# Instructor Guide

## for teaching micro:bit to students in classes 9-12

Refer: Free Online Course on Micro:bit in Hindi for school students:
https://kalateetkaushal.in/courses/micro:bit-in-hindi/

Timeless Lifeskills
FOUNDATION

# Basic Knowledge of Micro:bit

- This course guide assumes that students have a basic understanding of Micro:bit - what is Micro:bit (a pocket computer), its input and output features (buttons, LEDs, pins), and how to programme the Micro:bit.
- This course further assumes that students are familiar with MakeCode for Micro:bit (interface, how to write simple code, and how to transfer programmes to the Micro:bit).
- If students are NOT familiar with either of the above then it will be better to do a few activities from the Micro:bit course meant for classes 6 to 8.

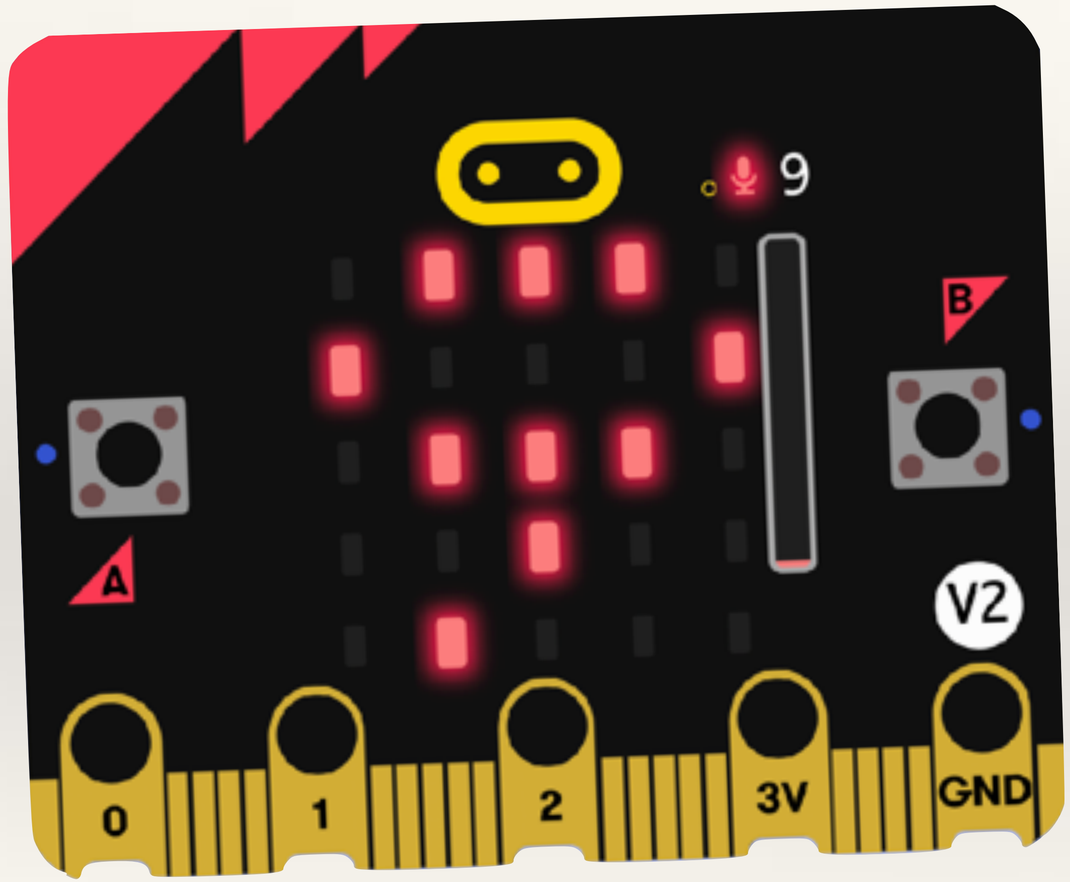
# Course Introduction

- In this course, students will learn about the sensors present onboard the micro:bit and they will do 8 hands-on projects.
- In the process they will learn how, by using sensors and computers, machines can be made autonomous i.e. capable of taking their own decisions. For example, a fan has to be operated by an on-off switch, but by adding a temperature sensor and Micro:bit, we can make it a smart fan that switches itself on when the temperature rises.
- Students will also learn about fundamental programming concepts like variables and conditional statements.

On-board Sensors

# SOUND SENSOR

# SOUND SENSOR




- When the microphone on the Micro:bit hears a sound, it sets a number for how loud the sound is.
- This number is the sound level and it ranges from 0 (low sound) to 255 (loud sound).

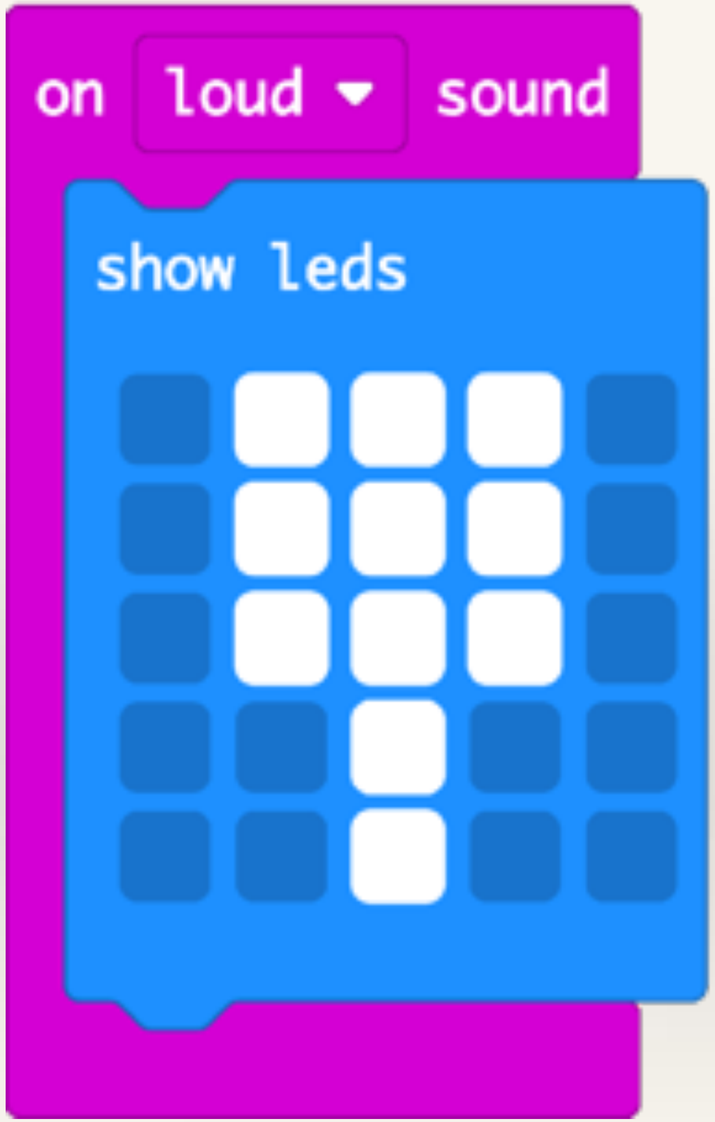
Project-1

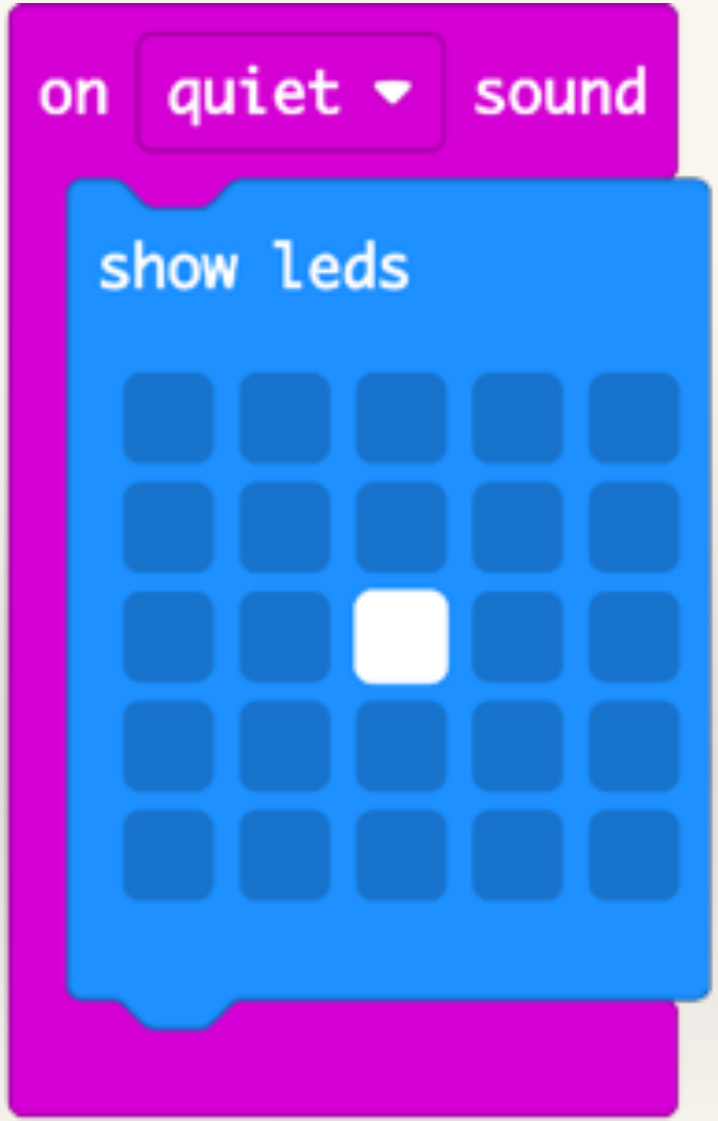
# Sound Sensor Switch

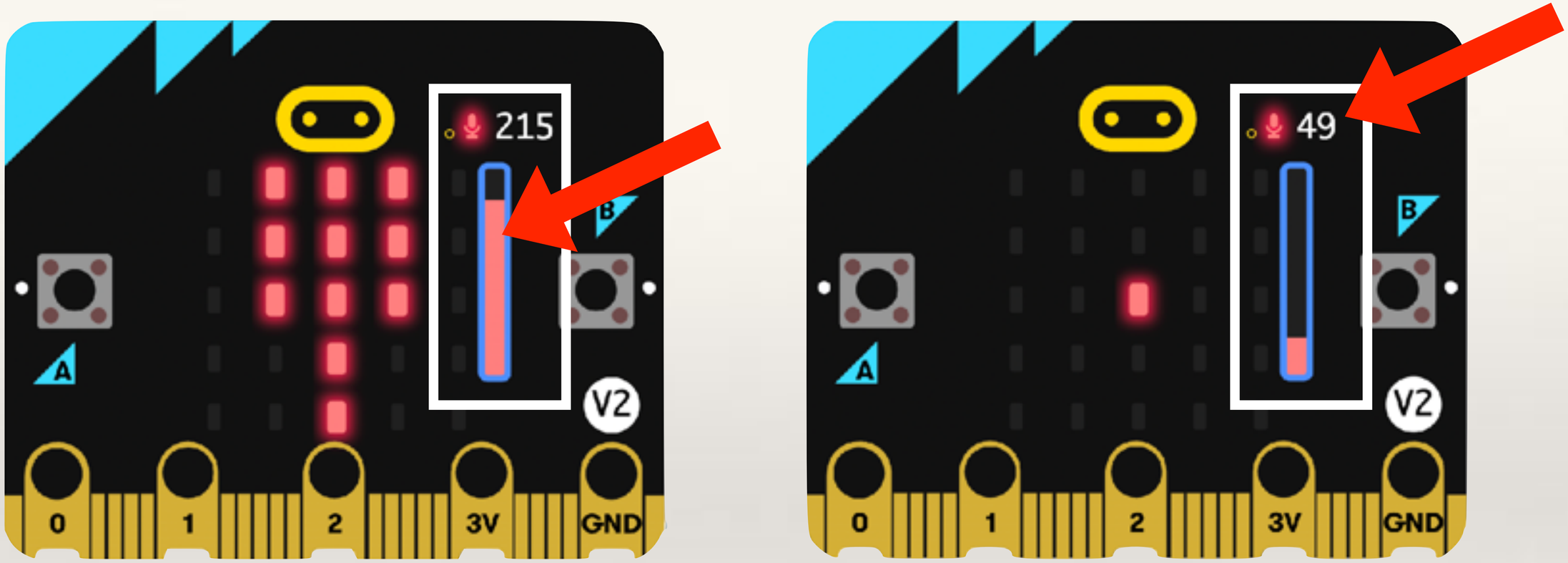
**Objective:** to help students understand sensor-based INPUT on the Micro:bit using the onboard sound sensor (microphone).

**Problem:**

- Students should write code such that when the Micro:bit hears a ‘loud’ sound, a set of onboard LEDs should light up and when the Micro:bit hears a ‘quiet’ sound only one centre LED should light up.
- Students should understand how the MakeCode simulator works to represent loud and quiet sounds.
- Students should transfer the code to Micro:bit and test their code by clapping their hand or making some other loud sound and quiet sound.


on loud sound
show leds


on quiet sound
show leds



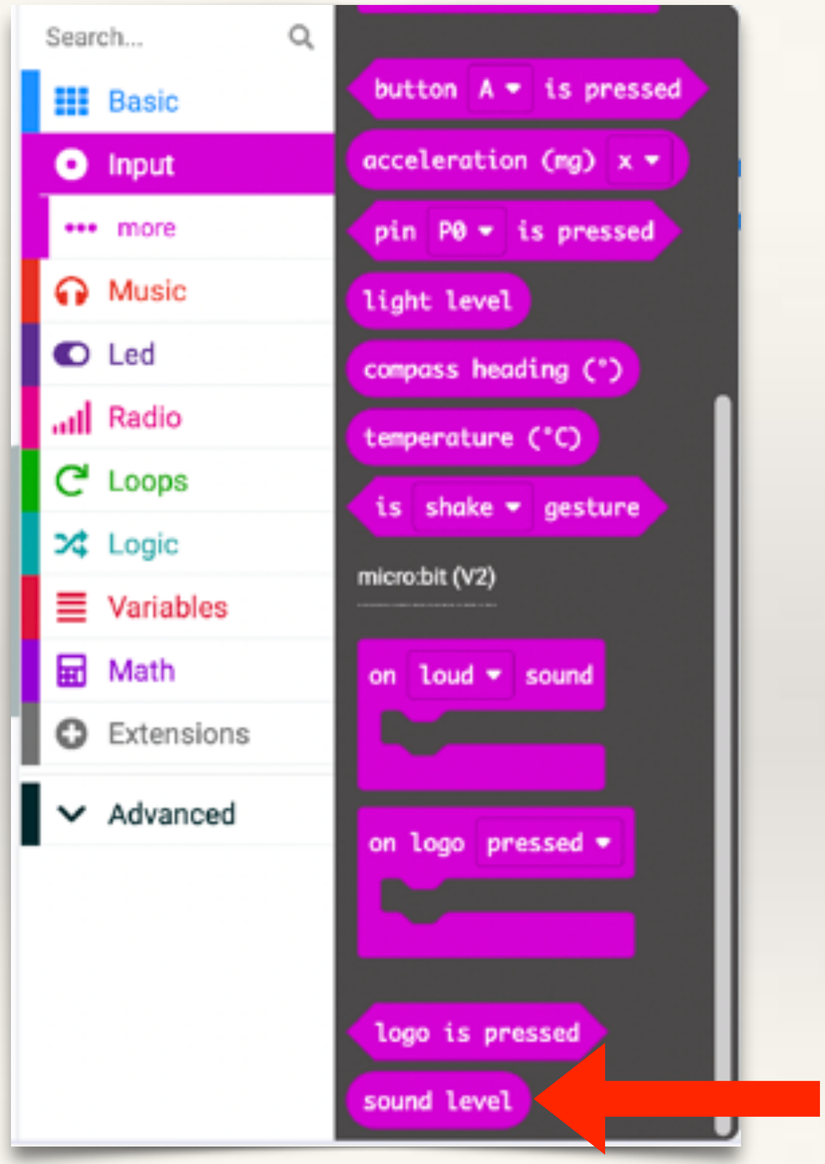

On the MakeCode Simulator, the Slider can be adjusted to depict different sound levels. The value ranges from 0 to 255.

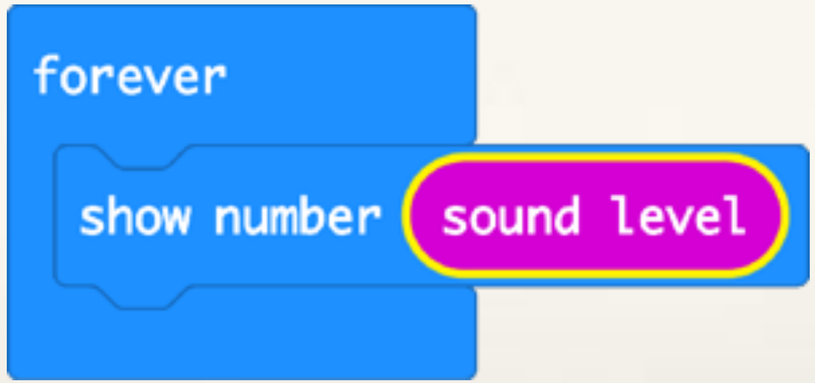
**Extend the Problem:** help students understand how to find out the actual sound level in the classroom. They should further understand how to change default threshold values for quiet and loud sounds in the code.

**Problem:**

- Students should find out the sound level in the classroom. For this, they need to use Show Number and Sound Level blocks.
- Students should write code to define what threshold level is ‘quiet’ and ‘loud’ (the values can range from 0 to 255).
- Students should experiment what threshold level should be defined as ‘loud’ keeping in view the average sound level in the classroom.

# Finding the Sound Level in the Classroom

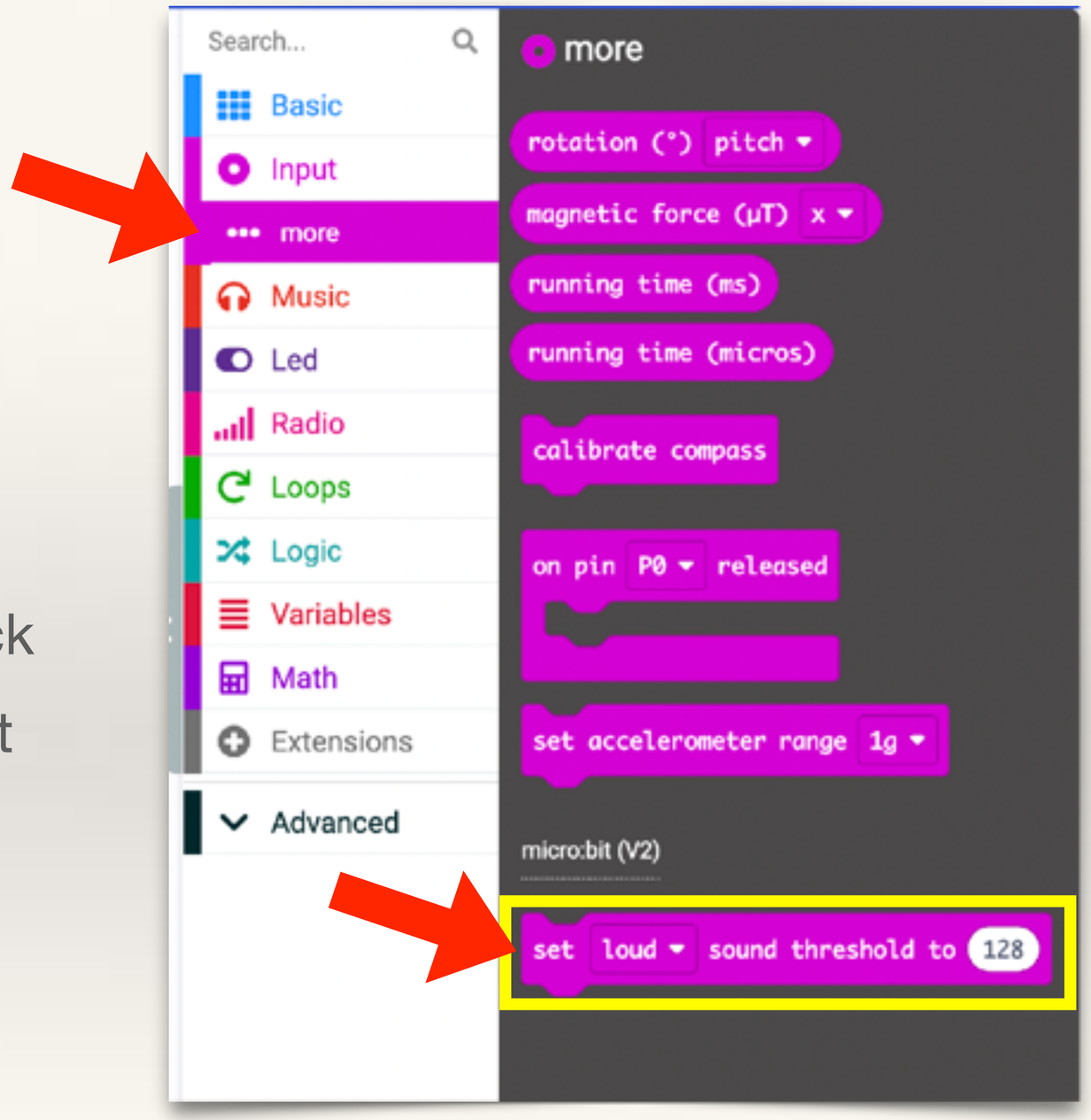


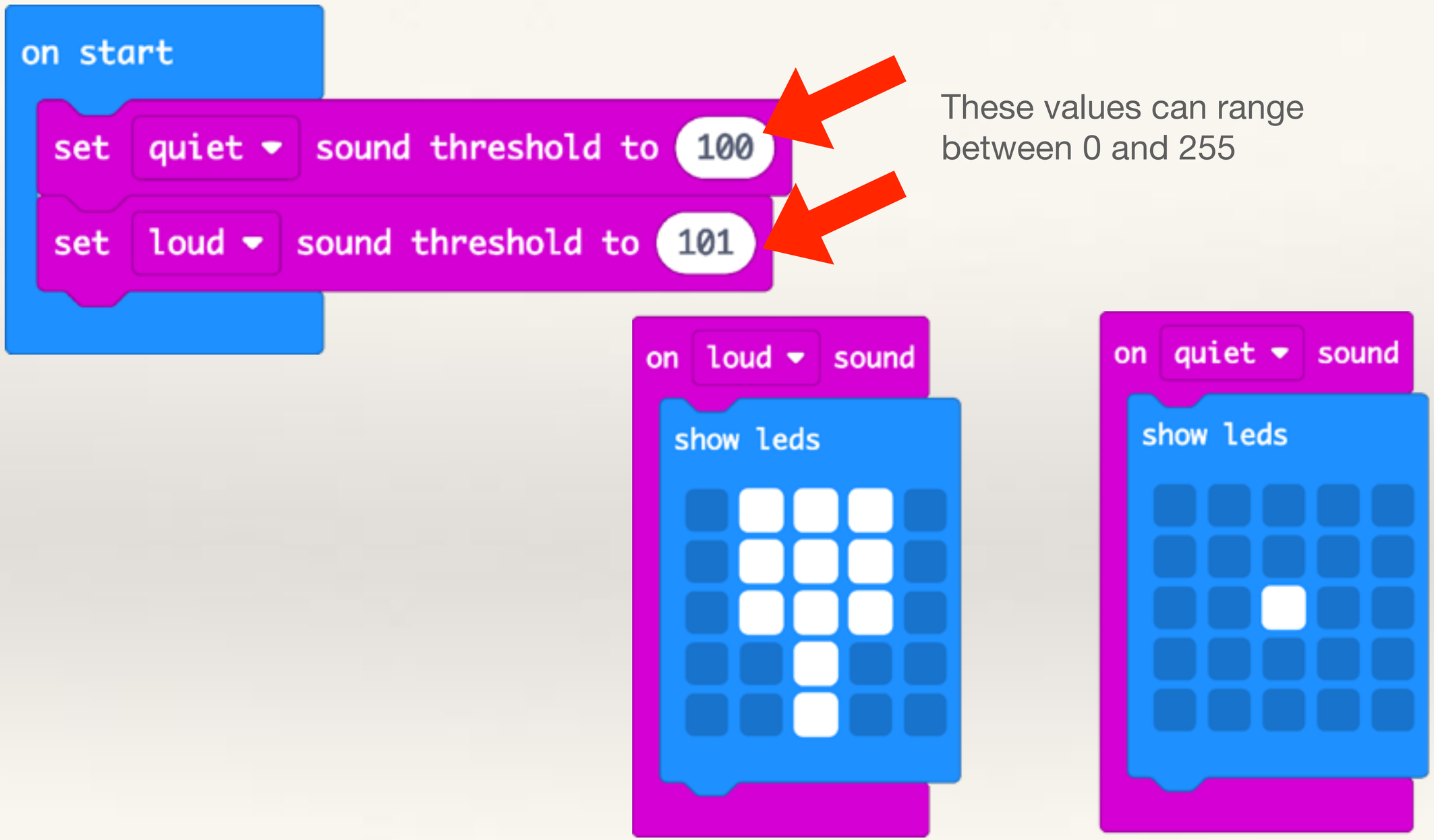




1. Go to Input > scroll down
2. Drag out “Sound Level” block
3. Put this block inside “Show Number” block
4. Put the “Show Number - Sound Level” inside Forever block.
5. Transfer code to Micro:bit. It now displays the sound level in the classroom

## Setting Threshold Level for Loud and Quiet Sound

1. Go to Input > More
2. Drag out “Set Loud Sound Threshold to” block
3. Put this block inside “Start” block
4. Students should understand that any instruction put inside the “Start” block gets executed first when the programme runs

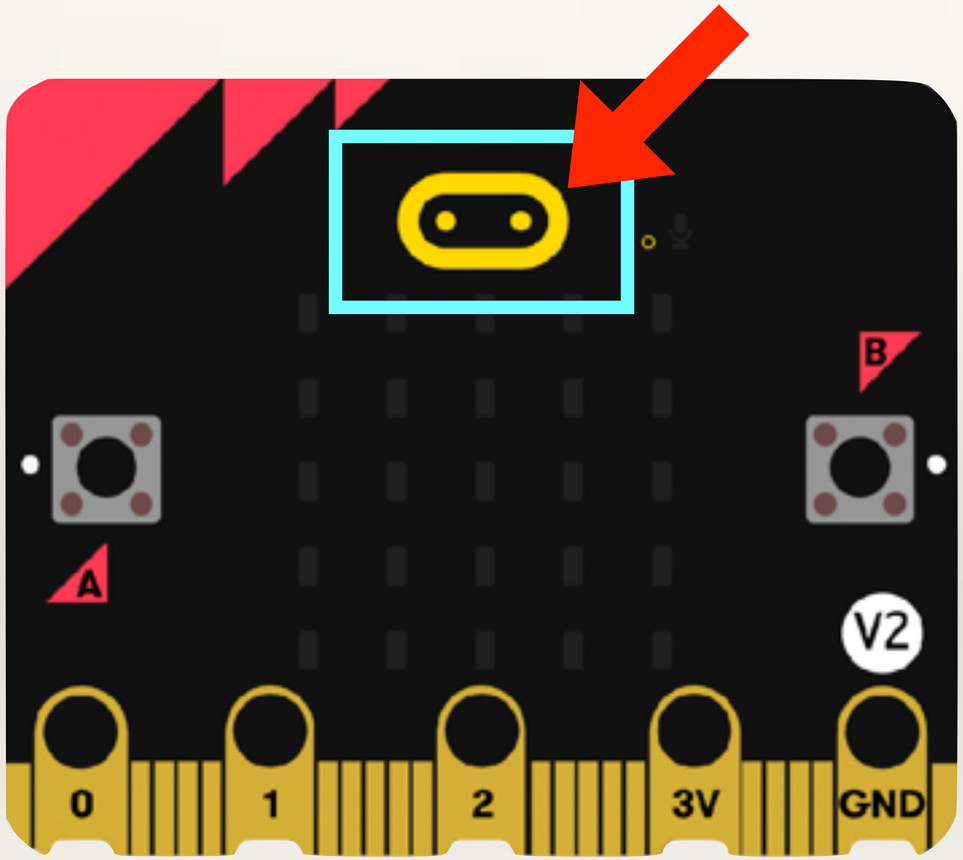



on start
set quiet sound threshold to 100
set loud sound threshold to 101
These values can range between 0 and 255
on loud sound
show leds
on quiet sound
show leds

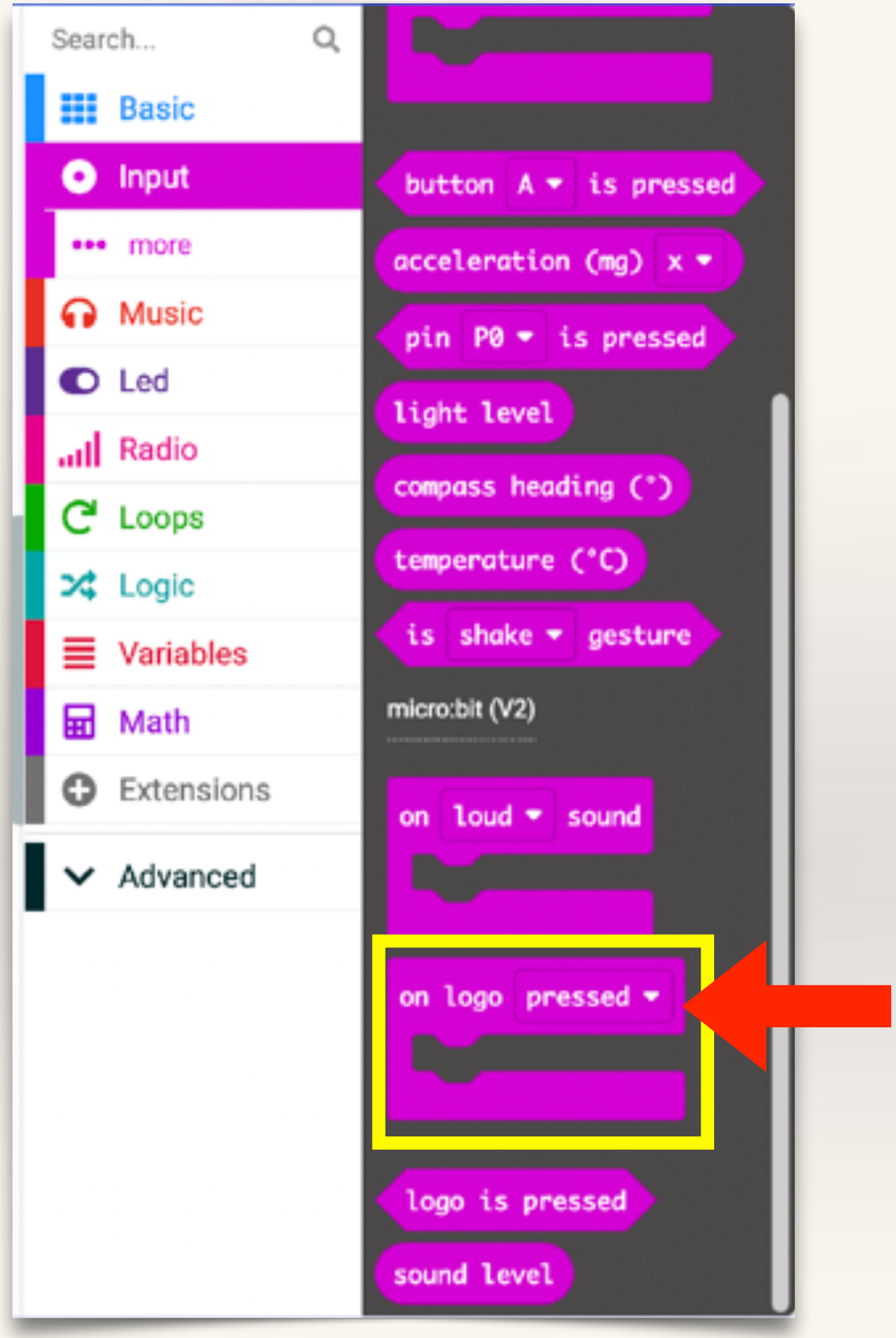
On-board Sensors

# TOUCH SENSOR

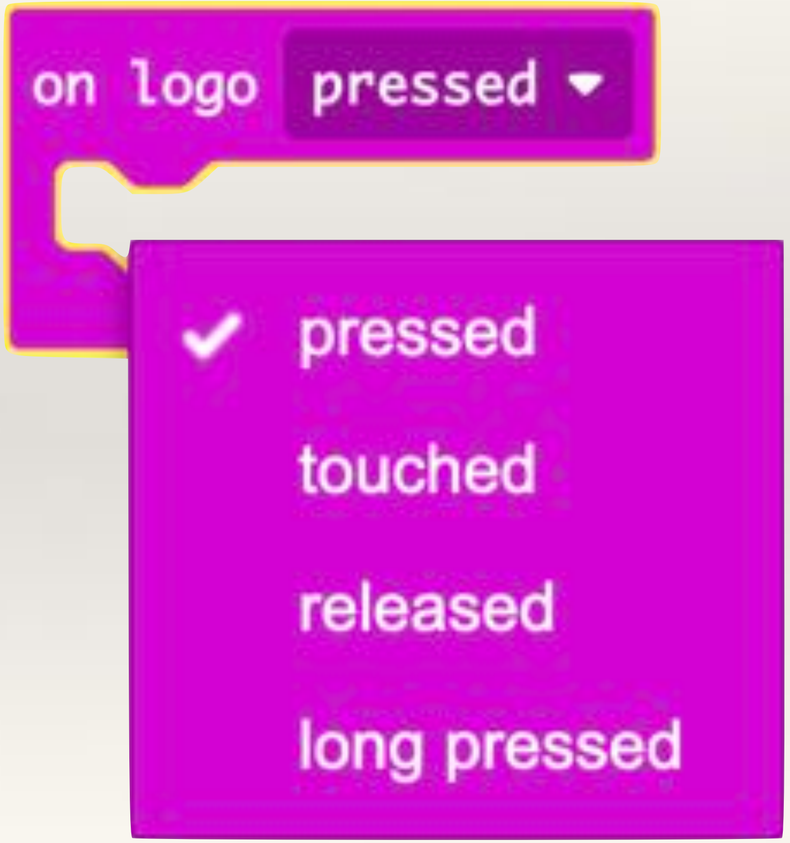
# TOUCH SENSOR




- The logo on top of the Micro:bit will detect your touch.
- Four states detected by this sensor are - Touched, Pressed, Long Pressed, and Released.




