Universitatea POLITEHNICA din Bucureşti
Facultatea de electronică, Telecomunicații și Tehnologia Informației

PROIECT 2

SEMAFOR INTELIGENT CU 3 FAZE

PROFESOR INDRUMĂTOR:

Ş.l.Dr.Ing. ADRIAN FLORIN PĂUN

STUDENȚI:

ANDRONE IONUȚ

DAMIAN DORINA-MARIANA

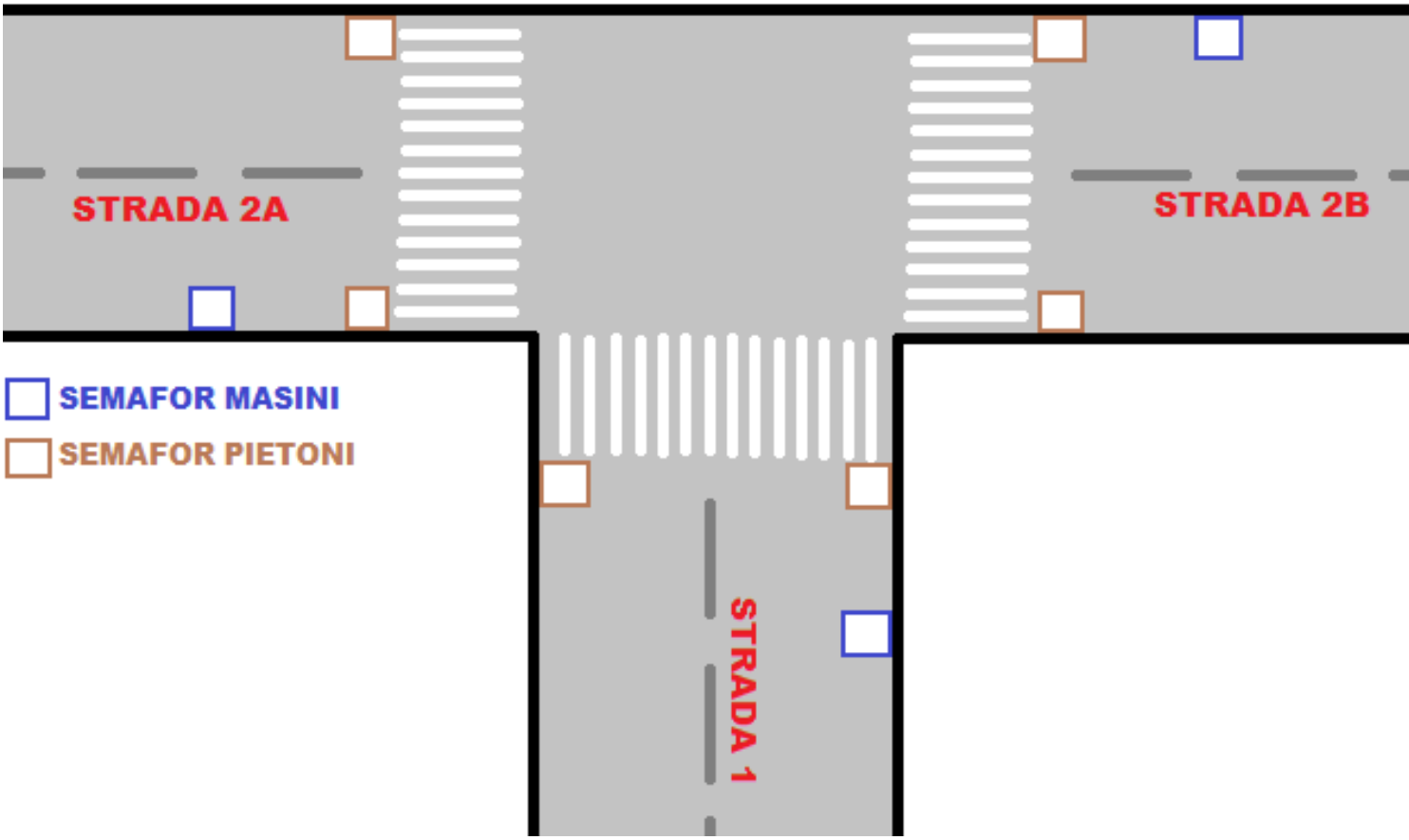
GRUPA 432C

CUPRINSUL PROIECTULUI

1. Tema proiectului

Tema proiectului consta in proiectarea unei intersectii cu 3 strazi. Am ales modelul intersectiei “in T”. Pentru trecerile de pietoni sunt atribuite semafoare cu doua culori (rosu si verde), iar pentru autoturisme vor fi atribuite semafoare cu trei culori(rosu, galben si verde). Perioada ciclului de faze este fixa, durata fiecarei faze fiind reglabila prin hypeterminal (intre 10% si 80% dintr-un ciclu, cu pasul de reglare 10%).

Pentru proiectarea intersectiei vom folosi urmatoarea schita:

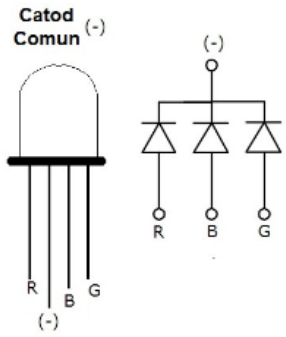


2. Lista de materiale folosite

REF	Denumire	VALOARE	CAPSULA	CANTITATE	FURNIZOR
C1	Condensator	1uF	THT	1	UPB
C2, C3	Condensator	18pF	SMD 0805	2	UPB
C4	Condensator	10uF	SMD 0805	1	UPB
C5, C6, C13	Condensator	100nF	SMD 0805	3	UPB
R1	Rezistor	10kΩ	SMD 0805	1	UPB
R2	Rezistor	470Ω	SMD 0805	1	UPB
D2	Led		THT	1	UPB
SW1, SW2	Switch		THT	2	UPB
X1	Cristal de cuart	20 MHz	SMD	1	UPB
CN2	Conector		THT	1	UPB
U1	Atmega164A		THT	1	UPB
LED2-LED10	LED RGB CATOD COMUN		THT	9	OPTIMUS DIGITAL
R3 – R12	Rezistor	510Ω	THT	10	OPTIMUS DIGITAL

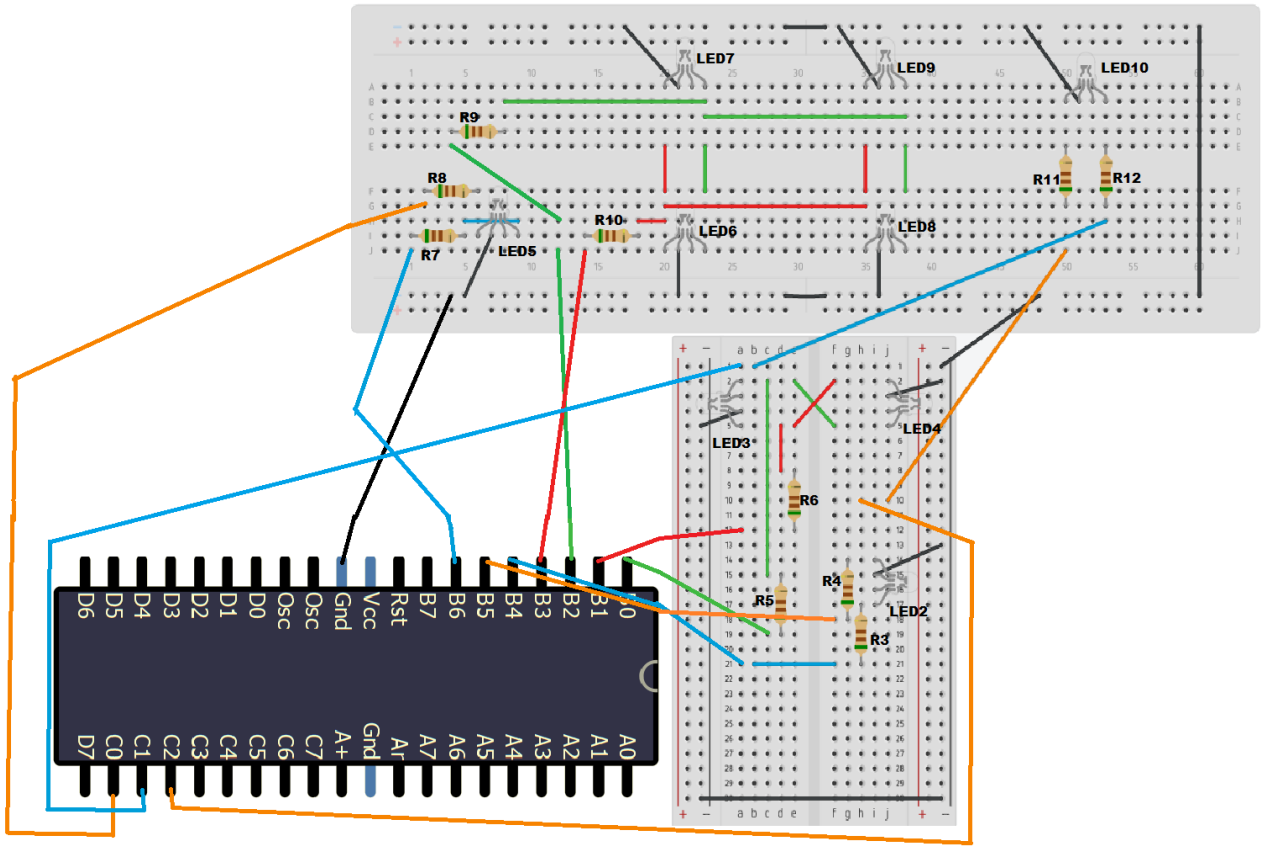
3. STRUCTURA HARDWARE A INTERSECTIEI PROIECTATE

Pentru simularea functionarii intersectiei proiectate, am folosit simulatorul online TinkerCad. Am ales acest simulator de circuite deoarece in biblioteca sa regasim o componenta ce se va utiliza in realizarea proiectului, si anume led-ul RGB. Dar o observatie importanta consta in faptul ca in TinkerCad, ledul RGB prezinta alta ordine a terminalelor decat cele achizitionate THT, si anume:

