หน่วยที่ 3

ซอฟต์แวร์ควบคุมหุ่นยนต์

หัวข้อเรื่อง

3.1.โครงสร้างโปรแกรมภาษา C สำหรับ Arduino

3.2 ตัวแปร

3.3 คำสั่งในภาษา Arduino

3.4 ส่วนของตัวกระทำทางคณิตศาสตร์

3.5 ส่วนของตัวกระทำเปรียบเทียบ

3.6 ส่วนของตัวกระทำระดับบิต

3.7 ไวยากรณ์ภาษา C/C++ ของ Arduino

3.8 ค่าคงที่ (constants)

3.9 คำสงวนของ Arduino

สาระสำคัญ

ภาษาของ Arduino จะอ้างอิงตามภาษา C/C++ จึงอาจกล่าวได้ว่าการเขียนโปรแกรมสำหรับ Arduino ก็คือการเขียนโปรแกรมภาษา C โดยเรียกใช้ฟังก์ชั่นและไลบรารี่ที่ทาง Arduino เตรียมไว้ให้แล้ว ซึ่งสะดวกและทำให้ผู้ที่ไม่มีความรู้ด้านไมโครคอนโทรลเลอร์อย่างลึกซึ้งสามารถเขียนโปรแกรมสั่งงานได้

โดยฟังก์ชั่น setup() เมื่อโปรแกรมทำงานจะทำคำสั่งของฟังก์ชั่นนี้เพียงครั้งเดียว ใช้ในการกำหนดค่าเริ่มต้นของการทำงาน ส่วนฟังก์ชั่น Loop() เป็นส่วนทำงานโปรแกรมจะทำคำสั่งในฟังก์ชั่นต่อเนื่องกันตลอดเวลาโดยปกติใช้กำหนดโหมดการทำงานของขาต่างๆกำหนดการสื่อสารแบบอนุกรม ฯลฯ

ส่วนของ LOOP() เป็นโค๊ดโปรแกรมที่ทำงาน เช่น อ่านค่าอินพุต ประมวลผล สั่งงานเอาต์พุต ฯลฯ โดยส่วนกำหนดค่าเริ่มต้น เช่นตัวแปรจะต้องเขียนที่ส่วนหัวของโปรแกรมก่อนถึงฟังก์ชั่น นอกจากนั้นยังต้องคำนึงถึงตัวพิมพ์เล็ก–ใหญ่ของตัวแปรและชื่อฟังก์ชั่นนั้นให้ถูกต้อง

สมรรถนะประจำหน่วยการเรียนรู้

แสดงความรู้การใช้ซอฟต์แวร์ควบคุมหุ่นยนต์

จุดประสงค์การเรียนรู้

จุดประสงค์ทั่วไป

1. เพื่อให้มีความรู้หลักการเขียนโปรแกรมภาษาซี สำหรับ Arduino
2. เพื่อให้มีความรู้ความเข้าใจในการใช้คำสั่งควบคุมในภาษาซีสำหรับ Arduino
3. เพื่อให้สามารถนำความรู้ไปประยุกต์ใช้ควบคุมหุ่นยนต์อุตสาหกรรม
4. เพื่อให้ตระหนักถึงความสำคัญของคำสั่งควบคุมหุ่นยนต์อุตสาหกรรม
5. เพื่อให้มีคุณธรรม จริยธรรม ค่านิยมและคุณลักษณะอันพึงประสงค์

จุดประสงค์เชิงพฤติกรรม

1. อธิบายการเขียนโปรแกรมภาษาซีสำหรับ Arduino
2. อธิบายหลักการใช้ตัวแปรในการเขียนโปรแกรมภาษาซีสำหรับ Arduino
3. อธิบายการใช้คำสั่งในการเขียนโปรแกรมภาษาซีสำหรับ Arduino
4. อธิบายการใช้ตัวกระทำทางคณิตศาสตร์ในการเขียนโปรแกรมภาษาซีสำหรับ Arduino
5. อธิบายการใช้ตัวกระทำเปรียบเทียบในการเขียนโปรแกรมภาษาซีสำหรับ Arduino
6. อธิบายการใช้ตัวกระทำระดับบิตในการเขียนโปรแกรมภาษาซีสำหรับ Arduino
7. เพื่อให้มีคุณธรรม จริยธรรม ค่านิยมและคุณลักษณะอันพึงประสงค์

แบบทดสอบก่อนเรียน

หน่วยที่ 3 ซอฟต์แวร์ควบคุมหุ่นยนต์

<u>คำชี้แจง</u> 1. อ่านคำถามต่อไปนี้แล้วทำเครื่องหมายกากบาท (X) ข้อที่ถูกที่สุดลงในกระดาษคำตอบเพียงข้อเดียว

2. เวลาสำหรับทำแบทดสอบ 10 นาที

****

ข้อที่ 1. ภาษาของ Arduino แบ่งได้เป็น 2 ส่วนหลักคือ

ก. โครงสร้างภาษา/ฟังก์ชั่น

ข. ตัวแปร/ไวยกรณ์

ค. ฟังก์ชั่น/ตัวกระทำ

ง. ตัวกระทำ/ ตัวเปรียบเทียบ

จ. ตัวเปรียบเทียบ/ตัวแปร

ข้อที่ 2. โครงสร้างโปรแกรมของ Arduino แบ่งได้เป็น 2 ส่วนคือ

ก. void up() และ void loop()

ข. Void set() และ void setup()

ค. void setup() และ void reset()

ง. Void setup() และ void loop()

จ. Void upload() และ void reload()

ข้อที่ 3. ฟังชั่น loop() ซึ่งมีการทำงานอย่างไร

ก. ทำงานแบบวนรอบต่อเนื่องตลอดเวลา

ข. ทำงานแบบรอบเดียว

ค. ทำงานแบบวนรอบ 5 รอบ

ง. ทำงานแบบวนรอบ 15 รอบ

จ. ทำงานแบบวนรอบครบ 1 นาที

ข้อที่ 4. คำสั่ง if...else ใช้เพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมอย่างไร

ก. ถ้าเงื่อนไขเป็นจริงให้ทำอะไร และ ถ้าเงื่อนไขเป็นเท็จให้ทำอะไร

ข. เงื่อนไขเป็นจริงให้ทำอะไร

ค. เงื่อนไขเป็นเท็จให้ทำอะไร

ง. เงื่อนไขให้ทำอะไรก็ได้ ไม่มีลิมิต

จ. ไม่สามารถทำตามเงื่อนใดๆได้เลย

ข้อที่ 5. ตัวแปรกระทำทางคณิตศาสตร์ประกอบด้วย

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

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

ค. +(บวก),-(ลบ),*(คูณ),/(หาร)และ -+ (บวกลบเอาเศษ)

ง. +(บวก),-(ลบ),*(คูณ),/(หาร)และ -+ (บวกลบไม่เอาเศษ)

จ. +(บวก),-(ลบ),*(คูณ),/(หาร)

ข้อที่ 6. float เป็นตัวแปรประเภทใด

ก. เลขจำนวนเต็ม

ข. เลขทศนิยม

ค. ตัวอักษร

ง. บูลลีน

จ. ข้อความ

ข้อที่ 7. คำสั่งใช้ทดสอบเงื่อนไขเพื่อกำหนดการทำงานของโปรแกรม ถ้าตัวแปรที่ทดสอบตรงกับเงื่อนไขใดก็ให้ทำงานตามที่กำหนดไว้ในพารามิเตอร์คือคำสั่งข้อใด

ก. คำสั่ง switch – case

ข. คำสั่ง if..else

ค. คำสั่ง if...else if

ง. คำสั่ง for

จ. คำสั่ง while

ข้อที่ 8. คำสั่งระดับบิต OR ของภาษา C เขียนได้โดยใช้เครื่องหมายใด

ก.&&

ข.|

ค. %%

ง. .

