Uniwersytet Zielonogórski

System Automatycznego Podlewania Roślin

Julia Stróżniak
112125@Stud.uz.zgora.pl

Wydział Nauk Inżynieryjno-Technicznych

Prowadzący
mgr inż. Norbert Łukaniszyn

12 stycznia 2026

Spis treści

Rozdział 1

Wstęp

Celem projektu jest zaprojektowanie i zbudowanie automatycznego systemu podlewania roślin sterowanego przez mikrokontroler Arduino Leonardo. Urządzenie ma na bieżąco mierzyć wilgotność gleby za pomocą czujnika i automatycznie włączać pompę wodną, gdy poziom wilgotności spadnie poniżej ustalonego progu.

Projekt pozwala zapoznać się z obsługą czujników analogowych, sterowaniem elementami wykonawczymi (pompą poprzez przekaźnik) oraz z podstawami budowy prostych systemów automatyki opartych na mikrokontrolerach. Dodatkowo zastosowany czujnik poziomu wody zabezpiecza system przed pracą przy pustym zbiorniku, a dioda RGB sygnalizuje aktualny stan urządzenia.

Rozdział 2

Cele projektu

- Zbudowanie działającego układu elektronicznego do automatycznego podlewania roślin.
- Opracowanie oprogramowania mierzącego wilgotność gleby i sterującego pompką wodną.
- Ćwiczenie pracy z czujnikami i elementami wykonawczymi w środowisku Arduino.

Rozdział 3

Wymagania funkcjonalne

3.1 Podstawowe funkcje

- Pomiar wilgotności gleby z czujnika.
- Automatyczne uruchamianie pompki, gdy gleba jest zbyt sucha.
- Zatrzymanie pompki po określonym czasie.
- Sygnalizacja stanu pracy systemu za pomocą diody lub zestawu diod.

3.2 Rozszerzone funkcje (opcjonalne)

- Pomiar temperatury oraz natężenia oświetlenia w celu uzyskania pełniejszego obrazu danych.
- Monitorowanie poziomu wody w zbiorniku.
- Wysyłanie danych do komputera lub urządzeń mobilnych za pomocą interfejsu USB lub Bluetooth.
- Panel WWW lub aplikacja umożliwiająca wyświetlanie danych dotyczących stanu rośliny.

Rozdział 4

Architektura systemu

System składa się z następujących elementów:

4.1 Mikrokontroler

- Arduino Leonardo (0 zł)

4.2 Czujniki

- Czujnik pojemnościowy wilgotności gleby (4 zł)
- Fotorezystor – pomiar natężenia światła (opcjonalnie) (4 zł)
- Czujnik temperatury DHT-22 (opcjonalnie) (6 zł)
- Czujnik poziomu wody w zbiorniku – pływak (opcjonalnie) (0 zł)

4.3 Elementy wykonawcze

- Pompka wodna (4 zł)
- Moduł przekaźnika (4 zł)

4.4 Elementy sygnalizacyjne

- Dioda RGB (0 zł)

4.5 Komunikacja i przechowywanie danych

- Moduł Bluetooth HC-05 (opcjonalnie) (17 zł)
- Moduł czytnika kart SD (opcjonalnie) (0 zł)

4.6 Elementy pomocnicze i ochronne

- Przewody połączeniowe (0 zł)
- Płytka stykowa (breadboard) (0 zł)

- Rezystory (0 zł)
- Wężyk silikonowy (4 zł)
- Pojemnik na wodę (0 zł)

Koszt projektu

Przewidywany koszt realizacji projektu wynosi około 50 zł.

Rozdział 5

Etapy realizacji

5.1 Etap 1 – Analiza i projekt koncepcyjny (21.10 – 27.10.2025)

- Określenie celu oraz zasad działania systemu.
- Dobór czujników, pompki oraz elementów pomocniczych.
- Opracowanie wstępnych schematów połączeń.
- Przygotowanie wstępnego projektu interfejsu (prosta strona do wizualizacji danych).

5.2 Etap 2 – Montaż układu (28.10 – 10.11.2025)

- Przygotowanie płytki stykowej oraz przewodów połączeniowych.
- Podłączenie czujnika wilgotności gleby do mikrokontrolera Arduino.
- Podłączenie modułu przekaźnika oraz pompki wodnej.
- Dodanie diod LED do sygnalizacji stanu pracy systemu.
- Sprawdzenie poprawności połączeń oraz zasilania.

5.3 Etap 3 – Oprogramowanie mikrokontrolera (11.11 – 24.11.2025)

- Napisanie programu do odczytu wilgotności gleby z czujnika.
- Implementacja warunku uruchamiania pompki przy zbyt niskiej wilgotności.
- Dodanie przerw czasowych między pomiarami w celu stabilizacji odczytów.
- Testy działania programu z wykorzystaniem narzędzia Serial Monitor.
- (Opcjonalnie) Rozpoczęcie prac nad wizualizacją danych lub symulacją systemu.

5.4 Etap 4 – Testy funkcjonalne (25.11 – 08.12.2025)

- Sprawdzenie działania pompki w zależności od poziomu wilgotności gleby.
- Testy systemu w różnych warunkach (gleba sucha, wilgotna, bardzo mokra).
- Obserwacja zachowania systemu w dłuższym okresie czasu.
- (Opcjonalnie) Testy działania strony lub wizualizacji danych w czasie rzeczywistym.

