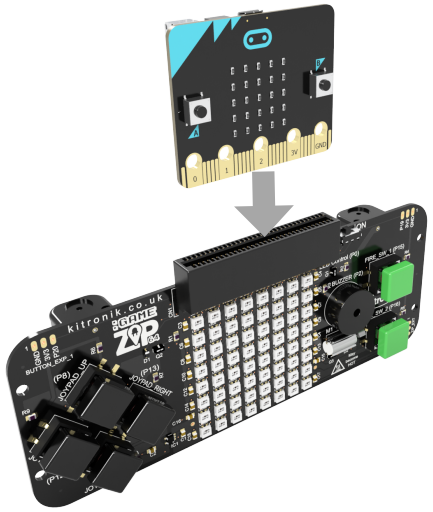
:GAME ZIP™64 *for the BBC micro:bit*

TECHNOLOGY DATA SHEET & SPECIFICATIONS

Introduction: The :GAME ZIP™64 is a programmable gamepad for the BBC micro:bit. It features 64 colour addressable LEDs arranged in an 8 x 8 display, a piezo buzzer for audio feedback, a vibration motor for haptic feedback, and 6 input buttons. It also breaks out P19, P20 & LED DOUT to standard 0.1" footprints. Each of these pins also have the required voltage and GND pads. The BBC micro:bit is connected via a standard card slot connector.

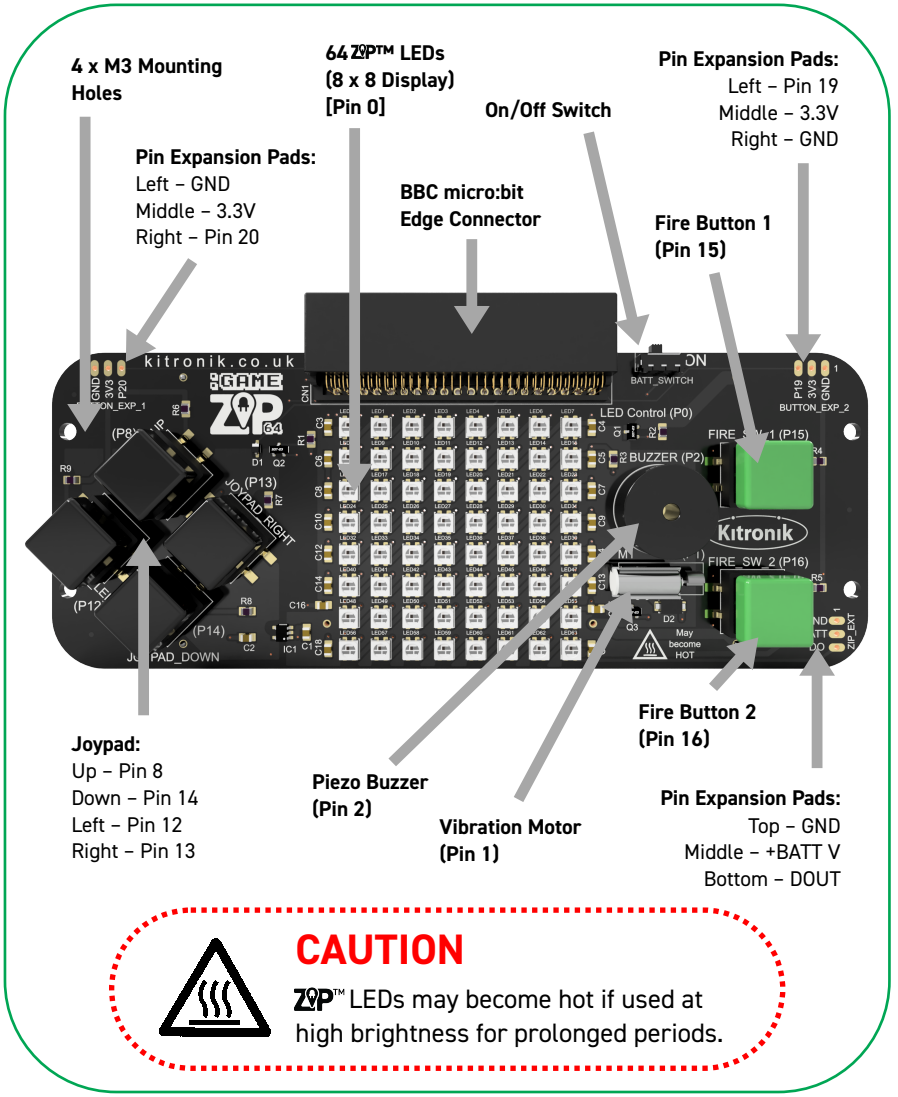
The board produces a **regulated supply** that is fed into the 3V and GND connections **to power the connected BBC micro:bit**, removing the need to power the BBC micro:bit separately. To protect the BBC micro:bit if power is supplied through it, the ZIP™LEDs will not illuminate.



