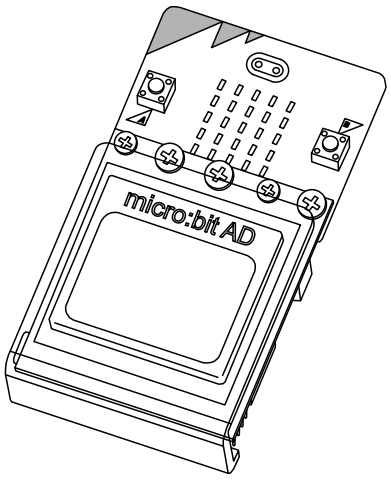
Python programiranje

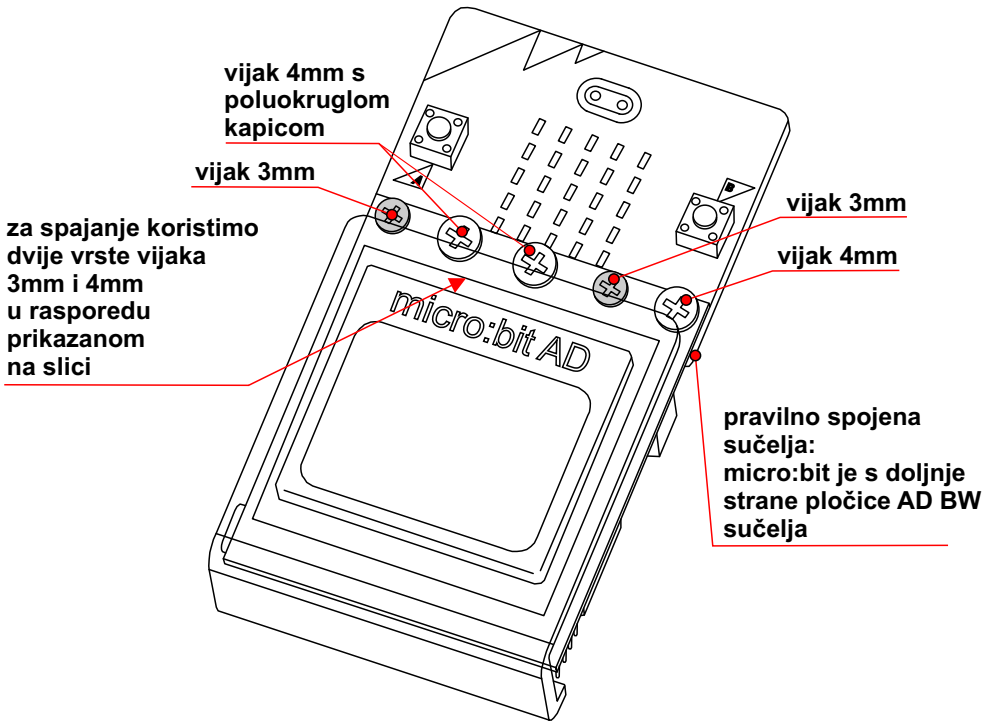
micro:bit AD BW



DIDACTA ADVANCE d.o.o.
Čakovec, Croatia (EU)
www.didacta.hr

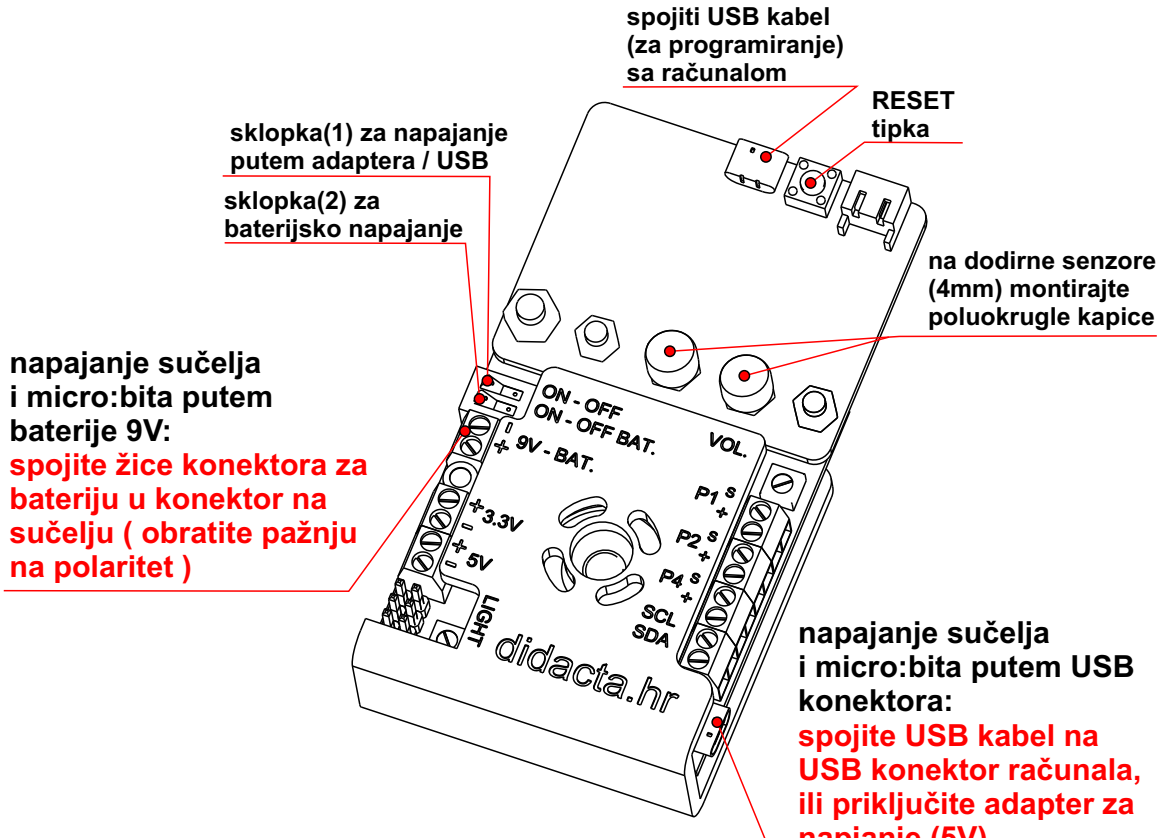
1. POČETAK

1.1. Spajanje micro:bita i AD sučelja vijcima



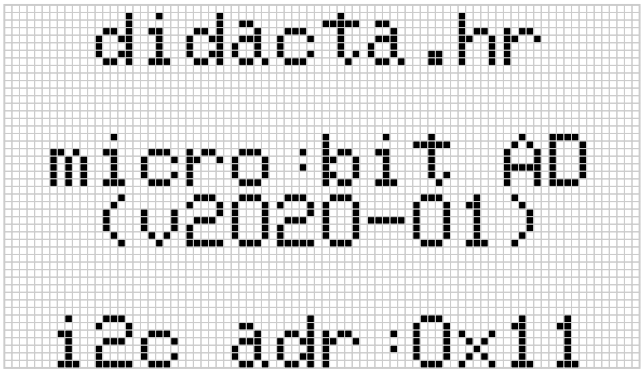
Manje vijke koristimo na pozicijama na kojima bi moglo doći do oštećenja elemenata na micro:bit pločici, a koji se nalaze blizu otvora za spajanje.

1.2. napajanje micro:bita i AD sučelja putem baterije, adaptera ili USB porta (PC)



1.3. Pokretanje sučelja micro:bit AD

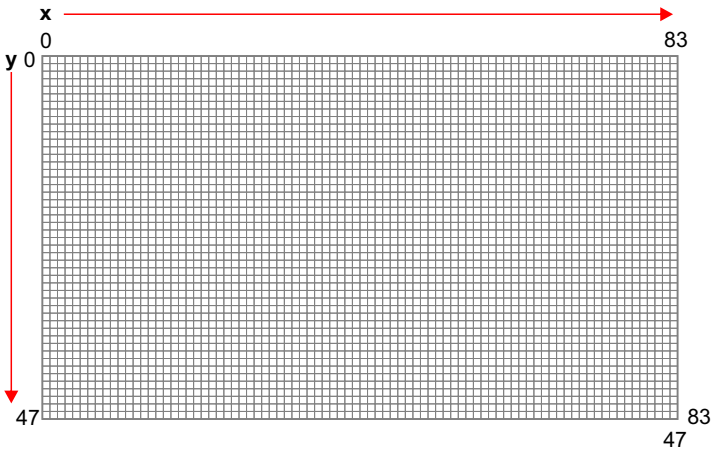
Nakon pokretanja, na ekranu sučelja micro:bit AD BW će se ispisati tekst prikazan na Slici 1.
Sučelje je spremno za rad.
Ako je već učitan program u micro:bit, za pokretanje trebate pritisnuti tipku RESET na micro:bitu.



slika 1.

1.4. Ekran sučelja

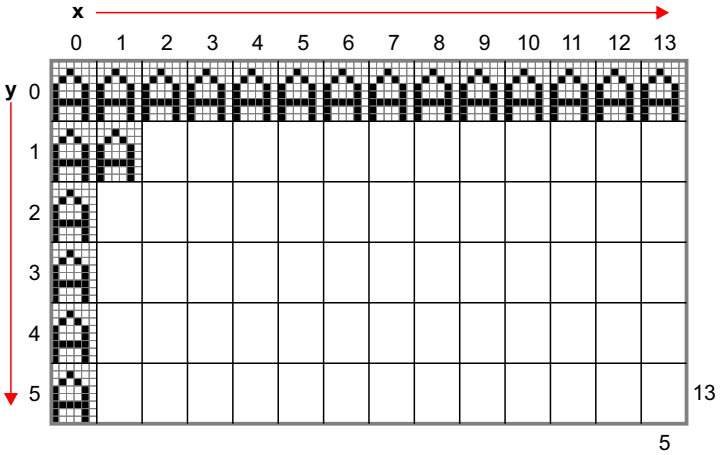
Micro:bit AD BW sučelje ima crno-bijeli ekran grafičke rezolucije 48 x 84 piksela (Slika 2.).



Slika 2. Grafički mod - rezolucija 48 x 84.

Za ispis teksta (tekst mod) se koristi rezolucija koja odgovara veličini fonta (znaka). Veličina standardnog tekstualnog znaka (slova - fonta je 7 x 5 piksela) je 8 x 6 piksela sa pikselima za razmak. Zato je rezolucija za ispis teksta 6 x 14 znakova (Slika 3.). Prilikom izrade programa moramo voditi računa koje programske naredbe koristimo i za koji mod rada su one namjenjene.

Grafičke funkcije koriste grafičku rezoluciju (linija, kružnica, ravokutnik,...) , a tekstualani mod se koristi kod standardnog ispisa teksta (ne grafičkog) i kod definiranja ekrana za igru, te pozicioniranja objekta igrača.



Slika 3. Tekstualni mod - rezolucija 6 x 14.

