หน่วยที่ 3
ชุดคำสั่งไมโครคอนโทรลเลอร์

Arduino

ชุดคำสั่งที่ใช้ในการเขียนโปรแกรม

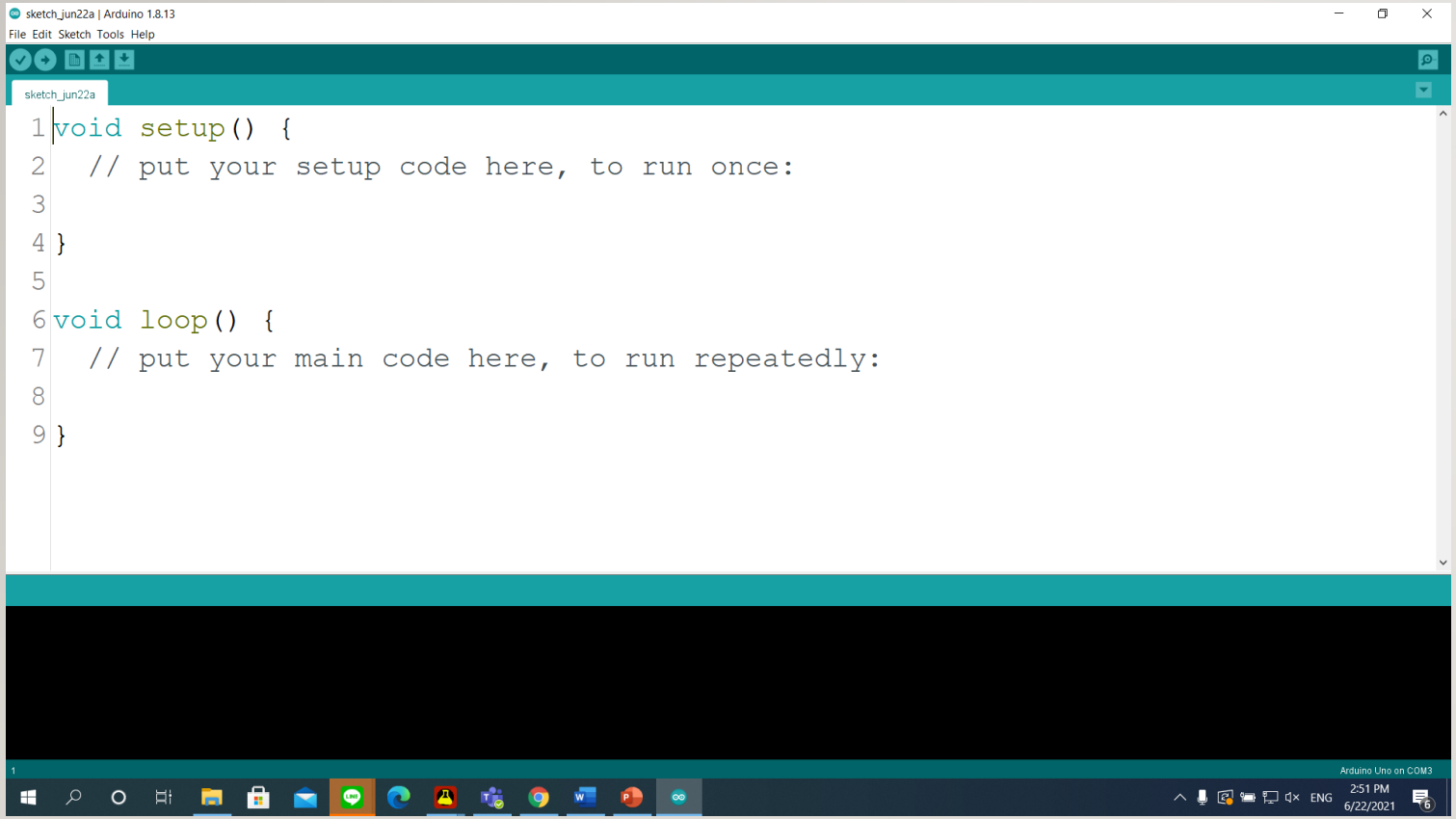
การเขียนโปรแกรมไมโครคอนโทรลเลอร์ Arduino สามารถเขียนได้ทั้งภาษาแอสเซมบลี และ ภาษาระดับสูง ได้แก่ ภาษาซี ขึ้นอยู่กับว่าผู้พัฒนาโปรแกรมเลือกใช้ภาษาใดมาใช้สำหรับเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ Arduino แต่เนื้อหาในหนังสือเล่มนี้เน้นการเขียนโปรแกรมด้วยภาษาซีเป็นหลัก

ภาษาแอสเซมบลีสำหรับ Arduino

ภาษาแอสเซมบลีจัดเป็นภาษาในระดับล่าง เป็นภาษาที่มีความใกล้เคียงกับภาษาเครื่องและฮาร์ดแวร์ของซีพียู โครงสร้างของภาษาแอสเซมบลีสำหรับไมโครคอนโทรลเลอร์ Arduinoไม่แตกต่าง จากภาษาแอสเซมบลีของสถาปัตยกรรมไมโครคอนโทรลเลอร์อื่นๆ ส่วนการเขียนโปรแกรมภาษาแอสเซมบลีต้องทำการใช้ชุดคำสั่งของซีพียูเพื่อเข้าถึงหน่วยความจำและข้อมูลในรีจิสเตอร์ซึ่งมีโดยตรง ส่งผลให้ภาษาแอสเซมบลีมีความยุ่งยากในการใช้งานมากกว่าภาษาระดับสูง

ภาษาแอสเซมบลีมีข้อดีกว่าภาษาระดับสูงตรงที่มีขนาดของโปรแกรมค่อนข้างเล็กมาก จึงทำให้ความเร็วในการทำงานที่สูงกว่าภาษาระดับสูง

ภาษาซีสำหรับ Arduino



โครงสร้างภาษาซีของไมโครคอนโทรลเลอร์ Arduino ใช้รูปแบบการเขียนโปรแกรมของภาษา C++ แต่ละโปรแกรมต้องมีฟังก์ชันหลักอย่างน้อย 2 ฟังก์ชัน ได้แก่ void setup() และ void loop()

