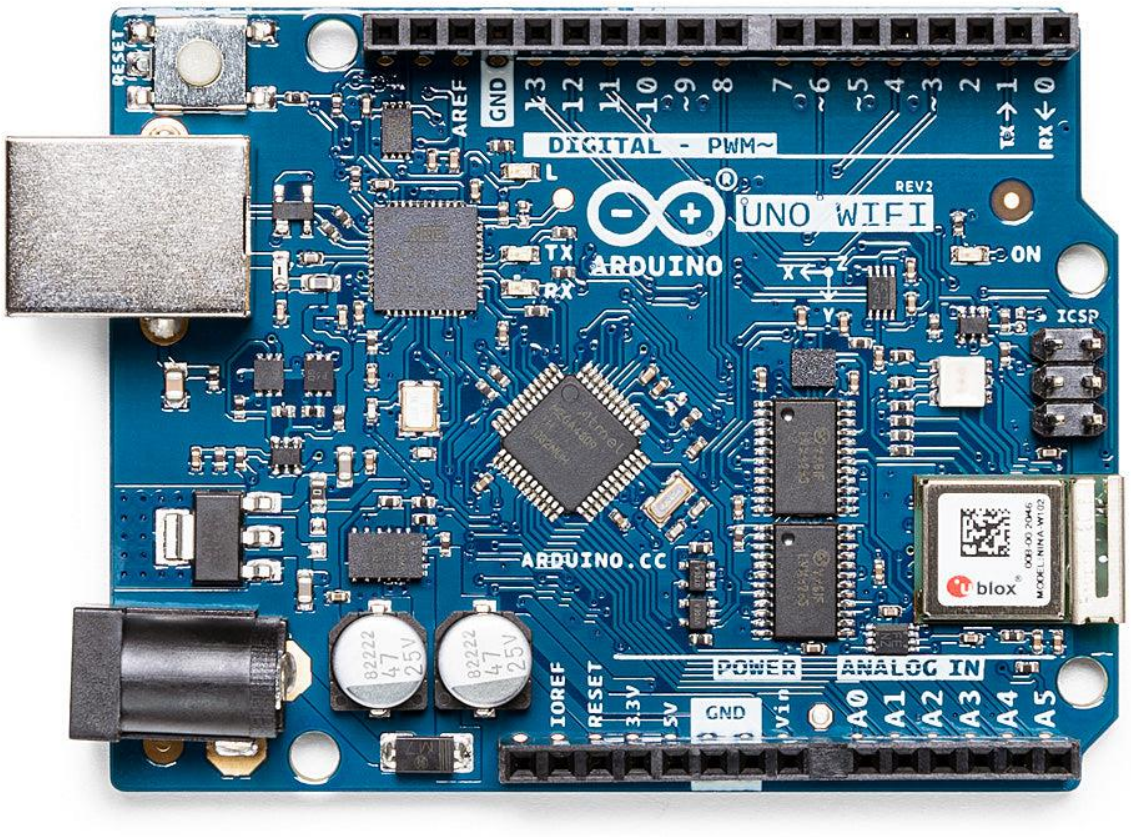
Proiect AD- LaserDodge

Acest proiect reprezintă un joc ce folosește accelerometrul integrat al microcontroller-ului Arduino pentru a controla personajul principal.

Componente:

ARDUINO UNO WiFi REV2

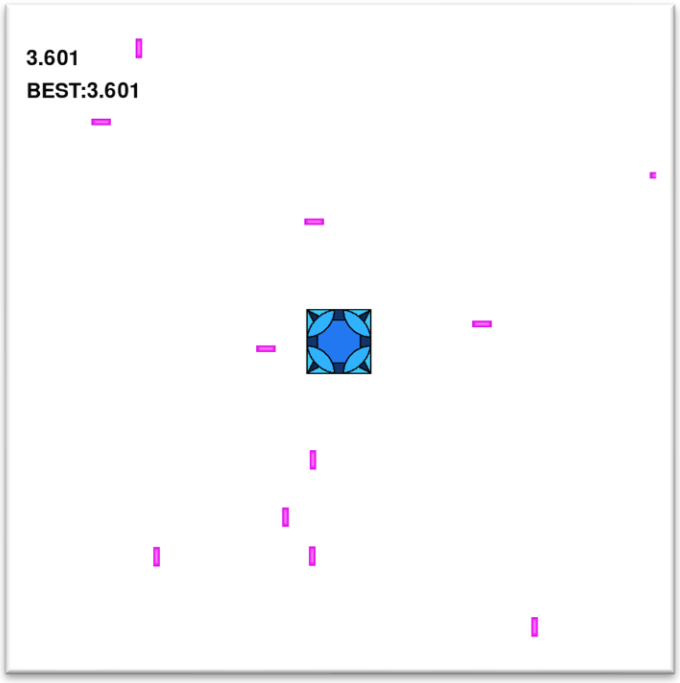
http://store.arduino.cc/products/arduino-uno-wifi-rev2



