

MICROBIT OVERVIEW AND MAKECODE INTRODUCTION

(Time required 30-minutes session)

The micro:bit is a pocket-sized computer that lets you get creative with digital technology. You can code, customize and control your micro:bit from anywhere! You can use your micro:bit for all sorts of unique creations, from robots to musical instruments and more. The micro:bit is the most recent project by the British Broadcasting Corp. (BBC) in an effort to bring computer science education and STEM topics to every student in the United Kingdom. It is an open development board that works in sync with onboard hardware components to get you started down the path of programming hardware.

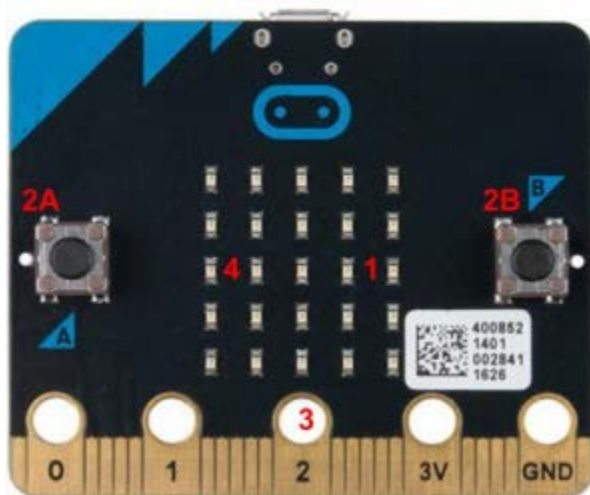


MicroBit components:

The micro:bit has a lot to offer when it comes to onboard inputs and outputs. In fact, there are so many things packed onto this little board that you would be hard-pressed to really need anything else if your goal is to explore the basics of programming and hardware.

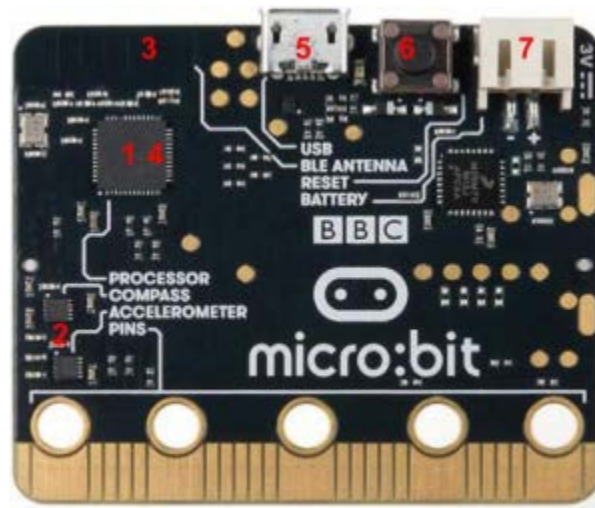
FRONT:

1. LED Array --- The micro:bit has a 5x5 LED array that you can use as a tiny screen to draw on and display words, numbers and other information.
2. A/B Buttons --- Two buttons in all of their clicky glory: A is on the left, B is on the right, and both are prime for controlling a game of your design.
3. Edge "Pins" --- The gold tabs at the bottom of the board are for hooking up external components. The tabs with larger holes can be easily used with alligator clips to prototype things quickly!
4. Light Sensor --- A bit of a hidden gem. The LED array doubles as a light sensor!



BACK:

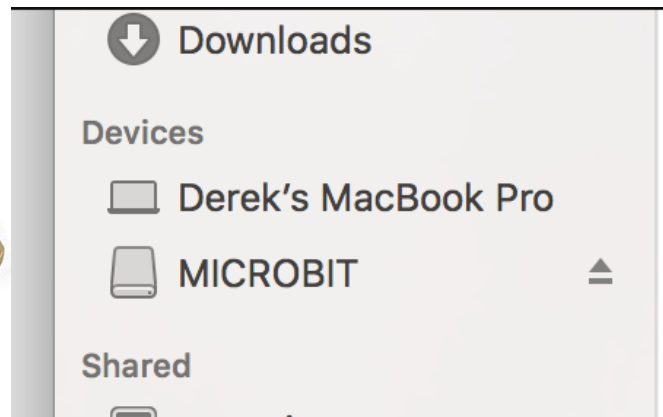
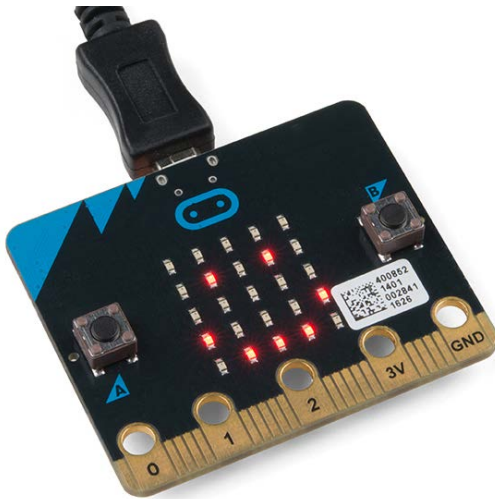
1. Microcontroller --- The brains of the outfit. The micro:bit is powered by a 16MHz ARM Cortex-M0 microcontroller with 256KB Flash and 16KB RAM.
2. Accelerometer/Compass --- The micro:bit has an onboard accelerometer that measures gravitational force, as well as a compass that can detect its orientation using Earth's magnetic field.
3. Bluetooth/Radio --- Communication is huge with the micro:bit. You can communicate with your phone or tablet using BLE or between two or more micro:bits using the standard "radio."
4. Temperature Sensor --- No, the drawing is not numbered incorrectly! The microcontroller doubles as a temperature sensor!
5. USB Port --- Used to upload code to your micro:bit or power from your computer or laptop.
6. Reset Button --- A button to reset your micro:bit and start your code over from the beginning.
7. JST Battery Connector --- A connector to hook up an external battery pack to your micro:bit.

**Connection:**

The micro:bit uses a microUSB cable to hook up to your computer or Chromebook. It is as simple as plugging the cable into your micro:bit and the other end into an open USB port. Once you plug your board in, you should see the small yellow LED on the back of your micro:bit light up and possibly blink a few times. Then whatever existing program that was put on the micro:bit will start running. If this is your first time plugging your micro:bit in, go ahead and play around with it a bit --- push buttons, shake it, and you will get a bit of an Easter egg.

Once your micro:bit boots up, check out your **Finder** if you are on a Mac, or your **My Computer Drives** if you are on a PC. The micro:bit should show up as an external storage device with two files stored in it.

If you are on a Chromebook, when you plug your micro:bit in you will be greeted with a dialog box to open the drive. Feel free to do so to make sure it works!



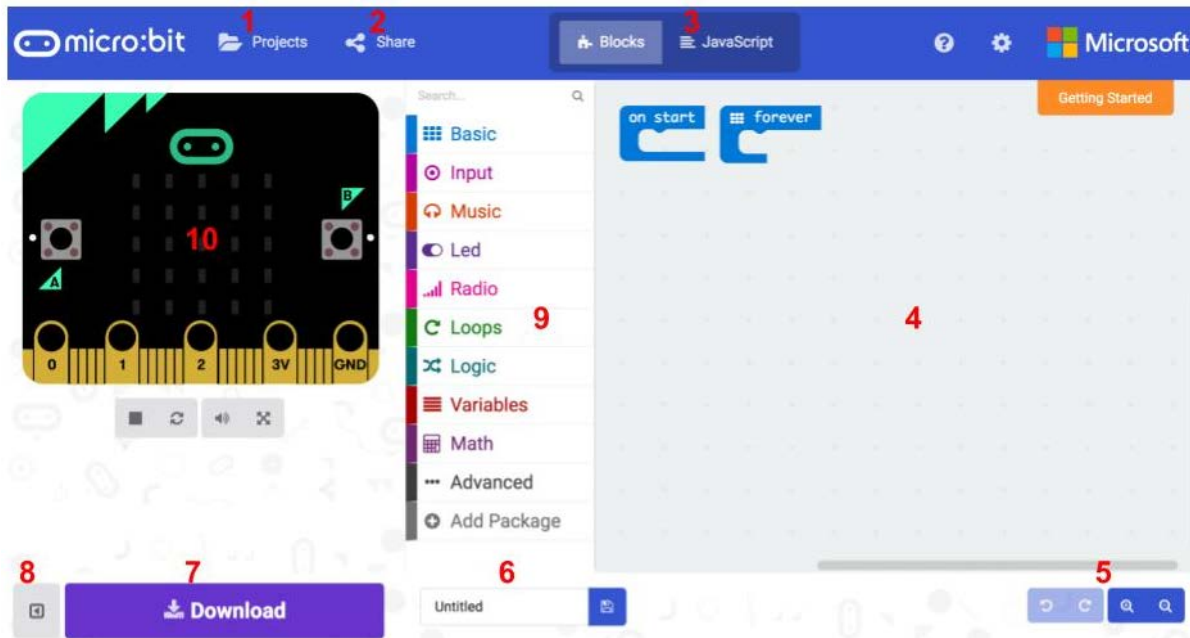
MakeCode Introduction

MakeCode is an open programming environment built by Microsoft for the micro:bit and other similar boards. To open the environment click on the link below.

<https://makecode.microbit.org/>

MakeCode IDE overview

1. **Projects** --- A cloud storage system connected to your computer with no account setup required.
2. **Share** --- Allows you to share your project code in a number of different ways with your friends!
3. **Blocks/JavaScript** --- Choose your own adventure by programming in blocks (default) or in JavaScript.
4. **Program Space** --- This is where the magic happens and where you build your program...where you "make code."
5. **Zoom/Undo-Redo** --- Sometimes you need to undo things, or zoom out and look around; these are the buttons for that.
6. **Name & Save** --- Name your program and save it (download it) to your computer.
7. **Download** --- Similar to Save, download your program as a .hex file and drag it into your micro:bit.
8. **Block Library** --- All of the options in terms of program building blocks, which are color-coded by function.
9. **Simulator Hide/Show** --- You can hide/show the simulator if you would like.
10. **Simulator** --- You don't need hardware! MakeCode has a real-time simulator! As you change your program, you can see what it will do on this virtual micro:bit!

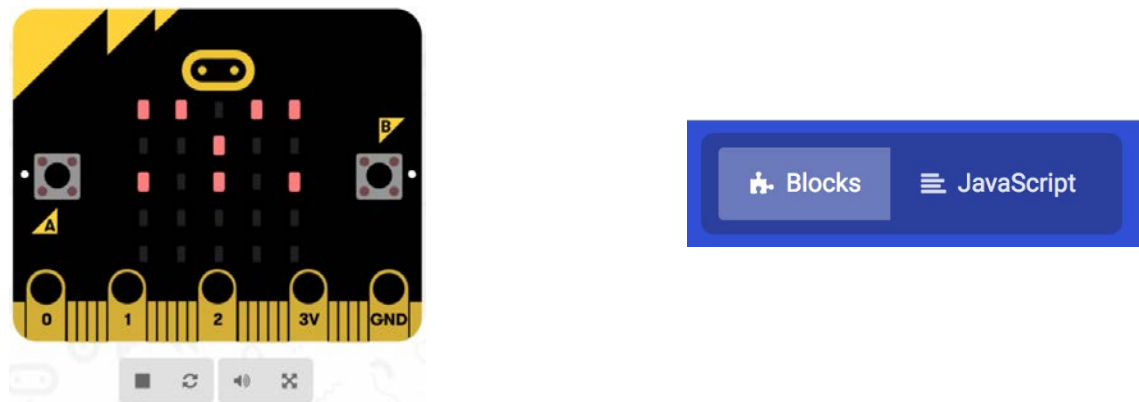


Block or Text Based Programming

We will be using Block-based programming in the majority of the activities. However, switching from block-based and JavaScript is very simple, and you may go back and forth. A nice IDE feature is that the same program will populate the other environment.

Simulator

MakeCode includes a simulator for the micro:bit, meaning if you do not have your micro:bit in hand you can still write code for it. Or if you want to try out an idea before you upload it to your micro:bit, you can do that too!



Congratulations! You have successfully completed this activity.

Reference: Sparkfun Inventor's Kit for micro:bit Experiment Guide

<https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-for-microbit-experiment-guide>

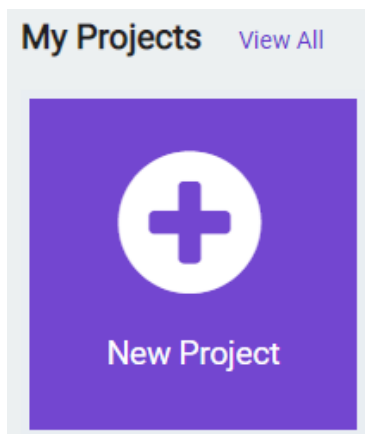
MICROBIT PROGRAMMING (BLOCK-BASED)

(Time required 30-minutes session)

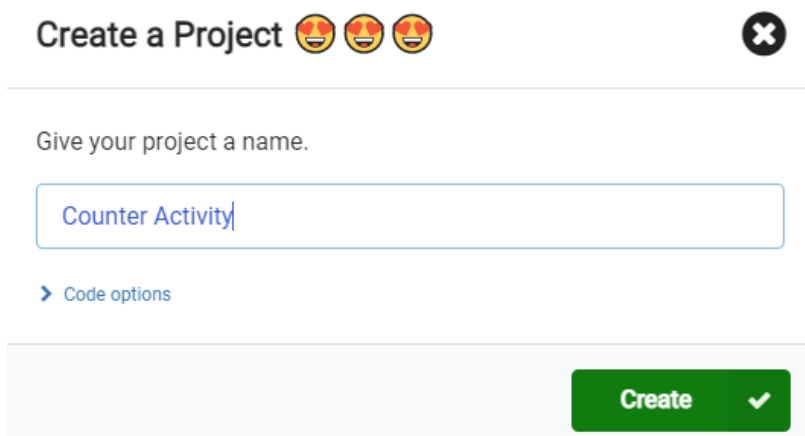
"Hello World" is the term we use to define that first program you write in a programming language or on a new piece of hardware. Essentially it is a simple piece of code that gives you a quick win (fingers crossed) and a first step in learning. For your first "Hello World" we are going to create a simple animation on the LED array that repeats forever.

