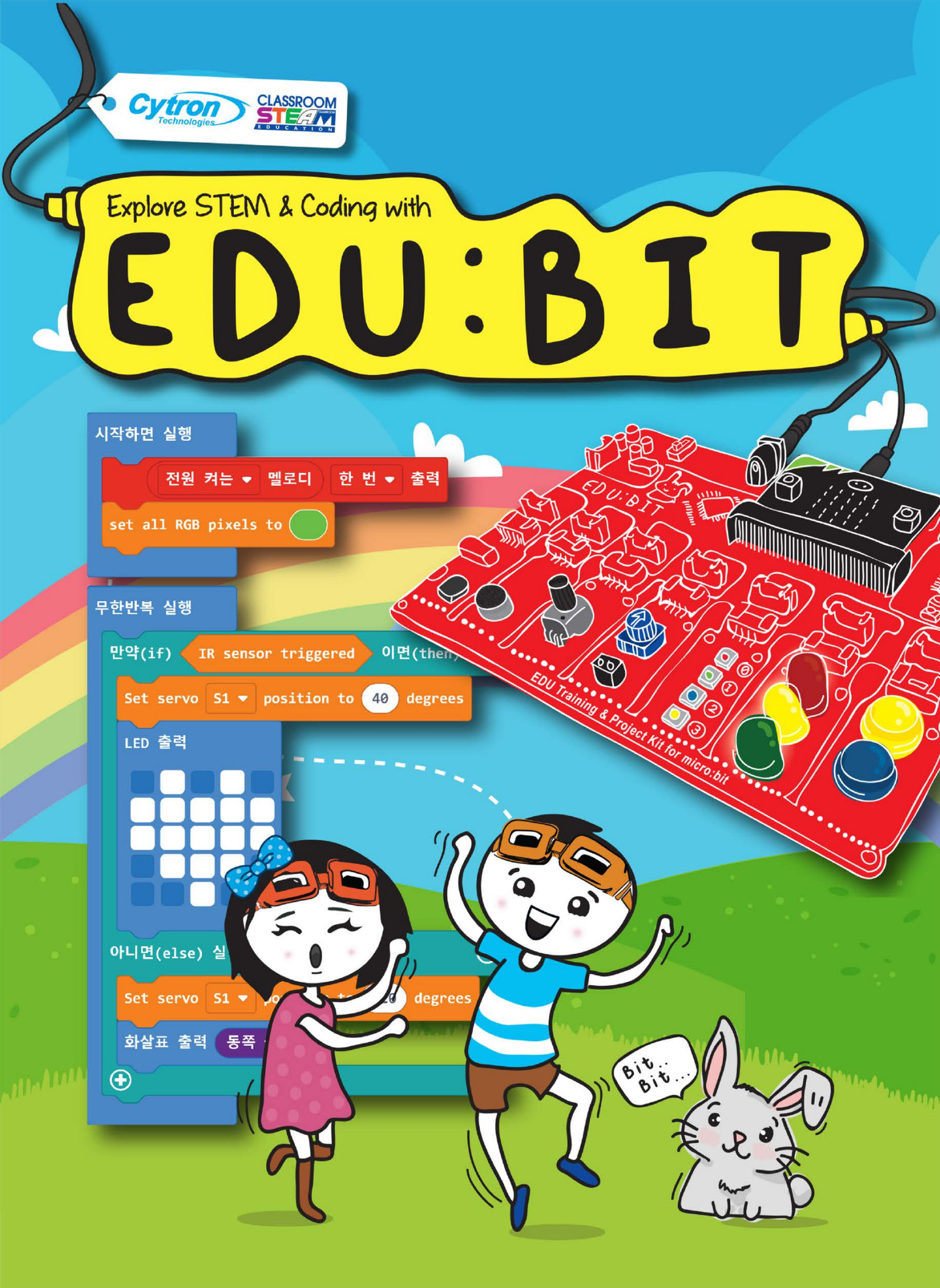
Cytron Technologies

CLASSROOM STEAM EDUCATION

Explore STEM & Coding with

# EDU:BIT



# RERO EDUTEAM @ CYTRON으로 부터 편지가 왔어요~

_____ 에게,

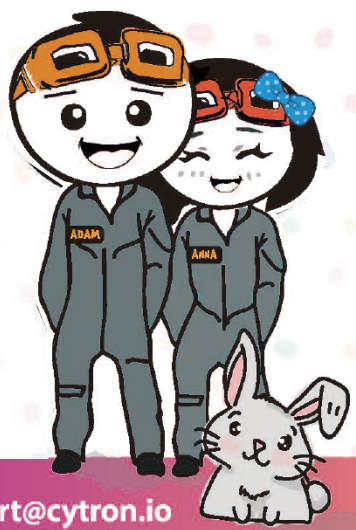
micro:bit에 대해 들어본적이 있나요?
micro:bit는 프로그래밍 할 수 있는 작은 보드로, 아이들이 쉽고 재미있게 코딩을 배울 수 있도록 영국에서 만들어져 전 세계에 널리 보급되었습니다.

Cytron의 기술자들은 EDU:BIT 보드를 만듦으로써
여러분이 프로그래밍을 조금씩 배울 수 있게 하겠다는 목표에 한 발짝 다가섰습니다.
이 키트에는 피에조 부저와 오디오 잭이 있어 음악을 재생할 수 있는 Music Bit,
소리를 감지할 수 있는 Sound Bit, 아날로그 제어를 위한 Potentio Bit, 물체를 탐지하는 IR Bit,
다양한 색의 빛을 출력하는 RGB Bit, 빨간색, 노란색 및 초록색 LED가 있는 Traffic Light Bit,
마지막으로 micro:bit 보드 푸쉬 버튼의 큰 버전인 Button Bit가 있습니다.
보드에서 DC모터와 서보모터도 함께 사용할 수 있습니다. 정말 멋지죠?!

앞으로 가위바위보, 뱀과 사다리 게임, 나 잡아봐라~, 장기자랑, 트위스터, Simon Says 같은 게임을 만들어보고 더 재미있게 수정해 볼 거예요. 친구들과 직접 만든 게임을 해보는 시간도 준비되어 있습니다.
차근차근 알려줄테니 걱정 말고 따라오세요~

각 챕터가 끝날 때마다 배운 내용을 적용해서 응용 프로그램을 만드는 과제가 있습니다.
여러분을 돕기 위해 항상 곁에 있을 테니 어려워도 포기하지 말고 한번 해보세요.

준비됐나요?
자 이제 신나는 여정을 떠나봅시다!




그럼 안녕,
Adam과 Anna가

# EDU:BIT를 이용한 STEM 및 코딩 교육용 트레이닝&프로젝트 키트

저자 Cheryl Ng, SC Lim & Adrian Teo
일러스트 Suhana Oazmi

번역 디바이스마트

2020

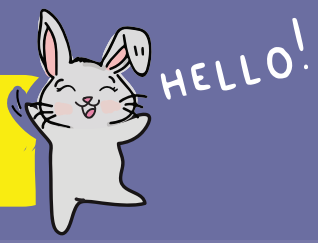
Published by Cytron Technologies

DEVICE MART

번　역 : 디바이스마트
펴낸곳 : 디바이스마트
주소 : 인천광역시 미추홀구 염전로 324
고객센터 : 070-7019-8887　　팩스 : 02-6008-4953
www.devicemart.co.kr

EDU:BIT Kit
보러가기

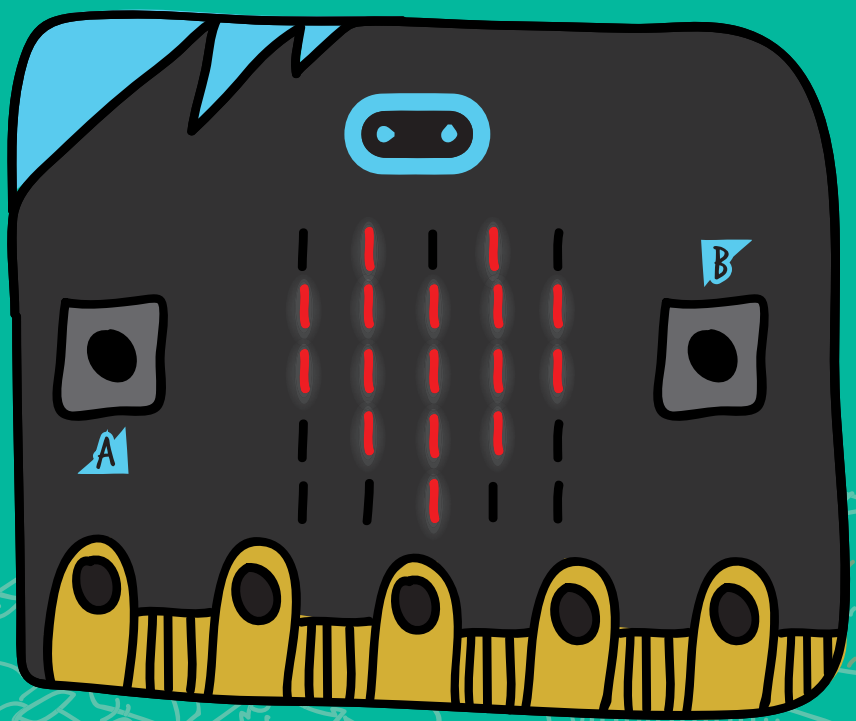
# 목 차



# CHAPTER 1

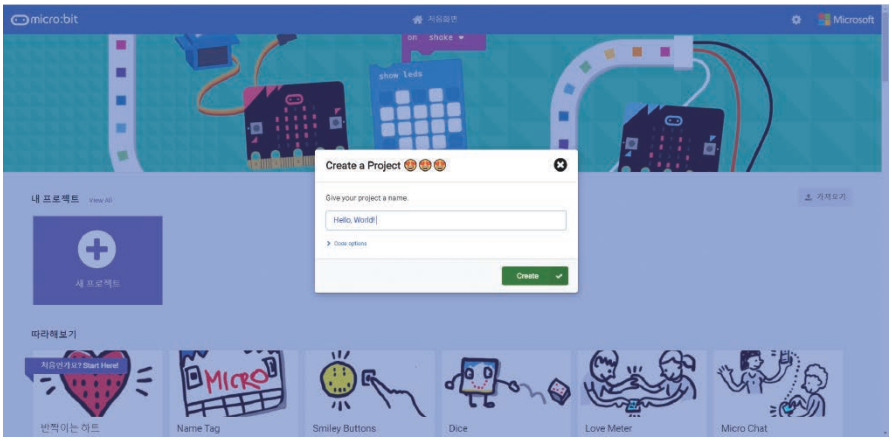
# > Hello, World!_

micro:bit의 LED 매트릭스

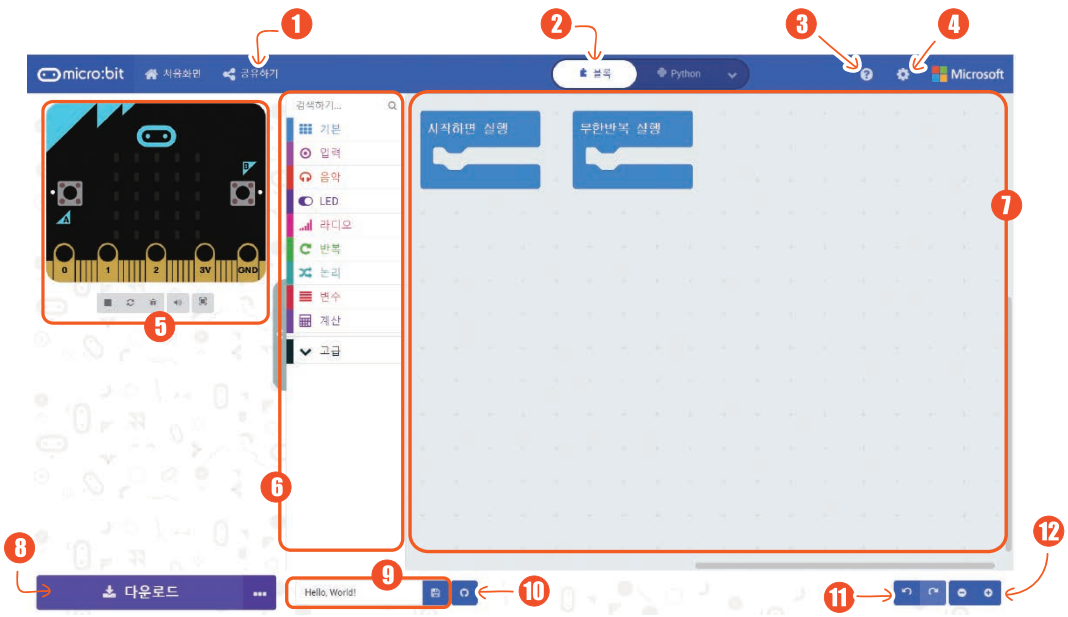



## 코딩을 시작해봅시다!

Step1 브라우저를 열고 https://makecode.microbit.org/에 접속합니다.
'새 프로젝트'를 누릅니다. 프로젝트 이름을 입력하고 'Create'를 클릭합니다.

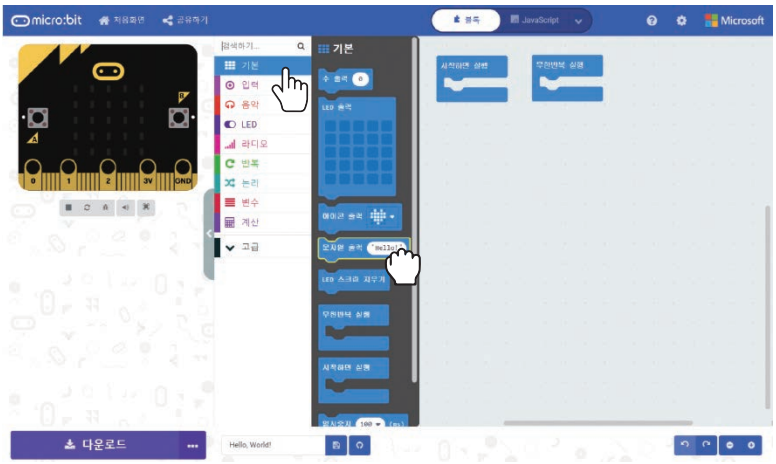



Microsoft MakeCode Editor 페이지에서 드래그해서 놓는 방식으로 쉽게 EDU:BIT를 프로그래밍 할 수 있습니다.




1) 프로젝트 게시 및 공유
2) 블록, JavaScript, Python중에서 프로그램 선택
3) 도움말 메뉴 열기
4) 설정 변경, 확장 프로그램, 장치 페어링
5) **시뮬레이터** – 프로그램이 micro:bit에서 어떻게 작동하는지 보여줍니다.
6) **도구상자 / 카테고리 서랍** – 각 카테고리에서 사용 가능한 코딩 블록들을 보기 위해 클릭합니다. 같은 카테고리의 모든 블록들은 같은 색으로 칠해져 있습니다.
7) **프로그래밍 작업공간** – 프로그램은 이 공간에 블록들이 함께 딱 맞춰짐으로써 구성됩니다.
8) 코드를 micro:bit에 다운로드하기
9) 컴퓨터에 현재 프로젝트 저장하기
10) GitHub repository 만들기
11) 이전상태 / 다시복구
12) 작게보기 / 크게보기

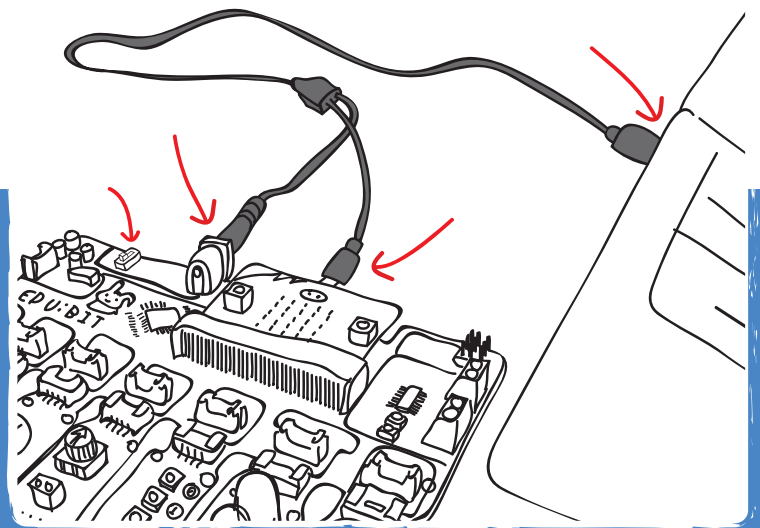
Step2 [ 기본 ]을 클릭하고 [ 문자열 출력 ] 블록을 선택합니다.



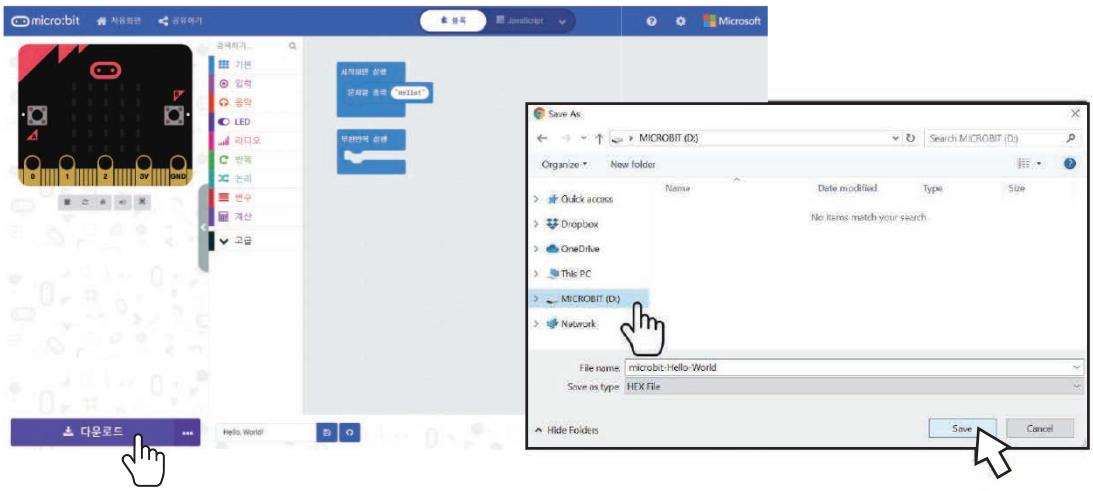

Step3 [ 문자열 출력 ] 블록을 클릭하고 [ 시작하면 실행 ] 슬롯에 맞추어 넣습니다.



Step4 보이는 것과 같이 USB 케이블을 컴퓨터와 EDU:BIT에 연결합니다.
스위치를 ON으로 옮겨 EDU:BIT의 전원을 켤 수 있습니다.



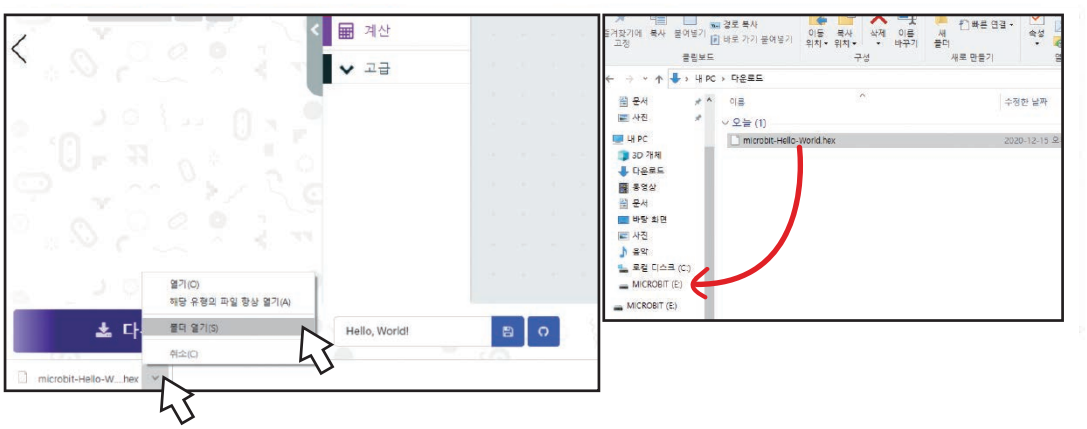
Step5 [ **다운로드** ] 버튼을 클릭합니다. 팝업창에서 MICROBIT 드라이브를 선택해 프로젝트를 저장합니다. '다운로드 완료'가 뜨면 창을 닫습니다.



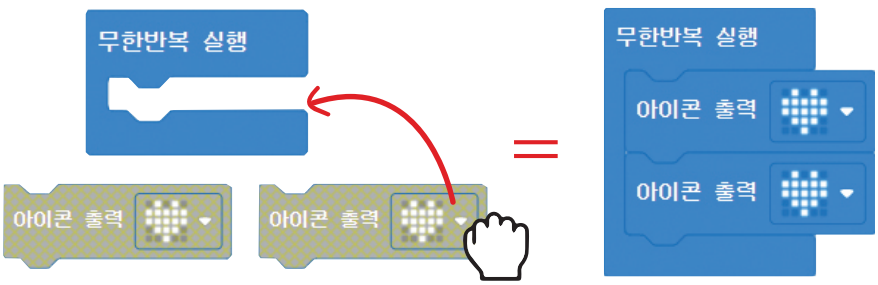

이러한 코드 전송 과정을 **플래시**라고 합니다.
옮겨지는 동안 micro:bit의 뒤에 있는 주황색 LED는
반짝거리고 일단 완료되면 그 코드는 자동적으로 실행됩니다.

## 주 목 !

만약 팝업창이 보이지 않는다면, 브라우저가 다운로드를 저장하도록 설정된 위치에 파일이 자동으로 다운로드 되었음을 의미합니다. 창 아래쪽에 보이는 다운로드된 .hex 파일을 클릭하고 '폴더 열기'를 선택합니다. 플래시 드라이브에 파일을 복사하는 것처럼, 다운로드된 'microbit-xxxx.hex' 파일을 클릭하고 MICROBIT 드라이브로 드래그합니다.



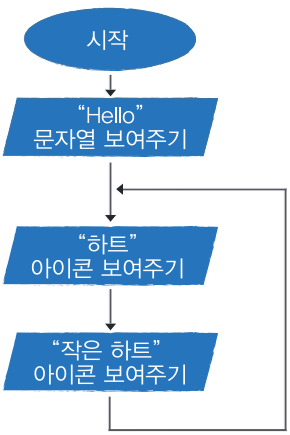
Step6 [ 기본 ]을 클릭하고 [ 아이콘 출력 ] 블록을 클릭합니다. 반복해서 다른 [ 아이콘 출력 ] 블록을 추가합니다. [ 아이콘 출력 ] 블록들을 클릭해서 [ 무한반복 실행 ] 슬롯에 맞추어 넣습니다.




Step7 두 번째 [ 아이콘 출력 ] 블록에서 아이콘을 클릭하고 팝업창에서 '작은 하트' 디자인을 선택합니다. EDU:BIT에 코드를 플래시합니다.

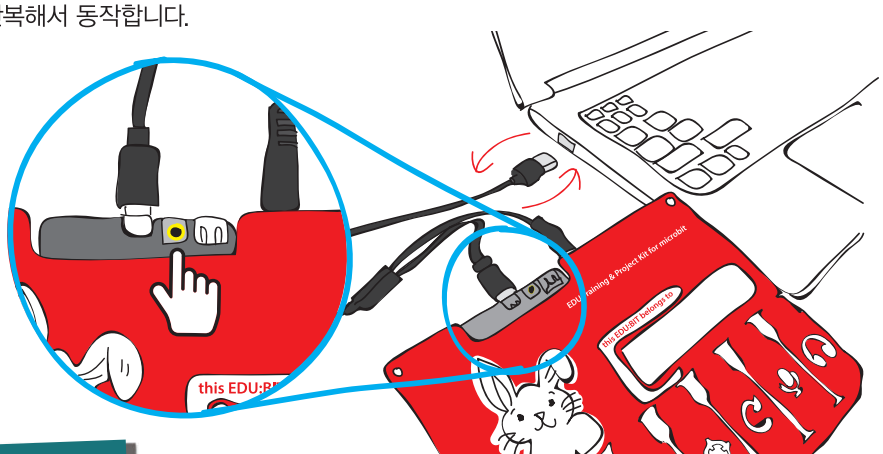

# 코딩 정복하기


"HELLO!" 글자는 단 한 번 지나간 반면
하트 모양 아이콘들은 계속 반복된다는 것을 알아챘나요?
왜일까요?





[ 시작하면 실행 ] 블록은 프로그램이 시작되면 코드가 한 번 동작합니다. [ 무한반복 실행 ] 블록은 코드가 계속 반복해서 동작합니다.



**주 목 !**

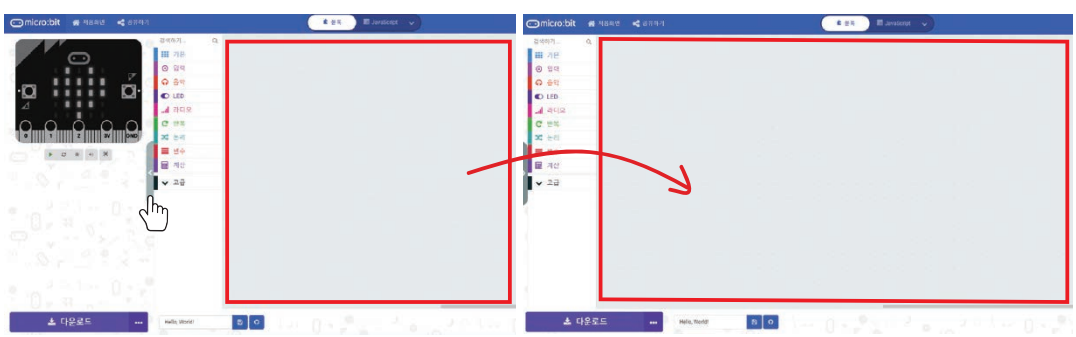
만약 프로그램을 처음부터 다시 시작하고 싶다면, 보드 리셋을 위해 간단하게 리셋 버튼을 누르거나 USB 케이블을 뺐다가 다시 꽂습니다.

# Quick Tip #1

지우고 싶은 블록은 클릭해서 툴박스 영역으로 드래그하면 삭제할 수 있습니다.
쓰레기통 아이콘이 나타났을 때 블록을 놓으면 지워집니다.
또는 마우스 오른쪽 버튼으로 블록을 클릭해서 '블록 삭제'를 선택해도 됩니다.

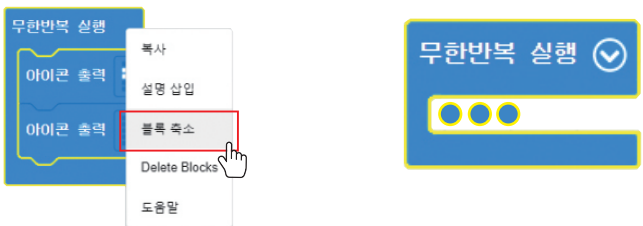

# Quick Tip #2

만약 시뮬레이터 창을 사용하지 않는다면 탭을 클릭해서 숨길 수 있고, 블록 코딩을 위한 더 넓은 공간이 생깁니다.



# Quick Tip #3

합쳐진 블록을 마우스 오른쪽 버튼으로 클릭하고 '블록 축소'를 선택함으로써 블록을 축소할 수 있습니다.
축소된 블록을 확장시키려면 간단하게 ⊘ 아이콘을 클릭하면 됩니다.




# Quick Tip #4

프로젝트를 게시하고 URL을 보냄으로써 선생님과 친구들에게 코드를 공유할 수 있습니다.
화면 위쪽의 [ **공유하기** ]를 클릭하고, 팝업창에서 [ **프로젝트 게시하기** ]를 누릅니다.
프로젝트 URL이 있는 새로운 팝업창이 뜨면, 프로젝트 URL을 복사하기 위해 [ **복사** ] 버튼을 클릭합니다.



# Quick Tip #5

선생님이나 친구들은 브라우저에서 프로젝트 URL을 열면 다음과 같은 페이지를 볼 수 있습니다.
코드를 볼 수 있고, [ **Edit Code** ] 버튼을 누름으로써 편집할 수 있습니다.

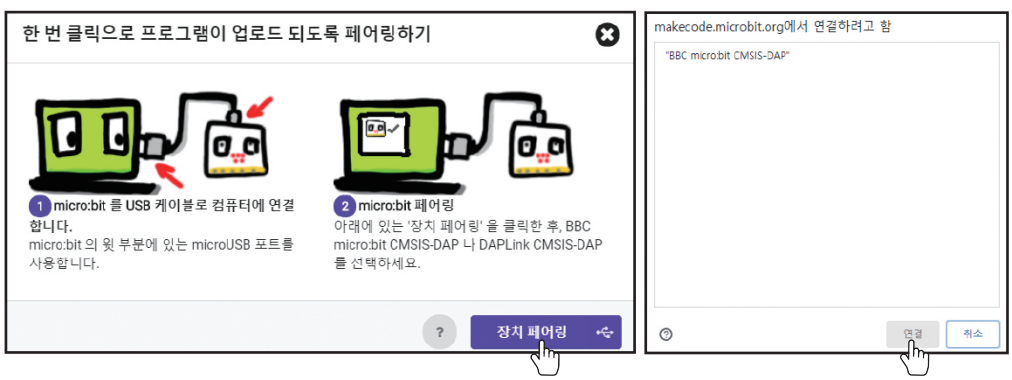
# Quick Tip #6

장치 페어링을 하면 한 번에 다운로드가 가능하다는 것을 알고 있었나요?
톱니바퀴 아이콘을 클릭하고 '장치 페어링'을 선택합니다.


**주 목 !**

micro:bit 장치에 최신 펌웨어가 설치되어 있어야 하며, New Edge 또는 Chrome 브라우저를 사용해야 합니다. 펌웨어를 업데이트해야 하는 경우 다음 설명을 따릅니다.
– https://microbit.org/get-started/user-guide/firmware/

EDU:BIT를 PC에 연결하고 팝업창에서 [ **장치 페어링** ] 버튼을 클릭합니다. 다음으로,
목록에서 BBC micro:bit CMSIS-DAP 또는 DAPLink CMSIS-DAP을 선택하고 [ **연결** ]을 클릭합니다.




장치를 페어링한 후, [ **다운로드** ] 버튼을 클릭하면 직접적으로 코드를 EDU:BIT에 플래시할 수 있습니다.
한번 해봅시다!

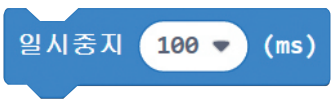
만약 장치를 페어링하는데 문제가 있다면, 더 많은 정보를 위해
https://makecode.microbit.org/device/usb/webusb/troubleshoot 을
참고합니다.

# 더 많은 블록 탐구하기

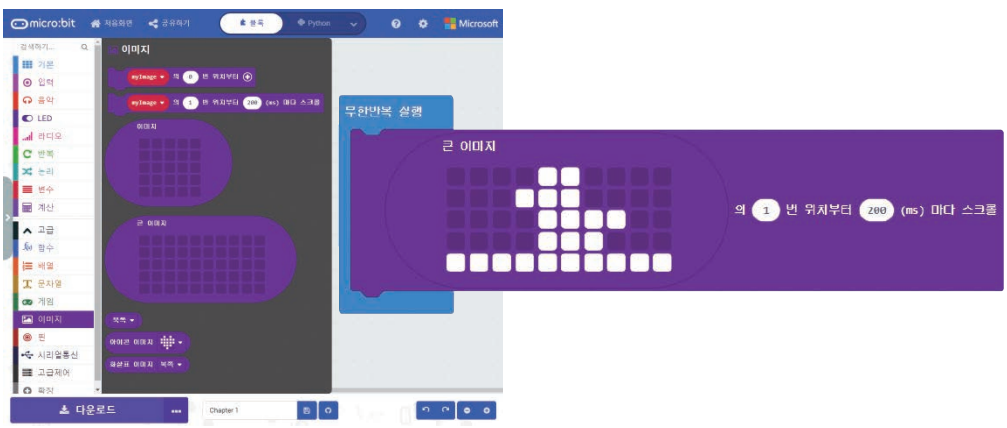
#1 나만의 아이콘을 디자인해서 보여줄 때는 [ LED 출력 ] 블록을 사용하고 숫자를 보여줄 때는 [ 수 출력 ] 블록을 사용합니다.


**알고 있나요~?**
1000 밀리초 (ms) = 1 초

#2 프로그램 속도를 늦추기 위해 [ 일시중지 ] 블록을 추가합니다. 이 기능은 설정한 밀리초(ms)만큼 프로그램이 잠시 멈춥니다.



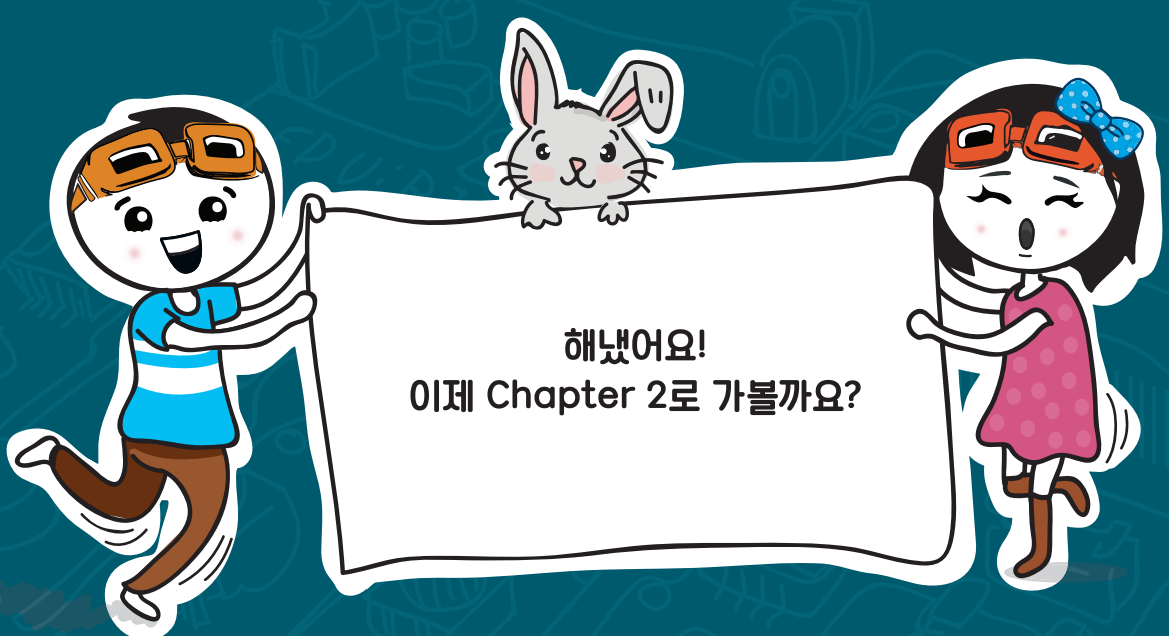

#3 LED 매트릭스 디스플레이를 가로지르는 이미지를 출력하려면, [ 이미지 ] 카테고리 서랍에 있는 [ _ 의 _ 번 위치부터 _ (ms) 마다 스크롤 ] 블록과 [ 이미지 ] 블록 또는 [ 큰 이미지 ] 블록을 함께 사용합니다. [ 이미지 ] 카테고리는 [ **고급** ] 카테고리를 누르면 볼 수 있습니다.




프로그램을 동작시키면 작은 오리들이 잇따라 가는 것을 볼 수 있습니다.

# 응용 과제

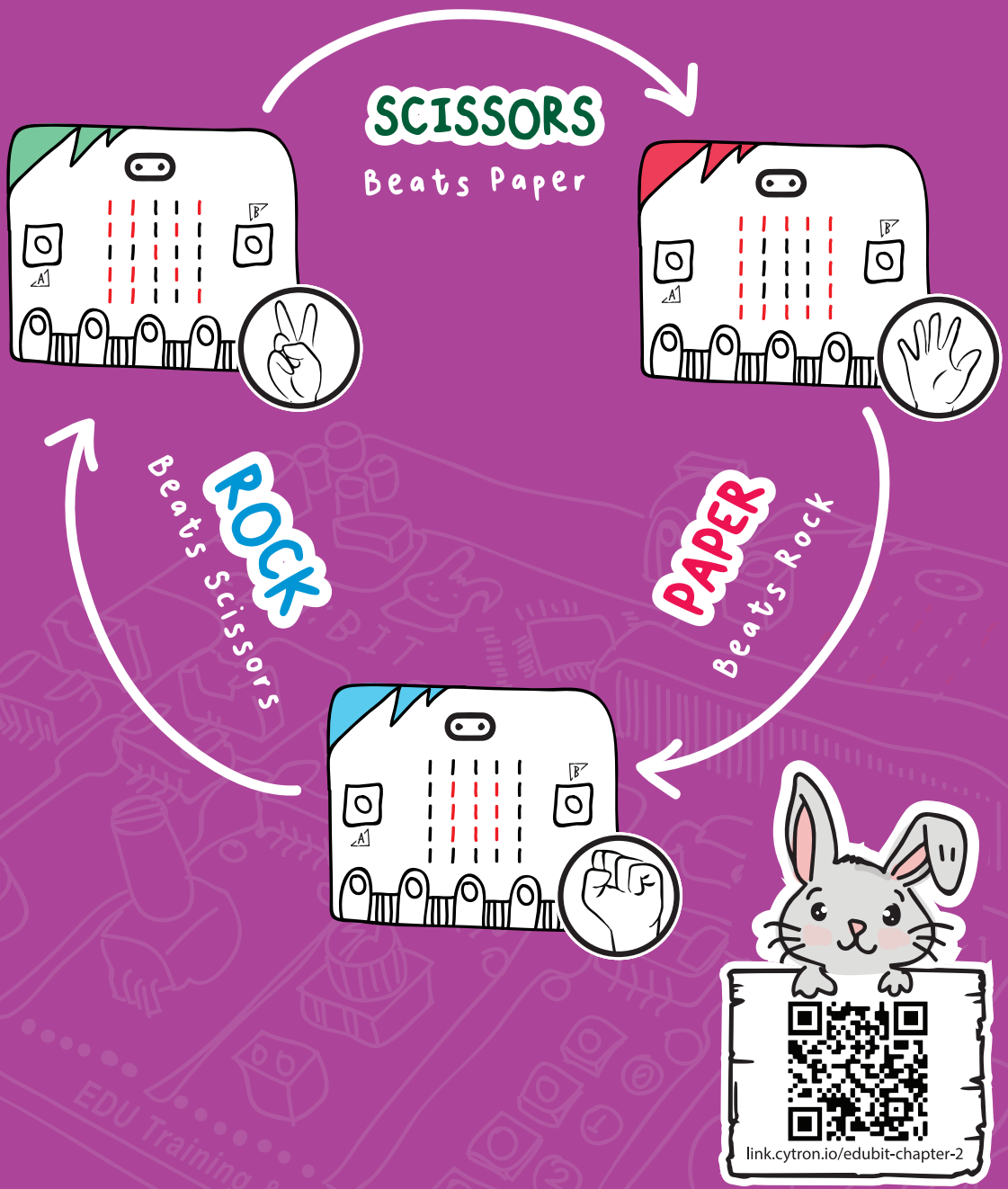
| EDU:BIT가 디지털 알림판으로 쓰이도록 프로그래밍 해봅시다. | |
|---|---|
| 시작하면 실행 | 간단한 애니메이션을 보여주고 학급 이름을 스크롤 해봅시다. |
| 무한반복 실행 | 오늘 날짜와 공지사항을 스크롤 해봅시다. |



# CHAPTER 2

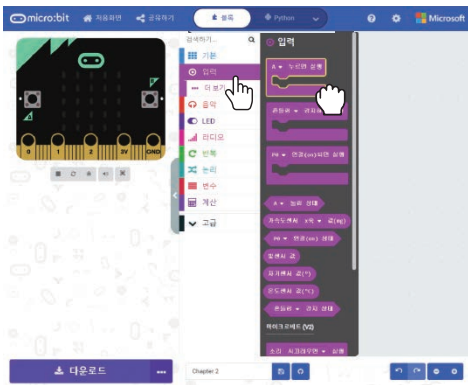
# 가위, 바위, 보!

micro:bit의 푸쉬 버튼과 Button Bit

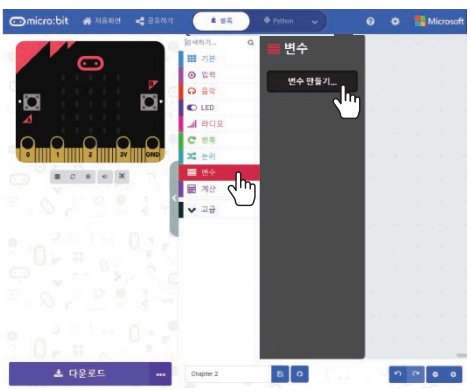


## 코딩을 시작해봅시다!

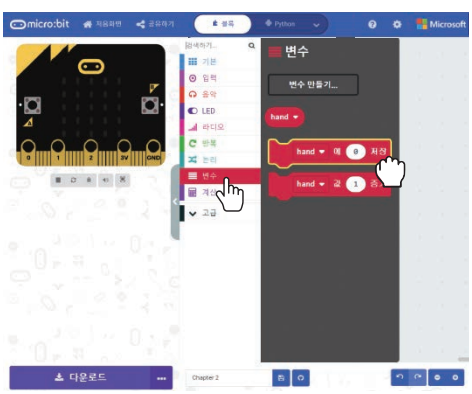
Step1 https://makecode.microbit.org/에 접속합니다. 이미 MakeCode Editor라면 처음화면 아이콘을 클릭하고 새 프로젝트를 생성합니다. [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다.

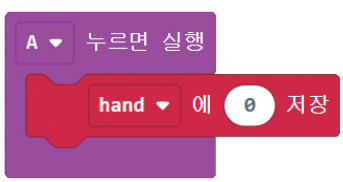



Step 2 [ 변수 ] 카테고리를 클릭하고 [ **변수 만들기** ]를 선택합니다.
팝업창에 'hand'를 입력하고 확인을 누릅니다.

