

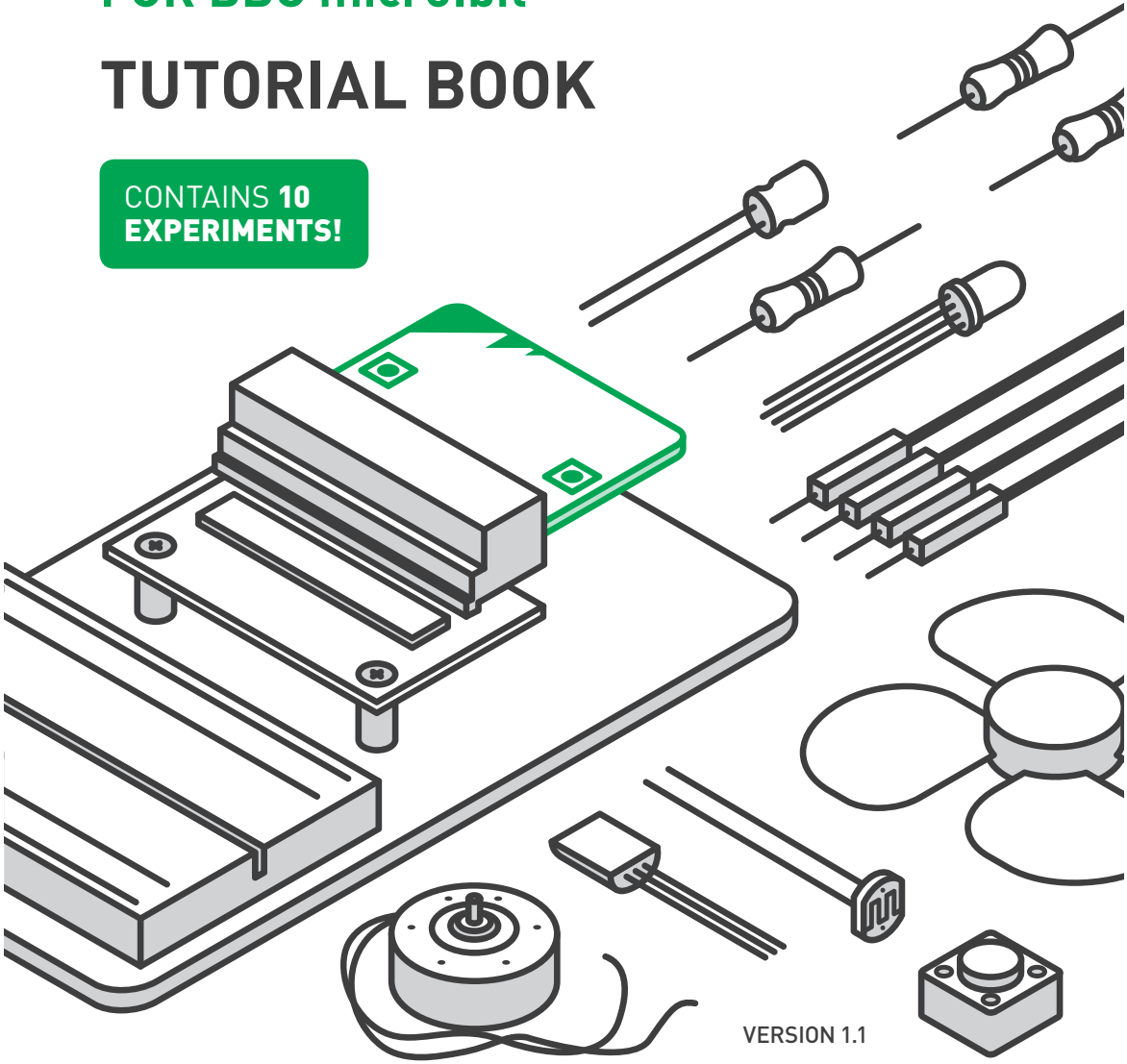


INVENTOR'S KIT

FOR BBC micro:bit

TUTORIAL BOOK

**CONTAINS 10
EXPERIMENTS!**



VERSION 1.1

The **Kitronik Inventors Kit for the BBC micro:bit** provides a fantastic way of learning how to construct and control electronic circuits. The possibilities are endless and to get you off to a flying start we have included this easy to follow tutorial book which guides the user through 10 projects – with many more available online by visiting www.kitronik.co.uk/5603

Every effort has been made to ensure this manual is accurate. Check to see if there is an updated version of this manual available to download (above link).

CONTENTS

ASSEMBLING THE PROTOTYPING PLATE	3
INVENTORY OF PARTS	4–5
HOW DOES A BREADBOARD WORK?	6
GETTING CONNECTED	7
THE BBC micro:bit SOFTWARE	8

MICROSOFT BLOCK EDITOR	INTRODUCTION	9
	COMMANDS	10–12
	GETTING A PROGRAM ONTO THE BBC micro:bit	13
	EXP. 1 – SAY “HELLO” TO THE BBC micro:bit!	14
	EXP. 2 – USING AN LDR & ANALOG INPUTS	18
	EXP. 3 – DIMMING AN LED USING A POTENTIOMETER	22
	EXP. 4 – USING A TRANSISTOR TO DRIVE A MOTOR	26

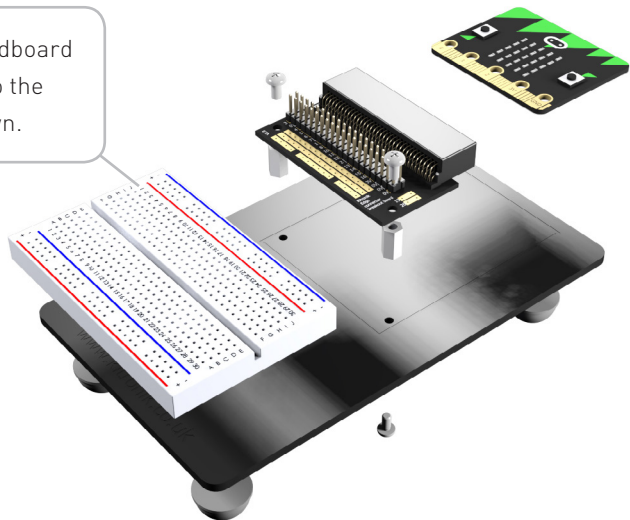
MICROSOFT TOUCH DEVELOP EDITOR	INTRODUCTION	30–31
	EXP. 5 – USING THE ACCELEROMETER TO CONTROL MOTOR SPEED	32
	EXP. 6 – SETTING THE TONE WITH A PIEZO BUZZER	36
	EXP. 7 – WIND POWER	40
	EXP. 8 – MAKING A GAME USING THE COMPASS	44
	EXP. 9 – CAPACITOR CHARGE CIRCUIT	48
	EX. 10 – USING AN RGB LED	52

ASSEMBLING THE PROTOTYPING PLATE

The Inventor's Kit is supplied with a simple prototyping plate on to which is attached a breadboard and Edge Connector breakout board. Once assembled, this will be used to connect all of the parts used in the experiments.

Assemble the prototyping plate as shown in the diagram below.
Insert the BBC micro:bit in the orientation shown below.

Make sure that your breadboard is fitted with column '1' to the **left** of the board, as shown.

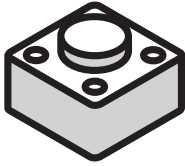


Note: Breadboard & feet are self-adhesive.

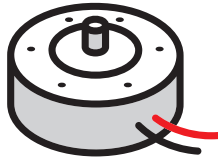
PROTOTYPING ASSEMBLY PARTS

PERSPEX MOUNTING PLATE	x1	PAN HEAD SCREW	x4
PLASTIC SPACER 10MM	x2	RUBBER FEET	x4
STICKY FIXER FOR BATTERY PACK (OPTIONAL USE)	x1	BREADBOARD	x1
EDGE CONNECTOR BREAKOUT BOARD FOR BBC micro:bit	x1		

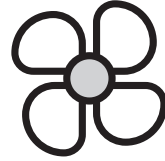
SUPPLIED WITH THIS KIT



PUSH SWITCH
x4



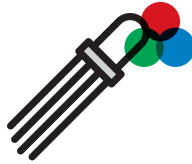
MOTOR
x1



FAN BLADE
x1



TRANSISTOR (NPN BC337)
x1



RGB 5MM LED
x1



RED 5MM LED
x2



GREEN 5MM LED
x2



YELLOW 5MM LED
x2



ORANGE 5MM LED
x2



47 Ω RESISTOR
x5



2.2k Ω RESISTOR
x5



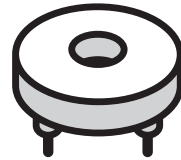
10k Ω RESISTOR
x5



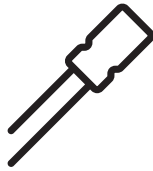
LDR (LIGHT SENSOR)
x1



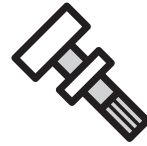
TERMINAL CONNECTOR
x1



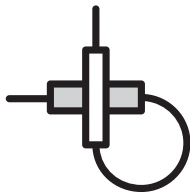
PIEZO ELEMENT BUZZER
x1



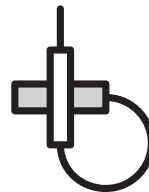
470uF ELECTROLYTIC CAPACITOR
x1



POTENTIOMETER & FINGER ADJUST SPINDLE
x1



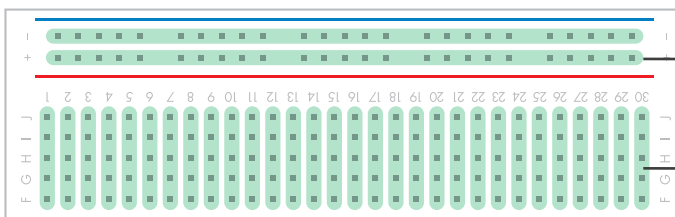
MALE TO MALE JUMPER WIRES
x10



MALE TO FEMALE JUMPER WIRES
x10

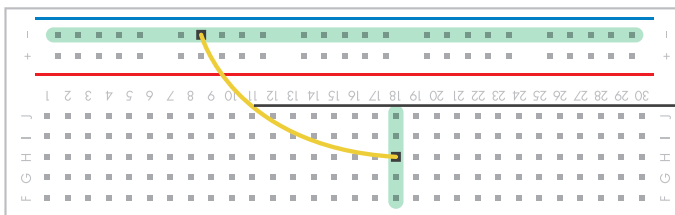
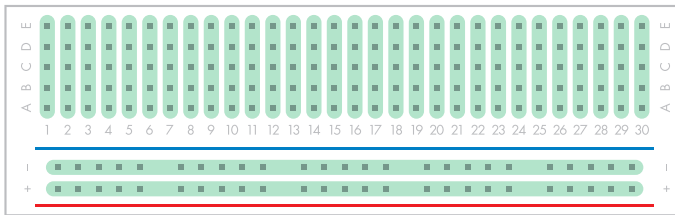
HOW DOES A BREADBOARD WORK?

A breadboard allows the prototyping of circuits without the need for soldering. Parts can be pushed into the breadboard and stay held in place. They can be removed and moved around as required to build and alter circuits. Breadboards have internal connections that connect rows or columns of parts. These rows or columns can also be connected to one another using jumper wires. The diagrams below show how both of these types of connections are made.

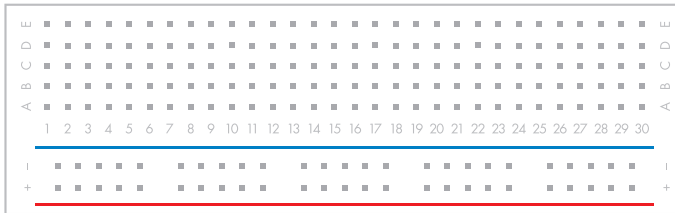


All contacts in this row are internally connected.

All contacts in this column are internally connected.

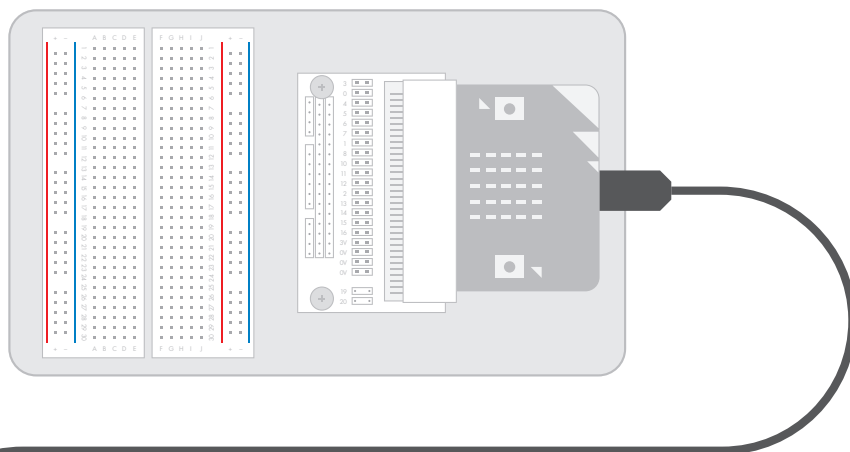


Adding a Male-to-Male Jumper Wire connects the highlighted column to the highlighted row.