จ. *$*

ข้อที่ 9. คำสั่งใดมีการทำงานซ้ำกันตามจำนวนรอบที่ต้องการคำสั่งนี้มีประโยชน์มากสำหรับการทำงานใดๆที่ต้องทำซ้ำกันและทราบจำนวนรอบของการทำงานที่แน่นอน

ก. คำสั่ง switch – case

ข. คำสั่ง if..else

ค. คำสั่ง if...else if

ง. คำสั่ง for

จ. คำสั่ง while

ข้อที่ 10. ตัวแปรใดใช้สำหรับเก็บค่าตัวเลข ตัวแปรหนึ่งตัวมีขนาด 2 ไบต์เก็บค่าได้จาก -32,768 ถึง 32,767

ก. เลขจำนวนเต็ม

ข. เลขทศนิยม

ค. ตัวอักษร

ง. บูลลีน

จ. ข้อความ

หน่วยที่ 3

ซอฟต์แวร์ควบคุมหุ่นยนต์

3.1 โครงสร้างโปรแกรมภาษา C สำหรับ Arduino

3.1.1 ปรีโปรเซสเซอร์ไดเร็กทีฟ (Preprocessor directives)

โดยส่วนนี้จะเป็นส่วนที่คอมไพลเลอร์จะมีการประมวลผลและทำตามคำสั่งก่อนที่จะมีการคอมไฟล์โปรแกรมซึ่งจะเริ่มต้นด้วยเครื่องหมายไดเร็กทีฟ (directive) หรือเครื่องหมายสี่เหลี่ยม # แล้วจึงตามด้วยชื่อที่ต้องการเรียกใช้ หรือกำหนด โดยปกติแล้วส่วนนี้จะอยู่ในส่วนบนสุด หรือส่วนหัวของโปรแกรมและต้องอยู่นอกของฟังก์ชั่นหลักใดๆก็ตาม

#include เป็นคำสั่งที่ใช้อ้างอิงไฟล์ภายนอก เพื่อเรียกใช้ฟังก์ชั่นหรือตัวแปรที่มีการส้รางหรือกำหนดไว้ในไฟล์นั้น รูปแบบการใช้งานคือ

#include<ชื่อไฟล์.h>

ตัวอย่างที่ 3.1

```
#include<Wire.h>
#include<Time.h>
```

จากตัวอย่างจะเห็นว่าได้มีการอ้างอิงไฟล์ Wire.h และไฟล์ Time.h ซึ่งเป็นไลบรารี่พื้นฐานที่มีอยู่ใน Arduino ทำให้เราสามารถใช้ฟังก์ชั่นเกี่ยวกับเวลาที่ไลบรารี่ Time มีการสร้างไว้ให้ใช้งานได้

การอ้างอิงไฟล์จากภายในหรือการอ้างอิงไฟล์ไลบารี่ที่มีอยู่แล้วใน Arduino หรือเป็นไลบารี่ที่เราเพิ่มเข้าไปเอง จะใช้เครื่องหมาย<> ในการคร่อมชื่อไฟล์ไว้ เพื่อให้โปรแกรมคอมไพลเลอร์เข้าใจว่าควรไปหาไฟล์เหล่านี้จากในโฟลเดอร์ไลบารี่ แต่หากต้องการอ้างอิงไฟล์ที่อยู่ในโฟลเดอร์โปรเจคจะต้องใช้เครื่องหมาย “ ” คร่อมแทน ซึ่งคอมไพล์เลอร์จะวิ่งไปหาไฟล์นี้โดยอ้างอิงจากโฟล์โปรแกรมที่คอมไพล์อยู่

เช่น

```
#include “myFunction.h”
```

จากตัวอย่างด้านบน คอมไพล์เลอร์จะวิ่งไปหาไฟล์ myFunction.h ภายในโฟลเดอร์โปรเจคทันที หากไม่พบก็จะแจ้งเป็นข้อผิดพลาดออกมา

#define เป็นคำสั่งที่ใช้ในการแทนข้อความที่กำหนดไว้ด้วยข้อความที่กำหนดไว้ซึ่งการใช้คำสั่งนี้ข้อดีคือจะไม่มีการอ้างอิงกับตัวโปรแกรมเลย

รูปแบบ

#define NAME VALUE

ตัวอย่างที่ 3.2

#define LEDPIN 13

จากตัวอย่างไม่ว่าคำว่า LEDPIN จะอยู่ส่วนใดของโค้ดโปรแกรมก็ตามคอมไพล์เลอร์จะแทนคำว่า LEDPIN ด้วยเลข 13 แทน ซึ่งข้อดีคือเราไม่ต้องสร้างตัวแปรขึ้นมาเพื่อเปลืองพื้นที่แรมและยังช่วยให้โปรแกรมทำงานเร็วขึ้นอีกด้วย

3.1.2 ส่วนของการกำหนดค่า (Global declarations)

ส่วนนี้จะเป็นส่วนที่ใช้ในการกำหนดชนิดตัวแปรแบบนอกฟังก์ชั่น หรือประกาศฟังก์ชั่นเพื่อให้ฟังก์ชั่นที่ประกาศสามารถกำหนด หรือเรียกใช้ได้จากทุกส่วนของโปรแกรม

เช่น int pin = 13;

Void blink(void);

3.1.3 ฟังก์ชั่น setup() และฟังก์ชั่น Loop()

ฟังก์ชั่น setup() และฟังก์ชั่น loop() เป็นคำสั่งที่ถูกบังคับให้ต้องมีในทุกโปรแกรม โดยฟังก์ชั่น setup() จะเป็นฟังก์ชั่นแรกที่ถูกเรียกใช้ นิยมใช้กำหนดค่า หรือเริ่มต้นใช้งานไลบารี่ต่างๆเช่นในฟังก์ชั่น setup() จะมีคำสั่ง pinMode() เพื่อกำหนดให้ขาใดๆ ก็ตามเป็นดิจิตอลอินพุต หรือเอาต์พุต ส่วนฟังก์ชั่น Loop() จะเป็นฟังก์ชั่นที่ทำงานหลังจากฟังก์ชั่น setup() ได้ทำงานเสร็จสิ้นไปแล้ว และมีการวนรอบแบบไม่รู้จบ เมื่อฟังก์ชั่น loop() งานครบตามคำสั่งแล้ว ฟังก์ชั่นloop() ก็จะถูกเรียกขึ้นมาใช้อีก

ตัวอย่างที่ 3.3

```
int pin = 13
void setup(){
    pinMode(pin,OUTPUT);
   }
 void loop(){
    digital(pin,HIGH);
```

```
    delay(1000);
    digitalWrite(pin,LOW);
    delay(1000);
}
```

จากตัวอย่างที่ 3.3 จะเห็นว่ามีการประกาศตัวแปรแบบนอกฟังก์ชั่น ทำให้สามารถกำหนดหรือเรียกใช้จากในฟังก์ชั่นใดๆก็ตามได้ ในฟังก์ชั่น setup () ได้มีการกำหนดให้ขาที่ 13 เป็นดิจิตอลเอาต์พุต และในฟังก์ชั่นเลา 1 วินาที แล้วจึงกำหนดให้เอาต์พุต 13 มีสถานะลอจิกเป็น 0 แล้วจึงหน่วงเวลา 1 วินาที จบฟังก์ชั่น loop() และจะเริ่มทำฟังก์ชั่น loop() ใหม่ ผลที่ได้คือไฟกระพริบบนบอร์ด Arduino Uno ในพอร์ต 13 ทำงานแบบไม่รู้จบ

3.1.4 การสร้างฟังก์ชั่นและการใช้งานฟังก์ชั่น (Users-defined function)

ในการสร้างฟังก์ชั่นขึ้นมา คำสั่งต่างๆที่อยู่ภายในฟังก์ชั่นต้องอยู่ภายใต้เครื่องหมายปีกกาเปิด{ และปีกกาปิด }เท่านั้น ภายใต้เครื่องหมาย {} เราสมารถนำฟังก์ชั่นหรือคำสั่งใดๆก็ได้มาใส่ไว้ แต่จะต้องคั่นด้วยเครื่องหมายเซมิโคล่อน ; โดยจะนำคำสั่งทั้งหมดไว้บรรทัดเดียวกันเลย หรือแยกบรรทัดกันก็ได้เพื่อความสวยงามของโค้ด (ไม่มีผลกับขนาดของโปรแกรมหลังคอมไพล์)

ตัวอย่างที่ 3.4

```
void  Mode(int pin) {
    pinMode(pin,OUTPUT);
    }
void setup(){
    Mode(13);
}
```

3.1.5 ส่วนอธิบายโปรแกรม (Program comments)

