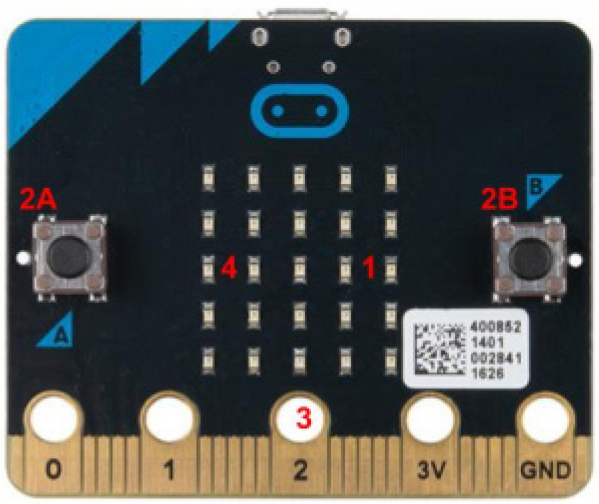
Hardware Overview

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 you were looking at just exploring the basics of programming and hardware.

Front

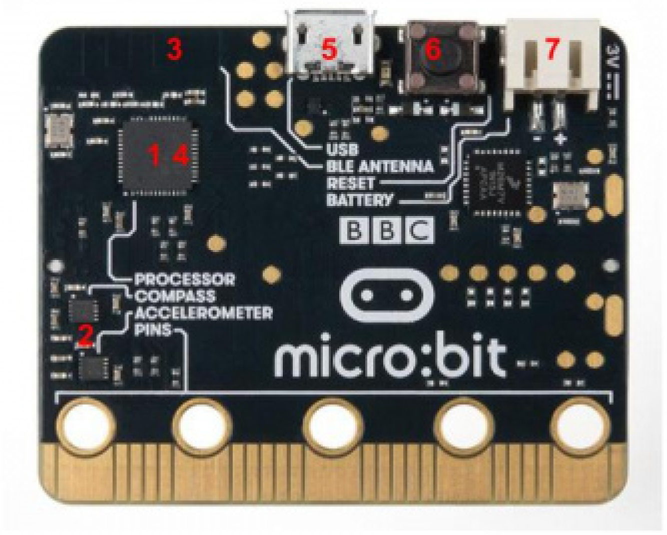
On the front of the board there are a number of components that are pretty visible right off the bat!



