UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU
FACULTATEA DE INGINERIE
DEPARTAMENTUL DE CALCULATOARE ŞI INGINERIE ELECTRICĂ

PROIECT DE DIPLOMĂ

Conducător științific : Prof. Dr. Ing. Florea Adrian

Absolvent: Buta Andra-Paraschiva
Specializarea: Calculatoare

- Sibiu, 2023 –

UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU
FACULTATEA DE INGINERIE
DEPARTAMENTUL DE CALCULATOARE ŞI INGINERIE ELECTRICĂ

SISTEM PENTRU IRIGAREA PLANTELOR

Conducător științific : Prof. Dr. Ing. Florea Adrian

Absolvent: Buta Andra-Paraschiva
Specializarea: Calculatoare

CUPRINS

SISTEM PENTRU IRIGAREA PLANTELOR .....

CUPRINS .....

1. INTRODUCERE 5

1.1 Scopul lucrării 5

1.2 Obiective 5

1.3 Descrierea problemei 6

1.4 Motivație 6

1.5 Structura lucrării 7

2. STUDIUL ACTUAL AL CERCETĂRILOR IN DOMENIU 8

2.1 Soluții existente 8

3. NOȚIUNI TEORETICE 10

3.1 Medii de dezvlotare 10

3.1.1 Visual Studio 10

3.1.2 Arduino IDE 11

3.2 Aplicatie Web 12

3.2.1 ASP.NET Core MVC 12

3.3 Baze de date 14

3.4 MQTT 15

4. DESCRIEREA LUCRĂRII 18

4.1 Procesul de irigare (Logica fuzzy) 19

4.2 Arhitectura Hardware 24

4.2.1 Collector 25

4.2.2 Controller 32

4.3 Interconectarea componentelor 40

4.3.1 Transferul de date (MQTT) 40

4.3.2 Baza de date 41

4.4 Arhitectura Software 43

4.4.1 Collector 43

4.4.2 Controller 46

4.4.3 Aplicația Web 48

5. REZULTATE PRODUSE 56

6. CONCLUZII SI DEZVOLTĂRI ULTERIOARE 59

6.1 Concluzii 59

6.2 Limitări 59

6.3 Dezvoltări ulterioare 59

REFERINȚE 60

1. INTRODUCERE

1.1 Scopul lucrării

Această lucrare își propune să creeze un sistem cu scopul de a automatiza procesul de irigare, folosind tehnologia IoT (Internet of Things) pentru irigarea unei grădini. Sistemul este format din două părți: hardware și software. Obiectivele acestuia sunt de a monitoriza în timp real viață plantelor, astfel diverși parametrii vor fi preluați prin intermediul senzorilor și de a controla procesul de irigare.

Prin intermediul aplicației web se vor putea urmări datele despre mediu preluate de către senzori, și de pe siteul "AccuWeather". Irigarea poate fi controlată atât manual cât și automat. Manual prin activare din aplicație de către utilizatori și automat deciziile fiind luate de către sistem în urmă unor calcule bazate pe datele despre mediu.

1.2 Obiective

Obiectivele acestei lucrări sunt:

- Evaluarea condițiilor necesare pentru implementarea unui mod adecvat de irigare bazat pe datele mediului înconjurător și necesitatea plantelor.
- Realizarea unor dispozitive embedded de colectare a datelor din mediul de viață al plantelor.
- Realizare unor dispozitive embedded de control al procesului de irigare.
- Implementarea unei căi de comunicare dintre componenta hardware si cea software prin protocolul MQTT (Message Queue Telemetry Transport).
- Realizarea unei aplicații web pentru a facilita interacțiunea dintre om si sistem.
- Colectarea datelor și stocarea acestora într-o bază de date.

1.3 Descrierea problemei

Problema actuală este lipsa unui sistem practic și eficient de irigare ceea ce duce la apariția mai multor disfuncționalități precum irigarea incorectă a plantelor, utilizarea în mod nesustenabil a resureslor de apă precum și cele de energie.

Irigarea corectă a plantelor poate duce atât la o viață mai lungă a acestora cât și la dezvoltarea la potențialul maxim prin oferirea unui mediu de viață propice dezvoltări corespunzătoare.

Utilizarea sustenabilă a resurselor de apă este importantă deoarece aceste resurse nu sunt infinite, iar datoria noastră este de a ne proteja și păstra resursele de care dispunem. De asemenea risipa de apă poate fi un element în defavoarea plantelor care pot fi afectate de excesul de apă, de aceea este necesar să ținem cont de necesitatea acestora și să le oferim cât mai precis cantitatea de care acestea au nevoie.

De asemene o irigare prea îndelungată duce și la un consum mai mare de energie care de asemenea este o problema pe care o putem evita.

1.4 Motivație

Lucrarea a fost dezvoltată în scop personal pentru implementarea unui mod mai practic și mai eficient de irigare a zonei verzi din curtea personală. Inițial irigarea se făcea manual. Considerând această metodă ineficientă din mai multe puncte de vedere am decis schimbarea acesteia. Factori negativi cauzați de vechea metodă erau: necesitatea timpului aproape zilnic pentru irigare, cantitatea de apă neaproximată corect astfel fiind una neadecvată pentru plante ceea ce ducea atât la un consum de apă și de energie mai mare cât și la afectarea plantelor, nerespectarea timpului la care era necesară irigarea, condițiile meteo nefavorabile când se făcea irigarea. Toate acestea duceau la o irigarea neadecvată care putea provoca distrugerea plantelor. Am dezvoltat un sistem de irigare care poate fi controlat manual și automat. Acesta analizează solul și condițiile meteo astfel determinând durata optimă de irigare, cantitatea optimă de apă și este simplu de utilizat de către oricine are acces la un dispozitv cu internet folosind aplicația web.

1.5 Structura lucrării

În continuare, vom prezenta capitolul 2, în care vom aborda informații extrase din diverse cercetări în domeniu. Capitolul 3 va descrie noțiunile teoretice utilizate în dezvoltarea proiectului, în timp ce capitolul 4 va detalia fiecare componentă în parte. În capitolul 5, vom prezenta rezultatele obținute, în timp ce capitolul 6 va descrie dezvoltările ulterioare și referințele bibliografice utilizate în proiect.

2. STUDIUL ACTUAL AL CERCETĂRILOR IN DOMENIU

2.1 Soluții existente

Această lucrare propune implementarea unui sistem de monitorizare bazat pe Internet of Things (IoT), care este atât eficient, cât și economic, pentru supravegherea permanentă a unei grădini. Obiectivele principale includ optimizarea și îmbunătățirea procesului de irigare, ceea ce ar conduce la creșterea productivității și îmbunătățirea calității în agricultură.

Internetul lucrurilor (IoT) este o idee fascinantă și complexă care a devenit realitate datorită progresului tehnologic. În cadrul acestei evoluții, sistemul de irigații nu face excepție, beneficiind de posibilitățile pe care tehnologia IoT le oferă.

În articolul lor intitulat "IoT-based smart homes: A review of system architecture, software, communications, privacy and security"[1], autorii Dragos Mocrii, Yuxiang Chen și Petr Musilek analizează principalele tehnologii ale caselor inteligente bazate pe IoT. Ei descriu o casă inteligentă ca fiind o aplicație a calculului ubicuitar care încorporează inteligența în administrarea și operarea caselor pentru confort, siguranță, securitate și eficiență energetică.

Prin această analogie, putem vedea cum aceste principii pot fi aplicate într-un sistem de irigații bazat pe IoT. În cadrul acestui sistem, funcționalitatea autonomă devine secundară, în timp ce conectivitatea și deciziile bazate pe date preiau prim-planul. Prin conectarea și comunicarea dispozitivelor între ele, sistemul de irigații devine mult mai util și eficient.

Așa cum se întâmplă în cazul caselor inteligente, un sistem de irigații bazat pe IoT poate aduce numeroase avantaje. Între acestea, eficiența și convenienta se află în prim-plan. Un astfel de sistem optimizează procesele de irigare prin automatizarea lor, oferind o intervenție umană minimă. Automatizarea este conștientă de context și adaptabilă, ceea ce înseamnă că irigarea se realizează la momentul potrivit și în modul corespunzător.

Pe lângă acestea, un sistem de irigații bazat pe IoT poate contribui la gestionarea eficientă a resurselor de apă, o problemă majoră în agricultura modernă. În plus, personalizarea și adaptabilitatea

sunt elemente cheie ale unui astfel de sistem, permițând ajustarea irigării în funcție de nevoile specifice ale fiecărei culturi.

Așa cum tehnologia IoT a îmbunătățit calitatea vieții în casele noastre, la fel poate face și pentru sistemul de irigații, aducând un plus de eficiență, convenienă și sustenabilitate în acest domeniu.

3. NOȚIUNI TEORETICE

3.1 Medii de dezvlotare

Mediile de dezvoltare utilizate în realizarea aplicației sunt Visual Studio 2019 pentru aplicația web fiind dezvoltată o aplicație de tipul ASP.NET Core Web App (Model-View-Controller) în limbajul de programare C# și Arduino IDE pentru controlul microcontrollerelor find utilizat limbajul de programare C++.

3.1.1 Visual Studio

Visual Studio este un mediu de dezvoltare creat de către Microsoft. Este folosit pentru a dezvolta programe de calculator, cum ar fi aplicații web, site-uri web, servicii web și aplicații mobile. Visual Studio folosește platformele de dezvoltare software Microsoft, cum ar fi Windows Forms, Windows API, Windows Presentation Foundation, Microsoft Silverlight și Windows Store. Poate genera atât cod nativ, cât și cod gestionat.

Visual Studio conține un editor de cod. Depanatorul încorporat funcționează atât ca depanator la nivel de mașină, cât și la nivel de sursă. Alte instrumente încorporate includ un profiler de cod, un designer de aplicații GUI, un designer web, un designer de clasă, un designer de schemă de baze de date și multe altele. Aceasta include suport pentru sistemele de control al sursei (cum ar fi Subversion și Git), noi seturi de instrumente, cum ar fi editori și designeri vizuali pentru limbaje specifice domeniului, sau adăugarea de seturi de instrumente pentru alte aspecte ale dezvoltării software, la aproape fiecare nivel.

Visual Studio acceptă 36 de limbaje de programare diferite, iar editorii de cod și depanatoarele pot suporta aproape orice limbaj de programare (în grade diferite), atâta timp cât sunt disponibile servicii specifice limbii. Limbile încorporate includ C, C++,Visual Basic .NET, F#, C#, JavaScript, TypeScript, XML, XSLT, HTML, CSS. Suportul pentru alte limbi, cum ar fi Python, Ruby, Node.js și M este disponibil prin pluginuri. Java și J# au fost acceptate anterior.

Cea mai simplă ediție a Visual Studio, Community Edition, este disponibilă gratuit. Sloganul pentru Visual Studio Community Edition este „IDE gratuit și complet pentru studenți, open source și

dezvoltatori individuali". Începând cu 10 ianuarie 2023, Visual Studio 2022 este o versiune gata de producție. Visual Studio 2013, 2015 și 2017 au suport extins, iar 2019 are suport general. [2]

3.1.2 Arduino IDE

IDE provine de la "Mediu de dezvoltare integrat" și este un software introdus de către Arduino, care este utilizat pentru editarea, compilarea și încărcarea codului pe dispozitivele Arduino. Apropae toate modulele Arduino sunt compatibile cu acest program care este open surce și este disponibil pentru instalare și compilare în orice moment.

Arduino IDE este un software oficial Arduino, care face compilarea codului ușoară, încât chiar și o persoană obișnuită, fără cunoștințe tehnice anterioare, poate începe procesul de învățare.

Este disponibil ușor pentru sisteme de operare precum MAC, Windows, Linux și rulează pe platforma Java care vine cu funcții și comenzi încorporate care joacă un rol vital pentru depanarea, editarea și compilarea codului în mediu.

O gamă de module Arduino disponibile, inclusiv Arduino Uno, Arduino Leonardo, Arduino Mega, Arduino Micro și multe altele. Fiecare dintre ele conține un microcontroller pe placa care este de fapt programat și acceptă informația sub formă de cod.

Codul principal, cunoscut și sub numele de schiță, creat pe platforma IDE va genera în cele din urmă un fișier Hex care este apoi transferat și încărcat în controller-ul de pe placa.

Mediu IDE conține în principal două părți de bază: Editor și Compilator, unde primul este folosit pentru scrierea codului necesar și al doilea este folosit pentru compilarea și încărcarea codului în Modulul Arduino dat. Acest mediu suportă atât limbajele C, cât și C ++.[3]

3.2 Aplicatie Web

Conform Software Quality o aplicație web este un program de aplicație stocat pe un server la distanță și servit pe Internet printr-o interfață de browser. Serviciile web sunt prin definiție aplicații web și multe, dacă nu toate, site-urile web conțin aplicații web.

Dezvoltatorii proiectează aplicații web pentru o varietate de utilizări și utilizatori, de la organizații la persoane fizice, dintr-o varietate de motive. Aplicațiile web utilizate în mod obișnuit includ poștă web, calculatoare online și magazine de comerț electronic. Unele aplicații web sunt accesibile numai printr-un anumit browser, dar majoritatea aplicațiilor web sunt independente de browser.

Aplicațiile web sunt accesate prin rețea și nu trebuie descărcate. Utilizatorii pot accesa aplicațiile web prin browsere web precum Google Chrome, Mozilla Firefox și Safari.

O aplicație web funcțională necesită un server web, un server de aplicații și o bază de date. Serverele web procesează cererile de la clienți, iar serverele de aplicații îndeplinesc sarcinile solicitate. Toate informațiile necesare sunt stocate în baza de date. Aplicațiile web au de obicei cicluri scurte de dezvoltare și echipe mici de dezvoltare. Dezvoltatorii creează majoritatea aplicațiilor web în JavaScript, HTML5 sau CSS. Programarea la nivelul clientului utilizează de obicei aceste limbaje pentru a ajuta la construirea front-end-ului aplicației dvs. Programarea pe partea serverului implică scrierea de scripturi care sunt utilizate de aplicațiile web. Limbaje precum Python, Java și Ruby sunt utilizate în mod obișnuit pentru programarea pe server.[4]

3.2.1 ASP.NET Core MVC

ASP.NET Core MVC este un framework pentru construirea de aplicații web și API-uri folosind modelul de design Model-View-Controller.

Model-View-Controller (MVC) este un model arhitectural care împarte aplicația în 3 grupuri principale: Models, Views si Controllers. Utilizând acest model, cererile utilizatorilor sunt direcționate

către un Controller care este responsabil pentru lucrul cu Modelul. Controller-ul alege View-ul care se va afișa și îi furnizează toate datele de model de care are nevoie.

