

***Proiect Proiectare cu Microprocesoare (PMP)***  
***-Joc de Memorie-***

**Profesor Indrumator:**  
**Mircea Paul Muresan**

**Realizat de:**  
**Bendea Sergiu Daniel**

# Cuprins

- I.    Specificatie Proiect**
- II.   Proiectarea si Schema Electrica**
- III.   Manual de Utilizare**

## I. Specificatie Proiect :

Proiectul consta in implementarea pe o placuta de dezvoltare Arduino a unui joc de memorie.

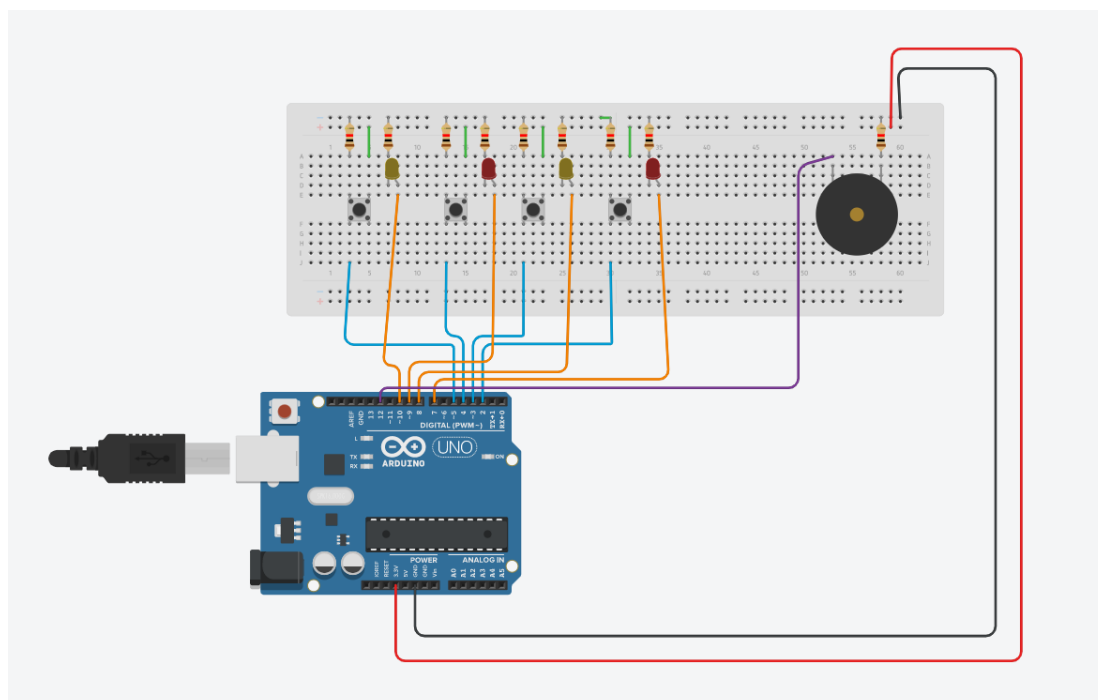
Problema consta in realizarea structurii si montarea circuitului necesar pentru realizarea jocului din punct de vedere hardware, precum si a aplicatiei in sine, care va veni apoi incarcata in memoria placii de dezvoltare.

## II. Proiectare si Schema Electrica :

Pentru implementarea acestui proiect am decis sa utilizez urmatoarele componente:

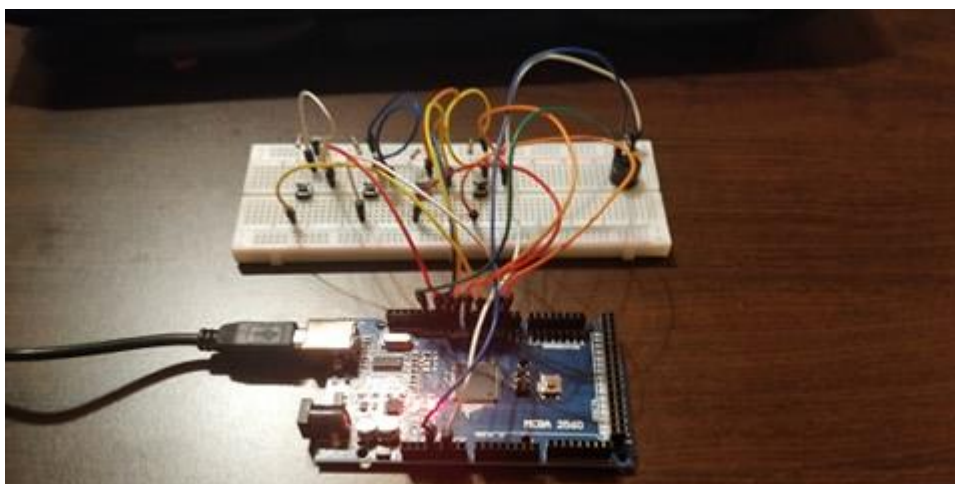
- 4 leduri – 2 rosii si 2 galbene
- 4 butoane
- 1 buzzer
- 9 rezistente
- Breadboard
- Placa de dezvoltare Arduino

Inainte de montarea fizica a circuitului, acesta a fost realizat in Tinkercad pentru o implementare ulterioara mai usoara. Schema realizata este urmatoarea:



*Modelul din figura actuala a fost actualizat pentru a corespunde implementarii fizice.*

Montarea fizica a circuitului este urmatoarea:



Aplicatia este dezvoltata utilizand Arduino, versiunea 1.8.13. Aceasta este impartita in functii dupa cum urmeaza:

