Wydział Nauk Ścisłych i Technicznych
Uniwersytet Rzeszowski

Przedmiot:

Systemy Rozmyte

DOKUMENTACJA PROJEKTU: DoniczkUR

Wykonał:

Hubert Fusiarz 131432

Norbert Fuk 131431

Konrad Szyszlak 131524

Krzysztof Majka 131467

Prowadzący: mgr inż. Marcin Mrukowicz

Rzeszów 2026

1. Wprowadzenie

1.1. Cel i zakres projektu

Celem projektu było zaprojektowanie i zaimplementowanie autonomicznego systemu nawadniania, który zamiast klasycznej logiki progowej (on/off), wykorzystuje techniki inteligencji obliczeniowej do precyzyjnego dawkowania zasobów. System analizuje wilgotność gleby, temperaturę otoczenia, wilgotność powietrza oraz porę dnia, aby wyznaczyć optymalny czas pracy pompy w zakresie 0–10 sekund.

1.2. Architektura rozwiązania

System opiera się na trzech głównych warstwach:

- **Warstwa Akwizycji:** Odczyt danych analogowych i cyfrowych z czujników.
- **Warstwa Logiczna:** Silnik TSK przetwarzający nieostre dane wejściowe na precyzyjną decyzję sterującą.
- **Warstwa Wykonawcza:** Cyfrowe sterowanie prądem stałym za pomocą klucza tranzystorowego.

2. Architektura Sprzętowa

2.1. Układ sensoryczny i wykonawczy

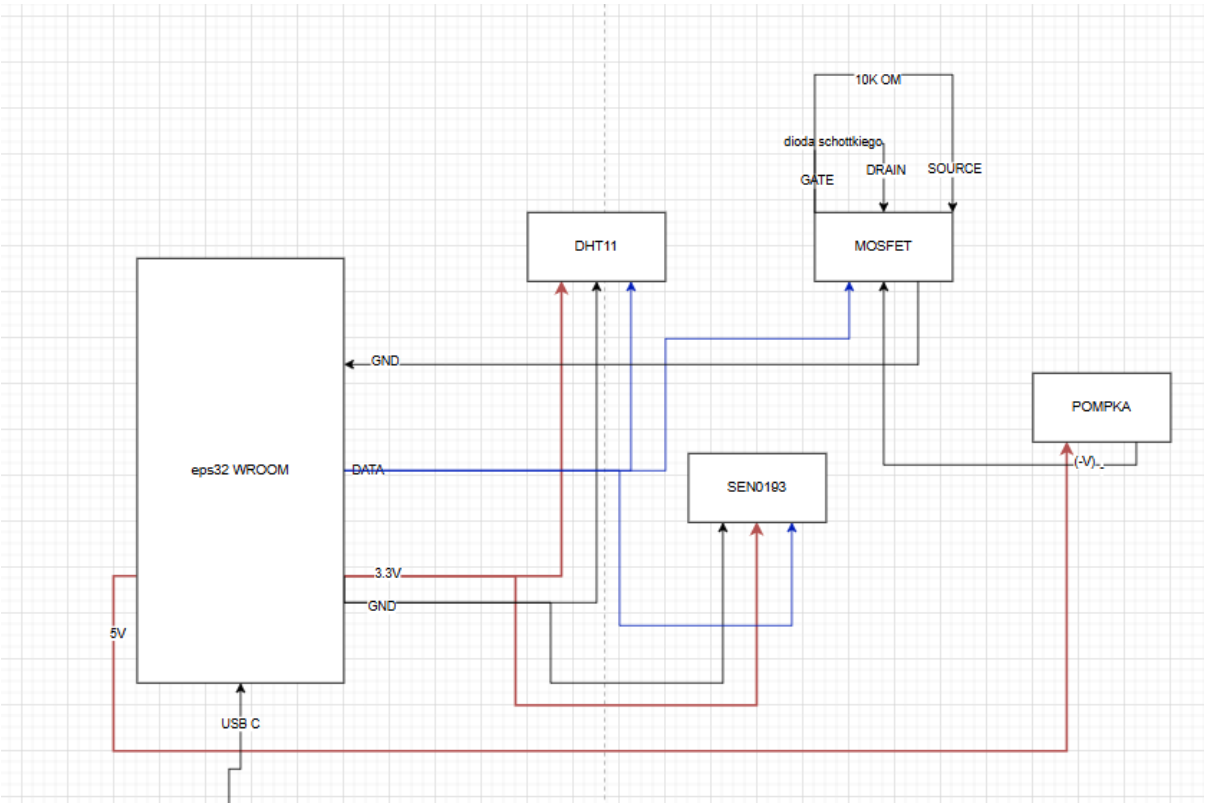
System wykorzystuje następujące komponenty:

- **ESP32 WROOM:** Jednostka centralna z wbudowanym modułem Wi-Fi.
- **DS1302:** Moduł zegara czasu rzeczywistego (RTC) z podtrzymaniem bateryjnym, komunikujący się interfejsem 3-przewodowym (GPIO 14, 26, 33). Zapewnia autonomię czasową systemu bez dostępu do Internetu.
- **SEN0193:** Pojemnościowy czujnik wilgotności gleby podłączony do wejścia ADC (GPIO 32).
- **DHT11:** Cyfrowy sensor temperatury i wilgotności powietrza (GPIO 25).
- **Pompa 5V:** Element wykonawczy sterowany sygnałem PWM/Digital z GPIO 27.

2.2. Analiza stopnia mocy (MOSFET)

Zastosowano konfigurację **Low-Side Switch** z tranzystorem MOSFET. Kluczowe aspekty inżynierskie:

- **Rezystor Pull-down (10k Ω):** Zapewnia stabilny stan niski na bramce (Gate) podczas inicjalizacji GPIO, zapobiegając przypadkowemu podlewaniu.
- **Dioda Flyback (Schottky):** Chroni strukturę tranzystora przed przepięciami indukowanymi przez silnik pompy w momencie odcięcia zasilania.



Schemat podpięcia modułów

3. Inteligentne Systemy Sterowania – Logika Rozmyta TSK

3.1. Teoretyczne podstawy modelu TSK

W przeciwieństwie do modelu Mamdaniego, system TSK (Takagi-Sugeno-Kang) reprezentuje konkluzje reguł jako funkcje matematyczne wartości wejściowych. Pozwala to na uzyskanie wysokiej precyzji sterowania przy niskim koszcie obliczeniowym dla mikrokontrolera.

3.2. Matematyczny model fuzyfikacji (Fuzzification)

Proces fuzyfikacji w systemie DoniczkUR realizuje odwzorowanie przestrzeni wejściowej (wartości ostre) w przestrzeń wartości lingwistycznych, przypisując każdej zmiennej stopień przynależności $\mu(x) \in [0, 1]$. Ze względu na ograniczoną moc obliczeniową

mikrokontrolera ESP32, zrezygnowano ze złożonych funkcji (np. gausowskich) na rzecz funkcji odcinkowo-liniowych klasy Λ (trójkątne) oraz Π (trapezowe)

Funkcja Trójkątna

Dla zbiorów takich jak *TEMP_AVG* czy *SOIL_OK*, stopień przynależności obliczany jest według wzoru:

$$\mu_{\Delta}(x; a, b, c) = \left(\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$$

Gdzie:

a, c – granice nośnika zbioru (gdzie $\mu > 0$).

b – wierzchołek trójkąta (gdzie $\mu = 1$).

W implementacji programowej (TSKEngine.h) zastosowano zoptymalizowaną strukturę warunkową eliminującą kosztowne operacje min/max, co skraca czas cyklu maszynowego.