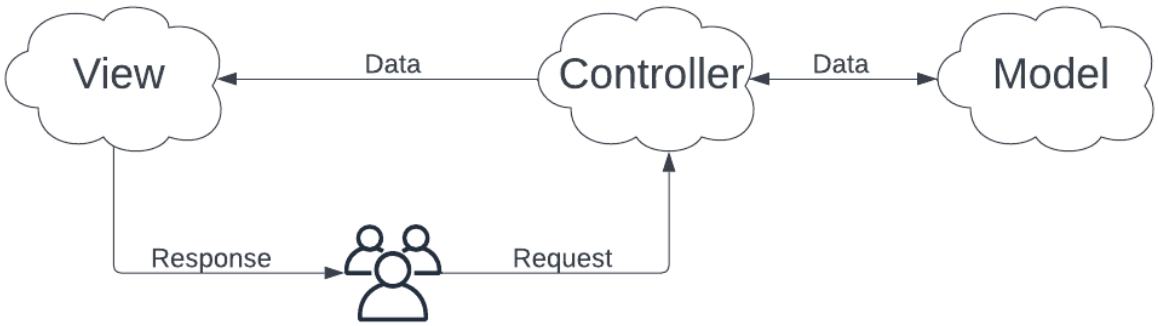

Figura 1-Modelul Arhitectural Model-View-Controller

Componenta Model se ocupă de toată logica legată de date cu care interacționăm. Aceasta poate reprezenta fie date transmise între componentele de vizualizare și controller, fie alte date. De exemplu, un obiect client preia informații despre client dintr-o bază de date, manipulează și actualizează datele din baza de date și le folosește pentru a reda date.

Componenta View este folosită pentru toată logica interfeței cu utilizatorul a aplicației. De exemplu, vizualizarea Client va include toate componentele necesare interfeței, cum ar fi casetele de text, meniurile etc. cu care interacționează utilizatorul final.

Controller-ul este componenta care se ocupă cu interacțiunea utilizatorului, lucrează cu modele și selectează pagina de vizualizare ca răspuns pentru utilizator. Într-o aplicație MVC, controller-ul gestionează și răspunde la intrarea și interacțiunea utilizatorului. În modelul MVC, controller-ul este punctul inițial și este responsabil pentru selectarea tipurilor de model cu care să lucreze și cu ce vizualizare să redea. [5]

3.3 Baze de date

O bază de date este o colecție organizată de informații structurate care este stocată sistematic, astfel încât să poată fi accesată, gestionată și actualizată eficient. Scopul său este de a stoca și organiza în mod consecvent datele astfel încât să poată fi utilizate în funcție de nevoi.

Bazele de date stochează informații structurate, orice tip de informații care trebuie stocate și gestionate eficient. Acestea pot fi utilizate într-o varietate de domenii, inclusiv aplicații web, sisteme de gestionare a bazelor de date, sisteme bancare, magazine online și multe altele.

Bazele de date vă permit să organizați datele în tabele, să le legați între ele, să le interogați pentru a prelua informații specifice și să efectuați analize. Structura și relațiile datelor dintr-o bază de date sunt definite folosind un sistem de management al bazei de date (DBMS) precum MySQL, Oracle, Microsoft SQL Server sau PostgreSQL.

Avantajele utilizării bazelor de date includ gestionarea eficientă a datelor, capacitatea de a rula interogări și rapoarte complexe, de a asigura securitatea datelor, de a gestiona accesul simultan la date și abilitatea de a face copii de rezervă și de a restaura datele în cazul unei defecțiuni a sistemului.

Există mai multe tipuri de baze de date, astfel în funcție de necesități se poate alege cea mai eficientă.

Baze de date:

- Relaționale: sunt bazele de date organizate în tabele cu rânduri și coloane. Acestea oferă acces la cele mai organizate și structurate date;
- Orientate pe obiecte: sunt bazele de date care oferă informații sub formă de obiecte;
- NoSQL: sunt bazele de date nerelaționale care facilitează atât stocarea cât și gestionarea datelor nestructurate în comparație cu bazele de date relaționale;
- Grafice: sunt bazele de date în care datele se stochează sub forma unor entități;
- Open source: sunt bazele de date care au codul sursă de tip open source;
- In cloud: sunt bazele de date care se află pe o platformă în cloud publică, privată sau hibridă;

- Multi-model: sunt bazele de date care combină mai multe modele de baze de date;
- Pentru documente/JSON: sunt bazele de date care gestionează datele sub forma de documente;
- Autonome: sunt cele mai noi tipuri de baze de date, acestea utilizând învățarea automată pentru realizarea atribuțiilor administratorilor bazei de date. [6]

3.4 MQTT

MQTT este cel mai des folosit protocol de mesagerie în Internetul lucrurilor (IoT). MQTT înseamnă MQ Telemetry Transport. Acest protocol este un set de reguli care definesc modul în care dispozitivele IoT publică și se abonează la date pe Internet. MQTT este utilizat pentru mesageria și schimbul de date între dispozitivele IoT și Industrial IoT (IIoT), cum ar fi dispozitivele încorporate, senzorii și PLC-urile industriale. Protocolul este bazat pe evenimente și utilizează un model de publicare/abonare (pub/sub) pentru a conecta dispozitive. Emițătorii (editorii) și receptorii (abonații) comunică prin subiecte și sunt izolați unul de celălalt. Conexiunile dintre ele sunt gestionate de brokerii MQTT. Brokerul MQTT filtrează toate mesajele primite și le livrează corect abonaților.

Atât editorii, cât și abonații pot trimite și primi mesaje fără a se întrerupe reciproc. De exemplu, abonații nu trebuie să aștepte ca editorii să trimită mesaje.

Protocolul MQTT este alcătuit din trei elemente cheie: brokerul, clientul și conexiunile.

Un client MQTT este reprezentat de orice dispozitiv care utilizează o bibliotecă MQTT. Un editor este un client care trimite date și acționează ca un emițător, pe când un abonat este un client care primește date și acționează ca un receptor. Orice dispozitiv care trimite sau primește date folosind protocolul MQTT este considerat un client MQTT.

Brokerul este sistemul care se ocupă cu coordonarea mesajelor. Responsabilitățile principale ale acestuia sunt de a primi și filtra mesajele, de a identifica clienții abonați la fiecare mesaj și de a trimite mesajul către fiecare în parte.

Clienții comunică cu brokerul prin utilizarea unei conexiuni MQTT. Clienții inițiază conexiunea prin trimiterea unui mesaj CONNECT către broker. Brokerul confirmă că a fost stabilită o

conexiune prin trimiterea unui mesaj CONNACK. Pentru comunicare, clienții și brokerul vor utiliza o stivă TCP/IP. Clienții se conectează doar cu brokerul, niciodată între ei.

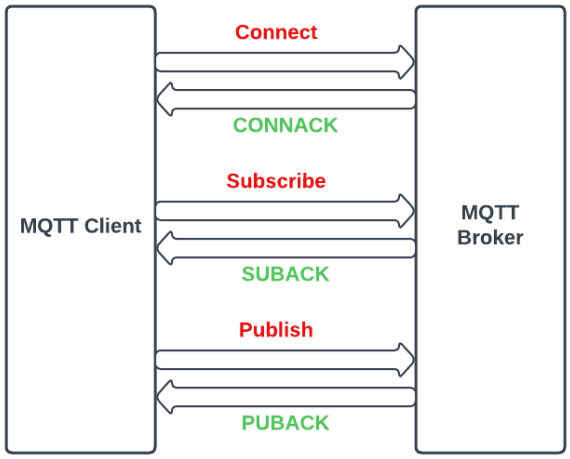

Figura 2-Fluxul de mesaje MQTT

În cadrul protocolului MQTT, termenii "topic", "publish" și "subscribe" sunt esențiali în comunicarea între broker și clienți.

"Topicul" sau subiectul este un canal sau un loc virtual unde clienții pot trimite sau primi mesaje. Acesta este un șir care definește subiectul mesajului și organizează informațiile într-o ierarhie tematică. De exemplu, un subiect ar putea fi „/collector/data” sau „ /valve/state”. Clienții pot comunica între ei abonându-se și postând mesaje pe anumite subiecte.

"Publish" sau publicare este o acțiune a clientului care trimite un mesaj unui broker pe un anumit subiect. Clientul care publică mesajul devine editorul. Mesajele pot conține date, instrucțiuni sau alte informații relevante. Brokerul primește mesaje și le distribuie tuturor clienților abonați la acest subiect.

"Subscribe" sau abonare este acțiunea unui client care se abonează la un anumit subiect pentru a primi știri publicate pe acel subiect. Clienții care s-au abonat vor fi abonați. Când un editor trimite un mesaj într-un subiect la care un client este abonat, brokerul trimite mesajul clientului respectiv.

Protocolul MQTT permite comunicarea flexibilă și eficientă între dispozitivele conectate prin utilizarea conceptelor de subiect, publicare și abonare. Clienții pot posta mesaje relevante pe subiecte adecvate, iar alți clienți se pot abona la aceste subiecte pentru a primi și procesa informațiile de care au nevoie. Acest model de publicare/abonare este una dintre principalele caracteristici ale protocolului MQTT.[7][8]

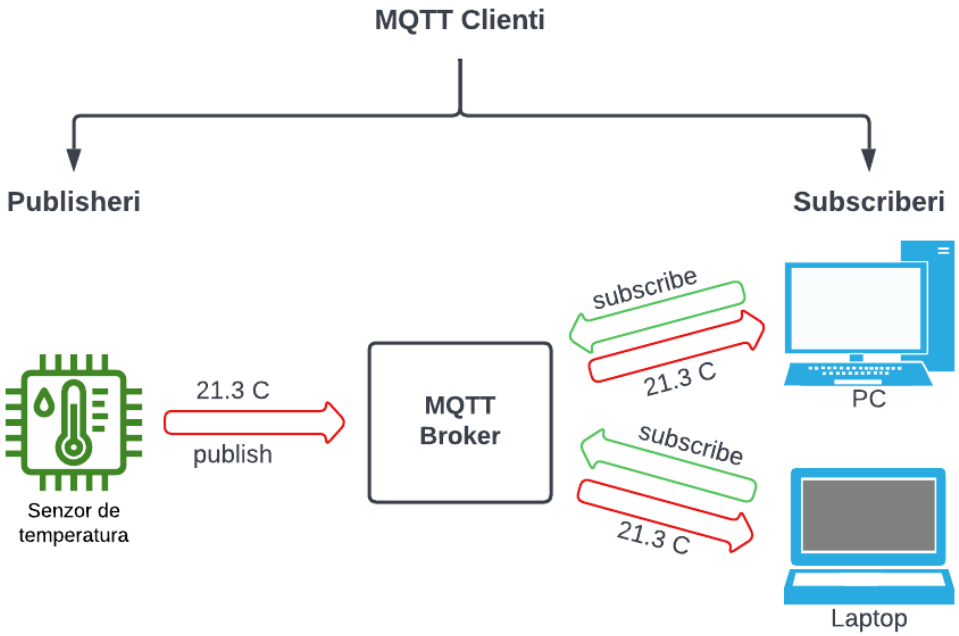

Figura 3-Exemplu arhitectură MQTT publish/subscribe

4. DESCRIEREA LUCRĂRII

Grădina în care va fi amplasat sistemul se află în zona rurală Apoldu de Jos, județul Sibiu, România. Pentru testare, am folosit o suprafață cu plante decorative, aproximativ 90 la număr, plasate pe conturul grădinii. Am amplasat un tub de irigare prin picurare care trece pe la rădăcinile plantelor. Debitul tubului este controlat de electrovalva care pornește de la sursa de apă.

Tehnica folosită era una ineficientă din mai multe puncte de vedere, scopul sistemului fiind de a furniza un mod mai simplu și mai eficient de irigare a plantelor. Sistemul actual integrează mai multe moduri de a fi controlat pentru a-i oferi utilizatorului cât mai multă libertate în procesul de irigare, dar totodată îi oferă și posibilitatea de a eficientiza acest proces. Sistemul are 3 moduri de funcționare precum: control manual, control programat și control automat.

În Figura 4 se poate vedea grădina, zona marcată în chenar roșu fiind zona pe care se va aplica sistemul pentru testare.

În Figura 5 este prezentat sistemul de tuburi cu picurători care este amplast la radacinile plantelor.

Figura 4-Gradina din Apoldu de Jos, Judetul Sibiu

Figura 5-Sistem de tuburi cu picurători

4.1 Procesul de irigare (Logica fuzzy)

Aplicația web permite utilizatorului să interacționeze cu sistemul în timp real. Aceasta oferă diverse funcționalități precum vizualizarea datelor despre mediu, dar și interacțiunea cu sistemul. Astfel, prin aplicație, utilizatorul are posibilitatea de a interacționa în diferite moduri cu sistemul: control manual, control automat și control programat. Controlul manual este o funcție care permite activarea și dezactivarea valvei în orice moment. Controlul programat oferă posibilitatea de a programa un moment în viitor la care să pornească și să se oprească irigarea. Controlul automat este o funcție mai "inteligentă" care permite sistemului să ia singur decizia când să fie irigate plantele în funcție de condițiile mediului. Modul automat este implementat printr-un sistem de logică fuzzy.

Logica fuzzy, cunoscută și sub denumirea de logica neclară, este un tip de logică care merge dincolo de simpla adevărat sau fals pe care o observăm în logică binară tradițională. A fost dezvoltată de Dr. Lotfi Zadeh în anii 1960 ca o metodă de a modela incertitudinea în lumea reală.

În loc să lucreze cu adevăr absolut sau falsitate, logica fuzzy lucrează cu grade de adevăr. Aceasta înseamnă că o propoziție poate fi "parțial adevărată" sau "parțial falsă". Acest lucru este foarte util atunci când modelăm sisteme complexe în care intrările și ieșirile nu sunt pur binare.

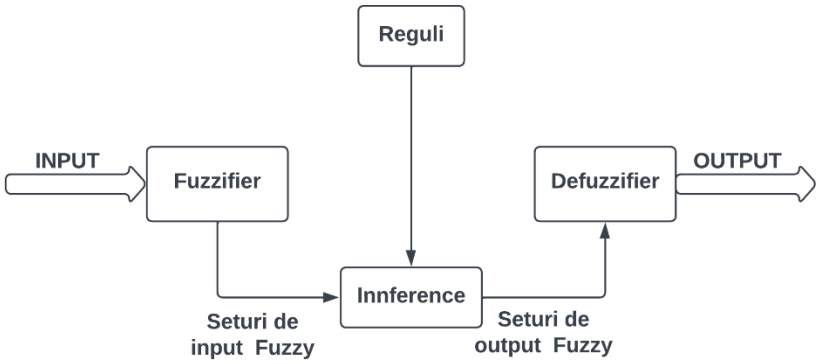

Figura 6-Structura unui sistem logic fuzzy

Un sistem de logică fuzzy funcționează în trei pași esențiali: fuzzificarea, inferența fuzzy și defuzzificarea.

