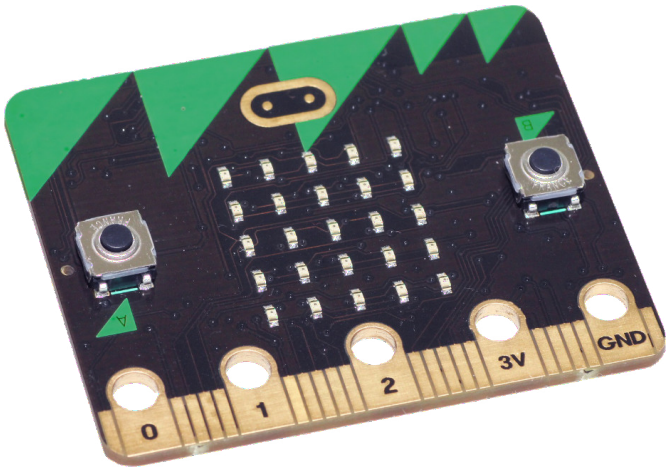


BBC microbit

1 Uvod

Napravica microbit je odprtokodni sistem zgrajen na osnovi procesorja ARM, ki jo je zasnoval BBC za pomoč pri računalniškem izobraževanju v Veliki Britaniji. Prvič so jo predstavili ob kampaniji Make It Digital, ki jo je pripravil BBC 12. marca 2015, ko so dijakom v Veliki Britaniji podarili milijon napravic. Končna zasnova in funkcije napravice so bile predstavljene 6. julija 2015, medtem, ko se je dejanska dobava napravic začela februarja 2016.

Napravica je velika kot polovica kreditne kartice. Njeno srce je procesor ARM Cortex-M0, ima senzor za merjenje pospeška, magnetometer, povezljivost z Bluetooth in USB. Zaslon sestavlja 25 LED diod, ima dva gumba, ki ju je mogoče programirati. Lahko se napaja bodisi preko USB, bodisi z zunanjo baterijo. Vhodi in izhodi naprave potekajo preko pet obročnih konektorjev, ki so del večjega 25-polnega nabora konektorjev.

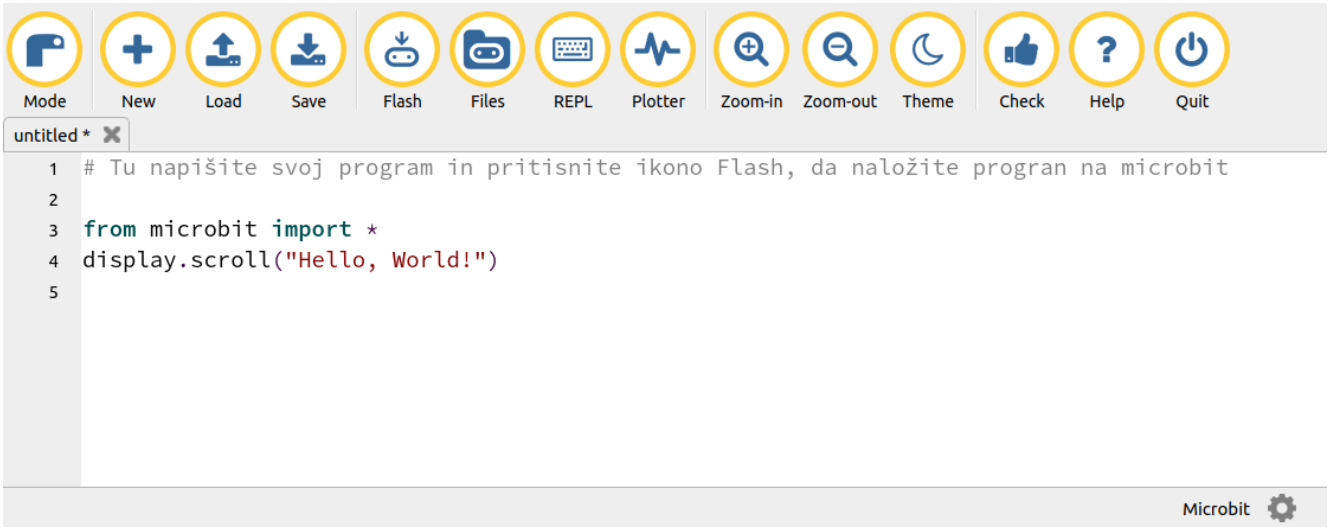
2 Komunikacija

Priključite napravico preko USB na vaš računalnik. Za komunikacijo priporočamo, da uporabljate mu-editor. Program in navodila za instalacijo boste našli na njihovi spletni strani.

https://codewith.mu
Navodila najdete tudi na strani
https://codewith.mu/en/tutorials/1.0/microbit

`Mu editor` teče na treh napogostejših platformah: `Windows`, `OSX` in `Linux`. Ko je `mu-editor` instaliran, priključite `microbit` na računalnik preko vtičnice `USB`, zaženete `mu-editor` in izberite opcijo `microbit`.

