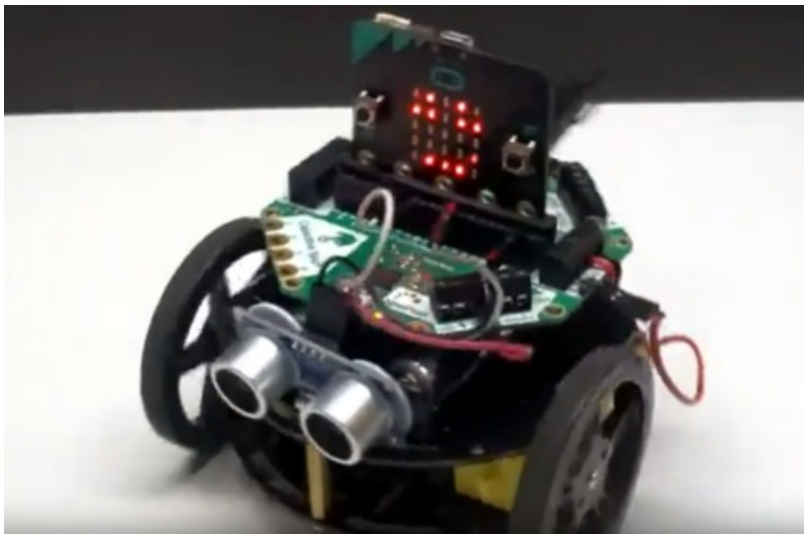
adafruit learning system

micro:bit Crickit Robot

Created by Richard Albritton



Last updated on 2020-02-16 10:38:16 PM UTC

Overview

This robot is easy to put together and you can program it using the MakeCode block based coding platform so that even the little ones can make robot friends.

For this build we will be connecting two motors so that the robot can move, a sonar sensor so that the robot can see, and a speaker so that your robot can make noises. The Crickit platform leaves many more things that can be added like other sensors, servos, NeoPixels, and more.

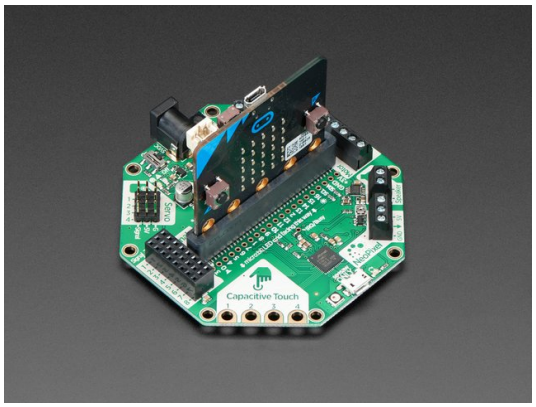
Parts

Your browser does not support the video tag.

Mini Round Robot Chassis Kit - 2WD with DC Motors

$19.95
IN STOCK

Add To Cart



Adafruit CRICKIT for micro:bit

$29.95
IN STOCK

Add To Cart

Your browser does not support the video tag.

BBC micro:bit Go Bundle

$17.50
IN STOCK

Add To Cart

Your browser does not support the video tag.

HC-SR04 Ultrasonic Sonar Distance Sensor + 2 x 10K resistors

$3.95
IN STOCK

Add To Cart

---

1 x Premium Male/Male Jumper Wires
20 x 3" (75mm)

Add To Cart

1 x Thin Plastic Speaker with wires
8 ohm 0.25W

Add To Cart

1 x Alkaline AA batteries
LR6 - 4 pack

Add To Cart

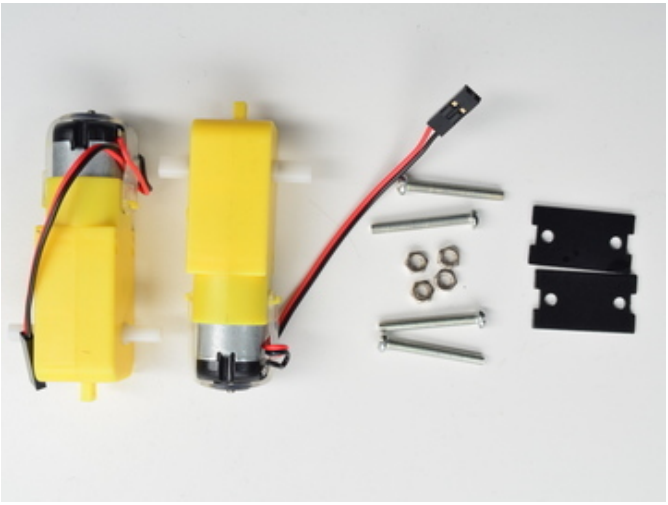
1 x 4 x AA Battery Holder with 2.1mm Plug and On/Off Switch

