Documentație Intersecție semaforizată
[Strada 20 Decembrie]

Andron Iulia Maria
Bolbotină Ioana Flavia
CTI-RO, anul II, grupa 1.1

Cuprins

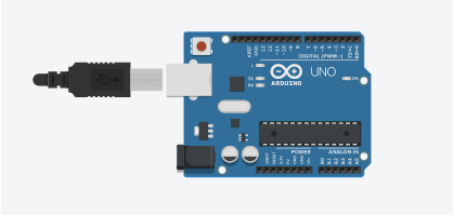
A. Componente utilizate

a. Tabel componente

Nume	Cantitate	Componentă
U1 U2	2	Arduino Uno R3
DB1_P	1	Yellow LED
R1 R2 R3 R16 R17 R18 R19 R20 R21 R4 R5 R6 R7 R8 R9	15	220 Ω Resistor
DB2_P DB3_P DB4_P DB5_P DB6_P DB7_P DB8_P DB9_P DB1_M DB2_M DB3_M DB4_M	12	LED RGB

b. Descriere componente

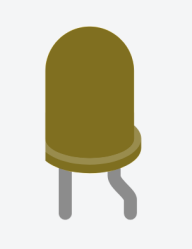
Arduino Uno R3



Board	**Name**	Arduino UNO R3
	SKU	A000066
Microcontroller	ATmega328P	
USB connector	USB-B	
Pins	**Built-in LED Pin**	13
	Digital I/O Pins	14
	Analog input pins	6
	PWM pins	6
Communication	**UART**	Yes
	I2C	Yes
	SPI	Yes
Power	**I/O Voltage**	5V
	Input voltage (nominal)	7-12V
	DC Current per I/O Pin	20 mA
	Power Supply Connector	Barrel Plug

Clock speed	**Main Processor**	ATmega328P 16 MHz
	USB-Serial Processor	ATmega16U2 16 MHz
Memory	**ATmega328P**	2KB SRAM, 32KB FLASH, 1KB EEPROM
Dimensions	**Weight**	25 g
	Width	53.4 mm
	Length	68.6 mm

LED



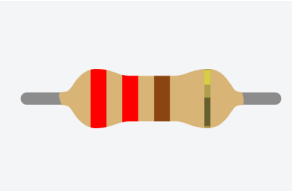
Absolute Maximum Ratings (T_a = 25°C)

Parameter	Maximum	Unit
Continuous Forward Current	30	mA
Derating Linear From 50°C	0.4	mA/°C
Reverse Voltage	5	V
Operating Temperature Range	-40°C to +85°C	
Lead Soldering Temperature [4mm (0.157 inch) from body]	260°C for 5 Seconds	

Electrical/Optical Characteristics at T_a = 25°C

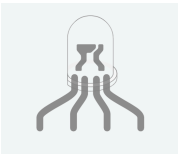
Parameter	Symbol	Minimum	Typical	Maximum	Unit	Test
Luminous Intensity	I_V	-	9.8	-	mcd	I_f = 20mA
Peak Emission Wavelength	λP	-	587	595	nm	Measurement at Peak
Dominant Wavelength	λD	-	590	-		I_f = 20mA-
Operating Voltage	V_{dd}	3	5	10	V	-
Blinking Frequency	F_{blk}	2	2.4	2.8	Hz	-
Reverse Current	I_R	-	-	100	µA	V_R = 5V

Rezistență



Resistance (Ohms)	220
Power (Watts)	0.25W, 1/4W
Tolerance	±5%
Packaging	Bulk
Composition	Carbon Film
Temperature Coefficient	350ppm /°C
Lead Free Status	Lead Free
RoHS Status	RoHS Compliant

LED RGB



Caracteristici și Specificații

- R - Terminal pentru culoarea roșu
- G - Terminal pentru culoarea verde
- B - Terminal pentru culoarea albastru
- (+) - GND Terminal de Catod Comun
- Rezistență termică scăzută
- Fără raze UV
- Flux de ieșire super mare și luminanță ridicată

- Curent direct pentru culorile roșu, albastru și verde: 20mA
- Tensiune directă
    - Roșu: 2v (tipic)
    - Albastru: 3,2 (tipic)
    - Verde: 3,2 (tipic)
- Intensitate luminoasă
    - Roșu: 800 mcd
    - Albastru: 4000 mcd
    - Verde: 900 mcd
- Lungime de undă
    - Roșu: 625 nm
    - Albastru: 520 nm
    - Verde: 467,5 nm
- Temperatură de operare: -25 ℃ până la 85 ℃
- Temperatura de depozitare: -30 ℃ până la 85 ℃

