Arduino Basic Guide

ARDUINO

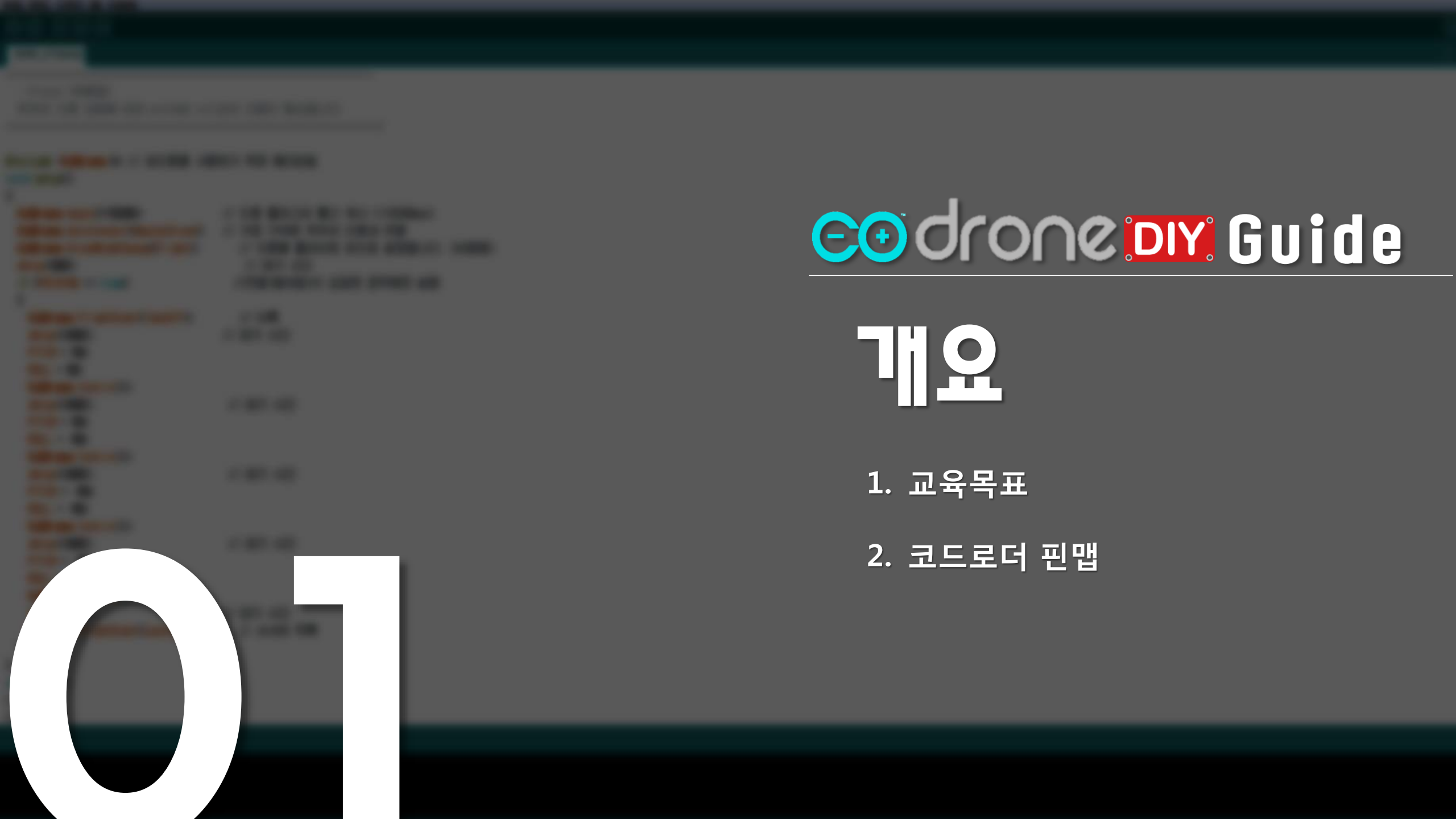
×

codrone DIY

ROBOLINK

INDEX

코드론DIY를 활용한 아두이노 기본 교육 가이드



edrone DIY Guide

개요

1. 교육목표
2. 코드로더 핀맵

01

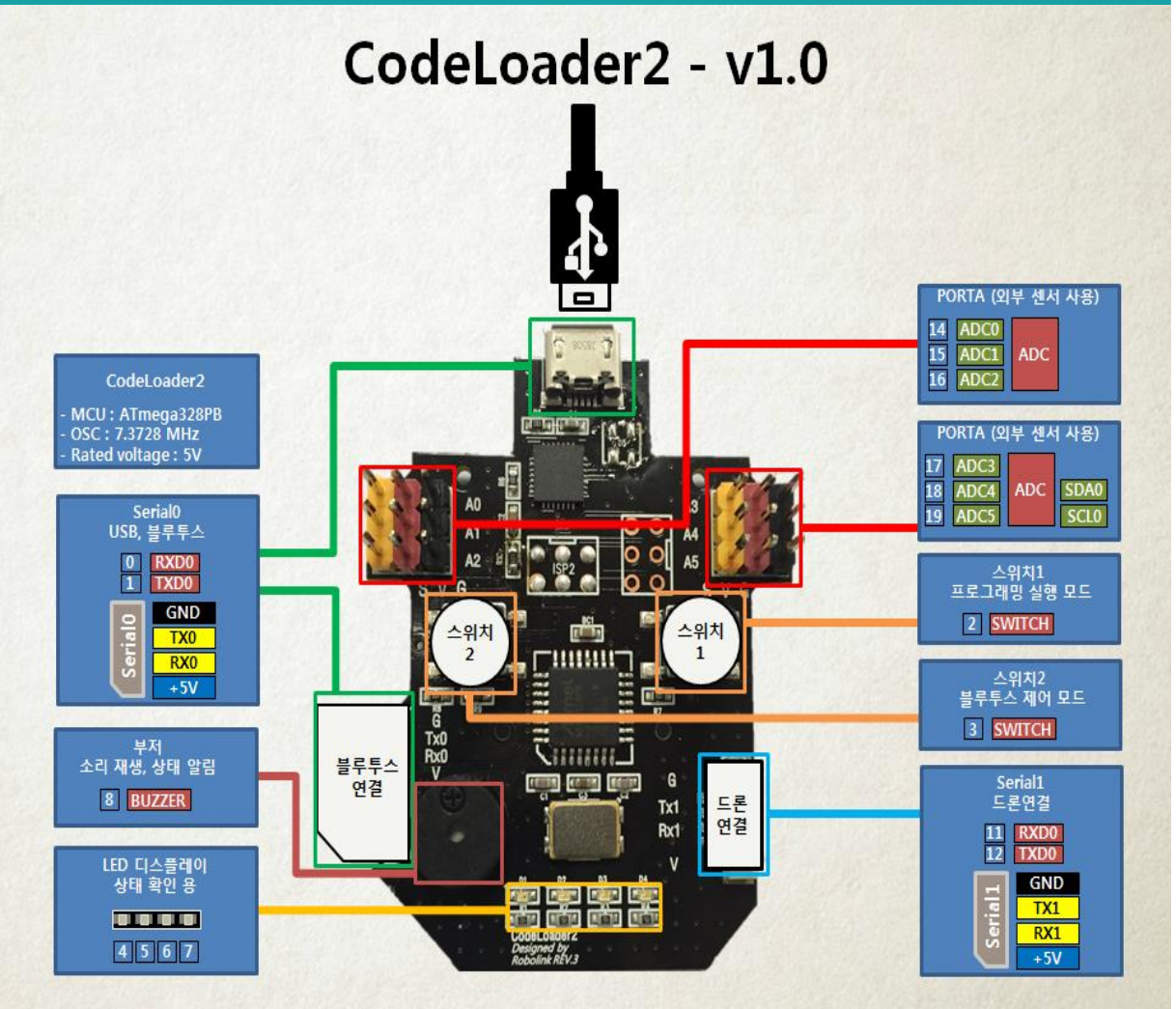