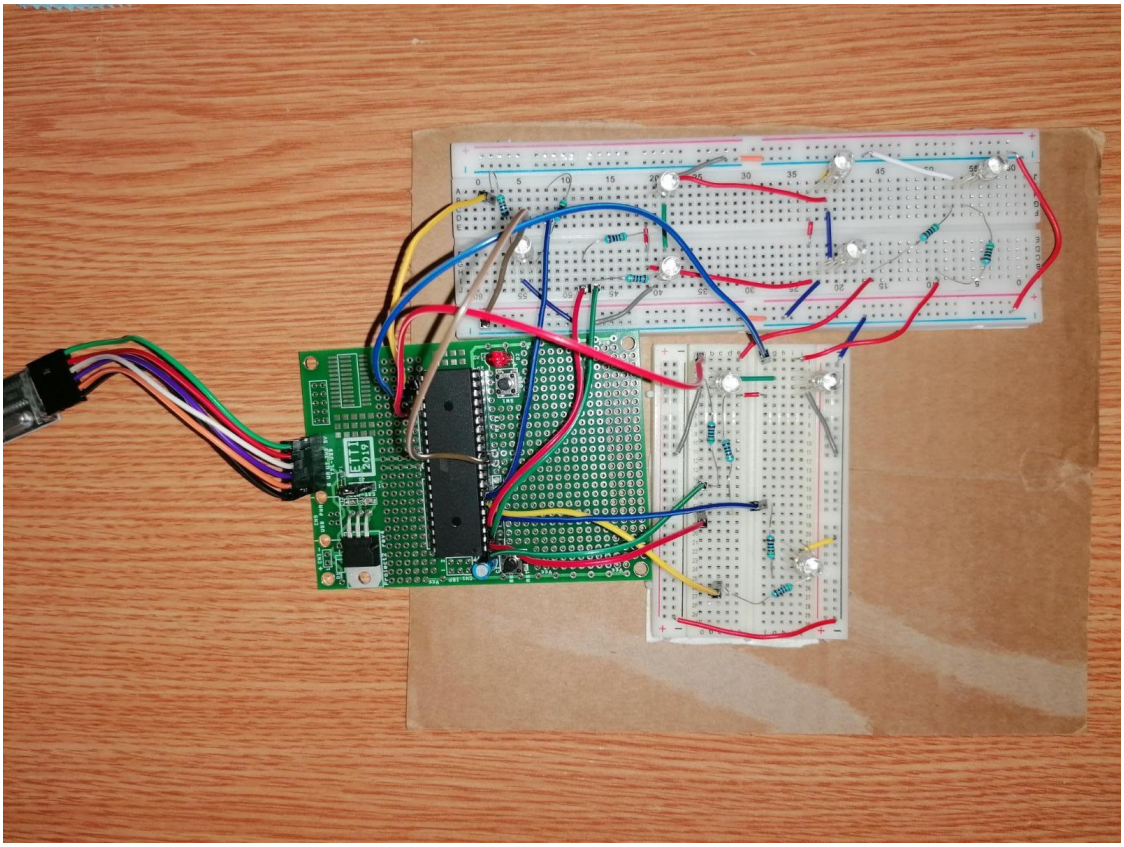


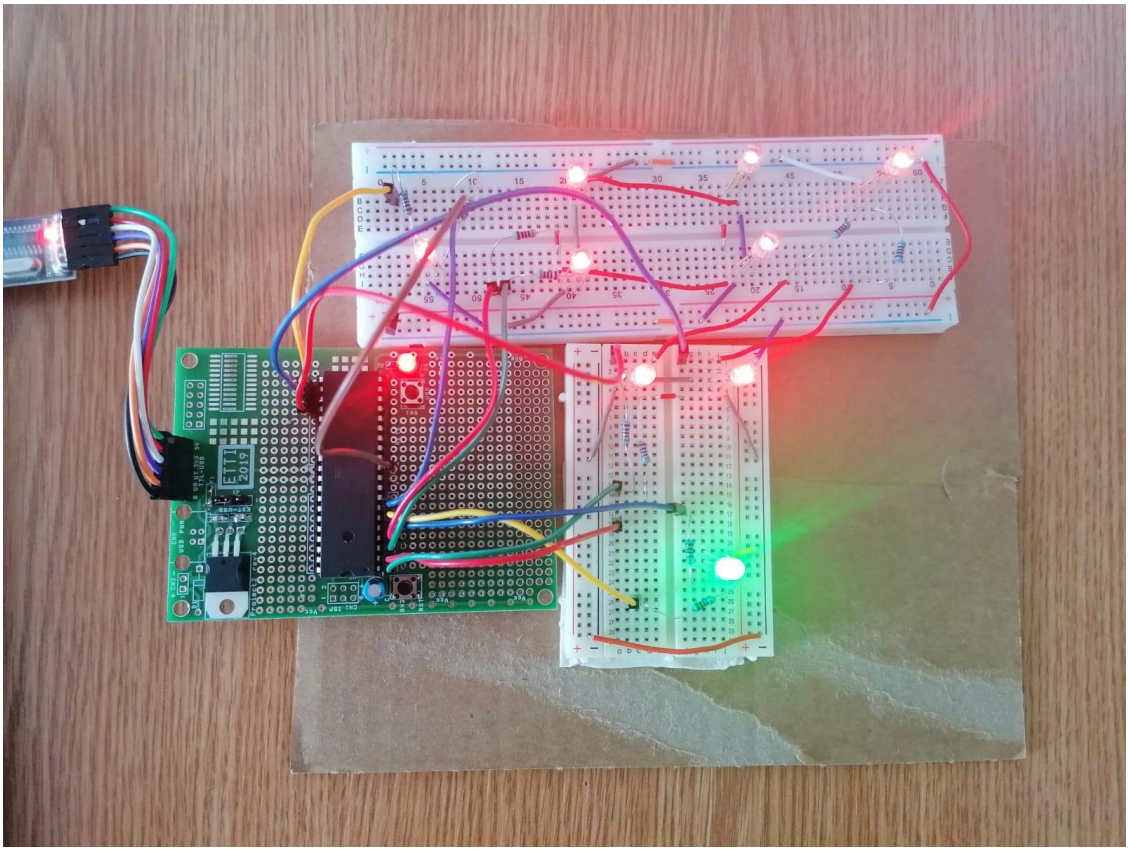
Functionalitatea acestor tipuri de leduri va fi analizata in sectiunile urmatoare ale proiectului.

Schema realizata in simulator, facand abstractie de pinii pe care nu ii utilizam sau au alt rol in proiect este:



Schema realizata fizic arata in felul urmator:





4. DESCRIEREA MICROCONTROLLERULUI ATMEGA164a

4.1. Microntrollerul ATMEGA164A – generalitati

Microcontrolerul AVR pe 8 biti, Atmega164A, produs de Atmel, are la baza un procesor RISC, adica contine un set de comenzi simple si rapide, in care viteza creste datorita simplificarii instructiunilor, cu arhitectura Harvard, ceea ce inseamna ca memoria de date si memoria de program folosesc magistrale diferite, astfel eficienta este sporita.

Acesta are trei tipuri de memorii integrate : FLASH – folosita ca memorie de program, EEPROM (Electrically Erasable Programmable Read-Only Memory), de tip nevolatila, utilizata pentru stocarea constantelor numerice care trebuie sa ramana nealterate si RAM (Random Access Memory), de tip volatila.

Caracteristicile principale ale acestui microcontroler sunt reprezentate de urmatoarele: set de 131 de intructiuni scrise în limbajul de programare C, 32 de registre de uz general direct adresabile de unitatea artimetrică si logică (UAL) a cate 8 biti fiecare, frecventa de lucru care poate fi controlată din software de la 0 la ~20 MHz, timer programabil, surse externe si interne de intrerupere, oscilator RC integrat, tensiune de alimentare intre 1.8 până la 5V.

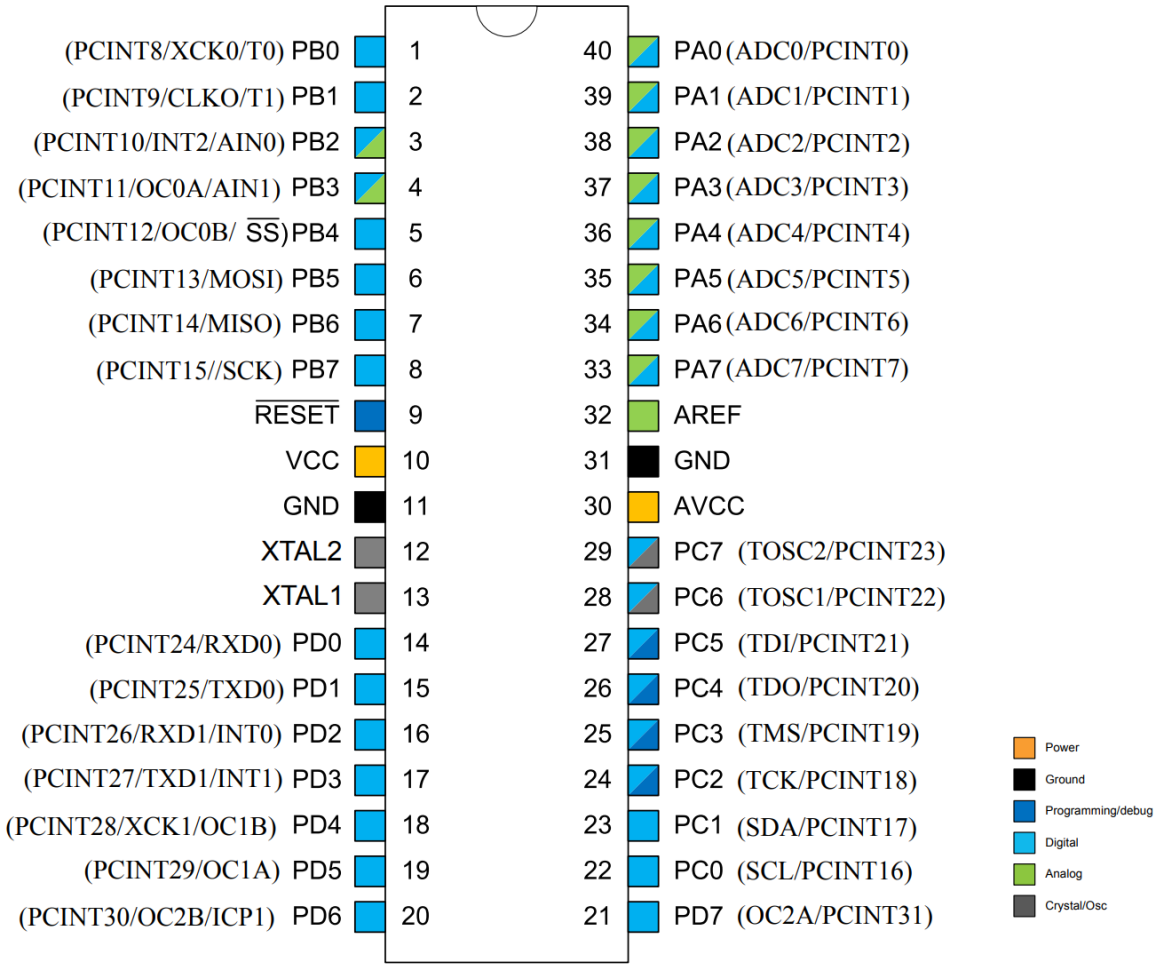
Blocurile sale interne, ce determina utilizarea sa in proiect, sunt oscilatorul intern, unitatea UART (Universal Asynchronous Receiver-Transmitter), interfata SPI, timere, convertoare analog-digital, comparatoare analogice. UART nu reprezinta un protocol de comunicatii, ci este un circuit integrat din cadrul microcontrollerului si al carui principal scop este sa transmita si sa primeasca informatii pe un port serial al unui computer sau dispozitiv periferic.

