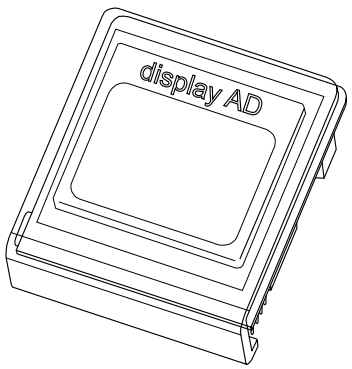
Arduino - C++ programiranje

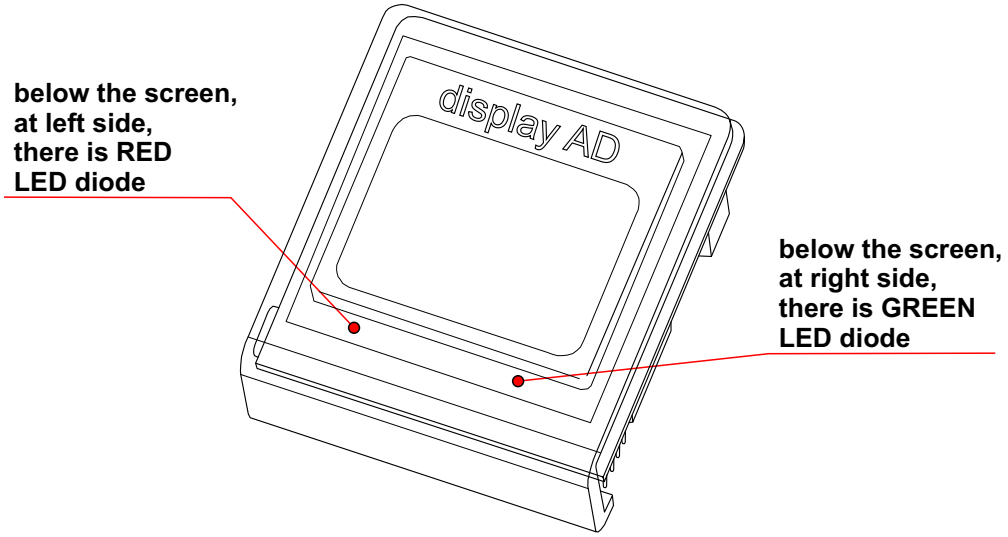
display AD



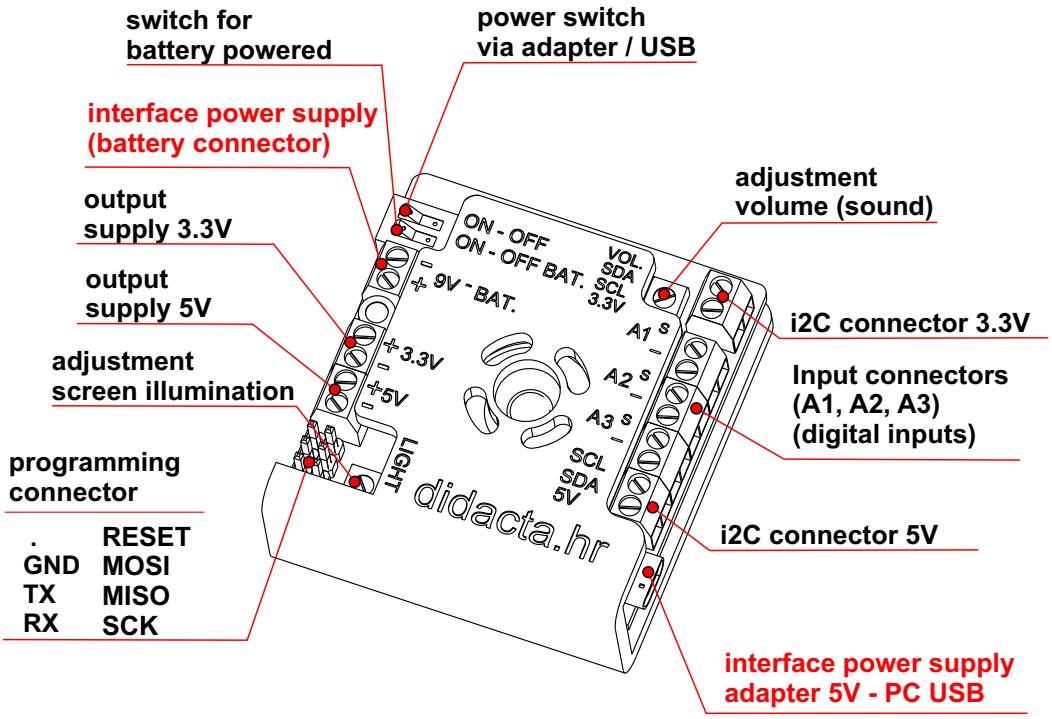
DIDACTA ADVANCE d.o.o.
Čakovec, Croatia (EU)
www.didacta.hr

