“IUSE” Hydroponics in a Nutshell

Or: some ways *you* can use hydroponics, too!

🚰 💪

Brought to you by Yihong Cheng, Dave Jackson, and Chris Asante
Innovation in Urban Science Education (IUSE) Lab, Boston College

This material is based upon work supported by the National Science Foundation under Grant DRL #1814001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Agenda

1. Intro: welcome, brief introductions, gauging interest
2. Breakout Groups
    a. Main Room: (Natural) Science- and Engineering-Intensive
    b. Breakout Room: Coding-Intensive
3. Outro: Takeaways & next steps

(Natural) Science- & Engineering-Intensive

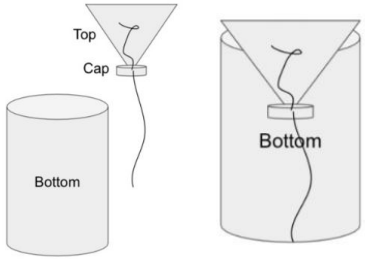
Brought to you by Dave Jackson

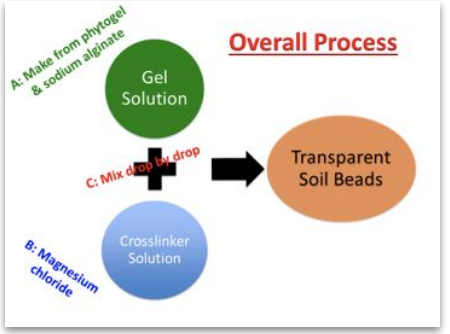
DRL #1814001

Overview of (natural-)science and engineering-intensive projects

1. Bottle hydroponics
2. Tabletop greenhouses
3. Two-tier systems
4. Transparent soil

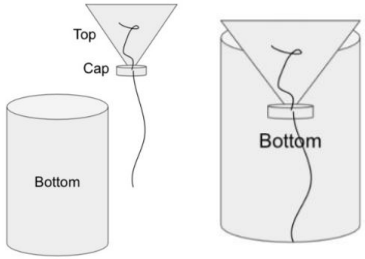
diagrams by Olivia L.





1. Bottle hydroponics

[...]



2. Tabletop greenhouses

1. Iteration 1: MicroPython + WioLink

https://growthings.netlify.app/unit1/lesson1/

2. Iterations 2+: makecode* + micro:bit

*can be blocked-based, (full-)Python, or JavaScript

Module 1 (moisture) Module 2 (air) Module 3 (servo + watering)

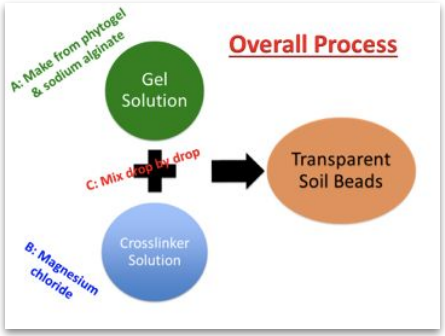
ALL FILES [warning: chaotic! :)]

3. Two-tier systems

- Classroom-, club-, or camp-based
  - Especially see “new design” on Slide 8 here
  - More materials here

- Home- / family-based
  - “LEaFS” (Learning Ecosystems and Family Science)
  - https://sites.google.com/bc.edu/leafs-project

4. Transparent soil



- Yes, it is pretty darn transparent! (See picture at left.)
- Relatively low-cost
- Materials you can easily order online
- Equipment you likely already have (e.g., use a microwave, as opposed to an autoclave)
- Applications (*some*)
  - Chemistry: Formulations, concentrations, reactions, lab techniques, etc.
  - Biology: Root growth (e.g., length, geometry, etc.)

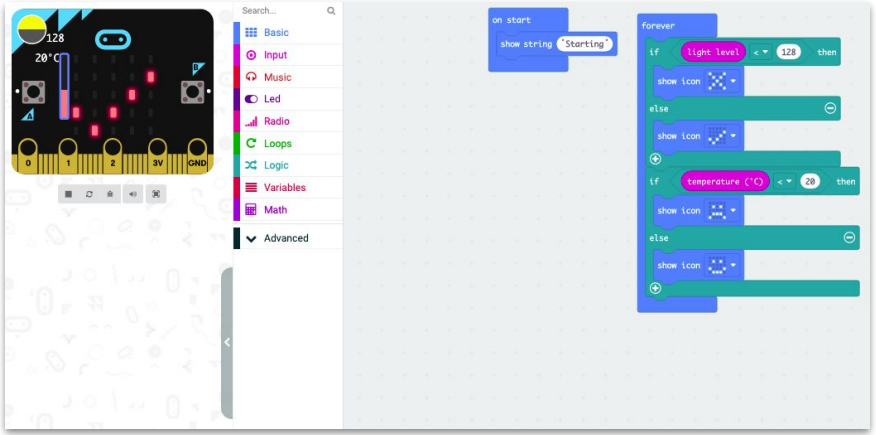
4+: Contexts

- Mix of urban and urban-ring partners
- In-school-time (so far)
  - MS environmental(/general) science
  - HS chemistry
  - HS biology
  - ES *proposed* (esp. with multilingual learning and computatation)
- Out-of-school time (so far)
  - College Bound: ~14 Saturdays per school year + ~12 days per summer
  - Food Justice Ambassadors (FJA's)
    - Summer: Practitioners train HS students (a.k.a. FJA's)
    - School-year: HS students teach+mentor MS students

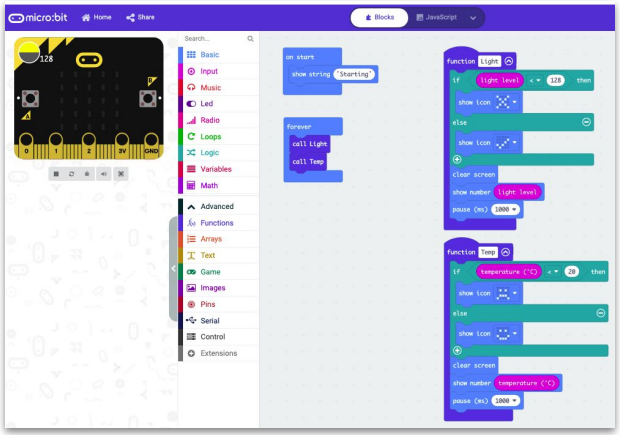
...and now, an activity!

