Proiect Arduino
programarea orientate pe obiecte

Sistem de comandă pentru irigații

Studenții:

Caraibot Andreea

Cioabă Armina-Mihaela

Cioplea Emil

An II grupa 2

Timișoara 2019

Cerințe inițiale

Componente:

1. Arduino Uno
2. LED RGB pentru semnalarea stării de umiditate (albastru – umed; verde – uscat)
3. 2 LED – roșu și verde pentru oprit/pornit sistem
4. Senzor de umiditate sol
5. Ecran LCD
6. Buton brick

La apăsarea butonului se pornește/oprește sistemul și se aprinde LED-ul roșu/verde.

În funcție de umiditatea măsurată la un interval de timp se citește valoarea senzorului de umiditate și se afișează pe ecranul LCD. Dacă senzorul de umiditate citește o valoare peste un prag prestabilit udarea nu are loc în ziua curentă și se aprinde LED-ul RGB de culoare albastră.

În Figura 1 este prezentată implementarea hardware a sistemului în programul de simulare.

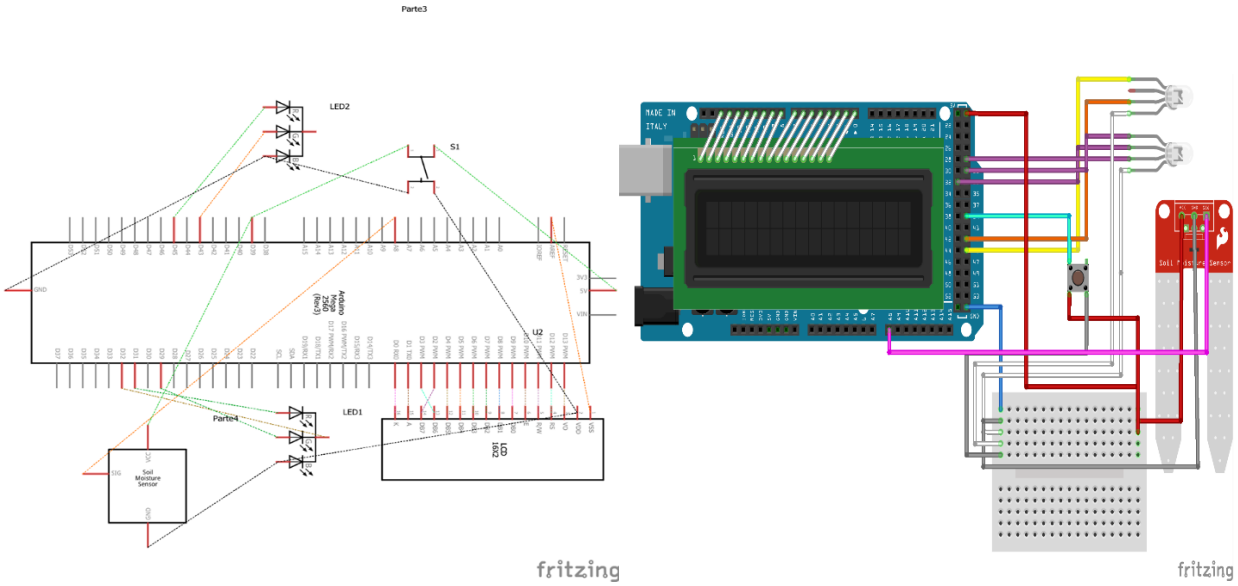

Figura 1. Schema bloc a sistemului și schema electrică

Sistemul implementează irigarea unei grădini în funcție de factorul de umiditate. Principalele funcții ale sistemului sunt cele de afișare pe ecranul LCD a umidității citite de la senzor, cât și

aprinderea unor leduri conform condițiilor legate de nivelul de umiditate și depășirea unui prag stabilit, de asemenea, posibilitatea de a porni sau opri sistemul prin comanda butonului.

