Proiect 2 – Electronică Aplicată
Joc de Lumini Bluetooth

Coordonatori : Ana Neacșu, Mihai Muntean, Alexandru Guzu, Andrei Dăescu , Dorin Enachescu

Studenți : Barbu Mihai, Gagiu Andreea-Maria, Stan Rareș-Gabriel 431B

An Universitar 2023 - 2024

Cuprins

I . Introducere

I.1. Descriere:

Cu ajutorul plăcii Intel Galileo Gen 2 se realizează un joc de lumini controlate wireless prin comenzi date de către un device. Se asamblează un cub de LED-uri de 4x4x4. Se programează patru jocuri de lumini care sunt comandate prin Bluetooth cu ajutorul modulului ESP32, care transmite comenzile la placa Galileo prin comunicație serială UART. Comanda led-urilor de către Intel Galileo se face prin intermediul a patru multiplexoare CD4051BE.

I.2. Obiective:

- Asamblarea Hardware:
  - Construirea celor 4 matrici de LED-uri
  - Asamblarea matricilor intr-un cub
  - Conectarea multiplexoarelor pe breadboard
  - Interfatarea cubului cu iesirile multiplexoarelor plasate pe breadboard
  - Interfatarea intrarilor multiplexoarelor plasate pe breadboard cu iesirile placii Intel Galileo

- Dezvoltarea Software:
  - Scrierea unui program pe placa Galileo Gen 2 care controlează secvențele de lumini ale cubului;
  - Dezvoltarea unei aplicații ESP32 care primeste comenzi Bluetooth si le trimite către placa Intel Galileo;
  - Utilizarea protocoalelor de comunicație între ESP32 și placa Galileo.

- Integrarea și Testarea Sistemului:
  - Testarea individuală a fiecărei componente pentru a asigura funcționarea corectă;
  - Integrarea componentelor hardware și software într-un sistem funcțional;
  - Realizarea testelor de sistem pentru a verifica fiabilitatea și performanța proiectului.

- Crearea unui Sistem Interactiv
  - Realizarea unui joc de lumini care poate fi controlat în timp real prin intermediul unui device.
  - Implementarea mai multor secvențe de lumini care să răspundă la diverse comenzi.

Resurse hardware

- ✓ Placa Intel Galileo Gen 2:

Se bazează pe Intel Quark SoC X1000, un sistem de clasă Intel Pentium pe un cip (SoC). Este prima placă bazată pe arhitectura Intel proiectată pentru a fi compatibilă din punct de vedere hardware și software cu pinii de pe shield-urile pentru Arduino Uno R3. Spre deosebire de Arduino, placa Galileo are mai multe porturi I/O standard pentru PC.

Galileo include un slot complet mini-PCI Express, port Ethernet 100Mb, header USB TTL UART, port USB Host, port USB Client și 8 MByte NOR flash și un suport de card microSD de până la 32GB. Procesorul este capabil să ruleze cod scris într-un limbaj de programare care este foarte similar cu limbajul C++. Suportă anumite distributii de Linux. Placa Intel Galileo Gen 2 se conectează la portul USB al calculatorului, folosind un cablu de tip USB Micro-B.

- ✓ ESP32 Development Board with Wi-Fi and Bluetooth 4.2 :

ESP32-WROOM-32 (ESP-WROOM-32) este un modul MCU (Microcontroller Unit) puternic, care oferă conectivitate Wi-Fi, Bluetooth și BLE (Bluetooth Low Energy), fiind destinat unei game largi de aplicații. Acestea variază de la rețele de senzori (IoT) cu consum redus de energie până la cele mai solicitante sarcini, cum ar fi codificarea vocală, streamingul de muzică și decodarea MP3.

Caracteristicile Modulului ESP32-WROOM-32:

- Chip-ul Central: ESP32-D0WDQ6 proiectat pentru a fi scalabil și adaptabil.
- CPU Dual-core: Două nuclee de procesare care pot fi controlate individual, cu frecvența ajustabilă între 80 MHz și 240 MHz.
- Co-procesor cu Consum Redus de Energie permite oprirea CPU-ului principal pentru a economisi energie și utilizarea co-procesorului pentru monitorizarea constantă a perifericelor.
- Conectivitate: WiFi 802.11 b/g/n permite conectarea la rețele WiFi pentru comunicații și transfer de date.
- Bluetooth v4.2 și BLE: Suportă atât Bluetooth Classic cât și Bluetooth Low Energy pentru conectivitate versatilă cu dispozitivele Bluetooth.
- Periferice și Interfețe:

Senzori Tactili Capacitivi: Pot fi utilizați pentru a detecta atingerea și implementarea interfețelor tactile.
Senzor Hall: Detectează câmpuri magnetice.
Interfață pentru Card SD: Permite adăugarea de stocare externă.
Interfață Ethernet: Pentru conectivitate rețea cu fir.
SPI de Mare Viteză: Interfață pentru comunicații rapide cu alte dispozitive.

