หน่วยที่ 3

ชุดคำสั่งไมโครคอนโทรลเลอร์ Arduino IDE

จุดประสงค์เชิงพฤติกรรม

1. สามารถอธิบายความหมายของคำสั่งภาษาซี
2. สามารถเขียนโปรแกรมด้วยคำสั่งภาษาซี Arduino
3. เตรียมความพร้อมด้านวัสดุ อุปกรณ์สอดคล้องกับงานได้อย่างถูกต้อง

ชุดคำสั่งที่ใช้ในการเขียนโปรแกรมไมโครคอนโทรลเลอร์ Arduino

การเขียนโปรแกรมไมโครคอนโทรลเลอร์ Arduino สามารถเขียนได้ทั้งภาษาแอสเซมบลี และ ภาษาระดับสูง ได้แก่ ภาษาซี ขึ้นอยู่กับว่าผู้พัฒนาโปรแกรมเลือกใช้ภาษาใดมาใช้สำหรับเขียนโปรแกรม เพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ Arduino แต่เนื้อหาในหนังสือเล่มนี้เน้นการเขียน โปรแกรมด้วยภาษาซีเป็นหลัก

ภาษาแอสเซมบลีสำหรับไมโครคอนโทรลเลอร์ Arduino

ภาษาแอสเซมบลีจัดเป็นภาษาในระดับล่าง เป็นภาษาที่มีความใกล้เคียงกับภาษาเครื่องและ ฮาร์ดแวร์ของซีพียู โครงสร้างของภาษาแอสเซมบลีสำหรับไมโครคอนโทรลเลอร์ Arduino ไม่แตกต่าง จากภาษาแอสเซมบลีของสถาปัตยกรรมไมโครคอนโทรลเลอร์อื่น ๆ ส่วนการเขียนโปรแกรมภาษา แอสเซมบลีต้องทำการใช้ชุดคำสั่งของซีพียูเพื่อเข้าถึงหน่วยความจำและข้อมูลในรีจิสเตอร์ซึ่งมีโดยตรง ส่งผลให้ภาษาแอสเซมบลีมีความยุ่งยากในการใช้งานมากกว่าภาษาระดับสูง

ตัวอย่างของภาษาแอสเซมบลี

```
include "m168.h"
global main
main :                          แปลคำสั่ง
        ldi r16, Ob00000001
        out DDRB,r16            ; Set PBO to output
        out PORTB,r16           ; Set PBO high
        ldi r16, Ob00000101
        out TCCROB,r16          ; Set prescaler to 1024

Loop :
        in r17, TCNT0           ; if the counter is >= 128,
```

```
        cpi r17, 128                ; branch to dim
        brge dim                    ; otherwise continue to light
light :
        sbi PORTB, 0
        rjmp loop
dim :
        cbi PORTB,0
        rjmp loop
```

ภาษาซีสำหรับไมโครคอนโทรลเลอร์ Arduino

โครงสร้างภาษาซีของไมโครคอนโทรลเลอร์ Arduino

โครงสร้างภาษาซีของไมโครคอนโทรลเลอร์ Arduino ใช้รูปแบบการเขียนโปรแกรมของภาษา C++ แต่ละโปรแกรมต้องมีฟังก์ชันหลักอย่างน้อย 2 ฟังก์ชัน ได้แก่

1. ฟังก์ชัน setup() เป็นฟังก์ชันการกำหนดค่าต่าง ๆ ในส่วนนี้มีการกำหนดค่าเพียงครั้งเดียว เท่านั้น เช่น กำหนดขาในการใช้งานให้เป็นขาอินพุตหรือขาเอาต์พุต, การกำหนดค่าของการเรียกใช้ ไลบรารี

```
        void setup ()
        {
                //เป็นส่วนของคำสั่ง สำหรับกำหนดการทำงานในโปรแกรม และทำเพียงครั้งเดียว
        }
```

ตัวอย่าง

```
 int buttonPin = 3;
void setup ()
{
        Serial.begin(9600);
        pinMode (buttonPin , INPUT);
}
void loop()
{
        //...
}
```