Modul de funcționare a sistemului:

Sistemul este realizat după principiul automatelor cu stări finite. Se folosesc trei stări: starea inițială, starea pornit și starea oprit. La început sistemul se găsește în starea inițială, în care ledul albastru este pornit si restul sunt oprite. După apăsarea butonului brick, sistemul pornește (stare pornit), în consecință, se începe verificarea umidității. În continuare, este citită valoarea senzorului la un interval de timp de o secundă, făcându-se media între valorile citite. În cazul în care valoarea citită de la senzorul de umiditate depășește pragul stabilit (se aprinde ledul RGB albastru), nu va avea loc irigarea în ziua curentă (delay mai mare) și se va afișa pe ecranul LCD un mesaj corespunzător. În cazul în care valoarea citită de la senzor nu depășește pragul (se aprinde ledul RGB verde), va avea loc irigarea; valoarea citită de la senzor se va verifica din nou peste un interval scurt de timp (delay 1s). Valoarea senzorului de umiditate va fi citită periodic din oră în oră. La o altă apăsare a butonului sistemul se va opri (starea oprit).

În Figura 2 este prezentată schema bloc a software-lui care comandă sistemul pentru irigații (schema bloc a fost realizată folosind schemele UML):

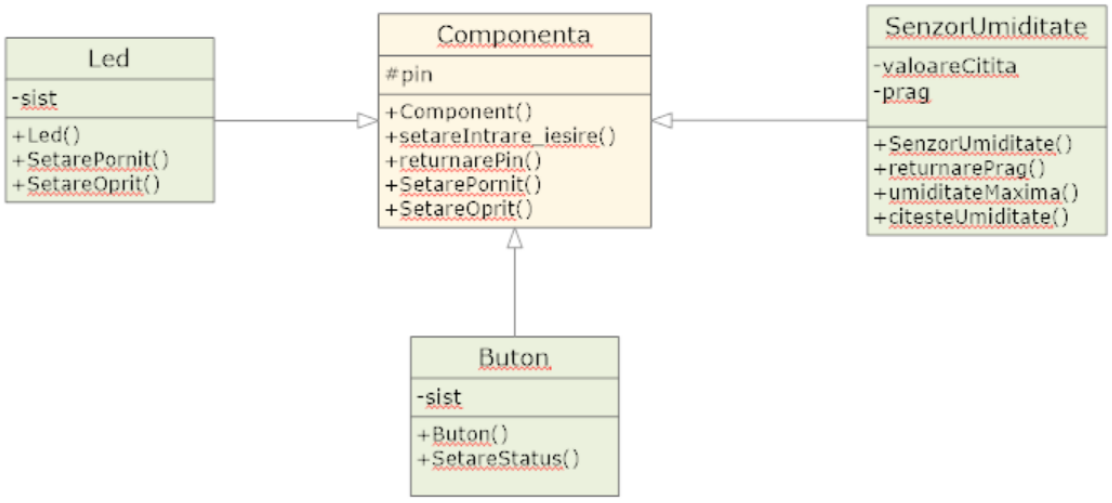

Figura 2. Schema UML a sistemului

Din schema UML din Figura 2 se poate observa clasa de bază Componenta și clasele derivate: Led, Buton, SenzorUmiditate. Clasa de bază are rolul de a seta pinii prin care sunt conectate echipamentele. Restul claselor oferă funcții pentru interfațarea perifericelor precum senzorul de umiditate, ledurile, butonul și ecranul LCD.

Schema logică de funcționare a softului din interiorul microcontrolerului este redată în Figura 3. Această schemă ne prezintă modul de funcționare a sistemului și ne oferă o posibilitate de a crea diverse scenarii de testare.