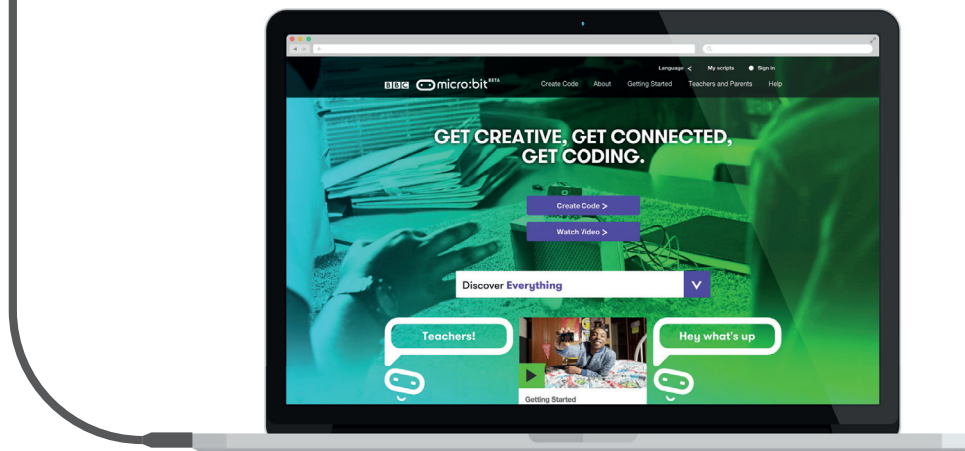
GETTING CONNECTED AND FINDING THE PROGRAMMING ENVIRONMENT

Using a USB to micro-USB type B cable, connect the BBC micro:bit to a computer.



Code will be created on the BBC micro:bit website:

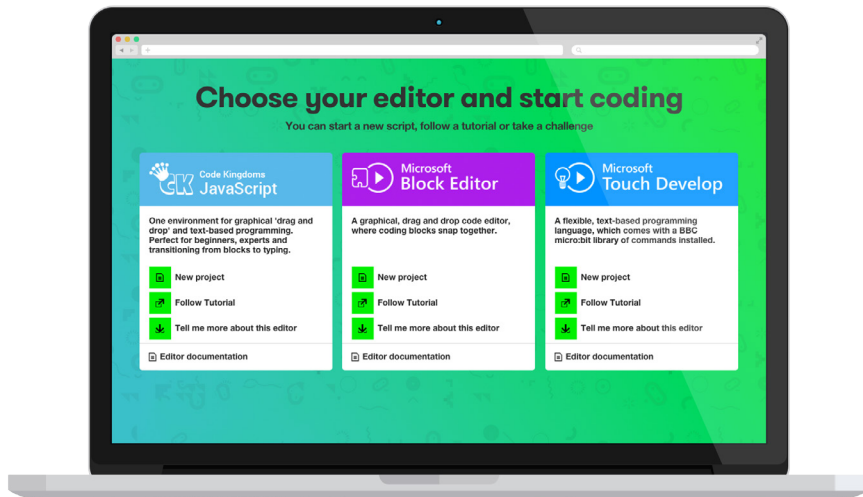
www.microbit.co.uk



THE BBC micro:bit SOFTWARE

The BBC micro:bit is programmed using a web based (internet access required) programming environment which is found at www.microbit.co.uk.

To save programs to access at a later date you will need to be logged in (although you can create a program without doing this). If this is the first time you have used your BBC micro:bit then please refer to our getting started guide at www.kitronik.co.uk/microbit.

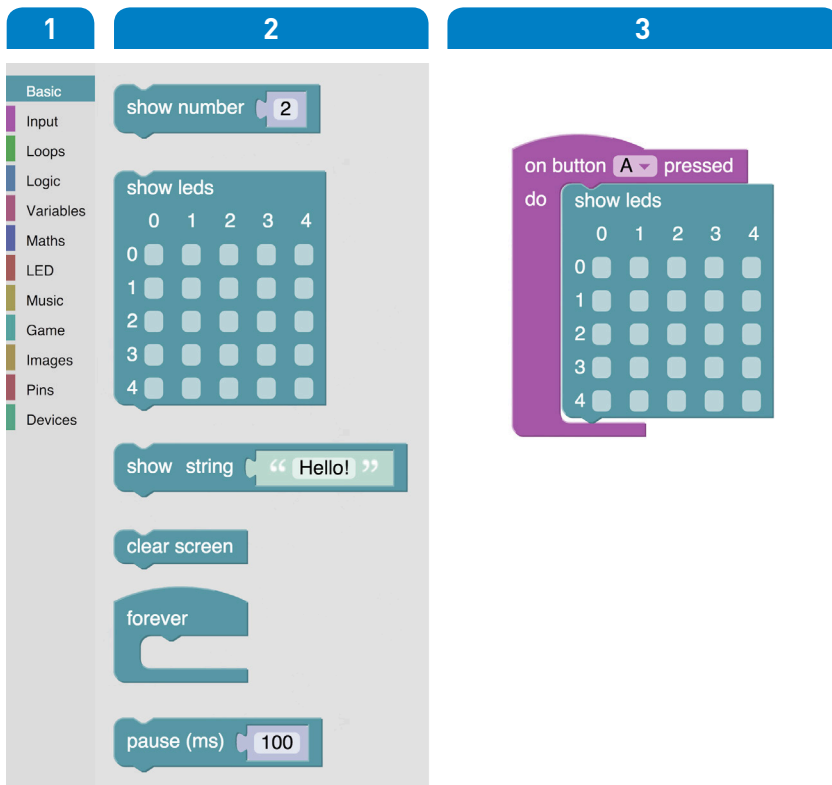


All of the experiments in this guide are based around the **Microsoft Block** and **Microsoft Touch Develop** editors. The Microsoft Block Editor is a very easy to use graphical editor. The Microsoft Touch Develop editor is a text based programming language which is ideal for slightly more complex programs. It is possible to convert a Block program into a Touch Develop script. This offers an easy way of progressing from Block programming to Touch Develop scripts.

Other editor options include JavaScript and Python. Refer to www.kitronik.co.uk/microbit for tutorials based on these.

USING THE MICROSOFT BLOCK EDITOR

The Microsoft Block Editor is a drag and drop visual editor that provides a simple introduction to programming. Blocks snap together to build programs and are grouped by the type of function they do. When a group is selected the commands in that group are shown and can be selected.



- 1** Select a Block category from the list on the left side of the page.
- 2** Select a Block from this category and drag it to the workspace area to the right.
- 3** Snap new Blocks onto existing Blocks in the workspace area. As new Blocks are dragged into the workspace the editor highlights when they are in a valid position to snap to existing Blocks.

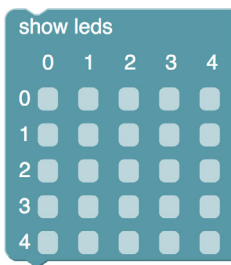
BLOCK COMMANDS

The following pages describe the Block commands that are used in experiments one to four in this booklet. Details of all of the available Microsoft Block Editor commands can be found at www.kitronik.co.uk/microbitblocks.

LOOP & DISPLAY BLOCKS



The **FOREVER** Block (**BASIC**) is one of the most commonly used Block commands. All of the commands that are placed in a forever loop are run in order and then repeated over and over **forever!**



The **SHOW LEDS** Block (**BASIC**) is used to display an image on the LED display. If a box is ticked then the corresponding LED will be lit up.



The **SHOW STRING** Block (**BASIC**) is used to show a string of characters on the LED display. Text will scroll across the display (from left to right) until all the characters have been shown.



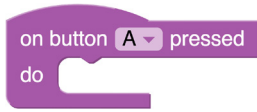
The **WHILE DO** Block (**LOOPS**) will **repeatedly** run the commands contained within it while the test condition is true. The test condition Block snaps to the right of the 'while' text. Once this condition is met the program will continue on to the commands below this Block.



The **IF DO ELSE** Block (**LOGIC**) tests the condition snapped to the right of the 'if' text. If the condition is 'true' it will execute once the Blocks snapped to the right of the 'do' text. If 'false' it will execute once the Blocks snapped to the right of the 'else' text.

The 'cog' wheel (when clicked on) lets extra 'else if' and 'else' elements to be added to this command. This allows for different commands to be run if different test conditions are met.

INPUT & OUTPUT BLOCKS



Commands placed within in the **ON BUTTON [A*] PRESSED DO** Block (**INPUT**) will run whenever the selected button or buttons are pressed.



Commands placed within in the **ON PIN [P0*] PRESSED DO** Block (**INPUT**) will run when the user holds the GND pin with one hand and presses the selected edge connector pin with the other hand.



The **DIGITAL WRITE (0,1) TO PIN [P0*]** Block (**PINS**) writes either a '0' (a low voltage) or a '1' (a high voltage) to pin P0, P1 or P2. To change the value that is written to the pin, select the box that has a '1' in it and edit it to '1' or '0' as required.



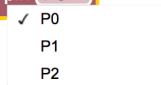
The **ANALOG READ PIN [P0*]** Block (**PINS**) reads the analog (variable) voltage on pin P0, P1 or P2 (depending on which is selected) and returns a number between 0 and 1024 depending on the level of the voltage present on that pin.



The **ANALOG WRITE [1023] TO PIN [P0*]** Block (**PINS**) writes an analog value (variable voltage) to pin P0, P1 or P2 (depending on which is selected). The voltage on that pin will be set to a level defined by the size of the number written, where 0 = 0V and 1023 = 3.3V.



The **DIGITAL READ PIN (0,1) [P0*]** Block (**PINS**) reads the digital voltage on pin P0,P1 or P2 and returns a '0' if the it is a 'low' voltage or a '1' if it is a 'high' voltage. (A 'low' voltage is anything below 0.8 Volts. A 'high' voltage is anything above 1.6 Volts).



*SELECTING THE PIN/BUTTON

The 'drop down' arrow next the default pin 'P0' or button 'A' allows the pin/button which the command writes/reads to be changed.

LOGIC, MATH & VARIABLES BLOCKS



The **COMPARISON OPERATORS** Block (**LOGIC**) is used to compare two items. For example to see if one item is bigger, smaller, equal to another etc... It returns a Boolean value (true or false).

This Block has spaces to place the variable or values that you want to compare. The drop down box in the centre allows the selection of the operator type (there are six you can select).



The **NUMERIC VALUE** Block (**MATHS**) is used to enter numeric values. These are Integers (whole numbers). The value defaults to '0' but can be changed. This Block is often used in Blocks such as the Comparison operator described above.



The **ARITHMETIC OPERATION** Block (**MATHS**) is used to perform an arithmetic (maths) operation on two items. For example by adding them together. It returns the result of this operation.

This Block has spaces to place the variable or values to perform the operation on. The drop down box in the centre allows the selection of the operator type (there are five you can select).



The **ITEM** Block (**VARIABLES**) gets a variables value. The drop down menu allows the selection of the desired variable, or to create a new variable.

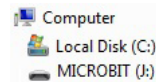


The **SET ITEM TO** Block (**VARIABLES**) assigns the value of the selected variable to a value. This value could be a fixed numeric value or that of another variable.

GETTING A PROGRAM ON TO THE BBC MICRO:BIT

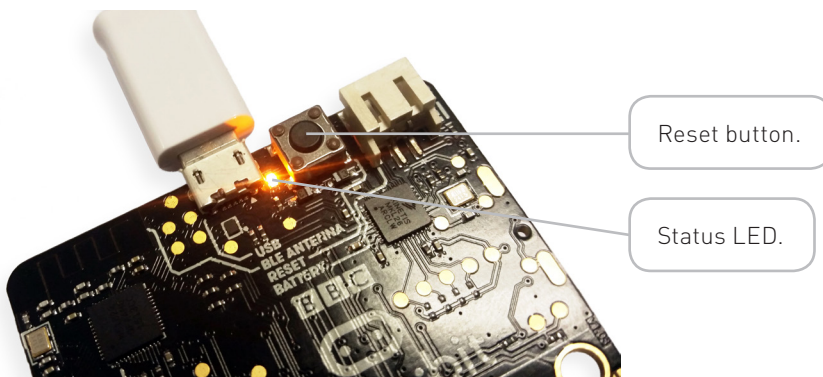
It is very easy to transfer a finished program to the BBC micro:bit. First of all select 'compile' (🔗). This is where the program is converted into a format that the BBC micro:bit can understand. This is known as a '.hex' file. If it has compiled successfully it will return the message 'Your .hex file is ready. Drag and drop onto your BBC Micro:bit device drive.' The message 'Do you want to open or save microbit-script.hex from microbit.co.uk?' will appear. Select 'SaveAs' from the 'Save' drop down menu and save the .hex file to a folder for BBC micro:bit .hex files.

Next plug a BBC micro:bit into the computer via the USB Cable. The BBC micro:bit will appear as a removable drive on the computer something like this.



To download the .hex file to the BBC micro:bit 'Drag' the .hex file from the folder where it was saved and 'Drop' it onto the MICROBIT removable drive. A message will appear saying 'Copying 1 item.....to MICROBIT'. At the same time the yellow LED on the back of the BBC micro:bit will flash.

After a few seconds the download will complete and the BBC micro:bit should now be running the program.



If it doesn't you may need to press the reset button, which is next to the status LED.

1

EXPERIMENT ONE

SAY "HELLO" TO THE BBC micro:bit!

THE AIMS OF THIS EXPERIMENT ARE...

