Chicken Invaders

Introducere

Proiectul are ca scop crearea unei clone a popularului joc din copilărie *Chicken Invaders*, joc foarte popular în perioada anilor 2000. Jocul presupune distrugerea cu ajutorul unei nave spatiale cu tun a găinilor malefice care vor să distrugă galaxia. Prin intermediul acestui proiect, am încercat să readuc nostalgia iubitorilor de jocuri vechi. Pentru acest joc sunt necesare un strop de concentrare, dar și niște reflexe bune.

Descriere generală

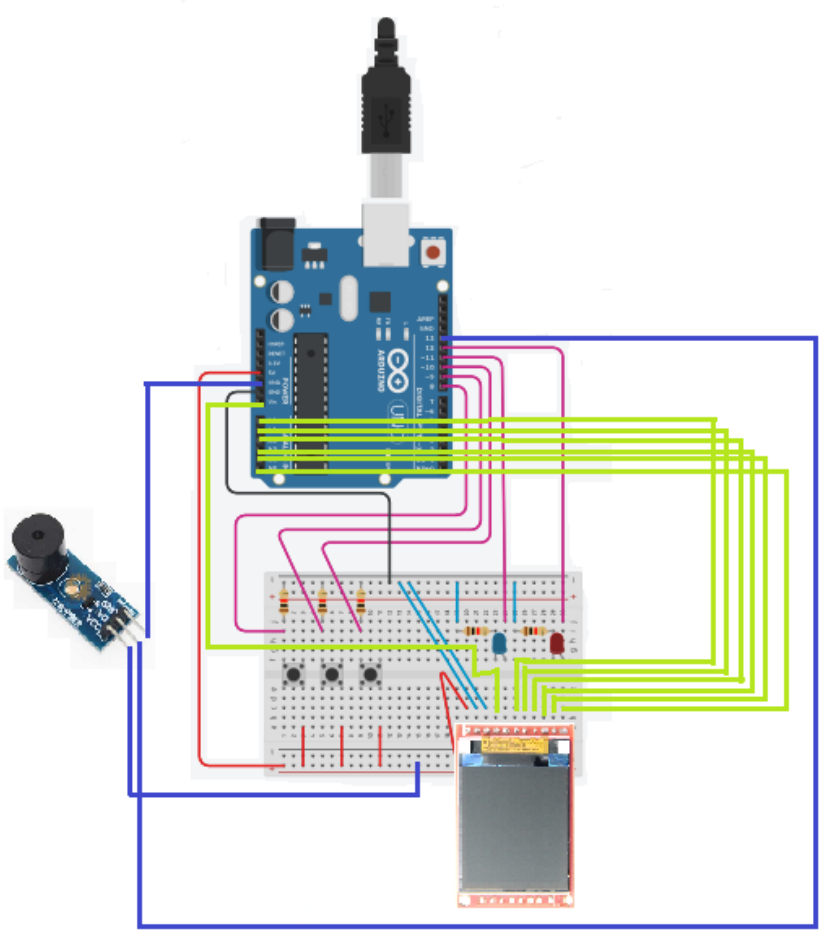
Aplicația este formată din 3 pagini text: de start, cu felicitări pentru învingător și cu păreri de rău pentru pierzător. Toate aceste pagini duc la jocul propriu-zis odată cu apăsarea butonului din mijloc. Ajunsă în joc, persoana poate să deplaseze naveta spațială folosind butoanele laterale(stânga-dreapta) și să tragă cu arma(mijloc) pentru a elimina toate păsările. Jocul începe cu 3 vieți și acestea se termină atunci când jucătorul a tras cu arma de 3 ori(nu neapărat consecutiv), dar nu a nimerit nicio pasăre. Păsările nu se pot deplasa, deci este necesară poziționarea tunului astfel încât acesta să se afle dedesubtul inamicilor. La fiecare aruncare corecta se aprinde LED-ul albastru, iar la fiecare aruncare greșită, cel roșu. Jocul este însoțit și de niște efecte sonore.