2. ฟังก์ชัน loop () เป็นส่วนในการเขียนโปรแกรมและสั่งให้โปรแกรมทำงาน ซึ่งมีการทำงาน เป็นแบบวนลูปไปเรื่อย ๆ ตามการเขียนโปรแกรมของผู้พัฒนาโปรแกรมเพื่อรับค่าจากอินพุต นำค่าที่ ได้มาประมวลผล แล้วทำการส่งข้อมูลออกเอาต์พุตเพื่อควบคุมการทำงานตามโปรแกรม

```
void loop ()
{
        //เป็นโปรแกรมหลักของคำสั่ง ซึ่งในส่วนนี้โปรแกรมมีการทำงานตลอดเวลา
}
```

ตัวอย่าง

```
const int buttonPin = 3;
void setup ()
{
        Serial.begin(9600);
        pinMode (buttonPin, INPUT);
}
void loop ()
{
        if (digitalRead(button Pin) == HIGH)
                Serial.write ('H');
        else
                Serial.write ('L');
                delay(1000);
}
```

1. คำสั่งการควบคุม

เป็นคำสั่งให้ไมโครคอนโทรลเลอร์ทำงานตามเงื่อนไขหรือรูปแบบที่ผู้พัฒนาโปรแกรม ต้องการ มีคำสั่งต่าง ๆ ดังต่อไปนี้

- คำสั่ง if เป็นคำสั่งในการตรวจสอบเงื่อนไขการทำงานของโปรแกรม ถ้าเงื่อนไขเป็นจริง ให้ทำงานตามคำสั่งที่กำหนดนั้น มีรูปแบบคำสั่งดังนี้

```
if (เงื่อนไขที่ตรวจสอบ)
```

{

คำสั่งที่ให้ทำงาน เมื่อเงื่อนไขเป็นจริง

}

- คำสั่ง if...else เป็นคำสั่งกำหนดเงื่อนไขการทำงานของโปรแกรม โดยมี 2 เงื่อนไข ถ้า เงื่อนไขเป็นจริง ทำงานตามคำสั่งที่กำหนดแบบหนึ่ง ถ้าเงื่อนไขเป็นเท็จทำงานตามคำสั่งที่กำหนดอีก แบบหนึ่ง มีรูปแบบคำสั่งดังนี้

if (เงื่อนไขที่ตรวจสอบ)

{

คำสั่งที่ให้ทำงาน เมื่อเงื่อนไขเป็นจริง

}

else

{

คำสั่งที่ให้ทำงาน เมื่อเงื่อนไขเป็นเท็จ

}

- คำสั่ง for เป็นคำสั่งให้โปรแกรมทำงานซ้ำตามจำนวนรอบที่ต้องการมีรูปแบบคำสั่งคือ

for (ค่าเริ่มต้น, เงื่อนไขการทำซ้ำ, การเพิ่มหรือลดค่าตัวแปรในแต่ละรอบ)

{

คำสั่งที่ให้ทำงาน

}

- คำสั่ง Switch case เป็นคำสั่งเพื่อกำหนดการทำงานของโปรแกรมหลาย ๆ เงื่อนไข ถ้า ตัวแปรที่กำหนดตรงกับเงื่อนไขนั้น ๆ ทำให้โปรแกรมทำงานตามที่กำหนดไว้แต่ละเงื่อนไข มีรูปแบบ คำสั่งดังนี้

Switch (ตัวแปร ที่ต้องการตรวจสอบ)

{

case 1:

คำสั่งที่ให้ทำงาน เมื่อตรวจสอบว่า ตัวแปร == 1 break;

case 2:

คำสั่งที่ให้ทำงาน เมื่อตรวจสอบว่า ตัวแปร == 2

break;

default:

คำสั่งที่ให้ทำงาน เมื่อตรวจสอบว่า ตัวแปรไม่ตรงกับเงื่อนไขใด ๆ

}

- คำสั่ง while เป็นคำสั่งทำซ้ำแบบวนรอบ ถ้าเงื่อนไขเป็นจริงโปรแกรมทำงานตามคำสั่ง ที่เขียนไว้ในวงเล็บปีกกา แต่ถ้าเงื่อนไขเป็นเท็จโปรแกรมจบการทำงานในคำสั่ง while มีรูปแบบคำสั่ง

while (เงื่อนไขที่ตรวจสอบ)