5.5 Etap 5 – Rozszerzenia funkcjonalne (09.12 – 19.12.2025)

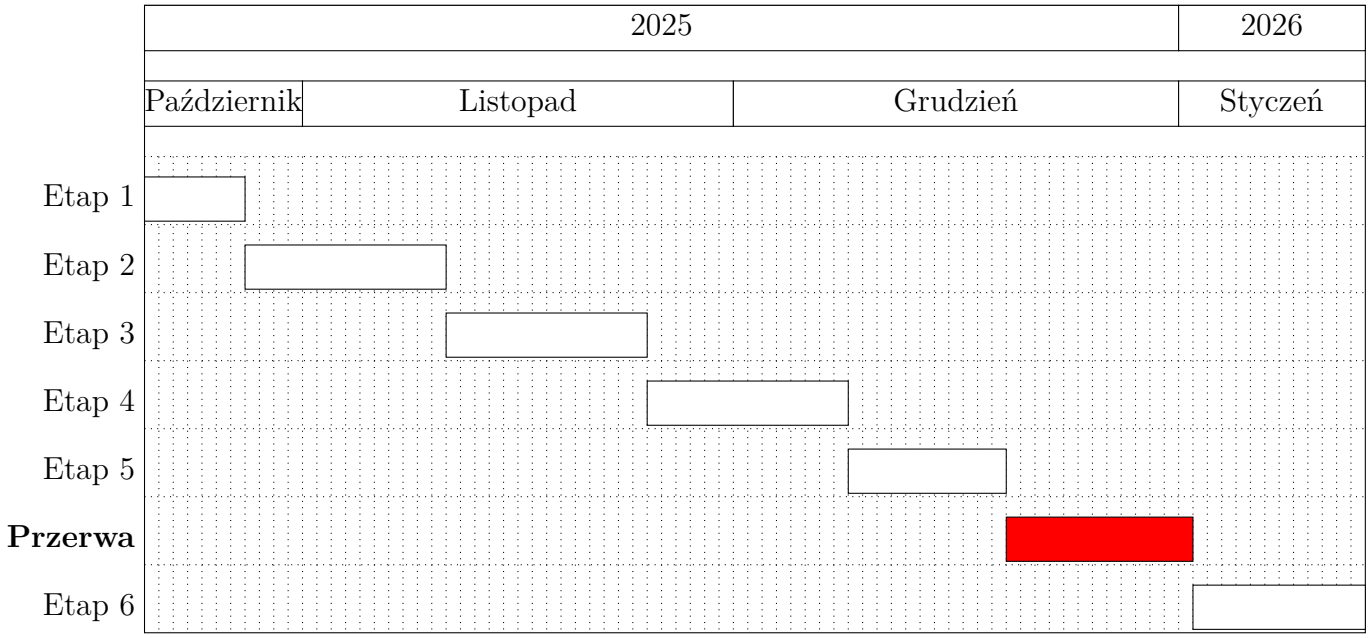
- Dodanie czujnika światła, temperatury lub poziomu wody (opcjonalnie).
- Implementacja komunikacji Bluetooth lub Wi-Fi (opcjonalnie).
- Prace nad wizualizacją danych lub symulacją systemu (opcjonalnie).
- Testy poprawności działania systemu po wprowadzeniu rozszerzeń.

5.6 Etap 6 – Dokumentacja i prezentacja projektu (02.01 – 13.01.2026)

- Przygotowanie opisów działania systemu oraz schematów połączeń.
- Zebranie wyników testów i obserwacji.
- Opracowanie końcowej dokumentacji projektu oraz prezentacji.

Rozdział 6

Plan realizacji (propozycja harmonogramu)



- **21.10 – 27.10.2025** – Etap 1: Analiza i projekt koncepcyjny (1 tydzień)
- **28.10 – 10.11.2025** – Etap 2: Montaż układu (2 tygodnie)
- **11.11 – 24.11.2025** – Etap 3: Oprogramowanie mikrokontrolera (2 tygodnie)
- **25.11 – 08.12.2025** – Etap 4: Testy funkcjonalne (2 tygodnie)
- **09.12 – 19.12.2025** – Etap 5: Rozszerzenia funkcjonalne (1,5 tygodnia)
- **20.12.2025 – 01.01.2026** – Przerwa świąteczna (2 tygodnie)
- **02.01 – 13.01.2026** – Etap 6: Dokumentacja i prezentacja projektu (2 tygodnie)

Rozdział 7

Schematy i diagramy połączeń

7.1 Tabela systemu podłączeń na Arduino

- Tabela systemu podłączeń na Arduino

PIN	NAZWA	ELEMENT	DODATKOWE INFORMACJE
A0	SOIL_MAIN_PIN	czujnik pojemnościowy wilgotności gleby	główny, roślinka 2
A1	SOIL_OPT1_PIN	czujnik pojemnościowy wilgotności gleby	opcjonalny, roślinka 1
A2	SOIL_OPT2_PIN	czujnik pojemnościowy wilgotności gleby	opcjonalny, roślinka 3
A3	SOIL_OPT3_PIN	czujnik pojemnościowy wilgotności gleby	opcjonalny, roślinka 4
A4	LDR_PIN	fotorezystor	z rezystorem 10k
D0	–	moduł bluetooth (COM15)	TX → pin 0 (RX1)
D1	–		RX → pin 1 (TX1)
D4	BT_STATE_PIN		STATE
D2	WATER_LEVEL_PIN	czujnik poziomu wody	+GND
D3	SD_CS_PIN	moduł karty SD na SPI	połączenie z ICSP Leonardo
D6	DHT_PIN	czujnik temperatury	DHT22, z rezystorem 10 kΩ między VCC i DATA
D7	RELAY_MAIN_PIN	4-kanałowy moduł przekaźnika	główny (IN1-M1), roślinka 3
D8	RELAY_OPT1_PIN		opcjonalny (IN2-M2), roślinka 4
D9	RELAY_OPT2_PIN		opcjonalny (IN3-M3), roślinka 2
D10	RELAY_OPT3_PIN		opcjonalny (IN4-M4), roślinka 1
D11	LED_R_PIN	dioda RGB LED	czerwony, (z rezystorami 220 Ω)
D12	LED_G_PIN		zielony, (z rezystorami 220 Ω)
D13	LED_B_PIN		niebieski, (z rezystorami 220 Ω)

7.2 Schemat elektryczny

7.2.1 Czujniki pojemnościowe wilgotności gleby

SOIL_MAIN_PIN (roślina 2):

- OUT → A0
- VCC → 5V
- GND → GND

SOIL_OPT1_PIN (roślina 1):

- OUT → A1
- VCC → 5V
- GND → GND

SOIL_OPT2_PIN (roślina 3):

- OUT → A2
- VCC → 5V
- GND → GND

SOIL_OPT3_PIN (roślina 4):

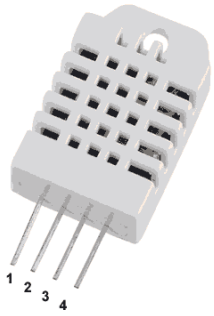
- OUT → A3
- VCC → 5V
- GND → GND

7.2.2 Fotorezystor (LDR)

- LDR → 5V
- LDR → A4
- A4 → rezystor 10 kΩ → GND

7.2.3 Czujnik temperatury DHT22

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



- VCC → rezystor 10 kΩ → 5V
- DATA → D6
- DATA ↔ VCC przez rezystor 10 kΩ (pull-up)
- GND → GND

7.2.4 Czujnik poziomu wody (pływak)

- Jeden styk → D2
- Drugi styk → GND

7.2.5 Dioda RGB

- Anoda → 5V
- Katoda R → rezystor 220 Ω → D11
- Katoda G → rezystor 220 Ω → D12
- Katoda B → rezystor 220 Ω → D13