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

The back is where a lot of the magic happens. Check it out…

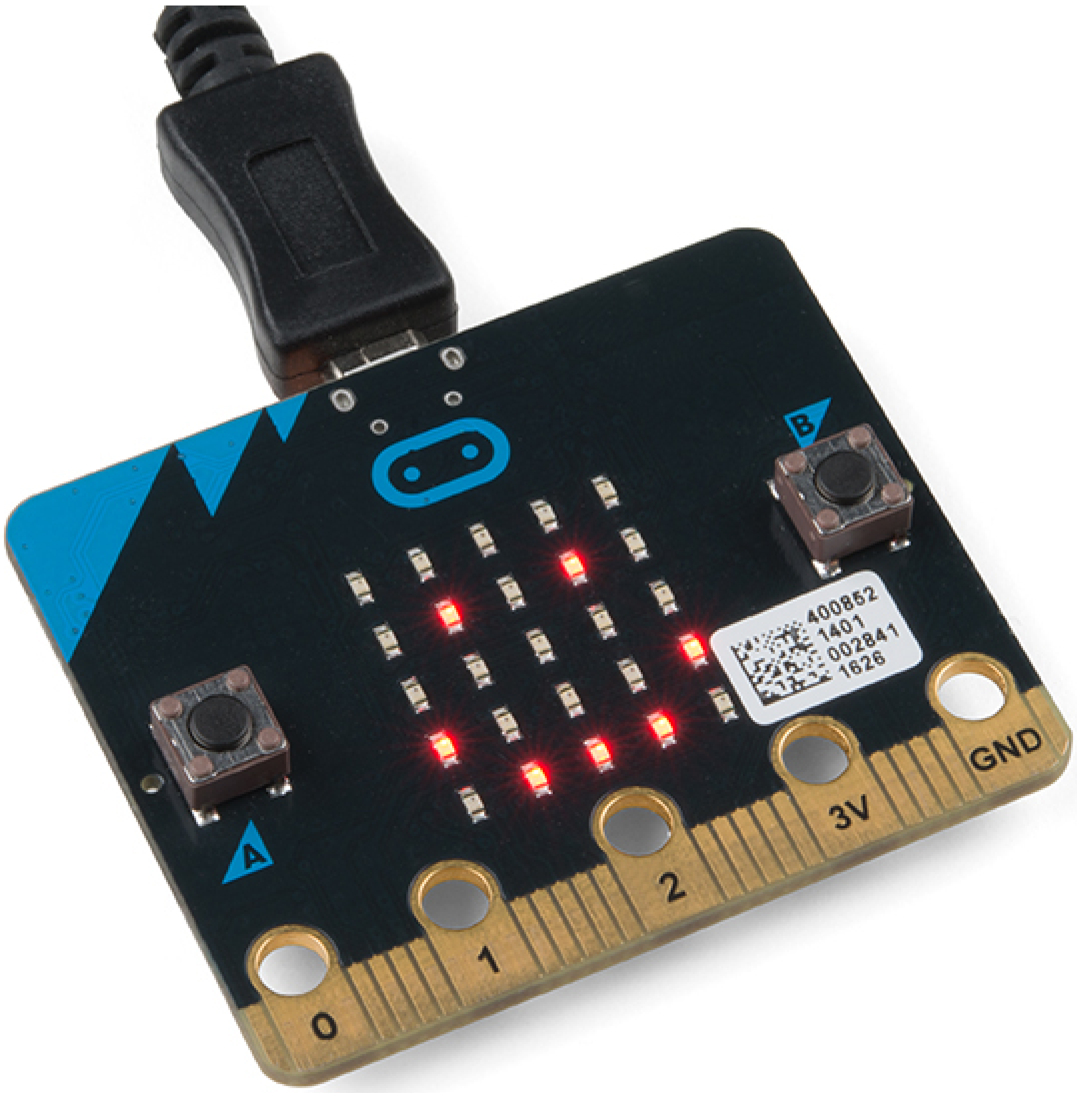


1. **Microcontroller** — The brains of the outfit. The micro:bit is powered by a 16MHz ARM Cortex-M0 microcontroller with 256KB Flash and 16KB RAM.
2. **Accelerometer/Compass** — The micro:bit has an onboard accelerometer that measures gravitational force, as well as a compass that can detect its orientation using Earth's magnetic field.
3. **Bluetooth/Radio** — Communication is huge with the micro:bit. You can communicate with your phone or tablet using Bluetooth Low Energy (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.

Phew! That is a lot of bells and whistles…a true Swiss army knife!

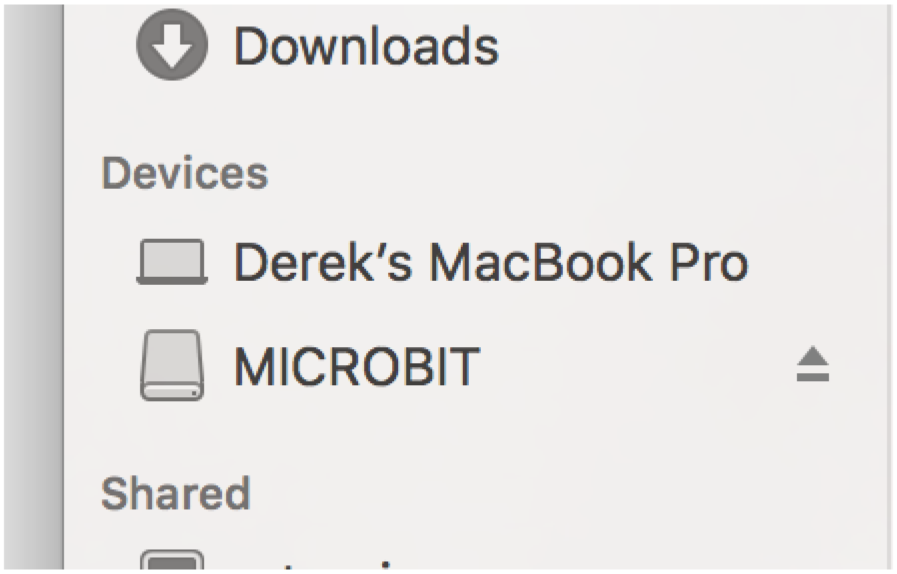
Hooking It Up

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!

Let's get programming!

Using MakeCode

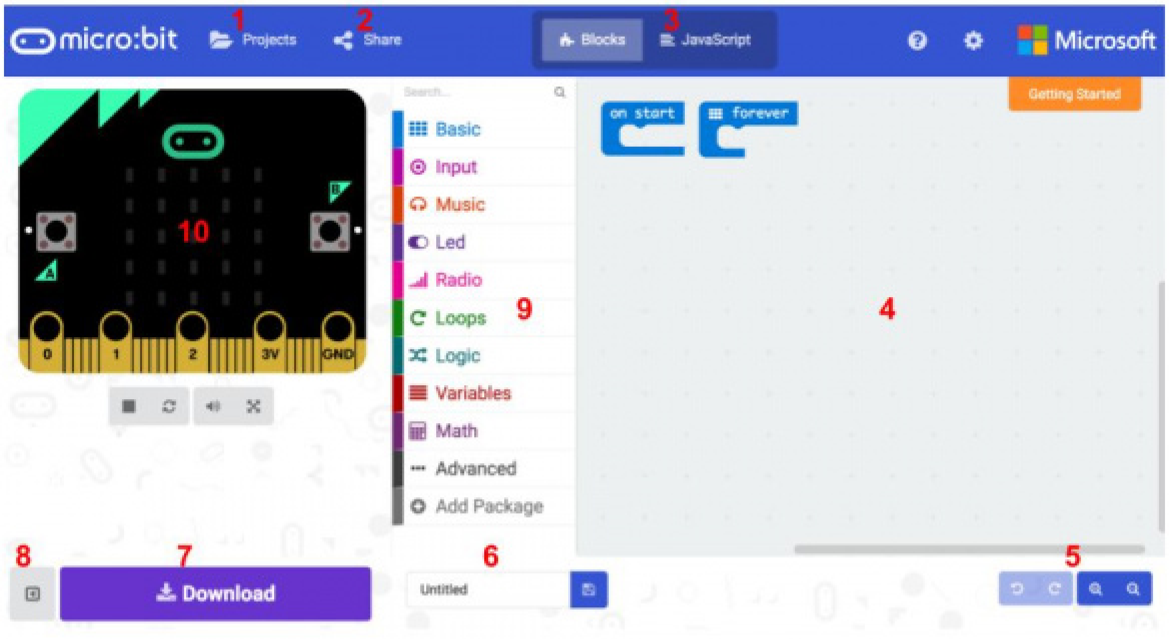
This guide and most of SparkFun's content around the micro:bit will use MakeCode by Microsoft for programming.

