Rover Obstacle Avoidance

If you've built our Rover Main Body you can add an Ultrasonic Distance Sensor so your Rover can avoid running into things.

Written By: Pete Prodoehl

INTRODUCTION

If you've built our Rover Main Body you can add an Ultrasonic Distance Sensor so your Rover can avoid running into things.

TOOLS:

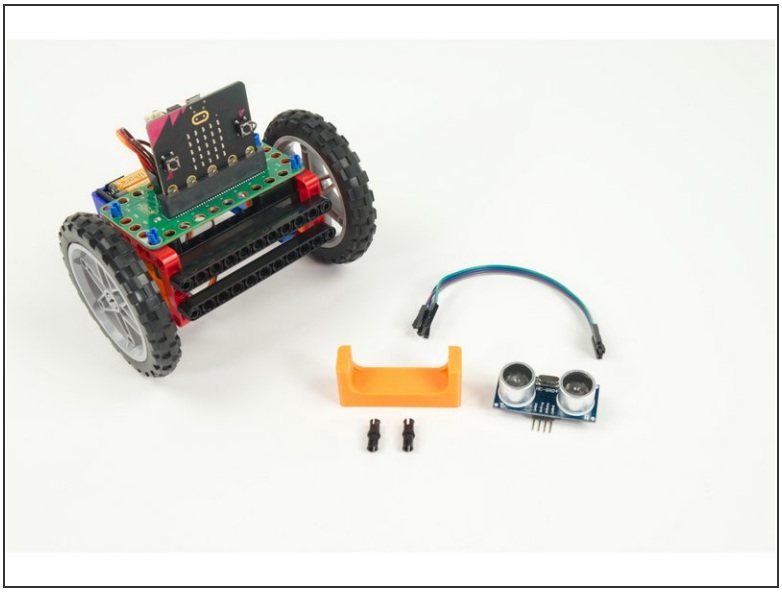
- Computer (1)

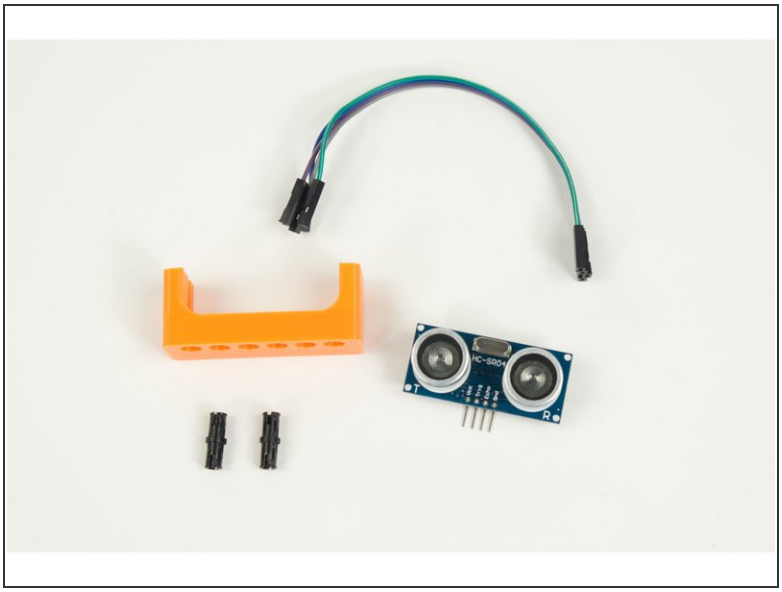
PARTS:

- Bit Board Rover Kit (1)
- micro:bit (1)
- Ultrasonic Distance Sensor (1)