Napišite vaš program in kliknite na gumb `Flash`, da prenesete program na napravico.



Slika 1: Pozdrav svetu

Poleg tega pa obstaja tudi spletna verzija urejevelnika `mu-editor`, ki je dosegljiva na naslovu `https://python.microbit.org/v/1.1`

V tem primeru morate program, ki ste ga napisali na spletni verziji, najprej shranit na vaš računalnik, nato pa ta program naložite na napravico, ki se pojavi na vašem sistemu kot zunanji disk, brž ko ste jo priklučili preko `USB`.

3 Python

Programski jezik `Python` je eden najpopularnejših programskih jezikov. Naj omenimo, da je bila programska koda za krmiljenje instrumetov, ki so zaznali gravitacijske valove zapisana v programskem jeziku `Python`. Ko se boste spoznali s tem programskim jezikom vam bodo odprta vrata v različna področja programerskega udejstvovanja.

Eno od teh je krmiljenje napravice `microbit`. Na njem teče verzija programskega jezika `Python` imenovana `MicroPython`. Prilagojena je za majhne računalnike, kot je na primer `microbit`. Je popolna implementacija verzije `Python 3`, zato boste z lahkoto presedlali na drugo strojno opremo, kot je na primer `Raspberry Pi`.

`MicroPython` ne vsebuje vseh standardnih knjižnic jezika `Python`. Ima pa posebne knjižnice, ki so prilagojene strojni opremi na kateri teče. `MicroPython` ima na napravici `microbit` posebno knjižnico `microbit` za delo s strojno opremo napravice.

4 Primeri

4.1 Hello World

Zapišimo program, ki bo izpisal sloviti stavek *pozdrav svetu* v angleškem jeziku *Hello World*. To je po navadi prvi program, ki se ga napiše v nekem programskem jeziku. Ta navada sega v začetke računalniške dobe. Prvi je bil Brian Kernighan, ki je napisal ta program kot del dokumentacije za programski jezik `BCPL` v Bellovih laboratorijih leta 1972.

Hello World

```
from microbit import *
display.scroll("Hello, World!")
```

Pomen verstic:

Prva vrstica: `from microbit import *`

pove interpreterju `MicroPython`, da naj naloži vse potrebno za delo s strojno opremo napravice `BBC microbit`. Zvezdica `*` pomeni, da kličemo funkkcije iz knjižnice z izvornim imenom brez predpone imena knjiznice. Na primer funkcijo izpis kličemo kot `display` brez predpone `microbit.display`, ki bi jo morali napisati v primeru, če bi naložili knjiznico `microbit` z ukazom:

Prva vrstica*: `import microbit`

Druga vrstica: `display.scroll("Hello, World!")`

uporabi program `display.scroll` za izpis črk besedila `"Hello, World!"`. Prvi del imena programa `display` je ime knjižnice za delo s prikazovalnikom (matrike diod) napravice. Program `scroll` se nahaja v knjižnici

`display`. Oba dela imena sta ločena s piko. Imenu programa `scroll` dodamo še v oklepajih `()` besedilo, ki ga želimo izpisati. Besedilo zapišemo v navednicah `"`.

V `mu editor` zapiši program, kot prikazuje slika 1 in klikni na ikono `Flash` oziroma `Download`. V drugem primeru morate še shraniti program na napravico.

Če bo šlo kaj narobe se bo `MicroPython` pritožil in izpisal na prikazovalnik številko vrstice in tip napake `SyntaxError`.

4.2 Ikone

`MicroPython` ima vnaprej definirano množico slikic (*ikon*), ki jih lahko prikažemo na matriki (zaslonu) 5×5 svetlečih `LED` diod.

Prikaz ikone, ki predstavlja smeška.

Happy

```
from microbit import *
display.show(Image.HAPPY)
```

Prva vrstica je enaka kot v prejšnjem programu.

Druga vrstica: V drugi vrstici smo uporabili funkcijo, za prikazovanje v naprej definiranih slikic (ikon) `show`, iz knjižnice `display`. V oklepaju je zapisano ime slikice `HAPPY`, ki se nahaja v skupini `Image`. Seznam v naprej definiranih slikic:

1. `Image.HEART`
2. `Image.HEART_SMALL`
3. `Image.HAPPY`
4. `Image.SMILE`
5. `Image.SAD`
6. `Image.CONFUSED`
7. `Image.ANGRY`
8. `Image.ASLEEP`
9. `Image.SURPRISED`
10. `Image.SILLY`
11. `Image.FABULOUS`
12. `Image.MEH`
13. `Image.YES`
14. `Image.NO`
15. `Image.CLOCK12`
16. `Image.CLOCK11`
17. `Image.CLOCK10`
18. `Image.CLOCK9`
19. `Image.CLOCK8`
20. `Image.CLOCK7`
21. `Image.CLOCK6`
22. `Image.CLOCK5`
23. `Image.CLOCK4`
24. `Image.CLOCK3`

