Raport SÎ

Ciobanu Victor

Ciobanu Nicolae

Roga Andrei

Cerințele exercițiului

Când sistemul este alimentat de la portul USB și utilizatorul va apăsa pe buton (se va folosi intrerupere), sistemul va porni și se vor realiza următoarele acțiuni:

- va măsura temperatura din exteriorul serei și va compara cu valorile normale:
  - dacă valorile citite sunt normale atunci culoarea verde a LED-ului RGB de exterior se va aprinde
  - dacă valorile citite nu sunt normale atunci se va aprinde culoarea albastru a LED-ului RGB de exterior dacă este prea rece sau culoarea rosu dacă este prea cald
- va măsura temperatura din interiorul serei și va compara cu valorile normale:
  - dacă valorile citite sunt normale atunci culoarea rosie a LED-ul RGB de interior se va aprinde
  - dacă valorile citite nu sunt normale atunci se va aprinde culoarea albastru a LED-ului RGB de interior dacă este prea rece sau culoarea rosu dacă este prea cald, iar difuzorul va avea un sunet diferit pentru cele două situații

- când sistemul este pornit se va afişa in Serial Monitor:

- Temperatura si umiditatea exterioara: valoare
- Temperatura si umiditatea interioara: valoare

- cand sistemul este oprit se va afisa in Serial Monitor: OPRIT

- când se va apăsa a doua oară pe buton se va opri sistemul din monitorizare și se va aprinde

LED-ul roșu, iar in Serial Monitor se va afisa OPRIT.

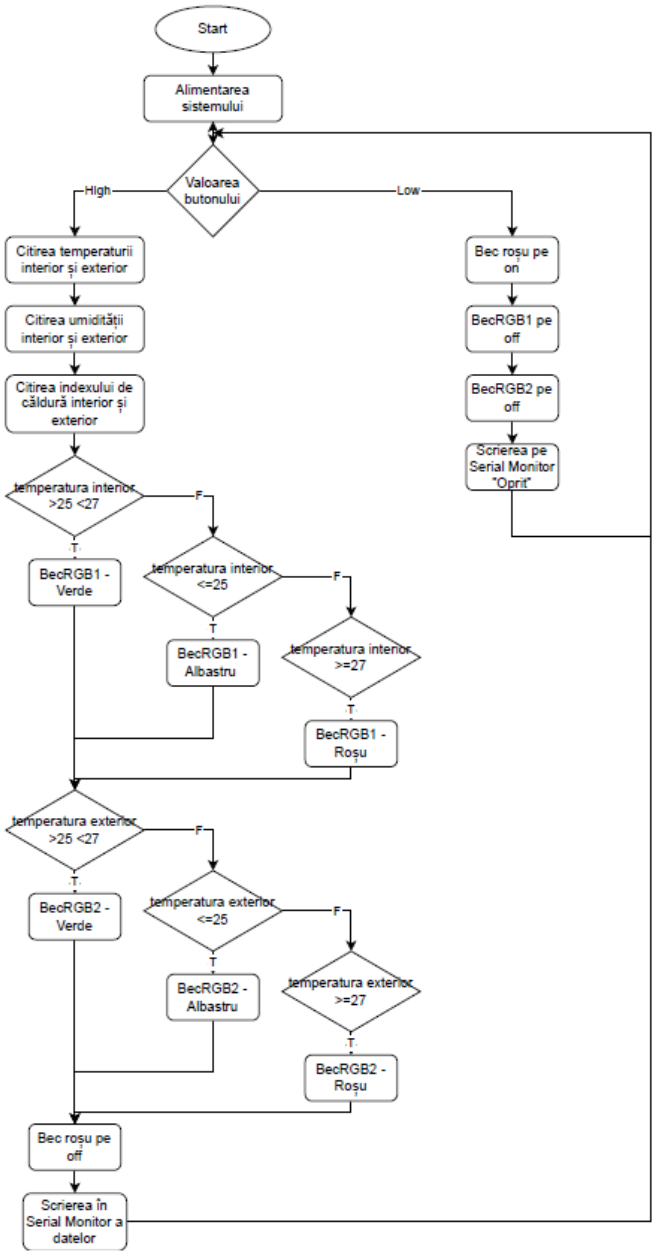
Utilitatea în lumea reală

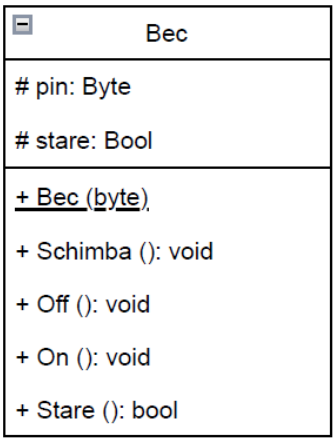
Automatizarea unei sere poate duce la o creștere semnificativă a producției de legume și fructe. Serele automate folosesc senzori și sisteme de control pentru a monitoriza și ajusta temperatura, umiditatea și lumina în seră, creând astfel condițiile ideale pentru creșterea plantelor. Aceasta poate duce la o producție mai mare și mai rapidă de legume și fructe, ceea ce poate reduce dependența de produsele importate și poate crește siguranța alimentară.