Schema bloc

Hardware Design

Listă de componente	Număr de bucăți
Placă Arduino UNO R3	1
Ecran 1.44” SPI ST7735S	1
Breadboard Small	1
Fire tată-tată	
Rezistențe 1 kohm	5
LED roșu	1
LED albastru	1
Modul Buzzer	1
Butoane	3

Schema circuit



Nu am găsit un soft ce conține toate componentele proiectului meu, asa că am folosit Paint și am încercat să fac schema circuitului cât mai vizibilă.

Software Design

Pentru partea de programare a jocului am folosit mediul de dezvoltare Arduino IDE și două biblioteci compatibile cu ecranul folosit: LCDWIKI_GUI.h și LCDWIKI_SPI.h
Din biblioteca LCDWIKI_SPI.h am folosit un constructor pentru a face legătura între ecran și partea de software.
Funcțiile folosite din biblioteca LCDWIKI_GUI.h sunt următoarele:

- virtual int16_t Get_Width(void) const – folosită pentru a afla lățimea ecranului(în pixeli)

- virtual int16_t Get_Height(void) const – folosită pentru a afla lungimea ecranului(în pixeli)
- void Fill_Screen(uint16_t color) – umple tot ecranul cu o culoare dată
- void Set_Draw_color(uint16_t color) – setează culoarea utilizată pentru desenarea de obiecte
- void Set_Text_colour(uint16_t color) - setează culoarea utilizată pentru scriere
- void Set_Text_Back_colour(uint16_t color) - setează culoarea fundalului folosit pentru evidențierea textului
- void Set_Text_Back_Size(uint8_t s) - setează dimensiunea textului
- void Print_String(const uint8_t *st, int16_t x, int16_t y) – afișează un șir constant, coordonatele primului caracter fiind date de ultimii doi parametri
- void Fill_Circle(int16_t x, int16_t y, int16_t radius) – desenează pe ecran un cerc
- void Fill_Triangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, int16_t y2) – desenează pe ecran un triunghi
- void Fill_Round_Rectangle(int16_t x1, int16_t y1, int16_t x2, int16_t y2, int16_t radius) – desenează pe ecran un dreptunghi ce are colțurile puțin rotunjite

Funcțiile *Fill_Circle*, *Fill_Triangle* și *Fill_Roung_Rectangle* au fost folosite pentru crearea graficii jocului(glonț, pasăre, navă), pe când cele destinate scrierii au servit la programarea paginii de start a jocului și a paginilor afișate în caz de victorie și înfrângere.

Descrierea funcțiilor implementate:

- *void start_Page()*

Această funcție crează pagina de start a jocului, pagină ce este afișată o singură data pe durata întregii sesiuni de joc. Pagina de start se poate părăsi numai dacă se apasă pe butonul de start.(butonul din mijloc)

- *void Win_Screen() și void Lose_Screen()*

De fiecare data când un jucător câștigă/pierde, se va crea una dintre paginile de mai sus. Ele pot apărea de mai multe ori în decursul unei sesiuni, existând posibilitatea de a juca din nou dacă este apăsat butonul de start.

- *void Draw_Chicken(int i, int j, int r)*

Cu ajutorul acestei funcții se poate desena o singură pasăre alcătuită dintr-un cerc galben(cap), două cercuri negre(ochi) și un triunghi rosu(cioc).

- *void Draw_Chickens()*

Se utilizează pentru a desena păsări pe aproape toată lățimea ecranului și jumătate din lungimea acestuia. Fiecare pasăre este asociată unei singure poziții dintr-o matrice. Această funcție desenează numai păsările ce nu au fost eliminate de către jucător, verificându-se de fiecare data valoarea corespunzătoare fiecărei poziții. În cazul în care valoarea este 1, atunci se desenează o pasăre pe locul respectiv. În caz contrar, se va lăsa un spațiu liber de dimensiunile unei păsări. Se apeleaza funcția *Draw_Chicken*.

- *void Draw_Fire(int id_fire)*

Este folosită pentru desenarea unui glonț. Culoarea acestuia nu este setată în funcție, deci se poate folosi si pentru a șterge un glonț deja existent.

- *int Killed_Chicken(int id_fire)*

Se parcurge partea ocupată de păsări exact ca în funcția *Draw_Chickens*, numai că, de data aceasta se verifică dacă poziția actuală este ocupată de un inamic și dacă glonțul respectiv a reușit să-l lovească pe acesta, caz în care se decrementează numărul de păsări și se modifică valoarea din matrice aflată pe poziția inamicului ce tocmai a fost eliminat. Returnează 1 dacă pasărea a fost eliminată, iar 0 în caz contrar.

