Pisasweeper

Autor: *Pavel Diana-Larisa* , *334CC*

Introducere

Ce este Pisaswipper?

Pisasweeper este un joc inspirat după celebrul Minesweeper, care are ca scop salvarea pisicilor, din satul Pisa, de o furtună devastatoare. Ideea inițială a proiectului a fost de a realiza un proiect simplu, la care să lucrez din plăcere si pasiune, iar atunci când am căutat idei de proiect pe OCW și am văzut implementat jocul Minesweeper, am stiut că aceasta este tema perfectă pentru proiectul meu, iar după ce am făcut și implementarea software, mi-a venit în minte această poveste pentru joc.

Motivația pentru acest proiect

Acest proiect este util pentru alții pentru că oferă oportunitatea de a învăța despre programarea Arduino și de a interacționa cu componente hardware, dar poate fi și o oportunitate, pentru cei care nu cunosc regulile acestui joc, de a descopri frumusețea jocului și poate dobândirea unor tactici de joc. Pentru mine, acest proiect reprezintă o modalitate excelentă de a aplica cunoștințele în programare și electronică într-un mod practic și creativ. De asemenea, construirea și programarea acestui joc oferă o provocare tehnică interesantă și mă ajută să dezvolt abilități noi în domeniul IoT și al dispozitivelor integrate. În plus, mă joc acest joc aproape zilnic și acest proiect mă va ajuta să desopăr și modul în care acest joc funcționează, înțelegându-l mai bine și deci dezvoltând noi tehnici și strategii de joc.

Descriere generală

Schema bloc

Cum decurge jocul?

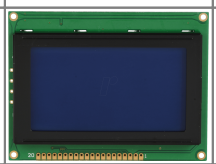
Input: Un buton pentru descoperirea unei zone, iar joystick-ul, pe lângă direcții, are incorporat un buton, pe care îl voi folosi pentru a pune umbrelele.
Output: Un LCD penrtu afișarea jocului și un Buzzer care va reda diverse sunete în funcție de parcursul jocului.
Descriere: În primul rând, jucătorul are un meniu principal, de unde poate selecta:

1. Să aleagă **nivelul de dificultate** al jocului: Aici are posibilitatea de a alege între **3** nivele de dificultate (**ușor**, **mediu** sau **greu**) sau poate alege să revină la meniul principal.
2. Să citească **povestea jocului**, după care, poate selecta să revină la meniul principal.
3. Să se citească **instrucțiunile TO DO** *(Voiam sa fac creditele, însă în timp ce scriam documentul, mi-am dat seama că ar avea mai mult sens să fac o secțiune unde explic controalele și asocierea cu minesweeper)*

După ce jucătorul își selectează nivelul de dificultate preferat, vine și momentul în care locuitorii satului Pisa trebuiesc salvați . Pe ecranul LCD va apărea harta satului, iar jucătorul va trebui să deblocheze zone de pe hartă. Fiecare zonă poate conține o pisică, un număr care indică câte pisici sunt în jurul zonei respective sau pajiște. Scopul jucătorului este de a adăpostii toate pisicile (*bombele*) de furtuna primejdioasă, folosindu-se de umbrele ☂ (*steguleț*e) și fără să descopere vreo zonă în care există pisici, deoarece furtuna va ajunge la acea pisică și sărăcuța va ajunge udă și tristă , ceea ce rezultă în sfârșitul negativ al jocului. Dacă jucătorul reușește să adăpostească corect toate pisicile sau să descopere toate zonele în care nu sunt pisici, locuitorii satului vor fi salvați , ceea ce conduce la un sfârșit bun al jocului. Dacă jucătorul vrea să mai ajute încă o dată satul Pisa, acesta va trebui să apese pe butonul de **RESTART**. Pisicile locuitoare le sunt recunoscătoare tuturor jucătorilor pentru ajutorul oferit, iar prin acest document, ele vă mulțumesc pentru timpul acordat .

