ARDUINO SOFTWARE

Arduino Hardware & Software Education KIT

PROGRAMMING ¦ SENSOR ¦ ROBOTICS ¦ IOT

MECHASOLUTION

PROGRAMMING BOOK

01장. 프로그래밍 시작하기

02장. 아두이노 프로그래밍 기초

03장. 아두이노 프로그래밍 함수

1. 프로그래밍 시작하기

1.1 소프트웨어 교육

> 이 나라 모든 사람들이 컴퓨터 프로그래밍을 배워야 하는
> 이유는 사고(생각)하는 법을 배워야 하기 때문이다.
>
> -Steve Jobs-

오늘날 우리들은 소프트웨어 시대에 살아가고 있다. 스마트폰에서 작동하는 어플리케이션, 컴퓨터 게임, 인터넷 등의 소프트웨어는 당연하다는듯 우리의 일상생활에서 사용되고 있지만 정작 이의 원리나 구조는 알지 못한다.

이러한 시대에서 프로그래밍 교육은 프로그램을 사용하는 것 뿐만 아니라, 프로그램을 직접 설계하고 만듬으로써, 소프트웨어 개발자로써의 역량을 갖출 수 있으며, 그 과정에서 창의성과 문제해결 능력을 향상시킬 수 있는 바탕이 된다.

이러한 이유로 미국, 유럽 등의 선진국에서는 프로그래밍 교육을 필수 교육과목으로 채택하여 국가 교육과정으로 운영하고 있으며, 정보화 시대에 걸맞는 인재를 양성하고 있다.

그림1-1 해외 프로그래밍 교육 (code.org 사진)

페이스북과 마이크로소프트 등의 IT 기업에서 지원하는 코드닷오알지(code.org)에서는 누구나 프로그래밍을 쉽게 접근할 수 있도록 교육을 진행하고 있다. 이에 우리나라도 다양한 사례가 생기고 있으며, 그 중 네이버에서 운영하는 '소프트웨어야 놀자' 또한 학생들이 쉽게 소프트웨어를 공부할 수 있도록 서비스를 제공하고 있다.

1.2 프로그래밍이란?

프로그래밍은 프로그램을 만드는 과정을 말한다. 그렇다면 프로그램은 무엇일까? 프로그램은 일의 순서(과정)을말한다. 한가지 프로그램을 예로 들어보자.

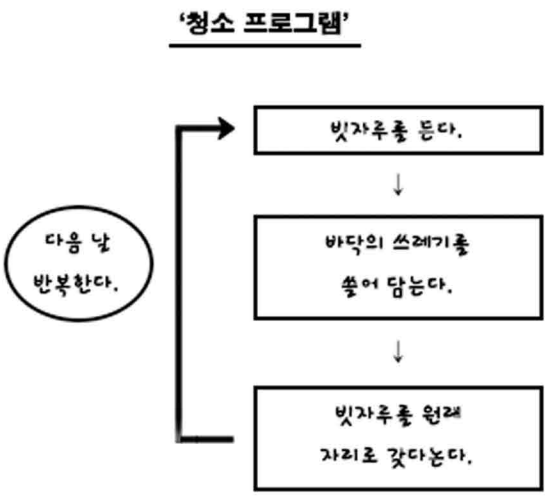

그림2-1 청소 프로그램의 예

위의 그림은 청소하는 과정을 그림으로 나타낸 것이며, 이 또한 프로그램 중 하나라고 할 수 있다. 우리가 스마트폰이나 컴퓨터에 사용하는 인터넷, 게임 등의 프로그램은 컴퓨터를 통해 제작하는 '컴퓨터 프로그램'이며, C언어, 파이썬 등의 프로그래밍 언어를 사용하여 만들 수 있다. 프로그래밍 언어란, 사람과 컴퓨터가 소통하기 위한 언어를 말하며, C언어, 파이썬, 블록형 언어 등 많은 언어들이 존재한다. 각 언어들은 사용 난이도나 사용목적, 범위 등이 모두 제 각각이다.

1.3 하드웨어와 소프트웨어란?