- Setup() : functia apelata pentru setarea datelor initiale

```
// initializarea variabilelor de mai sus
void setup() {
  // put your setup code here, to run once:
  // variabilele
  nrSequence = 1;

  nLevel = 8;
  nCurentLevel = 1;

  gameOn = 1;

  // butoanele
  pinMode(buton1, INPUT);
  pinMode(buton2, INPUT);
  pinMode(buton3, INPUT);
  pinMode(buton4, INPUT);
  // ledurile
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  // buzzerul
  pinMode(buz, OUTPUT);

  randomSeed(analogRead(0));

  Serial.begin(9600);
}
```

- Loop() : functia principala care ruleaza jocul

```
void loop() {
    // put your main code here, to run repeatedly:
    // testWindow();
    if(gameOn == 1) {
        refreshGameSequence(vecSequence, nrSequence);
        showSequence();
        evaluateSequence();
        if(gameOn == 1) {
            loadNextLevel();
        }
        else {
            loadLoseSignal();
        }
        if(nLevel == nCurentLevel) {
            loadVictorySignal();
        }
    }
}
```

- sunetBuzz(int ton, int timp) :
- refreshGameSequence(int v[], int n) : genereaza o noua secventa aleatorie de lungime egala cu nivelul actual al jocului – n

```
void refreshGameSequence(int v[], int n) {
    // v este un vector de dimensiune variabila, in care se va memora ordinea ledurilor care vor fi aprinse
    for(i = 0; i < n; i++) {
        v[i] = 0;
        // prea mic
        //int randN = random(1, 100); // luam o valoare aleatorie intre 1 si 100 si in functie de valoarea acesteia asignam un bec
        // 1 - 25 -> bec1
        // 26 - 50 -> bec2
        // 51 - 75 -> bec3
        // 76 - 100 -> bec4

        int randN = random(1, 200); // luam o valoare aleatorie intre 1 si 100 si in functie de valoarea acesteia asignam un bec
        // 1 - 50 -> bec1
        // 51 - 100 -> bec2
        // 101 - 150 -> bec3
        // 151 - 200 -> bec4

        if(randN <= 50) {
            v[i] = 0;
        }
        else {
            if(randN <= 100) {
                v[i] = 1;
            }
            else {
                if(randN <= 150) {
                    v[i] = 2;
                }
                else {
                    if(randN <= 200) {
                        v[i] = 3;
                    }
                }
            }
        }
    }
}
```

- showSequence() : afiseaza pe placuta secventa generate in cadrul functiei refreshGameSequence

```
void showSequence() {
    delay (200);
    Serial.print("seq :\n");
    for (i = 0; i < nCurentLevel; i++){
        becDelay = becTime/(1+(speedFactor/nLevel)*(nCurentLevel - 1));

        Serial.println(vecSequence[i]);
        Serial.print(" ");

        digitalWrite(vecSequence[i]+7, HIGH);
        sunetBuzz(sunet[vecSequence[i]], becDelay);
        digitalWrite(vecSequence[i]+7, LOW);

        delay(speedBase/speedFactor);
    }
}
```

- evaluateSequence() : dupa afisarea secventei aceasta secventa se ocupa de verificarea corectitudinii introducerii secventei de catre utilizator

```
void evaluateSequence() {
    int j = 0;
    int butonApasat = 0;
    while(j < nCurentLevel && gameOn == 1) {
        Serial.println("ButonApasat");

        butonApasat = 0;
        for(int i = 0; i < 4; i++) {
            stareButoane[i] = LOW;
        }

        while(butonApasat == 0) {
            for(i = 0; i < 4; i++) {
                stareButoane[i] = digitalRead(i+2);
                if(stareButoane[i] > 0) {
                    Serial.print(i);
                    butonApasat++;
                    break;
                }
            }
        }

        Serial.println("ButonApasat_2");
        for(i = 0; i < 4; i++) {
            if(stareButoane[i] == HIGH) {
                digitalWrite(i+7, HIGH);
                sunetBuzz(sunet[vecSequence[i]], becDelay);
                digitalWrite(i+7, LOW);
                Serial.print(i);
                if(vecSequence[j] == i) {
                    j++;
                }
                else {
                    gameOn = 0;
                }
            }
        }
    }
}
```

- loadNextLevel() : in cazul introducerii corecte a secventei aceasta functie are rolul de genera datele necesare urmatorului nivel

```
void loadNextLevel() {
    nCurentLevel++;
    speedFactor++;
    becTime = becTime - 25;
    if(nCurentLevel%5 == 0) {
        speedBase = speedBase + 50;
    }
}
```

- loadLoseSignal() : incarca o secventa in momentul in care jocul a fost pierdut
- loadVictorySignal() : incarca o sceventa ce anunta Victoria
- testWindow() : o functie ce se ocupa cu testarea componentelor si a anumitor functii

### III. Manual de Utilizare:

1. La inceputul fiecarui runde ledurile vor incepe sa se lumineze unul dupa celalalt, acompaniate de un sunet specific al buzzer-ului.
2. In urma afisarii secventei utilizatorul va apasaea pe butonul specific fiecarui led pentru a le aprinde pe acestea, fiind nevoit sa apase pe butoane in ordinea initiala a aprinderii ledurilor pentru a trece la nivelul urmator si a castiga jocul.
3. In cazul introducerii unei secvente gresite, se va genera o secventa care indica pierderea jocului si se asteapta resetarea acestuia.
4. In cazul introducerii corecte a tuturor secventelor, se va genera o secventa care indica castigarea jocului.
5. Pentru a incepe din nou jocul/sau al reseta se va apasa butonul de reset de pe placuta Arduino.

Explicatie video a functionalitatii aplicatiei: <https://youtu.be/cqZn9F3U2bM>