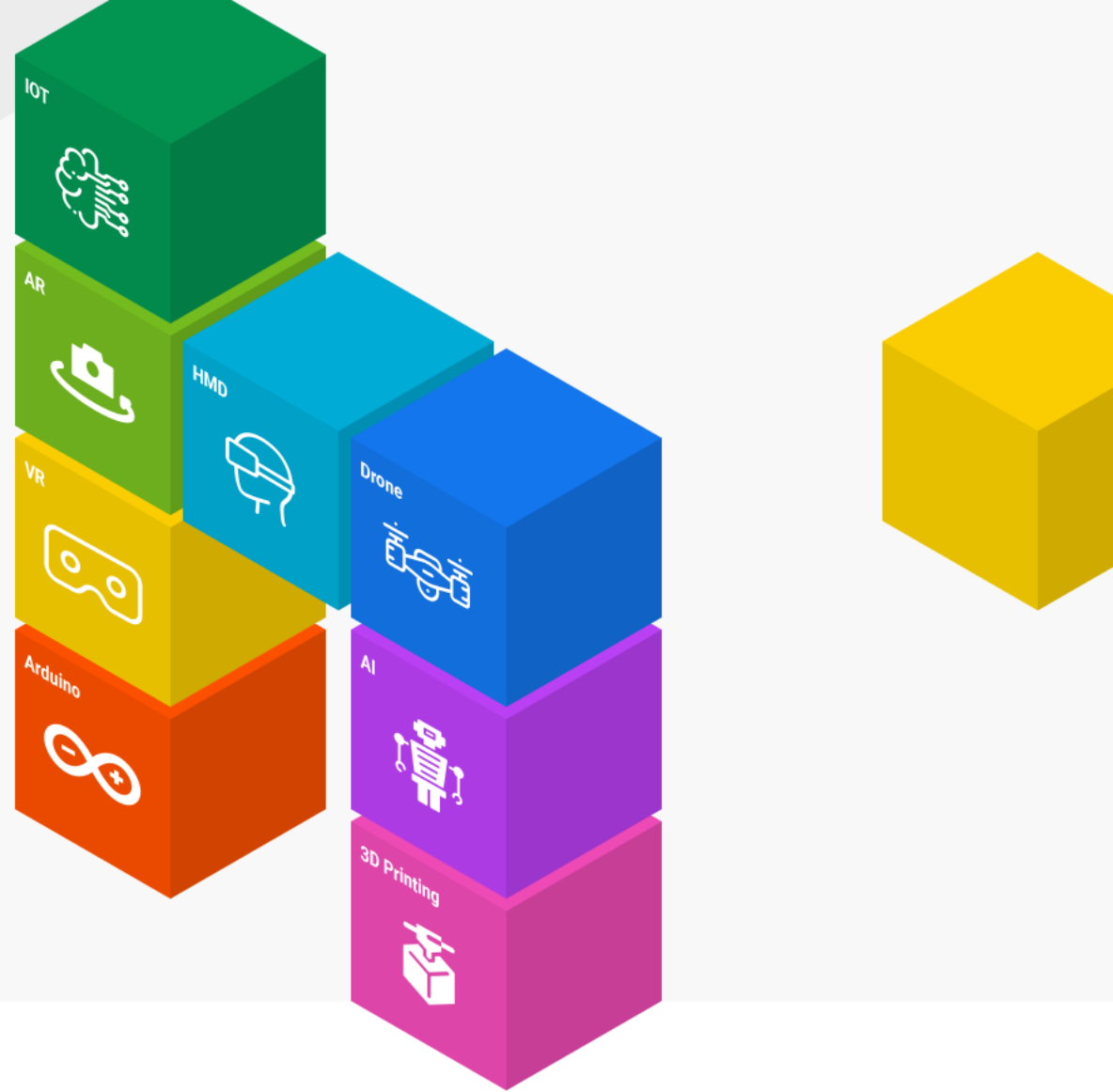


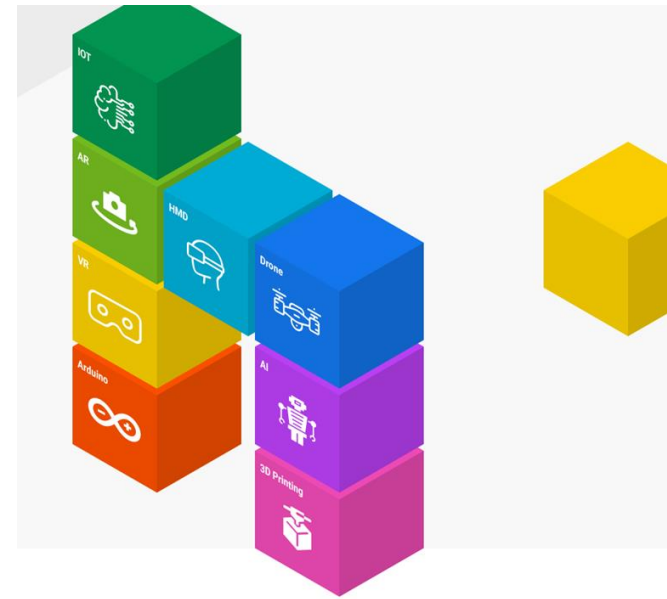
# [아두이노 시뮬레이션 코딩] 디지털 명령어 읽기 및 버튼 활용하기



[www.helloapps.co.kr](http://www.helloapps.co.kr)

김 영 준 / 070-4417-1559 / splduino@gmail.com

# 아두이노에서 읽기 용도의 센서



# 아두이노에서의 버튼 센서



버튼도 LED와 같이 디지털 핀에 연결합니다.



디지털 버튼

버튼이 눌러지면 1 또는 HIGH 값이  
읽혀짐

버튼이 눌러지지 않으면 0 또는 LOW 값이  
읽혀짐

# 시뮬레이션 연결 환경 이해하기

4번부터 8번까지 5개의  
버튼이 연결되어 있음

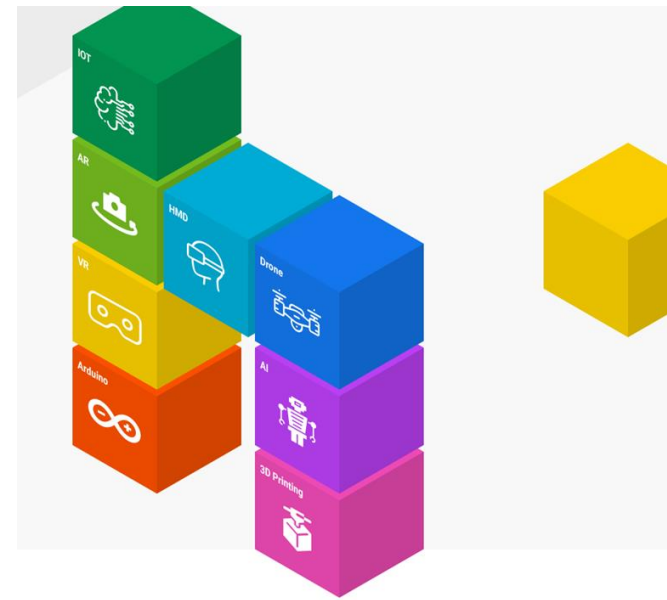
## 디지털 핀

D02 서보모터	0
D04 버튼	버튼1
D05 버튼	버튼2
D06 버튼	버튼3
D07 버튼	버튼4
D08 버튼	버튼5
D09 스피커	
D11 LED	GREEN
D12 LED	YELLOW
D13 LED	RED

## 아날로그 핀

A0 거리센서	0
A1 거리센서	982
A2 거리센서	0
A3 거리센서	983
A4 거리센서	471
A5 조도센서	

# 4번 핀 버튼의 값을 화면에 출력하기

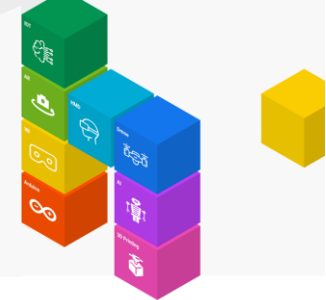


# 4번 핀 버튼의 값을 화면에 출력하기

4번 핀 버튼의 상태 값을  
읽어서 화면에 출력



# 4번 핀 버튼의 값을 화면에 출력하기



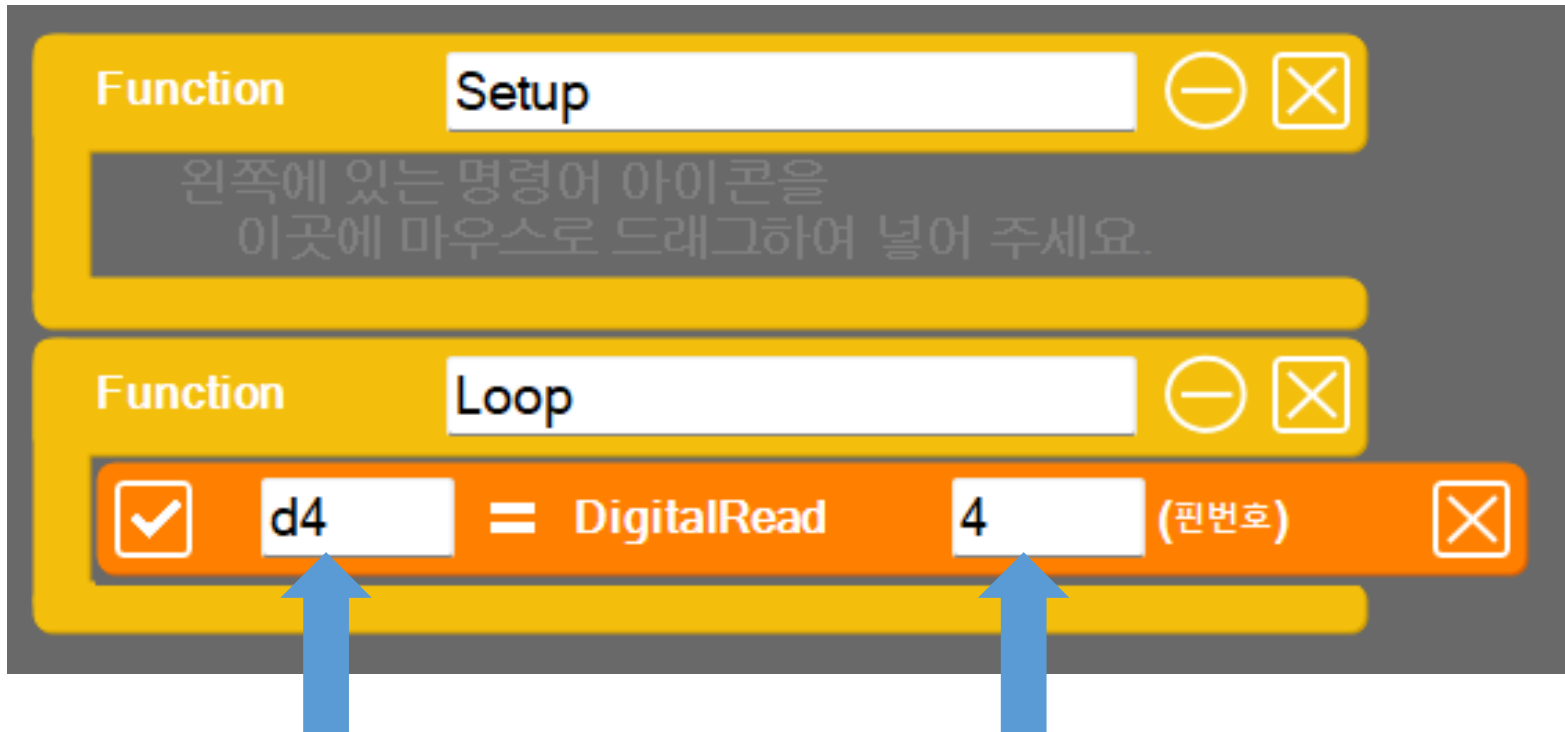
[1-1-2] DigitalRead 명령어를 Loop 함수 안에 추가합니다.