ฟังก์ชัน void setup ()

ฟังก์ชันนี้จะเขียนที่ส่วนต้นของโปรแกรม ทำงานเมื่อโปรแกรมเริ่มต้นเพียงครั้งเดียว ใช้เพื่อกำหนดค่าของตัวแปรโหมดการทำงานของขาต่างๆ กำหนดการสื่อสารแบบอนุกรม การกำหนดค่าของการเรียกใช้ไลบรารี

```
1  void setup()
2  {
3        //เป็นส่วนของคำสั่ง สำหรับกำหนดการทำงานในโปรแกรม และทำเพียงครั้งเดียว
4  }
```

ฟังก์ชัน void loop ()

ฟังก์ชัน loop() ซึ่งมีการทำงานตรงตามชื่อ คือจะทำงานตามฟังก์ชัน**วนต่อเนื่องตลอดเวลา** ภายในฟังก์ชันจะมีโปรแกรมของผู้ใช้เพื่อรับค่าจากพอร์ต ประมวลผลแล้วสั่งเอาต์พุตออกขาต่างๆ เพื่อควบคุมการทำงานของบอร์ด

```
1  void loop()
2  {
3        // เป็นโปรแกรมหลักของคำสั่ง ซึ่งในส่วนนี้โปรแกรมมีการทำงานตลอดเวลา
4  }
```

ไวยากรณ์ภาษาซีของ Arduino

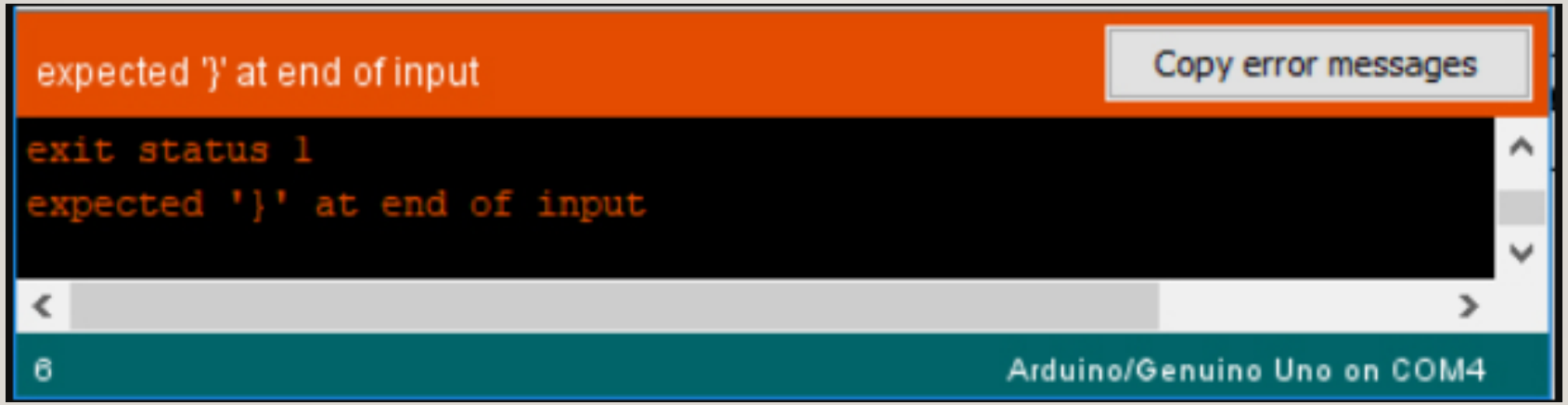
1. เครื่องหมาย ; (เซมิโคล่อน) เป็นการเขียนเพื่อจบคำสั่งในบรรทัดนั้นๆ

- บรรทัดคำสั่งที่ลืมเขียนปิดท้ายด้วยเซมิโคลอนจะทำให้แปลโปรแกรมไม่ผ่าน
- โดยตัวแปรภาษาอาจจะแจ้งให้ทราบว่าไม่พบเครื่องหมายเซมิโคลอน
- หรือแจ้งเป็นการผิดพลาดอื่นๆ บางกรณีที่ตรวจสอบบรรทัดที่แจ้งว่าเกิดการผิดพลาดแล้วไม่พบที่ผิดให้ตรวจสอบบรรทัดก่อนหน้านั้นใน

ไวยากรณ์ภาษาซีของ Arduino

2. วงเล็บปีกกา – curly brace { }

- เครื่องหมายวงเล็บปีกา เป็นส่วนสำคัญของภาษาซี
- โดยมีการใช้งานต่างตำแหน่ง สร้างความสับสนให้กับผู้ที่เริ่มต้น วงเล็บปีกกาเปิด { จะต้องเขียนตามด้วยวงเล็บปีกกาปิด } ด้วยเสมอ หรือที่เรียกว่าวงเล็บต้องครบคู่ซอฟต์แวร์



ไวยากรณ์ภาษาซีของ Arduino

3. เครื่องหมาย // (หมายเหตุบรรทัดเดียว)

- เป็นส่วนของผู้เขียนโปรแกรมอธิบายเพิ่มเติมในคำสั่งต่างๆ ว่าโปรแกรมทำงานอย่างไร
- เมื่อทำการคอมไพล์โปรแกรมประโยคที่อยู่ หลังเครื่องหมาย //ไม่ได้ถูกนำไปคอมไพล์ด้วย
- การใส่เครื่องหมาย // (หมายเหตุ) มีประโยชน์สำหรับ การตรวจสอบโปรแกรมภายหลัง หรือให้ผู้พัฒนาโปรแกรมท่านอื่นสามารถเข้าใจการเขียนโปรแกรมนั้น ๆ ได้