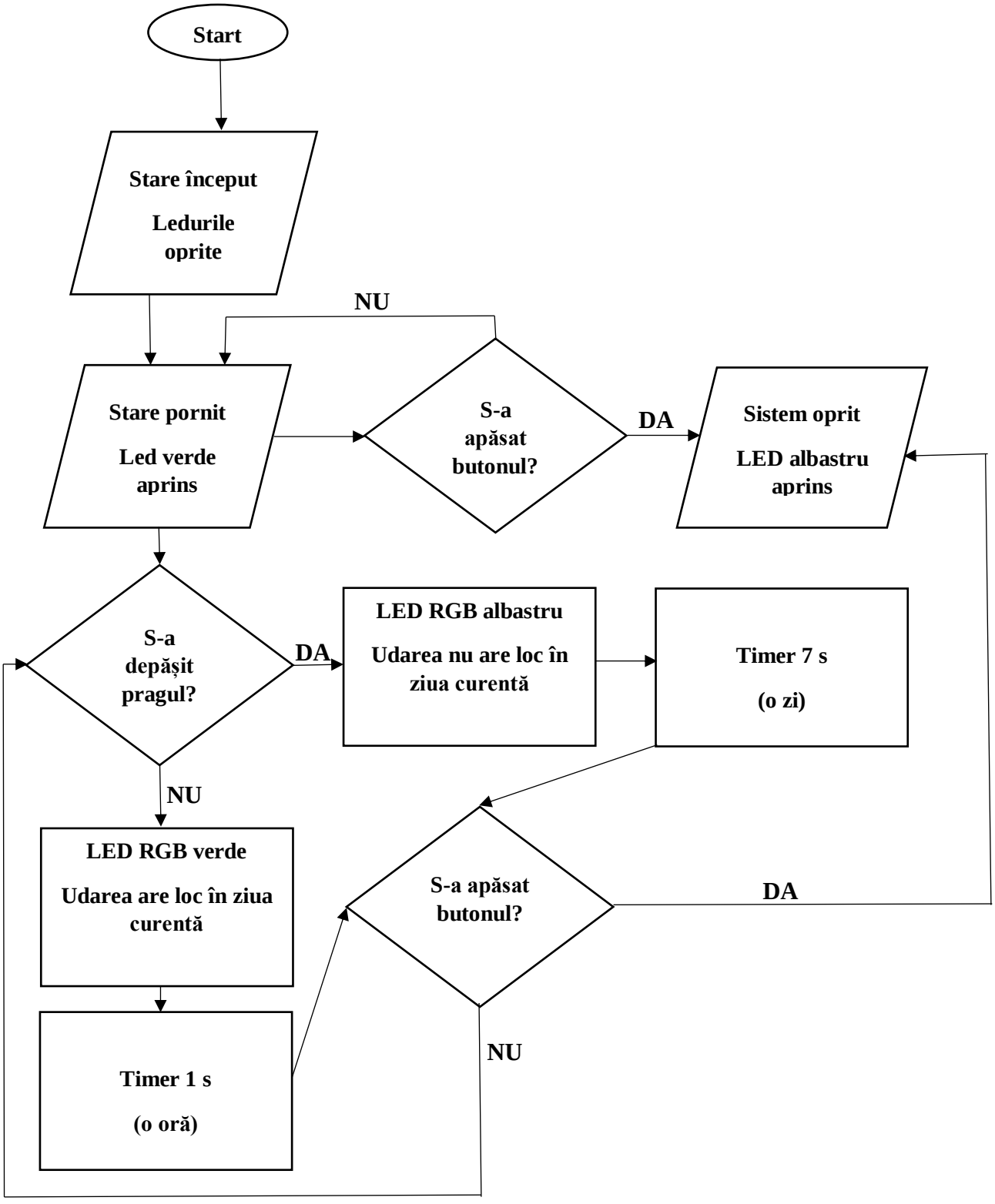

Figura 3. Schema logică a programului de comandă

Testarea aplicației

Acțiunea testelor	Răspuns așteptat	Răspuns real
Apăsarea butonului	- Pornirea sistemului și funcționarea în parametri normali a acestuia - Oprirea sistemului – întreruperea acțiunilor sistemului	- Se observă faptul că la apăsarea butonului mai insistentă se obține schimbarea de stări. - Ledul care indică starea sistemului nu acționează conform așteptărilor (sunt arse culori). - Observăm că după prima verificare a umidității, ledul de stare a sistemului se stinge.
Modificarea condițiilor de mediu pentru senzorul de umiditate a solului	- Aprinderea ledului RGB verde la udarea solului în caz de umiditate scăzută (sub prag). - Aprinderea ledului RGB albastru pentru valori ale umidității care depășesc pragul (prag 100).	- Ledul se va aprinde primul, iar afișajul pe ecran va avea delay. - Funcționează conform așteptărilor.

