Universitatea Națională de Știință și Tehnologie POLITEHNICA București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sistem Automat de Irigare Inteligentă bazat pe Microcontroler pentru Monitorizarea și Controlul Condițiilor din Seră

Tatu Rareș Mihai

424E

Cuprins

Lista figurilor:

Lista tabelelor:

Introducere

Obiectivul principal al acestui proiect este proiectarea și implementarea unui **sistem automat de irigare inteligentă** pentru o seră, capabil să ajusteze cantitatea de apă furnizată plantelor în funcție de nivelul de umiditate al solului și de condițiile de mediu (temperatură și lumină). Acest sistem are ca scop creșterea eficienței utilizării resurselor de apă, automatizarea procesului de îngrijire a plantelor și reducerea intervenției umane.

Metodologia utilizată include aplicarea cunoștințelor despre circuite logice combinaționale (CLC) și secvențiale (CLS), programarea microcontrolerelor, precum și utilizarea întreruperilor periodice pentru gestionarea timpului. Senzorii de umiditate, temperatură, lumină și nivel de apă din rezervor oferă date esențiale pentru luarea deciziilor, iar sistemul reacționează automat în funcție de acești parametri printr-un automat de stare finit (FSM). Semnalizarea vizuală este realizată prin intermediul unor LED-uri, oferind feedback constant utilizatorului.

Contribuția a constat în: analiza cerințelor, definirea stărilor FSM, proiectarea circuitelor CLC pentru detecția erorilor, programarea logicii de irigare automată și realizarea interfeței de afișare. În urma implementării, sistemul obținut poate detecta autonom condițiile favorabile sau nefavorabile de irigare, gestionează în mod sigur procesul de activare a pompei și semnalizează eventualele erori apărute în sistem.

Rezultatele obținute confirmă atingerea obiectivelor propuse: sistemul funcționează corect în toate scenariile de lucru prevăzute, asigurând o irigare eficientă și o interfață de utilizare intuitivă pentru monitorizarea parametrilor critici.

1. Codificarea intrărilor și a ieșirilor. Descrieea funcționalității porturilor de intrare și ieșire

Se va proiecta un modul software, implementat ca proces secvențial (PS), destinat blocului de afișare al unui sistem automat de irigare inteligentă pentru seră, în conformitate cu următoarele specificații:

Intrări	**SW7**	**SW6**	**SW5**	**SW4**	**SW3**	**SW2**	**SW1**	**SW0**
	SET	RST	UMD	TEMPL	SU	STL	SUA	SAR

(tabel 1.1)

Ieșiri	**LED7**	**LED6**	**LED5**	**LED4**	**LED3**	**LED2**	**LED1**	**LED0**
	I/O	NivUmd		NivTempLumin		Err		

(tabel 1.2)

Butoanele **SW7–SW4** reprezintă comenzi cu acțiune pe nivel logic 0 (active la apăsare și revenire automată):

- **SET** – inițializarea procesului de configurare a parametrilor de irigare
- **RST** – resetarea sistemului în caz de eroare
- **UMD** – selectarea nivelului de umiditate a solului (mică, medie, mare)
- **TEMPL** – selectarea parametrilor de temperatură și lumină

Modul de funcționare poate fi:

- Programarea valorilor pentru umiditatea solului, temperatura și lumina;
- Pornirea automată a irigării atunci când senzorii detectează un nivel scăzut de umiditate în sol.

Butoanele **SW3–SW0** corespund senzorilor activi pe nivel logic 1, astfel:

- **SU** – senzor de umiditate a solului
- **STL** – senzor de temperatură și lumină
- **SUA** – senzor de umiditate atmosferică
- **SAR** – senzor pentru nivelul apei din rezervor

Secvența de programare a nivelului de umiditate a solului este următoarea:

1. Apăsarea butonului **SET** – intrarea în modul de configurare
2. Apăsarea butonului **UMD** sau **TEMPL** – selectarea parametrului dorit (umiditate sau temperatură/lumină)
3. Fiecare apăsare suplimentară pe **SET** modifică valoarea selectată (mică → medie → mare)
4. Apăsarea butonului **RST** – ieșirea din modul de configurare și resetarea parametrilor la valorile actuale

În cazul apariției unei erori, sistemul intră într-o stare de blocare. Reluarea funcționării normale se face prin apăsarea butonului **RST**, moment în care sistemul semnalizează resetarea prin aprinderea temporară a LED-urilor **LED7–LED0**.

LED-urile **LED7–LED0** se activează pe nivel logic 0 și indică starea de funcționare a sistemului după cum urmează:

Semnalizarea erorilor este realizată printr-un circuit logic combinațional (CLC).

Nivelurile de temperatură și umiditate sunt determinate folosind un circuit logic secvențial (CLS).

Resursele utilizate:

PORTD – configurat ca port de **intrare**;

PORTB – utilizat ca port de **ieșire**;

TIMER0 – configurat să genereze întreruperi periodice la fiecare **20ms**.

I/O	**Semnificație**
0	Irigare pornită
1	Irigare oprită/umiditate sol suficientă

(tabel 1.3)

NivTempLumin	**Semnificație**
10	Temperatură mică + lumină
01	Temperatură medie +lumină
00	Temperatură mare + lumină
11	Tempereatură mică + fără lumină

(tabel 1.4)

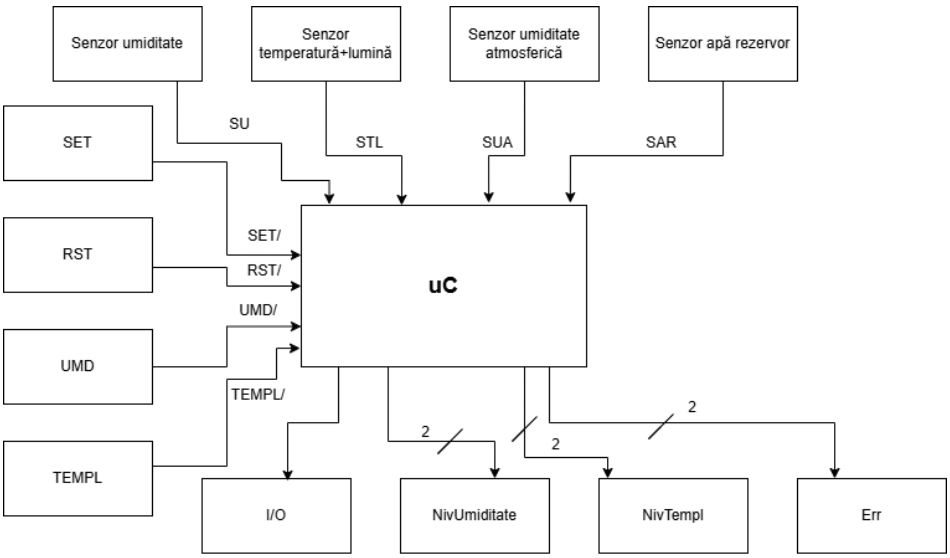
NivUmd	**Semnificație**
10	Umiditate mică
01	Umiditate medie
00	Umiditate mare

