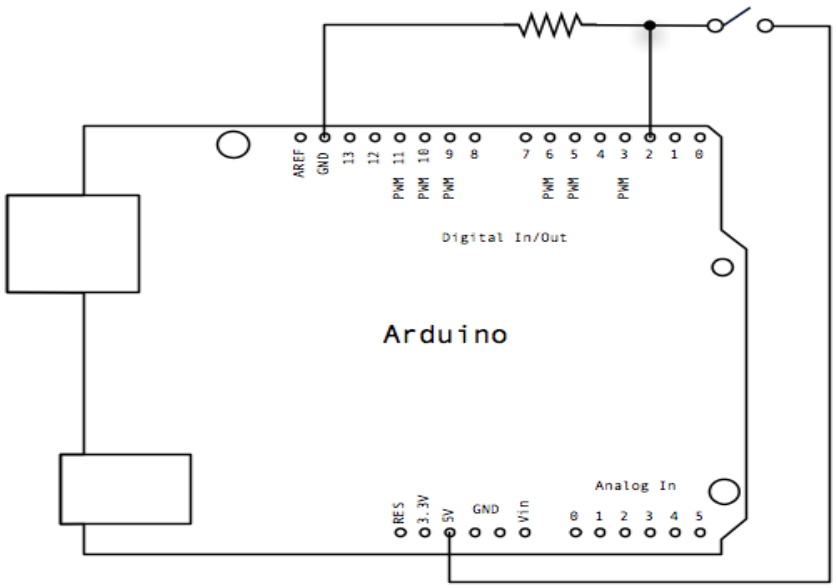
Hardware Required

1. Arduino uno Board
2. Momentary button or Switch
3. 10K ohm resistor
4. hook-up wires
5. breadboard

Circuit



Schematic



Code

```
const int buttonPin = 2;
 const int ledPin =  13;
 int buttonState = 0;
void setup()
{
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
}
void loop()
{
buttonState = digitalRead (buttonPin);
if (buttonState == HIGH)
{
digitalWrite(ledPin, HIGH);
}
//จากนั้นให้เปลี่ยนค่า if (buttonState == HIGH)เป็น if (buttonState == LOW)แล้วสังเกตุผล
```

ผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

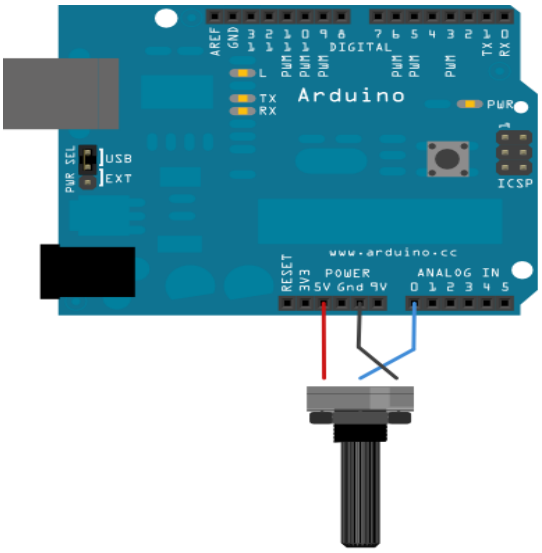
การทดลองที่ 3.4 คำสั่ง if...else

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมได้มากกว่าคำสั่ง if ธรรมดา โดยสามารถกำหนดได้ว่าถ้าเงื่อนไขเป็นจริงให้ทำอะไร ถ้าเป็นเท็จให้ทำอะไร

