

# :GAME ZIP™ 64 voor de BBC micro:bit



www.kitronik.co.uk/5626

De :GAME ZIP 64 is een programmeerbare gamepad voor de BBC micro:bit. Het beschikt over 64 adresseerbare LED's in een 8 x 8 display, een piezo-zoemer voor audiofeedback, een vibratiemotor voor haptische feedback en 6 invoerknoppen. Het geeft ook P19, P20 & LED DOUT uit naar standaard 0,1" footprints. Elk van deze pinnen heeft ook de vereiste voltage- en GND-pads. De BBC micro:bit is aangesloten via een standaard randaansluiting van de kaartsleuf.

Het bord produceert een **gereguleerde voeding** die in de 3V- en GND-aansluitingen wordt gevoerd **om de aangesloten BBC micro:bit** aan te sturen, waardoor de noodzaak om het BBC micro:bit afzonderlijk te voeden, wordt weggenomen. Om de BBC micro:bit te beschermen als er stroom door wordt geleverd, zullen de ZIP™-LED's niet oplichten.

## Bordindeling:

### Pin uitbreiding pads

Links - GND  
Midden - 3,3V  
Rechts - Pin 20  
4 x M3 montagegaten

Joypad omhoog [Pin 8]

Joypad links [Pin 12]

Joypad rechts [Pin 13]

Joypad omlaag [Pin 14]

Achterkant: 3 x AA-batterijhouders

64 ZIP™ LEDs  
(8 x 8 display)  
[Pin 0]

BBC micro:bit  
Edge Connector

Aan/Uit  
schakelaar

Pin uitbreiding pads  
Links - Pin 19  
Midden - 3,3V  
Rechts - GND

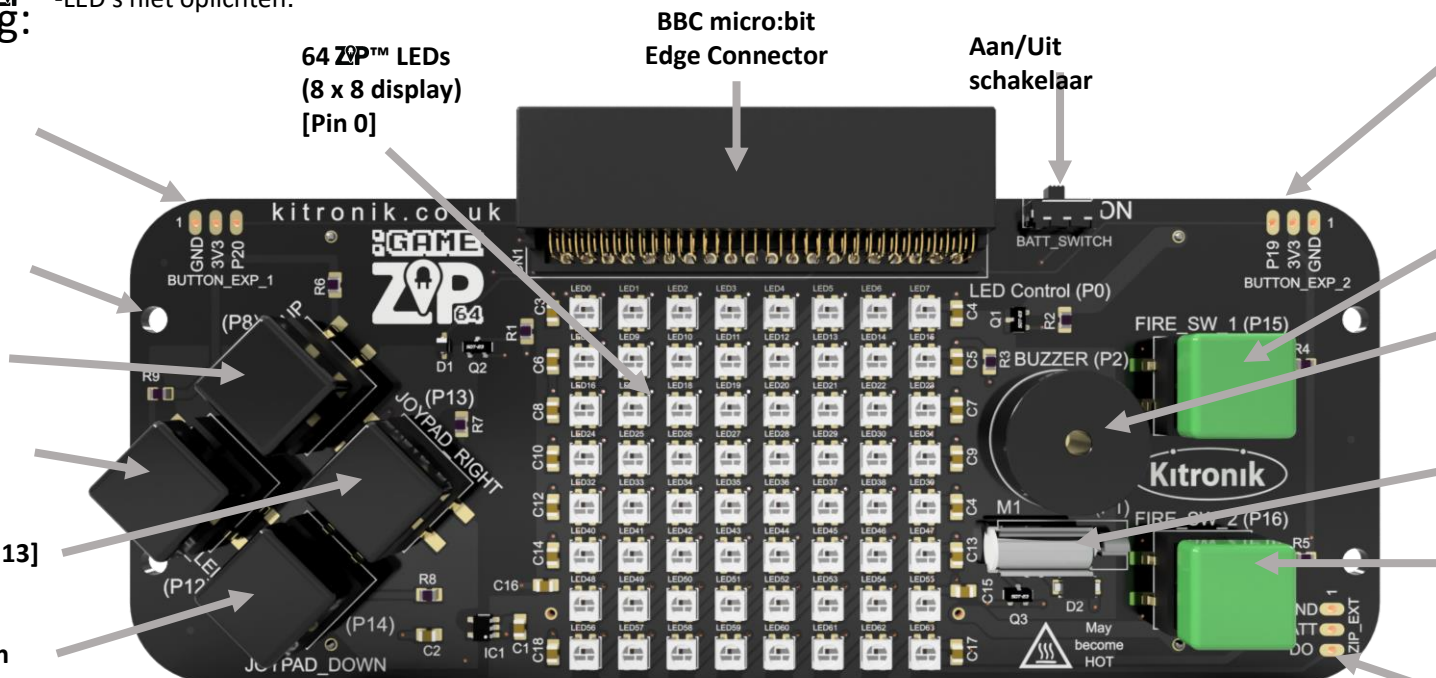
Vuurknop 1 [Pin 15]

