:GAME ZIP™ 64 til BBC micro:bit'en

www.kitronik.co.uk/5626

:GAME ZIP™ 64 er en programmerbar gamepad til BBC micro:bit. Den har 64 farvejusterbare lysdioder arrangeret i et 8 x 8-display, en piezo-brummer til lydfeedback, en vibrationsmotor til haptisk feedback og 6 indgangsknapper. Den outbreaker også P19, P20 og LED DOUT til standard 0,1"-footprints. Hver af disse stifter har også den påkrævede spænding og GND-blokke. BBC micro:bit'en er tilsluttet via et standard-kantstik med indstiksrum.

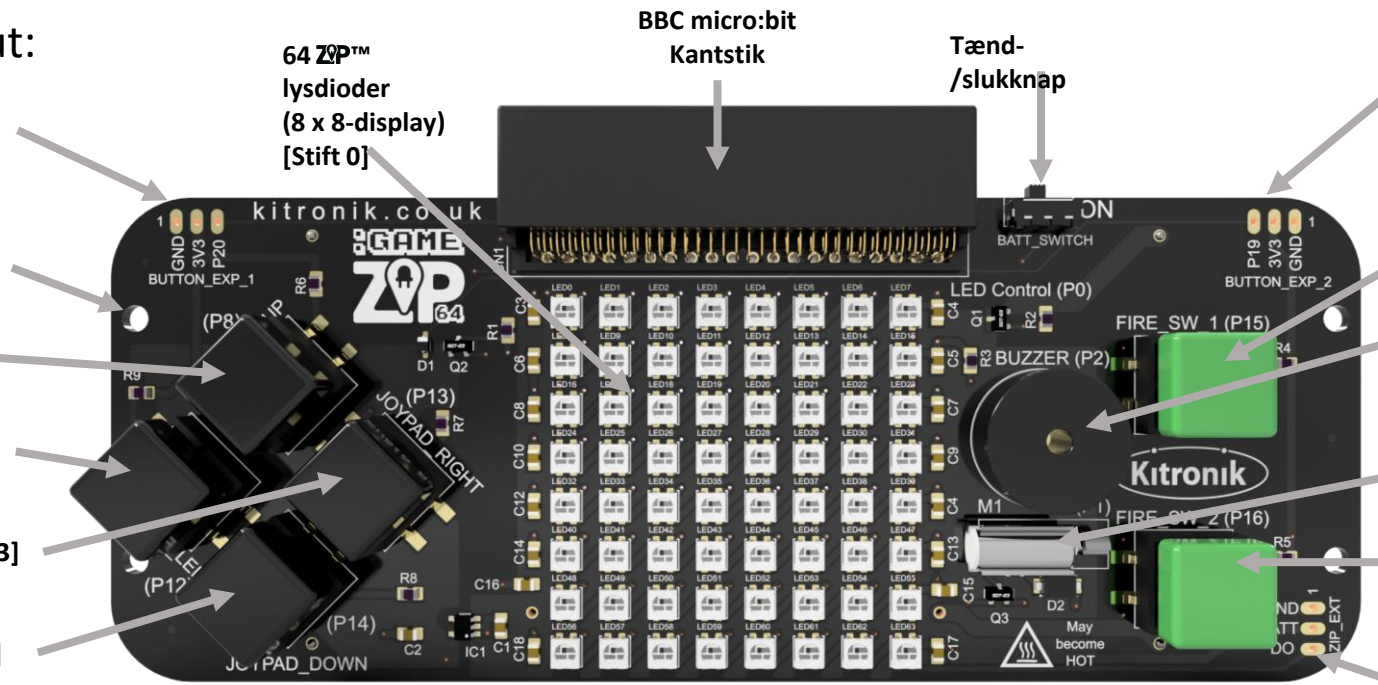
Kortet producerer en reguleret forsyning, der føres ind i 3V- og GND-forbindelserne for at give den tilsluttede BBC-micro:bit strøm, hvilket fjerner behovet for separat at forsyne BBC-microbit'en med strøm. For at beskytte BBC micro:bit'en vil ZIP™-lysdioderne ikke lyse, hvis strømforsyningen går gennem dem.

Sådan isættes en BBC-micro:bit:

For at bruge :GAME ZIP ™ 64 skal BBC micro:bit'en sættes ordentligt ind i kantstikket, mens man sikrer sig, at BBC micro:bit LED-displayet vender i samme retning som :GAME ZIP ™ 64 LED-displayet.