Add To Cart

1 x UBEC DC/DC Step-Down (Buck) Converter
5V @ 3A output

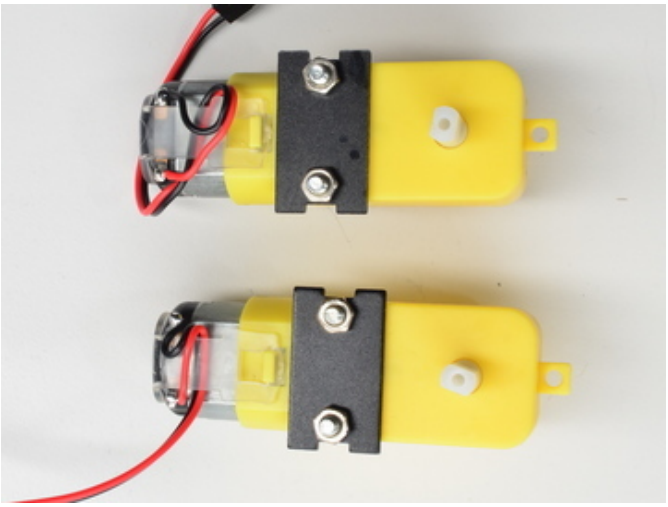
Add To Cart

Assemble the Wheels



To start, take the two motors, four long screws, four nuts, and two black panels.

Screw the two black panels onto the motors.



Screw the two black panels onto the motors. The metal panels go on the side with the red and black wires coming out

Have the hex nuts on the metal panel side so they dont interfere with the wheel!

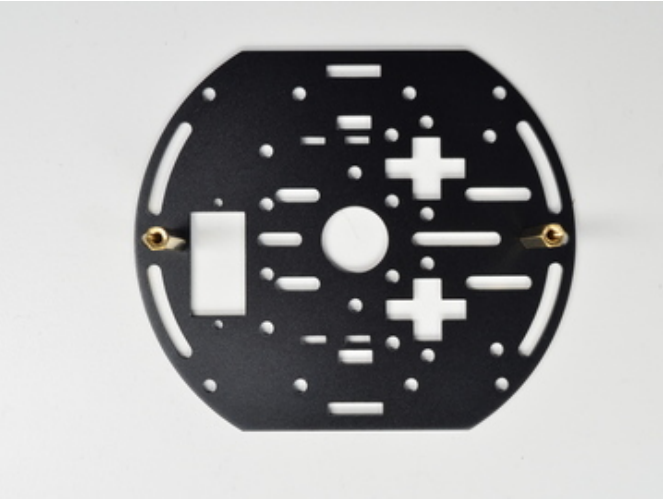
Take the two wheels, rubber treads, and 2x small screws found in the same bag as the wheels.

Put the rubber treads on the wheels. This is a lot of fun!

Fit the wheels onto the white knob on the motors, they will snap nicely onto the oval center.

Attach the wheels into place with the tiny screws

Assemble the Chassis

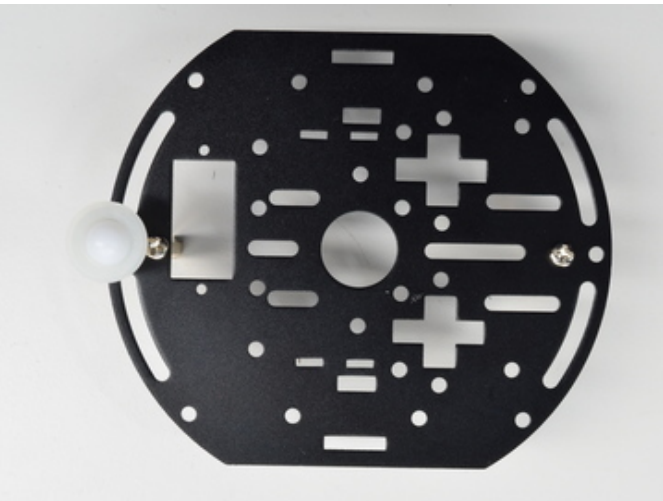


Take one of the black chassis layers. All three layers are identical.

Align it on your table as shown on the left. Note that the panel **is not symmetrical** - look on the left to see that rectangle cut out? make sure its aligned as you see here!

Attach two of the brass standoffs onto the black chassis layer.

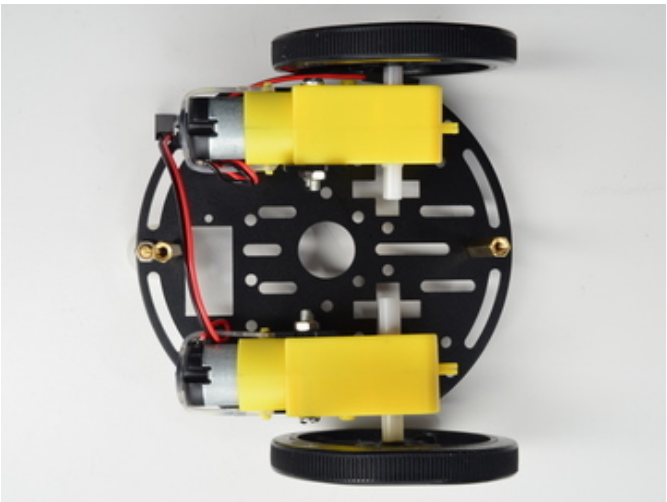
The standoffs should be screwed into the second set of holes from the outer edge - meaning the two interior holes.



