ภาษา C++ เป็นภาษาคอมพิวเตอร์เพื่อวัตถุประสงค์ทั่วไป ซึ่งสามารถเขียนโปรแกรมได้ทั้งแบบออบเจ็ค และการเขียนแบบปกติทั่วไป และยังมีเครื่องมืออำนวยความสะดวกในการจัดการและเข้าถึงระดับหน่วยความจำ นอกจากนี้มันยังถูกนำไปใช้ในการเขียนโปรแกรมแบบต่างๆ มากมาย เช่น โปรแกรมคอมพิวเตอร์ ระบบฝังตัว (Embedded) เว็บเซิร์ฟเวอร์ การพัฒนาเกม และแอพพลิเคชันที่ต้องการประสิทธิภาพอย่างสูง

ภาษา C++ เป็นภาษาที่ถูกออกแบบมาในการเขียนโปรแกรมระบบ ซึ่งมีประสิทธิภาพและความยืดหยุ่นในการออกแบบโปรแกรมสูง C++ เป็นภาษาที่ต้องคอมไพล์ก่อนที่จะนำไปใช้งาน ซึ่งสามารถพัฒนาได้ในหลายๆ แพลตฟอร์ม ซึ่งได้รับการสนับสนุนโดยองค์กรต่างๆ ที่ประกอบไปด้วย Free Software Foundation (FSF's GCC) LLVM Microsoft Intel และ IBM C++ นั้นถูกกำหนดให้เป็นภาษาที่เป็นมาตรฐานโดย International Organization for Standardization (ISO) ซึ่งเวอร์ชันล่าสุดนั้นเผยแพร่ในธันวาคม 2014 คือ ISO/IEC 14882:2014 หรือที่รู้จักกันในชื่อของ C++14 โดยที่ภาษา C++ ได้เริ่มกำหนดมาตราฐานครั้งแรกในปี 1998 คือ ISO/IEC 14882:1998 ภาษา C++ ถูกพัฒนาโดย Bjarne Stroustrup ที่ Bell Labs ตั้งแต่ปี 1979 ซึ่งในตอนแรกเป็นส่วนขยายของภาษา C โดยที่เขาต้องการที่จะพัฒนาภาษาที่มีประสิทธิภาพและยืดหยุ่นเหมือนกับภาษา C และยังมีคุณสมบัติใหม่ที่สูงกว่าสำหรับพัฒนาโปรแกรม Bjarne Stroustrup นักวิทยาศาสตร์คอมพิวเตอร์ชาวเดนมาร์ก ได้สร้างภาษา C++ ขึ้นในปี 1979 โดยเขาเริ่มจาก "C with Classes" ซึ่งเป็นภาษาก่อนหน้าของภาษา C++ แรงจูงใจสำหรับการสร้างภาษาใหม่นั้นมีต้นกำเนิดมาจากประสบการณ์ในการเขียนโปรแกรมสำหรับงานวิจัยในการศึกษาระดับปริญญาเอกของเขา ในขณะที่ Stroustrup เริ่มต้นการทำงานที่ AT&T Bell Labs เขามีปัญหาในการวิเคราะห์ UNIX kernel ซึ่งเกี่ยวกับ distributed computing จากการจดจำในประสบการณ์ปริญญาเอกของเขา Stroustrup ตั้งใจว่าจะเพิ่มความสามารถให้ภาษา C กับคุณสมบัติที่เหมือนภาษา Simula เขาเลือกภาษา C เพราะว่ามันเป็นภาษาเขียนโปรแกรมเพื่อวัตถุประสงค์ทั่วไป ที่ทำงานเร็ว สะดวกใช้งานง่ายและใช้กันอย่างแพร่หลาย จนกระทั่งในปี 2011 มาตฐานของ C++11 ได้ถูกเผยแพร่ โดยการเพิ่มคุณสมบัติใหม่เข้ามามากมาย รวมทั้งการเพิ่มเติมขนาดของไลบรารี่มาตรฐาน และให้ความสะดวกแก่โปรแกรมเมอร์ภาษา C++ เป็นอย่าง

มาก หลังจากบทเรียนนี้เสร็จสิ้น คุณจะเข้าใจและสามารถสร้างแอพพลิเคชันของคุณเอง เพราะว่าภาษา C++ เป็นพื้นฐานที่นำไปสู่การกำเนิดภาษาอื่นๆ และเป็นภาษาที่พัฒนามาจากภาษา C การเริ่มต้นเรียนรู้กับภาษา C++ ยังช่วยให้คุณเข้าใจและเรียนภาษาอื่นได้ง่ายขึ้น เช่น ภาษา C# ภาษา Java หรือ ภาษา PHP เป็นต้น

คอมไพเลอร์

คอมไพเลอร์คือโปรแกรมคอมพิวเตอร์หรือกลุ่มของโปรแกรมที่แปลงซอสโค้ดที่เขียนขึ้นในภาษา C++ ไปเป็นภาษาคอมพิวเตอร์ (Target language) หลังจากที่ทำการแปลงแล้วจะได้ข้อมูลในรูปแบบของฐานสอง (Binary) ที่เรียกกันว่า Object code เหตุผลที่ต้องแปลงโปรแกรมจากภาษาเขียนโปรแกรมไปเป็นภาษาเครื่องโดยคอมไพเลอร์ก็เพื่อสร้างโปรแกรมที่สามารทำงานได้ (Executable program)

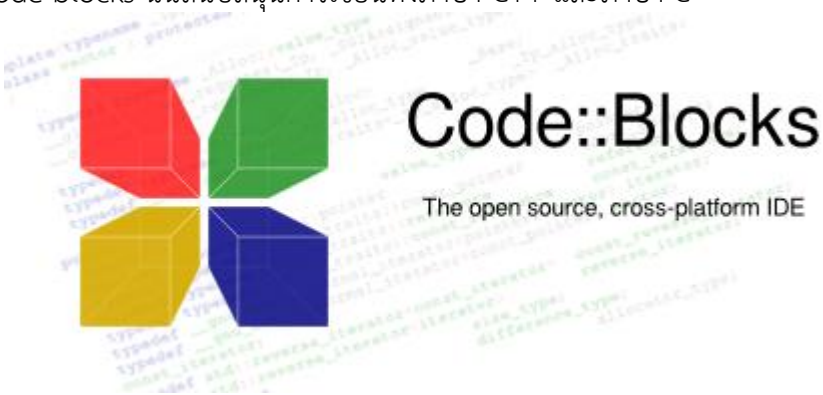
คอมพิวเตอร์สามารถเข้าใจแค่ภาษาเครื่อง ภาษาที่ประกอบไปด้วยตัวเลข 1 และ 0 เราจำเป็นต้องใช้คอมไพเลอร์เพื่อแปลงโปรแกรมที่เราเขียนไปเป็นภาษาเครื่องที่ให้คอมพิวเตอร์สามารถทำงานได้