1. The Logo Pressed block can be found under Input
2. The drop-down menu displays the four touch sensor options



Project-2

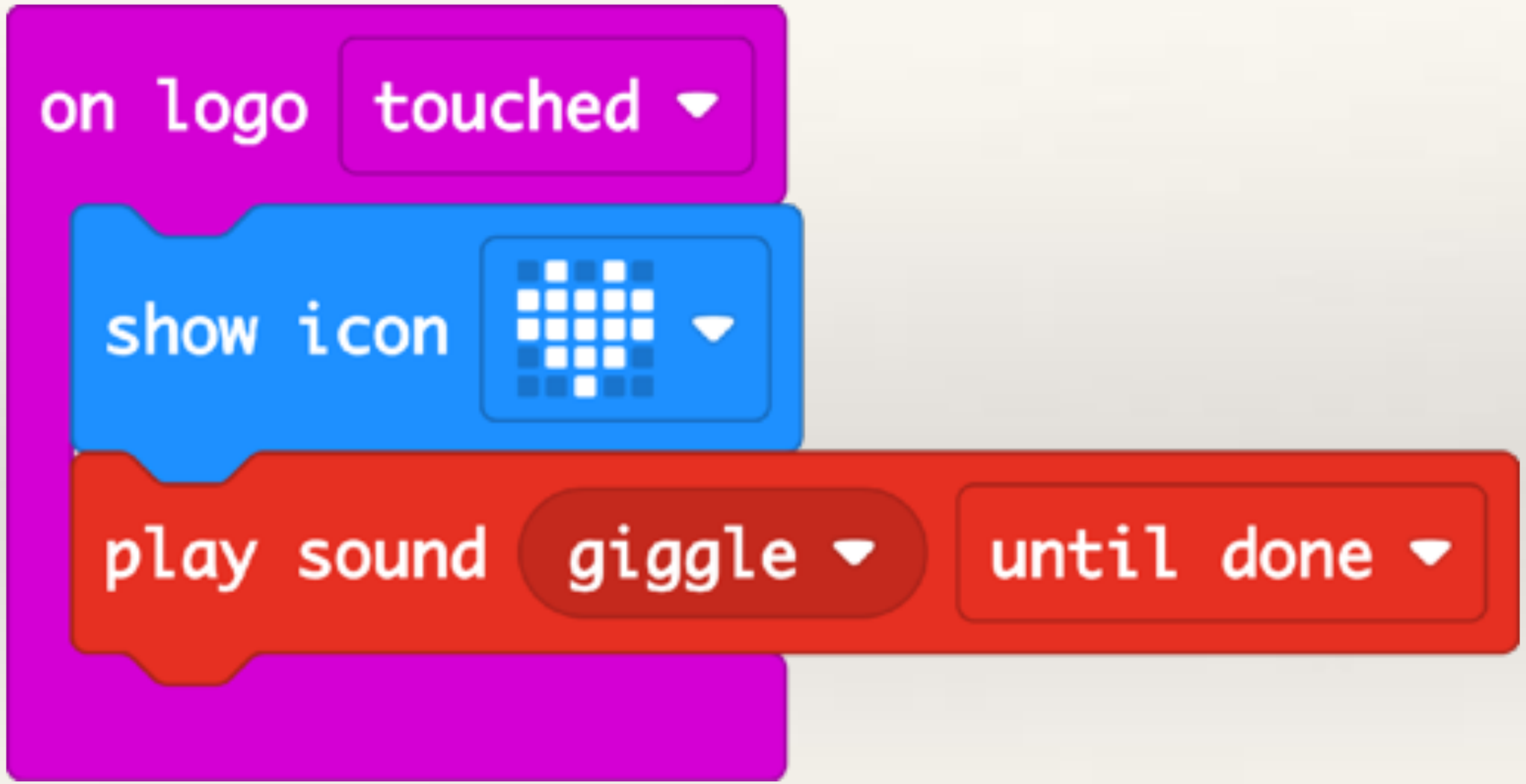
# Touch Sensor Giggle

**Objective:** help students understand how to use the touch sensor (logo).

**Problem:**

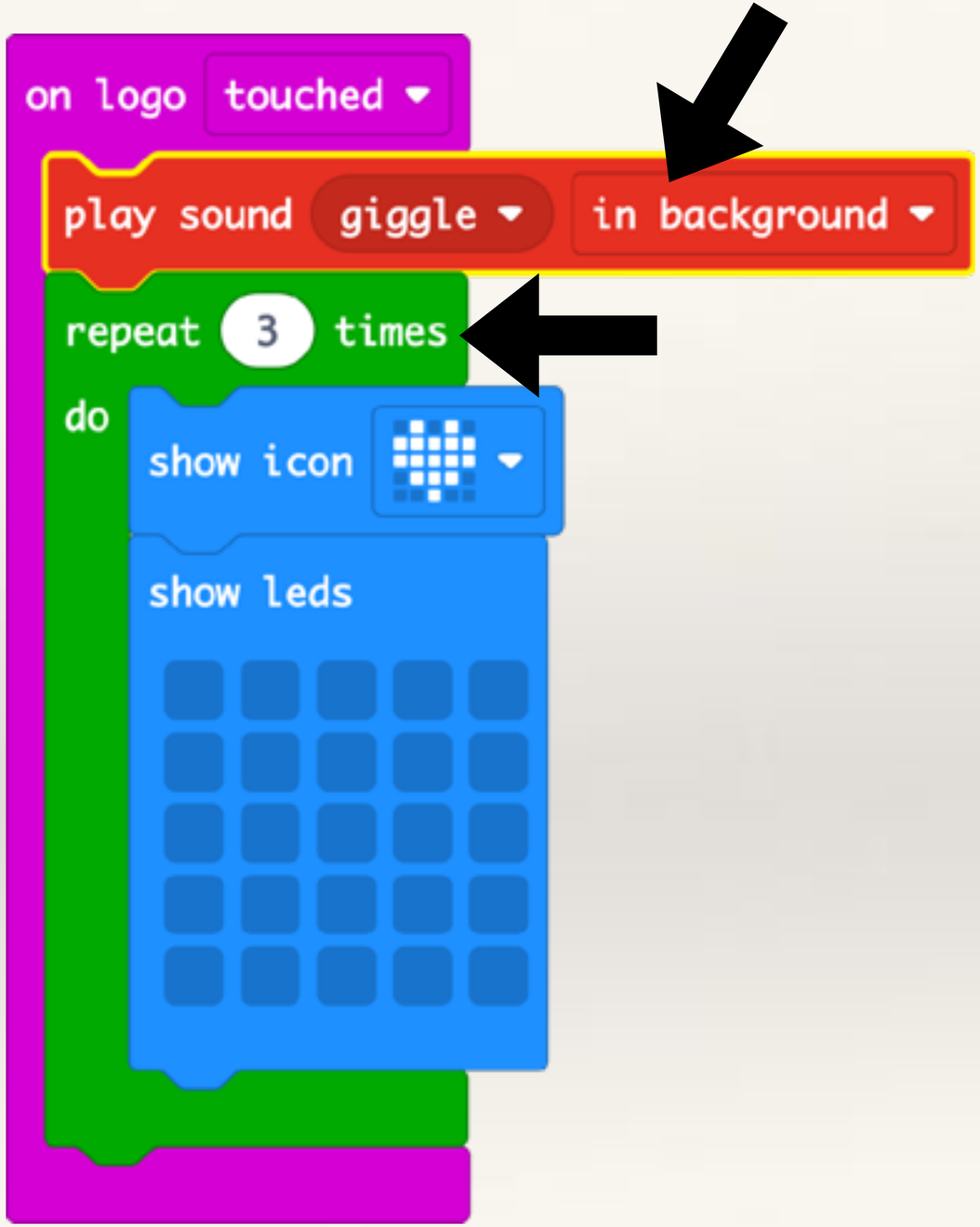
- Students should write code such that when the Micro:bit logo is ‘touched’ a heart icon gets displayed along with the sound effect ‘giggle’.
- Stretch the problem: when the Micro:bit logo is touched, the heart icon should flash 3 times and the sound ‘giggle’ should play in the background.

# Giggle Sound on Logo Touched



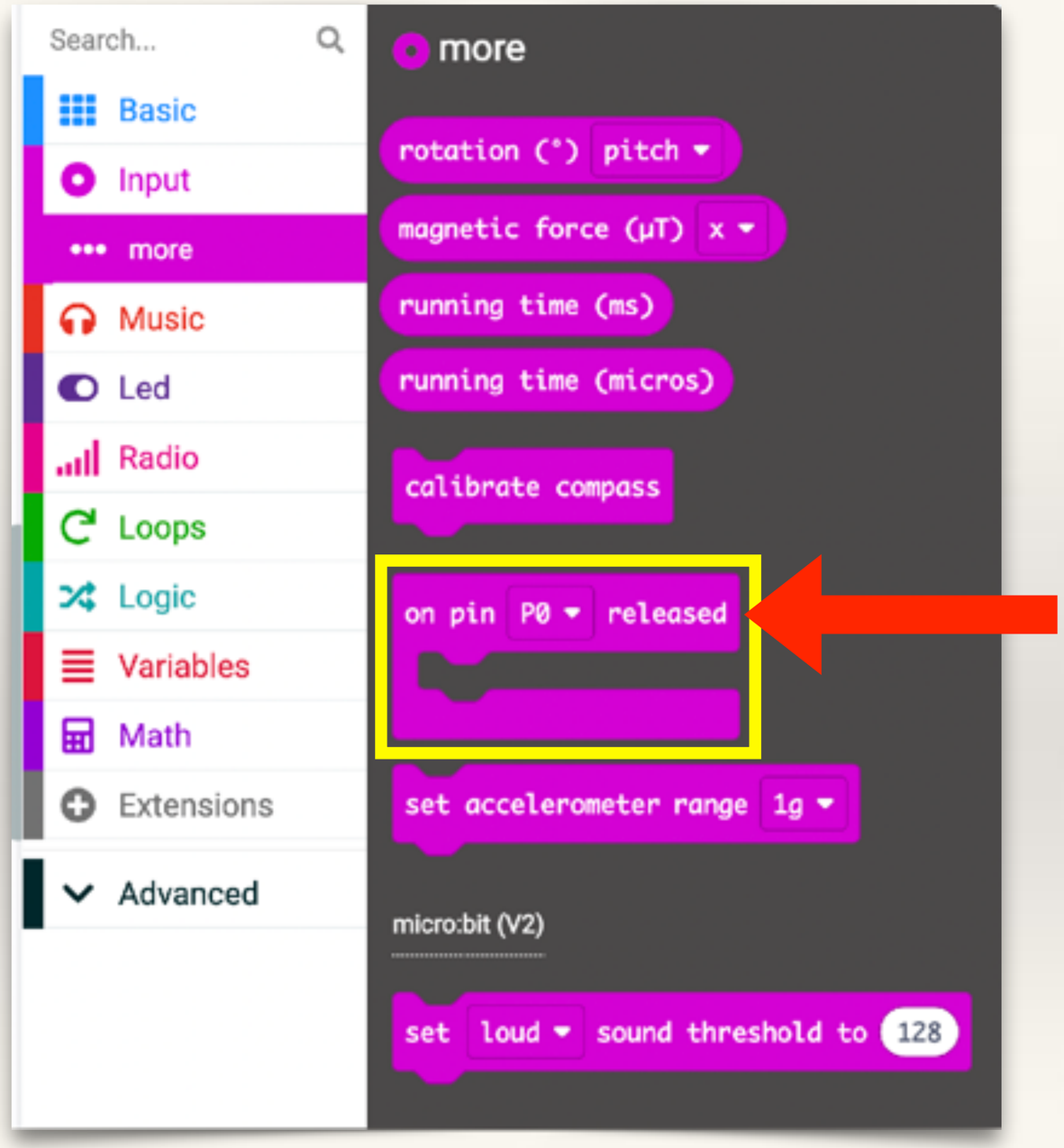
## Flashing Heart & Background Sound

1. In ‘Play Sound’ block, the drop-down for ‘until done’ shows the other option - ‘in background’
2. If you select ‘until done’ the complete sound will play and only then the next instruction will get executed.
3. If you select ‘in background’ the sound will start to play and the next instruction will get executed. This way, while the heart is flashing, the sound will also play in the background.
4. A ‘repeat’ loop can be used to make the heart icon flash.

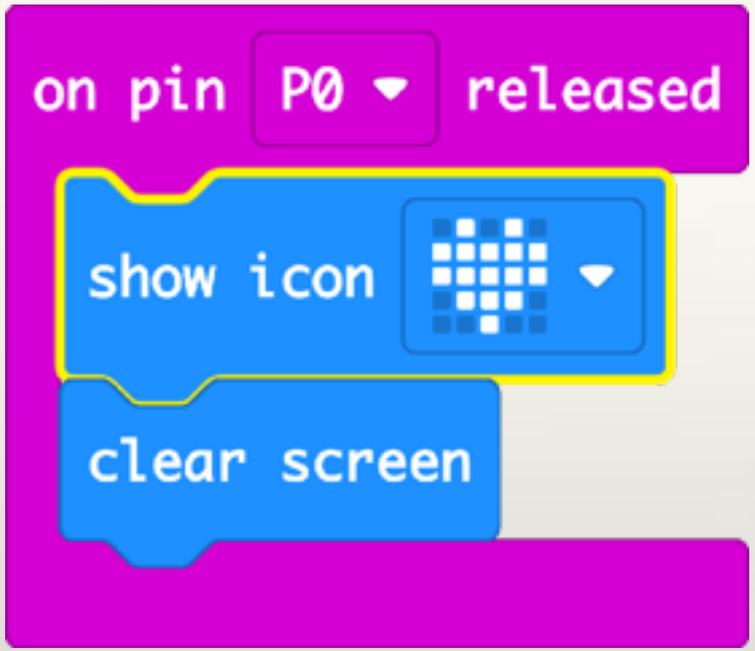


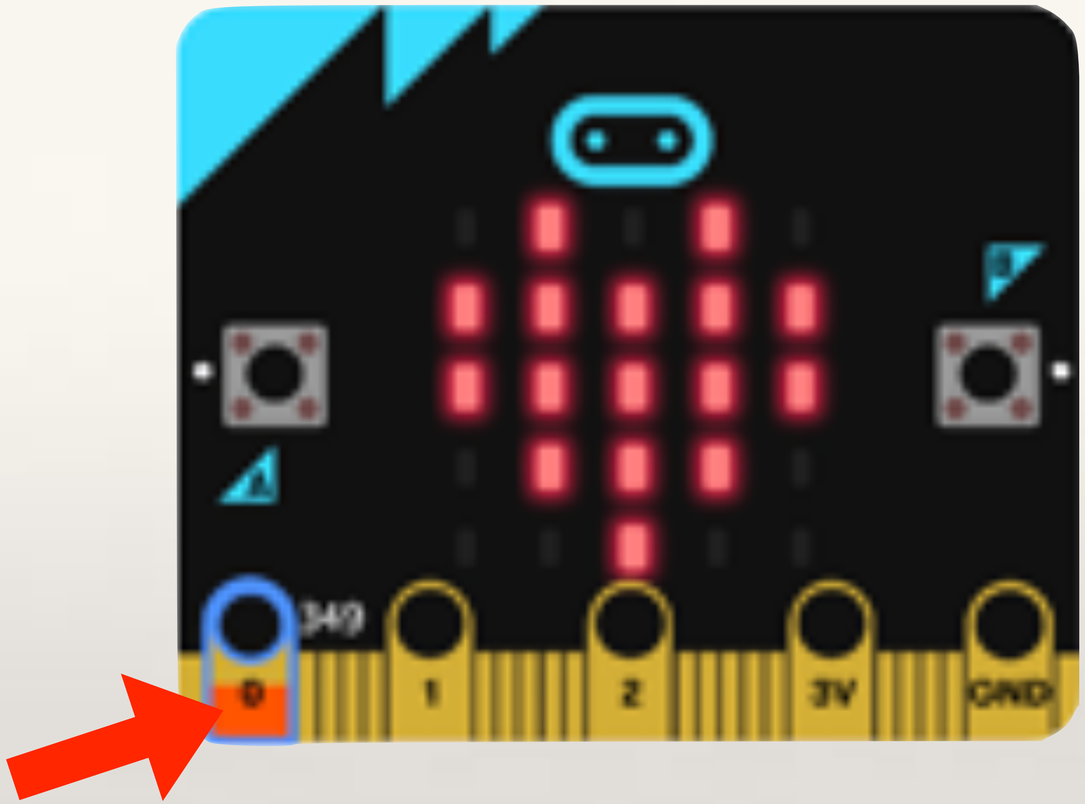
# Using Pins as Touch Sensor

1. Pins 0, 1 and 2 on the Micro:bit can also be used as a touch sensor
2. If you hold the GND pin with one hand and touch pin 0, 1, or 2 with the other, a very small (safe) amount of electricity will flow through your body and back into the Micro:bit and a circuit will be complete.
3. This feature can be used to make Pins 0, 1, and 2 behave like a touch sensor
4. This works best if you use batteries instead of the USB to power the Micro:bit.



## Pin-0 as Touch Sensor



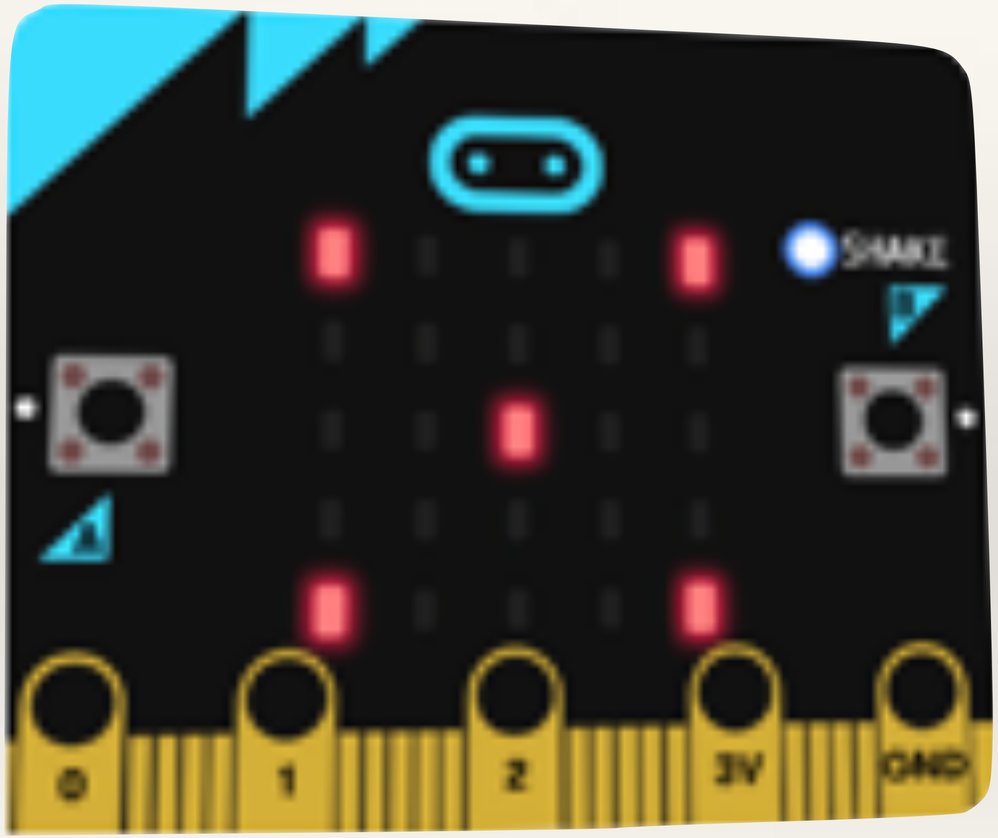




1. Students can write a simple programme to test Pins being used as a touch sensor
2. In the simulator, the selected pin (0, 1, or 2) will need to be touched and released to test the code.

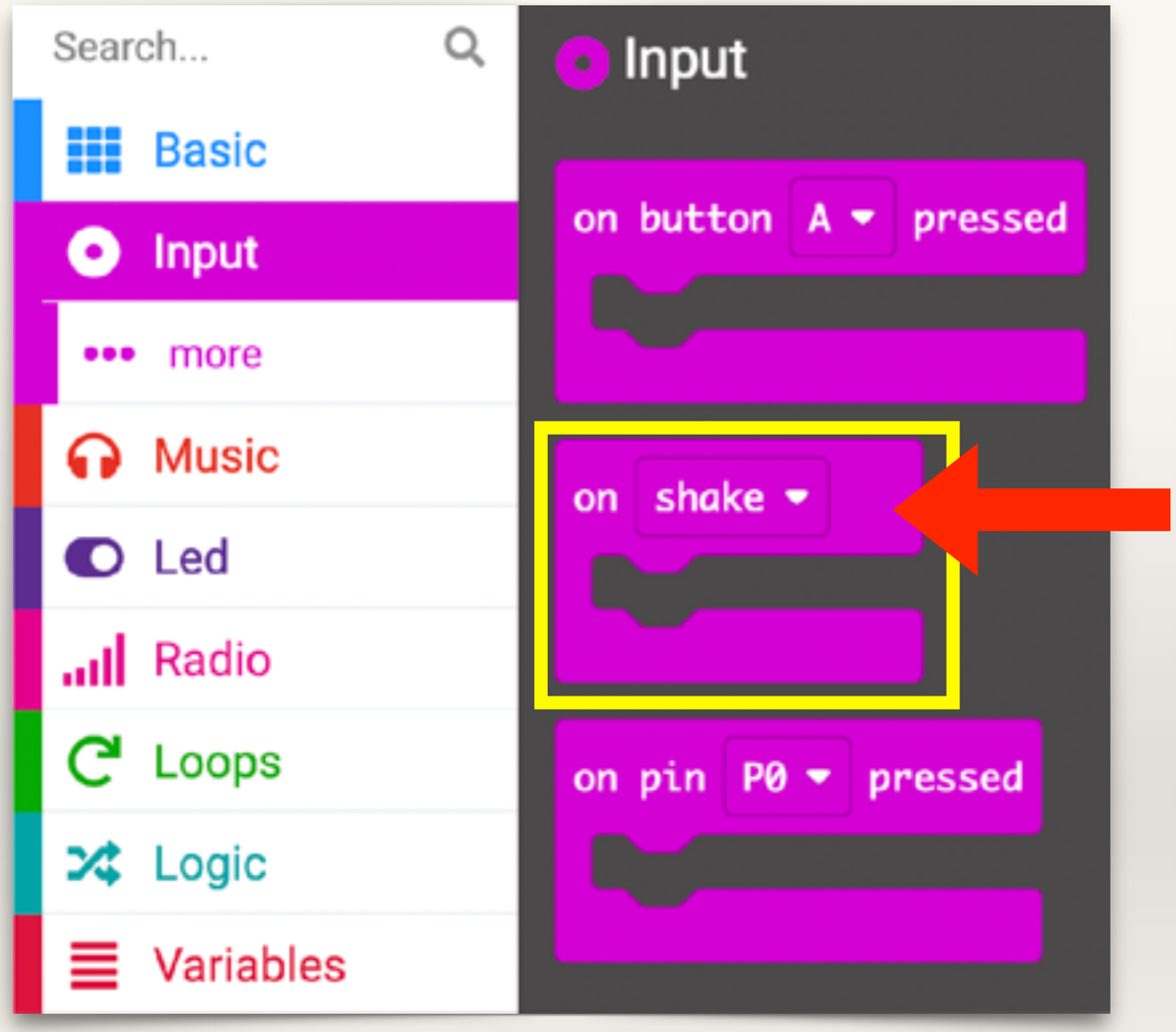
On-board Sensors

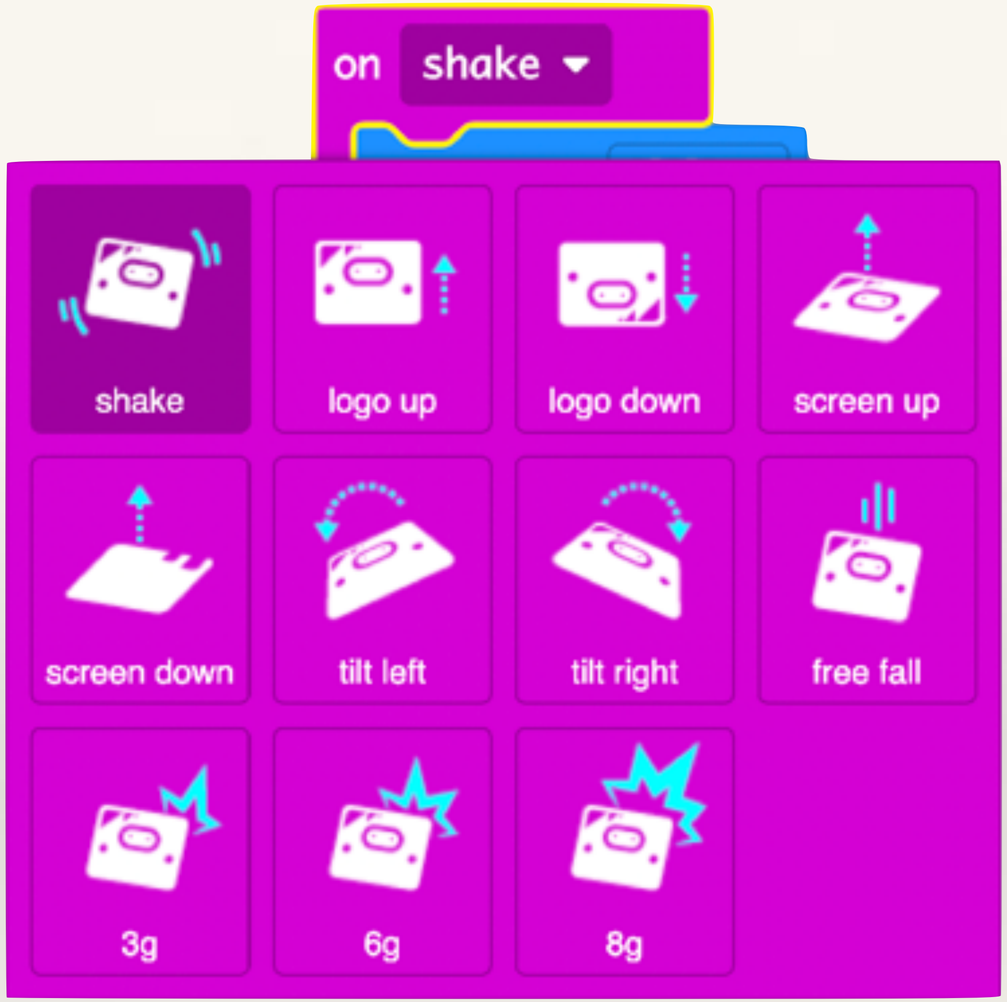
# ACCELEROMETER

# ACCELEROMETER SENSOR




- The accelerometer sensor measures change in motion (acceleration) in X, Y and Z axis.
- X-axis is left-right motion, Y-axis is forward-backward motion, and Z-axis is up-down motion.
- Motion in any of the axis is also called a gesture like on-shake, logo up, tilt right gestures.




