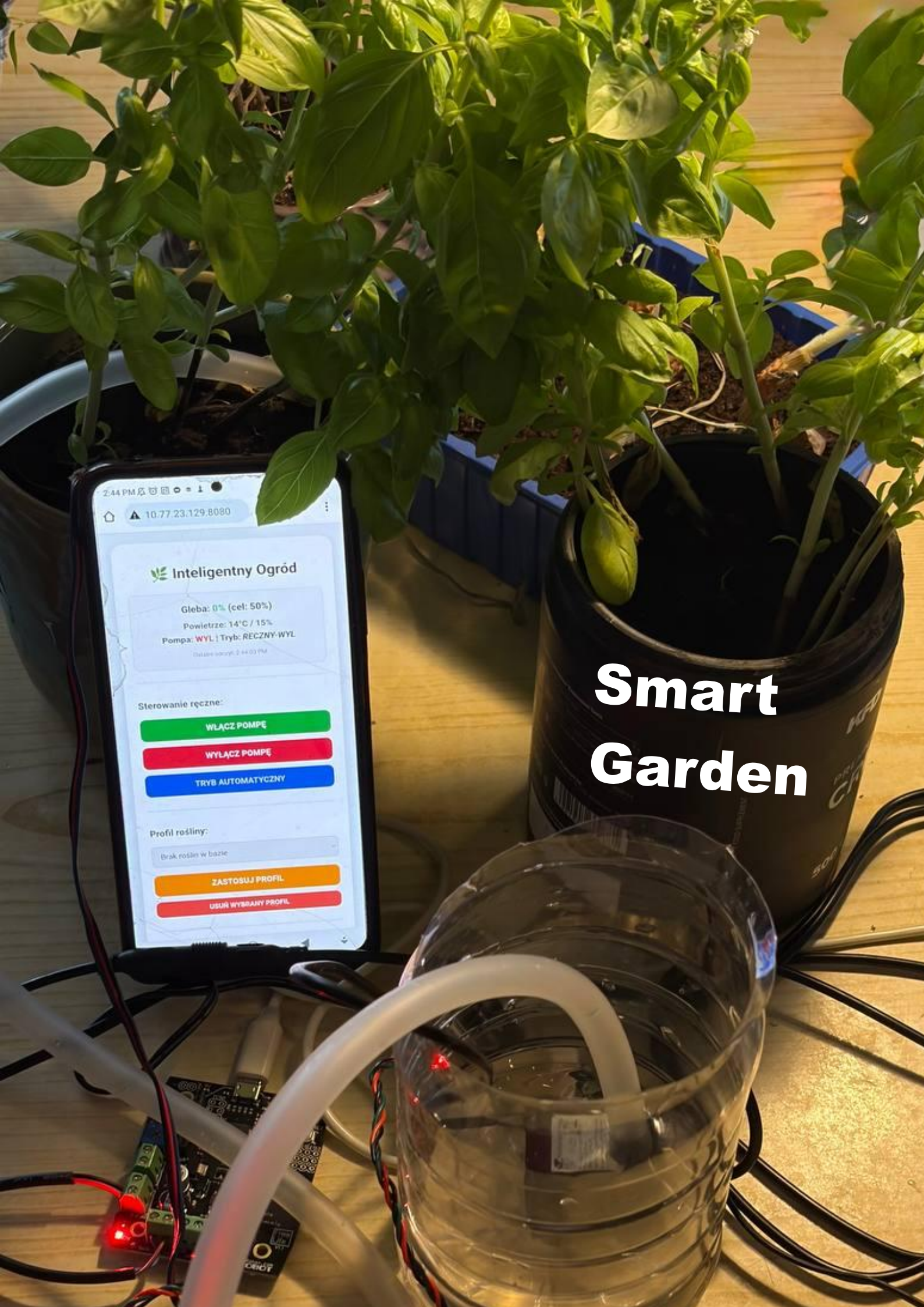

2:44 PM
10.77.23.129:8080
Inteligentny Ogród
Gleba: 0% (cel: 50%)
Powietrze: 14°C / 15%
Pompa: WYŁ | Tryb: RĘCZNY-WYŁ
Sterowanie ręczne:
WŁĄCZ POMPĘ
WYŁĄCZ POMPĘ
TRYB AUTOMATYCZNY
Profil rośliny:
Brak roślin w bazie
ZASTOSUJ PROFIL
USUŃ WYBRANY PROFIL
Smart
Garden

Smart Garden – Instrukcja użytkownika

Autorzy: Mikita Karabeika 252496, Maciej Walczak 251655.

