Chicken Invaders

Introducere

Proiectul constă în realizarea unei clone a jocului video “Chicken Invaders”. Jocul decurge in felul urmator: Jucatorul(gaina protagonista) incepe de la “level 1” si trebuie sa elimine extraterestrii malefici pentru a avansa la urmatorul nivel. Numarul inamicilor difera de la nivel la nivel si la fel si probabilitatea acestora de a riposta si a trage inapoi inspre jucator. Scopul jocului este de a face un scor cat mai mare sau poate chiar sa terminati jocul daca puteti:))) Good luck finding the easter egg!

Descriere generală

Implementarea acestui proiect se face sub forma unei console de joc minimalista, pe care utilizatorul sa se poata juca jocul “Chicken Invaders” folosind un joystick pentru controlul gainii protagoniste si apasand joystickul poate trage asupra inamicilor. Jocul va fi afisat pe un ecran LCD 16 x 2 I2C, iar efectele sonore vor fi redate printr-un modul Buzzer. Arduino Uno serveste ca si componenta de baza, conectandu-se la toate celelalte componente prin intermediul Breadboard-ului. Joystick-ul este interfata de control, permitand utilizatorului sa mute nava spatial, iar apasand pe joystick se poate trage asupra inamicilor. Ecranul LCD afiseaza grafica și informatiile despre joc, iar buzzer-ul reda efectele sonore, completand experiența de joc.

Diagrama bloc

Diagrama bloc realizata pe platforma → Draw.io

Hardware Design

Lista piese:

Componentă	Descriere
Arduino Uno R3	MC ATmega328P
Modul Joystick PS2	Interfață de control
Ecran LCD 1602 IC2	Afișaj joc

Modul Buzzer	Redare efecte sonore
Breadboard	Conexiune piese
Fire	Conectare pini

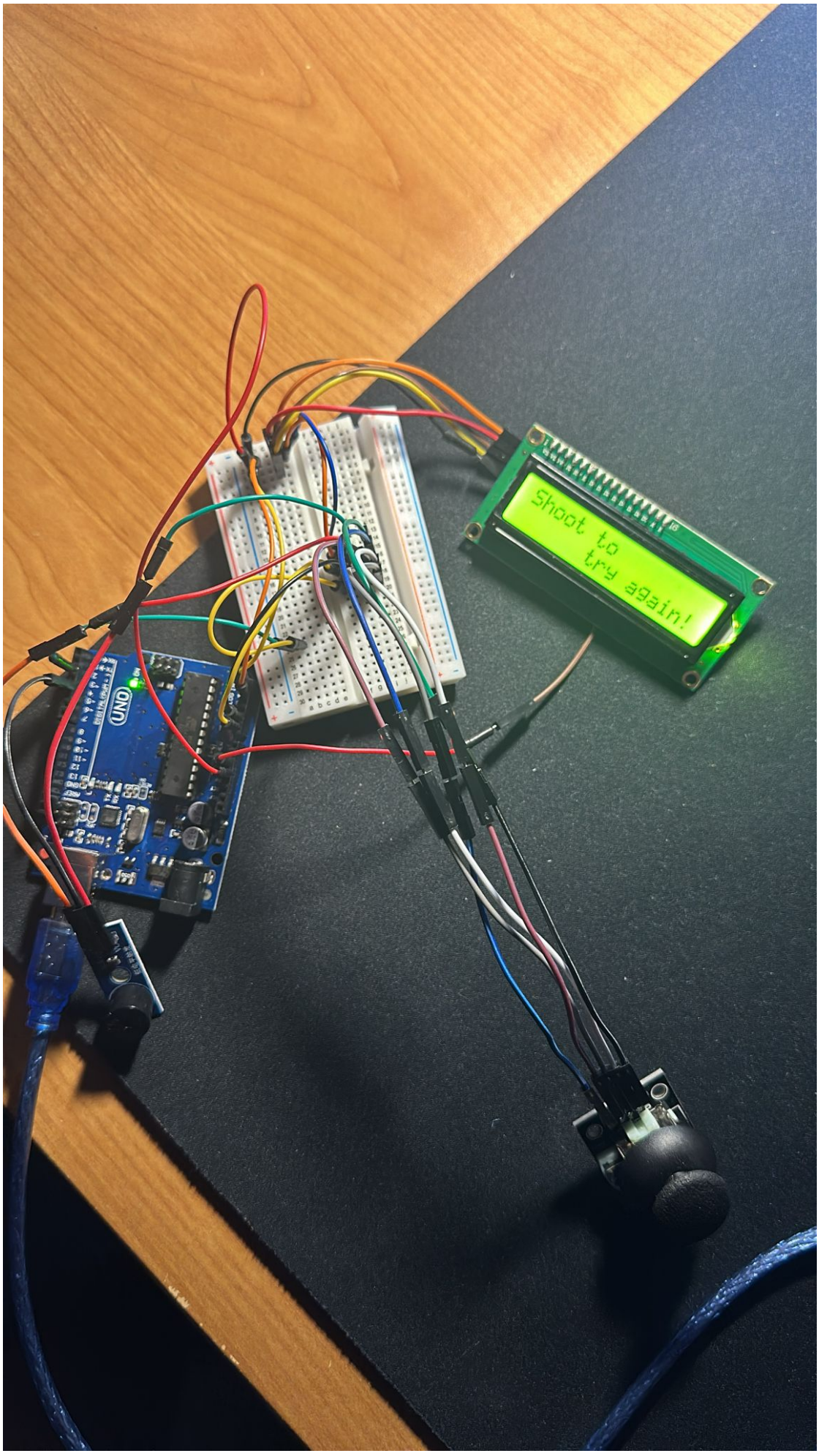
- **Arduino Uno R3**
  - 5V → ofera 5 volti pentru componentele conectate.
  - GND → ofera referinta de masa pentru circuit.
  - A0 → intrare analogica pentru citirea pozitiei orizontale (axa X) de la joystick.
  - A1 → intrare analogică pentru citirea pozitiei verticale (axa Y) de la joystick.
  - A4 → linie de date I2C utilizata pentru comunicarea cu modulul LCD I2C.
  - A5 → linie de ceas I2C utilizata pentru sincronizarea comunicatiei cu modulul LCD I2C.
  - D2 → intrare digitala pentru citirea starii butonului joystick-ului (SW).
  - D4 → iesire digitala pentru controlul modulului buzzer (emiterea sunetelor).

