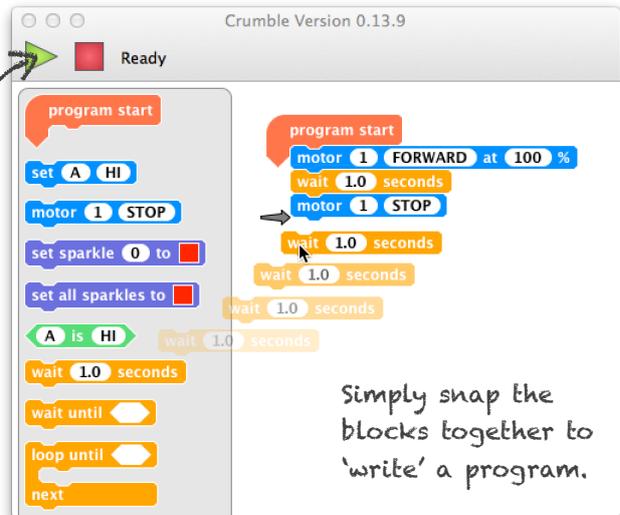


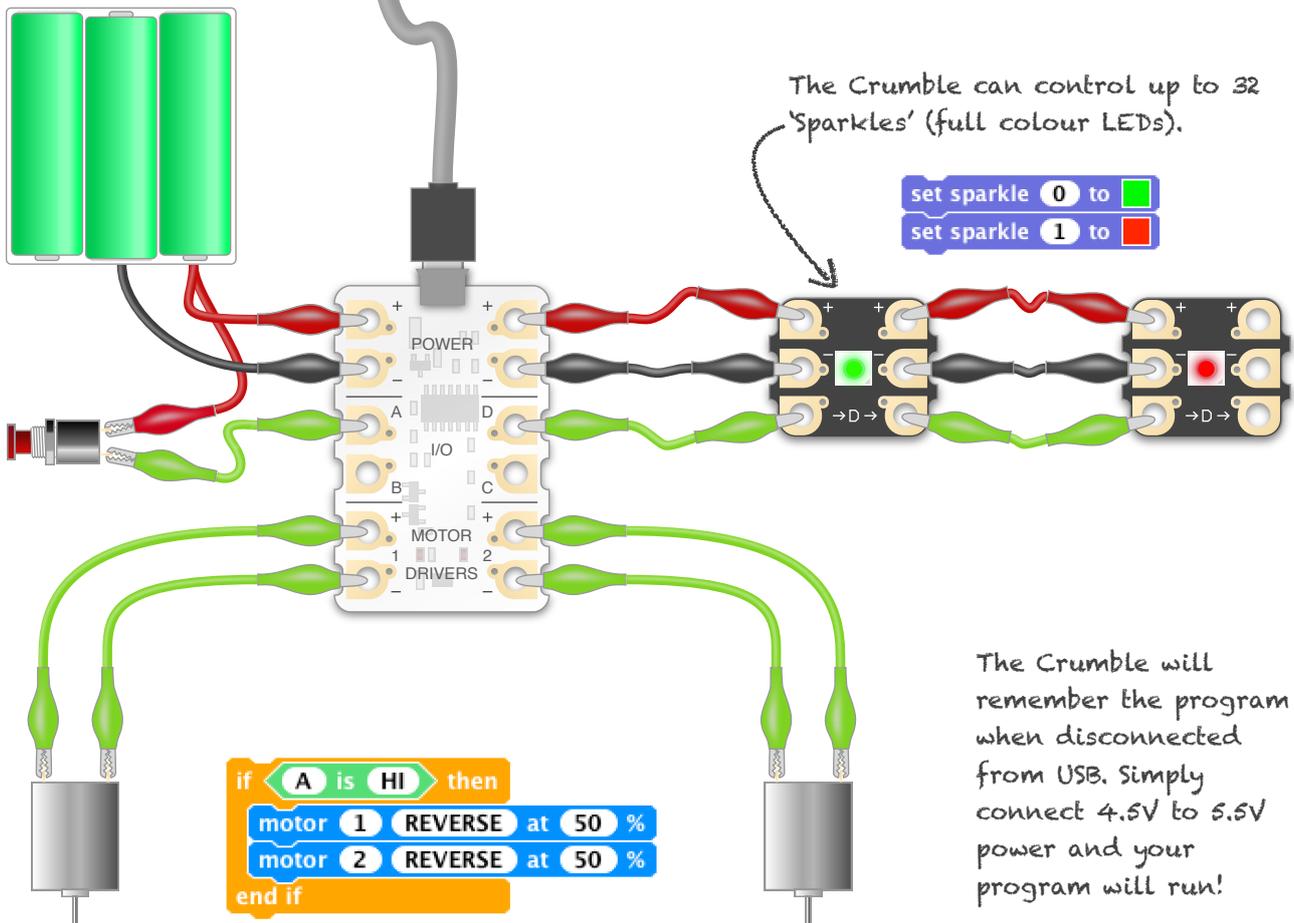
The Crumble Controller

The Crumble is a low-cost, easy-to-use electronics controller. A few 'croc' leads and a USB cable are all you need to connect motors, LEDs and sensors and begin experimenting. No programming experience is required as the FREE software is a graphical, drag-and-drop system inspired by MIT Scratch.



