아두이노 시뮬레이션 프로그래밍

v1.0

김영준 저

공학박사, 목원대학교 겸임교수

前 Microsoft 수석연구원

헬로앱스

http://www.helloapps.co.kr

02 LED 점멸하기

학습 목표

- 디지털 쓰기 명령어(DigitalWrite)의 활용 방법을 이해하고 응용할 수 있다.

실습 개요

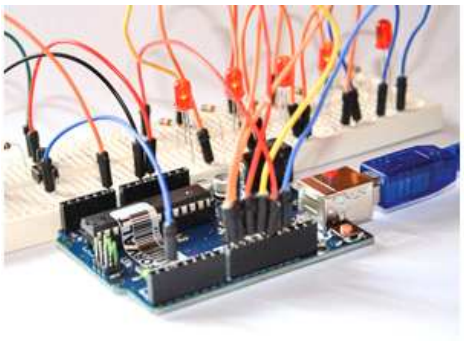
- 아두이노의 디지털 쓰기 (DigitalWrite) 명령어로 LED를 제어한다.
- 기다리기 함수를 이용하여 LED의 점멸 간격을 조절해 본다.
- setup()과 loop() 함수의 차이점을 이해할 수 있다.
- 일정한 시간 후에 자동으로 꺼지는 장치를 구현해 본다.

2.1 아두이노 쉴드 및 핀 정보

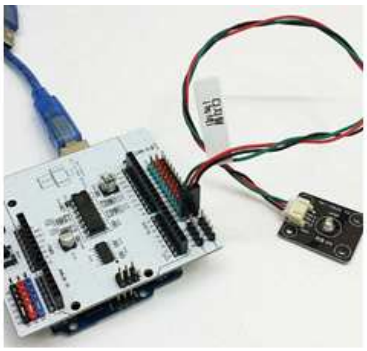
아두이노 쉴드

- 아두이노 보드 위에 적층식으로 쌓아 나가는 전자회로 모듈을 쉴드(Shield)라고 부른다. 보통 기능이 복잡하거나 연결되는 핀 수가 많은 모듈을 별도로 쉴드로 개발하는 경우가 많다.

- 브레드보드가 필요없는 올인원 쉴드 (All-In-One Shield)는 센서 연결시 브레드보드 없이 3핀 또는 4핀으로 되어 있는 모듈을 바로 아두이노에 연결할 수 있도록 도와주는 보드로서, 아두이노 작품 개발시 시간을 단축시켜 주고 아두이노의 편리성을 최대한 활용하는 보드이다.



브레드 보드를 사용한 모듈 연결

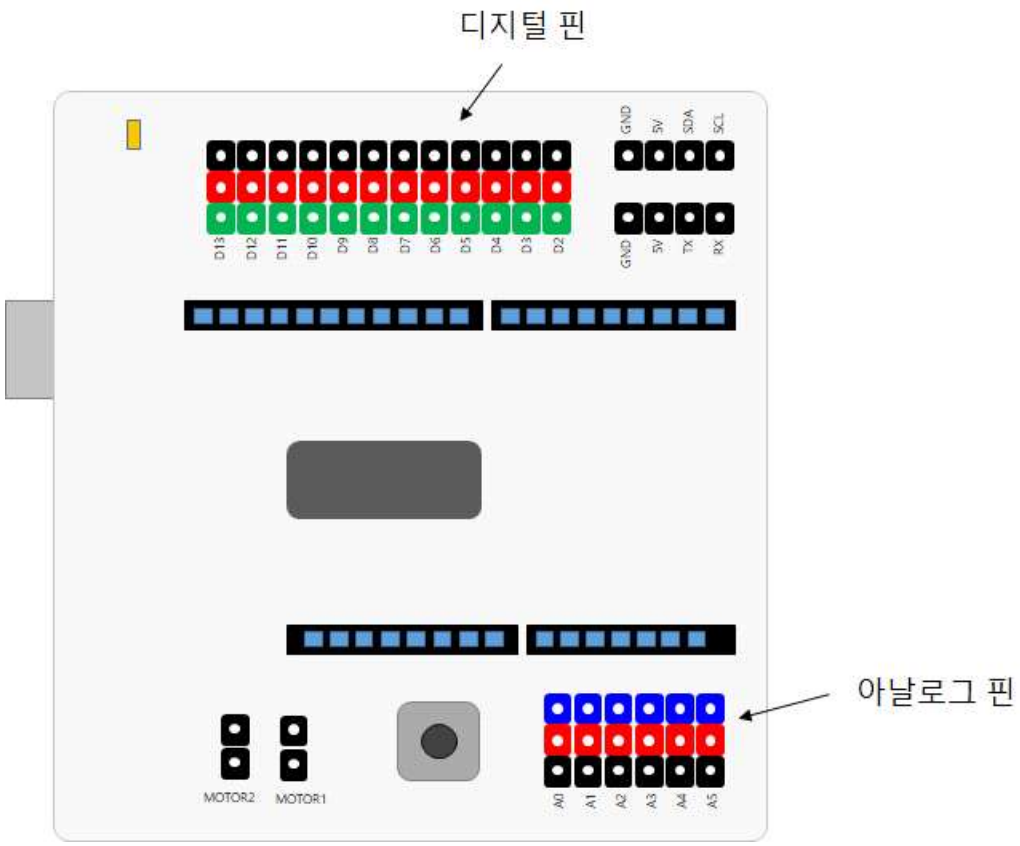


올인원 쉴드를 사용한 모듈 연결

아두이노 핀 정보

- 아두이노는 디지털 핀 14개와 아날로그 핀 6개를 가지고 있다.
- 디지털 핀과 아날로그 핀에는 핀번호가 0번부터 부여된다.