- *void Delete_Fire(int id_fire)*

Este bazată pe același principiu ca *Draw_Fire*, dar în acest caz culorile sunt deja prestabilite și se redesenează și nava spațială(glonțul porneste din vârful navei, iar aceasta se deteriorează). Se apelează funcția *Draw_Ship*.

- *void Draw_Ship(int shx, int shy)*

Se folosește pentru a se desena nava spațială formată dintr-un dreptunghi cu colțurile rotunjite și un triunghi pe post de tun. Coordonatele se calculează pe baza poziției vârfului triunghiului.

- *void Move_Ship()*

Are ca scop mutarea navei dintr-o parte în alta în funcție de direcția în care jucătorul decide să mute nava. Se face o desenare deasupra navei originale cu o navă de aceleași dimensiuni, dar de culoarea ecranului, după care se desenează la câțiva pixeli distantă o nouă navă.

- *void Launch_Fire(int id_fire)*

Funcția se folosește pentru a simula mișcarea glonțului. Aceasta este realizată prin ștergerea și desenarea repetată de glonțuri(cercuri) cu aceeași coordonată x, dar de coordonate y diferite(y scade). Se efectuează mișcarea până când glonțul „iese” din ecran, caz în care buzzer-ul emite un sunet specific(sunt atâtea bip-uri câte vieți au mai rămas jucătorului), numărul de vieți scade și LED-ul roșu se aprinde, sau până când funcția *Killed_Chicken* este 1, situație în care se aprinde LED-ul albastru și se aude un bip mai scurt. Funcția este făcută în așa fel încât să fie posibilă mutatea navei la acțiunea butoanelor.

- *void beep(unsigned char delayms)*

Utilizată pentru sunetul emis de modulul buzzer.

- *void setup()*

Se declară rolul LED-urilor, modulului buzzer și al butoanelor și se afișează pagina de start.

- *void loop()*

În primul rând, ne asigurăm că LED-urile nu sunt aprinse. Citim stările butoanelor și, în funcție de acestea și alți factori decidem ce acțiune dorim să executăm.

1. Se apasă butonul din mijloc și jocul încă nu a început(ne aflăm pe pagina de start): Desenăm toate păsările, dăm valori inițiale poziției navei, calculăm numărul de păsări afișate.
2. Se apasă butonul din mijloc și jocul a început(ne aflăm pe pagina principală): Lansăm un glonț. Nu pot fi lansate două glonțuri consecutive.
3. Se apasă butonul din stânga și jocul a început: Mutăm nava la stânga.
4. Se apasă butonul din dreapta și jocul a început: Mutăm nava la dreapta.
5. Suntem în timpul jocului și nu mai există păsări pe ecran și mai avem vieți: Se resetează matricea

și numărul de găini pentru a pregăti următoarea rundă și se afișează pagina creată de *Win_Screen*.
6. Suntem în timpul jocului și mai sunt păsări pe ecran, dar nu mai avem vieți: Aceleași acțiuni ca la punctul precedent, dar se afișează pagina create de *Lost_Screen*.