7.2.6 Moduł przekaźników

- VCC → 5V
- GND → GND

Kanały przekaźników:

- IN1 → D7 (pompka 3)
- IN2 → D8 (pompka 4)
- IN3 → D9 (pompka 2)
- IN4 → D10 (pompka 1)

Połączenia styków:

- NO (lewy) → czarny przewód pompy
- COM (środkowy) → GND
- NC (prawy) – brak połączenia

7.2.7 Pompki wodne

- Czarny przewód → NO przekaźnika
- Czerwony przewód → 5V

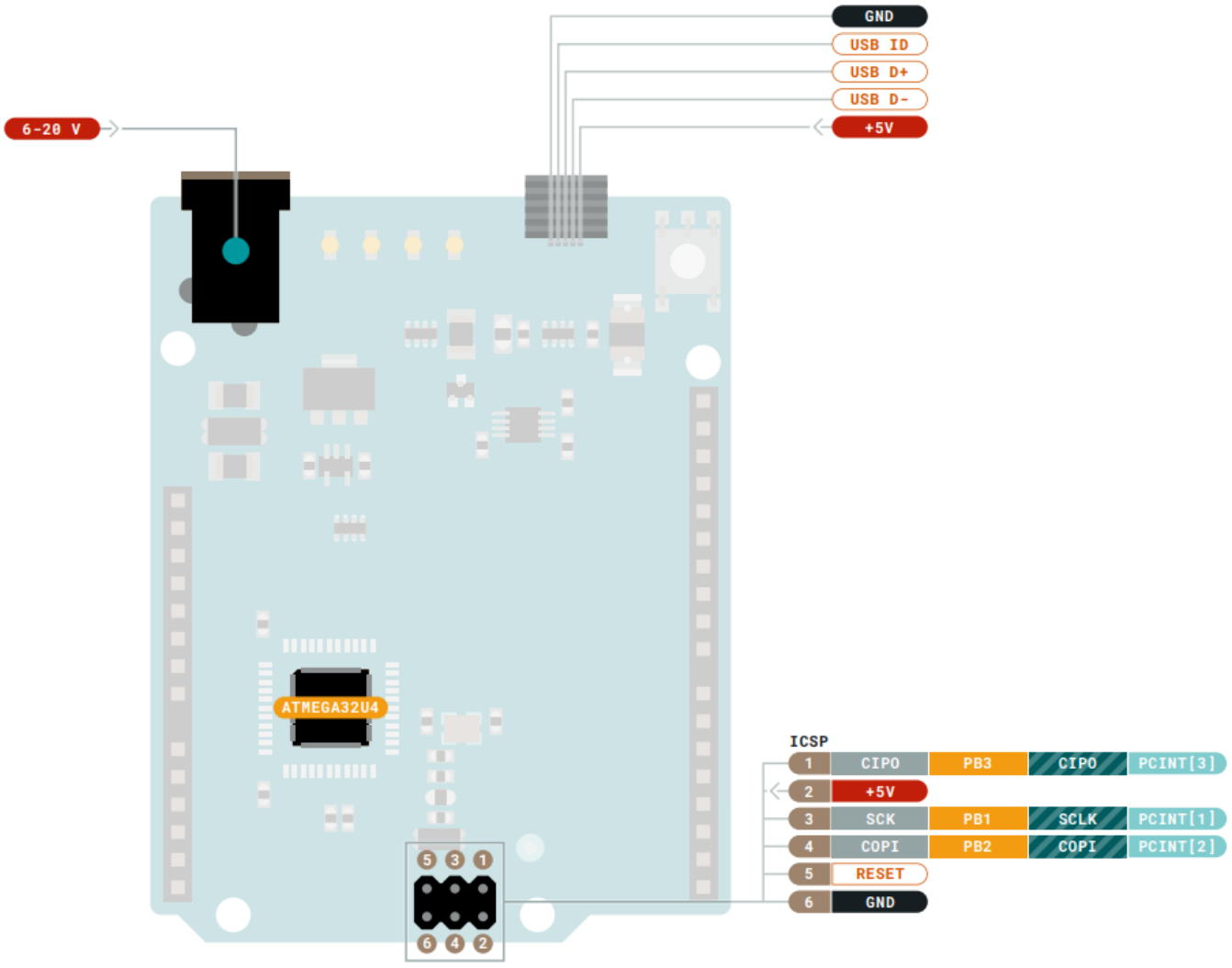
7.2.8 Bluetooth HC-05

- TX → D0 (RX1)
- RX → D1 (TX1)
- VCC → 5V
- GND → GND
- STATE → D4

7.2.9 Moduł karty SD (SPI)

- VCC → ICSP 5V (pin 2)
- GND → ICSP GND (pin 6)
- CS → D3
- MISO → ICSP CIPO (pin 1)
- MOSI → ICSP COPI (pin 4)
- SCK → ICSP SCK (pin 3)

7.3 Schemat pomocniczy ICSP w Arduino Leonardo



7.4 Program odpowiadający za obsługę czujników

7.4.1 Schemat blokowy

Znajduje się w załączniku.

7.4.2 Opis tekstowy algorytmu

1. Start programu i rozpoczęcie cyklu pracy.
2. Odczyt warunków środowiskowych:
   - odczytaj temperaturę powietrza,
   - odczytaj wilgotność powietrza,
   - odczytaj jasność.
3. Wyślij ramkę `ENV` (z danymi środowiskowymi).
4. Wejdź w pętlę po roślinach.
5. Wybierz aktualną roślinkę.
6. Odczytaj wilgotność gleby dla tej roślinki.
7. Sprawdź warunek: czy wilgotność gleby jest mniejsza od progu?
   - **Jeśli NIE** (gleba jest wystarczająco wilgotna):
     - ustaw `needWater = 0` (roślinka nie potrzebuje wody),
     - ustaw `waterState = -1` (brak podlewania / nie dotyczy),
     - ustaw `watered = 0` (nie była podlewana w tym cyklu),
     - wyślij ramkę `PLANT` z danymi roślinki.
   - **Jeśli TAK** (gleba za sucha):
     (a) Odczytaj czujnik poziomu wody.
     (b) Sprawdź: czy woda jest dostępna?
     - **Jeśli NIE** (brak wody):
       * włącz czerwoną LED,
       * ustaw `needWater = 1` (potrzebuje wody),
       * ustaw `waterState = 0` (nie podlano, bo brak wody),
       * ustaw `watered = 0`,
       * wyślij ramkę `PLANT`.
     - **Jeśli TAK** (woda dostępna):
       * włącz żółtą LED,
       * włącz pompę,
       * podlewaj przez czas `WATERING_TIME_MS`,
       * wyłącz pompę,
       * wyłącz LED,
       * mignij zieloną LED 3 razy (sygnalizacja sukcesu),
       * ustaw `needWater = 1`,
       * ustaw `waterState = 1` (podlewanie wykonane),
       * ustaw `watered = 1`,
       * wyślij ramkę `PLANT`.
8. Sprawdź: czy jest kolejna roślinka?
   - jeśli tak → wróć do wyboru następnej roślinki i powtórz pętlę,
   - jeśli nie → przejdź dalej.
9. Tryb czuwania po obsłużeniu wszystkich roślinek:
   - przejdź w tryb czuwania,
   - odczekaj `IDLE_DELAY_MS`.
10. Powrót do początku cyklu i wykonanie wszystkiego od nowa.