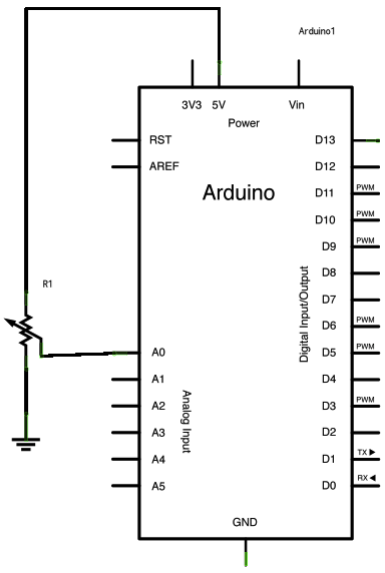
Hardware Required

1. Arduino or Genuino Board
2. Potentiometer or variable resistor

Circuit



Schematic



Code

```
const int analogPin = A0;    // pin that the sensor is attached to
const int ledPin = 13;       // pin that the LED is attached to
const int threshold = 400;   // an arbitrary threshold level that's in the range of the
analog input
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize serial communications:
  Serial.begin(9600);
}
void loop() {
  // read the value of the potentiometer:
  int analogValue = analogRead(analogPin);
  // if the analog value is high enough, turn on the LED:
  if (analogValue > threshold) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  // print the analog value:
  Serial.println(analogValue);
  delay(1);        // delay in between reads for stability
}
```

ผลการทดลอง

.....

.....

.....

.....

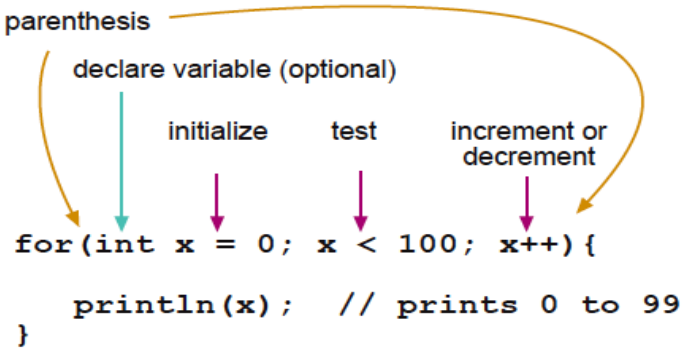
.....

.....

การทดลองที่ 3.5 คำสั่ง for()

คำสั่งนี้ใช้เพื่อสั่งให้คำสั่งที่อยู่ภายในวงเล็บปีกกาหลัง for มีการทำงานซ้ำกันตามจำนวนรอบที่ต้องการ คำสั่งนี้มีประโยชน์มากสำหรับการทำงานใดๆ ที่ต้องทำซ้ำกันและทราบจำานวนรอบของการทำงานที่แน่นอนใช้คู่กับตัวแปรอะเรย์ในการเก็บสะสมค่า ที่อ่านค่าได้จากขาอินพุตแอนะล็อกหลายๆ ขาที่มีหมายเลขขาต่อเนื่องกัน รูปแบบของคำสั่ ง for() แบ่งได้ 3 ส่วนดังนี้

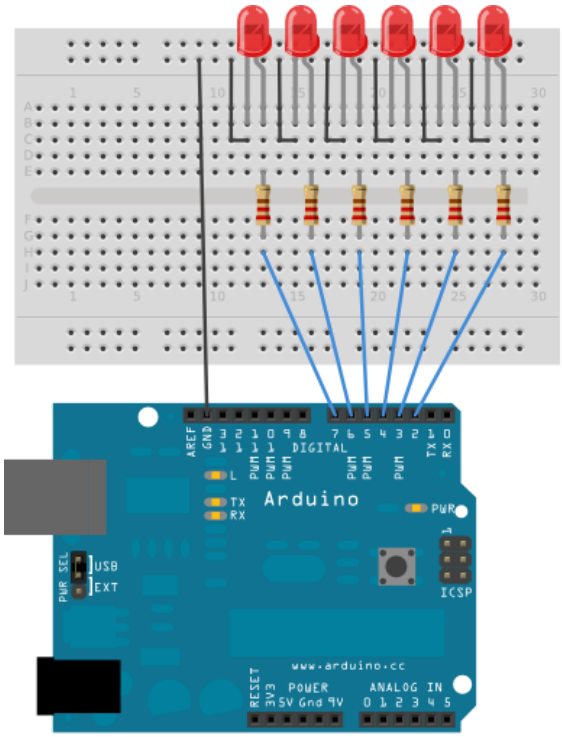
```
for (initialization; condition; increment)
{
//statement(s);
}
```



