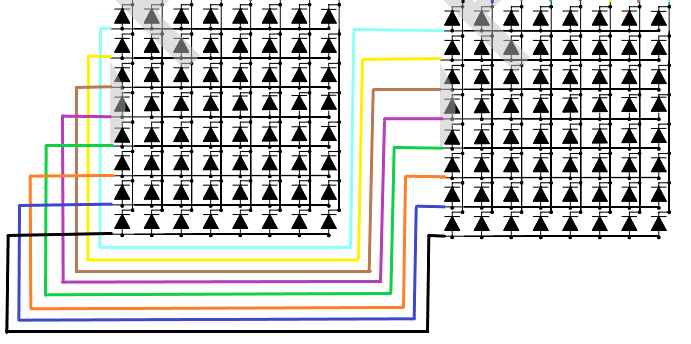
**Programozható LED mátrix kijelző**

**Az elképzelés**

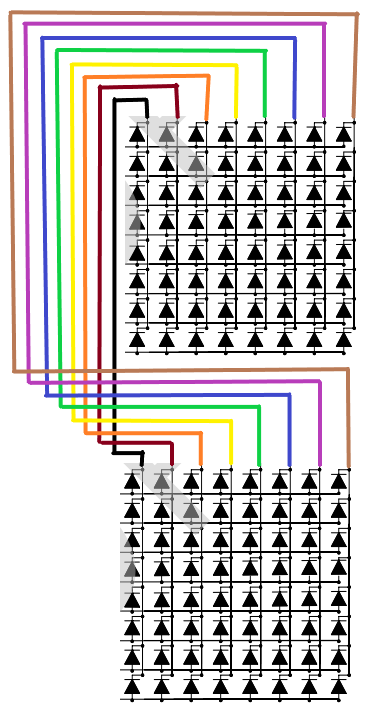
Komplex tervezésként programozható led mátrix kijelző készítése a célom, amely alkalmas lesz a C nyelven való programozás gyakorlására, például egyszerű játékok tervezésére és megjelenítésére. Olyan játékok elkészítése a célom, mint például a régi nyomógombos telefonokon rendszeresen előforduló Snake, Breakout, Spaceshooter, a valamivel újabb Flappy Bird és Doodle Jump. Ezekben a játékokban az a közös, hogy rendkívül egyszerű a grafikájuk, egy nagyon kis felbontású kijelzőn is ugyanazt az élmény nyújtják, mint a nagy felbontásúakon, továbbá nincs szükség egynél több szín alkalmazására a megjenítés során. A játékok programjának megírása során a programozónak két feladattal kell szembenéznie: a játék logikájának megértésével, működésének leképezésével, valamint a játék pillanatnyi állapotának megjelenítésével a játékos számára. A kijelzőt tehát a játékban bekövetkezett minden egyes változás során frissíteni kell majd. Annak érdekében, hogy a játékok között navigálni lehessen, valamint, hogy egyetlen nagy programba fogjuk össze őket, készíteni kell egy főmenüt, ahonnan minden egyes alkalmazás elérhető, és egy adott játékból való kilépés után a felhasználó ide térjen vissza. A játékok győzelem vagy vereség esetén szövegeket jelenítenek majd meg a kijelzőn, mint például az elmaradhatatlan „Game Over” felirat. Extraként egyes játékoknál lehetőség van a játék közbeni teljesítmény naplózására is, mint például a játékidő, a szerzett pontok, a találati pontosság feljegyzésére.

**A kivitelezés - felépítés**

A kijelző 16db(4x4) 8x8-as led mátrixból áll, melyekben az anódok soronként, a katódok pedig oszloponként össze vannak kötve. Ahhoz, hogy a teljes kijelzőt egyszerűen lehessen programozni össze kell kapcsolni ezeket a 8x8-as mátrixokat, mégpedig a megfelelő anód-sorokat, és katód-oszlopokat össze kell kötni. Ennek szemléltetésére szolgál a következő ábra, mely megmutatja, hogy az egymás mellett lévő mátrixokat hogyan kell összekapcsolni:



Az egymás alatt lévőket pedig:

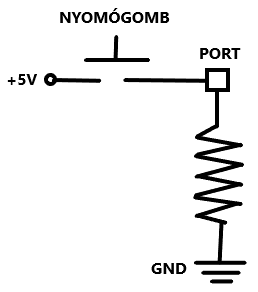


Tehát az egymás mellett lévő mátrixoknál az anód-sorokat, az egymás alatt lévő mátrixoknál a katód-oszlopokat kell összekötni. Ha ezt elvégeztük, akkor az elkészült nagy mátrix minden egyes LED-e egyértelműen kiválasztható, mivel minden anód-sor és katód oszlop csak egy pontban/LED-ben metszi egymást. Ennek köszönhető, hogy bármely LED-et fel tudjuk villantani, ha a megfelelő feszültségszintet adjuk a megfelelő sorra és oszlopra, azaz a LED-ek egyértelműen címezhetők.

Ha a LED-ek már megfelelően össze vannak kötve, akkor minden sorba vagy pedig minden oszlopba be kell építenünk egy ellenállást, hogy korlátozzuk a LED-ek áramát. A LED-eket a digitális logikának megfelelő 5V-ról fogjuk vezérelni, és egy led-re maximálisan 15mA juthat kb. 2,2V feszültség mellett, ezért a megfelelő ellenállás R=(5-2,2)/0,015~180 Ohm.

Az így összeállított kijelzőt már csatlakoztathatjuk a vezérlő eszközhöz, mely ez esetben egy PsoC mikrovezérlő, melynek 32db portjára az anód-sorokat, másik 32db portjára pedig a katód-oszlopokat kötjük.

Ahhoz, hogy a játékokkal játszani is lehessen, szükség van egy beviteli eszközre, amely az adott felhasználói interakcióhoz megfelelő jelet társít. Ez ebben az esetben egy egyszerű billentyűzet lesz, amelynek nyomógombjai a következő jelekkel lesznek ellátva: a számjegyek 0-9-ig, ←,↑,→,↓, enter, esc. A mikrovezérlő portjai egy lehúzóellenálláson keresztül földpontra lesznek kötve, és az egyik kivezetésükkel a +5V potenciálra kötött nyomógombokat a másik kivezetésükkel a portokra csatlakoztatjuk. A portokon tehát alapesetben földpotenciál jelenik meg, a nyomógombok lenyomásakor pedig +5V potenciálra kerülnek, ekkor a földpotenciál felé folyó áramot pedig a lehúzóellenállás korlátozza. A lehúzóellenállást úgy kell megválasztani, hogy ha a játékos az összes gombot meg találja nyomni egyszerre, akkor se folyhasson a megengedettnél nagyobb áram a földpotenciál felé. Portonként 10kΩ például megfelelő. A bekötést szemléltető kapcsolási rajz:



**A kivitelezés - programozás**

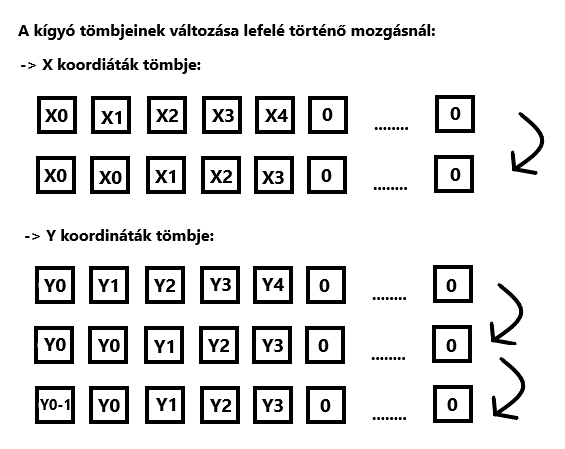
Az eszköz programozása a PsoC Creator nevű fejlesztőkörnyezetben fog megvalósulni, C nyelven.

A programozás a bevezetőben említetteknek megfelően két részre fog osztódni. Először a játék logikáját kell megérteni, és programnyelven megvalósítani. Ezt a Snake példáján keresztül fogom bemutatni.