코드로더 기본 아두이노 교육 목적

01 다양한 실습 예제로 아두이노에 쉽게 접근이 가능 하도록 지원

02 텍스트 코딩의 기본적인 구조와 문법을 습득

03 코드론을 아두이노로 제어하기 기본 지식을 습득

코드로더 핀맵



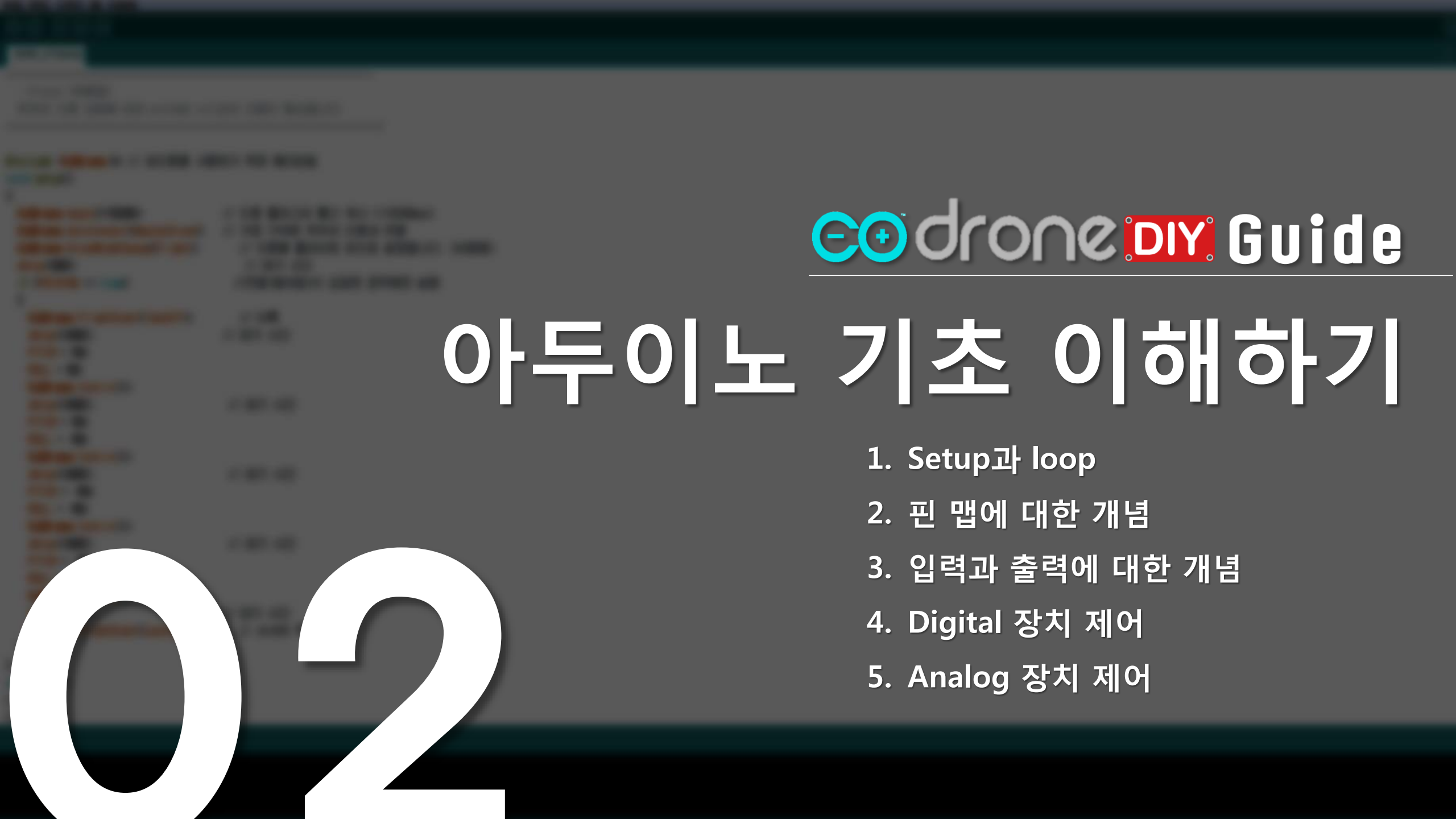
코드로더는 코드론에서 아두이노로 프로그래밍을 할 수 있도록 만들어진 보드입니다.

