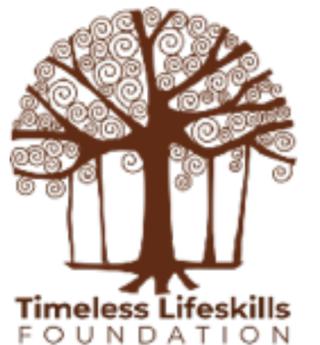


Micro:bit Advanced Level

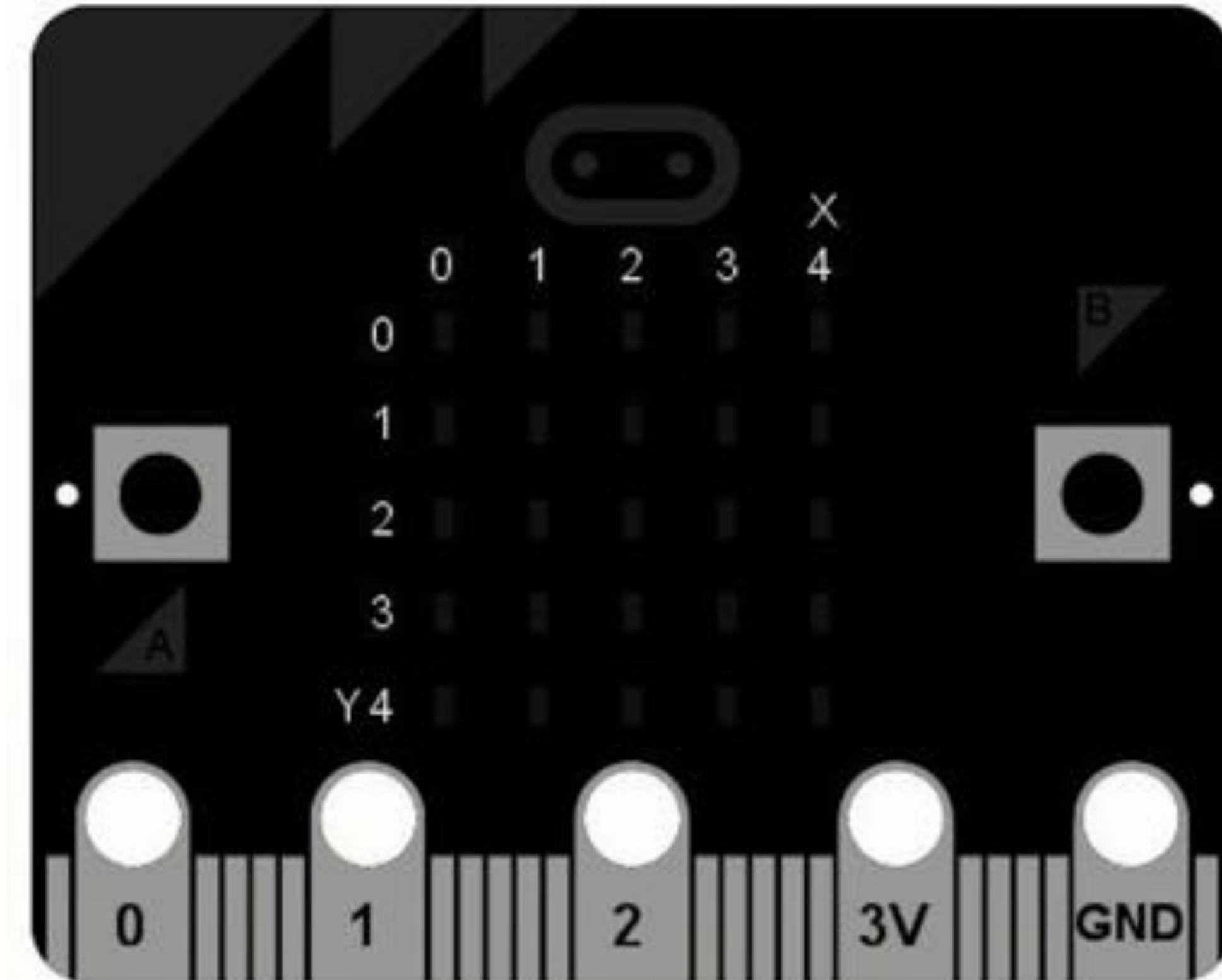
Instructor Guide



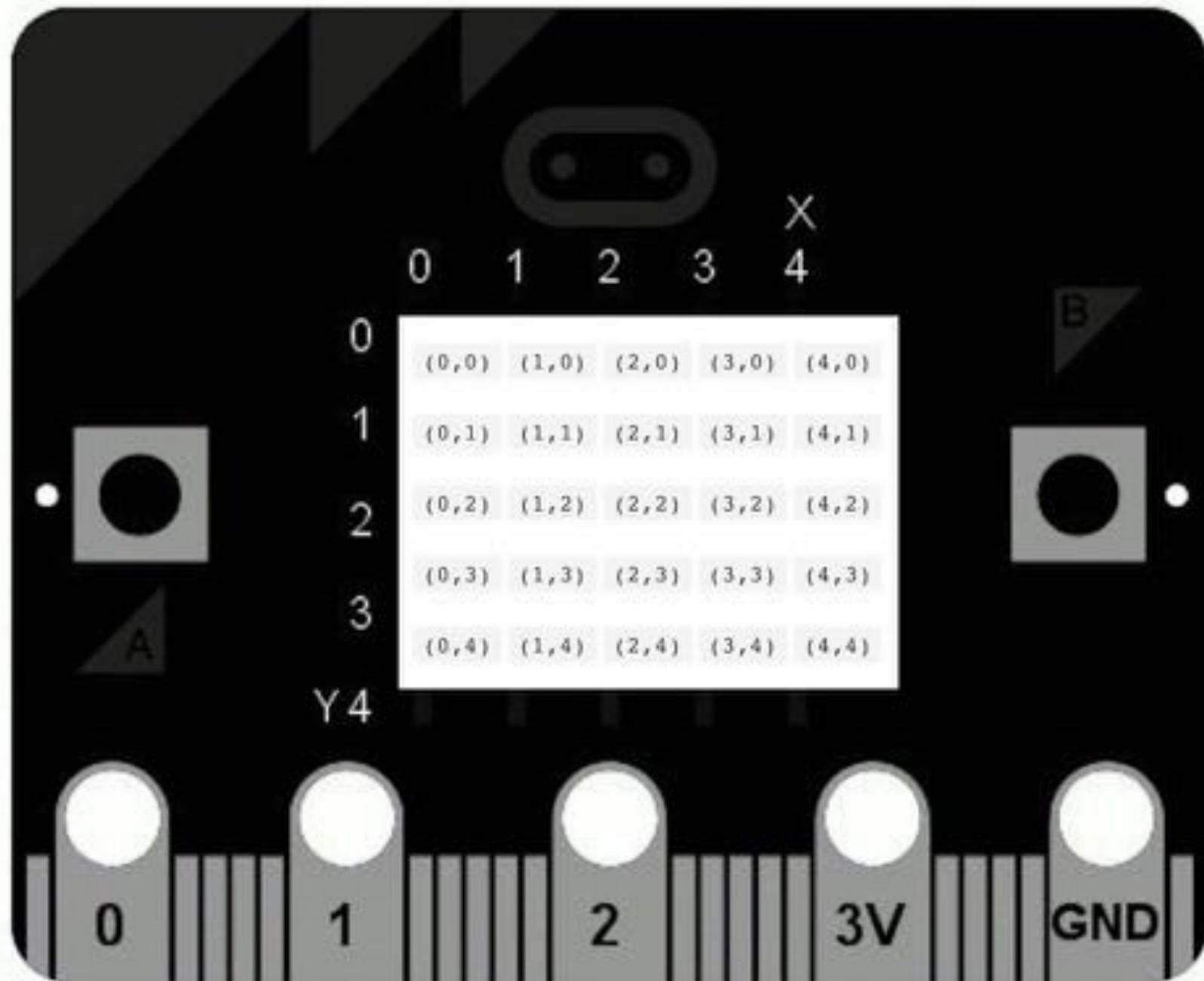
CONTENTS

1. Plotting LEDs
2. Bar Graph with LEDs
3. Radio
4. Step Counter
5. Neopixels
6. Smart Street Light (Tinkercad)
7. Smart Fan (Tinkercad)
8. Motor Driver Board
9. 8x8 LED Display
10. i2C LCD Display
11. MQ Sensors

Plotting LEDs



LEDs form a
Coordinate Plane



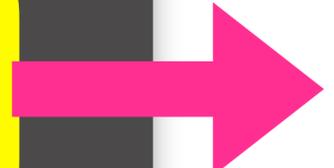
(x,y) coordinates of the 25 LEDs

Search... 🔍

- Basic
- Input
- Music
- Led**
- more
- Radio
- Loops
- Logic
- Variables

Led

- plot x 0 y 0
- toggle x 0 y 0
- unplot x 0 y 0
- point x 0 y 0
- plot bar graph of 0 up to 0



Switch LED (0,0) on



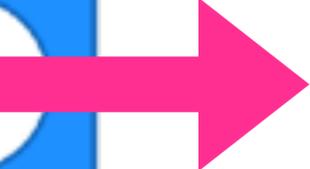
Switch LED (0,0) off

CHALLENGE

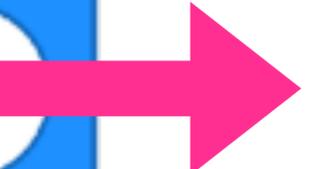
Make Middle LED Blink

- ◆ Centre LED, i.e. $x=2$, $y=2$, should be on for 1 second
- ◆ Then it should be off for 1 second
- ◆ This should repeat forever

```
forever
  plot x 2 y 2
  pause (ms) 500
  unplot x 2 y 2
  pause (ms) 500
```



Change value to change the rate of blinking



Change value to change the rate of blinking

CHALLENGE

Fun with LEDs

- ◆ On button-A pressed, the LEDs on the top row should light up one at a time.
- ◆ On button-B pressed, the centre LED should blink 5 times.
- ◆ Every time button-A+B is pressed, a different random pattern of LEDs should get created

```
on button A pressed
  plot x 0 y 0
  pause (ms) 500
  plot x 1 y 0
  pause (ms) 500
  plot x 2 y 0
  pause (ms) 500
  plot x 3 y 0
  pause (ms) 500
  plot x 4 y 0
  pause (ms) 1000
  clear screen
```

On button-A pressed, the LEDs on the top row should light up one at a time.

```
on button B pressed
  repeat 5 times
    do
      plot x 2 y 2
      pause (ms) 500
      unplot x 2 y 2
      pause (ms) 500
```

The image shows a Scratch script starting with an event block 'on button B pressed'. This is followed by a 'repeat 5 times' loop block. Inside the loop, there is a 'do' block containing four sub-blocks: 'plot x 2 y 2', 'pause (ms) 500', 'unplot x 2 y 2', and another 'pause (ms) 500' block.

On button-B pressed, the centre LED should blink 5 times

(using a loop or iteration)

```
on button A+B pressed
  repeat 25 times
    do
      plot x pick random 0 to 4 y pick random 0 to 4
      pause (ms) 100
    end
  end
  pause (ms) 1500
  clear screen
```

The image shows a Scratch script starting with an 'on button A+B pressed' event block. This is followed by a 'repeat 25 times' loop block. Inside the loop, there is a 'do' block containing three sub-blocks: a 'plot x pick random 0 to 4 y pick random 0 to 4' block, a 'pause (ms) 100' block, and another 'do' block. The second 'do' block is empty. After the loop, there is a 'pause (ms) 1500' block and a 'clear screen' block.

Every time button-A+B is pressed, a different random pattern of LEDs should get created.

(Loop + Random Number Math Operator)

Plotting a Bar Graph

Search... 🔍

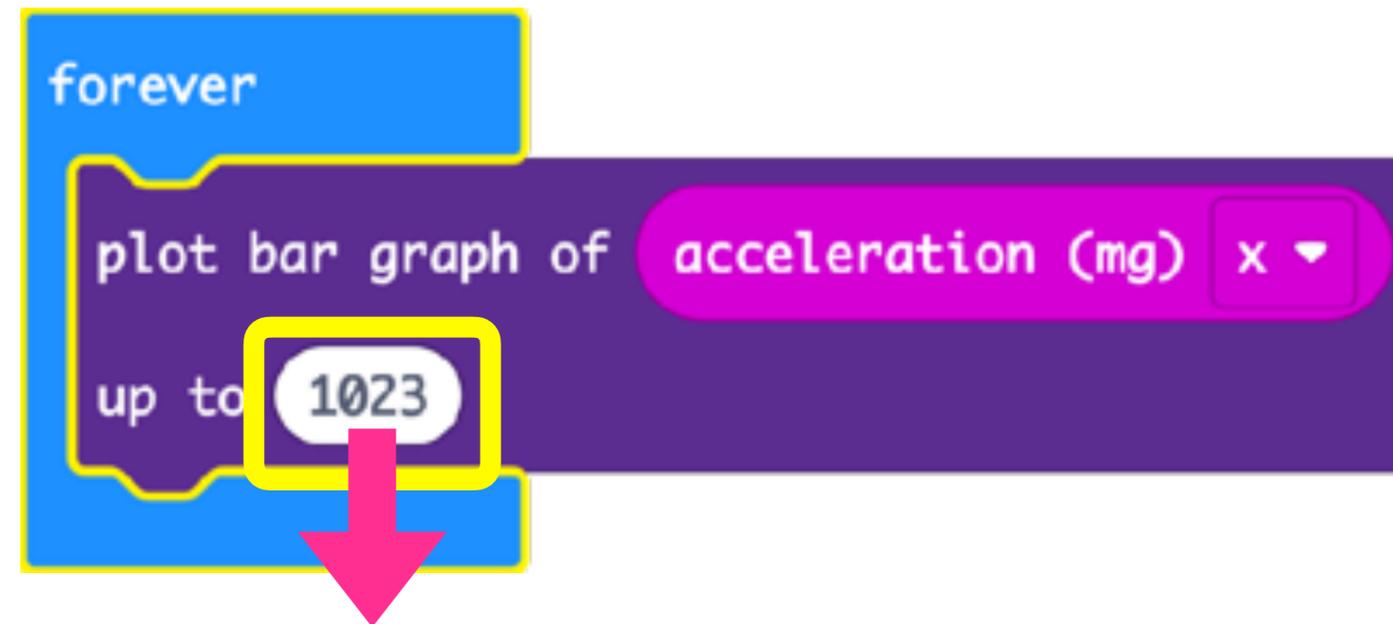
- Basic
- Input
- Music
- Led**
- more
- Radio
- Loops
- Logic
- Variables

Led

- plot x 0 y 0
- toggle x 0 y 0
- unplot x 0 y 0
- point x 0 y 0
- plot bar graph of 0 up to 0**

Makes the LEDs display a bar graph for a number value (parameter).

Plotting Bar Graph of Acceleration

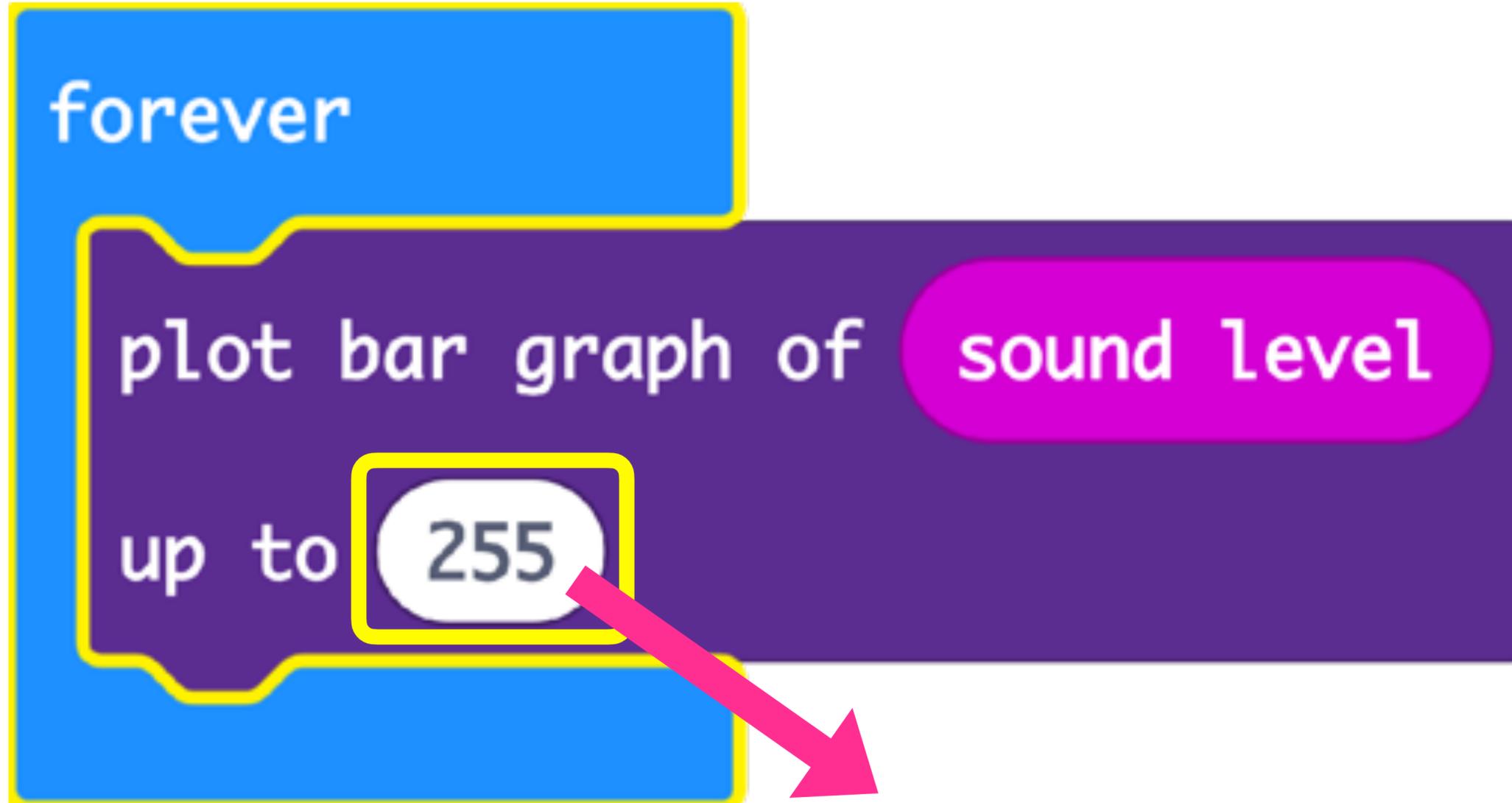


(+)1023 to (-)1023 is the maximum range of acceleration

CHALLENGE

Loudest Singer!

- ◆ The LEDs on the microbit should plot the graph of the sound level.
- ◆ Find out who can sing or blow the loudest.



255 is the maximum level of sound detected on the microbit.

Radio

```
on start
  radio set group 100

on button A pressed
  radio send string "How are you?"
```

Transmitting
Microbit Code

```
on start
  radio set group 100

on radio received receivedString
  show string receivedString

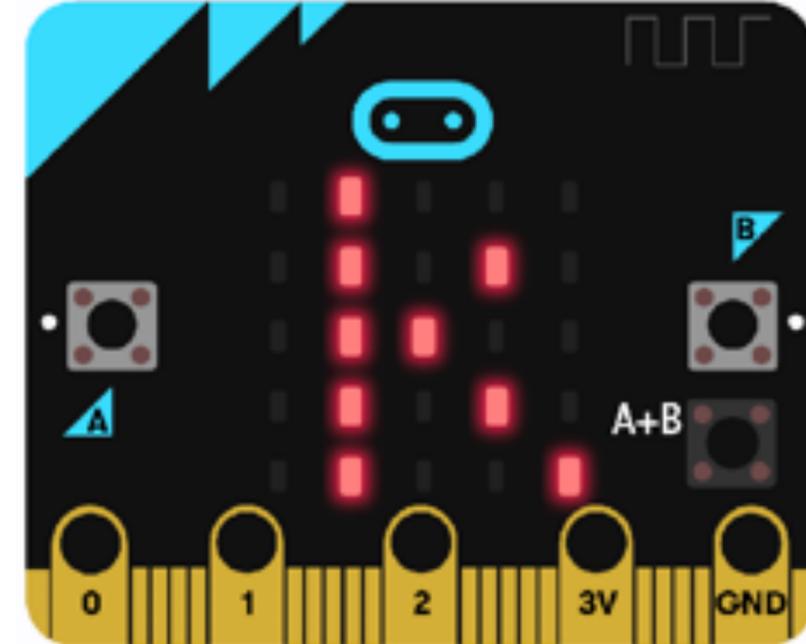
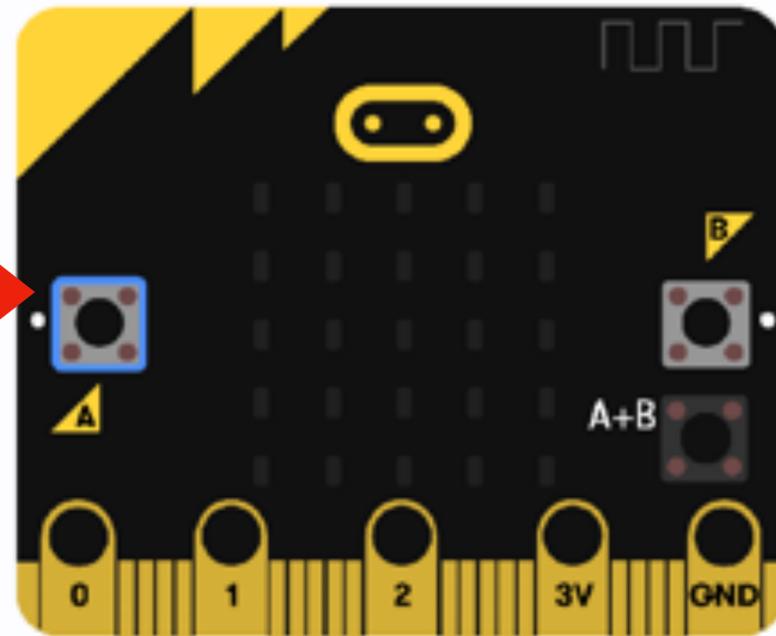
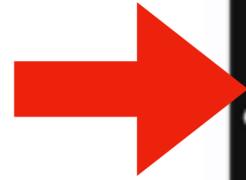
on button B pressed
  radio send string "ok"

on button A+B pressed
  radio send string "sos"
```