Turn over the plate

Attach the white freewheel into the exterior hole closest to the rectangular opening.

The white freewheel should be on the opposite side of the chassis of the standoff.

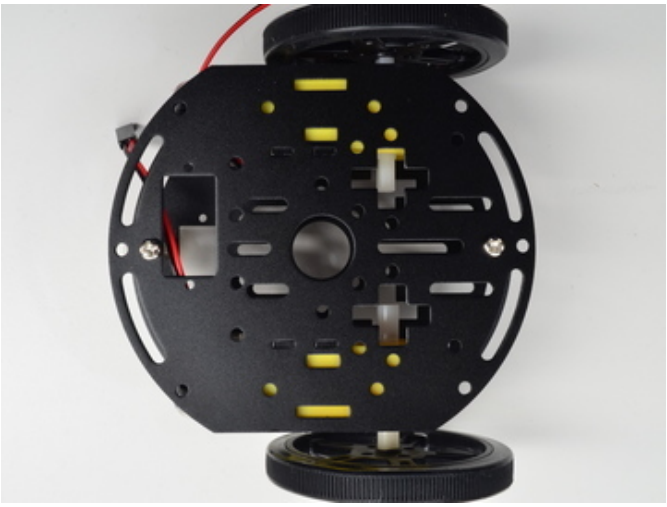


Turn over the plate again

Take your assembled wheels and fit them into the chassis layer.

There are 2 slots on the black panels that you attached to your motor that should fit perfectly into the chassis layer.

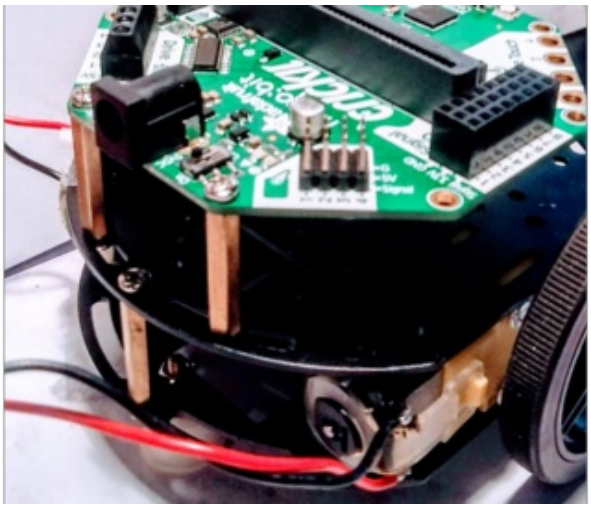
The metal front of the motor will be pointing toward the side of the chassis where you placed your white freewheel



Place the next layer of the chassis on top of the motors. *In this photo we didn't line up the second plate just like the first plate (note the off-set rectangle cut out) - this might fit for you or you might have to line them up with the same cutout to get a good fit.*

The two slots on the black panels you attached to the motors should fit perfectly into the next chassis layer. This sandwiches the motors in place so they can't slip