ส่วนอธิบายโปรแกรม หรือการคอมเม้นโปรแกรมเป็นส่วนที่สำคัญอย่างมากที่จะช่วยให้ผู้ที่ไม่ได้เขียนโปรแกรม หรือเป็นผู้เขียนโปรแกรมเข้าใจโปรแกรมได้ง่ายขึ้นโดยอ่านจากคอมเม้น แทนการทำความเข้าใจโปรแกรมโดยอ่านแต่ละฟังก์ชั่น ส่วนอธิบายโปรแกรมหรือส่วนคอมเม้นนี้จะไม่มีผลใดๆกับขนาดของโปรแกรมหลังคอมไพล์ เนื่องจากส่วนนี้จะถูกตัดทิ้งทั้งหมดเนื่องจากไม่ได้ถูกนำไปใช้งาน มีผล

เพียงแค่ว่าไฟล์โค้ดโปรแกรมจะใหญ่ขึ้นมา หากมีการคอมเม้นโค้ดเยอะๆแต่ขนาดก็จะเพิ่มขึ้นตามตัวอักษร ดังนั้นการคอมเม้นโค้ดจึงไม่คิดพื้นที่มากนัก

```
/*
 This code.....
 17/6/2559
*/
void setup(){....}
```

และแบบที่ 2 เป็นการคอมเม้นบรรทัดเดียว คือเปิดด้วยเครื่องหมาย // และปิดด้วยการขึ้นบรรทัดใหม่ เช่น

```
void setup() {
pinMode(13,OUTPUT);//set pin to OUTPUT
 }
```

3.2 ตัวแปร

ตัวแปรเป็นตัวอักษรหลายๆตัวที่กำหนดขึ้นในโปรแกรมเพื่อใช้ในโปรแกรมเพื่อใช้ในการเก็บข้อมูลต่างๆเช่น ค่าที่อ่านได้จากตัวตรวจจับที่ต่อกับขาพอร์ตแอนะล็อกของ Arduino ตัวแปรมีหลายประเภทดังนี้

3.2.1 char : ตัวแปรประเภทตัวอักขระ

เป็นตัวแปรที่มีขนาด 1 ไบต์(8 บิต) มีไว้เพื่อเก็บค่าตัวอักษรในภาษาซีจะเขียนอยู่ในเครื่องหมายคำพูดขีดเดียวเช่น ‘A’ (สำหรับข้อความที่ประกอบจากตัวอักษรหลายตัวเขียนต่อกันจะเขียนอยู่ในเครื่องหมายคำพูดปกติเช่น “ABC”) สามารถสั่งกระทำทางคณิตศาสตร์กับตัวอักษรได้ในกรณีจะนำค่ารหัส ASCII ของตัวอักษร A มีค่าเท่ากับ 65

รูปแบบคำสั่ง

```
Charsign = ‘  ’ ;
```

พารามิเตอร์

char var คือชื่อของตัวแปรประเภท char ที่ต้องการ

var คือชื่อของตัวแปรประเภท char ที่ต้องการ

x คือค่าที่ต้องการกำหนดให้กับตัวแปร ในที่นี้เป็นตัวอักษรหนึ่งตัว

3.2.2 byte: ตัวแปรประเภทตัวเลข 8 บิตหรือ 1 ไบต์

ตัวแปร byte ใช้เก็บค่าตัวเลขขนาด 8 บิต มีค่าได้จาก 0-255

ตัวอย่างที่ 3.5

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

3.2.3 int : ตัวแปรประเภทตัวเลขจำนวนเต็ม

ย่อมาจาก interger ซึ่งแปลว่าเลขจำนวนเต็ม int เป็นตัวแปรพื้นฐานสำหรับเก็บค่าตัวเลข ตัวแปรหนึ่งตัวมีขนาด 2 ไบต์เก็บค่าได้จาก -32,768 ถึง 32,767 ในการเก็บค่าตัวเลขติดลบ จะใช้เทคนิคที่เรียกว่าทูคอมพลีเมนต์ (2’s complement) บิตสูงสุดบางที่เรียกว่าเป็นบิตเครื่องหมายหรือ sign bit ถ้ามีค่าเป็น “1” แสดงว่าค่าติดลบใน Arduino จะจัดการกับตัวเลขค่าติดลบให้เอง ทำให้นำค่าตัวแปรไปคำนวณได้อย่างถูกต้อง อย่างไรก็ตามเมื่อนำตัวเลขค่าติดลบนี้เลื่อนบิตไปทางขวา(>>)จะมีปัญหาเรื่องค่าของตัวเลขที่ผิดพลาด

รูปแบบคำสั่ง

```
int var =val;
```

พารามิเตอร์

var คือชื่อของตัวแปรประเภท int ที่ต้องการ

val คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.6

```
Int ledPin=31;
```

เทคนิคสำหรับการเขียนโปรแกรม

เมื่อตัวแปรมีค่ามากกว่าค่าสูงสุดที่เก็บไว้จะเกิดการ “ล้นกลับ” (Roll Over) ไปยังค่าต่ำสุดที่เก็บได้และเมื่อมีค่าน้อยกว่าค่าต่ำสุดที่เก็บได้ จะล้นกลับไปยังค่าสูงสุด ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.7

```
Unsigned int x
x=0;
x=x-1;// x now contains 65535
x=x+1;//x now contains 0-roll over
```

3.2.4 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 คือชื่อของตัวแปรเต็ม long ที่ต้องการ

Val คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.8

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

3.2.5 unsigned long : ตัวแปรประเภทเลขจำนวนเต็ม 32 บิต แบบไม่คิดเครื่องหมาย

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

รูปแบบคำสั่ง

Unsigned long var =val;

พารามิเตอร์

Var คือชื่อของตัวแปร unsigned long ที่ต้องการ

Var คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.9

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

3.2.6 float : ตัวแปรประเภทเลขทศนิยม

เป็นตัวแปรสำหรับเก็บค่าเลขทศนิยม นิยมใช้เก็บค่าสัญญาณแอนะล็อกหรือค่าที่ต่อเนื่อง ตัวแปรแบบนี้เก็บค่าได้ละเอียดกว่าตัวแปร int โดยเก็บค่าได้ในช่วง 3.4028235x1038 ถึง 3.4028235x1038 ตัวแปรหนึ่งตัวจะใช้พื้นที่หน่วยความจำ 32 บิต(4 ไบต์) ในการคำนวณคณิตศาสตร์กับตัวแปร float จะช้ากว่าการคำนวณของตัวแปร int ดังนั้นพยายามหลีกเลี่ยงการคำนวณกับตัวแปร float เช่นในคำสั่งวนรอบที่ทำงานด้วยความเร็วสูงสุด

สำหรับฟังก์ชั่นทางเวลาที่ต้องแม่นยำอย่างมาก โปรแกรมเมอร์บางคนจะทำการแปลงตัวเลขทศนิยมให้เป็นจำนวนเต็มก่อน แล้วจึงคำนวณเพื่อให้ทำงานได้เร็วขึ้น จะเห็นว่าการคำนวณคณิตศาสตร์ของเลข floating point จะมีการใช้งานมากสำหรับการคำนวณค่าข้อมูลที่รับจาภายนอกเป็นตัวเลขทศนิยม ซึ่งทางผู้ศึกษาระบบไมโครคอนโทรลเลอร์มักจะมองข้ามไป

รูปแบบคำสั่ง

```
Float var=val;
```

พารามิเตอร์

Var คือชื่อของตัวแปร float ที่ต้องการ

Val คือค่าที่ต้องการกำหนดให้กับตัวแปร

ตัวอย่างที่ 3.10

```
Float myfloat;
Float sensorCalbrate=1.117;
```

ตัวอย่างที่ 3.11

```
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,not2)
```

ในฟังก์ชั่น serial.println() ของการส่งค่าตัวแปรไปยังพอร์ตอนุกรม จะตัดตัวเลขเศษทศนิยมออกให้เหลือเป็นจำนวนเต็ม ถ้าต้องการเพิ่มความละเอียดให้นำค่าตัวแปรคูณด้วย 10 ยกกำลังต่างๆ ตามที่ต้องการ

3.2.7 double : ตัวแปรประเภทเลขทศนิยมความละเอียดสองเท่า

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

3.2.8 string : ตัวแปรประเภทข้อความ

เป็นตัวแปรเก็บข้อความ ซึ่งในภาษา C จะนิยามเป็นอะเรย์ของตัวแปรประเภท char

ตัวอย่างที่ 3.12 การประกาศตัวแปรสตริง

```
Char Str1[15];
Char Str2[8]={'a', 'r', 'd', 'u', 'l', 'n', 'o'};
Char Str3[8]={ 'a', 'r', 'd', 'u', 'l', 'n', 'o', '\0'};
Char Str4[]= "arduino';
```