7.5 Program odpowiadający za nasłuchiwanie ramek

7.5.1 Schemat blokowy

Znajduje się w załączniku.

7.5.2 Opis tekstowy algorytmu nasłuchiwania ramek

1. Program uruchamia się, ustawia stałe konfiguracyjne (port, prędkość transmisji, parametry klastrowania ramek) oraz tworzy katalog `data_logs`, jeśli nie istnieje.
2. Otwiera połączenie szeregowe z modułem Bluetooth/Arduino (COM15, 9600) i wypisuje informację o lokalizacji zapisywanych logów.
3. Inicjuje zmienne pomocnicze:
   - `last_arrival = None` — czas nadejścia poprzedniej ramki,
   - `_pending_frames = []` — bufor ramek należących do jednego klastra.
4. Wchodzi w nieskończoną pętlę odbioru danych:
   - odczytuje jedną linię z portu szeregowego,
   - jeśli linia jest pusta — pomija ją i czyta kolejną,
   - próbuje sparsować linię funkcją `parse_line()`,
   - jeśli linia nie zaczyna się od znaku `@` — traktuje ją jako zwykły tekst z Arduino i ignoruje,
   - jeśli linia zaczyna się od `@` — tworzy strukturę danych zawierającą typ ramki (`kind`) oraz pary klucz–wartość.
5. Dla poprawnie sparsowanej ramki zapisuje czas jej nadejścia `arrival = datetime.now()` i sprawdza jej typ:
   - jeśli `kind == "ENV"` — wywołuje funkcję `handle_env()`, która wypisuje dane środowiskowe i zwraca dane,
   - jeśli `kind == "PLANT"` — wywołuje funkcję `handle_plant()`, która wypisuje status rośliny i zwraca dane,
   - jeśli typ jest inny — wypisuje informację `UNKNOWN`, ustawia ramce znacznik czasu `arrival`, zapisuje ją bezpośrednio do pliku i wraca do pętli.
6. Dla ramek typu `ENV` oraz `PLANT` program stosuje klastrowanie:
   - jeśli jest to pierwsza ramka (`last_arrival is None`) — rozpoczyna nowy klaster i dodaje ramkę do bufora,
   - w przeciwnym razie oblicza odstęp czasu `delta` od poprzedniej ramki:
     - jeśli `delta` $\leqslant$ `30 s` — dodaje ramkę do bieżącego klastra,
     - jeśli `delta` $>$ `30 s` — zapisuje bufor bieżącego klastra do plików, czyści bufor i rozpoczyna nowy klaster od aktualnej ramki.
7. Po obsłużeniu każdej ramki ustawia `last_arrival = arrival`.
8. Podczas zapisu klastra program:
   - wyznacza czas nadejścia najnowszej ramki w klastrze,
   - jeśli ramki zawierają pole `t` (czas z Arduino w ms) — wylicza znaczniki czasu starszych ramek względem najnowszej,

- jeśli pole `t` nie występuje — przypisuje ramkom znaczniki czasu co 1 sekundę w celu zachowania kolejności,
- zapisuje każdą ramkę jako rekord JSON do pliku dziennego:
  - `env_YYYY-MM-DD.jsonl` dla ramek `ENV`,
  - `plants_YYYY-MM-DD.jsonl` dla ramek `PLANT`,
  - `other_YYYY-MM-DD.jsonl` dla pozostałych typów.

9. Po przerwaniu programu przez użytkownika (Ctrl+C):
   - zapisuje pozostałe w buforze ramki (ostatni klaster),
   - wypisuje komunikat o zakończeniu pracy,
   - zamyka połączenie szeregowe i kończy działanie programu.

Rozdział 8

Technologie

- **C++ (Arduino)**
  - `DHT.h`
  - `SPI.h`
  - `SD.h`
- **Python**
  - `Flask`
  - `serial`
- **JavaScript**
- **HTML**
- **CSS**

Rozdział 9

Jak używać projektu oraz jego punkty dopasowań

Projekt został zaprojektowany jako system modułowy, umożliwiający łatwe uruchomienie oraz późniejsze dostosowanie do indywidualnych potrzeb użytkownika lub warunków środowiskowych.

9.1 Sposób użycia projektu

Aby uruchomić system, należy poprawnie wykonać połączenia elektryczne zgodnie z przygotowanym schematem oraz wgrać oprogramowanie sterujące do mikrokontrolera Arduino. Po podłączeniu zasilania układ automatycznie rozpoczyna cykliczną pracę polegającą na pomiarze warunków środowiskowych, kontroli wilgotności gleby oraz podejmowaniu decyzji o podlewaniu roślin.

Komunikacja z komputerem odbywa się bezprzewodowo za pomocą modułu Bluetooth. Program nasłuchujący uruchomiony na komputerze otwiera połączenie szeregowe i odbiera ramki przesyłane przez Arduino, a następnie zapisuje je w postaci plików JSONL. Dane te mogą być dodatkowo przetwarzane i prezentowane za pomocą aplikacji webowej.