Screw in the chassis layer by attaching to the brass standoffs



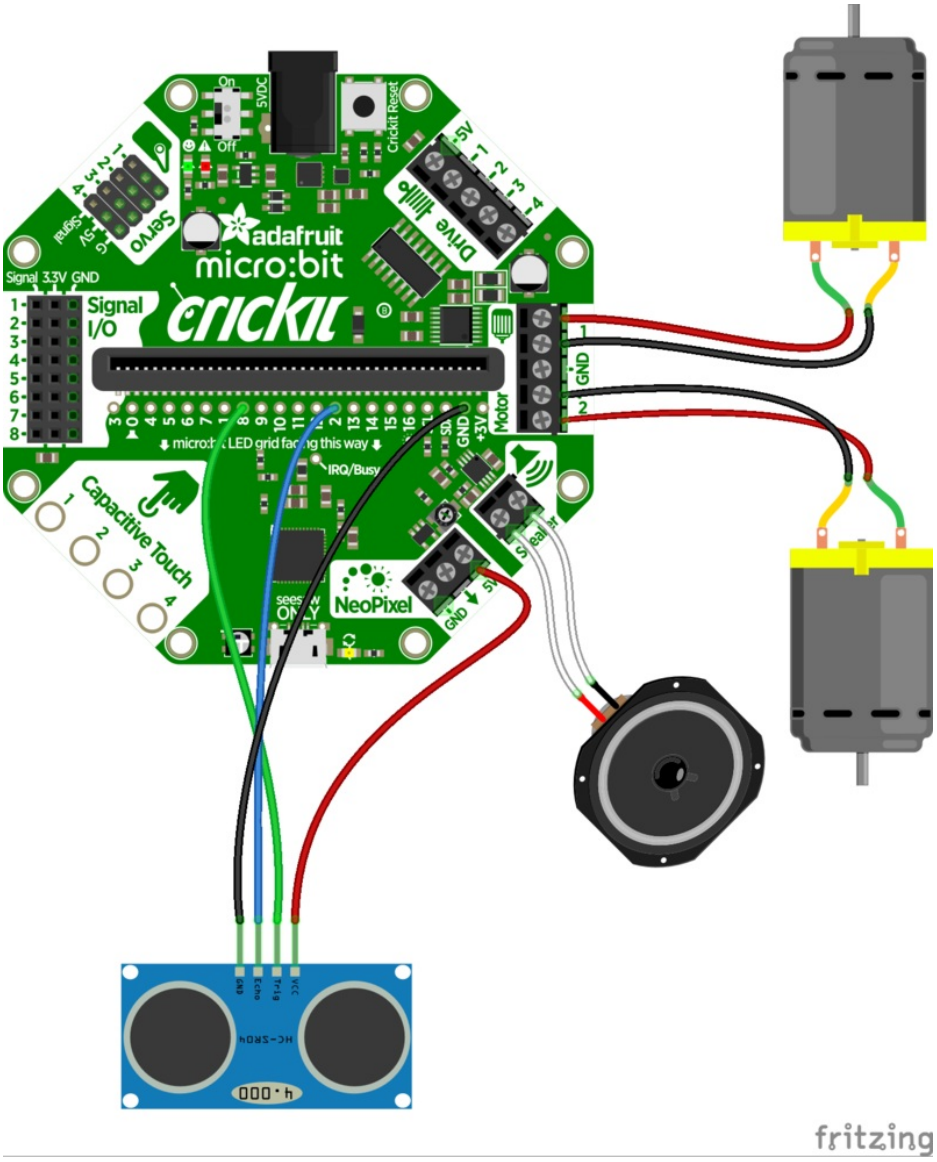
Screw the remaining 2 standoffs into the second chassis layer in the longer curved slots near the back wheel.

You can then use these to secure the Crickit to the robot.

Connect the Wires

The wiring is rather simple but you will have to do a small modification to the battery pack.

Here is what you will be connecting.



Motors

You will want to connect the wires form the motors into the Motor terminals with one set of wires going into each side of 1 and the other set going into 2. If you mix up the wires all that will happen is your robot will drive in the wrong direction. Just switch the wires around if any of the wheels move in the opposite direction for what your code is saying it should do.

You will have to extend the wires from one motor so that they can reach the Motor terminals. You can do this by soldering new wires or just use some jumper wires to extend them.

Sonar Sensor

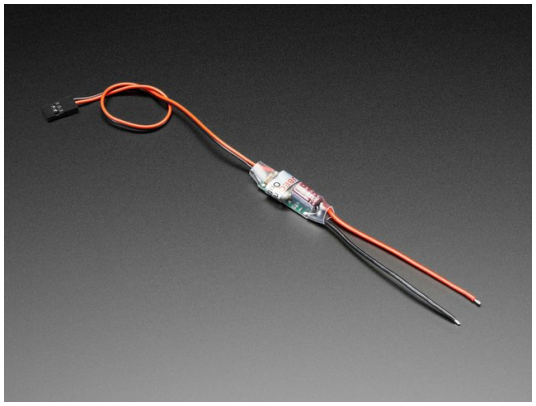
For the Sonar sensor, you want to connect the following pins to the micro:bit breakout pins on the Crickit

- **GND** --> **GND**
- **Echo** --> **Pin2**
- **Trig** --> **Pin8**
- **VCC** --> **5V** (this can go in the terminal for NeoPixel or Drive)

The Battery Box

The Crickit is a bit picky about the amount of power that it gets, and with all the things we are adding, it is hard to maintain that power using the recommended 3 AA batteries. However, 4 AA batteries will trip the Crickit's eFuse and it will not let power in until we get things just right.

The easiest way to fix this is to use a 5V step down voltage regulator that can handle 2 Amps or more.