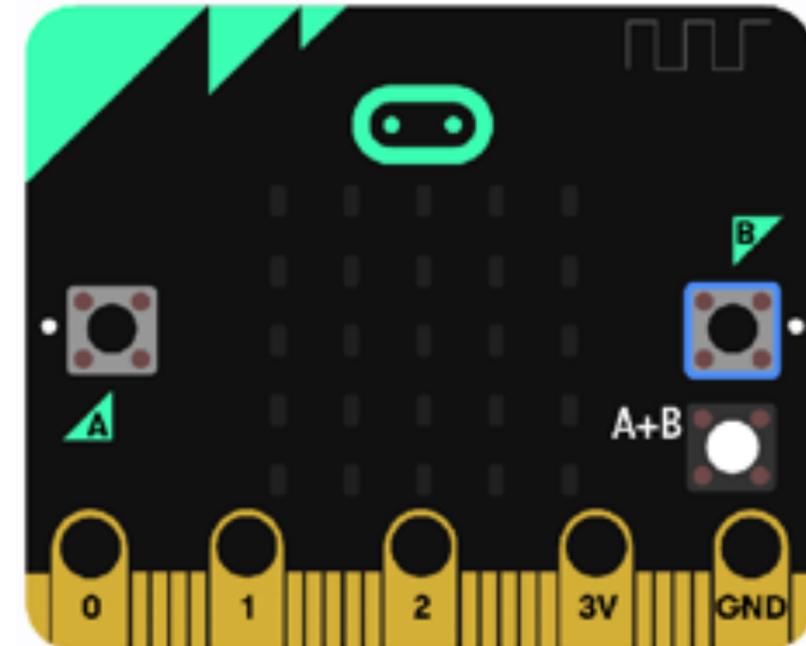
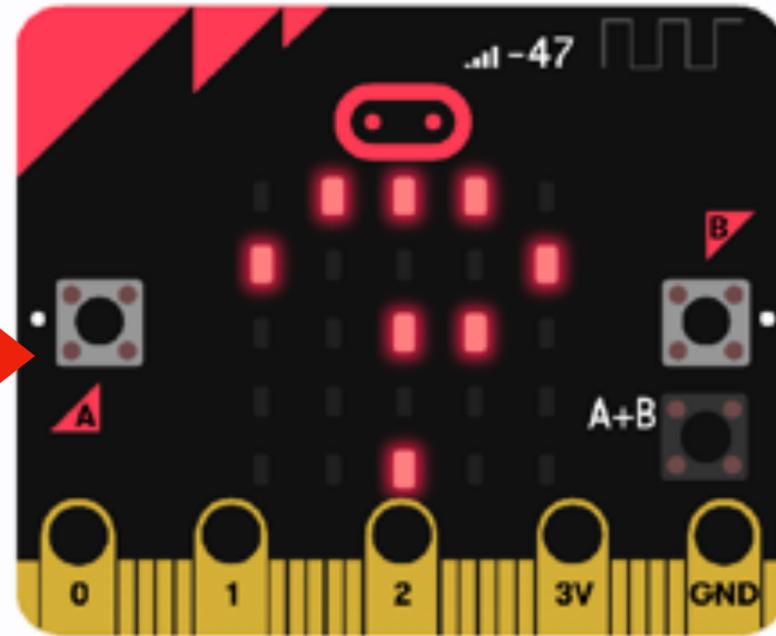
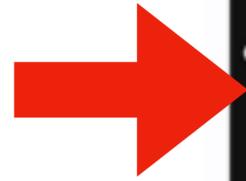
Receiving
Microbit Code

Both Microbits have to set to the same Radio Group

Transmitting
Microbit



Receiving
Microbit



Radio - Remote Control Bot

Transmitting
Microbit Code

```
on start  
  radio set group 100
```

```
on tilt left ▼  
  radio send string "L"
```

```
on tilt right ▼  
  radio send string "R"
```

```
on button A ▼ pressed  
  radio send string "F"
```

```
on button B ▼ pressed  
  radio send string "B"
```

```
on button A+B ▼ pressed  
  radio send string "S"
```

Receiving Microbit Code

```
on radio received receivedString
  if receivedString = "F" then
    motor 1 on direction forward speed 100
    motor 2 on direction forward speed 100
  else if receivedString = "B" then
    motor 1 on direction reverse speed 50
    motor 2 on direction reverse speed 50
  else if receivedString = "L" then
    motor 1 on direction forward speed 75
    motor 2 on direction forward speed 25
  else if receivedString = "R" then
    motor 1 on direction forward speed 25
    motor 2 on direction forward speed 75
  else if receivedString = "S" then
    turn off motor 1
    turn off motor 2
```

```
on start
  radio set group 100
```

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 Microbit to make a Step Counter.

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

Problem:

- If we walk with a Microbit tied to our shoe, on every step the Microbit 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

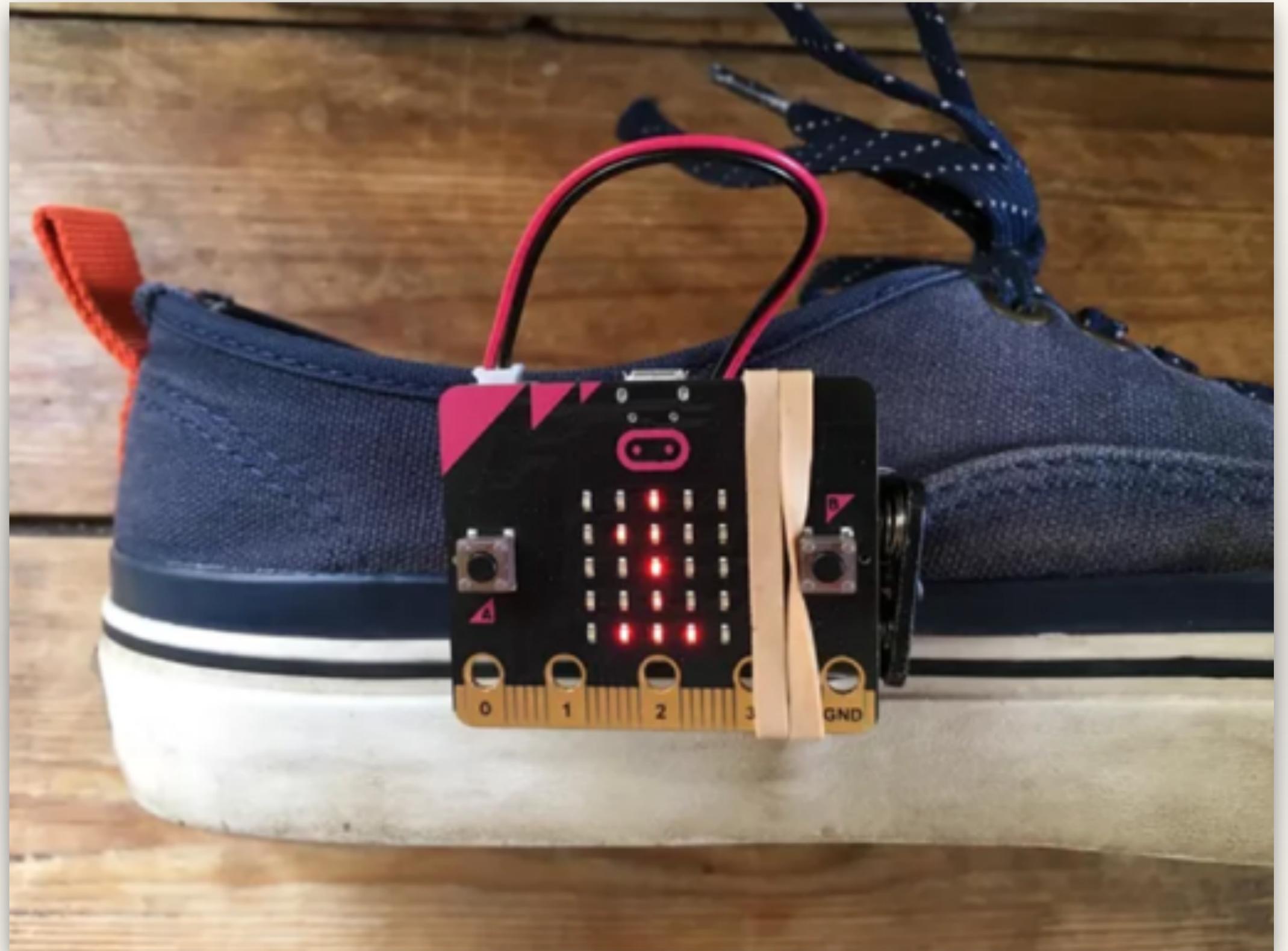
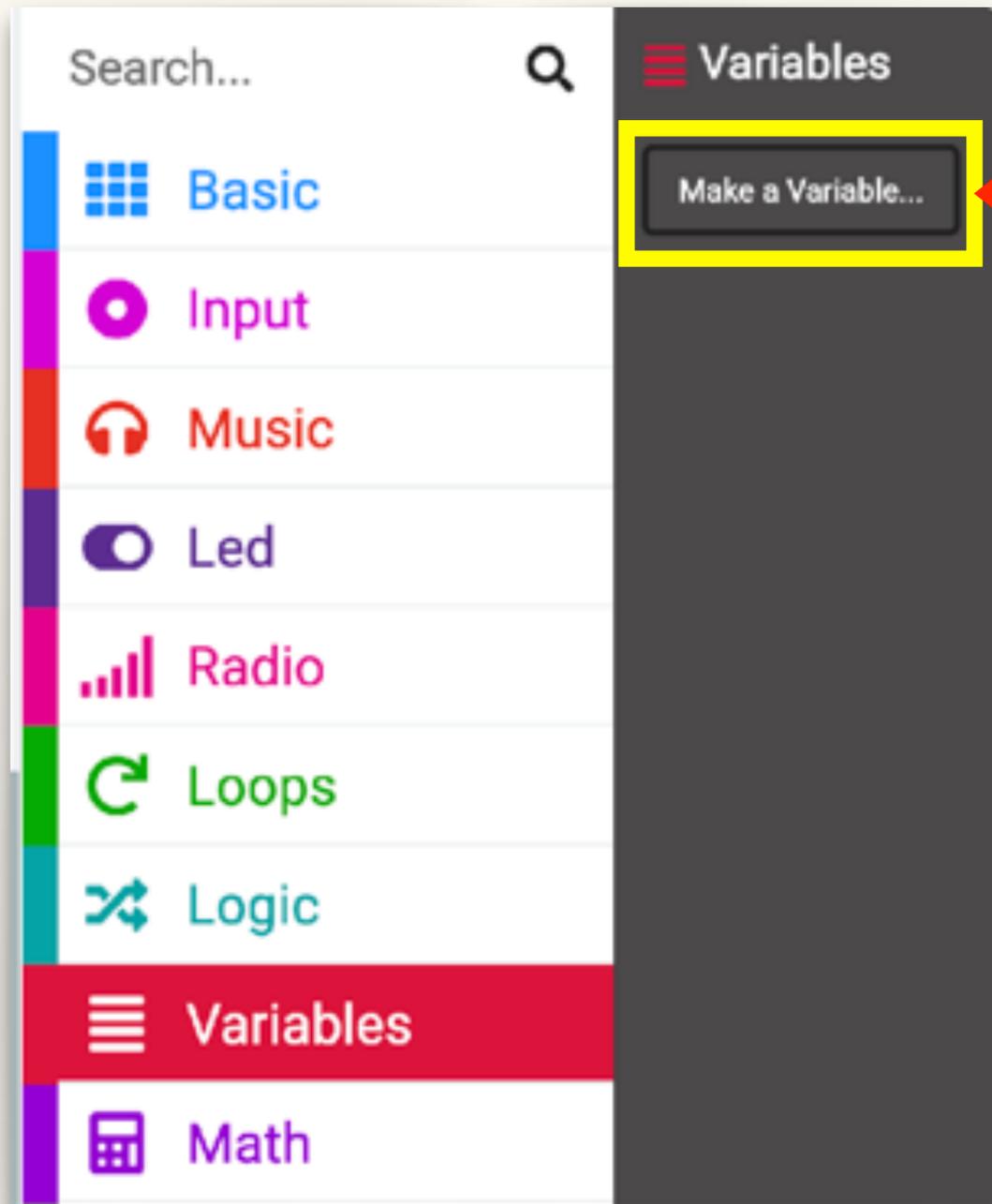


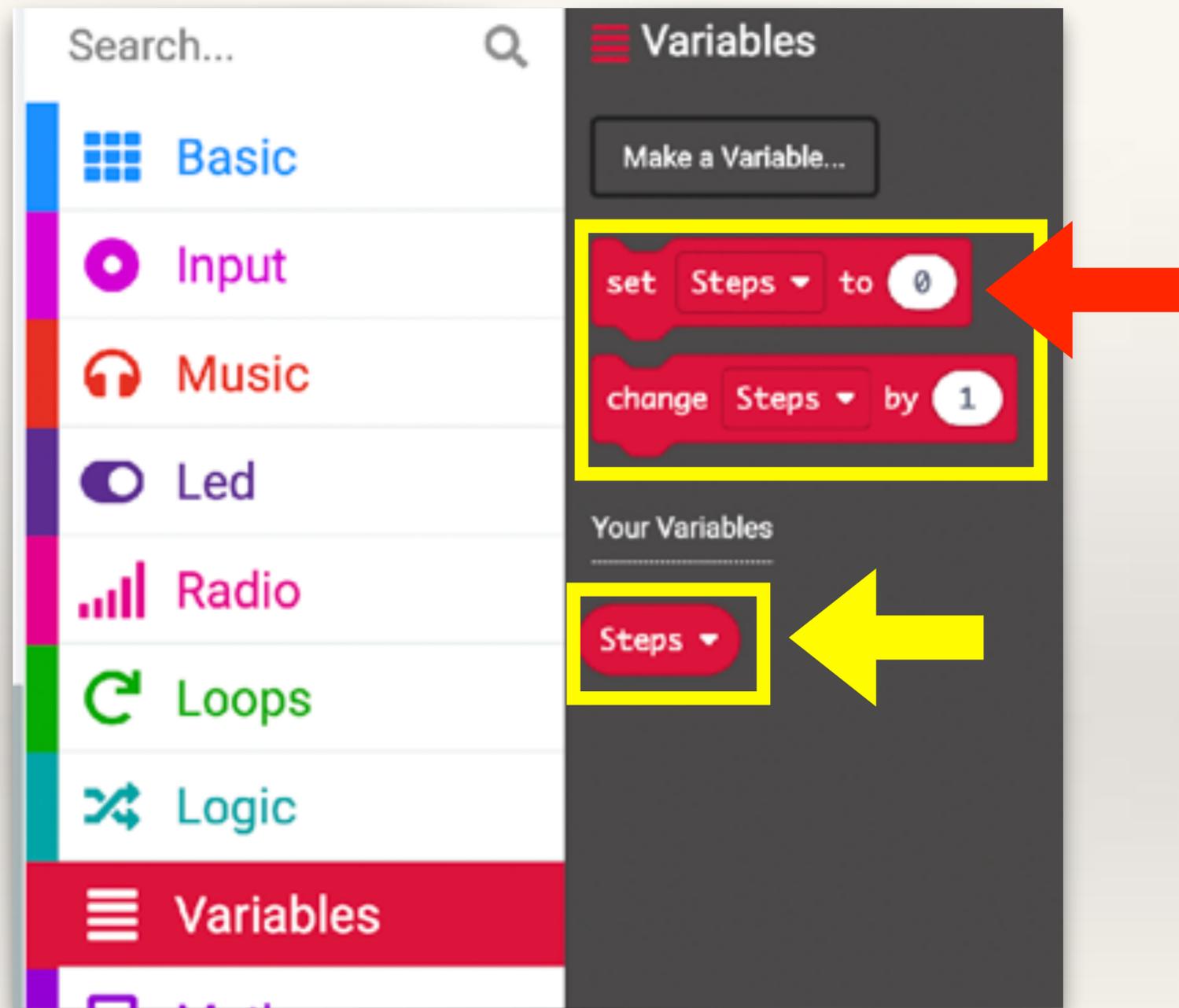
Image Source: <https://microbit.org/projects/make-it-code-it/sensitive-step-counter/>



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

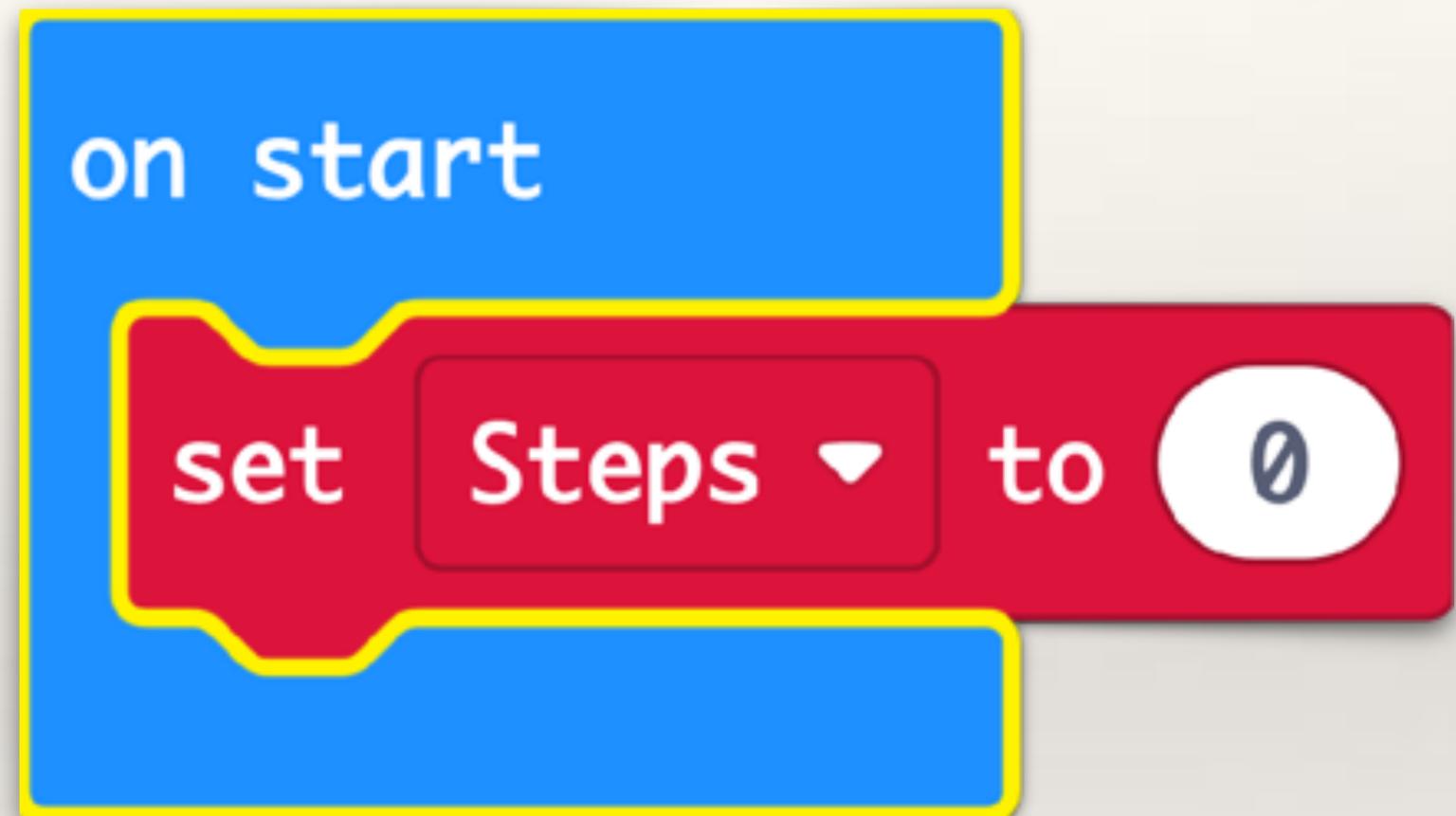
Once a new variable is created, new blocks will appear under 'Variables' - Set, Change, and the new variable (in this case 'Steps').

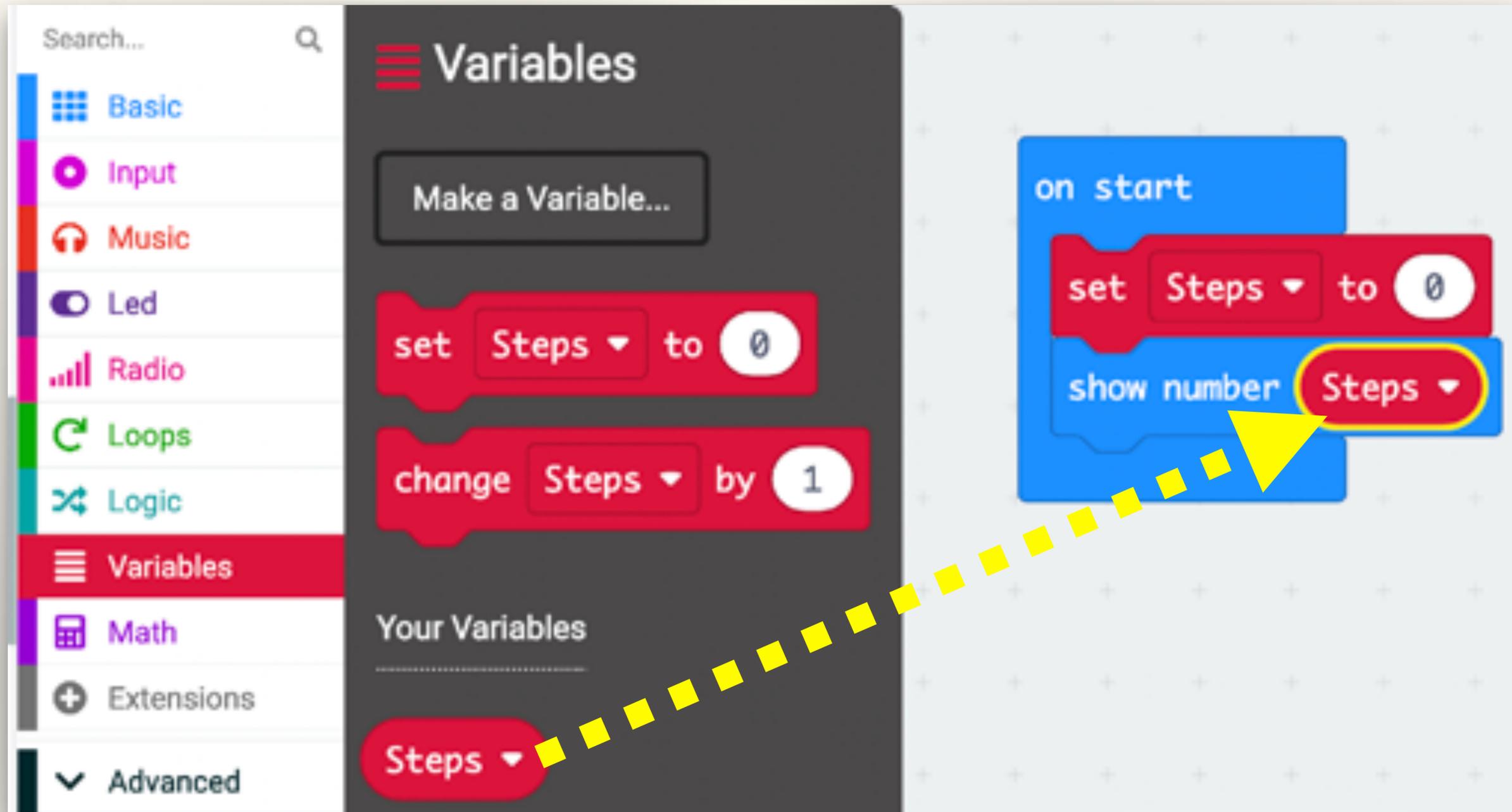
Drag out the 'Set Number' block and 'Steps' variable.