Char Str5[8]= “arduino’;

Char Str6[15]= “arduino’;

Str1 เป็นการประกาศตัวแปรสตริงโดยไม่ได้กำหนดค่าเริ่มต้น

Str2 ประกาศตัวแปรสตริงพร้อมกำหนดค่าให้กับข้อความทีละตัวอักษร จากตัวอย่าง คอมไพเลอร์จะเพิ่ม null character ให้เอง

Str3 ประกาศตัวแปรสตริงพร้อมกำหนดค่าให้กับข้อความทีละตัวอักษร จากตัวอย่างเพิ่ม ค่า null String เอง

Str4 ประกาศตัวแปรสตริงพร้อมกำหนดค่าตัวแปรในเครื่องหมายคำพูด จากตัวอย่างไม่ได้ กำหนดขนาดของตัวแปรคอมไพเลอร์ จะกำหนดขนาดให้เองตามจำนวนตัวอักษร +1 สำหรับ null struing

Str5 ประกาศตัวแปรสตริงพร้อมกำหนดตัวแปรในเครื่องหมายคำพูด จากตัวอย่างต้อง กำหนดขนาดตัวแปรเอง

Str6 ประกาศตัวแปรสตริงโดยกำหนดขนาดเผื่อไว้สำหรับข้อความอื่นที่ยาวมากกว่านี้

3.2.9 ตัวแปรอะเรย์ (array)

ตัวแปรอะเรย์เป็นตัวแปรหลายตัวที่ถูกเก็บรวมอยู่ในตัวแปรชื่อเดียวกันโดยอ้างถึงตัวแปรแต่ละตัวด้วยหมายเลขดัชนีที่เขียนอยู่ในวงเล็บสี่เหลี่ยมตัวแปรอะเรย์ของ Arduinoจะอ้างอิงถึงตามภาษา C ตัวแปรอะเรย์อาจจะซับซ้อนแต่ใช้แค่ตัวแปรอะเรย์อย่างง่ายจะตรงไปตรงมา

ตัวอย่างที่ 3.13 การประกาศตัวแปรอะเรย์

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 จะประกาศตัวแปรอะเรย์โดยไม่ระบุขนาดซึ่งทำได้เมื่อประกาศตัวแปรแล้วกำหนดค่าทันทีเพื่อให้คอมไพเลอร์นับว่าตัวแปรมีสมาชิกกี่ตัวและกำหนดค่าได้ถูกต้อง

การใช้งานตัวแปรอะเรย์

การใช้งานตัวแปรอะเรย์ทำได้โดยการพิมพ์ชื่อตัวแปร พร้อมระบุค่าดัชนีภายในเครื่องหมายวงเล็บสี่เหลี่ยม ค่าดัชนีของตัวแปรอะเรย์เริ่มต้นด้วยค่า0 ดังนั้นค่าตัวแปร mySensvals มีค่าดังนี้

mySenvals[0]==2,mySensvals[1]==4, ฯลฯ

การกำหนดค่าให้กับตัวแปรอะเรย์

mySensvals[0]=10;

การเรียกค่าสมาชิกของตัวแปรอะเรย์

X=mySensvals[4];

เทคนิคการเขียนโปรแกรมเกี่ยวกับตัวแปรอะเรย์

ในการเรียกใช้ค่าสมาชิกของตัวแปรอะเรย์ ต้องระวังอย่าอ้างถึงค่าในวงเล็บทีเกินกำหนด เช่น ประกาศตัวแปร int x [3] ตัวแปรมี 3 ตัว คือ x [10], x [1] และ x [2] ถ้าอ้างถึง x [3] จะเป็นการอ่นค่าจากหน่วยความจำซึ่งกำหนดไว้ใช้งานอย่างอื่น ค่าที่อ่านได้จะผิดพาด

การเขียนค่าให้กับตัวแปรอะเรย์ตัวที่เกินกว่าที่กำหนดไว้ อาจทำให้โปรแกรมแฮงค์(หยุดทำงาน) หรือ ทำงานผิดเพี้ยนไปการอ่านหรือเขียนค่าเกินค่าดัชนีของตัวแปรอะเรย์นี้ ทำให้เกิดบัก (ข้อผิดพลาด) ที่ยากต่อการค้นหา

อะเรย์และคำสั่งวนรอบ for โดยทั่วไปเราจะพบการใช้งานตัวแปรอะเรย์ภายในคำสั่ง for โดยใช้ค่าตัวแปรนับรอบคำสั่ง for เป็นค่าดัชนีของตัวแปรอะเรย์ ดังตัวอย่างการพิมพ์ค่าสมาชิกแต่ละตัวของตัวแปรอะเรย์ผ่านพอร์ตอนุกรมให้เขียนโปรแกรมดังนี้

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

ตัวอย่างโปรแกรมสาธิตการใช้งานตัวแปรอะเรย์ที่สมบูรณ์ดูได้ในตังอย่างในหัวข้อ Tutorials ในเว็ปไซต์ www.arduino.cc

สรุปชนิดของตัวแปรใน Arduino ที่ใช้มาก

- Boolean ใช้เก็บค่าข้อมูลเพียง 2 จำนวนคือ TRUE (จริง) และ FALSE (เท็จ)

- Char ใช้เก็บข้อมูลขนาด 8 บิต ใช้สำหรับเก็บค่ารหัสของตัวอักษร ซึ่งสามารถกำหนดเป็นค่าหรือตัวอักษรไว้ภายใต้เครื่องหมาย ฟันเดี่ยวก็ได้ เช่น ‘A’ หรือ 0x41 หรือ 65
- Byte ใช้เก็บค่าข้อมูลขนาด 8 บิตที่เป็นค่าจำนวนเต็มแบบไม่คิดเครื่องหมาย(เหมือนกันกับ uasinged char ในภาษาซี)ซึ่งสามารถเก็บค่าข้อมูลได้ 256 ค่า คือ 0-255
- Int หรือ integer ใช้เก็บค่าข้อมูลขนาด 16 บิต ที่เป็นค่าจำนวนเต็ม แบบไม่คิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 65536 ค่า คือ-32768 ถึง +32767
- Unsigned int ใช้เก็บค่าข้อมูลขนาด 16 บิต ที่เป็นค่าจำนวนเต็ม แบบไม่คิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 65536 ค่า 0-65535
- Long ใช้เก็บข้อมูลขนาด 32 บิต ที่เป็นค่าเลขจำนวนเต็มแบบคิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 4294967296 ค่าคือ –2,147,483,648 ถึง 2,147,483,647
- Unsinged long ใช้เก็บค่าข้อมูลขนาด 32 บิต ที่เป็นค่าเลขจำนวนเต็มแบบไม่คิดเครื่องหมายโดยสามารถใช้เก็บข้อมูลได้ 4294967296 ค่า คือ 0 ถึง 4,294,967,295
- Float ใช้เก็บค่าข้อมูลที่เป็นตัวเลข ทศนิยมแบบคิดเครื่องหมายขนาด 32 บิต โดยสามารถเก็บค่าได้ระหว่าง 3.4E-38 ถึง3.4E+38
- Double ใช้เก็บค่าข้อมูลที่เป็นเลขทศนิยมเช่นเดียวกันกับ float แต่มีค่าความละเอียดกว่า float ถึง 2 เท่า สามารถเก็บค่าได้มากถึง 1.7 E+308
- Void เป็นตัวแปรแบบไม่มีการเก็บค่าใดๆคือไม่มีค่านั่นเอง
- Arrays เป็นตัวเลขที่ใช้เก็บข้อมูลหลายๆค่าไว้ในตัวแปรตัวเพียงชื่อเดียวแต่มีตัวเลขสำหรับชี้ตำแหน่งการเก็บข้อมูลต่างกันโดยตัวเลขที่ใช้ทำหน้าที่เป็นตัวชี้ตำแหน่งของข้อมูล เรียกว่า Inbex Number โดยค่าลำดับของข้อมูลในตัวแปร array ตำแหน่งจะมีค่าเป็นศูนย์เสมอ
- String เป็นตัวแปรใช้เก็บข้อความหรือตัวอักษรหลายๆตัวซึ่ง string ก็คือ array ของตัวแปรแบบ char นั่นเอง
- Pointer เป็นตัวแปรที่ไม่ได้ใช้เก็บข้อมูลแต่ใช้เก็บค่าตำแหน่งแอดเดรสของหน่วยความจำที่ใช้สร้างเป็นตัวแปรสำหรับเก็บข้อมูลซึ่งตัวแปรแบบนี้จะใช้ทำหน้าที่เป็นตัวชี้ไปยังตำแหน่งแอดเดรสของตัวแปรอื่นๆอีกทีหนึ่ง