{

คำสั่งที่ให้ทำงาน เมื่อเงื่อนไขยังเป็นจริง

}

- คำสั่ง do...while เป็นคำสั่งทำซ้ำแบบวนรอบ โดยมีการทำงานตรงกันข้ามกับคำสั่ง hile คือทำงานตามคำสั่งที่เขียนไว้ในวงเล็บปีกกา แล้วจึงมาตรวจสอบเงื่อนไข แต่ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจบการทำงานในคำสั่ง do มีรูปแบบคำสั่งดังนี้

do

{

คำสั่งที่ให้ทำงาน

} while (เงื่อนไขที่ตรวจสอบ)

- คำสั่ง break เป็นคำสั่งใช้ร่วมกับคำสั่งการทำงานแบบวนรอบ ได้แก่ คำสั่ง do, for, while หรือ Switch เพื่อให้โปรแกรมหยุดการทำงานจากการวนรอบโดยไม่มีเงื่อนไข

- คำสั่ง continue เป็นคำสั่งใช้สำหรับข้ามการทำงานของคำสั่งถัดไป คำสั่งนี้เขียนอยู่ใน คำสั่งการทำงานแบบวนรอบ ได้แก่ คำสั่ง do, for หรือ while

- คำสั่ง return เป็นคำสั่งจบการทำงานในโปรแกรมย่อย

- คำสั่ง goto เป็นคำสั่งกระโดดโดยไม่มีเงื่อนไขไปยังตำแหน่งที่กำหนด โดยอ้างถึงตำแหน่ง Label ที่กระโดดไป

2. ข้อกำหนดของไวยากรณ์

เป็นกฎเกณฑ์ในการสร้างประโยคขึ้นมาอธิบายความหมายของโปรแกรม มีดังต่อไปนี้

- เครื่องหมาย ; (เซมิโคล่อน) เป็นการจบคำสั่งในบรรทัดนั้น ๆ

ตัวอย่าง

int ledPin = 13;

ข้อควรระวัง ถ้าไม่ได้พิมพ์เครื่องหมาย ; ปิดท้ายคำสั่งในบรรทัดหนึ่ง ส่งผลให้โปรแกรมนั้นคอมไพล์โปรแกรมไม่ผ่าน ตามรูปที่ 3.1

รูปที่ 3.1 ตัวอย่างการแจ้งข้อผิดพลาดเมื่อไม่พิมพ์เครื่องหมาย ;

- เครื่องหมาย { } (วงเล็บปีกกา) เป็นการกำหนดบล็อกของคำสั่ง ใช้กับคำสั่ง if, else, while หรือ for มีรูปแบบคำสั่งดังนี้

ตำแหน่งการเขียนวงเล็บปีกกา

รูปแบบฟังก์ชัน (function)

```
void ฟังก์ชันต่าง ๆ (รูปแบบข้อมูล)
{
        คำสั่งที่ให้ทำงาน
}
```

รูปแบบฟังก์ชัน loop

```
while (นิพจน์บูลีน)
{
        คำสั่งที่ให้ทำงาน
}

do
{
        คำสั่งที่ให้ทำงาน
}
while (นิพจน์บูลีน);

for (ค่าเริ่มต้น เงื่อนไขการทำซ้ำ, การเพิ่มหรือลดค่าตัวแปรในแต่ละรอบ)
{
```

คำสั่งที่ให้ทำงาน

}

รูปแบบคำสั่งเงื่อนไข

if (นิพจน์บูลีน)

{

คำสั่งที่ให้ทำงาน

}

if (นิพจน์บูลีน)