디지털 핀번호 (14개)	아날로그 핀번호 (6개)
D0 ~ D13	A0 ~ A5

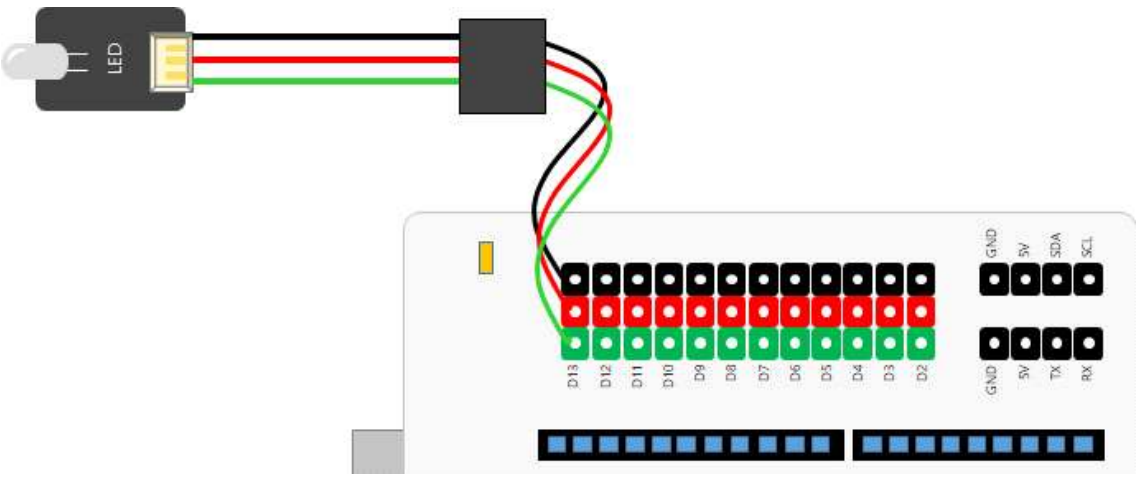


디지털 핀

- 디지털 핀들은 GND (검정), 5V (빨강), Data 선 (초록) 등 3개의 핀으로 구성되어 있으며, 부품을 연결할 때 방향을 맞추어 연결해 주어야 한다.

- 디지털 부품 연결시 케이블의 색상 또는 각 케이블이 극성을 확인하여 연결한다.

- 디지털 0번 (Rx)과 1번(Tx)은 시리얼 통신에 사용되기 때문에 실제로 D2부터 부품을 연결할 수 있다.

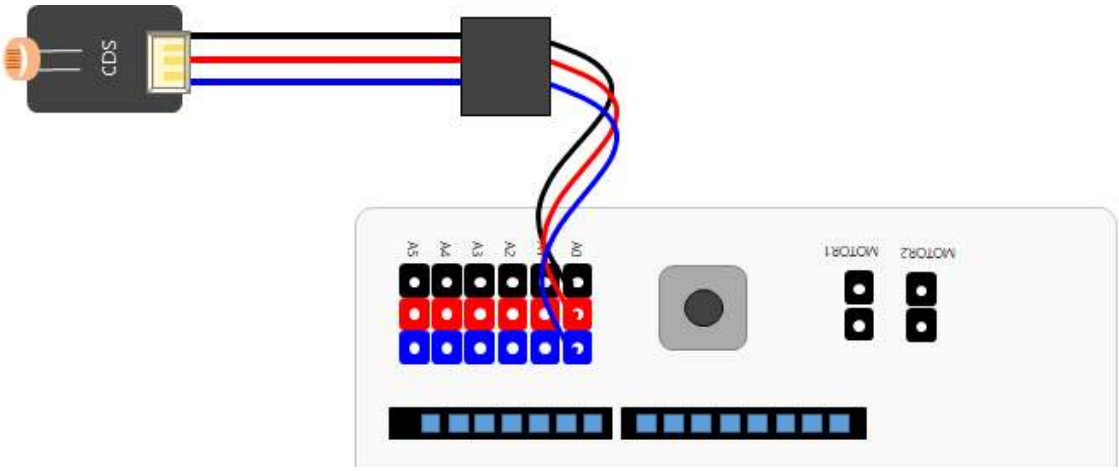


핀정보	케이블 색상
GND	검정
5V	빨강
Data	초록

아두이노 아날로그 핀

- 아날로그 핀들은 GND (검정), 5V (빨강), Data 선 (파랑) 등 3개의 핀으로 구성되어 있으며, 부품을 연결할 때 방향을 맞추어 연결해 주어야 한다.

- 아날로그 4번(A4)과 아날로그 5번(A5)은 I2C와 같은 다른 용도로도 사용되기 때문에 아날로그 부품을 연결할 때에는 A0부터 연결한다.