1. POČETAK

1.1. Gornja strana display AD sučelja

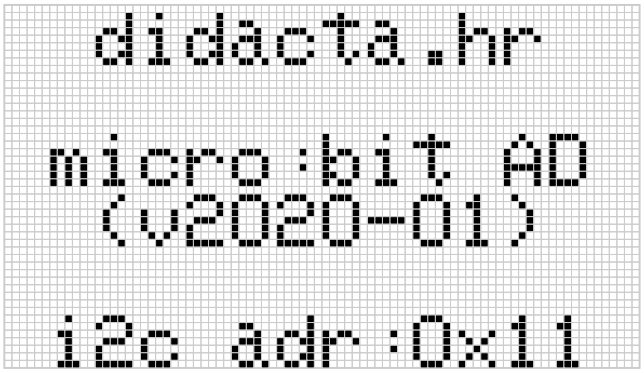


1.2. napajanje display AD sučelja putem baterije, adaptera ili USB porta (PC)



1.3. Pokretanje sučelja display AD

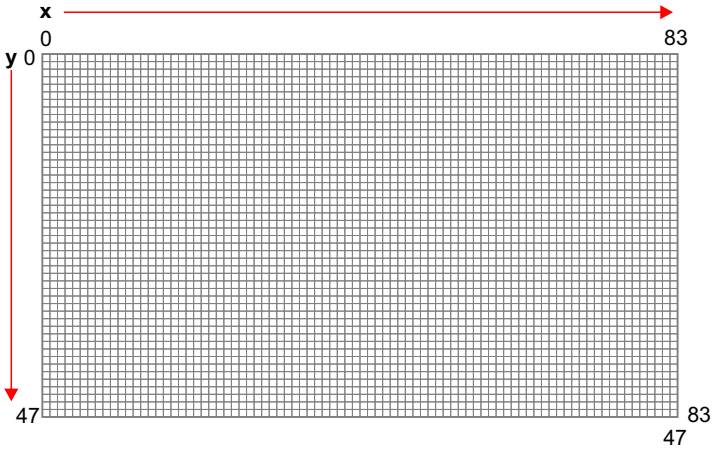
Nakon pokretanja, na ekranu sučelja display AD će se ispisati tekst prikazan na Slici 1.
Sučelje je spremno za rad.
Ako je već učitan program u Arduino, za ponovo pokretanje programa trebate pritisnuti tipku RESET.



slika 1.

1.4. Ekran sučelja

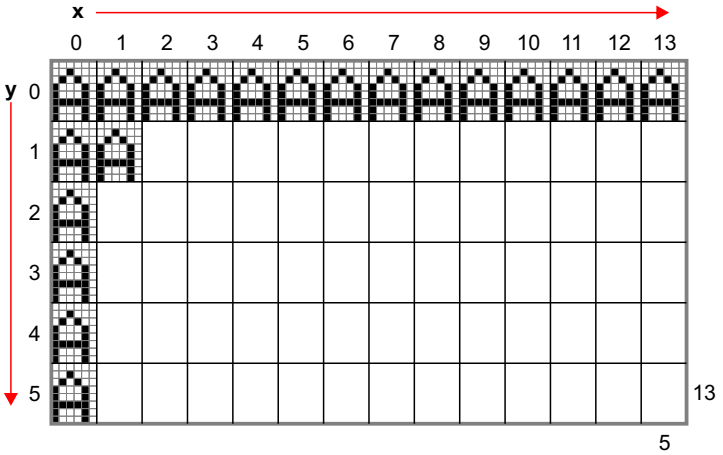
Display AD sučelje ima crno-bijeli ekran grafičke rezolucije 48 x 84 piksela (Slika 2.).