A Snake esetében van egy koordinátarendszer, és egy kígyó, amelyről a játék minden pillanatában tudnunk kell hogy hol van a koordinátarendszerben. Azonban a kígyóval az a probléma, hogy nem egy pontból áll, hanem több pontból, és ráadásul folyamatosan változik a hossza. A kígyó minden pontjának tartózkodási helyét tehát el kell tárolnunk tömbben, méghozzá két tömbben, mert egy pont helyzete 2 koordinátával írható le, és a könnyebb kezelés érdekében egydimenziós tömböket használunk. A kígyó hosszának változása miatt a tömböket olyan méretűre kell felvennünk, hogy a leghosszabb kígyó is beleférjen, és a tömb kígyóval nem kitöltött helyeit ki kell nullázzuk, hogy tudjuk hol ér véget a kígyó.

Ezzel a módszerrel tehát már fel tudjuk venni a kígyó kezdő állapotát, azonban ebben a játékban felhasználói beavatkozás nélkül is folyamatos mozgásban van a kígyó, tehát a helyzete folyamatosan változik adott időközönként. Tehát figyelni kell azt is, hogy mikor telik le az az idő, amég a kígyó egy helyben lehet, és ekkor a tömjeit is módosítani kell. Az idő figyelése a mikrovezérlővel egyszerűen megvalósítható, nem úgy, mint a tömbök módosítása. Amikor letelik az idő, és a kígyónak mozdulni kell, tudnunk kell, hogy merre van arccal előre. Ez a mi koordinátarendszerünkben 4 irányban lehet, tehát szükségünk van egy változóra, amely értéke 4 különböző szám lehet, melyekhez egy-egy irány társul. Amikor tehát a kígyónak menni kell, meg kell vizsgáljuk a változónk értékét, és annak függvényében frissítenünk kell a tömböket.

Ha például a változóban a lefelé mutató iránynak megfelelő érték van, akkor a kígyó leghátsó darabját le kell vágjuk, és a megfelelő irányban (ez esetben lefelé) egy darabot a feje elé kell helyeznünk. Tegyük fel, hogy a tömbökben a legelső helyeken a kígyó fejének a koordinátája, legutolsó nem 0 helyeken a kígyó farkának koordinátája van. Most már a mozgást le tudjuk képezni a tömbökre. Az X koordináták tömbjében az összes nem 0 elemet az utolsó kivételével egyel jobbra helyezzük át. Így a „kígyó farka” eltűnik. A tömb első helyét pedig változatlanul hagyjuk, mert a feje csak Y irányban mozdult, X koordinátája változatlan. Az Y koordináták tömbjénél is elvégezzük ezt az eltolást, azonban az első hely koordinátáját a korábbihoz képet egyel csökkentjük, mert lefelé történt mozgás. (Ha a kígyó kimenne a kijelzőből természetesen nem csökkentjük az Y koordinátát, hanem a legnagyobbra írjuk át.) Ábra a tömbök módosításához:

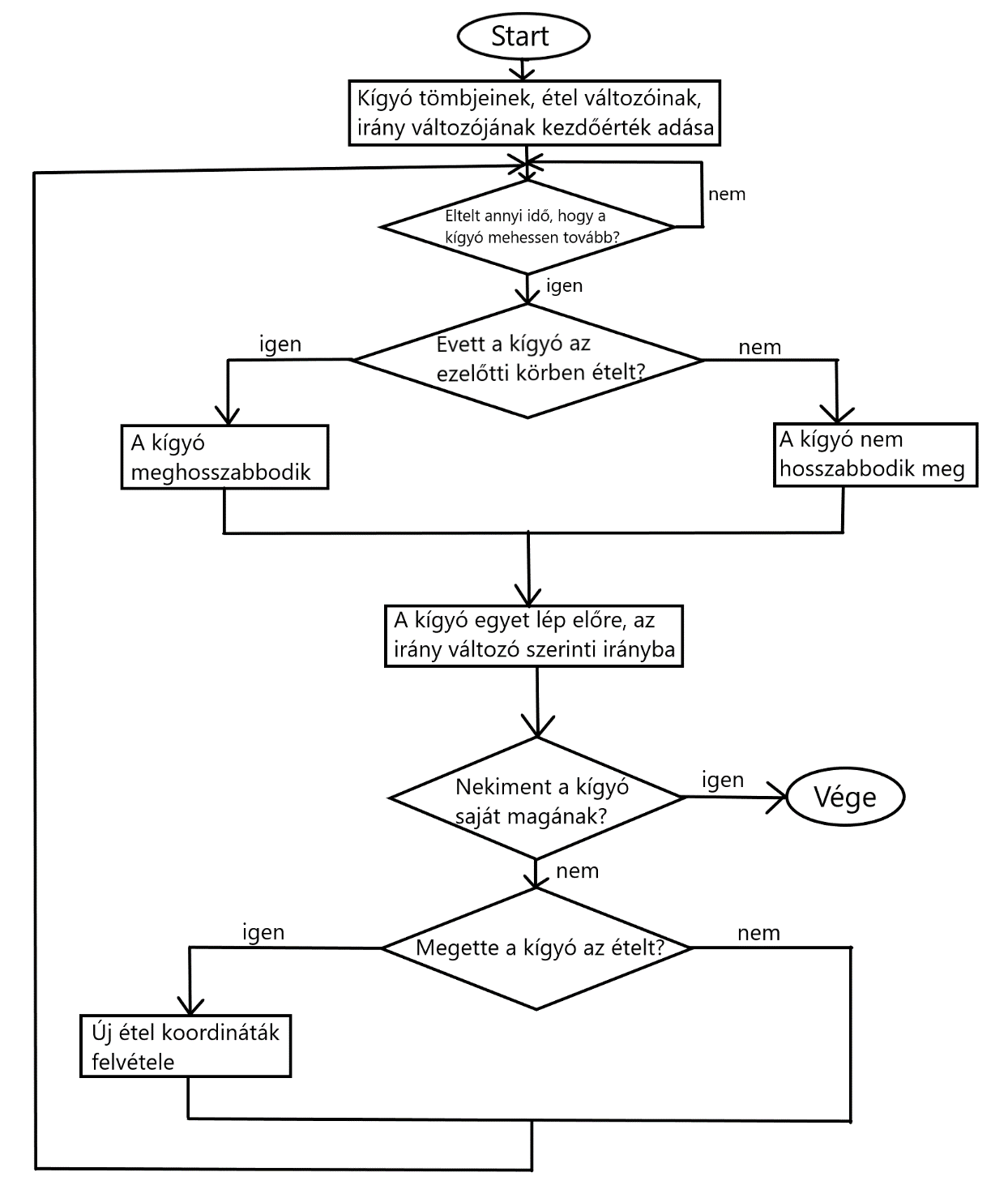


Most, hogy a kígyó magától való mozgását már tudjuk kezelni, bejönnek a képbe a felhasználói beavatkozások. Amikor a játékos megnyom egy irányt módosító gombot, akkor a kígyónak a mozgás irányát reprezentáló változója változik meg. Tehát folyamatosan figyelnünk kell a bemeneti jeleket, és ha gombnyomást észlelünk, akkor irányt kell módosítanunk. Oda kell figyelnünk arra, hogy ha a kígyó például felfelé megy, akkor azt a gombnyomást, ami az irányt lefelére módosítaná figyelmen kívül kell hagyjuk, ezt az elvet az oldalirányú mozgások esetén is alkalmazni kell. Egyelőre már tudjuk navigálni a kígyónkat a koordinátarendszerben.

A játék során szükség van egy ételt reprezentáló pontra, amelyet a koordinátarendszerben el kell helyezzünk. Ennek X és Y koordinátáját két változóban tároljuk. Amikor elhelyezzük ezt a pontot az kívül esik a kígyón, tehát 2 random koordinátát generálunk, és vizsgáljuk, hogy nem-e egyeznek meg a kígyó valamely pontjának koordinátáival. Az étel koordinátái mindaddig változatlanok, amég a kígyó „meg nem eszi”. A kígyó minden egyes elmozdulásánál vizsgálnunk kell, nem e futott bele a kígyó az ételbe, és ha ez megtörtént, akkor a kígyónak egy ponttal meg kell hosszabbodnia. A kígyó úgy hosszabbodik meg, hogy nem vágjuk le az utolsó koordinátáit, hanem tovább toljuk az első 0 helyre. Ekkor, mivel az étel el lett fogyasztva, új koordinátákat kell neki adnunk, és kezdődik a folyamat elölről.