- **Modul Joystick PS2**
  - 5V → conectat la pinul 5V de pe Arduino pentru a alimenta joystick-ul.
  - GND → conectat la pinul GND de pe Arduino pentru referinta de masa.
  - VRx → conectat la pinul A0 pentru a citi pozitia pe axa X.
  - VRy → conectat la pinul A1 pentru a citi pozitia pe axa Y.
  - SW → conectat la pinul D2 pentru a citi starea butonului joystick-ului.

- **Ecran LCD 1602 I2C**
  - VCC → conectat la pinul 5V de pe Arduino pentru a alimenta ecranul.
  - GND → conectat la pinul GND de pe Arduino pentru referinta de masa.
  - SDA → conectat la pinul A4 de pe Arduino pentru linia de date I2C.
  - SCL → conectat la pinul A5 de pe Arduino pentru linia de ceas I2C.

- **Modul Buzzer**
  - VCC → conectat la pinul 5V de pe Arduino pentru a alimenta buzzer-ul.
  - GND → conectat la pinul GND de pe Arduino pentru referința de masa.
  - I/O → conectat la pinul D4 de pe Arduino pentru a controla buzzer-ul.

Forma initiala


Shoot to
try again!

Schema electrica:

Schema realizata pe platforma → Circuito.io.

Software Design

Proiectul a fost implementat folosind Arduino IDE

Bibliotecile folosite :

- Arduino - hd44780 extensive I2C by Bill Perry → a fost folosita pentru generarea afisajului pe Display-ul LCD.
- Arduino - Wire by Nicholas Zambetti → a fost folosita pentru realizarea comunicarii intre Microcontroller si Display-ul LCD.