Fuzzificarea este procesul prin care un sistem fuzzy transformă datele de intrare, care sunt numerice și precise, în informații fuzzy. În acest proces, se utilizează funcții de apartenență care măsoară gradul în care datele de intrare se încadrează în diferite seturi fuzzy. De exemplu, să presupunem că avem un senzor de temperatură care produce valori numerice și vrem să evaluăm dacă temperatura este "rece", "caldă" sau "foarte caldă". Funcțiile de apartenență ne-ar putea ajuta să transformăm temperatura numerică în aceste categorii fuzzy.

Odată ce am transformat datele de intrare în seturi fuzzy, următorul pas este de a aplica reguli fuzzy pentru a determina ce acțiune trebuie luată. Aceste reguli sunt de obicei formulate într-un limbaj uman natural și implică adesea cuvinte vagi sau neclare, cum ar fi "dacă temperatura este caldă, atunci

porniți aerul condiționat". Sistemul fuzzy aplică aceste reguli la seturile fuzzy și generează o concluzie fuzzy.

Defuzzificarea este ultimul pas si are ca scop de a transforma concluziile fuzzy înapoi în valori numerice precise pe care sistemul le poate utiliza pentru a acționa. Acest proces este cunoscut sub numele de defuzzificare. Există mai multe metode de defuzzificare, dar unul dintre cele mai comune este metoda centrului de greutate, care ia concluzia fuzzy, aplică o funcție de apartenență inversă și generează un număr precis.

Aceste trei etape alcătuiesc ceea ce este cunoscut sub numele de sistem fuzzy. Sistemul fuzzy este capabil să modeleze incertitudinea și ambiguitatea într-un mod care este foarte apropiat de modul în care oamenii fac decizii în situații neclare.[9]

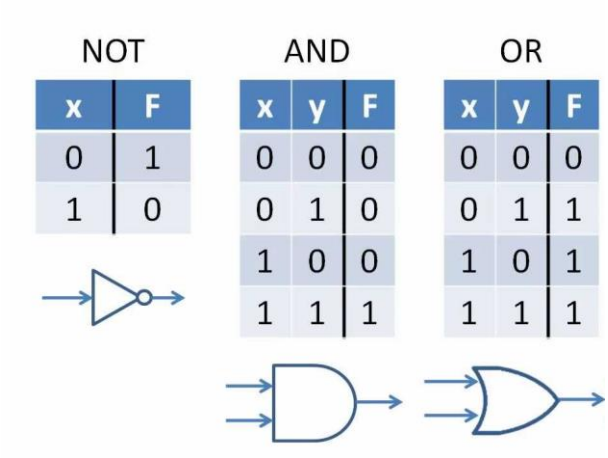

Figura 7-Relația dintre operatorii OR, AND și NOT

În lucrarea am folosit un sistem logic fuzzy cu 3 variabile de input (umiditatea sol = soil_moisture, temperatura = air_temperature, probabilitate de a ploua = rain_probability) și o variabilă de output (durata de irigare = irrigation_time).

Pentru variabilele de input soil_moisture si air_temperature am definit cate un set de 3 valori fuzzy si pentru variabila de input rain_probability am definit un set de 2 valori, iar pentru fiecare cate o funcție de apartenenta.

Setul fuzzy pentru umiditatea solului este compus din { DRY, MODERATE, WET }, iar valorile reale sunt exprimate în procente, într-un interval real de la 0 la 100. Pentru categoriile DRY și WET, s-au utilizat funcții de apartenență trapezoidale, în timp ce pentru MODERATE s-a folosit o funcție de apartenență triunghiulară.

Setul fuzzy pentru temperatura aerului este compus din { COLD, MODERATE, HOT }, iar valorile reale sunt exprimate în grade Celsisu, într-un interval real de la -30 la 50. Pentru categoriile COLD și HOT, s-au utilizat funcții de apartenență trapezoidale, în timp ce pentru MODERATE s-a folosit o funcție de apartenență triunghiulară.

Setul fuzzy pentru probabilitatea de ploaie este format din { NO, YES}, iar valorile reale sunt reprezentate ca un interval binar {0, 1}. Pentru ambele categorii s-au utilizat funcții de apartenență de tip singleton.

Setul fuzzy pentru variabila de ieșire (durata de irigare) este compus din 5 elemente { NONE, SHORT, MEDIUM, LONG, VERY_LONG }, iar valorile reale sunt exprimate în minute, într-un interval real cuprins între 0 și 10. Pentru categoria VERY_LONG, s-a utilizat o funcție de apartenență trapezoidală, în timp ce pentru celelalte categorii s-au folosit funcții de apartenență triunghiulare.

Numărul de reguli necesar într-un sistem fuzzy depinde de numărul de variabile de intrare și de numărul de seturi fuzzy pe care le ai pentru fiecare variabilă. Umiditatea solului are 3 seturi fuzzy: {DRY, MODERATE, WET}, temperatura aerului are 3 seturi fuzzy: {COLD, MODERATE, HOT}, probabilitatea de a ploua are 2 seturi fuzzy: {NO, YES}. Numărul total de reguli posibile va fi 3 (pentru umiditate) * 2 (pentru ploaie) * 3 (pentru temperatură) = 18 reguli.

1	IF (soil_moisture IS dry) AND (rain_probability IS no) AND (air_temperature IS cold) THEN (irrigation_time IS long).

2	IF (soil_moisture IS dry) AND (rain_probability IS no) AND (air_temperature IS moderate) THEN (irrigation_time IS very_long).
3	IF (soil_moisture IS dry) AND (rain_probability IS no) AND (air_temperature IS hot) THEN (irrigation_time IS very_long).
4	IF (soil_moisture IS dry) AND (rain_probability IS yes) AND (air_temperature IS cold) THEN (irrigation_time IS medium).
5	IF (soil_moisture IS dry) AND (rain_probability IS yes) AND (air_temperature IS moderate) THEN (irrigation_time IS long).
6	IF (soil_moisture IS dry) AND (rain_probability IS yes) AND (air_temperature IS hot) THEN (irrigation_time IS long).
7	IF (soil_moisture IS moderate) AND (rain_probability IS no) AND (air_temperature IS cold) THEN (irrigation_time IS short).
8	IF (soil_moisture IS moderate) AND (rain_probability IS no) AND (air_temperature IS moderate) THEN (irrigation_time IS medium).
9	IF (soil_moisture IS moderate) AND (rain_probability IS no) AND (air_temperature IS hot) THEN (irrigation_time IS long).
10	IF (soil_moisture IS moderate) AND (rain_probability IS yes) AND (air_temperature IS cold) THEN (irrigation_time IS none).
11	IF (soil_moisture IS moderate) AND (rain_probability IS yes) AND (air_temperature IS moderate) THEN (irrigation_time IS short).
12	IF (soil_moisture IS moderate) AND (rain_probability IS yes) AND (air_temperature IS hot) THEN (irrigation_time IS medium).

13	IF (soil_moisture IS wet) AND (rain_probability IS no) AND (air_temperature IS cold) THEN (irrigation_time IS none).
14	IF (soil_moisture IS wet) AND (rain_probability IS no) AND (air_temperature IS moderate) THEN (irrigation_time IS none).
15	IF (soil_moisture IS wet) AND (rain_probability IS no) AND (air_temperature IS hot) THEN (irrigation_time IS none).
16	IF (soil_moisture IS wet) AND (rain_probability IS yes) AND (air_temperature IS cold) THEN (irrigation_time IS none).
17	IF (soil_moisture IS wet) AND (rain_probability IS yes) AND (air_temperature IS moderate) THEN (irrigation_time IS none).
18	IF (soil_moisture IS wet) AND (rain_probability IS yes) AND (air_temperature IS hot) THEN (irrigation_time IS none).

Figura 8-Tabel cu regulile sistemului fuzzy

4.2 Arhitectura Hardware

Hardware-ul reprezinta partea fizica a unui sistem informatic și este format din totalitatea componentelor elctronice, electrice și mecanice care au ca scop permiterea execuției programelor software.

Partea hardware din aceasta lucrare este formata din două compnente separate cu scopuri bine definite: collectorul si controllerul.

4.2.1 Collector

Collectorul este componenta hardware care are rolul de a citi datele de la sol mai precis umiditatea solului, de a le pregăti și trimite mai departe prin protocolul MQTT la aplicația web care le va prelucra mai departe. În figura următoare este prezentată structura collectorului.

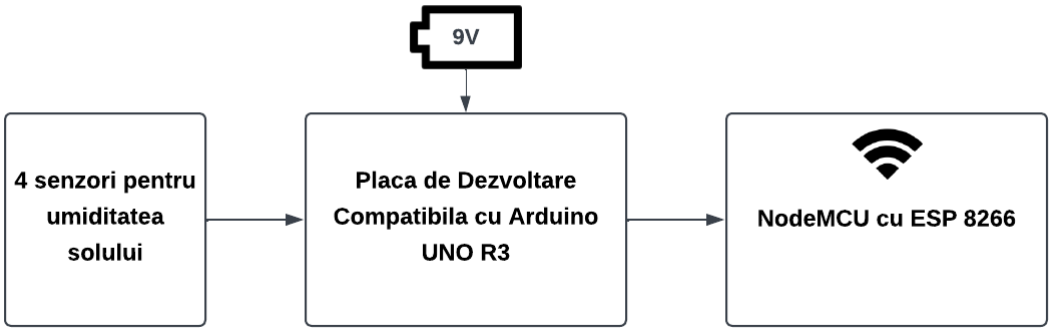

Figura 9-Arhitectura hardware a collectorului

Ca unitate de procesare, am utilizat o placă de dezvoltare compatibilă cu Arduino UNO R3 care se ocupă cu preluarea datelor de la cei 4 senzori de umiditate a solului și o placă de dezvoltare NodeMCU cu ESP 8266 cu Wi-Fi încorporat care are ca rol preluarea datelor de la cealaltă placă și transmiterea acestora utilizând protocolul MQTT către aplicația web.

Senzorii de umiditate a solului furnizează datele în format analogic către intrările analogice ale plăcii de dezvoltare compatibilă cu Arduino UNO R3. Aceasta le preia, le prelucrează și le transmite mai departe către NodeMCU prin comunicația Serial (UART) prin intermediul pinilor RX (receptor) și TX (transmițător). NodeMCU preia datele de la cealaltă placă și le trimite mai departe prin protocolul MQTT către aplicația web care se va ocupa mai departe de prelucrarea acestora.

Această componentă este alimentată cu o baterie de 9V, fiind o soluție pentru absența unei surse de alimentare în apropiere ei.

4.2.1.1 Placa de Dezvoltare Compatibilă cu Arduino UNO R3 (ATmega328p + ATmega16u2)

Placa de dezvoltare compatibilă cu Arduino UNO R3 este una din principalele componente ale collectorului.

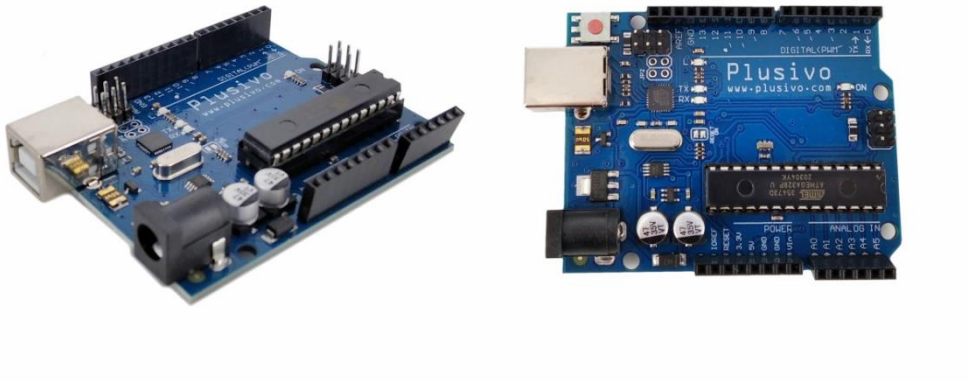

Figura 10-Placa de Dezvoltare Compatibilă cu Arduino UNO R3 (ATmega328p + ATmega16u2)

Această placă este formată din două microcontrollere ATmega328P ca microcontroller principal și un ATmega16U2 pentru gestionarea comunicațiilor USB.

ATmega328P este microcontrollerul principal. Are 32 KB de memorie flash pentru stocarea programului, 2 KB de SRAM, 20 de pini de intrare/ieșire digitali (dintre care 6 pot fi folosiți ca ieșiri PWM), 6 intrări analogice, un oscilator de 16 MHz, și o serie de alte funcționalități.

ATmega16U2 este un al doilea microcontroller care se ocupă de comunicarea între calculator și ATmega328P prin USB. Acesta înlocuiește chipul FT232RL utilizat pe versiunile anterioare ale Arduino Uno și permite mai multe funcționalități, inclusiv capacitatea de a fi programat ca un dispozitiv USB de diferite tipuri, precum tastatură, mouse, joystick, etc.

Placa are 14 pini digitali de intrare/ieșire (numițiți și GPIO - General Purpose Input/Output), dintre care 6 pot fi folosiți pentru a produce semnale PWM (modulare în lățime de impuls), precum și

6 intrări analogice. Acești pini permit plăcii să interacționeze cu o varietate de componente externe, precum LED-uri, motoare, senzori și multe altele.

Placa poate fi alimentată fie prin conectorul de alimentare în formă de baril (7-12V), fie prin portul USB (5V). De asemenea, are un pin de alimentare de 3.3V și un pin de alimentare de 5V pentru alimentarea componentelor externe.

Arduino Uno poate fi programat folosind mediul de dezvoltare Arduino (Arduino IDE). Acesta permite programarea într-un limbaj bazat pe C++, și oferă o bibliotecă de cod vastă care permite controlul ușor al multor tipuri de hardware.

Placa are o conexiune hardware serial (UART sau USART) pe pini 0 (RX) și 1 (TX). De asemenea, ATmega328P are suport pentru comunicație I2C și SPI. Folosind biblioteca software serial în Arduino IDE, practic orice pin digital poate fi folosit pentru comunicare serial.[10]

4.2.1.2 NodeMCU cu ESP 8266

Placa de dezvoltare NodeMCU cu ESP 8266 este cea de-a doua componentă principală a collectorului.

