**Szegedi Tudományegyetem**

**Informatikai Intézet**

**Moduláris Arduino játékdoboz bemutató alkalmakra**

**Modular Arduino game box for demonstrational purposes**

Szakdolgozat

**Készítette:**

**Tér Ábel**

mérnökinformatikus BSc szakos

hallgató

**Témavezető:**

**Dr. Mingesz Róbert**

adjunktus

**Szeged**

**2021**

# Feladatkiírás

//TODO

# Tartalmi összefoglaló

//TODO

A téma megnevezése:

A megadott feladat megfogalmazása:

A megoldási mód:

Alkalmazott eszközök, módszerek:

Elért eredmények:

Kulcsszavak:

Tartalom

[Feladatkiírás 2](#_Toc88337227)

[Tartalmi összefoglaló 3](#_Toc88337228)

[Bevezetés 5](#_Toc88337229)

[Irodalmi áttekintés 6](#_Toc88337230)

[Arduino, mint platform 6](#_Toc88337231)

[PIR mozgásérzékelő szenzor 6](#_Toc88337232)

[LCD kijelzők működése 7](#_Toc88337233)

[LCD1602 általános információk 8](#_Toc88337234)

[4-Wire mód 8](#_Toc88337235)

[MAX7219 LED meghajtó 8x8 LED mátrix-szal 8](#_Toc88337236)

[Szolenoid zár 9](#_Toc88337237)

[RFID technológia 9](#_Toc88337238)

[Címezhető LED Szalag 10](#_Toc88337239)

[Hardver tervezés 11](#_Toc88337240)

[Snake 12](#_Toc88337241)

[Memória 15](#_Toc88337242)

[Dino run 16](#_Toc88337243)

[Zárszerkezet 18](#_Toc88337244)

[Algoritmus tervezés 22](#_Toc88337245)

[Zárszerkezet 22](#_Toc88337246)

[Memória 26](#_Toc88337247)

[Snake 31](#_Toc88337248)

[Dino run 35](#_Toc88337249)

[Hardveres implementáció – 3D tervezés 35](#_Toc88337250)

[Tervezés 36](#_Toc88337251)

[Nyomtatás 38](#_Toc88337252)

[Összeszerelés 38](#_Toc88337253)

[Tesztelés 39](#_Toc88337254)

[Összefoglalás 40](#_Toc88337255)

[Hivatkozások 41](#_Toc88337256)

[Nyilatkozat 44](#_Toc88337257)

[Függelék 45](#_Toc88337258)

# Bevezetés

//TODO

# Irodalmi áttekintés

A dolgozat tematikája az Arduino köré épül, így fontos, hogy elsősorban megismerkedjünk magával a platformmal, valamint a későbbiekben felhasznált eszközök elméleti hátterével.

## Arduino, mint platform

Az Arduino egy nyílt forráskódú elektronikai platform, amely az eszközökön kívül egy teljes fejlesztőkörnyezetet biztosít számunkra. Az Arduino alaplapok képesek analóg, illetve digitális bementről olvasni és ezeket egy analóg, vagy digitális kimenetre írni. A különböző projektek megvalósítására egy C/C++ alapú programozási nyelv áll rendelkezésünkre, valamint egy teljes fejlesztői környezet. A platform egyik nagy előnye, hogy támogatja a harmadik féltől származó hardvereket, ezért a néha igen húzós árú eszközöket kiválthatjuk olcsóbb, viszont gyakran gyengébb minőségű alternatívákkal. [1]

## PIR mozgásérzékelő szenzor

A PIR, azaz Passive Infra Red mozgásérzékelő felismeri a környező tárgyak infravörös sugárzását. Az, hogy az érzékelendő objektum, legyen az ember, állat, vagy tárgy, mekkora mennyiségű infravörös sugárzást bocsát ki, függhet a hőmérsékletétől, valamint az anyagi felépítésétől. A PIR szenzor egy piroelektromos szenzor párt használ, ami a környezet hőenergiáját érzékeli. Ez a két érzékelő egymás mellett helyezkedik el és amikor a két érzékelő között a jelkülönbség megváltozik, akkor az érzékelő jelez. Az infravörös sugárzás erre a két szenzorra összpontosul, hála az eszköz borítójaként is szolgáló lencsék csoportjának.



1. ábra PIR szenzor működés közben [22]

Az ezközön található két potenciométer segítségével az érzékenységet, valamint a mozgás érzékelése utáni késleltetést állíthatjuk. A PIR szenzorok fő alkalmazási területe, az olyan feladatok, amiknél nem kell a mozgó tárgy pontos helyét meghatározni, hanem elég magát a mozgást észlelni, ilyen felhasználási kör lehet például egy egyszerű mozgásérzékelő lámparendszer, vagy biztonsági berendezések otthonainkban. [2]



2. ábra HC-SR501 PIR mozgásérzékelő alulról [2] (bal), HC-SR501 PIR mozgásérzékelő szemből [21] (jobb)

LCD kijelzők működése

A folyadékkristályos kijelzők régóta nagy népszerűségnek örvendenek. Alacsony áram felvételének köszönhetően előszeretettel alkalmazzák a mikrovezérlők világában. Az LCD kijelzők nem bocsátanak ki fényt. Ehelyett háttérvilágítással működnek és a fény áthaladását akadályozzák, mintha kis ablakokat nyitnánk ki és csuknánk be, a fény blokkolása érdekében. A kijelzők belsejében használt folyadékkristály két polarizált anyag közé van helyezve. A kristályok tájolásától függően képesek zárni, vagy átengedni a fényt. [3]



3. ábra LCD működése [23]

### LCD1602 általános információk

Az LCD1602 kijelző modul egy olcsó és népszerű LCD kijelző. Többféle színben is megtalálható a piacon, többek között kék, sárga, valamint zöld színben. Arduino-hoz és Raspberry Pi-hoz is egyszerűen csatlakoztatható. Két típusú adatot küldhetünk a kijelzőre. Az egyik az ASCII karakterek, amiket szeretnénk megjeleníteni a kijelzőn. A másik típus a vezérlő karakterek, amelyek a különböző funkciókat hivatottak aktiválni. [4]



4. ábra LCD1602 pin kiosztás [4]

### 4-Wire mód

Mivel az LCD modul párhuzamos adatbevitelt használ, 8 adatkapcsolati csatlakozásra van szüksége a mikrovezérlővel, csupán az adatok átküldéséhez. Ha ehhez még hozzávesszük a többi vezérlő pin-t, akkor láthatjuk, hogy ez már túl sok elhasznált foglalat lenne, ami később kellemetlenségekhez vezethet. Ennek a problémának az egyik megkerülési módja a „4-Wire” mód, amit a legtöbbször használnak az ilyen projekteknél. Ebben az esetben az adatokat fél byteonként küldjük, ami csak 4 adatkapcsolatot követel meg. Ilyenkor a D4-től a D7-es input pinek használatosak, míg a többi pin nem kapcsolódik semmihez. [4]

MAX7219 LED meghajtó 8x8 LED mátrix-szal

A MAX7219 egy kompakt, soros bemenet/kimenet közös katód kijelző meghajtó, ami interfészként szolgál a mikroprocesszoroknak, 7 szegmenses-, sávdiagram kijelzők, illetve önálló LED-ek meghajtására, amiből akár 64 darabot is vezérelhetünk. A chipen található egy binárisan kódolt decimális (BCD) – code-B dekóder, ami a 7 szegmenses kijelzők saját karakterkészlete, többszörös leolvasó áramkör**,** szegmens- és számjegy driver és egy 8x8-as statikus RAM, amely minden egyes számjegyet tárol. Csupán egyetlen egy külső ellenállás szükséges a LED-ek szegmensáramának beállításához. Egy 4 vezetékes soros interfész segítségével könnyen csatlakoztatható a legtöbb mikroprocesszorhoz. Az egyes számjegyek frissíthetőek, illetve címezhetőek a teljes kijelző átírása nélkül. A MAX7219 lehetővé teszi a felhasználónak, hogy minden számjegyhez megadja, hogy szeretne-e code-B dekódolást, vagy sem. Az eszköz rendelkezik egy 150μA-es alacsony fogyasztási móddal, tartalmaz analóg és digitális fényerősség szabályzót, valamint egy scan-limit regisztert, ami lehetővé teszi a felhasználónak, hogy 1 és 8 számjegy között jelenítsen meg számokat, egy teszt módot, ami az összes LED-et bekapcsolja. A működéshez 3V-ra van szüksége. A MAX7219 lehetővé teszi a 8x8-as LED mátrixok sorba kötését is, így kényelmesen használhatnánk egyszerre akár több kijelzőt, különösebb problémák nélkül. [5]

5. ábra MAX7219 láb kiosztás, alkalmazása 7 szegmenses kijelzővel, valamint 8x8 LED mátrixxal [5] [24]



Szolenoid zár

A szolenoid, másnéven mágneskapcsoló egy vezető huzalból készült, általában henger alakú tekercs, mely egy mozgatható vasmagot vesz körül. Amikor a tekercsen áram halad keresztül, egy mágneses tér keletkezik, ennek hatására a vasmag helyzetet változtat. Ezen az elven működik a szolenoid zár is, amik egy mozgó zárszerkezetet, vagy dugattyút használ a hagyományos zárak kulcsának és összekötőszerkezetének munkájához. A vezérlés aktiválásakor az előbb tárgyalt módon a mágneskapcsoló magához vonzza a szárszerkezetet és ennek következtében elmozdul a reteszelési helyzetéből. [6] [7]

RFID technológia

Az RFID (Rádió Frekvenciás Azonosítási Rendszer) két fő részből tevődik össze: egy azonosítandó tárgyhoz rögzített jeladóból és egy olvasó vevőkészülékből. Az olvasó egy rádiófrekvenciás modulból, valamint egy antennából áll össze, mely magasfrekvenciás elektromágneses mezőt generál. Ugyanakkor a címke általában csak egy passzív eszköz, aminek nincs szüksége áramellátásra. Helyette van benne egy microchip, ami tárolja és feldolgozza az információkat, melyet egy antenna segítségével fogad, majd küldi tovább. A címkén tárolt információk leolvasásához az olvasó közvetlen közelébe kell helyezni, viszont az elektromágneses tér segítsége miatt nem kell közvetlenül látótérben lennie. A címke ezt az elektromágneses mezőt használja arra, hogy az antennáján keresztül elektronokat fogadjon, miknek segítségével a chip energiával lesz ellátva.