- UART, I2S și I2C: Protocoale de comunicare standard pentru interfațarea cu diverse module și senzori.
- Caracteristici de Consum de Energie: Moduri de Economisire a Energiei: Permite configurarea modulului pentru consum redus de energie, util în aplicațiile portabile și IoT.
- Co-procesor ULP (Ultra Low Power): Poate monitoriza perifericele și senzorii cu un consum foarte redus de energie.

✓ Pentru construirea cubului s-au folosit următoarele:

- Led-uri (64 de led-uri de culoare verde):
  Specificații:
  - Lungimea pinului: >20 mm;
  - Diametru: 3 mm;
  - Tensiune directă: 0,6 V;
  - Curent direct: 20 mA.
- Fir de cupru:
  - Diametru conductor (inclusiv izolatia): Ø1,4mm;
  - Masa conductor: 0,25kg;
  - Lungime fir: 17m.

✓ Pentru circuit:

- 2 Breadboard -uri :

  Placa de tip breadboard poate fi utilizată pentru prototiparea rapidă a circuitelor electronice, fără a necesita lipire. Unele dintre cele 830 de puncte de contact ale acestei plăci sunt conectate orizontal, iar altele sunt conectate vertical, pentru o utilizare convenabilă.

  Specificații:

  - Dimensiuni: 16.5 x 5.4 x 0.85 cm
  - Număr de puncte de contact: 830
  - Diametrul firelor jumper compatibile: 0.8 mm

- Fire:
  Firele sunt perfecte pentru transmiterea semnalelor digitale sau analogice de intensitate redusă.

- 2 rezistențe (220 ohmi)

- CD4051BE CMOS Single 8-Channel Analog Multiplexer With Logic-Level Conversion:

Multiplexoarele și demultiplexoarele analogice CD4051BE sunt comutatoare analogice controlate digital. Aceste circuite de multiplexare disipă o putere mică pe întreaga plajă de tensiuni de alimentare V_DD - V_SS și V_DD - V_EE, independent de starea logică a semnalelor de control.

Resurse software

- ✓ Arduino IDE 1.8.19.

Arduino IDE (Integrated Development Environment) este un mediu de dezvoltare esențial pentru programarea și dezvoltarea proiectelor bazate pe plăcile de dezvoltare Arduino. Creat pentru a facilita programarea microcontrolerelor, Arduino IDE este ideal atât pentru începători, cât și pentru utilizatorii avansați, datorită interfeței sale intuitive și a suportului extensiv oferit de comunitate.

Caracteristici principale ale Arduino IDE:

- Interfața prietenoasă:

Arduino IDE oferă o interfață simplă și intuitivă, ușor de utilizat pentru toți utilizatorii. Interfața grafică include un editor de cod simplu, care dispune de funcții precum colorarea sintaxei și indentarea automată pentru a face codul mai lizibil.

- Compilare și încărcare:

Compilator integrat: Arduino IDE utilizează compilatoare GCC pentru a transforma codul sursă în fișiere binare executabile, care pot fi apoi încărcate pe plăcile Arduino.

- Funcționalitatea de încărcare directă:

Codul poate fi încărcat pe plăcile Arduino printr-un port USB cu un singur click.

- Biblioteci predefinite:

Biblioteca standard include un set vast de biblioteci care acoperă funcționalități comune, cum ar fi comunicarea serială, manipularea senzorilor și controlul motoarelor. Managerul de Biblioteci permite utilizatorilor să adauge și să gestioneze biblioteci suplimentare, oferind acces la o multitudine de biblioteci create de comunitatea Arduino.

- Serial Monitor:

Monitorul serial este un instrument util pentru depanare și interacțiune în timp real, care permite vizualizarea datelor trimise de placa Arduino prin portul serial.

- Manager de Plăci:

Suport pentru o gamă largă de plăci de dezvoltare Arduino și clone compatibile. Utilizatorii pot adăuga suport pentru noi plăci și microcontrolere folosind Managerul de Plăci.

- ✓ Biblioteca “BluetoothSerial” pentru comunicarea wireless.

- ✓ Aplicație pentru Android – “Serial Bluetooth Terminal”.

Implementare hardware

Am realizat cubul de leduri de dimensiuni 4x4x4 cu ajutorul a 32 de fire de cupru de 12 cm fiecare. Pe fiecare linie a fiecărui strat s-au conectat în serie anozii (A) a 4 led-uri, prin intermediul unui fir de cupru.
La capătul firului se aplică tesniunea de alimentare de 5V. Pe cealaltă parte, pe fiecare coloană s-au conectat în serie catozii (K) a 4 led-uri prin intermediul unui fir de cupru.

Astfel, în funcție de linia pe care aplicăm tensiune, respectiv de coloana pe care o legăm la ground, se va aprinde led-ul corespunzător (metoda este asemanatoare celei folosite pentru a citi tastele apăsate pe o tastatură).

