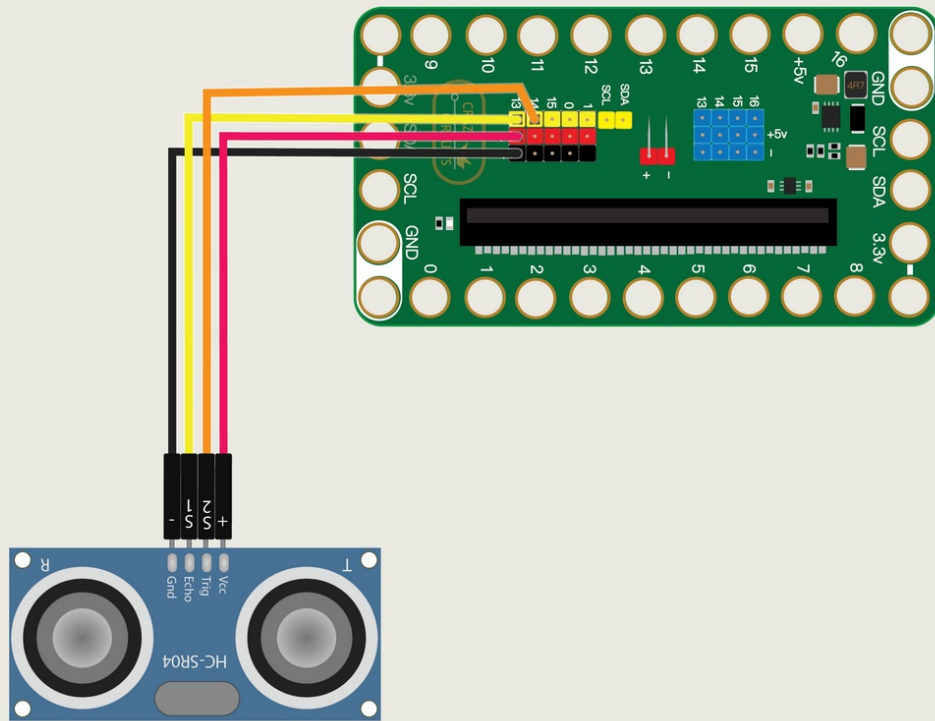




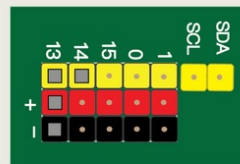
CH34-D - Distance Sensor Pixels

Connect a Distance Sensor to a Bit Board and view the output using the built-in LED Matrix on the micro:bit

Written By: Pete Prodoehl



DIST	Bit Board
VCC	-> + (Positive) Pin
TRIG	-> PIN 14 (S2)
ECHO	-> PIN 13 (S1)
GND	-> - (Negative) Pin



INTRODUCTION

Connect a Distance Sensor to a Bit Board and view the output using the built-in LED Matrix on the micro:bit



TOOLS:

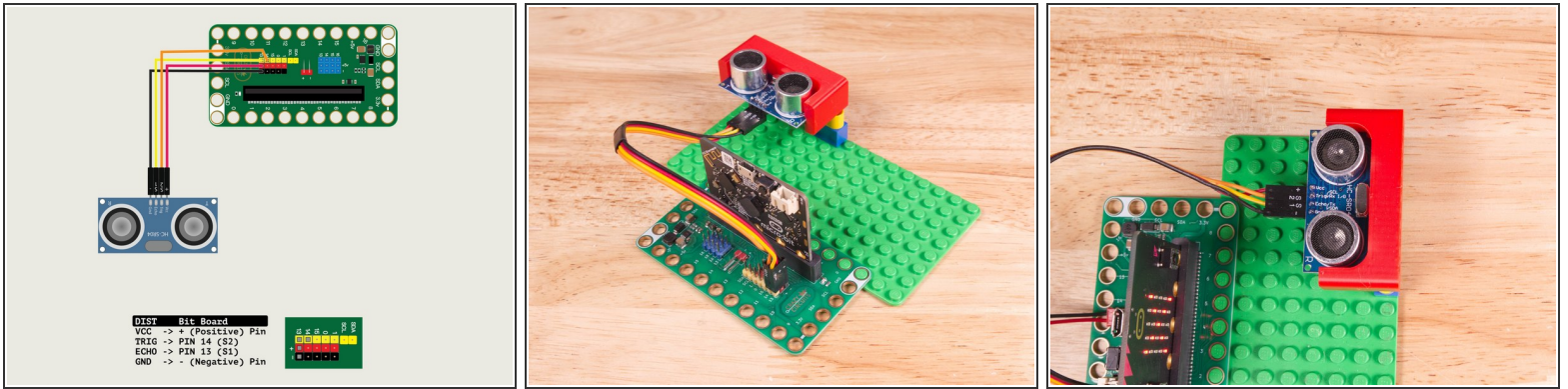
- [Computer](#) (1)
- [Scissors](#) (1)