www.redfernelectronics.co.uk/crumble-software

Click here to program the Crumble over USB

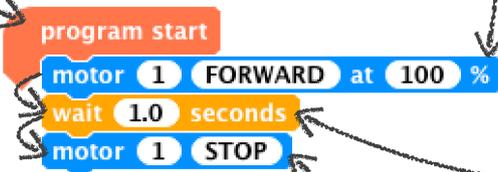


Sense inputs, control motors and much, much more!

What is a Program?

A program is simply a list of instructions that the Crumble will follow. The Crumble will 'run' a single line (i.e. a block) of a program, one at a time, starting from the top.

This block tells the Crumble where the program starts (you always need one of these).



This block will turn the motor 1 output on (forwards).

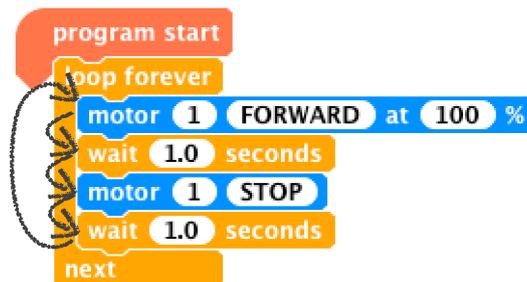
Most blocks only take a very small fraction of a second to complete.

This wait block takes 1 second to complete.

Finally, this block will turn off motor 1.

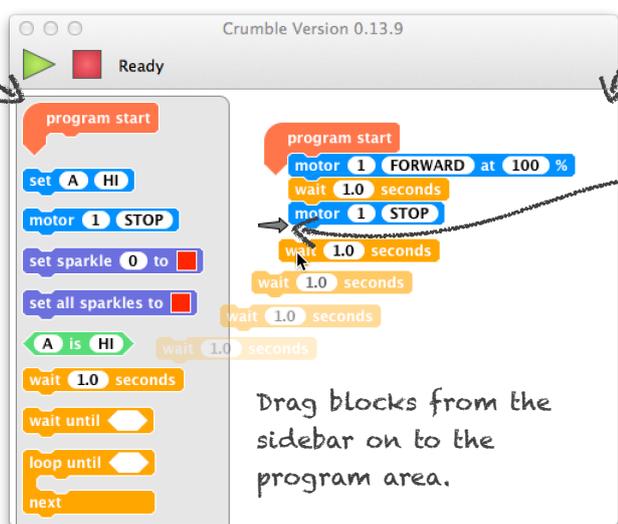
Motor 1 is turned on for 1 second, however, without the 'wait' block this program would appear to do nothing: motor 1 would be turned off immediately after it is turned on.

The flow of a program can also be controlled using blocks like this 'loop forever' block.



Once the Crumble reaches the last instruction within the 'loop forever' block, the Crumble will jump back to the top of the loop and repeat the instructions (see the step-by-step example on the next page).

Writing Programs for Crumble



Drag blocks from the sidebar on to the program area.

Snap blocks together to build a program.

Click on a block's parameters to change it.

A Step-by-Step Example

The following example show how to write a simple program to flash a motor output LED.



Drag a start block to the code area and attach a 'loop forever' block to it.



Drag a motor block inside the loop.



Click on STOP so it changes to FORWARD.

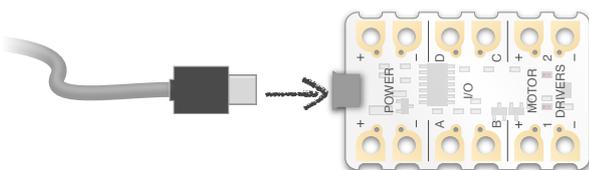


Now add a wait block underneath the motor block, but still within the loop.

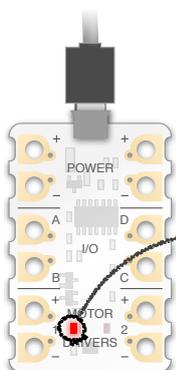
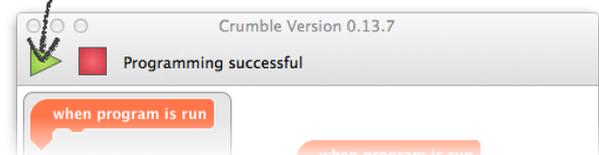


Complete the program with another motor block and a wait block.