คอมไพเลอร์ช่วยให้โปรแกรมเมอร์พัฒนาโปรแกรมของพวกเขาได้อย่างง่ายดายในการเขียนโปรแกรมด้วยภาษาระดับสูง อย่างเช่น ภาษา C++

เครื่องมือพัฒนาโปรแกรม

ทางที่ง่ายที่สุดที่คุณจะสามารถเขียนโปรแกรมในภาษา C++ ได้นั้นคือการใช้ IDE ซึ่ง IDE เป็นการรวบรวมชุดโปรแกรมที่จำเป็นสำหรับในการพัฒนาโปรแกรม มันเป็นโปรแกรมที่อำนวยความสะดวกและให้เครื่องมือที่จำเป็นสำหรับในการพัฒนาโปรแกรม โดยปกติแล้ว IDE จะประกอบไปด้วยโปรแกรมสำหรับเขียนและแก้ไขโค้ดที่มาพร้อมกับเครื่องมือโดยอัตโนมัติ คอมไพเลอร์ และตัวดีบักโปรแกรม

สำหรับในบทเรียนนี้ โปรแกรมที่เป็นที่นิยมที่สุดที่เราจะแนะนำคือ Code blocks มันสามารถใช้ได้บนแพลตฟอร์มต่างๆ เช่น Windows Linux และ MacOS ซึ่งมากับคอมไพเลอร์ GCC (MingW / GNU GCC) MSVC++ clang Digital Mars Borland C++ 5.5 Open Watcom และอื่นๆ Code blocks นั้นสนับสนุนการเขียนทั้งภาษา C++ และภาษา C



<u>Arduino ภาษา C/C++</u>

ภาษาของการเขียนโปรแกรมใช้งานArduino Board

ภาษา C/C++ โดยประกอบด้วย Structure,values (variables and constants) และ Functions

ฟังก์ชั่นหลัก(Structure)

เป็นฟังก์ชั่นหลักในการเขียนโปรแกรม จำเป็นต้องมีในทุกโปรแกรม

Setup()

คือ ฟังก์ชั่นใช้ในการประกาศค่าเริ่มต้น ตำแหน่งพอร์ตที่ใช้งาน รวมถึงฟังก์ชั่นที่อยู่ไลบารี่ที่ใช้งาน เป็นฟังก์ชั่นที่ทำงานเพียงครั้งเดียว จะทำงานทุกครั้ง ที่มีการรีเซต หรือรีบูตเครื่องใหม่ เท่านั้น

เช่น

```
intbuttonPin = 3;// การตั้งค่าตัวแปร buttonPin เท่ากับ 3
void setup()
{
    Serial.begin(9600); //ประกาศการใช้งานการสื่อสารรับส่งข้อมูลผ่าน พอร์ตRS232
     pinMode(buttonPin, INPUT);// การตั้งค่าโหมด ของตัวแปรแบบคงที่ buttonPin เป็นโหมด อินพุต
}
void loop()
{
  // ...
}
```

Loop ()

คือ ฟังก์ชั่นใช้ในการเขียนโค้ดโปรแกรมการทำงานของArduinoเป็นฟังก์ชั่นการวนลูปไปเรื่อยๆ

เช่น

```
constintbuttonPin = 3; // การตั้งค่าตัวแปร buttonPin เท่ากับ 3
void setup()
{
   Serial.begin(9600);//ประกาศการใช้งานการสื่อสารรับส่งข้อมูลผ่าน พอร์ตRS232
   pinMode(buttonPin, INPUT);// การตั้งค่าโหมด ของตัวแปรแบบคงที่ buttonPin เป็นโหมด อินพุต
}

void loop ()
{
   if (digitalRead(buttonPin) == HIGH)// ตรวจสอบค่าอินพุตที่รับมา เป็น HIGH ใช่หรือไม่
       Serial.write('H');// ใช่ ส่งค่าอักษร H ผ่าน พอร์ตRS232
   else
       Serial.write('L');// ไม่ ส่งค่าอักษร L ผ่าน พอร์ตRS232
   delay(1000); //หน่วงเวลา 1วินาที
}
```

ชุดคำสั่งในการควบคุม (Control Structures)

เป็นชุดคำสั่งในการใช้ในการตัดสินใจหาทางออก เพื่อใช้ในการทำงาน

If

คือ คำสั่งในการตัดสินใจ แบบตัวเลือกเดียว โดยใช้งานร่วมกับ And, Or Not, ==, !=, <, > เพื่อใช้ในการตัดสินใจในการหาคำตอบ เช่น

```
if (ตัวแปร> 50)
{
  // .....
}

if (ตัวแปร> 120)digitalWrite(LEDpin, HIGH);

if (ตัวแปร> 120)
 digitalWrite(LEDpin, HIGH);

if (ตัวแปร> 120) { digitalWrite(LEDpin, HIGH); }

if (ตัวแปร> 120) {
 digitalWrite(LEDpin1, HIGH);
 digitalWrite(LEDpin2, HIGH);
}
```

If...else

คือ คำสั่งในการตัดสินใจ แบบหลายตัวเลือก โดยใช้งานร่วมกับ And, Or Not, ==, !=, <, > เพื่อใช้ในการตัดสินใจในการหาคำตอบ เช่น

```
If (pinFiveInput< 500)
{
  // action A
}
Else
{
  // action B
}
```

For

คำสั่ง FOR เป็นคำสั่งกำหนดเงื่อนไขเป็นจำนวนครั้งที่จะทำตามชุดคำสั่งต่าง ๆ ภายใน loop เหมาะที่จะใช้กับงานประเภทที่ไม่มีการเปลี่ยนแปลง เช่น

```
void setup(){
  // no setup needed
}
void loop(){
        for (intตัวแปร=0; ตัวแปร<= 255; ตัวแปร++){
       analogWrite(PWMpin, i);
       delay(10);
        }
 }
```

Switch case

คำสั่ง switch case ใช้ในการจัดการเงื่อนไขหลายเงื่อนไขโดยเฉพาะการใช้งานโครงสร้าง การจำแนกเงื่อนไขไม่จำเป็นต้องอาศัยเฉพาะตัวแปรที่เก็บค่าจำนวนเต็มเท่านั้น ข้อมูลแบบอื่นก็ใช้ได้

เช่น

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

While

คำสั่ง While คือเงื่อนไขที่จะทำการตรวจสอบว่าเป็นจริงหรือเท็จ ชุดคำสั่งก็คือ ส่วนที่ทำงานซ้ำๆ โดยจะต้องมีคำสั่งที่จะทำให้ เงื่อน เป็นเท็จด้วย

เช่น

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

Do... while

คำสั่ง do while เป็นคำสั่งที่กำหนดให้มีการทำงานวนรอบ คล้าย ๆ คำสั่ง While แต่แตกต่างกันที่คำสั่ง do while จะให้ทำคำสั่งใน loop do ก่อน แล้วค่อยพิจารณาเงื่อนไขใน while ถ้าค่าเงื่อนไขใน while เป็นจริง จึงจะวนรอบทำคำสั่งในลูป do ต่อไป เช่น

```
do {
delay(50);          // wait for sensors to stabilize
ตัวแปร =  readSensors();  // check the sensors
} while (ตัวแปร< 100);
```

