Indoor Garden System -- Buf Sorina Anamaria 331CA

Student: Buf Sorina Anamaria

Grupă: 331CA

Introducere

Obiectivul acestui proiect constă în realizarea unui sistem automat de îngrijire a plantelor, cu aplicații în viața cotidiană, înlocuind mediul tradițional al florilor de interior artizanale cu unul aproape independent de asistența umană.

Acest sistem are la bază mai multe metode care să asigure mentenanța plantelor, preluând date în timp real despre factorii din mediul exterior și acționând diferite mecanisme de reglare a valorilor citite. Se pot colecta date privind temperatura, umiditatea aerului, intensitatea luminii ambientale, precum si gradul de umiditate al solului, asigurând menținerea acestora într-un interval benefic și prosper cultivării plantelor de interior.

Descriere generală

Partea de input a proiectului constă în folosirea mai multor senzori, care citesc la intervale de timp bine determinate parametri de mediu definiți în secțiunea anterioară și, implicit, furnizează datele de control ale sistemului automatizat.

Astfel, detectarea unei temperaturi peste limitele admise declanșează acționarea unui ventilator, în scopul diminuării parțiale a temperaturii excesive, care poate dăuna plantei. De asemenea, un sistem de led-uri este utilizat pentru luminarea ambientală a grădiniței cu flori, declanșabil de zilele noroase, odată cu venirea serii sau de orice alt factor care duce la scăderea cantității de lumină naturală sau artificială primită.

Un aspect vital în îngrijirea oricărei plante îl consitutie udarea, aici acționată prin intermediul unei pompe de apă în momentul în care senzorul de umiditate al solului semnalează parametri de sol uscat.

Valorile înregistrate de senzori sunt interfațate utilizatorilor prin intermediul unui display LCD și actualizate la intervale regulate de timp.

Schema Bloc

Hardware Design

Piesele folosite în cadrul proiectului sunt:

- Arduino Uno R3 ATmega328P
- Breadboard
- Ecran LCD 1602
- LED-uri 5mm
- Ventilator 12V 80x80mm
- Pompă de apă 3-6V
- Releu
- Tranzistor NPN
- Senzor temperatură și umiditate DHT11
- Senzor umiditate sol
- Senzor intensitate lumină BH1750
- Rezistențe
- Battery holder
- Baterie 9V
- Fire dupont și jumper
- Barete de pini
- Condensator
- Diodă

Schema Electrica

Software Design

Organizare

În realizarea codului implementării, am folosit mai multe fișiere auxiliare pentru organizarea logicii în funcție de metodele create, astfel:

- **setup.cpp (setup.h)**

⇒ conține implementarea funcțiilor folosite în scopul inițializării parametrilor necesari circuitului.

```
/* Initializes digital pins. */
void setupDigital();

/* Initializes sensors. */
```

```
void setupSensors();

/* Initializez timer interrupts. */
void setupTimer();
```

- **lcd.cpp (lcd.h)**

⇒ conține metodele asociate afișărilor pe display-ul LCD, folosite de-a lungul implementării, pentru tranziția mesajelor, afișările principalilor parametri de mediu, precum și crearea unor simboluri folosind caractere custom, utilizate pentru vizualizarea grafică a declanșării componentelor de reglaj ale temperaturii, umidității solului și intensității luminoase: ventilator , pompă de apă , respectiv LED-uri .

```
/* Computes transition of two messages leaving the screen. */
void transitionLcd(String message1, String message2, int delayValue);

/* Computes display waiting effect while collecting data. */
void waitingEffect(String message, int delayValue);

/* Displays two messages on the two rows of the 1602 LCD. */
void displayLcd(String message1, String message2, int delayValue);

/* Displays initial LCD message when starting the circuit. */
void initialLcdMessage();

/* Displays sensors parameters. */
void displayCollectedData();

/* Displays custom character in the center of the LCD. */
void displaySymbol(byte symbol[][8]);

/* Displays custom sun character => marks LEDs lighting. */
void displaySun();

/* Displays custom wind character => marks FAN turning on. */
void displayWind();

/* Displays custom watering can character => marks PUMP watering the plant.
*/
void displayWater();
```