A kígyó, ahogy egyre hosszabb lesz megnő a veszélye, hogy saját magának megy. Le kell tehát minden egyes elmozdulás után vizsgálnunk, hogy a kígyó fejének koordinátapárja nem-e egyezik meg valamely más testrészének koordinátapárjával. Ha ez megtörténne, akkor vége a játéknak, és tájékoztatnunk kell erről a felhasználót is egy „Game Over” felirattal.

A programozás ezen részét szemléltető folyamatábra:



A folyamatábra leírása pszeudo kódban:

------------------------------------------------------------------------------------------------------------------------------

//32x32 koordinátarendszer

//Legyenek az X és Y koordináták 1-32-ig

//Változók:

KigyoX (pozitív egészekből álló 32x32 elemű tömb)

KigyoY (pozitív egészekből álló 32x32 elemű tömb)

Irany (pozitív egész)

EtelX (pozitív egész)

EtelY (pozitív egész)

Megette (pozitív egész - 0 vagy 1 lehet)

MentettIdo (pozitív egész)

PontosIdo (mikrovezérlő belső órájáról kapja az értéket, mindig a pontos idő az értéke)

FutoValtozo (pozitív egész)

FutoValtozo2 (pozitív egész)

GameOver (pozitív egész - 0 vagy 1 lehet)

VarakozasiIdo (pozitív egész)

//Kezdőértékadás a változóknak

FutoValtozo = 0;

//3 pont hosszúságú kígyó inicializálása

amíg (FutoValtozo < 32\*32)

ha (FutoValtozo < 3)

KigyoX[FutoValtozo] = 16;

KigyoY[Futovaltozo] = FutoValtozo+10;

különben

KigyoX[FutoValtozo]=0;

KigyoY[FutoValtozo]=0;

FutoValtozo = FutoValtozo+1;

//Étel kezdőértéke

EtelX = 7;

EtelY = 2;

//Irány inicializálása: 0-le, 1-fel, 2-jobb, 3-bal

Irany = 0;

//Egyelőre a kígyó még nem evett semmit az előző körben

Megette = 0;

//A várakozási idő az, amég a kígyó egy helyben van, mi választjuk meg, ms-ban értendő általában

VarakozasiIdo = 200;

//Az idő mentése

MentettIdo = PontosIdo

//A játéknak egyelőre nincsen vége

GameOver = 0;

//Addig tartson a játék, amíg nincsen GameOver

amíg (GameOver == 0)

amíg (PontosIdo - MentettIdo < VarakozasiIdo)

nem csinálunk semmit, várunk a kígyó továbbindulására;

MentettIdo = PontosIdo;

FutoValtozo = 0;

amíg (KigyoX[FutoValtozo] != 0)

FutoValtozo = FutoValtozo +1;

//a kígyó végének helyére érünk a tömbben

ha (Megette == 0) FutoValtozo = FutoValtozo-1

amíg (FutoValtozo != 0)

KigyoX[FutoValtozo] = KigyoX[FutoValtozo-1];

KigyoY[FutoValtozo] = KigyoY[FutoValtozo-1];

FutoValtozo = FutoValtozo-1;

//megtörtént a jobbra tolás

ha (Irany = 0)

ha (KigyoY[0] != 1) KigyoY[0] = KigyoY[0]-1;

különben KigyoY[0] = 32;

ha (Irany = 1)

ha (KigyoY[0] != 32) KigyoY[0] = KigyoY[0]+1;

különben KigyoY[0] = 1;

ha (Irany = 2)

ha (KigyoX[0] != 32) KigyoX[0] = KigyoX[0]+1;

különben KigyoX[0] = 1;

ha (Irany = 3)

ha (KigyoX[0] != 1) KigyoX[0] = KigyoX[0]-1;

különben KigyoX[0] = 32;