(tabel 1.5)

Err	**Semnificație**
111	Fără erori la senzori
110	Eroare la senzorii de umiditate
101	Eroare la senzorii de temperatură şi lumină
100	Eroare la senzorii de umiditate amosferică
011	Eroare la senzorii de apă din rezervor
010	Erore la oricare doi senzori in aceeași timp

(tabel 1.6)

Schema bloc a sistemului:



(figura 1.1)

2.Circuitul Combinational CLC - Tabela de adevăr și determinarea măștilor corespunzătoare

Pentru detectarea și semnalizarea erorilor apărute în sistemul automat de irigare, se utilizează un circuit logic combinațional (**CLC**). Acest circuit are rolul de a monitoriza starea de funcționare a senzorilor esențiali instalației și de a genera semnale de avertizare în cazul unei funcționări defectuoase.

Metoda de implementare

Implementarea circuitului CLC se face prin:

- **calcularea măștilor binare** pentru fiecare intrare relevantă (SW3–SW0),
- aplicarea unei **operații logice AND** între valorile curente ale portului de intrare și fiecare mască,
- **deplasarea** rezultatelor pentru a alinia biții într-o poziție standard (daca acestia nu sunt in ordinea corespunzatore),
- generarea valorii de ieșire corespunzătoare (Y = TAB[Index]).

Scopul și avantajele utilizării CLC:

- Detectează rapid și eficient orice combinație invalidă a senzorilor;
- Permite extinderea ușoară a codificării pentru mai multe erori simultane;
- Reduce complexitatea logicii de control din FSM (automat de stare finit);
- Asigură o separare clară între semnalizarea erorilor și controlul secvențial al sistemului.

Erorile pot apărea atunci când:

- senzorii de umiditate a solului (**SU**),
- senzorii de temperatură și lumină (**STL**),

- senzorii de umiditate atmosferică (**SUA**),
- sau senzorii de nivel al apei din rezervor (**SAR**)

nu furnizează date valide sau sunt inactivi în condiții critice. Acești senzori sunt conectați la intrările **SW3–SW0**, fiecare fiind activ pe nivel logic 1.

Astfel, **CLC-ul** va avea 4 biți de intrare, corespunzători acestor senzori, ceea ce generează un spațiu de 16 combinații posibile (2^4) în tabela de adevăr.

Ieșiri și codificare a erorilor

Semnalizarea erorilor din cadrul sistemului automat de irigare se realizează prin intermediul a **trei biți de ieșire**: **LED2, LED1 și LED0**, care se activează pe **nivel logic 0**. Acești biți codifică informația despre starea senzorilor esențiali (umiditate sol, temperatură și lumină, umiditate atmosferică și nivel apă din rezervor) și permit identificarea rapidă a erorilor individuale sau a erorilor multiple.

Pe baza combinațiilor posibile ale celor **patru senzori** (SU, STL, SUA, SAR), sistemul poate recunoaște și semnaliza:

- **Stare normală (fără erori)** – când toți senzorii funcționează corect;
- **Erori izolate** – câte un singur senzor defect sau inactiv;
- **Erori multiple** – mai mulți senzori simultan defectuoși;

Err	**Semnificație**
111	Fara erori la senzori
100	Eroare la senzorii de umiditate
011	Eroare la senzorii de temperatura si lumina
101	Eroare la senzorii de umiditate amosferica
110	Eroare la senzorii de apa din rezervor
000	Erore la mai multi senzori in acelasi timp

(tabel 2.1)

Calculul măștii binare:

Deoarece biții de intrare ai circuitului CLC sunt deja aliniați pe cei mai puțin semnificativi patru biți ai registrului de intrare, nu mai este necesară aplicarea unor operații de deplasare. Astfel, este suficientă utilizarea unei singure măști pentru filtrarea valorilor.

Masca asociată fiecărui bit de intrare este obținută prin setarea valorii 1 pe poziția corespunzătoare bitului dorit și 0 pe toate celelalte poziții.

Intrare	SET	RST	UMD	TEMPL	**SU**	**STL**	**SUA**	**SAR**
Mască	0	0	0	0	**1**	**1**	**1**	**1**
Index	0	0	0	0	**su**	**stl**	**sua**	**sar**

&

= 0xF

(tabel 2.2)

CLC-ul are la intrare 4 biti, putandu-se scrie 16 combinatii posibile:

SU	**STL**	**SUA**	**SAR**	**Semnificație**
0	0	0	0	-
0	0	0	1	Activ SAR
0	0	1	0	Activ SUA
0	0	1	1	Activ SUA și SAR
0	1	0	0	Activ STL
0	1	0	1	Activ STL și SAR
0	1	1	0	Activ STL, SUA și SAR
0	1	1	1	Activ STL,SUA și SAR
1	0	0	0	Activ SU
1	0	0	1	Activ SU și SAR
1	0	1	0	Activ SU și SUA
1	0	1	1	Activ SU, SAR, și SUA
1	1	0	0	Activ SU și STL
1	1	0	1	Activ SU, STL și SAR
1	1	1	0	Activ SU, STL și SUA
1	1	1	1	Activ SU, STL, SUA și SAR

(tabel 2.3)

Pornind de la semnificația celor 8 combinații posibile ale biților de intrare, se pot stabili corespunzător valorile ieșirilor, după cum urmează:

Intrare	**Ieșire**

SU	STL	SUA	SAR	Semnificație	LED2	LED1	LED0	Semnificație	Val Hexa
0	0	0	0	-	1	1	1	Fără erori	0xF
0	0	0	1	Activ SAR	1	1	0	Eroare la SAR	0x06
0	0	1	0	Activ SUA	1	0	1	Eroare la SUA	0x05
0	0	1	1	Activ SUA și SAR	0	0	0	Eroare la mai mulți senzori	0x00
0	1	0	0	Activ STL	0	1	1	Eroare la STL	0x03
0	1	0	1	Activ STL și SAR	0	0	0	Eroare la mai mulți senzori	0x00
0	1	1	0	Activ STL, SUA și SAR	0	0	0	Eroare la mai mulți senzori	0x00
0	1	1	1	Activ STL,SUA și SAR	0	0	0	Eroare la mai mulți senzori	0x00
1	0	0	0	Activ SU	1	0	0	Eroare la SU	0x00
1	0	0	1	Activ SU și SAR	0	0	0	Eroare la mai mulți senzori	0x00
1	0	1	0	Activ SU și SUA	0	0	0	Eroare la mai mulți senzori	0x00
1	0	1	1	Activ SU, SAR, și SUA	0	0	0	Eroare la mai mulți senzori	0x00
1	1	0	0	Activ SU și STL	0	0	0	Eroare la mai mulți senzori	0x00
1	1	0	1	Activ SU, STL și SAR	0	0	0	Eroare la mai mulți senzori	0x00
1	1	1	0	Activ SU, STL și SUA	0	0	0	Eroare la mai mulți senzori	0x00
1	1	1	1	Activ SU, STL, SUA și SAR	0	0	0	Eroare la mai mulți senzori	0x00

(tabel 2.4)

Tabela de adevăr a CLC-ului va fi definită ca vector TAB cu 16 elemente, după cum urmează:

char TAB[8] = { 0xf, 0x06, 0x05, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

Vom implementa circuitul logic combinaţional ca o funcţie ce poate fi apelată în interiorul codului pentru a verifica la diferite moemnte de timp functionarea optima a senzorilor ce capteaza data.

```
char in;
char eroare;
char TAB[16] = {0xf, 0x06, 0x05, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
     0x00, 0x00};
void CLC(void) {
   char selectie;
   selectie = in & 0x0F;     // selectăm doar cei 4 biți ai senzorilor
   eroare = TAB[selectie];   // determinăm codul de eroare din tabel
}
```

3. Graful asociat CLS

Circuitul logic secvențial (CLS) are un rol esențial în configurarea și selectarea parametrilor de funcționare ai sistemului automat de irigare, mai exact în stabilirea nivelului de umiditate a solului, precum și a condițiilor de temperatură și lumină din mediul ambient. Acest modul logic permite utilizatorului să personalizeze comportamentul sistemului în funcție de nevoile plantelor sau condițiile specifice din seră.

Pentru realizarea CLS-ului, este necesară construirea unui graf de tranziție a stărilor, care să reflecte toate posibilele configurații ale parametrilor și modul în care sistemul trece de la o stare la alta. Acest graf este compus din stări bine definite, iar tranzițiile sunt declanșate prin apăsarea succesivă a unui buton de setare(**SET**). Astfel, utilizatorul poate modifica treptat nivelul unui parametru, prin parcurgerea ciclică a valorilor disponibile.

Se vor folosi două circuite logice secventiale, unul pentru setarea nivelului de umiditate și unul pentru setarea temperaturii și al luminii de irigare.

Configurarea valorilor parametrilor se face în mod ciclic, fiecare apăsare a butonului de setare ducând la avansarea către nivelul următor.

Parametrul setării nivelului de umiditate poate fi setat în trei trepte distincte:

- Umiditate mică
- Umiditate medie
- Umiditate mare

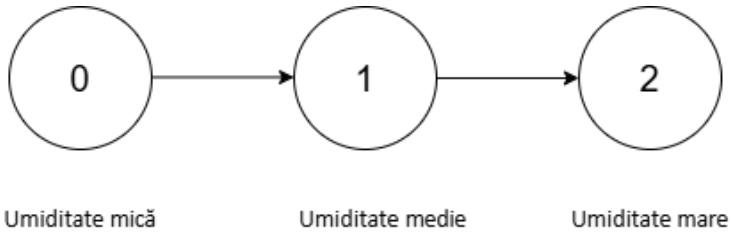
Parametrul setării nivelului de temperatură și lumină poate fi setat în patru trepte distincte:

- Temperatură mică +lumină,
- Temperatură medie +lumină,

- Temperatură mare +lumină,
- Temperatură mică + fără lumină

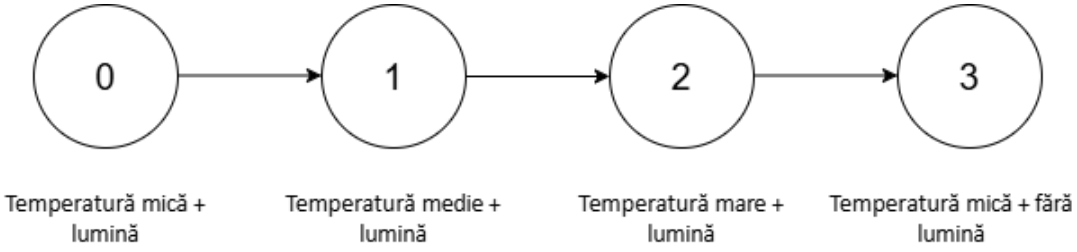
Prin urmare, pentru fiecare parametru în parte (umiditate, temperatură + lumină), CLS-ul definește trei, respectiv patru stări logice, corespunzătoare acestor niveluri.