※ 유저가 보드에 접근할 수 있도록 만들어진 자료입니다.

- MCU : ATmega328PB
- OSC : 7.3728 MHz
- 사용 전압 : 5V

- 내장 LED
- 내장 부저
- 시리얼0 통신 - UART0 (USB, 블루투스 연결)
- 시리얼1 통신 - UART1 (드론 연결)

- 외부 포트 핀 ADC 기능 (ADC0 ~ 5)
- 외부 포트 핀 I²C 기능 (SDA0, SCL0)



edrone DIY Guide

아두이노 기초 이해하기

1. Setup과 loop
2. 핀 맵에 대한 개념
3. 입력과 출력에 대한 개념
4. Digital 장치 제어
5. Analog 장치 제어

02

01 Setup 과 loop

아두이노 프로그래밍을 위해서는 Setup과 Loop라 부르는 함수의 내용을 입력해야 합니다. 가장 기본적인 아두이노 프로그램의 코드는 다음과 같습니다.

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

setup과 loop라 부르는 함수의 내용

Setup에 입력되는 내용

Setup함수는 프로그램 시작 시 최초 한번 호출됩니다.
따라서, 이곳에는 초기화나 프로그램 동작에 필요한
준비 동작을 넣어야 합니다.

Loop에 입력되는 내용

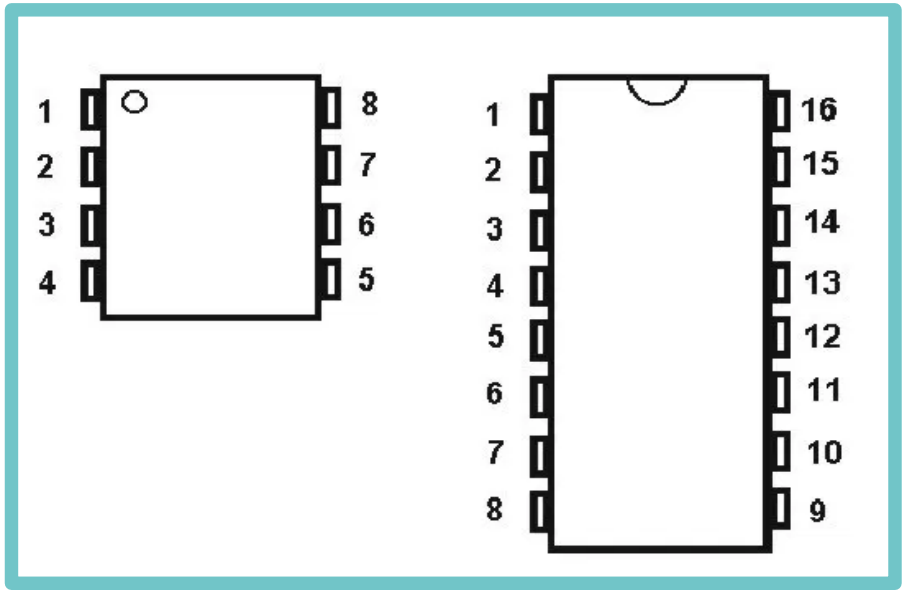
loop함수는 setup이 호출된 이후 프로그램 동작 중에 계속 **반복 호출**됩니다.
반복되는 주기는 정확히 알 수 없고 가능한 자주 호출되기 때문에 시간적으로 정확하게 작동해야 할 필요성이 있을 때에는 Time과 관련된 함수들을 이용해서 흐름 제어를 해야 합니다.

※ 만약, setup과 loop 중 하나라도 코드에 없다면 아두이노 IDE는 에러를 내기 때문에 반드시 입력해야합니다.

02 핀맵에 대한 개념

핀(Pin)이란, IC 끼리 연결할 수 있도록 나와 있는

하드웨어 인터페이스를 의미합니다.



IC에서 Pin의 모습

하드웨어는 **핀과 핀을 연결(Wiring)하는 방식**으로 제어됩니다.

IC에서의 Pin을 물리적 **핀(Physical Pin)**이라 부르며 각 핀에는
고유한 번호를 가지고 있습니다.
핀 맵은 이와 같은 핀을 번호순으로 나열한 것을 의미합니다.
아두이노에서는 제어기에 연결된 다른 하드웨어 기기를 프로그램 상에서 호출하기 위해 이 핀의 개념을 이용합니다.
아두이노에서는 물리적 핀 번호를 사용하지 않고 별도로 정의된 아두이노 핀 맵을 사용합니다.
따라서, 프로그래머는 이 아두이노 핀 맵을 알고 있어야 합니다.

03 입력과 출력에 대한 개념

구분	개념이해	실제활용
입력	Verify 어떤 정보를 확인하는 행위 Read 어떤 정보를 읽는 행위 Receive 어떤 정보를 전달받는 행위	- 센서의 값을 확인하는 것 - 외부 장치의 상태를 확인하는 것 - 메모리에 저장된 데이터를 확인하는 것 - 두 장치가 통신할 때 상대 장치로 부터 정보를 받는 것
출력	Action 어떤 동작을 시키는 행위 Write 어떤 정보를 기술하는 행위 Send 어떤 정보를 전달하는 행위	- 디스플레이 장치 제어하는 것 - 구동 장치(모터 등) 제어하는 것 - 메모리에 데이터를 저장하는 것 - 두 장치가 통신할 때 상대 장치에 정보를 전달하는 것

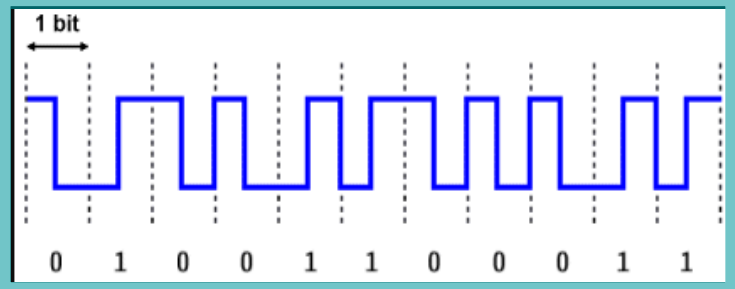
04 Digitlal 장치 제어

1. Digital

두 가지 상태로 정보를 전달하는 방식을 의미한다.