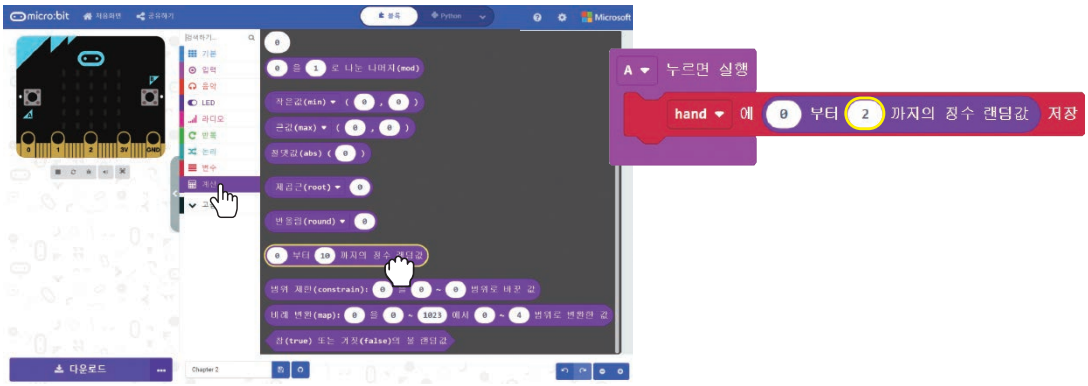



Step 3 [ 변수 ] 카테고리를 클릭하고 [ _ 에 _ 저장] 블록을 선택합니다.
블록을 [ A 누르면 실행 ] 블록에 맞추어 넣습니다.

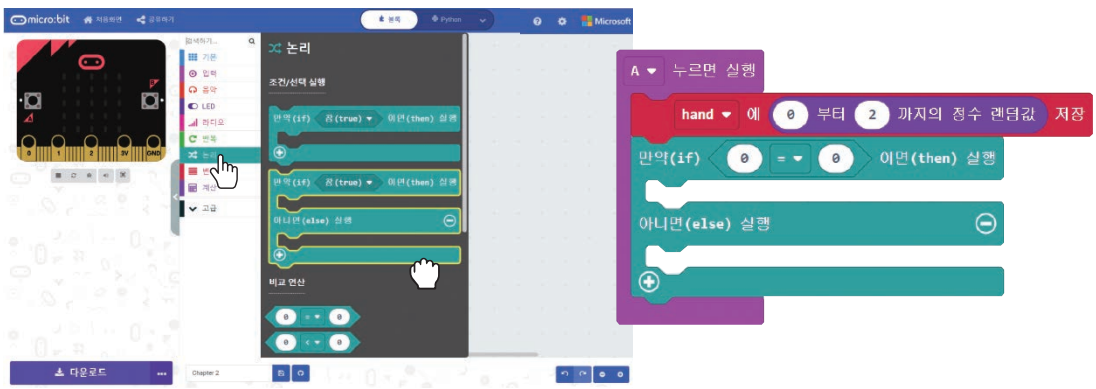




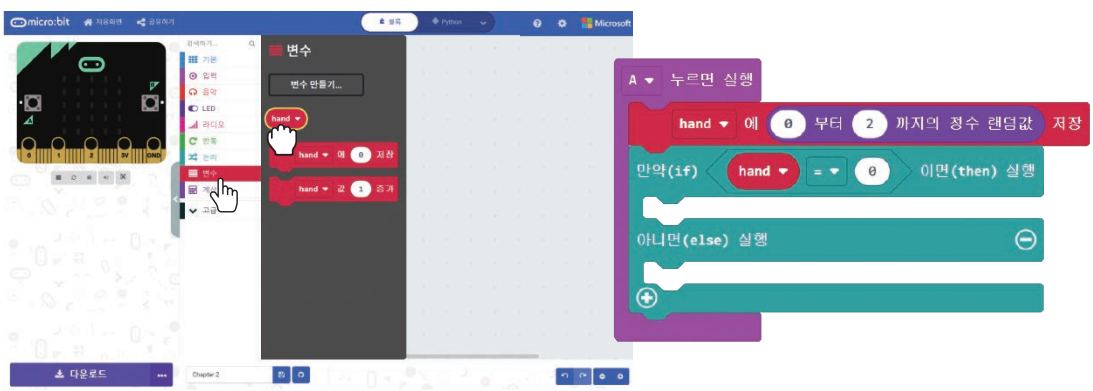
Step 4 [ 계산 ] 카테고리를 클릭하고 [ _ 부터 _ 까지의 정수 랜덤값 ] 블록을 선택합니다.
그리고 10을 2로 변경합니다.




Step 5 [ 논리 ] 카테고리를 클릭하고 조건/선택 실행의 [ 만약(if) _ 이면(then) 실행 아니면(else) 실행 ] 블록과 비교 연산의 [ _ = _ ] 블록을 선택합니다. 비교 연산 블록을 '만약(if)' 슬롯에 넣습니다.



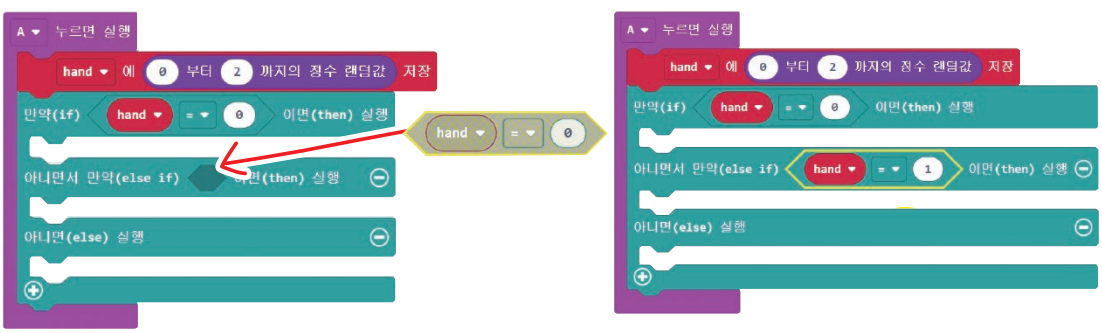

Step 6 [ 변수 ] 카테고리를 클릭하고 [ hand ] 블록을 선택합니다. 그리고 비교 연산 블록에 맞추어 넣습니다.



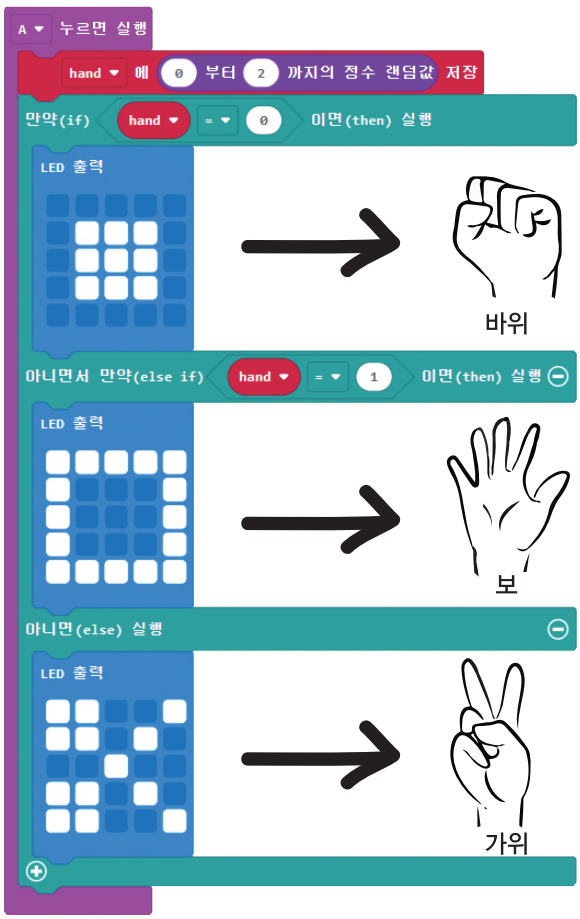
Step 7 [ 만약(if) _ 이면(then) 실행 아니면(else) 실행 ] 블록에서 (+) 아이콘을 클릭하면 [ 아니면서 만약(else if) _ 이면(then) 실행 ] 조건이 추가됩니다.


Step 8 비교 연산 블록에서 마우스 오른쪽 버튼으로 클릭하고 '**복사**'를 선택합니다.


Step 9 복사한 블록을 '아니면서 만약(else if)' 슬롯에 맞추어 넣고 값을 0에서 1로 변경합니다.



Step 10 [ 기본 ] 카테고리를 클릭하고 [ LED 출력 ] 블록을 '만약(if)', '아니면서 만약(else if)', '아니면(else)' 슬롯에 각각 추가합니다. 아래에 보이는 것처럼 [ LED 출력 ] 블록의 상자들을 클릭해서 이미지를 만듭니다.

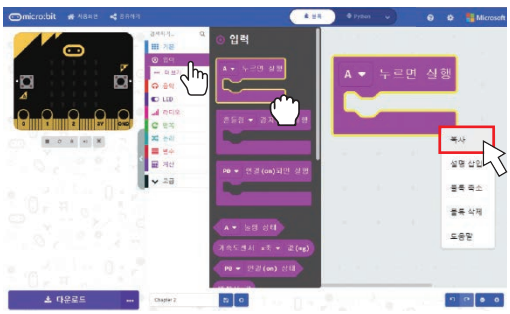




코드를 EDU:BIT에 플래시하면 친구들과 함께 가위바위보 게임을 할 수 있습니다.
micro:bit에서 A 버튼이나 노란색 버튼을 누를때마다
LED 매트릭스는 랜덤으로 '가위', '바위', '보'를 보여줍니다.

**만약 프로젝트를 보관하고 싶다면,**
**'파일 저장' 버튼을 눌러서 컴퓨터 폴더에 저장할 수 있음을 기억해둡니다.**

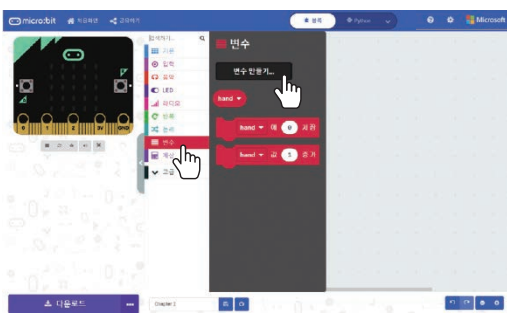
게임을 좀 더 재미있게 만들어볼까요?
'lives'라는 새로운 변수를 만들고 블록을 추가해서 한 사람당 목숨을 세 개 갖도록 코드를 수정해봅시다.

Step 11 [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다. 블록을 복사하고 각각 'B'와 'A+B'로 설정합니다.



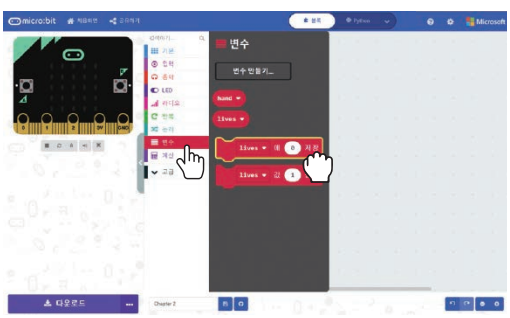


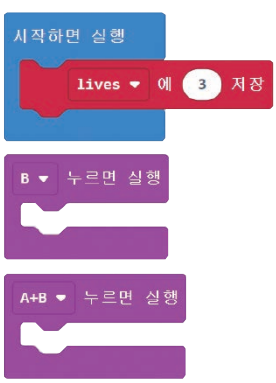
Step 12 [ 변수 ] 카테고리를 클릭하고 [ **변수 만들기** ]를 선택합니다. 팝업창에 'lives'를 입력하고 확인을 누릅니다.



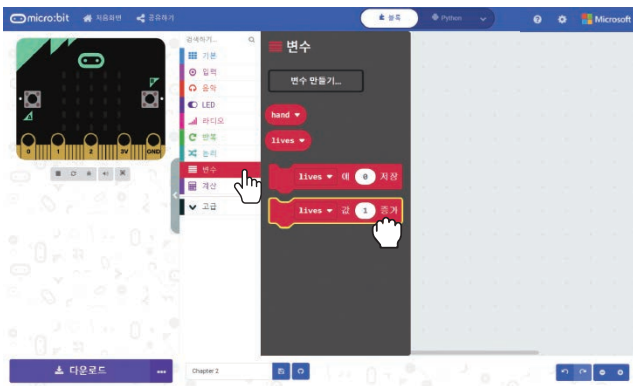


Step 13 [ 변수 ] 카테고리를 클릭하고 [ _ 에 _ 저장 ] 블록을 선택합니다.
[ 기본 ] 카테고리의 [ 시작하면 실행 ] 블록에 맞추어 넣습니다. 변수는 'lives'로 설정하고 값은 3으로 변경합니다.



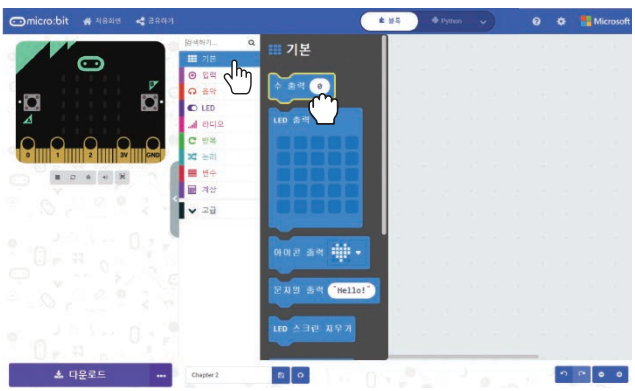




Step 14 다시 [ 변수 ] 카테고리를 클릭하고 [ _ 값 _ 증가 ] 블록을 선택합니다.
[ B 누르면 실행 ] 블록에 맞추어 넣습니다. 변수는 'lives'로 설정하고 값은 –1로 변경합니다.



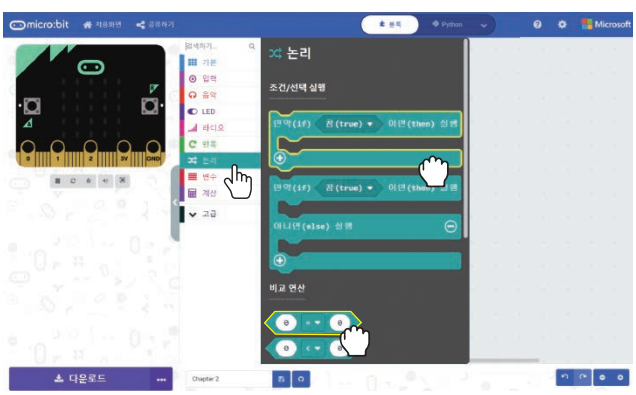


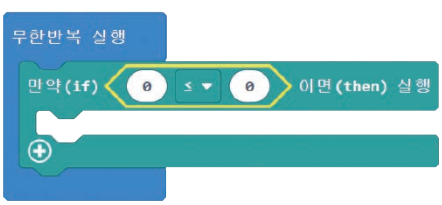
Step 15 [ 기본 ] 카테고리를 클릭하고 [ 수 출력 ] 블록을 선택합니다.
블록을 복사하고 [ B 누르면 실행 ] 블록과 [ A+B 누르면 실행 ] 블록에 각각 맞추어 넣습니다.



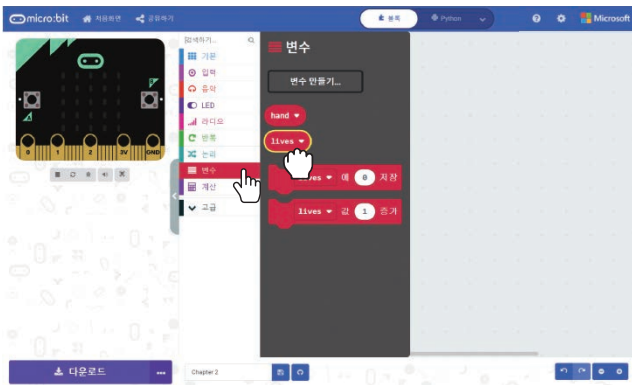


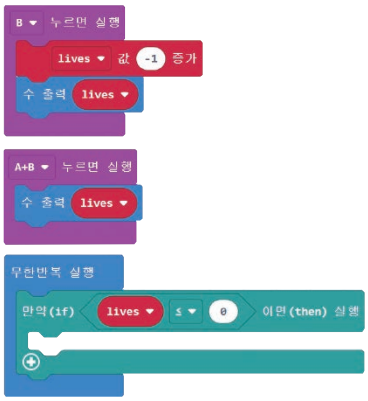
Step 16 [ 논리 ] 카테고리를 클릭하고 [ 만약(if) _ 이면(then) 실행 ] 블록과 [ _ = _ ] 비교 연산 블록을 추가합니다. [ 기본 ] 카테고리의 [ 무한반복 실행 ] 블록에 맞추어 넣고 ' ≤ ' 표시로 변경합니다.



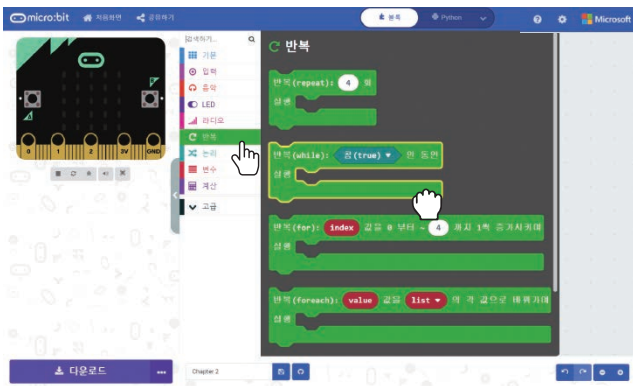



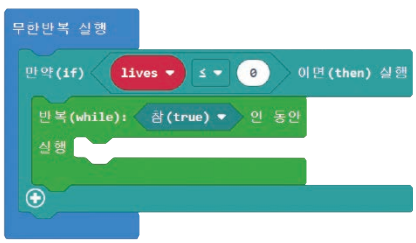
Step 17 [ 변수 ] 카테고리를 클릭하고 [ lives ] 블록을 선택합니다.
복사한 후 [ 수 출력 ] 블록들과 [ _ = _ ] 비교 연산 블록의 왼쪽 슬롯에 맞추어 넣습니다.



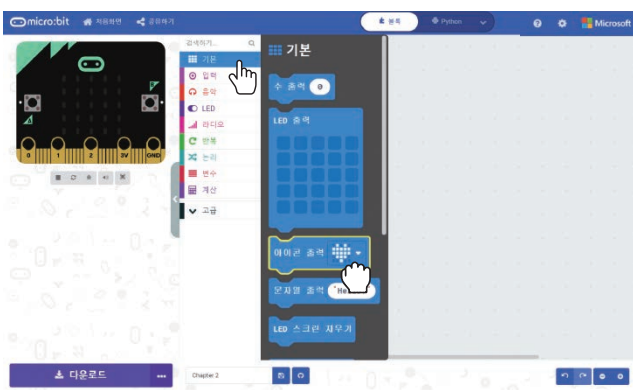



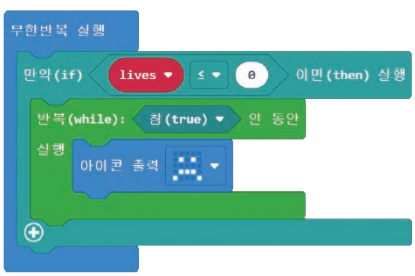

Step 18 [ 반복 ] 카테고리를 클릭하고 [ 반복(while): _ 인 동안 실행 ] 블록을 선택합니다.
그리고 [ 만약(if) _ 이면(then) 실행 ] 블록에 맞추어 넣습니다.



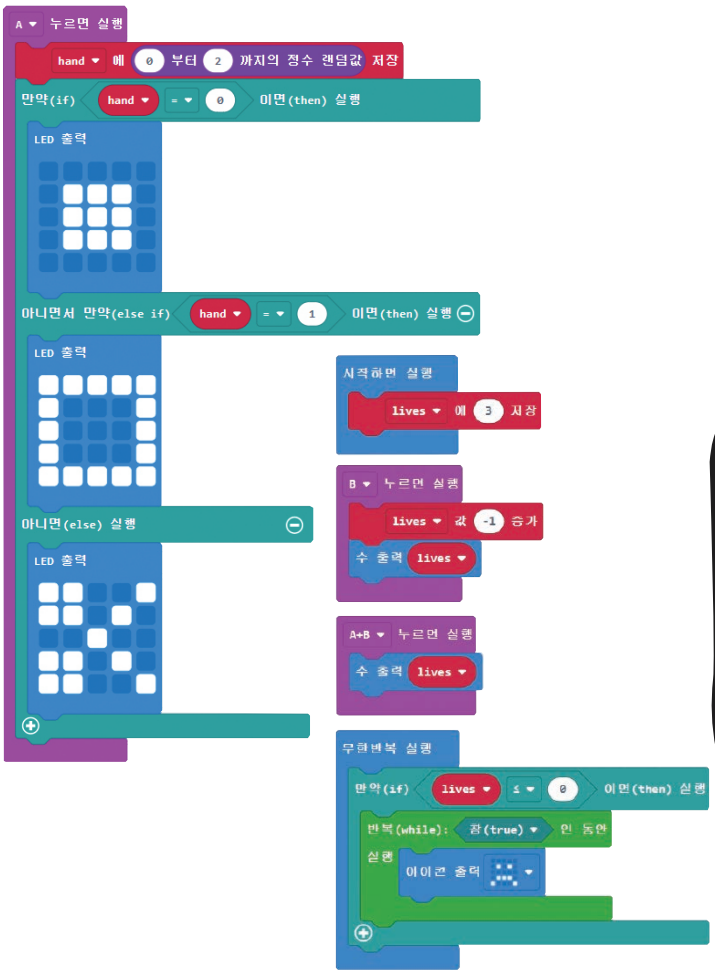



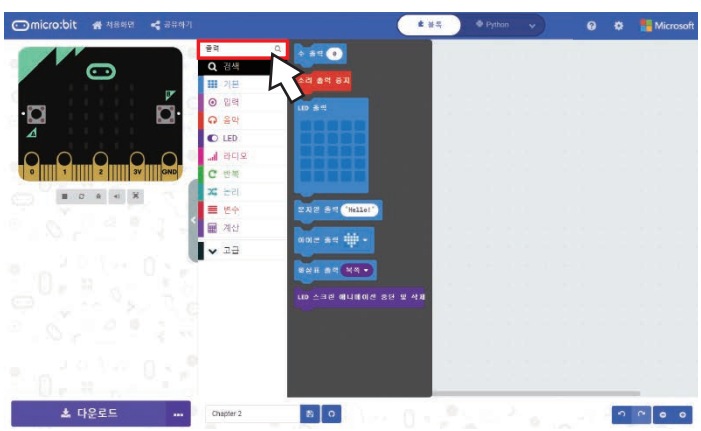

Step 19 [ 기본 ] 카테고리를 클릭하고 [ 아이콘 출력 ] 블록을 선택합니다.
[ 반복(while): _ 인 동안 실행 ] 블록의 슬롯에 맞추어 넣고 아이콘을 '**슬픔**'으로 변경합니다.



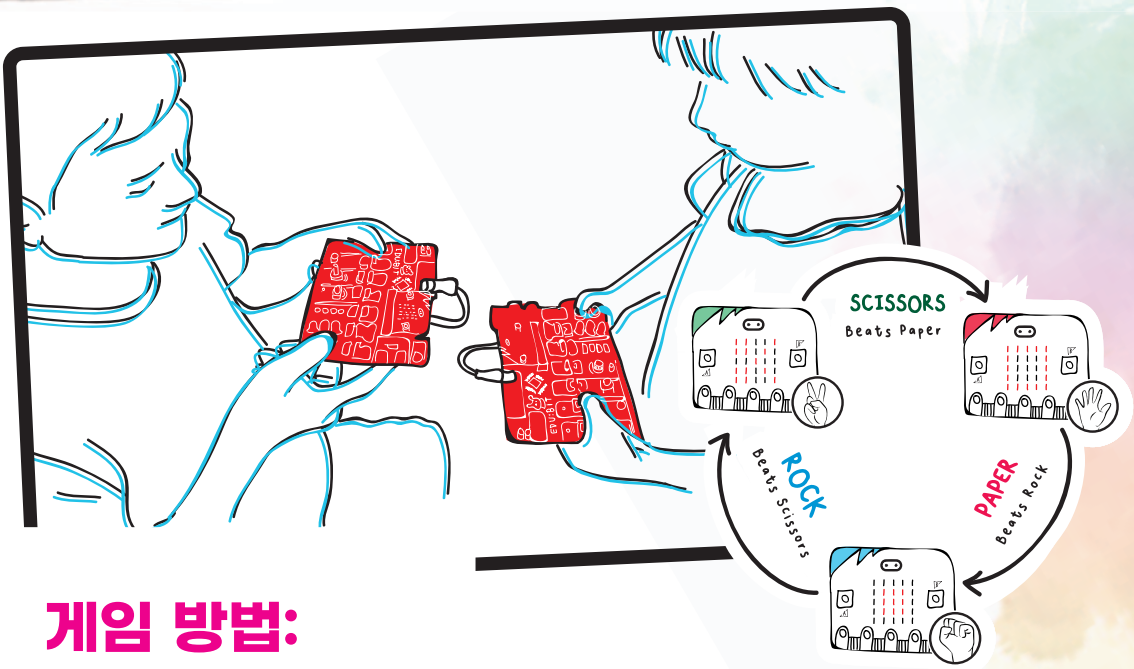
Step 20 코드가 완성되었습니다. EDU:BIT에 플래시하고 친구들과 가위바위보 게임의 승자를 결정해봅시다.




# 게임하기

## 가위바위보 심화 버전




## 게임 방법:

- 상대방과 마주 보고 섭니다. 두 사람 모두 준비가 되면 노란색 버튼 (또는 A 버튼)을 누릅니다. 랜덤으로 가위, 바위, 보가 나타납니다.
- 비교해보고 누가 이겼는지 판단합니다.
- 진 사람은 EDU:BIT의 파란색 버튼 (또는 B 버튼)을 누릅니다. 한 번 누를 때마다 목숨이 한 개 감소합니다.
- 노란색 버튼과 파란색 버튼을 동시에 누르면 남은 목숨이 몇 개인지 확인할 수 있습니다.
- 세 번 지면 게임이 끝나고, EDU:BIT에 '슬픔' 아이콘이 표시됩니다.

**주 목 !**

게임을 새로 시작하려면 보드를 리셋해야 합니다.
만약 함께 게임할 친구가 없다면, MakeCode Editor에서 시뮬레이터와 언제든지 겨루어 볼 수 있습니다.

# 코딩 정복하기

컴퓨터 프로그래밍에서, 런타임(프로그램 실행 시간) 동안 변경될 수 있는 정보나 값을 보관하기 위해 변수를 사용합니다.
변수는 정보가 적힌 종이가 들어있는 이름 적힌 봉투로 생각할 수 있습니다. 종이는 제거될 수 있고 새로운 정보가 적힌 다른 종이로 교체될 수 있습니다.

앞의 코드에서 'lives'라고 불리는 변수를 만들었고 초깃값을 3으로 할당했습니다.


'3'이라고 적혀있는 종이는 정보입니다.

'lives'라고 적혀있는 봉투는 변수입니다.

B 버튼이 눌릴 때마다, 값은 –1만큼 변경됩니다.

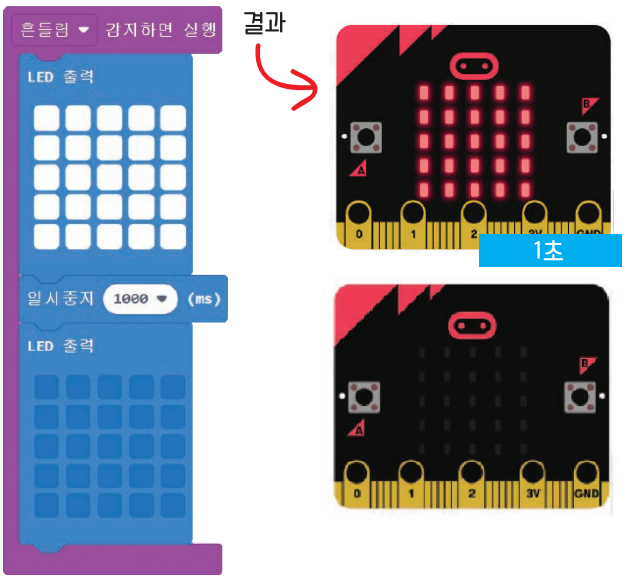

B 버튼이 눌렸을 때, '3'이라고 적힌 종이는 봉투에서 제거되고, '2' ( i.e. 3 - 1 = 2 )라고 적힌 다른 종이로 교체됩니다.

A 버튼과 B 버튼이 동시에 눌리면, 'lives' 변수의 현재 값을 LED 매트릭스에 보여줍니다.

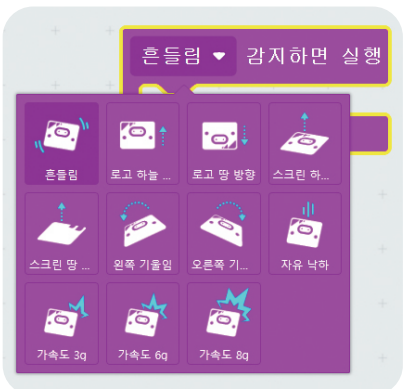

A 버튼과 B 버튼이 동시에 눌렸을 때, 봉투에 들어있던 종이를 꺼내고 적혀있는 정보를 읽습니다.

# 더 많은 블록 탐구하기

[ _ 누르면 실행 ] 블록뿐만 아니라, [ 입력 ] 카테고리 서랍으로부터 이벤트 기반 프로그래밍을 위한 다른 블록들을 사용할 수 있습니다.
사용자가 버튼을 누르거나 보드를 흔드는 것과 같은 이벤트에 의해서 동작이 실행됩니다.
예를 들어 다음 코드는 보드가 흔들릴 때마다 1초 동안 LED 매트릭스에 불이 들어옵니다.
한번 해볼까요 ~

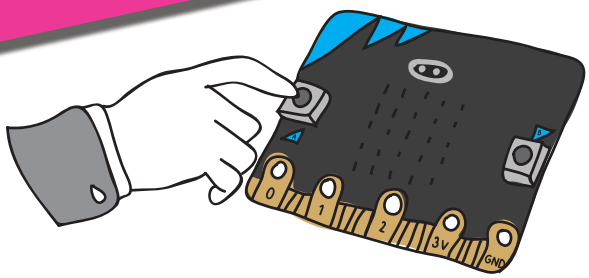




블록에서 [ 흔들림 ] 버튼을 클릭하면, 다른 트리거를 선택할 수 있는 팝업메뉴가 나타납니다.
EDU:BIT가 각 조건에 대하여 다른 아이콘을 보여주도록 프로그래밍 해봅시다.

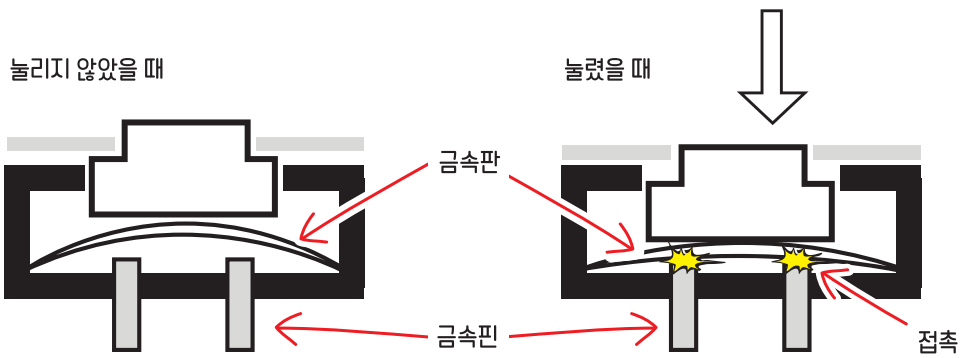


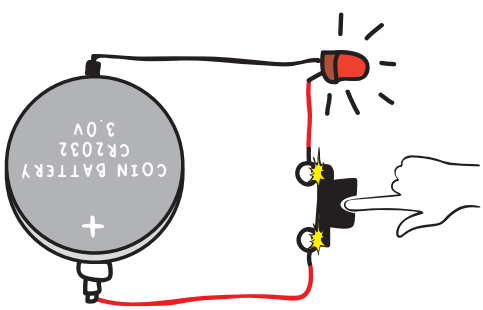

EDU:BIT는 micro:bit에 움직임 센서가 내장되어 있기 때문에 흔들림을 감지하고 움직인 방향을 알 수 있습니다.

# 재미있는 사실!




푸쉬 버튼(push button)은
눌리지 않았을 때와 눌렸을 때 두 가지 상태만
가능한 입력 장치입니다.




푸쉬 버튼이 눌렸을 때,
전기회로가 완성되어 LED에 불이 들어옵니다!
푸쉬 버튼이 눌리지 않았을 때는
어떤 일이 발생할까요?

**도와줘요!**

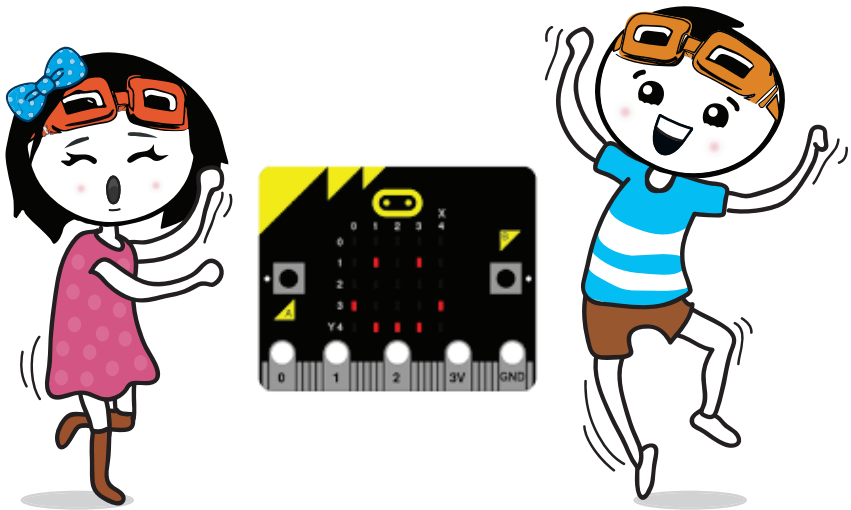
비상버튼

위급한 상황일 때
버튼을 누르세요

일반적으로 검은색, 회색, 초록색, 흰색 버튼들은 ON/OFF 기능을 위해 사용되고 빨간색은 비상버튼이나 기계류의 긴급 정지를 위해 사용됩니다.


youtu.be/t_Qujjd_38o

# 응용 과제

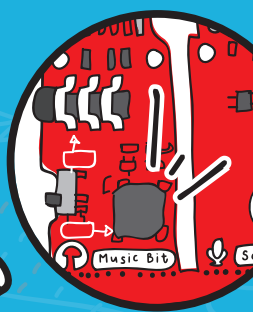



| EDU:BIT로 출석한 학생 수를 셀 수 있도록 프로그래밍 해봅시다.<br>학생들이 교실로 들어올 때, 여자는 A 버튼을 누르고 남자는 B 버튼을 누릅니다. | |
|---|---|
| 시작하면 실행 | 웃는 표정을 출력합니다.<br>변수를 Girl = 0, Boy = 0으로 설정합니다. |
| A 버튼 누르면 실행<br>(노란색 버튼) | 변수 Girl을 1만큼 변경합니다. |
| B 버튼 누르면 실행<br>(파란색 버튼) | 변수 Boy를 1만큼 변경합니다. |
| A+B 버튼 누르면 실행 | 다음 정보를 LED 디스플레이에 스크롤 합니다.<br>Total = (Girl+Boy) ; Girl = (Girl) ; Boy = (Boy) |

CHAPTER 3

# 음악을 들어보자~

Music Bit (피에조 부저 + 오디오 잭)



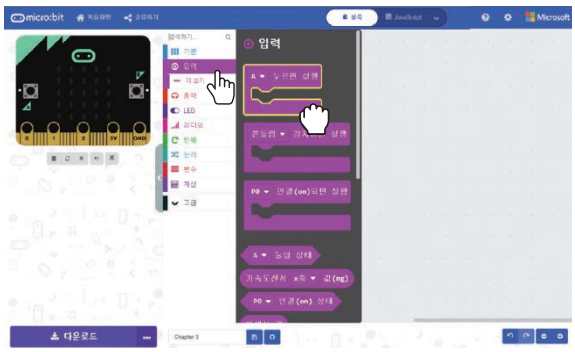

스캔하세요!

link.cytron.io/edubit-chapter-3

마이크로비트 V2와 호환 가능합니다. 단, 내장 스피커를 OFF로 설정해주세요. (상품페이지 참조)

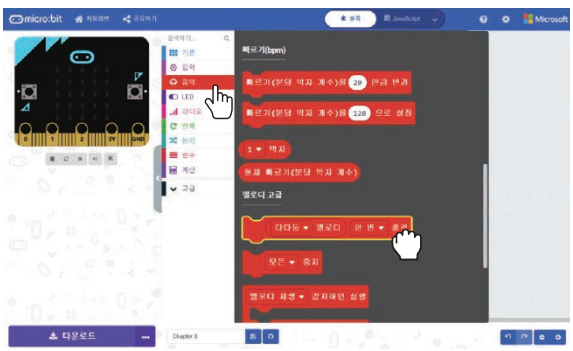
## 코딩을 시작해봅시다!

Step 1 MakeCode Editor에서 새 프로젝트를 생성합니다. [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다.





Step 2 [ 음악 ] 카테고리를 클릭하고 [ _ 멜로디 _ 출력 ] 블록을 선택합니다.





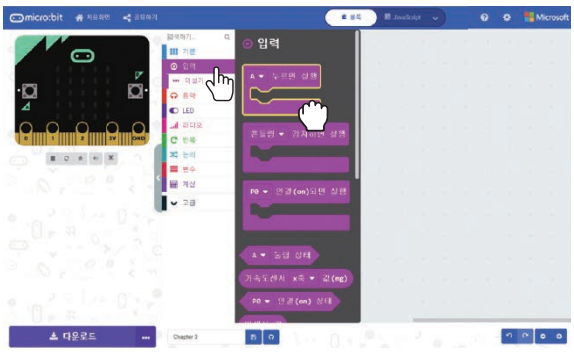
Step 3 [ 다다둠 ]을 클릭하고 목록에서 '생일' 멜로디를 선택합니다.



화면에 보이는 시뮬레이터의 A 버튼을 클릭합니다.
익숙한 곡이 들리나요? 다른 멜로디들도 확인해봅시다~
* 컴퓨터 스피커가 켜져있는지 확인해주세요.

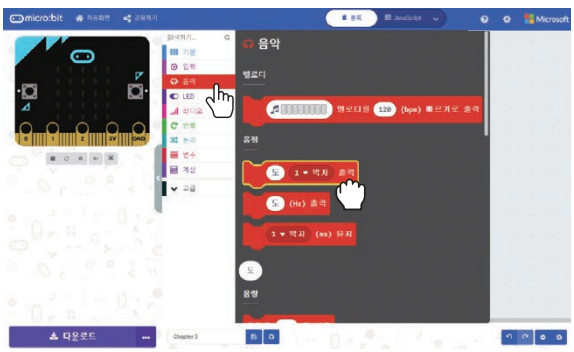
목록에 있는 멜로디 외에도, EDU:BIT로 좋아하는 노래를 연주하도록 프로그래밍 할 수 있습니다. 귀에 쏙 들어오는 곡을 연주해봅시다~

Step 4 [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다. B로 변경합니다.



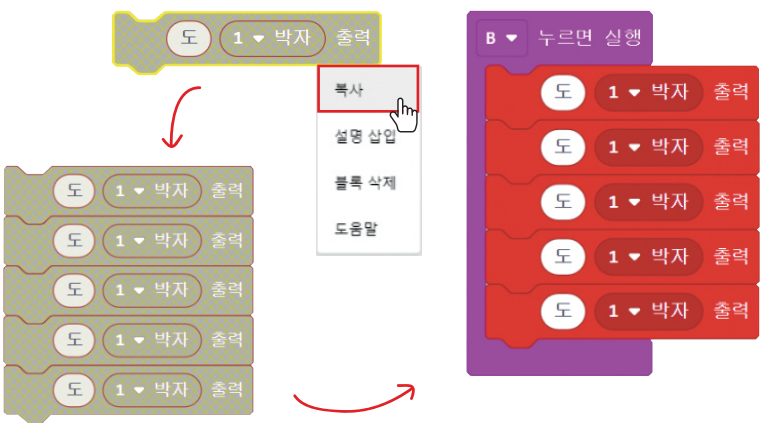


Step 5 [ 음악 ] 카테고리를 클릭하고 [ _ _ 박자 출력 ] 블록을 선택합니다.

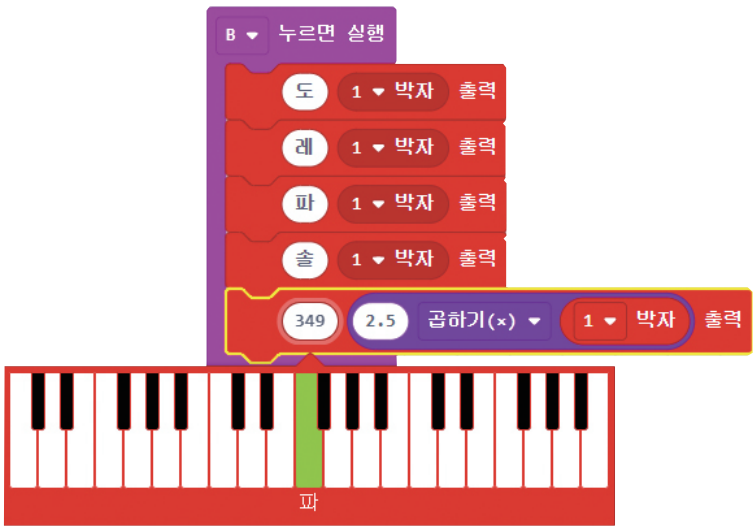




Step 6 작업공간에서 [ _ _ 박자 출력 ] 블록을 마우스 오른쪽 버튼으로 클릭하고 '복사'를 누릅니다. [ _ _ 박자 출력 ] 블록이 다섯 개가 될 때까지 반복합니다. 그리고 블록들을 [ B 누르면 실행 ] 블록의 슬롯에 맞추어 넣습니다.