Connect to PC using micro USB cable.



Click here to send the program to the Crumble and run it.



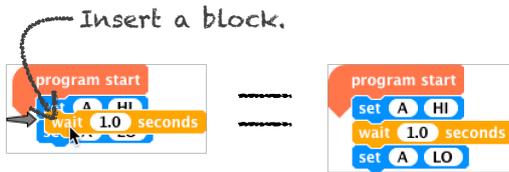
The on-board LEDs show the motor output status.

Output 1 turns on for 1 second every 2 seconds. The on-board LED will flash.

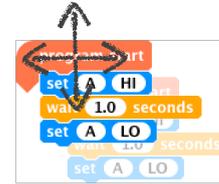
The motor outputs can act as high power outputs for many types of device, not just motors.

Crumble Software Basics

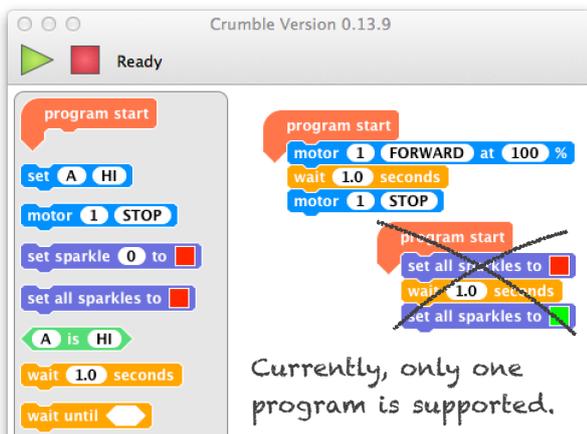
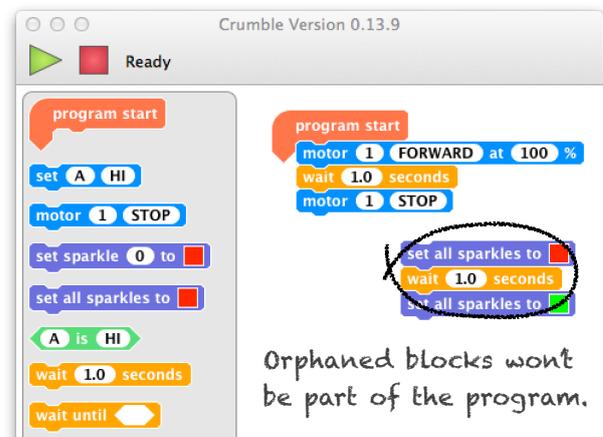
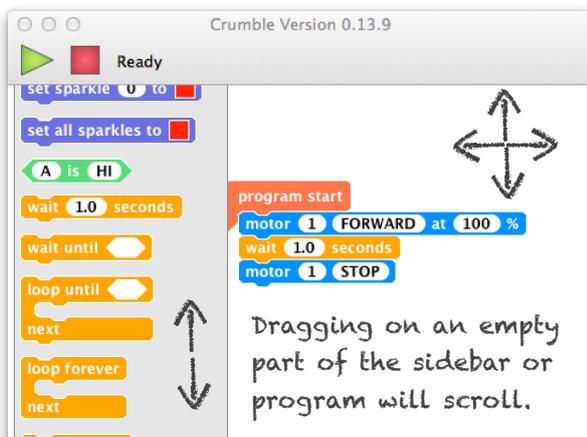
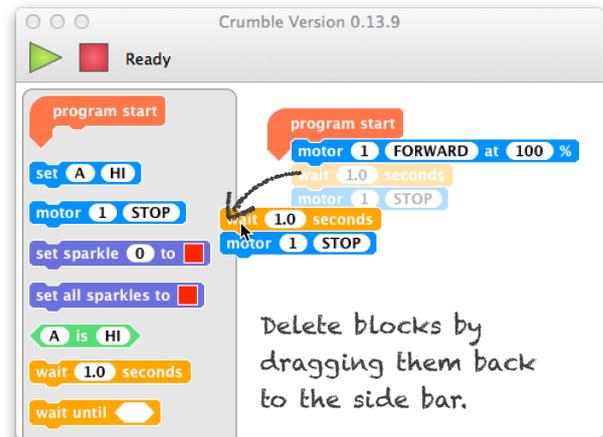
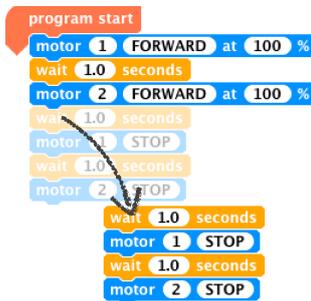
Below are few tips on how to use the Crumble software.