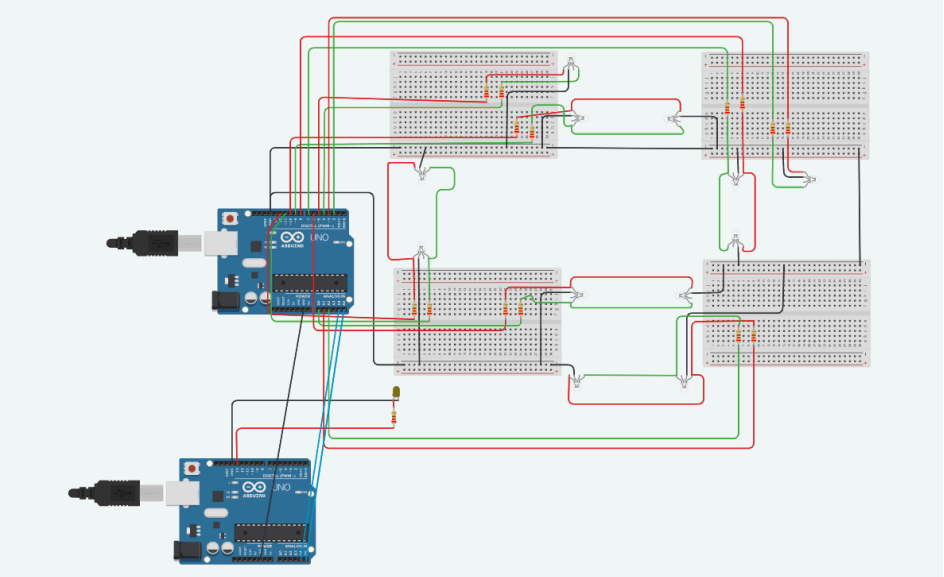
c. Pini utilizați

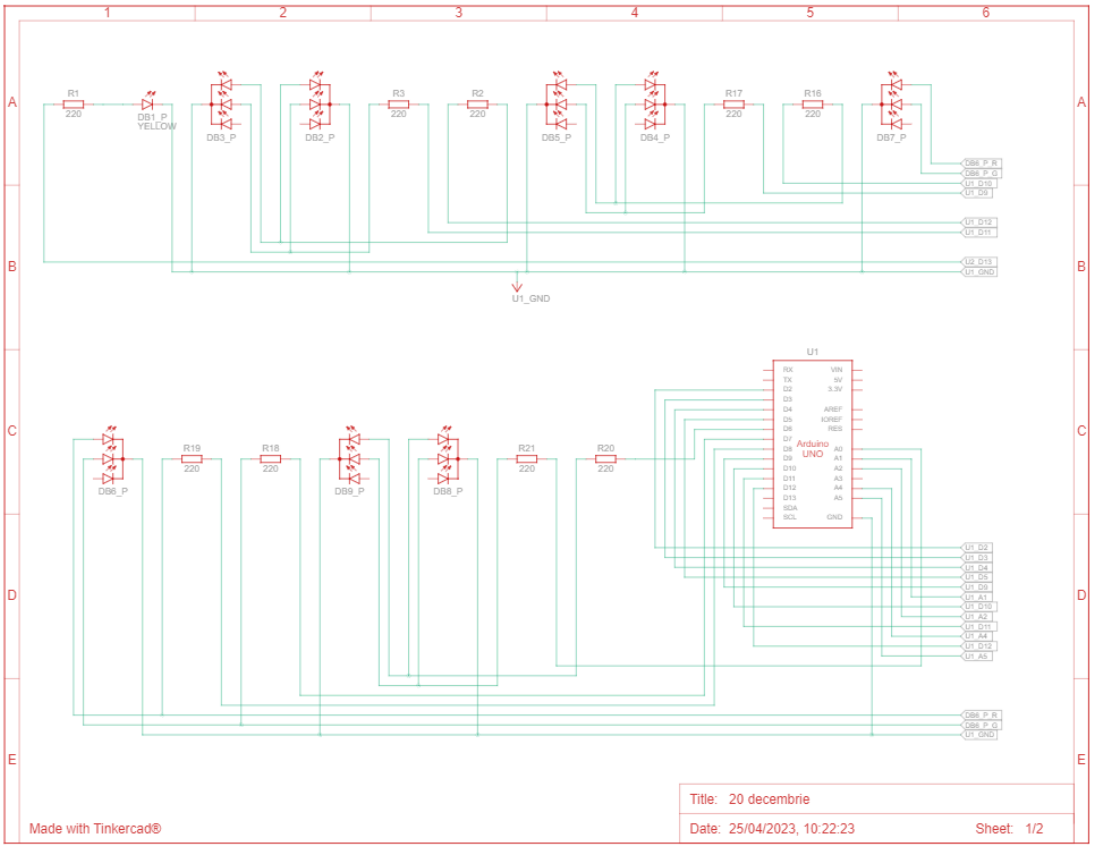
Plăcuță Arduino U2 :