The screenshot displays the HelloApps software interface. On the left, a sidebar contains a list of commands categorized into three groups: [0] 트직 명령어 (Logic Commands), [1] 기본 명령어 (Basic Commands), and [2] 내장 명령어 (Built-in Commands). Under the [1] 기본 명령어 group, the [1-1-2] DigitalRead command is highlighted with a red arrow pointing to its icon. The main workspace on the right shows a block-based programming environment with two function blocks: 'Function Setup' and 'Function Loop'. The 'Function Loop' block contains a 'DigitalRead' command block with the pin number '2' and a label '(핀번호)'. A red arrow points from the 'DigitalRead' command in the sidebar to the 'DigitalRead' command in the 'Function Loop' block. The 'Function Setup' block contains a text box with the Korean text: '왼쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.'

# 4번 핀 버튼의 값을 화면에 출력하기



DigitalRead 명령어의 변수 값과 핀번호를 다음과 같이 수정합니다.





# 4번 핀 버튼의 값을 화면에 출력하기



DigitalRead 명령어의 변수 값과 핀번호를 다음과 같이 수정합니다.



d4 = DigitalRead(4)

디지털 4번 핀에서 값을 읽어서 변수 d4에  
저장하라는 의미임

# 4번 핀 버튼의 값을 화면에 출력하기



PrintLine 명령어를 추가합니다.

The screenshot shows the Hello Apps software interface. On the left, a command palette lists various functions. Under the '기본 명령어' (Basic Commands) category, the 'PrintLine' command is highlighted. A red arrow points from this command to the 'Loop' function block in the main workspace. The 'Loop' block contains two sub-blocks: a 'DigitalRead' block set to pin 4, and a 'PrintLine' block with 'd4' as the input. The 'Setup' function block is also visible above the 'Loop' block.

Function Setup

왼쪽에 있는 명령어 아이콘을  
이곳에 마우스로 드래그하여 넣어 주세요.

Function Loop

☒ d4 = DigitalRead 4 (핀번호)

☒ PrintLine d4

# 4번 핀 버튼의 값을 화면에 출력하기

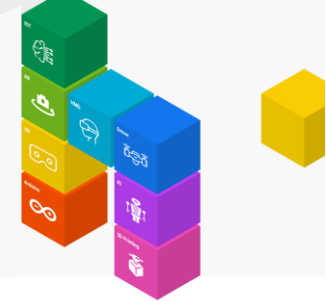


Delay 명령어를 추가한 후, 값을 100 밀리초 (0.1초)로 수정해 줍니다.

The screenshot shows the HelloApps IDE interface. On the left, a sidebar lists various blocks. Under the '기본 명령어' (Basic Commands) category, the 'Delay' block is highlighted. A red arrow points from this 'Delay' block in the sidebar to the 'Delay' block within the 'Loop' function in the main workspace. The 'Loop' function contains three sub-blocks: 'd4 = DigitalRead 4 (핀번호)', 'PrintLine d4', and 'Delay 100 (밀리초)'. A blue arrow points to the '100' value in the 'Delay' block, indicating the time delay in milliseconds.

값을 100으로 수정합니다 (0.1초)

# 실행하기



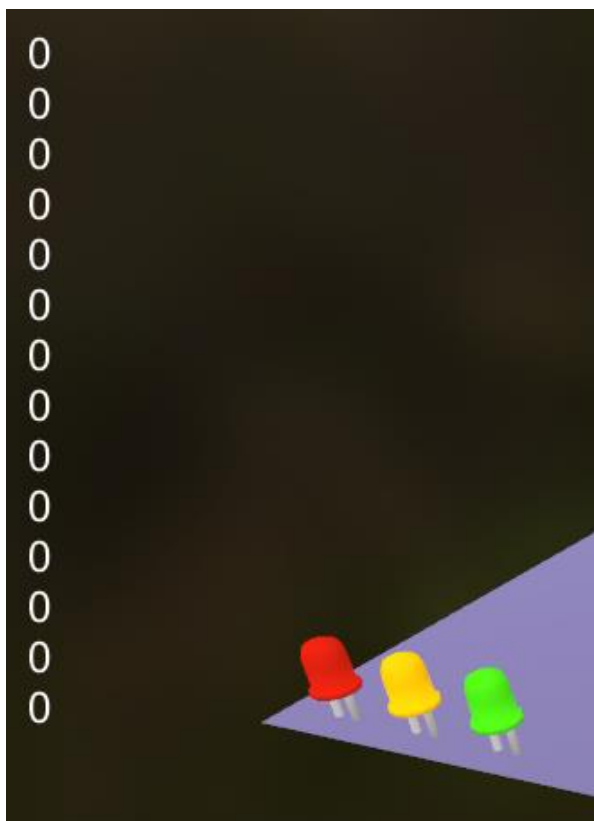
- 실행 버튼을 클릭합니다.



# 4번 핀 버튼의 값을 화면에 출력하기

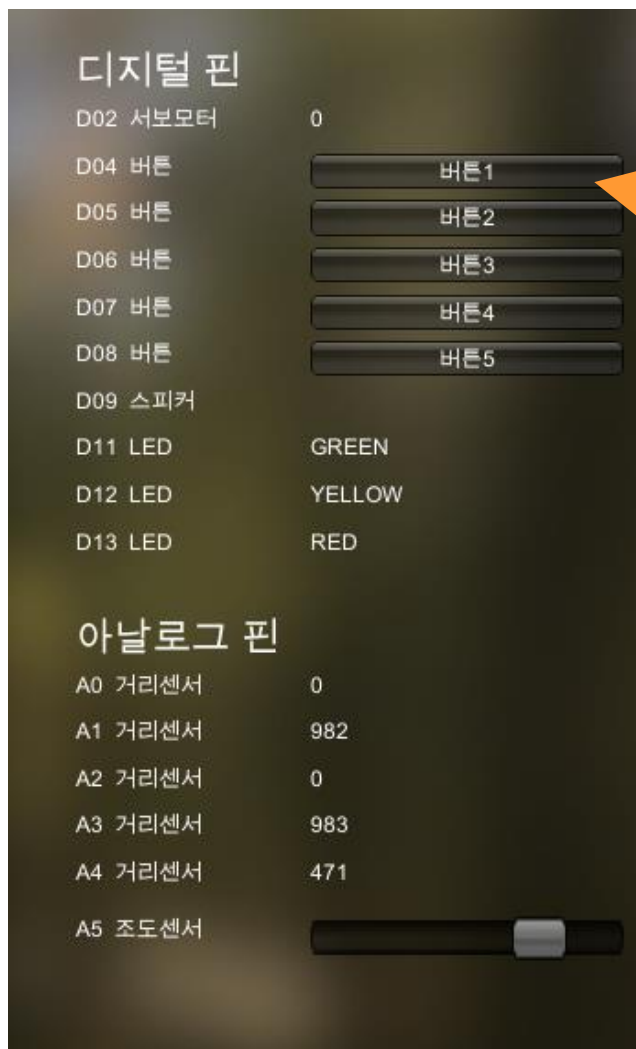


4번 핀 버튼이 눌러지지 않은 상태에서는 0 값이 출력됩니다.



0은 LOW와 같은 의미입니다.

# 4번 핀 버튼의 값을 화면에 출력하기



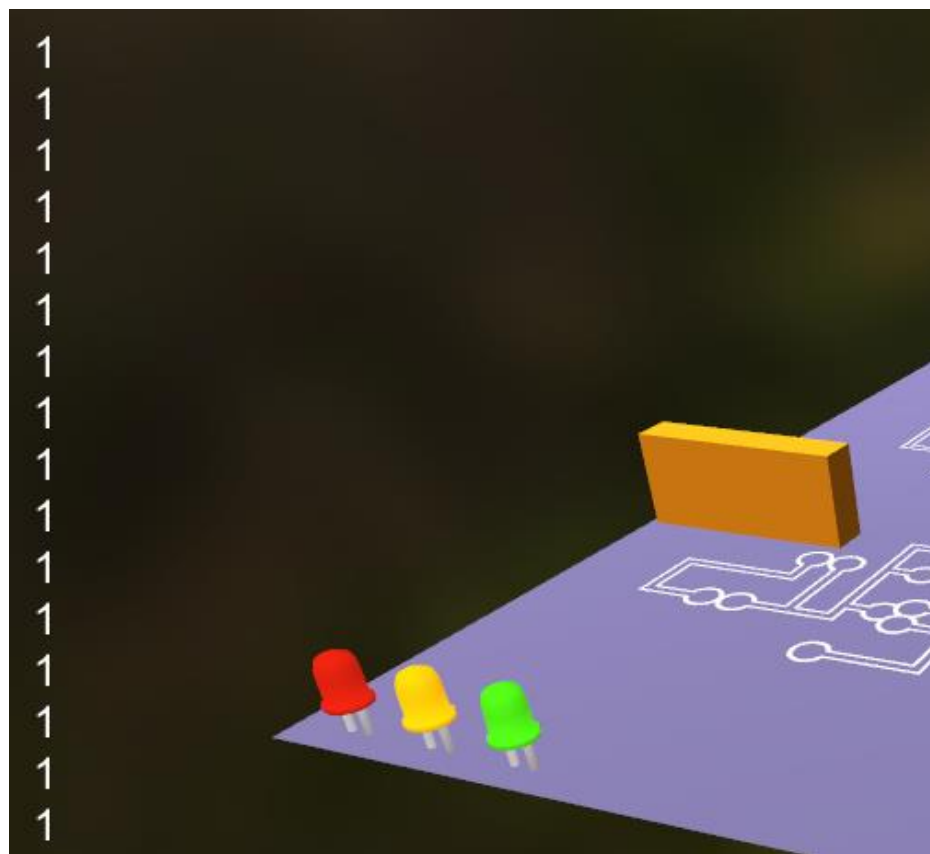
4번 핀에 연결된 버튼1을  
마우스로 클릭합니다.