- To learn how to create and display an image on the BBC micro:bit's LED matrix using Microsoft Block Editor.
- To learn how to use the 'Show String' command.
- To learn how to use the BBC micro:bit's A and B buttons as inputs.
- To compile the program to generate a .hex file.
- To upload the .hex file onto the BBC micro:bit.

THE EXPERIMENT

In this experiment we will use the buttons on the BBC micro:bit and show how we can use the breakout board to access these buttons externally using the supplied Push Switches.

PARTS USED



PUSH SWITCH x2



MALE-TO-FEMALE JUMPER WIRE x4

THE PROGRAM WILL NEED THESE BLOCKS...

BASIC

SHOW LEDs

SHOW STRING

INPUT

ON BUTTON A PRESSED

ON BUTTON B PRESSED

USE THE BLOCKS DETAILED ABOVE TO CREATE THE CODE BELOW

Create Code >

To start programming on www.microbit.co.uk select 'Create Code'.



Select 'Microsoft Block Editor' then drag and drop the Blocks below into the editor window.

on button **A** pressed

do

show leds	0	1	2	3	4
0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Click on the LEDs to illuminate to make a smiley face.

Select Button 'B'.

on button **B** pressed

do

show string "Hello world"

Type a message here.



Run the program on the simulator. Click on the 'A' button on the simulator to see the LED pattern. Click on the 'B' button to see the message.



Compile the program to generate and the .hex file. Save the .hex file ready to upload to the BBC micro:bit.

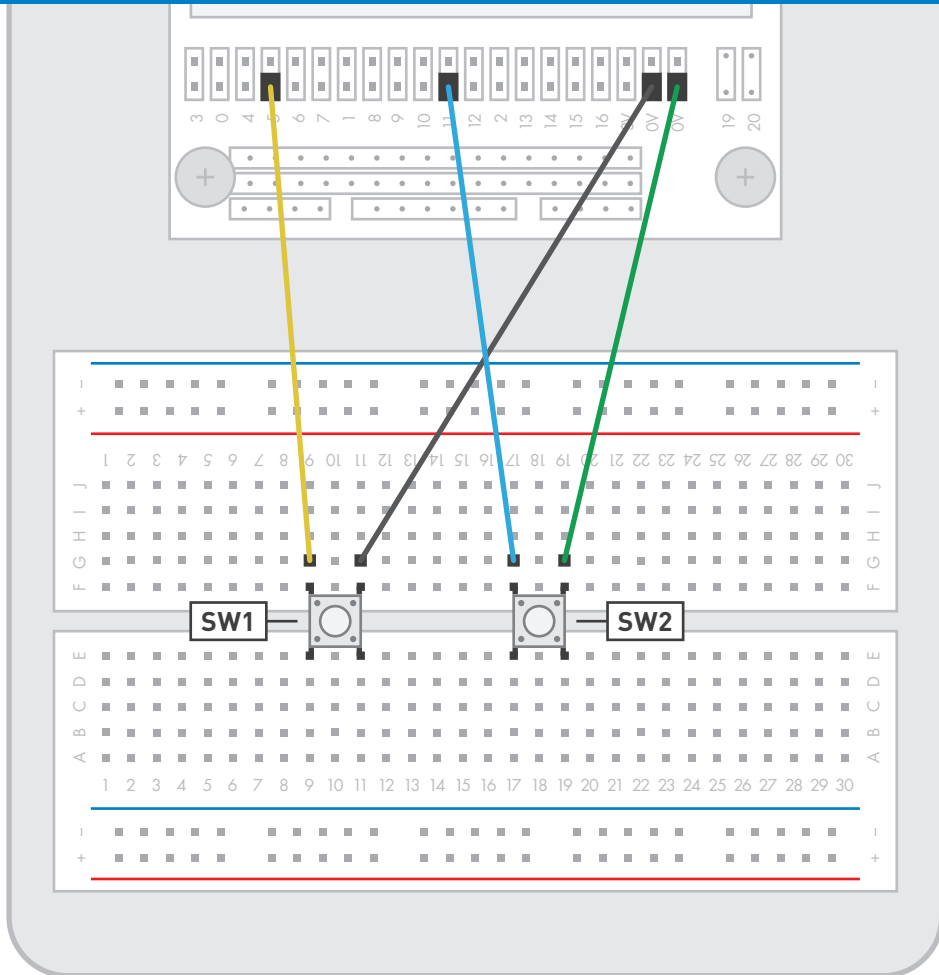


Plug the BBC micro:bit into a USB port then drag and drop the .hex file onto the BBC micro:bit window to upload the program.



HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

BUILD THIS CIRCUIT ON THE BREADBOARD

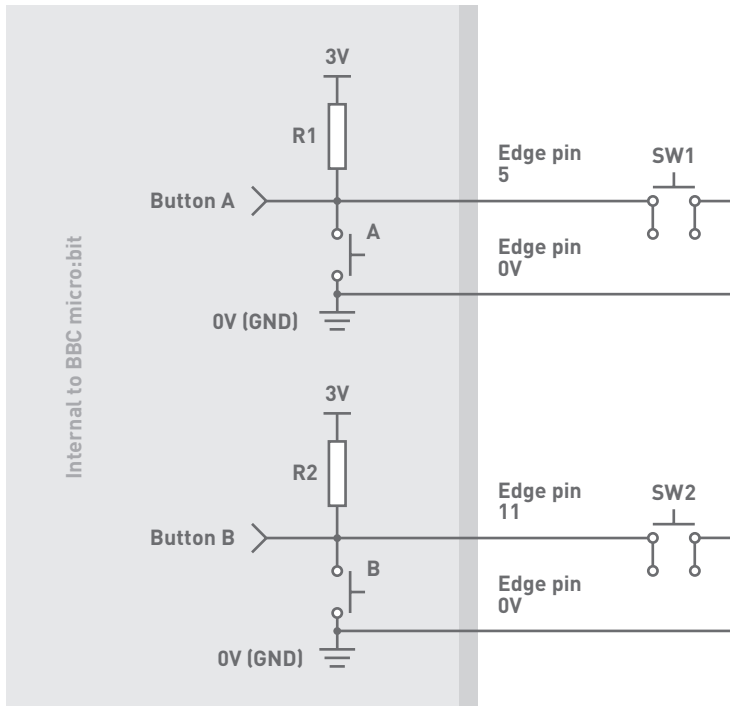


Note: The BBC micro:bit must be plugged into a power supply to work.

WHAT WILL HAPPEN?

Press the 'SW1' button on the BBC micro:bit and the smiley face should be displayed on the LED matrix. Press the 'SW2' button and the text 'Hello world' should be displayed instead.

CIRCUIT DIAGRAM



WHAT'S GOING ON?

The BBC micro:bit will respond to having push switch SW1 pressed by lighting up the LEDs selected in the 'show LEDs' Block. If push switch SW2 is pressed then the text in the 'show string' Block will scroll across the screen.

The switches on the breadboard are connected (in parallel) to the buttons on the BBC micro:bit through the breakout board. This means when the switches on the breadboard are pressed the BBC micro:bit thinks buttons A and B are being pressed. As far as the BBC micro:bit is concerned there is no difference, all it detects is that the input A goes to 0V when one switch is pressed and input B goes to 0V when the other switch is pressed.

2

EXPERIMENT TWO

USING AN LDR & ANALOG INPUTS

THE AIMS OF THIS EXPERIMENT ARE...

- To use a Light Dependant Resistor as a sensor.
- To perform an analog reading from the Light Dependant Resistor via input pin P0.
- To set a light threshold to decide whether to display a 'sun' or a 'moon' on the LED matrix.

THE EXPERIMENT

An LDR (Light Dependant Resistor) is an electrical component with unique properties. As its name suggests, it is a type of resistor that has its resistance determined by how much light is shining on it. The brighter the light the less resistance it has. These resistors are used along with normal resistors to form potential dividers. When used in this configuration it gives a voltage that changes depending on the light level. A microcontroller such as the BBC micro:bit can read this (analog) voltage allowing a program to react to different light levels. This experiment will explain how to use the LDR and take an analog reading.

PARTS USED



LDR x1



10kΩ RESISTOR x1



MALE-TO-FEMALE JUMPER WIRE x3

THE PROGRAM WILL NEED THESE BLOCKS...

BASIC

FOREVER

SHOW LEDS

LOGIC

IF DO ELSE

[] = []

MATH

0

PINS

ANALOG READ PINS

VARIABLES

SET ITEM TO

ITEM

USE THE BLOCKS DETAILED ABOVE TO CREATE THE CODE BELOW

Click on the variable
to select the one used
or to add new ones.

forever

- set **Light** to analog read pin P0
- if **Light** > 512
 - do
 - show leds

	0	1	2	3	4
0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
 - else
 - show leds

	0	1	2	3	4
0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



Run



Compile

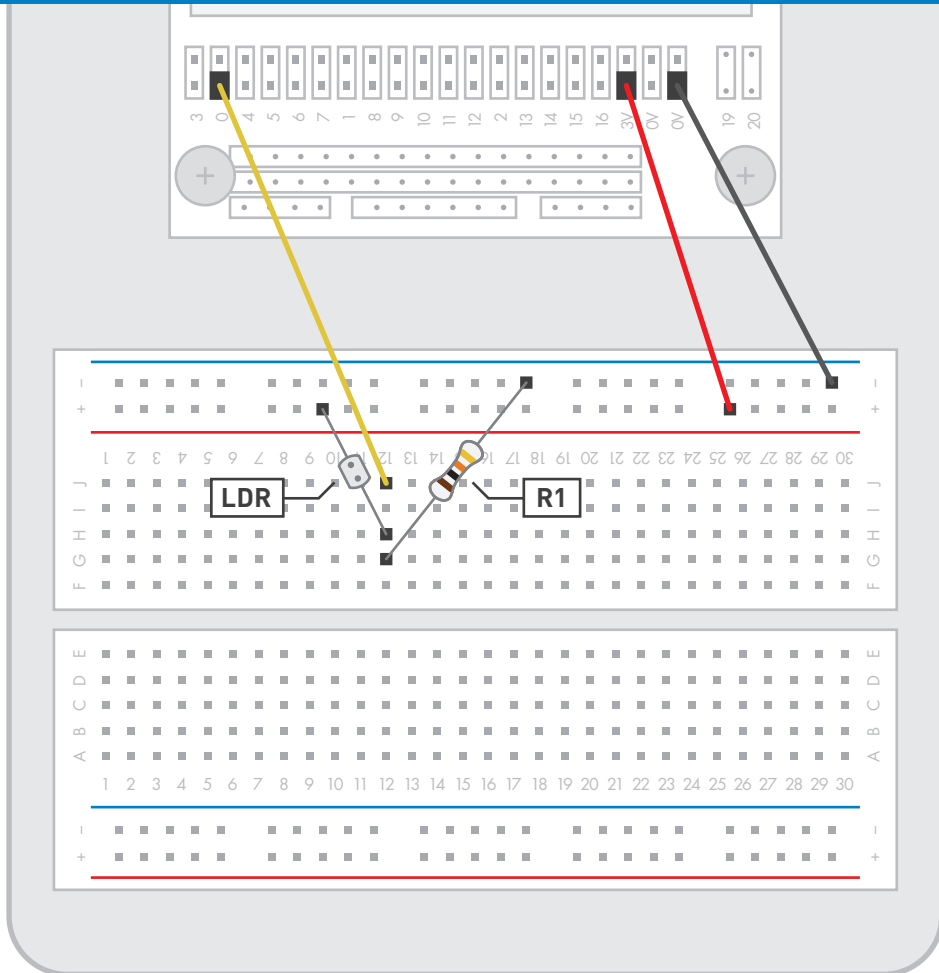


Upload



HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

BUILD THIS CIRCUIT ON THE BREADBOARD

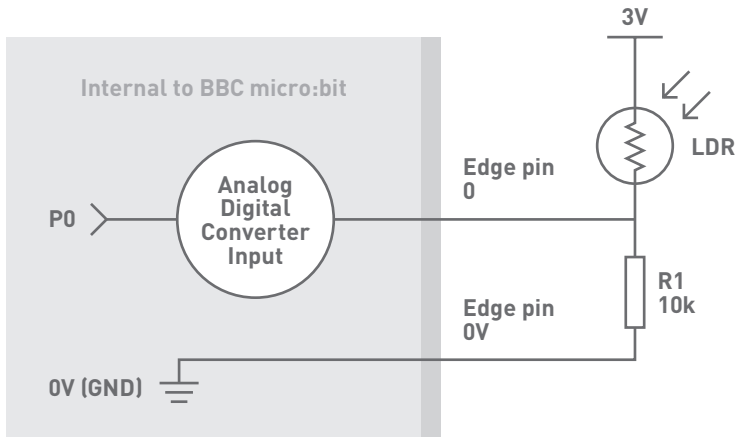


Note: 10kΩ resistor is marked with coloured bands: brown, black, orange, gold.

WHAT WILL HAPPEN?

The BBC micro:bit should display a sun image. If you cover the LDR with your finger it should show a moon image. When the LDR is in light the BBC micro:bit should display the 'sun' image. When the LDR is in dark (by covering it) the BBC micro:bit should display the 'moon' image.

CIRCUIT DIAGRAM



WHAT'S GOING ON?

