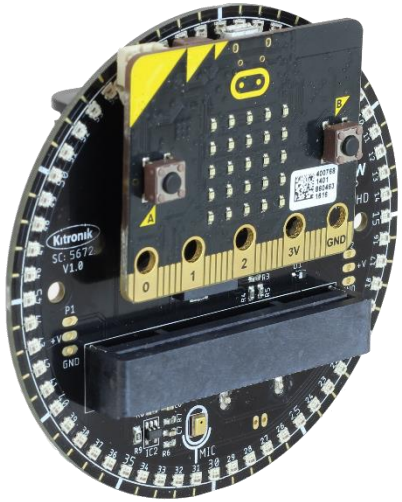
:VIEW ZIP™ Halo HD Board for the BBC micro:bit

www.kitronik.co.uk/5672

Introduction: This Halo HD board for the BBC micro:bit incorporates 60 full colour, addressable ZIP™ LEDs, connected to the BBC micro:bit. It also breaks out P1 and P2 to a standard 0.1" footprint. Included on the board is a real time clock (RTC) controlled by I2C lines from the BBC micro:bit, a MEMS microphone for detection of sound, and a piezo buzzer provides the ability to play sound. For details of pins connections, see the BBC micro:bit pin connection information table on the following page.

The board makes use of an edge connector for easy connection of the BBC micro:bit.

The board has a 3xAA battery holder mounted on the rear, with a power switch on the front of the board. A **regulated supply** is produced on the board which is fed into the 3V and GND connections to **power the connected BBC micro:bit**, removing the need to power the BBC micro:bit separately. To protect the BBC micro:bit if power is supplied through it, the ZIP™ Halo HD will not illuminate.

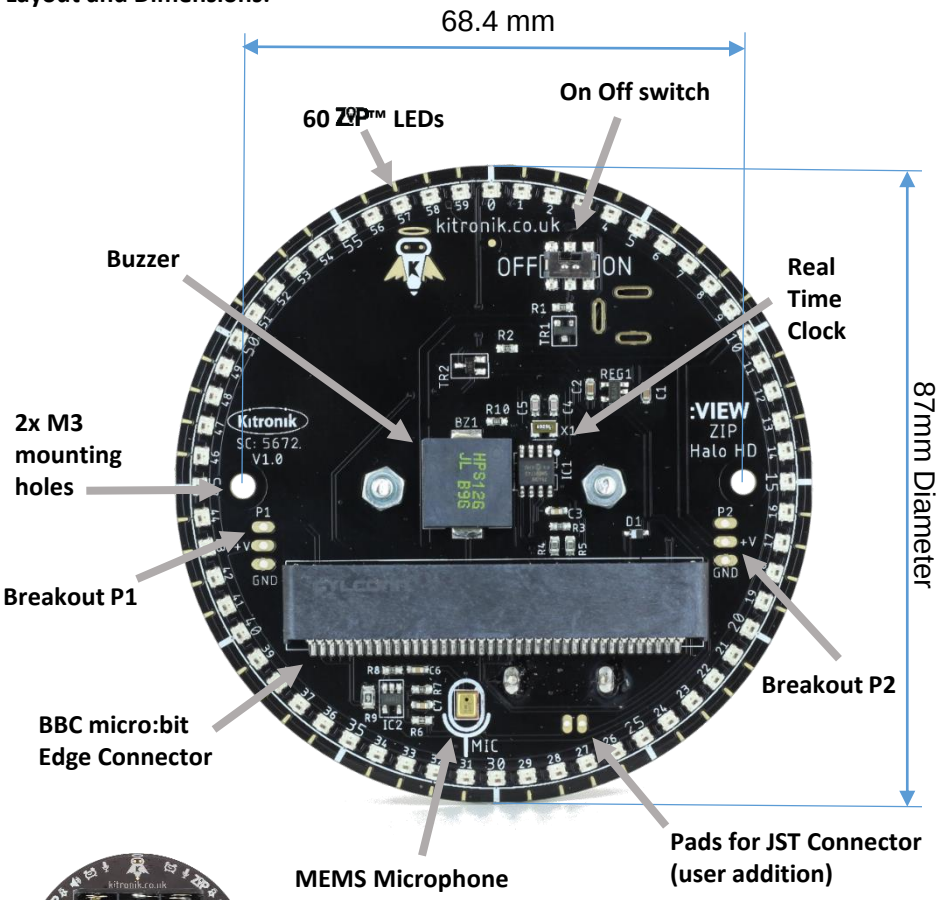


Connecting a BBC micro:bit:

The board has been designed so that the BBC micro:bit can be inserted into the edge connector requiring no additional tools.