În figura de mai jos este reprezentată schema electrică a unui layer din cub:

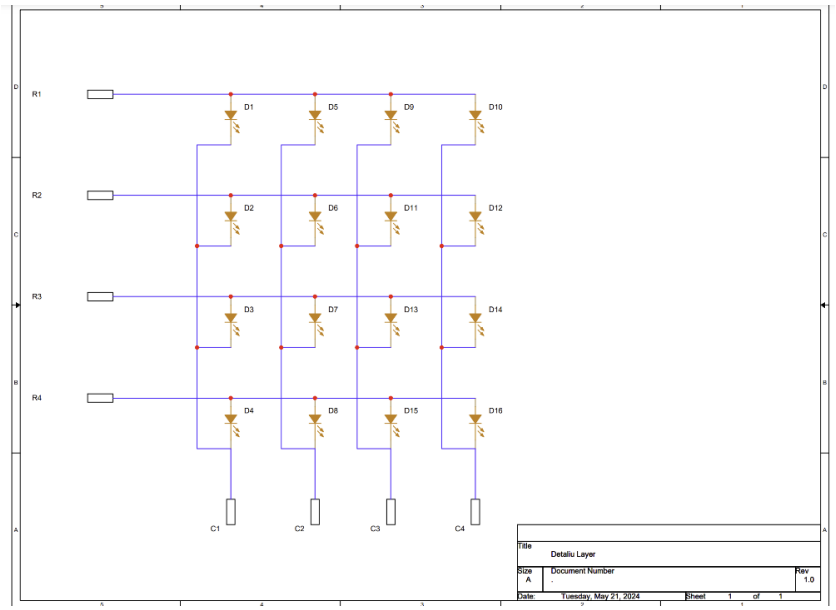

Figura 1. Schemă electrică layer cub.

Astfel, la aplicarea unei tensiuni de 5V pe una dintre intrările Rx (Row x) si la conectarea la masă (GND) a uneia dintre ieșirile Cy (Column y), prin intermediul comenzii multiplexoarelor, se va aprinde LED-ul corespunzător poziției XY.

Conexiunea dintre firele de cupru si led-uri a fost realizată prin intermediul unui cositor cu sacaz.

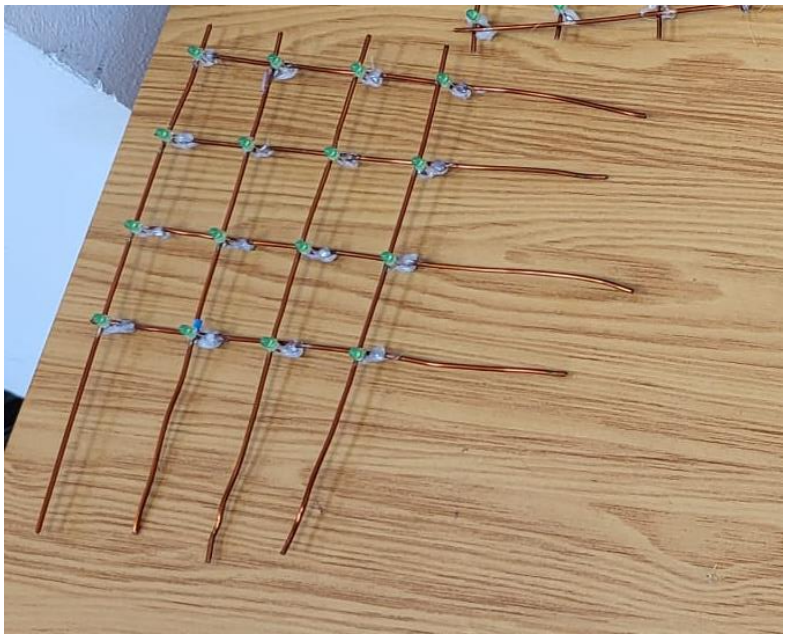

Figura 2. Un layer al cubului.