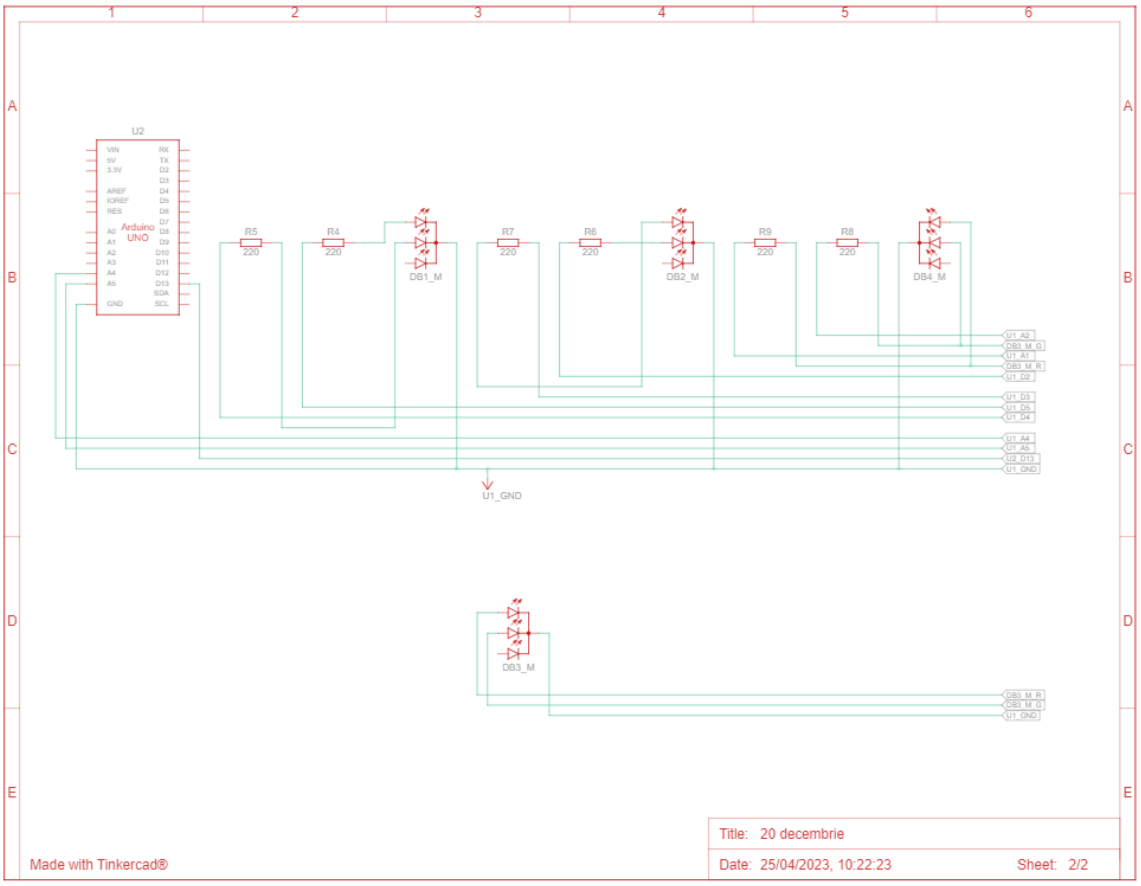
B1_P- pinul 13

Plăcuță Arduino U1 :
B23_r_P - pinul 12;
B23_g_P - pinul 11;
B45_r_P - pinul 10;
B45_g_P - pinul 9;
B67_r_P - pinul 8;
B67_g_P - pinul 7;
B89_r_P - pinul 6;
B89_g_P - pinul 14;
B1_r_M - pinul 5;
B1_g_M - pinul 4;
B2_r_M - pinul 3;
B2_g_M - pinul 2;
B34_r_M - pinul 15;
B34_g_M - pinul 16;