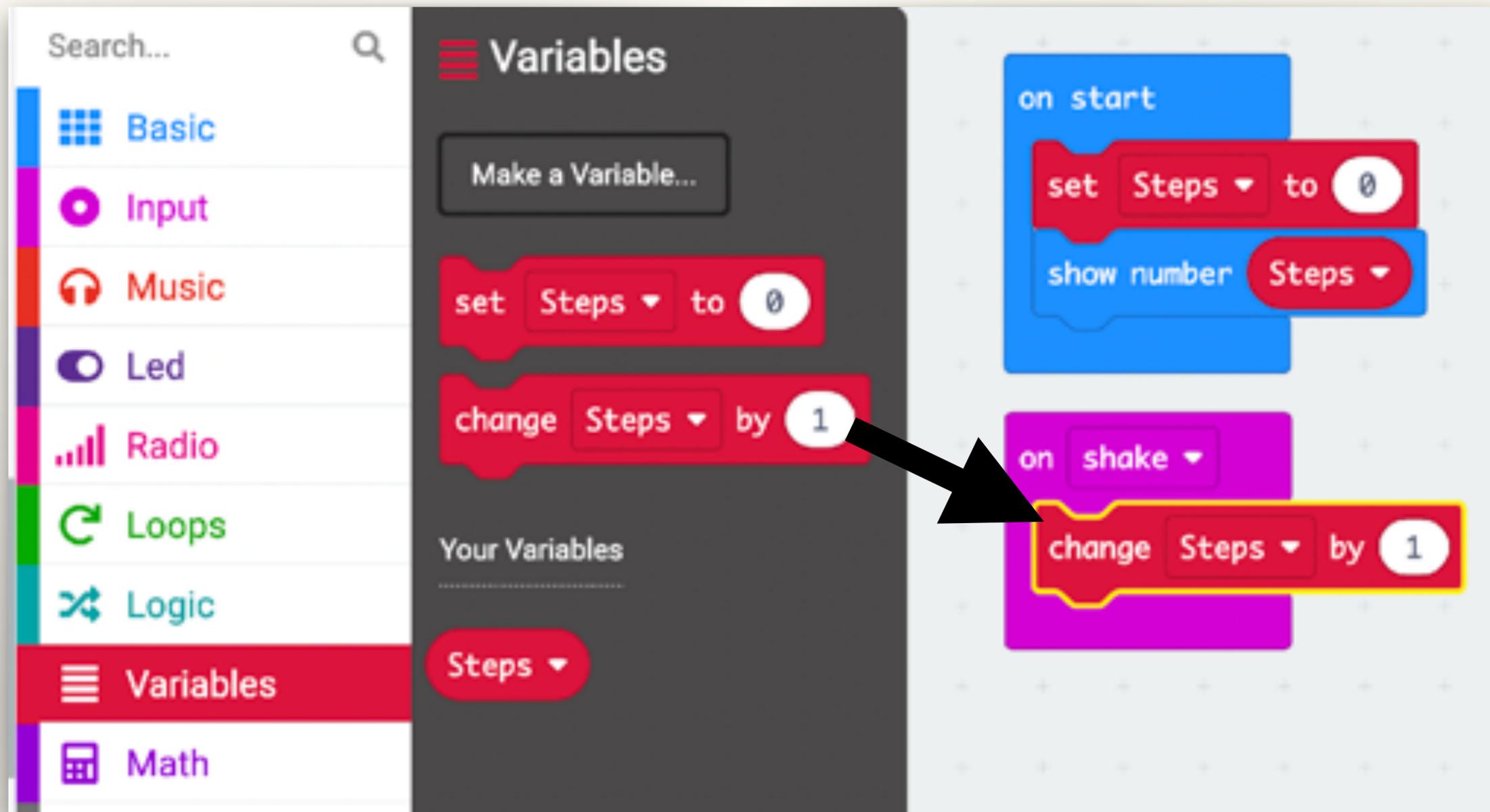
From 'Basic' drag out an 'On Start' block and put the 'Set Steps' command inside 'On Start' block.

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 microbit gets shaken, the value of the variable called Steps will get incremented by 1.

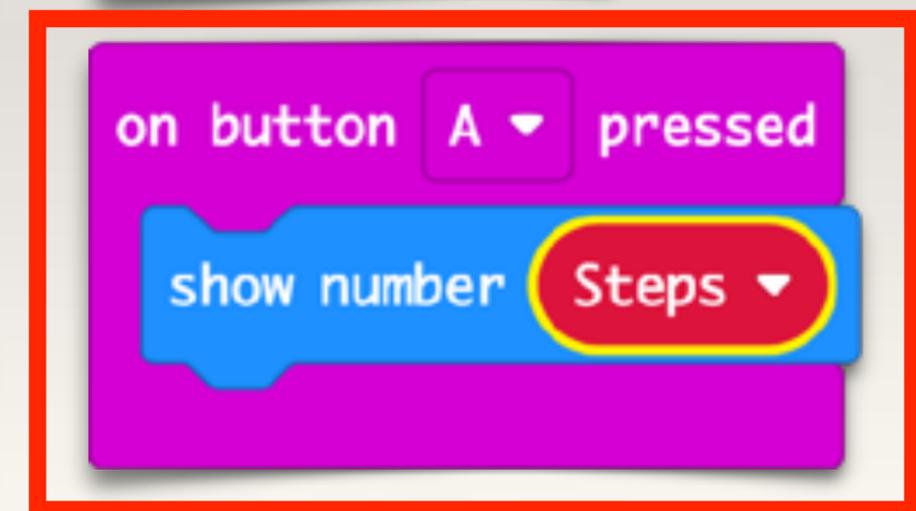
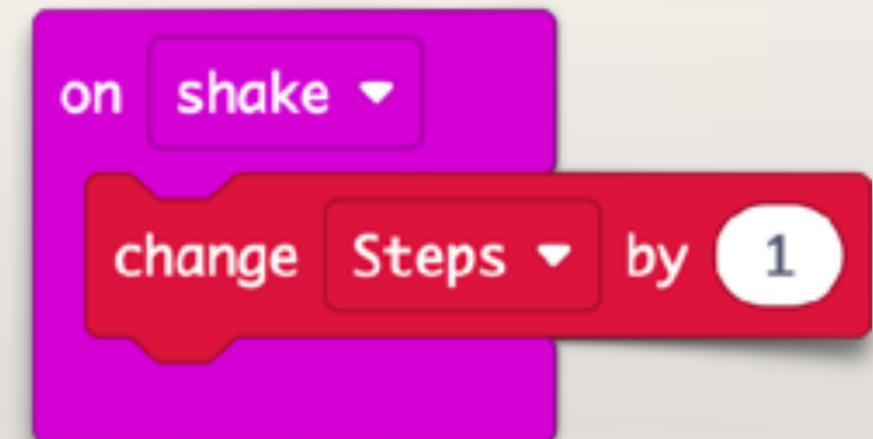
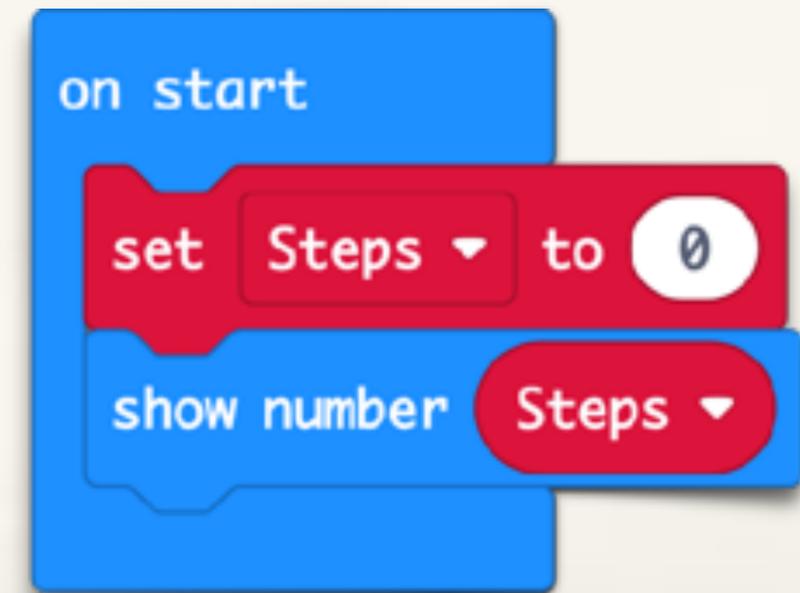
From 'Input' drag out an 'On button-A Pressed' event block.

From 'Basic' drag out a 'Show Number' block.

Put the variable 'Steps' inside 'Show Number' command.

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

Our basic Microbit Step Counter is ready. But we can make it look better.

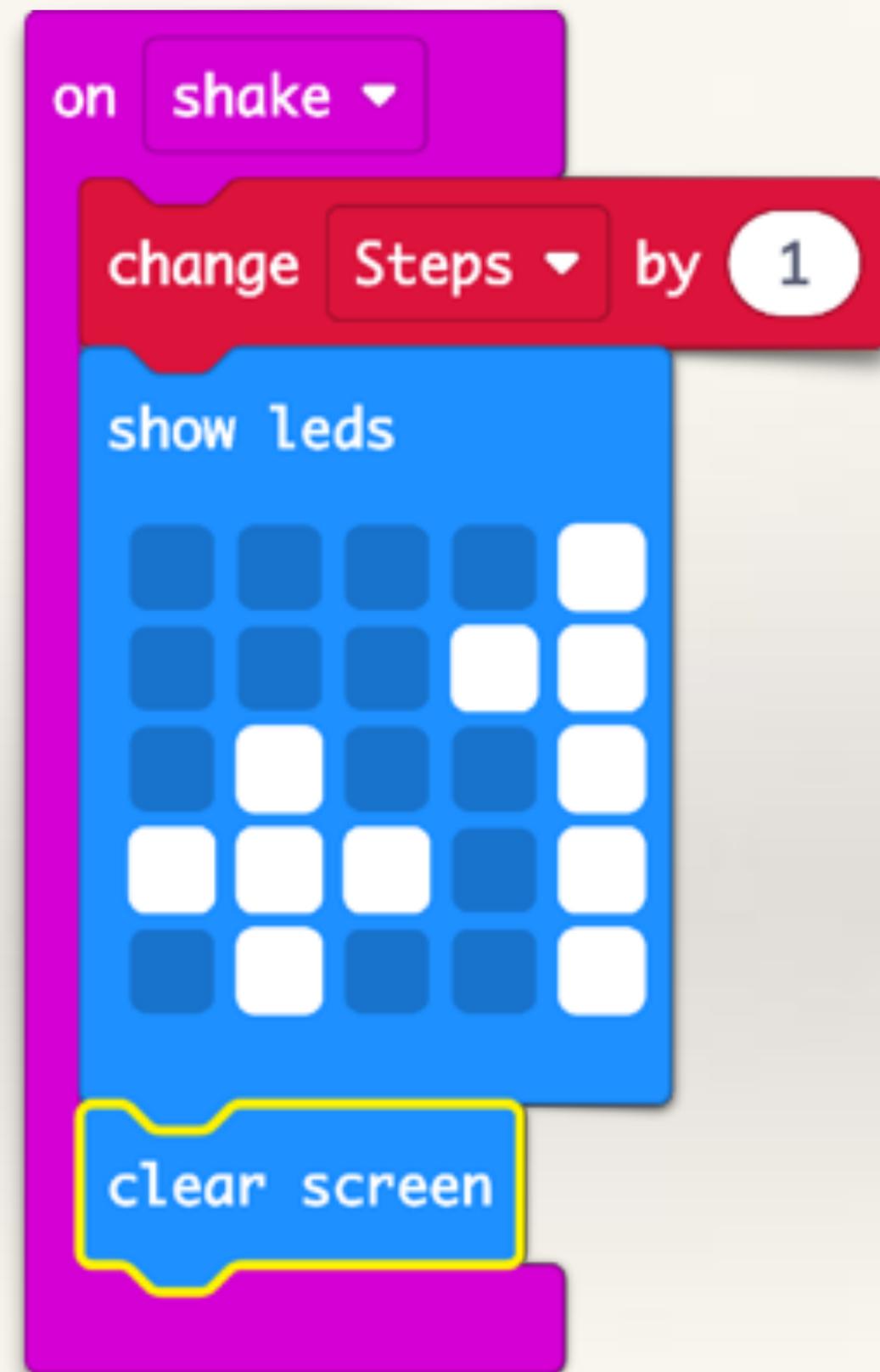


From 'Basic' drag out a 'Show LEDs' command. Put it in the On Shake block, under 'Change Steps' command.

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 microbit detects a shake, it will display "+1" briefly.



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

Whenever you take a step, the microbit 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

```
on start
  set Steps to 0
  show number Steps
```

```
on button A pressed
  show number Steps
```

```
on shake
  change Steps by 1
  show leds
  clear screen
```

Complete code for Microbit Step Counter

Neopixels

Neopixel

- Lights and Display
- Software
- Science
- Robotics
- Gaming
- Networking

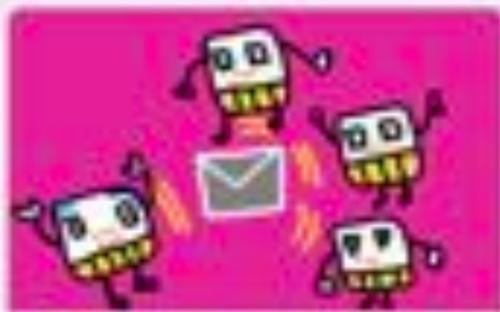
Recommended Local

Import File



datalogger
Data logging to flash memory.
micro:bit (V2) only.

[Learn More](#)



radio-broadcast
Adds new blocks for message communication in the radio category.

[Learn More](#)



servo
A micro-servo library.

[Learn More](#)



neopixel
Adafruit NeoPixel driver.

[Learn More](#)



microturtle
A LOGO-like turtle library.

[Learn More](#)



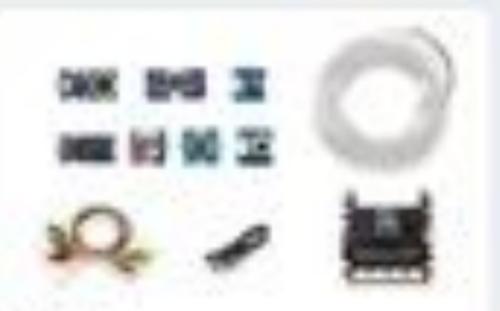
sonar
A MakeCode package to use sonar sensors.



kitronik-servo-lite
Blocks to simplify using Kitronik Servo Lite board in PXT.



kitronik-motor-driver
Blocks to simplify using Kitronik products in PXT.



Grove
A Microsoft MakeCode package for Seeed Studio Grove module.

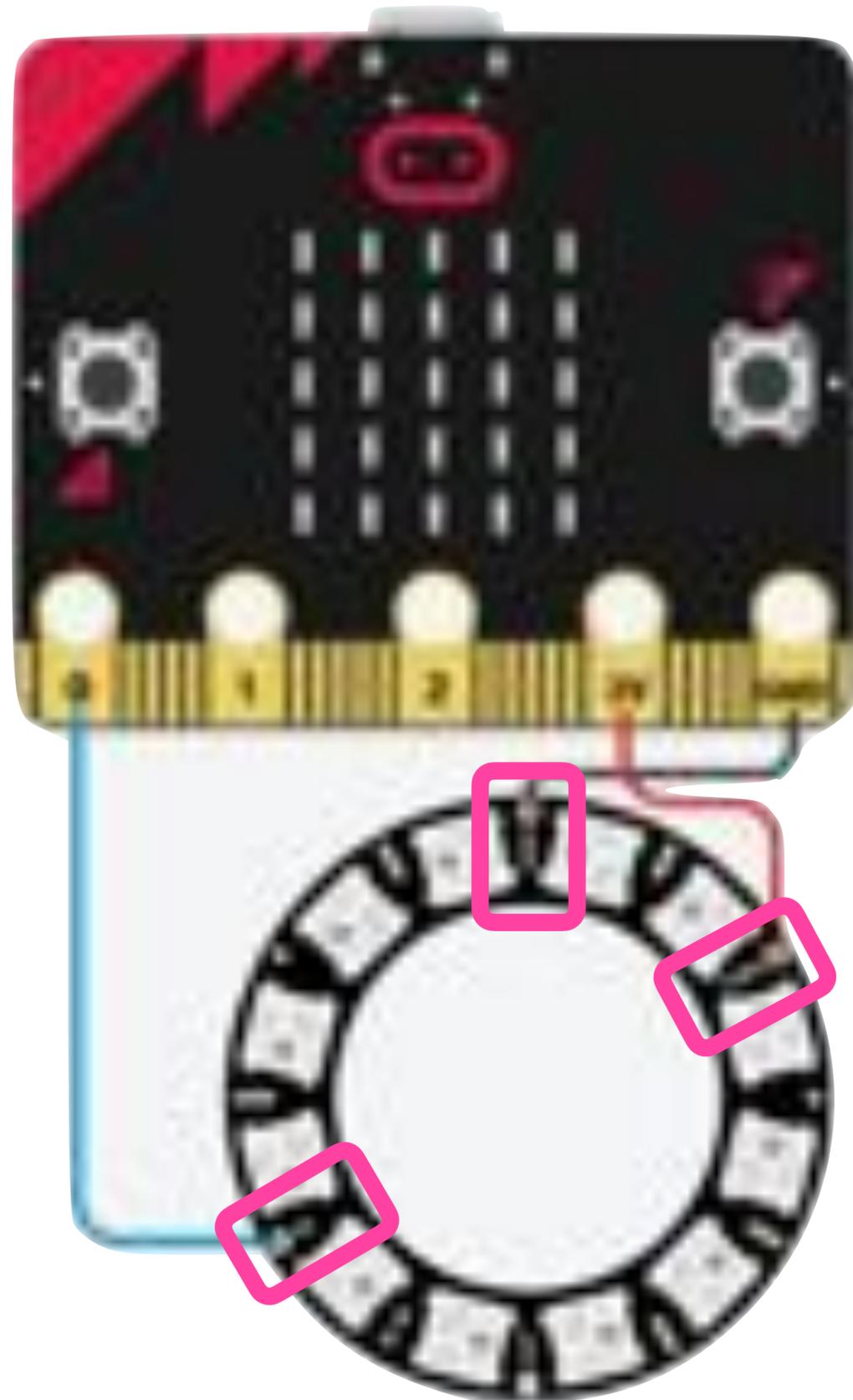


Environment and Science IoT
Environment and Science IoT Kit for micro:bit.

Neopixel Programming Blocks

The image shows a screenshot of a block-based programming environment's library for Neopixel. On the left is a sidebar with a search bar and a list of categories: Basic, Input, Music, Led, Radio, Loops, Logic, Variables, Math, Neopixel (highlighted), more, Extensions, and Advanced. The main workspace on the right is titled 'Neopixel' and contains several programming blocks:

- set strip2** to NeoPixel at pin P0 with 24 leds as RGB (GRB format)
- set range** to strip range from 0 with 4 leds
- strip** show rainbow from 1 to 360
- strip** show color red
- strip** show bar graph of 0 up to 255
- strip** show
- strip** clear
- hue 0 saturation 0 luminosity 0
- strip** shift pixels by 1
- strip** rotate pixels by 1



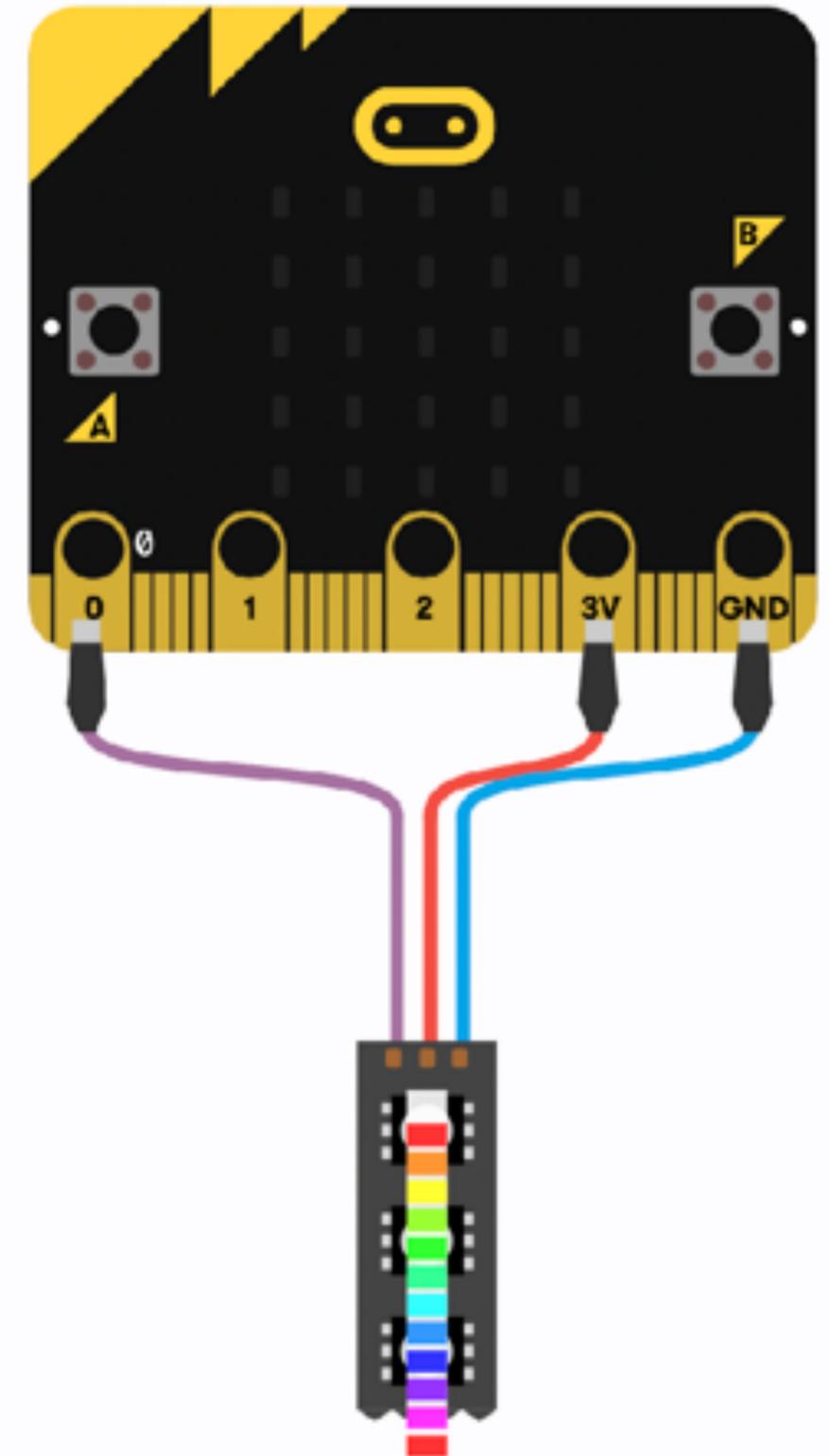
Connecting the Neopixels

For 8-12 neopixels, you can connect the strip directly to the microbit. If you want to connect more neopixels, it may be safer to use an external power supply for the neopixel strip.

