POLITECHNIKA GDAŃSKA
Wydział Elektroniki, Telekomunikacji i Informatyki

Imię i nazwisko studenta: Łukasz Danilewicz
Nr albumu: 93792
Poziom kształcenia: studia drugiego stopnia
Forma studiów: niestacjonarne
Kierunek studiów: Informatyka
Specjalność: Systemy i sieci komputerowe

PRACA DYPLOMOWA MAGISTERSKA

Tytuł pracy w języku polskim: Sterownik wielostrefowy z dedykowaną aplikacją mobilną do systemu nawadniania ogrodu uwzględniającego prognozy pogody

Tytuł pracy w języku angielskim: A multi-zone controller with a dedicated mobile application for the garden irrigation system, taking into account weather forecasts

Opiekun pracy: prof. dr hab. inż. Zdzisław Kowalczuk

Data ostatecznego zatwierdzenia raportu podobieństw w JSA:

STRESZCZENIE

Celem niniejszej pracy magisterskiej pt. „Sterownik wielostrefowy z dedykowaną aplikacją mobilną do systemu nawadniania ogrodu uwzględniającego prognozy pogody” jest zaprojektowanie, zbudowanie i zaprogramowanie urządzenia do sterowania systemem nawadniania ogrodu. Do systemu przynależą bezprzewodowe czujniki środowiskowe, które przy użyciu technologii LoRa, w sposób zaszyfrowany przesyłają dane pomiarowe do sterownika. Architektura systemu pozwala obsłużyć do 16 sekcji nawodnieniowych i wykorzystać do 4 czujników w każdej, dostarczając dane środowiskowe do jednostki sterującej. Możliwość wyposażenia systemu w czujnik przepływu cieczy oraz możliwość obsługi zaworu głównego podnoszą poziom bezpieczeństwa. Dodatkowo wprowadzone zostały alerty, przesyłane do brokera MQTT i możliwie bezzwłocznie dostarczane do użytkownika.

Kluczowym elementem czujnika środowiska jest element wykonawczy do pomiaru wilgotności gleby. Po dwóch stworzonych prototypach, zostaje wykonana finalna wersja na której przeprowadzone zostają badania, obejmujące 4 eksperymenty pomiarowe. Postawiona hipoteza badawcza: „jeżeli zawartość wody w glebie będzie się zmniejszać, to wraz z nią zmniejszać się będzie pasożytnicza pojemność elektryczna obwodu drukowanego” zostaje zweryfikowana pozytywnie, a wyniki eksperymentów, po analizie, pozwalają na wykorzystanie elementu pomiarowego w budowanym systemie.

Wspomniany bezprzewodowy czujnik środowiskowy zostaje wyposażony w unikalną implementację systemu pomiarowego, wykorzystującą pobudzenie do 3 elementów wykonawczych sygnałem prostokątnym o częstotliwości 40MHz. Jest to urządzenie częściowo posadowione w glebie o zasilaniu akumulatorowym, z możliwością doładowywania ogniwem słonecznym lub ładowarką ze standardem USB. Charakteryzuje je prosty interfejs użytkownika oraz architektura zapewniająca oszczędność energii.

W pracy zostaje wprowadzona koncepcja algorytmu OptiServ uruchamiania sekcji nawodnieniowych przy zdefiniowanej długość i okresie nawadniania, bez ustalania godzin rozpoczęcia. Decyzja o nawadnianiu zostaje podjęta z uwzględnieniem prognoz pogody (temperatura, wilgotność, ewapotranspiracja) oraz danych z czujników środowiskowych w zgodnym z naturą zakresie od wschodu do zachodu słońca. W celu przetestowania algorytmu zostają przeprowadzone symulacje, a następnie dzięki stworzonej przeglądarce wizualizującej wyniki oraz modułowi statystycznemu, następuje ich analiza i pozytywna weryfikacja.

Ostatecznie algorytm OptiServ zostaje zaimplementowany w sterowniku, a cel główny pracy zostaje osiągnięty. Poza tym osiągnięte zostają oba cele pośrednie. Pierwszy poprzez skuteczne włączenie do systemu czujników środowiskowych z unikalną implementacją układu pomiarowego. Natomiast drugi zrealizowany cel pośredni to implementacja dedykowanej aplikacji mobilnej do zmian w konfiguracji systemu i przesyłania komend do sterownika.

Słowa kluczowe: sterownik, nawadnianie, wilgotność, gleba, czujnik, LoRa, elektrozawór, optymalizacja, woda, ESP32, MQTT

Dziedzina nauki i techniki, zgodnie z wymogami OECD: 2.2 Elektrotechnika, elektronika, inżynieria informatyczna

ABSTRACT

The aim of this thesis entitled. "A multi-zone controller with a dedicated mobile application for the garden irrigation system, taking into account weather forecasts" is to design, build and program a device to control a garden irrigation system. The system includes wireless environmental sensors that, using LoRa technology, encryptedly transmit measurement data to the controller. The system architecture allows up to 16 irrigation sections to be operated and use up to four sensors in each, providing environmental data to the control unit. The option to equip the system with a liquid flow sensor and to prepare for the operation of a main valve increases the level of safety. In addition, alerts are introduced, sent to the MQTT broker and delivered to the user as promptly as possible.

A key component of the environmental sensor is the actuator for measuring soil moisture. After two prototypes have been created, a final version is made on which tests are carried out, including four measurement experiments. The research hypothesis: 'if the water content of the soil decreases, the parasitic capacitance of the circuit board will decrease with it' is positively verified and the results of the experiments allow the measuring element to be used in the system under construction.

The aforementioned wireless environmental sensor is equipped with a unique implementation of the measurement system, using the excitation of up to 3 actuators with a 40MHz rectangular signal. It is a partially embedded device with a battery power supply, rechargeable with a solar cell. It features a simple user interface and a power-saving architecture.

The concept of the OptiServ algorithm to trigger irrigation sections at a defined irrigation length and period, without setting start times, is introduced in the work. The irrigation decision is made taking into account weather forecasts (temperature, humidity, evapotranspiration) and data from environmental sensors in the nature-compatible range from sunrise to sunset. Simulations are carried out to test the algorithm, and the results are then analysed and positively validated thanks to a visualisation viewer and statistical module developed.

Finally, the OptiServ algorithm is implemented in the controller and the main objective of the work is achieved. In addition, both intermediate objectives are achieved. The first by successfully incorporating environmental sensors with a unique implementation of the measurement system. While the second intermediate objective realised is the implementation of a dedicated mobile application to configure the system for sending commands to the controller.

Keywords: controller, irrigation, moisture, soil, sensor, LoRa, electro-valve, optimization, water, ESP32, MQTT

SPIS TREŚCI

WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

TDR – Time Domain Reflectometry
FDR – Frequency Domain Reflectometry
EEPROM – Electrically Erasable Programmable Read-Only Memory
WWAN – Wireless Wide Area Network
WLAN – Wireless Local Area Network
SSID – *S*ervice Set Identifier
LoRa – Long Range
LoraWAN – LoRa Wide Area Network
LPWAN – Low–Power Wide-Area Network
MAC – Medium Access Protocol
IDE – Integrated Development Environment
RTC – Real-Time Clock
AES – Advanced Encryption Standard
RSA – algorytm Rivesta-Shamira-Adlemana
GCM – Galois/Counter Mode
GPIO – General-Purpose Input/Output
IOT – Internet of Things
SoC – System on a Chip
SNR – Signal-to-Noise Ratio
DSSS – Direct Sequence Spread Spectrum
SNTP – Simple Network Time Protocol
MQTT – Message Queue Telemetry Transport
M2M – Machine-to-Machine
SD – Secure Digital
PIR – Passive Infra Red
SVG – Scalable Vector Graphics
CPM – Critical Path Method

T – temperatura [°C]
Θ – wilgotność [%]
I_{inr} – początkowy prąd rozruchowy [A]
S_{inr} – moc pozorna podczas rozruchu [VA]
I_{hol} – prąd podtrzymania/pracy [A]
S_{hol} – moc pozorna podczas pracy [VA]
Ψ – potencjał wodny gleby [N/m2]

WSTĘP

Zgodnie z chińskim przysłowiem, aby być szczęśliwym przez całe życie, należy założyć ogród. Podjęcie tego działania implikuje zaś konieczność zdobycia wiedzy dotyczącej uprawy i pielęgnacji roślin, a następnie zdobycie niezbędnych środków do osiągnięcia postawionych pośrednich celów i jak również tego końcowego. Przedsięwzięcia zagospodarowania i utrzymania ogródka działkowego podjął się autor pracy, a jednym z koniecznych kroków stało się jego regularne nawadnianie. Natomiast motywacją do podjęcia tematu pracy stała się konieczność optymalizacji tego nawadniania w odniesieniu do roślin uprawnych, sadowniczych i ozdobnych.

Potrzeba oszczędnego i racjonalnego gospodarowania wodą jest zauważana w aktualnych opracowaniach dotyczących zagadnień społeczno-gospodarczych w Polsce[1,2]. Poza działaniami podejmowanymi w celu zwiększenia retencji wody, innym, ważnym sposobem na zapobieganie jej deficytom jest właściwe wykorzystanie m. in. odnawialnych zasobów wód powierzchniowych oraz wód podziemnych. Kolejnym, nie mniej istotnym argumentem jest możliwość zapewnienia zdrowego rozwoju flory, dla której deficyty i nadmiary wody w różnych fazach wzrostu, a nawet porach dnia, potrafi być szkodliwy[3]. Ostatnimi ważnymi, tym razem dla jednostki społecznej, aspektami automatyzacji i racjonalizacji nawadniania jest duża oszczędność czasu i ludzkiej pracy oraz pieniędzy.

Celem głównym pracy jest stworzenie wielostrefowego sterownika do systemu nawadniania ogrodu, działającego w oparciu o ustalony program, uwzględniając dane z bezprzewodowych czujników temperatury i wilgotności, nastaw użytkownika zlecanych lokalne lub zdalnie, a także uwzględniającego prognozy pogody. **Pierwszym celem pośrednim** jest zaprojektowanie, budowa i przeprowadzenie badań nad koncepcyjnymi czujnikami środowiskowymi i ewentualnie włączenie ich do systemu nawadniania. **Drugi cel pośredni** to implementacja dedykowanej aplikacji mobilnej i konfiguracja rutera bezprzewodowego, do bezpiecznego zdalnego sterowania przez tunel VPN przy wykorzystaniu protokołu komunikacyjnego IPv4.

Praca została podzielona na 5 rozdziałów, dobrze wpisujących się w fazy powstawania produktu końcowego. Pierwszy rozdział rozpoczyna się od przeglądu składowych systemów nawadniania ogrodów opartych o urządzenia elektroniczne. Następnie przybliżona zostaje harmonogramowanie projektu metodą ścieżki krytycznej, a dalej zostają scharakteryzowane platformy z mikrokontrolerami do sterownika i czujnika strefowego. W kolejnym kroku omówiono zastosowane technologie informatyczne, przy podkreśleniu protokołów komunikacyjnych i sposobów ich zabezpieczenia. Na zakończenie początkowego rozdziału opisane zostały wykorzystane w projekcie struktury danych.

W rozdziale drugim zostaje wprowadzony autorski algorytm do obsługi sterownika oraz stref nawadniania OptiServ. Po definicji pojęć podstawowych zostaje zaprezentowany jego

schemat blokowy, a następnie przedstawione warunki danych wejściowych, założenia oraz kryteria oceny rozwiązań algorytmu. Następnie opisano przeprowadzone symulacje implementacji algorytmu wraz ze stworzoną przeglądarką rozwiązań, by ostatecznie poddać je analizie i ocenie.

Trzeci rozdział rozpoczyna się od analizy czynników ryzyka, ukazując ostatecznie harmonogramu projektu. Następnie w odniesieniu do zebranych wymagań projektu przedstawiona jest jego specyfikacja wymagań. Staje się ona podstawą do stworzenia i przedstawienia modeli systemu nawadniania i jego urządzeń. Kolejną fazą przedstawioną w tej części pracy jest projektowanie oraz przedstawienie schematów elektrycznych urządzeń, płytek PCB oraz obudów. Rozdział zostaje zakończony opisaniem stworzonych scenariuszy testowych i koncepcji środowiska, w którym mają być przedstawione testy.

W kolejnym, czwartym rozdziale, przedstawiono plan eksperymentów pomiarowych związanych z elementem wykonawczym czujnika wilgotności gleby oraz architekturę środowiska doświadczalnego. Szczegółowo opisane i wyjaśnione zostają przeprowadzone 4 eksperymenty, po czym następuje analiza wyników i ocena przydatności zbudowanego czujnika wilgotności i temperatury gleby w ramach weryfikacji hipotezy badawczej.

W ostatnim rozdziale zawarto informacje o fazie realizacji projektu, zaczynając od budowy urządzań, poprzez implementację algorytmów czujnika i sterownika, uwzględniając implementację ich interfejsów, kończąc na aplikacji mobilnej. Piąty rozdział zostaje zwieńczony poprzez raport z testów sprzętu i oprogramowania, a także z realizacji scenariuszy testowych.

1. CHARAKTERYSTYKA DOMENY NAWADNIANIA, OMÓWIENIE KOMPONENTÓW ELEKTRONICZNYCH ORAZ TECHNOLOGII INFORMATYCZNYCH

1.1 Składniki systemu nawadniania ogrodu. Cel nawadniania

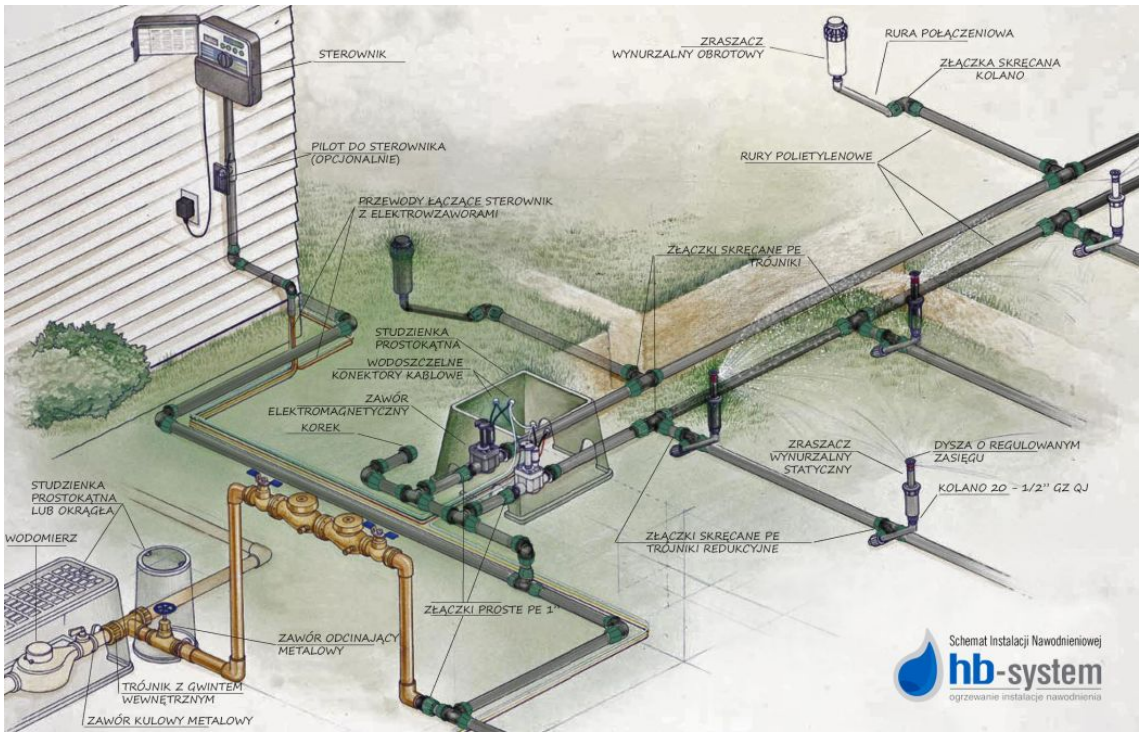
Podobnie jak w infrastrukturze lokalnych sieci komputerowych, w systemie nawadniania można dokonać podziału na elementy pasywne i aktywne. Co więcej, za odseparowane od siebie fizycznie lub logicznie podsieci z hostami i użytkownikami, w domenie irygacji odpowiadają sekcje nawodnieniowe różniące się uwarunkowaniami fizycznymi wraz roślinami o odmiennym zapotrzebowaniu na wodę. Kontynuując porównanie, szerokość pasma i przepustowość odpowiadają maksymalnej i osiągalnej przepływności cieczy, co znajduje odzwierciedlenie w średnicy i materiale z jakich wykonana jest infrastruktura systemu, aby efektywnie spełniać zdefiniowane zadania. Kończąc podobieństwa i zauważając podstawową różnicę, w sieciach komputerowych dochodzi do dwukierunkowej transmisji energii, w systemach irygacyjnych zaś transportu, jednej z najczęściej występującej cząsteczki we Wszechświecie[4] – wody.

Według dokonanego podziału, do składowych pasywnych systemu irygacyjnego, według autora zalicza się rury, złączki i kształtki połączeniowe, które ze względu na dobre parametry fizyczne, odporność chemiczną i biologiczną, są najczęściej wykonywane z polietylenu[5]. Zgrzane lub połączone mechanicznie, tworzą szkielet systemu nawadniania, który zwieńczony jest urządzeniami nawadniającymi – zraszaczami. W zależności od rodzaju roślinności, a także zagospodarowania i aranżacji terenu, stosuje się zraszacze, mikrozraszacze, a w przypadkach wymagających wyższej precyzji nawadniania: linie kroplujące lub kroplowniki indywidualne[5]. Innym nieodzownym elementem dobrze działającego systemu są filtry, oczyszczające mechanicznie eksploatowaną wodę i zapobiegające niedrożności końcowych elementów instalacji.

Natomiast drugą grupę, która można określić mianem aktywnej, stanowią: pompy, zawory, sterowniki i wyłączniki nawadnia. Pompy są maszynami przepływowymi, zamieniającymi energię zewnętrzną na energię cieczy przez nią przepływającą, za pomocą organu roboczego np. wirnika[6]. Sterowanie nawadnianiem jest realizowane ręcznie, za pomocą zaworów kulowych lub automatycznie, za pomocą zaworów elektromagnetycznych. Sygnał sterujący dla tych ostatnich ma źródło w sterowniku, którego zadaniem jest zarządzanie procesami nawadniania poszczególnych sekcji nawodnieniowych według wcześniej przypisanego programu lub ręcznego wyzwolenia przez użytkownika. Pojęcie wyłącznika nawadniania przypisuje się czujnikom np. opadów lub wilgotności gleby, których sygnał ma możliwość przerwania dostarczania wody, najczęściej w sytuacji opadów atmosferycznych.

Należy podkreślić, że dobór pomp, materiałów i wielkości przekrojów szkieletu systemu nawodnieniowego, podobnie jak rodzajów i rozmieszczenia zraszaczy, stanowi przedmiot projektu systemu nawadniania i jest poza zakresem tego opracowania. Jednak dla lepszego

zobrazowania całości zagadnienia na poniższym rysunku nr 1 przedstawiono przykładowy schemat instalacji nawodnieniowej.



Rysunek 1: Schemat instalacji nawodnieniowej[5]

Niezmierne istotne jest określenie celu nawadniania. W pierwszym przybliżeniu, cel może być określony jako dostarczenie wody do systemu korzeniowego roślin. Jednakże, jak pokazują badania, musi to być woda w odpowiedniej ilości, aby uzyskała odpowiedni potencjał, tak aby rośliny miały energię ją wchłonąć i włączyć do obiegu zapewniając warunki do prawidłowej wegetacji. Ostatecznie, **celem systemu nawaniania** jest optymalne wykorzystanie energii i wody, dostarczając jej potrzebną ilość w odpowiednie miejsce w odpowiednim czasie[7].

Realizacja wyznaczonego celu może się odbyć w dwóch trybach[8]:

1. uruchamianie nawadniania na żądanie – celem jest utrzymanie wilgotności gleby w zadanym przedziale,
2. uruchamianie nawadniania zgodnie z harmonogramem (z możliwością jego pominięcia lub skrócenia na podstawie danych środowiskowych).

Z racji celu głównego i celów pobocznych pracy, uwaga w kolejnych podrozdziałach zostanie skupiona na części grupy aktywnej urządzeń tj. sterownikach, elektrozaworach i czujnikach środowiskowych.

1.1.1 Sterowniki

Sterowniki[9,10,11,12] pozawalają na automatyzację procesu nawadniania poprzez zmianę stanu zaworów elektromagnetycznych i sterowanie dopływem wody do poszczególnych obszarów ogrodu, zgodnie z wcześniej ustalonym programem. Nawadnianie poszczególnych

sekcji może być zaprogramowane indywidualnie lub sekwencyjnie. Dla każdej wydzielonej części obszaru nawadniania są dostępne są opcje startu: w wybrane dnie tygodnia, parzyste lub nieparzyste dni kalendarzowe lub cyklicznie co wybraną ilość dni. Każdego dnia powinna być dostępna możliwość kilku niezależnych startów wraz z przypisaniem długości działania programu.

Tworzenie harmonogramów zazwyczaj odbywa się lokalnie przez panel urządzenia. W niektórych modelach można tego dokonać zdalnie poprzez aplikację mobilną zainstalowaną na urządzeniu przenośnym, przy wykorzystaniu bezprzewodowej sieci komputerowej. W razie potrzeby lub awarii powinna istnieć możliwość ręcznego sterowania sekcjami nawodnieniami. Cechą części kontrolerów jest opcja ręcznego zawieszenia wykonywania harmonogramu przez określony czas (np. ze względu na obfite opady deszczu). Bardzo przystępną użytkownikowi funkcjonalność daje możliwość kontrolowania sterownika przy pomocy poleceń głosowych przez usługi Amazon Alexa, Apple HomeKit, and Google Assistant oraz integracji z systemami inteligentnej automatyki domowej przez IFTTT, Wink, Control4 lub Nexia[12].

Opisywany typ urządzeń jest zasilany napięciem sieciowym 230V lub bateryjnie 9V. Data i czas najczęściej są podtrzymywane w razie zaniku zasilania poprzez dodatkową baterię wewnętrzną, natomiast harmonogram pracy poprzez zapis np. w pamięci nieulotnej EEPROM.

Niektóre sterowniki mają możliwość pobierania prognoz pogody z internetowych serwisów meteorologicznych i na podstawie tych danych optymalizować harmonogram nawadniania ze względu na zużycie wody. Często dostępna jest możliwość podłączenia do kontrolera innych urządzeń wpływających na decyzje dotyczące irygacji, takich jak czujniki: wiatru, deszczu, mrozu, wilgotności gleby lub przepływu cieczy.

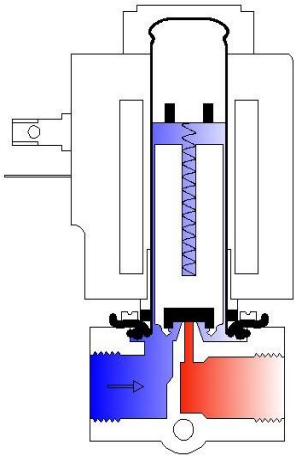
1.1.2 Elektrozawory

Elektrozawory nawadniające są jednym z najważniejszych elementów systemu irygacyjnego, ponieważ ich zadaniem jest otwieranie i zamykanie przepływu wody, na podstawie sygnału elektrycznego ze sterownika. Wynika stąd potrzeba zrozumienia zasady ich działania w celu prawidłowej aplikacji i użytkowania w tworzonym systemie.

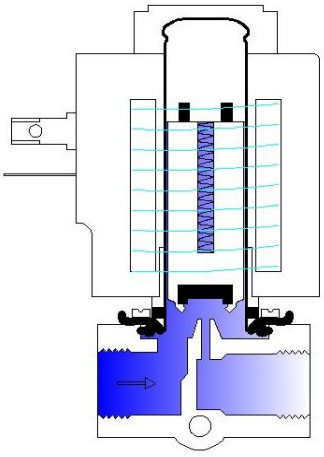
Elektrozawór składa się z cewki i korpusu z zaworem. Na rysunkach nr 2 i 3 przedstawiono przekroje przez elektrozawór odpowiednio w stanie zamkniętym i otwartym[13]. W pierwszym stanie ruchomy rdzeń (tłok) blokuje przepływ cieczy, natomiast w drugim zasilona cewka elektromagnesu odciąga rdzeń, umożliwiając przepływ medium.

Z ogólnego punktu widzenia istotnymi parametrami zaworów są wielkość przyłącza, dopuszczalna temperatura i ciśnienie przenoszonego medium oraz jego rodzaj[13]. Natomiast z punktu widzenia sterowania kluczowymi parametrami są napięcie i moc cewki, w uzupełnieniu zaś wielkość natężenia prądu rozruchowego oraz natężenia prądu podtrzymania. Na rynku dostępne są elektrozawory na prąd stały (np. 9V DC, 12V DC) lub prąd zmienny (24 V AC), jednak jak wynika analizy dostępnej oferty, ten drugi rodzaj jest przeważający. Jednym z powodów, uzasadniający taki stan rzeczy jest z pewnością porównanie mocy rozpraszanej

w cewce, której rezystancja przyjmować może wartość w granicach 30-40Ω. Wartość natężenia prądu stałego w takim obwodzie można oszacować w przedziale I=12V/40Ω=0,3A oraz I=12V/30Ω=0,4A. Wielkości natężeń prądów podtrzymujących w obwodach prądu zmiennego, jak wynika z tabeli nr 1, mogą być ponad 2 razy mniejsze (dla zbliżonych wartości rezystancji cewki). To z kolei prowadzi do wniosku, że moc rozpraszana w cewce jest nawet do 4 razy mniejsza, co z potencjalnie ma wpływ na jej długość bezawaryjnego działania. Wytłumaczeniem tak znacznej różnicy jest reaktancja indukcyjna cewki, stawiająca dodatkowy opór bierny indukcyjny przy pobudzeniu prądem zmiennym[14].



Rysunek 2: Zawór zamknięty - brak napięcia na cewce[13]



Rysunek 3: Zawór otwarty - zwora wciągana przez siłę elektromagnetyczną do cewki[13]

Tabela 1: Zestawienie wybranych elektrozaworów o napięciu cewki 24V i częstotliwości 50Hz.

Lp.	Model elektrozaworu	Φ_w [cal]	RP	Ciśnienie [bar]	Przepływ [m^3/h]	I_{inr} [A]	S_{inr} [VA]	I_{hol} [A]	S_{hol} [VA]
1.	Bermad IR-21T G	1	T	0,7-10,0	b. d.	0,25	b. d.	0,13	b. d.
2.	Bermad IR-110	3	T	0,5 -10.0	max: 60	0,25	b. d.	0,13	b. d.
3.	Rain Bird 100 DV	1	T	1,0-10,4	0,24-9,00	0,3	7,2	0,19	4,6
4.	Rain Bird LFV-075	¾	N	1,0-10,3	0,05-1,82	0,25	7,2	0,14	4,56
5.	Rain Bird PGA-150	2	T	1,0-10,4	6,00-21,0	0,41	9,9	0,23	5,5
6.	IRRITROL EURO-F	1	N	0,5-12,0	1,20-9,60	0,34	b. d.	0,27	b. d.
7.	Hunter PGV-201B	2	T	1,5-10,0	4,50-34,0	0,4	9,6	0,19	6,5
8.	Rain RN160	1½	T	1,0-12,0	6,00-19,5	0,48	b. d.	0,2	4,8
9.	TORO EZ-Flo Plus	1	T	0,7-10,0	0,25-30	0,34	11,5	0,2	5,75
10.	HydroSure Pro-Series 150	1	T	0,7-10,3	0,06-6,83	0,43	b. d.	0,25	b. d.

Φ_w - średnica wewnętrzna gwintu przyłącza, RP - możliwość ręcznej regulacji przepływu.
Źródło: opracowanie własne na podstawie[15,16,17,18,19,20,21], przedstawione elektrozawory dostępne były na europejskim rynku w 2021r.

W ramach przygotowania koncepcyjnego do stworzenia sterownika nawadniania, w powyżej zamieszczonej tabeli nr 1 przygotowano zestawienie parametrów charakterystycznych wybranych elektrozaworów wiodących producentów, zakładając parametry pracy cewki zaworu pod napięciem U=24V AC oraz częstotliwość f=50Hz. Jeżeli dostępna jest opcja regulacji przepływu to ustawia się ją w sposób ręczny.

1.1.3 Czujniki i parametry środowiskowe

Do otwartego układu sterowania systemu nawodnieniowego często wprowadza się czujniki pomiarowe, aby móc zmienić sygnał sterowania w reakcji na zakłócenia lub inne zdarzenia zewnętrzne. Taka konfiguracja pozwala dostosować zużycie wody do rzeczywistej sytuacji środowiskowej bądź pełni rolę ochronną poszczególnych składników systemu.

Jednym z przykładów są czujniki wiatru w systemach ze zraszaczami lub deszczowniami, które w przypadku znacznych prędkości powietrza mogą spowodować przerwanie nawadniania, aby ograniczyć znoszenie strug wody poza wyznaczony obszar.

Z kolei zastosowanie detektora deszczu pozwala ograniczyć lub ominąć trwający lub najbliższy zaplanowany proces podlewania, po wykryciu opadów atmosferycznych. Minimalna, aktywująca wyjście czujnika, wielkość opadu zazwyczaj jest na ustalonym poziomie, jednak niektóre urządzenia posiadają funkcję bezzwłocznego wystawienia sygnału wykrycia opadu[22]. Układ pomiarowy w tych urządzeniach składa się z higroskopijnych dysków, pęczniejących pod wpływem wody i przerywających, normalnie zamknięty, obwód elektryczny. Wyparowanie wody, skutkuje zmniejszeniem objętości dysku i przywrócenie obwodu w stan przewodzenia.

Na rynku dostępne są również wyżej wymienione czujniki w postaci zintegrowanej, nazywane stacjami pogodowymi. Wyniki pomiarów dostarczane do sterownika pozwalają na dostosowanie programu nawadniania do aktualnie panujących warunków atmosferycznych.

Natomiast pomiar ilości wody w jednostce czasu za pomocą przepływomierza zamontowanego ma głównej magistrali wodnej lub w prostszym układzie - binarne wykrycie tego przepływu - może pozwolić na monitoring anomalii, sprawne wykrywanie nieprawidłowości i możliwość reakcji na nie.

Inną zmienną środowiskową, której znajomość pozwala na ochronę instalacji nawodnieniowej jest temperatura przy powierzchni gleby. Wynik poniżej ustalonej temperatury krytycznej (np. 0.5°C, 3°C lub 5°C)[23] powinien wstrzymać nawadnianie chroniąc instalację przed oblodzeniem i wygenerować powiadomienie o zdarzeniu niepożądanym. Aczkolwiek część hodowców truskawek i jagód, ustawiając odpowiednio duży przepływ wody, stosuje system zraszaczy w temperaturach ujemnych (do około -5°C), aby chronić swoje uprawy przed wiosennymi mrozami[24].

Poza bezpośrednimi pomiarami, do zmniejszenia zużycia wody wykorzystuje się również prognozy parametrów meteorologicznych. Potencjalna wielkość opadów określona z wysokim prawdopodobieństwem jest dobrą przesłanką do zmniejszenia czasu najbliższego nawadniania. W odmiennej sytuacji zaś, duże prawdopodobieństwo upału, daje przyczynek

do pełnego wykonania programu i ograniczenie nawadniania podczas wysokich temperatur, w celu uniknięcia strat na zwiększone parowanie i przede wszystkim ograniczenia szoku termicznego dla roślin. Z kolei prędkość wiatru wraz z temperaturą powietrza, ma zaś swój wkład w istotny parametr agrometeorologiczny - ewapotranspirację wskaźnikową, która określa potencjalne możliwości parowania (mm/dzień)[25]. Uzyskując te dwie zmienne środowiskowe z prognozy pogody, można oszacować parowanie wody i odpowiednio zmodyfikować harmonogram nawadniania, celem ograniczenia start wody.

Wszystkie wymienione powyżej parametry i odpowiadające im urządzenia pomiarowe mają na celu oszczędności w zużyciu wody lub ochronę infrastruktury systemu nawadniania. Urządzeniami, które jako pierwsze są w stanie dostarczyć użytecznej informacji jak proces sterowania wpływa na osiągnięcie celu nawadniania są czujniki wilgotności gleby. Najbardziej należy wyróżnić jednak tensjometry, których wyniki niosą informację o sile ssącej gleby, symulując możliwości transportu wody na granicy korzenia i gleby. Jest to zbieżne z celem nawadniania w postaci dostarczenia użytecznej wody do systemu korzeniowego roślin.

1.1.3.1 Stan wodny gleby i metody jego określania

Do określenia stanu wodnego gleby używa się dwóch zupełnie odmiennych wielkości[26]:

a) wilgotności gleby Θ [% obj.] - wolumetryczna zawartość wody w glebie,
b) potencjału wodnego gleby Ψ [kPa, N/m2] - stan energetyczny wody jako wynik działania sił molekularnych, osmotycznych i grawitacyjnych.

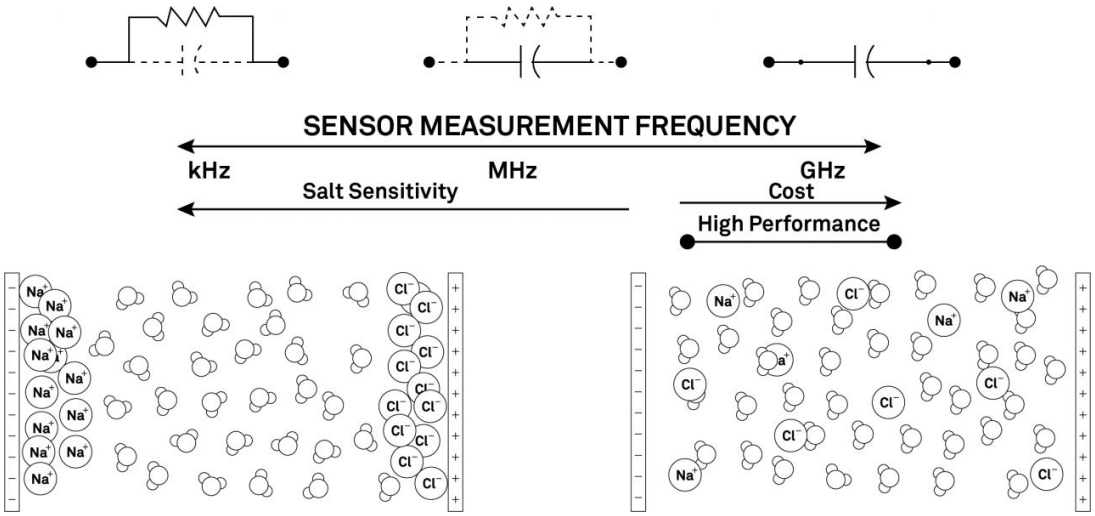
Pierwsza z wymienionych wielkości jest rodzaju ekstensywnego – określa zawartość wody w glebie. Druga zaś to zmienna intensywna, wskazująca na ilość wody dostępną do wykorzystania przez rośliny.

Znane są następujące metody *in situ* pomiaru stanu wodnego gleby[27,28]:

1. Tensjometryczna – mierząca potencjał wodny gleby poprzez pomiar podciśnienia w zamkniętej rurce z wodą (kontakt z ziemią przez ceramiczną, porowatą substancję, umożliwiają wypływ i kapilarne podciąganie wody).
2. Rezystancyjna – wykorzystująca zjawisko zmiany oporu gleby w zależności od zawartości wody, podczas przepływu ładunku elektrycznego pomiędzy umieszczonymi w podłożu elektrodami.
3. Dielektryczna (opierająca się na różnicy rzędu wielkości stałej dielektrycznej wody i gleby, przekładającej się na pośrednie wartości pojemności elektrycznej mieszaniny składników)
    a) pojemnościowa – wykorzystanie pojemności elektrycznej gleby i ocenę napięciowej odpowiedzi układu pomiarowego na okresowe pobudzenie; część pojemności elektrycznej układu zależy w dużym stopniu od otoczenia,
    b) reflektometria czasowa (TDR) - pomiar czasu propagacji impulsu w prętach otoczonych glebą,

c) reflektometria częstotliwościowa (FDR) - wyznaczenie częstotliwości rezonansowej w obwodzie elektrycznym i powiązanie jej z zawartością wody w podłożu.

4. Neutronowa, opierająca się na wykorzystaniu pierwiastków promieniotwórczych, wykorzystująca zjawisko spowalniania neutronów przez pierwiastki lekkie (np. atomy wodoru w cząsteczkach wody)
5. Przewodnictwa cieplnego.

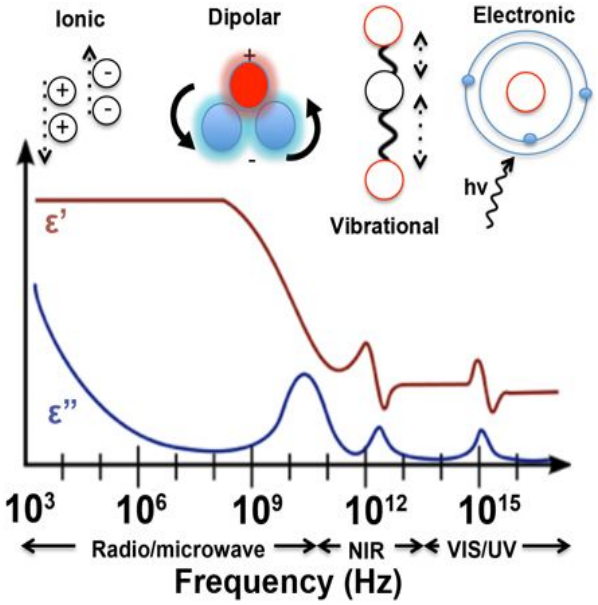


Rysunek 4: Wpływ częstotliwości sygnału pomiarowego na jony zawarte w wodzie oraz charakter elektryczny obwodu[29]

Spośród wymienionych technik, optymalną do zastosowania w bezprzewodowych czujnikach wilgotności, na podstawie których ma być podjęta decyzja o nawanianiu, jest wymieniona w punkcie 3a - pojemnościowa metoda pomiaru przenikalności dielektrycznej gleby ze względu na[29]:

a) odpowiednią do zastosowania dokładność,
b) stosunkowo nieduży koszt konstrukcyjny czujnika,
c) niski wydatek energetyczny pojedynczego pomiaru,
d) odporność wyniku pomiaru na obecność soli w glebie,
e) niewielki wpływ na środowisko pomiarowe,
f) bezobsługową pracę.

Wśród czujników implementujących metodę dielektryczną, bezwzględnie należy zwrócić uwagę na częstotliwość sygnału jaki generują. Wyniki pomiarów dostarczane przez tańsze modele pojemnościowe, wykorzystujące sygnały o pulsacji rzędu kiloherców, wykazują charakter silnie rezystancyjny (rysunek nr 4), polaryzując w trakcie pomiarów nie tylko cząsteczki wody, ale również zawarte w niej jony. Objawiać się to może niewłaściwymi wynikami, a poza tym wprowadza zmiany w środowisku pomiarowym. To niepożądane zjawisko traci na znaczeniu wraz ze zwiększaniem częstotliwości do rzędu wielkości kilku MHz, natomiast jest pomijalnie małe powyżej 50MHz[29]. Zależności te zostały wydatnie zobrazowane na poniższym rysunku nr 5, ukazującym zmianę wartości rzeczywistej i urojonej składowych przenikalności dielektrycznej dla wody w zależności od częstotliwości.



Rysunek 5: Widmo przenikalności dielektrycznej wody w zakresie częstotliwości[30]

1.2 Harmonogramowanie projektu przy użyciu metody ścieżki krytycznej

Metoda ścieżki krytycznej (z ang. *Critical Path Method*, w skrócie CPM) zalicza się do deterministycznych technik planowania sieciowego[31]. Jest to technika posługująca się grafową reprezentacją projektu i służy do ich planowania i kontroli[32]. Tworzony w niej wykres sieciowy, będący grafem skierowanym[33], przedstawia niezbędne czynności oraz zdarzenia definiujące projekt, a na jego podstawie dokonuje się wyliczeń wyznaczających plan realizacji przedsięwzięcia. Metoda CPM przedstawia więc w sposób graficzny powiązany zbiór działań, które zmierzają do realizacji postawionego celu w określonym czasie, przy dostępnych zasobach i przeznaczonym budżecie.

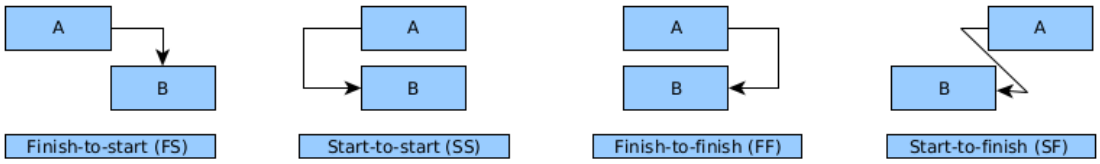
W CPM stosuje się dwa typy modeli grafowych[32]: AON (ang. *activity on node*) lub AOA (ang. activity on arrow). W pierwszym z modelu wierzchołki oznaczają zadania do wykonania, a łuki reprezentują zależności kolejnościowe tych zadań. W drugim modelu zaś wierzchołki reprezentują stany realizacji projektu, łuki zaś operacje do wykonania. Jako symbol węzła stosuje się koła lub prostokąty, natomiast do oznaczenia krawędzi grafu – strzałki[31]. Linia ciągła strzałki symbolizuje czynność prostą, zużywającą zasoby i czas. Natomiast linia przerywana oznacza czynność pozorną, ukazującą jedynie zależności kolejnościowe, przy których czas oraz zasoby nie są zużywane.

Aby wykorzystać metodę ścieżki krytyczniej w planowaniu określonego przedsięwzięcia konieczne jest wykonanie szeregu czynności przygotowawczych. Należy do nich ustalenie ilości i listy zadań składających się na cały projekt, określenie kolejności ich realizacji i czasu trwania każdego z nich. Ostatecznie dane te należy wykorzystać do stworzenia wykresu sieciowego, odwzorowującego i określającego powiązania wyszczególnionych wcześniej operacji.

Obliczenia wykonywane w metodzie wykonywanie są w dwóch fazach: w przód oraz wstecz. W pierwszej części kalkuluje się najwcześniejsze momenty rozpoczęcia (ES) i zakończenia(EF) operacji. W drugiej, wstecznej fazie, gdzie graf jest odwiedzany przeciwnie do zwrotów strzałek na krawędziach, oblicza się najpóźniejsze momenty rozpoczęcia (LS) i zakończenia (LF) operacji. Wyniki zapisywane są w wierzchołkach grafu i mogą zostać zaprezentowane zgodnie z rysunkiem nr 6, który przedstawia strukturę takiego wierzchołka.

Rysunek 6: Struktura wierzchołka grafu sieciowego w metodzie ścieżki krytycznej

Podstawowym typem zależności określającym wzajemne powiązania między stanami lub zadaniami projektu jest zależność koniec do początku (z ang. *finish to start*, FS). Oznacza ona, że aby przykładowa czynność „B" mogła się rozpocząć, czynność „A" musi się zakończyć. Poza tym wyróżnia się jeszcze typy zależności: *start to start (SS)*, *finish to finish* (FF) i *start to finish* (SF). W zależności typu SS, zadanie „A" może rozpocząć się wtedy, gdy rozpoczyna się „B" – co pozwala na ich równoczesny start. Przeciwnie zaś ustalono w typie FF, zadanie „B" może się kończyć dopiero gdy, zakończy się zadanie „A". Ostatni rodzaj SF, określa zależność w której czynność „B" nie może się zakończyć, przed rozpoczęciem czynności „A" – wymagane jest więc współistnienie obu przez pewien określony czas. Wymienione powiązania pomiędzy zadaniami zostały przedstawione w sposób graficzny na rysunku nr 7.



Rysunek 7: Powiązania kolejnościowe pomiędzy zadaniami

Wśród korzyści i celów stosowania metody ścieżki krytycznej wyróżnia się:[32,35]

a) możliwość wyliczenia czasu zakończenia projektu,

b) wyznaczenie operacji krytycznych, których każde opóźnienie powoduje opóźnienie planowanego przedsięwzięcia,

c) wyznaczenie operacji niekrytycznych i określenie ich możliwego opóźnienia nie wpływającego na czas zakończenia,

d) zarządzanie projektem ścieżki krytycznej ułatwia proces oceny ryzyka,

e) analiza ścieżki krytycznej daje realny pomiar postępów projektu.

Przed rozpoczęciem dalszych prac wykonany został harmonogram projektu na podstawie którego powstał diagram Gantta oraz graf sieciowy dotyczący realizacji sterownika nawadniania - zaprezentowany w załącznikach D i E do niniejszej pracy.

1.3 Charakterystyka mikrokontrolerów wybranych do sterownika i czujnika

Z uwagi na potrzeby wynikające ze specyfikacji wymagań z podrozdziału 3.2, jako główny układ sterownika i czujnika została wybrana ta sama platforma ESP32-WROOM-32E. Mimo, że do czujnika wilgotności wstępnie zakładano wybór mniej złożonego układu z rodziny Atmega, to uwzględniając ilość dodatkowych elementów elektronicznych, całkowitą cenę, dostępność i możliwość rozwoju w przyszłości dodatkowych funkcjonalności w ramach tylko samej aktualizacji oprogramowania korzystniejsza okazała się platforma ESP32. Argumenty ekonomiczne i techniczne przemawiające za tym wyborem w odniesieniu do czujnika to:

a) cena poniżej 15zł za moduł,
b) dobra dostępność w sklepach z komponentami elektronicznymi[36],
c) 26 wejść/wyjść ogólnego przeznaczenia (GPIO),
d) zintegrowany oscylator kwarcowy - możliwością generowania impulsów prostokątnych z częstotliwością 40MHz i natężeniem prądu powyżej 20mA,
e) sprzętowe wspomaganie obliczeń podczas stosowania algorytmów kryptograficznych AES i RSA,
f) zintegrowane układy komunikacyjne WiFi 802.11b/g/n i Bluetooth V4.2 BR/EDR,
g) niski pobór mocy w stanie głębokiego uśpienia o wartości ok. 10µA,
h) możliwość aktualizacji oprogramowania układowego przez użytkownika (OTA).

W pierwszej wersji czujnika wilgotności gleby układy komunikacyjne nie są wykorzystywane, jednakże ich obecność pozwoli zwiększyć funkcjonalność urządzeń jedynie za pomocą aktualizacji oprogramowania układowego. Zazwyczaj wiąże się to z wykorzystaniem komputera oraz otwarciem obudowy urządzenia przez użytkownika, podłączenia go komputera oraz wykonaniem programu aktualizującego. Dzięki funkcji over-the-air (OTA), wykorzystując możliwość podziału dostępnej przestrzeni magazynowej flash na partycje, aktualizacja jest możliwa bez konieczności nawiązywania połączenia szeregowego z ESP32. Te same zastrzeżenia dotyczą sterownika, w którym nieużywany pozostaje moduł Bluetooth.

1.3.1 Charakterystyka ESP32-WROOM-32E z mikroprocesorem ESP32-D0WD-V3

ESP32 stanowi bogatą rodzinę układów typu SoC produkowanych przez chińską firmę Espressif. Najistotniejszymi cechami, którymi różnią się poszczególne modele są[37]: procesor CPU, wielkość pamięci RAM, SRAM i flash, ilość dostępnych portów GPIO oraz dostępne interfejsy. Wśród danego modelu wyróżnia się jeszcze układy z anteną zintegrowaną lub zewnętrzną, podłączaną poprzez złącze typu U.FL. Najistotniejsze parametry, wybranego do projektu modelu ESP32-WROOM-32E, pochodzące z karty katalogowej, zostały zebrane i syntetycznie przedstawione w tabeli nr 2.

Tabela 2: Parametry układu ESP32-WROOM-32E[38]

Procesor CPU	Xtensa dual-core 32-bit LX6 microprocessor, Taktowanie do 240 MHz (zintegrowany chip ESP32-D0WD-V3)
Pamięć RAM	520 KB SRAM 16 KB SRAM w RTC
Pamięć ROM	448 KB ROM
Porty GPIO	26 (w tym 18 RTC GPIO)
WiFi	Standard 802.11 b/g/n 2,4 GHz, Prędkość transmisji do 150 Mb/s, Tryby pracy Wi-Fi: STA, SoftAP, SoftAP+STA, P2P zabezpieczenia: WEP, WPA/WPA2, PSK/Enterprise
Bluetooth	Bluetooth V4.2 BR/EDR Bluetooth LE
Układy peryferyjne	LED PWM (LEDC) Licznik impulsów PCNT Czujnik dotyku Czujnik Halla 12 kanałowy konwerter ADC (rozdzielczość: 12bit), 2 kanałowy konwerter DAC (rozdzielczość: 8bit)
Interfejsy	UART, SDIO, SPI, I2C, I2S
Zintegrowane komponenty	Oscylator kwarcowy 40 MHz Pamięć Flash 4/8/16 MB SPI
Warunki pracy	Zasilanie: 3.0 ~ 3.6 V Temperatura: -40 ~ 85 °C
Wymiary	18.0 × 25.5 × 3.1mm
Certyfikaty	Bluetooth: BQB, RF: FCC/CE-RED/SRRC, Green: REACH/RoHS
Aktualizacja uprogramowania	UART OTA

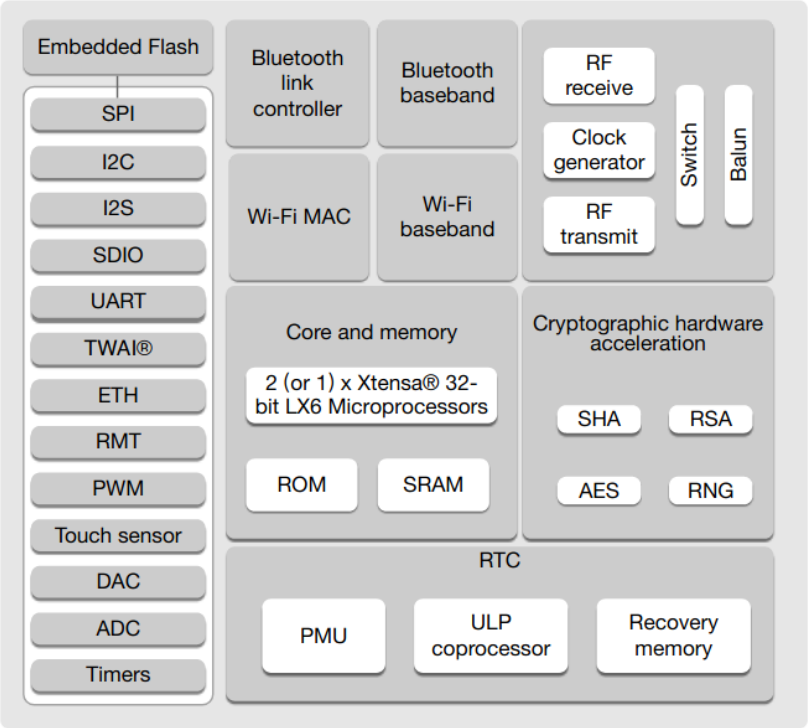
Tabela 3: Tryby oszczędzania energii ESP32[39]

Lp.	Nazwa trybu	Zużycie energii	Komponenty aktywne	Komponenty nieaktywne
1.	Active mode	160-260mA	ESP32 Core, ULP, WiFi, Bluetooth, Radio, układy peryferyjne, RTC	-
2.	Modem Sleep Mode	3-20mA	ESP32 Core, ULP, RTC	WiFi, Bluetooth, Radio, układy peryferyjne
3.	Light Sleep Mode	0.8mA	ESP32 Core(wstrzymany), ULP, RTC	WiFi, Bluetooth, Radio, układy peryferyjne
4.	Deep Sleep Mode	150µA	RTC, ULP(aktywny)	ESP32 Core, WiFi, Bluetooth, Radio, układy peryferyjne
		10µA	RTC, ULP(wstrzymany)	
5.	Hibernation Mode	2.5µA	RTC	ESP32 Core, ULP, WiFi, Bluetooth, Radio, układy peryferyjne

Legenda:
ESP32 Core – dwurdzeniowy 32-bitowy mikroprocesor wraz z pamięcią: 448 KB ROM, 520 KB SRAM i 4MB Flash.
ULP - Ultra Low Power co-processor
RTC – moduł zegara czasu rzeczywistego

Właściwe zastosowanie technologii w urządzeniach Internetu Rzeczy IOT wiąże się z należytym zabezpieczeniem komunikacji (*data in transit*), jak również danych w spoczynku (*data at rest*). W ESP32 jest do wykorzystania dedykowany sprzętowy moduł szyfrujący w postaci kooprocesora[40], wspomagający obliczenia wykonywane w kryptograficznych funkcjach haszujących SHA, symetrycznym szyfrze blokowym AES, asymetrycznym algorytmie RSA czy w generatorze liczb losowych RNG. Poza tym dostępna jest 1024 bitowa pamięć jednokrotnego zapisu[41] *One-Time Programming Read-Only Memory* (OTP). Dzięki tej pamięci i zapisanych w niej kluczach (4 bloki eFuse po 256 bitów) możliwe jest użycie funkcjonalności bezpiecznego bootowania *Secure Boot* oraz zaszyfrowanie pamięci *Flash Encryption*. Czwarty z bloków o nazwie EFUSE_BLK3, może być częściowo zajęty przez adres MAC urządzenia lub jest do wykorzystania przez uruchomiony na mikrokontrolerze program.

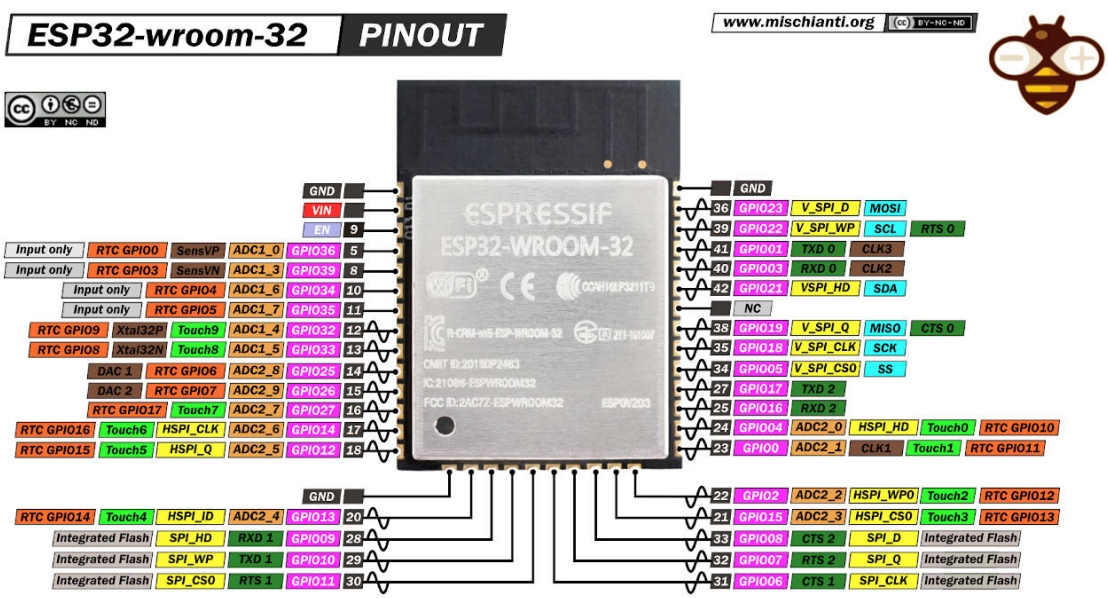
W celu optymalizacji poboru energii opisywany układ wyposażony został w mechanizm automatycznego skalowania mocy obliczeniowej w stosunku do aktualnych wymagań uruchomionej aplikacji oraz 5 trybów o zróżnicowanym poborze energii. Ich funkcjonalność wraz z szacowanym zużyciem prądu zostały przedstawione w powyższej tabeli nr 3. Poza tym posiada on odrębny blok w postaci ULP[42] (*Ultra Low Power co-processor*) do wykonywania m.in. zadań pomiarowych (temperatura, konwerter analogowo-cyfrowy ADC, zewnętrzne czujniki podłączone po magistrali I2C) w trybie głębokiego uśpienia i ultra-niskiego poboru energii. W uzupełnieniu diagram blokowy opisywanego układu uwidoczniono na rysunku nr 8.



Rysunek 8: Schemat blokowy układu ESP32[38]

Niezwykle użyteczne są dostępne na rynku są gotowe zestawy uruchomieniowe wyposażone w konwerter USB-TTL, stabilizator napięcia 3.3V oraz przyciski umożliwiające włączenie programowania mikrokontrolera oraz jego reset. Tak zbudowane platformy sprzętowe pozwalają jednocześnie zasilać i programować moduł przy użyciu kabla USB podłączonego do komputera. Poza tym linie wejścia/wyjścia ogólnego przeznaczenia (GPIO) są wyprowadzone z mikrokontrolera na obwód płytki przy użyciu szpilek goldpin o rastrze 2.54mm, co znacznie ułatwia tworzenie urządzeń prototypowych. Dla wybranego do projektu układu ESP32-WROOM-32E zobrazowanego na rysunku nr 10, oryginalną płytkę uruchomieniową stanowi platforma sprzętowa ESP32-DevKitC-32E, przedstawiona poniżej na rysunku nr 9.

Rysunek 9: ESP32-DevKitC-32E[44]



Rysunek 10: Układ ESP32-WROOM-32E wraz z opisem wyprowadzeń[43]

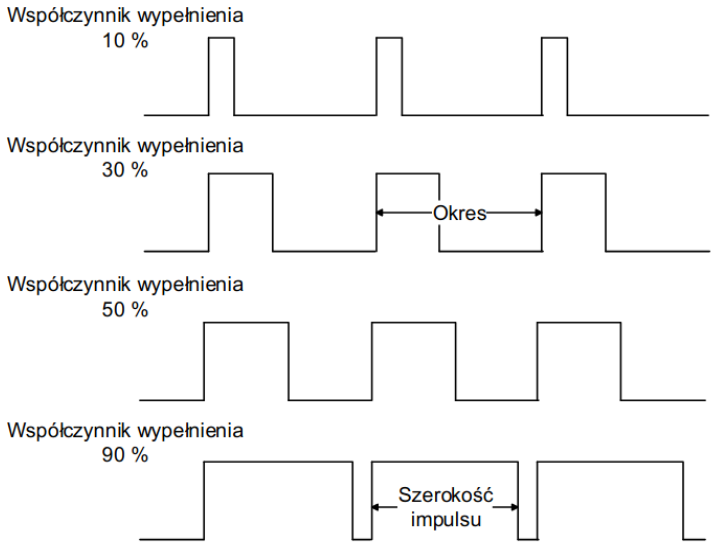
Patrząc z perspektywy oprogramowania, kod źródłowy dla układów ESP32 można tworzyć przy wykorzystaniu następujących języków[45]: Język C, język Wiring (oparty na C, znany z Arduino), LUA, Python, a nawet Javascript - dzięki projektom Espruino[46] lub Duktape[47]. Do dwóch pierwszych dostępne są kompilatory Espresiff SDK i Arduino

generujące programy w języku maszynowym. Aby skorzystać z 3 ostatnich wymienionych języków, należy najpierw wgrać do ESP32 odpowiedni firmware pełniący rolę interpretera tych języków. Skorzystanie z języków kompilowanych, wiąże się z bardziej złożoną organizacją środowiska programistycznego oraz trudniejszym debugowaniem kodu źródłowego, jednakże owocuje szybszym i zoptymalizowanym pamięciowo wykonywaniem programu na docelowej maszynie[48].

Podsumowując, dzięki dobrym parametrom technicznym, dedykowanemu zestawowi narzędzi programistycznych (SDK) od producenta i szerokiemu zestawowi peryferiów, układy z rodziny ESP32 znajdują szerokie zastosowanie. Obejmują one między innymi aplikacje związane z: automatyką domową i przemysłową, rozpoznawaniem dźwięków i obrazów, zabawkami z WiFi, elektroniką do noszenia, opieką zdrowotną, energooszczędnymi rejestratorami danych IOT oraz inteligentnym rolnictwem.