1. Accelerometer sensor is in Input > On-Shake
2. Micro:bit can sense many gestures like shake, tilt, free fall, logo up or down

Project-3

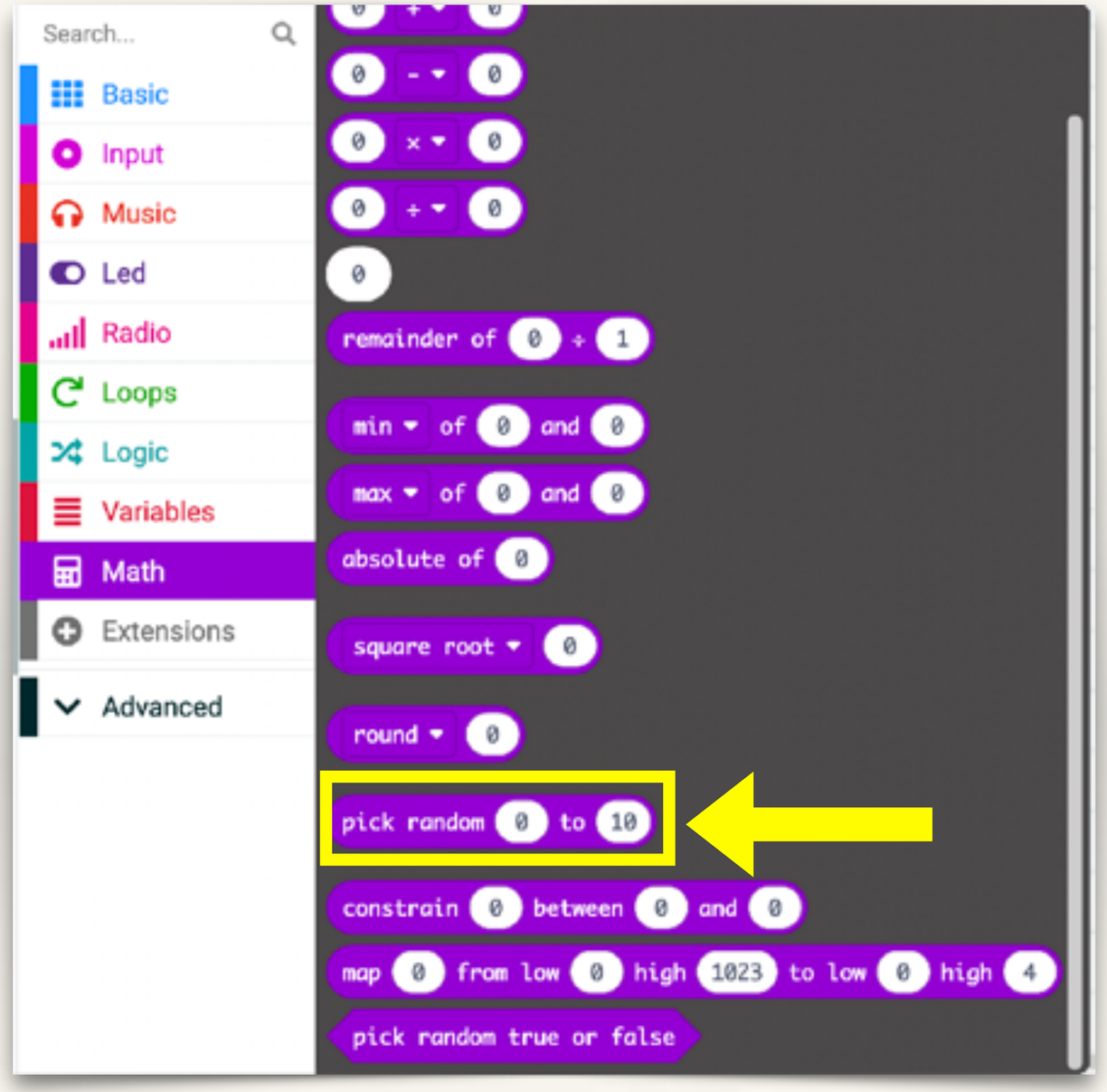
# Electronic Dice

**Objective:** use the accelerometer sensor (on-shake) to create an electronic dice.

**Problem:**

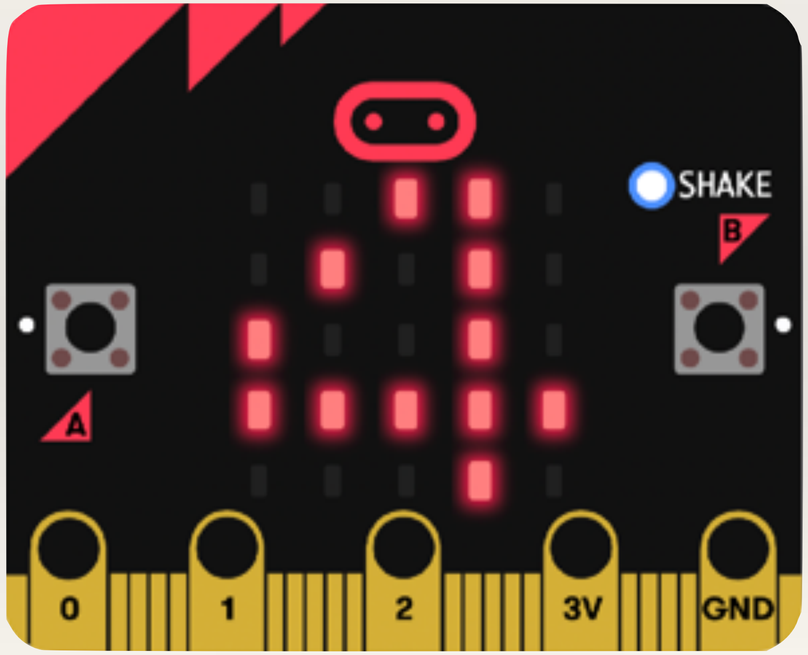
- Students should write code such that each time the Micro:bit is shaken, it generates a random number between 1 and 6 (like a dice).

To generate a random number between 1 and 6, students will need to use the ‘Pick Random Number’ block, which is available under the ‘Math' blocks



# Electronic Dice


Every time the Micro:bit is shaken, it will generate a random number between 1 and 6 and display that number



Project-4

# Stay Fit Step Counter

# FitBit




Electronic devices like the Fitbit and health apps use the accelerometer sensor to tell us how much exercise we are doing. In this project, we will use the Micro:bit to make a Step Counter.

**Objective:** use the accelerometer sensor (on-shake) to create a Step Counter.

**Problem:**

- If we walk with a Micro:bit tied to our shoe, on every step the Micro:bit will get shaken. Students need to use this property to create a Step Counter that will keep track of the number of steps taken and on pressing button-A it should show the total number of steps taken.
- Students will need a basic understanding of Variables.

In computer programming, a **Variable** is simply a location in the computer memory where we store value of something that keeps changing when the programme runs. For example, Score in a game is a variable because it keeps changing. Whereas Player Name is a constant because it does not change when a programme is executed.

# **Micro:bit**
Step Counter

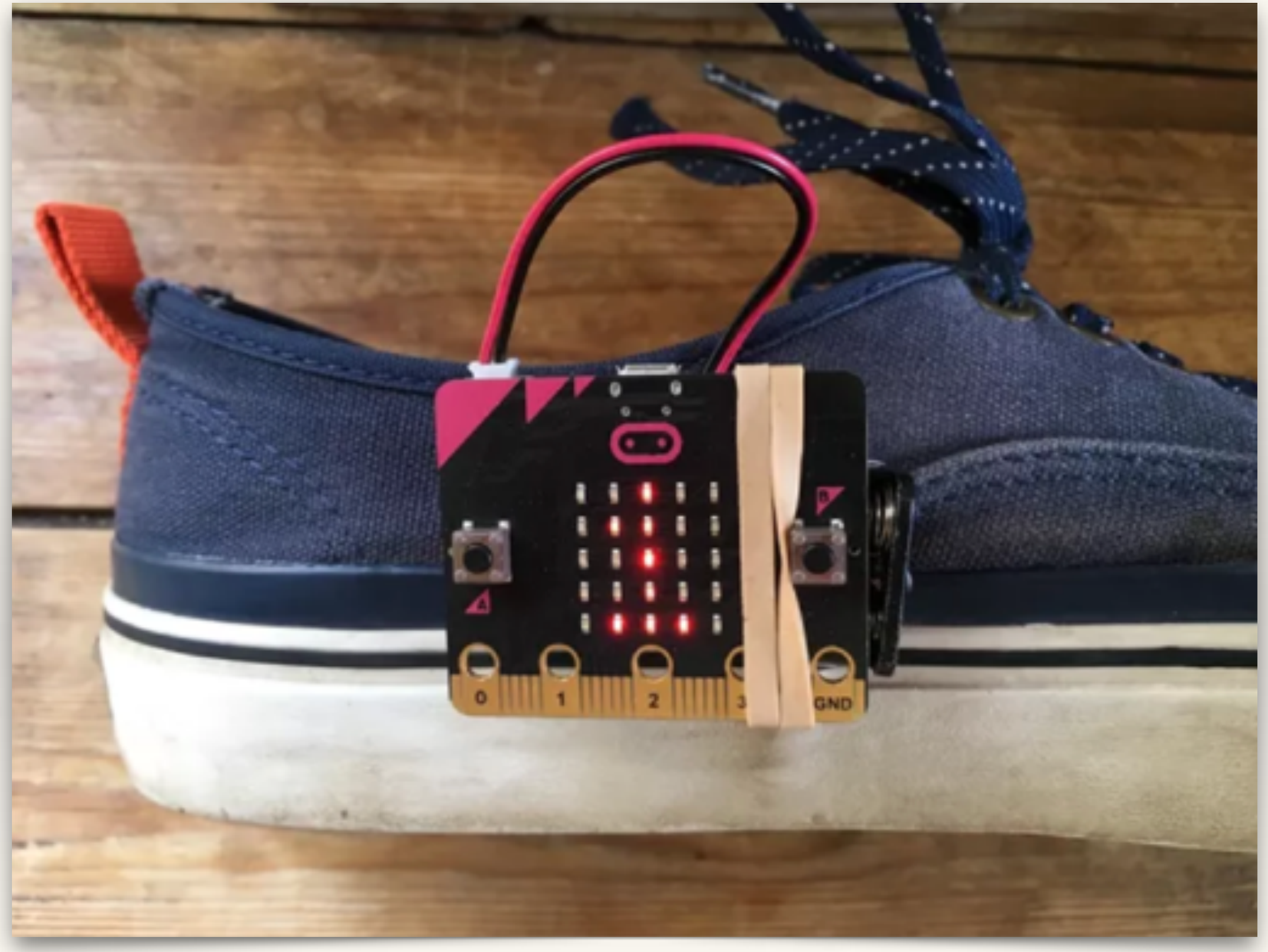
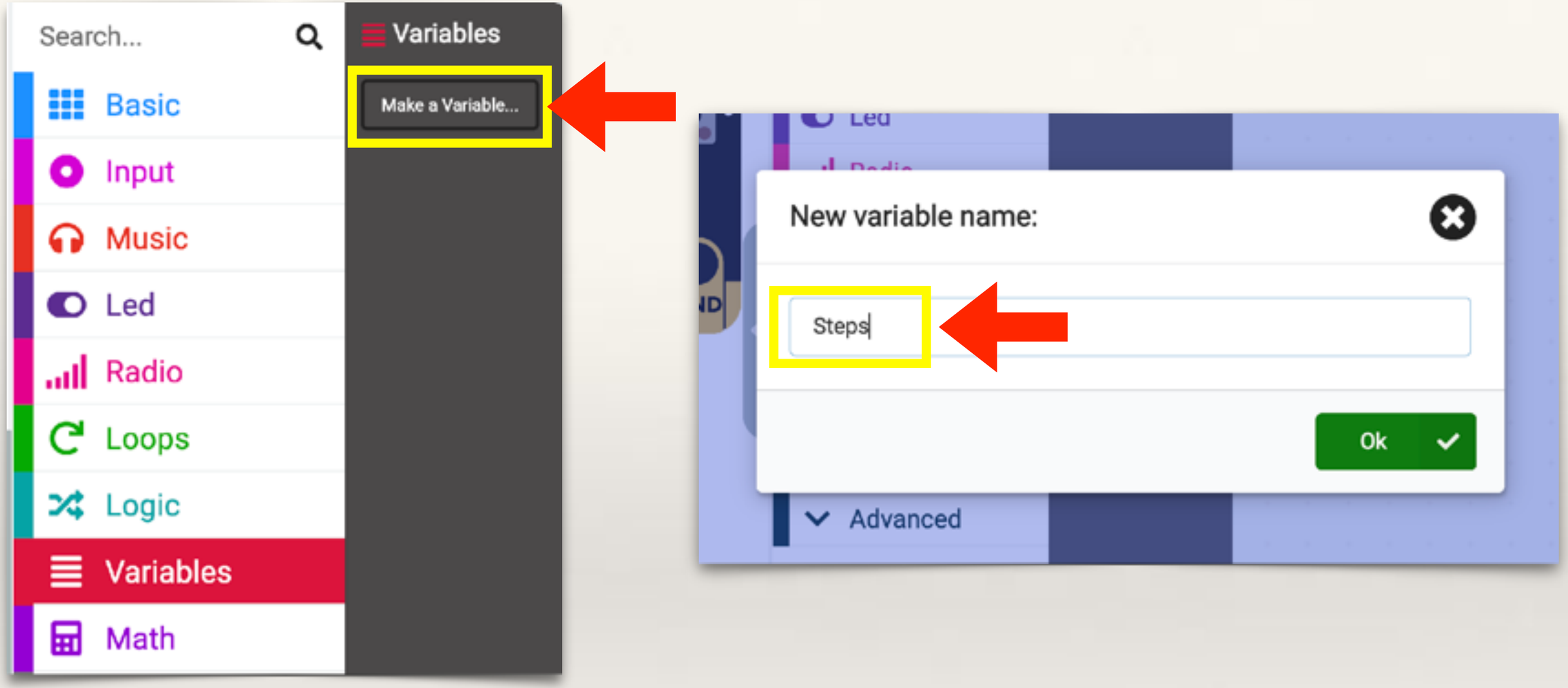


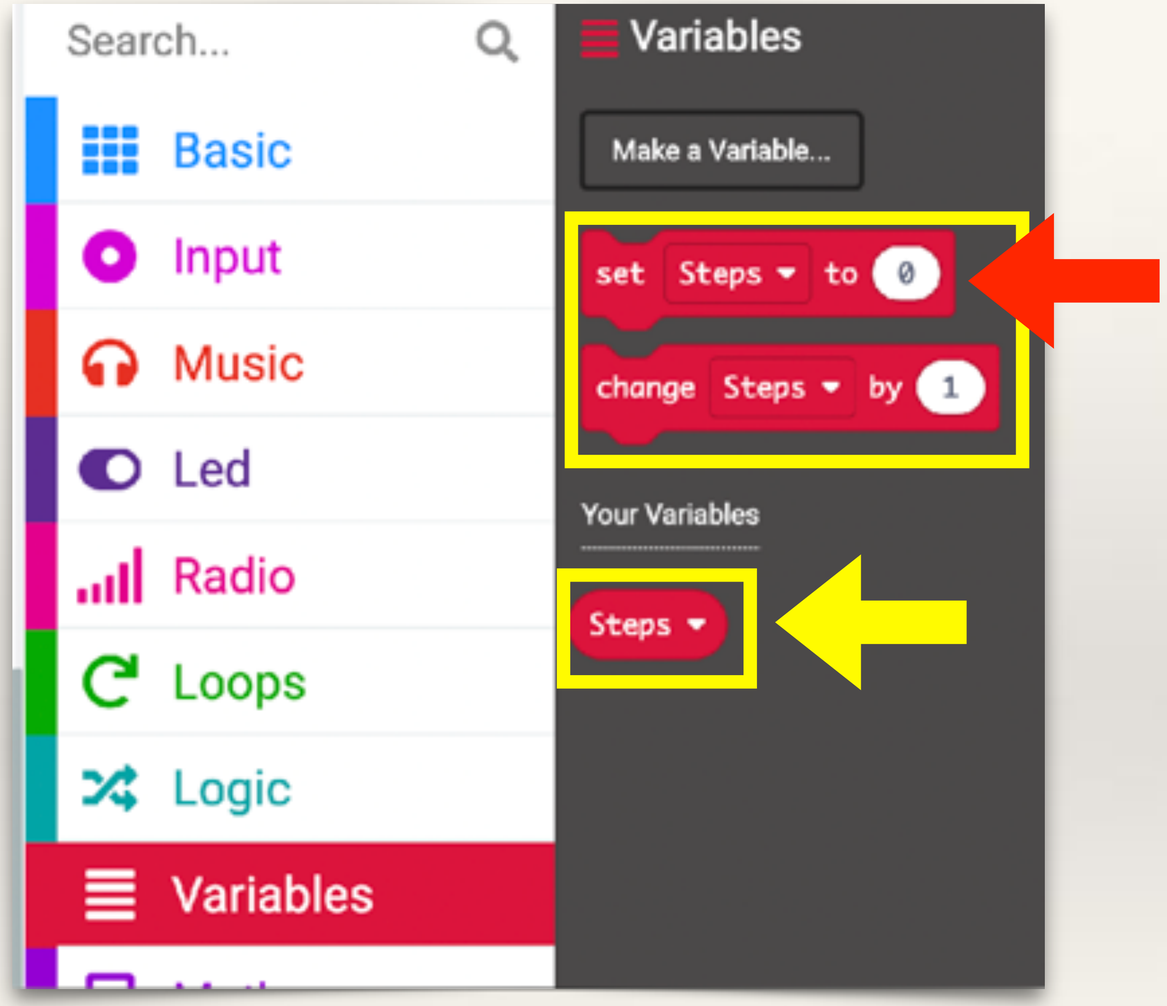

Image Source: https://micro:bit.org/projects/make-it-code-it/sensitive-step-counter/




To keep track of the number of steps taken (on each shake event), we will need to create a variable. Go to Variables > click Make a Variable > Type an appropriate variable name e.g. Steps

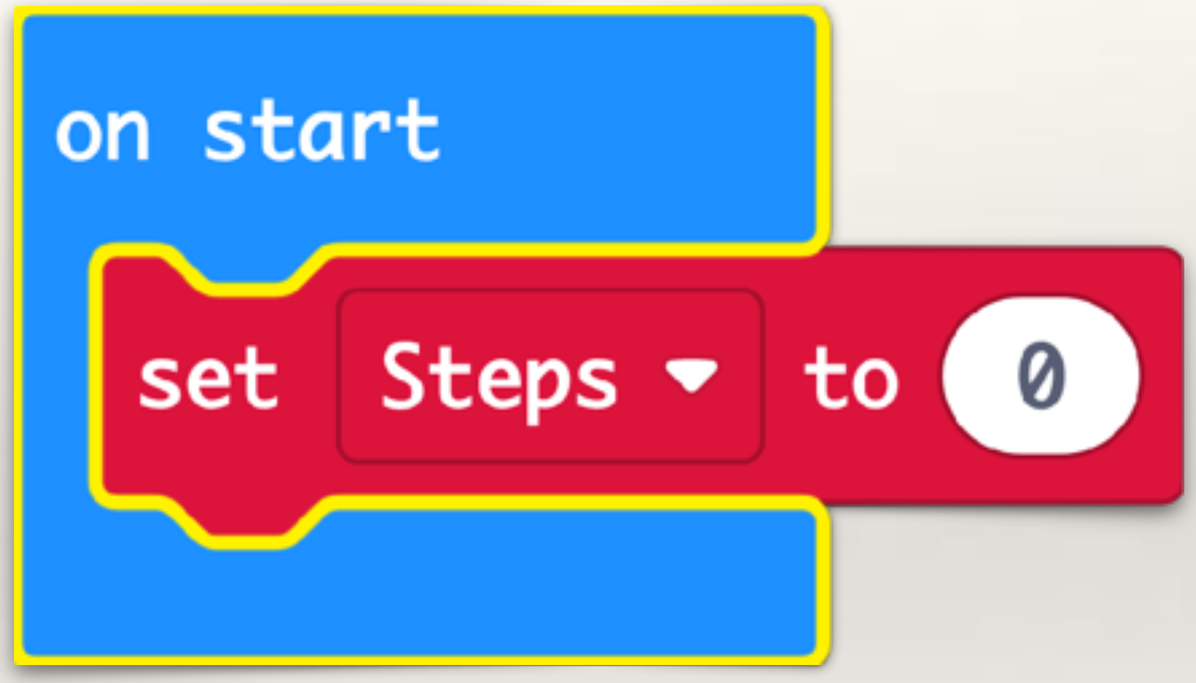
Once a new variable is created, new blocks will appear under ‘Variables’ - Set, Change, and the new variable (in this case ‘Steps’).

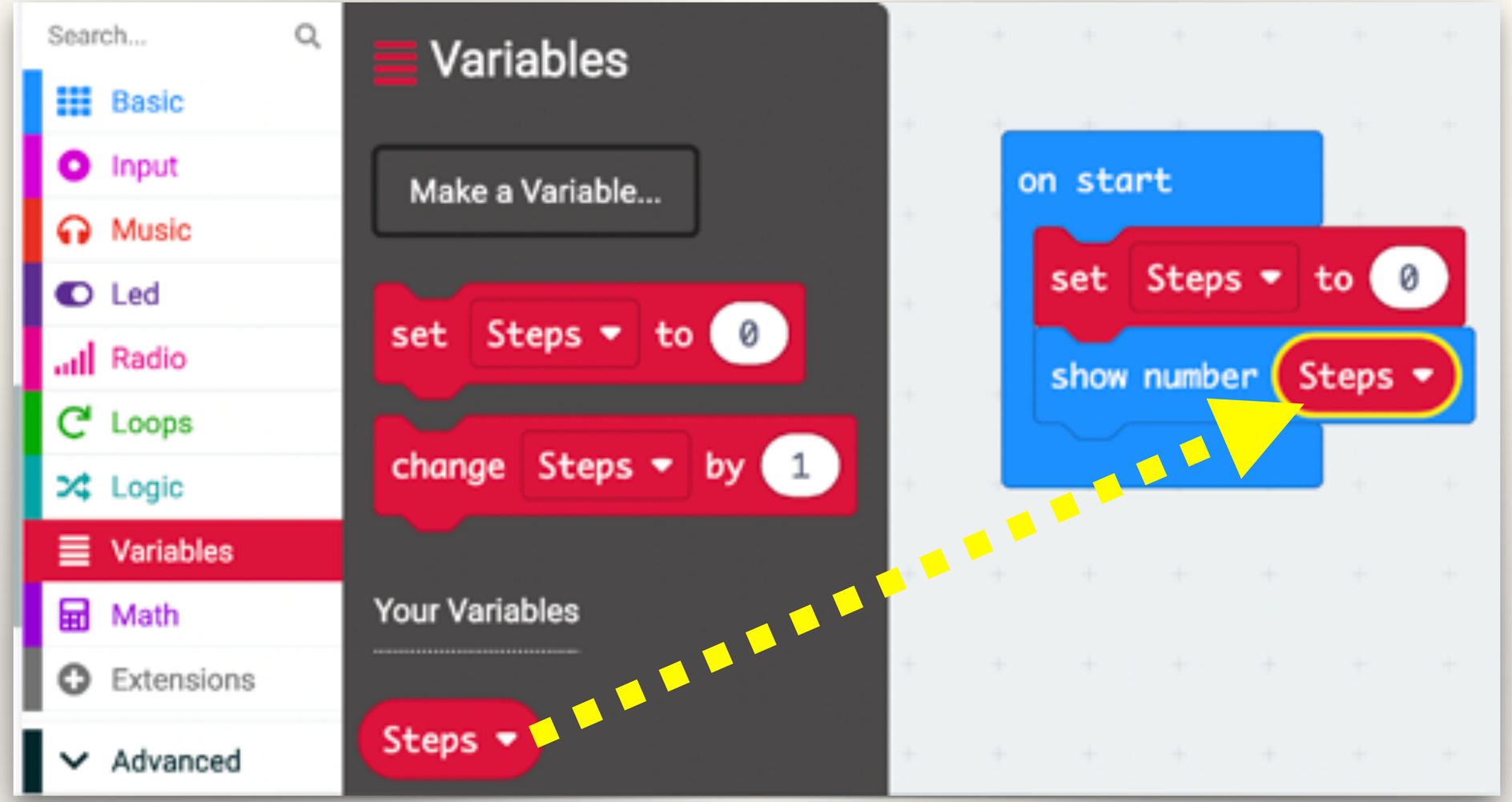
Drag out the ‘Set Number’ block and ‘Steps’ variable.



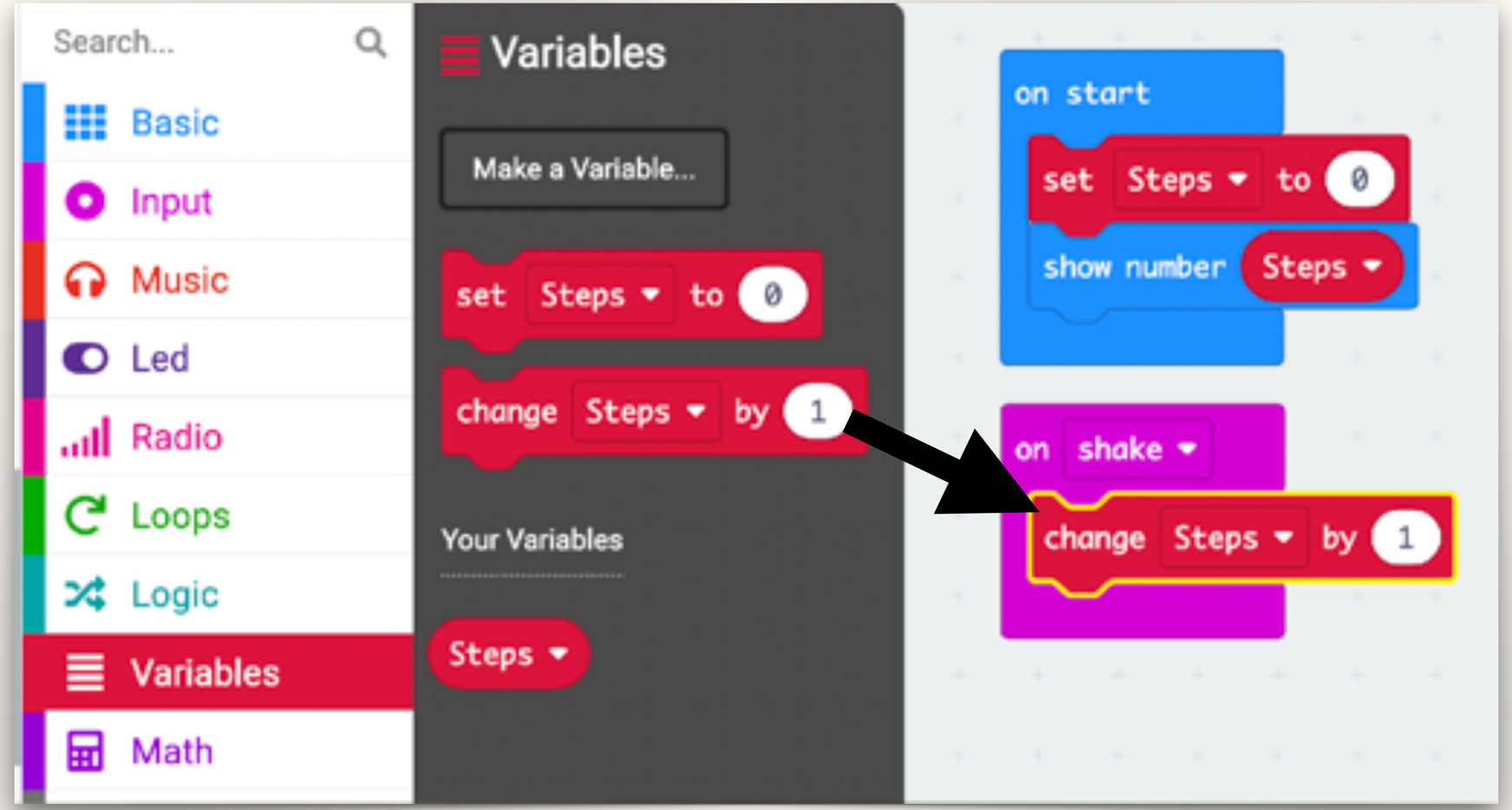
From ‘Basic’ drag out an ‘On Start’ block and put the ‘Set Steps’ command inside ‘On Start’ block.

This will ensure that whenever our programme restarts, the value of the variable called ‘Steps’ is set to 0.






From ‘Basic’ drag out ‘Show Number’ block and put the ‘Steps’ variable inside ‘Show Number’ block. This will confirm to the user that whenever the programme starts or restarts, the value of the variable called ‘Steps’ is set to 0.




From ‘Input’ drag out an ‘On Shake’ event block and put the ‘Change Steps’ command inside ‘On Shake’. Set the value to 1. Now, whenever the micro:bit gets shaken, the value of the variable called Steps will get incremented by 1.