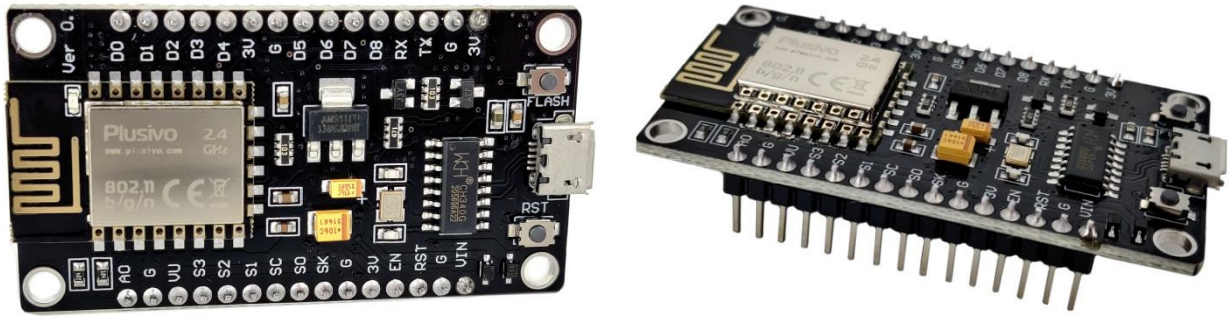

Figura 11-NodeMCU cu ESP 8266

NodeMCU este o placă de dezvoltare open-source care permite programarea ușoară a modulelor ESP8266 pentru a construi aplicații bazate pe Internetul Obiectelor (IoT). NodeMCU utilizează

ESP8266, un microcontroller dezvoltat de Espressif. Acesta este un cip de 32 de biți de dimensiuni mici.

O caracteristică centrală a ESP8266 este WiFi-ul încorporat. Acesta poate funcționa atât ca un client WiFi, conectându-se la rețele WiFi existente, cât și ca un punct de acces WiFi, creând propria rețea pentru alte dispozitive să se conecteze.

ESP8266 pe NodeMCU vine cu până la 17 pin-uri GPIO (General Purpose Input Output) care pot fi utilizate pentru a conecta diferite componente hardware, cum ar fi LED-uri, senzori, butoane, etc. Aceste pin-uri pot fi controlate folosind cod scris în limbajul de programare C++ (folosind mediul de dezvoltare Arduino).

NodeMCU poate fi alimentat prin intermediul unui cablu USB micro. Este capabil să funcționeze cu tensiuni de până la 20V, deși în general se recomandă să fie alimentat la 3.3V sau 5V pentru a preveni deteriorarea plăcii. Placa are de asemenea un pin de 3.3V pentru a furniza alimentare la alte componente care funcționează la 3.3V și un pin GND pentru masă.

NodeMCU suportă protocoale de comunicație precum SPI, I2C, și UART, astfel încât să se poată conecta la o varietate de senzori, dispozitive de stocare, și alte componente.[11]

4.2.1.3 Senzor capacitiv de umiditate a solului

Un senzor capacitive de umiditate a solului este un dispozitiv care măsoară umiditatea solului prin detectarea capacitivă, spre deosebire de un senzor rezistiv. Este fabricat din material rezistent la coroziune, ceea ce îl face mai puțin sensibil la coroziune și uzură, și adesea mai precis și mai fiabil.

Figura12-Senzor capacitiv de umiditate a solului

Senzorul capacitive de umiditate a solului dispune de o serie de caracteristici tehnice, printre care se numără o tensiune de operare cuprinsă între 3.3 și 5.5 VDC, o tensiune de ieșire de la 0 la 3.0 VDC și un curent de operare de 5mA. Senzorul are o greutate de 15 grame și dimensiuni de 99mm x 16mm. Pentru conectivitate, senzorul utilizează o interfață de tip PH2.0-3P.

Senzorul are doar 3 pini: VCC (alimentare), GND (masă) și un pin de semnal (analog).

Datorită faptului că plăcuța de dezvoltare compatibilă cu Arduino UNO R3 dispune de 6 intrări analogice, iar senzorul oferă voltajul doar în format analog acesta poate fi conectat direct la intrările analogice ale acesteia.

Senzorul de umiditate a solului funcționează pe baza principiului capacitiv. Capacitivitatea este proprietatea unui condensator de a stoca sarcină. În cazul senzorului de umiditate a solului, condensatorul este format din două plăci conductoare cu un dielectric între ele (solul, în acest caz).

Când tensiunea este aplicată la plăcile conductoare (sau electrozii) ale senzorului, se formează un câmp electric în dielectric (sol). Cantitatea de sarcină electrică stocată în acest câmp electric depinde de constanta dielectrică a dielectricului. Întrucât apa are o constantă dielectrică mare în comparație cu

celelalte componente ale solului (neregularități, aer etc.), cantitatea de sarcină electrică stocată în câmpul electric crește odată cu umiditatea solului.

Prin măsurarea acestei capacități, senzorul poate determina umiditatea solului. Tensiunea de ieșire a senzorului este proporțională cu umiditatea solului: cu cât este mai mare umiditatea, cu atât este mai mare tensiunea de ieșire.[12]

4.2.1.4 Baterie 9V

Bateria este sursa de alimentare pentru componenta collector.

Figura 13-Baterie 9V

Bateria de 9 volți este de obicei făcută din șase celule de 1,5 volți conectate în serie, care împreună oferă o tensiune totală de 9 volți. Majoritatea bateriilor de 9V folosesc tehnologia alcalină, dar sunt disponibile și variante cu NiMH (hidrură de nichel metal) sau Li-Ion (ioni de litiu). Dimensiunile standard pentru o baterie de 9V sunt aproximativ 48mm în înălțime, 26.5mm în lățime și 17.5mm în grosime.

Bateriile de 9V sunt distincte datorită conectorilor lor de tip "snap" - un conector pozitiv și unul negativ așezați unul lângă celălalt pe partea de sus a bateriei.

4.2.1.5 Schema electrică

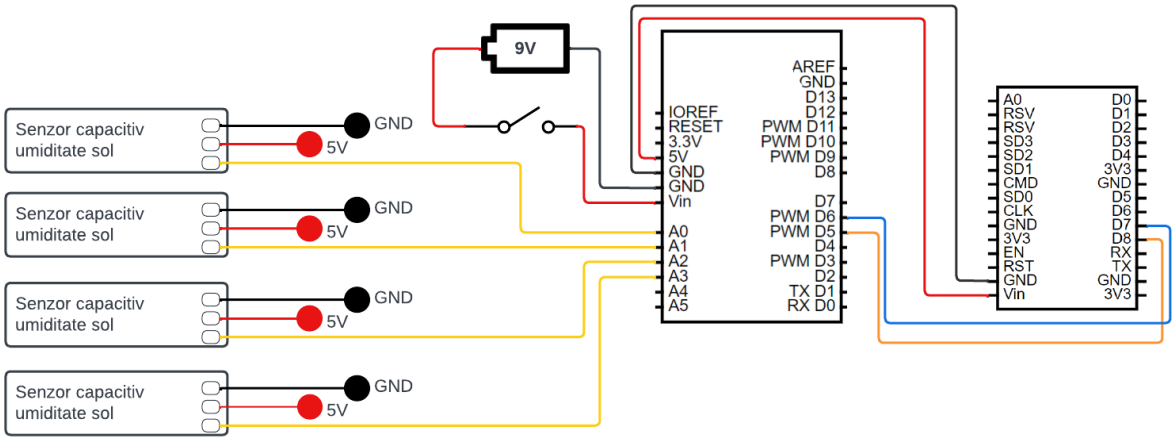

Figura 14-Schema electrică a collectorului

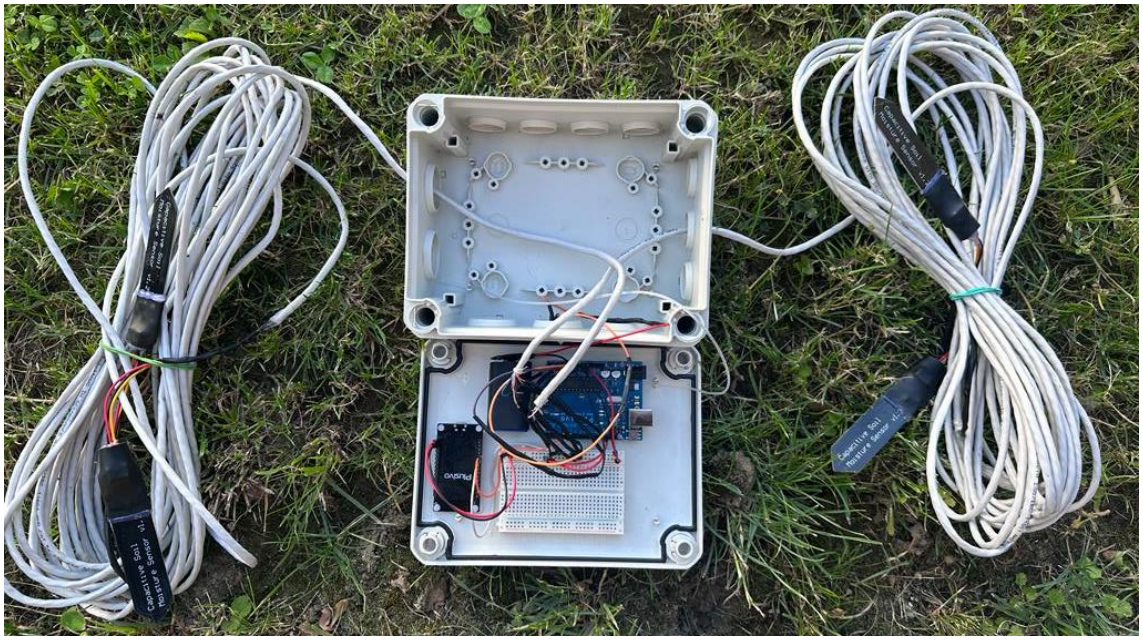

Figura 15-Configurație collector

Figura 16-Amplasarea collectorului pe teren

În figura 14 sunt reprezentate conexiunile electrice dintre placa de dezvoltare compatibilă Arduino UNO R3, NodeMCU, senzorii capacitivi de umiditate a solului si baterie. În figura 15 este reprezentat collectorul plasat în cutie de protecție.

4.2.2 Controller

Controllerul este componenta hardware care are ca rol activarea și dezactivarea valvelor care se ocupă de controlul apei. Decizia de activare sau dezactivare a valvelor este luată la nivelul aplicației web și este trimisă către controller folosind protocolul MQTT. În figura următoare este prezentată structura controllerului.

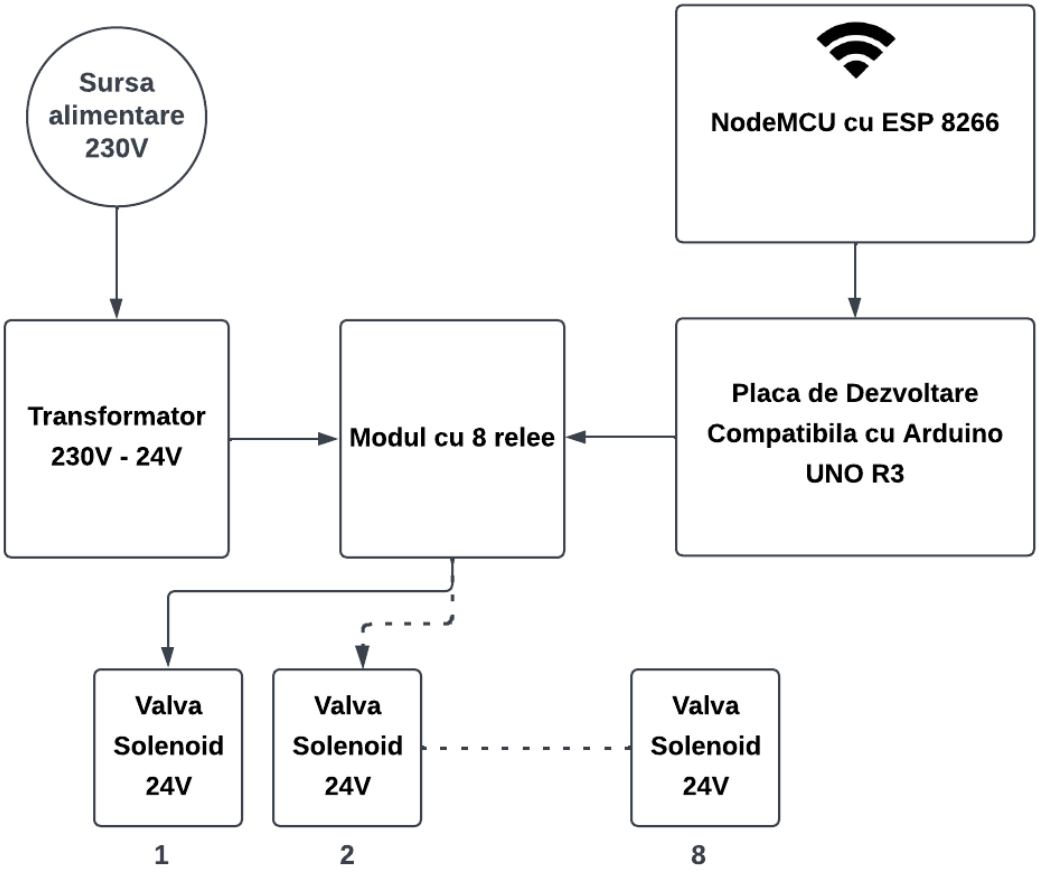

Figura 17- Arhitectura hardware a controllerului

Ca unitate de procesare, am utilizat, ca și în cazul collectorului, o placă de dezvoltare compatibilă cu Arduino UNO R3 care se ocupă cu activarea și dezactivarea valvelor prin controlarea modulului cu 8 relee prin intermediul pinilor GPIO. Fiecare releu îi corespunde o valvă solenoid de 24V. Deoarece la momentul de față este implementată o singură zonă, va fi utilizat un singur releu, restul de 7 fiind disponibile pentru implementări viitoare. Placa principală se mai ocupă și de preluarea datelor de la placa de dezvoltare NodeMCU cu ESP 8266 cu Wi-Fi încorporat, care are ca rol preluarea datelor de la aplicația web, utilizând protocolul MQTT.

4.2.2.1 Relee

O placă cu 8 relee este un modul de comutare care permite unui circuit de control de joasă tensiune/potență (cum ar fi cel de la un microcontroller precum Arduino sau ESP8266) să comute în mod sigur un circuit de încărcare de înaltă tensiune/potență.