PARTS:

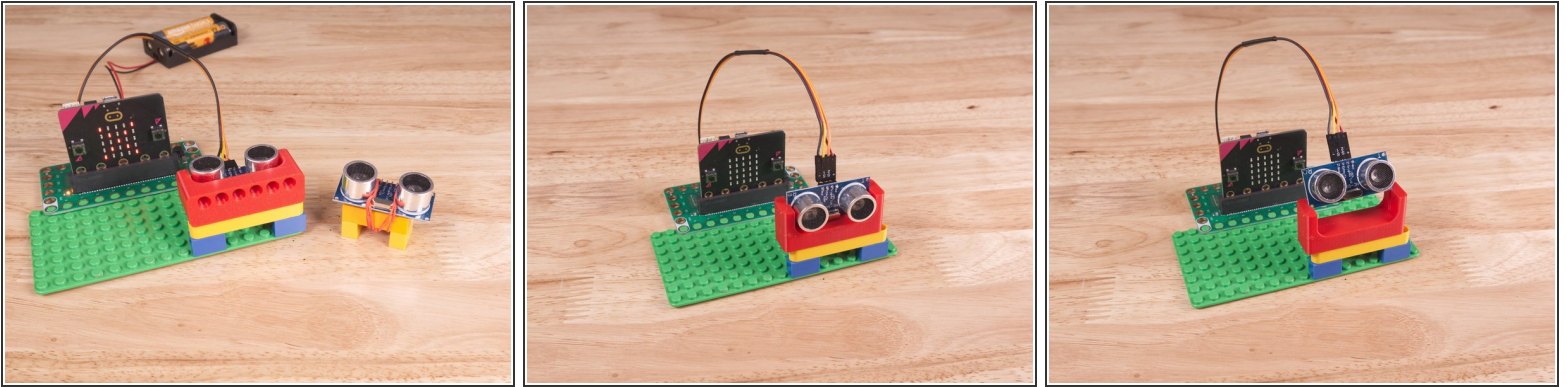
- [Crazy Circuits Bit Board](#) (1)
- [micro:bit](#) (1)
- [Crazy Circuits Ribbon Cable](#) (1)
- [Distance Sensor](#) (1)

Step 1 — Connect the Distance Sensor



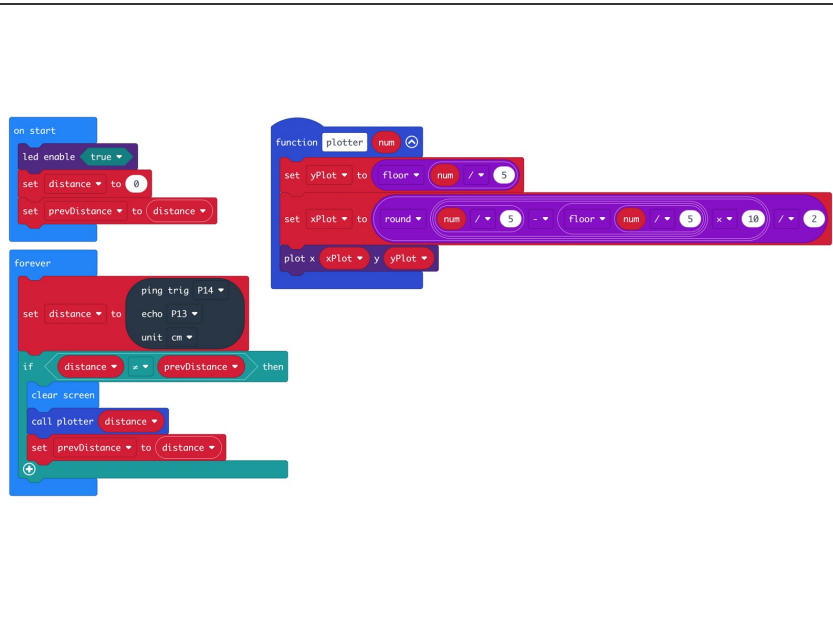
- We'll connect the **Distance Sensor** with a **Crazy Circuits Ribbon Cable** using all four wires.
- Plug the three-wire connector of the **Crazy Circuits Ribbon Cable** into the row with **Pin 13** on the Bit Board. The wire colors should match the pin header colors, with **S1** (yellow) going to the yellow pin.
- Next, plug **S2** (orange) into **Pin 14**. (*S2 is separated from the three-wire connector to allow you to plug it into another row.*)
- Plug **+** (red) into **VCC**; **S2** (orange) into **TRIG**; **S1** (yellow) into **ECHO**; and **-** (black) into **GND**.

