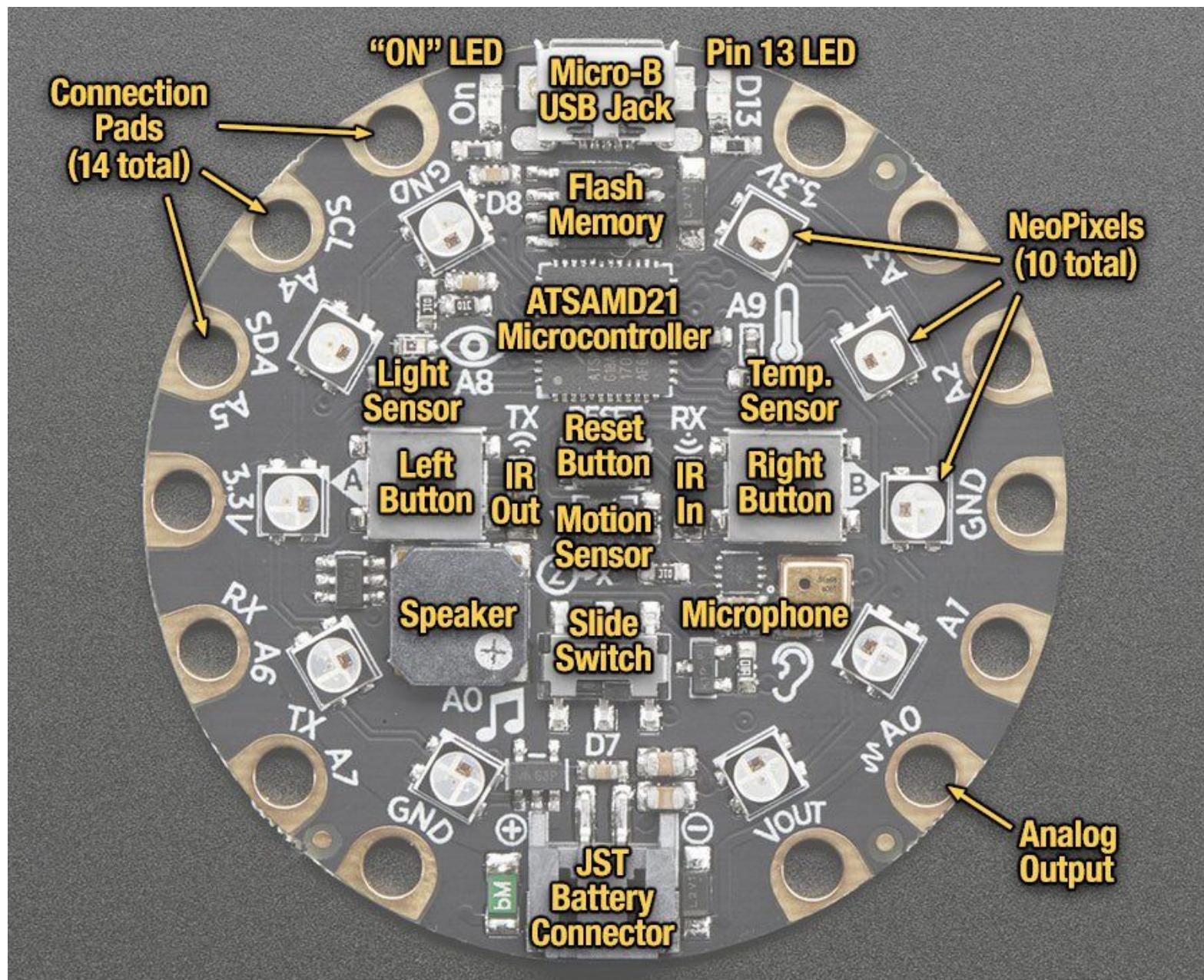


This is an example of the Circuit Playground Express board that we will be using today. The board has many sensor, lights, a switch, and buttons. All of these can be used in the programs we will be creating.



Schedule

8:30 Introductions

8:35 What we will be learning today

- Basic coding terminology
- Loops
- Conditionals/Boolean/Comparison
- Variables
- Math
- Functions

8:50 Connecting to the Guest network

9:00 What is Microsoft Makecode and how does it work

- Opening Microsoft Makecode for Circuit Playground
- Creating a new project
- How to use the project interface
 - Shapes in the project interface and how they fit together
 - Deleting shapes
 - Duplicating shapes
- Loading an existing program into Makecode
- Saving your program to a file
- Loading your program onto your Circuit Playground

9:30 Our first program - Lets make a light blink.

9:40 Demo - Lets make a light blink with a different language using the same logic.

9:45 Break

10:00 Loops - Lets build our first while loop

10:10 Loops with conditionals - If something is true do something while another thing is true.

10:20 Loops with logicals - A different way to do the same thing

10:30 Lets talk about what we just learned

10:35 What can you create using what we have already learned?

10:45 More conditionals - If/then/else logic

11:00 Break

11:10 Hot Potato - Variables, Random Number, Loops, and Luck

11:45 Free time to create with what we have learned so far

12:00 Lunch

1:00 Tilt Sensor - If/Then/Else with hardware input

1:15 Tilt Trumpet - Nested If/then/else conditionals

1:45 Break

2:00 Fireflies - updating a variable based on other inputs

2:30 Free time to create with what we have learned

2:45 Functions

3:00 Zombie Game

3:45 Break

4:00 Show and Tell

Schedule

1

Connecting to the guest network

4

What is Microsoft Makecode

4

Github - Source Code for CircuitPlayground Express

5

Connecting your CircuitPlayground to your Computer

5

Our first program - Lets make a light blink

6

Demo - Lets make a light blink with a different language using the same logic

7

Loops - Lets build our first while loop

8

Loops with conditionals

9

Loops with logicals

10

More conditionals - If/then/else logic

11

Hot Potato - Variables, Random Number, Loops, and Luck

12

Tilt Sensor - If/Then/Else with hardware input

13

Tilt Trumpet - Nested If/then/else conditionals

14

Fireflies - updating a variable based on other inputs

16

Functions - I need to call this more than once

17

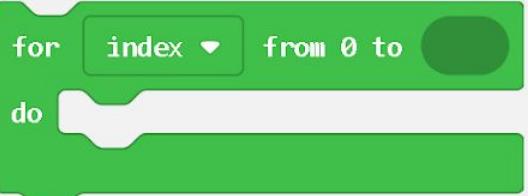
Zombie Game

18

Circuit Playground Links

21

Loops



for

Repeat code for a given number of times using an index.



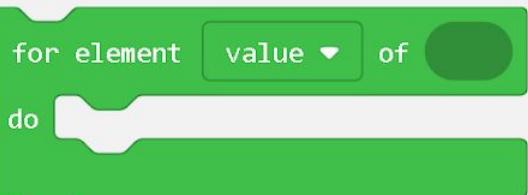
while

Repeat code while a condition is true.



repeat

Repeat code for a given number of times.



for of

Repeat code for each item in a list.



forever

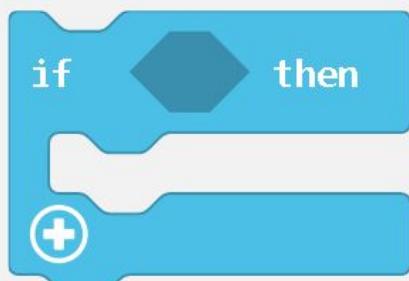
Repeats the code forever in the background.



pause

Pause for the specified time in milliseconds.

Logic



if

Conditional statement.



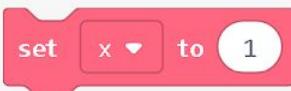
Boolean

True or false values.

