Autor

Saru Gabriel - Alexandru

- Grupa : 333CB
- Laborant : Alin Rosca

Introducere

Scopul proiectului este realizarea unui joc destul de cunoscut, mai exact jocul Snake, dar modul in care m-am gandit sa realizez implementarea acestuia este una putin mai diferita, m-am gandit la ceva mai interesant, mai exact controlul sarpelui prin miscarea placutei, unde se afla un accelerometrul cu 3 axe, componenta care se ocupa cu monitorizarea majoritatii actiunilor jucatorului cu rolul de a ghida Snake-ul pe unde sa se miste pe display(matrice 8×8).

Jocul se porneste prin intermediul unui buton de START care porneste display-ul, implementat prin intermediul unei matrici 8×8 bi-colora, incepand prin afisarea unui countdown (3 - 2 - 1), apoi se incepe efectiv jocul si trebuie miscata placuta.

Descriere generală

In implementarea sa, Jocul folosește o matrice LED bicoloră simplă 8×8, ca afișaj pentru jocul în sine, și pentru scorul după terminarea jocului. Utilizează un accelerometru cu trei axe, care este atașat la placa de breadboard chiar lângă matricea LED. Accelerometrul este folosit pentru a detecta mișcarea întregii table de joc. Practic, direcția în care se inclina dispozitivul va mișca șarpele în aceeași direcție în care este înclinat. De exemplu, dacă îl înclinați înainte, șarpele se va ridica și așa mai departe.

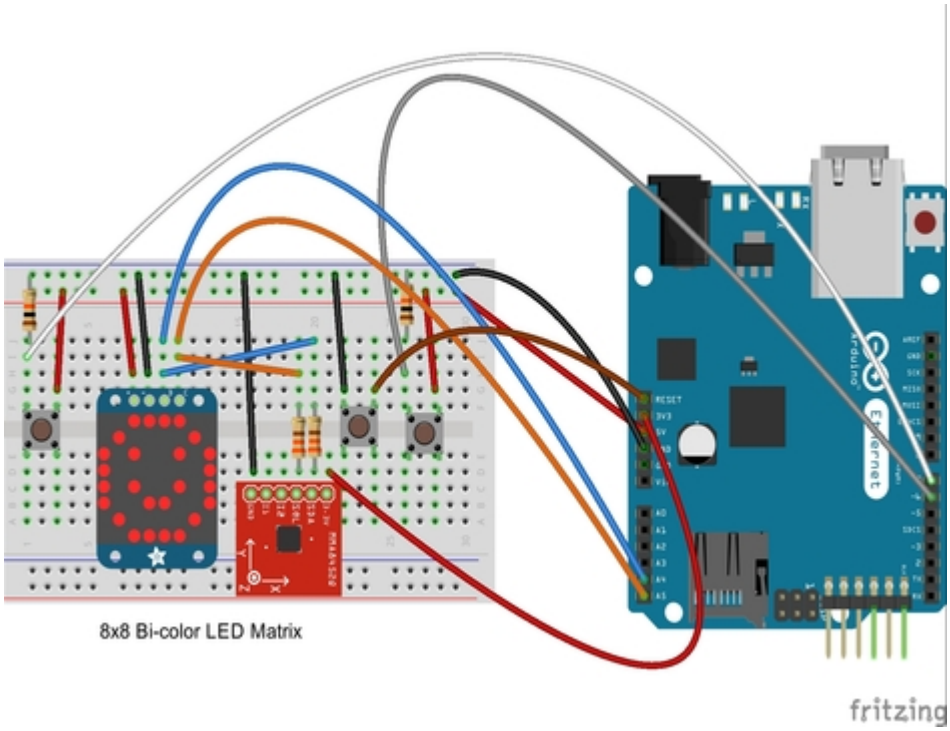
Cum functioneaza ?

Jocul se porneste prin apasarea butonului de start, se afiseaza un countdown de la 3 la 1 si apoi se controleaza sarpele prin simpla miscare a placutei, acesta trebuind sa ajunga la hrana pentru a evolua in marime. Atunci cand se ajunge in situatia in care s-a lovit de perete sau a apucat sa se incurce in dimensiune, jocul se va restarta prin butonul de RESTART.