25. `Image.CLOCK2`
26. `Image.CLOCK1`
27. `Image.ARROW_N`
28. `Image.ARROW_NE`
29. `Image.ARROW_E`
30. `Image.ARROW_SE`
31. `Image.ARROW_S`
32. `Image.ARROW_SW`
33. `Image.ARROW_W`
34. `Image.ARROW_NW`
35. `Image.TRIANGLE`
36. `Image.TRIANGLE_LEFT`
37. `Image.CHESSBOARD`
38. `Image.DIAMOND`
39. `Image.DIAMOND_SMALL`
40. `Image.SQUARE`
41. `Image.SQUARE_SMALL`
42. `Image.RABBIT`
43. `Image.COW`
44. `Image.MUSIC_CROTCHET`
45. `Image.MUSIC_QUAVER`
46. `Image.MUSIC_QUAVERS`
47. `Image.PITCHFORK`
48. `Image.XMAS`
49. `Image.PACMAN`
50. `Image.TARGET`
51. `Image.TSHIRT`
52. `Image.ROLLERSKATE`
53. `Image.DUCK`
54. `Image.HOUSE`
55. `Image.TORTOISE`
56. `Image.BUTTERFLY`
57. `Image.STICKFIGURE`
58. `Image.GHOST`
59. `Image.SWORD`
60. `Image.GIRAFFE`
61. `Image.SKULL`
62. `Image.UMBRELLA`
63. `Image.SNAKE`

Imena so seveda v angleščini. Če jih ne razumete vzemite v roke slovar.

4.3 Lastne slikice

`MicroPython` vam omogoča, da si sami definirate svoje slikice.

Vsako `LED` diodo (piksel) v matriki lahko krmilimo posebej tako, da ji določimo svetlost. Vrednosti za svetlost grejo od `0` dioda je ugasnjena do `9` ko sveti najmočneje.

Kako določimo in prikažemo svojo slikico?

Čoln

```
from microbit import *

boat = Image("05050:05050:05050:99999:09990")
display.show(boat)
```

4.4 Animacije

Na matriki diod lahko prikazujemo tudi animacije. Za animacijo je potrebno določiti zaporedje slikic.

`MicroPython` pozna v ta namen podatkovno strukturo, ki se imenuje `list` *seznam*. Elemente seznama naštejemo znotraj oglatih oklepajev `[]`. Posamezne elemente znotraj seznama ločimo z vejico `,`. Primer seznama:

`primes = [2, 3, 5, 7, 11, 13, 17, 19]`.

V seznamu je zapisanih prvih 8 praštevil.

Opomba

Števila ne zapišemo v navednicah.
`MicroPython` loči zapis `2` (številska vrednost 2) in `"2"` (znak za številko 2).

V seznam lahko zapišemo elemente različnih tipov. Na primer

`mixed_up_list = ["hello!", 1.234, Image.HAPPY]`

`MicroPython` zna animirati seznam slikic. Nekaj seznamov slikic je že definiranih v naprej. Na primer: `Image.ALL_CLOCKS` in `Image.ALL_ARROWS`. Poglejmo primer animacije.

Ura

```
from microbit import *
display.show(Image.ALL_CLOCKS, loop=True, delay=100)
```

`MicroPython` bo prikazoval v zanki `loop` slikice iz seznama `Image.ALL_CLOCKS`, vsaka slikica bo prikazana na matriki 100 ms oziroma 0.1 s. To povemo z dodatnima podatkoma (*argumentoma*) `loop=True` in `delay=100`.

Utrip

```
from microbit import *

srce = [Image.HEART,Image.HEART_SMALL]

display.show(srce, loop=True, delay=600)
```

Primer je animacije utripa srca. Dve sličici, eno manjšo in drugo večjo, srca izmenjujemo na matriki diod. Pri tem uporabimo neskončno zanko `loop=True` in `delay=600` premor, trajajoč 600 ms.

Naredimo še lastno animacijo.

Animacija

```
from microbit import *

boat1 = Image("05050:05050:05050:99999:09990")
boat2 = Image("00000:05050:05050:05050:99999")
boat3 = Image("00000:00000:05050:05050:05050")
boat4 = Image("00000:00000:00000:05050:05050")
boat5 = Image("00000:00000:00000:00000:05050")
boat6 = Image("00000:00000:00000:00000:00000")

all_boats = [boat1, boat2, boat3, boat4, boat5, boat6]

display.show(all_boats, delay=200)
```

4.5 Gumbi

Do sedaj, smo se ukvarjali s sporočanjem rezultatov programov, ki so tekli na napravici, preko *izhoda* `output`, ki je bila matrika `LED` diod. Napravica ima še druge možnosti sporočanja v svet. Vendar o tem kasneje. Sedaj pa si poglejmo najpreprostejši primer *vhoda* `input` v napravico. Kako napravica sprejema sporočila iz okolice. V tem poglavju se bomo osredotočili na gumbe. Napravica `microbit` ima dva gumba oznmačena z `A` in `B`. `MicroPython` lahko zazna položaj gumbov.

