# SnakeC

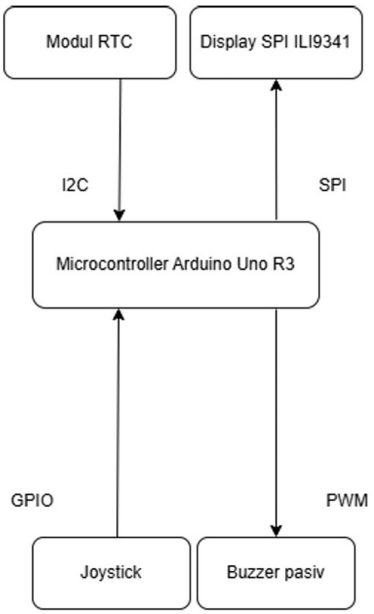
Munteanu Eugen 343C5

## Introducere

**SnakeC** este un sistem de calcul care ruleaza jocul clasic Snake, permitand utilizatorului sa controleze un sarpe prin intermediul unui joystick pentru a colecta fructe si a creste in dimensiune. Aplicatia include un meniu pentru selectarea nivelului de dificultate (fara obstacole sau cu obstacole) si introduce mecanici de tip power-up, precum incetinirea sarpelui sau distrugerea peretilor din nivelele complexe.

Ideea a pornit de la dorinta de a reimplementa un joc retro pe o arhitectura hardware simpla, adaugand elemente de gameplay pentru a creste complexitatea si interactivitatea fata de versiunea originala a jocului. Scopul reprezinta dezvoltarea unui sistem embedded capabil sa gestioneze logica unui joc in timp real, incluzand meniuri interactive, procesarea input-ului de la joystick si gestionarea conditiilor de finalizare a jocului.

## Descriere generală




Conform schemei bloc, unitatea centrala este reprezentata de microcontroller, care gestioneaza fluxul de date intre toate componentele sistemului. Controlul jocului este realizat prin intermediul joystick-ului, conectat prin pini GPIO (analogici A0 si A1). Microcontroller-ul citeste pozitia acestuia pentru a determina directia sarpelui, calculeaza noua stare a jocului si transmite datele procesate

catre display prin SPI. Aceasta conexiune asigura actualizarea elementelor grafice, precum sarpele, meniurile si hartile de joc.

Pentru feedback audio, sistemul include un buzzer pasiv controlat prin semnal PWM. Comunicarea unidirectionala microcontroller-buzzer are rolul de a emite sunete specifice evenimentelor din joc.

Gestionarea timpului si schimbarea hartii sunt realizate cu ajutorul modulului RTC, conectat prin interfata I2C. Acesta functioneaza ca un furnizor de date independent, trimitand informatii despre ora curenta catre microcontroller, in vederea alegerii hartii corespunzatoare.

# Hardware Design

## Componente folosite

| Componenta | Cantitate | Link magazin |
|---|---|---|
| Microcontroller de tip Arduino Uno R3 | 1 | Optimus Digital |
| Breadboard 400p | 2 | Optimus Digital |
| Fire mama-tata set 10buc | 2 | Optimus Digital |
| Fire tata-tata set 40buc | 1 | Optimus Digital |
| Modul RTC | 1 | Optimus Digital |
| Joystick analogic | 1 | Sigmanortec |
| Buzzer pasiv | 1 | Bitmi |
| LCD ILI9341 | 1 | Bitmi |

## Laboratoare folosite

- GPIO - citire analogica de la joystick
- SPI - conexiunea cu display-ul
- I2C - conexiunea microcontroller-RTC
- Timere+PWM - emiterea sunetelor specific jocului

## Conexiuni si asocieri pini

### Joystick

| Pin Joystick | Pin Arduino | Tip Semnal |
|---|---|---|
| VRx (HORZ) | A1 (PC1) | Analog In |
| VRy (VERT) | A0 (PC0) | Analog In |
| SW (Buton) | D6 (PD6) | Digital In |

| | | |
|---|---|---|
| VCC | 5V | Power |
| GND | GND | Ground |

#### Buzzer Pasiv

| Pin Buzzer | Pin Arduino | Tip Semnal |
|---|---|---|
| S (Signal) | D8 (PB0) | PWM Out |
| GND | GND | Ground |

#### Display LCD (interfata SPI)

| Pin Display | Pin Arduino | Rol Pin |
|---|---|---|
| MOSI | D11 (PB3) | Date SPI |
| MISO | D12 (PB4) | Date SPI |
| SCK | D13 (PB5) | Clock SPI |
| CS | D10 (PB2) | Chip Select |
| DC | D9 (PB1) | Data/Command |
| RST | RESET (PC6) | Reset |
| LED | 3.3V | Backlight |
| VCC | 5V | Power |
| GND | GND | Ground |

#### Modul RTC (interfata I2C)

| Pin RTC | Pin Arduino | Rol Pin |
|---|---|---|
| SCL | A5 (PC5) | I2C Clock |
| SDA | A4 (PC4) | I2C Data |
| VCC | 5V | Power |
| GND | GND | Ground |

# Software Design

# Stadiul implementarii software

In prezent, implementarea software este functionala si integrata cu toate componentele periferice. Structura meniului principal permite navigarea prin optiuni si selectarea corecta a tipului de nivel (Tutorial sau Level). Motorul jocului gestioneaza corect logica sarpelui, translatia cozii care urmareste capul, teleportarea pe margini si detectia coliziunilor cu peretii sau cu propriul corp.