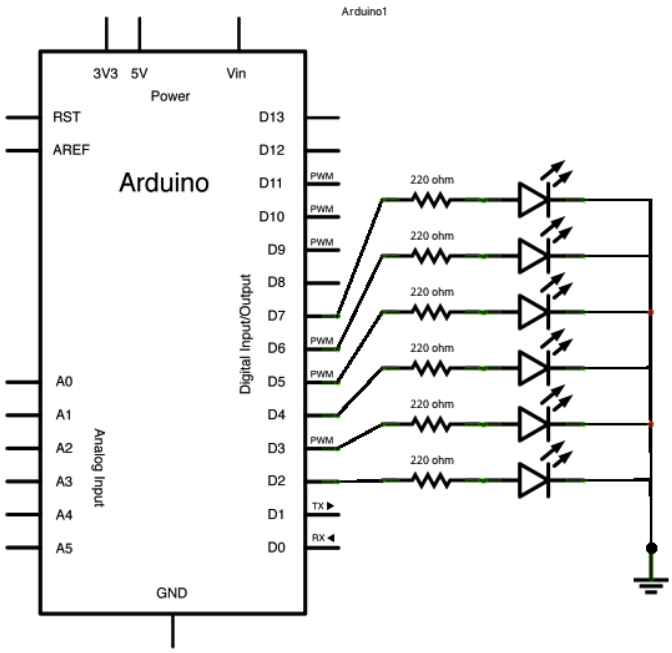
Hardware Required

1. Arduino or Genuino Board
2. 220 ohm resistors 6 PCS
3. LEDs 6 PCS
4. hook-up wires
5. breadboard

Circuit



Schematic



Code

```
int timer = 100;          // The higher the number, the slower the timing.
void setup() {
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}
void loop() {
  // loop from the lowest pin to the highest:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }
  // loop from the highest pin to the lowest:
  for (int thisPin = 7; thisPin >= 2; thisPin--) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }
}
}
```

ผลการทดลอง

.....

.....

.....

.....

การทดลองที่ 3.6 คำสั่ง switch-case

ใช้ทดสอบเงื่อนไขเพื่อกำหนดการทำงานของโปรแกรม ถ้าตัวแปรที่ทดสอบตรงกับเงื่อนไขใดก็ให้ทำงานตามที่กำหนดไว้พารามิเตอร์

var ตัวแปรที่ต้ องการทดสอบว่าตรงกับเงื่อนไขใด

default ถ้าไม่ตรงกับเงื่อนไขใดๆ เลยให้ทำคำสั่งต่อท้ายนี้

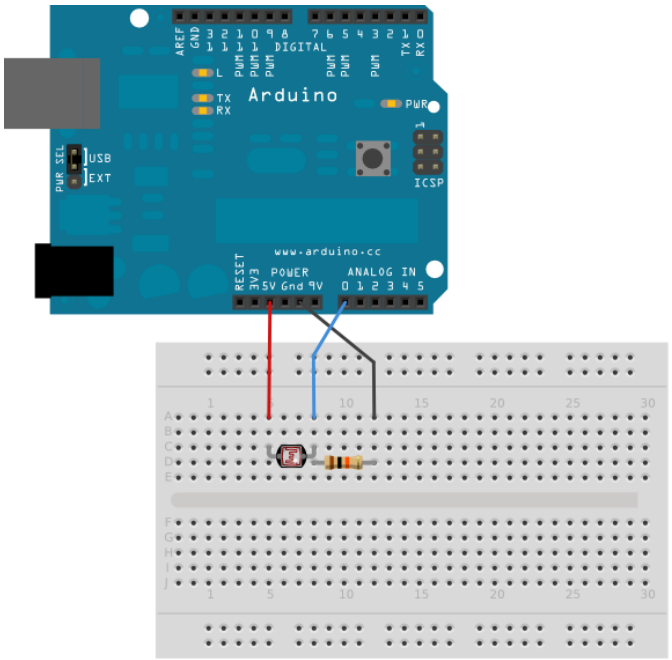
break เป็นส่วนสำคัญมากใช้เขียนต่อท้าย case ต่างๆ เมื่อพบเงื่อนไขนั้นแล้วทำตามคำสั่งต่างๆแล้วให้หยุดการทำงานของคำสั่ ง

switch-case ถ้าลืมเขียน break เมื่อพบเงื่อนไขทำตามเงื่อนไขแล้วโปรแกรมจะทำงานตามเงื่อนไขต่อไปเรื่อยๆ จนกว่าจะพบคำสั่ง break