Sample code in this folder:
https://drive.google.com/drive/folders/1-j638wm0mlB1puUwZiIdmlKpWfbERMlx?usp=sharing

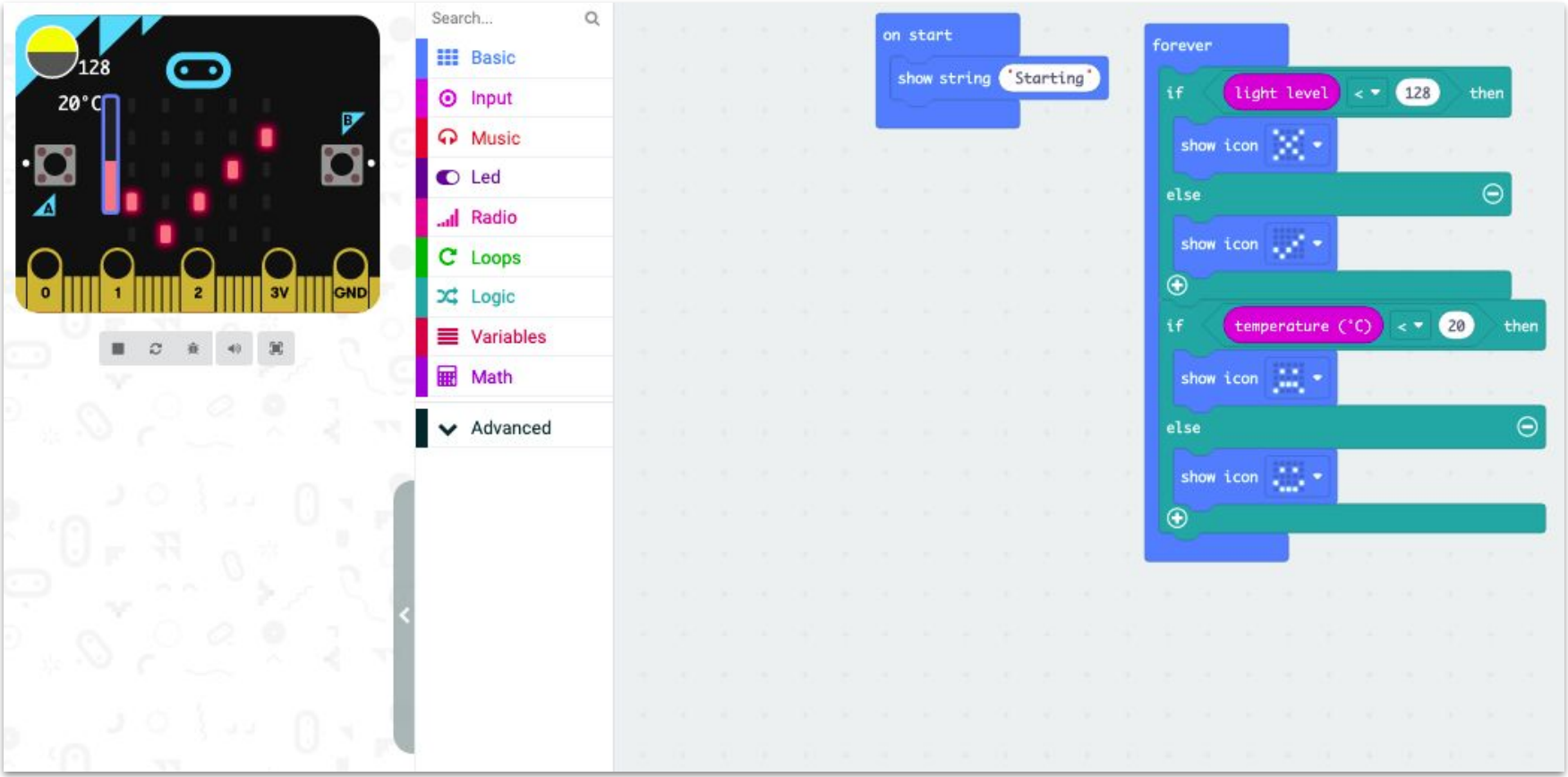
without functions here



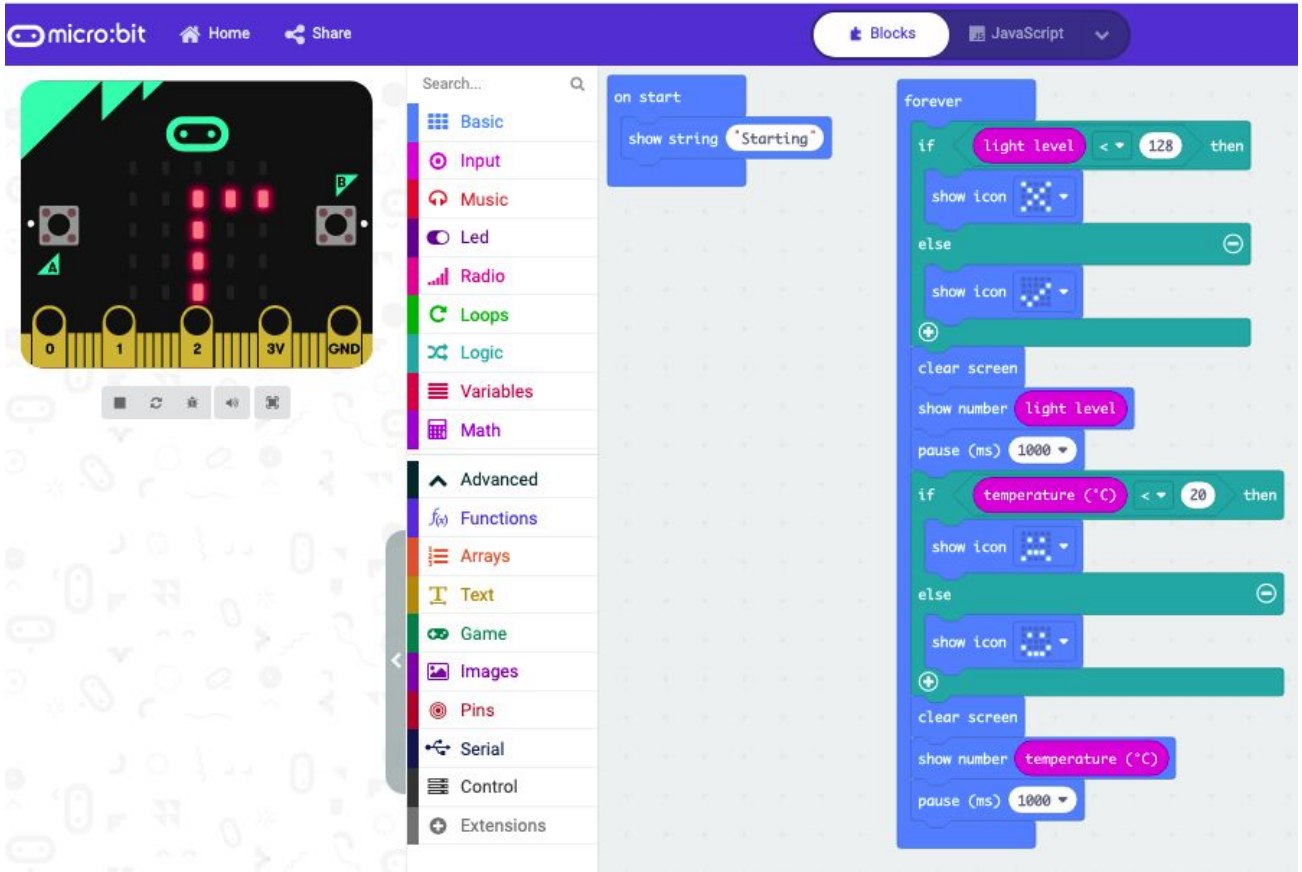
with functions here



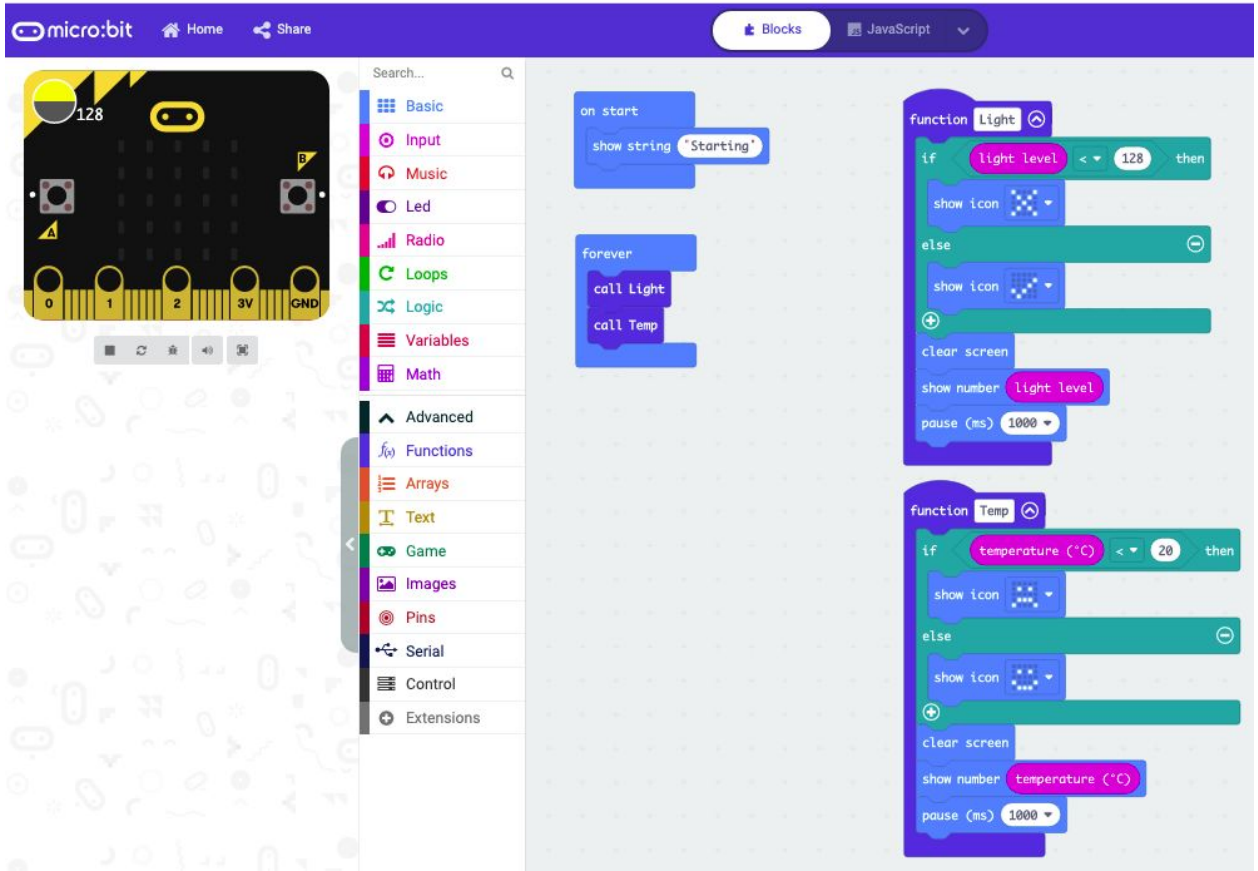
Part 1: Qualitative-only



Part 2: Qualitative and quantitative

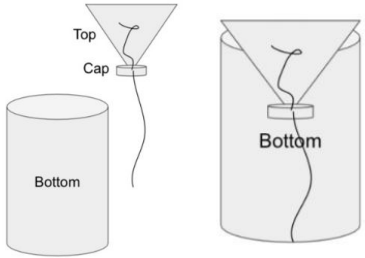


Part 3: Qualitative and quantitative, with functions



Q&A Re: (natural) science- and engineering-intensive projects

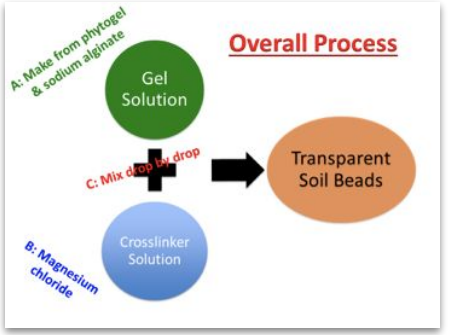
1. Bottle hydroponics



2. Tabletop greenhouses

3. Two-tier systems

4. Transparent soil

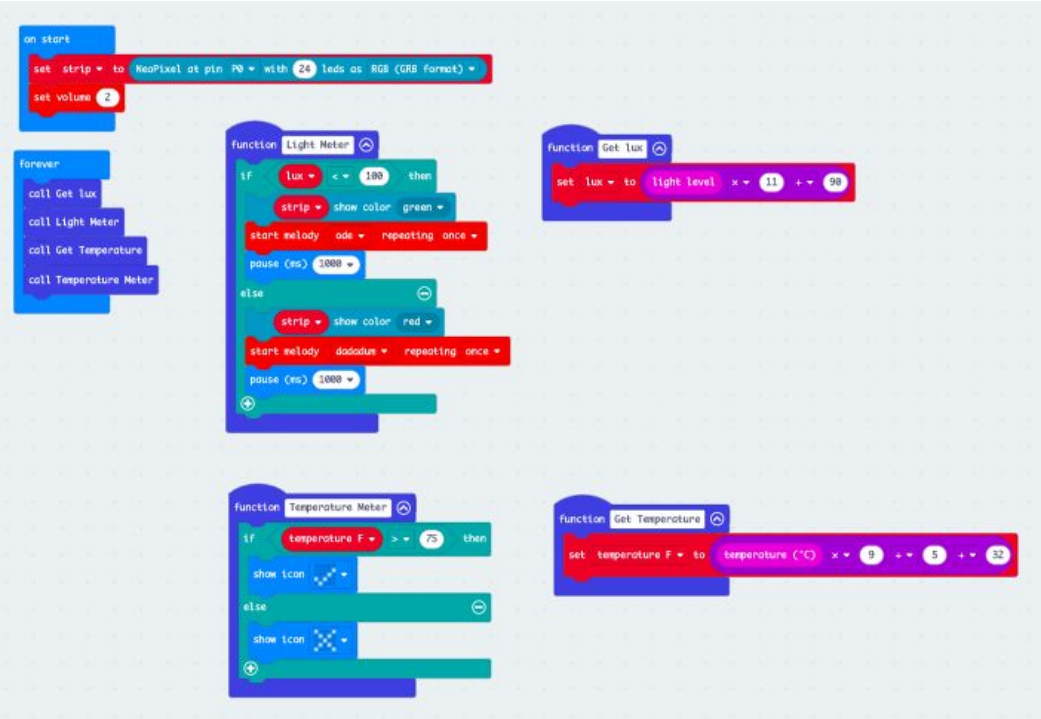


Coding-Intensive

Brought to you by Yihong Cheng

DRL #1814001