In cadrul procesoarelor AVR exista inclus pe cip un *Watchdog* care reseteaza procesorul dupa un anumit timp (nu mai mare de doua secunde), in cazul in care programul se blocheaza. Watchdog-ul contine un temporizator a carui expirare duce la generarea semnalului Reset pentru procesor. Daca watchdog-ul este activ, pentru a evita resetarea procesorului la fiecare 2 secunde, fara motiv, trebuie apelata instructiunea de resetare a watchdog-ului, definita in program ca *wdogtrig()*, la intervale oricat de mici, dar nu mai mari decat intervalul maxim la care a fost setat sa expire temporizatorul.

Pentru programarea microcontrollerului este necesar ca in flash-ul procesorului sa se incarce, in prima faza, un soft numit *bootloader*, dupa care acest soft comunica prin interfata seriala cu un program numit PC-loader pentru a transfera softul de aplicatie si a-l programa in flash. Aceasta metoda de programa controlerul vine insotita si de un dezavantaj reprezentat de faptul ca programul bootloader depinde in totalitate de configuratia de pe placa. Practic, folosirea altui microcontroller, a unui oscillator cu cuart de alta valoare decat cea specificata sau a plasarii butonului de la pinul D.5 presupune recompilarea bootloader-ului si reincarcarea sa folosind un programator classic.

4.2. Pinii microcontrollerului

Schema generala a pinilor microcontrollerului in care sunt afisate si porturile asociate este urmatoarea:



Fiecare port are doua registre asociate:

- Registrul DDRX (X = A, B, C, D): acesta specifica rolul fiecarui pin electric al portului X. Fiecare port are cate 8 pini, modelati prin 8 biti carora le atribuim valorile logice “1” si “0”, unde “1” inseamna iesire, iar “0” intrare. Trebuie avut in vedere ca un pin de port conectat la masa sau la Vcc sa nu fie definit ca “output”, deoarece s-ar obtine efectul unui scurtcircuit.
- Registrul PORTX: acesta are semnificatie diferita in functie de rolul fiecarui pin. Daca pinul ***n*** este setat ca iesire (DDRX.n = 1), atunci pinul PORTX.n va genera, in logica pozitiva, nivelul de tensiune corespunzator (“1” -> Vcc, “0” -> GND). Daca pinul ***n*** este setat ca intrare, aparent nu are sens scrierea unei valori in bitul respective. In acest ultim caz, scrierea unui “1” in PORTX.n va avea ca effect activarea unei rezistente interne de pull-up pentru acea intrare (aceasta rezistenta are valori mari).

Pentru realizarea conexiunii intre microcontroller si circuit am ales sa utilizam porturile B si C la care sa conectam led-urile prin care modelam semafoarele, iar declararea in cod a pinilor folositi este urmatoarea:

```
 //PENTRU PIETONII DE PE STRADA 1
#define LPS1_V PORTB.0
#define LPS1_R PORTB.1
 //PENTRU PIETONII DE PE STRADA 2
#define LPS2_V PORTB.2
#define LPS2_R PORTB.3
 //PENTRU  MASINILE DE PE STRADA 1
#define LMS1_V PORTB.4
#define LMS1_R PORTB.5
  //PENTRU MASINILE DE PE STRADA 2A
#define LMS2A_V PORTB.6
#define LMS2A_R PORTC.0
  //PENTRU MASINILE DE PE STRADA 2B
#define LMS2B_V PORTC.1
#define LMS2B_R PORTC.2
```

Dorim sa utilizam pinii din porturile B si C ca iesiri, iar setarea acestora se face atribuind valoarea “1” fiecarui bit:

```
DDRB = 0xFF;          //setarea pinilor din porturile B si C ca output
DDRC = 0xFF;

PORTB = PORTC = 0x00;//LOW = 0 HIGH = 1        //toate led-urile controlate de pini din B si C sunt stinse
```

Porturile B si C sunt porturi I/O bidirectionale pe 8 biti, cu rezistente interne de pull-up, selectabile individual pentru fiecare bit.

5. DESCRIEREA HARDWARE

5.1. LEDUL RGB

Pentru simularea semafoarelor am ales sa folosim led-uri RGB, pentru a evita supraaglomerarea de pe breadboard-urile utilizate. De exemplu, daca am fi folosit led-urile simple, cu 2 terminale ar fi trebuit sa le impartim astfel:

- Pentru fiecare dintre semafoarele masinilor ne-ar fi trebuit un numar de 3 led-uri, fiecare pentru culorile rosu, verde si galben. In total, fiind 3 semafoare de acest tip => 9 led-uri.
- Pentru fiecare dintre semafoarele pietonilor ne-ar fi trebuit un numar de 2 led-uri, fiecare pentru culorile rosu si verde. In total, fiind 6 semafoare de acest tip => 12 led-uri.

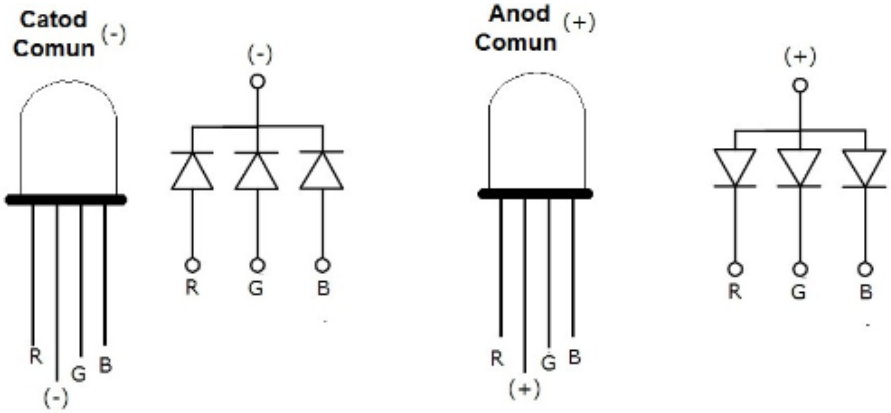
Astfel ne-ar fi trebuit un numar de 21 de leduri.
Utilizand led-urile RGB am reusit sa minimizam numarul de 21 de componente la 9.

Dar din ce sunt alcatuie aceste led-uri?

Led-urile RGB (Red, Green, Blue) sunt alcatuite din trei led-uri colorate diferit , intr-o singura capsula. Acestea se pot regasi in doua configuratii:

1. **Catod comun** : cele trei led-uri au terminalele pozitive separate si un singur catod
2. **Anod comun** : cele trei led-uri prezinta acelasi anod, dar fiecare catod este distinct

In realizarea proiectului, am ales sa utilizam ledurile RGB Catod comun.



In functie de tensiunea pe care o aplicam pe fiecare dintre anodul led-ului RGB, acesta va avea culoarea luminii emise diferita, un exemplu este prezentat in tabelul urmator, in cazul in care tensiunea pozitiva notata cu U(5V) este notata cu HIGH, iar LOW este echivalent lui 0V:

RED	GREEN	BLUE	COLOR
HIGH	LOW	LOW	
LOW	HIGH	LOW	
LOW	LOW	HIGH	
HIGH	HIGH	LOW	
HIGH	LOW	HIGH	
LOW	HIGH	HIGH	
HIGH	HIGH	HIGH	

In proiectul nostru, anodul caracteristic culorii albastru nu a fost folosit, deoarece am avut nevoie doar de culorile rosu, galben si verde.

5.2. UTILIZAREA REZISTENTELOR

Am utilizat un numar de 10 rezistente de valoare 510Ω cu scopul de a limita curentul prin fiecare led ce alcatuieste capsula RGB.

Din datele de catalog ale led-ului RGB, putem afla tensiunea directa suportata de fiecare led corespunzator celor 3 culori:

- Pentru rosu, tensiunea directa este intre 1.8 V si 2 V.

Daca presupunem ca la borna rezistorului (legat in serie cu led-ul) aplicam un potential in jurul valorii de 5V, am obtine un curent de (5V-2V)/(510Ω) = 5.88 mA.

- Pentru verde si albastru , tensiunea directa este intre 3.2 V si 3.4 V.

Daca presupunem ca la borna rezistorului (legat in serie cu led-ul) aplicam un potential in jurul valorii de 5V, am obtine un curent de (5V-3.4V)/(510Ω) = 3.14 mA.

Rosu: -lungime de unda: 630 - 640 nm
-intensitatea luminii: 1000-1200 mcd
-tensiune directa: 1.8 - 2.0 V
Verde: -lungime de unda: 515 - 525 nm
-intensitatea luminii: 3000-5000 mcd
-tensiune directa: 3.2 - 3.4 V
Albastru: -lungime de unda: 465 - 475 nm
-intensitatea luminii: 2000-3000 mcd
-tensiune directa: 3.2 - 3.4 V

6. DESCRIEREA SOFTWARE

6.1. DECLARAREA PARAMETRILOR FOLOSITI

In fisierul defs.h vom declara pinii microcontrollerului pe care ii vom utiliza. Am ales sa folosim porturile caracteristice registrilor B si C ca si functii de output.

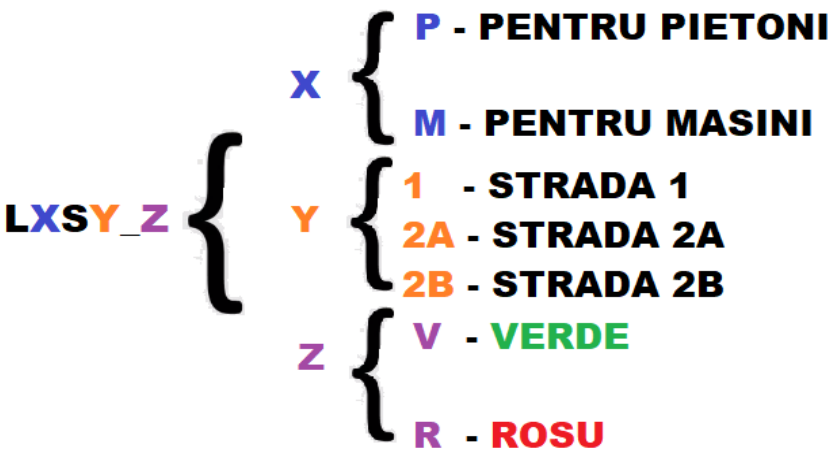
Fiecarui semafor pentru masini ii vor fi atribuiti 2 pini ai microcontrollerului:

- Daca aplicam tensiune pe anodul caracteristic culorii **ROSU** obtinem **ROSU**
- Daca aplicam tensiune pe anodul caracteristic culorii **VERDE** obtinem **VERDE**
- Daca aplicam tensiune pe anodul caracteristic culorii **ROSU** dar si pe anodul caracteristic culorii **VERDE** vom obtine **GALBEN**

Fiecarei perechi de semafoare pentru pietoni ii vor fi atribuiti 2 pini ai microcontrollerului:

- Daca aplicam tensiune pe anodul caracteristic culorii **ROSU** obtinem **ROSU**
- Daca aplicam tensiune pe anodul caracteristic culorii **VERDE** obtinem **VERDE**

In cod vom folosi urmatoarea notatie:




```
 //PENTRU PIETONII DE PE STRADA 1
#define LPS1_V PORTB.0
#define LPS1_R PORTB.1
 //PENTRU PIETONII DE PE STRADA 2
#define LPS2_V PORTB.2
#define LPS2_R PORTB.3
 //PENTRU  MASINILE DE PE STRADA 1
#define LMS1_V PORTB.4
#define LMS1_R PORTB.5
  //PENTRU MASINILE DE PE STRADA 2A
#define LMS2A_V PORTB.6
#define LMS2A_R PORTC.0
  //PENTRU MASINILE DE PE STRADA 2B
#define LMS2B_V PORTC.1
#define LMS2B_R PORTC.2
```

Totodata, am stabilit o notatie pentru “1” logic (5V), ce semnifica aprinderea led-ului dar si “0” logic (0V) ce semnifica stingerea led-ului:

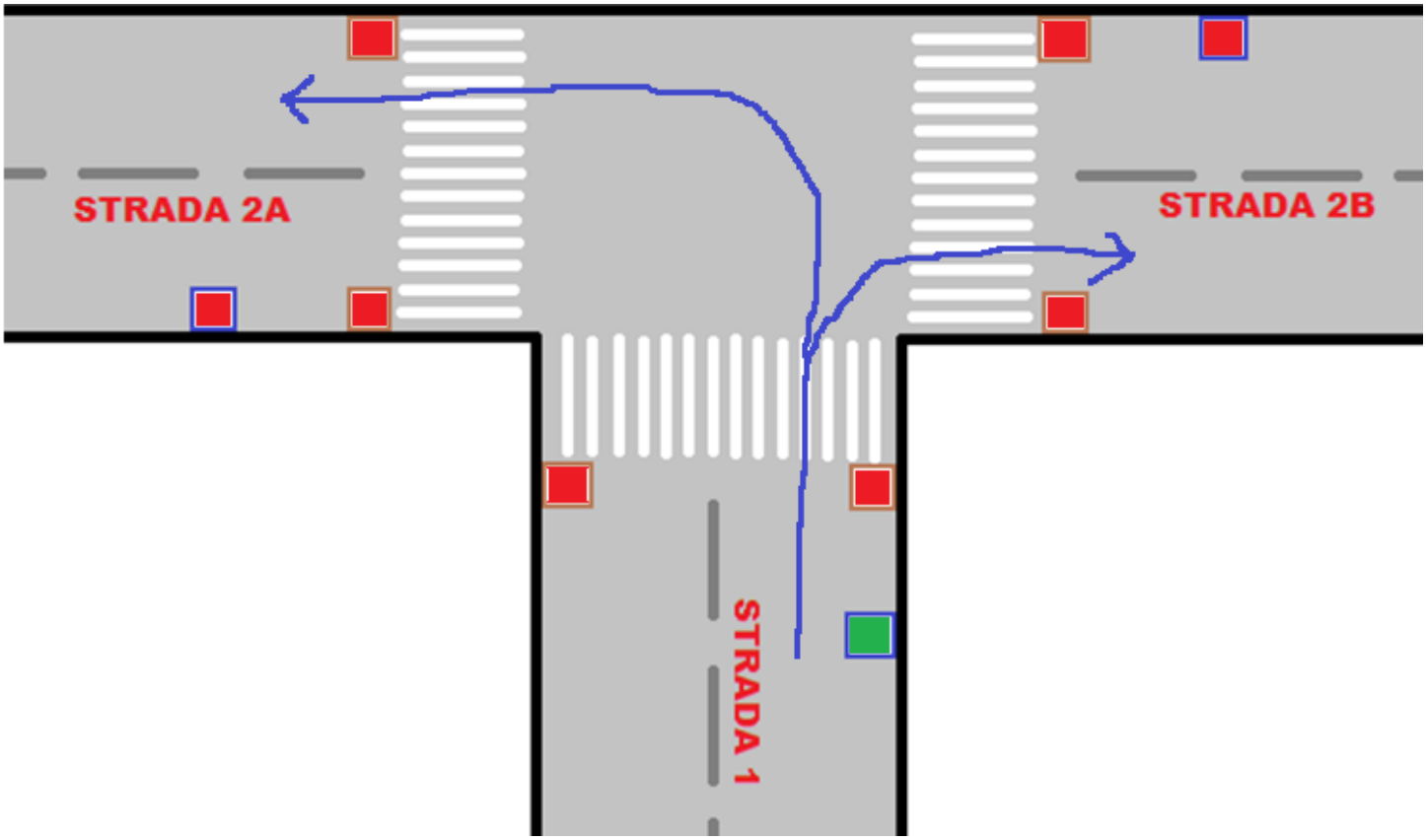
```
#define HIGH 1
#define LOW 0
```

6.2. DESCRIEREA ETAPELOR

Am impartit durata unui ciclu al intersectiei in 8 etape. Am create functii separate pentru fiecare dintre aceste etape, functii pe care le-am apelat in functia main a programului. HIGH corespunde aprinderii unui led, LOW corespunde stingerii unui led.

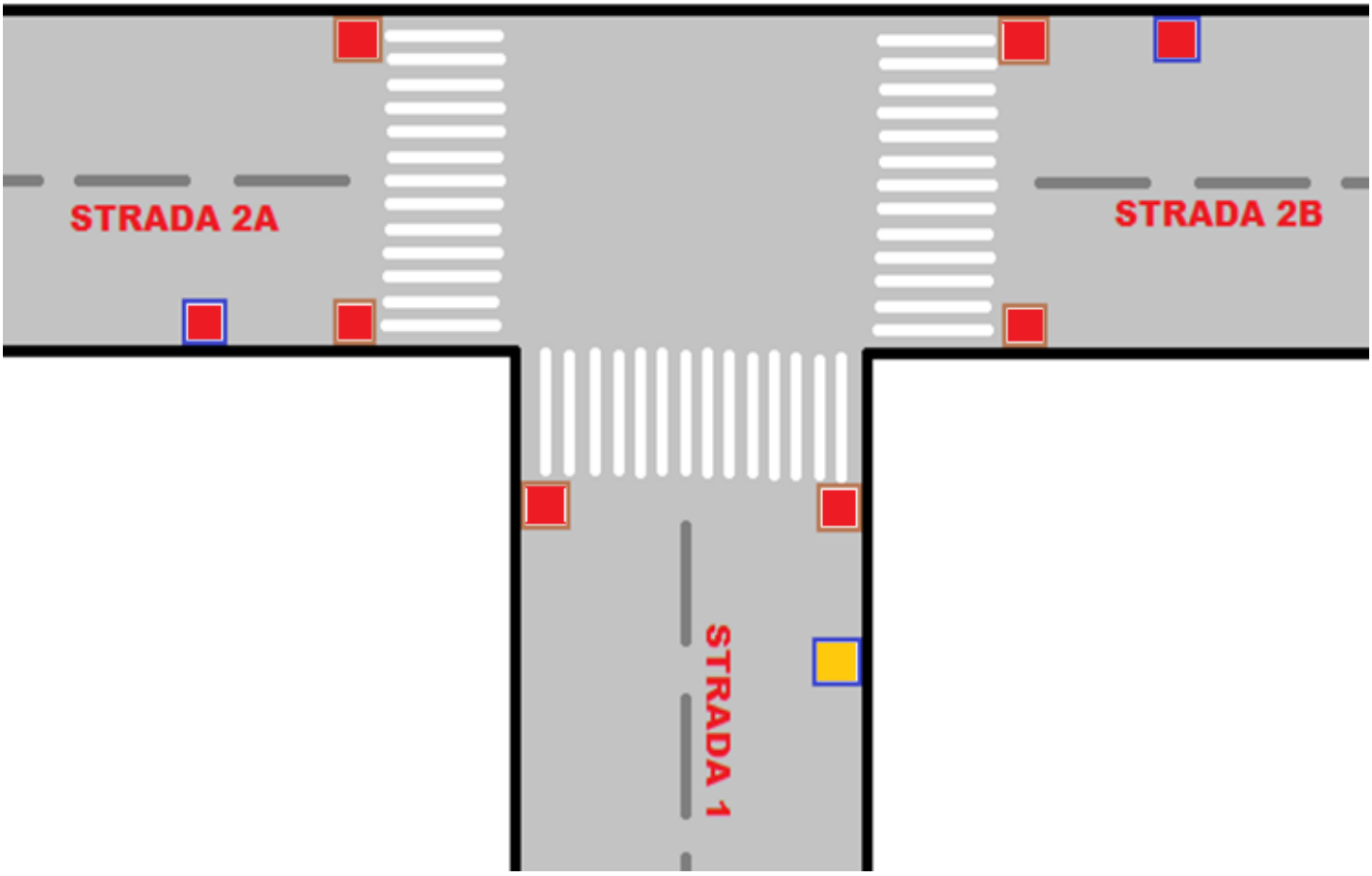
ETAPA 1:

```
void etapa1()
{                       //SEMAFORUL PE STRADA 2A PENTRU MASINI ARE CULOAREA ROSIE
      LMS2A_V = LOW;
      LMS2A_R = HIGH;

      delay_ms(1500);   //LA TERMINAREA CICLULUI MAI INTAI SE FACE ROSU PE STRADA 2A IAR CULORILE CELORLALTE SEMAFOARE COMUTA DUPA O INTARZIERE DE 1.5 SECUNDE
                        //SEMAFOARELE PE STRADA 1 PENTRU PIETONI AU CULOAREA ROSIE
      LPS1_V = LOW;
      LPS1_R = HIGH;
                        //SEMAFOARELE PE STRADA 2 PENTRU PIETONI AU CULOAREA ROSIE
      LPS2_V = LOW;
      LPS2_R = HIGH;
                        //SEMAFORUL PE STRADA 2B PENTRU MASINI ARE CULOAREA ROSIE
      LMS2B_V = LOW;
      LMS2B_R = HIGH;
                        //SEMAFORUL PE STRADA 1 PENTRU MASINI ARE CULOAREA ROSIE
      LMS1_V = HIGH;
      LMS1_R = LOW;
}
```