Step 2 — Mounting the Sensor



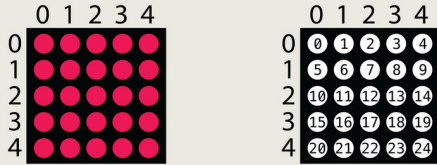
- We used a 3D printed part we designed to hold the Distance Sensor in place on the LEGO baseplate.
- A few Technic pins, a Technic beam, and some LEGO bricks are all that is needed.
- Alternately you can use some LEGO bricks and a rubber band or tape to hold the sensor in place. *(But don't use anything conductive!)*
- 3D print your own [Distance Sensor Holder with Holes](#). (It's the same one we use in our Bit Board Rover Kit).

Step 3 — 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 **Distance Sensor Pixels** program: https://makecode.microbit.org/_ihJLfKJLk...
- In the forever block we'll set **Pin 14** and **Pin 13** to TRIG and ECHO (respectively) so we can read the Distance Sensor. This allows us to continually read the sensor and assign the value to **distance**.
- The forever block looks pretty simple, and that's because all of the heavy lifting is done in the plotter function.
- The plotter function takes a single number between **1 and 25** and then gets the values to plot it on the **X/Y** grid of LEDs.

Step 4 — The Function



```
yPlot = Math.floor(num / 5)
13 / 5 = 2.6

yPlot = Math.floor(num / 5)
floor(2.6) = 2
```

```
xPlot = Math.round((num / 5 - Math.floor(num / 5)) * 10) / 2
13 / 5 = 2.6

xPlot = Math.round((num / 5 - Math.floor(num / 5)) * 10) / 2
floor(2.6) = 2

xPlot = Math.round((num / 5 - Math.floor(num / 5)) * 10) / 2
13 / 5 = 2.6

xPlot = Math.round((num / 5 - Math.floor(num / 5)) * 10) / 2
2.6 - 2 = 0.6

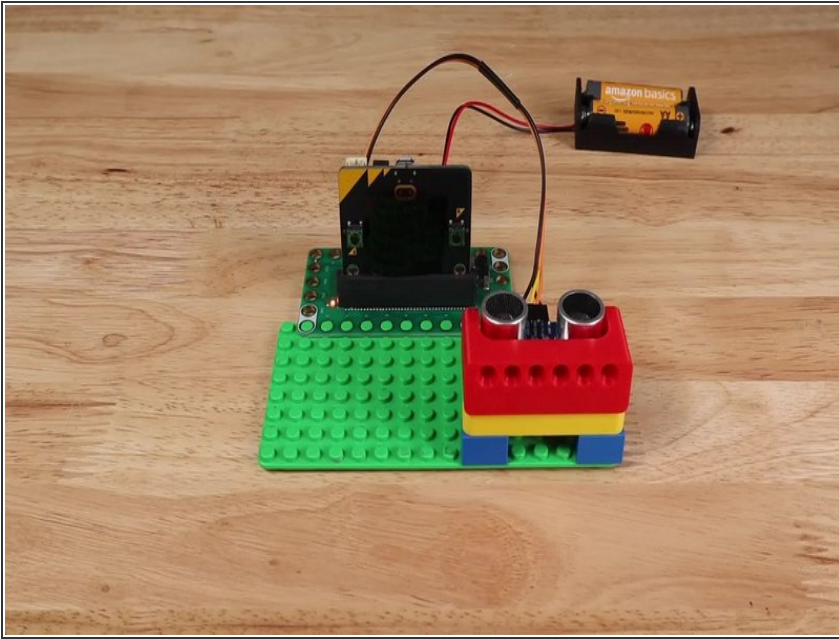
xPlot = Math.round((num / 5 - Math.floor(num / 5)) * 10) / 2
0.6 * 10 = 6.0

xPlot = Math.round((num / 5 - Math.floor(num / 5)) * 10) / 2
round(6) = 6

xPlot = Math.round((num / 5 - Math.floor(num / 5)) * 10) / 2
6 / 2 = 3
```