UBEC DC/DC Step-Down (Buck) Converter - 5V @ 3A output

$9.95
IN STOCK

Add To Cart

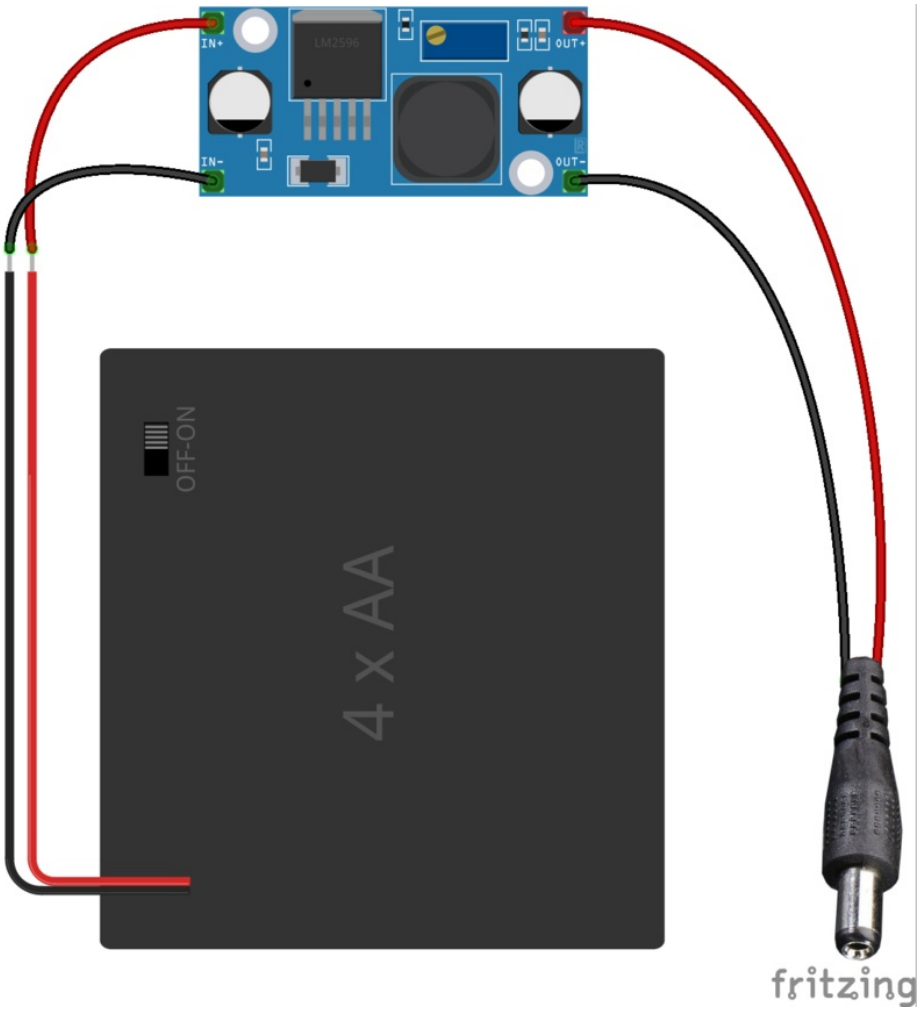
You could also use this one:

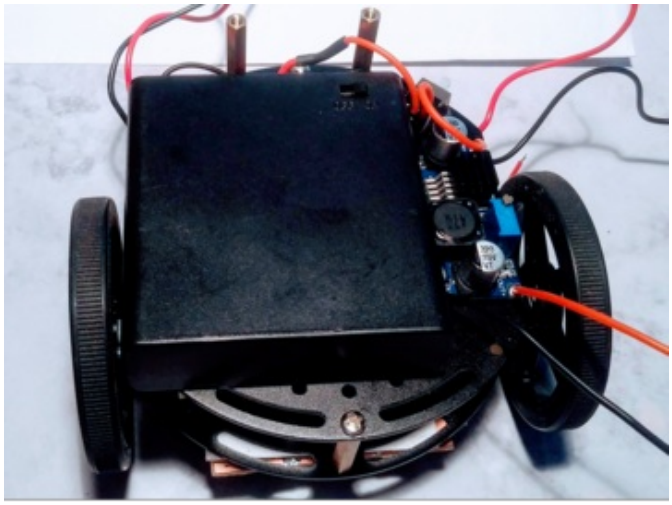
1 x LM2596 DC-DC Buck Converter

Input: DC 3.2V to 46V (The input voltage must be more than 1.5V higher than the output voltage. Can not boost); Output: DC 1.25V to 35V voltage continuously adjustable, maximum output current is 3A;

BUY NOW

You will need to cut the battery wire in the middle and solder the voltage regulator between the battery box and the 5.5mm DC Plug just like in the wiring diagram.





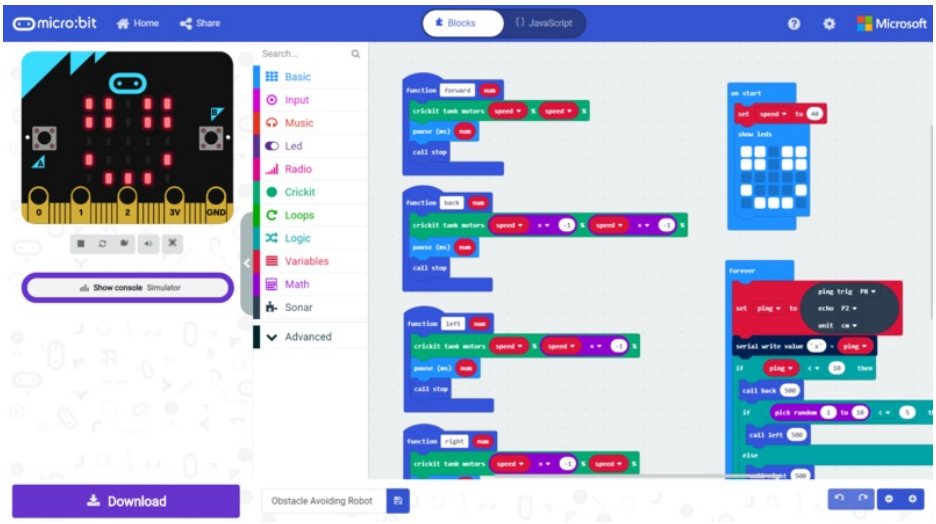
The battery box and 5V voltage regulator should fit on the tom layer of your robot with plenty of room for the Crickit on top.