# 4번 핀 버튼의 값을 화면에 출력하기

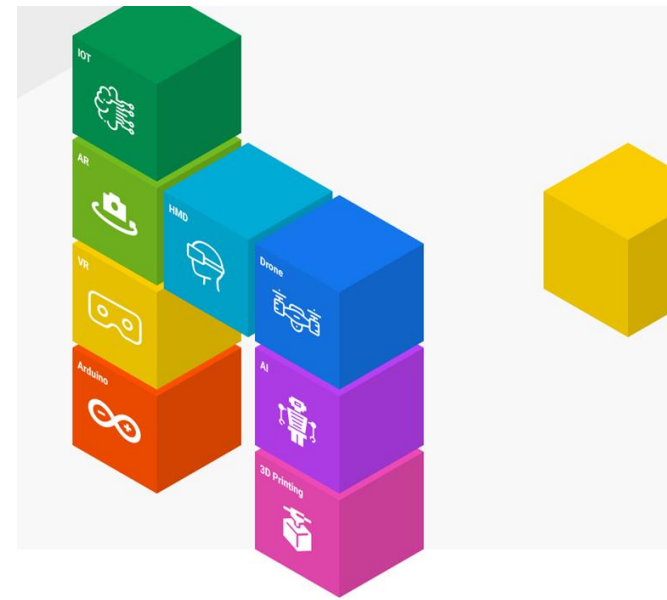


4번 핀에 연결된 버튼1을 누르고 있으면 1 값이 읽혀집니다.

1은 HIGH와 같은 의미입니다.



# 버튼으로 LED 켜기



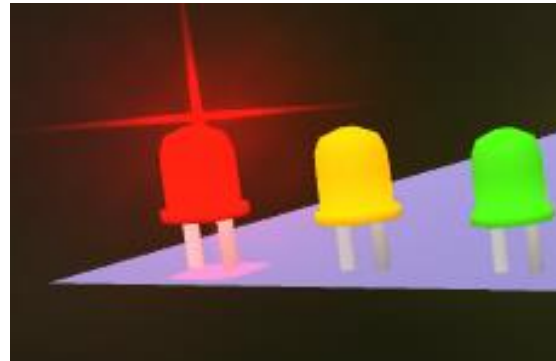


# 버튼으로 LED 켜기



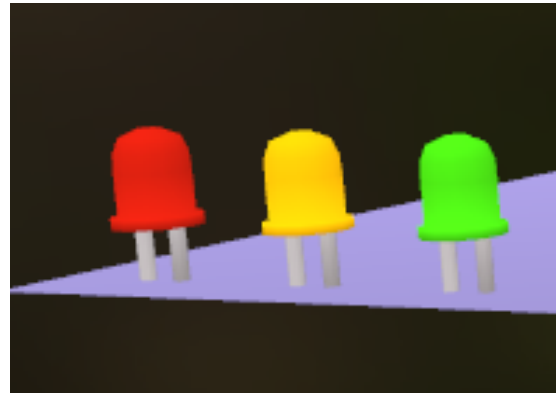
4번 핀에 연결된 버튼1을 누르면 13번 LED가 켜지고 그렇지 않으면 꺼지는 기능을 구현해 봅니다.

4번 핀에 연결된 버튼1을  
마우스로 클릭



13번 핀의 LED가  
켜짐

버튼을 클릭하지 않음



13번 핀의 LED가  
꺼짐

# if – else 명령어로 값 비교하기

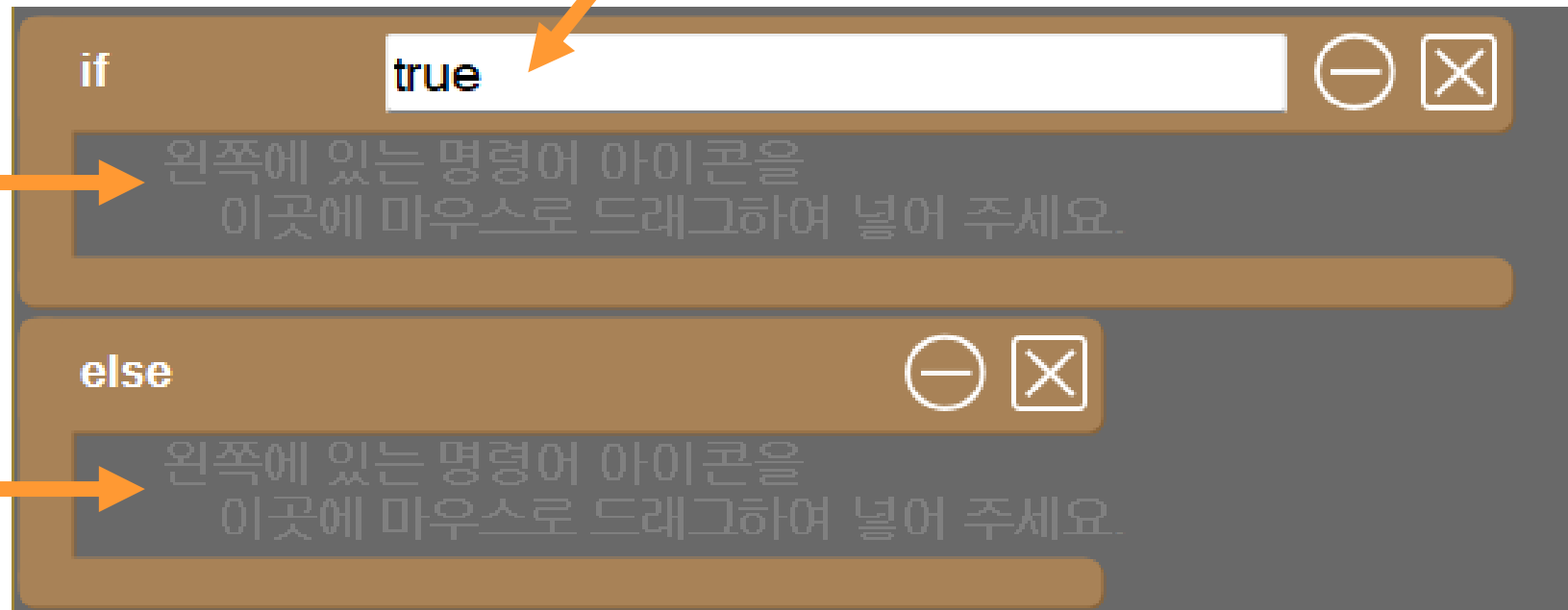


if와 else 블록을 이용하여 조건이 만족되는 경우와 그렇지 않은 경우에 따라서 명령어를 실행시킬 수 있습니다.

비교할 조건 수식을 이곳에 입력

if 조건이 참(true)이면  
if 블록 안의 명령어가  
실행됨

if 조건이 거짓(false)이면  
else 블록 안의 명령어가  
실행됨



# if – else 명령어 추가하기



if와 else 블록 명령어를 추가합니다. (Delay 명령어 위에 순서대로 추가합니다)