1.3.2 Urządzenia peryferyjne układu ESP32-WROOM-32E. LEDC

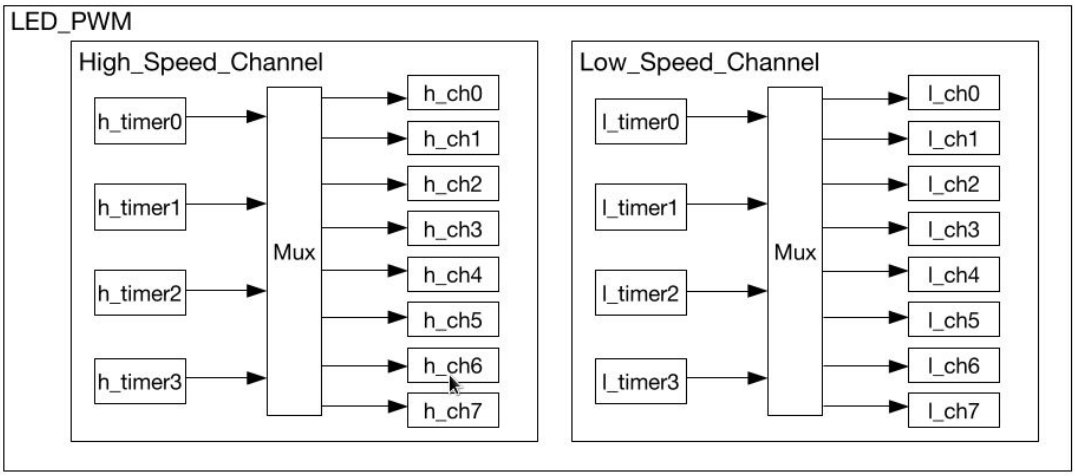
Z wielu wielu układów i interfejsów komunikacyjnych do urządzeń peryferyjnych mikroprocesorem ESP32-D0WD-V3[49], ukazanych w lewej kolumnie na rysunku nr 8, szczególną rolę w bezprzewodowym czujniku wilgotności odegrał układ *LED PWM Controller,* w skrócie LEDC. Jego natywnym przeznaczeniem jest ustalanie szerokości impulsów prostokątnych zachowując stałą częstotliwość i amplitudę[50] w celu sterowania jasnością diod LED (rysunek nr 11). Nic jednak nie stoi na przeszkodzi, aby wygenerowany sygnał wykorzystać w innym celu.



Rysunek 11: Współczynnik wypełnienia w sygnale PWM[51]

LEDC[52] posiada w sumie 16 kanałów, rozdzielonych na dwie grupy po 8 kanałów w każdej. Tylko jedna z grup potrafi pracować w trybie high-speed, który jest zaimplementowany sprzętowo, oferując automatyczną i płynną zmianę współczynnika wypełnienia impulsów.

Przeciwnie do drugiej grupy kanałów pracujących w trybie low-speed, dla których zmiana musi być dokonywana programowo przez sterownik. Architekturę kontrolera PWM przedstawiono na rysunku nr 12. Każda z grup może zostać skonfigurowana do używania innego źródła sygnału zegarowego, co zostało przedstawione w tabeli nr 4.



Rysunek 12: Architektura kontrolera PWM[52]

Tabela 4: Źródła sygnału zegarowego modułu LEDC układu ESP32[52]

Nazwa zegara	Częstotliwość	Tryb pracy
APB_CLK	80 MHz	High/Low
REF_TICK	1 MHz	High/Low
RTC8M_CLK	~8 MHz	Low

W odniesieniu do rysunku nr 12, *high-speed timer* składa się z multipleksera, w którym ustalane jest programowo źródło zegara REF_TICK albo APB_CLK. Sygnał zegarowy jest dzielony przez dzielnik *Divider*, przy użyciu dostarczanego do niego 18 bitowego współczynnika podziału LEDC_DIV_NUM_HSTIMERx (10 najstarszych bitów dla części całkowitej i 8 najmłodszych na część ułamkową). Tak podzielony sygnał zegarowy zostaje podany do 20-bitowego licznika *Counter,* generującego przerwanie w momencie osiągnięcia przez licznik maksymalnej wartości LEDC_HSTIMERx_LIM. Zgodnie z tym rysunkiem, licznik ten może być programowo odczytywany, zawieszany lub zerowany. Sygnałem wyjściowym całego *high-speed timer* jest właśnie 20 bitowa wartość generowana przez wspomniany licznik. Częstotliwość tego sygnału staje się częstotliwością bazową dla wszystkich kanałów programowo podłączonych do tego timera (rys. 12), natomiast częstotliwość na wyjściu kanału zależy jeszcze od wartości LEDC_HSTIMERx_LIM – mającej sens rozdzielczości współczynnika wypełnienia. Rozdzielczość z jaką można dokonać ustawienia współczynnika wypełnienia może przyjąć wartość N=1 do N=16 bitów, co przekłada się na zakres wartości całkowitych z przedziału $<0;2^n-1>$. Ostatecznie, częstotliwość generowanych impulsów zależy od wartości rozdzielczości, zgodnie z poniższym wzorem:

$$f_{out} = \frac{f_{CLK}}{DIV \cdot 2^N} \quad (1)$$

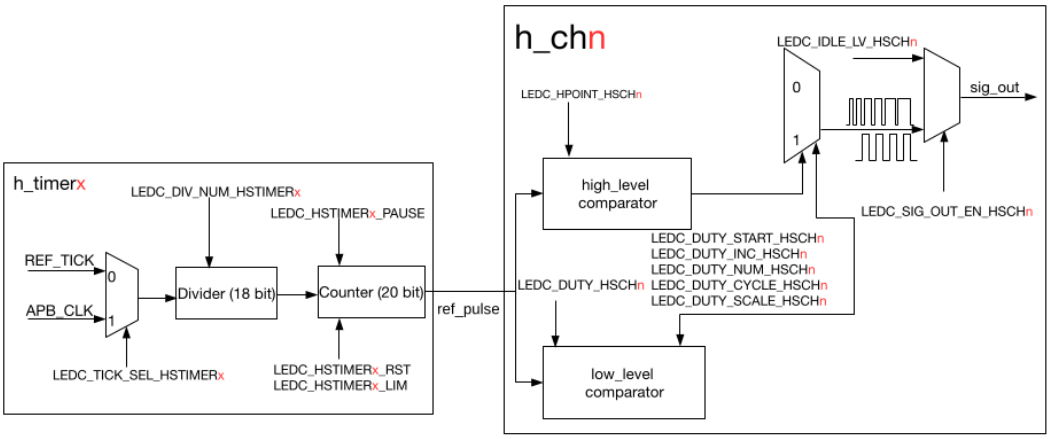
Gdzie f_{CLK} to częstotliwość sygnału zegarowego (*frequency*),
DIV – podzielnik LEDC_DIV_NUM_HSTIMERx (*width*),
N – rozdzielczość współczynnika wypełnienia LEDC_HSTIMERx_LIM (*resolution*).

Kanał ***n*** pobiera 20-bitową wartość generowaną przez licznik wybranego *high-speed* timera ***x*** i dokonuje jej porównania ją z dwoma parametrami przypisanymi do kanału: LEDC_HPOINT_HSCHn i LEDC_DUTY_HSCHn. Wyjście kanału jest w stanie wysokim, gdy wartość wyjścia licznika jest pomiędzy pierwszą z wymienionych wyżej wartości a ich sumą, w przeciwnym wypadku przyjmuje stan niski. W ten sposób sterowany jest stan wyjścia kanału, którego budowa przedstawiono na rys. 13.

Stąd, aby wygenerować impulsy o jak największej częstotliwości, w nawiązaniu do widma przenikalności dielektrycznej wody z rys. 5 i potrzebie minimalizacji udziału jej części urujonej, należy:

1. za źródło sygnałowe przyjąć APB_CLK: f_{CLK}=80MHz,
2. ustawić podzielnik DIV na wartość 1,
3. ustawić rozdzielczość współczynnika wypełnienia DUTY na wartość N=1.

Pozwoli to uzyskanie impulsów o częstotliwości 40MHz, z niezmienną (ze względu na niską, 1-bitową rozdzielczość) wartością współczynnika wypełnienia równą 50%.



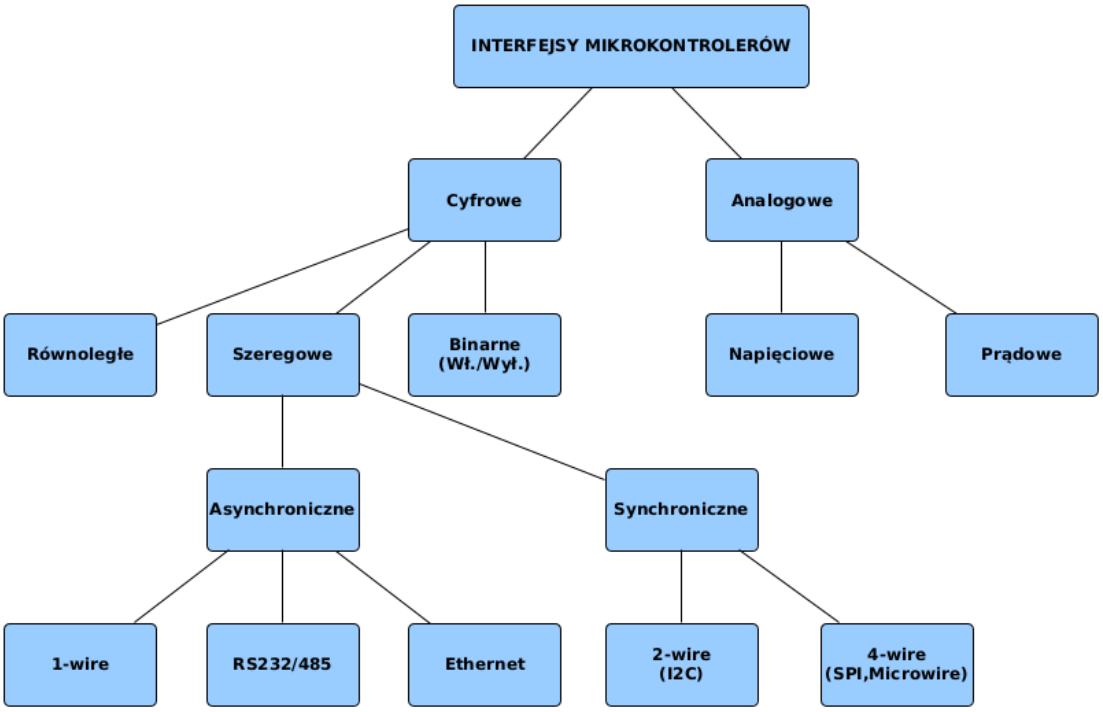
Rysunek 13: Budowa kanału wysokiej prędkości LEDC[52]

Realizacja tego zadania została może zostać wykonana następująco:

```
ledcSetup(ledChannel, frequency, resolution);
ledcAttachPin(PIN, ledChannel);
ledcWrite(ledChannel, width);                    // width ∈ <0;2^resolution-1>
ledcWrite(ledChannel, 0);
```

1.3.3 Sprzętowe interfejsy komunikacyjne w mikrokontrolerach

Na rysunku nr 14 przedstawiono podział sprzętowych interfejsów komunikacyjnych mikrokontrolerów ze względu na charakter przetwarzanego sygnału oraz jego cechy.



Rysunek 14: Podział interfejsów w mikrokontrolerach[52] (opracowanie własne)

1.3.3.1 Protokół komunikacyjny i magistrala danych I2C

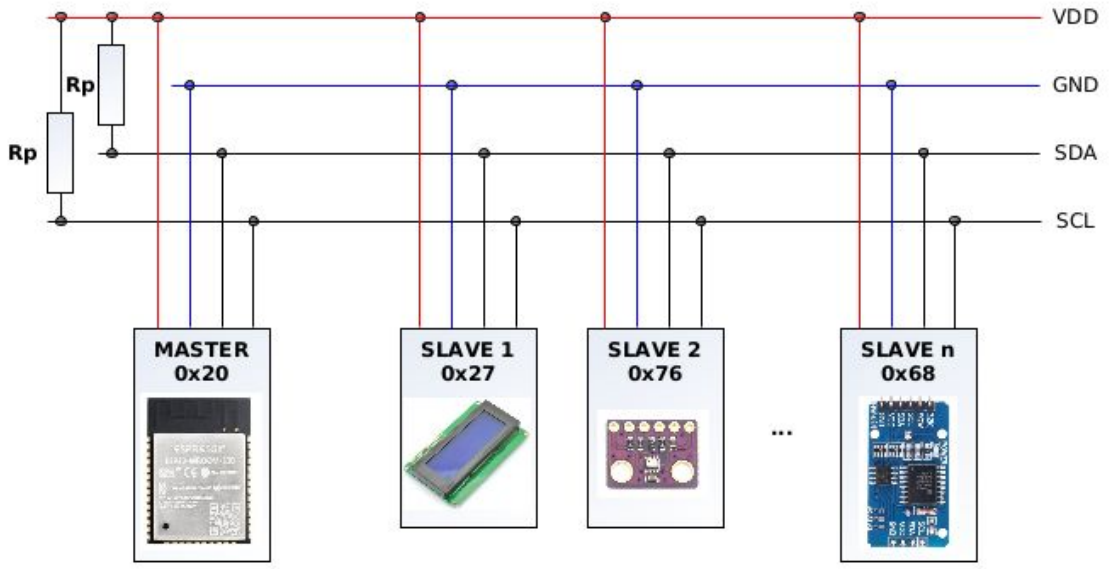
W celu zaspokojenia potrzeb zwiększenia wydajności sprzętu i uproszczenia obwodów elektronicznych, mając na względzie zapewnienie efektywnej komunikacji między komponentami elektronicznymi, firma Philips Semiconductors (obecnie NXP Semiconductors) w 1982r. opracowała dwukierunkową szeregową magistralę danych I2C[53]. Jest ona również znana pod nazwami *Inter-Integrated Circuit*, *Inter IC*, I^2C lub IIC. Urządzenia zgodne z magistralą I2C posiadają wbudowany interfejs, który pozwala im komunikować się przy jego pomocy. Magistrala składa się z dwóch linii:

a) linia *Serial Data* (SDA) do transmisji sygnału z danymi,

b) linia *Serial Clock* (SCL) z sygnałem zegarowym.

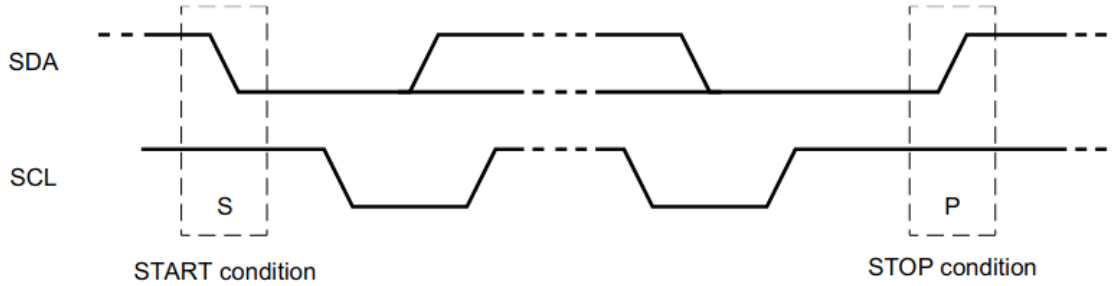
Z racji iż każda z linii jest aktywna w stanie niskim i pracują one w układzie otwartego drenu[55], we właściwej aplikacji wymagane jest zastosowanie rezystorów podciągających R_p (zakres wartości od 1 kΩ do 10 kΩ). Dzięki nim, możliwe jest podłączenie do magistrali więcej niż jednego urządzenia zdolnego do inicjacji transmisji (master) oraz spowolnienie transmisji przez podłączone urządzenia podrzędne (slave), na skutek przytrzymania przez nie linii zegarowej w stanie niskim[54]. Wartość napięcia zasilania V_{DD} nie jest wyspecyfikowana, ale często wynosi ona 3.3V lub 5V. Określone są natomiast graniczne poziomy logicznego „0” (LOW) i logiczne „1” (HIGH) odpowiednio jako $V_{IL}=0.3V_{DD}$ i $V_{IH}=0.7V_{DD}$. Na poniższym rysunku

nr 15 przedstawiono schemat magistrali I²C wraz z podłączonymi urządzeniami zewnętrznymi, zaadresowanymi w schemacie 7-bitowym (teoretycznie to 128 adresów od 0x00 do 0x7F, jednakże część adresów jest zastrzeżonych).

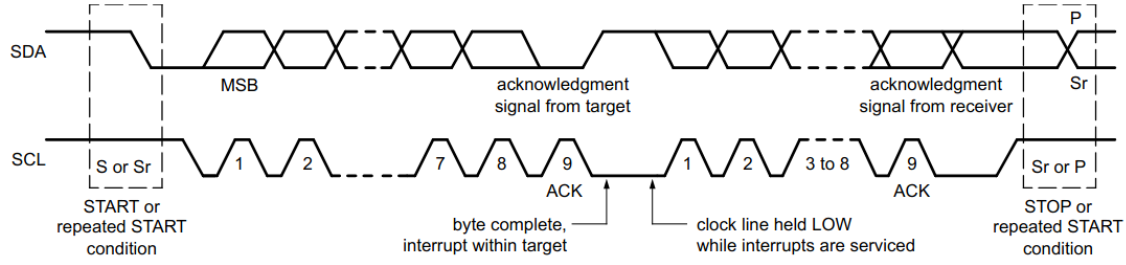


Rysunek 15: Schemat połączeń urządzeń do magistrali I2C (opracowanie własne)

Zgodnie z protokołem I2C każda transmisja danych rozpoczyna się stanem START i kończy stanem STOP, których warunki stanów linii zostały zobrazowane na rysunku nr 16. Dane są transmitowane po jednym bajcie naraz, który zawsze kończy się bitem potwierdzenia ACK. Ta informacja potwierdzająca i zachęcająca do kontynuacji transmisji jest przekazywana poprzez ściągnięcie linii danych SDA do stanu niskiego. Odbiornik może jednakże zasygnalizować potrzebę przerwania/zakończenia transmisji poprzez podciągniecie linii SDA do stanu wysokiego, co jest równoznaczne z brakiem potwierdzenia - NACK. Transmisja danych przez magistralę, z zaznaczeniem potwierdzeń, została przedstawiona na rys. nr 17.



Rysunek 16: I2C – warunki zajścia stanu START i STOP[53]



Rysunek 17: Transmisja danych w magistrali I2C[53]

Cechy charakteryzujące omawianą magistralę to[53]:

a) wymagane tylko dwie linie szeregowe SDA i SCL,

b) urządzenie podłączone do magistrali może być zaadresowane przez unikalny adres o długości 7 lub 10 bitów,

c) urządzenia na magistrali mogą pełnić rolę nadajników lub odbiorników; role te mogą się zmieniać (np. podczas komunikacji dwóch mikrokontrolerów),

d) magistrala jest w stanie obsłużyć wiele urządzeń nadawczych (kontrolerów) inicjujących transfer danych, dzięki mechanizmom wykrywania kolizji,

e) maksymalna przepływność transmisji dwukierunkowej (8-bit) w trybie *Standard* to 100 kbit/s w trybie *Fast* to 400 kbit/s, w trybie *Fast-mode Plus* to nawet 1 Mbit/s; dostępny jest również tryb *High-speed* z graniczną przepływnością na poziomie do 3,4 Mbit/s,

f) maksymalna przepływność transmisji jednokierunkowej to 5 Mbit/s w trybie Ultra Fast,

g) filtry wbudowane w magistralę danych chronią przed skokami napięcia i pomagają w zachowaniu integralności danych,

e) maksymalna ilość podłączonych urządzeń do magistrali jest ograniczona przez wnoszoną w sumie przez nie pojemność elektryczną.

Po 40 latach od stworzenia można stwierdzić, że magistrala I^2C stała się standardem, gdyż jest obecnie zaimplementowana w ponad 1000 różnych układach scalonych, produkowanych przez ponad 50 światowych firm[53]. Ponadto jej uniwersalność pozwoliła ją zastosować w różnych architekturach sterowania jak np. System Management Bus (SMBus), Power Management Bus (PMBus) czy Intelligent Platform Management Interface (IPMI). Ponadto, w 2017r. została zaproponowana nowa, wstecznie kompatybilna magistrala I3C, oferująca większe prędkości transmisji, mniejsze zużycie energii i brak opłat licencyjnych.

Natomiast w układzie ESP32WROOM32E[55] IIC jest obsługiwana za pomocą dedykowanego modułu peryferyjnego, udostępniającego dwie niezależne magistrale, oznaczane jako I2C0 i I2C1. Na każdej z nich ESP32 może zostać podłączony w roli *master* lub *slave*. Obsługiwane są tryby standardowy 100 Kbit/s i szybki 400 Kbit/s oraz krótki i długi schemat adresacji. Użyteczną funkcjonalnością jest możliwość włączenia i konfiguracji cyfrowego filtra do eliminacji zakłóceń na linii danych (SDA). Domyślnie linia ta jest wyprowadzona na GPIO 21, a SCL na GPIO 22, jednakże możliwa jest ich redefinicja w programie. Ostatecznie w celu ustanowienia skutecznej komunikacji na magistrali I2C w układzie ESP32 należy[55]:

1) wybrać tryb pracy *master/slave,*

2) skonfigurować linie do komunikacji(lub pozostawić domyślne) i zdecydować czy użyć rezystorów podciągających (R_p),

3a) w trybie *master*: określić częstotliwość sygnału zegarowego SCL (max. 4MHz),

3b) w trybie *slave*: określić schemat adresacji 7/10bit oraz określić wartość adresu.

W projekcie protokół i magistralę danych I2C wykorzystano: w urządzeniu bezprzewodowym do komunikacji układu ESP32 z czujnikiem wilgotności, temperatury i ciśnienia powierza oraz w sterowniku do podłączenia i obsługi ekranu, czujnika wilgotności, modułu zegara czasu rzeczywistego oraz ekspandera wyprowadzeń cyfrowych.

1.4 Technologie informatyczne

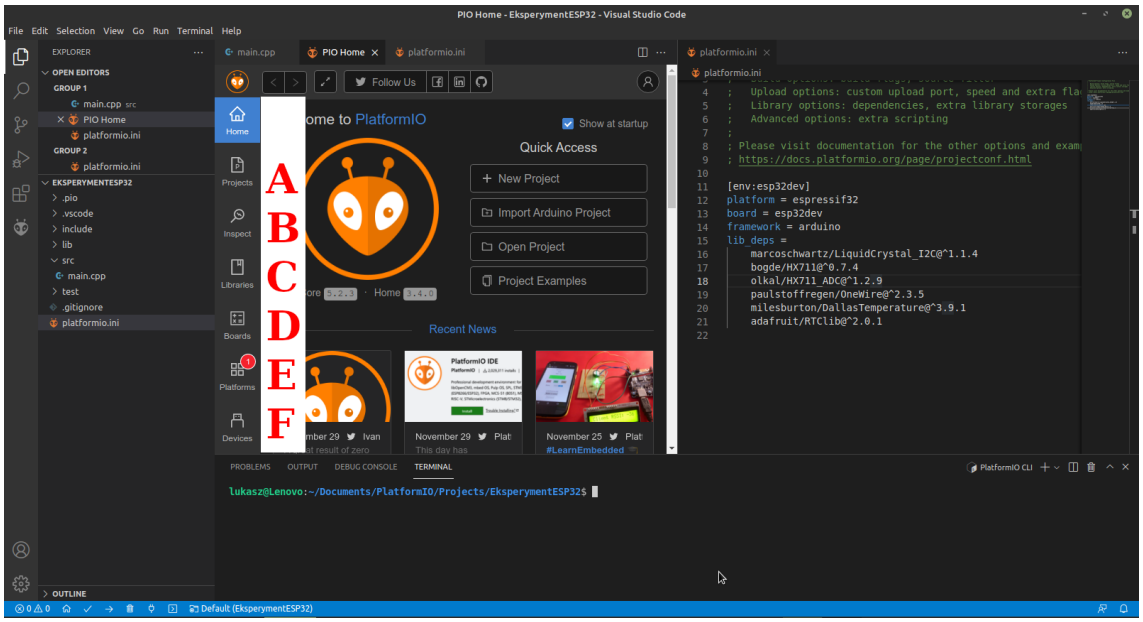
W ramach przygotowania do realizacji projektu potrzebne było znalezienie, poznanie i skonfigurowanie środowiska programistycznego. Nie mniej ważne, ze względu na zaplanowany bezprzewodowy charakter przesyłania informacji, okazały się solidne podstawy teoretyczne dotyczące technologii i protokołów komunikacyjnych. Dlatego też, wymienione zagadnienia zostały zawarte w niniejszym podrozdziale.

1.4.1 Wprowadzenie do zintegrowanego środowiska programistycznego PlatformIO

Visual Studio Code[56] jest lekkim, lecz posiadającym szerokie możliwości edytorem kodu źródłowego dostępnym dla systemów operacyjnych Windows, macOS oraz Linux. Posiada on domyślnie wsparcie m. in. dla języka JavaScript i oferuje rozbudowę wsparcia, poprzez bogaty ekosystem dodatków, do języków takich jak C++, C#, Java, Python czy PHP.

Jedno z takich natywnych rozszerzeń VS Code stanowi narzędzie PlatformIO[57]. Oceniane na poziomie 4.8/5 z ponad 2 mln instalacjami[58] jest zintegrowanym środowiskiem programistycznym do rozwoju oprogramowania dla systemów wbudowanych. Najważniejszymi cechami PlatformIO jest obsługa zestawów narzędzi programistycznych (SDK) lub frameworków, a także oferowanie zaawansowanych funkcji debugowania, przeprowadzania testów jednostkowych, automatycznej analizy kodu i zdalnego zarządzania. Niezwykle przydatną funkcjonalnościami są: inteligentne uzupełnianie kodu w C/C++ oraz menedżer bibliotek programistycznych obsługiwany w ramach środowiska. W ramach podniesienia efektywności pracy zapewniona jest możliwość pracy z wieloma projektami w wielu panelach, a samo środowisko graficzne wspiera ciemne i jasne motywy graficzne. Natomiast obsługa poleceń i komunikacji z programowanymi urządzeniami zapewniona została dzięki wbudowanemu terminalowi z narzędziami linii komend CLI (*Command Line Interface*) oraz monitorowi portów szeregowych.

Okno główne z początkowym widokiem *PlatformIO Home* przedstawione zostało na rysunku nr 18. Przy pomocy (A) dostępna jest możliwość zarządzania projektami. Sprawdzenie wykorzystania pamięci poprzez statyczna analizę kodu jest możliwe w opcji menu głównego *Inspect (B)* – dzięki czemu przeprowadza się analizę projektu. Listy dostępnych, zainstalowanych i wbudowanych bibliotek programistycznych oraz informacje o aktualizacjach zostały zgrupowane pod opcją *Libraries* (C). Natomiast wykaz obsługiwanych platform sprzętowych z wyszczególnieniem ich parametrów został umieszczony pod pozycją *Boards* (D). Oprogramowanie do nich w postaci platform programistycznych lub frameworków możliwe jest do przeglądania i zainstalowania przez *Platforms* (E). Aktualnie podłączone platformy sprzętowe uwidocznione są poprzez *Devices* (F).



Rysunek 18: Początkowy widok zintegrowanego środowiska programistycznego PlatformIO

Wygodną funkcjonalnością, co potwierdziła praktyka korzystania z PlatformIO, jest pasek informacyjno-narzędziowy umieszczony w lewym dolnym rogu okna środowiska deweloperskiego i przedstawiony na rysunku nr 19. Pokazuje on błędy i ostrzeżenia dotyczące aktualnego projektu (1), pozwala przenieść się do panelu początkowego *PlatformIO Home (2),* zbudować aktualny projekt (3) lub/i przesłać go do podłączonej platformy sprzętowej (4). Poza tym istnieje możliwość oczyszczenia projektu, poprzez usunięcie kompilacji i wszystkich jej produktów pośrednich (5). Użycie funkcjonalności kolejnej ikony (6) otwiera monitor portu szeregowego pozwalając na dwustronną komunikację z programowaną platformą sprzętową. Natomiast pod przyciskiem oznaczonym (7) kryje się opcja otwarcia wiersza poleceń używanego systemu operacyjnego wewnątrz środowiska IDE. Ostatnia z prawej funkcjonalność (8) umożliwia zmianę aktywnego środowiska projektowego, do którego odnoszą się wszystkie przytoczone powyżej funkcjonalności.

Rysunek 19: Widok paska narzędziowego PlatformIO

Właściwości każdego projektu definiowane są w tekstowym pliku konfiguracyjnym `platformio.ini` umieszczanym w głównym folderze każdego projektu. Struktura tego pliku składa się z sekcji oznaczanych w nawiasach kwadratowych (dopuszczalne są: [platformio], [env], [env:ENV]), wewnątrz których umieszczane są pary klucz – wartość, definiujące pożądane właściwości. W ramach przykładu przedstawionego poniżej, wyróżniono przez pogrubienie trzy pierwsze obowiązkowe elementy.

Procedura 1: Plik konfiguracyjny PlatformIO

```
; PlatformIO Project Configuration File
[env:esp32doit-devkit-v1]
platform = espressif32      (nazwa wykorzystywanego SoC)
board = esp32doit-devkit-v1 (płytka rozwojowa, z listy wspieranych)
framework = arduino  (środowisko programistyczne do uruchamiania kodu)
monitor_speed = 115200      (prędkość pracy portu szeregowego)
lib_deps =                  (zależności projektu, inst. automatycznie)
              marcoschwartz/LiquidCrystal_I2C@^1.1.4
```

Na liście dostępnych w PlatformIO jest ponad 800 wspieranych układów elektronicznych, 35 architektur sprzętowych oraz 20 frameworków (bez żadnych zewnętrznych zależności). Znajdują się tam także wykorzystanie w fazie rozwoju oraz wybrane w projekcie platformy sprzętowe: Arduino Mini Pro (Atmega328P) oraz ESP-WROOM-02 (ESP32). Właśnie możliwość programowania obu wyspecyfikowanych układów w jednym środowisku stała się głównych powodów wyboru środowiska programistycznego PlatformIO. Jednakże pierwszym pod względem ważności argumentem stanowiła obecność w omawianym IDE gotowych zestawów narzędzi do kompilacji kodu źródłowego (*toolchains*), skrytpów kompilacji (*build scripts*) w odniesieniu do wspieranych platform sprzętowych oraz konfiguracji i obsługi procesu wysyłania końcowego produktu kompilacji do platformy sprzętowej. Nie mniej ważna była niezależna obsługa bibliotek programistycznych w odniesieniu do każdego projektu oraz możliwość łatwego zarządzania nimi dzięki menedżerowi bibliotek oraz modułowi wyszukującemu zależności w bibliotekach (*Library Dependency Finder)*. Istotne okazały się również przytoczone wyżej funkcjonalności środowiska zdecydowanie podnoszące efektywność tworzenia oprogramowania. Poza tym wzięto pod uwagę dojrzałość środowiska deweloperskiego, bogatą i zrozumiale napisaną dokumentację oraz forum społecznościowe PlatformIO Community[57].

Ostatecznie specyfikacja środowiska deweloperskiego w podziale na *hardware* i *software* została przedstawiona w poniższej tabeli nr 5.

Tabela 5: Specyfikacja środowiska deweloperskiego

Sprzęt	Oprogramowanie
Komputer: Laptop Lenovo Z710 Procesor: Intel Core i7-4700MQ 4 rdzeniowy 3400 MHz Pamięć RAM: 16GB DDR3 1600 MHz Dysk: 1TB SSD	System operacyjny: Linux Mint 20.2 Uma (jądro 5.4.0-90-generic) IDE: Visual Studio Code wersja 1.63.2 PlatformIO Core: 5.2.3 Python: 3.8.10

1.4.2 Wybrane technologie i protokoły komunikacyjne oraz ich zabezpieczenie

Niezwykle istotną częścią projektu z niniejszej pracy stanowi łączność bezprzewodowa, dzięki której możliwa ma być dwukierunkowa wymiana informacji pomiędzy komponentami systemu. Sterownik nawiązując komunikację radiową z ruterem (z dostępem do internetu) w obrębie sieci bezprzewodowej WLAN, powinien uzyskać możliwość synchronizacji czasu,

pobierania danych meteorologicznych, publikowania do zewnętrznego serwera stanu stref nawadniania, a także zdalnej konfiguracji. Natomiast zadanie bezprzewodowego czujnika wilgotności gleby stanowi przesyłanie danych pomiarowych do sterownika i przechodzenie w stan uśpienia zgodnie z przesłanymi instrukcjami.

Zaimplementowane funkcjonalności wraz z wykorzystanymi protokołami i metodami zabezpieczenia komunikacji przedstawiono w poniższej tabeli nr 6.

Tabela 6: Protokoły wykorzystane w zaimplementowanych funkcjonalnościach systemu

Lp.	Funkcjonalność	Źródło	Protokół	Zabezpieczenie
1.	Komunikacja pomiędzy czujnikiem wilgotności gleby a sterownikiem	Czujnik wilgotności	autorski nad LoRa	AES 128bit (tryb GCM z AEAD)
2.	Programowane sterownika przy pomocy aplikacji mobilnej	Aplikacja mobilna	HTTP	Tunel L2TP nad IPSec
3.	Synchronizacja czasu sterownika z serwerem czasu	Sterownik	SNTP	-
4.	Przekazywanie informacji o alertach systemu	Sterownik	MQTT	TLS/SSL

Authenticated Encryption with Associated Data (AEAD)[59] jest jednym z trybów symetrycznego szyfru blokowego AES, zapewniając poufność i integralność w ramach jednej operacji. Część danych jest przesyłana w sposób jawny (zachowywana jest ich integralność i uwierzytelnienie), a zaszyfrowanej części danych zapewniana jest poufność i uwierzytelnienie. Podczas uwierzytelnionego szyfrowania wiadomości potrzebne są 4 dane: tajny klucz, wektor inicjalizacyjny, sama wiadomość jako tekst jawny i opcjonalnie dodatkowe dane uwierzytelniające (AAD). Operacja ta zwraca dwie wartości: znacznik uwierzytelniający oraz szyfrogram, którego długość jest taka sama jak tekst jawny. Natomiast odszyfrowywanie poza szyfrogramem wymaga jeszcze klucza, wektora inicjalizacyjnego, dodatkowych danych AAD oraz znacznika. Rezultatem tego procesu wykonywanym po stronie nadawcy jest tekst jawny przesłany przez nadawcę. Aby nie zmniejszać bezpieczeństwa należy dla danego klucza szyfrującego używać za każdym razem innych wartości wektora inicjalizującego.

Dla wybranej w projekcie architektury sprzętowej i frameworka Arduino dostępna jest biblioteka programistyczna *mbed TLS*[60] implementująca protokoły SSL, TLS oraz algorytmy szyfrowania. Wśród nich jest AES w trybie pracy GCM, wspierającym AEAD. Poza tym, ten tryb szyfrowania został wybrany ze względu na długość szyfrogramu identyczną z długością wiadomości, ze względu na ograniczenia długości pakietu LoRa[61] oraz mniejszą ilość energii potrzebną do przesłania krótszych wiadomości.

1.4.2.1 Technologia komunikacyjna LoRa

LoRa (skrót od *Long Range*) jest rodzajem komunikacji bezprzewodowej korzystającej z technologii modulacji fal elektromagnetycznych określanej jako *Chirp Spread Spectrum* (CSS) [62]. Technologia ta była przez dziesięciolecia stosowana w systemach wojskowych

i astronautyce, gdyż umożliwiała uzyskanie dużego zasięgu transmisji i zapewniała odporność na interferencje[63]. LoRa jest zaś pierwszą nisko kosztową pochodną implementacją CSS dostępną do użytku komercyjnego. Jest to technologia własnościowa, opatentowana przez firmę Semtech Corporation[64].

Do zalet Lora zaliczają się[63]:

a) duży zasięg – do 5km w terenach zurbanizowanych i podmiejskich, do 15km na terenach wiejskich oraz na większe odległości na otwartej przestrzeni; modulacja w LoRa jest odporna[63] na propagację wielodrogową i zanikanie sygnału,

b) niskie zapotrzebowanie na energię osiągane przez adaptacyjne dostosowywanie mocy nadajnika oraz wybór szybkości transmisji w zależności od warunków propagacyjnych; oznacza to wydłużenie czasu pracy na baterii lub wręcz rezygnacji z niej na rzecz np. ogniw fotowoltaicznych,

c) zastosowanie do urządzeń w ruchu; przesunięcie częstotliwości na osi czasu sygnału pasma podstawowego wywołane efektem Dopplera, nie pogarsza komunikacji, ze względu na brak ścisłego wymagania co do zegara referencyjnego[63],

d) brak opłat za pasmo – Lora wykorzystuje nielicencjonowane pasma częstotliwości spektrum radiowego ISM (Industrial, Scientific, Medical); w Europie są to częstotliwości 433MHz lub/i 867-869 MHz, w Ameryce Północnej 902-928 MHz, a w Chinach 470-510MHz. W Polsce dozwolone jest wykorzystanie m. in. zakresu częstotliwości 433,05-434,79 MHz[65] dla 20 mW zastępczej mocy promieniowania nadajnika, z aktywnością nadajnika <10%, z zastrzeżeniem, że sygnały zmodulowane z szerokością pasma większą niż 250kHz muszą mieć ograniczoną gęstość mocy do –13 dBm/10 kHz. Dodatkowo przytoczony zakres nie może być używany do transmisji sygnałów akustycznych i wizyjnych.

Zgodnie z modelem ISO/OSI, technologia Lora jest implementacją najniższej warstwy, określanej mianem fizycznej i oznaczanej PHY. Powietrze jest więc medium transmisyjnym fal radiowych podczas dwustronnej komunikacji pomiędzy urządzeniami wyposażonymi w modemy LoRa. Wychodząc z twierdzenia Shannona-Hartleya można wyprowadzić przybliżoną zależność, z której wynika, że zwiększenie szerokości pasma sygnału może być wykorzystane kompensacji degradacji sygnału w kanale radiowym[68], wyrażaną przez stosunek mocy sygnału do szumu/interferencji SNR. Na tej zależności opierają się tradycyjne systemy DSSS, w których pożądany sygnał z informacją jest multiplikowany jest przez tzw. *spreading code*. W implementacji modulacji LoRa, rozprzestrzenianie widma jest osiągane poprzez generowanie sygnału, który w sposób ciągły zmienia częstotliwość[68]. Modulacja sygnału jest zależna od trzech głównych parametrów[66]:

a) BW (*modulation bandwidth*) – określający zakres zmienności częstotliwości modulującej,

b) SF (*spread factor*) – ustalający szybkość zmiany częstotliwość modulującej,

c) CR (*code rate*) – stopień redundancji definiujący nadmiarowość, mającą na celu zapewnienie możliwości korekcji błędów powstałych podczas transmisji.

Modulacja widma rozproszonego LoRa jest więc realizowana poprzez reprezentowanie każdego bitu przesyłanej informacji przez wielokrotne zmianę częstotliwości fali nośnej[67]. N możliwych przebiegów na wyjściu modulatora staje się sygnałami kluczowanymi w przedziale częstotliwości(f0 - BW/2, f0 + BW/2) o N różnych częstotliwościach początkowych.

Większy współczynnik SF powoduje, że całkowita energia przenoszona przez sygnał jest rozłożona na szerszy zakres częstotliwości. Pozwala to odbiornikowi rozróżnić sygnały o niższym SNR, co z kolei pozwala je skutecznie transmitować i odbierać w większej odległości. Jednakże, to rozszerzenie zakresu okupowane jest zmniejszeniem przepustowości transmisji. Niezwykle ważne jest też, że sygnały nadawane w tym samym czasie na określonym kanale (BW), jednak z różnymi parametrem SF nie interferują ze sobą.

Zależność z jaką wymienione parametry wpływają na przepustowość łącza, zdefiniowana jest następująco[68,66]:

$$R_b = CR \cdot \frac{SF \cdot BW}{2^{SF}} \tag{2}$$

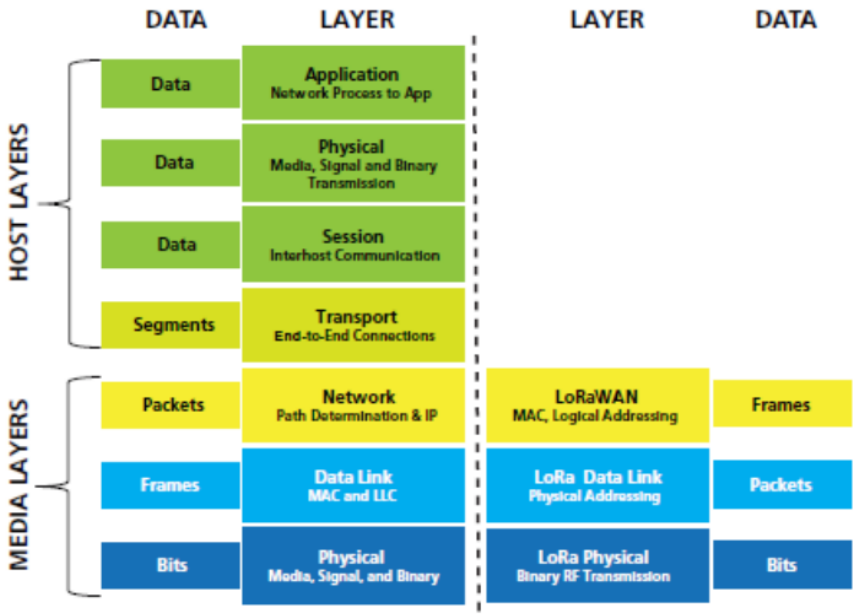
Gdzie dopuszczalne wartości to:
BW = {7.8kHz,10.4kHz,15.6kHz,20.8kHz,31.25kHz,41.7kHz,62.5kHz,125kHz,250kHz,500kHz};
SF = {6, 7, 8, 9, 10, 11, 12},
CR = {4/5, 4/6, 4/7, 4/8}.
Maksymalna przepustowość 37,5kb/s jest osiągana przy SF=6, BW=500kHZ i CR=4/5,
natomiast wartość minimalna jej wartość wynosi ok. 11b/s (dla SF=12, BW=7.8kHZ i CR=4/8).

Na bazie LoRa został stworzony protokół dostępu do łącza (MAC) - LoraWAN[62], zaprojektowany pod kątem wydajności, dalekiego zasięgu i niskiego poboru mocy przez urządzenia Internetu Rzeczy. Wykorzystuje on zalety LoRa, dodatkowo optymalizując zużycie energii oraz obsługując mechanizmy optymalizacji ruchu pomiędzy węzłami. Umiejscowienie technologii LoRa i protokołu LoRaWAN na 7 warstwowym modelu ISO-OSI zostało przedstawione na poniższym rysunku nr 21.

W obrębie rodziny produktów z rodziny LoRa Core firmy Semtech dostępne są między innymi modemy LoRa w postaci mikrochipów, operujących na subgigaheartzowych częstotliwościach. Ich zestawienie, wraz z opisanym powyżej parametrami technicznymi, zostało przedstawione na rysunku nr 20. Przy wykorzystaniu modelu SX1278 oraz biblioteki programistycznej *Arduino LoRa* w projekcie z niniejszej pracy została zaimplementowana komunikacja bezprzewodowa pomiędzy czujnikami wilgotności i sterownikiem nawadniania.

Part Number	Frequency Range	LoRa™ Parameters			
		Spreading Factor	Bandwidth	Effective Bitrate	Sensitivity
SX1272	860 - 1020 MHz	6 - 12	125 - 500 kHz	0.24 - 37.5 kbps	-117 to -137 dBm
SX1273	860 - 1020 MHz	6 - 9	125 - 500 kHz	1.7 - 37.5 kbps	-117 to -130 dBm
SX1276	137 - 1020 MHz	6 - 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm
SX1277	137 - 1020 MHz	6 - 9	7.8 - 500 kHz	0.11 - 37.5 kbps	-111 to -139 dBm
SX1278	137 - 525 MHz	6- 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm
SX1279	137 - 960MHz	6- 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm

Rysunek 20: Kluczowe parametry dostępnych na rynku modemów LoRa[64,66]



Rysunek 21: LoRa i LoRaWAN w modelu odniesienia ISO-OSI[63]

1.4.2.2 Protokół synchronizacji czasu SNTP

Protokół NTP (Network Time Protocol) jest powszechnie stosowany do synchronizacji zegarów komputerowych z rozproszonymi serwerami czasu. W dokumencie RFC5905[69] została opisana jego aktualna wersja NTPv4, wprowadzająca zmiany w nagłówkach dostosowujące protokół do adresacji IPv6 oraz zawierająca szereg poprawek i ulepszeń zwiększających potencjalną dokładność synchronizacji w szybkich sieciach LAN. NTPv4 zachowuje wsteczną kompatybilna z wcześniejszymi wersjami protokołu.

Działanie protokołu opiera się na architekturze klient-serwer, w której na żądanie klienta przesyłana jest odpowiedź zawierająca zbiór danych, pozwalający zsynchronizować zegar systemowy klienta z wybranym serwerem czasu. Niewątpliwą zaletą NTP jest możliwość implementacji szyfrowania pomiędzy klientem i serwerem w celu zapewnienie autentyczności źródła czasu. W zależności od jakości łączy telekomunikacyjnych oraz ich obciążenia, a także

oprogramowania klienta, protokół NTP umożliwia zsynchronizowanie czasu z niepewnością rzędu pojedynczych milisekund, a niekiedy nawet do kilkunastu mikrosekund[71]. Odbywa się to w oparciu o algorytmy statystyczne oraz wykorzystanie wielu źródeł czasu, którego korekta w systemie klienta następuje stopniowo.

Natomiast SNTP jest uproszczoną implementacją pełnej wersji NTP, używaną do synchronizacji czasu przez środowiska lub urządzenia sieciowe, w których wymagania dokładności czasu są mniej rygorystyczne. Do tej grupy urządzeń zaliczyć można[70]: rutery, przełączniki, drukarki oraz im podobne urządzenia z interfejsem sieciowym, jednak o ograniczonych zasobach obliczeniowych i pamięciowych. Głównymi różnicami pomiędzy NTP i SNTP polega na sprawdzaniu błędów i algorytmie rzeczywistej korekty czasu. Natomiast oba protokoły korzystają z tych samych serwerów czasu i pakietów sieciowych. Warte nadmienienia jest, że oba operują na uniwersalnym czasem koordynowanym (UTC), stąd za utrzymanie właściwej strefy czasowej odpowiada oprogramowanie systemu operacyjnego klienta.

Strukturę formatów czasu zaimplementowaną w protokole NTP przedstawiono na rysunkach 23 i 24 z numeracją bitów od 0 do 31. Skrócone, 32 bitowe znaczniki czasu są wykorzystywane do zapisu dyspersji zegarów lub opóźnienia transmisji. Ich skrajne wartości to 2^{-16}s (ok.15μs) i 2^{16}s (ok.18h) - do wymienionych zastosowań mają odpowiednią wielkość i precyzję. Tymczasem podstawowy znacznik czasu o wielkości 64 bitów, pozwala zapisać wielkości od 2^{-32} s (232 pikosekundy) do 2^{32} s (136 lat). Odnosi się on do daty 1 stycznia 1900r. 00:00 i oznacza ilość sekund (do aktualnego czasu UTC) jaka od niej upłynęła.
Poza wymienionymi używany jest jeszcze 128 bitowy format daty, w którego jedną ze składowych jest 32bitowy numer ery. Trwająca obecnie od 01.01.1900r. era 0, zakończy się 7.02.2036 kiedy to rozpocznie się era o numerze 1.

Pakiet NTP jest oparty na datagramie UDP, dzięki czemu ma prostą budowę i uzyskuje niski narzut transmisyjny, co umożliwia uzyskanie niewielkiego opóźnienia odpowiedzi usługi NTP. Jego struktura wraz z opisami została przedstawiona na rysunku nr 22. Port dedykowany samemu protokołowi otrzymał zaś numer 123.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LI			**VN**		**Mode**			**Stratrum**								**Poll**								**Precision**							
Root Delay [t32] Całkowite opóźnienie wynikające z transmisji (wg zegara serwera – odniesienia)																															
Root Dispersion [t32] Maksymalna różnica we wskazaniach zegarów klienta i serwera																															
Reference ID (32 bity) 32 bitowy kod identyfikacyjny serwera lub zegara odniesienia																															
Reference Timestamp [T64] Czas ostatniej korekty lub ustawienia zegara systemowego klienta																															
Origin Timestamp [T64] Czas wysłania żądania do serwera (na zegarze klienta)																															
Receive Timestamp [T64] Czas otrzymania żądania przez serwer(wg zegara serwera)																															
Transmit Timestamp [T64] Czas wysłania odpowiedzi na żądanie przez serwer(wg zegara serwera)																															
Extension Field 1 (zmienna długość)																															
Extension Field 2 (zmienna długość)																															
Key Identifier (32 bity)																															
Dgst [128bitowy MD5 hash]																															

Rysunek 22: Format nagłówka pakietu NTP[69] (opracowanie własne)

Legenda:

t32 - format skróconego znacznika czasu,

T64– format znacznika czasu,

LI – sekunda przestępna,

VN – wersja protokołu,

Mode – tryb pracy,

Stratum – warstwa wzorca czasu,

Pool – maksymalny czas pomiędzy pakietami przesłanymi z sukcesem,

Precison – dokładność zegara klienta.

Oznaczenia t32 i T64 są umowne.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sekundy																ułamki sekund															

Rysunek 23: Skrócony 32 bitowy znacznik czasu w formacie NTP[69] (opracowanie własne)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sekundy																															
ułamki sekund																															

Rysunek 24: 64 bitowy znacznik czasu w formacie NTP[69] (opracowanie własne)

W implementacji sterownika nawadniania wykorzystano protokół SNTP, a za adresy serwerów czasu urzędowego w postaci nazwy domenowej przyjęto tempus1.gum.gov.pl oraz tempus2.gum.gov.pl. Serwery czasu znajdują się w Głównym Urzędzie Miar[71], w Laboratorium Czasu i Częstotliwości i są zsynchronizowane z państwowym wzorcem.

1.4.2.3 Protokół transmisji danych MQTT

Message Queue Telemetry Transport, w skrócie MQTT, jest otwartym protokołem komunikacyjnym urządzeń sieciowych klasy M2M, rozwiniętym i wprowadzonym przez ówczesnego pracownika firmy IBM Andy Stanford-Clark'a oraz Arlen Nipper'a pracującego w Arcom,[72]. Jego przedostatnia wersja 3.1.1 od 2016r. została standardem ISO o symbolu ISO/IEC 20922, natomiast aktualną wersją protokołu jest MQTT 5[73]. Przez ostatnie 20 lat wspomniany protokół był najczęstszym obiektem badań i prac naukowych, na tle konkurencyjnych protokołów AMQP i CoAP, co więcej właśnie według badań MQTT stał się najbardziej rozpowszechnionym w użyciu[74].

W założeniach projektowych MQTT jest oszczędność zarówno przepustowości sieci komputerowej jak i zasobów urządzeń (w tym przede wszystkim zapotrzebowania energetycznego i pamięci operacyjnej), równocześnie gwarantując wiarygodne dostarczanie wiadomości. To wszystko powoduje, że MQTT często jest określany mianem lekkiego protokołu transmisji danych. Dzięki temu, posiada on bardzo szerokie zastosowanie i implementacje obejmujące tak monitorowanie instalacji kolejowych czy rurociągowych, przekazywanie danych pacjentów do placówek ochrony zdrowia, jak i w systemach domów inteligentnych. Jednakże, globalne zastosowanie omawianego protokołu, doskonale uwidacznia się w jego implementacji w komunikatorze internetowym *Facebook Messenger,* a także usłudze oraz usłudze *Amazon AWS IOT*[75]*,* nie pomijając branży transportowej i motoryzacyjnej[76].

MQTT zapewnia komunikację pomiędzy różnorodnymi systemami posiłkując się brokerem – który pełni rolę serwer pośredniczącego. Wykorzystując wzorzec *Publish-Subscribe* odbiorcy dokonują subskrypcji tematów (kanałów informacyjnych potrzebnych lub istotnych dla ich pracy), w których to tematach publikowane są dane pochodzące od nadawców. Wspomniany wzorzec został zobrazowany na rysunku nr 25 w ujęciu domeny nawadniania, ułatwiając jego zrozumienie i za razem ukazując potencjalne zastosowanie. Struktura tematów posiada hierarchiczny charakter, co z założenia umożliwia nasłuchiwanie na określonych poziomach. Aby przedstawić tę użyteczną funkcjonalność, w przykładzie ukazano subskrypcję tematów z wykorzystaniem symboli **+** (dopasowanie do dowolnej nazwy na określonym poziomie) oraz **#** (dopasowanie do wszystkich poziomów wgłąb).

Popularność komunikacji opartej o MQTT wynika między innymi z prostoty jej działania, która umożliwia publikację i subskrypcję informacji, bez wiedzy kto będzie nadawał lub odbierał treść. Powoduje to, iż każda wiadomość może mieć niewielki rozmiar, co przekłada się na wspomniane zredukowanie wymagań przepływności sieci i wykorzystania zasobów na monitorowanych urządzeniach.

Protokół MQTT z założenia przenosi dane typu tekstowego, tak więc poza pojedynczymi tekstami lub liczbami, transmisji mogą podlegać formaty tekstowe np. XML lub JSON. Ta opcja będzie skuteczna pod warunkiem, że nadawca dokona wcześniej serializacji danych, a na odbiorcy będzie ciążyła odpowiedzialność ich deserializacji. Otwiera to więc sposobność transmisji sposób struktur danych w uporządkowany sposób .

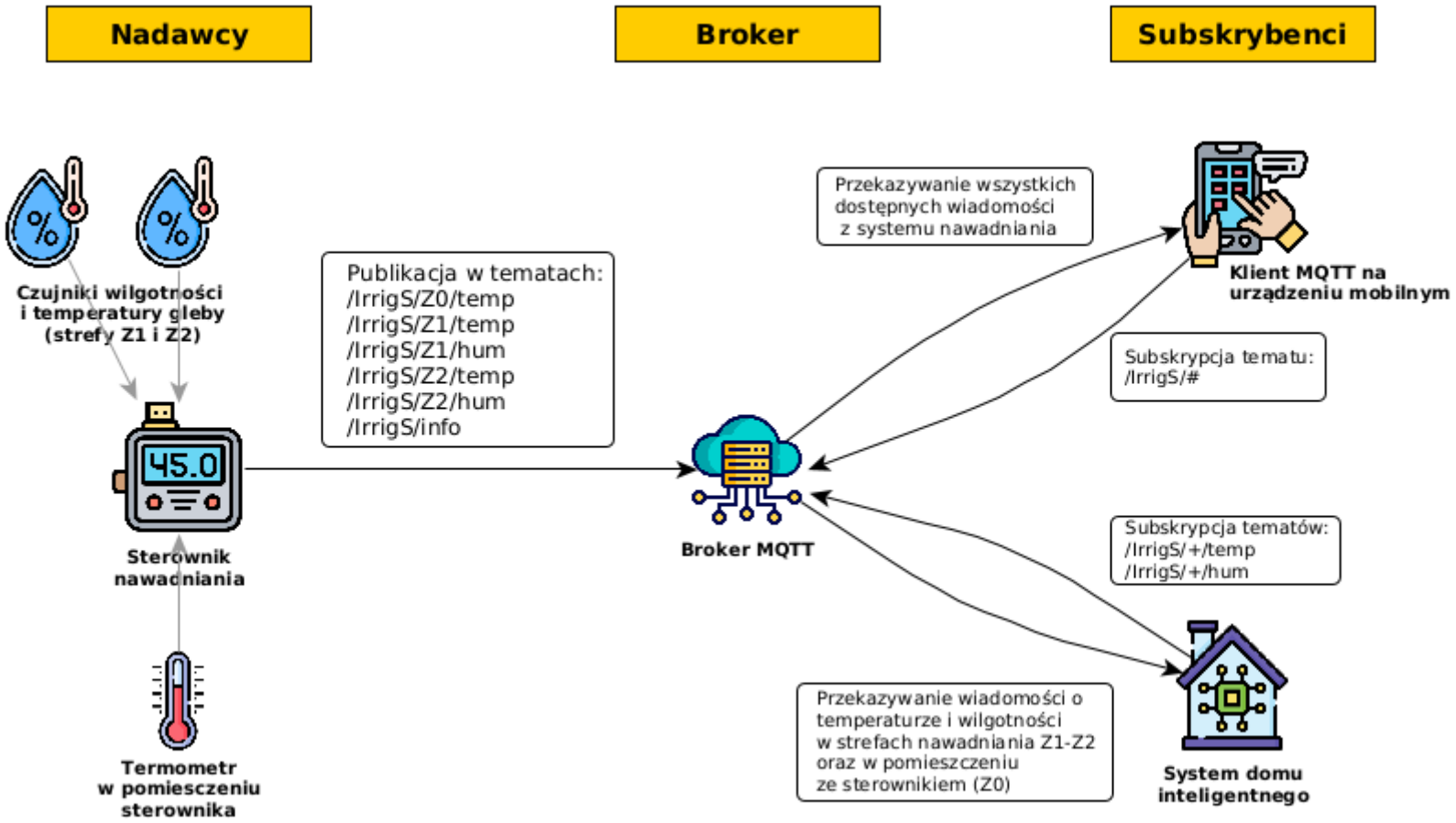
Istotne jest, że każda transmitowana wiadomość musi mieć określony poziom *Quality of Service* (QoS). W protokole zostały przewidziane trzy takie poziomy, umożliwiające kontrolę stanu odebrania wiadomości przez subskrybenta[72]:

a) Level 0 – wiadomości są wysyłane i nie podlegają dalszemu monitorowaniu,
b) Level 1 – wiadomość ma być dostarczona do odbiorcy co najmniej raz (obowiązkowe jest potwierdzenie, dopuszczalne jest, aby było wielu odbiorców),
c) Level 2 – wiadomość musi być dostarczona dokładnie 1 raz do odbiorcy (potwierdzenie odebrania jest wymagane).

Wyższe poziomy QoS zapewniają bardziej niezawodne dostarczanie wiadomości, jednak mogą zużywać więcej zasobów sieci komputerowej lub narażać wiadomość na większe opóźnienia.

Centralną rolę w modelu *Publish-Subscribe* w omawianej technologi odgrywa broker MQTT. Oprócz przytoczonych już zadań, jest on odpowiedzialny za autentykację oraz autoryzację wszystkich klientów, którym ten proces pozwala uzyskać uprawnienia do pełnienia roli nadawców i/lub odbiorców. Oprogramowanie realizujące rolę brokerów dostępne jest na najpopularniejsze systemy operacyjne takie jak: Windows, MacOS oraz Linux zaimplementowane w językach programowania m. in. Java, .NET, NodeJS i Erlang. Protokół MQTT pracuje ponad protokołami TCP/IP. Nasłuch przez brokera jest realizowany na porcie 8883 dla połączeń szyfrowanych protokołami kryptograficznymi TLS i SSL. Natomiast dla połączeń nieszyfrowanych przewidziany jest port 1883.

W najnowszej wersji 5 protokołu MQTT opublikowanej w marcu 2019r. wprowadzony został wariant MQTT-SN (*MQTT for Sensor Networks*) który jest dedykowany dla urządzeń zasilanych bateryjnie.



Rysunek 25: Role dostępne w protokole MQTT oraz wyjaśnienie wzorca Publish-Subscribe wraz z symbolami wieloznacznymi

Dane do sterownika nawadniania są przekazywane innymi technologiami/protokołami. Sterownik nawadniania publikuje dane w temacie „IrrigS" pełniącym tu rolę identyfikatora. Urządzenie mobilne subskrybuje wszystkie dostępne wiadomości, natomiast system domu inteligentnego wyłącznie te dotyczące wilgotności i temperatury (dotyczące dowolnej ze stref Z0-Z2).
Źródło ikon: FlatIcon, Icons created by AB Design [77], opracowanie własne

Ta odmiana protokołu została zaprojektowana do pracy w sieciach bezprzewodowych, dziedzicząc jak najwięcej funkcjonalności pierwotnej wersji. Głównymi różnicami w stosunku do podstawowego wariantu MQTT jest skrócenie wiadomości (m. in. transmitowanie identyfikatorów liczbowych zamiast nazw tematów) oraz rezygnację z utrzymywania stałego połączenie (poprzez wykorzystanie bezstanowego protokołu transportowego UDP)[78]. Mimo, iż opisywana wersja jest stosunkowo nowa, to już ukazują się implementacje zarówno po stronie klienckiej jak również dotyczącej serwerów[79].