Break

คำสั่ง break เป็นคำสั่งที่ให้โปรแกรมออกจาก loop ทันที โดยไม่ทำคำสั่งที่เหลือต่อ ส่วนมากก่อนจะใช้คำสั่งนี้ ก็จะมีการตรวจสอบอะไรซักอย่างเสียก่อน ซึ่งคำสั่ง break นี้ สามารถใช้ได้กับ loop หลาย ๆ loop ไม่ว่าจะเป็น while, do while, for, switch และอื่น ๆ

เช่น

```
for (  ตัวแปร = 0;  ตัวแปร< 255;  ตัวแปร ++)
{
  digitalWrite(PWMpin,  ตัวแปร);
  sens = analogRead(sensorPin);
  if (sens> threshold){      // bail out on sensor detect
       ตัวแปร = 0;
       break;
   }
   delay(50);
}
```

Continue

คำสั่ง continue ใช้สำหรับสั่งให้กลับไปเริ่มต้นที่จุดเริ่มต้นใหม่ ใช้ร่วมกับคำสั่งการวนลูปต่างๆ จะต่างกับคำสั่งเพราะว่า คำสั่ง break นั้นจะเป็นคำสั่งเพื่อออกจาก loop ส่วนคำสั่ง continue นั้นจะเป็นคำสั่งเพื่อกระโดดไปยังต้น loop เช่น

```
for (x = 0; ตัวแปร< 255;ตัวแปร ++)
{
    if (ตัวแปร> 40 &&ตัวแปร< 120){      // create jump in values
     continue;
    }
    digitalWrite(PWMpin, ตัวแปร);
    delay(50);
}
```

Return

คำสั่ง return คือคำสั่งที่ส่งค่าอะไรก็ได้กลับออกไปจากฟังก์ชั่น

เช่น

```
intcheckSensor(){
  if (analogRead(0) > 400) {
    return 1;
  } else{
 return 0;
   }
}
```

Goto

คำสั่ง gotoเป็นคำสั่งที่ทำให้ กระโดดไปทำบรรทัดนั้น

เช่น

```
for(byte ตัวแปรตัวที่1 = 0; ตัวแปรตัวที่1 < 255; ตัวแปรตัวที่1 ++){
      for(byte ตัวแปรตัวที่2 = 255; ตัวแปรตัวที่2> -1; ตัวแปรตัวที่2--){
          for(byte ตัวแปรตัวที่3 = 0; ตัวแปรตัวที่3< 255;ตัวแปรตัวที่3 ++){
              if (analogRead(0) > 250){ goto bailout;}
              // more statements ...
        }
    }
}
bailout:
```

Further Syntax

; (semicolon) จะใช้ ; (semicolon) เมื่อจบคำสั่ง

เช่น intตัวแปร = 13;

{} (curly braces)

วงเว็บปีกกาหรือ {} เป็นส่วนที่สำคัญของการเขียนโปรแกรมภาษา C มีการใช้ในโครงสร้างที่แตกต่างกันหลายประการที่ระบุ และบางครั้งอาจจะทำให้เกิดความสับสน

เช่น

Functions

```
voidmyfunction(datatype argument){
statements(s)
}
```

Loops

```
while (boolean expression)  {
```

```
 statement(s)
 }
do  {
statement(s)
} while (boolean expression);

for (initialisation; termination condition; incrementing expr)  {
 statement(s)
 }
```

Conditional statements

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

// (single line comment)

คำสั่งสำหรับอธิบายหรือ comment ในภาษาซี คือส่วนที่หมายถึงเหตุของโปรแกรมมีไว้เพื่อให้ผู้เขียนโปรแกรมอธิบายกำกับลงไปใน source code ซึ่งคอมไพเลอร์จะข้ามการแปลผลในส่วนที่เป็นคอมเม้น ส่วนของ // (single line comment) จะเป็นการคอมเม้นบรรทัดเดียว เช่น

```
ตัวแปร  = 5;  // This is a single line comment. Anything after the slashes is a
comment
  // to the end of the line
```

/* */ (multi-line comment)

คำสั่งสำหรับอธิบายหรือ comment ในภาษาซี คือส่วนที่หมายถึงเหตุของโปรแกรมมีไว้เพื่อให้ผู้เขียนโปรแกรมอธิบายกำกับลงไปใน source code ซึ่งคอมไพเลอร์จะข้ามการแปลผลในส่วนที่เป็นคอมเม้น ส่วนของ /* */ (multi-line comment) จะเป็นการคอมเม้นหลายบรรทัด เช่น

```
 /* 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!
 */
```

#define

Define เป็นการกำหนดค่านิพจน์ต่างๆ ให้กับชื่อของตัวคงที่

เช่น #define ledPin 3

// the compiler will replace any mention of ledPin with the value 3 at compile time.

#include

การกำหนดชื่อไฟล์ตามหลัง Include จะใช้เครื่องหมาย <>ซึ่งจะเป็นการอ่านไฟล์จากไดเร็กทอรีหรือโฟลเดอร์ที่กำหนดไว้ก่อนแล้ว โดยปกติจะเป็นโฟลเดอร์ include แต่ถ้าใช้เครื่องหมาย “ ” เป็นการอ่านไฟล์จาก โฟลเดอร์ หรือ ไดเร็กทอรี ที่กำลังติดต่ออยู่และไฟล์ที่จะ include เข้ามานี้จะต้องไม่มีฟังก์ชัน main() โดยมากจะประกอบไปด้วยโปรแกรมย่อย ค่าคงที่ หรือข้อความต่างๆ

เช่น

```
#include <avr/pgmspace.h>

prog_uint16_tmyConstants[] PROGMEM = {0, 21140, 702  , 9128,  0, 25764, 8456,
0,0,0,0,0,0,0,0,29810,8968,29762,29762,4500};
```

Arithmetic Operators

= (assignment operator)

เครื่องหมาย = หมายถึง นำค่าจาก operand ฝั่งขวาไปใส่ใน operand ฝั่งซ้าย

เช่น

```
intsensVal;                 // declare an integer variable named sensVal
senVal = analogRead(0);       // store the (digitized) input voltage at analog pin 0
in SensVal
```

+ (addition)

เครื่องหมาย + หมายถึง การบวก

เช่น

```
y = y + 3;
```

- (subtraction)

เครื่องหมาย - หมายถึง การลบ

เช่น

```
x = x - 7;
```

*** (multiplication)**

เครื่องหมาย * หมายถึง การคูณ

เช่น

```
i = j * 6;
```

/ (division)

เครื่องหมาย / หมายถึง การคูณ

เช่น

```
r = r / 5;
```

% (modulo)