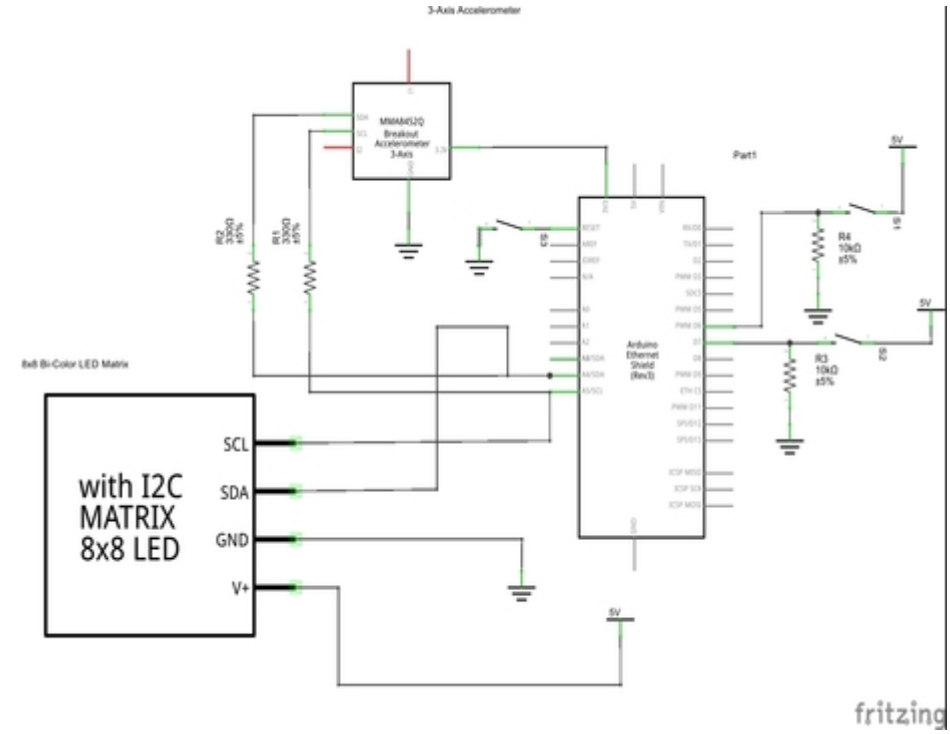
Hardware Design:

1. Placuta Arduino
2. Breadboard
3. Matrice LED 8×8 bi-colora + Circuit de Control
4. Accelerometru cu 3 axe
5. 2 butoane (POWER ON/OFF, RESTART)
6. 2 rezistente de 330 Ohm
7. 2 rezistente de 10K Ohm
8. Fire Tata - Tata

Schema Arduino



Schema Bloc



Software Design

In general, comentariile din cod explică cum funcționează cea mai mare parte a jocului, dar, pentru a explica mai mult in detaliu, board-ul isi seteaza valorile și apoi, dacă va citi o apăsare a butonului de pornire, va trece și va atrage atât ținta, cât și șarpele din informațiile primite în funcția de configurare. Apoi va monitoriza pentru a se asigura că șarpele se află în zona de joc și că nu s-a lovit singur, dacă ambele afirmații sunt adevărate, va apela o funcție pentru a detecta poziția accelerometrului și va muta șarpele în pozitia respectiva, dacă unul este fals, se va numi GAME OVER.

Functia principala care se ocupa cu miscarea sarpelui/snake-ului pe matricea 8×8 bi-colora, începe

prin a stabili valorile pe care ar trebui să fie x și y în timp ce accelerometrul și placa sunt ținute plate. În continuare, definim un prag pentru detectarea înclinării și, prin urmare, a mișcării. De exemplu, dacă vrem să ne mișcăm la stânga, avem nevoie de valoarea noastră y, care acționează ca o axă orizontală, minus valoarea noastră de repaus, să fie mai mare decât orice am stabilit pragul.

De asemenea, deoarece dorim să evităm eroarea diagonală în mișcare pe care am întâlnit-o în a doua încercare de a codifica mișcarea, trebuie să ne asigurăm că valoarea absolută a citirii curente x minus x în repaus este mai mică decât pragul nostru. Cu alte cuvinte, trebuie să ne asigurăm că valoarea noastră x este între prag și pragul negativ, ceea ce înseamnă că nu putem citi două direcții simultan. În acest caz, deoarece ne-am setat valorile de repaus la zero, probabil că am putea folosi doar valorile accel.cx și accel.cy în loc să scădem valoarea de repaus din ele, totuși acest lucru va face astfel încât, dacă vrem vreodată să schimbăm poziția neutră putem face asta cu ușurință. Celelalte trei direcții funcționează în moduri similare.