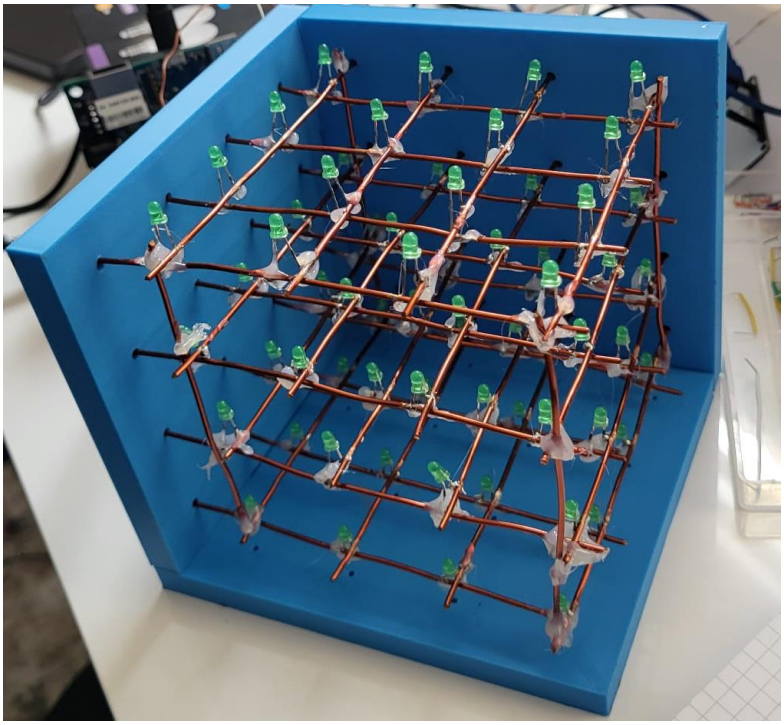

Figura 3. Cubul 4x4x4 cu suporți de plastic.

Toate firele de cupru sunt legate la ieșirile analogice a 4 demultiplexoare CD4051BE. Pentru primele două straturi, de jos în sus, au fost folosite 2 demultiplexoare. La pinul COM OUT/IN (pinul 3) al primului demultiplexor este conectată o rezistență de 220 ohmi și ieșirile sunt legate la cele 8 fire de cupru, ce comandă anozii celor 32 de led-uri (16 led-uri per strat). Cel de-al doilea are intrarea legată la masă și ieșirile sunt conectate la firele de legătură între catozii celor 32 de led-uri. În funcție de selecția celor două demultiplexoare se poate comanda aprinderea oricărui led de pe primele două straturi.

Se repetă același procedeu pentru următoarele două straturi, folosind celelalte două demultiplexoare CD4051BE.

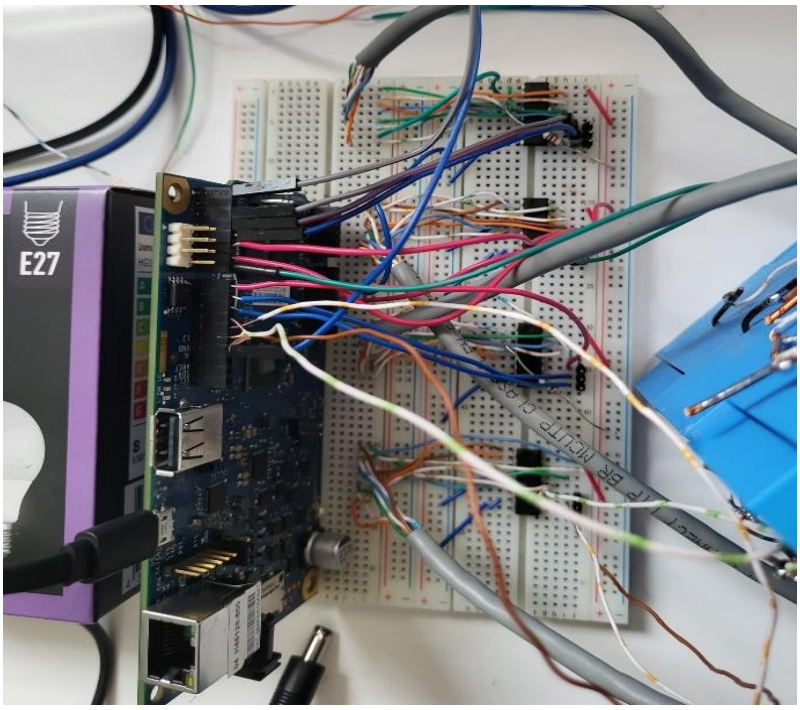

Figura 4. Circuit breadboard

Alimentarea celor 4 CD4051BE se face de la placa Intel Galileo Gen 2. Pinul VDD (pinul 16) al fiecărui demultiplexor este conectat la pinul 5V al plăcii, iar pinii VEE și VSS, la pinul de masă al plăcii de dezvoltare.

Pinii de selecție a celor 4 demultiplexoare sunt legați la ieșirile digitale ale plăcii Galileo, mai precis, de la pinul 2 la pinul 13. Primii 6 pini reprezintă biții de selecție pentru layer-ul 1 și layer-ul 2, iar următorii, până la pinul 13, sunt biții de selecție pentru layer-ele 3 și 4.

Pentru controlul wireless al jocurilor de lumini se stabilește o comunicație serială între modulul ESP32 și Intel Galileo Gen2. Se leagă între ei pinii de masă a celor două microcontrolere. Se stabilește prin fir conexiunea între pinul 0 al plăcii de dezvoltare, acesta fiind pinul de recepție prin UART (RX), la pinul TX0 de transmitere prin UART al modulului ESP32. Prin urmare, comanda primită prin Bluetooth de către ESP32 poate fi transmisă mai departe către Galileo.

Placa Intel Galileo primește alimentare de la priză, iar ESP32 este alimentată prin intermediul calculatorului.

