บทที่ 8

เรื่อง การใช้โปรแกรมทดสอบการทำงานของหุ่นยนต์

8.1 โครงสร้างการเขียนโปรแกรมภาษาซีของ Arduino

ภาษาซีของ Arduino จะจัดแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมออกเป็นส่วนหลายๆส่วน โดยเรียกแต่ละส่วนว่า ฟังก์ชัน และ เมื่อนำฟังก์ชันมารวมเข้าด้วยกัน ก็จะเรียกว่าโปรแกรม โดยโครงสร้างการเขียนโปรแกรมของ Arduino นั้น ทุกๆโปรแกรมจะต้องประกอบไปด้วยฟังก์ชันจำนวนเท่าใดก็ได้ แต่อย่างน้อยที่สุดต้องมีฟังก์ชัน จำนวน 2 ฟังก์ชัน คือ setup( ) และ loop( ) ดังตัวอย่าง

```
#include<Servo.h>              //สั่งผนวกไฟล์ชื่อ Servo.h เข้ามาในโปรแกรม
int Servo1=9;                  //กำหนดให้ Servo1 แทน Pin Digital-9
Servo myservo;                 //สร้าง object ชื่อ myservo เพื่อควบคุม Servo
void setup()
{
      myservo.attach(Servo1);    //กำหนดให้ใช้ขา Digital-9 สร้างสัญญาณควบคุม Servo
}

void loop()
{
      myservo.write(180);          //กำหนดค่าตำแหน่งให้กับ Servo=180 องศา
}
```

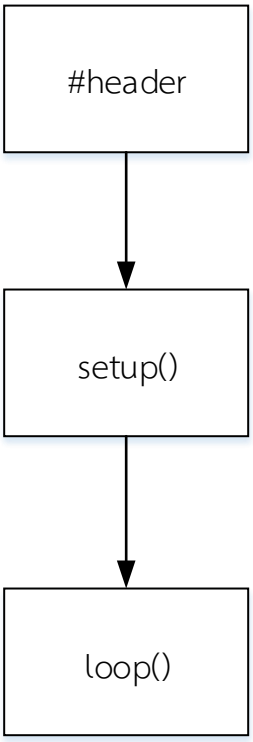
จะเห็นได้ว่าโครงสร้างพื้นฐานของภาษาซีที่ใช้กับ Arduino นั้น จะประกอบไปด้วย 3 ส่วนใหญ่ด้วยกัน คือ

1. header ในส่วนนี้จะมีหรือไม่มีก็ได้ ถ้ามีต้องกำหนดไว้ในส่วนเริ่มต้นของโปรแกรม ซึ่งส่วนของ Header ได้แก่ ส่วนที่เป็น Compiler Directive ต่างๆ รวมไปถึงส่วนของการประกาศตัวแปร และ ค่าคงที่ต่างๆที่จะใช้ในโปรแกรม

2. setup ( ) ในส่วนนี้เป็นฟังก์ชันบังคับที่ต้องกำหนดให้มีในทุกๆโปรแกรม ถึงแม้ว่าในบางโปรแกรมจะ ไม่ต้องการใช้งานก็ยังจำเป็นต้องประกาศไว้ด้วยเสมอ เพียงแต่ไม่ต้องเขียนคำสั่งใดๆไว้ในระหว่าง วงเล็บปีกกา { } ที่ใช้เป็นตัวกำหนดขอบเขตของฟังก์ชัน โดยฟังก์ชันนี้จะใช้สำหรับบรรจุคำสั่งใน ส่วนที่ต้องการให้โปรแกรมทำงานเพียงรอบเดียวตอนเริ่มต้นทำงานของโปรแกรมครั้งแรก

เท่านั้น ซึ่ง ได้แก่ คำสั่งเกี่ยวกับการ setup ค่าการทำงานต่างๆ เช่น การกำหนดหน้าที่การใช้งานของ Pin Mode และการกำหนดค่า Baudrate สำหรับใช้งานพอร์ตสื่อสารอนุกรม เป็นต้น

3. loop ( ) เป็นส่วนฟังก์ชันบังคับที่ต้องกำหนดให้มีในทุกๆโปรแกรมเช่นเดียวกันกับฟังก์ชัน setup ( ) โดยฟังก์ชัน loop() นี่จะใช้บรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเป็นวงรอบซ้ำๆ กันไปไม่รู้จบ ซึ่งถ้าเปรียบเทียบกับรูปแบบของ ANSI - C ส่วนนี่ก็คือ ฟังก์ชัน main ( ) นั้นเอง