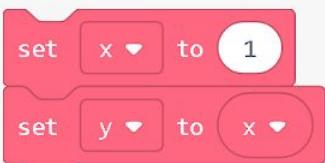
Variables



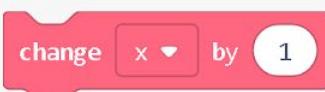
Assign (set) a variable's value



Get a variable's value

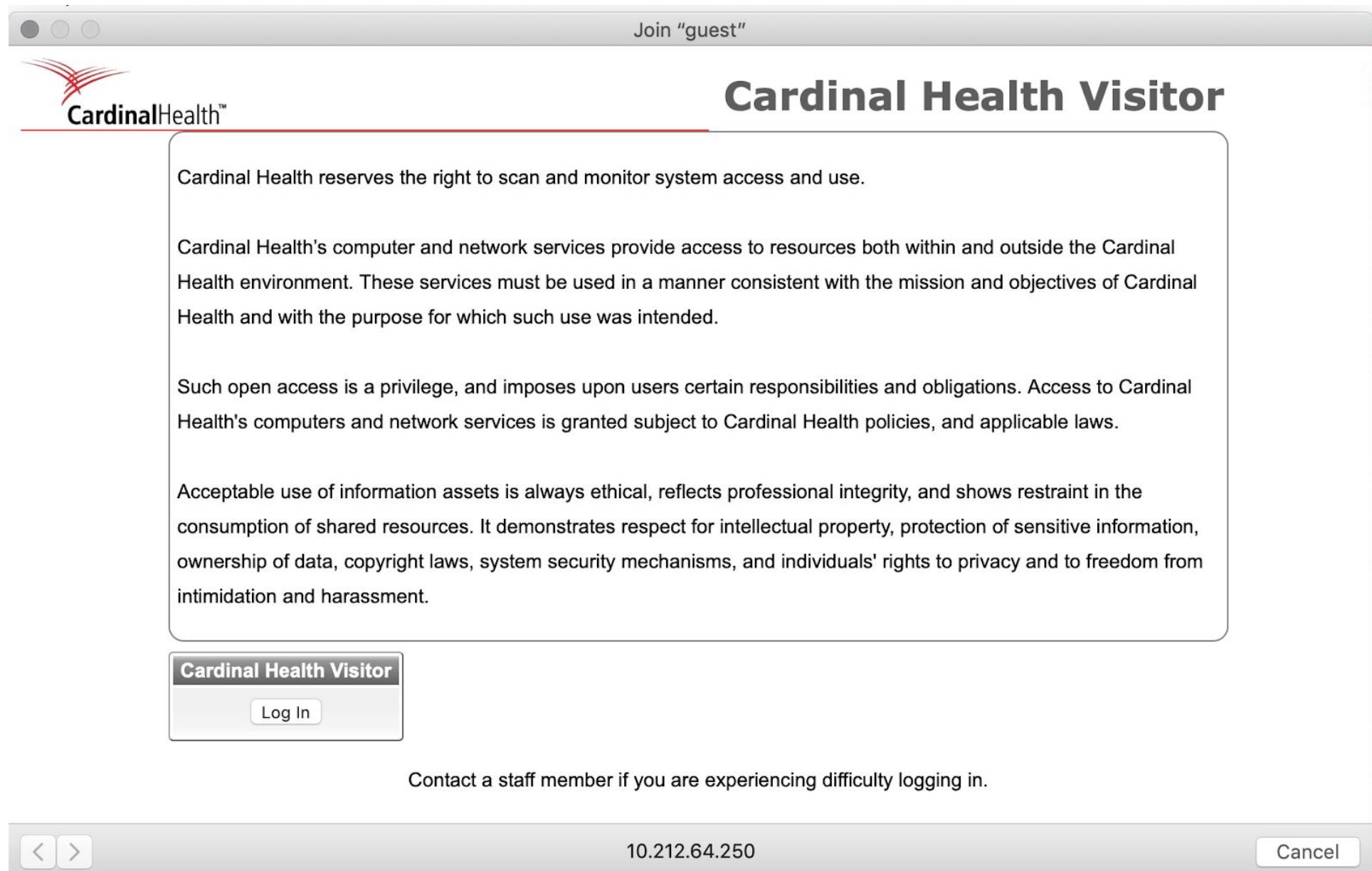


Change a variable's value



Connecting to the guest network

- On your computer click the proper icon for the wireless network
- Choose the guest network
- When it prompts you to accept the terms click the Log In button



What is Microsoft Makecode

- Opening Microsoft Makecode for Circuit Playground
 - <https://makecode.adafruit.com>
- Creating a new project
- How to use the project interface
 - Shapes in the project interface and how they fit together
 - Deleting shapes
 - Duplicating shapes
- Loading an existing program into Makecode
- Saving your program to a file
- Loading your program onto your Circuit Playground

Github - Source Code for CircuitPlayground Express

<https://github.com/fusecodecamp2019/CircuitPlaygroundExpress>

The screenshot shows the GitHub repository page for 'fusecodecamp2019 / CircuitPlaygroundExpress'. At the top, there's a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header, the repository name is displayed along with its statistics: 11 commits, 1 branch, 0 releases, and 1 contributor. A green 'Clone or download' button is prominently featured. The main content area shows a list of files and their commit history. A specific commit by 'cah-johnlusk' is highlighted, showing changes to 'Bootloader', 'CircuitPlayground', 'CircuitPython', 'Instructions', 'README.md', and 'circuit_playground_express_labeled.jpg'. The 'README.md' file is expanded, showing its content: 'CircuitPlaygroundExpress' and 'Circuit Playground Express code for FuseCodeCamp 2019'. At the bottom of the page, there are links for GitHub terms and conditions, and a footer with links to Contact GitHub, Pricing, API, Training, Blog, and About.

Open the URL for Github above and Click on the Green Clone or Download button. Click on the Download Zip option and save the zip file somewhere you can remember such as your Documents folder or your desktop. Go to your File explorer/Finder and browse to where you saved the file. You can right click on the file and extract the folder or for OSX you can double click on the file. This contains the majority of the code that we will be working with today.

Connecting your CircuitPlayground to your Computer

1. Open the box that contains your CircuitPlaygroundExpress Board, MicroUSB Cable, and Battery Pack
2. Plug the microUSB end of the cable into the port on the CircuitPlaygroundExpress -- **Be Careful and don't force it into the port**
3. Plug the other end of the cable into the USB port on your computer

The CircuitPlaygroundExpress should light up and all of the lights on the Neopixels should light up green. If the lights do not light up green press the small button in the center of the board.

Open your File Manager/Finder and you should see a drive named CPLAYGROUNDBOOT.

Our first program - Lets make a light blink

MakeCode - This will make an LED on the board blink when you press the A button. This is an example of conditionals. Conditionals use logic like If/Then/Else and If/Then/Else if/Else. We will cover different combinations of conditionals in many forms today.

Microsoft MakeCode for Circuit Playground Express

HOME SHARE BLOCKS JAVASCRIPT

Search...

LIGHT INPUT MUSIC NETWORK LOOPS LOGIC VARIABLES MATH ADVANCED

Download LED Example

This is the Javascript code which is generated by the MakeCode blocks

```
1 forever(function () {
2   if (input.buttonA.isPressed()) {
3     pins.LED.digitalWrite(true)
4   } else {
5     pins.LED.digitalWrite(false)
6   }
7 })
```

HOME SHARE BLOCKS JAVASCRIPT

Search...