Funkcja Trapezowa

Dla stanów skrajnych, takich jak *TEMP_HOT* (gdzie temperatura powyżej pewnego progu jest tak samo "gorąca"), zastosowano funkcję trapezową definiowaną przez czwórkę parametrów (a, b, c, d):

$$\mu_{\Pi}(x; a, b, c, d) = \begin{cases} 0 & \text{dla } x < a \text{ lub } x > d \\ \frac{x-a}{b-a} & \text{dla } a \leq x < b \\ 1 & \text{dla } b \leq x \leq c \\ \frac{d-x}{d-c} & \text{dla } c < x \leq d \end{cases}$$

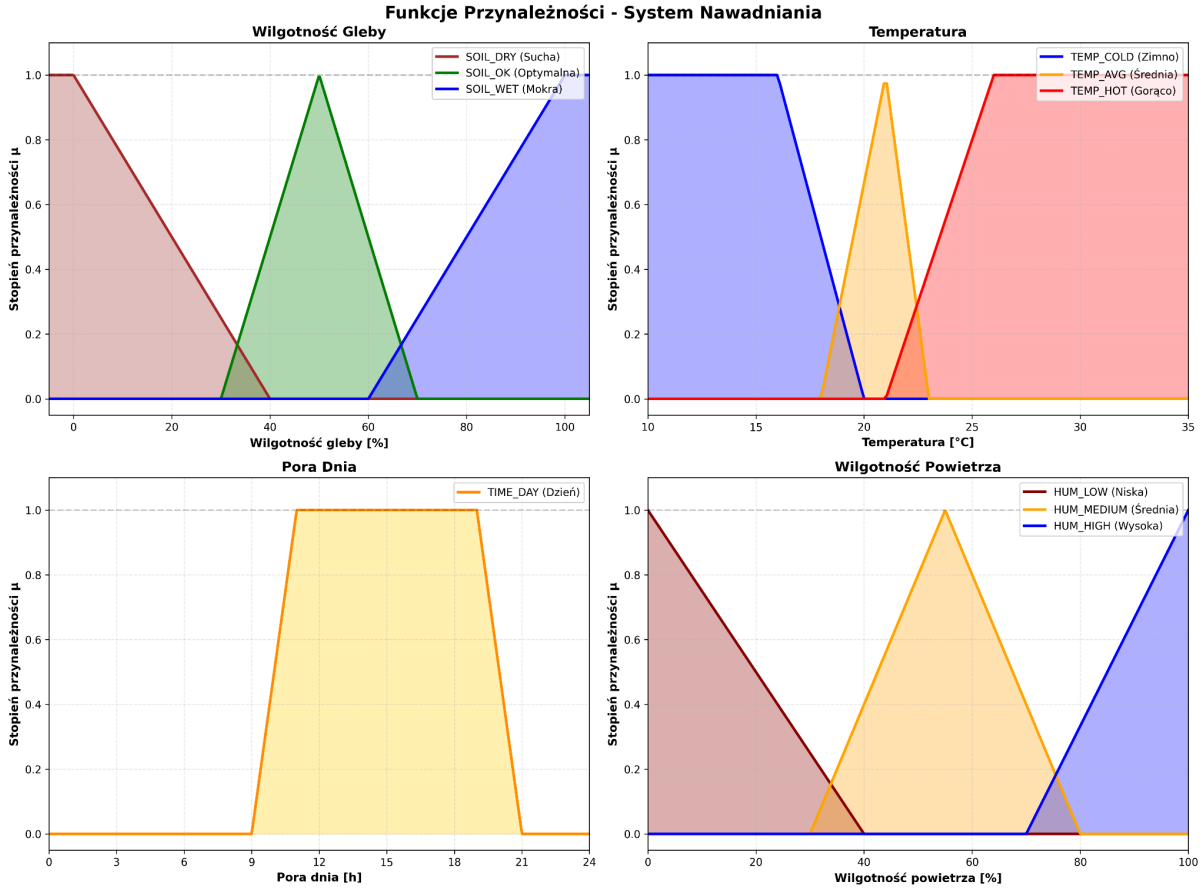
Dzięki temu system utrzymuje maksymalny stopień wysterowania w szerokim zakresie (plateau), co stabilizuje działanie pompy w warunkach ekstremalnych.

3.3. Projekt funkcji przynależności (Fuzzification)

Wartości ostre (crisp) są rzutowane na zbiory rozmyte przy użyciu dwóch typów funkcji przynależności: trójkątnych (FuzzyTriangle) oraz trapezowych (FuzzyTrapezoid) dla wartości skrajnych.