Stikalo

```
from microbit import *

sleep(10000)

display.scroll(str(button_a.get_presses()))
```

Počakamo 10000 ms oziroma 10 s nato pa sporočimo, kolikokrat smo medtem pritisnili gumb `A`. Funkcija `button_a.get_presses()` vrne število, ki ga spremenimo v znakovni niz `str`, da ga lahko izpišemo na prikazovalnik t.j. na matriko diod.

S pomočjo *zanke* `while` (*medtem ko ponavljaj*) čakamo na izpolnitev pogoja ali na dogodek iz kolice, ki ga zazna napravica.

```
from microbit import *

x=0
while x < 10:
    display.scroll(str(x))
    x+=1
```

Na začetku je vrednost spremenljivke `x` enaka 0. Program ostane v zanki, dokler ne postane vrednost spremenljivke `x` enaka 10. Medtem se v zanki vsakokrat izpiše vrednost `x` in njena vrednost poveča za 1.

Kako v programu preberemo položaj gumov `A` in `B` in ustrezno odgovorimo nanj.

```
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    elif button_b.is_pressed():
        break
    else:
        display.show(Image.SAD)

display.clear()
```

V neskončni zanki čakamo da se pritisne eden od gumbov. Če pritisnemo na gumb `A` se prikaže nasmejan obraz, sicer pa je obraz prikazan na prikazovalniku žalosten. Ko pritisnemo gumb `B` se diode na prikazovalniku ugasnejo in program se zaključi.

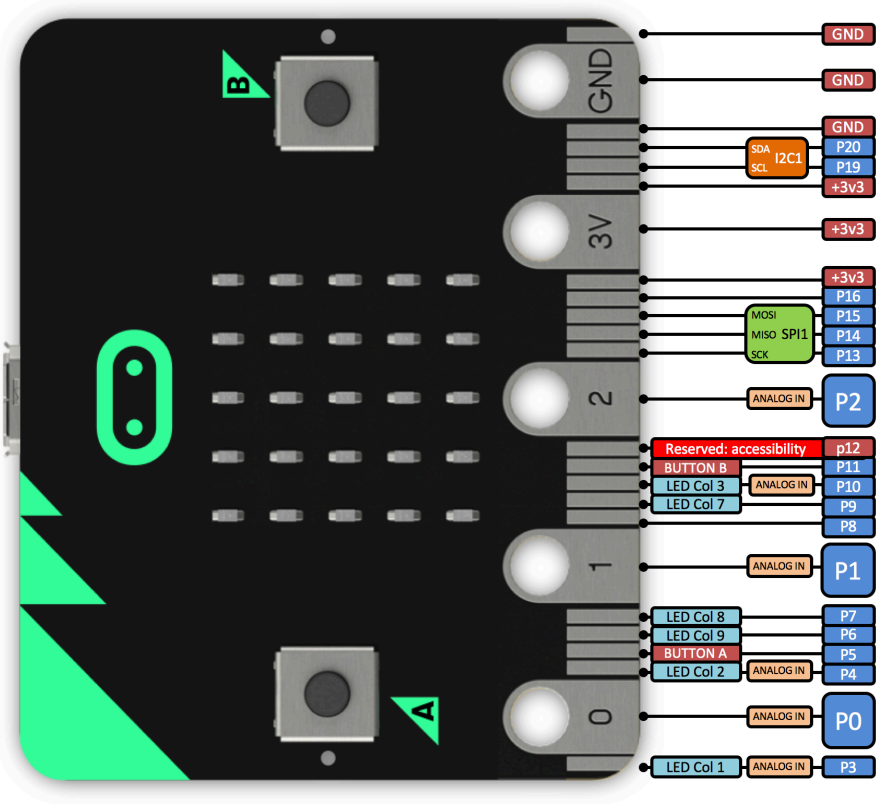
4.6 Pini `I/O`

Pini so namenjeni za komunikacijo napravice `BBC microbit` z okoljem (`I/O input/output`). Na sliki 2 je na kratko opisana njihova funkcija.

Nekaj od njih je širših, označeni so `0, 1, 2, 3V, GND`. Na njih se lahko priključimo s pomočjo krokodilčka.

Prvi trije pini so vhodno/izhodni podatkovni pini zadnja dva sta uporabljena za napajanje periferije.