Direct to Microbit

- Connect DI (input) pad on the neopixel to pin 0, 1 or 2 on the microbit
- Connect PWR pad on the neopixel to 3V pin on the microbit
- Connect GND pad on the neopixel to ground pin on the microbit

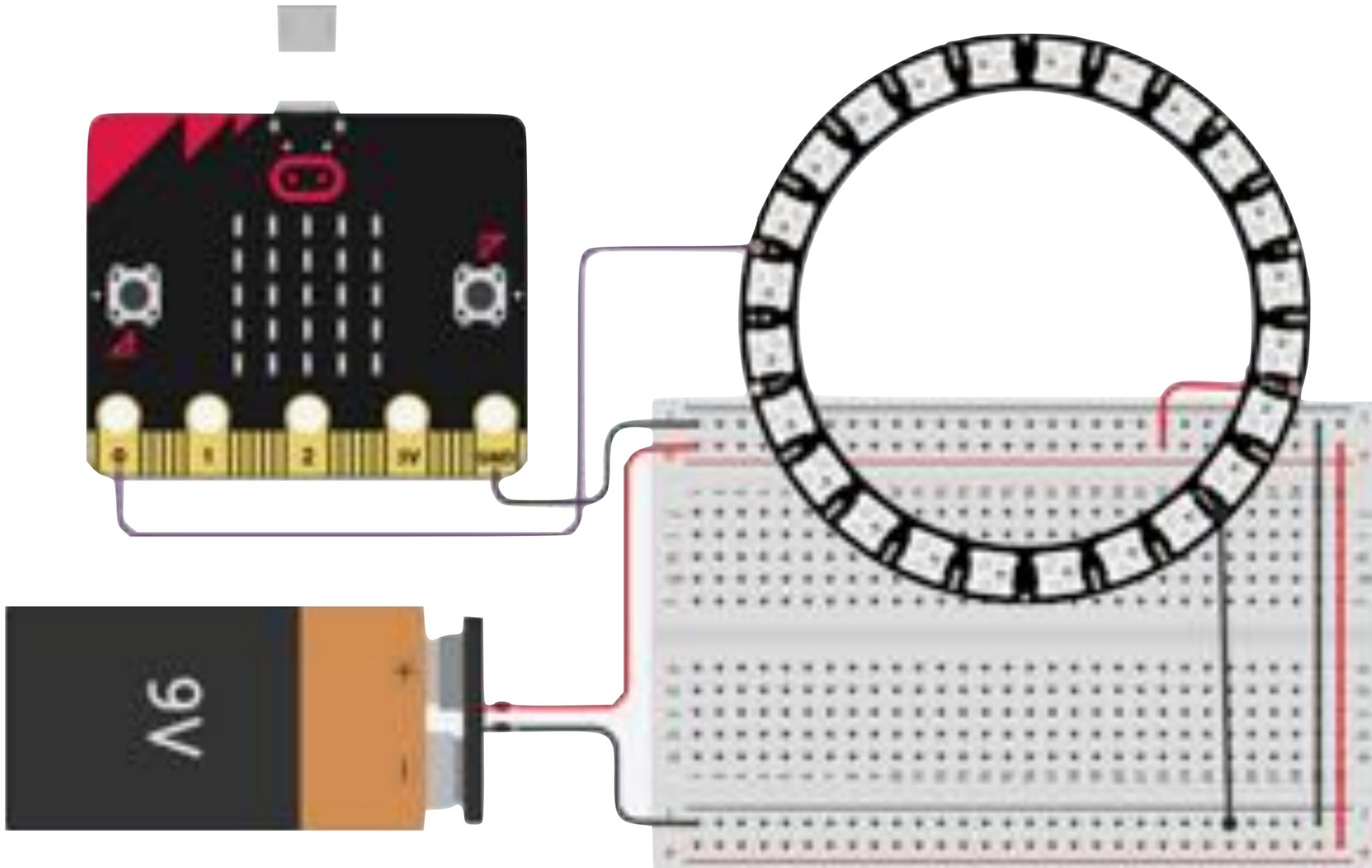
```
on start
  set strip to NeoPixel at pin P0 with 12 leds as RGB (GRB format)
  strip show rainbow from 1 to 360
```



1. Initialise the Neopixels with Set Strip command
2. Define the number of pixels your strip has (in this case it is 12)
3. Drag out Show Rainbow command
4. Transfer your code to the microbit, see result

Connecting the Neopixels to a battery pack / power supply

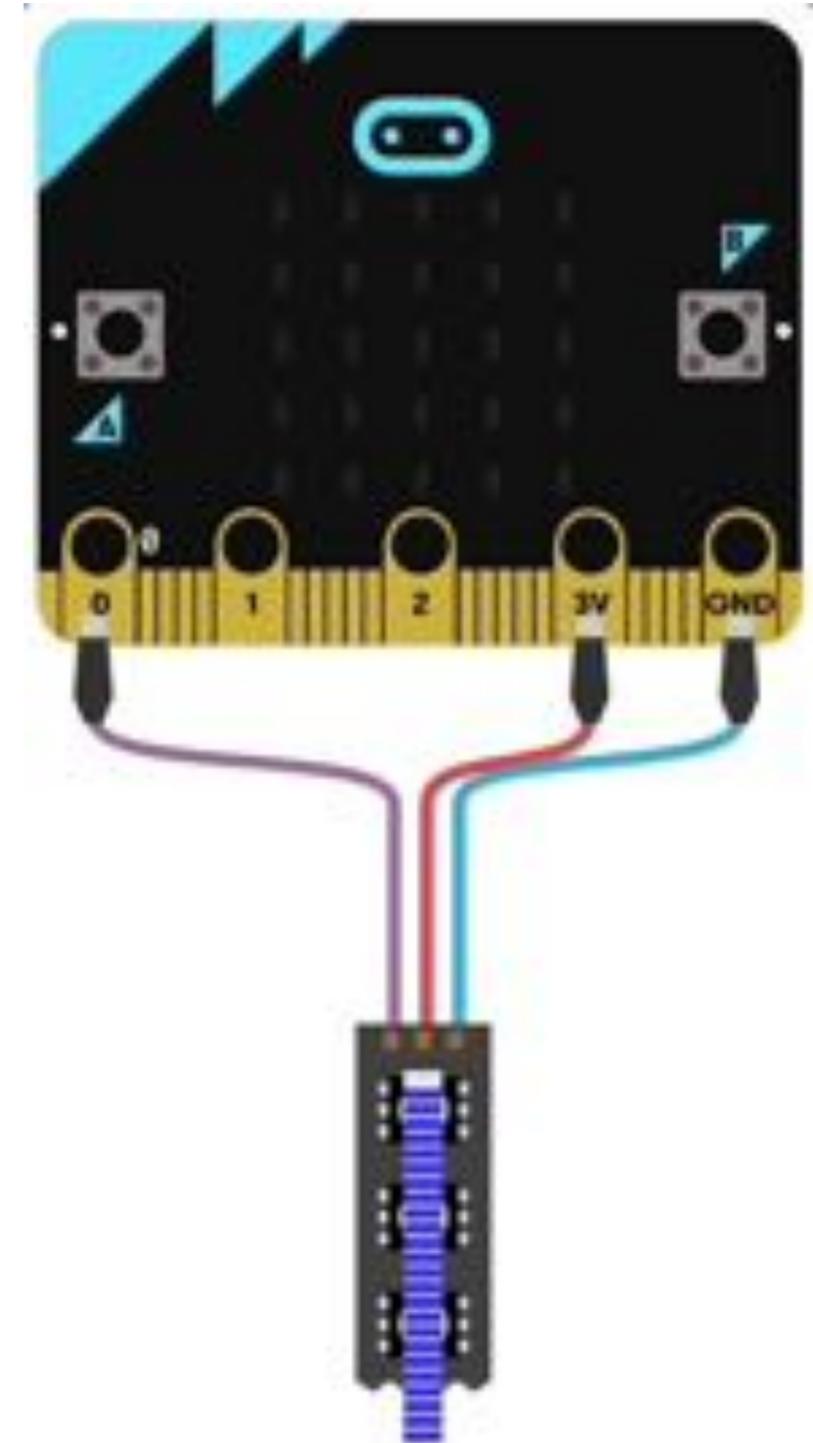
- Connect Di (input) pad on the neopixel to pin 0, 1 or 2 on the microbit
- Connect PWR pad on the Neopixel to the positive terminal of the battery
- Connect GND pad on the Neopixel to the negative terminal of the battery
- Connect GND of the microbit to the negative terminal of the battery (to have common ground for all components)



Change Colour of the Entire Neopixel Strip

```
on start
  set strip to NeoPixel at pin P0 with 24 leds

forever
  strip show color red
  pause (ms) 200
  strip show color green
  pause (ms) 200
  strip show color blue
  pause (ms) 200
```

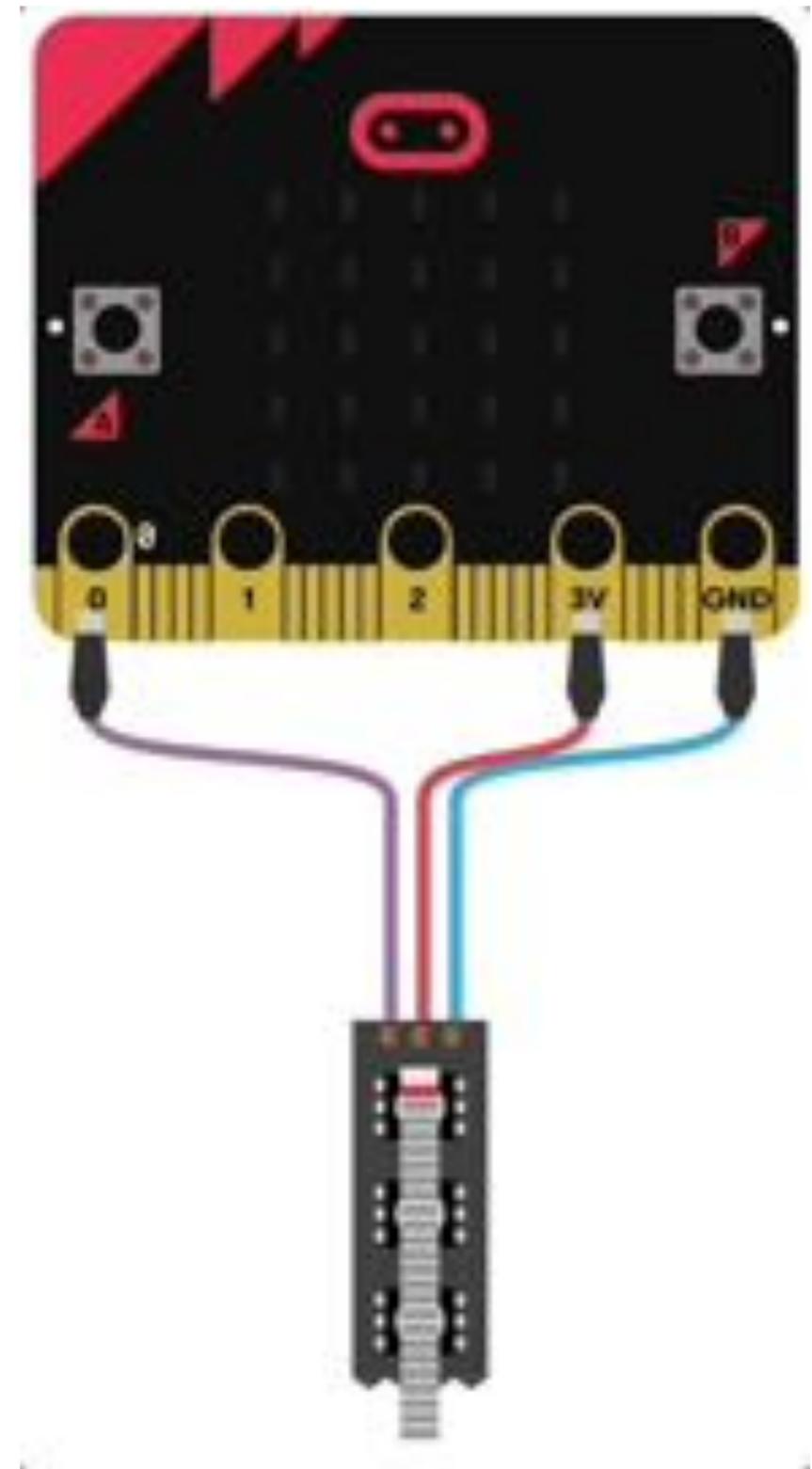


Shift Pixel and Show Command

```
on start  
  set strip to NeoPixel at pin P0 with 24 leds
```

```
forever  
  strip set pixel color at 0 to red  
  strip show  
  pause (ms) 200  
  strip shift pixels by 1
```

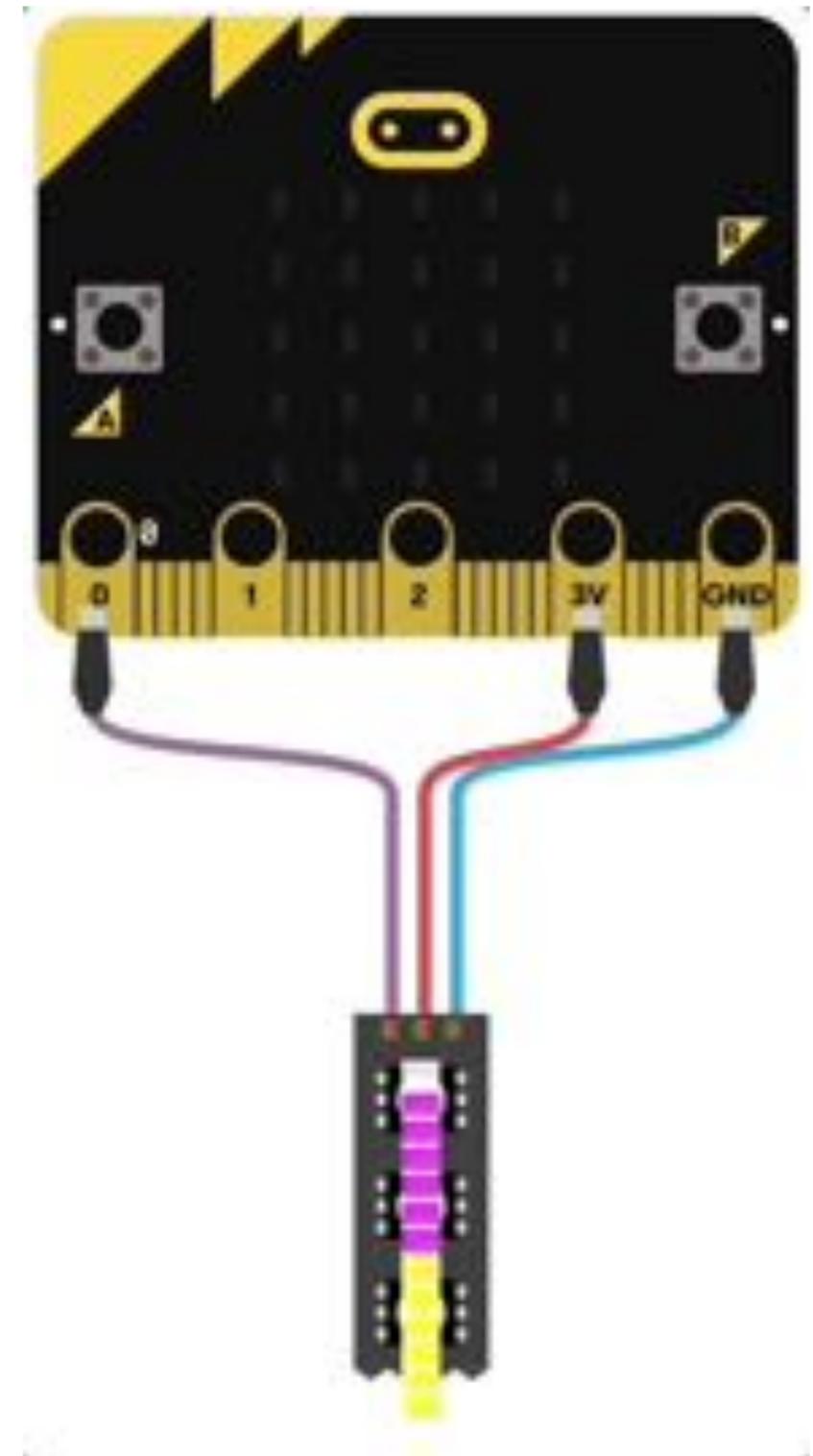
You will need to add this 'Show' command



Using Loops to Control Colour

```
on start
  set strip to NeoPixel at pin P0 with 12 leds as RGB (GRB format)

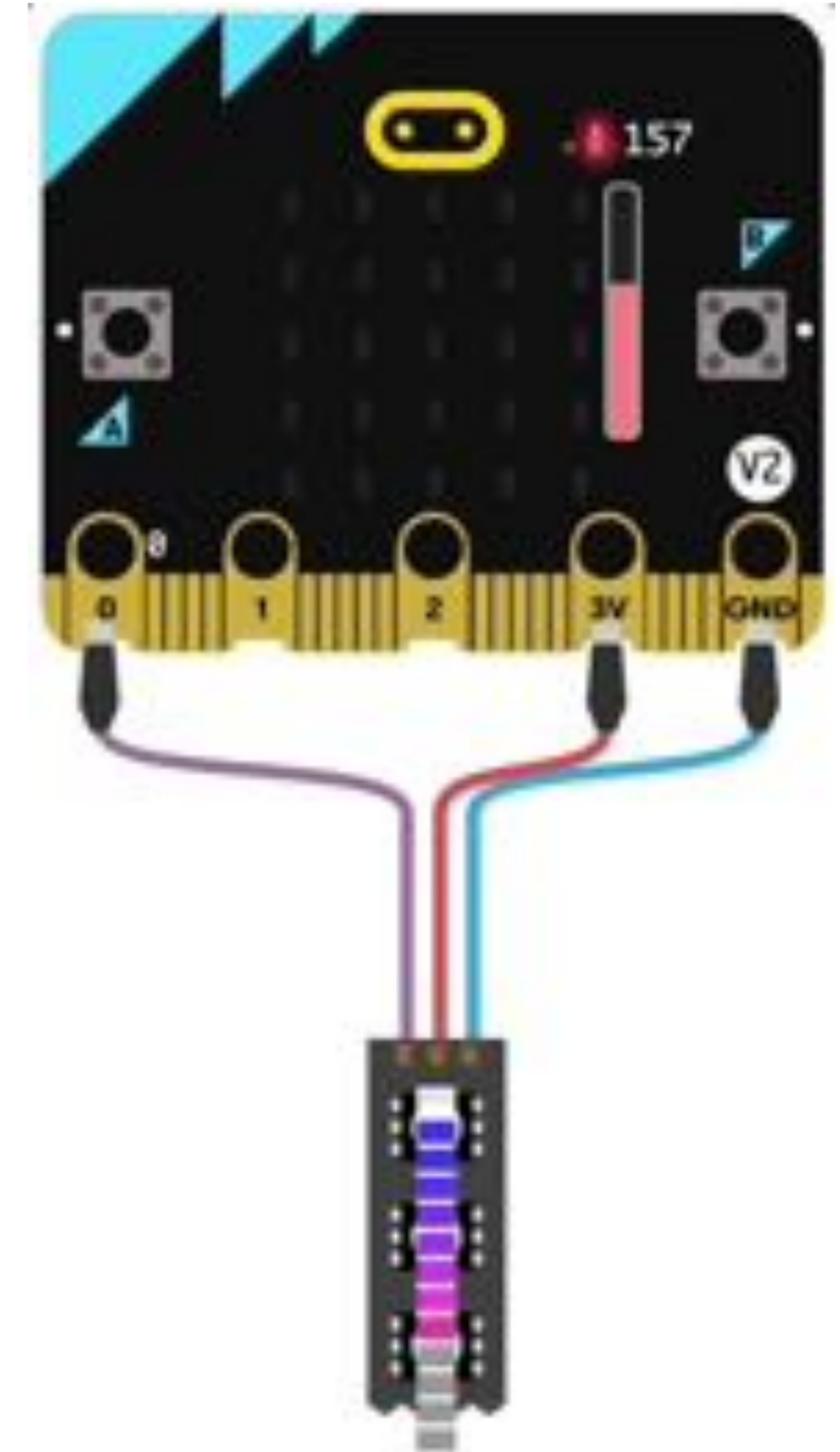
forever
  for index from 0 to 11
    do
      strip set pixel color at index to purple
      strip show
      pause (ms) 100
  for index from 0 to 11
    do
      strip set pixel color at index to yellow
      strip show
      pause (ms) 100
```



Neopixel Bar Graphs

```
on start
  set strip to NeoPixel at pin P0 with 12 leds as

forever
  strip show bar graph of sound level up to 255
```



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 microbit 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'