Umiditate:



(figura 3.1)

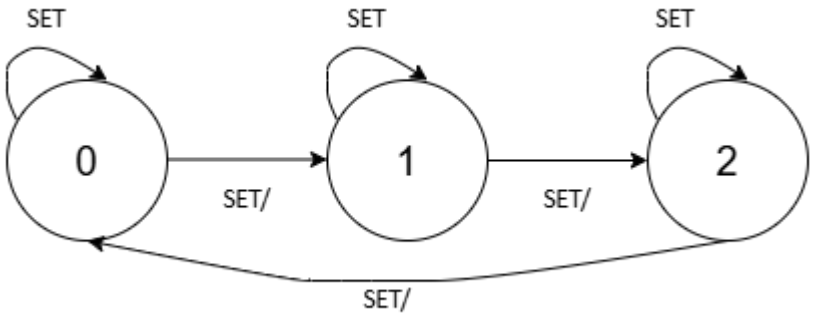
Temperatură+lumină:



(figura 3.2)

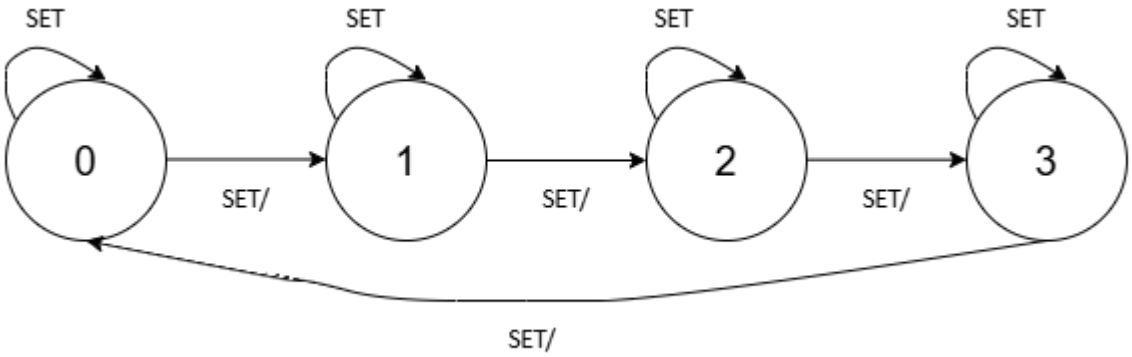
Tranzițiile între stările definite în cadrul grafului CLS sunt declanșate exclusiv prin acțiuni ale utilizatorului, mai precis prin apăsarea butonului **SET (SW7)**. Acesta funcționează ca un declanșator al modificării nivelului parametrului selectat. În absența unei acționări, sistemul rămâne în starea curentă, menținând valoarea configurată. Astfel, logica de comutare este simplă și intuitivă: o apăsare a butonului generează o tranziție către starea următoare, în timp ce lipsa unui impuls menține configurația activă.

Umiditate:



(figura 3.3)

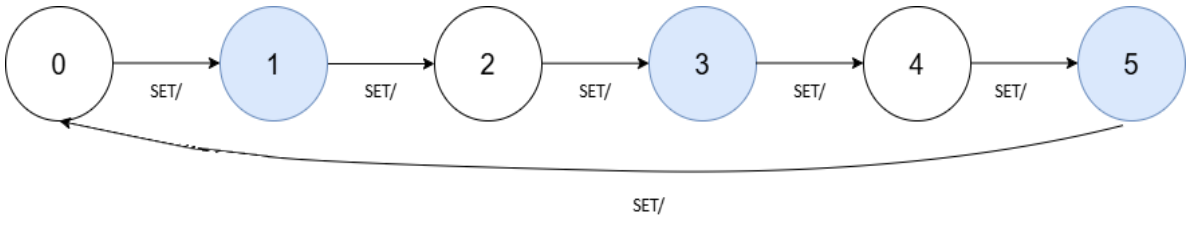
Temperatură+lumină:



(figura 3.4)

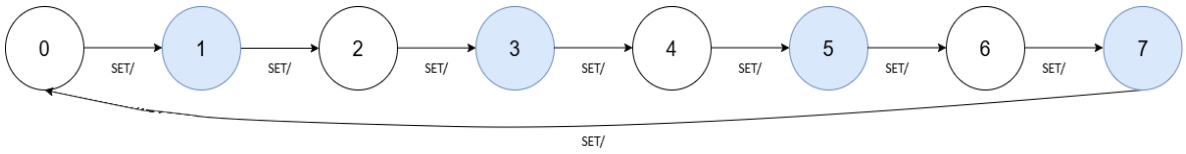
Butoanele sunt de tip cu revenire (apăsare și eliberare), ceea ce înseamnă că o apăsare este reprezentată prin două tranziții: una pentru apăsarea efectivă și alta pentru revenirea butonului. Având în vedere acest aspect, grafurile corespunzătoare CLS devin:

Umiditate:



(figura 3.5)

Temperatură+lumină:



(figura 3.6)

În funcționarea sistemului, acționarea butonului SET generează o tranziție de stare, determinând modificarea valorii umidității și a temperaturii+lumină.

Umiditatea sau **temperatura+lumina** nouă se stabilește exclusiv în momentul apăsării butonului **SET**, ceea ce corespunde stărilor numerotate 1, 3 și , respectiv 1,3,5 și 7 în diagramele de funcționare. În schimb, în stările ce corespund eliberării butonului SET, sistemul nu efectuează nicio modificare asupra valorii temperaturii.

Pentru o înțelegere detaliată a modului de funcționare, în tabelul de mai jos sunt prezentate toate stările sistemului, semnificația fiecărei stări, precum și valorile asociate ieșirii în acele stări.

Umiditate:

Stare	**Semnificație**	**Valoare ieșire**	
		Binar	**Zecimal**
0	Starea inițială – Umiditate mică	10	2
1	Apăsare SET – Umiditare medie	01	1
2	Eliberare SET	01	1
3	Apăsare SET – Umiditate mare	00	0
4	Eliberare SET	00	0
5	Apăsare SET – Umiditate Mică	10	2

(tabel 3.1)

Temperatură+lumină:

Stare	**Semnificație**	**Valoare ieșire**	
		Binar	**Zecimal**

0	Starea inițială – Temperatură mică + lumină	10	2
1	Apăsare SET – Temperatură medie + lumină	01	1
2	Eliberare SET	01	1
3	Apăsare SET – Temperatură mare + lumină	00	0
4	Eliberare SET	00	0
5	Apăsare SET – Temperatură mică + fără lumină	11	3
6	Eliberare SET	11	3
7	Apăsare SET – Temperatură mică + lumină	10	2

(tabel 3.2)

Conform teoriei Grafurilor cu Liste de Semnale (CLS), fiecărei stări Qi i se asociază un tabel propriu de semnale relevante, notat Ai. În cazul de față, grafurile CLS generează 6 stări pentru stabilirea **umidității** numerotate de la 0 la 5 și 8 stări pentru stabilirea **temperaturii+lumină** numerotate de la 0 la 7, pentru fiecare dintre acestea existând câte un tabel corespunzător cu semnale relevante. Cum am menționat și anterior tranzițiile între stări sunt declanșate prin apăsarea butonului **SET** (SW7).