The Speaker

Now you do not have to use a speaker for this robot, but who doesn't like a robot that drives around all day playing Nyan Cat. The speaker can also be used to alert you when your new robot friend encounters an obstacle using the sonar sensor.

Just connect the wires to the two terminal blocks labeled Speaker.

The Code



Setup

See this page in the Crickit guide (https://adafru.it/BKC) on setting up MakeCode for micro:bit to set up for using Crickit extensions (mid-page).

https://adafru.it/FTp

https://adafru.it/FTp

Code

This robot uses the Crickit and Sonar Extensions for the micro:bit MakeCode blocks. If you use the example code below, those are already added to that file. In the example, I have created some functions that handle the wheel movement code so that you just need to use the **call forward**, **call back**, **call left**, and **call back** blocks under **Advanced** > **Functions**.

Here is some code that will get you started by doing some simple obstacle avoidance:

https://adafru.it/FUc

https://adafru.it/FUc

Programming your Robot Friend

Connect your robot to a computer

The micro:bit slides right into the Adafruit Crickit just like you see in the photo. You will want to use a USB cable to connect the micro:bit to a computer.

Do not use the Micro USB port on the Adafruit Crickit to program your robot with MakeCode.

Once your code is downloaded, your robot will do it's thing.

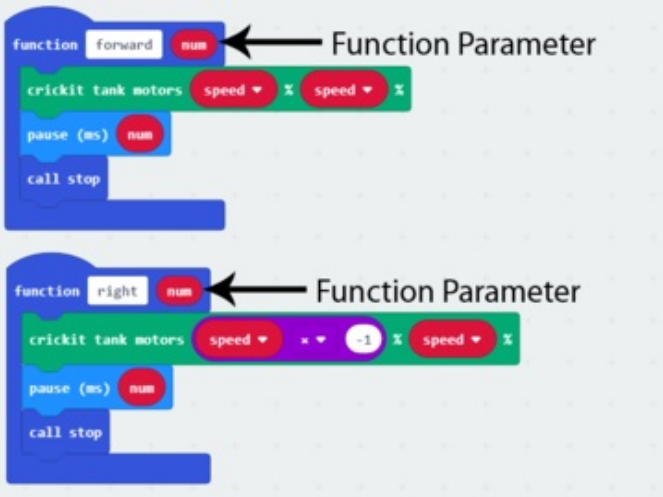
Here is a link to some more info if you get stuck.

https://microbit.org/guide/quick/ (https://adafru.it/FUb)

The USB cable can also be used to display information using the Serial monitor.

Basic Robot Movement

This robot moves using the two motors attached to the wheels. For this, we are using the **Crickit tank motors** command to tell both motors to move at the same time. To make things easier, there are some functions made that will handle the movement.



These functions use a parameter to set the number of milliseconds that you want your robot to do that particular function.

Each movement function will use tank motors to turn the wheels based on the value set by the speed variable.

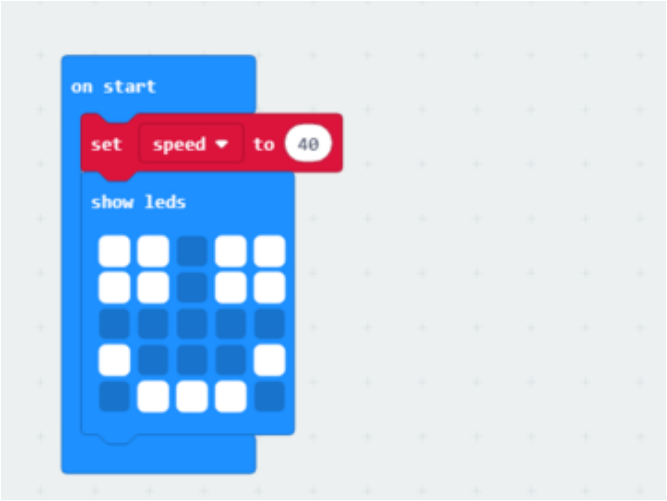
Once the motors are driving, there is a `pause` command that uses the value that was set in the function parameter. This is measured in milliseconds, so remember that 1000 milliseconds is equal to 1 second.