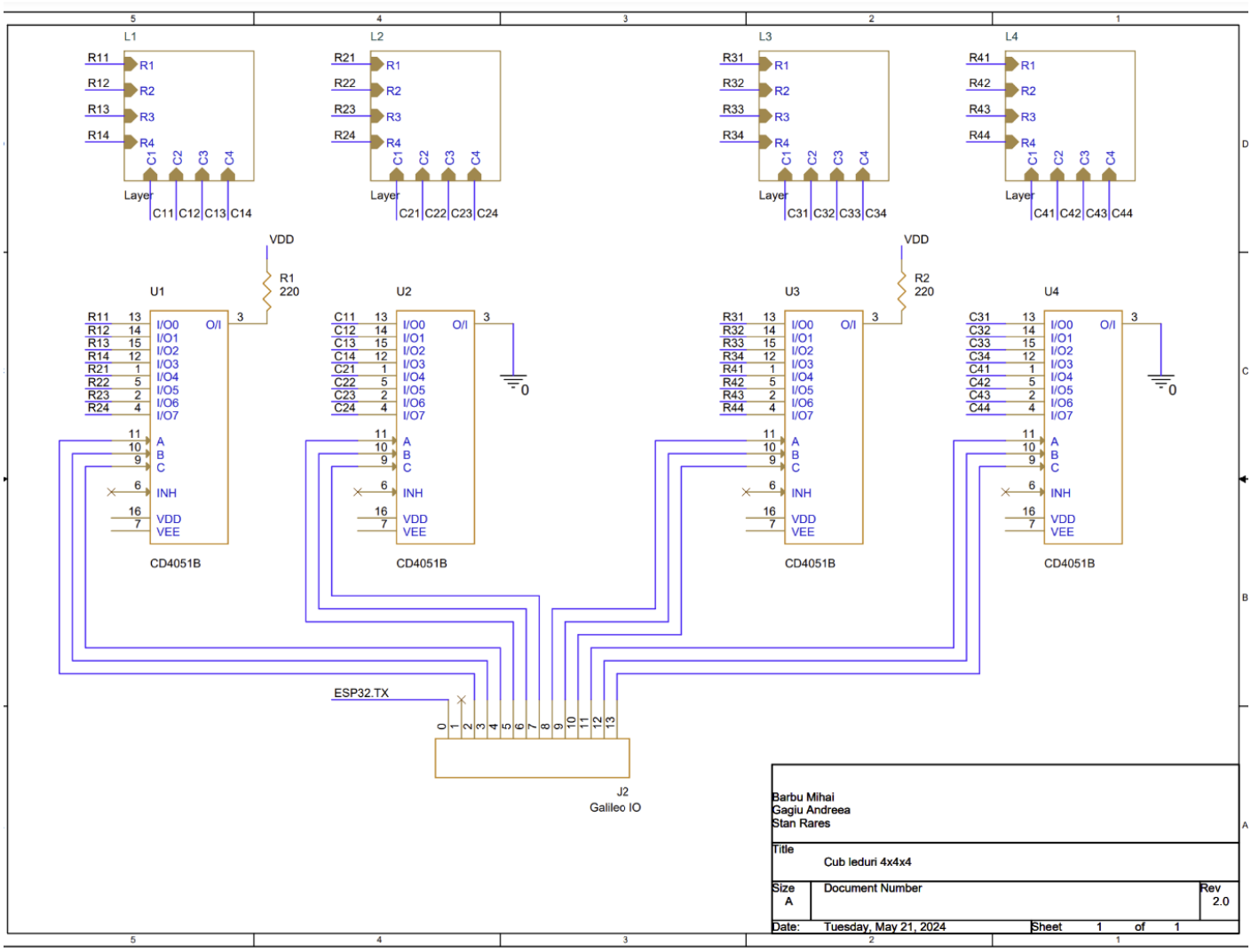

Figura 5. Schema electrică pentru comandarea cubului

Implementare software

Programul de mai jos este conceput pentru comunicarea Bluetooth între device-ul mobil și modulul ESP32, cât și pentru permiterea transmiterii prin UART către Galileo.

Am folosit biblioteca <*BluetoothSerial.h*> pentru comunicarea wireless.

În funcția de *setup()* se începe comunicația serială numită *"Serial1",* folosită pentru transmiterea prin UART, aceasta având un baudrate de 9600 bps. De asemenea, se începe comunicația serială wireless prin "*SerialBT"* și se stabilește numele device-ului utilizat.

Funcția *loop()* realizeză propagarea comenzii de la telefonul mobil la Intel Galileo Gen 2 prin *"Serial1"*.

```
#include <BluetoothSerial.h>
BluetoothSerial SerialBT;
char recieve;
void setup() {
  Serial1.begin(9600);
  Serial.begin(9600);
  SerialBT.begin("ESP32");
}

void loop() {
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
    Serial1.write(SerialBT.read());
  }
  delay(20);
}
```

Următorul program preia comanda transmisă prin *"Serial1"* și ,în funcție de valoarea acesteia, apelează una dintre cele 4 funcții ce creează jocurile de lumini.