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. Arduino - C++

2.1. Osnovni Moduli

2.1.1. Moduli za komunikaciju između Arduina i sučelja AD. (**NE MJENJATI**)

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

```
void ispis(String poruka)
{
 Serial.println(poruka);
 poruka = poruka + ";";  // dodatni ; na kraju
 int duz = poruka.length();
 if (duz > 30)
 {
  String por1 = poruka.substring(0,30)+"+";
  char dio1[31];
  por1.toCharArray(dio1,duz);
  Wire.beginTransmission(0x11);
  Wire.write(dio1);
  Wire.endTransmission();
  String por2 = poruka.substring(30);
  duz = por2.length();
  char dio2[duz];
  por2.toCharArray(dio2,duz);
  Wire.beginTransmission(0x11);
  Wire.write(dio2);
  Wire.endTransmission();
  delay(duz*spd);
 }
 else
 {
  char copy[duz];
  poruka.toCharArray(copy,duz);
  Wire.beginTransmission(0x11);
  Wire.write(copy);
  Wire.endTransmission();
  delay(duz*spd);
 }
}
```

```
//--------------- VARIABLES USED IN MODULE
//-- MUST BE
char pit = "";
int spd = 8;
//---------------
```

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

Ovaj modul resetira (postavlja na početna stanja) program koji radi na sučelju AD. 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**.

```
void rest(){
   ispis("RST");
   while(true)
   {
      trazi();
      if (pit == 5){
         break;
      }
      delay(20);
   }
}
```

2.1.3. 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**.

```
void trazi(){
  Wire.requestFrom(0x11,1);
  while (Wire.available()) {
    pit = Wire.read();
  }
}
```

2.1.4. 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** .

```
void sound_light(){
   if (pit == 2){  // POINTS
      ispis("BIP;200;50;");
      ispis("LED;G;50;");
   }
   if (pit == 3) {  // LIFES
      ispis("BIP;800;50;");
      ispis("LED;R;50;");
   }
   if (pit == 4){  // FALL
      ispis("BIP;1200;50;");
      ispis("LED;R;50;");
   }
}
```

2.1.5. 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.

```
void begin_prog()
{
   ispis("CLS;");
   ispis("START;2;1;2;");
   ispis("G A M E;3;3;");
   delay(2000);
   ispis("CLS;");
}
```

2.1.6. 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.

```
void end_prog(){
   delay(300); // mora biti ako koristite funkciju trazi()
   trazi(); //  čita broj osvojenih BODOVA (SCORE)
   delay(300);
   ispis("CLS;");
   ispis("E N D;2;1;2;");
   ispis("G A M E;3;3;");
   ispis("Score:"+String(int(pit))+";3;4;");
   while (true){
      delay(5000);
   }
}
```

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");

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

U prvom programu koristimo funkciju **TEXT** . 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)

Na početku svakog programa dobro je koristiti funkciju **RST** programa koja briše vrijednosti koje je program sučelja display AD koristio u prethodnom radu.

3.3. Brisanje ekrana - CLS

Funkcija za brisanje sadržaja ekrana **CLS**.

ispis("CLS");

3.4. 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)

3.5. Ispis pomoćne memorije - BUFFER

Ova funkcija se koristi za prikaz sadržaja pomoćne memorije u koju se upisuju rezultati grafičkih funkcija, kao što je ispis teksta u grafičkom modu. Ta funkcija prikazuje stanje cijelog ekrana sa svim prikazima koji su prethodno upisani u pomoćnu memoriju. Korištenje ove funkcije izbjegava se titranje prikaza kod promijena brisanjem, te ponovnim ispisom novog stanja.

ispis("BUF");

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

Primjer za ispis jedne linije teksta u grafičkom modu

```
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");
```

Primjer za ispis više linija teksta u grafičkom modu

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)

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**.