Moving a block will also move all blocks that are attached below.



Moving a block in a program will split the program.



Once the Crumble has been programmed over USB, it will remember the program and run it automatically when power is supplied (see below for battery/power connection).

Using Motors

Click to select motor 1 or 2.



Click to select FORWARD, REVERSE or STOP.

When FORWARD or REVERSE is selected, the speed can also be set (default is full speed).



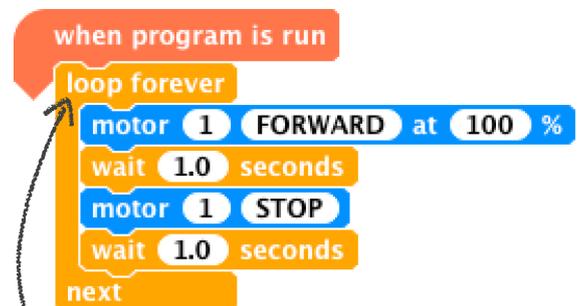
Click to edit speed. Examples:

100% Full speed

0% Stop

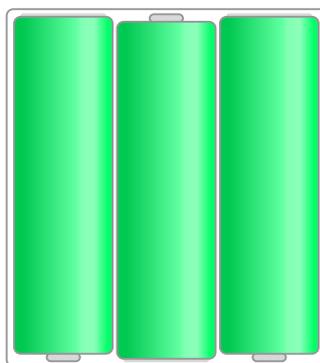
25% Quarter speed

Example program

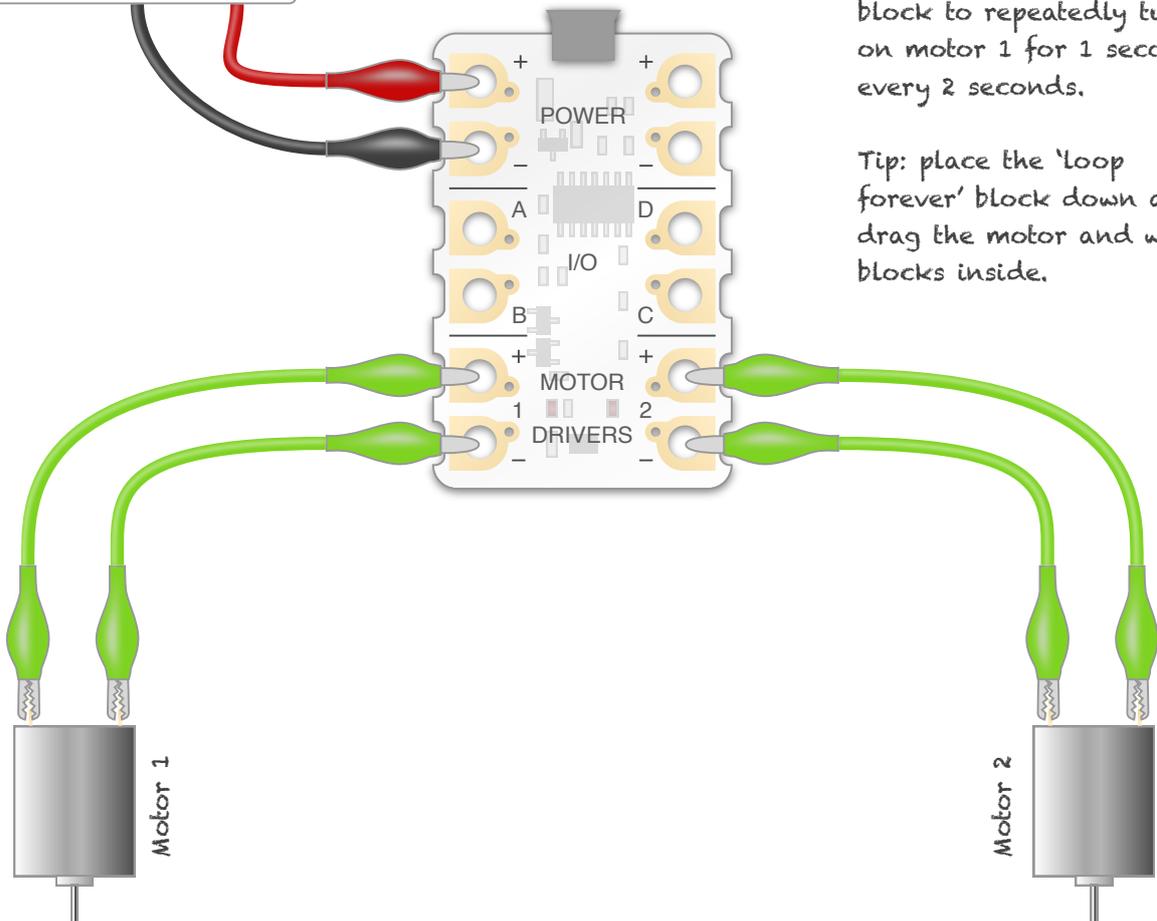


Use a 'loop forever' block to repeatedly turn on motor 1 for 1 second, every 2 seconds.