Hardware Design

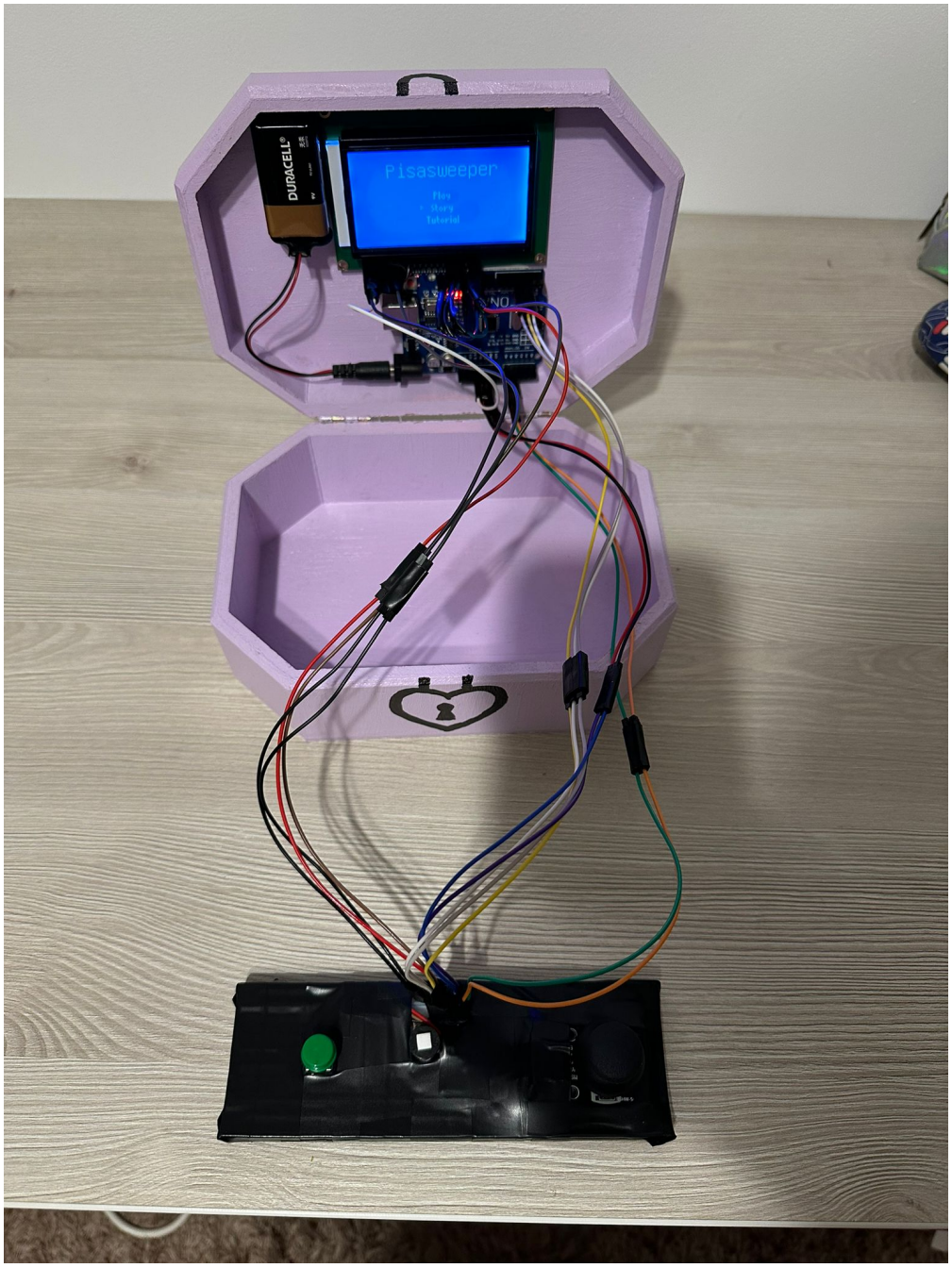
Listă de piese

Componentă	Cantitate	Imagine	Descriere
Arduino Uno R3	1		Placă cu care implementez logica și afișez parcursul jocului.
128×64 Graphic LCD	1		Afișează tabla de joc, cursorul, meniul și mesajele.
Joystick Module	1		Controlează cursorul pe tablă și plasează flag-ul.

Tactile Push Button	1		Descoperă o celulă.
Active Buzeer	1		Scoate sunete la apăsarea butoanelor și la sfârșitul jocului.
1K ohm Resistor	1		Folosită pentru funcționarea butonului.
9V Battery	1		Folosită pentru alimentarea placuței.
Battery Connector	1		Folosită pentru alimentarea placuței.