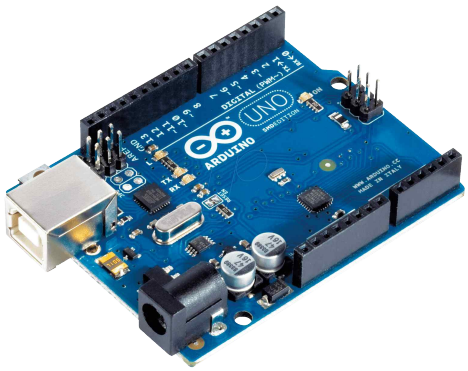


핀정보	케이블 색상
GND	검정
5V	빨강
Data	파랑

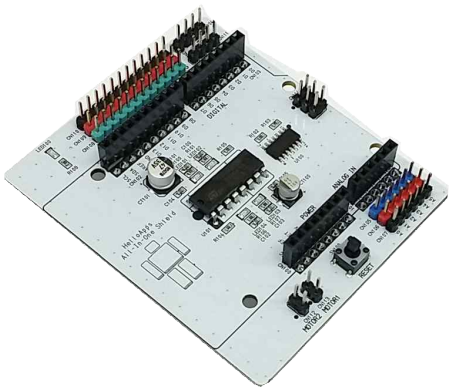
2.2 준비하기

준비물

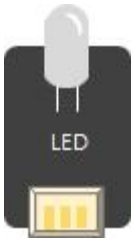
- 아래 준비물은 실제 아두이노 키트가 있는 경우를 예로 든 것이며, 아래 하드웨어 구성과 동일한 내용을 시뮬레이션으로 진행하게 된다.
- 아두이노 보드, 올인원 쉴드, 디지털 LED 모듈 3개