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)

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**).

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)

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 4.

ispis("FIL;n");

FIL = naziv funkcije
n = 0 - 255

Slika 4. Ispis bajtova memorije ekrana

Primjer ispune (bojanja) ekrana linijama u razmaku od jednog piksela. Izračun vrijednosti boje (bajta) možete vidjeti na Slici 5.

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

Slika 5. Izračun 'boje' (bajta)

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.

ispis("DIS;n");

DIS = naziv funkcije
n = 0 - crno/bijelo 1 - bijelo/crno

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)

Funkcija za ispis pixela na ekran u grafičkom modu.

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)

4.2. Pomak teksta prema DOLJE za jednu liniju - SCD

ispis("SCD;r");

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

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

ispis("SBU;n");

SBU = naziv funkcije
n = broj 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

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)

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

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).

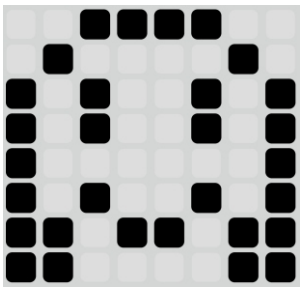
Kreirati bitmapu prema možete u Excel programu. Excel primjer sa predloškom prikazan je na slici 6. Kod prijenosa iz Excel datoteke trebate dodati broj botmape (CRNO - 1).

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								

bez broja bitmape

Slika 6.

ispis("**BIT;1;60;66;165;165;129;165;219;195**");

ispis("SPR;1;40;20;B"); //-- ispis bitmape u memoriji
ispis("BUF"); // -- ispis memorije na ekran

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

5.5. Kontrole pomaka - horizontalne i vertikalne - BUT

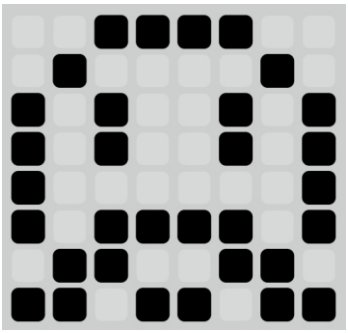
Za upravljanje objektom igrača u svim smjerovima potrebno je dodati i kontrole ulaza na Arduinu.

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

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

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.



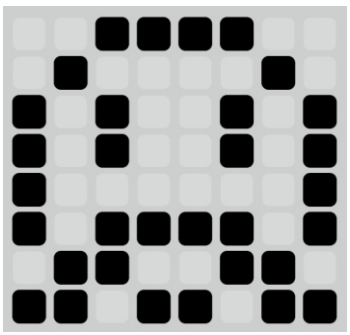
(9) Igrač

```
ispis("BIT;9;60;66;165;165;129;189;102;219"); // player
ispis("POZ;5;2");
```

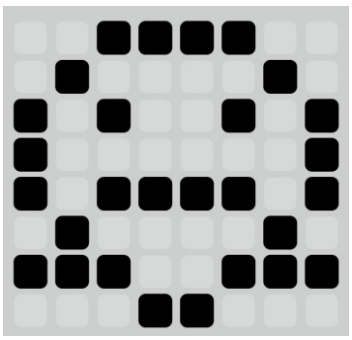
Funkcija PLAYER ujedno prikazuje stanje pomočne memorije (BUFFER) na ekranu, pa nije potrebno pozivanje funkcije **BUF**.

5.7. Animacija objekta igrača (bitmap 8)

Za animaciju objekta igrača potrebna je još jedna bitmapa. 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
```

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.

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
```

5.8. START game (2.1.5 - str. 5)

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_prog()** 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.

```
void begin_prog()
{
   ispis("CLS;");
   ispis("START;2;1;2;");
   ispis("G A M E;3;3;");
   delay(2000);
   ispis("CLS;");
}
```

5.9. COLLISION funkcija - KOL

Dopunite program novim objektom (**1**). Iscrtajte ga na ekran (**SPR**) nekoliko puta, na različite pozicije. 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

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.

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

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.)

```
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");
```

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).