2. MicroPython (MU)

2.1. MicroPython

MicroPython (MU) možete skinuti putem donjeg linka i instalirati na svoje računalo:

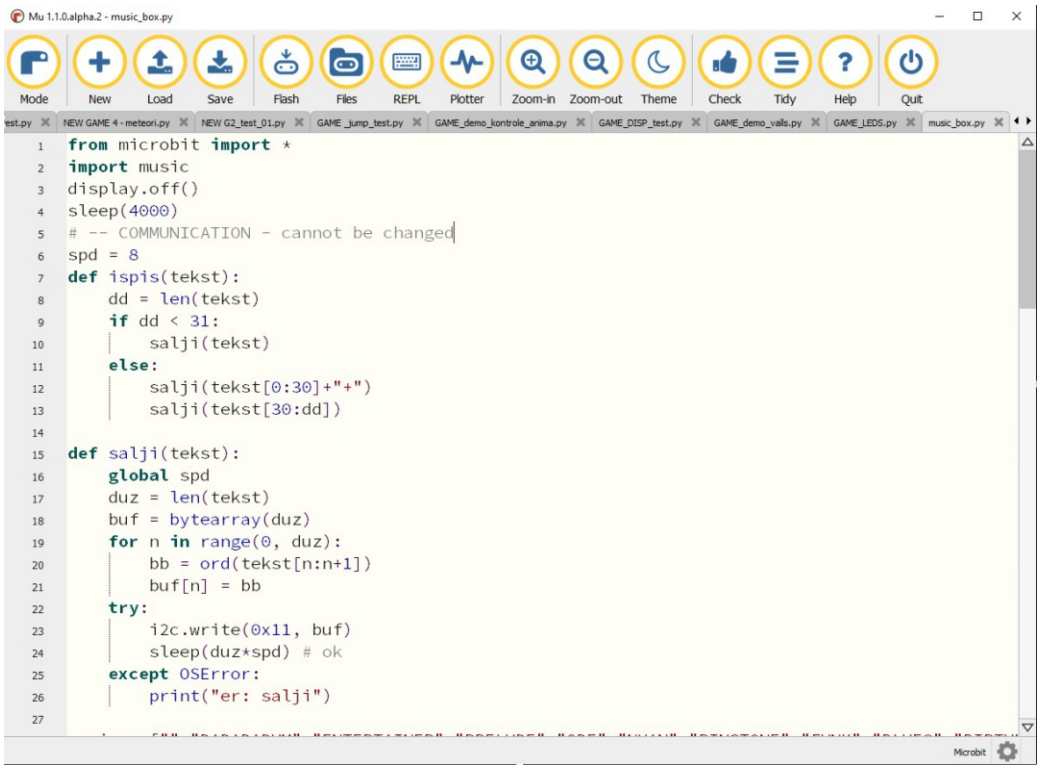
https://codewith.mu/en/download

Ako ne želite instalirati program na svoje računalo, možete koristiti verziju koja radi u web pretraživaću:

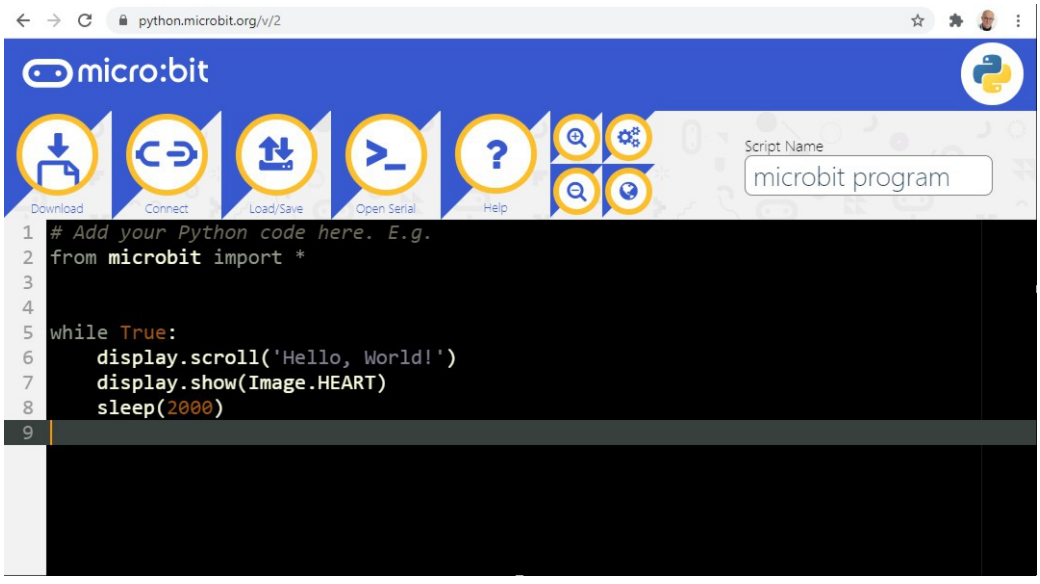
https://python.microbit.org/v/2

Upute za programiranje možete naći na adresi:

https://microbit.org/get-started/user-guide/python



Slika 4. MU - MicroPython instalirani



Slika 5. MicroPython putem web pretraživaća

2.2. Osnovni Moduli

2.2.1. Moduli za komunikaciju između micro:bita i sučelja AD. (**NE MJENJATI**)

Ovi moduli definiraju načim preijenosa podataka između micro:bita i sučelja AD BW, te ih treba koristiti **bez mjenjanja**. Ukoliko mijenjate neke vrijednosti ili dijelove modula, može se desiti da komunikacija između micro:bita i sučelja AD **prestane raditi**. **OVAJ MODUL JE OBAVEZAN**.

```
pit= bytearray(1)
pit[0] = 1
spd = 8
def ispis(tekst):
    dd = len(tekst)
    if dd < 31:
        salji(tekst)
    else:
        salji(tekst[0:30]+"+")
        salji(tekst[30:dd])

def salji(tekst):
    global spd
    duz = len(tekst)
    buf = bytearray(duz)
    for n in range(0, duz):
        bb = ord(tekst[n:n+1])
        buf[n] = bb
    try:
        i2c.write(0x11, buf)
        sleep(duz*spd)
    except OSError:
        print("er: salji")
```

Slika 6.

2.2.2. Modul za **RESET** programa. (**NE MJENJATI**)

Ovaj modul resetira (postavlja na početna stanja) program koji radi na sučelju AD BW. Ovaj modul je **preporučeno koristiti** (pokrenuti - rest()) na početku svakog programa, da se ne desi mješanje starih podataka koji su spremljeni u memoriju sučelja AD sa novim podacima. **NIJE OBAVEZAN**.

```
def rest():
    global pit
    ispis("RST")
    while True:
        trazi()
        if pit[0] == 5:
            break
        sleep(20)
```

Slika 7.

2.2.3. Modul za očitavanje podatka od sučelja AD. (**NE MJENJATI**)

Ovaj modul potražuje vrijednost od sučelja AD. Vrijednosti se koriste za određivanje događaja u igri, kao što su: osvajanje boda, gubitak života ili pad. Ako ne radite igru sa zvučnim ili svjetlosnim efektima ovaj modul **NIJE OBAVEZAN**.

```
def trazi():
    global pit
    try:
        pit = i2c.read(0x11,1) # AD BW display adresa
    except OSError:
        print("er: trazi")
```

Slika 8.

2.2.4. Modul za zvučne i svjetlosne efekte . (**MOŽE SE MJENJATI**)

Ovaj modul se koristi kod programa za igru. Vrijednosti koje se ispituju u modulu su određene i ne mogu se mjenjati. **Mijenjati se mogu** dijelovi koji su **ZELENE** boje. Ako želite koristiti ovaj modul, morate uključiti i prethodni modul (trazi() - koji očitava vrijednosti) u program. **NIJE OBAVEZAN** .

```
def zvuk():
    global pit
    if pit[0] == 2:  # POINT
        music.pitch(1500, 50)
        ispis("LED;R;30")
    if pit[0] == 3:  # LIFE
        music.pitch(800, 100)
        ispis("LED;G;30")
    if pit[0] == 4:  # FALL
        for freq in range(900, 1200, 30):
            music.pitch(freq, 8)
```

Slika 9.

2.2.5. Modul za poruku na POČETKU rada programa. (**MOŽE SE MJENJATI**)

Ovaj modul ispisuje poruku na početku rada programa i može se izmjeniti prema vašim potrebama.

```
def begin():
    ispis("CLS")
    ispis("START;2;1;2")
    ispis("G A M E;3;3")
    sleep(1000)
    ispis("CLS")
```

Slika 10.

2.2.6. Modul za poruku na KRAJU rada programa. (**MOŽE SE MJENJATI**)

Ovaj modul potražuje vrijednost od sučelja AD. Vrijednosti se koriste za prikaz broj bodova (Score) na kraju rada programa. Ako se u programu osvajaju bodovi, može se mijenjati samo dio modula zelene boje. Ostali dio modula očitava broj bodova i prikazuje ga ekranu. Varijable pozicije ispisa, naziv i poruku za kraj možete promijeniti ili dopuniti po želji.

```
def end():
    global pit
    sleep(100)
    trazi()
    ispis("CLS")
    ispis("E N D;2;1;2")
    ispis("G A M E;3;3")
    ispis("Score:"+str(pit[0])+";3;4")
    while True:
        sleep(5000)
```

Slika 11.

3. PROGRAMIRANJE - OSNOVNE FUNKCIJE

3.1. Prvi program - funkcija RST

Funkciju koristimo na početku programa, a nakon osnovnih modula. Pokretanjem ove funkcije brišu se vrijednosti u svim poljima koje koristi program AD sučelja. U sklopu **rest()** modula - 2.2.2 Strana 4.

ispis("RST")

Slika 12. Funkcija RST

3.2. Prvi program - ispis teksta ”HELLO” - funkcija TEXT

U prvom programu koristimo funkciju **TEXT** (slika 6.). Zadana vrijednost za veličinu (s) je **1** pa se funkcija može pozivati i bez upisa vrijednosti.

ispis("TEXT;x;y;s;c") **ili** **ispis("TEXT;x;y;c")**

TEXT = tekst koji treba ispisati
x = 0-13 text mod
y = 0-5 text mod
s = veličina 1 - 3 (1)
c = Boja (B-crna, W-bijela)

Slika 13. Funkcija TEXT (tekst mod)

Na početku svakog programa dobro je koristiti funkciju **RST** programa koja briše vrijednosti koje je program sučelja micro:bit AD koristio u prethodnom radu programa na micro:bitu.

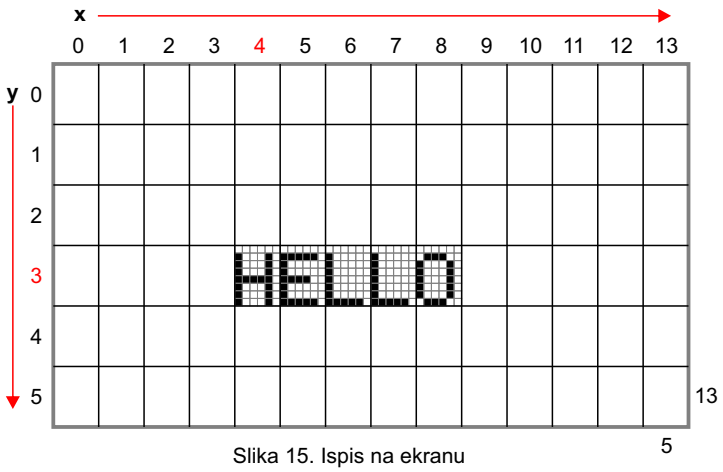
Učitajte program u micro:bit putem komande **FLASH**. Na ekranu bi se trebao ispisati tekst **HELLO** na poziciji upisanih vrijednosti (Slika 14.).

```
from microbit import *
import music
#  2.04.2021

# -- COMMUNICATION - start
pit= bytearray(1)
pit[0] = 1
spd = 8
def ispis(tekst):
    dd = len(tekst)
    if dd < 31:
        salji(tekst)
    else:
        salji(tekst[0:30] + "+")
        salji(tekst[30:dd])

def salji(tekst):
    global spd
    duz = len(tekst)
    buf = bytearray(duz)
    for n in range(0, duz):
        bb = ord(tekst[n : n + 1])
        buf[n] = bb
    try:
        i2c.write(0x11, buf)
        sleep(duz * spd)
    except OSError:
        print("er: send")
# -- COMMUNICATION --- end
```

```
# --
def trazi():
    global pit
    try:
        pit = i2c.read(0x11,1)
    except OSError:
        print("er: seek")
# --
def rest():
    global pit
    ispis("RST")
    while True:
        trazi()
        if pit[0] == 5:
            break
        sleep(20)
# --
rest()
while True:
    ispis("HELLO!;1;1;B")
    ispis("HELLO!;2;2;2;B")
```

Slika 14. Program za ispis teksta HELLO (prog01.py)

Osnovni dio programa do WHILE TRUE je isti za sve primjere koji se obrađuju.