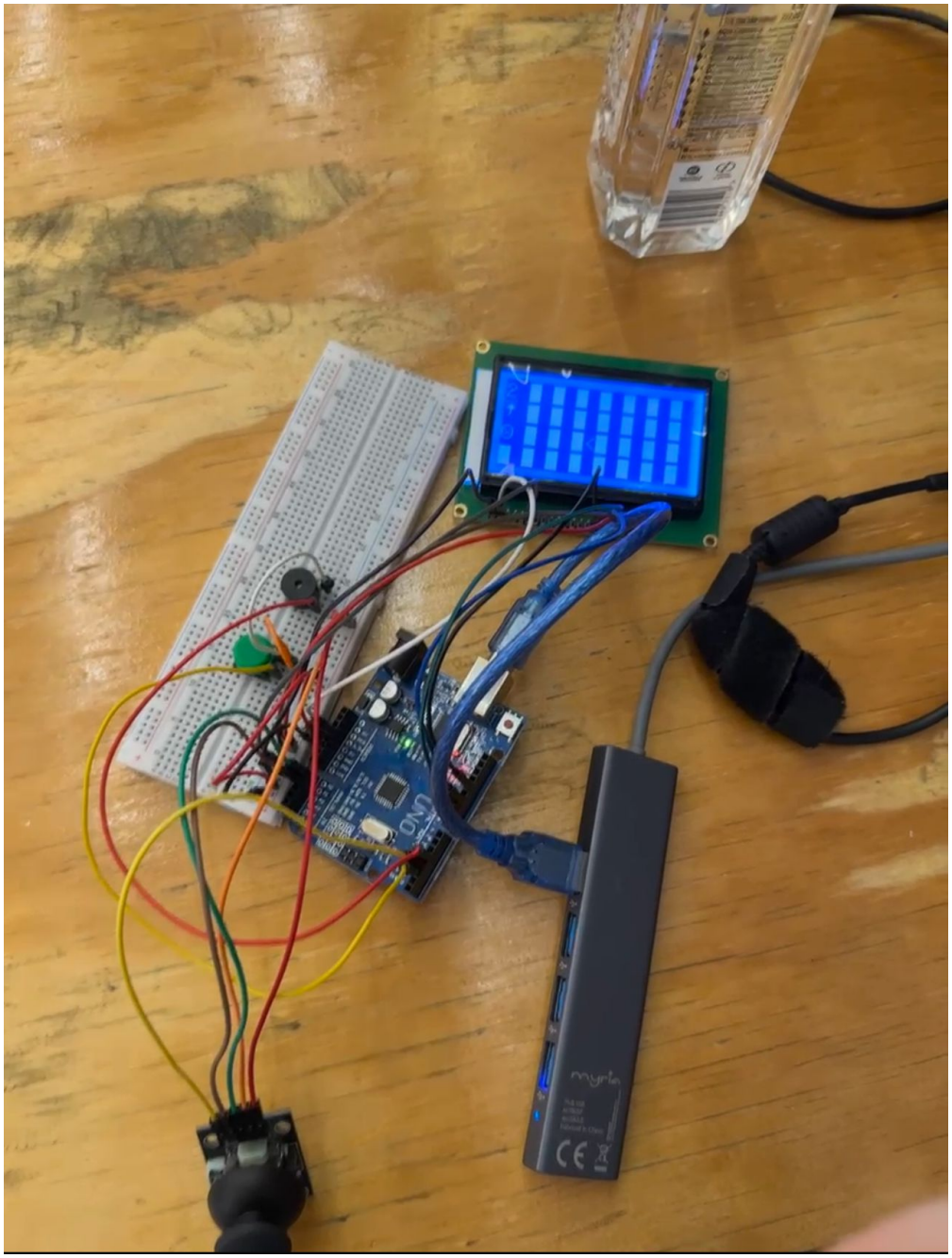
Scheme electrice

Implementare fizică

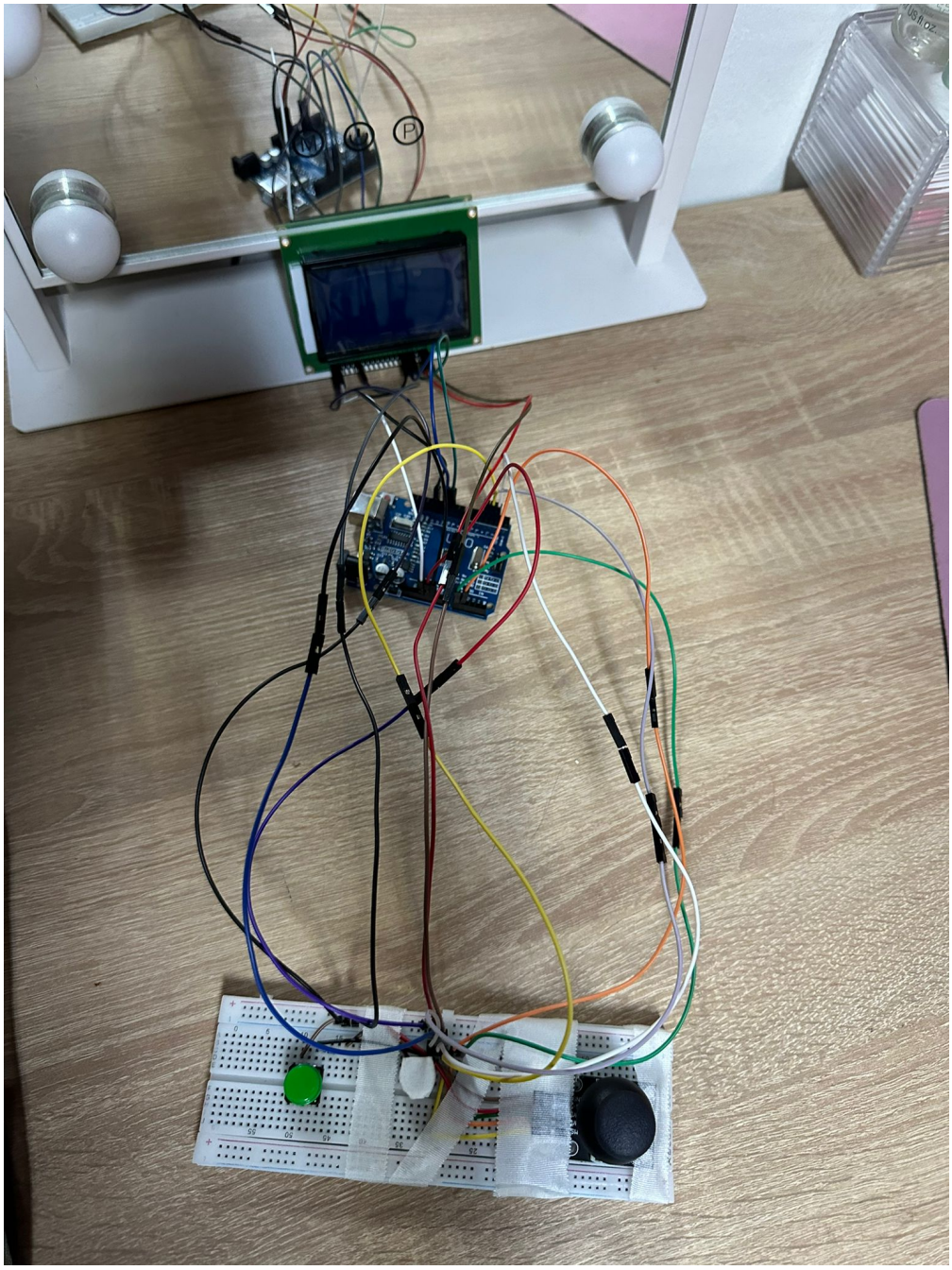


Progres personal

Implementarea fizică a componentelor a mers destul de rapid. Am urmarit tutorialele pe care le-am pus în secțiunea de Bibliografie/Resurse. În urma tutorialelor hardware, am obținut următoarea configurație:



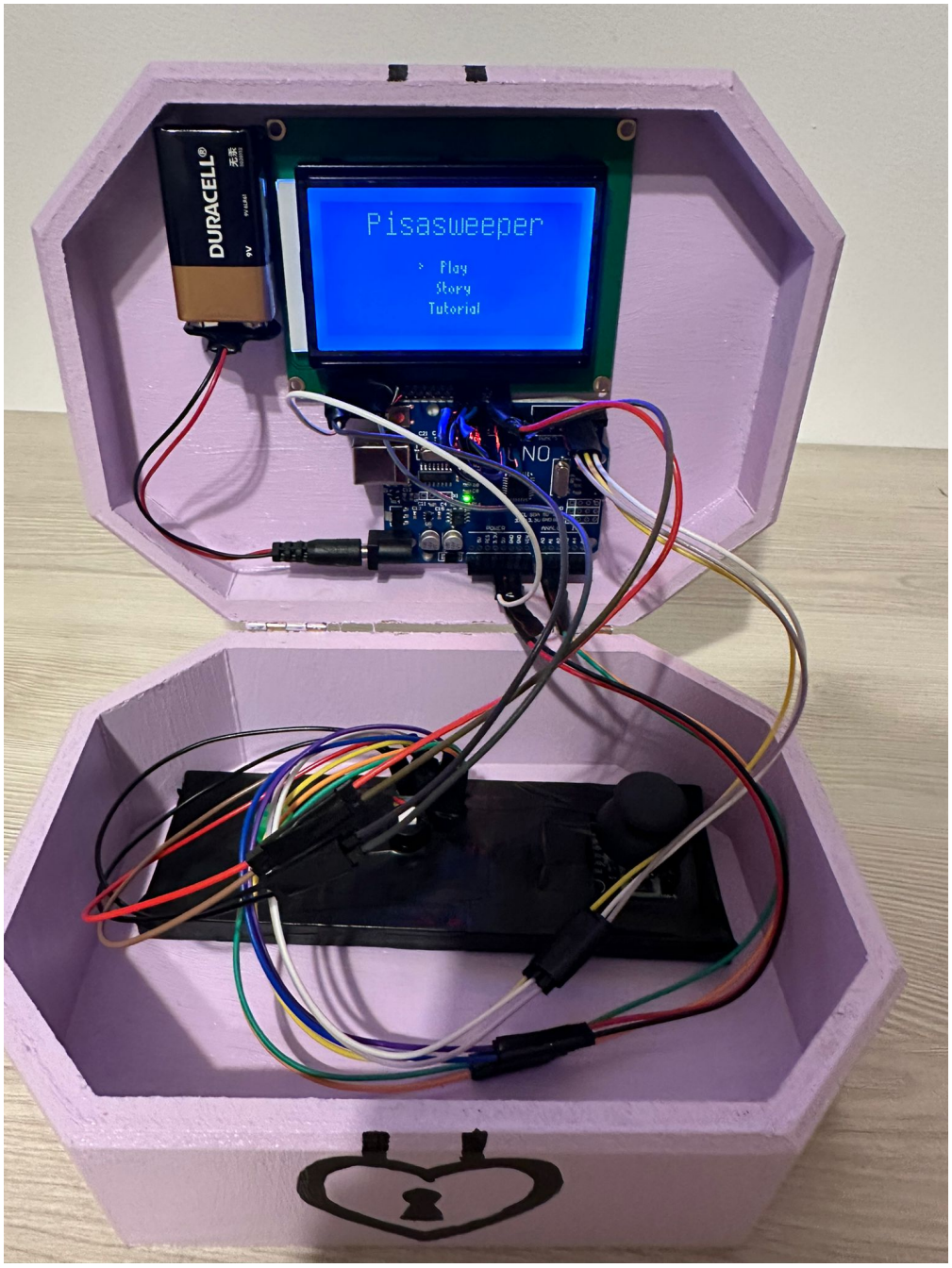
Pe parcurs am decis să schimb modul în care sunt legate buzzer-ul și butonul verde, și să adaug o rezistență la buton. Am încercat să-mi organizez mai bine ordinea cablurilor și să imit cât de mult posibil un controller (am pus cablurile care pornesc din breadboard și ajung în LCD și placuță cât mai central-față posibil).



Această configurație va rămâne finală, singurul lucru pe care aș fi vrut să îl mai adaug ar fi o baterie de 9V la placuță și să încerc să fac jocul portabil, ceea ce am și făcut într-un final. Pinii folosiți în final sunt:

- **Pinii Analogici A0 și A1** pentru citirea direcției pe axele Ox și Oy
- **Pinii Digitali 2,3 și 4** pentru butonul verde de selecție a opțiunilor din meniu și descoperire a zonelor din timpul jocului (pinul 2), butonul folosit pentru a plasa umbrelele de pe joystick (SW) (pinul 3) și cel din urmă pentru buzzer(pinul 4).
- **Piniul Digitali 10** pentru conectarea firului CS (chip select) la RS, ceea ce înseamnă Register Select, al LCD-ului.
- **Piniul Digitali 11** pentru conectarea Master In Slave Out MOSI al protocolului SPI la pinul RW (read/write) de pe LCD.
- **Piniul Digitali 13** pentru conectarea ceasului serial „SCK” în pinul E (Enable Signal) al LCD-ului

Hardware-ul final arată așa:



Am vrut să fac jocul portabil și să arate plăcut pentru utilizator. Prin această implementare, pot să continui proiectul, prin a face multiple jocuri de tip consolă, ceea ce reprezintă un avantaj și o lasă loc pentru creativitate. Am desenat pe cutie câteva detalii estetice:

Software Design

Mediul de dezvoltare folosit a fost **Arduino IDE** și pentru afișarea pe LCD a jocului, m-am folosit de librăria **U8g2**.

