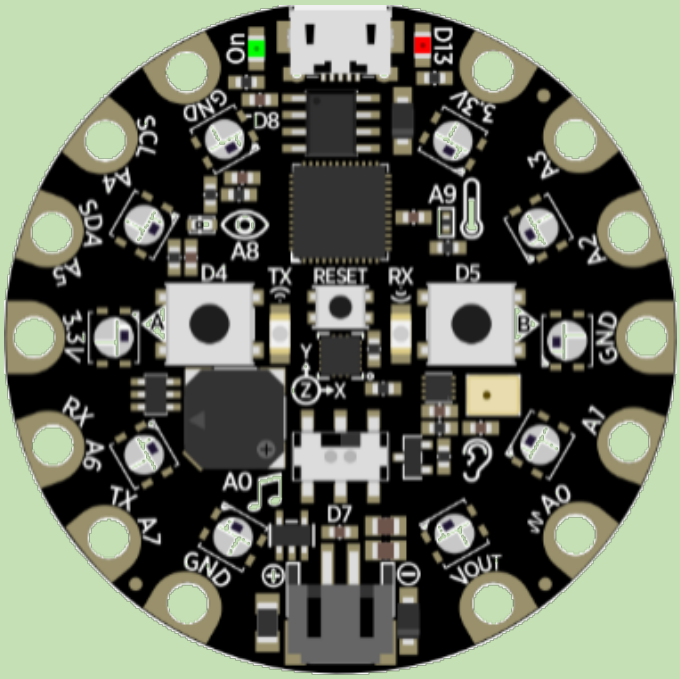
Digital Technology

CPX / MakeCode

Code and Challenges



Version 1
2021

Barry Butler
bbutl58@eq.edu.au

ROBOCOAST
Sunshine Coast Robotics
www.robocoast.tech

Coolum State High School
Care • Respect • Excellence

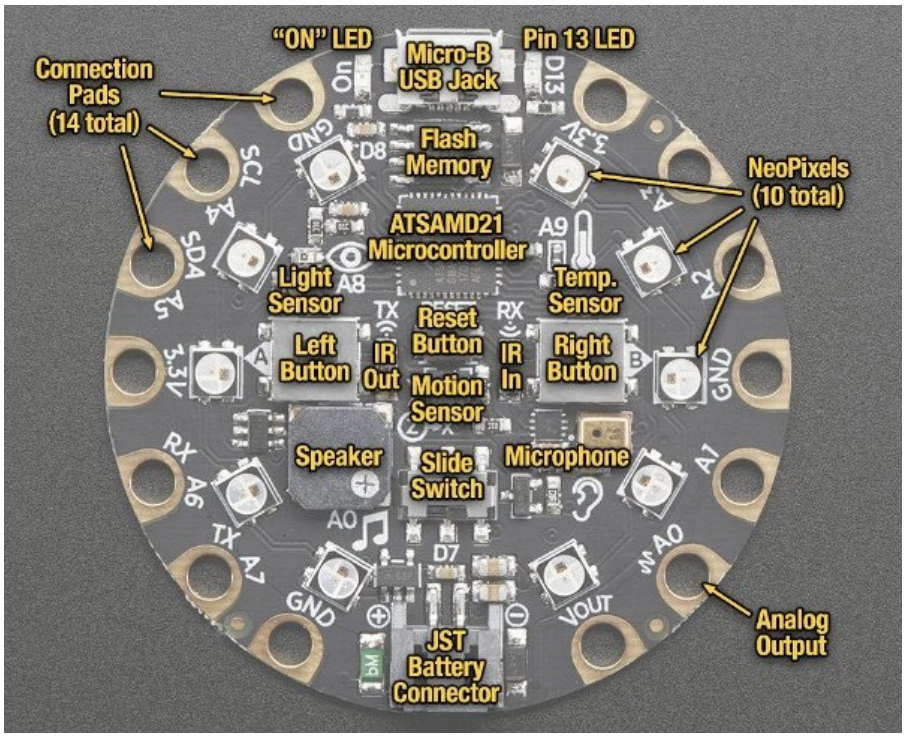
Content and Challenges

Section	Content	Challenges
A	Introduction and First Program	
B	CPX Buttons and Accelerometer	
C	Loops and Variables	
D	Sensors and Conditional Statements	
E	Connecting External Devices	
F	Connect a Potentiometer (Analog Sensor)	
G	Connect a Soil Moisture Sensor (Analog Sensor)	
H	Connect an Ultrasonic Sensor	
I	Connect Servos	
J	Connect Digital Sensors (e.g. Tracker, Collision Switch, Tilt Switch)	
K	Connect Motors	
L	Obstacle Avoidance/Line Follower Servo Vehicle	
M	Propeller Vehicle with Steering	

A. Introduction and First Program

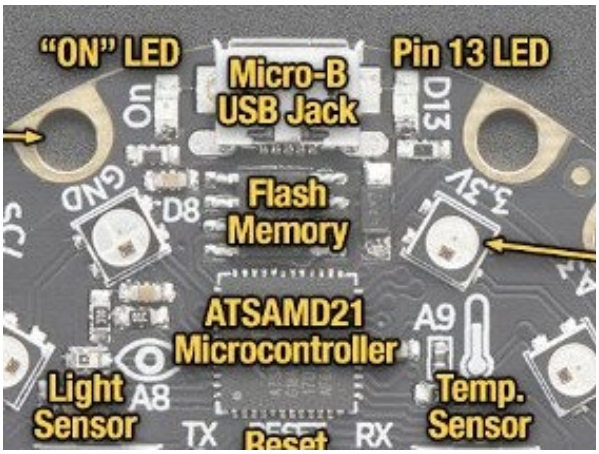
What Has It Got?

The CPX has many sensors (inputs); outputs such as lights and sound; and drive servos, motors and other devices.



Plug It In

Connect the USB cable to the CPX by pushing it straight in the micro-B USB jack.



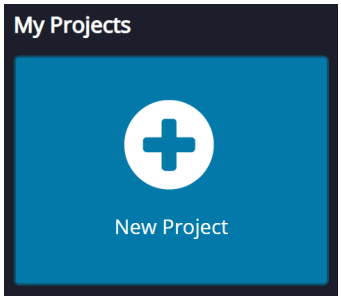
DO NOT wiggle the USB cable as you put it in or take it out. Doing this will break the connector

Start Microsoft MakeCode

In your browser type:

https://makecode.adafruit.com/

Click **New Project** to enter the coding window

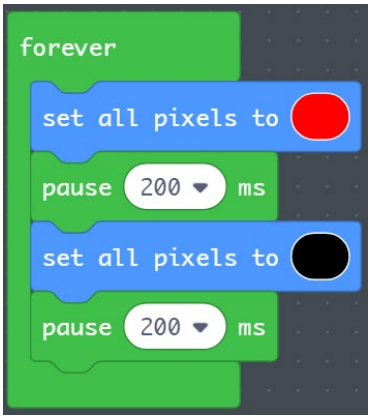


Your First Program

The first program we will write is to blink all the pixels on and off.