Umiditate:

TAB

Stare	**Adresa tabelei de semnale relevante**
0	A0
1	A1
2	A2
3	A3
4	A4
5	A5

(tabel 3.3)

Temperatură+lumină:

TAB

Stare	Adresa tabelei de semnale relevante
0	B0
1	B1
2	B2
3	B3
4	B4
5	B5
6	B6
7	B7

(tabel 3.4)

Structura generală a unui tabel de semnale relevante este de forma: Ai={X_{i0}, Q_{i0},X_{i1}, Q_{i1},...,T,i} unde fiecare pereche (X_{ik}, Q_{ik}) reprezintă prin: X_{ik} un semnal relevant k activ în starea curentă i, iar Q_{ik} este starea în care sistemul va trece în urma apariției acelui semnal.

Ultima pereche din tabel este întotdeauna (T,i) care semnalează sfârșitul listei de semnale relevante pentru starea respectivă și indică faptul că, în absența altor semnale, sistemul rămâne în starea i.

Tabelele de semnale relevante asociate CLS, prezentate mai sus, pot fi descrise astfel:

Umiditate:

A0[]={0x00, 1, ‘T’, 0}
A1[]={0x80, 2, ‘T’, 1}
A2[]={0x00, 3, ‘T’, 2}
A3[]={0x80, 4, ‘T’, 3}
A4[]={0x00, 5, ‘T’, 4}
A5[]={0x80, 0, ‘T’, 5}

Temperatură+lumină:

B0[]={0x00, 1, ‘T’, 0}
B1[]={0x80, 2, ‘T’, 1}
B2[]={0x00, 3, ‘T’, 2}
B3[]={0x80, 4, ‘T’, 3}
B4[]={0x00, 5, ‘T’, 4}
B5[]={0x80, 6, ‘T’, 5}
B6[]={0x00, 7, ‘T’, 6}
B7[]={0x80, 8, ‘T’, 7}

Semnalul relevant asociat fiecărei stări poate avea două valori, în funcție de poziția butonului **SET**. Astfel, când butonul SET (SW7) este apăsat, semnalul este activ în nivel logic 0, corespunzător valorii hexazecimale 0x00. În schimb, dacă butonul nu este apăsat, semnalul are nivel logic 1, ceea ce corespunde valorii hexazecimale 0x80.

Pentru a permite compararea valorii de la intrare cu semnalul relevant, masca asociată are rolul de a izola bitul corespunzător butonului **SET** (SW7), realizând astfel selecția acestuia din întregul semnal de intrare.

Intrare	SET	RST	UMD	TEMPL	SU	STL	SUA	SAR
Mască	**1**	0	0	0	0	0	0	0

0X80

(tabel 3.5)

Astfel, comparaţia se va realiza între rezultatul $\boldsymbol{IN}$ **&** $\boldsymbol{0x80}$ şi valorile semnalelor relevante – $0x00$ sau $0x80$.

Similar cu implementarea circuitului logic combinațional, și CLS-urile poate fi realizate sub forma unor funcții, apelate în contextul setării umidității și temperaturii+lumină pentru parametrii de irigare. În vederea construirii codului, se pot lua în considerare următoarele aspecte de implementare:

- Valoarea terminatorului de tabel poate fi definită ca o constantă utilizând directiva #define;
- Numărul total de stări ale CLS-ului poate fi, de asemenea, definit ca o constantă cu ajutorul directivei #define;
- Variabilele ce descriu intrarea, ieșirea și starea curentă a CLS pot fi declarate ca variabile globale de tip char;
- Tabela de adrese va fi definită sub forma unui pointer de tip char;
- Tabelele de semnale relevante vor fi implementate sub forma unor matrice de tip char;
- Ieșirile asociate fiecărei stări CLS vor fi stocate într-o matrice de tip char.

Codurile asociate circuitelor logic secvenţiale este următorul:

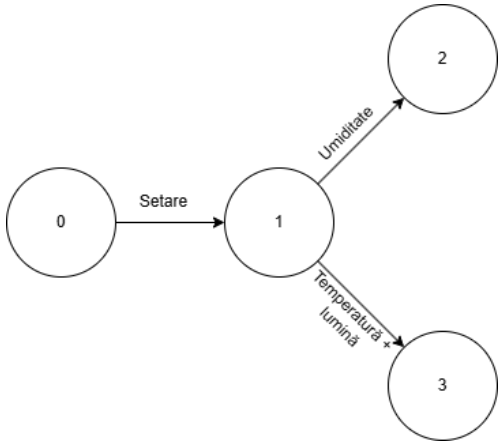
```
#define T         0x10
#define NR1 6
#define NR2 8
char Q;
char in;
char out;
char *TABA1[NR1];
char *TABA2[NR2];
char A0[] = {0x00,1,T,0};
char A1[] = {0x80,2,T,1};
char A2[] = {0x00,3,T,2};
char A3[] = {0x80,4,T,3};
char A4[] = {0x00,5,T,4};
char A5[] = {0x80,0,T,5};
```

```
char B0[] = {0x00,1,T,0};
char B1[] = {0x80,2,T,1};
char B2[] = {0x00,3,T,2};
char B3[] = {0x80,4,T,3};
char B4[] = {0x00,5,T,4};
char B5[] = {0x80,0,T,5};
char B6[]={0x00, 7, T, 6}
char B7[]={0x80, 8, T, 7}

char Tout1[] = {0x02, 0x01, 0x01, 0x00, 0x00, 0x02};
char Tout2[] = {0x02, 0x01, 0x01, 0x00, 0x00, 0x03, 0x03, 0x02};
TABA1[0]=A0;
TABA1[1]=A1;
TABA1[2]=A2;
TABA1[3]=A3;
TABA1[4]=A4;
TABA1[5]=A5;

TABA2[0]=B0;
TABA2[1]=B1;
TABA2[2]=B2;
TABA2[3]=B3;
TABA2[4]=B4;
TABA2[5]=B5;
TABA2[6]=B6;
TABA2[7]=B7;
Q=0;

void cls1(void) {
    char i;
    char *adr;
    char ready;
    adr = TABA1[Q];
     i=0;
     ready=0;
    while (!ready)
    {
        if ((in & 0x80) == *(adr+i)) { Q =*(adr+i+1); ready=1; }
            else if (*(adr+i) == T) ready=1;
                else i += 2;
    }
}

void cls2(void) {
    char i;
    char *adr;
    char ready;
    adr = TABA2[Q];
     i=0;
     ready=0;
    while (!ready) {
```

```
    if ((in & 0x80) == *(adr+i)) { Q =*(adr+i+1); ready=1; }
       else if (*(adr+i) == T) ready=1;
          else i += 2;
}
        }
```