Rozdział 1. Poradnik użytkownika.

1. Opis projektu

Smart Garden to system inteligentnego podlewania roślin, który automatycznie steruje pompą wodną na podstawie wilgotności gleby. System umożliwia również ręczne sterowanie oraz definiowanie profili roślin zapisanych w bazie danych.

Rozwiązanie składa się z trzech głównych części:

- mikrokontrolera Arduino odpowiedzialnego za odczyt czujników i sterowanie pompą,
- aplikacji webowej Spring Boot, która komunikuje się z Arduino i zarządza danymi.

System został zaprojektowany z myślą o prostocie obsługi oraz czytelnym podglądzie aktualnego stanu środowiska roślin.

2. Dane techniczne

2.1 Sprzęt

- Arduino (model Eco)
- Czujnik temperatury i wilgotności powietrza **DHT11**
- Czujnik wilgotności gleby
- Pompa wodna
- Komputer PC z portem USB

2.2 Oprogramowanie

- Arduino IDE
- Java 21
- Spring Boot
- Docker (z obrazem Postgres 15)
- Przeglądarka internetowa (np. Chrome, Firefox, Edge)

3. Struktura systemu

System działa według następującego schematu:

1. Arduino cyklicznie odczytuje dane z czujników.

2. Dane są przesyłane przez port szeregowy do aplikacji backendowej.
3. Backend udostępnia aktualne dane poprzez REST API.
4. Interfejs webowy pobiera dane i prezentuje je użytkownikowi.
5. Użytkownik może wysyłać komendy sterujące do Arduino (tryb pracy, próg wilgotności).

4. Uruchomienie systemu

Niniejszy rozdział opisuje proces uruchomienia systemu IoT – od pobrania kodu źródłowego, poprzez konfigurację mikrokontrolera Arduino, aż do uruchomienia backendu oraz interfejsu użytkownika.

4.1 Pobranie projektu i wymagania wstępne

Kod źródłowy projektu dostępny jest w repozytorium GitHub:
https://github.com/MikitaKarabeika/iot.git

Wymagania systemowe:

- Arduino IDE
- Płytka Arduino podłączona do komputera
- Docker oraz Docker Compose
- Java JDK oraz Maven
- Dostęp do sieci lokalnej (LAN)

Repozytorium należy sklonować na lokalny komputer:

git clone https://github.com/MikitaKarabeika/iot.git

4.2 Konfiguracja i uruchomienie Arduino

1. Otworzyć projekt Arduino w Arduino IDE.
2. Wybrać odpowiednią płytkę: *Arduino*.
3. Ustawić właściwy port szeregowy (COM).
4. Sprawdzić konfigurację portu szeregowego – prędkość transmisji 9600 baud.
5. Wgrać program na płytkę Arduino.

Po poprawnym wgraniu programu mikrokontroler jest gotowy do komunikacji z backendem.

4.3 Uruchomienie bazy danych (PostgreSQL)

W katalogu projektu /iot należy uruchomić kontenery Dockera:

docker-compose up -d

Baza danych PostgreSQL zostanie automatycznie utworzona z nazwą *plant_system*.

Dane przechowywane w bazie są zachowywane pomiędzy kolejnymi uruchomieniami systemu.

4.4 Uruchomienie backendu (Spring Boot)

W katalogu backendu należy uruchomić aplikację Spring Boot:

./mvnw spring-boot:run

Backend zostanie uruchomiony domyślnie na porcie 8080.