Use the Cheat Sheet to find code blocks.

The pause block is found in the Loops section.



Download and Save to the CPX

1. Plug in the Circuit Playground Express.
2. **Press the reset button** (in the middle) to go into BOOT mode. All the neopixels will turn green and the red led will pulse. This allows a new program to be downloaded.
3. In MakeCode, click the pink **Download** button at the bottom left.
4. Close the message that is displayed.
5. In Finder (on a Mac) or the File Manager (in Windows) you should see the CPX show as **CPLAYBOOT**.
6. **Copy the downloaded file** (with an extension uf2, from the Downloads folder) **to the CPLAYBOOT folder**.
7. After copying, your code will run on the CPX.

Alter Your Program

Look on the Cheat Sheet at the Light and Sound sections. Select other blocks to add into your program. You could:

- set the light brightness
- show an animation
- show the light ring and change colours of each neopixel
- play a tone
- play a sound
- set the sound volume

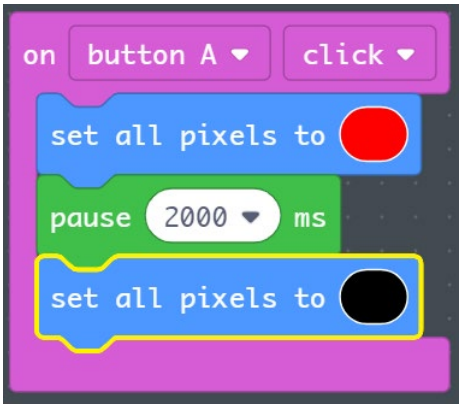
When you have finished, click download again. Copy the new uf2 file to the CPX (CPLAYBOOT).

B. CPX Buttons and Accelerometer

Use a Button to Turn On Lights

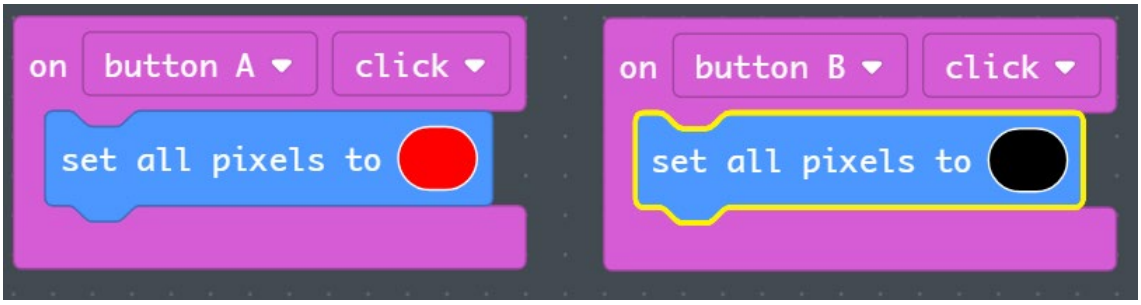
Instead of the lights being turned on when code is downloaded, let's turn on the lights when we press a button. Remove the <forever> loop, click the INPUT button and select the <on button click> block.

Try both of these



Use Buttons to Turn On and Off

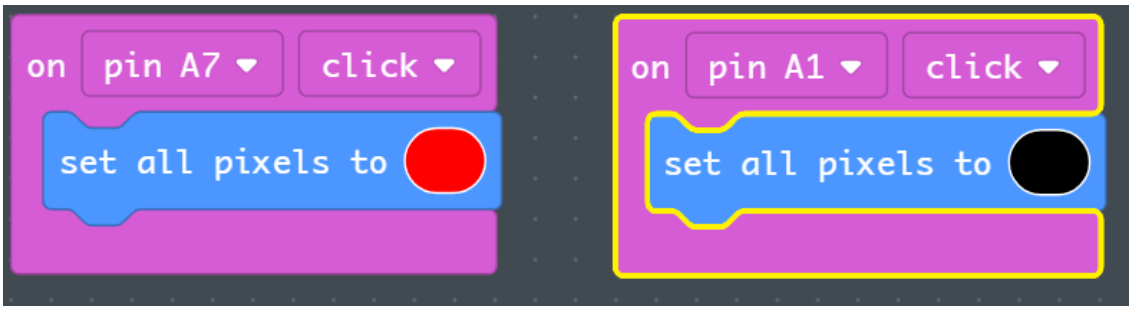
Try using the left button (A) to turn the light on, and the right button (B) to turn the light off.



Add other blocks to run when the buttons are pressed, such as animations and sound.

Use the Connection Pads as Switches

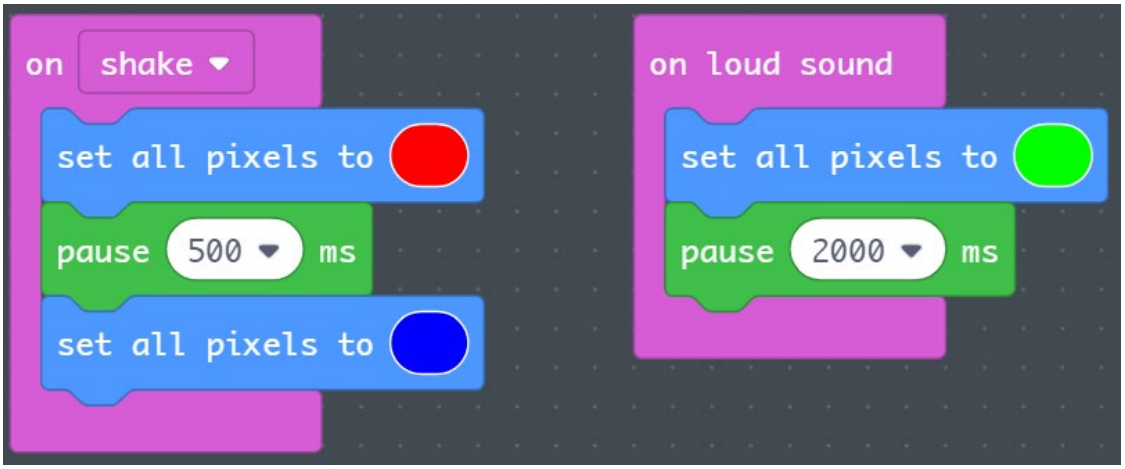
The CPX connection pads act as capacitance touch pads, that is, they respond to being touched. This code uses pin A7 to turn the light on when it is touched and A1 to turn it off.



Use the connection pads to play sounds. See if you can play a simple tune.

Shake It or Clap to Turn Lights On

We can use the motion sensor (accelerometer) or microphone to turn on the lights.