LIGHT INPUT MUSIC NETWORK LOOPS LOGIC VARIABLES MATH ADVANCED

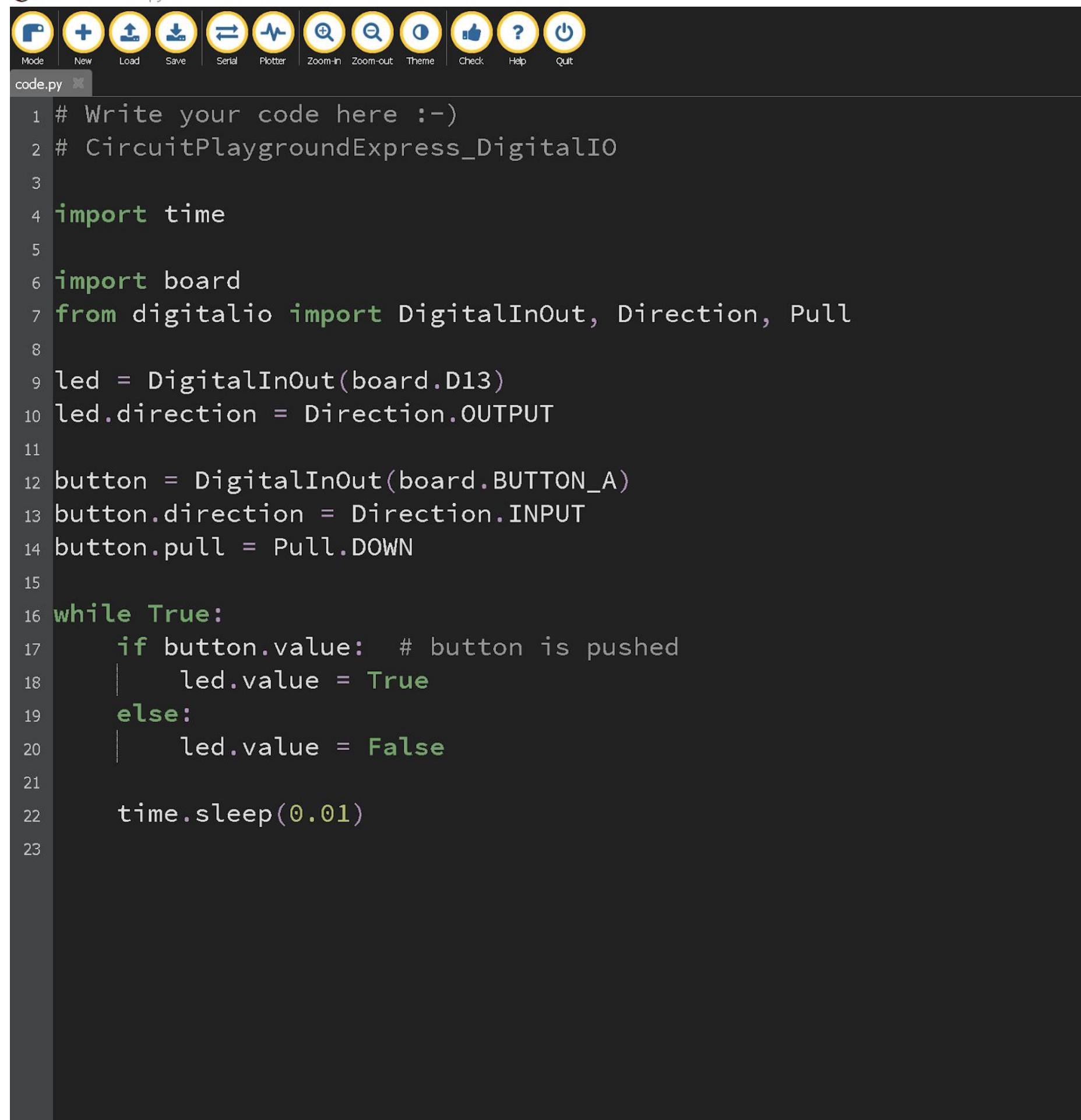
EXPLORER >

Download LED Example

Demo - Lets make a light blink with a different language using the same logic

This is CircuitPython code. It is python code which will perform the same function and uses the same logic.

¶ Mu 1.0.2 - code.py



The screenshot shows the Mu code editor interface. At the top, there is a toolbar with various icons: Mode, New, Load, Save, Serial, Plotter, Zoom-in, Zoom-out, Theme, Check, Help, and Quit. Below the toolbar, the file name "code.py" is displayed. The main area contains the following Python code:

```
1 # Write your code here :-)
2 # CircuitPlaygroundExpress_DigitalIO
3
4 import time
5
6 import board
7 from digitalio import DigitalInOut, Direction, Pull
8
9 led = DigitalInOut(board.D13)
10 led.direction = Direction.OUTPUT
11
12 button = DigitalInOut(board.BUTTON_A)
13 button.direction = Direction.INPUT
14 button.pull = Pull.DOWN
15
16 while True:
17     if button.value: # button is pushed
18         led.value = True
19     else:
20         led.value = False
21
22     time.sleep(0.01)
23
```

Break Time - 10 Minutes

Loops - Lets build our first while loop

This is an example of a loop. It is a simple while loop. The code inside the while loop will continue to execute **WHILE** the condition is true. When the condition is not true any longer the loop will exit.

The screenshot shows the Microsoft MakeCode interface for the Circuit Playground Express. On the left, there's a circular diagram of the board with pins labeled A through D and various sensors and connectors. The main workspace on the right displays a Scratch-style script:

```
forever
  while [button A is pressed]
    do
      show animation [rainbow] v
      for [500 ms]
    end
  end
  clear
```

The script consists of a green "forever" loop. Inside it, there's a pink "while" loop condition that checks if "button A" is pressed. If true, it enters a blue "do" loop which contains a green "show animation" block with the "rainbow" option, and a blue "for" loop set to 500ms. After the inner loops complete, the script ends with a blue "clear" block. On the far left, there are several small icons for file operations like back, forward, and save. At the bottom, there are buttons for "Download" (in a pink box) and "Loop Example".

This is the javascript that is generated for the example above.

The screenshot shows the Microsoft MakeCode interface again, but this time the generated JavaScript code is visible in the center workspace:

```
1 forever(function () {
2   while (input.buttonA.isPressed()) {
3     light.showAnimation(light.rainbowAnimation, 500)
4   }
5   light.clear()
6 })
7
```

The code is a direct translation of the Scratch script into JavaScript. It uses the `forever` function from the `input` and `light` objects. The `while` loop condition is `input.buttonA.isPressed()`. Inside the loop, it calls `light.showAnimation` with the `rainbow` animation and a duration of 500ms. After the loop condition is false, it calls `light.clear()`.

Loops with conditionals

This is another example of the same while loop but it is now nested inside a conditional block. This conditional will execute the code that is inside the block if the condition is true.

The screenshot shows the Microsoft MakeCode interface for the Circuit Playground Express. On the left, there's a circular pinout diagram of the board. The top navigation bar includes 'Microsoft MakeCode for Circuit Playground Express', 'HOME', 'SHARE', 'BLOCKS' (selected), and 'JAVASCRIPT'. A search bar says 'Search...'. Below the search bar is a category sidebar with 'LIGHT', 'INPUT', 'MUSIC', 'NETWORK', 'LOOPS' (selected), 'LOGIC', 'VARIABLES', 'MATH', and 'ADVANCED'. The main workspace contains a script starting with a 'forever' loop. Inside the 'forever' loop is an 'if' block that checks if 'button B' is pressed. If true, it enters a 'while' loop that checks if 'button A' is pressed. If true, it runs a 'show animation' block for 500ms. Finally, it runs a 'clear' block. The bottom of the screen features a pink 'Download' button and a blue 'Loop with conditional exam' button.