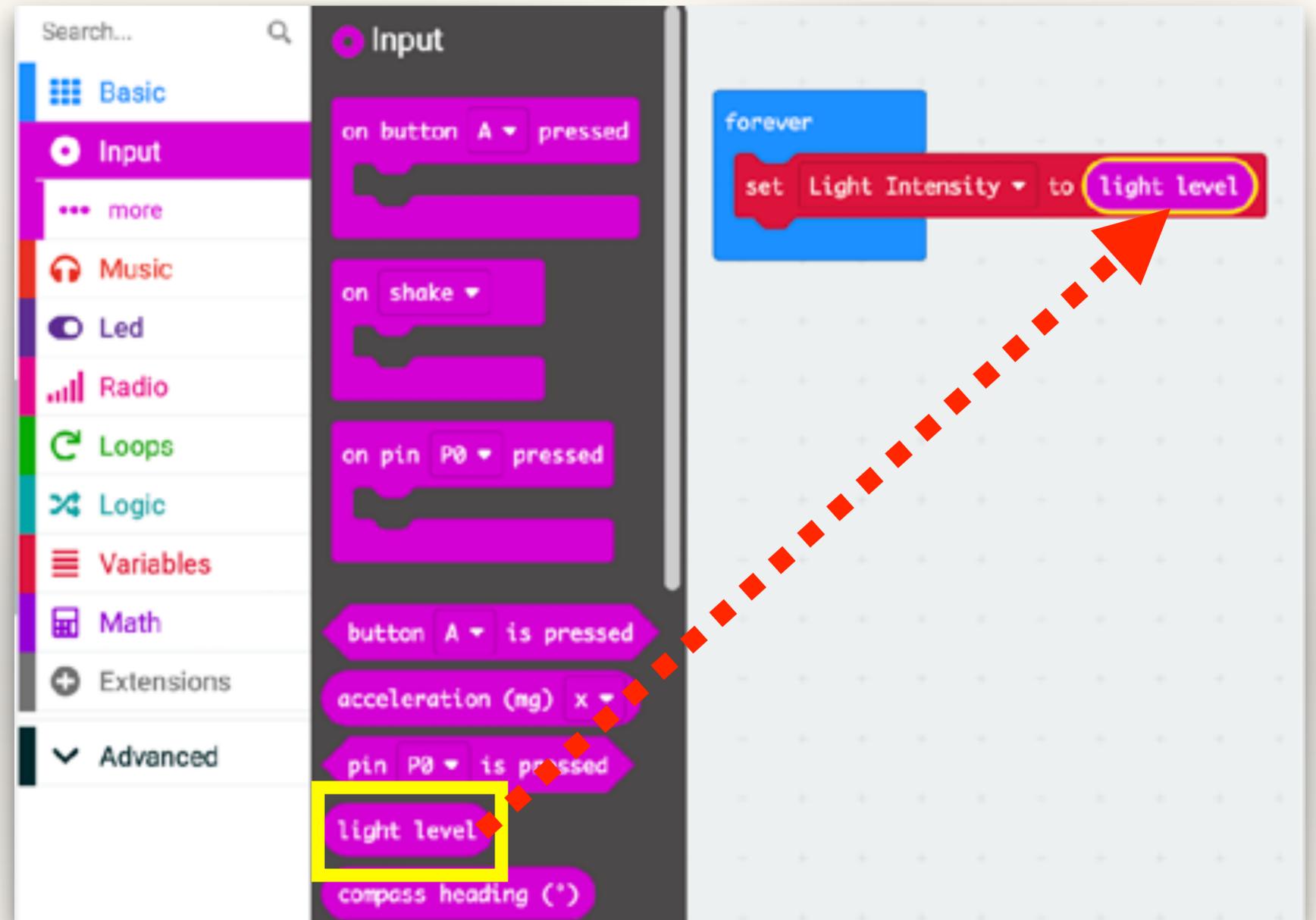
The image shows a screenshot of the Scratch IDE interface. On the left, the 'Variables' menu is open, showing a search bar and a list of categories: Basic, Input, Music, Led, Radio, Loops, Logic, Variables (highlighted), and Math. On the right, the 'Variables' panel is visible. It contains a 'Make a Variable...' button, two code blocks: 'set Light Intensity to 0' and 'change Light Intensity by 1', and a 'Your Variables' section with a dropdown menu showing 'Light Intensity'.

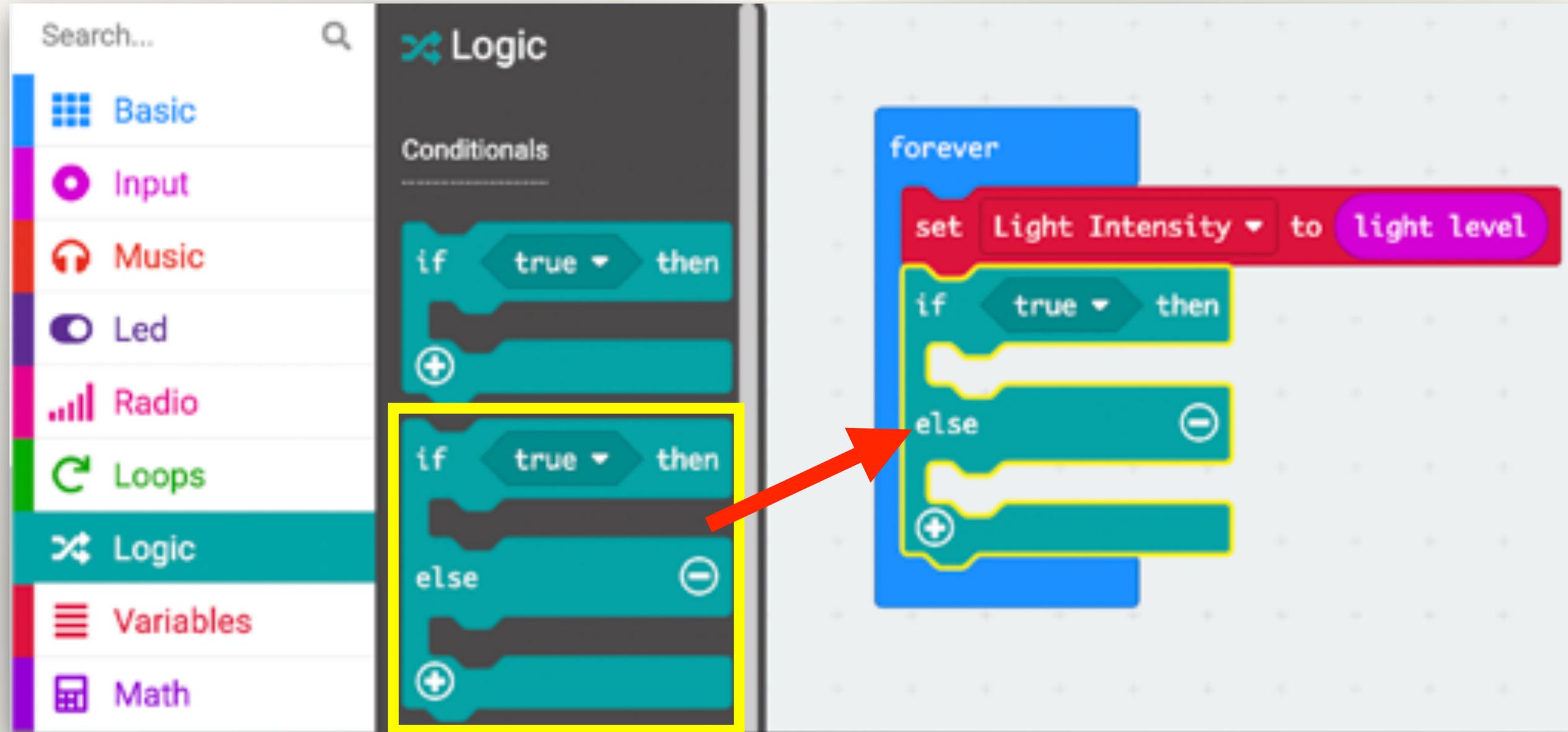
From 'Basic' drag out a 'Forever' block.

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 microbit, it will get updated and stored in the variable called Light Intensity.

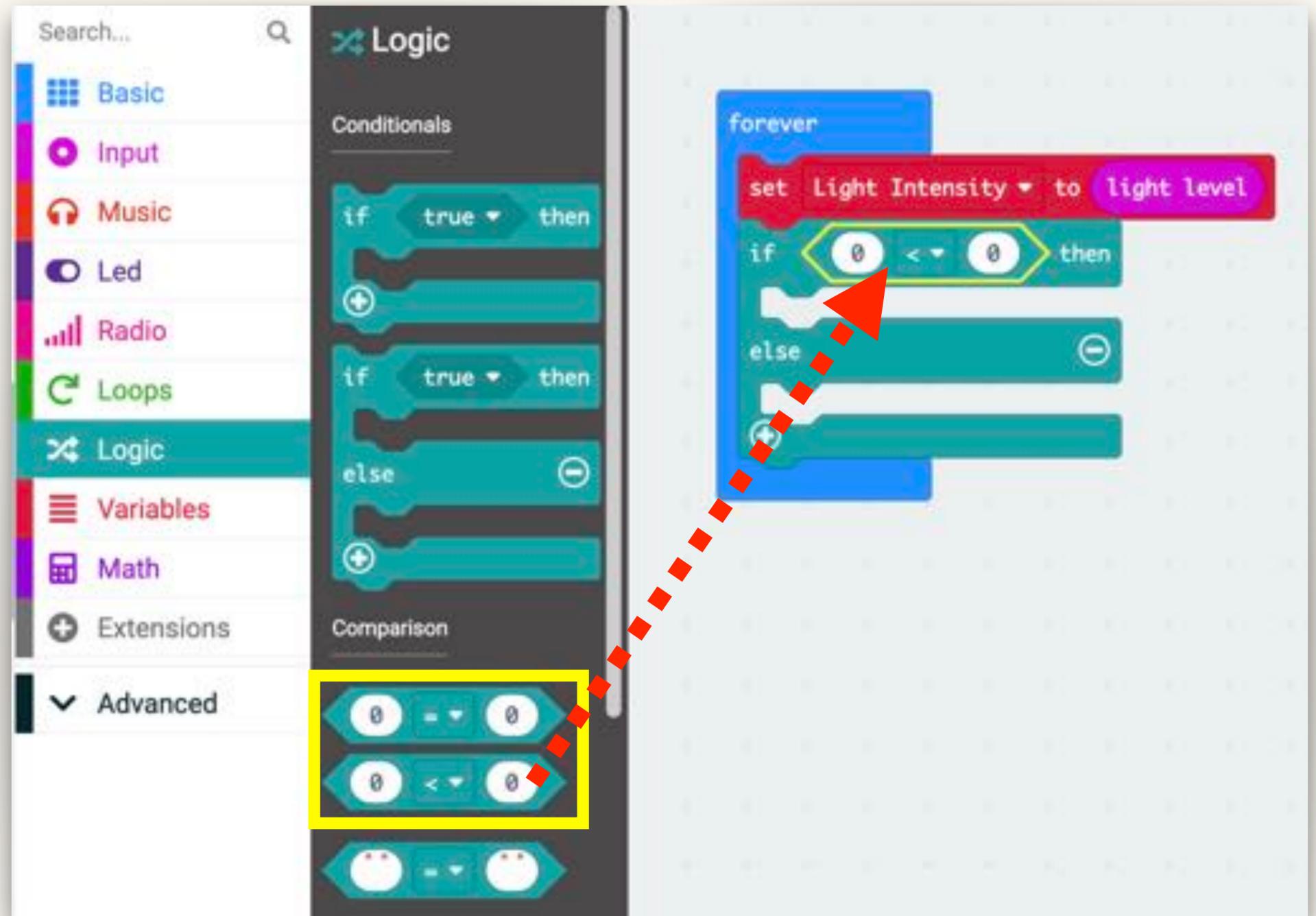




From 'Logic' drag out an 'If-Then-Else' block.

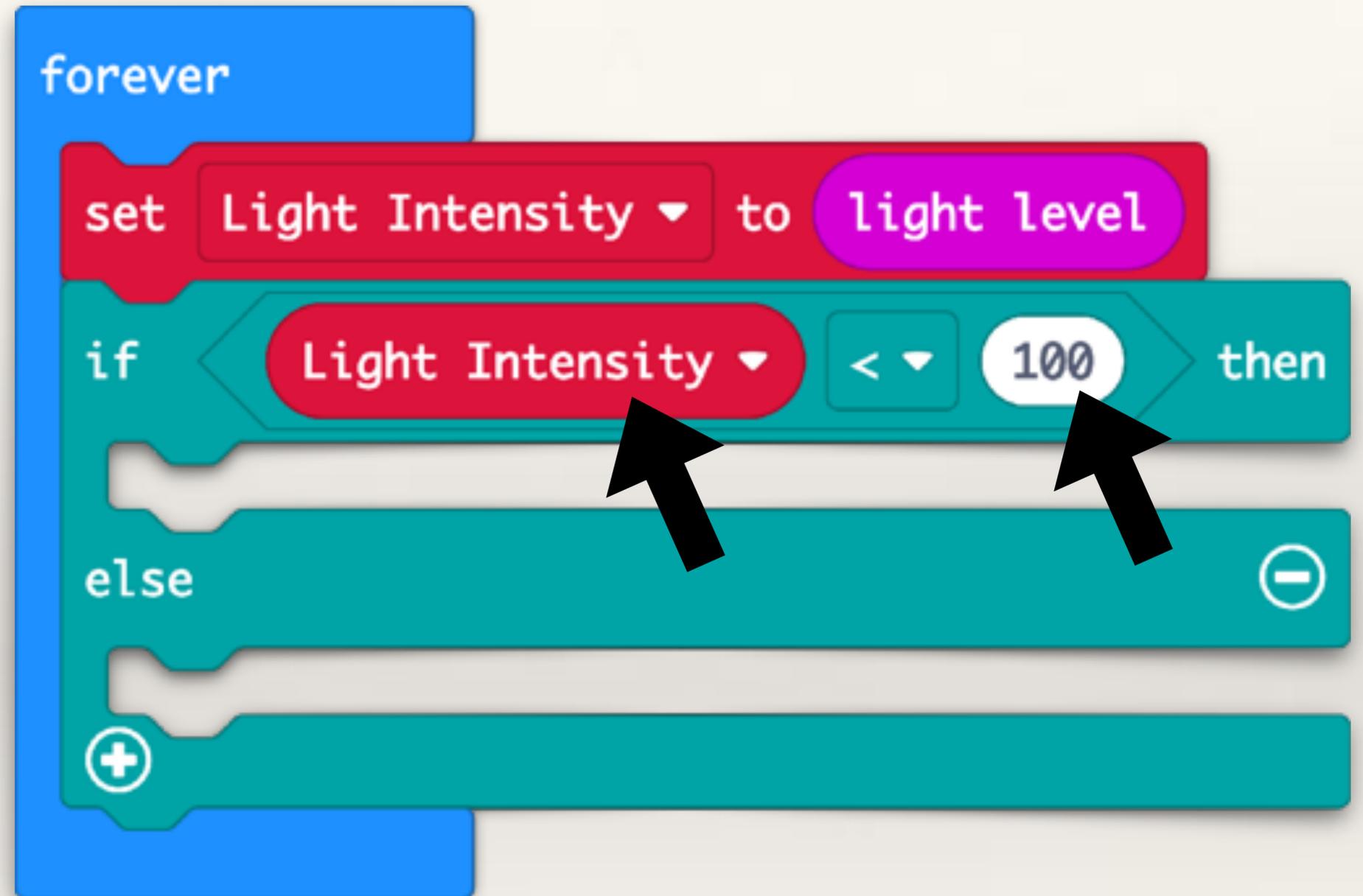
From Logic > Comparison > drag out a 'Less Than' block.

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



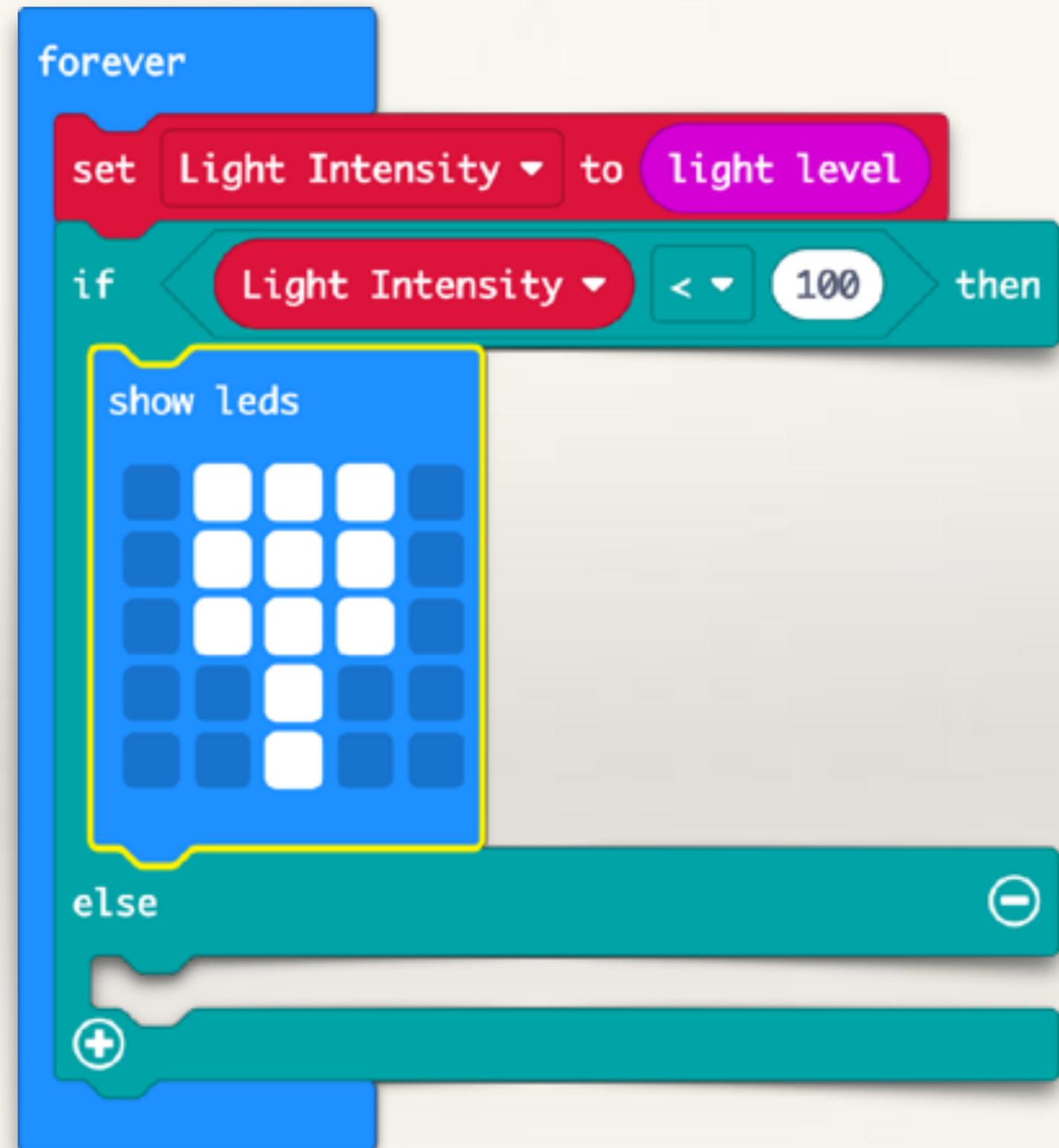
Let's check the condition needed for our Smart Streetlight - that Light Level falling on the microbit should be less than 100.

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 microbit is less than 100, this LED display will show.



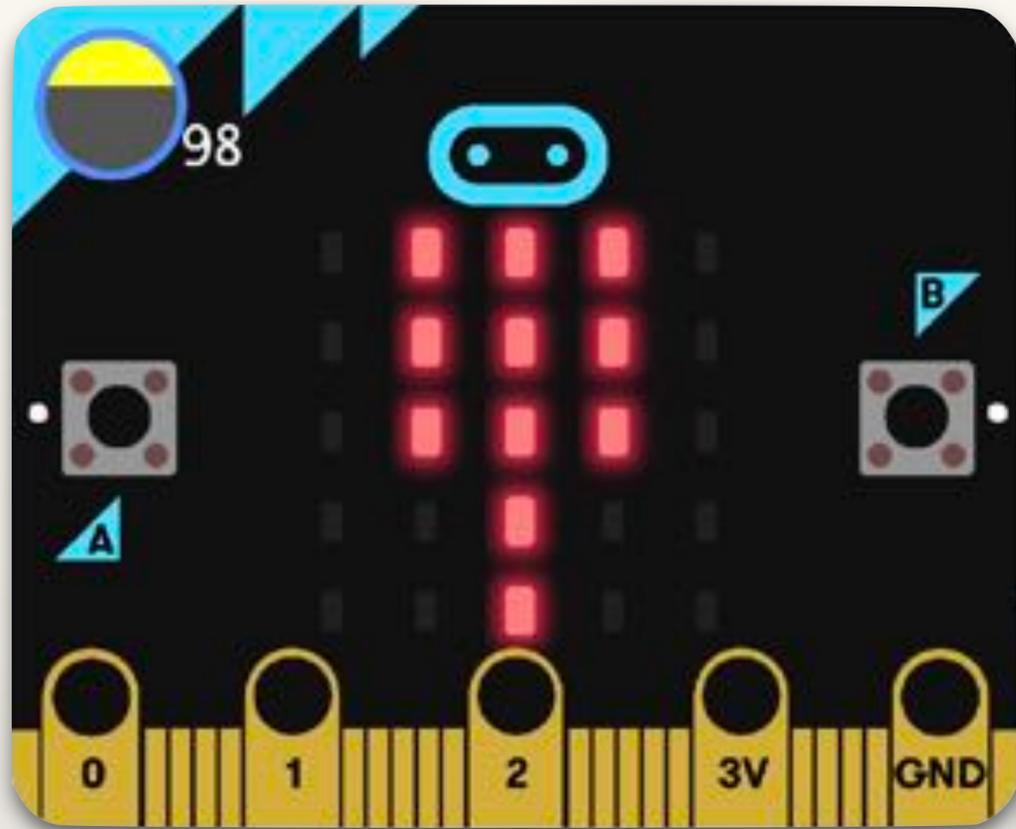
The only other condition we want to check is if the light level falling on the microbit is more than 100.