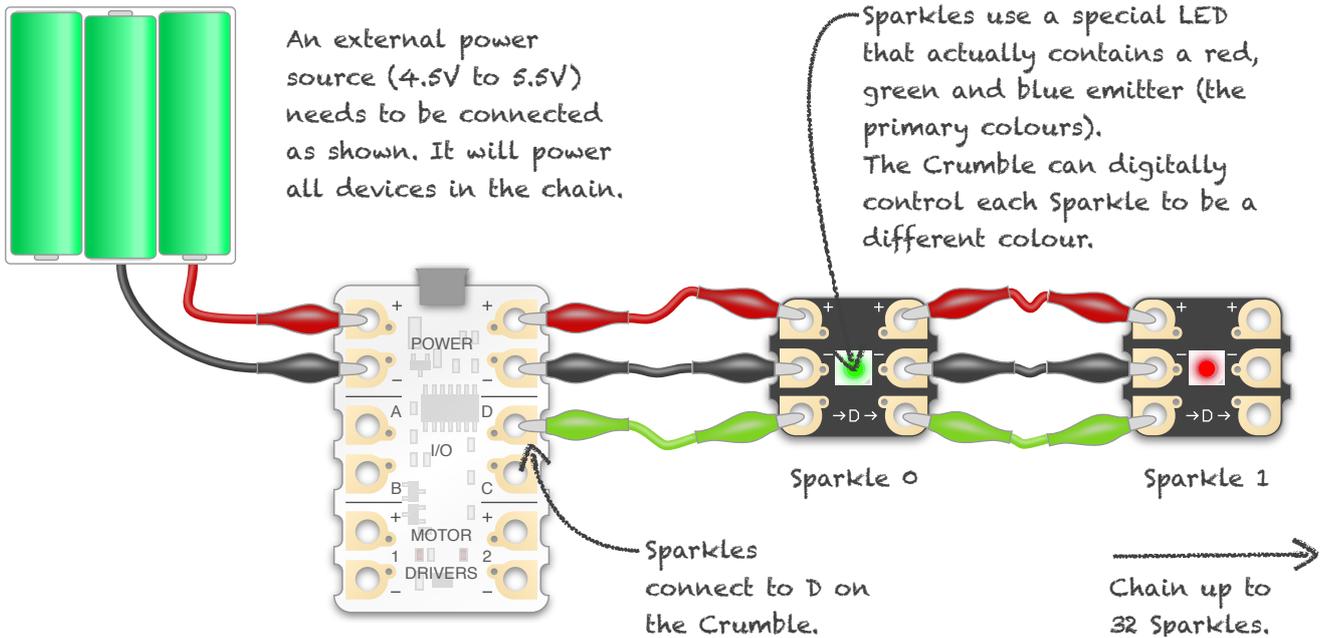
Tip: place the 'loop forever' block down and drag the motor and wait blocks inside.



The motor outputs are not powered by USB. An external power source (4.5V to 5.5V) needs to be connected as shown.

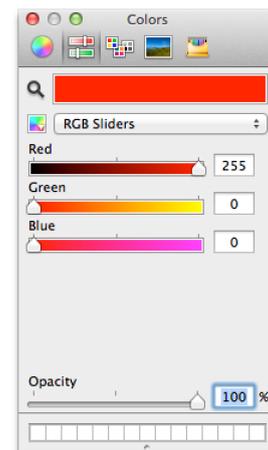


Using Sparkles



Click to enter an address. The block will only change the colour of the Sparkle it is addressing.

Click here to change the colour



The first sparkle in the chain is 0, the next is 1 etc.

Use this block to change the colour of every Sparkle connected to the Crumble.

set all sparkles to [yellow square]

Note: the colour picker window will be a little different on Windows, Mac and Linux.