เครื่องหมาย % หมายถึงหาเศษจากการหาร

เช่น

```
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
```

Comparison Operators

== (equal to)

เครื่องหมาย == หมายถึงเท่ากับ

เช่น

```
x == y (x is equal to y)
```

!= (not equal to)

เครื่องหมาย != หมายถึงไม่เท่ากับ

เช่น

```
x != y (x is not equal to y)
```

< (less than)

เครื่องหมาย < หมายถึงน้อยกว่า

เช่น

```
x <  y (x is less than y)
```

> (greater than)

เครื่องหมาย > หมายถึงมากกว่า

เช่น

x > y (x is greater than y)

<= (less than or equal to)

เครื่องหมาย <= หมายถึงมากกว่าหรือเท่ากับ

เช่น

x <= y (x is less than or equal to y)

>= (greater than or equal to)

เครื่องหมาย >= หมายถึงน้อยกว่าหรือเท่ากับ

เช่น

x >= y (x is greater than or equal to y)

Boolean Operators

&& (and)

เครื่องหมาย&& (and) หมายถึง และ(and)

เช่น

```
if (digitalRead(2) == HIGH  &&digitalRead(3) == HIGH) { // read two switches
  // ...
}
```

|| (or)

เครื่องหมาย || (or) หมายถึง หรือ (or)

เช่น

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

! (not)

เครื่องหมาย ! (not) หมายถึง ไม่ (not)

เช่น

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

}

Pointer Access Operators

*** dereference operator**

* dereference operator หมายถึงตัวดำเนินการ คูณ

& reference operator

& dereference operator หมายถึงตัวดำเนินการและ

Bitwise Operators

& (bitwise and)

เครื่องหมาย& (bitwise and) หมายถึง การ and บิตต่อบิต

เช่น

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

| (bitwise or)

เครื่องหมาย | (bitwise or) หมายถึง การ or บิตต่อบิต

เช่น

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

^ (bitwise xor)

เครื่องหมาย ^ (bitwise xor) หมายถึง การ xorบิตต่อบิต

เช่น

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

~ (bitwise not)

เครื่องหมาย ~ (bitwise not) หมายถึง การกลับบิตทั้งหมดเป็นค่าตรงกันข้าม

เช่น

```
int a = 103;    // binary:  0000000001100111
 int b = ~a;    // binary:  1111111110011000 = -104
```

<< (bitshift left)

เครื่องหมาย<< (bitshift left)หมายถึง การเลื่อนบิตมาทางซ้าย

เช่น

```
 int a = 5;         // binary: 0000000000000101
int b = a << 3;   // binary: 0000000000101000, or 40 in decimal
int c = b >> 3;   // binary: 0000000000000101, or back to 5 like we started with
```

>> (bitshift right)

เครื่องหมาย<< (bitshift left) หมายถึง การเลื่อนบิตมาทางขวา

เช่น

```
 int a = 5;         // binary: 0000000000000101
int b = a << 3;   // binary: 0000000000101000, or 40 in decimal
int c = b >> 3;   // binary: 0000000000000101, or back to 5 like we started with
```

Compound Operators

++ (increment)

เครื่องหมาย++ (increment) หมายถึง การเพิ่มค่าหนึ่งค่าให้กับตัวแปร

เช่น

```
x = 2;
y = ++x;       // x now contains 3, y contains 3
```

-- (decrement)

เครื่องหมาย-- (decrement) หมายถึง การลดค่าหนึ่งค่าให้กับตัวแปร

เช่น

```
x = 2;
y = x--;       // x contains 2 again, y still contains 3
```

+= (compound addition)

เครื่องหมาย+= (compound addition) หมายถึง การบวกค่าให้กับตัวแปร

เช่น

```
x = 2;
x += 4;       // x now contains 6
```

-= (compound subtraction)

เครื่องหมาย-= (compound subtraction) หมายถึง การลบค่าให้กับตัวแปร

เช่น

```
x = 2;
x -= 3;       // x now contains 3
```

***= (compound multiplication)**

เครื่องหมาย*= (compound multiplication) หมายถึง การคูณค่าให้กับตัวแปร
เช่น

```
x = 2;
x *= 10;      // x now contains 30
```

/= (compound division)

เครื่องหมาย/= (compound division) หมายถึง การหารค่าให้กับตัวแปร
เช่น

```
x = 2;
x /= 2;       // x now contains 15
```

&= (compound bitwise and)

เครื่องหมาย&= (compound bitwise and) หมายถึง การ and ค่าให้กับตัวแปร
เช่น

```
myByte =  10101010;
myByte&= B1111100 == B10101000;
```

|= (compound bitwise or)
เครื่องหมาย |= (compound bitwise or) หมายถึง การ or ค่าให้กับตัวแปร
เช่น

```
myByte =  B10101010;
myByte |= B00000011 == B10101011;
```

Variables

Constants เป็นค่าคงที่ตัวแปรที่กำหนดไว้ล่วงหน้าคือตัวอักขระที่นำมาประกอบกัน ตั้งแต่ 1 ตัวอักขระขึ้นไป เพื่อบอกลักษณะอย่างใดอย่างหนึ่งของข้อมูล

INPUT | OUTPUT| INPUT_PULLUP

ค่าคงที่เหล่านี้แสดงถึงระดับลอจิกที่ขาไอซีว่าเป็น HIGH หรือ LOW และใช้เมื่อมีการอ่านหรือเขียนไปที่ขาไอซี HIGH จะแทนระดับลอจิก 1, ON, หรือ 5 volts ในขณะที่ LOW คือระดับลอจิก 0, OFF, หรือ 0 volts
เช่น

```
digitalWrite(13, HIGH)
```

true | false

ค่าเหล่านี้เป็นค่าคงที่ booleanซึ่งบอกสถานะระดับลอจิก FALSE หมายถึง 0(ศูนย์) ในขณะที่ TRUE จะหมายถึง 1, หรืออะไรก็ได้ที่ไม่ใช่ ศูนย์ ดังนั้นในทางลอจิกแล้ว -1, 2, -200 จะหมายถึง TRUE

เช่น

```
if (b == TRUE);
{
doSomething;
}
```

Integer constants

เป็นตัวแปรพื้นฐานที่เก็บตัวเลขโดยไม่มีจุดทศนิยม และเก็บค่า 16 bit มีค่าระหว่าง 32,767 ถึง -32,768

เช่น

```
intledPin = 13;
```

Floating point constants

เป็นตัวแปรพื้นฐานที่เก็บตัวเลขโดยไม่มีจุดทศนิยม และเก็บค่า 16 bit มีค่าระหว่าง 32,767 ถึง -32,768