4.5 Dostęp do aplikacji użytkownika

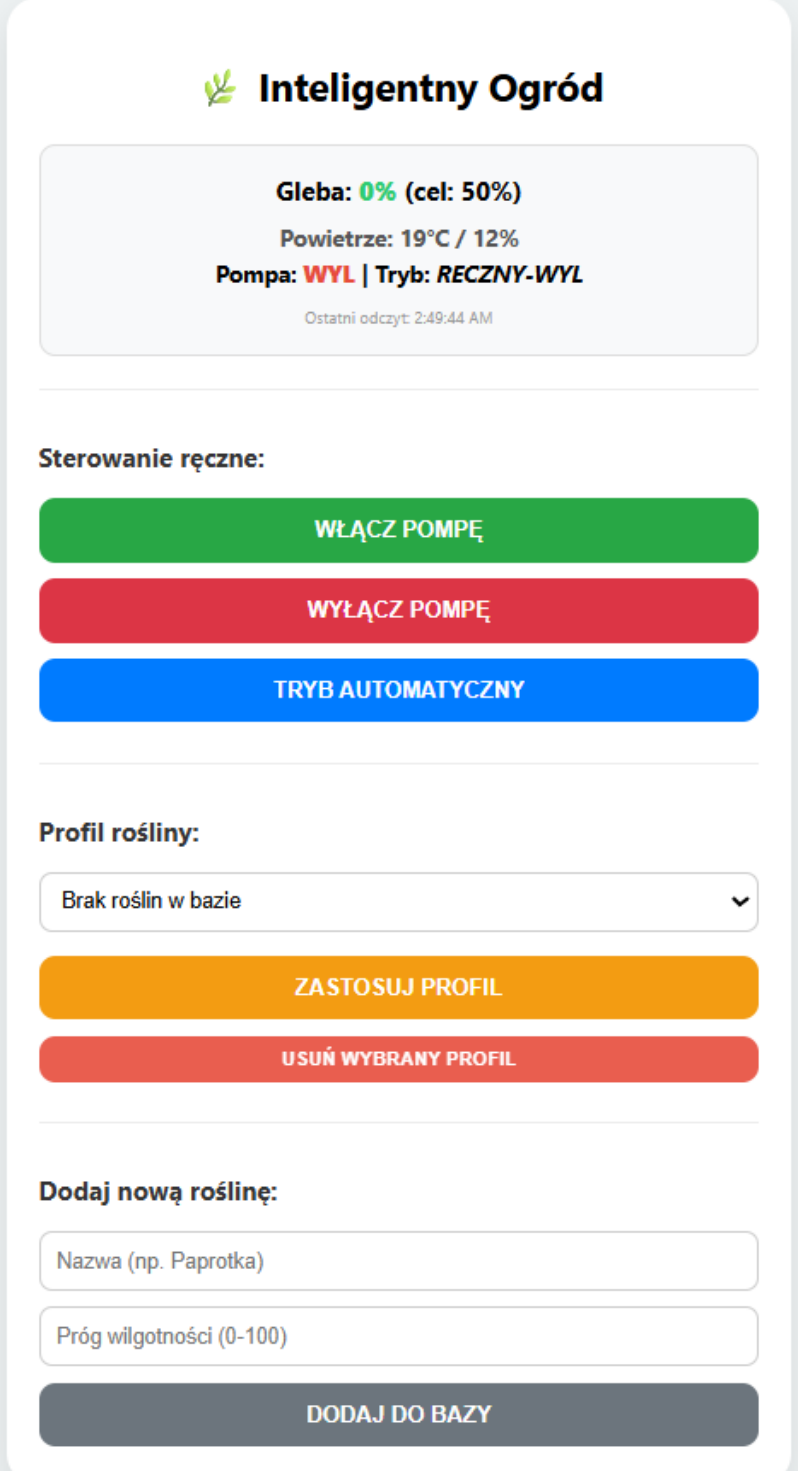
Po uruchomieniu wszystkich komponentów systemu aplikacja webowa jest dostępna poprzez przeglądarkę internetową pod adresem:

http://localhost:8080

Interfejs użytkownika został zaprojektowany z myślą o urządzeniach mobilnych i jest dostępny dla urządzeń znajdujących się w tej samej sieci lokalnej (LAN).

5. Interfejs użytkownika

Po uruchomieniu aplikacji dostępny jest panel webowy umożliwiający podgląd aktualnej wilgotności gleby, podgląd temperatury i wilgotności powietrza, sprawdzenie stanu pompy, zmianę trybu pracy systemu, zarządzanie profilami roślin.



Rys. 1. Widok aplikacji webowej Smart Garden (Inteligentny Ogród).

6. Tryby pracy systemu

System obsługuje trzy tryby pracy:

6.1 Tryb automatyczny (AUTO)

Pompa uruchamia się automatycznie, gdy wilgotność gleby spadnie poniżej zadanego progu.

6.2 Tryb ręczny – WŁĄCZONY

Pompa jest stale włączona, niezależnie od odczytów czujników.

6.3 Tryb ręczny – WYŁĄCZONY

Pompa pozostaje wyłączona, niezależnie od wilgotności gleby.

7. Profile roślin

System umożliwia zapis profili roślin w bazie danych.

Każdy profil zawiera nazwę rośliny, próg wilgotności gleby (0–100%).