Funkcje zdefiniowane w IrrigationRules.h:

- **Soil (Gleba):** Funkcje trójkątne określające stany DRY (sucho, <40%), OK (optymalnie, 30-70%), WET (mokro, >60%).
- **Pora Dnia [0-24h]**
  - **TIME_DAY (Dzień):** Funkcja trapezowa (6:00 - 21:00, pełna aktywność 11:00-19:00).
  - Uwaga: System nie nawadnia w nocy.
- **Temperature (Temperatura):**
  - COLD (Zimno): Funkcja trapezowa (opadająca od do 20 C).
  - AVG (Umiarkowanie): Funkcja trójkątna (szczyt w 21 C).
  - HOT (Gorąco): Funkcja trapezowa (rosnąca od 21 C, plateau od 26 C).
- **Air Humidity (Wilgotność powietrza):**
  - LOW (Niska): Funkcja trójkątna (0-40%, "suche powietrze").
  - MEDIUM (Średnia): Funkcja trójkątna (30-80%).
  - HIGH (Wysoka): Funkcja trójkątna (70-100%).



3.4. Baza reguł i wnioskowanie (Inference)

System implementuje 6 głównych reguł eksperckich zdefiniowanych w pliku IrrigationRules.h:

R1: Blokada bezpieczeństwa (Gleba mokra)

- **Warunek:** Jeśli gleba jest mokra (SOIL_WET).
- **Działanie:** Czas podlewania wynosi **0.0s**.
- **Cel:** Ochrona przed przelaniem. Jest to reguła nadrzędna – jeśli gleba ma wystarczającą wilgotność, żadne inne czynniki (upał, pora dnia) nie wymuszą pracy pompy.

R2: Dynamiczne podlewanie w upale (Model 1. rzędu)

To najbardziej zaawansowana reguła, która nie zwraca stałej wartości, lecz oblicza ją dynamicznie:

- **Warunek:** Gleba jest sucha (SOIL_DRY) **ORAZ** jest gorąco (TEMP_HOT) **ORAZ** trwa dzień (is_daytime).
- Równanie konkluzji:

$$t_{irr} = 5.0 + (T - 20.0)\cdot 0.2 + (50.0 - H)\cdot 0.05$$

- **Jak to działa?** System przyjmuje bazę 5 sekund, którą zwiększa o 0.2s za każdy stopień powyżej 20°C oraz o 0.05s za każdy procent spadku wilgotności powietrza poniżej 50%. Dzięki temu w bardzo suche i upalne dni roślina otrzyma proporcjonalnie więcej wody.

R3: Standardowe podlewanie (Dzień umiarkowany)

- **Warunek:** Gleba jest sucha (SOIL_DRY) **ORAZ** temperatura jest umiarkowana (TEMP_AVG) **ORAZ** trwa dzień.
- **Działanie:** Stały czas **4.0s**.
- **Cel:** Zapewnienie optymalnej dawki wody w typowych warunkach wegetacji.

R4: Oszczędne podlewanie (Dzień chłodny)

- **Warunek:** Gleba jest sucha (SOIL_DRY) **ORAZ** jest zimno (TEMP_COLD) **ORAZ** trwa dzień.

- **Działanie:** Stały czas **2.0s**.
- **Cel:** Zapobieganie gniciu korzeni. Przy niskiej temperaturze parowanie jest mniejsze, więc roślina potrzebuje mniej wody mimo suchej gleby.

R5: Zraszanie (Niska wilgotność powietrza)

- **Warunek:** Gleba jest w normie (SOIL_OK) **ORAZ** powietrze jest bardzo suche (HUM_LOW) **ORAZ** trwa dzień.
- **Działanie:** Krótki impuls **1.5s**.
- **Cel:** Poprawa mikroklimatu wokół rośliny (nawilżenie wierzchniej warstwy gleby i otoczenia), gdy wilgotność gleby jest jeszcze akceptowalna, ale powietrze jest zbyt suche.