Pentru fiecare direcție vom salva, de asemenea, direcția în care mergea într-o variabilă care, pentru a ne asigura că jocul are suficientă provocare, va fi activată atunci când accelerometrul revine la o stare plată și astfel șarpele va continua să se miște în orice direcție. ultima direcție în care se mișcase până când este din nou înclinată.

De asemenea, în timp ce șarpele se află pe terenul de joc, se mișcă și nu s-a lovit singur, jocul va monitoriza poziția șarpelui și a țintei pentru a vedea dacă jucătorul înscrie.

Alta functie destul de important de mentionat in acest proiect este cea care realizeaza cresterea sarpelui, mai exact aceasta verifică dacă capul șarpelui, reprezentat de snakeX[0] și snakeY[0], este același cu coordonatele țintei. Dacă acest lucru este fals, ținta rămâne acolo unde a fost și șarpele rămâne la aceeași lungime. Dacă este adevărat, atunci șarpele va crește cu un pixel și scorul va crește cu un punct. Codul va verifica cât de lung este șarpele atunci, dacă este mai mică decât lungimea maximă a șarpelui, atunci va apela o funcție și va face o altă țintă după care șarpele va urma. Dacă nu este mai mică decât lungimea maximă, nu va mai face ținte.

În codul meu am setat lungimea maximă a șarpelui la 64, care este întreaga matrice LED. Am făcut asta pentru că am vrut ca jocul să se termine doar dacă șarpele a murit și nu dacă a ajuns la o anumită lungime.

Un mic rezumat in ceea ce fac functiile:

- void playGame() –> această funcție va porni jocul
- void draw() –> aceasta functie va apela funcțiile pentru a desena șarpele și ținta
- void drawSnake() –> aceasta functie va desena sarpele
- void drawTarget() –> aceasta functie va atrage ținta
- boolean inPlayField(int x, int y) –> va stabili terenul de joc pentru țintă și șarpe pentru a se spawna pe harta
- void makeTarget() –> aceasta functie va alege o valoare aleatorie și va stabili ținta
- boolean isPartOfSnake(int x, int y) –> aceasta functie se va citi dacă valoarea face parte sau nu din șarpe.
- void makeSnake() –> aceasta functie va stabili valorile șarpelui
- void moveSnake() –> această funcție va muta șarpele în funcție de accelerometru
- void gameOver() –> aceasta functie va afișa spectacolul de lumini și va nota după terminarea jocului