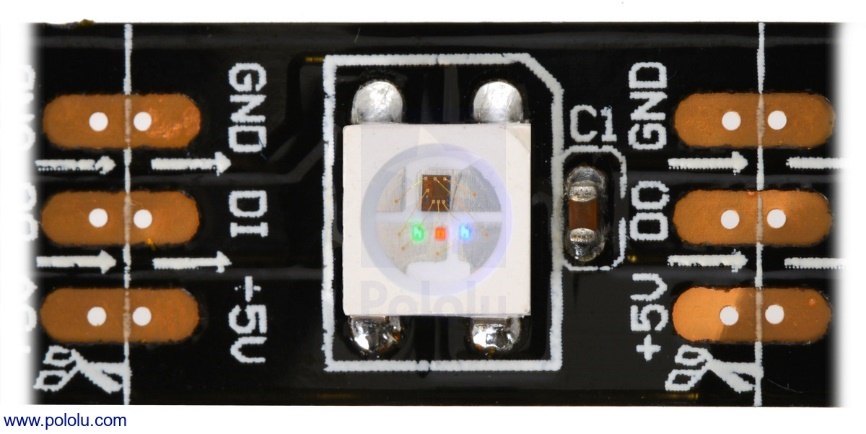
A címkében lévő árammal ellátott chip ezután visszaküldi az általa tárolt információkat egy rádiójel formájában. A visszaverődést, vagy az elektromágneses térben létrejövő változást érzékeli és értelmezi az olvasó, ami aztán a mikrovezérlőnek továbbítja az adatot. [8]



6. ábra RFID olvasó és tag egymás közelében [8]

Címezhető LED Szalag

A címezhető LED szalagok egymást követő, sorba kötött LED-ekből állnak, melyek három bemenettel és három kimenettel rendelkeznek. Kettő-kettő az áramellátásra, egy-egy pedig az adat fogadására és továbbítására szolgál. Minden LED integrált meghajtóval rendelkezik, amely lehetővé teszi, az egyes LED-ek színének és fényerejének egymástól független szabályzását. A csíkok kombinált LED/meghajtó IC-je a rendkívül kompakt WS2812B amely nagy LED-sűrűséget tesz lehetővé.



7. ábra Egy WS2812B alapú LED szalag egy szegmense közelről [27]

# Hardver tervezés

A projekt irányelve, hogy egy bemutató alkalom során az eszközt kipróbáló gyerekeknek megismertessük az Arduino világát. Ehhez olyan kapcsolások kellenek, amik nem túl bonyolultak, viszont látványos eredményt produkálnak egy megfelelően megírt kóddal. A következő oldalakon ismertetni szeretném a különböző részegységek kapcsolását. Igyekeztem egy olyan sorrendet felállítani, ami a legkevesebb alkatrészt/összeköttetést felhasznált, legegyszerűbben összeállíthatótól halad a bonyolultabbak felé. Ez a sorrend nem lesz azonos a részegységekre írt programok bonyolultsági sorrendjével.

A kapcsolási rajzokat a Fritzing nevű, nyílt forráskódú áramkörtervező programmal készítettem, ami bejelentkezés után ingyenesen letölthető, és rendkívül egyszerűen használható. Rengeteg Arduino-hoz is használható tervezőprogramot találhatunk az interneten, azonban ami a Fritzinget kiemeli a többi közül, hogy mi magunk is készíthetünk egyedi komponenseket [9], valamint használhatjuk a hatalmas felhasználóibázisnak köszönhetően létrejött rengeteg egyedi alkatrész egyikét. Ilyenkor nincs más dolgunk, mint letölteni a megfelelő *fzpz* kiterjesztésű fájlt és beimportálni azt.



8. ábra Saját beimportált alkatrészek, valamint új importálás lehetősége a Fritzing nevű szoftverben.

A szakdolgozat keretein belül nem készítettem saját alkatrészt, viszont felhasználtam több, mások által készített komponenst, melyek a következőek voltak:

* 8x8 LED modul MAX7219-vel [10]
* HC-SR501 PIR modul [11]
* KY-019 5V Relé Modul [12]
* Szolenoid szelep [13]
* Címezhető WS2812 RGB LED szalag [15]

A játékdoboz lényege, hogy moduláris legyen, részegységeit külön-külön is használni lehessen, illetve együtt egy egészet alkosson. Így fontos volt, hogy az egységek saját tápellátással rendelkezzenek. Ebből következik, hogy az összes kapcsoláson található egy 9V-os elem, egy tolókapcsolóval, ami rá lett kötve az Arduino megfelelő lábaira, ezzel biztosítva a hordozhatóságot, valamint lehetőség van bármikor ki- és bekapcsolni az eszközöket. Az elem, valamint a kapcsoló mindegyik részegységben megtalálható, ezért ezekre nem térek ki külön.

A tápellátáson kívül közös elem az Arduino Nano, mely az öszeszerelés során is a négy egységből háromban megtalálható. Azért válaszottam a Nano méretet, mivel a pin kiosztása elégséges a megvalósításokhoz, méretéből adódóan könnyebben integrálható a projektbe, valamint árát tekintve is jobb választás. A negyedikben egy Arduino Mega lapnak megfelelő Geekcreit Mega [16] nevű panel található, ennek oka, hogy a kezdőkészlet [17], amiből az egész projekt elindult ezt tartalmazta, így logikus volt ezt is felhasználni.

Snake

Összeszerelését tekintve talán kijelenthető, hogy a legegyszerűbb projektről van szó. Felhasznált alkatrészek:

* MAX7219 modul 1088AS 8x8 LED mátrix-szal
* Analóg joystick modul
* Tekerhető potenciométer

MAX7219 modul 1088AS 8x8 LED mátrix

Bekötése Arduino-val való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| *GND* | Földelés. | Földre kötve. |
| *VCC* | Tápellátás. | Tápfeszültségre kötve. |
| *CS* | Chip-választó bemenet. | Arduino **D11**-es lábára kötve. |
| *CLK* | 10MHz maximális sebességű soros óra bemenet. | Arduino **D10**-es lábára kötve. |
| *DIN* | Soros adatbemenet. Az adatok a CLK emelkedő élére töltődnek be a belső 16 bites shift regiszterbe. | Arduino **D12**-es lábára kötve. |

Felhasználása:  
A játék kijelzője, animációk megjelenítésére szolgál.

Joystick

Bekötése Arduino-val való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| *VCC* | Tápellátás. | Tápfeszültségre kötve. |
| *GND* | Földelés. | Földre kötve. |
| *VRx* | Vízszintes irányú változtatható ellenállás. Vízszintes irány meghatározása. | Arduino **A2**-as lábára kötve. |
| *VRy* | Függőleges irányú változtatható ellenállás. Függőleges irány meghatározása. | Arduino **A3**-as lábára kötve. |

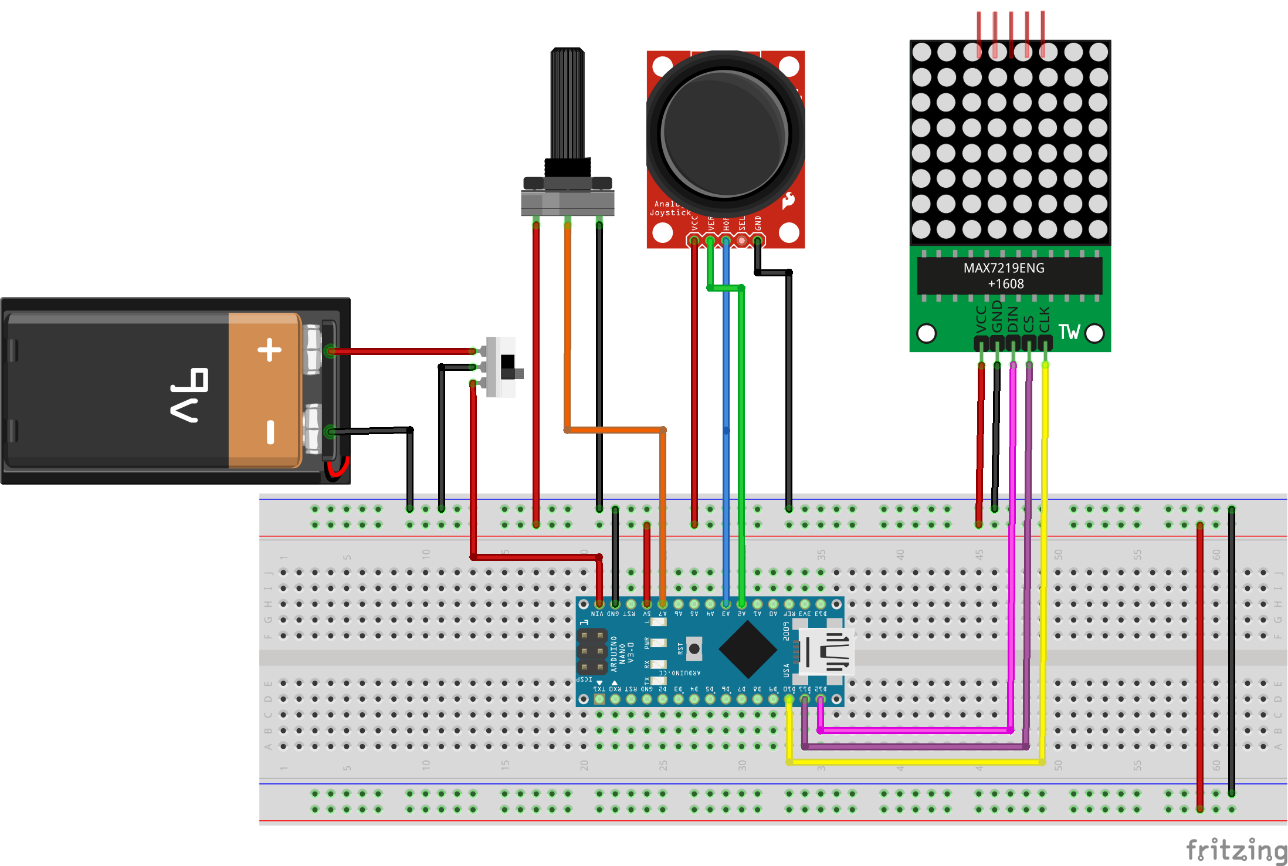
Felhasználása:   
A játék irányítására használt analóg bemenet.