รูปที่ 8.1 แสดงโครงสร้างโปรแกรมของ Arduino

จะเห็นได้ส่วนแรกซึ่งถือเป็นส่วนเริ่มต้นของโปรแกรม ซึ่งจะเรียกว่า Header โดยประกอบด้วยคำสั่ง #include ซึ่งเป็นคำสั่งพิเศษที่เรียกว่า Compiler Directive ซึ่งมันไม่ใช่คำสั่งสำหรับสั่งงานในโปรแกรมดังนั้นคำสั่งนี้จึงไม่ต้องมีเครื่องหมายเซมิโคลอนปิดท้ายคำสั่งเหมือนคำสั่งอื่นๆ โดย Compiler Directiveจะใช้ทำหน้าที่สำหรับบอกให้ Compiler รับรู้เงื่อนไขในการแปลคำสั่งเท่านั้น ซึ่งในกรณีคำสั่ง #include ใช้สำหรับบอกให้ Compiler รับรู้ว่าในการแปลคำสั่งของโปรแกรมนี้ มีไฟล์ภายนอกใดบ้างที่จำเป็นต้องใช้ ร่วมในการแปลคำสั่งให้กับโปรแกรมนี้ โดยจากตัวอย่างข้างต้นจะเป็นการบอกให้ Compiler ทำการผนวกไฟล์ชื่อ "Servo.h" เข้ามาใช้เพื่อเรียกใช้คำสั่งต่างๆที่บรรจุไว้เข้ามาใช้งานในโปรแกรม โดยใช้รูปแบบ

```
#include<Servo.h>
```

โดยเมื่อพบคำสั่ง #include ตัวแปลภาษาของ Arduino จะไปค้นหาไฟล์ที่ระบุไว้ในเครื่องหมาย<>หลังคำสั่ง #include จากตำแหน่ง Directory ที่เก็บรวบรวม Library ของโปรแกรม Arduinoโดยส่วนของ Header จะนับรวมไปถึงคำสั่งส่วนที่ใช้ประกาศสร้าง ตัวแปร(Variable Declaration)และค่าคงที่ (Constant Declaration) รวมทั้ง ฟังก์ชันต่างๆ (Function Declaration) ด้วย

```
int Servo1=9;
Servo myservo;
```

สำหรับส่วนที่มีความสำคัญและจำเป็นที่สุดของโปรแกรม Arduino ที่จำเป็นจะขาดไม่ได้ในการเขียนโปรแกรมของ Arduino คือ ฟังก์ชัน setup ( ) และ ฟังก์ชัน loop ( ) โดยฟังก์ชันทั้งสองมีโครงสร้างเหมือนกัน แต่จะเขียนฟังก์ชัน setup ( ) ไว้ก่อน ฟังก์ชัน loop ( ) ซึ่งทั้งสองฟังก์ชัน มีขอบเขต เริ่มต้นและสิ้นสุด ภายใต้เครื่องหมาย { }

```
void setup()
{
  คำสั่งค่าต่างๆที่ต้องการเขียนภายใต้ฟังก์ชัน setup ( )
}
```

หน้าที่ของฟังก์ชัน setup ( ) ใน Arduino คือใช้ทำหน้าที่เป็นส่วนของโปรแกรมย่อยสำหรับบรรจุคำสั่งต่างๆ ที่ใช้สำหรับกำหนดการทำงานของระบบ หรือ กำหนดคุณสมบัติการทำงานให้อุปกรณ์ต่างๆซึ่งคำสั่งทั้งหมดที่บรรจุไว้ภายใต้ฟังก์ชัน setup ( ) จะถูกเรียกขึ้นมาทำงานเพียงรอบเดียวคือตอนเริ่มต้นการทำงานของโปรแกรม (หลังการรีเซตใช้ MCU เริ่มทำงานเท่านั้น) โดยคำสั่งที่นิยมบรรจุไว้ในฟังก์ชั่นส่วนนี้ได้แก่ คำสั่งสำหรับกำหนดโหมดการทำงาน Digital Pin หรือ คำสั่งสำหรับกำหนดคุณสมบัติของพอร์ตสื่อสารอนุกรม

```
void setup()
{
คำสั่งค่าต่างๆที่ต้องการเขียนภายใต้ฟังก์ชัน setup ( )
}
```

หน้าที่ของฟังก์ชัน loop ( ) ใน Arduino คือใช้ทำหน้าที่เป็นส่วนของโปรแกรมหลัก สำหรับใช้บรรจุคำสั่งควบคุมการทำงานต่างๆโปรแกรม ที่ต้องการใช้โปรแกรมทำงาน โดยคำสั่งที่บรรจุไว้ในฟังก์ชันนี้จะถูกเรียกขึ้นมาทำงานซ้ำๆ กันตามลำดับและเงื่อนไขที่กำหนดไว้