Inserting a BBC micro:bit:

To use the :GAME ZIP™64, the BBC micro:bit should be inserted firmly into the edge connector, making sure that the BBC micro:bit LED display is facing in the same direction as the :GAME ZIP™64 LED display.

Examples: For some starter games and ideas for what else you could do, go to: http://www.kitronik.co.uk/5626



Electrical Information

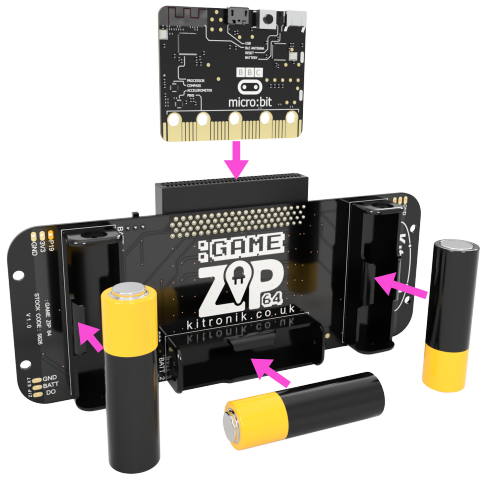
Operating Voltage (Vcc) [ZIP LEDs]	+3.5V – +5.3V
Regulated Voltage [BBC micro:bit, Buttons, Vibration Motor]	+3.3V
Max Current (ZIP LEDs White @ 100% brightness and all devices in use)	1.6A (21mA per ZIP LED, 250mA max on +3.3V reg. voltage)
Number of ZIP LEDs	64
Number of external channels	3 (1 x ZIP LED, 2 x I2C/IO pin, each IO pin rated +3.3V @ 5mA)

Note on External Channels:

Care should be taken when using the external breakouts for Pins 19 and 20 as GPIOs, as this can cause issues with the I2C devices on the BBC micro:bit itself (e.g. compass and accelerometer).

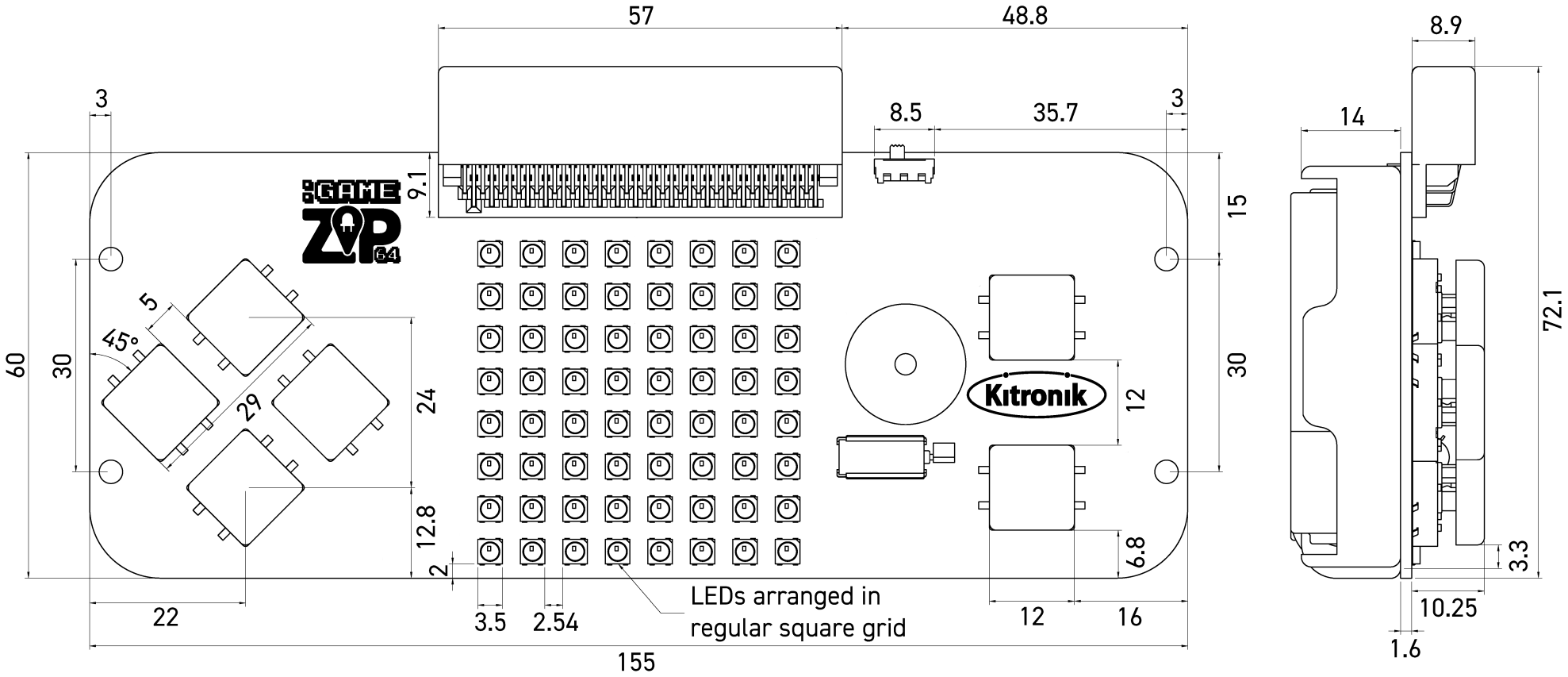
When using the 3.3V breakout pins, these should not draw more than 50mA each, or 100 mA in total, due to the current limit of the voltage regulator.