두 가지 상태의 예는 아래의 그림과 같습니다.

이 개념은 **TTL(Transistor-Transistor Logic)**에서 출발한 것으로 5V를 기준으로 0V와5V로 상태를 나누는 것에서 출발하여 이로 인해 Digital 신호가 만들어집니다.



Digital 신호

1. 0과 1
2. 참과 거짓(True/False)
3. On과 Off
4. HIGH와 LOW

정보전달 상태예시

2. Digital 장치의 예

Digital 장치는 두 가지 상태로 제어될 수 있는 것으로 대표적인 것은 오른쪽 설명과 같습니다

1. Output : LED 장치(On/Off로 상태 제어)

2. Input : Button 장치(눌림/안 눌림으로 상태 체크)

3. Digital Read/Wirte

Digital 방식으로 신호를 주고받는 Pin이 따로 있습니다.

Digital Pin을 통해 출력을 하려면 digitalWrite()함수를 사용하고 입력을 하려면 digitalRead() 함수를 사용해야 합니다.

```
int digitalRead(int pin);
void digitalWrite(int pin, int value);
```

여기서 Pin은 아두이노 핀 맵의 핀 번호를 뜻하며 value는 HIGH/LOW로 사용됩니다.

Digital Read/Write 함수

4. Pin 모드 설정

Digital Pin은 일반적으로 입력과 출력을 겸할 수 있습니다. 따라서 입력으로 사용할 지 출력으로 사용할 지 미리 설정해야 하는데 이것을 Pin 모드라 부릅니다.

아두이노에서는 이 Pin 모드를 pinMode함수를 이용해서 설정할 수 있습니다

```
void pinMode(int pin, int mode);
```

Digital Read 함수

여기서 Pin은 설정할 아두이노 핀 맵의 핀 번호를 뜻 하는 모드는 **INPUT, OUTPUT, INPUT_PULLUP** 중 선택할 수 있습니다.

(**INPUT과 INPUT_PULLUP**은 모두 **pinMode**를 Input용도로 설정해줍니다.

연결되는**Digital Input**장치의 회로에 따라 사용이 달라지게 되며 확인이 필요합니다)

Pin 모드 를 INPUT, INPUT_PULLUP 설정	**Digital Read() 를 사용**
Pin 모드 를 OUTPUT설정	**DigitalWrite() 를 사용**

05 Analog 장치 제어

1. Analog

Digital은 2가지 상태(0V와 5V)만 존재하지만,

Analog는 0V에서 5V까지 사이의 상태가 모두 존재합니다.

0V에서 5V까지 몇 가지 상태로 나누었나가 바로 **해상도(Resolution)**인데,

아두이노에서는 **8bit(256)과 10bit(1024) 해상도로 Analog**를 표현합니다.

2. Analog Input

Digital Pin과 달리 **Analog Pin**은 Input과 Output을 겸용으로 사용할 수 없고 Input과 Output 전용 Pin으로 존재합니다. Analog Input은 ADC(Analog Digital Converter)라 부르는 하드웨어 장치를 통해 사용할 수 있습니다.

Analog Input을 사용하기 위해서는 analogRead함수를 사용합니다.

```
int analogRead(int pin);
```

analogRead함수

Pin은 **Analog Input으로 사용할 아두이노 핀 맵의 핀 번호**를 의미하며 **ADC**란 이름으로 고정되어 있습니다. ADC 핀 들은 A0, A1, A2… 등의 이름으로도 사용될 수 있습니다. analogRead의 결과 값은 10bit 해상도로 반환되기에 0~1023까지의 수로 표현됩니다. 0이 0V이고 1023이 5V가 되기에 1은 약 0.00097V(1/1024)의 값을 의미합니다. 코드로더2에서는 A0, A1, A2, A3, A4, A5의 6개 ADC 핀을 유저가 사용할 수 있습니다.

3. Analog Output

Analog Input은 ADC란 장치에 의해 사용할 수 있는 반면,

Analog Output은 PWM이란 장치를 이용해 사용할 수 있습니다.

PWM은 Pulse Width Modulation의 약자로 **Pulse의 폭(Width)을 조절하여 정보를 전달하는 방식**을 의미합니다. PWM을 이용하면 평균 전압을 조종할 수 있어서 Analog Output 제어 시 많이 사용합니다.

아두이노에서 Analog Output은 analogWrite함수를 이용해서 사용할 수 있습니다.

```
void analogWrite(int pin, int value);
```

analogWrite 함수

여기서 pin은 아두이노 팬 맵에서 PWM 사용이 가능한 핀 번호를 의미하며 value는 PWM의 Level을 말합니다.

Analog Output은 8bit 해상도로 제어되기에 0~255값을 가지고 0은 0V를 의미하고 255는 5V를 말합니다. 1은 약 0.0039V(1/256) 값을 표현합니다.

코드로더2에서 사용 가능한 PWM 핀은 5번으로 LED에 할당되어 있습니다.

4. Analog 제어가 필요한 경우

1. **센서의 값을 Analog로 받아 정도를 알 수 있습니다.**
2. **LED등의 밝기를 조절할 수 있습니다.**
3. **DC 모터를 구동할 때 속도를 조절할 수 있습니다.**

edrone DIY Guide

아두이노 튜토리얼

1. LED
2. 시리얼 통신
3. 버튼
4. 부저

03

STEP 1 LED

01 LED를 제어합니다.

02 LED를 켜거나 끄면서 아두이노 사용법에 익숙해지도록 합니다.

03 순차 프로그램의 개념을 배웁니다.