Building the "Counter Activity"

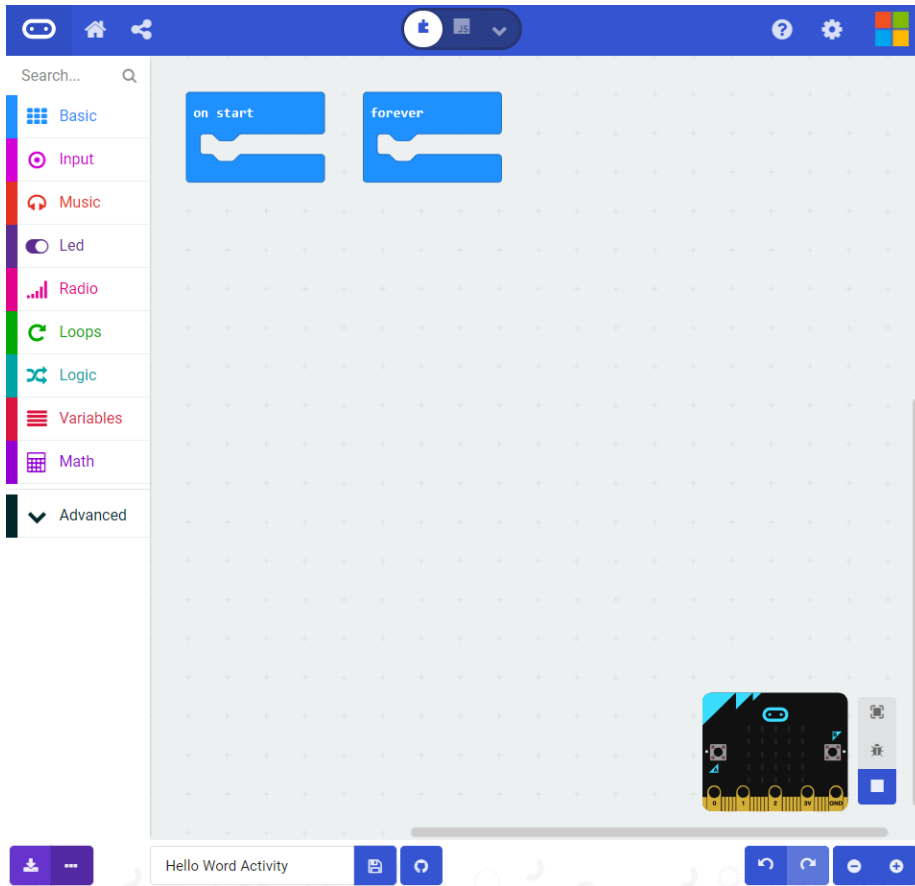
Step 1: Go to <https://makecode.microbit.org/#> and create a New Project



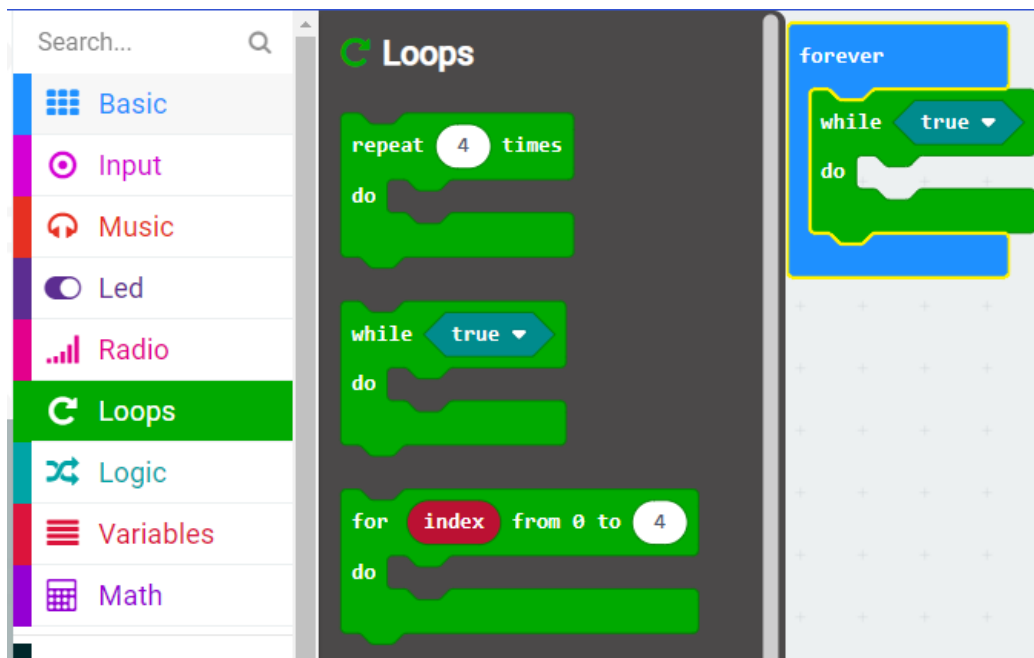
Step 2: Click on New Project and give it a project name – **Hello World Activity** and click Create

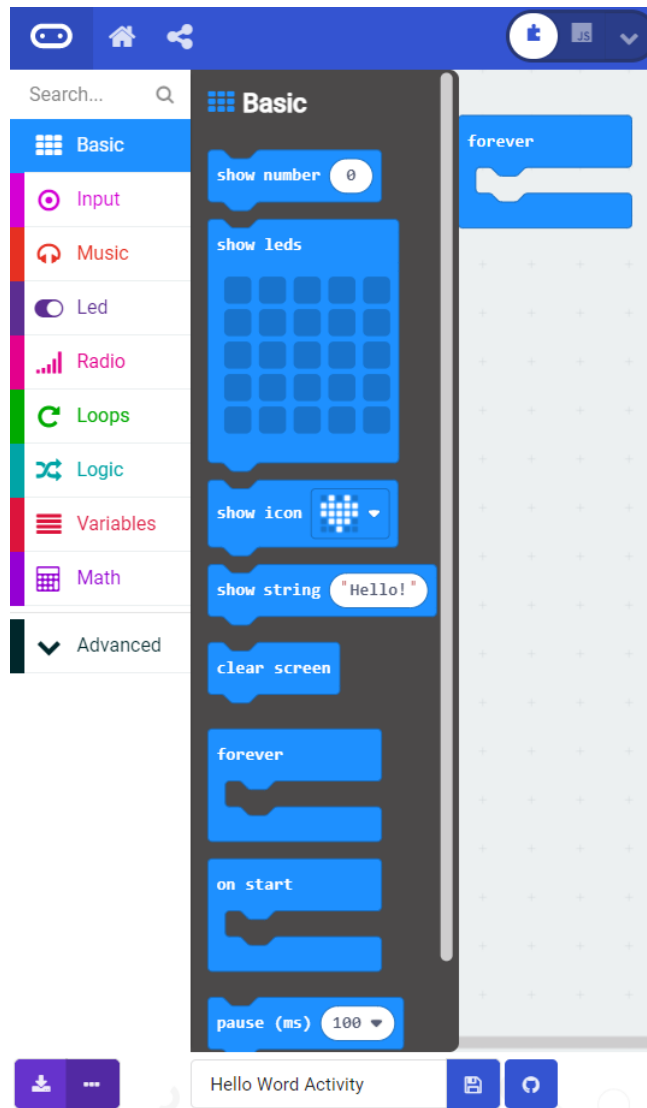
A screenshot of the 'Create a Project' form in the MakeCode web interface. The form has a title 'Create a Project' with three smiley face emojis and a close button (an 'X' in a circle). Below the title is a text input field containing the text 'Counter Activity'. Underneath the input field is a link that says '> Code options'. At the bottom of the form is a green 'Create' button with a checkmark icon.

Step 3: Once the MakeCode is launched you are greeted with two blocks: the On Start and forever. The On Start block is needed to execute all the code at the very beginning of your program and it only executes (run) once. The forever block is code that will loop over and over...it will run forever.



Step 4: Click on the **Loops** category then click and drag the **while <true> do** block into the **forever** block.





Step 5: Click on the **Variables** category and create a variable by selecting Make a Variable. In the New variable name type **counter** as the name then click **ok**

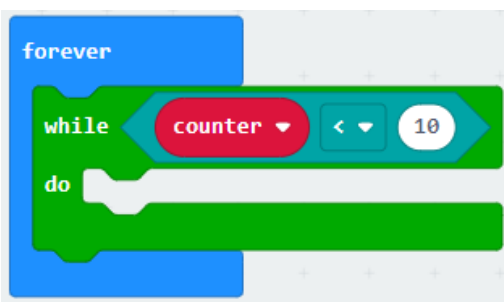
New variable name: ✕

Ok ✓

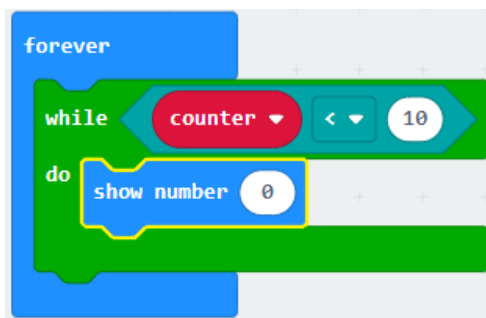
Step 6: Click on the **Logic | Comparison** category then select and drag the **<0> = <0>** block into the **while <true>** block



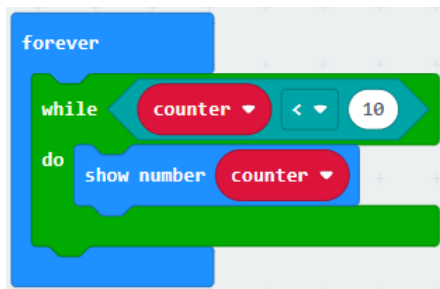
Step 7: Click on the **Variables** category then select and drag the **counter** block into the **while <true>** block. Change the equals to (=) to less than (<) and 0 to 10



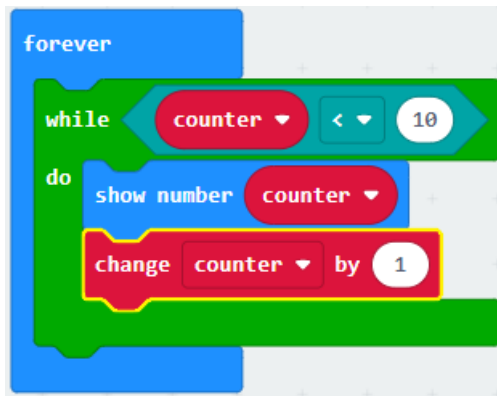
Step 8: Click on the **Basic** category then select and drag the **show number** block into the **while <true>** do block. Change the equals to (=) to less than (<) and 0 to 10



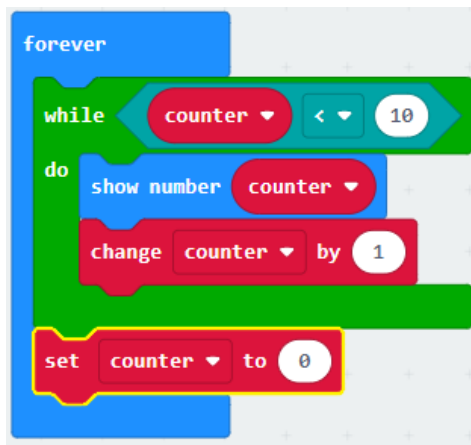
Step 8: Click on the **Variables** category then select and drag the **counter** block into the **show number <0>** block.



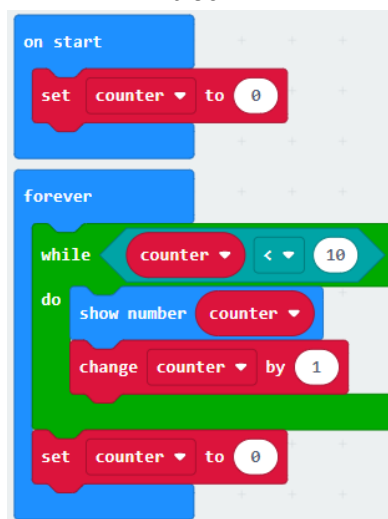
Step 9: Click on the **Variables** category then select and drag the **change counter by 1** block into the **while <true> do** block.



Step 10: Click on the **Variables** category then select and drag the **set counter to 0** block into **forever** block. Make sure it is outside the **while <true> do** block.



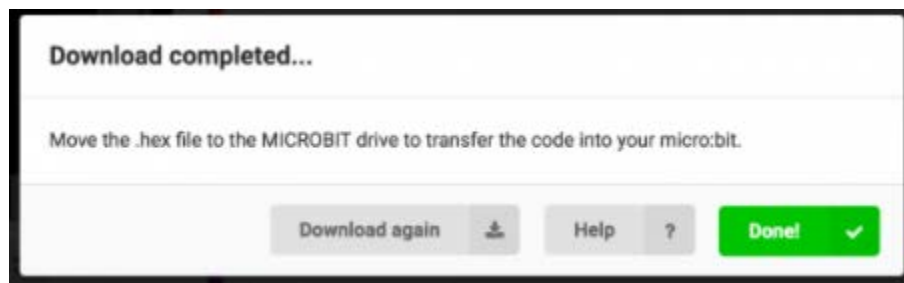
Step 11: Click on the **Variables** category then select and drag the **set counter to 0** block into **on start** block.



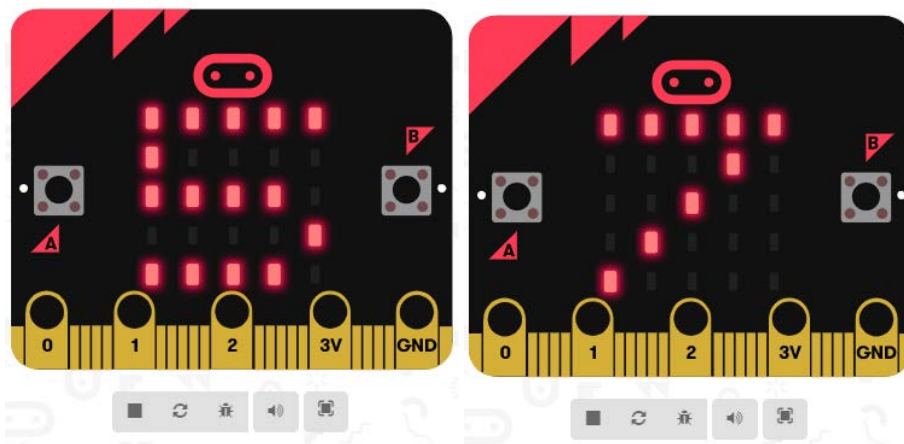
Step 12: Click the download button in the lower lefthand corner of the code window. It will downloaded most probably in the Downloads folder



Step 13: Simply click and drag your program file from its download location to your micro:bit drive, which shows up as an external device.



Step 14: Your micro:bit will flash for a few seconds, and then your program will start automatically.



Step 15: We create an ascending counter; it is your turn to create a descending counter.

Congratulations! You have successfully completed this activity.

Reference: Sparkfun Inventor's Kit for micro:bit Experiment Guide

<https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-for-microbit-experiment-guide>

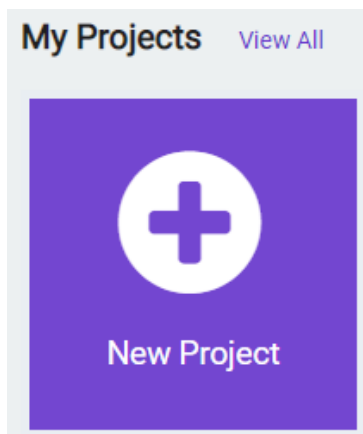
MICROBIT PROGRAMMING (BLOCK-BASED)

(Time required 30-minutes session)

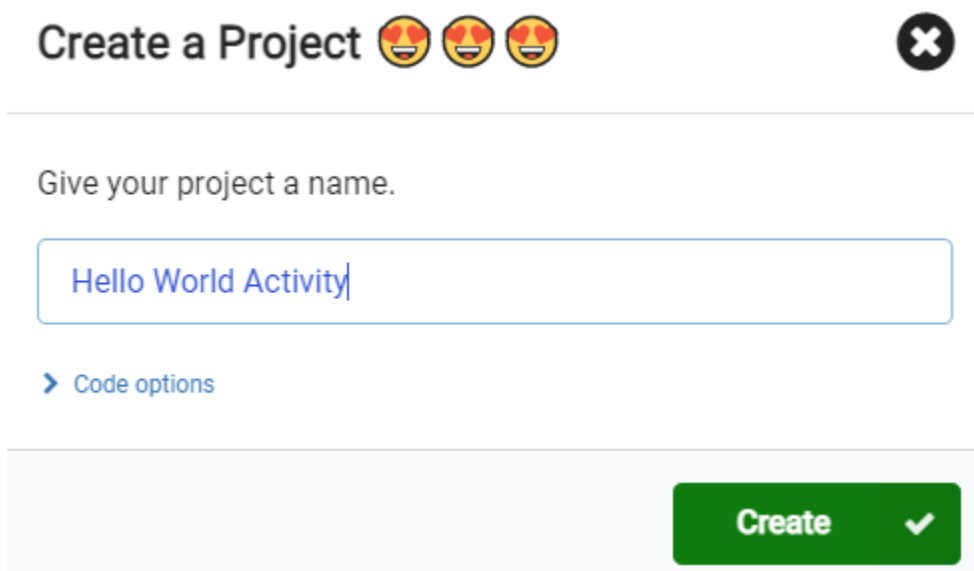
"Hello World" is the term we use to define that first program you write in a programming language or on a new piece of hardware. Essentially it is a simple piece of code that gives you a quick win (fingers crossed) and a first step in learning. For your first "Hello World" we are going to create a simple animation on the LED array that repeats forever.