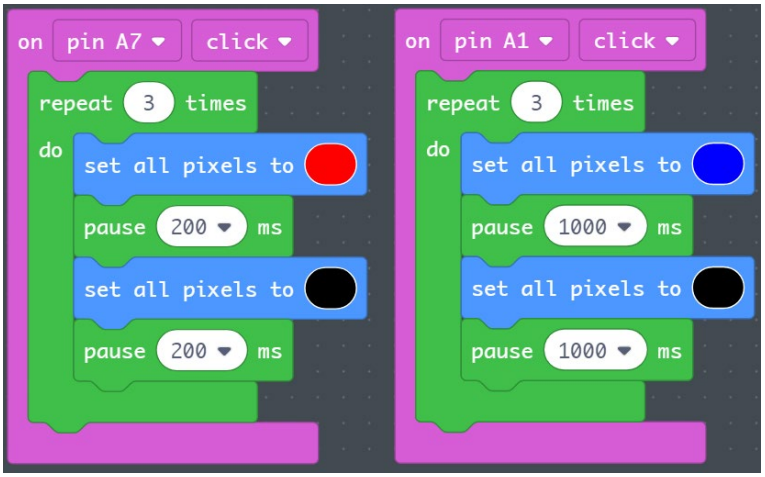


Add light animations and sound.

C. Loops and Variables

Use a Repeat Loop to Repeat Actions

There are several types of loops that can be used for repetitive actions. The first is the **Repeat** loop. Use this to make the lights blink a fixed number of times depending on the button or touch pad pressed.

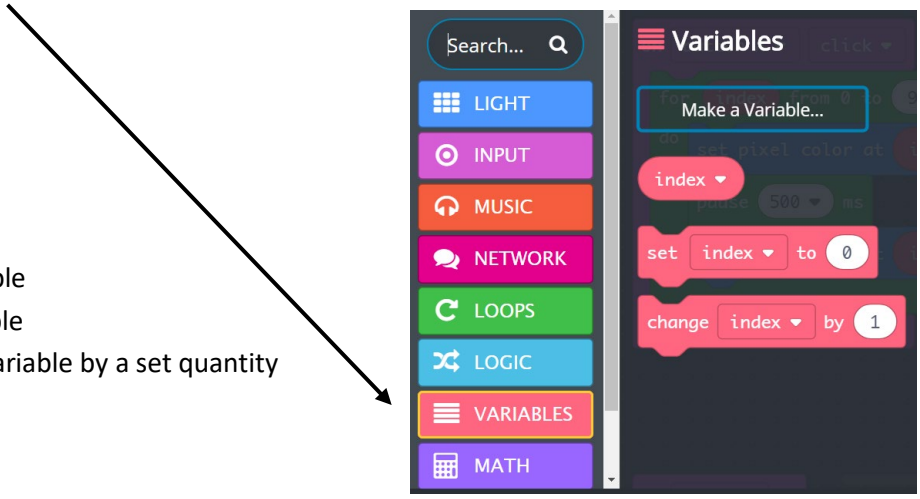


Substitute sounds instead of colours. Where would you put code to set the brightness or volume? Do that too.

Variables

We can use variables to **store values**, which can then be used later. To create a variable:

- click the Variable button
- click on **Make a Variable**
- type a name and click OK



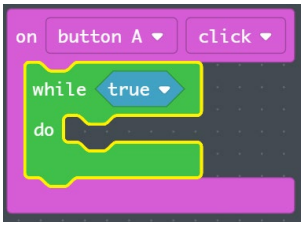
The list then shows blocks to:

- get the value of the variable
- set the value of the variable
- change the value of the variable by a set quantity

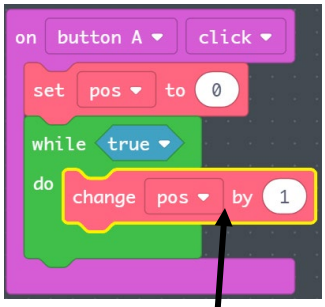
Use a While Loop to Turn on Individual Lights

Repeat loops are great for very simple repetitive tasks. What if we want to turn each neopixel in sequence. We use a **while** loop and a **variable** to do that. First create a variable called ***pos***.

1. Add the loop

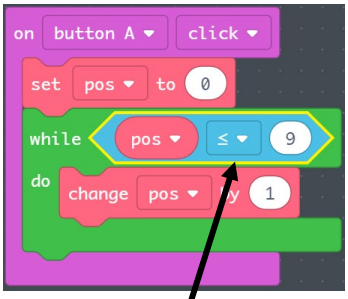


2. Add the variable pos:
set initial value to zero



count up

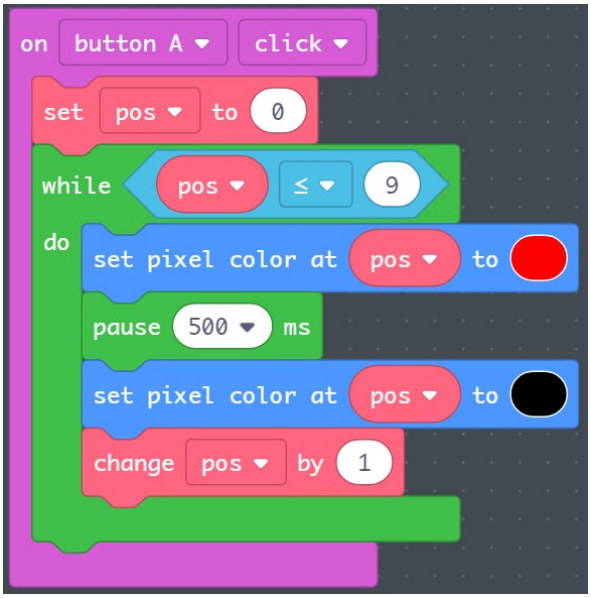
3. Test if the value of pos:
has it reached the last value



To find comparisons, click the Logic button

4. Do something inside the loop…

The Completed Loop



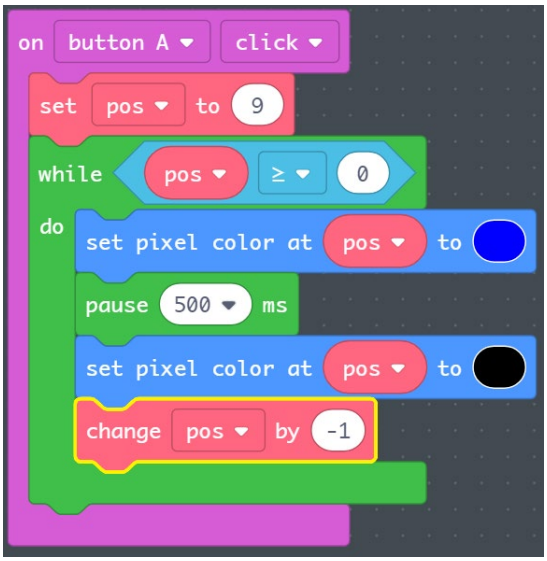
turn on the neopixel

turn off the neopixel

You must **set** the value of the variable (pos) before the loop and **change** its' value at the end of each loop – or it won't work.