Tekerhető potenciométer

Bekötése Arduino-val való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| *GND* | Földelés. | Földre kötve. |
| *OUTPUT* | Változtatható ellenállás. | Arduino **A7**-es lábára kötve. |
| *VCC* | Tápellátás. | Tápfeszültségre kötve. |

Felhasználása:  
A játék sebességének szabályzására. VCC és GND megcserélhető, csak az irány változik.



9. ábra Snake játék kapcsolási rajza

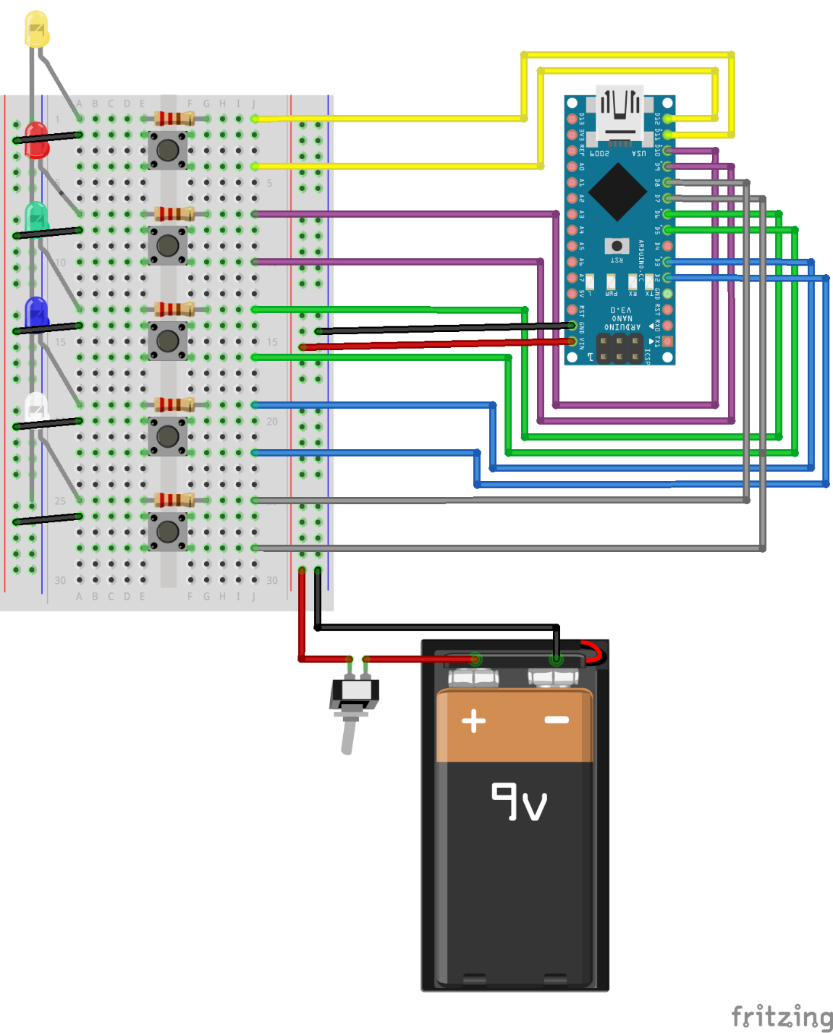
Memória

A memóriajáték valamivel több alkatrészből áll, így egy kicsit bonyolultabbnak tűnhet. Ugyanakkor ugyanazt a kapcsolást kell megismételni ötször, így ez is egy igen könnyen megvalósítható projekt kezdők számára. Itt, ami fontos, az a megfelelő sorrend, illetve, ami még trükkös lehet, hogy a LED-ek megfelelő irányba nézzenek, különben azok egész egyszerűen nem fognak világítani.

Felhasznált alkatrészek:

* LED - sárga
* LED - piros
* LED - zöld
* LED - kék
* LED - fehér
* 5 db nyomógomb
* 5 db 330Ω ellenállás
* 5 db 10kΩ ellenállás

Bekötés: //TODO át kell kötni a kódnak megfelelően

A nyomógombok az Arduino D2, D5, D7, D9, D11 pinjeire vannak kötve, velük párban pedig a LED-ek a 220Ω-os ellenállásokkal sorba kötve a D3, D6, D7, D10, illetve D12 lábra. A nyomógombok megfelelő lábai a szabályos működés biztosítása miatt a földre van kötve.

Dino run

A játékok közül a legösszetettebb az LCD-t használó Dino run. Bonyolultsága abból ered, hogy amellett, hogy szükségünk van több alkatrészre, a kijelző igen sok kapcsolást fog igényelni.

Felhasznált alkatrészek:

* Tekerhető potenciométer
* PIR mozgásérzékelő
* LCD1602 LCD modul
* 220Ω ellenállás

PIR szenzor bekötése Arduinoval való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| GND | Földelés | Földre kötve |
| Digital OUT | Mozgás érzékelés hatására jelet továbbít | Arduino D2-es lábára kötve |
| VCC | Tápellátás | Tápfeszültségre kötve |

Felhasználása:  
A játék irányítását hivatott elvégezni, mozgás hatására a felhasználó aktiválhatja a játékot.

Tekerhető potenciométer bekötése LCD-vel való használathoz:

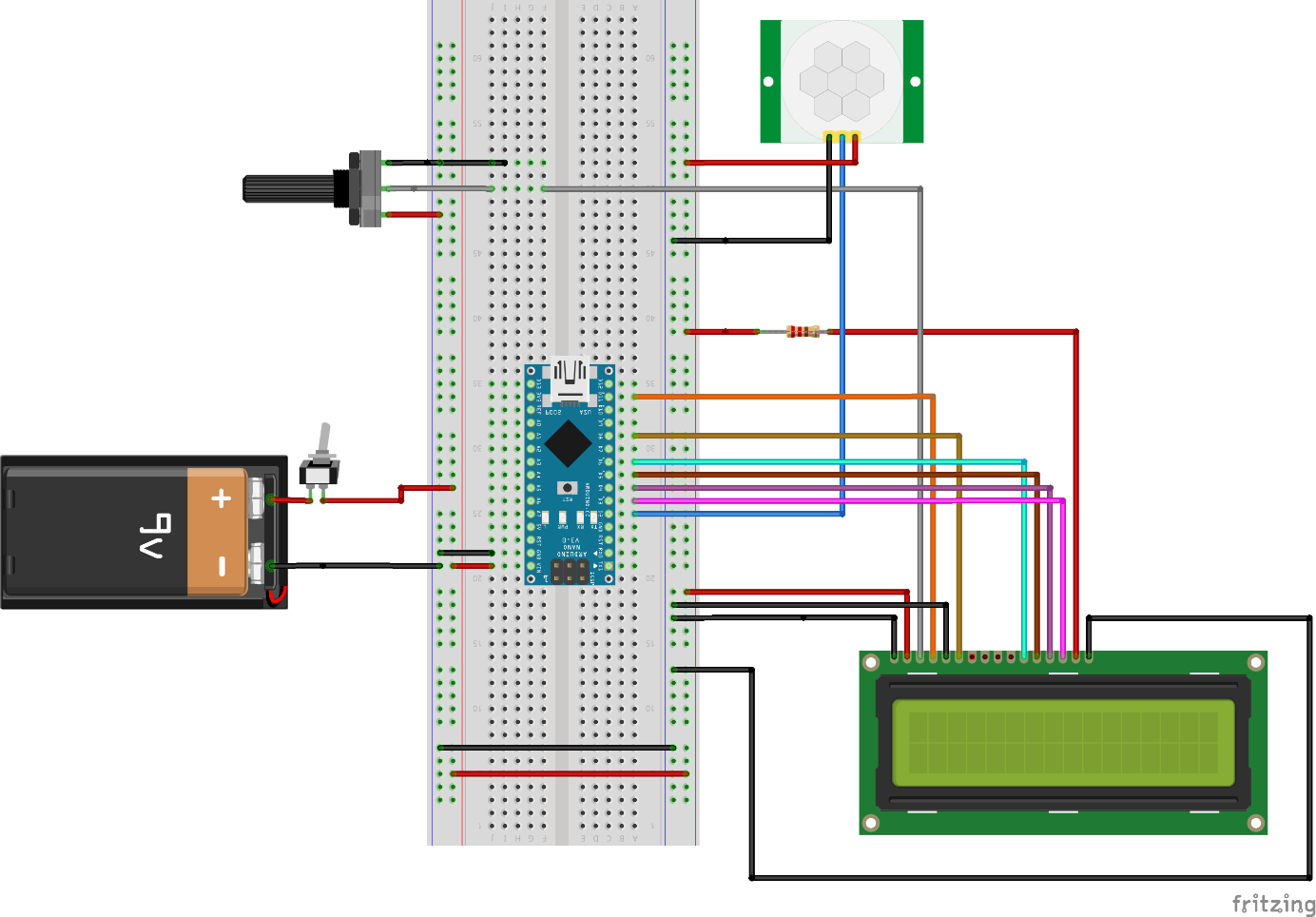
|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| GND | Földelés | Földre kötve |
| OUTPUT | Változtatható ellenállás | LCD V0 lábára kötve |
| VCC | Tápellátás | Tápfeszültségre kötve |

Felhasználása:  
A potenciométert eltekerve lehetőségünk van az LCD kontrasztjának szabályzására.