9.2 Punkty dopasowań projektu

Zaprojektowany system umożliwia wprowadzenie zmian konfiguracyjnych bez konieczności ingerencji w jego architekturę. Najważniejsze punkty dopasowań obejmują:

- **Parametry komunikacji** – możliwość zmiany portu szeregowego.

```
PORT = "COM15"   # Miejsce na port HC-05
BAUDRATE = 9600
```

Listing 9.1: Przypisanie portu HC-05 w listenerze

- **Częstotliwość pracy systemu** – regulacja czasu czuwania pomiędzy kolejnymi cyklami pomiarowymi. Zmiany obejmują zarówno program nasłuchujący, jak i zarządzający czujnikami.

```
# oczekiwany odstęp między kolejnymi ramkami (w sekundach), używany przy
    przeliczaniu pola 't' z Arduino
EXPECTED_FRAME_INTERVAL_SECONDS = 60 * 60  # ~60 minut między pomiarami
# dopuszczalna odchyłka przy dopasowywaniu (sekundy)
FRAME_INTERVAL_SLACK_SECONDS = 60
# jeśli kolejne ramki przychodzą w krótszym odstępie (sekund), traktujemy je
    jako jedną "paczkę"/klaster
CLUSTER_MAX_INTERARRIVAL_SECONDS = 30
```

Listing 9.2: Konfiguracja przyjmowania ramek

```
const unsigned long IDLE_DELAY_MS = 60UL * 60UL * 1000UL;   // czas czuwania
    między cyklami [1 h]
```

Listing 9.3: Ustawienie opóźnienia pomiędzy cyklami

- **Progi wilgotności gleby** – dostosowanie wartości granicznych decydujących o konieczności podlewania roślin.

```
const int SOIL2_DRY       = 551;  // odczyt dla suchej ziemi
const int SOIL2_WET       = 220;  // odczyt dla wody
const int SOIL2_THRESHOLD = 10;   // próg wilgotności (%) poniżej którego
    podlewamy
```

Listing 9.4: Kalibracja czujnika gleby

- **Czas podlewania** – zmiana czasu pracy pomp wodnych w pojedynczym cyklu.

```
const unsigned long WATERING_TIME_MS = 4000;   // jak długo podlewać (ms)
```

Listing 9.5: Dostosowanie czasu podlewania

- **Liczba obsługiwanych roślin** – możliwość łatwego dodania nowej rośliny. Po skonfigurowaniu portów wystarczy użycie funkcji:

```
processPlant(
  SOIL_OPT2_PIN,    // pin czujnika wilgotności
  RELAY_MAIN_PIN,   // przekaźnik
  "Roślinka␣3",     // nazwa rośliny
  SOIL2_DRY,        // parametry kalibracji czujnika wilgotności
  SOIL2_WET,
  SOIL2_THRESHOLD
);
```

Listing 9.6: Dodanie nowej rośliny

Dzięki powyższym możliwościom projekt może być łatwo rozbudowany, skalowany lub dostosowany do innych scenariuszy zastosowań, takich jak większa liczba roślin, inne czujniki lub alternatywne metody komunikacji.

Rozdział 10

Testy i wyniki

10.1 Czujniki pojemnościowe wilgotności gleby

- **Test suchej gleby** — gdy gleba była sucha (niepodlewana od kilku tygodni), odczyt wynosił ok. **550**.
- **Test świeżo podlanej gleby** — po podlaniu i przy odpowiednio bliskim umieszczeniu czujnika odczyt wynosił ok. **400**.

Tabela 10.1: Wyniki pomiarów czujników wilgotności gleby

Roślina	**Sucha gleba**	**Świeżo podlana**	**Woda**
Roślina 1	552	440	213
Roślina 2	558	518	230
Roślina 3	551	445	220
Roślina 4	556	348	215
Średnia	554,25	437,75	219,5

10.2 Fotorezystor

- **Test w ciemności** — zgaszenie światła w pokoju w nocy spowodowało odczyt **6**.
- **Test w ciągu dnia** — w trakcie dnia fotorezystor rejestrował wartość ok. **800–900**.
- **Test przy oświetleniu nocnym** — w nocy przy zapalonym świetle odczyt wynosił ok. **552**.

10.3 Moduł Bluetooth

- **Test połączenia z komputerem** — wykonany poprzez połączenie z modułem HC-05 i otwarcie portu `COM15`. Przygotowano skrypt w Pythonie do testowania i pobierania informacji.

```
Połączono z COM15. Czekam na ramki @ENV / @PLANT... (Ctrl+C aby wyjść)

Logi JSON będą zapisywane w: C:\Users\julka\Desktop\Projekty\SmartPlant_dashboard\data_logs

[ENV] OK  | temp=16.9°C, hum=80.7%, light=549
[PLANT Roślinka 1] GLEBA OK | soil=32% >= threshold=20%, soilRaw=464
[PLANT Roślinka 2] GLEBA OK | soil=79% >= threshold=5%, soilRaw=322
[PLANT Roślinka 3] GLEBA OK | soil=8% >= threshold=5%, soilRaw=547
[PLANT Roślinka 4] GLEBA OK | soil=58% >= threshold=10%, soilRaw=466
```

- **Test buforowania na SD przy braku Bluetooth** — gdy moduł Bluetooth jest podłączony, wysyła dodatkowe komunikaty na `Serial` z informacją o statusie. Po jego wyłączeniu komunikat ulega zmianie, a odczyty są zapisywane na karcie SD.

```
LIVE->BT: @PLANT;name=Roślinka 4;soilRaw=486;soil=46;threshold=10;needWater=0;waterState=-1;watered=0;t=1717
Czuwanie...
```

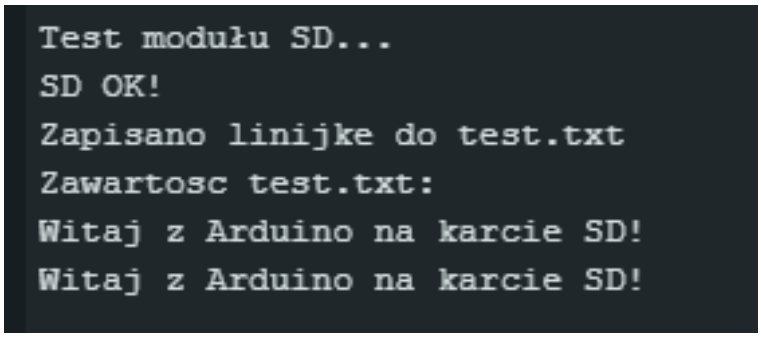
```
-------------------------------------
NOWY CYKL – odczyt warunków otoczenia:
Temperatura: 19.50 °C   Wilgotność powietrza: 46.50 %
Jasność (LDR): 3
SD BUF += @ENV;ok=1;temp=19.50;hum=46.50;light=3;t=30620
-------[Roślinka 1]-------
[Roślinka 1] Surowy odczyt gleby: 449   Wilgotność gleby: 37 %
[Roślinka 1] Gleba jest wystarczająco wilgotna. Nie podlewam.
SD BUF += @PLANT;name=Roślinka 1;soilRaw=449;soil=37;threshold=20;needWater=0;waterState=-1;watered=0;t=30638
-------[Roślinka 2]-------
[Roślinka 2] Surowy odczyt gleby: 393   Wilgotność gleby: 55 %
[Roślinka 2] Gleba jest wystarczająco wilgotna. Nie podlewam.
SD BUF += @PLANT;name=Roślinka 2;soilRaw=393;soil=55;threshold=5;needWater=0;waterState=-1;watered=0;t=30656
-------[Roślinka 3]-------
[Roślinka 3] Surowy odczyt gleby: 539   Wilgotność gleby: 13 %
[Roślinka 3] Gleba jest wystarczająco wilgotna. Nie podlewam.
SD BUF += @PLANT;name=Roślinka 3;soilRaw=539;soil=13;threshold=5;needWater=0;waterState=-1;watered=0;t=30673
-------[Roślinka 4]-------
[Roślinka 4] Surowy odczyt gleby: 466   Wilgotność gleby: 58 %
[Roślinka 4] Gleba jest wystarczająco wilgotna. Nie podlewam.
SD BUF += @PLANT;name=Roślinka 4;soilRaw=466;soil=58;threshold=10;needWater=0;waterState=-1;watered=0;t=30694
Czuwanie...
```