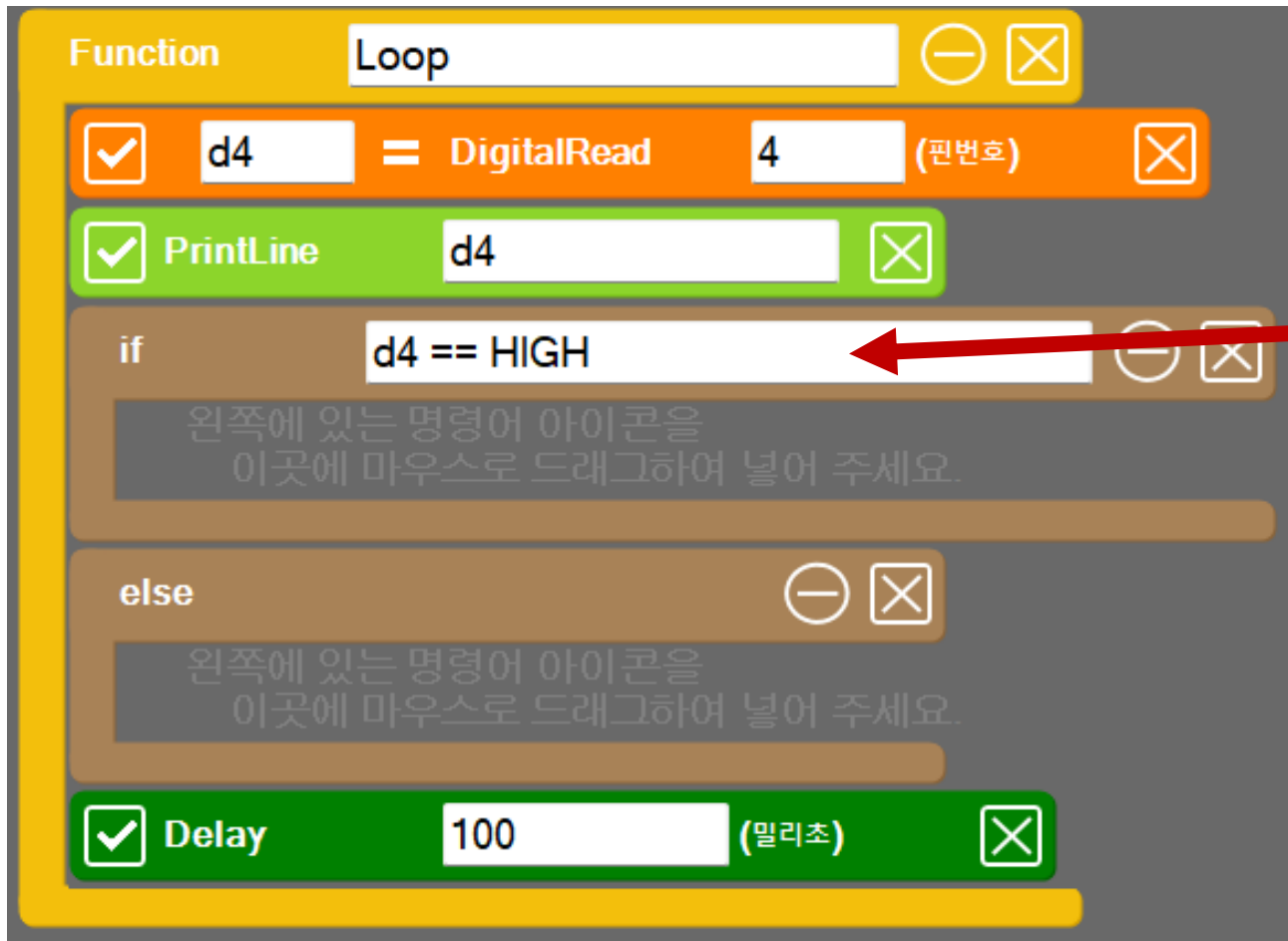
The screenshot displays the Hello Apps software interface. On the left, a panel lists various command blocks categorized by tabs: [0] 트릭 명령어 (Trick commands), [1] 기본 명령어 (Basic commands), [2] 내장 명령어 (Built-in commands), and [3] 3D 명령어 (3D commands). Under the [1] 기본 명령어 tab, the 'if' and 'else' blocks are highlighted with red arrows pointing towards the main workspace. The main workspace shows a 'Function' block named 'Loop' containing several commands: 'd4 = DigitalRead 4', 'PrintLine d4', an 'if true' block, and a 'Delay 100' block. The 'if' block is currently empty, and the 'else' block is also empty. The 'Delay' block is positioned at the bottom of the 'if-else' structure, indicating the sequence of addition.

hello apps 헬로앱스

# if – else 명령어 추가하기



if 조건문의 수식을 다음과 같이 수정합니다.



d4 == HIGH

= 기호가 두 번 반복됩니다.

==

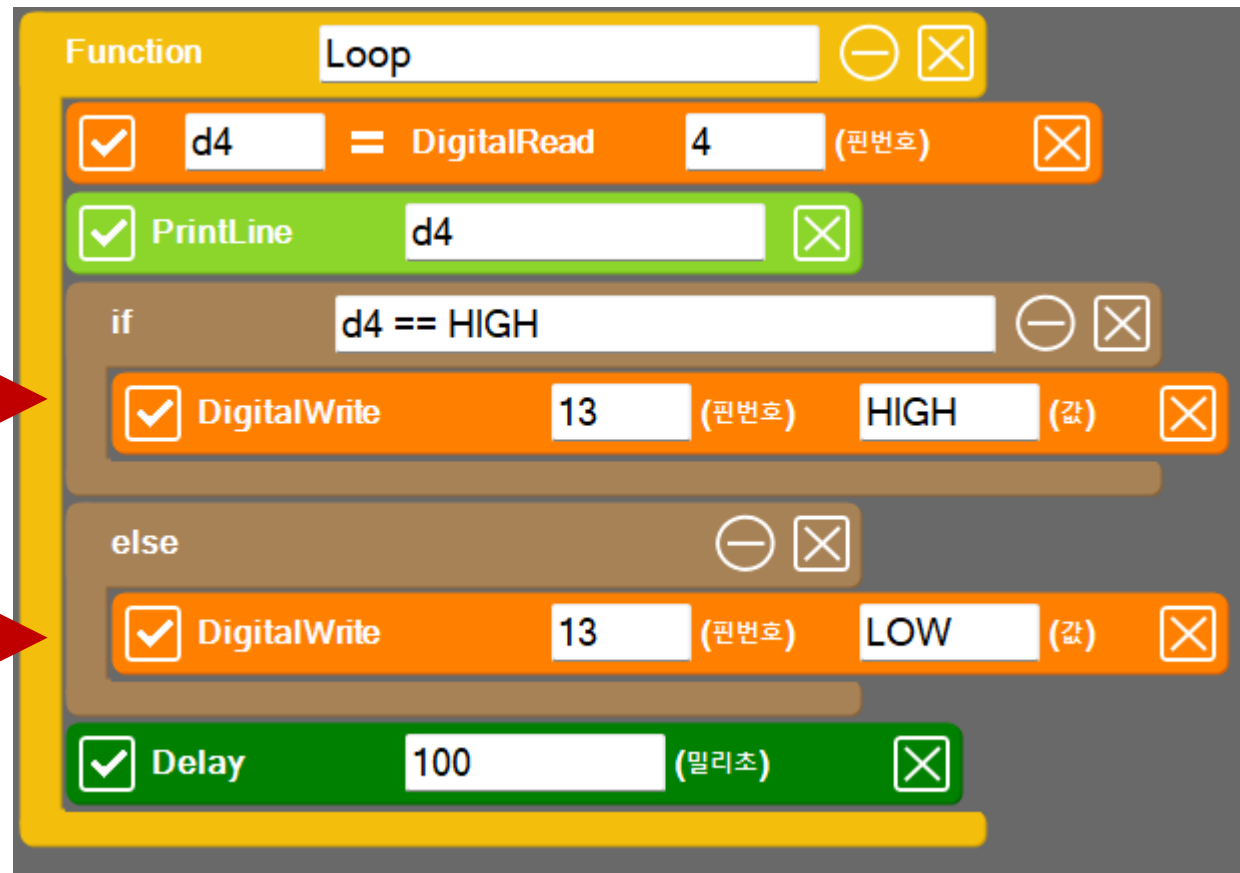
# LED 명령어 추가하기



if 블록 및 else 블록안에 각각 LED를 켜고 끄는 명령어를 추가합니다.

LED 켜기

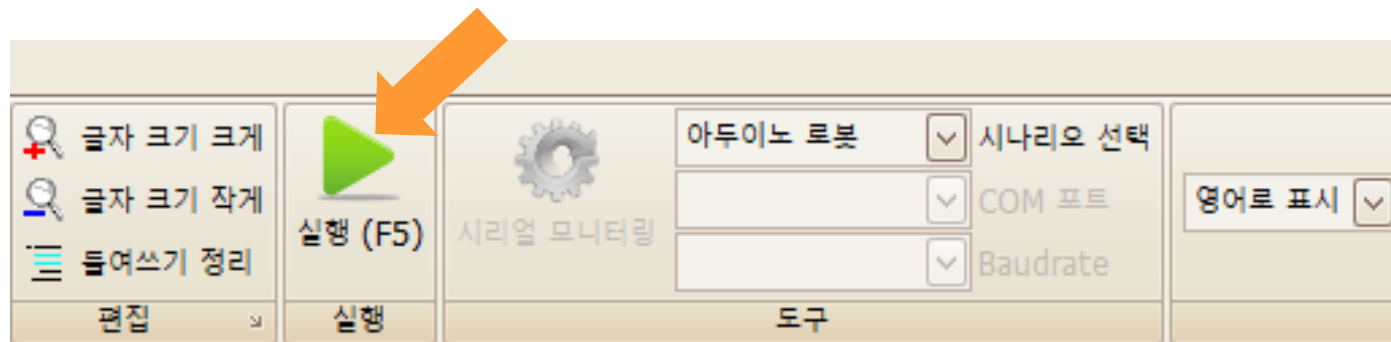
LED 끄기



# 실행하기



- 실행 버튼을 클릭합니다.



# 4번 핀 버튼으로 LED 제어하기



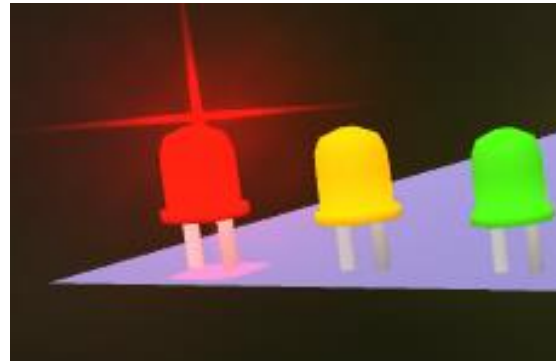
4번 핀에 연결된 버튼1을  
마우스로 클릭합니다.

# 버튼으로 LED 켜기



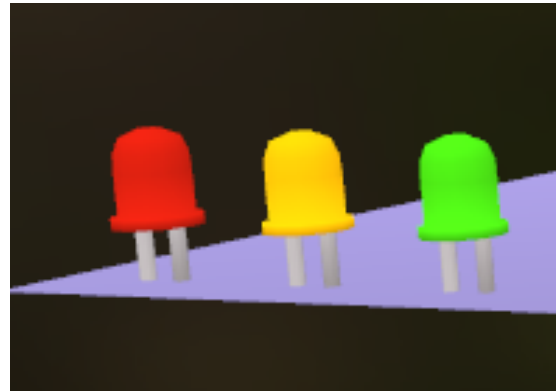
4번 핀에 연결된 버튼1을 누르면 13번 LED가 켜지고 그렇지 않으면 LED가 꺼지게 됩니다.