În funcția de *setup()* se începe comunicația serială numită *"Serial1",* folosită pentru recepționarea prin UART, aceasta având un baudrate de 9600 bps. În bucla *for* se inițializează ieșirile digitale reprezentate de pinii 2-13.

```
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  for(int i=2;i<14;i++)
  {
    pinMode(i,OUTPUT);
  }
}
```

Funcția *led(int x, int y, int z)* primește ca argumente linia, coloana și layer led-ului pe care dorim să-l aprindem. Fiecare două layere sunt comandate de câte o pereche de multiplexoare. Astfel, cand z este 0 prima pereche de demultiplexoare va avea MSB = 0, variabilele x si y comandand linia, respectiv coloana LED-ului care se va aprinde pe primul layer. Cand z = 1 prima pereche are MSB = 1, astfel x si y comandand linia, respectiv coloana de pe layerul doi. Similar, pentru z = 2 a doua pereche de demultiplexoare va avea MSB = 0, iar pentru z = 3 a doua pereche va avea MSB = 1.

```
void led(int x, int y, int z)
{
  int pin2, pin3, pin4, pin5 , pin6, pin7, pin8, pin9, pin10, pin11, pin12, pin13;
    //bitRead(x,0);
    if(z==0)
    {
      pin4 = 0;
      pin7 = 0;
      digitalWrite(4,pin4);
      digitalWrite(7,pin7);
      //stabilim z=layer 0
      pin3=bitRead(x,1);
      pin2=bitRead(x,0);
      pin6=bitRead(y,1);
      pin5=bitRead(y,0);
      digitalWrite(3,pin3);
      digitalWrite(2,pin2);
      digitalWrite(6,pin6);
      digitalWrite(5,pin5);
    }
    else if(z==1)
    {
      pin4 = 1;
      pin7 = 1;
      digitalWrite(4,1);
      digitalWrite(7,1);
      //stabilim z=layer 1
      pin3=x/2;
      pin2=x%2;
      digitalWrite(3,pin3);
      digitalWrite(2,pin2);
      pin6=y/2;
      pin5=y%2;
      digitalWrite(6,pin6);
      digitalWrite(5,pin5);
     }
    else if(z==2)
    {
      pin10 = 0;
      pin13 = 0;
      digitalWrite(10,0);
      digitalWrite(13,0);
      //stabilim z=layer 2
      pin9=bitRead(x,1);
      pin8=bitRead(x,0);
      pin12=bitRead(y,1);
      pin11=bitRead(y,0);
      digitalWrite(9,pin9);
      digitalWrite(8,pin8);
      digitalWrite(12,pin12);
```

```
      digitalWrite(11,pin11);
    }
    else if(z==3)
    {
      pin10 = 1;
      pin13=1;
      digitalWrite(10,1);
      digitalWrite(13,1);
      //stabilim z=layer 3
      pin9=bitRead(x,1);
      pin8=bitRead(x,0);
      pin12=bitRead(y,1);
      pin11=bitRead(y,0);
      digitalWrite(9,pin9);
      digitalWrite(8,pin8);
      digitalWrite(12,pin12);
      digitalWrite(11,pin11);
    }
   //Debug


   /* Serial.print("S-au aprins pinii: ");
    Serial.println();

    Serial.println("Pinul 2:  " + (String)pin2);
    Serial.println("Pinul 3:  " + (String)pin3);
    Serial.println("Pinul 4:  " + (String)pin4);

    Serial.println("Pinul 5:  " + (String)pin5);
    Serial.println("Pinul 6:  " + (String)pin6);
    Serial.println("Pinul 7:  " + (String)pin7);

    Serial.println("Pinul 8:  " + (String)pin8);
    Serial.println("Pinul 9:  " + (String)pin9);
    Serial.println("Pinul 10:  " + (String)pin10);

    Serial.println("Pinul 11:  " + (String)pin11);
    Serial.println("Pinul 12:  " + (String)pin12);
    Serial.println("Pinul 13:  " + (String)pin13); */

}
```

Următoarele 4 funcții creează jocurie de lumini prin apelarea funcției *led()* într-un mod repetitiv și ordonat. Fiecare joc de lumini primește ca argumente unul sau mai multe delay-uri cu valori diferite, în funcție de animație și de viteza de parcurgere a acesteia.