예를 들자면, 컴퓨터 본체는 하드웨어이고, 컴퓨터에서 실행되는 게임 프로그램, 인터넷 프로그램, 휴지통 프로그램 등은 소프트웨어에 해당된다. 다른 예로 스마트폰은 하드웨어, 어플리케이션이 소프트웨어라고 할 수 있다. 이 처럼 눈에 보이고 만질 수 있는 것들을 하드웨어라고 하며, 눈에 보이지 않는 명령어의 모음(프로그램)을 소프트웨어라고 한다. 우리가 앞으로 배울 아두이노 보드는 하드웨어에 해당되며, 이 안에 소프트웨어 (프로그램)를 입력함으로써 아두이노 보드(하드웨어)를 작동시킬 수 있다.

1.4 프로그래밍 배우기

프로그래밍 공부는 물리적인 공간이나 자원에 제약이 없으며, 인터넷이나 책 등을 통해 어렵지 않게 접근할 수 있다. 하지만 프로그래밍을 처음 접하게 되면 외계어 같이 생긴 프로그래밍 언어가 두려울 수 밖에 없고, 수 많은 프로그래밍 언어 중 어떤 언어를 배워야 할지, 그리고 어떤 방법으로 접근해야할지 막막하기만 하다. 그렇기 때문에 이 매뉴얼에서는 프로그래밍을 처음 시작하기 전에 알아야 할 것들을 소개하고, 더 나아가 아두이노 플랫폼을 통해 직접 프로그래밍을 진행해 보도록 한다.

1.5 프로그래밍 언어의 종류

프로그래밍 언어는 사람과 컴퓨터의 의사소통을 가능하게 해주는 언어이다. 우리는 컴퓨터에 A를 입력하지만 사실 컴퓨터는 이를 1과 0으로 이루어진 기계어로 인식한다. 1과 0뿐이 사용하지 않는 컴퓨터에서 프로그램을 제작하기란 쉽지 않다. 그렇기 때문에 만들어진 것이 C, C++, Python과 같은 프로그래밍 언어이다. 수많은 프로그래밍 언어들은 각각의 특징과 장단점이 있으며, 그에 맞게 여러 분야에서 사용되고 있다. 이 매뉴얼에서는 프로그래밍 언어 중 가장 많이 사용되고 있는 C, Java, Python에 대해 간단하게 소개해보도록 하겠다.

C 언어

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```

C언어는 오늘날 영어와 같은 위치를 차지하고 있는 프로그래밍 언어이다. 실질적으로 모든 컴퓨터 시스템에서 사용할 수 있는 프로그래밍 언어이지만, 그 만큼 배워야 할 영역이 넓다. 빠른 연산속도를 자랑하며, 운영체제 및 디바이스 드라이버, 마이크로 컨트롤러 임베디드 프로그래밍 등에 사용되고 있다. 프로그래밍의 거이 모든 영역에 걸쳐 도움이 되지만 관련 지식 범위가 넓기 때문에 모두 배우기란 매우 어렵다. 하지만 기초부터 차근차근 배우고 싶다면 C언어로 프로그래밍을 시작하는것도 나쁘지 않다.

본 매뉴얼에서 소개할 아두이노 플랫폼은 C, C++기반의 언어를 사용하므로 우리가 배울 언어이기도 하다. (C++언어는 C에서 객체 지향형 언어로 발전된 언어이다.)

Java

```java
package arduino.mechasolution;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

객체 지향적 언어로 개발된 프로그래밍 언어이다. C언어는 절차 지향적 언어로 소스코드의 짜여진 순서 위주로 동작하지만, Java는 절차 지향적 언어와는 다르게 각각의 구성요소(객체)가 중심이 되어 동작하는 언어이다. Java는 안드로이드 앱 개발이나 응용 프로그램 영역에서 주로 사용되고 있으며, 높은 생산성과 호환성이 특징이다. 앱 개발이나 응용 프로그램 개발에 관심이 많다면 Java로 프로그래밍을 입문하는것도 나쁘지 않은 선택이다.

파이썬 (Python)