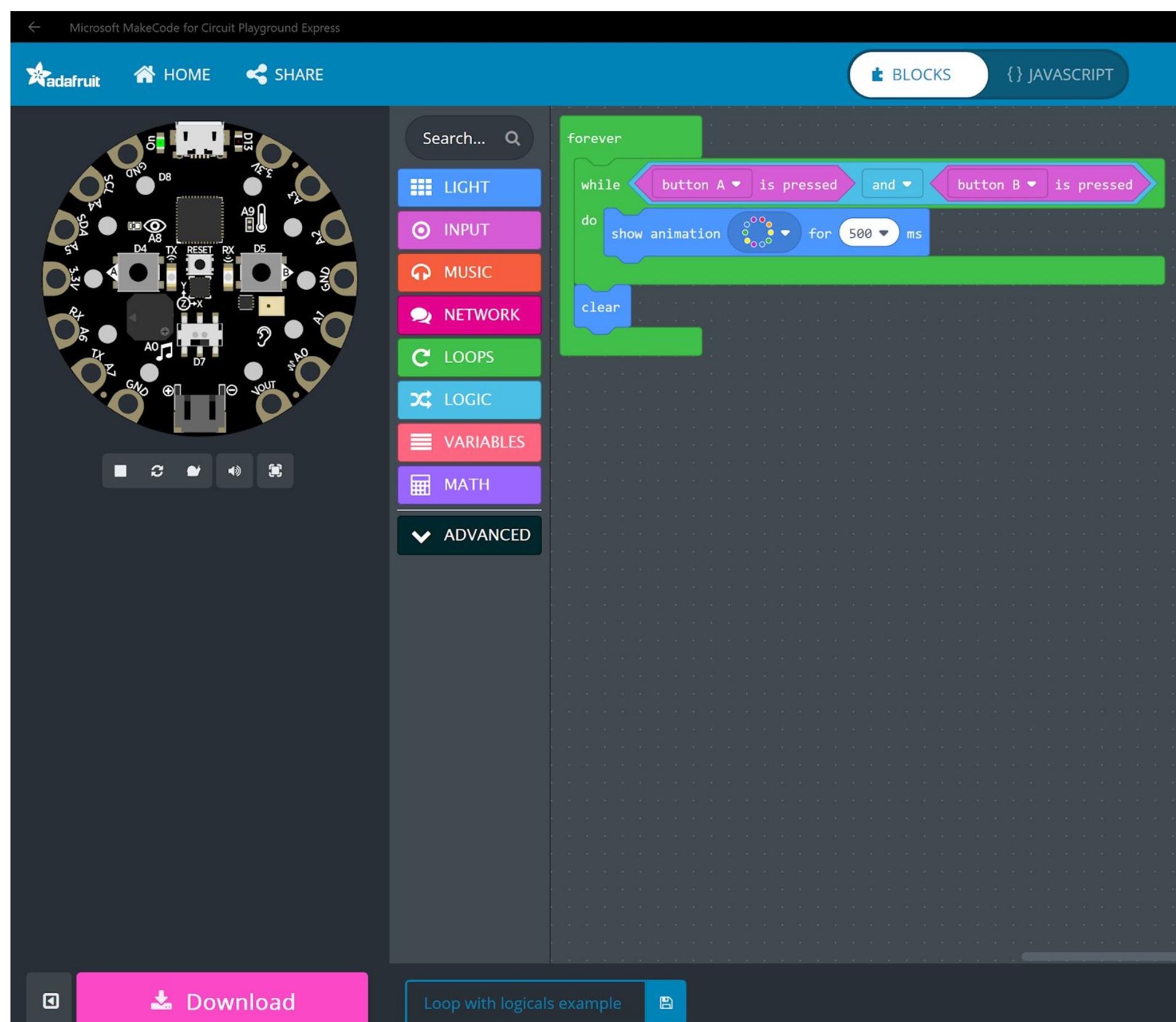
The screenshot shows the Microsoft MakeCode interface for the Circuit Playground Express, similar to the one above but with different content. The pinout diagram is on the left. The top navigation bar includes 'Microsoft MakeCode for Circuit Playground Express', 'HOME', 'SHARE', 'BLOCKS' (selected), and 'JAVASCRIPT'. A search bar says 'Search...'. Below the search bar is a category sidebar with 'LIGHT', 'INPUT', 'MUSIC', 'NETWORK', 'LOOPS' (selected), 'LOGIC', 'VARIABLES', 'MATH', and 'ADVANCED'. The main workspace displays the generated JavaScript code:

```
1 forever(function () {
2     if (input.buttonB.isPressed()) {
3         while (input.buttonA.isPressed()) {
4             light.showAnimation(light.rainbowAnimation, 500)
5         }
6         light.clear()
7     }
8 })
```

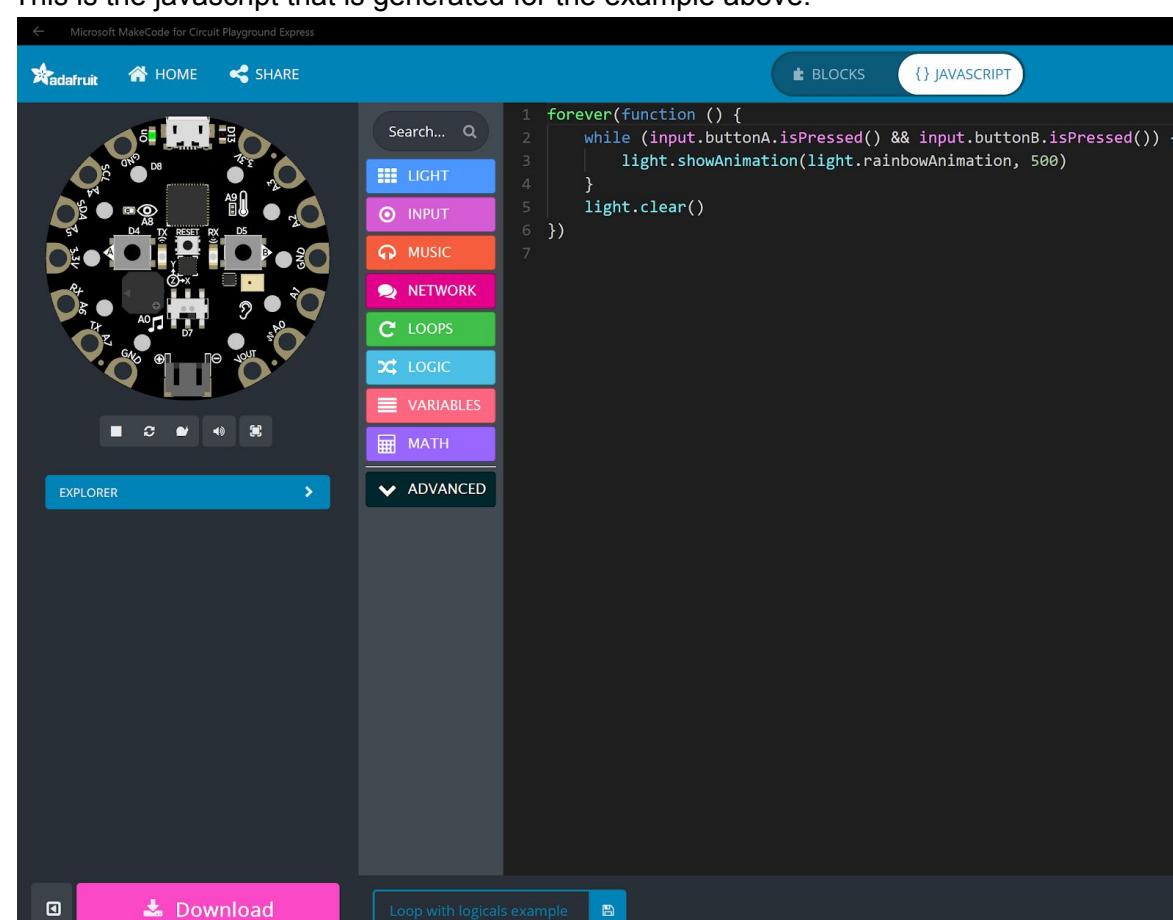
The bottom of the screen features a pink 'Download' button and a blue 'Loop with conditional exam' button.

Loops with logicals

This is an example of a while loop using logicals. Logicals or logical operators are comparisons such as AND/OR/NOT. In this case two conditions must be met in order for the while to be true and to execute the code.



This is the javascript that is generated for the example above.



More conditionals - If/then/else logic

This example shows the expanded use of conditionals and more specifically the if/then/else condition. We also use the random number function to pick a random number and use that number to determine which of our conditions to run. Let's build this program and walk through the logic together.

The Scratch 3.0 interface is shown with a micro:bit board on the left. The script area contains the following code:

```
forever
  on shake
    set [rand v] to [pick random 0 to 4]
    if [rand v] = [0] then
      show animation [running lights v] for [2000] ms
    else if [rand v] = [1] then
      show animation [rainbow v] for [2000] ms
    else if [rand v] = [2] then
      show animation [comet v] for [2000] ms
    else
      show animation [theater chase v] for [2000] ms
  end
  clear
```

Below the script are two buttons: "Download" and "Random Number and If-The".

The Scratch 3.0 interface is shown with the "JAVASCRIPT" tab selected. The script area contains the following code:

```
let rand = 0
input.onGesture(Gesture.Shake, function () {
  rand = Math.randomRange(0, 4)
  if (rand == 0) {
    light.showAnimation(light.runningLightsAnimation, 2000)
  } else if (rand == 1) {
    light.showAnimation(light.rainbowAnimation, 2000)
  } else if (rand == 2) {
    light.showAnimation(light.cometAnimation, 2000)
  } else {
    light.showAnimation(light.theaterChaseAnimation, 2000)
  }
  light.clear()
})
forever(function () {})
```

Below the script are two buttons: "Download" and "Random Number and If-The".

