Universitatea POLITEHNICA București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Proiect 2 – Electronică aplicată
Joc “Repetă culorile”

Coordonator: Andrei Dăescu
Titular: Corneliu Burileanu

Studenți: Vlad-Ștefan Moreanu, Teodora-Alexandra Opriș

An universitar: 2022-2023

Cuprins

Listă acronime

1. DIP – Dual In-line Package
2. IDE – Integrated Development Environment
3. LCD – Liquid Crystal Display
4. LED RGB – Light Emitting Diode (Red, Green, Blue)

Listă figuri

1. Introducere

"Repetă culorile" este un joc asemănător lui "Simon" [1] (a nu se confunda cu "Simon Says"), inventat în 1978 și foarte popular în perioada anilor 1980.

Un joc de memorare, "Simon" generează o serie de tonuri si lumini, iar jucătorul trebuie să repete secvența dată. Dacă jucătorul reușește, secvența devine mai lungă și mai complexă. Când jucătorul greșește sau când timpul alocat încercării expiră, jocul se oprește și se resetează.
Pe același model funcționează și proiectul nostru, "Repetă culorile". Sunt folosite patru LED-uri RGB (Light Emitting Diodes) de diferite culori, patru butoane (push-button) și un LED verde adițional. Microcontrolerul generează un șir de numere aleatoare; acesta se mărește de fiecare dată când există concordanță între LED-urile aprinse și apăsarea butoanelor aferente. Șirul de numere trebuie transpus în culori, iar jucătorul trebuie să apese butoanele în ordinea în care s-au aprins LED-urile. Deci, de fiecare dată când este apăsat butonul corect (sau butoanele corecte), secvența de LED-uri care se aprind devine mai lungă și implicit mai complexă.

Scorul curent va fi afișat pe un display LCD (Liquid Crystal Display) și va fi actualizat după fiecare rundă.

Pentru a modifica gradul de dificultate al jocului, se folosește un comutator DIP Switch (Dual In-line Package Switch). Prin grad de dificultate înțelegem viteza cu care se aprind LED-urile, respectiv timpul de așteptare între apăsarea a două butoane de către jucător.

În vederea efectuării proiectului, s-au realizat schema electrică, cablajul, flowchart-ul (diagrama software), iar codul a fost scris folosind IDE-ul (Integrated Development Environment) Arduino.

2. Resurse hardware

Componentele hardware folosite sunt următoarele:

- Intel Galileo Gen2
- 2 Breadboard-uri – folosite pentru realizarea cablajului
- 4 Butoane (push button)
- 4 LED-uri RGB
- 1 LED Verde – folosit pentru a atenționa jucătorul dacă jocul merge mai departe (butoanele au fost apăsate corect) sau dacă se resetează (butoanele au fost apăsate într-o ordine incorectă)
- LCD – folosit pentru redarea scorului
- DIP Switch – folosit pentru a modifica dificultatea jocului (viteza aprinderii LED-urilor și timpul de așteptare între două apăsări de butoane)
- Rezistoare (100 Ω, 220 Ω) – folosite pentru limitarea curentului ce trece prin LED-uri
- Potențiometru (10 kΩ) – folosit pentru a regla contrastul de pe LCD
- Fire dupont tată-tată – folosite pentru realizarea conexiunilor de pe breadboard-uri

3. Resurse software

Resursele software folosite sunt următoarele:

- Arduino IDE – mediul de dezvoltare folosit pentru scrierea și implementarea codului
- Librăria "LiquidCrystal" [2]
  - necesară în situațiile în care se folosește un LCD
  - permite plăcii Galileo să controleze LCD-uri, pe baza chipset-ului Hitachi HD44780 (sau altceva compatibil), care este regăsit în majoritatea LCD-urilor text-based
  - conține funcții precum: LiquidCrystal(), begin(), clear(), setCursor(), print() etc.
- Librăria "vector" [3] – necesară în programele în care se folosesc vectori

