„Raspberry Kreatív: Te és a Raspberry-mit tudtok kihozni egymásból?”

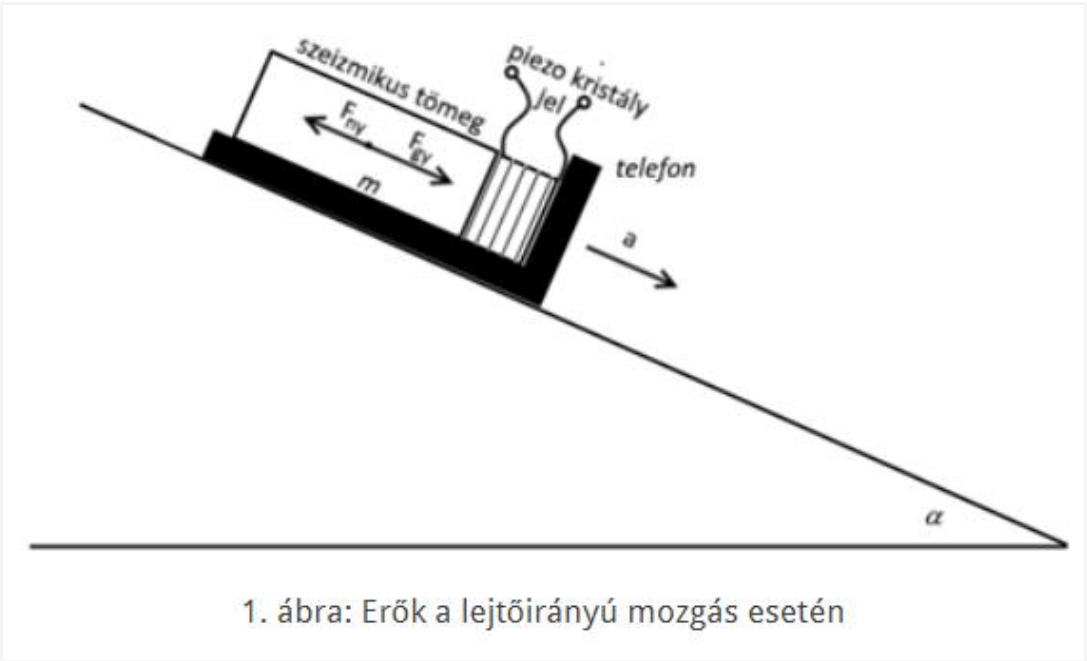
GyroPong

Barna Mihály Imre

Tartalomjegyzék

Feladat:
Egy falilabda szerű játék alkalmazás készítése raspberry pi-ra, amelyet a telefon gyorsulásmérő szenzora segítségével lehet irányítani, bluetooth kapcsolat segítségével. A raspberry pi egy konzolként viselkedik, míg a telefon egy app segítségével képes kapcsolatba lépni a konzollal és elküldeni neki a gyorsulásmérő adatait (Mivel a raspberry pi 2-nek nincs beépítette bluetooth-ja, így egy bluetooth dongle biztosítja a konzol számára a bluetooth-ot). A konzol és a telefonos alkalmazás is a GyroPong nevet viseli. A giroszkóp és a gyorsulásmérő hasonlóan működik, csak míg a gyorsulásmérővel az elforgatás irányát és szögét tudjuk megmérni, addig a giroszkóppal a pillanatnyi elmozdulás irányát és mértékét. A projektemnek azért adtam a GyroPong nevet, mert csak utólag derült ki számomra, hogy a gyorsulásmérő és nem a giroszkóp segítségével fogom irányítani a játékot, de ekkor már feltöltöttem a telefonos alkalmazást a Google Play-re GyroPong néven. Igazából a laikusok számára a két szenzor között nem sok a különbség, ezenkívül a giroszkóp segítségével is lehetne irányítani a játékot csak kicsit macerásabb lenne.

Gyorsulásmérő szenzor működése dióhéjban:
A telefonhoz rögzített gyorsulásmérő az 1. ábrán látható. F_{gy} az ábrán a mozgásirányú gyorsító erő, ami lehet a nehézségi erő, vagy a lejtőn a nehézségi erő mozgásirányú komponense. F_{ny} pedig az érzékelő által a tömegre ható nyomóerő. Ennek az ellenerejét méri az érzékelő.



1. ábra: Erők a lejtőirányú mozgás esetén

Vizsgáljuk meg dinamikai szempontból néhány példát

1) A telefon fekszik az asztalon, szenzorba épített mérőtömeg tömege m gyorsító erő a nehézségi erő.

$$m \cdot g - F_{ny} = m \cdot a$$

$$\frac{F_{ny}}{m} = g - a = 9{,}81 - 0 = 9{,}81$$

A várható érték tehát 9,81, ami összhangban van a méréssel.

2) A telefon szabadon esik

$$m \cdot g - F_{ny} = m \cdot a$$

$$\frac{F_{ny}}{m} = g - a = 9{,}81 - 9{,}81 = 0$$

A várható érték ebben az esetben 0, ami szintén összhangban van a mért értékkel.