4번 핀에 연결된 버튼1을  
마우스로 클릭



13번 핀의 LED가  
켜짐

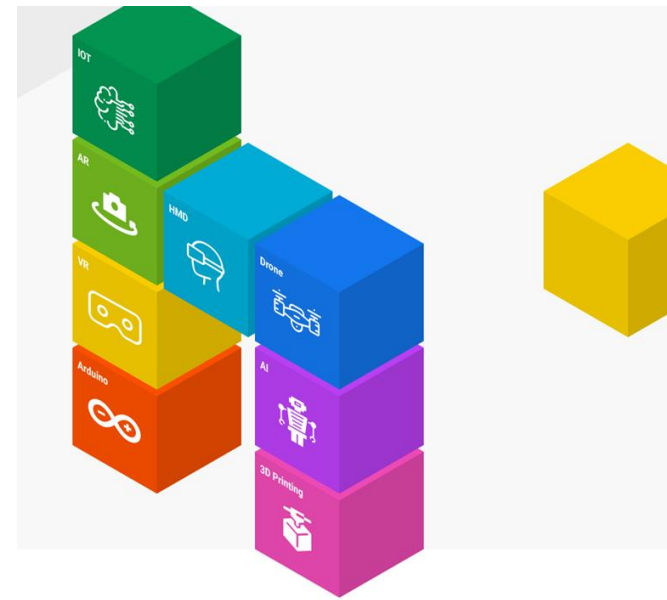
버튼을 클릭하지 않음



13번 핀의 LED가  
꺼짐



# LED 전체 켜기 실습



# LED 전체 켜기 실습

3개의 LED를 모두 켭니다.

The image shows a Scratch-like code editor with a yellow background. The code is as follows:

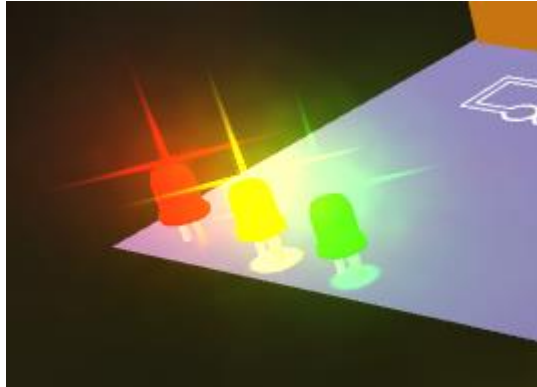
```
Function Loop
  [ ] d4 = DigitalRead 4 (핀번호)
  [ ] PrintLine d4
  if d4 == HIGH
    [ ] DigitalWrite 11 (핀번호) HIGH (값)
    [ ] DigitalWrite 12 (핀번호) HIGH (값)
    [ ] DigitalWrite 13 (핀번호) HIGH (값)
  else
    [ ] DigitalWrite 11 (핀번호) LOW (값)
    [ ] DigitalWrite 12 (핀번호) LOW (값)
    [ ] DigitalWrite 13 (핀번호) LOW (값)
  [ ] Delay 100 (밀리초)
```

The code is written in a Scratch-like block-based language. The 'Function' block is a yellow 'Loop' block. Inside the loop, there is a 'DigitalRead' block (orange) that reads the value of pin 4 into variable 'd4'. This is followed by a 'PrintLine' block (green) that prints the value of 'd4'. Then, there is an 'if' block (brown) that checks if 'd4 == HIGH'. If true, it executes three 'DigitalWrite' blocks (orange) that set pins 11, 12, and 13 to 'HIGH'. If false, it executes three 'DigitalWrite' blocks (orange) that set pins 11, 12, and 13 to 'LOW'. Finally, there is a 'Delay' block (green) that delays the execution for 100 milliseconds.

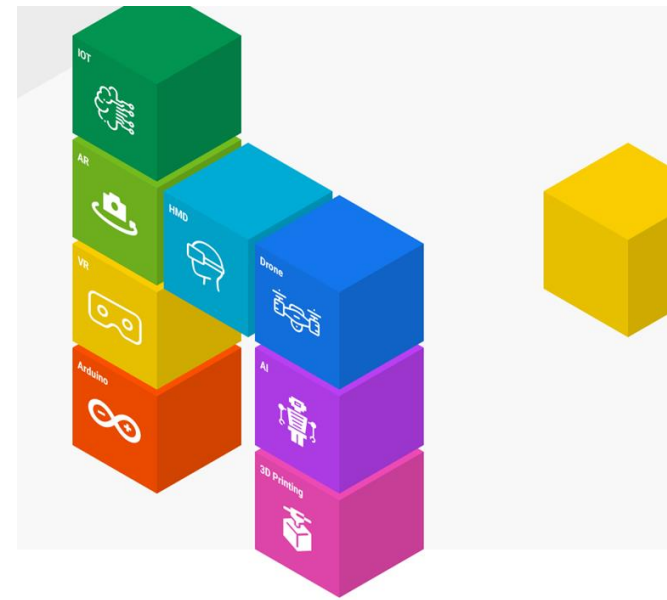
# LED 전체 켜기 실습



3개의 LED를 모두 켭니다.



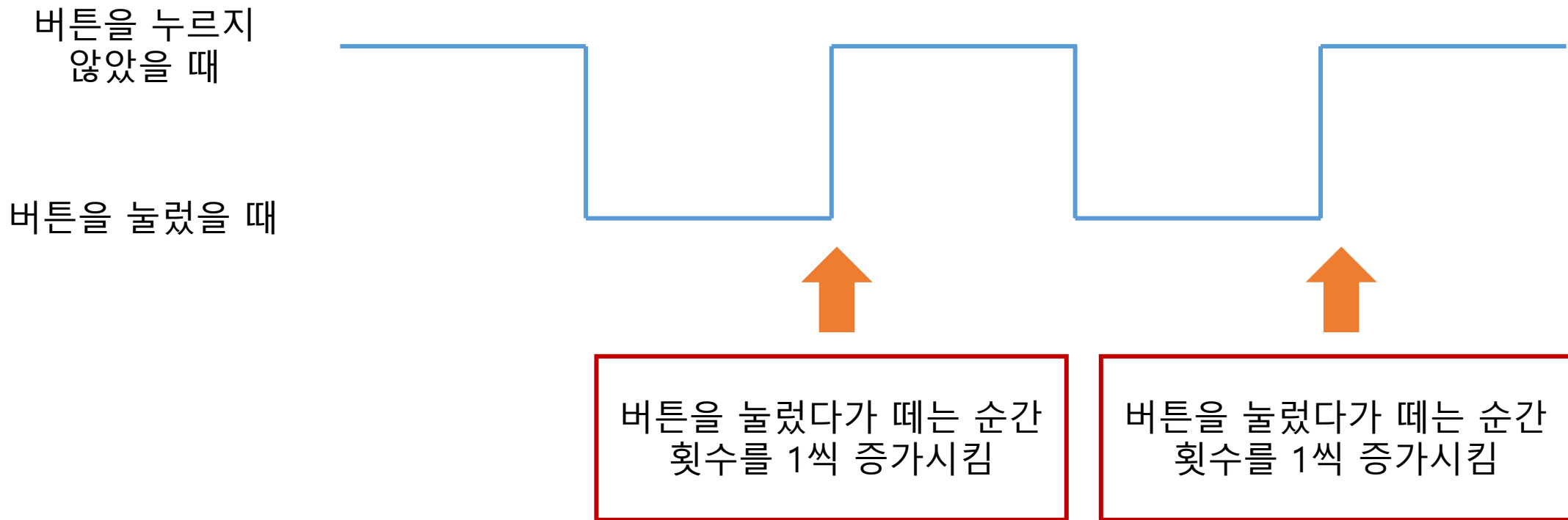
# 숫자 카운터 만들기



# 버튼으로 카운터 횡수 세기 기능



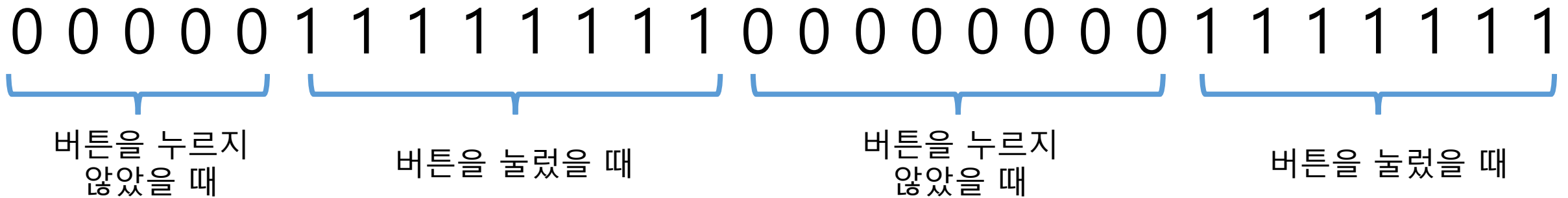
버튼을 한번 누를 때 마다 숫자가 1씩 증가하는 기능 만들기



# 버튼 읽기 상태값



버튼 상태값을 0.1초 간격으로 읽었을 때의 출력값



# 버튼 읽기 상태값



버튼 상태값을 0.1초 간격으로 읽었을 때의 출력값

버튼을 눌렀다가 떼는 순간  
버튼의 값은 1에서 0으로  
바뀜



0 0 0 0 0 1 1 1 1 1 1 1 **1** **0** 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1