LCD1602 LCD modul bekötése Arduinoval való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| VSS | Földjel | Földre kötve |
| VDD | LCD tápellátás | Tápfeszültségre kötve |
| V0 | Kontraszt szabályzás | Tekerhető potenciométer OUT lábára kötve |
| RS | Regiszterválasztó jel | Arduino D11 lábára kötve |
| R/W | Olvasás/írás kiválasztó jel | Földre kötve |
| E | Műveletet engedélyező jel | Arduino D8 lábára kötve |
| D0-D7 | 8 bites üzemmódban használt adatbuszvonalak | - |
| D4-D7 | 4 bites üzemmódban használt adatbuszvonalak | Arduino D2-D6 lábára kötve |
| A | LCD háttérvilágítás anódja | 220Ω-os ellenállással sorban a tápfeszültségre kötve |
| K | LCD háttérvilágítás katódja | Földre kötve |

Felhasználása:  
A játék kijelzőjeként használt, animációkat megjelenítő LCD panel.



Zárszerkezet

A zárszerkezet projekt kitűnik a többi közül. Azért érdemelte meg a bonyolultsági lista utolsó helyét, mivel a hozzá felhasznált alkatrészek között több olyan is szerepel, ami egy Arduino kezdőkészletben nem található meg, viszont elengedhetetlen az alapműködéshez. Mindemellett ügyelnünk kell az RFID szenzor bekötésére, ugyanis azt nem köthetjük 5V-ra, hanem muszáj 3.3V-on feszültség alá helyezni.

Felhasznált alkatrészek:

* Piezo hangszóró
* RFID szenzor
* Relé modul
* Szolenoid zár
* 4 x 18650 elemtartó, elemekkel

Piezo hangszóró bekötése Arduinoval való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| Negatív láb | Földelés | Földre kötve |
| Pozitív láb | Feszültség állításával a hang magassága változtatható | Arduino D2-es lábára kötve |

Felhasználása:  
Illetéktelen behatolás esetén hangjelzés kísérli a meghiúsult kísérletet.

RFID szenzor bekötése Arduinoval való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| SDA | Az SS pin az SPI-kommunikációban a chip engedélyező pin. Ezért akkor kapja a jelet, amikor a Mesternek (Arduino) SPI-kommunikációt kell végrehajtania. | Arduino D9-es lábára kötve |
| SCK | Az órajel küldésére szolgál. | Arduino D12-es lábára kötve |
| MOSI | Az RFID modul adatbemeneti csatlakozója SPI kommunikációban. | Arduino D11-es lábára kötve |
| MISO | SPI kommunikációért felelős az Arduinoval, valamint továbbítja az adatokat a modulból az Arduino felé. | Arduino D10-es lábára kötve |
| IRQ | A készülék alvó üzemmódba kapcsolhat, hogy energiát takarítson meg. Ebben az esetben az IRQ segít felébreszteni. | - |
| GND | Közös földelés. | Földre kötve |
| RST | Reset pin. Hiba esetén alaphelyzetbe állítja az eszközt, amennyiben az adott ideig nem válaszol. | Arduino D8-as lábára kötve |
| 3.3V | Tápfeszültség. Nem lehet nagyobb 3.3V-nál. | 3.3V-ra kötve |

Felhasználása:  
A megfelelő RFID tag közelbe helyezésével jelet küld az Arduinonak, ami ennek hatására kinyitja a zárat.

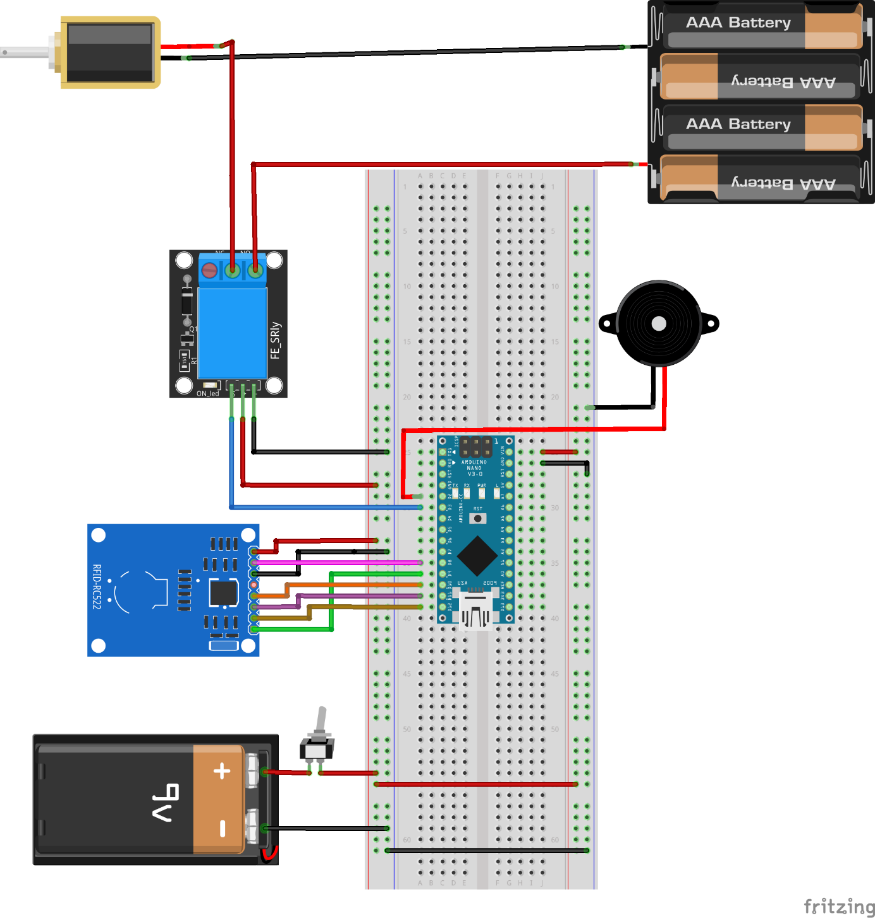
Relé bekötése Arduinoval való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| GND | Földelés | Földre kötve |
| VCC | Tápellátás | Tápfeszültségre kötve |
| IN | Jelbemenet | Arduino D3-as lábára kötve |
| NO | Normál esetben nyitott | Elemek pozitív lábára kötve |
| COM | Közös kapcsolat | Szolenoid zár pozitív lábára kötve |
| NC | Normál esetben zárt | - |

Felhasználása:  
A szolenoid zár nyitott, valamint zárt pozíciójába való helyezéséhez szükségünk van egy nagyobb feszültségű, külső áramforrásra, ami közvetlen bekötve kisüthetné az Arduinot. A relé az összeköttetés az Arduino és a zár között.

Szolenoid zár bekötése Arduinoval való használathoz:

|  |  |  |
| --- | --- | --- |
| **Modul pin** | **Felhasználása** | **Bekötése** |
| Pozitív | Tápellátás | Relé COM csatlakozójára kötve |
| Negatív | Földelés | Földre kötve |

Felhasználása:  
A szolenoid zár nyitott, valamint zárt pozíciójába való helyezéséhez szükségünk van egy nagyobb feszültségű, külső áramforrásra, ami közvetlen bekötve kisüthetné az Arduinot. A relé az összeköttetés az Arduino és a zár között.

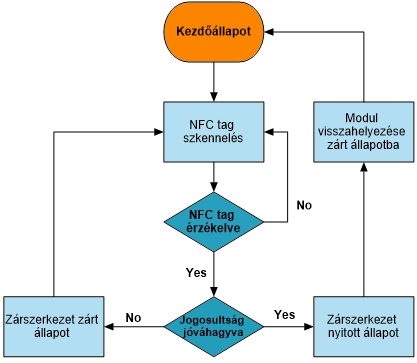
# Algoritmus tervezés

Az Arduinora írt programok tervezésénél fontosnak tartottam, hogy különböző bonyolultságú programok legyenek a projektben. A dokumentáció során úgy döntöttem, hogy a legegyszerűbben megérthetőtől kezdem a felsorolást, amihez minimális programozási ismeret elegendő, és onnan haladok a bonyolultabbak felé.

Mielőtt nekikezdtem a megvalósításnak, szerettem volna vizuálisan felvázolni, hogy mit is szeretnék a programtól elvárni. Ennek okán minden alegységhez készítettem folyamatábrákat, amik nagyvonalakban felvázolják, hogy a program működése közben az egyes állapotokból mik következnek, illetve, ha több lehetőség van a tovább haladásra, akkor milyen feltételek mellett fog bekövetkezni a kívánt eredmény.

A flowchartokat a ClickCharts [15] nevű szoftverrel terveztem meg, amelynek elérhető egy ingyenes verziója, ami főleg a kényelmi funkciók limitálásában tér el a fizetős verziótól.

//TODO DESIGNOLÁS

Zárszerkezet

10. ábra Zárszerkezet folyamatábra

Már a folyamatábrából is kivehető, hogy az NFC-vel nyíló zárszerkezet programja nem nagy bonyolultságú. Vegyük sorra, minek kell történnie a program futása közben.