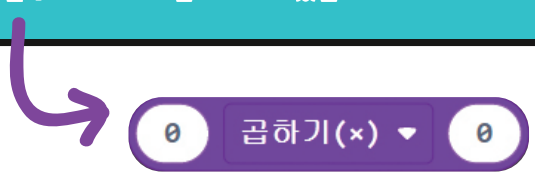


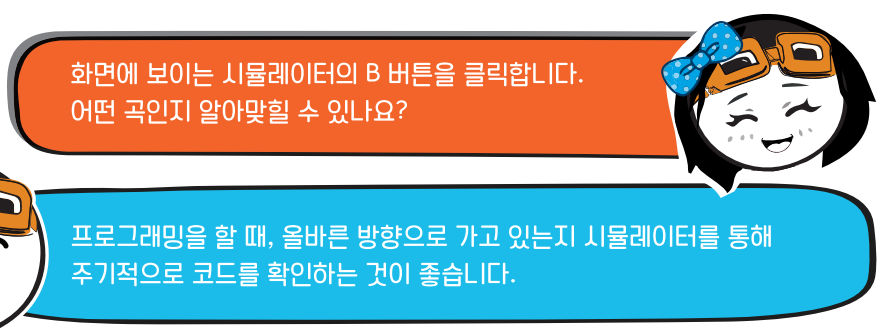
Step 7 아래 예시처럼 [ _ _ 박자 출력 ] 블록들의 '음'과 '박자'를 선택합니다.




**주목!**

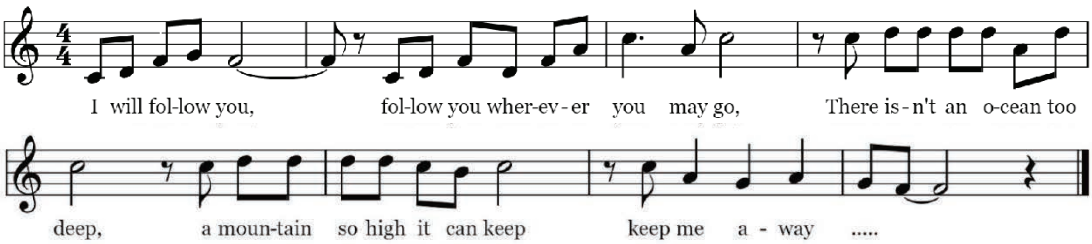
보라색 블록은 [ 계산 ] 카테고리 서랍에서 가져왔습니다.



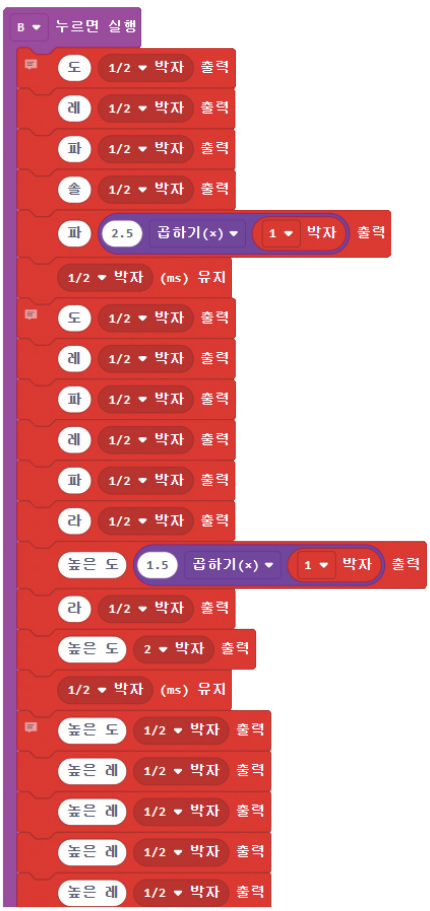




Step 8 [ _ _ 박자 출력 ] 블록을 더 추가하고 '음'과 '박자'를 변경함으로써 곡의 나머지 부분들을 코딩해봅시다. 다음 페이지에서 음과 박자를 참고해봅시다.


# I Will Follow You



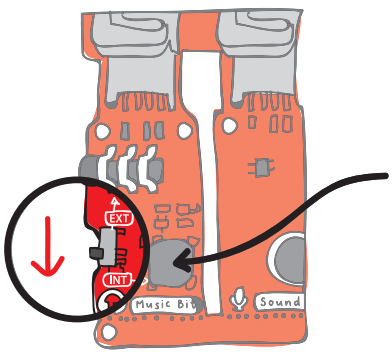

I will fol- low you, Fol- low you wher- ev- er you may go, There is n't an o-



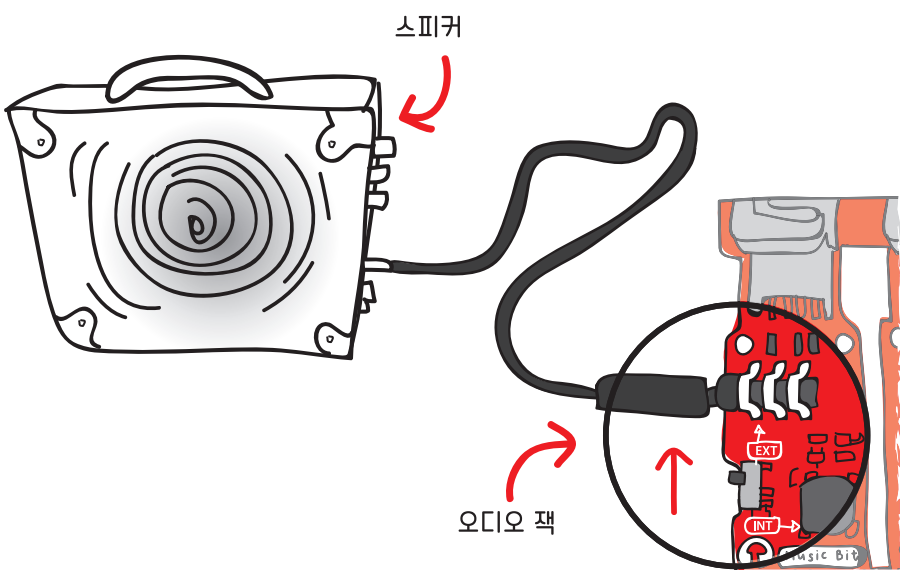

cean too deep A moun- tain so high it can keep, Keep me a- way ... ...


Step 9 완성된 코드를 EDU:BIT에 플래시합니다.

EDU:BIT에서 파란색 버튼(B 버튼)을 누를 때마다 'I Will Follow You' 노래가 나올 것입니다. EDU:BIT의 전원을 켜고 스위치를 INT(internal)로 옮겨서 피에조 부저를 켜야 함을 기억하세요.

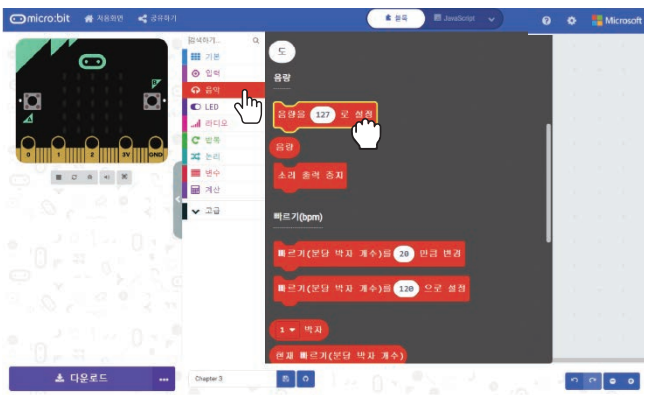



다른 방법으로는 오디오 잭으로 EDU:BIT에 외부 스피커나 헤드셋을 연결할 수 있습니다.
이때 스위치는 EXT(external)로 옮겨야 합니다.



음악이 너무 작은가요? 또는 너무 큰가요?
음량을 변경하기 위해서 [ 음량을 _ 로 설정 ] 블록을 추가하고 소리 크기는 0부터 최대 255사이의 값으로 설정합니다.

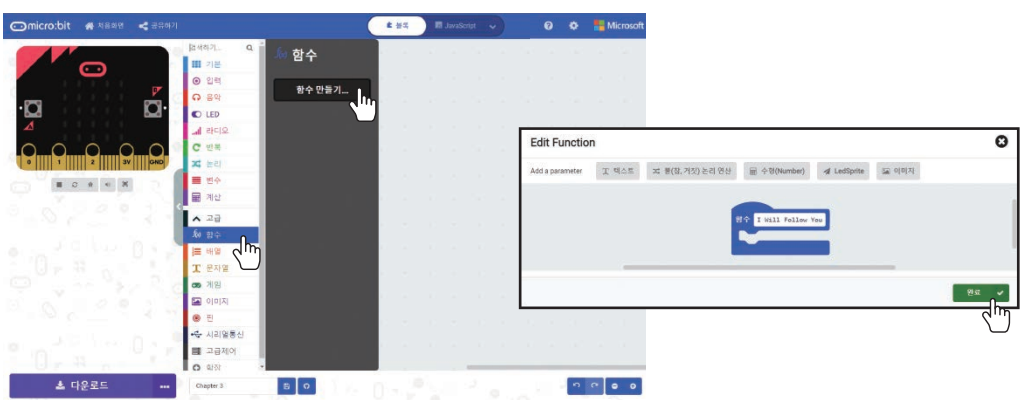
Step 10 [ 음악 ] 카테고리를 클릭하고 [ 음량을 _ 로 설정 ] 블록을 선택합니다.
[ 시작하면 실행 ]에 맞추어 넣고 음량을 200으로 변경합니다.



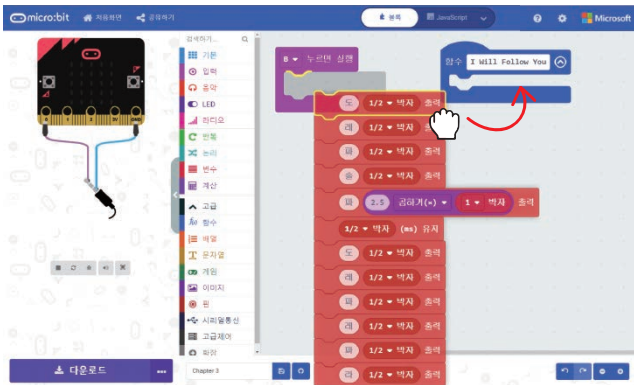

## 주 목 !

I Will Follow You 노래를 실행하는 코드처럼 특정한 기능을 수행하는 코드 블록들을 함수로 만들 수 있습니다. 프로그래밍에서, 함수는 루틴이나 일련의 절차를 나타냅니다. 함수가 한 번 정의되면, 반복해서 동일한 코드 블록들을 다시 쓰지 않고도 프로그램의 여러 장소에서 사용될 수 있습니다.

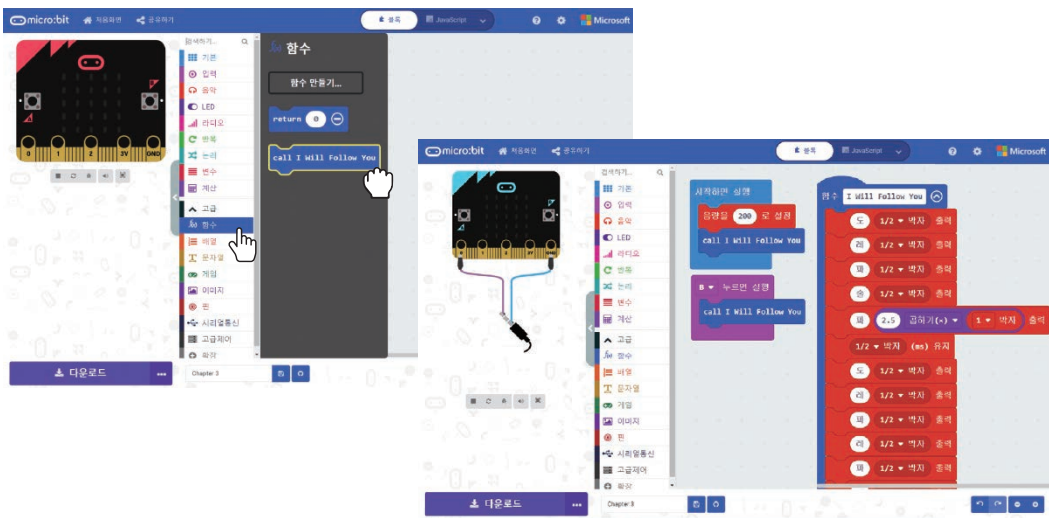
Step 11 [ 고급 ] 카테고리를 클릭하고 [ 함수 ] 카테고리를 선택합니다. [ 함수 만들기 ]를 클릭하고, 팝업창에서 doSomething을 'I Will Follow You'로 변경합니다. 그다음 '**완료**'를 클릭합니다.



Step 12 [ 함수 I Will Follow You ] 블록이 작업 공간에 보일 것입니다. [ B 누르면 실행 ] 블록에서 가장 위에 있는 블록을 클릭한 채로 [ 함수 I Will Follow You ] 슬롯으로 드래그합니다.



Step 13 [ 함수 ] 카테고리를 클릭하고 [ call I Will Follow You ] 블록을 선택하고 복사합니다.
[ call I Will Follow You ] 블록들을 각각 [ 시작하면 실행 ] 블록과 [ B 누르면 실행 ] 블록에 맞추어 넣습니다.
예시 코드입니다.

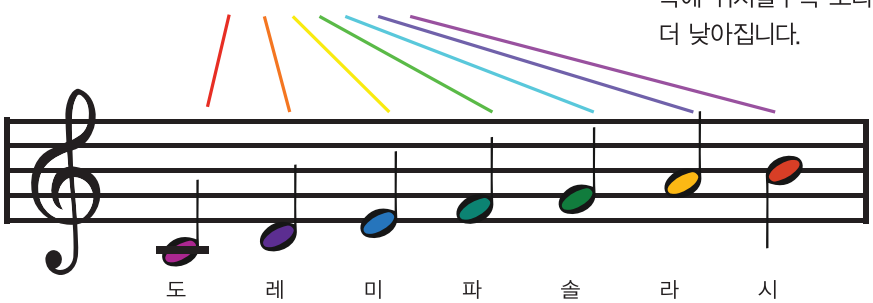


Step 14 완성된 코드를 EDU:BIT에 플래시합니다. 음악을 감상해봅시다~

# 코딩 정복하기

악보를 볼 줄 안다면, EDU:BIT로 다른 노래들을 연주하도록 프로그래밍 할 수 있습니다. 여기에 악보를 '디코드' 하는 데 도움이 되는 간단한 안내가 나와있습니다.


오선지에서 음표의 위치는 연주할 음을 말해줍니다. 음표가 오선지에서 위쪽에 위치할수록 소리의 음높이와 주파수가 더 높아집니다. 마찬가지로 음표가 오선지에서 아래쪽에 위치할수록 소리의 음높이와 주파수가 더 낮아집니다.



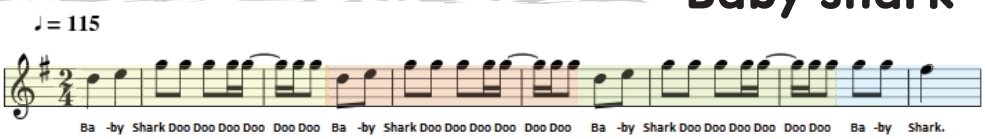

| 음표 | 쉼표 | 상대 길이 | 지속시간 |
|---|---|---|---|
| 𝅝 | 𝄻 | 온음표 | 4박자 |
| 𝅗𝅥 | 𝄼 | 2분음표 | 2박자 |
| ♩ | 𝄽 또는 𝄾 | 4분음표 | 1박자 |
| ♪ | 𝄿 | 8분음표 | 1/2박자 |
| 𝅘𝅥𝅯 | 𝅀 | 16분음표 | 1/4박자 |

음악적 표기법들은 음표가 얼마나 길게 연주되어야 하는지 알려주기 위해 사용됩니다.

# 코딩 정복하기

알려준 것들을 사용해서, 아래 음을 '디코드' 할 수 있나요?

## Baby shark



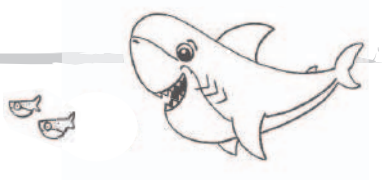

| Line 1 | Ba | –by | Shark | doo | doo, | doo | doo | doo | doo |
|---|---|---|---|---|---|---|---|---|---|
| 음표 | 높은 레 | 높은 미 | 높은 솔 | | 높은 솔 | 높은 솔 | 높은 솔 | | 높은 솔 |
| 박자 | 1 | | 1/2 | 1/2 | 1/2 | | 1/2 | 1/4 | 1/2 |

| Line 2 | Ba | –by | Shark | doo | doo, | doo | doo | doo | doo |
|---|---|---|---|---|---|---|---|---|---|
| 음표 | 높은 레 | | 높은 솔 | 높은 솔 | 높은 솔 | | 높은 솔 | 높은 솔 | 높은 솔 |
| 박자 | 1/2 | 1/2 | 1/2 | | 1/2 | 1/4 | 1/2 | | 1/2 |

Line3은 똑같이 반복합니다.

| Line 4 | Ba | –by | Shark |
|---|---|---|---|
| 음표 | | 높은 솔 | 높은 파# |
| 박자 | 1/2 | 1/2 | |

노란색 버튼(A 버튼)과 파란색 버튼(B 버튼)을 동시에 눌렀을 때 EDU:BIT가 Baby Shark 음을 연주하도록 프로그래밍 해봅시다.



**주 목 !**

소리 크기를 조절하기 위해 [ 음량을 _ 로 설정 ] 블록을 사용하세요.

# 더 많은 블록 탐구하기

#1 [ 빠르기(분당 박자 개수)를 _ 만큼 변경 ] 블록을 사용해서 '템포'(노래의 속도)를 설정할 수 있습니다.
bpm(분당 박자 개수)이 클수록, 음이 더 빠르거나 더 박진감이 있습니다.
템포를 변경하기 위해 [ 빠르기(분당 박자 개수)를 _ 만큼 변경 ] 블록을 사용합니다.

#2 현재 재생되고 있는 멜로디를 중지시키기 위해 [ _ 중지 ] 블록을 사용합니다.

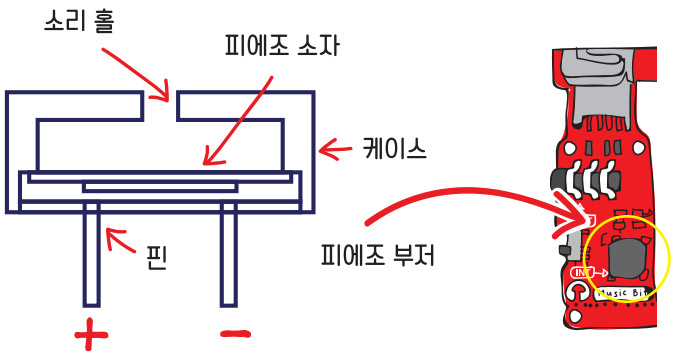
#3 코드에서 멜로디 시작과 멜로디 종료와 같은 이벤트가 발생했을 때 실행되도록 [ _ 감지하면 실행 ] 블록에서 조건을 선택할 수 있습니다.

## 예시 코드


1. 이 프로그램에서, 초기 템포는 120bpm으로 설정되어 있습니다.
2. 멜로디가 시작되면, LED 매트릭스에 음표 아이콘이 나타납니다.
3. 멜로디가 종료되면, 매트릭스의 모든 LED가 꺼집니다.
4. A 버튼을 누를 때마다 템포가 50bpm씩 증가됩니다.
5. '엔터테이너' 멜로디는 B 버튼이 눌릴 때마다 재생됩니다.
6. 멜로디는 A와 B 버튼이 동시에 눌리면 중지됩니다.

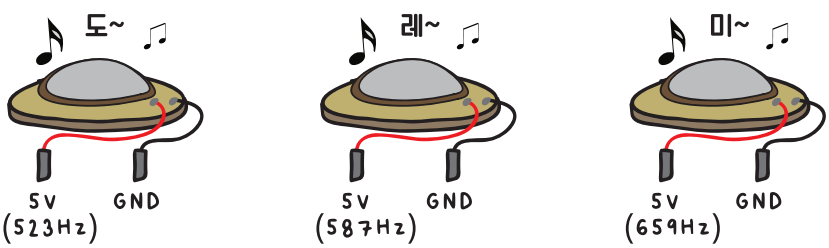
# 재미있는 사실!

피에조 부저는 일반적으로 전기신호가 통과할 때 피에조 소자의 일부를 진동시켜 소리를 만듭니다.



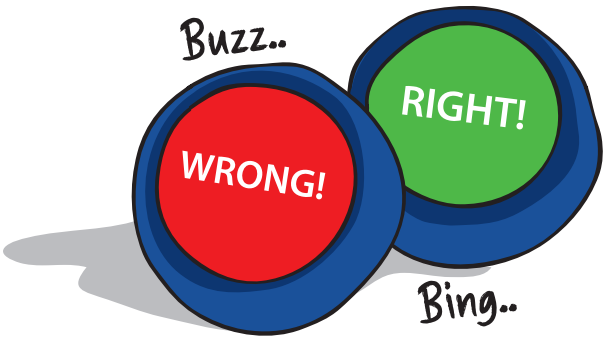

전기신호의 주파수가 변경되면, 진동 속도도 변경됩니다.
이러한 이유로 피에조 부저는 다른 음의 소리를 낼 수 있습니다.

**피에조 소자**




사람의 귀는 20Hz에서 20000Hz까지 들을 수 있습니다.
20Hz 아래의 소리는 초저주파라고 불리며, 20000Hz 위의 소리는 초음파라고 불립니다.

# 응용 과제




| EDU:BIT가 정답/오답을 알려주는 Game Show 부저 기능을 하도록 프로그래밍 해봅시다. | |
|---|---|
| 시작하면 실행 | 웃는 얼굴이 나타납니다. |
| A 누르면 실행<br>(노란색 버튼) | ✔ 아이콘이 나타나고 '전원 켜는' 멜로디가 한 번 출력됩니다. |
| B 누르면 실행<br>(파란색 버튼) | ✘ 아이콘이 나타나고 '와와와와아' 멜로디가 한 번 출력됩니다. |
| A+B 누르면 실행 | 화면이 지워집니다. |

# CHAPTER 4

# 그림 보고 맞추기 게임!

Traffic Light Bit

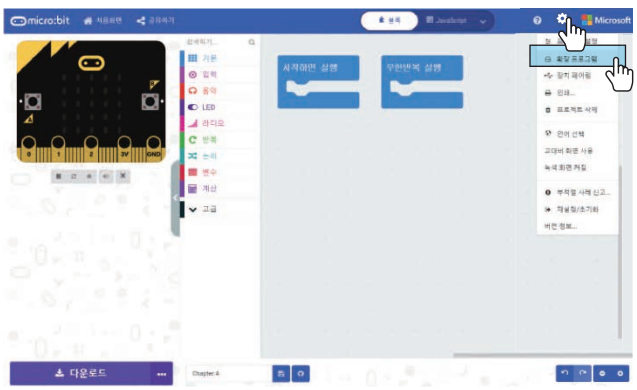


스캔하세요!

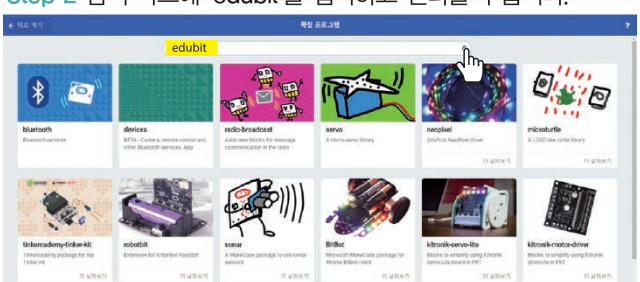
EDU:BIT에 빨간색, 노란색, 초록색 LED가 있다는 걸 알고 있나요?
그것을 Traffic Light Bit 라고 합니다. 프로그래밍 하려면 MakeCode Editor에서 EDU:BIT 확장 프로그램을 추가해야 합니다. 확장 프로그램은 EDU:BIT 보드처럼 micro:bit 부속품들을 쉽게 프로그래밍 할 수 있도록 편집기에 추가하는 커스텀 블록 세트입니다.

## 코딩을 시작해봅시다!

Step 1 MakeCode Editor에서 새 프로젝트를 생성합니다. 톱니바퀴 아이콘 ⚙을 클릭하고 '확장 프로그램'을 선택합니다. * 확장 프로그램을 추가하기 위해서 인터넷에 연결해야 합니다.

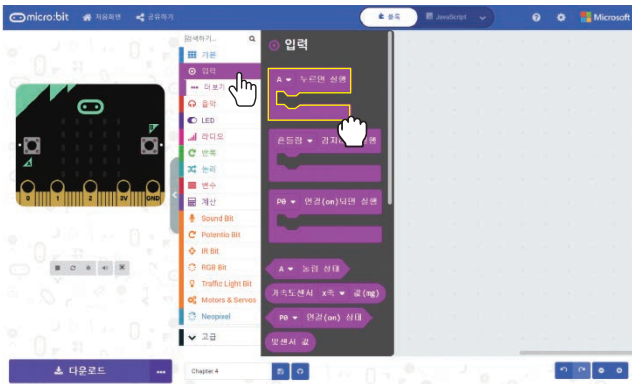



Step 2 검색 박스에 'edubit'를 입력하고 엔터를 누릅니다.

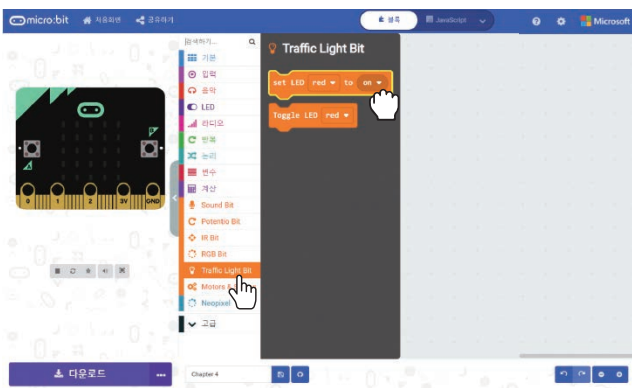


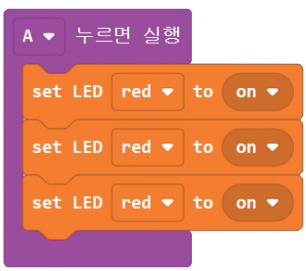

Step 3 'edubit' 확장 프로그램을 클릭합니다. 로드 될 때까지 기다리면 다음과 같이 MakeCode Editor에 새 카테고리 서랍들이 생긴 것을 볼 수 있습니다.

Step 4 [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다.

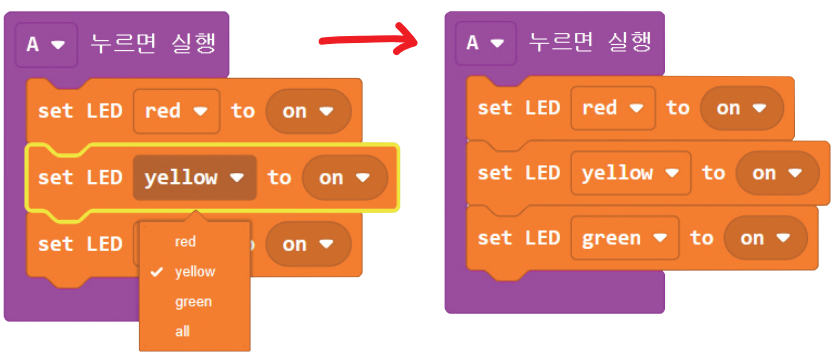



Step 5 [ Traffic Light Bit ] 카테고리를 클릭하고 [ set LED _ to _ ] 블록을 선택합니다.
작업공간에서, [ set LED _ to _ ] 블록을 마우스 오른쪽 클릭하여 '복사'를 선택합니다.
[ set LED _ to _ ] 블록이 세 개가 될 때까지 반복합니다. 그리고 [ A 누르면 실행 ] 슬롯에 맞추어 넣습니다.

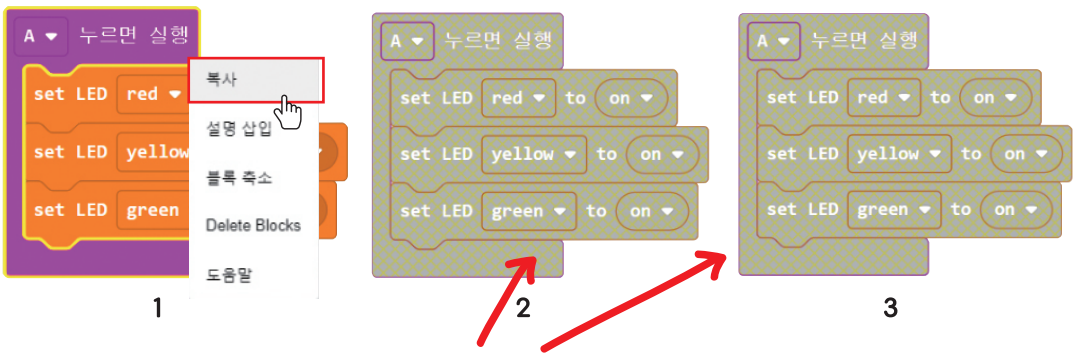





Step 6 두 번째와 세 번째 블록을 각각 'yellow'와 'green'으로 변경합니다.



Step 7 [ A 누르면 실행 ] 블록을 마우스 오른쪽 버튼으로 클릭하고 '복사'를 선택합니다. 같은 블록이 세 개가 될 때까지 반복합니다.

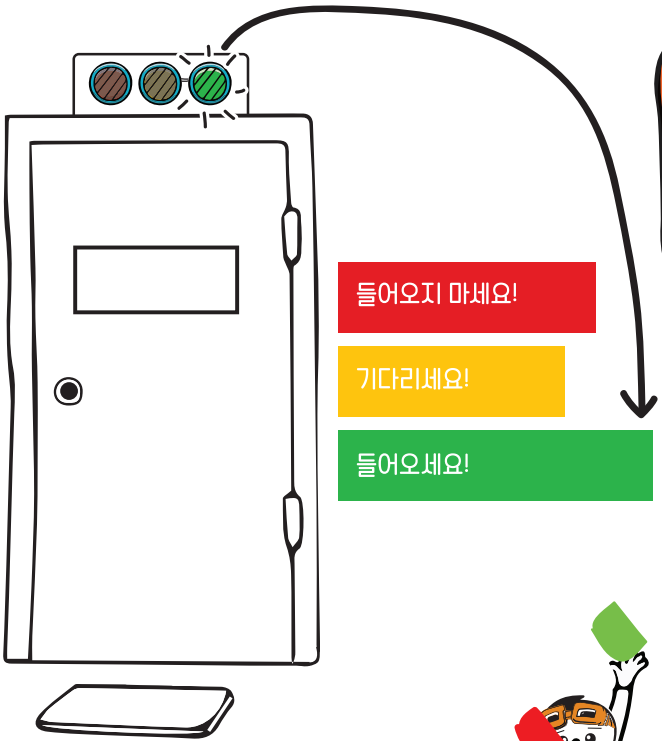



* 이 블록들은 [ A 누르면 실행 ] 블록과 중복되기 때문에 사용할 수 없고 동작하지 않습니다.

Step 8 두 번째와 세 번째 [ _ 누르면 실행 ] 블록의 'A'를 각각 'B'와 'A+B'로 변경합니다.

Step 9 다음과 같이 LED의 on/off 상태를 변경합니다.

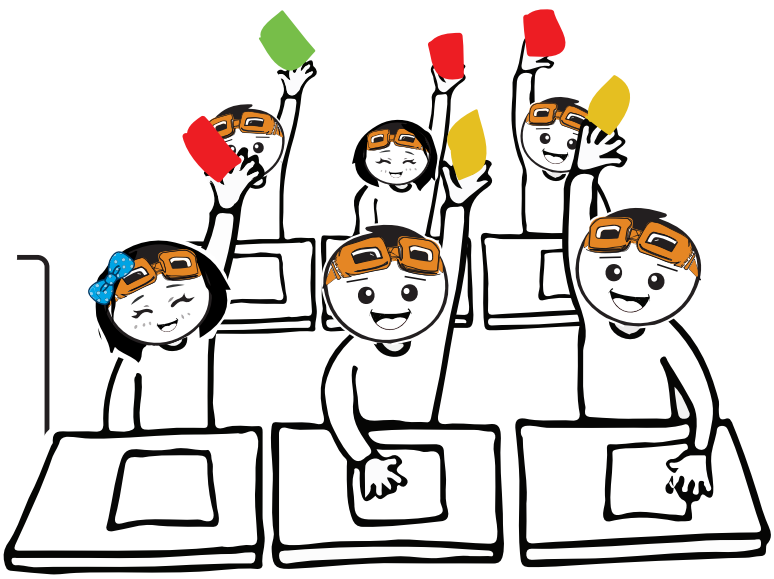

Step 10 코드를 EDU:BIT에 플래시하고 각각 A 버튼, B 버튼 그리고 A 버튼과 B 버튼을 동시에 눌렀을 때 어떤 일이 발생하는지 관찰해봅니다.




EDU:BIT를 의사표현 수단으로
사용할 수 있습니다.
다른 용도를 생각해 볼까요?

TRAFFIC LIGHT 피드백

- 잘 모르겠어요! 도와주세요.
- 공부/시도하는 중이에요.
- 이해했어요!



LED 또는 발광 다이오드는 디지털 출력 장치의 한 종류입니다.
오직 ON/OFF 두 가지 상태만 가능하며
일반적으로 ON은 1(one), OFF는 0(zero)으로 표현됩니다.

EDU:BIT가 타이머 기능을 하도록 프로그래밍 할 수 있습니다.

EDU:BIT를 흔들면 타이머가 작동합니다.

타이머가 시작되면 소리가 출력됩니다.

초록색 LED에 불이 들어옵니다.

노란색 LED에 불이 들어옵니다.

빨간색 LED에 불이 들어옵니다.

시간이 다 되면 와와와와아 멜로디가 시작됩니다.

빨간색 LED를 10번 토글합니다.

```
흔들림 감지하면 실행
    도 1 박자 출력
    set LED red to off
    set LED yellow to off
    set LED green to on
    일시중지 2000 (ms)
    set LED red to off
    set LED yellow to on
    set LED green to off
    일시중지 2000 (ms)
    set LED red to on
    set LED yellow to off
    set LED green to off
    일시중지 2000 (ms)
    와와와와아 멜로디 한 번 출력
    반복(repeat): 10 회
    실행
        Toggle LED red
        일시중지 500 (ms)
```

이 예시 코드에서, 각각의 LED는 2000ms(2초)동안 불이 들어옵니다.

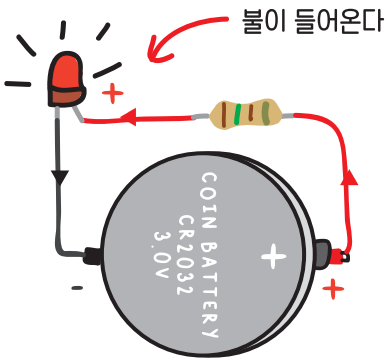
만약 각각의 LED에 1분 동안 불이 들어오게 하고 싶으면, 여기에 어떤 값을 입력해야 할까요?
TIP: 1분 = 60초

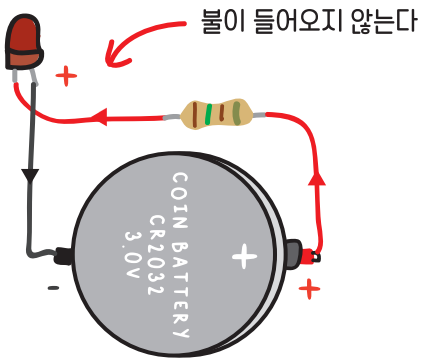
```
Toggle LED red
```

토글(Toggle)은 어떤 상태에서 다른 상태로 전환하는 것을 의미합니다. 만약 최근 상태가 ON이라면, 그것은 OFF로 바뀔 수 있습니다. 그러므로 우리가 LED를 반복적으로 토글 할 때 LED는 깜빡거리는 것처럼 보입니다.

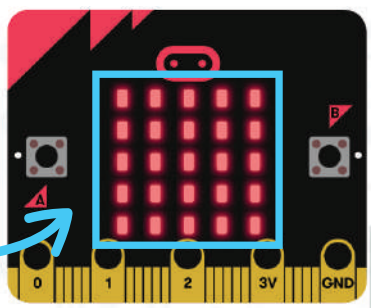
# 재미있는 사실!

**발광다이오드(LED)**는 전기로부터 빛을 생산하는 반도체 장치입니다. 두 개의 단자를 가지고 있으며, 각각 양극 단자와 음극 단자입니다.
LED가 올바른 극에 연결되면 전류가 흐르고, LED에 불이 들어옵니다.




micro:bit에서 사용되는 LED는 표면실장기술(SMT)을 기반으로 하며, 매우 작게 만들 수 있습니다.




micro:bit에 있는 것 외에도 EDU:BIT 보드에는 SMT LED가 41개 있습니다.
그것들을 모두 찾을 수 있나요?

# 응용 과제

| EDU:BIT가 '그림 보고 맞추기 게임'과 '몸짓 보고 맞추기 게임'을 위한 타이머와 점수 카운터 기능을 하도록 프로그래밍 해봅시다. | |
|---|---|
| 시작하면 실행 | 변수 Team A를 0으로 설정<br>변수 Team B를 0으로 설정 |
| A 누르면 실행<br>(노란색 버튼) | Team A 값 1 증가<br>Team A의 현재 점수 출력 |
| B 누르면 실행<br>(파란색 버튼) | Team B 값 1 증가<br>Team B의 현재 점수 출력 |
| A+B 누르면 실행<br>(노란색 + 파란색 버튼) | Team A와 Team B의 점수 스크롤 |
| 흔들림 감지하면 실행 | 1분 타이머가 시작되면 30초 동안<br>초록색 LED에 불이 들어오고,<br>다음 20초 동안 노란색 LED에 불이 들어옵니다.<br>마지막으로 10초 동안 빨간색 LED에<br>불이 들어옵니다.<br>시간이 다 되면 '와와와와아' 멜로디가 나오고<br>빨간색 LED가 10번 토글합니다. |

힌트입니다. 변수를 두 개 만들고 각각의 이름을 Team A와 Team B로 설정하세요.

# 게임하기

## 그림 보고 맞추기 게임


## 게임 방법:

- 학급을 Team A와 Team B로 나눕니다.
- Team A에서 한 사람이 랜덤으로 카드를 선택합니다.
  카드에 적힌 단어를 조용히 확인한 후, 1분 타이머를 시작하기 위해 EDU:BIT를 흔듭니다.
- 팀원들이 추측할 수 있도록 보드에 그림을 그립니다. 말이나 몸짓은 허용되지 않습니다.
- 만약 팀원 중 누군가가 시간이 끝나기 전에 정확하게 단어나 문장을 맞춘다면 Team A에게 1점이 주어집니다. 이때 A 버튼을 누르거나 노란색 버튼을 눌러 점수를 부여합니다.
- 만약 Team A가 실패하면 Team B에게 맞출 수 있는 기회가 주어지고, 맞추면 Team B에게 점수가 부여됩니다.
- 두 팀 모두 게임이 끝날 때까지 번갈아 가면서 그림을 그리고 추측합니다.
- 가장 많은 점수를 얻은 팀이 승리합니다!

스캔하세요

**주목!**

프린트할 수 있는 챌린지 단어 카드를 다운로드하려면 여기를 스캔하세요.
그림을 그리는 대신에 몸짓을 이용할 수 있습니다. 규칙은 그림 그리는 것을 제외하고 동일하게 적용하면서, 팀원들이 추측할 수 있도록 단서를 제공하는 몸짓을 보여주면 됩니다.

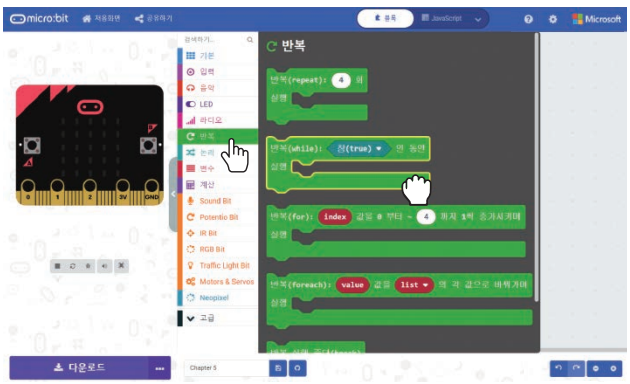
# CHAPTER 5

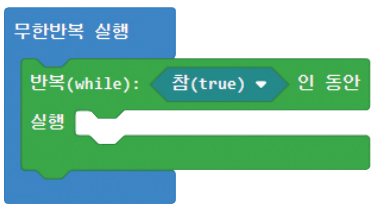
## 적외선 디지털 주사위를 굴려보자~

IR Bit

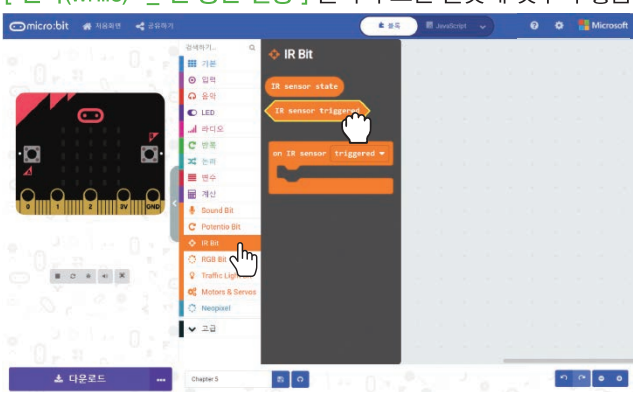
## 코딩을 시작해봅시다!