Included with Rover Kit

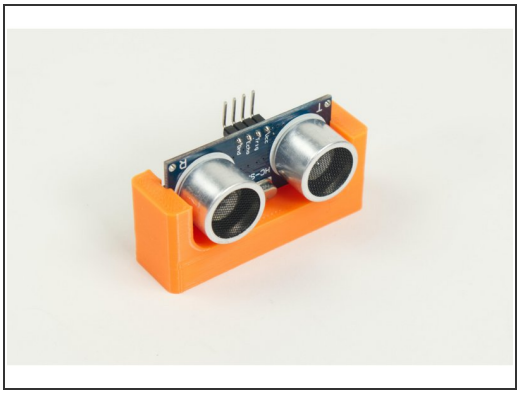
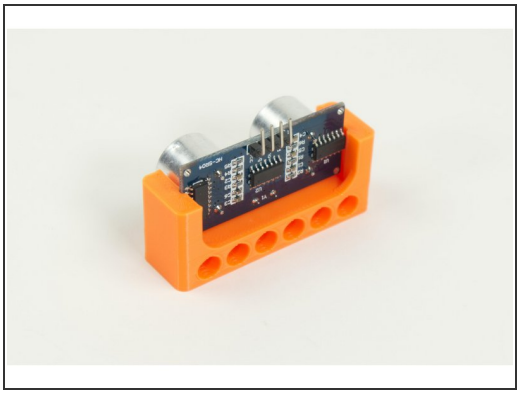
Step 1 — Gather Parts





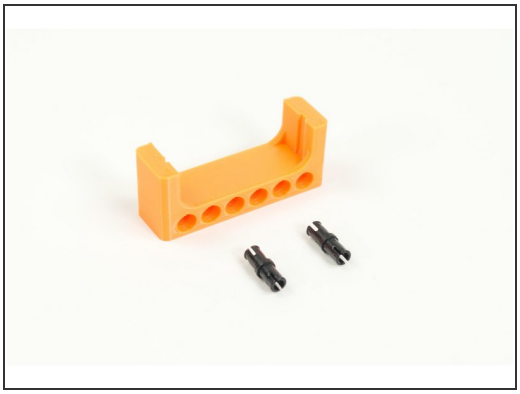
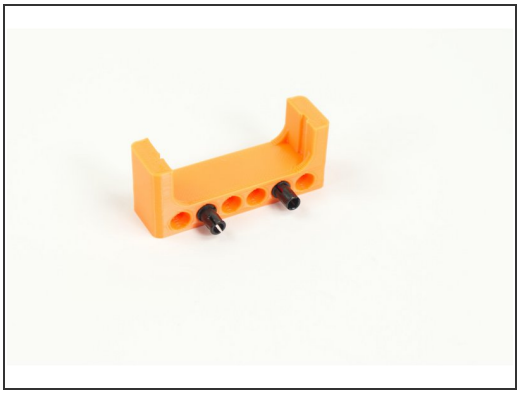
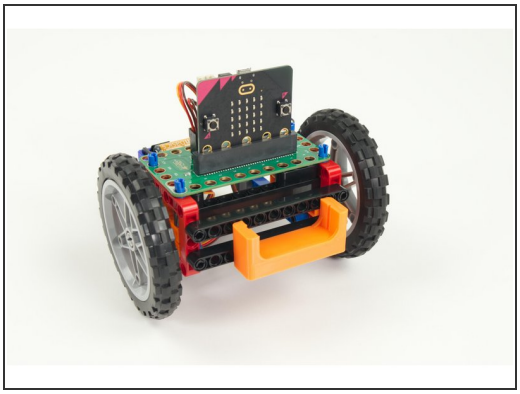
- Along with your Rover you'll need a few other components.
  - One Ultrasonic Distance Sensor.
  - Distance Sensor Holder with Holes.
  - Two pins.
  - Four F/F Jumper Wires.