```python
print('hello world!')
```

문법 구조가 타 언어에 비해 매우 쉽기 때문에 초보자들의 입문용 언어로 추천되는 언어이다. 실제로 국내의 많은 대학에서도 입문용 언어로 많이 사용되고 있다. 파이썬의 장점은 전체적으로 코딩 해야하는 C언어와 다르게 한 줄, 한줄 언어를 작성, 실행할 수 있다는 것이다. 이러한 장점으로 인해 개발 기간이 매우 단축되는 것이 특징이다. 하지만 단점으로는 타 언어에 비해 연산속도가 느리며, 복잡한 프로그래밍에는 적합하지 않다는 평이 있다. Pyhon은 굉장히 쉬운 편이지만 복잡한 프로그램을 만들기에는 약간 제한이 따른다. (하지만 할 수 있는건 많다.) 비교적 쉽게 프로그래밍을 배우고 싶다면 Python을 추천.

1.6 컴파일러와 인터프리터

컴파일러는 작성한 소스코드를 컴퓨터가 이해할 수 있는 기계어로 변환해주는 프로그램을 말한다. 소스코드를 작성할 때 오타나 잘못 된 부분이 있으면 컴파일 에러가 발생하게 되는데, 이는 프로그래밍 언어를 기계어로 변환하는 과정에서 정상적으로 인식하지 못하였기 때문이다. 아두이노에서 사용되는 통합개발환경 (IDE : 스케치) 또한 컴파일러에 해당된다.
인터프리터는 컴파일 과정 없이 코드를 읽으면서 동작하는 프로그램을 말한다. Java Script, Python, LUA등의 언어가 이에 해당된다. 컴파일을 하는 프로그램에 비해 동작이 느리지만 코드의 동작을 확인하고 알고리즘을 수정하는 등의 작업은 컴파일러를 이용하는 것보다 더 수월하다.

2. 아두이노 프로그래밍 기초

앞서 다양한 프로그래밍 언어를 알아보았으나 우리가 이 매뉴얼에서 공부하게 될 주 프로그래밍 언어는 당연히 아두이노 프로그래밍 언어이다.

아두이노 보드는 마이크로 컨트롤러(아주 작은 컴퓨터)를 탑재한 하드웨어이다. 이 보드를 제어하기 위해서는 프로그래밍 과정을 통해 프로그램을 만들어 아두이노 보드에 설치(업로드)해줘야 한다. 이 과정에 사용되는 컴파일러는 아두이노 통합개발환경(IDE)이라는 컴퓨터 프로그램이며, C, C++ 기반의 언어로 작동한다. 그렇기 때문에 이 매뉴얼을 통해 아두이노를 공부하게 되면 C와 C++의 기본을 함께 공부할 수 있다.

2.1 아두이노 프로그램의 기본 구조

아두이노 프로그램은 크게 void setup()과 void loop() 부분으로 구성되어 있다.

```
void setup() {
  // put your setup code here, to run once:

}
```

void setup() 안에 들어가는 코드는 프로그램이 시작될 때 한번만 실행되는 코드이다. 주로 초기 설정을 넣어준다.

```
void loop() {
  // put your main code here, to run repeatedly:

}
```

void loop() 안에 들어가는 코드는 void setup()이 한번 실행 된 후 반복적으로 실행되는 코드이다. 프로그램의 주 내용이 들어간다.

void setup() 혹은 void loop()가 선언되기전에 변수 혹은 라이브러리 추가와 같은 코드들이 사용되기도 한다. 이는 프로그램에 대한 설정에 가깝다고 보면 된다. 그 외에 프로그램을 만들 때 사용되는 기본 요소들은 아래와 같다.

■ 명령어 - 길거나 짧은 모든 프로그래밍 언어(코드)를 지칭한다.
■ 중괄호 { } - 함수 혹은 여러 명령을 하나로 묶기 위해 사용한다.
■ 세미콜론 ; - 명령의 끝을 나타낸다.
■ 대, 소문자 - 대문자와 소문자를 구분하지 않을 경우 컴파일 오류가 발생한다.
■ 주석 - 코드에 대한 메모를 적을 때 사용하며 프로그램에는 아무런 영향을 주지 않는다. // 뒤에 메모를 적어 사용할 수 있다. 여러줄은 /* 메모*/를 사용

2.2 변하는 수, 변수

변수란 이름 그대로 해석해보자면 '변할 수 있는 수'라고 해석할 수 있다. 즉 고정되어 있지 않은 수를 말한다. 직관적인 의미로 표현해보자면 변수는 '변할 수 있는 수를 담아두는 공간'으로 표현할 수 있다.

변수를 사용하는 이유는 정수나 실수, 문자 등의 자료를 다루기 위해서 이다.
사람의 경우 '천 번은 1000번이다'라는 문자열에서 천과 1000의 의미가 같다는 것을 알 수 있지만, 컴퓨터는 이를 문자 그대로 보기 때문에 두 의미가 같다는 것을 알지 못한다. 이에 변수를 선언, 정의하여 사용한다.

변수는 다음과 같이 선언 할 수 있다.