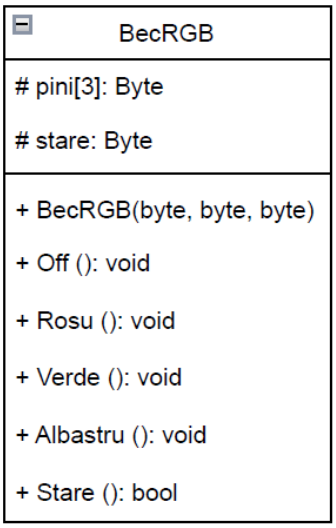
Componentele montajului

- Arduino Uno
- 2 LED-uri RGB (pentru exterior și interior)
- 2 Senzori de temperatură și umiditate DHT11
- 1 buton
- 1 LED roșu
- 1 rezistor 1kOhm
- 1 rezistor 10kOhm

Schema logică și diagrama de clase







Codul sursă în limbajul C++

```
// Includerea bibliotecii care operează cu senzorul DHT11
#include <DHT.h>

// Definirea pinilor analogici ai senzorilor DHT11
#define DHTpin A0
#define DHTpin2 A1

// Definirea unei enumerații care reprezintă stările unui led RGB
enum StareRGB {RosuRGB, VerdeRGB, AlbastruRGB, Oprit};

// Declararea clasei Bec
class Bec
{
  private:
    // Declararea variabilei pin
    byte pin;
    // Declararea variabilei stare
    bool stare = 0;

  public:
    // Declararea funcțiilor clasei Bec
    Bec(byte pin);
    void Schimba();
    void Off();
    void On();
    bool Stare();
};

// Definirea constructorului clasei Bec
Bec::Bec(byte pin) {
  // Setarea variabilei pin
  this->pin = pin;
  // Definirea modului pinului ca fiind OUTPUT
  pinMode(pin, OUTPUT);
}

// Definirea funcției Schimba a clasei Bec
void Bec::Schimba() {
  // Schimbă starea pe cea opusă
  this->stare = !this->stare;
  // In dependență de stare schimbă pinul ca fiind pe HIGH sau LOW
  if (this->stare)
    digitalWrite(this->pin, HIGH);
  else
    digitalWrite(this->pin, LOW);
```

```
}

// Definirea functiei Off a clasei Bec
void Bec::Off() {
  // În dependență de stare apelează funcția Schimba
  if (this->stare) {
    this->Schimba();
  }
}

// Definirea functiei Off a clasei Bec
void Bec::On() {
  if (!this->stare) {
    // În dependență de stare apelează funcția Schimbă
    this->Schimba();
  }
}

// Definirea funcției Stare a clasei Bec
bool Bec::Stare() {
  // Returnează variabila stare
  return this->stare;
}

// Declararea clasei BecRGB
class BecRGB
{
  private:
    // Declararea unui array pini
    byte pini[3];
    // Declararea variabilei stare
    byte stare = 0;

  public:
    // Declararea funcțiilor clasei BecRGB
    BecRGB(byte pinRosu, byte pinVerde, byte pinAlbastru);
    void Off();
    void Rosu();
    void Verde();
    void Albastru();
    bool Stare();
};

// Definirea constructorului clasei BecRGB
BecRGB::BecRGB(byte pinRosu, byte pinVerde, byte pinAlbastru) {
  // Setarea array-ului pini
  this->pini[RosuRGB] = pinRosu;
  this->pini[VerdeRGB] = pinVerde;
  this->pini[AlbastruRGB] = pinAlbastru;
```