The resistor R1 and LDR are in series so they divide the 3V. The proportion of the 3V across each of them depends on their relative resistance. If they both have equal resistance then the voltage in the middle will be half. If the resistor makes up 75% of the total resistance then the voltage in the middle will be 75% of the 3V and so on. As more light shines on the LDR its resistance goes down meaning resistor R1 makes up a greater percentage of the total resistance. This increases the voltage on P0. The analog read function measures the voltage on the selected pin and converts it into a value between 0 and 1023. If the voltage on pin P0 is 0V then this value will be 0. If the voltage on this pin is the maximum of 3V then this value will be 1023. If the voltage on pin P0 is 1.5V the value will be approximately 512.

The pin P0 is read and the value is stored in the variable 'light'. The 'if' statement then checks if the value stored in 'light' is greater than 512. If it is then the LEDs in the pattern of a sun is displayed on the LED matrix, otherwise the moon pattern is displayed.

3

EXPERIMENT THREE

DIMMING AN LED USING A POTENTIOMETER

THE AIMS OF THIS EXPERIMENT ARE...

- To use a push switch to turn an external LED 'on'.
- To use a potentiometer as a potential divider.
- To read the analog voltage from the potentiometer and use it to set the brightness of the external LED.

THE EXPERIMENT

This experiment will use a 'variable' to track whether an LED should be turned on or off when a push switch is pressed. A potentiometer will be used to alter the brightness of the LED when it is on. This is achieved by two separate loops of code that run at the same time. One remembers the state of the switch (on or off). The other outputs a signal to the LED based on the voltage it reads from the potentiometer. The higher the voltage from the potentiometer the brighter the LED shines. It does this using Pulse Width Modulation (PWM) and the later experiments go into more detail about how these signals work.

PARTS USED



POTENTIOMETER x1



M/F JUMPER WIRE x7



PUSH SWITCH x1



47Ω RESISTOR x1



RED 5MM LED x1

THE PROGRAM WILL NEED THESE BLOCKS...

BASIC	FOREVER		
INPUT	ON PIN PRESSED		
LOGIC	IF DO ELSE	[] = []	
MATH	0		
PINS	ANALOG WRITE TO PIN	ANALOG READ PIN	DIGITAL WRITE TO PIN
VARIABLES	SET ITEM TO	ITEM	

USE THE BLOCKS DETAILED ABOVE TO CREATE THE CODE BELOW

```
on pin P0 pressed
do
  if Light State = 0
  do
    set Light State to 1
  else
    set Light State to 0

forever
  if Light State = 1
  do
    analog write (analog read pin P1) to pin P2
  else
    digital write (0,1) 0 to pin P2
```



Run



Compile

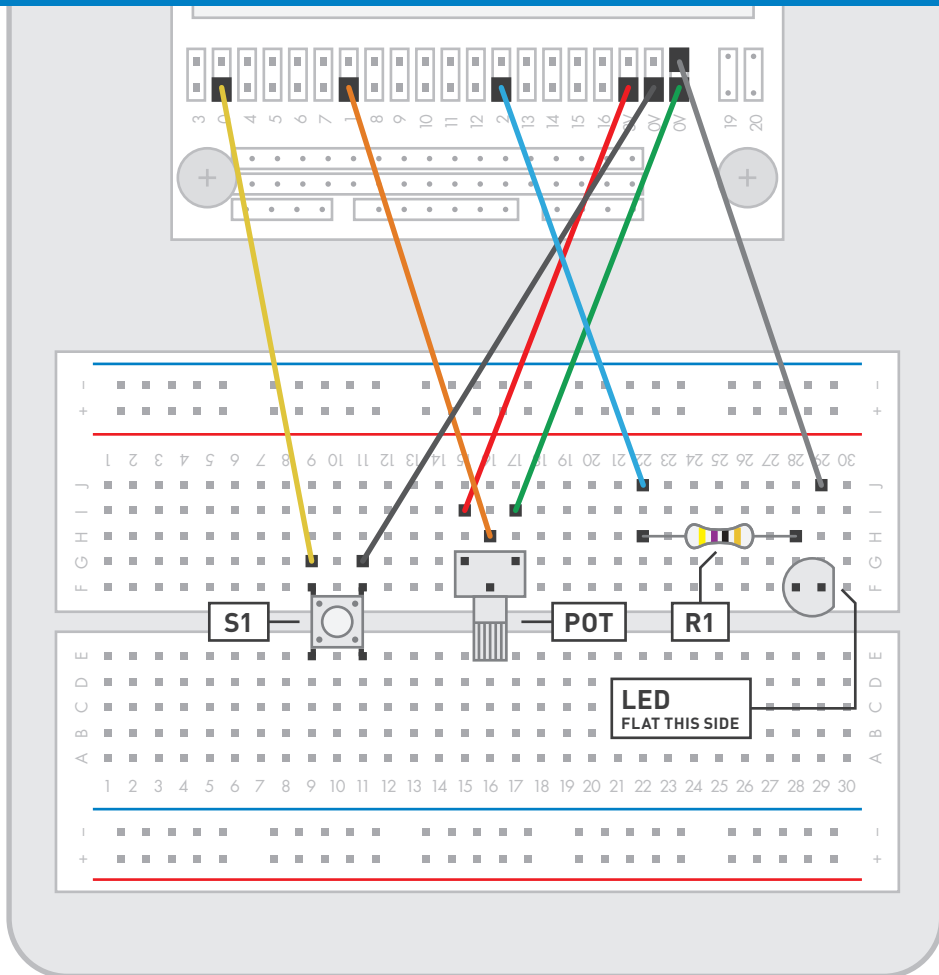


Upload



HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

BUILD THIS CIRCUIT ON THE BREADBOARD

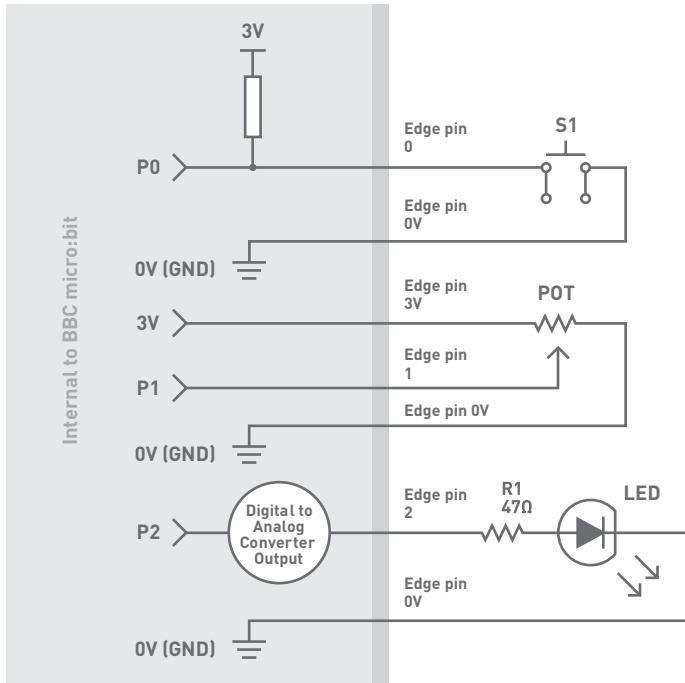


Note: 47 Ω resistor is marked with coloured bands: yellow, purple, black, gold.

WHAT WILL HAPPEN?

Turn the potentiometer all the way anti-clockwise and press the switch once. The LED should come on. Now turn the potentiometer clockwise and the LED should dim. Pressing the switch again will turn the LED off.

CIRCUIT DIAGRAM



WHAT'S GOING ON?

The 'on P0 pin pressed' loop is activated if the push switch is pressed and toggles the value of the 'light state' variable between 1 and 0. The use of a variable here lets the same switch be used as both an on and off switch by remembering the previous state (on or off) of the system.

The forever loop constantly runs and if the 'light state' is set to 1 'on' then it takes an analog reading from the potentiometer connected to P1 and writes that value to analog output P2, driving the LED. This analog value determines the brightness of the LED.

If 'light state' is set to 0 then it writes a '0' to the LED, turning it off.

4

EXPERIMENT FOUR

USING A TRANSISTOR TO DRIVE A MOTOR

THE AIMS OF THIS EXPERIMENT ARE...

- To use a transistor to drive a fan motor.
- To control the speed of the motor using Pulse Width Modulation (PWM).

THE EXPERIMENT

The output pins on most microprocessors can only supply a small amount of current, not enough for a power hungry device such as a motor. The BBC micro:bit is no exception to this. A transistor can be used to solve this problem.

A transistor is like a gate for electricity, a small amount of current can be used to open the gate to let a lot of current flow through to power hungry components.

PARTS USED



TRANSISTOR x1



MOTOR x1



2.2kΩ RESISTOR x1



M/F JUMPER WIRE x3



TERMINAL CONNECTOR x1



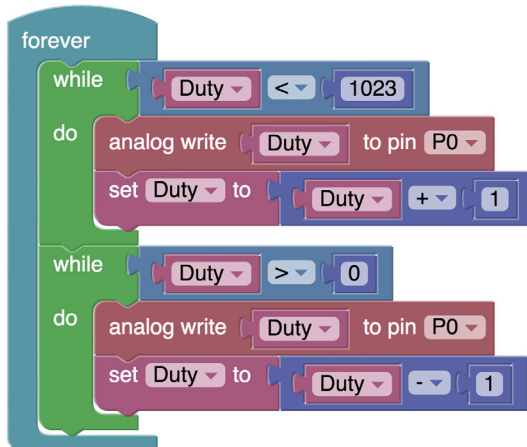
FAN BLADE x1

THE PROGRAM WILL NEED THESE BLOCKS...

BASIC	FOREVER
LOGIC	[] = []
LOOPS	WHILE DO
MATH	[] + [] [] 0
PINS	ANALOG WRITE TO PIN
VARIABLES	SET ITEM TO [] []

USE THE BLOCKS DETAILED ABOVE TO CREATE THE CODE BELOW

set Duty to 0



Note: A 20ms delay is built into each loop of the While Do Block.



Run



Compile

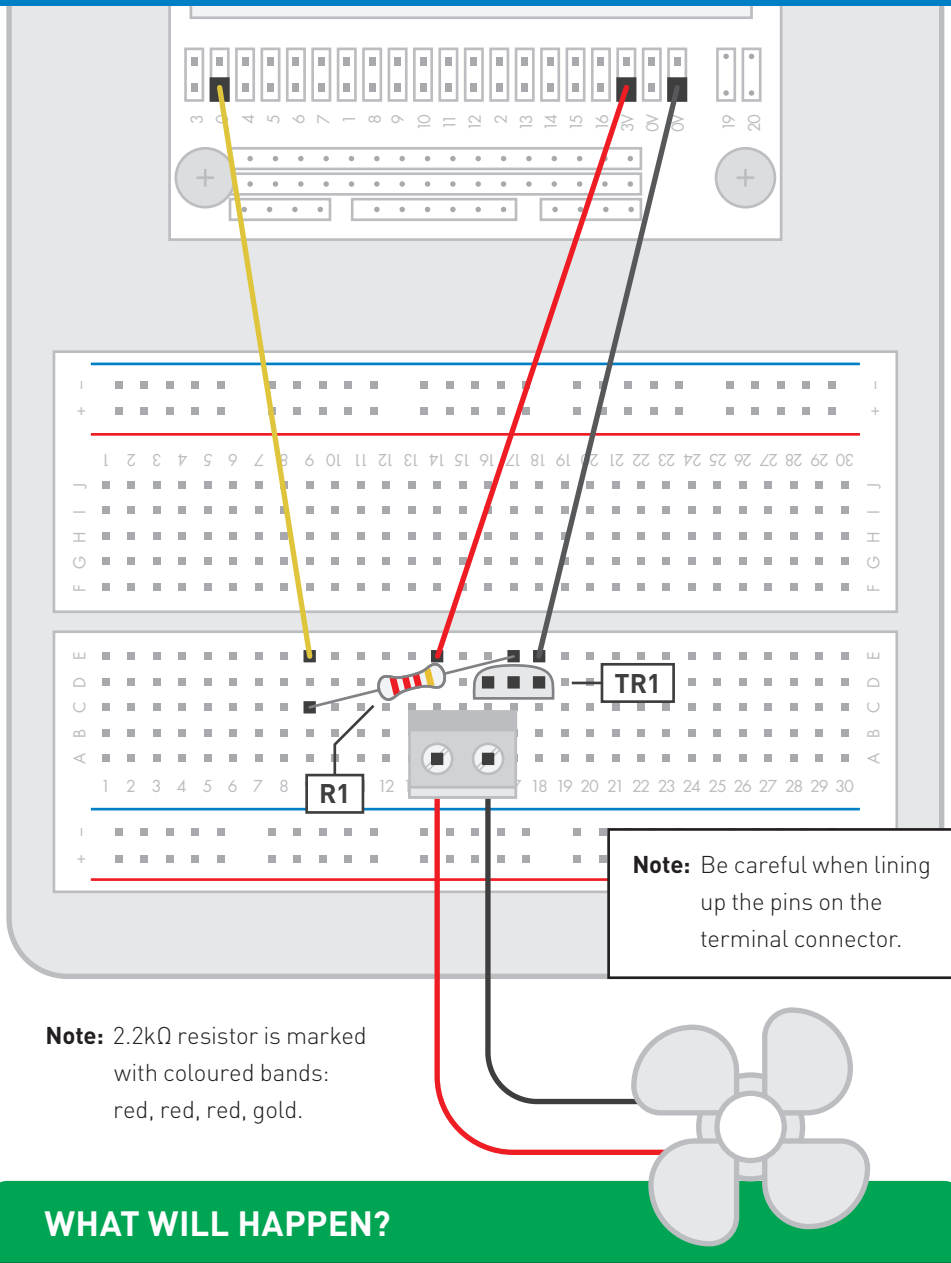


Upload



HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

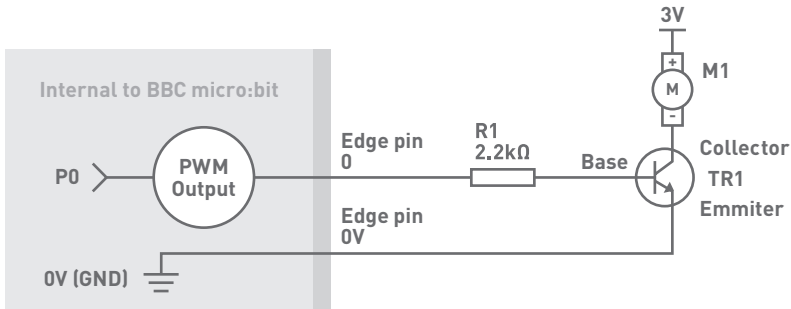
BUILD THIS CIRCUIT ON THE BREADBOARD



WHAT WILL HAPPEN?