Building the "Hello World"

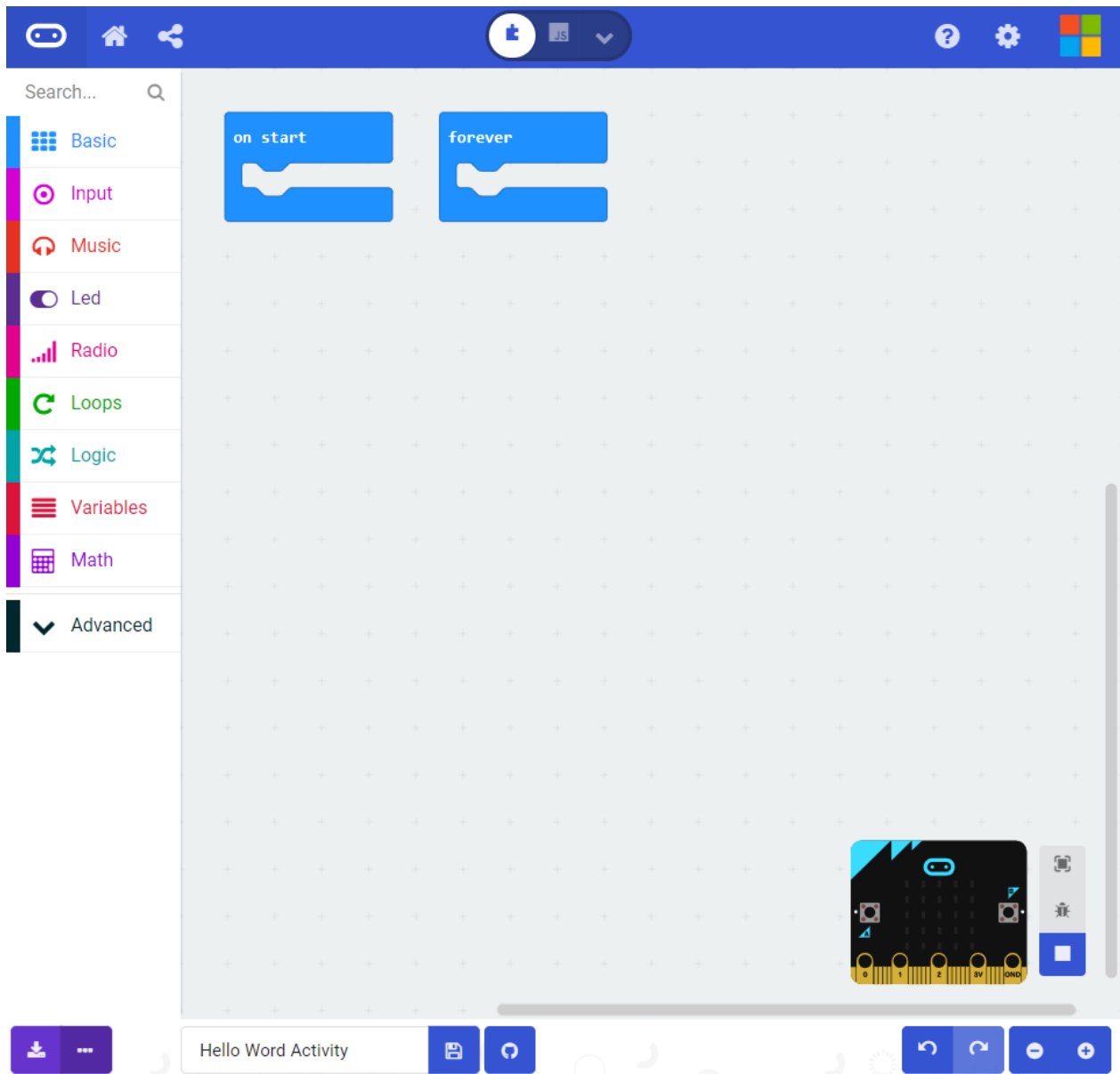
Step 1: Go to <https://makecode.microbit.org/#> and create a New Project



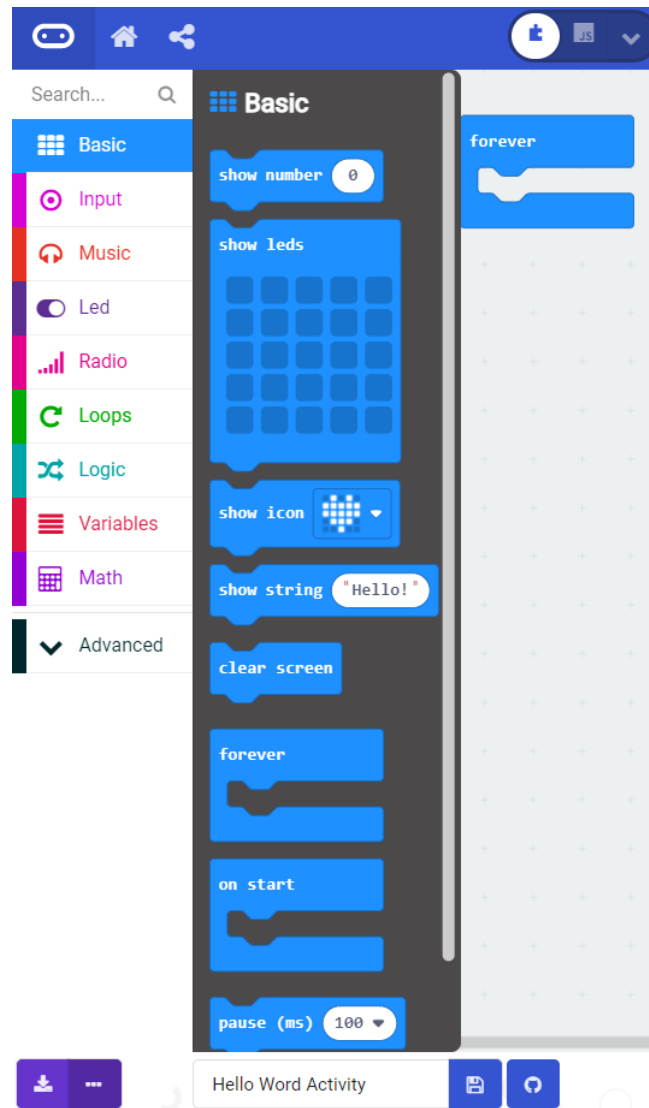
Step 2: Click on New Project and give it a project name – **Hello World Activity** and click Create

A screenshot of the 'Create a Project' form in the MakeCode web interface. The form has a title 'Create a Project' followed by three smiley face emojis and a close button (an 'X' in a circle). Below the title is the instruction 'Give your project a name.' and a text input field containing 'Hello World Activity'. There is a link for 'Code options' with a right-pointing arrow. At the bottom right of the form is a green 'Create' button with a white checkmark.

Step 3: Once the MakeCode is launched you are greeted with two blocks: the On Start and forever. The On Start block is needed to execute all the code at the very beginning of your program and it only executes (run) once. The forever block is code that will loop over and over...it will run forever.



Step 4: click on the **Basic** category. These blocks are, well, the basic building blocks of a BuildCode program. It will expand into a number of options. Click and drag the **show leds** block over and place it inside of your forever block. Notice that the block is keyed to fit inside of the forever block, and if you have the volume up on your computer you will hear a satisfying 'click' noise when you let go of the block.

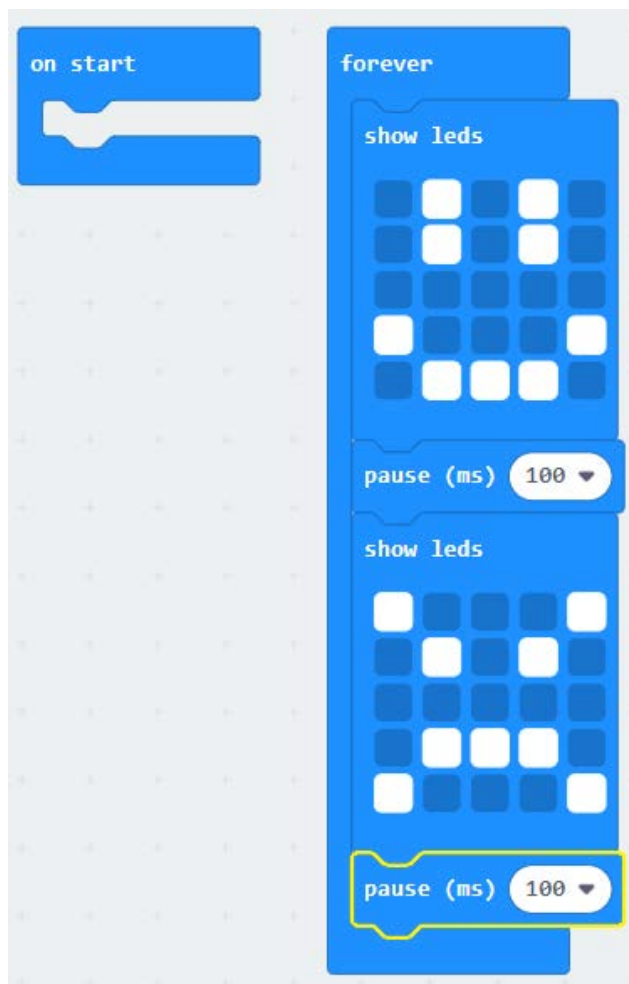


Step 5: The **show leds** block has an array of squares that symbolize the LED array. If you click on a square, it will turn white (program space) and red (simulator), which means that it is on. Draw a simple pixel art shape by turning different LEDs on or off; you should be able to see the outcome in your simulator on the lefthand side of your window.



Step 6: To turn this static image into an animation, we need another **show leds** block to place just under the first block. You can then make a second drawing with this set of rectangles. In your simulator you will see the images switching really, really fast. We need to slow this down!

To slow your animation down, you will use the pause block, which is under the basic block set. The pause block is just what it says; it tells the micro:bit to pause and wait for a certain amount of time. Place two pause blocks in the program as shown.



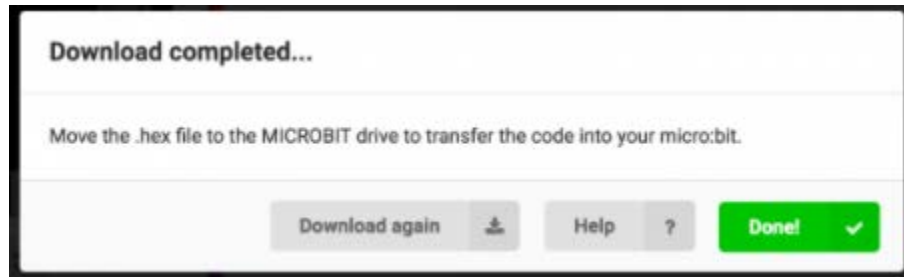
The reason we are using two and placing one at the end is that this program is a loop. Without the block at the end, the image in your animation will change really, really fast.

We have built up an example in the next section where you can download the file and try it out on your own micro:bit, or use the simulator. If you want to play around with the code and make some changes, go ahead.

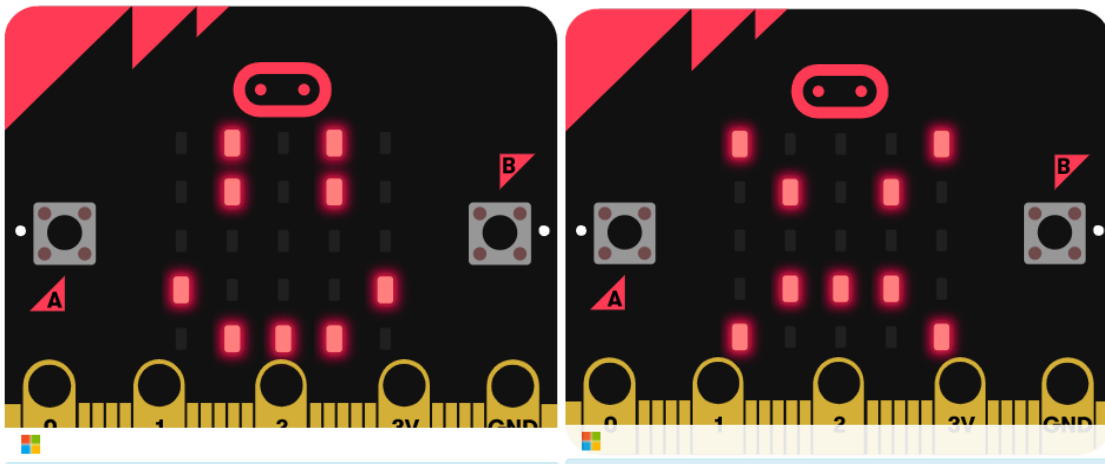
Step 7: Click the download button in the lower lefthand corner of the code window. It will download most probably in the Downloads folder



Step 8: Simply click and drag your program file from its download location to your micro:bit drive, which shows up as an external device.



Step 9: Your micro:bit will flash for a few seconds, and then your program will start automatically.



Congratulations! You have successfully completed this activity.

Reference: Sparkfun Inventor's Kit for micro:bit Experiment Guide

<https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-for-microbit-experiment-guide>

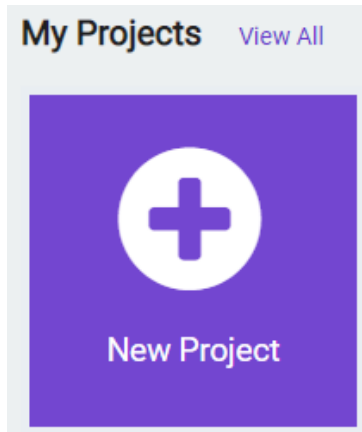
MICROBIT PROGRAMMING (BLOCK-BASED)

(Time required 15-minutes session)

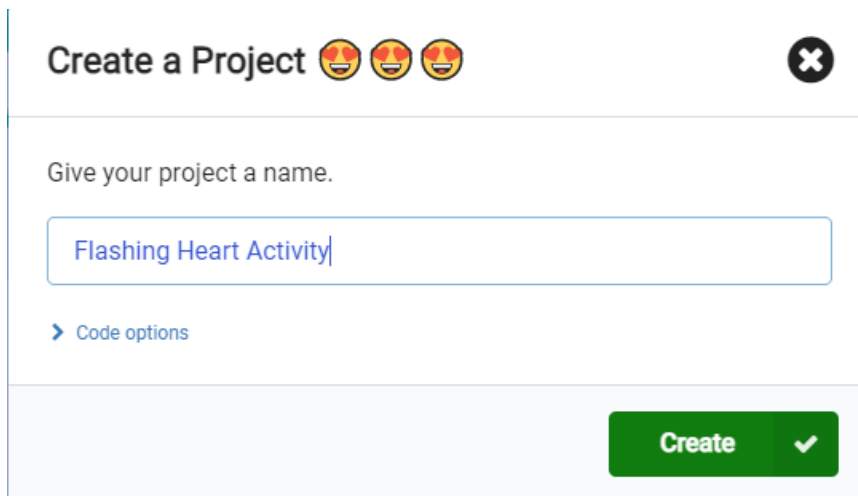
"Flashing Heart" program: simple animation on the LED array that repeats forever.