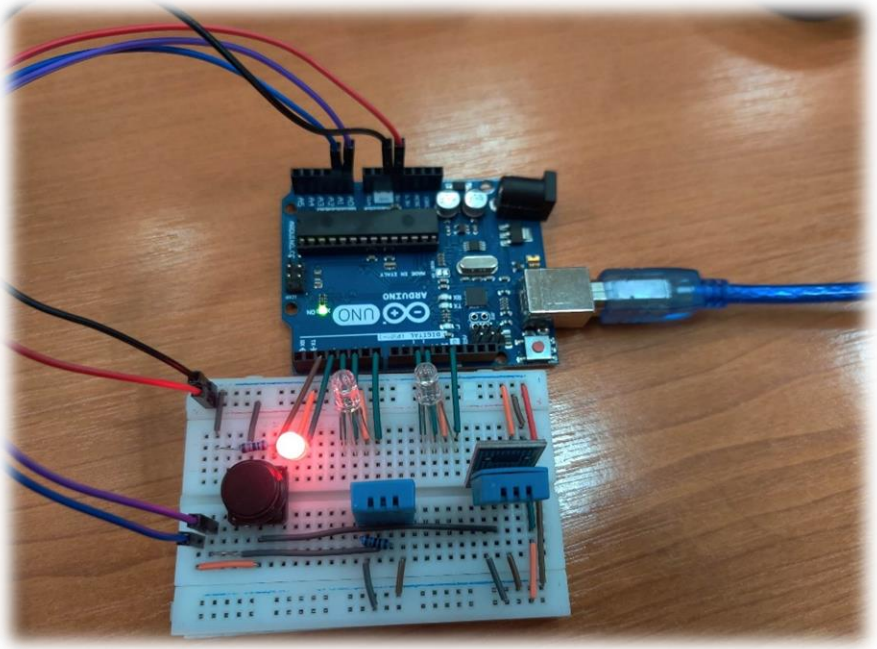
```
  // Setarea modului pinilor ca fiind de tip OUTPUT
  for(int i=0; i<3; i++)
    pinMode(this->pini[i], OUTPUT);
}

// Definirea funcției Off a clasei BecRGB
void BecRGB::Off() {
  // In dependență de stare, setează toți pinii pe 0
  if (this->stare != Oprit) {
    analogWrite(pini[RosuRGB], 0);
    analogWrite(pini[VerdeRGB], 0);
    analogWrite(pini[AlbastruRGB], 0);
    // Shimbă variabila stare pe 0
    this->stare = Oprit;
  }
}

// Definirea funcției Rosu a clasei BecRGB
void BecRGB::Rosu() {
  // In dependență de stare, setează pinul Rosu pe 255
  if (this->stare != RosuRGB) {
    analogWrite(pini[RosuRGB], 255);
    analogWrite(pini[VerdeRGB], 0);
    analogWrite(pini[AlbastruRGB], 0);
    // Shimbă variabila stare pe 1
    this->stare = RosuRGB;
  }
}

// Definirea funcției Rosu a clasei BecRGB
void BecRGB::Verde() {
  // In dependență de stare, setează pinul Verde pe 255
  if (this->stare != VerdeRGB) {
    analogWrite(pini[RosuRGB], 0);
    analogWrite(pini[VerdeRGB], 255);
    analogWrite(pini[AlbastruRGB], 0);
    // Shimbă variabila stare pe 2
    this->stare = VerdeRGB;
  }
}

// Definirea funcției Rosu a clasei BecRGB
void BecRGB::Albastru() {
  // In dependență de stare, setează pinul Verde pe 255
  if (this->stare != AlbastruRGB) {
    analogWrite(pini[RosuRGB], 0);
    analogWrite(pini[VerdeRGB], 0);
    analogWrite(pini[AlbastruRGB], 255);
    // Shimbă variabila stare pe 3
```

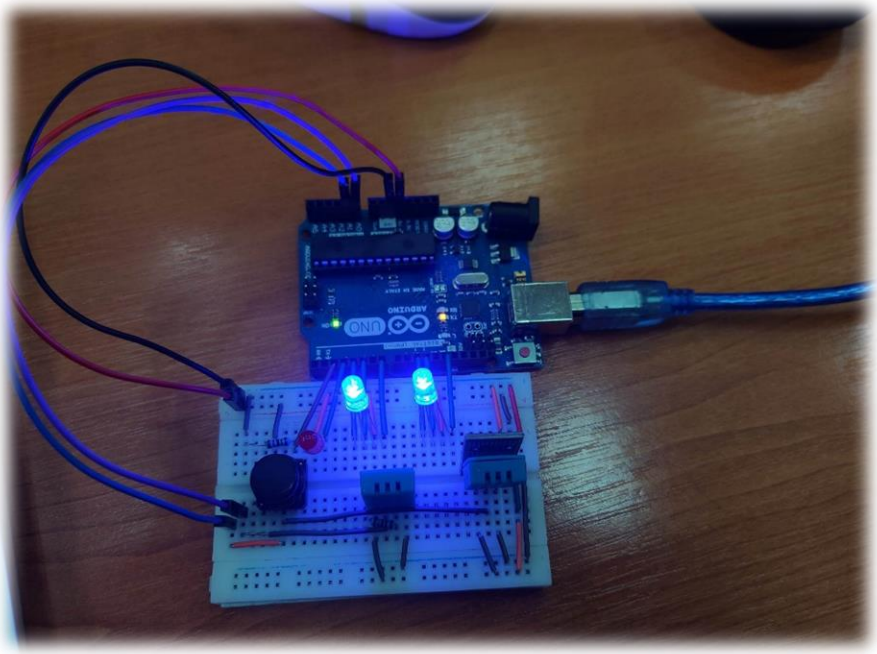
```
    this->stare = AlbastruRGB;
  }
}

// Definirea funcției Stare a clasei BecRGB
bool BecRGB::Stare() {
  // Returnează variabila stare
  return this->stare;
}

// Crearea variabile becRosu de tipul Bec
Bec becRosu(3);
// Crearea variabile becRGBInterior de tipul BecRGB
BecRGB becRGBInterior(7, 5, 4);
// Crearea variabile becRGBExterior de tipul BecRGB
BecRGB becRGBExterior(13, 11, 10);
// Crearea varibilei stare pentru intreruperea sistemului
volatile byte stare = LOW;
// Declararea varibilelor pentru senzorii DHT11
DHT dhtInterior(A0, DHT11);
DHT dhtExterior(A1, DHT11);
// Declararea variabilelor pentru memorarea datelor de la senzor
float TemperaturaInterior;
float TemperaturaExterior;
float UmiditateInterior;
float UmiditateExterior;
float IndexCalduraInterior;
float IndexCalduraExterior;
// Declarare variabilei change pentru a memora trecerea la valorea LOW a
întreruperii
int change = 0;
// Declarare variabilei change pentru a memora trecerea starea sistemului
int onoff = 0;

void setup() {
  // Declarearea întreruperii sistemului
  attachInterrupt(digitalPinToInterrupt(2), stop, CHANGE);
  // Declarearea Serial Monitor
  Serial.begin (115200);
  // Afișarea pe Serial Monitor ”Oprit!”
  Serial.println(F("Oprit!"));
  // Inițializarea senzorilor DHT11
  dhtInterior.begin();
  dhtExterior.begin();
}

// Definirea funției stop pentru întrerupere
void stop() {
  stare = !stare;
```

```
}

void loop() {
  // Dacă butonul e apăsat și anterior a fost LOW atunci se schimbă starea
sistemului
  if (stare == HIGH && change == 1) {
    onoff = !onoff;
    change = 0;
  }
  // Dacă valoarea e LOW se schimba valoarea change
  if (stare == LOW) {
    change = 1;
  }

  // Dacă starea sistemului este 1 atunci se îndeplinesc următoarele condiții
  if (onoff == 1) {
    // Se citesc valorile pe de senzori
    TemperaturaInterior = dhtInterior.readTemperature();
    UmiditateInterior = dhtInterior.readHumidity();
    IndexCalduraInterior = dhtInterior.computeHeatIndex(TemperaturaInterior,
UmiditateInterior, false);
    TemperaturaExterior = dhtExterior.readTemperature();
    UmiditateExterior = dhtExterior.readHumidity();
    IndexCalduraExterior = dhtExterior.computeHeatIndex(TemperaturaExterior,
UmiditateExterior, false);

    // Se scrie pe println într-un format pentru aplicația C#
    Serial.println((String) TemperaturaInterior + "T"
                   + UmiditateInterior + "H"
                   + IndexCalduraInterior + "I"
                   + TemperaturaExterior + "TT"
                   + UmiditateExterior + "HH"
                   + IndexCalduraExterior + "II");

    // Dacă temperatura din interior este între 25 și 27, atunci becul RGB se
schimbă pe verde
    if (TemperaturaInterior > 25 && TemperaturaInterior < 27)
      becRGBInterior.Verde();
    // Dacă temperatura din interior este mai mică sau egală cu 25, atunci
becul RGB se schimbă pe albastru
    else if (TemperaturaInterior <= 25)
      becRGBInterior.Albastru();
    // Dacă temperatura din interior este mai mare sau egală cu 27, atunci
becul RGB se schimbă pe roșu
    else if (TemperaturaInterior >= 27)
      becRGBInterior.Rosu();

    // Dacă temperatura din exterior este între 25 și 27, atunci becul RGB se
schimbă pe verde
```