1.5 Struktury danych – format wymiany danych JSON

JavaScript Object Notation (w skrócie JSON) jest otwartym tekstowym formatem wymiany danych pomiędzy systemami informatycznymi[80]. Format ten został wywiedziony, jak wskazuje nazwa, z języka JavaScript, jednakże funkcjonuje niezależnie do niego, a jego notacja opisuje uniwersalne struktury danych. Stworzony przez Douglasa Crockforda[81] w 2001 roku, 5 lat później został nazwany i sformalizowany w postaci dokumentu RFC 4627, w którym to została opisana składnia formatu i zarejestrowany został nowy internetowy typ danych *application/json*, a także rozszerzenie plików *.json*. Aktualnie JSON jest opisywany przez dwa niezależne standardy RFC 71592[82] and ECMA-404[83].

Korzystając z zestawu znaków Unicode, w składni formatu JSON zapisać można cztery, wymienione poniżej, podstawowe typy danych:

a) łańcuch tekstowy (z ang. *string*),

b) liczba (z ang. *number*),

c) literał `true` / `false` (z ang. *boolean*),

d) literał `null`.

Dwie możliwe do zapisania struktury danych, zaliczane do złożonych typów danych, to:

a) **tablica** (z ang. *array*) - uporządkowany zbiór wartości, które mogą być każdym z podstawowych (*string*, *number, boolean*, *null*) lub złożonych (*array*, *object*) typów danych. Opis tablicy zaczyna się od lewego nawiasu kwadratowego **[** a kończy się prawym nawiasem kwadratowym **]** , wartości zaś separowane są za pomocą przecinków. Przykładowa tabela reprezentowana w opisywanym formacie wymiany danych wygląda następująco:

["40%", 0, "wilgotność", false, null]

b) **obiekt** (z ang. *object*) w postaci nieuporządkowanego zbioru par nazwa/wartość, gdzie nazwa to łańcuch znakowy, a wartość może być każdym z podstawowych (*string*, *number, boolean*, *null*) lub złożonych (*array*, *object*) typów danych. Pary nazwa/wartość zapisywane w obiektach, otaczane są z lewej strony lewą klamrą **{**, a zakańczane są prawą klamrą **}**. Pomiędzy nazwą i wartością występuje znak separujący w postaci dwukropka **:**, zaś kolejne pary rozdzielane są znakiem przecinka. Przykładowy tekst reprezentujący obiekt JSON, zawierający 3 podstawowe typy danych, został przedstawiony poniżej:

```
{
"wilgotność": "1020hPa",
"temperatura": false,
"ciśnienie": 256
}
```

Pomimo niezłożonej gramatyki, format JSON umożliwia zapisywać złożone struktury danych, przez stosowania zagnieżdżeń wartości w tablicach i obiektach. Przy sprawdzeniu poprawności składniowej, jak również wizualizacji takich struktur danych, niezwykle przydatne są liczne walidatory np. JSON Editor Online[84]. Korzystając z jego funkcjonalności na rysunku nr 26 przedstawiono przykładowy test w formacie JSON wraz z jego obiektową reprezentacją.

W projekcie sterownika wielostrefowego opisywany format danych został wykorzystany wielokrotnie, szczególnie podczas:

a) wymiany danych pomiędzy bezprzewodowym czujnikiem wilgotności i sterownikiem,

b) pobierania i przetwarzania danych z serwisu meteorologicznego,

c) przesyłania instrukcji sterujących z aplikacji mobilnej do sterownika,

d) zapis danych konfiguracyjnych zarówno w sterowniku jak i czujniku wilgotności.

Do obsługi tekstu w formacie JSON, w oparciu o publikację[85], została użyta biblioteka Arduino JSON, dedykowana i zoptymalizowana do użycia w urządzeniach wbudowanych.

Rysunek 26: Tekst zagnieżdżony w JSON oraz jego reprezentacja obiektowa

Niezaprzeczalnymi cechami JSON są jego prosta budowa, stosunkowo niewielki narzut danych (np. w porównaniu do XML), otwartość oraz przenośność. Wszystkie współczesne języki programowania[80] np. Java, JavaScript, Ruby, C#, PHP, Python czy Groovy posiadają gotowe wsparcie w serializacji i deserializacji tego formatu. Nieskomplikowana budowa powoduje, że nie są oczekiwane zmiany w gramatyce JSON, co przyczynia się do jego wyjątkowej stabilności[82] i spowodowało wzrost jego popularności do tego stopnia, że JSON stał się standardem wymiany danych w internecie.

2. ALGORYTM STEROWNIKA NAWADNIANIA OPTISERV

Przed przystąpieniem do planowania projektu oraz zbieraniem do niego wymagań, przeanalizowano ofertę czołowych producentów sterowników nawadniania przeznaczonych do ogrodów. Wyniki zostały zebrane w poniższej tabeli nr 6. Szczególną uwagę zwrócono na opcje dotyczące parametrów obsługiwanych sekcji (ilość, czas pracy, jednoczesność), możliwość sterowania (ręczne (lokalne/zdalne), automatyczne), rodzaje czujników zewnętrznych, obsługiwane elektrozawory oraz sposób programowania przez użytkownika.

Wszystkie wymienione w tabeli sterowniki, jak również 12 innych wybranych modeli wyszczególnionych producentów, posiadają wspólną cechę. Ustalanie programu nawadniania dla wybranej lub wybranych sekcji opiera się o wskazanie trzech parametrów:

1) czasu startu nawadniania,

2) długości nawadniania,

3) okresu nawadniania (co ustaloną ilość dni, dni parzyste/nieparzyste, wybrane dni tygodnia).

W celu ułatwienia definiowania harmonogramów nawadniania przez użytkowników, niektórzy producenci oferują funkcję, dzięki której można kopiować programy z istniejących do aktualnie programowanych sekcji.

Mając na względzie istotność potrzeby uproszczenia obsługi programowania sterowników przez użytkownika, oraz fakt, że w projektowanym systemie są przewidziane czujniki wilgotności gleby autor pracy zaproponował koncepcję obsługi i sterowania sekcjami nawodnieniowymi OptiServ (skrót z ang. *Optimal Services*). Zostanie ona omówiona i zweryfikowana pod kątem użyteczności w tym rozdziale pracy.

2.1 Cel, objaśnienia i opis algorytmu

Celem algorytmu jest sterowanie strefami nawadniania, uwzględniając dane o wilgotności gleby oraz graniczne warunki okresu nawadniania stref zadane przez użytkownika. Cechą wyróżniającą jest brak wymagania ustalenia czasu startu nawadniania - wystarczające są długość i okres nawadniania. Zakres zastosowania obejmuje systemy nawadniania bez czujników wilgotności gleby, jednakże zastosowanie tych czujników w kilku lub wszystkich sekcjach jest obsługiwane i uwzględnione (daje przede wszystkim realne skrócenie czasu nawadniania i oszczędność wody). Nie ma również przeciwwskazań do zastosowania zawieszania nawadniania w ujęciu globalnym systemu lub tylko wybranych stref, gdyż harmonogram nawadniania tworzony jest w sposób niedeterministyczny. Otwiera to możliwość sterowania ręcznego systemem bądź wykorzystanie danych meteorologicznych lub środowiskowych do wstrzymywania pracy, a następnie płynny i bezproblemowy powrót do wykorzystania OptiServ.

W celu dobrego zobrazowania różnic pomiędzy klasyczną metodą programowania sterowników ze z góry ustalonym harmonogramem nawadniania, a wykorzystaniem OptiServ, stworzono ich poglądowe porównanie w tabeli 9, zamieszczonej na końcu tego podrozdziału.

Tabela 6: Porównanie wybranych funkcji ogrodowych sterowników nawadniania dostępnych na rynku europejskim w 2021/22r.

Lp.	Marka i model	Rodzaj	Ilość sekcji	Czas pracy sekcji	Programowanie	Ilość prog.	Zdalne sterowanie	Ręczne uruch.	Liczba sekcji jednocz.	Czujniki Zew.	Elektro-Zawory	Cena*
1	Rain Bird ESP-RZX	zew.	4/6/8 stałe	1-199min	cykliczne, STAND	b. d.	TAK, WiFi	TAK	1	pogoda	230V AC 24V AC	801,53*
2	Hunter PCC 1201E	zew.	6/9/12/15 opcja rozb.	1min-6h	cykliczne, STAND	3	NIE	TAK	1	pogoda	230V AC 24V AC	604,00 zł
3	Galcon AC-6S	wew.	4/6 stałe	1sek-12h	cykliczne2, STAND	3	NIE	TAK	2	pogoda	230V AC 24V AC	500,91 zł
4	Orbit B-Hyve WiFi	zew.	6/12 stałe	1-99min	cykliczne, STAND	3	TAK, WiFi	TAK	1	pogoda, deszcz, mróz	230V AC 24V AC	1 012,54 zł
5	Rain I-Dial	zew.	8 stałe	1min-4h	cykliczne2, STAND	4	NIE	TAK	1	deszcz	230V AC	465,81 zł
6	Cepex CMC 24	zew.	8/16/24 opcja rozb.	1min-9h	cykliczne2, STAND	3	NIE	TAK	1	pogoda	230V AC 24V AC	916,30 zł
7	Rain Bird ESP-9V	zew.	1/2/4/6 stałe	1min-4h	cykliczne, STAND	b. d.	NIE	TAK	1	deszcz	9V DC	387,38 zł
8	Orbit Buddy B-Hyve	zew.	2 stałe	2min-4h	cykliczne, STAND	1	TAK, WiFi Bluetooth	TAK	1	NIE	9V DC	475,27 zł
9	Gardena Classic 6030	zew.	6 stałe	1min-4h	cykliczne, STAND	3	NIE	TAK	1	deszcz, wilg. gleby	24V AC	353,64 zł

Źródło: Internetowy serwis i sklep dotyczący nawadniania [86]

*cena bez modułu WiFi

cykliczne - cykl 7-dniowy, różnorodny 1-6 dni, dni parzyste / nieparzyste, wybrane dni;

cykliczne2 - w określone dni tygodnia, z przerwą 1-31 dni;

opcja rozb. - opcja rozbudowy, ilość stref w zależności od zakupionego modułu rozszerzeń;

STAND - programowanie przez 3 parametry: 1. godzina rozpoczęcia 2. długość 3. częstotliwości nawadniania.

Z kolei dla ustalenia uwagi i jednoznacznego przedstawienia działania algorytmu, poniżej wprowadzono symbole i oznaczenia dotyczące danych wejściowych i parametrów opisujących nawadnianie, a także zdefiniowano łączące je zależności.

Dane wejściowe:

N - ilośc stref (w symulacji N=6)

T_n - okres nawadniania dla strefy *n* [d]

L_n - maksymalna długość pojedynczego nawadniania dla strefy *n* [h]

$N_{paralel}$ - ilość jednocześnie możliwych stref do nawadniania

Inne parametry i oznaczenia:

Z_n - oznaczenie sekcji/strefy o numerze *n*

ZP - punkty strefowe, na podstawie których tworzony jest ranking nawadniania stref; w strefie z największą ilością punktów jako pierwsze rozpocznie się nawadnianie

L_n - maksymalny czas nawadniania sekcji/strefy o numerze *n* [h]

D_{sunset_off} - ilość godzin przed zachodem Słońca, w których nie można rozpoczynać nawadniania

D_{len} – liczba godzin dostępna do nawadniania każdego dnia, długość dnia uwzględniająca D_{sunset_off} (przy większym zakresie dat - średnia liczba godzin) [h]

$\mathbf{T_{NWW}}$ - okres podstawowy, najmniejsza wspólna wielokrotność $T_1, T_2 ... T_N$ [d]

H_{avail} - liczba godzin dostępnych do nawadniania w okresie podstawowym, zmienna w czasie sezonu [h]

H_{plan} - liczba godzin nawadniania, wynikająca z planu nawadniania w okresie podstawowym T_{NWW} [h]

$W_{sh} \in \{1,5,\mathbf{10},50\}$ - współczynnik wagowy, określający istotność składnika wilgotnościowego w stosunku do składnika czasowego podczas wyznaczania punktów strefowych

Zależności:

$$T_{NWW} = NWW\left(T_1, T_2, ... T_N\right) \tag{3}$$

$$H_{avail} = N_{paralel} \cdot T_{NWW} \cdot D_{len} \tag{4}$$

$$H_{plan} = \sum_{i=1}^{N} \frac{T_{NWW}}{T_i} \cdot L_i = T_{NWW} \cdot \sum_{i=1}^{N} \frac{L_i}{T_i} \tag{5}$$

$$ZP = proc_T + W_{sh} \cdot proc_{SH} \tag{6}$$

Wprowadzone powyżej pojęcie okresu podstawowego T_{NWW} dotyczy globalnej konfiguracji sterownika, gdyż zależy od poszczególnych zadanych okresów nawadniania stref $T_1 ... T_N$. Cyklicznie po upływie T_{NWW} dni, wszystkie strefy będą wymagały nawodnienia, jest

to więc dzień o krytycznym znaczeniu w harmonogramie nawadniania. Dodatkowo wprowadzone zmienne H_{plan} i H_{avail} wyrażające ilość godzin odpowiednio dostępnych i planowanych dotyczą właśnie okresu podstawowego, będącą cyklem odniesienia.

Uzyskanie przez strefę dodatnich wartości punktów strefowych ZP (wzór nr 6) jest możliwe po spełnieniu jednego z dwóch warunków:

a) czasowego (składnik $proc_T$) - określonego przez okres T_n (jeśli nawadnianie ma wystąpić danego dnia lub później to uzyskuje tyle punktów, ile % czasu minęło od poprzedniego nawadniania danej strefy do czasu kolejnego nawodnienia). Punkty są przyznawane tylko od dnia wynikającego z okresu T_n, w pozostałych dniach wartość tego składnika wynosi 0,

b) wilgotnościowego (składnik $proc_{SH}$) - określonego przez przekroczenie(powyżej) progu wyzwalania nawadniania (np. 1600). Strefa otrzymuje tyle punktów, o ile procent przekroczono ten próg, przemnożone przez współczynnik $W_{sh.}$ Składnik jest uwzględniany, tylko dla dodatnich wartości $proc_{SH,}$ w przeciwnym przypadku przypisuje się mu wartość równą 0.

Zgodnie z powyższymi ograniczeniami, wartość punktów strefowych może być wyłącznie nieujemna. Natomiast założeniem rankingu stref i punktów strefowych jest możliwość uruchomienia nawadniania tylko w strefach z dodatnią wartością punktów ZP.

Poniżej została przeprowadzona analiza możliwych wartości obu wymienionych składników oraz współczynnika $W_{sh,}$ rzutujących na ilość punktów strefowych zdobywanych przez poszczególne sekcje nawadniania. W ostatniej kolumnie poniższej tabeli nr 7 przedstawiono wyliczone wartości składnika czasowego $proc_T$ na początku dnia, w którym powinno odbyć się nawadniania, w zależności od zadanego przez użytkownika okresu T_n. Założono, że nawadnianie ma odbyć się po 10h od rozpoczęcia dnia.

Tabela 7: Ilość punktów strefowych ZP na początku dnia nawadniania w zależności od okresu nawadniania strefy (10h przed rozpoczęciem)

T_n [d]	T_n [h]	$proc_T$/h [%]	10·$proc_T$/h [%]	ZP_{procT0} [pkt]
1	24	4,2	42	58
2	48	2,1	21	79
3	72	1,4	14	86
4	96	1	10	90
5	120	0,8	8	92
6	144	0,7	7	93
7	168	0,6	6	94

Legenda:
$proc_T$/h - tempo przyrostu składnika $proc_T$ w czasie 1h
ZP_{procT0} - wartość *procT* na początku dnia nawadniania, 10h przed czasem nawadniania

$$ZP_{procT0} = 100\,\% - 10\,h \cdot \frac{proc_T}{h} \qquad (7)$$

Na podstawie powyższej tabeli, algorytm z założeniem W_{sh}=1, promuje najpierw nawadnianie w strefach z największym okresem T_n. Jeśli nie wystarczyłoby dostępnego czasu danego dnia, to na kolejny dzień zostaną przełożone/przełożona strefy lub strefa "najczęściej" nawadniane, uzyskując wtedy wartości powyżej 100pkt, dające pewność uruchomienia w pierwszej kolejności.

W celu przeanalizowania wpływu składnika $proc_{SH}$ na punkty strefowe, w poniższej tabeli nr 8 przedstawiono wpływ wartości parametru W_{sh} na wielkość składnika wilgotnościowego punktów ZP. Ze wzoru nr 6 wynika jego wprost proporcjonalny wpływ na ilość punktów, jednakże z porównania wartości z tabel 7 i 8 wynikają wnioski:

a) niewielkie wartości parametru W_{sh}<=5 będą powodowały, że na wyzwalanie nawadniania będzie miał większy wpływ czynnik czasowy $proc_T$ (osiągający w dniu nawadniania wartości ok. 60pkt i więcej, tabela nr 7)

b) relatywnie duże wartości W_{sh}>=50 promują wpływ składnika wilgotnościowego. Oznacza to w szczególności, że w pierwszy dzień okresu podstawowego T_{NWW}, w którym suma długości nawadniania wszystkich stref jest większa od długości dnia: $\sum_{i=1}^{N} L_n > D_{len}$, wyzwalane będą w pierwszej kolejności strefy osiągające punkt wyzwolenia nawadniania, a część planowych nawadnień (zgodnie z okresem ich strefy T_n) zostanie przesunięta na następny dzień.

Tabela 8: Wpływ wielkości przekroczenia punktu wyzwalania oraz wartości współczynnika W_{sh} na wielkość składnika wilgotnościowego

		W_{sh}=1	W_{sh}=5	W_{sh}=10	W_{sh}=50
$proc_{SH}$	ΔSH	W_{sh}·$proc_T$	W_{sh}·$proc_T$	W_{sh}·$proc_T$	W_{sh}·$proc_T$
1	16	1	5	10	50
2	32	2	10	20	100
5	80	5	25	50	250
10	160	10	50	100	500
20	320	20	100	200	1000
25	400	25	125	250	1250

ΔSH – różnica pomiędzy aktualną wartością określającą wilgotność gleby a punktem wyzwalania nawadniania

Mając na uwadze wartości współczynnika wilgotnościowego, przedstawionego w tabeli nr 8, należy rozważyć możliwość ustalenia przez użytkownika parametru W_{sh} poprzez wybór charakteru głównego składnika przyznawania puntów:

A) czasowego W_{sh}=1,

B) mieszanego (czasowo-wilgotnościowego) W_{sh}=10,

C) wilgotnościowego W_{sh}=50.

Wybór czasowy oznacza, że silnie preferowane będzie przestrzeganie zadanych okresów nawadniania, a tylko w pozostałych wolnych slotach czasowych (pod warunkiem

przekroczenia progu wyzwalania) zostanie uruchomione nawadniane - ze względu na niską wilgotność gleby.

Wybór wilgotnościowy oznacza, że pierwsze w kolejce do nawadniania będą ustawiane strefy, w których został przekroczony próg nawadniania. W pozostałych wolnych slotach w harmonogramie nawadnianie będzie uruchamiane, pod warunkiem osiągnięcia dnia nawadniania, określonego przez T_n.

Natomiast wybór środkowy, czasowo-wilgotnościowy jest wypośrodkowaniem skrajnych preferencji A) i C) i to właśnie on, tzn. W_{sh}=10, został wybrany do implementacji w symulacjach.

Tabela 9: Porównanie algorytmów sterowania nawadnianiem

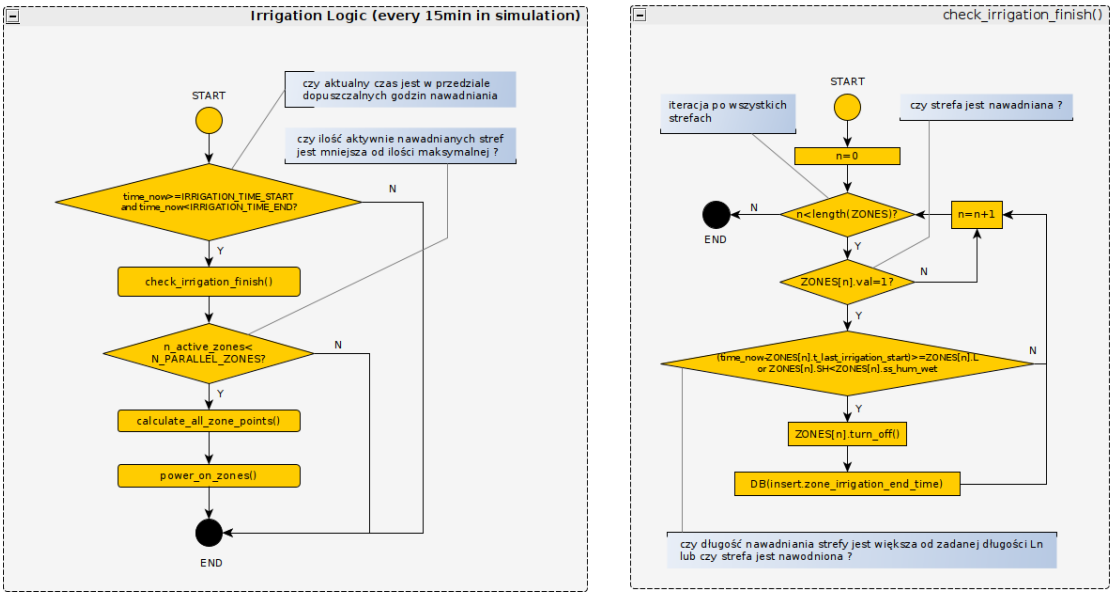
Kryterium porównania	Wariant klasyczny	OptiServ
start programu nawadniania	TAK	**NIE**
długość programu nawadniania	TAK	TAK
okres programu nawadniania	TAK	TAK, jako wartość graniczna (górna granica)
Możliwa zmiana długości nawadniania np. ze względu na informację z czujnika wilgotności	TAK	TAK
Zawieszenie nawadniania	Nawadnianie wypada z harmonogramu (lub jest skracane), kolejne następuje zgodnie z harmonogramem	**Brak harmonogramu** - nic nie wypada z harmonogramu(może nastąpić skrócenie aktualnego nawadniania); po upływie czasu zawieszenia w razie potrzeby ze względu na wilgotność gleby, może nastąpić kolejne nawadnianie nawet tego samego dnia
Ilość jednocześnie działających stref	W przeanalizwanych sterownikach ogrodowych to 1 strefa	**Brak** softwarowego **ograniczenia**. Jedynie ograniczenia są natury fizycznej (maksymalne natężenie prądu transformatora, wydajność źródła wody)

2.2 Schemat blokowy algorytmu

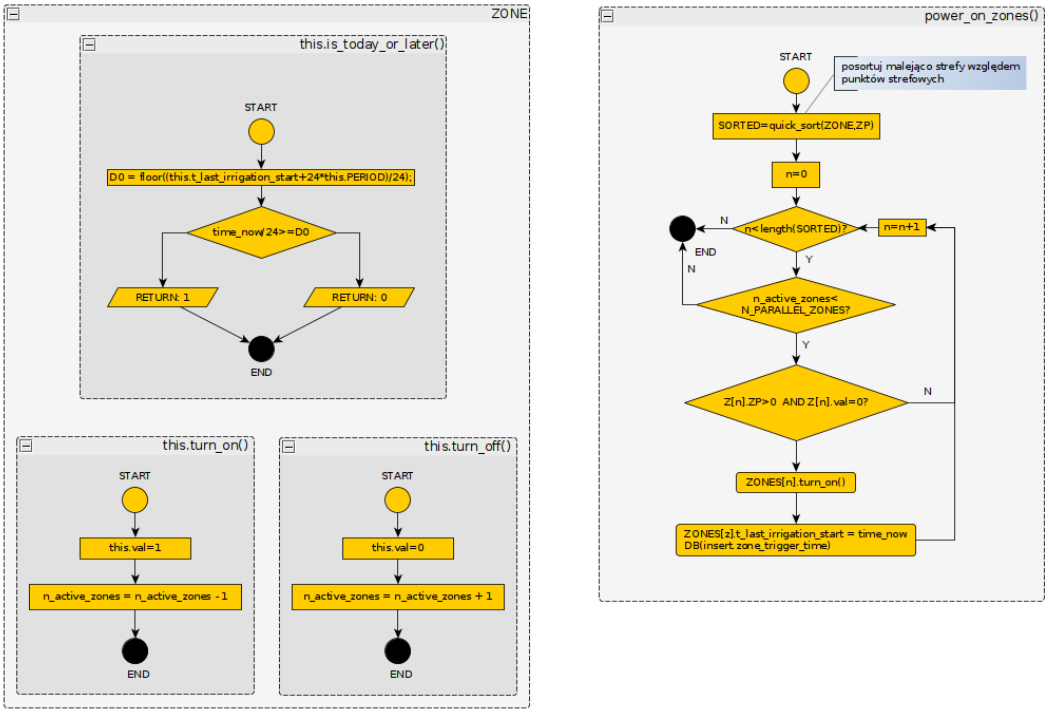
Na poniższych rysunkach 27, 28 i 29 przedstawiono schemat blokowy algorytmu OptiServ. Ze względów na przejrzystość wydzielono logiczne bloki funkcyjne odpowiedzialne za zakańczanie nawadniania stref *check_irrigation_finish,* rozpoczynanie nawadniania – *power_on_zones,* obliczanie punktów strefowych *calculate_all_zone_points* oraz pomocnicze dotyczące włączania/wyłączania poszczególnych stref *turn_on, turn_off* jak również sprawdzającą czy dany dzień jest dniem nawadniania: *is_irrigation_today_or_later.*

Dane dotyczące stanu poszczególnych stref przechowywane są w tablicy *ZONES*. Są to: czas rozpoczęcia ostatniego nawadniania (*t_last_irrigation_start*), stan wł./wył strefy (*val*), próg wyzwalania nawaniania (*ss_hum_dry*), zadany okres nawadniania (*PERIOD*), zadaną długość nawadniania (*L*) oraz ilość punktów strefowych (*ZP*). Wszystkie warunki w algorytmie zostały opatrzona komentarzem w ramkach z błękitnym tłem. Natomiast dane dotyczące

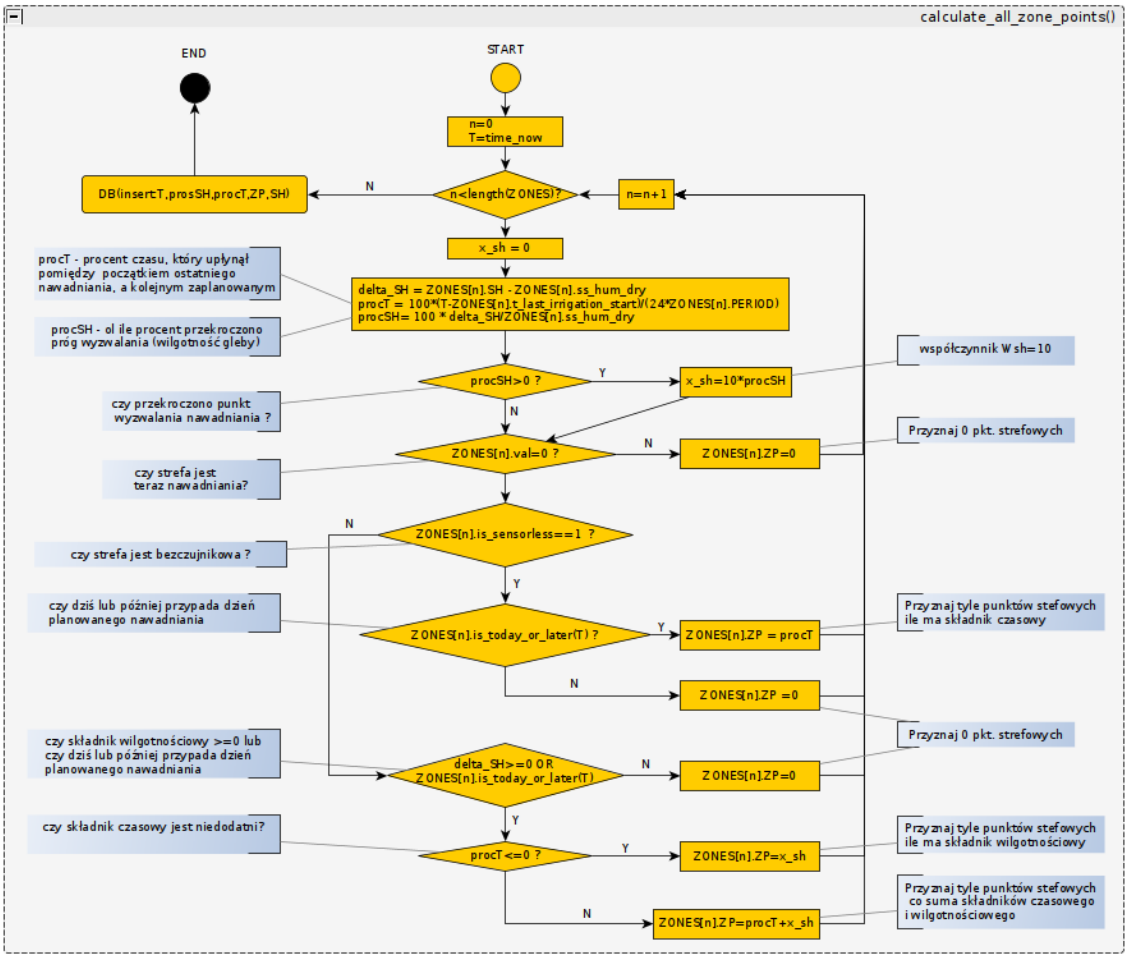
aktualnej sytuacji w strefach: punkty strefowe ZP wraz z ich składnikami $porc_T$ i $proc_{SH}$, wilgotność, czas wystąpienia, czasy rozpoczęcia i zakończenia nawadniania w poszczególnych strefach są zapisywane i przechowywane w zewnętrznej bazie danych. Na schematach operacje związane z ich zapisem zostały oznaczone przez *DB(insert…).*



Rysunek 27: Algorytm OptiServ – główna pętla oraz zakańczanie nawadniania stref



Rysunek 28: Algorytm OptiServ – pomocnicze funkcje strefowe oraz włączanie stref nawadniania



Rysunek 29: Algorytm OptiServ – wyznaczanie punktów strefowych

2.3 Warunki danych wejściowych, przyjęte założenia oraz kryteria oceny rozwiązań

Zdefiniowano 4 warunki dotyczących danych wejściowych: N, T_n, L_n, $N_{paralel}$, których spełnienie powinno zagwarantować całkowite (W1,W3) lub w sensie wartości średniej (W2,W4) uzyskanie przez algorytm zadanych okresów nawadniań wszystkich stref T_n, o ustalonej maksymalnej długości pojedynczego nawadniania L_n.

$$\text{Warunek W1:} \quad \sum_{i=1}^{N} L_i < N_{paralel} \cdot D_{len.min} \tag{8}$$

$$\text{Warunek W2:} \quad \sum_{i=1}^{N} L_i < N_{paralel} \cdot D_{len.śr} \tag{9}$$

$$\text{Warunek W3:} \quad H_{plan} < H_{avail.min} \tag{10}$$

$$\text{Warunek W4:} \quad H_{plan} < H_{avail.śr} \tag{11}$$

Gdzie indeksy dolne *min* i *śr* określają odpowiednio minimalne i średnie długość dnia D_{len} lub godzin $H_{avail.}$

Warunki W1 oraz W2 dotyczą krytycznych dni T_{NWW} będących wielokrotnościami wszystkich naraz okresów nawadniań poszczególnych stref. Nie spełnienie tego warunku oznacza, że w te dni będzie niewystarczająca ilość czasu, żeby nawodnić wszystkie strefy.
Natomiast warunki W3 oraz W4 dotyczą całego okresu podstawowego T_{NWW}, czyli wszystkich dni od 1 do T_{NWW}. Nie spełnienie tego warunki oznacza, że suma dostępnych godzin do nawadniania jest mniejsza niż ilość godzin wynikająca z planu (L_n).

Mając na uwadze dane wejściowe (reprezentowane przez powyższe warunki) oraz możliwe wartości parametrów stref zdefiniowano zdefiniowano 7 grup symulacyjnych charakteryzowanych przez: spełnienie warunków W1-W4, ilość stref równolegle nawadnianych $N_{paralel}$, okres podstawowy T_{NWW}, parametr D_{sunset_off} oraz ilość stref nie wyposażonych w czujniki wilgotności gleby. Każda grupa symulacyjnych ma posłużyć do sprawdzenia zachowania w różnych warunkach i oceny działania algorytmu OptiServ. Podstawowe zestawienie grup zostało przedstawione w poniższej tabeli nr 10. Natomiast szczegółowa charakterystyka stref w poszczególnych grupach symulacyjnych zawierająca ponadto okresy T_{NWW} oraz długości nawadniania L_n, wartości zmiennych w warunkach W1-W4, rezultat oczekiwany, wyniki symulacji oraz rezultat uzyskany, zostały zamieszczone w załączniku F do pracy pt. „podsumowanie symulacji OptiServ”.

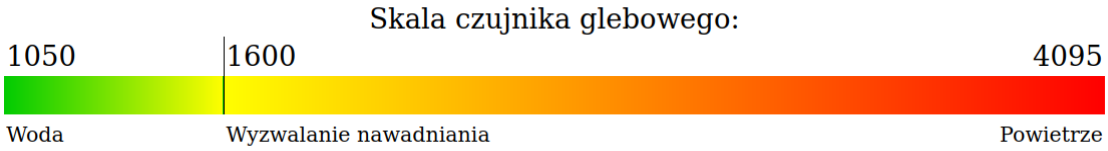
Tabela 10: Zestawienie grup symulacyjnych do testu działania algorytmu OptiServ

Grupa symulacji	Warunki				Ilość stref równoległych	Parametry		Ilość stref bez czujników
	W1	W2	W3	W4		T_{NWW}	D_{sunset_off}	
G1 (01-10)	N	N	T	T	1	6d	2h	0/6
G2 (11-20)	N	N	T	T	1	3d	2h	0/6
G3 (21-30)	T	T	T	T	1	1d	2h	0/6
G4 (31-40)	N	N	N	T	1	2d	2h	0/6
G5 (41-50)	T	T	T	T	1	7d	2h	0/6
G6 (51-60)	N	T	N	N	2	1d	2h	0/6
G7 (61-70)	T	T	T	T	2	6d	2h	3/6

Istotne założenia dotyczące symulacji przedstawiają się następująco:
Z1) Nie została uwzględniona wcześniejsza możliwości zakończenia nawadniania - każde nawadnianie trwa przez zadany okres L_n - co umyślnie nie ma ułatwiać osiągnięcia celu przez algorytm. W warunkach rzeczywistych pewien procent nawodnień będzie krótszy, gdyż cel zostanie osiągnięty przed zadanym czasem z powodu osiągnięcia wymaganej wilgotności gleby, co będzie spowodowane przez czynniki zewnętrzne np. opady deszczu (co zostanie zaraportowane do sterownika przez czujnik wilgotności gleby).
Z2) W symulacji nie uwzględniono opadów atmosferycznych. Z założenia, woda w glebie pochodzi wyłącznie z systemu nawadniania. W warunkach rzeczywistych opady deszczu będą powodowały osiągnięcie celu bez uruchamiania nawodnienia w strefach, przez co niektóre strefy nie będą w tych okresach uzyskiwały punktów strefowych.
Z3) Działanie czujnika wilgotności zostało zamodelowane zgodnie z poniższym rysunkiem nr 30. Za wartość graniczną wyzwalania nawadniania przyjęto wartość 1600, natomiast pełne

nawodnienie po osiągnięciu progu 1200. Skrajne wartości 1050 i 4095 to odczyty sensora uzyskane odpowiednio w wodzie i powietrzu.

Z4) Przyjęto występowanie trzech rodzajów dni: pochmurny (25% szansy), pochmurno-słoneczny (15% szansy) oraz słoneczny (60% szansy wystąpienia). Tempo parowania dla tych dnito odpowiednio: 350, 200 i 500 jednostek pomiarowych czujnika. Wartości te aplikowane są w stosunku do doby krokowo co 15 minut i stanowią odpowiednio ok. 17%, 29% i 42% w odniesieniu do utraty wody dla pełnego nawodnienia. Wartość dla dnia słonecznego (500) powoduje, że dana strefa przy założeniu, że jest na początku w pełni nawodniona(1200), w trakcie pojedynczej doby przekroczy poziom wyzwalania nawadniania ustalony na 1600. [1200+500=1700>1600]. Utrata wilgotności w trakcie nocy została ustalona na 50 jednostek/noc niezależnie od jej długości.



Rysunek 30: Skala czujnika wilgotności gleby

Założenia powyższego modelu powodują, że realnie w pojedynczy dzień w symulacji z prawdopodobieństwem 85% nastąpi osiągnięcie progu wyzwalania w przeciągu poniżej 24h od pełnego nawodnienia, a z szansą 15% w przeciągu 48h. Wartość oczekiwana osiągnięcia punktu wyzwalania nawadniania wynosi zatem 23h. Zakładane są więc restrykcyjne warunki, tym bardziej, że odnoszą się one do całego 180-dniowego okresu symulacji. Podsumowując, szybkość parowania jest celowo przeszacowana w górę (średnio wszystkie strefy przesychają co najwyżej raz na dobę) i aplikowana w symulacji z przytoczonym rozkładem prawdopodobieństwa. Celem takiego działania nie jest odtworzenia warunków rzeczywistych, lecz ustanowienie bardziej wymagających reguł wirtualnego środowiska do przetestowania działania algorytmu.

Do oceny rozwiązań nawadniania uzyskanych za pomocą OptiServ zastosowano cztery wskaźniki zdefiniowane poniżej, odnoszące się do okresu nawadniania L_n:

K1) Różnica pomiędzy okresem zadanym dla strefy, a średnim okresem uzyskanym w symulacji; jednostka: %,

K2) Przekroczenie okresu nawadniania; jednostka:% (sumowane są godziny liczone od następnego dnia, po dniu w którym powinno odbyć się nawadnianie),

K3) Suma przekroczeń okresu nawadniania; jednostka:h (sumowane są godziny liczone od następnego dnia, po dniu w którym powinno odbyć się nawadnianie),

K4) Maksymalne przekroczenie okresu; jednostka: h.

Wszystkie powyższe kryteria odnoszą się do poszczególnych stref, w odniesieniu do wszystkich symulacji przeprowadzanych w obrębie zdefiniowanej grupy.

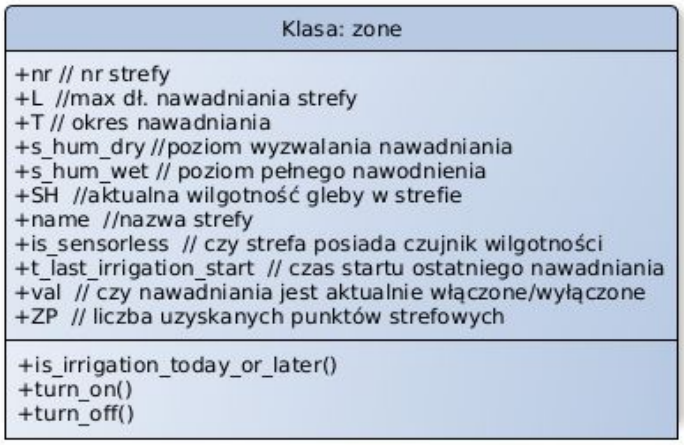
Progi oceny wyników symulacji algorytmu zostały ustalono następująco: (wynik jest przyznawany w pierwszej napotkanym i spełnionym wierszu z warunkami)
K1<=1% i K2<=1% i K3<=12h i K4<=12h - wynik WYSOKI,
K1<=5% i K2<=5% i K3<=24h i K4<=24h - wynik ŚREDNI,
w pozostałych przypadkach - wynik NISKI.
Dla ustalenia układu odniesienia: 12h to 1/360 długości symulacyjnego sezonu nawadniania - tzn. około 0,28%, natomiast 24h to 2/360 okresu nawadniania (ok. 0,56%).

2.4 Stworzenie środowiska symulacyjnego i przeglądarki symulacji

Postępując w myśl współzałożyciela serwisu *Stack Overflow* Jeffa Atwooda, określaną nieraz jako jego prawo: *„Każda aplikacja, którą można napisać w JavaScript, ostatecznie zostanie napisana w JavaScript", autor pracy* stworzył środowisko symulacyjne oraz wizualizacje jego wyników w rzeczonym języku programowania. Doprecyzowując, było to wieloplatformowe środowisko uruchomieniowe Node.js[87] w wersji v10.19.0.

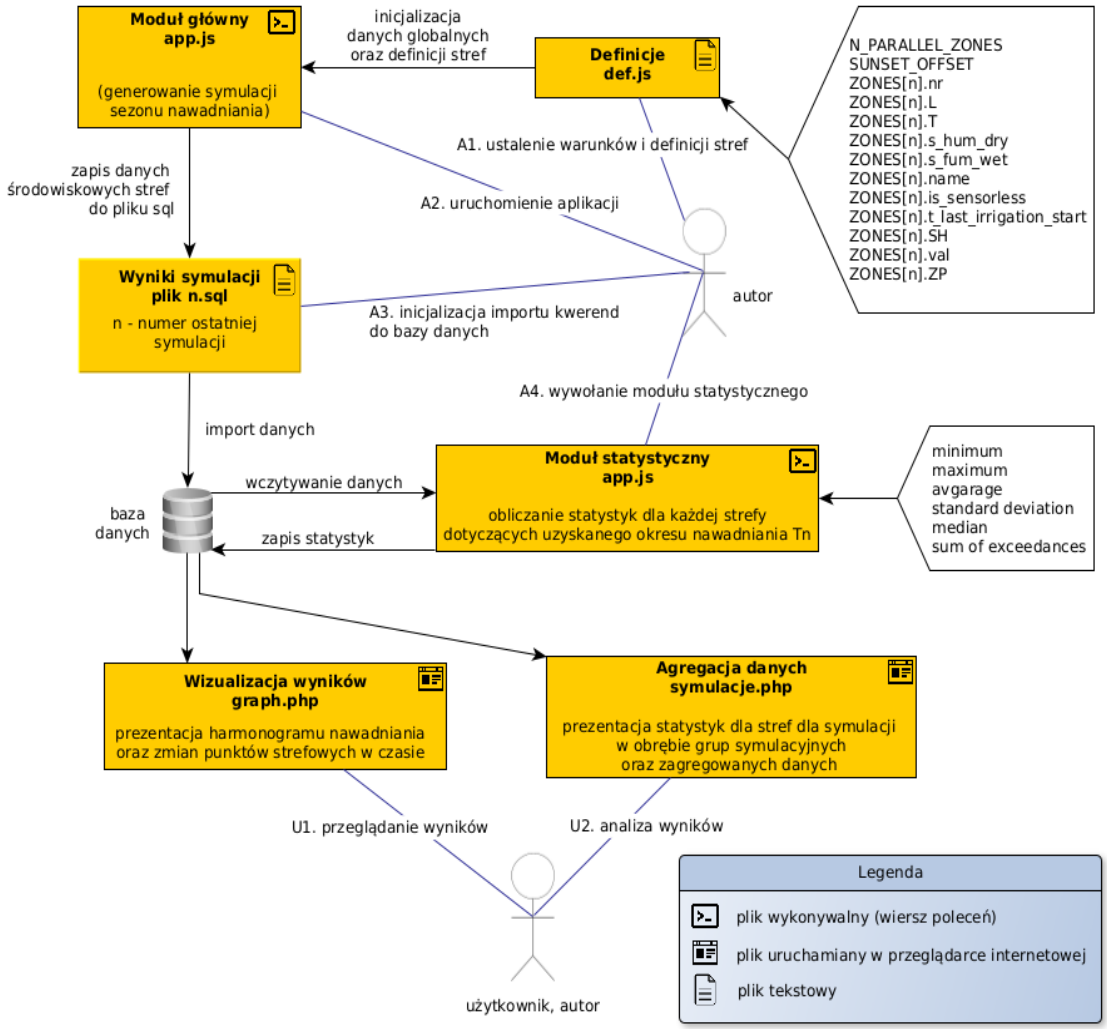
Aplikacja do generowania symulacji działania algorytmu OptiServ podzielona została na trzy moduły: app.js, stat.js (uruchamiane z wiersza poleceń) oraz def.js (plik z eksportem struktur danych). W pierwszym zawarta została implementacja algorytmu, główna pętla sterująca, generująca symulację z krokiem czasowym 15minut oraz funkcje zapisu do bazy danych. Zadaniem drugiego z nich jest obliczanie statystyk z danych wygenerowanych przez pierwszy moduł - również odkładanych w tabelach bazy. Natomiast w pliku def.js znajdują się definicje: stref wraz z ich parametrami, rodzajami dni ze względu na szybkość parowania wody z gleby, zakresem czasowym symulacji, parametrami globalnymi $N_{paralel}$ i D_{sunset_off} oraz danymi konfiguracyjnymi do połączenia z bazą danych MySQL.

Każda ze stref została przedstawiona w postaci obiektu klasy *zone* z właściwościami oraz metodami jak na rysunku nr 31. Wartości *SH*, *ZP*, *t_last_irrigation_start* oraz *val* stanowią opis stanu danej strefy i w każdym kroku symulacji były zapisywane w tabeli bazy danych.



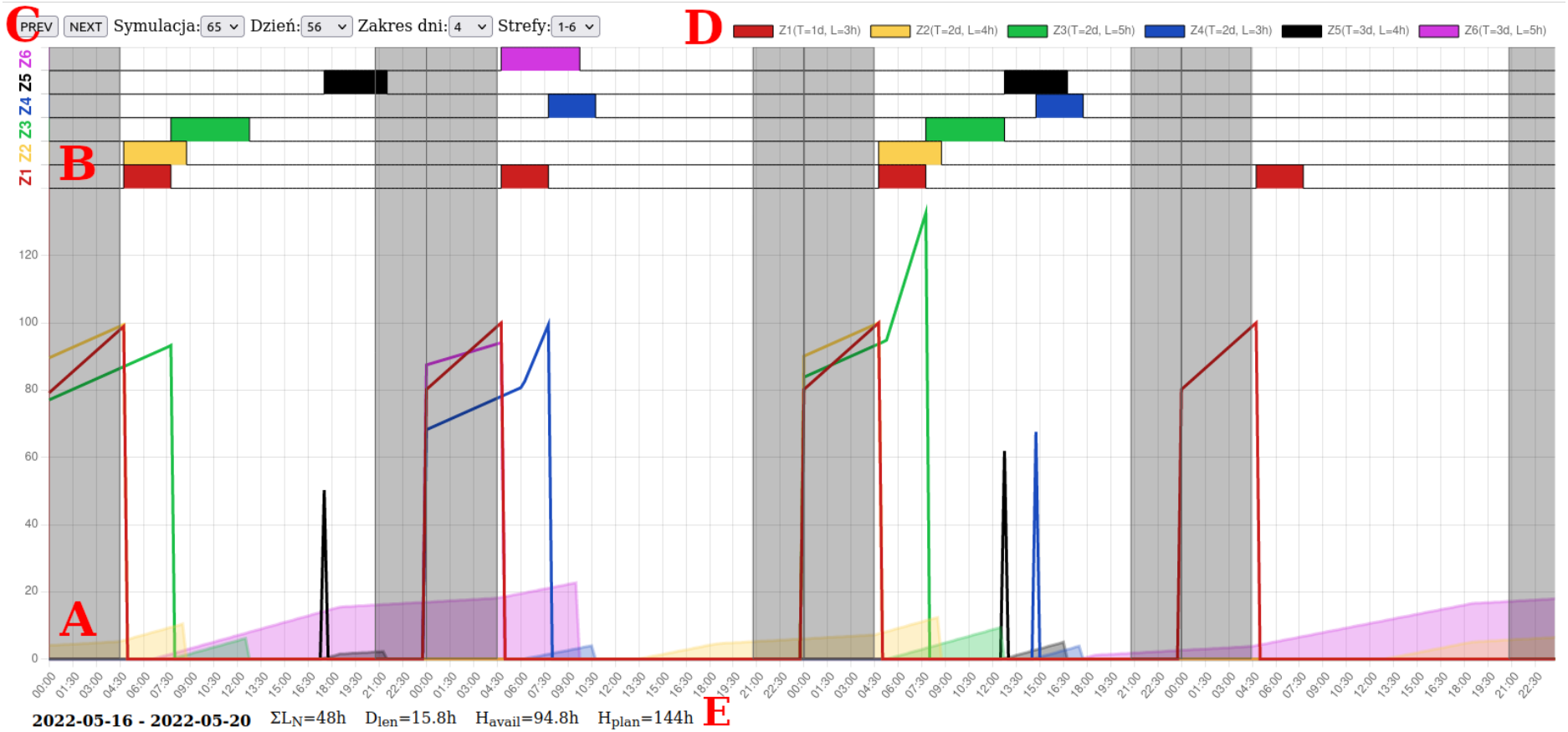
Rysunek 31: Diagram klasy zone w aplikacji do symulacji

Moduły aplikacji do generowania symulacji oraz przeglądarki wyników, wraz z kolejnością uruchamiania i zależnościami zostały przedstawione na poniższym rysunku nr 32. Celem części symulacyjnej jest zapis stanów wirtualnego systemu nawadniania zmieniających się w czasie, do bazy danych oraz wyznaczenie statystyk tych danych. Z kolei przeglądarka wyników obejmuje prezentację harmonogramu nawadniania połączoną z wykresami punktów strefowych oraz podsumowanie wyników uzyskanych w poszczególnych grupach.



Rysunek 32: Składniki aplikacji symulacyjnej wraz z sekwencją ich wywołania

Ostatecznie dla wirtualnego 6 sekcyjnego systemu nawadniania, przeprowadzono 70 symulacji, z których każda obejmuje okres 6 miesięcy od pierwszego dnia wiosny do pierwszego dnia jesieni(180 dni, krok 15min). Długości dni zostały przyjęte dla miasta Płock, PL (długość geograficzna: 19°42′E, szerokość geograficzna: 52°33′N) i mieściły się w przedziale <12h13min;16h51min>. Do wyznaczenia wymienionych wartości wykorzystano bibliotekę *sunrise-sunset-js* korzystającą z implementacji *SunTimes Java* Kevina Boone'a algorytmu Obserwatorium Marynarki Wojennej Stanów Zjednoczonych[88].



Rysunek 33: Zrzut ekranu z aplikacji prezentującej przeprowadzone symulacje

Legenda:

A – obszar wykresu punktów strefowych ZP w zależności od czasu,
B – obszar wykresu prezentujący harmonogram nawadniania,
C – obszar funkcjonalny; wybór parametrów przez użytkownika,
D – legenda wykresu podstawowymi parametrami stref: okresem oraz czasem nawadniania,
E – podsumowanie symulacji: zakres dat, suma godzin w okresie podstawowym, długość dnia, ilość godzin dostępnych i planowanych w okresie podst.

Na rysunku 33 zaprezentowano zrzut ekranu z przeglądarki symulacji (*graph.php*). Na szczególną uwagę zwraca sposób prezentacji danych – połączonych wykresów punktów strefowych złączonych wraz z harmonogramem nawadniania. Zastosowanie takiego sposobu prezentacji ułatwia analizę podjętych decyzji o nawadnianiu w poszczególnych strefach, a w ogólności analizę działania algorytmu. Podstawową zależnością stworzonej przeglądarki stała się biblioteka ChartJS[89], napisana w języku JavaScript, wybrana ze względu na szereg użytecznych parametrów konfiguracyjnych, wyczerpującą dokumentację oraz przykłady użycia.

Cały obszar wykresów poprzecinany jest pionowymi, szarymi pasami oznaczając godziny noce – zakres czasu od zachodu do wschodu Słońca. Dokładność ich wykreślenia jest taka jak krok symulacji i wynosi 15 minut. Podziałka skali poziomej jest zaś zależna od wybranego zakresu dni - wybór od 1 do 24 - i wynosi odpowiednio od 30min do 8h30min. Z kolei pionowa podziałka na skali punktów jest ustalana dynamicznie w zależności od maksymalnej ilości uzyskanych punktów w przeglądanym zakresie dni.

Każdej strefie został przypisany inny kolor, co zostało zdefiniowane w legendzie (*D*) w prawym górnym rogu. Natomiast w części (*A)* wykresu, pełną kreską tworzony jest wykres punktów strefowych, półprzezroczystą zaś składnik wilgotnościowy tych punktów. Obszar pod tym wykresem jest zamalowany z taką samą przezroczystością w celu odróżnienia od punktów strefowych oznaczonych tym samym kolorem. W lewym górnym rogu umieszczono kontrolki, za pomocą których użytkownik może przeglądać pożądany zbiór danych. Przyciski *PREV* oraz *NEXT* przesuwają początkowy dzień widoku o ilość dni określonej w polu wyboru *'zakres dni'* odpowiednio do tyłu i przodu. *'Symulacja'* to kontrolka umożliwiająca określenie numeru z przedziału 1-70, podczas gdy '*dzień'* w zakresie 1-180. Dostępny zakres widocznych dni wynosi od 1 do 24. Z powodu dużej ilości danych w niektórych zakresach dat poszczególnych symulacji, przewidziano możliwość wyświetlania danych ze wszystkich lub tylko jednej wybranej strefy, co zostało zaimplementowane i było dostępne pod postacią ostatniego pola wyboru *'Strefy'.*

Aplikacja okazała się niezwykle pomocna, gdyż jej wykorzystanie dwukrotnie przyczyniło się do wprowadzenia korekt. W pierwszym przypadku była to korekta samego algorytmu, w którym został dodany konieczny warunek dotyczący braku aktualnego nawadniania stref podczas weryfikacji, gdzie należy uruchomić nawadnianie (jest to 3 warunek od góry na rysunku nr 27, *check_irrigation_finish*). W drugim przypadku zaś wykryto nieprawidłową implementację formuły liczącej składnik czasowy punktów strefowych, co powodowało naliczanie zaniżonej ilości punktów i niewłaściwe decyzje o uruchamianiu stref.

Użyteczna okazała się również wizualizacja harmonogramu, dzięki której podczas analizy danych statystycznych okazało się, dlaczego w grupach symulacji G1, G2 i G6 mimo niespełnionych warunków danych wejściowych wyniki wygenerowane przez algorytm uzyskały poziom wysoki (tabela nr 12 poniżej). Właśnie dzięki wizualizacji, wykryto m. in. fakt, że część nawodnień kończyła się po zachodzie Słońca.

Korzystając ze stworzonej przeglądarki dla każdej z grup wybrano jedną symulację dla której ostatecznie zostało przeanalizowanych 30 następujących po sobie dni, pracując na różnych zakresach widocznych dni. Dopiero po pozytywnej weryfikacji produktu algorytmu przy użyciu przeglądarki w porównaniu do danych z modułu statystycznego przystąpiono do oceny rozwiązań, w ujęciu zdefiniowanych wcześniej kryterium.

2.5 Ocena rozwiązań i wnioski

Korzystając z modułu do agregacji danych *symulacje.php* dokonano przeglądu wyników grup symulacyjnych biorąc pod uwagę charakterystykę stref, co bezpośrednio miało wpływ na spełnienie warunków W1-W4. Przykład dla grupy G2 (symulacje 11-20) przedstawiono poniżej (rys. nr 34), natomiast pozostałe zostały zamieszczone w załączniku F.

Nr strefy	L_n [h]	T_n [d]	Wilgotność początkowa	Strefa z czujnikiem?
1	3h	3d	1987	TAK
2	4h	1d	1431	TAK
3	3h	3d	1604	TAK
4	2h	1d	1293	TAK
5	3h	3d	1535	TAK
6	2h	1d	1685	TAK
	ΣL_n=17h	**T_{NWW}=3d**		

Rysunek 34: Charakterystyka stref grupy G2 dla symulacji nr 11-20, z modułu symulacje.php

$$D_{len} \in \langle 12.2\,h; 16.9\,h \rangle \quad D_{lensr} = 15.1\,h \quad H_{avail} \in \langle 36.6\,h; 50.5\,h \rangle$$
$$H_{availsr} = 43.6\,h \quad H_{plan} = 33\,h$$

Warunki dotyczące ostatniego dnia okresu T_{NWW} (co 3 dni):

$$W_1 : \sum_{i=1}^{N} L_i > N_{paralel} \cdot D_{len.min} \left(w\ najkrótszym\ dniu\ brakuje\ 17\,h - 12.2\,h = 4.8\,h \right)$$

$$W_2 : \sum_{i=1}^{N} L_i > N_{paralel} \cdot D_{len.sr} \left(w\ dniu\ o\ średniej\ długości\ 17\,h - 15.1\,h = 1.9\,h \right)$$

Warunki dotyczące okresu T_{NWW}:

$$W_3 : H_{plan} < H_{avail.min}$$

$$W_4 : H_{plan} < H_{avail.sr}$$

Rezultat oczekiwany:

Plan jest ułożony nieprawidłowo (niespełnione warunki W1 i W2). Suma godzin ΣL_n=17h a okres T_{NWW} a okres wynosi 3dni. Z tych powodów spodziewane jest niedotrzymywanie okresów nawadniania dla stref nr 2, 4 i 6, posiadających ten okres najkrótszy (24h).

Tabela 11: Podsumowanie okresów nawadniania grupy G2 dla symulacji 11-20, tabela wygenerowana przez moduł symulacje.php

Strefa	T_N	średnia T_{N_real}	odch. std. T_{N_real}	mediana T_{N_real}	**K1** Różnica procentowa: średni okres/zadany	**K2** Ilość przekroczeń [%]	**K3** Średnia suma godzin przekraczających T_N	**K4** MAX(T_{N_real}-T_N) [h]
1	72h	50.4h	0.0h	0.0h	**-30.0%**	**0.0%**	**0h**	**0h**
2	24h	24.0h	0.3h	2.4h	**0.0%**	**0.1%**	**9h**	**9h**
3	72h	50.6h	0.0h	0.0h	**-29.7%**	**0.0%**	**0h**	**0h**
4	24h	24.1h	1.5h	16.8h	**0.4%**	**0.4%**	**6h**	**6h**
5	72h	50.5h	0.0h	0.0h	**-29.9%**	**0.0%**	**0h**	**0h**
6	24h	24.0h	1.2h	14.4h	**0.0%**	**0.4%**	**10h**	**10h**

Rezultat uzyskany:

Wyniki symulacji dla stref nr 2 i 6 pokazują jednak, że nawet najkrótsze okresy zostały dotrzymane (i średnio wynoszą 24h). W przypadku strefy nr 4 średni okres to 24.1h (średnio o 0,4% przekroczono okres zadany). Wytłumaczeniem uzyskania satysfakcjonującego rezultatu, jest fakt, że co pewien czas nawadnianie odbywało się po zachodzie Słońca dla stref Z2 i Z3.

Podsumowanie wyników wszystkich symulacji w rozumieniu uzyskanego maksymalnego przekroczenia kryteriów K1-K4, przedstawiono w poniższej tabeli.

Tabela 12: Wyniki symulacji w ujęciu zdefiniowanych kryterium

Grupa symulacji	Warunki				MAX przekroczenie kryterium			
	W1	W2	W3	W4	K1	K2	K3	K4
G1 (01-10)	N	N	T	T	0,00%	1,10%	9h	9h
G2 (11-20)	N	N	T	T	0,40%	0,40%	10h	10h
G3 (21-30)	T	T	T	T	0,00%	0,00%	0h	0h
G4 (31-40)	N	N	N	T	41,20%	39,30%	251h	18h
G5 (41-50)	T	T	T	T	0,00%	0,00%	0h	0h
G6 (51-60)	N	T	N	N	0,00%	0,00%	0h	0h
G7 (61-70)	T	T	T	T	0,20%	0,00%	0h	0h

Na podstawie powyższego podsumowania, wyniki działania algorytmu dla grup symulacji: G3,G5,G7, spełniające kryteria danych wejściowych, **uzyskały poziom wysoki**.

Na szczególną uwagę zasługują dobre wyniki dla grup G1, G2 oraz G6 - z niespełnionymi warunkami danych wejściowych. Wytłumaczenie zostało zamieszczone w komentarzu "Rezultat oczekiwany / Rezultat uzyskany" pod podsumowaniem każdej z grup w załączniku w załączniku F do pracy pt. „podsumowanie symulacji OptiServ”.

Podsumowując, na podstawie wyników symulacji, algorytm OptiServ może być skutecznie zastosowany do sterowania sekcjami nawadniania wspomaganymi czujnikami wilgotności jak również bez takiego wspomagania, pod warunkami W1 i W3. Niespełnienie ich nie gwarantuje dotrzymania zadanych okresów nawadniania w poszczególnych strefach.