4. Implementare hardware

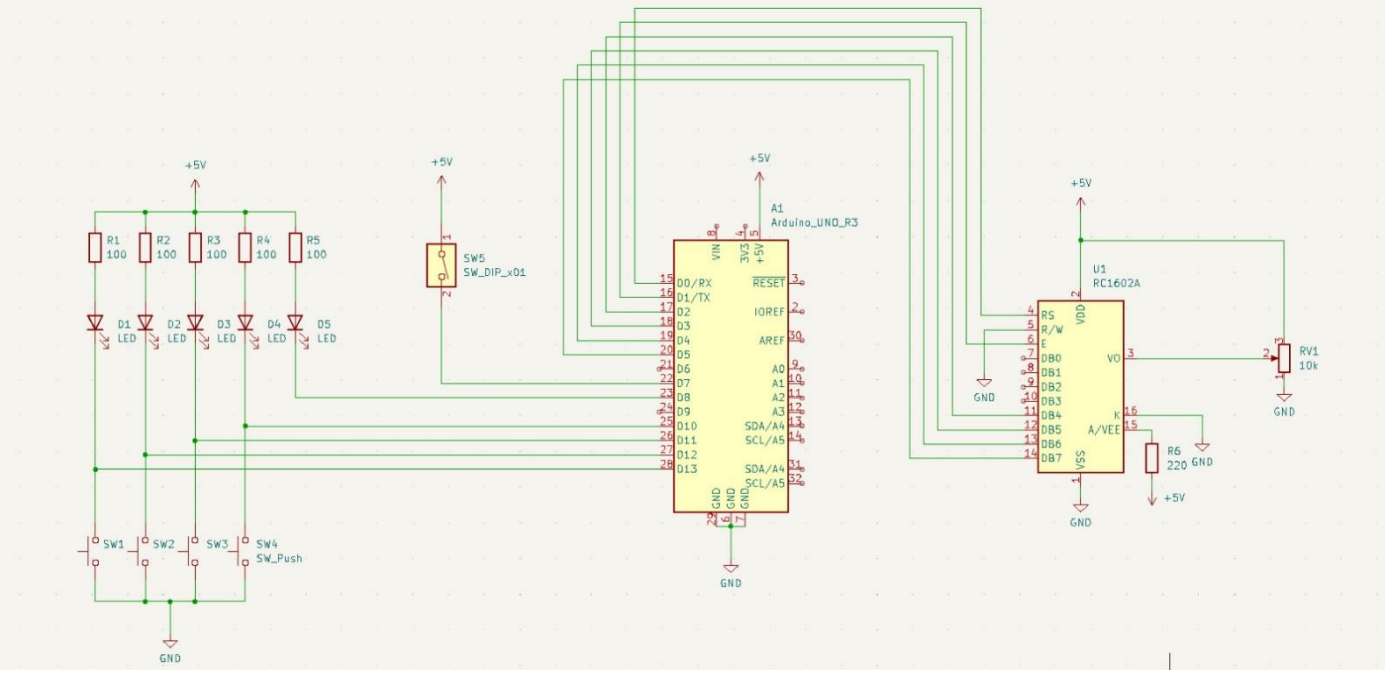


Fig. 4.1. Schema electrică

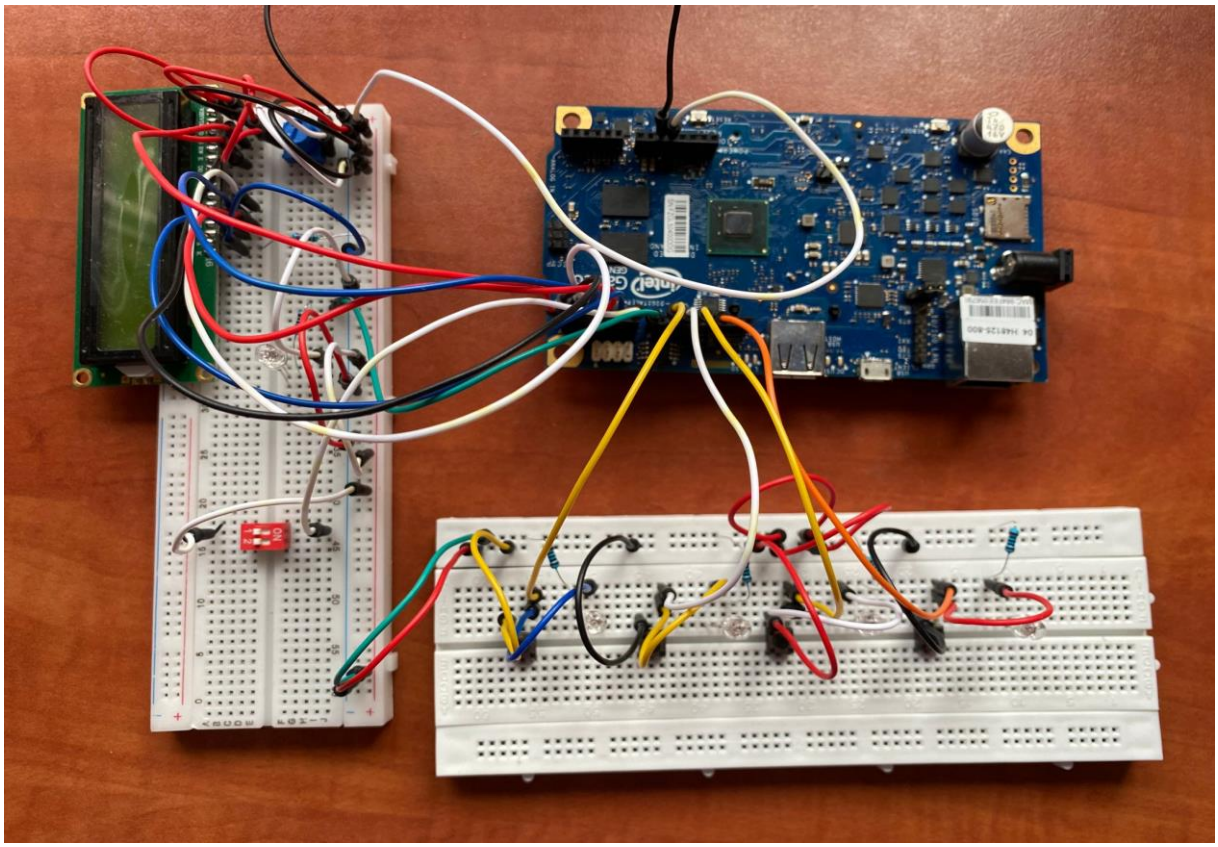


Fig. 4.2. Cablajul

4. Implementare Hardware

Primul breadboard conține cele 4 LED-uri RGB, 4 butoane și 4 rezistoare. Este o configurație simplă: fiecare buton este, la un terminal, legat de catodul LED-ului care îi corespunde, iar celălalt terminal este legat la masă. LED-urile sunt, de asemenea, legate de câte un rezistor de 100 de Ohmi pentru a asigura limitarea curentului. În absența acestor rezistoare, LED-urile ar putea "trage" din pinii plăcii Galileo un exces de curent, ceea ce ar duce la o potențială daună a pinilor sau a LED-urilor în sine. Pe scurt, rezistoarele sunt folosite pentru a limita curentul ce trece prin LED-uri la un nivel ce asigură buna funcționare a lor. Valoarea de 100 de Ohmi a fost aleasă datorită faptului că, în general, rezistoarele folosite în acest scop au valori cuprinse între 100 și 330 de Ohmi, în funcție de specificațiile LED-urilor. De asemenea, o valoare mai mare decât cea aleasă ar face LED-urile să lumineze mai slab, așa că varianta de 100 de Ohmi a fost considerată cea mai potrivită.