Building the "Flashing Heard"

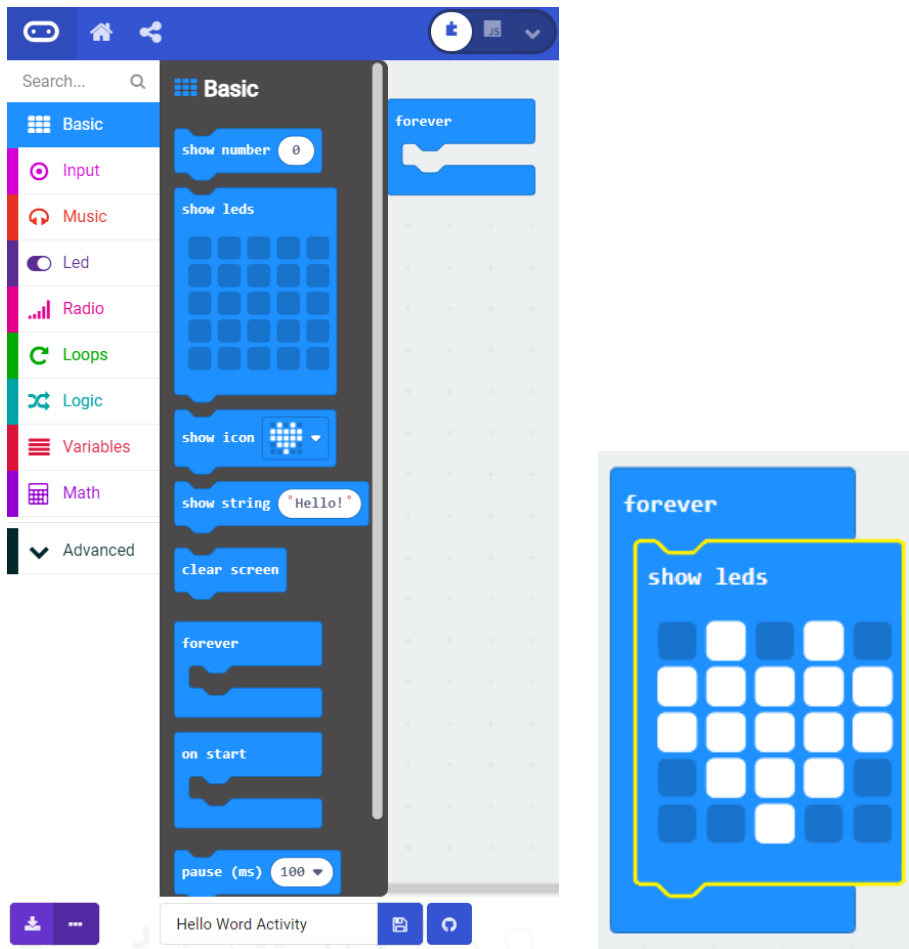
Step 1: Go to <https://makecode.microbit.org/#> and create a New Project



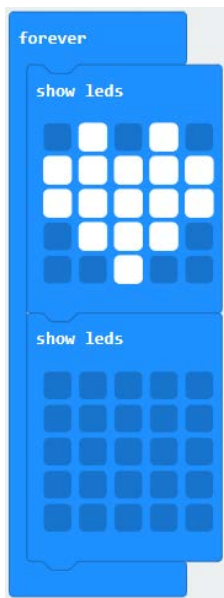
Step 2: Click on New Project and give it a project name – **Flashing Heart Activity** and click Create



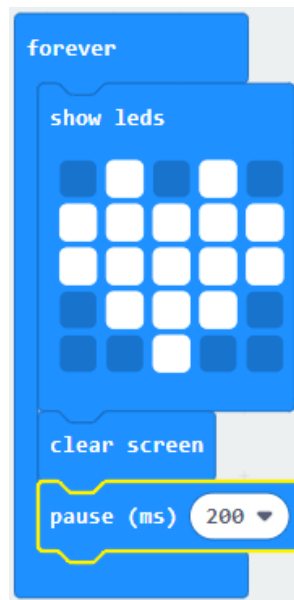
Step 4: click on the **Basic** category. Click and drag the **show leds** block over and place it inside of your forever block. Draw a heart in the block.



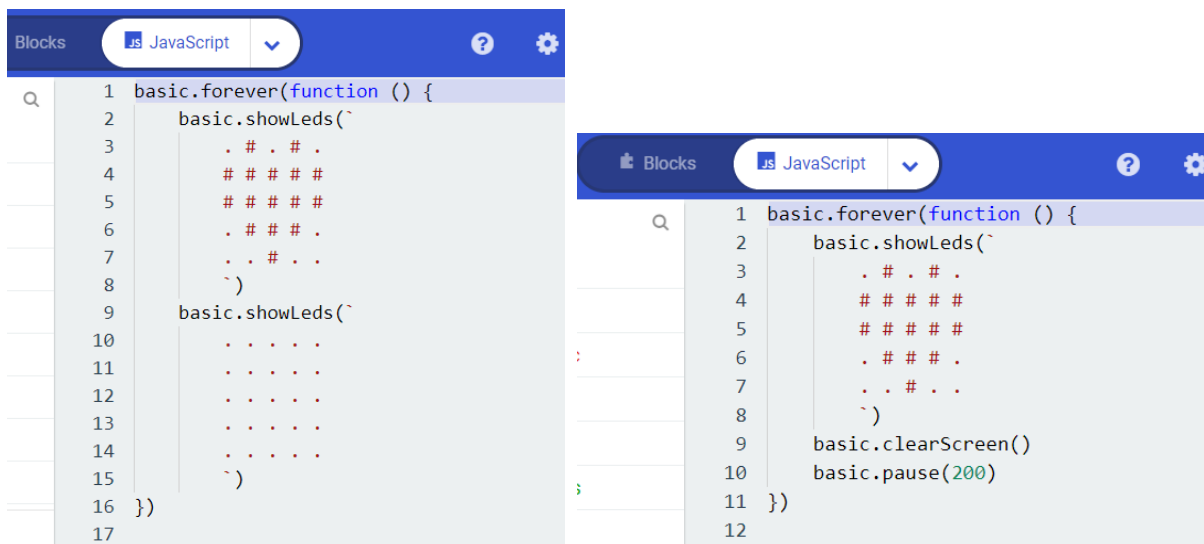
Step 5: Place another **show leds** block. You can leave it blank. By leaving the show leds blank we turn off all leds. An alternative solution is to place the **clear screen** block and place the **pause (ms) 200** block.



OR



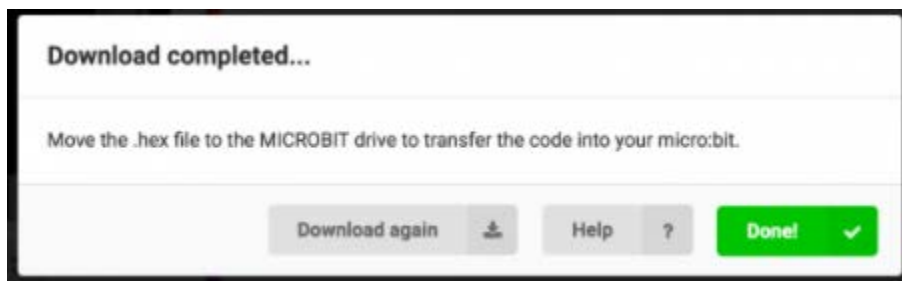
Step 6: Click on the JavaScript slider to convert the block-based program to text-based program



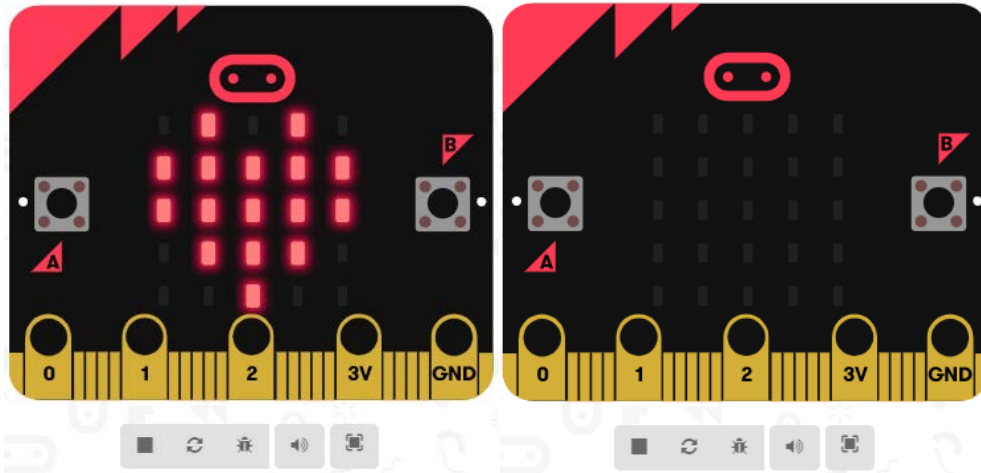
Step 7: Click the download button in the lower lefthand corner of the code window. It will download most probably in the Downloads folder



Step 8: Simply click and drag your program file from its download location to your micro:bit drive, which shows up as an external device.



Step 9: Your micro:bit will flash for a few seconds, and then your program will start automatically.



Step 10: If you want to modify the flashing heart program go ahead, when finish, show it to everyone.

Congratulations! You have successfully completed this activity.

Reference: Micro:bit Projects

<https://makecode.microbit.org/projects/flashing-heart>

MICROBIT PROGRAMMING (BLOCK-BASED)

(Time required 30-45 minutes session)

Introduction

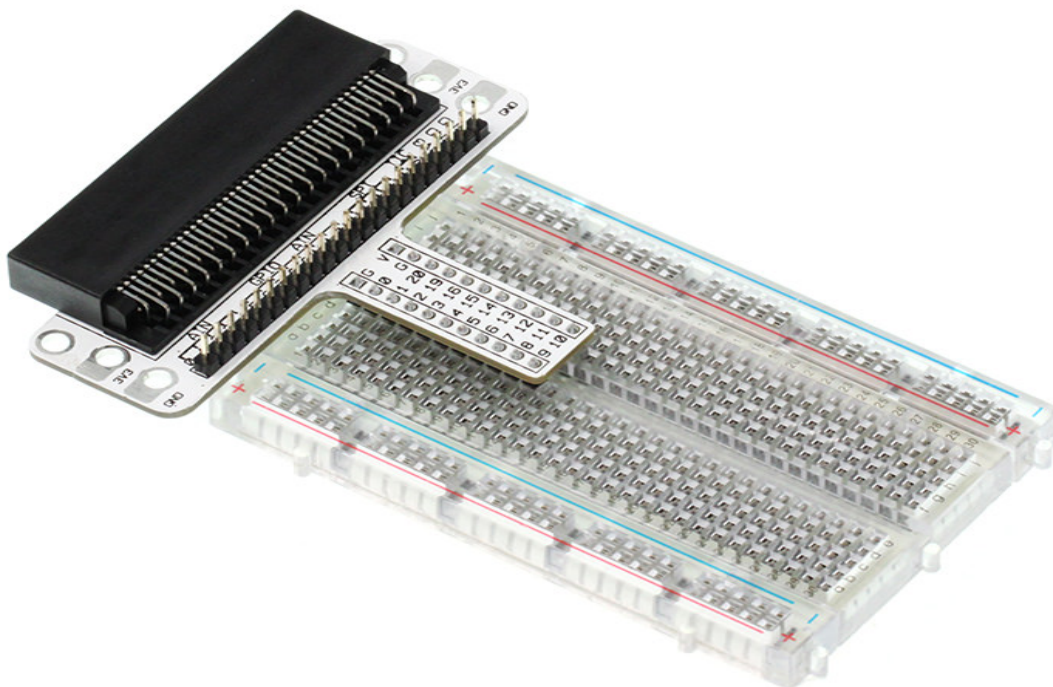
LEDs are small, powerful lights that are used in many different applications. To start off, we will work on blinking an LED, the basic introduction of microcontrollers and building circuits. It's as simple as turning a light on and off. It might not seem like much, but establishing this important baseline will give you a solid foundation as we work toward more complex experiments.

Parts Needed:

- 1x micro:bit
- 1x Micro B USB Cable
- 1x micro:bit Breakout (with Headers)
- 1x Breadboard
- 1x Jumper Wire
- 1x LED
- 1x 100 Ω Resistor

Building the Prototype

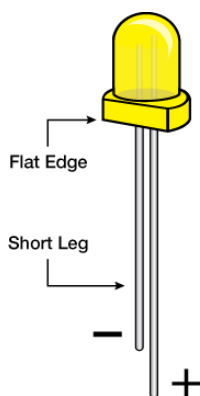
To extend the functionality of the micro:bit we need to use the micro:bit Breakout. In combination with the breadboard, this module makes much more easier to use all the pins on the micro:bit. The breakout board lines up with the pins of a breadboard. We recommend using a full-sized breadboard with this breakout to give you enough room to prototype circuits on either end of the breadboard. Also, for durability's sake, insert the breakout pins about halfway into the breadboard so there is support under the board for when you insert a micro:bit and/or pull it out.



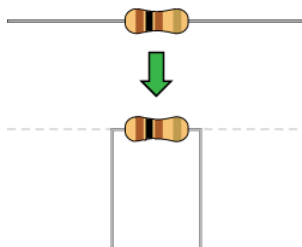
LED Introduction

A Light-Emitting Diode (LED) will only let current through in one direction. Think of an LED as a one-way street. When current flows through the LED, it lights up! When you are looking at the LED, you will notice that its legs are different lengths. The long leg, the "anode," (positive) is where current enters the LED. This pin should always be connected to the current source. The shorter leg, the "cathode," (negative) is the current's exit. The short leg should always be connected to a pathway to ground.

LEDs are finicky when it comes to how much current you apply to them. Too much current can lead to a burnt-out LED. To restrict the amount of current that passes through the LED, we use a resistor in line with the power source and the LED's long leg; this is called a current-limiting resistor. With the micro:bit, you should use a 100 Ω resistor.



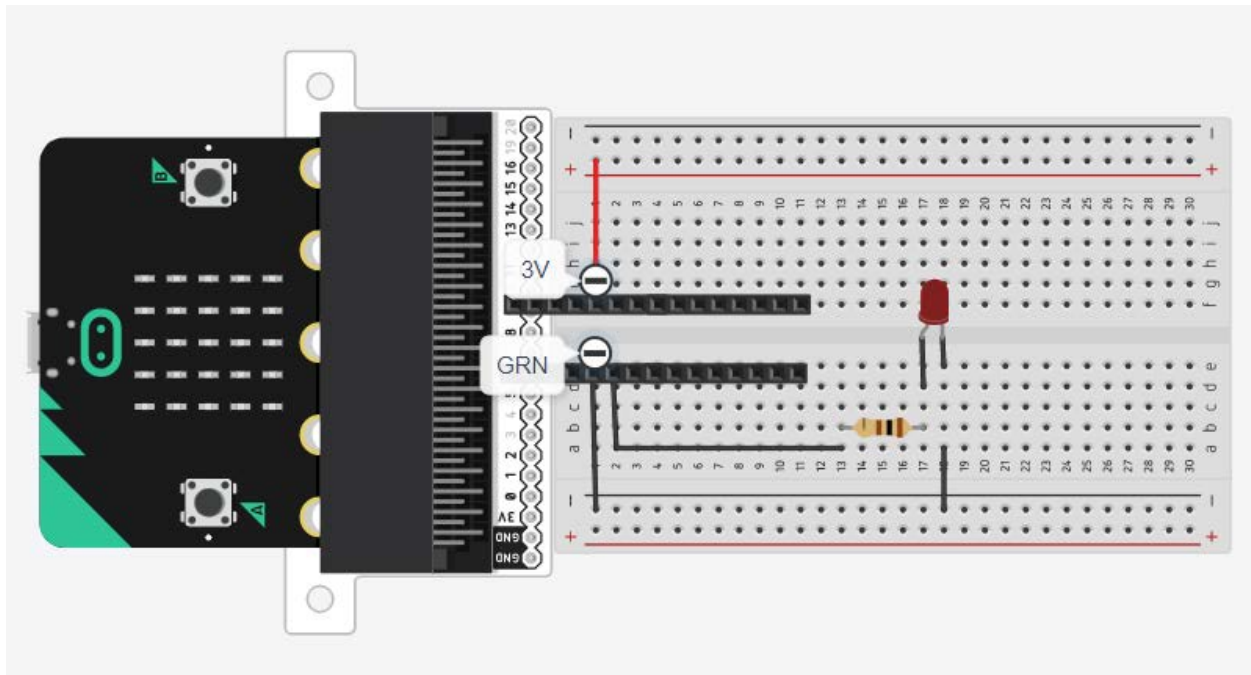
Components like resistors need to have their legs bent into 90° angles in order to correctly fit the breadboard sockets. You can also cut the legs shorter to make them easier to work with on the breadboard.



Blinking LED Wiring Diagram

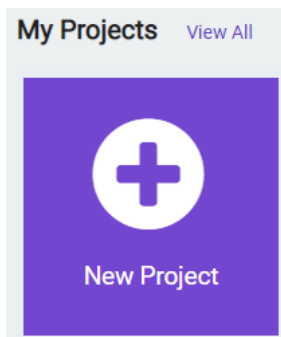
Step 1: Once we connect the micro:bit board into the micro:bit breakout board (make sure the LED arrays are facing up as shown in the figure below).

Step 2: build the circuit diagram as the figure below. Use a black jumper wire to connect the ground pin to the LED cathode (-) and a 100 Ω resistor to the LED anode (+).