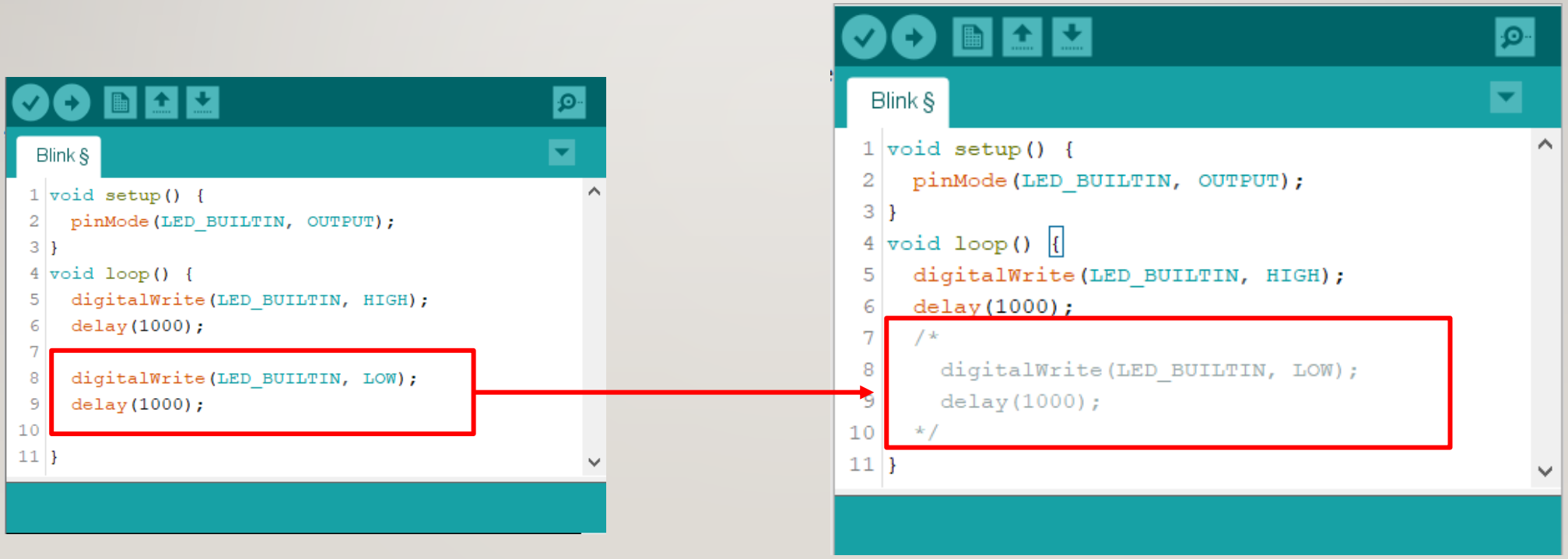
Blink §

```
1
2 void setup() {
3   // initialize digital pin LED_BUILTIN as an output.
4   pinMode(LED_BUILTIN, OUTPUT);
5 }
```

ไวยากรณ์ภาษาซีของ Arduino

4. เครื่องหมาย /* */ (หมายเหตุหลายบรรทัด)

- เป็นส่วนของผู้เขียนโปรแกรมอธิบายเพิ่มเติมในคำสั่งต่างๆ ว่าโปรแกรมทำงานอย่างไร
- สามารถอธิบายได้หลายบรรทัด



ไวยากรณ์ภาษาซีของ Arduino

5. #define

- เป็นคำสั่งที่ใช้งานมาก ในการกำหนดค่าคงที่ให้กับโปรแกรม
- การกำหนดค่าคงที่ไม่ได้เปลืองพื้นที่ หน่วยความจำของไมโครคอนโทรลเลอร์แต่อย่างไร

รูปแบบ #define constantName (ชื่อตัวแปรค่าคงที่) value (ค่าคงที่)

ตัวอย่างที่ #define ledpin 3 // เป็นการกำหนดให้ตัวแปร ledpin เท่ากับค่าคงที่ 3

เทคนิคสำหรับการเขียนโปรแกรม

ท้ายคำสั่ง #define ไม่ต้องมีเครื่องหมายเซมิโคลอน ถ้าใส่เกินแล้วเวลาคอมไพล์โปรแกรมจะแจ้งว่าเกิดการผิดพลาดในบรรทัดถัดไป

ไวยากรณ์ภาษาซีของ Arduino

6. #include

ใช้สั่งให้รวมไฟล์อื่นๆ เข้ากับไฟล์โปรแกรมหลักก่อน แล้วจึงทำการคอมไพล์โปรแกรม

รูปแบบคำสั่ง #include <file> #include “file”

ตัวอย่างที่ #include <stdio.h>

#include “lcd.h”

บรรทัด 1 จะสั่งให้เรียกไฟล์ stdio.h มารวมกับไฟล์โปรแกรมหลัก โดยค้นหาไฟล์จากตำแหน่งที่เก็บไฟล์ระบบของ Arduino โดยปกติเป็นไฟล์มาตรฐานที่มาพร้อมกับ Arduino

บรรทัด 2 สั่งให้รวมไฟล์ lcd.h มารวมกับไฟล์โปรแกรมหลัก โดยหาไฟล์จากตำแหน่งของไฟล์ภาษา C ปกติเป็นไฟล์ที่ผู้ใช้สร้างขึ้นเอง

ตัวดำเนินการทางคณิตศาสตร์

เครื่องหมาย	การกระทำ	ตัวอย่าง	คำอธิบาย
+	บวก	x = y + z	x เท่ากับ ค่าในตัวแปร y บวกกับค่าในตัวแปร z
-	ลบ	x = y - z	x เท่ากับ ค่าในตัวแปร y ลบกับค่าในตัวแปร z
*	คูณ	x = y * z	x เท่ากับ ค่าในตัวแปร y คูณกับค่าในตัวแปร z
/	หาร	x = y / z	x เท่ากับ ค่าในตัวแปร y หารกับค่าในตัวแปร z
%	หารเอาเศษ	x = y % z	x เท่ากับ เศษของการหารระหว่างตัวแปร y กับตัวแปร z