Cel de al doilea breadboard conține 1 LED RGB, un DIP switch, potențiometru și un LCD. Switch-ul este legat la pinul 7 al plăcii Galileo și la alimentarea de +5V. Este folosit pentru a ajusta dificultatea jocului, adică pentru a modifica viteza cu care se aprind LED-urile, precum și timpul alocat pentru apăsarea butoanelor de către jucător. LED-ul (ales verde) este folosit pentru a da flash-uri diferite, în funcție de starea jocului. Acesta alertează jucătorul dacă jocul se resetează sau dacă merge mai departe (au fost alese frecvențe diferite ale flash-ului pentru fiecare scenariu în parte). Se folosește și un rezistor de 100 de Ohmi din aceleași considerente ca mai sus (pentru a limita curentul ce trece prin LED, precum și pentru a asigura o bună luminozitate a LED-ului). Potențiometrul este folosit pentru a regla contrastul de pe display. Un terminal este legat la LCD, altul la alimentare, iar ultimul la masă. Catodul LCD-ului este legat la masă, iar anodul este conectat la un rezistor de 220 Ohmi, care mai apoi este conectat la alimentarea de +5V. Modulul LCD funcționează cu o diferență de potențial pe materialul său (liquid crystal), iar prin conectarea catodului la masă, se asigură o referință pentru această diferență de potențial. Anodul este conectat la un rezistor cu scopul de a limita curentul ce trece prin LCD la un nivel sigur, pentru a garanta buna funcționare și pentru a evita posibile daune ale LCD-ului. Celălalt terminal al rezistorului este conectat la alimentarea de +5V pentru a asigura tensiunea necesară de-a lungul LCD-ului. Această configurație permite LCD-ului să afișeze caractere și simboluri (în cazul acestui proiect, litere și cifre).

Aceste două breadboard-uri sunt conectate, prin intermediul firelor dupont tată-tată, la placa Intel Galileo Gen2.

5. Implementare software

Codul jocului:

```
#include <vector>
#include <LiquidCrystal.h>

// PINS
enum { rs = 0, en, d4, d5, d6, d7 };
enum { b1 = 10, b2, b3, b4 };
int led = 8;
int sw = 7;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

std::vector<int> i_Seq;

// VARS
int g_StartTime = 0;
int g_NoPins = 4;
int g_InputCount = 0;
int g_Score = 0;
int i_ExpRd;
int p_LastBtn;
int d_WaitTime = 5000;
int d_SeqDelay = 1000;

// FLAGS
bool f_Wait = false;
bool f_Press = false;
bool f_Reset = false;

void SetPinDirection(byte dir)
{
  for(int i = 0; i < g_NoPins; i++)
  {
    pinMode(b1 + i, dir);
  }
}

void WriteAllPins(int val)
{
  //SetPinDirection(OUTPUT);
  for(int i = 0; i < g_NoPins; i++)
  {
    digitalWrite(b1 + i, val);
  }
}

void Flash(int halfPeriod)
```

5. Implementare Software

```
{
  SetPinDirection(OUTPUT);
  for(int i = 0; i < 5; i++)
  {
    WriteAllPins(LOW);
    digitalWrite(led, HIGH);
    delay(halfPeriod);
    WriteAllPins(HIGH);
    digitalWrite(led, LOW);
    delay(halfPeriod);
  }
}

void PlaySequence()
{
  //SetPinDirection(OUTPUT);
  WriteAllPins(HIGH);
  for(int i=0; i < i_Seq.size(); i++)
  {
    digitalWrite(b1 + i_Seq[i], LOW);
    delay(d_SeqDelay);
    digitalWrite(b1 + i_Seq[i], HIGH);
    delay(d_SeqDelay/10);
  }
}

void ResetFunc()
{
  if(digitalRead(sw) == HIGH)
  {
    d_WaitTime = 2000;
    d_SeqDelay = 500;
  }
  else
  {
    d_WaitTime = 5000;
    d_SeqDelay = 1000;
  }

  Flash(100);
  g_Score = 0;
  f_Reset = 0;
  i_Seq.clear();
  lcd.clear();
  lcd.print(„Score:”);
  lcd.setCursor(0, 1);
  f_Wait = false;
  delay(100);
}

void setup()
{
```

5. Implementare Software

```
  pinMode(led, OUTPUT);
  lcd.begin(16, 2);
  Serial.begin(9600);
  randomSeed(millis());
  ResetFunc();

  Serial.println(„SETUP COMPLETE, STARTING...”);
}


void loop()
{
  //print score
  lcd.setCursor(0, 1);
  lcd.print(g_Score);

  if(!f_Wait)
  {
    i_Seq.push_back(random(g_NoPins));
    PlaySequence();
    g_InputCount = 0;
    g_StartTime = millis();
    f_Reset = false;
    f_Press = false;
    f_Wait = true;
  }
  else if(millis() – g_StartTime <= d_WaitTime)
  {
    SetPinDirection(INPUT_PULLUP);
    i_ExpRd  = i_Seq[g_InputCount];
    if(!f_Press)
    {
      for(int i=0; i < g_NoPins; i++)
      {
        if (b1+i_ExpRd == b1+i) continue;
        if (digitalRead(b1+i) == LOW)
        {
          f_Press = true;
          p_LastBtn = b1+i;
          Serial.println(„RECEIVED WRONG INPUT:”);
          Serial.println(b1+i);
          Serial.println(„YOU LOSE”);
          f_Reset = true;
        }
      }
    }
    if(!f_Press && digitalRead(b1+i_ExpRd) == LOW)
    {
      g_StartTime = millis();
      g_InputCount++;
      p_LastBtn = b1+i_ExpRd;
      f_Press = true;
      Serial.println(„Correct button press:”);
```