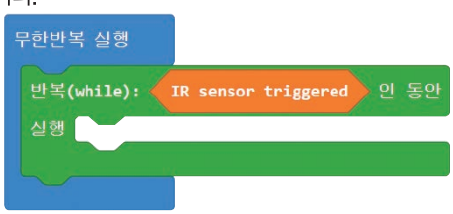
Step 1 MakeCode Editor에서 새 프로젝트를 생성하고 EDU:BIT 확장 프로그램을 추가합니다. (40페이지 참고)
[ 반복 ] 카테고리를 클릭하고 [ 반복(while): _ 인 동안 실행 ] 블록을 선택합니다.
그리고 [ 무한반복 실행 ] 슬롯에 맞추어 넣습니다.



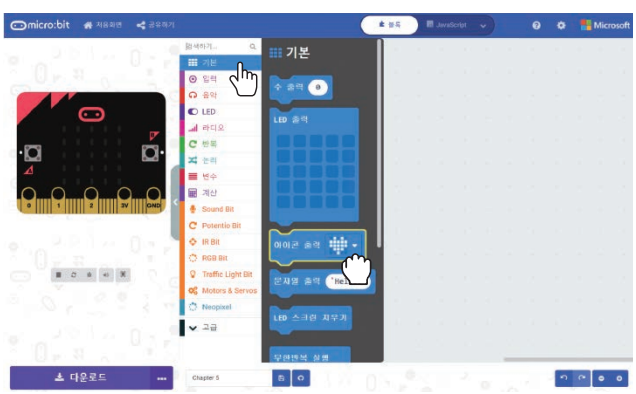



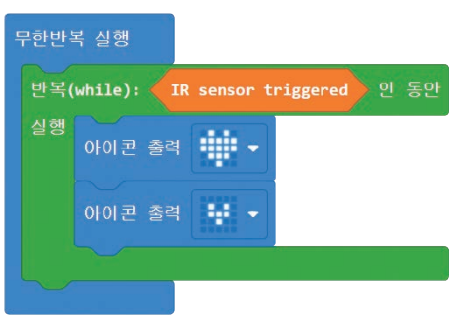

Step 2 [ IR Bit ] 카테고리를 클릭하고 [ IR sensor triggered ] 블록을 선택합니다.
[ 반복(while): _ 인 동안 실행 ] 블록의 조건 슬롯에 맞추어 넣습니다.



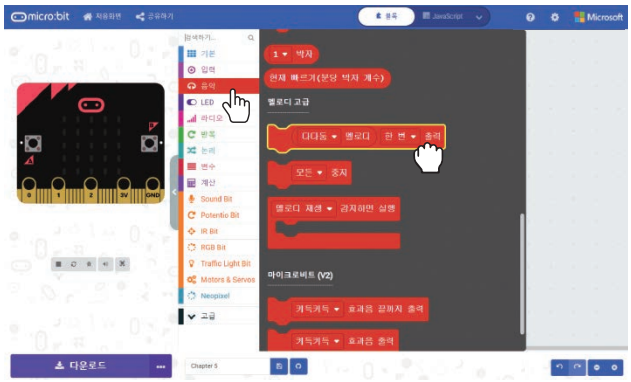



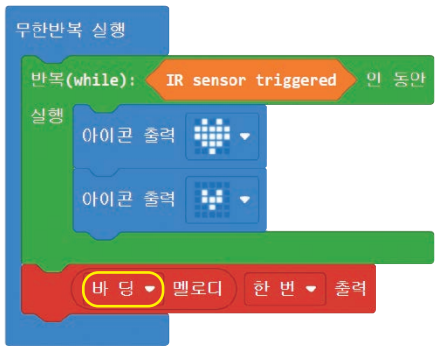

Step 3 [ 기본 ] 카테고리를 클릭하고 [ 아이콘 출력 ] 블록 두 개를 추가합니다.
하나는 '작은 하트' 아이콘으로 변경합니다. 블록들을 [ 반복(while): _ 인 동안 실행 ] 블록에 맞추어 넣습니다.



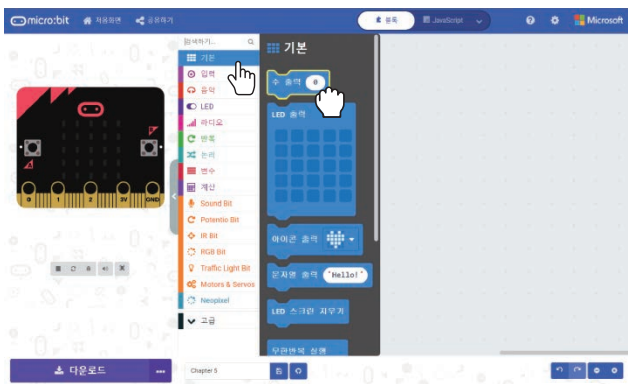



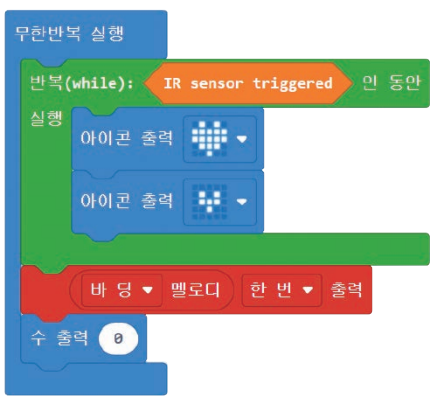
Step 4 [ 음악 ] 카테고리를 클릭하고 [ _ 멜로디 _ 출력 ] 블록을 선택합니다.
멜로디를 '다다둠'에서 '바 딩'으로 변경합니다.



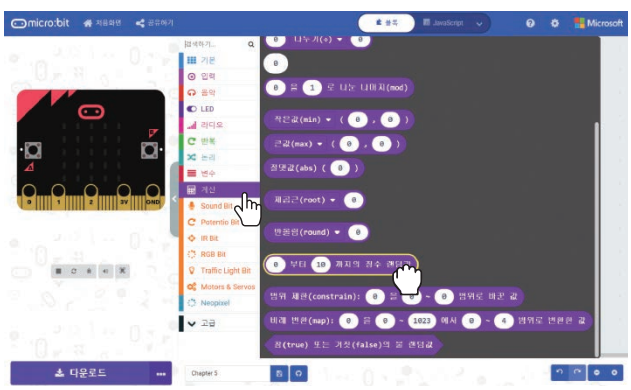


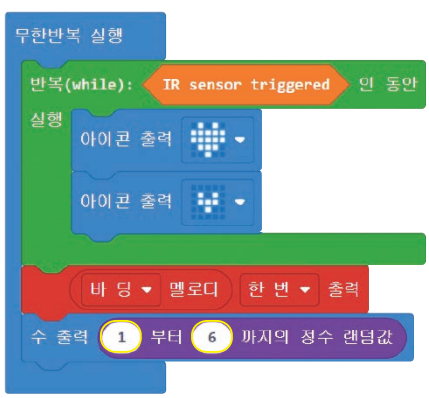

Step 5 [ 기본 ] 카테고리를 클릭하고 [ 수 출력 ] 블록을 선택합니다.



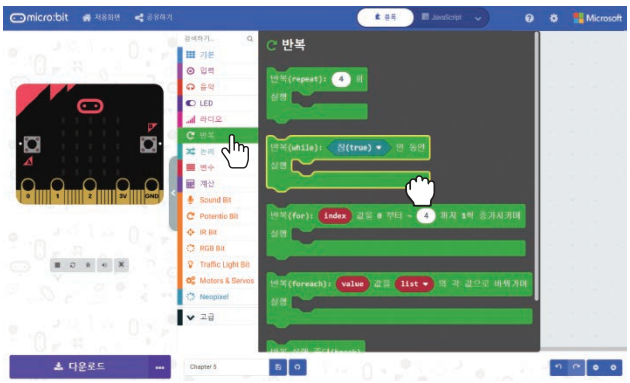


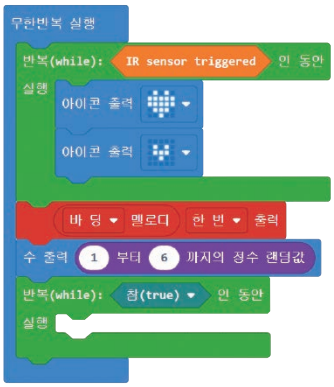

Step 6 [ 계산 ] 카테고리를 클릭하고 [ _ 부터 _ 까지의 정수 랜덤값 ] 블록을 선택합니다. 값을 1과 6으로 설정합니다.



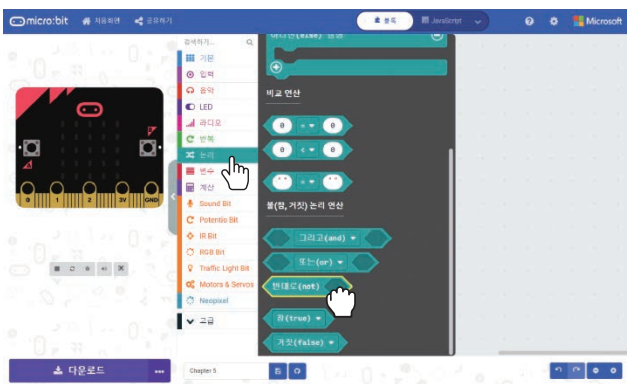


Step 7 [ 반복 ] 카테고리를 클릭하고 [ 반복(while): _ 인 동안 실행 ] 블록을 선택합니다.



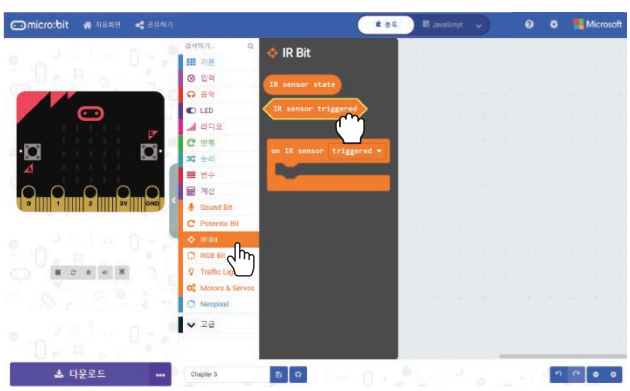



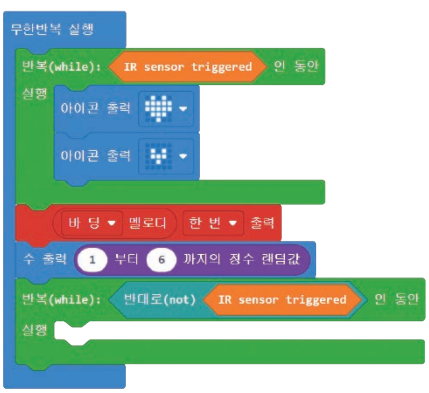

Step 8 [ 논리 ] 카테고리를 클릭하고 불(참, 거짓) 논리 연산의 [ 반대로(not) ] 블록을 선택하고 [ 반복(while): _ 인 동안 실행 ] 블록의 슬롯에 맞추어 넣습니다.





Step 9 [ IR Bit ] 카테고리를 클릭하고 [ IR sensor triggered ] 블록을 선택합니다.
블록을 [ 반대로(not) ] 블록의 빈 슬롯에 끼워 넣습니다.

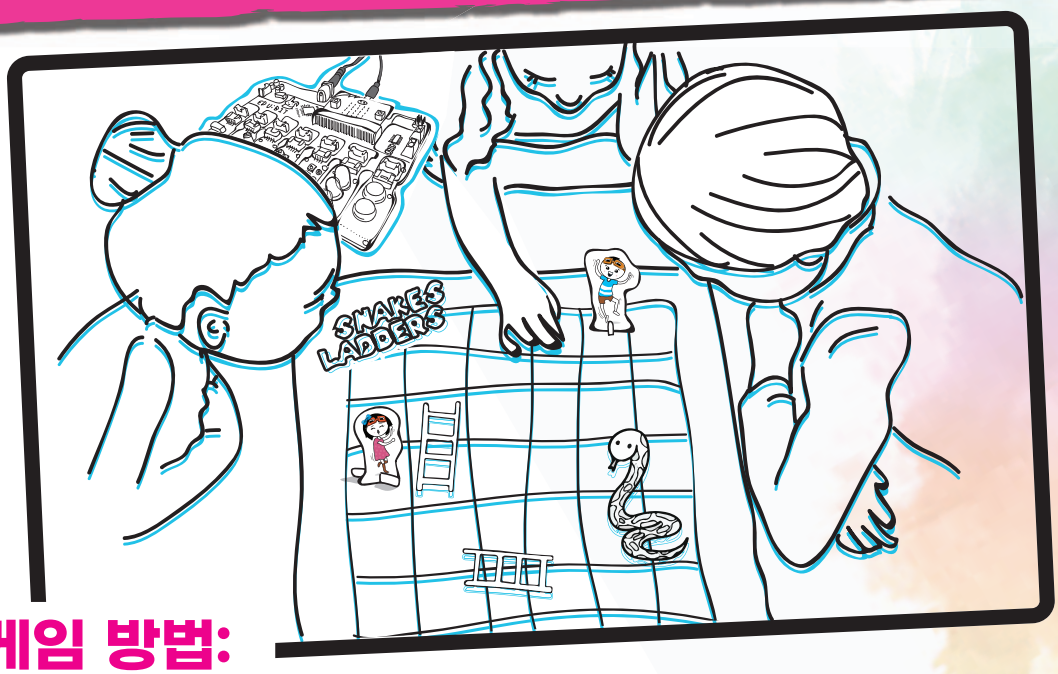



Step 10 EDU:BIT에 코드를 플래시합니다.

# 게임하기

## 뱀과 사다리 게임



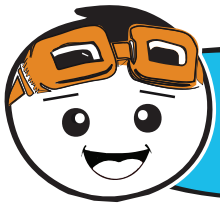

## 게임 방법:

- 캐릭터가 그려진 종이 말을 하나씩 선택하고 'Start Here'라고 적혀진 출발점에 둡니다.
- 돌아가면서 주사위를 굴립니다. 주사위를 굴리는 방법은 다음과 같습니다.
  손바닥을 IR Bit 위에 갖다 댑니다. 쿵쾅쿵쾅 심장이 뛰는 애니메이션이 보이면 손바닥을 치웁니다.
- LED 매트릭스에 보이는 숫자만큼 종이 말을 앞으로 옮깁니다.
  이때 숫자는 1에서 6사이의 값만 나올 수 있습니다.
- 만약 말이 사다리의 맨 아래쪽에 도착하면, 사다리 제일 윗부분으로 이동할 수 있습니다.
  만약 종이 말이 뱀의 머리와 만나면, 뱀의 꼬리 부분으로 내려가야 합니다.
- 제일 먼저 100에 도착한 사람이 게임의 승자가 됩니다. 자 이제 게임을 시작해봅시다!

**주목!**

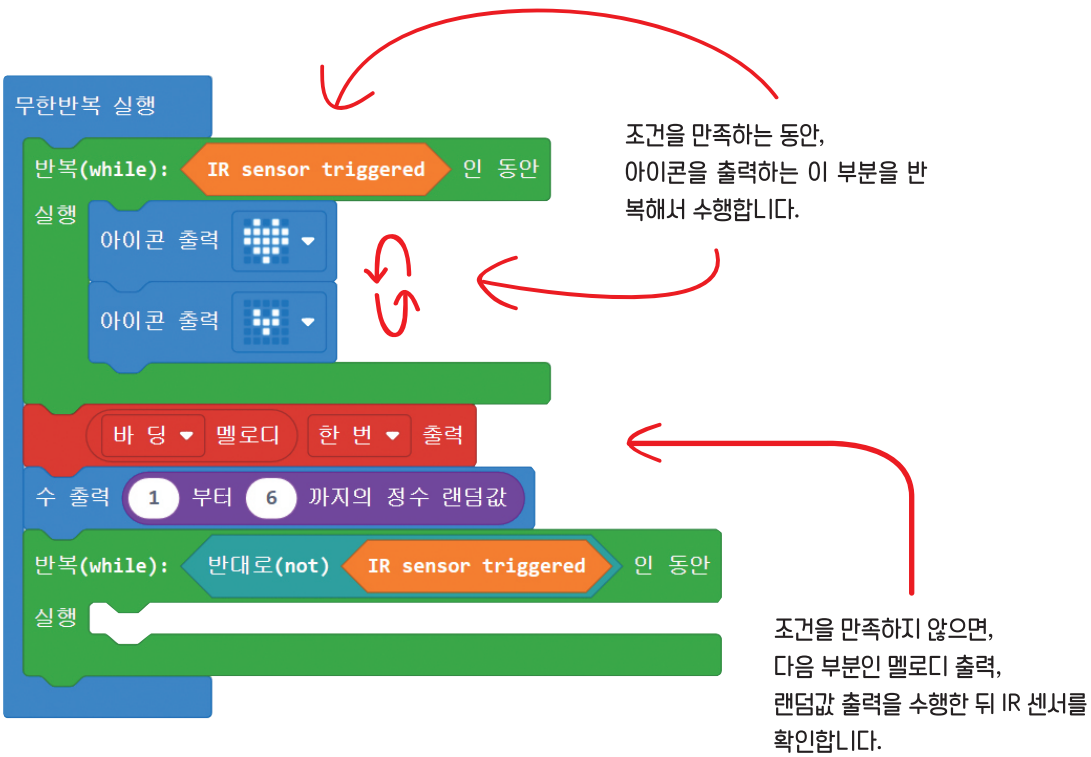
뱀과 사다리 게임 보드와 캐릭터가 그려진 종이 말은 박스 안에 제공되어 있습니다.
캐릭터와 밑받침을 분리한 후, 조립하여 종이 말을 만듭니다.

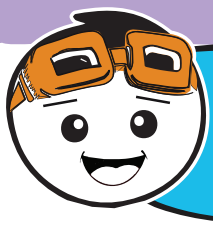
# 코딩 정복하기



앞에서 **[ 반복 ]** 카테고리의 **[ 반복(while): _ 인 동안 실행 ]** 블록을 사용했습니다.
while 반복문이 어떻게 동작하는지 알고 있나요?

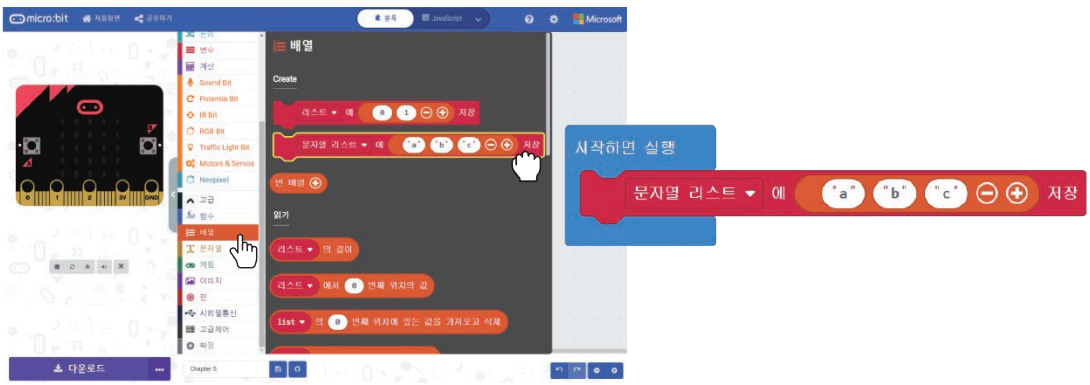
프로그램이 [ 반복(while): _ 인 동안 실행 ] 블록에 도착하면, 조건을 확인합니다.
조건을 만족하는 (또는 TRUE) 동안, 프로그램은 [ 반복(while): _ 인 동안 실행 ] 블록 안에 있는 블록들을 수행합니다. 계속 반복하다가 조건을 만족하지 않게 되는 경우 (또는 FALSE), 프로그램은 반복문을 빠져나가고 코드의 다음 블록을 수행합니다.





EDU:BIT를 디지털 주사위로 사용하는 것에 이어서, 점심으로 뭘 먹을지 고민하는 상황처럼 몇 가지 선택지 사이에서 고르지 못할 때 EDU:BIT가 도와주도록 코드를 수정할 수 있습니다.

Step 11 [ 고급 ]:[ 배열 ] 카테고리를 클릭하고 [ 문자열 리스트에 _ _ _ 저장] 블록을 선택합니다.
[ 기본 ] 카테고리의 [ 시작하면 실행 ] 슬롯에 블록을 맞추어 넣습니다.



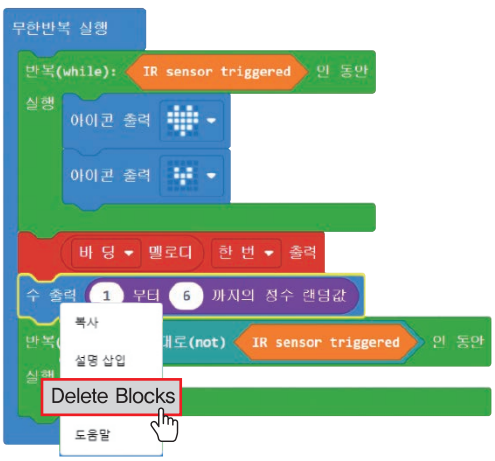

Step 12 [ 문자열 리스트 ]를 클릭하고 '변수 이름 바꾸기'를 선택합니다.
팝업창에 'Lunch options'를 입력하고 확인을 누릅니다.


Step 13 배열 블록을 하나씩 클릭해서 점심 메뉴 선택지를 입력합니다.

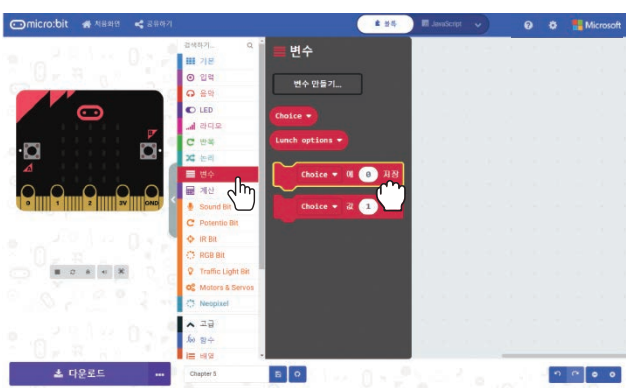

선택지를 추가하고 싶으면 이 버튼을 클릭합니다.

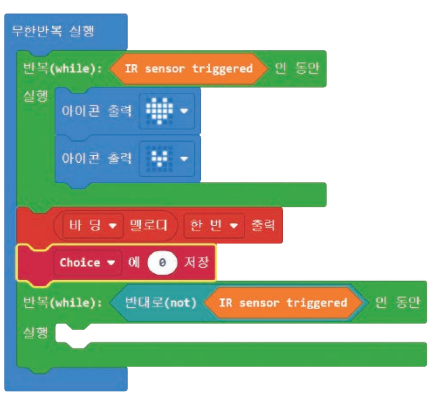
Step 14 [ 변수 ] 카테고리를 클릭하고 'Choice'라는 새 변수를 만듭니다.


Step 15 [ 수 출력 [ _ 부터 _ 까지의 정수 랜덤값 ] ] 블록을 마우스 오른쪽 버튼으로 클릭하고 'Delete Blocks'를 선택합니다.

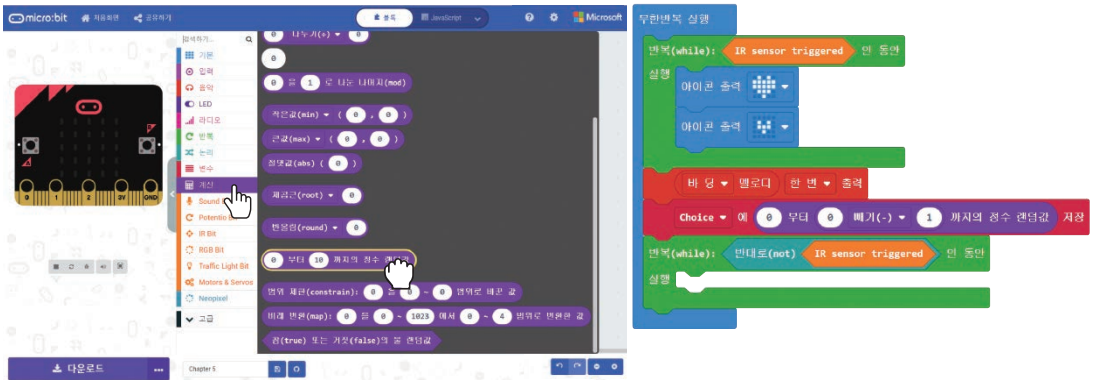



Step 16 [ 변수 ] 카테고리를 클릭하고 [ _ 에 _ 저장 ] 블록을 선택합니다. 블록을 [ _ 멜로디 _ 출력 ] 블록과 [ 반복(while): _ 인 동안 실행 ] 블록 사이에 넣습니다. 그리고 변수를 'Choice'로 변경합니다.

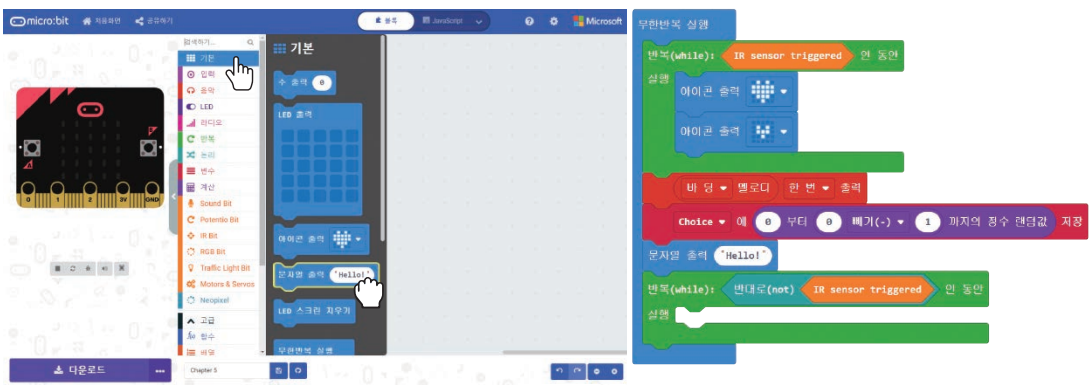


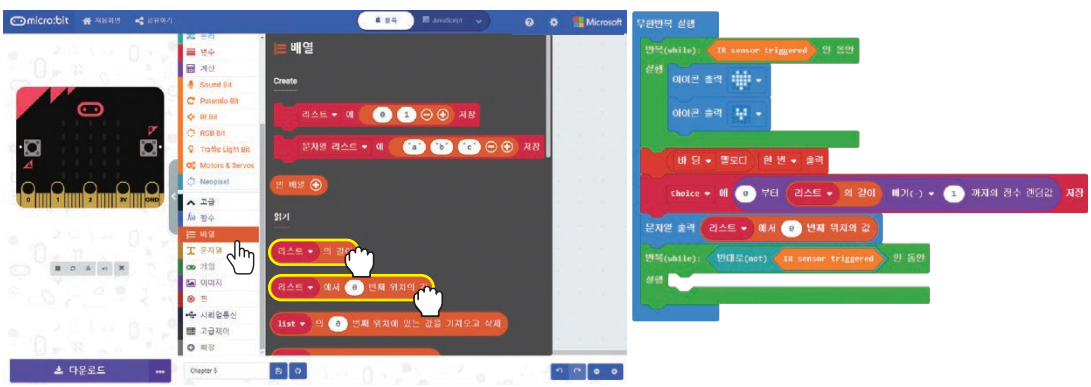
Step 17 [ 계산 ] 카테고리를 클릭하고 [ _ 부터 _ 까지의 정수 랜덤값 ]과 [ _ – _ ]블록을 추가합니다.
마지막 값은 1로 변경합니다.



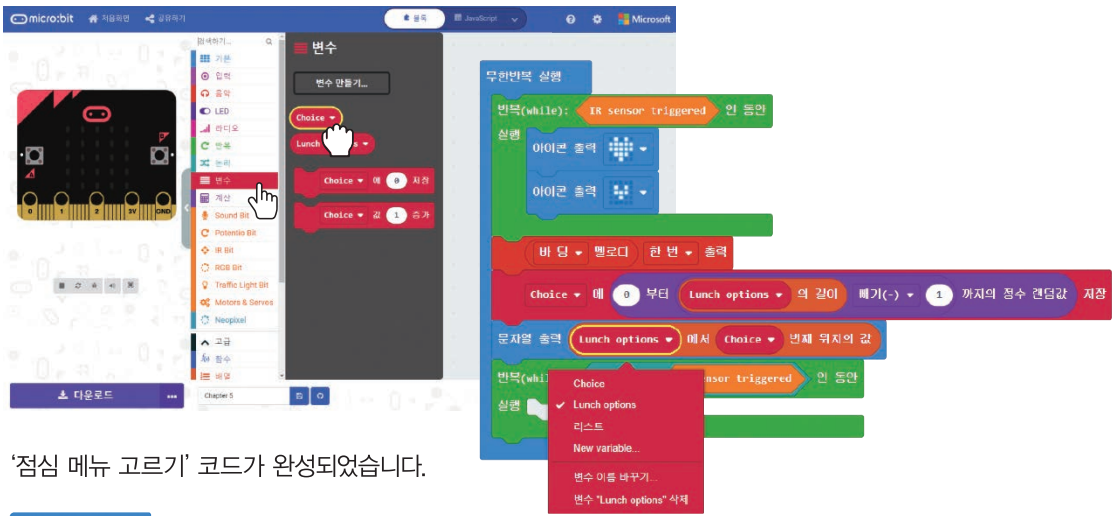

Step 18 [ 기본 ] 카테고리를 클릭하고 [ 문자열 출력 ] 블록을 선택합니다.




Step 19 [ 고급 ]:[ 배열 ] 카테고리를 클릭하고 [ _ 의 길이 ]와 [ _ 에서 _ 번째 위치의 값 ] 블록을 추가합니다.



Step 20 두 블록 모두 [ 리스트 ]를 클릭하고 변수를 'Lunch options'로 변경합니다.
마지막으로 [ 변수 ] 카테고리를 클릭하고 [ Choice ] 블록을 선택합니다.
[ _ 에서 _ 번째 위치의 값 ] 블록의 빈 슬롯에 맞추어 넣습니다.

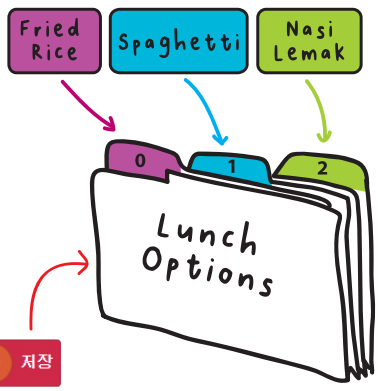


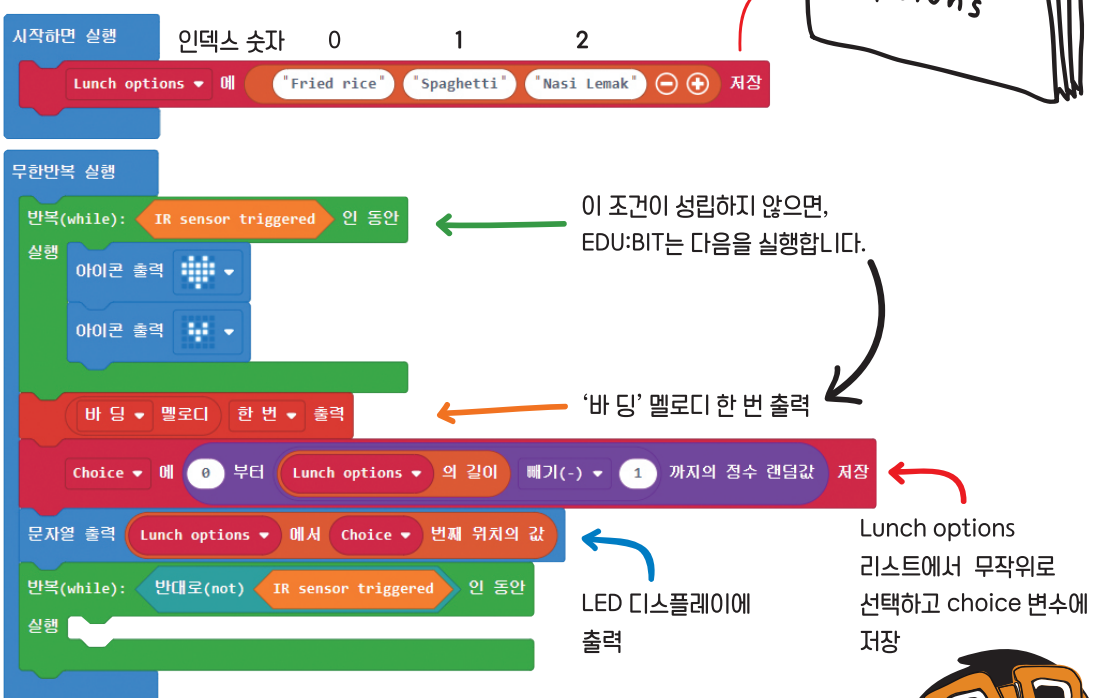

'점심 메뉴 고르기' 코드가 완성되었습니다.


다음부터 뭘 먹을지 고르지 못할 때,
IR Bit에 손을 가까이 가져갔다가 멀리 떨어짐으로써 EDU:BIT가 결정하도록 시킬 수 있습니다.
또한 친구와 어떤 게임을 할지 결정하는 것을 도와줄 수 있도록 코드를 수정할 수 있습니다.
그렇게 하려면 어느 부분을 수정해야 할까요?

# 코딩 정복하기

배열은 리스트 또는 연관된 변수들의 집합입니다.
여러 섹션이 있는 폴더로 생각할 수 있으며, 각 섹션은 한 가지 정보를 저장하는 데 사용됩니다.
리스트로부터 원소들을 추가하거나 제거하고 싶을 때 쉽게 코드를 수정하기 위해서 배열을 사용합니다.
예를 들어 이 코드에서 원소를 세 개 가진 배열을 만들고 이름을 'Lunch options'라고 설정했습니다.
각 원소에 해당하는 음식 항목을 쉽게 편집할 수 있습니다.
또는 간단하게 ⊕나 ⊖버튼을 클릭함으로써 선택지들을 추가하거나 리스트에 있는 원소의 수를 줄일 수 있습니다.

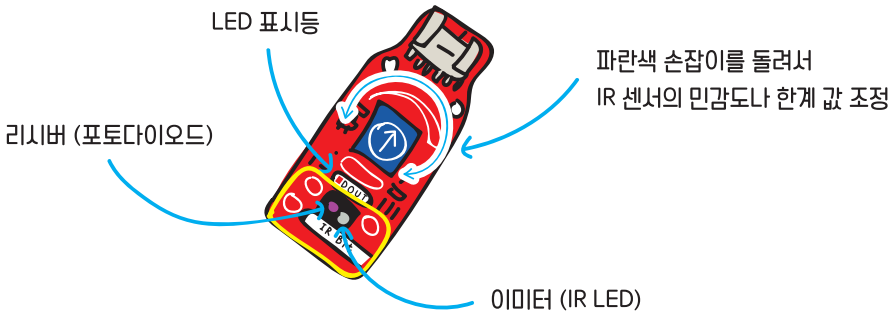



프로그래밍에서는 1 대신 0 부터 숫자를 셉니다.
그러므로 리스트에서 "Fried rice"의 인덱스 숫자는 0이고, 마지막 원소인 "Nasi Lemak"은 세번째 항목임에도 불구하고 인덱스 숫자가 2 입니다.

# 재미있는 사실!

**적외선(infrared, IR) 센서**는 일반적으로 장애물을 감지하는 데 사용됩니다.
IR 센서는 이미터(IR LED)와 리시버(포토다이오드)로 구성됩니다.

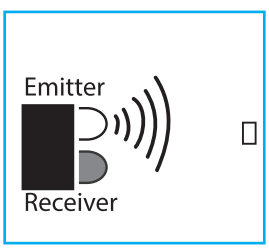



## 어떻게 작동할까요?

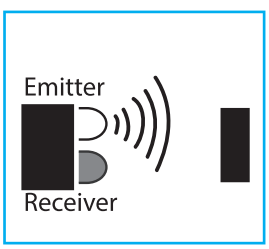
이미터(IR LED)가 적외선을 방출했을 때, 만약 물체가 센서 앞에 있으면 리시버(포토다이오드)에 반사됩니다.
한계값보다 더 많은 빛이 반사되면 IR Bit는 '트리거' 됩니다.
트리거 되었을 때, IR Bit에 있는 LED 표시등에 불이 들어옵니다.

만약 물체가 없거나 물체가 매우 멀리 있을 때, 적외선은 리시버에게 거의 반사되지 않습니다.
그러므로 IR Bit는 트리거 되지 않습니다.

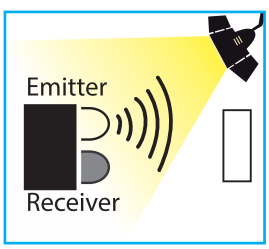
하지만, 적외선 센서는 다음과 같은 상황에서 잘 동작하지 않습니다.



물체가 너무 작을 때



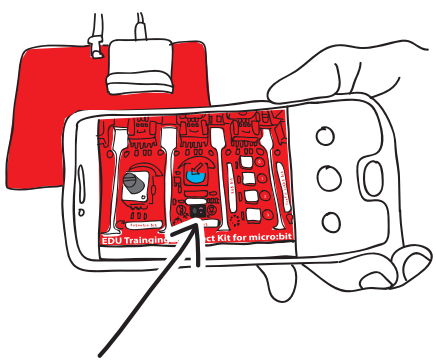
물체가 검은색이거나
어두운색일 때



빛의 간섭이 있을 때

## 알고 있나요?

적외선은 맨눈으로는 볼 수 없지만, 간단하게 휴대폰 카메라로
이미터를 보면 적외선을 볼 수 있습니다.

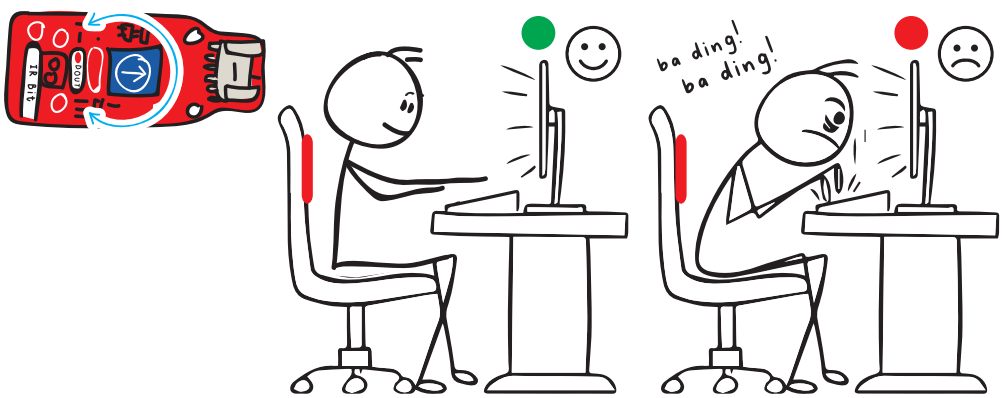


# 응용 과제

| EDU:BIT가 구부정한 자세 탐지기 기능을 하도록 프로그래밍 해봅시다. | |
|---|---|
| 시작하면 실행 | 리마인더를 스크롤합니다. – ‘Mind your posture’ |
| IR sensor triggered | LED 매트릭스에 웃는 표정을 보여주고<br>초록색 LED를 켭니다. |
| IR sensor not triggered | ‘바 딩’ 멜로디를 한 번 출력합니다.<br>LED 매트릭스에 슬픈 표정을 보여주고 빨간색 LED를 토글합니다. |

## 작동 원리:

아래에 보이는 것처럼 의자 등받이에 EDU:BIT를 부착합니다.
바른 자세로 편안하게 앉습니다.
IR Bit가 등을 감지해 LED 표시등에 불이 들어올 때까지 IR Bit에 있는 파란 손잡이를 돌려 조정합니다.
이 과정을 calibration(교정)이라고 말합니다.

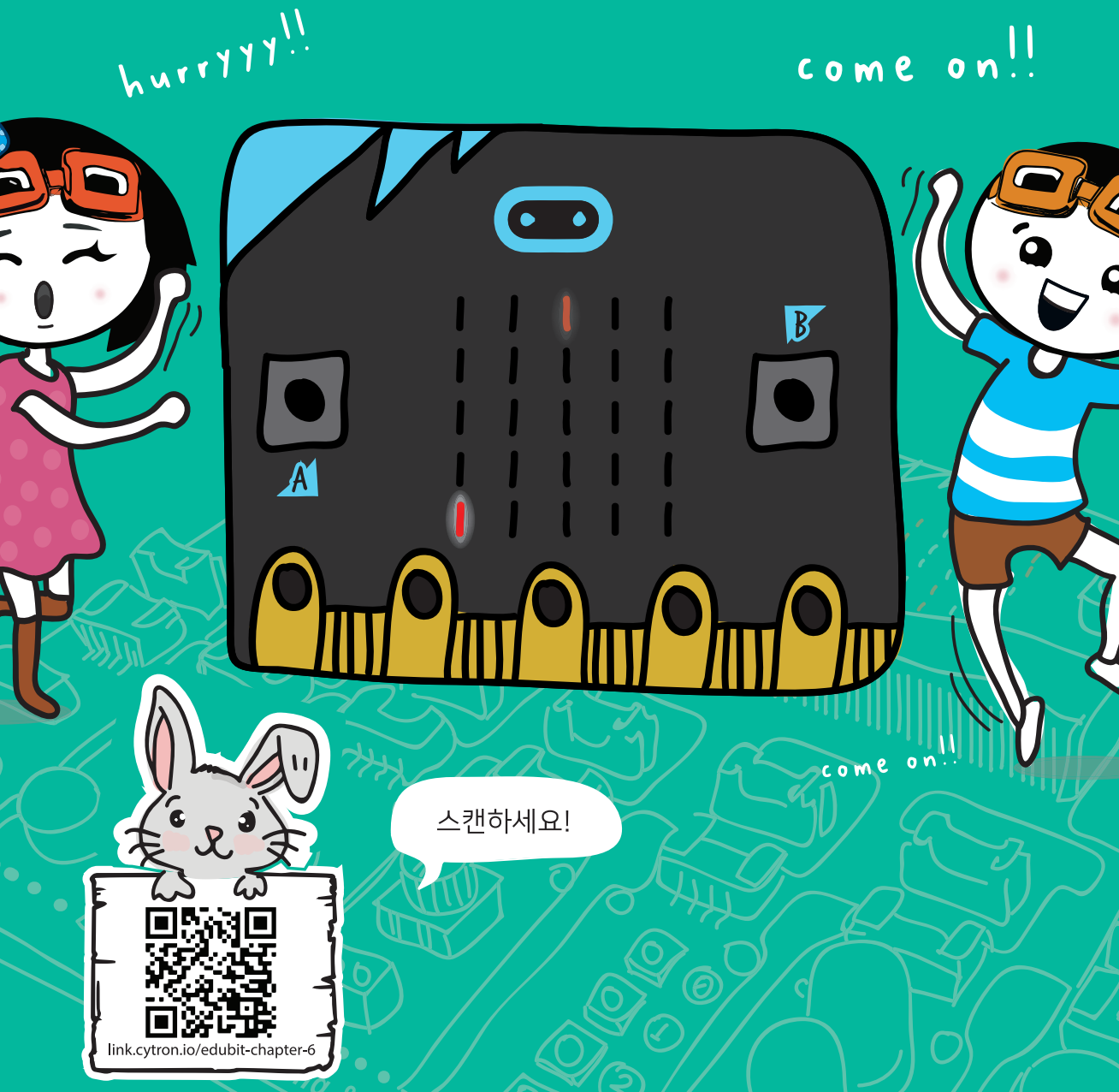



다른 색 옷을 입으면 IR Bit를 다시 교정해야 합니다. 왜일까요?