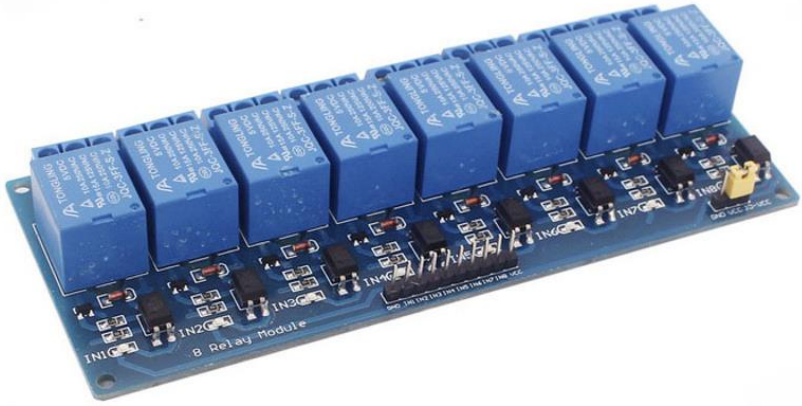

Figura 18- Placa cu 8 relee

O placă cu 8 relee, așa cum sugerează și numele, are 8 canale independente. Acest lucru înseamnă că poate controla până la 8 dispozitive separate, fiecare cu propria sa alimentare și circuit de comutare.

Releele pot fi de mai multe tipuri, dar cele mai comune sunt cele electromecanice care folosesc un electromagnet pentru a deschide sau a închide contactele fizice.

Capacitatea de comutare a unei plăci cu relee se referă la cantitatea maximă de curent pe care o poate manipula în siguranță. Majoritatea plăcilor cu 8 relee pot gestiona tensiuni de până la 250VAC sau 30VDC, cu un curent de până la 10A sau mai mult.

Plăcile cu 8 relee au de obicei o interfață simplă, cu un pin pentru fiecare releu care poate fi conectat la un pin digital al unui microcontroller. De asemenea, vor avea nevoie de o sursă de alimentare separată pentru a alimenta releele.

Multe plăci cu relee au caracteristici de siguranță integrate, cum ar fi optocuploare pentru a izola circuitul de control de cel de încărcare, reducând riscul de deteriorare a microcontrollerului.

Pinii de control ai plăcii de relee includ un pin VCC care alimentează modulul de relee, doi pini GND care reprezintă masa, opt pini IN folosiți pentru controlul fiecărui releu individual, un alt pin VCC care permite selectarea sursei de alimentare și un pin RY-VV care oferă o sursă alternativă de alimentare pentru modulul de relee. [13]

Următoarele conexiuni au fost realizate între pinii plăcuței de dezvoltare și cei ai plăcii cu 8 relee:

- VCC1 – 5V (Arduino)
- IN1 – Pin4
- IN2, IN3, IN4, IN5, IN6, IN7, IN8 (disponibili pentru implementări ulterioare)
- GND1 – GND (Arduino)
- GND2 –
- VCC2 – RY-VCC
- RY-VCC – VCC2

Partea electrică gestionată de releu include 3 pini de conectare: NC (În mod normal închis), COM (Comun), NO (În mod normal deschis). La COM vom conecta un cablu al circuitului pe care vrem să-l controlăm, iar cel de-al doilea cablu îl vom conecta fie la NC, fie la NO, în funcție de ce preferăm.

4.2.2.2 Electrovalve

Electrovalvele solenoid, cunoscute și sub numele de supape solenoid sau robinete solenoid, sunt dispozitive electromecanice care controlează fluxul unui fluid sau al unui gaz.

Figura 19- Electrovalva Hunter PGV FE 1"

Principiul de funcționare al unei electrovalve solenoid se bazează pe legea electromagnetismului. Când un curent electric trece prin bobina solenoidului, acesta creează un câmp magnetic. Acest câmp magnetic acționează asupra unui piston sau plunger în interiorul valvei, forțându-l să se miște și astfel deschizând sau închizând supapa.

Valvele solenoid pot fi Normally Closed (NC), fluxul de fluid sau gaz este blocat atunci când valvei nu îi este aplicat curent electric sau Normally Open (NO), atunci când nu există curent aplicat solenoidului, fluidul sau gazul poate trece liber. [14]

În desfășurarea lucrării am utilizat o valvă solenoid produsă de compania Hunter, alimentată cu curent alternativ 24V Normally Closed.

4.2.2.3 Transformator

Un transformator este o componentă electrică pasivă care transferă energie electrică între două sau mai multe circuite prin procesul de inducție electromagnetică. Acesta este folosit pentru a crește (step-up) sau a reduce (step-down) tensiunile în rețelele electrice.

Un transformator se compune dintr-o bobină de intrare, numită înveliș primar, și una sau mai multe bobine de ieșire, numite învelișuri secundare. Aceste bobine sunt înfășurate pe un nucleu comun de fier sau de oțel siliconat, care amplifică câmpul magnetic produs de bobine.

Figura 20- Transformator Step Down 100VA

Transformatorul pune la dispoziție 5 borne, dintre care 2 vor fi folosite pentru conectarea circuitului la 230VAC, 2 vor fi folosite pentru alimentarea circuitului de 24VAC, iar ultima va fi folosită pentru conectarea la împământarea sursei de 230VAC. [15]

În desfășurarea lucrării am utilizat un transformator Step Down de la 230VAC la 24VAC, pentru alimentarea valvelor solenoid, care are o putere aparentă de 100VA. Acest lucru sugerează că amperajul oferit la ieșire este de 100VA/24V, adică 4.14A. Având în vedere că o electrovalvă solenoid necesită 0.37A pentru a funcționa, acest transformator ar putea alimenta simultan 11 astfel de valve. În prezent, există doar o valvă care necesită alimentare, așadar acest transformator este ideal. În plus, oferă suficientă capacitate pentru alimentarea unor eventuale valve suplimentare în viitor.

4.2.2.4 Schema electrică

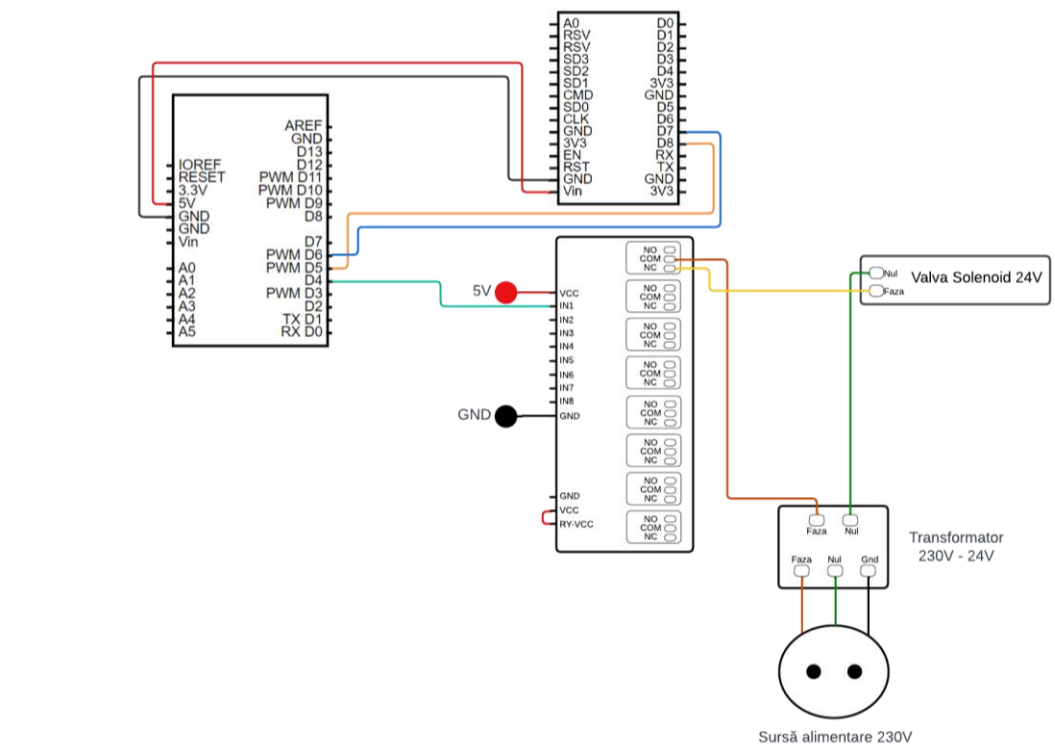

Figura 21- Schema electrică a controllerului

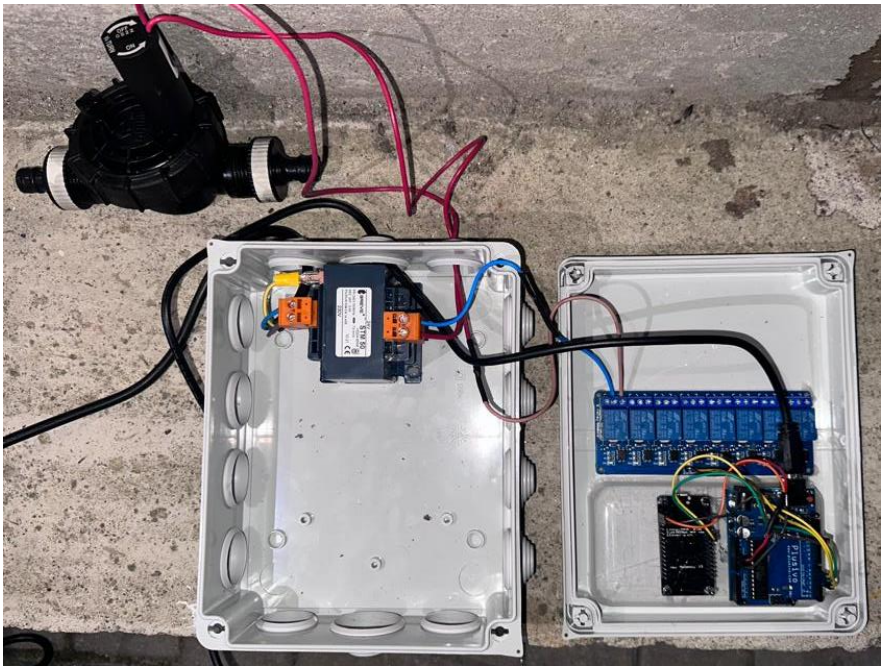

Figura 22-Configurație controller

Figura 23-Amplasarea controllerului pe teren

În figura 21 sunt reprezentate conexiunile electrice dintre placa de dezvoltare compatibilă Arduino UNO R3, NodeMCU, placa cu 8 relee, transformator și valva solenoid. În figura 22 este reprezentat controllerul plasat în cutie de protecție.

4.3 Interconectarea componentelor

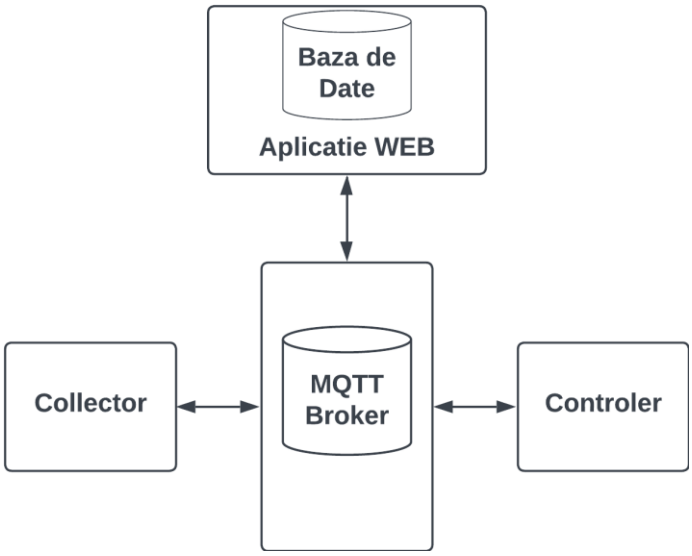

Figura 24-Interconectarea componentelor

Ca și arhitectură pe întreg sistemul componentele acestuia (collectorul, controllerul, aplicația web) comunică între ele utilizând protocolul MQTT prin intermediul unui MQTT Broker. Comunicarea cu baza de date este executată de către aplicația WEB aceasta fiind integrată aplicației.

4.3.1 Transferul de date (MQTT)

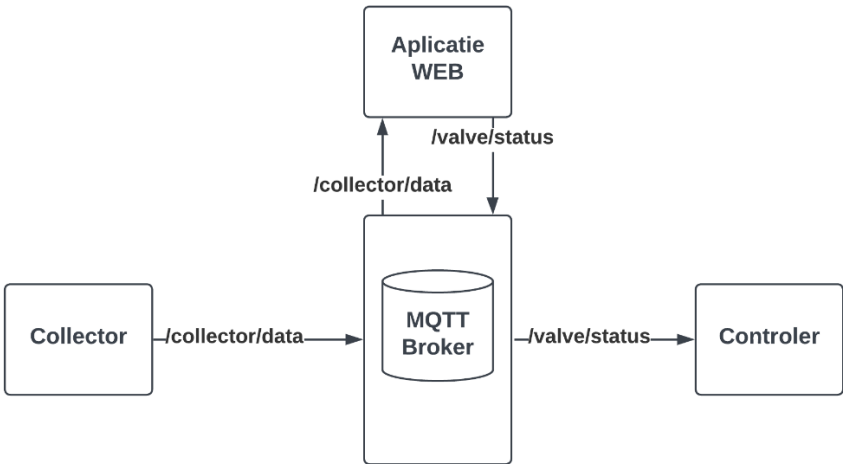

Figura 25-Comunicarea dintre componente prin protocolul MQTT

Transferul de date bazat pe protocolul MQTT se realizează între componentele hardware ale sistemului și aplicația web. Collectorul publică datele citite de la senzori pe topicul "/collector/data", acestea urmând să fie preluate de aplicația web care este abonată la acest topic. Aplicația web prelucrează datele și în urma interacțiunii utilizatorului, aceasta publică pe topicul "/valve/status" starea electrovalvei, astfel aceasta se va porni și opri în urma deciziei utilizatorului sau a sistemului, în cazul controlului automat.

Broker-ul utilizat este unul open-source oferit de EMQ X. Pentru conectarea la acesta am avut nevoie de numele de domeniu al brokerului MQTT "broker.emqx.io" și portul 1883.

Clienți sunt atât componentele hardware: collectorul și controllerul, cât și aplicația web. Topicurile utilizate sunt: "/collector/data" și "/valve/status".