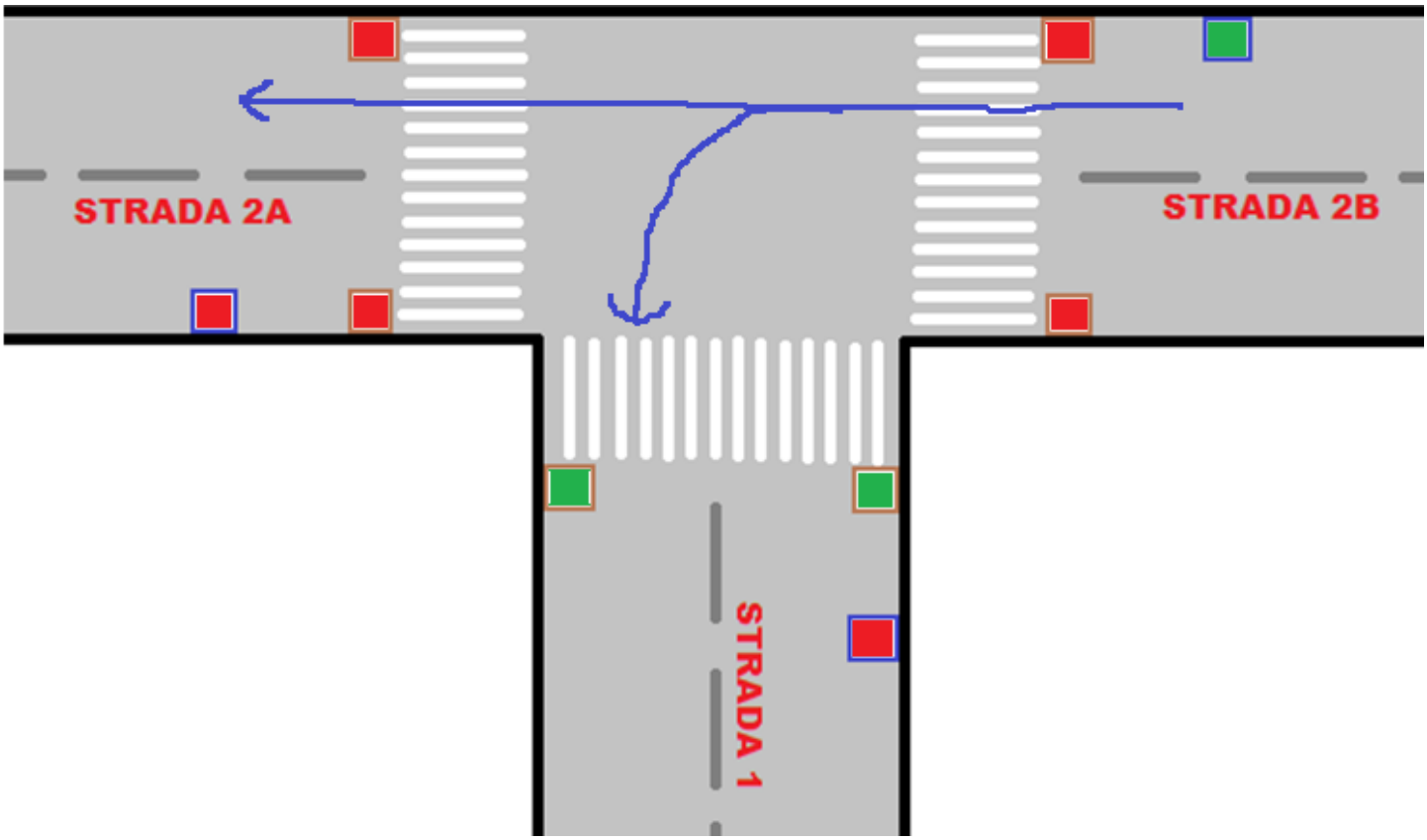
ETAPA 2:

```
void etapa2()
{                       //SEMAFORUL PE STRADA 2A PENTRU MASINI ARE CULOAREA ROSIE
      LMS2A_V = LOW;
      LMS2A_R = HIGH;
                        //SEMAFOARELE PE STRADA 1 PENTRU PIETONI AU CULOAREA ROSIE
      LPS1_V = LOW;
      LPS1_R = HIGH;
                        //SEMAFOARELE PE STRADA 2 PENTRU PIETONI AU CULOAREA ROSIE
      LPS2_V = LOW;
      LPS2_R = HIGH;
                         //SEMAFORUL PE STRADA 2B PENTRU MASINI ARE CULOAREA ROSIE
      LMS2B_V = LOW;
      LMS2B_R = HIGH;
                        //SEMAFORUL PE STRADA 1 PENTRU MASINI ARE CULOAREA GABLBEN
      LMS1_V = HIGH;
      LMS1_R = HIGH;
}
```



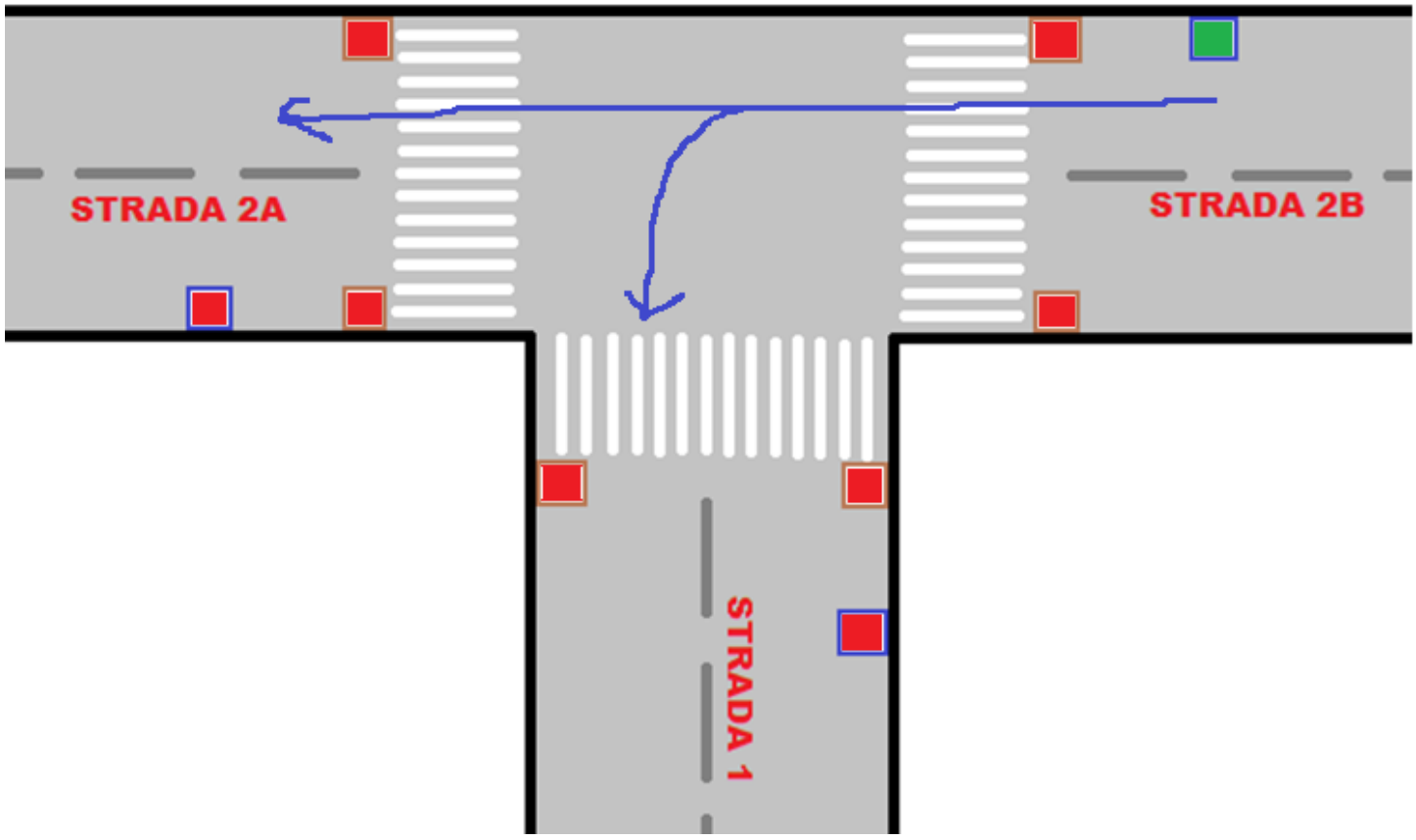
ETAPA 3:

```
void etapa3()
{                          //SEMAFORUL PE STRADA 1 PENTRU MASINI ARE CULOAREA ROSIE
      LMS1_V = LOW;
      LMS1_R = HIGH;

      delay_ms(1500);  //DUPA CE SE FACE ROSU PE STRADA 1, CULORILE CELORLALTE SEMAFOARE COMUTA DUPA 1.5 SECUNDE
                       //SEMAFORUL PE STRADA 2A PENTRU MASINI ARE CULOAREA ROSIE
      LMS2A_V = LOW;
      LMS2A_R = HIGH;
                        //SEMAFOARELE PE STRADA 1 PENTRU PIETONI AU CULOAREA VERDE
      LPS1_V = HIGH;
      LPS1_R = LOW;
                       //SEMAFOARELE PE STRADA 2 PENTRU PIETONI AU CULOAREA ROSIE
      LPS2_V = LOW;
      LPS2_R = HIGH;
                       //SEMAFORUL PE STRADA 2B PENTRU MASINI ARE CULOAREA VERDE
      LMS2B_V = HIGH;
      LMS2B_R = LOW;

}
```