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

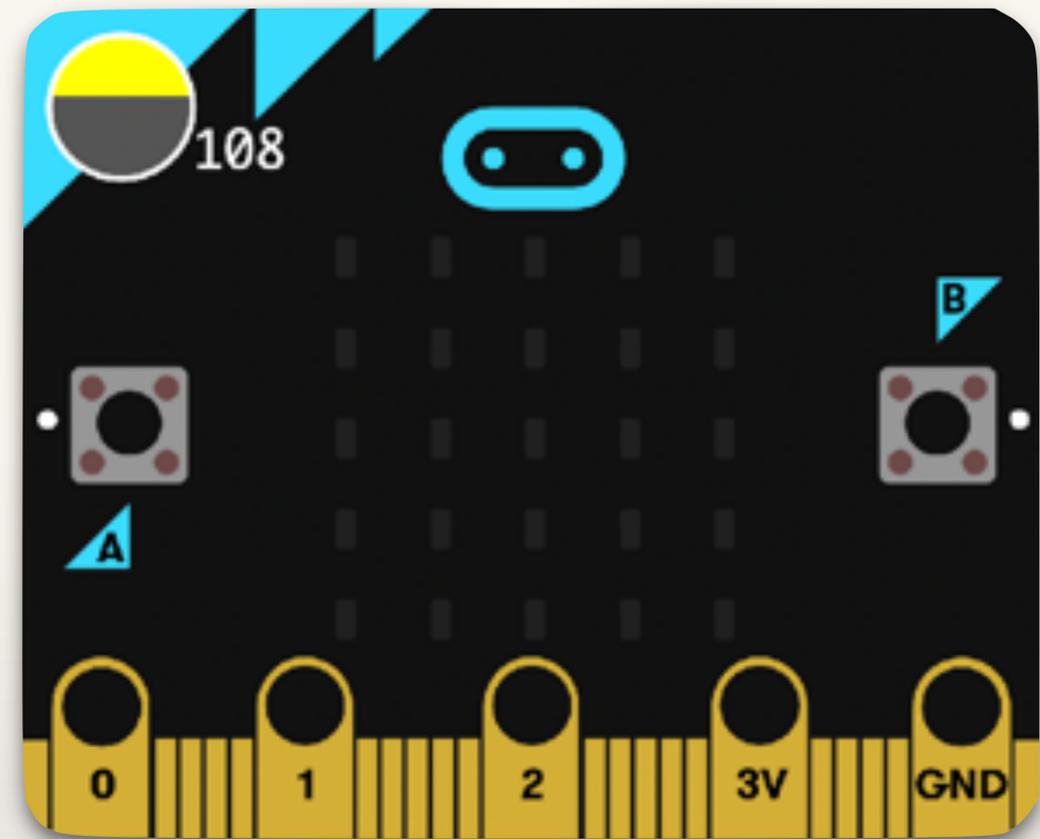
In this case, we want the streetlight to switch off. This is depicted by none of the LEDs lighting up.

```
forever loop
  set Light Intensity to light level
  if Light Intensity < 100 then
    show leds (5x5 grid with 10 white LEDs)
  else
    show leds (5x5 grid with 0 white LEDs)
```

The image shows a Scratch script for a microbit program. It starts with a 'forever' loop block. Inside the loop, there is a 'set' block that sets the variable 'Light Intensity' to the value of 'light level'. This is followed by an 'if' block with the condition 'Light Intensity < 100'. If this condition is true, a 'show leds' block displays a 5x5 grid of LEDs with 10 white LEDs (the top two rows are fully lit, and the middle LED of the third row is lit). If the condition is false, the 'else' block's 'show leds' block displays a 5x5 grid of LEDs with all LEDs turned off (represented by blue squares).



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

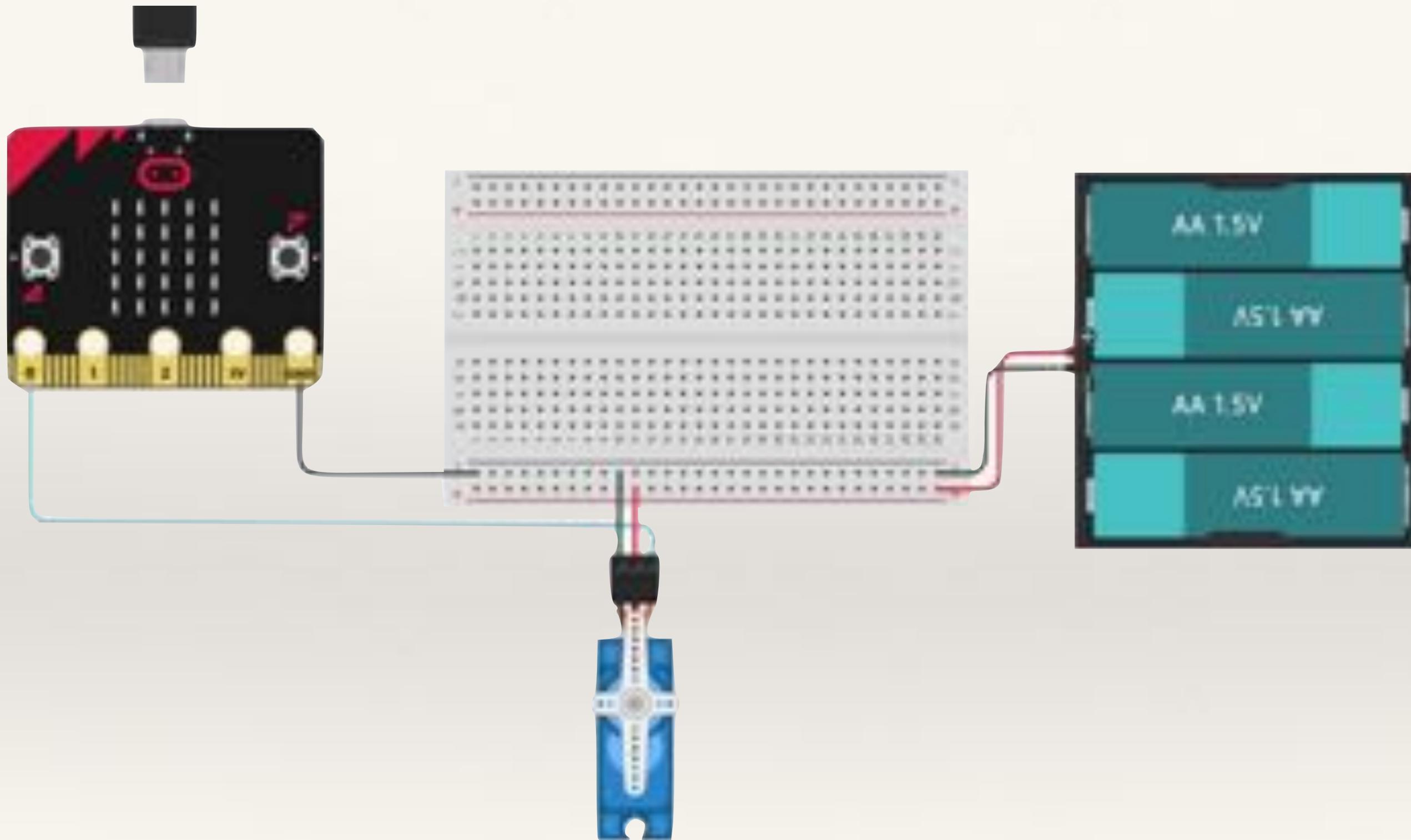


In the simulator, if the light level (stored in the variable called Light Intensity) is more than 100, LEDs will not glow.

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.

Smart Street Light on TinkerCad

Smart Fan on TinkerCad



forever

show number temperature (°C)