ตัวดำเนินการเปรียบเทียบ

เครื่องหมาย	การกระทำ	ตัวอย่าง	คำอธิบาย
>	มากกว่า	x>10	x มากกว่า10
<	น้อยกว่า	x<10	x น้อยกว่า10
>=	มากกว่าหรือเท่ากับ	x>=10	x มากกว่าหรือเท่ากับ 10
<=	น้อยกว่าหรือเท่ากับ	x<=10	x น้อยกว่าหรือเท่ากับ 10
==	เท่ากับ	x==10	x เท่ากับ 10
!=	ไม่เท่ากับ	x!=10	x ไม่เท่ากับ 10

ตัวดำเนินการทางตรรกะ

เครื่องหมาย	การกระทำ	คำอธิบาย
&&	และ	ผลลัพธ์ของการดำเนินการจะเป็นจริง เมื่อทั้งสองข้างของตัวดำเนินการเป็นจริงทั้งคู่
||	หรือ	ผลลัพธ์ของการดำเนินการจะเป็นเท็จ เมื่อทั้งสองข้างของตัวดำเนินการเป็นเท็จทั้งคู่
!	นิเสธ	ผลลัพธ์จะตรงกันข้ามกับค่าปัจจุบัน คือ กลับจากเท็จเป็นจริง กลับจากจริงเป็นเท็จ

ตัวดำเนินการทางตรรกะ

A	B	A && B (และ)	A || B (หรือ)	!A (นิเสธ)	!B (นิเสธ)
เท็จ (0)	เท็จ (0)	เท็จ (0)	เท็จ (0)	จริง (1)	จริง (1)
เท็จ (0)	จริง (1)	เท็จ (0)	จริง (1)	จริง (1)	เท็จ (0)
จริง (1)	เท็จ (0)	เท็จ (0)	จริง (1)	เท็จ (0)	จริง (1)
จริง (1)	จริง (1)	จริง (1)	จริง (1)	เท็จ (0)	เท็จ (0)

ตัวดำเนินการระดับบิต

เครื่องหมาย	การกระทำ
&	การแอนด์ของบิต (AND)
|	การออร์ของบิต (OR)
^	การเอ็กคลูซีฟออร์ของบิต (XOR)
~	กลับค่าของบิต (1’s Complement)
>>	เลื่อนบิตไปทางซ้าย
<<	เลื่อนบิตไปทางขวา

ตัวดำเนินการระดับบิต

A	B	A & B (แอนด์)	A | B (ออร์)	~A (กลับค่า)	~B (กลับค่า)	A ^ B (เอ็กคลูซีฟออร์)
เท็จ (0)	เท็จ (0)	เท็จ (0)	เท็จ (0)	จริง (1)	จริง (1)	เท็จ (0)
เท็จ (0)	จริง (1)	เท็จ (0)	จริง (1)	จริง (1)	เท็จ (0)	จริง (1)
จริง (1)	เท็จ (0)	เท็จ (0)	จริง (1)	เท็จ (0)	จริง (1)	จริง (1)
จริง (1)	จริง (1)	จริง (1)	จริง (1)	เท็จ (0)	เท็จ (0)	เท็จ (0)

ตัวดำเนินการ

เครื่องหมาย	การกระทำ	ตัวอย่าง	คำอธิบาย
++	เพิ่มค่า 1 ค่า	x++;	เพิ่มค่า x ขึ้น 1 ค่า
--	ลดค่า 1 ค่า	x--;	ลดค่า x ลง 1 ค่า
+=	บวก	x+=2;	x ใหม่เท่ากับ x เดิมบวกกับ 2
-=	ลบ	x-=2;	x ใหม่เท่ากับ x เดิมลบด้วย 2
*=	คูณ	x*=2;	x ใหม่เท่ากับ x เดิมคูณด้วย 2
/=	หาร	x/=2;	x ใหม่เท่ากับ x เดิมหารด้วย 2
%=	หารเอาเศษ	x%=2;	x ใหม่เท่ากับ x เดิมหารด้วย 2 แล้วเอาเศษ
&=	แอนด์	x&=2;	x ใหม่เท่ากับ x เดิมแอนด์ด้วย 2
|=	ออร์	x|=2;	x ใหม่เท่ากับ x เดิมออร์ด้วย 2

ตัวแปร

ตัวแปรเป็นชื่อเรียกแทนพื้นที่เก็บข้อมูลในหน่วยความจำของไมโครคอนโทรลเลอร์ โดยมีการตั้งชื่อเรียกหน่วยความจำในตำแหน่งนั้นเพื่อความสะดวกในการเรียกใช้ข้อมูล ถ้าจะใช้ข้อมูลใดก็ให้เรียกผ่านชื่อของตัวแปรที่เก็บเอาไว้

ค่าคงที่

- 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

ชนิดของข้อมูล

ชนิดข้อมูล	การเก็บข้อมูล	ขนาด
boolean	จริง (True : 1) หรือ เท็จ (False : 0)	1 บิต
char	อักขระหรือตัวเลขในรูปตัวอักขระ	1 ไบต์เก็บได้ตั้งแต่ -128 ถึง 127
unsigned char	อักขระหรือตัวเลขในรูปตัวอักขระ	1 ไบต์ เก็บได้ตั้งแต่ 0 ถึง 255
byte	ไบต์	1 ไบต์ เก็บได้ตั้งแต่ 0 ถึง 255
int	เก็บจำนวนเต็ม	2 ไบต์ เก็บจำนวนเต็ม -32768 ถึง 32767
unsigned int	เก็บจำนวนเต็ม ไม่คิดเครื่องหมาย	2 ไบต์ เก็บจำนวนเต็ม ไม่คิดเครื่องหมาย 0 ถึง 65535
long	เก็บจำนวนเต็มแบบยาว	4 ไบต์ เก็บจำนวนเต็มแบบยาว -2147483648 ถึง 2147483649
unsigned long	เก็บจำนวนเต็มแบบยาว ไม่คิดเครื่องหมาย	4 ไบต์ เก็บจำนวนเต็มแบบยาว ไม่คิดเครื่องหมาย 0 ถึง 429467296
float	เก็บทศนิยม	4 ไบต์ เก็บตัวเลขแบบทศนิยม 3.4*10^-38 ถึง 3.4*10^38
double (เฉพาะบอร์ด Arduino Due)	เก็บตัวเลขแบบทศนิยมแบบยาว	8 ไบต์ เก็บตัวเลขแบบทศนิยม 3.4*10^-308 ถึง 3.4*10^308
String	ตัวแปรสำหรับเก็บข้อความ	ไม่ระบุ
array	ตัวแปรหลายตัวที่ถูกเก็บรวมไว้ในตัวแปรชื่อเดียวกัน	ไม่ระบุ