1. Blink
(LED 1개를 깜박이기)

- 목표 digitalWrite 사용방법 익히기
- 함수 digitalWrite (핀번호, 출력 선택)
- 설명 이 예제는 코드로더 보드에서 원하는 1개의 LED 핀이 깜박입니다

```
void setup()
{
  // 사용할 수 있는 LED 번호는 4, 5, 6, 7 번

  pinMode(4, OUTPUT);     // 4번핀을 출력으로 설정
}

void loop()
{
  // LED 깜박이기

  digitalWrite(4, HIGH);   // LED 켜기  (4번 핀을 HIGH로 변경)

  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(4, LOW);    // LED 끄기  (4번 핀을 LOW로 변경)

  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)
```

2. Blink Two LED
(LED 2개를 깜빡이기)

- 목표 digitalWrite 사용방법 익히기
- 함수 digitalWrite (핀번호, 출력 선택)
- 설명 1개의 LED를 깜박이는 예제를 수정하여 2개의 LED가 깜박이도록 만들어 봅시다.

```
//--------------------------- Mission 답안 ------------------------------//

void setup()
{
  // 사용할 수 있는 LED 번호는 4, 5, 6, 7 번
  pinMode(4, OUTPUT);      // 4번 핀을 출력으로 설정
  pinMode(5, OUTPUT);      // 5번 핀을 출력으로 설정
}

void loop()
{
  digitalWrite(4, HIGH);   // LED 켜기  (4번 핀을 HIGH로 변경)
  digitalWrite(5, HIGH);   // LED 켜기  (5번 핀을 HIGH로 변경)

  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(4, LOW);    // LED 끄기  (4번 핀을 LOW로 변경)
  digitalWrite(5, LOW);    // LED 끄기  (5번 핀을 LOW로 변경)

  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)
}
```

3. Wave Two LED
(LED 2개를 차례로 켜고 끄기)

- 목표 LED 를 사용하여 순차 프로그래밍 익히기
- 함수 digitalWrite (핀번호, 출력 선택)
- 설명 2개의 LED 가 차례로 켜지고 꺼집니다.

```
void setup()
{
  // 사용할 수 있는 LED 번호는 4, 5, 6, 7 번

  pinMode(4, OUTPUT);       // 4번 핀을 출력으로 설정
  pinMode(5, OUTPUT);       // 5번 핀을 출력으로 설정
}

void loop()
{
  // LED 차례로 켜기

  digitalWrite(4, HIGH);   // LED 켜기  (4번 핀을 HIGH로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(5, HIGH);   // LED 켜기  (5번 핀을 HIGH로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)


  // LED 차례로 끄기

  digitalWrite(5, LOW);    // LED 끄기  (5번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(4, LOW);    // LED 끄기  (4번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)
}
```

4. Wave Three LED
(LED 3개를 차례로 켜고 끄기)

- 목표 LED 를 사용하여 순차 프로그래밍 익히기
- 함수 digitalWrite (핀번호, 출력 선택)
- 설명 2개의 LED 가 차례로 켜지는 예제를 3개의 LED 가 켜지고 꺼지게 만들어 봅시다.

```
//------------------------------ Mission 답안 --------------------------- //

void setup()
{
  // 사용할 수 있는 LED 번호는 4, 5, 6, 7 번

  pinMode(4, OUTPUT);      // 4번 핀을 출력으로 설정
  pinMode(5, OUTPUT);      // 5번 핀을 출력으로 설정
  pinMode(5, OUTPUT);      // 6번 핀을 출력으로 설정
}

void loop()
{
  // LED 차례로 켜기
  digitalWrite(4, HIGH);   // LED 켜기  (4번 핀을 HIGH로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(5, HIGH);   // LED 켜기  (5번 핀을 HIGH로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(6, HIGH);   // LED 켜기  (6번 핀을 HIGH로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  // LED 차례로 끄기
  digitalWrite(6, LOW);    // LED 끄기  (6번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(5, LOW);    // LED 끄기  (5번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(4, LOW);    // LED 끄기  (4번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)
}
```

5. Move Two LED
(LED 2개를 한 개씩 차례로 켜고 끄기)

- 목표 LED 를 사용하여 순차 프로그래밍 익히기
- 함수 digitalWrite (핀번호, 출력 선택)
- 설명 2개의 LED 가 차례대로 번갈아 켜지고 꺼집니다.

```
void setup()
{
  // 사용할 수 있는 LED 번호는 4, 5, 6, 7 번

  pinMode(4, OUTPUT);       // 4번 핀을 출력으로 설정
  pinMode(5, OUTPUT);       // 5번 핀을 출력으로 설정
}

void loop()
{
  digitalWrite(4, HIGH);   // LED 켜기  (4번 핀을 HIGH로 변경)
  digitalWrite(5, LOW);    // LED 끄기  (5번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(5, HIGH);   // LED 켜기  (5번 핀을 HIGH로 변경)
  digitalWrite(4, LOW);    // LED 끄기  (4번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)
}
```

Mission

6. Move Three LED
(LED 3개를 한 개씩 차례로 켜고 끄기)

- 목표 LED 를 사용하여 순차 프로그래밍 익히기
- 함수 digitalWrite (핀번호, 출력 선택)
- 설명 다음 예제를 수정하여 3개의 LED가 차례대로 번갈아 켜지고 꺼집니다.

```
//------------------------------ Mission 답안 ------------------------------//

void setup()
{
  // 사용할 수 있는 LED 번호는 4, 5, 6, 7 번

  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(5, OUTPUT);
}

void loop()
{
  digitalWrite(4, HIGH);   // LED 켜기  (4번 핀을 HIGH로 변경)
  digitalWrite(6, LOW);    // LED 끄기  (6번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(5, HIGH);   // LED 켜기  (5번 핀을 HIGH로 변경)
  digitalWrite(4, LOW);    // LED 끄기  (4번 핀을 LOW로 변경)
  delay(1000);             // 1초 기다리기 (1000 Milliseconds = 1second)

  digitalWrite(6, HIGH);   // LED 켜기  (6번 핀을 HIGH로 변경)
  digitalWrite(5, LOW);    // LED 끄기  (5번 핀을 LOW로 변경)
  delay(1000);
}
```

STEP 2 Serial

시리얼통신

01 시리얼 통신을 사용해서 문자를 나타내봅니다.