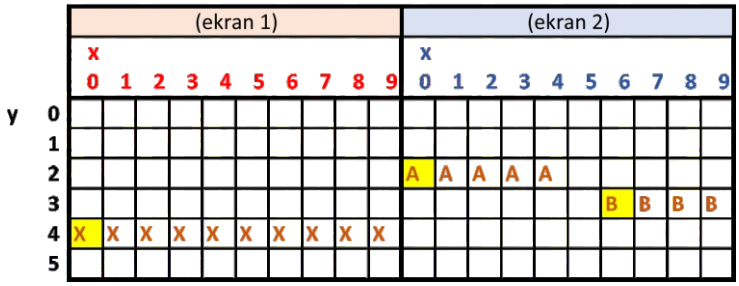
Za vertikalno isctavanje bitmape potrebno je promijeniti vrijednost polja «**hor/ver**» u **1**.

```
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");
```

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 7.

Kada želite kreirati više «**ekrana**» dobro je napraviti skicu kao što prikazuje primjer na Slici 7.
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.

```
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");
```

GAME_display_AD_04.ino

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 trebamo dodati aktiviranu naredbu **ASD (1)**.
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

Brzina horizontalnog pomaka objekata može se mijenjati naredbom **SPD**. Manja vrijednost varijable znači veću brzinu pomaka. Maksimalna brzina je ograničena na **10**, a početna (default) je postavljena na 100 (0 = 10). Brzinu možemo povećati 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

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

Da bi igra mogla normalno funkcionirati moramo zamijeniti klasične kontrole kretanja **igrača** novim. 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

5.15. Kontrolne funkcije

5.15.1. **Status igre - GAME status - trazi() (2.1.3 - 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.

```
void trazi(){
  Wire.requestFrom(0x11,1);
  while (Wire.available()) {
    pit = Wire.read();
  }
}
```

5.15.2. **Zvučni i svjetlosni efekti - sound_light() (2.1.4 - strana 5)**

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

```
void sound_light(){
    if (pit == 2){  // POINTS
        ispis("BIP;200;50;");
        ispis("LED;G;50;");
    }
    if (pit == 3) {  // LIFES
        ispis("BIP;800;50;");
        ispis("LED;R;50;");
    }
    if (pit == 4){  // FALL
        ispis("BIP;1200;50;");
        ispis("LED;R;50;");
    }
}
```

5.15.3. **Poruka za kraj igre - end_prog() (2.1.5 - strana 5)**

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

```
void end_prog(){
    delay(300); // mora biti ako koristite funkciju trazi()
    trazi(); //  čita broj osvojenih BODOVA (SCORE)
    delay(300);
    ispis("CLS;");
    ispis("E N D;2;1;2;");
    ispis("G A M E;3;3;");
    ispis("Score:"+String(int(pit))+";3;4;");
    while (true){
        delay(5000);
    }
}
```

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)

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

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

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

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.

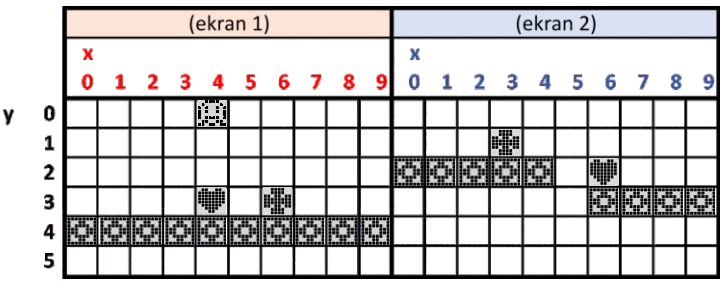
ispis("BON");

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");

5.16. Kompletna platformska igra



Slika 8.