Turning on Neopixels in Reverse

A **while** loop can start at a high number and end at a low number. We can use that to turn on each neopixel in reverse.



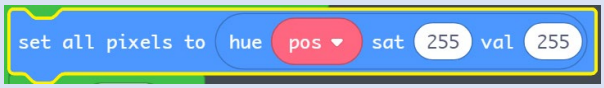
Start at 9

Stop when pos gets to zero

Count down

CPX Challenges

- Change the code to show every second or third neopixel
- Write a program to play tones, starting at 50 (hz) and going up by 100 at a time, until 5000 (hz) is reached. Then do the reverse.
- Write a program to smoothly change the colour of all the neopixels. Hint: The value of hue is 0 to 255.



- Show all neopixels then brighten or dim them all (0-100)

D. Sensors and Conditional Statements

Read the Light Sensor

To read and information from sensors, variables must be used to store the sensor values. Information can be displayed in the form of the neopixel graph. First, create a variable called ***light***.

Pass your phone light over, then away from, the CPX light sensor. Watch the neopixel graph.

Responding to Light Sensor Values

We can respond to sensor values (**inputs**) in many other ways – turn on lights, change light brightness, play sounds, and turn on motors or servos (all of these are **outputs**).

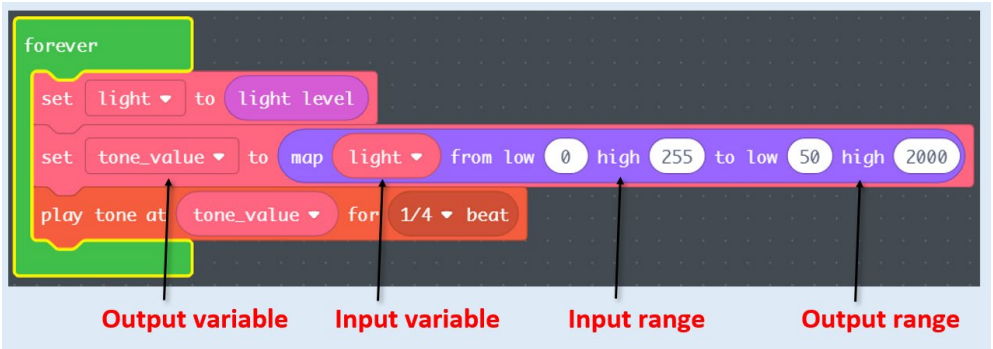
To use sensor values, they must be mapped to the range of values required by the output.

Input:	Light sensor	0		320
			Map the input range to the output range	
Output:	Neopixel brightness	0		100
	Sounds (tones)	50		6000
	Servos	0		180

Using the Map Block

Click the Math button and select the **map** block, to change (or map) the range of **input** values to the range of the **output** values. Variables are used to store the input and output values.

For example, to play a tone depending on the light level:

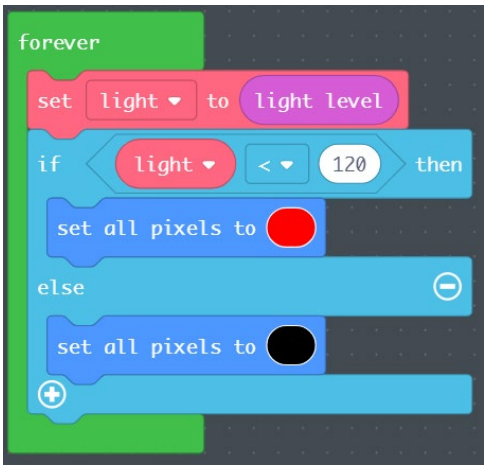


CPX Challenges

- Map the light sensor value to neopixel brightness
- Make neopixels brighter as the light sensor value decreases (try reversing the map low and high values)
- Change the colour (hue) of neopixels depending on the light sensor value

Conditional Statements (if-else)

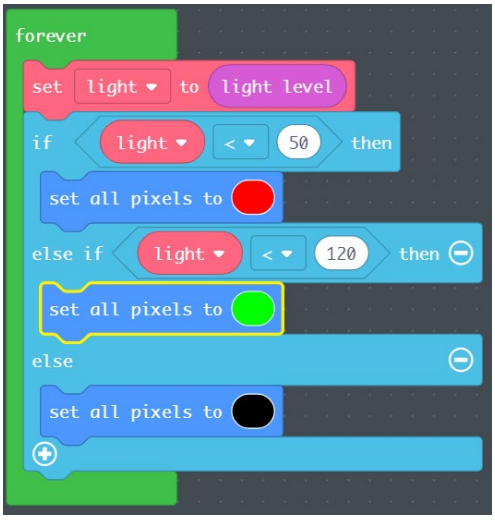
Conditional statements are decision statements. In other words, if a **condition is true** then a sequence of code is executed. If it is b, a different sequence of code is executed (or no code is executed).



If statements allow different actions to occur each side of a threshold (in this case a light sensor value of 120)

Multiple Conditional Statements

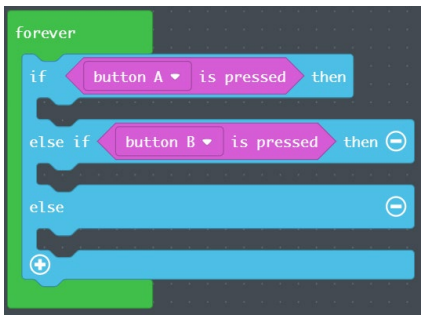
Conditional statements can have more than 2 levels:



Add sounds, light brightness, animations, repeat loops (to make lights flash on and off), and additional levels.

CPX Challenges

- Use if-then statements to test whether button A or B *is pressed* (or *was pressed*) and take different actions. Use this template.

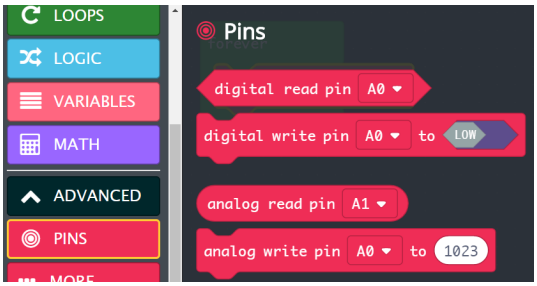


- Use if-then statements to test whether any of the touch pads is pressed (or was pressed) and take different actions.