B. Arhitectură Hardware



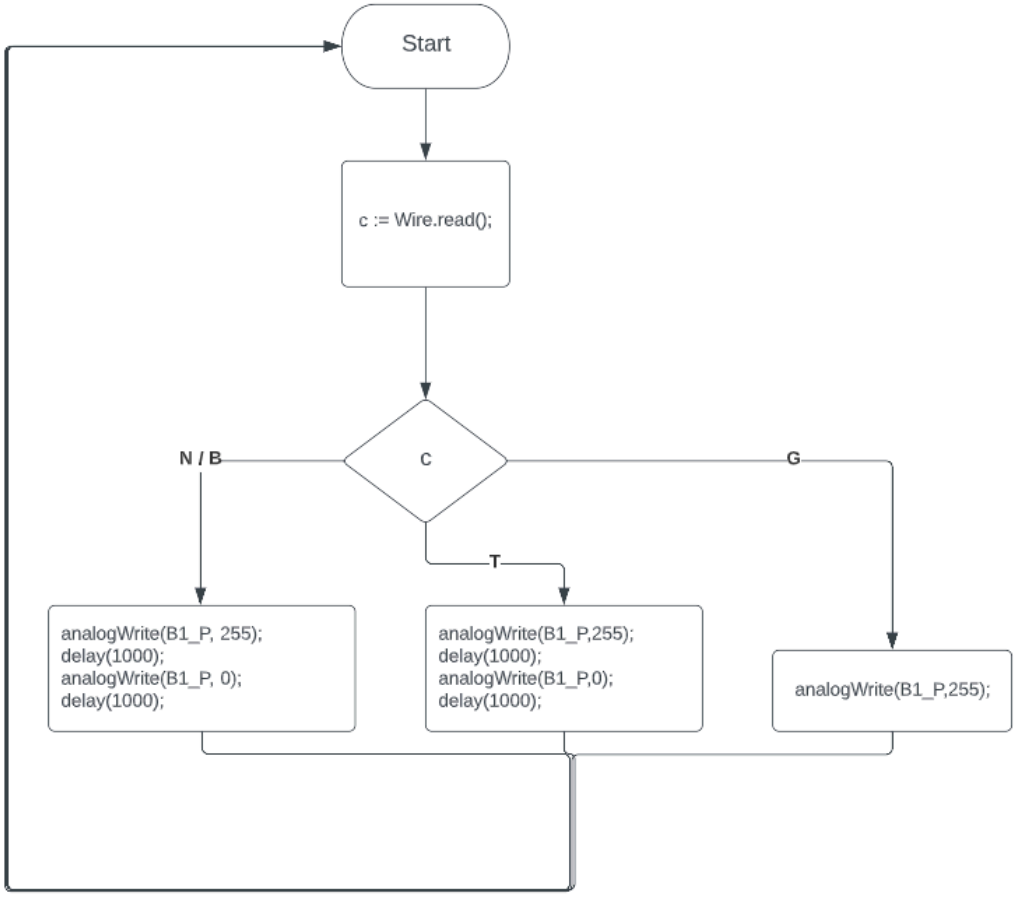



1
2
3
4
5
6
A
B
C
D
E
U2
Arduino UNO
R5
220
R4
220
DB1_M
R7
220
R6
220
DB2_M
R9
220
R8
220
DB4_M
U1_GND
DB3_M
Made with Tinkercad®
Title: 20 decembrie
Date: 25/04/2023, 10:22:23
Sheet: 2/2