What Is MakeCode?

MakeCode is an open programming environment built by Microsoft for the micro:bit, as well as other boards. You can navigate to MakeCode for the micro:bit by following this link:

LAUNCH MAKECODE!

Once you have launched MakeCode, you will be greeted by its basic layout with a simulator on the left and a block-based environment on the right when your browser's window is maximized, as shown here.



Click on the image above for a larger view.

Let's take a quick tour and check out what is available to us!

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!

Phew! Now you have a choice - blocks or text-based programming?

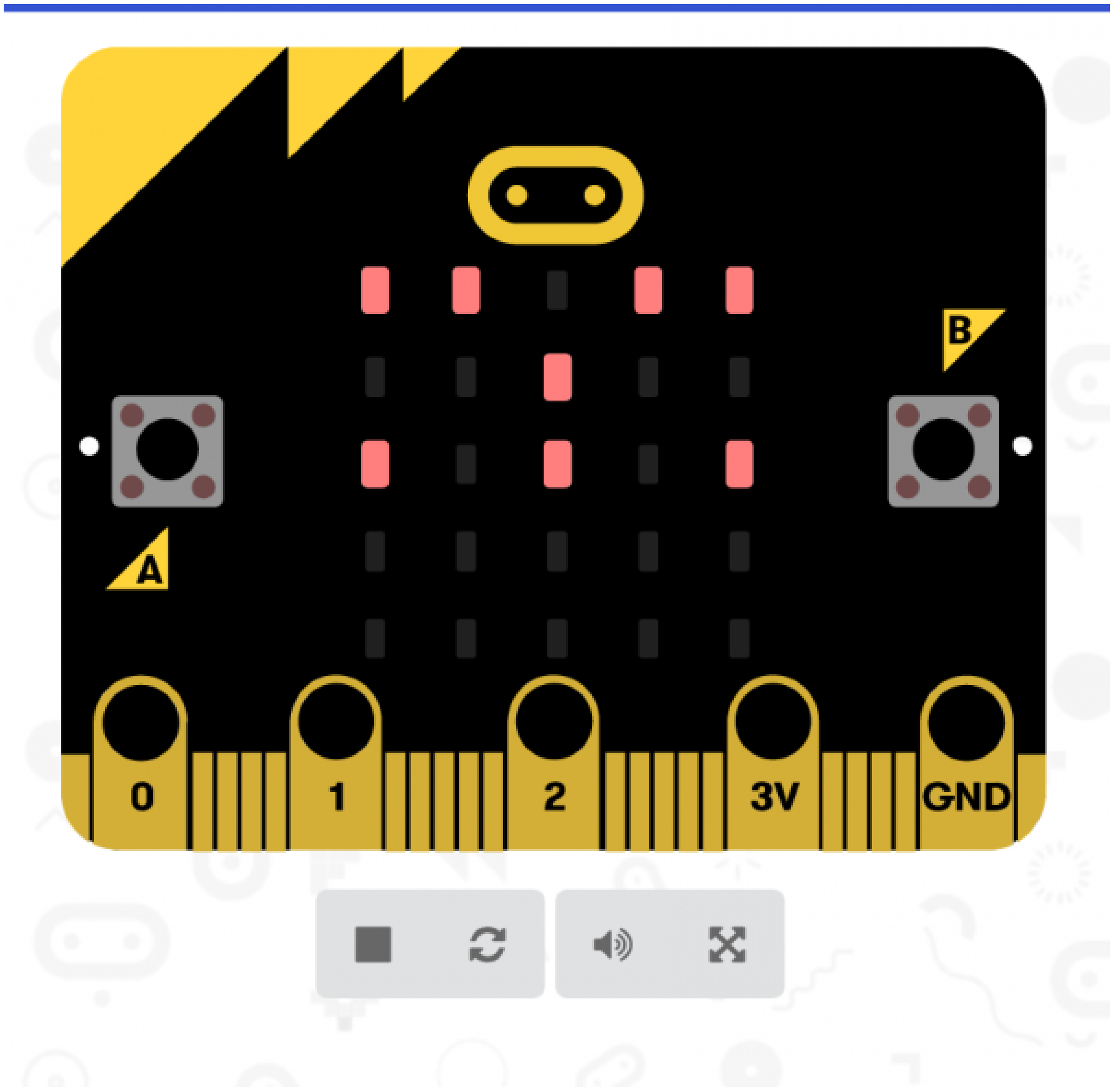
Blocks or Text

For this guide and the majority of the content that you will find on SparkFun for the micro:bit, we will be using block-based programming examples.

But, if you so choose there is a JavaScript option to use as well. The choice is yours, and the good news is that you can switch back and forth from one to the other in the same program; one will populate the other, which is really nice if you are new to programming!

Simulator

MakeCode includes a simulator for the micro:bit, meaning if you don't 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!