Cod

```
#include <LCDWIKI_GUI.h>
#include <LCDWIKI_SPI.h>

#define MODEL ST7735S128
#define CS   A5
#define CD   A3
#define RST  A4
#define SDA  A2
#define SCK  A1
#define LED  A0
#define STANGA 10
#define MIJLOC 9
#define DREAPTA 8
#define BAD 12
#define GOOD 13
#define BUZZER 11
int start = 0;
int stare_stanga = 0;
int stare_mijloc = 0;
int stare_dreapta = 0;
int chickens = 0;
int killed = 0;
int chicken_matrix[4][8] =
{ {1, 1, 1, 1, 1, 1, 1, 1},
  {1, 1, 1, 1, 1, 1, 1, 1},
  {1, 1, 1, 1, 1, 1, 1, 1},
  {1, 1, 1, 1, 1, 1, 1, 1},
};
int shooterX = 0;
int shooterY = 0;
int fireX[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int fireY[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int fires = -1;
int dir = 0;
int moveX = 4;
int moveY = 0;
int lives = 3;

LCDWIKI_SPI mylcd(MODEL, CS, CD, -1, SDA, RST, SCK, LED);
//model,cs,dc,sdo,sda,reset,sck,led

#define BLACK   0x0000
#define BLUE    0x20B4
```

```
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xBD96
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

void Draw_Chicken(int i, int j, int r)
{
  mylcd.Set_Draw_color(YELLOW);
  mylcd.Fill_Circle(i, j, r);
  mylcd.Set_Draw_color(BLACK);
  mylcd.Fill_Circle(i + 2, j - 2, 1);
  mylcd.Fill_Circle(i - 2, j - 2, 1);
  mylcd.Set_Draw_color(RED);
  mylcd.Fill_Triangle(i + 2, j + 2, i, j, i - 2, j + 2);
}
void Draw_Chickens()
{

  mylcd.Set_Draw_color(YELLOW);
  int r = 6, i, j, ii = 0, jj = 0;
  for (i = 2 * r - 1; i < mylcd.Get_Display_Width(); i += 2 * r + 3)
  {
    for (j = 2 * r - 3; j < mylcd.Get_Display_Height() / 2; j += 2 * r + 3)
    {
      Serial.print(chicken_matrix[jj][ii]);

      if (chicken_matrix[jj][ii] == 0)
      {
        mylcd.Set_Draw_color(BLUE);
        mylcd.Fill_Circle(i, j, r);
      }

      else
      {
        Draw_Chicken(i, j, r);
      }
      //

      Serial.print(jj);
      jj = jj + 1;
    }
    ii = ii + 1;
    jj = 0;
    Serial.println(ii);

  }

}
```

```
void Draw_Fire(int id_fire)
{
  mylcd.Set_Draw_color(RED);
  mylcd.Fill_Circle(fireX[id_fire], fireY[id_fire], 2);
}

int Killed_Chicken(int id_fire)
{
  int r = 6, i, j, ii = 0, jj = 0;
  for (i = 2 * r - 1; i < mylcd.Get_Display_Width(); i += 2 * r + 3)
  {
    for (j = 2 * r - 3; j < mylcd.Get_Display_Height() / 2; j += 2 * r + 3)
    {
      if (chicken_matrix[jj][ii] == 1 && fireY[id_fire] <= j &&
fireX[id_fire] >= i - 6 && fireX[id_fire] <= i + 6)
      {
        chicken_matrix[jj][ii] = 0;
        chickens--;
        return 1;