아두이노 우노보드



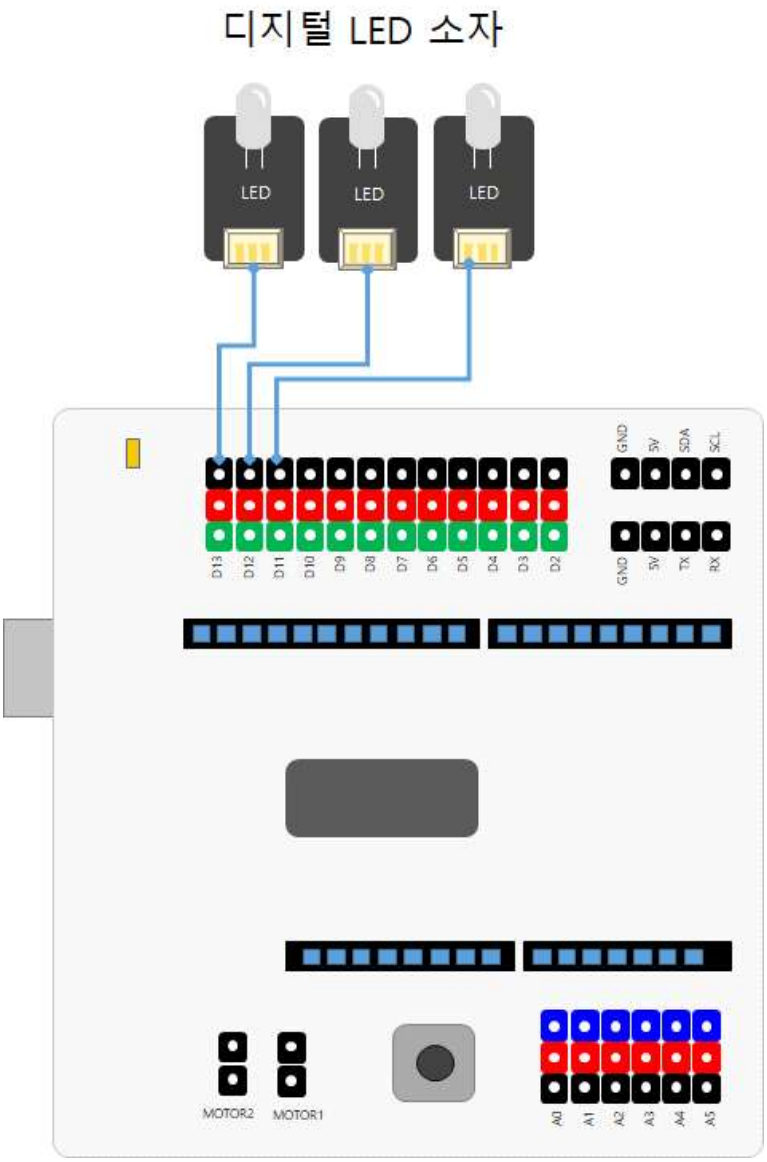
올인원 쉴드



디지털 LED 소자 3개

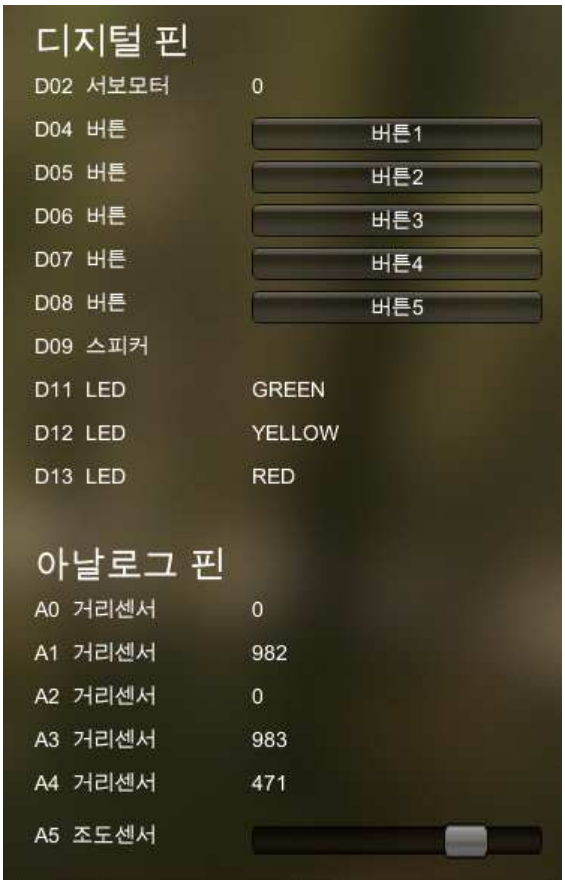
H/W 연결하기

■ 디지털 LED 소자를 디지털 11번, 12번, 13번에 각각 연결한다.



시뮬레이션 상에서의 연결 정보

- 시뮬레이션 상에서는 디지털 LED 소자가 각각 디지털 11번, 12번, 13번에 연결되어 있다.



- 디지털 핀에 연결된 부품
  - 디지털 2번: 서보 모터
  - 디지털 4번 ~ 8번: 버튼 센서
  - 디지털 9번: 스피커
  - 디지털 11번: 초록색 LED
  - 디지털 12번: 노란색 LED
  - 디지털 13번: 빨간색 LED

2.3 디지털 쓰기 명령어 (DigitalWrite)

디지털 쓰기 명령어

- LED는 디지털 소자로서 0과 1 또는 HIGH와 LOW 등 2가지 상태 중에 한 가지 상태의 값을 가진다.

디지털 소자 값의 범위
0 또는 LOW : 꺼짐 1 또는 HIGH : 켜짐

- **디지털 쓰기 명령어 (DigitalWrite)**

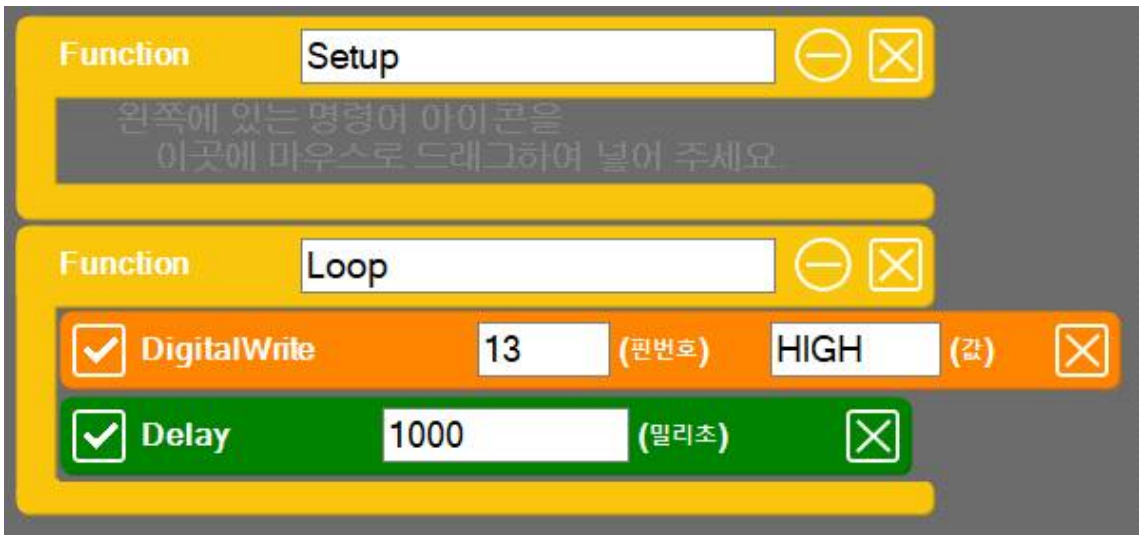
  디지털 소자에 값을 전달하기 위한 명령어는 다음과 같이 DigitalWrite 명령어를 사용한다. 핀번호와 상태값을 인수로 전달한다.

디지털 소자에 값을 쓰는 명령어
DigitalWrite(13, LOW) : 디지털 13번 핀의 값을 끔 DigitalWrite(13, HIGH) : 디지털 13번 핀의 값을 켬

2.4 LED 켜기

LED 소자 켜기

■ 디지털 13번에 연결된 LED 소자를 켜기 위해서는 아래와 같이 DigitalWrite 명령어를 사용하고, 핀번호 13, 그리고 값을 HIGH로 설정해 준다.