3) A telefon egy alfa hajlásszögű lejtőn áll.

$$m \cdot g \cdot \sin(\alpha) - F_{ny} = m \cdot a$$

$$\frac{F_{ny}}{m} = g \cdot \sin(\alpha) - a = 9{,}81 \cdot \sin(\alpha) - 0 = 9{,}81 \cdot \sin(\alpha)$$

A várható érték 9,81·sin(α). Ez az érték is megegyezik a mért értékekkel

A projektről:
A projekt során felhasznált eszközök, program nyelvek stb.:

- Raspberry pi 2
- Bluetooth dongle (otthon találtam, ezer éves szerintem, de működik 😊)
- MIT App Inventor webes alkalmazás a telefonos program elkészítéséhez
- C++ programozási nyelv és az SFML library
- Egy tucat shell script
- rc.local a rendszerrel együtt induló komponensek kezeléséhez
- /home/pi/.config/autostart/start.desktop, amely a rendszer és az X11 indulása után indít programokat (grafikus megjelenéshez kell)
- Néhány config file átírása
- Splash image kicserélése

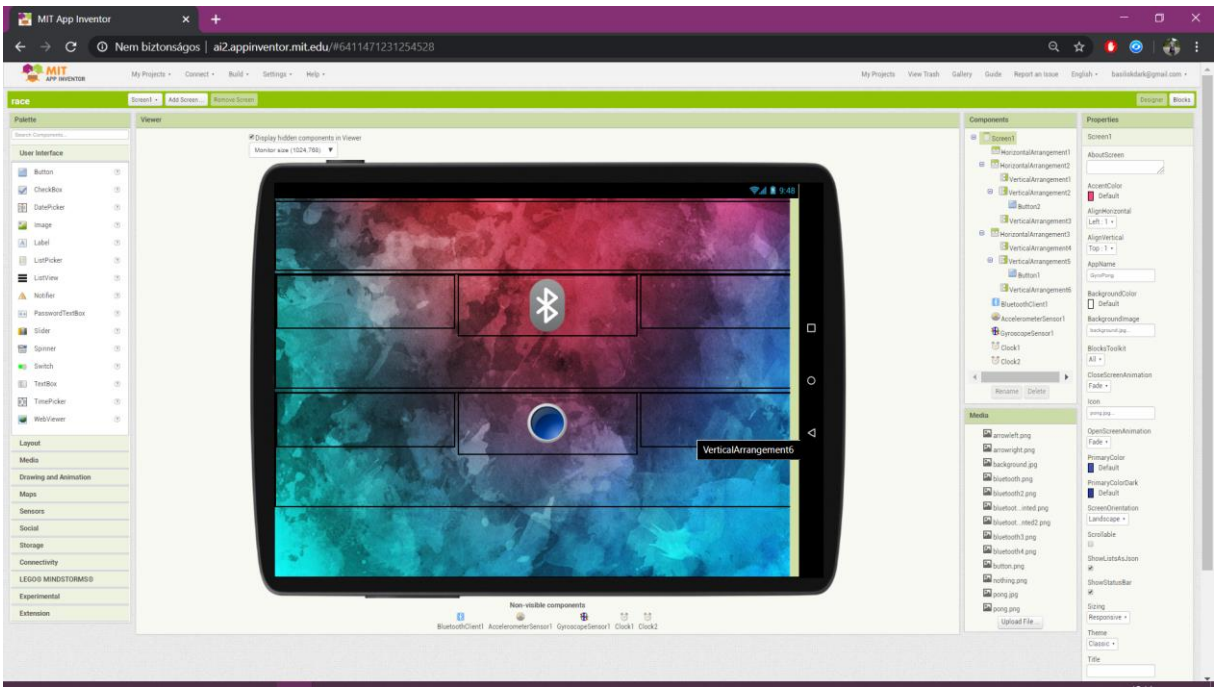
GyroPong telefonos alkalmazás:

Mint említettem, ezt az alkalmazást a MIT App Inventor segítségével készítettem.