Examples: There is example code within this datasheet to get started with the board. Other uses for the Halo HD could include a lamp, clock or a compass. For more details see: http://www.kitronik.co.uk/halohd

Layout and Dimensions:



3xAA Battery cage on the rear side of the PCB

The Halo HD board is 30mm thick (including BBC micro:bit and battery cage)

:VIEW ZIP™ Halo HD Board for the BBC micro:bit

www.kitronik.co.uk/5672

Electrical Information

Operating Voltage (Vcc)	3.5V – 5V
Number of ZIP LEDs	60
Number of external channels	3 (1x ZIP LED, 2x IO pin, each IO channel rated +3V at 50mA)
Max Current (ZIP LED running full RGB brightness)	0.9A (15mA per ZIP)
MEMS Microphone Sensitivity	Typ: -42 dBV/Pa
MEMS Microphone Signal to Noise Ratio	59dB

BBC micro:bit Pin Connection Information

MEMS Microphone	P0 (analogue input)
P1 Breakout pin header	1 – P1 2 - +V (Battery Voltage) 3 - GND
P2 Breakout pin header	1 - P2 2 - +V (Battery Voltage) 3 - GND
ZIP LED	P8
Buzzer	P14
RTC	P19 & P20 (using the I2C lines)

:VIEW ZIP™ Halo HD Board for the BBC micro:bit

www.kitronik.co.uk/5672

MakeCode Blocks Editor Code

Kitronik have created custom blocks for the Halo HD for use with MakeCode.

To add these blocks, first go to makecode.microbit.org and start a new project.
Under the “Advanced” section click on “Extensions”. In the next window search for “Halo HD”.
Once the icon appears, click on the icon to import it into MakeCode.

In the Halo HD section there will appear three sub-categories: ZIP LEDs, Microphone and Clock. The example code pictured will be using blocks from the ZIP LEDs and Microphone sections.

The first example will make the HaloHD to display a rotating rainbow pattern.

In the on start block, the number of ZIP LEDs is defined. Then, a rainbow spectrum is shown across all the LEDs.

In the forever loop, the LED colour is rotated by 1 and then shown on the ZIP LEDs.
There is then a pause of 100ms before rotating the LEDs. This will enable the colours to move around the LEDs.

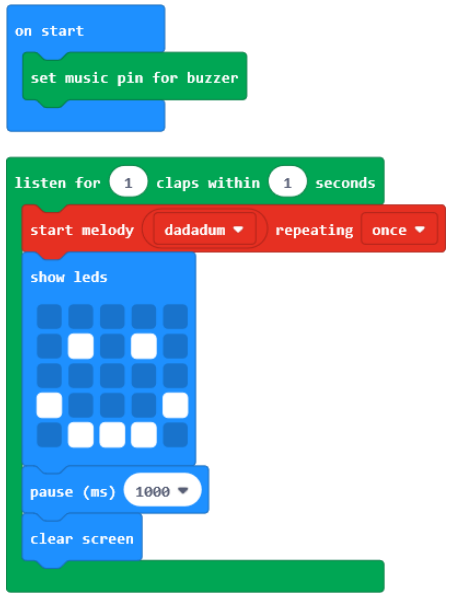
The second example sets the music pin to the correct pin for the buzzer on the Halo HD board.

The interrupt block waits for a single clap to be detected by the MEMS microphone.
Once detected, a melody will be played (this is from the standard music blocks in MakeCode), followed by a smiley face held for 1 second.

After the 1 second wait, the micro:bit screen is cleared ready to display for the next clap detected.

Example 1

Example 2



MicroPython Editor Code

The example will cause the HaloHD to display a rotating rainbow pattern.

Note: The neopixel and microbit pin8 imports are required for this program.

First, the number of LEDs and which pin the LEDs are connected to are declared.

Next, the red, green and blue variables are all set to a value from 0-240 (although the colours of the LEDs can go to 255, 240 can be divided by 60, so each LED will have an even colour difference).

Next, a flag for each colour is set to indicate if the value needs to be increased or decreased. This is changed when the particular colour value reaches 0 or 240.

The red, green and blue values are set for the LED and then the show function is called to send it to the LED. This operation is carried out within a for loop, so the next time round in the loop it addresses the next LED until all 60 LEDs have had their colour values set.

Once complete, the loop will start again, but this time with the colour offset. The LEDs will change slightly, giving the impression of the spectrum of colours moving round the LEDs.

For more examples visit the Kitronik website at:
http://www.kitronik.co.uk/halohd

```
from microbit import pin8
import neopixel

# Declare constants
LEDS_ON_HALO = 60
zip_leds = neopixel.NeoPixel(pin8, LEDS_ON_HALO)
r = 240
g = 120
b = 0
zip = 0
rCount = "minus"
gCount = "minus"
bCount = "plus"

while True:
    for zip in range(60):
        if r == 240:
            rCount = "minus"
        elif r == 0:
            rCount = "plus“

        if g == 240:
            gCount = "minus"
        elif g == 0:
            gCount = "plus"

        if b == 240:
            bCount = "minus"
        elif b == 0:
            bCount = "plus"

        if rCount == "plus":
            r = r + 6
        else:
            r = r - 6

        if gCount == "plus":
            g = g + 6
        else:
            g = g - 6

        if bCount == "plus":
            b = b + 6
        else:
            b = b - 6

        zip_leds[zip] = (r, g, b)
        zip_leds.show()
```