```
int val = 2;  // 정수형 변수 val를 선언하고, 2를 대입한다.
```

변수는 선언하기 전에 데이터의 형식이 정수인지, 실수인지, 문자열인지 그리고 몇 바이트를 사용할 것인지 정의한다. (바이트는 저장공간의 크기로 생각하면 된다.)

변수의 기본 자료형(데이터의 타입)은 다음과 같다.

자료형		크기	표현범위	
			최소	최대
문자형	char	1바이트	아스키코드	
정수형	byte	1바이트	0	255
	int	2바이트	-32768	32767
	long	4바이트	-2147483648	2147483647
실수형	float	4바이트	-3.4028235E+38	3.4028235E+38
	double	4바이트	-3.4028235E+38	3.4028235E+38

타입이 여러 가지가 존재하는 이유는 데이터의 표현방식이 다양하고(정수인지, 실수인지, 문자열인지) 효율적인 메모리 공간을 활용하기 위함이다.

2.3 변하지 않는 수, 상수

앞에서 설명한 변수가 '변할 수 있는 수'라면 상수는 '변하지 않는 고정된 수'를 말한다. int val = 2; 에서 val가 변수라면 2는 상수에 해당된다. 이 때 상수(2) 또한 데이터 타입(int)을 맞춰줘야 하므로 주의한다.

2.4 연산자의 종류

수학에 사칙 연산자(덧셈, 뺄셈, 곱셈, 나눗셈)가 있듯 프로그래밍 언어에도 이와 같은 연산자들이 존재한다. 각 연산자들은 수치, 논리, 비교와 같은 연산을 표시하기 위해 존재하며 그 종류는 다음과 같다.

=	오른쪽 값을 왼쪽에 대입	A=B
+	더하기	A+B
-	빼기	A-B
*	곱하기	A*B
/	나누기	A/B
%	나머지 구하기	A=3, B=2 일 경우 A%B 는 1
++	1증가 시키기	A++
--	1감소 시키기	A--
==	같다면	A==B
!=	다르다면	A!=B
>	A가 B보다 크면	A>B
<	A가 B보다 작으면	A<B
>=	A가 B보다 크거나 같으면	A>=B
<=	A가 B보다 작거나 같으면	A<=B
&&	(AND) A그리고 B가	A&&B
||	(OR) A 또는 B가	A||B
!	(NOT) A가 아니면	!A

2.5 함수의 구조

함수란 소프트웨어에서 특정 동작을 수행하는 일정 코드 부분을 말한다. 기계로 비유하자면 구성 부품이라 할 수 있다. 함수의 종류는 크게 표준 함수와 사용자 정의 함수로 나눌 수 있다.

표준 함수는 프로그래밍 언어 자체에서 제공되어 컴파일러와 함께 배포되는 함수로, 함수의 기능과 호출하는 방법만 알고 있으면 언제든지 사용할 수 있다.

사용자 정의 함수는 사용자가 직접 만든 함수를 말한다. 제공되지 않은 코드를 직접 만들어 사용하거나 만들어진 라이브러리를 불러와 응용, 사용할 수 있다.

함수의 기본 구조는 다음과 같다.