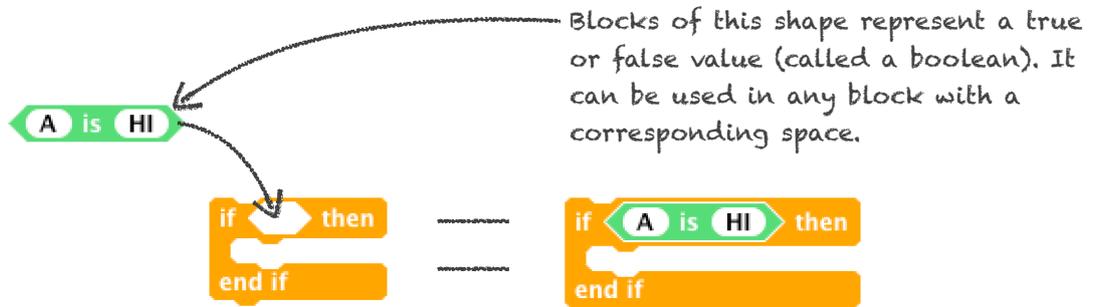
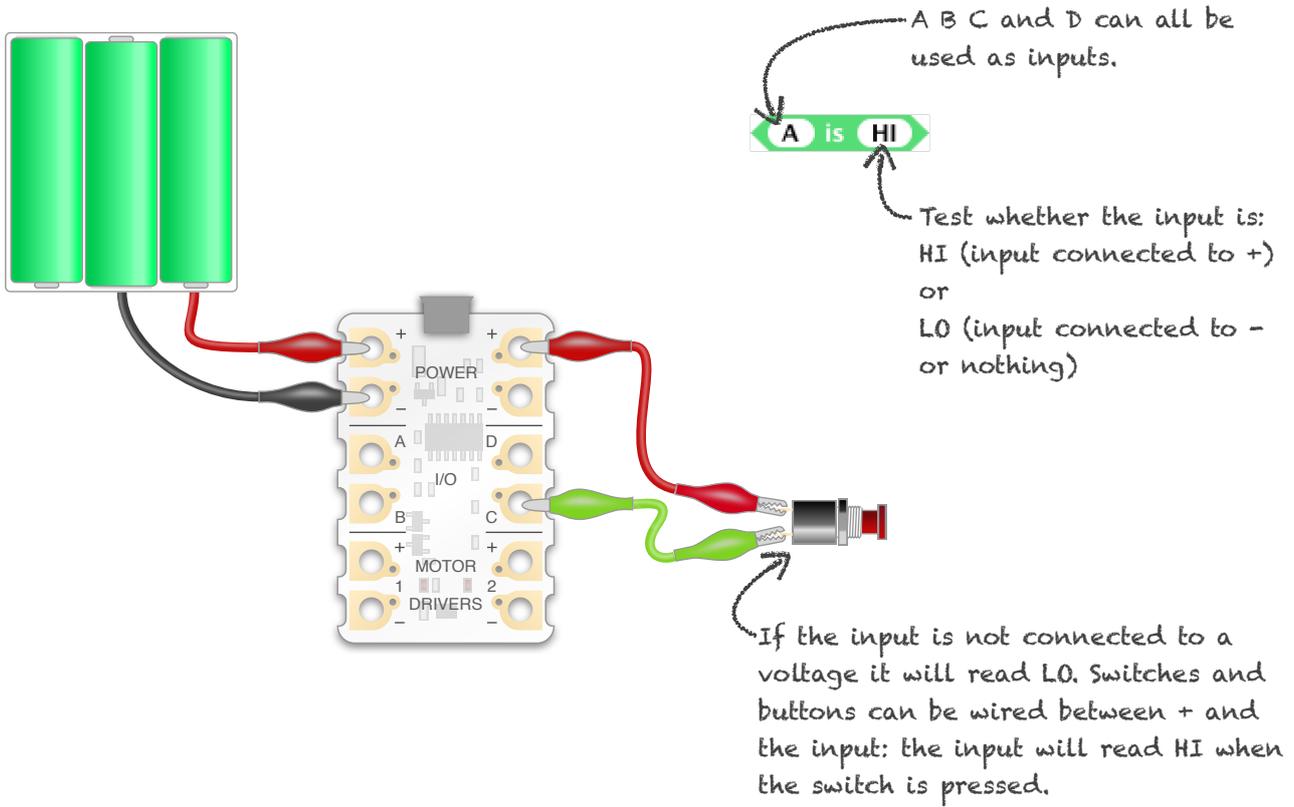
Example program

```

program start
loop forever
  set sparkle 0 to [red square]
  wait 1.0 seconds
  set sparkle 0 to [green square]
  wait 1.0 seconds
next
  
```

This program switches the first Sparkle between red and green repeatedly.