{

คำสั่งที่ให้ทำงาน

}

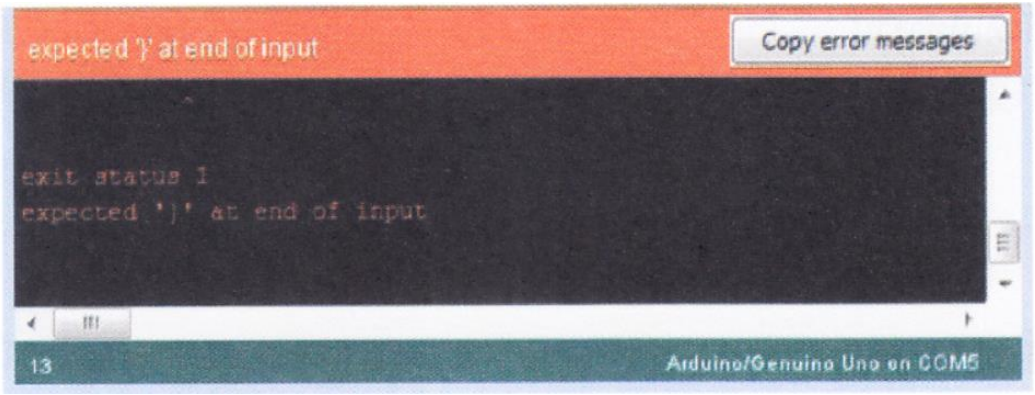
else

{

คำสั่งที่ให้ทำงาน

}

ข้อควรระวัง การใส่วงเล็บปีกกาต้องใส่ทั้งวงเล็บปีกกา { และ } ให้ครบคู่ กรณีที่ใส่ไม่ครบคู่ ส่งผลให้โปรแกรมนั้นคอมไพล์โปรแกรมไม่ผ่าน ตามรูปที่ 3.2



รูปที่ 3.2 ตัวอย่างการแจ้งข้อผิดพลาด กรณีใส่วงเล็บปีกกาไม่ครบๆ

- เครื่องหมาย // (หมายเหตุบรรทัดเดียว) เป็นส่วนของผู้เขียนโปรแกรมอธิบายเพิ่มเติม ในคำสั่งต่าง ๆ ว่าโปรแกรมทำงานอย่างไรในแต่ละบรรทัด เมื่อทำการคอมไพล์โปรแกรมประโยคที่อยู่ หลังเครื่องหมาย // ไม่ได้

ถูกนำไปคอมไพล์ด้วย การใส่เครื่องหมาย // (หมายเหตุ) มีประโยชน์สำหรับ การตรวจสอบโปรแกรมภายหลัง หรือให้ผู้พัฒนาโปรแกรมท่านอื่นสามารถเข้าใจการเขียนโปรแกรม นั้น ๆ ได้

ตัวอย่าง

```
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
    }
```

- เครื่องหมาย /* */ (หมายเหตุหลายบรรทัด) เป็นส่วนของผู้เขียนโปรแกรมอธิบาย เพิ่มเติม ในคำสั่งต่าง ๆ ว่าโปรแกรมทำงานอย่างไรสามารถอธิบายได้หลายบรรทัด

ตัวอย่าง

```
/* this is multiline comment-use it to comment out whole blocks of code
if (gwb == 0) { // single line comment is OK inside a multiline comment
X = 3;          /* but not another multiline comment - this is invalid */
}
    // don't forget the "closing" comment - they have to be balanced!
*/
```

- เครื่องหมาย #define เป็นคำสั่งในการกำหนดค่าคงที่ให้กับโปรแกรม มีรูปแบบคำสั่ง

#define ชื่อตัวแปรค่าคงที่ ค่าคงที่

ตัวอย่าง

```
#define ledpin 7
```

เป็นการกำหนดให้ตัวแปร ledpin มีค่าคงที่เท่ากับ 7

- เครื่องหมาย #include เป็นคำสั่งให้นำไฟล์อื่นเข้ามาร่วมกับไฟล์โปรแกรมหลัก มี รูปแบบคำสั่งดังนี้

#include <ชื่อไฟล์>

ตัวอย่าง

#include <avr/io.h>

เป็นการเรียกไฟล์ avr/io.hมาร่วมกับไฟล์โปรแกรมหลัก โดยปกติแล้วไฟล์มาตรฐานมาพร้อม กับโปรแกรม Arduino

3. การดำเนินการทางคณิตศาสตร์

เป็นเครื่องหมายทางคณิตศาสตร์ ในการเขียนโปรแกรม เพื่อหาผลลัพธ์จากการคำนวณ ซึ่งสามารถกระทำกับข้อมูลได้หลายรูปแบบ มีเครื่องหมายดังต่อไปนี้

- เครื่องหมาย + เป็นการบวกของตัวถูกกระทำสองตัว
- เครื่องหมาย เป็นการลบของตัวถูกกระทำสองตัว
- เครื่องหมาย * เป็นการคุณของตัวถูกกระทำสองตัว
- เครื่องหมาย / เป็นการหารของตัวถูกกระทำสองตัว
- เครื่องหมาย % เป็นการหารเอาเศษ ใช้หาค่าเศษที่ได้จากการหาร