กฎการตั้งชื่อตัวแปร

ในการตั้งชื่อตัวแปร หรือชื่อฟังก์ชันจะต้องตั้งตามกฎการตั้งชื่อของภาษาซี ดังนี้

1. ชื่อตัวแปรจะประกอบด้วยตัวอักขระหรือตัวเลขก็ได้ แต่ต้องขึ้นต้นด้วยตัวอักขระเสมอ
2. ชื่อตัวแปรต้องขึ้นต้นด้วยอักษรหรือเครื่องหมายขีดเส้น (underscore) "_" เท่านั้น
3. สามารถมีตัวเลขร่วมอยู่ในชื่อตัวแปรได้
4. ต้องไม่มีการเว้นวรรคในชื่อตัวแปร หากต้องการแยกตัวอักษรต่างๆ ออกจากกันสามารถใช้เครื่องหมายขีดเส้น "_" เพื่อแยกคำได้
5. ตัวอักษรตัวเล็กและตัวใหญ่ในภาษาซีถือว่าเป็นคนละตัวกัน
6. ไม่ใช้เครื่องหมายทางคณิตศาสตร์และสัญลักษณ์พิเศษในการตั้งชื่อตัวแปร
7. การตั้งชื่อตัวแปรจะยาวเท่าไดก็ใด แต่ชื่อของฟังก์ชันไม่ควรตั้งให้ยาวมาก โดยทั่วไปไม่ควรเกิน 32 ตัว เพราะจะทำให้ยุ่งยากในการเขียนโปรแกรม
8. ชื่อตัวแปรตั้งไม่ซ้ำกับคำสงวนและคำมาตรฐานของคอมไพเลอร์

คำสงวนในภาษาซี

คำสงวนในภาษาซี (Reserved Word) เป็นคำที่ทำหน้าที่เฉพาะอย่างไม่สามารถนำไปใช้เป็นตัวแปรหรือซื่อของฟังก์ชันได้ ในการเขียนโปรแกรมคอมไพเลอร์จะตีความคำหรืออักษรย่อเหล่านี้เป็นคำสั่งพิเศษ คำสงวนของภาษาซีตามมาตรฐาน ANSI มี 33 คำดังนี้

asm	auto	break	case	char
const	continue	default	do	double
else	enum	extern	float	for
goto	short	int	long	register
return	switch	signed	sizeof	static
struct	volatile	typedef	union	unsigned
void	if	while		

คำสั่งดิจิตอล I/O

1. pinMode() ใช้กำหนดพอร์ตเป็นอินพุตหรือเอาต์พุต

Syntax : pinMode(pin, mode)

2. digitalRead() ใช้อ่านค่าจากขาดิจิทัลที่ถูกกำหนดให้เป็นอินพุต

Syntax : digitalRead(pin)

3.digitalWrite() ใช้เขียนค่า (LOW/HIGH) ให้ขาดิจิทัลที่ถูกกำหนดให้เป็นเอาต์พุต

Syntax : digitalWrite(pin, value)

คำสั่งอนาล็อก I/O และคำสั่งเวลา

4. analogRead() ใช้อ่านแรงดันไฟฟ้าแบบอนาล็อกและแปลงเป็นจำนวนเต็มมีค่าระหว่าง 0 ถึง 1023

Syntax : analogRead(pin)

5. analogWrite() ใช้สัญญาณ PWM เป็นเอาต์พุต

Syntax : analogWrite(pin, value)

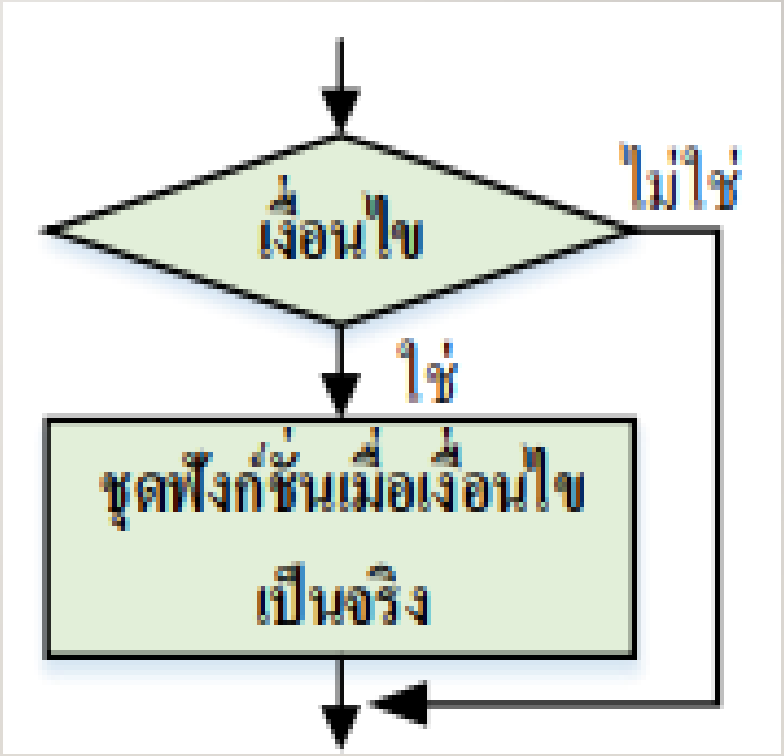
6. delay() เป็นการหน่วงเวลาตามค่าที่กำหนด มีหน่วยเป็นมิลลิวินาที

Syntax : delay(ms)