- **sensors.cpp (sensors.h)**

⇒ conține implementări folosite atât pentru citirea valorilor principalilor parametri înregistrați de senzorii prezenți în circuit, cât și pentru declanșarea funcționării componentelor de output, menționate anterior, în funcție de anumite condiții care vor fi detaliate ulterior.

```
/* Reads light intensity value and turns on the LEDs if needed. */
void readLight();

/* Reads temperature value and turns on fan if needed. */
```

```
void readTemperature();

/* Reads humidity value. */
void readHumidity();

/* Reads soil moisture value and waters the plant through the pump is needed.
*/
void readMoisture();
```

Logica principală a programului rămâne în fișierul Arduino dedicat setup-ului componentelor și variabilelor folosite, precum și menținării în stare de funcționare a circuitului: **plant_system.ino**.

Program Flow

Circuitul funcționează independent de asistența umană, odată cu setarea pe ON a battery holder-ului. Intervenția umană este necesară doar pentru reumplerea cu apă a recipientului folosit pentru udarea florii.

Odată pus în funcțiune, circuitul realizează următorii pași:

► **setup** ⇒ folosește fișierele și funcțiile definite anterior

- se inițializează senzorii, timer-ul și componentele de output.

```
setupDigital();
setupSensors();
setupTimer();
```

- se afișează mesaje de întâmpinare pentru utilizator.

```
initialLcdMessage();
```

- se începe colectarea datelor.

```
collectData();
```

► **loop** ⇒ la fel ca metoda de setup, folosește fișierele și metodele definite anterior

- folosind o variabilă de contorizare a timpului și o rutină de tratare de întreruperi, se generează colectarea de noi date la intervale regulate de aproximativ un minut, în perioadele dintre două înregistrări consecutive de date fiind afișate pe ecran valorile eșantionate din cadrul ultimei colectări: *Ligth Intensity*, *Temperature*, *Humidity* și *Soil Moisture*.

```
/* collects new data every minute */
if (timer >= COLLECT_DATA) {
        collectData();

        /* starts timer for measuring time between collecting data */
        timer = 0;
```

```
} else {
        displayCollectedData();
}
```

- în momentul colectării datelor, se verifică anumiți parametri limită, care pot declanșa funcționarea uneia dintre componentele de output.

Declanșare Ventilator

Ventilatorul este pornit în momentul în care, în urma colectării datelor, se constată că temepratura ambientală depășește temperatura optimă pentru o plantă de interior (~ 24°C).

Odată pornit, acesta va funcționa pentru un anumit interval de timp prestabilit (momentan ales de a fi ~30 minute). De asemenea, va exista o pauză obligatorie de timp între două porniri succesive, atât pentru a nu solicita componenta hardware, cât și pentru a nu conduce la un comportament dăunător plantei.

```
void readTemperature()
{
        /* reads temperature value from the sensor */
        temperature = dhtSensor.readTemperature();

        if (temperature > TEMPERATURE_HIGH) {
                /* first time the fan is turned on */
                if (firstFan) {
                        firstFan = false;
                        fanOn = true;

                        /* starts timer for measuring fan on */
                        fanTimer = 0;

                        /* turns on the fan */
                        digitalWrite(FAN, HIGH);

                        /* next times the fan is turned on */
                } else if (!fanOn) {
                        /* fan has been off for a period of time */
                        if (fanTimer >= FAN_OFF) {
                                fanOn = true;

                                /* starts timer for measuring fan on */
                                fanTimer = 0;

                                /* turns on the fan */
                                digitalWrite(FAN, HIGH);
                        }
                }
        }
}
```

```
void loop()
{
        /* fan has been on for a period of time */
        if (fanOn) {
                if (fanTimer >= FAN_ON) {
                        fanOn = false;

                        /* starts timer for measuring fan off */
                        fanTimer = 0;

                        /* turns off the fan */
                        digitalWrite(FAN, LOW);
                }
        }
}
```

Declanșare LED-uri

LED-urile sunt aprinse dacă lumina primită de senzor ajunge sub valoarea de 10lx. Acestea rămân aprinse până la următoarea eșantionare de date care determină o intensitate a luminii ambientale mai mare de 80lx.