Slika 15. Ispis na ekranu

Ispis na ekranu nakon pokretanja prvog programa Slika 15.
Upisom vrijednosti 2 ili 3 u crveno polje, možete ispisati tekst veći od standardne dimenzije (Slika 16.).

ispis("TEXT;x;y;s;c")

TEXT = tekst koji treba ispisati
x = 0-13 text mod
y = 0-5 text mod
s = veličina 1 - 3 (1)
c = Boja (B-crna, W-bijela)

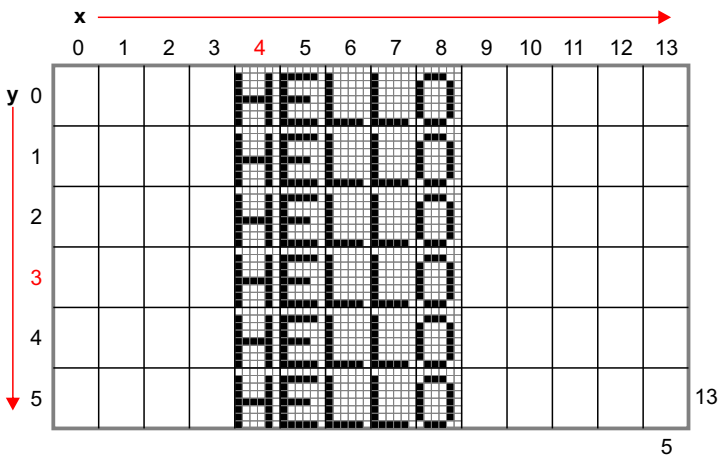
Slika 16. veličina ispisa

3.3. Višestruki ispis teksta putem funkcije for

U ovom programu koristimo funkciju za ponavljanje - **for** i varijablu **n** koju koristimo kao **y** vrijednost u funkciji **TEXT**. Napravite promijenu u prethodnom programu prema slici 17.

```
while True:
    for n in range(0, 6):
        ispis("HELLO!;4;"+str(n)+";B")
```

Slika 17. Višestruki ispis teksta (prog02.py)



Slika 18. Ispis na ekranu

3.4. Ispis teksta na dvije pozicije i brisanje ekrana - CLS

U ovom programu koristimo funkciju **TEXT** i funkciju za brisanje ekrana **CLS** (Slika 19.).

ispis("CLS")

Slika 19. Funkcija CLEAR SCREEN - brisanje ekrana

Programu dodamo još jednu funkciju **TEXT** i upišemo vrijednosti prema primjeru. Nakon prve funkcije dodamo **pauzu** od pola sekunde (500 ms). To ponovimo još jedan puta i upišemo druge vrijednosti u novu funkciju, prema primjeru na slici 20.

```
while True:
    ispis("HELLO!;4;1;B")
    sleep(500)
    ispis("HELLO!;4;3;B")
    sleep(500)
```

Slika 20. Program za ispis teksta HELLO na dvije pozicije (prog03.py)

Dodajte u program funkciju za brisanje ekrana - **CLS** i učitajte program.

```
while True:
    ispis("HELLO!;4;1;B")
    sleep(500)
    ispis("CLS")
    ispis("HELLO!;4;3;B")
    sleep(500)
    ispis("CLS")
```

Slika 21. Program za ispis teksta HELLO na dvije pozicije (prog04.py)

U čemu je razlika kod prikaza na ekranu ?

Maknite **CLS** funkciju iz programa. Upišite jednake vrijednosti za poziciju u obe funkcije **TEXT**, te drugoj funkciji promijenite boju ispisa (B -> W).

Koji je rezultat ove promijene ?

3.5. Ispis teksta u grafičkom modu (G)

U ovom programu koristimo funkciju **TEXT** za ispis teksta u grafičkoj rezoluciji (grafički mod Slika 2.). Putem ove funkcije možemo ispisati tekst na bilo kojoj poziciji na ekranu.

Ova funkcija ne ispisuje tekst direktno na ekran već u pomoćnu memoriju (**BUFFER**), zato se nakon jedne ili više **TEXT(Graphics)** funkcija mora izvršiti funkcija **BUF** koja prikazuje zapis iz pomoćne memorije na ekranu. Zadane vrijednosti su za **s = 1** i **c = B** pa ih nije potrebno uvijek navoditi.

ispis("TEXT;x;y;s;c;G") ili ispis("TEXT;x;y;G")

TEXT = tekst koji treba ispisati
x = 0-83 grafički mod
y = 0-47 grafički mod
s = veličina 1 - 3 (**1**)
c = Boja (B-crna, W-bijela) (**B**)
G = grafički mod (84 x 48)

Slika 22. Funkcija TEXT u grafičkom modu

```
ispis("BUF")
```

Slika 23. Funkcija za ispis pomoćne memorije na ekranu

```
while True:
    ispis("HELLO!;4;1;B;G")
    ispis("BUF")
```

Slika 24. Primjer program za ispis jedne linije teksta u grafičkom modu (prog05.py)

```
while True:
    ispis("HELLO!;10;10;G")
    ispis("HELLO!;11;17;G")
    ispis("HELLO!;12;24;1;B;G")
    ispis("HELLO!;13;31;1;B;G")
    ispis("BUF")
```

Slika 25. Primjer program za ispis više linija teksta u grafičkom modu (prog06.py)

Primjer programa koji ispisuje numeričke vrijednosti na određenoj poziciji putem TEXT funkcije.

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        for n in range(1,5):
            ispis(str(n)+". HELLO;10;"+str(n*8)+";B;G")
        ispis("BUF")
```

Slika 26. Primjer program za ispis više linija teksta u grafičkom modu (prog07.py)

3.6. Iscrtavanje linije - LIN

U ovom programu koristimo funkciju **LIN** za iscrtavanje linje na ekranu u grafičkom modu. Kod linije trebamo odrediti početnu (x1,y1) i završnu (x2,y2) točku na ekranu. Liniju možemo iscrtati u crnoj ili bijeloj boji. Ako već iscrtanu liniju crne boje želimo izbrisati trebamo iscrtati na istu poziciju liniju u bijeloj boji.

Isprobajte!

ispis("LIN;x1;y1;x2;y2;c")

LIN = naziv funkcije
x1 = 0-83 grafički mod
y1 = 0-47 grafički mod
x2 = 0-83 grafički mod
y2 = 0-47 grafički mod
c = Boja (B-crna, W-bijela)

Slika 27. Iscrtavanje linije

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        for n in range(10,40,2):
            ispis("LIN;10;"+str(n)+";60;"+str(n)+";B")
```

Slika 28. Iscrtavanje više linija (prog08.py)

```
import random - na početku
---

while True:
    if button_a.was_pressed():
        ispis("CLS")
        for n in range(0,30):
            x2 = random.randint(0,83)
            y2 = random.randint(0,47)
            ispis("LIN;0;0;"+str(x2)+";"+str(y2)+";B")
```

Slika 29. Iscrtavanje više linija uz uporabu funkcije **RANDOM** (prog09.py)

3.7. Iscrtavanje kružnice ili kruga - CIR

Kružnica ili krug nisu pravilnog oblika radi rezolucije ekrana.

U ovim primjerima koristimo funkciju **CIR** za iscrtavanje kružnice, u grafičkom modu. Kod kružnice trebamo odrediti poziciju središta (x,y) kružnice na ekranu i radijus. Kružnicu možemo iscrtati u crnoj ili bijeloj boji. Ako već iscrtanu kružnicu crne boje želimo izbrisati, trebamo iscrtati na istu poziciju kružnicu bijeloj boji. Za iscrtavanje kruga koristimo se **bojom ispune**.

Isprobajte!

ispis("CIR;x;y;r;c;f")

CIR = naziv funkcije
x = 0-83 grafički mod
y = 0-47 grafički mod
r = radius kružnice (pix)
c = Boja (B-crna, W-bijela)
f = Boja popunjavanja (B,W)

Slika 30. Funkcija za iscrtavanje kružnice ili kruga

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("CIR;40;20;10;B")
        ispis("CIR;40;20;20")
```

Slika 31. (prog10.py)

Iscrtavanje kružnice radijusa 10 piksela i radijusa 20 piksela na poziciji x = 40, y = 20 Slika 31.

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("CIR;40;20;10")
        ispis("CIR;40;20;15")
        ispis("CIR;40;20;20")
```

Slika 32. (prog11.py)

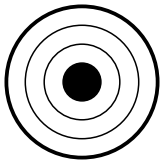
Iscrtavanje nekoliko kružnica različitog radijusa na istoj na poziciji x = 40, y = 20 (Slika 32.).

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("CIR;40;20;20;B;B")
        ispis("CIR;40;20;10;B;W")
```

Slika 33. (prog12.py)

Iscrtavanje bijelog kruga radijusa 10 unutar crnog kruga radijusa 20 piksela na poziciji x = 40, y = 20 (Slika 33.).

Za viježbu nacrtajate metu sa vanjskim debelim rubom i centrom u crnoj boji.



3.8. Iscrtavanje pravokutnika - REC

U ovim primjerima koristimo funkciju **REC** za iscrtavanje pravokutnuka u grafičkom modu. Kod pravokutnika trebamo odrediti poziciju lijevog gornjeg kuta (x,y), širinu (0-83) i visinu (0-47). Pravokutnik možemo iscrtati u crnoj ili bijeloj boji. Ako već iscrtani pravokutnik crne boje želimo izbrisati, trebamo iscrtati, na istu poziciju, pravokutnik bijele boje. Pravokutnik može biti iscrtan samo linijama ili popunjen bojom ispune (**filled with**).

Isprobajte!

ispis("REC;x;y;w;h;c;f")

REC = naziv funkcije
x = 0-83 grafički mod
y = 0-47 grafički mod
w = širina (pix)
h = visina (pix)
c = Boja (B-crna, W-bijela)
f = Boja popunjavanja (B,W)

Slika 34. Funkcija za iscrtavanje pravokutnika

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("REC;10;10;60;30;B")
```

Slika 35. (prog13.py)

Iscrtavanje pravokutnika na poziciji x = 10, y = 10 širine 60 i visine 30 piksela (Slika 35.).

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("REC;5;5;60;30;B")
        ispis("REC;10;10;50;20;B")
        ispis("REC;15;15;40;10;B")
```

Slika 36. (prog14.py)

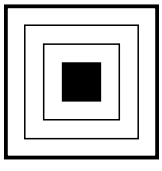
Iscrtavanje nekoliko pravokutnika različite pozicije i veličine (Slika 36.).

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("REC;5;5;60;30;B;B")
        ispis("REC;15;15;40;10;B;W")
```

Slika 37. (prog15.py)

Iscrtavanje bijelog pravokutnika širine 40 i visine 10, unutar crnog pravokutnika širine 60 i visine 30 piksela (Slika 37.).