      }

      jj = jj + 1;
    }

    ii = ii + 1;
    jj = 0;
  }

  return 0;
}

void Throw_Egg()
{
  //if
}

void Delete_Fire(int id_fire)
{
  mylcd.Set_Draw_color(BLUE);
  mylcd.Fill_Circle(fireX[id_fire], fireY[id_fire], 2);
  mylcd.Set_Draw_color(MAGENTA);
  Draw_Ship(shooterX, shooterY);
}

void Draw_Ship(int shx, int shy)
{

  mylcd.Fill_Round_Rectangle(shx - 17, shy + 7, shx + 17, shy + 14, 4);
  mylcd.Fill_Triangle(shx - 7, shy + 7 , shx, shy, shx + 7, shy + 7);
}
```

```
void Move_Ship()
{
  if (dir == 1 && shooterX + 10 + moveX <= mylcd.Get_Width())
  {
    mylcd.Set_Draw_color(BLUE);
    Draw_Ship(shooterX, shooterY);
    mylcd.Set_Draw_color(MAGENTA);
    Draw_Ship(shooterX + moveX, shooterY);
    delay(100);
    shooterX = shooterX + moveX;
  }

  else if (dir == -1 && shooterX - 10 - moveX >= 0)
  {
    mylcd.Set_Draw_color(BLUE);
    Draw_Ship(shooterX, shooterY);
    mylcd.Set_Draw_color(MAGENTA);
    Draw_Ship(shooterX - moveX, shooterY);
    delay(100);
    shooterX = shooterX - moveX;
  }

  else
  {
    return;
  }
}

void Launch_Fire(int id_fire)
{
  fireX[fires] = shooterX;
  fireY[fires] = shooterY;

  while (fireY[fires] != 0)
  {
    stare_stanga = digitalRead(STANGA);
    stare_dreapta = digitalRead(DREAPTA);

    if (stare_stanga == HIGH && start == 1)
    {
      dir = -1;
      Move_Ship();
    }

    if (stare_dreapta == HIGH && start == 1)
    {
      dir = 1;
      Move_Ship();
    }

    Draw_Fire(fires);
```

```
    killed = Killed_Chicken(fires);
    if (killed == 1)
    {
      digitalWrite(GOOD, HIGH);
      beep(50);
      Delete_Fire(fires);
      break;
    }

    Delete_Fire(fires);
    fireX[fires] = fireX[fires];
    fireY[fires] = fireY[fires] - 2;
    if (fireY[fires] <= 0)
    {
      digitalWrite(BAD, HIGH);
      lives--;
      if(lives==2){
        beep(100);
        delay(10);
        beep(100);
      }
      if(lives==1){
        beep(100);
      }
      if(lives==0){
        beep(200);
        beep(200);
      }
    }
  }

  Draw_Chickens();
}

void Win_Screen()
{
  mylcd.Fill_Screen(GREEN);
  mylcd.Set_Text_colour(BLACK);
  mylcd.Set_Text_Back_colour(GREEN);
  mylcd.Set_Text_Size(2);
  mylcd.Print_String("CONGRATS! <3", 14, 54);
  mylcd.Set_Text_Size(1);
  mylcd.Print_String("Wanna play again?", 7, 74);
  mylcd.Print_String("Press the middle button", 0, 84);
}

void Lose_Screen()
{
  mylcd.Fill_Screen(RED);
  mylcd.Set_Text_colour(BLACK);
  mylcd.Set_Text_Back_colour(RED);
```