3. SPECYFIKACJA WYMAGAŃ, MODELOWANIE I PROJEKT WIELOSTREFOWEGO STEROWNIKA NAWADNIANIA ORAZ BEZPRZEWODOWEGO CZUJNIKA WILGOTNOŚCI GLEBY

W rozdziale zostaje przedstawiony harmonogram projektu sterownika wilgotności, zdefiniowanie specyfikacji wymagań, po czym następuje faza modelowania systemu i jego składników. Prowadzi to w konsekwencji do wykonania projektu urządzeń oraz oprogramowania, a ostatecznie do stworzenia scenariuszy testowych.

3.1 Planowanie projektu

Ocenę ryzyka projektu dokonano metodą Thomsetta, której wyniki uwidoczniono w poniższej tabeli nr 13. W wyniku przeglądu opisu uzyskano opis projektu, którego złożoność została na wysoką. Pomimo pewnego doświadczenia autora pracy, jednak ze względu na fakt, że projekt realizowany będzie jednoosobowo, kategoria środowiska zespołu otrzymała maksymalną ilość punktów i co reprezentuje również wysokie ryzyko. Natomiast w kategorii środowiska docelowego ocenę ryzyka oszacowano jako średnie. Ostatecznie ryzyko całego przedsięwzięcia zostało ustalone jako wysokie. Na tej podstawie, zgodnie z zaleceniami[90] wyboru strategii rozwoju projektu, dla projektu powyżej 6 miesięcy z wysokim ryzykiem przyjęto strategię mieszaną. W celu zaadresowania wysokiego ryzyka, budowa urządzeń czujnika wilgotności i sterownika zostanie przeprowadzona poprzez prototypowanie, natomiast ich oprogramowanie zostanie napisane w oparciu o zwinną metodykę Feature Driven Development (FDD).

Tabela 13: Składowe oceny ryzyka według metody Thomsetta

Kategoria	Ocena ryzyka (liczba)	Ocena ryzyka
Klient i środowisko docelowe	200/330	Średnie
Środowisko zespołu	269/269	Wysokie
Złożoność systemu	220/308	Wysokie
Suma:	689/907	**Wysokie**

W celu zarządzania ryzykiem dokonano identyfikacji i analizy czynników sprzyjających powstaniu niechcianych sytuacji. Wyniki tych działań w postaci określenia ryzyk i sposobów ich przeciwdziałania przedstawiono w poniższej tabeli nr 14.

Kolejnym istotnym krokiem wykonanym podczas planowania projektu była identyfikacja i dobór zadań, w wyniku której uzyskano i wyszczególniono: cel główny projektu, jego produkty pośrednie oraz listę zadań. Na podstawie powyższego dokonano szacowania kosztów i nakładów projektu, które posłużyły do stworzenia harmonogramu przedstawionego na rysunku nr 35. Przy użyciu oprogramowania ProjectLibre[91], w oparciu o poradnik ze strony[92] z stworzono diagram Gantta oraz wykres sieciowy metody ścieżki krytycznej projektu, stanowiące załączniki odpowiednio D i E do niniejszej pracy.

Tabela 14:Wykaz zdarzeń niepożądanych i sposób na zmniejszenie ryzyka

Lp.	Zdarzenie niepożądane	Ryzyko	Sposób na zmniejszenie ryzyka
1	Awaria fizyczna komputera	Zagrożenie powstania projektu Niedotrzymanie terminu projektu (wrzesień)	Codzienna kopia zapasowa pracy dyplomowej, plików projektowych i innych powiązanych, zapisywana w prywatnej chmurze Nextcloud (z kopią zapasową na serwerze w odrębnej lokalizacji), Zabezpieczone środki na naprawę lub zakup nowego komputera
2	Nieudana aktualizacja systemu operacyjnego lub komponentu środowiska programistycznego	Opóźnienie w pracach	Codzienny obraz plików systemowych wykonywany za pomocą narzędzia Timeshift
3a	Brak elektronicznych komponentów aktywnych ze względu na globalną pandemię i problem z dostępnością	Niedotrzymanie terminu projektu lub opóźnienie w pracach	Znalezienie alternatywy dla każdego układu aktywnego już na etapie projektowym
3b	Brak układu głównego ESP32 lub komunikacyjnego Lora Ra-02	Zagrożenie	Wykorzystanie droższego, zintegrowanego modułu ESP32 z chipem LoRa, firmy Heltec[93]
4	Brak sukcesu w konstrukcji autorskiego czujnika wilgotności	Niedotrzymanie terminu projektu Niespełnienie założeń projektowych	Przeznaczenie dodatkowego czasu na badania Wykorzystanie gotowych czujników aktywnych i podłączenie ich do czujnika bezprzewodowego
5	Problem z projektem układu zasilającego 24VAC i 5VDC w sterowniku	Niespełnienie założeń projektowych	Wykorzystanie gotowego modułu zasilającego z przetwornicą 24V do 5V
6	Znacząca zmiana w zasadach świadczenia usług przez serwis meteorologiczny lub zaprzestanie świadczenia usług	Brak funkcjonalności związanej z prognozą pogody	Znalezienie dwóch niezależnych dostawców danych meteorologicznych o zbliżonej technologii/strukturze dostarczanych danych
7	Brak możliwości/umiejętności stworzonego algorytmu sterowania	Niedotrzymanie terminu projektu Niespełnienie założeń projektowych	Przeznaczenie dodatkowego czasu na rozwój i implementację W ostateczności uproszczenie założeń/funkcjonalności i próba implementacji takiej wersji
8	Brak możliwości znalezienia gotowej obudowy (sterownik lub czujnik)	Niespełnienie założeń projektowych	Zaprojektowanie dedykowanej obudowy
9	Uszkodzenie prototypu urządzeń w wyniku wypadku/zwarcia	Opóźnienie w pracach	Zamawianie komponentów z uwzględnieniem zapasu
10	Dłuższa, nieplanowana przerwa w pracy nad projektem (choroba, zdarzenie losowe)	Niedotrzymanie terminu projektu (czerwiec)	Możliwość wzięcia do 10dni urlopu do czerwca celem nadgonienia prac. W skrajnym przypadku wykorzystanie możliwości przesunięcia terminu na wrzesień

	⊟Prace badawcze	**48 days**	**13.09.2021,...**	**30.10.2021...**	
	Ustanowienie komunikacji bezprzewodowej przy użyciu LoRa	14 days	13.09.2021, ...	30.09.2021,...	
	Prace badawcze nad autorskim układem wykonawczym do czujnika gleby	30 days	01.10.2021, ...	30.10.2021,...	2
	⊟Projektowanie i rozwój bezprzewodowego czujnika wilgotności	**44 days**	**31.10.2021,...**	**30.12.2021...**	**1**
	Projekt elektryczny czujnika bezprzewodowego	14 days	31.10.2021, ...	18.11.2021,...	
	Projekt elektryczny układu pomiarowego do badań czujnika wilgotności	14 days	01.11.2021, ...	14.11.2021,...	5SS
	Zamówienie i dostawa części elektronicznych (wraz z zapasem)	7 days	19.11.2021, ...	25.11.2021,...	5;6
	Wykonanie prototypu układu wykonawczego czujnika gleby	4 days	26.11.2021, ...	29.11.2021,...	7
	Potwierdzenie założeń projektowych, podstawowe badania na prototypie	5 days	30.11.2021, ...	04.12.2021,...	8
	Zamówienie i dostawa części elektronicznych i mechanicznych(wraz z zapasem)	21 days	05.12.2021, ...	25.12.2021,...	9;11
	Projekt obudowy czujnika bezprzewodowego	10 days	19.11.2021, ...	28.11.2021,...	5
	Budowa bezprzewodowego czujnika gleby	5 days	26.12.2021, ...	30.12.2021,...	10
	⊟Budowa środowiska badawczego, układów elektronicznych i oprogramowania	**34 days**	**31.12.2021,...**	**02.02.2022...**	**4**
	Budowa środowiska pomiarowego (sprzęt+oprogramowanie)	10 days	31.12.2021, ...	09.01.2022,...	
	Badanie bezprzewodowego czujnika gleby	24 days	10.01.2022, ...	02.02.2022,...	14
	⊟Projektowanie i rozwój sterownika wielostrefowego	**74 days**	**03.02.2022,...**	**17.04.2022...**	**13**
	Projekt elektryczny sterownika wielostrefowego	7 days	03.02.2022, ...	09.02.2022,...	
	Projekt obudowy sterownika wielostrefowego	5 days	03.02.2022, ...	07.02.2022,...	17SS
	Budowa sterownika	7 days	10.02.2022, ...	16.02.2022,...	17;18
	Rozwój oprogramowania dla sterownika i czujnika bezprzewodowego	60 days	17.02.2022, ...	17.04.2022,...	19
	Zakup i dostawa rutera, elektrozaworów i pozostałych pomniejszych	7 days	02.05.2022, ...	08.05.2022,...	
	Konfiguracja rutera	2 days	09.05.2022, ...	10.05.2022,...	21
	⊟Projekt aplikacji na urządzenia mobilne	**29 days**	**18.04.2022,...**	**16.05.2022...**	**20**
	Projekt aplikacji	5 days	18.04.2022, ...	22.04.2022,...	
	Implementacja aplikacji mobilnej	21 days	23.04.2022, ...	13.05.2022,...	24
	Integracja ze sterownikiem	3 days	14.05.2022, ...	16.05.2022,...	20;25
	Realizacja scenariuszy testowych	14 days	02.06.2022, ...	15.06.2022,...	16;20;22;23;26
	Sporządzenie dokumentacji	180,75 d...	01.01.2022, ...	30.06.2022,...	
	Zakończenie projektu	0 days	30.06.2022, ...	30.06.2022,...	27;28

Rysunek 35: Harmonogram projektu wielostrefowego sterownika nawadniania

3.2 Zbieranie, analiza i specyfikacja wymagań projektu

Na podstawie przeglądu sterowników nawadniania przedstawionego w podrozdziale 1.1.1, zestawienia elektrozaworów z ich parametrami z tabeli nr 1, a także porównania wybranych funkcji ogrodowych sterowników nawadniania dostępnych na rynku europejskim (synteza została przedstawiona w tabeli nr 6), wstępnie określono listę potencjalnych możliwości i cech wielostrefowego sterownika nawadniania. Funkcjonalności i ich ograniczenia zostały przeanalizowane pod kątem użyteczności, możliwości realizacji jednocześnie z implementacją algorytmu Optiserv oraz w ogóle wykonalności w ramach projektu dyplomowego.

Na tej podstawie opracowano specyfikację wymagań projektu i jego składników. Poniżej wymienione zostały wymagania dotyczące systemu nawadniania (A), sterownika (B), czujnika wilgotności gleby (C) oraz aplikacji mobilnej (D).

Oznaczenie [ext] to funkcje zaplanowane do realizacji w kolejnych wydaniach oprogramowania układowego, pozostałe to wymagania podstawowe do wykonania w pierwszej, realizowanej w tej pracy. Specyfikacja sprzętowa powinna uwzględniać wymagania z obu wersji software, tak ażeby możliwe było ich uzyskanie na tej samej platformie sprzętowej. Dodatkowo wymagania zostały podzielne na kategorię sprzętową [H] oraz oprogramowania (nie oznaczaną w specyfikacji)

A. System nawadniania

A.1. Komunikacja: [H]

a) odległość skutecznej komunikacji bezprzewodowej pomiędzy sterownikiem a czujnikami wilgotności gleby: minimalnie 1500m,

b) dostęp sterownika do internetu przez istniejącą lokalną sieć bezprzewodową WiFi.

A.2. Funkcje: [H],

a) obsługa do 8 niezależnych stref nawadniania, [H]

b) przygotowanie do rozszerzenia do 16 stref za pomocą zew. pasywnego modułu [H],

c) do każdej strefy należy umożliwić przypisanie do 4 czujników wilgotności gleby [H]

d) w strefach z wieloma czujnikami należy umożliwić wybór funkcji statycznej (średnia, mediana, maksimum, minimum) do obliczenia strefowej wartości temperatury i wilgotności gleby

A.3. Bezpieczeństwo:

a) szyfrowana komunikacja (przy wykorzystaniu protokołu Lora) pomiędzy sterownikiem i czujnikami z kluczem o minimalnej długości 128 bitów, [H]

b) wykrywanie ruchu w pomieszczeniu ze sterownikiem, możliwość zazbrojenia alarmu i w razie naruszenia strefy przesłanie informacji do brokera MQTT, [H]

c) w układach sterownika, czujnika środowiskowego oraz w instalacji do sterowania elektrozaworami ma być napięcie bezpieczne dla życia ludzkiego, [H]

d) nie dopuszcza się pozostawiania otwartych portów na ruterze z przekierowaniem do sterownika, np. do strony logowania lub przekierowanie poleceń sterujących,

e) standard szyfrowania sieci bezprzewodowej zaleca się na co najmniej WPA2, [H]

f) opcja podłączenia elektrozaworu głównego umożliwiającego odcięcie wody od systemu nawadniania [H]

g) możliwość lokalnego sterowania wyłącznie po odblokowaniu kodem PIN sterownika, wpisywanym w aplikacji mobilnej. [ext]

B. Sterownik

B.1. Budowa [H]

a) urządzenie przeznaczone do użytku wewnątrz pomieszczeń, ochrona IP min. 40,

b) należy przewidzieć miejsce na opis stref nawadniania obok zacisków urządzenia z przewodami do elektrozaworów (np. numer strefy, nazwa),

c) należy wyprowadzić odpowiednie linie sygnałowe na krawędź obudowy w celu umożliwienia rozszerzenia ilości stref do 16,

d) zasilanie napięciem 24AC, wykorzystywanym równocześnie do zasilanie elektrozaworów,

e) podświetlany ekran,

f) możliwość zawieszenie na ścianie,

g) listwa barierowa do podłączenia zasilania, elektrozaworów i przewodowych czujników zewnętrznych ma być w umieszczona w dolnej części obudowy

h) obsługa sterownika przez pokrętło z przyciskiem oraz przyciski funkcyjne,

i) antena LoRa wyprowadzona na górną część obudowy

j) należy wyprowadzić na zewnątrz obudowy co najmniej dwie linie GPIO w celu wykorzystania w przyszłości np. do podłączenia zewnętrznych przewodowych czujników

B.2. Tryby pracy:

a) tryb podstawowy – realizacja nawadniania zgodnie z algorytmem Optiserv w trybie automatycznym lub poprzez nastawy manualne zadawane lokalnie lub przez aplikację mobilną,

b) tryb konfiguracji – inicjalizacja urządzenia, umożliwienie wprowadzenia parametrów konfiguracyjnych do połączenia z siecią bezprzewodową, przywrócenie hasła domyślnego, wybór opcji sprzętowej – obsługa 8 lub 16 stref.

B.3. Zawartość ekranu w trybie podstawowym:

a) aktualny status stref nawadniania (włączona, wyłączona, nieskonfigurowana),

b) data i godzina (które powinny być utrzymywane w przypadku braku zasilania),

c) temperatura w pomieszczeniu ze sterownikiem,

d) informacja o połączeniu z siecią bezprzewodowej WiFi,

e) status systemu nawadniania,

f) status karty pamięci,

g) informacja o aktualnym trybie sterowania nawadnianiem: manualny lub automatyczny,

h) informacja o zazbrojeniu/rozbrojeniu alarmu.

B.4. Podświetlenie ekranu [H]

Ekran ma być podświetlony po wykryciu ruchu przed urządzeniem. Podświetlenie ma być wyłączane po około 60 sekundach nieaktywności użytkownika.

B.5. Monitoring środowiska w pomieszczeniu [H]

Należy zapewnić możliwość uzyskania informacji o temperaturze, wilgotności w pomieszczeniu ze sterownikiem.

B.6. Ręczne sterowanie [H]

Przy użyciu panelu urządzenia w stosunku do każdej ze stref ma być możliwość:

a) włączania/wyłączania nawadniania ,

b) aktywacji/dezaktywację (włączenie/wyłączenie z użytkowania).

B.7. Zadawanie parametrów stref nawadniania oraz parametrów sterownika

a) przy użyciu aplikacji mobilnej,

b) przy pomocy panelu sterownika. [ext]

B.8. Sterowanie elektrozaworami U=24V AC, f=50Hz uwzględniając:

a) lokalne nastawy,

b) nastawy zlecone zdalnie przez aplikację mobilną,

c) dane z bezprzewodowych strefowych czujników wilgotności i temperatury gleby,

d) prognozy pogody (wykorzystanie API wybranego serwisu pogodowego).

B.9. Data i godzina:

a) sterownik ma mieć bateryjne podtrzymanie czasu, [H]

b) sterownik ma co najmniej raz na tydzień synchronizować czas z wiarygodnym źródłem z internetu (możliwość podania adresu źródła czasu).

B.10. Monitorowanie przepływu cieczy:

a) określanie czy po uruchomieniu programu w głównej magistrali wody jest przepływ. Jeśli minie określony czas np. 3s to należy wstrzymać działanie programu i wygenerować powiadomienie, [H]

b) oszacowanie zużytej wody podczas doby. [ext]

B.11. Wykorzystanie czujnika ruchu i wprowadzenie stanów alarmu uzbrojony/rozbrojony. [H]

B.12. Dane historyczne – logowanie zdarzeń:

Zapis wyników pomiarów wilgotności, godziny włączenia i wyłączenia nawadniania w danej strefie i innych zdarzeń na karcie pamięci sterownika. Karta zgodna ze standardem SD, o wymiarach microSD. [H]

B.13. Powiadomienia

Sterownik ma mieć możliwość generowania powiadomień za pomocą komunikatów do zewnętrznego brokera MQTT w przypadku:

a) braku przepływu cieczy po uruchomieniu programu,

b) przepływ wody pomimo wyłączonych stref nawadniania,

c) naruszenia stref monitorowanych przez czujniki ruchu,

d) aktualny stan wilgotności gleby w strefach (z dokładnością do częstotliwości raportowania przez czujniki),

d) raport z dziennym podsumowaniem uruchomień nawadniania[ext]

Wybrane komunikaty np. (a, b, d) mogą być wysyłane za pomocą wiadomości e-mail [ext]

C. Czujnik strefowy

C.1. Obudowa [H]

a) Urządzenie przeznaczone do użytku na zewnątrz pomieszczeń, ochrona IP min. 54

C.2. Zasilanie [H]

a) zasilanie z akumulatora,

b) ładowanie akumulatora z zewnętrznego panelu słonecznego,

c) ładowanie akumulatora przez port mini USB (po zdjęciu obudowy).

C.3. Interfejs użytkownika [H]

a) dwukolorowa dioda LED sygnalizująca status urządzenia i zmierzoną wilgotność gleby (przy wykorzystaniu kolorów: czerwonego, zielonego, żółtego)

b) przycisk do wybudzenia urządzenia i wykonania pomiarów na żądanie

C.4. Funkcje

a) pomiar wilgotności gleby na głębokości 2-50cm, [H]

b) pomiar wilgotności gleby 1,2 lub 3 zewnętrznymi elementami pomiarowymi (połączenie przewodowe), [H]

c) pomiar temperatury, wilgotności i ciśnienia powietrza nad glebą, [H]

d) przypisywanie czujnika do wybranej strefy/ odłączanie od strefy,

e) pomiar temperatury gleby, [H]

f) pomiar przybliżonego stopnia naładowania baterii, [H]

D. Aplikacja mobilna

D.1. System operacyjny: [H]

Android w wersji 5.0+

D.2. Funkcjonalności

Aplikacja powinna umożliwiać :

a) zapamiętanie adresu IP i hasła do sterownika oraz nazwy ogrodu

b) włączanie/wyłączanie nawadniania każdej ze stref,

c) aktywację/dezaktywację danej strefy (włączenie/wyłączenie z użytkowania),

d) nadawanie nazw strefom i konfigurację jej parametrów (nazwa strefy, minimalna długość nawadniania w trybie automatycznym, ustalenie czy strefa jest a. odblokowana b. bezczujnikowa c. szklarnią, numer strefy referencyjnej dla stref bezczujnikowych, rodzaj gleby w strefie, ilość godzin wprzód do obliczania prognozowanych wartości oczekiwanej opadu i ewapotranspiracji, funkcje statystyczną dla obliczania wartości temperatury i wilgotności z wielu czujników przypisanych do jednej strefy, ustalenie granicznych temperatur pracy dla czujników podłączonych do strefy, ustalenie czy czujniki w strefie mają wykorzystywać do sygnalizacji pracy diodę LED)

e) dodawaniu i usuwanie czujników wilgotności gleby oraz edycję ich parametrów (adres MAC, klucz kryptograficzny, ilość i sposób podłączenia do czujnika elementów pomiarowych, przypisanie czujnika do strefy)

f) modyfikację parametrów sterownika (dane dostępowe brokera MQTT, serwera czasu i serwisu pogodowego, strefa czasowa, współrzędne geograficzne ogrodu, dni tygodnia bez wody, zakres dat nawadniania, maksymalną ilość jednocześnie nawadnianych stref, opcję zainstalowania czujnika przepływu cieczy, czas przejścia z trybu ręcznego do automatycznego)

g) dostęp do informacji lokalnych sterownika: data i czas, temperatura i wilgotność,

h) odczyt i ustalenie trybu pracy: automatyczny/manualny,

i) odczyt i ustalenie statusu alarmu: zazbrojony/rozbrojony,

j) pojemność i zajętość karty pamięci,

k) odczyt logów/zdarzeń sterownika dla wybranej daty,

l) wyświetlanie tabeli rankingowej stref algorytmu Optiserv,

m) dostęp do prognozy pogody i obliczonych na jej podstawie wartości: wartości oczekiwanej opadu, ewapotranspiracji, średniej temperatury i wilgotności.

D.3. Łączność ze sterownikiem

Aplikacja powinna nawiązywać i prowadzić komunikację ze sterownikiem w obrębie lokalnej sieci bezprzewodowej po uwierzytelnieniu za pomocą loginu i hasła. Dostęp do sterownika z internetu powinien być tylko i wyłącznie przez tunel VPN zestawiany przed uruchomieniem aplikacji pomiędzy telefonem użytkownika a ruterem lokalnej sieci bezprzewodowej sterownika.

Po ukończeniu specyfikacji wymagań przeprowadzono analizę wykonalności projektu zarówno w kontekście sprzętowym jak i oprogramowania. Przygotowano zestawienie komponentów potrzebnych do spełnienia specyfikacji, które w późniejszym czasie uszczegółowiono parametrami technicznymi, a jego ostateczna postać została przedstawiona w rozdziale 3.6 w tabelach nr 20 i 21. Wybierając moduły sprzętowe, kierowano się spełnieniem wymagań, dostępnością bibliotek je obsługujących dla platformy ESP32, dostępnością na rynku, rozmiarami fizycznymi oraz ceną. Niestety problemy z dostępnością elementarnych układów elektronicznych spowodowanych pandemią COVID-19[94] objawiły się podczas specyfikacji komponentów. Dotyczyło to czujnika ciśnienia, wilgotności i temperatury oraz regulatorów napięcia. Ostatecznie jednak udało udało się się skompletować listę z dostępnych na rynku produktów o wymaganych parametrach. W kwestii bibliotek programistycznych podjęto analogiczne działania. Kluczowymi stały się obsługujące chip LoRa (bezprzewodowej komunikacji pomiędzy sterownikiem i czujnikami), tworzące funkcjonalność asynchronicznego serwera web (komunikacja aplikacji mobilnej ze sterownikiem) oraz dające metody przetwarzania formatu tekstowego JSON (dane z serwisu meteorologicznego, pliki konfiguracyjne, komendy z aplikacji mobilnej). Pełną lista bibliotek z krótkim opisem, adresem repozytorium, rodzajem licencji oraz wykorzystaniem w projekcie zamieszczono w „Spisie bibliotek programistycznych” na str. nr 142.

3.3 Modelowanie systemu nawadniania i jego urządzeń

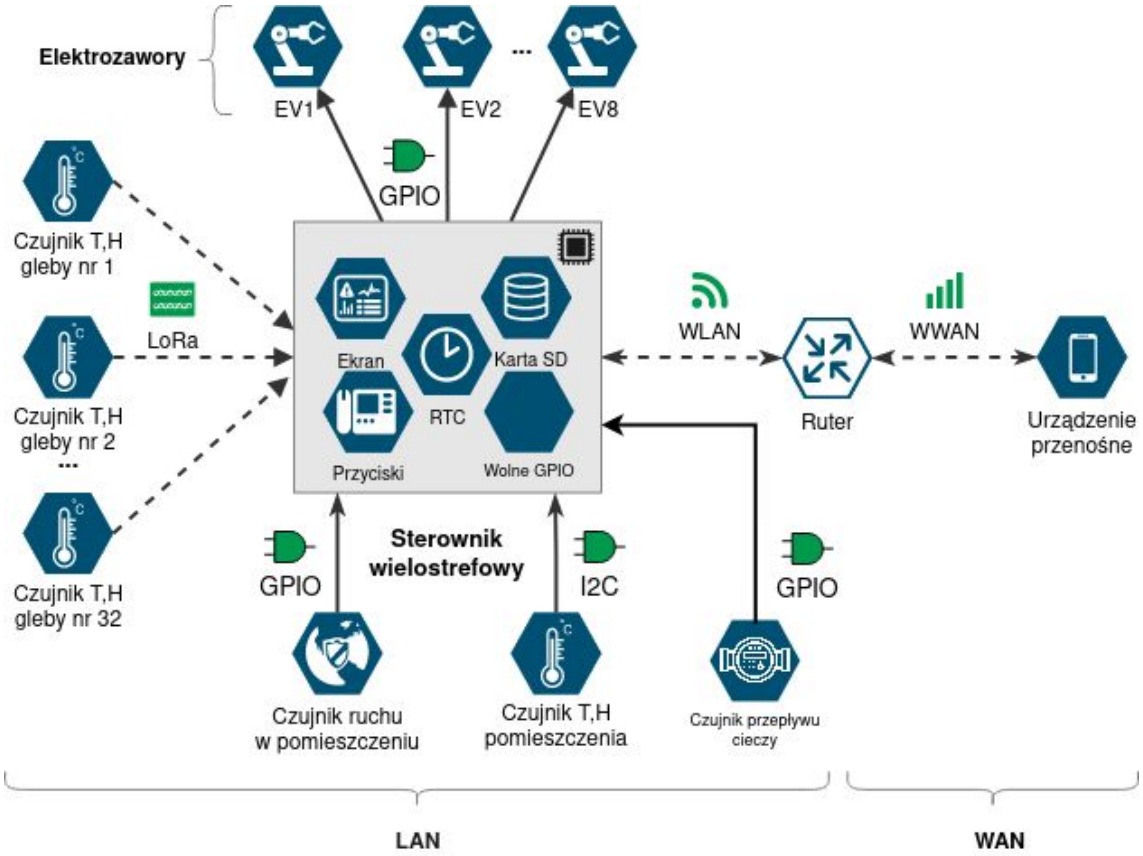
Na podstawie specyfikacji wymagań (z oznaczeniem [H] - hardware) został stworzony diagram z komponentami sterownika nawadniania, który został przedstawiony na rysunku nr 36. Daje on ogólny obraz na architekturę projektowanego systemu, dodatkowo ukazując w jego zakresie styk sieci lokalnej z internetem.

Natomiast analiza wyspecyfikowanych wymagań doprowadziła do powstania diagramu przypadków użycia sterownika oraz czujnika wilgotności, które zostały zamieszczone na rysunkach nr 37 i 38. Uwagę należy zwrócić na na dwa typy użytkowników: zdalnego i lokalnego (uogólnionych do użytkownika). Zdecydowanie ilość możliwych działań jest większa u tego tego pierwszego, rozumianego jako aktora obsługującego aplikację mobilną. Przypadki przypisane jedynie do użytkownika lokalnego związane są zabezpieczeniami oraz dostępem i są to wprowadzanie do sterownika poświadczeń lokalnej sieci bezprzewodowej oraz reset hasła dostępu zdalnego do sterownika. Wszystko to spowoduje uproszczenie fizycznego interfejsu użytkownika w sterowniku, że rzecz większej liczby funkcjonalności w aplikacji na urządzeniach przenośnych. Natomiast w strefowym czujniku wilgotności opcjonalnym działaniem jest ręczne wymuszenie pomiarów i transmisję danych do sterownikiem, uwidaczniające użytkownikowi na diodach led skwantowaną do trzech stanów wilgotność gleby.

Wśród aktorów nieożywionych kluczowy jest serwis pogodowy wymagający połączenia z internetem. Szczególnie istotne jest w jego przypadku obsługa wyjątków oraz filtrowanie danych, aby bez powodu nie zużywać zasobów pamięci operacyjnej układu ESP32 jaki i jego mikroprocesora podczas przetwarzania prognoz zawierających dane w ujęciu godzinowym. Jest to tym bardziej zasadne, że większość otrzymywanych danych jest w tym przypadku niepotrzebna. Innym, istotnym aktorem jest broker MQTT, pośredniczący w bezzwłocznym przekazywaniu komunikatów alarmowych ze sterownika do swoich klientów, którymi może być telefon komórkowy lub system domu inteligentnego.

Kolejny utworzony model dotyczył wymiany komunikatów transmitowanych pomiędzy sterownikiem a czujnikami (LoRa) oraz sterownikiem i internetowym serwisem pogodowym (HTTP). Zgodnie z założeniami algorytmu Optiserv, nawadnianie jest dopuszczalne od wschodu do zachodu Słońca (wraz z opcjonalnym, definiowanym przez użytkownika przesunięciem), dlatego też wymiana wiadomości została rozpięta na osi czasu właśnie z takimi punktami granicznymi. Każdego dnia po północy sterownik powinien zaktualizować informacje o godzinie wschodu i zachodu Słońca. Następnie w regularnych odstępach czasów powinna się odbywać aktualizacja prognoz hydrometeorologicznych dotyczących temperatury i wilgotności powietrza oraz opadów deszczu, które są dostarczane w ujęciu godzinowym. Z drugiej strony do sterownika spływają raporty z pomiarami środowiskowymi. Za obowiązkowe uznane zostały cztery: przed wschodem Słońca, po około 1/3 i 2/3 długości dnia oraz po zajściu gwiazdy za horyzont .Opcjonalna transmisja danych zachodzi po przekroczeniu zdefiniowanych progów wilgotności gleby i temperatury oraz na żądanie użytkownika (naciśnięcie przycisku w czujniku). Warto podkreślić, że w celu oszczędności energii - pomimo regularnych pomiarów - nie

wszystkie zestawy danych pomiarowych podlegają wysyłce, jedynie te obowiązkowe oraz opisane powyżej opcjonalne przypadki. Zobrazowanie tych informacji w postaci diagramu sekwencji zostało zamieszczone na rysunku nr 39.

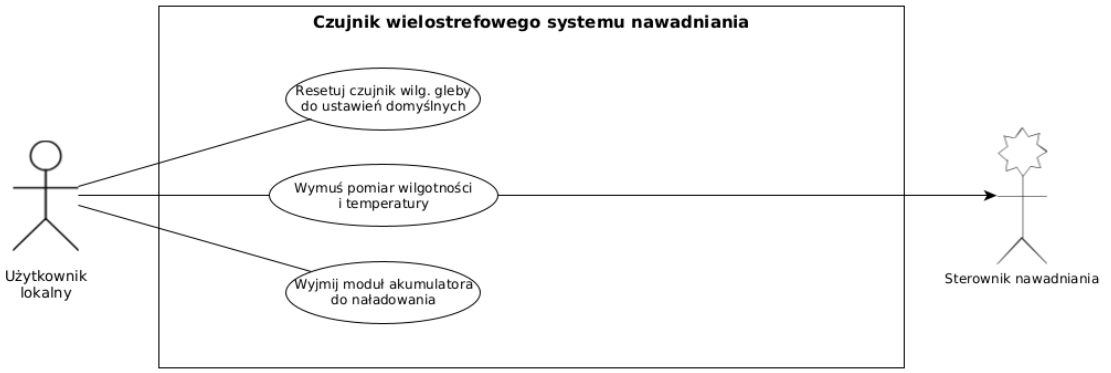


Rysunek 36: Komponenty wielostrefowego sterownika nawadniania

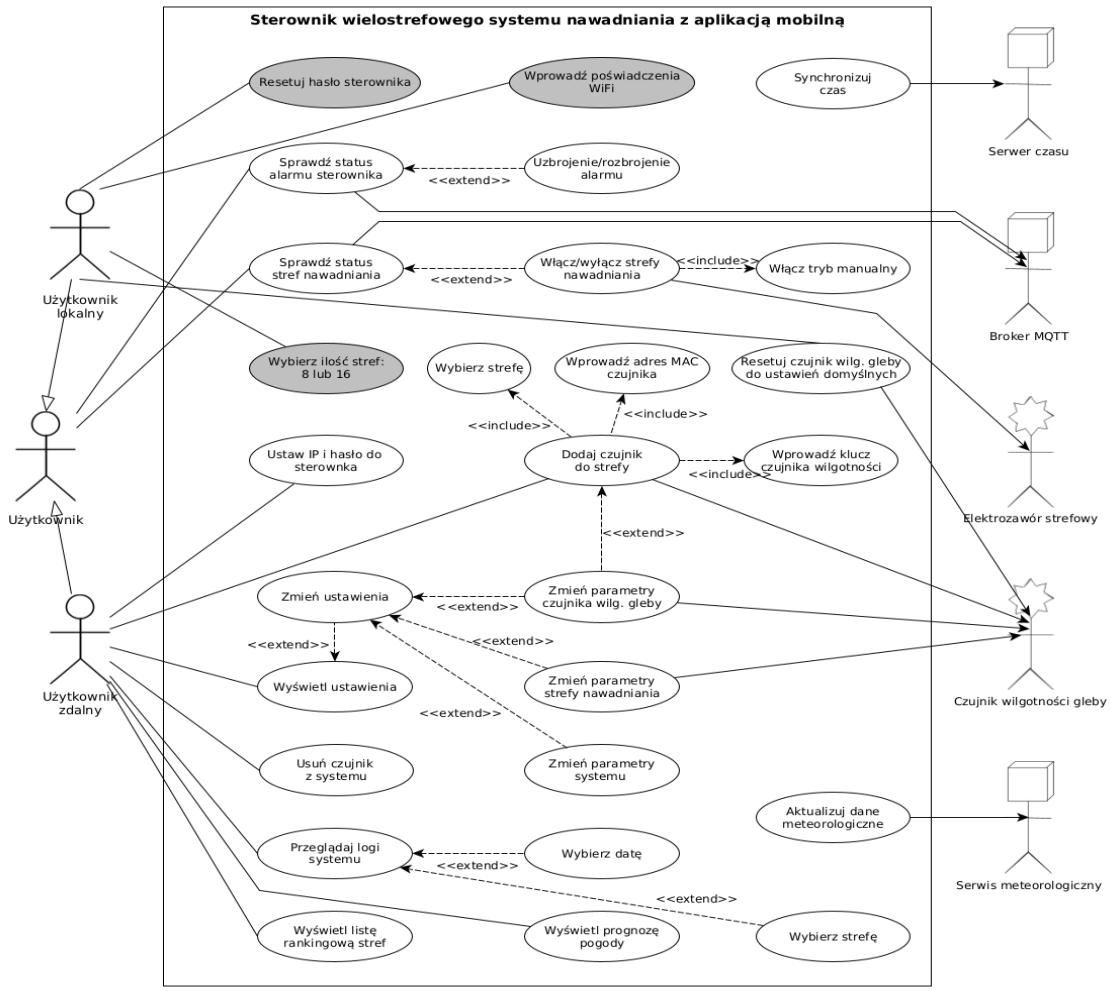
Sterownik powinien posiadać dwa tryby pracy: normalny i konfiguracyjny, a ostatni z nich, ze względów bezpieczeństwa, może być uaktywniony wyłącznie przez użytkownika lokalnego. Odpowiadające przypadki użycia na diagramie z rys. 38 otrzymały szary kolor tła, natomiast ich realizacja ma przebiegać przez przeglądarkę internetową lub aplikację mobilną. Przejście do trybu konfiguracyjnego ma nastąpić, jeżeli w trakcie uruchamiania urządzenia zostanie przytrzymany przycisk funkcyjny. W poniższej tabeli nr 15 przedstawiono tryby pracy sterownika w zestawieniu z trybami pracy jego karty radiowej oraz warunkami pod jakimi zachodzą.

Tabela 15: Tryby pracy sterownika oraz jego karty radiowej

Lp.	Tryb pracy sterownika	Tryb pracy karty radiowej WiFi	Warunek	Nazwa sieci
1	konfiguracyjny	AP (Access Point)	Brak konfiguracji parametrów sieci bezprzewodowej lub brak możliwości nawiązania połączenia z siecią	Sterownik Optiserv
2	konfiguracyjny	STA (Station)	Połączenie z siecią bezprzewodową	<<podana przez użytkownika>>
3	normalny	STA (Station)	Połączenie z siecią bezprzewodową	<<podana przez użytkownika>>



Rysunek 37: Diagram przypadków użycia dla bezprzewodowego czujnika wilgotności gleby



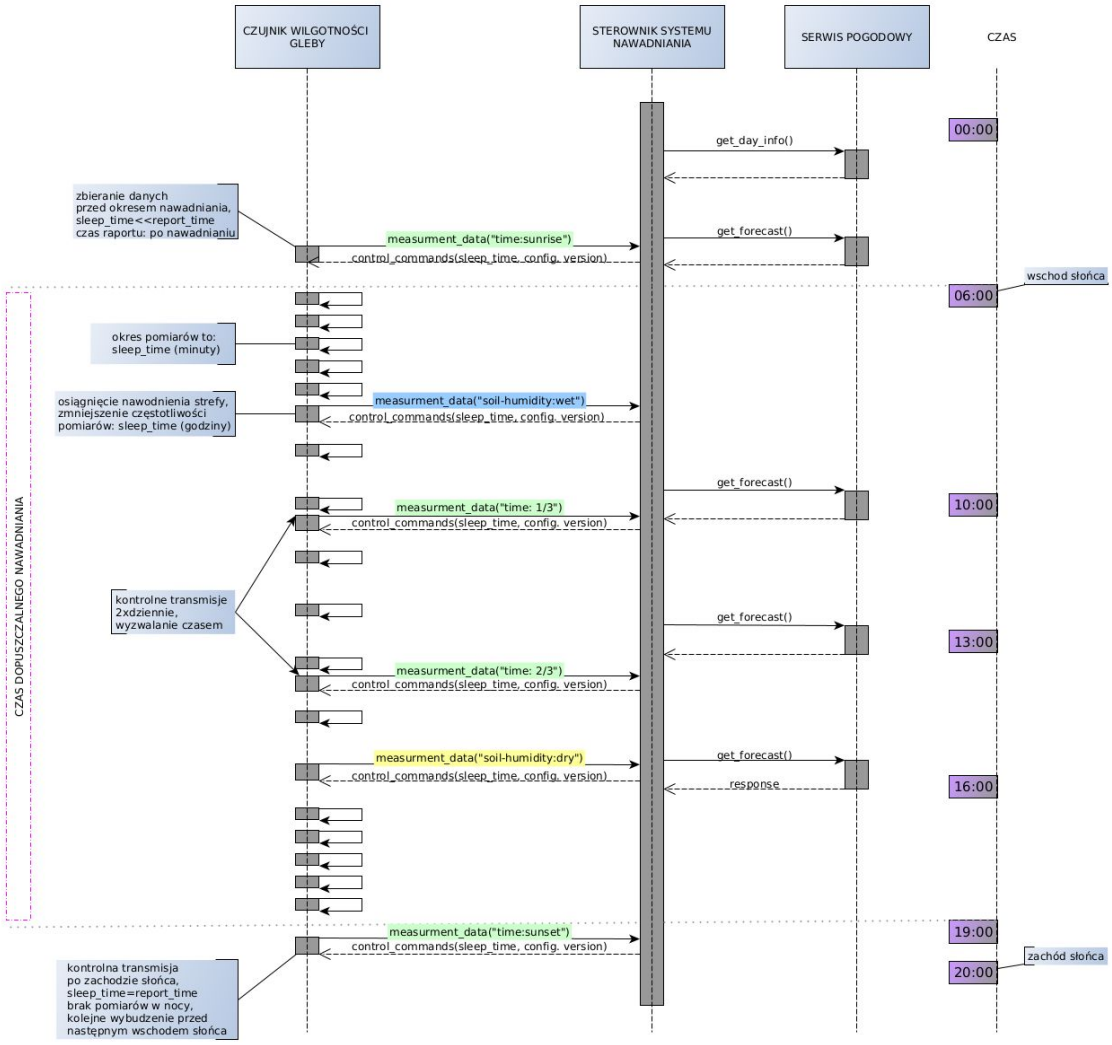
Rysunek 38: Diagram przypadków użycia dla sterownika wielostrefowego z aplikacją mobilną

Aby umożliwić zmianę parametrów konfiguracyjnych czujnika, który posiada niezwykle ograniczony interfejs użytkownika (1 dioda LED + 1 przycisk), dane konfiguracyjne muszą dotrzeć do niego również drogą radiową, przy wykorzystaniu protokołu LoRa, zapewniającym znacznie większy zasięg i mniejsze zużycie energii w stosunku do sieci WiFi. Wprowadzono więc uwarunkowaną odpowiedź sterownika z aktualnymi ustawieniami. Każda wiadomość

pochodząca z czujnika, może być traktowana jako raport środowiskowy i składa się z wartości danych pomiarowych i aktualnej wersji konfiguracji tak jak poniżej.

Procedura 2: Raport środowiskowy z czujnika wilgotności, JSON

```
{
  "SH": 2078,      //wilgotność gleby
  "Ts": 26.5,      //temperatura gleby
  "Ta": 26.3,      //temperatura powietrza
  "H": 40.99,      //wilgotność powietrza
  "B": 89,         //stopień naładowania baterii
  "W": 0,          //przyczyna wybudzenia czujnika
  "v": 4           //wersja konfiguracji czujnika
}
```



Rysunek 39: Diagram sekwencji wymiany komunikatów pomiędzy sterownikiem i czujnikiem wilgotności

W przypadku gdy numer wersji zapisanej w sterowniku jest identyczny w stosunku do tej otrzymanej z czujnika, jego odpowiedź ma następująca strukturę:

Procedura 3: Podstawowa odpowiedź sterownika na raport środowiskowy, JSON

```
{
   "nr": 10800,    //czas następnego raportu w sekundach
   "v": 4          //aktualna wersja konfiguracji czujnika
}
```

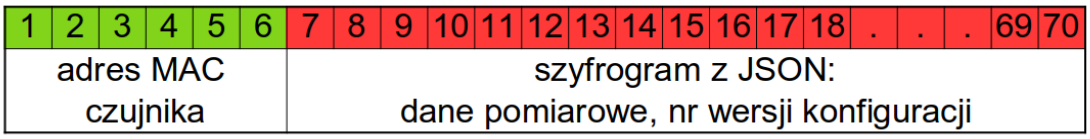
Gdy zaś numery wersji nie są zgodne, zestaw danych jest rozbudowany właśnie o parametry konfiguracyjne, które od momentu otrzymania powinny być trwale zapisane w czujniku:

Procedura 4: Rozszerzona odpowiedź sterownika na raport środowiskowy, JSON

```
{
   "nr": 10800,          //czas następnego raportu w sekundach)
   "v": 5,               //aktualna wersja konfiguracji czujnika
   "hum_option": 1,      //sposób montażu elem. pomiar. w czujniku
   "hum_dry": 2000,      //wilgotność - próg wyzwalania nawadniania
   "hum_wet": 1200,      //wilgotność - próg wyłączenia nawadniania
   "temp_hi": 35,        //maksymalna dopuszczalna temperatura pracy
   "temp_lo": 2,         //minimalna dopuszczalna temperatura pracy
   "led": 1              //opcja wykorzystania diody LED do
                           sygnalizacji pracy
}
```

Parametr `hum_option` jest przypisywany indywidualnie do każdego czujnika i definiuje w których gniazdach zostały podłączone elementy wykonawcze, natomiast ostatnie 5 parametrów odnosi się do wszystkich czujników w danej strefie.

Dane przesyłane przy użyciu LoRa przesyłane są otwartym tekstem. Aby spełnić wymaganie A.3.a, dane w ruchu powinny zostać zaszyfrowane. Wykorzystując rodzaj szyfrowania, w którym długość szyfrogramu jest równa długości wiadomości, struktura każdej wiadomości może przyjąć poniższą postać (rys. 40). Pierwsze 6 bajtów to unikalna liczba, będąca adresem MAC karty radiowej czujnika (nie używanej przez niego) przesyłana jawnie, natomiast pozostałe 64 bajtów stanowi zaszyfrowana właściwa treść komunikatu w postaci JSON. Długość została dobrana tak, aby pomieścić wiadomość z największymi wartościami parametrów z kilkunastobajtowym zapasem.

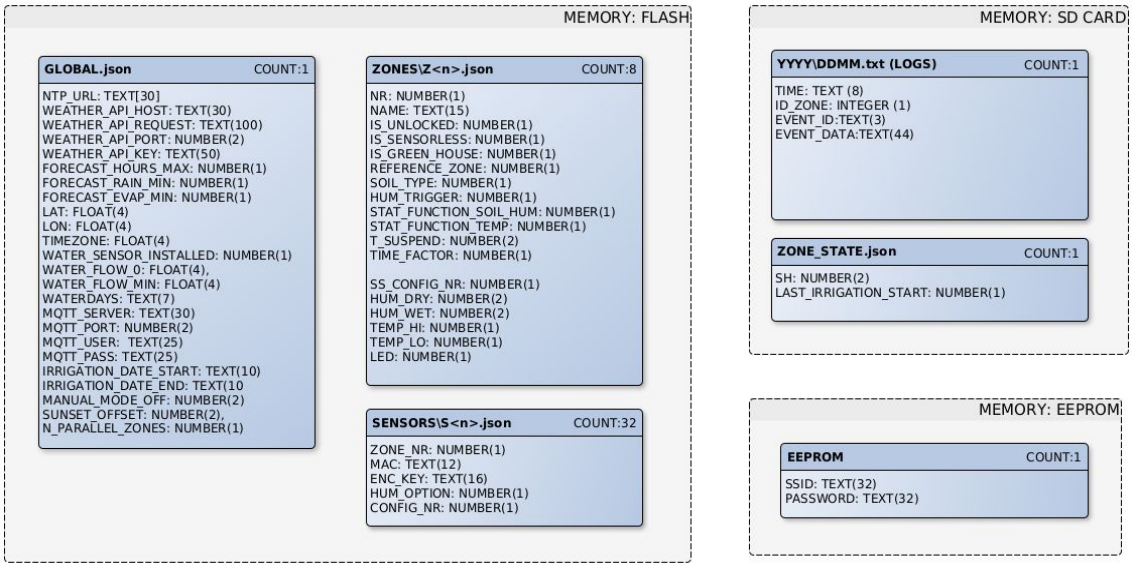


Rysunek 40: Budowa wiadomości LoRa (w bajtach)

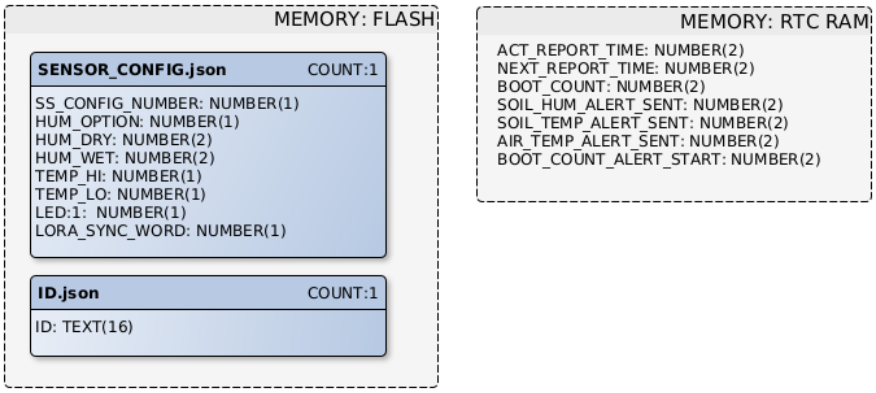
Następną wymagająca zaplanowania kwestią stało się miejsce i sposób przechowywania danych. W całym projekcie za format przechowywania i wymiany danych został wybrany opisany w podrozdziale 1.5 format JSON. Posłużył więc on do przechowywania konfiguracji globalnej, strefowej i dotyczącej czujników w sterowniku, a także konfiguracji w samym czujniku. Uwzględniając ograniczenie żywotności (ilość zapisów/sektor) pamięci Flash oraz jej wielkość, ostatecznie rozlokowanie struktur danych przyjęło postać jak rysunkach

odpowiednio 41 i 42. Każdej zmiennej przypisano nazwę, typ oraz wielkość w bajtach. Zaczynając od sterownika i pamięci typu *flash*, plik `GLOBAL.json` przechowuje konfigurację dotyczącą tworzonego systemu, pliki `Z1-Z16.json` odpowiadają za ustawienia poszczególnych stref, natomiast S1-S32.json zawierają dane dodanych do systemu czujników wilgotności. Dane te, z racji, że stanowią konfigurację, nie będą zmieniały się często, stąd wybór rodzaju pamięci. Zupełnie odmienny charakter mają dane dotyczące zdarzeń i alertów w systemie (LOGS) oraz plik odzwierciedlający zmienne stanu potrzebne w Optiserv w wypadku utraty zasilania (`ZONES_STATE.json`). Przewidywana częste dopisywanie i nadpisywanie plików spowodowało, że zostały one przeznaczone do umieszczenia na karcie pamięci. Do utrwalenia SSID sieci bezprzewodowej oraz odpowiadającego jej hasła, została wykorzystana nieulotną pamięć EEPROM dostępna w układzie ESP32.

Przechodząc do czujnika wilgotności (rys. 42), analogicznie do sterownika w pamięci flash zapisywana jest konfiguracja (`SENSOR_CONFIG.json`). 128-bitowy klucz kryptograficzny przechowywany jest natomiast w pliku ID.json i jest unikalnym numerem seryjnym termometru cyfrowego, będącym jednym z elementów pomiarowych. Pomimo, że nie należą do pamięci nieulotnej, na rysunku nr 42 przedstawiono również zmienne stanu przechowywane w pamięci RTC RAM. Zmienne te zachowują swoje wartości po wyjściu ze stanu głębokiego uśpienia, w którym to czujnik przebywa przez większość czasu od uruchomienia.



Rysunek 41: Organizacja struktur danych w pamięci sterownika



Rysunek 42: Organizacja struktur danych w pamięci czujnika wilgotności

W oprogramowaniu sterownika zaplanowano 4 klasy, `Settings`, `Zone`, `Sensor`, `WeatherForecast` opisujące struktury odpowiednio: ustawień (1 obiekt), stref (8 obiektów/16 obiektów), czujników (do 64 obiektów) oraz prognozy pogody (1 obiekt). Z racji iż większość pól trzech wymienionych na początku klas pokrywa się z zawartością plików json z rysunku nr 41, stąd też dla przykładu na poniższym listingu podano definicję ostatniej klasy WeatherForecast, w wykorzystywanym w implementacji języku C++ wraz z opisem pól i metod.

Procedura 5: Definicja klasy WeatherForecast

```
struct WeatherForecast {
    float temperature_avg = 0;       //średnia temperatura (obliczana)
    float rain_ex = 0;               //oczekiwana wartość opadu (obliczana)
    float humidity_avg = 0;          //średnia wilgotność (obliczana)
    float ET = 0;                    //ewapotranspiracja (obliczana)
    uint32_t updated = 0;            //czas aktualizacji prognozy
    uint32_t sunrise = 0;            //czas wschodu Słońca
    uint32_t sunset = 0;             //czas zachodu Słońca
    uint32_t t_suspend_rain = 0;     //czas wstrzymania nawadnianiania z powodu
                                      prognozowanych opadów desczu
    uint32_t t_suspend_evapor = 0;   //czas wstrzymania nawadnianiania z
                                     powodu prognozowanej ewaportranspiracji
    uint8_t wifi_try = 0;            //ilość prób poł. Z WiFi
    uint8_t upd_try  = 0;            //ilość prób poł. Z serwisem pogodowym
    Settings* config;                //wskaźnik do obiektu z konfiguracją
                                     (obiekt klasy Settings)
    bool updateForecast();           //aktualizacja danych hydrometeorol.
    void calculate_ET();             //obliczenie wartości ewapotranspiracji
};
```

Do przechowywania czasu w formacie *Unix time* w zmiennych `updated`, `sunrise`, `sunset`, `t_suspend_rain`, `t_suspend_evapor` wytypowano użycie zmiennej 32-bitowej jednak bez znaku (unsigned). Jest to jeden ze sposobów rozwiązania *problemu roku 2038*[95], wydłużający prawidłowe działanie systemu do 2106r., kosztem braku możliwości zapisania dat przed rokiem 1970, które w projektowanym systemie nie mają zastosowania.

W celu umożliwienia użytkownikowi zdalnemu realizacji przypisanych mu przypadków użycia (rys. 38) zaprojektowano węzły końcowe, których nazwy, metoda protokołu, parametry wymagane oraz wartości zwracane zostały przestawione w poniższej tabeli nr 16.

Tabela 16: Definicja węzłów końcowych sterownika

Nazwa węzła końcowego	Metoda HTTP	Wymagane parametry	Zwracane wartości	Opis
/delete-sensor	GET	id	msg	usuwanie czujnika o podanym id
/forecast	GET	-	temp, humd, rain, updated, et	pobranie wartości prognoz meterologiczych
/get-global-settings	GET	-	plik: global.json	pobranie konfiguracji sterownika
/get-info	GET	-	date_time, work_mode, alarm_status	pobranie daty, trybu pracy i statusu alarmu
/get-logs	GET	date	plik: [date].txt	pobranie zdarzeń z wybranego dnia
/get-sensor-settings	GET	sens_nr	plik: [sens_nr].json	pobranie ust. wybranego czujnika
/get-sensors	GET	-	sensors.json	pobranie listy czujników
/get-zone-ranking	GET	-	ranking.json	pobranie rankingu stref
/get-zone-settings	GET	zone_nr	Z[zone_nr].json	pobranie ustawień wybranej strefy
/get-zone-values	GET	-	zone_status, date_time, work_mode, alarm_status	odczyt statusu nawadnianych stref
/s3deODwaDWy	GET	value	alarm_status	zmiana statusu alarmu
/save	POST	data, file	msg	zapis plików konfiguracyjnych
/set-work-mode	GET	value	work_mode	ustawienie trybu pracy
/set-zone-values	GET	zone_nr, value	msg	włączanie wyłączanie wybranej strefy
/resetpwd	GET	-	domyślne hasło	przywrócenie domyślnego hasła
/reboot	GET	-	-	reset sterownika
/N	GET	N	-	wybór ilości stref 8/16

Ostatnie trzy pozycje dotyczą trybu konfiguracji.

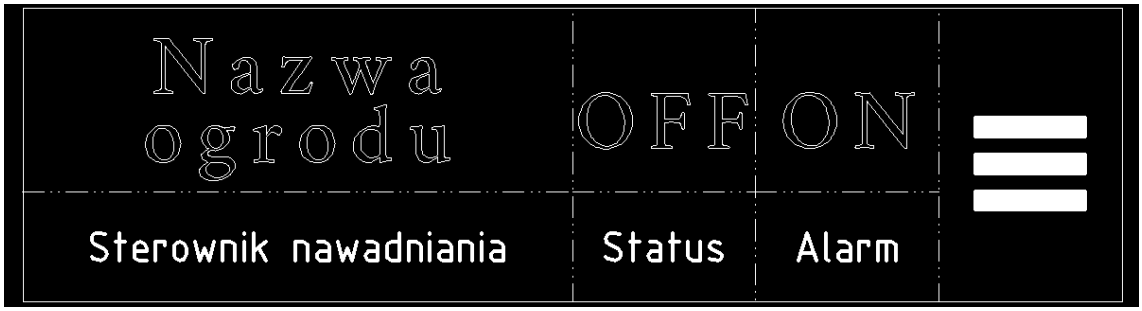
3.4 Projekt aplikacji mobilnej dla systemu Android

Aplikacja powinna działać w trybie pełnoekranowym, pozostawiając widoczny jedynie pasek systemowy. W przypadku braku możliwości nawiązania komunikacji ze sterownikiem, w celu naprowadzenia użytkownika na możliwą przyczynę problemu, należy wyświetlić komunikat ze stosowną informacją oraz linkami o nazwach i znaczeniu:

1. Ustawienia WiFi - ustawień systemowych związanych z sieciami bezprzewodowymi,
2. Ustawienia VPN - ustawień systemowych dotyczących połączeń VPN,
3. Dane sterownika - formularza do odczytu/zmiany danych dostępowych do sterownika.

Ekran aplikacji powinien być podzielony na część nawigacyjną w postaci prostokątnej belki u góry ekranu oraz kontener do prezentowania treści na pozostałym obszarze. Główna belka aplikacji powinna zawierać następujące napisy, informacje i obiekty:

1. Nazwę ogrodu 2. Status instalacji nawadniania 3. Status alarmu 4. Menu



Rysunek 43: Widok głównej belki aplikacji mobilnej

Powyżej określona belka z menu nawigacyjnym powinna być zawsze widoczna, treści wykraczające poza pionowy rozmiar kontenera z treścią powinny być możliwe do przesunięcia w górę/dół. W tej sytuacji po prawej stronie kontenera powinien wyświetlać się pasek do przewijania. Wymaga się, aby menu główne po naciśnięciu jego ikony otwierało się oraz zamykało w sposób animowany, jednakże nie przekraczając czasu 500ms.
Mając na uwadze specyfikację wymagań (D) oraz przypadki użycia przypisane do użytkownika zdalnego (rys. 38) zaproponowano następującą strukturę nawigacji dostępną w menu głównym.

Tabela 17: Struktura menu głównego aplikacji mobilnej

Lp.	Nazwa	Funkcjonalności	Spełnienie wymagań
1	Stan stref	Wyświetlenie stanu stref z przełącznikiem włącz/wyłącz oraz ikoną koła zębatego po kliknięciu którego ma wyświetlać się lista parametrów strefy z możliwością edycji wartości. Na dole listy opcje anulowania zmian oraz ich zapisu. Wyświetlenie aktualnej daty i czasu sterownika. Wyświetlanie listy rankingowej algorytmu OptiServ.	D.2 b,c,d,l
2	Czujniki wilgotności	Wyświetlenie listy czujników: identyfikator, numer oraz ikona koła zębatego po kliknięciu którego istnieje możliwość przypisania strefy nawadniania do wybranego czujnika. Dodatkowo należy umożliwić edycję adresu MAC, klucza kryptograficznego oraz wybór opcji instalacji elementów pomiarowych w wybranym czujniku wilgotności.	D.2 e
3	Ustawienia sterownika	Wyświetlanie listy parametrów sterownika z możliwością ich edycji. Na górze i dole listy opcje anulowania zmian oraz ich zapisu.	D.2 f
4	Logi	Wyświetlanie zdarzeń(logów) ze sterownika po wcześniejszym wybraniu daty zdarzeń.	D.2 k
5	Prognoza pogody	Wyświetlanie wartości oczekiwanej opadu, ewapotranspiracji, średniej temperatury i wilgotności.	D.2 m
6	Wybór ogrodu	Zachowania nazwy ogrodu, adresu IP i hasła do sterownika.	D.2 a
7	Informacje	Wyświetlanie daty i czasu sterownika, aktualnych parametrów środowiskowych, informacji o karcie pamięci. Umożliwienie zmiany trybu pracy sterownika. Umożliwienie zmiany statusu alarmu.	D.2 g,h,i,j

Wybrana nazwa czcionki oraz kolory elementów aplikacji w notacji heksadecymalnej:

Czcionka: sans-serif

Kolor tła i ikon w menu: |……..| #F9F9F9

Kolor czcionki: #000000

Kolor belki głównej i menu: #36383F
Kolor ramek tabelki: #85888C
Kolor tła tabeli: #D1D1D1
Kolor statusu ok: #3AE04
Kolor statusu błąd: #E03A42

Ikonografia w menu ma odzwierciedlać wyglądem funkcjonalność każdą pozycji, a ikony powinny być płaskie, jednokolorowe oraz spójne ze względu na grubość tworzących je linii. Tabele przedstawiające listę czujników, ustawienia sterownika oraz logi, w celu poprawy jakości ich odbioru i ułatwienia rozróżniania pozycji, powinny mieć wiersze z naprzemiennie zmieniającym się kolorami: *kolor tła* oraz k*olor tła tabeli*. Tabela zawierająca parametry ustawień sterownika oraz czujnika wilgotności powinna udostępniać przy nazwie parametru ikonę informacyjną z literą „i", po kliknięciu której ma wyświetlać się objaśnienie znaczenia wybranego parametru. Wymaga się, aby okno z komunikatami z aplikacji pojawiało się nad aktualną treścią, która powinna być w tej sytuacji przyciemniona, przy zachowaniu widoczności belki głównej z menu.