4.Automatul de stări finite

Funcționarea modulului software al sistelului automat de irigare inteligentă pentru setă urmează un proces secvențial, bazat pe o succesiune de stări și tranziții. La construirea grafului asociat programului secvențial (PS), se pornește de la o stare inițială – starea 0 – în care sistemul de irigare este activ, dar se află în așteptarea comenzilor utilizatorului. Din această stare, sistemul poate evolua pe două direcții distincte, corespunzătoare următoarelor tranziții: setarea parametrilor de irigare pentru umiditate(**UMD**) și setarea parametrilor de irigare pentru temperatură + lumină(**TEMPL**).



(figura 4.1)

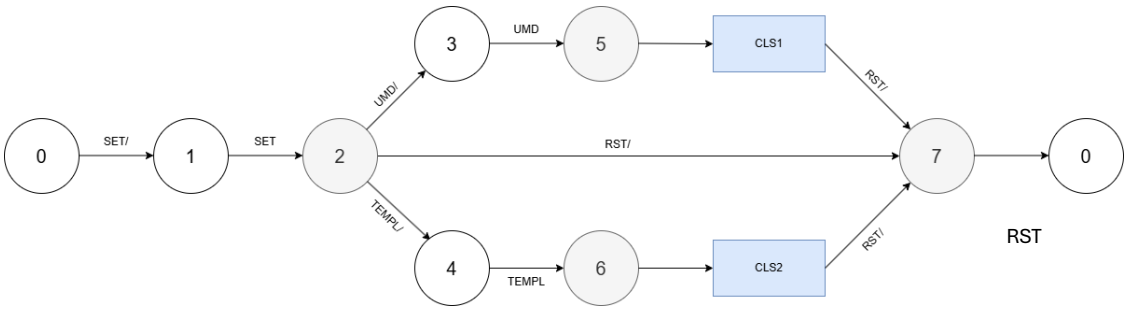
După selectarea unui parametru de irigare, din cei doi mentionati anterior, setarea parametrilor se face astfel:

În ceea ce privește modul de reglare a umidității, trecerea în această stare se realizează prin apăsarea butonului **UMD** (SW5). Odată activat, parametrul de umiditate poate fi ajustat în trei trepte (umiditate scăzută, umiditate medie, umiditate ridicată), prin apăsări succesive ale butonului **SET** (SW7). Ieșirea din modul de configurare și salvarea noilor valori ale parametrilor de umiditate se realizează prin apăsarea butonului RST (SW6).

În cazul regimului combinat temperatură + lumină, activarea se face prin apăsarea butonului **TEMPL** (SW4). În această stare, parametrul poate fi setat în patru trepte distincte: temperatură scăzută cu lumină, temperatură medie cu lumină, temperatură ridicată cu lumină, respectiv temperatură scăzută fără lumină. Selectarea treptei dorite se realizează tot prin apăsări succesive ale butonului **SET** (SW7). La final, ieșirea din modul de configurare și salvarea valorilor stabilite pentru parametrii de temperatură și lumină se face prin apăsarea butonului RST (SW6).

Există posibilitatea ca în urma apăsării a unui dintre cele doua butoane **UMD/TEMPL**, utilizatorul să renunţe la programarea temperaturii şi să revină în starea iniţială prin apăsarea butonului **RST**.

Ținând cont de faptul că modificarea parametrilor de irigare se realizează prin intermediul circuittelor logice secvenţiale (CLS), graful asociat programării parametrilor devine:



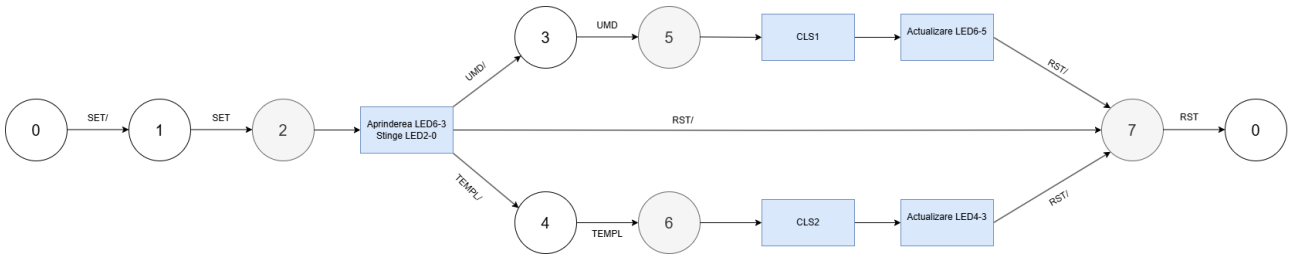
(figura 4.2)

În prezent, graful redă doar tranzițiile între stări, fără a evidenția și ieșirile asociate fiecărei stări. Pentru o descriere completă a comportamentului sistemului, se pot include următoarele aspecte:

- Prina apăsarea butonului de setare SET se determină aprinderea LED-urilor corespunzătoare parametrilor curenți ai modurilor **UMD** și **TEMPL**, concomitent cu stingerea celorlalte LED-uri.
- Ieșirea din starea **CLS1** aferentă modului de setare **UMD** are ca efect actualizarea LED-urilor ce indică valoarea noului parametru al umidității.
- Ieșirea din starea **CLS2** aferentă modului de setare **TEMPL** determină actualizarea LED-urilor corespunzătoare noii valori ai parametrului ce indică temperatura + lumina.
- Revenirea în starea inițială din modul de programare implică stingerea tuturor LED-urilor ce indica valoarea parametrilor de irigare.

  Având în vedere aceste completări, graful de stare poate fi extins astfel:

(figura 4.3)



Cât timp utilizatorul nu apasă niciun buton sistemul ilustrează prin aprinderea ledului LED7 daca irigarea este activă sau oprită și posibilele erori ale senzorilor de umiditate, temperatură + lumină, umiditate atmosferică și apă in rezervor, pe LED-urile LED2-0. Irigarea se va opri în momentul în care senzorii detectează un nivel optim de umiditate, temperatură + lumină, umiditate atmosferică și apă in rezervor.