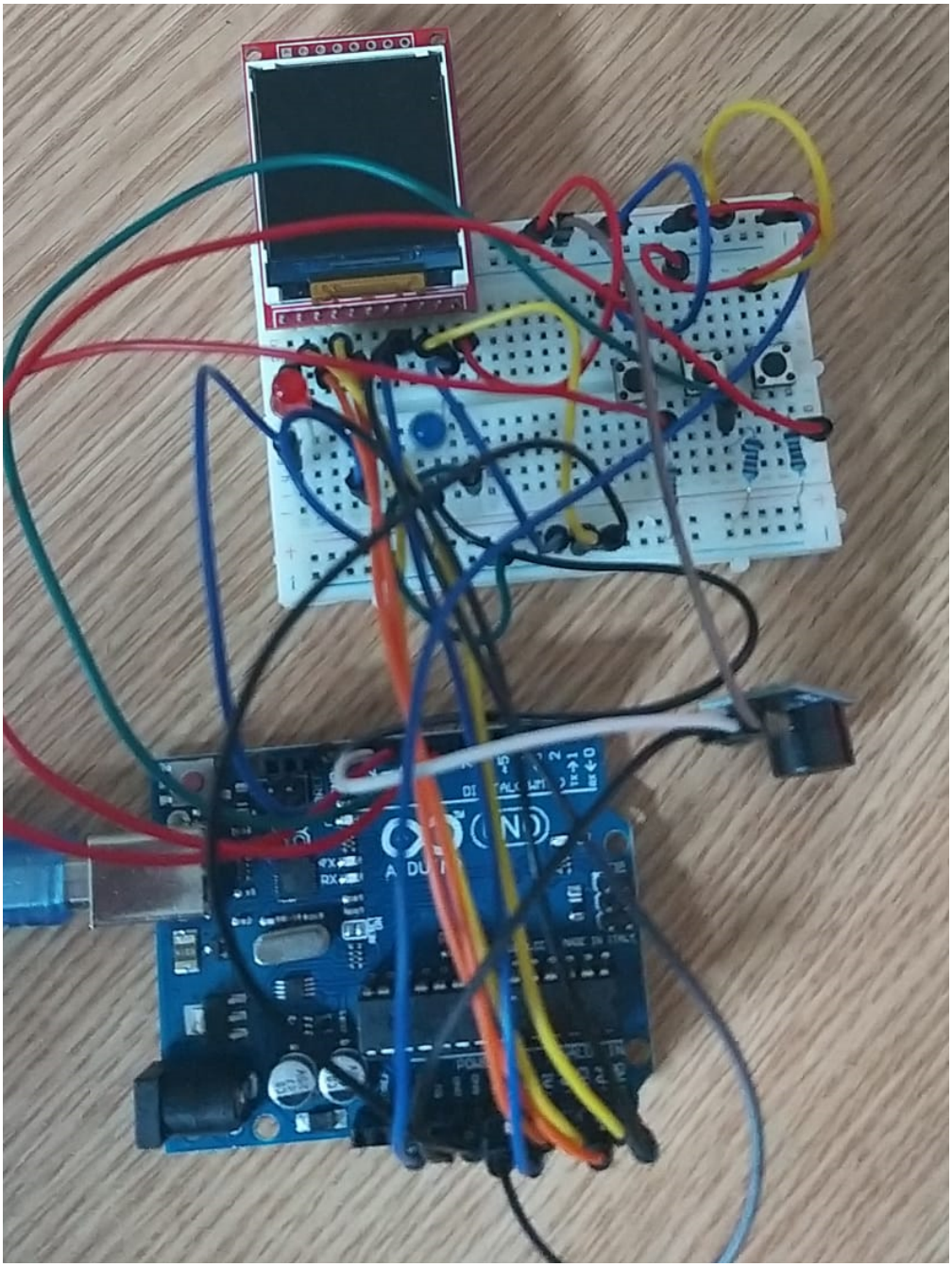
```
  mylcd.Set_Text_Size(2);
  mylcd.Print_String("BAD LUCK", 14, 54);
  mylcd.Set_Text_Size(1);
  mylcd.Print_String("Wanna try again?", 7, 74);
  mylcd.Print_String("Press the middle button", 0, 84);
}

void start_Page()
{
  mylcd.Fill_Screen(RED);
  mylcd.Set_Text_colour(BLACK);
  mylcd.Set_Text_Back_colour(RED);
  mylcd.Set_Text_Size(3);
  mylcd.Print_String("START", 14, 54);
  mylcd.Set_Text_Size(1);
  mylcd.Print_String("Press the middle button", 0, 84);
}

void beep(unsigned char delayms)
{
  analogWrite(BUZZER, 20);
  delay(delayms);
  analogWrite(BUZZER ,0);
  delay(delayms);
}

void setup()
{
  pinMode(STANGA, INPUT);
  pinMode(MIJLOC, INPUT);
  pinMode(DREAPTA, INPUT);
  pinMode(BAD, OUTPUT);
  pinMode(GOOD, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  Serial.begin (9600);
  mylcd.Init_LCD();
  start_Page();
}

void loop()
{
  digitalWrite(GOOD, LOW);
  digitalWrite(BAD, LOW);

  stare_mijloc = digitalRead(MIJLOC);
  stare_stanga = digitalRead(STANGA);
  stare_dreapta = digitalRead(DREAPTA);

  if (stare_mijloc == HIGH && start == 0)
  {
    start = 1;
```

```
    mylcd.Fill_Screen(BLUE);
    Draw_Chickens();
    chickens = 4 * 8;
    shooterX = mylcd.Get_Width() / 2;
    shooterY = mylcd.Get_Height() - 14;
    mylcd.Set_Draw_color(MAGENTA);
    Draw_Ship(shooterX, shooterY);
    delay(1000);
    return;
  }

  if (stare_mijloc == HIGH && start == 1)
  {
    fires++;
    Launch_Fire(fires);
    fires--;
  }

  if (stare_stanga == HIGH && start == 1)
  {
    dir = -1;
    Move_Ship();
  }

  if (stare_dreapta == HIGH && start == 1)
  {
    dir = 1;
    Move_Ship();
  }

  if (chickens == 0 && start == 1 && lives > 0)
  {
    for(int i=0; i<10; i++)
    {
      beep(50);
      delay(20);
    }
    start = 0;
    for (int i = 0; i < 4; i++)
      for (int j = 0; j < 8; j++)
        chicken_matrix[i][j] = 1;
    chickens = 4 * 8;
    lives = 3;
    delay(1000);
    Win_Screen();
  }

  if (chickens != 0 && start == 1 && lives == 0)
  {
    start = 0;
    for (int i = 0; i < 4; i++)
```