Piezo zoemer [Pin 2]

Trilmotor [Pin 1]

Vuurknop 2 [Pin 16]

ZIP™ LED  
Uitbreiding pads:  
Top - GND  
Midden - + BATT  
V  
Onderkant -  
DOUT



## Een BBC micro:bit invoegen:

Om de :GAME ZIP™ 64 te gebruiken, moet de BBC micro:bit stevig in de randconnector worden geplaatst, waarbij ervoor wordt gezorgd dat het BBC micro:bit LED-display in dezelfde richting kijkt als het :GAME ZIP™ 64 LED-display.

**Voorbeelden:** Voor sommige starterspellen en ideeën over wat je nog meer zou kunnen doen, ga je naar: <http://www.kitronik.co.uk/5626>

## Voorzichtig:

ZIP™-LED's kunnen heet worden als ze langdurig met hoge helderheid worden gebruikt.



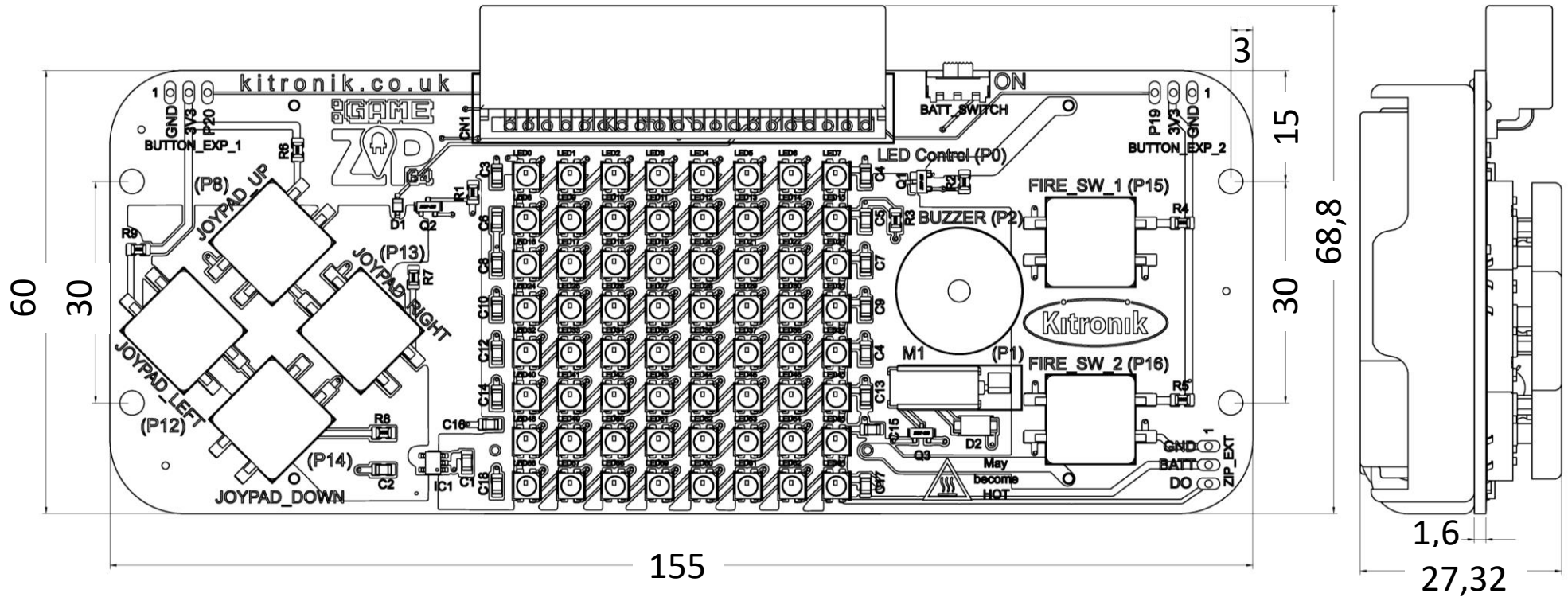
# :GAME ZIP<sup>1</sup> 64 voor de BBC micro:bit

www.kitronik.co.uk/5626



## Bordmaten:

(Alle maten zijn aangegeven in mm)