În momentul aprinderii LED-urilor, este recalculată intensitatea luminii înainte de afișarea valorii pe ecran.

```
void readLight()
{
        /* reads light intensity from the sensor */
        lightIntensity = lightSensor.readLightLevel();

        /* light intensity is low*/
        if (lightIntensity < LIGHT_LOW) {
                /* turns on LEDs */
                digitalWrite(LED_1, HIGH);
                digitalWrite(LED_2, HIGH);

                /* rereads light intensity after adjusting light */
                lightIntensity = lightSensor.readLightLevel();

                /* light intensity is high */
        } else if (lightIntensity > LIGHT_HIGH) {
                /* turns off LEDs */
                digitalWrite(LED_1, LOW);
                digitalWrite(LED_2, LOW);
                delay(500);

                /* rereads light intensity after adjusting light */
                lightIntensity = lightSensor.readLightLevel();
```

```
        }
}
```

Declanșare Pompă de Apă

Udarea plantei prin intermediului furtunului și pompei de apă este declanșată în momentul în care senzorul de umiditate sol detectează parametri de sol uscat. Între două udări succesive există o perioadă de timp în care planta nu mai poate fi udată, indiferent de parametri eșantionați, astfel încât apa primită să poată fi absorbită în sol, iar noii parametri de umiditate să fie furnizați în mod corespunzător.

```
void readMoisture()
{
        /* reads soil moisture from the sensor */
        soilMoisture = analogRead(SOIL);

        /* soil is dry */
        if (soilMoisture > DRY_SOIL) {
                /* first time the pump is turned on */
                if (firstWater) {
                        firstWater = false;

                        /* waters plant */
                        digitalWrite(WATER, HIGH);
                        delay(2000);
                        digitalWrite(WATER, LOW);

                        /* starts timer for measuring pump off */
                        waterTimer = 0;

                        /* next times the pump is turned on => if the plant
has
                         * not been watered for a period of time
                         */
                } else if (waterTimer >= WATER_OFF) {
                        /* waters plant */
                        digitalWrite(WATER, HIGH);
                        delay(2000);
                        digitalWrite(WATER, LOW);


                        /* starts timer for measuring pump off */
                        waterTimer = 0;
                }
        }
}
```

Biblioteci

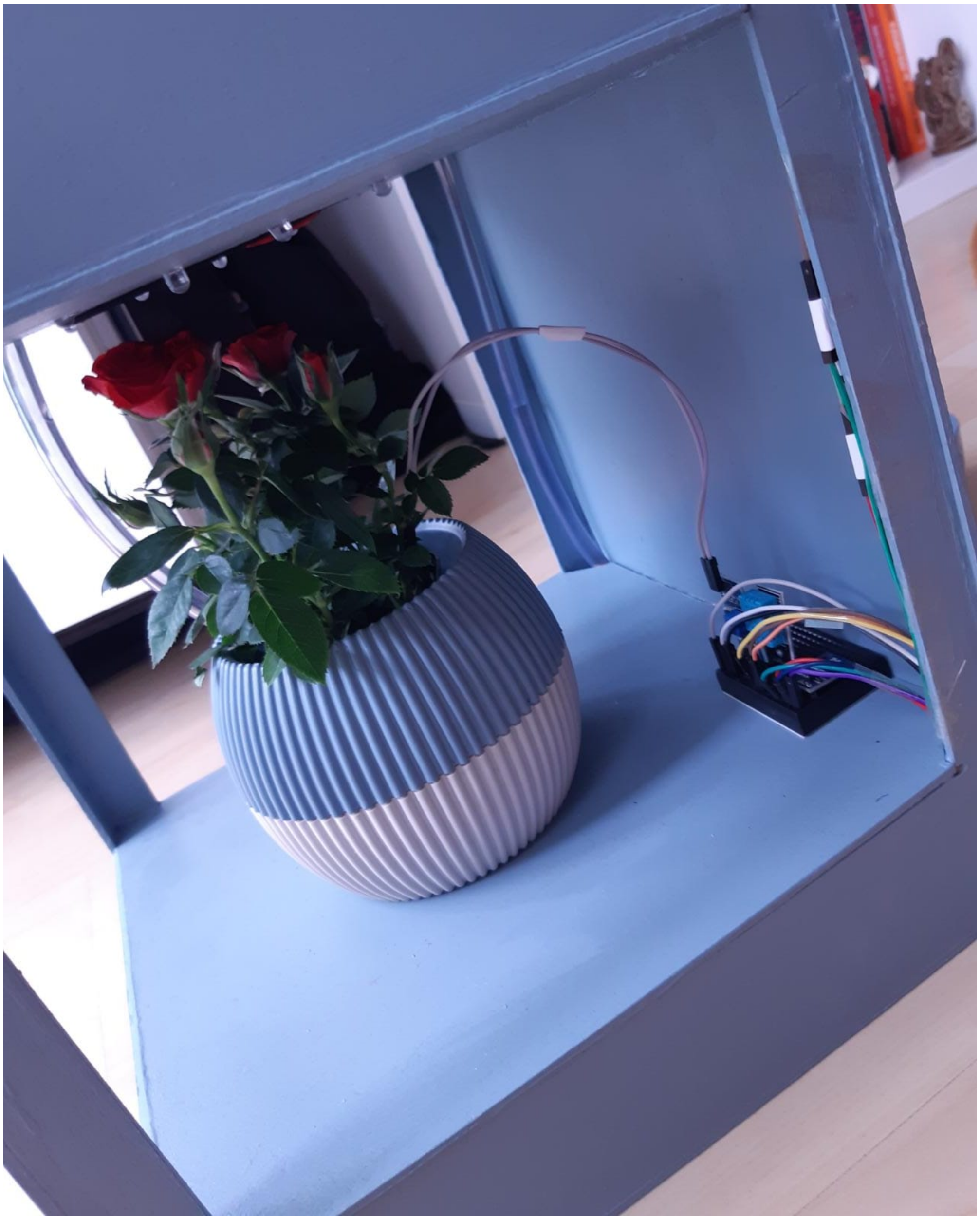
- *BH1750.h* ⇒ folosirea senzorului BH1750 pentru înregistrarea intensității luminii ambientale
- *DHT.h* ⇒ folosirea senzorului DHT11 pentru înregistrarea temperaturii și umidității
- *LiquidCrystal.h* ⇒ folosirea display-ului LCD 1602
- *Wire.h* ⇒ comunicarea cu dispozitivul I2C BH1750