4.3.2 Baza de date

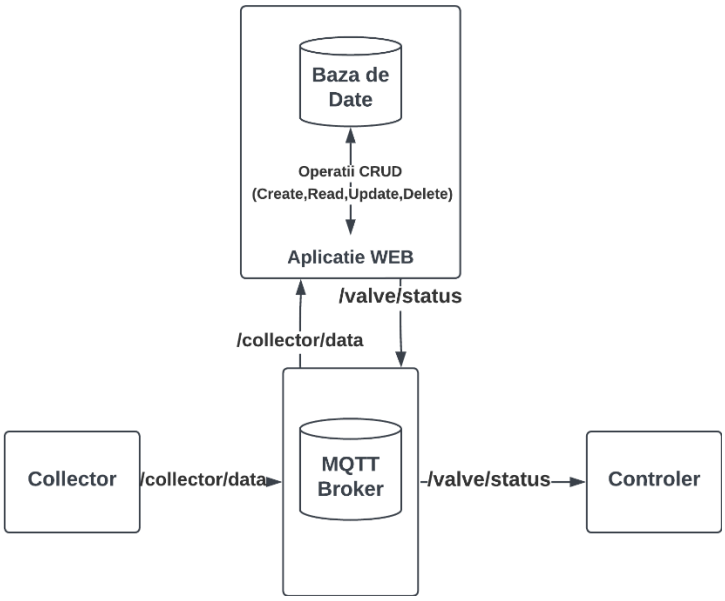

Figura 26-Comunicarea dintre componente cu baza de date

Baza de date utilizată este una SQL Server și conține următoarele tabele: DataRead, DataReadAccuWeather, ValveState, Schedule, Users, IrrigationTime. Aceste tabele au fost create utilizând clasele corespondente în codul aplicației, prin intermediul Entity Framework.

Tabelul “Users” conține informațiile despre utilizatori necesare la înregistrare și conectare, tabelul “DataRead” conține datele preluate de la sol de către senzorii de umiditate, tabelul “DataReadAccuWeather” conține datele meteo preluate de la API-ul “AccuWeather”, tabelul “ValveStatus” conține datele despre starea actuală a valvei/valvelor și timpul la care s-a modificat acea stare, tabelul “Schedule” conține datele despre programările la care trebuie să fie activat sistemul, precum timpul la care să pornească și la care să se oprească, dar și statusul actual al programării, iar tabelul “IrrigationTime” conține timpul de irigare calculat de către aplicație la utilizarea modului automat.

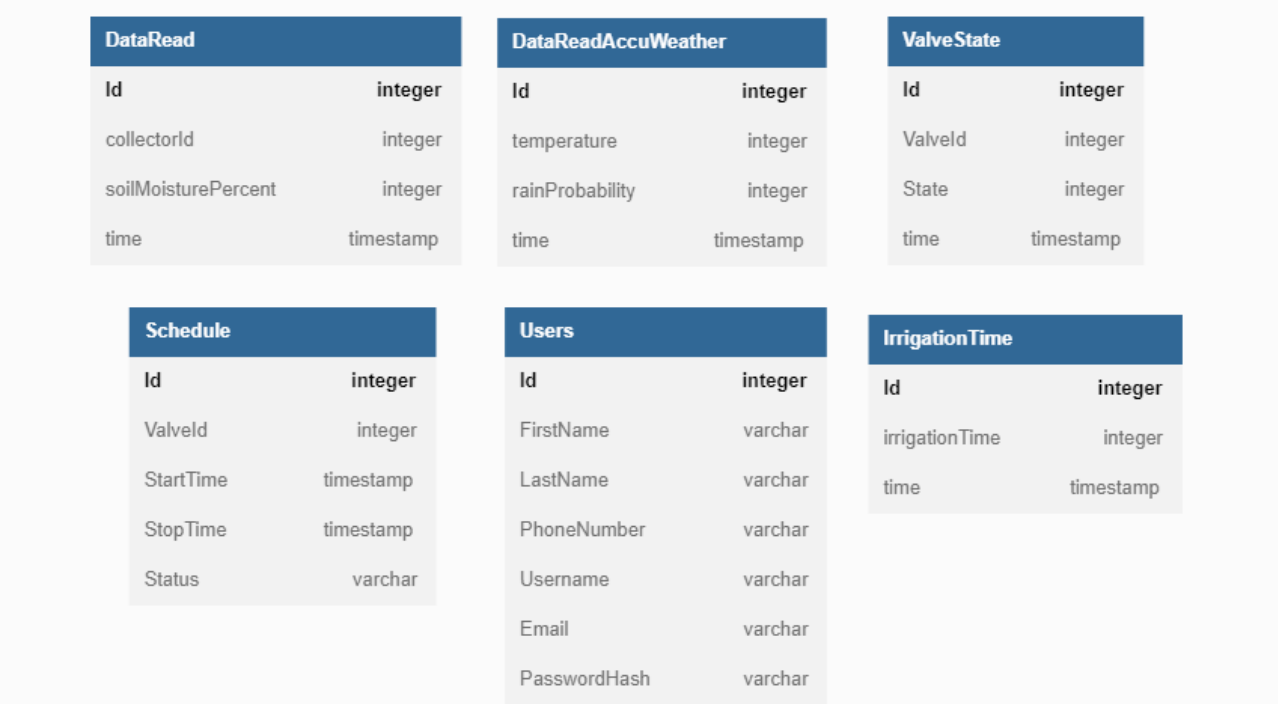

Figura 27-Structura tabelelor din baza de date

Baza de date este integrată in aplicatia web, iar toate operațile cu aceasta (“Creeate”,”Read”,”Update”,”Delete”) sunt realizate de către aplicația web.

4.4 Arhitectura Software

Software-ul este o colecție de programe, date și instrucțiuni care permit funcționarea și utilizarea unui sistem informatic. Este componenta non-fizică a unui sistem informatic, care este formată din codul de programare și fișierele asociate acestuia.

Aplicațiile software realizate în desfășurarea lucrării sunt cinci. Aplicația web care este aplicația principală și este scrisă în limbajul de programare C#, utilizând framework-ul ASP.NET. Collectorul și controllerul rulează aplicații dezvoltate în mediul de programare Arduino IDE, în limbajul de programare C++. Codurile sursă ale celor cinci aplicații sunt disponibile pe GitHub, fiecare în propriul său repozitoriu. [16][17][18]

4.4.1 Collector

Rolul collectorului este acela de a prelua în permanență datele de la sol, mai precis umiditatea acestuia, de a le pregăti și de a le trimite mai departe folosind protocolul MQTT către aplicația web care le va stoca mai departe în baza de date.

Aplicațiile destinate collectorului sunt dezvoltate în limbajul de programare C++, folosind librării open source care ne ajută la conectarea componentelor utilizate în această lucrare.

Librăriile folosite de către aplicația care rulează pe plăcuța compatibilă cu Arduino UNO R3 sunt:

- SoftwareSerial.h: utilizată pentru conectarea serială la plăcuța NodeMCU cu ESP8266.

Librăriile folosite de către aplicația care rulează pe plăcuța NodeMCU cu ESP8266 sunt:

- SoftwareSerial.h: utilizată pentru conectarea serială la plăcuța compatibilă cu Arduino UNO R3;
- PubSubClient.h: utilizată pentru conectarea la Broker-ul MQTT;
- ESP8266WiFi.h: utilizată pentru conectarea plăcuței la rețeaua WiFi.

Logica aplicației care rulează pe plăcuța compatibilă cu Arduino UNO R3 este simplă, având ca scop citirea valorilor de umiditate a solului de la patru senzori diferiți și trimiterea acestor valori prin intermediul unei conexiuni seriale către un alt dispozitiv care are un receptor serial configurat.

În secțiunea de început, se include biblioteca SoftwareSerial pentru a configura comunicația serială între Arduino și placuta NodeMCU. Apoi sunt declarate câteva variabile de tip String pentru a stoca valorile de umiditate ale fiecărui senzor.

Urmează declarațiile și inițializările pentru valorile de referință ale aerului și apei pentru a calibra citirile senzorilor de umiditate. Aceste valori sunt folosite în funcția "map" pentru a converti valorile citite de la senzori în procente de umiditate.

În cadrul funcției setup() se realizează următoarele acțiuni:

- Serial.begin(9600): Această linie inițializează și activează comunicarea serială între placa Arduino și calculator (prin intermediul portului USB). Se specifică o viteză de transmisie de 9600 de biți pe secundă (baud rate).
- espSerial.begin(9600): Această linie inițializează și activează o comunicație serială suplimentară cu un dispozitiv extern conectat la placa Arduino prin intermediul pini RX (recepție) și TX (transmisie). Această comunicație este realizată cu placuta NodeMCU, care are propriul port serial. Viteza de transmisie specificată este, de asemenea, 9600 de biți pe secundă.

Prin inițializarea comunicațiilor seriale în funcția setup(), placa Arduino și modulul ESP8266 sunt pregătite să primească și să transmită date prin intermediul porturilor seriale definite. Aceasta asigură o conexiune corectă și stabilă între componentele hardware și dispozitivele externe.

Pentru fiecare senzor (1-4), codul efectuează următoarele acțiuni în bucla principală "loop":

- Citirea valorii analogice de la pinul corespunzător (A0-A3).
- Convertirea valorii citite în procente de umiditate folosind funcția "map".

- Verificarea și ajustarea valorilor extreme ale procentelor de umiditate pentru a se încadra în intervalul 0-100.
- Trimiterea valorii de umiditate prin conexiunea serială la modulul ESP8266.

Codul se repetă la intervale regulate de timp pentru a citi și a trimite valorile de umiditate de la toți cei patru senzori.

Logica aplicatiei care ruleaza pe plăcuța NopdeMCU cu ESP8266 face ca ESP8266 să funcționeze ca un pod între un dispozitiv conectat prin serial și un broker MQTT, transmițând datele primite prin serial la broker.

Codul utilizează librării pentru gestionarea conexiunii WiFi, comunicarea cu un broker MQTT și comunicarea serială.

Se stabilesc datele de autentificare pentru WiFi (SSID și parolă) și adresa serverului MQTT.

În funcția setup_wifi(), ESP8266 se conectează la rețeaua WiFi cu datele de autentificare furnizate. Dacă conexiunea este reușită, afișează adresa IP pe portul serial.

Funcția callback() este apelată atunci când se primește un mesaj MQTT pe unul din subiectele la care ESP8266 este abonat.

Funcția reconnect() se ocupă de reconectarea la serverul MQTT în cazul în care conexiunea este pierdută. Ea generează un ID de client aleator, încearcă să se conecteze la server și, odată conectată, publică un anunț și se abonează la un subiect.

În funcția setup(), este configurată conexiunea serială atât pentru portul serial hardware (conexiunea la computer) cât și pentru portul serial software (conexiunea la un alt dispozitiv). Se apelează funcția setup_wifi(). De asemenea, serverul MQTT este setat și se stabilește funcția callback pentru clientul MQTT.

În funcția loop(), verifică dacă clientul este conectat la serverul MQTT și dacă nu, încearcă să se reconecteze. De asemenea, verifică dacă există date disponibile pe portul serial. Dacă există, le citește, le afișează pe portul serial și le publică pe un subiect MQTT.

4.4.2 Controller

Rolul controllerului este de a primi datele de la aplicația web și de a activa sau dezactiva valva în funcție de datele primite. La fel ca aplicațiile pentru collector, și aplicațiile pentru controller sunt dezvoltate în mediul de dezvoltare Arduino IDE, în limbajul de programare C++, utilizând aceleași biblioteci, având nevoie de aceleași funcționalități.

Programul care rulează pe plăcuța Arduino este destinat să citească date de la un port serial și să controleze un releu în funcție de aceste date.

La început se include biblioteca SoftwareSerial, care permite comunicarea serială pe alți pini decât cei standard (0 și 1) pentru Arduino. Apoi se creează un obiect de tip SoftwareSerial pentru care pini Rx și Tx sunt 5 și 6. Inițializăm șirul de caractere "mod" cu 0, acesta fiind un șir care va stoca starea primită prin serial.

Funcția setup():

- Serial.begin(9600) și espSerial.begin(9600): Inițializează comunicația serială atât pentru portul serial hardware, conexiunea la computer, cât și pentru portul serial software, conexiunea la NodeMCU.
- pinMode(4, OUTPUT): Setează pinul 4 ca ieșire. Acest pin este conectat la placa cu relee care poate fi comandat prin ON/OFF.

În funcția loop(), programul verifică în mod constant dacă există date disponibile pe portul serial. Dacă sunt date disponibile, le citește și le stochează în variabila "mood". După aceea, în funcție de valoarea citită, se controlează starea pinului 4. Dacă valoarea este "1", pinul este setat pe LOW, ceea ce activează releul și se afișează mesajul "Relay activated" în Monitorul Serial. Dacă valoarea nu este "1", pinul

este setat pe HIGH, ceea ce dezactivează releul și se afișează mesajul "Relay deactivated" în Monitorul Serial.

Programul servește ca un simplu mecanism de control al unui releu, cu starea acestuia fiind determinată de informațiile primite printr-o conexiune serială.

Programul pentru placuța NodeMCU cu ESP8266 stabilește o conexiune WiFi, se conectează la un server MQTT și comunică cu palcuța Arduino prin portul serial.

La început sunt incluse librăriile necesare pentru conectarea la WiFi, comunicarea cu un server MQTT și comunicarea serială. Se stabilesc datele de autentificare pentru WiFi și adresa serverului MQTT.

Se declară un client WiFi și un client MQTT, câteva variabile pentru a stoca timpul ultimului mesaj și mesajul în sine, precum și un port serial software.

Funcția setup_wifi() inițiază conexiunea la WiFi. Dacă conectarea este reușită, afișează adresa IP pe portul serial hardware.

Funcția callback() este apelată când un mesaj este primit pe unul dintre subiectele la care ESP8266 este abonat. Mesajul este afișat pe portul serial hardware și este trimis către alt dispozitiv prin portul serial software.

Funcția reconnect() se ocupă de reconectarea la serverul MQTT dacă conexiunea este pierdută. Generează un ID de client aleator, încearcă să se conecteze la server și, odată conectat, publică un anunț și se abonează la un subiect.

În funcția setup(), sunt inițializate porturile seriale și se stabilește conexiunea WiFi. De asemenea, se setează serverul MQTT și funcția callback pentru clientul MQTT.

În funcția loop(), verifică dacă clientul MQTT este conectat și, dacă nu, încearcă să se reconecteze. Apoi, verifică dacă există date disponibile pe portul serial software. Dacă există, le citește și le trimite pe portul serial hardware.

4.4.3 Aplicația Web

Aplicația web are ca scop oferirea unei modalități de a interacționa cu sistemul într-o variantă mai prietenoasă și mai ușor de înțeles de către oricine. Aceasta are o interfață simplă și ușor de utilizat, oferindu-i utilizatorului acces la diverse funcționalități. Controlul manual al sistemului, crearea de programe de irigare, activarea modului automat de irigare sau vizualizarea datelor și a statisticilor sunt funcționalități care sunt puse la dispoziție utilizatorului aplicației. Vizualizarea datelor este permisă oricui interacționează cu aplicația, dar pentru a interacționa cu sistemul este necesară crearea unui cont de utilizator.