Variabile globale

- Variabile pentru pini:

```
const int joy_X = A0;
const int joy_B = 2;
const int buzz_P = 4;
```

- Variabile pentru input-urile primite(sub forma de butoane):

```
const int X_RIGHT = 0;
const int X_LEFT  = 1;
const int JOY_PRESS = 2;
const int DO_NOTHING = 3;
```

- Variabile pentru caracterele speciale folosite in dezvoltarea jocului:

```
const byte CHICKEN = 0;
const byte BULLET_UP = 1;
const byte BULLET_DOWN = 2;
const byte CHICKEN_BULLET = 3;
const byte ENEMY1 = 4;
const byte ENEMY2 = 5;
const byte ENEMY1BULLET = 6;
const byte ENEMY2BULLET = 7;
```

- Variabile pentru definirea spatiului de joc:

```
const int WIDTH = 16;
const int HEIGHT = 4;
```

- Variabile pentru logica jocului:

```
byte frame;
char screen_buff[HEIGHT / 2][WIDTH + 1];
byte enemy_frame_delay = 5;
byte fire_chance = 20;
int score = 0;
byte level = 1;
byte enemy_left = 0;
byte enemy_active = 6;
bool reset = false;
```

Clase

- **GameObject** :

Clasa GameObject este folosita ca superclasa pentru toate clasele aferente si implementeaza functionalitati simple a tuturor obiectelor ce vor fi folosite pentru ilustrarea efectelor jocului: variabile care reprezinta coordonata x si y, viteza cat si getteri si setteri si o functie pentru a afla daca 2 obiecte s-au intersectat.

```
class GameObject
{
  protected:
    int8_t obj_x;
    int8_t obj_y;
    int8_t obj_speed;

  public:
    GameObject(): obj_x(0), obj_y(0), obj_speed(0) {}
    GameObject(int8_t x, int8_t y): obj_x(x), obj_y(y), obj_speed(0) {}
    GameObject(int8_t x, int8_t y, int8_t speed): obj_x(x), obj_y(y),
obj_speed(speed) {}


    int8_t x() const { return obj_x; }
    int8_t y() const { return obj_y; }
    int8_t speed() const { return obj_speed; }

    bool setX(int8_t x) {
      if ((x < 0) || (x > WIDTH))
        return false;
      obj_x = x;
      return true;
```

```
    }

    bool setY(int8_t y) {
      if ((y < 0) || (y > HEIGHT))
        return false;
      obj_y = y;
      return true;
    }

    bool setSpeed(int8_t speed) {
      obj_speed = speed;
      return true;
    }


    bool collide(const GameObject &o) {
      return ((obj_x == o.x()) && (obj_y == o.y())) ? true : false;
    }
};
```

- **Chicken** :

Clasa Chicken reprezinta clasa gainii (jucatorul) ce implementeaza miscarea acesteia pe orizontala, la stanga → moveLeft() si la dreapta → moveRight().

```
class Chicken: public GameObject
{
  public:

    bool moveRight() {
      obj_x++;
      if (obj_x >= WIDTH) {
        obj_x--;
        return false;
      }
      return true;
    }

    bool moveLeft() {
      obj_x--;
      if (obj_x < 0) {
        obj_x++;
        return false;
      }
      return true;
    }
} chicken;
```

- **Enemy** :

Clasa Enemy prezinta 2 variabile folosite pentru a reda starea si viata inamicilor. Boolean-ul de stare

este folosit pentru a repreznta dinamic inamicul(true → ENEMY1 ; false → ENEMY2), iar cel de viata pentru a determina daca inamicul a fost eliminat sau nu. De asemenea, clasa prezinta setteri pentru variabilele descrise si o functie de move pentru miscarea efectiva a inamicilor.

```
class Enemy: public GameObject
{
  private:
    bool enemy_alive; // shows whether enemy is alive
    bool enemy_state; // enemy's current state for animation purpose

  public:
    Enemy(): GameObject(), enemy_alive(false), enemy_state(false) {}

    void setAlive(bool alive) { enemy_alive = alive; }
    bool alive() { return enemy_alive; }

    void setState(bool state) { enemy_state = state; }
    bool state() { return enemy_state; }

    bool move() {
      obj_x += obj_speed;
      enemy_state = !enemy_state;
      if ((obj_x < 0) || (obj_x >= WIDTH)) {
        obj_x -= obj_speed;
        return false;
      }
      return true;
    }
} enemy[MAX_ENEMY];
```

- **Bullet** :