Diagrama de stări a jocului

Meniuri

În primul rând, pentru a realiza un **meniu** funcțional, am avut nevoie de joystick pentru a putea naviga cursorul pe direcția *sus-jos*, de un buton pentru a putea selecta opțiunile (am ales butonul verde, care coincide cu butonul de descoperire a unei zone de pe hartă din timpul jocului). Pe parcursul programului, m-am folosit de fonturile u8g2_font_unifont_t_symbols și u8g2_font_squeezed_r6_tr și de funcțiile u8g2.setFont(), u8g2.drawUTF8() (asociată fontului de simboluri) și u8g2.drawStr() (asociată fontului normal) pentru a putea reda pe LCD elementele care alcătuiesc jocul. Pentru fiecare pagină din meniu, care conține mai mult de o posibilitate de navigare, folosesc variabile diferite pentru numerotarea butoanelor (nr_butt) și identificarea poziției cursorului (menuCrusor).

```
int menuCursor = 33;
int nr_butt = 0;
int val = 0;
void Menu() {
  u8g2.firstPage();
  do {
    //joystick
    val = analogRead(VRY_PIN);
    if(val < 400 && nr_butt > 0) {
      menuCursor-=10;
      nr_butt--;
      delay(200);
    }
    if(val > 800 && nr_butt < 2) {
      menuCursor+=10;
      nr_butt++;
      delay(200);
    }
    //select button
    int select = digitalRead(REVEAL_BUTTON);
    if(select == HIGH) {
      if (nr_butt == 0) {
        gameStatus = 5; //select level difficulty
      } else if (nr_butt == 1) {
        gameStatus = 4; //story
      } else if (nr_butt == 2) {
        gameStatus = 6; //tutorial
      }
      delay(200);
    }
    u8g2.setFont(u8g2_font_unifont_t_symbols);
    u8g2.drawUTF8(22,15, "Pisasweeper");
    u8g2.setFont(u8g2_font_squeezed_r6_tr);
    u8g2.drawStr(59, 33, "Play");
    u8g2.drawStr(57, 43, "Story");
```

```
    u8g2.drawStr(53, 53, "Tutorial");
    u8g2.drawStr(48, menuCursor, ">");
  } while (u8g2.nextPage());
}
```

Similar, am implementat și **meniul** secundar pentru **selecția nivelului** de dificultate

```
int nr_butt2 = 0;
int menuCursor2 = 27;
void LevelSelect() {
  u8g2.firstPage();
  do {
    //joystick
    val = analogRead(VRY_PIN);
    if(val < 400 && nr_butt2 > 0) {
      menuCursor2-=10;
      nr_butt2--;
      delay(200);
    }
    if(val > 800 && nr_butt2 < 3) {
      menuCursor2+=10;
      nr_butt2++;
      delay(200);
    }
    //select button
    int select = digitalRead(REVEAL_BUTTON);
    if(select == HIGH) {
      if (nr_butt2 == 0) {
        gameStatus = 0;
        difficulty = 4; //Easy
        umbrellasLeft = difficulty;
      } else if (nr_butt2 == 1) {
        gameStatus = 0;
        difficulty = 5; //Medium
        umbrellasLeft = difficulty;
      } else if (nr_butt2 == 3) {
        gameStatus = 3;
        nr_butt2 = 0; //Back Button
        menuCursor2 = 25; //Reset cursor for the next visit
      } else if (nr_butt2 == 2) {
        gameStatus = 0;
        difficulty = 7; //Hard
        umbrellasLeft = difficulty;
      }
      delay(200);
    }
    u8g2.setFont(u8g2_font_squeezed_r6_tr);
    u8g2.drawUTF8(22,14, "Please select the difficulty:");
    u8g2.drawStr(59, 27, "Easy");
    u8g2.drawStr(53, 37, "Medium");
    u8g2.drawStr(58, 47, "Hard");
```

```
    u8g2.drawStr(59, 57, "Back");
    u8g2.drawStr(48, menuCursor2, ">");
  } while (u8g2.nextPage());
}
```

În functia principala (loop), metoda de decizie a paginilor afișate a fost implementată astfel

```
if (gameStatus == 3) {
    Menu();
  } else if (gameStatus == 4) {
    Story();
  } else if (gameStatus == 5) {
    LevelSelect();
  }else if (gameStatus == 1){
    BoardGame();
  } else if (gameStatus == -1) {
    LostGame();
  } else if (gameStatus == 2) {
    WonGame();
  } else if (gameStatus == 6) {
    Tutorial();
  }
```