if temperature (°C) > 21 then

repeat 10 times

rotate servo on pin P0 to 0 degrees

wait 1 secs

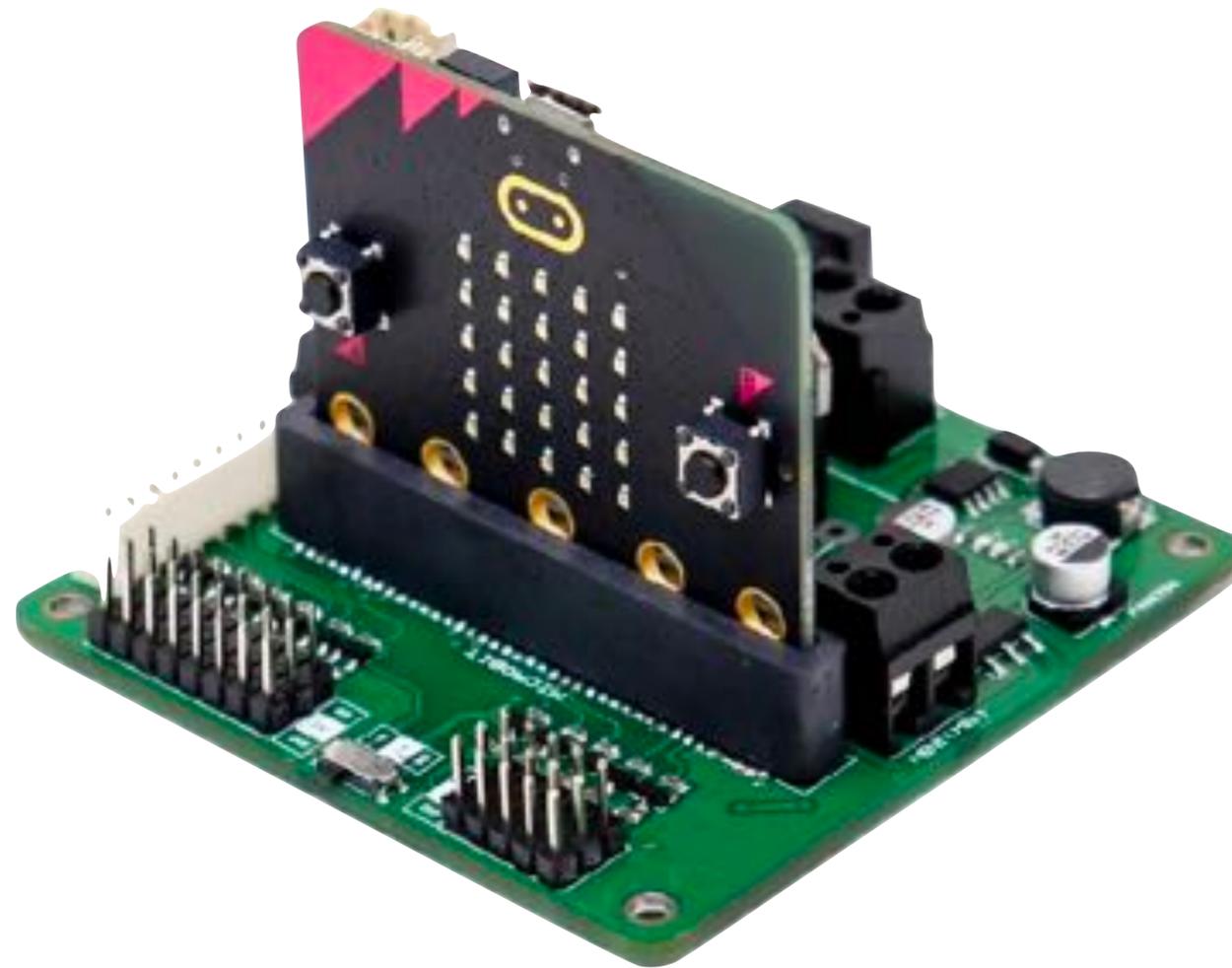
rotate servo on pin P0 to 180 degrees

wait 1 secs

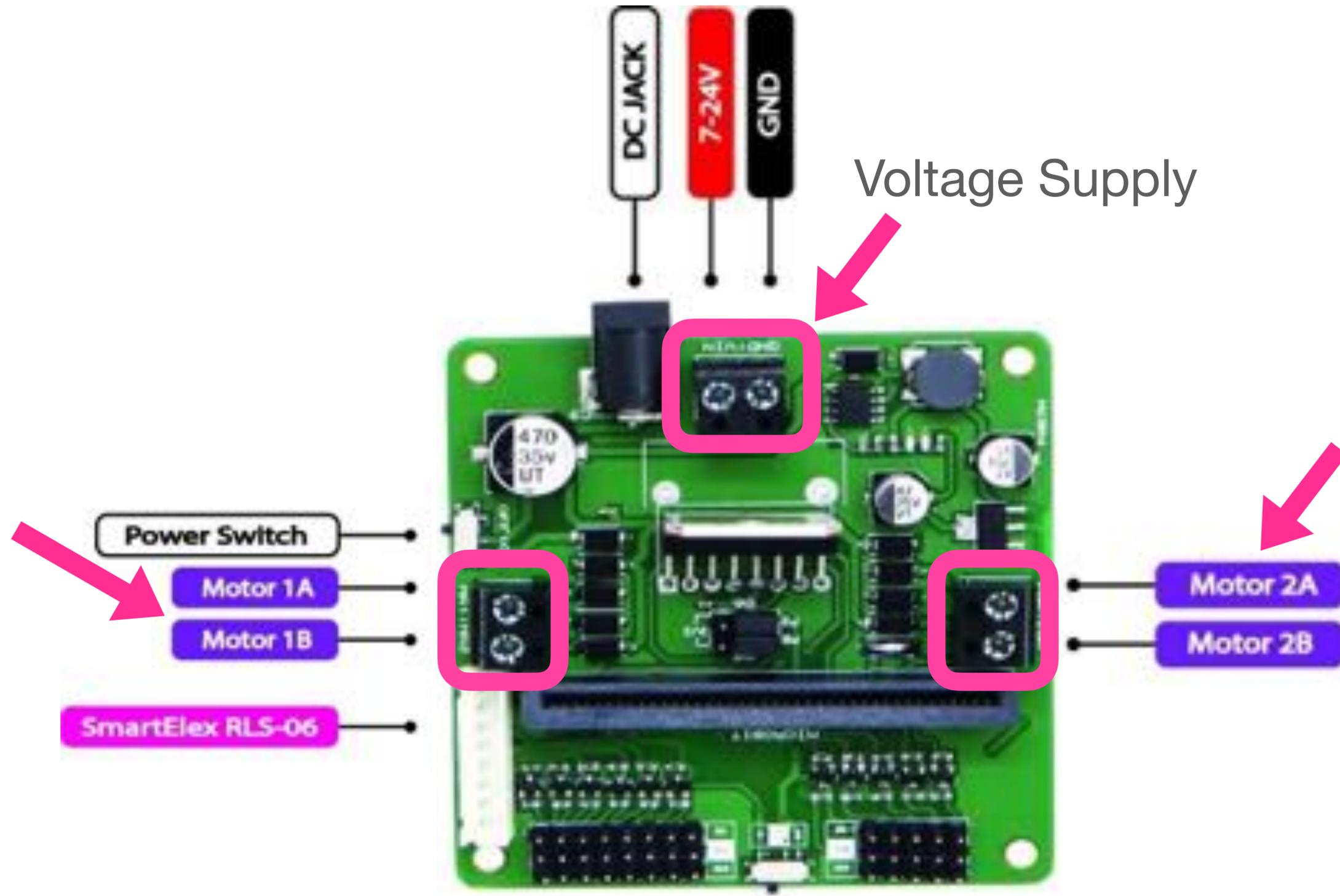
else

digital write pin P0 to LOW

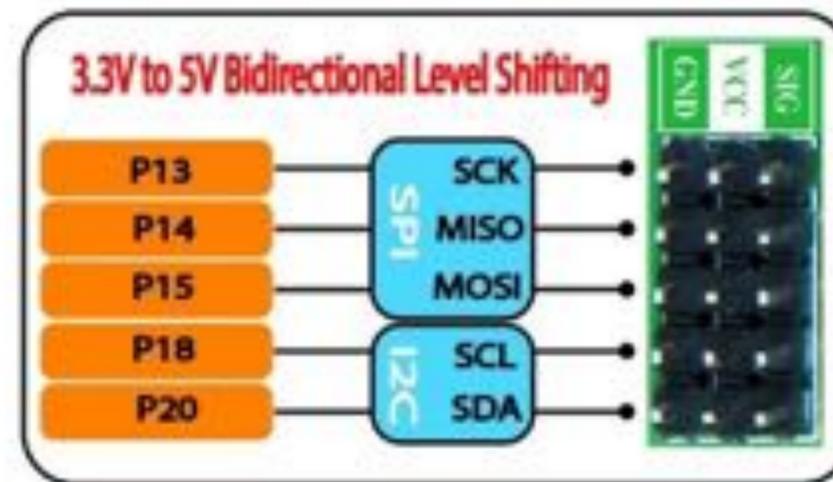
Motor Driver



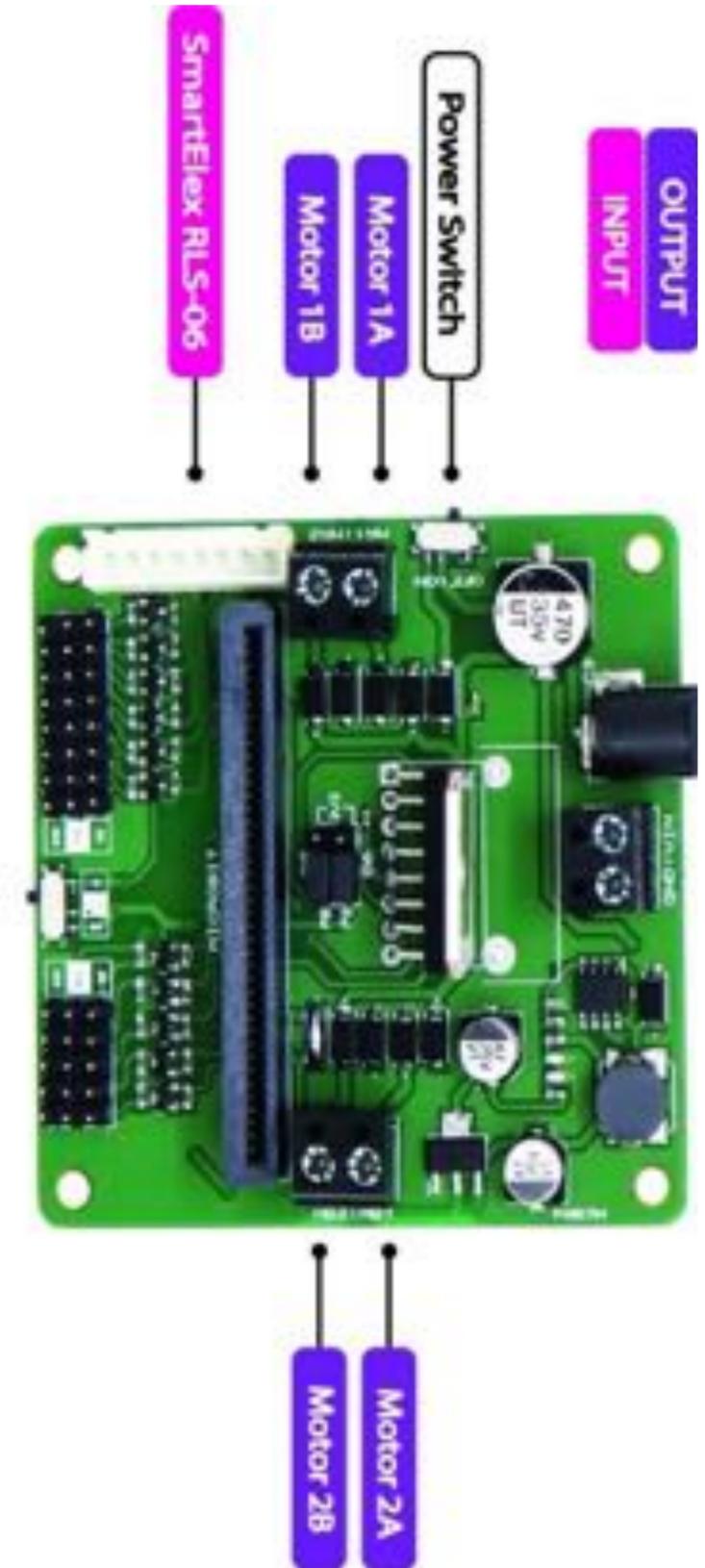
Motor Driver Board - SmartElex



Complete PinOut for Motor Driver Board - SmartElex

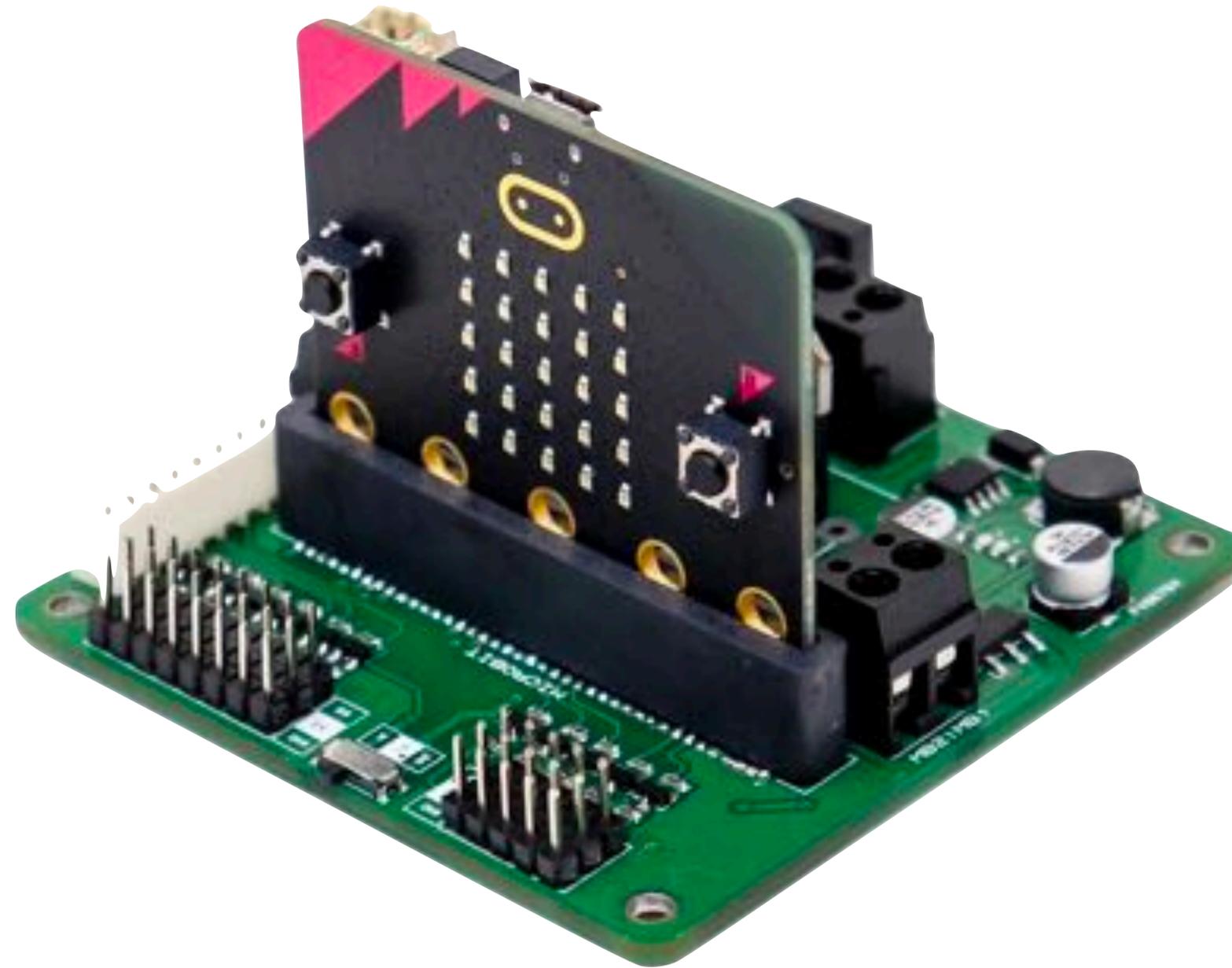


Vcc Switch

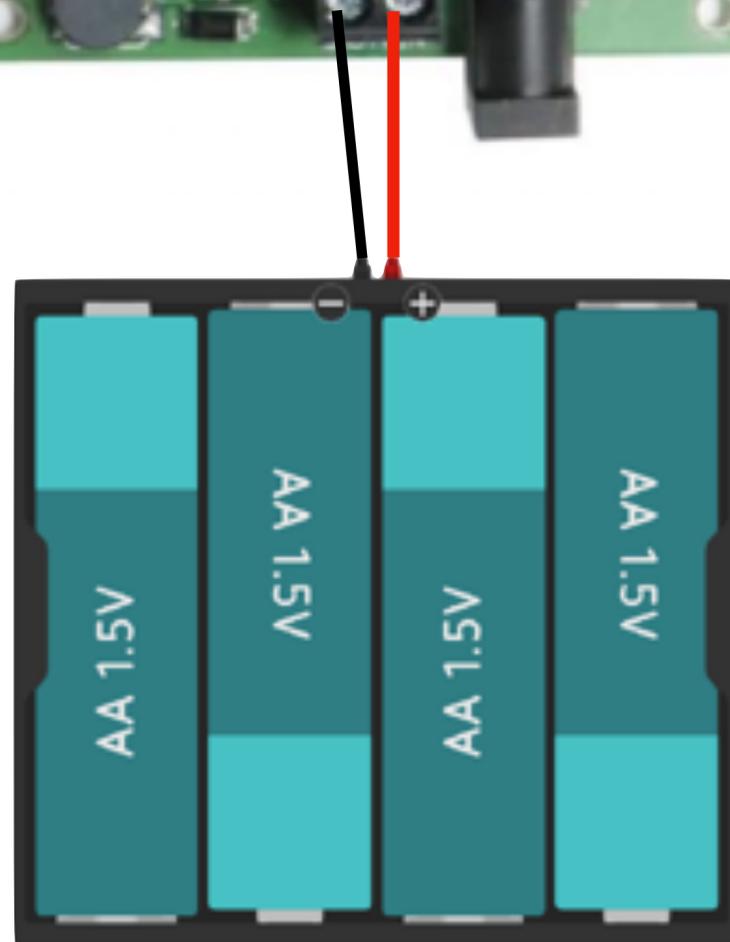
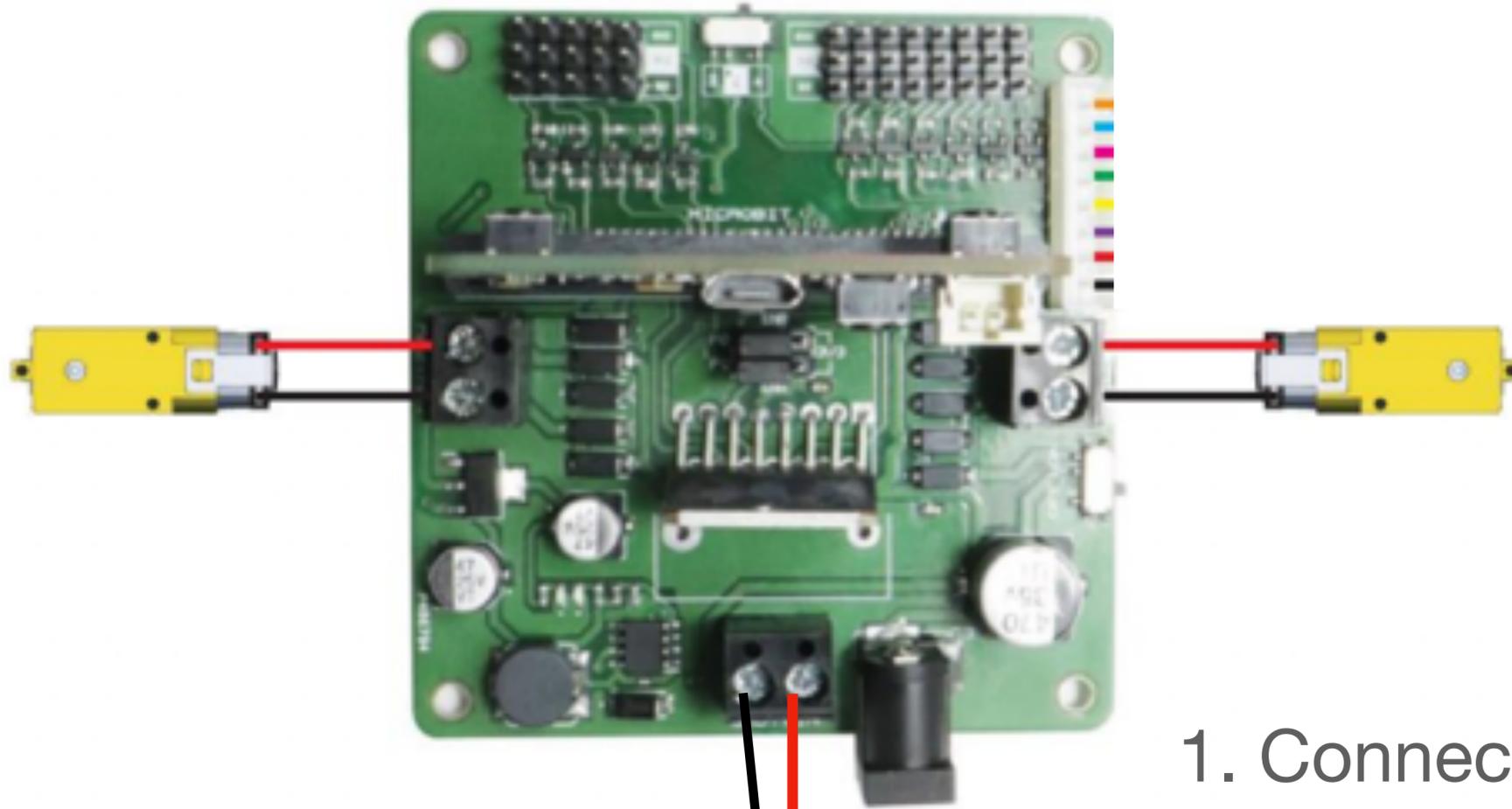


Data Sheet:

<https://robu.in/wp-content/uploads/2021/07/SmartElex-MicroBit-L298n-Driver.pdf>



Insert the Microbit into its slot on the Motor Driver Board



1. Connect two geared DC motors as shown
2. For programming in MakeCode:
 - Motor-1: Pin-1 and Pin-8
 - Motor-2: Pin-12 and Pin-16

```
on button A pressed
digital write pin P1 to 0
digital write pin P8 to 1
digital write pin P12 to 0
digital write pin P16 to 1
```

Forward

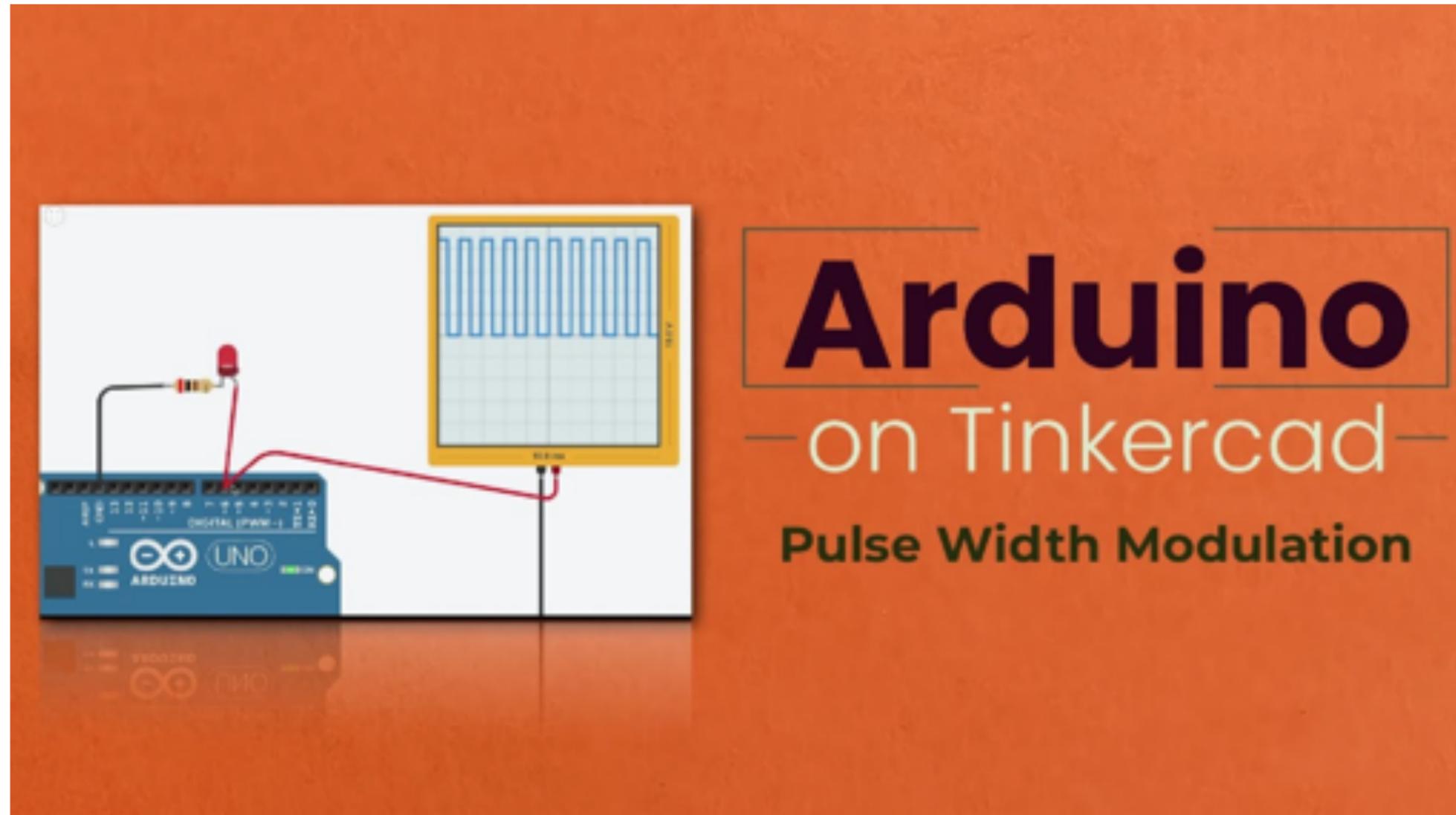
```
on button B pressed
digital write pin P1 to 1
digital write pin P8 to 0
digital write pin P12 to 1
digital write pin P16 to 0
```

Reverse

```
on button A+B pressed
digital write pin P1 to 0
digital write pin P8 to 0
digital write pin P12 to 0
digital write pin P16 to 0
```

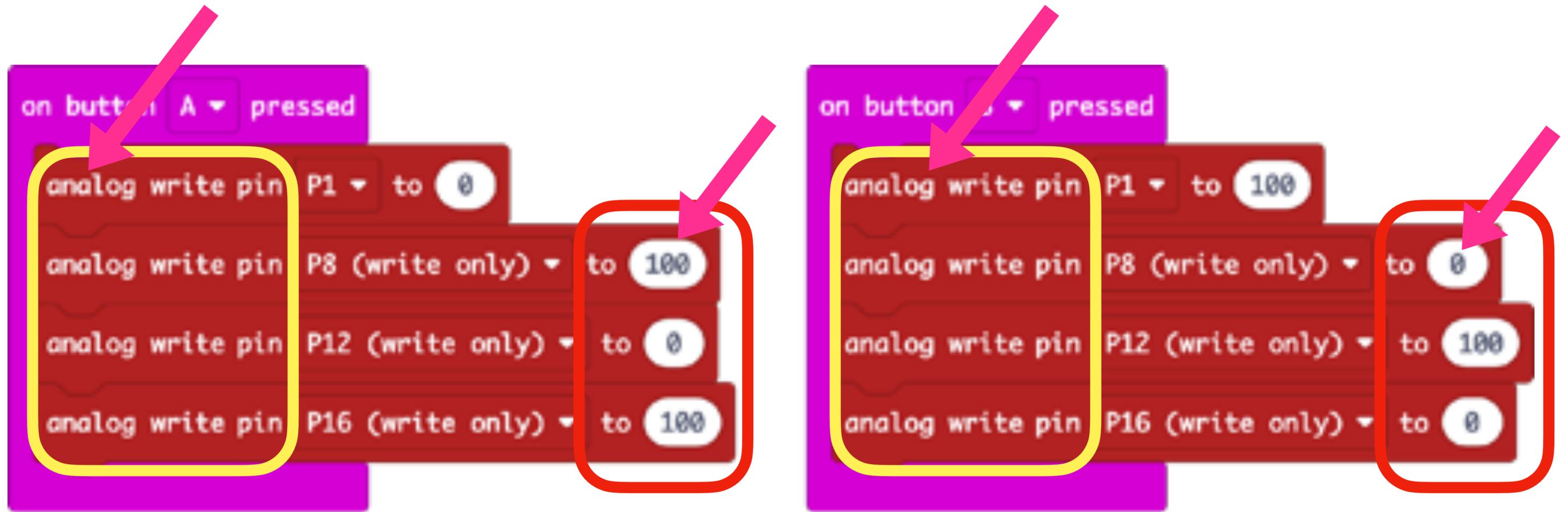
Stop

Controlling the Speed - Understanding PWM



<https://youtu.be/bGZZ2FMwsuA>

Controlling the Speed with PWM



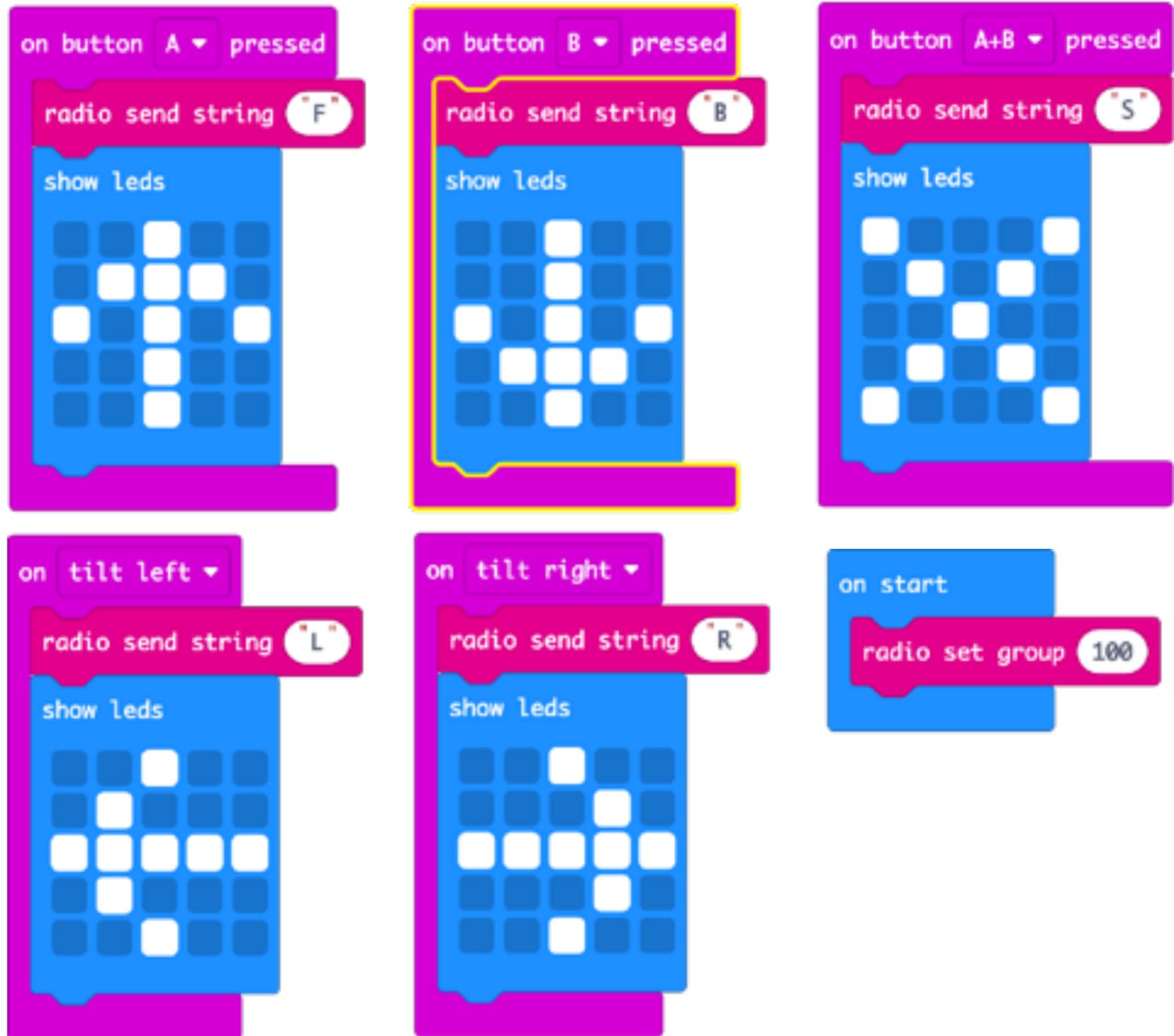
Using Analog Write Command and put Numerical values

Use Gesture (tilt left or right) to Turn

```
on tilt right ▾  
  analog write pin P1 ▾ to 0  
  analog write pin P8 (write only) ▾ to 0  
  analog write pin P12 (write only) ▾ to 100  
  analog write pin P16 (write only) ▾ to 0
```

```
on tilt left ▾  
  analog write pin P1 ▾ to 0  
  analog write pin P8 (write only) ▾ to 100  
  analog write pin P12 (write only) ▾ to 0  
  analog write pin P16 (write only) ▾ to 0
```

Microbit Radio- Controlled Car: TRANSMITTER



Microbit Radio-Controlled Car - RECEIVER

```
on radio received receivedString
  if receivedString == "F" then
    show string "F"
    analog write pin P1 to 0
    analog write pin P8 (write only) to 100
    analog write pin P12 (write only) to 0
    analog write pin P16 (write only) to 200
  if receivedString == "B" then
    show string "B"
    analog write pin P1 to 100
    analog write pin P8 (write only) to 0
    analog write pin P12 (write only) to 200
    analog write pin P16 (write only) to 0
```

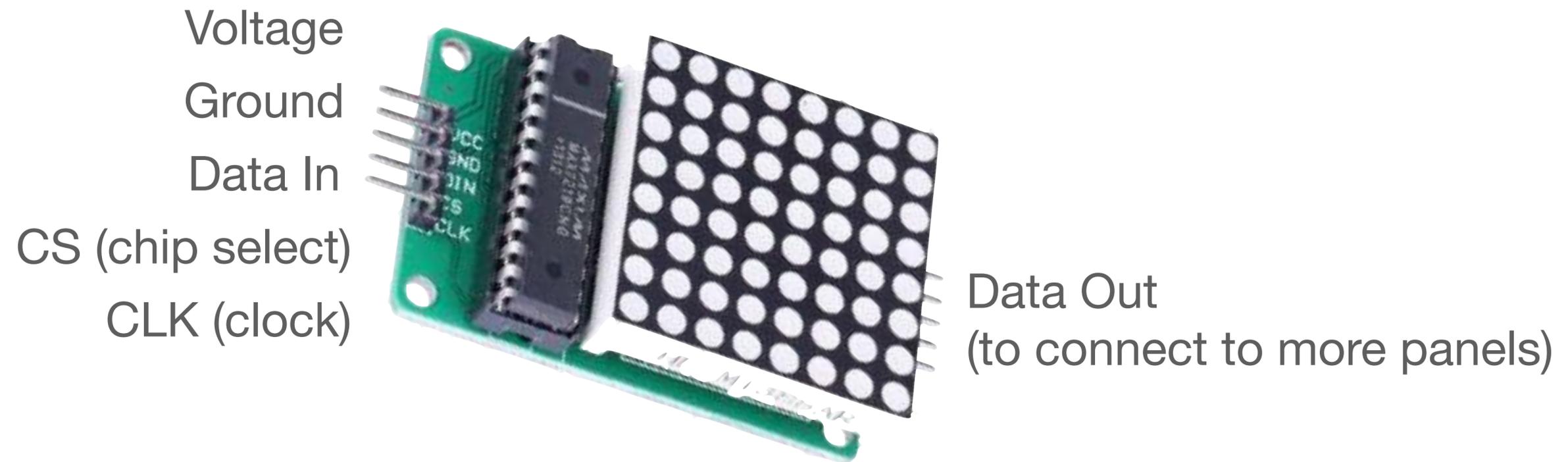
```
if receivedString == "L" then
  show string "L"
  analog write pin P1 to 0
  analog write pin P8 (write only) to 100
  analog write pin P12 (write only) to 0
  analog write pin P16 (write only) to 0
+
if receivedString == "R" then
  show string "R"
  analog write pin P1 to 0
  analog write pin P8 (write only) to 100
  analog write pin P12 (write only) to 0
  analog write pin P16 (write only) to 0
```

```
on start
  radio set group 100
```

```
if receivedString == "S" then
  show string "S"
  analog write pin P1 to 0
  analog write pin P8 (write only) to 0
  analog write pin P12 (write only) to 0
  analog write pin P16 (write only) to 0
+
```

8x8 LED Display

Max7219 8x8 LED Display - Pin Out



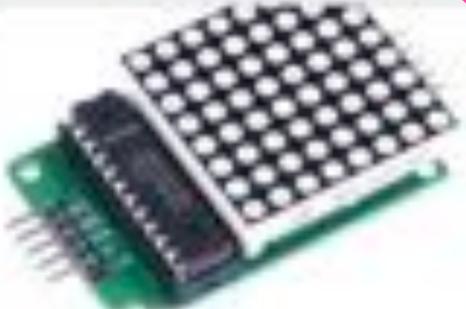
Add Extension - MAX7219_8x8

← Go Back Extensions ?