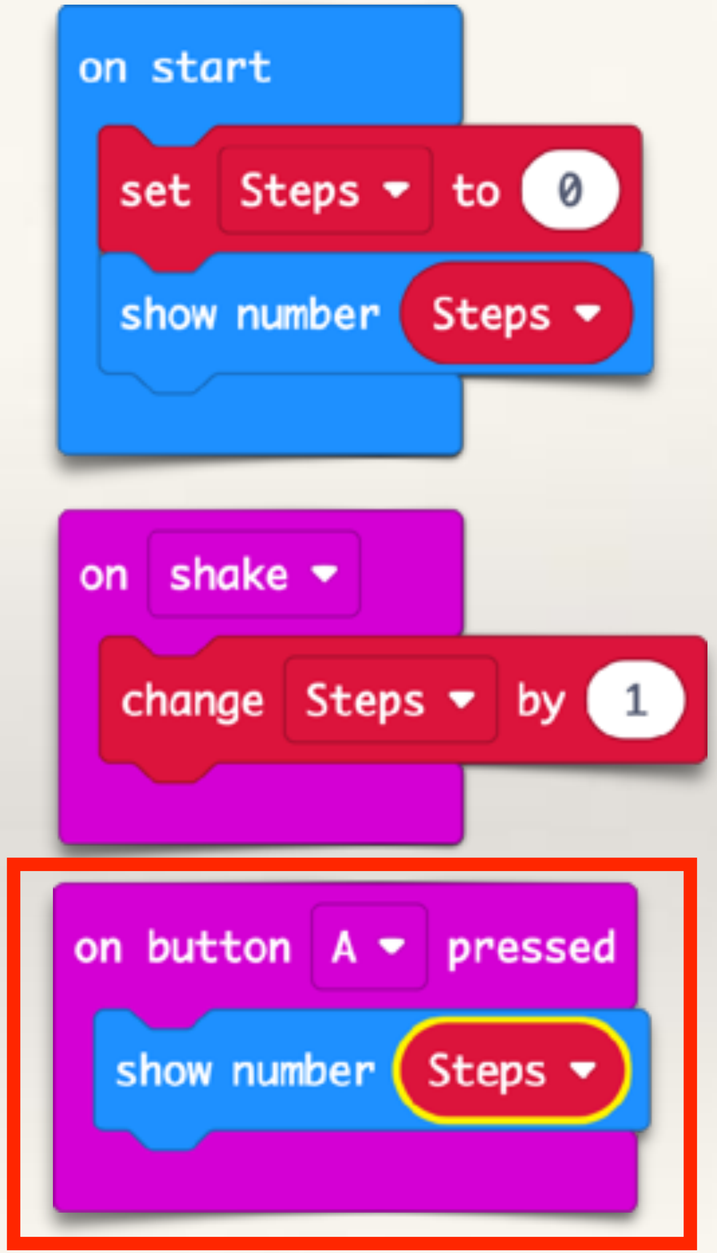
From ‘Input’ drag out an ‘On button-A Pressed’ event block.

From ‘Basic’ drag out a ‘Show Number’ block.

Put the variable ‘Steps’ inside  ‘Show Number’ command.

Now, after the micro:bit has been shaken a few times, what is the total value of the variable Steps can be known by pressing button-A.

Our basic Micro:bit Step Counter is ready. But we can make it look better.

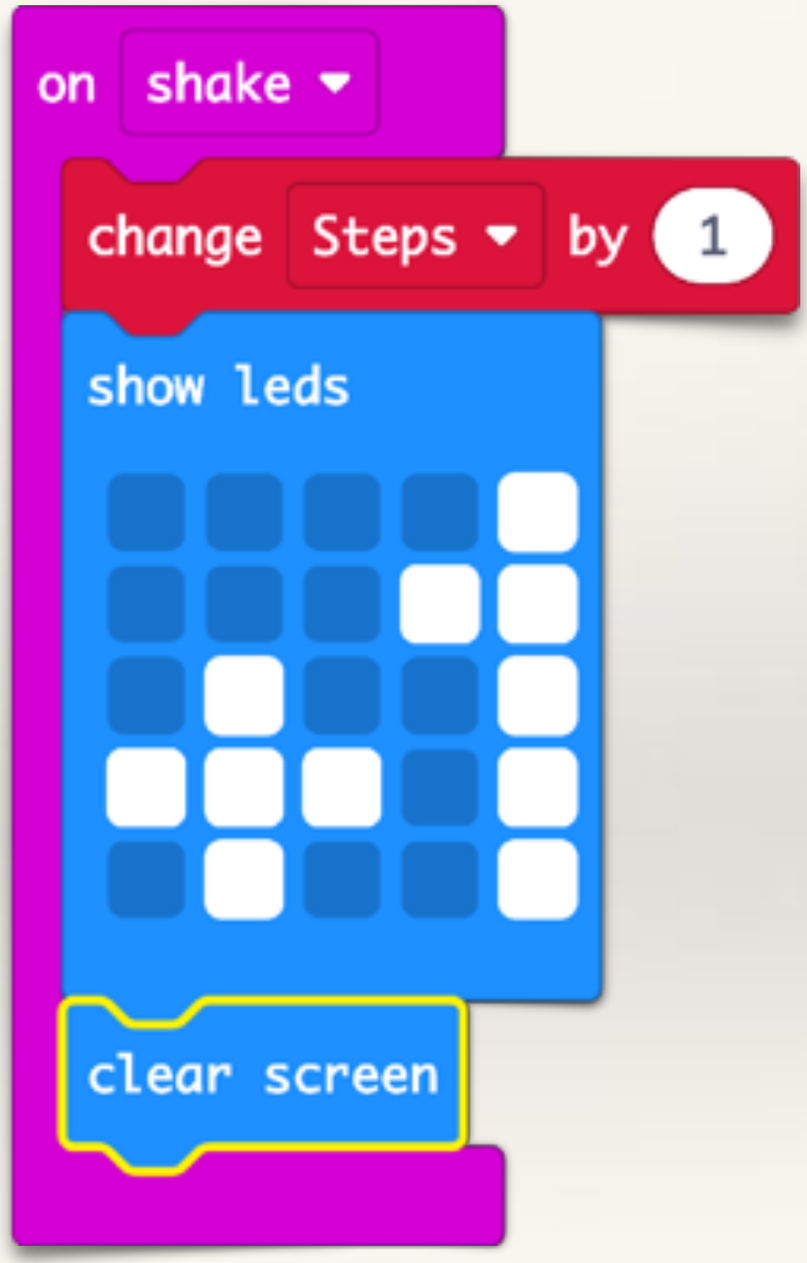


From ‘Basic’ drag out a ‘Show LEDs’ command. Put it in the On Shake block, under ‘Change Steps’ command.

Light up the appropriate LEDs to display “+1”.

From ‘Basic’ drag out a ‘Clear Screen’ command. Put this below the ‘Show LEDs’ command.

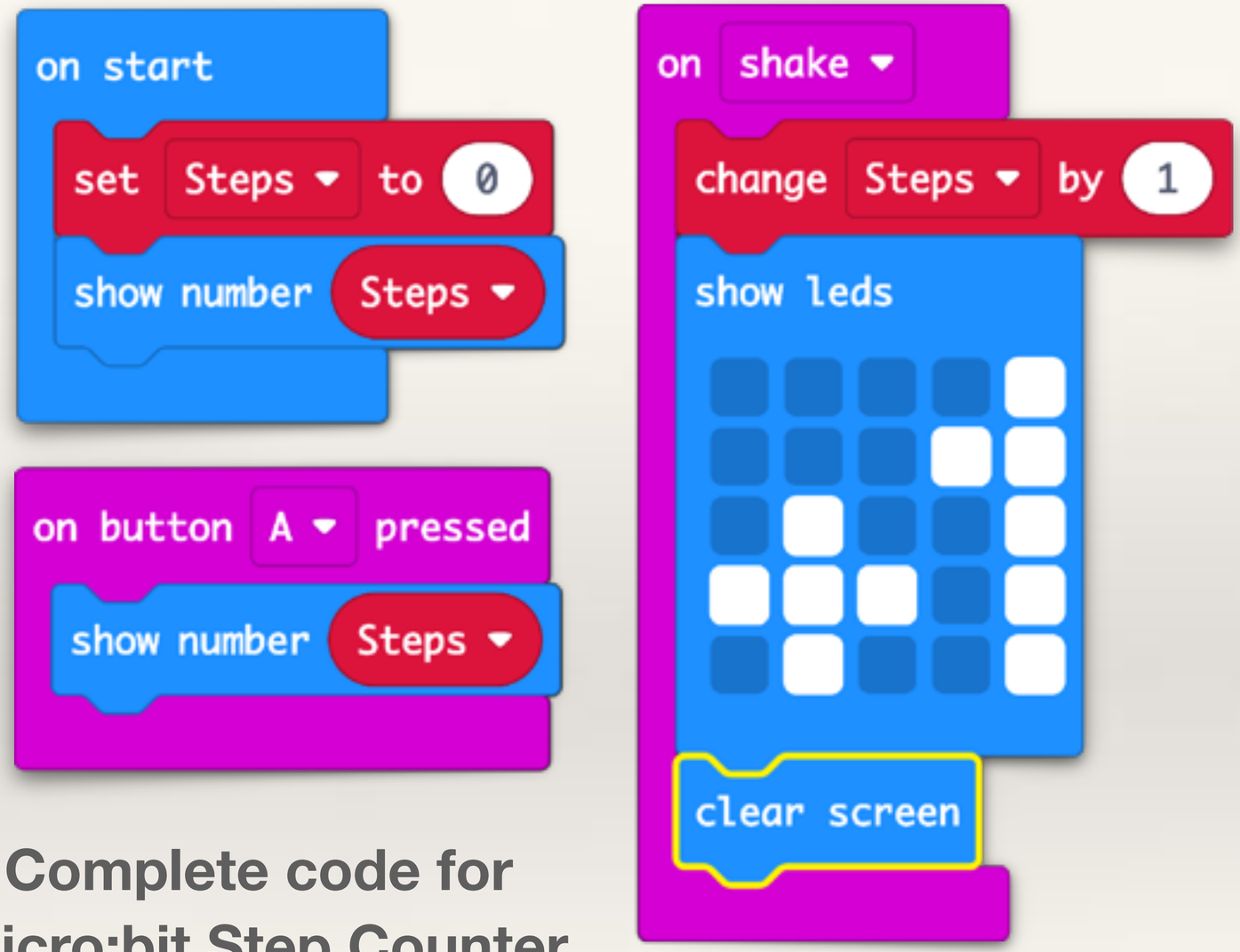
Now, each time the micro:bit detects a shake, it will display “+1” briefly.



Attach the 3 volt battery-pack to the micro:bit and put both of them inside your socks.

Whenever you take a step, the micro:bit will detect a shape and increment the variable called Steps by 1.

You can see the total number of steps you have taken by pressing button-A




**Complete code for Micro:bit Step Counter**

Project-5

# Musical Dice

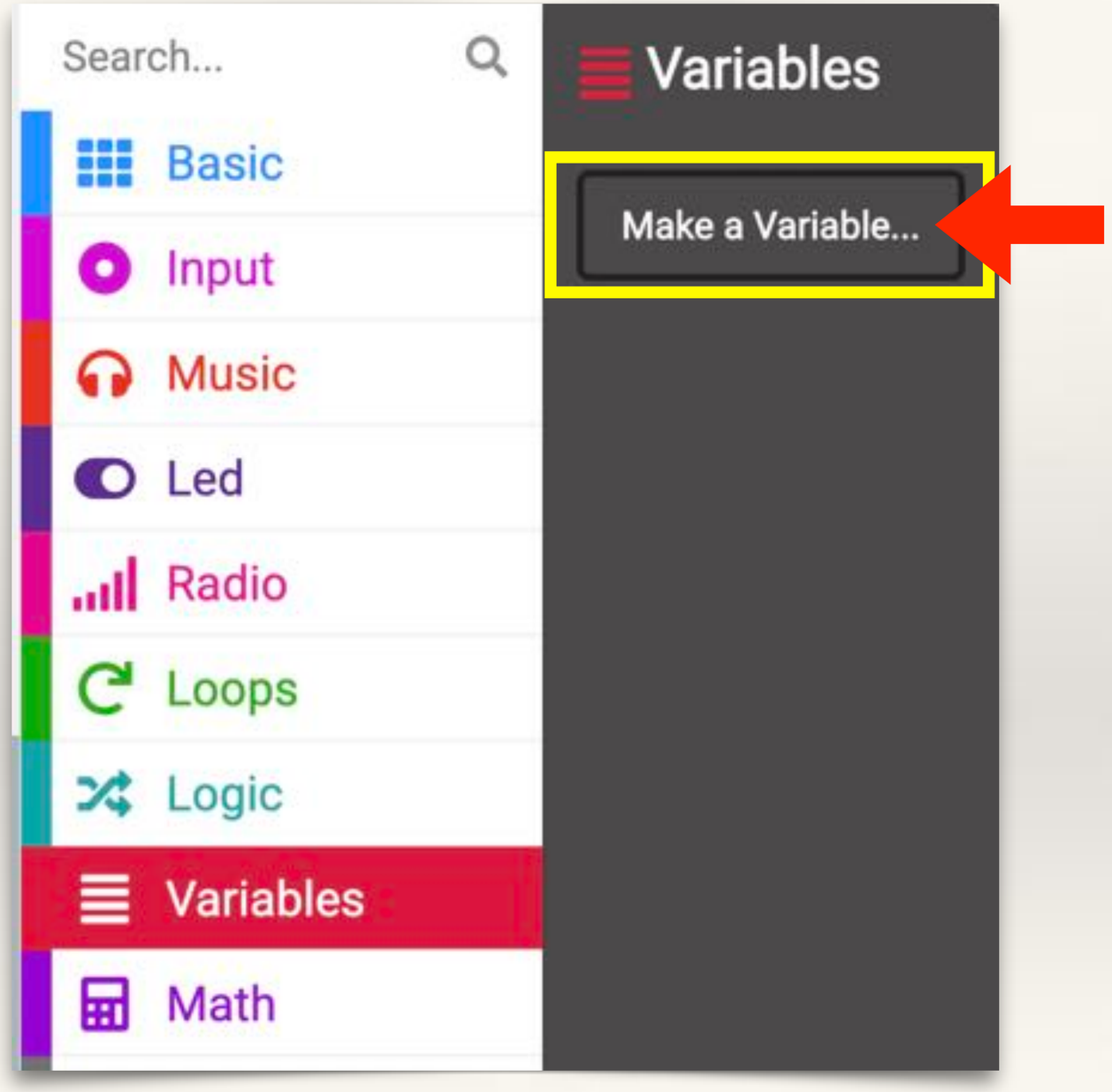
**Objective:** use the accelerometer sensor (on-shake) to create a musical dice. This project requires basic knowledge of variables and If-Then conditional statements.

**Problem:**

- Stretch the electronic dice project!
- Instead of showing numbers, on-shake the Micro:bit should display equivalent number of LEDs (1 = 1 LED, 6 = 6 LEDs) and also equivalent number of short beeps should sound.

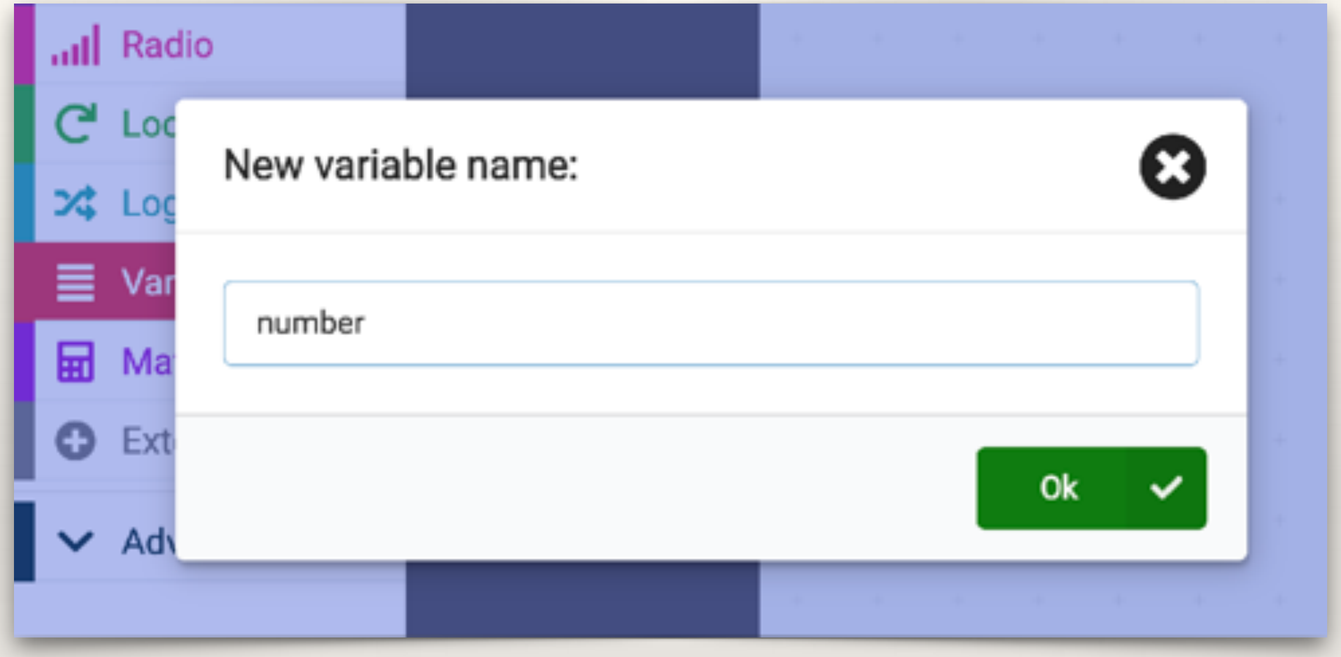
## STEP-1

In order to display the right number of LEDs (equivalent to the random number generated) students will need to create a variable which is under Variables > Make a Variable



## STEP-2

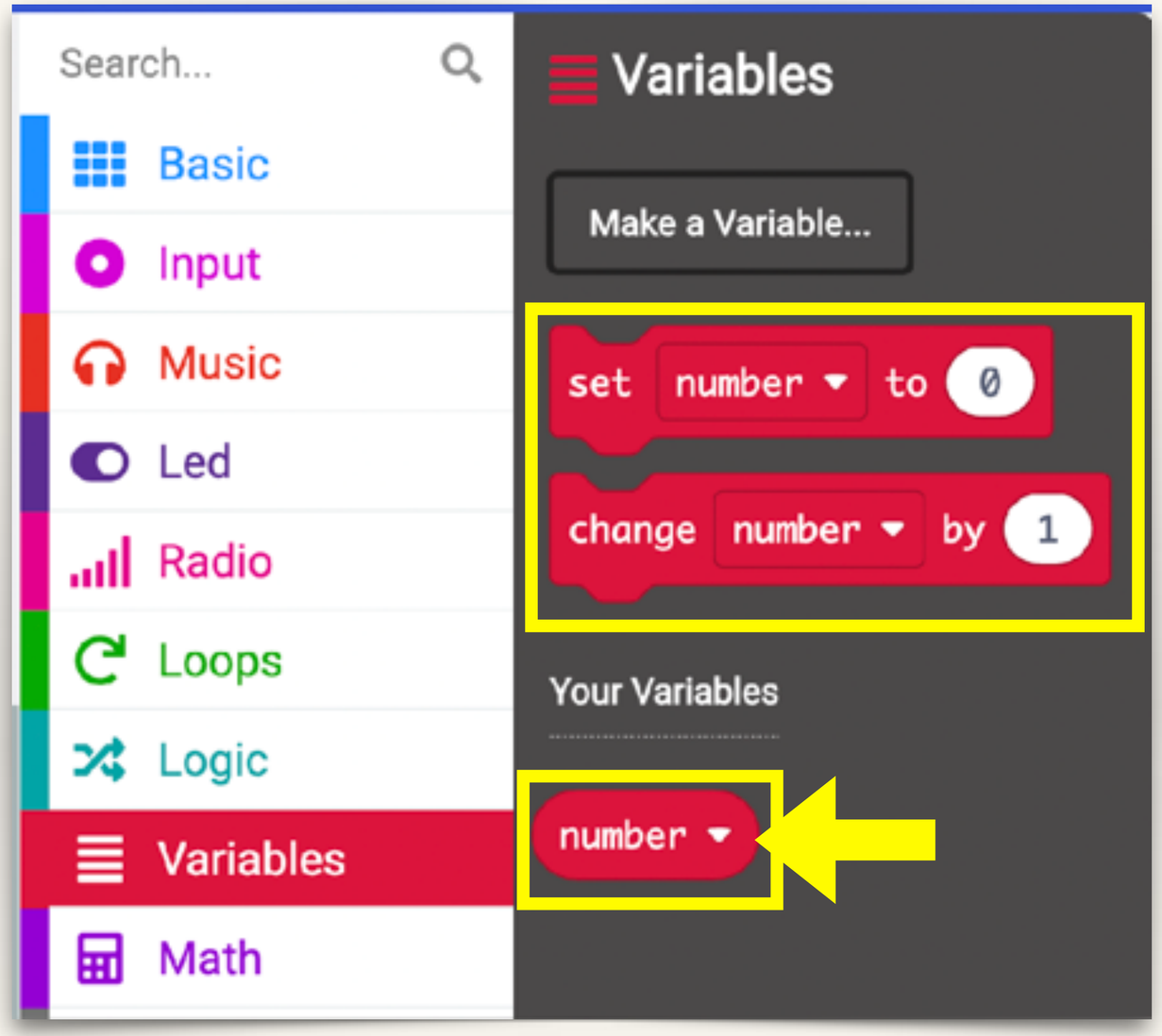
Give the new variable an appropriate name like number, or random number

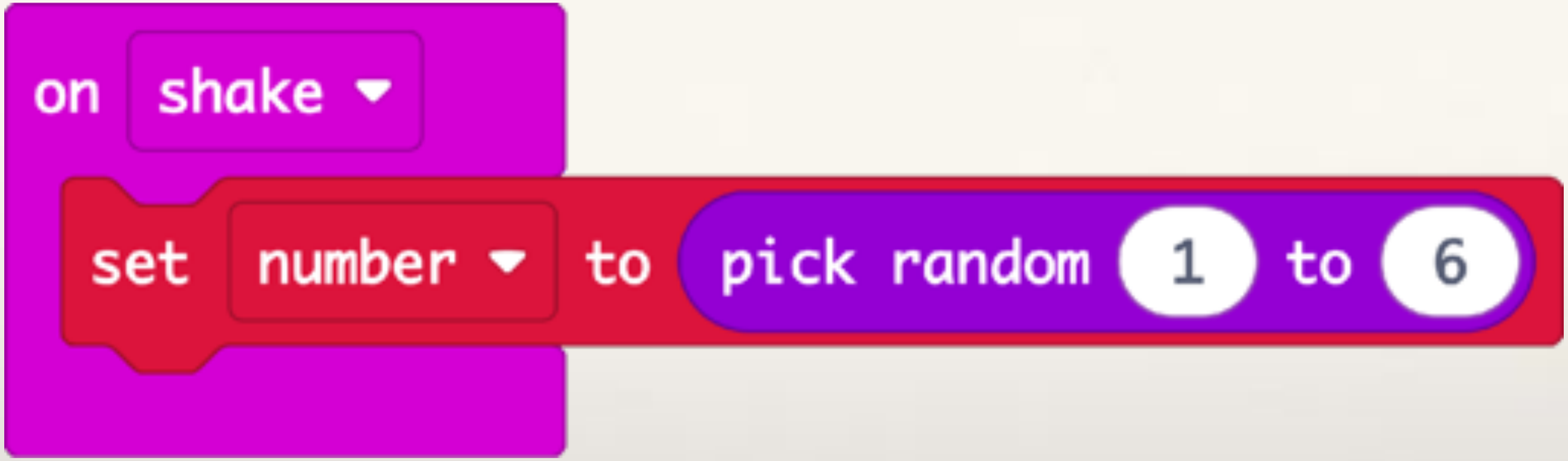


Once a new variable is created, new blocks will appear under ‘Variables’ - Set, Change, and the new variable (in this case ‘Number’).

**STEP-3**

Drag out the Set Number block and Number variable.





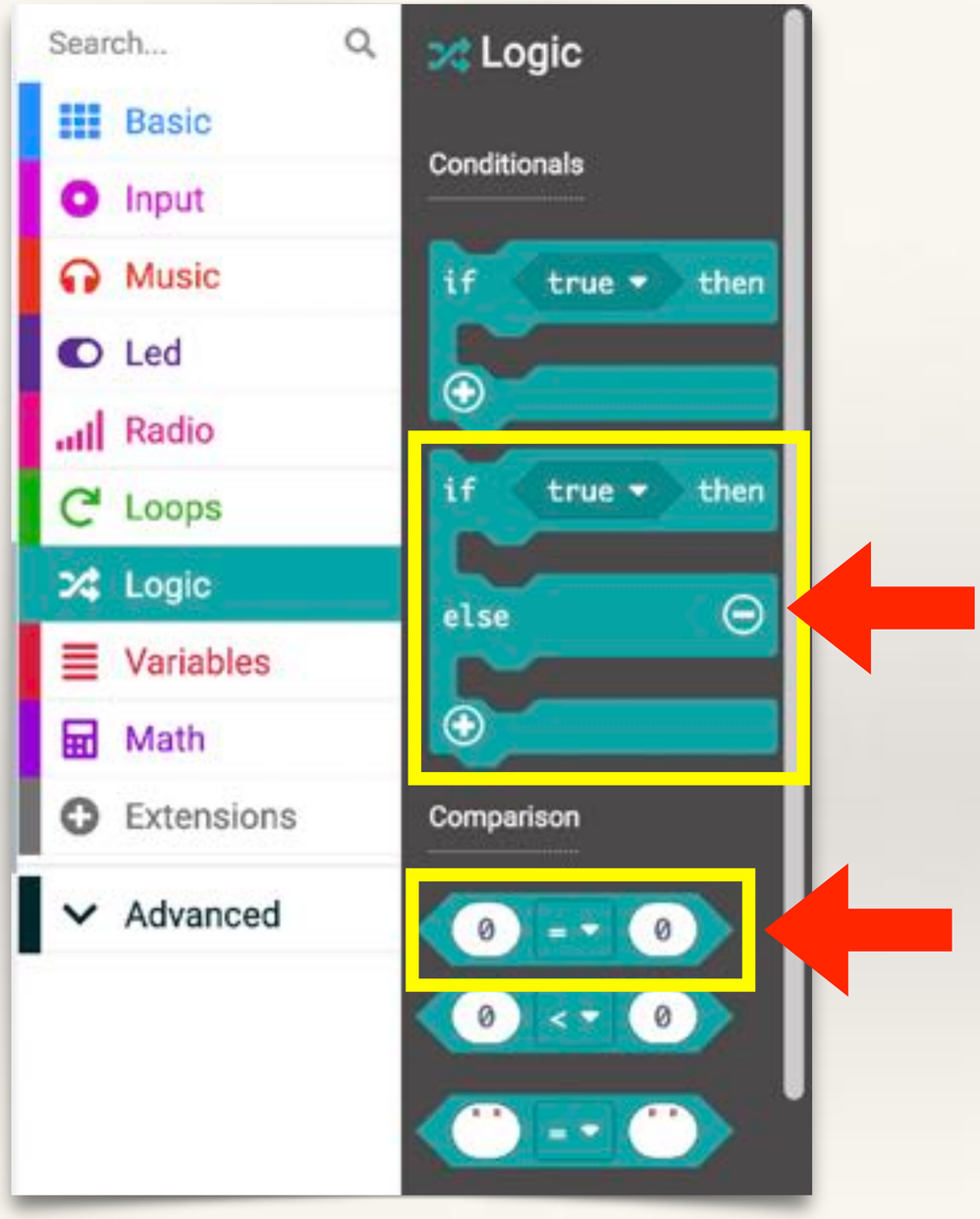

Put the “pick random 1 and 6” block inside Set Number block

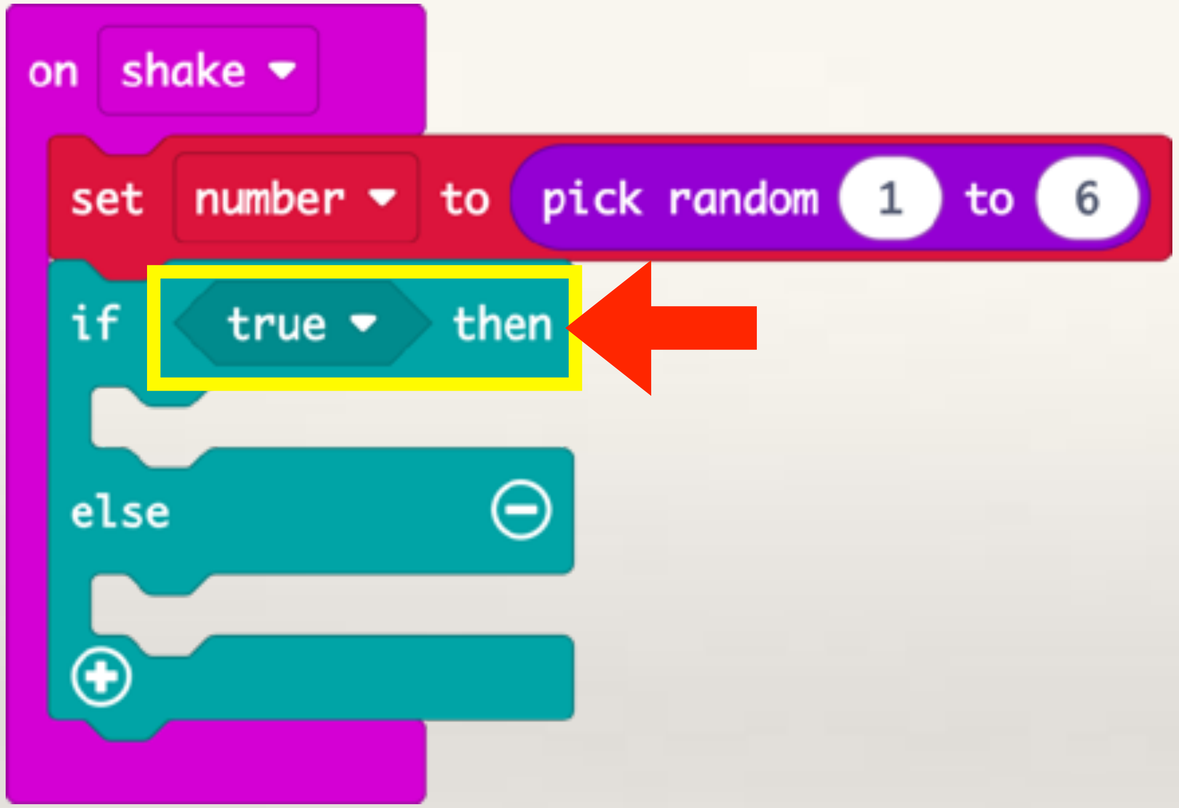
Thus, every time the Micro:bit is shaken, a random number between 1 and 6 will get generated and it will get saved under the variable called Number (that we created).