# CHAPTER 6

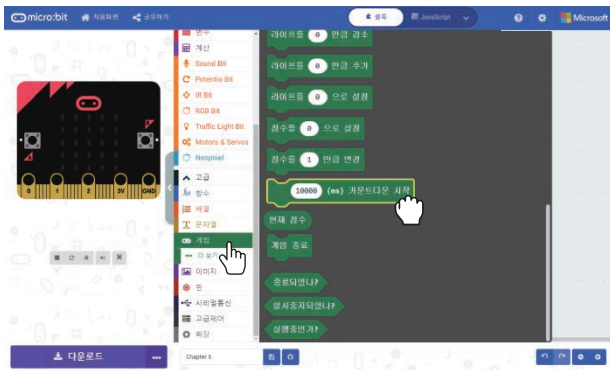
# 나 잡아봐라~

Potentio Bit

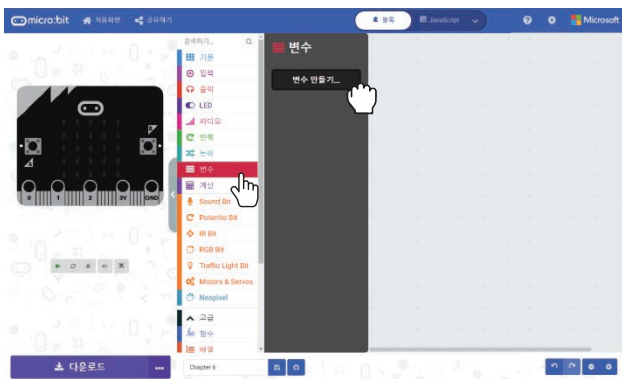



## 코딩을 시작해봅시다!

Step 1 MakeCode Editor에서, 새 프로젝트를 생성하고 EDU:BIT 확장 프로그램을 추가합니다. *(40페이지 참고)* [ 고급 ]을 누른 후 [ 게임 ] 카테고리를 클릭합니다. [ _ (ms) 카운트다운 시작 ] 블록을 선택한 후, [ 시작하면 실행 ] 블록에 맞추어 넣고 값을 30000으로 변경합니다.



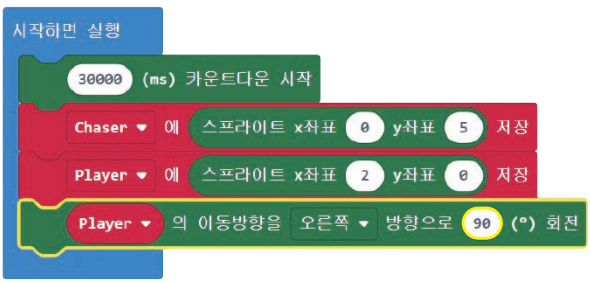

Step 2 [ 변수 ] 카테고리에서 [ **변수 만들기** ]를 클릭합니다. 팝업창에 'Player'를 입력하고 확인을 누릅니다. 'Chaser'라는 이름의 또 다른 변수도 생성합니다.



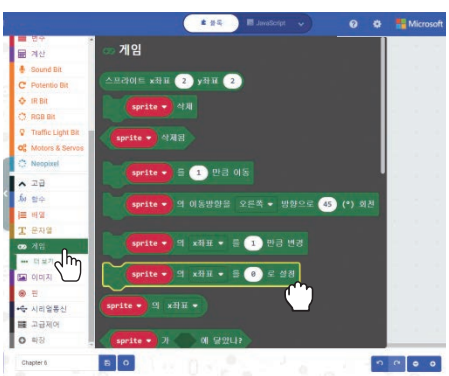

Step 3 [ 변수 ] 카테고리를 클릭하고 [ _ 에 _ 저장 ] 블록을 선택합니다. [ _ 에 _ 저장 ] 블록을 복사하고 둘 다 [ 시작하면 실행 ] 블록에 맞추어 넣습니다. 변수들은 각각 'Chaser'와 'Player'로 설정합니다.

Step 4 [ 고급 ]을 누른 후 [ 게임 ] 카테고리를 클릭합니다. [ 스프라이트 x좌표 _ y좌표 _ ] 블록을 선택합니다. 복사해서 [ _ 에 _ 저장 ] 블록에 맞추어 넣습니다. 'Chaser'는 x좌표 0 y좌표 5, 'Player'는 x좌표 2 y좌표 0으로 값을 변경합니다.

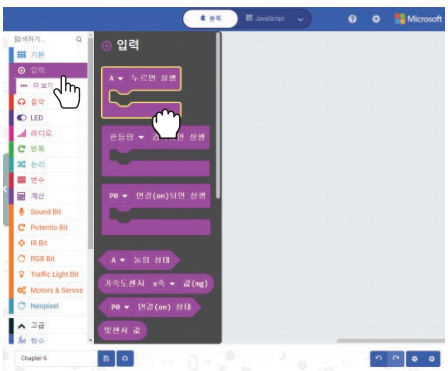

Step 5 [ 고급 ]:[ 게임 ] 카테고리를 클릭하고 [ _ 의 이동방향을 _ 방향으로 _ ( ° ) 회전 ] 블록을 선택합니다. [ 시작하면 실행 ] 슬롯에 블록을 맞추어 넣습니다. 변수는 'Player', 각도는 90으로 설정합니다.



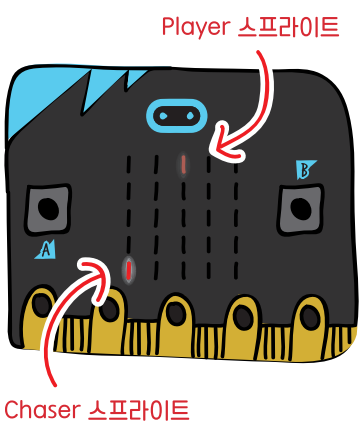

Step 6 [ 고급 ]:[ 게임 ] 카테고리를 클릭하고 [ _ 의 _ 를 _ 로 설정 ] 블록을 선택합니다.
변수는 'Player'를 선택하고 'x좌표'를 '스프라이트 밝기'로 변경한 후, 값을 50으로 설정합니다.




Step 7 [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다. 블록을 복사하고 복사된 블록은 B' 버튼으로 변경합니다.

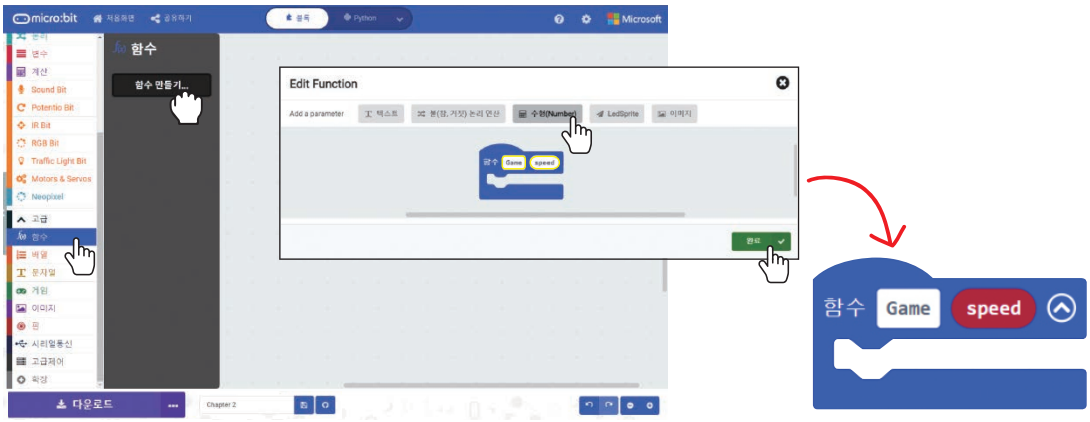



Step 8 [ 게임 ] 카테고리를 클릭하고 [ _ 를 _ 만큼 이동 ] 블록을 선택합니다. 블록을 복사하고 [ _ 누르면 실행 ] 슬롯에 맞추어 넣습니다. 두 블록 모두 변수는 'Player'로 선택하고 값을 [ A 누르면 실행 ] 블록은 −1, [ B 누르면 실행 ] 블록은 1로 변경합니다.

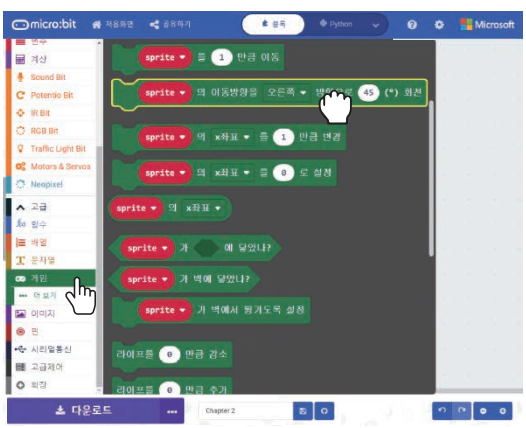



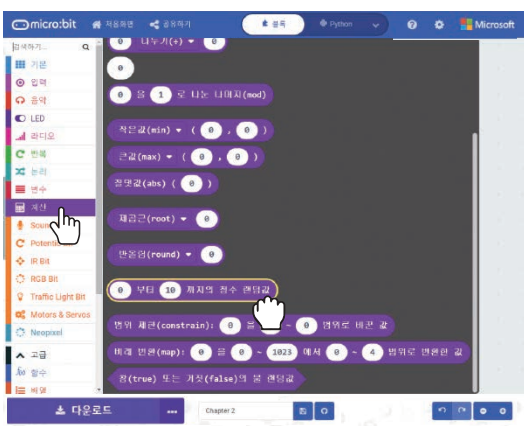

EDU:BIT에 코드를 플래시합니다.
파란색 버튼 (B 버튼)을 눌렀을 때,
더 어두운 LED 빛이 아래로 움직이나요?
그것이 Player 스프라이트 입니다.
스프라이트는 이동 방향을 바꾸거나 이동시킬 수 있는 '작은 LED 점'을 의미합니다.
만약 노란색 버튼 (A 버튼)을
누르면 어떤 일이 발생할까요?

Step 9 [ 고급 ]에서 [ 함수 ] 카테고리를 클릭하고 [ 함수 만들기 ]를 선택합니다. Edit Function 창에서 'doSomething'은 'Game'으로 변경합니다. 파라미터를 추가하기 위해 수형(Number)을 클릭하고 함수 블록에서 'num'을 'speed'로 변경합니다. 그 다음 **완료**를 클릭합니다.




Step 10 아래에 보이는 것처럼 코드를 구성합니다. [ 고급 ]:[ 게임 ] 카테고리에서 [ _ 의 이동방향을 _ 방향으로 _ ( ° ) 회전 ], [ _ 를 _ 만큼 이동 ], [ _ 가 벽에서 튕기도록 설정 ] 블록을 선택합니다. [ 계산 ] 카테고리에서 [ _ 부터 _ 까지의 정수 랜덤값 ] 블록을 추가합니다. 변수는 'Chaser'로 설정하고 값은 90으로 변경해야 합니다.

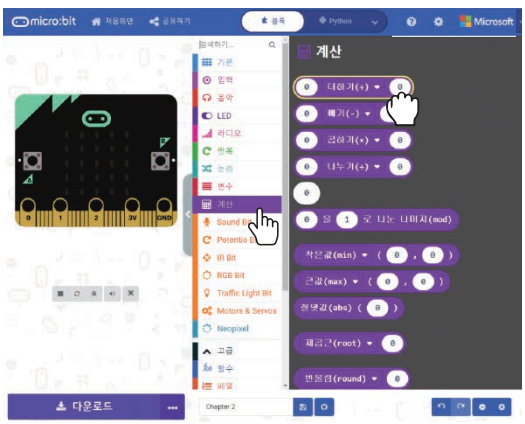




Step 11 [ 논리 ] 카테고리에서 [ 만약(if) _ 이면(then) 실행 ] 블록을 두 개 추가합니다.


Step 12 아래에 보이는 것 처럼 코드를 구성합니다. [ 고급 ]:[ 게임 ] 카테고리에서 [ _ 가 _ 에 닿았나? ], [ _ 가 벽에 닿았나? ], [ 점수를 _ 으로 설정 ], [ 현재 점수 ], [ 게임 종료 ] 블록을 선택합니다. [ 계산 ] 카테고리에서 [ _ 더하기(+) _ ]블록을 추가하고 값을 1로 변경합니다.



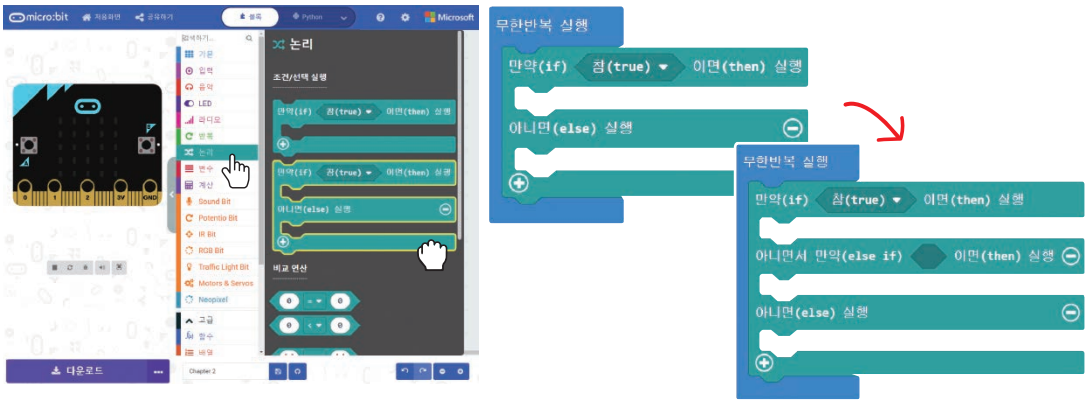
Step 13 [ sprite ] 블록 둘 다 'Chaser'로 변경합니다. [ 변수 ] 카테고리를 클릭하고 [ Player ] 블록을 선택합니다. [ _ 가 _ 에 닿았나? ] 블록의 빈 슬롯에 맞추어 넣습니다.


Step 14 [ 기본 ] 카테고리를 클릭하고 [ 일시중지 _ (ms) ] 블록을 선택해 코드에 추가합니다.
함수 블록에서 [ speed ]를 클릭하고, 잡은 채로 드래그해서 [ 일시중지 _ (ms) ] 블록의 빈 슬롯에 맞추어 넣습니다.

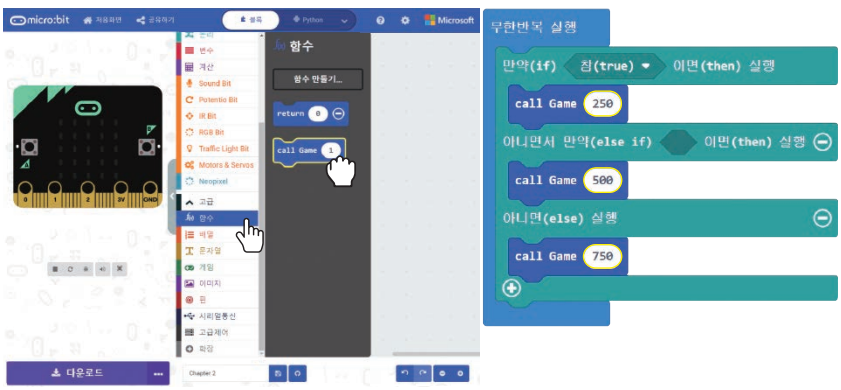

Chaser가 Player에 닿았을 때 멜로디가 나오게 설정하여 게임에 흥미요소를 추가할 수 있습니다. 어떤 블록을 어디에 추가해야 할까요?

## 이제, 게임에 난이도를 추가해봅시다!

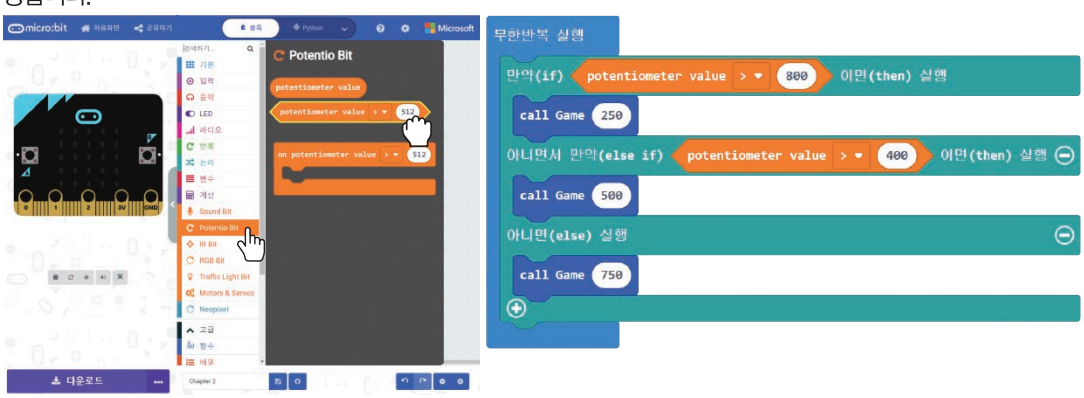
Step 15 [ 논리 ] 카테고리를 클릭하고 [ if_then_else ] 블록을 선택합니다.
[ 무한반복 실행 ] 슬롯에 맞추어 넣습니다. 다른 조건을 추가하기 위해 (+) 버튼을 클릭합니다.




Step 16 [ 함수 ] 카테고리를 클릭하고 [ call Game _ ] 블록을 선택합니다. 블록을 복사하고 블록들을 [ if_then_else ] 슬롯에 각각 맞추어 넣습니다. [ call Game _ ] 블록들의 값을 각각 250, 500, 750으로 변경합니다.




Step 17 [ Potentio Bit ] 카테고리를 클릭하고 [ potentiometer value 〉 _ ] 블록을 선택합니다. 복사한 후 [ if_then_else ] 블록의 조건 슬롯에 맞추어 넣습니다. 첫 번째 블록은 값을 800, 두 번째 블록은 값을 400으로 설정합니다.



완성된 코드입니다.

애니메이션을 보여주고,
30초 카운트다운 타이머 시작

Chaser 스프라이트와 Player
스프라이트 지정된 좌표에 생성

Player 스프라이트 90도 회전시켜
위아래로 움직이게 설정

Chaser 스프라이트와 구별하기 위해
Player 스프라이트의 밝기 조정

A 버튼과 B 버튼을 누를 때, Player
스프라이트가 위나 아래로 움직임

Game 함수는 Chaser 스프라이트 제어

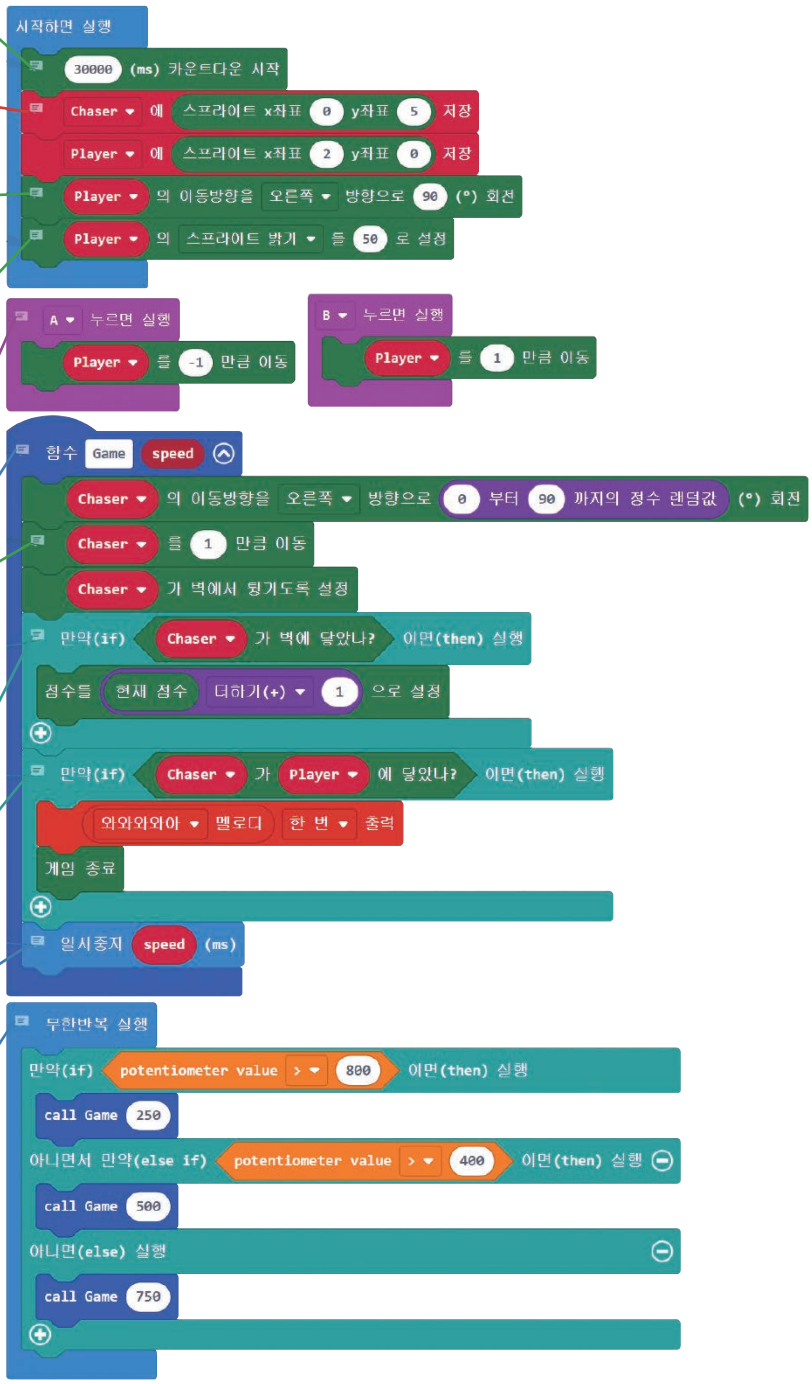
Chaser 스프라이트를 0에서 90 사이의
랜덤값만큼 오른쪽으로 회전해서
1만큼 이동하게 설정
만약 Chaser 스프라이트가 벽에 닿으면
튕기도록 설정

Chaser 스프라이트가 벽에 닿을 때마다
"현재 점수" 변수는 1점씩 증가

만약 Chaser 스프라이트가
Player 스프라이트에 닿으면,
와와와와아 멜로디가 나오면서 화면에
Game Over 애니메이션과 점수 표시

"speed" 변수값에 따른 ms 만큼
프로그램 일시 중지

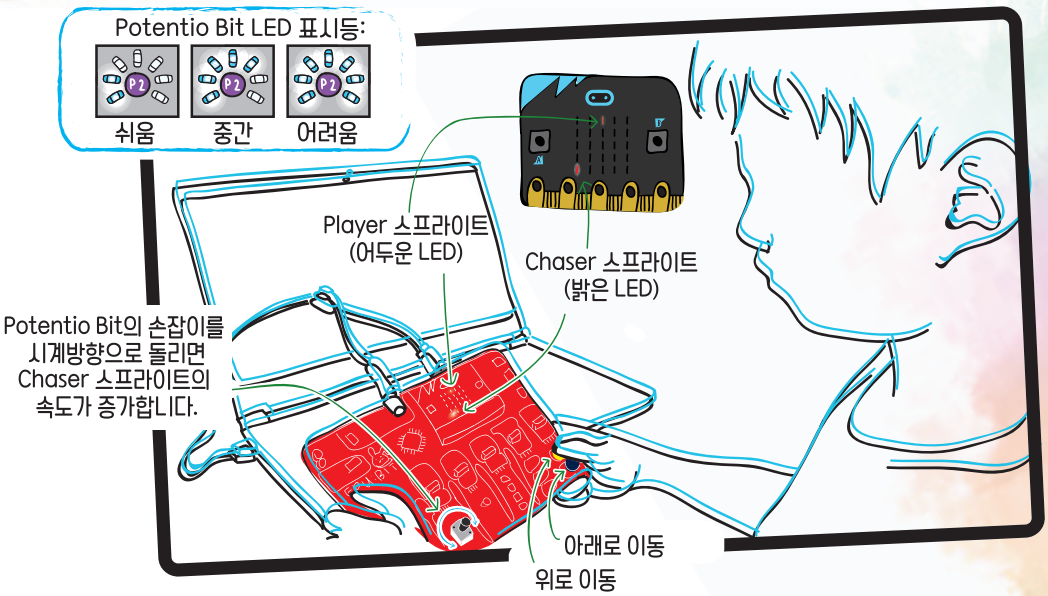
항상 potentio bit 값 확인
potentiometer 값이 800보다 큰 경우,
"Game" 함수를 호출하고
"speed" 변수값을 250ms로 설정
potentiometer 값이 400보다 큰 경우
(즉, 400초과 800이하),
"Game" 함수를 호출하고
"speed" 변수값은 500ms로 설정
이외에는 "Game" 함수를 호출하고
"speed" 변수값은 750ms로 설정




Step 18 EDU:BIT에 완성된 코드를 업로드하고 친구들과 '나 잡아봐라~' 게임을 해보세요.

# 게임하기

나 잡아봐라~




## 게임 방법:

게임이 시작되면 Chaser 스프라이트는 랜덤 방향으로 움직입니다.

Chaser 스프라이트를 피해서 Player 스프라이트를 위나 아래로 움직입니다.
노란색 버튼 (A 버튼)을 누르면 위로 움직이고 파란색 버튼 (B 버튼)을 누르면 아래로 움직입니다.

만약 Player가 Chaser와 닿거나 주어진 30초가 지나면 게임이 종료됩니다.

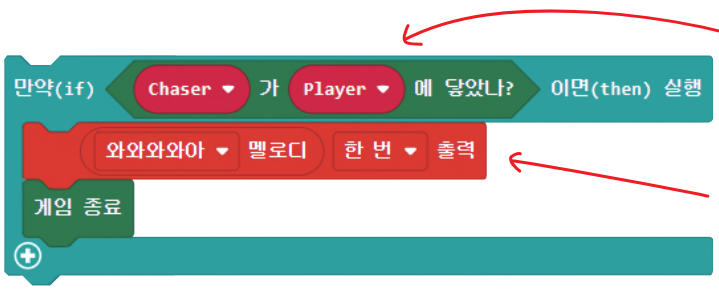
Chaser 스프라이트가 벽에 닿을 때마다 점수를 1점씩 얻을 수 있습니다.
가장 높은 점수를 얻은 Player가 게임의 승자가 됩니다. 자 이제 게임을 시작해봅시다!

**팁!**

#1 _ 30초 시간 제한안에 많은 점수를 얻기 위해서, Chaser 스프라이트의 속도를 증가시키면 벽에 더 자주 닿을 수 있습니다.
#2 _ 게임이 끝난 후, A와 B 버튼을 동시에 누르면 새 게임이 시작됩니다. 이것은 [ 게임 ] 블록에 내장된 기능입니다.

# 코딩 정복하기

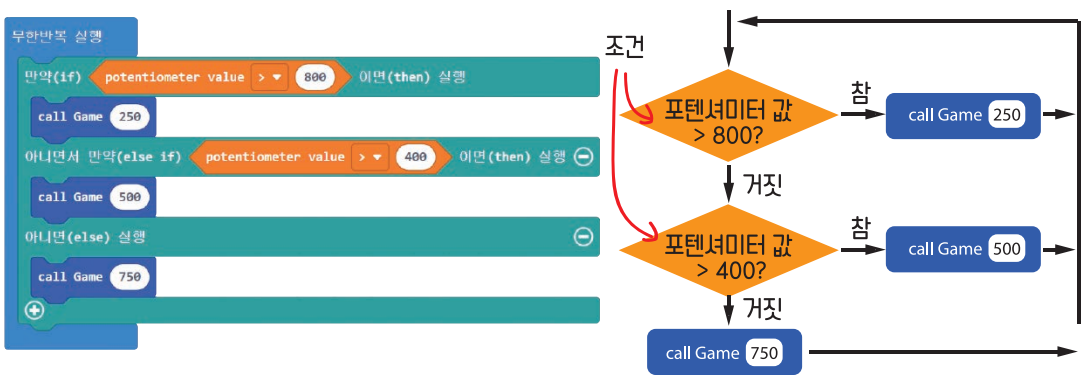
프로그래밍에서, 판단하기 위해 if 조건문을 사용합니다.
MakeCode에서 [ 논리 ] 카테고리에 있는 [ 만약(if) _ 이면(then) 실행 ]이나 [ 만약(if) _ 이면(then) 실행 아니면(else) 실행 ] 조건 블록들을 사용해 조건을 만들 수 있습니다. 프로그램은 조건 상태를 확인하고 참(TRUE)이면 조건 블록에 있는 코드를 실행합니다.
만약 거짓(FALSE)이면 코드의 다음 블록으로 이동합니다.




**만약(if)** 이 조건이 성립
(조건: Chaser 스프라이트가 Player 스프라이트에 닿음)
**이면(then)** 이 부분 실행
(와와와와아 멜로디 출력,
'game over' 애니메이션 출력)

여러 가지 조건이 있을 때, 프로그램은 위에서 아래로 순차적으로 조건을 확인하고, 처음으로 참(TRUE)을 반환하는 조건에 해당하는 코드를 실행합니다. 그러므로, 조건이 더 위에 위치할수록, 아래 조건에 비해 우선순위가 더 높습니다.

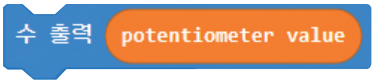
예를 들어, 게임에서 이 코드는 설정된 한계값과 포텐셔미터(potentiometer)의 값을 비교함으로써 Chaser 스프라이트의 움직이는 속도를 결정합니다.




**만약(if)** potentiometer value가 800보다 크면, speed를 250ms로 설정해 Game 함수를 호출합니다.
**아니면서 만약(else if)** potentiometer value가 400보다 크면, speed를 500ms로 설정해 Game 함수를 호출합니다.
**아니면(else)** speed를 750ms로 설정해 Game 함수를 호출합니다.

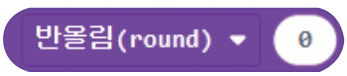
# 더 많은 블록 탐구하기

#1 현재 potentiometer 값을 읽고 출력하기 위해 [ 기본 ] 카테고리의 [ 수 출력 ] 블록과 [ Potentio Bit ] 카테고리의 [ potentiometer value ] 블록을 사용합니다.




#2 potentiometer는 0과 1023사이의 값을 반환합니다.
읽어온 값을 더 의미 있는 다른 범위로 비례 변환하기 위해 [ 고급 ]:[ 핀 ] 카테고리에서
[ 비례 변환(map): _ 최소 _ 최대 _ 에서 최소 _ 최대 _ 범위로 변환한 값 ] 블록을 사용합니다.


#3 비례 변환 블록은 소수점이 있는 숫자로 반환됩니다. (예를 들어 1.68, 3.998)
반올림하기 위해서, [ 계산 ] 카테고리에서 [ 반올림(round) ] 블록을 사용합니다.

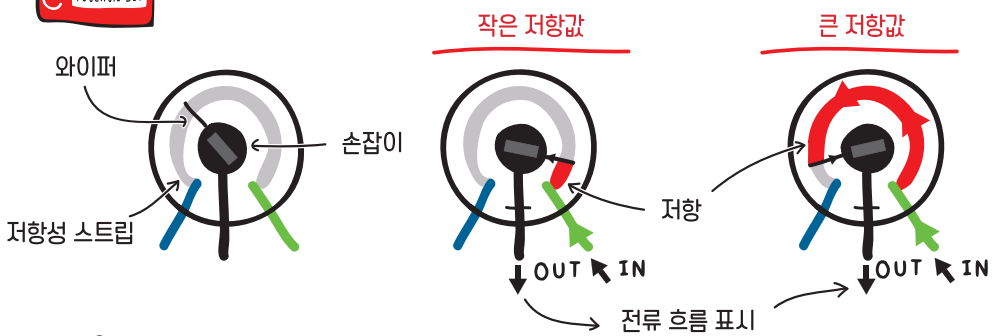


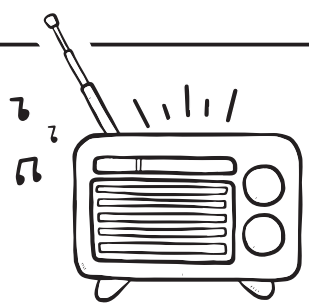

0에서 7범위의 Potentio Bit로 비례 변환하는 예시 코드입니다.
그 값은 반올림해서 LED 매트릭스에 표시됩니다.


# 재미있는 사실!


POT라고 불리는 **포텐셔미터(전위차계, Potentiometer)**는
손잡이나 슬라이더를 사용해서 쉽게 조절할 수 있는 저항을 가진 가변 저항기입니다.

만약 10,000Ω 포텐셔미터가 있다면, 와이퍼 위치를 변경함으로써 저항값을 0Ω에서 10,000Ω사이로 만들 수 있습니다.

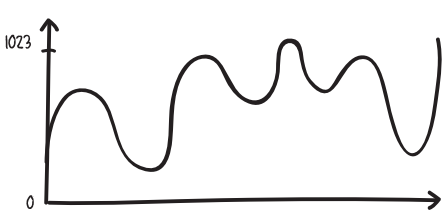


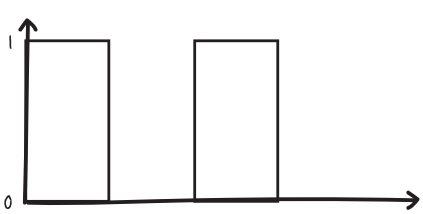
**포텐셔미터가 적용된 것**

- 스피커의 오디오 볼륨
- 라디오의 주파수 조절
- 온수기 온도 조절

EDU:BIT에 있는 포텐셔미터는 아날로그 입력 장치의 한 종류입니다.
전위(electric potential)를 측정하고 측정된 0V에서 3.3V 사이의 전압을 0에서 1023사이의 정수값으로 변환합니다.




아날로그신호



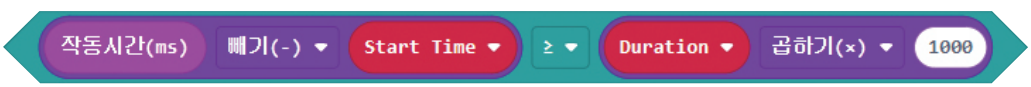

디지털신호

# 응용 과제

EDU:BIT가 타이머 역할을 하도록 프로그래밍 해봅시다.
0 ~ 60초 사이로 지속 시간을 조정하기 위해 Potentio Bit를 사용합니다.
타이머를 시작하려면 A 버튼을 누르고 다시 시작하려면 B 버튼을 누릅니다.

| | |
|---|---|
| 시작하면 실행 | Mode에 0 저장 |
| A 누르면 실행<br>(노란색 버튼) | Mode에 1 저장<br>Start Time에 작동시간 저장<br>웃는 표정 아이콘 출력 |
| B 누르면 실행<br>(파란색 버튼) | Mode에 0 저장 |
| 무한반복 실행 | 항상 Mode를 확인합니다.<br>● 만약 Mode=0이면, 포텐셔미터 측정 값을 최소 0에서 최대 60으로 비례 변환한 후 반올림해서 Duration에 저장합니다. 그리고 LED 매트릭스에 Duration을 출력합니다.<br>● else if(아니면서 만약) Mode=1이면,<br>(작동시간 − StartTime) ≥ (Duration × 1000) 인지 아닌지 확인합니다. 만약 참(True)이면, 와와와와아 멜로디를 출력하고 Mode를 2로 설정합니다.<br>● else(아니면) 슬픈 표정 아이콘을 출력합니다. |

힌트입니다.
#1 세 가지 변수를 만들어야 합니다. (Mode, Start Time, Duration)
#2 작동시간(ms) 블록은 [ 입력 ] 카테고리에 있습니다.
#3 시간이 다 되었는지 아닌지를 확인하기 위해 이 조건식을 사용합니다.



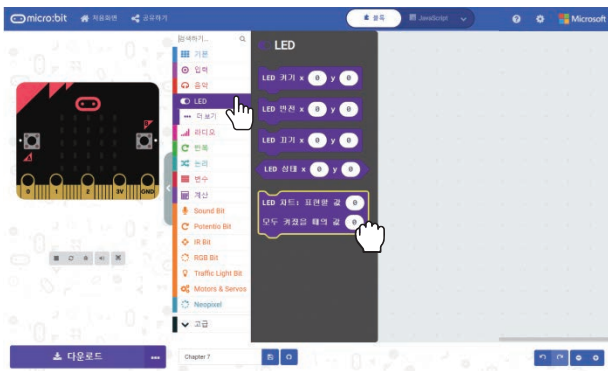
# CHAPTER 7

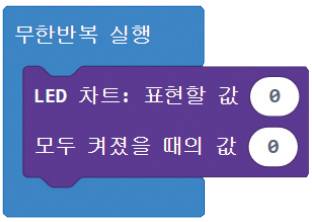
# 박수소리를 들어보자!

Sound Bit

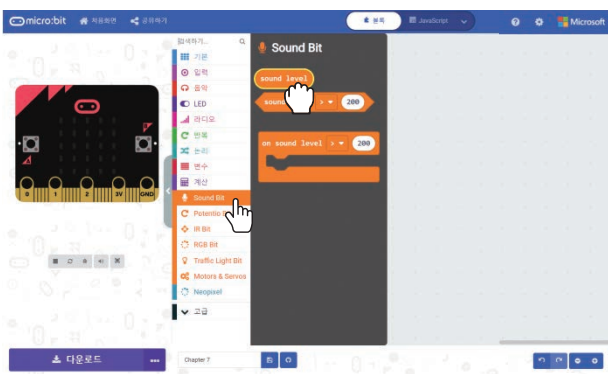
## 코딩을 시작해봅시다!

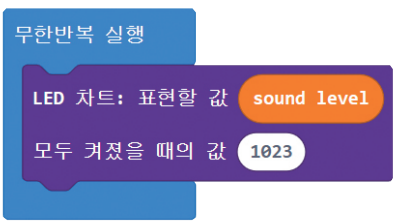
Step 1 MakeCode Editor에서 새 프로젝트를 생성하고 EDU:BIT 확장 프로그램을 추가합니다. *(40페이지 참고)*
[ LED ] 카테고리를 클릭하고 [ LED 차트: 표현할 값 _ 모두 켜졌을 때의 값 _ ] 블록을 선택합니다. 블록을 [ 무한반복 실행 ] 슬롯에 맞추어 넣습니다.



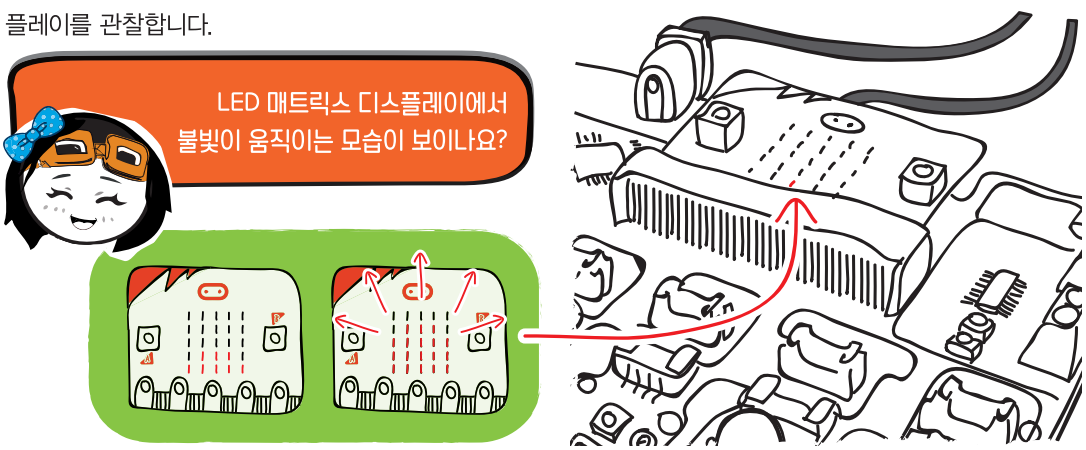




Step 2 [ Sound Bit ] 카테고리를 클릭하고 [ sound level ] 블록을 선택합니다.
[ LED 차트: 표현할 값 _ 모두 켜졌을 때의 값 _ ] 블록에 맞추어 넣고 값을 0에서 1023으로 변경합니다.