Da bi igra bila kompletna dodali smo u svaki «**ekran**» objekte za dobivanje bodova ( ) i gubitak života ( ) prema pozicijam na skici (Slika 8.), te objekt za animaciju **igrača**.

```
//------- - 15.04.2021
// PLATFORM GAME LIKE
// PYTHON PROG40.PY
//-------
#include <Wire.h>

int tip1 = A0; // jump switch 1
int tip2 = A1; // jump switch 2
//--------------- VARIABLES - MUST BE
char pit = "";
int spd = 8;
//---------------

void setup() {
  pinMode(tip1, INPUT);
  pinMode(tip2, INPUT);
  //---------- START I2C COMM. - MUST BE
  Wire.setClock(100000);
  Wire.begin();  // join i2c bus
  //----------
 begin_def();
}

void loop() {
  trazi();
  if (pit == 9)
  {
    end_prog();
  }
  if (pit > 1 && pit < 5)
  {
    sound_light();
  }
  if (digitalRead(tip1) == LOW)
  {
    ispis("JMP;15;+;2");
  }
  if (digitalRead(tip2) == LOW)
  {
    ispis("JMP;15;-;2");
  }
    delay(100);
}
```

```
//---------------------------------------------- SETTINGS
void begin_def() // all settings you need in program
{
    delay(2000);
    ispis("RST");
    delay(200);
    ispis("CLS");
    delay(500);
    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"); // p.a.
    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");

    begin_prog();

    ispis("CLS");
    ispis("FX;1");
    ispis("POZ;4;0");
    ispis("ANI;3");
    ispis("KOL;1");
    ispis("GRV;1");
    ispis("ASD;1");
    ispis("SPD;10;1");
    ispis("PAD;1");
    ispis("BOD;0");
    ispis("LIV;5");
}
```

(nastavak programa na sljedećoj stranici)

```
//---------------------------------------- ALL MODULES
void ispis(String poruka)
{
  Serial.println(poruka);
  poruka = poruka + ";";
  int duz = poruka.length();
  if (duz > 30)
  {
    String por1 = poruka.substring(0,30)+"+";
    char dio1[31];
    por1.toCharArray(dio1,duz);
    Wire.beginTransmission(0x11);
    Wire.write(dio1);
    Wire.endTransmission();
    String por2 = poruka.substring(30);
    duz = por2.length();
    char dio2[duz];
    por2.toCharArray(dio2,duz);
    Wire.beginTransmission(0x11);
    Wire.write(dio2);
    Wire.endTransmission();
    delay(duz*spd);
  }
  else
  {
    char copy[duz];
    poruka.toCharArray(copy,duz);
    Wire.beginTransmission(0x11);
    Wire.write(copy);
    Wire.endTransmission();
    delay(duz*spd);
  }
}

void trazi(){
  Wire.requestFrom(0x11,1);
  while (Wire.available()) {
    pit = Wire.read();
  }
}

void begin_prog()
{
    ispis("CLS");
    ispis("START;2;1;2");
    ispis("G A M E;3;3");
    delay(2000);
}

void end_prog(){
    delay(300);
    trazi();
    delay(300);
    ispis("CLS");
    ispis("E N D;2;1;2");
    ispis("G A M E;3;3");
    ispis("Score:"+String(int(pit))+";3;4");
    while (true){
        delay(5000);
    }
}
void rest(){
    ispis("RST");
    while(true)
    {
        trazi();
        if (pit == 5){
            break;
        }
        delay(20);
    }
}

void sound_light(){
    if (pit == 2){  // POINTS
        ispis("BIP;200;50");
        ispis("LED;G;50");
    }
    if (pit == 3) {  // LIFES
        ispis("BIP;800;50");
        ispis("LED;R;50");
    }
    if (pit == 4){  // FALL
        ispis("BIP;1200;50");
        ispis("LED;R;50");
    }
}
```

GAME_display_AD_05.ino (prog40.py)

5.16.1. **Brisanje podataka iz pomoćne 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");

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.

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

5.16.3. **Brzina razmjene podataka (Arduino - 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 Arduino 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**.