Za viježbu nacrtajate pravokutnumetu sa vanjskim debelim rubom i centrom u crnoj boji.



3.9. Ispuna ekrana - FIL

Funkcija **FIL** ispunjava (boji) ekran bajtovima vrijednosti upisane u polje **color** (boja).
Ekran se ispunjava bajtovim koji su položen okomito kao na Slici 38.

ispis("FIL;n")

FIL = naziv funkcije
n = 0 - 255

Slika 38. Ispis bajtova memorije ekrana

Primjer ispune (bojanja) ekrana linijama u razmaku od jednog piksela. Izračun vrijednosti boje (bajta) možete vidjeti na Slici 39.

vrijednost = 1 + 4 + 16 + 64
vrijednost = 85

Slika 39. Izračun 'boje' (bajta)

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("FIL;85")
```

Slika 40. Primjer programa za **FIL** funkciju (prog16.py)

3.10. Prikaz crno/bijelo ili bijelo/crno - DIS

Ekran može postavljen u "**normalan**" mod (0) - bijeli ekran s crnim ispisom, ili u **inverzni** (obrnuti) mod (1) - crni ekran s bijelim ispisom. Standardno (default) je ekran postavljen u "**normalan**" mod (0). Promjenom moda putem funkcije **DIS** mijenja se kompletan sadržaj ekrana.
Isprobajte slijedeći primjer na Slici 41. Možete ga dopuniti tekstom.

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("DIS;0")
    if button_b.was_pressed():
        ispis("CLS")
        ispis("DIS;1")
```

Slika 41. Primjer programa za **DIS** funkciju (prog17.py)

3.11. Ispis piksela - PIX

ispis("PIX;x;y;c")

PIX = naziv funkcije
x = 0-83 grafički mod
y = 0-47 grafički mod
c = Boja (B-crna, W-bijela)

Slika 42.

Funkcija za ispis pixela na ekan u grafičkom modu.

```
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("PIX;20;20;B")
```

Slika 43. (prog18.py)

Ispis jednog pixela na ekan u grafičkom modu, na poziciji x=20, y=20.

```
while True:
    ispis("PIX;40;0;B")
    ispis("SBD;2")       # - funkcija SBD strana 16
```

Slika 44. (prog19.py)

```
import random

--

while True:
    x = random.randint(0,83)
    ispis("PIX;"+str(x)+";0;B")
    ispis("SBD;1")
```

Slika 45. (prog20.py)

Isprobajte programe na prethodnim slikama 44 i 45.

4. PROGRAMIRANJE - FUNKCIJE POMAKA - SCROLL

4.1. Pomak teksta prema GORE za jednu liniju - SCU

ispis("SCU;r")

SCU = naziv funkcije
r = rotacija (R = da , "" = ne)

Slika 46. Funkcija za pomak (SCROLL) teksta prema GORE

```
ispis("HELLO WORLD;1;5")
while True:
    ispis("SCU;R")
    sleep(500)
```

Slika 47. Primjer vertikalnog pomaka teksta (prog21.py)

Tekst ispisan na početku pomiče se za jednu liniju prema gore, svakih pola sekunde (Slika 47). Pomak teksta može se koristiti za sve veličine ispisa.

Funkcija ROTACIJA ima dva stanja da = "R" i ne = "". Probajte promijeniti stanje u NE (obrišite ;R).
Koja je razlika u pomaku teksta prema prethodnom stanju (R)?

4.2. Pomak teksta prema DOLJE za jednu liniju - SCD

ispis("SCD;r")

SCD = naziv funkcije
r = rotacija (R = da , "" = ne)

Slika 48. Funkcija za pomak (SCROLL) teksta prema DOLJE

```
ispis("HELLO WORLD;1;0")
while True:
    ispis("SCD;R")
    sleep(500)
```

Slika 49. Primjer vertikalnog pomaka teksta (prog22.py)

Tekst ispisan na početku pomiče se za jednu liniju prema dolje, svakih pola sekunde (Slika 49). Pomak teksta može se koristiti za sve veličine ispisa.

Funkcija ROTACIJA ima dva stanja da = "R" i ne = "". Probajte promijeniti stanje u NE (obrišite ;R).
Koja je razlika u pomaku teksta prema prethodnom stanju (R)?

4.3. Pomak ekrana (slike) prema GORE za jedan ili više piksela (liniju piksela) - SBU

ispis("SBU;n")

SBU = naziv funkcije
n = broj piksela

Slika 50. Funkcija za pomak (SCROLL) ekrana prema GORE

```
ispis("HELLO WORLD;1;5")
while True:
    ispis("SBU;1")
    sleep(200)
```

Slika 51. Primjer pomaka ekrana za jedan piksel prema GORE (prog23.py)

Tekst ispisan na početku pomiče se za jednu liniju piksela prema GORE, svakih 200 mili sekundi (Slika 48). Pomak ekrana (slike) se može povečati upisom večeg broja piksela.

4.4. Pomak ekrana (slike) prema DOLJE za jedan ili više piksela (liniju piksela) - SBD

ispis("SBD;n")

SBD = naziv funkcije
n = broj piksela

Slika 52. Funkcija za pomak (SCROLL) ekrana prema DOLJE

```
ispis("HELLO WORLD;1;0")
while True:
    ispis("SBD;1")
    sleep(200)
```

Slika 53. Primjer pomaka ekrana za jedan piksel prema GORE (prog24.py)

Tekst ispisan na početku pomiče se za jednu liniju pksela prema DOLJE, svakih 200 mili sekundi (Slika 53). Pomak ekrana (slike) se može povečati upisom večeg broja piksela.

4.5. Horizontalni pomak ekrana (slike) za jedan piksel - SCC

ispis("SCC;s;a;b;r")

SCC = naziv funkcije
s = smjer pomaka (R,L)
a = od linije teksta (0-5)
b = do linije teksta (0-5)
r = rotacija (R)

Slika 54.

Funkcija za horizontalni pomak ekrana (slike) za jedan piksel. Funkcija omogučava odabir smijera pomaka (LIJEVO - left ili DESNO - right), područja ekrana od linije (tekst) do linije ili cijelog ekrana (0-5) .

Kao i kod pomaka teksta može se uključiti opcija rotacije za kružni pomak teksta ili slike.

```
ispis("HELLO WORLD;1;1")
ispis("HELLO WORLD;1;2")
ispis("HELLO WORLD;1;3")
ispis("HELLO WORLD;1;4")
while True:
    ispis("SCC;R;2;3;R")
    sleep(200)
```

Slika 55.

Primjer pomaka slike u desnu stranu (Right) dvije srednje linije teksta s kružnim prikazom (Slika 55). Dodajte redove za ispis teksta u liniji 0 i 5, te promijenite u funkciji SCC vrijednost 2 u 0 i 3 u 5.

Koja se promjena desila?

Probajte promjeniti smijer pomaka.

4.5. Animacija funkcijama za pomak

```
from microbit import *
import music
#  2.04.2021

# -- COMMUNICATION - cannot be changed
pit= bytearray(1)
pit[0] = 1
spd = 4
def ispis(tekst):
    dd = len(tekst)
    if dd < 31:
        salji(tekst)
    else:
        salji(tekst[0:30] + "+")
        salji(tekst[30:dd])

def salji(tekst):
    global spd
    duz = len(tekst)
    buf = bytearray(duz)
    for n in range(0, duz):
        bb = ord(tekst[n : n + 1])
        buf[n] = bb
    try:
        i2c.write(0x11, buf)
        sleep(duz * spd)
    except OSError:
        print("er: send")
# -- COMMUNICATION --- end
def trazi():
    global pit
    try:
        pit = i2c.read(0x11,1)
    except OSError:
        print("er: seek")
# --
def rest():
    global pit
    ispis("RST")
    while True:
        trazi()
        if pit[0] == 5:
            break
        sleep(20)
# --
rest()
ispis("LIN;32;40;52;40")
ispis("LIN;32;41;52;41")
ispis("CIR;20;20;15;B;B")
ispis("CIR;62;20;15;B;B")
for n in range(0,3):
    ispis("CIR;20;20;"+str(n+11)+";W")
    ispis("CIR;62;20;"+str(n+11)+";W")
for n in range(0,4):
    ispis("SCC;L;0;5;R")
    sleep(10)
while True:
    for n in range(0,4):
        ispis("SCC;R;0;5;R")
        sleep(10)
    for n in range(0,4):
        ispis("SCC;L;0;5;R")
        sleep(10)
```

(prog26.py)

Jednostavna animacija putem različitih funkcija.

Isprobajte!

5. PROGRAMIRANJE - GAME FUNKCIJE

5.1. Kontrola LED svjetla - LED

Sučelje ima ugrađena dva LED svjetla, jedno **CRVENO** i jedno **ZELENO**. CRVENO se nalazi ispod ekrana na lijevoj strani, a ZELENO na desnoj strani.

ispis("LED;c;t")

LED = naziv funkcije
c = boja (R - crvena, G - zelena)
t = vrijeme milisekundi

Slika 56.

```
while True:
    ispis("LED;R;100")
    sleep(100)
    ispis("LED;G;100")
    sleep(100)
```

Slika 57.

Primjer programa koji naizmjenično pali CRVENO i ZELENO led svjetlo (Slika 57). (prog27.py)

5.2. Kreiranje BITMAPE (sprajta) - objekata - BIT

Kreiranje grafičkih objekata (BIT-mapa ili sprajtova) izvodi se u grafičkom modu (bufferu - memoriji) radi bržeg ispisa na ekranu i izbjegavanja određenih loših efekata (titranja). Zato se prvo objekt (jedan ili više objekata) spremaju u po močnu memoriju (buffer), a na kraju se memorija putem funkcije BUF prikazuje na ekranu.

Nakon kreiranja objekta (BITMAP), u ovom primjeru **1**, potrebno je odrediti poziciju na kojoj će se objekt iscrtati i kojom BOJOM (SPR).
Kratki primjer s osnovnim funkcijama prikazan je na Slici 59.

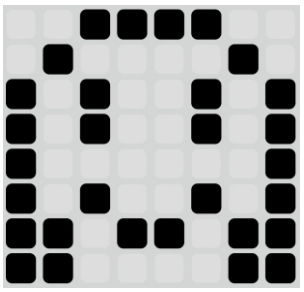
Kreirati bitmapu prema možete u Excel programu. Excel primjer sa predloškom prikazan je na slici 58.
Kod prijenosa iz Excel datoteke trebate dodati broj botmape (CRNO - 1).