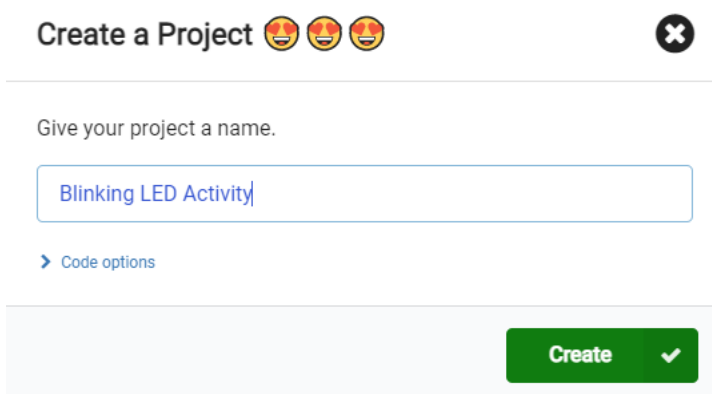


Building the “Blinking LED Activity”

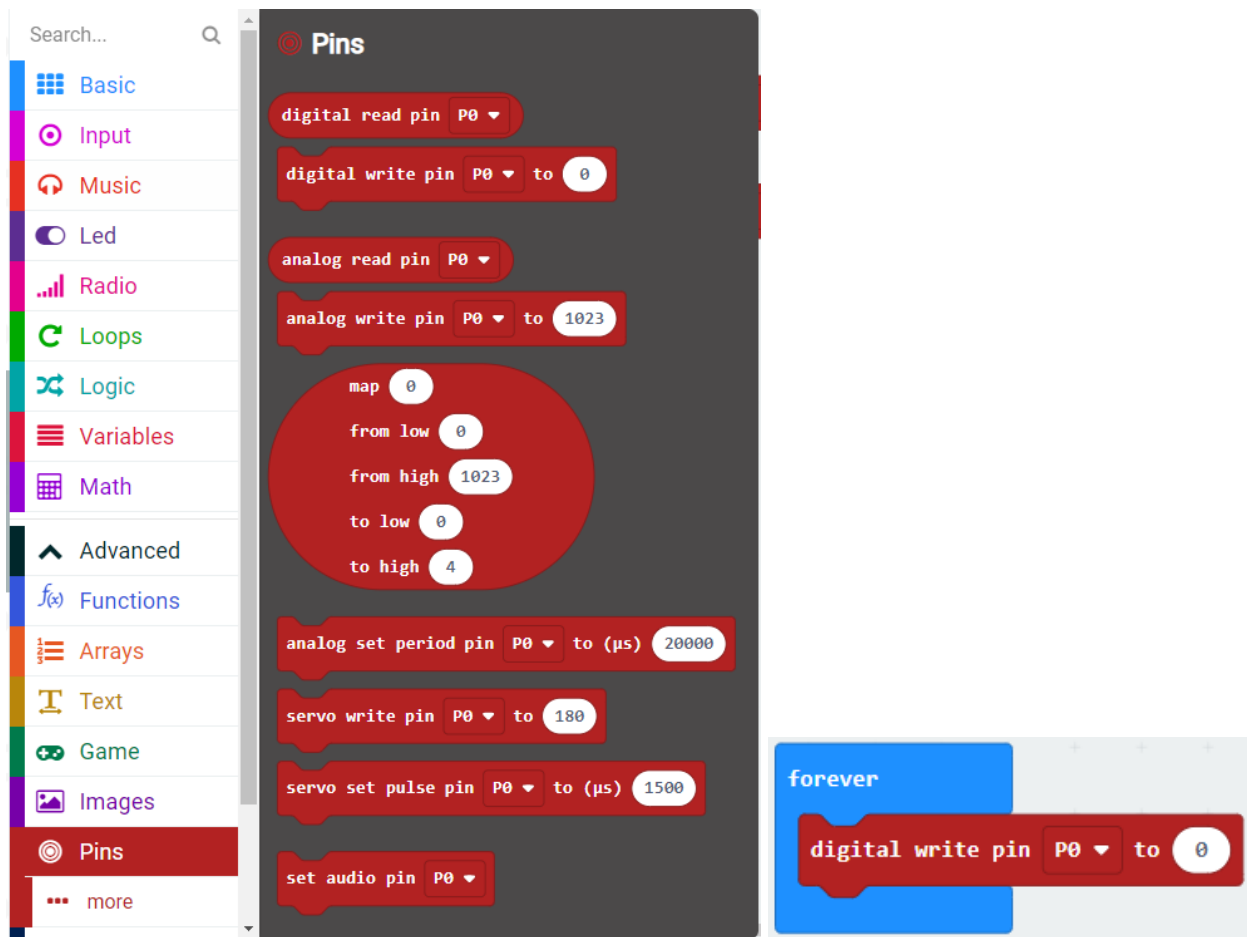
Step 1: Go to <https://makecode.microbit.org/#> and create a New Project



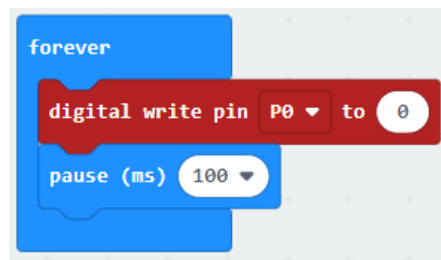
Step 2: Click on New Project and give it a project name – **Blinking LED Activity** and click Create



Step 3: Once the MakeCode is launched, click on the **Advance** category and then select **Pins**. Select and drag the **digital write pin P0 to 0** block into the **forever** block. The Digital write enable us to turn the pin on or off (1 or 0/true or false)



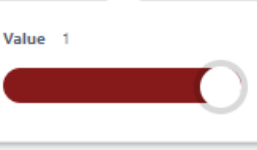
Step 4: Click on the **Basic** category and select and drag the **pause (ms) 100** block into the forever block. Change 100 ms to 1000 ms (1 second)



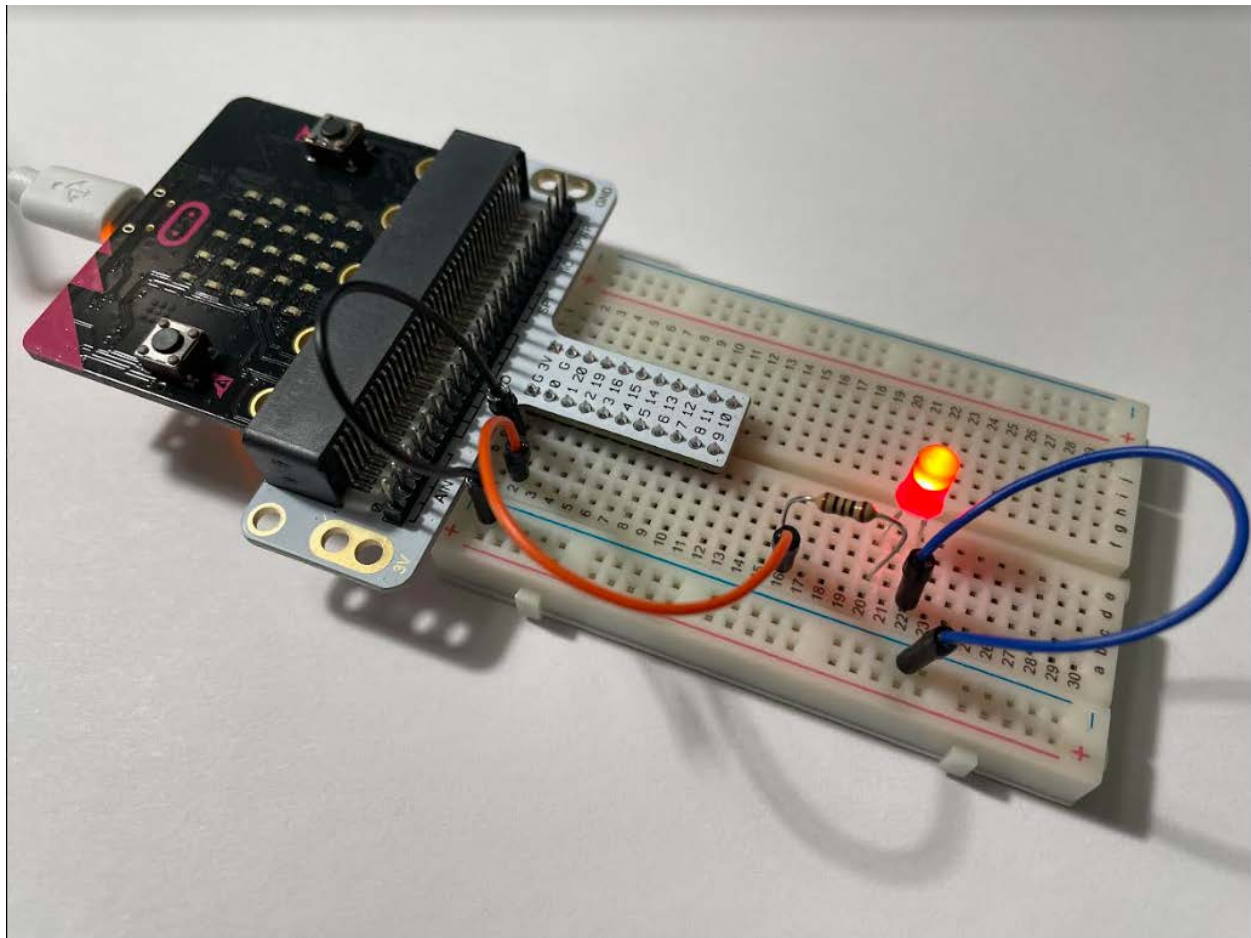
Step 5: Repeat **Step 3** and **4** to add an additional **digital write** and **pause (ms)** blocks. On the second digital write block change the value from 0 to 1 (just type or use the slider value to make the change)

```
forever
  digital write pin P0 to 0
  pause (ms) 1000
  digital write pin P0 to 1
  pause (ms) 1000
```

Value 1



Final Prototype

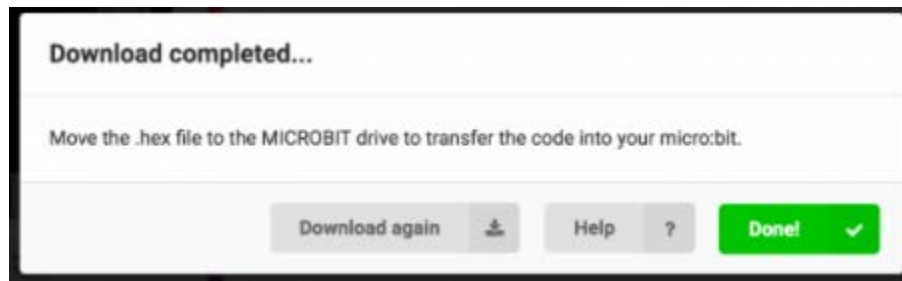


Plug the USB cable to the micro:bit

Step 7: Click the download button in the lower lefthand corner of the code window. It will downloaded most probably in the Downloads folder



Step 8: Simply click and drag your program file from its download location to your micro:bit drive, which shows up as an external device.



Step 9: Your micro:bit will flash for a few seconds, and then your program will start automatically. The LED connected in the breadboard should start blinking.

Step 10: Disconnect the USB cable and make changes to your program. Add an additional LED (connect it to PIN 2 (P2)), your task is to make both LEDs blink, one at the time (when the first LED is on the second LED must be OFF and when second LED is ON, the first LED must be OFF). When you are finished let your mentor know.

Congratulations! You have successfully completed this activity.

Reference: Sparkfun Inventor's Kit for micro:bit Experiment Guide

<https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-for-microbit-experiment-guide>

MICROBIT PROGRAMMING (BLOCK-BASED)

(Time required 30-45 minutes session)

Introduction

In this activity you'll be using a photoresistor, which changes resistance based on how much light the sensor receives. You will read the light value of the room and have an LED turn on if it is dark and turn off if it is bright. That's right; you are going to build a night light!

Parts Needed (included in the kit):

1x micro:bit

1x Micro B USB Cable

1x micro:bit Breakout (with Headers)

1x Breadboard

8x Jumper Wires

1x Photoresistor

1x 10k Ω Resistor

1x LED

1x 100 Ω Resistor

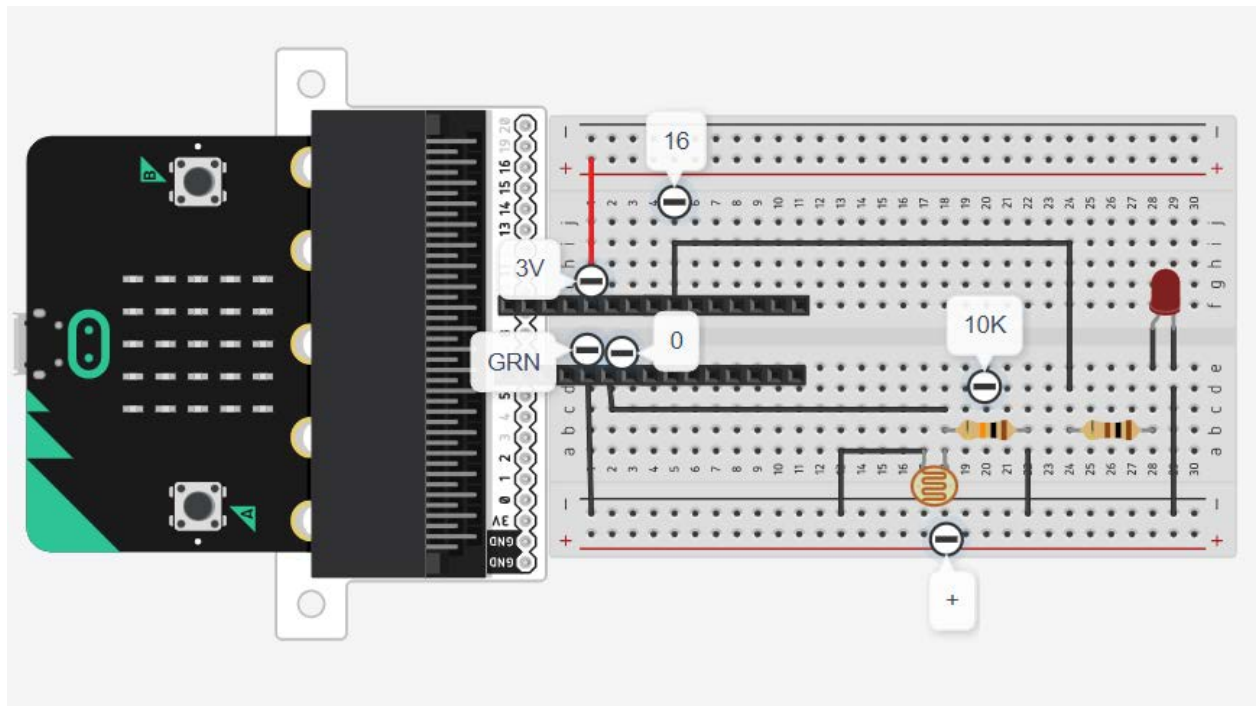
Photoresistor Introduction

The photoresistor changes its resistance based on the light to which it is exposed. To use this with the micro:bit, you will need to build a voltage divider with a 10k Ω resistor. The micro:bit cannot read a change in resistance, only a change in voltage. A voltage divider allows you to translate a change in resistance to a corresponding voltage value.

The voltage divider enables the use of resistance-based sensors like the photoresistor in a voltage-based system.



Photoresistor Wiring Diagram



On Start

The On start block is a block of code that only runs once at the very beginning of your program. In this program we use it to set a calibration value once, and then compare the changing value in the forever loop. This is a great spot for code that you only want to run a single time. To update this value you can press the RESET button on the back of your micro:bit or power cycle the board.

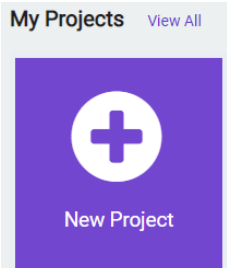
If/Else

If the light value variable that is constantly being updated in the forever block is less than the calibration value minus 50, it is dark and the LED should turn on. The (-50) portion of the if block is a sensitivity value. The higher the value, the less sensitive the circuit will be; the lower the value, the more sensitive it will be to lighting conditions.