```
반환타입 함수이름(매개변수){
  함수 본체;
  반환 값;
}
```

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

반환 타입은 변수에서 배운 내용이다. 반환 값이 정수형일 경우 int, 실수형일 경우 double, 없을 경우 void를 입력한다. 정해진 함수의 이름과 매개변수를 입력하여 함수를 호출하며, 호출된 함수는 제 기능에 맞게 수행된다.

2.6 라이브러리

라이브러리는 사용자가 직접 만든 코드(사용자 정의 함수)를 의미한다. 아두이노의 장점 중 하나는 많은 사람들이 라이브러리를 만들어 이를 공유한다는것이다. 즉, 이미 만들어져 있는 라이브러리 찾아 사용하면 간단하게 프로젝트를 진행할 수 있다 . 라이브러리를 사용하기 위해서는 통합개발환경이 설치된 폴더의 libraries 폴더에 라이브러리 구성 폴더를 넣은 후 아두이노 프로그램 프로그래밍시 '#include 라이브러리명' 코드를 선언하여 사용할 수 있다.

3. 아두이노 프로그래밍 함수

아두이노 프로그래밍시 자주 사용되는 함수들에 대해 알아본다.

3.1 디지털 신호 입출력 함수

■ pinMode(핀번호, 모드)

사용할 디지털 핀의 모드(입력or출력)를 설정하기 위한 함수이다.

매개 변수

핀번호 - 모드를 설정할 디지털 핀 번호를 적는다.

모드 - OUTPUT 또는 INPUT 또는 INPUT_PULLUP

OUTPUT : 디지털 신호 출력모드로 설정한다.
INPUT : 디지털 신호 입력모드로 설정한다.
INPUT_PULLUP : 회로없이 풀업저항을 사용한다.

■ digitalWrite(핀번호, 신호)

설정 핀에 디지털 신호를 출력하기 위한 함수이다.

매개 변수

핀번호 - 신호를 출력할 디지털 핀 번호를 설정한다.

모드 - HIGH 또는 LOW

HIGH : HIGH신호(5V)를 출력한다.
LOW : LOW신호(0V)를 출력한다

■ digitalRead(핀번호)

설정된 핀에 가해지는 디지털 신호를 읽는다.

매개 변수

핀번호 - 신호를 입력받을 핀을 설정한다.

■ pulseIn(핀번호, 신호) 또는 pulseIn(핀번호, 신호, 시간제한)

설정 신호를 다시 입력 받을 때까지 걸리는 시간을 측정하기 위한 함수이다.

매개 변수

핀번호 - 함수를 사용할 핀을 설정한다.

신호 - HIGH 또는 LOW

HIGH : HIGH신호가 다시 입력되기까지.

LOW : LOW신호가 다시 입력되기까지

시간제한 - 제한 시간을 설정한다. 지정한시간만큼 기다려도 신호 변화가 없을경우 0을 반환하며, 마이크로 초 단위로 지정할 수 있다. 지정하지 않을 경우 1초(1,000,000)로 자동 설정된다.

반환 값: 설정 신호가 입력될 때까지의 시간을 반환한다. 지정된 시간동안 신호변화가 없을 경우 0을 반환한다.

3.2 아날로그 신호 입출력 함수

■ analogWrite(핀번호, 값)

설정핀에 아날로그 값의 신호를 출력하기 위한 함수이다.

매개 변수

핀번호 - 아날로그 신호를 출력할 핀을 설정한다.

값 - 출력할 값(0~255)을 설정한다.

■ analogRead(핀번호)

설정핀에 입력되는 아날로그 신호를 읽기 위한 함수이다.

매개 변수

핀번호 - 아날로그 신호를 입력받을 핀을 설정한다.

반환 값: 입력 받은 신호를 0~1023 값으로 반환 받는다.

3.3 시간 관련 함수

■ delay(시간)

다음 명령어(코드)까지 입력 시간만큼 대기한다.

매개 변수

시간 - 밀리초 단위로 숫자를 입력한다. 1000 = 1초.

■ delayMicroseconds(시간)

다음 명령어(코드)까지 입력 시간만큼 대기한다.

매개 변수

시간 - 마이크로초 단위로 숫자를 입력한다. 1,000,000 = 1초

■ millis()

아두이노가 켜진 이후의 시간을 밀리초 단위로 반환하는 함수이다.

반환 값

아두이노가 켜진 이후의 시간을 밀리초 단위(1000=1초)로 반환한다. 0~4294967295 범위이며, 대략 50일 정도된다. 최대치가 넘어가면 0부터 다시 시작된다.

■ micros()

아두이노가 켜진 이후의 시간을 마이크로초 단위로 반환하는 함수이다.

반환 값

아두이노가 켜진 이후의 시간을 마이크로초 단위(1,000,000=1초)로 반환한다. 0~4294967295 범위이며, 대략 71분 정도된다. 최대치가 넘어가면 0부터 다시 시작된다.

3.4 시리얼 통신 관련 함수

■ Serial.begin(통신속도)

시리얼 통신의 시작을 알리며, 통신속도를 설정한다.

매개 변수

통신속도 - 300~115200, 보통 9600으로 설정한다.

■ Serial.print(값) 또는 Serial.println(값)

시리얼 통신의 데이터를 출력한다. 시리얼 모니터로 확인이 가능하다.

매개 변수

값 - 출력할 값을 설정한다.