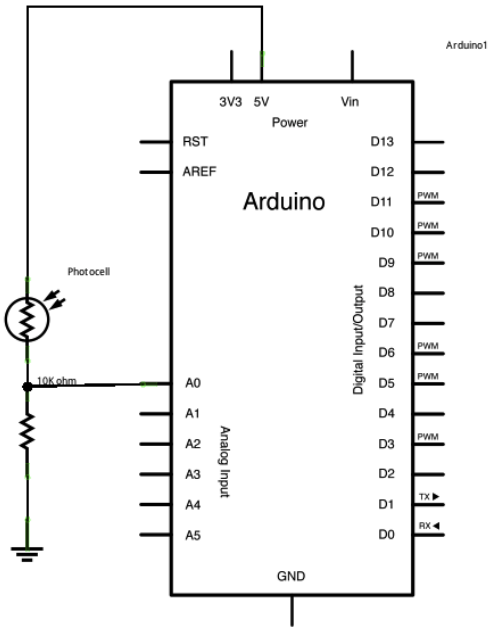
Hardware Required

1. Arduino or Genuino Board
2. photoresistor, or another analog sensor
3. 10k ohm resistors
4. hook-up wires
5. breadboar

Circuit



Schematic



Code

```
const int sensorMin = 0;      // sensor minimum, discovered through experiment
const int sensorMax = 600;    // sensor maximum, discovered through experiment
void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}

void loop() {
  // read the sensor:
  int sensorReading = analogRead(A0);
  // map the sensor range to a range of four options:
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // do something different depending on the
  // range value:
  switch (range) {
    case 0:    // your hand is on the sensor
```

```
      Serial.println("dark");
      break;
    case 1:    // your hand is close to the sensor
      Serial.println("dim");
      break;
    case 2:    // your hand is a few inches from the sensor
      Serial.println("medium");
      break;
    case 3:    // your hand is nowhere near the sensor
      Serial.println("bright");
      break;
  }
  delay(1);        // delay in between reads for stability
}
```

ผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

การทดลองที่ 3.7 คำสั่ง while

เป็นคำสั่งวนรอบโดยจะทำคำสั่งที่เขียนในวงเล็บปีกกาอย่างต่อเนื่อง จนกว่าเงื่อนที่เขียนในวงเล็บของคำสั่ง while() จะเป็นเท็จคำสั่งที่ให้ทำซ้ำจะต้องมีการเปลี่ยนแปลงค่าตัวแปรที่ใช้ทดสอบ เช่น มีการเพิ่มค่าตัวแปร หรือมีเงื่อนไขภายนอก เช่นอ่านค่าจากตัวตรวจจับได้เรี ยบร้อยแล้วให้หยุดการอ่านค่า มิฉะนั้นเงื่อนไขในวงเล็บของ while() เป็นจริงตลอดเวลา ทำให้คำสั่ ง while ทำงานวนรอบไปเรื่อยๆ ไม่รู้จบ รูปแบบคำสั่ ง

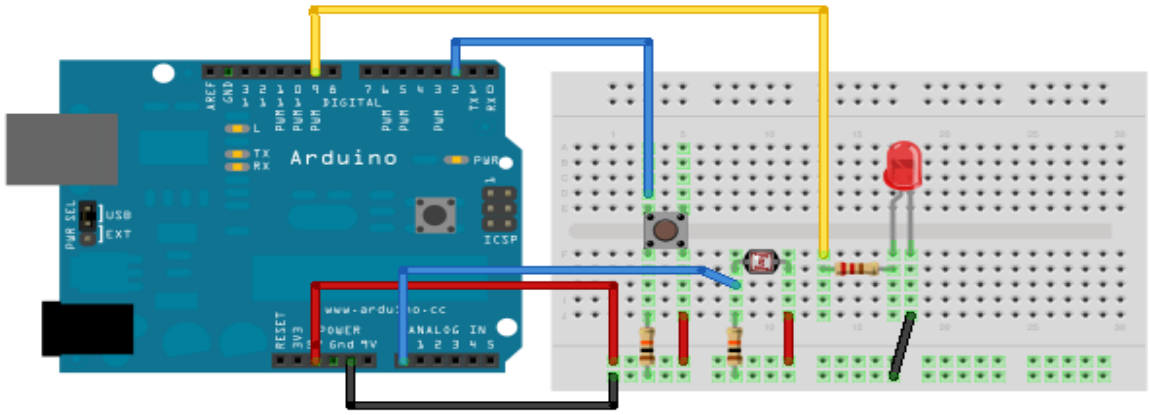
```
while(expression)
{
// statement(s)
}
```