Mediu de Dezvoltare

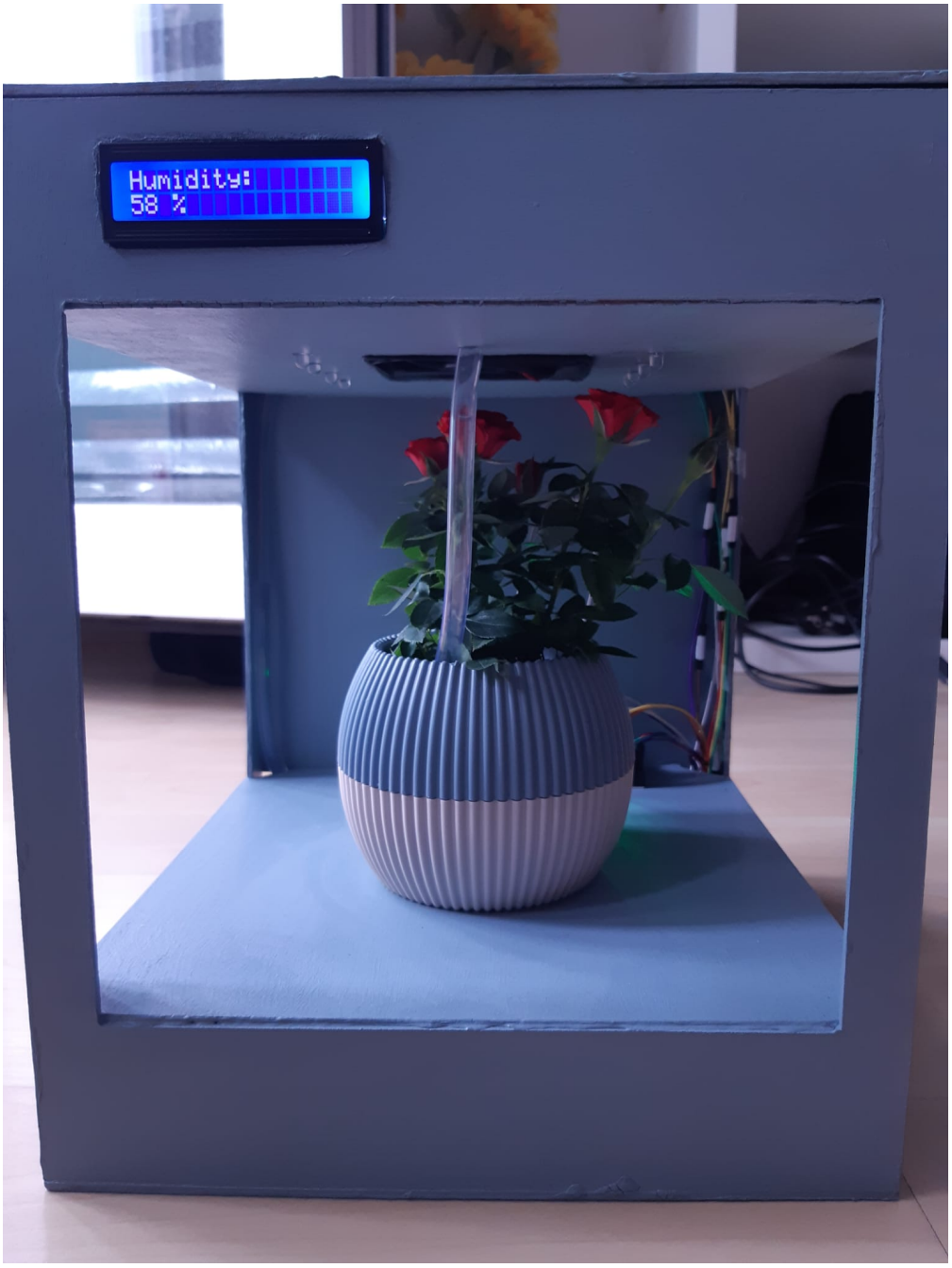
Mediul de dezvoltare folosit în realizarea proiectului: **Arduino IDE**.