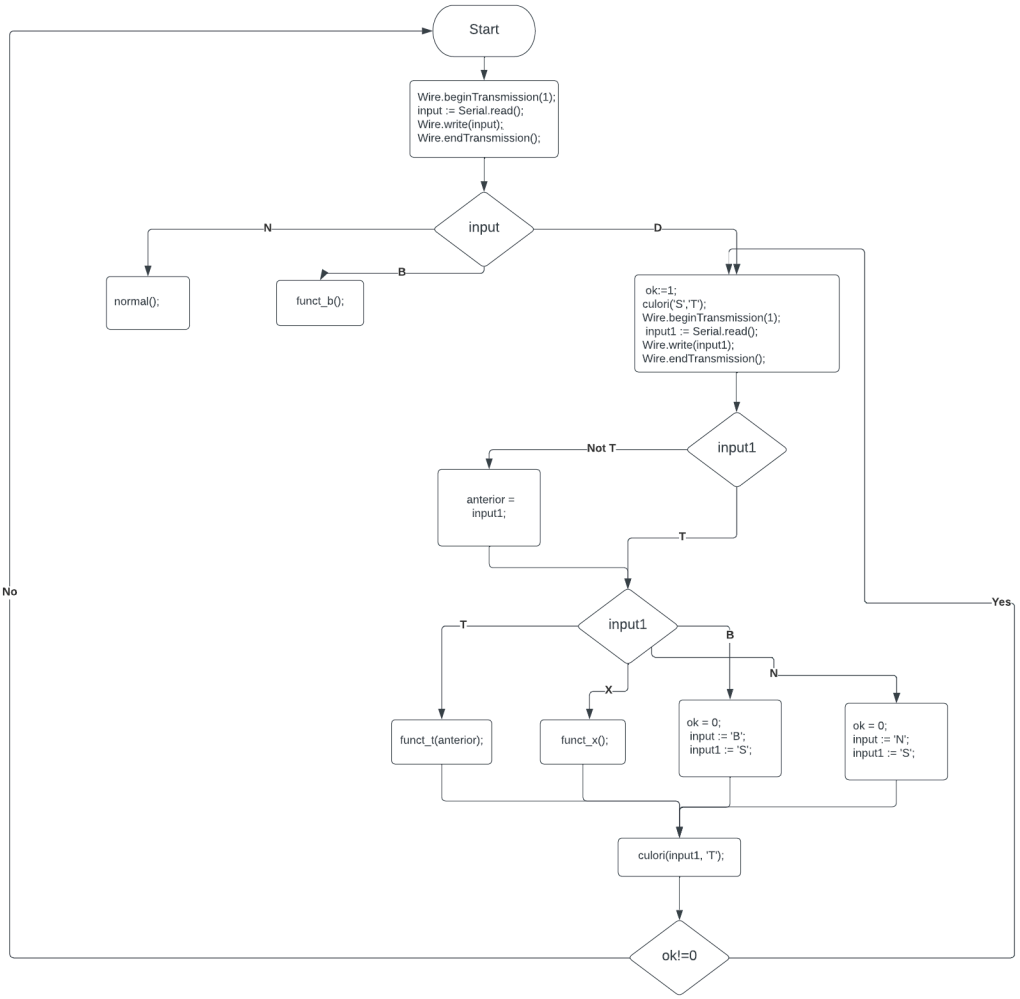
C. Arhitectura Software

a. Schemă logică

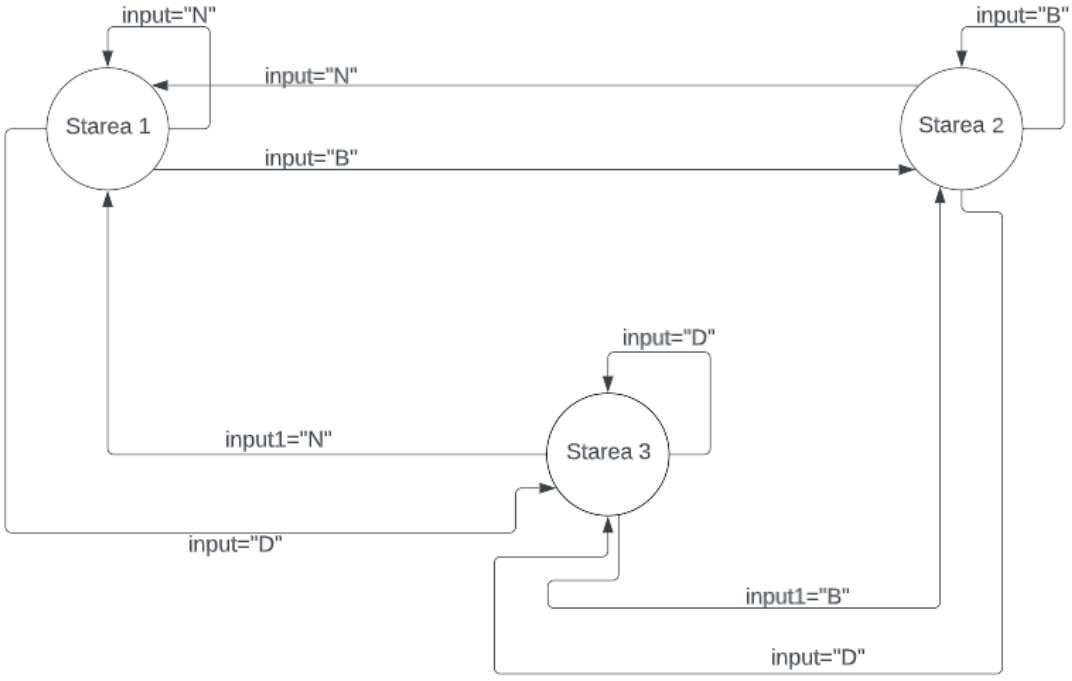
Schema logica plăcuță U2

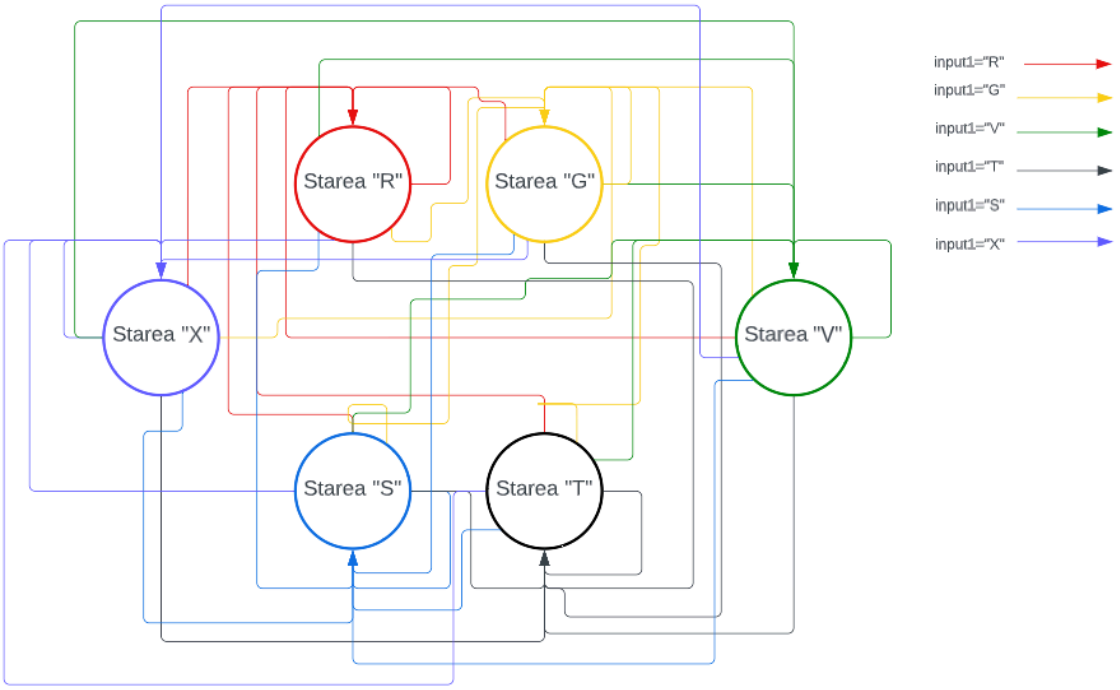


Schemă logică plăcuță U1



b. Diagramă de stare





Specificații:

- Starea 1 activată prin trimiterea caracterului „N” pe monitorul serial: Sistemul funcționează în modul „normal”, în acest mod toate semafoarele funcționează cum au fost programate.
- Starea 2 activată prin trimiterea caracterului „B” pe monitorul serial: Sistemul este în modul „blinking” sau “intermitent", în acest mod toate semafoarele au culoarea galbenă intermitentă cu o latență de 1 secunda aprins și 1 secundă stins.
- Starea 3 activată prin trimiterea caracterului „D” pe monitorul serial: Sistemul este în modul „diagnoza”, în acest mod toate semafoarele sunt stinse.
- În stare 3:
    - La primirea pe serial monitor a caracterului „R” toate semafoarele se vor aprinde cu culoarea roșie.
    - La primirea pe serial monitor a caracterului „G” toate semafoarele se vor aprinde cu culoarea galbenă.
    - La primirea pe serial monitor a caracterului „V” toate semafoarele se vor aprinde cu culoarea verde.
    - La primirea pe serial monitor a caracterului „T” semafoarele vor începe să se aprindă și să se stingă cu o latență de 1 secundă cu culoarea care a fost aleasă înainte, dacă nu a fost aleasă nici o culoare în mod implicit se va folosi galben.
    - La primirea pe serial monitor a caracterului „S” toate semafoarele se vor stinge.
    - La primirea pe serial monitor a caracterului „X” semafoarele vor trece prin toate culorile (pentru mașini: stins, roșu, galben, verde, iar pentru pietoni: stins, roșu, stins, verde) la fiecare 2 secunde.
    - Pentru a ieși din starea de diagnoză se trimite fie „N” fie „B” pentru a reveni în starea 1 sau 2.
    - Semaforul care are doar culoarea galbenă în modul de diagnoza va face funcția de diagnoza doar pentru caracterele „T” sau „G”.