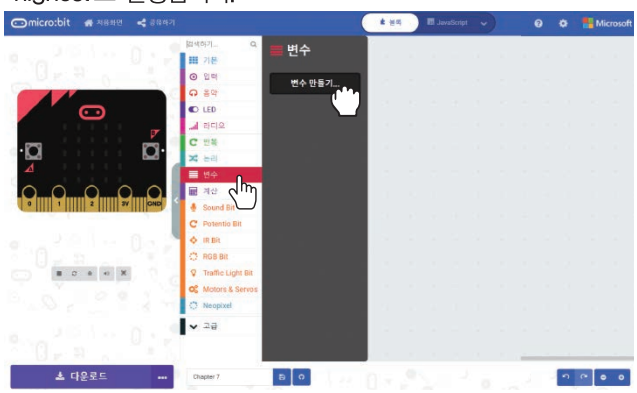



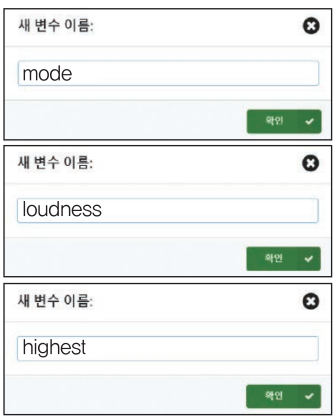

Step 3 EDU:BIT에 코드를 플래시합니다. 박수를 치거나 손가락으로 테이블을 두드리면서 LED 매트릭스 디스플레이를 관찰합니다.



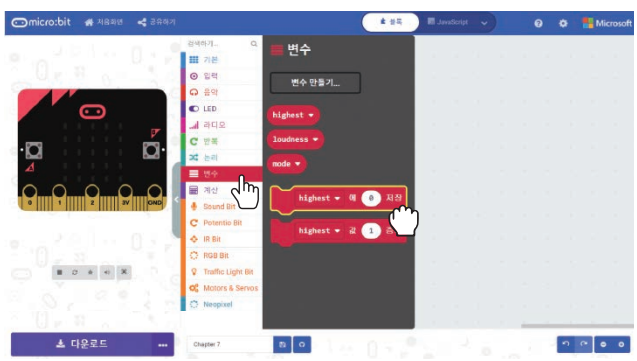
## 이제 EDU:BIT로 박수소리 크기를 측정해봅시다.

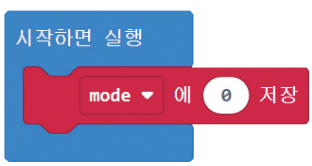
Step 4 새 프로젝트를 생성하고 EDU:BIT 확장 프로그램을 추가합니다. [ 변수 ] 카테고리에서 [ 변수 만들기 ]를 클릭합니다. 팝업창에 'mode'를 입력하고 확인을 누릅니다. 변수를 두 개 더 생성하고 이름은 'loudness'와 'highest'로 설정합니다.



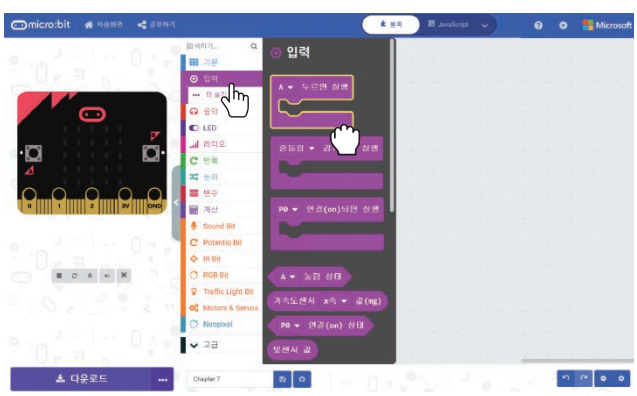




Step 5 [ 변수 ] 카테고리에서 [ _ 에 _ 저장 ] 블록을 선택합니다. [ 시작하면 실행 ] 슬롯에 블록을 맞추어 넣고 변수를 'mode'로 설정합니다.

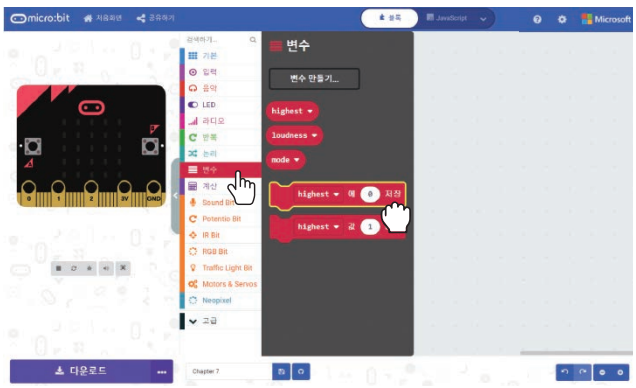



Step 6 [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다.

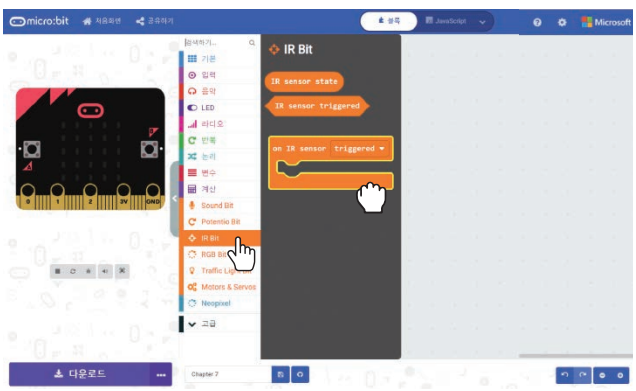


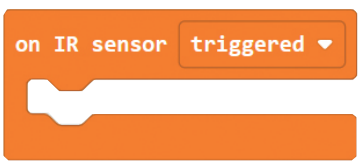

Step 7 [ 변수 ] 카테고리에서 [ _ 에 _ 저장 ] 블록을 두 개 추가하고, [ A 누르면 실행 ] 슬롯에 맞추어 넣습니다. 첫 번째 블록의 변수는 'mode', 값은 1로 변경하고 두 번째 블록의 변수는 'highest', 값은 0으로 설정합니다.



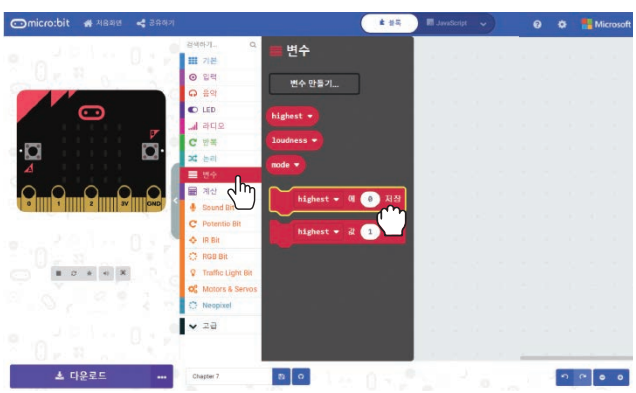


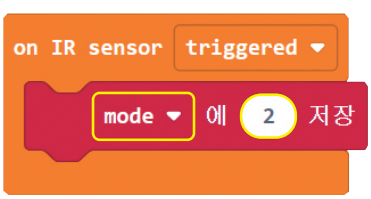
Step 8 [ IR Bit ] 카테고리를 클릭하고 [ on IR sensor triggered ] 블록을 선택합니다.



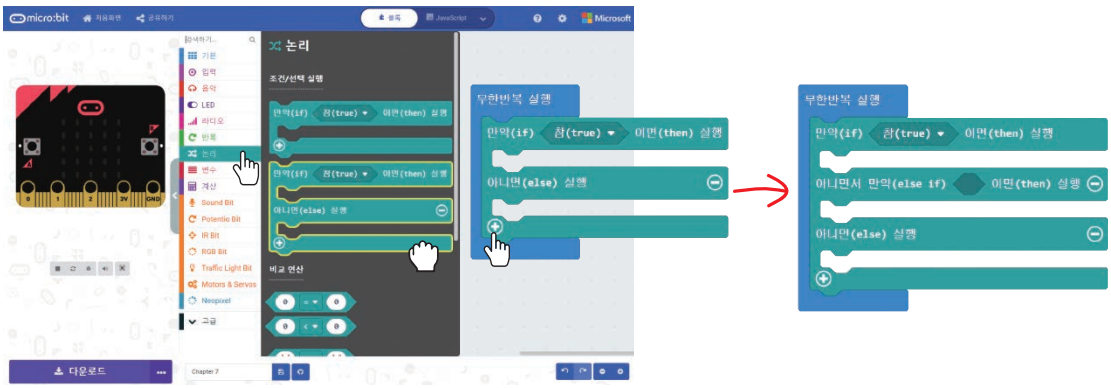




Step 9 [ 변수 ] 카테고리를 클릭하고 [ _ 에 _ 저장 ] 블록을 선택합니다.
[ on IR sensor triggered ] 슬롯에 블록을 맞추어 넣고 변수는 'mode', 값은 2로 변경합니다.

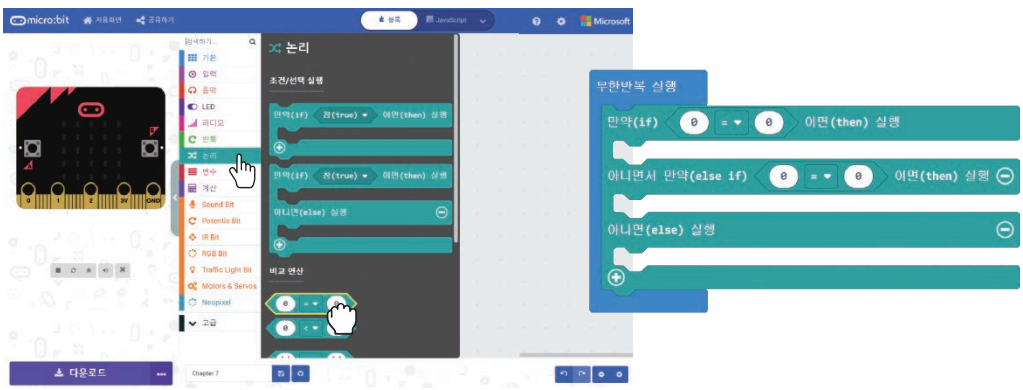


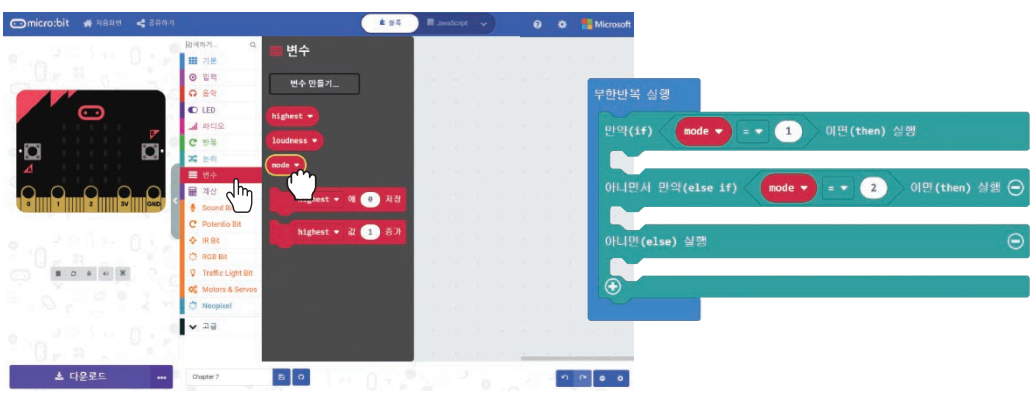
Step 10 [ 논리 ] 카테고리를 클릭하고 [ 만약(if) _ 이면(then) 실행 아니면(else) 실행 ] 블록을 선택합니다.
[ 무한반복 실행 ] 슬롯에 블록을 맞추어 넣습니다. 블록에 아니면서 만약(else if) 조건을 더 추가하기 위해 (+) 아이콘을 클릭합니다.




Step 11 [ 논리 ] 카테고리에서 [ _ = _ ] 비교 연산 블록을 선택합니다.
블록을 복사하고 [ if_then_else ] 블록의 조건 슬롯에 블록을 맞추어 넣습니다.

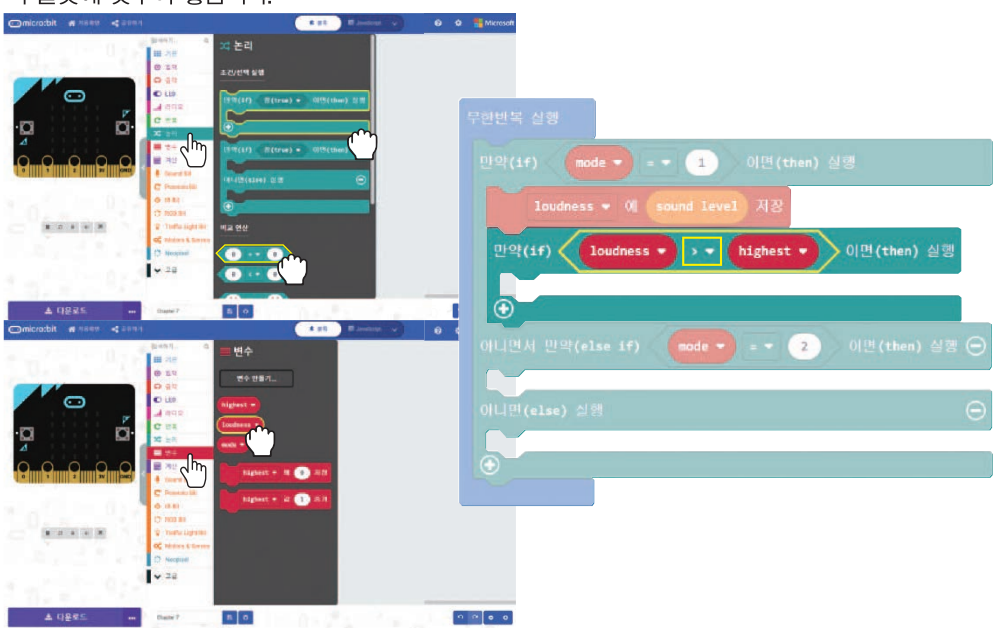



Step 12 [ 변수 ] 카테고리로 부터 [ mode ]를 선택하고 비교연산 블록의 왼쪽 슬롯에 맞추어 넣습니다.
다른 슬롯들은 각각 1과 2로 설정합니다.

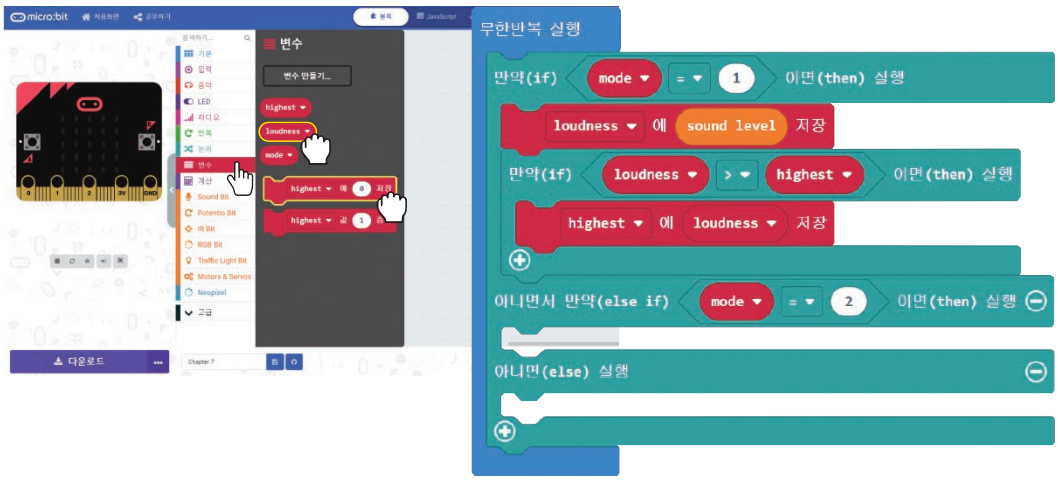


Step 13 [ 변수 ] 카테고리로 부터 [ _ 에 _ 저장 ] 블록을 선택하고 [ if_then_else ]의 첫번째 슬롯에 맞추어 넣습니다. 변수를 'loudness'로 설정하고 값 슬롯에 [ Sound Bit ] 카테고리의 [ sound level ] 블록을 맞추어 넣습니다.


Step 14 [ 논리 ] 카테고리를 클릭하고 [ 만약(if) _ 이면(then) 실행 ] 블록과 [ _ = _ ] 비교 연산 블록을 선택합니다. 기호를 = 에서 〉로 변경합니다. [ 변수 ] 카테고리로 부터 [ loudness ]와 [ highest ]를 비교 연산 블록의 슬롯에 맞추어 넣습니다.

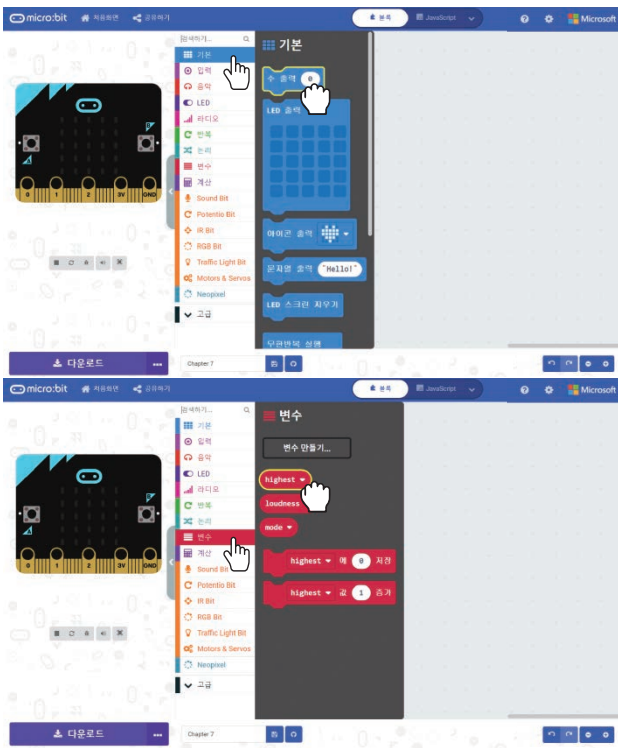


Step 15 [ 변수 ] 카테고리를 클릭하고 [ _ 에 _ 저장 ] 블록을 선택합니다. [ 만약(if) _ 이면(then) 실행 ] 슬롯에 블록을 맞추어 넣고 [ 변수 ] 카테고리의 [ loudness ]를 선택해 값 슬롯에 넣습니다.



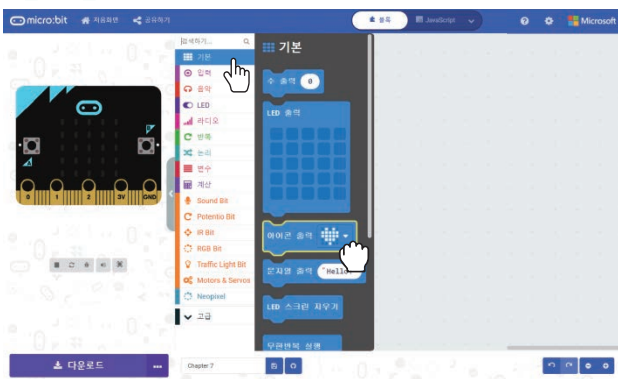

Step 16 [ LED ] 카테고리를 클릭하고 [ LED 차트: 표현할 값 _ 모두 켜졌을 때의 값 _ ] 블록을 선택합니다. [ 변수 ] 카테고리를 클릭하고 [ highest ] 블록을 선택합니다. [ LED 차트: 표현할 값 _ 모두 켜졌을 때의 값 _ ] 블록에 맞추어 넣고 값을 1023으로 변경합니다.

Step 17 [ 기본 ] 카테고리를 클릭하고 [ 수 출력 ] 블록을 선택합니다. [ if_then_else ] 블록의 두 번째 슬롯에 끼워 넣습니다. [ 변수 ] 카테고리에서 [ highest ] 블록을 선택하고 [ 수 출력 ] 블록의 값 슬롯에 맞추어 넣습니다.

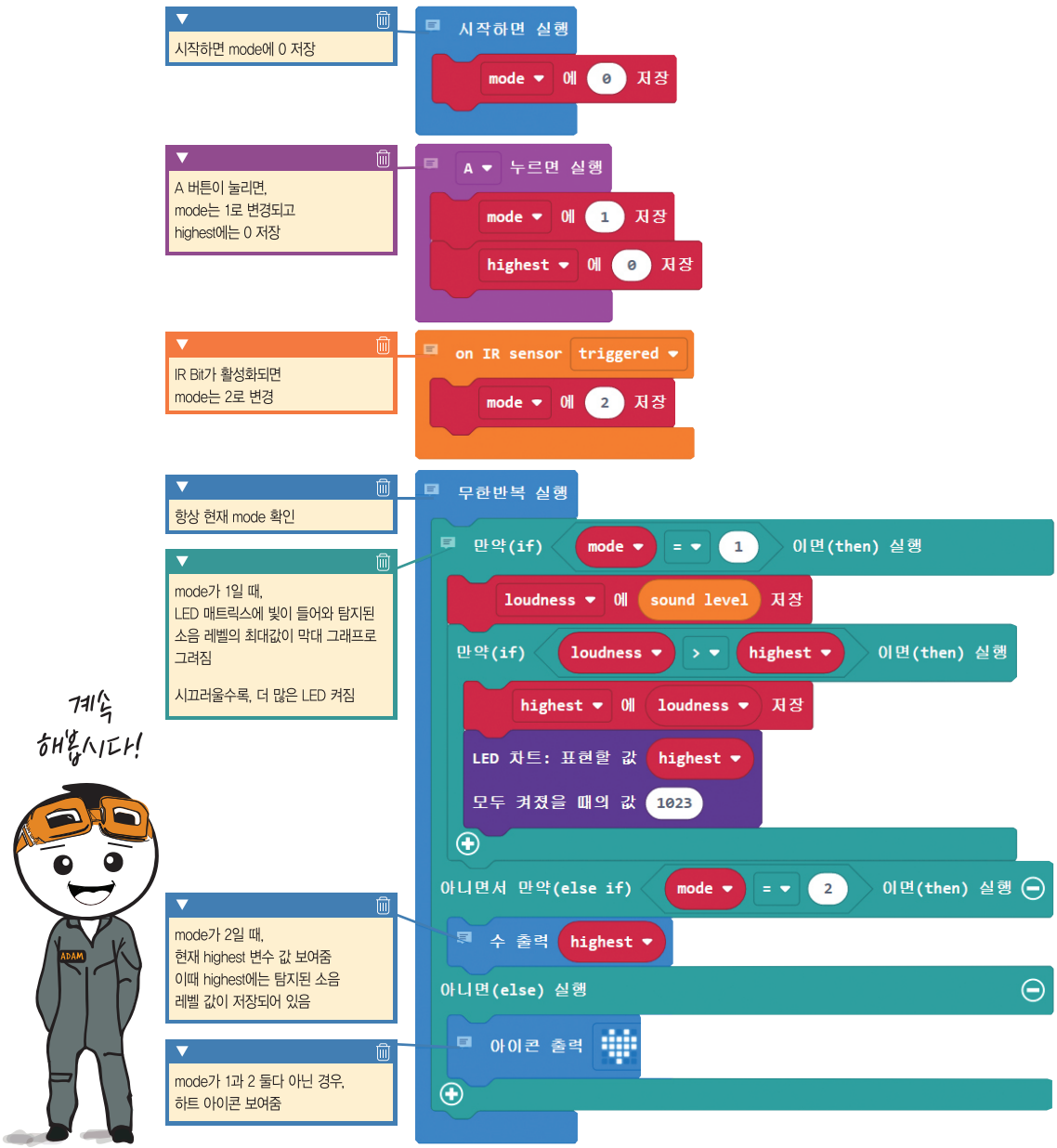




Step 18 [ 기본 ] 카테고리를 클릭하고 [ 아이콘 출력 ] 블록을 선택합니다.
[ if_then_else ] 블록의 마지막 슬롯에 맞추어 넣습니다.



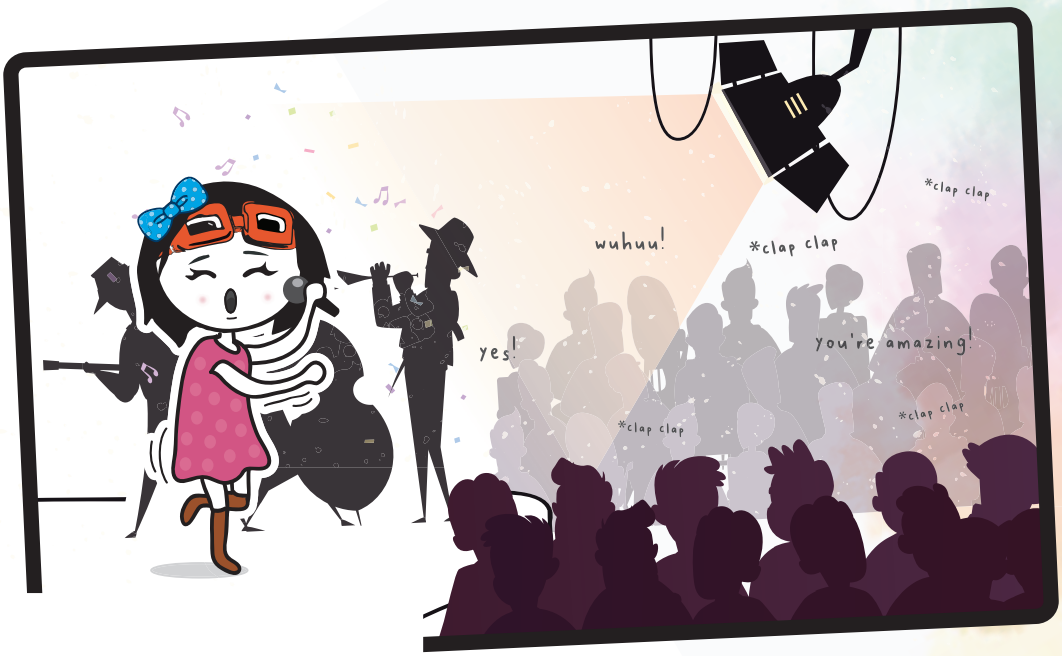

전체 코드입니다.




Step 19 EDU:BIT에 코드를 플래시합니다. 장기자랑 시간에 박수 소리 크기 측정기를 사용해봅시다.

# 게임하기

박수 소리를 들어보자!




## 게임 방법:

- 참가자들에게 노래, 춤, 개그 등 짧은 공연을 준비할 시간이 주어집니다.
  개별적으로, 짝을 지어, 또는 팀으로 참여할 수 있습니다.

- 모두 준비가 되면, 장기자랑을 시작합니다.
  각 공연 후에, 관객은 박수로 응답합니다. 공연이 재미있을수록 더 크게 박수를 칩니다.

- 박수 소리가 잦아들면, IR Bit를 작동시켜 점수를 출력합니다. 가장 큰 소리 레벨이 점수로 기록되어 있습니다.

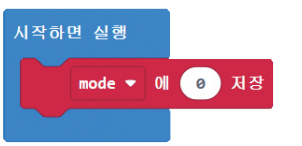
- 다음 공연 전에 점수를 초기화하기 위해 A 버튼을 눌러야 함을 기억합니다.

- 가장 큰 박수 소리를 받은 참가자가 게임의 승자가 됩니다. 자 이제 게임을 시작해봅시다!

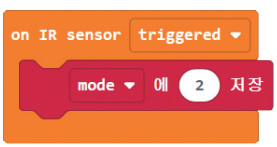
# 코딩 정복하기

한 프로그램에서 여러 작업을 수행할 때 이벤트 트리거를 사용하여 한 작업에서 다른 작업으로 전환할 수 있습니다. 프로그램이 원활하게 실행되도록 하기 위해, 무한반복 루프를 사용하여 지속적으로 현재 모드를 확인한 다음 해당하는 코드 블록을 실행합니다.

모드를 0으로 설정합니다.
(대기 모드)








런타임(실행 시간) 동안에
한 모드에서 다른 모드로 전환하기 위한
이벤트 트리거 블록입니다.

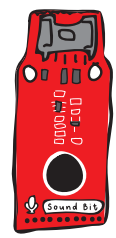
항상 현재 모드가 무엇인지 확인하고,
특정 모드에 해당하는 작업을 수행합니다.


만약 더 포함하고 싶은 모드나 작업이 있다면,
코드의 **[if_then_else]** 블록에서 (+) 버튼을 눌러 조건을 더 만들고
**[ 흔들림 감지하면 실행 ]**이나 **[ on sound level >_ ]**과 같은
이벤트 트리거 블록을 추가하면 됩니다.

# 재미있는 사실!

소리는 드럼을 칠 때처럼 물체의 진동에 의해 생성됩니다. 그 진동은 공기 분자를 진동시키고 음파를 발생시킵니다. **소리 센서**는 소리 세기를 감지하여 전기 신호로 변환시키는 모듈입니다.


어떤 면에서는, 소리 센서가 우리 귀와 유사하게 작동합니다.
우리의 귀는 공기 중의 진동을 전기화학 신호로 변환하고,
뇌는 우리가 아는 소리로 바꾸어줍니다.





## 소리 센서가 적용된 것

- 소음 감지 도난 경보기
- 소리로 작동하는 조명
- 소리 감지 베이비 모니터

우주공간에서 소리 센서가 소음을 감지할 수 있을까요?
이유는 무엇인가요?

# 응용 과제

EDU:BIT가 학급의 소음을 모니터링하도록 프로그래밍 해봅시다.
Traffic Light Bit의 LED로 소음 레벨을 표시합니다.

| 소음레벨 | 소음 레벨 범위 | 켜지는 Traffic Light Bit |
|---|---|---|
| 매우 시끄러움<br>목소리를 낮춰주세요. | (  ) to 1023 | 빨간색 LED |
| 약간 시끄러움<br>목소리 크기에 신경쓰세요. | (  ) to (  ) | 노란색 LED |
| 수용 가능한 소음 레벨<br>훌륭해요! | 0 to (  ) | 초록색 LED |

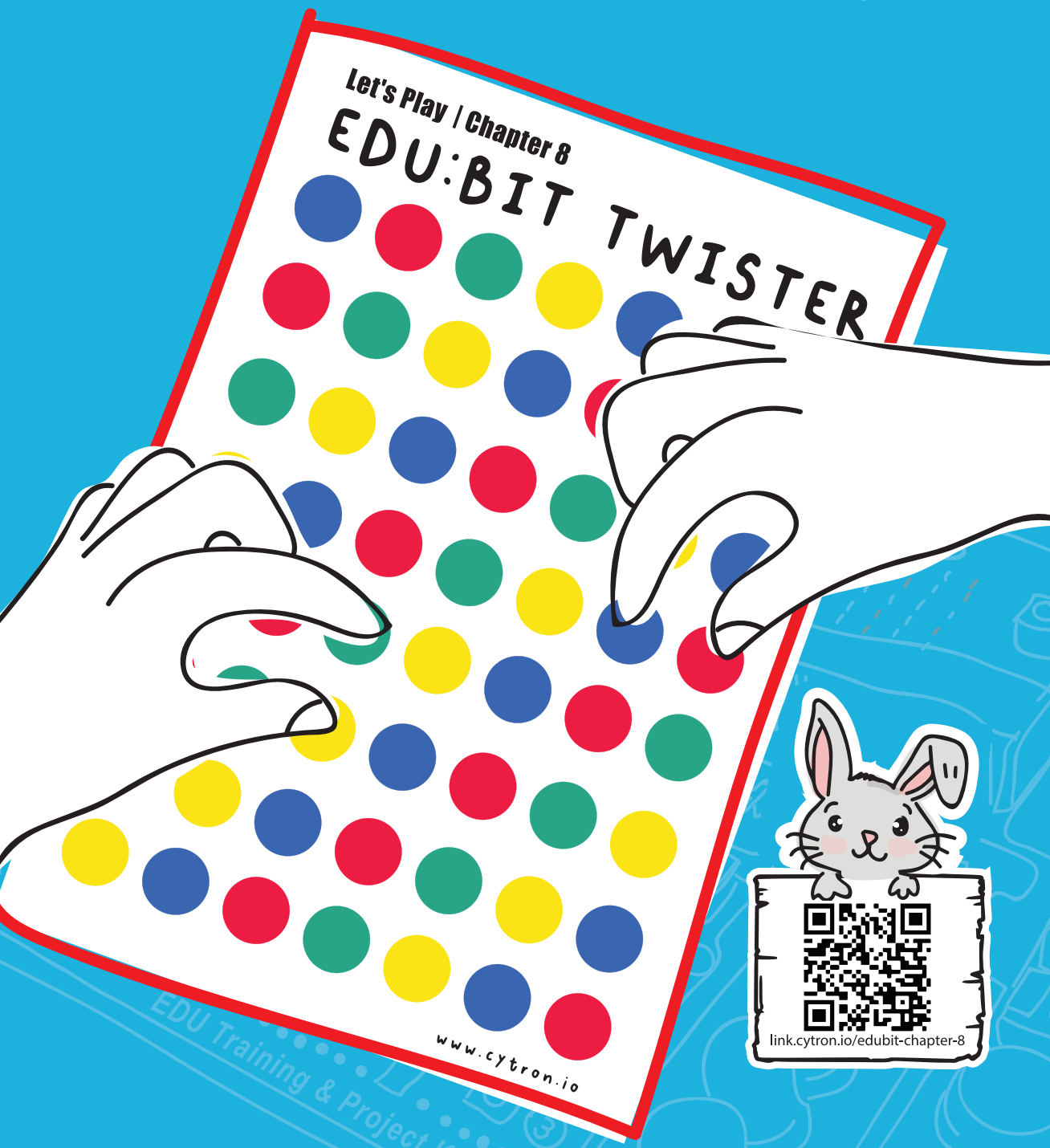
힌트입니다!
#1 각 소음 레벨에 대한 한계값을 미리 결정해야 합니다.
#2 더 안정적으로 모니터링하기 위해,
일정한 간격으로 소음 레벨의 평균 값을 구합니다.

너무 쉬운가요? 레벨업 과제에 도전해봅시다!
한계값이 포텐셔미터 값(potentiometer value)과
관련되도록 코드를 수정해봅시다.

# CHAPTER 8

# 빙글빙글 돌려보자!

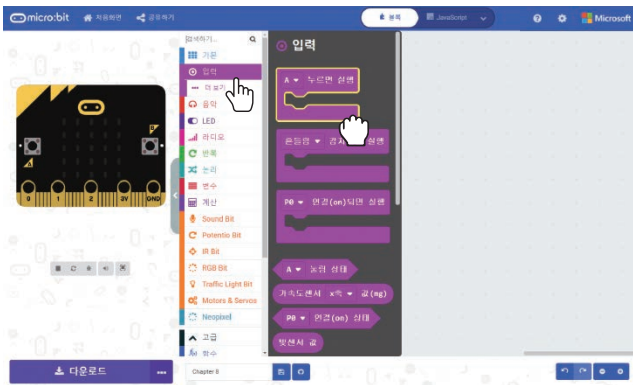
DC Motor



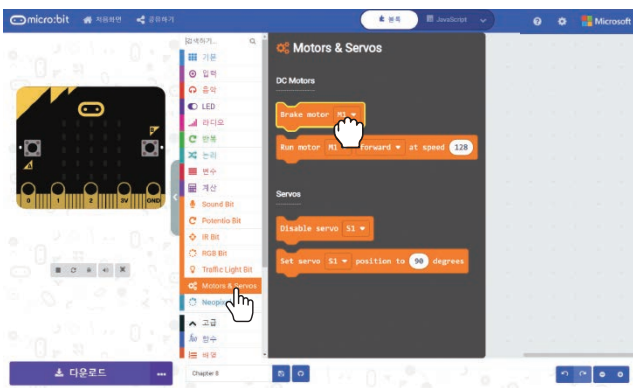

link.cytron.io/edubit-chapter-8

## 코딩을 시작해봅시다!

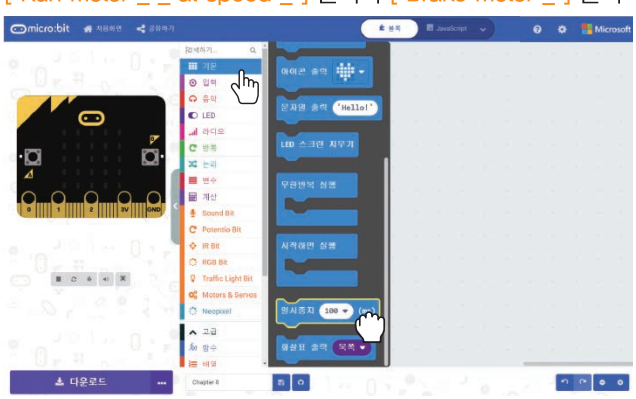
Step 1 MakeCode Editor에서 새 프로젝트를 생성하고 EDU:BIT 확장 프로그램을 추가합니다. (40페이지 참고)
[ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다.



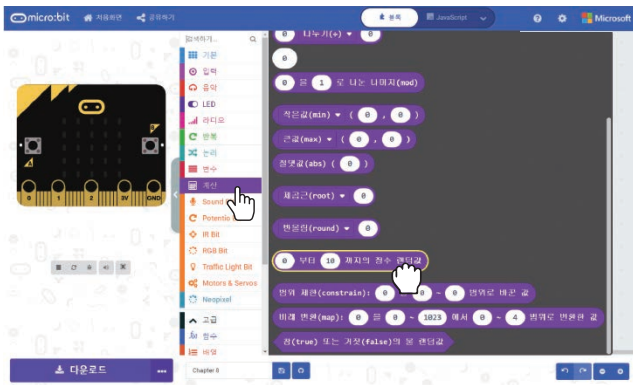

Step 2 [ Motors & Servos ] 카테고리를 클릭하고 [ Run motor _ _ at speed _ ] 블록과 [ Brake motor _ ] 블록을 추가합니다. speed 값을 80으로 변경합니다.



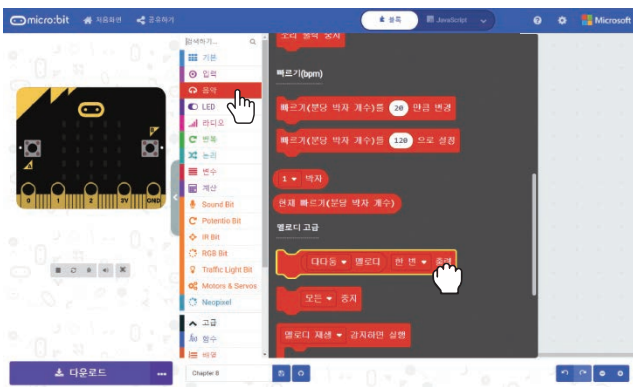

Step 3 [ 기본 ] 카테고리를 클릭하고 [ 일시중지 _ (ms) ] 블록을 추가합니다.
[ Run motor _ _ at speed _ ] 블록과 [ Brake motor _ ] 블록 사이에 끼워 넣습니다.




Step 4 [ 계산 ] 카테고리를 클릭하고 [ _ 부터 _ 까지의 정수 랜덤값 ] 블록을 선택합니다.
[ 일시중지 _ (ms) ] 블록에 맞추어 넣고 값을 각각 200과 1000으로 변경합니다.



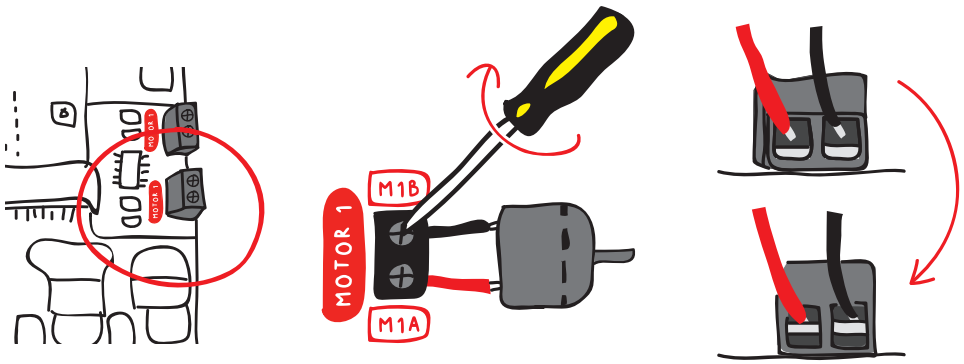

Step 5 [ 음악 ] 카테고리를 클릭하고 [ _ 멜로디 _ 출력 ] 블록을 선택합니다.
'바 딩' 멜로디나 원하는 멜로디로 변경합니다.




Step 6 완성된 코드를 EDU:BIT에 플래시합니다.

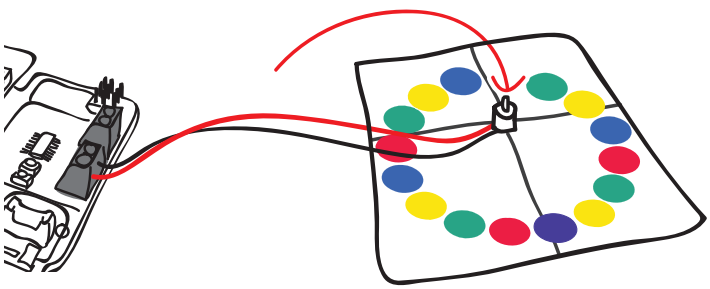

랜덤 스피너를 필요로 하는 어떤 게임에서든지 이 코드를 사용할 수 있습니다.
기본적으로 모터는 트리거 될 때 돌아가기 시작하고 랜덤 시간 후에 멈춥니다.

Step 7 DC모터를 MOTOR 1 단자(터미널)에 연결합니다. 우선 선의 노출된 부분을 집어넣습니다. 올바른 위치에 고정하기 위해 제공된 드라이버를 사용해서 나사를 조입니다.

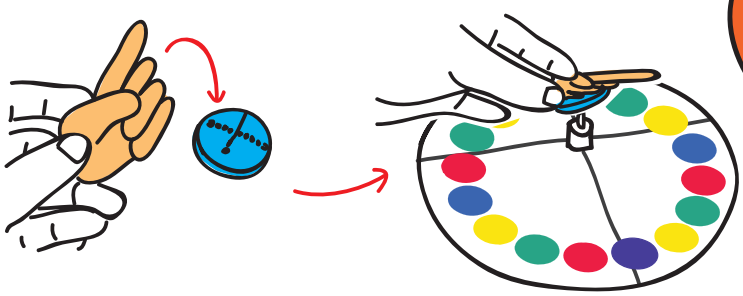



테스트를 위해 노란색 버튼 (A 버튼)을 누릅니다. 만약 모터가 돌아가지 않으면, 단자에 선이 확실하게 연결되어 있고 EDU:BIT의 전원이 켜져 있는지 확인합니다.