Break Time - 10 Minutes

Hot Potato - Variables, Random Number, Loops, and Luck

This is a game of hot potato. After we create this program and follow through the logic together we will play a few rounds of the game. This program uses inputs, variables, the random number function, and loops. As we play the game we will modify some of the parameters of the program to see how it affects the program.

The screenshot shows the Scratch 3.0 interface. On the left, there's a circular preview of the micro:bit board with pins labeled D0-D13, A0-A7, GND, VOUT, TX, RX, and RESET. Below the preview are several control buttons. To the right of the preview is a sidebar with categories: LIGHT, INPUT, MUSIC, NETWORK, LOOPS, LOGIC, VARIABLES, MATH, and ADVANCED (which is currently selected). The main workspace contains a script for the micro:bit. It starts with a green `forever` loop. Inside, it sets the volume to 100. It then waits for a button A click, setting a delay variable to 500 plus a random value from 0 to 1501. A `while` loop runs as long as the delay is greater than 0. Inside the loop, it plays a tone at Middle C for a quarter beat, shows a rainbow animation for the delay duration, changes the delay by -50, plays a sound effect, and shows a theater chase animation for 2500ms. The bottom of the screen shows a pink "Download" button and a project name "HotPotato".

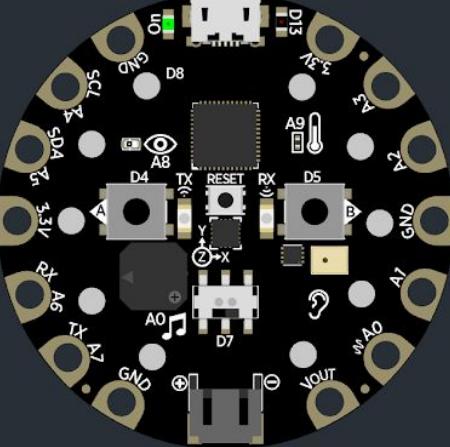
The screenshot shows the Scratch 3.0 interface with the generated JavaScript code for the Hot Potato script. The code is as follows:

```
let delay = 0
input.buttonA.onEvent(ButtonEvent.Click, function () {
  delay = 500 + Math.randomRange(0, 1501)
  while (delay > 0) {
    music.playTone(262, music.beat(BeatFraction.Quarter))
    light.showAnimation(light.rainbowAnimation, delay)
    delay += -50
  }
  music.playSound(music.sounds(Sounds.Wawawawaa))
  light.showAnimation(light.theaterChaseAnimation, 2500)
})
forever(function () {
  music.setVolume(100)
})
```

The interface includes the same circular preview of the micro:bit board, sidebar categories, and project controls as the top screenshot. The bottom features a pink "Download" button and a "HotPotato" tab.

Tilt Sensor - If/Then/Else with hardware input

For this exercise we will see if you can follow along with the logic of a program using some of the concepts that we have learned but this time I am not going to give you the block code first. Does anyone think they can tell me what this code will do if we load it on the Circuit Playground?



The screenshot shows the Adafruit Circuit Playground Express development board. The board has a circular layout with various pins labeled: D13, D12, D11, D10, D9, D8, A7, A6, A5, A4, A3, A2, GND, VOUT, A0, A1, TX, RX, A8, A9, D5, D4, D3, D2, D1, D0, and GND. There are also several small components like resistors and capacitors. The board is connected to a computer via USB.

The interface includes a top navigation bar with 'adafruit' logo, 'HOME', 'SHARE', 'BLOCKS' (selected), and 'JAVASCRIPT'. On the left, there's a 'Search...' field and a sidebar with categories: LIGHT (blue), INPUT (purple), MUSIC (orange), NETWORK (pink), LOOPS (green), LOGIC (light blue), VARIABLES (light purple), and MATH (dark purple). Below the sidebar is a 'EXPLORER' button. In the center, the code editor displays the following JavaScript code:

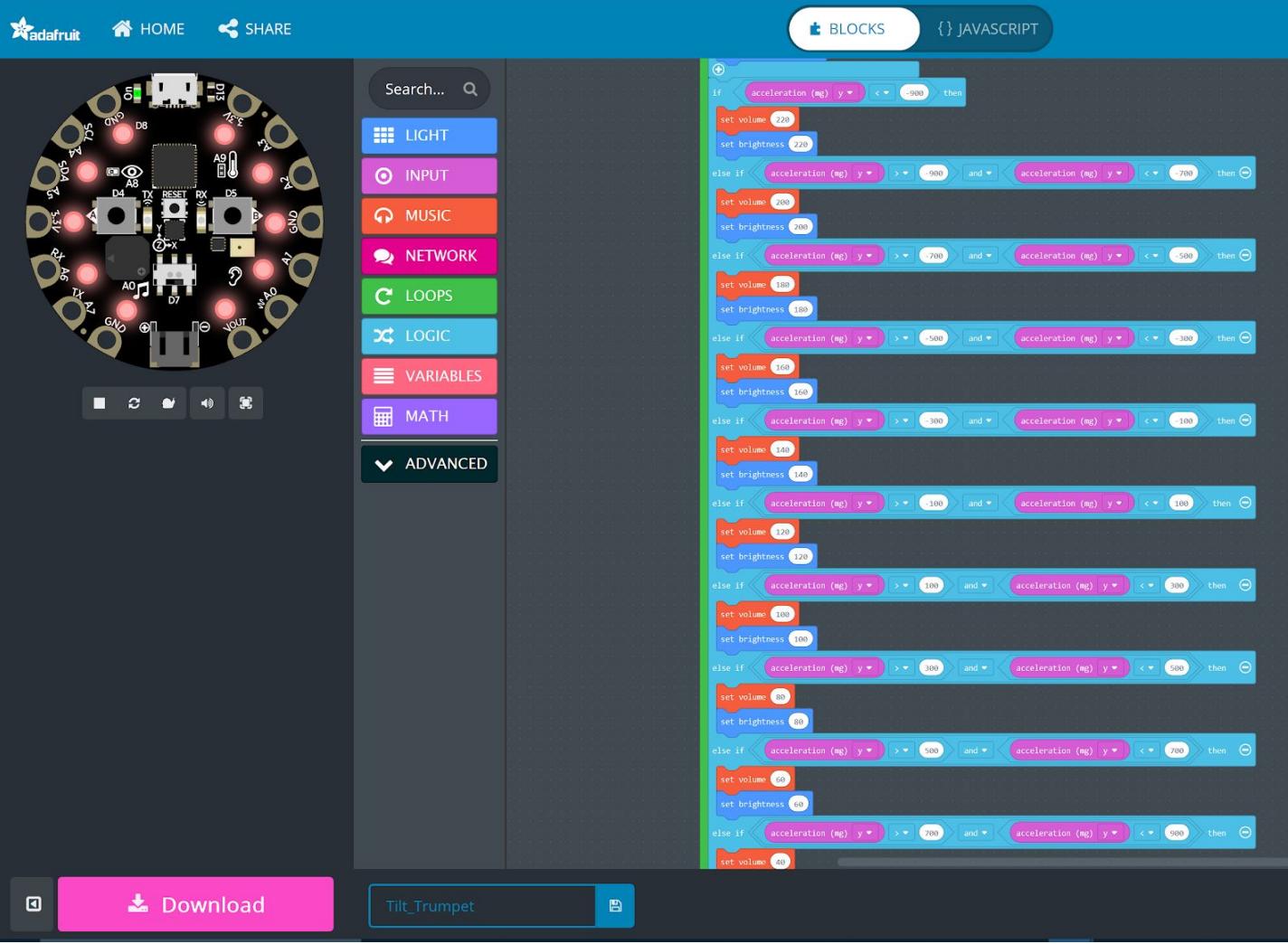
```
1 forever(function () {
2   if (input.acceleration(Dimension.Y) >= 500) {
3     light.showRing(
4       `red black black black black black black black red` 
5     )
6   } else if (input.acceleration(Dimension.Y) <= -500) {
7     light.showRing(
8       `black black black black red red black black black` 
9     )
10 } else if (input.acceleration(Dimension.X) >= 500) {
11   light.showRing(
12     `black red red red black black black black black` 
13   )
14 } else if (input.acceleration(Dimension.X) <= -500) {
15   light.showRing(
16     `black black black black black black red red red black` 
17   )
18 } else {
19   light.clear()
20 }
21 })
22
```

At the bottom, there are 'Download' and 'Tilt Sensor' buttons.