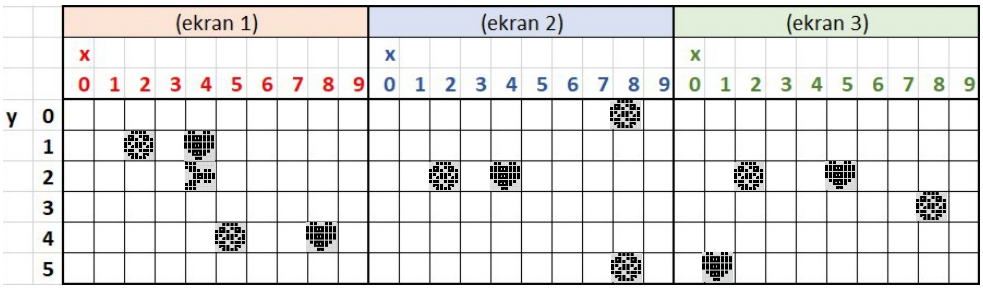
Program AD sučelja dozvoljava da probate i sa manjim vrijednostima.

spd = 8;

6. PRIMJER PROGRAMA

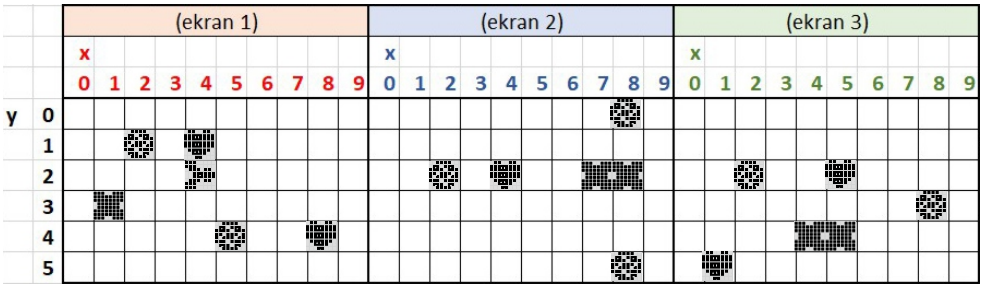
6.1. METEORI

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 9.).

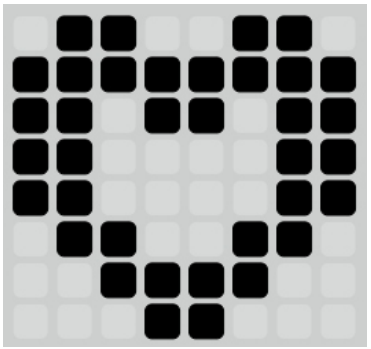


Slika 9.

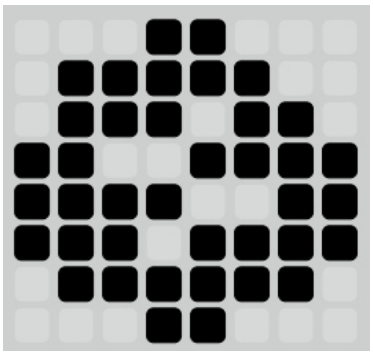
Ako želite otežati osvajanje bodova možete još dodati objekte koji će samo otežavati kretanje **igrača**, kao što je prikazano na Slici 10.



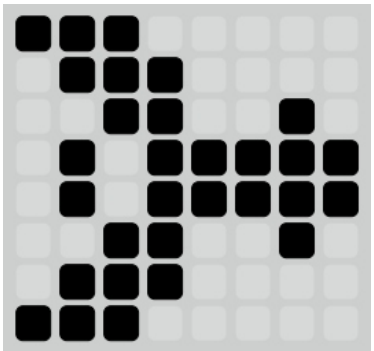
Slika 10.