Jocul

Pentru a implementa jocul, m-am folosit de:

- Matricea 4×8 cats[][] cu care reținpoziția bombelor.
- Matricea 4×8 mapOfPisaVillage[][] cu care rețintoate elementele ce alcătuiesc harta jocului
- Variabilele CursorX și CursorY cu care reținpoziția pe hartă a cursorului.
- Variabila revealed cu care țin cont de numărul de zone care au fost descoperite de jucător.
- Variabila umbrellasLeft pentru a limita numărul de umbrele pe care jucătorul le poate pune pe hartă.
- Variabila difficulty pentru a avea un număr diferit de pisici în funcție de dificultate.

Din funcția **LevelSelect()** prezentată anterior, se observă că numărul pentru fiecare nivel de dificultate avem un număr diferit de pisici (Easy = 4 pisici, Medium = 5 pisici, Hard = 7 pisici). Aceste numere au fost alese cu scopul de a corescpunde procentului de densitate al minelor prezent în nivelele de pe Minesweeper aferente.
Pentru a genera aleator pisicile, m-am folosit de randomSeed() și de random(). În momentul în care jucătorul alege nivelul de dificultate dorit din **LevelSelect()**, statusul jocului devine 0, care are scopul de a fi o stare intermediară între meniu și funcția **BoardGame()**, în care se întamplă afișarea completă a jocului. Această generare are loc în funcția principală (loop()) astfel:

```
if (gameStatus == 0) {
    //initialize the map
    for (int i = 0; i < ROWS; i++) {
      for (int j = 0; j < COLUMNS; j++) {
        mapOfPisaVillage[i][j] = NOT_REVEALED;
```

```
      }
    }
    //generate the cats randomly
    for (int i = 0; i < ROWS; i++) {
      for (int j = 0; j < COLUMNS; j++) {
        cats[i][j] = 0;
      }
    }
    randomSeed(analogRead(A2));
    for (int i = 0; i < difficulty; i++) {
      int row = random(ROWS);
      int col = random(COLUMNS);
      if (cats[row][col] == 1) {
        i--;
      } else {
        cats[row][col] = 1;
      }
    }
    gameStatus = 1; //change game status to PLAYING
  }
```

Funcția **BoardGame()** este cea mai complexă funcție din aplicație și conține implementările pentru logica controaleor și a afișării elementelor ce alcătuiesc harta.

```
void BoardGame(){
  int xValue = 0 ;
  int yValue = 0 ;
  u8g2.firstPage();
  do {
    //JOYSTICK
    xValue = analogRead(VRX_PIN);
    yValue = analogRead(VRY_PIN);
    if(xValue > 800 && CursorX < 110) {
      CursorX+=16;
      delay(200);
    }
    if(xValue < 400 && CursorX > 12) {
      CursorX-=16;
      delay(200);
    }
    if(yValue > 800 && CursorY < 60) {
      CursorY+=16;
      delay(200);
    }
    if(yValue < 400 && CursorY > 13) {
      CursorY-=16;
      delay(200);
    }
    //start of board indexing
    int x = 0;
    int y = 13;
```

```
    char buf[7];
    for (int i = 0; i < ROWS; i++) {
      for (int j = 0; j < COLUMNS; j++) {
        if(mapOfPisaVillage[i][j] == NOT_REVEALED) {
          u8g2.setFont(u8g2_font_unifont_t_symbols);
          u8g2.drawUTF8(x + 16*j, y + 16*i, "■");
        } else if (mapOfPisaVillage[i][j] == UMBRELLA) {
          u8g2.setFont(u8g2_font_unifont_t_symbols);
          u8g2.drawUTF8(x + 16*j, y + 16*i, "☂");
        } else if (mapOfPisaVillage[i][j] == BOMB) {
          u8g2.setFont(u8g2_font_unifont_t_0_76);
          u8g2.drawUTF8(x + 16*j, y + 16*i, "☹");
        } else if (mapOfPisaVillage[i][j] == CAT) {
          u8g2.setFont(u8g2_font_unifont_t_0_76);
          u8g2.drawUTF8(x + 16*j, y + 16*i, "☺");
        } else if (mapOfPisaVillage[i][j] == 0) {
          u8g2.setFont(u8g2_font_unifont_t_0_76);
          u8g2.drawUTF8(x + 16*j, y + 16*i, " ");
        } else {
          u8g2.setFont(u8g2_font_unifont_t_symbols);
          u8g2.drawUTF8(x + 16*j, y + 16*i, itoa(mapOfPisaVillage[i][j], buf,
10));
        }
      }
    }
    if (gameStatus == 1) {
      u8g2.drawUTF8(CursorX, CursorY, "<");
    }

  } while ( u8g2.nextPage() );
}
```