Rear View with BBC micro:bit & Batteries:



Board Dimensions

(All measurements are given in mm)



MakeCode Blocks Editor Code

This program was created in the Microsoft MakeCode Blocks Editor for the BBC micro:bit. It creates a single pixel sprite which can be moved around the display using the Joypad buttons and have its colour changed using the Fire buttons. When the sprite reaches the display edge, the motor will vibrate and the buzzer will play a short tune.

Note: There is Kitronik package available for the :GAME ZIP™ 64 on Microsoft MakeCode (the green blocks shown here).

MicroPython Editor Code

This program was created in the MicroPython Mu Editor for the BBC micro:bit.
It provides exactly the same functionality as the MakeCode Blocks program.

```
from microbit import *
import neopixel
import music

# Enable ZIP LEDs to use x & y values
def zip_plot(x, y, colour):
    zip_led[x+(y*8)] = (colour[0], colour[1],
colour[2])

# Function to play tune on buzzer and run
motor for 500ms
def hit_edge():
    music.play(music.BA_DING, pin2, False)
    pin1.write_digital(1)
    sleep(500)
    pin1.write_digital(0)

# Setup variables and initial ZIP LED display
zip_led = neopixel.NeoPixel(pin0, 64)
sprite_x = 3
sprite_y = 3

# Colours: Red, Yellow, Green, Blue, Purple,
White
colours = [[20, 0, 0], [20, 20, 0], [0, 20, 0], [0,
0, 20], [20, 0, 20], [20, 20, 20]]
sprite_colour = colours[3]
zip_plot(sprite_x, sprite_y, sprite_colour)
zip_led.show()

# While loop to run forever
while True:
    # Check button presses
    if pin8.read_digital() == 0 and sprite_y ==
0:
        hit_edge()
    elif pin8.read_digital() == 0 and sprite_y !=
0:
        sprite_y = sprite_y - 1

 if pin14.read_digital() == 0 and sprite_y == 7:
        hit_edge()
    elif pin14.read_digital() == 0 and sprite_y !
= 7:
        sprite_y = sprite_y + 1

    if pin12.read_digital() == 0 and sprite_x ==
0:
        hit_edge()
    elif pin12.read_digital() == 0 and sprite_x !
= 0:
        sprite_x = sprite_x - 1

    if pin13.read_digital() == 0 and sprite_x ==
7:
        hit_edge()
    elif pin13.read_digital() == 0 and sprite_x !
= 7:
        sprite_x = sprite_x + 1

  if pin15.read_digital() == 0:
        if colours.index(sprite_colour) - 1 < 0:
            sprite_colour = colours[0]
        else:
            sprite_colour =
colours[(colours.index(sprite_colour) - 1)]

    if pin16.read_digital() == 0:
        if colours.index(sprite_colour) + 1 > 5:
            sprite_colour = colours[5]
        else:
            sprite_colour =
colours[(colours.index(sprite_colour) + 1)]

    # Clear and redisplay the ZIP LEDs after
each button press check
    zip_led.clear()
    zip_plot(sprite_x, sprite_y, sprite_colour)
    zip_led.show()

    # 100ms pause before restarting the while
loop
    sleep(100)
```