Cod Sursa

```
#include <SparkFun_MMA8452Q.h>


//First we need to set up some libraries so that we can use all the parts we
want
#include <Wire.h> // we need this to use I2C which is a requirement of the
// acceleromter and LED matrix
//#include <SFE_MMA8452Q.h> // this will include the library so we can use
the accelerometer

//these two libraries are so we can use the LED matrix
#include "Adafruit_LEDBackpack.h"
#include "Adafruit_GFX.h"

// these three libraries are needed to connect the Arduino with the internet
and Twitter.
#include <SPI.h> // needed in Arduino 0019 or later
#include <Ethernet.h>
#include <EEPROM.h> // this will allow us to store memory on the Arduino.

//Now we need to set up some features for our accelerometer
MMA8452Q accel;

//Now we need to set up some features for our LED matrix
Adafruit_BicolorMatrix matrix = Adafruit_BicolorMatrix();

//Now to set up our Ethernet shield and Twitter

byte mac[] { 0x90, 0xA2, 0xDA, 0x0D, 0x14, 0x79 };
//This is where you enter the information for your Ethernet shield
//You should find your code on a label on the bottom of the shield
//You want to enter it 0x## where ## is the information found on your label.

//If you want to specify a specific a IP address you can enter it like seen
below
// byte ip[] = { 10, 3, 19, 73 };

//Specify your Token so we can use your Twitter
//You can get your token from (http://arduino-tweet.appspot.com/)
//this is my token, make sure to use your own for it to show up on your
Twitter news feed

//this will set up a message which we will use later when we want to post to
Twitter
char msg[70];

//Now we need to set up some variables
const int powerButton = 7;
const int sendButton = 6;
int resetButton;
```

```
const int maxSnake = 64; //this is the maximum length the snake can reach
int snakeX[maxSnake]; // this is the x coordinate of the snake
int snakeY[maxSnake]; // this is the y coordinate of the snake
int snakeLength = 1; // this is the starting length of the snake
int targetX; //this is the x coordinate of the item we are going after
int targetY; //the y coordinate of the item we are going after
unsigned long targetPrevTime = 0; //these will be used to help with the
timing of the target
unsigned long targetBlinkTime = 1000 / 250;
int targetLED = LED_GREEN;
unsigned long prevTime = 0;
unsigned long delayTime = 500;
int score = 0;

int8_t collision = 0;

// these variables will be used to make our switch work properly
int mode = 0;
const int NUMMODES = 3;

int storedScore = 0; //this will be used to store our score inthe EEPROM

static int prevDirection; //this will be used to store the previous
direction the snake was moving in

void setup() {
  //Set up our pins as inputs
  pinMode(powerButton, INPUT);
  pinMode(sendButton, INPUT);
  pinMode(resetButton, INPUT);

  Serial.begin(9600);

  matrix.begin(0x70); //pass in the address, set up the matrix

  accel.init(); //this will intiate the accelerometer
  randomSeed(analogRead(A0)); //we use this so that we will be able to
  //make a random pixel light up that will serve as the target our snake is
after.

  makeSnake(); //this will call a function to set up where our snake will
start on the matrix
  makeTarget(); // this will call a function to set up where our target will
be on the matrix.
}

void loop() {
  // this will clear the display just in case something else was on it
before
  matrix.clear();
  matrix.writeDisplay();
```

```
  static int prevPowerState = 0; //this will ensure we read only the button
push, and not how long it is pushed.
  int powerButtonState = digitalRead(powerButton); //this will moniter the
state of the button
  if ((powerButtonState == HIGH) && (prevPowerState == LOW)) {
    mode = (mode + 1) % NUMMODES;
  }
  prevPowerState = powerButtonState;

  switch (mode) {
    case 0: //if the button has not been pushed nothing happens
      //Serial.println("OFF"); debug to make sure switch works properly
      break;
    case 1: //if the button is pushed the game starts
      playGame();
      break;
    case 2:
      gameOver();
      break;
  }


  int sendButtonState = digitalRead(sendButton); //this will send a Tweet of
the score by pushing the red button
  if (sendButtonState == HIGH) {
  }
}

void playGame() { // this funciton will start up the game

  unsigned long currentTime = millis();
  if (currentTime - prevTime >= delayTime) {
    nextstep();
    prevTime = currentTime;
  }
  draw(); // this will call the draw function which will draw the snake and
target
}

void nextstep() {
  for (int i = snakeLength - 1; i > 0; i--) {
    snakeX[i] = snakeX[i - 1];
    snakeY[i] = snakeY[i - 1];
  }

  for (int8_t h = 3; h <= snakeLength; h++) { // this will monitor if the
snake's head is colliding with another part
    // of the snake. we use 3 because it is basically impossible for the
snake to hit itself before then.
    if ((snakeX[0] == snakeX[h]) && (snakeY[0] == snakeY[h]))
```

```
    {
      collision = 1; // if it is then it will send a message of collision.
    }
  }
  Serial.print("Collision:"); //used for debugging to ensure the collision
detection worked
  Serial.println(collision);

  if ((inPlayField(snakeX[0], snakeY[0])) && collision == 0) { // if the
snake is in the playing field
    // or not colliding with itself then we will be able to continue to play
    moveSnake(); //this will call a function to move the snake

    if ((snakeX[0] == targetX) && (snakeY[0] == targetY)) { //this will read
if the snake is on the target
      snakeLength++; //if it is it will gain a length of one
      score++; // and the score will increase by one
      Serial.print("Score: "); // the score will then be written on the
serial monitor
      Serial.println(score);
      if (snakeLength < maxSnake) { // if the snake is less than the maxium
lenght
        makeTarget(); //then we will make a new target to go after
      }
      else {
        targetX = targetY = -1; //if it is not less than the max, then we
will not
        //make any more targets
      }
    }
  }
  else { // if it is not in the play field or it collids with itself then
the game is over.
    mode = 2;
  }
}

void draw () { //this will call the functions to draw the snake and target
  matrix.clear();
  drawSnake();
  drawTarget();
  matrix.writeDisplay();
}

void drawSnake() { //this will draw the snake
  for ( int i = 0; i < snakeLength; i++) {
    matrix.drawPixel(snakeX[i], snakeY[i], LED_GREEN);
  }
}

void drawTarget() { //this will draw the target
```

```
  if (inPlayField(targetX, targetY)) {
    unsigned long currenttime = millis();
    if (currenttime - targetPrevTime >= targetBlinkTime) {
      targetLED = (targetLED == LED_RED) ? LED_OFF : LED_RED;
      targetPrevTime = currenttime;
    }
    matrix.drawPixel(targetX, targetY, targetLED);
  }
}

boolean inPlayField(int x, int y) { //this will set up the playing field for
the target
  // and snake to spawn in.
  return (x >= 0) && (x < 8) && (y >= 0) && (y < 8);
}

void makeTarget() { //this will choose a random value and set up the target
  int x;
  int y;
  x = random(0, 8);
  y = random(0, 8);
  while (isPartOfSnake(x, y)) { //here it will check if the target is part
of the snake
    x = random(0, 8); // if it is it will choose a new random point and thus
continue the game
    y = random(0, 8);
  }
  targetX = x;
  targetY = y;
}

boolean isPartOfSnake(int x, int y) { //this will read if the value is part
of the snake or not.
  for (int i = 0; i < snakeLength - 1; i++) {
    if ((x == snakeX[i]) && (y == snakeY[i])) {
      return true; //if it is it will return true
    }
  }
  return false; // if it is not part of the snake it will return false
}


void makeSnake() { //this will set up the values of the snake

  snakeX[0] = 3;
  snakeY[0] = 4;

  for (int i = 1; i < maxSnake; i++) {
    snakeX[i] = snakeY[i] = -1;
  }
```

```
}

void moveSnake() { // this function will move the snake according to the
accelerometer
  if (accel.available()) {
    accel.read(); //this will read the values of the accelerometer
    Serial.print("x: ");
    Serial.print(accel.cx, 3);
    Serial.print("\t");
    Serial.print("y: ");
    Serial.print(accel.cy, 3);
    Serial.print("\t");
    Serial.print("z: ");
    Serial.print(accel.cz, 3);
    Serial.print("\t");
    Serial.println(prevDirection);
    float restingX = 0; //the accelerometer at rest
    float restingY = 0;
    float threshold = 0.350; //how much it should be tilted to move the
snake.


    Serial.println(accel.cy - restingY);
    Serial.println(accel.cx - restingX);
    Serial.println(abs(accel.cx - restingX));


    if (((accel.cy - restingY) > threshold) && (abs(accel.cx - restingX) <
threshold)) { //if the accelerometer is tilted left
      Serial.println("Left");
      snakeY[0] = snakeY[0] + 1; // the snake will move to the left.
      prevDirection = 1;
    }

    if (((accel.cy - restingY) < -threshold) && (abs(accel.cx - restingX) <
threshold)) { //if the accelerometer is tilted right
      Serial.println("Right");
      snakeY[0] = snakeY[0] - 1; // snake will move to the right
      prevDirection = 2;
    }

    if (((accel.cx - restingX) > threshold) && (abs(accel.cy - restingY) <
threshold)) { //if the accelerometer is tilted up
      Serial.println("Up");
      snakeX[0] = snakeX[0] - 1; // the snake will move up
      prevDirection = 3;
    }

    if (((accel.cx - restingX) < -threshold) && (abs(accel.cy - restingY) <
threshold)) { //if the accelerometer is tilted down
      Serial.println("Down");
```