Za priključitev na ostale pine potrebujemo poseben priključek (glej sliko 3)



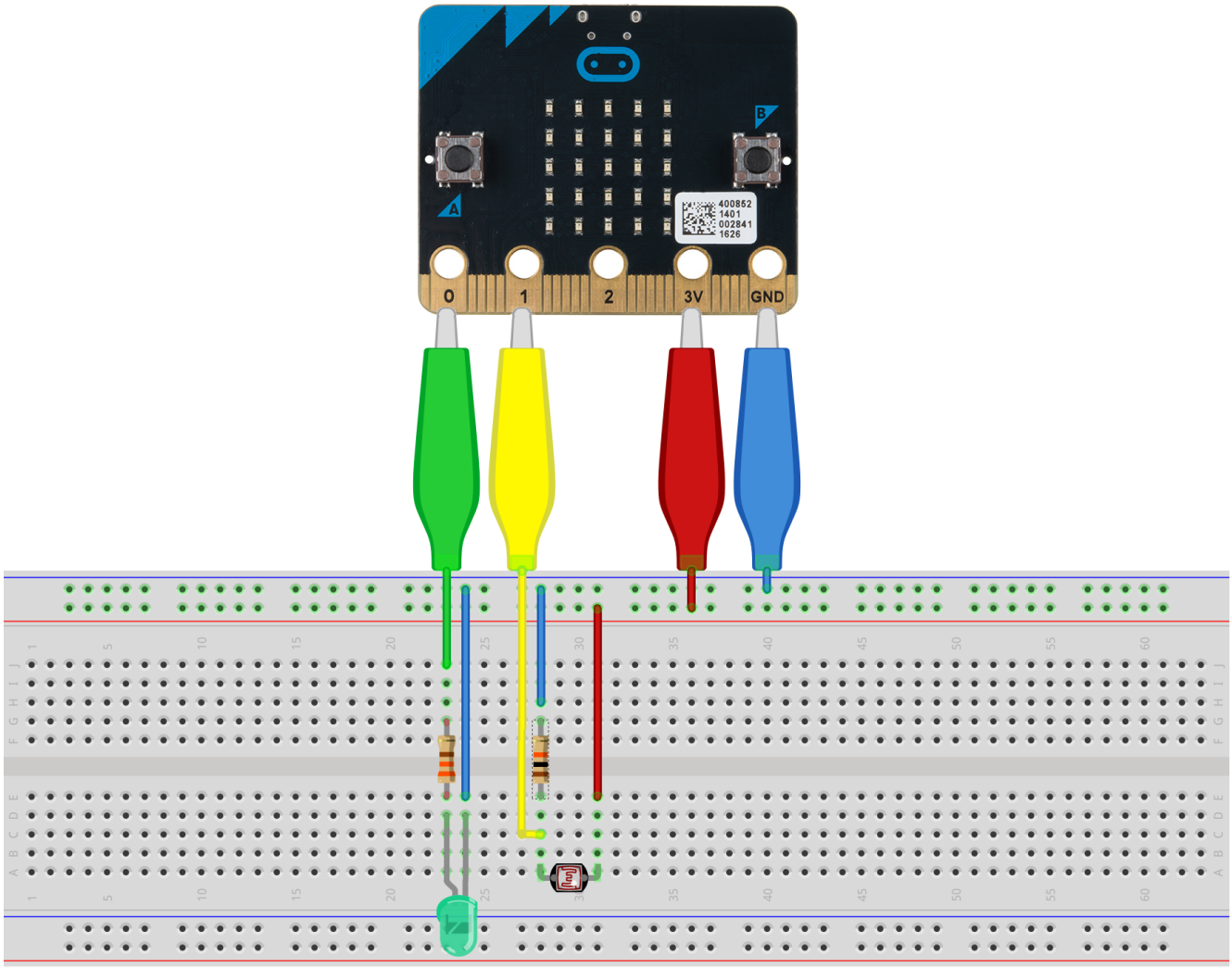
Slika 2: Razporeditev pinov

Slika 3: Priključki na pine

Vsak pin napravice je predstavljen z objektom `pinN` kjer je `N` številka pina. Na primer pin številka `0` se imenuje `pin0`.

Razložimo funkcijo širokih pinov. Pin `GND` je ozemljen medtem, ko je pin `3V` namenjen napajanju. Na njem je napetrost $3.3\,\mathrm{V}$. Ostale tri pine lahko krmilimo programsko. Oštevilčeni so z `0`, `1` in `2`. Namenjeni so komunikaciji z okolico `I/O`. Med drugim se lahko napetost na pinu med 0 in $3.3\,\mathrm{V}$, preko analogno-digitalnega pretvornika `ADC` spremeni v številsko vrednost med 0 in 1023.

Pini lahko delujejo v digitalnem ali analognem načinu. Smer informacije je lahko branje `input` ali pisanje `output`. V digitalnem načinu pini pišejo ali berejo digitalne vrednosti `0` ali `1` oziroma `False` ali `True`, v analognem načinu pa tečejo vrednosti od `0` do `1023`.



Slika 4: Krmiljenje `LED` diode in branje analogne vrednosti

Digitalni način

V digitalnem načinu sporočamo informacijo preko pinov z dvema vrednostima `False` in `True`.

