Nokia Snake

Introducere

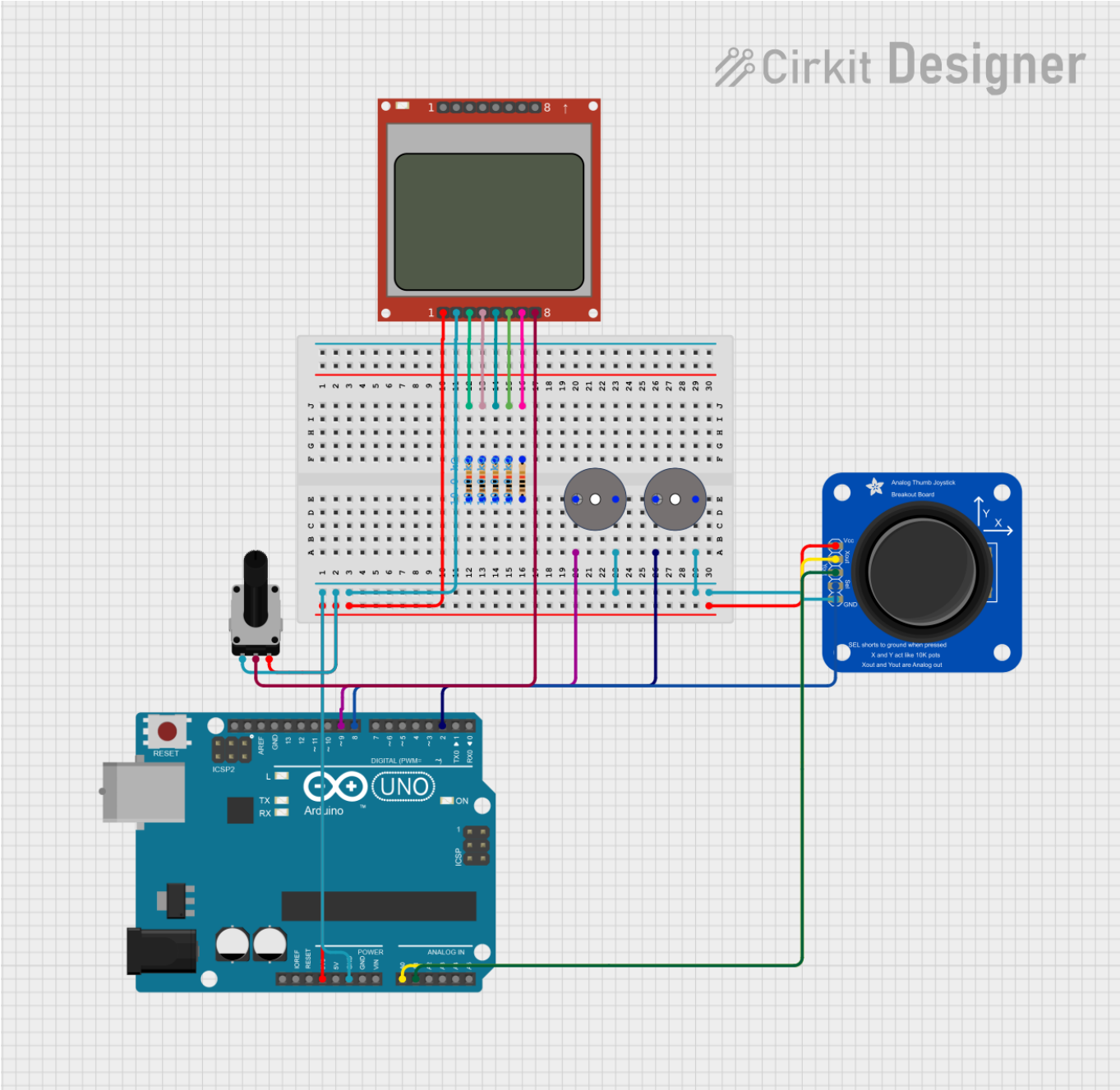
Jocul *Snake* reprezintă un joc clasic, despre care o mare parte din lume a auzit/cu care a interacționat. Astfel, consider că o reproducere a sa este un pas bun în dezvoltarea abilităților mele de a folosi un microproceesor, având un punct de referință pentru produsul final.

În plus, pentru a fi o experiență completă, am decis să folosesc un ecran de tip *Nokia 5110*.

Descriere generală

Plăcuța de Arduino UNO reprezintă creierul proiectului, comandând ecranul prin SPI (pinii 7-3) și buzzer-ul activ prin pinul 2.

Acestuia îi sunt transmise informații despre starea joystick-ului prin cele două porturi analog A0 și A1, precum și starea butonului prin intermediul pinului 8.