8.2 ตัวแปรใน Arduino

ตัวแปร หมายถึง กลุ่มของ ตัวอักษร ตัวเลข และ เครื่องหมายใดๆ ที่รวมกันเป็นชื่อ เพื่อใช้กำหนดเป็นตัวแทนของค่าข้อมูลที่เราต้องการจะอ้างถึงในโปรแกรม ทั้งนี้ก็เนื่องจากว่าในการทำงานของโปรแกรมจริงๆนั้นจะใช้ค่าตัวเลขที่ผู้ใช้กำหนดให้มาทำการประมวลผล ซึ่งในการเขียนโปรแกรมถ้าเราต้องเขียนโปรแกรมโดยกำหนดเป็นค่าตัวเลขให้กับโปรแกรมตรงๆเลย ก็จะทำให้โปรแกรมที่เราเขียนขึ้นเต็มไปด้วยค่าตัวเลขต่างๆมากมาย ซึ่งยากต่อการอ่าน ยากต่อการทำความเข้าใจ และยากต่อการตรวจสอบความถูกต้องและอาจทำให้เกิดความผิดพลาดได้ง่ายด้วย ดังนั้นทุกภาษา จึงยอมให้มีการกำหนดชื่อ ขึ้นมาใช้งานแทนคาตัวเลขเพื่อให้เขียนโปรแกรมได้สะดวกและง่ายต่อการอ่าน ทำความเข้าใจได้มากยิ่งขึ้น ซึ่งลักษณะของข้อมูล อาจมีทั้งแบบที่เป็นค่าซึ่งสามารถเปลี่ยนแปลงได้ (variable) หรือ อาจเป็นแบบที่มีค่าคงที่ไมสามารถเปลี่ยนแปลงได้ (constant) ในการประกาศใช้งานตัวแปรจำเป็นต้องประกาศ ชนิดของตัวแปร หรือบางครั้งอาจมีการกำหนดค่าเริ่มต้นให้กับตัวแปรด้วยก็ได้

8.3 ชนิดและประเภทของตัวแปร

ถ้าหากว่าเราจะเปรียบเทียบว่า ตัวแปร คือ ภาชนะสำหรับบรรจุสิ่งของ และ ข้อมูล คือ สิ่งของที่เรา ต้องการจะเก็บ จะเห็นได้ว่า สิ่งของต่างๆรอบๆตัวเรานั้น จะมีคุณสมบัติที่แตกต่างกันไป ดังนั้นในการเลือก ภาชนะสำหรับใช้บรรจุสิ่งของ เราก็จำเป็นต้องเลือกชนิดของภาชนะให้มีความเหมาะสมที่จะใช้เก็บสิ่งของด้วย ซึ่งสิ่งแรกที่ต้องพิจารณาคือ เราจะต้องรู้จักคุณสมบัติของสิ่งของที่ต้องการจะจัดเก็บ และ จุดประสงค์การใช้งานก่อน จากนั้นจึงจัดหาภาชนะที่มี ขนาด และ รูปทรงของภาชนะ เหมาะสมที่จะใช้เก็บสิ่งของ

เพื่อให้สามารถจัดเก็บและนำสิ่งของออกมาใช้งานได้อย่างง่ายประหยัดมากที่สุด ในภาษาชีนั้น มีการกำหนด และ จำแนก ชนิดของตัวแปร ไว้เป็น 5 ชนิดด้วยกัน โดยแต่ละชนิดจะมี คุณสมบัติการใช้งานที่ต่างกันเพื่อใช้ในการเก็บข้อมูลที่มีรูปแบบแตกต่างกันคือ

Char ใช้เก็บข้อมูลที่เป็นตัวอักษร (character) ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็มได้ 256 ค่า

int ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็ม(integer) ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็มได้ 65536 ค่า

float ใช้เก็บข้อมูลที่เป็นเลขทศนิยมแบบ Single Precision

double ใช้เก็บข้อมูลที่เป็นเลขทศนิยมแบบ Double Precision ซึ่ง สามารถเก็บค่าตัวเลขทศนิยมที่มีความละเอียดและถูกต้องของทศนิยมมากว่าแบบ floatถึง 2 เท่า