Tilt Trumpet - Nested If/then/else conditionals

This program is one of the examples that Adafruit makes available on their website. This is an example of what is commonly called a nested if. That means that we have two logical if blocks inside of each other. If the first condition is true then the second if block is evaluated and all of the else/if conditions are evaluated as part of that if block. Don't worry we are not going to make you create this by hand. We will load this program from a file and follow along with the logic. Once we are done then we can load the program onto the circuit playground and make some music.

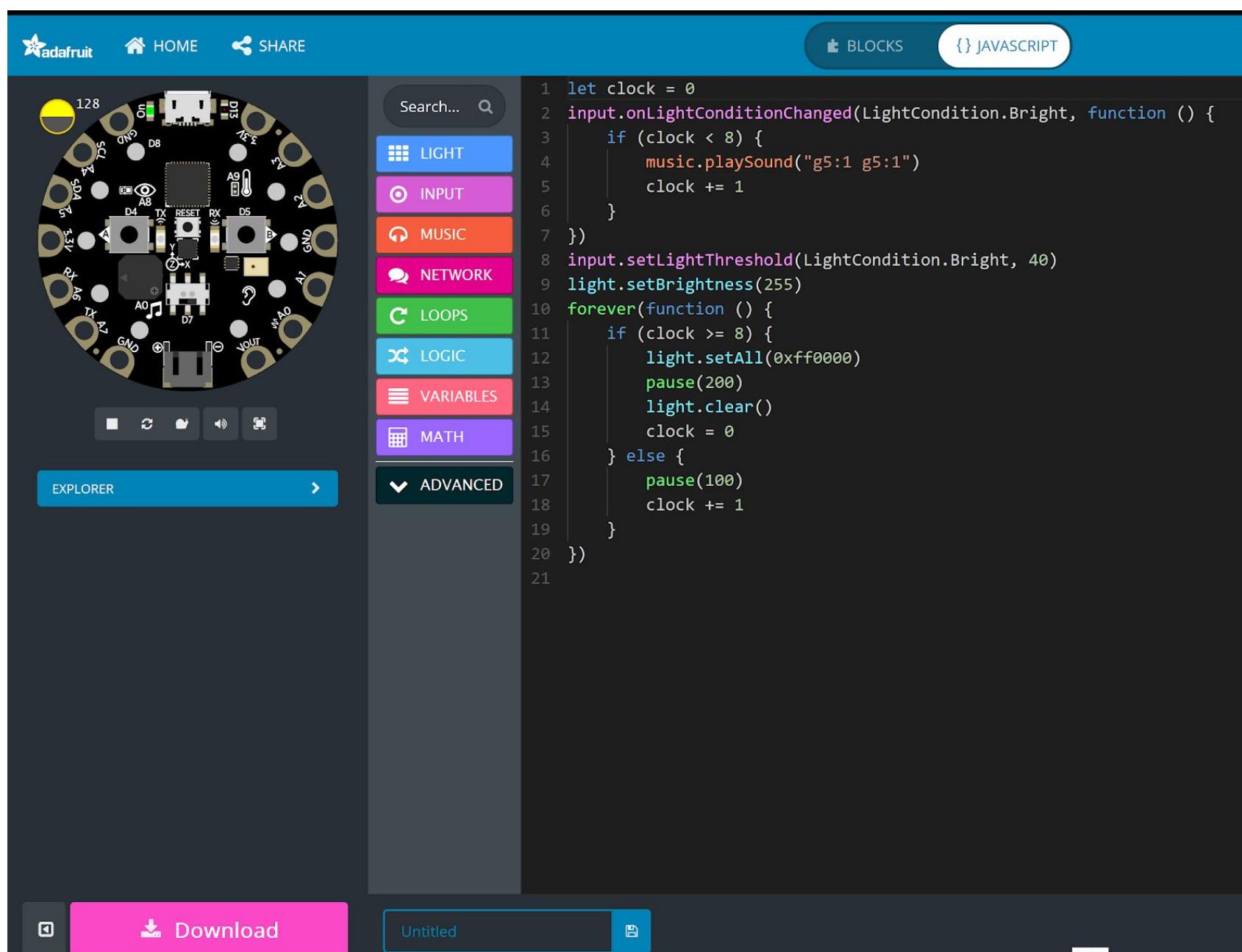
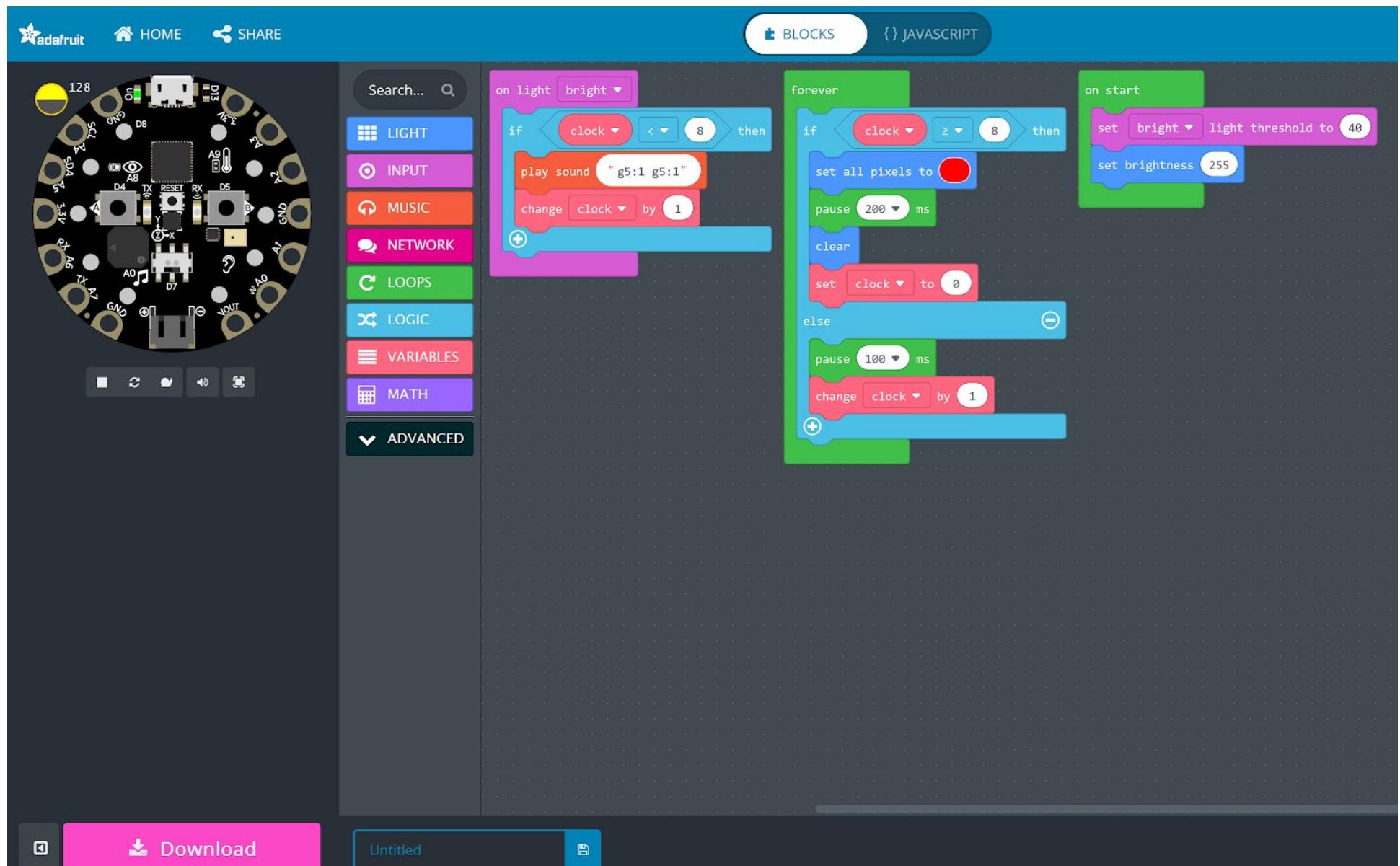
The screenshot shows the Adafruit Circuit Playground Express code editor interface. On the left, there's a circular diagram of the board with pins labeled A0 through A9, D0 through D7, and various ground (GND) and power (VOUT) pins. The right side contains the Scratch-style code blocks for the 'Tilt Trumpet' project. The code is organized into a 'forever' loop. It starts with an 'if' block for button A being pressed. Inside this 'if' block, there are multiple nested 'if' blocks based on the X-axis acceleration values. These nested blocks determine which musical note to play ('low E', 'low B', etc.) and how long to play it ('1/8 beat'). Between the nested 'if' blocks, there are 'set all pixels to [red]' blocks. After the innermost 'if' block, there's another 'if' block for button B being pressed. Inside this 'if' block, there are more nested 'if' blocks for different X-axis acceleration ranges, leading to notes like 'middle C' and 'high C'. There are also 'set all pixels to [red]' blocks here. Finally, there's an 'else' block for button A and an 'else' block for button B, both of which set all pixels to red. At the bottom of the code editor, there are 'Download' and 'Tilt_Trumpet' buttons.