E. Connecting External Devices

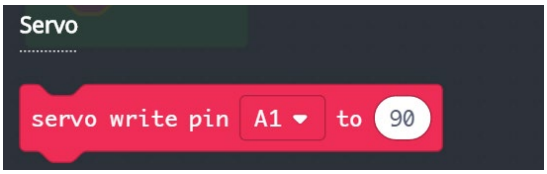
Coding the CPX to Communicate with External Devices

1. Click Advanced > Pins to access the blocks to read information from sensors and write values to servos.



read from a digital sensor (values are 0 and 1)

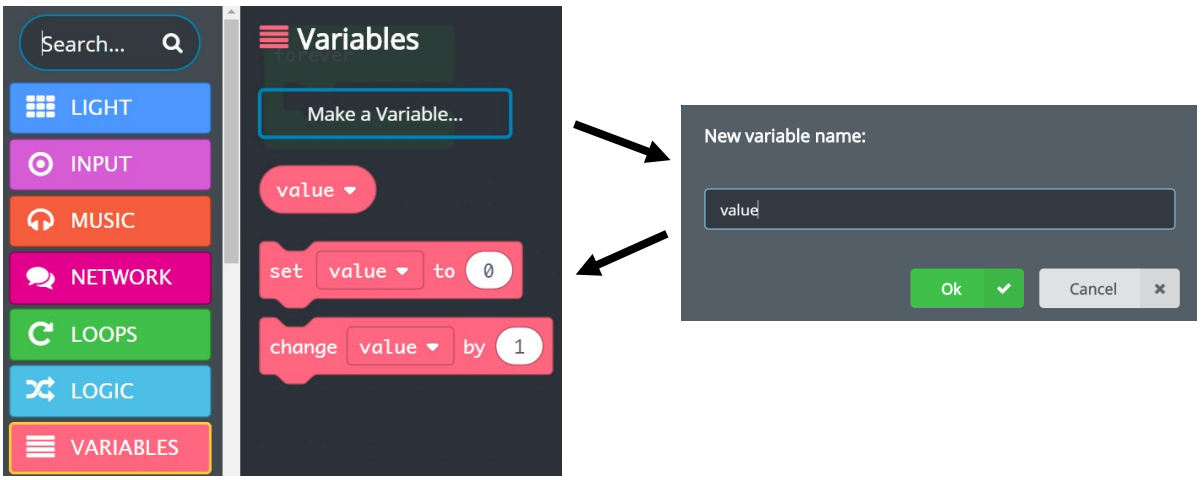
read from an analog sensor (usually 0 – 1024)



write values to a servo

2. We use variables to store a sensor value each time it is read.

   Click Variables > Make a Variable and enter the name (e.g. value)



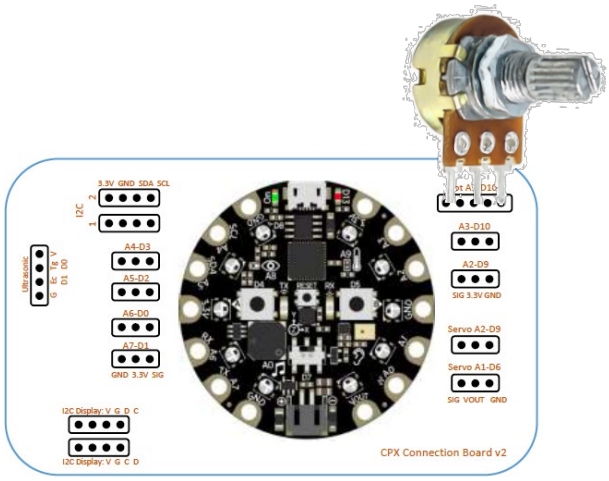
VERY IMPORTANT RULE

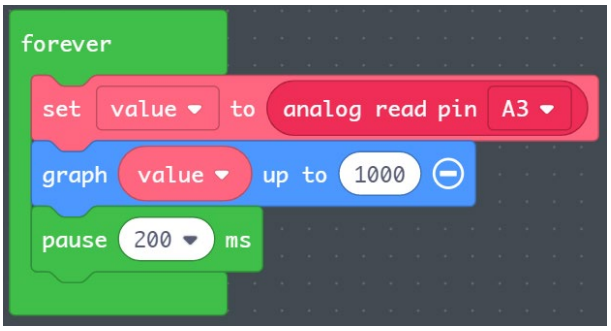
NEVER connect wires with the CPX connected to your computer

ALWAYS DISCONNECT THE USB CABLE
(pulling out the big connector to the computer)
or TURN OFF THE BATTERY

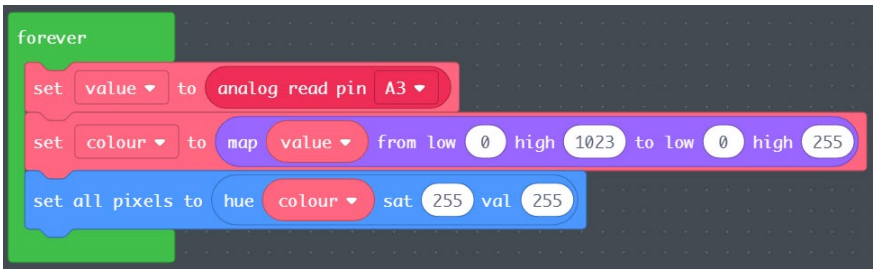
F. Connect a Potentiometer (Analog Sensor)

- A potentiometer takes the rotation of the shaft and turns it into a value between 0 and 1024.
- A potentiometer operates on 3.3V, one outside pin connects to **3.3V**.
- The other outside pin connects to **GND**
- The middle pin is the output, so can connect to any pin from **A2 – A7**. This sensor is **analog** and produces values between 0 and 1024.

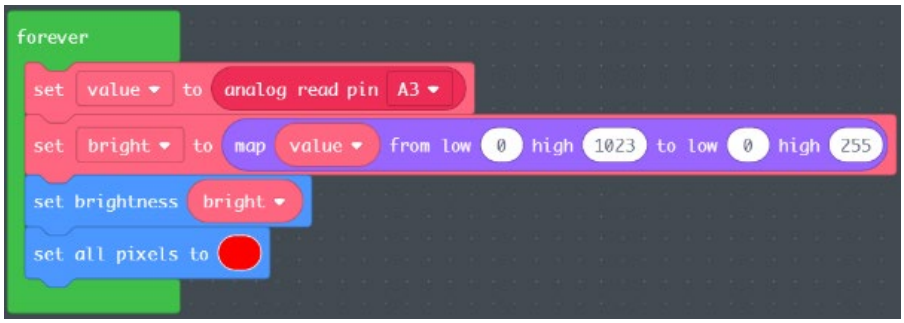


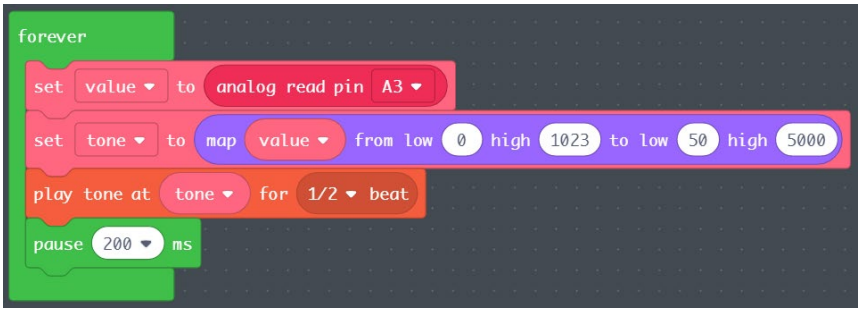


Change the colour of neopixels (create a new variable called *colour*)