Step 2 — The Distance Sensor Holder

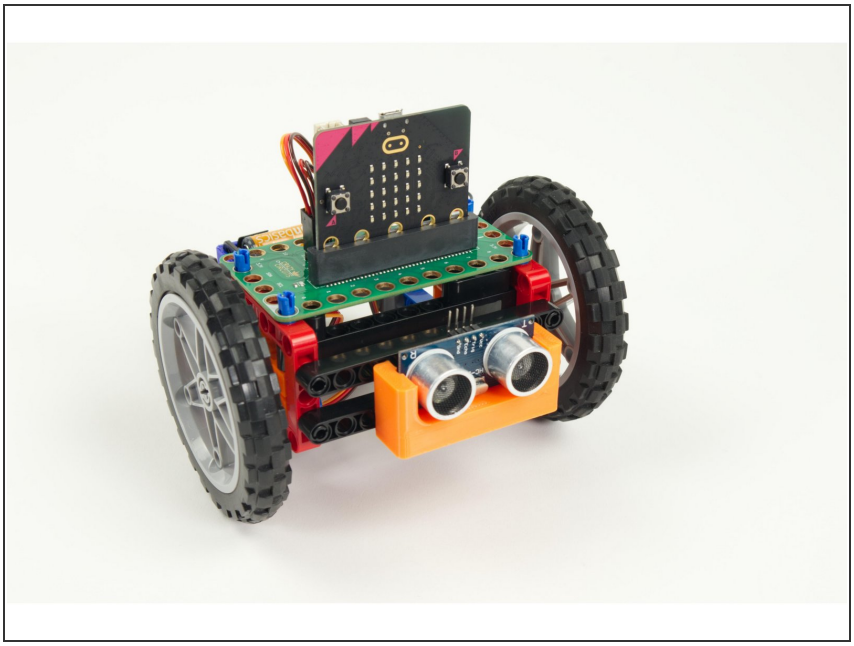

- The Distance Sensor Holder with Holes is a 3D printed part that is LEGO Technic compatible.
- The Ultrasonic Distance Sensor slides into two slots on the Holder so the four connection pins on the Sensor are at the top so we can easily add jumper wires.

Step 3 — Mount the Distance Sensor Holder

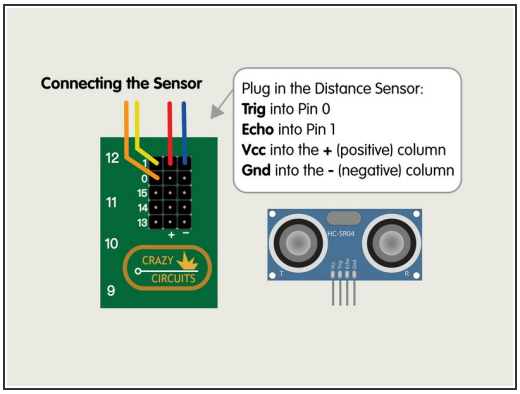
- The Distance Sensor Holder has mounting holes on the bottom as well as the back side.
- To mount it to the front of the Rover we'll use two pins placed into holes on the back.
- Slide the Distance Sensor Holder with pins into the beam on the front of the Rover as shown.

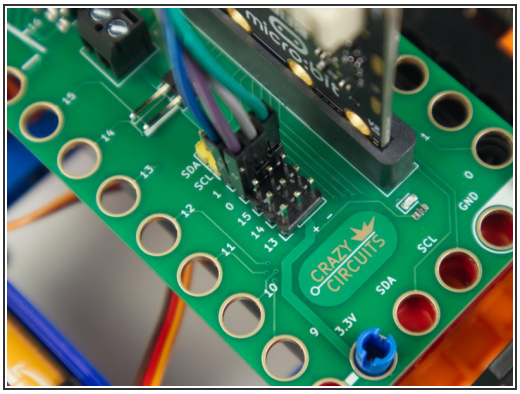
Step 4 — Add the Distance Sensor

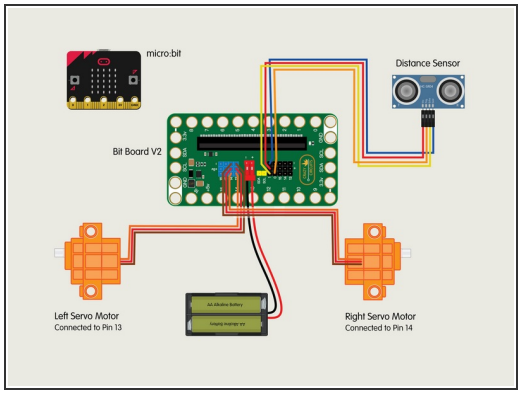


- After you have the Distance Sensor Holder mounted to the front of your Rover you can slide the Distance Sensor into place.