## Electrische informatie

Bedrijfsspanning (Vcc) [ZIP LED's]	+3,5V - +5,3V
Gereguleerde spanning [BBC micro: bit, knoppen, vibratiemotor]	+3,3V
Max. stroom (ZIP-leds met volledige RGB-helderheid )	1,6A (21mA per ZIP LED, 250mA max op +3,3V reg. spanning)
Aantal ZIP LED's	64
Aantal externe kanalen	3 (1 x ZIP-LED, 2 x I2C / IO-pin, elke IO-pinaansluiting +3,3V bij 5mA)

### Opmerking over externe kanalen:

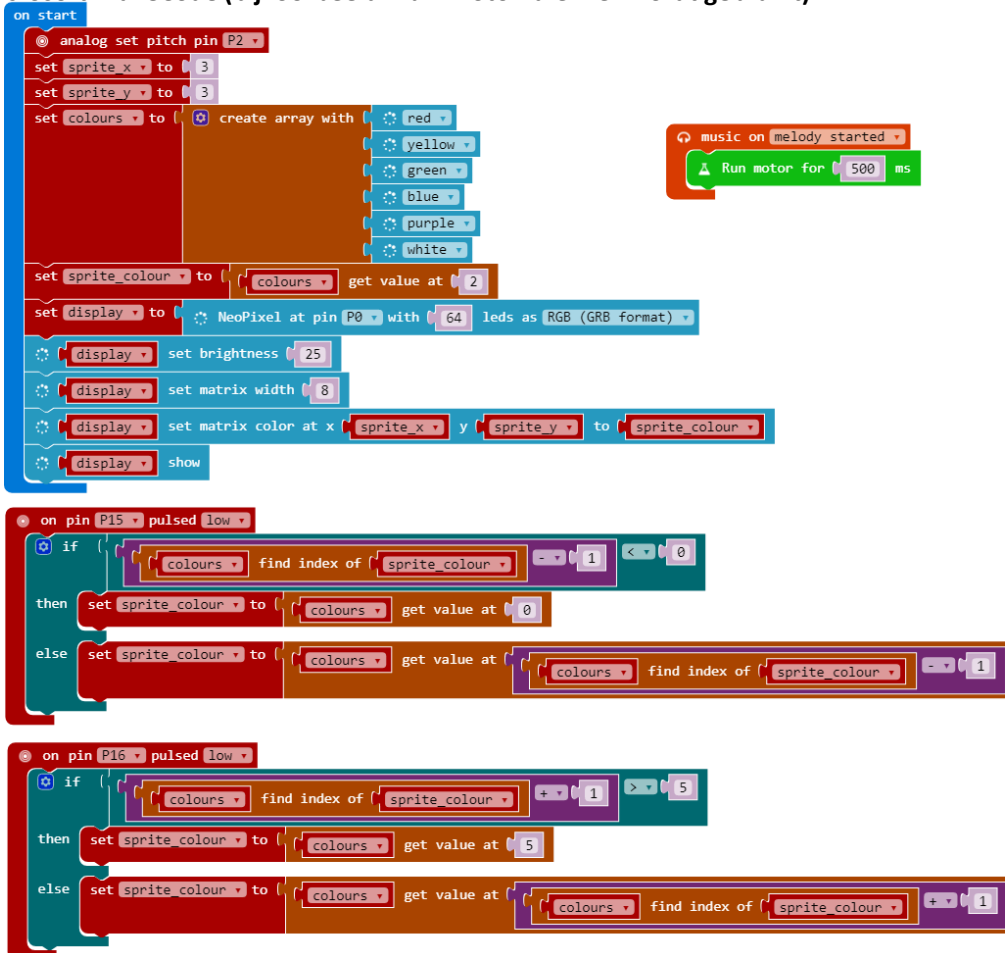
Wees voorzichtig als u de externe pannes voor pinnen 19 en 20 als GPIO's gebruikt, omdat dit problemen kan veroorzaken met de I2C-apparaten op de BBC micro: bit zelf (bijv. Kompas en versnellingsmeter).