Először is, a Serial.begin(9600) segítségével beállítjuk a soros adatátvitel adatátviteli sebességét másodpercenkénti bitben (baud). Ennek segítségével láthatjuk a soros monitoron a kiírt értékeket. Szükségünk van az SPI.h könyvtárra, ami lehetővé teszi az SPI eszközökkel való kommunikációt, ahol az Arduino a master eszköz. A könyvtárból az SPI.begin() az SPI-busz inicializálására szolgál, az SCK, MOSI és SS kimenetekre állításával, az SCK és MOSI alacsonyra, az SS pedig magasra húzásával. Az RFID olvasó használásához elengedhetetlenül szükségünk van az RFID.h könyvtárra, ami MFRC522 és más RFID RC522 alapú modulokhoz is használható. A könyvtár init() függvényével inicializálhatjuk a változók között példányosított RC522 chip-et. Mivel szeretnénk egy relét, valamint egy piezo csengőt használni, ezek pinjeit úgy kell konfiguráljuk, hogy kimenetként működjenek. A buzzernek, ha esetleg lenne generált hangja, a noTone() függvény segítségével leállítjuk a tone() által kiváltott négyszöghullám generálását. Nincs hatása, ha nem generál hangot, viszont a kéreltlen zajok elkerülhetőek vele. Fontos, hogy a szolenoid zárat úgy szeretnénk használni, hogy magas jelre nyíljon ki, ezért alapértelmezetten alacsonyra állítjuk a relé digitális jelét.

A következő állapotban az a feladat, hogy keresse, valamint érzékelje a közelben lévő RFID tag-et. Amennyiben nem érzékel neki megfelelő objektumot, folytassa tovább a keresést. Szerencsére az RFID függvénykönyvtárának segítségével igazán könnyen megoldható a folyamatos keresés. 

Az isCard() igazat ad vissza, ha egy kártya érzékelhető a közelben. Amennyiben a függvény hamis értékkel tér vissza, az azt jelenti, hogy nem érzékelt kártyát, tehát újra kell kezdeni a keresést. readCardSerial() igazat fog visszaadni abban az esetben, ha beolvasható egy UID az RFID tagből. Amennyiben ez nem lehetséges, újraindul a keresés. Fontos megjegyezni, hogy ezt a függvényt mindenképpen meg kell előzze az előbb említett isCard() metódus.

Amennyiben sikeres a beolvasás, ellenőrizni kell, hogy az adott tagnek mi az azonosítója. Ehhez létrehoztam egy változót, amiben eltárolom az összes karakter. Ahhoz, hogy ez kimenthető legyen, végig kell menni az egész UID-n és kimenteni a karaktereket. Debugolás céljából érdemes kiíratni a soros monitorra az azonosítót.

Szerencsére itt elég egy for ciklus használata, valamint a karakterek összefűzésére szolgáló concat() függvény.

A végső ellenőrzés egy nagyon egyszerű összehasonlítás lesz. Amennyiben a beolvasott UID megegyezik a programnak megadott UID-k valamelyikével, akkor jóváhagyásra kerül a belépés, ellenkező esetben nem. 

Jóváhagyott státusz esetén írunk a Serial Monitorra, magas jelet kell küldeni a relére, aminek hatására nyílik a zár, lejátszódik a címezhető LED szalagon az elfogadást jelző animáció, majd előre definiált idő után újra visszaáll a zár a kezdeti csukott pozíciójába.

Amennyiben olyan kártyával próbálkozunk, aminek az azonosítója nincs a jóváhagyottokhoz hozzáadva, akkor egy adott ideig tartó hangjelzés fogja jelezni a meghiúsult kísérletet.

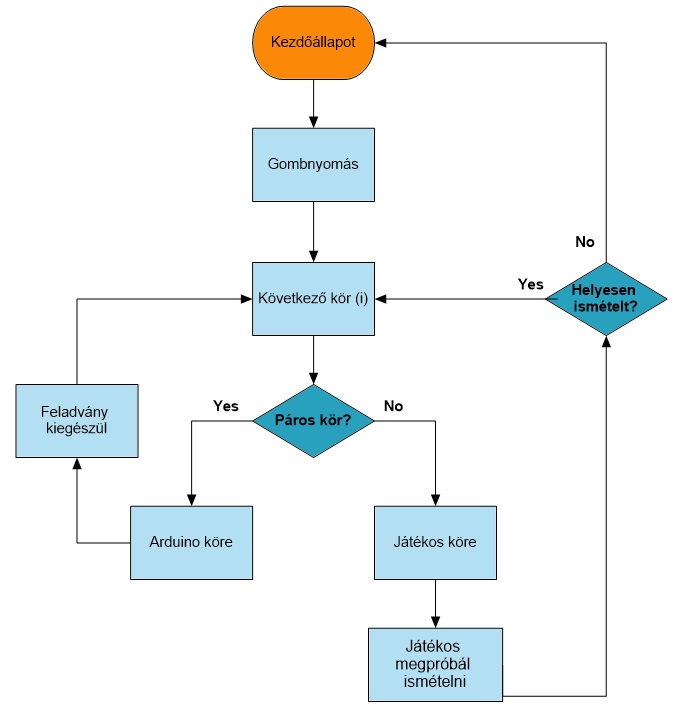


A LED szalag programozásához a <FastLED.h> könyvtár [17] került felhasználásra. Ennek segítségével a WS2812 típusú szalagon az inicalizálása után könnyedén lehet műveleteket végrehajtani. Így a szalag úgy viselkedik, mint egy tömb. Végig lehet az elemeken – amik konkrétan az egyes LED-ek – iterálni és azok színét, fényerősségét külön-külön módosítani.

A playAcceptSequence(), valamint a playDeclineSequence() is csak for ciklusokat használ. Az első balról jobbra felkapcsolja a LED-eket, amik zöld színt vesznek fel. Ehhez elég volt két ciklus, egymásba ágyazva. A belső ciklus a sorba következő LED-et kapcsolja, míg a külső reseteli az összeset és újraindítja a folyamatot ötször. 

Az elutasítást jelző függvény valamivel bonyolultabb. Mivel itt egyszerre megy középről a két irányba az animáció, így szükség volt egy segédváltozóra, a megvalósítás, pedig kettő-kettő egymásba ágyazott for ciklussal történik. A lényeg nem változik, csak a szín, illetve egyszerre két irányba terjed szét az animáció.

Memória



A memóriajáték lényege, hogy az Arduino mutat egy feladványt, amit a játékosnak el kell tudni ismételni. Amennyiben helyesen ismételt, folytatódik a játék, a feladvány kiegészül még egy taggal. Ez addig ismétlődik, amíg a játékos rosszul nem ismétel. Ilyenkor a játéknak vége és visszaáll az alapállapot addig, amíg új játék nem kezdődik.

Láthatjuk, hogy itt már több állapot van, mint az előző játéknál. Több feltételt kell kezelnünk és egy kicsit át kell látni a tömböket, illetve azok bejárását. Viszont itt is for ciklusokból, valamint if - else feltételes elágazásokból fog állni a program nagy része.

A programunkhoz most egy kicsit több változóra lesz szükségünk, ezek a következők:



Mielőtt a loop() függvény tartalmára térnénk, érdemes végig nézni néhány metódust, ami a későbbiekben sokat fog egyszerűsíteni a dolgunkon.



A setPinMode(byte mode) függvénynek a segítségével egyszerre állíthatjuk be, hogy INPUT, vagy OUTPUT módban szeretnénk használni a pin-eket, amik a pins[] tömbben lettek felsorolva .



writeAllPins(byte val) függvénynek köszönhetően az összes LED egyszerre lesz írható, így azok be-, vagy kikapcsolhatóak egyszerre.



Az előzőn alapszik a flashLeds(short frequency) függvény, ami arra szolgál, hogy a frequency paraméterben megkapott frekvencián villantsa fel az összes LED-et, ötször egymás után. Ezzel fogjuk jelezni, hogy a játékos hibázott és a játék véget ért.



A playSolution() végigmegy a jelenleg tárolt feladványon, kiírja, valamint felvillantja a megfelelő LED-eket.

Ezeken kívül van még egy ResetGame() függvény, ami a változók alapértékeire való visszaállításra szolgál.

A játék végekor az endGame() függvény hívódik meg, ami meghívja a flashLeds(50) függvényt 50-es frekvenciával, valamint a villogás után egy másodperccel meghívja a ResetGame()-et.

A loop() függvényben két fő állapotot kell megkülönböztetni. Az egyik, amikor az Arduino, a másik pedig amikor a játékos következik.

Először vizsgáljuk meg, mi történik, amikor páros kör van, az isPlayersTurn = false, azaz az Arduino van soron.



A setPinMode() segítségével OUTPUT módba állítjuk a pineket, ugyanis ebben az esetben a LED-eket szeretnénk használni. A randomSeed() inicializálja a pszeudo-véletlenszám-generátort, és a véletlen sorozat egy tetszőleges pontjáról indítja. Ez a szekvencia, bár nagyon hosszú és véletlenszerű, mindig ugyanaz. Ahhoz, hogy a random() által generált értékek sorozata későbbi futtatásakor eltérő legyen, akkor a randomSeed() segítségével inicializáljuk a véletlenszám-generátort egy viszonylag véletlenszerű, például esetünkben az analogRead(A0) olvasásával [16]. Így a solution[currentLength] = pins[random(0,numberOfPins)] sor segítségével már egy ténylegesen véletlen értéket adhatunk hozzá a feladványunkat tároló tömb végére. A hozzáadás, illetve feladvány hosszának növelése után a playSolution() segítségével megjelenik a játékos számára a feladvány a LED-eken. Ha ez megtörtént, az isPlayersTurn igazra vált, ezzel jelezve, hogy a játékos következik. Ezt követően az inputTime változóba elmentésre kerül a kör befejezésének ideje a millis() függvény segítségével, ami visszaadja, hogy hány milliszekundum telt el azóta, hogy az Arduino elkezdte futtatni az aktuális programot. [17]

A következő lehetőség, hogy a játékos van soron. Ebben az esetben, mivel a gombok vannak használatban a setPinMode() segítségével INPUT-ra állítjuk a megadott pineket.