3.3 คำสั่งในภาษา Arduino

3.3.1 คำสั่ง if

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรม เช่น ถ้าอินพุตมีค่ามากกว่าค่าที่กำหนดไว้จะให้ทำอะไรโดยรูปแบบการเขียนดังนี้

```
If(somevarible>50){
// do  something Here
}
```

ตัวโปรแกรมจะทดสอบว่าตัวแปร someVariable มีค่ามากกว่า 50 หรือไม่ ถ้าไม่ใช้ให้ทำอะไร ถ้าไม่ใช้ให้ข้ามการทำงานส่วนนี้ การทำงานของคำสั่งจะทดสอบเงื่อนไขที่เขียนในเครื่องหมายวงเล็บปีกกา ถ้าเงื่อนไขเป็นเท็จข้ามการทำงานส่วนนี้ไป ส่วนของการทดสอบเงื่อนไขที่เขียนอยู่ในวงเล็บ จะต้องใช้ตัวกระทำเปรียบเทียบต่างๆดังนี้

X==y(x เท่ากับ y)

X!=y(x ไม่เท่ากับ y)

X<y(xน้อยกว่า y)

x>y (xมากกว่า y)

x<=y(xน้อยกว่าหรือเท่ากับy)

x>=y(xมากกว่าหรือเท่ากับy)

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

3.3.2 คำสั่ง if...else

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมได้มากกว่าคำสั่ง if ธรรมดา โดยสามารถกำหนดได้ว่าถ้าเงื่อนไขเป็นจริงให้ทำอะไรถ้าเป็นเท็จให้ทำอะไรเช่นถ้าค่าอินพุตแอนะล็อกที่อ่านได้น้อยกว่า 500 ให้ทำอะไรถ้ามากกว่าหรือเท่ากับ 500 ให้ทำอีกอย่างจะเขียนคำสั่งได้ดังนี้

ตัวอย่างที่ 3.14

```
if(pinFiveInput<500){
```

```
  //do thing A
 }
else{
  //do thing B
 }
```

หลังคำสั่ง else สามารถตามด้วยคำสั่ง if สำหรับการทดสอบอื่นๆทำให้รูปแบบคำสั่งกลายเป็น if...else..if เป็นการทดสอบเงื่อนไขต่างๆ เมื่อเป็นจริงให้ทำตามดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.15

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

หลังคำสั่ง else สามารถตามด้วยคำสั่ง if ได้ไม่จำกัดจำนวน สามารถใช้คำสั่ง case แทนคำสั่ง if...else...if สำหรับการทดสอบเงื่อนไขจำนวนมากๆได้) เมื่อใช้คำสั่ง if...else แล้วต้องกำหนดด้วยว่าถ้าทดสอบไม่ตรงตามเงื่อนไขใดๆเลยให้ทำอะไรโดยให้กำหนดที่คำสั่ง else ตัวสุดท้าย

3.3.3 คำสั่ง for()

คำสั่งนี้ใช้เพื่อสั่งให้คำสั่งที่อยู่ภายในวงเล็บปีกกาหลัง for มีการทำงานซ้ำกันตามจำนวนรอบที่ต้องการคำสั่งนี้มีประโยชน์มากสำหรับการทำงานใดๆที่ต้องทำซ้ำกันและทราบจำนวนรอบของการทำงานที่แน่นอน ใช้คู่กับตัวแปรอะเรย์ในการเก็บสะสมค่าที่อ่านได้จากขาอินพุตแอนะล็อกหลายๆขา ที่มีเลขขาต่อเนื่องกันรูปแบบของคำสั่ง for() แบ่งได้ 3 ส่วนดังนี้

```
for(initialization;condition;increment)
 {
```

```
//Statement(as);
 }
```

เริ่มต้นด้วย initialization ใช้กำหนดค่าเริ่มต้นของตัวแปรควบคุมการวนรอบในการทำงานแต่ละรอบจะทดสอบ condition ถ้าเงื่อนไขเป็นจริงทำตามคำสั่งในวงเล็บปีกกา แล้วมาเพิ่มหรือลดค่าตัวแปรตามที่สั่งใน increment แล้วทดสอบเงื่อนไขอีก ทำซ้ำจนกว่าเงื่อนไขจะเป็นเท็จ

ตัวอย่างที่ 3.16

```
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++ ที่มีตัวแปรที่ไม่เกี่ยวข้องมาเขียนใหม่ในส่วนของ initialization condition และ increment ของคำสั่ง for ได้

3.3.4 คำสั่ง switch – case

ใช้ทดสอบเงื่อนไขเพื่อกำหนดการทำงานของโปรแกรม ถ้าตัวแปรที่ทดสอบตรงกับเงื่อนไขใดก็ให้ทำงานตามที่กำหนดไว้ในพารามิเตอร์

Var ตัวแปรที่ต้องการทดสอบว่าตรงกับเงื่อนไขใด

Default ถ้าไม่ตรงกับเงื่อนไขใดๆเลย ให้ทำคำสั่งต่อท้ายนี้

Break เป็นส่วนสำคัญมากใช้เขียนต่อท้าย case ต่างๆ เมื่อพบเงื่อนไขแล้วโปรแกรมจะทำงานตามเงื่อนไขต่อไปเรื่อยๆ จนกว่าจะพบคำสั่ง break

ตัวอย่างที่ 3.17

```
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
}
```

3.3.5 คำสั่ง while

เป็นคำสั่งวนรอบโดยจะทำคำสั่งที่เขียนในวงเล็บปีกกาอย่างต่อเนื่อง จนกว่าเงื่อนไขที่เขียนวงเล็บของคำสั่ง while() จะเป็นเท็จ คำสั่งที่ให้ทำซ้ำจะต้องมีการเปลี่ยนแปลงค่าตัวแปรที่ใช้ทดสอบ โดยมีการเพิ่มค่าตัวแปรหรือมีเงื่อนไขภายนอก เช่นอ่านค่าจากตัวตรวจจับได้เรียนร้อยแล้วให้หยุดการอ่านค่า มิฉะนั้นเงื่อนไขในวงเล็บของ while() เป็นจริงตลอดเวลา ทำให้คำสั่ง while ทำงานวนรอบไปเรื่อยๆไม่รู้จบ

รูปแบบคำสั่ง

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

พารามิเตอร์ expression เป็นคำสั่งทดสอบเงื่อนไข(ถูกหรือผิด)

ตัวอย่างที่ 3.18

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

3.4 ส่วนของตัวกระทำทางคณิตศาสตร์

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

3.4.1 ตัวกระทำทางคณิตศาสตร์

บวก ลบ คูณ และหาร ใช้หาค่าผลรวม ผลต่าง ผลคูณ และผลหาร ค่าของตัวถูกกระทำสองตัวโดยให้คำตอบมีประเภทตรงกับตัวถูกกระทำทั้งสองตัวเช่น 9/4 ให้คำตอบเท่ากับ 2 เนื่องจากทั้ง 9 และ 4 เป็นตัวเลขจำนวนเต็ม (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 : เป็นคำสั่งของตัวแปรหรือค่าคงที่ใดๆ

Value2 : เป็นคำสั่งของตัวแปรหรือค่าคงที่ใดๆ

ตัวอย่างที่ 3.19

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

เทคนิคสำหรับการเขียนโปรแกรม

1.) เลือกขนาดของตัวแปรให้ใหญ่พอสำหรับเก็บค่าผลลัพธ์ที่มากที่สุดของการคำนวณ

2.) ต้องทราบว่าที่ค่าใดตัวแปรที่เก็บจะมีการวนซ้ำค่ากลับ และวนกลับอย่างไร ตัวอย่างเช่น (0ไป1)หรือ(0ไป-32768)

3.) สำหรับการคำนวณที่ต้องการเศษส่วนให้ใช้ตัวแปรประเภท float แต่ให้ระวังผลลบ เช่น ตัวแปรมีขนาดใหญ่ คำนวณได้ช้า

4.) ใช้ตัวกระทำ cast เช่น(int)myfloat ในการเปลี่ยนประเภทของตัวแปรชั่วคราวขณะที่โปรแกรมทำงาน

3.4.2 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 3.20