How to Monitor Light Level and Temperature...



...and receive alarms when light level drops too low or when air temperature is too high?

This program will help.

On Start

Codes in the on start loop defines devices and/or constants that we will use the whole time. Here it's defining NeoPixel to be connected to pin 0 and setting the volume of our future alarm to be 2 so that it won't cover my talking.

Forever

This forever loop ensures that functions inside it will keep on running forever and ever and ever.

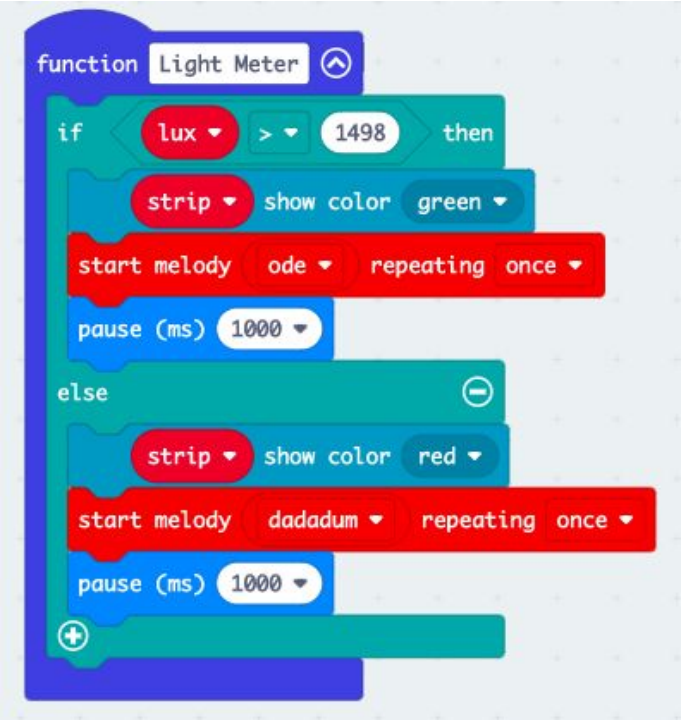
Each function has its own short or long story, which we will go through right away.

Get Lux

This block converts micro:bit's built-in light sensor's reading of light level to a new variable named lux.