02 간단한 통신의 개념을 배웁니다.

1. Serial Print
(시리얼 통신 문자 나타내기)

- 목표 Serial 의 사용 방법 익히기

- 함수 Serial.begin (통신속도)
  Serial.println (보낼 데이터)

- 설명 시리얼 통신기능을 사용하여 문자를 보냅니다. 프로그램입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  "Hello World!" 메시지가 시리얼 모니터창에 나타납니다.

```
void setup()
{
  Serial.begin(57600);
// 시리얼 통신을 사용하고, 통신 속도를 57600 bps로 설정

  Serial.println("Hello World!"); // 문자를 출력하고 줄을 바꿉니다.
}

void loop()
{
}
```

2. Serial Print 2
(시리얼 통신 문자 나타내기)

- **목표** Serial 의 사용방법 익히기

- **함수** Serial.begin(통신속도)
  Serial.println(보낼 데이터)
  Serial.println(보낼 데이터)

- **설명** 시리얼 통신기능을 사용하여 문자를 보냅니다. 프로그램입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  일정 시간마다 "Hello World!" 메시지가 시리얼 모니터창에 나타납니다.
  Serial.print를 사용하면 줄을 바꾸지 않고 문자를 나타낼 수 있습니다.
  Serial.println을 사용하면 문자를 나타내고 줄을 바꿉니다.

```
void setup()
{
  Serial.begin(57600);
// 시리얼 통신을 사용하고, 통신 속도를 57600 bps로 설정
}

void loop()
{
  Serial.print("Hello ");      // 문자를 출력하고 줄을 바꾸지 않습니다.
  delay(1000);                 // 1초 기다리기 (1000 Milliseconds = 1second)

  Serial.println("World!");    // 문자를 출력하고 줄을 바꿉니다.
  delay(2000);                 // 2초 기다리기 (2000 Milliseconds = 2second)
}
```

Mission

3. Intoroduce Myself
(자기 소개를 나타내기)

- **목표** Serial 의 사용방법 익히기
- **함수** Serial.begin(통신속도)
  Serial.println(보낼 데이터)
  Serial.println(보낼 데이터)
- **설명** 시리얼 통신기능을 사용하여 문자를 보냅니다.
  프로그램입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  비어있는 "" 안에 이름과 취미를 넣어서 내용을 완성해보세요.

```
void setup()
{
  Serial.begin(57600);
// 시리얼 통신을 사용하고, 통신 속도를 57600 bps로 설정

  Serial.println("Hi~");
// 문자를 출력하고 줄을 바꿉니다.

  Serial.print("My Name is ");
 // 문자를 출력하고 줄을 바꾸지 않습니다.

  Serial.println("Kevin.");
  // 문자를 출력하고 줄을 바꿉니다. 이름을 입력합니다.

  Serial.print("My hobby is ");
  // 문자를 출력하고 줄을 바꾸지 않습니다.

  Serial.println("listening to music.");
// 문자를 출력하고 줄을 바꿉니다. 취미를 입력합니다.

}

void loop()
{
}
```

STEP 3 Button

버튼

01 버튼을 사용합니다.

02 입력에 대한 개념을 배웁니다.

1. Digital Read Serial
(핀의 디지털 상태 값을 읽기)

- 목표 digitalRead 의 사용 방법 익히기
- 함수 digitalRead (핀번호)
- 설명 이 기능은 코드로더 보드에 있는 버튼의 디지털 상태 값을 읽을 수 있습니다.
버튼으로 사용 가능한 번호는 2,3 입니다.
버튼을 누르지 않을 때의 상태값은 HIGH(1)이며, 누른 경우에는 LOW(0) 값을 나타냅니다.
프로그램입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.
한개 버튼의 디지털 상태를 시리얼 모니터창에 출력하는 예제입니다.

```
void setup()
{
  Serial.begin(57600);
// 시리얼 통신을 사용하고, 통신 속도를 57600 bps로 설정
  pinMode(2, INPUT);
// 2번핀을 입력으로 설정 (사용 가능한 버튼 번호 : 2, 3)
}

void loop()
{
  int state = digitalRead(2);  // 버튼의 상태값을 읽어옵니다.
  Serial.print("State : ");    // "State : " 문자열 출력
  Serial.println(state);       // 버튼의 상태값을 출력
}
```

2. Button All
(두 개 버튼의 디지털 상태 값을 읽기)

- **목표** digitalRead 의 사용방법 익히기
  여러 개의 입력에 대한 처리방법
- **함수** digitalRead (핀번호)
- **설명** 이 기능은 코드로더 보드에 있는 버튼의 디지털 상태 값을 읽을 수 있습니다.
  버튼으로 사용 가능한 번호는 2,3 입니다.

  버튼을 누르지 않을 때의 상태값은 HIGH(1)이며, 누른 경우에는 LOW(0) 값을 나타냅니다.