```
x-7%5;//x now contains2
x-9%5;//x now contains4
x-5%5;//x now contains0
x-4%5;//x now contains4
```

ตัวกระทำหารเอาเศษนี้ใช้ประโยชน์มากในงานที่ต้องการให้เหตุการณ์เกิดขึ้นด้วยช่วงเวลาที่สม่ำเสมอหรือใช้ทำให้หน่วยความจำที่เก็บตัวแปรอะเรย์ เกิดการส่งค่ากลับ(Roll Over)

ตัวอย่างที่ 3.21

```
//check a senser every 10 times through a loop
void setup()
 {
   i++;
if((i%10)--0)
   {
   X=analogRead(sendPin);//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 memory slot
    senVal[(i++)%5]-analogRead(sensPin);
    average=0;
for(j=0;j<5;j++)
 {
  average+=sensVall[j];//add samples
  }
  average= average/5;//divide by total
}
```

3.5 ส่วนของตัวกระทำเปรียบเทียบ

ใช้ประกอบกับคำสั่ง if() และ while() เพื่อทดสอบเงื่อนไขหรือเปรียบเทียบค่าตัวแปรต่างๆโดยจะอยู่ภายในเครื่องหมาย()

x==y(x เท่ากับ y)

x!=y(xไม่เท่ากับ y)

x<y(x น้อยกว่าy)

x>y(x มากกว่า y)

x<=y(x น้อยกว่าหรือเท่ากับ y)

x>=y(x มากกว่าหรือกับ y)

ใช้ในการเปรียบเทียบของคำสั่ง if() มี 3 ตัว คือ &&,||และ !

3.5.1 && (ตรรกะ และ)

ให้ค่าเป็นจริงเมื่อผลการเปรียบเทียบทั้งสองข้างเป็นจริงทั้งคู่

ตัวอย่างที่ 3.22

```
if(x>0 && x < 5)
```

```
{
///..
}
```

ให้ค่าเป็นจริงเมื่อ x มากกว่า 0 น้อยกว่า 5 (มีค่า 1 ถึง 4)

3.5.2 ||(ตรรกะ หรือ)

ให้ค่าเป็นจริงเมื่อผลการเปรียบเทียบพบว่า มีตัวแปรใดเป็นจริงหรือเป็นจริงทั้งคู่

ตัวอย่างที่ 3.23

```
if(x>0||y>0)
{
///..
} ให้ผลเป็นจริงเมื่อ x หรือ y มีค่ามากกว่า 0
```

3.5.3 ! (ใช้กลับผลเป็นตรงกันข้าม)

ให้ค่าเป็นจริงเมื่อผลการเปรียบเทียบเป็นเท็จ

ตัวอย่างที่ 3.24

```
If(!x)
{
///..
} ให้ผลเป็นจริงถ้า x เป็นเท็จ(เช่น ถ้า x =0 ให้ผลเป็นจริง)
```

3.5.4 ข้อควรระวัง

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

ตัวอย่างที่ 3.25

```
if(a>=10&& a<20)
{
```

```
}
// true  if a is between 10 and 20
```

3.6 ส่วนของตัวกระทำระดับบิต

ตัวกระทำระดับบิตจะนำบิตของตัวแปรมาประมวลผล ใช้ประโยชน์ในการแก้ปัญหาด้านการเขียนโปรแกรมได้หลากหลายตัวกระทำ ระดับของภาษา C (ซึ่งรวมถึง Arduino) มี 6 ตัวได้แก่

&(bitwise AND)

|(OR)

^(Exclusive OR)

~ (NOT)

<< (เลื่อนบิตไปทางขวา)

>>(เลื่อนบิตไปทางซ้าย)

3.6.1 ตัวกระทำระดับบิต AND (&)

คำสั่ง AND ในระดับบิตของภาษา C เขียนได้โดยใช้ & หนึ่งตัวโดยต้องเขียนระหว่างนิพจน์หรือตัวแปรที่เป็นตัวเลขจำนวนเต็ม การทำงานจะนำข้อมูลแต่ละบิตของตัวแปรทั้งสองตัวมากระทำทางตรรกะและ(AND) โดยมีกฎดังนี้ ถ้าอินพุตทั้งสองตัวเป็น “1” ทั้งคู่ เอาต์พุตเป็น “1” ดังตัวอย่างต่อไปนี้ ในการดูให้คู่ของตัวกระทำตามแนวตั้ง

```
0   0   1   1  Operand 1
0   1   0   1  Operand 2
---------------------------------------
0   0   0   1   returned result
```

ใน Arduino ตัวแปรประเภท int จะมีขนาด 16 บิต ดังนั้นเมื่อใช้ตัวกระทำบิต AND จะมีการกระทำตรรกะและพร้อมกันกับข้อมูลทั้ง 16 บิต ดังตัวอย่างในส่วนของโปรแกรมต่อไปนี้

ตัวอย่างที่ 3.26

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

ในตัวอย่างนี้จะนำข้อมูลทั้ง 16 บิต ของตัวแปร a และ b มากระทำทางตรรกะ AND แล้วนำผลลัพธ์ที่ได้ทั้ง 16 บิต ไปเก็บไว้ที่ตัวแปร c ซึ่งได้ค่าเป็น 01000100 ในเลขฐานสองหรือเท่ากับ 68 ฐานสิบ

นิยมใช้ตัวกระทำระดับบิต AND เพื่อเลือกข้อมูลบิตที่ต้องการ(อาจเป็นหนึ่งบิตหรือหลายบิต) จากตัวแปร int ซึ่งการเลือกเพียงบางบิตนี้จะเรียกว่า masking

3.6.2 ตัวกระทำระดับบิต OR (|)

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

0 0 1 1 Operand 1

0 1 0 1 Operand 2

---

0 1 1 1 returned result

ตัวอย่างที่ 3.27

ส่วนของโปรแกรมแสดงการใช้ตัวกระทำระดับบิต OR

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

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

ตัวแปรชื่อ PORT D 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 ถ้าไปเปลี่ยนกระทบต่อการสื่อสารแบบอนุกรม

อัลกอริธึมสำหรับโปแกรมเป็นดังนี้

1.) อ่านค่าจาก PORTD แล้วล้างค่าเฉพาะบิตที่ต้องการควบคุม(ใช้ตัวกระทำแทนบิต AND)

2.) นำค่า PORTD ที่แก้ไขจากข้างต้นมารวมกับค่าบิตที่ต้องการควบคุม (ใช้ตัวกระทำแบบบิต OR)

ซึ่งเขียนเป็นโปแกรมได้ดังนี้

```
 int i;//counter  variable
 int j
void setup()
  {
   DDRD  =DDRD|B11111100;
   //set direction bits for pin 2 to 7'
   //Leave 0 and 1 untouched(xx|00  ==xx)
   //same as pinMode(pin,OUTPUT)for pins 2 to 7
   //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 marking
  delay(100);
  }
}
```

3.6.3 คำสั่งระดับบิต Exclusive OR(^)

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

```
0  0  1  1   Operand 1
0  1  0  1   Operand 2
--------------
0   1   1  0   Returned result
```

หรือกล่าวได้อีกอย่างว่าด้วยตัวกระทำระดับบิต XOR จะให้เอาต์พุตเป็น “0” เมื่ออินพุตทั้งสองตัวมีค่าเหมือนกัน และให้เอาต์พุตเป็น “1” เมื่ออินพุตทั้งสองมีค่าตรงกัน

ตัวอย่างที่ 3.28

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

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

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

ตัวอย่างที่ 3.29