# Elementul de noutate al jocului + functionalitati utilizate

Un element important al jocului consta in transformarea unui joc retro intr-un sistem dinamic influentat direct de factori externi asincroni, un sistem adaptat la limitarile unei arhitecturi embedded minimaliste. In plus, modulul RTC este folosit pentru a schimba configuratia fizica a hartii la un timp setat (doua minute), fara interventia utilizatorului.

Utilizarea conceptelor din laborator este justificata de nevoile fiecarei componente din sistem:

- Pinii GPIO - citesc starea butonului de pe joystick folosind rezistenta pull-up
- Canalele ADC - transforma semnalele analogice de pe axele X si Y in valori numerice pentru calculul directiei sarpelui.
- Protocolul SPI - utilizat datorita ratei mari de transfer necesare actualizarii rapide a pixelilor pe ecran
- Protocolul I2C - utilizat pentru modulul RTC deoarece foloseste doar doua fire pentru un volum mic de date,
- Timere + PWM - genereaza frecventele audio pentru buzzer, folosind note predefinite

# Mediu de dezvoltare, biblioteci, implementare

In cadrul dezvoltarii jocului, s-a folosit *Arduino IDE* care integreaza nativ toolchain-ul *avr-gcc*, permite optimizarea codului de tip C pentru arhitectura procesorului si ofera utilitarul *Serial Monitor*, foarte util pentru debugging, monitorizare in timp real si testarea componentelor.

# Biblioteci folosite

- `Adafruit_GFX.h` - generarea formelor geometrice si gestionarea fonturilor text
- `Adafruit_ILI9341.h` - controlul ecranului TFT prin magistrala SPI
- `Wire.h` - gestionarea hardware-ului dedicat protocolului I2C
- `Rtc_Pcf8563.h` - interactiunea cu modulul RTC folosit, PCF8563

Firmware-ul proiectului SnakeC este structurat sub forma unui sistem organizat in jurul unui *Automat cu stari finite* (switch-case). Acesta izoleaza executia meniurilor interactive, a ecranului de final si a jocului propriu-zis. In timpul starii active, codul implementeaza un timing bazat pe functia `millis()`, eliminand intarzierile si permitand procesarea asincrona a input-ului analogic de la joystick si monitorizarea expirarii boosterelor. Validarea functionala a componentelor a fost realizata incremental (unit testing hardware), prin **multe** teste de hardware-check, urmarire de *Serial Monitor* si analiza datasheet-urilor.

# Calibrarea elementelor de senzoristica

Calibrarea senzoristica a fost aplicata pentru joystick-ul analogic conectat la pinii ADC (A0 si A1). Convertorul intern al microcontroller-ului returneaza valori intr-o plaja de la 0 la 1023 (in practica cam intre 30 si 1000), unde centrul teoretic ideal ar trebui sa fie in jurul valorii de 512. In practica, din cauza imperfectiunilor mecanice, a uzurii si a zgomotului electric de pe fire, valorile de repaus fluctueaza. Am ales ca un threshold valoarea 120 pentru a evita input-uri accidentale si miscari nedorite ale sarpelui.

# Optimizari

Ca **optimizari**: Miscarea sarpelui foloseste un algoritm de tip coada FIFO, unde fiecare segment copiaza coordonatele celui din fata sa. Performanta grafica pe magistrala SPI este optimizata prin randare incrementala, stergand doar vechea coada si desenand noul cap. Legat de **memorie**, am eliminat operatiile folosite initial `strcmp()` si `strcpy()`, am folosit `uint8_t` unde s-a putut optimiza, iar string-urile de text au fost puse in wrapper-ul `F()` pentru mai multa memorie SRAM disponibila.

# Rezultate, concluzii, download

In urma finalizarii proiectului SnakeC, s-a obtinut un sistem embedded compact, stabil si functional, capabil sa gestioneze logica unui joc in timp real. Se demonstreaza deci ca implementarea unei aplicatii interactive cu cerinte grafice si de timp real este pe deplin realizabila pe o arhitectura hardware minimalista.

Bineinteles, codul poate fi dezvoltat si/sau optimizat hardware ori software, de pilda prin implementarea intreruperilor hardware sau timerelor dedicate (e.g. *timer interrupt* la ~150ms), sau printr-o reorganizare fizica a proiectului (cabluri, carcasa). Cautarea si gasirea unei teme nu au fost cele mai usoare experiente, dar prin acest proiect am invatat ce sta in spatele unui sistem simplu embedded si cum functioneaza protocoalele de comunicatii.

| **Prezentare Video Proiect** |
| --- |
| **Cod Sursa GitHub** |

# Jurnal

2026-05-08 - Pagina OCW creata
2026-05-08 - *Introducere* + *Hardware Design*
2026-05-09 - Actualizare *Introducere* + *Hardware Design*
2026-05-09 - Adaugare *Descriere generală*, schema-bloc, link-uri datasheet
2026-05-11 - Completare *Hardware Design*, adaugarea conexiunilor si asocierilor de pini
2026-05-18 - Actualizarea codului sursa, testare si debugging
2026-05-19 - Adaugare *Software Design* partial: mediu de dezvoltare, biblioteci, functionalitati, implementare

2026-05-23 - Adaugare video + link Github
2026-05-23 - Completare *Software Design* si concluzii ale proiectului

# Bibliografie/Resurse

Datasheet-uri:

- LCD ILI9341 Datasheet
- RTC PCF8563 Datasheet
- ATmega328P Datasheet

Export to PDF

From:
https://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
**https://ocw.cs.pub.ro/courses/pm/prj2026/bianca.popa1106/eugen.munteanu2604**

Last update: **2026/05/23 19:20**