The motor should, after a few seconds start to spin slowly, then faster until it reaches a maximum speed. At this point it will slow down and stop and the cycle will begin again.

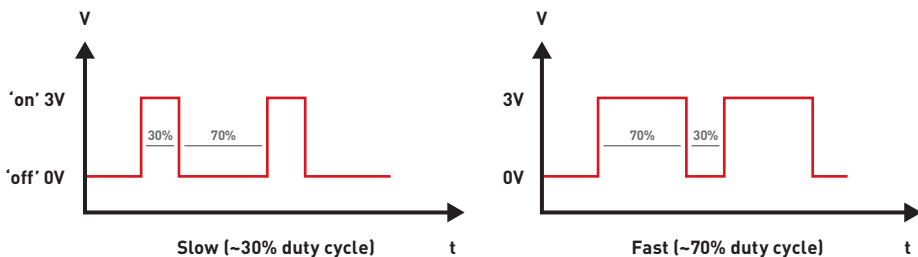
CIRCUIT DIAGRAM



WHAT'S GOING ON?

If the transistor is turned on and off very quickly and it spends half its time on and half its time off then the fan motor will spin at half speed. By changing the percentage of the time that the transistor spends on (known as the duty cycle) the speed of the motor can be finely controlled. As the power pulses on and off and the width of the pulses are controlled, we call this process Pulse Width Modulation (PWM).

The code works in two stages. The first loop writes the duty value to Pin P0 then increases the duty value by one and writes the value to P0 again. This repeats until the value reaches the maximum of 1023 (full speed). The second loop then kicks in and reduces the duty by 1 and writes it to P0 until the value reaches 0 (stopped). This whole cycle is inside a forever loop so the motor will speed up and slow down forever. The PWM output varies the duty cycle of the output voltage as shown below to vary the speed of the fan motor.



USING THE MICROSOFT TOUCH DEVELOP EDITOR

Microsoft Touch Develop has been designed for mobile devices with touchscreens but it can also be used with a PC by using a keyboard and mouse. This editor introduces a statically-typed scripting language with syntax-directed editor.



Any Microsoft Block Editor script, like those covered in the earlier experiments, can be converted into a Microsoft Touch Develop script with a single press of a button.

Code can be created by selecting commands from the section at the bottom of the screen.

```
script supernatural script 4
function main ()
(
end function
```

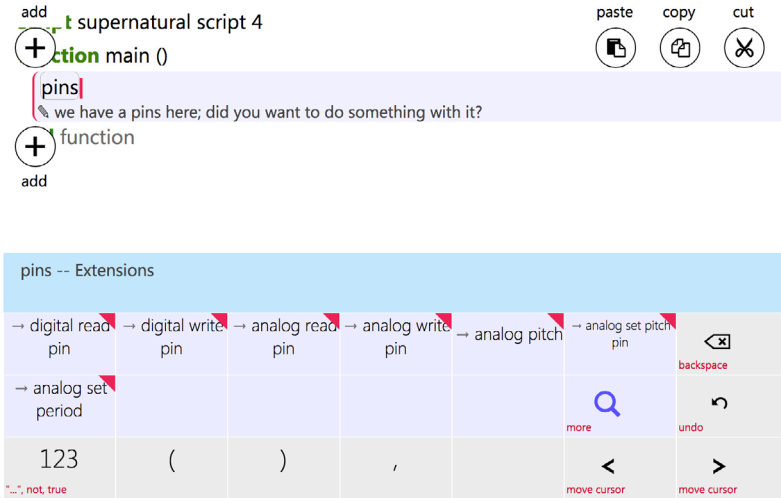
paste copy cut
select

var <small>new variable</small>	if <small>conditional</small>	for <small>repeat n times</small>	while <small>repeat while</small>			more
basic	led	input	math	image	pins	backspace
music	game	events	control	bits	more 1/2	undo
123 <small>"...", not, true</small>	()	,		move cursor	move cursor

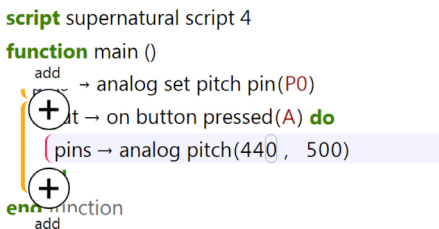
Commands can be selected from this area.

CREATING A TOUCH DEVELOP SCRIPT

When a command is selected the next option (if needed) for that command is presented.



Commands are entered at the current line that this cursor is on. It is also possible to add commands before or after this line. This is done by selecting the relevant '+' circle (as shown below). This will add a new line at this position, which can then be used to add the required commands.



Compiling and uploading of the script is done in the same way as it is when using the Microsoft Block Editor.

More information on Microsoft Touch Develop can be found at www.kitronik.co.uk/touchdevelop.

5

EXPERIMENT FIVE

USING THE ACCELEROMETER TO CONTROL MOTOR SPEED

THE AIMS OF THIS EXPERIMENT ARE...

- To understand how the accelerometer works and how to interpret and use its readings.
- To use the accelerometer and PWM to control the speed of a fan motor by tilting the BBC micro:bit.

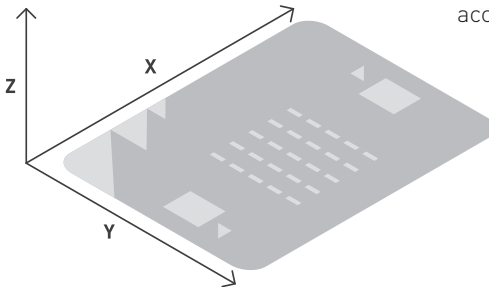
THE EXPERIMENT

The BBC micro:bit has a built in accelerometer. The accelerometer detects if it is speeding up or slowing down in three directions labelled below as X, Y and Z. Each axis can measure between -4096 and 4096 which correlate with -4G and +4G respectively.

A G is an acceleration of 9.81 metres per second per second; this is the acceleration that objects under the influence of earth's gravity experience.

If the values of X, Y and Z are read when the BBC micro:bit is placed horizontally on a table with the display facing upwards the values read will be X=0, Y=0 and Z= -1023. This is because there is no acceleration in the X or Y directions but in the Z direction it is experiencing the force of the table pushing upward on it resisting gravity's pull. The reason this number is negative

rather than positive is because the accelerometer is mounted on the bottom of the BBC micro:bit, so it is upside down when the display is facing upwards.



PARTS USED



TRANSISTOR x1



MOTOR x1



2.2kΩ RESISTOR x1



M/F JUMPER WIRE x3



TERMINAL CONNECTOR x1



FAN BLADE x1

CREATE THE FOLLOWING TOUCH DEVELOP SCRIPT



Microsoft Touch Develop Editor

script Mapping a Tilt to a Speed

The name of the script can be edited.

function main ()

basic → forever **do**

{ Tilt := math → clamp(0, 1023, input → acceleration(y))

{ pins → analog write pin(P0, Tilt)

end

end function

Use the Microsoft Touch Develop commands to enter the lines of code.



Run



Compile

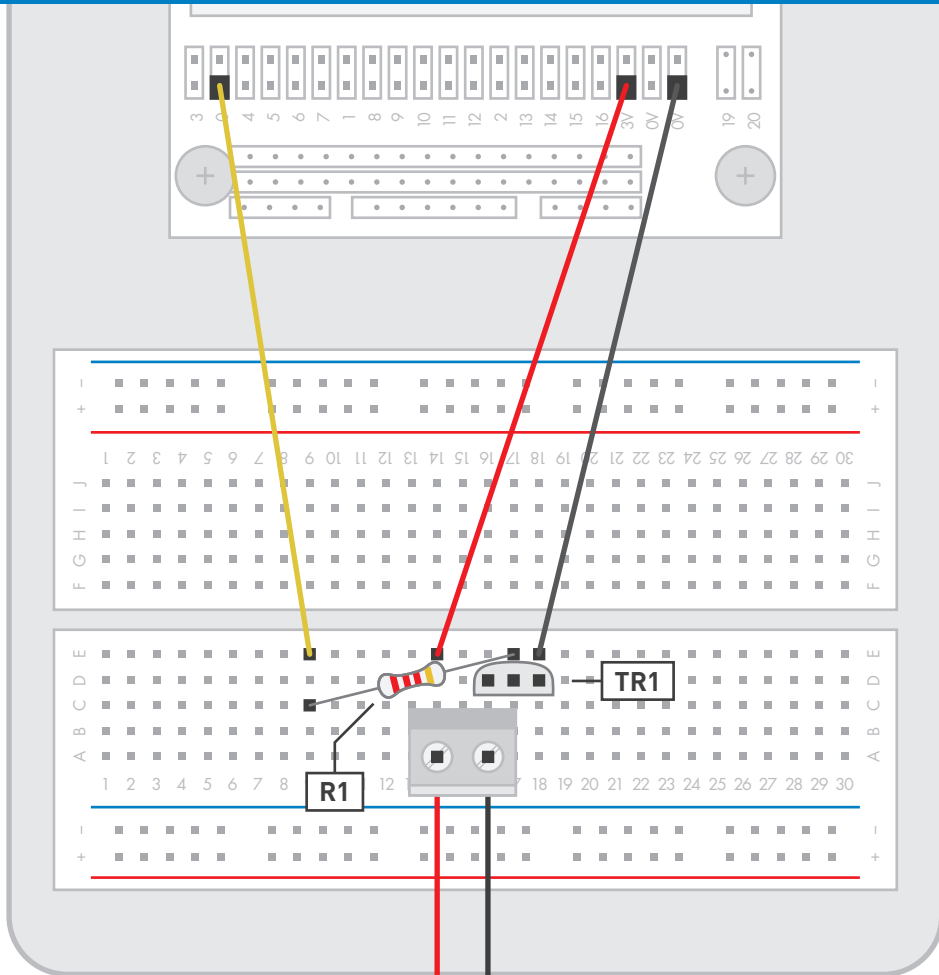


Upload

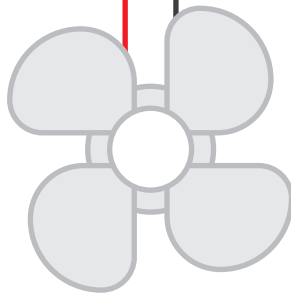


HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

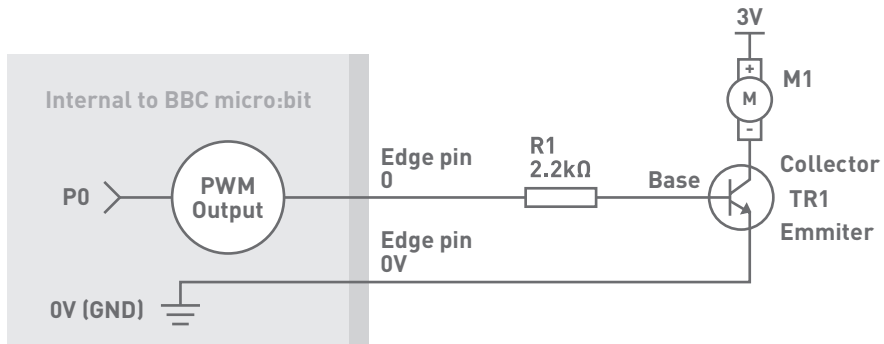
BUILD THIS CIRCUIT ON THE BREADBOARD



Note: This is the same circuit used in Experiment 4.



CIRCUIT DIAGRAM



WHAT WILL HAPPEN?

The speed of the fan motor will match the tilt of the BBC micro:bit. If the BBC micro:bit is flat the fan will not spin, if tilted upright so that the edge connector faces downwards the fan will spin at maximum speed. If slowly tilted between flat and upright the speed should vary.

WHAT'S GOING ON?

The Touch Editor code uses the 'Tilt' function to determine if the BBC micro:bit is being tilted and outputs a value to set the fan motor speed. The valid values for an analog PWM output are 0 to 1023. Tilting the BBC micro:bit forward or back will give us a value between -1023 and 1023 on the Y axis, depending on how much it is tilted. Tilted back vertically will give $Y = -1023$. Tilted forward will give $Y = 1023$.