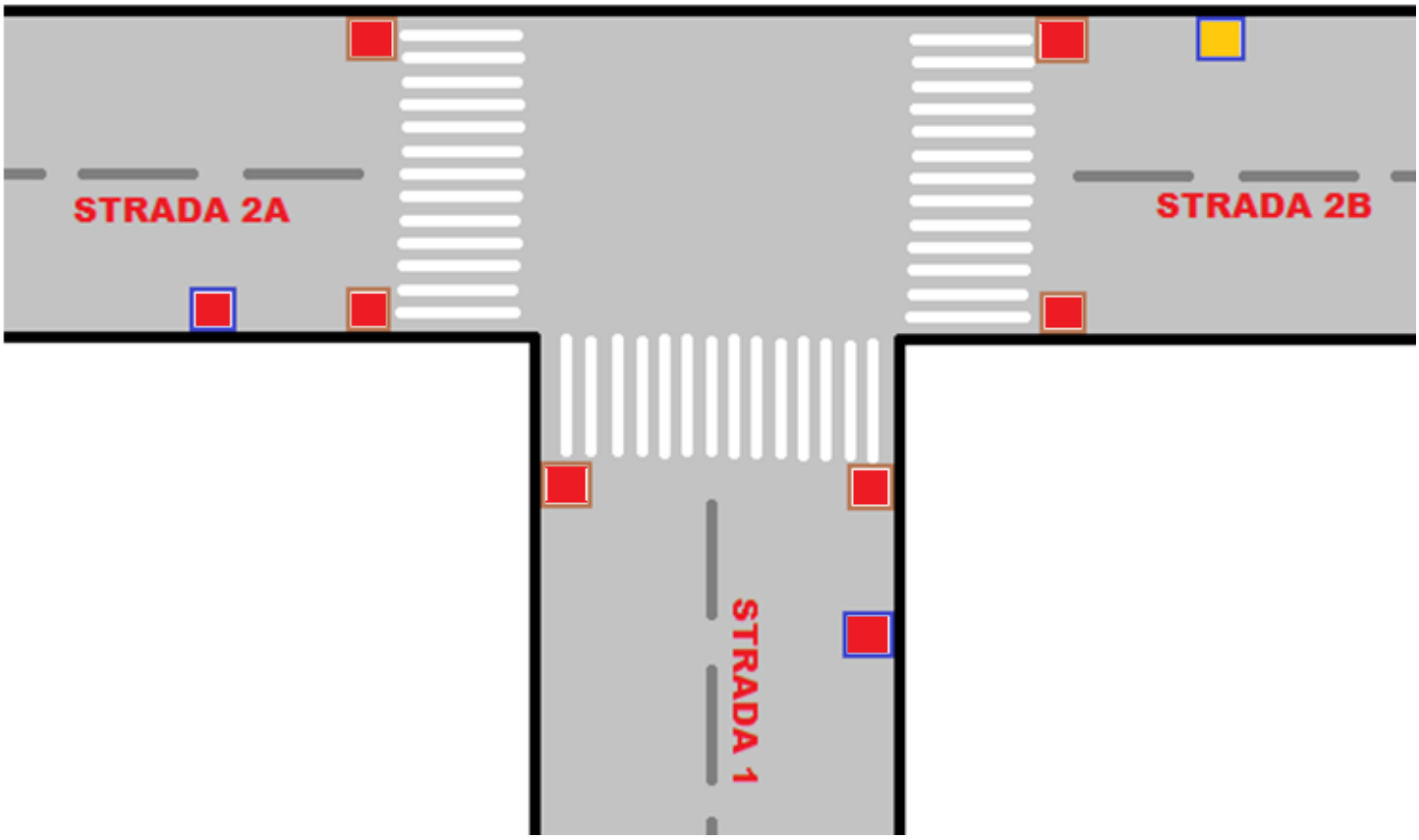
ETAPA 4:

```
void etapa4()
{                       //SEMAFOARELE PE STRADA 1 PENTRU PIETONI AU CULOAREA ROSU
      LPS1_V = LOW;
      LPS1_R = HIGH;
                        //SEMAFOARELE PE STRADA 2 PENTRU PIETONI AU CULOAREA ROSU
      LPS2_V = LOW;
      LPS2_R = HIGH;
                        //SEMAFORUL PE STRADA 1 PENTRU MASINI ARE CULOAREA ROSIE
      LMS1_V = LOW;
      LMS1_R = HIGH;
                        //SEMAFORUL PE STRADA 2A PENTRU MASINI ARE CULOAREA ROSIE
      LMS2A_V = LOW;
      LMS2A_R = HIGH;

                         //SEMAFORUL PE STRADA 2B PENTRU MASINI ARE CULOAREA VERDE
      LMS2B_V = HIGH;
      LMS2B_R = LOW;

      delay_ms(4000);
}
```



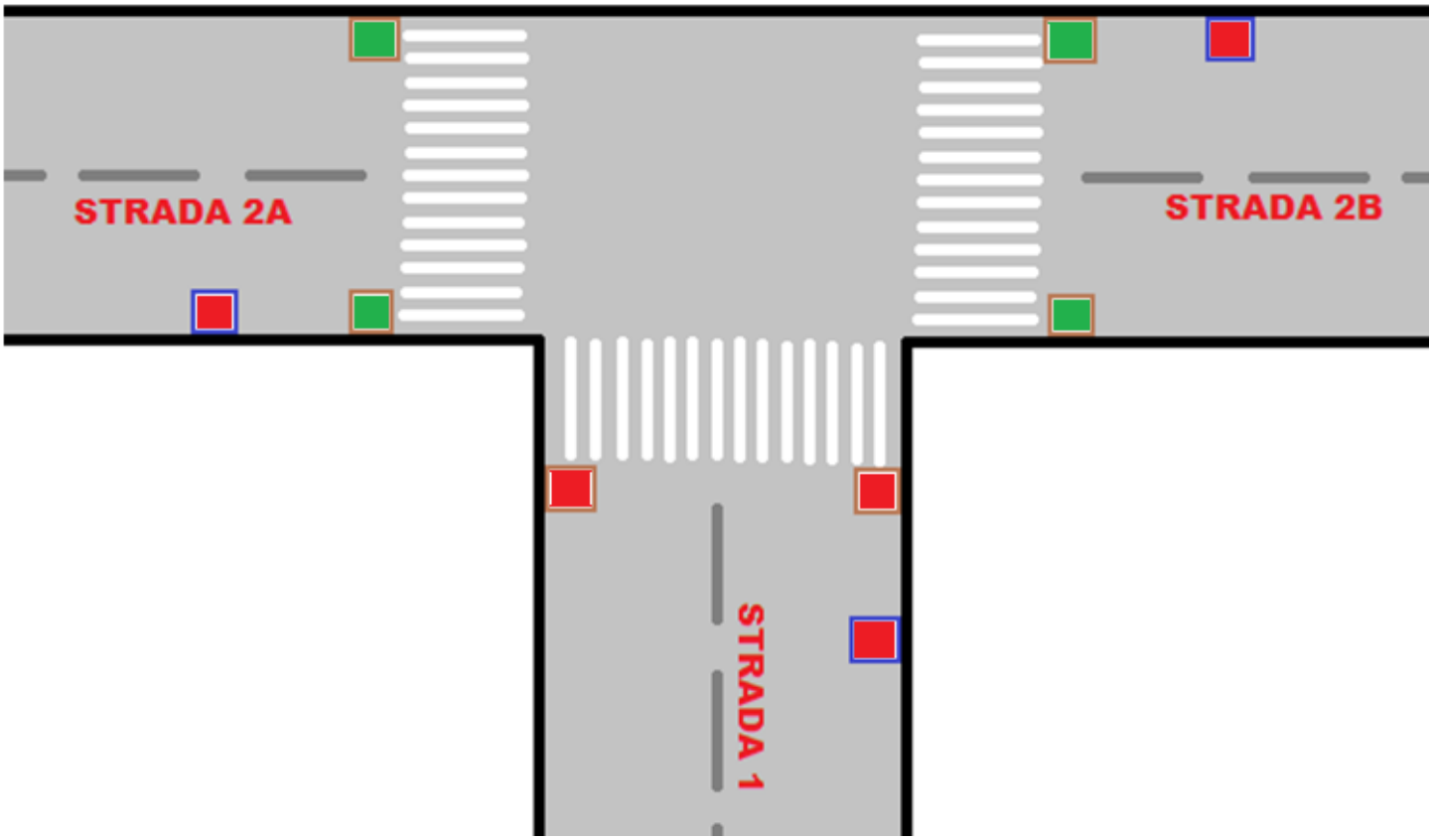
ETAPA 5:

```
void etapa5()
{                       //SEMAFOARELE PE STRADA 1 PENTRU PIETONI AU CULOAREA ROSU
      LPS1_V = LOW;
      LPS1_R = HIGH;
                        //SEMAFOARELE PE STRADA 2 PENTRU PIETONI AU CULOAREA ROSU
      LPS2_V = LOW;
      LPS2_R = HIGH;
                        //SEMAFORUL PE STRADA 1 PENTRU MASINI ARE CULOAREA ROSIE
      LMS1_V = LOW;
      LMS1_R = HIGH;
                        //SEMAFORUL PE STRADA 2A PENTRU MASINI ARE CULOAREA ROSIE
      LMS2A_V = LOW;
      LMS2A_R = HIGH;

                         //SEMAFORUL PE STRADA 2B PENTRU MASINI ARE CULOAREA GALBEN
      LMS2B_V = HIGH;
      LMS2B_R = HIGH;
}
```



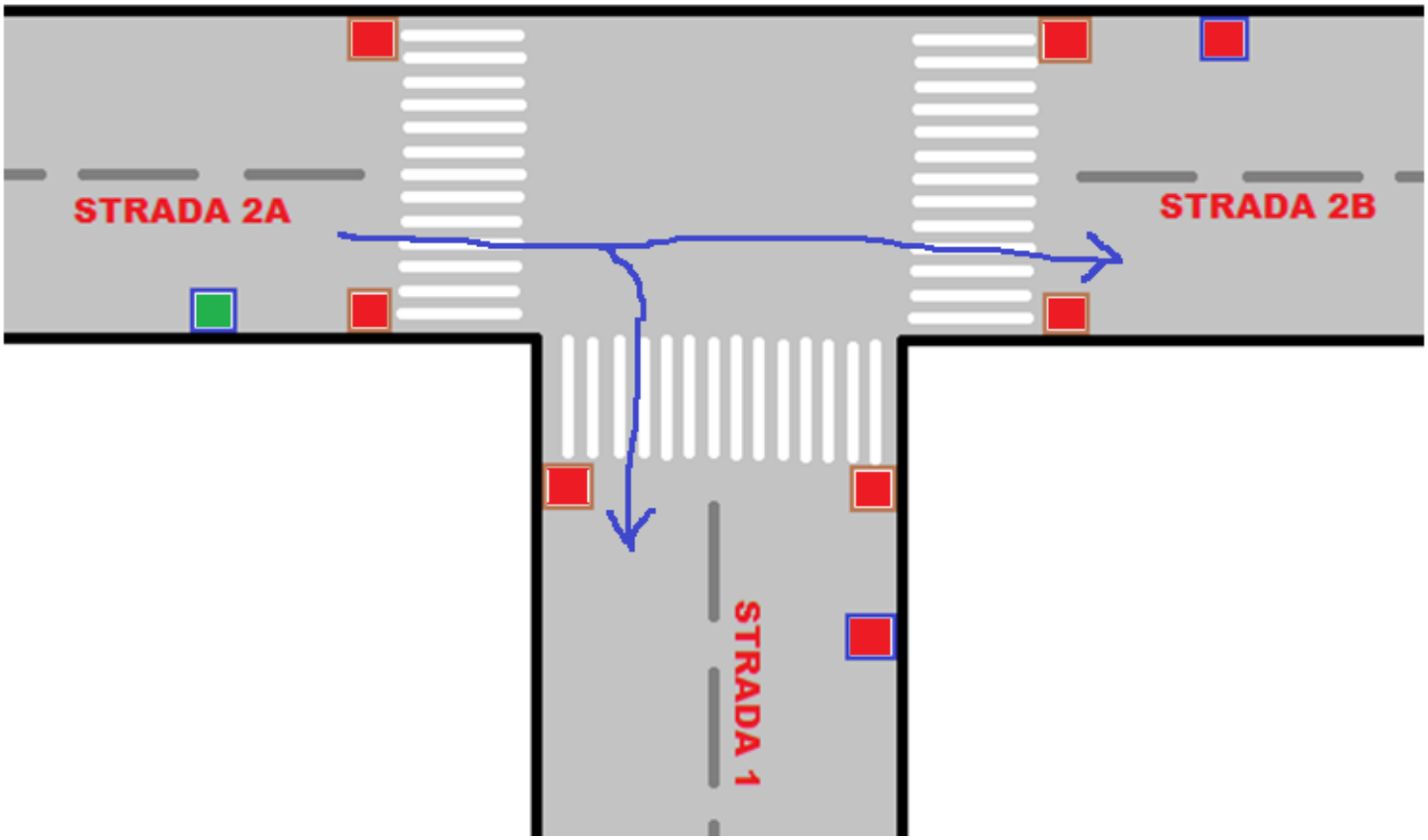
ETAPA 6:

```
void etapa6()
{                          //SEMAFORUL PE STRADA 2B PENTRU MASINI ARE CULOAREA ROSIE
      LMS2B_V = LOW;
      LMS2B_R = HIGH;

      delay_ms(1500);    //DUPA CE SE FACE ROSU PE STRADA 2B, CULORILE CELORLALTE SEMAFOARE COMUTA DUPA 1.5 SECUNDE
                         //SEMAFOARELE PE STRADA 1 PENTRU PIETONI AU CULOAREA ROSU
      LPS1_V = LOW;
      LPS1_R = HIGH;
                         //SEMAFOARELE PE STRADA 2 PENTRU PIETONI AU CULOAREA VERDE
      LPS2_V = HIGH;
      LPS2_R = LOW;
                          //SEMAFORUL PE STRADA 1 PENTRU MASINI ARE CULOAREA ROSIE
      LMS1_V = LOW;
      LMS1_R = HIGH;
                         //SEMAFORUL PE STRADA 2A PENTRU MASINI ARE CULOAREA ROSIE
      LMS2A_V = LOW;
      LMS2A_R = HIGH;
}
```