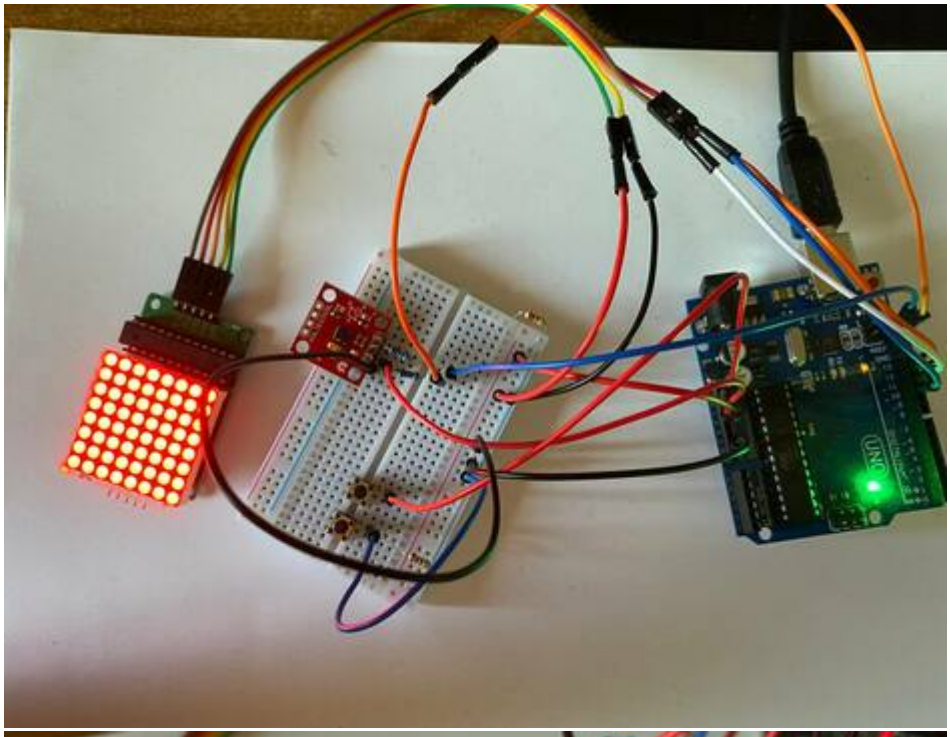
```
      snakeX[0] = snakeX[0] + 1; // it will move the snake down
      prevDirection = 4;
    }

    if ((-threshold < accel.cx) && (accel.cx < threshold) && (-threshold <
accel.cy) && (accel.cy < threshold)) {
      if (prevDirection == 1) {
        snakeY[0] = snakeY[0] + 1;
      }
      if (prevDirection == 2) {
        snakeY[0] = snakeY[0] - 1;
      }
      if (prevDirection == 3) {
        snakeX[0] = snakeX[0] - 1;
      }
      if (prevDirection == 4) {
        snakeX[0] = snakeX[0] + 1;
      }
    }
  }
}

void gameOver() { //this will display the lightshow and score after the game
is over
  Serial.println("Game Over");
  Serial.print("Final Score: ");
  Serial.println(score);
  EEPROM.write (storedScore, score);


  matrix.clear();
  matrix.drawPixel(3, 4, LED_YELLOW); //start a spiral lightshow from the
middle to edge
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(3, 4, 4, 4, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(4, 4, 4, 3, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(4, 3, 2, 3, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(2, 3, 2, 5, LED_YELLOW);
  matrix.writeDisplay();
```