Funkcija: `pinN.write_digital(value)`,

kjer je vrednost `value` enaka `False` oziroma `0` ali `True` oziroma `1` medtem, ko je `N` številka pina. Pri vrednosti `0` je pin ozemljen, na njem je napetost enaka 0, pri vrednosti `1` pa je napetost na pinu enaka $3.3\,\mathrm{V}$.

Funkcija: `pinN.read_digital()` vrne vrednost 0, če je na pinu napetost manj kot $0.8\,\mathrm{V}$ in vrne vrednost 1, če je na pinu napetost več kot $2.0\,\mathrm{V}$. Vmes je vrednost, ki jo dobimo, naključna.

Analogni način

Funkcija: `pinN.write_analog(value)`,

kjer je vrednost `value` med 0 in 1023, postavi napetost na pinu `N` na vrednost $3.3 \times \mathrm{value}/1023\,\mathrm{V}$.

Funkcija: `pinN.read_analog()`,

vrne celoštevilsko vrednost med 0 in 1023, v odvisnosti od napetosti, ki je priključena na pin `N`.

Dioda in branje analogne vrednosti

```
from microbit import *

pin1.write_digital(1)
sleep(10000)
pin1.write_digital(0)

while True:
    a = pin2.read_analog()
    display.scroll(str(a))
```

Najprej program za $10\,\mathrm{s}$ prižge diodo, potem pa prebere napetost na svetlobnem senzorju in sporoči vrednost, ki jo je prebral. Svetlobnemu senzorju se spreminja upor glede na intenziteto vpadle svetlobe.

Funkcija: `pinN.is_touched()` zazna dotik.

Naslednji program bo zaznal dotik pina `0`. Na začetku se prikaže ikona z žalostnim obrazom, ko pa se hkrati dotaknemo z eno roko pina `0` in z drugo roko pina `GND`, se prikaže ikona z veselim obrazom.

Funkcioja Dotik

```
from microbit import *

while True:
    if pin0.is_touched():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

4.7 Naključja

Kadar želimo v program vnesti naključja uporabljamo funkcije iz knjižnice `random`. Primer programa, ki naključno izbere ime iz seznama in ga izpiše na prikazovalniku.

```
from microbit import *
import random

names = ["Mary", "Yolanda", "Damien", "Alia", "Kushal", "Mei Xiu", "Zoltan" ]
display.scroll(random.choice(names))
```

Seznam `names` vsebuje nize z imeni. Funkcija `random.choise(names)` pogleda v seznam `names` in vrne naključni niz iz seznama. Ta niz se izpiše na koncu na prikazovalniku.

Napiši program, kjer zamenjaš imena v seznamu z imeni po lastni izbiri.

Naključna števila se pogosto uporabljajo v igricah. Spomnimo se na metanje kocke pri različnih igrah.

Knjižnica `random` vsebuje še druge funkcije, ki vračajo naključna čtevila. Naslednji program izpiše naključno število med 1 in 6.

```
from microbit import *
import random

display.show(str(random.randint(1, 6)))
```

Vsakokrat ko pritisnemo gumb reset, ki se nahaja na zadnji strani napravice se izpiše naključno izbrano

število med `1` in `6`. Sedaj lahko napravico uporabljamo namesto kocke.

Če potrebujemo naključno število med `0` in `N-1`, potem uporabimo funkcijo `random.randrange(N)`. Bodite pozorni, da funkcija vrača vrednosti med `0` do `N-1`, vključno `0` in `N-1`, število `N` ni vključeno.

Funkcija `random.random()` vrača naključne realne vrednosti med 0.0 in 1.0 vključno z 0.0 in 1.0.
Za večja realna števila si pomagamo s funkcijama `random.randrange` in `random.random` kot na primer:

```
from microbit import *
import random

answer = random.randrange(100) + random.random()
display.scroll(str(answer))
```

Generatorji naključnih števil, ki jih uporabljajo računalniki ne vračajo resnično naključna števila, so le navidezno naključna. Dodatno naključje uvedemo s pomočjo *semen*, ki inicializirajo generatorje naključnih števil. Seme naj bo čim bolj naključno. Za to uporabljamo vrednost, ki jo preberejo iz trenutnega časa in/ali iz vrednosti senzorjev, kot so vgrajeni termometri itd.

Včasih pa želimo ponovljiva zaporedja naključnih števil. V tem primeru pa seme nastavimo sami s funkcijo `random.seed(N)` Število `N` je celo število. Naslednja verzija programa generira vedno enako zaporedje naključnih števil.

