:GAME ZIP™ 64 för BBC micro:bit

www.kitronik.co.uk/5626

:GAME ZIP™64 är en programmerbar speldosa för BBC micro:bit. Den har 64färgadresserbara LED:er arrangerade i en 8x8-skärm, en piezo-sändare för ljudåterkoppling, en vibrationsmotor för haptisk återkoppling och 6 ingångsknappar. Den bryter också ut P19, P20 och LED DOUT till standard 0,1”-footprint. Vart och ett av dessa stift har också den erforderliga spänningen och GND-kuddar. BBC micro:bit är ansluten via en standard kortplats med kantanslutning.

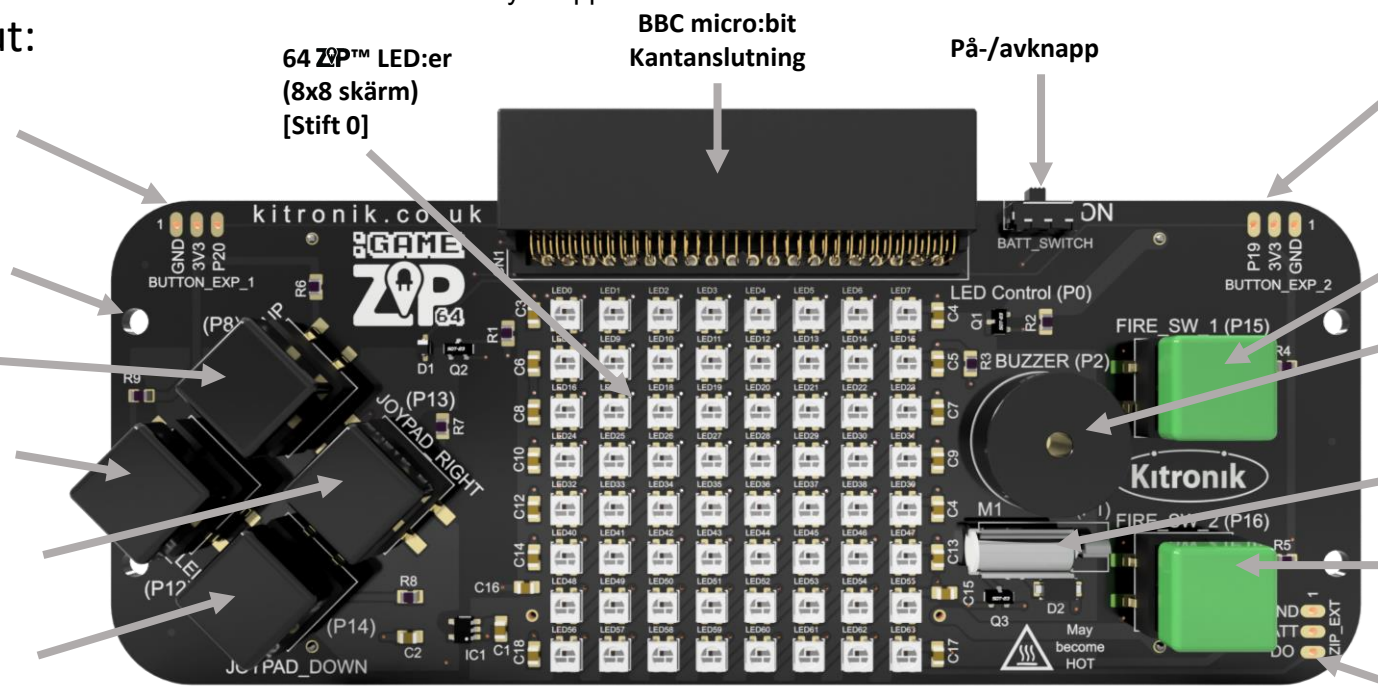
Kortet producerar en **reglerad försörjning** som matas in i 3V- och GND-anslutningar **för att driva den anslutna BBC micro:biten**, vilket tar bort behovet för separat ström till BBC micro:bit. För att skydda BBC micro:bit om strömmen försörjs genom den kommer ZIP™ LED:en inte att lysa upp

Sätta in ett BBC micro:bit:

För att använda :GAME ZIP™ 64, bör BBC micro:bit sättas in ordentligt i kantanslutningen, se till att BBC micro:bitens LED-skärm är vänd i samma riktning som :GAME ZIP™ 64:s LED-display.