Step 5 — Connect the Distance Sensor



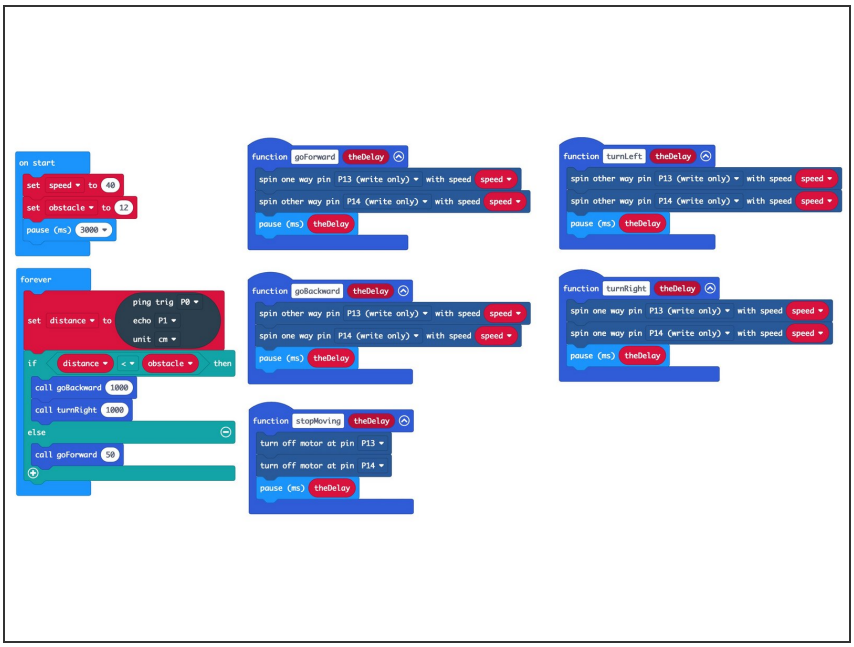




- Plug in the Distance Sensor and connect the **Trig** pin to **Pin 0**, the **Echo** to **Pin 1**, and then **Vcc** to a pin in the **+** (positive) column and **Gnd** to a pin in the - (negative) column.

Wire colors in the photo may not match wire colors in the diagram. The important part is to plug each pin into the correct position.

Step 6 — Load the Code



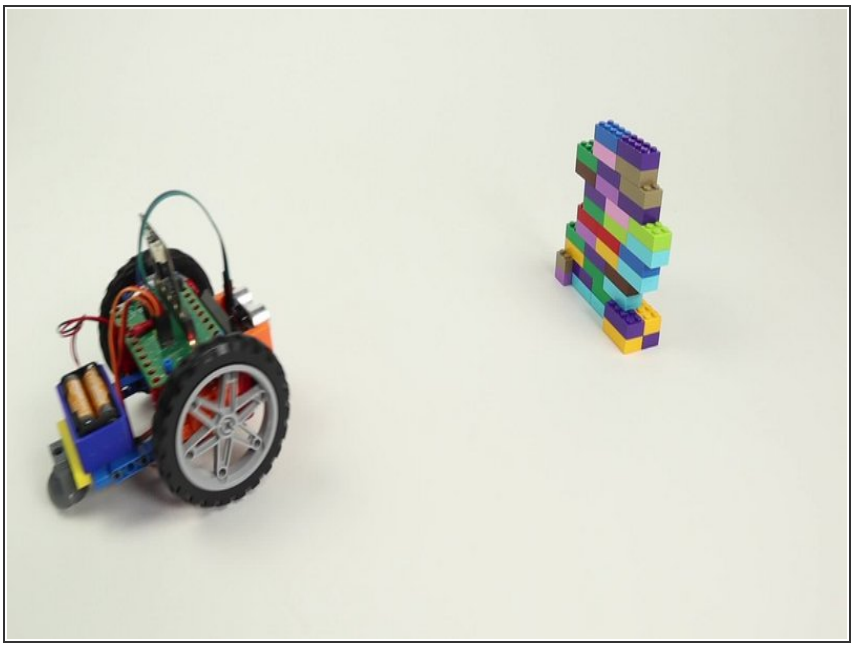
⚠ If you've never used a micro:bit before you'll want to check out this guide: Bit Board V2 Setup and Use

- We're going to load the following code for our **Rover Obstacle Avoidance** program: https://makecode.microbit.org/_4wHXzvEVj...

⚠ Note! We've put in a 3 second delay. It's the pause block in the on start section. This is so your Rover doesn't roll away as soon as the code is uploaded (or a battery pack is connected).