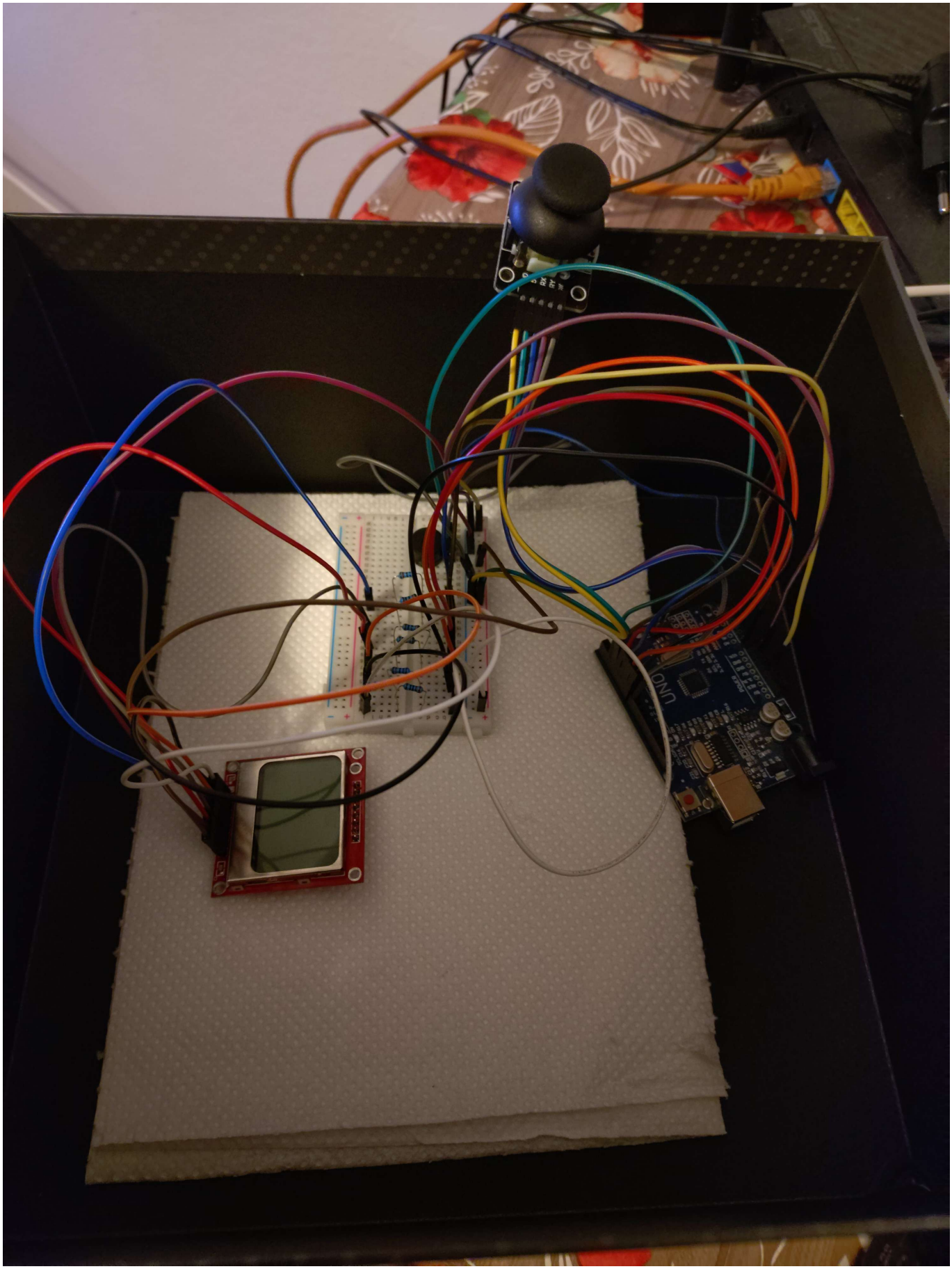
Din punct de vedere software, este folosită biblioteca Adafruit PCD 8544 împreună cu un singur fișier de tip main.

Hardware Design

Piesă	#	Use case	Datasheet
Arduino Uno	1	Microcontroller	https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf
PCD8544	1	Display	https://cdn.sparkfun.com/assets/b/1/b/e/f/Nokia5110.pdf
Buzzer Pasiv	2	Sunet	N/A
Rezistențe 10k ohmi	5	Conectare ecran LCD	N/A
Potentiometru 10k	1	Alimentare backlight	N/A
Joystick	1	Input	N/A

Breadboard	1	Conexiuni	N/A
Fire	Multe	Conexiuni	N/A

Ecranul funcționează la 3.3V, așadar rezistențele sunt necesare pentru a asigura funcționarea în parametrii.



Mai multe poze cu cablajul pot fi văzute aici.

Software Design

Fișierul principal a fost scris în Arduino IDE pentru a putea fi pus pe placă facil, în timp ce restul codului a fost scris în mod spartan în nvim.