Dostępne operacje:

- dodawanie nowego profilu,
- wybór profilu i zastosowanie jego parametrów,
- usuwanie profilu z bazy danych.

Zastosowanie profilu powoduje automatyczne przesłanie odpowiedniego progu wilgotności do Arduino.

8. Format danych i komunikacja

Arduino przesyła dane w formacie tekstowym:

DATA|*temperatura*|*wilgotność_powietrza*|*wilgotność_gleby*|*próg*|*stan_pompy*|*tryb*

Dane są aktualizowane co 2 sekundy po stronie Arduino i odświeżane co 1 sekundę w interfejsie webowym.

9. Typowe scenariusze użycia

- automatyczne podlewanie roślin domowych,
- ręczne sterowanie pompą podczas testów,
- szybka zmiana rośliny bez ingerencji w kod Arduino,
- podgląd warunków środowiskowych w czasie rzeczywistym.

10. Rozwiązywanie problemów

Brak danych w interfejsie

- sprawdzić połączenie USB z Arduino,
- upewnić się, że ustawiony jest właściwy port COM,
- sprawdzić czy backend jest uruchomiony.

Pompa nie reaguje

- sprawdzić tryb pracy systemu,
- sprawdzić połączenie pinów sterujących,
- upewnić się, że próg wilgotności został poprawnie ustawiony.

11. Podsumowanie

Smart Garden to modularny i prosty w obsłudze system IoT, który łączy sprzęt, backend i interfejs webowy w spójną całość. Projekt umożliwia łatwą rozbudowę o kolejne czujniki oraz funkcjonalności, zachowując czytelność i przejrzystość obsługi.

Rozdział 2. Techniczne aspekty.

12. Architektura systemu

System Smart Garden posiada architekturę trójwarstwową:

1. Warstwa sprzętowa (Arduino)
2. Warstwa backendowa (Spring Boot)
3. Warstwa prezentacji (aplikacja webowa)

Komunikacja pomiędzy warstwami realizowana jest poprzez:

- port szeregowy SERIAL (Arduino ↔ Backend),
- REST API (Backend ↔ Frontend).

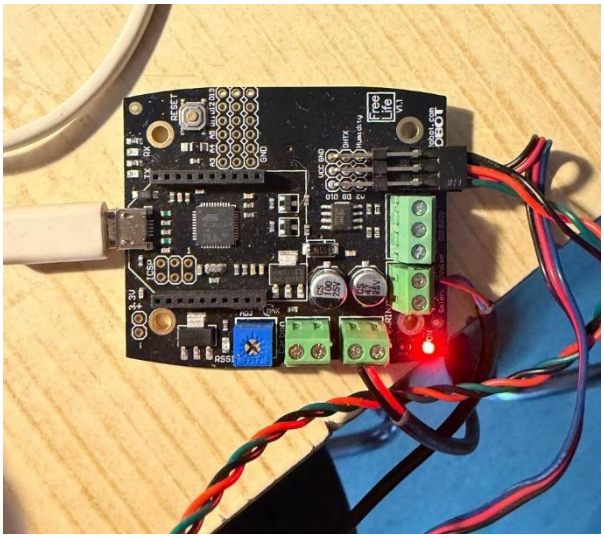
13. Warstwa sprzętowa – Arduino

13.1 Czujniki i elementy wykonawcze

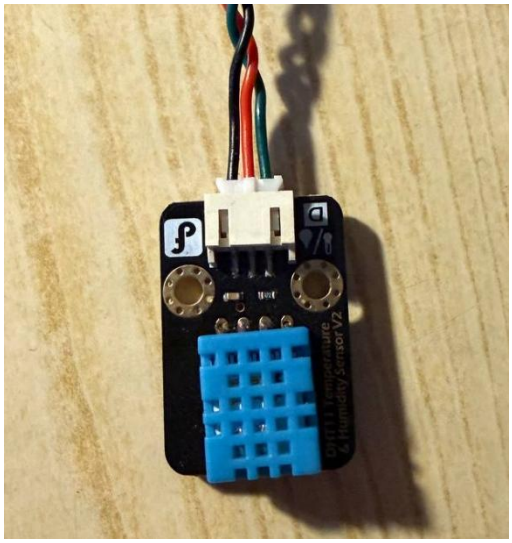
Arduino odpowiada za odczyt temperatury i wilgotności powietrza z czujnika DHT11, odczyt wilgotności gleby z czujnika analogowego, sterowanie pompą wodną poprzez wyjścia cyfrowe.