```
//Blink Pin 5
// demo for Exclusive OR
void setup()
  {
  DDRD = DDRD | 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);
 }
```

3.6.4 **ตัวกระทำระดับบิต NOT (~)**

ตัวกระทำระดับบิต NOT จะเขียนโดยใช้สัญลักษณ์เครื่องหมาย ~ ตัวกระทำนี้จะใช้กับตัวถูกกระทำเพียงตัวเดียวที่อยู่ขวามือ โดยการสลับบิตทุกบิตให้มีค่าตรงกันข้ามคือจาก “0” “1” และจาก “1” เป็น “0” ดังตัวอย่าง

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

เมื่อกระทำแล้วทำให้ตัวแปร b มีค่า -104(ฐานสิบ) ซึ่งคำตอบที่ได้ติดลบ เนื่องจากบิตที่มี

ความสำคัญสูงสุด (บิตซ้ายมือสุด) ของตัวแปร int อันเป็นบิตแจ้งว่าตัวเลขเป็นบวกหรือลบ มีค่าเป็น “1” แสดงว่าค่าที่ได้ติดลบ โดยในคอนโทรลเลอร์จะเก็บค่าตัวเลขทั้งบวกและลบตามระบบทูคอมพลีเมนต์ (2,s complement)

การกระทำประกาศตัวแปร int ซึ่งมีความหมายเหมือนกับการประกาศตัวแปรเป็น signed int ต้องระวังค่าของตัวแปรจะติดลบได้

3.7 ไวยากรณ์ภาษา C/C++ ของ Arduino

3.7.1 เซมิโคลอน – semicolon;

ให้เขียนแจ้งว่าจบคำสั่ง

ตัวอย่างที่ 3.30

```
int  a = 13;
```

บรรทัดคำสั่งที่ลืมเขียนปิดท้ายด้วยเซมิโคลอน จะทำให้แปลโปรแกรมไม่ผ่าน โดยตัวแปรภาษาอาจจะแจ้งให้ทราบว่า ไม่พบเครื่องหมายเซมิโคลอน หรือแจ้งเป็นการผิดพลาดอื่นๆ บางกรณีที่ตรวจสอบบรรทัดที่แจ้งว่าเกิดการผิดพลาดแล้วไม่พบที่ผิด ให้ตรวจสอบบรรทัดก่อนหน้านั้น

3.7.2 วงเล็บปีกกา - curly brac { }

เครื่องหมายวงเล็บปีกกา เป็นส่วนสำคัญของภาษาซี โดยมีการใช้งานต่างตำแหน่ง สร้างความสับสนให้กับผู้ที่เริ่มต้นวงเล็บปีกกาเปิด{ จะต้องเขียนตามด้วยวงเล็บปีกกาปิด } ด้วยเสมอ

3.7.3 หมายเหตุบรรทัดเดียวหรือหลายบรรทัด // และ /*....*/

เป็นส่วนของโปรแกรมที่ผู้ใช้เขียนเพิ่มเติมว่าโปรแกรมทำอย่างไร โดยส่วนที่เป็นหมายเหตุจะไม่ถูกคอมไพล์ ไม่นำไปประมวลผล มีประโยชน์มากสำหรับการตรวจสอบโปรแกรมภายหลังหรือใช้แจ้งให้เพื่อนร่วมงานหรือบุคคลอื่นทราบว่าบรรทัดนี้ใช้ทำอะไร ตัวหมายเหตุภาษา C มี 2 ประเภท คือ

1.) หมายเหตุบรรทัด เขียนเครื่องหมายสเลช // 2ตัวหน้าบรรทัด

2.) หมายเหตุหลายบรรทัด เขียนเครื่องหมายสเลช / คู่กับดอกจัน * คร่อมข้อความที่เป็นหมายเหตุ เช่น /* blabal */

3.7.4 #define

เป็นคำสั่งที่ใช้งานมาก ในการกำหนดค่าคงที่กับโปรแกรม ในการกำหนดค่าคงที่ไม่ได้เปลืองพื้นที่หน่วยความจำของไมโครคอนโทรลเลอร์แต่อย่างไร เมื่อถึงขั้นตอนการแปลภาษา คอมไพเลอร์จะแทนที่ตัวอักษรข้อความด้วยค่าที่กำหนดไว้ใน Arduino จะใช้คำสั่ง #define ตรงกับภาษา C

รูปแบบ

#define constantName value

อย่าลืมเครื่องหมาย #

ตัวอย่าง 3.31

#define ledpin3 เป็นการกำหนดให้ตัวแปร ledpin เท่ากับค่าคงที่ 3

เทคนิคสำหรับการเขียนโปรแกรมท้ายคำสั่ง # define ไม่ต้องมีเครื่องหมายเซมิโคลอน ถ้าใส่เกินแล้วเวลาคอมไพล์โปรแกรมจะแจ้งว่าเกิดการผิดพลาดในบรรทัดถัดไป

3.7.5 #include

ใช้สั่งให้รวมไฟล์อื่นๆเข้ากับไฟล์โปรแกรมหลักก่อน แล้วจำทำการคอมไพล์โปรแกรม

รูปแบบคำสั่ง

#include<file>

#include “file”

ตัวอย่างที่ 3.32

#inckude<stdio.h>

#include “lcd.h”

บรรทัดแรกจะสั่งให้เรียกไฟล์ stdio.h มารวมกับไฟล์โปรแกรมหลัก โดยค้นหาไฟล์จากตำแหน่งที่เก็บไฟล์ระบบของ Arduino

บรรทัดที่ 2 สั่งให้รวมไฟล์ lcd.h มารวมกับไฟล์โปรแกรมหลัก โดยหาไฟล์จากตำแหน่งของไฟล์ภาษา C ปกติเป็นไฟล์ที่ผู้ใช้สร้างขึ้นเอง

ในการแก้ไขโปรแกรมใน Arduino มีข้อแนะนำว่าอย่าแก้ไขบรรทัดนั้นทันที ให้ทำบรรทัดนั้นเป็นหมายเหตุก่อนแล้วจึงแก้โปรแกรมในบรรทัดนั้น

3.8 ค่าคงที่ (constants)

ค่าคงที่เป็นกลุ่มตัวอักษรหรือข้อความที่ได้กำหนดค่าไว้ล่วงหน้าแล้ว ตัวคอมไพเลอร์ของ Arduino จะรู้จักกับค่าคงที่เหล่านี้แล้ว ไม่จำเป็นต้องประกาศหรือกำหนดค่าคงที่

3.8.1 HIGH,LOW : ใช้กำหนดค่าทางตรรกะ

ในการอ่านหรือเขียนค่าให้กับขาที่เป็นดิจิตอล ค่าที่ได้มี 2 ค่าคือ HIGH หรือ LOW เท่านั้น

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

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

3.8.2 INPUT ,OUTPUT : กำหนดทิศทางของขาพอร์ตดิจิตอล

ขาของพอร์ตดิจิตอลทำหน้าที่ได้ 2 อย่าง คือ เป็นอินพุต(INPUT) หรือเอาต์พุต(OUTPUT) ซึ่งค่าคงที่นี้ก็ระบุไว้ชัดเจน

ตัวกระทำอื่นๆที่เกี่ยวข้องกับตัวแปร

cast : การเปลี่ยนประเภทตัวแปรชั่วคราว

Cast เป็นตัวกระทำที่ใช้สั่งให้เปลี่ยนประเภทของตัวแปรไปเป็นประเภทอื่น และบังคับให้คำนวณค่าตัวแปรเป็นประเภทใหม่

รูปแบบคำสั่ง

(type) variable

เมื่อ type เป็นประเภทของตัวแปรใดๆ(เช่น int, float,long)

Variable เป็นตัวแปรหรือค่าคงที่ใดๆ

ตัวอย่างที่ 3.33

```
Int I;
Float f;
F=3.6;
I=(int t)f;//now I is 3
```

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

sizeof : แจ้งขนาดของตัวแปร

ใช้แจ้งบอกจำนวนไบต์ของตัวแปรที่ต้องการทราบค่า ซึ่งเป็นทั้งตัวแปรปกติและตัวแปรอะเรย์

รูปแบบคำสั่ง

เขียนได้ทั้งสองแบบดังนี้

sizeiof(variable)

sizeof(variable)

เมื่อ variable คือตัวแปรปกติหรือตัวแปรอะเรย์(int,float,long) ที่ต้องการทราบขนาด

ตัวอย่างที่ 3.34

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);
    }
  }
```