Clasa Bullet contine o variabila ce tine cont de activitatea proiectilului (true → BULLET UP / DOWN; false → None). Si acesta clasa contine un setter pentru boolean-ul de activitate si o functie pentru miscarea efectiva a proiectilului.

```
class Bullet: public GameObject
{
  private:
    bool bullet_active;

  public:
    Bullet(): GameObject(), bullet_active(false) {}

    void setActive(bool active) { bullet_active = active; }
    bool active() { return bullet_active; }

    bool move() {
      obj_y += obj_speed;
      if ((obj_y < 0) || (obj_y >= HEIGHT)) {
        obj_y -= obj_speed;
```

```
        bullet_active = false;
        return false;
      }
      return true;
    }
} chicken_bullet, enemy_bullet[MAX_ENEMY];
```

Funcții

- **buttonPress()** :

Aceasta functie transforma input-ul hardware in input software si ajuta la realizarea unei desfasurari simple si interactive a jocului. Joystick-ul prezinta valori intre 0 si 1024 pe axa Ox, folosite pentru a determina in ce directie a fost miscat joystick-ul. Threshold-ul este la 512 (mijlocul axei → pozitia initiala), insa am adaugat o margine de eroare de 128 pentru a fi mai responsive. Asadar, cand valoare scade sub acest threshold inseamna ca jucatorul se misca la stanga, la dreapta in sens contrar.

```
byte buttonPress() {
  int xVal = analogRead(joy_X);
  int buttonVal = digitalRead(joy_B);

  int threshold = 512;

  if (xVal < threshold - 128) return X_LEFT;
  if (xVal > threshold + 128) return X_RIGHT;
  if (buttonVal == LOW) return JOY_PRESS;

  return DO_NOTHING;
}
```

- **refreshScreen()** :

Aceasta functie este esentiala pentru derularea jocului intrucat deseneaza toate elementele jocului la fiecare refresh de ecran.Asadar, la fiecare refresh se curata buffer-ul ecranului, se deseneaza mai intai glontul gainii si inamicii deorece interactiunea intre acestea prezinta elementul principal al jocului. Urmatorul element desenat este glontul inamicului in cazul in care acesta trage si la final dupa trimiterea bufferului, se afiseaza gaina deoarece are valoarea 0, iar lcd.print() ar fi tratat acest char ca si EOL(End Of Line).

```
void refreshScreen() {
  bool chicken_bullet_on = false;

  for (byte i = 0; i < HEIGHT / 2; i++) {
    for (byte j = 0; j < WIDTH; j++)
      screen_buff[i][j] = ' ';
    screen_buff[i][WIDTH] = '\0';
  }
```

```
  if (chicken_bullet.active()) {
    if ((chicken.x() == chicken_bullet.x()) && (chicken_bullet.y() == 2)) {
      screen_buff[chicken_bullet.y() / 2][chicken_bullet.x()] =
CHICKEN_BULLET;
      chicken_bullet_on = true;
    } else {
      screen_buff[chicken_bullet.y() / 2][chicken_bullet.x()] =
chicken_bullet.y() % 2 ? BULLET_DOWN : BULLET_UP;
    }
  }

  for (byte i = 0; i < enemy_active; i++) {
    if (enemy[i].alive()) {
      screen_buff[enemy[i].y() / 2][enemy[i].x()] = enemy[i].state() ?
ENEMY1 : ENEMY2;
    }
  }

  bool enemy_bullet_on = false;
  for (byte i = 0; i < enemy_active; i++) {
    if (enemy_bullet[i].active()) {
      enemy_bullet_on = false;
      for (int j = 0; j < enemy_active; j++) {
        if ((enemy[j].x() == enemy_bullet[i].x()) && (enemy_bullet[i].y() ==
1) && (enemy[i].alive())) {
          screen_buff[enemy_bullet[i].y() / 2][enemy_bullet[i].x()] = enemy[
i].state() ? ENEMY1_BULLET : ENEMY2_BULLET;
          enemy_bullet_on = true;
        }
      }
      if (!enemy_bullet_on) {
        if ((chicken.x() == enemy_bullet[i].x()) && (enemy_bullet[i].y() ==
2)) {
          screen_buff[enemy_bullet[i].y() / 2][enemy_bullet[i].x()] =
CHICKEN_BULLET;
          chicken_bullet_on = true;
        } else {
          screen_buff[enemy_bullet[i].y() / 2][enemy_bullet[i].x()] =
enemy_bullet[i].y() % 2 ? BULLET_DOWN : BULLET_UP;
        }
      }
    }
  }

  for (byte i = 0; i < HEIGHT / 2; i++) {
    lcd.setCursor(0, i);
    lcd.print(screen_buff[i]);
  }

  if (!chicken_bullet_on) {
```

```
    lcd.setCursor(chicken.x(), chicken.y() / 2);
    lcd.write(byte(CHICKEN));
  }
}
```