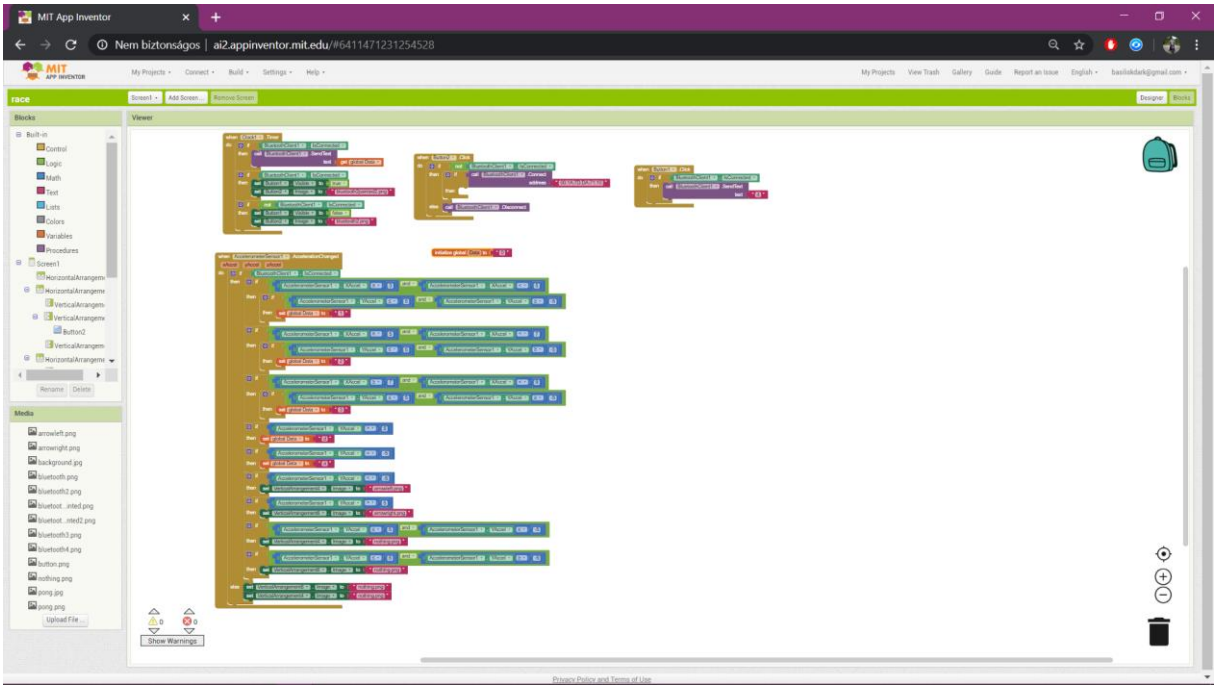
A program két részből épül fel:

- Design: Itt lehet beállítani az alkalmazás kinézetét.
- Blocks: A program blokkokból épül fel, amelyeket egymásba/egymásra tudunk építeni.

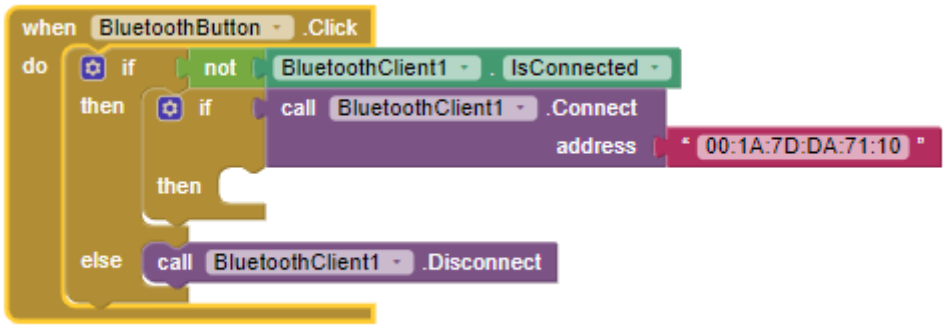
Design:



Blocks:


MIT App Inventor
Nem biztonságos | ai2.appinventor.mit.edu/#6411471231254528
race
Blocks
Viewer
Media
Show Warnings

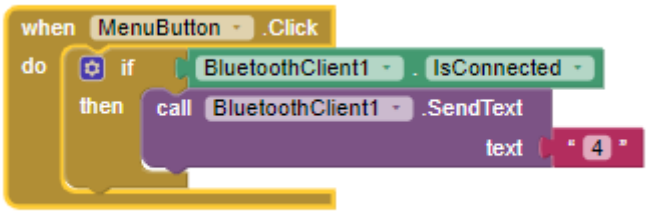
A Blocks részei:



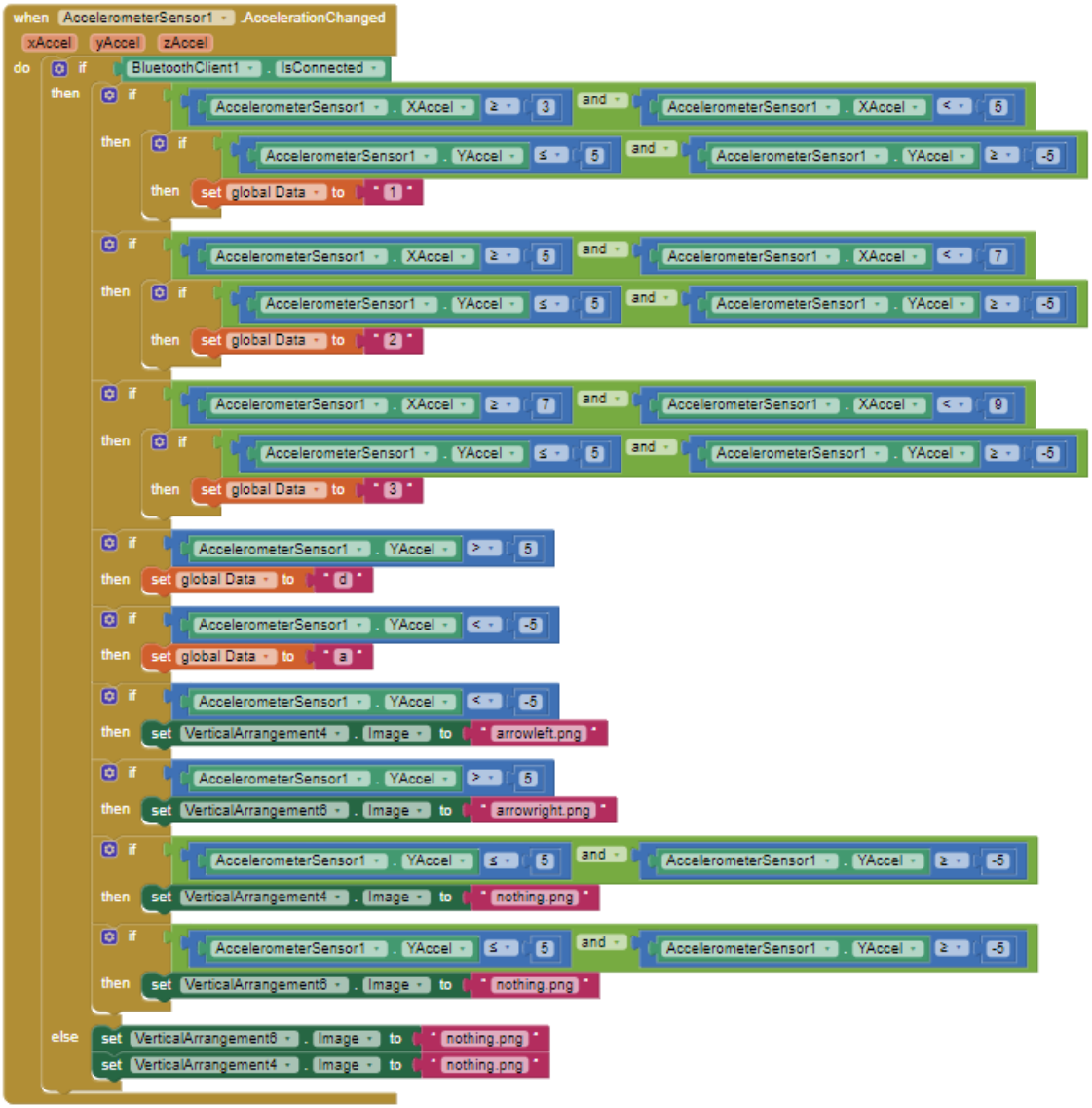
Ez a rész akkor aktiválódik, ha rákoppintunk a bluetooth ikonra.

Működése:
Ha még nincs csatlakozva a telefon a bluetooth dongle-hez, akkor csatlakozik hozzá (Itt a 00:1A:7D:DA:71:10 a bluetooth dongle címe), ha már csatlakoztattuk az eszközt, akkor lecsatlakozik a bluetooth-ról.

Ez egy változó deklarálása/definiálása amiben a program letárolja majd a gyorsulásmérő szenzor által adott pillanatnyi egész szám tipusú értéket.

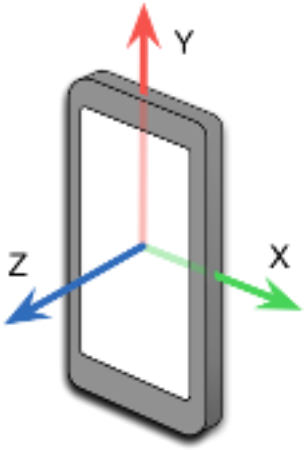


Ez a rész akkor aktiválódik ha rákoppintunk a kék gombra és a telefon már csatlakozott a bluetooth dongle-hez. Ha rákoppintunk elküldi a 4-es ASCII karaktert a raspberry pi-nak (Nem a négyes számút, hanem magát a 4-et mint karaktert, száma: 52).

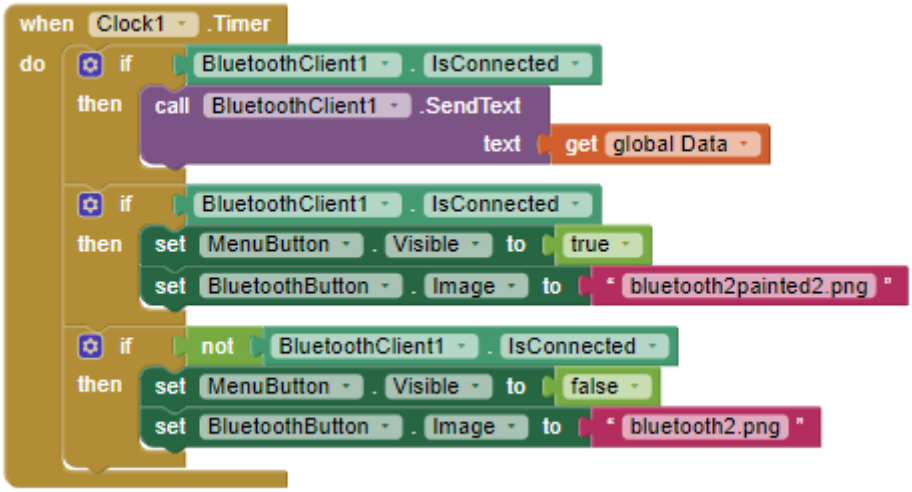


Itt történik a gyorsulásmérő pillanatnyi értékének a változóba történő írása. Ezenkívül ha az x tengely körül forgatjuk a telefont, megjelenik egy nyíl a forgatás irányával.