FutoValtozo = 0;

FutoValtozo2 = 0;

amíg (KigyoX[FutoValtozo] != 0)

amíg (KigyoX[FutoValtozo2] != 0)

ha (KigyoX[FutoValtozo]==KigyoX[FutoValtozo2]&&KigyoY[FutoValtozo]==KigyoY[FutoValtozo2]&&FutoValtozo!=FutoValtozo2)

GameOver = 1;

FutoValtozo2 = FutoValtozo2+1;

FutoValtozo2 = 0;

FutoValtozo = FutoValtozo+1;

ha (KigyoX[0] == EtelX && KigyoY[0] == EtelY)

Megette = 0;

FutoValtozo = 0;

amíg (Megette == 0)

Megette = 1;

EtelX = random(1-32);

EtelY = random(1-32);

amíg(KigyoX[FutoValtozo] != 0)

ha (KigyoX[FutoValtozo] == EtelX && KigyoY[FutoValtozo] == EtelY) Megette = 0;

FutoValtozo = FutoValtozo+1;

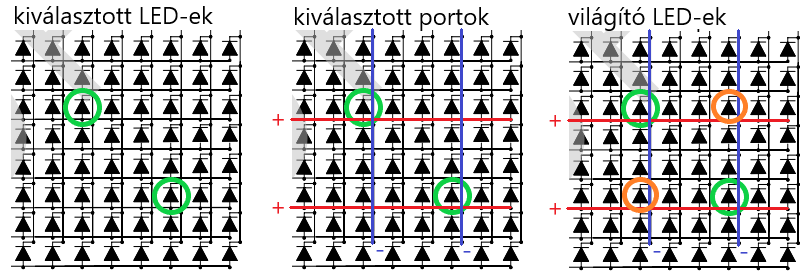
különben Megette = 0;

------------------------------------------------------------------------------------------------------------------------------

A programozáshoz tartozó másik nagy feladat a játék pillanatnyi állapotának a kijelzőn való megjelenítése. A Snake koordinátarendszerében lévő pontoknak a mátrix LED-ei felelnek meg. A koordinátákat tehát hozzá kell rendelni a nagy mátrix anód-soraihoz és katód-oszlopaihoz, amik már a mikrovezérlő portjaira vannak kötve, így minden X és Y koordinátához tartozni fog egy adott port, amelyre a megfelelő feszültség értékeket kell kiadni programozáskor.

Hogyan tudunk felkapcsolni egyetlen LED-et a nagy mátrixban? Arra az anód sorra, amelyre a LED anódja kötve van, +5V-ot adunk, és arra a katód-oszlopra, ahova a katódja van kötve 0V-ot adunk a mikrovezérlő két portja segítségével. Ha ki-bemeneti portokra kötöttük a sorokat és oszlopokat, akkor a nem használt portokat bemeneti módba kell állítani, csak a 2 szükséges port legyen kimenet. Ekkor csak ez az egy LED fog világítani az egész kijelzőn.

Nekünk azonban több pontból álló képeket kell megjelenítenünk a kijelzőn, de ezt nem tehetjük meg úgy, hogy a kép összes pontjának kikeressük a portpárjait, és a megfelelő feszültségeket adjuk rájuk, mivel ilyenkor olyan LED-ek is világítanának, amelyeket nem akartunk felkapcsolni. A jelenségnek az a magyarázata, hogy ha például egy adott anód-portra kikerül +5V feszültség, akkor ez a feszültség nem csak a világításra szánt LED anódján jelenik meg, hanem még egy sornyi LED anódja +5V-ra kerül. Ez egyetlen LED esetén nem okozna problémát, mivel csak a kiválasztott LED-nek megfelelő katód-oszlopra kerülne 0V feszültség, így csupán egy LED lenne, aminek anódján és katódján is a világításhoz szükséges feszültség jelenne meg. Ha azonban van 2db nem egy sorban lévő LED-ünk, és a két anód-sorra kiadjuk a +5V-ot, a két katód-oszlopra pedig a 0V-ot, akkor ezzel akaratlanul is 4db LED-ed címeztünk meg. Ennek szemléltetésére szolgál a következő ábra:



A probléma megoldása a multiplexelés, amely azt jelenti, hogy a kép összes pontját reprezentáló LED-et felvillanjuk, majd eloltjuk egymás után. Ez nagy sebességgel történik, így a szemünk nem érzékeli, hogy különálló villanásokat lát, hanem egy kép rajzolódik ki előttünk.