c. Cod

Placuța Arduino U1

```
//definire întărzieri
                    globale
#define DELAY1 1000    12; // tipul numele = pinul
#define DELAY2 2000    11;
#define DELAY3 5000    10;
#define DELAY4 15000
#define DELAY5 20000
```

```
void setup() { //setează pinii, inițializează variabile, începe să utilizeze biblioteci
     pinMode(B34_g_M, OUTPUT);
     //...
     Wire.begin(); //începe o legătură
     Serial.begin(9600);//începe transmiterea serială
 }

//seteaza culoarea becului RGB primit ca referinta
void setColor(const int P1, const int P2, int R, int G) {
    analogWrite(P1, R);
    analogWrite(P2, G);
}


// functie pentru setarea semafoarelor
void culori(char c, char s) {
    int r = 0, g = 0;
    switch (c) { // switch case pentru cele 3 culori de semafor si cand e stins
    case 'V': r = 0; g = 255; break;
    case 'R': r = 255; g = 0; break;
    case 'G': r = 255; g = 255; break;
    case 'S': r = 0; g = 0; break;
    }
    if (s == 'T') { //seteaza toate semafoarele
    //...


    if (s == 'M') { //seteaza semafoarele pentru masini
    //...


     if (s == 'P') { //seteaza semafoarele pentru pietoni
     //...


bool delay_n(int d){
  //funcție care primește ca parametru un delay și daca se citește
  //alt caracter de la monitorul serial returnează true
    for(int i=0;i<d;i=i+100)
    {
      delay(100);
      if(Serial.available()) {
        input = Serial.read();
        return true;
        }
    }
}


bool delay_n1(int d){
  //funcție care primește ca parametru un delay și daca se citește
  //alt caracter de la monitorul serial returnează true
    for(int i=0;i<d;i=i+100)
    {
      delay(100);
      if(Serial.available()) {
        Wire.beginTransmission(1);
         if(input1!='T'&&input1!='X')
             anterior=input1;
        input1 = Serial.read();
        Wire.write(input1);
        Wire.endTransmission();
        return true;
        }
    }
  return false;
}
```

```
void normal() // starea 1/functionarea normala
{
    setColor(B23_r_P, B23_g_P, 0, 255);
    setColor(B45_r_P, B45_g_P, 255, 0);
    setColor(B67_r_P, B67_g_P, 0, 255);
    setColor(B89_r_P, B89_g_P, 255, 0);
    setColor(B1_r_M, B1_g_M, 0, 255);
    setColor(B2_r_M, B2_g_M, 255, 0);
    setColor(B34_r_M, B34_g_M, 0, 255);
    //dacă este true se iese din funcţie
    if(delay_n(DELAY4)==true)
      return;
    //...


void funct_x() //starea 3, la primirea caracterului 'X'
//...


void funct_t(char anterior) //starea 3, la primirea caracterului 'T'
//...


 void funct_b() //starea 2 /modul intermitent
 //...


void loop() {
    Wire.beginTransmission(1);
    if (Serial.available()) {
        Wire.beginTransmission(1); //se incepe comunicarea intre cele 2 placute
        input = Serial.read(); //se citeste un caracter de la monitorul serial
        Serial.print("You typed:");
        Serial.println(input);
        Wire.write(input); //se scriu date de la un dispozitiv periferic în răspuns la o cerere din partea unui dispozitiv de control
        Wire.endTransmission();//se incheie transmisia
    }
    if (input == 'N') { //starea 1
        normal();
    }
    if (input == 'B') { //starea 2
        funct_b();

    }
    if (input == 'D') { //starea 3
        ok = 1;
        culori('S', 'T'); //se sting toate semafoarele
        do {
            if (Serial.available()) {
              //...


                if (input1 != 'T') {  //se retine inputul anterior
                    anterior = input1;
                }
            }
            if (input1 == 'T') { // se executa funct_t avand ca parametru inputul anterior
                funct_t(anterior);
            }
            culori(input1, 'T');
            if (input1 == 'X') {
                funct_x();
            }

            if (input1 == 'B') { //ok se pune pe 0 si se iese din do while
                ok = 0;
                input = 'B';
                input1 = 'S';
            }
            if (input1 == 'N') { //ok se pune pe 0 si se iese din do while
                ok = 0;
                input = 'N';
                input1 = 'S';
            }
        } while (ok != 0);
    }


}
```