În situația apariției unei erori, sistemul intră într-o stare de blocare, motiv pentru care este necesară definirea unei stări distincte, dedicată gestionării acestei condiții. Pentru reluarea funcționării normale, este necesară apăsarea butonului **RST**, care determină stingerea și aprinderea tuturor LED-urilor, urmată de reinițializarea sistemului prin revenirea în **starea 0**

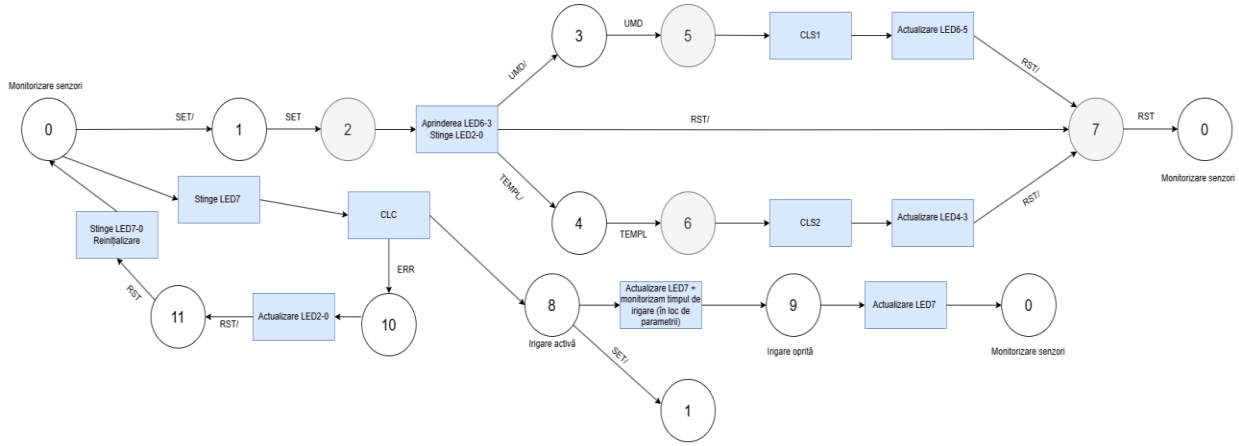
Am modificat puțin funcționarea sistemului pentru testare, deoarece am realizat ulterior că, prin limitarea numărului de 8 switch-uri, nu putem introduce valori specifice senzorilor. Era mai util dacă cei 4 senzori erau pe 2 biți fiecare, astfel aveam diferite combinații pentru aceștia și puteam simula niște valori obținute de ei. De exemplu, pentru senzorul de umiditate sol UMD puneam următoarele semnificații:

- 11 – eroare senzor
- 01 – umiditate nivel mic sol
- 10 – umiditate medie/mare sol

Astfel, cu aceste codări puteam face verificări pentru a vedea dacă parametrii actuali ai solului și ai mediului înconjurător satisfac cerințele introduse de utilizator, iar dacă acești parametri nu corespund, vom porni irigarea solului pentru a se îndeplini cererile utilizatorului.

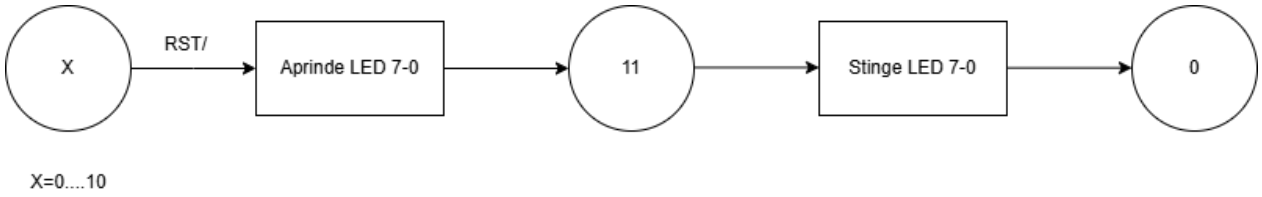
Modificarea adusă este că vom porni automat irigarea dacă nu există erori, fără a mai ține cont de parametrii introduși de utilizator; aceasta va fi pornită pentru un anumit interval de timp, iar după acest interval irigarea se va opri. Utilizatorul poate în continuare sa modifice cei doi parametrii de irigare (umiditate și temperatura + lumina), însă pornirea irigării nu depind de acestea.

În aceste condiții, graful de stare se modifică astfel:



(figura 4.4)

In plus in orice stare se poate apasa butonul RST, coresprunzatoare reinitializarii, astfelm graful poate fi completat cu urmatoarele tranzitii:



(figura 4.5)