Change the brightness of neopixels (create a new variable called *bright*)

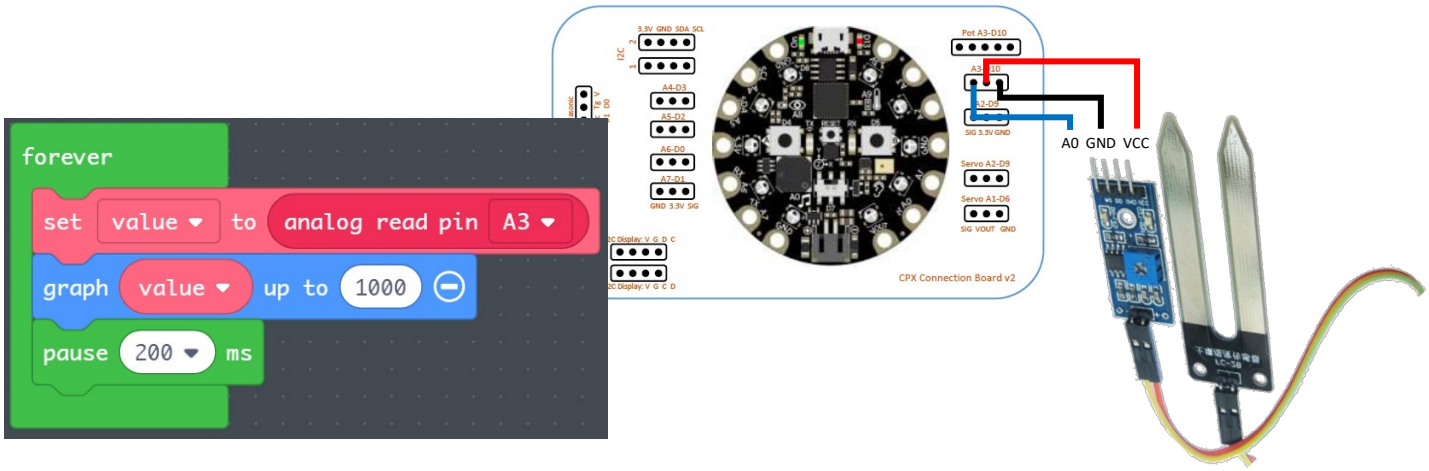




Play a tone (create a new variable called *tone*)

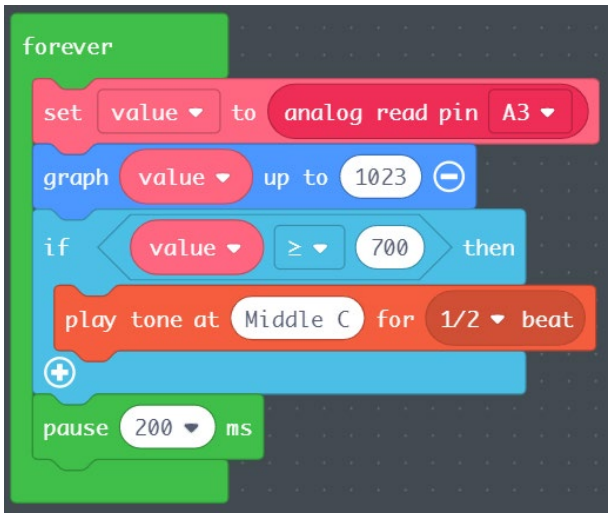
A

G. Connect a Soil Moisture Sensor (Analog Sensor)



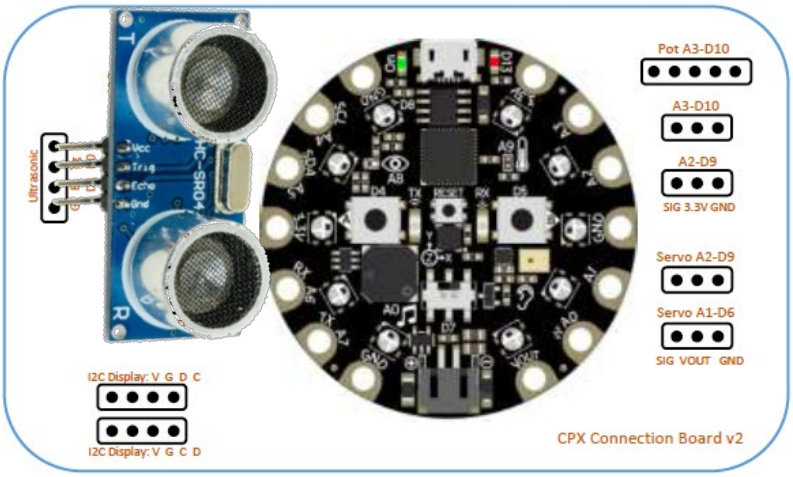
Make an Alarm Sound if Soil Moisture Falls Below a Certain Value

When there is no moisture the sensor reads about 1000. When put in water the sensor reads about 400. Therefore, when soil moisture falls the value **above** a certain point must be tested.

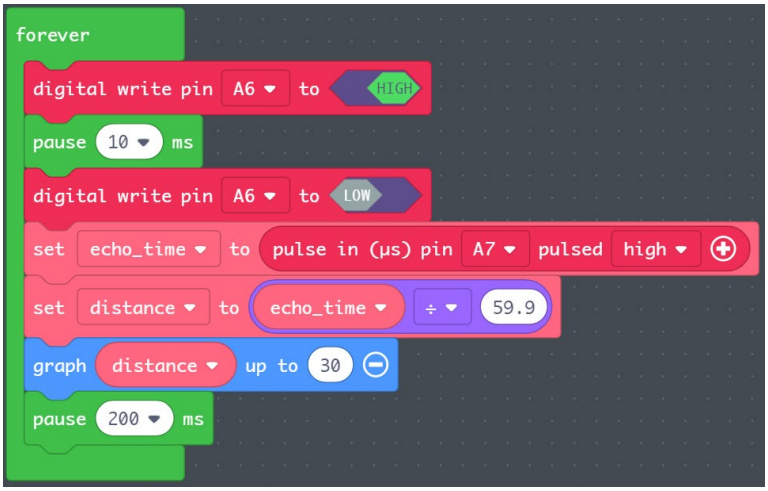


H. Connect an Ultrasonic Sensor

- A sensor needs 3.3V – the red wire goes to **3.3V**.
- The GND connects to **GND**
- The Trig and Echo are the control signal and can connected to any pins from **A3 – A7**. This sensor is **analog** (values 0 – 1024).