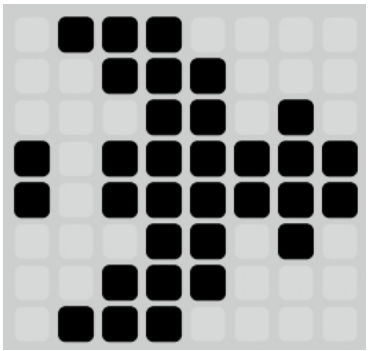
bitmap 2



bitmap 3



bitmap 9



bitmap 8

```
//------- - 15.04.2021
// DEMO METEORS GAME SAME LIKE PYTHON PROG42.PY
//-------
#include <Wire.h>

int tip1 = A0; // jump switch 1
int tip2 = A1; // jump switch 2
//--------------- VARIABLES - MUST BE
char pit = "";
int spd = 8;
//---------------

void setup() {
  pinMode(tip1, INPUT);
  pinMode(tip2, INPUT);
  //----------- START I2C COMM. - MUST BE
  Wire.setClock(100000);
  Wire.begin();  // join i2c bus
  //-----------
 begin_def();
}

void loop() {
  trazi();
  if (pit == 9)
  {
    end_prog();
  }
  if (pit > 1 && pit < 5)
  {
    sound_light();
  }
  if (digitalRead(tip1) == LOW)
  {
    ispis("BUT;+;Y;2");
  }
  if (digitalRead(tip2) == LOW)
  {
    ispis("BUT;-;Y;2");
  }
    delay(50);  // you can change this delay
}
```

```
//------------------------------------------  SETTINGS
void begin_def() // all settings you need in program
{
    delay(2000);
    ispis("RST");
    delay(200);
    ispis("CLS");
    delay(500);
    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");

    begin_prog();
    ispis("CLS");

    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;10;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;

}
//------------------------------------------- ALL MODULES
// jednako kao i u prethodnom programu
```

7. PRIMJER PROGRAMA

7.1. OSTALE FUNKCIJE

7.1.1. **Preuzimanje pozicije igrača ili očitanje digitalnih ulaza - GET (X, Y, 1, 2, 3)**

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. Ovu funkciju koristimo i za očitavanje vrijednosti digitalnih ulaza.

ispis("GET;n");

GET = naziv funkcije
n = varijabla (X, Y,A1=1,A2=2,A3=3)
A1-A3 digitalni ulazi - vrijednost 0 - 1

7.2. Primjer programa - očitavanje ulaza A1 - A3

```
// ------  - 15.04.2021
// DIGITAL READING INPUTS - A1,A2,A3
// ------
#include <Wire.h>

// variables for this sample
int I1 = 0;
int I2 = 0;
int I3 = 0;

int I1p = 0;
int I2p = 0;
int I3p = 0;
//--------------- VARIABLES - MUST BE
char pit = "";
int spd = 8;
//---------------

void setup() {
  //---------- START I2C COMM. - MUST BE
  Wire.setClock(100000);
  Wire.begin();  // join i2c bus
  //----------
  begin_def(); // start screen
}

void loop() {
    ispis("GET;1");
    trazi();
    I1 = String(int(pit)).toInt();
    if (I1 != I1p) {
        ispis("  ;5;2");
        ispis(String(I1)+";5;2");
        sound();
        I1p = I1;}
    ispis("GET;2");
    trazi();
    I2 = String(int(pit)).toInt();
    if (I2 != I2p) {
        ispis("  ;5;3");
        ispis(String(I2)+";5;3");
        sound();
        I2p = I2;}
    ispis("GET;3");
    trazi();
    I3 = String(int(pit)).toInt();
    if (I3 != I3p) {
        ispis("  ;5;4");
        ispis(String(I3)+";5;4");
        sound();
        I3p = I3;}
}

//-------------- for this sample
void sound(){
        ispis("BIP;800;50");
      //  salji("LED;G;50"); // LIGHT
}

//----------------------------------------  SETTINGS
void begin_def() // all settings you need in program
{
    ispis("DEL");
    delay(200);
    ispis("RST");
    delay(200);
    ispis("CLS");
    delay(500);
    ispis("INPUTS:;0;0");
    ispis("I1 =;0;2");
    ispis("I2 =;0;3");
    ispis("I3 =;0;4");
}
//---------------------------------------------- ALL MODULES
// jednako kao i u prethodnom programu
```

GAME_display_AD_03.ino

```
ispis("   ;0;0;G");              //-- brisanje vrijednosti
ispis("   ;65;0;G");
ispis("GET;X");                  //-- upit za X vrijednost
trazi();                         //-- očitaj X vrijednost
ispis(String(pit)+";0;0;G");   //-- ispis X vrijednosti na ekran
ispis("GET;Y");                  //-- upit za Y vrijednost
trazi();                         //-- očitaj Y vrijednost
ispis(String(pit)+";65;0;G"); //-- ispis Y vrijednosti na ekran
```

Primjer očitanja i prikaza x i y pozicije igrača.

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.