13.2 Przypisanie pinów

Element	Pin
DHT11	9
Czujnik wilgotności gleby	A2
Pompa – sterowanie	5, 6



Rys. 2. Płyta Arduino.



Rys. 3. Czujnik DHT11.

Rys. 4. Czujnik wilgotności gleby.

Rys. 5. Pompa.

13.3 Logika programu Arduino

Program Arduino działa w pętli głównej i realizuje następujące zadania cykliczny odczyt czujników co 2 sekundy, przetwarzanie komend przychodzących z portu szeregowego, podejmowanie decyzji o włączeniu lub wyłączeniu pompy, wysyłanie aktualnego stanu systemu do aplikacji backendowej.

13.4 Tryby pracy Arduino

Arduino obsługuje trzy tryby pracy:

- **AUTO** – automatyczne podlewanie na podstawie wilgotności gleby,
- **RECZNY-WL** – pompa zawsze włączona,
- **RECZNY-WYL** – pompa zawsze wyłączona.

Tryb pracy jest ustawiany na podstawie komend tekstowych odbieranych przez port szeregowy.

13.5 Format komunikacji szeregowej

Komendy przychodzące do Arduino:

MODE_AUTO, MODE_ON, MODE_OFF, SET_HUM:<wartość>.

Dane wysyłane przez Arduino:

DATA|temperatura|wilgotność_powietrza|wilgotność_gleby|próg|stan_pompy|tryb

14. Warstwa backendowa – Spring Boot

14.1 Rola backendu

Backend pełni rolę pośrednika komunikacji pomiędzy Arduino a frontendem, serwera REST API, warstwy dostępu do bazy danych.

14.2 Komunikacja z Arduino

Do komunikacji z Arduino wykorzystywana jest biblioteka jSerialComm.

Backend otwiera port COM przy starcie aplikacji, uruchamia osobny wątek do ciągłego odczytu danych, filtruje dane rozpoczynające się od prefiksu DATA|, przechowuje ostatni poprawny odczyt w pamięci aplikacji.

14.3 REST API

Backend udostępnia następujące endpointy:

Metoda	Endpoint	Opis
GET	/api/data	Pobranie ostatnich danych z Arduino
POST	/api/command	Wysłanie komendy do Arduino
GET	/api/plants	Lista profili roślin
POST	/api/plants	Dodanie profilu rośliny
DELETE	/api/plants/{id}	Usunięcie profilu

14.4 Zarządzanie stanem aplikacji

Backend przechowuje aktualny stan czujników w pamięci, nie zapisuje danych pomiarowych w bazie (tylko konfigurację), zapewnia szybki dostęp do ostatniego odczytu dla frontendowej aplikacji.

15. Baza danych

15.1 Technologia

- PostgreSQL 15
- uruchamiana w kontenerze Docker

15.2Struktura danych

Tabela plants zawiera identyfikator (id), nazwę rośliny (name), próg wilgotności (threshold).

15.3 Trwałość danych

Baza danych zachowuje dane pomiędzy restartami systemu, umożliwia wielokrotne użycie wcześniej zdefiniowanych profili roślin.

16. Warstwa frontendowa

16.1 Technologia

- HTML5
- CSS3
- JavaScript (Fetch API)

16.2 Zasada działania

Frontend cyklicznie pobiera dane z /api/data, interpretuje dane tekstowe przesyłane przez backend, umożliwia wysyłanie komend sterujących, zarządza profilami roślin poprzez REST API.

16.3 Responsywność

Interfejs jest zoptymalizowany pod urządzenia mobilne, dostosowany do obsługi dotykowej, działa poprawnie w sieci lokalnej.

17. Przepływ danych w systemie

1. Arduino odczytuje czujniki.
2. Arduino wysyła dane przez port szeregowy.
3. Backend odbiera dane i zapisuje ostatni stan.

4. Frontend pobiera dane z API.
5. Użytkownik wysyła komendę z interfejsu.
6. Backend przekazuje komendę do Arduino.

18. Możliwości rozbudowy systemu

System Smart Garden może zostać rozbudowany o dodatkowe czujniki (światło, pH gleby), harmonogram podlewania, zapisywanie historii pomiarów, obsługę wielu stref podlewania, autoryzację użytkowników.

19. Podsumowanie techniczne

Smart Garden jest modularnym systemem IoT, który separuje warstwy odpowiedzialności.