- **startLevel(byte l)** :

Aceasta functie primeste ca parametru nivelul curent si dupa acesta se implementeaza factorii de dificulatate ai jocului precum numarul de inamici activi ce creste o data la 3 nivele, viteza inamicilor ce creste o data la 2 nivele si probabilitatea de tragere ce creste cu fiecare nivel(functionalitatea este descrisa mai in detaliu la functia de loop()).

```
void startLevel(byte l) {
  level = l;
  enemy_active = 6 + ((level - 1) / 3);
  if (enemy_active > MAX_ENEMY) {
    enemy_active = MAX_ENEMY;
  }

  chicken.setX(WIDTH / 2);
  chicken.setY(3);
  chicken_bullet.setX(WIDTH / 2);
  chicken_bullet.setY(3);
  chicken_bullet.setActive(false);

  for (byte i = 0; i < enemy_active; i++) {
    enemy[i].setX(i);
    enemy[i].setY(0);
    enemy[i].setSpeed(1);
    enemy[i].setAlive(true);
    enemy[i].setState(false);
    enemy_bullet[i].setActive(false);
  }

  frame = 0;
  enemy_frame_delay = 6 - level / 2;
  if (enemy_frame_delay < 1)
    enemy_frame_delay = 1;
  fire_chance = 110 - level * 10;
  if (fire_chance < 10)
    fire_chance = 10;
  enemy_left = enemy_active;

  lcd.clear();
  lcd.setCursor(4, 0);
  lcd.print("Level: ");
  lcd.print(level);
  delay(1000);
  for (uint8_t y = 0; y < 2; y++) {
    for (uint8_t x = 0; x < 16; x++) {
      lcd.setCursor(x, y);
```

```
      lcd.write(byte(ENEMY1));
      lcd.write(byte(ENEMY2));
      delay(75);
      lcd.setCursor(x, y);
      lcd.print(" ");
    }
    break;
  }
  lcd.clear();
}
```

- **gameOver()** :

Aceasta functie afiseaza pe ecran un mesaj de final de joc (“G A M E O V E R.”), scorul jucatorului (“Score: <nr> ”) si asteapta ca jucatorul sa apese joystick-ul pentru a reincerca sa isi invinga scorul (“Shoot to try again!”). Dupa ce este apasat joystick-ul, se modifica flag-ul de reset pentru a fi apelata functia de resetGame().

```
void gameOver() {
  lcd.clear();
  lcd.setCursor(4, 0);
  lcd.print("G A M E");
  lcd.setCursor(4, 1);
  lcd.print("O V E R.");

  playTone(1500, 50);
  playTone(1450, 50);
  playTone(1400, 50);
  playTone(1350, 50);
  playTone(1300, 50);
  playTone(1250, 50);
  playTone(1200, 50);
  playTone(1150, 50);
  playTone(1100, 50);
  playTone(1050, 50);
  playTone(1000, 250);

  for (uint8_t y = 0; y < 2; y++) {
    for (uint8_t x = 0; x < 16; x++) {
      lcd.setCursor(x, y);
      lcd.write(byte(ENEMY1));
      lcd.write(byte(ENEMY2));
      delay(75);
      lcd.setCursor(x, y);
      lcd.print(" ");
    }
  }

  lcd.clear();
  lcd.setCursor(4, 0);
  lcd.print("Score:");
```

```
  lcd.print(score);
  delay(1000);

  for (uint8_t y = 0; y < 2; y++) {
    for (uint8_t x = 0; x < 16; x++) {
      lcd.setCursor(x, y);
      lcd.write(byte(ENEMY1));
      lcd.write(byte(ENEMY2));
      delay(75);
      lcd.setCursor(x, y);
      lcd.print(" ");
    }
    break;
  }

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Shoot to");
  lcd.setCursor(6, 1);
  lcd.print("try again!");

  while (buttonPress() != JOY_PRESS);
  for (uint8_t y = 0; y < 2; y++) {
    for (uint8_t x = 0; x < 16; x++) {
      lcd.setCursor(x, y);
      lcd.write(byte(ENEMY1));
      lcd.write(byte(ENEMY2));
      delay(50);
      lcd.setCursor(x, y);
      lcd.print(" ");
    }
  }

  reset = true;
}
```