ชนิดของตัวแปร	ขนาด (bits)	ขอบเขต	ข้อมูลที่เก็บ
char	8	-128 ถึง 127	ข้อมูลชนิดอักขระ ใช้เนื้อที่ 1 byte
unsigned char	8	0 ถึง 255	ข้อมูลชนิดอักขระ ไม่คิดเครื่องหมาย
int	16	-32,768 ถึง 32,767	ข้อมูลชนิดจำนวนเต็ม ใช้เนื้อที่ 2 byte
unsigned int	16	0 ถึง 65,535	ข้อมูลชนิดจำนวนเต็ม ไม่คิดเครื่องหมาย
short	8	-128 ถึง 127	ข้อมูลชนิดจำนวนเต็มแบบสั้น ใช้เนื้อที่ 1 byte
unsigned short	8	0 ถึง 255	ข้อมูลชนิดจำนวนเต็มแบบสั้น ไม่คิดเครื่องหมาย
long	32	-2,147,483,648 ถึง 2,147,483,649	ข้อมูลชนิดจำนวนเต็มแบบยาว ใช้เนื้อที่ 4 byte
unsigned long	32	0 ถึง 4,294,967,296	ข้อมูลชนิดจำนวนเต็มแบบยาว ไม่คิดเครื่องหมาย
float	32	3.4*10e(-38) ถึง 3.4*10e(38)	ข้อมูลชนิดเลขทศนิยม ใช้เนื้อที่ 4 byte
double	64	3.4*10e(-308) ถึง 3.4*10e(308)	ข้อมูลชนิดเลขทศนิยม ใช้เนื้อที่ 8 byte
long double	128	3.4*10e(-4032) ถึง 1.1*10e(4032)	ข้อมูลชนิดเลขทศนิยม ใช้เนื้อที่ 16 byte

8.3 ตัวดำเนินการของ Arduino

8.3.1 ตัวดำเนินการทางคณิตศาสตร์

เครื่องหมาย	ความหมาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	Mod หรือการหารแบบเอาเศษมาใช้

8.3.2 ตัวดำเนินการเปรียบเทียบ

เราสามารถกำหนดการทำงานของโปรแกรมให้เป็นไปตามเงื่อนไขต่างๆ ที่ต้องการได้ โดยตรวจสอบเงื่อนไขด้วยตัวดำเนินการเปรียบเทียบ

เครื่องหมาย	ความหมาย
==	เท่ากับ
!=	ไม่เท่ากับ
<	น้อยกว่า
<=	น้อยกว่าหรือเท่ากับ
>	มากกว่า
>=	มากกว่าหรือเท่ากับ

8.3.3 ตัวดำเนินการกำหนดค่า

เราสามารถกำหนดค่าให้กับตัวแปรได้ด้วยตัวดำเนินการกำหนดค่า

เครื่องหมาย	ความหมาย
=	ใช้กำหนดค่าให้กับตัวแปร
++	การเพิ่มค่า x = x+1
- -	การลดค่า x = x-1
+=	การเพิ่มค่าเท่ากับค่าทางขวา
-=	การลดค่าเท่ากับค่าทางขวา
*=	นำตัวเองคูณค่าเท่ากับค่าทางขวา
/=	นำตัวเองหารค่าเท่ากับค่าทางขวา
%=	นำตัวเอง mod ค่าเท่ากับค่าทางขวา

8.3.4 ตัวดำเนินการตรรกศาสตร์

เมื่อต้องการนำผลจากการตรวจสอบเงื่อนไขมาใช้ร่วมกัน เราสามารถนำตัวดำเนินการตรรกศาสตร์ (AND, OR, NOT) มาใช้ได้

เครื่องหมาย	ความหมาย
&&	AND
||	OR
!	NOT

8.4 การใช้งาน Digital Input / Output

คำสั่งในกลุ่มนี้เป็นกลุ่มคำสั่งคำสั่ง สำหรับใช้งาน Pin I/O ของ Arduino ในแบบของ digital I/O ซึ่งตามปรกติแล้วในการจะกำหนดหน้าที่ใช้งานขาสัญญาณ ของไมโครคอนโทรลเลอร์ AVR ซึ่งนิยมเรียกกันว่า Pin I/O นั้นเราจะต้องเข้าไปกำหนดค่าให้กับรีจิสเตอร์ต่างๆในตัว MCU โดยตรง เพื่อเลือกกำหนดรูปแบบของการทำงานของขาสัญญาณ PIN I/O ต่างๆ ให้มีคุณสมบัติตามที่เราต้องการ ซึ่งกรรมวิธี และขั้นตอนดังกล่าวข้างต้นจะมีความยุ่งยากซับซ้อนพอสมควร และ อาจดูเป็นเรื่องลำบากสำหรับผู้เล่นเริ่มใช้ไมโครคอนโทรลเลอร์ด้วยซ้ำไป แต่ใน Arduino นั้นได้มีการสร้างฟังก์ชั่นหรือคำสั่ง สำหรับช่วยลดความยุ่งยากซับซ้อนตรงนี้ไว้ได้แล้ว ทำให้ผู้ใช้สามารถกำหนดหน้าที่ของการทำงาน และสั่งงาน Pin I/O ต่างๆของ MCU ได้โดยง่าย