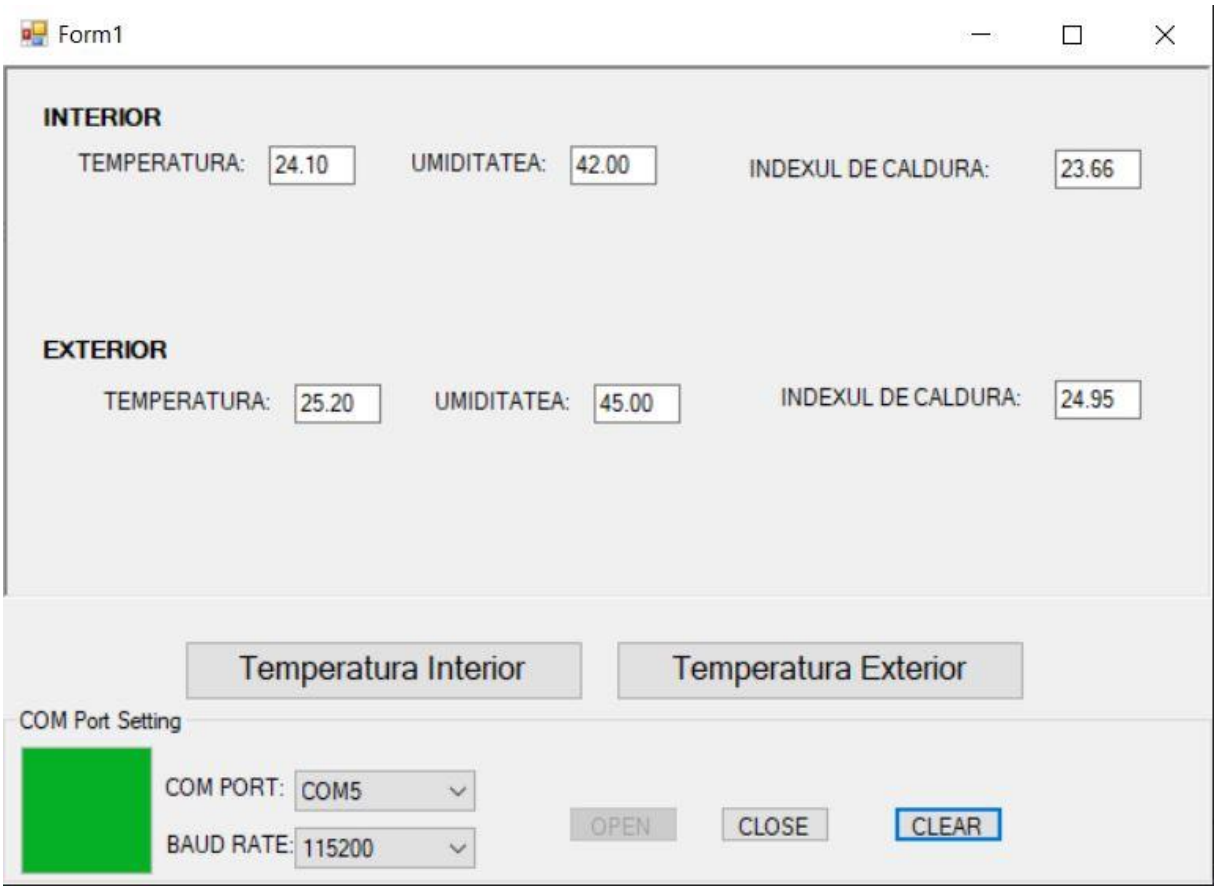
```
    if (TemperaturaExterior > 25 && TemperaturaExterior < 27)
      becRGBExterior.Verde();
    // Dacă temperatura din exterior este mai mică sau egală cu 25, atunci
becul RGB se schimbă pe albastru
    else if (TemperaturaExterior <= 25)
      becRGBExterior.Albastru();
    // Dacă temperatura din exterior este mai mare sau egală cu 27, atunci
becul RGB se schimbă pe roșu
    else if (TemperaturaExterior >= 27)
      becRGBExterior.Rosu();
    // Becul roșu se stinge
    becRosu.Off();
  }
  else {
    // Becurile RGB se sting
    becRGBInterior.Off();
    becRGBExterior.Off();
    // Becul roșu se aprinde
    becRosu.On();
  }
}
```