A legelső ellenőrzés az lesz, hogy a játékos kifutott-e a gondolkodási időből. 

Amennyiben a játékos több ideig gondolkozott, mint a megengedett, vesztett.

Ha a megadott időn belül döntött, akkor a következő feltétel vizsgálat az lesz, hogy elengedte-e a gombot. 

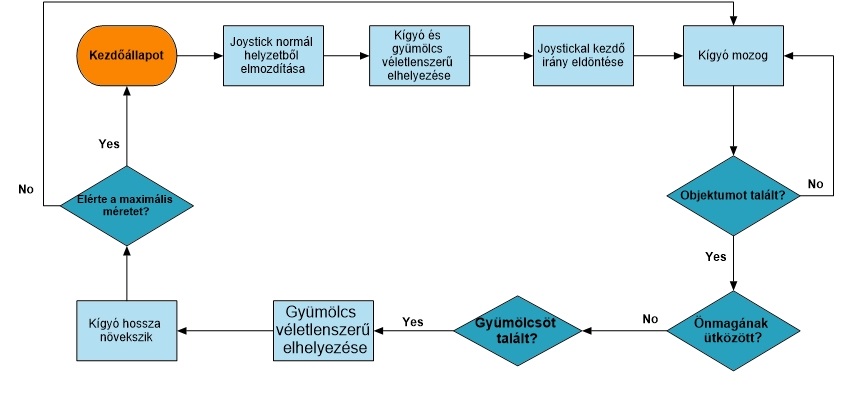
Az elvárt választ a solution[] tömb legvégéről kell kivenni. Ezek után végig kell menni az összes INPUT pinen, és megnézni, hogy van-e olyan aktív pin, ami nem az elvárt megoldás. Ha van, akkor az azt jelenti, hogy a játékos rossz gombot nyomott. Az isGameOver true értéket vesz fel, a játéknak vége.



A következő ellenőrzés megnézi, hogy valóban azt nyomtuk-e meg amit vártunk, valamint, ellenőrizzük, hogy nem történik folyamatos gombnyomás. Itt az inputTime felveszi a millis() függvény által felvett értéket, hogy a következő ellenőrzéskor figyelembe tudjuk venni a gombnyomások között eltelt időt. Ebben a kódrészletben növekszik az inputCount, valamint az isButtonPressed igaz értéket vesz fel, hogy biztosan ne ragadjon be a programunk és ne fusson véletlenül se rossz ágra.

A legutolsó rész a gomb felengedése utáni ellenőrzéseket hivatott elvégezni. Abban az esetben, ha erre a részre fut a kód, az isButtonPressed változó hamis értéket vesz fel. Ezután két ellenőrzés fog következni. Amennyiben az isGameOver változó igaz, azaz a játék során rosszul ismételt a játékos az endGame() szcenárió fog lejátszódni. Helyes ismétlés esetén, ha annyi gombnyomás történt, mint amennyi az adott körben elvárt lett volna, akkor a játékos köre befejeződött, újra az Arduino következik.

Snake

A Snake játék már a bonyolult kategóriába tartozik. Szükséges a vezérlési szerkezetek mellett még a struktúrák ismerete, valamint a kétdimenziós tömbök megértése is. Ennek átlátásához már valamivel több idő, illetve tapasztalat szükséges, ugyanakkor ez is elsajátítható némi idő ráfordításával.

A flowcharton is látszik, hogy annak ellenére, hogy nincs sok útvonal, és viszonylag lineáris a folyamat, mégis elég sok állapotból áll a program.

Első sorban a LED mátrix vezérléshez elengedhetetlen a LedControl.h könyvtár használata. Mivel a korábbiakhoz képest több pin lett felhasználva, ezért ebben a projektben a pinek változói egy struktúrában lettek létrehozva, ezzel átláthatóbbá téve a későbbi használathoz.

Sok globális változóra is szükség van, a kígyó kezdeti hosszának megadásához, a pillanatnyi hossz lekéréséhez, aktuális sebesség lekéréséhez, valamint a kígyó menetirányának lekéréséhez, illetve megadásához szükséges konstansok. A játékot szabályzó változók a fényerő szabályzásához szükséges változó, a játékteret tároló kétdimenziós 8x8-as tömb, valamint a játék állapotát tároló boolean változók, amik a nyerést, illetve vesztést hivatottak jelezni. A kígyó, a gyümölcs és a joystick kalibrálásához szükséges adatok tárolására szintén struktúra lett létrehozva, amely a Point és a Coordinate nevet viselik. Mindkettőnek két adattagja van, a Point ami a kígyó és a célpont egy sor és egy oszlop adattaggal, míg a Coordinate egy x és egy y -nal rendelkezik. Default értékük 0,0.



Az Arduino setup() függvényében a soros adatátvitel beállítása történik meg, a joystick kalibrálása a calibrateJoystic() függvény segítségével. Ezen felül a setup()-on belül megtalálható a játék kezdőállapotának beállítására létrehozott initGame() metódus. Az utóbbi a pinek ki-, valamint bemenetre való állítását, a LED mátrix újraindítását, fényerejének beállítását, valamint a kígyó kezdőpozíciójának beállítását végzi el.

A program futása közben a loop() függvény négy másik metódust fog meghívni folyamatosan egymás után. Ezek sorban a gyümölcs generáló generateFood(), a joystick aktuális állapotát felmérő scanJoystick(), a snake helyzetét kiszámító calculateSnake(), valamint a nyerés illetve vesztés állapotot beállító setState() függvény.

Sorban haladva a generateFood() a gyümölcs véletlenszerű generálását végzi egy nagyon egyszerű ellenőrzéssel. 

Abban az esetben, ha a gyümölcs a (-1,-1) -es pozícióban van, az azt jelenti, hogy a kígyó előzőleg „megette”, ezért a pályán kívül jött létre. Ha a kígyó hossza eléri a kívánt hosszúságot, akkor nem kell újra generálni, hiszen nyertünk, a win változó igaz értéket vesz fel. Ha ezt a számot még nem érte el a kígyó hossza, akkor a gyümölcsöt egy új, véletlen helyen kell létrehozni.

A scanJoystick() függvény a kígyó sebességétől mérten olvassa be a joystick pozícióját az analóg inputról. Működése nagyon egyszerű, amennyiben adott irányban átlépi a threshold értéket a joystick pozíciója, a négy irány közül felveszi a megfelelőt.



Egy egyszerű ellenőrzéssel kezelésre kerül, hogy a kígyó ne tudjon 180°-os fordulatot tenni, hiszen az nem lenne a működésnek megfelelő. 

A snake útvonalának kiszámításához és megjelenítéséhez szükség van egy fal detektáló függvényre, aminek a lényege, hogy ha falhoz érne a kígyó, akkor a szemközti faltól folytassa az útját. 

A kígyó mozgásának érdemi részét a calculateSnake() metódus végzi. Először a snakeDirection változó alapján egy switch-case vezérlési szerkezet megállapítja, hogy melyik irányba kell növelni a kígyó hosszát. Mivel a LED mátrix sorai, illetve oszlopai balról jobbra, valamint fentről lefele növekednek, ha a kígyót feljebb szeretnénk vinni, akkor a sorainak számát csökkentenünk kell. Amennyiben falhoz érne, a wallDetection() átviszi az ellentétes oldalra, a setLed() pedig bekapcsolja az adott LED-et. A program hasonlóképpen működik a négy irányra.



Ezek után az objektumokkal való érintkezés van lekezelve. Amennyiben a kígyó egy olyan szegmensre lépne, ahol a kígyó szerepel, akkor a játék véget ért.A másik eshetőség, hogy nem saját magával, hanem a célponttal, egy gyümölccsel kerül egy cellába. Ebben az esetben, a gyümölcsöt a pályán kívülre kell elhelyezni, hiszen ez alapján fogja tudni a korábban megírt generateFood(), hogy egy új helyre kell elhelyezni a gyümölcsöt. Ezen kívül a kígyó hosszát tároló snakeLength változó értékét meg kell növelni. 

Azoknak a mezőknek az értéke, amik nullánál nagyobbak voltak, azaz volt rajta kígyó, megnövelődnek eggyel. Erre azért lesz szükség, mert az ezt követőben szekcióban, ahhoz, hogy ne egy végtelenbe nyúló vonal legyen, a végéről el kell venni egyet. Ez a következő kódrészletben fog megvalósulni. 

A loop() függvény legutoljára a setState() -et hívja, ami nemes egyszerűséggel a győzelmi, valamint a vereség állapotok között vált.

Dino run

A harmadik játéknak szerettem volna egy olyat megvalósítani, ami széles körben elterjedt és biztosan sokan találkoztak már az ötletadójával. A Dinosaur Game egy beépített böngészős játék a Google Chrome böngészőben. A játékos egy pixelezett Tyrannosaurus rexet vezet egy oldalra gördülő tájon, elkerülve az akadályokat a magasabb pontszám elérése érdekében. [18]

A dolgozat kereteibe nem fért bele, hogy egy teljesen saját játékot írjak a kijelzőre, ezért egy, az Arduino kezdőprojektek között igen népszerű játékot fejlesztettem tovább, ami sok felhasználó által már meg lett valósítva. [19]

A továbbfejlesztés célja az volt, hogy a szenzorok működését is belehessen vonni a bemutató projektbe. Így a sokszor felhasznált gombokkal való irányítás helyett egy PIR szenzort felhasználva hoztam mozgásba a karaktert.