The conversion is needed because lux is a more widely used unit and experiments show that micro:bit's light level has a linear relationship with it.

Light Meter



This function tells mico:bit to turn the LED strip green and play some happy Ode to Joy if lux is higher than 128x11+90=1498, so that you know everything is ok.

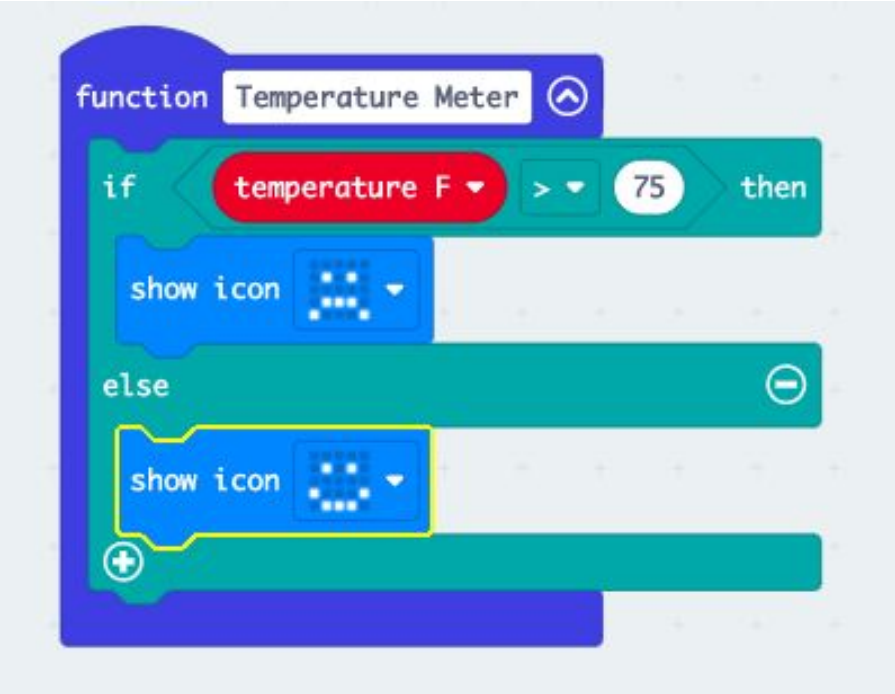
If lux is lower than that, it will play some intense Fate Symphony and turn the LED strip red, so that you know your plants need saving.

Get Temperature

Similar to the Get Lux function, this function converts micro:bit's built in temperature sensor's readings of temperature into a new variable, Temperature F.

The original reading is in Celsius so we need some math formula to convert it into Fahrenheit.