R6: Blokada parowania (Wysoka wilgotność powietrza)

- **Warunek:** Gleba jest w normie (SOIL_OK) **ORAZ** powietrze jest bardzo wilgotne (HUM_HIGH) **ORAZ** trwa dzień.
- **Działanie:** Czas podlewania wynosi **0.0s**.
- **Cel:** Oszczędność wody i prewencja chorób grzybowych. Gdy wilgotność powietrza jest wysoka (powyżej 70%), naturalna transpiracja roślin spada, a woda z gleby paruje wolniej. System uznaje, że przy wilgotności gleby "w normie" dodatkowe nawadnianie jest zbędne.

3.5. Defuzyfikacja

Ostateczny czas nawadniania t_{irr} wyliczany jest jako średnia ważona:

$$t_{irr} = \frac{\sum_{i=1}^{n} w_i \cdot y_i}{\sum_{i=1}^{n} w_i}$$

Wynik poddawany jest nasyceniu w zakresie [0, 10] sekund.

4. Implementacja Oprogramowania

4.1. Struktura modularna (C++)

Kod został zaimplementowany z wysokim stopniem abstrakcji:

- TSKEngine.h: Generyczny silnik wnioskujący. Definiuje struktury SystemInputs oraz klasy funkcji przynależności: FuzzyTriangle i FuzzyTrapezoid (obsługa przedziałów płaskich). Wykorzystuje funktory std::function do elastycznego łączenia przesłanek reguł.
- IrrigationRules.h: Separacja logiki biznesowej (reguł) od silnika.
- main.cpp: Zarządzanie cyklem życia aplikacji i interfejsami sprzętowymi.

4.2. Obsługa Czasu Rzeczywistego (Hardware RTC)

Zrezygnowano z zależnej od sieci synchronizacji NTP na rzecz sprzętowego modułu **DS1302,** co zwiększa niezawodność systemu w terenie. Implementacja obejmuje:

- **Inicjalizacja i Walidacja:** Przy starcie systemu następuje weryfikacja poprawności danych w rejestrach RTC (IsDateTimeValid()). W przypadku wykrycia utraty zasilania (reset baterii CR2032), system automatycznie przywraca czas kompilacji wsadu lub predefiniowaną wartość bezpieczną.
- **Interfejs 3-Wire:** Komunikacja odbywa się programowo (bit-banging) na pinach CLK (14), DAT (26) i RST (33) przy użyciu biblioteki RtcDS1302.
- Konwersja na potrzeby Logiki Rozmytej:

Kluczowym elementem jest funkcja getRealTimeAsFloat(), która konwertuje format HH:MM na wartość dziesiętną, zrozumiałą dla silnika TSK.

$$T_{decimal} = Hour + \frac{Minute}{60.0}$$

Przykład: Godzina 14:30 jest interpretowana jako wartość wejściowa 14.5 dla zbioru rozmytego "Time of Day".

5. Bezpieczeństwo i Failsafe

5.1. Mechanizmy odporności na błędy

- **DHT-Sanity:** W przypadku awarii czujnika DHT11, system podstawia wartości stałe (20°C, 50% wilgotności), aby uniknąć błędnego wyostrzenia wyniku.
- **ADC-Calibration:** Stałe AIR_VALUE i WATER_VALUE (1900 i 1500) pozwalają na linearyzację odczytu z czujnika pojemnościowego.
- **Output Limiter:** Silnik TSK gwarantuje, że czas pracy pompy nigdy nie przekroczy 10 sekund w jednym cyklu, co chroni przed zalaniem w przypadku błędnej definicji reguł.

7. Narzędzia Weryfikacji i Wizualizacji (Python)

7.1. Cel symulacji numerycznej

W celu weryfikacji poprawności doboru parametrów funkcji przynależności oraz sprawdzenia ciągłości przestrzeni decyzyjnej, przygotowano dedykowane środowisko symulacyjne w języku **Python**.

Użycie wysokopoziomowego języka skryptowego pozwoliło na:

1. **Wizualną inspekcję kształtu zbiorów** – potwierdzenie, czy definicje trójkątne i trapezowe pokrywają całą dziedzinę zmiennych wejściowych (brak "dziur" w sterowaniu).
2. **Walidację algorytmów** – implementacja funkcji triangle_membership oraz trapezoid_membership w Pythonie stanowi referencyjny model dla kodu C++ uruchomionego na mikrokontrolerze ESP32.