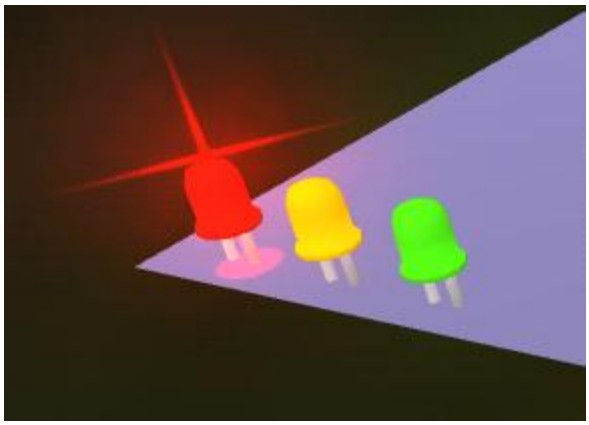


SPL 스크립트	스케치 코드

void setup()
{
 	
}
 	
void loop()
{
  //디지털 13번 핀에 연결된 LED를 켠다.
  DigitalWrite(13, HIGH)
 	
  //1초간 기다린다.
  Delay(1000)
}
	void setup()
	{
	  //디지털 13번 핀을 쓰기 모드로 지정
	  pinMode(13, OUTPUT);
	}
	
	void loop()
	{
	  //디지털 13번 핀에 연결된 LED를 켠다.
	  digitalWrite(13, HIGH);
	
	  //1초간 기다린다.
	  delay(1000);
	}

실행하기

▶ 실행 버튼을 클릭하여 13번 LED가 계속 켜져 있는 지 확인해 본다.



실습

▶ 켜진 LED를 끄는 기능으로 프로그램을 수정해 보기 바란다.

※ 참고자료) 상수와 변수

- 프로그램에서 상수는 변경할 수 없는 값을 저장해 놓은 저장소 이며, 이와 반대로 변수는 값을 변경할 수 있는 저장소이다.

  - 상수: 값을 변경할 수 없다.

    ```
    const int a = 10;
    ```

  - 변수: 값을 변경할 수 있다.

    ```
    int a = 10;
    ```

- 상수는 자료형 앞에 const 라는 단어가 더 추가된다.

- 위의 예에서 사용된 HIGH나 LOW와 같은 단어는 프로그램의 내부에서 상수로 정의된 이름이다. 즉 다음과 같이 내부에 정의되어 있다.

  ```
  const int HIGH = 1;
  const int LOW = 0;
  ```

- DigitalWrite 명령어에서 상수 대신에 다음과 같이 직접 값을 사용해도 된다.

  ```
  DigitalWrite(13, 1)
  DigitalWrite(13, 0)
  ```

※ 참고자료) 자료형

- C언어에서는 변수를 정의할 때 변수 이름 앞에 자료형을 표기해 주어야 한다. 저장하고자 하는 값이 어떠한 종류인지를 설명해 주는 것이다.

- 변수의 자료형에는 다음과 같은 종류들이 있다.

자료형	설명
int	• 정수형(Integer) • 0, -50, 300 등의 정수형 숫자를 저장한다.
float	• 실수형(float) • 0.5, -50.5, 303.345 등의 실수형 숫자를 저장한다.
byte	• 바이트형(byte) • 0 ~ 255 사이의 숫자를 저장한다.
char	• 문자형(char) • 알파벳 문자나 기호를 저장한다.
String	• 문자열형(String) • 여러 문자 데이터를 하나로 이어서 저장한다.
boolean	• 논리형(Boolean) • true 또는 false와 같이 논리형 값을 저장한다.

- int a = 10; 이라는 의미는 정수형 값을 저장하는 변수 a에 10을 저장하라는 의미이다.

실습

▶ 앞에서 소개된 코드 중에서 상수로 정의된 부분을 찾아 본다.

▶ 아래 제시된 각각의 자료형에 대해 사례가 되는 코드를 작성해 본다.

자료형	사례
int	int a = 255;
float	float a = 123.56;
byte	byte a = 25;
char	char c = 'A';
String	String s = "Hello";
boolean	boolean b = true;

※ 참고자료) C언어는 명령어 뒤에 ';'로 끝난다.

■ 앞에서 제시된 예제에서 스크립트와 C언어를 비교해 봄으로써 C언어 사용시 어떠한 특징들이 있는 지 살펴 보자.

■ 다음은 SPL 스크립트와 C언어의 차이점이다.

SPL 명령어	스케치 코드 (C언어)
DigitalWrite(13, HIGH) Delay(1000)	digitalWrite(13, HIGH); delay(1000);

■ C언어 문법에서는 각 명령어가 ';'로 끝나는 것에 유의하기 바란다.

※ 참고자료) 아두이노 명령어가 소문자로 시작하는 이유

- 모든 스크립트 코딩에서 변수나 함수 등의 이름을 부여할 때 크게 4가지 표기법 중에 한가지를 따른다.

- 다음은 4가지 표기법의 차이점을 설명한 것이다.