Temperature Meter



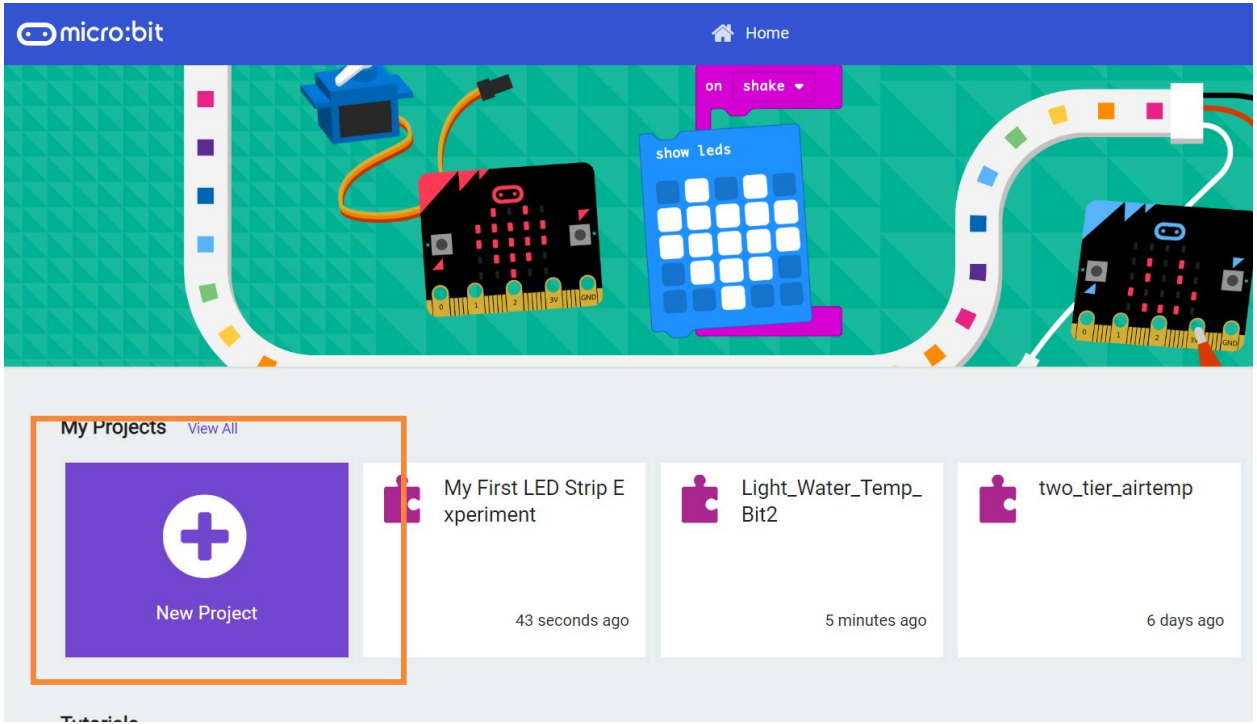
This function tells micro:bit to display a sad face when temperature is above 75 degrees, so that you know it’s time to blow some wind to your plants.

Otherwise, micro:bit will display a happy face so that you know you don’t need to worry about your plants being overheated.