```
from microbit import *
import random

random.seed(1337)
while True:
    if button_a.was_pressed():
        display.show(str(random.randint(1, 6)))
```

Vsakokrat ko pritisnemo gumb `A` dobimo novo naključno število. Ko pritisnemo gumb `reset` se bo s pritiskanjem na gumb `A` zaporedje ponovilo.

4.8 Gibanje

Napravica `microbit` ima vgrajen pospeškometer.

Pospeškometer meri gibanje vzdolž treh osi:

- X - nagib levo desno.
- Y - nagib naprej nazaj.
- Z - gibanje gor in dol.

Funkcija `accelerometer.get_XYZ()`, kjer je niz `XYZ` enak `x`, `y` ali `z`. Funkcija vrne pozitivno ali negativno število. Meritve so izražene v tisočinkah težnostnega pospeška na Zemlji $g = 9.8\text{m/s}^2$. Če je vrednost enaka 0, pomeni, da je napravica poravnana z dano osjo.

Primer programa, ki meri poravnanost napravice vzdolž osi X.

Uporabimo funkcijo `accelerometer.get_x()`.

```
from microbit import *

while True:
    reading = accelerometer.get_x()
    if reading > 20: display.show("R")
    elif reading < -20: display.show("L")
    else: display.show("-")
```

Če je napravica poravnana z osjo X potem se na prikazovalniku izriše. `-` sicer pa pove na katero stran je nagnjena. Izpis `L` pomeni nagnjenost na levo medtem, ko `R` pomeni nagnjenost na desno.

Želimo, da se napravica stalno odziva na premike, zato smo teste zaprli v neskončno zanko. Ker je pospeškometer zelo občutljiv smo dločili prag odziva ± 20 mili-g.

Poleg te funkcije obstajata še funkciji `get_y` in `get_z` ki meri nagib oziroma pomik v osi Y oziroma Z.

Gibi

S pomočjo pospeškometra ima napravica implementirano funkcijo, ki ki zaznava posebne gibe.

Program lahko zazna naslednje gibe

1. `'up'`, gor
2. `'down'`, dol
3. `'left'`, levo
4. `'right'`, desno

5. `'face up'`, obraz navzgor
6. `'face down'`, obraz navzdol
7. `'freefall'`, prosti pad
8. `'3g'`, $3g$
9. `'6g'`, $6g$
10. `'8g'`, $8g$
11. `'shake'`, strese

Gibi so izpisani kot nizi. Gibi `'3g'`, `'6g'` in `'8g'` pomenijo, da je pospeškometer zaznal pospeške $3g$, $6g$ oziroma $8g$, kjer je g težni pospešek na Zemlji $g = 9.8\,\mathrm{m/s^2}$.

Gibe v programu razpoznamo s pomočjo funkcije `accelerometer.current_gesture()`. Funkcija vrne niz z imenom ustreznega giba.

```
from microbit import *

while True:
    gesture = accelerometer.current_gesture()
    if gesture == "face up":
        display.show(Image.HAPPY)
    else:
        display.show(Image.ANGRY)
```

V neskončni zanki testiramo ali je obraz obrnjen navzgor in, če je prikažemo vesel obraz, sicer pa je obraz žalosten. `Python` uporablja znak `==` za testiranje enakosti, medtem, ko se znak `=` uporablja za prireditev vrednosti spremenljivkam.

4.9 Radio

Dve napravici se lahko pogovarjata preko radijske povezave. Radijsko povezavo prižgemo s pomočjo funkcije `radio.on()` in jo ugasnemo z `radio.off()`. Na razpolago imamo 101 kanal za vzpostavitev radijske povezave, da se lahko izognemo sporočilom, ki niso namenjena nam. Kanali so označeni s števili od 0 do 100. Kanal nastavimo s pomočjo funkcije `radio.config(channel=N)`, kjer je `N` številka kanala. Moč oddajanja nastavimo z `radio.config(power=N)`, kjer je `N` število med 1 do 7.

Sporočila pošiljamo s pomočjo funkcije `radio.send("sporocilo")` medtem, ko sporočila sprejemamo z `new_message = radio.receive()`

Sporočila se zapisujejo v čakalno vrsto `message queue`. Funkcija `receive` vrne najstarejše sporočilo v vrsti

in sprosti prostor za nova sporočila. Če je čakalna vrsta polna potem, se prihajajoča sporočila ignorira.

Pošiljanje

```
from microbit import *
import radio
radio.on()
radio.config(channel=19) # Kanal komunikacije
radio.config(power=7) # Jakost oddajanega signala je maksimalna

display.show('SEND')
sleep(500)
display.clear()

while True:
    if button_a.is_pressed():
        radio.send('A')
    elif button_b.is_pressed():
        radio.send('B')
    sleep(500)
```

Sprejem

```
from microbit import *
import radio
radio.on()
radio.config(channel=19) # Kanal komunikacije
radio.config(power=7) # Jakost oddajanega signala je maksimalna

display.show('RECEIVE')
sleep(500)
display.clear()

while True:
    incom = radio.receive()
    if incom:
        display.show(incom)
    sleep(500)
```

5 Naloge

5.1 Gumbi

Program, ki bo na pritisk gumba `A` odgovoril `"A"` in na pritisk na gumb `B` odgovoril z `"B"` sicer pa bo prikazal sliko srčka.

Gumbi