Animație cu efect de spirală:

```
void spirala(int intarziere)
{
  delay(intarziere);
  int x=0, y=0,z=0;
  for(int i=0;i<10;i++)
  {
  for(z=0;z<4;z++)
  {
    for(x=0;x<4;x++)
    {
```

```
      y=0;
      led(x,y,z);
      delay(intarziere);
    }
    for(y=0;y<4;y++)
    {
      x=3;
      led(x,y,z);
      delay(intarziere);
    }
    for(x=3;x>=0;x--)
    {
      y=3;
      led(x,y,z);
      delay(intarziere);
    }
    for(y=3;y>=0;y--)
    {
      x=0;
      led(x,y,z);
      delay(intarziere);
    }
  }
  for(z=3;z>=0;z--)
  {
    for(y=0;y<4;y++)
    {
      x=0;
      led(x,y,z);
      delay(intarziere);
    }
    for(x=0;x<4;x++)
    {
      y=3;
      led(x,y,z);
      delay(intarziere);
    }
    for(y=3;y>=0;y--)
    {
      x=3;
      led(x,y,z);
      delay(intarziere);
    }
    for(x=3;x>=0;x--)
    {
      y=0;
      led(x,y,z);
      delay(intarziere);
    }
  }
  }
}
```

Animație de aprindere a tuturor led-urilor :

```
void on_all(int intarziere)
{
   int x=0, y=0,z=0;
   for(int i=0;i<300;i++)
   {
   for(z=0;z<2;z++)
   {
      for(x=0;x<4;x++)
      {
        for(y=0;y<4;y++)
        {
          led(x,y,z);
          led(x,y,z+2);
          delay(intarziere);
        }
      }
   }
   }

}
```

Animație cu efect de val :

```
void val(int intarziere_val, int intarziere_layer)
{
  delay(intarziere_val);
  int x=0, y=0,z=0;
  for(int j=0;j<10;j++)
  {
  for(z=0;z<4;z++)
  {
    if(z==0)
    {
      led(x,y,z);
    }
    if(z==1)
    {
      for(int i=0;i<40;i++)
      {
      y=1;
      for(x=0;x<=z;x++)
      {
        y=1;
        led(x,y,z);
        delay(intarziere_layer);
      }
      for(y=x-1;y>=0;y--)
      {
        x=z;
        led(x,y,z);
        delay(intarziere_layer);
      }
      }
    }
```

```
  if(z==2)
  {
    for(int i=0;i<20;i++)
    {
    y=2;
    for(x=0;x<=z;x++)
    {
      y=2;
      led(x,y,z);
      delay(intarziere_layer);
    }
    for(y=x-1;y>=0;y--)
    {
      x=z;
      led(x,y,z);
      delay(intarziere_layer);
    }
    }
  }
  if(z==3)
  {
    for(int i=0;i<20;i++)
    {
    y=3;
    for(x=0;x<=z;x++)
    {
      y=3;
      led(x,y,z);
      delay(intarziere_layer);
    }
    for(y=x-1;y>=0;y--)
    {
      x=z;
      led(x,y,z);
      delay(intarziere_layer);
    }
    }
  }
  delay(intarziere_val);
}
for(z=3;z>=0;z--)
{
  if(z==3)
  {
    for(int i=0;i<20;i++)
    {
    y=0;
    for(x=3;x>=0;x--)
    {
      y=0;
      led(x,y,z);
      delay(intarziere_layer);
    }
    for(y=x+1;y<=3;y++)
    {
      x=0;
      led(x,y,z);
      delay(intarziere_layer);
    }
```

```
      }
    }
    if(z==2)
    {
      for(int i=0;i<20;i++)
      {
      y=1;
      for(x=3;x>=1;x--)
      {
        y=1;
        led(x,y,z);
        delay(intarziere_layer);
      }
      for(y=x+1;y<=3;y++)
      {
        x=1;
        led(x,y,z);
        delay(intarziere_layer);
      }
      }
    }
    if(z==1)
    {
      for(int i=0;i<40;i++)
      {
      y=2;
      for(x=3;x>=2;x--)
      {
        y=2;
        led(x,y,z);
        delay(intarziere_layer);
      }
      for(y=x+1;y<=3;y++)
      {
        x=2;
        led(x,y,z);
        delay(intarziere_layer);
      }
      }
    }
    if(z==0)
    {
      x=3;
      y=3;
      z=0;
      led(x,y,z);
    }
    delay(intarziere_val);
  }
  }
}
```