So that an invalid value is not written to the analog pin P0, the 'clamp' function is used. The 'clamp' function locks an input value between two user specified numbers. The program uses 0 and 1023 as the clamp values and 'input acceleration (Y)' as the input for the clamp function. So if Y is less than 0 the clamp function will return a value of 0 rather than a negative value. The clamped value is then used for the analog write to pin P0.

6

EXPERIMENT SIX

SETTING THE TONE WITH A PIEZO BUZZER

THE AIMS OF THIS EXPERIMENT ARE...

- To use Microsoft Touch Develop to access more functions of the BBC micro:bit.
- To learn how to control the tone of a piezo buzzer.

THE EXPERIMENT

The Piezo Element Buzzer has to be driven with a signal that alternates in voltage (from 0V to 3V) at the desired frequency of the tone that wants to be produced. This experiment will explain how the BBC micro:bit can do this and be used to play a tone.

PARTS USED



PIEZO ELEMENT BUZZER x1



MALE-TO-FEMALE JUMPER WIRE x2

CREATE THE FOLLOWING TOUCH DEVELOP SCRIPT

script Setting the Tone

function main ()

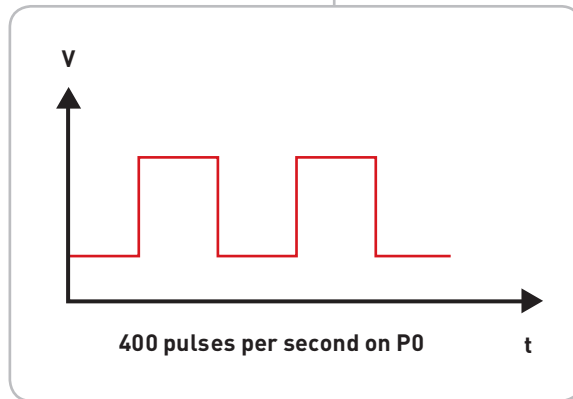
{ pins → analog set pitch pin(P0)

{ input → on button pressed(A) **do**

{ pins → analog pitch(400, 500)

end

end function



Run



Compile

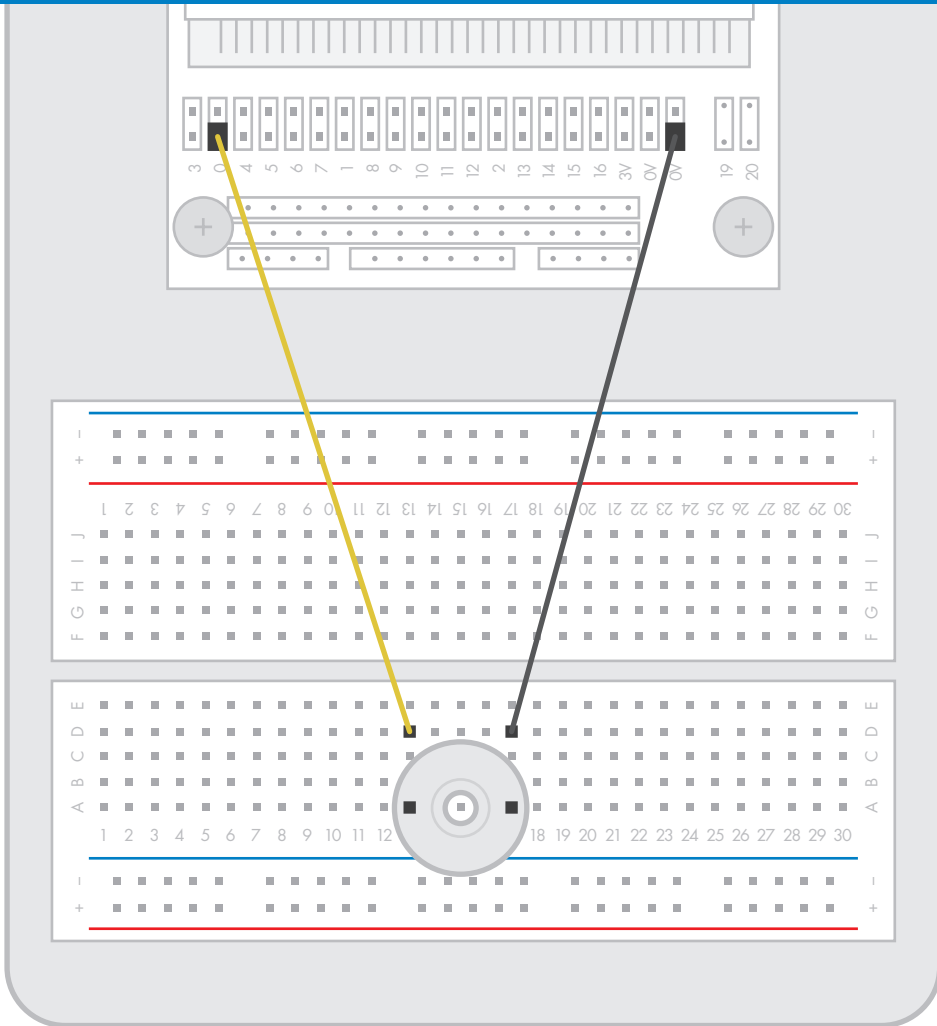


Upload



HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

BUILD THIS CIRCUIT ON THE BREADBOARD

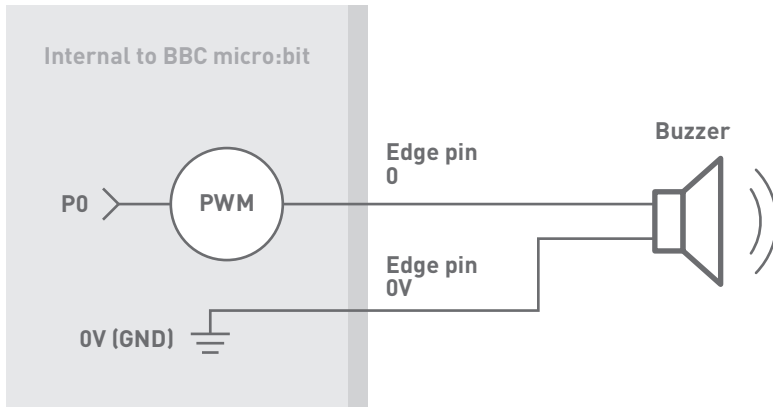


WHAT WILL HAPPEN?

Press the 'A' button on the BBC micro:bit, the buzzer should make a brief noise.

Note: This only happens on the release of the button.

CIRCUIT DIAGRAM



WHAT'S GOING ON?

The first line selects pin P0 as the analog pitch pin output, this is the pin used to drive the piezoelectric buzzer. The second line looks for button A on the BBC micro:bit being pressed. The third line (executed if button A is released) provides two numbers to the BBC micro:bit what to output to the pitch pin. The first number (400) is the frequency. This means that in one second the pin will go from low to high 400 times. A higher frequency will give a higher note and a lower frequency will give a lower note.

The second number is the duration that the note will play for in milliseconds. 500 milliseconds means the note will play for half a second when button A is pressed.

7

EXPERIMENT SEVEN

WIND POWER

THE AIMS OF THIS EXPERIMENT ARE...

- To generate a voltage by blowing on a fan blade to spin a motor.
- To measure this voltage by using an analog input pin on the BBC micro:bit.

THE EXPERIMENT

Wind turbines convert the kinetic energy of air into electrical power. In this experiment the motor and fan are used in reverse to generate a voltage which can be measured and displayed by the BBC micro:bit.

PARTS USED



TERMINAL CONNECTOR x1



MOTOR x1



22kΩ RESISTOR x2



FAN BLADE x1



MALE-TO-FEMALE JUMPER WIRE x2



MALE-TO-MALE JUMPER WIRE x1

CREATE THE FOLLOWING TOUCH DEVELOP SCRIPT

```
script Wind turbine
function main ()
  basic → forever do
    { var value := pins → analog read pin(P0)
      { if value > □ Highest then
        { □ Highest := value
          { else add code here end if
        }
      }
    } end
  input → on button pressed(A) do
    { basic → show number(□ Highest, 150)
    }
  } end
end function
```



Run



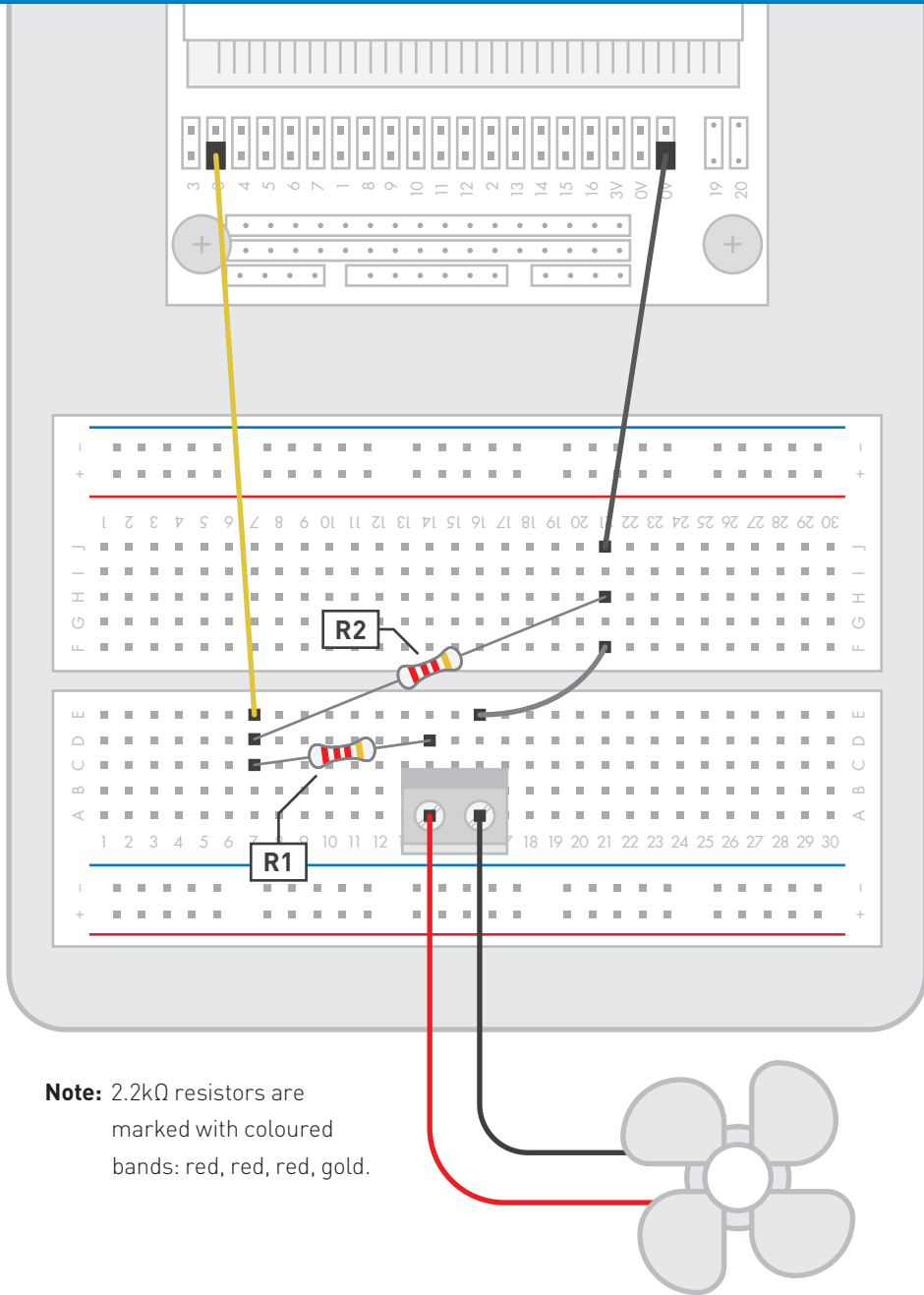
Compile



Upload

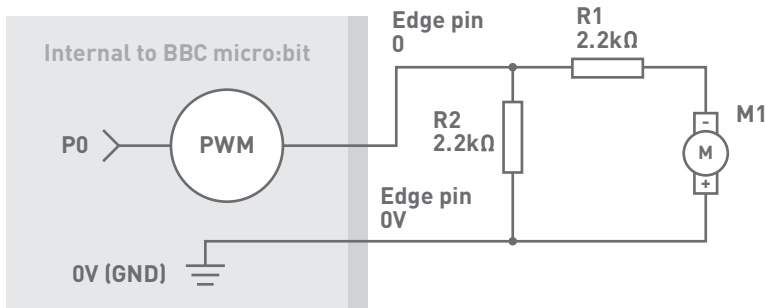


BUILD THIS CIRCUIT ON THE BREADBOARD



Note: 2.2k Ω resistors are marked with coloured bands: red, red, red, gold.

CIRCUIT DIAGRAM



WHAT WILL HAPPEN?

Once the code is uploaded to the BBC micro:bit press button A and it should display a low number in the range of 1-5. This is the voltage read from P0. The reason it does not display 0 in this configuration is because P0 is very sensitive so will likely pick up some small voltage. Try blowing gently on the fan until it just begins to spin then press the button A again. This time it should display a reading of about 200-300. Finally try taking a deep breath and blowing hard on the fan. This should give a reading close to 1000!

WHAT'S GOING ON?