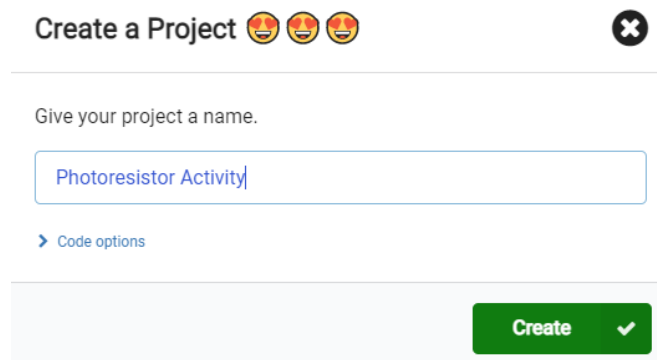
The if block is a logical structure. If the logical statement that is attached to it (item < calibrationVal -50) is true, then it will execute the code blocks inside of the if. If that statement is false, it will execute the else blocks. In this case if the statement is true (the room is dark), then the micro:bit will turn on the LED on pin 16; else (if the room is bright), it will turn the LED off using a digital write block.

Building the “Photoresistor Activity”

Step 1: Go to <https://makecode.microbit.org/#> and create a New Project

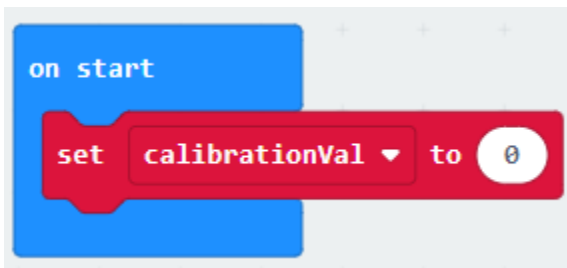


Step 2: Click on New Project and give it a project name **Push Button Activity** and click Create

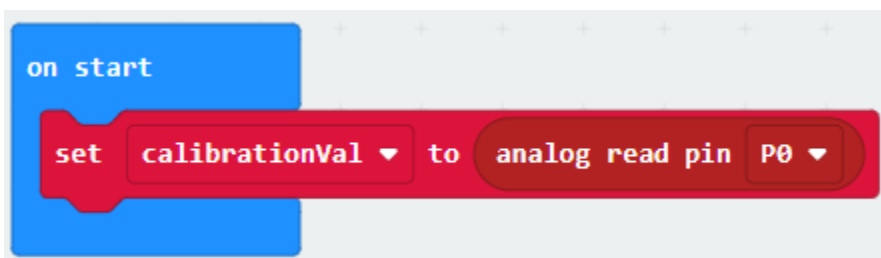


Step 3: Once the MakeCode is launched, click on the **Variables** category and then click on **Make a Variable**. In the New Variable name window type **calibrationVal** then click **Ok**.

Step 4: Once the **calibrationVal** variable has been created select and drag the **set calibrationVal to 0** block into the **on start block**



Step 5: Click on the **Advance | More** category and select and drag the **analog read pin P0** block into the **set calibrationVal to 0** block.



Step 6: click on the **Variables** category and then click on **Make a Variable**. In the New Variable name window type **item** then click **Ok**. Once the **item** variable has been created select and drag the **set item to 0** block into the **forever** block

Step 7: Click on the **Advance | More** category and select and drag the **analog read pin P0** block into the **set item to 0** block.

Step 8: Click on the **Logic | Conditionals** category then select and drag the **if <true> then <> else** block

Step 9: Click on the **Logic | Comparison** category then select and drag the **<0> = <0>** block into the **if <true> then** block. Change the equals to (=) to less than (<)

Step 10: Click on the **Variables** category then select and drag the **item** block into the **if <true> then** block

Step 11: Click on the **Math** category then select and drag the **<0> - <0>** block into the **if <true> then** block

Step 12: click on the **Variables** category then select and drag the **calibrationVal** block into the **<0> - <0>** block. Change the value of 0 to 50.



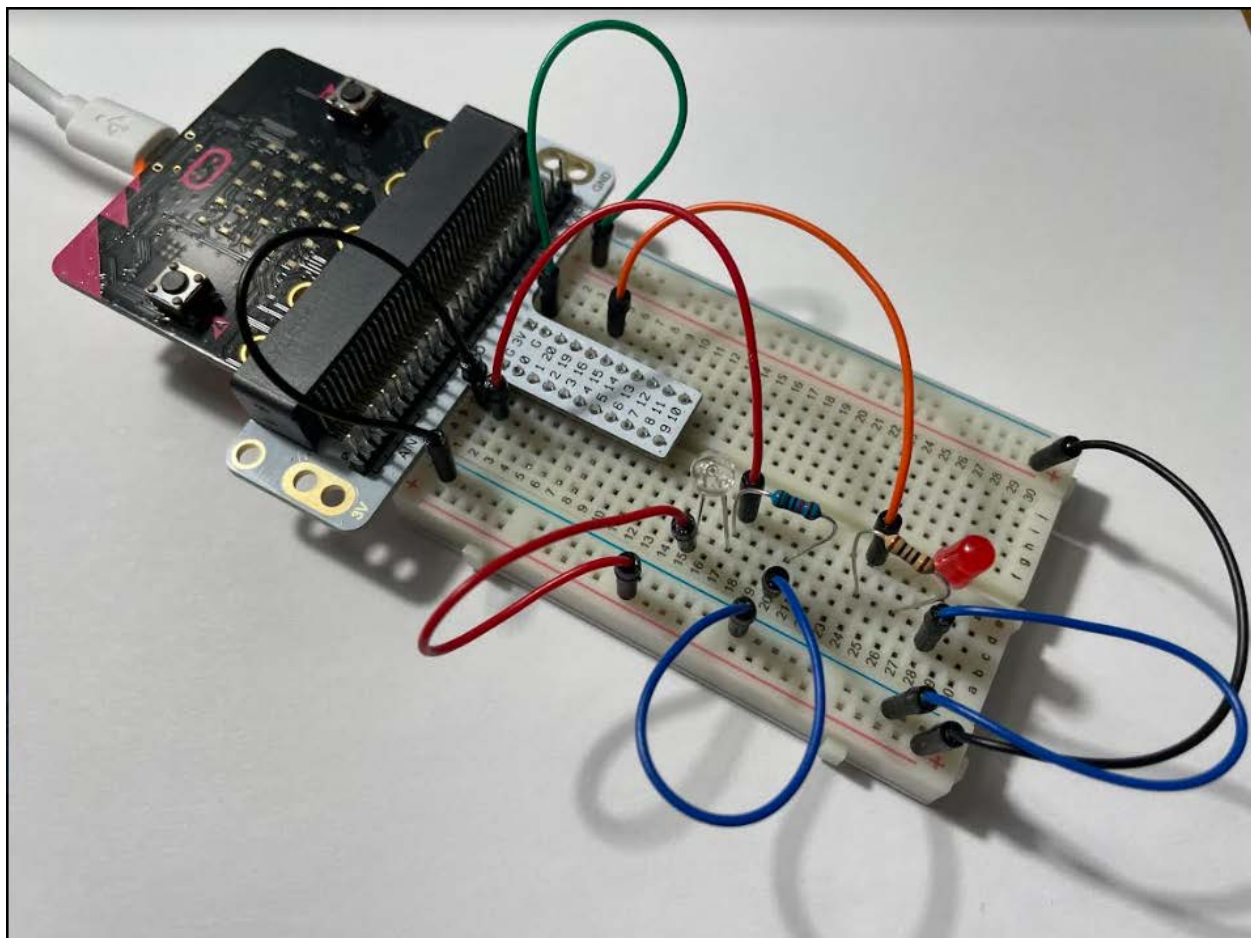
Step 13: Click on the **Advance** category then **Pins** then select and drag the **digital write pin P0 to 0** block into the **if <true> then** block. Change pin **P0** to **P16** and its value to 1.

Step 13: Click on the **Advance** category then **Pins** then select and drag the **digital write pin P0 to 0** block into the **else** block. Change pin **P0** to **P16** and its value to 0

```
on start
  set calibrationVal to analog read pin P0

forever
  set item to analog read pin P0
  if item < calibrationVal - 50 then
    digital write pin P16 to 1
  else
    digital write pin P16 to 0
```

Final Prototype

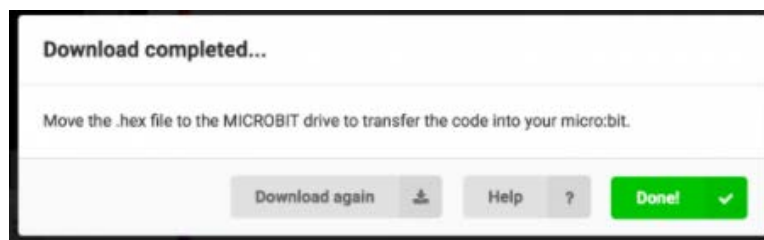


Plug the USB cable to the micro:bit

Step 14: Click the download button in the lower lefthand corner of the code window. It will downloaded most probably in the Downloads folder



Step 15: Simply click and drag your program file from its download location to your micro:bit drive, which shows up as an external device.



Step 16: When the micro:bit runs the program it will take a single reading from the light sensor and use that as a calibration value of the "normal" state of the room. When you place your hand over the light sensor or turn the lights off, the LED will turn on. If you turn the lights back on or uncover the light sensor, the LED will turn off.

Congratulations! You have successfully completed this activity.

Reference: Sparkfun Inventor's Kit for micro:bit Experiment Guide

<https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-for-microbit-experiment-guide>

MICROBIT PROGRAMMING (BLOCK-BASED)

(Time required 30-45 minutes session)

Introduction

In this activity, we will again bridge the gap between the digital world and the analog world. We'll be using a piezo buzzer that makes a small "click" when you apply voltage to it (try it!). By itself that isn't terribly exciting, but if you turn the voltage on and off hundreds of times a second, the piezo buzzer will produce a tone. And if you string a bunch of tones together, you've got music! This circuit and set of code blocks will create a simple sound machine.

Parts Needed (included in the kit):

- 1x micro:bit
- 1x Micro B USB Cable
- 1x micro:bit Breakout (with Headers)
- 1x Breadboard
- 14x Jumper Wires
- 1x Piezo Buzzer
- 2x Momentary Push Buttons
- 2x 10k Ω Resistors

Buzzer Introduction

The buzzer is a small component with a piece of metal in it that moves when you apply a voltage across it. This motion causes a small sound, or "click."

If you turn the voltage on and off fast enough, you get different beeps, squeals, chirps and buzzes. You will use PWM to control the speed of turning the piezo on and off --- and, in turn, the audio frequency coming out of the buzzer. Adjusting the PWM enables you to get legitimate notes out of the buzzer. If you flip the buzzer over and look at the bottom, you will see that one pin has a (+) next to it. That pin gets connected to a signal from the P0 pin. The other pin should be connected to ground.



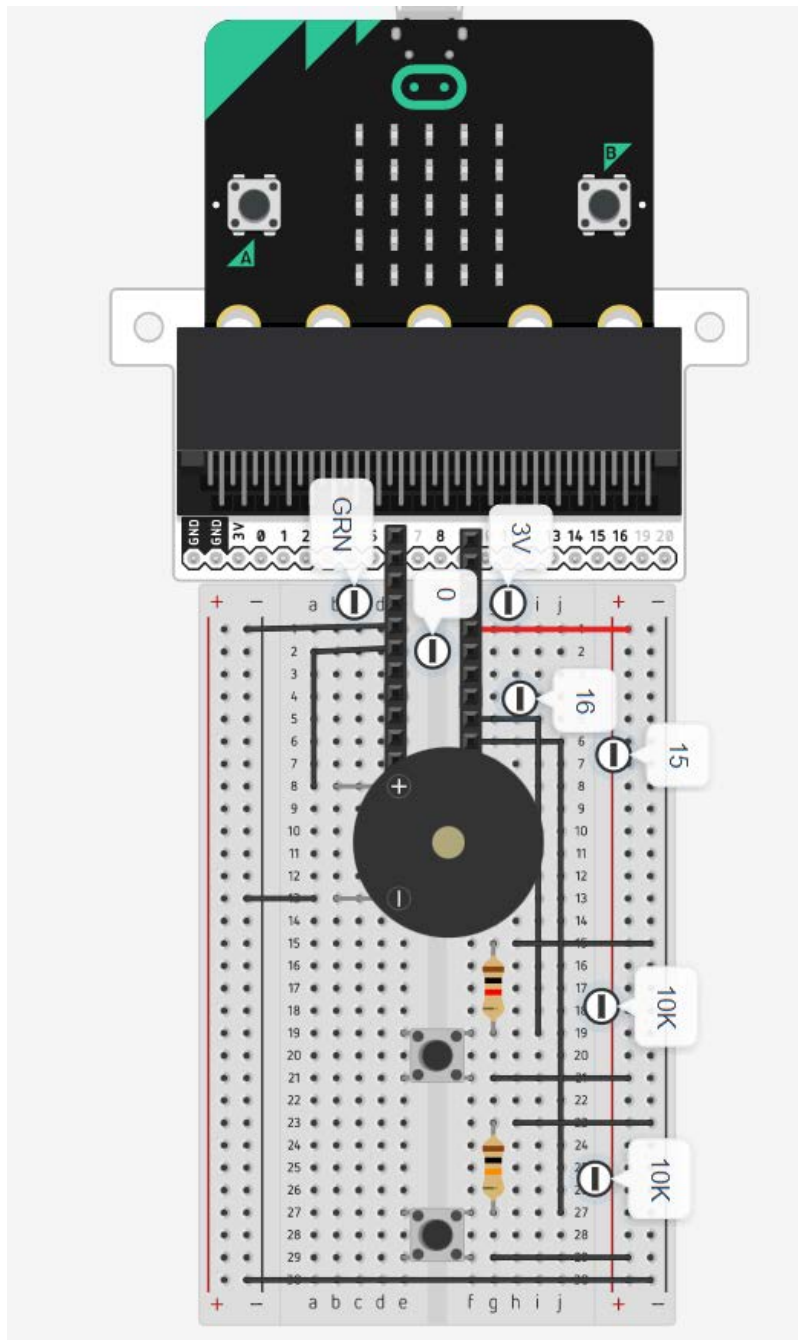
Start Melody Repeating

The Start Melody Repeating block takes all of the frustration out of getting music out of a microcontroller. It is as simple as selecting one of a number of songs that are preprogrammed into MakeCode and how many times you want it to repeat and you are done! Note that when a melody is playing no other code can run, this is called "blocking" code and has to be accounted for you in your program.

Play Tone for

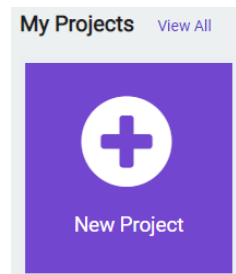
The play tone for block is pretty standard if you are used to making sound with other microcontrollers. For example, tone() function in Arduino is pretty much the same as this block. The play tone for block accepts a note that you would like the buzzer to produce and the length of time in beats per second that you would like it to play. So if you are a musician, you are golden to write horrible robot music for your friends!