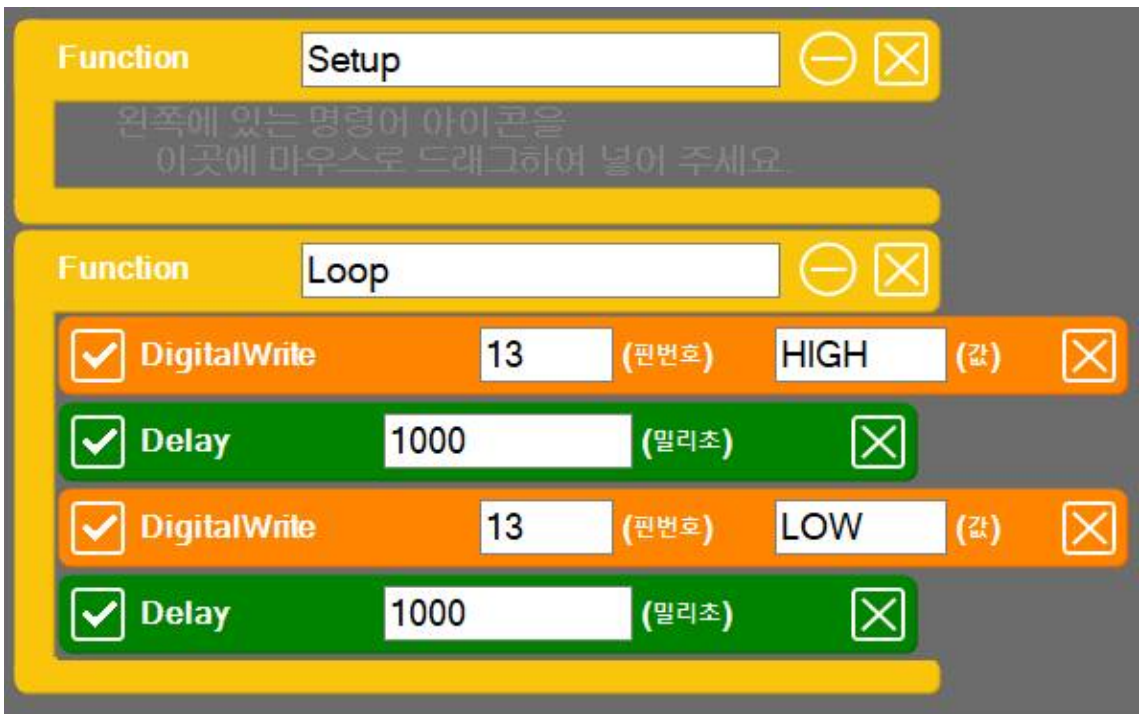
변수 및 함수 표기법	설명
헝가리안 표기법	변수나 함수 이름 시작을 자료형을 나타내는 알파벳을 붙인다. 예) int iNum = 10;
파스칼 표기법	변수나 함수 이름이 대문자로 시작한다. 파스칼 언어나 C# 언어, SPL 등이 이 표기법을 따른다. 예) DigitalWrite(13, HIGH)
카멜 표기법	시작은 소문자로 하고 중간에 있는 단어는 대문자로 시작한다. Java나 아두이노 함수 등이 이에 해당한다. 예) digtalWrite(13, HIGH)
언더바 표기법	변수 단어 사이에 '_'를 붙여서 단어를 구분한다. 예) digital_write(13, HIGH)

- 변수나 함수 이름이 표기법은 정답이 있는 것은 아니고 개발자의 취향에 따라서 달라지며, 자신이 이해하기 쉬운 표기법을 정해서 프로그램 전체적으로 일관되게 적용하기만 하면 된다.

2.5 LED 점멸시키기

LED 소자 점멸시키기

■ 디지털 LED 소자를 점멸시키기 위한 코드는 아래와 같다. 프로그램을 작성한 후 LED 점멸 상태를 확인해 보기 바란다.



SPL 스크립트	스케치 코드

void setup()
{
 	
}
 	
void loop()
{
//디지털 13번 핀에 연결된 LED를 켠다.
DigitalWrite(13, HIGH)
 	
//1초간 기다린다.
Delay(1000)
 	
//디지털 13번 핀에 연결된 LED를 끈다.
DigitalWrite(13, LOW)
 	
//1초간 기다린다.
Delay(1000)
}
	void setup()
	{
	//디지털 13번 핀을 쓰기 모드로 지정
	pinMode(13, OUTPUT);
	}
	
	void loop()
	{
	//디지털 13번 핀에 연결된 LED를 켠다.
	digitalWrite(13, HIGH);
	
	//1초간 기다린다.
	delay(1000);
	
	//디지털 13번 핀에 연결된 LED를 끈다.
	digitalWrite(13, LOW);
	
	//1초간 기다린다.
	delay(1000);
	}

실습

▶ LED의 점멸 주기를 각각 500밀리초, 100밀리초, 50밀리초, 20밀리초 등으로 수정하여 점멸되는 결과를 확인해 본다.

2.6 순차적으로 켜지는 신호등 만들기

신호등 만들기

■ 1초 간격으로 초록색 -> 노란색 -> 빨간색의 순서로 LED 가 켜지도록 기능을 작성해 본다.

※ 참고자료) 블록 복사하기

■ 블록을 복사할 때에는 Shift 키를 누른 상태에서 복사하고자 하는 블록을 마우스로 이동해 주기만 하면 된다.

실습

▶ LED의 점멸 주기를 각각 500밀리초, 100밀리초, 50밀리초, 20밀리초 등으로 수정하여 점멸되는 결과를 확인해 본다.

2.7 setup() 함수와 loop() 함수

setup과 loop

- 아두이노 기본 프로그램은 다음과 같이 setup() 함수와 loop() 기본 함수가 자동으로 만들어진 상태에서 시작한다.


```
void setup()
{
}

void loop()
{
}
```

- 아두이노 프로그램을 실행하면 내부적으로 다음과 같이 main() 함수가 만들어 진다.

```
void setup();
void loop();

int main()
{
    setup();
    while(true)
    {
        loop();
```

```
    }

    return 0;
}

void setup()
{
}

void loop()
{
}
```

- 위의 main 함수를 살펴보면 setup() 함수의 내용이 먼저 실행되고, 그 다음으로는 loop() 함수만 계속 실행된다.