คำสั่งการควบคุม

เป็นคำสั่งให้ไมโครคอนโทรลเลอร์ทำงานตามเงื่อนไขหรือรูปแบบที่ผู้พัฒนาโปรแกรมต้องการ มีคำสั่งต่าง ๆ ดังต่อไปนี้

คำสั่ง if (ทางเลือกเดียว) เป็นคำสั่งในการตรวจสอบเงื่อนไขการทำงานของโปรแกรมถ้าเงื่อนไขเป็นจริง ให้ทำงานตามคำสั่งที่กำหนดนั้น มีรูปแบบคำสั่งดังนี้



คำสั่ง if (ทางเลือกเดียว)

โค้ดโปรแกรม

```
if (เงื่อนไขที่ตรวจสอบ)
{
    // คำสั่งที่ให้ทำงาน
       เมื่อเงื่อนไขเป็นจริง
}
```

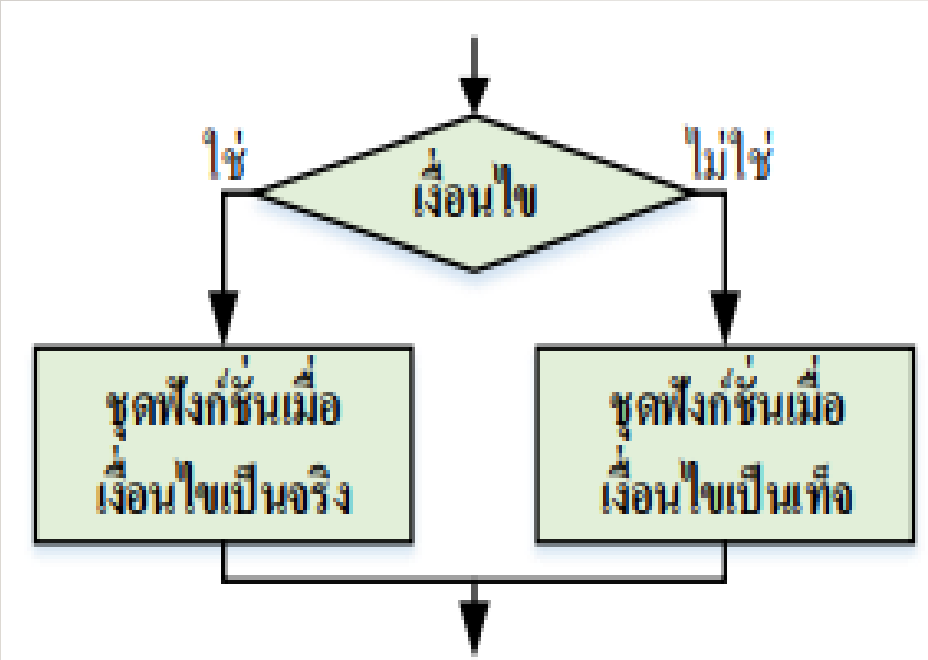
Example:

```
if (value>50)
{
 digitalWrite(13,HIGH);
}
```

- ตัวโปรแกรมจะทดสอบว่าถ้าตัวแปร value มีค่ามากกว่า 50 หรือไม่
- ถ้าใช่ให้ทำอะไรถ้าไม่ใช่ให้ข้ามการทำงานส่วนนี้
- การทำงานของคำสั่งนี้จะทดสอบเงื่อนไขที่เขียนในเครื่องหมายวงเล็บถ้าเงื่อนไขเป็นจริงทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นเท็จข้ามการทำงานส่วนนี้ไป
- ส่วนของการทดสอบเงื่อนไขที่เขียนอยู่ภายในวงเล็บจะต้องใช้ตัวกระทำเปรียบเทียบต่างๆ ดังนี้

คำสั่ง if...else (ทางเลือกเดียว)

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมได้มากกว่าคำสั่ง if ธรรมดา โดยสามารถกำหนดได้ว่าถ้าเงื่อนไขเป็นจริงให้ทำอะไร ถ้าเป็นเท็จให้ทำอะไร เช่น ถ้าค่าอินพุตแอนะล็อกที่อ่านได้น้อยกว่า 500 ให้ทำอะไร ถ้าค่ามากกว่าหรือเท่ากับ 500 ให้ทำอีกอย่าง จะเขียนคำสั่งได้ดังนี้

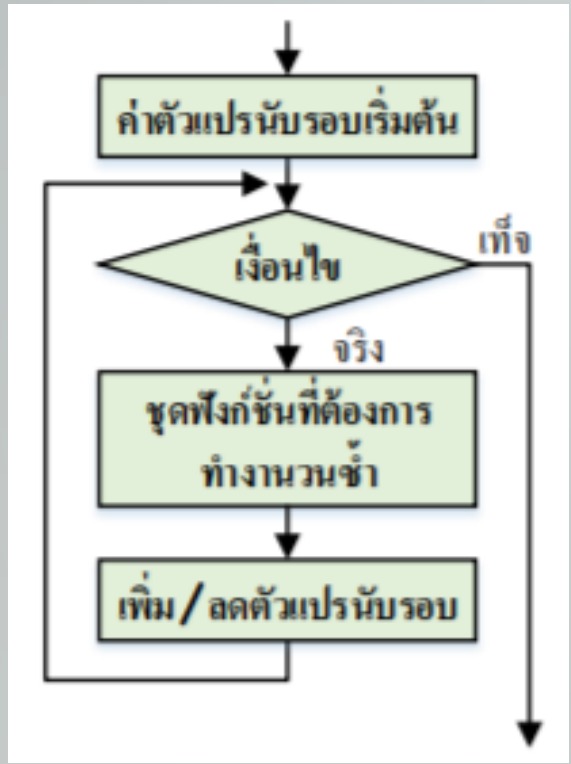



```
if (เงื่อนไขที่ตรวจสอบ)
{
    // put main code here
}
else
{
    //put main code here
}
```

คำสั่ง for (หลายทางเลือก)