SVE BITMAPE MOŽETE KORISTITI PO POTREBI, AKO NE KORISTITE FUNKCIJE UZ KOJE SU POVEZANE.

ispis("**BIT;n;b;b;b;b;b;b;b;b**")

BIT = naziv funkcije
n = broj bitmape (1,2,3,4,5,8,9)
b = byte (vrijednost bajta)

ODREĐENE BITMAPE: **2** - BOD + **3** - ŽIVOT - **4** - VRATA **8** - ANIMA **9** - IGRAČ
SLOBODNE BITMAPE: **1** i **5**



	128	64	32	16	8	4	2	1	
128			1	1	1	1			60
64		1					1		66
32	1		1			1		1	165
16	1		1			1		1	165
8	1							1	129
4	1		1			1		1	165
2	1	1		1	1		1	1	219
1	1	1					1	1	195
	60;66;165;165;129;165;219;195								

Slika 58.

bez broja bitmape

```
ispis("BIT;1;60;66;165;165;129;165;219;195")
while True:
    if button_a.was_pressed():
        ispis("CLS")
        ispis("SPR;1;40;20;B")   # -- ispis bitmape u memoriji
        ispis("BUF")   # -- ispis memorije na ekran
```

Slika 59. (prog28.py)

Boja ispisa nam omogučava da objekt ispišemo CRNOM bojom, a obrišemo BIJELOM bojom.

```
ispis("BIT;1;60;66;165;165;129;165;219;195")
ispis("CLS")
while True:
    for n in range(30,50):
        ispis("SPR;1;"+str(n)+";20;B")
        ispis("BUF")
        ispis("SPR;1;"+str(n)+";20;W")
    for n in range(50,30,-1):
        ispis("SPR;1;"+str(n)+";20;B")
        ispis("BUF")
        ispis("SPR;1;"+str(n)+";20;W")
```

Slika 60. (prog29.py)

Izmijenite prethodni program prema Slici 60.

5.5. Kontrole pomaka - horizontalne i vertikalne - BUT

Za upravljanje objektom igrača u svim smjerovima potrebno je dodati i kontrolu dodirnih senzora koji se nalaze na dolnjoj strani micro:bita. Analogno očitanje dodirnih senzora nije jednako sa priključenim USB kablom na micro:bit i kad nije priključen. Primjer na Slici 61 prikazuje vrijednosti sa priključenim USB kablom. Vrijedonst bez priključenog USB kabla na micro:bit je **< 100**.

ispis("BUT;s;p;n")

BUT = naziv funkcije
s = smjer + ili -
p = X = horizontalno Y = vertikalno
n = broj piksela pomaka

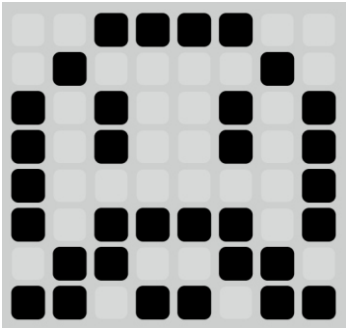
```
while True:
    p1 = pin1.read_analog() # touch sensor
    p2 = pin2.read_analog() # touch sensor
    if button_a.is_pressed():
        ispis("BUT;-;X;1")
    if button_b.is_pressed():
        ispis("BUT;+;X;1")
    if p1 > 240 or p1 < 230:
        ispis("BUT;-;Y;1")
    if p2 > 240 or p2 < 230:
        ispis("BUT;+;Y;1")
```

Slika 61.

Na slici 61 mogu se vidjeti primjeri za kontrolu horizontalnog i vertikalnog pomaka igrača putem dodirnih senzora ili tipkala.

5.6. Objekat igrača (bitmap 9)

Ako želimo da BITMAPA bude objekt igrača u izborniku treba odabrati **9 (Player)**. Nakon kreiranja objekta potrebno je pokrenuti funkciju za njegov prikaz na ekranu. Za pozicioniranje se koristi tekst mod rezolucija (14 x 6). U slijedećem primjeru, objekt igrača se iscrtava na poziciji x=5, y=2.



(9) Igrač

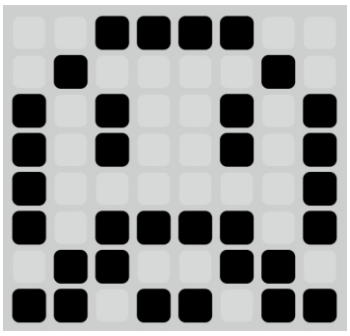
```
ispis("BIT;9;60;66;165;165;129;189;102;219") # player
ispis("POZ;5;2")
while True:
    sleep(5000)
```

Slika 62. (prog30.py)

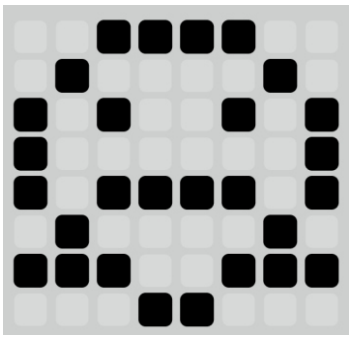
Funkcija PLAYER ujedno prikazuje stanje pomočne memorije (BUFFER) na ekranu, pa nije potrebno pozivanje funkcije **BUF**.

5.7. Animacija objekta igrača (bitmap 8)

Za animaciju objekta igrača potrebna je još jedna bitmapa. Dopunite program prema Slici 63. Kretanjem objekta igrača na ekranu se naizmjenično iscrtavaju bitmape **igrača (9)** i **animacijske bitmape igrača (8)**.



(9) Igrač



(8) Animacijska bitmapa igrača

```
ispis("BIT;9;60;66;165;165;129;189;102;219") # player
ispis("BIT;8;60;66;165;129;189;66;231;24") # player anima map
ispis("POZ;5;2")
while True:
    p1 = pin1.read_analog() # touch sensor
    p2 = pin2.read_analog() # touch sensor
    if button_a.is_pressed():
        ispis("BUT;-;X;1")
    if button_b.is_pressed():
        ispis("BUT;+;X;1")
    if p1 > 240 or p1 < 230:
        ispis("BUT;-;Y;1")
    if p2 > 240 or p2 < 230:
        ispis("BUT;+;Y;1")
```

Slika 63. (prog31.py)

5.8. Kontrola brzine animacije - ANI

Animacija objekta igrača može biti brža ili sporija. Za kontrolu brzine koristimo funkciju **ANI (animation speed)**. Ako želimo da animacija (izmjena bitmapa) bude sporija potrebno je upisati veću vrijednost. **Animacija se izvodi samo dok je objekt igrača u pokretu**.

ispis("ANI;n") ANI = naziv funkcije
n = brzina (manji broj veća brzina)

```
ispis("BIT;9;60;66;165;165;129;189;102;219") # player
ispis("BIT;8;60;66;165;129;189;66;231;24") # player anima map
ispis("POZ;5;2")
ispis("ANI;1")   #  manji broj veća brzina
while True:
    p1 = pin1.read_analog() # touch sensor
    p2 = pin2.read_analog() # touch sensor
    if button_a.is_pressed():
        ispis("BUT;-;X;1")
    if button_b.is_pressed():
        ispis("BUT;+;X;1")
    if p1 > 240 or p1 < 230:
        ispis("BUT;-;Y;1")
    if p2 > 240 or p2 < 230:
        ispis("BUT;+;Y;1")
```

Slika 64. (prog32.py)

Dopunite prethodni program funkcijom za kontrolu brzine animacije prema Slici 64.
Probajte sa različitim vrijednostima za brzinu.

5.8. START game (2.2.5 - str. 6)

Imamo osnovnu konstrukciju igre sa kontrolom kretanja objekta igrača. Na početku trebamo staviti poruku koja će se ispisati nakon pokretanja programa. Funkciju za prikaz standardne poruke **begin()** umetnite na početku programa, a nakon modula za komunikaciju. Nakon ispisa potrebno je pokrenuti funkciju **sleep**, tako da se poruka stigne pročitati, a nakon toga obrisati ekran funkcijom **CLS**. Ako ne pokrenemo funkciju za brisanje ekrana, objekt igrača će se iscrtati preko startnog teksta.

Ovaj modul ispisuje poruku na početku rada programa i može se izmjeniti prema vašim potrebama.

```
def begin():
    ispis("CLS")
    ispis("START;2;1;2")
    ispis("G A M E;3;3")
    sleep(1000)
    ispis("CLS")  # brisanje ekrana prije nastavka programa
```

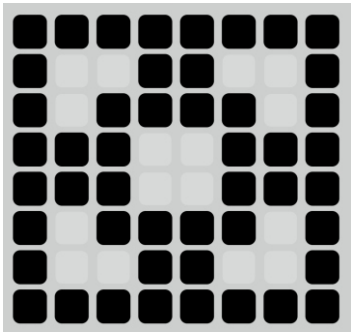
Slika 65. (prog33.py)

5.9. COLLISION funkcija - KOL

Dopunite program novim objektom (**1**). Iscrtajte ga na ekran (**SPR**) nekoliko puta, na različite pozicije prema Slici 67 ili proizvoljno. Dopunite program funkcijom **KOL**. Isprobajte koja je razlika sa uključenom funkcijom **KOL** (**1**) i isključenom (**0**).

Šta se dešava sa objektima prilikom pomicanja objekta igrača prema njima u jednom i drugom slučaju?

ispis("KOL;s") KOL = naziv funkcije
s = 0 - ne, 1 - da



bitmapa 1

```
begin()
ispis("BIT;1;255;153;189;231;231;189;153;255")
ispis("BIT;9;60;66;165;165;129;189;102;219") # player
ispis("BIT;8;60;66;165;129;189;66;231;24") # player anima map
ispis("SPR;1;28;28;B")
ispis("SPR;1;36;28;B")
ispis("SPR;1;44;28;B")
ispis("BUF")
ispis("POZ;5;2")
ispis("ANI;1")
ispis("KOL;1")
while True:
    p1 = pin1.read_analog() # touch sensor
    p2 = pin2.read_analog() # touch sensor
    if button_a.is_pressed():
        ispis("BUT;-;X;1")
    if button_b.is_pressed():
        ispis("BUT;+;X;1")
    if p1 > 240 or p1 < 230:
        ispis("BUT;-;Y;1")
    if p2 > 240 or p2 < 230:
        ispis("BUT;+;Y;1")
```

Slika 66. (prog34.py)

5.10. Gravitacija - GRV

Da bi kretanje objekta igrača bilo što prirodnije, te da bi mogao skakati i padati potrebno je u igru uključiti funkciju gravitacije - GRV (Slika 67 crveno).

ispis("GRV;s")

GRV = naziv funkcije
s = 0 - ne, 1 - da

```
from microbit import *
import music
#  2.04.2021

# -- COMMUNICATION - cannot be changed
pit= bytearray(1)
pit[0] = 1
spd = 8
def ispis(tekst):
    dd = len(tekst)
    if dd < 31:
        salji(tekst)
    else:
        salji(tekst[0:30] + "+")
        salji(tekst[30:dd])

def salji(tekst):
    global spd
    duz = len(tekst)
    buf = bytearray(duz)
    for n in range(0, duz):
        bb = ord(tekst[n : n + 1])
        buf[n] = bb
    try:
        i2c.write(0x11, buf)
        sleep(duz * spd)
    except OSError:
        print("er: send")
# -- COMMUNICATION --- end
def trazi():
    global pit
    try:
        pit = i2c.read(0x11,1)
    except OSError:
        print("er: seek")
# --
def rest():
    global pit
    ispis("RST")
    while True:
        trazi()
        if pit[0] == 5:
            break
        sleep(20)
# --
def begin():
    ispis("CLS")
    ispis("START;2;1;2")
    ispis("G A M E;3;3")
    sleep(1000)
    ispis("CLS")
# --
rest()
begin()
ispis("BIT;1;255;153;189;231;231;189;153;255")
ispis("BIT;9;60;66;165;165;129;189;102;219") # pl
ispis("BIT;8;60;66;165;129;189;66;231;24") # ani
ispis("SPR;1;28;28;B")
ispis("SPR;1;36;28;B")
ispis("SPR;1;44;28;B")
ispis("BUF")
ispis("POZ;5;2")
ispis("ANI;1")
ispis("KOL;1")
ispis("GRV;1")
while True:
    if button_a.is_pressed():
        ispis("BUT;-;X;1")
    if button_b.is_pressed():
        ispis("BUT;+;X;1")
```

Radi dodavanja funkcije GRV (gravitacija) za kontrolu objekta igrača koristimo samo horizontalne kontrole. Vertikalne smo maknuli iz programa.