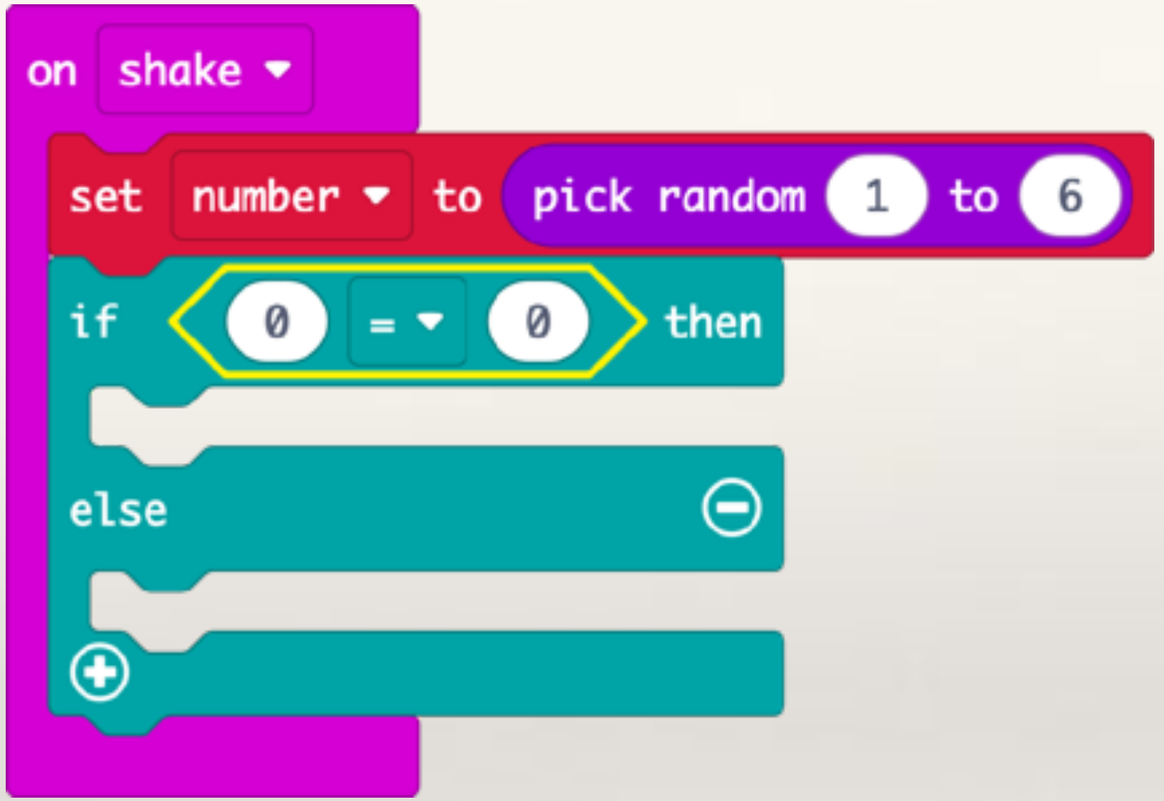
We will also need If-Then-Else conditional block and “=“ comparison block. Both are in ‘Logic’.

**STEP-4**

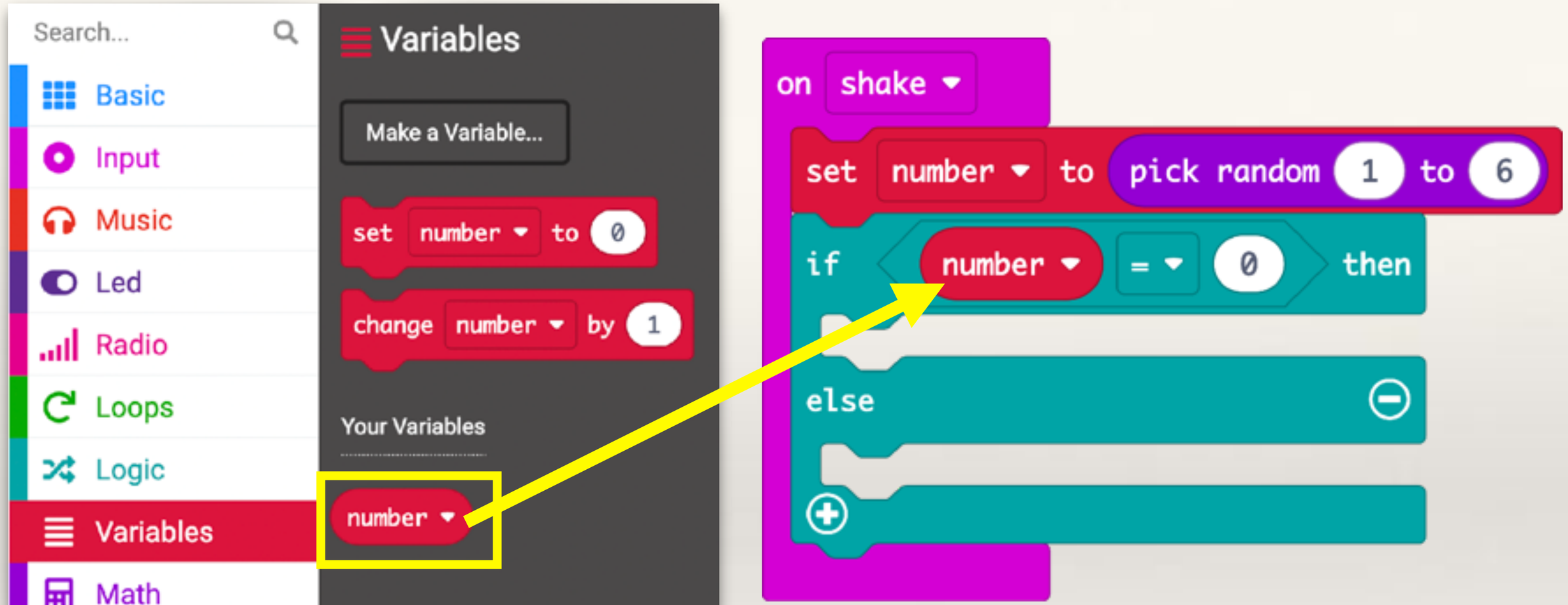
Drag out the If-Then-Else and Equal-to blocks (we will use these later)





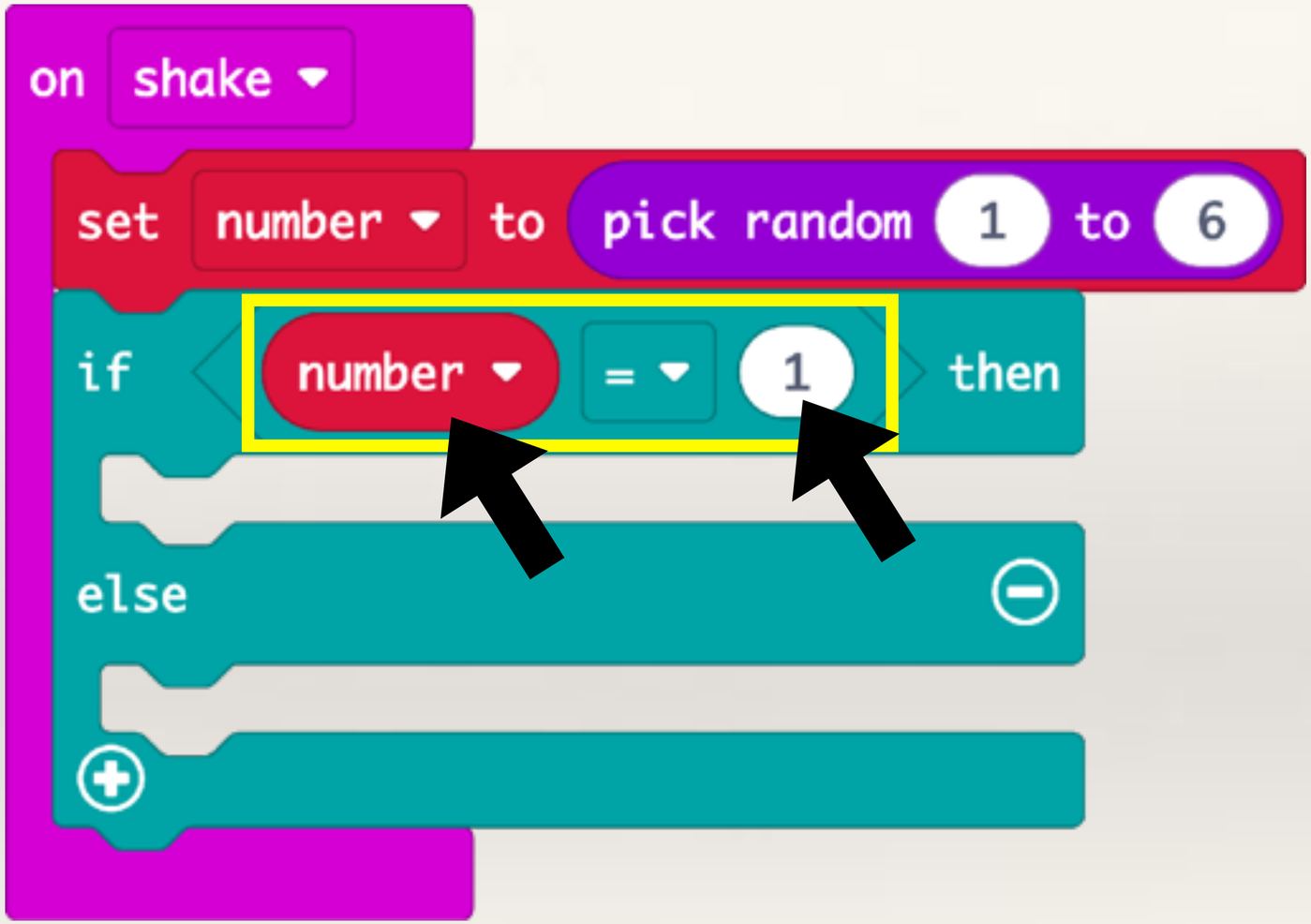

Drag the If-Then-Else block below Set Number command and put the Equal-to block where it say “true”




Drag the Number variable where it says “True”

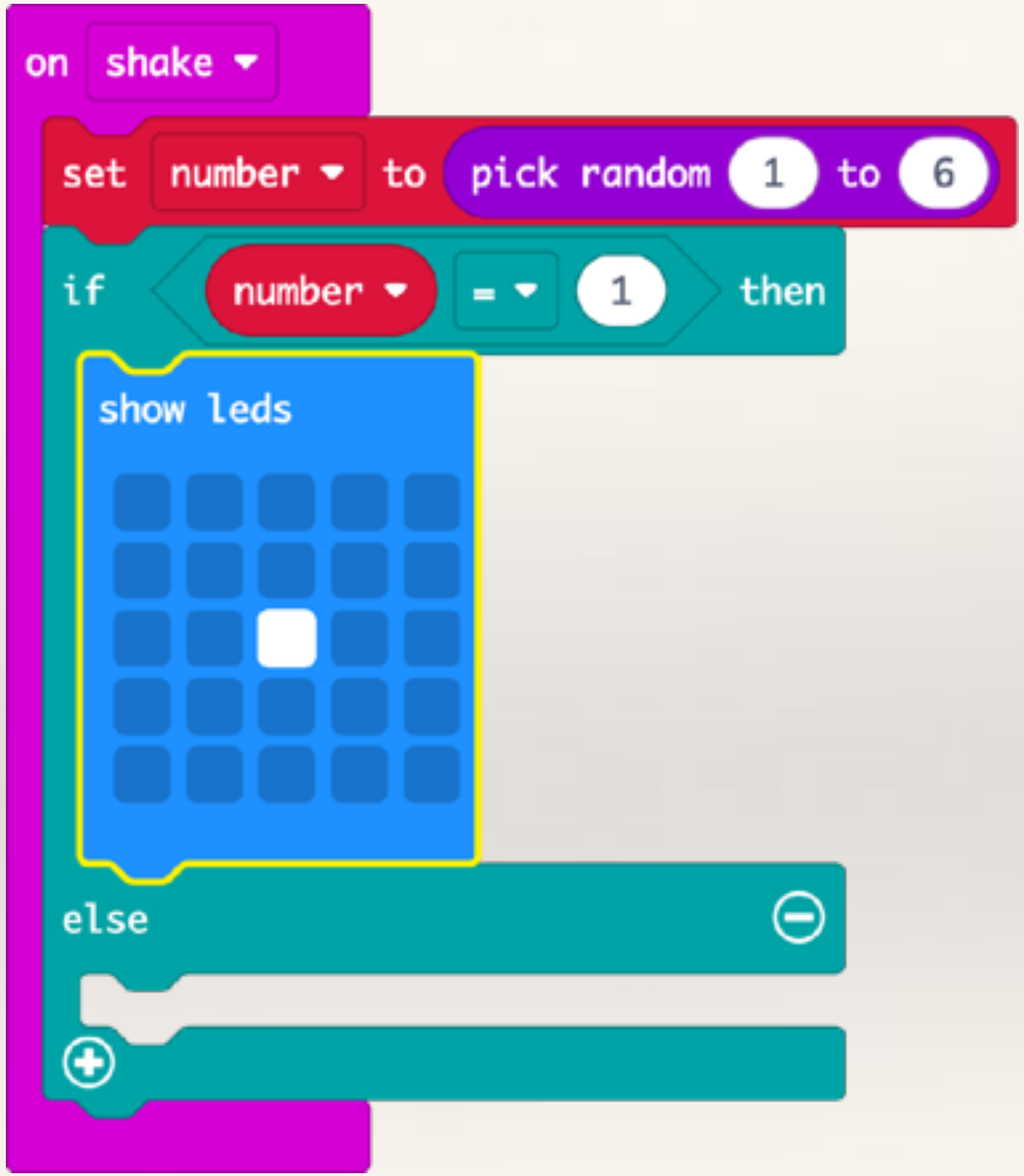
## STEP-5

Now, we want to start checking what random number gets generated when the Micro:bit is shaken. We first check if the number generated (which is stored in the variable called number) = 1



## STEP-6

If Number = 1 is ‘True’ then we want to light up 1 LED

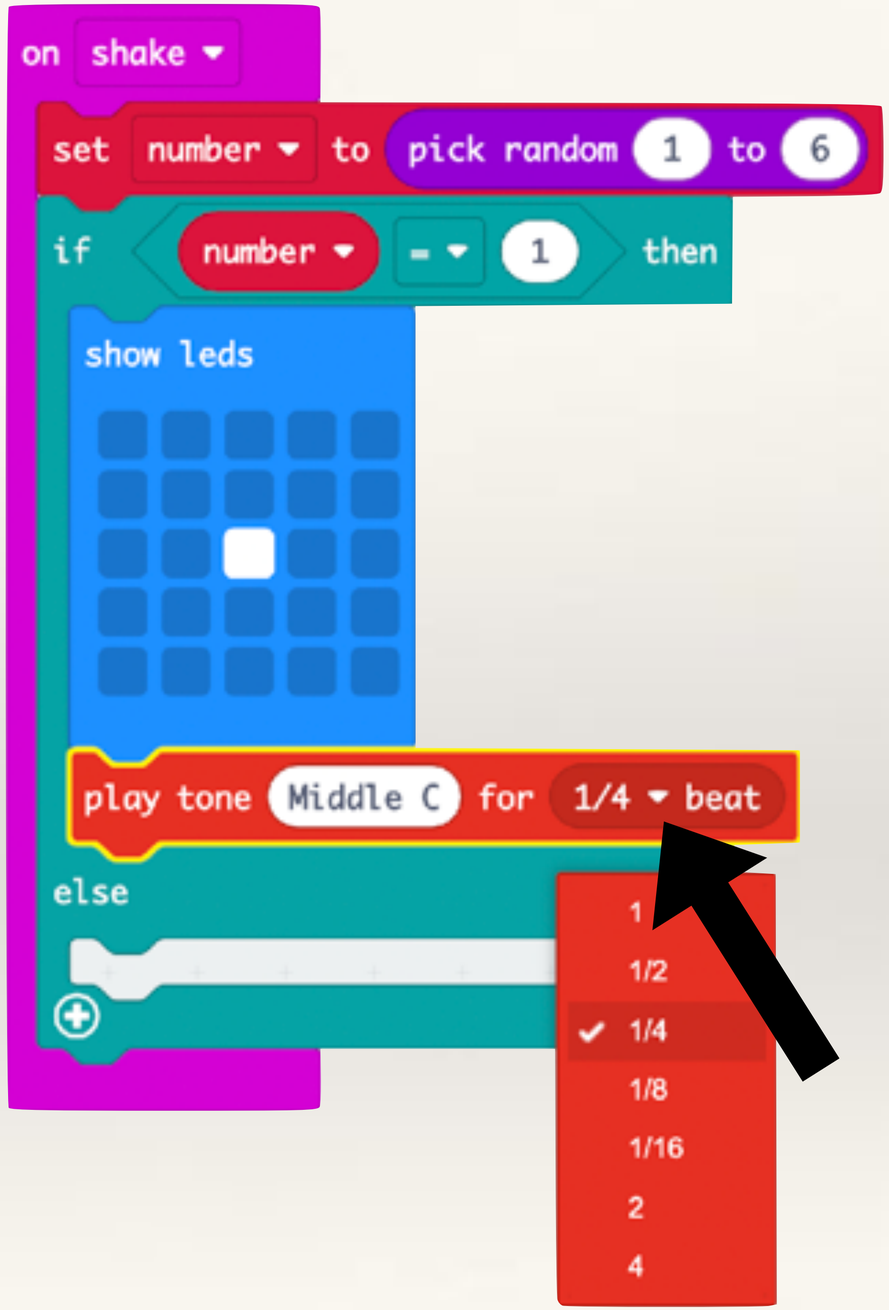


## STEP-7

If Number = 1 is ‘True’ then we want to light up 1 LED

AND

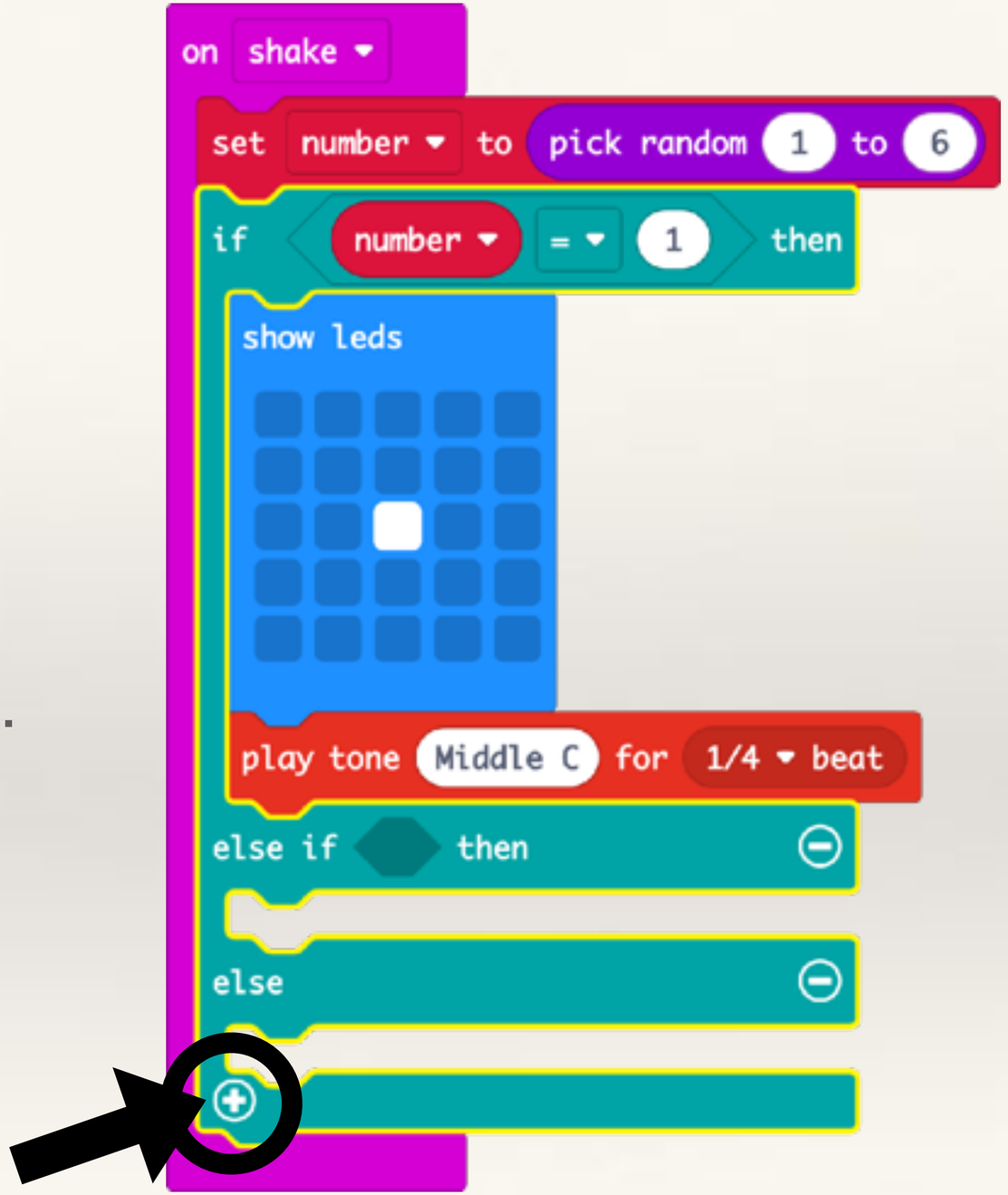
We want to make 1 short beep. For this we need to drag out a ‘Play Tone’ block from ‘Music’ and to keep the duration of the beep very short, we choose 1/4 beat (otherwise it will take a long time for the beep to play).

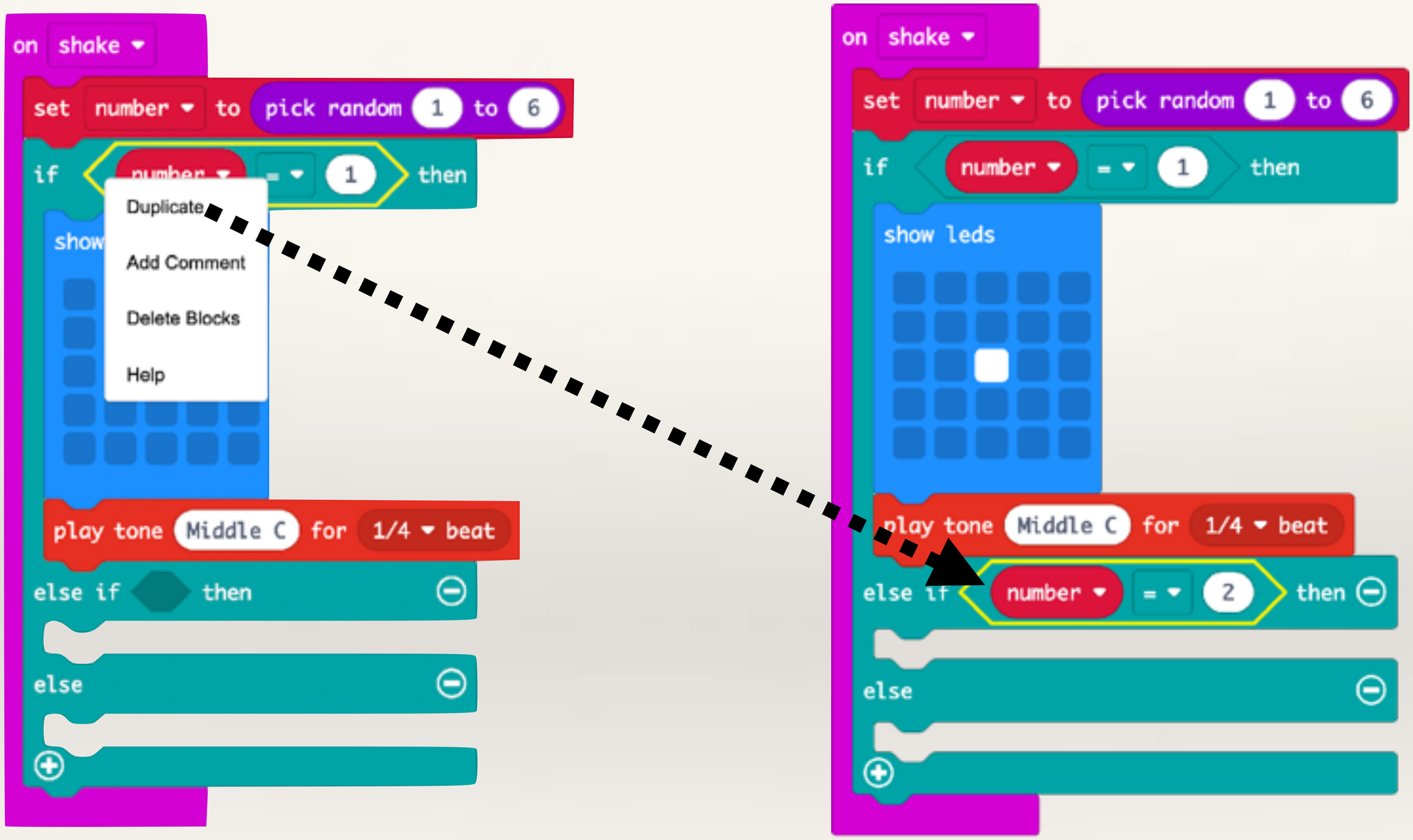


## STEP-8

To check more conditions, I.e. if random number generated = 2, we need to press the “+” button.

This will add a “Else If” statement. We can more Else If statements, simply press “+” again.

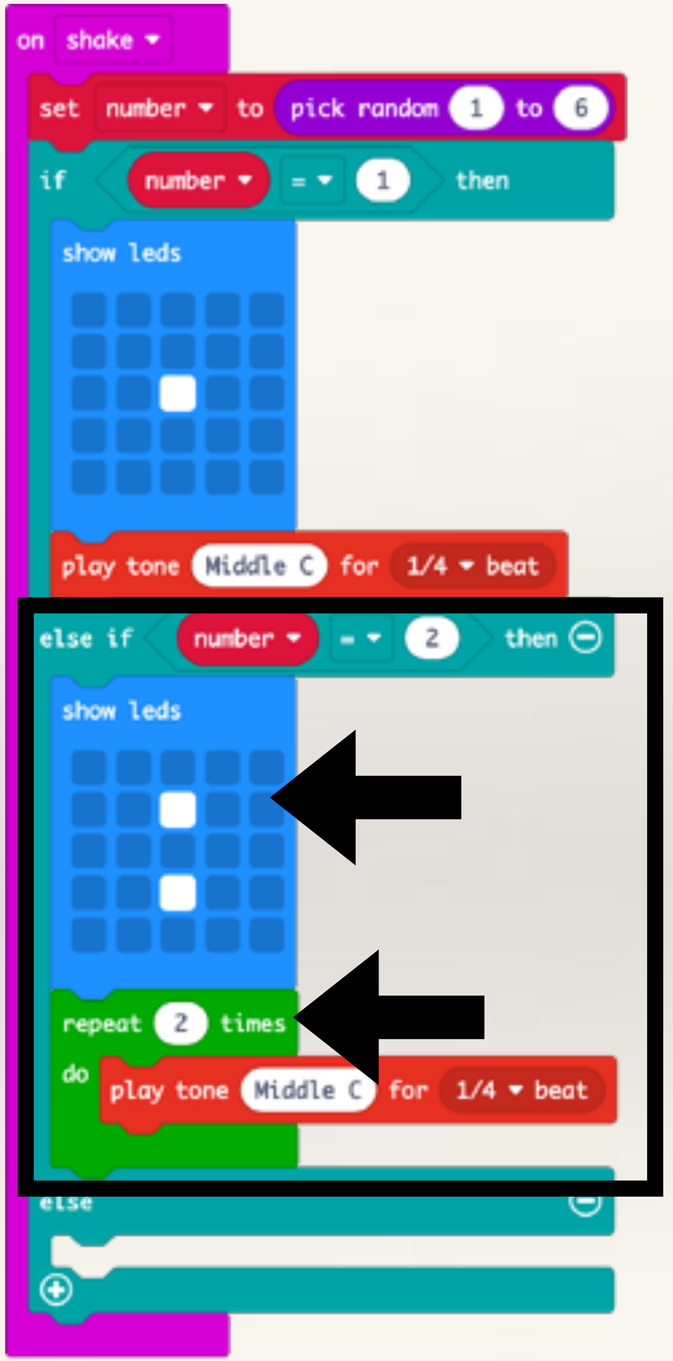





Right-Click > Duplicate and drag inside the Else-If statement and check if the random number generated (and stored in the variable called Number) = 2

## STEP-9

If Number = 2 is True, then we need to light up two LEDs and we need to play the musical tone twice (we can use Repeat command for this).



## STEP-10

Click “+” to generate more Else-If statements, check if random number generated is 3, 4, or 5  and light up equivalent LEDs and play equivalent beeps (using Repeat).

For the last possibility, Number = 6, you can simply use the “Else” command (and not Else-If). This is because the outcomes can only be 1, 2, 3, 4, 5, or 6. We have checked for 1 to 5 and if the outcome of the shake is not one of these then the only possibility left is that random number generated = 6. So we don’t have to check this condition. In the “Else” block we can simply light up 6 LEDs and sound 6 beeps.

See the next page for complete code.


on shake
set number to pick random 1 to 6
if number = 1 then
show leds
play tone Middle C for 1/4 beat
else if number = 2 then
show leds

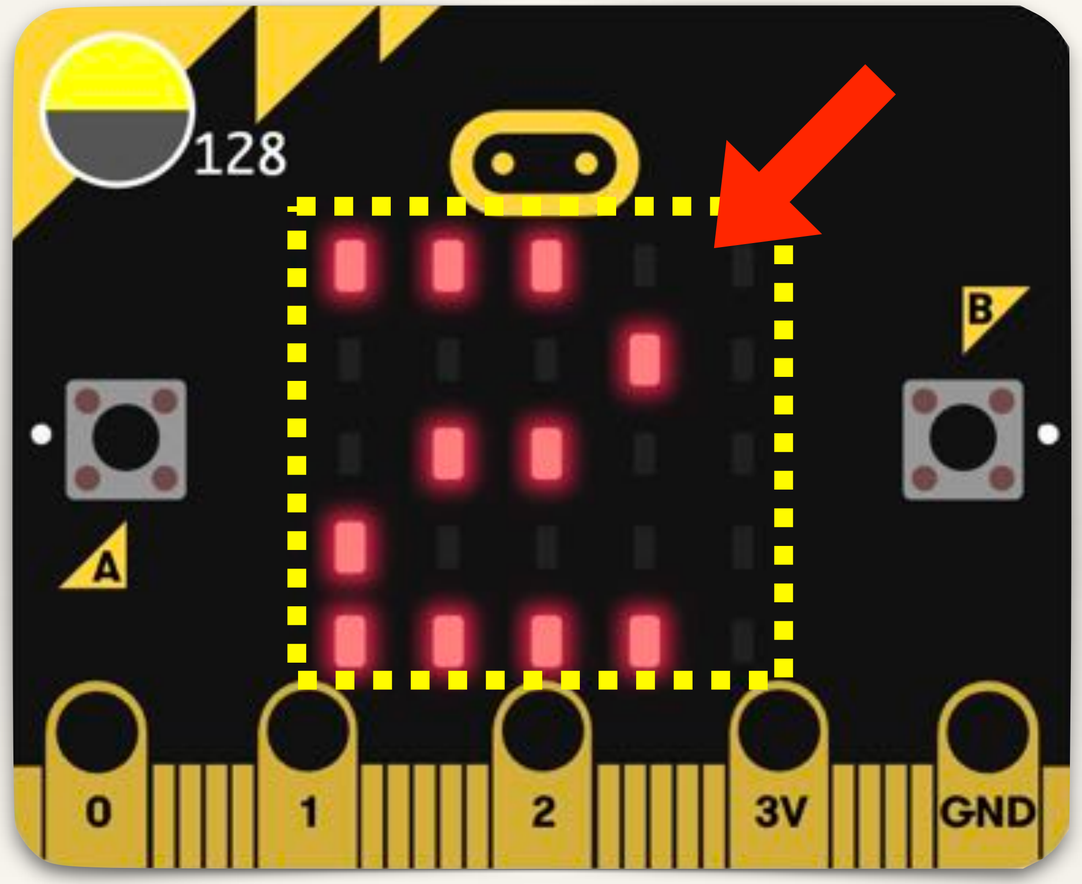

repeat 2 times
do
play tone Middle C for 1/4 beat
pause (ms) 100
else if number = 3 then
show leds
repeat 3 times
do
play tone Middle C for 1/4 beat
pause (ms) 100
else if number = 4 then
show leds


repeat 4 times
do
play tone Middle C for 1/4 beat
pause (ms) 100
else if number = 5 then
show leds
repeat 5 times
do
play tone Middle C for 1/4 beat
pause (ms) 100
else
show leds
repeat 6 times
do
play tone Middle C for 1/4 beat
pause (ms) 100

On-board Sensors

# LIGHT SENSOR

# LIGHT SENSOR




- The Micro:bit measures the light around it by using some of the onboard LEDs.
- Light level 0 means darkness and 255 means bright light.
- The first time you the light sensor the reading will say 0. After that, it will say the real light level. This is because the light sensor has to be turned on first.