The simulator will update as you build your code, and if you want to run it from the beginning you can click the stop and run buttons to start it over again!

Speaking of code, let's write a quick program and get it onto your micro:bit!

Hello, World!

Now comes the good stuff — writing your first program for your micro:bit in the MakeCode programming environment!

"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. It also gives you a chance to make sure everything is up and running and A-OK.

For your first "Hello World" we are going to create a simple animation on the LED array that repeats forever. If you just want the complete program, you can see it here. To see a step-by-step explanation of how we built the program, continue reading!

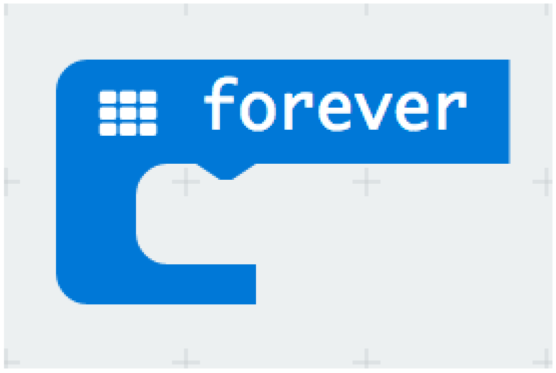
Note: You may need to disable your ad/pop blocker to interact with the MakeCode programming environment and simulated circuit!

Building 'Hello World'

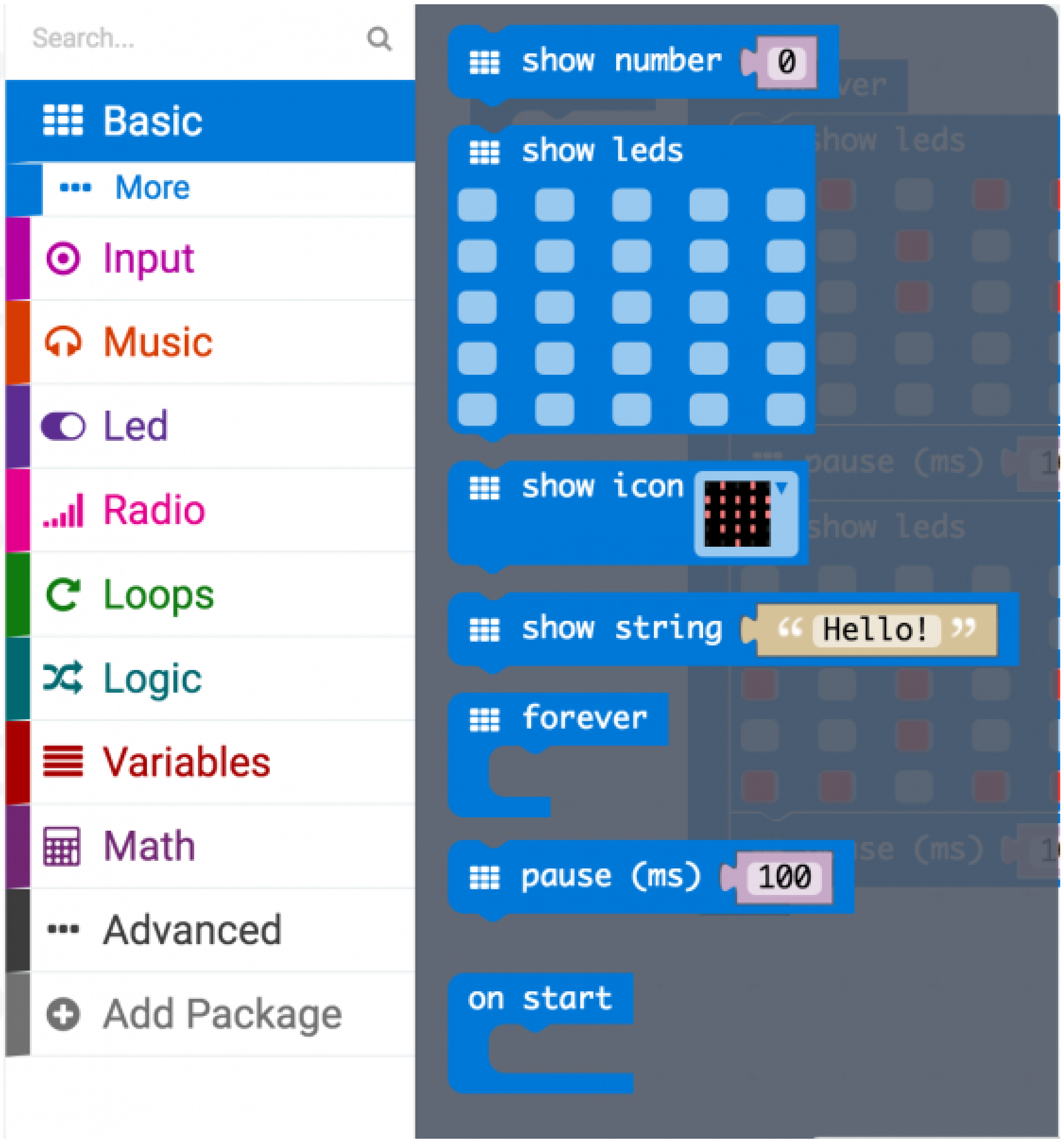
A "Hello World" on the micro:bit is a little different than on a normal run-of-the-mill microcontroller. The micro:bit has no single LED to blink on its own, as you would find on the Arduino or similar boards. What the micro:bit does have is an LED array! So, the "Hello World" for the micro:bit is to draw something using the LED array!