Using Inputs



Example program

```

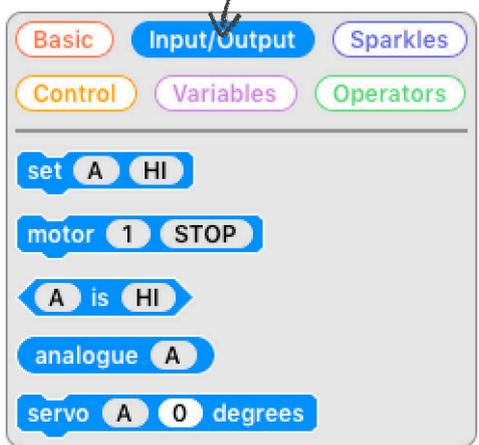
program start
loop forever
  wait until C is HI
  motor 1 FORWARD at 100 %
  wait 5.0 seconds
  motor 1 STOP
next
  
```

In this example, motor 1 runs forward for 5 seconds every time the button is pressed (C goes HI).

Analogue Inputs

The Crumble controller can measure voltages between 0V and the voltage of the power supply (i.e. 4.5V when connected to 3xAA batteries). The voltage is converted to a number between 0 and 255. The crumble can also be connected directly to analogue sensors like LDRs and thermistors, without any additional components.

The analogue input block can be found under the Input/Output section of the sidebar.

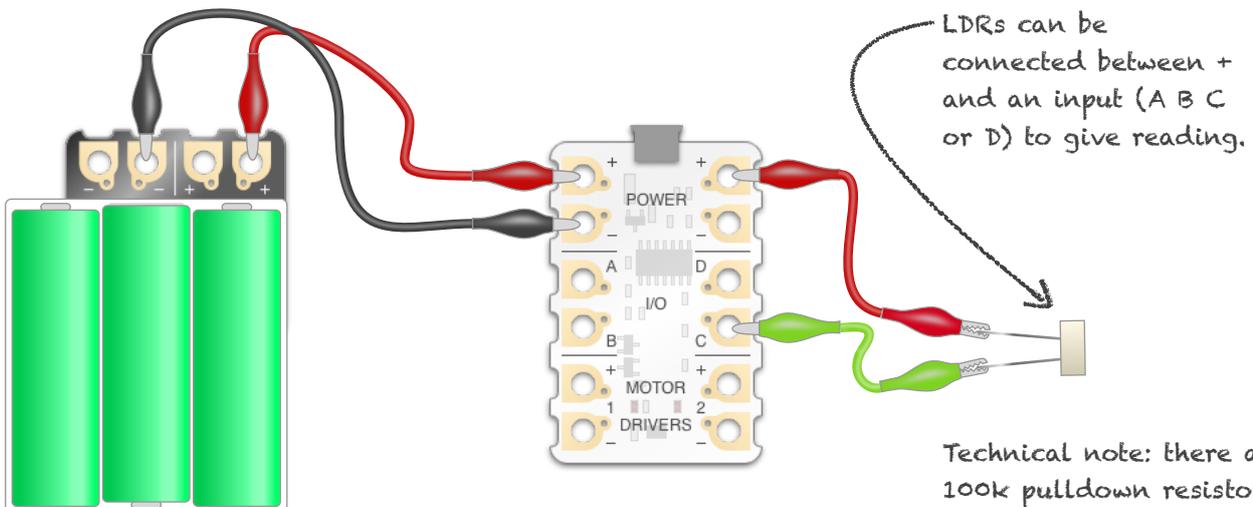


A B C and D can all be used as analogue inputs.

analogue A

This block will have a value between 0 and 255 depending on the voltage measured on A.

The current value can be monitored over USB using a variable (see the next section).



Technical note: there are 100k pull-down resistors onboard the Crumble. Hence, the LDR forms a potential divider.

Example program

```

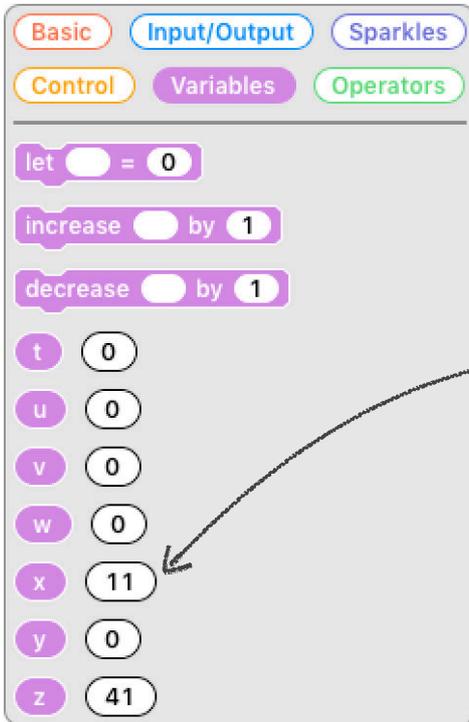
program start
do forever
  if analogue C < 100 then
    set sparkle 0 to [ ]
  else
    turn sparkle 0 off
  end if
end loop
  
```

In this example, the LDR is connected as above (with the addition of a sparkle) to form a simple night light.