A gyorsulásmérő a képen látható tengelyek körüli forgatást méri:

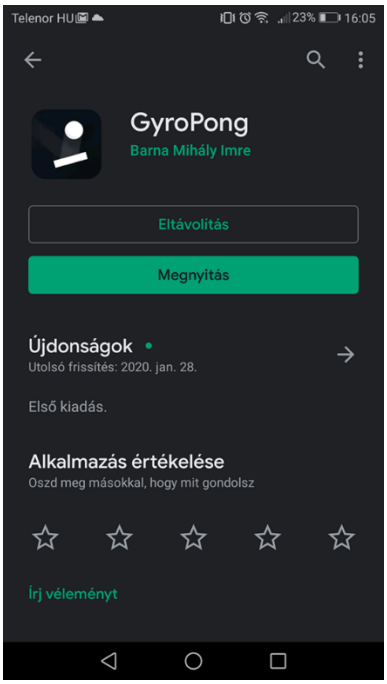


Az y tengely körüli forgatással lehet a konzolon lévő játék menüpontja közül választani (Plusz a kék gomb megnyomása), míg az y tengely körüli forgatással a játékban lévő ütőt tudjuk irányítani (lásd később).



Ez a rész a gyorsulásmérő által a változóban tárolt érték raspberry pi-hoz történő elküldéséért felelős. Ezenkívül ha a telefon csatlakoztatva van/nincs csatlakoztatva a készülékhez akkor különböző logók jelennek meg a felhasználó tájékoztatására. Ha az eszköz csatlakoztatva van akkor megjelenik a kék gomb a képernyőn, amivel ki tudjuk választani a játék valamelyik menüpontját (lásd később).
Az adatot 100 ms-onként küldi a telefon (Clock), így nem terheli le a sok adat fogadása a raspberry pi-t (100 ms a másodperc 10-ed része).

Az általam készített alkalmazás elérhető a Google Play-en is.



Shell scriptek a raspberry pi-on:

- bluetooth.sh
- bluetooth0write.sh
- bluetooth1write.sh
- chmod.sh
- exit.sh
- rfcomm0.sh
- rfcomm1.sh
- start.sh
- bgkill.sh

Az általam készített shell scriptek nagy részben parancsok listái, de lehet bennük vezérlési szerkezeteket (elágazás, ciklus stb..) is létrehozni, tehát programnyelv is. Az első sorokban szereplő #!/bin/bash azért felelős, hogy tudassuk a shell-el, hogy a bash dolgozza fel a scriptben lévő sorokat. Minden parancsot admin-ként futtattam a biztonság kedvéért (sudo).

bluetooth.sh:

```
#!/bin/bash
sudo hciconfig hci0 piscan
sudo sdptool add SP
echo -e "discoverable on" | bluetoothctl
```

Ez a script felelős a bluetooth által fogadott adatok emulált soros portra küldésének biztosításához.

sudo hciconfig hci0 piscan (page and inquiry scan):
Az egyik legáltalánosabb feladat a Bluetooth eszközök esetében a közelben levő további eszközök felderítése. Ezt a műveletet *tudakozódásnak* ("inquiry") nevezik.

sudo sdptool add SP:
A Service Discovery Protocol (SDP) segítségével a kliens alkalmazások képesek felderíteni, hogy a szerver alkalmazások részéről milyen szolgáltatások érhetőek el, valamint ezek a szolgáltatások milyen tulajdonságokkal rendelkeznek. A szolgáltatások tulajdonsági közé soroljuk többek között a felajánlott szolgáltatás típusát vagy osztályát, illetve a szolgáltatás kihasználásához szükséges mechanizmusra vagy protokollra vonatkozó információkat.
Ez a parancs a soros portot mint szolgáltatást adja hozzá a bluetoot-hoz.

echo -e ”discoverable on” | bluetoothctl:
Ez a parancs felfedezhetővé teszi a raspberry pi bluetooth-ját, amit GyroPongnak neveztem el.

bluetooth0write.sh és bluetooth1write.sh:

```
#!/bin/bash
while :
do
if test -e /dev/rfcomm0
then
sudo cat /dev/rfcomm0 > bluetoothdata1.txt
fi
done
```

```
#!/bin/bash
while :
do
if test -e /dev/rfcomm1
then
sudo cat /dev/rfcomm1 > bluetoothdata2.txt
fi
done
```