4. การดำเนินการเปรียบเทียบ

เป็นเครื่องหมายที่ใช้ในการเปรียบเทียบทางคณิตศาสตร์ มีเครื่องหมายดังต่อไปนี้

- เครื่องหมาย == เป็นการเปรียบเทียบเท่ากับ
- เครื่องหมาย != เป็นการเปรียบเทียบไม่เท่ากับ
- เครื่องหมาย < เป็นการเปรียบเทียบน้อยกว่า
- เครื่องหมาย > เป็นการเปรียบเทียบมากกว่า
- เครื่องหมาย <= เป็นการเปรียบเทียบน้อยกว่าหรือเท่ากับ
- เครื่องหมาย >= เป็นการเปรียบเทียบมากกว่าหรือเท่ากับ

5. การดำเนินการทางตรรกะ

เป็นเครื่องหมายที่ใช้เชื่อมเงื่อนไข 2 เงื่อนไข หรือมากกว่า เพื่อให้การเปรียบเทียบมี ความละเอียดมากขึ้น โดยใช้สัญลักษณ์แทนในแต่ละเครื่องหมาย มีเครื่องหมายดังต่อไปนี้

- เครื่องหมาย 88 ให้ค่าเป็นจริง เมื่อผลการเปรียบเทียบค่าทั้งสองเป็นจริงทั้งคู่

  ตัวอย่าง (A < 10) && (B > 5)

- เครื่องหมาย || ให้ค่าเป็นจริงเมื่อผลการเปรียบเทียบค่าทั้งสองเป็นจริงทั้งคู่หรือตัวแปรใดตัวหนึ่งเป็นจริง

  ตัวอย่าง (A < 10) || (B > 5)

- เครื่องหมาย ! ให้ค่าเป็นจริง เมื่อผลการเปรียบเทียบเป็นเท็จ

ตัวอย่าง I(A < 10)

6. การดำเนินการระดับบิต

เป็นการนำค่าแต่ละบิตของตัวแปรเดียวหรือ 2 ตัวมากระทำกันในระดับบิต มีเครื่องหมาย ดังต่อไปนี้

- เครื่องหมาย & เป็นการแอนด์ระดับบิต
- เครื่องหมาย เป็นการออร์ระดับบิต
- เครื่องหมาย ^ เป็นการเอ็กคลูซีฟออร์ระดับบิต
- เครื่องหมาย ~ เป็นการนอตระดับบิต
- เครื่องหมาย << เป็นการเลื่อนบิตไปทางซ้าย
- เครื่องหมาย >> เป็นการเลื่อนบิตไปทางขวา

7. ตัวดำเนินการ

เป็นเครื่องหมายทางคณิตศาสตร์ของตัวแปรเดียว เพื่อหาผลลัพธ์จากการคำนวณนั้น มี เครื่องหมาย ดังต่อไปนี้

- เครื่องหมาย ++ เป็นการบวกครั้งละ 1 นำผลลัพธ์เก็บในตัวแปรเดิม

  ตัวอย่าง X++ ความหมาย X = X + 1
- เครื่องหมาย - เป็นการลบครั้งละ 1 นำผลลัพธ์เก็บในตัวแปรเดิม

  ตัวอย่าง X -- ความหมาย x = X - 1
- เครื่องหมาย += เป็นการการบวก นำผลลัพธ์เก็บในตัวแปรเดิม

  ตัวอย่าง X += 2 ความหมาย X = X + 2
- เครื่องหมาย -= เป็นการลบ นำผลลัพธ์เก็บในตัวแปรเดิม

  ตัวอย่าง X = 2 ความหมาย X = X - 2
- เครื่องหมาย *= เป็นการคูณ นำผลลัพธ์เก็บในตัวแปรเดิม

  ตัวอย่าง X - 2 ความหมาย x = x * 2
- เครื่องหมาย /= เป็นการหาร นำผลลัพธ์เก็บในตัวแปรเดิม

  ตัวอย่าง x/= 2 ความหมาย x = X / 2
- เครื่องหมาย %ะ เป็นการหารเอาเศษ นำผลลัพธ์เก็บในตัวแปรเดิม