3.9 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 หลังจากการทำงานในฟังก์ชั่น setup() จึงสรุปได้ว่า ฟังก์ชั่น setup() คือส่วนต้นของโปรแกรมที่ใช้ในการประกาศ หรือตั้งค่าการทำงานในตอนเริ่มต้นทำงาน ในขณะที่ฟังก์ชั่น loop() เป็นเสมือนส่วนของโปรแกรมหลักที่ต้องวนการทำงานอย่างต่อเนื่องตลอดเวลาอย่างไรก็ตามในบางโปรแกรมอาจมีเฉพาะส่วนของฟังก์ชั่น setup() และไม่มีฟังก์ชั่น loop() ก็ได้แสดงว่าโปรแกรมนั้นๆ ต้องการตั้งค่าการทำงานหรือกำหนดให้มีการทำงานเพียงครั้งเดียวหรือรอบเดียว แล้วจบการทำงานทันที เทคนิคสำหรับการเขียนโปแกรม เลือกขนาดของตัวแปรให้ใหญ่พอสำหรับเก็บค่าผลลัพธ์ที่มากที่สุดของการคำนวณ ต้องการทราบว่าที่ค่าใด ตัวแปรที่เก็บจะมีการวนซ้ำค่ากลับ และวนกลับอย่างไร สำหรับการคำนวณที่ต้องการเศษส่วนให้ใช้ตัวแปรประเภท float แต่ให้ระวังผลลบ เช่นตัวแปรมีขนาดใหญ่ คำนวณได้ช้า ใช้ตัวกระทำ cast เช่น (int)myfloat ในการเปลี่ยนประเภทของตัวแปรชั่วคราวขณะโปรแกรมทำงาน

เอกสารอ้างอิง

เอกชัย มะการ.(2552).เรียนรู้ เข้าใจ ใช้งาน ไมโครคอนโทรลเลอร์ตระกูล AVR ด้วย Arduino.บริษัทอีทีทีจำกัด,กรุงเทพฯ.

บทความ Arduino.(2559).arduino and Motor Control.(ออนไลน์).สืบค้นจาก :

https://www.arduino.cc/en/Tutorial/BuiltInExamples.(15 มีนาคม2559).

แบบทดสอบหลังเรียน

หน่วยที่ 3 ซอฟต์แวร์ควบคุมหุ่นยนต์

<u>คำชี้แจง</u> 1. อ่านคำถามต่อไปนี้แล้วทำเครื่องหมายกากบาท (X) ข้อที่ถูกที่สุดลงในกระดาษคำตอบเพียงข้อเดียว

2. เวลาสำหรับทำแบบทดสอบ 10 นาที

****

ข้อที่ 1. คำสั่ง if ใช้เพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมอย่างไร

ก. ถ้าเงื่อนไขเป็นเท็จทำตามคำสั่งที่เขียนไว้ในวงเล็บปีกกา ถ้าเงื่อนไขเป็นจริงข้ามการทำงาน

ข. ถ้าเงื่อนไขเป็นจริงทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นเท็จข้ามการทำงาน

ค. ถ้าเงื่อนไขเป็นจริงทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นจริงข้ามการทำงาน

ง. ถ้าเงื่อนไขเป็นเท็จทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นเท็จข้ามการทำงาน

จ. ถ้าเงื่อนไขเป็นจริงและเท็จให้ทำตามคำสั่งที่เขียนในวงเล็บปีกกา

ข้อที่ 2. ผลที่ได้จากการหารเอาเศษของ x=5%5 คือข้อใด

ก. 1

ข. 0

ค. 0.5

ง. 0.1

จ. 0.2

ข้อที่ 3. ตัวกระทำเปรียบเทียบใช้สำหรับประกอบกับคำสั่งใด

ก. If() และ while()

ข. switch() และ case()

ค. for() และ while()

ง. for() และ if()

จ. switch case() และ for

ข้อที่ 4. int i;

for(i=0;i<5;i=i+1)

{

```
Serial.println(myPins[i]);
}
```

จาก code ข้างบน ใช้ตัวแปรประเภทกับคำสั่งใด

ก. ตัวแปรอาเรย์กับคำสั่งวนรอบซ้ำ

ข. ตัวแปรอาเรย์กับคำสั่งมีเงื่อนไขเป็นจริง

ค. ตัวแปรอาเรย์กับคำสั่งมีเงื่อนไขเป็นเท็จ

ง. ตัวแปรอาเรย์กับคำสั่ง switch – case

จ. ตัวแปรอาเรย์กับคำสั่ง for

ข้อที่ 5. การทำงานจะนำข้อมูลแต่ละบิตของตัวแปรทั้งสองตัวมากระทำทางตรรกะ โดยมีกฎถ้าอินพุตทั้งสองตัวเป็น “1” ทั้งคู่ เอาต์พุตเป็น “1” คือการทำงานของตรรกะใด

ก. NAND

ข. NOT

ค. AND

ง. EX-OR

จ. Or

ข้อที่ 6. ตรรกะที่ใช้ในการเปรียบเทียบของคำสั่ง if() คือข้อใด

ก. &&และ!

ข. || และ !

ค. && และ ||

ง. &&,||และ !

จ. &&,**,และ ||

ข้อที่ 7.
```
if(buttonPushCounter%4==0){
digitalWrite(ledPin,HIGH);
}else{
digitalWrite(ledPin,LOW);
}
```

จาก code การทำงานของcode ตรงกับข้อใด

ก. หาก buttonPushCounter%4==0 เป็นจริง ledPin จะมีสถานะเป็น HIGH
เป็นเท็จ ledPin จะมีสถานะเป็น LOW

ข. หาก buttonPushCounter%4==0 เป็นจริง ledPin จะมีสถานะเป็น LOW
เป็นเท็จ ledPin จะมีสถานะเป็น LOW

ค. หาก buttonPushCounter%4==0 เป็นเท็จ ledPin จะมีสถานะเป็น HIGH
เป็นจริง ledPin จะมีสถานะเป็น LOW

ง. หาก buttonPushCounter%4==0 เป็นเท็จ ledPin จะมีสถานะเป็น LOW
เป็นเท็จ ledPin จะมีสถานะเป็น HIGH

จ. หาก buttonPushCounter%4==0 เป็นเท็จ ledPin จะมีสถานะเป็น HIGH
เป็นจริง ledPin จะมีสถานะเป็น HIGH

ข้อที่ 8. digitalWrite(motorPin3, HIGH); มีความหมายตรงกับข้อใด

ก. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็น HIGH

ข. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็น LOW

ค. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็นจริง

ง. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็นเท็จ

จ. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็น “0”

ข้อที่ 9. digitalWrite(motorPin3, LOW); มีความหมายตรงกับข้อใด

ก. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็น HIGH

ข. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็น “0”

ค. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็นจริง

ง. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็นเท็จ

จ. กำหนดให้ค่าสัญญาณดิจิตอลของ motorPin3 มีสถานะเป็น LOW

ข้อที่ 10. ((digitalRead(LS)) && digitalRead(RS)) มีความหมายตรงกับข้อใด

ก. อ่านค่าสัญญาณดิจิตอลตำแหน่ง LS ทำการเปรียบเทียบ OR กับค่าสัญญาณตำแหน่ง RS

ข. อ่านค่าสัญญาณดิจิตอลตำแหน่ง LS ทำการเปรียบเทียบ AND กับค่าสัญญาณตำแหน่ง RS

ค. อ่านค่าสัญญาณดิจิตอลตำแหน่ง LS ทำการเปรียบเทียบ NOT กับค่าสัญญาณตำแหน่ง RS

ง. อ่านค่าสัญญาณดิจิตอลตำแหน่ง LS ทำการเปรียบเทียบ EX-OR กับค่าสัญญาณตำแหน่ง

RS

จ. อ่านค่าสัญญาณดิจิตอลตำแหน่ง LS ทำการเปรียบเทียบ NAND กับค่าสัญญาณตำแหน่ง RS

เฉลยแบบประเมินผล

หน่วยที่ 3 ซอฟต์แวร์ควบคุมหุ่นยนต์

ก่อนเรียน		หลังเรียน	
ข้อที่	**คำตอบ**	**ข้อที่**	**คำตอบ**
1.	ก	1.	ข
2.	ง	2.	ข
3.	ก	3.	ก
4.	ก	4.	ก
5.	ก	5.	จ
6.	ข	6.	ง
7.	ก	7.	ก
8.	ข	8.	ก
9.	ง	9.	จ
10.	ก	10.	ข