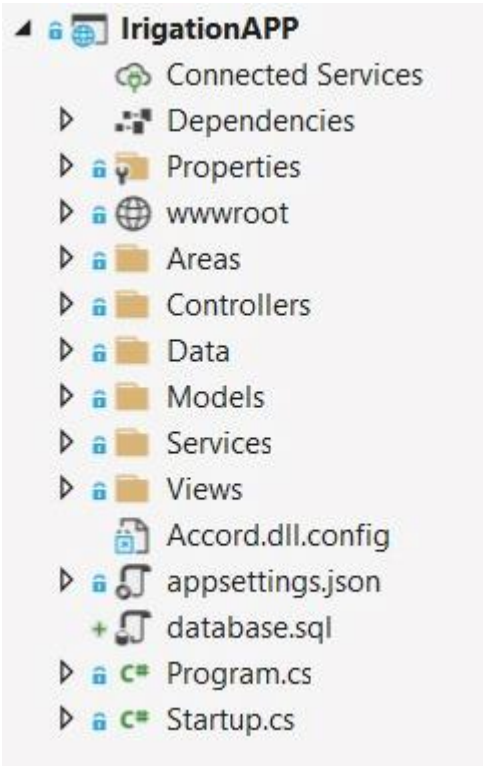

Figura 28-Structura aplicatiei web

Aplicația web este realizată în limbajul de programare C# utilizând framework-ul ASP.NET Core. Este o aplicație web de tipul MVC (Model-View-Contrloller) respectând acest design architectural.

Modelul reprezintă componenta de date a arhitecturii aplicației. Acesta se ocupă cu gestionarea datelor si a operatiilor cu date. Modelul servește ca punte între componentele View și Controller. Modelele conține definițiile de clase C# care reprezintă datele sau obiectele cu care aplicația lucrează. Acestea sunt modelele de date care reprezintă structura tabelelor din baza de date.

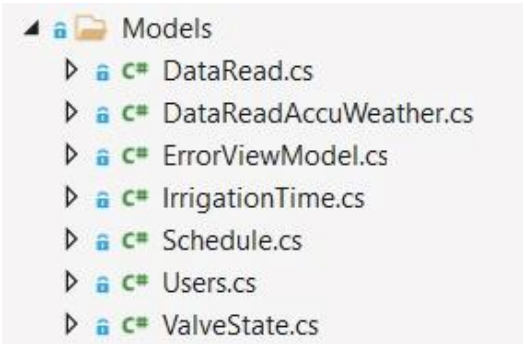

Figura 29-Structura directorului Models

View-ul este componenta care se ocupă cu prezentarea datelor către utilizator. Aceasta este, în esență, partea de "interfață cu utilizatorul" a aplicației. View-urile sunt fișiere HTML îmbogățite cu cod Razor, un limbaj de programare special conceput pentru generarea dinamică a conținutului HTML. Codul Razor utilizeaza datele provenite din Model și de la Controller pentru a genera HTML care se modifică în funcție de starea aplicației.

View-urile sunt responsabile pentru prezentarea datelor, ele nu conțin logica de afaceri sau logica de acces la date. Acestea se limiteaza la prezentarea datelor pe care le primesc de la Controller.

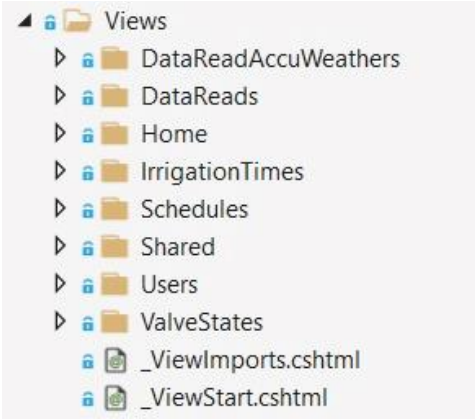

Figura 30-Structura directorului Views

Controllerul funcționează ca intermediar între Model, care reprezintă datele și logica de afaceri, și View, care este interfața cu utilizatorul.

Controllerul este responsabil pentru gestionarea solicitărilor utilizatorilor, cum ar fi click-uri pe butoane, selecții din dropdown-uri, introducerea de date în formular, etc. Când o astfel de solicitare ajunge la controller, acesta va executa logica adecvată pentru a procesa solicitarea. Acest lucru ar putea implica preluarea sau actualizarea datelor în Model și selectarea unui View pentru a răspunde utilizatorului.

Controllerul este componenta care coordonează interacțiunea între Model și View, gestionează fluxul datelor între acestea și controlează fluxul aplicației.

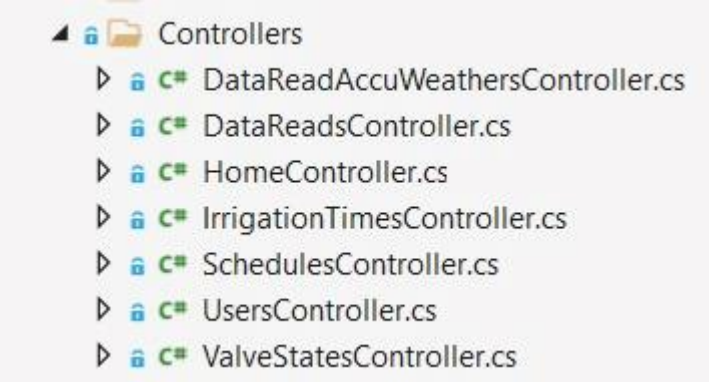

Figura 31-Structura directorului Controllers

Directorul "Data" conține clasa “ApplicationDbContext” care gestionează interacțiunile cu baza de date. Aceasta conține proprietăți de tip DbSet<T> pentru fiecare tip de entitate corespondente cu modelele din directorul "Models.

De asemenea mai continue si directorul “Migartion”. Migrațiile sunt utilizate de Entity Framework pentru a gestiona modificările schemelor bazei de date în timp. Fiecare migrare reprezintă o schimbare a schemei bazei de date, și Entity Framework le poate utiliza pentru a actualiza baza de date la o versiune specifică, creând sau modificând tabele în conformitate cu modelele din aplicație.

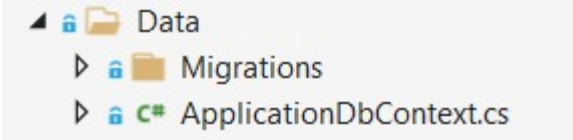

Figura 32-Structura directorului Data

Directorul “Services” este utilizat pentru a organiza clasele de servicii care conțin logica de afaceri sau interacțiunile cu servicii externe. Aceste servicii sunt injectate în Controlleri sau în alte servicii, ceea ce permite o bună separare a responsabilităților.

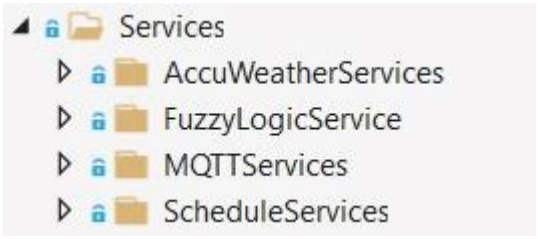

Figura 33-Structura directorului Services

AccuWeatherServices este responsabil pentru comunicarea cu API-ul AccuWeather pentru a prelua informații despre vreme.

FuzzyLogicServices conține logica de afaceri complexă pentru a lua decizii pe baza unor reguli difuze.

MQTTService gestionează comunicarea cu un broker MQTT prin creare unui client pentru a se abona la un topic.

ScheduleServices este responsabil cu verificarea constanta a planuri de irigare pentru activarea si dezactivarea la timpul corespunzător a sistemului.

Prin împărțirea acestei logici în servicii ajută la menținerea principiului separării responsabilităților, deoarece fiecare serviciu se ocupă de o anumită sarcină sau de un anumit aspect al aplicației.

Clasa Program servește ca punct de lansare pentru Aplicația Web, cu setările stabilite în clasa Startup.

Aplicția se bazează pe 7 controllere:

Controller	**Descriere**
DataReadAccuWeathers	Controllerul conține acțiunile: Index(), Index2(), Delete(int? id), DeleteConfirmed(int id), DataReadAccuWeatherExists(int id). Datele sunt perluate de la API-ul AccuWeather de către serviciul "AccuWeatherServices" și salvate în baza de date. Oferă: - citirea datelor din DataReadAccuWeather, - afișarea acestora într-un View, - ștergerea unei înregistrări specifice.
DataReads	Controllerul conține acțiunile: Index(), Index2(), Delete(int? id), DeleteConfirmed(int id), DataReadExists(int id). Datele sunt preluate de la Brokerul MQTT de către serviciul "MQTTService" care este abonat la topicul pe care sunt publicate datele de către componenta collector. Oferă: - citirea datelor din DataRead, - afișarea acestora într-un View, - ștergerea unei înregistrări specifice.
Home	Controllerul este utilizat pentru gestionarea acțiunilor legate de pagina principală. Acesta dă ca răspuns pagina principală de pornire (Index) care oferă un conținut diferit dacă este conectat un utilizator, față de cazul când nu este conectat.
IrrigationTimes	Controllerul este utilizat pentru gestionarea acțiunilor legate de datele de timp pentru irigare. Pentru calcularea și gestionarea timpului de irigare în modul automat este responsabil serviciul "FuzzzyLogicServices". Oferă: - citirea datelor din IrrigationTime, - afișarea acestora într-un View.
Schedules	Controllerul este utilizat pentru a gestiona programările pentru planurile de irigare.

	Controllerul conține acțiunile: Index(), Index2(), Create(), Delete(). Oferă: - citirea datelor din Schedule, - afișarea acestora într-un View, - crearea de noi înregistrări în baza de date, daca datele sunt valide, - ștergerea unei înregistrări specifice.
Users	Controllerul gestionează actiunile legate de utilizator. Controllerul contine acțiunile: Index(), Register(), Login(), Logout(). Oferă: - înregistrarea utilizatorilor, - logarea si delogarea acestora.
ValveStates	Controllerul gestionează starea valvei. Acesta include publicarea mesajelor către un broker MQTT, permițând sistemului de irigație să fie controlat de la distanță. Controllerul conține acțiunile: Index(), TurnOn(int id), TurnOff(int id), ValveStateExists(int id) Oferă: - citirea datelor din ValveState, - afișarea acestora într-un View, - schimbarea stării valvei prin actionarea butoanelor “Turn On” si ”Turn Off”.