10.4 Czujnik poziomu wody (pływak)

- **Test ruchu pływaka** — pływak porusza się swobodnie w górę i w dół.
- **Test braku wody** — gdy brak wody lub poziom jest zbyt niski, aby unieść pływak, czujnik nadaje sygnał **0**.
- **Test obecności wody** — gdy poziom wody jest wystarczający i pływak unosi się, czujnik nadaje sygnał **1**.

10.5 Moduł karty SD

- **Test połączenia oraz zapisu/odczytu** — sprawdzono inicjalizację modułu oraz zapis i odczyt prostego pliku na karcie SD.



10.6 Czujnik temperatury (DHT22)

- **Test dokładności** — czujnik DHT22 oraz termometr szpilkowy umieszczono w tym samym miejscu i pozostawiono na ok. **10 min** w celu wyrównania temperatur. DHT22 wskazywał temperaturę o ok. **0,8°C** niższą.

10.7 4-kanałowy moduł przekaźnika

- **Test pojedynczego kanału** — wymuszenie stanu `LOW` aktywuje przekaźnik (słyszalne kliknięcie), a pompa rozpoczyna pracę.

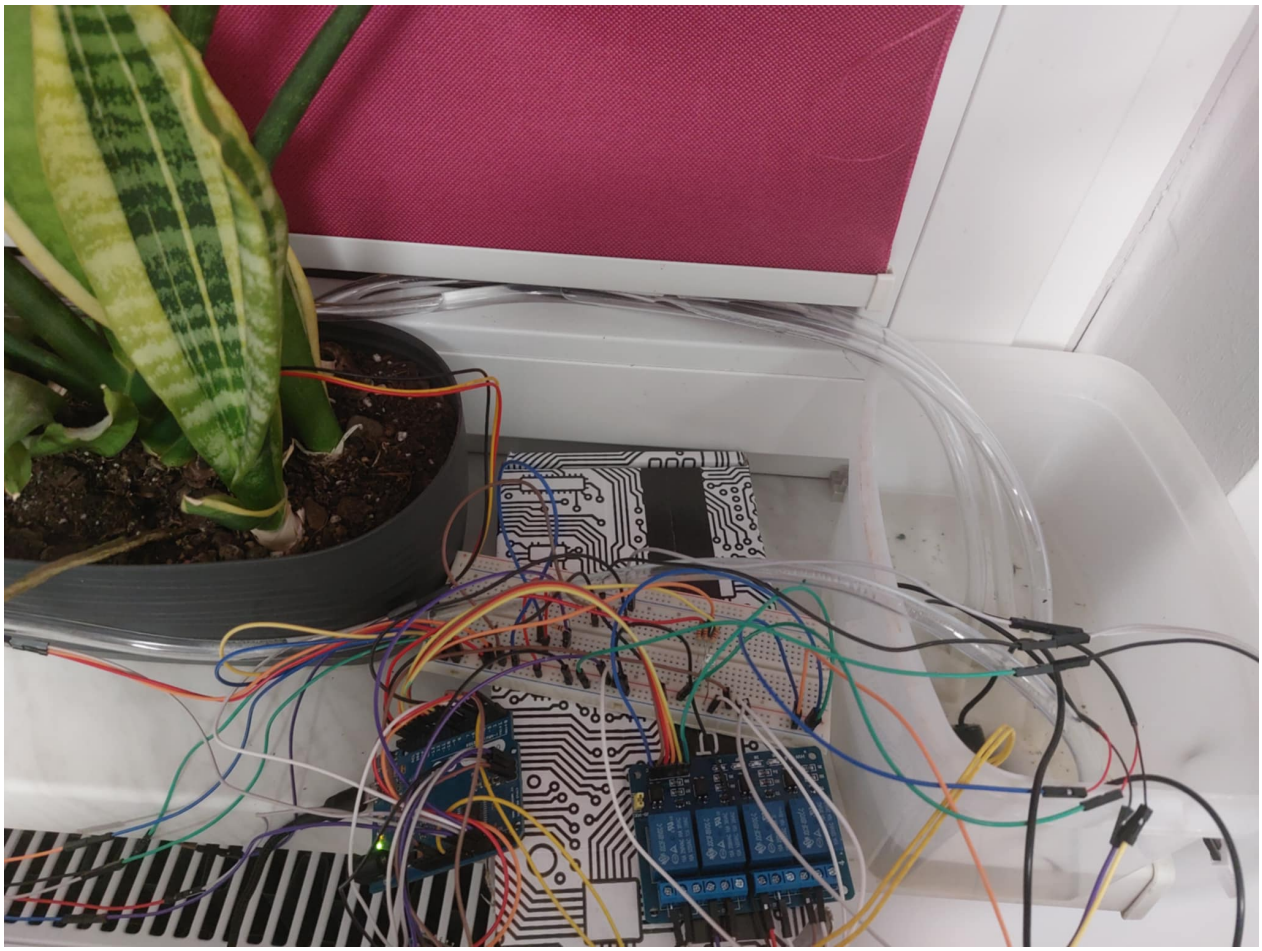
10.8 Dioda RGB LED

- **Test świecenia** — ze względu na wspólną anodę dioda świeci po podaniu sygnału `LOW`.
- **Test gaszenia** — po podaniu sygnału `HIGH` dioda gaśnie.