When you open MakeCode you are greeted with two blocks: the `On Start` block and the `forever` block. The `On Start` block is all of your code that will execute at the very beginning of your program and only run once. The `forever` block is code that will loop over and over again…forever.

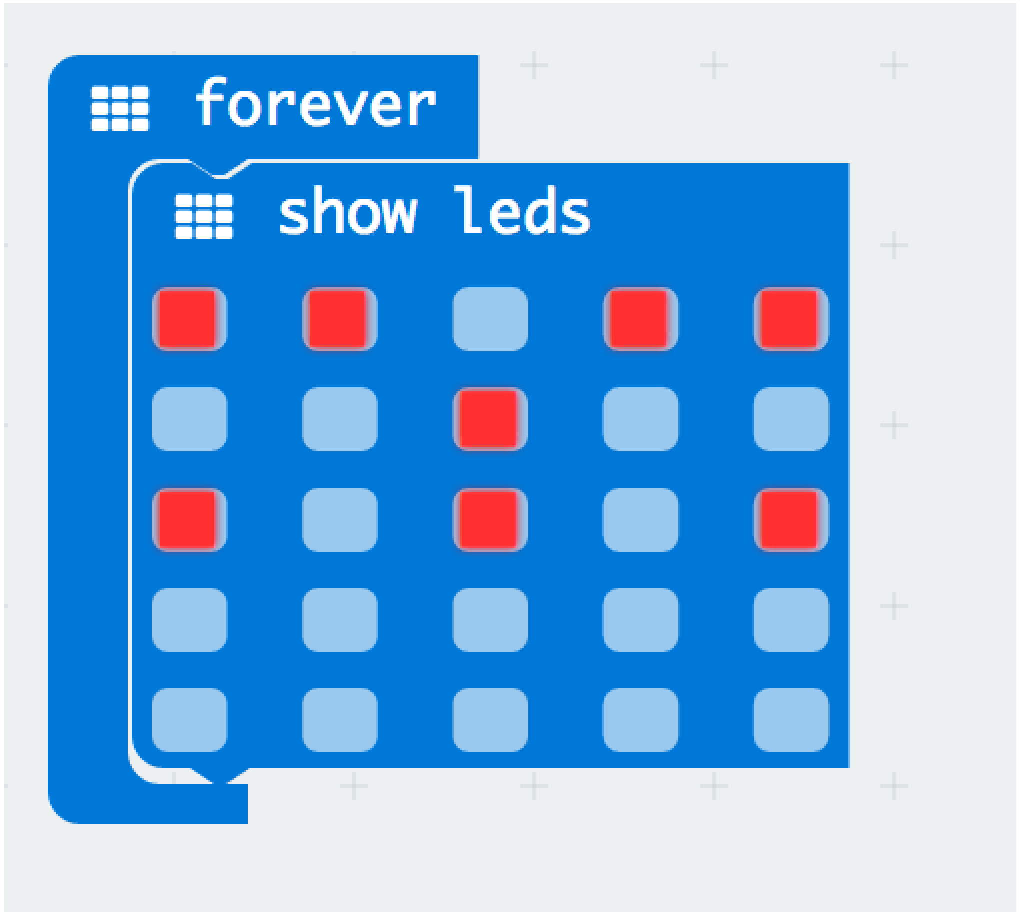
We are going to use the `forever` block for building this "Hello World." We now need to start adding blocks to `forever`.



First, click on the Basics 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.