```
from microbit import *

while True:
  if button_a.is_pressed():
    display.scroll("A")
  elif button_b.is_pressed():
    display.scroll("B")
  else:
    display.show(Image.HEART)
```

5.2 Igralna kocka

Program, ki bo simuliral metanje kocke. Ko pritisnemo gumb `A` se prikaže naključna stranica igralne kocke.

Kocka 1

```
from microbit import *
import random

kocka = [Image("00000:00000:00900:00000:00000"),
         Image("00000:09000:00000:00090:00000"),
         Image("00000:09000:00900:00090:00000"),
         Image("00000:09090:00000:09090:00000"),
         Image("00000:09090:00900:09090:00000"),
         Image("00000:09990:00000:09990:00000")]

while True:
    if button_a.is_pressed():
        display.show(random.choice(kocka))
```

5.3 Igralna kocka drugič

Popravimo program tako, da se bo napravica odzvala na tresenje.

Kocka 2

```
from microbit import *
import random

kocka = [Image("00000:00000:00900:00000:00000"),
         Image("00000:09000:00000:00090:00000"),
         Image("00000:09000:00900:00090:00000"),
         Image("00000:09090:00000:09090:00000"),
         Image("00000:09090:00900:09090:00000"),
         Image("00000:09990:00000:09990:00000")]

while True:
    display.show(Image.HEART)
    if accelerometer.was_gesture('shake'):
        display.clear(); sleep(1000)
        display.show(random.choice(kocka)); sleep(3000)
```

5.4 Pika

Program, ki ob stresenju napravice prižge naključno izbrano diodo.

Pika 1

```
from microbit import *
import random

while True:
    display.show(Image.HEART)
    if accelerometer.was_gesture('shake'):
        display.clear(); sleep(1000)
        x=random.randrange(0,5); y=random.randrange(0,5)
        display.set_pixel(x,y,9); sleep(3000)
```

5.5 Pika drugič

Napišimo program, ki prižiga po eno diodo na matriki diod tako, da daje vtis kot, da bi se svetla točka pomikala po zaslonu glede na to, kako nagibamo napravico

Program naj zazna nagib napravice v smeri X in Y osi in prižge ustrezno diodo na zaslonu.

Funkcija `add(t,d)` prišteje vrednosti t vrednost $+1$, če je $d > 0$ in -1, če je $d < 0$, zraven pa še poskrbi, da vrednost spremenljivke t vedno ostaja v mejah $0 \leq t \leq 4$, ne gre čez rob.

V zanki gledamo nagib v smeri `X` in `Y` in s pomočjo funkcije `add` popravljamo trenutni položaj pike.

Pika 2

```
from microbit import *
import random, math

def add(t,d):
    if abs(d)>100: t+=int(math.copysign(1,d))
    if t<0: t=0
    if t>4: t=4
    return(t)

x, y = 2, 2
display.set_pixel(x,y,9)

while True:
    xg=accelerometer.get_x()
    yg=accelerometer.get_y()
    display.clear()
    x=add(x,xg);
    y=add(y,yg);
    display.set_pixel(x,y,9)
    sleep(100)
```

5.6 Kača

Zaporedje prižganih diod predstavlja kačo. Z nagibanjem napravice se kača premika po matriki diod napravice. Program deluje podobno kot pri piki, le da piki v glavi kače sledijo še tri pike, ki predstavljajo telo

in rep kače. Funkcija `add` skrbi za premik glave kače, medtem ko funkcija `prikazi` pomakne še telo in rep kače tako, da sledita glavi.

Funkcija `display.set_pixel(x,y,9)` prižge diodo v matriki na mestu `(x,y)`.

Funkcija `int(math.copysign(1,d))` vrne $+1$, če je $d > 0$ in -1, če je $d < 0$.

Kača

```
from microbit import *
import random, math

kaca = [[0,2],[1,2],[2,2],[3,2]]

def add(t,d):
    if abs(d)>100: t+=int(math.copysign(1,d))
    if t<0: t=0
    if t>4: t=4
    return(t)

def prikazi(p):
    global kaca
    kaca = kaca[1:]+[p]
    display.clear()
    for x, y in kaca:
        display.set_pixel(x,y,9)

x, y = kaca[-1]
while True:
    xg=accelerometer.get_x()
    yg=accelerometer.get_y()
    x=add(x,xg);
    y=add(y,yg);
    prikazi([x,y])
    sleep(100)
```

5.7 Zunanja dioda

Dioda

```
from microbit import *

shake = False
while True:
    if shake:
        pin0.write_digital(1)
        display.show(Image.SQUARE)
    else:
        pin0.write_digital(0)
        display.clear()
    if accelerometer.was_gesture('shake'):
        shake = not shake
        sleep(500)
```