Concluzii

În urma realizării acestui proiect ne-am aprofundat cunoștințele în domeniul programării orientate pe obiecte (C++), întrucât am folosit principii de lucru specifice acestuia: abstractizare, încapsulare, am folosit clase (Component – clasă de bază, Led- clasă derivată, etc.) și instanțe ale acestora, moștenire, funcții virtuale și multe altele. De asemenea, am avut șansa să punem în practică C++ în domeniul microcontrolerelor.

Pe parcursul proiectului am întâmpinat diverse probleme, unele legate de lucrul cu clasele și organizarea optimă a acestora, altele legate de softul de simulare, care nu dispunea de toate componentele utilizate în cadrul aplicației. O altă problemă tehnică întâmpinată a fost încărcarea codului pe plăcuță (din cauza unor fire conectate greșit - scurtcircuit), precum și omiteri în logica programului descoperite în faza de testarea a aplicației.

Detaliile care au condus la buna desfășurare a lucrurilor și finalizarea proiectului a fost munca în echipă, o bună gestionarea a sarcinilor de lucru, comunicarea eficientă.

Codul sursă

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(7,6,5,4,3,2);
int stare;
//enumerare care se folosește pentru setarea sistemului
typedef enum
{
 stare_oprit, stare_pornit, stare_inceput
}stare_sistem;

//clasa de bază
class Componenta
{
 protected:
 int pin;
 public:
 //constructor
 Componenta(int p)
 {
 pin = p;
 }
 //funcție pentru setarea intrării sau ieșirii
 void setareIntrare_iesire(int intrare_iesire)
 {
  pinMode(pin, intrare_iesire);
 }
 //funcție pentru returnarea pin-ului
 int returnarePin()
```

```
 {
 return pin;
 }
 //funcție virtuală pentru oprire a unei componente
 virtual void SetareOprit() { }
 //funcție virtuală pentru pornire a unei componente
 virtual void SetarePornit() { }
};
//clasa derivată Led din clasa de bază Componenta
class Led : public Componenta {
private:
 stare_sistem sist;
public:
 Led(int pin, stare_sistem s) : Componenta(pin)  //constructorul clasei derivate
{
 sist = s;
}
//setare pornire led
void SetarePornit()
{
 digitalWrite(pin, HIGH);
 sist = stare_pornit;
}
//setare oprire led
void SetareOprit()
{
 digitalWrite(pin, LOW);
 sist = stare_oprit;
```

```
}
};

//se folosește pentru senzorul de umiditate
class SenzorUmiditate : public Componenta {
private:
 //valoarea citită de la senzor
 int valoareCitita;

//valoare pragului

int prag;
public:
 SenzorUmiditate(int pin, int prg) : Componenta(pin)
{
 prag=prg;

}
//returnează valoarea de min/max a umidității
int returnarePrag()
{
 return prag;
}

//funcție folosită care ajută la aflarea valorii maxime a umidității dintr-un interval de tip
int umiditateMaxima(int , int );
//funcție care returnează valoarea umidității
int citesteUmiditate();
```

```
};
//funcție care returnează valoarea maximă a senzorului de umiditate
int SenzorUmiditate::umiditateMaxima(int pin, int count)
{
 //variabila care se folosește pentru determinarea maximului
 int valMaxima = 0;

 for(int i = 0; i < count; i++)
 {
 valMaxima = max(analogRead(pin), valMaxima);
 }
 return valMaxima; }
//funcție care returnează valoare citită de la senzorul de umiditate
int SenzorUmiditate :: citesteUmiditate() {
 valoareCitita = umiditateMaxima(pin, 1000);
 return valoareCitita;
}
//clasa derivată din clasa de bază Componenta
//butonul este folosit pentru a opri sau a începe funcționarea sistemului
class Buton : public Componenta
{
private:
 stare_sistem sist_stat;
public:
 //constructor
 Buton(int pin, stare_sistem s) : Componenta(pin) {
 sist_stat = s;
}
```