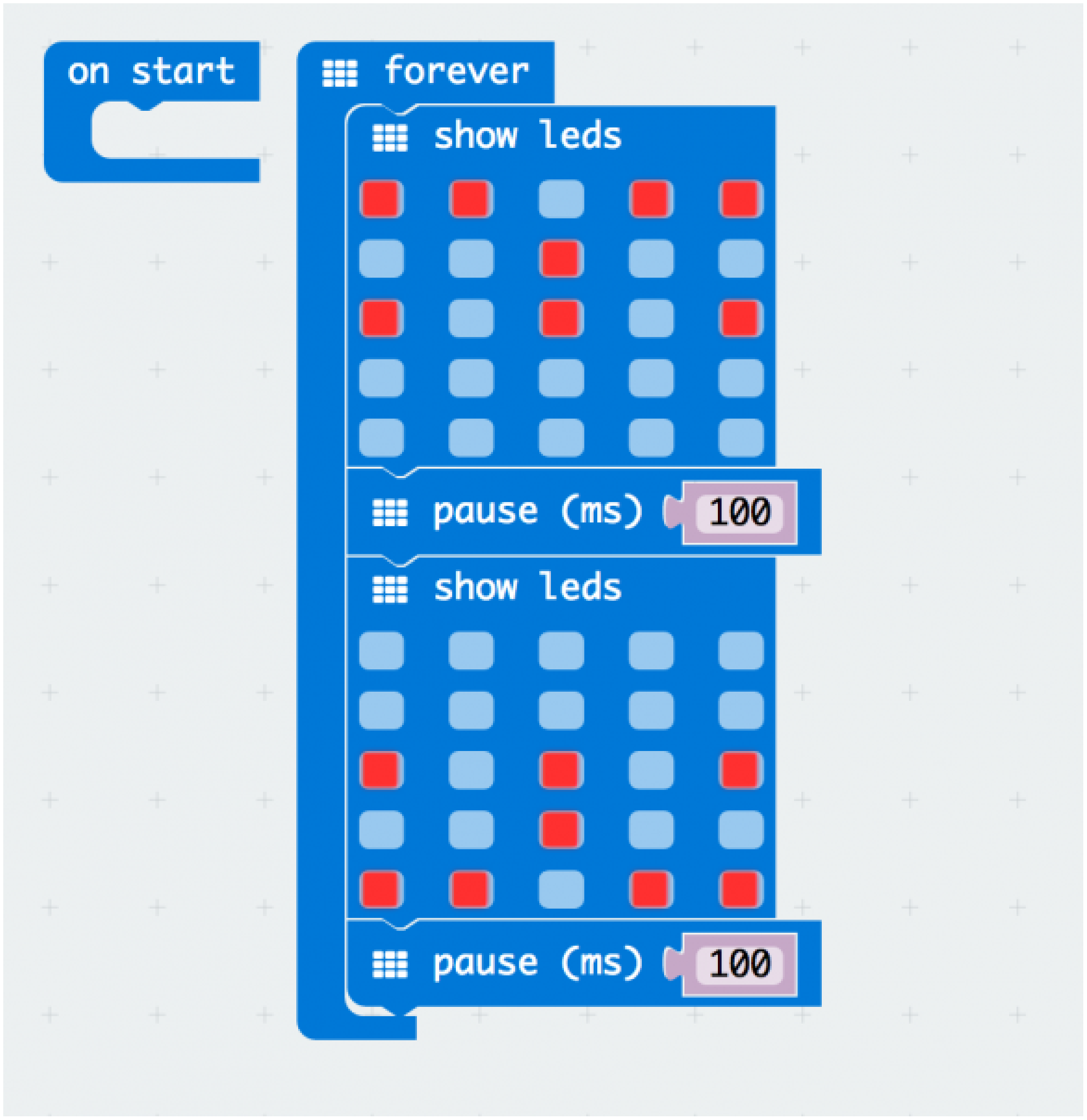


The `show leds` block has an array of squares that symbolize the LED array. If you click on a square, it will turn red, which means that it is 'on'. Draw a simple pixel art shape by turning different LEDs on or off; you should be able to see the outcome in your simulator on the lefthand side of your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



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.

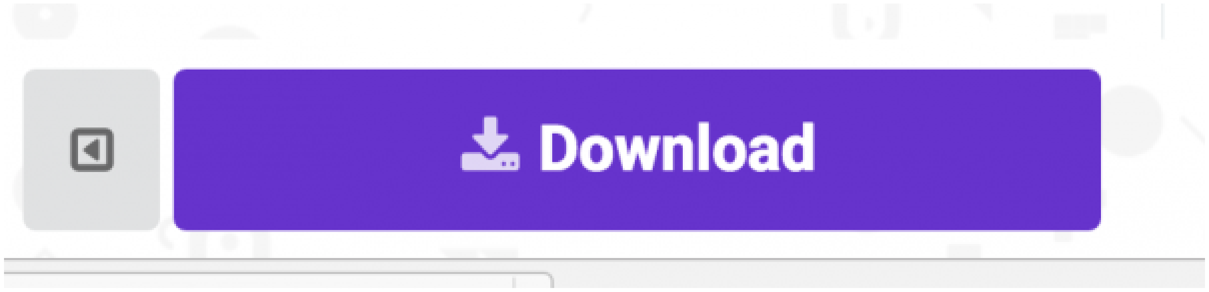
We have built up an example in the next section where you can download the file and try it out on your own micro:bit, or use the simulator. If you want to play around with the code and make some changes, go ahead and click the Edit button in the widget, and it will open a MakeCode editor for you to start hacking “Hello World.” Enjoy!