print()를 사용할 경우 한줄로(옆으로) 출력되며, println()을 사용할 경우 데이터를 출력할때마다 줄바꿈이 설정된다. (아래로 출력)

■ Serial.available()

시리얼 통신으로 수신된 읽지 않은 상태의 데이터의 수를 반환한다.

반환 값

수신된 읽지 않은 상태의 데이터 수를 반환한다. 없을 경우 0을 반환.

■ Serial.read()

수신 데이터 중 하나의 데이터를 반환한다. (저장된 순서대로)

반환 값

읽지 않은 상태의 데이터 중 가장 먼저 저장된 데이터를 반환한다.

■ Serial.end()

시리얼 통신을 끝내기 위한 함수이다. 사용하지 않아도 된다.

3.5 주파수 출력 관련 함수

■ tone(핀번호, 주파수) 또는 tone(핀번호, 주파수, 출력시간)

주파수 신호를 출력하여 소리를 설정하기 위한 함수이다.

매개 변수

핀번호 - 주파수 신호를 출력할 핀번호를 설정한다.

주파수 - 31~65535 범위의 주파수(Hz)를 설정한다.

출력시간 - 얼마동안 주파수를 출력할지 설정한다. 밀리초 단위

■ noTone()

주파수 신호 출력을 중지하기 위한 함수이다.

3.6 LCD 라이브러리(LiquidCrystal) 관련 함수

■ LiquidCrystal lcd(핀1, 핀2, 핀3, 핀4, 핀5, 핀6)

LCD 사용을 선언하고, 제어(통신)할 핀을 설정하기 위한 함수이다.

매개 변수

핀1~핀6 - LCD와 연결(통신)할 핀을 설정한다.

■ lcd.begin(가로,세로)

LCD사용의 시작을 알리고, 사용할 LCD의 크기를 설정하는 함수이다.

매개 변수

가로 - LCD의 가로 크기를 설정한다. 예: 16x2 LCD라면 16

세로 - LCD의 세로 크기를 설정한다. 예: 20x4 LCD라면 4

■ lcd.print(문자열)

LCD에 출력할 문자를 설정하는 함수이다.

매개 변수
문자열 - LCD화면에 출력할 문자를 설정한다.

■ lcd.setCursor(가로,세로)

LCD에 출력될 문자 출력 위치를 설정하기 위한 함수이다.

매개 변수
가로 - 가로 좌표를 설정한다. 16x2 LCD라면 0~15
세로 - 세로 좌표를설정한다. 16x2 LCD라면 0~1

■ lcd.clear()

LCD에 출력된 문자를 모두 지운다.

3.7 DHT11 라이브러리(DHT) 관련 함수

■ DHT dht(핀번호, 타입)

DHT를 연결할 핀과 DHT타입을 설정한다.

매개 변수
핀번호 - 데이터핀을 연결할 핀번호를 설정한다.
타입 - DHT타입을 설정한다. DHT11, DHT22 등

■ dht.readTemperature() 또는 dht.readHumidity()

DHT센서의 측정 값(온도, 습도)을 읽어들이는 함수이다.

반환 값
센서에 측정된 Temperature(온도) 및 Humidity(습도) 값을 반환한다.

3.8 서보모터(Servo) 라이브러리 관련 함수

■ Servo myservo;

서보모터를 사용, 선언하기 위한 함수이다.

■ myservo.attach(핀번호)

서보모터의 신호선을 연결할 핀을 설정하기 위한 함수이다.

매개 변수
핀번호 - 서보모터의 신호선을 연결할 핀을 설정한다.

■ myservo.write(값)

서보모터의 회전을 제어하기 위한 함수이다.

매개 변수
값 - 원하는 각도를 설정한다. (0~180)

3.9 스텝모터(Stepper) 라이브러리 관련 함수

■ Stepper stepper(모터 스텝수, 핀1, 핀2, 핀3, 핀4)

모터의 스텝 수를 설정하고 데이터 통신(제어)핀을 설정하기 위한 함수이다.

매개 변수
모터 스텝수 - 사용할 스텝 모터의 스텝 수를 설정한다. 핀1~핀4 - 스텝모터의 데이터선을 연결할 핀을 설정한다.

■ stepper.setSpeed(속도)

스텝모터의 속도를 설정하기 위한 함수이다.

매개 변수
속도 - 분당 회전수(RPM)값을 입력한다. 너무 빠를 경우 오류가 발생.

■ stepper.step(값)

스텝모터의 회전을 제어하기 위한 함수이다.

매개 변수
값 - 모터로 움직일 스텝 수를 설정한다. 마이너스(-) 값을 입력할 경우 반대로회전한다

3.10 흐름 제어문

프로그램을 구성하는 명령어들은 위에서 아래로 순차적으로 실행되며, 한 가지 일밖에 진행하지 못한다. 그렇기 때문에 프로그램을 효율적으로 구성하려면 프로그램의 흐름을 제어하는 제어문(반복문, 조건문 등)을 사용해야한다.

제어문은 특정 조건에 해당될 때 명령어를 실행한다던지(조건문), 비슷한 명령어를 반복적으로 사용해야할 때(반복문) 혹은 그외의 여러 경우에 명령의 흐름을 바꾸기 위해 사용할 수 있다.

if문 (조건문)

if문을 사용하면 '만약 ~하면 ~ 한다.' 와 같은 명령을 만들어줄 수 있다. 여기에 else문 혹은 else if문를 추가하면 코드의 흐름을 더 부드럽게 만들 수 있다.