  프로그램입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  다음 예제를 수정하여 두개 버튼의 디지털 상태를 시리얼 모니터창에 출력하도록 만들어 보세요.

```
void setup()
{
  Serial.begin(57600);
// 시리얼 통신을 사용하고, 통신 속도를 57600 bps로 설정
  pinMode(2, INPUT);
// 2번 핀을 입력으로 설정 (사용 가능한 버튼 번호 : 2, 3)
  pinMode(3, INPUT);
// 3번 핀을 입력으로 설정 (사용 가능한 버튼 번호 : 2, 3)
}

void loop()
{
  int state1 = digitalRead(2);   // 첫번째 버튼의 상태값을 읽어옵니다.
  int state2 = digitalRead(3);   // 두번째 버튼의 상태값을 읽어옵니다.

  Serial.print("State1 : ");     // "State1 : " 문자열 출력
  Serial.print(state1);
// 두번째 버튼의 상태값을 출력 (줄을 변경합니다.)

  Serial.print(" , ");           // 문장에 공백을 입력 합니다.

  Serial.print("State2 : ");     // "State2 : " 문자열 출력
  Serial.println(state2);
// 두번째 버튼의 상태값을 출력 (줄을 변경합니다.)
}
```

3. Button with LED
(버튼으로 LED를 켜고 끄기)

- **목표** digitalRead 의 사용방법 익히기
  입력과 출력의 동시 제어
- **함수** digitalRead (핀번호)
- **설명** 이 기능은 코드로더 보드에 있는 버튼의 디지털 상태 값을 읽을 수 있습니다.
  버튼으로 사용 가능한 번호는 2,3 입니다.
  버튼을 누르지 않을 때의 상태값은 HIGH(1)이며, 누른 경우에는 LOW(0) 값을 나타냅니다.

  한개 버튼의 디지털 상태에 따라서 LED를 켜거나 끄는 예제입니다.

```
void setup()
{
  pinMode(7, OUTPUT);
// 7번 핀을 출력으로 설정 (사용 가능한 LED 번호 : 4, 5, 6, 7)
  pinMode(2, INPUT);
// 2번 핀을 입력으로 설정 (사용 가능한 버튼 번호 : 2, 3)
}

void loop()
{
  int buttonState = digitalRead(2); // 버튼의 상태값을 읽어옵니다.

  if (buttonState == LOW)
  // 버튼 상태 값이 LOW 인경우 (버튼을 누른 경우)
  {
    digitalWrite(7, HIGH);      // LED 7번 켜기  (7번 핀을 HIGH로 변경)
  }
  else
  {
    digitalWrite(7, LOW);       // LED 7번 끄기  (7번 핀을 LOW로 변경)
  }
}
```

Mission

4. Button ALL with LED
(버튼으로 두개의 LED를 켜고 끄기)

- **목표** digitalRead 의 사용방법 익히기
  여러개의 DLQFURRHK 출력의 동시제어
- **함수** digitalRead (핀번호)
- **설명** 이 기능은 코드로더 보드에 있는 버튼의 디지털 상태 값을 읽을 수 있습니다.
  버튼으로 사용 가능한 번호는 2,3 입니다.
  버튼을 누르지 않을 때의 상태값은 HIGH(1)이며, 누른 경우에는 LOW(0) 값을 나타냅니다.
  다음 예제를 수정하여 두개 버튼의 디지털 상태에 따라서 각각의 LED를 켜거나 끄도록 만들어 보세요.

```
void setup()
{
  pinMode(7, OUTPUT);     // 4번 핀을 출력으로 설정
  pinMode(4, OUTPUT);     // 7번 핀을 출력으로 설정

  pinMode(2, INPUT);   // 2번 핀을 입력으로 설정
  pinMode(3, INPUT);   // 3번 핀을 입력으로 설정
}
void loop()
{
  int buttonState1 = digitalRead(2);
// 첫번째 버튼의 상태값을 읽어옵니다.
  int buttonState2 = digitalRead(3);
// 두번째 버튼의 상태값을 읽어옵니다.
  if (buttonState1 == LOW)
// 첫번째 버튼상태값이 LOW 인경우 (버튼을 누른 경우)
  {
    digitalWrite(7, HIGH);  // LED 7번 켜기  (7번 핀을 HIGH로 변경)
  }
  else
  {
    digitalWrite(7, LOW);   // LED 7번 끄기  (7번 핀을 LOW로 변경)
  }
  if (buttonState2 == LOW)
// 두번째 버튼상태값이 LOW 인경우 (버튼을 누른 경우)
  {
    digitalWrite(4, HIGH);  // LED 4번 켜기  (4번 핀을 HIGH로 변경)
  }
  else
  {
    digitalWrite(4, LOW);   // LED 4번 끄기  (4번 핀을 LOW로 변경)
  }
}
```

STEP 4 Buzzer

부저

01 부저를 사용하여 소리를 냅니다.

02 음계를 사용하여 멜로디를 만들어 봅니다.

03 자신만의 멜로디를 만들어 봅니다.

1. Tone (소리내기)

- **목표** Tone의 사용방법 익히기
- **함수** tone(pin, frequency)
  tone(pin, frequency, duration)
- **설명** 이 기능은 코드로더 보드에 있는 부저로 소리를 발생시켜 멜로디를 재생할수 있습니다. 부저로 사용 가능한 핀 번호는 8번입니다.
  간단하게 소리를 출력하는 예제입니다.

```
int buzzPin = 8;        // 변수에 부저 핀번호를 입력합니다.

void setup()
{
  tone(buzzPin, 300);   // 주파수가 300에 해당하는 소리를 출력합니다.
  delay(500);           // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, 800);   // 주파수가 800에 해당하는 소리를 출력합니다.
  delay(500);           // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, 500);   // 주파수가 500에 해당하는 소리를 출력합니다.
  delay(500);           // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  noTone(buzzPin);      // 음을 멈춥니다.
}

void loop() {
}

// 프로그램을 실행하면 8개의 음으로 된 멜로디가 재생됩니다.
```

2. Tone Melody
(멜로디 연주하기)

- **목표** Tone의 사용방법 익히기

- **함수** tone(pin, frequency)
  tone(pin, frequency, duration)

- **설명** 이 기능은 코드로더 보드에 있는 부저로 음을 발생시켜 멜로디를 재생할수 있습니다.
  부저로 사용 가능한 핀 번호는 8번 입니다.
  입력 값은 다음과 같습니다.