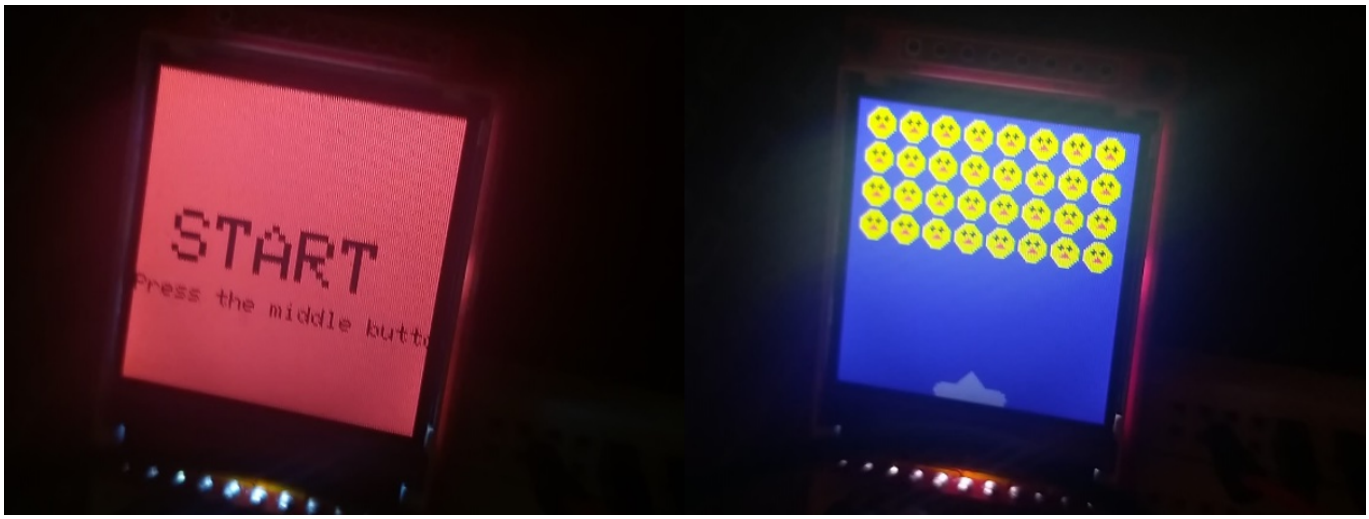
```
      for (int j = 0; j < 8; j++)
        chicken_matrix[i][j] = 1;
    chickens = 4 * 8;
    lives = 3;
    delay(1000);
    Lose_Screen();
  }
}
```