```
if (a<300) {                    // 만약 A가 300보다 작다면
  digitalWrite(6, HIGH);        // 디지털 6번핀에 HIGH 신호를 출력한다.
}
else if (a>300) {               // 그렇지 않고 만약 a가 300보다 크다면
  digitalWrite(7, HIGH);        // 디지털 7번핀에 HIGH 신호를 출력한다.
}
else {                          // 그렇지 않다면 (모든조건이 만족하지 않으면)
  digitalWrite(8, HIGH);        // 디지털 8번핀에 HIGH 신호를 출력한다.
}
```

for문 (반복문)

for문을 사용하면 반복되는 비슷한 명령어를 간단한 표현으로 실행시킬 수 있다. for문은 특정조건을 벗어날 때까지 실행된다.

```
for(int i=0;i<5;i+ 1) {              // for (초기식; 조건식; 증감식)
        // i를 0으로 초기화하며,  i는 5보다 작고, i는 1씩 커진다.
  digitalWrite(6, HIGH);             // 디지털 6번핀에 HIGH 신호를 출력한다.
  delay(i);                          // i가 대입되며 반복된다.
}
```

while문 (반복문)

for문과 같은 반복문이지만, 성격이 조금 다르다. while문은 조건을 만족하는 동안 명령을 반복한다. 즉 '~인 동안 ~을 반복한다'가 성립된다.

```
while(i<100) {               // i가 100보다 작은 경우 반복한다.
  sum=sum+ i;                // sum은 sum+ 1이다.
  i=i+ 1;                    // i는 1씩 커진다.
}
```

switch/case문 (조건 선택문)

1일땐 ~하고, 2일땐 ~하고, 3일땐 ~하고.. 와 같은 명령어가 성립된다.

```
if (i >5) {
  i=i+ 1;                     //  i는 1, 2, 3, 4다.
}
switch (i) {                  // 조건 i (1, 2, 3, 4)
  case 1 : a=100; break;      // 1이 성립될 경우 a는 100이다.
  case 2 : a=200; break;      // 2가 성립될 경우 a는 200이다.
  case 3 : a=300; break;      // 3이 성립될 경우 a는 300이다.
  case 4 : a=400; break;      // 4가 성립될 경우 a는 400이다.
}
```