When electric current passes through a wire inside the magnetic field it produces torque (turning force) that turns a motor. The opposite is also true, if torque is applied to a motor current is generated. The torque is supplied by blowing on the fan. Although the current generated by blowing on the fan is very small, the voltage can reach 8V. R1 and R2 together form a potential divider which halves the voltage from the motor keeping it mostly in the measurable range. This halved voltage is read by the BBC micro:bit and converted into a number between 0 and 1023. This value is then compared to the value currently stored in the variable 'Highest'. If this value is higher than the currently stored value then it replaces the old value. When button A is pressed the highest recorded value is scrolled on the screen. Resetting the BBC micro:bit by using the reset button on the BBC micro:bit will clear this value.

8

EXPERIMENT EIGHT

MAKING A GAME USING THE COMPASS

THE AIMS OF THIS EXPERIMENT ARE...

- To calibrate and use the on-board compass.
- To make a game that integrates the on-board compass.

THE EXPERIMENT

The BBC micro:bit comes with a built in compass so it can tell which way is north. It does this using a chip that is very sensitive to magnetism and can detect earth's magnetic field. This means it will not work properly if it's held near a magnet. The compass also needs calibrating before use, don't worry though as this is a lot easier than it sounds.

The purpose of this experiment is to design a game where a secret compass heading is selected randomly and an LED blinks faster or slower depending on how close to the correct heading the BBC micro:bit is facing. When the player is confident they are facing the correct heading they press a button and see if they are correct.

PARTS USED



RED 5MM LED x1



47Ω RESISTOR x1



MALE-TO-FEMALE JUMPER WIRE x2

CREATE THE FOLLOWING TOUCH DEVELOP SCRIPT

script Hotter or Colder

function main ()

```
input → calibrate
{
  □ img := image → create image(📺)
  □ img → show image(0)
  □ goal := math → random(360)
  basic → forever do
  {
    var degrees := input → compass heading
    □ difference := math → abs(□ goal - degrees)
    pins → digital write pin(P0, 1)
    basic → pause(□ difference * 5)
    pins → digital write pin(P0, 0)
    basic → pause(□ difference * 5)
  }
end
input → on button pressed(A) do
{
  if □ difference < 15 then
  {
    basic → show string("Winner", 150)
    □ goal := math → random(360)
  }
  else
  {
    basic → show string("Try Again", 150)
    □ goal := math → random(360)
  }
  end if
}
end
end function
```



Run



Compile

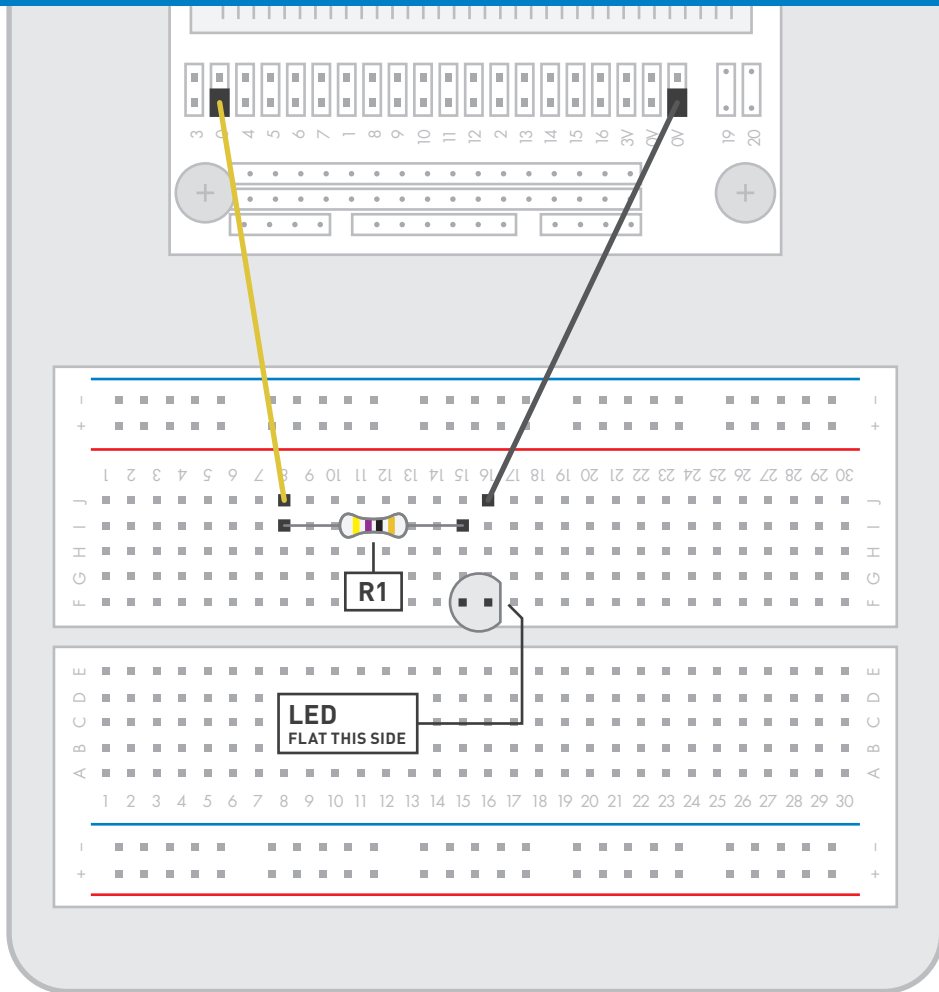


Upload



HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

BUILD THIS CIRCUIT ON THE BREADBOARD

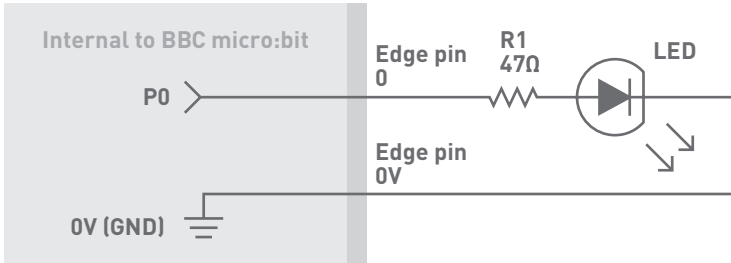


Note: 47Ω resistor is marked with coloured bands: yellow, purple, black, gold.

WHAT WILL HAPPEN?

Two arrows will appear on the LED display, rotate the BBC micro:bit in the direction indicated by the arrows until the smiley face is displayed. At this point the LED will begin to flash. Slowly rotate the BBC micro:bit until the LED flashes very rapidly then press the A button and the text 'winner' should appear.

CIRCUIT DIAGRAM



WHAT'S GOING ON?

The first line of code activates the calibration sequence for the compass. This asks the user to rotate the BBC micro:bit in a direction shown on the LED display with arrows. After this an image is created and displayed to let the user know the calibration has finished.

Next a global variable called 'goal' is created and used to store a random number between 0 and 360. This is the winning angle that the player must find.

In the forever loop of the code a variable is created and filled with the current reading of the compass. The next line creates a global variable that stores the difference between the angle recorded by the compass and the winning angle. The maths function 'abs' is used here. 'abs' forces a number to be positive, so if the difference is -10 then the 'abs' function changes it to +10. The pin P0 pulses high and low but the time it spends in each state is equal to the 'difference' variable multiplied by 5. So the smaller the difference between the current and goal angles the faster the connected LED will flash.

The next segment of code checks for button A being pressed, when pressed it checks that the current angle the BBC micro:bit is facing is within 15 degrees of the winning 'goal' angle. If it is then a winning message is displayed on the LED matrix and a new goal angle is set. If the player loses then the message is instead replaced with a 'try again' message.

9

EXPERIMENT NINE

CAPACITOR CHARGE CIRCUIT

THE AIMS OF THIS EXPERIMENT ARE...

- To charge an electrolytic capacitor through a potentiometer.
- To measure the charge of a capacitor by using an analog input.
- To display the charge percentage on the LED display.
- To create a colour coded scale to show capacitor charge level.

THE EXPERIMENT

In this experiment one of the analog inputs on the BBC micro:bit will be used to measure the voltage at the positive leg of a capacitor. A circuit will be created that allows the capacitor to be charged through a variable resistor (also known as a potentiometer). The BBC micro:bit will be programmed to display the charge level of the capacitor both numerically and using external LEDs to create a colour coded display.

PARTS USED

	RED 5MM LED	x1		M/F JUMPER WIRE	x8
	ORANGE 5MM LED	x1		M/M JUMPER WIRE	x1
	YELLOW 5MM LED	x1		220uF CAPACITOR	x1
	GREEN 5MM LED	x1		POTENTIOMETER	x1
	2.2kΩ RESISTOR	x2		47Ω RESISTOR	x4
	PUSH SWITCH	x2			

CREATE THE FOLLOWING TOUCH DEVELOP SCRIPT

script Capacitor Charge

function main ()

```
basic → forever do
  ( var cap voltage := pins → analog read pin(P0)
  ( var percentage := cap voltage / 10
  basic → show number(percentage, 100)
  if percentage > 25 and percentage ≤ 50 then
    ( pins → digital write pin(P1, 1)
  else if percentage > 50 and percentage ≤ 75 then
    ( pins → digital write pin(P1, 1)
    ( pins → digital write pin(P2, 1)
  else if percentage > 75 and percentage ≤ 90 then
    ( pins → digital write pin(P1, 1)
    ( pins → digital write pin(P2, 1)
    ( pins → digital write pin(P8, 1)
  else if percentage > 90 then
    ( pins → digital write pin(P1, 1)
    ( pins → digital write pin(P2, 1)
    ( pins → digital write pin(P8, 1)
    ( pins → digital write pin(P12, 1)
  else
    ( pins → digital write pin(P1, 0)
    ( pins → digital write pin(P2, 0)
    ( pins → digital write pin(P8, 0)
    ( pins → digital write pin(P12, 0)
  end if
end
```



Run



Compile

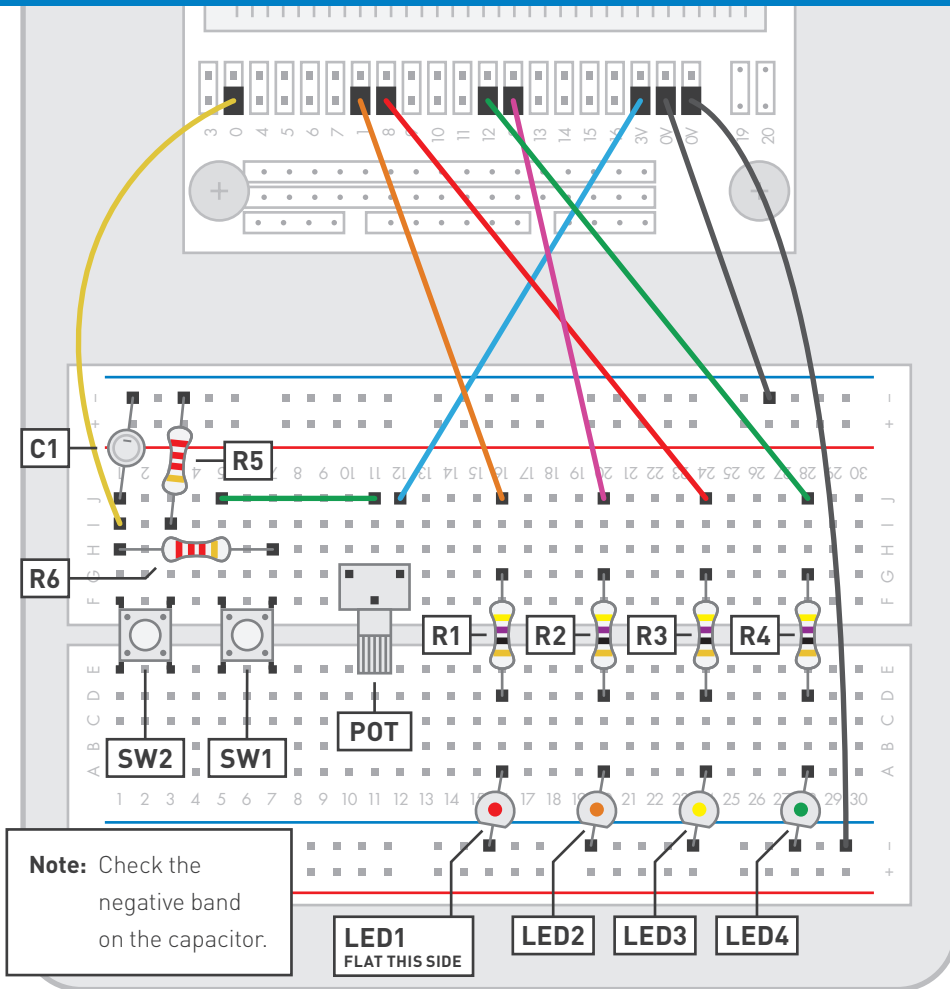


Upload



HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

BUILD THIS CIRCUIT ON THE BREADBOARD

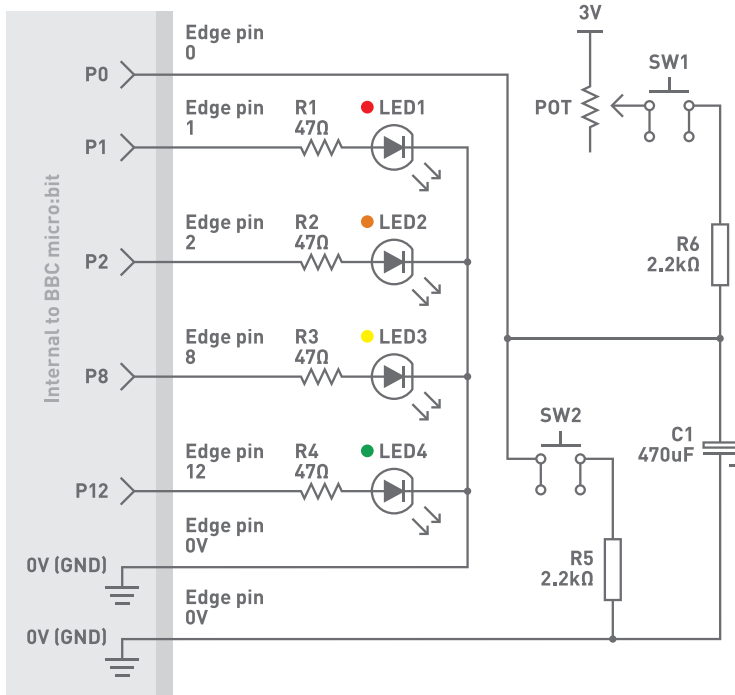


Note: 470Ω resistors are marked with coloured bands: yellow, purple, black, gold.
2.2kΩ resistors are marked with coloured bands: red, red, red, gold.