สำหรับ digital I/O ใน Arduino นั้น จะมีทั้งหมดจำนวน 14 pin โดย Arduino ได้กำหนดรหัสของตัวเลขซึ่งเป็นเลขจำนวนเต็มค่าระหว่าง 0 ถึง 13 สำหรับอ้างถึง digital I/O ทั้ง 14 pin แต่ตามปรกติแล้วจะมี digital I/O pin จำนวน 2 pin ซึ่งถูกสงวนไว้สำหรับใช้เป็นพอร์ตสื่อสารอนุกรม rs232 สำหรับใช้ Upload Code ของ โปรแกรม ให้กับบอร์ด จึงเหลือ digital I/O สำหรับใช้งาน Digital I/O จริงๆจำนวน 12 pin คือ Digital I/O หมายเลข 2 ถึง 13 เท่านั้นโดยคำสั่งในกลุ่มนี้จะมีอยู่ด้วยกัน 3 คำสั่ง คือ

- void pinMode(pin,mode)

- void digitalWrite(pin,value)

- int digitalRead(pin)

pinMod(pin,Mode)

ใช้ทำหน้าที่สำหรับกำหนดหน้าที่การทำงานของขา I/O ที่เป็น digital I/O pin ของไมโครคอนโทรลเลอร์ AVR ตามที่ Arduino กำหนดไว้ ว่าต้องการกำหนดใช้งานขา digital I/O ขาใดเพื่อใช้งานเป็น INPUT หรือ OUTPUT

```
pinMode (pin,Mode);
```

pin หมายถึง หมายเลข รหัส pin ของขาสัญญาณที่ทำหน้าที่เป็น digital I/O pin ซึ่งจะมีทั้งหมดจำนวน 14 pin คือ 0 ถึง 13 โดยต้องการกำหนดรูปแบบของตัวเลข ให้เป็นแบบจำนวนเต็ม (int) ด้วย

Mode หมายถึง หน้าที่การทำงานของ digital I/O pin ที่ต้องการกำหนด ซึ่งสามารถกำหนดได้ 2 หน้าที่ โดยใช้รหัสข้อความ เป็น INPUT และ OUTPUT

digitalWrite (pin, value)

ใช้ทำหน้าที่สำหรับกำหนดสถานะทาง OUTPUT ให้กับ Digital output pin ว่าต้องการให้มีสภาวะทางโลจิกเป็น High หรือ LOW ซึ่งขาสัญญาณที่จะสั่งงานด้วยคำสั่งนี้ จะต้องถูกกำหนดคุณสมบัติให้ทำหน้าที่เป็น OUTPUT เรียบร้อยแล้ว

```
digitalWrite(pin, value);
```

Pin หมายถึง รหัส pin ของขาสัญญาณที่ทำหน้าที่เป็น Digital I/O pin ซึ่งจะมีทั้งหมดจำนวน 14 pin คือ 0 ถึง 13 โดยต้องกำหนดรูปแบบของตัวเลข ให้เป็นแบบจำนวนเต็ม (int) ด้วย

Value หมายถึง ค่าสถานะทาง OUTPUT ของ digital output pin ที่ต้องการกำหนด ซึ่งสามารถกำหนดสถานะให้กับ pin ได้ 2 คำ คือ high และ low

digitalRead(pin)

ใช้ทำหน้าที่ อ่านสถานะ Logic input ของ digital input pin ว่ามีค่าสถานะเป็น high หรือ low ซึ่งขาสัญญาณที่จะสั่งอ่านด้วยคำสั่งนี้ จะต้องถูกกำหนดคุณสมบัติให้ทำหน้าที่เป็น input เรียบร้อยแล้ว

```
Var = digitalRead(pin)
```

Pin หมายถึง รหัส pin ของขาสัญญาณที่ทำหน้าที่เป็น digital input pin ซึ่งจะมีทั้งหมดจำนวน 14 pin คือ 0 ถึง 13 โดยต้องกำหนดรูปแบบของตัวเลข ให้เป็นจำนวนเต็ม (int) ด้วย

Var คือ ตัวแปรแบบ int สำหรับใช้รอรับคำสั่งที่ส่งคืนกลับมาจากฟังก์ชั่น ซึ่งเป็นค่าสถานะทางโลจิกของ digital input pin ซึ่งมีค่าเป็น HIGH หรือ LOW