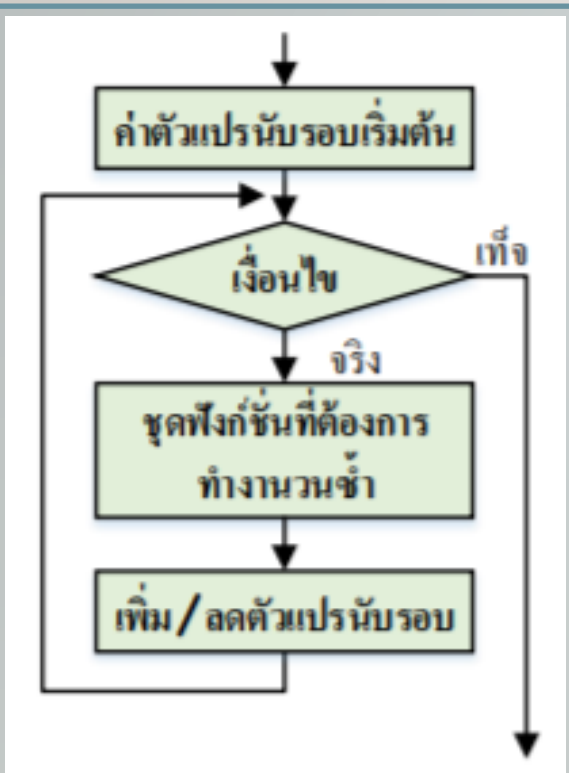
คำสั่ง for ใช้ในกรณีที่ทราบจำนวนรอบที่จะทำงานซ้ำ โดยมีรูปแบบดังนี้

ผังงาน	โค้ดโปรแกรม

	for (ค่าเริ่มต้น, เงื่อนไขการทำซ้ำ; การเพิ่มหรือลดค่าตัวแปร)
	{
	//คำสั่งที่ให้ทำงาน
	}

คำสั่ง for (หลายทางเลือก)

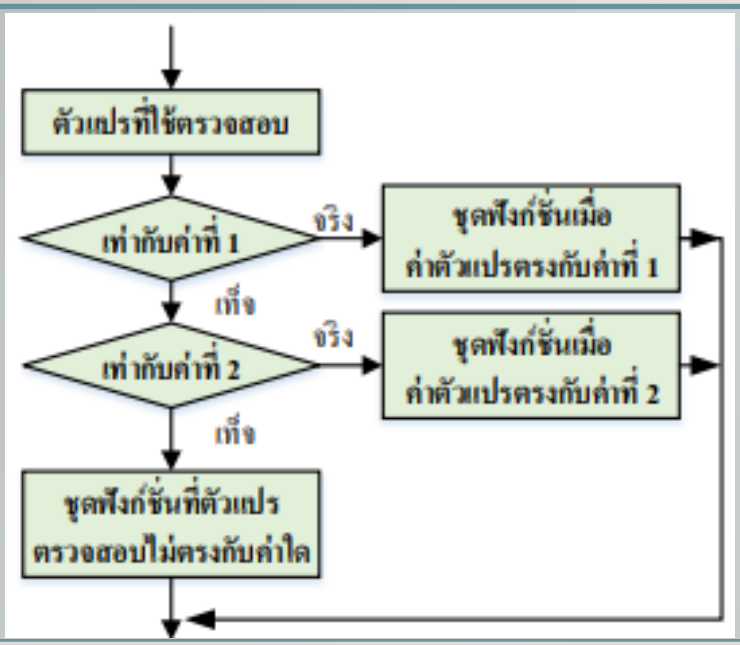
ผังงาน	โค้ดโปรแกรม

	for(int i=0;i<10;i++)
	{
	digitalWrite(13,HIGH);
	delay(500);
	digitalWrite(13,LOW);
	delay(500);
	}
	ประกาศตัวแปร i เป็นตัวแปรชนิด integer โดยให้มีค่าเริ่มต้นเท่ากับศูนย์ทำวนซ้ำ ไปเรื่อยๆ หากค่าตัวแปรยังน้อยกว่า 10 โดยในรอบถัดไป ให้เพิ่มค่าในตัวแปรนับรอบขึ้น 1 ค่า

คำสั่ง switch...case (หลายทางเลือก)

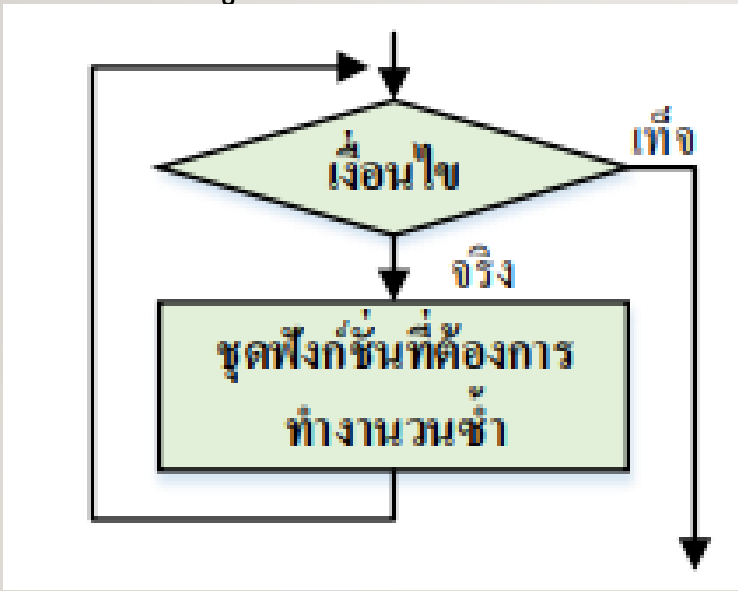
- ใช้ทดสอบเงื่อนไขเพื่อกำหนดการทำงานของโปรแกรม ถ้าตัวแปรที่ทดสอบตรงกับเงื่อนไขใดก็ให้ทำงานตามที่กำหนดไว้พารามิเตอร์

ผังงาน	โค้ดโปรแกรม

	switch (var)
	{
	case 1: คำสั่งที่ให้ทำงาน เมื่อตรวจสอบว่า ตัวแปร == 1
	break;
	case 2: คำสั่งที่ให้ทำงาน เมื่อตรวจสอบว่า ตัวแปร == 2
	break;
	default: คำสั่งที่ให้ทำงานเมื่อตรวจสอบว่า ตัวแปรไม่ตรงกับเงื่อนไขใด ๆ
	}