ตัวอย่าง x 96= 2 ความหมาย x = X 6 2

- เครื่องหมาย &= เป็นการแอนด์ นำผลลัพธ์เก็บในตัวแปรเดิม

ตัวอย่าง x &= 2 ความหมาย X = x & 2

- เครื่องหมาย |= เป็นการออร์ นำผลลัพธ์เก็บในตัวแปรเดิม

ตัวอย่าง x = 2 ความหมาย X = X | 2

- เครื่องหมาย AE เป็นการเอ็กคลูซีฟออร์ นำผลลัพธ์เก็บในตัวแปรเพิ่ม

ตัวอย่าง x 4 = 2 ความหมาย X = X A 2

- เครื่องหมาย <<= เป็นการเลื่อนบิตไปทางซ้าย นำผลลัพธ์เก็บในตัวแปรเดิม

ตัวอย่าง X<<= 2 ความหมาย X = x 442

- เครื่องหมาย >>= เป็นการเลื่อนบิตไปทางขวา นำผลลัพธ์เก็บในตัวแปรเดิม

ตัวอย่าง X>>= 2 ความหมาย X = X >> 2

ตัวแปร

ตัวแปร เป็นชื่อเรียกแทนพื้นที่เก็บข้อมูลในหน่วยความจำของไมโครคอนโทรลเลอร์ โดยมีการตั้งชื่อเรียกหน่วยความจำในตำแหน่งนั้น เพื่อความสะดวกในการเรียกใช้ข้อมูล ซึ่งมีชนิดของข้อมูล หรือแบบของตัวแปรต่าง ๆ ดังนี้

1. ค่าคงที่

เป็นคำสั่งข้อความที่กำหนดไว้ในโปรแกรม Arduino มีคำสั่งดังต่อไปนี้

- คำสั่ง HIGH/LOW แทนสถานะลอจิก “1” กับลอจิก “0”
- คำสั่ง INPUT/OUTPUT ใช้สำหรับกำหนดค่าอินพุตกับเอาต์พุต
- คำสั่ง true/false เป็นค่าคงที่แบบบูลีน โดย true แทนสภาวะค่าใด ๆ ที่ไม่ใช่ 0 ถือว่า เป็นจริง ส่วน false มีค่าเป็น 0 หรือเป็นเท็จ
- คำสั่ง integer constants เป็นค่าคงที่ของเลขจำนวนเต็ม
- คำสั่ง floating point constants เป็นค่าคงที่ของเลขทศนิยม

2. ชนิดของข้อมูล

สามารถแบ่งชนิดของข้อมูลได้ดังนี้

- void ใช้เฉพาะในการประกาศฟังก์ชัน
- boolean มีค่าจริงหรือเท็จ
- char มีค่าตั้งแต่ 127 ถึง 127 ใช้สำหรับเก็บข้อมูลที่เป็นตัวอักษร
- unsigned char มีค่าตั้งแต่ 0 ถึง 255

- byte มีค่าตั้งแต่ 0 ถึง 255
- int มีค่าตั้งแต่ -32,767 ถึง 32,767
- unsigned int มีค่าตั้งแต่ 0 ถึง 65,555
- word มีค่าตั้งแต่ 0 ถึง 65,555
- long มีค่าตั้งแต่ -2,147,483,548 ถึง 2,147,483,647
- unsigned long มีค่าตั้งแต่ 0 ถึง 4,294,967,295
- float มีค่าตั้งแต่ 3.4028235E+38 ถึง 3.4028235E+38
- double มีค่าตั้งแต่ 3.4028235E+38 ถึง 3.4028235E+38
- string ตัวแปรสำหรับเก็บข้อความ
- array ตัวแปรหลายตัวที่ถูกเก็บรวมไว้ในตัวแปรชื่อเดียวกัน

ชุดคำสั่ง

ชุดคำสั่ง เป็นชุดคำสั่งในการเขียนโปรแกรมเพื่อให้ไมโครคอนโทรลเลอร์ทำงานตามโปรแกรม ที่ออกแบบไว้ โดยมีคำสั่งต่าง ๆ ดังนี้