Figura 34-Tabel cu acțiunile Controllerului

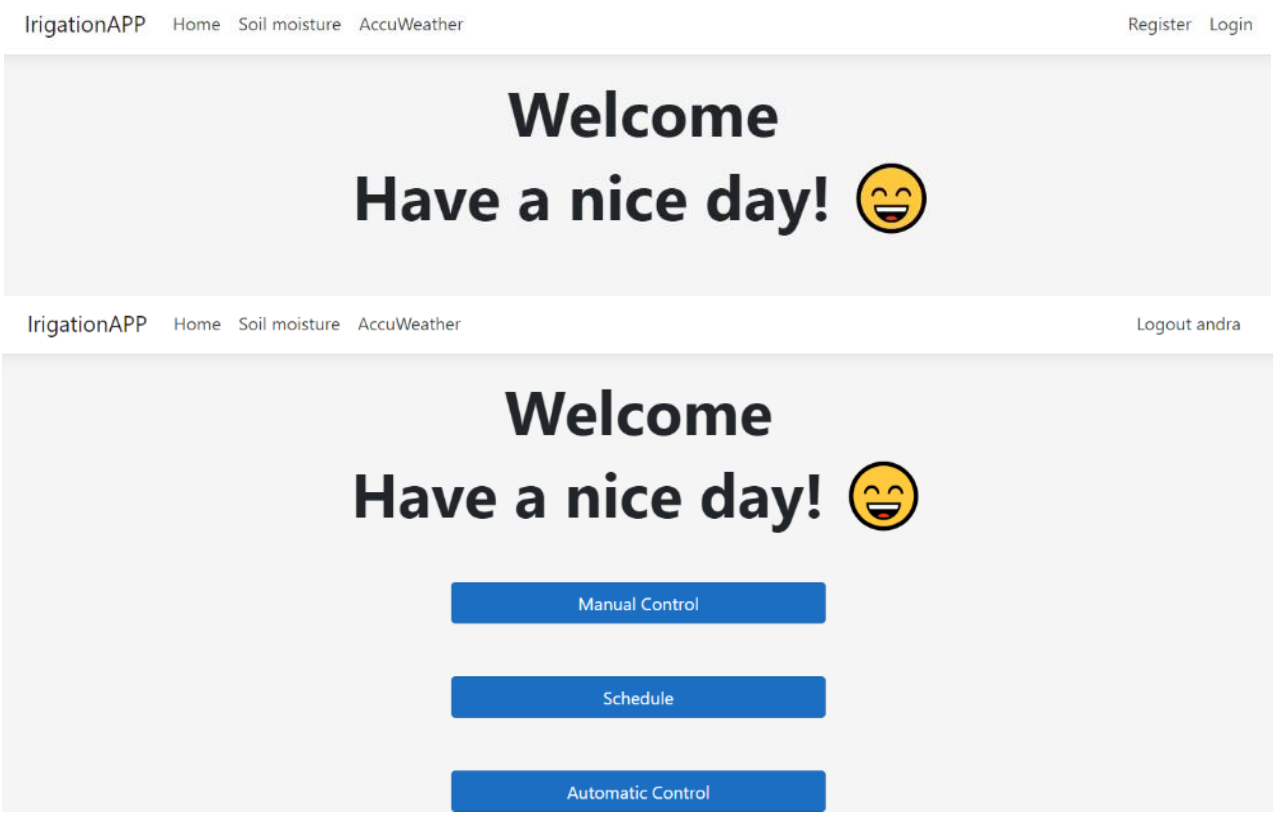

Figura 35-Pagina principală în cele 2 situații utilizator deconectat si conectat

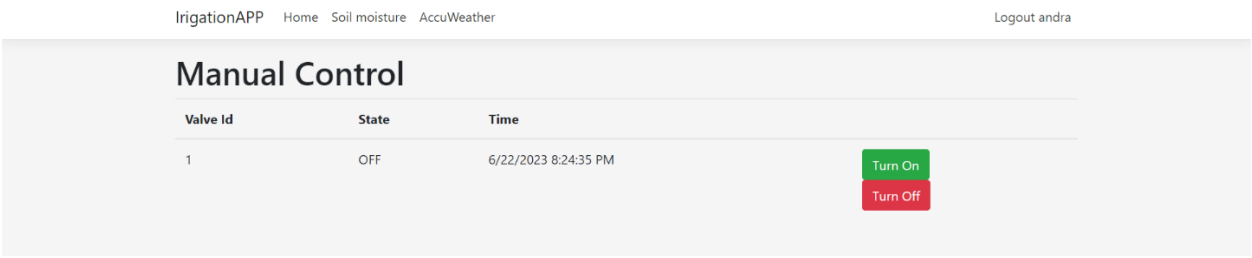

Figura 36-View-ul pentru controlul manual

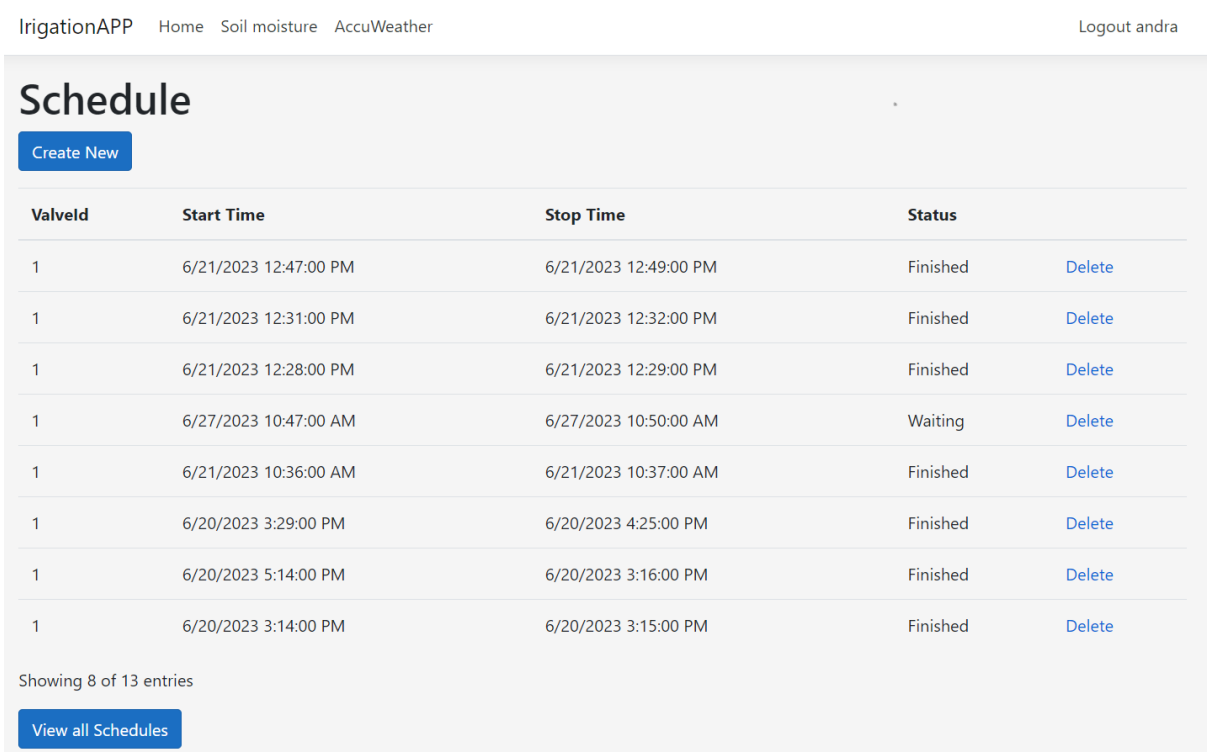

Figura 37-View-ul pentru controlul programat

5. REZULTATE PRODUSE

În această secțiune sunt prezentate rezultatele produse de sistem în urma testării acestuia. Acesta a fost testat pentru o perioadă scurtă de timp, motiv pentru care datele produse nu sunt cele mai relevante pentru evaluarea sistemului.

În figura 38 sunt prezentate rezultatele produse de către modul automat. Acesta a rulat de mai multe ori cu aceleași valori, motiv pentru care se repetă de mai multe ori același rezultat. Scopul figurii este de a arăta cum sunt prezentate rezultatele în aplicația web. Modul automat calculează timpul periodic la un interval de 3 ore. În aplicația web, datele sunt afișate atât sub formă de grafic cât și sub formă de tabel.

În figura 39 sunt prezentate datele preluate de la API-ul AccuWeather, dar și datele de la collector. Acestea se pot vizualiza în aplicația web cu și fără autentificare. Acestea sunt structurate atât sub formă de grafic cât și sub formă de tabel. Datele de la collector sunt preluate astfel: fiecare senzor trimite la 10 minute după celălalt. Date de la API-ul AccuWeather sunt preluate periodic o dată pe oră.

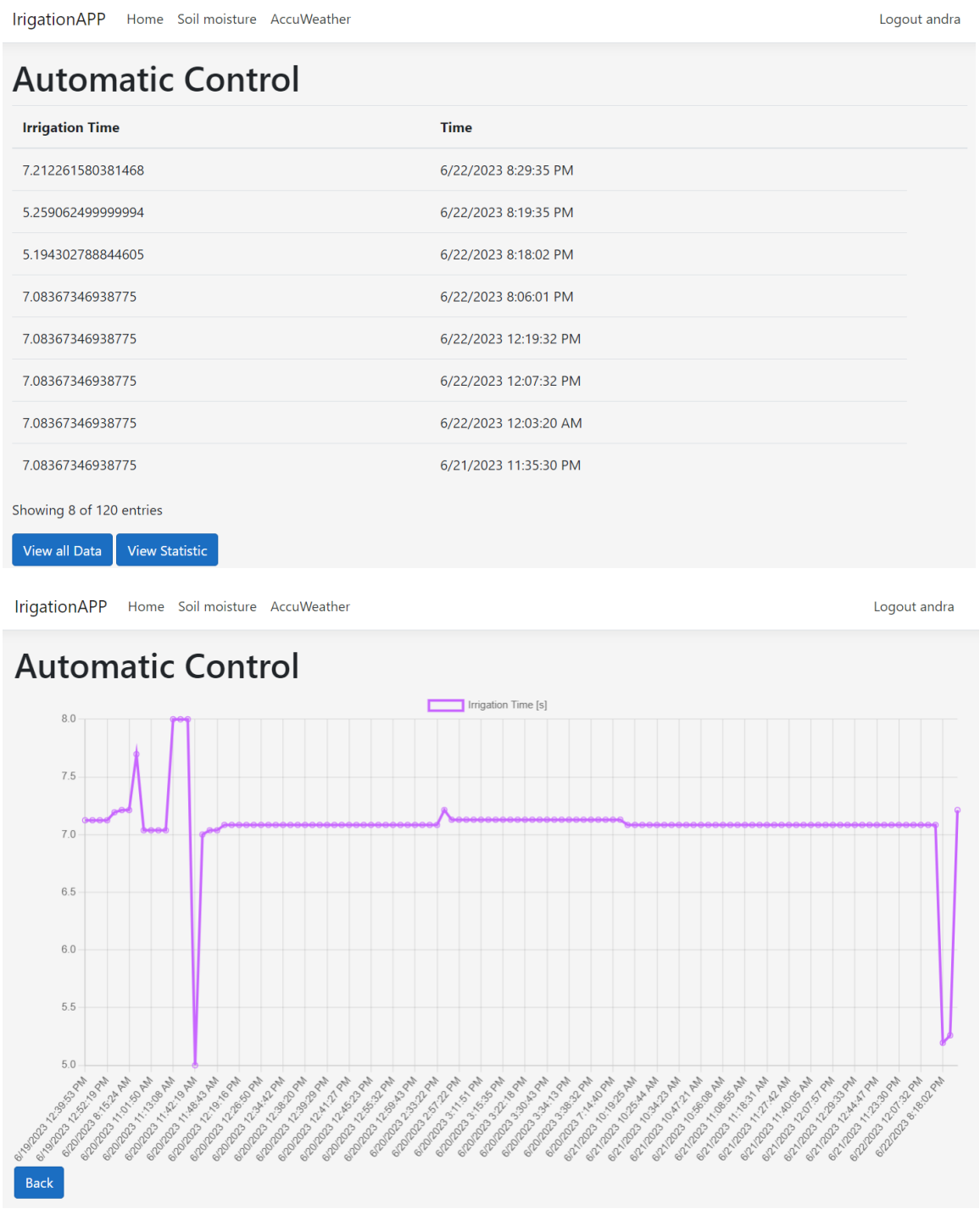

Figura 38-Rezultate produse de modul automat

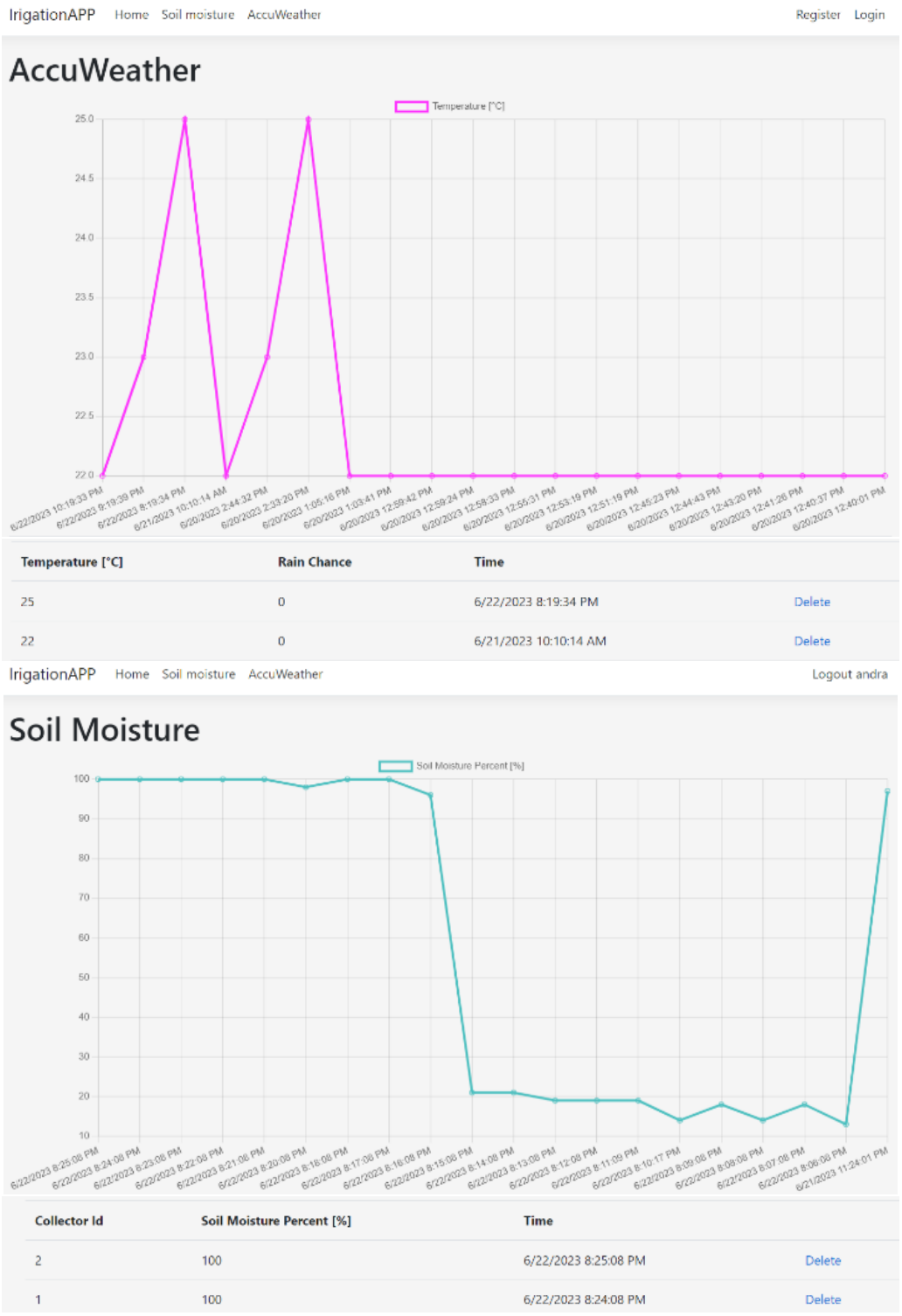

Figura 39-Datele preluate de la AccuWeather si citirile colctorului sub forma de garfic si table preluate din aplicatia web

6. CONCLUZII SI DEZVOLTĂRI ULTERIOARE

6.1 Concluzii

Soluția propusă vine în ușurarea și eficientizarea procesului de irigare. Prin conectarea tuturor componentelor, am realizat un sistem de irigare a plantelor care permite utilizatorilor acestuia să monitorizeze viața plantelor și să controleze procesul de irigare. Sistemul realizat vine cu o aplicație web care este ușor de utilizat și permite controlarea de la distanță, vizualizarea datelor despre mediu, oferind un control manual, automat sau crearea de planuri de irigare.

6.2 Limitări

Limitările actuale ale acestei soluții sunt: necesitatea unei acoperiri bune a semnalului Wi-Fi pentru facilitarea comunicării între componente și din lipsa unei surse de alimentare, collectorul este alimentat de la baterie, ceea ce necesită o verificare constantă și schimbarea acesteia periodic.

6.3 Dezvoltări ulterioare

Pentru implementări ulterioare, intenționez să extind sistemul pe suprafața întregii grădini pentru a acoperi și celelalte zone rămase, să îmbunătățesc și să adaptez metoda de calcul automat și la tipul plantelor și necesitățile acestora de umiditate, să adaug un plan de rezervă în cazul în care serverul este indisponibil sau în lipsa datelor de la collector sau de la API-ul AccuWeather.

REFERINȚE

[1] Mocrii, Dragos, Yuxiang Chen, and Petr Musilek. "IoT-based smart homes: A review of system architecture, software, communications, privacy and security." Internet of Things 1 (2018): 81-98.

[2] https://visualstudio.microsoft.com/

[3] Fezari, Mohamed, and Ali Al Dahoud. "Integrated development environment "IDE" for Arduino." WSN applications (2018): 1-12. ARDUINO

[4] https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

[5] https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-7.0

[6] https://www.oracle.com/ro/database/what-is-database/

[7] https://aws.amazon.com/what-is/mqtt/

[8] https://www.hivemq.com/mqtt/mqtt-protocol/

[9] https://www.geeksforgeeks.org/fuzzy-logic-introduction/

[10] https://www.robofun.ro/platforme-de-dezvoltare/arduino-uno-r3-atmega328p-placa-de-dezvoltare-compatibila-cu-arduino-cablu-usb.html?gclid=Cj0KCQjwmtGjBhDhARIsAEqfDEfgwg0LrsS0iqD2pIqJfXOsAzpYRzniT69YIaTQV0Me3aQWAig2pQoaAtxwEALw_wcB

[11] https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet

[12] https://media.digikey.com/pdf/data%20sheets/dfrobot%20pdfs/sen0193_web.pdf

[13] https://www.handsontec.com/dataspecs/module/8Ch-relay.pdf

[14] https://www.hunterindustries.com/sites/default/files/BR_PGV_em.pdf

[15] https://www.tme.eu/Document/b5f4f0781d1f64b68c027fd7a10bd1a0/STM-EN.pdf

[16] https://github.com/AndraB16/Collector.git

[17] https://github.com/AndraB16/Controller.git

[18] https://github.com/AndraB16/Irrigation-WEB-Application.git