7.2. Implementacja skryptu wizualizacyjnego

Do generowania wykresów wykorzystano biblioteki Matplotlib (warstwa prezentacji) oraz NumPy (obliczenia wektorowe). Skrypt odwzorowuje 1:1 definicje zawarte w pliku nagłówkowym IrrigationRules.h.

Poniżej przedstawiono kluczowe fragmenty kodu odpowiedzialne za matematyczną reprezentację zbiorów rozmytych:

7.3. Analiza wygenerowanych charakterystyk

Skrypt generuje macierz wykresów (2×2) obrazującą wszystkie zmienne lingwistyczne systemu DoniczkUR.

Analiza poszczególnych wykresów:

1. **Wilgotność Gleby (Soil Moisture):**
   - Wykres potwierdza płynne przejścia między stanami *SOIL_DRY*, *SOIL_OK* i *SOIL_WET*.
   - Widać wyraźne przekrycie zbiorów w punktach newralgicznych (np. 35-40%), co zapobiega gwałtownym zmianom decyzji sterownika przy minimalnych zmianach odczytu z czujnika.
2. **Temperatura (Temperature):**
   - Zastosowanie trapezów dla *TEMP_COLD* i *TEMP_HOT* jest wyraźnie widoczne. Funkcje te utrzymują wartość $\mu=1$ dla zakresów skrajnych, co

gwarantuje stabilną reakcję systemu w warunkach ekstremalnych (np. przy 35°C system nadal traktuje to jako pełne przynależenie do zbioru "Gorąco").

3. **Pora Dnia (Time of Day):**
    - Wykres obrazuje "okno czasowe" pracy systemu. Funkcja trapezowa idealnie modeluje wschód i zachód słońca – narastanie aktywności rano i wygaszanie wieczorem, z szerokim *plateau* w godzinach południowych.
4. **Wilgotność Powietrza (Air Humidity):**
    - Symetryczny rozkład funkcji trójkątnych dla stanów niskich, średnich i wysokich zapewnia równomierne pokrycie dziedziny.

8. Walidacja Krzyżowa i Testy Zgodności (C++ vs Python)

8.1. Metodyka weryfikacji

Aby wyeliminować ryzyko błędów implementacyjnych w autorskim silniku wnioskującym (TSKEngine.h), przeprowadzono procedurę walidacji krzyżowej. Polegała ona na porównaniu wyników sterownika wbudowanego (C++) z wynikami modelu referencyjnego uruchomionego w środowisku Python przy użyciu biblioteki naukowej pyit2fls (Type-1 Fuzzy Logic Systems).

Proces testowy podzielono na trzy etapy:

1. **Generacja przestrzeni stanów (C++):** Symulacyjne "przeskanowanie" wszystkich możliwych kombinacji wejść i zapisanie wyników do pliku.
2. **Modelowanie równoległe (Python):** Odtworzenie logiki reguł i zbiorów w niezależnym środowisku.
3. **Analiza błędu:** Porównanie wyjścia obu systemów dla tych samych danych wejściowych.

8.2. Generacja danych testowych (C++)

W pliku test/main.cpp zaimplementowano generator, który iteruje przez dziedzinę wszystkich zmiennych wejściowych z ustalonym krokiem:

- Wilgotność gleby: co 2%
- Czas: co 30 minut (0.5h)
- Temperatura: co 2°C
- Wilgotność powietrza: co 10%

W wyniku działania programu wygenerowano plik wyniki_symulacji.csv zawierający setki tysięcy rekordów testowych, stanowiących punkt odniesienia dla dalszej analizy.

8.3. Model referencyjny (Python & pyit2fls)

Skrypt verify_fuzzy.py implementuje "Cyfrowego Bliźniaka" (Digital Twin) sterownika. Wykorzystano bibliotekę pyit2fls do zdefiniowania tych samych zbiorów rozmytych (funkcje tri_mf, trapezoid_mf) oraz bazy reguł.