1. คำสั่งดิจิตอล อินพุต/เอาต์พุต

มีคำสั่งดังต่อไปนี้

- คำสั่ง pinMode() เป็นการกำหนดพอร์ตเป็นอินพุตหรือเอาต์พุต
- คำสั่ง digitalWrite() เป็นการเขียนข้อมูลออกพอร์ตที่กำหนด
- คำสั่ง digitalRead() เป็นการอ่านข้อมูลเข้าพอร์ตที่กำหนด

2. คำสั่งอนาล็อก อินพุต/เอาต์พุต

มีคำสั่งดังต่อไปนี้

- คำสั่ง analogReference() เป็นการกำหนดค่าแรงต้นอ้างอิงที่ใช้สำหรับอนาล็อกอินพุต
- คำสั่ง analogRead() เป็นการอ่านแรงดันไฟฟ้าแบบอนาล็อกและแปลงเป็นจำนวนเต็ม มีค่าระหว่าง 0 ถึง 1023
- คำสั่ง analogWrite() เป็นการใช้ PWM เขียนค่าออกทางพอร์ตที่กำหนด

3. คำสั่งเวลา

มีคำสั่งดังต่อไปนี้

- คำสั่ง mitis() เป็นการหน่วงเวสามีหน่วยเป็นมิลลิวินาทีของ Arduino ทันทีที่มีไฟเลี้ยง เข้า Arduino

- คำสั่ง micros() เป็นการหน่วงเวลามีหน่วยเป็นไมโครวินาทีของ Arduino ทันทีที่มีไฟ เลี้ยงเข้า Arduino

- คำสั่ง delay() เป็นการหน่วงเวลาตามค่าที่กำหนด มีหน่วยเป็นมิลลิวินาที

- คำสั่ง delayMicroseconds) เป็นการหน่วงเวลาตามค่าที่กำหนด มีหน่วยเป็นไมโครวินาที

4. คำสั่งคณิตศาสตร์

มีคำสั่งดังต่อไปนี้

- คำสั่ง min() เป็นการหาค่าต่ำสุด
- คำสั่ง max() เป็นการหาค่ามากสุด
- คำสั่ง abs() เป็นการหาค่าสมบูรณ์ของตัวแปร
- คำสั่ง constrain() เป็นการตรวจสอบและการปรับค่าตัวแปรที่กำหนด
- คำสั่ง map() เป็นการปรับค่าตัวแปรจากเดิมให้อยู่ระหว่างค่าที่กำหนด
- คำสั่ง pow() เป็นการหาค่าของตัวเลขยกกำลัง
- คำสั่ง sqrt() เป็นการหาค่าของรากที่สองของตัวเลข

5. คำสั่งตรีโกณมิติ

มีคำสั่งดังต่อไปนี้

- คำสั่ง sin() เป็นการคำนวณหาค่า sin
- คำสั่ง cos() เป็นการคำนวณหาค่า CO5
- คำสั่ง tan() เป็นการคำนวณหาค่า tan

6. คำสั่งสุ่มตัวเลข

มีคำสั่งดังต่อไปนี้

- คำสั่ง randomSeed() เป็นการกำหนดค่าเริ่มต้นของฟังก์ชั่น random
- คำสั่ง random) เป็นการสุ่มค่าตัวเลขระหว่างตัวเลขที่กำหนด

7. คำสั่งบิตและไบต์

มีคำสั่งดังต่อไปนี้

- คำสั่ง lowByte() เป็นตัวแปรของไบต์ต่ำสุด
- คำสั่ง highByte() เป็นตัวแปรของไบต์สูงสุด
- คำสั่ง bitRead() เป็นการอ่านบิตของตัวแปร
- คำสั่ง bitWrite() เป็นการเขียนบิตของตัวแปร
- คำสั่ง bitSet() เป็นการตั้งบิตของตัวแปรเท่ากับ 1
- คำสั่ง bitClear() เป็นการตั้งบิตของตัวแปรเท่ากับ 0

- คำสั่ง bit() เป็นการตั้งค่าบิตตามค่าที่กำหนด

8. คำสั่งอินเตอร์รัพท์ภายนอก

มีคำสั่งดังต่อไปนี้

- คำสั่ง attachInterrupt() เป็นคำสั่งกำหนด และสร้างอินเตอร์รัพท์
- คำสั่ง detachInterrupt() เป็นคำสั่งปิดการรับอินเตอร์รัพท์