5.Codul in limbaj C pentru întreg modulul software(CLC,CLS și PS):

```
#include <mega164.h>

#define T       0x10  // terminator CLS
#define NR1     6     // numar stari CLS umiditate
#define NR2     8     // numar stari CLS temp+lumină

char err;        // cod erori (bitii 0..2)
char umd;        // nivelul de umiditate
char templ;      // nivelul de temperatura + lumina
char S;          // starea PS
char Q;          // starea CLS curenta
char in;         // PIND
char out;        // PORTB
int timp = 0;

//  tabela CLC
char TAB[] = {
   0xF,0x06,0x05,0x00,
   0x03,0x00,0x00,0x00,
   0x00,0x00,0x00,0x00,
   0x00,0x00,0x00,0x00
};

//  tabele CLS umiditate
char *TABA1[NR1];
char A0[] = {0x00,1, T,0};
char A1[] = {0x80,2, T,1};
char A2[] = {0x00,3, T,2};
char A3[] = {0x80,4, T,3};
char A4[] = {0x00,5, T,4};
char A5[] = {0x80,0, T,5};
char Tout1[] = { 0x02,0x01,0x01,0x00,0x00,0x02};

// tabele CLS temp+lumină
char *TABA2[NR2];
char B0[] = {0x00,1, T,0};
char B1[] = {0x80,2, T,1};
char B2[] = {0x00,3, T,2};
char B3[] = {0x80,4, T,3};
char B4[] = {0x00,5, T,4};
char B5[] = {0x80,6, T,5};
char B6[] = {0x00,7, T,6};
char B7[] = {0x80,0, T,7};
char Tout2[NR2] = { 0x02,0x01,0x01,0x00,0x00,0x03,0x03,0x02 };

void test_RST(void) {
   if ((in & 0x40) == 0)  //testam resetul RST
   {
```

```
        out = out & 0x00;  // aprinde toate LED-urile
        S = 11;
    }
}

void clc(void)
{
    char senzor = in & 0x0F;
    err = TAB[senzor];  //eroare bitii 0,1,2
    if (err != 0x07) S = 10; // stare eroare
        else
        {
            out=(out | 0xFF) & 0X7F;//irigarea activa
            S = 8;
        }
}

void cls1(void) {
    char i;
    char *adr;
    char ready;
    adr = TABA1[Q];
     i=0;
     ready=0;
    while (!ready)
    {
        if ((in & 0x80) == *(adr+i)) { Q =*(adr+i+1); ready=1; }
            else if (*(adr+i) == T) ready=1;
                else i += 2;
    }
}

void cls2(void) {
    char i;
    char *adr;
    char ready;
    adr = TABA2[Q];
     i=0;
     ready=0;
    while (!ready) {
        if ((in & 0x80) == *(adr+i)) { Q =*(adr+i+1); ready=1; }
            else if (*(adr+i) == T) ready=1;
                else i += 2;
    }
}

// ISR PS + CLC + CLS
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    TCNT0 = 0x3C;//reinitializam timerul la 0
```

```
in = PIND;

switch (S) {
   case 0:
      if ((in & 0x80) == 0) S = 1;//testez SET/
         else
            {
               out = (out | 0xFF) & 0x7F;  //stingem ledul ce indica daca irigarea este pornita sau oprita
               clc();
            }
      test_RST();//testam daca s a apasat RST/
      break;

   case 1:
      if ((in & 0x80) != 0) S = 2; //testam SET
      test_RST();
      break;

   case 2:
      out = (out & 0x00) | 0x87; // LED6-3 pornite
      if ((in & 0x20) == 0) S = 3;  //testam UMD/
      if ((in & 0x10) == 0) S = 4;  //testam TEMPL/
      if ((in & 0x40) == 0) S = 7;  //testam /RST
      test_RST();
      break;

   case 3:
      if ((in & 0x20) != 0) S = 5;
      test_RST();
      break;

   case 4:
      if ((in & 0x10) != 0) S = 6;
      test_RST();
      break;

   case 5:
      cls1();
      umd = Tout1[Q] << 5;//iesire cls bitii 5,6
      out = out & 0x9F;//stergem bitii 5, 6 - masca 10011111
      out = out | umd;//actualizez out
      if ((in & 0x40) == 0) S = 7; //testam /RST
      test_RST();
      break;

   case 6:
      cls2();
      templ = Tout2[Q] << 3;//iesire cls bitii 4,3
      out = out & 0xE7;//stergem bitii 4, 3 - masca 11100111
      out = out | templ;//actualizez out
      if ((in & 0x40) == 0) S = 7; //testam /RST
```

```
    test_RST();
    break;

case 7:
    if ((in & 0x40) != 0) S = 0;//testez RST
    test_RST();
    break;

case 8:
    if ((in & 0x80) == 0) S = 1; //testam SET/
    timp = (timp+1) % 10; //ca sa contorizam timpul, am pus 10 ca sa mearga mai rapid in simulare
    if ((in & 0xF) != 0)   //testez senzorii
     {
        out = (out | 0xFF) & 0x7F;
        S = 9;
     }
     else if(timp == 0)
     {
        out = (out | 0xFF) & 0x7F;
        S = 9;
     }
    test_RST();
    break;

case 9:
    out = (out | 0xFF) & 0xFF;
    S = 0;
    timp = 0;
    test_RST();
    break;

case 10:
    out = out & 0xFF; // inchid toate becurile
    out = (out | err) << 0; // actualizeaza out
    out = out | 0xF8; //inchid toate LED-urile in afara de cele de eroare LED 2-0
    test_RST();// testeaza RST/
    break;

 case 11:
    if ( (in & 0x40) != 0 ) // testeaza RST
        {
            out = out | 0xFF; // stinge LED7-0
            err=0x07;
            out = out | err; // sterge erori
            S=0;
        }
    test_RST();// testeaza RST/
    break;
    }

    PORTB = out; //iesirea
```

```
}


void main(void) {
   out = 0xFF;
   S = 0; Q = 0;
   TABA1[0] = A0; TABA1[1] = A1; TABA1[2] = A2;TABA1[3] = A3; TABA1[4] = A4; TABA1[5] = A5;
   TABA2[0] = B0; TABA2[1] = B1; TABA2[2] = B2; TABA2[3] = B3;TABA2[4] = B4; TABA2[5] = B5;
TABA2[6] = B6; TABA2[7] = B7;
   while (1) { }
}
```

6.Debugging:

Nu reușesc să găsesc ce este greșit în cod și să-l fac să funcționeze în AVR Studio. Am rescris codul de 2 ori de la zero și se comportă aproape identic.

7.Concluzii:

Am atins parțial obiectivul proiectului „Sistem Automat de Irigare Inteligentă bazat pe Microcontroler pentru Monitorizarea și Controlul Condițiilor din Seră”, implementând o soluție care, pe baza datelor furnizate de senzorii de umiditate a solului, temperatură și lumină, umiditate atmosferică și nivelul apei din rezervor, declanșează automat pompa de irigare atunci când parametrii nu corespund valorilor setate și o oprește imediat ce ating valorile optime . (ulterior realizând codul cu o modificare datorită limitării programului de testare – se declanșează irigarea automat dacă nu apare nicio eroare). Semnalizarea vizuală prin LED-uri ne oferă feedback constant, iar la apariția oricărei erori la senzori, sistemul intră în stare de eroare până la resetarea manuală cu butonul RST, reinițializându-se ulterior . Contribuția practică a constat în definirea stărilor automatului finit, proiectarea circuitului logic combinațional pentru detectarea erorilor, a circuitelor secvențiale pentru configurarea parametrilor și programarea logicii microcontrolerului, demonstrând în toate scenariile de lucru că funcționează corect și oferind o interfață intuitivă pentru monitorizarea și controlul procesului de irigare