Buzzer Wiring Diagram

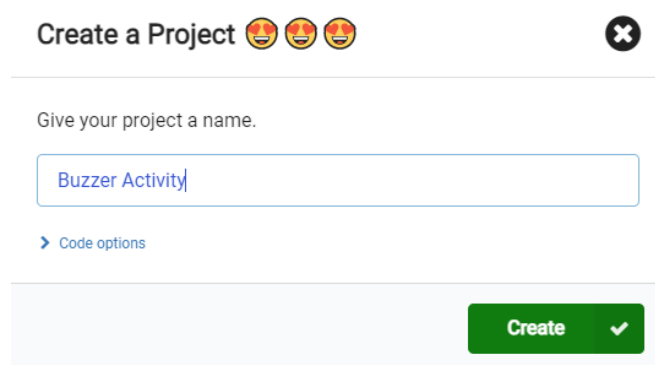


Building the “Buzzer Activity”

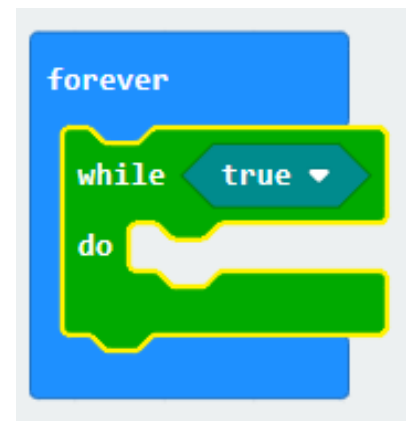
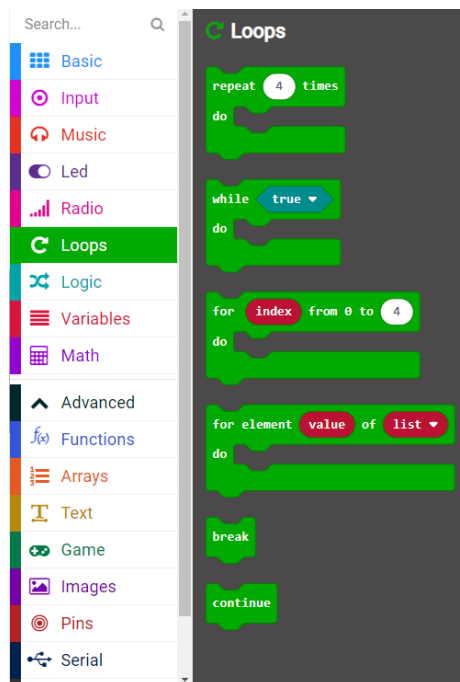
Step 1: Go to <https://makecode.microbit.org/#> and create a New Project



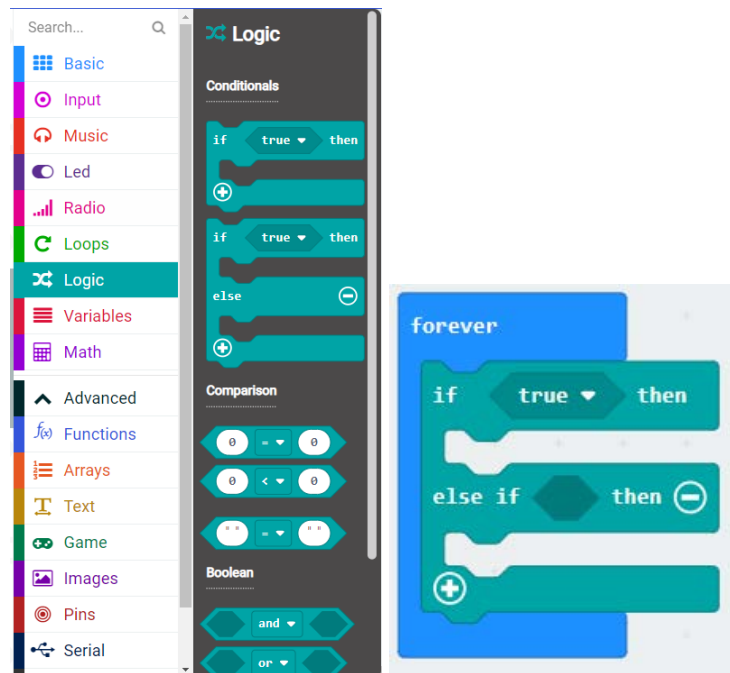
Step 2: Click on New Project and give it a project name **Buzzer Activity** and click Create



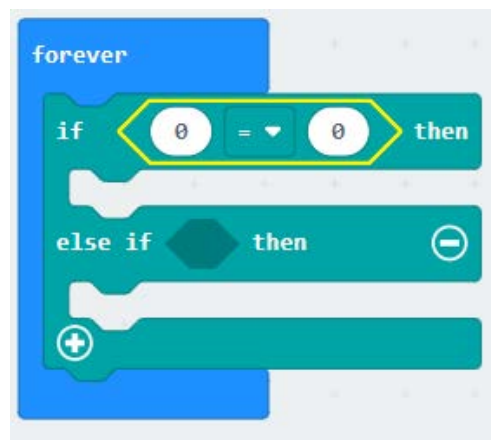
Step 3: Once the MakeCode is launched, click on the **Loops** category and then select and drag **while <true> do** block into the forever block.



Step 4: Click on the **Logic | Conditionals** category and select and drag the **if <true> then else** block into the **while <true> do** block.



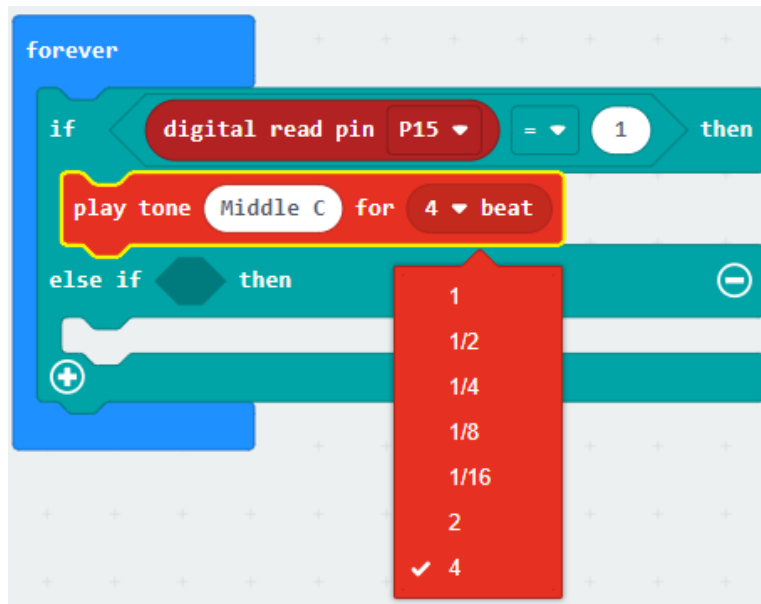
Step 5: Click on the **Logic | Comparison** category then select and drag the **<0> = <0>** block into the **if <true>** block



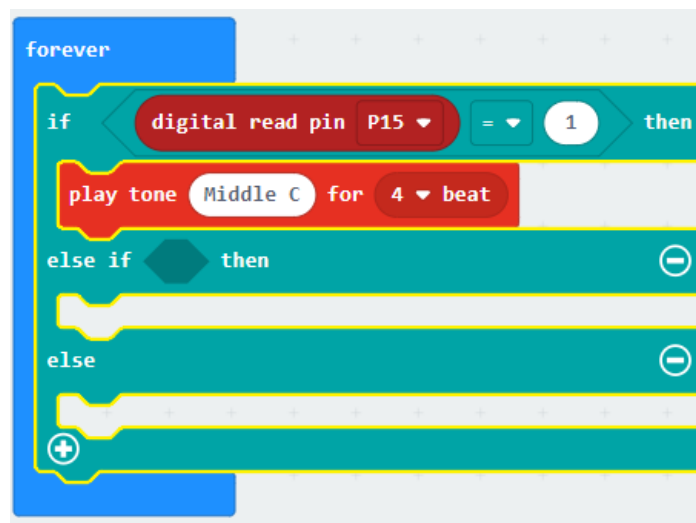
Step 6: Click on the **Advance** category then **Pins** then select and drag the **digital read pin P0** block into the **left comparison** block. Change the pin value to **pin 15 (P15)**. Finally, replace the 0 by 1.



Step 7: Click on the **Music | Tone** category then select and drag the **play tone Middle C for 1 beat** block into the **if <true> then** block. Change the number of repetition to 4 beats.



Step 8: Click on the plus (+) to add of the **if <true> then else** block to add an additional condition for button 2



Step 9: Click on the **Logic | Comparison** category then select and drag the **<0> = <0>** block into the **else if <true> then** block.

Step 10: Click on the **Advance** category then **Pins** then select and drag the **digital read pin P0** block into the **left comparison** block. Change the pin value to **pin 16 (P16)**. Finally, replace the 0 by 1.

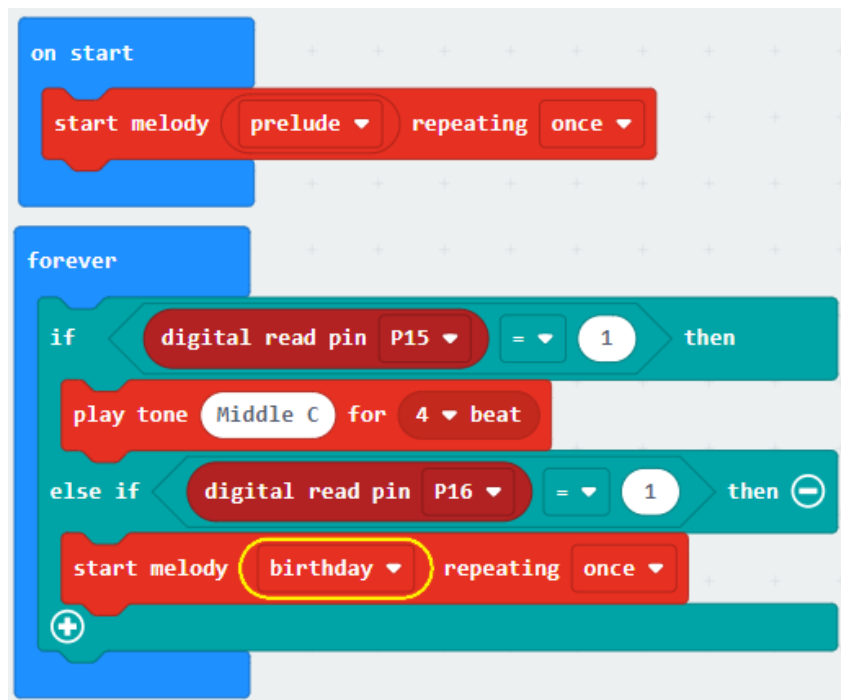
Step 11: Click on the **Music | Melody Advance** category then select and drag the **start melody dadadum repeating once** block into the **if <true> then** block.

Step 12: Click on the minus (-) of the **else** block to remove the else statement block

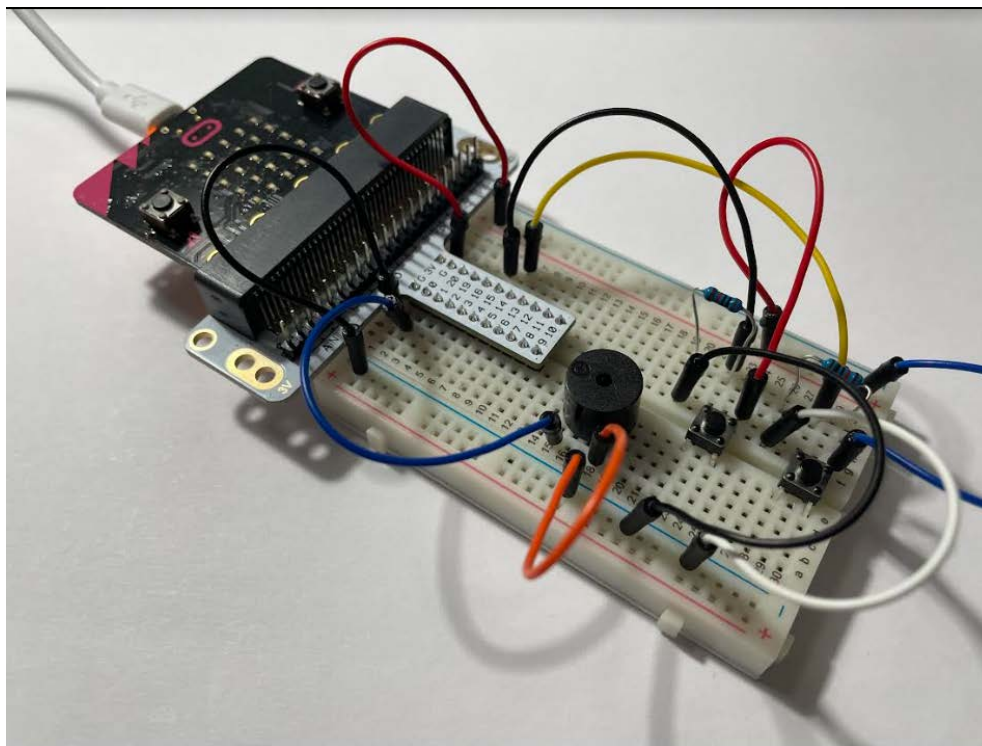
Step 13: Change the melody to birthday



Step 14: Click on the **Music | Melody Advance** category then select and drag the **start melody dadadum repeating once** block into the **on start** block. Change **dadadum** to **prelude repeating once**



Final Prototype

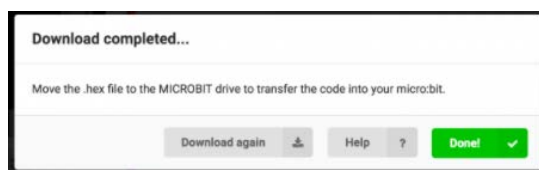


Plug the USB cable to the micro:bit

Step 7: Click the download button in the lower lefthand corner of the code window. It will download most probably in the Downloads folder



Step 8: Simply click and drag your program file from its download location to your micro:bit drive, which shows up as an external device.



Step 9: Your micro:bit will flash for a few seconds, and then your program will start automatically. A prelude melody will start playing once. If you want to hear the birthday melody press the button2 that is connected to P16. If you want to hear Middle C tone press the button1 that is connected to P15

Step 10: Make changes to your program by replacing the melody of your choice.

Congratulations! You have successfully completed this activity.

Reference: Sparkfun Inventor's Kit for micro:bit Experiment Guide

<https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-for-microbit-experiment-guide>

MICROBIT PROGRAMMING (BLOCK-BASED)

(Time required 30-45 minutes session)

Introduction

In this activity, we will be reading one of the most common and simple inputs – a push button – by using a digital input. We will use it to cycle through different colors on the RGB.

Parts Needed (included in the kit):

- 1x micro:bit
- 1x Micro B USB Cable
- 1x micro:bit Breakout (with Headers)
- 1x Breadboard
- 8x Jumper Wires
- 1x Momentary Push Button
- 1x 10k Ω Resistor
- 1x Common Cathode RGB LED
- 3x 100 Ω Resistors

Push Button Introduction

A momentary push button closes or completes the circuit only while it is being pressed. The button has four pins, which are broken out into two sets of two pins. When you press down on the button and get a nice "click," the button bridges the two sets of pins and allows current to flow through the circuit.

How do you know which pins are paired up? The buttons included in this kit will only fit across the breadboard ditch in one direction. Once you get the button pressed firmly into the breadboard (across the ditch), the pins are horizontally paired. The pins toward the top of the breadboard are connected, and the pins toward the bottom of the breadboard are connected.



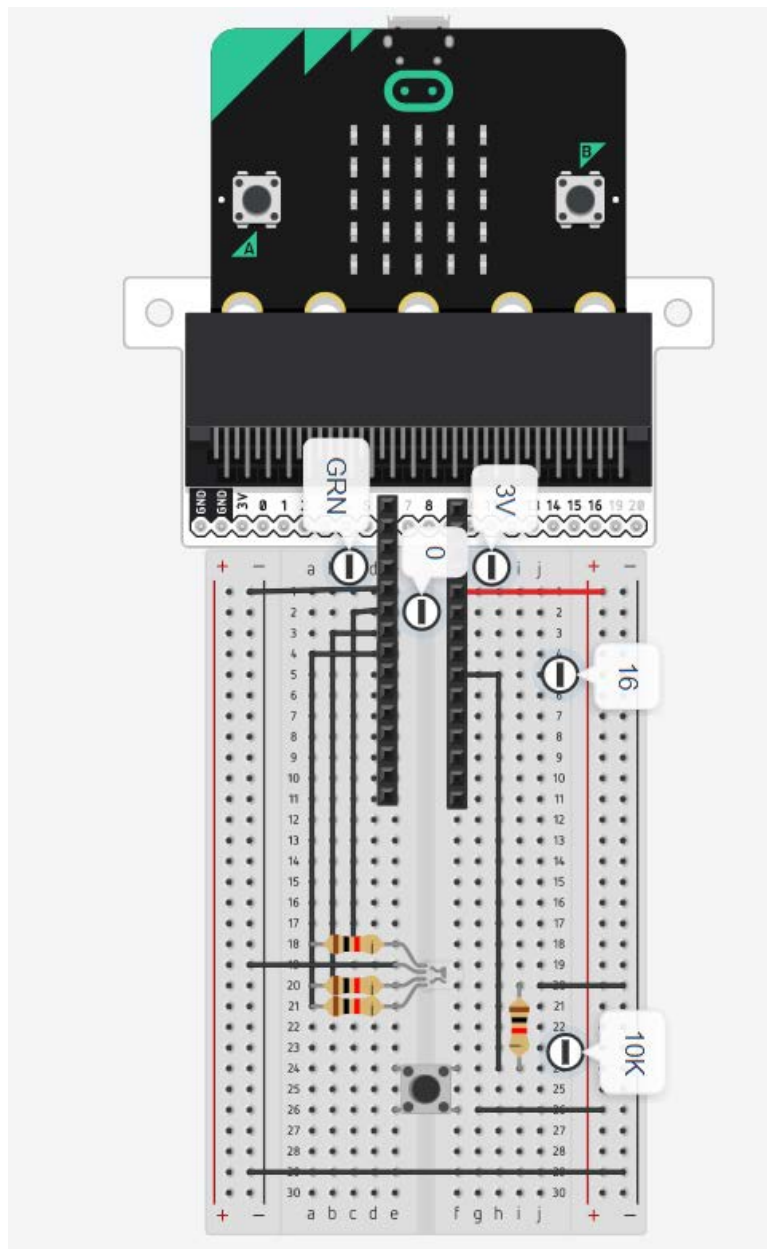
Introduction the Red/Green/Blue (RGB) LED

The RGB has four pins with each of the three shorter pins controlling an individual color: red, green or blue. The longer pin of the RGB is the common ground pin. You can create a custom-colored LED by turning different colors on and off to combine them. For example, if you turn on the red pin and green pin, the RGB will light up as yellow.