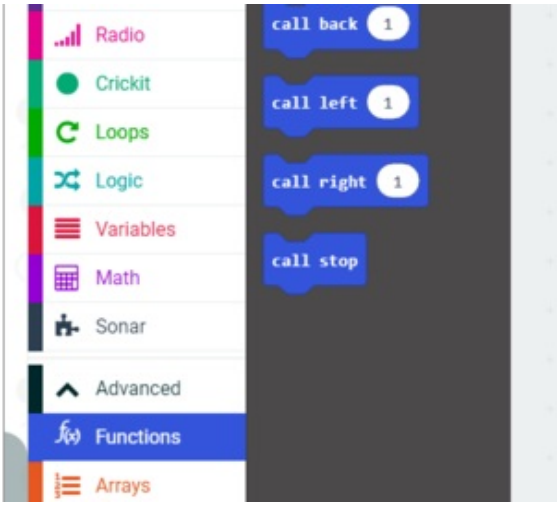
Afre the `pause`, the robot will use the call stop function to stop the robot form moving until the next command.

So how do I make it move?

These functions make it super easy for you to program your robots movements.

First you will want to set a value of **0** to **100** for the **speed** and place that in the **on start** block. I also added a happy face, but that part is optional.

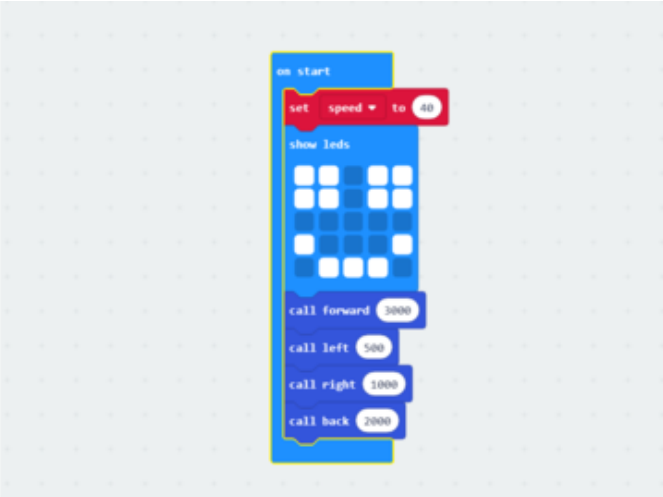


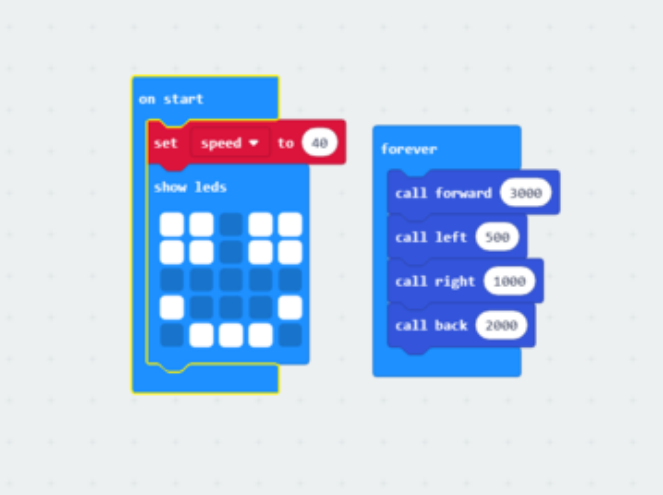


First you will want to set a value of **0** to **100** for the **speed** and place that in the **on start** block. I also added a happy face, but that part is optional.

Next you want to click **Advanced** in the blocks list and click **Functions**.

From here you can just drag out any of the functions that you want to use.



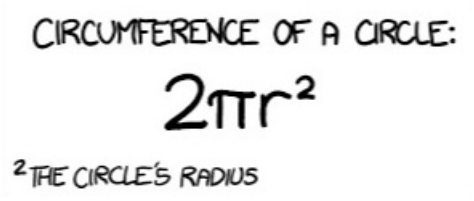


For my code, the robot will drive forward for 3 seconds, turn left for a half of a second, turn right for one second, then drive backwards for 2 seconds before stopping.

This will only happen one time when the robot is powered on or the reset button is pressed. If you want this to happen over and over again, move your function calls to the `forever` block.

But I want my robot to drive in inches

Your robot does not know what an inch is unless you teach it. This is a good chance for you to work with your robot friend and learn something new. If you have programmed a robot before and it let you tell it to drive for a certain distance, that is because a person showed the robot how to calculate that distance based on the diameter of the wheels, the speed of the motor, and the time that the motor spent moving.