Eksempler: For spil og ideer til at komme i gang kan du gå ind på: http://www.kitronik.co.uk/5626

Kortets layout:



Advarsel:

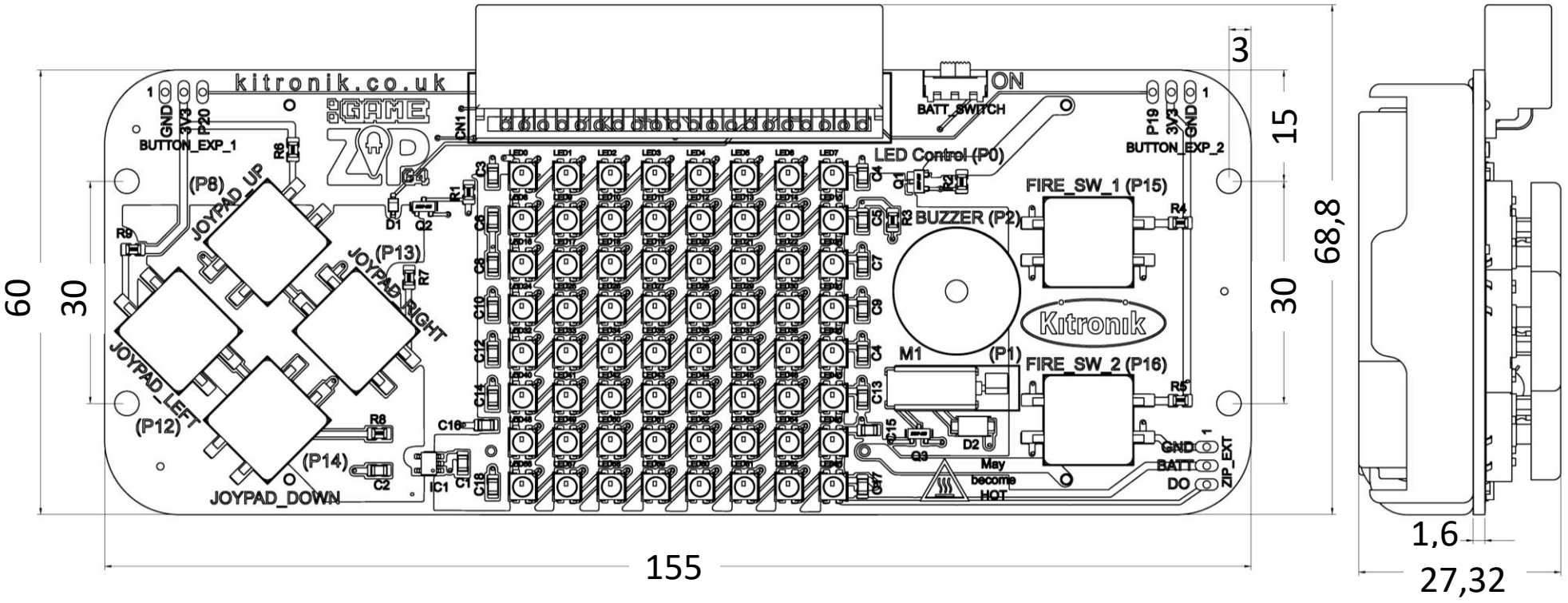
ZIP™ lysdioder kan blive varme, hvis de bruges ved høj lysstyrke over længere perioder.

:GAME ZAP™ 64 til BBC micro:bit'en

www.kitronik.co.uk/5626

Kortets mål:

(Alle mål er angivet i mm)



:GAME ZIP™ 64 til BBC micro:bit'en

www.kitronik.co.uk/5626

Elektriske oplysninger

Driftsspænding (Vcc) [LED-lysdioder]	+3.5V – +5.3V
Reguleret spænding [BBC micro:bit, knapper, vibrationsmotor]	+3.3V
Maksimal strøm (ZIP-lysdioder ved fuld RGB-lysstyrke)	1,6A (21mA pr. ZIP-lysdiode, 250mA max på +3,3V reg. spænding)
Antal ZIP-lysdioder	64
Antal eksterne kanaler	3 (1 x ZIP-lysdiode, 2 x I2C/IO stift, hver IO-stift har nominel spænding +3.3V ved 5mA)

Bemærk på eksterne kanaler:

Der skal udvises forsigtighed ved brug af eksterne breakouts til stifterne 19 og 20 som GPIO'er, da dette kan skabe problemer med I2C-enhederne på selve BBC-micro:bit'en (f.eks. kompas og accelerometer).