Exempel: För nybörjarspel och idéer för andra saker du kan göra, besök: http://www.kitronik.co.uk/5626

Kortets layout:



Varning:

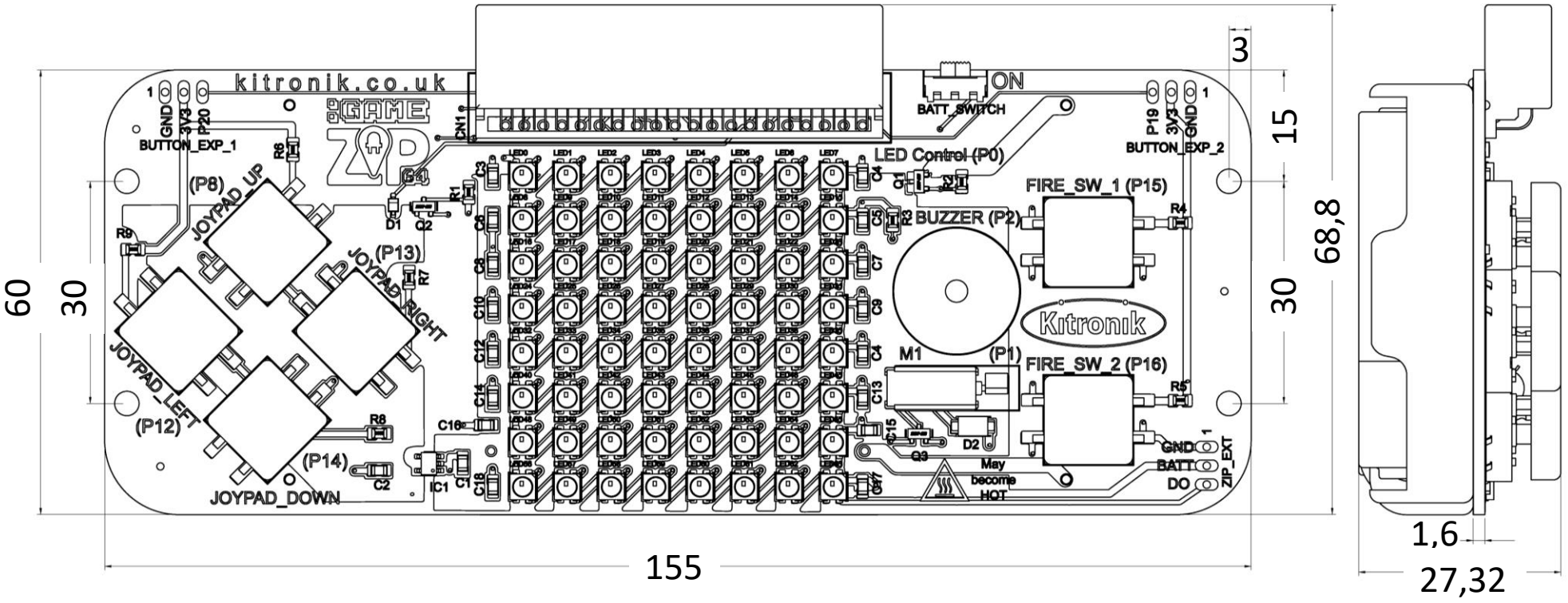
ZIP™ LED:er kan bli varma om de används med hög ljusstyrka under längre perioder.

:GAME ZIP™ 64 för BBC micro:bit

www.kitronik.co.uk/5626

Kortets dimensioner:

(Alla mått är angivna i mm)



:GAME ZIP™ 64 för BBC micro:bit

www.kitronik.co.uk/5626

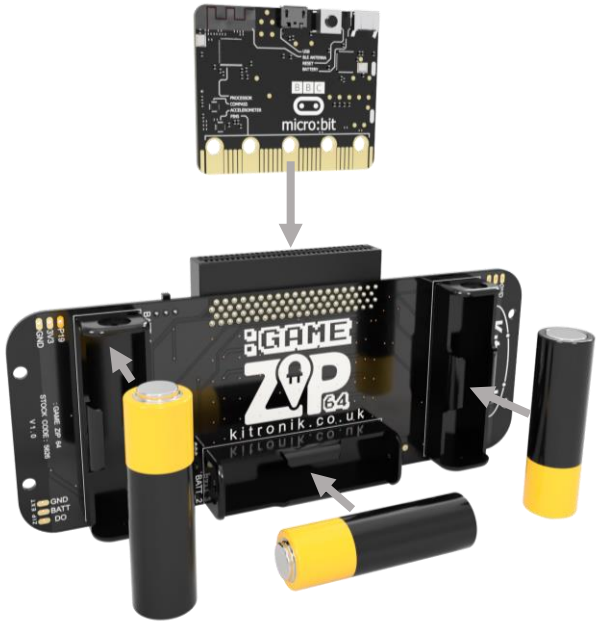
Elektrisk information

Driftspänning (Vcc) [ZIP-LED:er]	+3,5V – +5,3V
Reglerad spänning [BBC micro:bit, knappar, vibrationsmotor]	+3,3V
Maxström (ZIP-LED:er som körs med full RGB-ljusstyrka)	1,6A (21mA per ZIP-LED, 250mA max på +3,3V vanlig spänning)
Antal ZIP-LED	64
Antal externa kanaler	3 (1xZIP-LED, 2 x I2C/IOstift, varje IO-stift graderad +3,3V vid 5mA)

Varning för externa kanaler:

Försiktighet bör vidtas iakttagas vid användning av de yttre kantanslutningarna för stift 19 och 20 som GPIO, eftersom det kan orsaka problem med I2C-enheterna på själva BBC micro:biten (t.ex. kompass och accelerometer).