เช่น

```
floatmyfloat;
floatsensorCalbrate = 1.117;
```

Data Types

Void

เป็นตัวแปรประเภทหนึ่ง ไม่มีขนาด

เช่น

```
void setup()
{
  // ...
}

void loop()
{
  // ...
}
```

Boolean

เป็นค่าคงที่ มีสองค่า คือ true และ false

เช่น

```
intLEDpin = 5;       // LED on pin 5
intswitchPin = 13;   // momentary switch on 13, other side connected to ground
boolean running = false;
void setup()
```

```
{
pinMode(LEDpin, OUTPUT);
pinMode(switchPin, INPUT);
digitalWrite(switchPin, HIGH);       // turn on pullup resistor
}
void loop()
{
if (digitalRead(switchPin) == LOW)
{  // switch is pressed - pullup keeps pin high normally
delay(100);                          // delay to debounce switch
running = !running;                  // toggle running variable
digitalWrite(LEDpin, running)       // indicate via LED
  }
}
```

Char

ข้อมูลชนิดอักขระ ใช้เนื้อที่ 1 byte

เช่น

```
charmyChar = 'A';
charmyChar = 65;
```

Unsigned char

ข้อมูลชนิดอักขระ ไม่คิดเครื่องหมาย

เช่น

```
unsigned char myChar = 240;
```

Byte

ตัวแปร byte เก็บตัวเลข 8 bit ไม่มีทศนิยม มีค่า 0 - 255

เช่น

```
byte b = B10010;  // "B" is the binary formatter (B10010 = 18 decimal)
```

Int

เป็นตัวแปรพื้นฐานที่เก็บตัวเลขโดยไม่มีจุดทศนิยม และเก็บค่า 16 bit มีค่าระหว่าง 32,767 ถึง -32,768

เช่น

```
IntledPin = 13;
```

UnsignedInt

ข้อมูลชนิดจำนวนเต็ม ไม่คิดเครื่องหมาย

เช่น

```
unsignedintledPin = 13;
```

Word

คำเก็บ 16 บิต เก็บค่าตั้งแต่ 0-65535 เช่นเดียวกับ int

เช่น

```
word w = 10000;
```

Long

เป็นตัวแปรจำนวนเต็มแบบขยายโดยไม่มีจุดทศนิยม เก็บค่าแบบ 32 bit มีค่าระหว่าง 2,147,483,647 ถึง -2,147,483,648

เช่น

```
ongspeedOfLight = 186000L;   // see Integer Constants for explanation of the 'L'
```

Unsigned long

ข้อมูลชนิดจำนวนเต็มแบบยาว ไม่คิดเครื่องหมาย

เช่น

```
unsigned long time;

void setup()
{
Serial.begin(9600);
}

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

Short

ข้อมูลชนิดจำนวนเต็มแบบสั้น ใช้เนื้อที่ 1 byte

เช่น

shortledPin = 13;

Float

ตัวแปรชนิด floating-point หรือตัวแปรที่มีจุดทศนิยม ตัวแปรนี้มีค่ามากกว่าค่าของตัวแปรจำนวนเต็ม โดยใช้เนื้อที่เก็บ 32 bit

เช่น

floatmyfloat;

floatsensorCalbrate = 1.117;

Double

ข้อมูลชนิดเลขทศนิยม ใช้เนื้อที่ 8 byte

String - char array

จะมีการเก็บข้อมูลอยู่ 2 ส่วน ส่วนแรกจะเป็นข้อมูลตัวอักษรโดยเก็บเรียงกันไป และส่วนที่ 2 จะเก็บจุดสิ้นสุดของสตริง

เช่น

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";

String – object

การเก็บข้อมูลของลองสตริงบนหน่วยความจำ

Array

ตัวแปร arrays หรือตัวแปรหลายมิติเป็นตัวแปรที่สามารถเข้าถึงได้ด้วยค่าตัวชี้หรือ index ค่าตัวแปรใน array อาจเรียกใช้โดยระบุชื่อ array และระบุตัวชี้index number ตัวแปร array จะมี index เริ่มต้นจาก 0 ตัวแปร array จะต้องตั้งค่า ก่อนจะนำไปใช้งาน และอาจกำหนดค่าเริ่มต้นหรือไม่ก็ได้

เช่น

intmyInts[6];

intmyPins[] = {2, 4, 8, 3, 6};

intmySensVals[6] = {2, 4, -8, 3, 2};

char message[6] = "hello";

Conversion การแปลงค่า

Char() แปลงค่าข้อมูลให้เป็น char
Byte() แปลงค่าข้อมูลให้เป็น byte
Int() แปลงค่าข้อมูลให้เป็น int
Word() แปลงค่าข้อมูลให้เป็น word
Long() แปลงค่าข้อมูลให้เป็น long
Float() แปลงค่าข้อมูลให้เป็น float

Variable Scope & Qualifiers

Variable scope

ตัวแปรสามารถตั้งค่าตอนเริ่มต้นโปรแกรมก่อน void setup(), หรือตั้งค่าตัวแปรภายในฟังชั่น,และบางครั้งก็ตั้งค่าตัวแปรภายในกลุ่มคำสั่ง for loop ซึ่งการตั้งค่าตัวแปรในแบบต่างๆ มีผลถึงขอบเขตการใช้ตัวแปร, หรืออีกนัยหนึ่งการที่โปรแกรมจะสามารถใช้ตัวแปรนั้น ตัวแปรแบบ global เป็นตัวแปรที่โปรแกรมมองเห็นและใช้งานได้จากทุกฟังชั่นและทุกกลุ่มคำสั่งในโปรแกรม ตัวแปรนี้จะตั้งค่าที่ตอนเริ่มต้นโปรแกรม, ก่อน setup() function
ตัวแปรแบบ local เป็นตัวแปรที่ตั้งค่าภายในฟังชั่นหรือภายในกลุ่มคำสั่ง for loop ตัวแปรนี้จะมองเห็นและใช้งานได้เฉพาะภายในฟังชั่นที่มันตั้งค่า ชื่อตัวแปรแบบนี้อาจมีชื่อซ้ำกันในแต่ละฟังชั่น แต่ในการทำงานฟังชั่นแต่ละตัวจะใช้ตัวแปรตัวนั้นเฉพาะที่ตั้งค่าภายในตัวมันเท่านั้น
เช่น

```
intgPWMval;  // any function will see this variable

void setup()
{
  // ...
}

void loop()
{
inti;    // "i" is only "visible" inside of "loop"
float f;  // "f" is only "visible" inside of "loop"
  // ...

for (int j = 0; j <100; j++){
  // variable j can only be accessed inside the for-loop brackets
  }

 }
```

Static

คำหลักแบบคงที่ใช้ในการสร้างตัวแปร ได้เพียงหนึ่งฟังก์ชั่น

เช่น

```
/* RandomWalk
* Paul Badger 2007
* RandomWalk wanders up and down randomly between two
* endpoints. The maximum move in one loop is governed by
* the parameter "stepsize".
* A static variable is moved up and down a random amount.
* This technique is also known as "pink noise" and "drunken walk".
*/