Odświeżenie stanu stref nawadniania powinno się dokonać po:

1. uruchomieniu aplikacji,
2. powrotu aplikacji z pracy w tle,
3. wybraniu pozycji *Stan stref* z menu,
4. wciśnięciu napisu z nazwą ogrodu na belce główne,
5. każdorazowej zmianie stanu dowolnej ze stref dokonanej w aplikacji.

Mając na uwadze pracę aplikacji (klienta) oraz sterownika (serwera) w sieciach bezprzewodowych WiFi lub/oraz WWAN oraz występowanie zmiennych czasów realizacji żądań klienta (zlecanych przez użytkownika w aplikacji mobilnej) bezwzględnie wymaga się zastosowania animowanej grafiki w trakcie oczekiwania na odpowiedź. Czas przekroczenia odpowiedzi należy zdefiniować na 10 sekund, a jego przekroczenie powinno być sygnalizowane komunikatem ostrzegawczym.

3.5 Projekt urządzeń

Do zaprojektowania wszystkich płytek PCB zostało wykorzystane oprogramowanie EasyEDA w wersji 1.16 Standard dostępne jako aplikacja uruchamiana w przeglądarce internetowej[96]. Chronologicznie jako pierwsza została podjęta próba wykonania elementu wykonawczego czujnika wilgotności gleby. W razie niepowodzenia plan przewidywał wykorzystanie jednego z dostępnych na rynku urządzeń i włączenie go do strefowego czujnika wilgotności. Ostatnim, co również zostało odzwierciedlone w kolejności podrozdziałów, był sterownik nawadniania.

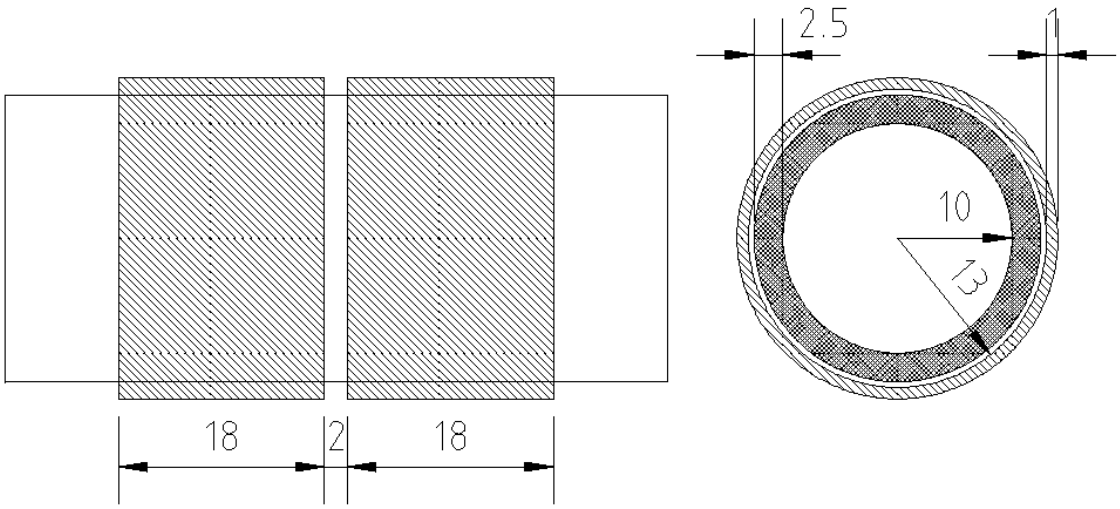
3.5.1 Element wykonawczy czujnika wilgotności gleby

Do określenia stanu wodnego gleby, ze względu na cechy wymienione w podrozdziale 1.1.3.1 ‘Stan wodny gleby i metody jego określania’, została przyjęta metoda dielektryczna w wersji pojemnościowej. Właściwości te dobrze wpisują się w specyfikację wymagań, podkreślając szczególnie niski koszt energetyczny pojedynczego pomiaru, co ma znaczenie dla urządzenia zasilanego bateryjnie.

3.5.1.1 Wersje rozwojowe elementu wykonawczego i doboru układu pobudzającego

Pierwsza koncepcja konstrukcyjna polegała na umieszczeniu współosiowo dwóch pierścieni miedzianych (Φ_{zew}=28mm), osadzając je na polipropylenowej rurze(Φ_{zew}=25mm), wszystko zaś pokrywając rurką termokurczliwą, zgodnie z poniższym rysunkiem nr 44. Natomiast pierwsza idea pomiaru pojemności tak zbudowanego kondensatora polegała na stworzeniu szeregowego układu RC i pomiarze czasu jego rozładowywania. Na podstawie obliczeń przy użyciu kalkulatora kondensatorów kooplanarnych[97] oszacowano rząd wielkości pojemności dwóch współosiowych pierścieni miedzianych w różnych ośrodkach:
w powietrzu 2pF $(\varepsilon_r \approx 1)$, w glebie dla 10pF $(\varepsilon_r \approx 5)$, oraz w wodzie dla 100pF $(\varepsilon_r \approx 80)$. Stąd przy założeniu rezystancji opornika na poziomie 10kΩ uzyskano stałe czasowe RC około 1s dla wody i 100ms dla gleby. Do pomiaru czasu wykorzystano Arduino Pro Mini 328 3.3V/8MHz oparty na mikrokontrolerze ATmega328, który docelowo miał się znaleźć wewnątrz polipropenowej rury. Idea i implementacja pomiaru czasu w celu określenia pojemności kondensatora została wzięta z przykładu z oficjalnej strony Arduino[98], jednakże sam sposób pomiaru czas przy użyciu funkcji `millis()` wykazywał zbyt małą dokładność w stosunku do oszacowanych potrzeb.

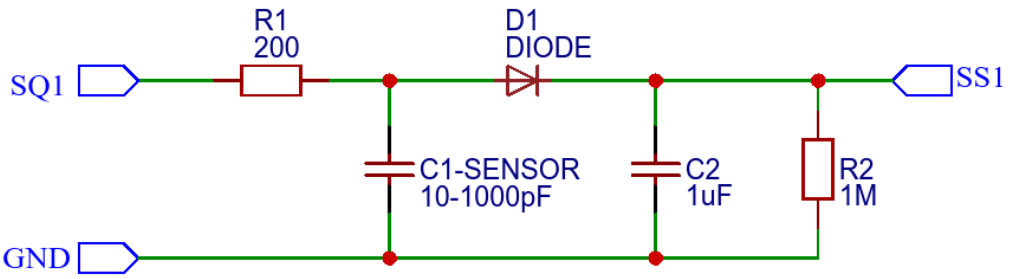
Na podstawie kursu[99] układ Arduino został zaprogramowany w oparciu o dostępny 16-bitowy Timer1 z wykorzystaniem jego opcji Input Compare do zapamiętania stanu licznika w momencie zajścia zewnętrznego zdarzenia. Zdarzeniem tym było zmiana stanu z wysokiego na niski, jednego z wyprowadzeń pracujących w trybie wysokiej impedancji (INPUT). Poprzez bezpośrednie operacje na rejestrze PORTD oraz zliczaniu taktów procesora wraz uwzględnieniem przepełnień i wykorzystaniu przerwań do zakończenia zliczania możliwe było osiągnięcie rozdzielczości na poziomie 1/8MHz = 0,125μs).



Rysunek 44: Pierwsza koncepcja budowy kondensatora do elementu wykonawczego

Wyniki pomiarów wyświetlane były na konsoli komputera. Układ pomiarowy został sprawdzony podczas pomiarów kondensatorów ceramicznych 4,7pF-270pF dając dokładność do około 7%. Jednakże rzeczywiste próby pomiarów pojemności kondensatora z pierścieni miedzianych za pomocą Arduino w wodzie(w osłonie rurki termokurczliwej) i w powietrzu dawały zbyt małą różnicę, aby dała się ona wykorzystać do określenia stanu wilgotności gleby. Ustalono, że wynikało to między innymi z faktu, że do całego układu wchodziła pojemność pasożytnicza połączeń i przewodów połączeniowych. W docelowej konstrukcji musiałby one zostać zredukowane do ścieżek. Inne problemy związane z tą koncepcją to stosunkowo duże wymiary i grubość pierścieni nie uwzględnione w obliczeniach. Ostatecznie zbyt mała rozpiętość wyników, potencjalne problemy konstrukcyjne związane z zachowaniem niewielkich odległości pomiędzy mikrokontrolerem i pierścieniami doprowadziły do porzucenia omawianej koncepcji. Jak pokazały dalszy przegląd literatury takie podejście może być skutecznie zastosowane[100].

W drugiej koncepcji budowy układu wykonawczego do czujnika wilgotności gleby zmieniono zarówno budowę kondensatora jak również charakter jego pobudzenia. Idea polegała na pomiarze napięciowej odpowiedzi układu RC na pobudzenie sygnałem prostokątnym o określonej częstotliwości. Na podstawie pracy[101] za obwód do pobudzenia przyjęto filtr dolnoprzepustowy wraz z detektorem szczytowym, zgodnie z poniższym rys. nr 45.



Rysunek 45: Schemat elektryczny elementy wykonawczego czujnika wilgotności gleby

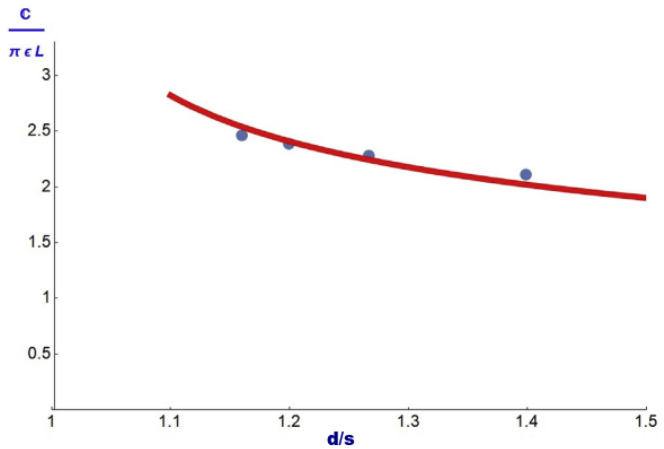
Rezystor R1 i C1 stanowią rzeczony filtr RC, pobudzany przez wejście SQ1. Pomiar napięcia szczytowego następuje na wyjściu SS1, które jest napięciem kondensatora C2, doładowywanego przez diodę D1. Stała czasowa obwodu R2C2 na poziomie 1s pozwala utrzymać napięcie szczytowe (pomniejszone o spadek na diodzie D1) dostatecznie długo, aby udało się je zmierzyć. Kluczowy jest kondensator C1-SENSOR, którego pojemność zależy od względnej przenikalność elektryczna ośrodka w którym się znajduje (powietrze, gleba, woda). Zanurzenie go w wodzie, zwiększa jego pojemność elektryczną zmniejszając za razem jego reaktancję, co z kolei wpływa na mniejszą wartość napięcia, która się na nim odkłada w stosunku do sytuacji gdy byłby umieszczony w powietrzu. Ze względu na uwagi dotyczące częstotliwości sygnału pobudzającego, poczynione na końcu podrozdziału 1.1.3.1, za pobudzenie zamiast układu timera 555 z pracy[100] generującego przebiegi do około 430 kHz, przyjęto - programowalny przez magistralę I^2C - generator sygnału oparty o układ Si5351A. Jego zakres wyjściowy zawiera się w przedziale 8 kHz do 160 MHz[102], dając łatwą możliwość zmiany częstotliwości pobudzenia. Do sterowania generatorem użyto poprzednio wykorzystaną platformę Arduino Pro Mini 328. W oparciu o pracę [103] zlecono stworzenie dwóch płytek PCB, których jednakowy po obu stronach przewodzący nadruk stanowił dwa równolegle połączone kondensatory, co zostało przedstawione na rysunkach 47 – wersja A (wzorowana na płytce z przytoczonego opracowania[102]) oraz 48 w wersji B według własnej, uzasadnionej poniżej koncepcji. Pierwsza z nich, ze względu na większy obwód pozwala zgromadzić więcej ładunku ze względu na większy długość obwodu ścieżek[104], jednakże druga zapewnia:

a) głębszą penetrację ośrodka w którym się znajduje oraz osiąga[105],

$$T = a \cdot \sqrt{\left(1 + \frac{w}{a}\right)^2 - 1} \tag{12}$$

gdzie: T–głębokość penetracji, a–połowa odległości między elektrodami, w–szerokość elektrody

b) większą pojemność na jednostkę długości (na podstawie poniższego wykresu i tab. nr 18)



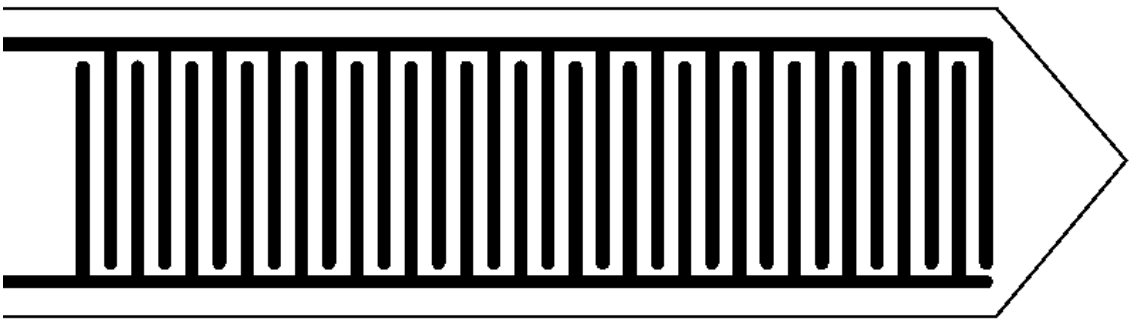
Rysunek 46: Przeskalowana zależność pojemności na jednostkę długości elektrod od stosunku odległości między symetralnymi elektrod przez ich szerokość[103]

W poniższej tabeli nr 18 przedstawiono zestawienie parametrów obu projektów dotyczących geometrii kondensatorów wraz z obliczoną głębokością penetracji oraz efektywną pojemnością (w odniesieniu do długości elektrod).

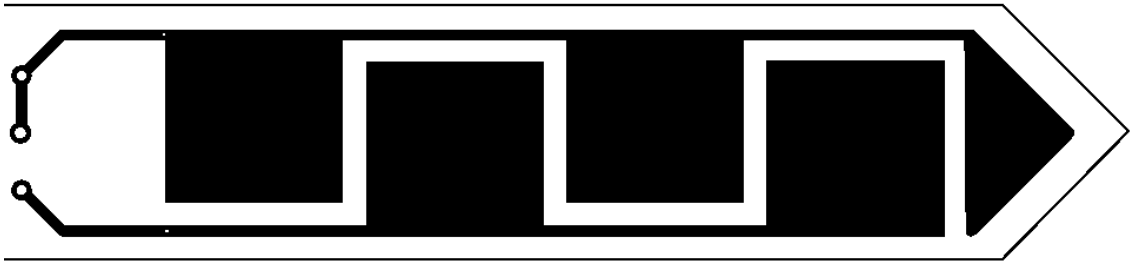
Tabela 18: Zestawienie parametrów płytek w wersji A i B

Opis	Oznaczenie	Wersja A	Wersja B
odległość pomiędzy elektrodami [mm]	w	1	2
szerokość elektrody	s	1	16
odległość pomiędzy symetralnymi elektrod [mm]	d	2	18
połowa odległości między elektrodami [mm]	$a=\frac{w}{2}$	0,5	1
głębokość penetracji [mm] (wzór nr 12)	**T**	**1,41**	**2,83**
iloraz d / s	$\frac{d}{s}$	2	1,13
przeskalowany stosunek pojemności do długości (wyznaczony z wykresu na rys. 46)	$\frac{C}{\Pi\cdot\varepsilon\cdot L}$	**<1,8**	**2,8**

Z powyższych danych za szczególnie istotną uznano głębokość penetracji otaczającego ośrodka, ze względu na zwiększenie szans na wykrycie wody, w obliczu tworzących się w glebie pustek powietrznych z powodów naturalnych lub nieprawidłowej instalacji czujników wilgotności[106].



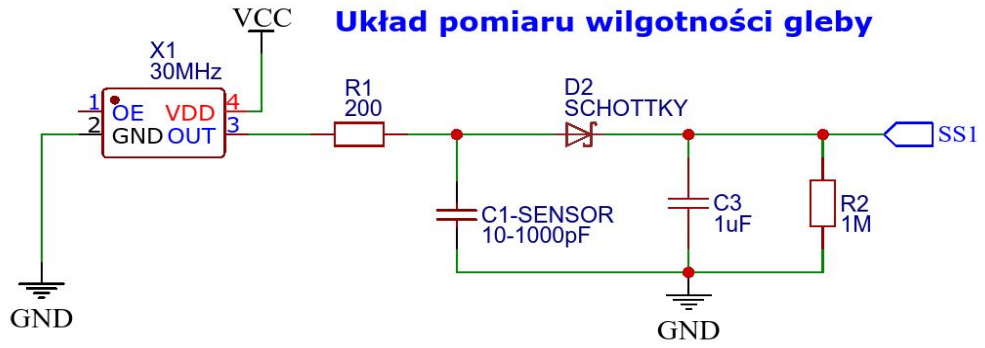
Rysunek 47: Kondensator PCB – wersja A (na podstawie [102])



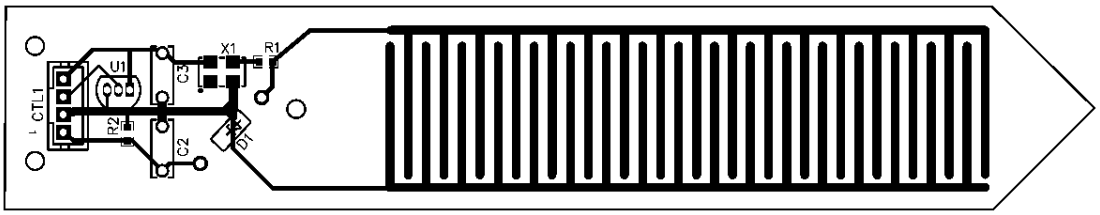
Rysunek 48: Kondensator PCB – wersja B (autorska)

Przeprowadzono szereg prób z różnymi częstotliwościami pobudzenia. Dla częstotliwości w przedziale 30-32MHz uzyskano wyniki podczas zanurzenia w wodzie na poziomie około 1.9V

przy 3.3V w powietrzu. Doprowadziło to zaprojektowania i wykonania układu w którym generatorem sygnału prostokątnego stał się oscylator kwarcowy HSO531S, o częstotliwości drgań 30Mz i maksymalnym natężeniu wyjściowym 25mA[107]. Schemat elektryczny układu pomiarowego przedstawiono na poniższym rys. nr 49, zaś projekt płytki na kolejnym rys. 50.



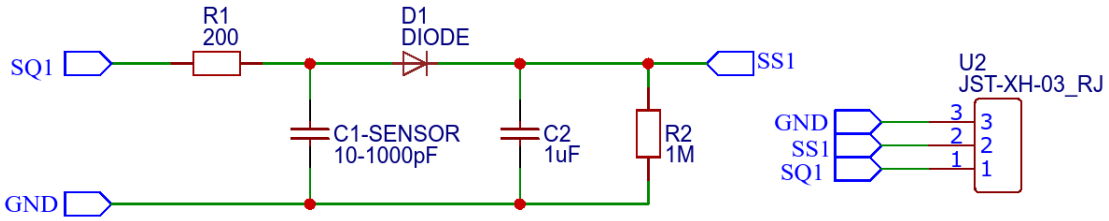
Rysunek 49: Układ pomiaru wilgotności gleby z oscylatorem kwarcowym 30MHz



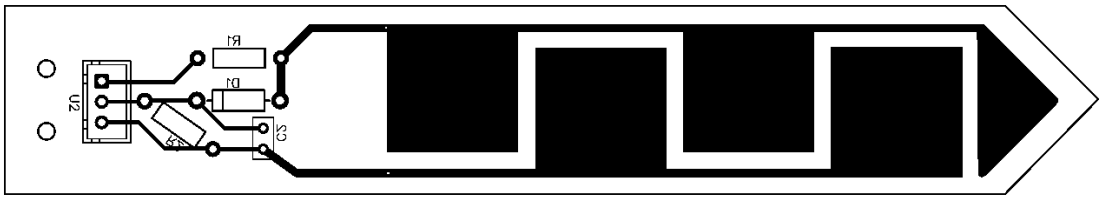
Rysunek 50: Projekt PCB elementu wykonawczego z oscylatorem kwarcowym

Uzyskane rezultaty były zbieżne z wcześniejszymi tzn. podczas pomiarów w wodzie osiągnięto spadek napięcia z 3.3V do około 1.9V (**4095 do 2400** dla surowych odczytów z 12-bitowego konwertera analogowo-cyfrowego ADC).

Pomimo zbudowania układu dającego wyniki pozwalające na oszacowanie wilgotności gleby, dokonano dalszego przeglądu literatury i artykułów naukowych. Zaowocowało to znalezieniem rodziny układów ESP32, w szczególności ESP32-WROOM-32E, opisanego w podrozdziale 1.3. Platforma ta posiadała układ peryferyjny ze zintegrowanym oscylatorem kwarcowym i umożliwiała generowania impulsów prostokątnych z częstotliwością do 40MHz i natężeniem do 20mA. Płytkę uruchomieniową z ESP32 (rys. 9) zaprogramowano do generowania impulsów prostokątnych o f=40MHz i wypełnieniu 50% oraz pomiaru napięcia na jednym z GPIO. Przy wykorzystaniu płytki stykowej oraz 3 żył kabla UTP kat. 5e do wyprowadzeń ESP32 podłączono płytkę PCB (wersja B) opartą na schemacie elektrycznym z rysunku 45. Zakres pomiarowy woda-powietrze wyniósł 3.3V – 1.5V (**4095 do 1500** dla surowych odczytów z 12-bitowego konwertera ADC). Wynik ten został jeszcze poprawiony do wartości ok. 0.9V (**1050)** poprzez stworzenie dedykowanej płytki PCB, zastosowania połączeń lutowanych oraz kabla zewnętrznego żelowanego UTP kat.5e. Zaprojektowano nowy, pasywny układ wykonawczy którego schemat elektryczny i wygląd PCB przedstawiono na poniższych rysunkach 51 i 52.

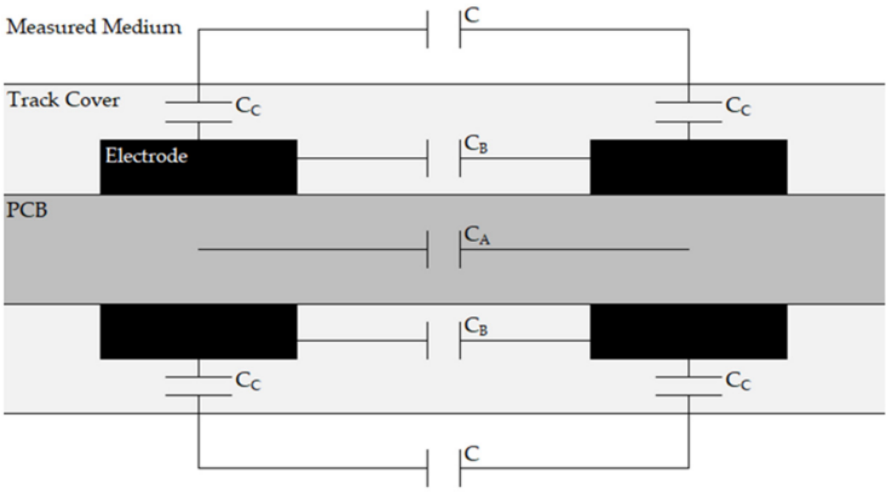


Rysunek 51: Schemat elektryczny finalnego układu wykonawczego czujnika wilgotności



Rysunek 52: Widok PCB finalnego układu wykonawczego czujnika wilgotności (płytka dwustronna)

Badaniom finalnej wersji układu wykonawczego poświęcony jest cały 4 rozdział pracy. Warty podkreślenia jest fakt, że kondensator C1-SENSOR z rys. 51, może zostać zamodelowany tak jak na rysunku nr 53 wraz z dodatkową, niezaznaczoną tam pojemnością medium transmisyjnego (kabla UTP), dla ustalenia oznaczoną jako C_{UTP}. Jest to więc układ równolegle połączonych pojemności C_A, $2*C_B$, $2*(2*C_C+C)$ i właśnie C_{UTP}. Pojemność skuteczna podawana przez producenta przewodów w odniesieniu do pary żył dla 1kHz wynosi około 50nF/km[108], co w przeliczeniu na 0,5m (jakie zastosowano) daje wartość 25pF (w rzeczywistości będzie to jeszcze mniejsza wartość ze względu na częstotliwość sygnału f=40MHz).



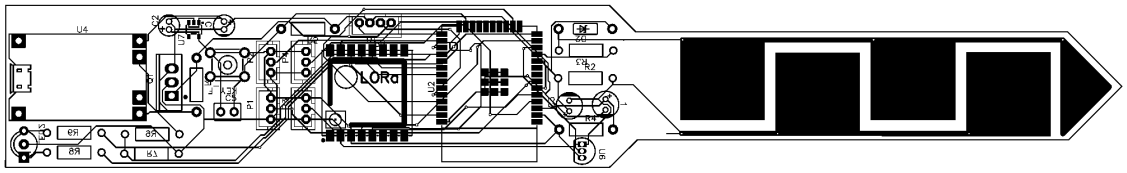
Rysunek 53: Model elementu wykonawczego w ujęciu pojemności elektrycznej[102]

Pojemność C jest zmienna i wprostproporcjonalnie związana z przenikalnością dielektryczną ośrodka w jakim umieszczony jest układ. Pozostałe pojemności można uznać za stałe – zależą one od materiałów płytki PCB oraz maski lutowniczej. Ostatecznie, oznaczając symbolem C_x wartości stałe:

$$C_{1SENSOR} = C_A + 2 \cdot C_B + 2 \cdot (C + 2 \cdot C_C) + C_{UTP} = 2 \cdot C + C_x \qquad (13)$$

Materiałem z których zostaną wykonane płytki PCB będzie laminat epoksydowy wzmacniany włóknem szklanym, którego przenikalność dielektryczna ε_r w zależności od składu mieści się w przedziale <3.8;4.7>[109] co ma wpływ na pojemność C_A. Wartość ε_r dla maski lutowniczej może zostać przyjęta za 3.4 (wpływ na pojemności C_B i C_C). Z kolei dla suchej gleby wartość ε_r =4, a dla wody ε_r =78.54 w temperaturze 25°C (maleje wraz ze wzrostem temperatury)[110]. Wartości te pozwalają w przybliżeniu oszacować, że w sytuacji niewielkiej zawartości wody w glebie i osiąganiu przenikalności dielektrycznej tej mieszaniny wartości poniżej 5, pojemność płytki i maski lutowniczej mogą przeważać i będą czyniły wyniki pomiarów nieużytecznymi.
Natomiast w celu uproszczenia i ograniczenia konieczności kalibracji elementów wykonawczych przyjęto ich podłączanie na kablu o jednakowej, ustalonej długości 50cm.

Na koniec warto zauważyć, że optymalną ze względu projektowych (minimalizacja C_{UTP}) oraz eksploatacyjnych (brak przewodów) byłaby wersja w której element wykonawczy i układ pobudzający znajdują się na jednej płytce PCB. Taka została również zaprojektowana i może w przyszłości posłużyć do rozbudowy funkcjonalności systemu o pomiar wilgotności w płytkich pojemnikach lub przerodzić się w osobny projekt wykorzystywany w domu, ze względu na obecność modułu radiowego WiFi w ESP32 (rys. nr 54).



Rysunek 54: Projekt PCB doniczkowego czujnika wilgotności

3.5.2 Bezprzewodowy czujnik wilgotności i temperatury gleby oraz powierza

Wymaganie C.2 obligujące do zasilania bateryjnego narzuciło w trakcie projektowania spojrzenie na każdy komponent i jego pracę przez pryzmat oszczędnego gospodarowania energią. W konsekwencji spowodowało to:

a) wybór komponentów zgodnych z logiką zasilania 3.3V,
b) uwzględnienie wykorzystania w oprogramowaniu układowym trybu głębokiego uśpienia ESP32 (tabela nr 3) z zachowaniem danych w pamięci RTC i możliwością wybudzenia zdarzeniem zewnętrznym (wymaganie C.3.b) przez RTC GPIO (doprowadzonych do ko-procesora Ultra Low Power),
c) odcięciem zasilania od wszystkich modułów pomiarowych i układu radiowego LoRa w trakcie uśpienia poprzez zastosowanie tranzystora MOSFET sterowanego dedykowaną linią ESP32, podnoszoną w stan aktywny tylko podczas pomiarów,
d) wybór tranzystora MOSFET (IRF3708PBF) z stosunkowo niską wartością oporu wewnętrznego R_{DS}=12mΩ oraz napięciem progowym max. V_{gs} poniżej 3.3V
e) wybór pojemnego akumulatora litowo-jonowy typu 18650,

f) wybór ładowarki Li-Pol TP4056, umożliwiającej ładowanie akumulatorów przez złącze microUSB lub zewnętrzne źródło zasilania z napieciem od 4.5V do 5.5V – np. poprzez ogniwo słoneczne,

g) wybór liniowego, nieregulowanego stabilizatora napięcia (AP7365-33WG-7) o bardzo niskim prądzie spoczynkowym I_Q=35μA.

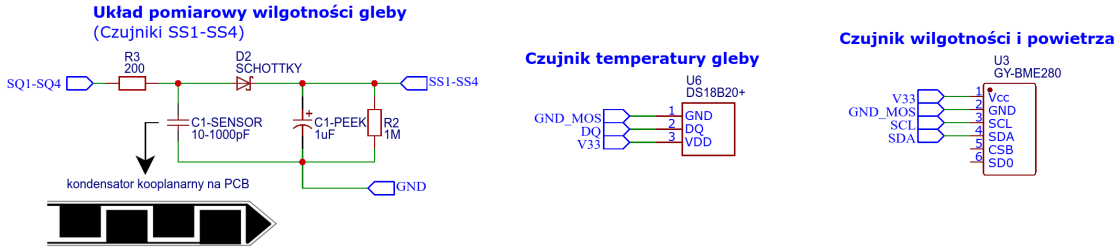
Schemat elektryczny strefowego czujnika wilgotności został przedstawiony na rys 57 i 55. Wspomniany stabilizator napięcia został wybrany również dzięki maksymalnemu prądowi w wielkości 600mA jaki jest w stanie dostarczać. Jest to wartość dająca ponad 33% zapas w sytuacji jednoczesnej pracy obu modułów radiowych WiFi z ESP32 oraz LoRa z RA-02 SX1278. Zgodnie ze specyfikacją wymagań taka sytuacja nie będzie miała miejsca (moduł WiFi nie jest wykorzystywany), jednakże pozwala to w przyszłości rozwijać nowe funkcje u rządzenia na tej samej platformie sprzętowej. W celu zapewnienia stabilnej pracy, nawet w trakcie zwiększonego zapotrzebowania na energię, przy modułach ESP32 oraz LoRa umieszczono kondensatory stabilizujące napięcie – C7 i C6 na schemacie elektrycznym z rys. 57.

Dołączane do projektowanego urządzenia układy peryferyjne łączą się z nim poprzez gniazda typu JST w wersji PH o rozstawie pinów 2mm. W ten sposób umożliwia się podłączenie czujników wilgotności gleby (gniazda P1, P2, P3, P4), termometru cyfrowego (P-SOIL-T1) oraz czujnika parametrów powietrza (P-AIR-HT1). Daje to możliwość ich prostej wymiany, nawet przez użytkownika. Do celów serwisowych, diagnostyki lub aktualizacji oprogramowania układowego został przeznaczony port P-SERW1 do którego wyprowadzone linie RX, TX, GND oraz BOOT (zwarcie tych ostatnich podczas uruchamiania przełącza ESP32 w tryb programowania). Urządzenie zostało podzielone na dwa moduły rozmieszczone na osobnych płytkach : główny oraz zasilający, co przedstawiono na rysunkach 58 i 59. Łączące je linia zasilająca wpinana jest do gniazd JST typu XH o rozstawie 2,54mm. Zastosowany większy rozmiar złącza, ze względu ja jego możliwą częstszą eksploatację, w przypadku wyjmowania z akumulatorem w celu ładowania.

Komunikacja pomiędzy układem ESP32 a czujnikiem wilgotności i temperatury powietrza BME280 oparta będzie na magistrali I^2C, natomiast termometru cyfrowego DS18B20+ umieszczonego w glebie przez interfejs 1-Wire. Do połączenia modułu LoRa wykorzystano zaś obsługiwany przez niego interfejs SPI (oznaczono jako MISO, MOSI, SCK, RST i NSS) oraz dodatkowe wyjście GPIO tego układu DIO0, w celu transmisji sygnału o nowej przychodzącej wiadomości. Pozwoli to w oprogramowaniu głównego modułu bezzwłocznie obsługiwać transmisje przychodzące ze sterownika nawadniania, ponieważ jego interfejs GPIO15 obsługuje przerwania.

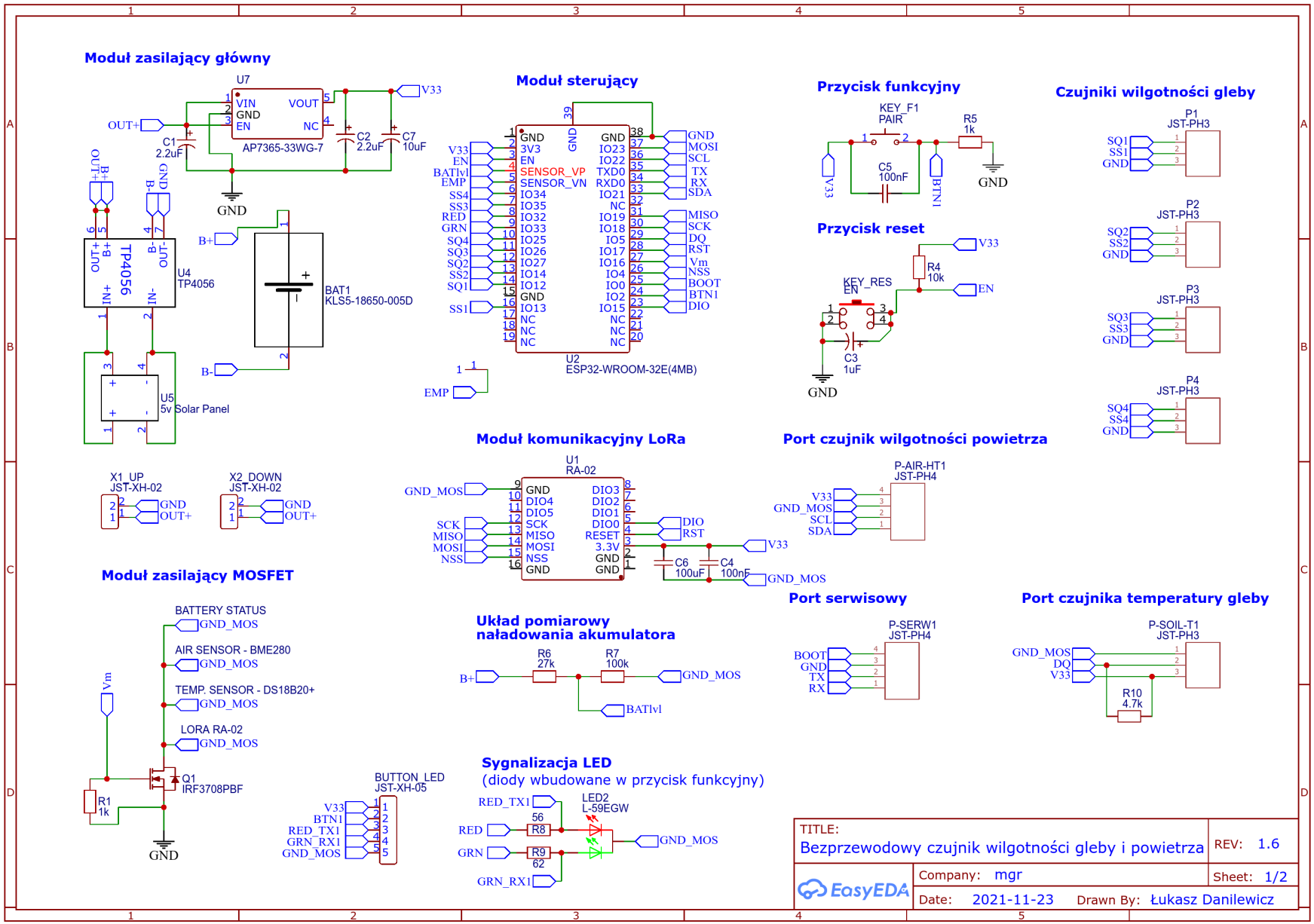
Zważając na wymaganie C.1.a o stopniu ochrony na poziomie min. IP 54, znaleziono rozwiązanie zapewniające spełnienie C.3.a dotyczące interfejsu użytkownika. Stał się nim przełącznik chwilowy z podświetleniem LED w kolorach czerwonym i zielonym, z rysunku 56.
Dostosowanie napięcia zasilającego 3.3V dla każdej z diod zostało osiągnięte przez indywidualnie dobrane rezystory R8 i R9, zaś sam przycisk podłączony jest przez 5-pinowe

gniazdo BUTTON_LED (typu JST-XH). Trzeci wymagany kolor – pomarańczowy, osiągany jest przez jednoczesne zasilenie diod.



Rysunek 55: Schemat elektryczny strefowego czujnika wilgotności gleby cz. 1/2 – moduły pomiarowe

Rysunek 56: Przycisk 16mm z podświetleniem czerwonym i zielonym



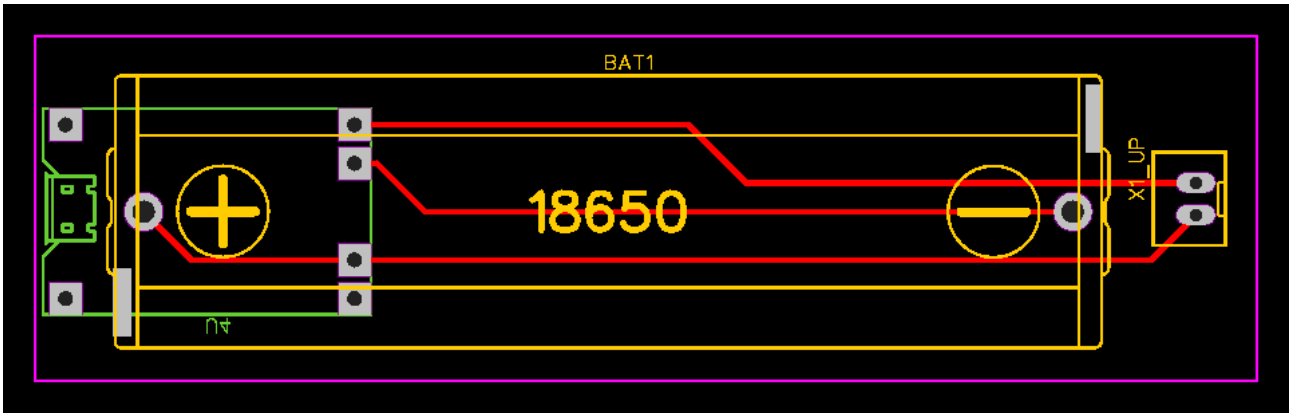
Rysunek 57: Schemat elektryczny strefowego czujnika wilgotności gleby cz. 2/2

Drugim, niedostępnym z poziomu obudowy przyciskiem jest mikro-przełącznik do trwałego umieszczenia w miejscu KEY_RES (był on bardzo przydatny w trakcie rozwoju oprogramowania).

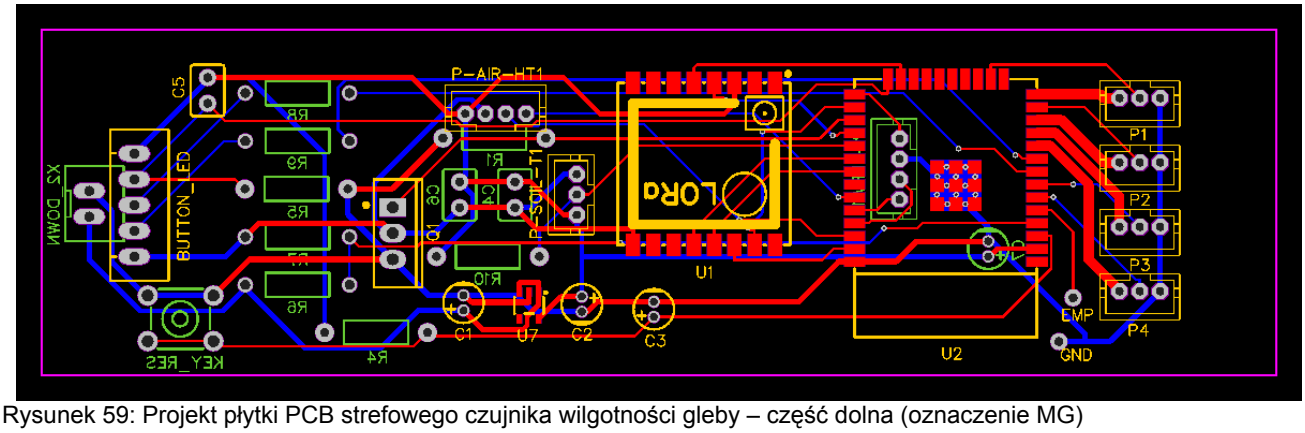
Przybliżony pomiar stanu naładowania baterii (C.1f.) został zaprojektowany za pomocą dzielnika napięcia na rezystorach R6 i R7. W pełni naładowany akumulator posiada napięcie 4.2V, które po odłożeniu na dzielniku uzyskuje wartość 3.3V, która może zostać bezpiecznie zmierzona przez ESP32. Za 100% naładowania przyjęto więc 4.2V, a 0% dla 3.3V (którym nominalnie powinny być zasilany układ główny oraz LoRa). Zastosowane wartości oporników dają maksymalny prąd rzędu 33µA.

Z dostępnych w trybie high-speed 8 kanałów w module LEDC układu ESP32 wykorzystano 4 i przypisano do GPIO głównego układu SQ1, SQ2, SQ3 i SQ4 (GPIO 12, 27, 26 i 25). Dzięki temu możliwe jest więc podłączenie, aż 4 elementów wykonawczych omawianych w poprzednim podrozdziale. Takie działanie może posłużyć do pomiarów na zbliżonej głębokości i uśrednienia wyniku. Jednakże ze względów konstrukcyjnych i potrzeby wyprowadzenia przewodów 4 przewodów z obudowy w głąb gleby (w tym jeden dla termometru cyfrowego) do wykorzystania pozostają 3 sztuki. Odczyt wyniku pomiaru układów wykonawczych czujnika wilgotności (rys. 52) w postaci pomiaru napięcia odbywa się na liniach SS1, SS2, SS3 i SS4.

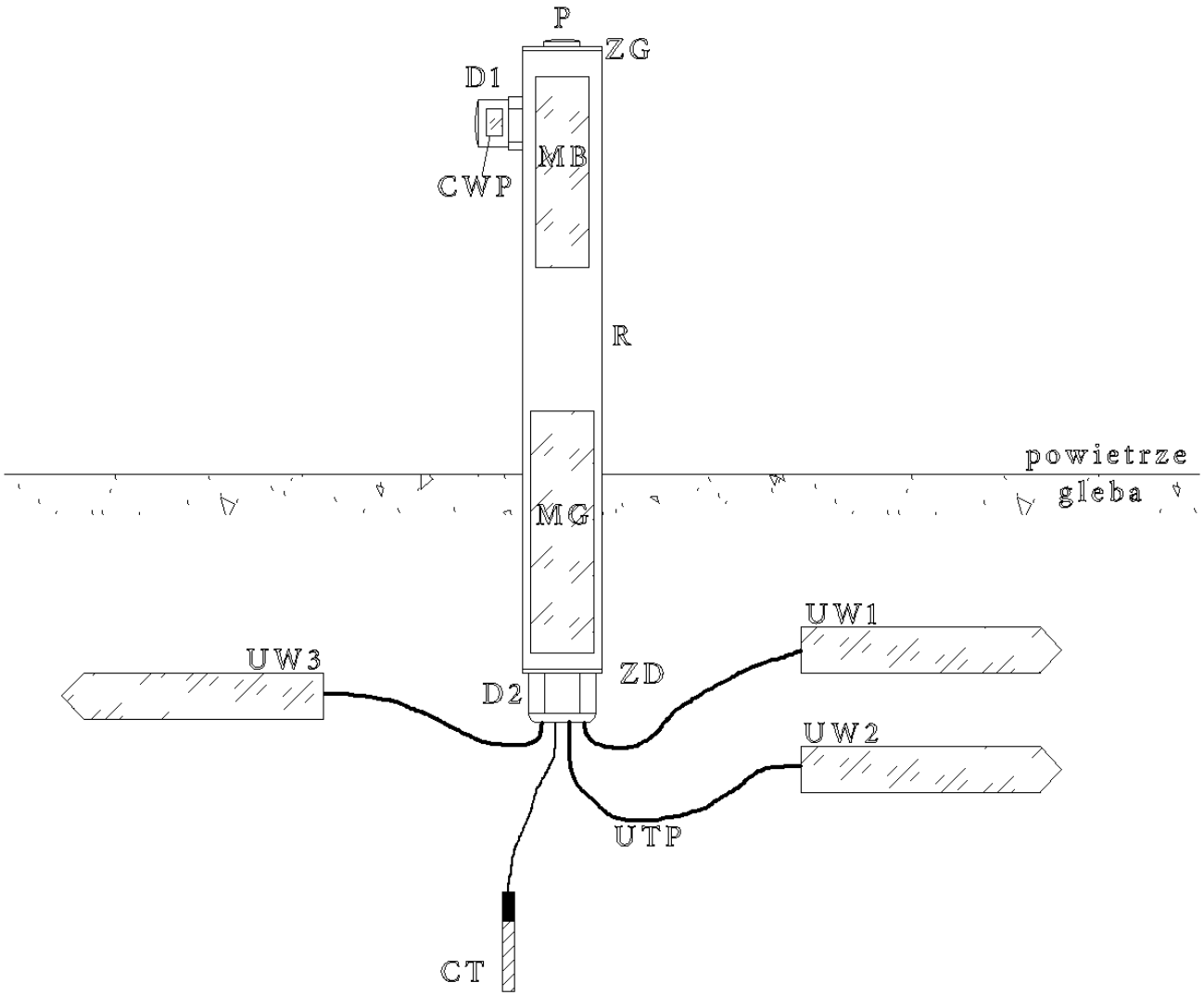
Na kolejnym rysunku nr 60 przedstawiono budowę strefowego czujnika wilgotności wraz z opisem zastosowanych oznaczeń. Ostatecznie w warstwie sprzętowej spełnione zostały wszystkie wymagania ze specyfikacji „A. System nawadniania” i „C. Czujnik strefowy”, uwzględniając przy tym potrzeby funkcjonalności dostarczanych przez oprogramowanie.



Rysunek 58: Projekt płytki PCB strefowego czujnika wilgotności gleby – moduł akumulatora z ładowarką (ozn. MB)



Rysunek 59: Projekt płytki PCB strefowego czujnika wilgotności gleby – część dolna (oznaczenie MG)



Rysunek 60: Architektura strefowego czujnika wilgotności

Legenda:

R – rura kanalizacyjna biała, średnica 40 mm, długość 315 mm

MB – wyjmowany moduł zasilania (ładowarka + akumulator) (z rys. 58)

MG – moduł główny z układem komunikacyjnym (z rys. 59)

D1 – rozszerzenie do dławicy M16/M20

D2 – dławnica kablowa poliamidowa M25 z wkładem uszczelniającym na 4przewody x 5mm

CT – cyfrowy czujnik temperatury

UW1, UW2, UW3 – autorskie moduły wykonawcze czujników wilgotności gleby

CWP – czujnik wilgotności, temperatury i ciśnienia powietrza

P – metalowy przycisk monostabilny z diodą led (czerwony, zielony) (z rys. 56)

UTP – kabel do ziemi UTP, z żelem hydrofobowym (8żył, 3 używane)

ZG – zaślepka 40mm wraz z uszczelką oring

ZD – korek zaślepiający do rury 40 mm (uszczelka w kielichu rury + dodatkowy oring na korku)

3.5.3 Sterownik nawadniania

Schemat elektryczny sterownika nawadniania oraz jego modułów wykonawczych umieszczono na rys. nr 62 i 63. Celem obsługi wyspecyfikowanych w tabeli nr 20 modułów (podrozdział 3.6) , na płytce PCB zaprojektowano linie zasilające z wartościami docelowo przenoszonych napięć 24V AC, 5V DC oraz 3.3V DC. Prąd przemienny z zacisków złącza Z1, podawany jest na prostownik jedno-połówkowy[111] stworzony z diody prostowniczej 1N4007 (1A/1000V) oraz kondensatora wygładzającego (100µF/63V). Następnie trafia na wejście modułu przetwornicy step-down, której wyjście zostało poprzez pokrętło regulacyjne ustawione na napięcie 5V. Te z kolei jest podawane na wejście regulatora napięcia AP7365-33WG-7(model taki jak w czujniku wilgotności). Działanie znakomitej większości układów odbywa się przy zasilaniu i w logice 3.3V. Wyjątek stanowi wyświetlacz LCD, czujnik ruchu PIR oraz czujnik przepływu cieczy, wymagające napięcia zasilającego na poziomie 5V. W przypadku tego ostatniego zastosowano konwerter poziomów logicznych, ażeby napięcie wynikowego sygnału prostokątnego obniżyć do 3.3V w celu pomiaru jego częstotliwości przez ESP32 (linia WTR_F, GPIO 35). Poza tym prąd przemienny 24V jest doprowadzony do układów wykonawczych w celu zasilenia elektrozaworów systemu nawadniania.

Układy wykonawcze oparte zostały o optotriak MOC3023, zapewniający galwaniczną separację układu sterującego i sterowanego. Przy wyborze kierowano się m. in. jak najmniejszą wartością prądu wyzwalania wewnętrznej diody optotriaka, która w tym modelu wynosi tylko 5mA. Za strukturę układu przełączanego przez optotriak wykorzystano przykładową aplikację z karty katalogowej[112], z tą różnicą, że zamiast obwodu gasikowego do ochrony tranzystora sterującego zastosowano diodę TVS (z parametrem V_C=146V - *maximum clamping voltag*e), równoległe połączoną z dołączanymi elektrozaworami. Dioda włączona w ten sposób tłumi przepięcia powstające podczas przełączeń przekaźnika mogących uszkodzić tranzystor sterujący BT1308.

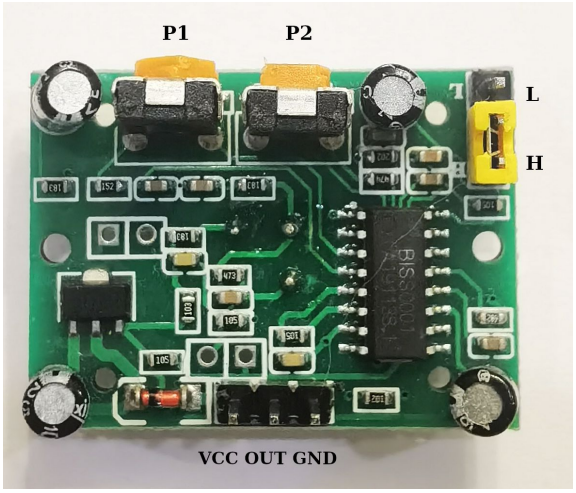
Wypełnienie wymagania B.9.a zrealizowano przez wykorzystanie modułu zegara precyzyjnego RTC DS3231 (operującego na magistrali IIC), do podtrzymania daty i czasu w przypadku utraty zasilania. Dodatkowo jego wyjście SQW zostało przeznaczone do skonfigurowania, aby zmieniać swój stan raz na minutę, celem umożliwienia odświeżania daty i godziny na ekranie.

W sterowniku interfejs SPI jest współdzielony przez dwa moduły: SX1278 LoRa oraz czytnika kart pamięci. Do aktywacji wybranego z nich, przez ustawienie stanu niskiego, doprowadzono odrębne linie pełniące linię *chip select*: NSS (GPIO4) oraz CS(GPIO5). Podobnie jak w czujniku wilgotności wykorzystano linię GPIO DIO0 (układu SX1278) do przekazania stanu, w którym otrzymywana jest nowa wiadomość przez protokół LoRa.

W celu spełnienia wymagania B.1.c dotyczącego ilości obsługiwanych stref wykorzystano ekspander wyprowadzeń MCP23017 sterowany przez magistralę I2C, ustawiając jego 3 bitowy adres na logiczne 0 (wyprowadzenia A1, A2, A3). Osiem (Z01...Z08) z jego 16

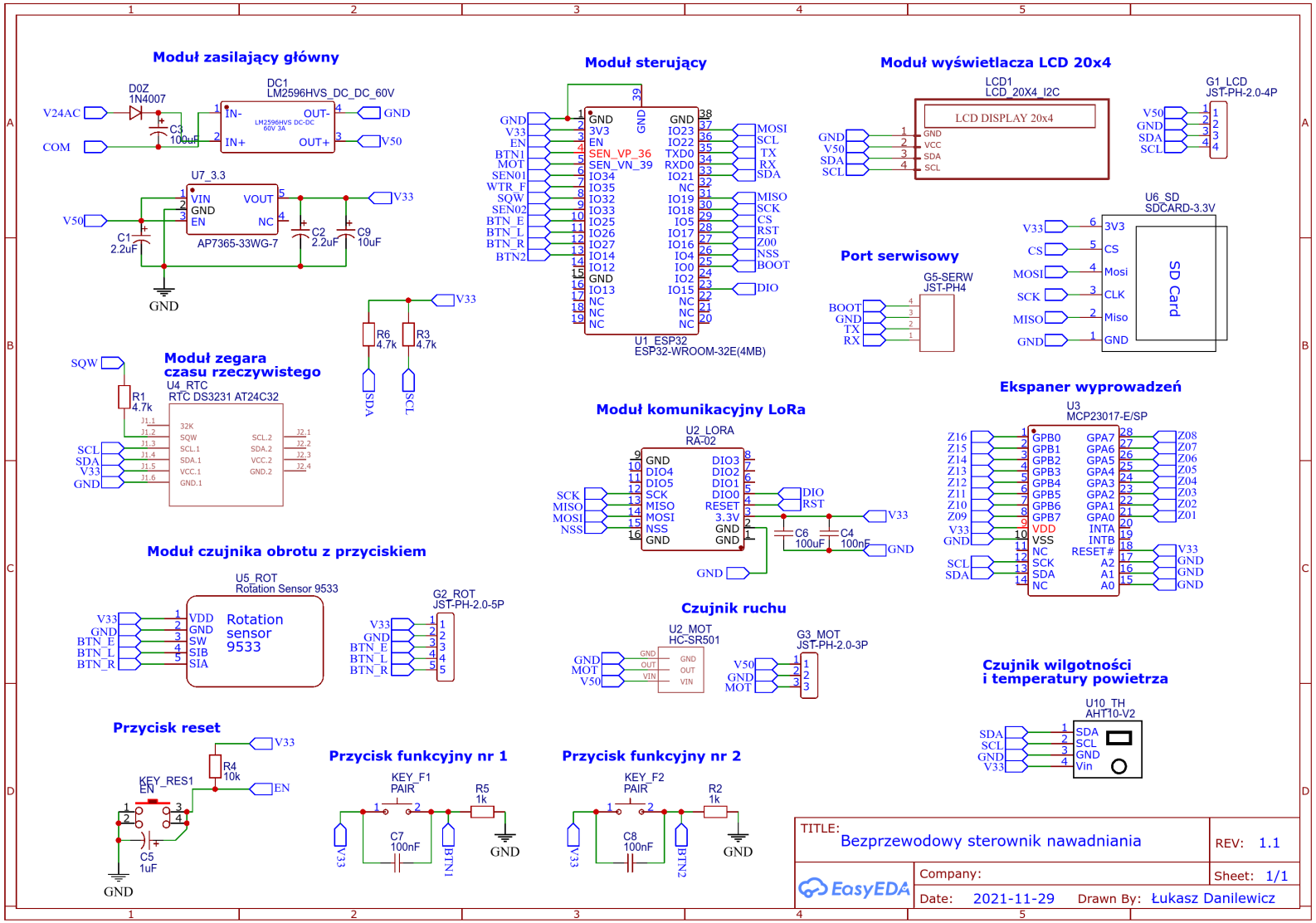
wyprowadzeń zostało poprowadzonych do modułów wykonawczych na płytce PCB, pozostałe zaś (Z09...Z16) w pobliżu jej krawędzi do 10-pinowego złącza JST-PH. Tym samym stworzono port za pomocą którego można będzie, po dołączeniu dodatkowego modułu wykonawczego(będącego poza zakresem tego opracowania), rozszerzyć ilość stref z 8 do 16. Jednakże schemat elektryczny tego urządzenia będzie podobny do schematu układów wykonawczych z rys. nr 63. Stworzono również strefę Z00 (dedykowana linia sterująca z GPIO16 ESP32 wraz z układem wykonawczym), aby obsłużyć główny elektrozawór systemu nawadniania.

Wypełnienie wymagań B.4 (czasowe podświetlenie ekranu) i B.11 (alarm) może być zrealizowane przez wybrany czujnik ruchu PIR HC-SR501 (rys nr. 61). Jego tryb pracy został wybrany na *non-retriggering*, przez odpowiednie ustawienie zworki. Powoduje to, że wyjście osiąga stan wysoki tylko raz po wykryciu obiektu i przechodzi w stan niski po czasie ustawionym za pomocą potencjometru P1. Poprzez jego regulację do wartości około 60s, stan wyjścia OUT można będzie wykorzystać do uruchamiania podświetlenia ekranu. Potencjometr regulujący czułość wykrywania ruchu P2 został ustawiony w połowie zakresu. Tak skonfigurowany czujnik przeznaczono również do obsługi prostego alarmu, przy wykorzystaniu jego wyjścia OUT wraz z stanami rozbrojony i zazbrojony do implementacji w oprogramowaniu. Natomiast dane z moduł AHT10 na temat wilgotności temperatury mają posłużyć do spełnienia wymagania B.5.