### Achteraanzicht met BBC micro:bit & batterijen:



## Microsoft MakeCode Blocks Editor Code

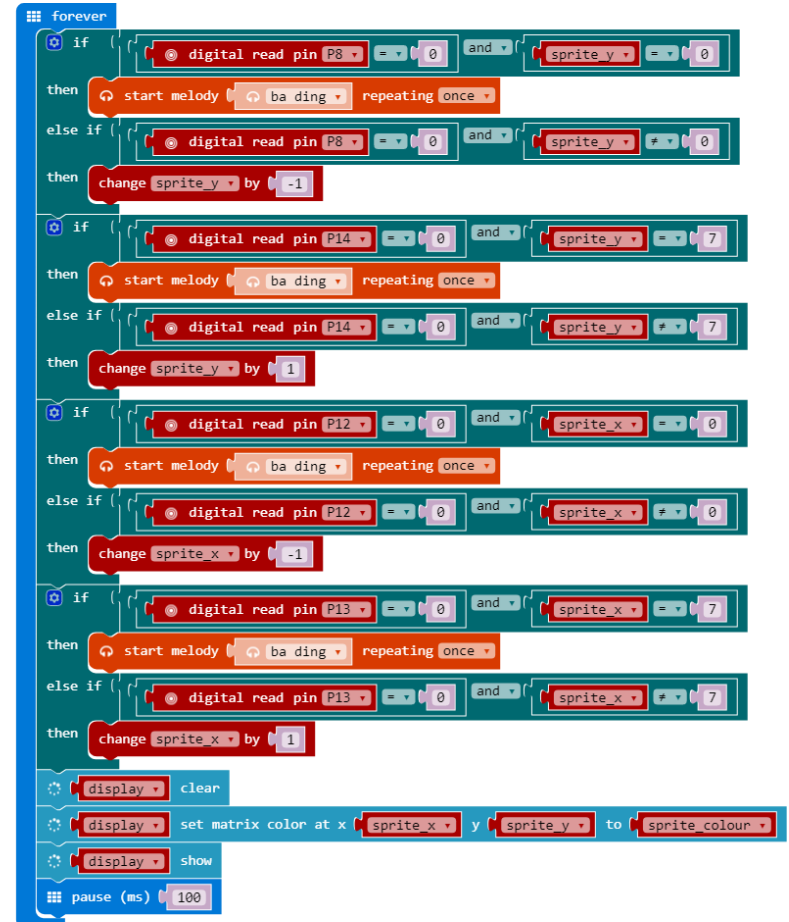
Dit programma is gemaakt in de Microsoft MakeCode Blocks Editor voor de BBC micro:bit. Het creëert een sprite van één pixel die met behulp van de knoppen van de Joypad over het display kan worden verplaatst en waarvan de kleur kan worden gewijzigd met behulp van de Vuur knoppen. Wanneer de sprite de rand van het scherm bereikt, trilt de motor en speelt de zoemer een kort deuntje. **Opmerking:** Er zijn enkele Kitronik Custom Blocks beschikbaar voor de :GAME ZYP<sup>TM</sup> 64 op Microsoft MakeCode (bijvoorbeeld 'Run-motor' die hier wordt gebruikt).



```
on start
  analog set pitch pin P2
  set sprite_x to 3
  set sprite_y to 3
  set colours to [red, yellow, green, blue, purple, white]
  music on melody started
  Run motor for 500 ms
  set sprite_colour to [colours] get value at 2
  set display to NeoPixel at pin P0 with 64 leds as RGB (GRB format)
  display set brightness 25
  display set matrix width 8
  display set matrix color at x [sprite_x] y [sprite_y] to [sprite_colour]
  display show

on pin P15 pulsed low
  if [colours] find index of [sprite_colour] - 1 < 0
  then set sprite_colour to [colours] get value at 0
  else set sprite_colour to [colours] get value at [colours] find index of [sprite_colour] - 1

on pin P16 pulsed low
  if [colours] find index of [sprite_colour] + 1 > 5
  then set sprite_colour to [colours] get value at 5
  else set sprite_colour to [colours] get value at [colours] find index of [sprite_colour] + 1
```



```
forever
  if [digital read pin P8] == 0 and [sprite_y] == 0
  then start melody [ba ding] repeating once
  else if [digital read pin P8] == 0 and [sprite_y] != 0
  then change sprite_y by -1

  if [digital read pin P14] == 0 and [sprite_y] == 7
  then start melody [ba ding] repeating once
  else if [digital read pin P14] == 0 and [sprite_y] != 7
  then change sprite_y by 1

  if [digital read pin P12] == 0 and [sprite_x] == 0
  then start melody [ba ding] repeating once
  else if [digital read pin P12] == 0 and [sprite_x] != 0
  then change sprite_x by -1

  if [digital read pin P13] == 0 and [sprite_x] == 7
  then start melody [ba ding] repeating once
  else if [digital read pin P13] == 0 and [sprite_x] != 7
  then change sprite_x by 1

  display clear
  display set matrix color at x [sprite_x] y [sprite_y] to [sprite_colour]
  display show
  pause (ms) 100
```

## MicroPython Editor Code

Dit programma is gemaakt in de MicroPython Mu-editor voor de BBC micro:bit. Het biedt precies dezelfde functionaliteit als het MakeCode Blocks programma.

```
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 NeoPixels 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)
```