버튼을 누르지  
않았을 때

버튼을 눌렀을 때

버튼을 누르지  
않았을 때

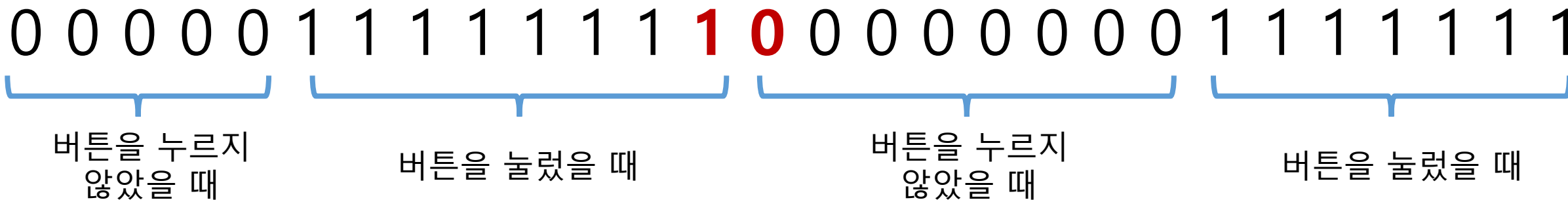
버튼을 눌렀을 때

# 버튼 읽기 상태값으로 횟수 증가시키기



버튼 상태값을 0.1초 간격으로 읽었을 때의 출력값

버튼의 상태값이 1에서 0으로  
바뀔때를 감지하여  
횟수를 1씩 증가시킴

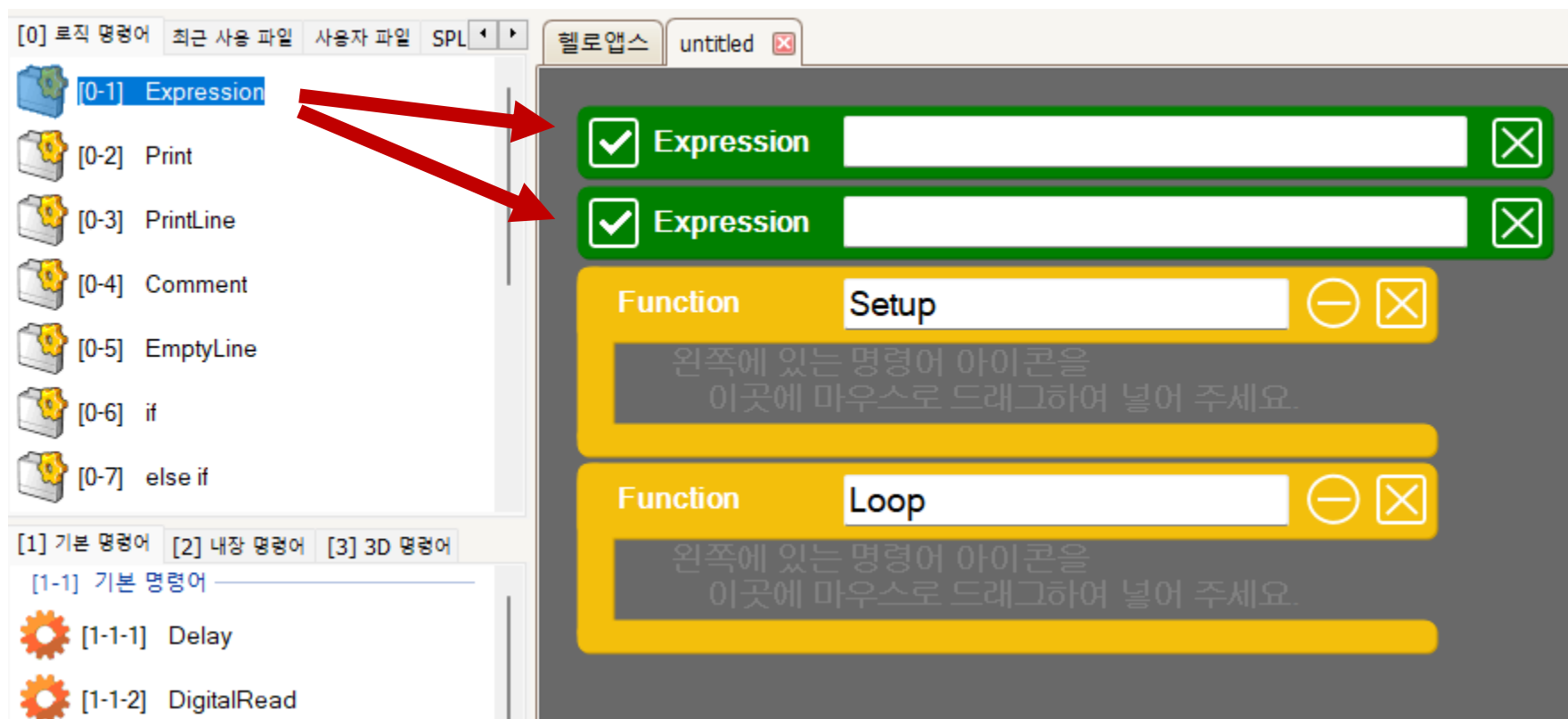




# 버튼 읽기 상태값으로 횟수 증가시키기



전역변수 선언을 위해 Expression 수식 명령어를 2개 추가합니다.



# 버튼 읽기 상태값으로 횟수 증가시키기



전역변수  $cnt = 0$ 과  $p = 0$ 을 입력해 줍니다.

cnt 횟수를 저장하는  
전역 변수

p 이전 버튼의 읽기  
값을 저장하고 있는  
전역변수

The image shows a Scratch-style code editor with the following blocks:

- Expression Block 1:** A green block with a checkmark icon, the label "Expression", and the text "cnt = 0". A red arrow points to the input field.
- Expression Block 2:** A green block with a checkmark icon, the label "Expression", and the text "p = 0". A red arrow points to the input field.
- Function Block 1:** A yellow block with the label "Function" and the text "Setup". It has a minus icon and a close icon.
- Function Block 2:** A yellow block with the label "Function" and the text "Loop". It has a minus icon and a close icon.

Below each Function block is a grey area with the text: "왼쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요."

# 버튼 읽기 상태값으로 횟수 증가시키기



0.1초 간격으로 4번 핀의 버튼 값을 읽어오는 명령어를 추가합니다.

The image shows a code editor with the following blocks:

- ☒ Expression `cnt = 0` [X]
- ☒ Expression `p = 0` [X]
- Function `Setup` [−] [X]
  - 원쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.
- Function `Loop` [−] [X]
  - ☒ `d4` `=` `DigitalRead` `4` (핀번호) [X]
  - ☒ Delay `100` (밀리초) [X]

Two red arrows point to the `DigitalRead` and `Delay` blocks in the `Loop` function.

# 버튼 읽기 상태값으로 횟수 증가시키기



if 블록을 추가한 후, 다음과 같이 조건을 입력합니다.

$p == 1$

And

$d4 == 0$

And 조건은 && 로 표시합니다.

$p == 1 \ \&\& \ d4 == 0$

Expression  $cnt = 0$

Expression  $p = 0$

Function Setup

Function Loop

$d4 = \text{DigitalRead } 4$  (핀번호)

if  $p == 1 \ \&\& \ d4 == 0$

Delay 100 (밀리초)

# 버튼 읽기 상태값으로 횟수 증가시키기



버튼을 눌렀다가  
떼는 순간 if 블록  
안쪽의 명령어가  
실행됨

이 곳에 횟수를 1씩  
증가시키는 명령어를  
추가합니다.

Expression ☒ cnt = 0

Expression ☒ p = 0

Function

왼쪽에 있는 명령어 아이콘을  
이곳에 마우스로 드래그하여 넣어 주세요.

Function

☒ d4 = DigitalRead 4 (핀번호)

if

왼쪽에 있는 명령어 아이콘을  
이곳에 마우스로 드래그하여 넣어 주세요.

☒ Delay 100 (밀리초)

# 버튼 읽기 상태값으로 횟수 증가시키기



2개의 Expression 명령어를  
그림과 같이 추가합니다.

$cnt = cnt + 1$

횟수를 1씩 증가시킴

$p = d4$

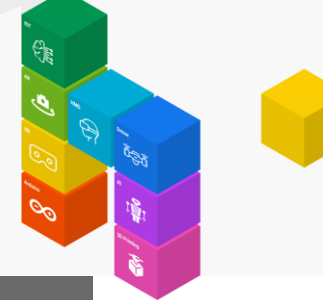
버튼 읽기 값을 p에 저장함

The image shows a Scratch script editor with the following code:

- Expression**  $cnt = 0$
- Expression**  $p = 0$
- Function** **Setup**
  - 원쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.
- Function** **Loop**
  - Expression**  $d4 = \text{DigitalRead } 4$  (핀번호)
  - if**  $p == 1 \ \&\& \ d4 == 0$ 
    - Expression**  $cnt = cnt + 1$
    - Expression**  $p = d4$
    - Delay** 100 (밀리초)

Red arrows point from the text boxes on the left to the corresponding code blocks in the Loop function.

# 버튼 읽기 상태값으로 횟수 증가시키기



횟수를 출력하는 명령어를  
추가합니다.

The code editor displays the following blocks:

- Expression** `cnt = 0`
- Expression** `p = 0`
- Function** `Setup`
  - 원쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.
- Function** `Loop`
  - Expression** `d4 = DigitalRead 4` (핀번호)
  - if** `p == 1 && d4 == 0`
    - Expression** `cnt = cnt + 1`
    - PrintLine** `cnt`
  - Expression** `p = d4`
  - Delay** `100` (밀리초)

# 실행하기

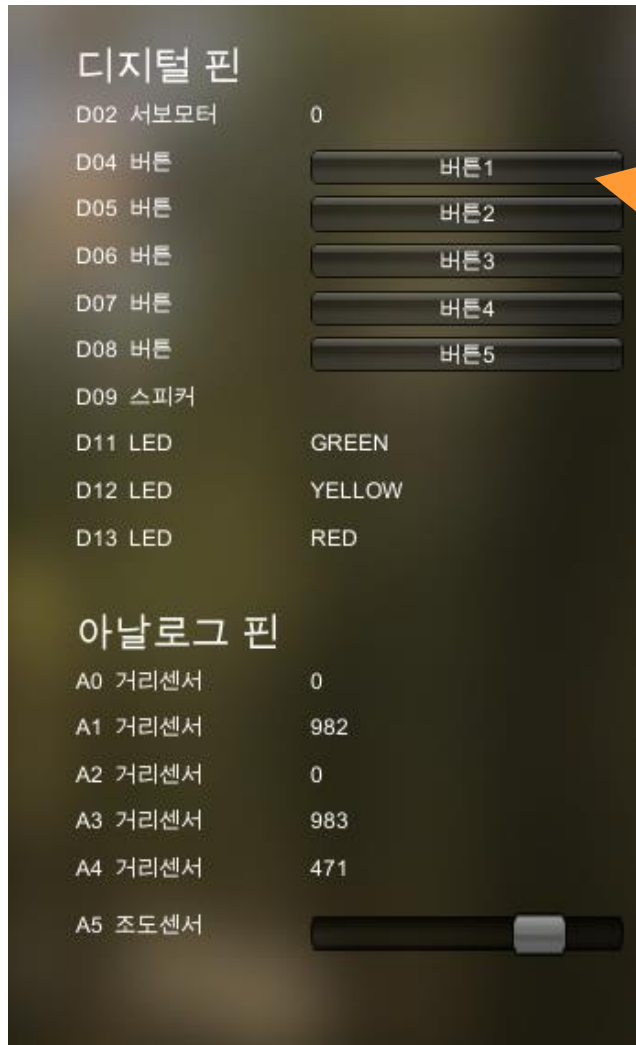


- 실행 버튼을 클릭합니다.