Rozdział 11

Działanie algorytmu podlewania roślin

Dla potrzeb automatycznego podlewania roślin system działa cyklicznie. W każdym cyklu urządzenie zbiera dane środowiskowe, a następnie przechodzi po wszystkich roślinach podłączonych do systemu i decyduje, czy wymagana jest akcja podlewania. Decyzja zależy od porównania aktualnej wilgotności gleby z progiem przypisanym do danej rośliny.



11.1 Przebieg pojedynczego cyklu

W każdym przebiegu programu wykonywane są kroki:

- odczyt temperatury powietrza,
- odczyt wilgotności powietrza,
- odczyt jasności,
- wysłanie ramki `ENV`,
- iteracja po roślinach:
  - odczyt wilgotności gleby,
  - sprawdzenie warunku podlewania,
  - ewentualne sprawdzenie poziomu wody i uruchomienie pompy,
  - wysłanie ramki `PLANT` dla każdej rośliny,
- przejście w tryb czuwania (`IDLE_DELAY_MS`) i ponowienie cyklu.

Wszystko to jest zamknięte w specjalnie przygotowanych funkcjach, dzięki czemu pętla główna jest czytelna i umożliwia łatwe dopisanie obsługi kolejnej rośliny i czujników.

```
void loop() {
  btConnected = digitalRead(BT_STATE_PIN) == HIGH;
  int lightRaw = 0;

  Serial.println("----------------------------------------");
  Serial.println("NOWY CYKL - odczyt warunkow otoczenia:");

  // Odczyt temperatury, wilgotnosci powietrza i jasnosci
  readEnvironment(lightRaw);

  // Roslinka 1
  processPlant(
    SOIL_OPT1_PIN,
    RELAY_OPT3_PIN,
    "Roslinka 1",
    SOIL1_DRY,
    SOIL1_WET,
    SOIL1_THRESHOLD
  );

  Serial.println("Czuwanie...");
  delay(IDLE_DELAY_MS);
}
```

Listing 11.1: Pętla główna programu Arduino (`loop()`)

11.2 Progi wilgotności i decyzja o podlewaniu

Dla każdej rośliny wykonywany jest odczyt czujnika wilgotności gleby. Każda z nich może mieć inny próg, ponieważ różne rośliny wymagają różnego poziomu wilgotności. Próg oznacza minimalną akceptowalną wilgotność gleby. Jeśli wilgotność spadnie poniżej progu, roślina jest uznawana za wymagającą podlania.

```
const int SOIL2_DRY       = 551;  // odczyt dla suchej ziemi
const int SOIL2_WET       = 220;  // odczyt dla wody
const int SOIL2_THRESHOLD = 10;   // prog wilgotnosci (%) ponizej ktorego podlewamy
```

Listing 11.2: Przykład kalibracji i progu dla czujnika wilgotności

11.3 Znaczenie pól w ramce `PLANT`

Podczas wysyłania informacji o roślinie system przekazuje pola, które opisują decyzję o podlewaniu oraz stan wykonania akcji.

`needWater`

- `0` — gleba wystarczająco wilgotna, nie trzeba podlewać,
- `1` — gleba za sucha, roślina wymagała podlewania (nie zawsze oznacza, że udało się podlać).

`waterState`

- `-1` — nie sprawdzano zbiornika (nie było potrzeby podlewania),
- `0` — roślina wymagała podlewania, ale nie podlano (brak wody w zbiorniku),
- `1` — roślina wymagała podlewania i podlewanie zostało wykonane.

`watered`

- `0` — roślina nie została podlana,
- `1` — roślina została podlana.

`soil` i `soilRaw`

- `soil` — zmapowana wartość wilgotności (np. w %),
- `soilRaw` — surowa wartość wilgotności pobrana z czujnika.

Rozdział 12

Wizualizacja danych

Dla potrzeb wizualizacji powstała konieczność pozyskiwania danych z Arduino w celu tworzenia wykresów. Przesyłanie danych wymaga jednak przygotowania odpowiedniego formatu logów. Z tego powodu zaprojektowano dwie struktury plików: osobną dla części środowiskowej oraz osobną dla danych dotyczących roślin podłączonych do systemu. Pliki są zapisywane w formacie *JSON Lines* (`.jsonl`), co oznacza, że jeden rekord JSON odpowiada jednej linii w pliku.

12.1 Struktura danych środowiskowych (ENV)

Rekord typu `ENV` zawiera:

- stan czujnika DHT22,
- temperaturę powietrza,
- wilgotność powietrza,
- jasność otoczenia (wartość z LDR),
- czas systemowy (`t`),
- pełny znacznik czasu wykorzystywany w wizualizacji (`timestamp`).

```
1 {"kind": "ENV", "ok": "1", "temp": "16.90", "hum": "80.70", "light": "549", "t":
    "611329", "timestamp": "2025-12-01T23:53:50"}
```

Listing 12.1: Przykład rekordu ENV w formacie JSONL

12.2 Struktura danych o roślinach (PLANT)

Rekord typu `PLANT` zawiera:

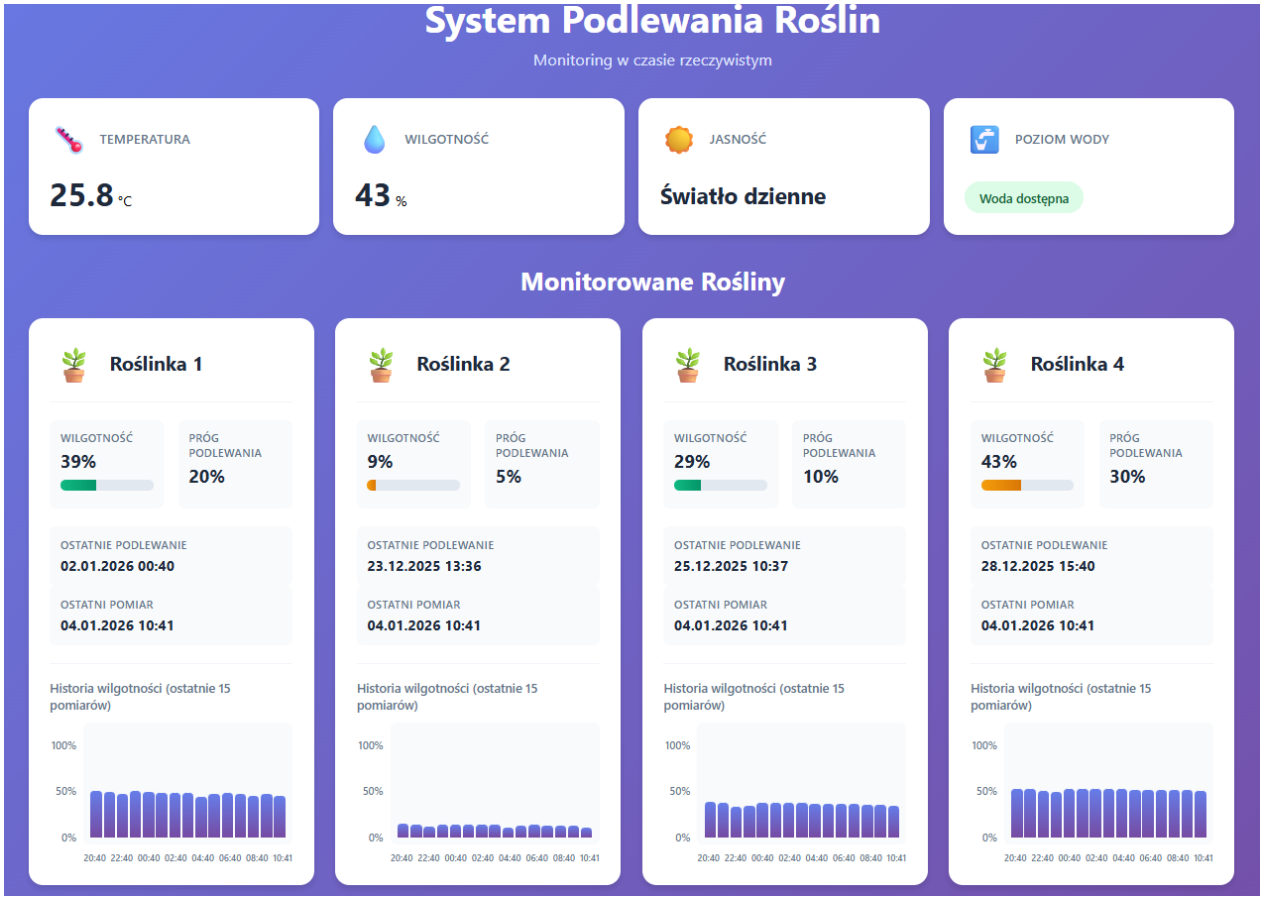
- nazwę rośliny,
- surowy odczyt wilgotności gleby (`soilRaw`),
- przeliczoną wilgotność w % uzyskaną z mapowania (`soil`),
- próg działania dla danej rośliny (`threshold`),
- informację, czy roślina wymagała podlewania (`needWater`),
- informację o stanie zbiornika wody (`waterState`, gdzie `-1` oznacza brak sprawdzenia),

- informację, czy roślina została podlana (`watered`),
- czas systemowy (`t`),
- pełny znacznik czasu (`timestamp`).

```
{"kind": "PLANT", "name": "Roslinka 1", "soilRaw": "465", "soil": "31", "threshold": "20", "needWater": "0", "waterState": "-1", "watered": "0", "t": "32", "timestamp": "2025-12-01T23:58:35"}
{"kind": "PLANT", "name": "Roslinka 2", "soilRaw": "320", "soil": "80", "threshold": "5", "needWater": "0", "waterState": "-1", "watered": "0", "t": "116", "timestamp": "2025-12-01T23:58:35"}
{"kind": "PLANT", "name": "Roslinka 3", "soilRaw": "551", "soil": "5", "threshold": "5", "needWater": "0", "waterState": "-1", "watered": "0", "t": "220", "timestamp": "2025-12-01T23:58:35"}
{"kind": "PLANT", "name": "Roslinka 4", "soilRaw": "470", "soil": "56", "threshold": "10", "needWater": "0", "waterState": "-1", "watered": "0", "t": "322", "timestamp": "2025-12-01T23:58:35"}
```

Listing 12.2: Przykładowe rekordy PLANT w formacie JSONL

12.3 Strona z wizualizacją danych



Strona została stworzona w języku Python z wykorzystaniem frameworka Flask. Dane z Arduino są odbierane poprzez nasłuchiwanie portu szeregowego, co umożliwia moduł `serial`. Po odebraniu ramki typu

`ENV` lub `PLANT` program zapisuje ją odpowiednio do plików w formacie `.jsonl` opisanych w poprzednich sekcjach. Następnie dane są wczytywane przez aplikację webową, dzięki czemu możliwe jest przedstawienie czytelnych informacji o otoczeniu oraz o roślinach, w tym m.in. ostatniej daty podlewania oraz historii wilgotności (dla 15 pomiarów).

Ze względu na możliwość występowania braków komunikacji (np. skrypt nasłuchujący nie jest uruchomiony, a urządzenie zapisuje dane na karcie SD), program grupuje ramki w klastry i zapisuje je dopiero po wykryciu przerwy pomiędzy kolejnymi ramkami większej niż ustalony limit `CLUSTER_MAX_INTERARRIVAL_SECONDS = 30`. Z tego powodu dane prezentowane na wizualizacjach nie są aktualizowane w czasie rzeczywistym i mogą pochodzić np. sprzed godziny.

Rozdział 13

Załączniki

- **Schemat blokowy Arduino** (`schematy_blokowe.pdf`) — schemat blokowy programu `smartplant.ino`, odpowiedzialnego za pobieranie informacji z czujników oraz sterowanie podlewaniem.
- **Schemat blokowy Listener** (`schematy_blokowe.pdf`) — schemat blokowy programu `data_listener.py`, przeznaczonego do nasłuchiwania, odbierania oraz zapisywania ramek wysyłanych z Arduino.
- **Kod źródłowy programu** (`SmartPlant_Dashboard.zip`) — kompletne źródła projektu obejmujące oprogramowanie mikrokontrolera, skrypt nasłuchujący oraz aplikację webową.

Rozdział 14

Wnioski

Opracowany system spełnia założone cele projektu i działa poprawnie w warunkach domowych. Zastosowane rozwiązanie umożliwia skuteczne monitorowanie wilgotności gleby oraz automatyczne podlewanie roślin z uwzględnieniem poziomu wody w zbiorniku, co eliminuje ryzyko podlewania w sytuacji braku wody i pozwala na utrzymanie odpowiedniego poziomu wilgotności gleby dla każdej rośliny.

Integracja pomiarów środowiskowych z archiwizacją oraz wizualizacją danych znacząco zwiększa funkcjonalność systemu i ułatwia długoterminową kontrolę stanu roślin. Projekt może być z powodzeniem wykorzystywany do automatycznej pielęgnacji roślin doniczkowych, a jego modułowa budowa umożliwia dalszą rozbudowę o kolejne czujniki oraz funkcje.