Ezek a scriptek felelősek a bluetooth-on érkező adatok fájlba írására. Ezen fájlok kiolvasásával lehet majd irányítani a konzolon lévő játékot. Azért írtam két scriptet, mert van lehetőség kétszemélyes módra is (Sajnos csak raspberry pi 3 és afeletti verziókban, mert raspberry pi 2-ben az általam használt bluetooth dongle nem támogat csak egy csatornát, míg a raspberry pi 3 beépített bluetooth-ja több csatornát is támogat).

chmod.sh:

```
#!/bin/bash
if test -e /dev/rfcomm0
then
sudo chmod a+rwx /dev/rfcomm0
fi
if test -e /dev/rfcomm1
then
sudo chmod a+rwx /dev/rfcomm1
fi
```

Ez a script az összes felhasználó számára engedélyt ad, hogy írni, olvasni, futtatni tudják a /dev/rfcomm0 és a /dev/rfcomm1 fájlokat.

rfcomm0.sh és rfcomm1.sh:

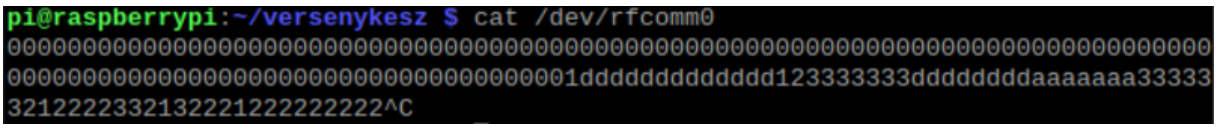
```
#!/bin/bash
sudo rfcomm listen /dev/rfcomm0
```

```
#!/bin/bash
sudo rfcomm listen /dev/rfcomm1
```

Az RFCOMM protokoll a soros portok emulációját valósítja meg az L2CAP protokollon keresztül. A protokoll az ETSI TS 07.10. RFCOMM szabványán alapszik, és egy egyszerű átviteli protokoll, amelyet a 9 tűs RS-232 (EIATIA-232-

E) soros portok emulációjára készítettek fel. Az RFCOMM protokoll legfeljebb 60 kapcsolat (RFCOMM csatorna) párhuzamos használatát támogatja két Bluetooth eszköz között.

Az RFCOMM számára a teljes kommunikációs útvonal két különböző eszközön futó alkalmazást (kommunikációs végpontot) és köztük levő kommunikációs szegments foglalja magában. Az RFCOMM az adott eszközön a soros portot használó alkalmazások részére készült. A kommunikációs szegmens az egyik eszköztől a másikig vezető Bluetooth alapú összeköttetés (közvetlen kapcsolat).



start.sh:

```
#!/bin/bash
cd /home/pi/versenykesz && ./GyroPong
```

Ez a script indítja el a játékot.

exit.sh:

```
#!/bin/bash
sudo shutdown -h now
```

Ez a script kikapcsolja a konzolt.

bgkill.sh:

```
#!/bin/bash
sudo killall bluetooth0writ
sudo killall bluetooth1writ
sudo killall rfcomm
sudo killall rfcomm0
sudo killall rfcomm1
rm bluetoothdata1.txt
rm bluetoothdata2.txt
```

Ezt a scriptet csak teszt jelleggel hoztam létre. Amikor kilépek a játékból akkor SIGKILL signalt küld a megadott processzeknek, amelyek ettől leállnak.

rc.local és /home/pi/.config/autostart/start.desktop:

- *rc.local:*

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
#Start bluetooth service
sudo /bin/bash /home/pi/versenykesz/bluetooth.sh
exit 0
```

Azok a parancsok amelyek az rc.local-ban szerepelnek, a rendszer bootolásával együtt indulnak.

- */home/pi/*.config/autostart/start.desktop:

```
[Desktop Entry]
Type=Application
Name=Start
Exec=/bin/bash /home/pi/versenykesz/start.sh
```

Az ebben lévő parancs, akkor indul el, amikor már betöltődött az X11, ami a grafikus elemekért felelős. Ha ez előtt indítjuk el a programot akkor elindul, de a játék nem látszik.

C++ és SFML library:

Az SFML (Simple Fast Media Library) egy függvénykönyvtár, amely C++-ban írodott és C++-ban használatos. Az SFML a grafikai elemekért felelős a konzolon lévő alkalmazásban (Ablak, objektumok megjelenítése, képfrissítés stb..).

Szükséges includeok:

```
#include <SFML/Graphics.hpp>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<unistd.h>
#include<iostream>
#include<string>
#include<fstream>
#include<sstream>
#include<chrono>
#include "defines.h"
#include "functions.h"
```

C++ programozási nyelven írt fájlok:

- main.cpp
- defines.h
- functions.h

main.cpp:
Ez maga a főprogram.

defines.h:
Ebben találhatók a menüpontok makrói, az átláthatóság érdekében.

functions.h:
Ebben találhatóak a főprogram által használt függvények.

Makrók:

#define OneTwo 0
#define MainMenu 1
#define OPGame 2
#define GameOverMenu 3
#define TPGame 4

A függvények a következők:

- *bool is_file_exist(const char *fileName):* Ellenőrzi, hogy létezik -e a fájl (/dev/rfcomm0 és /dev/rfcomm1).
- *void NotConnect(sf::Clock Clock,bool& Once,sf::RenderWindow& windows,sf::Text& newplayer,sf::Text& connect,sf::Text& Nav,sf::Sprite&*

bg, sf::Sprite& Tutorial, sf::Sprite& Tutorial2, sf::Sprite& Tutorial3 , sf::Sprite& Accel): Ez jelenik meg, amíg nem csatlakozunk a konzolhoz.

- *void OnePlayerFile (const char *fileName, std::ifstream& OPFile,std::string& Last,std::string& LastButOne, std::string& Ch):* Beolvassa a bluetooth-ról érkező adatokat, egyszemélyes módban.
- *void TwoPlayerFile (const char *fileName, std::ifstream& TPFile,std::string& Last, std::string& Ch):* Beolvassa az adatokat, kétszemélyes módban.
- *void PersonsState(int& MenuState, sf::Text& Player1, sf::Text& Player2, std::string& Last, std::string& LastButOne, bool& Player2Connect, int& Menu):* Egyszemélyes vagy kétszemélyes mód kiválasztása.
- *void PersonDraw(sf::RenderWindow& windows, sf::Sprite& bg, sf::Text& Player1, sf::Text& Player2):* Ugyanez csak, ennek a megjelenítése a képernyőn.
- *void MainMenuState(int& MenuState, sf::Text& Play, /*sf::Text& Inst,*/ sf::Text& Exit, std::string& Last, std::string& LastButOne, bool& Player2Connect, bool& Edge, float& SpeedX, float& SpeedY, float& DefaultSX, float& DefaultSY, sf::RectangleShape& Rectangle, sf::RectangleShape& Rectangle3, int& Rnd, int& Rnd2, int& Menu):* A Main Menu valamelyik pontjának kiválasztása.
- *void MainMenuDraw(sf::RenderWindow& windows, sf::Sprite& bg, sf::Text& Play, /*sf::Text& Inst,*/ sf::Text& Exit):* Ugyanez megjelenítése.
- *void OPGameMove(sf::CircleShape& Circle, sf::RectangleShape& Rectangle, float& SpeedX, float& SpeedY, float& MulDivSX, bool& Edge, int& Point, std::string& Last, std::ifstream& RecordFileRead, std::ofstream& RecordFileWrite, std::string& Ch, int& Record, int& Menu, sf::Text& Points):* Egyszemélyes módban az ütő és a labda mozgásának koordinációja.
- *void OPGameDraw(sf::RenderWindow& windows, sf::Text& Records, sf::Text& Points, sf::Sprite& bg, sf::Text& RecordText, sf::Text& PointsText, sf::CircleShape& Circle, sf::RectangleShape& Rectangle, sf::RectangleShape& Rectangle2, int& Record, int& Point):* Ugyanez kirajzolása a képernyőre.
- *void GameOverMenuState(int& MenuState2, sf::Text& Restart, sf::Text& Mainmenu, sf::Text& Quit, std::string& Last, std::string& LastButOne, bool& Edge, float& SpeedX, float& SpeedY, float& DefaultSX, float& DefaultSY, sf::RectangleShape& Rectangle, sf::RectangleShape& Rectangle3, bool& Player2Connect, int& Rnd, int& Rnd2, sf::CircleShape&*

Circle, int& Point, int& Menu): Game Over menu valamelyik pontjának kiválasztása.

- *void GameOverMenuDraw(sf::RenderWindow& windows, sf::Sprite& bg, sf::Text& Restart, sf::Text& Mainmenu, sf::Text& Quit):* Ugyanez kirajzolása a képernyőre.
- *void TPGameMove(sf::CircleShape& Circle, float& SpeedX, float& SpeedY, sf::RectangleShape& Rectangle, sf::RectangleShape& Rectangle3, bool& Edge, float& MulDivSX, std::string& Last, std::string& Last2, int& Menu):* Kétszemélyes módban az ütők és a labda mozgásának koordinációja.
- *void TPGameDraw(sf::RenderWindow& windows, sf::Sprite& bg, sf::CircleShape& Circle, sf::RectangleShape& Rectangle, sf::RectangleShape& Rectangle2, sf::RectangleShape& Rectangle3):* Ugyanez kirajzolása a képernyőre.
- *system("./bluetooth0write.sh&") és system("./bluetooth1write.sh&"):* Ezek a scriptek a háttérben futnak (&), hogy ne blokkolják a főprogramot.
- *system("./rfcomm0.sh&") és system("./rfcomm1.sh&"):* Ezek is háttérben futnak.
- *system("./chmod.sh"):* Ez felelős, azért, hogy tudjuk írni, olvasni, futtatni a bluetooth által írt fájlt.
- *system("./bgkill.sh"):* Bezárja a háttérben futó processzeket.

A main.cpp fontosabb részei:

```
system("./bluetooth0write.sh&");

        ////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

          system("./rfcomm0.sh&");

////////////////////////////////////////////////////////////////////////////
////////////////////////////

                sf::Texture texture;
        if (!texture.loadFromFile("background.png"))
        {
                //error
        }
                sf::Sprite background(texture);

        ////////////////////////////////////////////////////////////////////////////
////////////////////////////////

                sf::Texture texture2;
        if (!texture2.loadFromFile("tutorial.png"))
        {
                //error
        }
```

```
        sf::Sprite tutorial(texture2);
        tutorial.setPosition(125.0f,200.0f);

    ////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
            sf::Texture texture3;
    if (!texture3.loadFromFile("tutorial2.png"))
    {
        //error
    }
        sf::Sprite tutorial2(texture3);
        tutorial2.setPosition(125.0f,200.0f);

    ////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
            sf::Texture texture4;
    if (!texture4.loadFromFile("tutorial3.png"))
    {
        //error
    }
        sf::Sprite tutorial3(texture4);
        tutorial3.setPosition(225.0f,0.0f);

    ////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
        sf::Texture texture5;
    if (!texture5.loadFromFile("accelerometer.png"))
    {
```

```
            //error
        }
            sf::Sprite accel(texture5);
            accel.setPosition(275.0f,0.0f);

        ////////////////////////////////////////////////////////////////////////////
////////////////////////////////
            sf::RenderWindow window(sf::VideoMode(640, 480),
"Pong",sf::Style::Fullscreen);
            window.setMouseCursorVisible(false);
   while (window.isOpen())
   {
            sf::Event event;
            while (window.pollEvent(event))
      {
         if (event.type == sf::Event::Closed)
            window.close();
      }
            if(is_file_exist("/dev/rfcomm0")==false) // eszköz nincs
csatlakoztatva vagy chmod még nincs
            {
                  player2connect=false;
                  menu=OneTwo;
                  if(attach==false)
                  {

        NotConnect(clock,once,window,NewPlayer,Connect,Navigate,backgroun
d,tutorial,tutorial2,tutorial3,accel);
```

```
system("./chmod.sh");
sleep(1);
	}
	if(attach==true)
	{
		system("./rfcomm0.sh&");
		attach=false;
	}
}
if(is_file_exist("/dev/rfcomm0")==true)  //játék (eszköz csatlakoztatva van)
{
	if(is_file_exist("/dev/rfcomm1")==true) //2 játékos mód aktiválva
	{
		if(player2mode==0)
		{
		system("./bluetooth1write.sh&");
		player2mode=1;
		}
		TwoPlayerFile("bluetoothdata2.txt",tpfile,last2,ch2);
	//tpfile.open("bluetoothdata2.txt");
	//std::getline(tpfile,ch2);
	//last2=ch2[ch2.length()-1];
	//tpfile.close();
	}
	attach=true;
```

```
OnePlayerFile("bluetoothdata1.txt",opfile,last,lastbutone,ch);
        if(menu==OneTwo) //0
        {

PersonsState(menustate,player1,player2,last,lastbutone,player2connect,
menu);
                PersonDraw(window,background,player1,player2);
        }
        if(menu==MainMenu) //1
        {

MainMenuState(menustate,play,/*inst,*/exit,last,lastbutone,player2con
nect,edge,speedx,speedy,defaultsx,defaultsy,rectangle,rectangle3,rnd,rnd2,me
nu);
                MainMenuDraw(window,background,play,/*inst,*/exit);
        }
        if(menu==OPGame) //2
        {

OPGameMove(shape,rectangle,speedx,speedy,muldivsx,edge,point,last,r
ecordfileread,recordfilewrite,ch2,record,menu,points);

OPGameDraw(window,records,points,background,recordstext,pointstext
,shape,rectangle,rectangle2,record,point);
        }
        if(menu==GameOverMenu) //3
        {

GameOverMenuState(menustate2,restart,mainmenu,quit,last,lastbutone
```

```
,edge,speedx,speedy,defaultsx,defaultsy,rectangle,rectangle3,player2connect,r
nd,rnd2,shape,point,menu);

        GameOverMenuDraw(window,background,restart,mainmenu,quit);
                }
                if(menu==TPGame) //
                {

        TPGameMove(shape,speedx,speedy,rectangle,rectangle3,edge,muldivsx,
last,last2,menu);

        TPGameDraw(window,background,shape,rectangle,rectangle2,rectangle
3);
                }
    }
        if(sf::Keyboard::isKeyPressed(sf::Keyboard::Key::X)) //kilépés
        {
           system("./bgkill.sh");
           window.close();
        }
        }
    return 0;
}
```

Felhasznált források:

- https://stackoverflow.com/
- https://www.sfml-dev.org/
- https://unix.stackexchange.com/
- https://www.raspberrypi.org/forums/