Fotografii din timpul laboratorului





Interfața din aplicația C# și explicații



La început se selecteza COM PORT.

La apăsarea butonului se introduc datele de pe senzor în C#.

La fiecare 2 secunde se modifica datele (temepretura, umiditate, indexul de caldură).

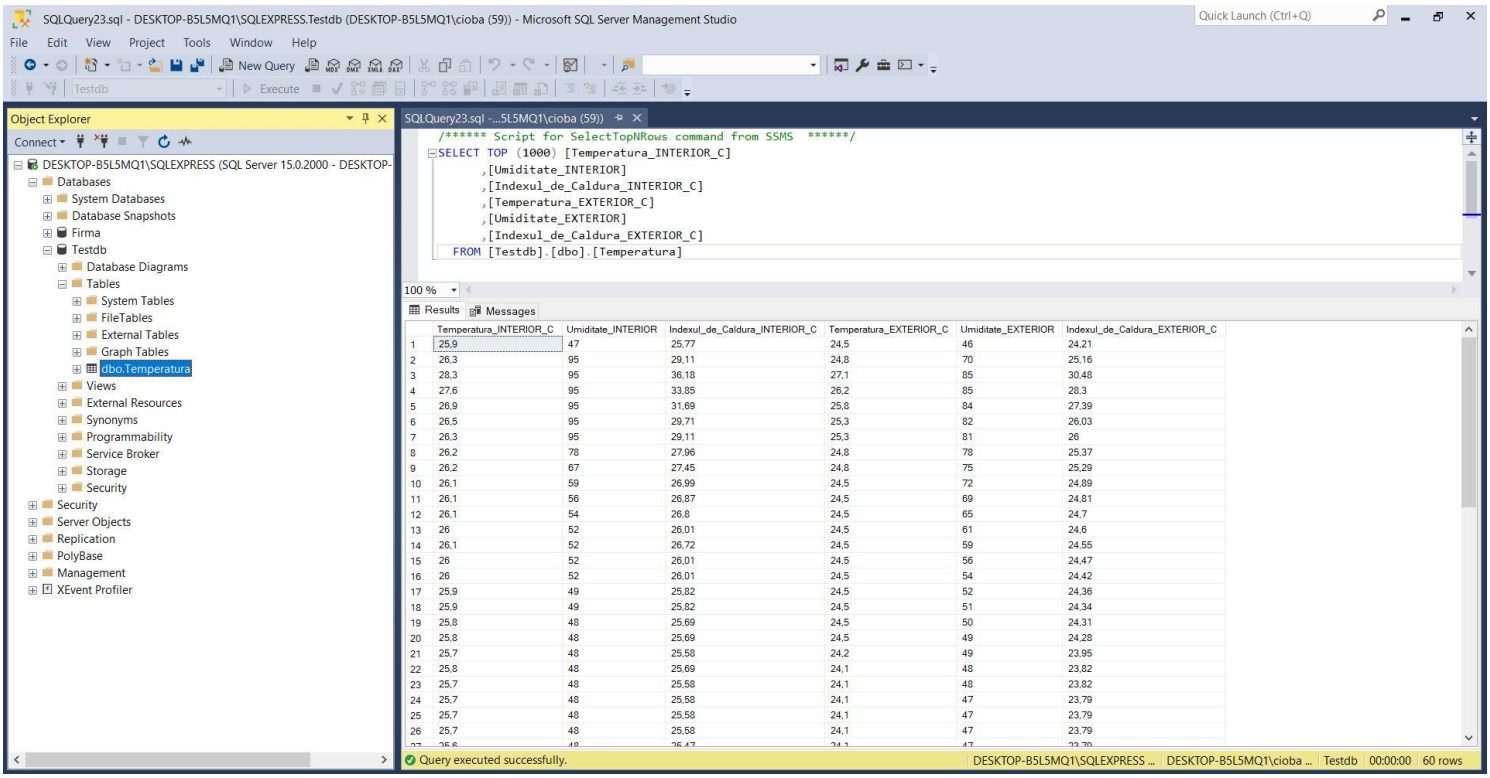
La fiecare 10 secunde se introduc în baza de date.