8x8 LED

Lights and Display Software Science Robotics Gaming Networking

Home Import File



MAX7219_8x8
MakeCode extension for MAX7219 8x8 matrix LED modules

[Learn More](#)



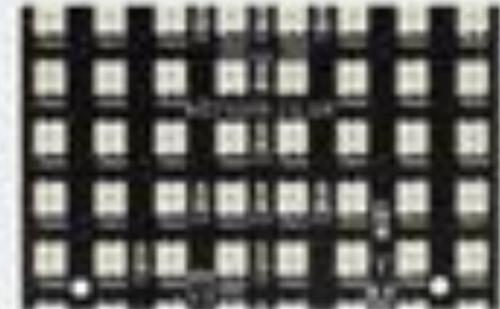
MAX7219_8x8
MakeCode extension for MAX7219 8x8 matrix LED modules

[Learn More](#)



kitronik-zip-64
Custom blocks for www.kitronik.co.uk/5626-GAME-ZIP64 for micro:bit

[Learn More](#)



kitronik-zip-tile
Custom blocks for www.kitronik.co.uk/5645-ZIP-File for BBC micro:bit

[Learn More](#)

```
on start
  Fill all LEDs
  Setup MAX7219:
    Number of matrixs 1
    CS(LOAD) = P16
    MOSI(DIN) = P15
    MISO(not used) = P14
    SCK(CLK) = P13
  Rotate matrix display counter-clockwise
  Reverse printing order false
```

```
forever
  Scroll text "Hello World!"
  delay (ms) 150
  at the end wait (ms) 500
```

Code for "Hello World!"

```
on start
  Fill all LEDs
  Setup MAX7219:
    Number of matrixs 4
    CS(LOAD) = P16
    MOSI(DIN) = P15
    MISO(not used) = P14
    SCK(CLK) = P13
  Rotate matrix display clockwise
  Reverse printing order false
```

```
forever
  Scroll text "Hello World!"
  delay (ms) 150
  at the end wait (ms) 500
```

```
Rotate matrix display clockwise
Reverse printing order
```

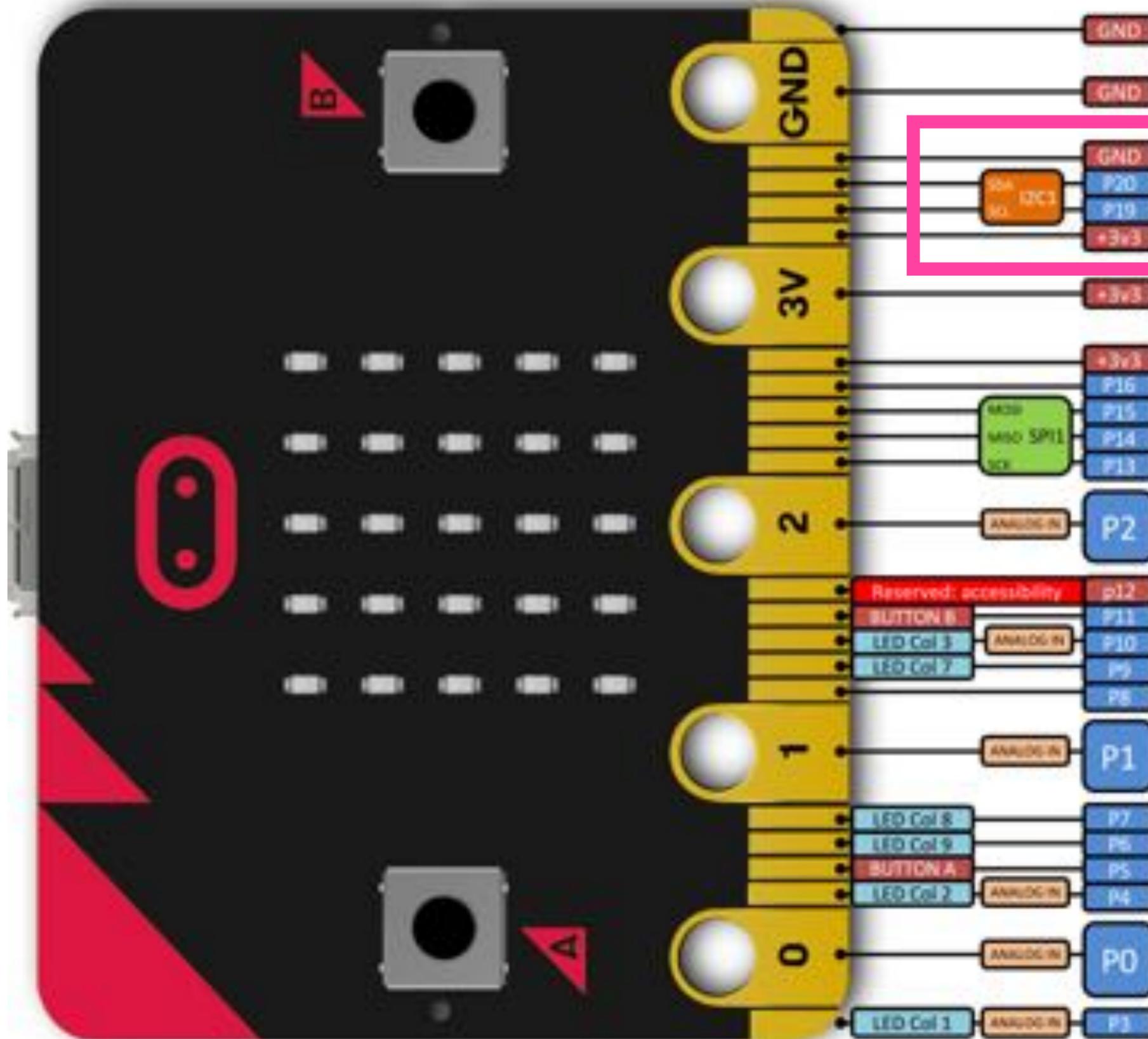
- none
- ✓ clockwise
- counter-clockwise
- 180-degree



i2c LCD Display



I2C and SPI 1602 Serial Character LCD for Micro:bit
by Cytron Technologies



The i2c LCD needs to be connected to pin-19 (SCL) and pin-20 (SDA) of the microbit.

Note that the Cytron Technologies 1602 LCD runs on 3 volts that is why we can connect it directly to the microbit.

i2c (inter-integrated circuit) bus is a chip-level serial communications mechanism that operates over just two wires - SCL (serial clock) line and SDA (serial data) line.

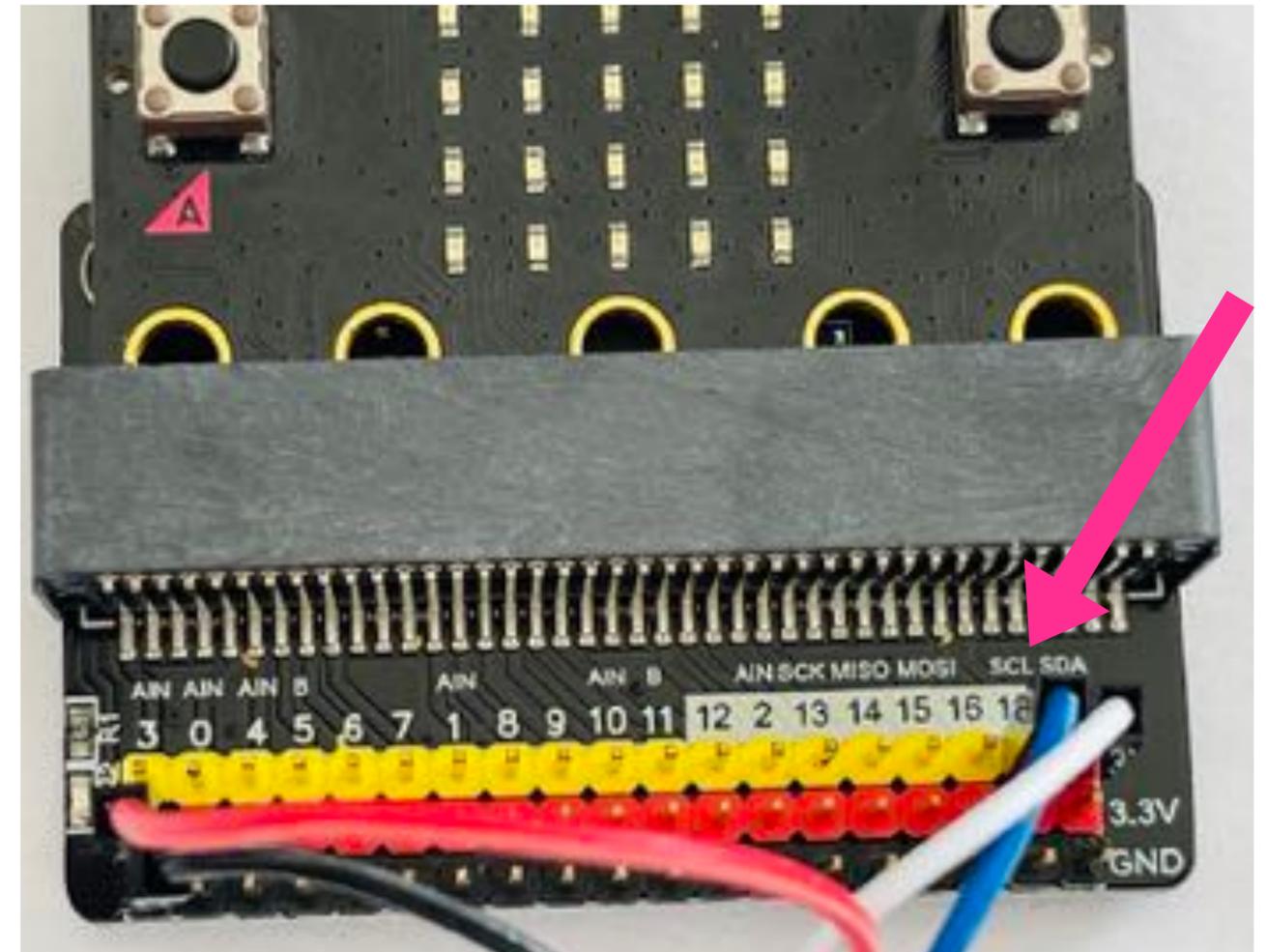
Connecting the LCD with Micro:bit

VSS = Ground pin on microbit

VDD = Voltage pin on microbit

SCK = SCL (serial clock) Pin 19

SDK = Serial Data, Pin 20



Micro:bit on a Breakout Board



On MakeCode Extensions, paste this URL:
<https://github.com/CytronTrainee/microbit-LCM1602-14-LCD-extension>
and install the extension.

Search... 🔍

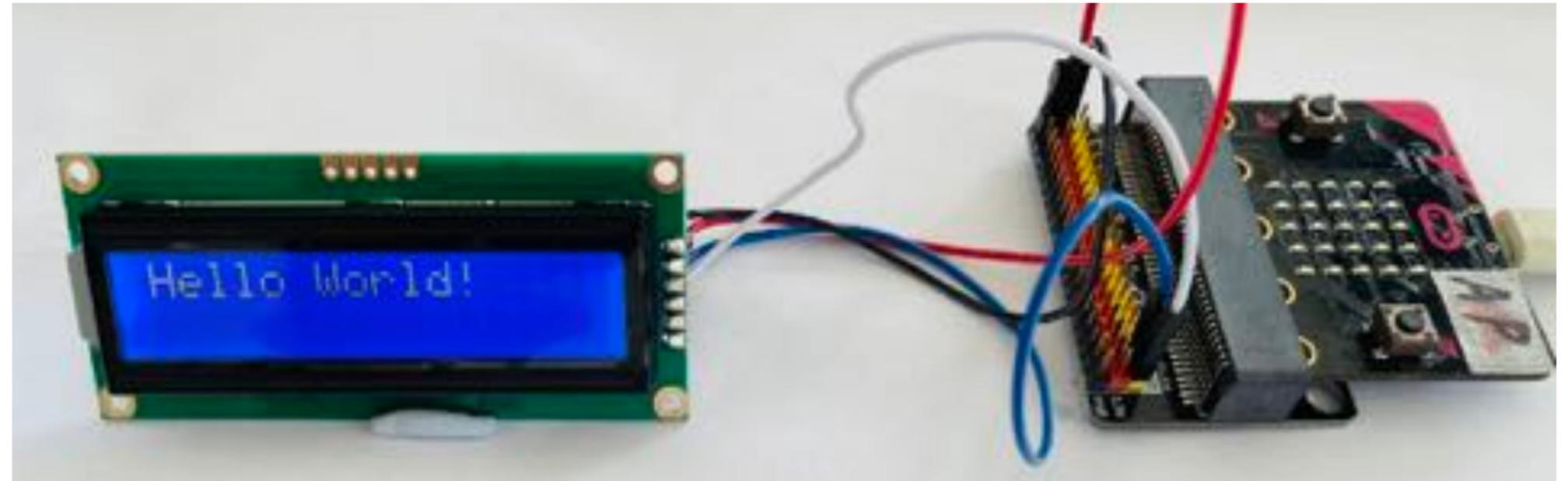
- Basic
- Input
- Music
- Led
- LCD_i2c**
- Radio
- Loops
- Logic

LCD_i2c

- LCD Display Scroll Left
- LCD Display Scroll Right
- LCD Initialize with Address 0
- Write Your Text Here
- Enter Row at 0 and Column at 0

You will get these additional programming blocks

Simple “Hello World!” Display



on start

LCD Initialize with Address

Enter Row at and Column at

Write Your Text Here

```
on start
  LCD Initialize with Address 0
  Enter Row at 1 and Column at 16
  Write Hello World!
```

Line 1

Row can be the top row or bottom row

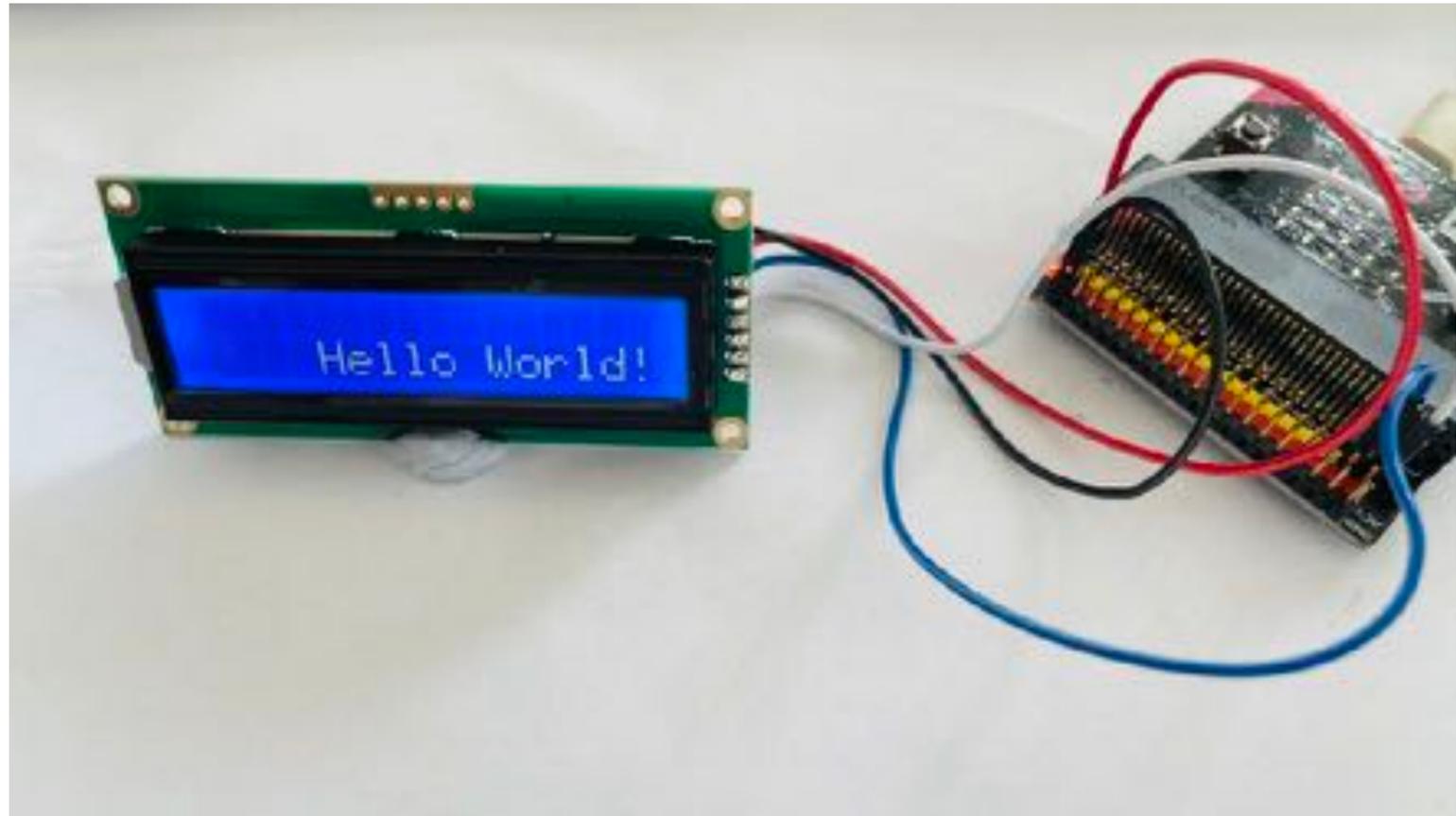


```
on start
  LCD Initialize with Address 0
  Enter Row at 1 and Column at 16
  Write Your Text Here H
```

Column 16

Columns represent the 16 characters that can be displayed

```
on start
  LCD Initialize with Address 0
  Enter Row at 1 and Column at 4
  Write Your Text Here "Hello World!"
```



```
on start
  LCD Initialize with Address 0
  Enter Row at 1 and Column at 4
  Write Your Text Here "Hello World!"
```

```
forever
  LCD Display Scroll Left
  pause (ms) 500
```



Scrolling the Display and Controlling the Speed of the Scroll

Using the LCD to Display Micro:bit Sensor Information

on start

LCD Initialize with Address 0

forever

set temperature to temperature (°C)

set Acceleration to acceleration (mg) strength

Enter Row at 0 and Column at 0

Write Your Text Here join "Room Temp: " convert temperature to text °C

Enter Row at 1 and Column at 0

Write Your Text Here join "Accelr: " convert Acceleration to text

MQ Sensors

MQ Sensors



MQ-135: Air Quality Sensor

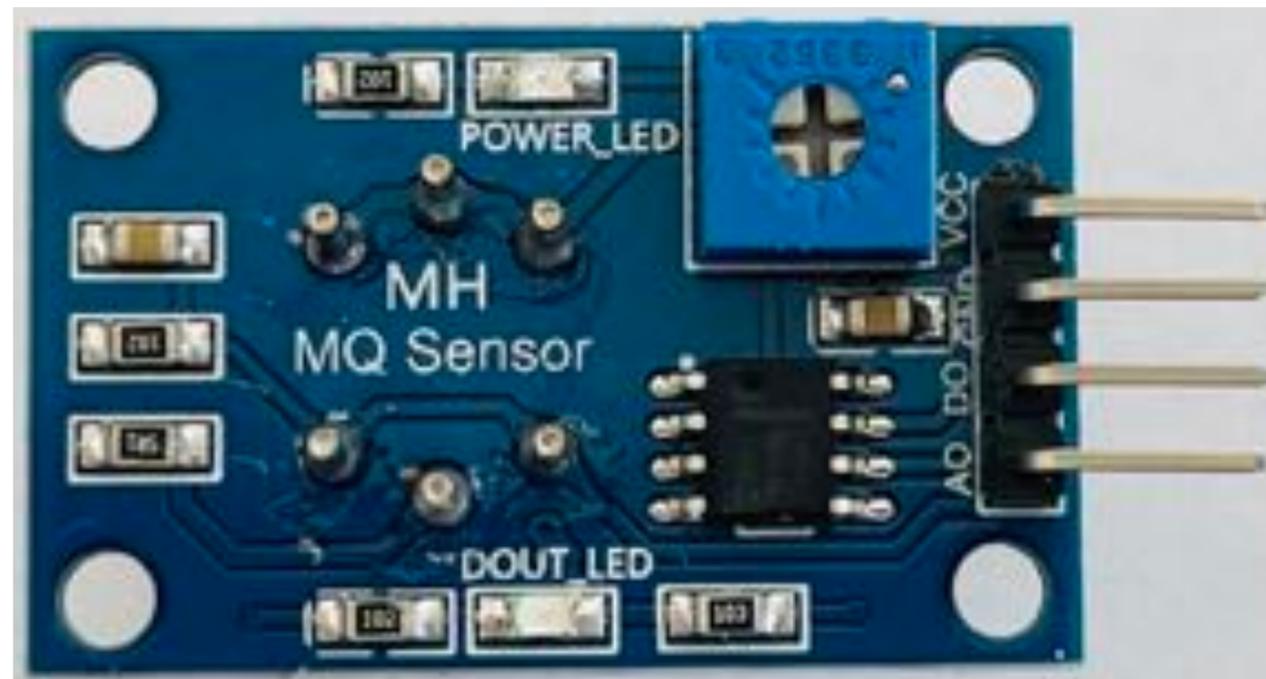
MQ-2: Methane, Butane, LPG, Smoke

MQ-3: Alcohol, Ethanol, Smoke

MQ-9: Carbon Monoxide, flammable gasses

There are several more

MQ Sensors Pin Out



Voltage
Ground
Digital Out
Analog Out

```
forever
  set 1 to analog read pin P0
  pause (ms) 100
  set 2 to analog read pin P0
  pause (ms) 100
  set 2 to analog read pin P0
  pause (ms) 100
  set Total to 1 + 2 + 3
  set Mean Reading to Total ÷ 3
  show number round Mean Reading
  pause (ms) 2000
```

The image shows a Scratch script within a 'forever' loop. It consists of the following blocks: a 'set 1 to analog read pin P0' block, a 'pause (ms) 100' block, a 'set 2 to analog read pin P0' block, another 'pause (ms) 100' block, a third 'set 2 to analog read pin P0' block, a fourth 'pause (ms) 100' block, a 'set Total to 1 + 2 + 3' block, a 'set Mean Reading to Total ÷ 3' block, a 'show number round Mean Reading' block, and a final 'pause (ms) 2000' block. The 'set 2 to analog read pin P0' block is highlighted with a red box, and the 'set Total to 1 + 2 + 3' and 'set Mean Reading to Total ÷ 3' blocks are highlighted with a purple box.

1. We are connecting the sensor with the Microbit - VCC to 3v, Gnd to Gnd, and Analog Out to Pin-0
2. We are taking three readings from the MQ sensor using 'Analog Read Pin' command
3. We are storing the three readings into three variables (1, 2, 3)
4. Then we are taking the average of these three readings and displaying that.

1. Once we know the average reading in normal circumstances, we should introduce the change factor.
2. For example, if we are checking for smoke, we should introduce smoke (say by lighting an incense stick) and take fresh average reading.
3. Using the base (normal) reading and the reading with the change factor (smoke) we can write code to create the project we want.
4. For example, to make a Smoke Detection alarm, we can use a conditional statement

IF sensor detects <smoke present reading> THEN sound alarm

IF sensor detects <normal reading> THEN do nothing

Hope you enjoyed
making these projects :-)