Animație cu efect de clepsidră :

```
void clepsidra(int intarziere_val, int intarziere_layer)
{
  delay(intarziere_layer);
  int x=0, y=0,z=0;
    for(int j=0;j<10;j++)
  {
  for(z=0;z<4;z++)
  {
    if(z==0)
    {
      for(int i=0;i<40;i++)
      {
      for(x=0;x<4;x++)
      {
        y=0;
        led(x,y,z);
        delay(intarziere_layer);
      }
      for(y=0;y<4;y++)
      {
        x=3;
        led(x,y,z);
        delay(intarziere_layer);
      }
      for(x=3;x>=0;x--)
      {
        y=3;
        led(x,y,z);
        delay(intarziere_layer);
      }
      for(y=3;y>=0;y--)
      {
        x=0;
        led(x,y,z);
        delay(intarziere_layer);
      }
      }
    }
    if(z==1)
    {
      for(int i=0;i<50;i++)
      {
        for(x=1;x<3;x++)
        {
          for(y=1;y<3;y++)
          {
            led(x,y,z);
            delay(intarziere_layer);
          }
        }
      }
    }
    if(z==2)
    {
      for(int i=0;i<40;i++)
      {
```

```
    for(x=0;x<4;x++)
    {
      y=0;
      led(x,y,z);
      delay(intarziere_layer);
    }
    for(y=0;y<4;y++)
    {
      x=3;
      led(x,y,z);
      delay(intarziere_layer);
    }
    for(x=3;x>=0;x--)
    {
      y=3;
      led(x,y,z);
      delay(intarziere_layer);
    }
    for(y=3;y>=0;y--)
    {
      x=0;
      led(x,y,z);
      delay(intarziere_layer);
    }
    }
  }
  if(z==3)
  {
    for(int i=0;i<50;i++)
    {
    for(x=1;x<3;x++)
      {
        for(y=1;y<3;y++)
        {
          led(x,y,z);
          delay(intarziere_layer);
        }
      }
    }
  }

  delay(intarziere_val);
}
for(z=3;z>=0;z--)
{
  if(z==3)
  {
    for(int i=0;i<40;i++)
    {
    for(x=0;x<4;x++)
    {
      y=0;
      led(x,y,z);
      delay(intarziere_layer);
    }
    for(y=0;y<4;y++)
    {
      x=3;
      led(x,y,z);
```

```
    delay(intarziere_layer);
  }
  for(x=3;x>=0;x--)
  {
    y=3;
    led(x,y,z);
    delay(intarziere_layer);
  }
  for(y=3;y>=0;y--)
  {
    x=0;
    led(x,y,z);
    delay(intarziere_layer);
  }
  }
}
if(z==2)
{
  for(int i=0;i<50;i++)
  {
    for(x=1;x<3;x++)
    {
      for(y=1;y<3;y++)
      {
        led(x,y,z);
        delay(intarziere_layer);
      }
    }
  }
}
if(z==1)
{
  for(int i=0;i<40;i++)
  {
  for(x=0;x<4;x++)
  {
    y=0;
    led(x,y,z);
    delay(intarziere_layer);
  }
  for(y=0;y<4;y++)
  {
    x=3;
    led(x,y,z);
    delay(intarziere_layer);
  }
  for(x=3;x>=0;x--)
  {
    y=3;
    led(x,y,z);
    delay(intarziere_layer);
  }
  for(y=3;y>=0;y--)
  {
    x=0;
    led(x,y,z);
    delay(intarziere_layer);
  }
  }
```

```
    }
    if(z==0)
    {
      for(int i=0;i<50;i++)
      {
      for(x=1;x<3;x++)
        {
          for(y=1;y<3;y++)
          {
            led(x,y,z);
            delay(intarziere_layer);
          }
        }
      }
    }

    delay(intarziere_val);
  }
  }
}
```

Functia loop() realizează recepționarea din Serial1 a unui caracter. În funcție de ce cifra de la 1 la 4 este recepționată, se pornește una dintre cele 4 animații.

```
char c;
void loop() {
  if(Serial1.available()){
    c = (char)Serial1.read();
    Serial.println(c);
  }
    if(c=='1')
    {
      on_all(0.1);
    }
    if(c=='2')
    {
      spirala(10);
    }
    if(c=='3')
    {
      val(50,2);
    }
    if(c=='4')
    {
      clepsidra(10,10);
    }
    delay(1);

}
```

Concluzii

Cubul de led-uri a necesitat o planificare atentă a designului și a construcției în sine. Am întâmpinat dificultăți în cositorirea fiecărui led de firul de cupru. Deoarece firul a fost emailat, pentru a nu se realiza conexiuni accidentale între alimentări, ceea ce ar duce la distrugerea circuitelor, acesta a trebuit să fie șlefuit înainte de aplicarea aliajului de lipit .

În ceea ce privește partea de software, implementarea a fost mult mai rapidă, programul inițial fiind realizat înaintea suportului hardware (cubul de LEDuri), singurul dezavantaj fiind inabilitatea de a testa codul în absența cubului. Fiecare joc de lumini a necesitat o testare a mai multor valori pentru parametrii de întârziere pentru a crea o senzație vizuală plăcută (găsirea frecvenței critice pentru ca animațiile să pară continue).

În ciuda dificultăților pe care le-am întâmpinat, procesul de realizare a proiectului fost unul plăcut, din care am învățat lucruri interesante și utile, stârnindu-ne interesul pentru realizarea unor astfel de proiecte în viitor.

Bibliografie

- https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/
- https://www.youtube.com/watch?v=TesnKjba0l0&ab_channel=DIYBuilder
- https://www.circuitbasics.com/how-to-setup-an-led-matrix-on-the-arduino/
- https://www.instructables.com/Led-Cube-8x8x8/
- https://docs.arduino.cc/retired/getting-started-guides/IntelGalileoGen2/
- https://www.youtube.com/watch?v=_QETdyeMyZw&ab_channel=50Hz