Când butonul îi apasat și e rosu datele sunt înca pe interfață și dupa clear se șterg.

Când datele sunt nule, nu se introduc în baza de date.

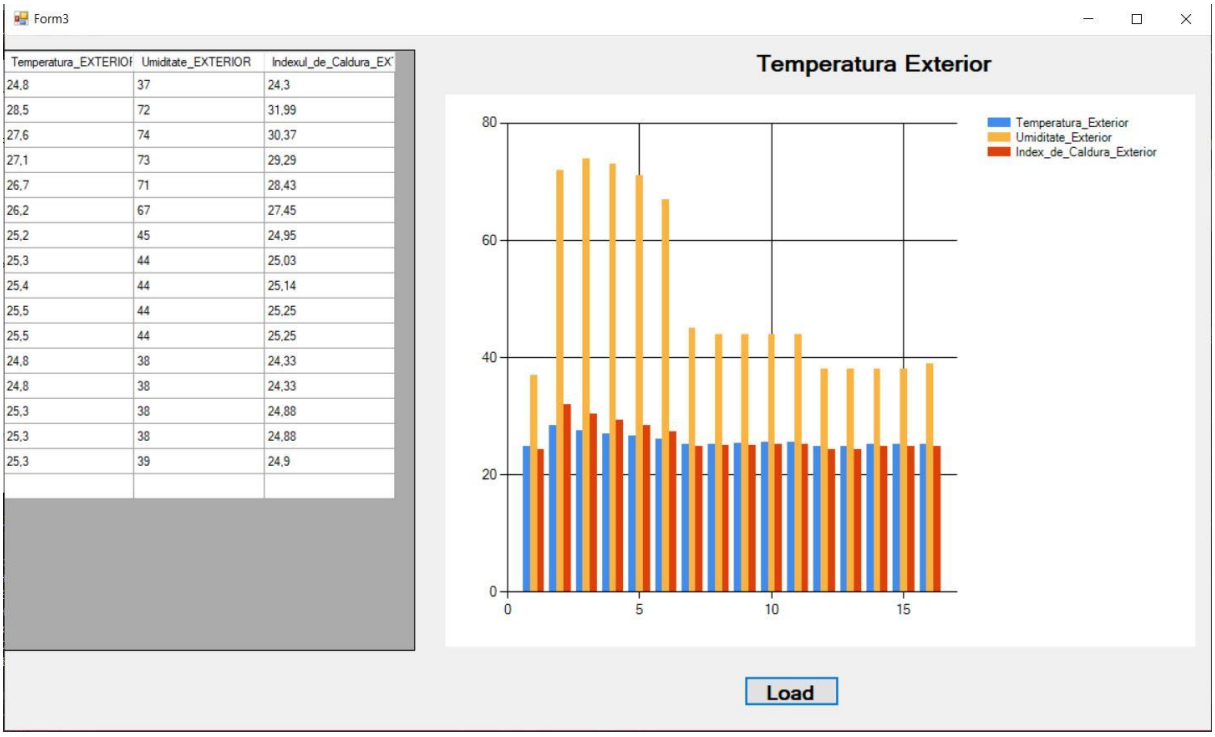
Dacă butonul îi apăsat din nou apar datele pe interfață.

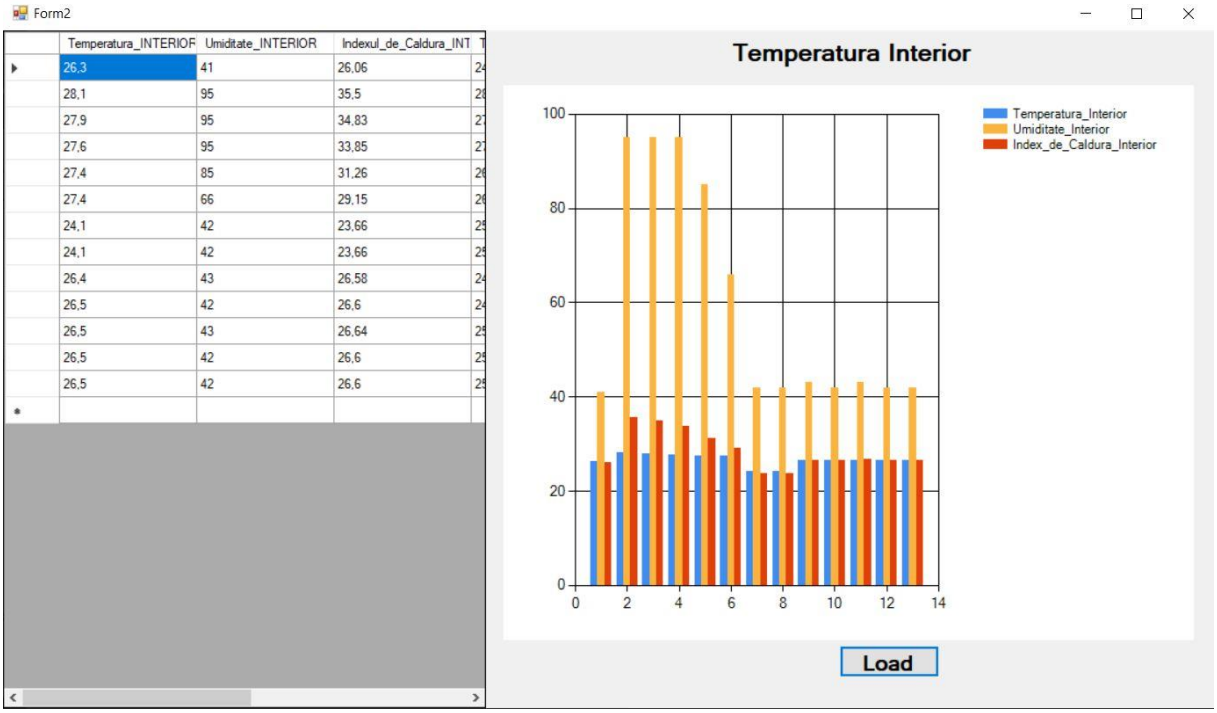
Capturi de ecran cu tabelele din baza de date