WHAT WILL HAPPEN?

Holding SW1 will increase the number displayed on the LED matrix. Once the number rises above 25 the first LED will light. The next three LEDs will light at 50, 75 and 90. Pressing SW2 will reset the experiment and the lights will go off.

CIRCUIT DIAGRAM



WHAT'S GOING ON?

Pressing SW1 charges the capacitor through the potentiometer POT. The rate of charge depends on the resistance of POT, a high resistance gives a low charge rate and a low resistance gives a high charge rate. The voltage at the positive side of the capacitor C1 is measured by analog input P0 on the BBC micro:bit. When C1 is empty this voltage will be 0V, when fully charged this voltage will be 3V. The BBC micro:bit reads this voltage and converts it to a number between 0 and 1023. This number is in turn converted to a percentage and an LED will light as this percentage passes 25%, 50%, 75% and 90%.

Pressing SW2 connects the positive side of the capacitor to ground, rapidly emptying it as the current flows to ground. This effectively resets the experiment.

10

EXPERIMENT TEN

USING AN RGB LED

THE AIMS OF THIS EXPERIMENT ARE...

- To use an RGB LED
- To observe how different colours are made from mixing red, green and blue light.

THE EXPERIMENT

This experiment puts the RGB LED to use. An RGB LED is a special LED that contains three separate LEDs in one package. As you might have guessed the three LEDs are Red, Green and Blue. The light from these LEDs can be mixed together to allow for the creation of many colours.

We can use the PWM outputs of the BBC micro:bit to have a very fine control over the colours and shades

The RGB LED included in this Inventor's pack is a common cathode LED which means all three LEDs inside the package share the same negative leg.

PARTS USED



RGB LED x1



M/F JUMPER WIRE x9



TACT SWITCH x3



M/M JUMPER WIRE x3



10k RESISTOR x3



47Ω RESISTOR x3

CREATE THE FOLLOWING TOUCH DEVELOP SCRIPT

```
script RGB LED
function main ()
{
  Red := 0
  Green := 0
  Blue := 0
  basic → forever do
  {
    pins → analog write pin(P0, Green)
    pins → analog write pin(P1, Red)
    pins → analog write pin(P2, Blue)
    if pins → digital read pin(P8) = 1 and Green < 1020 then
    {
      Green := Green + 10
    }
    else if pins → digital read pin(P8) = 1 and Green = 1020 then
    {
      Green := 0
    }
    else add code here end if
    if pins → digital read pin(P12) = 1 and Red < 1020 then
    {
      Red := Red + 10
    }
    else if pins → digital read pin(P12) = 1 and Red = 1020 then
    {
      Red := 0
    }
    else add code here end if
    if pins → digital read pin(P16) = 1 and Blue < 1020 then
    {
      Blue := Blue + 10
    }
    else if pins → digital read pin(P16) = 1 and Blue = 1020 then
    {
      Blue := 0
    }
    else add code here end if
  }
end
end function
```

Note: The diagonal arrow button can be pressed to add a new line of code outside of the current function.



Run



Compile

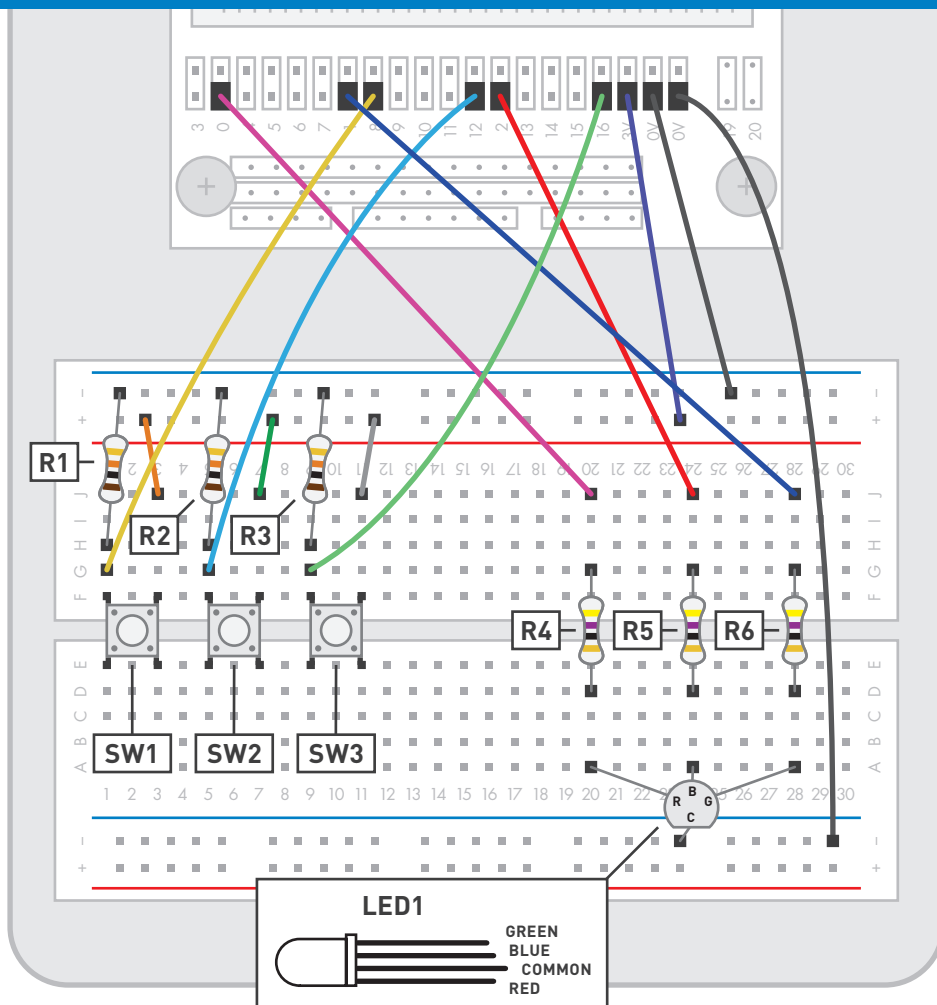


Upload



HAVING TROUBLE? Visit www.kitronik.co.uk/5603 for support

BUILD THIS CIRCUIT ON THE BREADBOARD

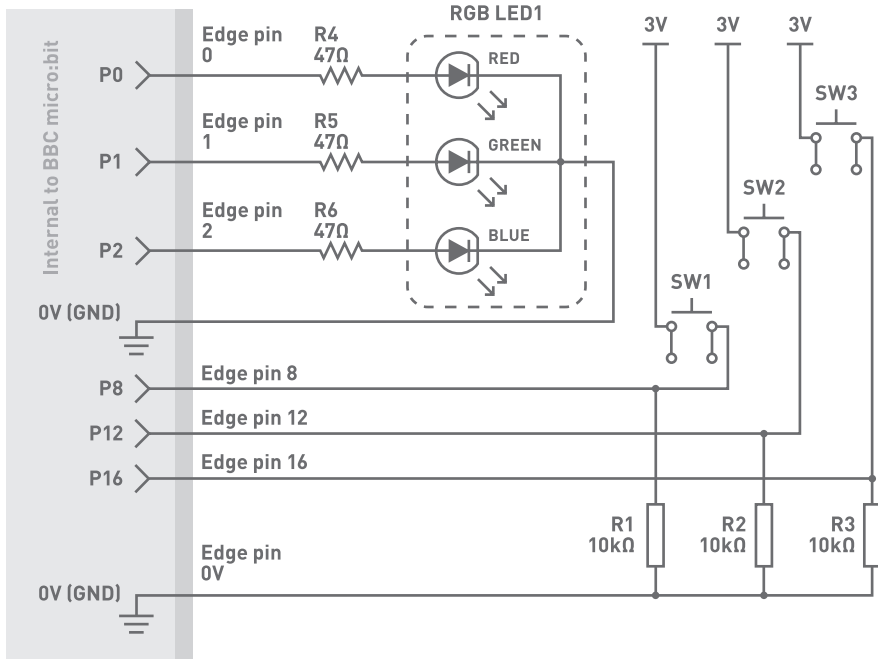


Note: 10k resistors are marked with coloured bands: brown, black, orange, gold.
470Ω resistors are marked with coloured bands: yellow, purple, black, gold.

WHAT WILL HAPPEN?

Each switch controls the brightness of one of the colours (SW1 = Green, SW2 = Red, SW3 = Blue). Pressing a switch will increase the brightness of that switch's colour. Once the colour is at a maximum pressing it again will reset that colour to 0 brightness, turning it off.

CIRCUIT DIAGRAM



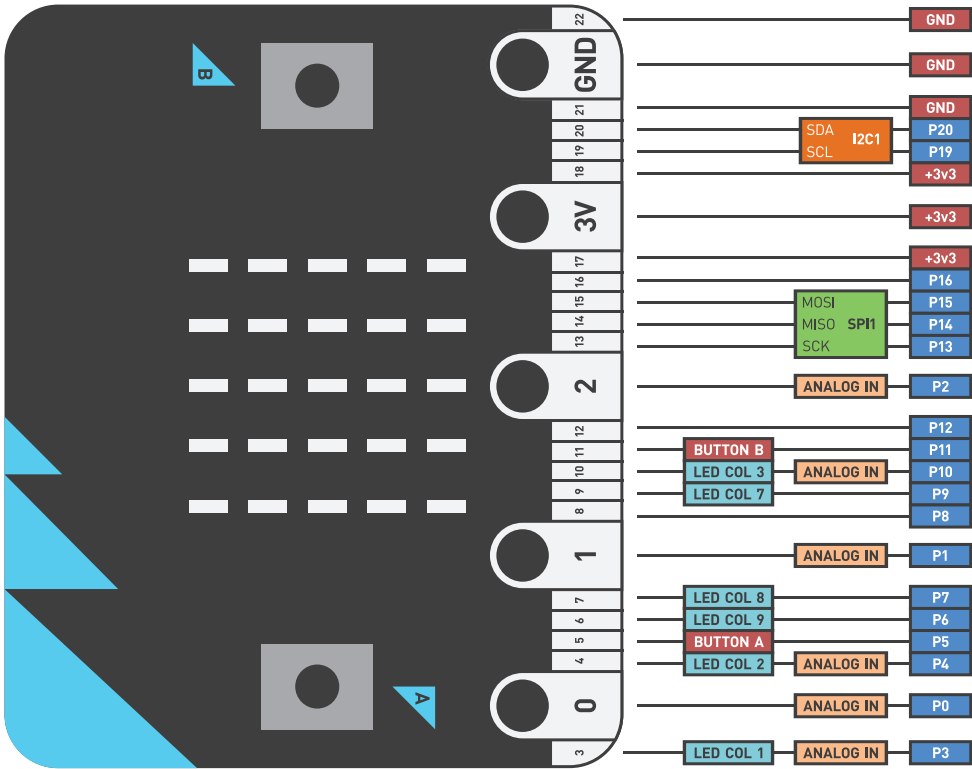
WHAT'S GOING ON?

The circuit uses the three analog PWM output pins from the BBC micro:bit, one for each of the three LEDs inside the RGB LED package. These pins output a PWM signal determined by the user pressing each of the three switches. As a switch is held it will raise the analog out value continuously in increments of 10, up to a maximum of 1010. If this maximum value is exceeded then the value resets back to 0. The switches have pull down resistors that hold P8, P12 and P16 low until the switch is pressed at which point they go high.

By having more than one colour active at a time new colours can be created. A huge spectrum of colours can be produced just by using this code and an RGB LED.



For more experiments & support visit www.kitronik.co.uk/5603

BBC micro:bit PIN CONNECTIONS



W: www.kitronik.co.uk

E: support@kitronik.co.uk


 Designed & manufactured
 in the UK by 


[kitronik.co.uk/twitter](https://twitter.com/kitronik)


[kitronik.co.uk/facebook](https://facebook.com/kitronik)


[kitronik.co.uk/youtube](https://youtube.com/kitronik)


[kitronik.co.uk/google](https://google.com/kitronik)


STOCK CODE
 5603