Extra Slides: NeoPixel Traffic Light

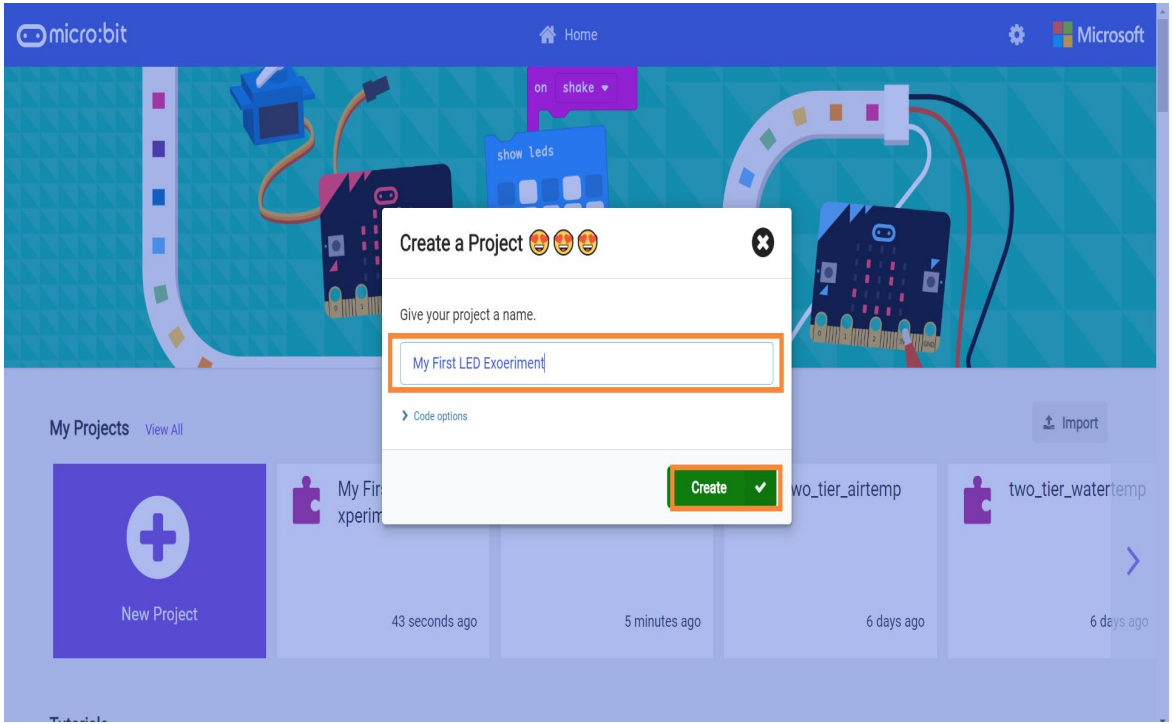
LED Light Strip Simulator

Step 1: Create Your New Project



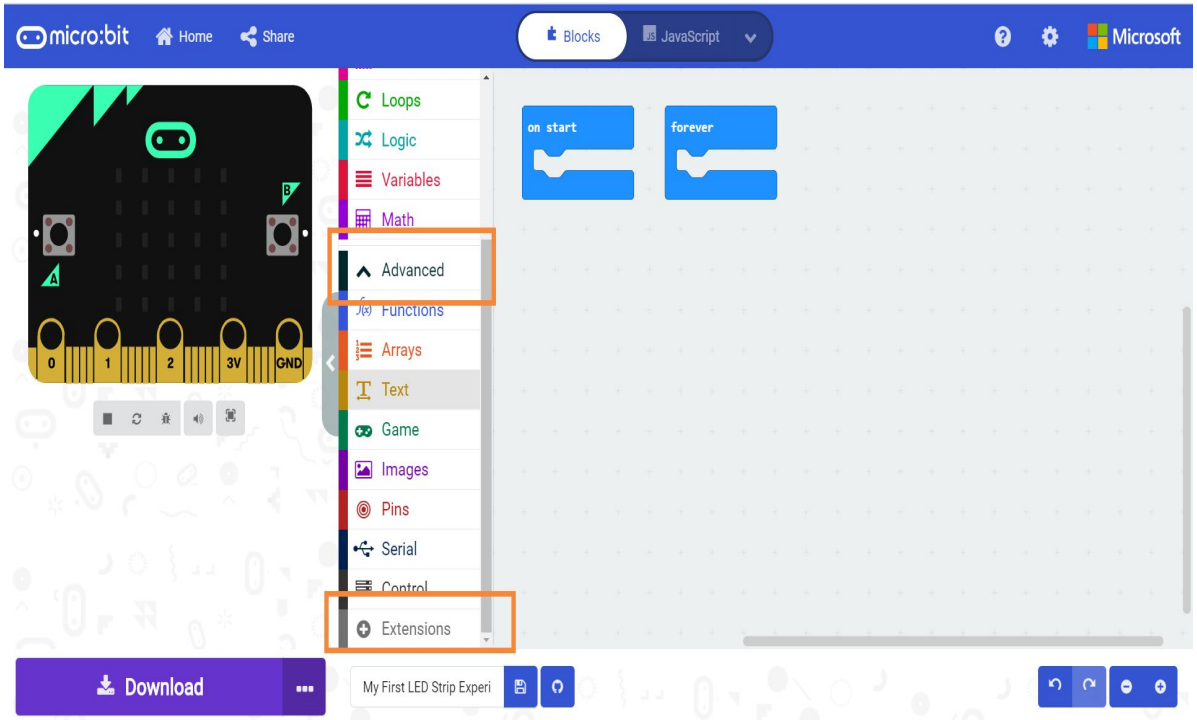
makecode.microbit.org

Step 2: Name Your Project



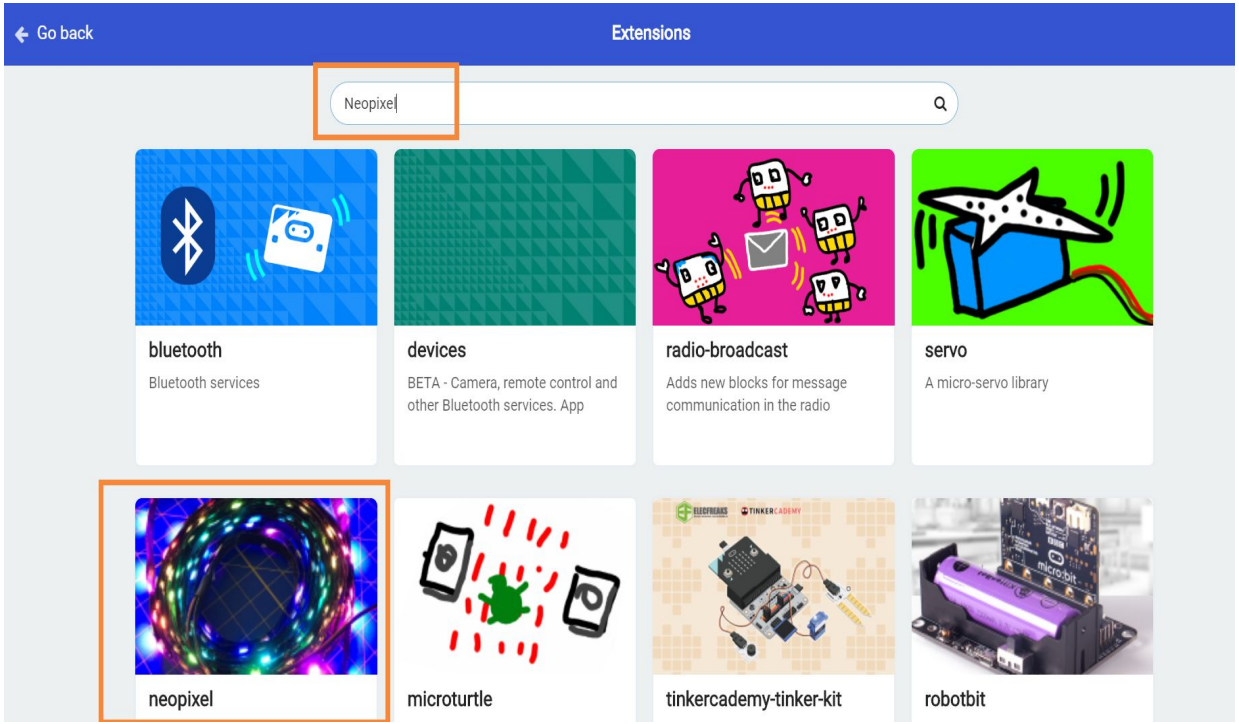
Let’s call it “My First LED Experiment” for now.

Step 3: Add Extensions



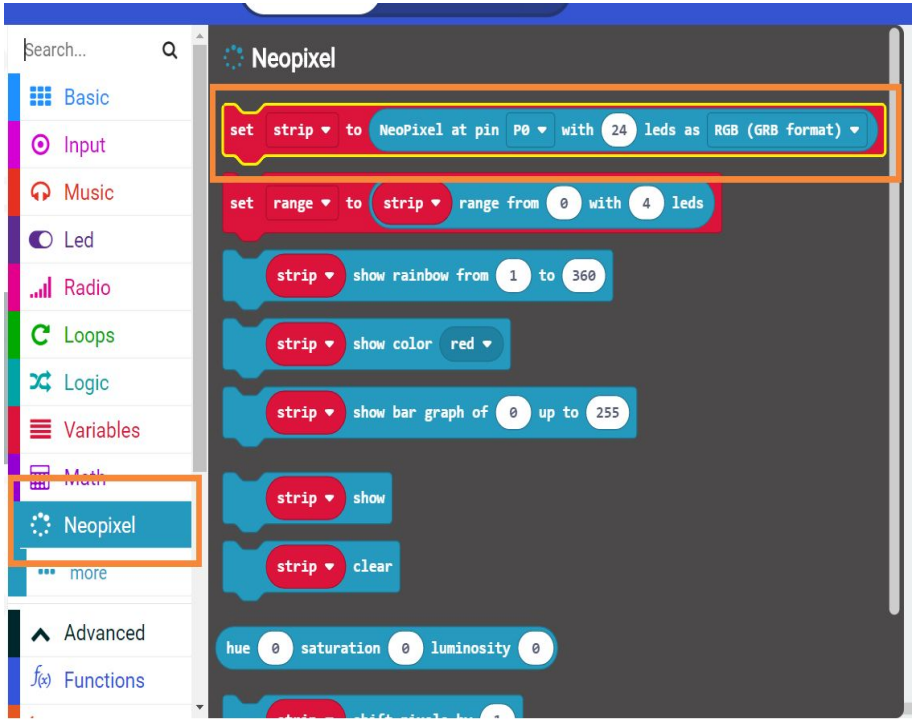
Click “Advanced”, then click “Extensions”