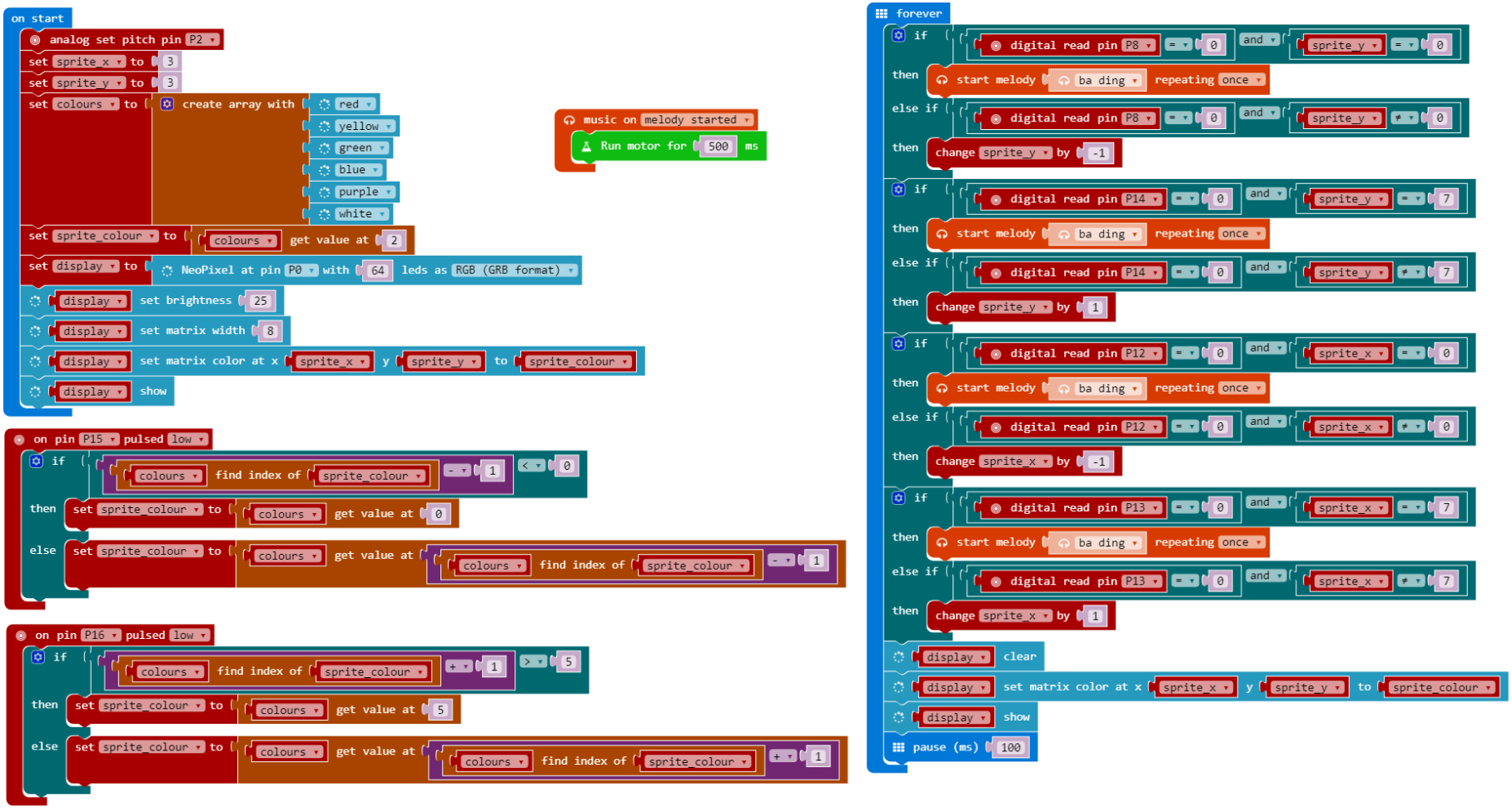
Set bagfra med BBC micro:bit og batterier:

:GAME ZAP™ 64 til BBC micro:bit'en

www.kitronik.co.uk/5626

Microsoft MakeCode Blocks Editor Code

Dette program blev oprettet i Microsoft MakeCode Blocks Editor til BBC micro:bit. Det skaber en enkelt pixel-sprite, som kan flyttes rundt på displayet ved hjælp af joypad'ens knapper og kan få ændret sin farve ved hjælp af affyringsknapperne. Når spriten når displaykanten, vibrerer motoren, og brummeren afspiller en kort melodi. **Bemærk: Der er Kitronik brugerdefinerede blokke til rådighed til :GAME TM 64 pZAPMicrosoft MakeCode (f.eks er 'Kør motor' brugt her).**



:GAME ZIP™ 64 til BBC micro:bit'en

www.kitronik.co.uk/5626

MicroPython Editor Code

Dette program blev skabt i MicroPython Mu Editor til BBC micro:bit'en. Det giver nøjagtig samme funktionalitet som MakeCode Blocks-programmet.

```
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)
```