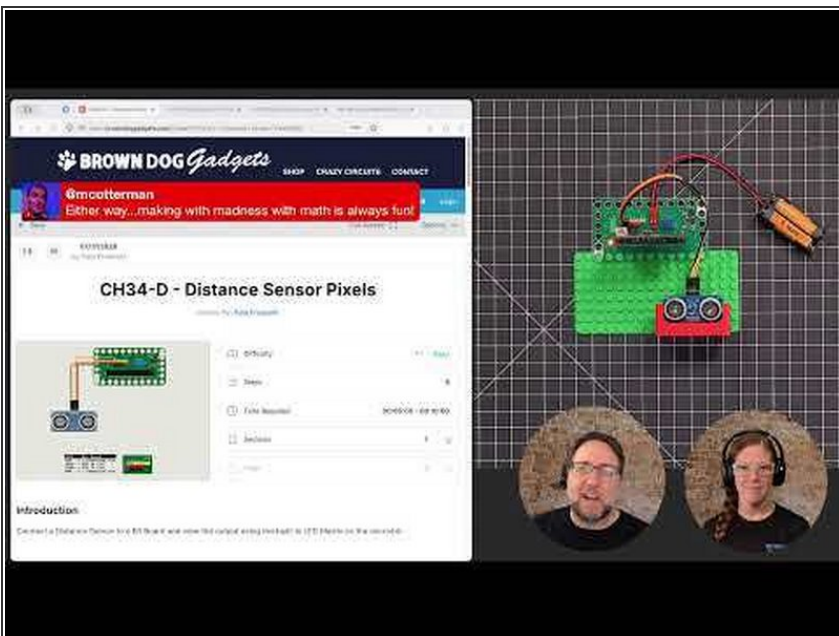
- We'll first number our LEDs from **1 to 25** and look at how that maps out to our **X axis** (0 through 4) and our **Y axis** (0 through 4).
- The **Y** value is easy. When we pass **13** to our function we divide by **5** (which is the number of LED rows) and get **2.6**. If we then **floor** that value (which means to "round down") we get **2**. (13 is in row number **2**.)
- The **X** value is a little more complicated, but you can work through it just like any calculation, starting from the innermost parenthesis working outward.
- The decimal part of the **2.6** gives us the **6** which eventually becomes **3** and you can see that **13** is in column **3** of the LED matrix.
- ☑ The code for this was a bit complex and we ended up writing it in Python outside of MakeCode and then implemented it using the blocks available... and it works!
- ☑ We copied the text from the Python mode of MakeCode to create the diagrams for this step. If you are familiar with Python (or JavaScript!) remember that you can always switch the mode in MakeCode.

Step 5 — Test it Out!



- Once the code is loaded it should start running immediately.
- ❗ You can power the micro:bit via the USB cable you used to load the code or you can use a battery pack plugged into the Bit Board.
- Move your hand (or a flat object) in front of the sensor. As you get closer you'll see the LED matrix react by moving the lit LED corresponding from 1 to 25.

Step 6 — Follow Along



- Follow along with our recorded Live Stream!
- ❗ You can watch the full video of us walking through this project, along with explaining and exploring the code:
<https://youtube.com/live/BTIGtgqRKUM>