Project-6

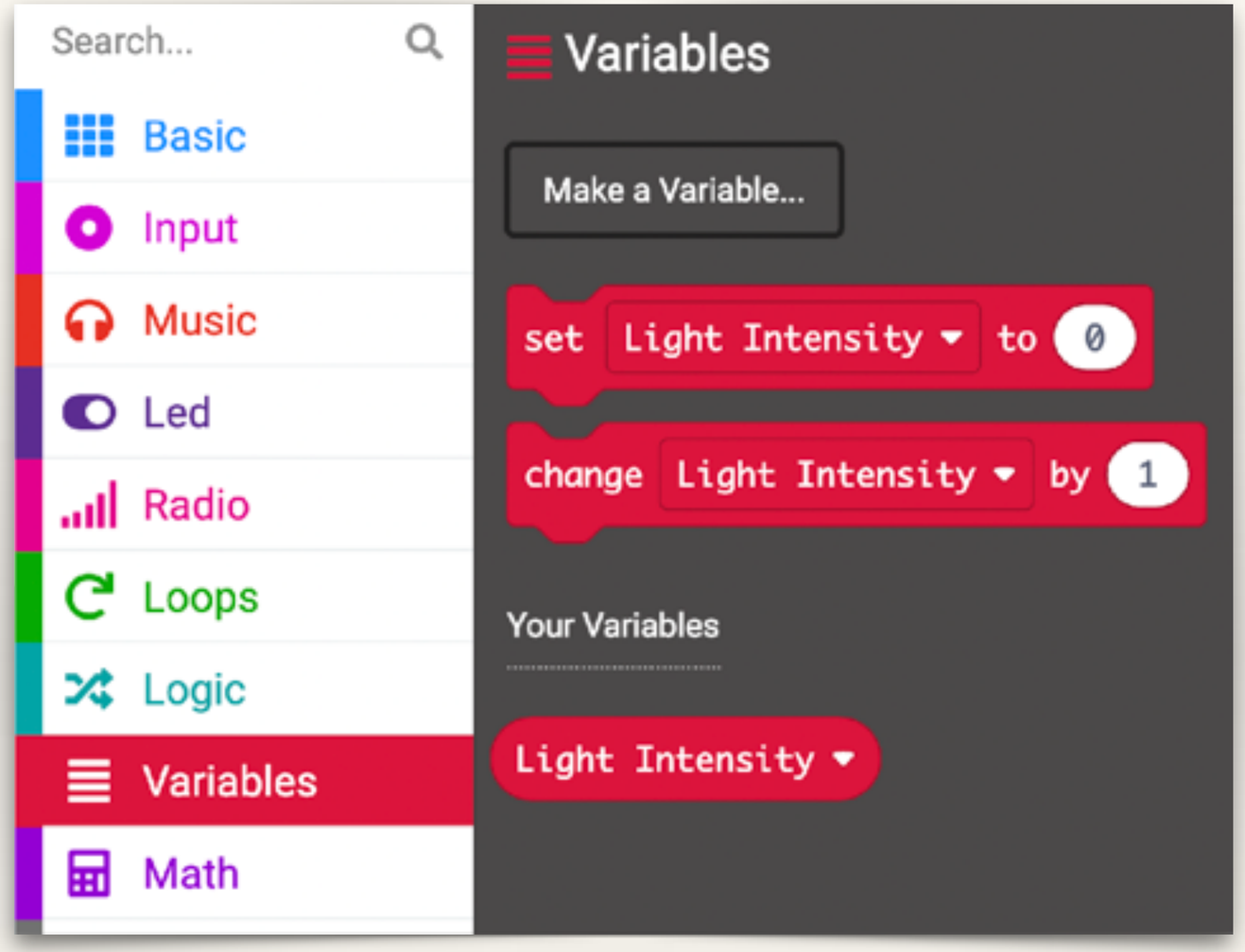
# Smart Street Light

**Objective:** to learn about the light sensor and how it can be used.

**Problem:** use the light sensor to create a smart street light. When the Sun is shining, the light sensor reading is high (200+) and when the Sun sets the light sensor reading is low (less than 100). Use this property to glow some onboard LEDs on the micro:bit when the light level is less than 100. The onboard LEDs depict a street light which automatically switches on when the sun sets and it starts to become dark.

This project requires basic knowledge of Variables and If-Then conditional statements.

Create a new variable and give it an appropriate name. Here, we are calling our variable ‘Light Intensity’

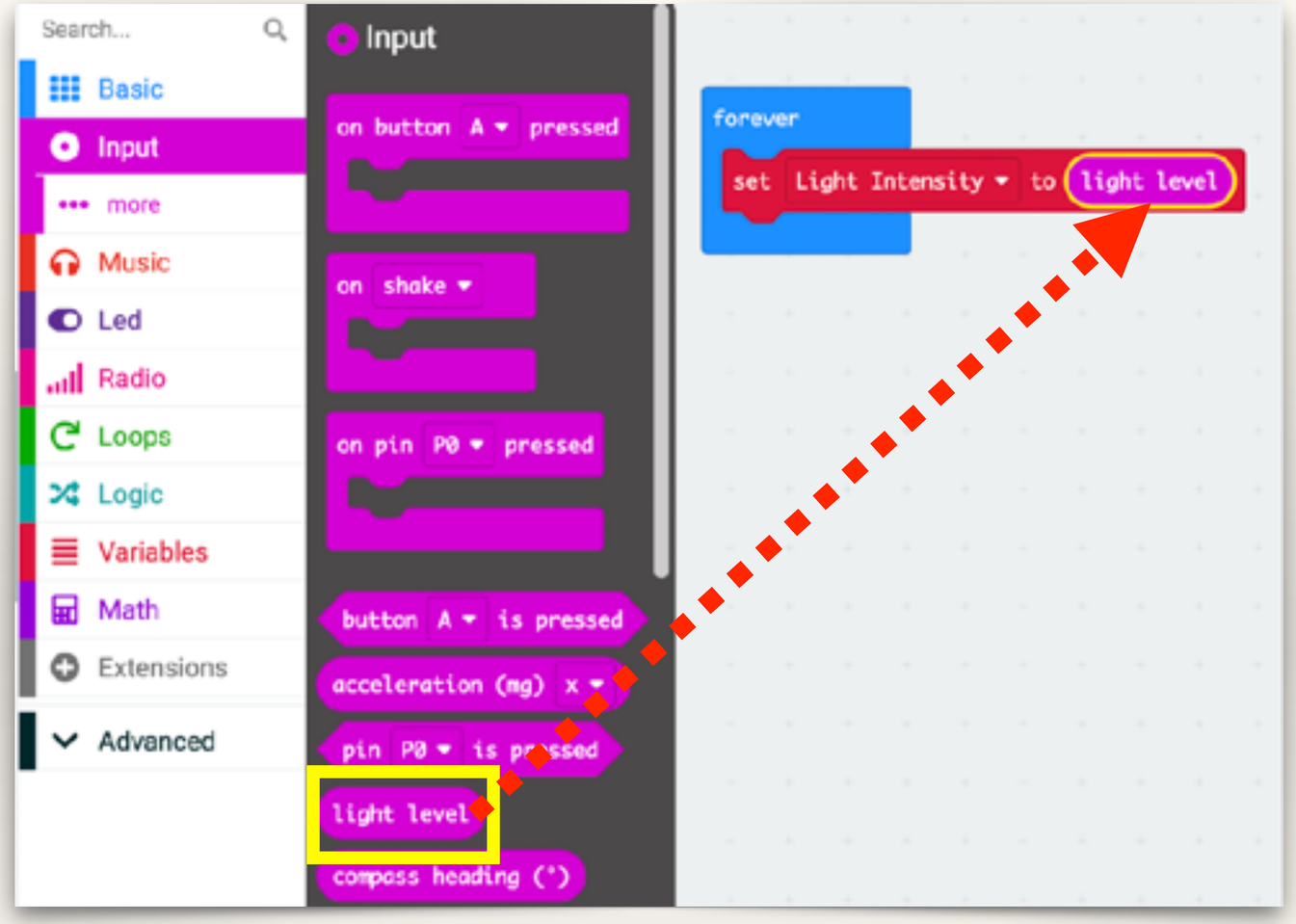


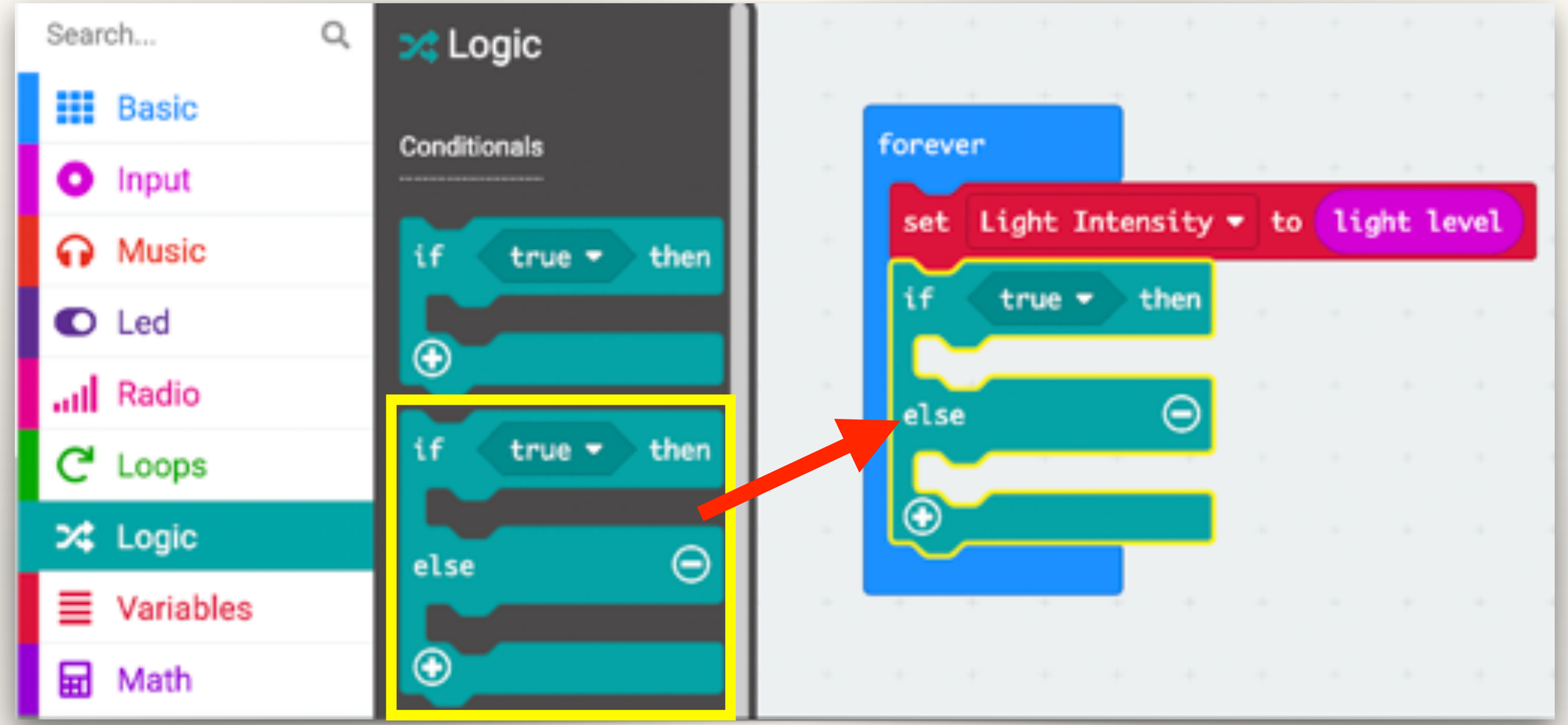
From ‘Basic’ drag out a ‘Forever’ block.

From ‘Variables’ drag out a ‘Set Variable’ block and put it inside the Forever block.

From ‘Input’ drag out ‘Light Level’ block and put it inside the Set Light Intensity block.

Now, whatever is the level of light falling on the micro:bit, it will get updated and stored in the variable called Light Intensity.

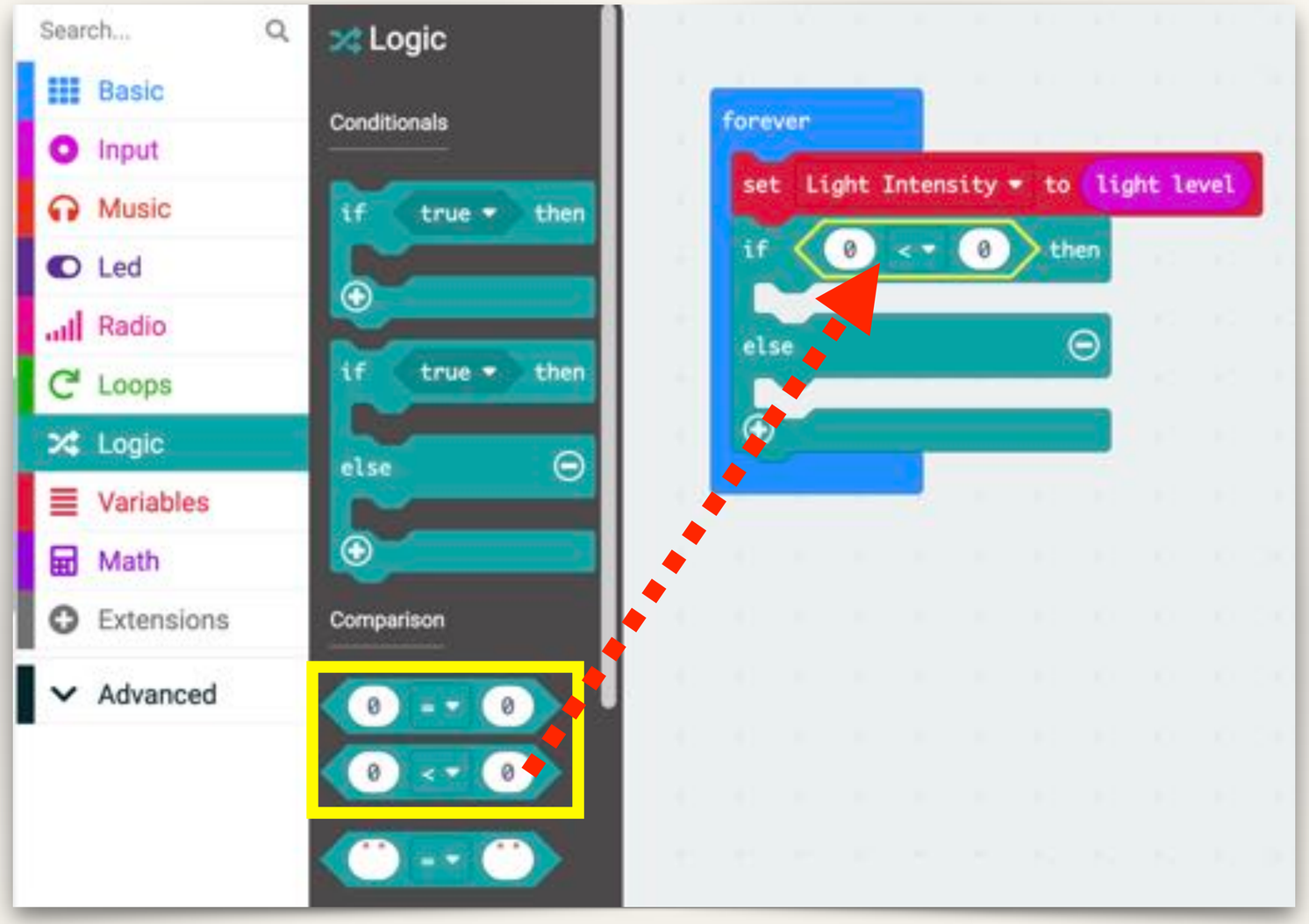





From ‘Logic’ drag out an ‘If-Then-Else’ block.

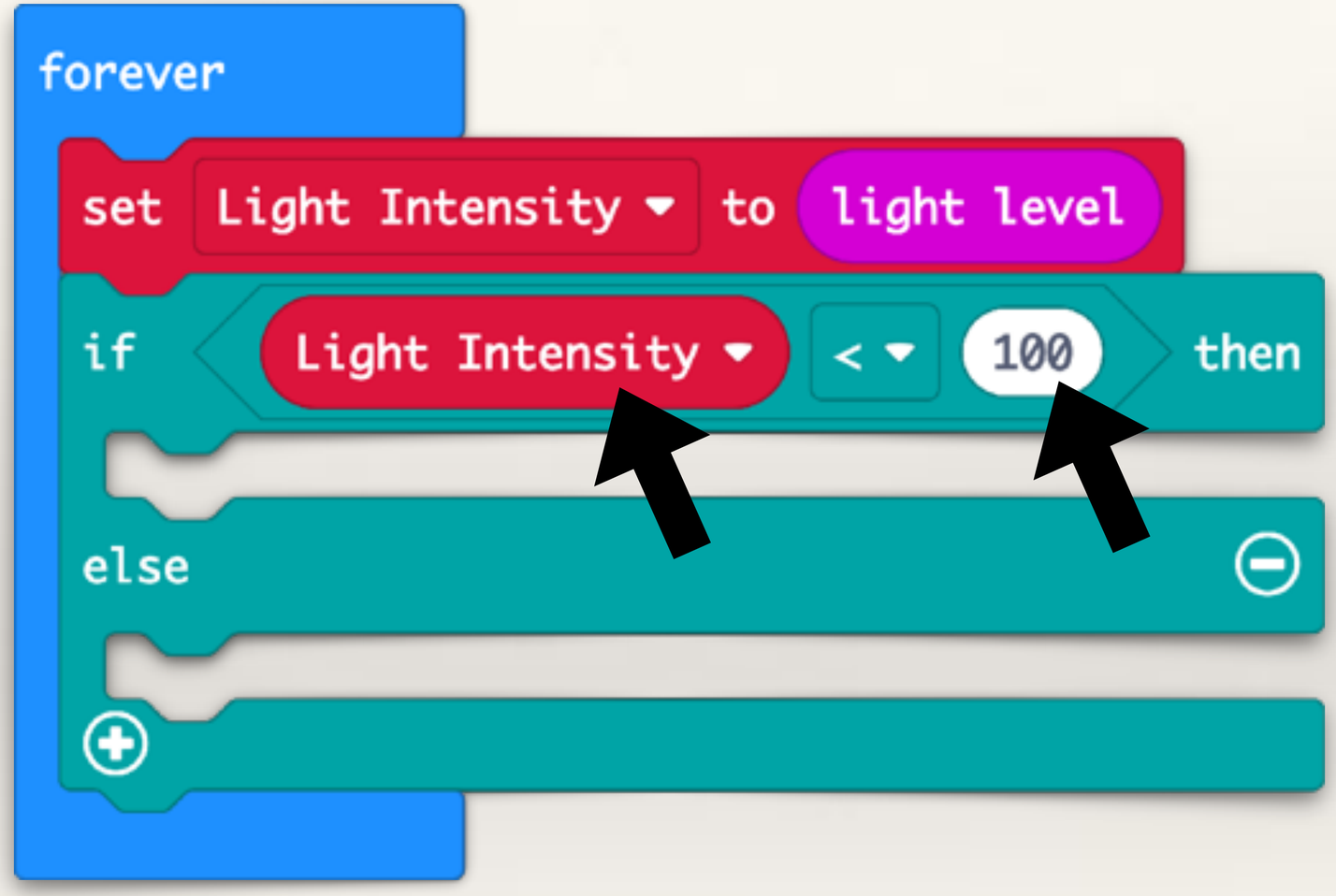
From Logic > Comparison > drag out a ‘Less Than’ block.

Put it inside the If-Then block (where it says ‘True’)



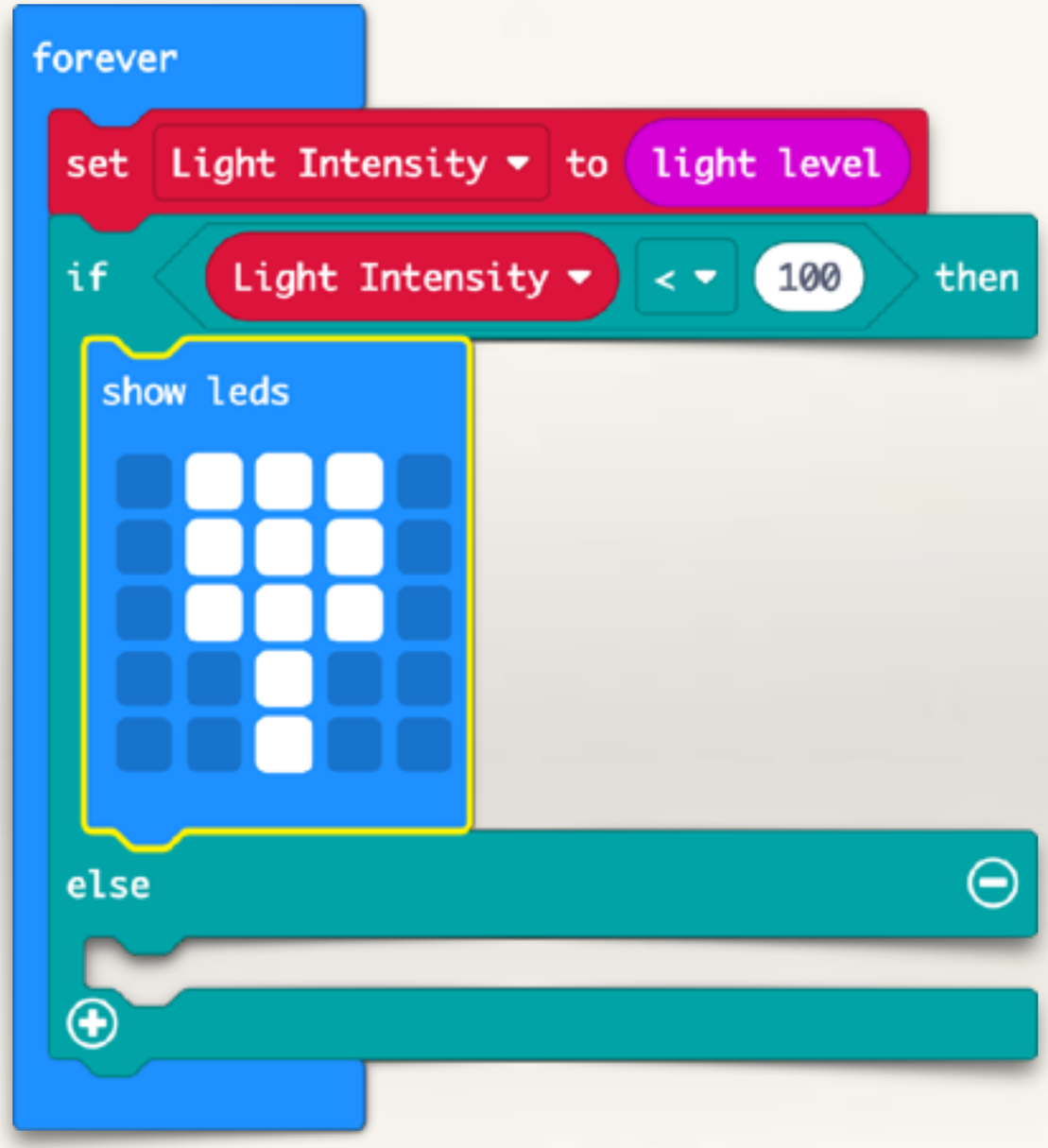
Let’s check the condition needed for our Smart Streetlight - that Light Level falling on the micro:bit should be less than 100.

If this condition is true, which means Sun is setting and it is getting dark, so we want the streetlight to switch on.



From ‘Basic’ drag out the ‘Show LEDs’ block. Light up a few LEDs to depict a street light.

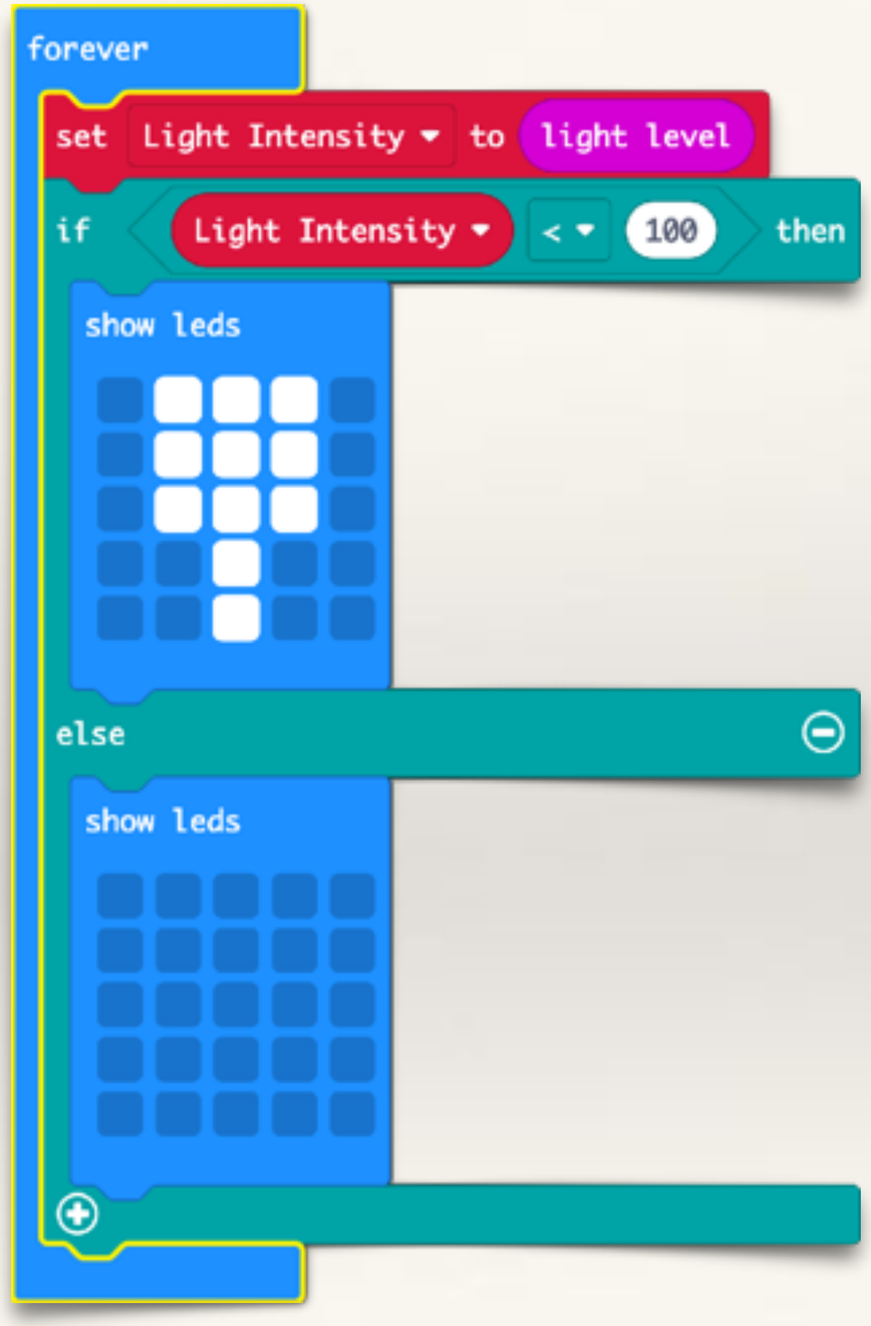
Now, if it is getting dark and the light level falling on the micro:bit is less than 100, this LED display will show.

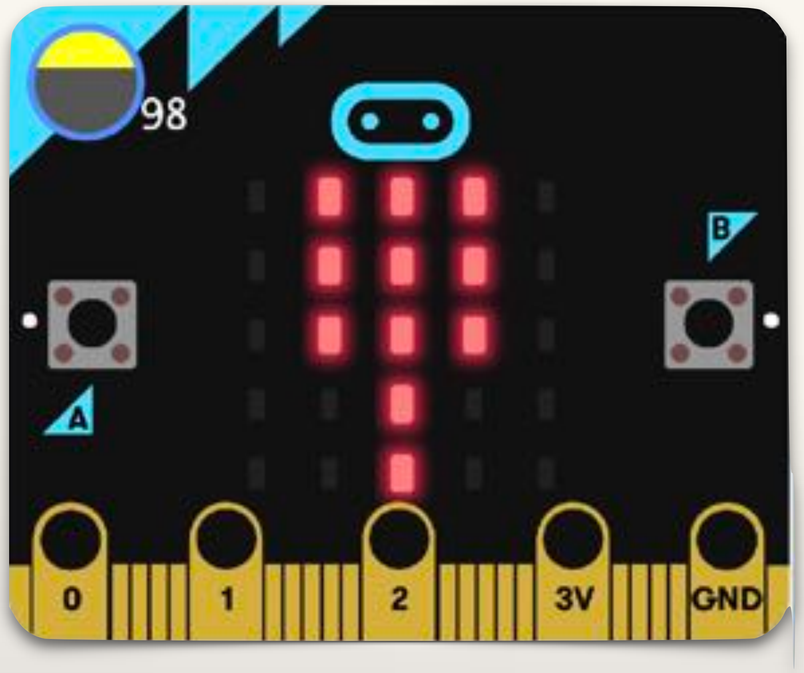


The only other condition we want to check is if the light level falling on the micro:bit is more than 100.