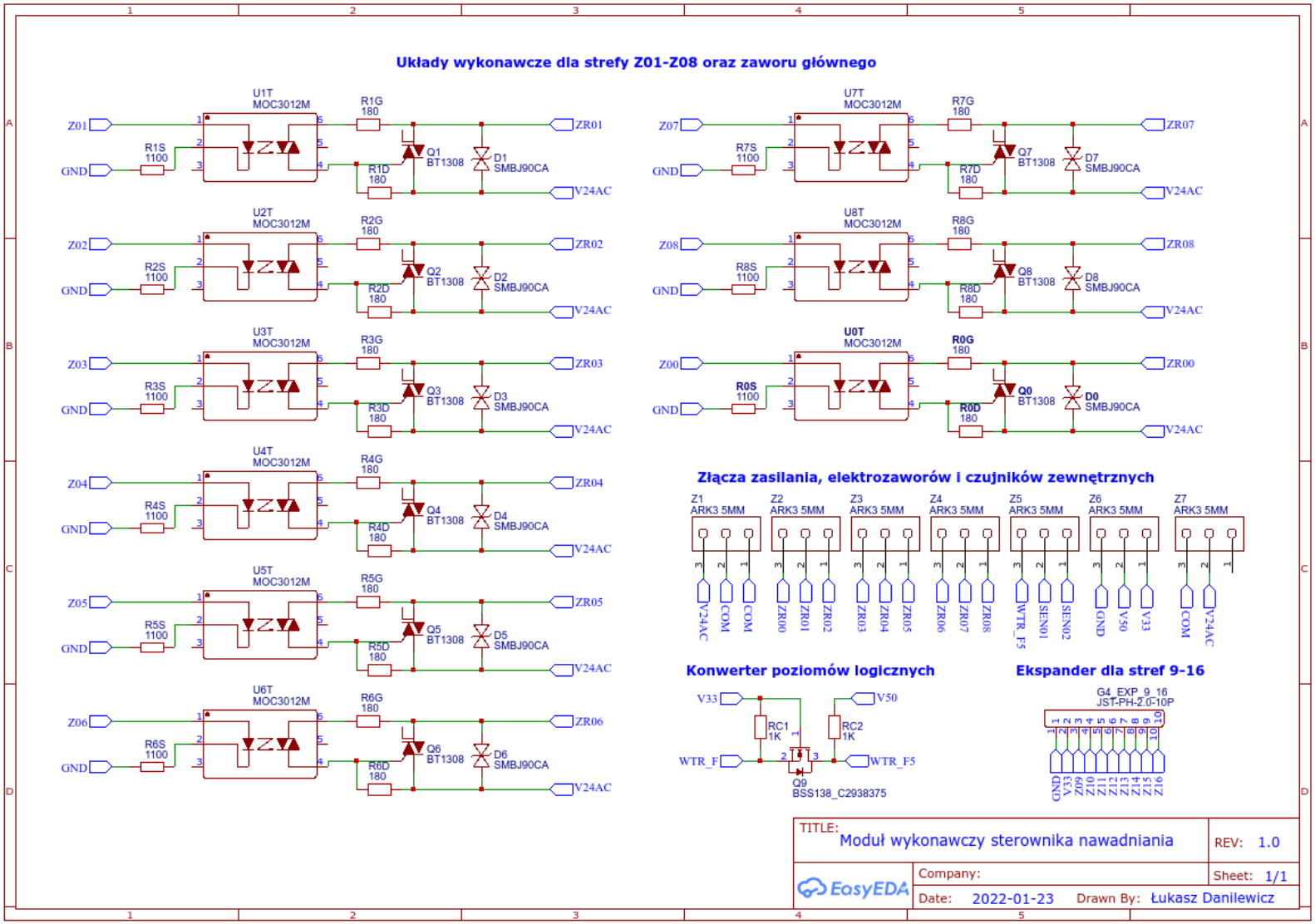


Rysunek 61: Moduł czujnika ruchu PIR HC-SR501

Interfejs użytkownika, poza wspomnianym ekranem LCD będą stanowić: a) dwa monostabilne przyciski funkcyjne KEY_F1 oraz KEY_F2, b) monostabilny przycisk KEY_RES1, c) enkoder obrotu z przyciskiem U5_ROT. Celem eliminacji drgań styków w trakcie przełączania zastosowano kondensatory C7, C8 C5. Wyszczególnione komponenty pozwolą spełnić wymaganie B.5 dotyczące manualnego sterowania strefami nawadniania. Miejsce na drugi przycisk funkcyjny przeznaczono w celu do wykorzystania w przyszłości, gdyż w rzeczywistości nie znajdzie on zastosowania w pierwszej wersji oprogramowania.



Rysunek 62: Schemat elektryczny sterownika nawadniania



Rysunek 63: Schemat elektryczny modułów wykonawczych sterownika nawadniania

Dodatkowo warte nadmienia jest, że przestrzegane były zalecenia projektowe z dokumentacji dotyczące modułu głównego ESP32 jak np. rozmieszczenie anteny oraz szerokość ścieżek linii zasilającej. W tabeli 20 (podrozdział 3.6) zebrane zostały wszystkie komponenty i moduły wraz z wyszczególnieniem wykorzystanej magistrali danych.

3.6 Wyszczególnienie komponentów systemu oraz modułów elektronicznych

W pracach projektowych niezwykle przydatne stały się szybko i w jednym miejscu dostępne informacje na temat użytych modułów elektronicznych. Najistotniejsze z nich w tabelarycznej formie zostały przedstawione w tabelach nr 20 i 21, uwzględniając nazwę, krótki opis i przeznaczenie, wymiary, napięcie zasilające, maksymalny pobierany lub dopuszczalny prąd, wykorzystywaną magistralę danych, oznaczenie PCB oraz numery wybrane dla nich numery złącz GPIO. Pełna tabela zawierała dodatkowo link do sklepu, cenę za sztukę oraz odnośnik do karty katalogowej.

W celu uzyskania ogólnego oglądu na składniki systemu nawadniania stworzono tabelę nr 19, w której wymienione zostały role pełnione przez poszczególne elementy systemu wraz w wyszczególnieniem głównego komponentu oraz wymienieniem zakresu prac. Do zestawienia tego został również włączony elektrozawór, który zostanie wykorzystany podczas realizacji scenariuszy testowych w warunkach rzeczywistych.

Tabela 19: Zestawienie ról pełnionych przez poszczególne komponenty systemu

Lp.	Rola	Główny komponent	Zakres prac
1.	Sterownik	Układ ESP32-WROOM-32E	Projekt, budowa, implementacja
2.	Elektrozawór	Hunter PGV 100/101MMB 1" GZ x GZ	Integracja
3.	Brama sieciowa, ruter, zapora	Ruter Mikrotik RBwAPR-2nD&R11e-LTE	Konfiguracja
4.	Aplikacja do zdalnego sterowania	Urządzenie przenośne z systemem operacyjnym Android 5+	Projekt, implementacja
5.	Strefowy czujnik wilgotności i temperatury	Układ ESP32-WROOM-32E	Projekt, badania, budowa, implementacja
6.	Układ eksperymentalny do badani układu wykonawczego czujnika z p. 5	Platforma Espressif ESP32-WROOM-32E	Budowa, implementacja

Tabela 20: Moduły i części elektroniczne sterownika

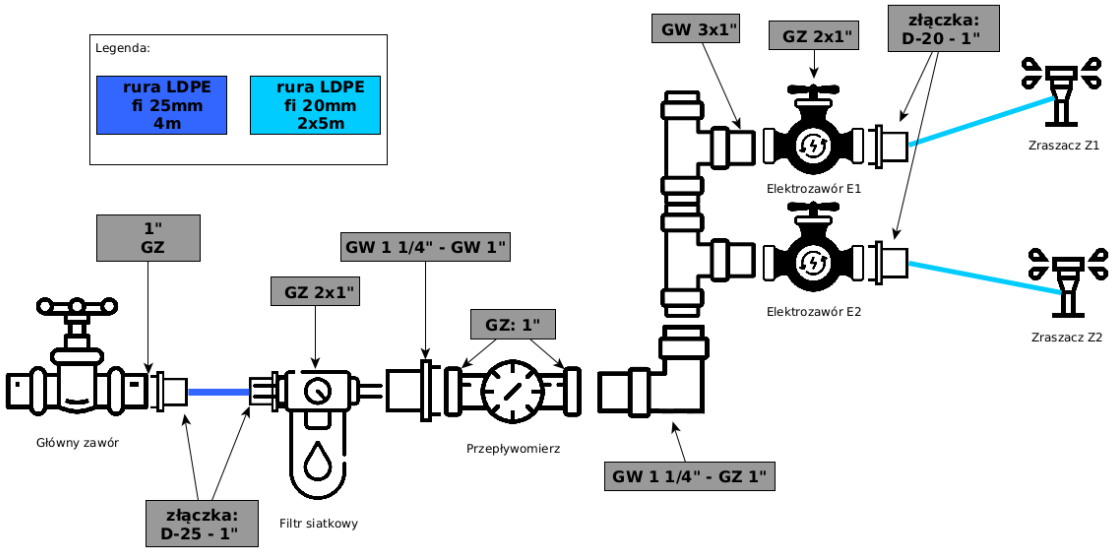
LP.	Nazwa	Opis, przeznaczenie	Wymiary [d x s x w]	Zasilanie [V]	Prąd max [mA]	Oznacz. PCB	Magistrala danych	PIN ESP32
1	ESP32-WROOM-32E 4MB	Układ główny	18.0 × 25.5 × 3.1mm	3.0V-3.6V	239mA	U1_ESP32	-	-
2	Wyświetlacz LCD 4x20 znaków	Wyświetlanie stanu stref, daty i godziny i innych parametrów	9.8 x 6.0 x 1.2 cm	5V	120 mA	port: G1_LCD	I2C	SDA-21, SCL-22
3	Czujnik ruchu PIR HC-SR501	Wykrywanie ruchu: alarm, wzbudzenia wyświetlacza	32.5 x 24.5 x 30mm	4.5-20V	65 mA	port: G3_MOT	Cyfr.,1 pin	39
4	Moduł pomiaru temperatury i wilgotności AHT10	Pomiar temperatury, wilgotności i ciśnienia w pomieszczeniu	16mm x 11mm	1.8V-5.0 V	1mA	U10_TH	I2C	SDA-21, SCL-22
5	Zegar precyzyjny DS3231 z modułem AT24C32	Zachowanie czasu po utracie zasilania, pamięć 32kB	38mm x 22.5mm	3.3-5V	0,65 mA	U4_RTC	I2C	SDA-21, SCL-22
6	SX1278 LoRa Module 433M 10KM Ra-02 Ai-Thinker	Komunikacja bezprzewodowa między czujnikami i sterownikiem	30mm x25mm	1.8-3.7V	10,8 /120mA	U2_LORA	Cyfr.,6 pin (SPI)	23,19,18, CS-5
7	Moduł kart pamięci microSD	Czytnik kart pamięci	18mm x 18mm	3.3-5.5V	80mA	U6_SD	Cyfr.,6 pin (SPI)	23,19,18, CS-5
8	MCP23017	ekspander wyprowadzeń I2C 16-kanałowy	DIP28	1.8V-5.5V	125mA	U3	I2C	SDA-21, SCL-22
9	Waveshare 9533	Czujnik obrotu, impulsator, enkoder z przyciskiem	32 x 15 mm	3.0V-5.3V	b. d.	port: G2_ROT	Cyfr.,3 pin	25,26,27
10	Moduł przetwornicy step-down LM2596HVS	Regulacja napięcia do 5V	20.2 x 43.8 x 14.2 mm	4.5-50V	3A	DC1	-	-
11	Regulator napięcia LDO AP7365-33WG-7	Regulacja napięcia do 3.3V	SOT-25-5	2V-6V	600mA	U7_3.3	-	-
12	Optotriak MOC3023	Sterowanie napięciem zmiennym 24V AC przy pomocy logiki 3.3V	DIP6	500V	5mA	U0T,U1T… U8T	-	-
13	Tranzystor BSS138	Konwersja poziomu logicznego 5V do 3.3V do obsługi przepływomierza	SOT23	3.3V	-	Q9	-	35
14	Przycisk (2szt)	Zmiana aktualnej funkcji zegara	12 x 12mm	3-5V	5mA	KEY_F1, KEY_F2	Cyfr.,1 pin	14,36
15	433 Mhz antena LORA 5dbi GSM	antena LoRa	b. d.	-	-	-	nie dotyczy	-
16	Czujnik przepływu cieczy YF-B10	Oszacowanie il. cieczy przepływającej przez główną magistralę	b. d.	5V-18V	15mA	WTR_F	Analog.,1 pin	35 (przez BSS138)
17	Dioda SMBJ90CA	Zabezpieczenie triaka BT1308	DO214AA	-	4.1A	D0,D1...D8	-	-
18	Triak BT1308	Zasilanie elektrozaworów	SOT223	-	800mA	Q0,Q1...Q8	-	-

Tabela 21: Moduły i części elektroniczne bezprzewodowego czujnika strefowego

LP.	Nazwa	Opis, przeznaczenie	**Wymiary [d x s x w]**	**Zasilanie [V]**	Prąd max [mA]	Oznacz. PCB	Magistrala danych	PIN ESP32
1	ESP32-WROOM-32E 4MB	Moduł sterujący	18.0 × 25.5 × 3.1mm	3.0V-3.6V	239mA	U2_LCD	-	-
2	Ładowarka Li-Ion Li-Poly TP4056	Zasilacz buforowy	17.4mm x 22.4mm	4.5-5.5 V	b. d.	U4	-	-
3	DS18B20 (sonda)	Cyfrowy czujnik temperatury	50mm x 6mm	3.0V-5.5V	1.5mA	port: P-SOIL-T1	Cyfr.,1 pin	5
4	Moduł LoRa SX1278 433M 10KM Ra-02 Ai-Thinker	Komunikacja bezprzewodowa między czujnikami i sterownikiem	30mm x25mm	1.8-3.7V	120mA	U1	Cyfr.,6 pin (SPI)	MOSI-23, MISO-19, SCLK-18, CS-5
5	Tact Switch 6x6mm	Mikro-przełącznik do resetu	6x6x8mm	max: 12V	50mA	KEY_RES	Analog.,1 pin	EN
6	Przełącznik chwilowy z podświetleniem	Sygnalizacja statusu urządzenia, budzenie ze stanu uśpienia	D=17.8mm, H=35mm	3.0V-5.0V	25mA	KEY_F1	Analog.,3 pin	2, 33, 32
7	Tranzystor MOSFET–N IRF3708PBF	MOSFET – N	TO-220-3	Vds = 30V Vgs = 12V	62A	Q1	Analog.,1 pin	16
8	Czujnik temp., wilgotności i ciśnienia BME280	Pomiar temperatury, wilgotności i ciśnienia	10mm x 15 mm	3.3V	1mA	port: P-AIR-HT1	I2C	SDA-21, SCL-22
9	Autorski czujnik wilgotności gleby (4szt)	Pomiar wilgotności gleby	23mm x 133mm	3.3V	10mA	porty: P1, P2, P3, P4	Cyfr.,1 pin; Analog., 1pin	(13,12), (14,27), (26,35), (25,34)
10	Regulator napięcia LDO AP7365-33WG-7	Regulacja napięcia do 3.3V	SOT-25-5	2V-6V	600mA	U7	-	3V3
11	Koszyk na akumulator 18650	Mocowanie akumulatora do PCB	77 x 20 x 20 mm	-	-	BAT1	-	-
12	Ogniwo 18650 Li-Ion Samsung INR18650-30Q 3000mAh	Akumulator	65mm x 18.3mm	2.5V-4.2V	4A	-	-	-
13	Antena sprężynowa 433MHz 3dBi IPEX U. FL	Antena do czujnika	-	-	-	-	-	-

3.7 Stworzenie scenariuszy testowych i koncepcji środowiska do testowania

Zostały zdefiniowane przedstawione poniżej scenariusze testowe S1, S2, S3 i S4, których celem jest weryfikacja działania tworzonego systemu w stosunku do specyfikacji wymagań. Scenariusz S1 ma zostać przeprowadzony w warunkach rzeczywistych, w skonstruowanej do tego celu instalacji nawodnieniowej, wykonanej zgodnie z rys. 64.



Rysunek 64: Schemat instalacji nawodnieniowej do scenariusza testowego S1
GW – gwint wewnętrzny, GZ – gwint zewnętrzny

Scenariusz S1 – próba w warunkach rzeczywistych.

S1. Budowa instalacji nawodnieniowej z dwiema strefami, każda obsługiwana osobnym elektrozaworem i zakończonej zraszaczem wynurzalnym. Zakres działań to:

a) prezentacja zawartości ekranu sterownika,

b) sprawdzenie włączenia/wyłączenia lokalnego za pomocą przycisku,

c) włączenie/wyłączenie nawadniania przy pomocy aplikacji mobilnej,

d) uruchomienie jednocześnie dwóch stref nawadniania,

e) próba uruchomienia 3 stref, przy ustawionym parametrze maksymalnie 2 stref,

f) próba uruchomienia nawadniania w sytuacji odcięcia wody w zaworze głównym.

Realizacja scenariusza zweryfikuje rzeczywiste możliwości uruchamiania elektrozaworów przez sterownik oraz przestrzeganie maksymalnej ilości jednocześnie uruchamianych stref. Poza tym sprawdzona zostanie działanie aplikacji mobilnej w telefonie podłączonym do tej samej sieci bezprzewodowej co sterownik nawadniania. Materiały wideo zamieszczono w plikach ScenariuszS1ab.mp4, ScenariuszS1cde.mp4 oraz ScenariuszS1f.mp4

Scenariusz S2 – próba w warunkach rzeczywistych.

S2.1 Nadanie dostępu sterownikowi do bezprzewodowej sieci komputerowej.

S2.1a W stanie początkowym urządzenie nie posiada dostępu do sieci bezprzewodowej. Celem jest zmiana tego i ustalenie dostępu do sieci podanie identyfikatora sieci SSID i hasła.

S2.1b W stanie początkowym urządzenie łączy się z siecią bezprzewodową (zostanie to zrealizowane w punkcie a) Celem działań jest zmiana tego dostępu poprzez podanie wskazanie nowej sieci przez SSID i odpowiadającego jej hasła.
S2.2 Uruchomienie aplikacji mobilnej na telefonie podłączonym do sieci GSM. Wprowadzenie i zapisanie adresu IP oraz hasła do sterownika. Zestawienie tunelu VPN z ruterem sieci bezprzewodowej w której pracuje sterownik. Ponowne uruchomienie aplikacji i weryfikacja komunikacji. Prezentacja wyników działań: wpisanie nieprawidłowego hasła, reset hasła do domyślnego, zmiana nazwy strefy, zmiana ilości stref na z 8 na 16. Uruchomienie sterownika bez baterii w zegarze RTC;
S2.3 Podstawowe sprawdzenie funkcji aplikacji: wyświetlenie stanu stref, listy czujników, list zdarzeń/logów, prognozy pogody oraz zakładki informacje. Dodatkowo należy pokazać możliwe do ustawienia parametry globalne, strefowe i dotyczące czujników oraz sposób ich opisu.
1. Włączenie trybu manualnego przyciskiem oraz telefonem, wyłączenie działających stref.
2. Wymuszenie synchronizacji czasu, weryfikacja czasu sterownika
3. Odświeżenie stanu stref przez naciśnięcie przycisku z nazwą ogrodu.
4. Zablokowanie strefy nr 2 przez aplikację, odblokowanie lokalne, włączenie lokalne,
5. Wyświetlenie czujników wilgotności.
6. Włączenie opcji przepływomierza, symulacja przepływu przy wyłączonych strefach, brak przepływu przy włączonej strefie, reset statusu kontrolera, weryfikacja otrzymania powiadomienia o zdarzeniu niepożądanym, weryfikacja logów sterownika.
S2.4 Zazbrojenie alarmu przez aplikację mobilną, wyzwolenie alarmu i weryfikacja otrzymania powiadomienia na innym telefonie, podłączonym do brokera MQTT przez sieć komórkową.

Realizacja scenariusza zweryfikuje możliwości wstępnej konfiguracji sterownika w sieci bezprzewodowej, działanie aplikacji spoza tej sieci przez tunel VPN oraz działanie alarmu. Materiał wideo z realizacji zamieszczono w pliku ScenariuszS2.1.mp4, ScenariuszS2.2.mp4, ScenariuszS2.3.mp4 i ScenariuszS2.4.mp4.

Scenariusz S3 – próba w warunkach symulowanych.
Parowanie czujnika wilgotności gleby ze sterownikiem oraz sprawdzenie komunikacji pomiędzy tymi urządzeniami.
S3.1 Dodanie czujnika do sterownika.
S3.2 Pomiar wilgotności w wodzie oraz w powietrzu (ręczne wyzwolenie pomiaru).
S3.3 Pomiar wilgotności w powietrzu (automatyczne wyzwolenie pomiaru).
S3.4 Prezentacja danych wymienianych pomiędzy urządzeniami.
S3.5 Zmiana konfiguracji czujnika poprzez zmianę ustawień w sterowniku
Realizacja scenariusza zweryfikuje współpracę zbudowanego czujnika wilgotności ze sterownikiem oraz zaprezentuję komunikaty wymieniane pomiędzy urządzeniami.
Materiał wideo z realizacji zamieszczono w pliku ScenariuszS3.mp4 i ScenariuszS3.3.mp4.

Scenariusz S4 – próba w warunkach symulowanych.

Sprawdzenie implementacji algorytmu Optiserv z uwzględnieniem danych z czujnika bezprzewodowego. W systemie należy skonfigurować 3 strefy, zgodnie z poniższą tabelą nr 22. Konfiguracja ta powinna być przywracana przed rozpoczęciem każdego z wariantów scenariusza S4.1, S4.2 i S4.3.

Tabela 22: Konfiguracja sterownika nawadniania do 4 scenariusza

Parametr	Strefa nr 1	Strefa nr 2	Strefa nr 3	Strefa nr 4
Czujnik wilg. gleby	TAK	NIE	NIE	NIE
Strefa referencyjna	-	1	1	-
Minimalny czas nawadniania L[min]	8min	5min	3min	4min
Okres nawadniania PERIOD [dni]	0.5dni	1dni	0.75dni	1dni
Ostatnie nawadnianie	-1dni	-1dni	-1dni	-1dni
Wilgotność	4095	4095	4095	4095

Natomiast ustawienia globalne sterownika należy skonfigurować następująco:

Minimalna wartość oczekiwana opadu: 5mm,

Ewapotranspiracja: 5mm,

Ilość maksymalnie działających stref: 2,

Dopuszczalny przedział temperatur czujnika to: <2°C;30°C>,

Data graniczne nawadniania: 1 kwiecień – 1 październik,

Czas przeprowadzenia pomiędzy wschodem a zachodem Słońca.

Symulację należy uruchomić z czujnikiem z układem pomiarowym umieszczonym w powietrzu. **S4.1:** Pomiędzy 3 i 5 minutą należy włożyć czujnik wilgotności do wody symulując skutecznie nawodnienie (czujnik sam powinien zainicjować transmisję danych pomiarowych).

S4.2: Po 3 minucie, po zakończeniu nawadniania strefy nr 3, w trakcie gdy nawadniana będzie strefa nr 1 i 2, należy doprowadzić do zwiększenia temperatury czujnika glebowego powyżej 30°C (strefy powinny mieć zawieszone nawadnianie).

S4.3: W ustawieniach sterownika ustawić minimalną ewapotranspirację np. 1mm/dzień (jej wartość powinna być poniżej aktualnej wyliczonej na podstawie prognozy pogody). Po uruchomieniu sterownika, przed przejściem przez pierwszą pętlę algorytmu Optiserv należy wyświetlić ranking punktów strefowych.

Realizacja scenariusza S4 zaprezentuje działanie implementacji algorytmu Optiserv w sterowniku nawadniania. Ponadto zweryfikowanie zostanie ograniczenia temperaturowego w strefie z czujnikiem wilgotności gleby oraz zawieszenie nawadniania ze względu na przekroczenie minimalnej (ustalonej przez użytkownika) wartości ewapotranspiracji. Materiał wideo z realizacji zamieszczono w plikach ScenariuszS4.1.mp4, ScenariuszS4.2.mp4 oraz ScenariuszS4.3.mp4

4. PRZEPROWADZANIE BADAŃ CZUJNIKA WILGOTNOŚCI GLEBY

W ramach badań zbudowanego czujnika wilgotności gleby przeprowadzono następujące eksperymenty pomiarowe:

E1) wyznaczenie charakterystyki czujników podczas wolno zmiennej wilgotności podłoża,

E2) wyznaczanie wilgotności gleby metodą grawimetryczną (suszarkowo-wagową),

E3) wyznaczenie dynamicznej charakterystyki czujników poprzez zmieniające się warunki środowiska zewnętrznego,

E4) określenie stabilności pomiarów w temperaturze 24-25°C.

Problem badawczy: Czy pojemność kondensatora koplanarnego na płytce PCB, zasilanego impulsami napięcia o maksymalnej amplitudzie ok 3.3V, może zostać wykorzystana do określenia masowej zawartości wody w glebie.

Hipoteza badawcza: Jeżeli zawartość wody w glebie będzie się zmniejszać, to wraz z nią zmniejszać się będzie pasożytnicza pojemność elektryczna obwodu drukowanego.

4.1 Plan eksperymentów

W celu sprawdzenia hipotezy badawczej należy przeprowadzić eksperyment E1. Polega on na cyklicznym wykonywaniu pomiarów wilgotności nasączonej gleby pozostawionej do wysychania i jednoczesnym rejestrowaniu masy pojemnika z glebą. Do wykonywania pomiarów zostaną wykorzystane: jeden autorski (f=40MHz) oraz jeden referencyjny czujnik wilgotności (f=75Mhz). Czas przeprowadzenia pomiaru został wstępnie oszacowany na 4 do 7 dni z możliwym wcześniejszym warunkiem stopu związanym z osiągnięciem przez podłoże 22-27% VWC, stanowiąca warunek wyzwalania nawadniania różnego rodzaju niepiaszczystych gleb lub 6-8% dla piaszczystych[113].

Wykorzystując możliwość jaką daje pierwszy eksperyment, zaplanowano również kalibrację prototypowych czujników. Aby jej dokonać, należy ustalić początkową zawartość wody w glebie, użytej do eksperymentu E1. Zostanie to osiągnięte przez destrukcyjną metodę pomiaru z eksperymentu E2.

W uzupełnieniu do głównego doświadczenia E1, zostanie podjęta próba wyznaczenia charakterystyki czujników wilgotności w środowisko wodnym, wilgotnej oraz w wyprażonej glebie, w zależności od temperatury. Stanowi to przedmiot zadania nr E3.

Aby w budowanym sterowniku skorzystać z punktów wyzwalania nawadniania, określonych w opracowaniu[113] Instytutu Rolnictwa i Zasobów Naturalnych Uniwersytetu Nebraska w Lincoln, wykonano eksperymenty pomiarowe dla trzech rodzajów gleb:

1. Niezamulony torf - uniwersalnej ziemia kwiatowa zakupiona w sklepie ogrodniczym i pakowana w foliowe worki,
2. Czarnoziem, gleby pozyskana z działki ogrodniczej autora pracy,

3. Drobnoziarnisty piasek, pozyskany ze zbocza pagórka typu morenowego,
4. Gliny pylastej, z niewielką domieszką gleby próchniczej, z pobliskiej budowy.

4.2 Budowa środowiska doświadczalnego

Główną część środowiska doświadczalnego stanowiła samodzielnie zbudowana waga, która została oparta o belkę tensometryczną NA27 z maksymalnym obciążeniem o wartości 10kg. Jej zadaniem było utrzymanie i ważenie doniczki z ziemią o średnicy podstawy ok. 15cm. Waga została wykonana ze sklejki o grubości 12mm i składa się z prostokątnej podstawy dolnej o wymiarach 45x21 cm oraz prostokątnego podestu o wymiarach 26x20cm (pełniącego rolę szali wagi). Niezakryta przez szalkę część podstawy została przeznaczona na rozmieszczenie elektronicznego układu pomiarowego, co zostało przedstawione rysunku nr 65.

Układ pomiarowy został oparty o płytkę rozwojową ESP32-WROOM-32E z mikrokontrolerem ESP32, opisanym w rozdziale 1.3.1. Moduły funkcjonalne wraz ze sposobem ich podłączenia do głównej płytki oraz ich parametry zasilania zostały wyszczególnione w tabeli nr 23, natomiast moduły zasilające w tabeli nr 24. Schemat elektryczny całego układu pomiarowego został zaś uwidoczniony na rysunku nr 66.

Tabela 23: Moduły funkcjonalne układu pomiarowego

Lp.	Nazwa modułu	Interfejs	Piny (moduł:platforma sprzętowa)	Napięcie zasilające	Natężenie max
1	Wyświetlacz LCD 4x20 znaków	I2C	SDA:GPIO21(SDA) SCL:GPIO22(SCL)	5V	120mA
2	Moduł czytnika kart micro SD	SPI	CS:GPIO5(SS) CLK:GPIO18(SCK) MISO:GPIO19(MISO) MOSI:GPIO23(MOSI)	3.3V-5V	b. d.
3	Czujnik wilgotności gleby DFRobot SEN0308	GPIO	GPIO25(ADC)	3.1V-5V	b. d.
4	Czujnik wilgotności gleby Pino-Tech SoilWatch	GPIO	GPIO26(ADC)	3.3V-5.5V	24mA
5	Prototyp czujnika wilgotności	GPIO	GPIO27(ADC)	3.0V-3.3V	b. d.
6	Cyfrowy czujnik temperatury DS18B20	GPIO	DQ:GPIO15	3.0V-5.5V	1.5mA
7	Wzmacniacz HX711z belką tensometryczną NA27	GPIO	SCK:GPIO1 DT:GPIO16	2.6V-5.5V	1.5mA
8	Moduł zegara czasu rzeczywistego (RTC) DS3231	I2C	SDA:GPIO21(SDA) SCL:GPIO22(SCL)	3.3-5V	0,65mA

Tabela 24: Moduły zasilające układu pomiarowego

Lp.	Nazwa modułu	Napięcie wejściowe	Napięcie wyjściowe	Natężenie max	Uwagi
1	Moduł zasilający do płytek stykowych MB102	6.5V-12V	3.3V i 5V	700mA	-
2	Zasilacz impulsowy 12V/2A - wtyk DC 5,5/2,1mm	230V AC	12V	2A	-
3	UPS Fideltronik Ares 500	230V AC	230V AC	10A	moc rzeczywista 300W moc pozorna 500VA

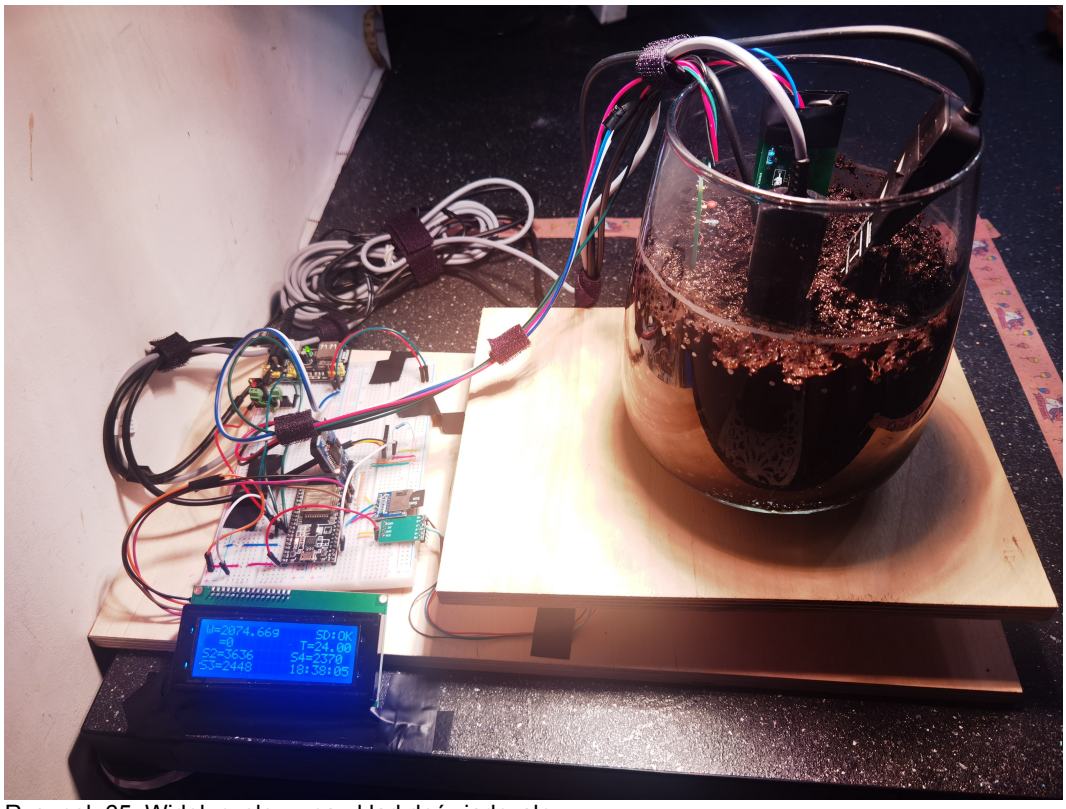
Na ekranie LCD, który odświeżany jest co około 30 sekund, w sposób ciągły wyświetlane są informacje: waga doniczki z glebą i czujnikami, odczyty pomiarów z badanych czujników, temperatura gleby, godzina ostatniego pomiaru oraz poprawność zapisu na nośnik pamięci. Na czas odświeżania składa się stała wartość 28 sekund oraz 2 czasy sekwencyjnych pomiarów czujników po około 1s. Pomiary przebiegały w sposób bezobsługowy, nie przewidziano żadnych przycisków sterujących, a ciągłość pracy urządzenia można określić porównując wyświetlaną godzinę z aktualnym czasem.

Zgodnie z przykładem w dokumentacji do wzmacniacza belki tensometrycznej (biblioteka: HX711_ADC) dokonano kalibracji zbudowanej wagi, poprzez kompilację i przesłanie programu kalibracyjnego do układu ESP32. Komunikacja z programem następowała poprzez komputer PC za pomocą połączenia szeregowego w terminalu. Na wadze kuchennej trzykrotnie zważono opakowanie cukru obliczając jego średnią masę. Następnie uruchomiono program do kalibracji, który dokonał tarowania, a następnie ustawiono na wadze znaną masę. Zadaniem programu było obliczenie współczynnika kalibracji, który następnie zastosowano w kodzie źródłowym oprogramowania obsługującego przebieg eksperymentu pomiarowego.

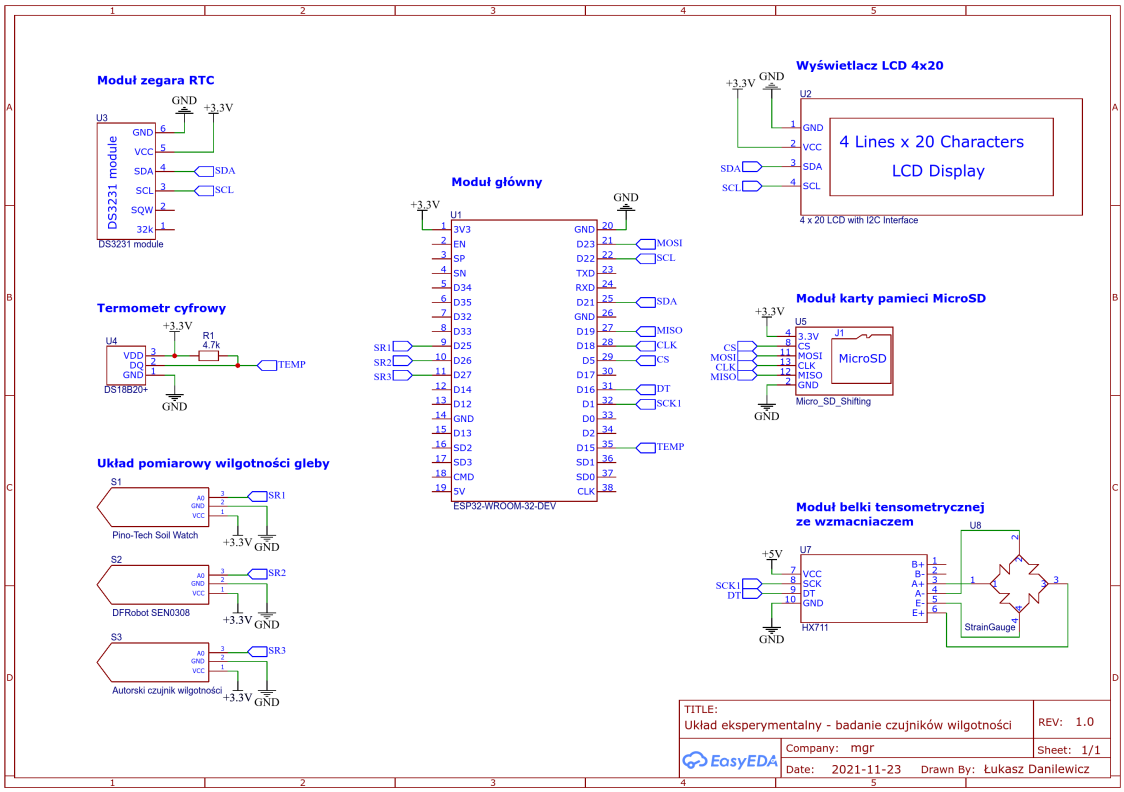
Zabezpieczeniem przed utratą zasilania sieciowego lub jego wahaniom, a w konsekwencji przerwaniem eksperymentu, stanowi zasilacz awaryjny UPS Fideltronik Ares 500. Jednakże, w sytuacji utraty zasilania przez układ pomiarowy, przy kolejnym uruchomieniu zostanie zachowany aktualny czas, a wskazania wagi zostaną skorygowane za pomocą wstępnie ustalonego współczynnika kalibracji. W razie problemów z otwarciem pliku, np. z powodu jego uszkodzenia po utracie zasilania, zostaje utworzony plik z inną nazwą (zawierającą znacznik czasowy w nazwie) w którym zapisywane będą kolejne wyniki.

Podsumowując, w skład układu doświadczalnego wchodzą następujące komponenty, które zostały przedstawione na rysunkach od 65, 67 i 68:

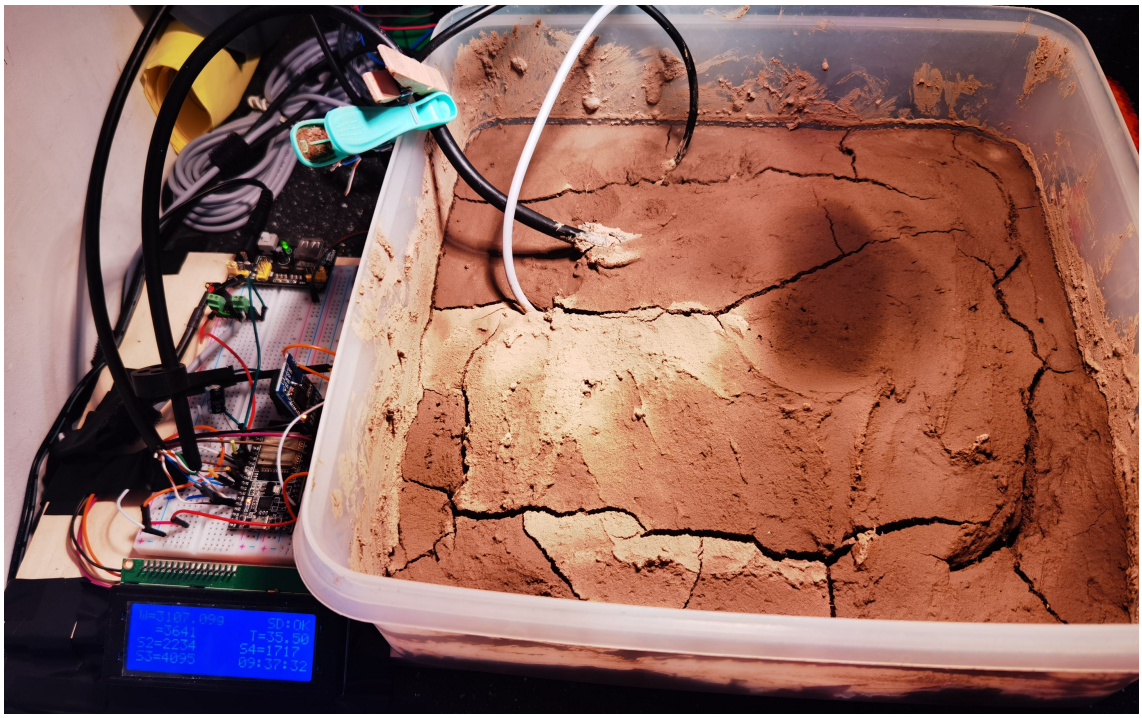
1) elektroniczny układ pomiarowy z wyświetlaczem LCD(ESP32 z modułami oraz okablowaniem na płytkach stykowych),
2) żarówka grzewcza (promiennik podczerwieni) o mocy znamionowej 125W,
3) waga,
4) donica z ziemią,
5) 3 czujniki wilgotności(SEN0308 prod. DFRobot, Soil Watch prod. Pino-Tech oraz autorski)
6) 1 czujnik temperatury,
7) zasilacz awaryjny UPS.

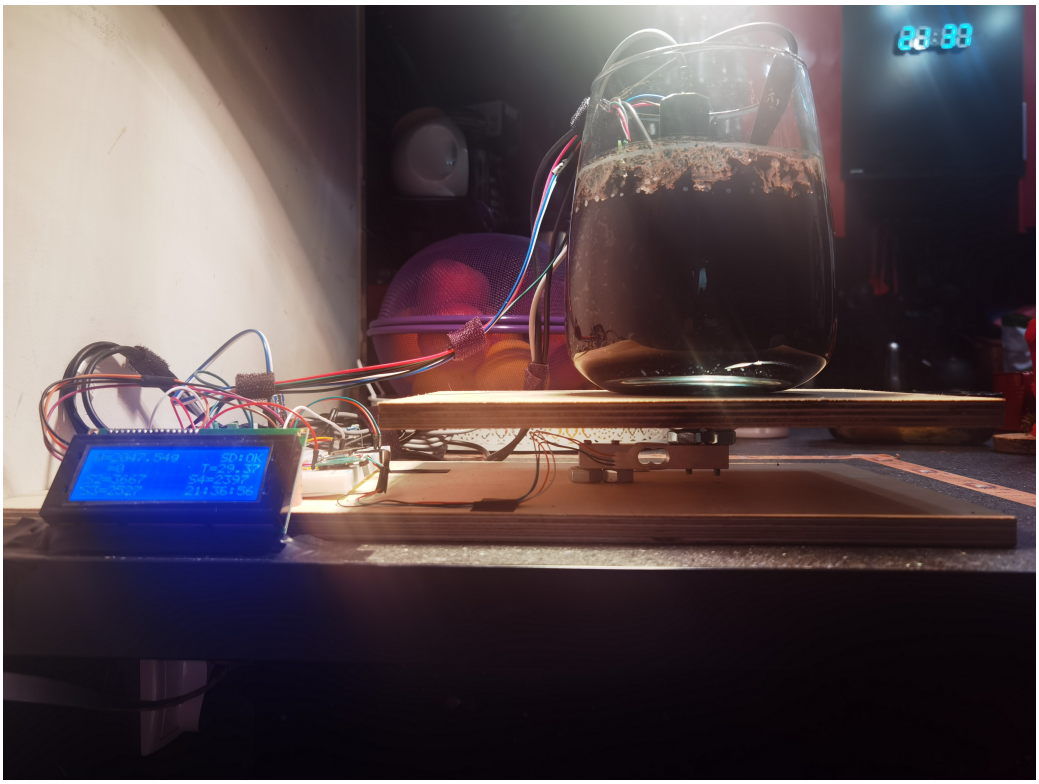


Rysunek 65: Widok czołowy na układ doświadczalny



Rysunek 66: Schemat elektryczny układu z eksperymentu pomiarowego


Rysunek 67: Widok w trakcie pomiarów dla gleby nr 2


Rysunek 68: Widok na szlę wagi i belkę tensometryczną z glebą nr 1

Zgodnie z diagramem maszyny stanów z rysunku nr 69, po sprawdzeniu komunikacji z modułami, następuje ich inicjalizacja i rozpoczyna się zamknięty cykl pomiarowy, kończący się

poprzez odłączenie zasilania. Po serii pomiarów z 3 czujników wilgotności, temperatury i masy układu, następuje zapis zmierzonych wartości na kartę pamięci micro SD. Zapis każdej takiej serii pomiarowej jest poprzedzony znacznikiem czasu. Struktura pozyskiwanych danych została przedstawiona w tabeli nr 25, a plik z danymi został zawarty w elektronicznych załącznikach do niniejszej pracy.

Tabela 25: Struktura tabeli z danymi pomiarowymi

DT (Data i godzina)	$X_{DFRobot}$	$X_{SoilWatch}$	$X_{prototyp1}$	$X_{prototyp2}$	T_{gleby} [°C]	$M_{układu}$ [g]

DT – data i godzina rozpoczęcia pomiaru w formacie YYYY-MM-DD HH:MM:SS

$X_{prototyp}$ – liczba określająca wilgotność zmierzoną za pomocą badanych czujników

$X_{soilWatch}$ – liczba określająca wilgotność zmierzoną za pomocą czujnika SoilWatch

$X_{DFRobot}$ – liczba określająca wilgotność zmierzoną za pomocą czujnika DFRobot

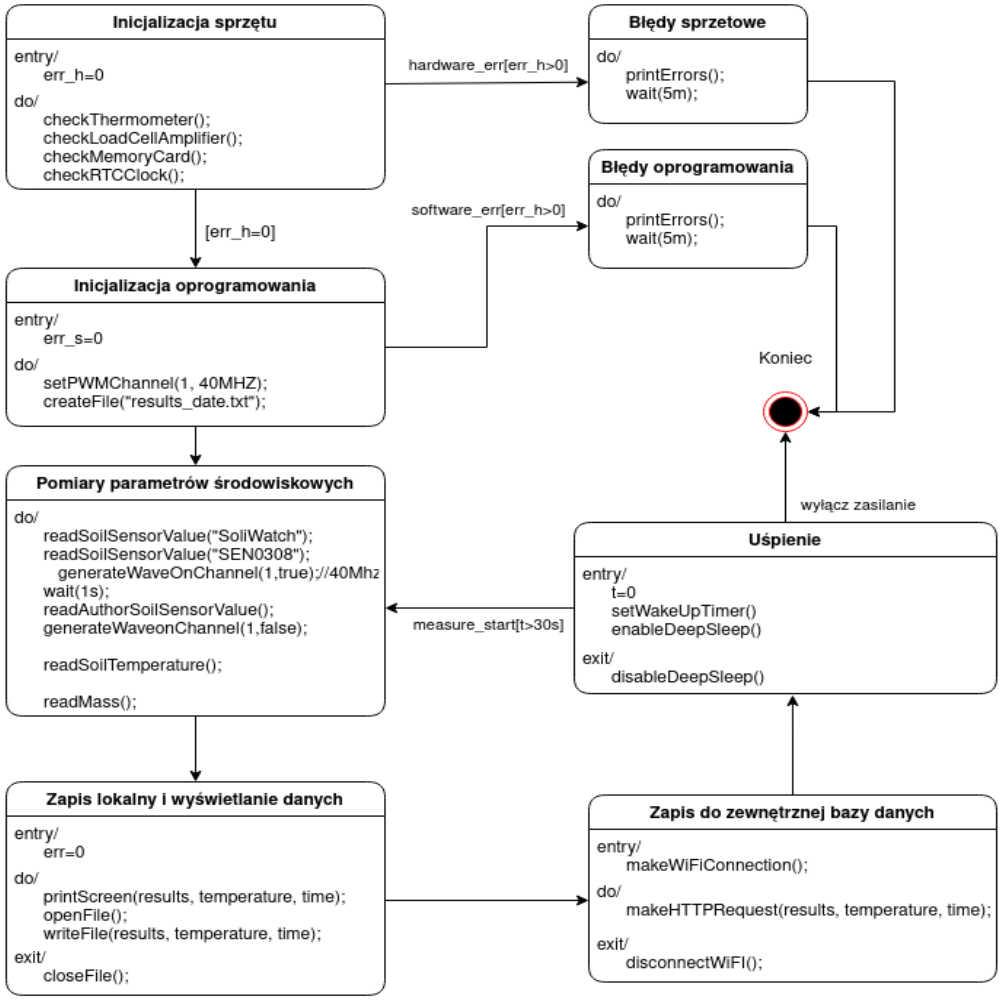
T_{gleby} – temperatura gleby zmierzona sondą z termometrem cyfrowym DS18B20

$M_{układu}$ – masa pojemnika, gleby i czujników wilgotności zmierzona przy wykorzystaniu belki tensometrycznej NA27 i wzmacniacza HX711

Dodatkowo, każdy zestaw danych jest zapisywany do zewnętrznej bazy danych MySQL na serwerze dostępnym pod domeną autora - danilewicz.com. Wykorzystano do tego kartę radiową ESP32, połączenie z lokalną siecią bezprzewodową z dostępem do internetu oraz protokół HTTP. Poza zapewnieniem zabezpieczenia danych wynikowych poprzez przechowywanie ich w innej, odległej lokalizacji, zapewniło to możliwość bieżącej kontroli przebiegu eksperymentu. W tym celu dane te były przedstawiane na wykresie i dostępne pod adresem http://danilewicz.com/wykres.php. Przykładową wizualizację danych podczas trwania eksperymentu przedstawiono na rysunku nr 70. Na osi rzędnych odkładane są wartości pomiarów, zaś oś odciętych stanowią kolejne numery pomiarów. W ramach postępowania zgodnie z dobrymi praktykami podczas tworzenia aplikacji internetowych, do zabezpieczenia skryptu do którego kierowane były żądania HTTP i dokonującego tworzenia kolejnych rekordów:
a) programowo wymuszono obecność wszystkich parametrów w żądaniu,
b) filtrowano parametry żądania przez określenie dopuszczalnych znaków,
c) ograniczono uprawnienia użytkownika MYSQL tylko do zapytań typu INSERT i SELECT,
d) zastosowano tzw. *prepared statements* podczas zapisu do bazy danych drastycznie ograniczając możliwości atak SQL Injection[114],
e) ograniczono zakres dopuszczalnych adresów IP do poprzez ustawienie 1 oktetu adresu takiego jak u dostawcy internetu autora pracy.

Produkt działania każdego czujnika dostępny jest w sposób analogowy i wystawiany na jego porcie wyjściowym jako sygnał napięciowy. Podawany jest on na wejście 12-bitowego konwertera analogowo-cyfrowego mikrokontrolera ESP32, dając odczyt w postaci liczby

z przedziału <0;4095>, co odpowiada wartościom napięcia z przedziału <0; 3.3V>. Z racji braku potrzeby obliczeń zmiennoprzecinkowych i zaokrągleń na nośnik pamięci zapisywane są właśnie surowe, całkowitoliczbowe odczyty z konwertera ADC, a nie odpowiadające im wartości napięcia elektrycznego.



Rysunek 69: Diagram maszyny stanów oprogramowania nadzorującego eksperyment pomiarowy

Maksymalna częstotliwość próbkowania konwertera analogowo-cyfrowego, zgodnie z kartą katalogową ESP32, to 200ksps (kilo samples per second). Jednakże, przy oprogramowaniu układowym pod kontrolą frameworka Arduino, kontrolnie zmierzona wartość oscylowała wokół 20ksps. Na tej podstawie przyjęto, aby czas pomiaru napięcia z każdego z czujników wynosił około 1s, co w konsekwencji oznaczało, że wynik pomiaru z danego czujnika będzie wartością średnią z 20.000 odczytów z konwertera ADC.

Eksperyment pomiarowy został wykonany zgodnie z poniższą instrukcją:

1. Zmontować układ pomiarowy zgodnie ze schematem nr 7 i ustawić na stabilnym równym podłożu. Połączenia elektryczne do czujnika i w jego obrębie mają być lutowane. Przewody

powinny być połączone w wiązki. Zasilacz układu pomiarowego podłączyć do naładowanego zasilacza awaryjnego UPS.

2. Podać zasilanie do układu pomiarowego i dokonać tarowania wagi.

3. Za pomocą zbudowanej wagi dokonać pomiaru masy m0 - pustego pojemnika na glebę.

W naczyniu pomiarowym zaznaczyć markerem linię oznaczającą objętość ok 2l. Nalać wody do tej linii i zmierzyć jej masę.

4. Następnie po osuszeniu naczynia nasypać ziemi do wyznaczonej linii i dokonać pomiaru masy gleby.

5. W pojemniku na glebę, korzystając ze wskazań zbudowanej wagi, umieścić ok 2kg suchej gleby.

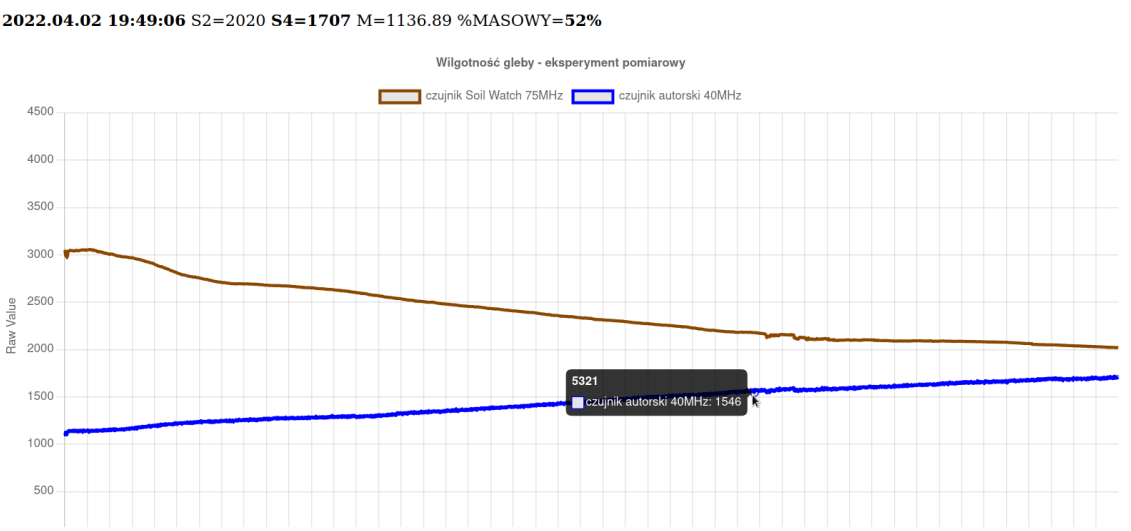
6. Dodawać porcjami wody do gleby, dokładnie mieszając składniki, aż do osiągnięcia ok 50% zawartości masowej wody. Ciecz powinna mieć temperaturę w przybliżeniu taką jak gleba. Zanotować ilość dolanej wody jako m_w, obliczoną jako różnicę mas układu pomiarowego.

7. W środkowej części pojemnika umieścić 3 czujniki wilgotności gleby i rozpocząć eksperyment pomiarowy resetując główny moduł (ESP32) układu pomiarowego. Zanotować datę i godzinę rozpoczęcia pomiarów a także początkową masę pojemnika, nasączonej gleby i czujników.

8. Co najmniej 5 razy na dobę kontrolować stan układu pomiarowego. Dodatkowo warto sprawdzać jego przebieg na dedykowanej stronie www.

9. Po 5 dniach trwania eksperymentu lub po odparowaniu dolanej wody, zdjąć zasilanie z układu pomiarowego (w czasie kiedy nie dokonywane są pomiary, co jest sygnalizowane na ekranie).

10. Wykreślić zależności z uzyskanych z uzyskanych wyników i wyciągnąć wnioski na temat charakteru zmienności danych.



Rysunek 70: Wyniki pomiarów dostępne zdalnie w trakcie trwania eksperymentu

W poniższej tabeli nr 26 zaprezentowano, na przykładzie doświadczenia z glebą nr 1, zestaw wstępnych danych pomiarowych uzyskanych przed rozpoczęciem właściwego eksperymentu.

Tabela 26: Masy potrzebne do opracowania wyników eksperymentu pomiarowego

Lp.	Nazwa	Oznaczenie	Wartość
1	Masa pojemnika	m_p	140±0.5g
2	Masa wody do znacznika	m_{wz}	2037±0.5g
3	Masa ziemi do znacznika	m_{zz}	1262±0.5g
4	Masa dolanej wody	m_w	484±0.5g
5	Masa gleby, wody i pojemnika	m_1	1879±0.5g
6	Masa początkowa pojemnika wraz z czujnikami	m_0	1970±0.5g

4.3 Przeprowadzenie eksperymentów pomiarowych

4.3.1 Wyznaczenie charakterystyki czujnika podczas wolno zmiennej wilgotności podłoża

Celem doświadczenia jest uzyskanie charakterystyk wartości pomiarowych czujników wilgotności w zależności o wolumetrycznej zawartości wody w różnych rodzajach podłoży. Zestawienie informacji o wykorzystanych glebach, wyliczone gęstości i procentowy udział wody, statystykę pomiarów oraz przebadany zakres wolumetrycznej zawartości wody zostały przedstawione w tabeli nr 27. W rachunku błędów wykorzystano wzory na błąd maksymalny. Za błędy pomiaru masy i objętości przyjęto odpowiednio $\Delta m=1g$ i $\Delta V=10cm^3$.

Tabela 27: Zestawienie danych o glebach z doświadczenia pomiarowego

	Gleba nr 1	Gleba nr 2	Gleba nr 3	Gleba nr 4
Rodzaj gleby	niezamulony torf	gleba brunatna	drobnoziarnisty piasek	glina pylasta
gęstość próbki	0,66±0,01 g/cm3	1,40±0,01 g/cm3	1,40±0,01 g/cm3	2,32±0,01 g/cm3
udział masowy wody	75,19%	14,60%	3,89%	14,86%
gęstość nasypowa	0,13±0,01 g/cm3	1,36±0,02 g/cm3	1,48±0,02 g/cm3	1,99±0,01 g/cm3
czas eksperymentu	114h 03m	37h 43m	32h 38m	27h 06m
ilość pomiarów	13482	2920	2355	3738
średnia temperatura	32,9°C	38,2°C	33,9°C	35,1°C
zakres zbadanej wolumetrycznej zawartości wody (VWC)	25,6%-58,7%	5,3%-42,4%	2,4%-31,3%	1%-47,8%

Przeprowadzono cztery eksperymenty pomiarowe. Dla gleby nr 1 czujniki były posadowione pionowo w doniczce, a głębokość ziemi była około 5mm powyżej najdłuższego czujnika. Dzięki energii dostarczanej od góry próbka przesychała nierównomiernie, jednakże pomiar, mający charakter uśrednionego, dokonywany był w przybliżeniu na całej głębokości. Wybór gleby nr 1 oraz kierunek umiejscowienia czujnika zostały dokonane celowo, mając na uwadze, że taka gleba bywa wykorzystywania do uprawach w doniczkach.

Natomiast w stosunku pozostałych serii pomiarowej, której środowiskiem były gleby nr 2, 3 oraz 4, zastosowano poziome rozmieszczenie czujników wilgotności, również

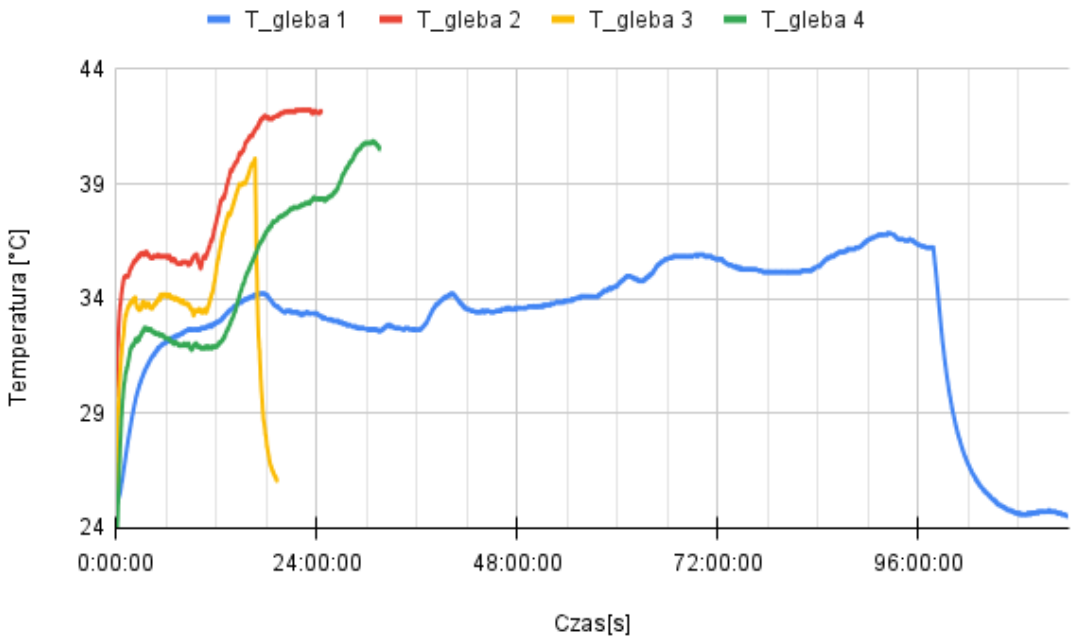
przykrytych kilkumilimetrową warstwą ziemi (rysunek nr 67). Dało to większą powierzchnię parowania próbki i około 5 krotnie mniejszą głębokość niż poprzednio, co znacząco wpłynęło na skrócenie długości doświadczeń. Objętość gleby wynosiła ok. 1900±10cm^3. Taki sposób instalacji, ze względu na wymiary prostokątne wymiary elementu pomiarowego, pozwala lepiej wyznaczyć obecność wody na określonej głębokości.