- setup() 함수는 맨 처음 한번 만 실행된다.

- loop() 함수는 setup() 함수가 실행된 이후에 실행되며, 아두이노에 전원에 들어와 있는 동안에는 무한히 반복하여 실행된다.

2.8 일정한 시간 후에 자동으로 꺼지는 LED 등 만들기

설계하기

- 일정한 시간 동안 LED가 켜진 후, 그 이후에는 LED가 계속 꺼져 있게 하려고 한다. LED를 켜고 끄는 명령어를 setup() 함수와 loop() 함수 중 어느 곳에 입력해야 이 기능을 구현할 수 있을지 생각해 본다.

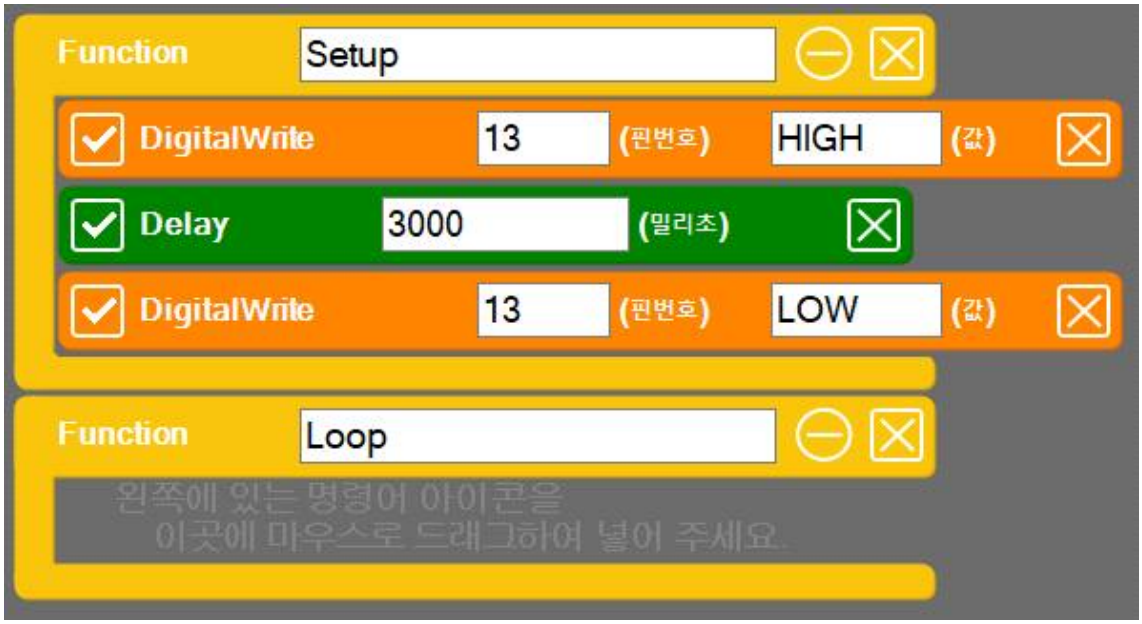
구현하기

- 아래의 코드를 일정한 시간 후에 자동으로 꺼지는 기능이다. 아래의 코드들을 setup() 함수와 loop() 함수 중 어느 곳에 추가해야 할지 생각해 보고 기능을 완성해 본다.

SPL 스크립트	스케치 코드

DigitalWrite(13, HIGH)
Delay(3000)
DigitalWrite(13, LOW)
	digitalWrite(13, HIGH);
	delay(3000);
	digitalWrite(13, LOW);

■ 아래의 코드를 실행한 후 결과를 확인해 본다.

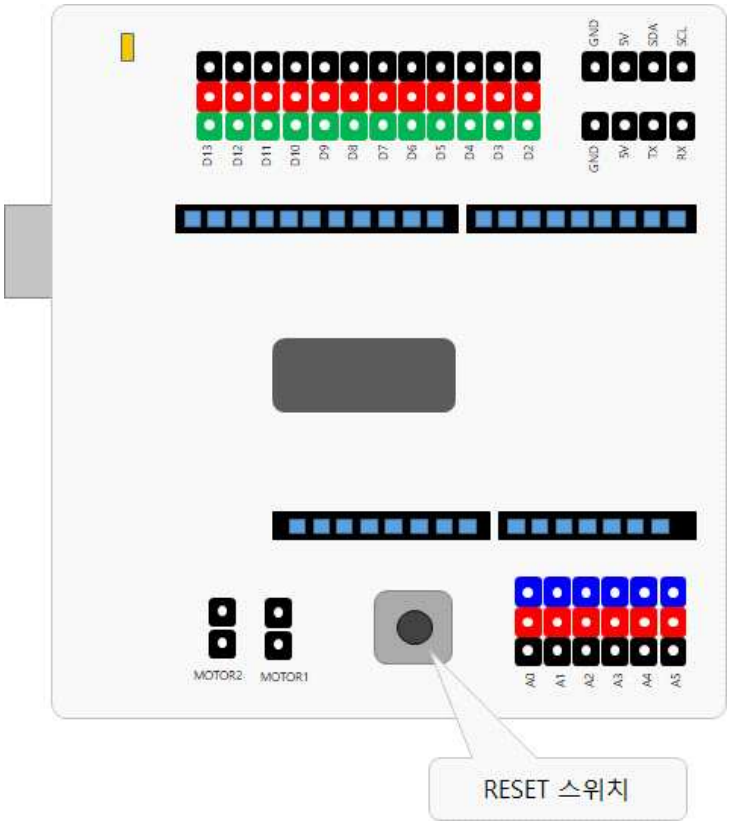



```
void setup()
{
  digitalWrite(13, HIGH);
  delay(3000);
  digitalWrite(13, LOW);
}

void loop()
{

}
```