Getting Your Program Onto Your micro:bit

You’ve built your first program in MakeCode, and it works in your simulator. Now, how do you get it onto your micro:bit?

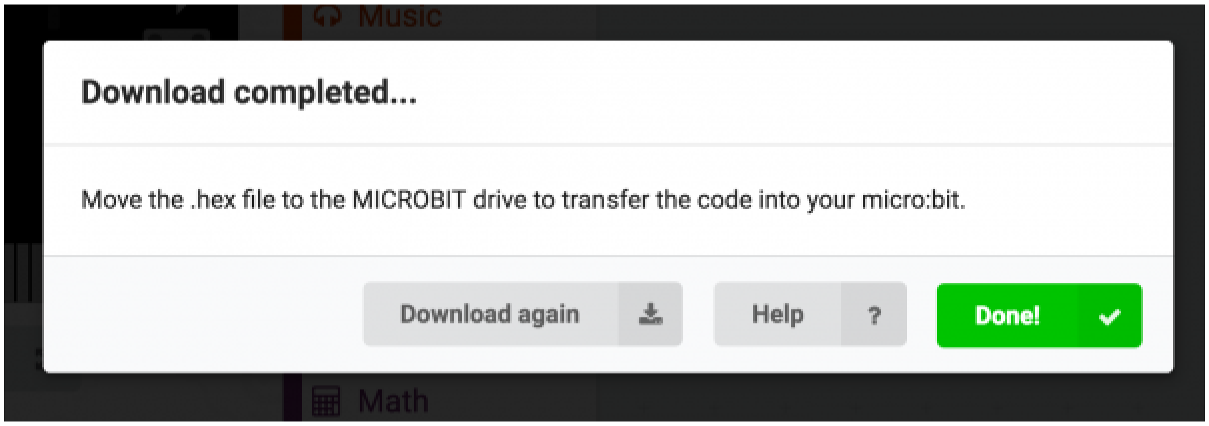
Download Your Program

Once you are happy with your program, you can click the Download button in MakeCode.



This will download your program file to your standard download location, probably the Downloads folder on your computer, or whatever location you have set in your download preference.

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


Download completed...

Move the .hex file to the MICROBIT drive to transfer the code into your micro:bit.

That's it!

Your micro:bit will flash for a few seconds, and then your program will start automatically. Yes! Win!

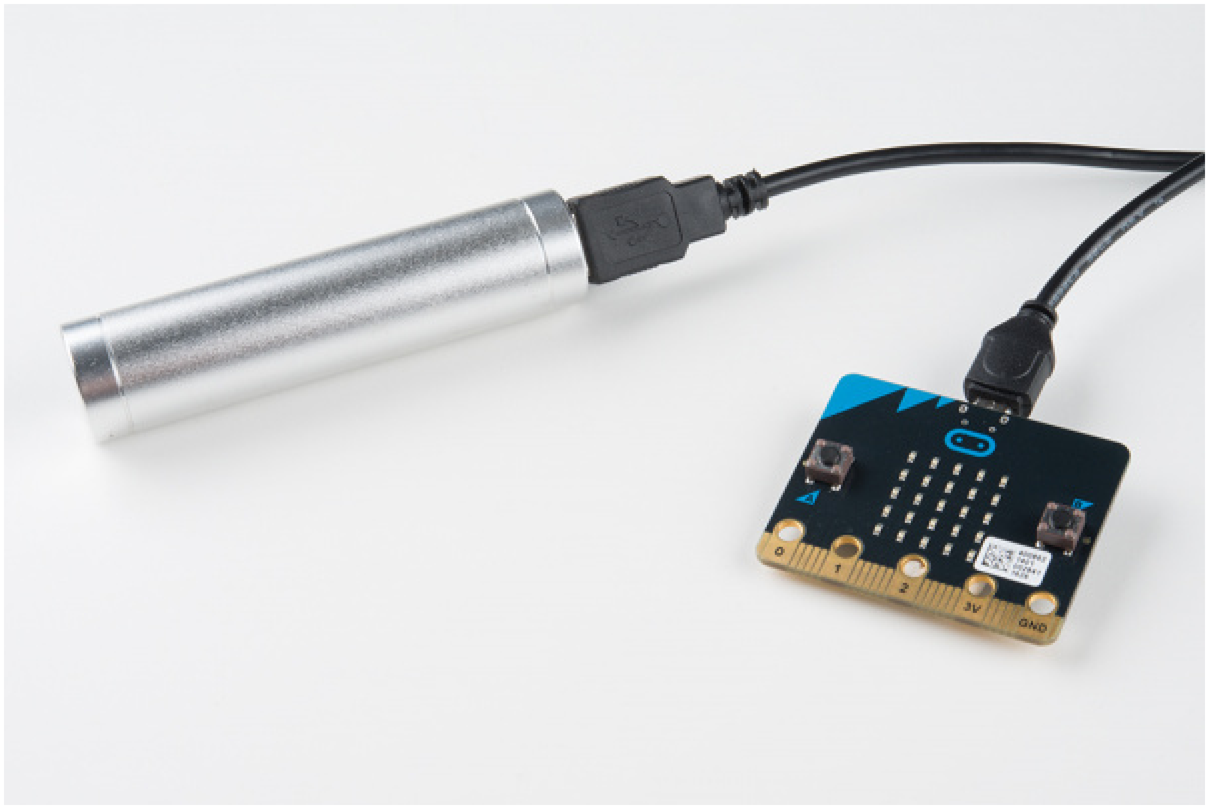
Note: You may need to disable your ad/pop blocker to see the simulated circuit!