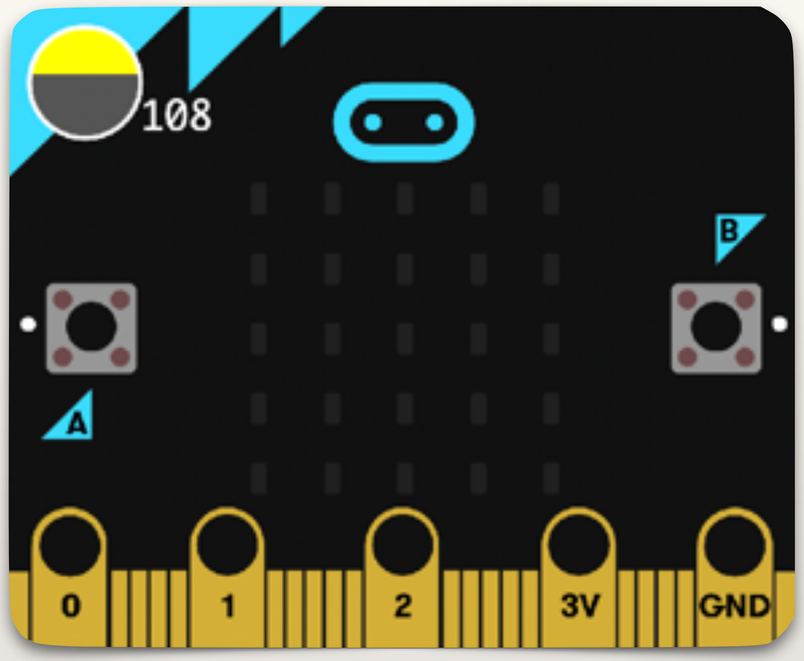
So we only need to use the Else condition (i.e. if the condition “Light Level < 100” is NOT true).

In this case, we want the streetlight to switch off. This is depicted by none of the LEDs lighting up.






In the simulator, if the light level (stored in the variable called Light Intensity) is less than 100, LEDs will glow.



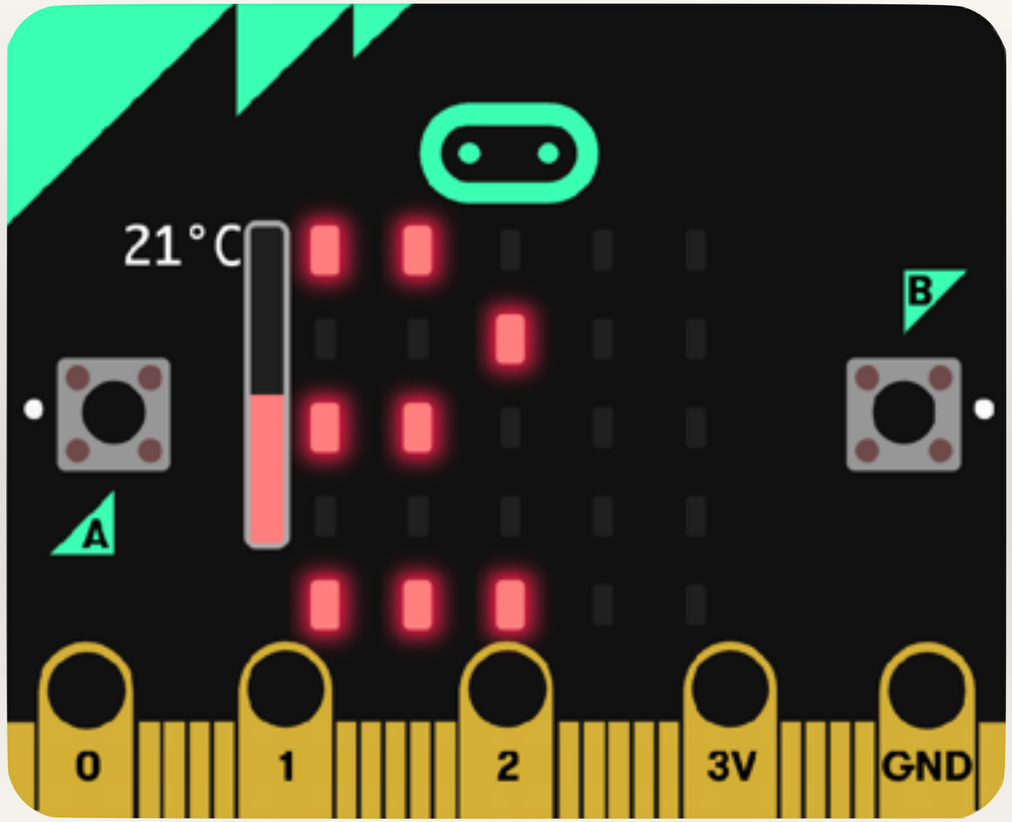

In the simulator, if the light level (stored in the variable called Light Intensity) is more than 100, LEDs will not glow.

By using sensors, we can make machines ‘autonomous’ that is, machines that are capable of taking decisions on their own. For example, in this case, we don’t have to physically switch the streetlight on and off, based on the light sensor reading, the streetlight will detect if it is becoming dark and switch on by itself.

On-board Sensors

# TEMPERATURE SENSOR

# TEMPERATURE SENSOR




- To find the temperature of its surroundings, the micro:bit checks how hot its processor chip is.
- Since this chip usually does get very hot by itself, the temperature of the CPU is usually close to the temperature of wherever the micro:bit is.

Project-6

# Smart Fan

**Objective:** to learn about the temperature sensor present onboard the Micro:bit and how it can be used.

**Problem:** use the temperature sensor to create a smart fan.

- When the room temperature rises over 21°C a mini servo attached to the Micro:bit should start oscillating.
- To depict a fan, we are using a Continuous Servo motor instead of a geared DC motor. This is because a geared DC motor draws more current than what the Micro:bit outputs so you need to attach a motor-driver board. For test purposes, one mini continuous servo can be safely attached directly to the Micro:bit.

This project requires basic knowledge of Variables and If-Then conditional statements.

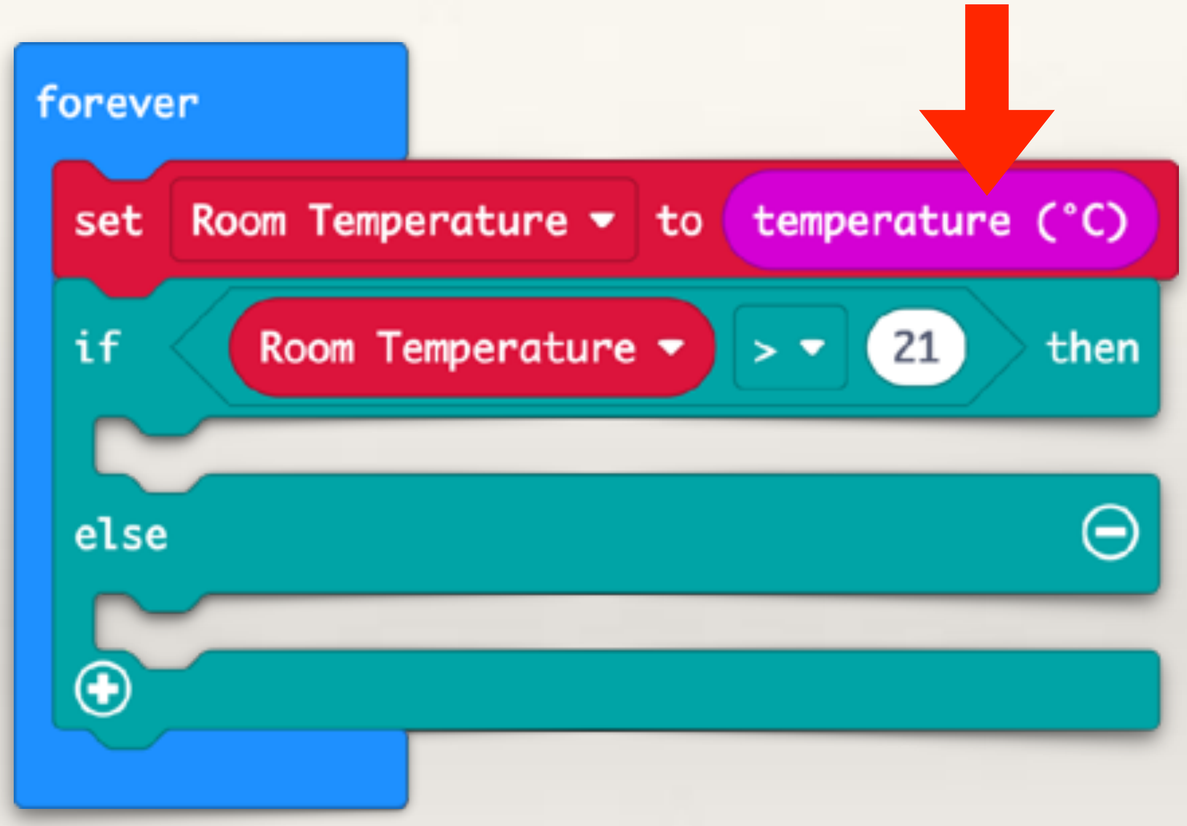
Create a new variable. Call it Room Temperature.

Use the Set Variable command and assign the temperature read by the micro:bit (you will find this under Input) to Room Temperature variable.

Drag out a If-Then-Else conditional and a Greater Than block from Logic.

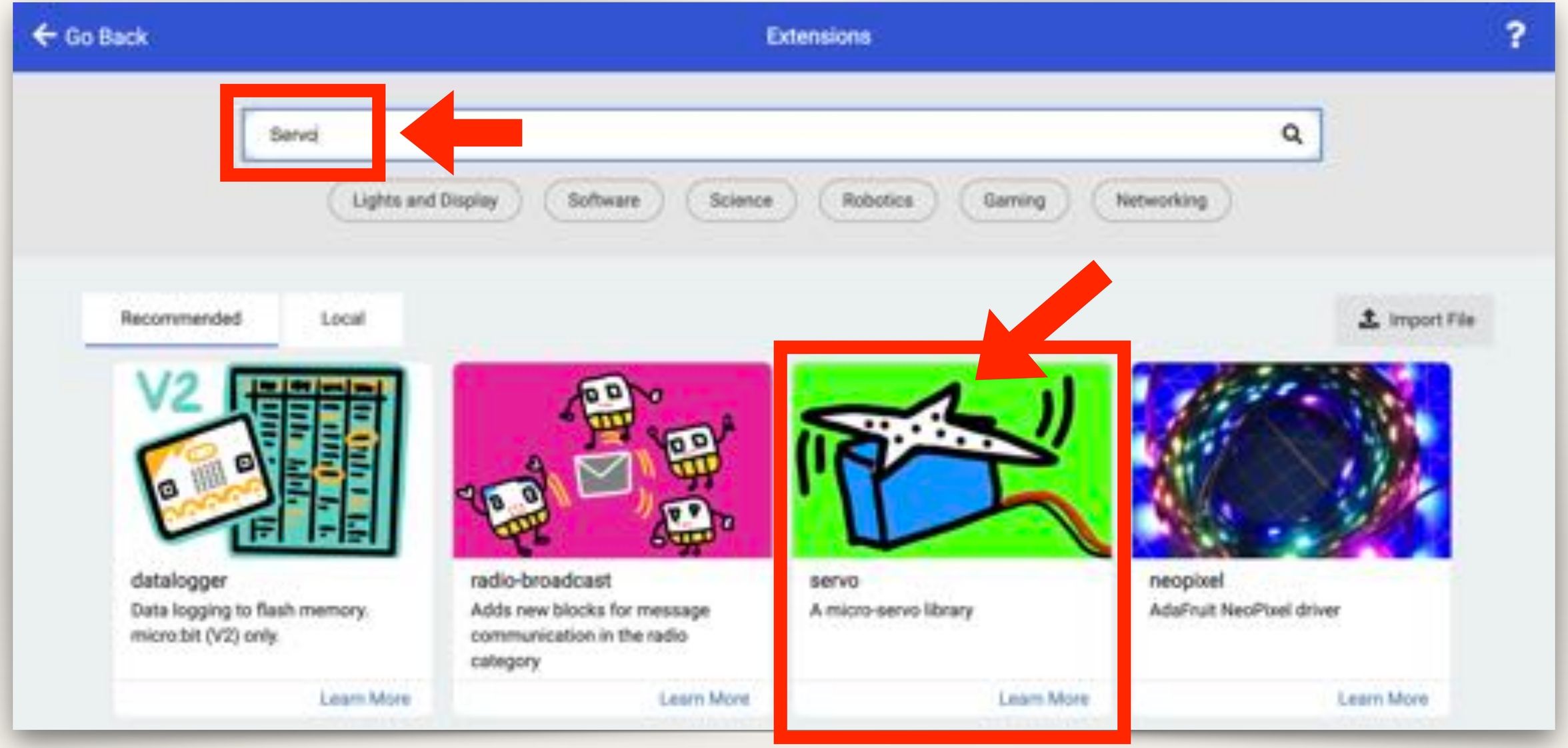
Check for the condition: Room Temperature > 21

Temperature read by the micro:bit (you will find this under Input)




We need to add an Extension to get the programming blocks for a Servo motor.

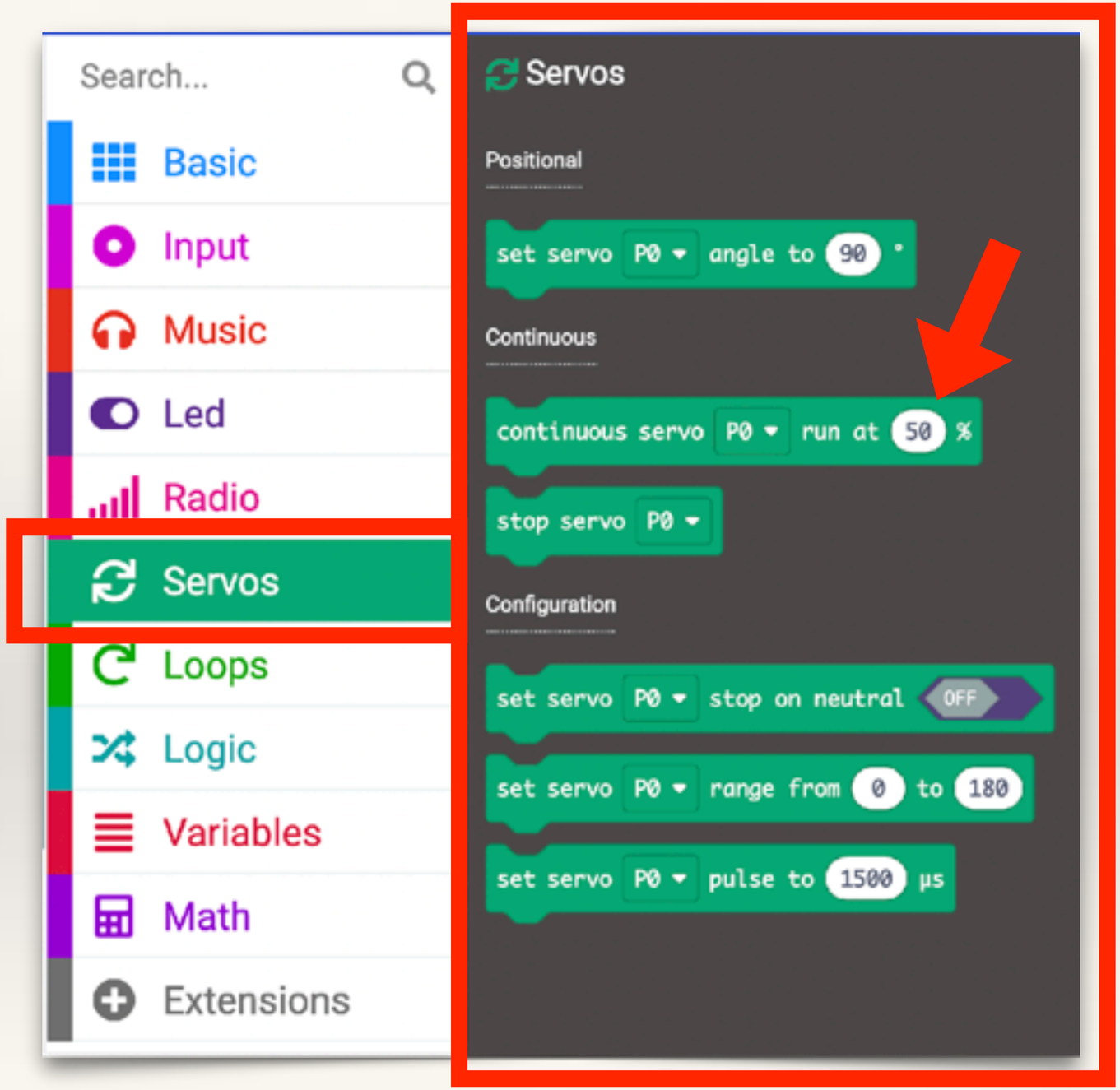
For this, click the ‘gear’ icon on top right and then click on ‘Extensions’.




If you don’t see the Servo extension, you can search for it. Click on it to install.

Once you install the Servo Extension, a new set of blocks should appear.

Drag out ‘Continuous Servo at P0 run at 50%’ block.

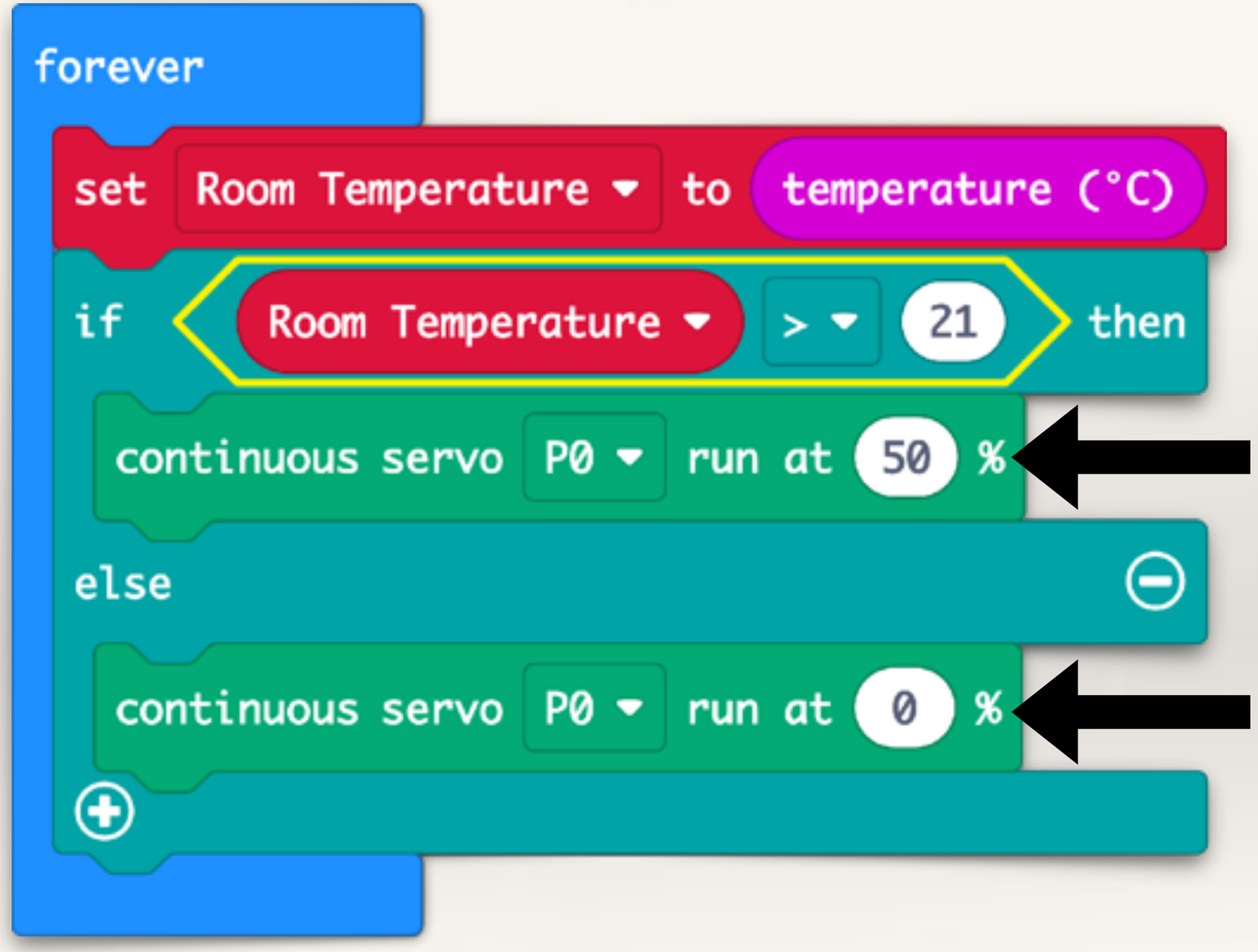


## Set the Conditions

IF Room Temperature > 21 (is true) THEN rotate the servo motor.

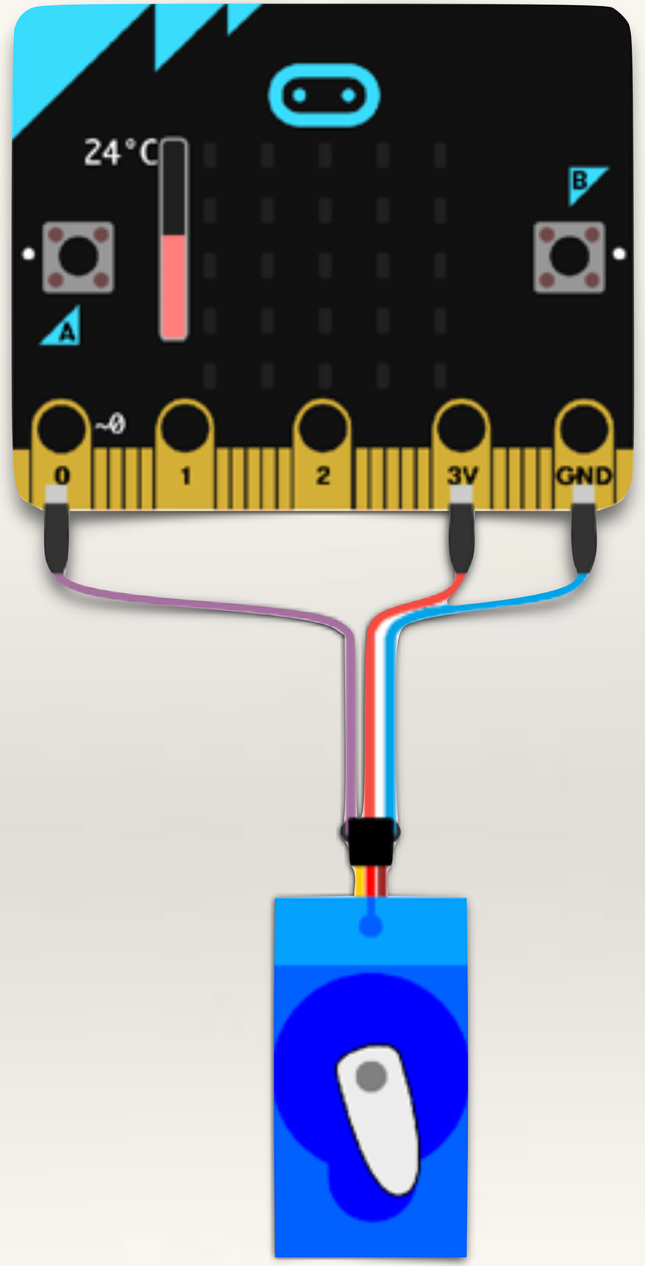
ELSE (which means if Room Temperature is less than or equal to 21°C)

Don’t rotate the servo motor.



## Your Smart Fan is Ready!

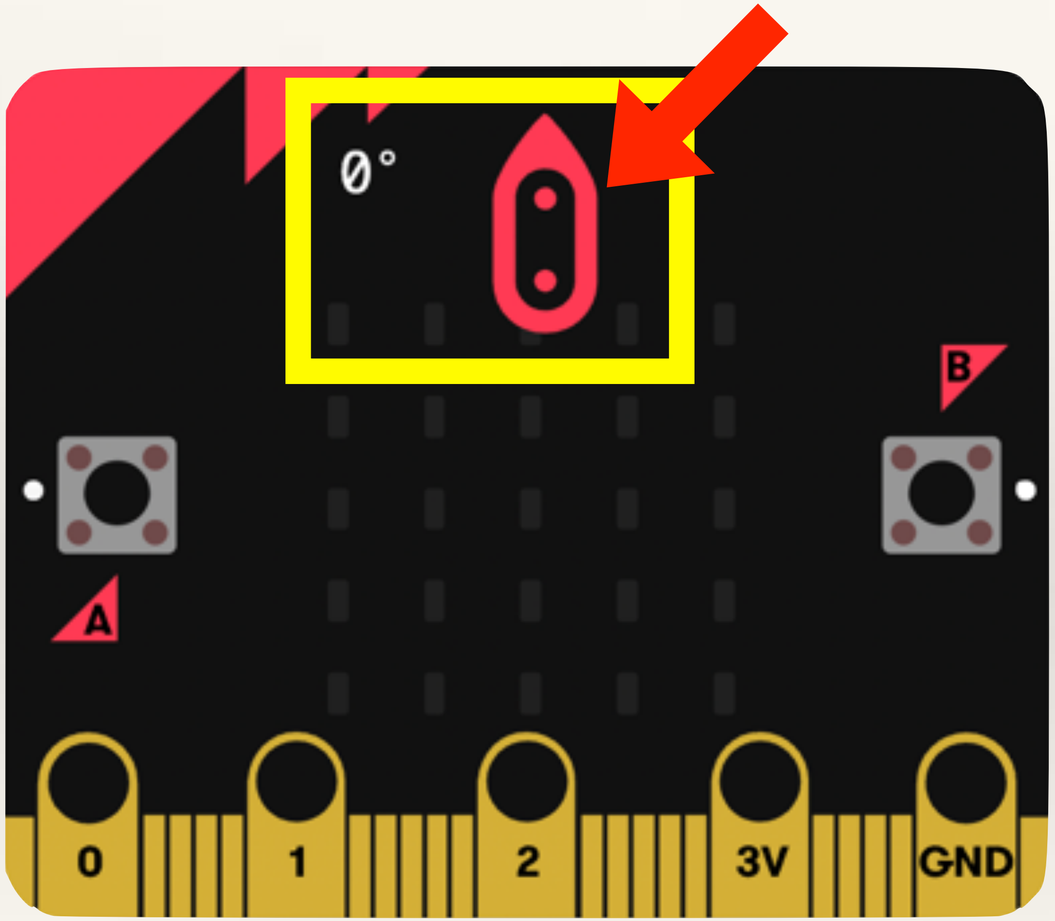
You have made a fan that is ‘autonomous’ i.e. it can take its own decision — if the room temperature goes over 21°C it will switch on. Else, it will be off.



# On-board Sensors

# **COMPASS** (Magnetometer Sensor)

# Magnetometer / Compass




- The micro:bit measures the compass heading from 0 to 359 degrees with its magnetometer chip. Different numbers mean north, east, south, and west.
- Every time you start to use the compass, the micro:bit will calibrate the compass. It will ask you to fill a pattern on the screen by tilting the micro:bit.

Project-7

# Compass

**Objective:** to learn how to use the Magnetometer sensor to use the Micro:bit like a compass.

**Problem:** use the magnetometer sensor to find out magnetic North and display “N” on the micro:bit when it is pointing to the North direction.

- When the Compass Reading is 90° it means that direction is North.
- Use the Compass Reading block and If-Then conditional statement to make a simple compass that shows which direction is North.

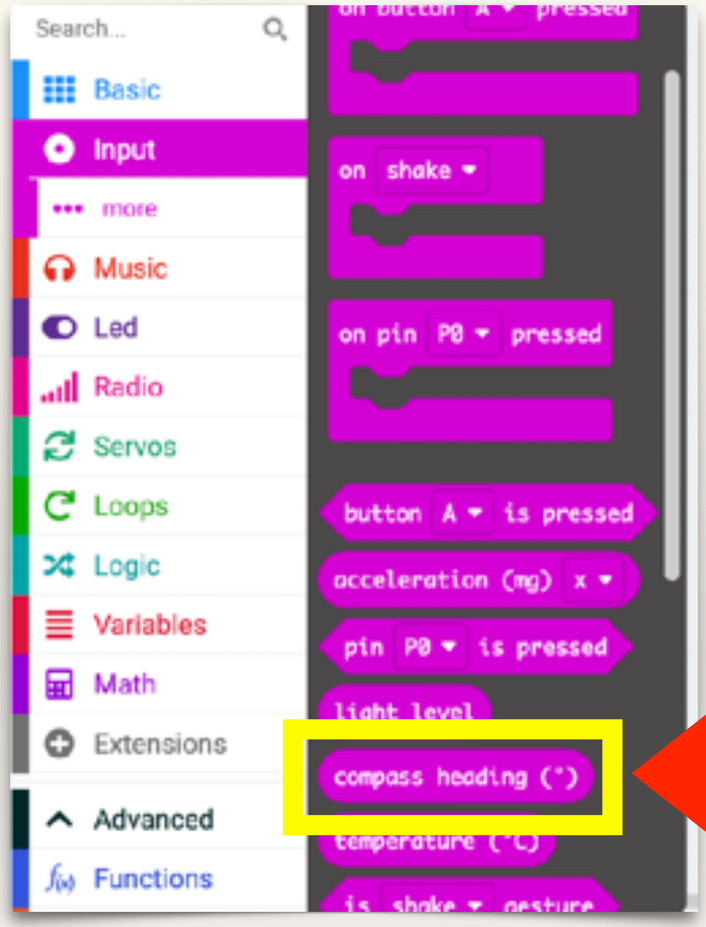
This project requires basic knowledge of Variables and If-Then conditional statements.

Create a new variable. Call it Compass Reading.