When it's dark the analogue value of C decreases below 100 and the

Variables and Maths

A variable can be used to store a value. There are currently 7 variables available under the 'Variables' section of the sidebar.



This block would set x to 10.

```
let x = 10
```

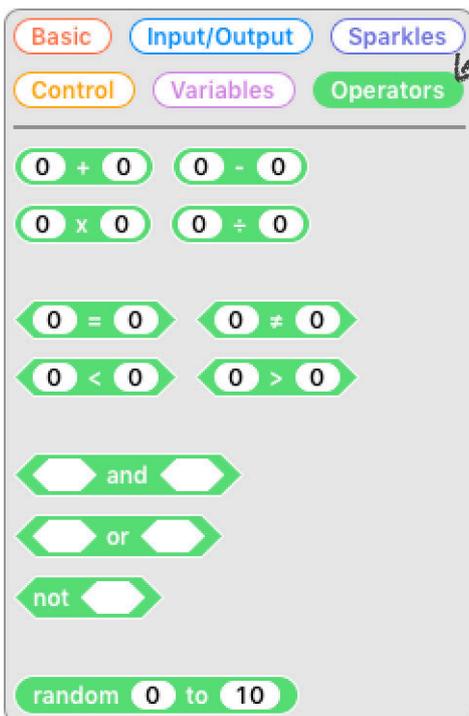
This block increases the value of x by 1

```
increase x by 1
```

The number next to each variable shows the current value of a variable (if the Crumble is connected).

E.g. these can be used monitor analogue inputs.

```
program start
do forever
  let z = analogue A
loop
```



Basic mathematical functions can be found in the 'Operators' section.

This would set x to double the value of y.

```
let x = y x 2
```

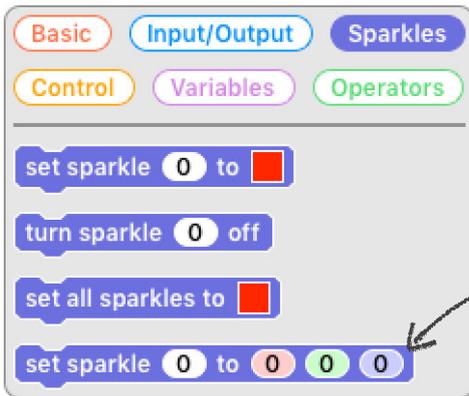
This would set v to a random number between 0 and 10 (inclusive).

```
let v = random 0 to 10
```

This would set B HI if x is equal to 10, otherwise B will be set LO.

```
if x = 10 then
  set B HI
else
  set B LO
end if
```

Advanced Sparkle Control



Each Sparkle contains three LEDs, one for each primary colour: red, green and blue. The apparent colour of each Sparkle is controlled by mixing different amounts of the primary colours, just like a pixel on a colour screen.

This block controls the brightness of each of the primary colours: a value of zero is off, and a value of 255 is maximum brightness.

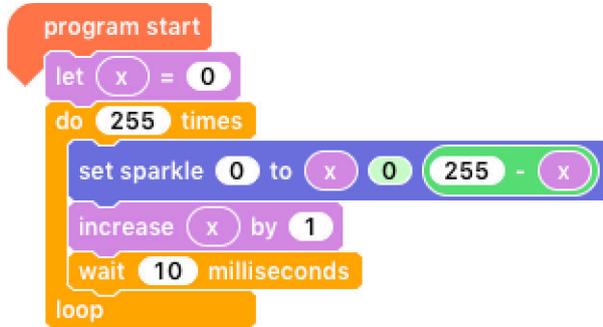


For example, this block sets the first sparkle (Sparkle 0) red at about half brightness.



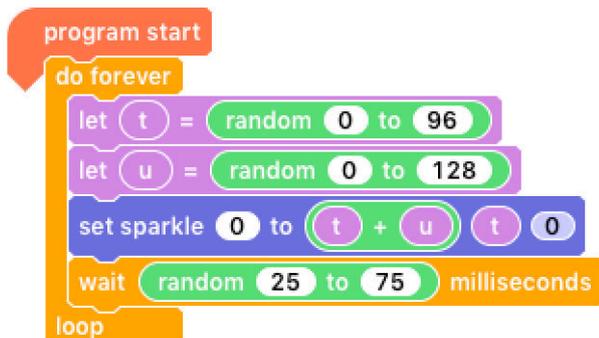
This block sets both red and blue to half brightness to produce a purple light.

Example program



Variables can be used to control the colour of Sparkles. This program fades a Sparkle from blue to red.

Example program



This example produces a flickering candle effect using random numbers and a sparkle.

A red-yellow colour is chosen by mixing a random amount of red and green.

Waiting for a random delay replicates the flickering effect.