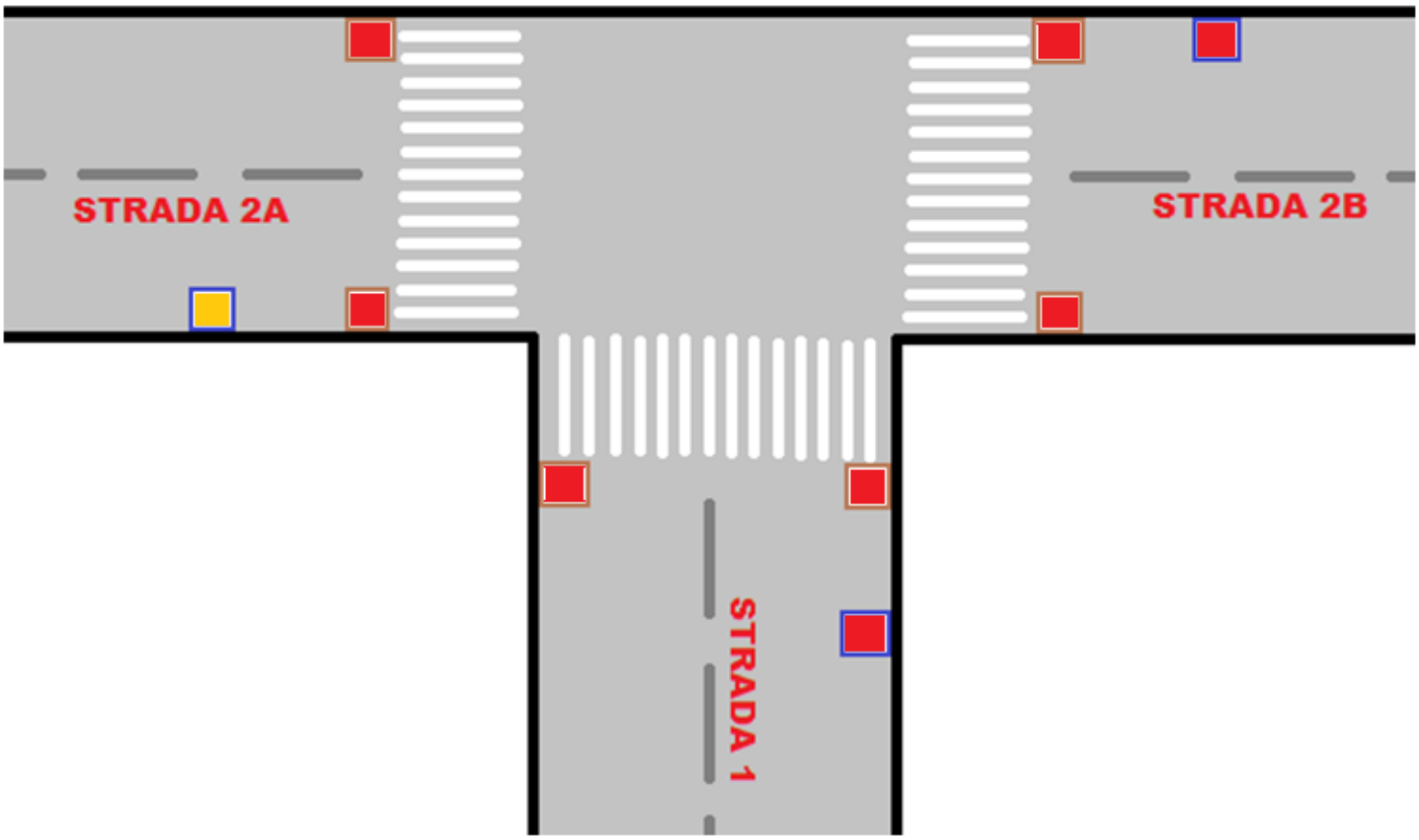
ETAPA 7:

```
void etapa7()
{                       //SEMAFOARELE PE STRADA 2 PENTRU PIETONI AU CULOAREA ROSU
      LPS2_V = LOW;
      LPS2_R = HIGH;

      delay_ms(1500);  //DUPA CE SE FACE ROSU LA PIETONII DE PE STRADA 2, CULORILE CELORLALTE SEMAFOARE COMUTA DUPA 2 SECUNDE
                       //SEMAFOARELE PE STRADA 1 PENTRU PIETONI AU CULOAREA ROSU
      LPS1_V = LOW;
      LPS1_R = HIGH;
                        //SEMAFORUL PE STRADA 1 PENTRU MASINI ARE CULOAREA ROSIE
      LMS1_V = LOW;
      LMS1_R = HIGH;
                        //SEMAFORUL PE STRADA 2A PENTRU MASINI ARE CULOAREA VERDE
      LMS2A_V = HIGH;
      LMS2A_R = LOW;
                         //SEMAFORUL PE STRADA 2B PENTRU MASINI ARE CULOAREA ROSIE
      LMS2B_V = LOW;
      LMS2B_R = HIGH;
}
```



ETAPA 8:

```
void etapa8()
{                       //SEMAFOARELE PE STRADA 2 PENTRU PIETONI AU CULOAREA ROSU
      LPS2_V = LOW;
      LPS2_R = HIGH;
                        //SEMAFOARELE PE STRADA 1 PENTRU PIETONI AU CULOAREA ROSU
      LPS1_V = LOW;
      LPS1_R = HIGH;
                        //SEMAFORUL PE STRADA 1 PENTRU MASINI ARE CULOAREA ROSIE
      LMS1_V = LOW;
      LMS1_R = HIGH;
                        //SEMAFORUL PE STRADA 2B PENTRU MASINI ARE CULOAREA ROSIE
      LMS2B_V = LOW;
      LMS2B_R = HIGH;
                         //SEMAFORUL PE STRADA 2A PENTRU MASINI ARE CULOAREA GALBEN
      LMS2A_V = HIGH;
      LMS2A_R = HIGH;
}
```



6.3. ROLUL FUNCTIILOR FLASH_PSX()

Pentru ca simularea intersectiei sa se apropie cat mai mult de realitate, am construit 2 functii: flash_PS1() si flash_PS2(). Acestea au rolul de a semnala terminarea timpului culorii verzi a semafoarelor pentru pietoni. Am ales o implementare usoara a acestor functii, ce au urmatoarele structuri:

```
void flash_PS1()       //este functia luminare intermitenta pentru semoaferele pietonilor de pe strada 1 si se activeaza atunci
                        //cand timpul pentru traversare este aproape de final
{
    int initialState = LPS1_V;   //retinem starea culorii verzi a semafoarelor pentru pietonii de pe strada 1
    int i = 0;
    for(i = 0 ; i < 4; i++)     //prin parcurgerea buclei for asiguram aprinderea si stingerea led-ului de 4 ori
    {
       LPS1_V = LOW;
       delay_ms(500);
       LPS1_V = HIGH;
       delay_ms(500);
    }
    LPS1_V = initialState;
}
```

```
void flash_PS2()         //este functia luminare intermitenta pentru semoaferele pietonilor de pe strada 2 si se activeaza atunci
                         //cand timpul pentru traversare este aproape de final
{
    int initialState = LPS2_V;   //retinem starea culorii verzi a semafoarelor pentru pietonii de pe strada 2
    int i;
    for(i = 0 ; i < 4; i++)     //prin parcurgerea buclei for asiguram aprinderea si stingerea led-ului de 4 ori
    {
       LPS2_V = LOW;
       delay_ms(500);
       LPS2_V = HIGH;
       delay_ms(500);
    }
    LPS2_V = initialState;
```

6.4. DESCRIEREA DURATEI UNUI CICLU

Cerinta temei impune stabilirea unei durate fixe a ciclului. Am considerat ca un ciclu al intersectiei are durata de 2 minute si 30 de secunde. Totodata, o cerinta foarte importanta este ca durata fiecarei faze sa fie reglabila prin hyperterminal (intre 10% si 80% dintr-un ciclu, cu pasul de reglare 10%).

Pentru inceput, trebuie sa calculam secundele pierdute in interiorul functiilor fiecarei etape, dar si in programul principal ce simuleaza intregul ciclu. Printr-o adunare simpla, aflam ca se pierd 27 de secunde.

Asadar ne raman 150 secunde (2 minute si 30 de secunde) – 27 secunde = 123 secunde. Acestea trebuie impartite intre timpul specific culorii verzi pentru: semafoarele pietonilor dupa strada 2, semafoarele masinilor dupa strada 1, semafoarele masinilor dupa strada 2A, semafoarele masinilor dupa strada 2B (implicit semafoarele pietonilor dupa strada 1). Modificarea delay-ului pentru unul dintre aceste semafoare implica automat schimbarea timpului si pentru celelalte.