Cablu cu mufa USB-B

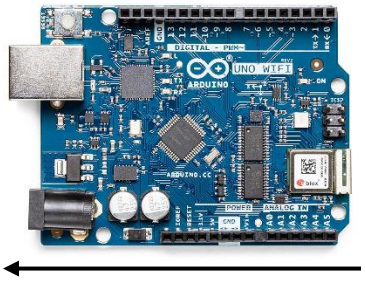
Proiectul folosește senzorul integrat LSM6DS3, ce continue un accelerometru digital si un giroscop digital.

Descriere joc:



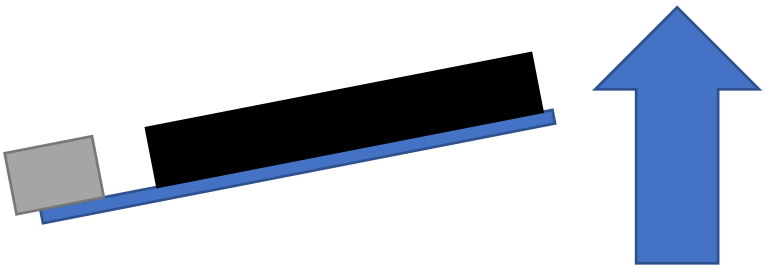
Jocul consta in dirijarea navete, jucătorul fiind nevoit sa evite laserele pentru cat mai mult timp. Scopul jocului este de a supraviețuii cat mai mult.

Dirijarea este realizata prin inclinarea microcontroller-ului.

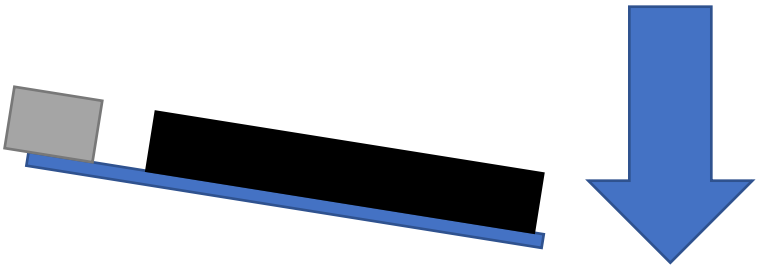


Definim “înainte” drept partea care are mufa USB-B.

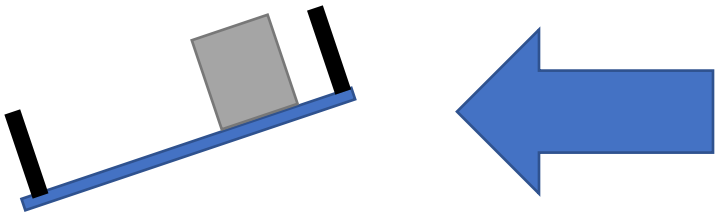
Daca inclinam înspre înainte, naveta accelerează spre partea de sus a ecranului.



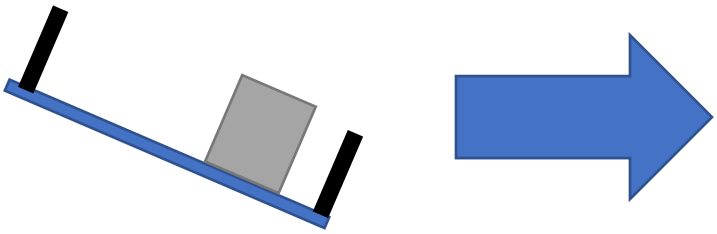
Daca inclinam înspre înapoi, naveta accelerează spre partea de jos a ecranului.