În final, am implementat o afișare pentru cazul în care jucătorul câștigă jocul și una pentru cazul în care pierde.

```
void WonGame(){
  digitalWrite(BUZZER, LOW);
  BoardGame();
  delay(4000);
  u8g2.firstPage();
  do {
    digitalWrite(BUZZER, LOW);
    u8g2.setFont(u8g2_font_squeezed_r6_tr);
    u8g2.drawStr(50, 10, "YOU WON!");
    u8g2.drawStr(30, 30, "The cats are happy :)");
    u8g2.drawStr(22, 40, "Press 'RESET' to play again.");
  } while (u8g2.nextPage());
  delay(4000);
}
```

```
void LostGame() {
```

```
  digitalWrite(BUZZER, LOW);
  BoardGame();
  delay(4000);
  u8g2.firstPage();
  do {
    digitalWrite(BUZZER, LOW);
    u8g2.setFont(u8g2_font_squeezed_r6_tr);
    u8g2.drawStr(50, 10, "YOU LOST!");
    u8g2.drawStr(28, 30, "The cats are now wet :(");
    u8g2.drawStr(22, 40, "Press 'RESET' to play again.");
  } while (u8g2.nextPage());

  delay(4000);
}
```

Am mai implementat și alte funcții care alcătuiesc acest joc, însă vă invit să descărcați codul sursă și să urmăriți implementarea mea .

Rezultate Obținute

Mai jos, am atașat un videoclip în care demonstrez funcționalitatea jocului:
Pisasweeper

Concluzii

Download

pisasweeper.zip

Jurnal

05.05.2024: Am creat pagina proiectului.
09.05.2024: Am dat comandă de componentele necesare pentru joc.
12.05.2024: Am preluat componentele și am început asamblarea lor.
13.05.2024: Am implementat toate componentele, înafara de joystick (probabil e stricat joystick-ul dar trebuie confirmat cu Teo la laborator miercuri).
15.05.2024: În urma laboratorului, Teo mi-a confirmat că joystick-ul e stricat. A incercat să mi-l lipească, însă are momente când citește aiurea mișcările, așa că o să-mi iau alt joystick.
16.05.2024: Am obținut noul joystick, l-am testat și a mers perfect, așa că mi-am propus să mă apuc de implementat jocul. În această zi mi-am făcut doar un research scurt.

18.05.2024: M-am apucat efectiv de implementarea proiectului și am reușit să fac funcționalitățile de bază (un nivel mediu și un meniu).
19.05.2024: Am mai făcut modificări de estetică logică și vizuală asupra softului și pot să spun că e 100% pregatiti din punct de vedere software. Am modificat puțin modul partea hardware, va trebui să refac schemele electrice.
20.05.2024: Am observat că îmi flicărește ecranul când mișc puțin firele de tensiune de pe breadboard, așa că am mutat niște fire, iar pentru estetică, am mai scurtat niște fire și am lipit joystick-ul și câteva fire de breadboard cu un leocoplast.
21.05.2024: Am cumpărat o cutie de lemn, vopsea mov, bandă dublu adezivă, baterie de 9V și bandă izolatoare. Am vopsit cutia, am învelit breadboardul cu banda izolatoare, după care am lipt plăcuța, LCD-ul și bateria și am terminat oficial proiectul.

Bibliografie/Resurse

Resurse Hardware
https://arduino-tutorials.net/tutorial/control-graphic-lcd-display-spi-st7920-128x64-with-arduino
https://arduinogetstarted.com/tutorials/arduino-joystick
https://arduinogetstarted.com/tutorials/arduino-button-piezo-buzzer
Resurse Software
https://github.com/olikraus/u8g2/wiki/fntlistallplain#10-pixel-height
https://www.arduino.cc/reference/en/language/functions/random-numbers/randomseed/
https://www.arduino.cc/reference/en/language/functions/random-numbers/random/
https://forum.arduino.cc/t/u8g-library-omega-symbol/499990/7

Export to PDF