The ultrasonic sensor requires a signal (A6) to trigger a sound wave and a signal when the echo is received back (A7).



CPX Challenge

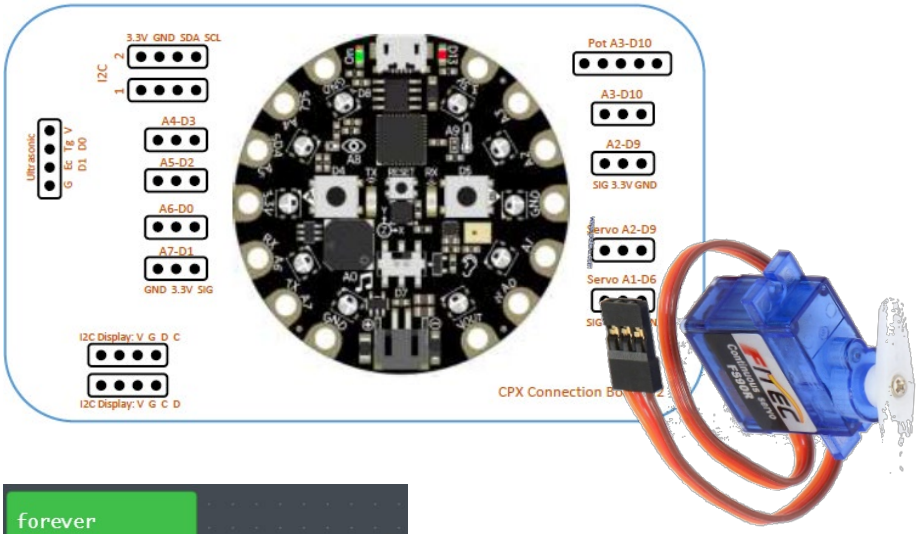
Write code to display different colours depending on the sensor distance or create a light graph of the value.

I. Connect Servos

- A servo needs 5V – the red wire connects to VOUT.
- The other outside pin connects to GND
- The orange pin is the control signal and can only connect to A1 or A2. A value between 0 and 180 must be sent to the servo.

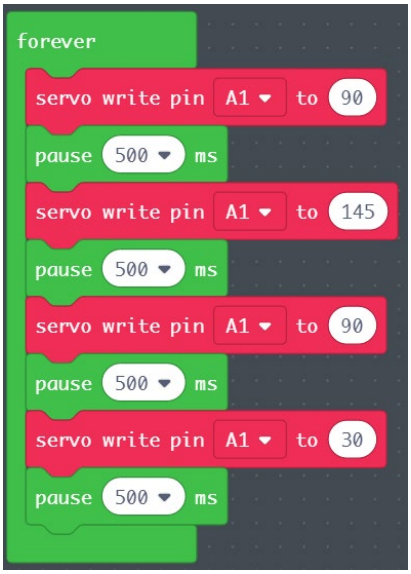
Ground (brown wire)
VOUT (red wire)
Signal (orange wire)

Servos



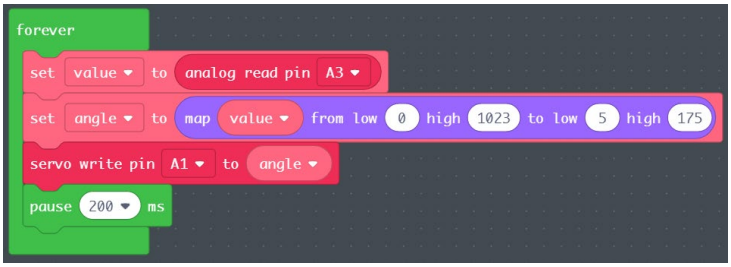
Micro Servo
0-180 angle of servo

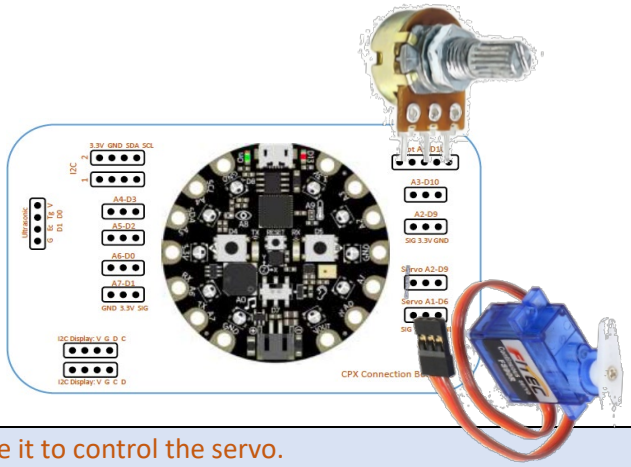
Continuous Servo
0-85 Backward
90 Stop
95-180 Forward



A continuous rotation servo acts like a motor. A value of 90 stops the servo. Values between 0 and 85 rotate the shaft backwards, and between 95 and 180 rotate the shaft forwards.

Use a Potentiometer to Move a Servo



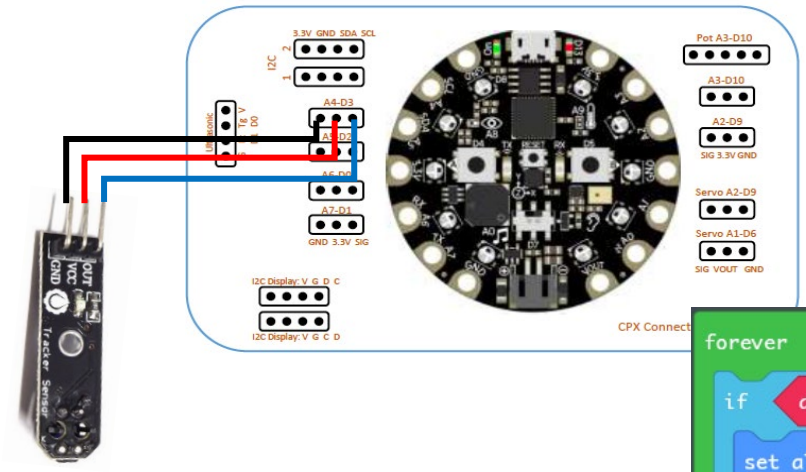


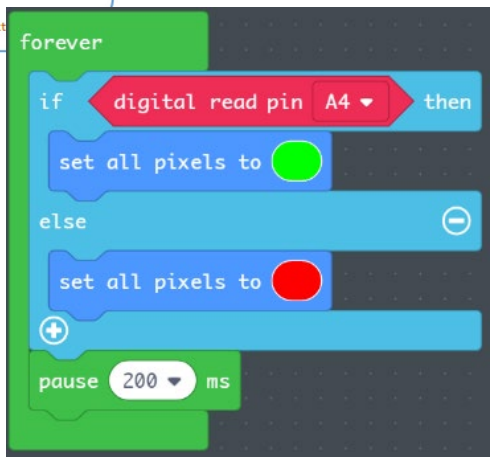
Change the potentiometer with the soil moisture sensor, and use it to control the servo.
Try both types of servo.