La pasul 2, pentru a regla timpul pentru fiecare semafor, avem nevoie de o functie ce intoarce procentul din ciclul total pe care il dorim:

```
int get_delay (char arg) {     //functie prin care atribuim caracterelor introduse valori prestabilite ale delay-urilor pe care dorim sa le setam
  switch(arg) {                //switch-ul functioneaza doar cu numere, iar noi chiar daca ii dam un caracter, el o sa foloseasca coduri ASCII
   case '1': return 15;       //daca se introduce in terminal '1', functia returneaza o durata de 15 secunde (10% din ciclul de 2 minute si 30 sec = 150 sec)
   case '2': return 30;       //daca se introduce in terminal '2', functia returneaza o durata de 30 secunde (20% din ciclul de 2 minute si 30 sec)
   case '3': return 45;       //daca se introduce in terminal '3', functia returneaza o durata de 45 secunde (30% din ciclul de 2 minute si 30 sec)
   case '4': return 60;       //daca se introduce in terminal '4', functia returneaza o durata de 60 secunde (40% din ciclul de 2 minute si 30 sec)
   case '5': return 75;       //daca se introduce in terminal '5', functia returneaza o durata de 75 secunde (50% din ciclul de 2 minute si 30 sec)
   case '6': return 90;       //daca se introduce in terminal '6', functia returneaza o durata de 60 secunde (60% din ciclul de 2 minute si 30 sec)
   case '7': return 105;      //daca se introduce in terminal '7', functia returneaza o durata de 60 secunde (70% din ciclul de 2 minute si 30 sec)
   case '8': return 120;      //daca se introduce in terminal '8', functia returneaza o durata de 60 secunde (80% din ciclul de 2 minute si 30 sec)

   default : return 5;

  }
```

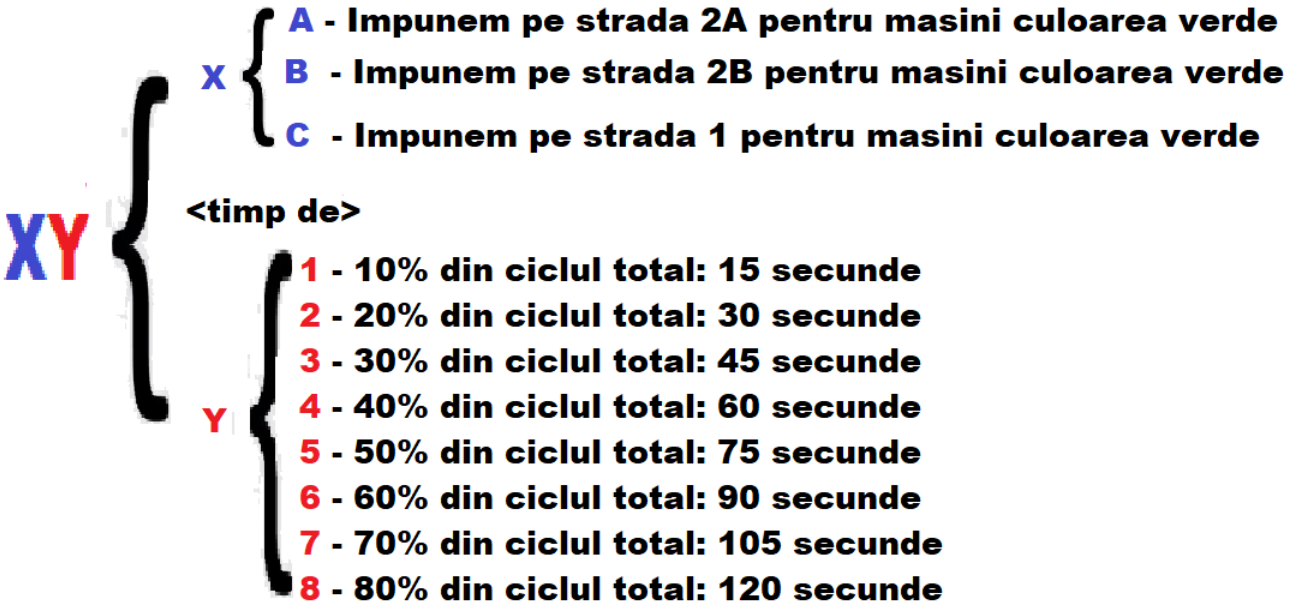
Pentru usurinta, am ales ca si argument al functiei get_delay(), o variabila de tip char, variabila care va face parte dintr-un vector, o secventa de litere introdusa ca si input in terminal. Aceasta functie va fi folosita pentru setarea delay-urilor pentru fiecare semafor de masini, respectiv de pietoni.

La pasul 3, declaram un vector in care vom retine caracterele introduce necesare setarii delay-ului, dar si un index prin care vom parcurge acest vector.

```
char SS[4] = "";      //vector in care retinem caracterele introduse necesare setarii delay-ului
int s=0;              //index cu care parcurgem vectorul
```

Varibila temp va fi cea care retine caracterul scris in terminal, cu ajutorul functiei getchar().

Vom impune o conditie, si anume ca ciclul va incepe o data ce vom introduce in terminal o comanda de tipul:



Retinem in vectorul SS caracterele introduce (ce alcatuiesc comanda), primul caracter (litera A, B sau C) este retinut in SS[0], urmatorul caracter (1-8) este retinut in SS[1].

Un alt aspect, este declararea variabilei flag, ce are rol in a semnala ca ciclul se poate efectua, acesta primind valoarea 1 in fiecare functie din descrierea delay-urilor.

La pasul 4, atribuim cate o variabila pentru fiecare delay pe care dorim sa il setam, asadar:

- ➔ Pentru culoarea verde a semaforului de pe strada 1 pentru masini : t_S1
- ➔ Pentru culoarea verde a semaforului de pe strada 2A pentru masini: t_S2A
- ➔ Pentru culoarea verde a semaforului de pe strada 2B pentru masini (implicit pentru culoarea verde a semafoarelor de pe strada 1 pentru pietoni) : t_S2B
- ➔ Pentru culoarea verde a semafoarelor de pe strada 2 pentru pietoni: t_P

Am decis ca timpul de asteptare/trecere a pietonilor de pe strada 2 sa fie stabilit de functiile ce seteaza fiecare timp pentru culorile verzi ale semafoarelor pentru masini.

Asadar de exemplu, daca dorim sa setam un timp in care masinile de pe strada 2A pot traversa intersectia, vom atribui variabilei t_S2A delay-ul in secunde citit din terminal, folosind astfel si functia creata get_delay().

Dar, totodata, trebuie sa decidem cum impartim restul timpului ramas pentru celelalte semafoare. Varianta pe care am ales-o consta in impartirea in mod egal a acestui timp. In variabila dif este memorata diferenta dintre 123 secunde si delay-ul setat, t_S2A.

Procedam la fel si pentru restul functiilor:

```
temp = getchar();
if( temp ==  'A' ||  temp ==  'B'  || temp ==  'C'  || (temp >= '1' && temp <= '8'))
{

SS[s++] = temp;   //retinem in SS caracterele introduse, primul caracter e retinut in SS[0] iar s=0, urmatorul caracter este retinut
                  // in SS[1], iar s=1
if(s >= 2) {          //asigurare ca va rula codul chiar si atunci cand sunt introduse caractere in plus
if (SS[0] == 'A' )    //compara codul ASCII
{   flag = 1;
    t_S2A = get_delay(SS[1]);    //retinem in t_S2A valoarea intoarsa de functia get_delay
    dif = 123 - t_S2A;          //ciclul are 150 secunde, dar 27 sec sunt pierdute in cadrul fazelor
    t_S1 = dif/3;               //alegem ca timpul ramas sa fie distribuit in mod egal
    t_P = dif/3;
    t_S2B = dif/3;

}


if (SS[0] == 'B' )
{  flag = 1;
   t_S2B = get_delay(SS[1]);      //retinem in t_S2B valoarea intoarsa de functia get_delay
   dif = 123 - t_S2B;             //ciclul are 150 secunde, dar 27 sec sunt pierdute in cadrul fazelor
   t_S1 = dif/3;                  //alegem ca timpul ramas sa fie distribuit in mod egal
   t_P =  dif/3;
   t_S2A = dif/3;

}


if (SS[0] == 'C' )
{   flag = 1;
    t_S1 = get_delay(SS[1]);       //retinem in t_S1 valoarea intoarsa de functia get_delay
    dif = 123 - t_S1;              //ciclul are 150 secunde, dar 27 sec sunt pierdute in cadrul fazelor
    t_S2B = dif/3;                 //alegem ca timpul ramas sa fie distribuit in mod egal
    t_P =  dif/3;
    t_S2A = dif/3;


 }
```

La pasul 5, verificam daca a fost scris un caracter de identificare a strazilor si introducem ciclul intr-o bucla infinita necesara asigurarii continuitatii ciclului. Schimbarea semafoarelor din culoarea galben in rosu se realizeaza intr-un timp de 3 secunde.

```
if(flag) {           // verificam daca a fost scris un caracter de identificare a strazilor

   while(1) {        // am introdus ciclul intr-o bucla infinita pentru a asigura continuitatea acestuia


   etapa1();
   for(i=0; i< t_S1; i++){
   delay_ms (1000);   //cat timp au verde masinile de pe strada 1 identificata prin 'C'
   }
   etapa2();

   delay_ms(3000);         //cat dureaza culoarea galben a semaforului de pe strada 1 ('C')

   etapa3();
   for(i=0; i< t_S2B; i++){

   delay_ms(1000);  //cat timp au verde masinile de pe strada 2B (in acelasi timp pietonii de pe strada 1 au verde)
   }
   flash_PS1();         //semaforul pietonilor de pe strada 1 lumineaza intermitent cand acestia mai au putin timp pentru a traversa

   etapa4();

   etapa5();

   delay_ms(3000);     //cat dureaza culoarea galben a semaforului pe strada 2B

   etapa6();
   for(i=0; i< t_P; i++){

   delay_ms(1000);   // semafoarele pietonilor de pe strazile 2A si 2B au culoarea verde
   }
   flash_PS2();          //semafoarele pietonilor de pe strazile 2A si 2B lumineaza intermitent cand acestia mai au putin timp pentru a traversa

   etapa7();
   for(i=0; i< t_S2A; i++){

   delay_ms(1000);   // cat timp au verde masinile de pe strada 2A
   }
   etapa8();

   delay_ms(3000);         //cat dureaza culoarea galben a semaforului de pe strada 2A
   }
}
```

In cazul in care conditia de la pasul 3 nu este respectata si nu avem identificatorul prestabilit al strazii scris in terminal, atunci am ales un ciclu default:

```
else {          //in cazul in care sunt introduse alte caractere decat cele prin care sunt identificate strazile, am stabilit un ciclu implicit
   while(1) {
   etapa1();

   delay_ms(35000);

   etapa2();

   delay_ms(3000);

   etapa3();

   delay_ms(35000);

   flash_PS1();

   etapa4();

   etapa5();

   delay_ms(3000);

   etapa6();

   delay_ms(18000);

   flash_PS2();

   etapa7();

   delay_ms(35000);

   etapa8();

   delay_ms(3000);
   }
```

Pentru a schimba in orice moment delay-ul unei strazi, vom folosi functia de reset a microcontrollerului, atribuindu-i o noua functionalitate, si anume restabilirea duratei fazelor.

7. BIBLIOGRAFIE

- https://www.optimusdigital.ro/ro/optoelectronice-led-uri/483-led-rgb-catod-comun.html?search_query=RGB&results=118
- http://ham.elcom.pub.ro/proiect2/doc.htm
- http://ww1.microchip.com/downloads/en/devicedoc/atmel-8272-8-bit-avr-microcontroller-atmega164a_pa-324a_pa-644a_pa-1284_p_datasheet.pdf