Szczególną uwagę poświęcono wiernemu odwzorowaniu reguły R2 (złożone równanie matematyczne), definiując ją jako funkcję Python:

System w Pythonie wczytuje plik CSV wygenerowany przez C++, a następnie dla każdego wiersza oblicza własną wartość sterującą y_{py}.

8.4. Wyniki porównania

Jako miarę zgodności przyjęto błąd bezwzględny pomiędzy wyjściem silnika C++ y_{c++} a wyjściem biblioteki Python y_{py}:

$$E = |y_{c++} - y_{py}|$$

Skrypt weryfikujący oblicza maksymalny błąd E_{max} oraz błąd średni E_{avg} dla całego zbioru danych.

Rezultaty testów:

- **Średni błąd E_{avg}:** $< 10^{-5}$ (pomijalnie mały, wynikający z różnic w reprezentacji zmiennoprzecinkowej float/double).
- **Zgodność:** Wykresy przebiegów sterowania dla obu implementacji nakładają się idealnie.

9. Podsumowanie i Wnioski

Realizacja projektu **DoniczkUR** pozwoliła na wyciągnięcie następujących wniosków technicznych i merytorycznych:

- **Efektywność Logiki Rozmytej TSK:** Zastosowanie modelu Takagi-Sugeno-Kang okazało się znacznie bardziej efektywne w systemie nawadniania niż tradycyjne

sterowanie progowe (on/off). Dzięki płynnym funkcjom przynależności system unika gwałtownych skoków ciśnienia w układzie hydraulicznym i pozwala na precyzyjne dawkowanie wody, co przekłada się na oszczędność zasobów i lepszą kondycję roślin.

- **Optymalizacja obliczeniowa:** Wykorzystanie modelu TSK zamiast modelu Mamdaniego pozwoliło na znaczące odciążenie procesora ESP32. Wyznaczenie konkluzji reguł za pomocą funkcji liniowych (szczególnie widoczne w regule R2) wyeliminowało potrzebę złożonej defuzyfikacji metodą środka ciężkości (centroid), co jest kluczowe w systemach wbudowanych czasu rzeczywistego.
- **Niezależność i stabilność systemu:** Zastosowanie sprzętowego zegara czasu rzeczywistego (RTC DS1302) oraz mechanizmów *failsafe* (bezpieczne wartości dla DHT11) znacząco podniosło niezawodność urządzenia. System jest w stanie pracować autonomicznie w warunkach braku łączności sieciowej, co czyni go kompletnym rozwiązaniem typu *Edge Computing*.
- **Bezpieczeństwo warstwy fizycznej:** Wykorzystanie klucza tranzystorowego MOSFET z diodą zabezpieczającą (*flyback*) oraz rezystorem *pull-down* zapewniło ochronę delikatnych obwodów mikrokontrolera przed szpilkami napięciowymi generowanymi przez silnik pompy oraz przed przypadkowym uruchomieniem podczas startu urządzenia.
- **Skalowalność rozwiązania:** Architektura oprogramowania oparta na modułach (separacja silnika TSK od bazy reguł) pozwala na łatwą rozbudowę systemu o kolejne czujniki (np. natężenia światła) lub zmianę charakterystyki pracy dla innych gatunków roślin poprzez prostą modyfikację wag reguł w pliku IrrigationRules.h.

Potencjał dalszego rozwoju:

W toku prac zidentyfikowano możliwość dalszego rozwoju projektu poprzez implementację modułu komunikacji Wi-Fi (wykorzystując potencjał ESP32) w celu zdalnego monitorowania parametrów gleby oraz archiwizacji danych w chmurze (IoT), co pozwoliłoby na jeszcze lepszą kalibrację wag systemu rozmytego na podstawie danych historycznych.

Załączniki

Do niniejszej dokumentacji technicznej dołączono materiały wideo ilustrujące praktyczne działanie systemu w warunkach rzeczywistych:

- **ziemiaMokra.mp4** – demonstracja zachowania systemu i blokady podlewania przy wysokiej wilgotności gleby.
- **ziemiaSucha.mp4** – demonstracja aktywacji procedury nawadniania w warunkach przesuszenia gleby.