Biblioteca folosită, precizată și mai sus, a fost biblioteca Adafruit PCD 8544 pentru a controla ecranul.

Din punct de vedere al logicii programului, acesta urmează o logică de FSM, cu următoarele stări:

```
enum STATE {
        STATE_MAIN_MENU,
        STATE_SETTINGS,
        STATE_GAME,
        STATE_PAUSE_MENU,
        STATE_GAME_OVER
};
```

Astfel, fiecare stare a mașinii reprezintă un “ecran”: jocul principal, meniul principal ș.a.m.d. Asta a facilitat implementarea diferitelor ecrane într-un mod modular.

Implementarea efectivă are următoarea structură:

```
main.ino   <- Conține logica de FSM: un switch/case
common.h   <- Conține variabilele partajate, pinii, stările, prototipurile
ecranelor
utils.h    <- Citirea JoyStick-ului
utils.cpp  <- idem
main_menu.cpp   <- Ecranul principal
settings.cpp    <- Ecranul setări
game.cpp        <- Ecranul joc
pause_menu.cpp  <- Ecranul pauză
game_over.cpp   <- Ecranul game over
```

Pentru a înțelege mai bine implementarea, o să detaliez funcționalitatea meniului de pauză.

Preambul:

```
#include "common.h"
#include "utils.h"

const int pauseMenuItems = 3; // Numărul de opțiuni
int pauseMenuIndex = 0;       // Opțiunea selectată
STATE pause_options[3] = {STATE_GAME, STATE_GAME, STATE_MAIN_MENU};
unsigned long pauseLastMoveTime = 0; // Timestamp pentru debounce
const unsigned long pauseMoveDebounce = 200;

extern bool gameInitialized;
```

“Driver” code:

```
void handlePauseMenu() {
        // Input handling
        uint8_t dir = readJoystick();
        if (millis() - pauseLastMoveTime > pauseMoveDebounce) { // Debounce
                if (dir & UP) {
                        pauseMenuIndex = (pauseMenuIndex - 1 +
pauseMenuItems) % pauseMenuItems;
                        pauseLastMoveTime = millis();
                        tone(PIN_BUZZER, 440, 200);
                } else if (dir & DOWN) {
                        pauseMenuIndex = (pauseMenuIndex + 1) %
pauseMenuItems;
                        pauseLastMoveTime = millis();
                        tone(PIN_BUZZER, 440, 200);
                }
        }

        // Afișare opțiuni
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(BLACK);
        display.setCursor(0, 0);
        display.println("Paused");

        display.setCursor(10, 15);
        display.print((pauseMenuIndex == 0) ? "> " : " "); // Decorăm
opțiunea selectată.
        display.println("Resume");

        display.setCursor(10, 25);
        display.print((pauseMenuIndex == 1) ? "> " : " ");
        display.println("Restart");

        display.setCursor(10, 35);
        display.print((pauseMenuIndex == 2) ? "> " : " ");
        display.println("Main Menu");

        display.display();

        if (pressJoystick()) {
                delay(150);
                tone(PIN_BUZZER, 220, 200);
                if (pauseMenuIndex >= 1) {
                        gameInitialized = false;
                }
                state = pause_options[pauseMenuIndex];
        }
}
```

Alte detalii care îmi plac sunt în utils și encodarea pentru direcții:

```
#define UP            1
#define DOWN  2
#define LEFT  4
#define RIGHT 8

uint8_t readJoystick() {
        uint8_t dir = 0;
        int y = analogRead(PIN_STICK_Y);
        int x = analogRead(PIN_STICK_X);
        if (y > THRESHOLD_TOP ) dir |= DOWN;
        else if (y < THRESHOLD_BOT) dir |= UP;
        if (x > THRESHOLD_TOP) dir |= RIGHT;
        else if (x < THRESHOLD_BOT) dir |= LEFT;
        return dir;
}
```

Acest mecanism îmi permite să verific simultan atât pentru direcții verticale cât și pentru direcții orizontale.

Rezultate Obţinute

- Implementarea elementelor de grafică: Lucrul pe ecranul de Nokia a fost o experiență plăcută, asemănătoare cu lucrul folosind biblioteca ncurses din C, din care m-am inspirat pentru ecrane.
- Implementarea unei melodii, chiar simplificată.
- Implementarea unui mecanism de meniu funcțional.

https://files.catbox.moe/ni9nyv.mp4 ← Jocul în acțiune

Download

Link proiect GitHub

Conectare cablaj → Upload main.ino :)

Bibliografie/Resurse

Hardware:

- https://cdn.sparkfun.com/assets/b/1/b/e/f/Nokia5110.pdf
- https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

Software

- https://github.com/adafruit/Adafruit-PCD8544-Nokia-5110-LCD-library

Export to PDF