J. Connect Digital Sensors (e.g. Tracker, Collision Switch, Tilt Switch)

A Tracker Sensor can tell the difference between dark and light colours by the reflection of light.

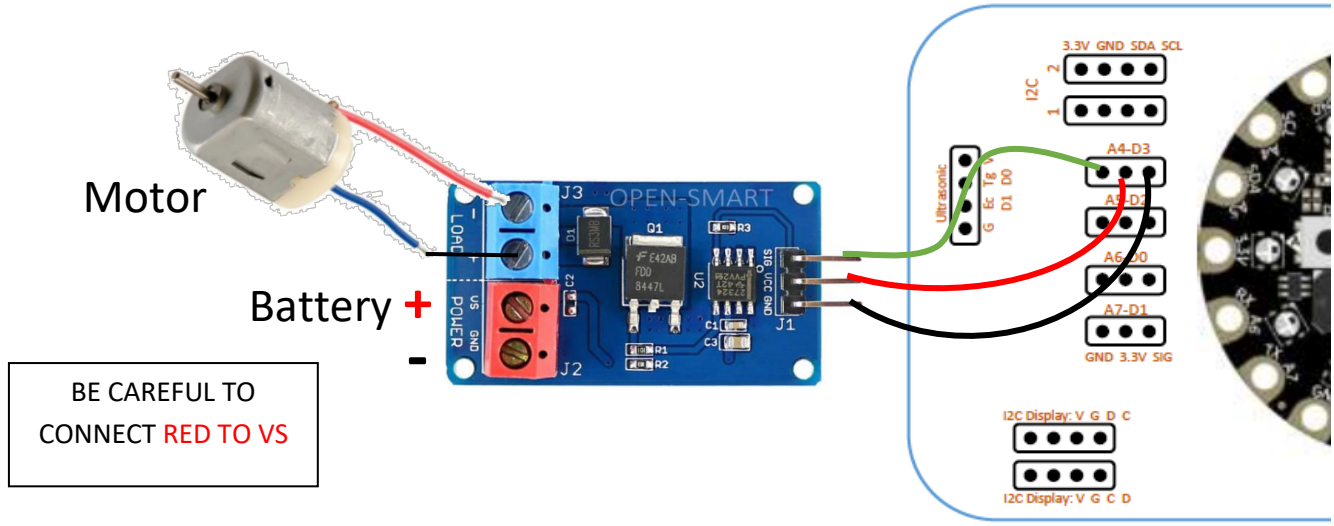
- A Tracker Sensor operates on 3.3V, so VCC connects to 3.3V.
- GND connects to GND
- OUT is the output signal (connect to any pin from A3 – A7). This is a digital sensor – the output values are 0 and 1 (True and False).





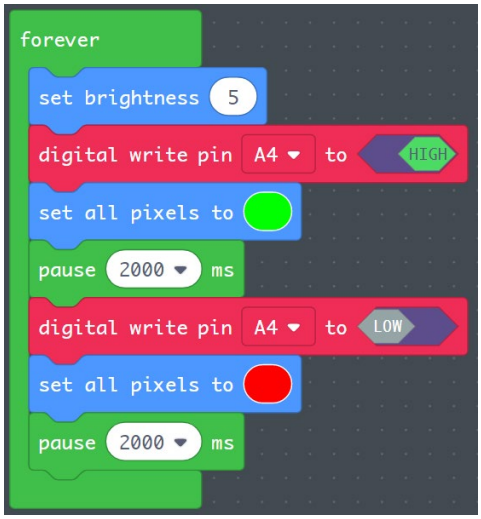
K. Connect Motors

Motors cannot be powered using the same power source as the CPX board and other sensors. A motor driver board (e.g. OPEN-SMART Single MOS Switch) is required to supply battery power to the motor, and control the motor.



Turn the motor on and off (full power)

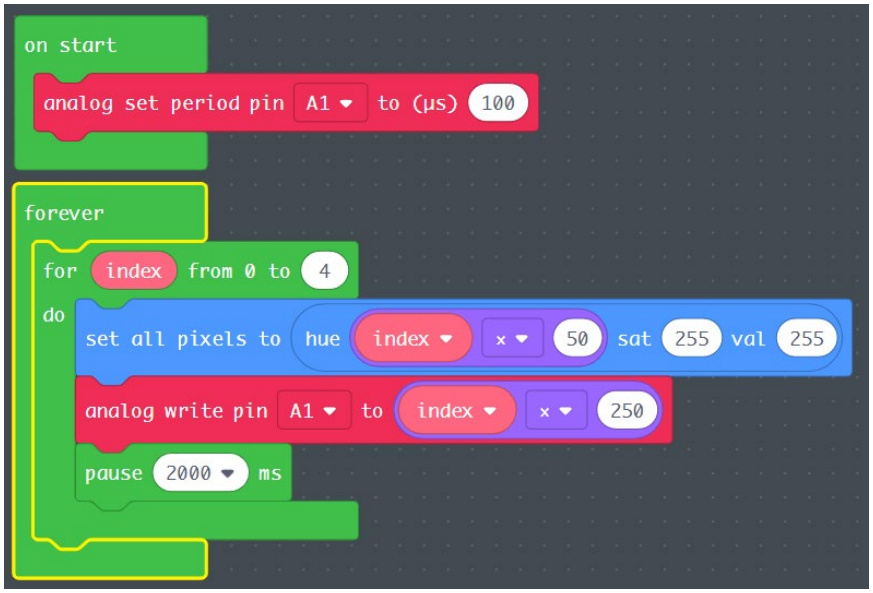
Any pin can be connected using *digital write*.



Control Power to the Motor (using PWM– D0, D6, D9, D10 only)

Only pins A1 and A2 can be connected using *analog write*.

Analog write requires values between 0 (off) and 1023 (full on)



CPX Challenge

Connect a Potentiometer or Soil Moisture Sensor to run the motor (set a threshold value to turn it on and off)

L. Obstacle Avoidance/Line Follower Servo Vehicle

M. Propeller Vehicle with Steering