Kompletan program je prikazan na Slici 67.

Koja je razlika u kretanju objekta igrača sa uključenom gravitacijom i isključenom?

Slika 67. kompletan program (prog35.py)

5.11. Kreiranje objekata (horizontalnih i vertikalnih) - OBJ - max. 20 objekata

Objekti koji su duži od jedne bitmape (8x8 piksela) mogu biti položeni horizontalno ili vertikalno.
Objekti se kreiraju višekratnim ponavljanjem iste bitmape. **Objekti putem kojih se zarađuju bodovi ili gube životi običajeno su dužine jedne bitmape, ako su duži, samo prva pozicija je aktivna za dobivanje bodova ili gubitak života.**

ispis("OBJ;e;b;x;y;n;s")

OBJ = naziv funkcije
e = ekran broj (1 - 5)
b = bitmapa
x = horizontalna pozicija (0-10)
y = vertikalna pozicija (0-5)
n = dužina (broj ponavljanja)
s = smjer ("" ili 0 = hor., 1 = ver.)

```
begin()
ispis("BIT;1;255;153;189;231;231;189;153;255")
ispis("BIT;9;60;66;165;165;129;189;102;219") # player
ispis("BIT;8;60;66;165;129;189;66;231;24") # player anima map
ispis("OBJ;1;1;0;3;6")
ispis("OBJ;1;1;8;0;5;1")
ispis("FX;1")     # -- prikaz ”ekrana” 1
ispis("POZ;5;2")
ispis("ANI;1")
ispis("KOL;1")
ispis("GRV;1")
while True:
    if button_a.is_pressed():
        ispis("BUT;-;X;1")
    if button_b.is_pressed():
        ispis("BUT;+;X;1")
```

Slika 68. (prog36.py)

Napravili smo izmjene u prethodnom programu prema primjeru na slici 68. Prvo je potrebno definirati objekte koji će se iscrtati na ekranu putem naredbe **OBJ**. Sve naredbe se grupiraju u «**ekrane**» koji se putem naredbe **FX** prikazuju na ekranu. U ovom primjeru definiramo «**ekran**» **1** s dva objekta. Oba objekta su složena od iste bitmape (1). Za pozicioniranje potrebno je odrediti x i y poziciju početne bitmape objekta. Dužina (broj ponavljanja) se određuje upisom vrijednosti u polje **dužina.** Maksimalna horizontalna dužina je 11 (84 / 8 = 10,5 bitmapa).

Za vertikalno isctavanje bitmape potrebno je promijeniti vrijednost polja «**hor/ver**» u **1**.

```
ispis("OBJ;1;1;0;0;10")
ispis("OBJ;1;1;0;5;10")
ispis("OBJ;1;1;0;1;4;1")
ispis("OBJ;1;1;9;1;4;1")
```

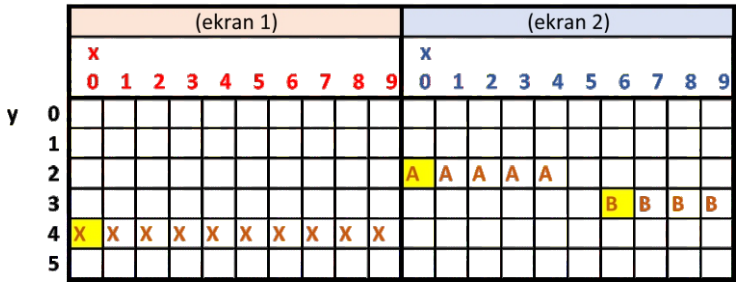
Slika 69. (prog37.py)

Zamijenite objekte definirane u prethodnom primjeru objektima prema Slici 69.

Ne zaboravite naredbu FX (1) koja obavezno dolazi iz definiranih objekata.

5.12. Kreiranje više od jednog «ekrana» - max. 5 «ekrana»

Ako želimo kreirati više različitih «**ekrana**» potrebno je za svaki «**ekran**» kreirati objekte. Slijedeći primjer je sa dva «**ekrana**» i tri objekta.



Slika 70.

Kada želite kreirati više «**ekrana**» dobro je napraviti skicu kao što prikazuje primjer na Slici 70.
Ekrane možemo skicirati putem tabele u Excelu ili raster papiru. Na taj način je puno lakše vizualizirati sve ekrane, naročito ako se koriste horizontalni i vertikalni objekti.

```
begin()
ispis("BIT;1;255;153;189;231;231;189;153;255")
ispis("BIT;9;60;66;165;165;129;189;102;219") # player
ispis("BIT;8;60;66;165;129;189;66;231;24") # player anima map
ispis("OBJ;1;1;0;4;10")
ispis("OBJ;2;1;0;2;5")
ispis("OBJ;2;1;6;3;4;1")
ispis("POZ;5;2")
ispis("ANI;1")
ispis("KOL;1")
ispis("GRV;1")
while True:
    if button_a.is_pressed():
        ispis("CLS")
        ispis("FX;1")
    if button_b.is_pressed():
        ispis("CLS")
        ispis("FX;2")
```

Slika 71. (prog38.py)

Primjer na Slici 71 koristi dva «**ekrana**» koji se mogu putem tipke **A** ili **B** naizmjenično prikazati.

5.13. Tekući «ekrani» - prikaz ekrana s horizontalnim pomakom (scroll) - ASD

Kod kreiranja platformske igre koristimo pokretne platfome koje se pomiču s jedne strane ekrana u drugu. Da bi pokrenuli platforme (objekte) koje smo kreirali u prethodnom programu (Slika 73) trebamo dodati aktiviranu naredbu **ASD (1)** (Slika 72.). Objekti na «**ekranima**» se ispisuju na ekranu u kružnom redoslijedu brojeva «ekrana» (1,2,1,2,1,2,...)

ispis("ASD;s")

ASD = naziv funkcije
s = ("" ili 0) = ne, 1 = da

Slika 72.

Brzina horizontalnog pomaka objekata može se mijenjati naredbom **SPD** (Slika 73.). Manja vrijednost varijable znači veću brzinu pomaka. Maksimalna brzina je ograničena na **10**, a početna (default) je postavljena na 100 (0 = 10). Brzinu možemo povečati i pomakom za 2 piksela (default 1), ali pomak više neće biti tako 'fini' kao kod pomaka za 1 piksel. Ako program sadrži **puno objekata** za kontrolu, prilikom maksimalne brzine (10) se može desiti da prestane s radom. **U tom slučaju trebate smanjiti brzinu, jer program ne stigne obraditi sve operacije u prekratkom vremenu.**

ispis("SPD;s;p")

SPD = naziv funkcije
s = brzina max.preporučena brzina = **10** (veći broj = manja brzina)
p = broj piksela

Slika 73.

5.14. Pokretanje objekta igrača kod ekrana s pomakom - skok - JMP

Da bi igra mogla normalno funkcionirati moramo zamijeniti klasične kontrole kretanja **igrača** novim. **Ako pokušamo koristiti klasične kontrole kretanja igrača, program neće moći normalno raditi**.
Skok '**igrača**' direktno prema gore kontoliramo putem naredbe **JMP**, koja se najčešće koristi kod platformske igre gdje se igrač nalazi istoj poziciji na ekranu (horizontalno). Visina skoka se određuje u pikselima. Ako želimo, smjer skoka određujemo upisom + ili - varijable i kuta skoka. Da bi funkcija bila aktivna morali smo dodati još neke obavezne funkcije za poziciju i kontrolu kretanja objekta **igrača** (COLLISION, GRAVITY,PLAYER start position).

ispis("JMP;h;s;a")

JMP = naziv funkcije
h = visina skoka - piksela
s = smjer (- , +)
a = kut skoka (45%) 1 - 5

Slika 74.

```
begin()
ispis("BIT;1;255;153;189;231;231;189;153;255")
ispis("BIT;8;60;66;165;153;66;255;231;0") # player anima
ispis("BIT;9;0;0;60;66;165;153;66;255") # player
ispis("OBJ;1;1;0;4;10")
ispis("OBJ;2;1;0;2;5")
ispis("OBJ;2;1;6;3;3;1")
ispis("FX;1")
ispis("POZ;5;2")
ispis("ANI;5")
ispis("KOL;1")
ispis("GRV;1")
ispis("ASD;1")  # -- auto pomak
ispis("SPD;50;1")  # -- brzina 50, pomak za 1 piksel
spd = 4  # ---------- možete povečati brzinu komunikacije
while True:
    if button_a.is_pressed():
        ispis("JMP;15")
    if button_b.is_pressed():
        ispis("JMP;15;+;1")
    sleep(100)  # ---- usporavanje prečestog korištenja tipkala
```

Slika 75. (prog39.py)

Da bi vidjeli koje su razlike u brzini pomaka probajte program s različitim vrijednostima za brzinu. Možete proširiti program za još jedan «**ekran**» (3).

5.15. Kontrolne funkcije

5.15.1. **Status igre - GAME status - trazi() (2.2.3 - strana 4)**

Da bi program koji se pokreče u micro:bitu mogao izvršiti neke funkcije potrebno je očitati određene vrijednosti koje se koriste u igri. Status igre se koristi za izvođenje zvučnih efekata tijekom igre i da bi se znalo kada je kraj igre. Funkcija se poziva **SAMO JEDNOM** prije ispitivanja dobivenih vrijednosti, ili kod upita za X ili Y poziciju igrača.

```
def trazi():
    global pit
    try:
        pit = i2c.read(0x11,1) # AD BW display adresa
    except OSError:
        print("er: trazi")
```

Slika 76.

5.15.2. **Zvučni i svijetlosni efekti - zvuk() (2.2.4 - strana 5)**

Za izvođenje određenih zvučnih i svijetlosnih efekata povezanih uz određeni događaj u igri (bod, gubitak života, pad) koristimo funkciju **zvuk()**. Obavezno je prethodno pokrenuti funkciju **trazi()**.

```
def zvuk():
    global pit
    if pit[0] == 2:  # POINT
        music.pitch(1500, 50)
        ispis("LED;R;30")
    if pit[0] == 3:  # LIFE
        music.pitch(800, 100)
        ispis("LED;G;30")
    if pit[0] == 4:  # FALL
        for freq in range(900, 1200, 30):
            music.pitch(freq, 8)
```

Slika 77.

5.15.3. **Poruka za kraj igre - end() (2.2.5 - strana 5)**

Da bi program završio igru porukom, potrebno je uključiti funkciju **end()** za završnu poruku**.** Obavezno je prethodno pokrenuti funkciju **trazi().**

```
def end():
    global pit
    sleep(100)
    trazi()
    ispis("CLS")
    ispis("E N D;2;1;2")
    ispis("G A M E;3;3")
    ispis("Score:"+str(pit[0])+";3;4")
    while True:
        sleep(5000)
```

Slika 78.

5.15.4. **Bodovi - BOD**

Za praćenje broja osvojenih bodova u igri potrebno je pokrenuti funkciju **BOD** sa početnim brojem bodova. U igri u kojoj samo možemo osvojiti plus bodove običajeno je početna vrijednost 0. U igri sa mogućnosti dobitka i gubitka bodova, početna vrijednost je veča od nule. Da bi **igrač** dobio bodove potrebno je u igru uključiti **Points (+)** objekte.

ispis("BOD;n")

BOD = naziv funkcije
n = broj bodova na početku igre (0)

Slika 79.