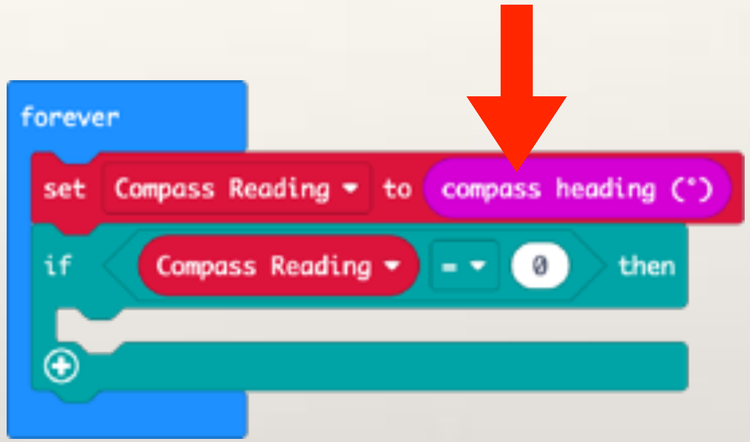
Use the Set Variable command and assign the Compass Heading reading on the micro:bit (you will find this under Input) to Compass Reading variable.

Drag out a If-Then conditional statement.

Check for the condition: Compass Reading = 0° (which means micro:bit is pointing North).



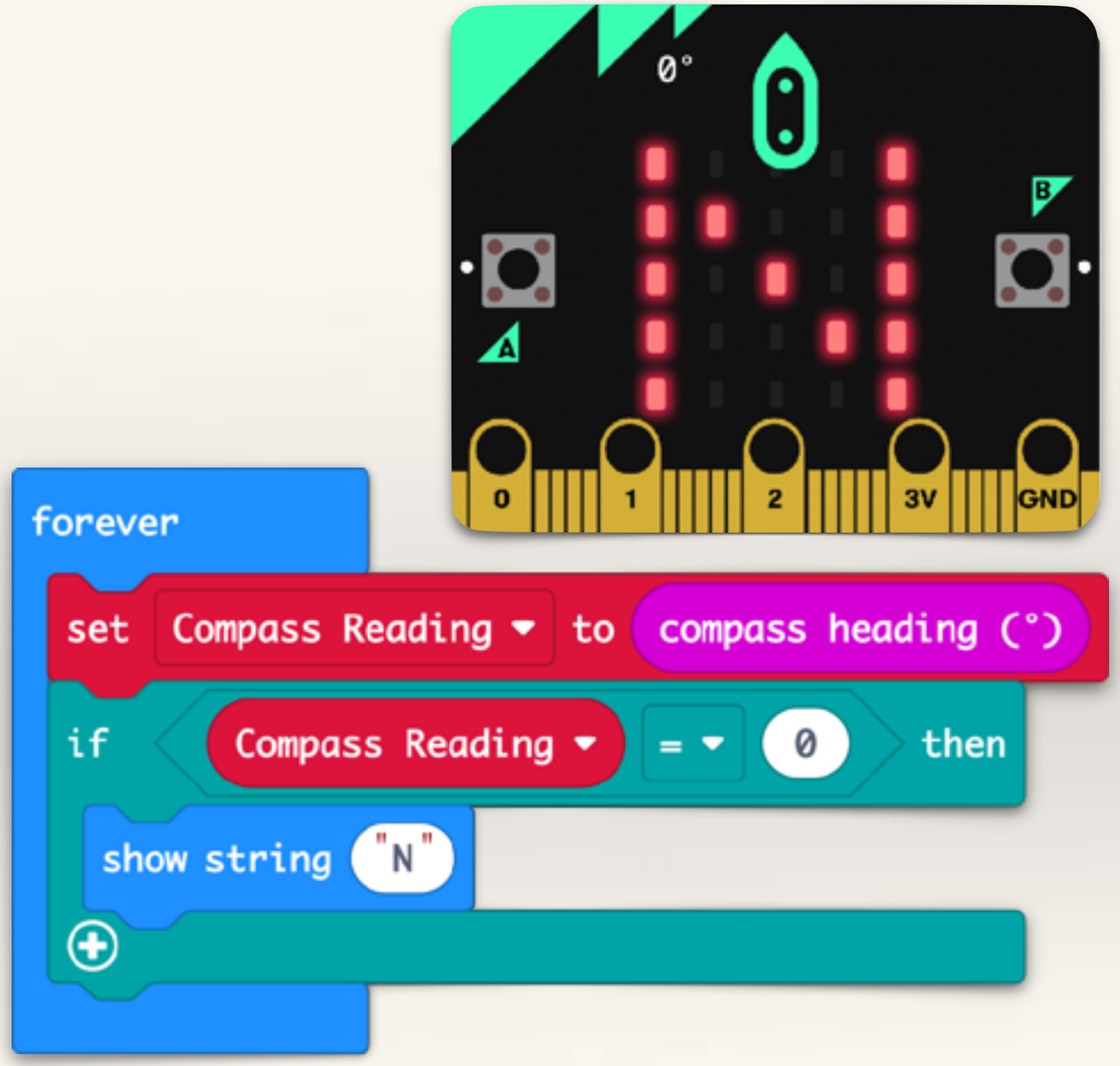

Compass Heading block
(you will find this under Input)



You have made a simple compass. Transfer the code to micro:bit.

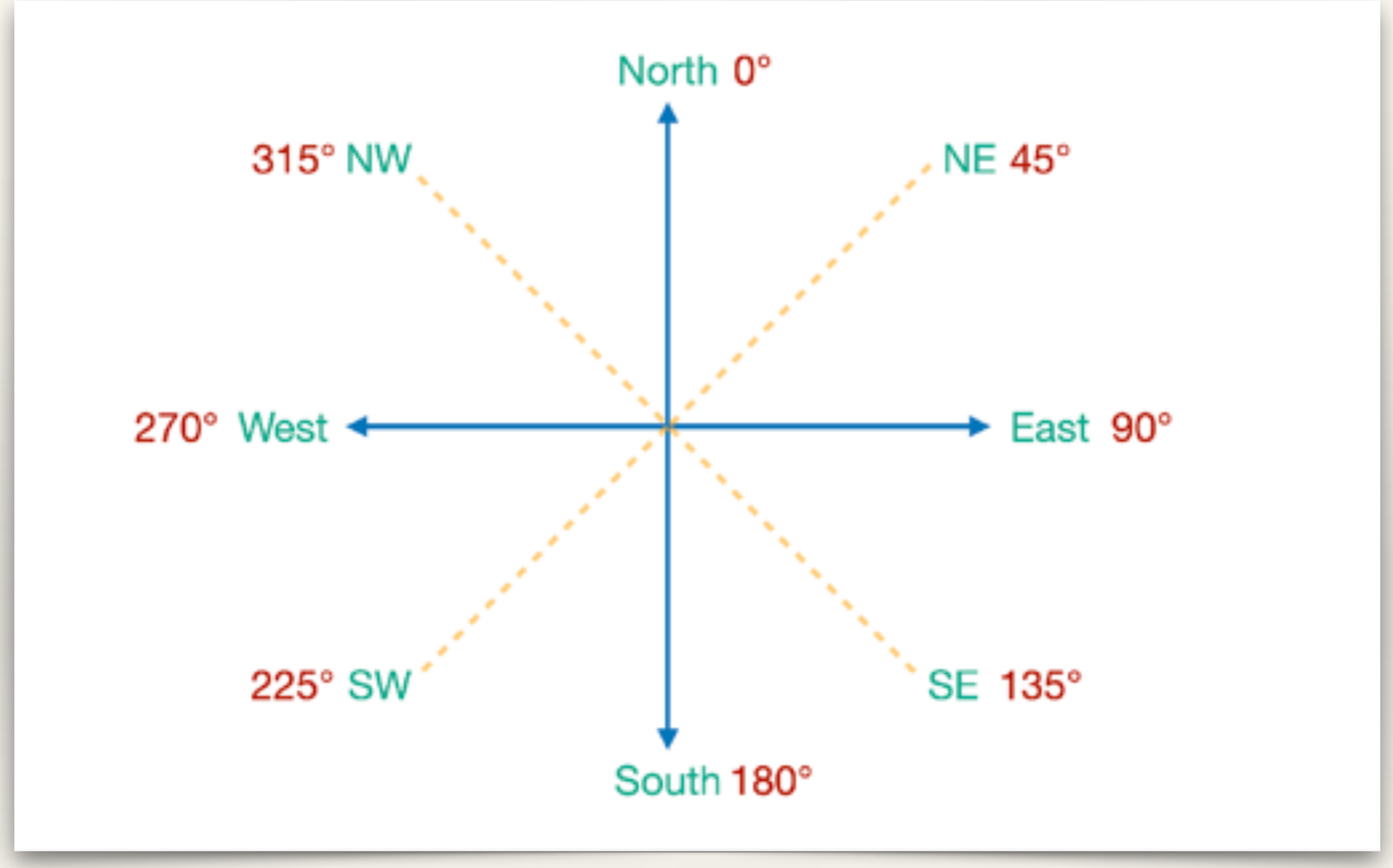
You will have to calibrate the compass when you use it for the first time.

When the micro:bit faces North, the reading will be 0° and “N” will get displayed.



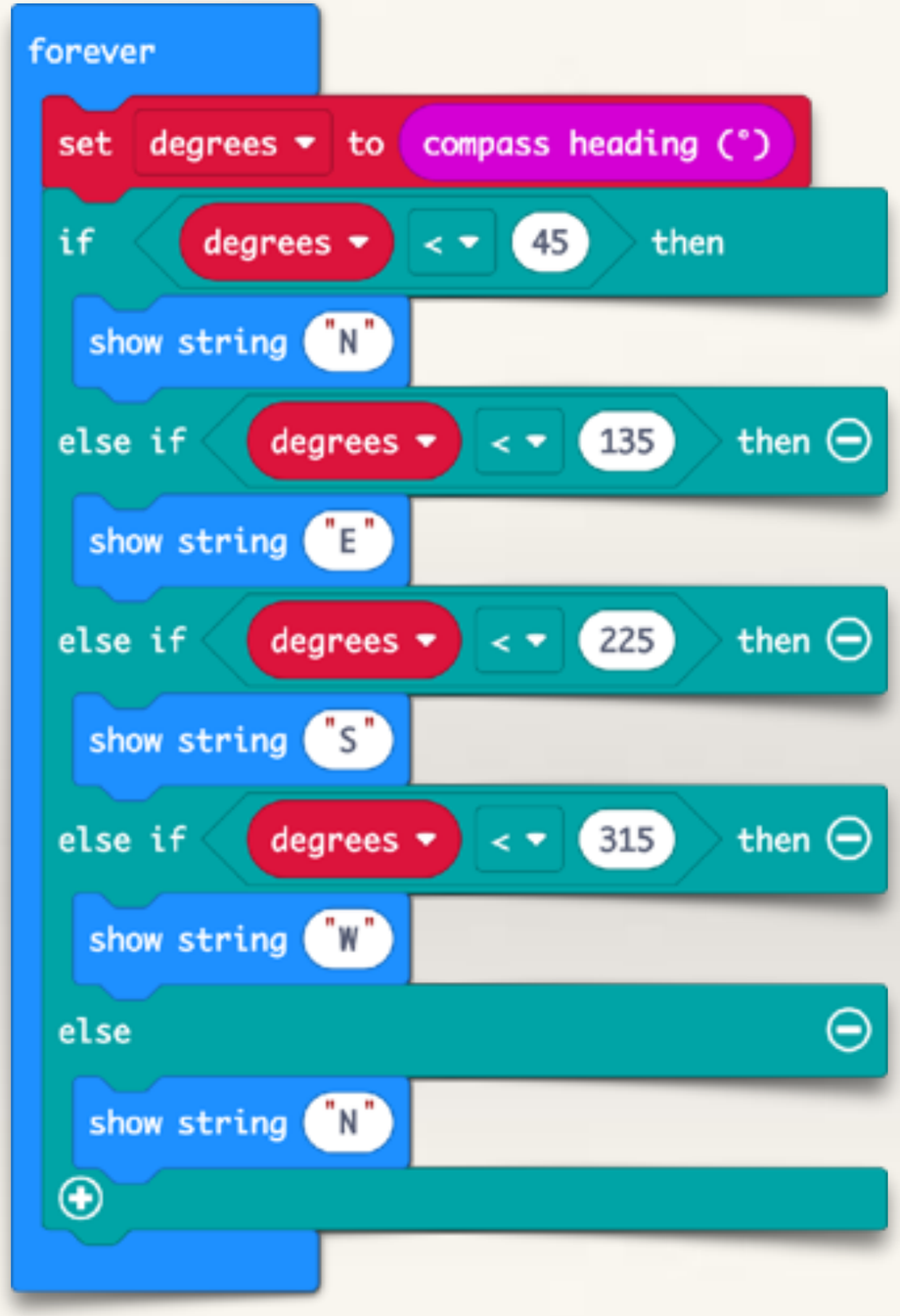
You can fine-tune the micro:bit compass by mapping the Compass Reading to different directions more accurately.

Here is how the reading shown by Compass Heading block map to different directions.



## Better Compass!

See the final code



You can calibrate the compass anytime by using the Calibrate Compass block inside Input > More

Project-7

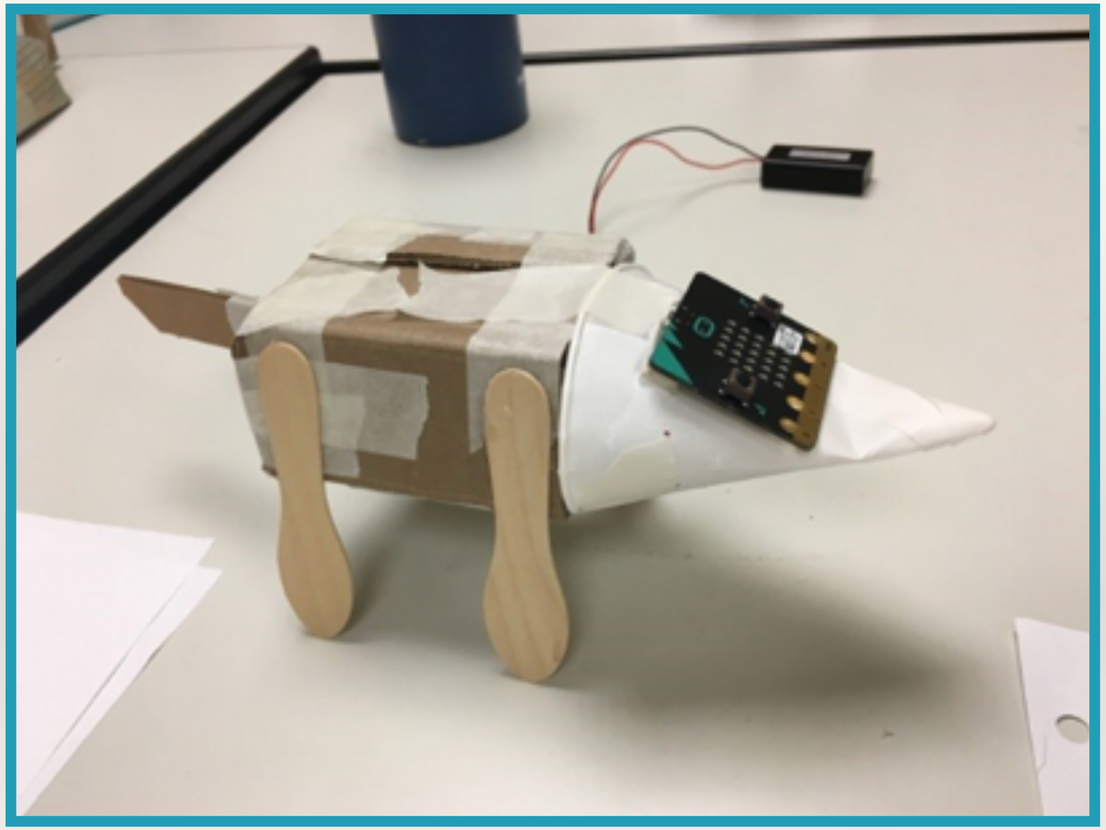
# MICROPET

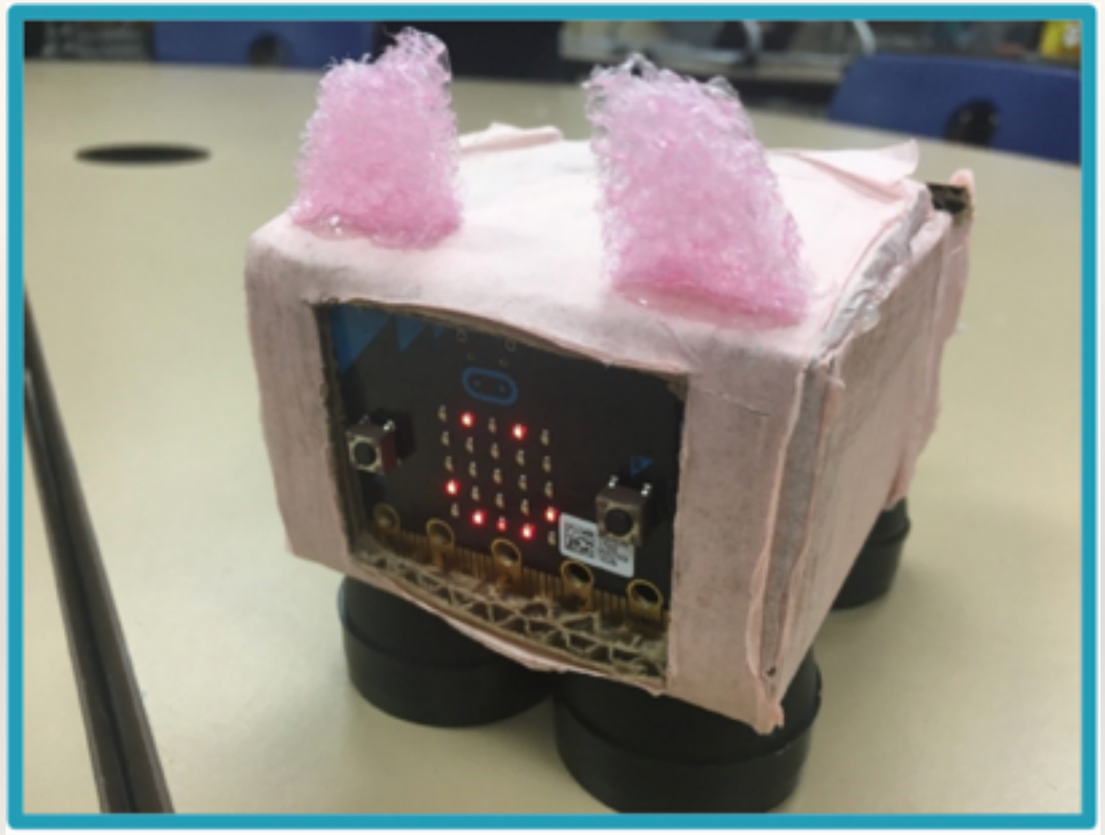
**Objective:** take an old toy or doll (or make your own) and use the onboard sensors on micro:bit to make an interactive pet.

**Problem:** fix the micro:bit (with a battery pack) on a toy/doll. Code the micro:bit such that the toy/doll behaves in an interactive say:

- On tilting the toy left or right, it should make different sounds and display different icons
- On touching the logo, make a giggle sound (as if the toy/doll likes being tickled).
- On pressing button-A, the toy/doll should say Hello and show a smily face
- Use your imagination to create different interactions

# Micropet Examples

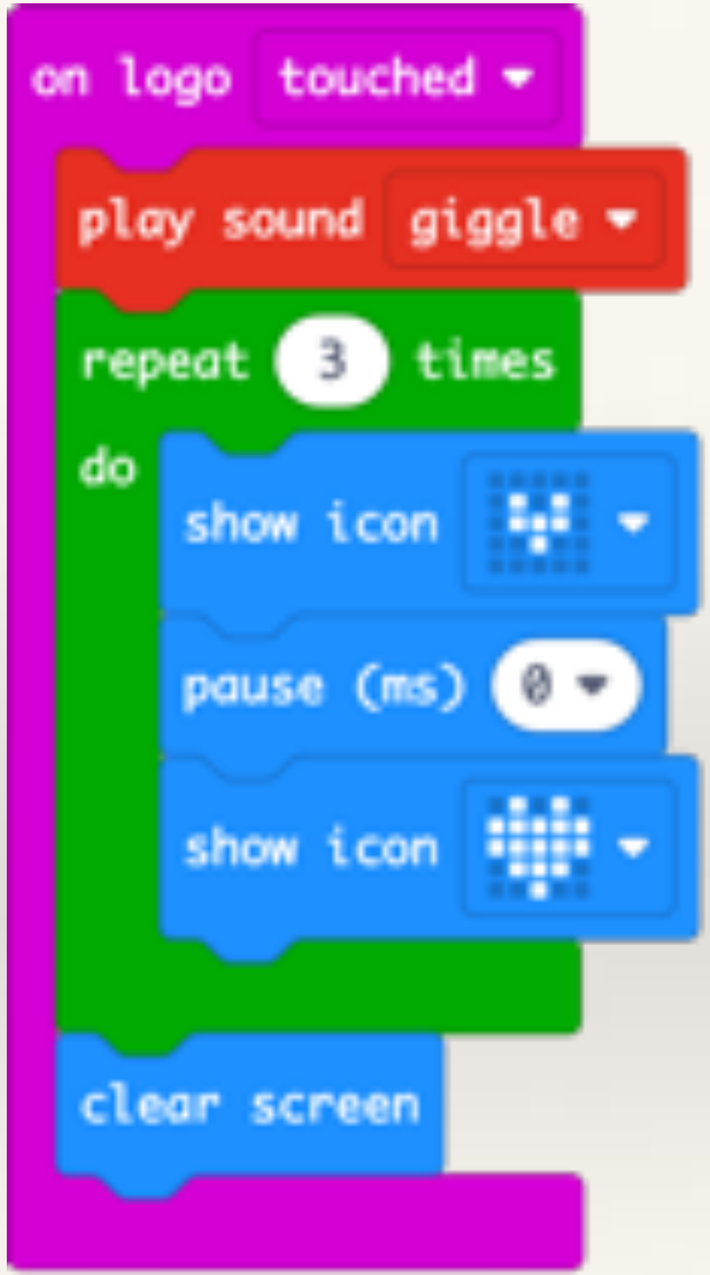


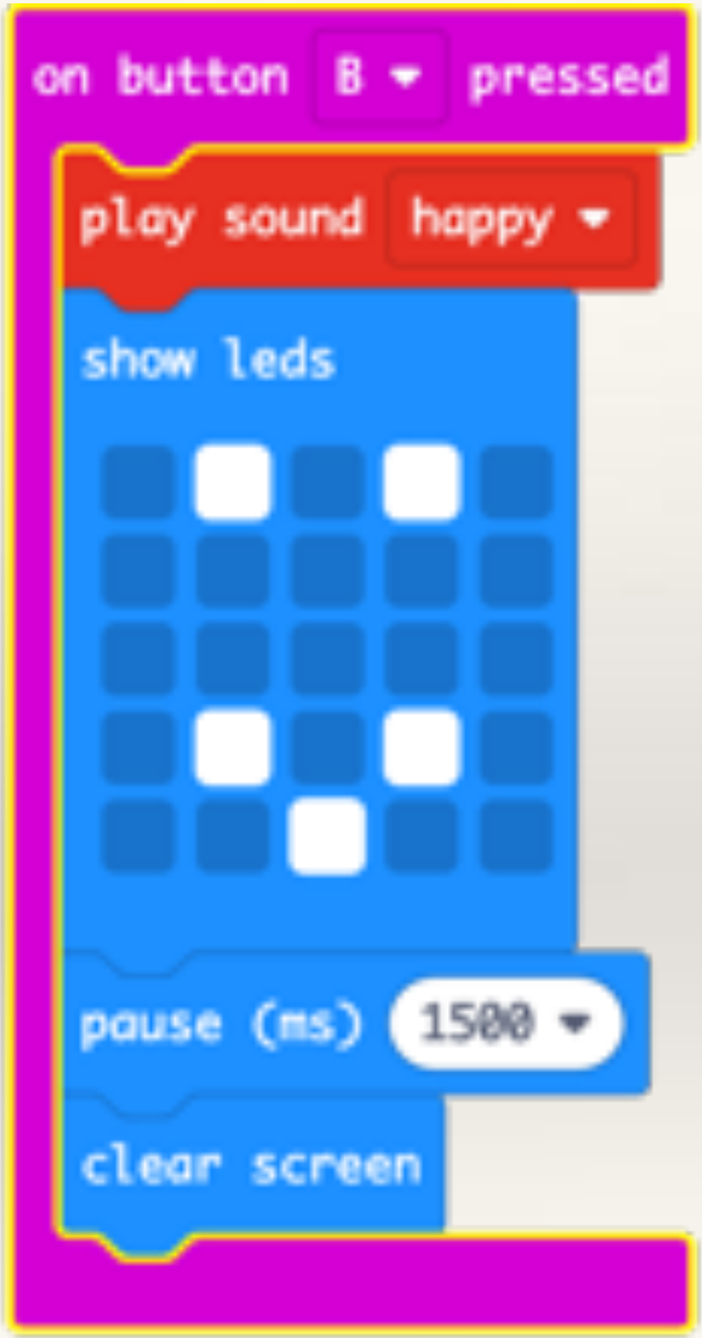


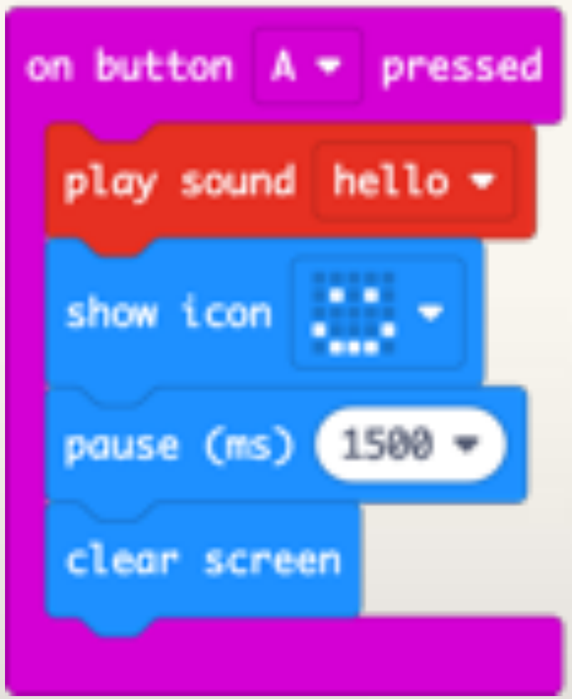
*Source: https://makecode.micro:bit.org/courses/csintro-educator*

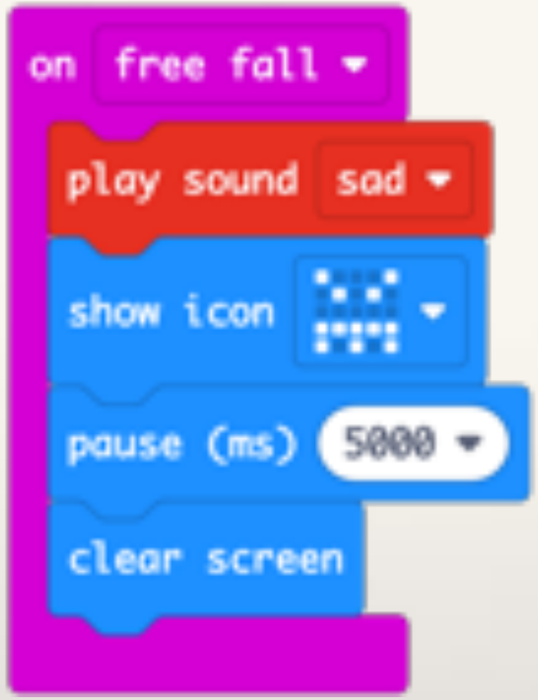
## Sample Code

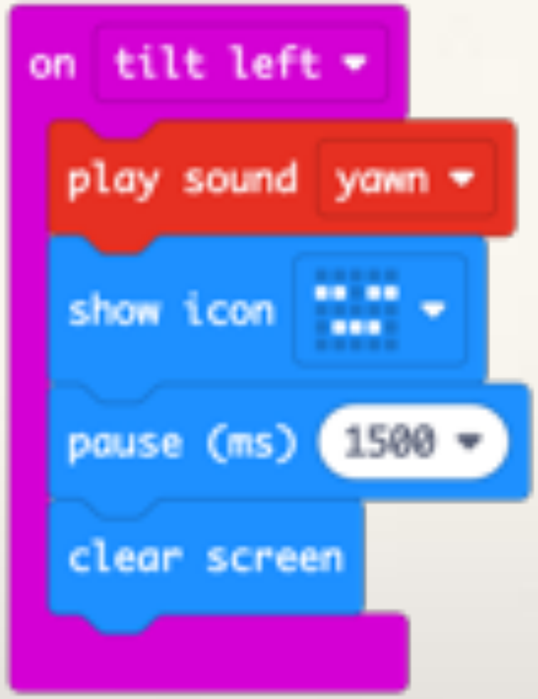
- On Logo Touched, the Micropet giggles and flashes the heart icon.
- On button-B pressed, the Micropet makes a happy sound and shows a smily icon.

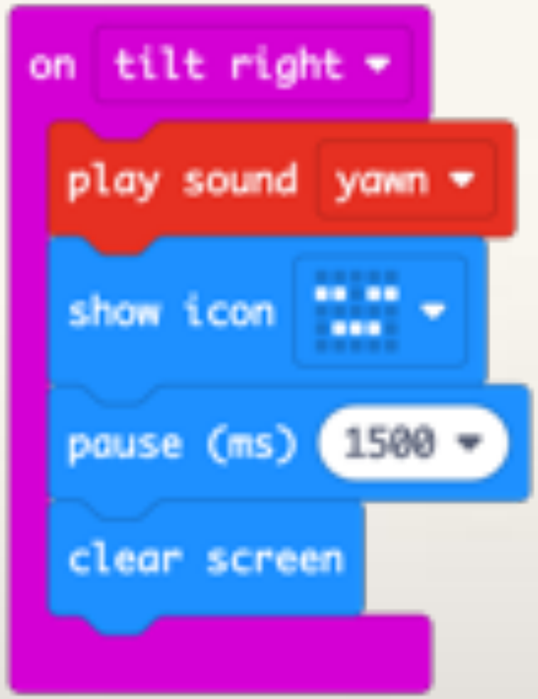





## More Sample Code

- On button-A pressed, the Micropet says Hello and shows a smily face.
- On Free Fall (under On Shake), the Micropet plays a sad sound and shows an angry face.
- On Tilt Left and Tilt Right, the Micropet makes a yawning sound and shows a sleepy face.
- Use your imagination to create more interactions.

Hope you enjoyed making these projects and in the process learnt more about the world of sensors and pocket computers.