คำสั่ง while (หลายทางเลือก)

- คำสั่ง while เป็นคำสั่งที่ให้ทำงานวนซ้ำ หรือวนรอบโดยมีการตรวจสอบเงื่อนไขก่อนถ้าหากเงื่อนไขเป็นจริงจะทำงานตามคำสั่งที่เตรียมไว้เมื่อทำงานในชุดคำสั่งที่เตรียมไว้เสร็จจะมีการวนกลับไปตรวจสอบเงื่อนไขอีกหากเงื่อนไขเป็นจริงจะทำงานในชุดที่เตรียมไว้โดยทำแบบนี้ไปเรื่อยๆ จนกว่าเงื่อนไขจะเป็นเท็จจะออกจากวงรอบการทำงาน โดยมีรูปแบบดังนี้




```
while(เงื่อนไข)
  {
      //ชุดฟังก์ชันที่ต้องการทำซ้ำ

  }
```

คำสั่ง while (หลายทางเลือก)

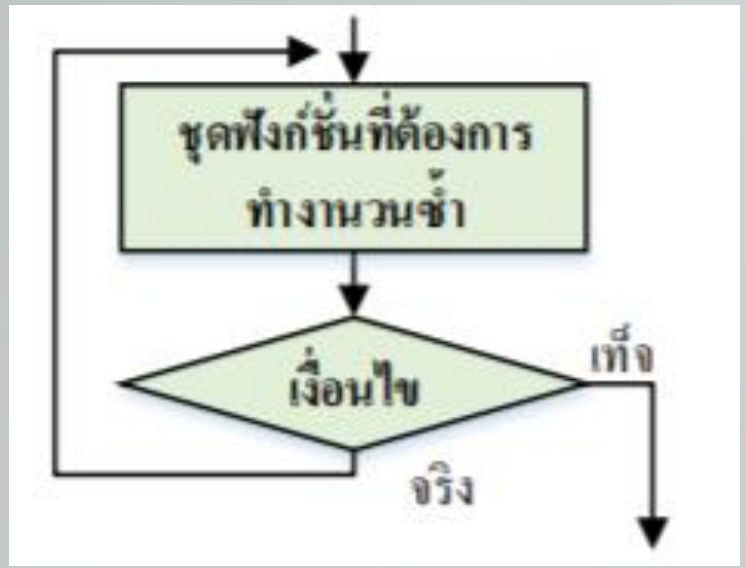
ผังงาน	โค้ดโปรแกรม

<u>ตัวอย่าง</u>
ตรวจสอบก่อนว่าเงื่อนไขเป็นจริงอยู่หรือไม่ (i ยังน้อยกว่า 10) หากเงื่อนไขเป็นจริงให้ทำงานที่เตรียมไว้ เมื่อทำงานครบให้กลับมาตรวจสอบเงื่อนไขใหม่วนซ้ำไปเรื่อยๆ จนกว่าเงื่อนไขจะเป็นเท็จ
	i=0;
	while(i<10)
	{
	digitalWrite(13,HIGH);
	delay(500);
	digitalWrite(13,LOW);
	delay(500);
	i++;
	}

คำสั่ง do - while

- เป็นคำสั่งทำซ้ำแบบวนรอบ โดยมีการทำงานตรงกันข้ามกับคำสั่ง while คือจะทำ งานในชุดฟังก์ชั่นที่เตรียมไว้ทำซ้ำก่อน 1 รอบแล้วจึงตรวจสอบเงื่อนไข

ผังงาน	โค้ดโปรแกรม

	do
	{
	//ชุดฟังก์ชันที่ต้องการทำซ้ำ
	}
	while(เงื่อนไข)

คำสั่ง do - while

ผังงาน	โค้ดโปรแกรม

<u>ตัวอย่าง</u>
ทำงานในฟังก์ชั่นที่เตรียมไว้แล้วตรวจสอบเงื่อนไขว่าตัวแปร i ยังมีค่าน้อยกว่า 10 หรือไม่หากยังน้อยกว่าให้วนกลับไปทำใหม่ซ้าๆ จนกว่าเงื่อนไขจะเป็นเท็จ
	i=0;
	do
	{
	digitalWrite(13,HIGH);
	delay(500);
	digitalWrite(13,LOW);
	delay(500);
	i++;
	}
	while(i<10)

สรุป

การเขียนโปรแกรมไมโครคอนโทรลเลอร์ Arduino สามารถเขียนได้ทั้งภาษาแอสเซมบลี และภาษาซี ขึ้นอยู่กับว่าผู้เขียนโปรแกรมเลือกใช้ภาษาใด มาใช้สำหรับการพัฒนาโปรแกรม ซึ่งแต่ละภาษาเมื่อผู้เขียนโปรแกรมทำการเขียนโปรแกรมเสร็จสิ้นแล้ว ต้องทำการแปลจากภาษาที่เขียนขึ้นให้เป็นภาษาเครื่องซึ่งเป็นภาษาที่ไมโครคอนโทรลเลอร์สามารถทำงานได้ ซึ่งในบทนี้เน้นเนื้อหาในการเขียนโปรแกรมด้วยภาษาซี ดังนั้นผู้เขียนโปรแกรมต้องศึกษาโครงสร้างและการทำงานของคำสั่งต่างๆ ของภาษาซีสำหรับไมโครคอนโทรลเลอร์ Arduino

แบบฝึกหัดท้ายหน่วยที่ 3

1. จงอธิบายความหมายของคำสั่งต่อไปนี้

คำสั่ง if	คำสั่ง |
คำสั่ง if...else	คำสั่ง HIGH/LOW
คำสั่ง for	คำสั่ง INPUT/OUTPUT
คำสั่ง ; (semicolon)	คำสั่ง &
คำสั่ง { } (curly braces)	คำสั่ง // (single line comment)

จบการเรียนการสอน