  - pin: tone을 발생시킬 핀
  - frequency: tone의 주파수 (Hz 단위)
  - unsigned int
  - duration (옵션) : tone 의 지속 시간 (밀리초 단위)
  - unsigned long

특정 음에 해당하는 주파수를 **pitch.h**에서 정의해놓았습니다.
이를 사용하면 원하는 음계를 낼 수 있습니다.

음계는 다음과 같이 표시됩니다.
(C 도) (D 레) (E 미) (F 파) (G 솔) (A 라) (B 시) (C 도)
NOTE_C4는 4옥타브의 '도' 라는 의미입니다.
NOTE_C6는 6옥타브의 '도' 라는 의미입니다.

다음 예제는 '도,레,미' 멜로디를 출력 합니다.

```
#include "pitches.h"  // 특정 음에 해당되는 주파수를 모아놓은 헤더파일

int buzzPin = 8;      // 변수에 부저 핀번호를 입력합니다.
int duration = 500;   // 음길이 : Tone 의 지속 시간 (밀리초 단위 500 = 0.5초)

void setup()
{
  tone(buzzPin, NOTE_C4, duration);   // 4 옥타브 '도'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_D4, duration);   // 4 옥타브 '레'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_E4, duration);   // 4 옥타브 '미'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)
}
void loop() {
}
```

3. Tone Melody 2
(멜로디 연주하기)

- **목표** Tone의 사용방법 익히기
- **함수** tone(pin, frequency)
  tone(pin, frequency, duration)
- **설명** 이 기능은 코드로더 보드에 있는 부저로 음을 발생시켜 멜로디를 재생할수 있습니다. 부저로 사용 가능한 핀 번호는 8번 입니다.
  입력 값은 다음과 같습니다.
  - pin: tone을 발생시킬 핀
  - frequency: tone의 주파수 (Hz 단위)
  - unsigned int
  - duration (옵션) : tone 의 지속 시간 (밀리초 단위)
  - unsigned long

특정 음에 해당하는 주파수를 pitch,h에서 정의해 놓았습니다.
이를 사용하면 원하는 음계를 낼 수 있습니다.
음계는 다음과 같이 표시됩니다.
(C 도) (D 레) (E 미) (F 파) (G 솔) (A 라) (B 시) (C 도)
NOTE_C4는 4옥타브의 '도' 라는 의미입니다.
NOTE_C6는 6옥타브의 '도' 라는 의미입니다.
다음 예제를 수정하여 '도,레,미,파,솔,라,시,도' 소리를 내도록 만들어 봅시다.

```
#include "pitches.h"  // 특정 음에 해당되는 주파수를 모아놓은 헤더파일

int buzzPin = 8;      // 변수에 부저 핀번호를 입력합니다.
int duration = 500;   // 음길이 : Tone 의 지속 시간 (밀리초 단위)

void setup()
{
  tone(buzzPin, NOTE_C4, duration);   // 4 옥타브 '도'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_D4, duration);   // 4 옥타브 '레'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_E4, duration);   // 4 옥타브 '미'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_F4, duration);   // 4 옥타브 '파'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_G4, duration);   // 4 옥타브 '솔'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_A4, duration);   // 4 옥타브 '라'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_B4, duration);   // 4 옥타브 '시'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)

  tone(buzzPin, NOTE_C5, duration);   // 4 옥타브 '도'를 0.5초 내기
  delay(500);                         // 0.5 초 기다리기 (500 Milliseconds = 0.5 second)
}

void loop() {
}
```

4. Tone Melody Button
(버튼으로 멜로디 연주하기)

- **목표** Tone의 사용방법 익히기
  센서 입력에 따른 음의 재생
- **함수** tone(pin, frequency)
  tone(pin, frequency, duration)
- **설명** 이 기능은 코드로더 보드에 있는 부저로 음을 발생시켜 멜로디를 재생할수 있습니다. 부저로 사용 가능한 핀 번호는 8번 입니다. 입력 값은 다음과 같습니다.

- **pin: tone을 발생시킬 핀**
- **frequency: tone의 주파수 (Hz 단위)**
- **unsigned int**
- **duration (옵션) : tone 의 지속 시간 (밀리초 단위)**
- **unsigned long**

특정 음에 해당하는 주파수를 pitch,h에서 정의해 놓았습니다.
이를 사용하면 원하는 음계를 낼 수 있습니다.
음계는 다음과 같이 표시됩니다.
(C 도) (D 레) (E 미) (F 파) (G 솔) (A 라) (B 시) (C 도)
NOTE_C4는 4옥타브의 '도' 라는 의미입니다.
NOTE_C6는 6옥타브의 '도' 라는 의미입니다.
다음 예제는 2개의 버튼을 사용하여 각각의 음이 출력 되도록 하는 예제입니다.

```
#include "pitches.h"    // 특정 음에 해당되는 주파수를 모아놓은 헤더파일

int buzzPin = 8;        // 변수에 부저 핀번호를 입력합니다.

void setup()
{
  pinMode(2, INPUT);    // 2번 핀을 입력으로 설정 (사용 가능한 버튼 번호 : 2, 3)
  pinMode(3, INPUT);    // 3번 핀을 입력으로 설정 (사용 가능한 버튼 번호 : 2, 3)
}

void loop()
{
  int buttonState1 = digitalRead(2);  // 2번 버튼의 상태 값을 읽어옵니다.
  int buttonState2 = digitalRead(3);  // 3번 버튼의 상태 값을 읽어옵니다.

  if (buttonState1 == LOW)
// 첫번째 버튼 상태 값이 LOW 인경우 (버튼을 누른 경우)
  {
    tone(buzzPin, NOTE_G3);
// 3 옥타브의 '솔'을 내기
  }

  else if (buttonState2 == LOW)
// 두번째 버튼 상태 값이 LOW 인경우 (버튼을 누른 경우)
  {
    tone(buzzPin, NOTE_C4);           // 4 옥타브의 '도'를 내기
  }
  else
  {
    noTone(buzzPin);                  // 음을 멈춤니다.
  }
}
```