Step 4: Add neopixel



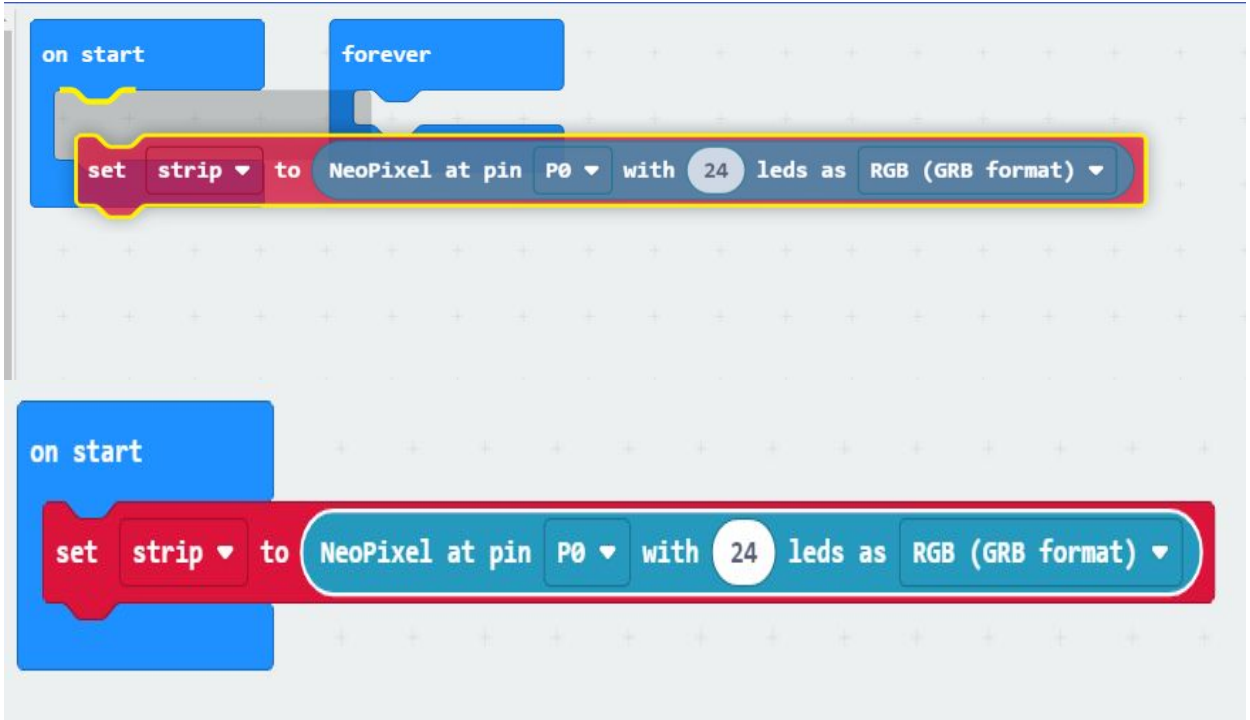
Click the large neopixel icon or type “neopixel” in the search bar if the icon is not there.

Step 5: set strip



Click “Neopixel”, then click “set strip to (xxx)” and hold on to it and drag the block out to the right.

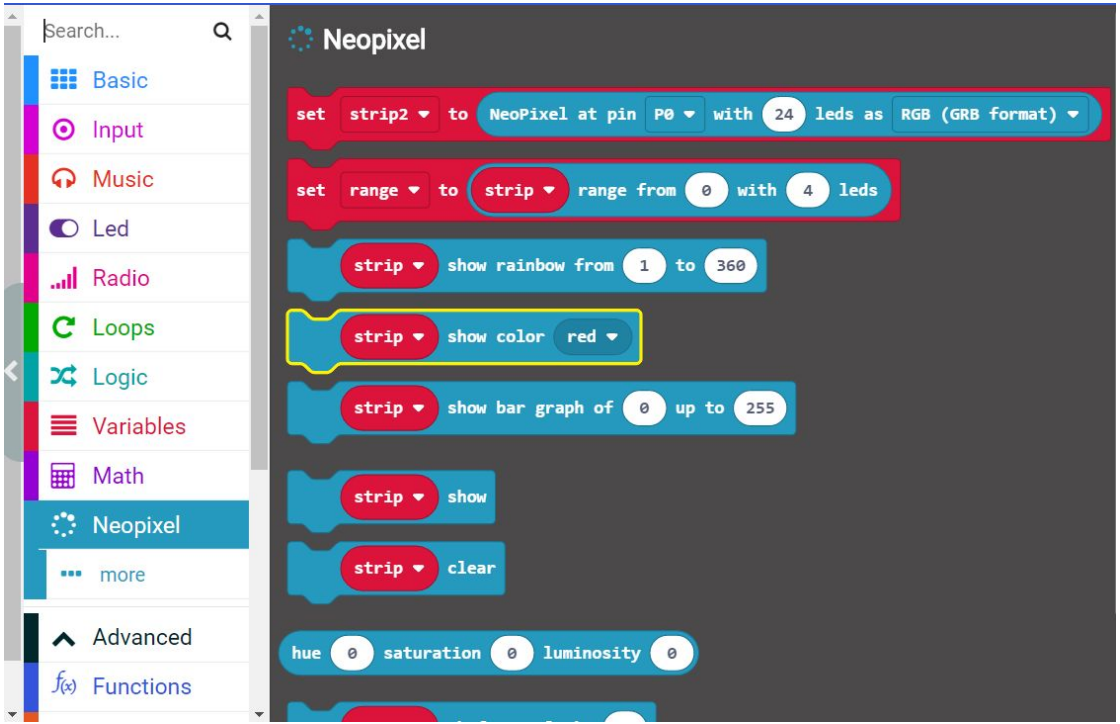
Step 6: On Start



Move the “set strip” block next to the “on start” block until you see a yellow margin appear under “on start”. Release the click and the “set strip” block will fall in correctly.