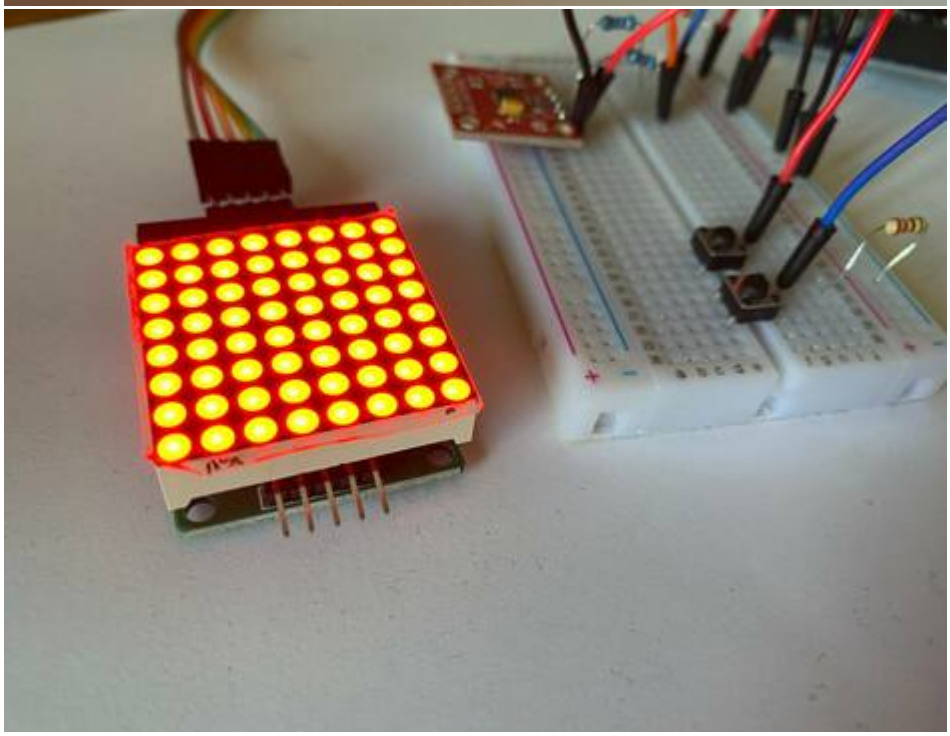
```
  delay(100);

  matrix.drawLine(2, 5, 5, 5, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(5, 5, 5, 2, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(5, 2, 1, 2, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(1, 2, 1, 6, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(1, 6, 6, 6, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(6, 6, 6, 1, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(6, 1, 0, 1, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(0, 1, 0, 7, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(0, 7, 7, 7, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(7, 7, 7, 0, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.drawLine(7, 0, 0, 0, LED_YELLOW);
  matrix.writeDisplay();
  delay(100);

  matrix.setTextWrap(false); //this will display our score on our matrix
  matrix.setTextSize(1);
  matrix.setTextColor(LED_YELLOW);
  matrix.setRotation(3);
```

```
  for (int8_t x = 7; x >= -36; x--) {
    matrix.clear();
    matrix.setCursor(x, 0);
    matrix.print(score);
    matrix.writeDisplay();
    delay (100);
  }
}
```