#define randomWalkLowRange -20
#define randomWalkHighRange 20
intstepsize;

intthisTime;
int total;

void setup()
{
Serial.begin(9600);
}

void loop()
{       //  tetstrandomWalk function
stepsize = 5;
thisTime = randomWalk(stepsize);
Serial.println(thisTime);
delay(10);
}

intrandomWalk(intmoveSize){
staticint  place;     // variable to store value in random walk - declared static so
that it stores
                      // values in between function calls, but no other functions can
change its value
```

```
place = place + (random(-moveSize, moveSize + 1));

if (place <randomWalkLowRange){                    // check lower and upper limits
place = place + (randomWalkLowRange - place);     // reflect number back in
positive direction
  }
else if(place >randomWalkHighRange){
place = place - (place - randomWalkHighRange);     // reflect number back in
negative direction
  }

return place;
}
```

Volatile

Volatile เอาไว้บอกคอมไพเลอร์ว่าตัวแปรที่ถูกประกาศมีโอกาสที่จะถูกเปลี่ยนค่าได้ เช่น

```
// toggles LED when interrupt pin changes state

int pin = 13;
volatileint state = LOW;

void setup()
{
pinMode(pin, OUTPUT);
attachInterrupt(0, blink, CHANGE);
}


void loop()
{
digitalWrite(pin, state);
}

void blink()
{
state = !state;
```

```
}
```

Const

คำหลัก constย่อมาจากค่าคงที่ มันเป็นตัวแปรที่ปรับเปลี่ยนพฤติกรรมของตัวแปรที่ทำให้ตัวแปร "อ่านอย่างเดียว"

เช่น

```
const float pi = 3.14;
float x;

// ....

x = pi * 2;    // it's fine to use const's in math

pi = 7;        // illegal - you can't write to (modify) a constant
```

Utilities

sizeof() นับจำนวน element ใน array

เช่น

```
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.write(myStr[i]);
    Serial.println();
  }
  delay(5000); // slow down the program
}
```

Functions

Digital I/O

-Pin Mode ()

Pin Mode ใช้ในกลุ่ม void setup() เพื่อกำหนดหน้าที่ขาของไมโครคอนโทรลเลอร์ให้เป็น ขารับสัญญาณ INPUT หรือขาส่งสัญญาณ OUTPUT

เช่น

```
int ledPin = 13;                // LED connected to digital pin 13
void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

-Digital Write ()

Digital Write คือส่งค่าลอจิกHIGH หรือ LOW (เปิด หรือ ปิด) ไปยังขา digital ที่กำหนด หมายเลขขาไอซีอาจกำหนดเป็นตัวแปรหรือค่าคงที่ (0-13)

เช่น

```
 intledPin = 13;                // LED connected to digital pin 13
void setup()
{
pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}
void loop()
{
digitalWrite(ledPin, HIGH);   // sets the LED on
delay(1000);                  // waits for a second
digitalWrite(ledPin, LOW);    // sets the LED off
delay(1000);                  // waits for a second
}
```

-Digital Read ()

Digital Read คำสั่งนี้อ่านค่าจาก ขาไอซีที่ถูกกำหนดให้เป็น digital pin ซึ่งจะได้ผลลัพท์เป็น HIGH หรือ LOW หมายเลขขาไอซีอาจกำหนดเป็นตัวแปรหรือค่าคงที่ (0-13)

เช่น

```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7;   // pushbutton connected to digital pin 7
int val = 0;     // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin 13 as output
  pinMode(inPin, INPUT);       // sets the digital pin 7 as input
}
void loop()
{
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);    // sets the LED to the button's value
}
```

Analog I/O

- Analog Reference ()

คำสั่งนี้ควรจะใช้ในการตั้งแรงดันอ้างอิงซึ่งเป็นสิ่งจำเป็นสำหรับอนาล็อก ตัวเลือกที่เป็นไปได้ หลังจากที่แรงดันไฟฟ้าอ้างอิงอนาล็อกที่มีการเปลี่ยนแปลงก็อาจจะดีว่าผลแรกจากอนาล็อกอ่าน () จะไม่ถูกต้อง

- Analog Read ()

คำสั่งนี้อ่านค่าจากขา Analog จะได้ค่า 10 bit คำสั่งนี้จะทำงานกับขา analog input (0-5) เท่านั้น และได้ผลลัพท์เป็นเลขจำนวนเต็มค่า 0 – 1023

เช่น

```
intanalogPin = 3;      // potentiometer wiper (middle terminal) connected to
analog pin 3
 // outside leads to ground and +5V
intval = 0;           // variable to store the value read
void setup()
{
Serial.begin(9600);        //  setup serial
}
void loop()
{
val = analogRead(analogPin);    // read the input pin
Serial.println(val);              // debug value
```

```
}
```

- AnalogWrite() - PWM

เป็นคำสั่งเขียนค่า analog เทียมโดยใช้ hardware enabled pulse width mdulation(PWM) ไปยังขา oututที่สามารถทำ PWM ได้ ใน Arduinoรุ่นใหม่ที่ใช้ชิพ Atmega168 คำสั่งนี้จะทำงานกับขา 3, 5, 6, 9, 10, และ 11 ส่วน Arduinoรุ่นเก่าที่ใช้ Atmega8 จะรองรับเพียงขา 9, 10 และ 11 ค่าที่เขียนสามารถใช้เป็นตัวแปรหรือค่าคงที่จาก 0 – 255

เช่น

```
intledPin = 9;      // LED connected to digital pin 9
intanalogPin = 3;   // potentiometer connected to analog pin 3
intval = 0;         // variable to store the read value
voidsetup()
{
  pinMode(ledPin, OUTPUT);   // sets the pin as output
}
voidloop()
{
  val = analogRead(analogPin);   // read the input pin

   analogWrite(ledPin, val / 4);  // analogRead values go from 0 to 1023,
analogWrite values from 0 to 255
}
```

Due only ใช้งานกับบอร์ด Due เท่านั้น

- AnalogReadResolution()

Analog Read Resolution () เป็นส่วนขยายของ API อนาล็อกสำหรับ Arduinoเนื่องจาก ชุด ขนาด (ในบิต) ของค่าที่ส่งกลับโดย Analog Read Resolution () มันเริ่มต้นเป็น 10 บิต (ค่าตอบแทนระหว่าง 0-1023) สำหรับความเข้ากันได้ย้อนหลังกับตามแผง AVR

เช่น

```
void setup() {
  // open a serial connection
  Serial.begin(9600);
}
void loop() {
  // read the input on A0 at default resolution (10 bits)
  // and send it out the serial connection
  analogReadResolution(10);
```

```
  Serial.print("ADC 10-bit (default) : ");
  Serial.print(analogRead(A0));

  // change the resolution to 12 bits and read A0
  analogReadResolution(12);
  Serial.print(", 12-bit : ");
  Serial.print(analogRead(A0));

  // change the resolution to 16 bits and read A0
  analogReadResolution(16);
  Serial.print(", 16-bit : ");
  Serial.print(analogRead(A0));

  // change the resolution to 8 bits and read A0
  analogReadResolution(8);
  Serial.print(", 8-bit : ");
  Serial.println(analogRead(A0));

  // a little delay to not hog serial monitor
  delay(100);
}
```