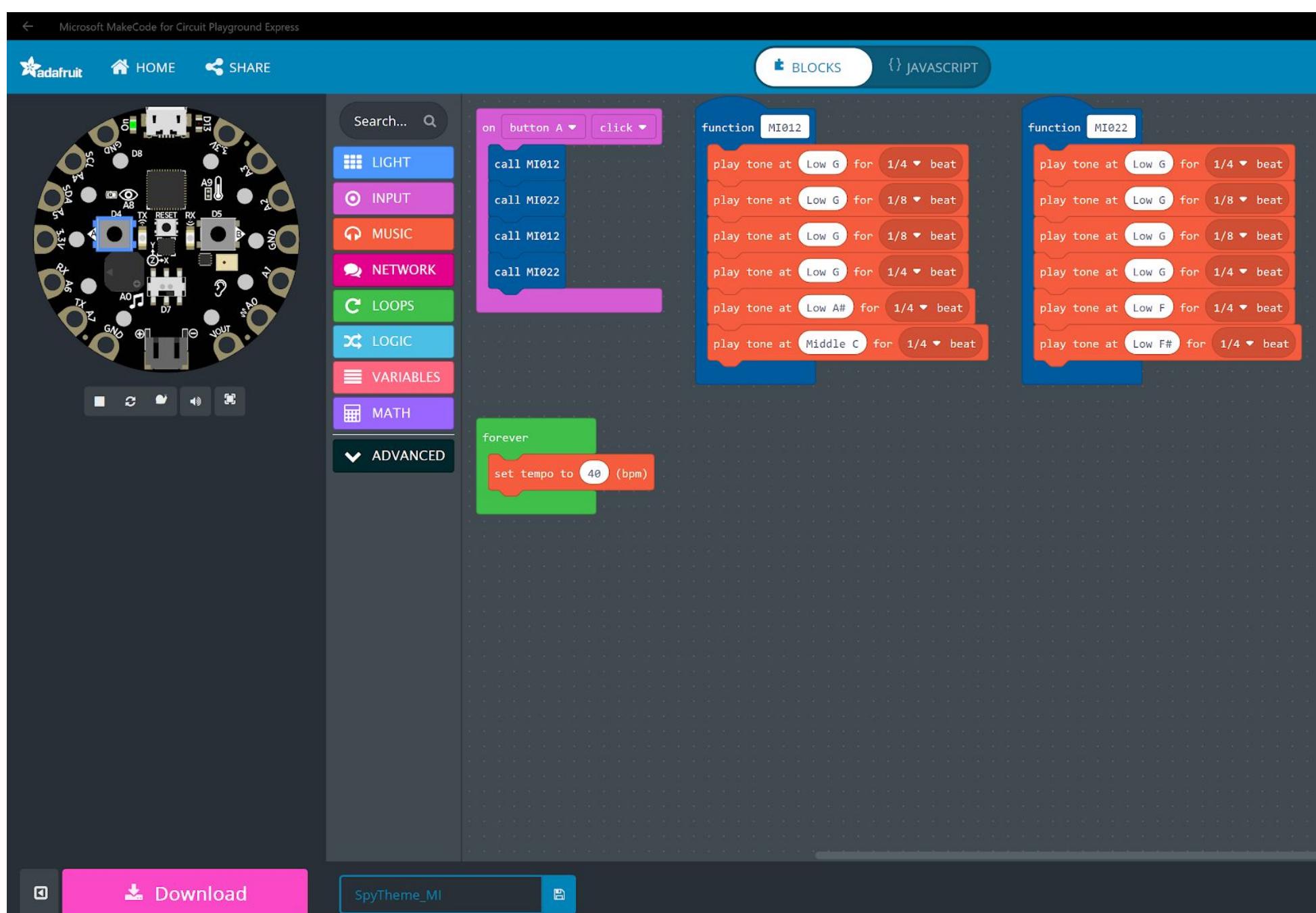
Break Time - 10 Minutes

Fireflies - updating a variable based on other inputs

This example is called fireflies and will be interactive and will involve the entire group. In nature if you watch fireflies they will blink and after a while you will notice that they are all blinking at the same rate at the same time. This example will use the lights and the light sensor to slowly adjust a clock of when the "firefly" will blink. After enough time all of your "fireflies" should be blinking at the same time.



Functions - I need to call this more than once



```
1 input.buttonA.onEvent(ButtonEvent.Click, function () {
2   MI012()
3   MI022()
4   MI012()
5   MI022()
6 }
7 function MI012 () {
8   music.playTone(196, music.beat(BeatFraction.Quarter))
9   music.playTone(196, music.beat(BeatFraction.Eighth))
10  music.playTone(196, music.beat(BeatFraction.Eighth))
11  music.playTone(196, music.beat(BeatFraction.Quarter))
12  music.playTone(233, music.beat(BeatFraction.Quarter))
13  music.playTone(262, music.beat(BeatFraction.Quarter))
14 }
15 function MI022 () {
16   music.playTone(196, music.beat(BeatFraction.Quarter))
17   music.playTone(196, music.beat(BeatFraction.Eighth))
18   music.playTone(196, music.beat(BeatFraction.Eighth))
19   music.playTone(196, music.beat(BeatFraction.Quarter))
20   music.playTone(175, music.beat(BeatFraction.Quarter))
21   music.playTone(185, music.beat(BeatFraction.Quarter))
22 }
23 loops.forever(function () {
24   music.setTempo(40)
25 })
26 }
```