Rezultate Obţinute

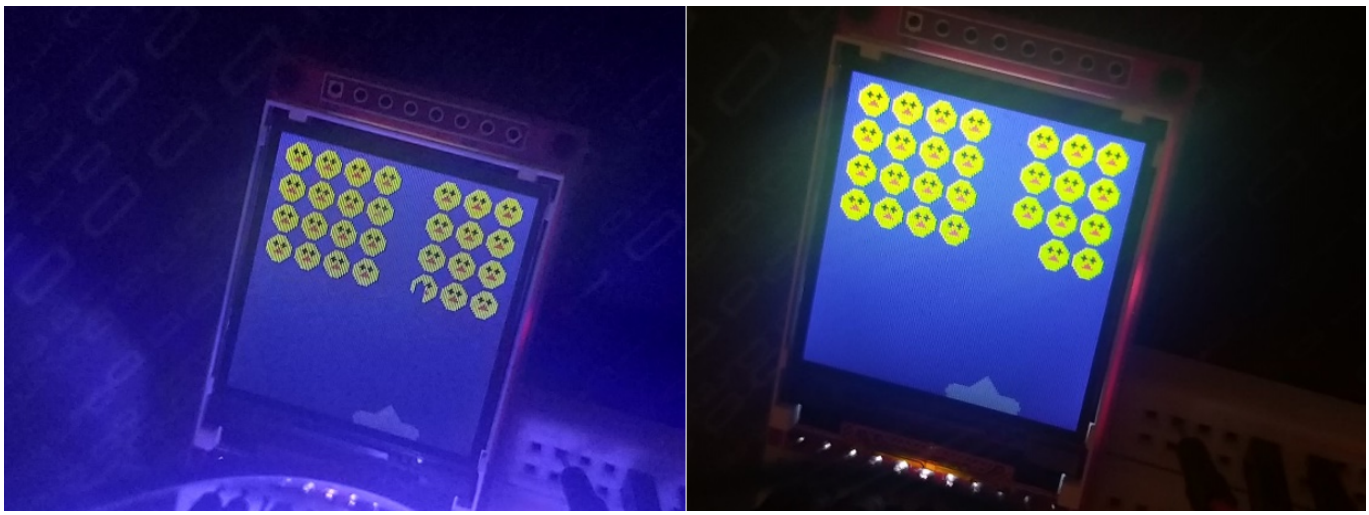
Circuit



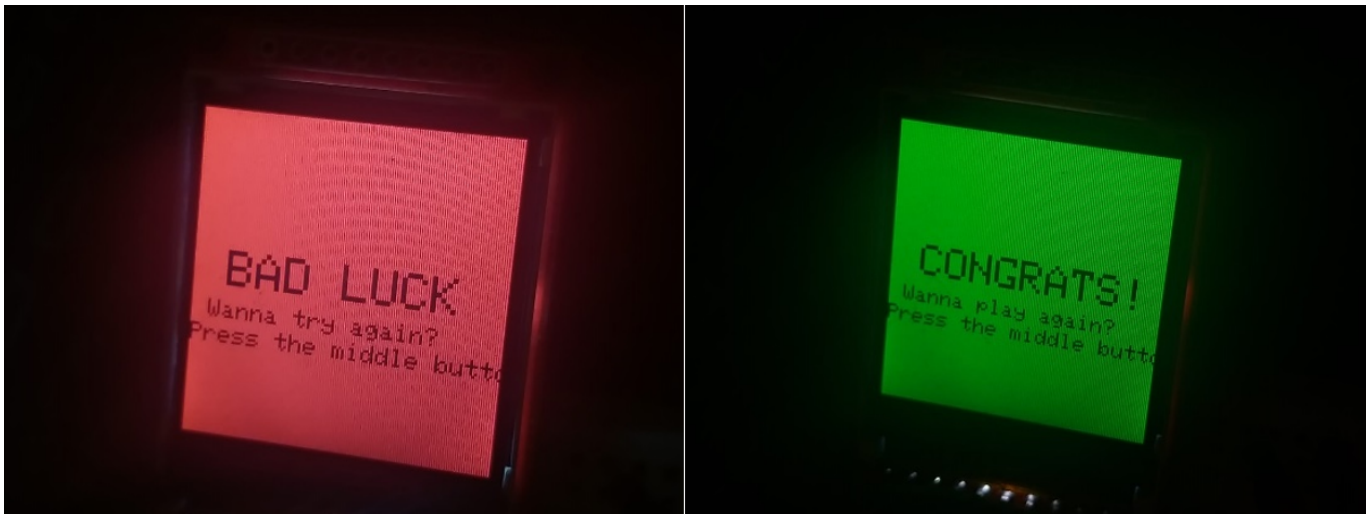
Pagină de start + Începutul jocului



Procedeu de eliminare a unei găini



Pagină pentru câștigător/pierzător



Concluzii

- Am aprofundat logica din spatele modului de conectare al componentelor
- Plănuiesc pe viitor să mai lucrez la acest joc deoarece nu am reușit să implementez absolut tot ce îmi doream.
- A fost destul de dificil să găsesc o bibliotecă ce este compatibilă cu ecranul meu. Din cauza acestui fapt, nu am găsit foarte multe informații utile în privința bibliotecii.
- În ciuda acestui „ghinion”, nu regret deloc că am ales această temă, acest joc fiind probabil unul dintre preferatele mele. A fost interesant să lucrez cu ceva cu totul nou pentru mine.

Download

În arhiva de mai jos sunt incluse codul sursă, un README, scheme și poze cu proiectul.

chicken_invaders_vespan_diana-gabriela.zip

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

http://www.lcdwiki.com/1.44inch_Arduino_SPI_Module_ST7735S_SKU:MAR1441#How_to_use_on_Arduino
https://www.instructables.com/Arduino-YL-44-Buzzer-module/
LCDWIKI GUI lib Manual
LCDWIKI SPI lib Manual

Export to PDF