- AnalogWriteResolution()

Analog Write Resolution () เป็นส่วนขยายของAPI อนาล็อกสำหรับ Arduinoเนื่องจาก Analog Write Resolution () กำหนดความละเอียดของการ Analog Write Resolution () ฟังก์ชัน มันเริ่มต้นเป็น 8 บิต (ค่าระหว่าง 0-255) สำหรับความเข้ากันได้ย้อนหลังกับตามแผง AVR

เช่น

```
void setup(){
  // open a serial connection
  Serial.begin(9600);
  // make our digital pin an output
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}
void loop(){
  // read the input on A0 and map it to a PWM pin
```

```
  // with an attached LED
  int sensorVal = analogRead(A0);
  Serial.print("Analog Read) : ");
  Serial.print(sensorVal);

  // the default PWM resolution
  analogWriteResolution(8);
  analogWrite(11, map(sensorVal, 0, 1023, 0 ,255));
  Serial.print(" , 8-bit PWM value : ");
  Serial.print(map(sensorVal, 0, 1023, 0 ,255));

  // change the PWM resolution to 12 bits
  // the full 12 bit resolution is only supported
  // on the Due
  analogWriteResolution(12);
  analogWrite(12, map(sensorVal, 0, 1023, 0, 4095));
  Serial.print(" , 12-bit PWM value : ");
  Serial.print(map(sensorVal, 0, 1023, 0, 4095));

  // change the PWM resolution to 4 bits
  analogWriteResolution(4);
  analogWrite(13, map(sensorVal, 0, 1023, 0, 127));
  Serial.print(", 4-bit PWM value : ");
  Serial.println(map(sensorVal, 0, 1023, 0, 127));
  delay(5);
}
```

Advanced I/O

- Tone ()

Tone คือการ สร้างคลื่นสี่เหลี่ยมความถี่ที่ระบุ (50% และเอฟเอ็ม) ในขา ระยะเวลาที่สามารถระบุมิฉะนั้นคลื่นจะไม่หยุดจนกว่าจะได้รับการเรียกร้องให้NOTONE () พินสามารถเชื่อมต่อกับออด Piezoหรือสถานที่ตั้งลำโพง
มันสามารถเสมอถูกสร้างขึ้นพร้อมกันหนึ่งเสียง ถ้าเสียงที่มีอยู่แล้วเล่นบนขาที่แตกต่างกันเสียงโทร () ไม่มีผล ถ้าเสียงที่เล่นบนขาเดียวกันจะมีการกำหนดความถี่ของการโทร

- No Tone()

No Tone คือ หยุดการสร้างคลื่นสี่เหลี่ยมที่เกิดจาก Tone () มีผลถ้าไม่มีเสียงจะถูกสร้างขึ้นไม่มี

- Shift Out()

Shift Out ของข้อมูลหนึ่งบิต เริ่มต้นจากสิ่งที่สำคัญที่สุด (เช่นซ้ายสุด) ที่สำคัญน้อยที่สุด (ขวาสุด IE) บิต บิตในการเปิดแต่ละครั้งจะถูกเขียนด้วยขาข้อมูลหลังจากที่เข็มนาฬิกาชีพจร (สูงต่ำแล้ว) คือการแสดงให้เห็นว่าขาสามารถใช้ได้

เช่น

```
//************************************************************//
//  Name    : shiftOutCode, Hello World                       //
//  Author  : CarlynMaw,TomIgoe                               //
//  Date    : 25 Oct, 2006                                    //
//  Version : 1.0                                             //
//  Notes   : Code for using a 74HC595 Shift Register         //
//          : to count from 0 to 255                          //
//************************************************************
//Pin connected to ST_CP of 74HC595
int latchPin = 8;
//Pin connected to SH_CP of 74HC595
int clockPin = 12;
////Pin connected to DS of 74HC595
int dataPin = 11;
void setup() {
  //set pins to output because they are addressed in the main loop
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}
void loop() {
  //count up routine
  for (int j = 0; j < 256; j++) {
    //ground latchPin and hold low for as long as you are transmitting
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, j);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(latchPin, HIGH);
    delay(1000);
  }
}
```

- ShiftIn()

Shift In () ของข้อมูลหนึ่งบิต เริ่มต้นจากสิ่งที่สำคัญที่สุด (เช่นซ้ายสุด) ที่สำคัญน้อยที่สุด (ขวาสุด IE) บิต สำหรับบิตของขานาฬิกาแต่ละคนจะตั้งค่าให้สูงอ่านบิตต่อไปจากสายข้อมูลและเข็มนาฬิกาให้ต่ำลงอีกครั้ง

- Pulse In ()

Pulse In (สูงหรือต่ำ) เมื่อขาออก ตัวอย่างเช่นถ้าค่าสูงรอ Pulse In จนกระทั่งขาไป HIGH, เวลาเริ่มต้นแล้วรอจนขาอยู่ในระดับต่ำและเวลาที่จะหยุด ถูกส่งกลับไปตามความยาวของ Pulse In ใน microseconds หลังจากเวลาที่แน่นอน, 0 ถูกส่งกลับถ้าไม่มี Pulse In มานับเวลาฟังก์ชั่นนี้ได้รับการพิจารณาตามประสบการณ์และจะแสดงข้อผิดพลาดในอีกพอร์ตอื่นต่อไป ทำงานร่วมกับพอร์ตไมโคลระหว่าง 10 วินาทีและ 3 นาที

เช่น

```
int pin = 7;
unsigned long duration;
void setup()
{
pinMode(pin, INPUT);
}
void loop()
{
duration = pulseIn(pin, HIGH);
}
```

Time

- Millis ()

คำสั่งนี้จะได้ผลลัพธ์ค่าเวลาเป็นมิลลิวินาทีแสดงค่าที่ Arduino board1 เริ่มต้นทำโปรแกรมปัจจุบัน ค่าที่ได้เป็นค่า unsigned long ขนาด 32 bit

เช่น

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

```
delay(1000);
}
```

- Micros ()

ส่งกลับจำนวนของ microseconds ที่ได้ผ่านไปนับตั้งแต่บอร์ด Arduinoทำงานปัจจุบัน โปรแกรม จำนวน microseconds ตั้งแต่โปรแกรมเริ่มต้น

เช่น

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

Delay ()

หยุดการทำงานโปรแกรมสำหรับจำนวนของเวลา (ใน milliseconds) ระบุเป็นพารามิเตอร์ (มี 1,000 มิลลิวินาทีในที่สองเป็น.)Milliseconds = จำนวนมิลลิวินาทีในการหยุดการทำงาน ชั่วคราว