9. คำสั่งการอินเตอร์รัพท์

มีคำสั่งดังต่อไปนี้

- คำสั่ง interrupts() เป็นการเปิดการใช้งานอินเตอร์รัพท์
- คำสั่ง noInterrupts() เป็นการปิดการใช้งานอินเตอร์รัพท์

10. คำสั่งการติดต่อสื่อสาร

มีคำสั่งดังต่อไปนี้

- คำสั่ง Serial.begin() เป็นการกำหนดอัตราการส่งข้อมูล
- คำสั่ง Serial.end() เป็นการปิดใช้งานการสื่อสารแบบอนุกรม
- คำสั่ง Serial.available() เป็นการตรวจสอบการรับข้อมูลจากการสื่อสารแบบอนุกรม
- คำสั่ง Serial.read() เป็นการอ่านข้อมูลจากการสื่อสารแบบอนุกรมที่เข้ามา
- คำสั่ง Serial.peek() เป็นการส่งกลับไบต์ต่อไปของข้อมูลการสื่อสารแบบอนุกรม
- คำสั่ง Serial.flush() เป็นการลบข้อมูลทั้งหมดในบัฟเฟอร์
- คำสั่ง Serial.print() เป็นการพิมพ์ข้อมูลไปยังพอร์ตอนุกรม
- คำสั่ง Serial.println() เป็นการพิมพ์ข้อมูลไปยังพอร์ตอนุกรม และขึ้นบรรทัดใหม่
- คำสั่ง Serial.write() เป็นการส่งข้อมูลไบต์ไปยังพอร์ตอนุกรม

ตัวอย่าง การเขียนโปรแกรมภาษาซีสำหรับไมโครคอนโทรลเลอร์ Arduino

```
const int kPinLed = 13;
void setup()
{
        pinMode(KPinLed, OUTPUT);
}
void loop()
{
        digitalWrite(kPinLed, HIGH);
        delaM500);
```

```
        digitalWrite(kPinLed, LOW);
        delay(500);
}
```

คำอธิบาย

```
const int kPinLed = 13;                 // เป็นการกำหนดค่าคงที่ ซึ่งกำหนดครั้งเดียวใช้ได้ตลอดทั้งโปรแกรม
void setup()
{
        pinMode(kPinLed, OUTPUT);   //เป็นการกำหนดขาที่เชื่อมต่อวงจรให้เป็นพอร์ตเอาต์พุต
}
void loop()
{
        digitalWrite(kPinLed, HIGH);
        delay(500);
        digitalWrite(kPinLed, LOW);
        delay(500);
}
```

//อธิบายการทำงานของโปรแกรม เป็นการส่งข้อมูลลอจิก “1” ออกไปยังพอร์ตเอาต์พุตที่ กำหนด แล้วหน่วงเวลา 500 มิลลิวินาที จากนั้นส่งข้อมูลลอจิก “0” ออกไปยังพอร์ตเอาต์พุต แล้ว ทำการหน่วงเวลา 500 มิลลิวินาที เช่นเดิม ซึ่งโปรแกรมทำงานวนในลูปของ void loop เช่นนี้ตลอด ทั้งโปรแกรม

สรุป

การเขียนโปรแกรมไมโครคอนโทรลเลอร์ Arduino สามารถเขียนได้ทั้งภาษาแอสเซมบลี และ ภาษาซี ขึ้นอยู่กับว่าผู้เขียนโปรแกรมเลือกใช้ภาษาใด มาใช้สำหรับการพัฒนาโปรแกรม ซึ่งแต่ละภาษา เมื่อผู้เขียนโปรแกรมทำการเขียนโปรแกรมเสร็จสิ้นแล้ว ต้องทำการแปลจากภาษาที่เขียนขึ้นให้เป็นภาษาเครื่องซึ่งเป็นภาษาที่ไมโครคอนโทรลเลอร์สามารถทำงานได้ ซึ่งในบทนี้เน้นเนื้อหาในการเขียน โปรแกรมด้วยภาษาซี ดังนั้นผู้เขียนโปรแกรมต้องศึกษาโครงสร้างและการทำงานของคำสั่งต่าง ๆ ของ ภาษาซีสำหรับไมโครคอนโทรลเลอร์ Arduino