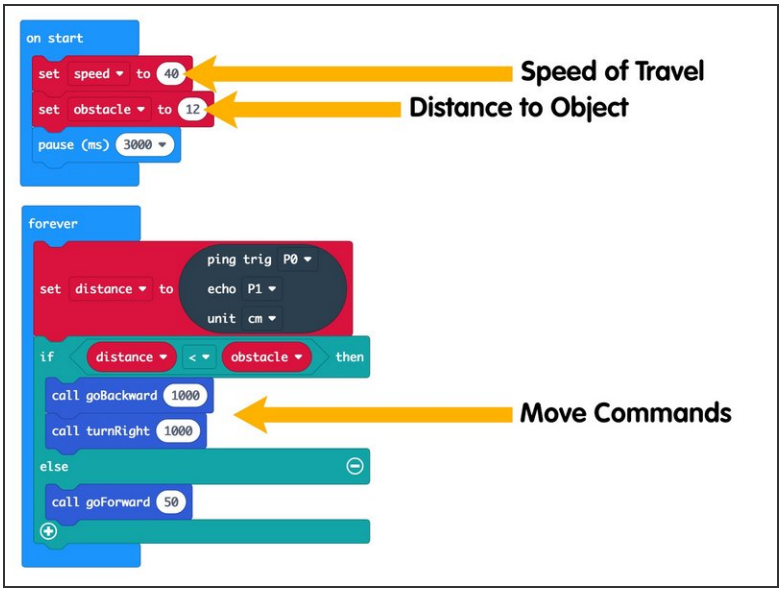
- We recommend you remove the micro:bit from the Bit Board for programming, then after the code is loaded disconnect the USB cable and place the micro:bit into the Bit Board.
- The 3 second delay is there so that when you plug in the battery pack you have a few seconds to put the Rover down on the floor before it starts moving.

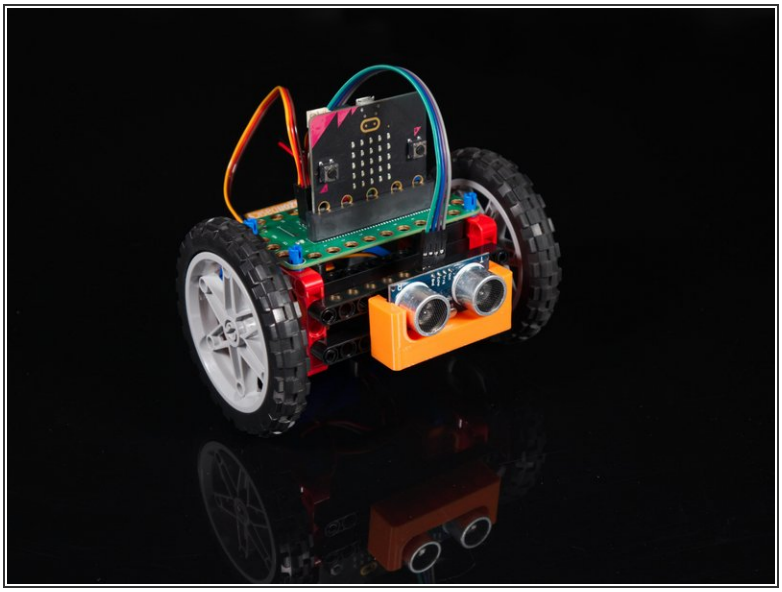
Step 7 — Test it Out!



- Let's test the Rover with the Distance Sensor!
- Remember, when you plug in the battery pack the Rover will wait 3 seconds before it starts moving.
- Our code tells the Rover to check the Distance Sensor to see if an obstacle is in front of it. **If** there is an obstacle the Rover will move *backwards*, then *turn right.* **Else** the Rover will move *forward*.
- The Distance Sensor will keep looking for something in front of it, and move forward when it can, and backup and then turn right, when it cannot move forward.

Step 8 — Take it Further





- While you probably don't want to change any of the **functions** in the code (**goForward**, **goBackward**, **turnLeft**, **turnRight**, and **stopMoving**) there are a few things you can change.
- **Speed of Travel** will affect how fast the Rover moves. We do not recommend setting this higher than 50. (See note below.)
- **Distance to Object** is how many Centimeters away an object is from the sensor. You can adjust this number higher or lower.
- The **Speed of Travel** and the **Distance to Object** affect each other. If the Rover is moving too fast and the distance to an object is a small amount the Rover might hit the object before it can react and avoid it! Experimenting with values is a great way to experiment.
- Finally, the **Move Commands** could be adjusted as well. Right now when an object is detected the Rover will backup for one second (1000 ms) and turn right for one second (1000 ms). You could change these times to see how it affects the Rover. You could also choose to turn left instead of right by calling the **turnLeft** function.
- If you're up for a challenge you could try to make the Rover go *around* an object. What would the code for that look like?