■ setup() 함수에 코드를 추가하게 되면 한번만 실행되게 된다. setup() 함수에 있는 명령어들을 다시 실행시키려면 아두이노 보드에 업로드된 프로그램을 처음부터 다시 실행시켜야 주어야 하는데, 이렇게 아두이노 프로그램을 처음부터 다시 실행하기 위해서는 아두이노 보드를 리셋시켜 주면 된다. 아래 그림과 같이 아날로그 핀 옆에 있는 리셋 스위치를 눌러서 아두이노 프로그램이 처음부터 다시 실행되는 지 확인해 보기 바란다.



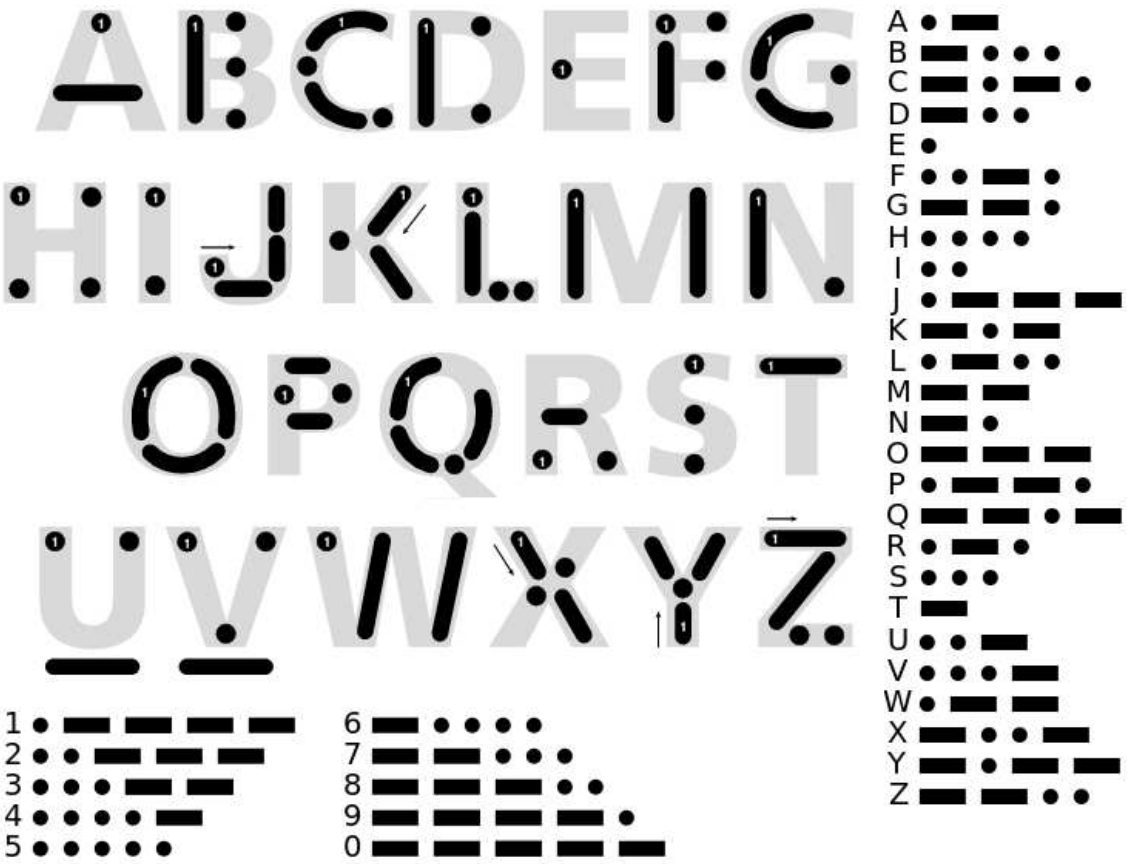
실습

▶ 앞의 활동에서 작성한 코드 중에서 직접 값을 사용하지 않고 상수로 선언하여 사용할 수 있는 부분이 어느 부분이 있는 지 생각해 보고 상수를 정의하여 프로그램을 수정해 본다.

2.9 LED로 모르스 부호 만들기

모르스 부호

- 모르스 부호는 도트(점)와 대시(선)의 조합으로 구성된 메시지 전달용 부호로, 모르스(미국)에 의해 고안된 것이다. 이것을 약간 수정한 것이 국제 모르스 부호로 널리 사용되고 있다. A에서 Z까지의 문자, 0에서 9까지의 숫자, 이에 구두점, 그 밖의 것이 부호화되어 무선전선에 사용되고 있다.



실습

▶ 모르스 부호로 SOS 글자를 LED를 이용해 전송해 본다.