5.15.5. **Životi - LIV**

Za gubitak života, u igri, potrebno je pokrenuti funkciju **LIV** koja postavlja početnu vrijednost broja života u igri. Gubitak života se događa prilikom pada (PAD) ili dodira sa objektom **Lives (-).**

ispis("LIV;n")

LIV = naziv funkcije
n = broj života na početku igre

Slika 80.

5.15.6. **Pad gubitak života - PAD**

Igra se sastoji od platformi kojima se **igrač** kreće. Prilikom pada objekt igrača može izgubiti život ili se samo ponovo pojaviti na početnoj poziciji. Uključivanjem funkcije **PAD** uključujemo gubitak života prilikom pada van ekrana.

ispis("PAD;b")

LIV = naziv funkcije
b = 0 - ne , 1 - da

Slika 81.

5.15.7. **Ograničeno trajanje igre - TIM**

Za definiranje vremena trajanja igre u sekundama potrebno je uključiti funkciju TIM. Funkciju koristimo u igrama u kojima je cilj skupiti što više bodova u jednakom vremenskom roku. U takvim igrama se koristi funkcija samo za bodove.

ispis("TIM;n")

TIM = naziv funkcije
n = vrijeme u sekundama

Slika 82.

5.15.8. **Negativni bodovi - BON**

Ako želite uključiti oduzimanje bodova potrebno je uključiti funkciju **BON**.
Da bi ova funkcija bila aktivna potrebno je aktivirati i funkciju **LIV**. Ova funkcija oduzima bodove gubitkom života.

ispis("BON")

Slika 83.

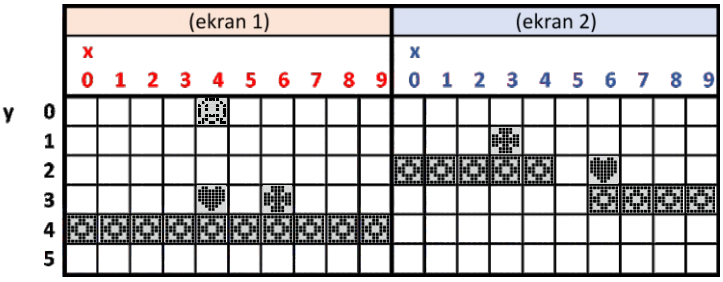
5.15.9. **Redoslijed prikaza «ekrana» slučajnim odabirom - RND**

Ako igra ima **VIŠE OD DVA «ekrana»**, da bi izbjegli stalno ponavljanje istog redosljeda možemo uključito ovu funkciju. Funkcija slučajnim odabirom (**random**) kreira redosljed prikazivanja «**ekrana**».

ispis("RND")

Slika 84.

5.16. Kompletna platformska igra



Slika 85.

Da bi igra bila kompletna dodali smo u svaki «**ekran**» objekte za dobivanje bodova ( ) i gubitak života ( ) prema pozicijam na skici (Slika 85.), te objekt za animaciju **igrača**.

```
from microbit import *
import music
#  2.04.2021

# -- COMMUNICATION - cannot be changed
pit= bytearray(1)
pit[0] = 1
spd = 8
def ispis(tekst):
    dd = len(tekst)
    if dd < 31:
        salji(tekst)
    else:
        salji(tekst[0:30] + "+")
        salji(tekst[30:dd])

def salji(tekst):
    global spd
    duz = len(tekst)
    buf = bytearray(duz)
    for n in range(0, duz):
        bb = ord(tekst[n : n + 1])
        buf[n] = bb
    try:
        i2c.write(0x11, buf)
        sleep(duz * spd)
    except OSError:
        print("er: send")
# -- COMMUNICATION --- end
def trazi():
    global pit
    try:
        pit = i2c.read(0x11,1)
    except OSError:
        print("er: seek")
# --
def rest():
    global pit
    ispis("RST")
    while True:
        trazi()
        if pit[0] == 5:
            break
        sleep(20)
# --
def begin():
    ispis("CLS")
    ispis("START;2;1;2")
    ispis("G A M E;3;3")
    sleep(1000)
    ispis("CLS")
# --
def end():
    global pit
    sleep(100)
    trazi()
    ispis("CLS")
    ispis("E N D;2;1;2")
    ispis("G A M E;3;3")
    ispis("Score:"+str(pit[0])+";3;4")
    while True:
        sleep(5000)
# --
def zvuk():
    global pit
    if pit[0] == 2:  # POINT
        music.pitch(1500, 50)
        ispis("LED;G;30")
    if pit[0] == 3:  # LIFE
        music.pitch(800, 100)
        ispis("LED;R;30")
    if pit[0] == 4:  # FALL
        for freq in range(900, 1200, 30):
            music.pitch(freq, 8)
# --
```

(nastavak programa na sljedećoj stranici)

```
rest()
begin()
ispis("BIT;1;255;153;189;231;231;189;153;255")
ispis("BIT;2;102;255;255;255;255;126;60;24")
ispis("BIT;3;60;60;219;255;255;219;60;60")
ispis("BIT;8;60;66;165;153;66;255;231;0") # player anima
ispis("BIT;9;0;0;60;66;165;153;66;255") # player
ispis("OBJ;1;1;0;4;10")
ispis("OBJ;1;2;4;3;1")
ispis("OBJ;1;3;6;3;1")
ispis("OBJ;2;1;0;2;5")
ispis("OBJ;2;1;6;3;4")
ispis("OBJ;2;2;6;2;1")
ispis("OBJ;2;3;3;1;1")
ispis("FX;1")
ispis("POZ;4;0")
ispis("ANI;3")
ispis("KOL;1")
ispis("GRV;1")
ispis("ASD;1")
ispis("SPD;50;1")
ispis("PAD;1")
ispis("BOD;0")
ispis("LIV;5")
spd = 4
while True:
    if button_a.is_pressed():
        ispis("JMP;15;-;2")
    if button_b.is_pressed():
        ispis("JMP;15;+;2")
    sleep(100)
    trazi()
    if pit[0] != 9:
        zvuk()
    else:
        sleep(300)  # -- vrijeme za očitavanje end statusa
        end()
```

Slika 86. (prog40.py)

5.16.1. **Brisanje podataka iz memorije - DEL**

Tijekom izrade programa i mjenjanjem definicija objekata može se desiti, da neki objekti koje ste izbrisali iz programa, ostanu zapisani u memoriji sučelja. U tom slučaju bi se mogli na ekranu pojaviti '**fantomski**' objekti koje ste obrisali iz programa. Da bi se to izbjeglo na početku programa možete svaki puta gasiti i paliti sučelje AD ili jednostavno dodati funkciju **DEL** na početku programa.

Ova funkcija se može maknuti iz programa, nakon što je program završen.

```
ispis("DEL")
```

Slika 87.

5.16.2. **Automatska kontrola nivoa (levela) igre** - **LVL**

Da bi igra bila zahtjevnija možemo dodati više težinskih novoa. Viši nivo ima veću brzinu izvođenja igre i samim time ga je teže završiti. Uz funkciju za automatsku kontrolu možemo odrediti vrijednosti koje određuju nivoe igre. Na početku upisujemo vrijednost koja definira najveću brzinu igre (zadnji nivo). Nakon toga početnu brzinu kojom započinjemo igru. Za koliko se povećava brzina igre prelaskom na viši nivo. Zadnja vrijednost određuje koliko je potrebno bodova osvojiti da bi se prešlo na viši nivo.

Ova funkcija možda neće podržavati sve oblike igara.

Iz prethodnog programa obrišite **SPD** funkciju i dodajte **LVL** s vrijednostima iz primjera na slici 88.

```
ispis("LVL;s;f;d;b")
```

LVL = naziv funkcije
s = najveća brzina (zadnji nivo)
f = početna brzina (prvi nivo)
d = povečanje brzine za vrijednost d
b = potreban broj bodova za novi nivo

Slika 88. (prog41.py)

5.16.3. **Brzina razmjene podataka (micro:bit - AD display) - spd (vrijednost)**

Na početku rada programa, kad se šalje puno podataka za definiranje različitih funkcija i objekata potrebno je **spd** postaviti na **8** (početna vrijednost) ili više. Na taj način program AD sučelja ima dovoljno vremena da obradi sve podatke. U koliko je vrijeme prekratko (brzina prevelika), program neće stići obraditi sve podatke koje mu micro:bit pošalje i nedostajati će neki objekti ili neće raditi neke funkcije, ili će program prestati sa radom.

Ako želite da se radnje (nakon dijela programa koji šalje postavke za objekte i funkcije) odvijaju brže, te da igra bude što brža, možete **spd** postaviti na **4** (najmanja preporučena vrijednost, za najveću brzinu).
Ne preporuča se vrijednost manja od **4**.

Program AD sučelja dozvoljava da probate i sa manjim vrijednostima.

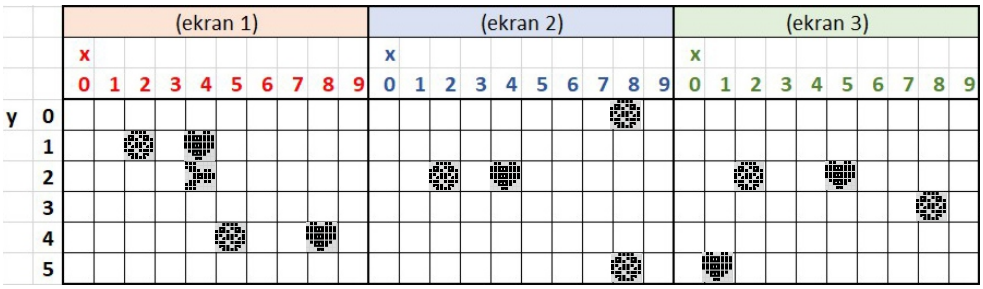
```
spd = 8
```

Slika 89.

6. PRIMJER PROGRAMA

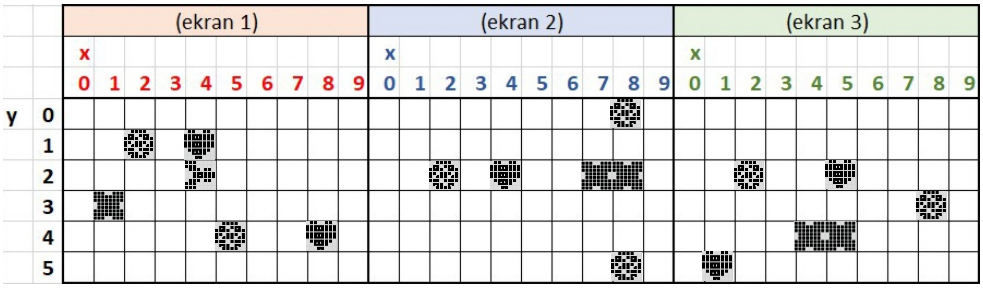
6.1. METEORI

Program uključuje neke od funkcija koje su opisane na stranici 29. Igra ima tri «**ekrana**» koji se prikazuju slučajnim redosljadom uz pomoć funkcije **RND** (5.15.9.) i ograničena je na **30 sekundi** funkcijom **TIM** (5.15.7.). Skica rasporeda objekata je prikazana na doljnjoj slici (Slika 90.).