- **resetGame()** :

Aceasta functie are rolul de a reseta hard jocul adica reseteaza si scorul si nivelul jucatorului si pune flag-ul de resetare inapoi pe false. In aceasta functie se apeleaza si startLevel() deci se reseteaza in intermediul functiei respective restul variabilelor.

```
void resetGame() {

  lcd.clear();
  lcd.print("Resetting...");
  delay(1000);

  score = 0;
  level = 1;
  reset = false;
```

```
  startLevel(level);
}
```

- **playTone(int frequency, int duration)** :

Aceasta functie este o functie special creata pentru buzzer-ul meu deoarece starile de LOW si HIGH sunt inversate. Astfel, functia primeste ca parametru o frecventa si o durata pentru care buzzer-ul ramane activ emitand un sunet la frecventa primita, aceasta fiind procesata ca o perioada de delay intre activarea si dezactivarea buzzer-ului.

```
void playTone(int frequency, int duration) {
  int period = 1000000 / frequency;
  int halfPeriod = period / 2;
  unsigned long startTime = millis();
  while (millis() - startTime < duration) {
    digitalWrite(buzz_P, LOW);
    delayMicroseconds(halfPeriod);
    digitalWrite(buzz_P, HIGH);
    delayMicroseconds(halfPeriod);
  }
}
```

- **setup()** :

Aceasta functie are rolul de a initializa toate piesele hardware folosite, precum display-ul LCD, pinii joystick-ului si al buzzer-ului precum si caracterle speciale folosite pentru afisaj. Pentru a simula un joc retro, se realizeaza o animatie de inceput cu numele jocului. De asemenea, se initializeaza scorul cu 0, se activieaza functia de random cu un seed aleatoriu si trec pinul buzzer-ului in starea HIGH pentru a fi oprit. Nu in ultimul, rand se apeleaza functia startLevel(byte l) cu parametrul l = 1 pentru a incepe primul nivel al jocului.

```
void setup() {
  lcd.begin(16, 2);
  lcd.print("Chicken Invaders");
  lcd.setCursor(3, 1);
  lcd.print("The Game");
  delay(2000);
  for (uint8_t y = 0; y < 2; y++) {
    for (uint8_t x = 0; x < 16; x++) {
      lcd.setCursor(x, y);
      lcd.write(byte(ENEMY1));
      lcd.write(byte(ENEMY2));
      delay(75);
      lcd.setCursor(x, y);
      lcd.print(" ");
    }
  }
  lcd.createChar(CHICKEN, chicken_char);
  lcd.createChar(BULLET_UP, bullet_up_char);
  lcd.createChar(BULLET_DOWN, bullet_down_char);
```

```
  lcd.createChar(CHICKEN_BULLET, chicken_bullet_char);
  lcd.createChar(ENEMY1, enemy1_char);
  lcd.createChar(ENEMY2, enemy2_char);
  lcd.createChar(ENEMY1_BULLET, enemy1_bullet_char);
  lcd.createChar(ENEMY2_BULLET, enemy2_bullet_char);
  score = 0;
  randomSeed(analogRead(1));
  pinMode(joy_B, INPUT_PULLUP);
  pinMode(buzz_P, OUTPUT);
  digitalWrite(buzz_P, HIGH);
  startLevel(1);
}
```

- **loop()** :