Rezultate Obţinute

Implementarea proiectului este urmatoarea:





Concluzii

Consider ca proiectul de fata implementat, este destul de util pentru o intelegere mai buna a modului de utilizare al unui Arduino, in special functiile sale principale. De asemenea, in implementarea proiectului este folositor pentru a observa si studia modul in care s-a efectuat lucrul cu datele provenite de la mai multe butoane.

Download

*

Arhiva cod sursa + comentarii + README

In aceasta sectiune de DOWNLOAD putem gasi arhiva care contine urmatoarele:

- Codul sursa al proiectului;
- README, pentru o explicare mai succinta a functiilor din proiect;
- Libraria accelerometrului;

Jurnal

Ca o secțiune de jurnal, in care laborantul să poată urmări progresul proiectului.

1. **Alegere Tema Proiect** - **12.04.2022**
2. **Cautare componente pentru proiect** - **18.04.2022**
3. **Achizitionare Componente Proiect** - **3.05.2022**

Mici probleme tehnice si de logica asupra proiectului

M-am confruntat și cu alte câteva probleme. Inițial am plănuit ca butonul de PORNIRE să acționeze atât ca butonul de pornire, cât și ca butonul de resetare, totuși, după ce am întâmpinat unele probleme în încercarea de a codifica asta și am văzut că aveam loc pentru un alt buton, am decis să conectez pur și simplu un alt buton, care ar face resetarea in sine pentru mine.

În cele din urmă, o problemă cu care am întâlnit a fost că uneori jocul parea să se blocheze când foloseam o baterie de 9 volți. Jocul s-ar fi jucat bine, dar se bloca între timp. L-am testat din nou în timp ce scriam acest lucru și mi s-a părut că funcționează bine, totuși, pentru a evita această posibilă eroare, prefer să-l conectez la computer pentru alimentare. Aceasta poate fi o problemă cu bateria slabă sau pur și simplu nu are suficientă putere pentru a rula toate funcțiile jocului.

De asemenea, am mai constatat că atunci când folosesc bateria, din cauza greutății sale, poate fi dificil să joci. Având bateria atârnată pe lateral pare să fie astfel încât să trebuie să înclinați cu mai multă forță pentru a face ca accelerometrul și șarpele să recunoască mișcarea. Acest lucru ar putea fi corectat probabil prin ajustarea pragurilor din cod.

Bibliografie/Resurse

Mai jos este o listă cu documente, datasheet-uri, resurse Internet folosite, grupate pe

- **Resurse Software**
- **Resurse Hardware**

Resurse Software

1. https://ocw.cs.pub.ro/courses/_media/pm/diverse/pm_cheatsheet_2020.pdf
2. https://github.com/liffiton/Arduino-Cheat-Sheet
3. https://docs.arduino.cc/software/ide-v2
4. https://github.com/adafruit/Adafruit-GFX-Library
5. https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide

Resurse Hardware

1. https://ocw.cs.pub.ro/courses/_media/pm/hardware-cheatsheet.pdf
2. https://ocw.cs.pub.ro/courses/_media/pm/placa_lab_2018.pdf
3. https://github.com/phodal/awesome-hardware-cheatsheet

Documentatie_SaruGabriel_333CB