Na podstawie wstępnych danych z tabeli 27 sprzed doświadczenia oraz przy wykorzystaniu zgromadzonych wyników pomiarów czujników ze znacznikiem czasu oraz aktualną masą gleby wyznaczano przybliżony VWC dla gleby w czasie. Za niepewność związaną z opisaną wyżej metodą pomiaru przyjęto Δp=5%, uzależniając jego wartość od m. in. od różnicy natężenie oświetlenia pomiędzy częścią środkową a brzegami próbki. Powierzchnia próbki z glebą była ogrzewana nierównomiernie – największa moc docierała do centralnej części. Ponadto, warto podkreślić, że pomiary czujników dotyczą przestrzeni w zasięgu pojedynczych milimetrów od ich elementów wykonawczych, natomiast powyższy parametr określający ilość wody jest wyliczany w stosunku do całej objętości gleby. Jednakże, zaproponowane umieszczenie czujników w stosunku do głębokości naczynia daje również uśredniony pomiar wielkości określającej wilgotność gleby. Ostatecznie, wyliczone maksymalne niepewności przypadkowe pomiarów dla poszczególnych gleb wyniosły odpowiednio: 4%, 1,5%, 1% i 1,2%.

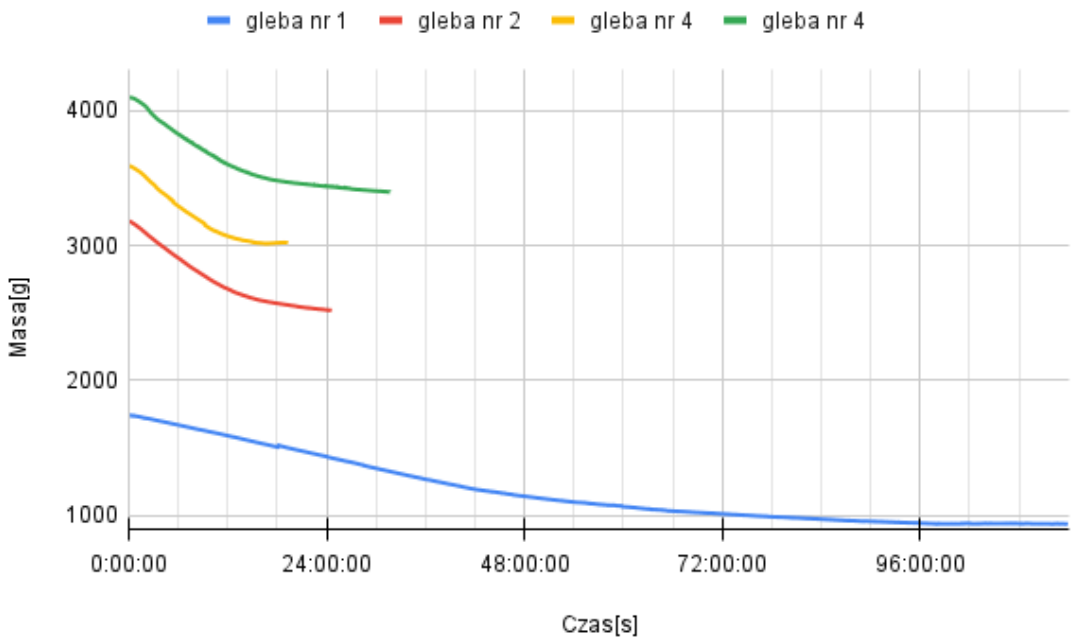
Przebieg doświadczeń w domenie czasu ze względu na temperaturę i masę próbek przedstawiono na wykresach zbiorczych na rysunkach o numerach odpowiednio 71 i 72.

Natomiast przebieg wyznaczonych zależności dla każdego z badanych czujników w odniesieniu do każdego z rodzaju podłoża zostały przedstawione na rysunkach 73, 74, 75 i 76. Zupełnie różne charaktery trendów wykreślonych zależności wynikają tylko i wyłącznie z ich sposobu działania/budowy: dla czujnika wzorcowego SoilWatch wyższy odczyt oznacza wyższą zawartość wody, dla czujnika autorskiego zaś większa wartość pomiarowa oznacza niższą zawartość wody. Do obliczenia VWC wykorzystano wzory nr 14 i 15, do których wartości gęstości nasypowej oraz grawimetrycznej zawartości wody uzyskano po przeprowadzeniu eksperymentu E2 (metoda suszarkowo-wagowa).

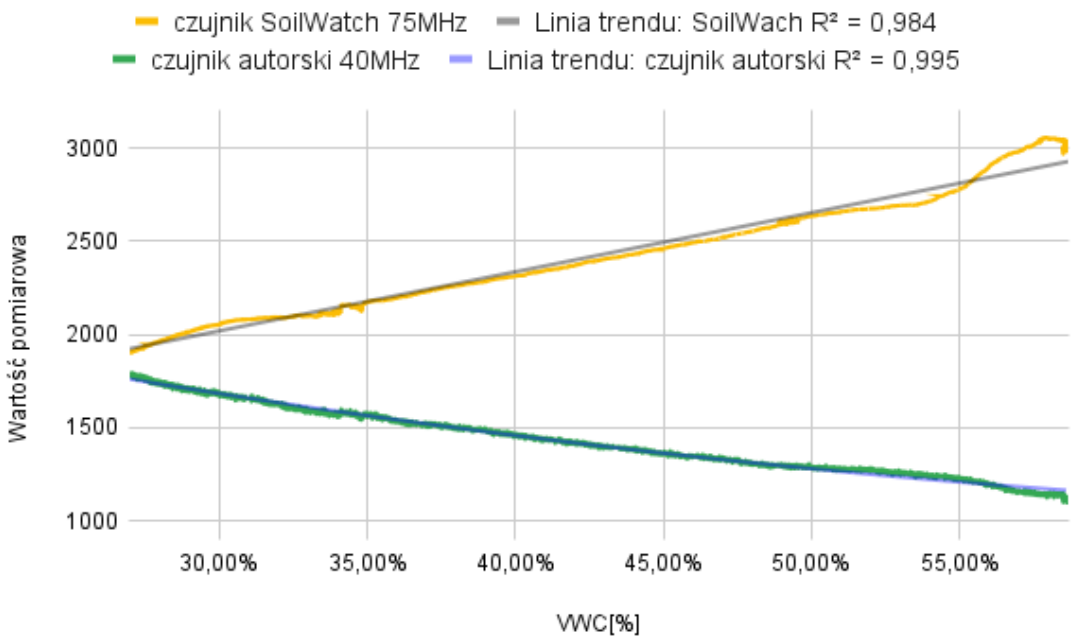
Na wykresach przedstawiono wielomianową linię trendu wraz z wyliczoną wartością współczynnika determinacji R-kwadrat. Charakter trendu jest zgodny z przewidywaniami teoretycznymi, posiadając silną korelację dodatnią dla każdej z gleb na poziomie R_1^2=0,995, R_2^2=0,996. R_3^2=0,994 oraz R_4^2=0,956.



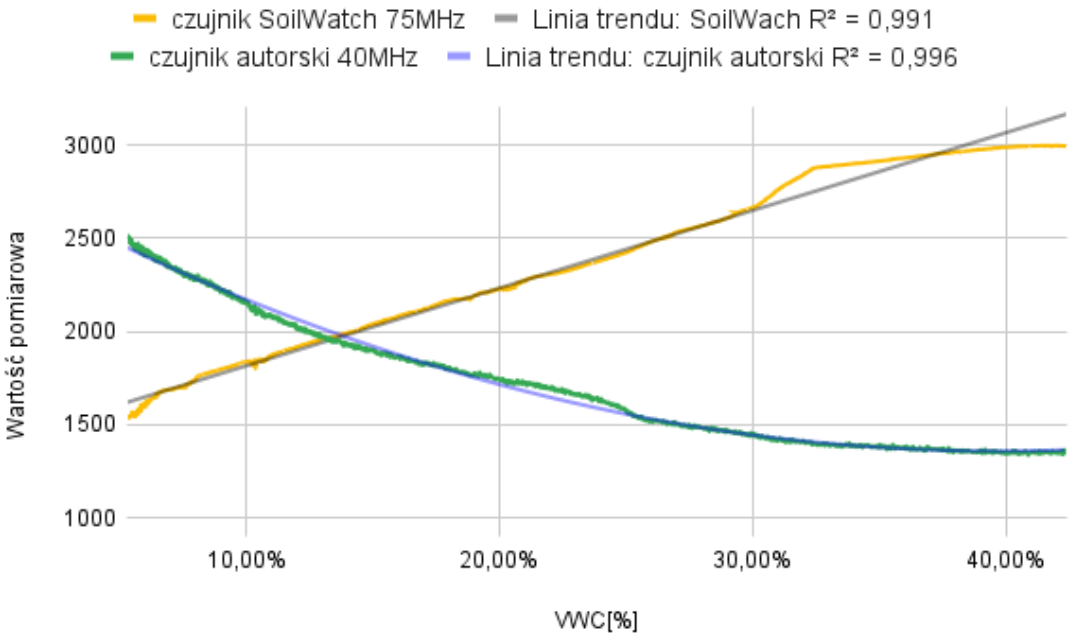
Rysunek 71: Zmiana temperatury gleby w czasie dla każdego z eksperymentów pomiarowych



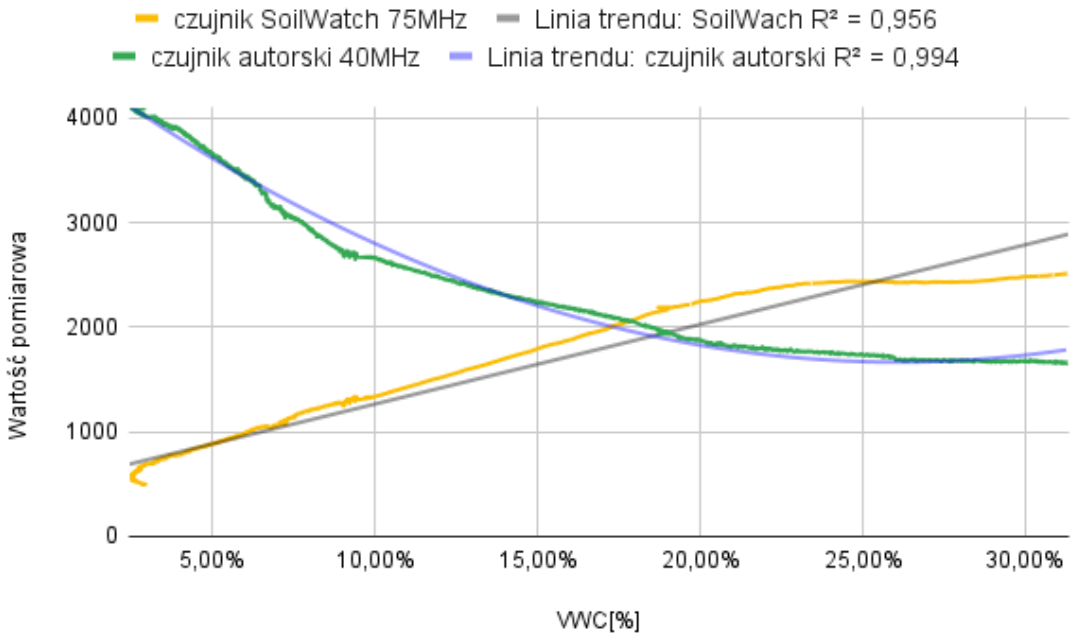
Rysunek 72: Zmiana masy gleby w czasie dla każdego z eksperymentów pomiarowych



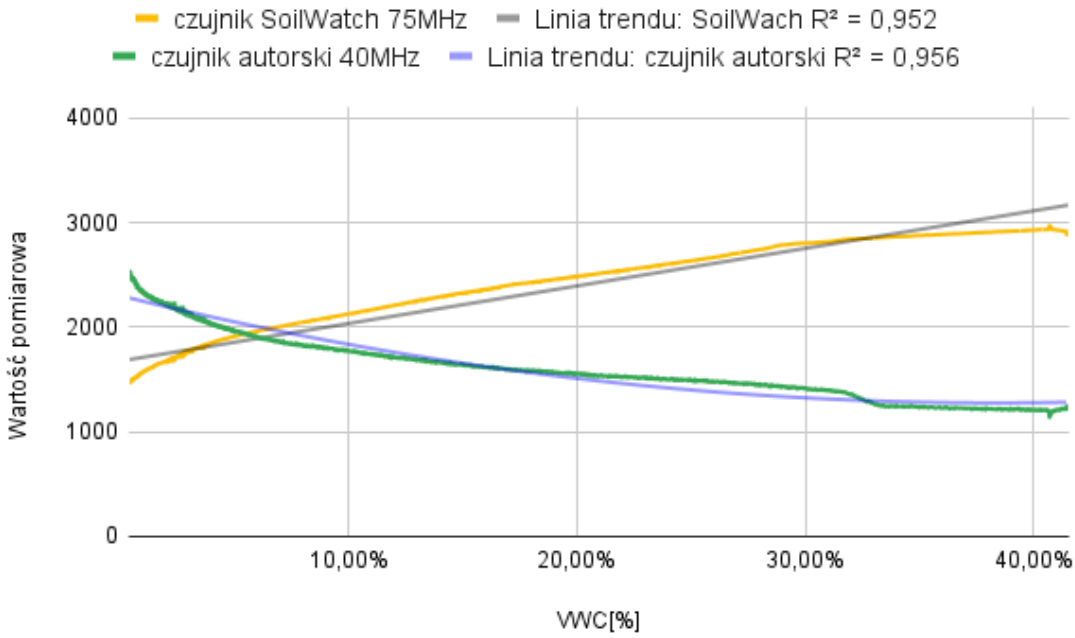
Rysunek 73: Zależność wartości pomiarowych od objętościowej zawartości wody (VWC) dla gleby nr 1



Rysunek 74: Zależność wartości pomiarowych od objętościowej zawartości wody (VWC) dla gleby nr 2



Rysunek 75: Zależność wartości pomiarowych od objętościowej zawartości wody (VWC) dla gleby nr 3



Rysunek 76: Zależność wartości pomiarowych od objętościowej zawartości wody (VWC) dla gleby nr 4

Z uzyskanych wyników pomiarów, przyjmując 5% margines na niestabilność pomiarów, wyliczono wartości czujnika, które odpowiadają sugerowanym[113] punktom wyzwalania nawadniania wyrażonych w VWC[%] i zaprezentowano w tabeli nr 28. Z uzyskanych wartości wynika, że w docelowym systemie należy przewidzieć możliwość wyboru rodzaju gleby i związanego z nim punktu wyzwalania. Z drugiej jednak strony podłoża nr 1 (niezamulony torf)

i 3 (drobnoziarnisty piasek) nie są typowymi pod uprawy. W przytoczonym artykule nie ma odniesienia w ogóle do torfu, stąd podane w tabeli dane 44-46% zostały określone doświadczalnie przez autora pracy. Biorąc ten fakt pod uwagę można stwierdzić, że dla czarnoziemu i mieszaniny gliny pylastej z glebą z powierzchni otrzymane wyniki są zbliżone – mając na uwadze zakres 1050-4096. Ustalenie punktu do rozpoczęcia nawadniania na poziomie odczytu z czujnika 1500, zapewni z zapasem sugerowaną wolumetryczną zawartość wody w podłożu od której system nawadniania powinien dążyć do jej uzupełnienia.

Tabela 28: Podsumowanie wyników eksperymentu E1

	Gleba nr 1	Gleba nr 2	Gleba nr 3	Gleba nr 4
Rodzaj gleby	niezamulony torf	czarnoziem	drobnoziarnisty piasek	glina pylasta
%VWC[113]	44-46%	25-27%	6-8%	23-25%
zakres wyników pomiarowych czujnika	1383-1347	1579-1504	3447-2944	1516-1492
przyjęty punkt wyzwalania	1347	**1579**	2944	**1492**

4.3.2 Wyznaczanie wilgotności gleby metodą grawimetryczną (suszarkowo-wagową)

Metoda suszarkowo-wagowa[115] polega na pobraniu i zważeniu próbek gleby, a następnie suszeniu ich w temperaturze 105°C, aż do ustabilizowania się masy. Grawimetryczna zawartość wody obliczana jest jako stosunek ubytku masy wody do masy początkowej próbki.

Ze względu na wysoką temperaturę, w doświadczeniu nie mogły wziąć udziału czujniki wilgotności. Jednakże jego przeprowadzenie jest warunkiem koniecznym do spełnienia przed rozpoczęciem eksperymentów E2 i E3. Zawartość procentowa wody w glebie jest potrzebna do ustalenia początkowej ilości wody w pierwszym z wymienionych eksperymentów. Natomiast produkt w postaci wyprażonej gleby jest konieczny do doświadczenia E3 do określenia granicznych odczytów czujnika.

W doświadczeniu zostały użyte:

- płaska, okrągła, metalowa foremka do ciasta o głębokości 4cm,
- szklana, kuchenna miara o objętości pomiarowej do 1l,
- piekarnik elektryczny,
- waga kuchenna,
- 400-500ml gleby każdego rodzaju.

Dokonano wytarowania wagi, po czym dokonano pomiaru masy foremki (m_f=226±0.5g) oraz szklanej miary (m_p=318±0.5g). Do miary wsypano 500±10ml gleby po czym została ona przesypana do foremki, ubita i zważona (m_0=484±0.5g). Stąd początkowa masa gleby wynosiła $m_{go}=m_0-m_f$=484g-226g). Równolegle do podjętych działań następowało rozgrzewanie piekarnika do 105°C. Po osiągnięciu tej temperatury metalowy pojemnik z glebą został umieszczony w komorze piekarnika.

Każdorazowo czas podgrzewania ustawiano na 15min, a po jego upłynięciu sygnalizowanym dźwiękiem, próbka gleby była wyjmowana, ważona oraz mieszana. Ta ostatnia czynność miała na celu doprowadzenia bardziej wilgotnego materiału na powierzchnię celem przyspieszenia odparowywania wody. Ostatecznie gleba wracała do komory i ten ciąg czynności był powtarzany 16 razy, kiedy to dwa kolejne pomiary masy dały jednakowy wynik m_{15}=290g i m_{16}=290g, co odpowiadało masie $m_{g16}=m_{16}-m_f$=290g-226g=64g. Całkowity czas trwania prażenia to 4h. Wyniki wszystkich serii pomiarowych zebrano w tabeli nr 29.

Obliczeń grawimetrycznej i wolumetrycznej zawartości wody dokonano zgodnie z poniższymi wzorami[116]:

$$\theta_g = \frac{m_{wody}}{m_{gleby}} = \frac{m_p - m_k}{m_k} \tag{14}$$

$$\theta_v = \frac{V_{wody}}{V_{gleby}} = \frac{\frac{m_{wody}}{\rho_{wody}}}{\frac{m_{nas.gleby}}{\rho_{gleby}}} = \frac{m_{wody}\cdot\rho_{nas.gleby}}{m_{gleby}\cdot\rho_{wody}} = \frac{\theta_g\cdot\rho_{nas.gleby}}{\rho_{wody}} \tag{15}$$

gdzie:

$m_p - masa\ próbki\ przed\ suszeniem$,

$m_k - masa\ próbki\ po\ suszeniu$,

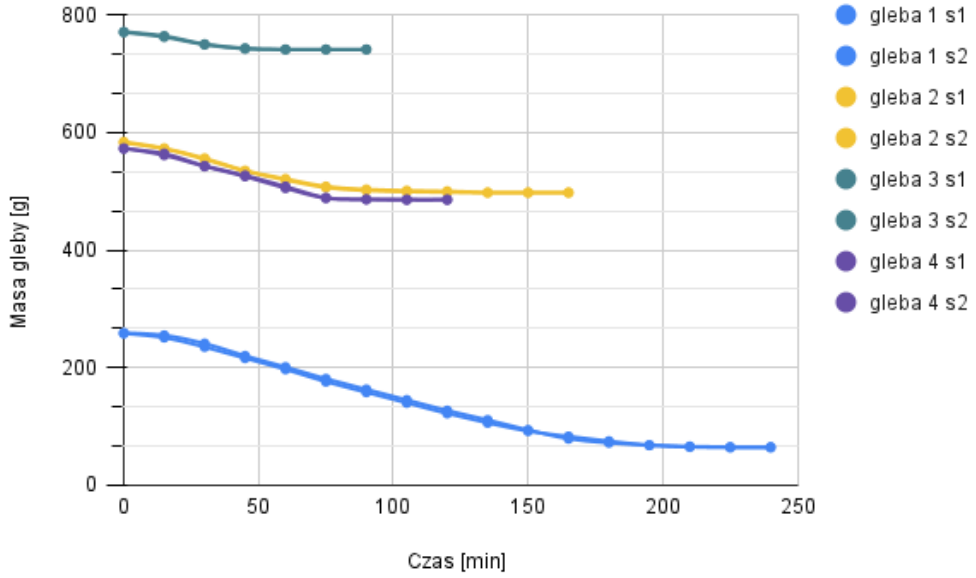
$\rho_{nas.gleby} - gęstość\ nasypowa\ gleby$,

$\rho_{wody} - gęstość\ wody$.

Dla każdego rodzaju ziemi przeprowadzone zostały dwie serie pomiarowe, których wyniki przedstawiono w tabeli nr 29, a ich graficzną reprezentację na rysunku 77. Dla przykładu, uśrednione wyniki obliczeń gęstości nasypowej gleby oraz grawimetrycznej i wolumetrycznej zawartości wody w badanych próbkach przedstawiono w tabelach 30 i 31. Otrzymane wyniki w obrębie danego rodzaju próbki są spójne, różniąc się w zakresie błędu pomiarowego wagi, dlatego też zostały użyte do obliczeń w eksperymencie E1.

Tabela 29: Wyniki pomiarów masy obu rodzajów gleb podczas prażenia w temp. 105°C

Czas [min]	0	15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240
m. gleby nr 1 [g]	258	251	235	216	197	176	158	140	122	106	92	83	75	68	65	64	64
m. gleby nr 1 [g]	259	255	241	220	201	181	163	145	127	111	94	79	72	68	66	65	65
m. gleby nr 2 [g]	582	572	554	534	519	506	501	499	498	497	497	497	-	-	-	-	-
m. gleby nr 2 [g]	584	573	556	535	521	508	503	501	500	498	498	498	-	-	-	-	-
m. gleby nr 3 [g]	770	762	749	742	741	741	741	-	-	-	-	-	-	-	-	-	-
m. gleby nr 3 [g]	771	764	750	743	741	741	741	-	-	-	-	-	-	-	-	-	-
m. gleby nr 4 [g]	572	564	543	527	508	489	486	485	485	-	-	-	-	-	-	-	-
m. gleby nr 4 [g]	573	561	542	525	505	488	486	486	486	-	-	-	-	-	-	-	-



Rysunek 77: Zmiana masy gleby w czasie prażenia w temp. 105°C – eksperyment E2

Na niedokładność wyniku pomiary masy gleby składa się niedokładność wskazań wagi jak również możliwa utrata materiału w trakcie mieszania, szczególnie końcowej fazie kiedy był suchy. Uzyskana stosunkowo niska w stosunku do pozostałych wartość gęstości nasypowej pierwszej gleby na poziomie 0,13±0,01g/cm^3 mieści się w zakresie <0,08;0,75> charakteryzującym gleby organiczne[117].

Tabela 30: Wyniki pomiarów masy i objętości dotyczące prażenia gleby nr 1

Nazwa	**Seria 1**	**Seria 2**	**Seria 3**	**Średnia**
Masa początkowa gleby m_p[g]	301	258	259	-
Objętość początkowa gleby V_p[cm3]	580	500	500	-
Masa końcowa gleby m_k[g]	75	64	65	-
Objętość końcowa gleby V_k[cm3]	260	220	220	-
Gęstość nasypowa gleby [g/cm3]	0,129	0,128	0,130	0,129
Zawartość % wody w glebie (masa)	75,08%	75,19%	74,90%	**75,06%**
Grawimetryczna zawartość wody θ_g[%]	301,33%	303,13%	298,46%	300,97%
Wolumetryczna zawartość wody θ_v[%]	38,65%	38,88%	38,28%	38,60%

Tabela 31: Wyniki pomiarów masy i objętości dotyczące prażenia gleby nr 2

Nazwa	**Seria 1**	**Seria 2**	**Średnia**
Masa początkowa gleby m_p[g]	582	584	-
Objętość początkowa gleby V_p[cm3]	500	500	-
Masa końcowa gleby m_k[g]	497	498	-
Objętość końcowa gleby V_k[cm3]	366	366	-

Gęstość nasypowa gleby [g/cm3]	1,358	1,360	1,36
Zawartość % wody w glebie (masa)	14,60%	14,72%	**14,67%**
Grawimetryczna zawartość wody θ_g[%]	17,10%	17,27%	17,19%
Wolumetryczna zawartość wody θ_v[%]	23,24%	23,47%	23,36%

4.3.3 Wyznaczenie dynamicznej charakterystyki czujników

W tym podrozdziale zostanie określona maksymalna i minimalna wartość pomiarowa czujnika oraz wpływ temperatury na wyniki jego pomiarów.

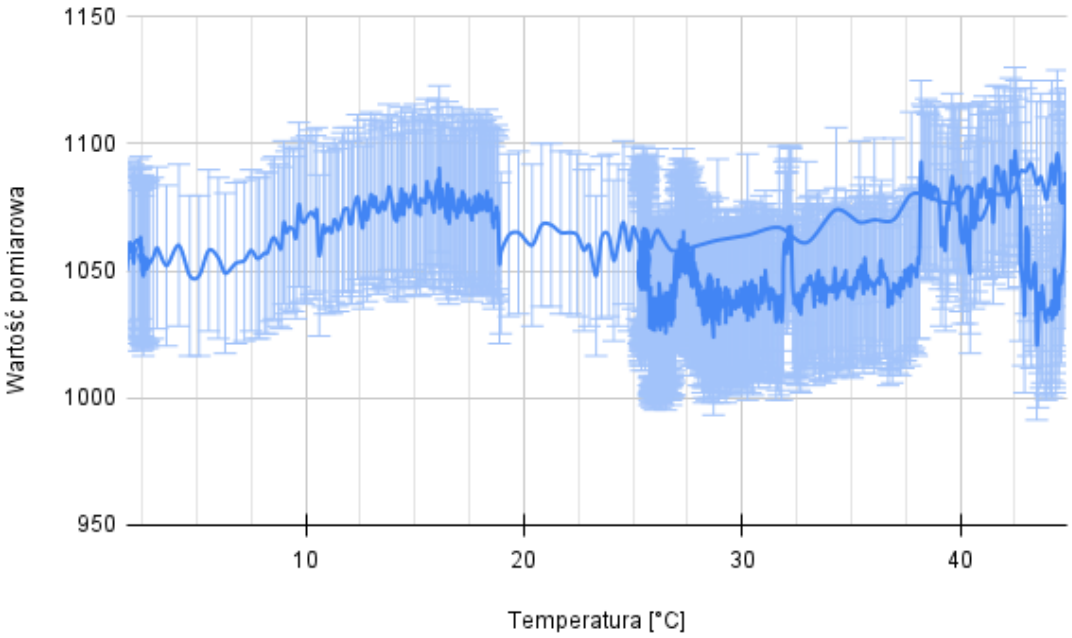
W celu określenia charakterystyki temperaturowej czujnika przeprowadzono pięć doświadczeń pomiarowych. Do pierwszego została użyta wyprodukowana podczas prażenia sucha gleba nr 1,2,3 oraz 4, do drugiego zaś woda z sieci wodociągowej. Uzasadniając, sprawdzenie dotyczyło dwóch ekstremalnych sytuacji dotyczących wilgotności gleby i zachowania wyników pomiarowych podczas zmiany temperatury w zakresie 0-45°C. W obu przypadkach materiał został zamknięty szczelnie w słoikach i schłodzony do temperatury około 0°C. W tych samych warunkach umieszczono czujnik pomiarowy, tak aby zniwelować jego pojemność cieplną po włożeniu do schłodzonych próbek.

Po schłodzeniu czujnik został podłączony do układu pomiarowego i umieszczony w kolejno w poszczególnych rodzajach gleby, a w drugim przypadku w wodzie. Temperatura zewnętrzna wahała się podczas doświadczeń w granicach 25-26°C. Słoik został w swojej górnej części kilkukrotnie owinięty folią spożywczą i zaciśnięty taśmą izolacyjną, w celu spowolnienia wymiany ciepła z otoczeniem, a następnie został umieszczony w pustym naczyniu na zbudowanej wadze. Pomiary odbywały się pod kontrolą układu z rysunku nr 66, z tą różnicą, że w oprogramowaniu zmieniono okres pomiaru na 10s. Wartości były na bieżąco prezentowane na ekranie i zapisywane na kartę pamięci. Przy temperaturze ok. 18-20°C do naczynia, w którym znajdował się słoik, została 3-krotnie dolana woda o temperaturze powyżej 70°C, co ostatecznie spowodowało wzrost temperatury wody w słoiku do 44,9°C. Elektroniczny termometr został trwale przymocowany do górnej części czujnika wilgotności, w której znajdowały się jego zabezpieczone przed wodą elementy elektroniczne (opornik, kondensator i dioda), gdyż temperatura ma wpływ na ich właściwości elektryczne.

Wyniki doświadczeń z każdym rodzajem prażonej gleby uzyskały w czasie wynik na poziomie 4095 niezależnie od temperatury. Jest to największa wartość na skali pomiarowej, oznaczająca brak wykrycia wody w otoczeniu czujnika. Należy więc zauważyć, że jest to wynik spodziewany, gdyż po prażeniu podczas eksperymentu E2 w próbkach są śladowe ilości wody.

Natomiast wyniki doświadczenia z wodą zostały przedstawione na rysunku nr 78, na którym zostały dodane 3% słupki błędów, ze względu na obliczoną wcześniej stabilność wyników. Zarejestrowano 1035 pomiarów w czasie 8h10m. Z otrzymanej zależności nie można wyciągnąć jednoznacznego wniosku o zależności wyników pomiarów od temperatury w zakresie 0-45°C. Uzyskana wartość minimalna to 1022, średnia 1052 natomiast maksymalna

to 1097. Górna część wykresu odpowiada za fazę zrostu temperatury, natomiast dolna część (przedział 45-25°C) obrazuje fazę spadku temperatury.

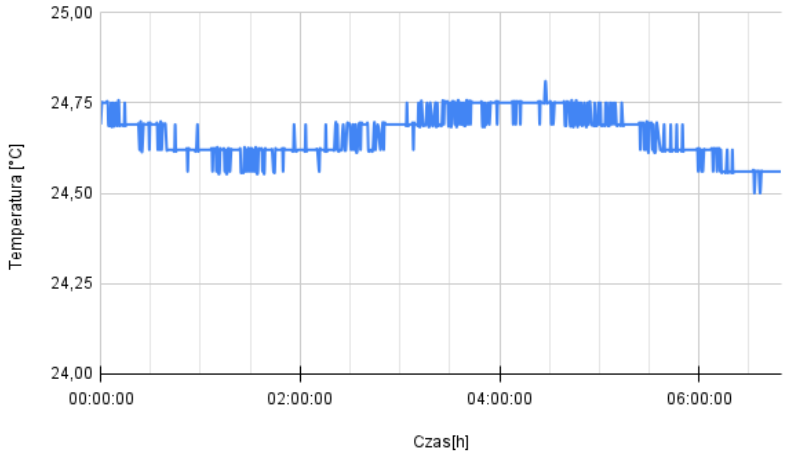


Rysunek 78: Zależność wyniku pomiaru od temperatury w zakresie 0-45°C dla wody

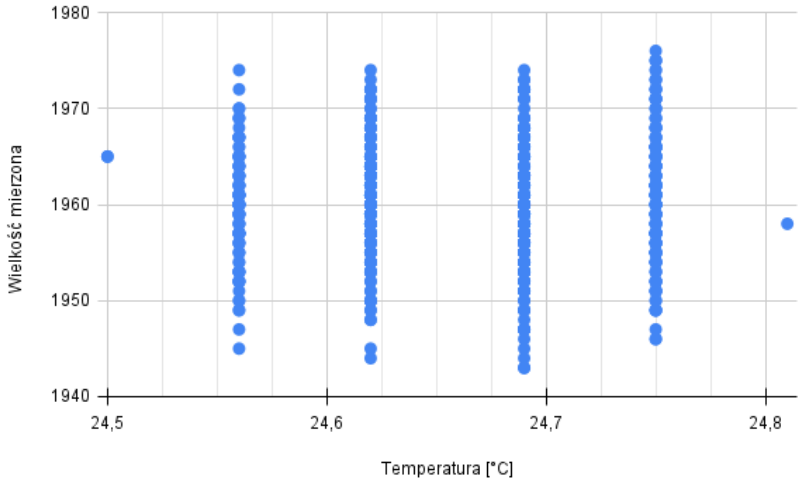
4.3.4 Określenie stabilności pomiarów czujnika wilgotności w temperaturze 24-25°C.

Eksperyment E1, w którym wykorzystano glebę nr 2, po dostatecznym wysuszeniu, nie został przerwany, lecz próbka została pozostawiona do wychłodzenia i osiągnięcia temperatury pokojowej. Lampa grzewcza została wyłączona, celem ograniczenia parowania i przez kolejne 6.5h kontynuowano pomiary. Opisane działanie miało na celu umożliwienie zebrania danych w nieznacznie zmieniającej się temperaturze i określenie stabilności wyników pomiarów. Dla próby 783 pomiarów ostatniej fazy, przy temperaturze $24{,}66 \pm 0{,}06\,°C$ uzyskano dane pomiarowe scharakteryzowane statystyką przedstawiona w poniższej tabeli nr 32. Wynika z nich, że dla obu czujników przy tej temperaturze maksymalny rozrzut wielkości mierzonej stanowi ±7-8%, natomiast w ujęciu odchylenia standardowego jest to około ±3%. W rzeczywistości wartości te są jeszcze mniejsze, gdyż nie uwzględniają ubytku masy wody. Zobrazowanie temperatury i wielkości mierzonej od czasu umieszczono na rysunkach nr 79 i 81, natomiast ich wzajemne powiązanie na rysunku nr 80. Dyskretne wartości temperatury wynikają z faktu, iż wielkość ta była mierzona termometrem cyfrowym (model DS18B20, o 12 bitowej rozdzielczości).

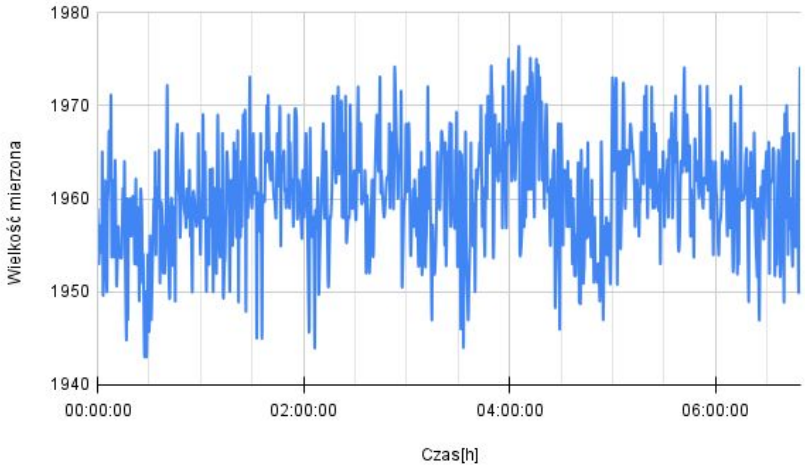
Z wyników tego eksperymentu wyciągnąć można wniosek, że implementując punkt wyzwalania nawadniania przez sterownik, należy uwzględnić co najmniej 3% margines błędu wynikający ze stabilności pomiaru, której wartość dla danej gleby odzwierciedla VWC.



Rysunek 79: Zależność temperatury od czasu dla gleby nr 1 – faza końcowa



Rysunek 80: Wielkość mierzona dla zarejestrowanych temperatur



Rysunek 81: Zależność wielkości mierzonej od czasu dla gleby nr 1 – faza końcowa

Tabela 32: Statystyka pomiarów przy temperaturze zbliżonej do stałej

Statystyka	czujnik SoilWatch	czujnik autorski	Masa[g]
Wartość maksymalna	1635	1976	1174,45
Wartość minimalna	1612	1943	1168,37
Rozstęp	23	33	6,08
Średnia arytmetyczna	1622,6	1960,31	1171,03
Odchylenie standardowe	5	6	1

4.4 Podsumowanie etapu doświadczalnego

W wyniku przeprowadzonych doświadczeń pomiarowych określono zakres stosowalności oraz charakterystykę ze względu na zawartość wody w podłożu dla bezprzewodowego czujnika wilgotności. Stabilność wyników dla ustalonej temperatury określono na 3%. Zakres przebadanych temperatur to 0-44°C. Szczególne punkty z zakresu pomiarowego zostały zaś przedstawione w tabeli nr 33.

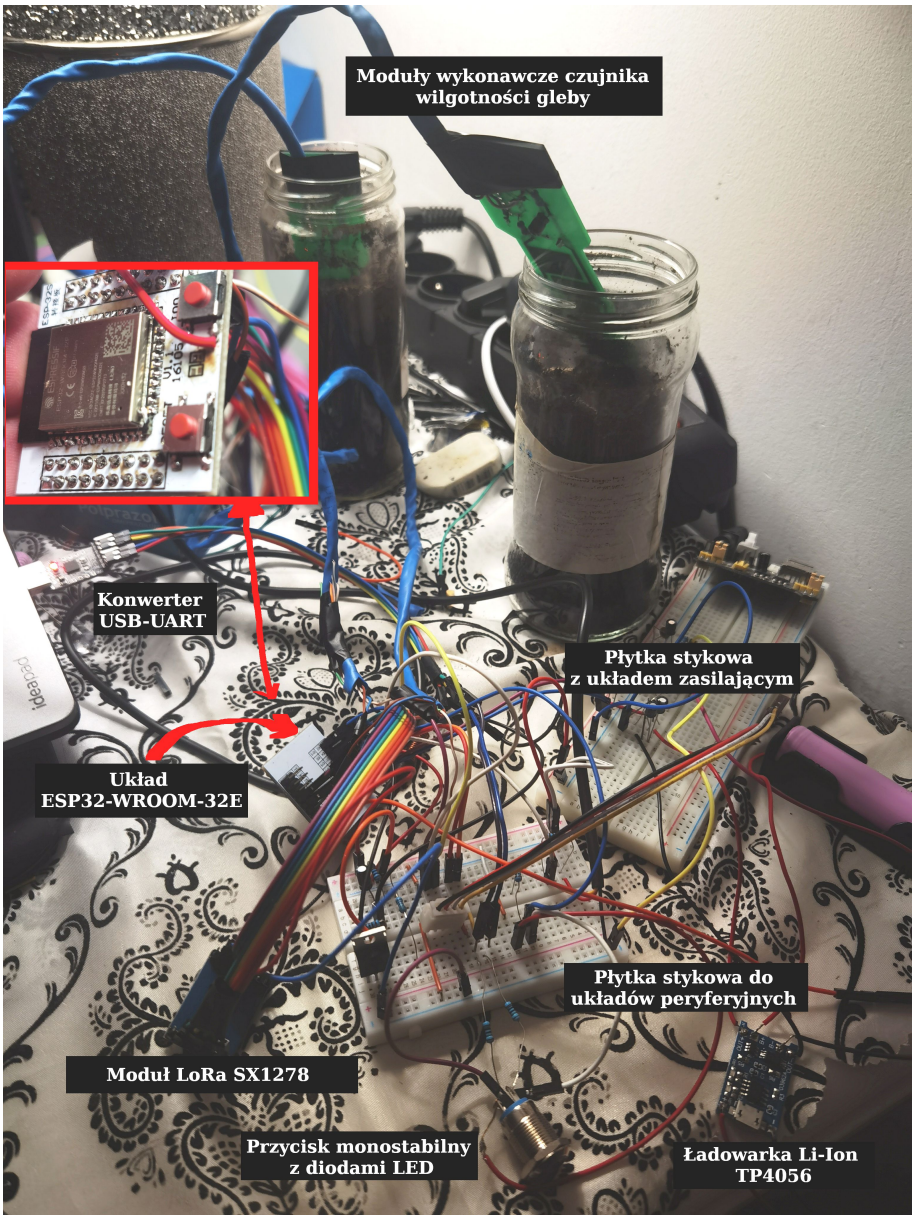
Tabela 33: Zestawienie zakresu pomiarowego i poziomów wyzwalania nawadniania dla elementu wykonawczego budowanego czujnika wilgotności gleby

Rodzaj materiału otaczającego czujnik	VWC[%]	Wynik pomiaru
Maksymalna wartość: **sucha gleba, powietrze**	0%	4095
Wyzwalanie nawadniania dla podłoża typu: **drobnoziarnisty piasek**	6-8%	2944
Wyzwalanie nawadniania dla podłoża typu: **czarnoziem, glina pylasta** i ich mieszaniny	25-27%	1492
Wyzwalanie nawadniania dla podłoża typu: **torf**	44-46%	1347
Minimalna wartość: **woda**	100%	1052
Teoretyczna minimalna wartość pomiarowa	-	0

Z analizy wyników pomiarów wszystkich eksperymentów, a także z ich porównania do czujnika wzorcowego, wynika, iż zbudowane urządzenie może posłużyć do określania punktu wyzwalania systemu nawadniania. Ta graniczna wartość zależy zaś od rodzaju podłoża. Zmieniająca się zawartość wody w glebie wpływa proporcjonalnie na zmianę pasożytniczej pojemności elektrycznej obwodu drukowanego, stanowiącej element pomiarowy czujnika. W zbudowanym urządzeniu układ pomiarowy zasilany był impulsami napięcia o maksymalnej amplitudzie około 3.3V i pozwolił na określenie z około 7% dokładnością objętościowej zawartości wody w glebie (VWC, *volumetric water content*). Wobec powyższego hipoteza badawcza została potwierdzona.

5. BUDOWA URZĄDZEŃ ORAZ IMPLEMENTACJA FUNKCJONALNOŚCI WRAZ Z FAZĄ TESTOWANIA

Po opracowaniu schematów elektrycznych, ale przed wykonaniem płytek PCB, zostały one zrealizowane na płytkach stykowych przy wykorzystaniu dedykowanych przewodów połączeniowych. Pozwoliło to na weryfikację założeń projektowych oraz działanie wyspecyfikowanych bibliotek programistycznych do obsługi modułów elektronicznych. Przykładowe zdjęcie takiej realizacji czujnika, w trakcie pomiarów wilgotności gleby na zasilaniu z akumulatora przedstawia poniższy rysunek 82. Szczególnie ważne było, sprawdzenie układu



Rysunek 82: Realizacja czujnika na płytkach stykowych

ESP32, który w przedstawionej powyżej aplikacji pracował nie na płytce rozwojowej, lecz samodzielnie, w minimalnie wymaganej konfiguracji sprzętowej (konfiguracja przycisku reset, kondensator C7 z rys. 62).

Analogiczna realizacja dotyczyła sterownika nawadniania, podczas której szczególny problem sprawił 16-kanałowy ekspander wyprowadzeń MCP23017, zasadniczy komponent pozwalający zapewnić sterownikowi wielostrefowość. Nieprawidłowość w jego działaniu objawiała się przez to, że skutecznie można było tylko jednokrotnie zmienić stan jego wyprowadzeń. Podjęto nawet próbę zmianę biblioteki programistycznej, które okazała się bezskuteczna. Ostatecznie ponowna analiza karty katalogowej[118] doprowadziła do źródła problemu, którym było niepodłączone wyprowadzenie $\overline{\text{RESET}}$ ekspandera. Dopiero trwałe podanie logicznej wartości 1 doprowadziło do prawidłowej pracę opisywanego układu.

Innym zidentyfikowanym problemem był błąd z komunikatem *„Brownout detector was triggered”* po którym układ ESP32 resetował się. Zaobserwowano, że pojawia się on podczas wykorzystywania modułu radiowego WiFi, który podczas transmisji może pobierać wartość szczytową natężenia prądu 379mA, z wartością średnią na poziomie 239mA[38]. Objawy występowały zarówno na zasilaniu akumulatorowym jak przy wykorzystaniu zasilacza sieciowego (moduł zasilający płytek stykowych, widoczny na rys. 82 nad akumulatorem). Stabilne działania zostało przywrócone przez trwały montaż kondensatora C9(rys. 61) w pobliżu linii zasilających ESP32, tak jak przewidywał to projekt PCB. Poprzednie umieszczenie na płytce stykowej w odległości ponad 20cm było w istocie błędnym uproszczeniem.

Jednym z wprowadzonych udogodnień, na bazie doświadczenia z budowy czujnika na płytkach stykowych, było oklejenie istotniejszych przewodów samoprzylepnymi etykietami z opisem ich znaczenia. To wbrew pozorom mało istotne w całym przedsięwzięciu działanie, ułatwiło i przyspieszyło wielokrotnie przełączanie przewodów połączeniowych, minimalizując ryzyko popełnienia błędu oraz porządkując przestrzeń wokół urządzenia. Innym stałym elementem organizacyjnym stało się dokumentowanie zmian w obwodzie przed dłuższą przerwą lub końcem pracy danego dnia. Odbywało się to w postaci krótkich notatek, a czasem zdjęć, co przyspieszało ponowne podjęcie tematu podczas kolejnej sesji pracy jak również ułatwiało wycofanie się do poprzedniej konfiguracji połączeń. Dopiero po weryfikacji działania wszystkich komponentów zarówno sterownika jak i czujnika strefowego oraz wprowadzeniu korekt do schematów elektrycznych przystąpiono do kolejnego etapu prac.

5.1 Budowa platformy sprzętowej

Wszystkie projekty obwodów drukowanych zostały zlecone do wykonania firmie JLCPCB[119] z Shen Zhen w Chinach. Obejmowało to cztery dostawy: prototypy układu wykonawczego (rys. 84), finalny układ wykonawczy, obwody drukowane czujnika wilgotności oraz płytkę PCB sterownika. Czas realizacji zamówienia, był zgodny z deklarację firmy i w żadnym z przypadków nie przekroczył dwóch tygodni od daty zamówienia. Natomiast sam proces zamówienia polegał na wgraniu do internetowego formularza pliku projektowego *.gerber* (eksport z aplikacji EasyEDA[96]) oraz wybrania właściwości tworzonej płytki. Dla wszystkich projektów były one takie same i zostały zebrane w poniższej tabeli nr 34. Ostatecznie wybranie zagranicznego wykonawcy do realizacji zlecenia był dobrym wyborem: pomimo nieznacznie droższej opłaty

za wysyłkę oraz terminem dostawy dłuższym o około tydzień, ponosząc porównywalne koszty realizacji uzyskano 5 sztuk każdej z płytek zamiast jednej.

Tabela 34: Parametry zamawianych płytek PCB

Nazwa	Wartość
Materiał bazowy	FR-4
Typ materiału	FR4 - standardowa T_g 130-140°C
FR-4	FR-4
Ilość warstw	2
Grubość miedzi	1 Oz (1.4Mils)
Ilość PCB*	5
Grubość PCB	1.6mm
Kolor PCB	Zielony
Kolor warstwy opisowej	Biały
Wykończenie powierzchni:	HASL (z ołowiem)
Jakość wyglądu:	IPC Klasa 2 Standard

*minimalna ilość zamówienia to 5szt

Montaż elementów zarówno przewlekanych jak i powierzchniowy został wykonany przez autora pracy, przy użyciu cyfrowej stacji lutowniczej ZHAOXIN 936DH o mocy 75W. Urządzenie posiada regulację temperatury w zakresie od 200°C do 480° i odczyt temperatury z grota, co było bardzo pomocne i pozwoliło przeprowadzić lutowanie nie przekraczając temperatur określonych w kartach katalogowych elementów elektronicznych. Największe wyzwanie stanowił układ główny ESP32-WROOM-32E, ze względu na odległość między polami lutowniczymi 0.37mm z szerokością szerokość pola 0.9mm (rys. 83). Jednakże zastosowanie uchwytu montażowego wraz ze szkłem powiększającym, praktyka uzyskana podczas lutowania układów do weryfikacji (rys. 82) pozwoliły na pomyślne ukończenie zadania. Zgodnie z dokumentacja[38] 9 pól lutowniczych umieszczonych pod tym modułem wraz z sugerowanymi przelotkami, służy do odprowadzania ciepła, dlatego też zastosowano na tym obszarze pastę termoprzewodzącą, wobec braku możliwości lutowania (rys. 83). W sterowniku nawadniania pod ekspander wyprowadzeń zastosowano podstawkę DIP28 celem umożliwienia wymiany układu w przypadku awarii/uszkodzenia. Pozostałe elementy, poza peryferiami podłączanymi przez złącza JST, zostały przylutowane bezpośrednio do płytki PCB. Kolejnym przeprowadzonym etapem była weryfikacja poprawności montażu. Przy użyciu miernika uniwersalnego zostały sprawdzone wszystkie ścieżki zasilania i sygnałowe. Wykryto nieciągłość na jednej ze ścieżek z powodu niewłaściwego lutu i została ona niezwłocznie poprawiona.

Na kolejnych rysunkach 86, 87 przedstawiono wykonane układy sterownika nawadniania i czujnika wilgotności wraz z podłączonymi peryferiami, poza elementem wykonawczym czujnika zaprezentowanym na rysunku nr 85.

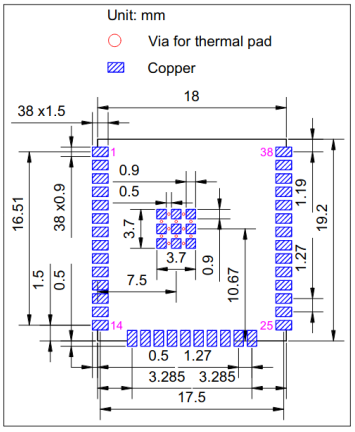
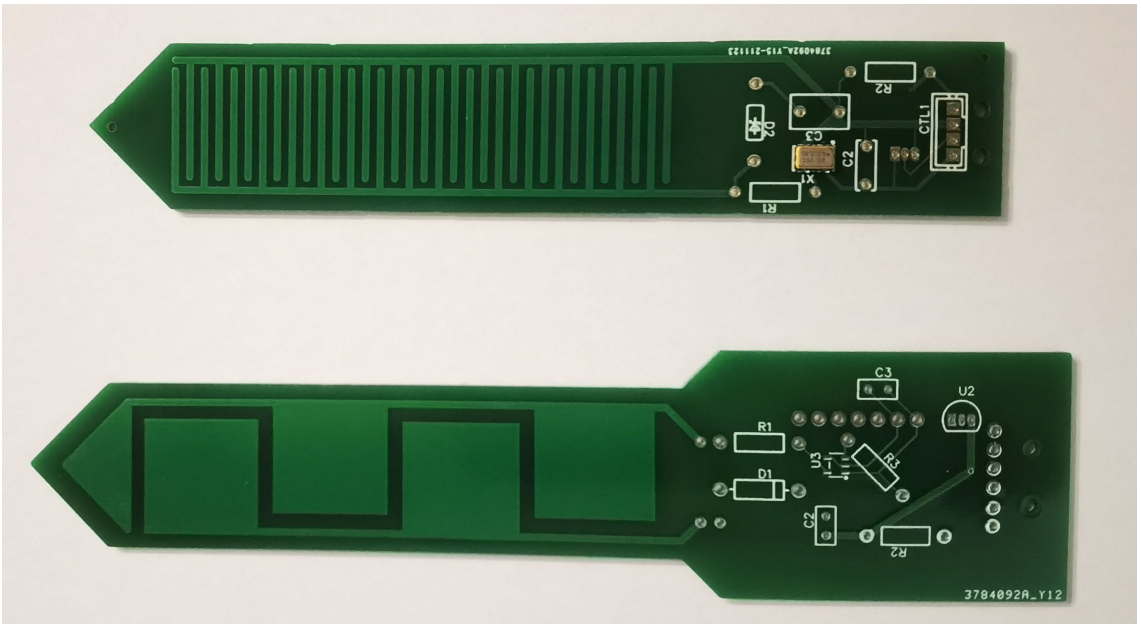
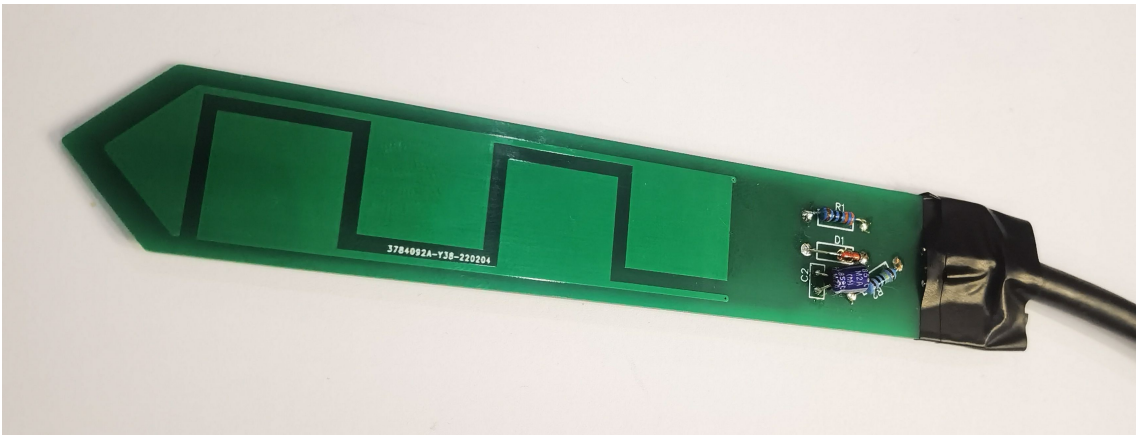


Figure 10: ESP32-WROOM-32UE Recommended PCB Land Pattern

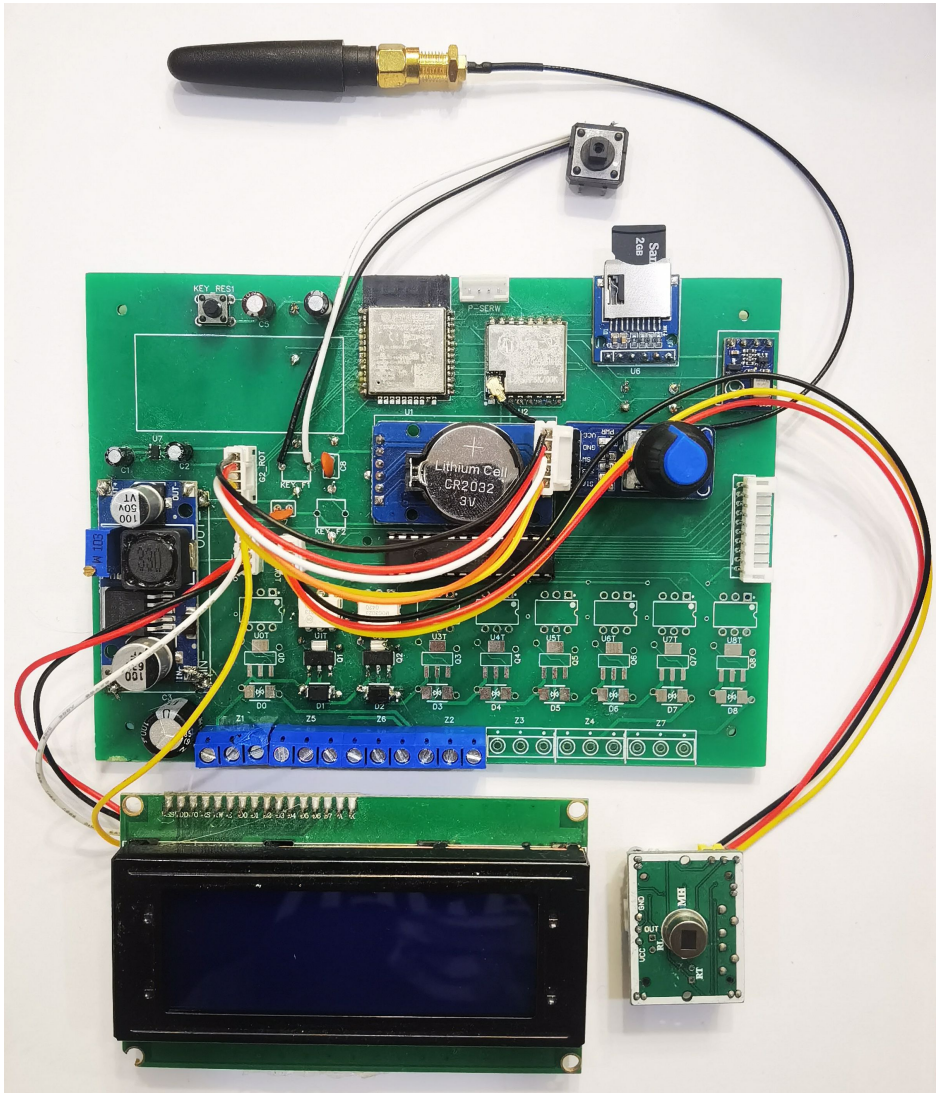
Rysunek 83: Wzór pól lutowniczych ESP32-WROOM-32E[38]



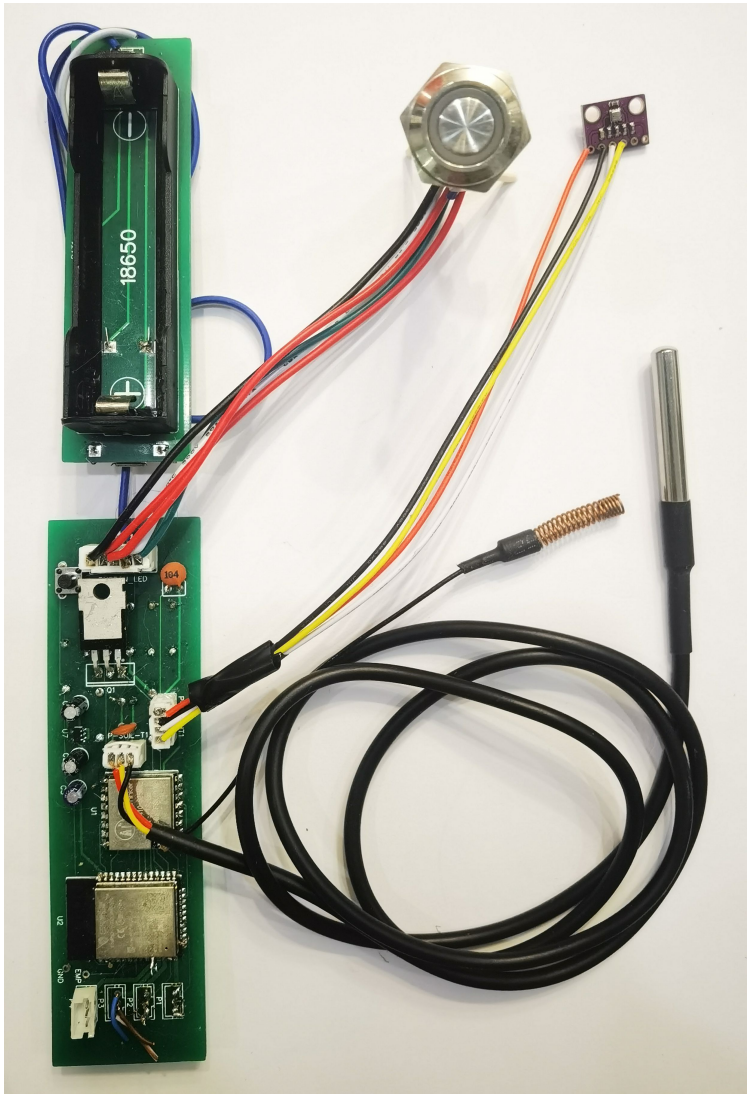
Rysunek 84: Prototypowe elementy wykonawcze czujnika wilgotności



Rysunek 85: Finalny elementy wykonawczy czujnika wilgotności



Rysunek 86: Sterownik nawadniania



Rysunek 87: Czujnik wilgotności

5.2 Implementacja oprogramowania

Na wstępie, w formie poniższej tabeli, zostały zebrane wszystkie zrealizowane prace implementacyjne z podziałem na użyty język programowania. Wyróżnione pozycje stanowią produkty projektu, natomiast liczby z ostatniej kolumny określają kolejność realizacji.