# 버튼 읽기 상태값으로 횟수 증가시키기

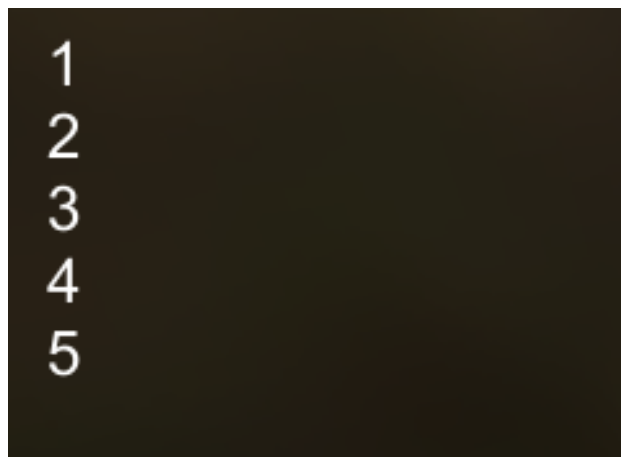


4번 핀에 연결된 버튼1을  
마우스로 클릭합니다.

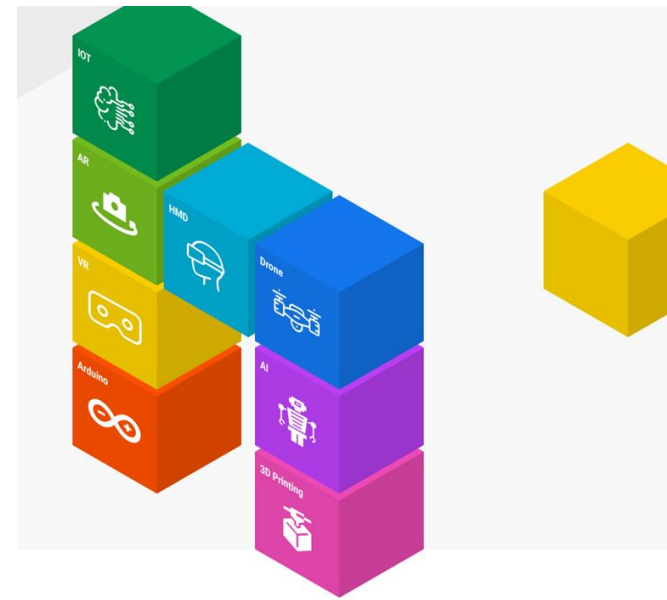
# 버튼 읽기 상태값으로 횟수 증가시키기



버튼을 클릭할 때 마다 숫자가 1씩 증가하여 화면에 출력됩니다.



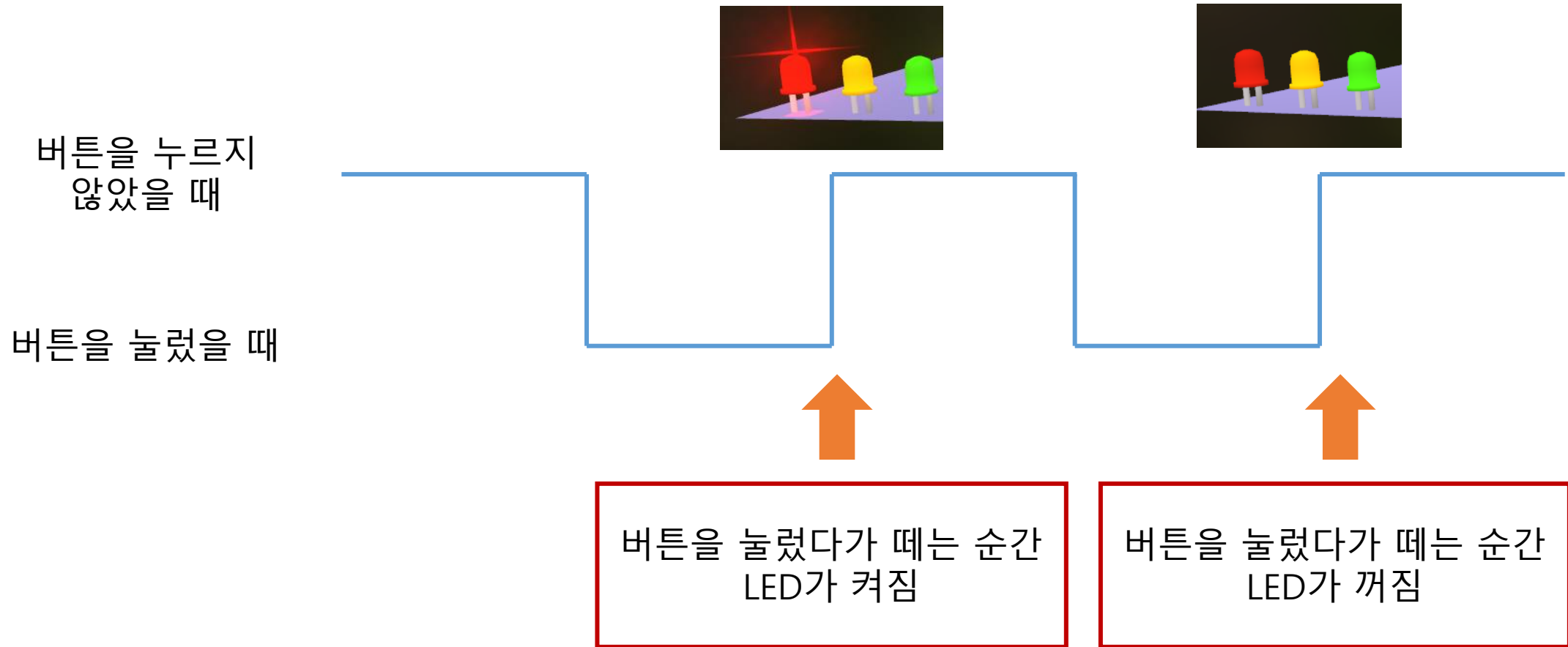
# 토글 스위치 구현하기



# 토글 스위치 구현하기



버튼을 눌렀다가 떼면 LED가 켜지고, 다시 버튼을 눌렀다가 떼면 LED가 꺼지도록 함



# 버튼 읽기 상태값으로 LED 상태 바꾸기



버튼 상태값을 0.1초 간격으로 읽었을 때의 출력값

버튼의 상태값이 1에서 0으로  
바뀔때를 감지하여  
LED 상태를 바꿈



0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1

버튼을 누르지  
않았을 때

버튼을 눌렀을 때

버튼을 누르지  
않았을 때

버튼을 눌렀을 때

# 버튼 읽기 상태값으로 LED 상태 바꾸기



이전 예제에서 전역변수 cnt를  
on으로 수정

이곳에 LED 상태를 변경하는  
코드를 추가합니다.

✓ Expression `on = 0` ✕

✓ Expression `p = 0` ✕

Function `Setup` ⓪ ✕

원쪽에 있는 명령어 아이콘을  
이곳에 마우스로 드래그하여 넣어 주세요.

Function `Loop` ⓪ ✕

✓ `d4` `=` `DigitalRead` `4` (핀번호) ✕

if `p == 1 && d4 == 0` ⓪ ✕

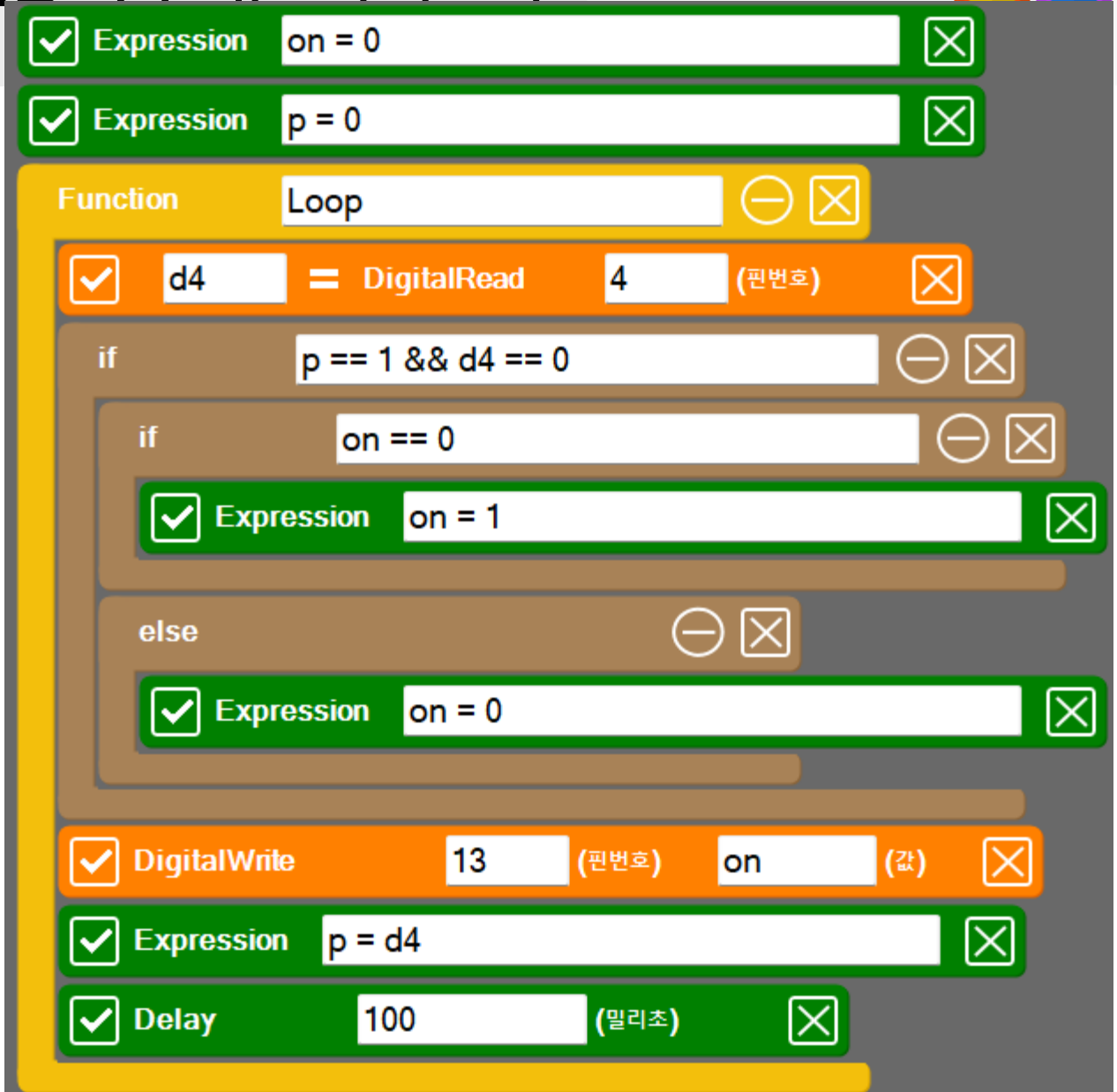
원쪽에 있는 명령어 아이콘을  
이곳에 마우스로 드래그하여 넣어 주세요.