	Temperatura_INTERIOR_C	Umiditate_INTERIOR	Indexul_de_Caldura_INTERIOR_C	Temperatura_EXTERIOR_C	Umiditate_EXTERIOR	Indexul_de_Caldura_EXTERIOR_C
1	25,9	47	25,77	24,5	46	24,21
2	26,3	95	29,11	24,8	70	25,16
3	28,3	95	36,18	27,1	85	30,48
4	27,6	95	33,85	26,2	85	28,3
5	26,9	95	31,69	25,8	84	27,39
6	26,5	95	29,71	25,3	82	26,03
7	26,3	95	29,11	25,3	81	26
8	26,2	78	27,96	24,8	78	25,37
9	26,2	67	27,45	24,8	75	25,29
10	26,1	59	26,99	24,5	72	24,89
11	26,1	56	26,87	24,5	69	24,81
12	26,1	54	26,8	24,5	65	24,7
13	26	52	26,01	24,5	61	24,6
14	26,1	52	26,72	24,5	59	24,55
15	26	52	26,01	24,5	56	24,47
16	26	52	26,01	24,5	54	24,42
17	25,9	49	25,82	24,5	52	24,36
18	25,9	49	25,82	24,5	51	24,34
19	25,8	48	25,69	24,5	50	24,31
20	25,8	48	25,69	24,5	49	24,28
21	25,7	48	25,58	24,2	49	23,95
22	25,8	48	25,69	24,1	48	23,82
23	25,7	48	25,58	24,1	48	23,82
24	25,7	48	25,58	24,1	47	23,79
25	25,7	48	25,58	24,1	47	23,79
26	25,7	48	25,58	24,1	47	23,79