PIR 4 másodpercenként tud érzékelni.

//TODO fejlesztés leírása

# Hardveres implementáció – 3D tervezés

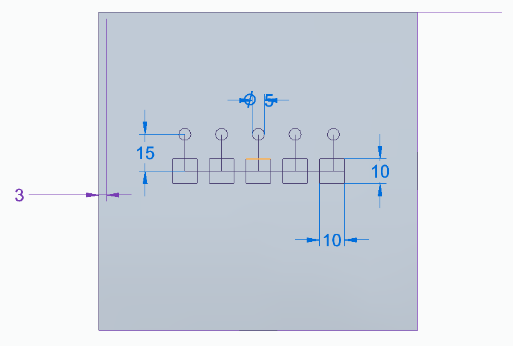
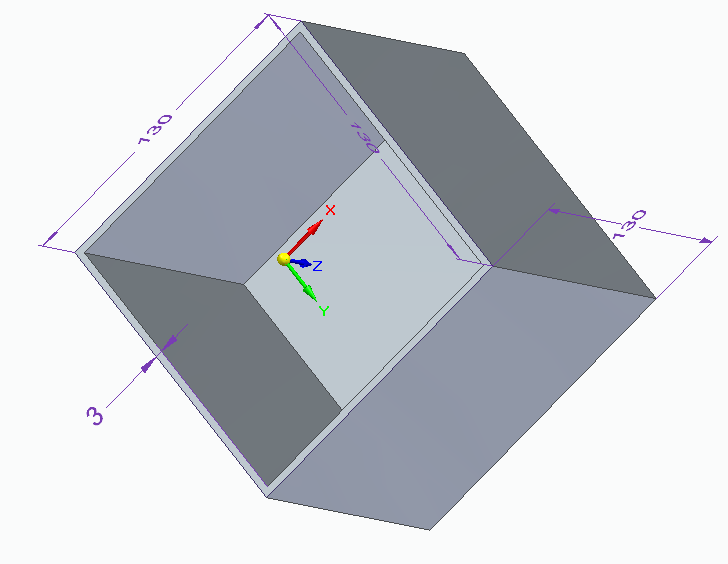
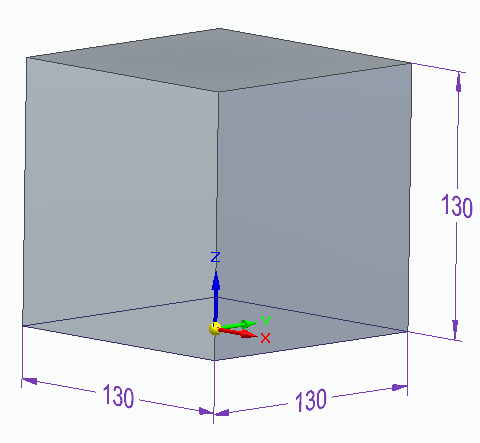
A kapcsolások megtervezésének és programok megírásának elkészültével a tényleges, fizikai összeszerelés maradt hátra. Mindenképpen olyan formában szerettem volna megvalósítani a tárolókat, ami az otthoni barkácsolástól, fából összerakott elemektől eltér. Ésszerű választásként a 3D nyomtatást választottam, amire a tanszéken is van lehetőség, így a vállalkozó szellemű diákok kinyomtathatják megtervezett modelljeiket. Nekem ez nem volt opció, ugyanis a tervezett modell túl széles lett, így egy külső cégnél végeztem el a nyomtatást.

Tervezés

A tervezést a Siemens Solid Edge szoftverével valósítottam meg, melynek diákok számára ingyenes verziója segíti a kreatív munkát.

Mivel a projekt célja, hogy bemutató alkalmakon a fiatalok számára betekintést nyújtson az Arduino világába, ezért fontos volt, hogy lehetőség szerint, minél több elem moduláris, könnyen cserélhető, akár bővíthető legyen. A három játék tárolója egy-egy, minden irányba tizenhárom centiméter széles kocka, melynek alja teljesen nyitott, a tetején pedig a bemenetként, kimenetként szolgáló eszközök kivágásai találhatók, illetve közös nevező mindhárom kocka esetén a kettő sarokokban megtalálható kivágás, ezeknek a funkciója, hogy segítségükkel a kockák felölről kivehetőek lesznek.

Egy példán keresztül szeretném bemutatni nagyvonalakban, hogyan is készültek ezek a stílusú dobozok. Első lépésként egy 130x130x130mm nagyságú kockát kell felvenni. A következő lépésben, egy „Thin Wall” nevű eszközt kell használni, ami amennyiben „Open-Faces” beállításban van, akkor adott falvastagságú, egyik oldalán nyitott, üreges kockát csinál a tömör tömbből. Itt jön az izgalmasabb rész, „Sketch” módban a felső panelen meg kell tervezni a kivágásokat. Segédvonalakat használva fel kell venni a megfelelő alakú és méretű kivágásokat. Ehhez a korábbiakban tolómérővel lemértem a gombok, valamint LED-ek méretét is, viszont minden esetben -ahol ilyenre van lehetőség- hagytam helyet az esetleges hibáknak. Itt azonban érdemes azt is észben tartani, hogy a műanyag nyomaton a kivágásokat utólag bármikor könnyen lehet tágítani, azonban szűkítésre nincs lehetőség.

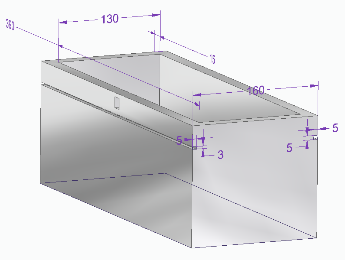
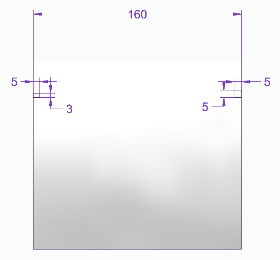
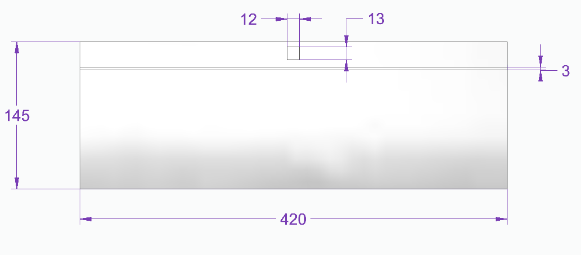


A képen szöveg, névjegykártya látható

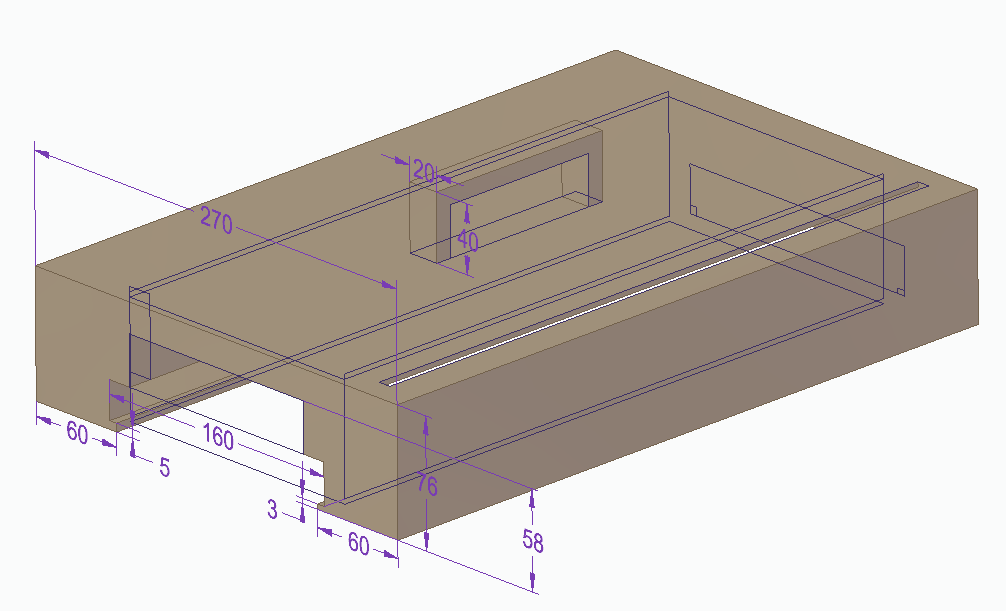
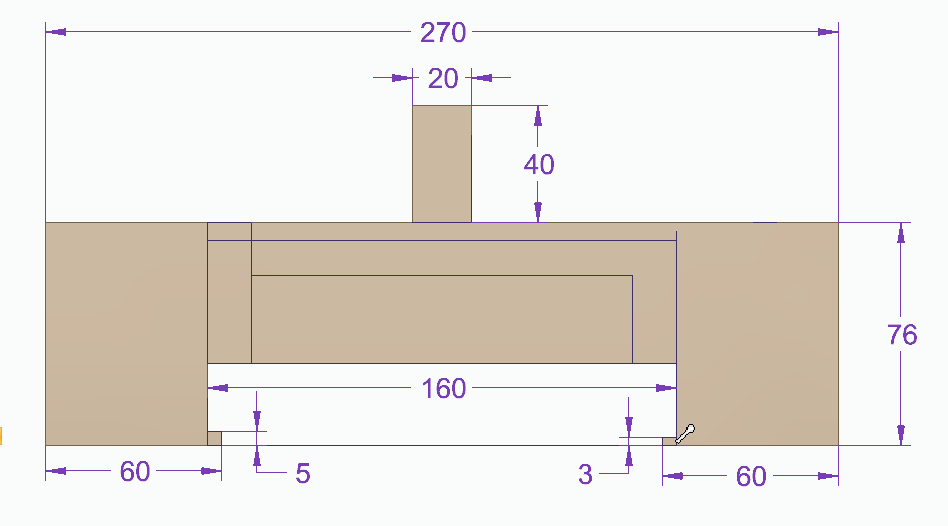
Automatikusan generált leírásLegutolsó lépésként az előzőleg megrajzolt sablon alapján a „Cut” eszköz segítségével kivághatók a megfelelő formák. Ezzel elkészült a memóriajáték tartója.