Plăcuța U2

```
#include <Wire.h>
//inițializare variabilă globală
const int B1_P=13;
```

```
void setup() //setează pinii, inițializează variabile, începe să utilizeze biblioteci
{
  pinMode(B1_P, OUTPUT);
  Wire.begin(1);//începe comunicarea
  Wire.onReceive(receiveEvent);//înregistrează o funcție care este apelată când se primește o informație
}


void receiveEvent(int  howMany) { //primește informații în format de biți de la plăcuța U1
  if(Wire.available())// verifică dacă există informație transmisă de la U1
    c = Wire.read();// primește bitul ca și caracter
  if(c!='T'&&c!='X')
    ant=c;
}


void loop() {

    if (c == 'B' || c== 'N') { //stările N și B(Starea 1 și Starea 2)
        analogWrite(B1_P, 255);
        delay(1000);
        analogWrite(B1_P, 0);
        delay(1000);

    }
        //Starea 3
            if (c == 'T') { //Starea "T"
              analogWrite(B1_P,255);
              delay(1000);
              analogWrite(B1_P,0);
              delay(1000);
            }
            if(c == 'G'){ //Starea "G"
                analogWrite(B1_P,255);
            }
    }
```

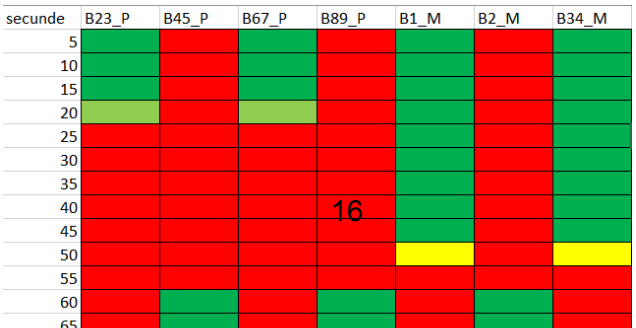
D. Concluzii

a. Dificultăți

Dificultățile întâlnite au fost legate de implementarea stărilor și funcționarea în paralel în cazul stării „N”.

Dificultățile întâlnite în implementarea stărilor au fost stabilirea timpul optim de funcționare al intersecției și sincronizarea tuturor semafoarelor în cazul stării „N”.

Rezolvarea acestei dificultăți a fost realizarea unui tabel excel în care am pus semafoarele și timpul lor de funcționare în paralel cu restul. Astfel am văzut cum se sincronizează și cât de mult ar trebui să funcționeze fiecare bec. În acest tabel culoarea verde deschis reprezintă verde intermitent, iar restul culorilor arată culoarea semaforului.



Dificultatea a fost găsirea unei soluții pentru funcționarea în paralel a becului intermitent cu restul semafoarelor din intersecție, în starea „N”. Această problemă s-a creat datorită faptului că majoritatea semafoarelor aveau un timp mare de secunde până la următoarea schimbare, iar becul B1_P trebuie să își modifice valoarea la fiecare secundă. Am decis să folosim o plăcuță separată pentru becul intermitent, pentru a funcționa în paralel. Astfel, am folosit un „wire” ca soluție. Legarea printr-un „wire” a constat în folosirea librăriei <Wire.h> și folosirea instrucțiunilor Wire.begin(), Wire.beginTransmission(), Wire.write() și Wire.endTransmission().

b. Optimizare

O optimizare a fost definirea globală a delay-urilor de dimensiune mare.

Alta a constat în folosire unei funcții „culori” cu switch case pentru setarea semafoarelor pe verde, roșu, galben sau pentru stingerea lor în funcție de inputul primit.

Pentru lizibilitatea codului în loop, am făcut funcții pentru funcționarea normală, intermitentă și de diagnoză, în sub stările când input1 este egal cu„X” sau „T”.

În timpul funcționării normale, dacă se primește alt caracter de la serial monitor, prin intermediul funcției „delay_n()” se oprește funcția de „normal”.