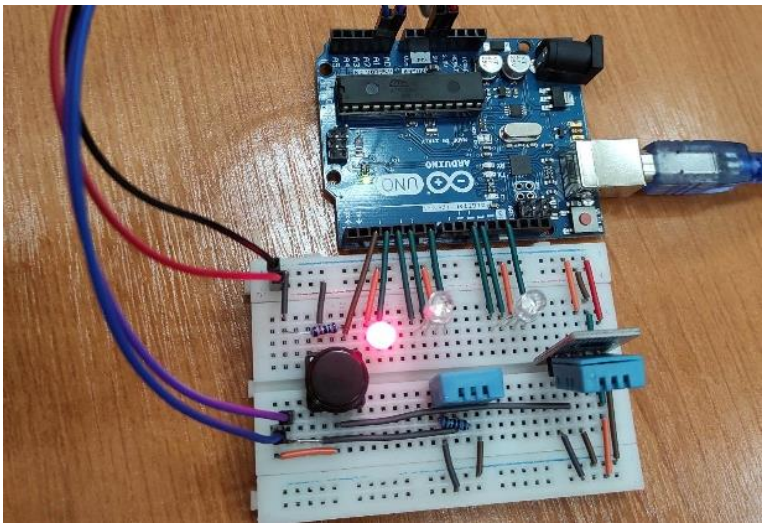
Grafice/rapoarte



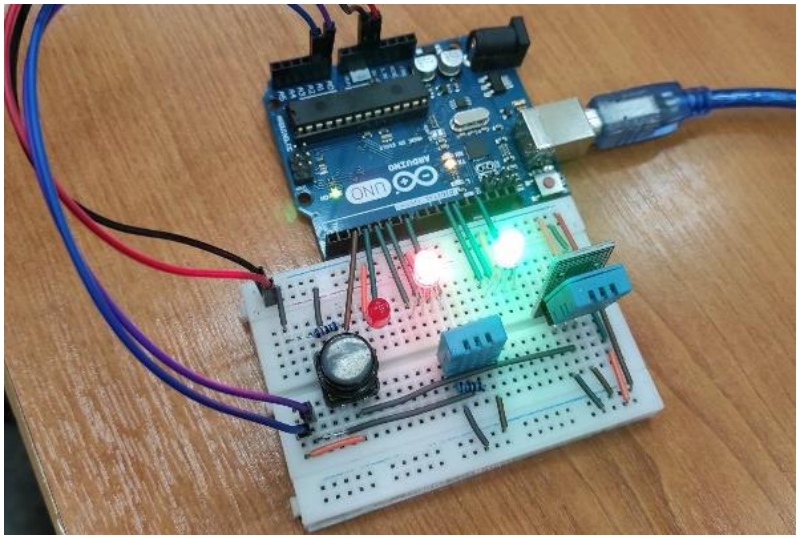


Plan de testare

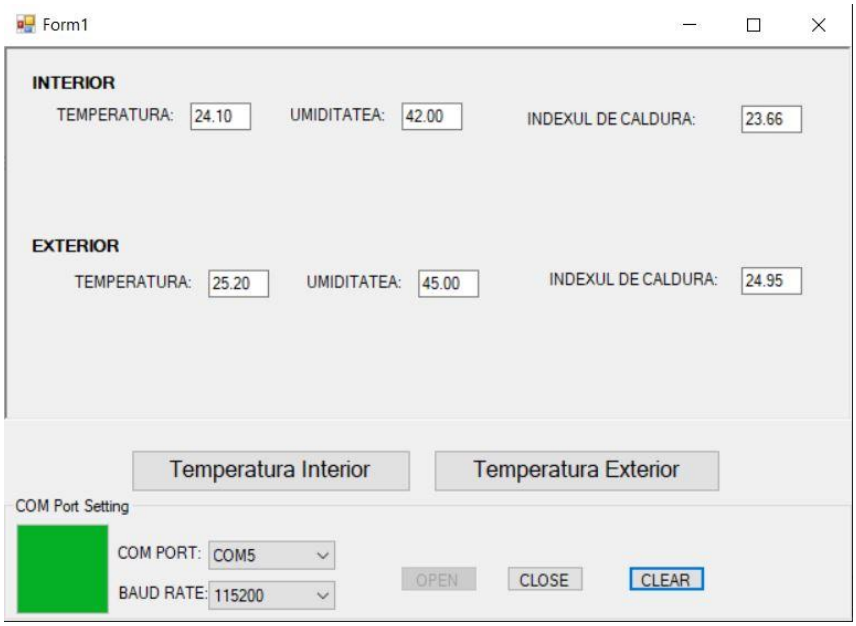
1) Alimentăm sistemul Arduino (se aprinde led-ul roșu)



2) La apăsarea butonului se stinge led-ul roșu, și senzorii DHT11 încep să măsoare temperatura și umididatea din aer



3) La fiecare 2 secunde valorile din arduino sunt transmise către C#.

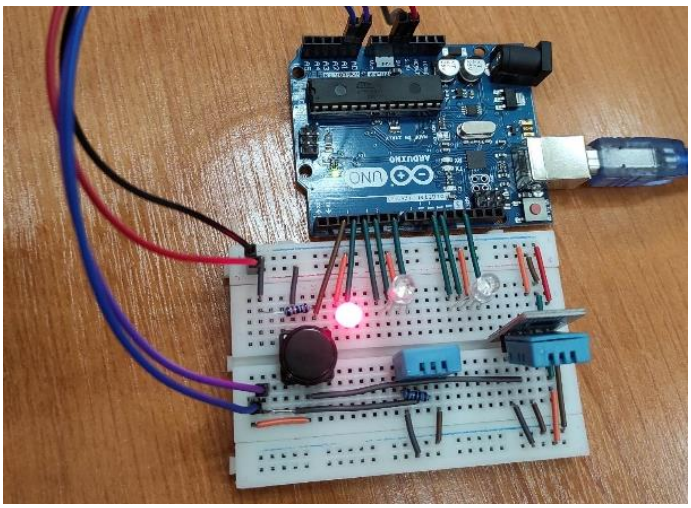


4) Din C# valorile sunt transime în baza de date SQL server la fiecare 10 secunde.

Results | Messages

	Temperatura_INTERIOR_C	Umiditate_INTERIOR	Indexul_de_Caldura_INTERIOR_C	Temperatura_EXTERIOR_C	Umiditate_EXTERIOR	Indexul_de_Caldura_EXTERIOR_C
10	25,5	95	26,83	24,5	55	24,44
11	25,5	73	26,01	24,5	52	24,36
12	25,5	61	25,7	24,5	49	24,28
13	25,6	53	25,6	24,5	47	24,23
14	25,6	49	25,49	24,5	45	24,18
15	25,6	45	25,39	24,5	44	24,15
16	25,6	43	25,34	24,5	43	24,13
17	25,7	41	25,4	24,5	42	24,1
18	25,6	40	25,26	24,5	41	24,08
19	25,7	39	25,34	24,5	40	24,05
20	25,8	38	25,43	24,5	39	24,02
21	25,8	38	25,43	24,8	39	24,35
22	25,8	38	25,43	24,8	39	24,35

5) La apăsarea din nou a butonului valorile nu mai sunt citite și nu mai sunt introduse în baza de date, led-urile se sting și se aprinde led-ul roșu.



Concluzii

Concluzia este că utilizarea a 2 senzori de temperatură și transmiterea datelor într-o bază de date folosind Arduino poate fi o soluție utilă și eficientă pentru monitorizarea temperaturii în diverse aplicații, cum ar fi controlul climatului, monitorizarea mediului ambiant sau monitorizarea temperaturii la distanță.

Transmiterea datelor într-o baza de date este utilă pentru monitorizarea temperaturii în timp real și pentru analiza datelor colectate într-un mod mai convenabil.