Bonyolultabb tervezést igényelt a látszólag egyszerű platform, valamint tető, ami körül veszi a játékokat és helyet adnak az RFID-val működtetett szolenoid zárnak.

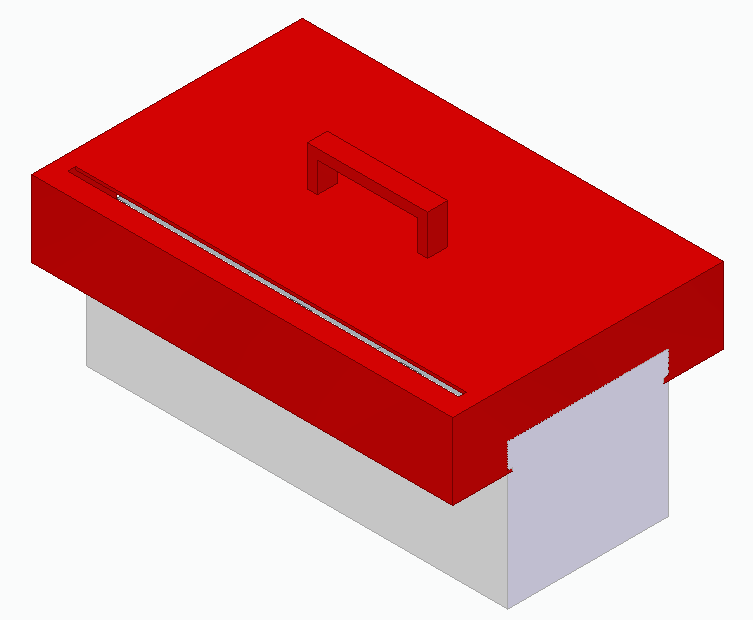
Az alap nincs megbonyolítva, egy 15mm vastag oldalfalú, a tetején nyitott téglatest, amin található egy bemélyedés, valamint a két oldalán egy-egy különböző méretű, sínként szolgáló kivágás. A méretkülönbség a stabilitás, valamint annak érdekében került megvalósításra, hogy biztosan jó oldalon legyen a zár nyelve.



A tető tervezésénél az okozta a problémát, hogy a játékok nem egysíkban helyezkednek el, vannak kilógó elemek, LED-ek, potenciométerek, gombok. Ezért egy olyan tervet kellett készíteni, amivel a vizuális megjelenés sem romlik túl sokat és a funkcióját is betölti. Ezért egy rácsúsztatós megoldásra esett a végső választás. Ez egy szélességében megnyújtott téglatest, aminek a kilógó részeiben helyezkedhet el a zár, valamint az áramkör. Egyik oldalán lévő kivágásnak köszönhetően akadás mentesen rácsúsztatható a tető a platform elemre. Ez a kivágás hibakezelési szempontból is hasznos lehet, hiszen csukott állapotban is rá lehet látni valamilyen szinten a zárszerkezetre.



Az egyes részegységek megtervezése után természetesen egymásba helyeztem a megtervezett modelleket, hogy megfelelőek-e az illesztések. Itt egészen sok apró hiba kijött, amit szerencsére percek alatt lehetett módosítani. Az első ilyen egymásba helyezés alkalmával tűnt fel, hogy ez az egész eszköz nagyon hasonlít egyben egy gombára, ami nagyon megtetszett és gondoltam beleviszem a dizájnba, ezért módosítottam egy picit a méreteken, hogy még inkább „gombaszerű” legyen, valamint a színeket is úgy válogattam össze, hogy erre hajazzon.



A képen szöveg, névjegykártya látható

Automatikusan generált leírásEzeken a nagyobb modelleken kívül készítettem még néhány tartó elemet, amik segítségével alulról támogathatók az Arduinohoz használt eszközök. Ezek igencsak egy sémára mennek. Van egy platform, amin az eszköz elhelyezkedik, valamint kettő, vagy három szárnya, ami a rögzítésre szolgál.

Nyomtatás

Összeszerelés

# Tesztelés

# Összefoglalás

# Hivatkozások

|  |  |
| --- | --- |
| [1] | A. AG, „Arduino - Introduction,” [Online]. Available: https://www.arduino.cc/en/guide/introduction#. [Hozzáférés dátuma: 03. október 2021.]. |
| [2] | „PIR Motion Sensor: How to Use PIRs w/ Arduino & Raspberry Pi,” [Online]. Available: https://create.arduino.cc/projecthub/electropeak/pir-motion-sensor-how-to-use-pirs-w-arduino-raspberry-pi-18d7fa. [Hozzáférés dátuma: 03. október 2021.]. |
| [3] | N. Miller, „How LCD Displays Work,” [Online]. Available: https://www.nelson-miller.com/lcd-displays-work/. |
| [4] | D. Workshop, „Using LCD Displays with Arduino,” [Online]. Available: https://dronebotworkshop.com/lcd-displays-arduino/. |
| [5] | „MAX7219/MAX7221 Serially Interfaced, 8-Digit LED Display Drivers,” Maxim Integrated Products Inc, [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf. |
| [6] | P. Scott, „What Is a Solenoid Door Lock?,” InfoBloom, [Online]. Available: https://www.infobloom.com/what-is-a-solenoid-door-lock.htm. |
| [7] | HMika, „vilaglex: szolenoid,” [Online]. Available: http://www.vilaglex.hu/Lexikon/Html/Szolenoi\_.htm. |
| [8] | L. M. Engineers, „InDepth: What is RFID? How It Works? Interface RC522 RFID Module with Arduino,” [Online]. Available: https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/. |
| [9] | HELLOTECHIE, „Make Your Own Fritzing Parts,” [Online]. Available: https://learn.sparkfun.com/tutorials/make-your-own-fritzing-parts. |
| [10] | Thomas\_W59, „MAX7219 Dot Matrix Led Modul part submit,” [Online]. Available: https://forum.fritzing.org/t/max7219-dot-matrix-led-modul/1914. |
| [11] | RafaGS, „GitHub: Fritzing/HC-SR501 Body Sensor Module.fzpz at master,” [Online]. Available: https://github.com/RafaGS/Fritzing/blob/master/HC-SR501%20Body%20Sensor%20Module.fzpz. |
| [12] | ArduinoModules, „KY-019 5V RELAY MODULE ZIP FILE - ArduinoModulesInfo,” [Online]. Available: https://arduinomodules.info/download/ky-019-5v-relay-module-zip-file/. |
| [13] | D. John, „GitHub: Fritzing-Library/Plastic Solenoid Valve.fzpz at master,” [Online]. Available: https://github.com/adafruit/Fritzing-Library/blob/master/parts/Plastic%20Solenoid%20Valve.fzpz. |
| [14] | vanepp, „Special TYPE of SPST (part) - parts help - fritzing forum,” [Online]. Available: https://forum.fritzing.org/t/special-type-of-spst-part/6125/2. |
| [15] | „ClickCharts Diagram & Flowchart Software,” NCH Software, [Online]. Available: https://www.nchsoftware.com/chart/index.html. |
| [16] | „randomSeed() - Arduino Reference,” Arduino, [Online]. Available: https://www.arduino.cc/reference/en/language/functions/random-numbers/randomseed/. |
| [17] | „millis() - Arduino Reference,” Arduino, [Online]. Available: https://www.arduino.cc/reference/en/language/functions/time/millis/. |
| [18] | „Dinosaur Game - Wikipedia,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Dinosaur\_Game. |
| [19] | M. Magdy, „Arduino Game By LCD - Arduino Project Hub,” [Online]. Available: https://create.arduino.cc/projecthub/muhamd-magdy/arduino-game-by-lcd-9a3bc2. |
| [20] | „Arduino Project Hub,” [Online]. Available: https://create.arduino.cc/projecthub/electropeak/pir-motion-sensor-how-to-use-pirs-w-arduino-raspberry-pi-18d7fa. [Hozzáférés dátuma: 03. október 2021.]. |
| [21] | „HC-SR501 PIR Motion Detection Sensor,” Electropeak, [Online]. Available: https://electropeak.com/pir-motion-sensor. [Hozzáférés dátuma: 03. október 2021.]. |
| [22] | „Adapting PIR sensor technology to new applications,” [Online]. Available: https://www.avnet.com/wps/wcm/connect/onesite/47d7ffe0-ad6f-4b5d-a772-1a4a06502988/PIR-Sensor-Operation-EN-Image.jpg?MOD=AJPERES&CACHEID=ROOTWORKSPACE.Z18\_NA5A1I41L0ICD0ABNDMDDG0000-47d7ffe0-ad6f-4b5d-a772-1a4a06502988-m5WnanK. |
| [23] | LignUp, „An explanation of LCD Panel Technology,” [Online]. Available: https://lignup.com/computer-tips/82-explanation-lcd-technology-.html. |
| [24] | D. Nedelkovski, „8×8 LED Matrix MAX7219 Tutorial with Scrolling Text & Android Control via Bluetooth,” How To Mechatronics , [Online]. Available: https://howtomechatronics.com/tutorials/arduino/8x8-led-matrix-max7219-tutorial-scrolling-text-android-control-via-bluetooth/. |

# Nyilatkozat

# Függelék