Zombie Game

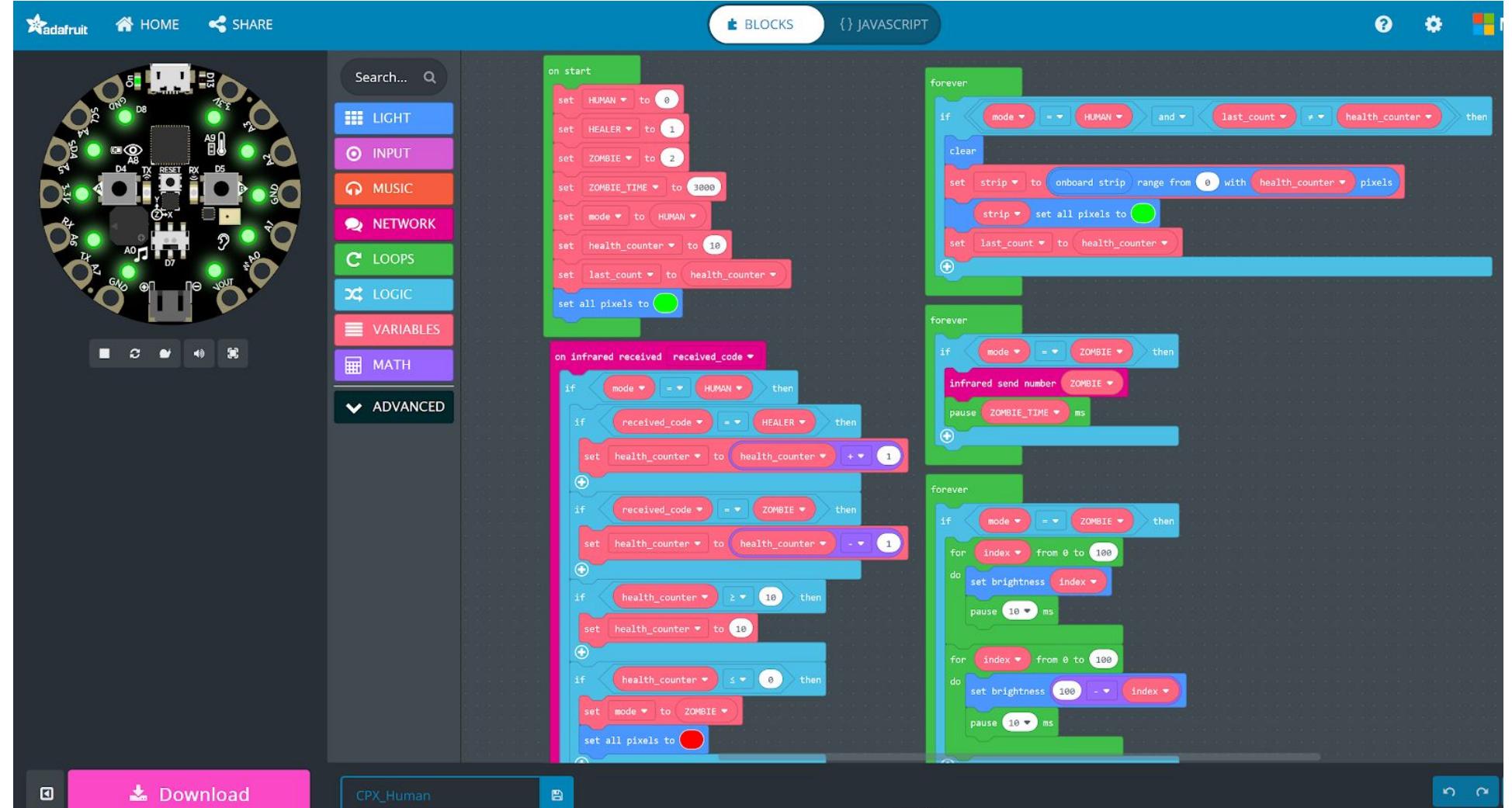
This is a zombie game which is actually three separate programs. The most complicated program is the human. The human has a health which starts at 10. The Human program is always listening for IR signals. The zombie and healer programs are always sending IR signals. The zombie will take away health and the healer will add health. If a human loses all of their health they will become a zombie and will start to send IR signals which will take health away from other humans.

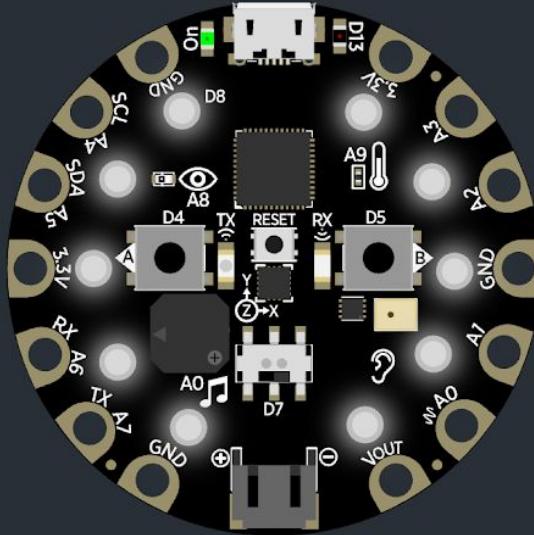
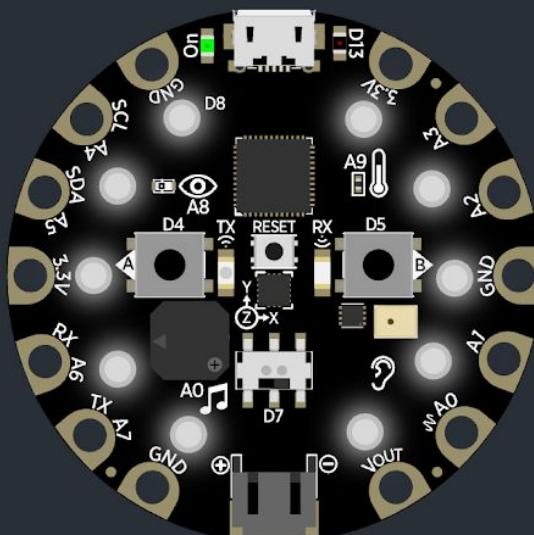
How could we make these programs better?

What could we do to make this one program but still have humans, zombies, and healers?

What concepts that we learned today could be used to combine all three programs into one?

Human



Search... 

LIGHT

INPUT

MUSIC

NETWORK

LOOPS

LOGIC

VARIABLES

MATH

ADVANCED

on start

set HEALER to 1
set HEALER_TIME to 1000
set all pixels to

forever

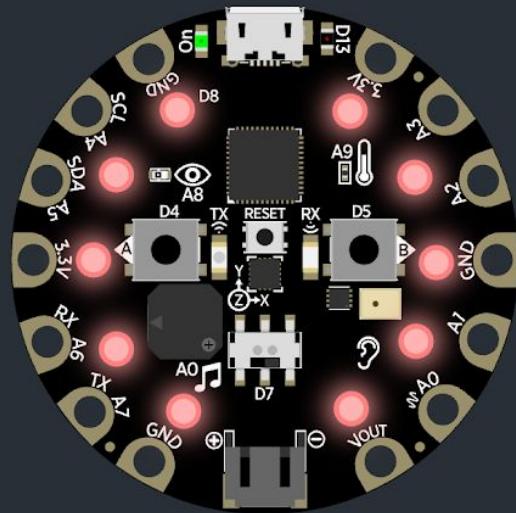
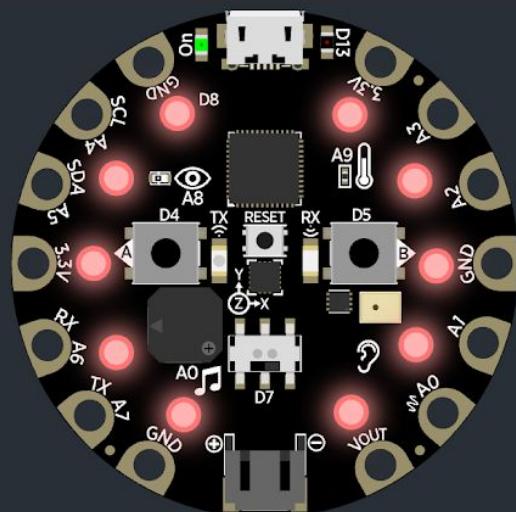
infrared send number HEALER
pause HEALER_TIME ms

forever

for index from 0 to 100
do set brightness index
pause 10 ms
for index from 0 to 100
do set brightness 100 - index
pause 10 ms Download

CPX_Healer



Search... [LIGHT](#)[INPUT](#)[MUSIC](#)[NETWORK](#)[LOOPS](#)[LOGIC](#)[VARIABLES](#)[MATH](#)[ADVANCED](#)

on start

set ZOMBIE ▾ to 2

set ZOMBIE_TIME ▾ to 3000

set all pixels to red

forever

infrared send number ZOMBIE ▾

pause ZOMBIE_TIME ▾ ms

forever

for index ▾ from 0 to 100

do set brightness index ▾

pause 5 ▾ ms

for index ▾ from 0 to 100

do set brightness 100 - index ▾

pause 5 ▾ ms

 Download

CPX_Zombie

Circuit Playground Links

MakeCode Site

<https://makecode.adafruit.com/>

MakeCode USB connection Site

<https://makecode.adafruit.com/?webusb=1#editor>

MakeCode Reference for the Circuit Playground

<https://makecode.adafruit.com/reference>