พารามิเตอร์ expression เป็นคำสั่งทดสอบเงื่อนไข (ถูกหรือผิด)

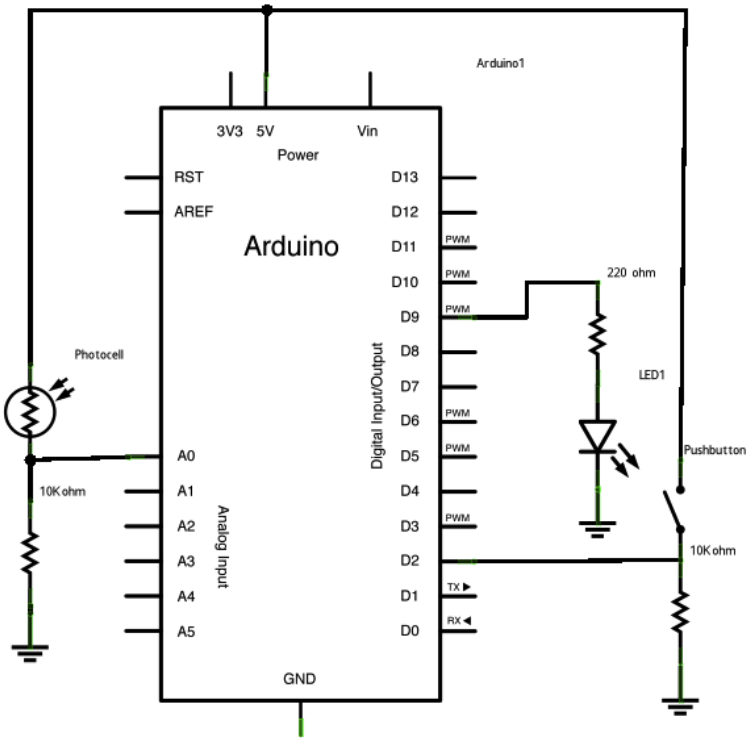
Hardware Required

1. Arduino or Genuino Board
2. pushbutton or switch
3. photoresistor or another analog sensor
4. 10k ohm resistors 2 PCS
5. Breadboard

Circuit



Schematic



Code

```
const int sensorPin = A2;      // pin that the sensor is attached to
const int ledPin = 9;          // pin that the LED is attached to
const int indicatorLedPin = 13; // pin that the built-in LED is attached to
const int buttonPin = 2;       // pin that the button is attached to
// These variables will change:
int sensorMin = 1023;  // minimum sensor value
int sensorMax = 0;     // maximum sensor value
int sensorValue = 0;         // the sensor value
void setup() {
  // set the LED pins as outputs and the switch pin as input:
  pinMode(indicatorLedPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop() {
```

```
  // while the button is pressed, take calibration readings:
  while (digitalRead(buttonPin) == HIGH) {
    calibrate();
  }
  // signal the end of the calibration period
  digitalWrite(indicatorLedPin, LOW);
  // read the sensor:
  sensorValue = analogRead(sensorPin);
  // apply the calibration to the sensor reading
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);
  // in case the sensor value is outside the range seen during calibration
  sensorValue = constrain(sensorValue, 0, 255);
  // fade the LED using the calibrated value:
  analogWrite(ledPin, sensorValue);
}
void calibrate() {
  // turn on the indicator LED to indicate that calibration is happening:
  digitalWrite(indicatorLedPin, HIGH);
  // read the sensor:
  sensorValue = analogRead(sensorPin);
  // record the maximum sensor value
  if (sensorValue > sensorMax) {
    sensorMax = sensorValue;
  }
  // record the minimum sensor value
  if (sensorValue < sensorMin) {
    sensorMin = sensorValue;
  }
}
```

ผลการทดลอง

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ปัญหาอุปสรรคหรือข้อเสนอแนะ

.....

.....