Aceasta functie ruleaza continuu dupa cum ii spune si numele si are rolul de a sustine rularea efectiva a jocului si se ocupa de procesarea input-urilor si implementeaza logica jocului. Mai intai se verifica daca flag-ul de resetare a fost setat, caz in care se reseteaza jocul. Daca nu, se proceseaza input-ul de la joystick si pe baza input-ului oferit se misca gaina, respectiv si proiectilul lansat de ea. De asemenea, se misca inamicii intr-un mod aleator si se apeleaza functia de random pentru a decide daca inamicii trag sau nu. random(fire_chance) ia un numar intre 0 si fire_chance - 1 si daca numarul returnat de catre functie este 0 atunci inamicul trage. Deci sansa inamicilor de a trage este de 1 / fire_chance. In ultimul rand se apeleaza functia de refreshScreen() pentru afisaj si daca nu mai este niciun inamic in viata se trece la urmatorul nivel.

```
void loop() {
  if (reset) {
    lcd.clear();
    resetGame();
  }

  switch(buttonPress()) {
    case X_RIGHT: {
      chicken.moveRight();
      break;
    }
    case X_LEFT: {
      chicken.moveLeft();
      break;
    }
    case JOY_PRESS: {
      if (!chicken_bullet.active()) {
        chicken_bullet.setX(chicken.x());
        chicken_bullet.setY(chicken.y());
        chicken_bullet.setSpeed(-1);
        chicken_bullet.setActive(true);
        playTone(7000, 10);
        playTone(6000, 10);
        playTone(5000, 10);
        playTone(4000, 10);
        playTone(3000, 10);
```

```
        playTone(2000, 10);
        playTone(1000, 10);
        playTone(500, 5);
      }
      break;
    }
    default:
      break;
  }

  if (chicken_bullet.active())
    chicken_bullet.move();

  for (byte i = 0; i < enemy_active; i++) {
    if (enemy_bullet[i].active()) {
      enemy_bullet[i].move();
      if (enemy_bullet[i].collide(chicken))
        gameOver();
    }

    if (!(frame % enemy_frame_delay))
      enemy[i].move();

    if ((enemy[i].collide(chicken_bullet)) && (chicken_bullet.active()) && (
enemy[i].alive())) {
      enemy[i].setAlive(false);
      score += 10 * level;
      enemy_left--;

      playTone(3000, 10);
      playTone(2750, 10);
      playTone(2500, 10);
      playTone(2250, 10);
      playTone(2000, 10);
      playTone(1750, 10);
      playTone(1500, 10);
      playTone(1000, 5);
    }

    if ((!random(fire_chance)) && (!enemy_bullet[i].active()) && (enemy[i].
alive())) {
      enemy_bullet[i].setX(enemy[i].x());
      enemy_bullet[i].setY(enemy[i].y() + 1);
      enemy_bullet[i].setSpeed(1);
      enemy_bullet[i].setActive(true);

      playTone(6000, 10);
      playTone(5500, 10);
      playTone(5000, 10);
      playTone(4500, 10);
      playTone(4000, 10);
```

```
      playTone(3500, 10);
      playTone(3000, 10);
      playTone(2000, 5);
    }
  }

  if ((!(frame % enemy_frame_delay)) && ((enemy[0].x() == 0) || (enemy[
enemy_active - 1].x() == WIDTH - 1))) {
    for (byte i = 0; i < enemy_active; i++)
      enemy[i].setSpeed(-enemy[i].speed());
  }


  refreshScreen();
  frame++;
  delay(FRAME_DELAY);


  if (!enemy_left)
    startLevel(level + 1);
}
```

Rezultate Obţinute

Demo

Aici se poate gasi un demo al proiectului → Chicken Invaders.

Poza finala

Concluzii

Am aprofundat o multitudine de cunostiinte pe parcursul realizarii proiectului, cunostiinte dobandite in cadrul laboratorului de PM, mai specific laboratoarele despre:

- PWM → Buzzer
- ADC → Joystick-ului
- I2C → Display LCD

Am inteles cum se realizeaza un proiect incepand de la alegerea pieselor hardware, proiectarea

diagramei bloc si a schemei electrice pana la dezvoltarea software-ului folosit si realizarea unei documentatii detaliate.

Jurnal

Dupa ce mi-am comandat piesele de pe cleste.ro am incercat sa-mi conectez display-ul LCD 2004 la Arduino Uno si pentru un motiv sau altul nu am reusit sa-l conectez. Asa ca am decis sa imi comand alt display de pe OptimusDigital deoarece am auzit de la colegii mei ca ajung rapid piesele si erau si usor de ridicat deoarece au sediul relativ aproape de facultate, insa a trebuit sa iau unul de 1602 pentru ca nu mai aveau in stoc 2004.. Si asa mi-a fost intarziat tot proiectul. Planuiam sa implementez si niste led-uri, butoane si sa le lipesc pe ceva insa nu am mai avut timp.. Oricum proiectul nu este in stare finala so expect more updates!

Download

Aici se poate gasi arhiva proiectului cu cod, data sheet-uri si schemele implementate pentru proiect:

- chicken_invaders.zip

Bibliografie/Resurse

- Instructables - Space Invaders
- Arduino - Space Invaders