Rezultate Obţinute

Proiectul Final



❁❁❁
❁❁❁❁❁



Demo

Mai multe poze și un scurt demo cu circuitul obținut se găsesc la următorul link:

- https://drive.google.com/drive/folders/1xVRN09i7DQV9JBrq3BgPt1eU9bdjHugu?usp=sharing

De menționat că pentru realizarea demo-ului au fost modificate valorile limită ale parametrilor pentru a putea genera aprinderea tuturor componentelor: LED-uri, ventilator, pompă de apă.

Concluzii

Proiectul a fost destul de fun de realizat, dar pe cât de drăguț și easy părea la început, pe atât mi-a dat de furcă partea hardware. Sunt fericită că am ajuns să înțeleg și să folosesc, într-un mod aprofundat și chiar independent de ajutor extern, diverse componente și noțiunile întreprinse din laboratoare, pentru a-mi crea propriul circuit. Chiar dacă am întâmpinat bug-uri pe parcurs, sunt mulțumită de implementarea obținută și faptul că este un proiect practic și utilizabil.

Am încercat să aduc o notă personală în modul de prezentare al circuitului final, atât prin caracterele custom create și afișate pe ecranul LCD, cât și prin realizarea fizică a mediului plantei (crearea design-ului cutiuței și vopsirea acesteia a fost cea mai fun parte) și manipularea parametrilor de mediu.

Pe viitor aș vrea să îmi extind implementarea, poate prin crearea unei aplicații sau adaugarea unor butoane care să permită interacțiunea într-un mod interactiv cu utilizatorii, astfel încât aceștia să își poată seta propriile preferințe legate de temperatură și lumină ambientală.

Download

Sursele proiectului pot fi vizualizate aici: plant_system.

Jurnal

- 10.05.2022 → Alegere tema proiect
- 15.05.2022 → Completare Milestone 1: Introducere, Descriere, Schema Bloc și Componente
- 29.05.2022 → Completare Milestone 2: Schema electrică, Software Design, Rezultatele Obținute, Concluzii, Arhivă

Bibliografie/Resurse

Resurse Hardware

- https://create.arduino.cc/projecthub/diyguyChris/arduino-indoor-garden-5d975c
- https://www.youtube.com/watch?v=Z0SZ-jzu_q8

Resurse Software

- https://lastminuteengineers.com/soil-moisture-sensor-arduino-tutorial/
- https://forum.arduino.cc/t/switching-a-cooling-fan-on-and-off-through-npn-transistor-connected-to-arduino/512176
- https://maxpromer.github.io/LCD-Character-Creator/