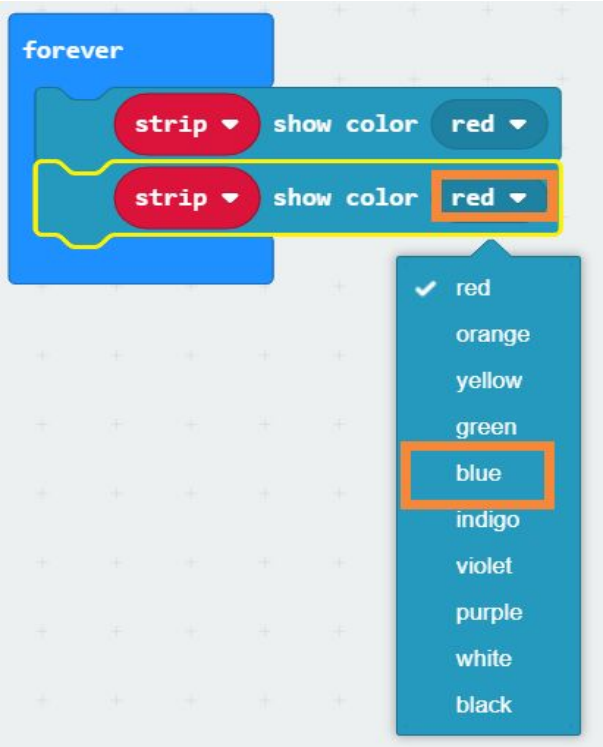
Now micro:bit is able to recognize NeoPixel.

Step 7: Strip Show Color



Go to “Neopixel” again and drag “strip show color xxx” out.

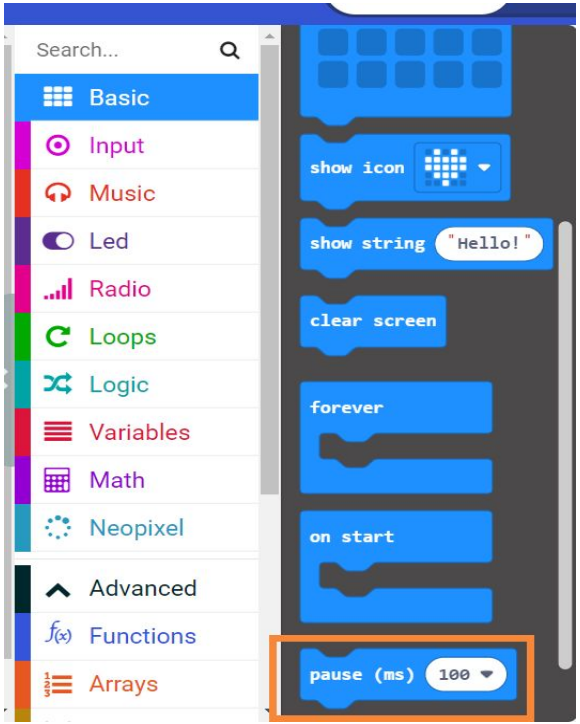
Step 8: Forever Loop



Put “strip show color red” into a forever loop, then repeat the previous step and add another one of it. This time, open the drop down menu for color to select another color you like.

The forever loop is basically going to tell micro:bit to keep on executing everything inside it forever and ever and ever.

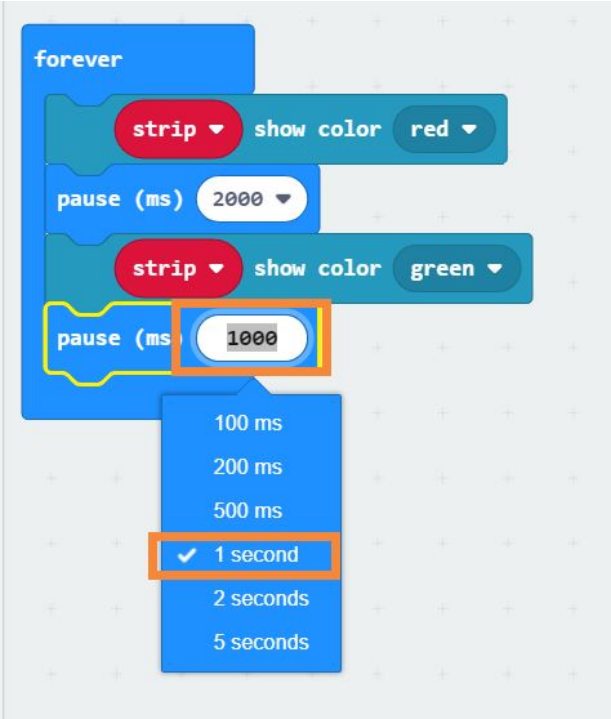
Step 9: Pause



Now we need to drag out two “pause (ms)” blocks from the “Basic” menu, so that we can define how long each color will stay later.

What “pause” does is essentially telling micro:bit to keep on doing what it’s doing for x milliseconds before taking the next action in line.

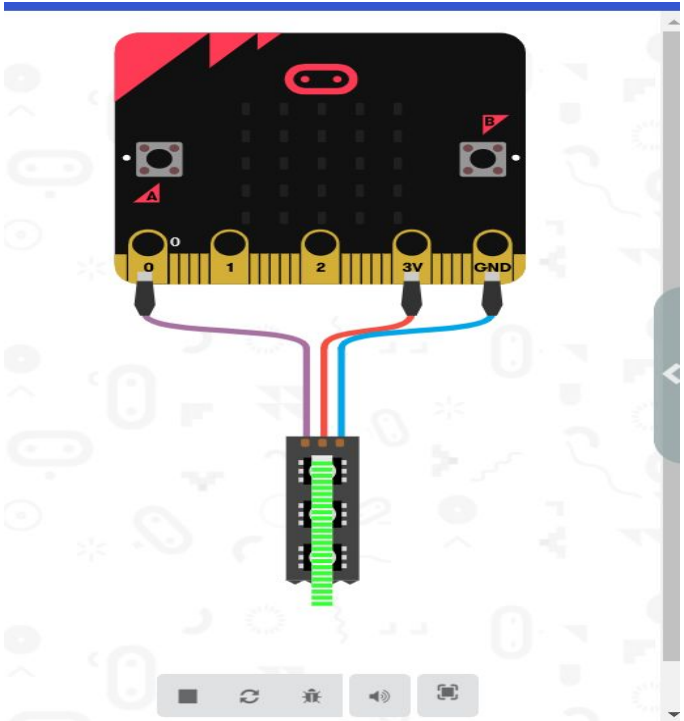
Step 10: Define the Duration of Pause



For this step, place one“pause” block after each “strip show color” block, then open the drop down menus to define how long you want the pauses to last.

You can also directly type specific values in the white bubble. Note that the unit here is milliseconds.

Step 11: Check out your simulator



Your simulator should now look like this and be blinking in two colors in turns!

Quiz Time

Using what you just learned, can you design a traffic light that stays green for 4 seconds, yellow for 2 seconds, and red for 4 seconds?