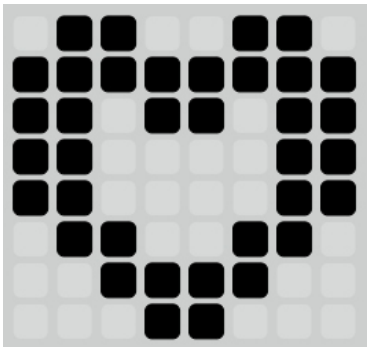


Slika 90.

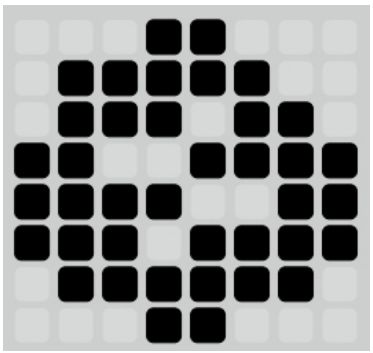
Ako želite otežati osvajanje bodova možete još dodati objekte koji će samo otežavati kretanje **igrača**, kao što je prikazano na Slici 91.



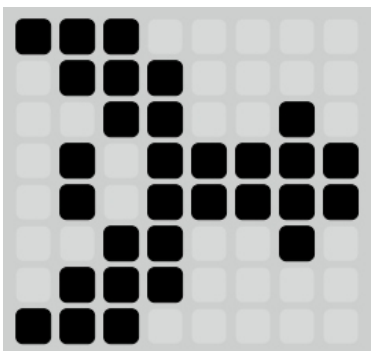
Slika 91.



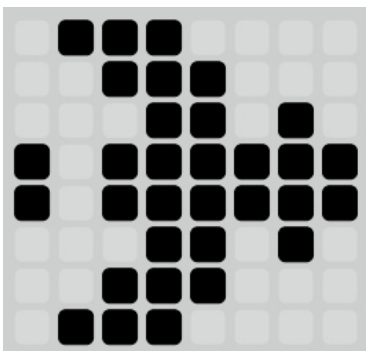
bitmap 2



bitmap 3



bitmap 9



bitmap 8

```
from microbit import *
import music
#  2.04.2021

# -- COMMUNICATION - cannot be changed
pit= bytearray(1)
pit[0] = 1
spd = 8
def ispis(tekst):
    dd = len(tekst)
    if dd < 31:
        salji(tekst)
    else:
        salji(tekst[0:30] + "+")
        salji(tekst[30:dd])

def salji(tekst):
    global spd
    duz = len(tekst)
    buf = bytearray(duz)
    for n in range(0, duz):
        bb = ord(tekst[n : n + 1])
        buf[n] = bb
    try:
        i2c.write(0x11, buf)
        sleep(duz * spd)
    except OSError:
        print("er: send")
# -- COMMUNICATION --- end
def trazi():
    global pit
    try:
        pit = i2c.read(0x11,1)
    except OSError:
        print("er: seek")
# --
def rest():
    global pit
    ispis("RST")
    while True:
        trazi()
        if pit[0] == 5:
            break
        sleep(20)
# --
def begin():
    ispis("CLS")
    ispis("START;2;1;2")
    ispis("G A M E;3;3")
    sleep(1000)
    ispis("CLS")
# --
def end():
    global pit
    sleep(100)
    trazi()
    ispis("CLS")
    ispis("E N D;2;1;2")
    ispis("G A M E;3;3")
    ispis("Score:"+str(pit[0])+";3;4")
    while True:
        sleep(5000)
# --
def zvuk():
    global pit
    if pit[0] == 2:  # POINT
        music.pitch(1500, 50)
        ispis("LED;G;30")
    if pit[0] == 3:  # LIFE
        music.pitch(800, 100)
        ispis("LED;R;30")
    if pit[0] == 4:  # FALL
        for freq in range(900, 1200, 30):
            music.pitch(freq, 8)
# --
rest()
begin()
ispis("BIT;2;102;255;219;195;195;102;60;24")
ispis("BIT;3;24;124;118;207;243;239;126;24")
ispis("BIT;8;224;112;50;95;95;50;112;224") # anim
ispis("BIT;9;112;56;26;191;191;26;56;112") # play.

ispis("OBJ;1;3;2;1;1") -- ekran 1
ispis("OBJ;1;3;5;4;1")
ispis("OBJ;1;2;4;1;1")
ispis("OBJ;1;2;8;4;1")

ispis("OBJ;2;3;2;2;1") -- ekran 2
ispis("OBJ;2;3;8;5;1")
ispis("OBJ;2;2;4;2;1")

ispis("OBJ;3;3;2;2;1") -- ekran 3
ispis("OBJ;3;3;8;3;1")
ispis("OBJ;3;2;1;5;1")
ispis("OBJ;3;2;5;2;1")

ispis("FX;1")
ispis("POZ;4;2")
ispis("ANI;3") -- animacija igrača
ispis("KOL;1")
# --------->              ispis("GRV;1") - ne koristimo
ispis("ASD;1")
ispis("SPD;30;1") -- brzina pomaka
# --------->              ispis("PAD;1") - ne koristimo
ispis("BOD;0")
ispis("LIV;5")
ispis("TIM;30")   -- vrijeme igre 30 sekundi
ispis("RND")      -- slučajni redoslijed
spd = 4           -- maksimalna brzina komunikacije
while True:
    if button_a.is_pressed():
        ispis("BUT;+;Y;2")
    if button_b.is_pressed():
        ispis("BUT;-;Y;2")
    trazi()
    if pit[0] != 9:
        zvuk()
    else:
        sleep(300)
        end()
```

Slika 92. (prog42.py)

7. PRIMJER PROGRAMA

7.1. OSTALE FUNKCIJE

7.1.1. **Preuzimanje pozicije igrača - GET (X ili Y)**

Putem ove funkcije možete preuzeti vrijednost za horizontalni (**X**) ili vertikalni (**Y**) položaj **igrača** na ekranu. Vrijednost prikazuje grafičku poziciju igrača za x (0-83) ili y (0-47) vrijednost. Ovu funkciju možete koristiti u bilo kojoj kombinaciji pri izradi programa. **U slijedećem programu će biti prikazan način upotrebe ove funkcije.**

ispis("GET;n")

GET = naziv funkcije
n = varijabla (X, Y)

Slika 93.

7.2. Primjer programa

```
from microbit import *
import music
#  2.04.2021

# -- COMMUNICATION - cannot be changed
pit= bytearray(1)
pit[0] = 1
spd = 8
def ispis(tekst):
    dd = len(tekst)
    if dd < 31:
        salji(tekst)
    else:
        salji(tekst[0:30] + "+")
        salji(tekst[30:dd])

def salji(tekst):
    global spd
    duz = len(tekst)
    buf = bytearray(duz)
    for n in range(0, duz):
        bb = ord(tekst[n : n + 1])
        buf[n] = bb
    try:
        i2c.write(0x11, buf)
        sleep(duz * spd)
    except OSError:
        print("er: send")
# -- COMMUNICATION --- end
def trazi():
    global pit
    try:
        pit = i2c.read(0x11,1)
    except OSError:
        print("er: seek")
# --
def rest():
    global pit
    ispis("RST")
    while True:
        trazi()
        if pit[0] == 5:
            break
        sleep(20)
# --
def begin():
    ispis("CLS")
    ispis("START;2;1;2")
    ispis("G A M E;3;3")
    sleep(1000)
    ispis("CLS")
# --
def end():
    global pit
    sleep(100)
    trazi()
    ispis("CLS")
    ispis("E N D;2;1;2")
    ispis("G A M E;3;3")
    ispis("Score:"+str(pit[0])+";3;4")
    while True:
        sleep(5000)
# --
def zvuk():
    global pit
    if pit[0] == 2:  # POINT
        music.pitch(1500, 50)
        ispis("LED;G;30")
    if pit[0] == 3:  # LIFE
        music.pitch(800, 100)
        ispis("LED;R;30")
    if pit[0] == 4:  # FALL
        for freq in range(900, 1200, 30):
            music.pitch(freq, 8)
# --
```

(nastavak programa na sljedećoj stranici)

```
def dis():                    -- funkcija za očitanje x i y pozicije igrača
    global pit
    ispis("   ;0;0;G")        -- brisanje vrijednosti
    ispis("   ;65;0;G")
    ispis("GET;X")            -- upit za X vrijednost
    trazi()                   -- očitaj X vrijednost
    ispis(str(pit[0])+";0;0;G")  -- ispis X vrijednosti na ekran
    ispis("GET;Y")            -- upit za Y vrijednost
    trazi()                   -- očitaj Y vrijednost
    ispis(str(pit[0])+";65;0;G") -- ispis Y vrijednosti na ekran

rest()
begin()
ispis("BIT;1;255;153;189;231;231;189;153;255")
ispis("BIT;2;102;255;255;255;255;126;60;24")
ispis("BIT;3;60;60;219;255;255;219;60;60")
ispis("BIT;9;0;0;60;66;165;153;66;255") # player
ispis("OBJ;1;1;0;4;11")
ispis("OBJ;1;3;9;3;1")
ispis("FX;1")
ispis("POZ;3;0")
ispis("KOL;1")
while True:
    p1 = pin1.read_analog() # touch sensor
    p2 = pin2.read_analog() # touch sensor
    if button_a.is_pressed():
        ispis("BUT;-;X;1")
        dis()                 -- pozivanje funkcije dis()
    if button_b.is_pressed():
        ispis("BUT;+;X;1")
        dis()
    if p1 > 240 or p1 < 230:
        ispis("BUT;-;Y;1")
        dis()
    if p2 > 240 or p2 < 230:
        ispis("BUT;+;Y;1")
        dis()
```

Slika 94. (prog43.py)

U ovom primjeru koristimo funkcije za preuzimanje pozicije **igrača** samo radi prikaza na ekranu. Iste funkcije možete koristiti za kontrolu **igrača** ili ograničavanje njegovog kretanja po ekranu.

```
def dis():
    global pit
    global x
    global y
  #  ispis("   ;0;0;G")
  #  ispis("   ;65;0;G")
    ispis("GET;X")
    trazi()
    x = pit[0]
  #  ispis(str(x)+";0;0;G")
    ispis("GET;Y")
    trazi()
    y = pit[0]
  #  ispis(str(y)+";65;0;G")
```

(da bi program radio brže možete isključiti prikaz vrijednosti na ekranu, prema gornjem primjeru)

```
x = 0
y = 0
while True:
    p1 = pin1.read_analog() # touch sensor
    p2 = pin2.read_analog() # touch sensor
    if button_a.is_pressed():
        if x > 20:
            ispis("BUT;-;X;2")
        dis()
    if button_b.is_pressed():
        if x < 30:
            ispis("BUT;+;X;2")
        dis()
    if p1 > 240 or p1 < 230:
        if y > 8:
            ispis("BUT;-;Y;2")
        dis()
    if p2 > 240 or p2 < 230:
        if y < 16:
            ispis("BUT;+;Y;2")
        dis()
```

Slika 95. (prog44.py)

Primjer ograničenja horizontalnog i vertikalnog kretanja igrača (Slika 95).
U prethodnom programu (Slika 94) napravite izmjenu prema primjeru na slici 95.

8. ZAKLJUČAK

Željeli smo napraviti sučelje s ekranom koje će omogućiti prikaz podataka ili izradu jednostavnih igrica. Pri izradi igre, koriste se neke funkcije za definiranje rada igre, a koje imamo i u pravim računalnim igrama (gravitacija). Da bi omogućili maksimalnu kreativnost **većina funkcija nema limitirane vrijednosti**, a to znači da će se dešavati greške kao što je prestanak rada programa ili ispis krivih podataka na ekranu.

Želimo vam ugodan rad.