Bakifrån med BBC micro:bit & batterier:

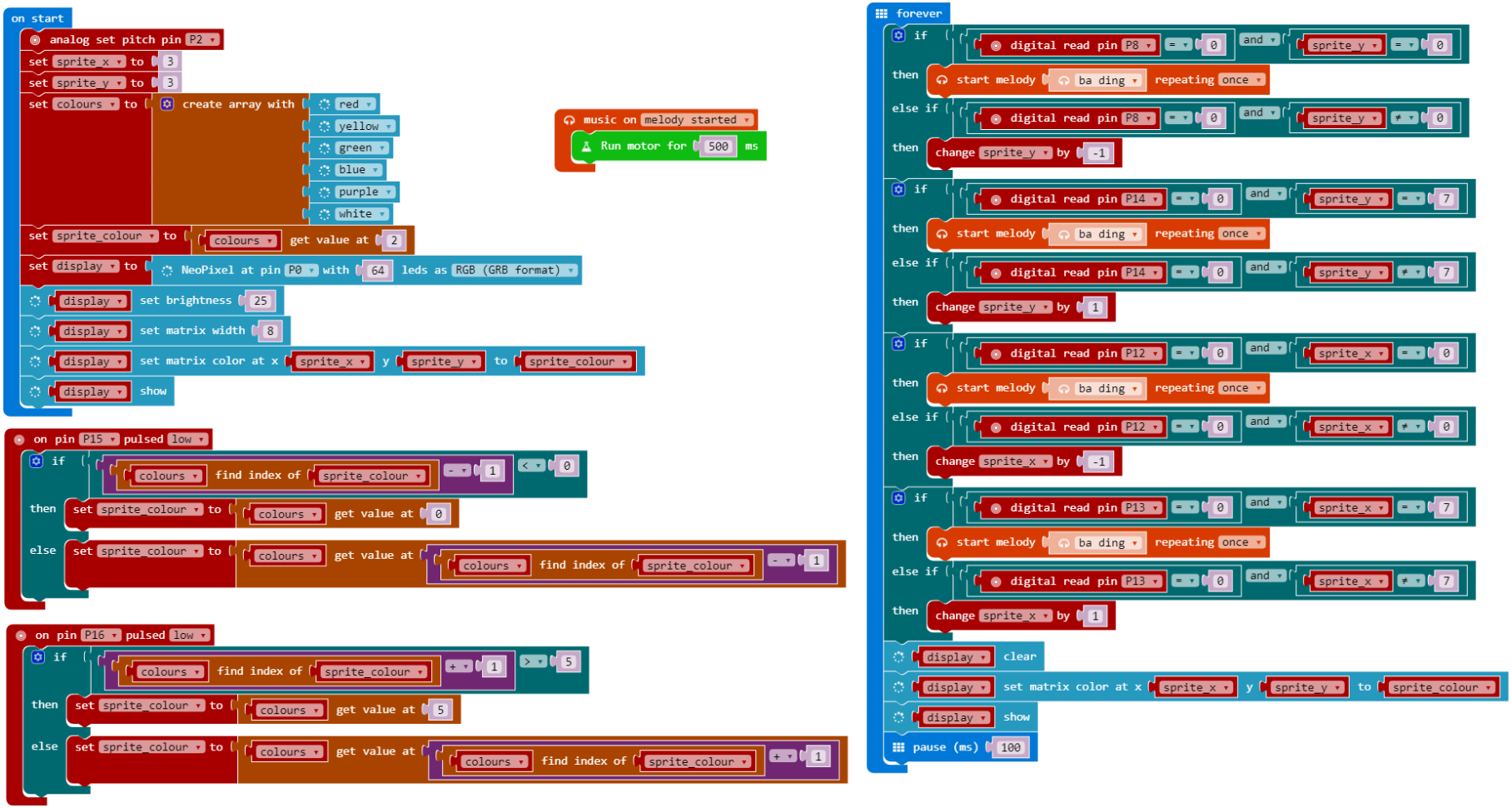


:GAME ZAP™ 64 för BBC micro:bit

www.kitronik.co.uk/5626

Microsoft MakeCode Blocks Editor Code

Detta program skapades i Microsoft MakeCode Blocks Editor för BBC micro:bit. Det skapar en enda pixelsprite som kan flyttas runt på skärmen med Joypad-knapparna och färgen kan ändras med hjälp av avfyrningsknapparna. När spriten når skärmens kant kommer motorn att vibrera och sändaren kommer att spela upp en kort melodi. **Observera: Det finns Kitronik Custom Blocks tillgängliga för :GAME ZAP™ 64 på Microsoft MakeCode (t.ex. 'Run motor' som används här).**



:GAME ZIP™ 64 för BBC micro:bit

www.kitronik.co.uk/5626

MicroPython Editor Code

Detta program skapades i MicroPython Mu Editor för BBC micro:bit. Det ger exakt samma funktionalitet som programmet MakeCode Blocks.

```
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)
```