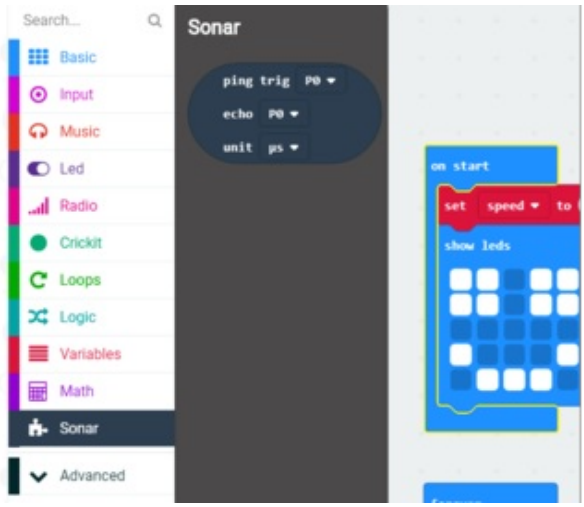
It is a rather simple calculation and I am sure that you can find some info with a little research on the internet. Also keep in mind that while it is good to do the math for things like this, things in the real world are subject to all kinds of things that are very tricky to predict. When it comes to the movement of a robots wheels, things like traction, slippage, and even the exact angle of the wheels can account for small changes in how far a robot actually travels. Advanced

robots use a bunch of sensors to account for this because better motors and wheels will only get you so far.

The Sonar Sensor

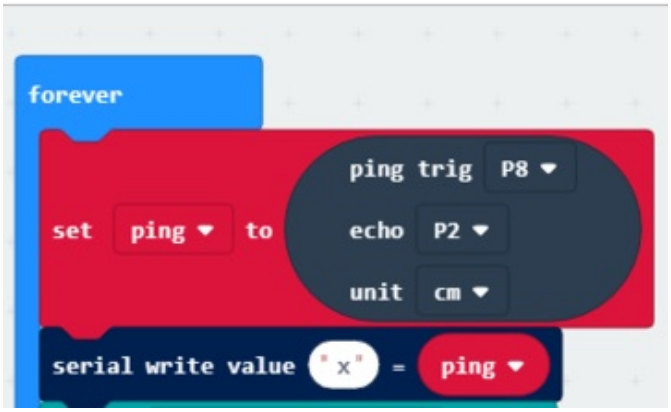
Your new robot friend can see but not like you and me. A sonar sensor used sound like bats and dolphins do for *echo location*. Basically the sensor sends out a sound called a Ping that we can not hear. That Ping will bounce off of most solid objects and the sensor will listen for the echo from that Ping. Since we know how fast sound can travel, just a little math can tell us how far away something is based on the time it took for our Ping to echo back to the sensor. You just take half of that value and you know about how far the sensor is from an object.

Thankfully someone has made a special block that we can use to read the distance from our sonar sensor and assign it to a variable so that we can use some logic to help our robot not bump into so many things when it drives.



Click on Sonar from the blocks list.

There is only one block here and it will need to be placed into a block that accepts a value like a variable.



You will want to put this `Sonar` block into a `set variable to` block because you may want to use that sensor reading a few times.

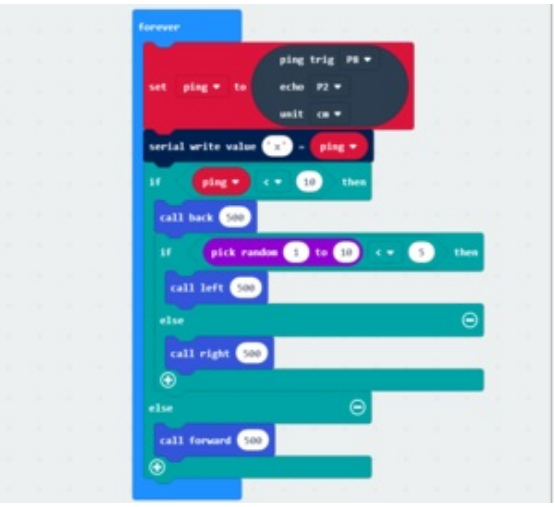
Set the `ping trig` to **P8**

Then set `echo` to **P2**

The **unit** can be set however you like.

I made a variable named `ping` and it will hold the distance in centimeters that my robot is away for any solid object in front of the sonar sensor.

I also added a `serial write value` block so that you can see the data from the sonar sensor plotted out using the Serial monitor... because data can be useful.



From there you just need to use logic to tell the robot what to do when it gets close to an object.

In my code, the robot will drive forward until the sonar sensor can detects something less than 10cm in front of it. If it detects an object less than 10cm in front, the robot will drive backwards for a half of a second. The robot will then pick a random number to decide what direction it will turn in and then turn right or left. If there is nothing in the way of the robot, it will drive forward until it sees another object less than 10cm in front.

Experiment Time!

The code example that I gave you is a good start, but you may notice that it does not work perfectly. Like the wheels, object detection can be a tricky thing. Some objects absorb sound rather than reflect it and may not give back an echo. Other objects may just bounce sound in another direction so that the echo never gets back to your sensor. Even the sensor itself only has a small range of space where it can accurately detect objects.

I challenge you to look at the data from the Serial monitor while your robot is connected to your computer. See if you can figure out the blind spots for your robot and even test to see what objects may be invisible to it.