Daca inclinam înspre stânga, naveta accelerează spre stânga ecranului.



Daca inclinam înspre dreapta, naveta accelerează spre dreapta a ecranului.



Forța aplicata navetei este direct proporțională cu sinusul unghiului de inclinare.

Înclinările pot fi compuse pentru a crea orice direcție.

Asupra navetei acționează si o forță de frecare, direct proporțională cu viteza si aplicata in sens opus acesteia.

Daca naveta lovește marginea ecranului cu o viteza, ea va fi reflectata.

Daca naveta este lovita de un laser, atunci ea este distrusa si pe ecran va apărea următorul text.

Jucătorul va avea oportunitatea de a reseta jocul apăsând pe tasta săgeată jos.

In coltul stânga-sus va apărea mereu timpul de la începerea rundei curente. Sub aceasta, apare cea mai mare durata a unei runde.

Descriere cod:

Arduino:

```
include <Arduino_LSM6DS3.h>

void setup() {
 Serial.begin(9600);
 while (!Serial);

 if (!IMU.begin()) {
   Serial.println("Failed to initialize IMU!");

   while (1);
 }




void loop() {
 float x, y, z;
 while(!Serial.available());
 char c = Serial.read();
 if (IMU.accelerationAvailable()) {
   IMU.readAcceleration(x, y, z);
   Serial.print("Acc ");
   Serial.print(x);
   Serial.print(' ');
   Serial.print(y);
   Serial.print(' ');
   Serial.println(z);
 }
 if (IMU.gyroscopeAvailable()) {
   IMU.readGyroscope(x, y, z);
   Serial.print("Gyro ");
   Serial.print(x);
   Serial.print(' ');
   Serial.print(y);
   Serial.print(' ');
   Serial.println(z);
 }
```

Placa Arduino așteaptă un octet de la port, după care scrie valorile primite de la accelerometru si giroscop.

Senzorul LSM6DS3 este accesat folosind biblioteca <Arduino_LSM6DS3.h>.

Functia IMU.begin() initializeaza senzorul.

Functiile IMU.accelerationAvailable() si IMU.gyroscopeAvailable() verifica daca exista valori de citit.

Functiile IMU.readAcceleration() si IMU.readGyroscope() scriu datele in trei variabile de tip float.

Python:

Programul este realizat folosind biblioteca pyGame.

Programul are o serie de parametrii ce influențează rularea jocului:

SCREEN_WIDTH = lățimea ferestrei de joc

SCREEN_HEIGHT = înălțimea ferestrei de joc

SCREEN_COLOR = culoarea fundalului

FRAME_RATE = numărul maxim de cadre pe secunda

PLAYER_SIZE = dimensiunea navetei

ACCEL_FACTOR = numărul cu care sunt înmulțite valorile acceleratiilor primite

FRICTION = coeficientul de frecare

BOUNCE = coeficientul de restituție la lovirea de margine

ENEMY_SIZE = dimensiunea proiectilelor

SPAWN_RATE = numărul de proiectile pe secunda

ENEMY_SPEED_MIN = viteza minima a proiectilelor

ENEMY_SPEED_MAX = viteza maxima a proiectilelor

Pentru fiecare cadru, programul trimite un octet la microcontroller, apoi primește 2 linii cu cate 3 valori: datele de la accelerometru si cele de la giroscop. Dintre acestea, le folosim pe cele de accelerație. Apoi, se actualizează pozițiile navetei si ale laserelor: cele din urma se deplasează in funcție de viteza setata când au fost create:

P_x_nou = P_x_vechi + v_x,

P_y_nou = P_y_vechi + v_y,

iar cea dintâi, după următoarea formula:

V_x_nou = V_x_vechi + (A_x – FRICTION * V_x_vechi)

P_x_nou = P_x_vechi + V_x_nou

V_y_nou = V_y_vechi + (A_y – FRICTION * V_y_vechi)

P_y_nou = P_y_vechi + V_y_nou

Daca exista o coliziune intre naveta si unul dintre lasere, aceasta moare si are loc sfârșitul rundei curente.

Când trebuie creat un laser, acesta apare in afara ferestrei vizibile, având o probabilitate egala de merge fie de sus in jos sau de jos in sus, de la stânga la dreapta sau de la dreapta la stânga.