✓ Expression `p = d4` ✕

✓ Delay `100` (밀리초) ✕

# 버튼 읽기 상태값으로 LED 상태 바꾸기

if 명령어로 값을  
비교해 봅니다.



The image shows a Scratch script designed to control an LED based on a button's state. The script is as follows:

- Expression** (Green): `on = 0`
- Expression** (Green): `p = 0`
- Function** (Yellow): `Loop`
- Code** (Orange): `d4 = DigitalRead 4` (핀번호)
- if** (Brown): `p == 1 && d4 == 0`
  - if** (Brown): `on == 0`
    - Expression** (Green): `on = 1`
  - else** (Brown):
    - Expression** (Green): `on = 0`
- DigitalWrite** (Orange): `13` (핀번호), `on` (값)
- Expression** (Green): `p = d4`
- Delay** (Green): `100` (밀리초)

# 버튼 읽기 상태값으로 LED 상태 바꾸기



나머지 연산자 % 를 이용하여  
on 값이 0이면 1로, 1이면  
0으로 수정되는 수식을  
추가합니다.

The code editor shows the following blocks:

- Expression** `on = 0`
- Expression** `p = 0`
- Function** `Setup`
  - 원쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.
- Function** `Loop`
  - Variable** `d4` **=** **DigitalRead** `4` (핀번호)
  - if** `p == 1 && d4 == 0`
    - Expression** `on = (on + 1) % 2` (A red arrow points to this block from the text box on the left.)
    - Expression** `p = d4`
    - Delay** `100` (밀리초)



# 버튼 읽기 상태값으로 LED 상태 바꾸기

변수 on은 0과 1 값 만을 가지기  
때문에 이 값으로 LED를 켜거나  
끄 수 있음

변수 on 값을 이용하여  
LED를 켜고 끄는 명령어를  
추가합니다.

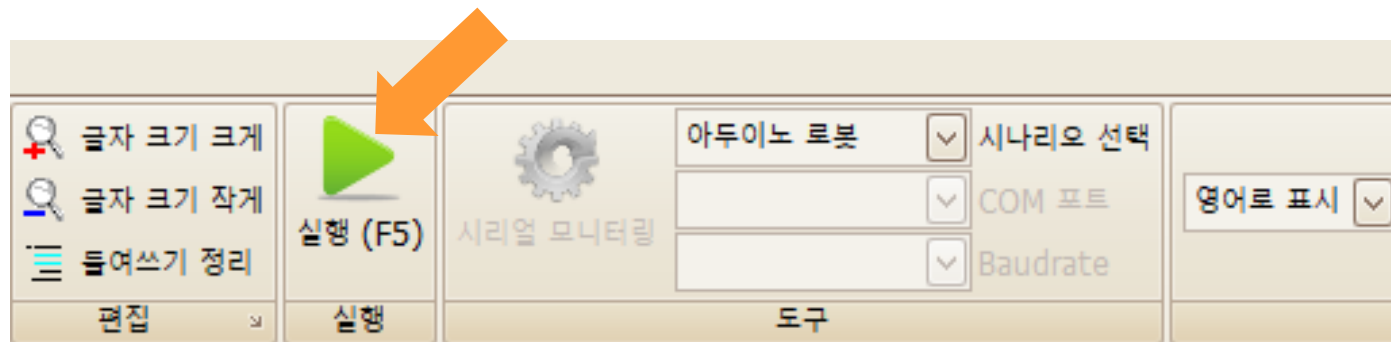
Scratch-like block diagram showing the logic for controlling an LED based on a button state.

- Expression** `on = 0`
- Expression** `p = 0`
- Function** `Setup`
  - 왼쪽에 있는 명령어 아이콘을 이곳에 마우스로 드래그하여 넣어 주세요.
- Function** `Loop`
  - DigitalRead** `d4 = DigitalRead 4` (핀번호)
  - if** `p == 1 && d4 == 0`
    - Expression** `on = (on + 1) % 2`
  - DigitalWrite** `13` (핀번호) `on` (값)
  - Expression** `p = d4`
  - Delay** `100` (밀리초)

# 실행하기



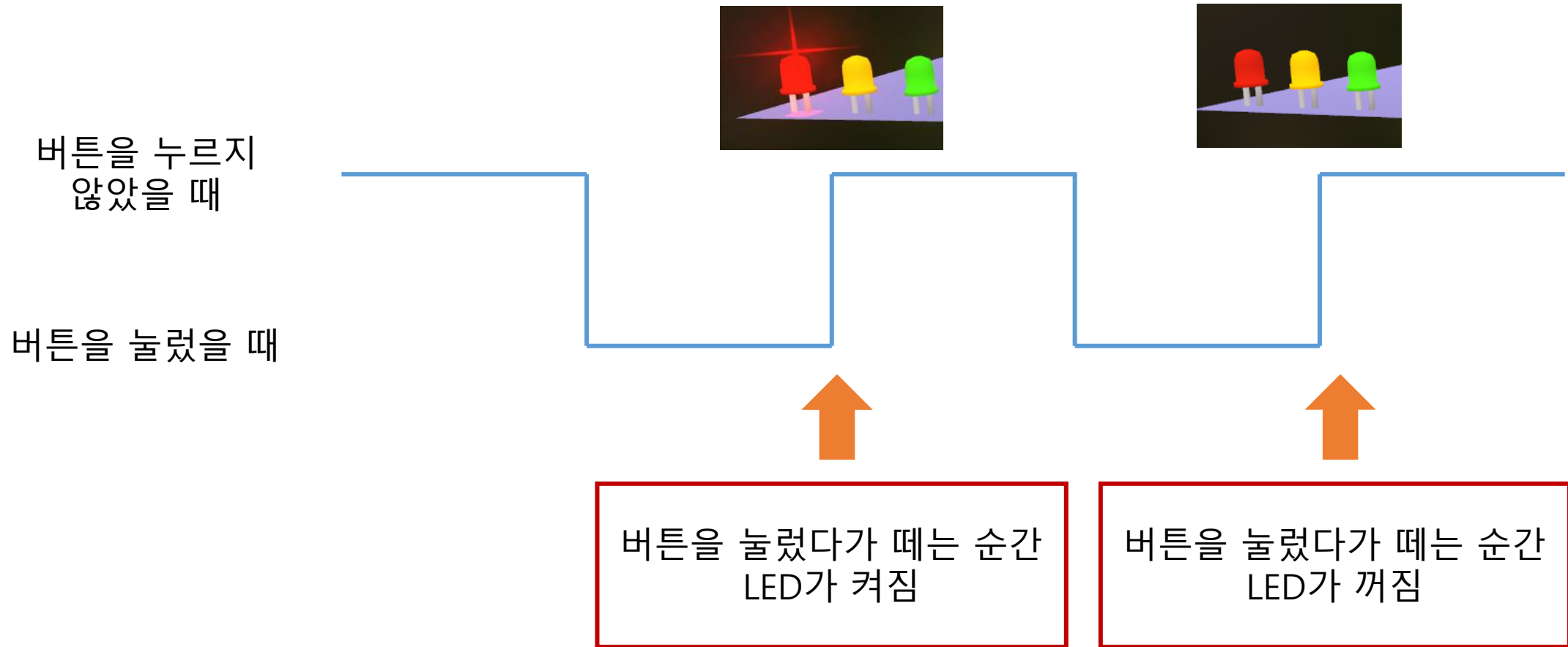
- 실행 버튼을 클릭합니다.



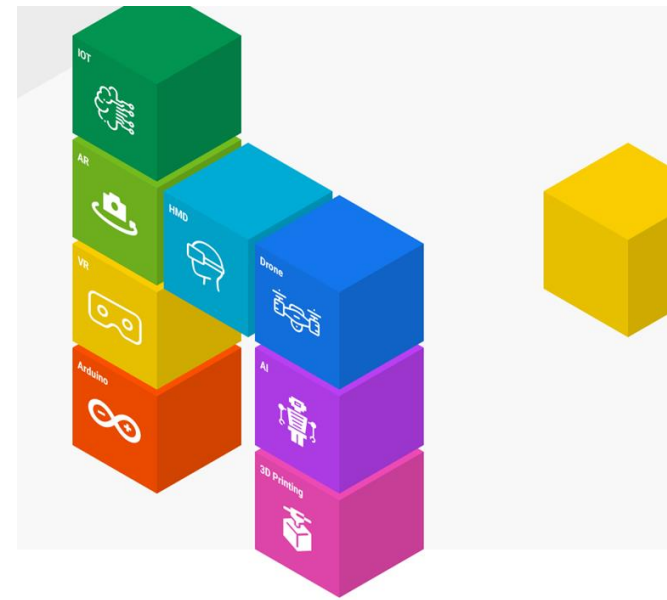
# 토글 스위치 구현하기



버튼을 눌렀다가 떼면 LED가 켜지고, 다시 버튼을 눌렀다가 떼면 LED가 꺼짐



# 토글 스위치로 3개 LED 켜기



# 3개 LED 켜기

The image shows a Scratch-like code editor with a script designed to turn on three LEDs. The script is as follows:

- Expression** `on = 0`
- Expression** `p = 0`
- Function** `Loop`
- Block** `d4 = DigitalRead 4 (핀번호)`
- if** `p == 1 && d4 == 0`
  - if** `on == 0`
    - Expression** `on = 1`
  - else**
    - Expression** `on = 0`
- DigitalWrite** `11 (핀번호) on (값)`
- DigitalWrite** `12 (핀번호) on (값)`
- DigitalWrite** `13 (핀번호) on (값)`
- Expression** `p = d4`
- Delay** `100 (밀리초)`

# 토글 스위치로 LED 전체 켜기 실습



3개의 LED를 모두 켭니다.