Visszatérve a Snake-re, multiplexeléssel kell kirajzolunk minden pillanatban a kígyót és az ételt. A programkódba adott időközönként be kell építenünk egy képkirajzolási kódrészletet. Ha a kirajzolást túl ritkán ejtjük meg, akkor villogó képet fogunk kapni a kijelzőn, ha pedig túl gyakran, akkor szükségtelenül kötjük le a mikrovezérlőt a kirajzolás idejére. Az is hibaforrás lehet, ha a játékban két kijelzőfrissítés között lezajlik egy olyan állapot, amiről nem kap képet a felhasználó. Például az első kirajzolásnál a kígyó az étel előtt áll, aztán megeszi az ételt és még egyel tovább halad. Ha csak ekkor következik be a frissítés, akkor a játékos számára kimarad egy képkocka.

A kirajzolás során a kígyó tömbjeiből megállapítjuk a kígyó koordinátáit, azokhoz hozzárendeljük a megfelelő portokat, és azokra a megfelelő feszültségeket adjuk a vezérlő segítségével. Ugyanígy járunk el az étel két koordinátájával.

Ennek a megvalósítására szolgáló pszeudo kód:

---------------------------------------------------------------------------------------------------------

//Kirajzolási kódrészlet

FutoValtozo = 0;

amíg (KigyoX[FutoValtozo] != 0)

Portra5VotAd(AnodPortHozzarendelo(KigyoY[FutoValtozo]));

Portra0VotAd(KatodPortHozzarendelo(KigyoX[FutoValtozo]));

portok visszaállítása bemenetekké;

FutoValtozo = FutoValtozo+1;

Portra5VotAd(AnodPortHozzarendelo(EtelY));

Portra0VotAd(KatodPortHozzarendelo(EtelX));

portok visszaállítása bemenetekké;

-------------------------------------------------------------------------------------------------------------------

A porthozzárendelő függvények argumentuma az X vagy Y koordináta, ezek a függvények minden koordinátához egy portszámot rendelnek, és ez a portszám lesz a visszatérési értékük. A portokra feszültséget adó függvények argumentuma így egy port száma lesz. Ezekben a függvényekben található meg az a kódrészlet, ami a kiválasztott portot kimeneti állapotba teszi.

Ezekkel a módszerekkel már egyszerű játékok írhatók a mikrovezérlőre. A felhasználó tájékoztatására szolgáló feliratokat a következőképpen juttathatjuk a kijelzőre. Mivel a kijelzőnk szélessége nem elegendően nagy egy hosszabb szöveg megjelenítésére, így a szövegnek „futnia” kell a kijelzőn, jobbról balra. Oda kell figyelnünk, hogy amikor már a szöveg vége fut a képernyőn, akkor a szöveg elejének meg kell várnia, hogy elegendő hely maradjon ki a vége után, tehát nagyjából 6 LED-nyi távolságot be kell iktatnunk a kettő közé, hogy ne tűnjön úgy, hogy egybefolyt a karaktersorozat eleje és vége.

A fent meghatározott szempontokat az alábbi módokon képezhetjük le programnyelvre. Szükségünk van több olyan tömbre, amelyek a szöveg egy adott LED-sorát szimbolizálják. Ezeket a tömböket 0-kal és 1-ekkel töltjük fel úgy, hogy ahol nincsen szövegrész, oda 0-t, ahol van, oda 1-et írunk, a tömbök a program működése során változatlanok maradnak. Annyi elemet kell tartalmazniuk, ahány LED-oszloppal leképezhető a teljes szöveg, és az eleje és a vége között tér is, és annyi ilyen tömbre van szükség, ahány LED-sorra felbontható a szöveg. Kelleni fog továbbá egy futóváltozó, amelynek értékéből megállapítható, hogy a hosszú tömbök éppen melyik eleme jelenjen meg a kijelző elején. Ezt a változót léptetve érhetjük el, hogy a szöveg „fusson” a kijelzőn.