5. Implementare Software

```
      Serial.println(p_LastBtn);
      //CORRECT
    }
    else if(f_Press && digitalRead(p_LastBtn)==HIGH)
    {
      f_Press = false;
      if(f_Reset) {
        ResetFunc();
      }
    }
    else if(g_InputCount == i_Seq.size())
    {
      g_Score += g_InputCount;
      Serial.println(„End of sequence, next round...”);
      g_InputCount = 0;
      f_Wait = false;
      Flash(200);
      delay(500);
    }
  }
  else
  {
    Serial.println(„TIME OUT, YOU LOSE”);
    ResetFunc();
  }

}
```

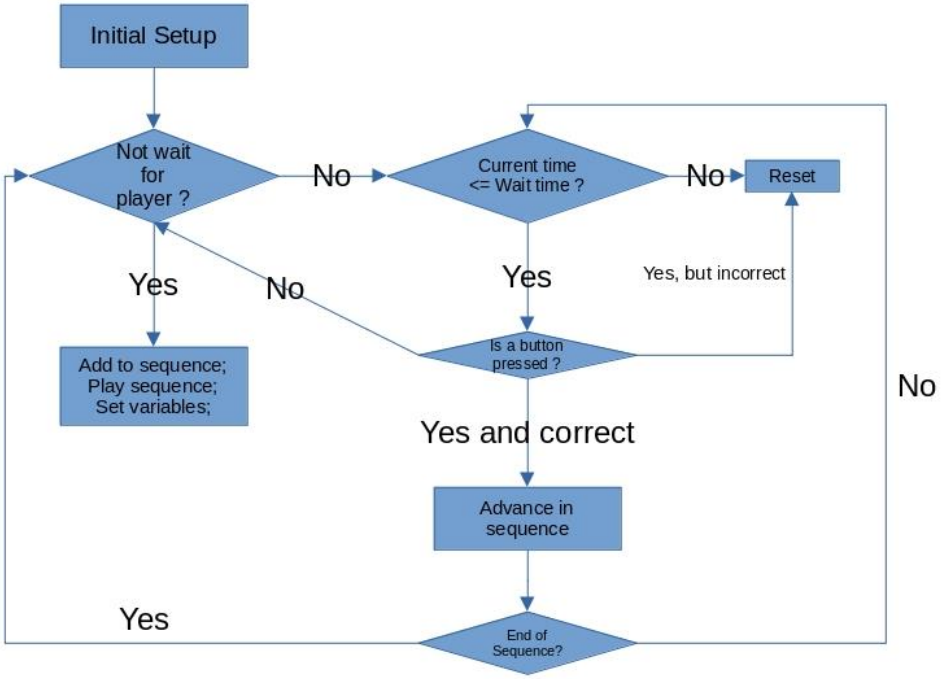
5. Implementare Software

Explicație:

Se includ librăriile necesare, “vector” și “LiquidCrystal”.
Se aleg configurațiile pinilor: LCD-ul este conectat la pinii 0-5 ai plăcii Galileo, cele 4 butoane la pinii 10-13, LED-ul verde la pinul 8 și DIP Switch-ul la pinul 7.
Se declară vectorul “i_Seq”, care va fi folosit pentru a stoca o secvență de numere.
Se declară mai multe variabile: “g_StartTime” reprezintă timpul de start al unei secvențe, “g_NoPins” reprezintă numărul de pini folosiți pentru butoane, “g_InputCount” numără inputurile făcute (apăsările de buton), “g_Score” stochează scorul jucătorului, “i_ExpRd” stochează indicele apăsării de buton care este așteptată, “p_LastBtn” stochează valoarea ultimului buton apăsat, “d_WaitTime” reprezintă timpul de așteptare pentru input (apăsarea unui buton), iar “d_SeqDelay” este delay-ul dintre flash-urile de LED.
Se declară variabilele tip boolean “f_Wait”, “f_Press”, “f_Reset”. Prima are rolul de a indica dacă programul așteaptă inputul jucătorului, a doua indică dacă a avut loc o apăsare de buton, iar ultima se setează atunci când e nevoie de reset.
Funcția “SetPinDirection” setează direcția (“INPUT”, “OUTPUT” sau “INPUT_PULLUP”) pentru toți pinii folosiți în program.
Funcția “WriteAllPins” setează valoarea output-ului digital pentru toți pinii folosiți în program.
Funcția “Flash” setează direcția pinilor ca “OUTPUT”, pune toți pinii LOW, aprinde LED-ul verde (led), întârzie cu “halfPeriod”, apoi pune toți pinii HIGH, stinge LED-ul verde, și întârzie din nou cu “halfPeriod”. Acest proces se repetă de cinci ori.
Funcția “PlaySequence” iterează prin elementele vectorului “i_Seq” și aprinde LED-urile corespunzătoare pe baza valorilor stocate în vector.
Funcția “ResetFunc” resetează jocul și “pregătește” unul nou. Mai întâi verifică starea lui “sw” (un DIP Switch conectat la pinul 7). Dacă este HIGH, atunci variabilele “d_WaitTime” și “d_SeqDelay” sunt setate să aibă valori mai mici, ceea ce duce la un joc mai rapid și mai dificil. În caz contrar (switch-ul este LOW), se folosesc valorile default ale celor două variabile. Apoi, funcția “golește” si resetează variabilele și fanioanele, precum și display-ul.
Funcția “Setup” este o funcție Arduino standard și este apelată odată ce pornește programul. Funcția aceasta setează ca “OUTPUT” pinul la care este legat LED-ul verde, inițializează display-ul LCD-ului folosind funcția “begin” (se specifică numărul de coloane (16) și rânduri (2) ale LCD-ului), se începe comunicația serială cu un baudrate de 9600 și apelează funcția “ResetFunc” pentru a inițializa jocul. De asemeanea, afișează pe display un mesaj care înștiințează jucătorul că setup-ul este complet, și că jocul va începe.
Funcția “loop” este bucla programului care va rula continuu. Verifică inputul jucătorului în mod repetat, actualizând starea jocului. Se bazează pe mai multe fanioane și variabile pentru a verifica progresul jocului, inputul jucătorului și pentru a determina când și dacă jocul se resetează sau avansează la următoarea rundă.

5. Implementare Software

Flowchart:



6. Concluzii

Deși nu este un proiect deosebit de complex, am întâmpinat probleme pe parcursul implementării sale. În general, problemele aveau legătură cu partea de hardware. Componentele nu făceau un bun contact cu breadboard-ul (sau acest contact nu era menținut cât timp jocul era în desfășurare), ceea ce ducea la disfuncționalități ale jocului. Numărul mare de fire este, probabil, un alt factor ce a dus la apariția unor probleme.
Deși schema electrică a fost concepută în așa fel încât să fie relativ simplă, cablajul prezintă mai multe fire de legătură, ceea ce a aglomerat breadboard-urile, și implicit întregul proiect.
După o serie de încercări, am reușit să aducem proiectul într-o stare funcțională, deși pe partea de aspect ar mai fi fost loc pentru îmbunătățiri.
Acest proiect a reprezentat un mod bun de a dobândi noi cunoștințe despre Intel Galileo Gen2. De asemenea, a fost o ocazie de a ne îmbunătăți abilitățile de debugging și de a lucra cu software-ul dedicat realizării de scheme electrice (KiCad).

7. Bibliografie

[1] Simon Game, https://en.wikipedia.org/wiki/Simon_(game)

[2] Liquid Crystal Displays (LCD) with Arduino, https://docs.arduino.cc/learn/electronics/lcd-displays

[3] Vector header, https://cplusplus.com/reference/vector/