```
//setarea butonului
void SetareStatus(stare_sistem s) {
 sist_stat = s; }
};
//declararea variabilelor folosite în program
stare_sistem stareSistem = stare_inceput;
Led LedAlbastru1(47, stare_oprit);
Led LedVerde1(49, stare_oprit);  //pentru stare
Led LedVerde2(33, stare_oprit); //pentru udat
Led LedAlbastru(31, stare_oprit);
int umiditate;
Buton buton1(43, stare_oprit);  //butonul pentru oprirea și pornirea sistemului
SenzorUmiditate s_umiditate(A8, 100);  //senzorul de umiditate
//funcție folosită pentru a schimba starea sistemului la apăsarea butonului
void schimbareStareSistem()
{
 //dacă sistemul este oprit sistemul se va întoarce
 if (stareSistem == stare_oprit)
 {
 stareSistem = stare_pornit;  //în loc de început
 }
 else
 if (stareSistem == stare_pornit)
 {
 stareSistem = stare_oprit;
 }
}
//inițializarea modului în care sunt setați pinii fiecărei componente (intrare / ieșire)
```

```
void setup()
{
 lcd.begin(16,2);
 lcd.clear();
 //1 umed, 2 uscat;
 LedAlbastru1.setareIntrare_iesire(OUTPUT);
 LedVerde1.setareIntrare_iesire(OUTPUT);  //stare
 LedVerde2.setareIntrare_iesire(OUTPUT);  //udare
 LedAlbastru.setareIntrare_iesire(OUTPUT);
 //buton
 buton1.setareIntrare_iesire(INPUT);
 s_umiditate.setareIntrare_iesire(INPUT);
stareSistem= stare_oprit;
}

//codul propriu-zis ce îndeplinește funcția cerută
void loop()
{
 //se verifică în ce stare este sistemul
 switch (stareSistem)
 {
 //cazul în care sistemul este oprit; toate ledurile sunt oprite
 case stare_oprit:
 lcd.clear();
 lcd.setCursor(0,0);
lcd.print("Sistem Irigatii");
 LedAlbastru1.SetarePornit();
```

```
 LedVerde1.SetareOprit();
 LedVerde2.SetareOprit();
 LedAlbastru.SetareOprit();
 delay(1000);
 if (digitalRead(buton1.returnarePin()) == HIGH)
{
 schimbareStareSistem();
}
 break;
 //cazul în care sistemul a intrat în starea de început a programului
 case stare_inceput:
 LedVerde1.SetarePornit();
 stareSistem = stare_pornit;
 break;
 case stare_pornit:
 umiditate = s_umiditate.citesteUmiditate();
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Umiditate: ");
lcd.print(umiditate);
delay(5000);
 if (s_umiditate.citesteUmiditate() < s_umiditate.returnarePrag())
 stare=1;
else
stare=2;

if(stare==1)
{
```

```
 LedAlbastru1.SetareOprit();
 LedVerde1.SetarePornit();
 LedVerde2.SetarePornit();
 LedAlbastru.SetareOprit();
 delay(1000);
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Sol uscat");
 lcd.setCursor(0,1);
 lcd.print("Pornim udarea");
 delay(1000);
 }
 else
 if(stare==2){
 LedAlbastru1.SetareOprit();
 LedVerde1.SetarePornit();
 LedVerde2.SetareOprit();
 LedAlbastru.SetarePornit();
 delay(1000);
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Sol umed");
 lcd.setCursor(0,1);
 lcd.print("Oprim udarea");
 delay(5000);
 }
 if (digitalRead(buton1.returnarePin()) == HIGH)
{
```

```
 schimbareStareSistem();
 break;
}
  break;
 }
}
```