Powering the micro:bit

You have your program running on your micro:bit, but it is still tethered to your computer. There are a number of ways to solve this problem…batteries, batteries and more batteries!

USB Battery Pack

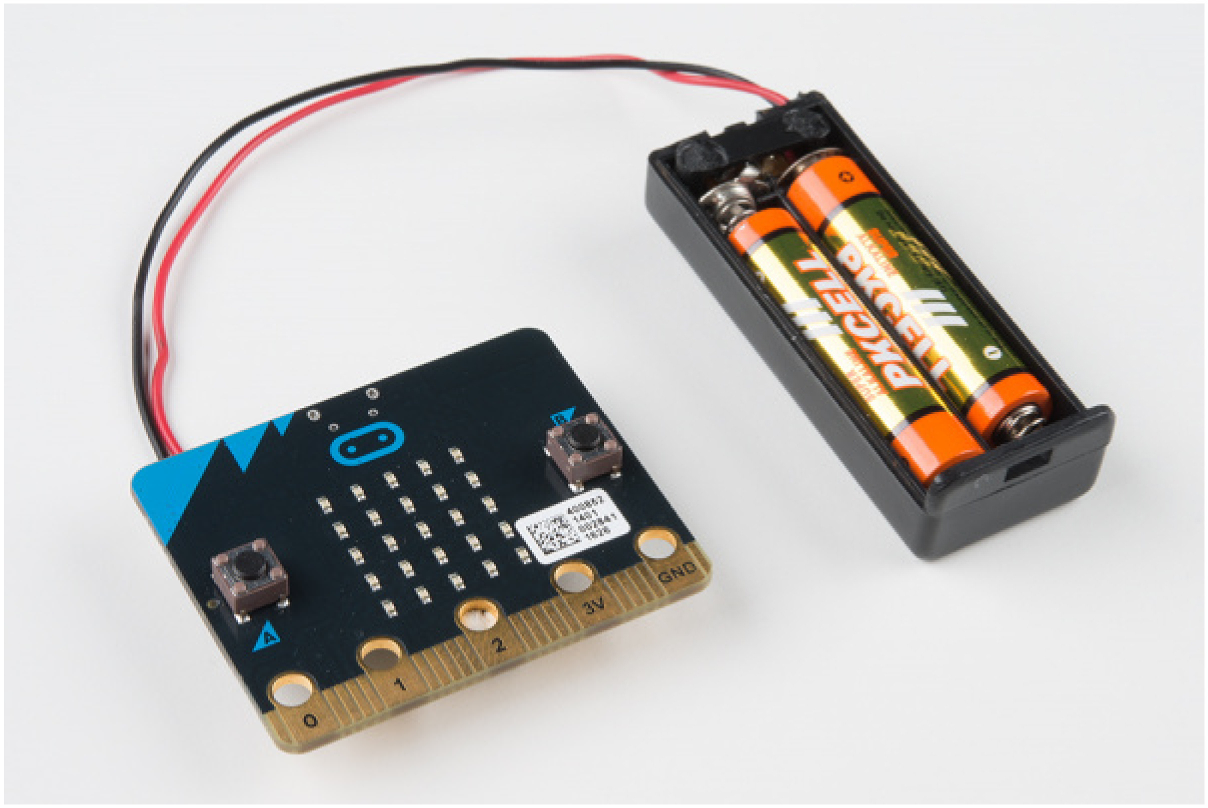
USB battery packs are becoming pretty commonplace out there. You can use one to run your micro:bit project for quite a long time.



It is handy to have a shorter USB cable to keep from dragging a super-long wire around.

2xAA Battery Pack

The 2xAA battery holder with JST-PH connector is a great solution if you are looking to power a whole lot of micro:bits for a good amount of time, such as in a classroom setting.



These batteries can be purchased in bulk for pretty cheap.