Step 8 양면테이프나 글루건 같은 접착제를 사용해서 표시된 대로 DC모터를 게임 맵의 중앙에 부착합니다.




Step 9 포인터를 분리하고 플라스틱 판에 접착제로 부착합니다. 그리고 모터 축에 판을 고정시킵니다.



박스에 트위스터 휠, 포인터, 게임 맵이 제공되어 있습니다.

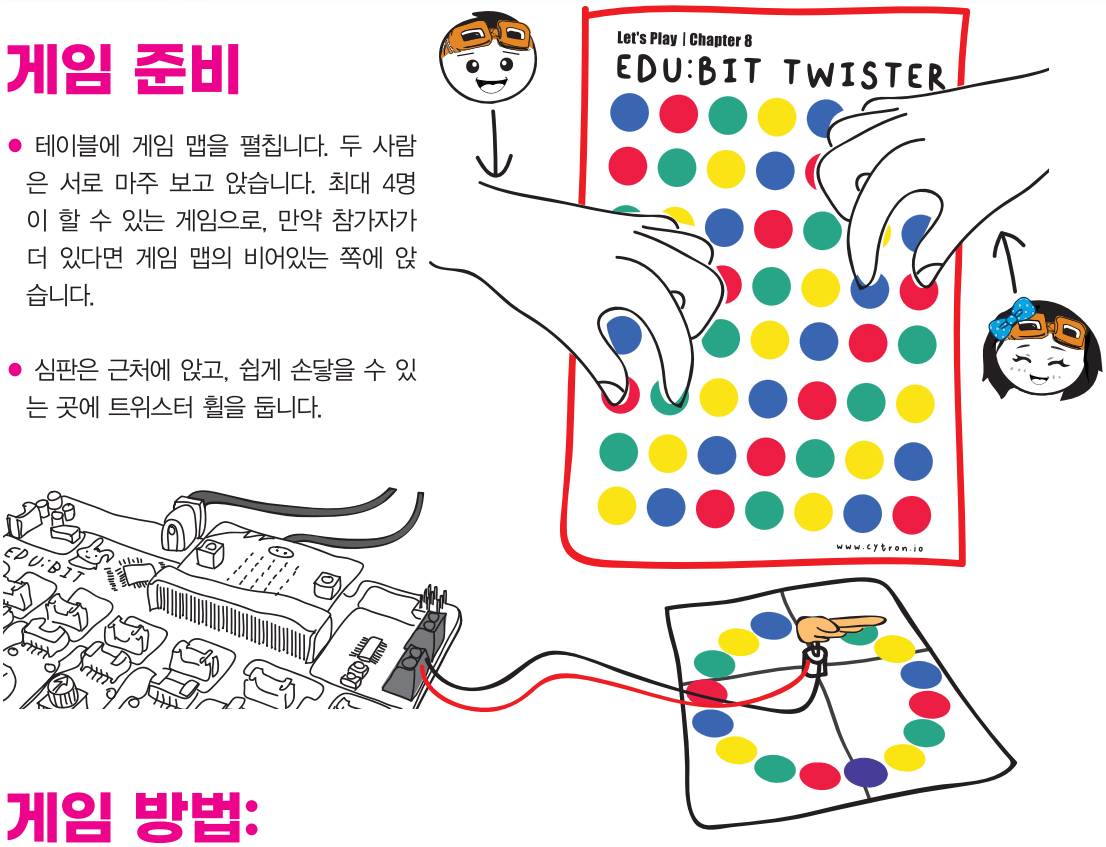
# 게임하기

빙글빙글 돌려보자!

## 게임 준비

- 테이블에 게임 맵을 펼칩니다. 두 사람은 서로 마주 보고 앉습니다. 최대 4명이 할 수 있는 게임으로, 만약 참가자가 더 있다면 게임 맵의 비어있는 쪽에 앉습니다.

- 심판은 근처에 앉고, 쉽게 손닿을 수 있는 곳에 트위스터 휠을 둡니다.




## 게임 방법:

이 게임에서, 참가자들은 심판의 지시에 따라 돌아가면서 손가락을 게임 맵의 색칠된 원 위에 놓습니다.

심판의 역할은 노란색 버튼(버튼 A)을 눌러 포인터를 돌리고, 포인터가 가리키는 손가락과 색을 외칩니다.
예를 들어 '검지 손가락(index finger), 빨간색'이라고 외치면 됩니다.

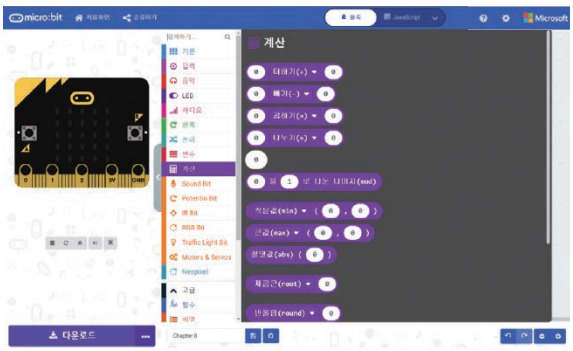
차례가 돌아오면, 심판의 지시를 듣고 올바른 색의 원에 해당하는 손가락을 갖다 댑니다.
만약 이미 그 손가락이 해당하는 색의 원에 있다면, 같은 색의 다른 원으로 옮깁니다.

만약 성공적으로 옮기지 못했으면 탈락입니다.

마지막까지 남은 사람이 게임의 승자가 됩니다.

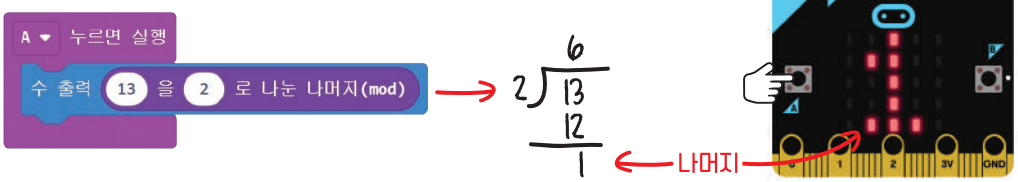
# 더 많은 블록 탐구하기

변수들을 산술연산하려면 [ 계산 ] 카테고리에 있는 블록들을 사용하면 됩니다.
#1 다음 블록들을 사용해서 더하기, 빼기, 곱하기, 나누기를 할 수 있습니다.

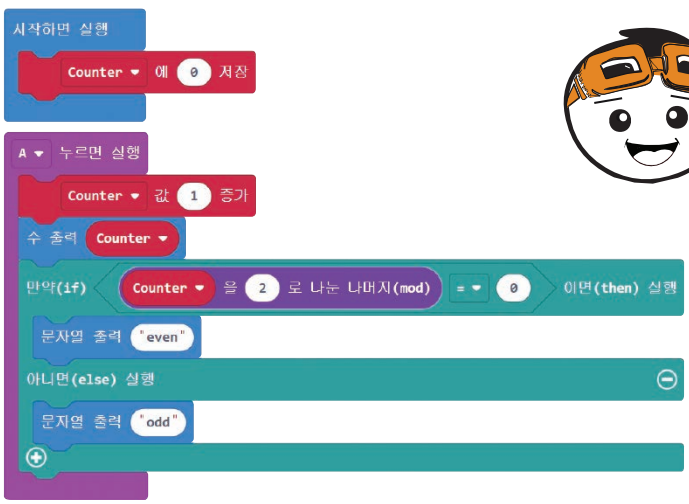




#2 [ _ 을 _ 로 나눈 나머지(mod) ] 블록을 사용해서 어떤 수가 다른 수로 균등하게 나누어지지 않았을 때 나머지를 알아낼 수 있습니다.




#3 [ _ 을 _ 로 나눈 나머지(mod) ] 블록을 사용해서 숫자가 홀수(odd)인지 짝수(even)인지 알아낼 수 있습니다. 간단하게 2로 나누어 봅시다. 만약 나머지가 '1'이면 '홀수'이고, '0'이면 '짝수'입니다. 한번 해봅시다!

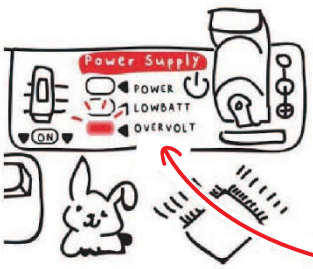





처음 A 버튼을 눌렀을 때 EDU:BIT에 odd가 출력되었는데, A 버튼을 다시 눌렀더니 even이 나온다는 것을 알아차렸나요?
A 버튼을 99번 누르면 어떤 결과가 출력될까요?

# 재미있는 사실!

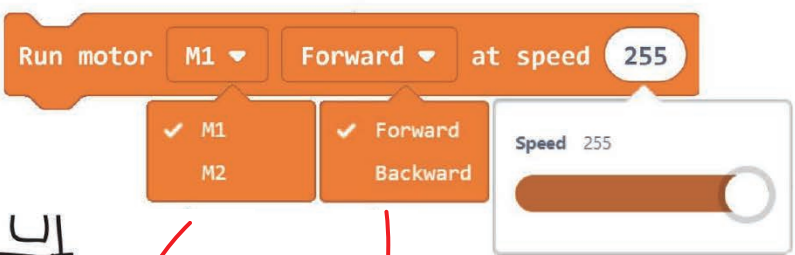
흔히 **DC모터**라고 알려진 직류 모터는
전기 에너지를 역학적 에너지로 변환해주는 장치입니다.
DC모터 회전을 만들기 위해 입력 전압을 주어야 합니다. 입력 전압을 조정함으로써 회전 속도를 조절할 수 있습니다. 입력 전압이 높아질수록 모터 회전은 더 빨라집니다. EDU:BIT 키트에 있는 DC모터의 권장 전압은 3.6V ~ 6V 입니다.

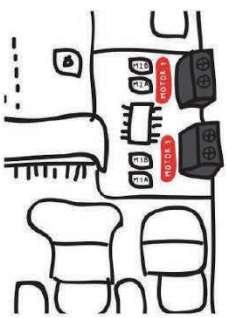



과전압 표시

**경고!**
모터에 권장 전압 이상의 전압을 가하면
장기적으로 모터의 수명이 단축됩니다.


다음과 같은 프로그래밍 블록을 이용해서 DC모터의 회전 방향과 속도를 쉽게 조절할 수 있습니다.



EDU:BIT에는
두 개의 DC모터 단자가 있습니다.
올바른 단자를 선택합니다.

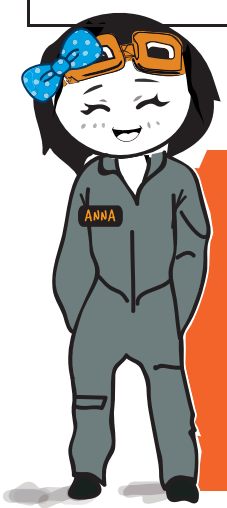
회전 방향

0 ~ 255 사이의 상대값입니다.
값이 커질수록 모터 회전이
빨라집니다.

EDU:BIT에 모터 테스트 회로가 내장되어 있다는 사실을 알고 있나요?
연결이 잘 되었는지, 모터가 잘 작동하는지 확인하기 위해서
M1A, M1B, M2A, M2B라고 적혀있는 흰색 버튼을 누르면 됩니다.

# 응용 과제

| EDU:BIT가 Potentio Bit로 속도를 조절하고 소리로 껐다 켤 수 있는 선풍기 기능을 하도록 프로그래밍 해봅시다. | |
|---|---|
| 시작하면 실행 | 원하는 아이콘을 보여줍니다.<br>Mode 변수를 0으로 설정합니다. |
| On sound level 〉 _<br>(한계값) | Mode 변수를 1 증가시킵니다. |
| 무한반복 실행 | 포텐셔미터 값(potentiometer value)을 최소 0 최대 1023에서 최소 0 최대 255로 비례 변환한 값을 Speed 변수에 저장합니다.<br><br>항상 Mode를 확인합니다.<br>● 만약(if) 모드가 짝수라면, M1 모터를 정지(brake)합니다.<br>● 아니면(else) 모드가 홀수이므로, M1 모터를 'Speed' 변수 값에 따라 작동시킵니다. 여기서 Speed 변수는 포텐셔미터 값(potentiometer value)을 비례 변환한 값 입니다. |




힌트입니다!
#1 모터를 켜고 끄기 위해 소리 레벨 트리거로 한계값을 결정해야 합니다.
#2 두 개의 변수를 생성합니다. (변수: Mode, Speed)
#3 선풍기 날개를 모터 축에 부착하고 프로그램을 실행합니다.
만약 모터가 돌아갈 때 바람이 부는 것이 느껴지지 않는다면,
코드에서 회전 방향을 변경해야 합니다.

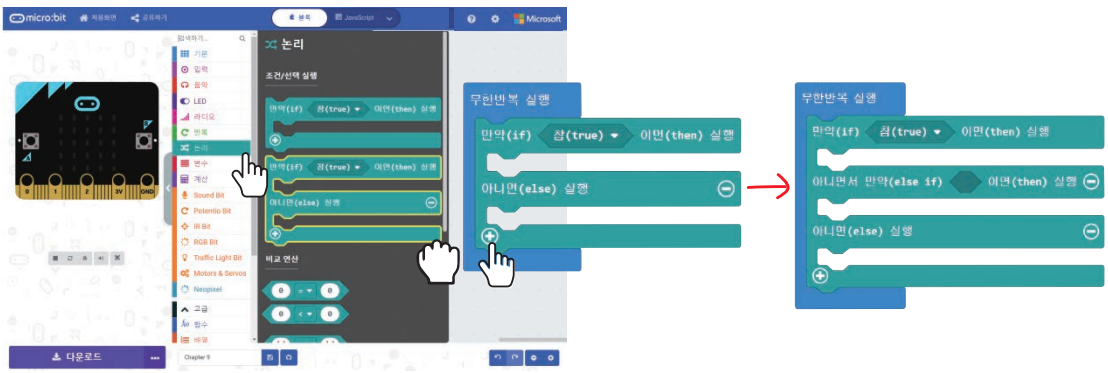
# CHAPTER 9

# 승부차기... 골!!!

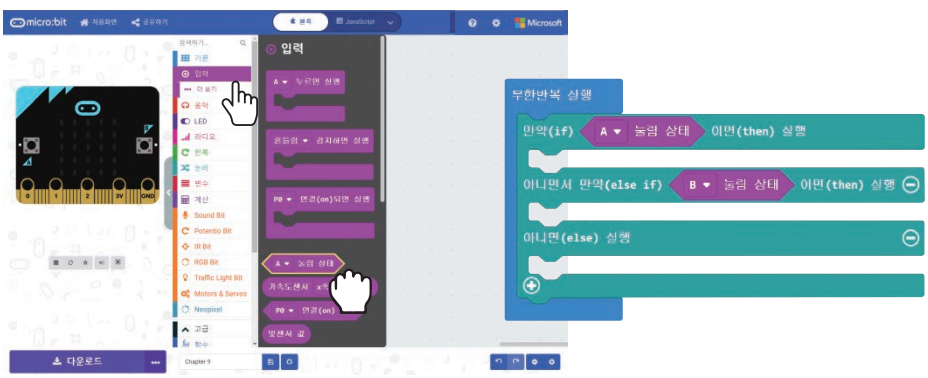
Servo Motor

## 코딩을 시작해봅시다!

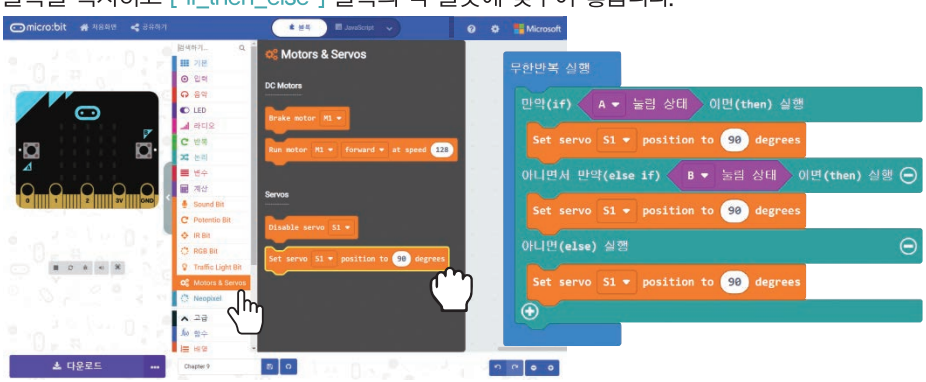
Step 1 MakeCode Edtior에서 새 프로젝트를 생성하고 EDU:BIT 확장 프로그램을 설치합니다. *(40페이지 참고)*
[ 논리 ] 카테고리를 클릭하고 [ 만약(if) _ 이면(then) 실행 아니면(else) 실행 ] 블록을 선택합니다.
[ 무한반복 실행 ] 블록에 맞추어 넣습니다. 아니면서 만약(else if) 조건을 추가하기 위해 (+) 아이콘을 클릭합니다.




Step 2 [ 입력 ] 카테고리를 클릭하고 [ _ 눌림 상태 ] 블록을 선택합니다. 블록을 복사하고 [ if_then_else ] 블록의 조건 슬롯에 맞추어 넣습니다. 두 번째 블록의 A를 'B'로 변경합니다.

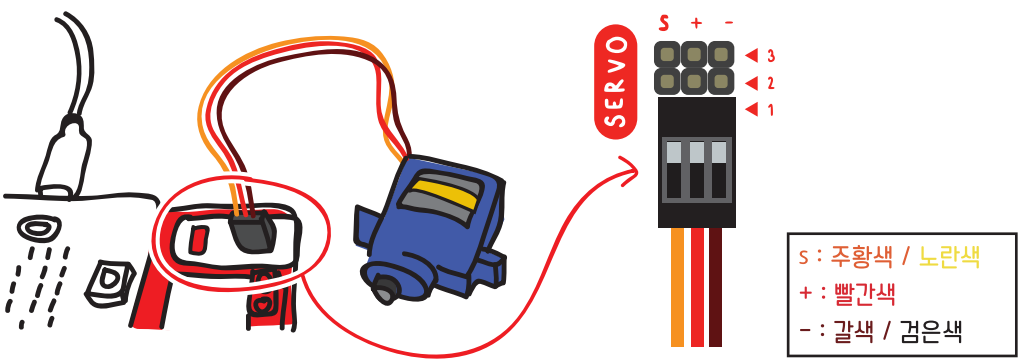



Step 3 [ Motors & Servos ] 카테고리를 클릭하고 [ Set servo _ position to _ degrees ] 블록을 선택합니다.
블록을 복사하고 [ if_then_else ] 블록의 각 슬롯에 맞추어 넣습니다.

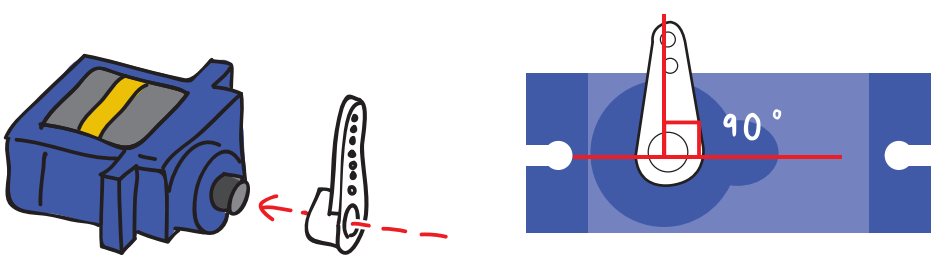



Step 4 첫 번째와 두 번째 블록의 값을 각각 30과 150으로 변경합니다. EDU:BIT에 코드를 플래시합니다.

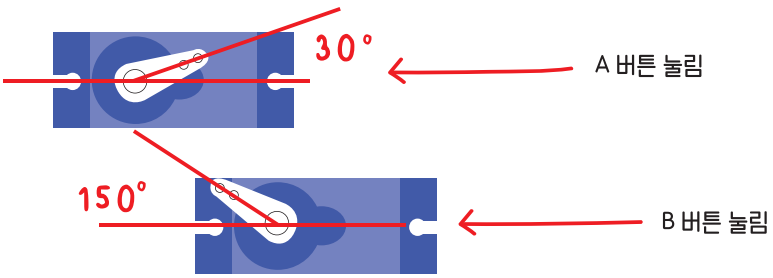

Step 5 아래 보이는 것 처럼 서보모터(servo motor) 케이블을 EDU:BIT의 Servo Port 1에 연결합니다.



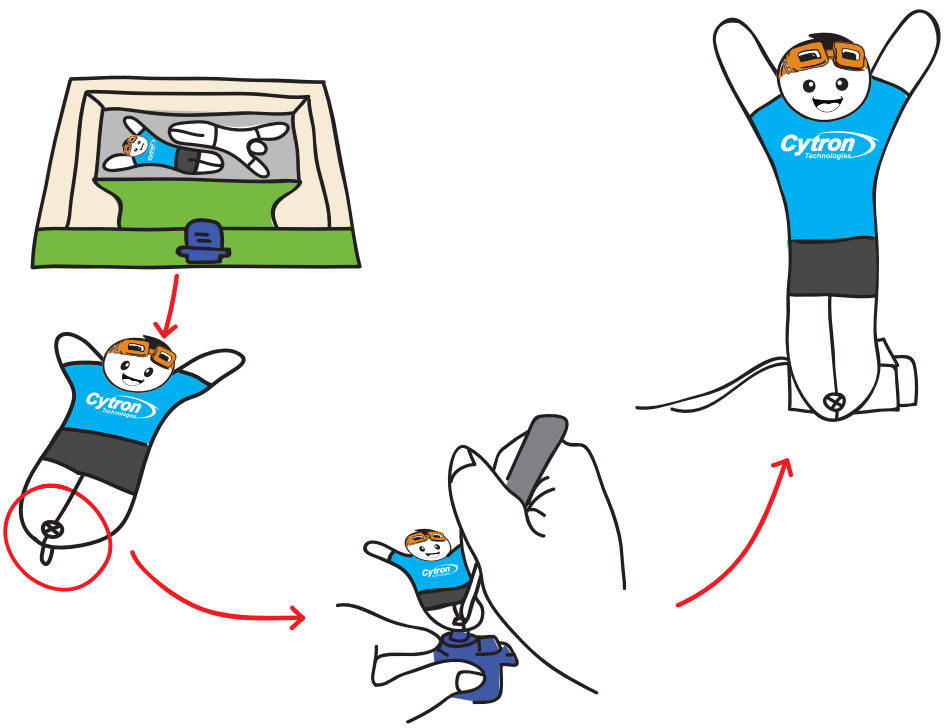

Step 6 EDU:BIT의 전원을 켜고, 아래에 보이는 것처럼 서보모터의 축에 서보 암 혼을 90도 위치에 끼웁니다.



Step 7 확인을 위해서 A 버튼을 누른 후 B 버튼을 누릅니다.

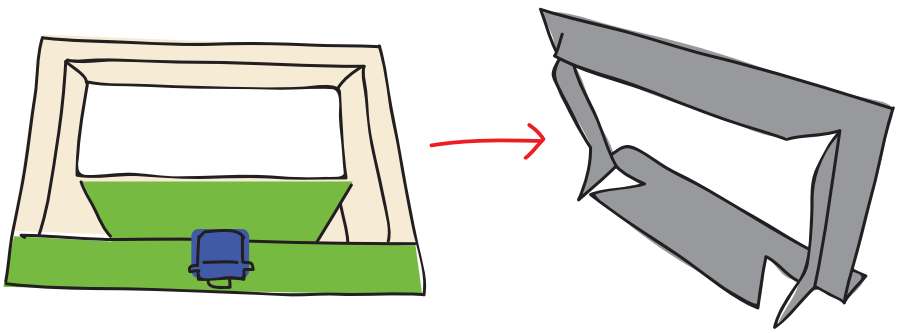



Step 8 박스에 제공된 재료에서 골키퍼를 뜯어냅니다. 제공된 드라이버와 나사를 이용해서 골키퍼를 서보 암 혼에 고정합니다. 양면테이프나 글루건 같은 접착제를 사용해서 서보 암 혼에 골키퍼를 단단하게 고정시킵니다.

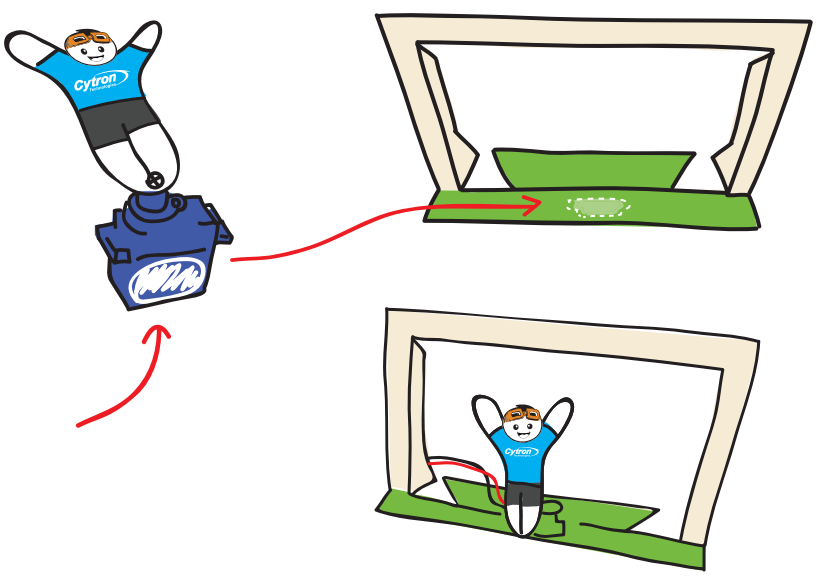



원한다면 제공된 다른 피규어를 사용해서 골키퍼의 운동복을 디자인 할 수 있습니다.

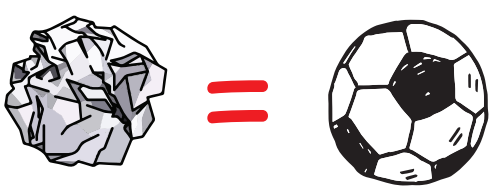

Step 9 골대를 아래에 보이는 것처럼 만듭니다.



Step 10 양면테이프나 글루건 같은 접착제를 사용해서, 서보모터를 표시된 자리에 단단하게 부착합니다.

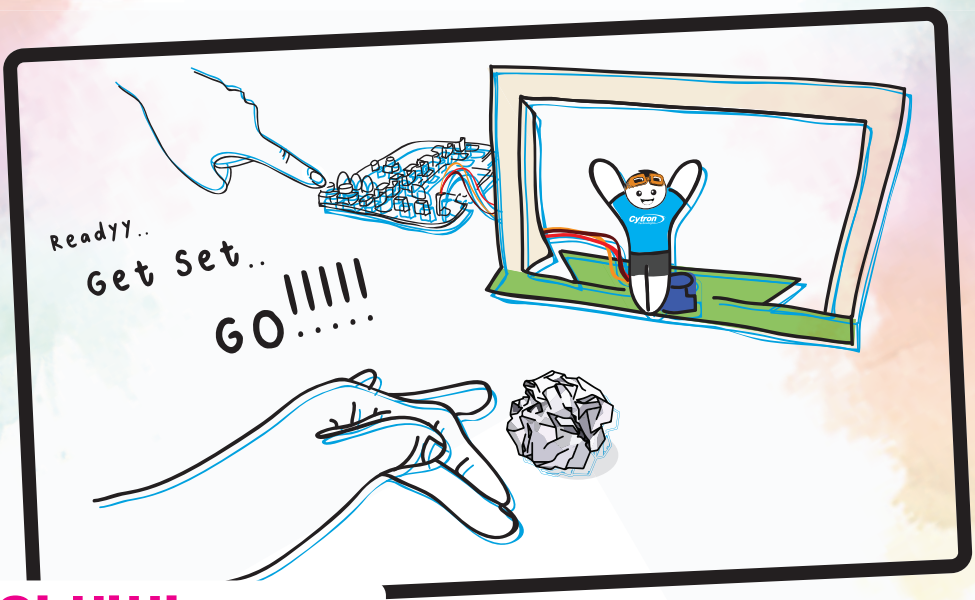



Step 11 작은 종이 조각을 구겨서 '축구공'을 대체합니다. 승부차기 게임을 위한 모든 준비를 마쳤으니 게임을 시작해볼까요?



# 게임하기

승부차기... 골!!!




## 게임 방법:

골대를 설치하고 공을 찰 위치를 표시합니다.
(대략 골대로부터 1미터로 설정하고 어린 친구들을 위해 거리를 조정합니다.)

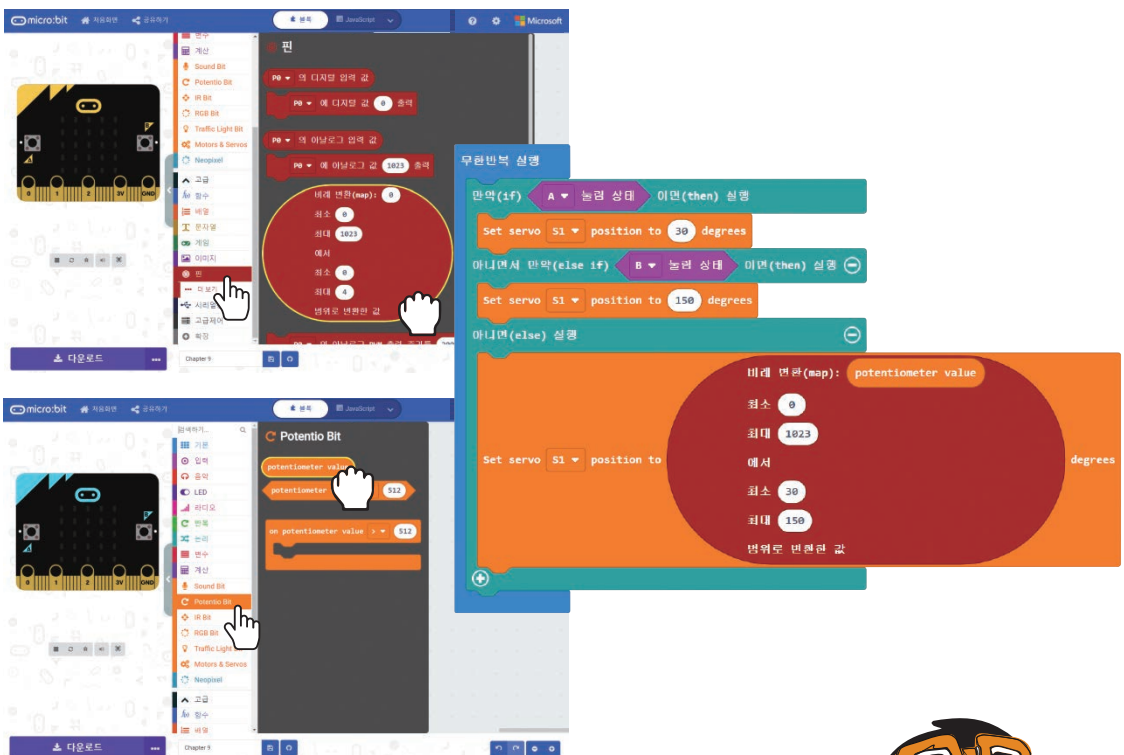
- 돌아가면서 킥커와 골키퍼가 됩니다.
- 킥커는 골대를 향해서 공을 튕깁니다.
- 골키퍼는 노란색 버튼 (A 버튼)을 눌러 왼쪽으로 움직이거나 파란색 버튼(B 버튼)을 눌러 오른쪽으로 움직여 공을 막습니다.
- 한 라운드에서 한 사람당 5번의 기회를 갖습니다. 골을 가장 많이 넣은 사람이 게임의 승자가 됩니다.

알고 있나요?
승부차기는 축구 경기에서 정규 게임의 마지막에 점수가 동점이고 연장전에도 무승부로 남았을 때, 승부를 가리기 위해서 행해집니다.
승부차기에서 각 팀은 공을 다섯 번 찰 수 있습니다. 선수는 패널티 마크에서 상대 팀의 골키퍼만 지키고 있는 골대로 공을 찹니다.
더 많은 골을 넣은 팀이 승리합니다.

앞의 코드에서, 푸쉬 버튼을 누름으로써 골키퍼가 왼쪽이나 오른쪽으로 움직이도록 만들었습니다. Potentio Bit를 사용해서 골키퍼의 위치를 조정할 수 있도록 코드를 수정할 수 있습니다.

Step 12 [ 고급 ] 카테고리를 클릭하고 [ 핀 ] 카테고리를 선택합니다.
[ 비례 변환(map): _ 최소 _ 최대 _ 에서 최소 _ 최대 _ 범위로 변환한 값 ] 블록을 코드에 추가합니다.

Step 13 [ Potentio Bit ] 카테고리를 클릭하고 [ potentiometer value ] 블록을 선택합니다.
[ 비례 변환(map): _ 최소 _ 최대 _ 에서 최소 _ 최대 _ 범위로 변환한 값 ] 블록에 맞추어 넣고, 마지막 두 값을 각각 30과 150으로 변경합니다.



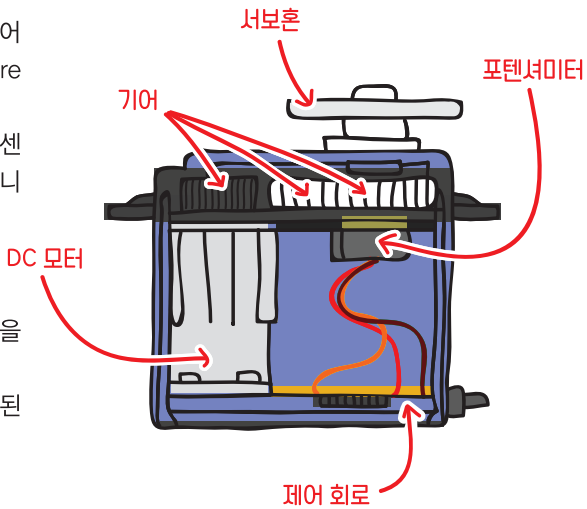

Step 14 코드를 EDU:BIT에 플래시합니다.
이제 Potentio Bit를 사용해서 골키퍼를 조정할 수 있습니다.


혼자 승부차기 연습을 하고 싶다면, 코드를 수정해서 골키퍼가 왼쪽과 오른쪽으로 계속 움직이는 '연습 모드'를 만들어 봅시다.

# 재미있는 사실!

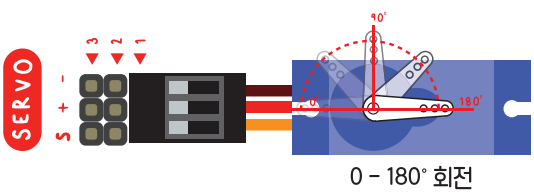
EDU:BIT 키트에 있는 서보모터(servo motor)는 RC(radio control) 서보로 알려져 있습니다. RC카와 소형 로봇의 움직임을 제어하는 데 널리 사용되곤 합니다.

서보모터는 전원(+), 그라운드(–), 제어 또는 신호(s)를 위해 삼선식(three-wire system)을 사용합니다.
보통 DC모터, 기어, 포텐셔미터 (위치 센서) 그리고 제어 회로로 구성되어 있습니다.



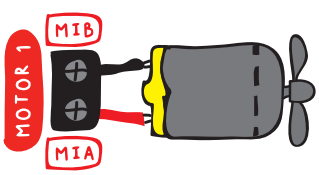

내장된 컨트롤러는 펄스 형태의 명령을 회전할 각도로 변환합니다.
서보모터가 지속적으로 회전하며 수신된 펄스에 해당하는 위치를 유지합니다.

360도를 연속적으로 회전하는 DC모터와 달리, 서보모터는 0도에서 180도 사이의 원하는 각도로 회전을 제어할 수 있습니다.



0 – 180° 회전

## 서보모터 VS DC모터



360° 회전

# 응용 과제

EDU:BIT가 메트로놈 기능을 하도록 프로그래밍 해봅시다.
전원을 켰을 때, 서보모터 혼에 부착된 포인터가 정해진 박자대로 왼쪽과 오른쪽으로 반복적으로 움직입니다.
박자는 Potentio Bit에 의해 조정되도록 설정합니다.
노란색 버튼(A 버튼)을 눌렀을 때, 현재 박자를 출력합니다. ( 예시: 120bpm )


힌트입니다.
#1 두 개의 변수를 만듭니다. (변수: Tempo, Delay)
#2 일반적인 메트로놈 범위는 40에서 200 bpm 입니다.
#3 60 bpm인 Tempo(박자)는 포인터가 한쪽 끝에서 다른쪽까지 1초에 60번 움직임을 의미합니다.
#4 Tempo(박자)가 빨라질수록 Delay가 적어집니다.

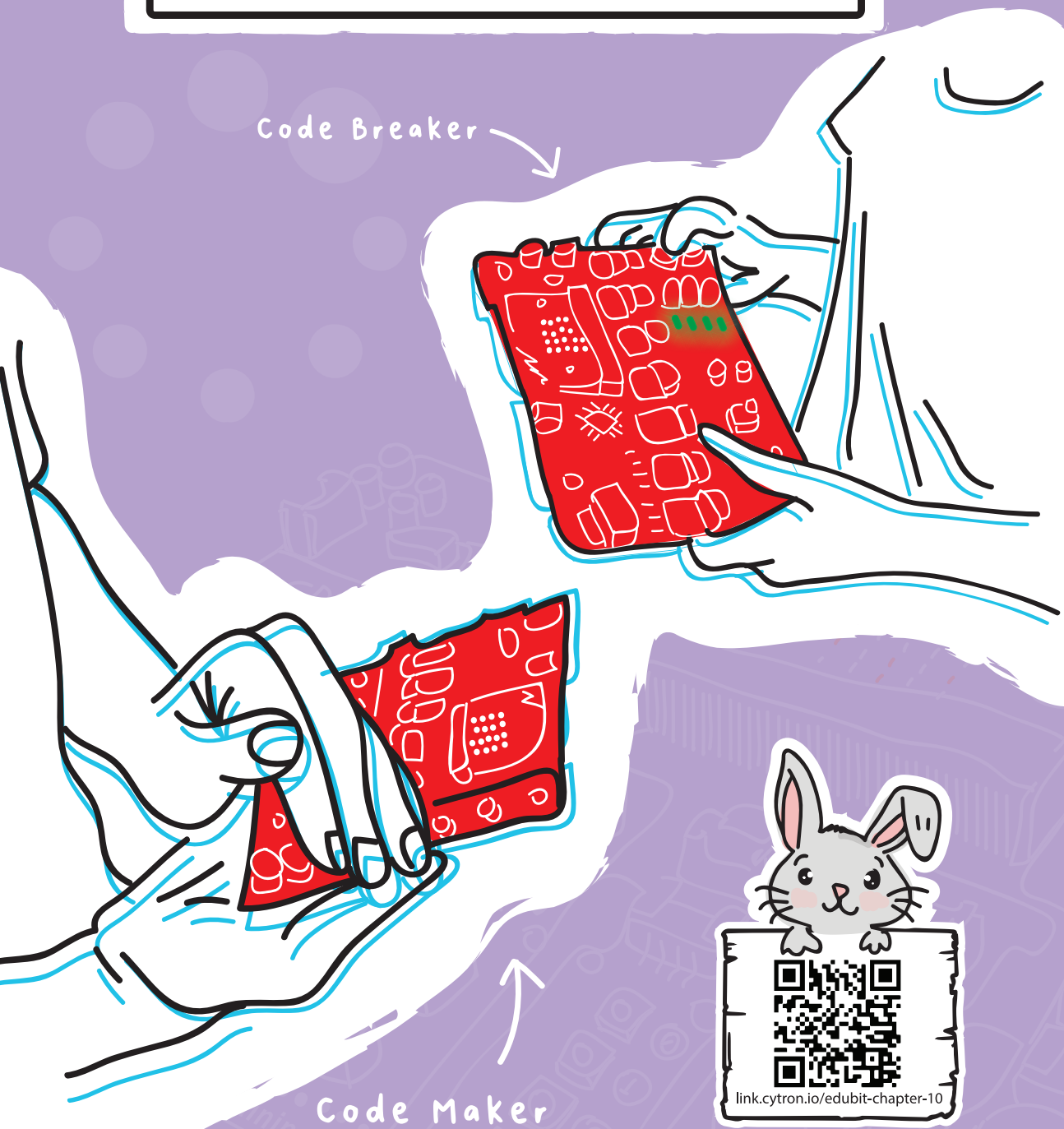
메트로놈은 사용자가 보통 bpm(분당 박자 개수)으로 설정한
일정한 시간 간격으로 딸깍 소리를 내는 장치입니다.
일반적으로 소리뿐만 아니라 시각적 움직임이 동시에 이루어집니다.
음악가들은 일정한 박자로 연주하는 것을 연습하기 위해
이 장치를 사용했습니다.

- 위키피디아 -

# CHAPTER 10

## 비밀코드, 풀 수 있으면 풀어보시지!

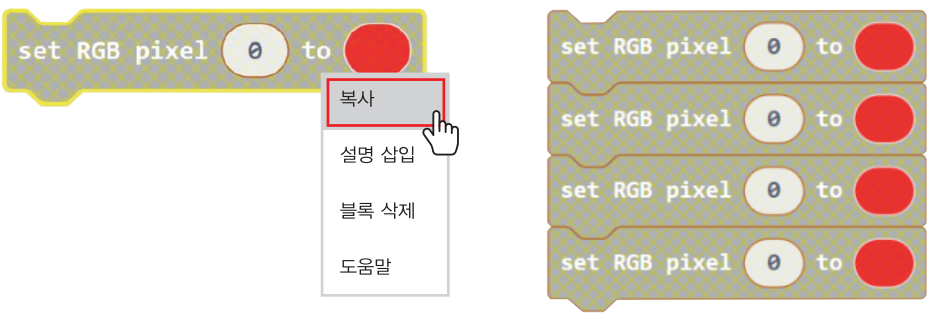
RGB Bit



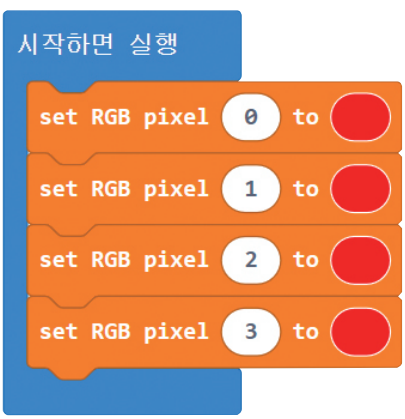
## 코딩을 시작해봅시다!