Példa a Game Over felirat leprogramozására egy kisebb-8x8-as mátrixra pszeudo kódban:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 5 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 4 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 3 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 2 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 |

Az ábrán a szürkített rész jelzi, hogy a kijelzőn pillanatnyilag mi jelenik meg. A programozás során ezt a tartományt „léptetjük” a futóváltozónk segítségével.

-------------------------------------------------------------------------------------------------------------------

//8x8-as mátrix, X koordináták: 0-7, Y koordináták: 0-7

//Változók

//47 elemű tömbök -> futóváltozó 0-46-ig mehet

sor5 = [1,1,1,1,0,1,1,1,1,0,1,0,0,0,1,0,1,1,1,0,0,0,1,1,1,1,0,1,0,0,0,1,0,1,1,1,0,1,1,1,1,0,0,0,0,0,0];

sor4 = [1,0,0,0,0,1,0,0,1,0,1,1,0,1,1,0,1,0,0,0,0,0,1,0,0,1,0,1,0,0,0,1,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0];

sor3 = [1,0,1,1,0,1,1,1,1,0,1,0,1,0,1,0,1,1,1,0,0,0,1,0,0,1,0,1,0,0,0,1,0,1,1,1,0,1,1,1,1,0,0,0,0,0,0];

sor2 = [1,0,0,1,0,1,0,0,1,0,1,0,0,0,1,0,1,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0];

sor1 = [1,1,1,1,0,1,0,0,1,0,1,0,0,0,1,0,1,1,1,0,0,0,1,1,1,1,0,0,0,1,0,0,0,1,1,1,0,1,0,0,1,0,0,0,0,0,0];

FutoValt = 0;

FutoValt2 = 0;

//Az időzítéshez szükséges változók

MentettIdo;

PontosIdo;

VarakozasiIdo=200;

//Egy felhasználói beavatkozásig tartó ciklus

amíg (nincsen felhasználói beavatkozás)

FutoValt2 = 0;

MentettIdo = PontosIdo;

amíg (PontosIdo-MentettIdo<VarakozasiIdo)

amíg (FutoValt2<8)

ha (sor5[(FutoValt+FutoValt2)%47]==1)

Portra5VotAd(AnodPortHozzarendelo(5));

Portra0VotAd(KatodPortHozzarendelo(FutoValt2));

portok visszaállítása bemenetekké;

ha (sor4[(FutoValt+FutoValt2)%47]==1)

Portra5VotAd(AnodPortHozzarendelo(4));

Portra0VotAd(KatodPortHozzarendelo(FutoValt2));

portok visszaállítása bemenetekké;

ha (sor3[(FutoValt+FutoValt2)%47]==1)

Portra5VotAd(AnodPortHozzarendelo(3));

Portra0VotAd(KatodPortHozzarendelo(FutoValt2));

portok visszaállítása bemenetekké;

ha (sor2[(FutoValt+FutoValt2)%47]==1)

Portra5VotAd(AnodPortHozzarendelo(2));

Portra0VotAd(KatodPortHozzarendelo(FutoValt2));

portok visszaállítása bemenetekké;

ha (sor1[(FutoValt+FutoValt2)%47]==1)

Portra5VotAd(AnodPortHozzarendelo(1));

Portra0VotAd(KatodPortHozzarendelo(FutoValt2));

portok visszaállítása bemenetekké;

FutoValt2 = FutoValt2+1;

ha (FutoValt == 46) FutoValt = 0;

különben FutoValt = FutoValt+1;

---------------------------------------------------------------------------------------------------------

Ha mindezen lépéseket (játék leképezése, megjelenítése, feliratok leprogramozása) több játéknál is elvégezzük, már csak egy főmenüt kell szerkesztenünk, ahonnan minden játékunk elérhető. Ebben a főmenüben a felhasználónak tudnia kell léptetni a játékok között, bármelyiket el kell tudnia indítani, és a játékokból történő kilépés után ide kell visszatérnie.