ตารางการประเมินผลคะแนนภาคปฏิบัติ

หัวข้อการพิจารณาภาคปฏิบัติ	ระดับคะแนน
การทดลองที่ 3.1 ฟังก์ชั่น setup()	1 คะแนน
การทดลองที่ 3.2 ฟังก์ชั่น loop()	1 คะแนน
การทดลองที่ 3.3 คำสั่ง if	2 คะแนน
การทดลองที่ 3.4 คำสั่ง if...else	4 คะแนน
การทดลองที่ 3.5 คำสั่ง for()	4 คะแนน
การทดลองที่ 3.6 คำสั่ง switch-case	4 คะแนน
การทดลองที่ 3.7 คำสั่ง while	4 คะแนน
รวมคะแนนภาคปฏิบัติ	.....คะแนน

แบบทดสอบหลังเรียน หน่วยที่ 3
เรื่อง โครงสร้างโปรแกรมของ Arduino

เรื่อง โครงสร้างโปรแกรมของ Arduino **ใช้เวลา 20 นาที**

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

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

****

<u>คำชี้แจง</u> **1. แบบทดสอบมีทั้งหมด 10 ข้อ (10 คะแนน)**

2. ให้ผู้เรียนเลือกคำตอบที่ถูกที่สุดแล้วกาเครื่องหมายกากบาท (✕) ลงในกระดาษคำตอบ

1. ภาษาของ Arduino แบ่งได้เป็น 2 ส่วนหลักคือ
   - ก. ตัวแปร / ไวยกรณ์
   - ข. โครงสร้างภาษา / ฟังก์ชั่น
   - ค. ตัวกระทำ / ตัวเปรียบเทียบ
   - ง. ฟังก์ชั่น / ตัวกระทำ
2. โครงสร้างโปรแกรมของ Arduino แบ่งได้เป็นสองส่วนคือ
   - ก. void set() และ void setup()
   - ข. void up() และ void loop()
   - ค. void setup() และ void loop()
   - ง. void setup () และ void looping()
3. ฟังก์ชั่น loop() ซึ่งมีการทำงานแบบใด
   - ก. ทำงานแบบวนรอบ 5 ครั้ง
   - ข. ทำงานแบบวนรอบ 10 ครั้ง
   - ค. ทำงานวนแบบวนรอบ 15 ครั้ง
   - ง. ทำงานแบบวนรอบต่อเนื่องตลอดเวลา
4. คำสั่ง if...else ใช้เพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมอย่างไร
   - ก. ไม่สามารถทำอะไรได้อีกแล้ว
   - ข. ถ้าเงื่อนไขเป็นจริงให้ทำอะไร
   - ค. ถ้าเงื่อนไขเป็นเท็จให้ทำอะไร
   - ง. ถ้าเงื่อนไขเป็นจริงให้ทำอะไร ถ้าเป็นเท็จให้ทำอะไร

5. ตัวกระทำทางคณิตศาสตร์ ประกอบด้วย

ก. + (บวก),- (ลบ), และ % (หารเอาเศษ)

ข. + (บวก),- (ลบ), * (คู ณ), / (หาร) และ % (หารเอาเศษ)

ค. + (บวก),- (ลบ), * (คู ณ), / (หาร) และ % (หารไม่เอาเศษ)

ง. * (คูณ), / (หาร) และ % (หารเอาเศษ)

6. ตัวกระทำระดับบิต AND ใช้สัญญลักษณ์ใด

ก. AND

ข. &&

ค. &

ง. และ

7. ตัวกระทำระดับบิต OR ใช้สัญญลักษณ์ใด

ก. OR

ข. I

ค. II

ง. หรือ

8. คำสั่งระดับบิต Exclusive OR ใช้สัญญลักษณ์ใด

ก. $

ข. ^

ค. ∞

ง. f

9. ไวยากรณ์ภาษา C เซมิโคลอน (semicolon ;) ใช้ทำหน้าที่ใด

ก. ใช้เขียนแจ้งว่าเริ่มคำสั่ง

ข. ใช้เขียนแจ้งว่าจบคำสั่ง

ค. ใช้เขียนแจ้งว่าเพิ่มคำสั่ง

ง. ใช้เขียนแจ้งว่าลบคำสั่ง

10. ตัวแปรประเภทเลขทศนิยมคือ

ก. float ใช้พื้นที่หน่วยความจำ 2 ไบต์

ข. float ใช้พื้นที่หน่วยความจำ 4 ไบต์

ค. long ใช้พื้นที่หน่วยความจำ 2 ไบต์

ง. long ใช้พื้นที่หน่วยความจำ 4 ไบต์