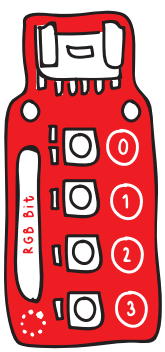
Step 1 MakeCode Editor에서, 새 프로젝트를 생성하고 EDU:BIT 확장 프로그램을 추가합니다.
[ RGB Bit ] 카테고리를 클릭하고 [ set RGB pixel _ to _ ] 블록을 선택합니다.


Step 2 작업공간에서, [ set RGB pixel _ to _ ] 블록을 마우스 오른쪽 버튼으로 클릭하고 '복사'를 클릭합니다.
[ set RGB pixel _ to _ ] 블록이 네 개가 될 때까지 반복합니다.



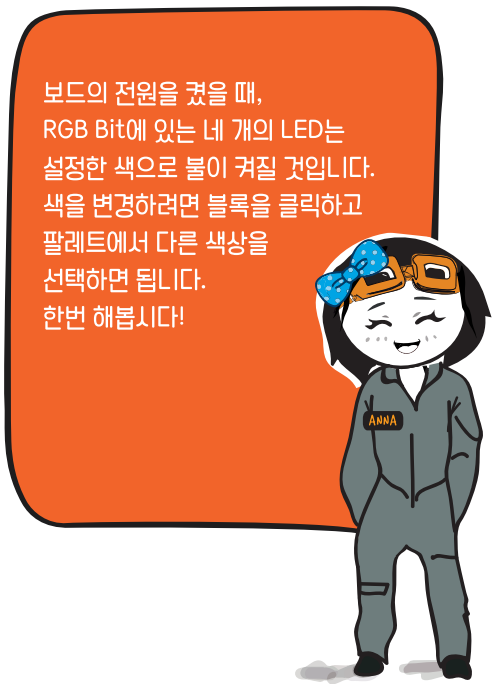

Step 3 [ 시작하면 실행 ] 블록의 슬롯에 블록들을 맞추어 넣습니다. 두 번째에서 네 번째 블록까지 RGB 픽셀 번호를 각각 1, 2, 3으로 변경합니다.



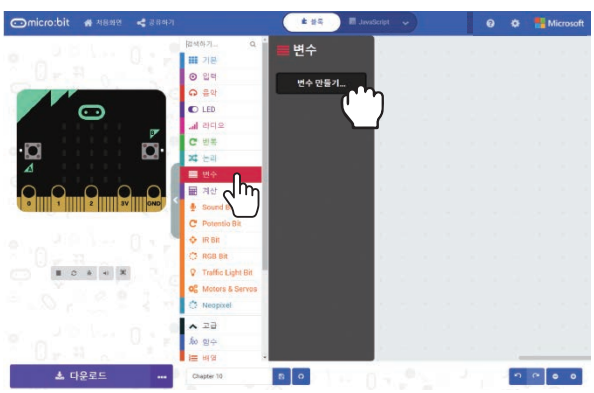




RGB Bit에는 4개의 RGB LED가 있고
각각 식별 번호(0~3)가 할당되어 있습니다.
각 LED를 프로그래밍 하기 위해서
이 번호를 사용합니다.

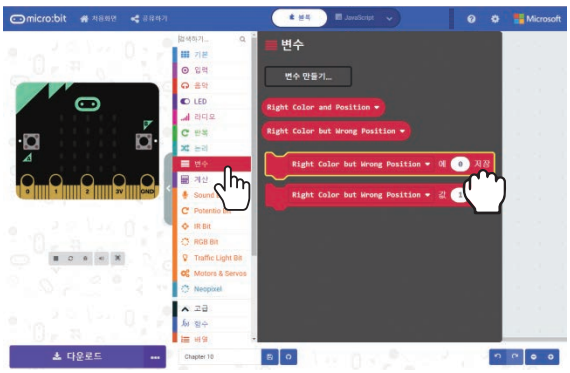
Step 4 아래에 나와있는 것처럼 픽셀의 색을 변경하고 EDU:BIT에 코드를 플래시합니다.


Step 5 두 개의 새로운 변수들을 추가하고 이름을 'Right Color and Position'과 'Right Color but Wrong Position'으로 설정합니다.

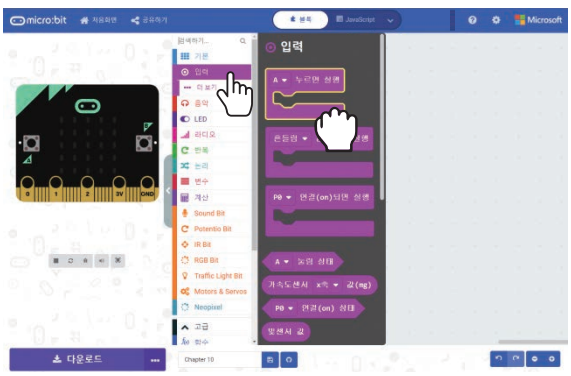



Step 6 [ 변수 ] 카테고리를 클릭하고 [ _ 에 _ 저장 ] 블록을 선택합니다. 블록을 복사하고 [ 시작하면 실행 ] 블록에 맞추어 넣습니다. 한 변수는 'Right Color and Position'으로 설정하고, 다른 하나는 'Right Color but Wrong Position'으로 설정합니다.



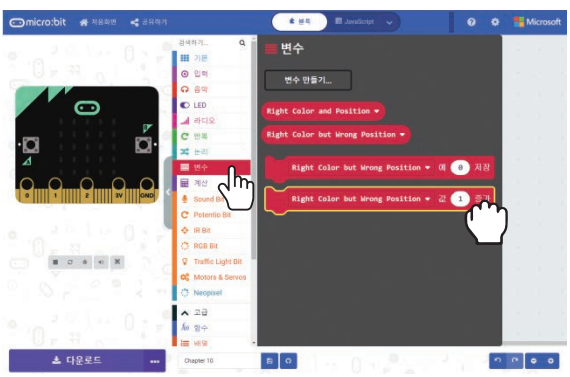


Step 7 [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다. 블록을 두 번 복사합니다. 두 번째 블록은 B, 세 번째 블록은 A+B로 변경합니다.



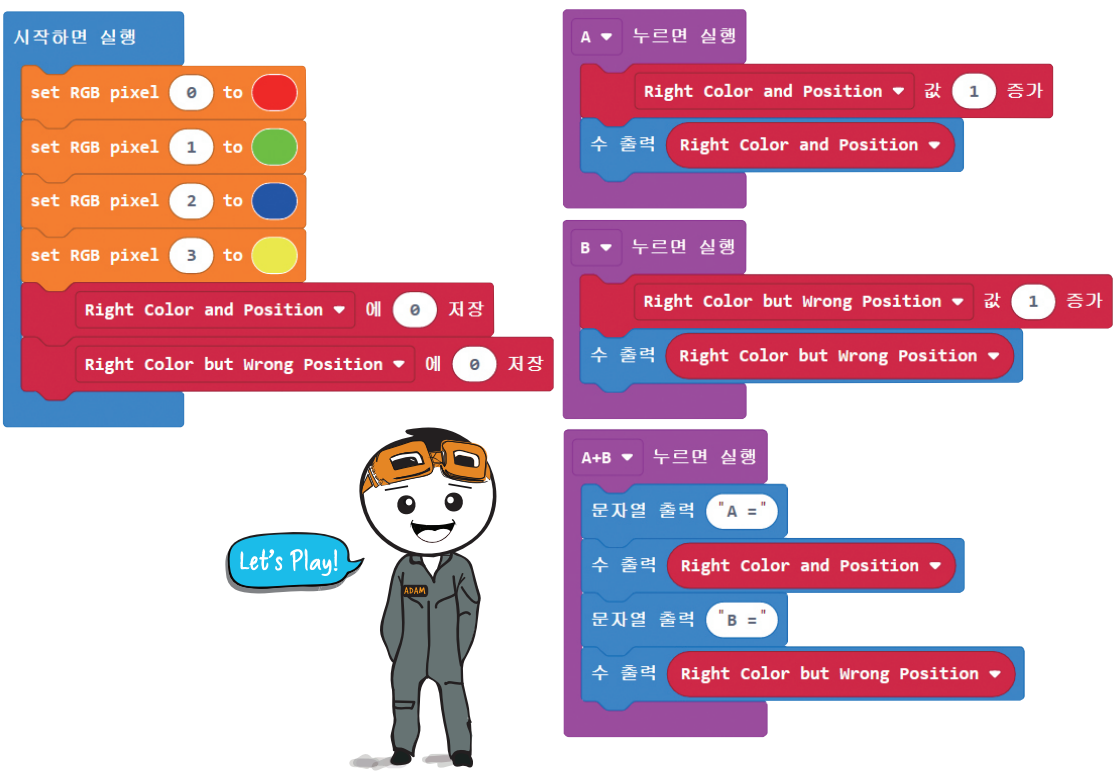


Step 8 [ 변수 ] 카테고리를 클릭하고 [ _ 값 _ 증가 ] 블록을 선택합니다. 복사해서 [ A 누르면 실행 ] 과 [ B 누르면 실행 ] 블록에 맞추어 넣습니다. 한 변수는 'Right Color and Position'으로 설정하고, 다른 하나는 Right Color but Wrong Position'으로 설정합니다.




Step 9 [ 기본 ] 카테고리에서 [ 수 출력 ]과 [ 문자열 출력 ] 블록을 추가하고,
[ 변수 ] 카테고리에서 [ Right Color and Position ]과 [ Right Color but Wrong Position ] 블록을 추가합니다.

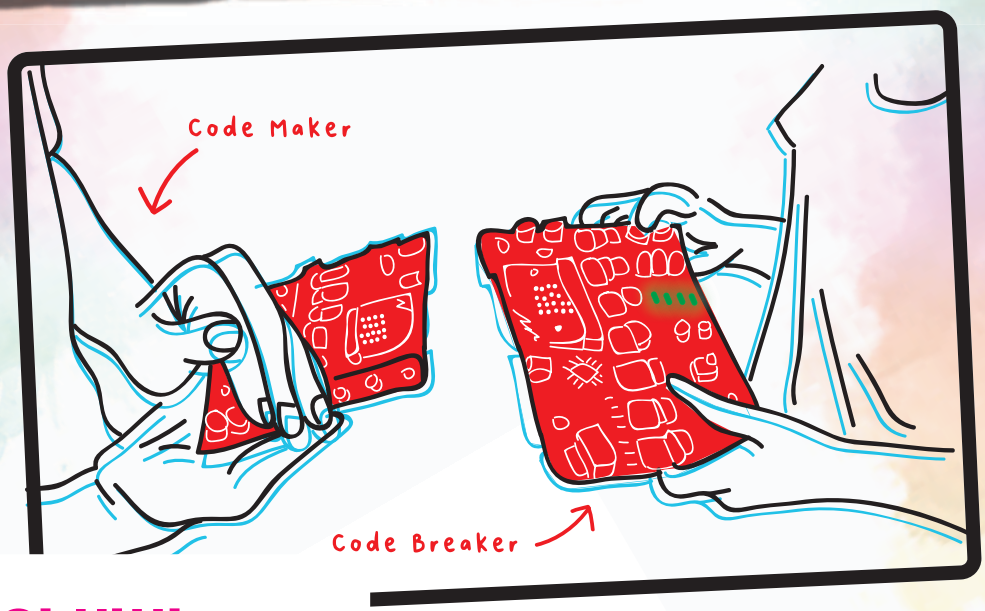

Step 10 [ 문자열 출력 ] 블록에서 "Hello!"를 각각 = 과 = 로 변경합니다.
완성된 코드입니다.




Step 11 코드를 EDU:BIT에 플래시합니다. 이제 비밀코드 풀기 게임을 해봅시다.

# 게임하기

비밀코드 풀기 게임




## 게임 방법:

한 사람은 코드 메이커가 되어 원하는 대로 네 개의 LED에 불을 켬으로써 비밀 코드를 만듭니다. 첫 번째 자리는 빨간색이나 초록색만 가능하도록 제한합니다. 다른 사람이 색 패턴을 볼 수 없도록 보드를 가려야 함을 기억하세요.

다른 사람은 코드 브레이커가 되어 비밀 코드를 추측합니다. 자신의 EDU:BIT로 RGB LED들의 불을 켜고 코드 메이커에게 보여줍니다.

코드 메이커는 확인한 후, 코드 브레이커의 EDU:BIT에서 노란색 버튼(A 버튼)과 파란색 버튼(B 버튼)을 눌러 몇 개의 LED가 'Right Color and Position'(색과 위치 둘 다 정답)이고 'Right Color but Wrong Position'(색은 정답이지만 위치가 오답)인지 표시합니다.

코드 브레이커는 결과를 확인하기 위해 노란색과 파란색 버튼을 동시에 누릅니다.

코드 브레이커가 색의 순서를 올바르게 추측할 때까지 추측하고 확인하는 과정을 라운드 당 최대 10번 반복합니다.

다음 라운드에서는 역할을 바꿉니다.

### 승자는 누구일까요?

코드 브레이커가 색 순서를 추측해서 코드를 성공적으로 맞추었다면 게임의 승자가 됩니다. 실패했다면, 코드 메이커가 승자가 됩니다.

# 더 많은 블록 탐구하기

#1 [ set RGB pixel _ to _ ] 블록을 [ show rainbow on RGB pixels ]블록으로 대체함으로써 RGB Bit의 LED에 무지개 색을 출력할 수 있습니다.

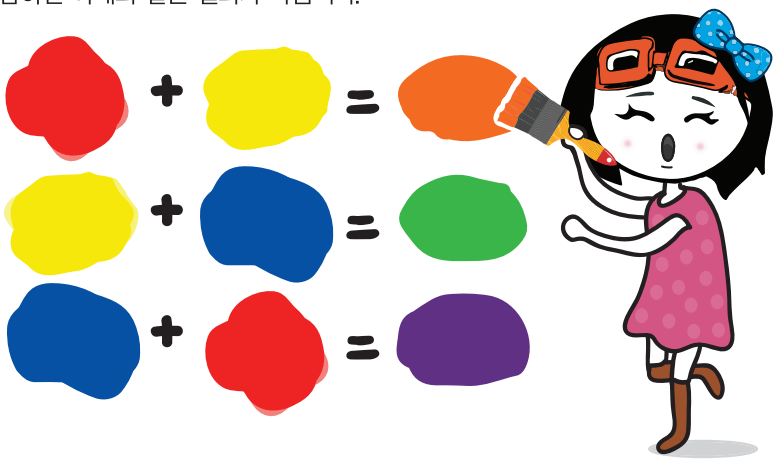

#2 [ 무한반복 실행 ] 블록에 [ rotate RGB pixels color by _ ] 블록을 넣음으로써 빛이 움직이는 효과를 만들 수 있습니다. 효과를 눈으로 확인하기 위해 [ 일시중지 _ (ms) ] 블록을 추가해서 프로그램 속도를 늦춰야 합니다.
예시 코드입니다.


#3 [ 무한반복 실행 ] 블록에 [ shift RGB pixels color by _ ] 블록을 사용해서 픽셀이 한 칸씩 움직이게 할 수 있습니다. [ 일시중지 _ (ms) ] 블록을 추가해서 프로그램 속도를 늦추면, 픽셀이 한 칸씩 꺼지는 것을 눈으로 확인할 수 있습니다.
예시 코드입니다.

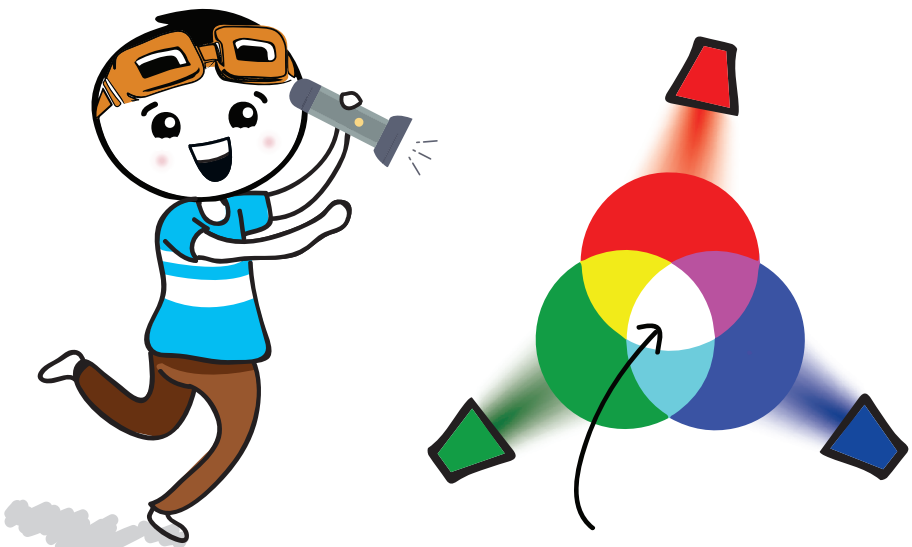

#2와 #3에서 값을 양수에서 음수로 바꾸면 효과의 방향을 변경할 수 있습니다.

# 재미있는 사실!

미술 수업에서 3원색은 빨간색, 노란색, 파란색이라고 배웠죠?
그 색들을 혼합하면 아래와 같은 결과가 나옵니다.



하지만 TV와 컴퓨터 화면처럼 색을 표현하기 위해 빛을 사용하는 장치에서는 RGB 컬러 모델이 사용됩니다.



이 모델에서 3원광은 빨간색(R), 초록색(G), 파란색(B)이고, 이들을 혼합하면 흰색 빛이 만들어집니다.

# 응용 과제

EDU:BIT를 기억력 게임 훈련 수단으로 사용할 수 있도록 프로그래밍 해봅시다.

## 어떻게 동작할까요?

- EDU:BIT를 왼쪽으로 기울이면 프로그램이 시작되고, RGB Bit의 LED에 몇 초 동안 랜덤 색 패턴으로 불이 들어옵니다.
- RGB Bit의 LED를 관찰하다가 불이 꺼지면 색의 순서를 말합니다.
- 답을 확인하기 위해 파란색 버튼(B 버튼)을 누르면, 동일한 패턴의 RGB LED들이 다시 켜집니다.
- 만약 정확하게 답했으면, 노란색 버튼(A 버튼)을 눌러 점수를 부여하고 현재 점수를 보여줍니다. 만약 오답이면, 게임이 종료됩니다.
- LED에 불이 켜져 있는 시간을 늘리거나 줄이기 위해 Potentio Bit의 손잡이를 돌려 게임의 난이도를 조절할 수 있습니다.
- 가장 높은 점수를 얻은 사람이 승리합니다!

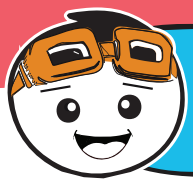

힌트입니다.
#1 두 개의 변수를 생성합니다. (변수: Score, Pattern)
#2 각 Pattern에 대해 RGB LED의 색 순서를 미리 설정합니다.
게임을 더 어렵게 하기 위해서 더 많은 색을 사용하거나 더 많은 Pattern을 미리 설정할 수 있습니다. 반대로, 어린 친구들을 위해서 색과 Pattern을 제한해도 됩니다.

# BONUS CHAPTER

# LED로 Simon Says 게임하기

라디오 통신

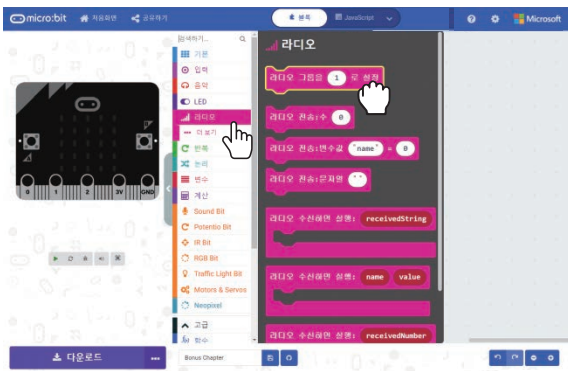
link.cytron.io/edubit-bonus-chapter



어떤 형태의 통신이 일어나기 위해서는 적어도 발신자와 수신자가 필요합니다. 이 게임에서는, 라디오 방송 신호를 발신하고 수신하면서 서로 통신하기 위해 두 개의 EDU:BIT가 필요합니다.

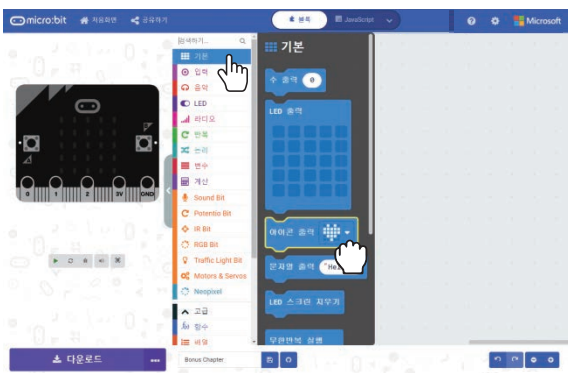
## 코딩을 시작해봅시다!

Step 1 MakeCode Editor에서 새 프로젝트를 생성하고 EDU:BIT 확장 프로그램을 추가합니다. [ 라디오 ] 카테고리를 클릭하고 [ 라디오 그룹을 _ 로 설정 ] 블록을 선택합니다. [ 시작하면 실행 ] 블록에 맞추어 넣습니다.



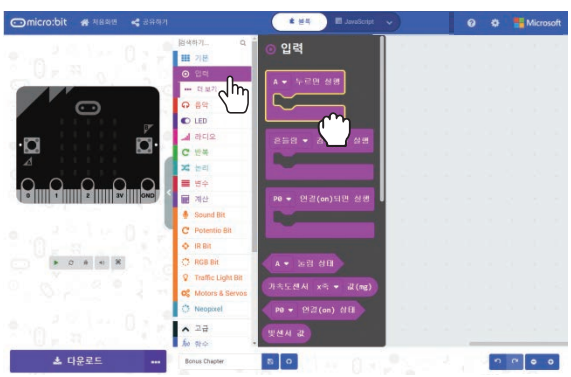


Step 2 [ 기본 ] 카테고리를 클릭하고 프로그램에 [ 아이콘 출력 ] 블록을 추가합니다.

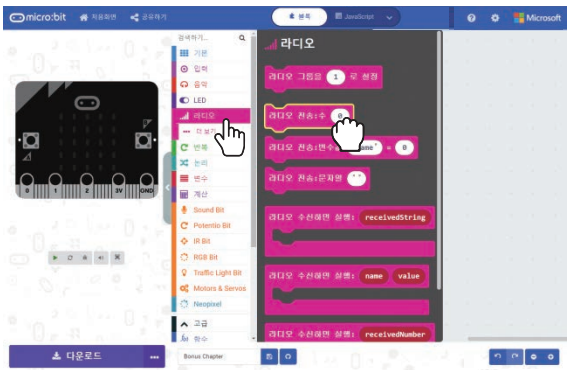




Step 3 [ 입력 ] 카테고리를 클릭하고 [ _ 누르면 실행 ] 블록을 선택합니다.

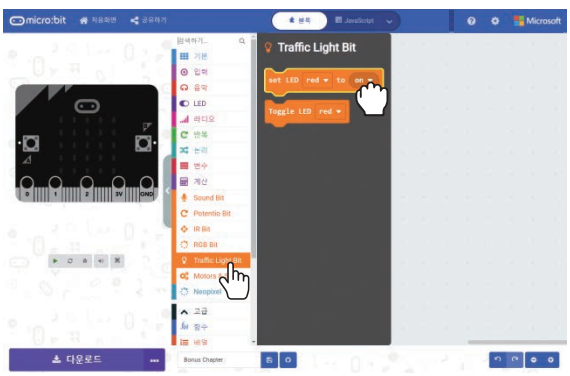



Step 4 [ 라디오 ] 카테고리를 클릭하고 [ 라디오 전송: 수 _ ] 블록을 선택합니다. 값을 1로 변경합니다.



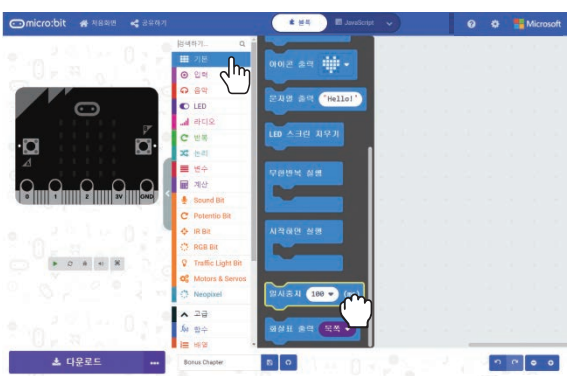


Step 5 [ Traffic Light Bit ] 카테고리를 클릭하고 [ set LED _ to _ ] 블록을 선택합니다. 블록을 복사하고 [ A 누르면 실행 ] 블록에 맞추어 넣습니다. 두번째 블록에서 설정을 'off'로 변경합니다.





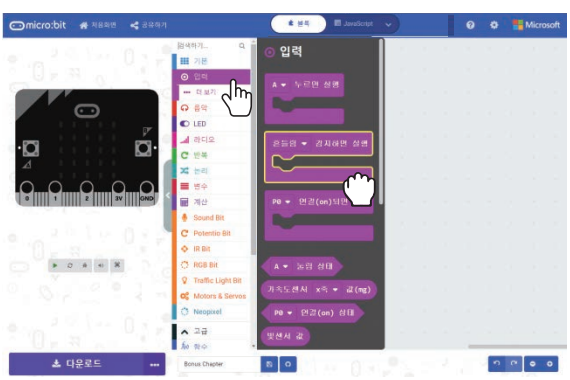
Step 6 [ 기본 ] 카테고리를 클릭하고 [ 일시중지 _ (ms) ]를 선택합니다. [ set LED _ to _ ] 블록들 사이에 끼워 넣고 값을 500으로 변경합니다.

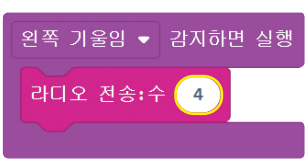


Step 7 [ A 누르면 실행 ] 블록을 마우스 오른쪽 버튼으로 클릭하고 '복사'를 선택합니다.
같은 블록이 세 개가 될 때까지 반복합니다.

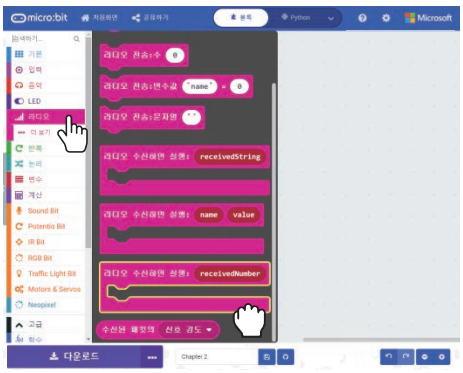

Step 8 아래에 나와있는 것처럼 복사된 블록들의 버튼, 숫자, 색을 변경합니다.

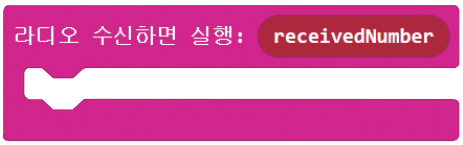

Step 9 [ 입력 ] 카테고리를 클릭하고 [ _ 감지하면 실행 ] 블록을 선택합니다. '왼쪽 기울임'으로 설정을 변경합니다. [ 라디오 ] 카테고리를 클릭하고 [ 라디오 전송: 수 _ ] 블록을 선택합니다. 값을 4로 변경합니다.



Step 10 [ 라디오 ] 카테고리를 클릭하고 [ 라디오 수신하면 실행: receivedNumber ] 블록을 선택합니다.

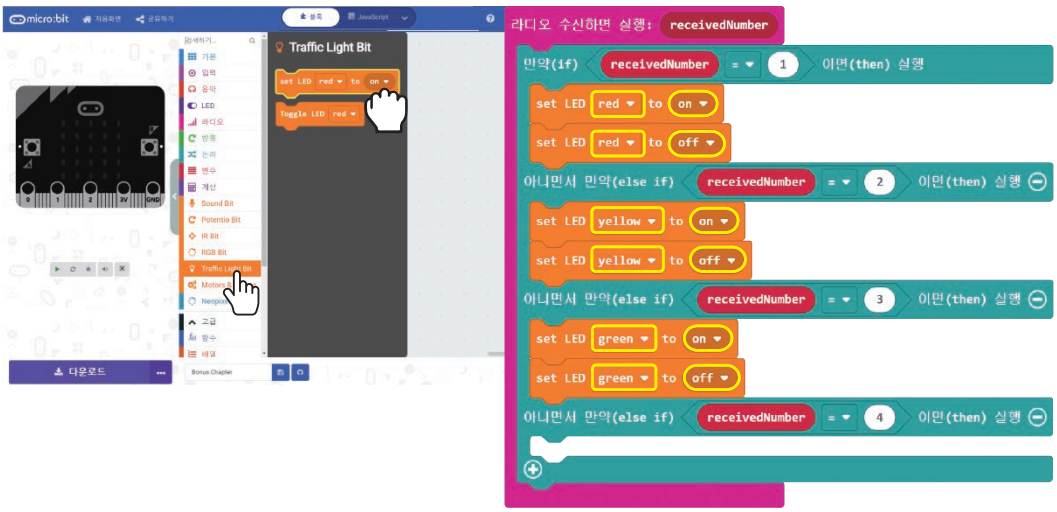




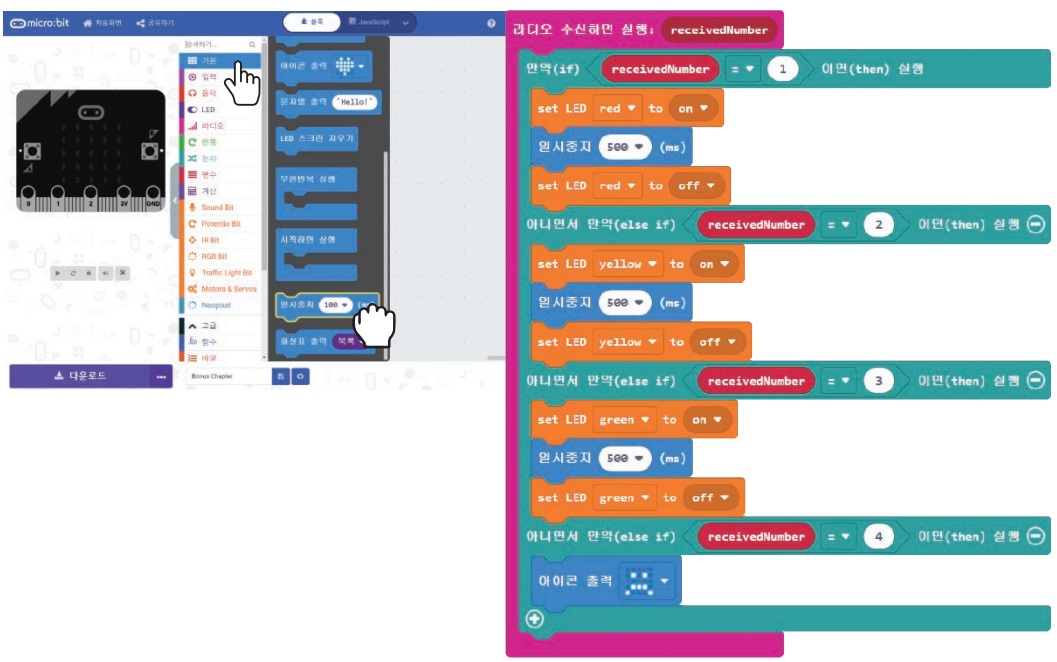

Step 11 [ 논리 ] 카테고리를 클릭하고 [ 만약(if) _ 이면(then) 실행 아니면(else) 실행 ] 블록을 선택합니다. (+)버튼을 클릭해서 [ 아니면서 만약(else if) _ 이면(then) 실행 ] 조건문들을 추가하고 (–) 버튼을 눌러서 [ 아니면(else) 실행 ] 조건을 제거합니다. [ 논리 ] 카테고리의 [ _ = _ ] 비교 연산 블록을 각 조건문의 슬롯에 추가합니다.


Step 12 'receivedNumber' 변수를 클릭한 후 비교 연산 블록으로 드래그하고 값을 각각 1, 2, 3, 4로 변경합니다.

Step 13 [ Traffic Light Bit ] 카테고리를 클릭하고 [ set LED _ to _ ] 블록을 선택합니다. 블록을 복사하고 앞에서부터 세 개의 슬롯에 맞추어 넣습니다. 아래에 보이는 것처럼 색과 on/off 설정을 변경합니다.




Step 14 [ 기본 ] 카테고리를 클릭하고 [ 일시중지 _ (ms) ] 블록을 선택합니다. 블록을 복사해서 [ set LED _ to _ ] 블록들 사이에 끼워 넣고, 값을 500으로 변경합니다. [ 기본 ] 카테고리의 [ 아이콘 출력 ] 블록을 선택해서 마지막 슬롯에 맞추어 넣고, 아이콘을 '슬픔'으로 변경합니다.




Step 15 [ 음악 ] 카테고리를 클릭하고 [ _ 멜로디 _ 출력 ] 블록을 코드에 추가합니다.
'와와와와아' 멜로디나 원하는 멜로디로 변경합니다.
완성된 코드입니다.


Step 16 EDU:BIT에 완성된 코드를 플래시합니다. 함께 통신할 친구의 EDU:BIT에도 코드를 플래시해야 합니다.

통신할 EDU:BIT들의 전원이 모두 켜져있을 때,
친구의 EDU:BIT LED를 켜는 라디오 신호를 보낼 수 있습니다. 반대로,
친구도 자신의 EDU:BIT 버튼을 눌러 내 보드에 있는 LED를 켤 수 있습니다.

# 게임하기

## LED로 Simon Says 게임하기

소통할 EDU:BIT들이 모두 코드를 플래시한 후에 Simon Says 게임을 시작할 수 있습니다.

## 게임 방법:

두 사람은 돌아가면서 'Simon'이 됩니다.
순서가 돌아오면 버튼을 눌러 빨간색, 노란색, 초록색 LED의 불을 켭니다.

- 게임을 시작하기 위해, 한 사람은 Traffic Light Bit의 LED 중 하나를 켜야합니다.
- 다른 사람은 관찰하다가 같은 LED를 켜고, 이어서 다른 LED를 켭니다.
- 게임은 두 사람이 돌아가면서 지난 순서들을 반복하고, 새로운 LED를 켜서 순서에 추가합니다.
- 만약 어떤 사람이 LED를 잘못 켜면, 상대방은 EDU:BIT를 기울여 게임을 종료시킵니다.
- 진 사람은 새 게임을 시작하기 위해 EDU:BIT를 초기화해야 합니다.

이 게임은 처음에는 쉽지만 매 순서마다 점점 더 길고 복잡해집니다.
이기기 위해서, 순서를 주의 깊게 관찰해야 합니다. 기억력을 훈련시킬 수 있는 재미있는 게임입니다.

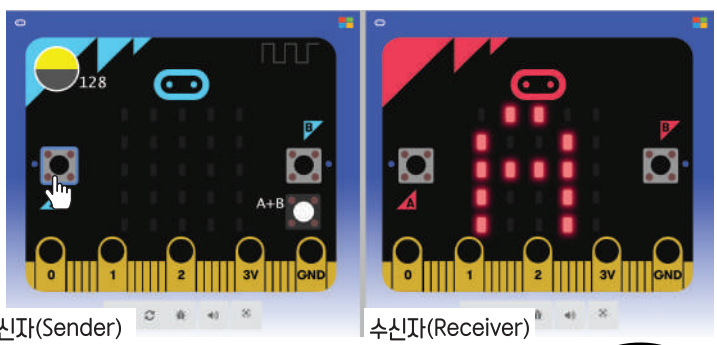
# 더 많은 블록 탐구하기

#1 숫자를 전송하는 것 외에도, [ 라디오 전송: 문자열 _ ] 블록을 사용해서 문자 메시지를 보낼 수 있습니다. 문자열 방송 신호를 수신하기 위해서 [ 라디오 수신하면 실행: receivedString ] 블록을 사용해야 합니다. 문자열 최대 길이는 19자입니다.

#2 만약 문자와 숫자를 동시에 주고받고 싶다면, [ 라디오 전송: 변수값 _ = _ ]과 [ 라디오 수신하면 실행: name value ] 블록을 사용합니다. 문자열 최대 길이는 8자 입니다.

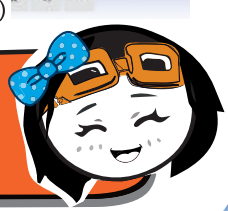
만약 여러 개의 EDU:BIT를 이용할 수 없다면, makecode.com/multi 에 접속해서 라디오 통신 함수를 테스트해 볼 수 있습니다. 'sender'(발신자)와 'receiver'(수신자) 코드를 작성하면, 시뮬레이터 창에서 결과를 볼 수 있습니다.

전체화면으로 보기 위해서 micro:bit 보드를 클릭하거나 '전체화면으로 실행' 버튼을 클릭합니다.




발신자(Sender) 수신자(Receiver)

'발신자' micro:bit에서 A 버튼을 누르면, '수신자' micro:bit는 라디오 신호를 수신하고 받은 문자열을 출력합니다. A+B 버튼을 누르면 어떤 일이 발생할까요?



# 재미있는 사실!

EDU:BIT를 뒤집으면, 내장된 라디오와 블루투스 안테나를 볼 수 있습니다.


안테나는 위성 전송뿐만 아니라 TV나 라디오 방송에도 흔히 사용되는 전자기파 형태의 신호를 전송합니다.

## 전자기파 스펙트럼


## 주 목 !

EDU:BIT가 다른 EDU:BIT들로부터 라디오 방송 신호를 수신하고 전송하기 위해서, 모두 동일한 라디오 그룹으로 설정해야 합니다.

# 응용 과제

학급을 위한 피드백 네트워크 통신 시스템을 설정합니다.

## 어떻게 동작할까요?

학급의 모든 EDU:BIT는 같은 라디오 그룹으로 설정합니다.
선생님의 EDU:BIT는 '문자열' 신호를 수신했을 때 수신한 문자를 스크롤하고,
'숫자'를 수신했을 때 해당하는 Traffic Light Bit의 LED에 불이 들어옵니다.

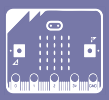
| 수신된 숫자 | 켜지는 LED | 의미 |
|---|---|---|
| 1 | 빨간색 | A / NO / 거짓 |
| 2 | 노란색 | B / 아마도 / 불확실 |
| 3 | 초록색 | C / YES / 참 |

학생들은 EDU:BIT로 자신의 이름이 적힌 문자열과 선생님의 EDU:BIT에서 켜고 싶은 LED에 해당하는 숫자를 전송합니다.

1을 전송하기 위해 A 버튼을 누릅니다.
2를 전송하기 위해 B 버튼을 누릅니다.
3을 전송하기 위해 A+B 버튼을 누릅니다.

힌트입니다.
텍스트 스크롤 시간을 줄이기 위해서 학생들은 두세 글자의 짧은 별명을 정하고, 이 별명을 문자열로 전송합니다.

# 무엇을 배웠을까요?

- [ ] LED 매트릭스에 글자와 애니메이션을 출력하도록 프로그래밍 하기
- [ ] .hex 파일을 다운로드, 저장, 게시하고 MakeCode에서 편집하기

- [ ] 이벤트 기반의 프로그래밍을 하기 위해서 입력 블록 사용하기
- [ ] 변수를 만들고 사용하기

- [ ] 간단한 멜로디를 연주하기 위해 Music Bit에서 피에조 부저 사용하기
- [ ] 함수 만들고 사용하기
- [ ] 악보 읽기

- [ ] Traffic Light Bit에 있는 LED를 on, off, toggle하도록 프로그래밍 하기
- [ ] MakeCode Editor에 확장 프로그램 추가하기

- [ ] IR Bit를 사용해서 물체를 탐지하도록 프로그래밍 하기
- [ ] while 반복문 사용하기
- [ ] 배열 만들고 사용하기

- [ ] Potentio Bit에서 아날로그 값을 읽어오도록 프로그래밍 하기
- [ ] 읽어온 아날로그 입력 값을 비례 변환하기
- [ ] 조건문을 프로그래밍 하기 위해 논리 블록 사용하기

- [ ] Sound Bit로 소음을 탐지하도록 프로그래밍 하기
- [ ] 다른 모드로 전환하기 위해 이벤트 트리거 사용하기

- [ ] DC모터의 회전 방향과 속도를 제어하도록 프로그래밍 하기
- [ ] 산술 연산을 수행하기 위해 계산 블록 사용하기

- [ ] 서보모터의 각도를 제어하도록 프로그래밍 하기

- [ ] RGB Bit에 있는 RGB LED들을 다른 색/패턴으로 켜도록 프로그래밍 하기

- [ ] 라디오 신호를 보내고 받을 수 있도록 프로그래밍 하기

※ 기술을 잘 익혔다면 체크박스에 표시하고, 아니라면 복습을 위해 관련된 챕터로 되돌아갑니다.

## RERO EDUTEAM @ CYTRON 으로부터 편지가 왔어요~

# 축하합니다!!!

EDU:BIT의 모든 챕터를 공부하면서 MakeCode Editor로 코딩하는 법을 배웠습니다.
직접 게임을 만들어서 해보고, 더 재미있게 수정해보았는데 재미있었나요?
응용 과제는 어렵지 않았나요? 포기하지 않고 끝까지 해냈다는 점 칭찬해 줄게요~

이제 EDU:BIT 보드의 비트들과 micro:bit를 이용해서 무엇을 할 수 있을지 생각해봅시다.
쉿... 혹시 비트들을 떼어낼 수 있다는 사실... 알고 있나요?
원한다면 메인보드에서 분리한 후 제공된 케이블을 사용해 연결하면 됩니다.

자 이제 여러분의 독창적인 아이디어로 새로운 게임과 프로그램을 만들어 봅시다.
여러분이 어떤 놀라운 프로젝트를 내놓을지 정말 기대가 됩니다.

잊지 말고 창작물을 공유해 주세요. 페이스북 페이지에 메시지를 남기거나 메일을 보내주면 됩니다.
소식 기다릴게요~

그럼 안녕,
Adam과 Anna가


Bit..
Bit...