- 1 - RED
- 2 - GROUND
- 3 - GREEN
- 4 - BLUE

Push Button Wiring Diagram



Set Pull Pin

When you start your micro:bit, some pins can be set to be naturally on or naturally off. The set pull pin block allows you to set an initial state of a pin by selecting a pin and then its pull state, which is UP, DOWN or NONE.

Set Pin to Emit Event

You use the set pin to emit event block to create a type of event for a specific pin to emit or send out when it reaches that state. As an example, we set P16 to emit an EDGE event, which means that it changed from HIGH to LOW or LOW to HIGH.

On Event

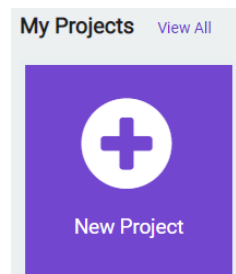
Under the Advanced blocks you can find the Control blocks. These are the blocks that are the most complicated to use, but are the most powerful. The On Event block accepts an event type to watch for and a pin that event should happen on. When that specific event is emitted from that pin, it will trigger whatever code is inside of it. Take a moment to look through the list of different events that you can listen for!

if / Else if / Else

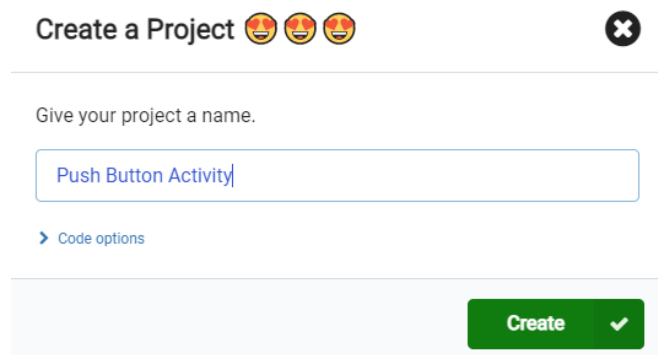
Finally, inside of the forever block is a more complex if block, which is an if / if else / else tree. To build this more complex "if" statement, add a standard if / else block into your program. Then click on this small gear in the upper left-hand corner of the block. This will open a tiny interface with more blocks in it. You can drag more else if blocks into the structure here to build your decision tree.

Building the "Push Button Activity"

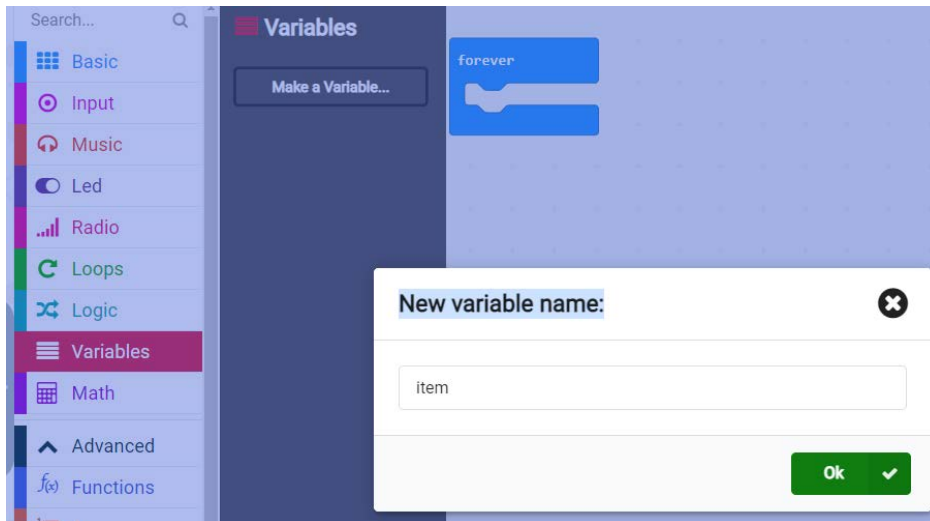
Step 1: Go to <https://makecode.microbit.org/#> and create a New Project



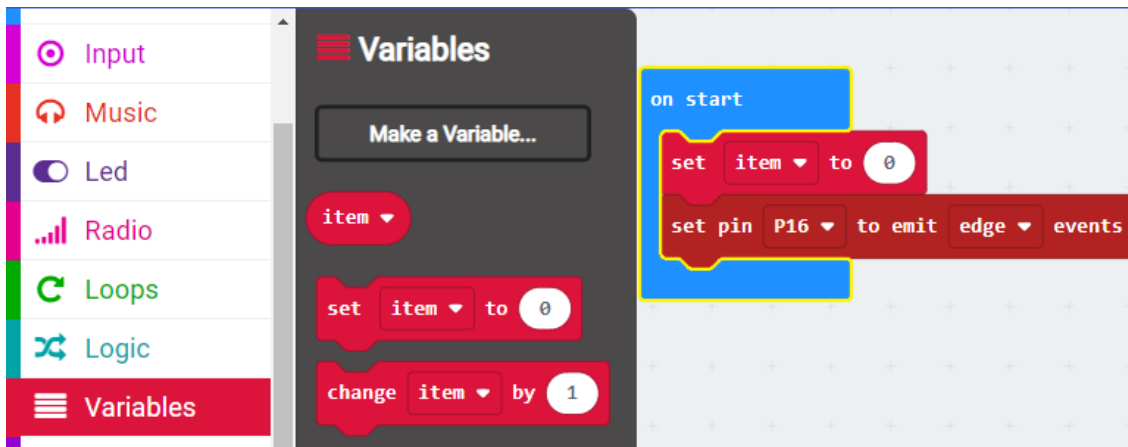
Step 2: Click on New Project and give it a project name **Push Button Activity** and click Create



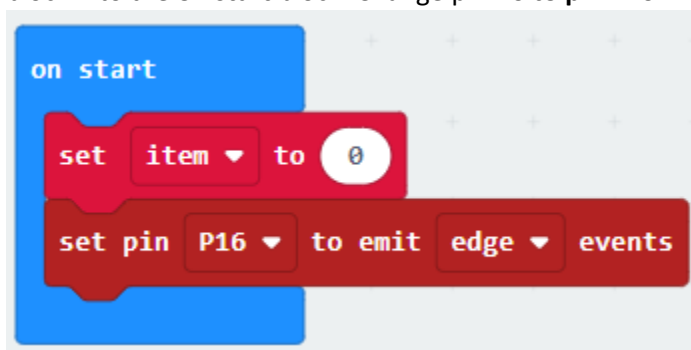
Step 3: Once the MakeCode is launched, click on the **Variables** category and then click on **Make a Variable**. In the New Variable name window type **item** then click **Ok**



Step 4: Once the **item** variable has been created select and drag the **set item to 0** block into the **on start** block



Step 5: Click on the **Advance** | **...more** category and select and drag the **set pin P0 to emit edge events** block into the **on start** block. Change pin **P0** to **pin P16**



Step 6: Click on the **Logic** | **Conditionals** category then select and drag the **if <true> then** block into the **forever** block

Step 6: Click on the **Logic | Comparison** category then select and drag the **<0> = <0>** block into the **if <true> then** block

Step 7: Click on the **Variables** category then select and drag the **item** block into the **if <true> then** block

Step 8: Click on the **Advance** category then **Pins** then select and drag the **analog write pin P0 to 1023** block into the **if <true> then** block.

Step 9: Right click on the **analog write pin P0 to 1023** and select duplicate. Drag the copy block it to the **if <true> then** block. Change ping P0 to pin P1 and **1023** to 0

Step 10: Right click on the **analog write pin P1 to 0** and select duplicate. Drag it to the **if <true> then** block. Change ping P1 to pin P2



Step 11: Click on the plus (+) to create an additional **else if** statement block

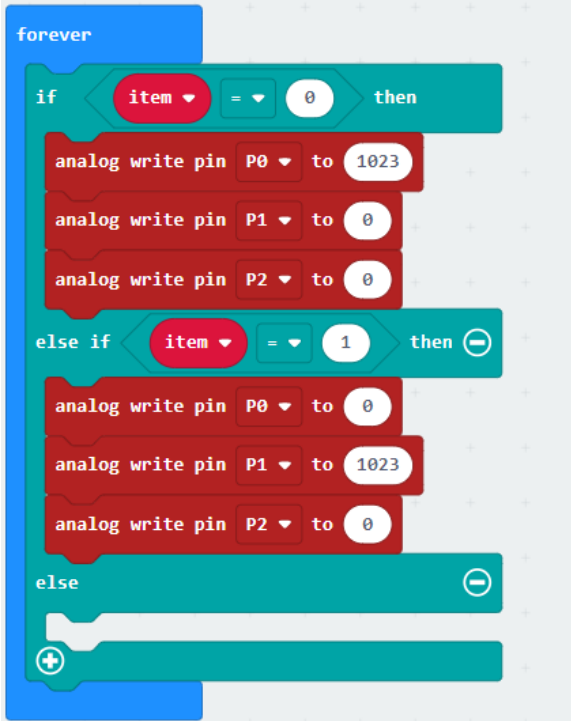
Step 12: Click on the **Logic | Comparison** category then select and drag the **<0> = <0>** block into the **else if <true> then** block. Change the value of 0 to 1

Step 13: Click on the **Variables** category then select and drag the **item** block into the **else if <true> then** block

Step 14: Click on the **Advance** category then **Pins** then select and drag the **analog write pin P0 to 1023** block into the **else if <true> then** block. Change 1023 to 0

Step 15: Right click on the **analog write pin P0 to 0** and select duplicate. Drag it to the **else if <true> then** block. Change ping P0 to pin P1 and **0** to 1023

Step 16: Right click on the **analog write pin P1 to 1023** and select duplicate. Drag it to the **else if <true> then** block. Change ping P1 to pin P2 and 1023 to 0



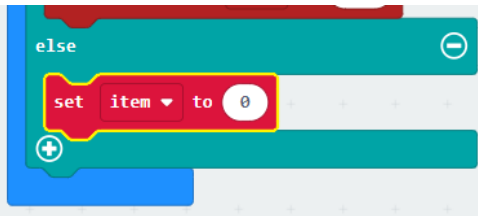
```
forever
  if item = 0 then
    analog write pin P0 to 1023
    analog write pin P1 to 0
    analog write pin P2 to 0
  else if item = 1 then
    analog write pin P0 to 0
    analog write pin P1 to 1023
    analog write pin P2 to 0
  else
```

Step 17: Repeat steps 11 – 16, but with the following changes as the figure below

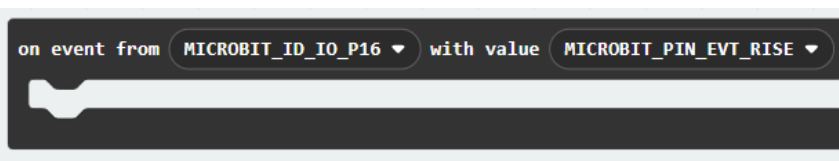


```
forever
  if item = 0 then
    analog write pin P0 to 1023
    analog write pin P1 to 0
    analog write pin P2 to 0
  else if item = 1 then
    analog write pin P0 to 0
    analog write pin P1 to 1023
    analog write pin P2 to 0
  else if item = 2 then
    analog write pin P0 to 0
    analog write pin P1 to 0
    analog write pin P2 to 1023
  else
```

Step 18: Click on the **Variables** category then select and drag the **set item to 0** block into the **else block**



Step 19: Click on the **Control** category then select and drag the **on event from** <MICROBIT_ID_BUTTON_A> with value <MICROBIT_EVT_ANY> block into the **program space**. Then change <MICROBIT_ID_BUTTON_A> to <MICROBIT_ID_IO_P16> and <MICROBIT_EVT_ANY> to <MICROBIT_PIN_EVT_RISE>

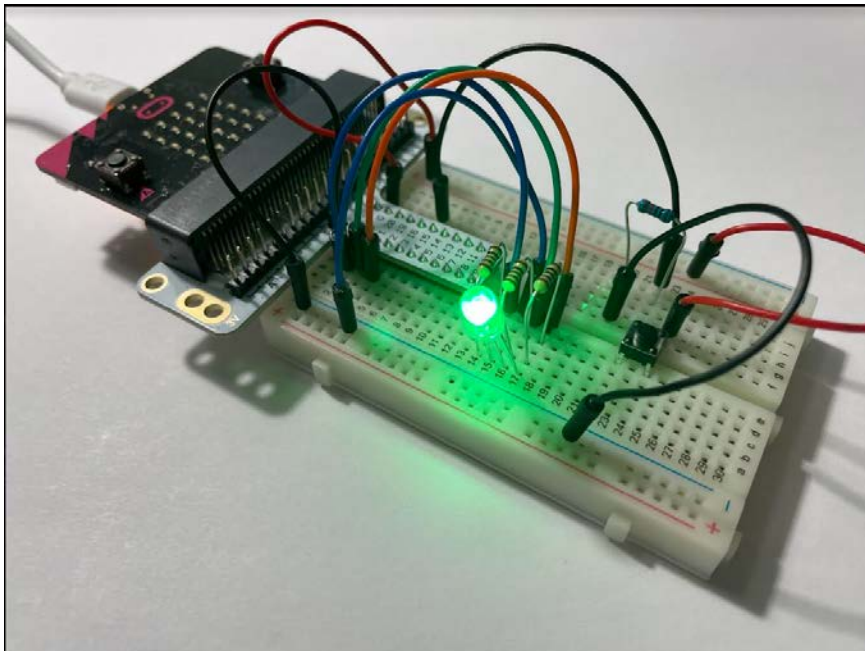


Step 20: Click on the **Variables** category then select and drag the **change item to 1** block into the **on event from** block

Step 21: Click on the **Basic** category then select and drag the **pause (ms)** into the **on event from** block



Final Prototype

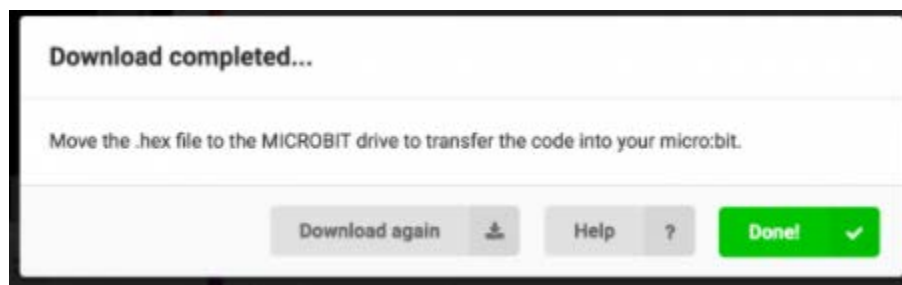


Plug the USB cable to the micro:bit

Step 22: Click the download button in the lower lefthand corner of the code window. It will downloaded most probably in the Downloads folder



Step 23: Simply click and drag your program file from its download location to your micro:bit drive, which shows up as an external device.



Step 24: When you press the button, the RGB will turn on to a color. When you press it again, the color will change and another press will change the color once again. Press it one more time, and it will turn off. Every time you press the button, it increments a variable, and then we check against it to set the color. If the variable goes over the value of 2, we reset it to 0, which is off.

Step 25: Make changes to your program by modifying the values in the analog write pins.

Congratulations! You have successfully completed this activity.

Reference: Sparkfun Inventor's Kit for micro:bit Experiment Guide

<https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-for-microbit-experiment-guide>