เช่น

```
int ledPin = 13;                  // LED connected to digital pin 13
void setup()
{
  pinMode(ledPin, OUTPUT);        // sets the digital pin as output
}
void loop()
{
  digitalWrite(ledPin, HIGH);     // sets the LED on
  delay(1000);                    // waits for a second
  digitalWrite(ledPin, LOW);      // sets the LED off
  delay(1000);                    // waits for a second
}
```

- Delay Microseconds ()

จำนวน microseconds เพื่อหยุดโปรแกรมจำนวนของเวลา (ใน microseconds) ระบุเป็นพารามิเตอร์ มีพัน microseconds ในมิลลิวินาที

เช่น

```
intoutPin = 8;                  // digital pin 8
void setup()
{
pinMode(outPin, OUTPUT);       // sets the digital pin as output
}
void loop()
{
digitalWrite(outPin, HIGH);   // sets the pin on
delayMicroseconds(50);         // pauses for 50 microseconds
digitalWrite(outPin, LOW);    // sets the pin off
delayMicroseconds(50);         // pauses for 50 microseconds
}
```

Math

- Min ()

คำสั่งนี้คำนวณหาค่าที่น้อยกว่าของค่าที่ให้มาในวงเล็บและคืนค่าที่น้อยกว่า

เช่น

```
 sensVal = min(sensVal, 100);  // assigns sensVal to the smaller of sensVal or 100
    // ensuring that it never gets above 100.
```

- Max()

คำสั่งนี้คำนวณหาค่าที่มากกว่าของค่าที่ให้มาในวงเล็บและคืนค่าที่มากกว่า

เช่น

```
sensVal = max(senVal, 20);  // assigns sensVal to the larger of sensVal or 20
   // (effectively ensuring that it is at least 20)
```

- Abs()

คำนวณค่าสัมบูรณ์ของจำนวนเนื่องจากวิธีการ abs () ฟังก์ชันจะดำเนินการหลีกเลี่ยงการใช้ฟังก์ชั่นอื่น ๆ ภายในวงเล็บอาจนำไปสู่ผลลัพธ์ที่ไม่ถูกต้อง

เช่น

- Constrain()

Constrain คือการจำกัดจำนวน Constrain จะจำกัดประเภทข้อมูลทั้งหมดทั้ง ปลายบนและปลายล่างของช่วงประเภทข้อมูล

เช่น

```
sensVal = constrain(sensVal, 10, 150);  // limits range of sensor values to between 10 and 150
```

- Map()

แผนที่จำนวนจากช่วงหนึ่งไปยังอีก นั่นคือค่าของ from Low จะได้รับการmapไป to Low ค่าของจากสูงไป to High ค่าในระหว่างกับค่าในระหว่าง ฯลฯ

เช่น

```
/* Map an analog value to 8 bits (0 to 255) */
void setup() {}
void loop()
{
  int val = analogRead(0);
  val = map(val, 0, 1023, 0, 255);
  analogWrite(9, val);
}
```

- Pow()

คำนวณค่าของจำนวนที่ยกกำลัง Pow () สามารถใช้เพื่อเพิ่มจำนวนการใช้พลังงานที่เป็นเศษส่วน นี้จะเป็นประโยชน์สำหรับการสร้างแผนที่การชี้แจงของค่าหรือเส้นโค้ง

- Sqrt()

คำนวณค่ารากที่สองของจำนวน

Trigonometry

- Sin ()

คำนวณค่า sin ของมุม (เรเดียน) ผลที่ได้จะอยู่ระหว่าง -1 และ 1

- Cos ()

คำนวณค่า cosของมุม (เรเดียน) ผลที่ได้จะอยู่ระหว่าง -1 และ 1

- Tan ()

คำนวณแทนเจนต์ของมุม (เรเดียน) ผลที่ได้จะอยู่ระหว่างอินฟินิตี้ลบและอินฟินิตี้

Random Number

- Random Seed ()

เป็นการกำหนดค่าเริ่มต้นให้กับ function random.Random() ที่จะใช้ในครั้งต่อไป โดยค่าที่ออกมาจะเป็นค่าเดิมเสมอ ซึ่ง function นี้ไม่มีการ return ค่าออกมา

เช่น

```
longrandNumber;

void setup(){
Serial.begin(9600);
randomSeed(analogRead(0));
}

void loop(){
randNumber = random(300);
Serial.println(randNumber);

delay(50);
}
```

- **Random ()**

ฟังก์ชันสุ่มตัวเลข

เช่น

```
longrandNumber;

void setup(){
Serial.begin(9600);

  // if analog input pin 0 is unconnected, random analog
  // noise will cause the call to randomSeed() to generate
  // different seed numbers each time the sketch runs.
  // randomSeed() will then shuffle the random function.
randomSeed(analogRead(0));
}

void loop() {
  // print a random number from 0 to 299
randNumber = random(300);
Serial.println(randNumber);

  // print a random number from 10 to 19
randNumber = random(10, 20);
Serial.println(randNumber);
```

```
delay(50);
}
```

Bits and Bytes

- Low Byte ()
แยกตัวแปรที่เป็น byte ที่ต่ำสุดออก
- **High Byte ()**
แยกตัวแปรที่เป็น byte ที่สูงสุดหรือมีขนากใหญ่ออก
- **Bit Read ()**
อ่านบิตของจำนวน
- **Bit Write ()**
เขียนบิตของตัวแปรที่เป็น "0" , "1"
- **Bit Set ()**
เซตค่าบิตของตัวแปรที่เป็น "1"
- **Bit Clear ()**
Clear บิตของตัวแปรที่เป็น "0"
- Bit ()
คำนวณค่าของบิตที่ระบุ

External Interrupts การขัดจังหวะจากภายนอก
- Attach Interrupt ()
การกำหนดฟังก์ชันให้เกิดการขัดจังหวะที่ภายนอก
เช่น

```
int pin = 13;
volatile int state = LOW;

void setup()
{
  pinMode(pin, OUTPUT);
  attachInterrupt(0, blink, CHANGE);
}

void loop()
{
  digitalWrite(pin, state);
}
```

```
void blink()
{
  state = !state;
}
```

- **Detach Interrupt ()**
  ปิดการกำหนดการขัดจังหวะ

Interrupts การขัดจังหวะจากภายใน

- Interrupts ()
  การขัดจังหวะ
เช่น

```
void setup() {}

void loop()
{
noInterrupts();
  // critical, time-sensitive code here
interrupts();
  // other code here
}
```

- No Interrupts ()
  ปิดการใช้งานการขัดจังหวะ
เช่น

```
void setup() {}

void loop()
{
noInterrupts();
  // critical, time-sensitive code here
interrupts();
  // other code here
}
```

Communication การสื่อสารข้อมูล

- Serial

ใช้สำหรับการสื่อสารระหว่างบอร์ด Arduinoและคอมพิวเตอร์หรืออุปกรณ์อื่น ๆ ทั้งหมด

- **Stream**

Stream เป็นชั้นฐานสำหรับลักษณะตามไบนารี

ที่มา:แปลจาก > https://www.arduino.cc/en/Reference/HomePage