Tabela 35: Języki programowania i skryptowe użyte w implementacji

Lp.	Język	Wersja	Miejsce zastosowania
1.	C++	C++11	1) Układ do pomiaru czasu rozładowania kondensatora 2) Układ do przeprowadzania doświadczenia pomiarowego **3) Strefowy czujnik wilgotności** **6) Sterownik nawadniania**
2.	JavaScript (*client-side*)	ECMAScript 2017	4) Symulacja algorytmu OptiServ
3.	JavaScript (*server-side*)	ECMAScript 2015	5) Przeglądarka wyników symulacji OptiServ
4.	PHP	PHP 7.4.30	2) Zbieranie wyników doświadczenia pomiarowego 5) Przeglądarka wyników symulacji OptiServ
5.	Java	openjdk 17.0.4	**7) Aplikacja mobilna na platformę Android**

5.2.1 Implementacja oprogramowania sterownika i bezprzewodowego czujnika wilgotności

Dzięki stworzonym układom prototypowym na płytkach stykowych i skutecznym sprawdzeniu na nich obsługi układów peryferyjnych przez biblioteki programistyczne, właściwą implementację oprogramowania prowadzono na stabilnej platformie sprzętowej. Działanie to miało kluczowe znaczenie, gdyż ułatwiało debugowanie kolejnych wersji oprogramowania, zakładając, że błędy zostały popełnione przez programistę, a nie wynikają z niewłaściwego działania danego komponentu np. z powodu niewłaściwego zasilania, niewłaściwej konfiguracji sprzętowej lub nietrwałych połączeń.

Wykorzystanie każdej z bibliotek programistycznych (kompletna lista znajduje się na końcu pracy w spisie bibliotek) poprzedzało zapoznanie się z jej licencją, instrukcją, przykładami oraz zgłoszonymi i nierozwiązanymi jeszcze problemami (ang. *issues*) w repozytorium serwisu Github. Kilkukrotnie wymagane jednak było sięgniecie do kodu źródłowego czy to w plikach nagłówkowych czy właściwych z implementacją, ze względu na powstałe niejasności lub zbyt skromny opis czy instrukcję użycia danej biblioteki.

Kod źródłowy został podzielony na pliki o nazwach odzwierciadlających funkcje w nim zawarte, co wraz wielkością plików oraz opisem zostało przedstawione w poniższej tabeli nr 36. W związku z tym podziałem umiejętnie stosowano dyrektywę `#include` do włączania plików nagłówkowych oraz słowo kluczowego `extern` do określenia definicji zmiennych zdefiniowanych w innych plikach. W pętli głównej programu w trybie ciągłym następuje sprawdzanie stanu przycisków i pokrętła, oraz odczytu licznika impulsów PCNT z czujnika przepływu cieczy (o ile wybrana jest opcja, że jest on podłączony). Co 15 minut następuje aktualizacja prognozy pogody,po której obliczane są średnia temperatura i wilgotność,

oczekiwana wartość opadu oraz ewapotranspiracja, za ilość godzin, którą zdefiniowano w ustawieniach (domyślnie 3h). Natomiast co 60 sekund wykonywana jest główna pętla algorytmu OptiServ, której schemat blokowy przedstawiono na rysunkach 27-29.

Tabela 36: Struktura katalogu z plikami kodu źródłowego sterownika

Lp.	Nazwa pliku	Rozm. [B]	Opis
1	ConfigManager.cpp	4387	Inicjalizacji trybu konfiguracji do wprowadzenia danych dostępowych do sieci bezprzewodowej
2	custom_char.h	1582	Definicja ikon do wyświetlania na ekranie LCD
3	data_logger.cpp	7620	Metody do obsługi systemu plików (m. in. tworzenie plików i folderów, odczyt i zapis plików, informacje o wolnej przestrzeni), zapis zdarzeń na kartę pamięci, odczyt ostatnich linii z pliku
4	data_logger.h	885	
5	date_time.cpp	2583	Inicjalizacja zegara RTC oraz klienta NTP, metody do aktualizacji czasu
6	date_time.h	235	
7	global.cpp	532	definicje identyfikatorów pomocniczych przy użyciu dyrektywy #define
8	global.h	1638	
9	helpers.cpp	4454	Metody pomocnicze dotyczące formatowania daty i czasu oraz statystyczne (min., max., mediana i wart. średnia)
10	helpers.h	807	
11	lcd.cpp	14168	Obsługa ekranu, jego odświeżania, pokrętła z przyciskiem oraz przycisków funkcyjnych; inicjalizacja i obsługa wiadomości MQTT
12	lcd.h	1228	
13	lora.cpp	9348	Odbiór, przetwarzanie i wysyłka komunikatów z protokołu LoRa, szyfrowanie i deszyfrowanie, tworzenie i obsługa kolejki, ustalanie numeru strefy nadawcy komunikatu, aktualizacja danych środowiskowych strefy
14	lora.h	479	
15	main.cpp	7955	Deklaracja i inicjalizacja struktur danych i zmiennych pomocniczych, konfiguracja wyprowadzeń ESP32, inicjalizacja układów peryferyjnych, główna pętla programu
16	optiserv.cpp	25835	Implementacja algorytmu OptiServ i jego metod pomocniczych
17	optiserv.h	2259	
18	poulseCounter.cpp	5772	Monitorowanie przepływu cieczy przez czujnik przepływu
19	poulseCounter.h	277	
20	settings.cpp	8576	Odczyt i ładowanie do pamięci / zapis plików konfiguracyjnych JSON sterownika oraz czujników wilgotności
21	settings.h	1937	
22	weatherForecast.cpp	4948	Pobieranie prognoz pogody, filtrowanie wyników, obliczanie średniej temperatury i wilgotności oraz ewapotranspiracji
23	weatherForecast.h	152	
24	webServer.cpp	22567	Inicjalizacja serwera HTTP, definicja i implementacja węzłów końcowych, obsługa wyjątków
25	webServer.h	306	

Do odświeżania zawartości ekranu dochodzi:

a) co 60 sekund, w przypadku gdy aktualna sekunda wynosi 0,

b) w przypadku użycia pokrętła, do aktualizacji położenia kursora,

c) w przypadku zmiany stanu strefy nawadniania za pomocą przycisku w pokrętle,

d) w przypadku zmiany stanu strefy nawadniania, zmiany trybu pracy, zmiany statusu alarmu których źródłem komendy jest aplikacja mobilna.

Pierwsze z trzech wymienionych scenariuszy zostały obsłużone przy użyciu przerwań. W przypadku a) zegar RTC został skonfigurowany, aby o każdej pełnej minucie wyzwalał alarm tzn. podnosił stanu linii SQW (rys. 62) do wysokiego, a następnie do niskiego. Taką zmiana jest wykrywana, dzięki temu, że do wejścia z linii SQW (GPIO 32) dołączono przerwanie ustawione na wykrywanie zbocza opadającego sygnału. Następuje wtedy wywołanie wcześniej zdefiniowanej procedury obsługi przerwania (ISR) w której zmiennej opisującej konieczność odświeżenia ekranu przypisywana jest wartość prawdziwa. Ta oraz inne zmienne, których wartość jest zmieniania w ciele procedury ISR zostały podczas deklaracji opatrzone modyfikatorem `volatile`, aby nie doszło jej optymalizacji przez kompilator, a jej wartość zawsze była z pamięci operacyjnej zamiast z rejestru pamięci[98]. Podczas definicji procedur ISR użyto słowa kluczowego `IRAM_ATTR`, wymaganego do umieszczenia danych w pamięci IRAM i zapewnienia do nich najkrótszego możliwego czasu dostępu[39]. Innym miejscem, gdzie zostały wykorzystane przerwania to obsługa sygnału z czujnika ruchu do włączania i wyłączania podświetlenia ekranu oraz wyzwalania alarmu w przypadku zazbrojenia strefy.

Poza wyżej wymienionymi, dzięki poprowadzeniu linii DIO z modułu LoRa do GPIO 15 (ESP32) wykorzystano funkcjonalność dostarczaną przez bibliotekę Arduino-LoRa do obsługi wiadomości przychodzących. Wykorzystano wywołanie zwrotne (ang. *callback*) w którym nastąpiła konsumpcja poszczególnych bajtów komunikatu, aż do uzyskania jego całości. Z racji że operacje wewnątrz procedur ISR powinny być jak najkrótsze (blokują wykonanie programu głównego), nie można było w wywołaniu zwrotnym odszyfrować wiadomości i poddać dalszemu przetwarzaniu. Dlatego też zdefiniowana została kolejka wiadomości do której dodawane są komunikaty i procedura wykonywana podczas przerwania jest opuszczana. Dalsze przetwarzanie wiadomości następuje w pętli głównej programu i obejmuje jej odszyfrowanie, ustalanie numeru strefy nadawcy komunikatu, aktualizacja danych środowiskowych strefy źródłowej, a następnie odesłanie odpowiedzi do czujnika wilgotności z czasem następnego raportu i aktualną wersją konfiguracji (zgodnie z diagramem sekwencji z rys. nr 39).

Korzystając z funkcjonalności biblioteki ESPAsyncWebServer utworzono instancję serwera www, działającego asynchronicznie, nasłuchującego na porcie numer 80. Następnie zdefiniowano i zaimplementowano węzły końcowe świadczonej usługi (zgodnie z tabelą nr 16), umożliwiając komunikację z potencjalnym klientem. Istotne jest, że obsługa przychodzących żądań zachodzi w tle, co nie powoduje przerwania programu głównego, dzięki wykorzystaniu przez bibliotekę dwurdzeniowej architektury ESP32. Przed obsłużeniem każdego żądania sprawdzane jest czy klient został wcześniej uwierzytelniony, używając zaimplementowanego w bibliotece szkieletu *HTTP authentication*[120] w schemacie podstawowym. Obsłużony został również kod odpowiedzi HTTP 404, dotyczący próby wywołania żądania o niezdefiniowanej nazwie, przez odesłanie do klienta stosownego komunikatu.

Z racji iż za podstawowy format wymiany i przechowywania danych w projekcie został wybrany JSON, wielokrotnie wykorzystano zoptymalizowane metody biblioteki ArduinoJSON do przetwarzanie dokumentów tego rodzaju. Najznamienitszym przykładem stało się pobieranie

danych z serwisu pogodowego, w którym dostarczany jest plik JSON zawierające bogaty zbiór danych metrologicznych, które dzięki ArduinoJSON mogą w locie zostać odfiltrowane w celu uzyskania wymaganych danych. Oznacza to że, wystarcza alokacja zasobów pamięciowych kilkukrotnie mniejsza od pobieranego dokumentu. Na poniższym listingu zaprezentowano tworzenie filtra, zaczepionego do węzła *hourly,* definiującego pożądane składniki czas(*dt*), temperaturę, wilgotność, prawdopodobieństwo opadów oraz ich wielkość. Następnie do wcześniej zdefiniowanego dokumentu *doc* następuje próba deserializacji odpowiedzi, pochodzącej od źródła *client,* odpowiedzialnym za zarządzanie połączeniami sieciowymi. Tak zdeklarowana lokalnie struktura pamięci typu `DynamicJsonDocument` jest w dalszej kolejności poddawana iteracji w celu uzyskania i wykorzystania pobranych wartości do obliczenia wartości średnich i oczekiwanych. Warto uzupełnić, że alokacja obejmowała 6.144kB pamięci, a wielkość przetwarzanego pliku jest z przedziału 30-33KB.

Procedura 6: Filtrowanie danych z serwisu meteorologicznego

```
WiFiClient client;
(...)
DynamicJsonDocument doc(6144);
(...)
JsonObject filter_hourly_0 = filter["hourly"].createNestedObject();
  filter_hourly_0["dt"] = true;
  filter_hourly_0["temp"] = true;
  filter_hourly_0["humidity"] = true;
  filter_hourly_0["pop"] = true;
  filter_hourly_0["rain"] = true;
DeserializationError error = deserializeJson(doc, client,
DeserializationOption::Filter(filter));
```

W celu podniesienia wrażeń wizualnych oraz reprezentacji informacji na ograniczonej przestrzeni ekranu 20x4 znaków, zaprojektowano ikony o wielkości 8x5 pixeli, dotyczące statusów sterownika oraz pomocnicze dotycząca znaku stopni Celcjusza oraz oznaczenia adresu IP. Widok ekranu urządzenia w trybach konfiguracyjnym i normalnym przedstawiono na poniższych rys. 88 i 89.

Rysunek 88: Ekran sterownika w trybie konfiguracji

Rysunek 89: Ekran sterownika w normalnym trybie pracy

Ekran w trybie normalnym (rys. 89) został podzielony na trzy obszary:

a) statusy komponentów i stany funkcjonalności systemu (ikony w lewym górnym rogu ekranu: 1. tryb automatyczny/manualny 2. status połączenia z siecią bezprzewodową 3. status karty pamięci 4. stan uzbrojenia alarmu 5. status instalacji nawadniania),

b) data i godzina wraz z temperaturą w pomieszczeniu (prawa część ekranu),

c) status stref nawadniania (dolna część ekranu).

W trybie konfiguracji (rys. 88) w drugiej linijce ekranu wyświetla się identyfikator sieci bezprzewodowej (SSID), a w kolejnej adres IP, pod który trzeba nawigować w przeglądarce internetowej, zgodnie z warunkami z tabeli nr 15. Zrzut ekranu z przeglądarki na urządzeniu mobilnym przedstawia rysunek nr 90.

Rysunek 90: Interfejs użytkownika w trybie konfiguracji

Przechodząc do implementacji oprogramowania w czujniku wilgotności, to strukturę katalogu z plikami z kodem źródłowym wraz z opisem przestawiono w tabeli nr 37.

Tabela 37: Struktura katalogu z plikami kodu źródłowego strefowego czujnika wilgotności

Lp.	Nazwa pliku	Rozm. [B]	Opis
1	helpers.cpp	1823	Metody pomocnicze z statystyczne (min., max., mediana i wart. średnia) oraz sprawdzające i walidujące wyniki pomiarów
2	helpers.h	596	
3	main.cpp	19533	Deklaracja i inicjalizacja struktur danych i zmiennych pomocniczych, konfiguracja wyprowadzeń ESP32, inicjalizacja układów peryferyjnych w tym modułu LEDC, główna pętla programu zgodnie z maszyną stanów z załącznika C; odbiór, przetwarzanie i wysyłka komunikatów z protokołu LoRa, szyfrowanie i deszyfrowanie
4	utlis.cpp	6211	Odczyt i zapis klucza kryptograficznego do pamięci nieulotnej, definicja metod inicjalizacji peryferiów; metoda do obliczania średniej wartości pomiaru z wybranego wejścia z N próbek, ustalanie przyczyny wyjścia ze stanu głębokiego uśpienia
5	utils.h	1281	

Podstawową różnicą w sposobie działania czujnika w stosunku do sterownika jest brak ciągłego wykonywania programu głównego i przechodzenie w stan głębokiego uśpienia. Możliwość wyjścia z tego stanu przy pomocy zmiany stany przycisku fizycznego oraz definicja okresu wybudzeń została zrealizowane przez poniższe instrukcje:

Procedura 7: Definicja portu przycisku wybudzającego z uśpienia oraz jego czasu

```
esp_sleep_enable_ext0_wakeup(GPIO_NUM_2,1);
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * 1000000);
```

W pierwszej z nich następuje definicja wyprowadzania, którego stan będzie monitorowany w trakcie uśpienia oraz, że ma się do dokonać po przejściu w stan wysoki (drugi argument). W drugiej instrukcji zaś żądany czas uśpienia aplikowany jest jako argument po konwersji na mikrosekundy. Komenda egzekwująca zmianę stanu, wydawana po wykonaniu pomiarów oraz otrzymaniu odpowiedzi ze sterownika ma postać:

```
esp_deep_sleep_start();
```

Kluczowa dla czujnika wilgotności gleby jest realizacja pobudzenia jego elementów wykonawczych. Jest ona realizacją uogólnionego sposobem podanego na końcu podrozdziału 1.3.2 i przedstawia się następująco:

Procedura 8: Pobudzenie elementów wykonawczych

```
01. #define PWMChannel1 0
02. #define PWMChannel2 1
03. #define PWMChannel3 2
04. uint8_t SQ1 = 12;
05. uint8_t SS1 = 13;
06. Settings CFG;
07. CFG.loadFromJSON();
08. (...)
09. if(ledcSetup(PWMChannel1, 40000000, 1))
```

```
10.   ledcAttachPin(SQ1, PWMChannel1);
11. (...)
12. if(CFG.SH_OPT!=2 && CFG.SH_OPT!=4) {
13.   ledcWrite(PWMChannel1,1);
14.   ledcWrite(PWMChannel2,0);
15.   ledcWrite(PWMChannel3,0);
16.   delay(1000);
17.   vSS1 = readAVG(SS1,20000);
18.   ledcWrite(PWMChannel1, 0);
19. }
```

W pierwszych trzech wierszach do identyfikatorów `PWMChannelX` następuje przypisanie wartości oznaczających numery kanałów high-speed modułu LEDC. Wiersze 4 i 5 definiują wyprowadzenia GPIO na których odpowiednio będzie zachodziło generowanie impulsów prostokątnych oraz pomiar napięcia (oznaczenia SQ1 i SS1 są zgodnie z tymi ze schematu elektrycznego z rys. 57). Wiersz 6 i 7 to deklaracja obiektu klasy `Settings` przechowującego ustawienia sterownika oraz jego wypełnienie danymi z pliku JSON odczytanego z pamięci flash. W wierszu 9 dokonuje się ustalenie częstotliwości kanału `PWMChannel1` na 40MHz oraz rozdzielczości 1 bitowej. W linii 12 następuje sprawdzenie czy w opcji konfiguracji sprzętowej definiowanej przez użytkownika wykorzystuje się element wykonawczy podłączony do gniazda SS1 na płytce PCB. Następnie przez komendę z 13. wiersza zlecane jest wytworzenie sygnału prostokątnego w kanale `PWMChannel1` oraz wygaszenie pobudzenia w pozostałych. Współczynnik wypełnienia tego sygnału to $\frac{1}{2^1}=50\%$, gdzie drugi argument funkcji ledcWrite to wykładnik mianownika. W kolejnym wierszu nr 16 następuje 1-sekundowe opóźnienie na właściwe uruchomienie LEDC oraz na osiągnięcie stanu ustalonego w elemencie wykonawczym, a następnie przypisanie do zmiennej vSS1 wyniku średniej z 20tys. pomiarów napięcia podanego na konwerter analogowo portu GPIO 13 (SS1). Ostatecznie instrukcja z wiersza 18. wygasza pobudzenie. Pomiary dokonywane są analogiczny sposób na pozostałych dwóch elementach pomiarowych, o ile są podłączone do czujnika i ustawione w konfiguracji jako dostępne.

Aby nie utracić po wybudzeniu informacji o stanie czujnika (przechowywanych w ulotnej pamięci operacyjnej) zdefiniowane zmienne pomocnicze zostały opatrzone modyfikatorem `RTC_DATA_ATTR`. W taki szczególny sposób postąpiono ze zmienną zawierającą informację o czasie w którym trzeba wysłać kolejny raport pomiarowy do sterownika. Deklaracja:

```
RTC_DATA_ATTR unsigned long NEXT_REPORT_TIME;
```

Analogicznie zdeklarowano zmienne przechowujące informacje o wysłaniu alertu dotyczącego przekroczenia zdefiniowanych dopuszczalnych progów temperatury pracy oraz wilgotności gleby: `soil_hum_alert_sent`, `soil_temp_alert_sent`, `air_temp_alert_sent` oraz

czasu w którym wystąpiło zdarzenie alarmowe `boot_count_alert_start` (według projektu z rys. 42 dotyczącego organizacja struktur danych w czujniku wilgotności).

5.2.2 Implementacja aplikacji mobilnej do zarządzania sterownikiem

Ze względu na kilkanaście wymagań funkcjonalnych ze specyfikacji wymagań dotyczącej aplikacji (D.2) oraz 21-dniowe opóźnienia w harmonogramie realizacji projektu (1/3 przewidzianego czasu) podjęto niełatwą decyzję o realizacji aplikacji głównie w oparciu o komponent WebView[121]. Jest to aplikacja systemowa, wstępnie instalowana na każdym urządzeniu Android, służąca do wyświetlania treści internetowych. Doprecyzowując, oznacza to, że ustawienia użytkownika oraz obsługa wyjątków została oprogramowana natywnie w Java, natomiast integracja z API sterownika i realizacja większości wymagań funkcjonalnych w JavaScript w obrębie dedykowanej mobilnej strony www (znaczniki HTML oraz kaskadowe arkusze stylów CSS). Praktyka taka jest oficjalnie dopuszczalna przez Centrum Zasad dla Deweloperów Android[122], pod warunkiem, że aplikacja prowadzi do strony administrowanej przez wytwórcę. Jeśli chodzi zaś o platformę Apple to wymagana jest dodatkowo funkcjonalność poza tą dostarczaną przez stronę www[123]. Na podstawie powyższego, podjęta decyzja nie będzie negatywnie wpływała na ewentualną publikację w sklepie z aplikacjami Android jak również, w przypadku rozwoju aplikacji na urządzenia Iphone, w sklepie AppStore.

Aplikacja otrzymała nazwę Opti-serv,a pełna nazwa pakietu w której ją zbudowano to: `com.danilewicz.optiservapp`. Za minimalną wersję funkcjonalności ustawiono API Android 21, spełniając wymaganie D.1, o konieczności działania na urządzeniach mobilnych z wersją Android 5 lub wyższą[124]. Jedyną wymaganą zgodą użytkownika do prawidłowego działania aplikacji jest pozwolenie na dostęp do internetu (*android.permission.INTERNET*). Aplikacja zawiera dwie aktywności (*activity*): główną o nazwie `WebActivity` (związaną z wyświetlaniem strony internetowej sterownika, obsługą żądań i wyjątków), drugą: `Garden` dotyczącą danych dostępowych do sterownika, wspieraną przez pomocniczą klasę do inicjalizacji oraz przeprowadzania operacji odczytu i zapisu w bazie danych.

Rozpoczynając od drugiej z wymienionych aktywności, opiera się ona o natywnie dostępną bazę danych SQLite, w której mogą zostać zapisane: nazwa ogrodu, adres IP oraz hasło do sterownika. Zastosowane podejście pozwoli na łatwą rozbudowę aplikacji, w przypadku gdy zajdzie potrzeba, aby użytkownik mógł wybierać sterownik nawadniania z listy i z nim się łączyć. Podczas uruchomienia, aplikacja odczytuje oraz pobiera i próbuje wyświetlić stronę zarządzania ze sterownika (adres IP) jednocześnie podejmując próbę uwierzytelnienia (hasło). W przypadku braku danych wyświetlany jest formularz pozwalający je wprowadzić. W przypadku przekroczenia czasu odpowiedzi wyświetlany jest komunikat sugerujący zmianę ustawień sieci lub danych dostępowych do sterownika (rys. 97).

Natomiast w aktywności `WebActivity` deklarowana jest prywatna klasa rozszerzająca klasę WebViewClient. Realizowana jest przez to własna implementację metod:

a) `onReceivedHttpAuthRequest` – obsługa żądania autoryzacji, którą w przypadku sterownika jest HTTP Basic Authentication[125],

b) `onReceivedHttpError` – obsługa błędów na podstawie kodów odpowiedzi HTTP,

c) `onReceivedError` – obsługa błędów sieci,

d) `onPageFinished` – realizacja zadań po pomyślnym wczytaniu strony (m. in. przekazanie do aplikacji na stronie konfiguracyjnej sterownika przypisanej przez użytkownika nazwy ogrodu). Poza tym stworzona została publiczna klasa `WebAppInterface`, której metody zostają udostępnione dla aplikacji JavaScript z przeglądarki przez zastosowanie adnotacji *`@JavascriptInterface`* przed ich definicją. Należą do nich natywne dla Android alerty i dialogi z opcjami wybory, które inicjuje aplikacja JavaScript uruchamiana w WebView. Przykład został przedstawiony na rys. 93.

Źródłem dokumentu HTML, wraz z kodem JavaScript oraz stylizującym go arkuszem CSS, mogą być wewnętrzne zasoby aplikacji Android lub bezpośrednio sterownik nawadniania. Pierwsza z opcji jest znacznie prostsza do wdrożenia i polega na załadowaniu do instancji WebView pliku wprost z zasobów np. `file:///android_asset/index.html`. Druga zaś wymaga wysłania zapytania HTTP metodą GET do sterownika i wyświetlenia w WebView odpowiedzi. Opcja ta z perspektywy serwera, w którego roli występuje sterownik, wymaga oszczędnego gospodarowania rozmiarem plików, ze względu na bardziej ograniczone zasoby pamięci flash, w porównaniu z pamięcią urządzenia mobilnego. Daje to jednak, niejako pozytywny skutek uboczny, w postaci możliwości konfiguracji sterownika nawadniania za pomocą przeglądarki internetowej i jego obsługę po uwierzytelnieniu. Z racji powyższego wybrano drugą z przedstawionych opcji. W celu ograniczenia wielkości przechowywanych plików w ESP32 za format graficzny ikon przyjęto uniwersalną dwuwymiarową grafikę wektorową SVG[126]. Tymczasem samą aplikację zaimplementowano w czystym JavaScript bez bibliotek wspomagających typu jQuery[127] (zajmującej nawet w wersji skompresowanej około 90KB).

Natomiast w celu poniesienia wrażeń użytkownika aplikacja działa na jednokrotnie wczytanym szkielecie strony zdefiniowanej w pliku *index.html*, z arkuszami stylów *style.css* oraz plikami z kodem źródłowym JavaScript. Następnie komunikacja ze sterownikiem następuje przez obiekt `XMLHttpRequest`, umożliwiający dokonywanie wywołań HTTP w tle. Wszystkie zapytania realizowane są przy użyciu metody GET, z wyjątkiem dotyczących zmian konfiguracji obsłużonych metodą HTTP POST. Każda taka operacja sieciowa jest sygnalizowana animowaną grafiką (*loader)*, imitującym krople spadającej na powierzchnię wody, zwiększając komfort obsługi oraz informując użytkownika trwającej w tle obsłudze żądań. W komponencie `WebView` nie wyłączono pamięci podręcznej *cache,* dlatego też kolejne uruchomienia aplikacji Android nie powodują ponownego pobierania zasobów ze sterownika. Czyszczenie tej pamięci zostało wymuszone jedynie po zmianie ustawień i będzie zachodziło po wyborze nowych

ustawień (w przyszłości po zmianie ogrodu na inny, przewidując sytuację w której obsługiwane są sterowniki z różnymi wersjami oprogramowania układowego).

Większość węzłów końcowych (ang. *endpoints*) API sterownika w wyniku uwierzytelnionego zapytania zwraca odpowiedź JSON, która po deserializacji, jest wyświetlana w aplikacji mobilnej jak np. *stan stref* (rys. 92), *logi zdarzeń* (rys. 95) lub *prognoza pogody* (rys. 98). Nieco rozszerzony scenariusz zachodzi w przypadku wybrania opcji zmiany ustawień sterownika lub czujników wilgotności (dodawanie lub edycja istniejących). Na takie żądanie sterownik odpowiada zwracając plik konfiguracyjny JSON, który jest w aplikacji transformowany do formularza (rys. 94), z odpowiednimi nazwami pól oraz wyjaśnieniem ich znaczenia, dla użytkownika dostępnym po wciśnięciu przycisku informacyjnego (przykład na rys. 93). Zakresy liczb dla pól numerycznych oraz długości tekstu i dopuszczalne znaki dla pól tekstowych zostały dostosowane do wielkości i rodzaju zmiennych jakie zostały im przypisane w sterowniku. Działanie to zapewnia to walidację danych po stronie klienta. Po zatwierdzaniu zmian, dane formularza podlegają serializacji i są odsyłane do sterownika nawadniania w formacie JSON, celem zapisu do jego pamięci nieulotnej.

Wprowadzenie danych dostępowych do sterownika dokonuje się za pomocą formularza (rys. 96) dostępnego z menu pod pozycją ‘Wybór ogrodu’ lub podczas niepomyślnych prób połączenia za pomocą opcji ‘Dane sterownika’ (rys. 97).

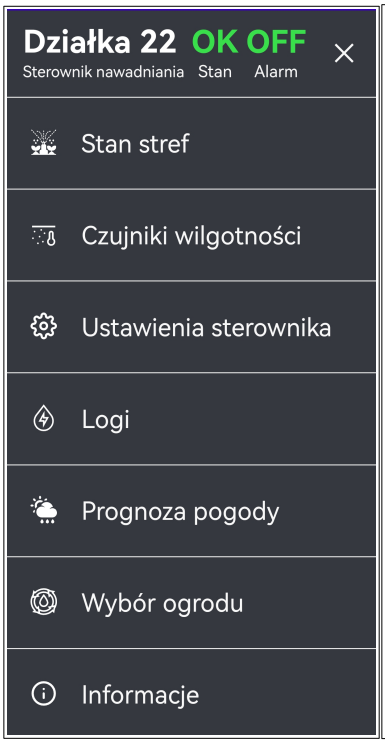
Ostatecznie pliki aplikacji w sterowniku zajmują 180.4KB, z czego 52,4% to wspomniana animacja *loadera* w formacie gif, 17,5% to grafiki formatu .svg, a pozostałe miejsce zajmują pliki js, html i css. Natomiast plik pakietu *apk* na Android na w przybliżeniu zajmuje 6,3MB. Na rysunkach od 91 do 99 przedstawiono zrzuty ekranu z aplikacji działającej na telefonie z systemem Android 11: menu, stan stref, przykład pomocy kontekstowej, ustawienia sterownika, logi zdarzeń, zmiana parametrów dostępowych, przykład obsługi błędu, prognozę pogody oraz informacje.

5.3 Przeprowadzenie testów sprzętu, oprogramowania oraz realizacja scenariuszy testowych

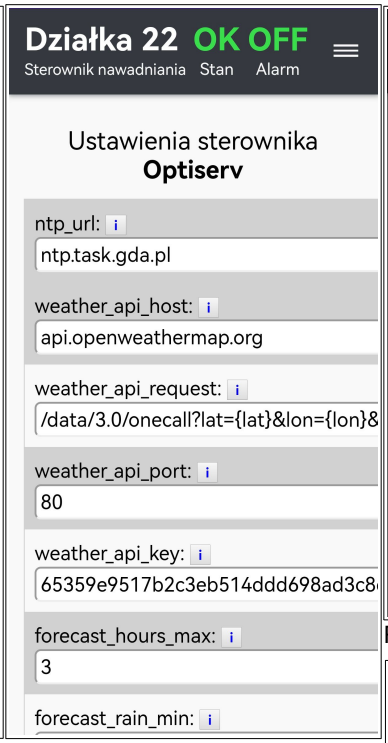
Po fazach projektowana czujnika wilgotności i sterownika, stworzone zostały pierwsze plany dotyczące poziomów testów zarówno funkcjonalnych jak i niefunkcjonalnych w stosunku do produktów w poszczególnych fazach powstawania. Uwidoczniono je na rysunku nr 100.

Podczas wstępnego etapu realizacji projektu, gdy obwody elektryczne budowane były na płytkach stykowych, ostatecznie zweryfikowano kompatybilność i działanie modułów elektronicznych (tabela 20 i 21) z platformą ESP32, opierając się o obsługujące je biblioteki programistyczne. Przejście do kolejnej fazy projektu nastąpiło dopiero po ich pomyślnej weryfikacji i osiągnięciu stabilnego działania.

W celu jak najwcześniejszego wyeliminowania usterek, a także ze względu czytelny podział kodu ze względu na funkcjonalność (tabela 36), pierwsze testy wykonywane były już w trakcie implementacji. Między innymi, testami jednostkowymi objęto metody klasy `WeatherForecast` dotyczące aktualizacji pogody oraz obliczania ewapotranspiracji

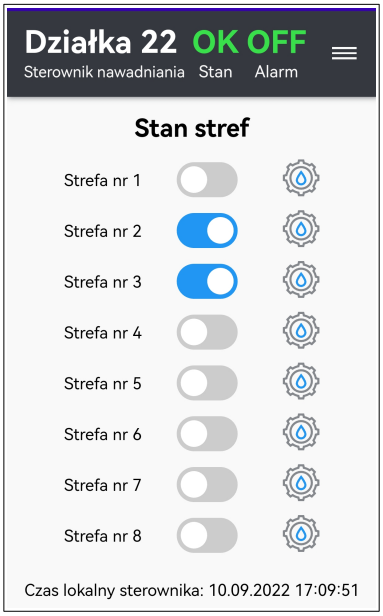


Rysunek 91: Menu

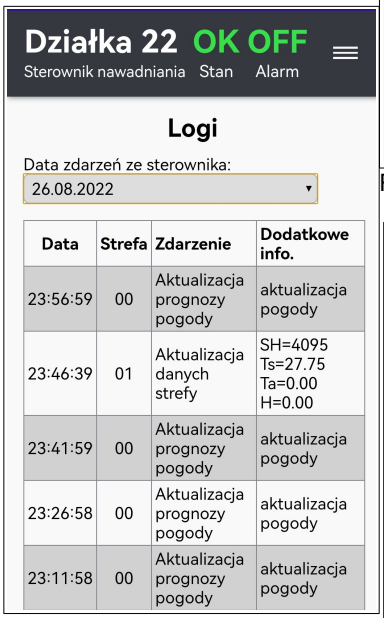


Rysunek 94:Ustawienia sterownika

Rysunek 97: Obsługa wyjątków

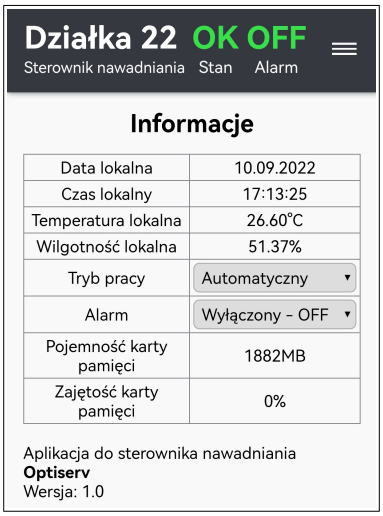


Rysunek 92: Stan stref

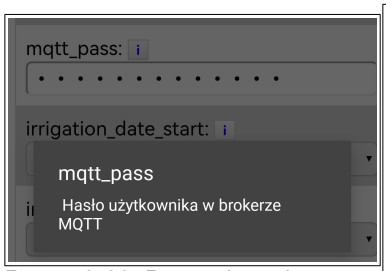


Rysunek 95: Logi zdarzeń

Rysunek 98: Prognoza pogody



Rysunek 99: Informacje



Rysunek 93: Pomoc kontekstowa

Rysunek 96: Parametry dostępowe

Źródłem ikon wykorzystanych w aplikacji jest strona https://freeicons.io/

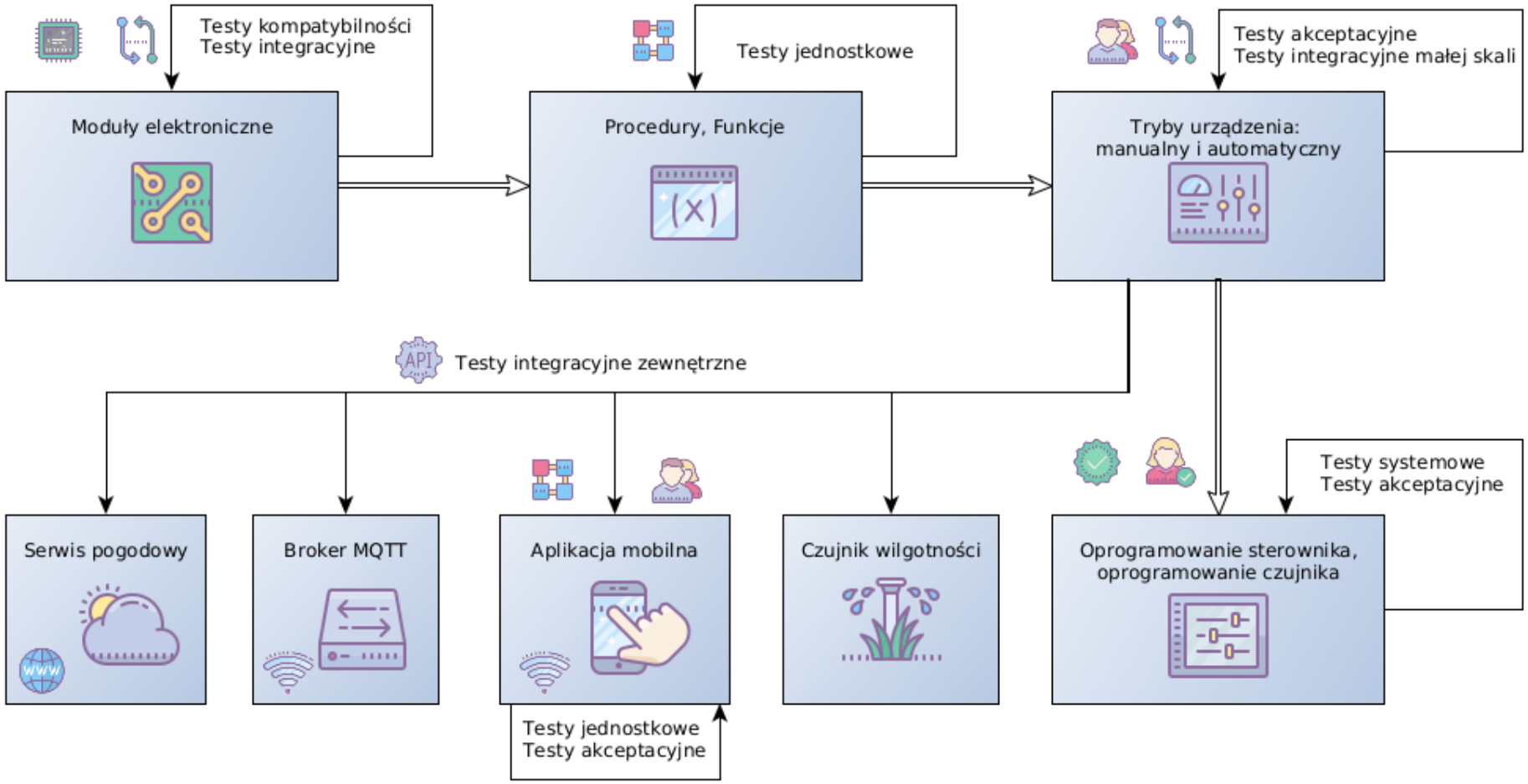
czy szereg metod pomocniczych zdefiniowanych w pliku `helpers.cpp`. Służyły one do formatowania daty i czasu, obliczania statystyk z przekazywanej tablicy wartości lub walidacji wartości wielkości fizycznych. Testy zostały zaimplementowane przy użyciu frameworka Unity[128], zintegrowanego z wykorzystywanym środowiskiem deweloperskim PlatformIO, pozwalającym na ich uruchamianie na docelowym urządzeniu wbudowanym z rodziny ESP32 z ograniczonymi zasobami.

Natomiast do przetestowania węzłów końcowych w sterowniku dostępnych za pomocą protokołu HTTP wykorzystano oprogramowanie Postman[129]. Wstępnie obejmowało to obsługę odpowiedzi na żądania, następnie weryfikacją działania autoryzacji *Basic Auth*, aż po proste sekwencje wywołań zapisane w postaci skryptów. Dzięki temu zoptymalizowano pamięciowo implementację węzłów dotyczących zapisu konfiguracji (`/save`) oraz pobierania zdarzeń z określonego dnia (`/get-logs`). Testy pomogły również w wykryciu błędów w postaci użycia niewłaściwego typu zmiennej, co powodowało utratę informacji. Następnie udostępniany przez sterownik interfejs programistyczny aplikacji został zaimplementowany w aplikacji mobilnej, na której przeprowadzono testy akceptacyjne dotyczące zdalnych funkcjonalności.

Już w fazie implementacji powstały scenariusze testowe do testów akceptacyjnych, uwzględniające wszystkie zdefiniowane o przypadki użycia. Zostały one zrealizowane zgodnie z opisem przedstawionym w podrozdziale 3.7 oraz utrwalone w postaci materiałów wideo, co stanowi zawartość załącznika G do niniejszej pracy. W trakcie ich przeprowadzania następowała weryfikacja z wyspecyfikowanymi wymaganiami. Niezgodności były odnotowywane w arkuszu kalkulacyjnym, naprawiane i po pomyślnym przebiegu kolejnych testów oznaczane jako wykonane. Szczególną uwagę poświęcono na weryfikację działania implementacji algorytmu OptiServ. Analiza wyników Scenariusza S4 pozwala stwierdzić, że sterownik zgodnie z oczekiwaniami czasowymi dokonywał włączania nawadniania w strefach, reagował na zmiany w środowisku skracają czas nawadniania, a także zawieszał nawadnianie w przypadku przekroczenia zakresu dopuszczalnych temperatur.

W tym miejscu warto wspomnieć, że na badania elementu wykonawczego i analizie wyników czterech eksperymentów (rozdział 4) poświęcono znaczne zasoby czasu. Ostatecznie sprawdzony w działaniu element pomiarowy, którego określono charakterystyki pomiarowe został włączony do czujnika wilgotności i pomyślnie przeszedł testy systemowe.

Podsumowując, czas poświęcony na przeprowadzenie szerokiego zakresu testów i poprawki związane z niejednokrotnie niepomyślnymi wynikami zajął zdecydowanie więcej czasu niż sama implementacja stworzonych wcześniej modeli i algorytmu. Czas przeznaczony na analizę kodu źródłowego zaowocował jego wyraźnemu ulepszeniu, co zapobiegnie ewentualnym przyszłym problemom. Pomijając niezaprzeczany cel testowania jakim było znajdowanie usterek, przeprowadzony proces spowodował nabranie zaufania do wytworzonego oprogramowania i urządzeń pracujących pod jego kontrolą.



Rysunek 100: Poziomy przeprowadzonych testów z uwzględnieniem usług i modułów zewnętrznych

Źródło: opracowanie własne

Grafika: https://icons8.com

ZAKOŃCZENIE

Droga do postawionego celu głównego, została dobrze zaplanowana oraz określona w czasie i zakresie za pomocą harmonogramu projektu. Analiza dostępnych na rynku sterowników nawadniania i ich funkcjonalności, doprowadziła do stworzenie specyfikacji wymagań wpisującej się w panujące trendy. Jednakże w trakcie prac projektowych dotyczących struktur i organizacji pamięci w sterowniku, jedna myśl trendy te przełamała. Dlaczego nie wykorzystać danych z czujnika strefowego do uruchamiania nawadniania? To doprowadziło do pogłębienia przeglądu literatury dotyczącej potrzeb wodnych roślin i określenia granic czasowych irygacji od wschodu do zachodu Słońca i wykorzystania zegara, w jakim czas postrzega flora, a ostatecznie do powstania algorytmu Optiserv. Połączenie potencjału bezprzewodowego czujnika wilgotności, jaki stanowią zbierane przez niego dane środowiskowe z prognozami meteorologicznymi pobieranymi z internetu, pozwoliło sterownikowi na zmniejszenie zużycia wody potrzebnej do nawadniania. Mechanizmy bezpieczeństwa zarówno w samym algorytmie (graniczny okres podlewania) jaki i na płaszczyźnie fizycznej (czujnik przepływu cieczy, czujnik ruchu), niezwłoczne powiadamianie o zdarzeniach niepożądanych (komunikaty MQTT) oraz szyfrowana komunikacja między elementami systemu dają solidne podstawy do zaufania.

Warte odnotowania jest, że osiągnięto **pierwszy cel pośredni**, poprzez wyposażenie bezprzewodowych czujników wilgotności gleby w autorski i poddany badaniom układ pomiarowy. Podobnie, poprzez implementację dedykowanej aplikacji mobilnej **drugi cel pośredni** pracy został skutecznie zrealizowany. Ostatecznie, osiągnięto **cel główny pracy,** gdyż stworzono wielostrefowy sterownik do systemu nawadniania ogrodu, opierając jego działanie na algorytmie Optiserv, uwzględniając dane z bezprzewodowych czujników temperatury i wilgotności oraz prognoz pogody.

Podczas projektowania kierowano się specyfikacją wymagań gwarantującą osiągnięcie celów pracy, jednakże wzięto też pod uwagę dalszy rozwój systemu. W samym sterowniku przygotowano port do łatwego rozszerzenia ilości stref do 16, uwzględniając tą sytuację w oprogramowaniu. Poza tym do wykorzystania się dwa fizyczne wyprowadzenia linii układu głównego sterownika wraz zasilanie 3.3V i 5V np. w celu podłączenia przewodowych czujników. Tym samym zostawiono zapas do zasilenia innych urządzeń peryferyjnych przez dobór odpowiednich parametrów np. w stabilizatorach napięcia. Wykorzystano 32% pojemności pamięci flash, co umożliwia w przyszłości implementację aktualizacji oprogramowania układowego np. przez aplikację mobilną korzystając z mechanizmu Over-the-Air. Dalszym pożądanym kierunkiem rozwoju jest wzbogacenie systemu o inne rodzaje czujników środowiskowych, jak na przykład wersja doniczkowa czujnika wilgotności gleby (zaprojektowana, jednak nie omówiona w tej pracy, rys. 54), czy czujnik czystości powietrza. Możliwa jest też stworzenie integracji z systemem domu inteligentnego opartego na już zaimplementowanym protokole MQTT. W czujniku wilgotności analogicznie dostępne jest jedno wolne wyprowadzenie.

Zakres materiału obejmujący wiedzę z gleboznawstwa i botaniki odkrył i poszerzył wiedzę autora pracy z tych dziedzin. Projekt i samodzielnie zlutowanie układów elektronicznych stanowiło realizację jego zainteresowań. Natomiast projekt oprogramowania i jego implementacja w układach ESP32 w sterowniku i czujnikach, stworzenie aplikacji na urządzenia mobilne z systemem Android oraz szeregu pomocniczych w projekcie i badaniach skryptów, stało się kolejnym krokiem praktyki inżynierskiej. Nieocenione i niezbędne wiedza i umiejętności z podjętego kierunku studiów ‘systemy i sieci komputerowe’ zostały wykorzystane podczas tworzenia bezpiecznej komunikacji bezprzewodowej. Podejście zastosowane do opisu, modelowania i symulacji algorytmu Optiserv oraz badań elementu wykonawczego czujnika stanowią zaś wyraz naukowych aspiracji.

Szczęście, o którym mowa w przysłowiu ze wstępu, według autora pracy nie wynika z samego założenia i urządzenia ogrodu, a bardziej z jego utrzymania i pielęgnacji. Stworzony wielostrefowy sterownik nawadniania zdecydowanie ułatwią utrzymanie ogrodu w dobrej kondycji, dając równolegle więcej czasu jego właścicielowi, choćby na odpoczynek i radość z przebywania w nim.

WYKAZ RYSUNKÓW

WYKAZ TABEL

SPIS OPROGRAMOWANIA WYKORZYSTANEGO W PROJEKCIE I NINIEJSZEJ PRACY

LP.	Nazwa	Wersja	Rodzaj	Licencja
1	Linux Mint	20.2	System operacyjny	GPL
2	LibreOffice Writer	6.4.7.2	Edytor tekstu	MPLv2.0
3	LibreOffice Calc	6.4.7.2	Arkusz kalkulacyjny	MPLv2.0
4	yEd Graph Edytor	3.21.1	Edytor graficzny diagramów	Freeware
5	PlatformIO	Core: 6.1.4 Home: 3.4.3	Zintegrowane środowisko deweloperskie	Apache 2.0
6	FreeCAD	0.20	Modelowanie CAD 3D	LGPLv2+
7	EasyEDA	6.3.22	Projektowanie płytek PCB	Freeware
8	Android Studio	2021.2.1	Zintegrowane środowisko deweloperskie	Freeware
9	Visual Studio Code	1.63.2	Edytor kodu źródłowego	MIT License
10	Project Libre	1.9.3	Zarządzanie projektami	Common Public Attrib. License 1.0
11	GIMP	2.10.18	Edytor obrazów	GPL v3.0+

SPIS BIBLIOTEK PROGRAMISTYCZNYCH UŻYTYCH W PROJEKCIE

Lp.	Nazwa blibioteki	Opis	Repozytorium biblioteki	Licencja	Zastosowanie
1	Arduino-LoRa	Protokół LoRa	https://github.com/sandeepmistry/arduino-LoRa	MIT License	St, Cz
2	ArduinoJson	Przetwarzanie JSON	https://github.com/bblanchon/ArduinoJson	MIT License	St, Cz
3	mbedtls	Biblioteka kryptograficzna	https://github.com/Mbed-TLS/mbedtls	Apache-2.0 license	St, Cz
4	Adafruit BME280 Library	Czujnik 3w1 temp, wilg, ciśnienie	https://github.com/adafruit/Adafruit_BME280_Library	BSD license	Cz
5	Adafruit_AHTX0	Czujnik 3w1 temp, wilg, ciśnienie	https://github.com/adafruit/Adafruit_AHTX0	BSD License	St
6	MCP23017_RT	Expander wyjść	https://github.com/RobTillaart/MCP23017_RT	MIT License	St
7	ConfigManager	Menedżer konfiguracji	https://github.com/nrwiersma/ConfigManager	MIT License	St
8	Rtc	Zegar RTC	https://github.com/Makuna/Rtc	LGPL-3.0 license	St
9	RotaryEncoder	Pokrętło z przyciskiem	https://github.com/mathertel/RotaryEncoder	BSD License	St
10	ESPAsyncWebServer	Asynchroniczny serwer www	https://github.com/me-no-dev/ESPAsyncWebServer	-	St
11*	Adafruit_Si5351	Generator sygnału	https://github.com/adafruit/Adafruit_Si5351_Library	BSD License	Cz
12	Time	Przetwarzanie czasu	https://github.com/PaulStoffregen/Time	-	St
13	NTPClient	Klient NTP	https://github.com/arduino-libraries/NTPClient	-	St
14	Arduino-MQTT	Protokół MQTT	https://github.com/256dpi/arduino-mqtt	MIT License	St
15	SD_MMC	Karta SD	https://github.com/espressif/arduino-esp32/tree/master/libraries/SD_MMC	LGPL v2.1	Ex
16	ArduinoTemperature Control Library	Czujnik temperatury	https://github.com/milesburton/Arduino-Temperature-Control-Library	LGPL v2.1	Cz, Ex
17	LiquidCrystal_I2C	Ekran LCD 20x4	https://registry.platformio.org/libraries/marcoschwartz/LiquidCrystal_I2C	MIT License	St, Ex
18	HX711_ADC	Wzmacniacz do belki tensometrycznej	https://github.com/olkal/HX711_ADC	MIT License	Ex
19	Ace Button	Obsługa przycisków mechanicznych	https://github.com/bxparks/AceButton	MIT License	St, Cz

*wykorzystano w wersjach rozwojowych, ale nie w wersji finalnej

Legenda:

St - sterownik,

Cz - czujnik,

Ex - układ do przeprowadzenia eksperymentu

ZAŁĄCZNIKI:

A. Instrukcja obsługi,

B. Wskazówki dla projektanta i dalszego rozwoju systemu,

C. Opis aplikacji,

D. Diagram Gantta,

E. Wykres sieciowy metody ścieżki krytycznej,

F. Podsumowanie symulacji algorytmu OptiServ,

G. Materiały wideo z realizacji scenariuszy testowych,

H. Pliki z wynikami i projektowe.

[1] Berkowska E., Gwiazdowicz M.: *Deficyt wody w Polsce*, Wydawnictwo Sejmowe dla Biura Analiz Sejmowych, Warszawa 2020
[2] Państwowe Gospodarstwo Wodne Wody Polskie, https://stopsuszy.pl/, (dostęp 16.10.2021 r.)
[3] Kaczmarczyk S., Nowak L, *Nawadnianie roślin, PWRiL*, Warszawa 2005, s.217-220
[4] *Oficjalna strona internetowa NASA, The Solar System and Beyond is Awash in Water, https://www.nasa.gov/jpl/the-solar-system-and-beyond-is-awash-in-water* (dostęp 18.10.2021r.)
[5] HB-System, *Automatyczny system nawadniania*, https://www.nawodnienia.eu/files/1676974311/file/poradnik_hbsystem__automatyczny_system_nawadnia nia__doborprojektmontaz.pdf, s.3-4,25 (dostęp 18.10.2021r.)
[6] Stępniewski M., Pompy, Wydawnictwa Naukowo-Techniczne, Warszawa 1985r., s.25-31
[7] *Irrigation Scheduling with Tensiometers,* Ministry of Agriculture, British Columbia, Order No. 577.100-2, https://www2.gov.bc.ca/assets/gov/farming-natural-resources-and-industry/agriculture-and-seafood/agricultural-land-and-environment/water/irrigation/577100-2_irrigation_scheduling_with_tensiometers.pdf (dostęp 23.10.2021r.)
[8] Muñoz-Carpena R., Dukes M., *Automatic Irrigation Based on Soil Moisture for Vegetable Crops*, AE354, Department of Agricultural and Biological Engineering (dostęp 23.10.2021r.), s.1-3
[9] Strona internetowa producenta sterownika do nawadniania Rachio, https://rachio.com
[10] Strona internetowa producenta sterownika do nawadniania Rainbird, https://www.rainbird.com
[11] Strona internetowa producenta sterownika do nawadniania Hunter, https://www.hunterindustries.com
[12] Strona internetowa producenta sterownika do nawadniania Rainbird, https://www.orbitonline.com
[13] Strona zakładu produkcyjnego EOTECH, http://eotech.pl/oznaczenie-budowa-i-dzialanie-elektrozaworu--poradnik, (dostęp 18.10.2021r.)
[14] Bolkowski S., *Elektrotechnika*, Wydawnictwa Szkolne i Pedagogiczne Spółka Akcyjna, Warszawa 2005, s.170-171
[15] Strona internetowa producenta elektrozaworów: https://rainpolska.pl
[16] Strona internetowa producenta elektrozaworów: https://www.rainbird.com
[17] Strona internetowa producenta elektrozaworów: https://bermad.com.cn
[18] Strona internetowa producenta elektrozaworów: http://www.irritrol.it
[19] Strona internetowa producenta elektrozaworów: https://hunterpolska.pl
[20] Strona internetowa producenta elektrozaworów: https://www.toro.com
[21] Strona internetowa producenta elektrozaworów: https://www.waterirrigation.co.uk
[22] Karta katalogowa bezprzewodowego czujnika deszczu Rain Click Sensor, producent: Hunter https://hunterpolska.pl/pdf/wrc_instrukcja_2575732786.pdf
[23] Karta katalogowa bezprzewodowego czujnika deszczu i mrozu WR2, producent: Raind Bird, https://www.rainbird.com.pl/products/bezprzewodowy-czujnik-deszczumrozu-z-serii-wr2-i-wr2-48
[24] Longstroth M., U*sing sprinklers to protect plants from spring freezes*, Michigan State University Extension, https://www.canr.msu.edu/news/using_sprinklers_to_protect_plants_from_spring_freezes, (dostęp 24.10.2021r.)
[25] Łabędzki L., *Wielkość i zmienność ewapotranspiracji wskaźnikowej według Penmana-Monteitha w okresie wegetacyjnym w latach 1970–2004 w wybranych rejonach Polski*, Instytut Technologiczno-Przyrodniczy, Falenty, 2012, s.159-162
[26] Campbell G., Campbell C., Cobos. D., *The researcher's complete guide to water potential*, https://www.metergroup.com/environment/articles/the-researchers-complete-guide-to-water-potential (dostęp 22.10.2021r.)
[27] Jamróz P., Socha K., *Modułowy system pomiarowy do monitoringu wilgotności spągu w kopalni soli,* Prace Instytutu Mechaniki Górotworu PAN Tom 22, nr 1-4, Kraków 2020, s.45-53
[28] Sample D., Owen J., Fields J. Barlow S., *Understanding Soil Moisture Sensors: A Fact Sheet for Irrigation Professionals in Virginia*, Virginia Cooperative Extension (VCE), https://vtechworks.lib.vt.edu/bitstream/handle/10919/75547/BSE-198.pdf (dostęp 22.10.2021r.), s.2-6

[29] Campbell G., Campbell C., Cobos. D., *TDR, FDR, capacitance, resistance methods – compared*, https://www.metergroup.com/en/meter-environment/measurement-insights/tdr-fdr-capacitance-compared (dostęp 22.10.2021r.)
[30] P. Raja & A. Barron, Physical Methods in Chemistry and Nano *Science,* Rice University 2022r., s.202, Figure 2.9.1, https://batch.libretexts.org/print/Finished/chem-55670/Full.pdf (dostęp 31.08.2022r.)
[31] Grześ A., *Wykres gantta a metoda ścieżki krytycznej (CPM)*, Optimum. Studia Ekonomiczne, nr 4(70) 2014r.
[32] Makuchowski M., Wykład pt. *Metoda CPM/PERT*, Wydział Informatyki i Telekomunikacji, http://mariusz.makuchowski.staff.iiar.pwr.wroc.pl/download/courses/komputerowe.wspomaganie.zarzadzania/wyk.slajdy/cpmpert.pdf , (dostęp 15.02.2022r.)
[33] Wróbel M., *Wykład z Zarządzania Projektem Informatycznym pt. „Szacowanie i harmonogramowanie”*, Politechnika Gdańska, WETI, 2021r.
[35] Oficjalna strona internetowa usługi Dropbox, https://experience.dropbox.com/pl-pl/resources/critical-path (dostęp 07.02.2022r.)
[36] Sprawdzenia na stronach internetowych dostawców komponentów: www.tme.eu; www.mouser.pl; www.maritex.com.pl; pl.rs-online.com, (dostęp 01.10.2021r.)
[37] Wyszukiwarka produktów firmy Espressif, https://products.espressif.com, (dostęp 07.02.2022r.)
[38] Karta katalogowa modułu ESP32WROOM32E, producent: Espressif, https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf, str. 2-4,15 ,26
[39] Dokumentacja interfejsu programistycznego API modułu ESP32,https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/memory-types.html, (dostęp 07.02.2022r.)
[40] Vedat O., *Developing IoT Projects with ESP32*, Packt Publishing, Birmingham, 2021r., s. 30-34
[41] Dokumentacja interfejsu programistycznego API modułu ESP32,https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/efuse.html (dostęp 07.02.2022r.)
[42] Strona internetowa z dokumentacją ESP32, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/ulp.html (dostęp 06.02.2022r.)
[43] Forum dotyczące elektroniki cyfrowej, https://www.mischianti.org/wp-content/uploads/2021/02/ESP32-wroom-32-pinout-mischianti.jpg (dostęp 08.02.2022r.)
[44] Strona internetowa sklepu dla elektroników Botland, https://botland.com.pl/ (dostęp 06.02.2022r.)
[45] Kolban N., *Kolban's Book on the ESP32*, 2017r., https://www.robolinkmarket.com/Data/EditorFiles/datasheet/kolban-ESP32.pdf, s.54,331,338
[46] Strona internetowa interpretera JavaScript dla mikrokontrolerów Espruino, https://www.espruino.com (dostęp 07.02.2022r.)
[47] Strona internetowa interpretera JavaScript dla mikrokontrolerów Duktape, https://duktape.org (dostęp 07.02.2022r.)
[48] Strona internetowa Wydziału Matematyki i Informatyki Uniwersytetu Oksfordzkiego, http://math.oxford.emory.edu/site/cs170/interpreterVsCompiler/ (dostęp 06.02.2022r.)
[49] Podręcznik specyfikacji technicznej ESP32*, ESP32 Technical Reference Manual*, Version 4.6, Espressif Systems, Shanghai, 2021r., s.278-280,378-380
[50] Marwedels P., *Embedded System Designs*, wyd.4, Springer International Publishing, Cham, 2018r. s.187-188
[51] Tomsik M., Lis S., Korenko M., *Modulacja szerokości impulsu PWM w sterowaniu automatycznym*, Przegląd Elektrotechniczny, R. 93 NR 12/2017
[52] Dokumentacja interfejsu programistycznego API modułu ESP32, *LED Control (LEDC)* https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/ledc.html, (dostęp 06.02.2022r.)
[52] Tulson R., Wilmshurst T., Fast and Effective Embedded Systems Design - Applying the ARM mbed, wyd.2, Elsevier, Cambridge 2017r., s.135-163

[53] NXP Semiconductors, *I2C-bus specification and user manual*, Rev. 7.0, 2021r., https://www.nxp.com/docs/en/user-guide/UM10204.pdf, s.3-14 (dostęp 06.02.2022r.)
[54] Strona internetowa poświęcona magistrali I2C, Telos Systementwicklung GmbH, https://www.i2c-bus.org/i2c-primer/how-i2c-hardware-works (dostęp 16.02.2022r.)
[55] *ESP-IDF Programming Guide*, Espressif Systems, Shanghai, 2022r., s.703-708
[56] Strona internetowa edytora kodu Visual Studio Code, https://code.visualstudio.com (dostęp 29.11.2021r.)
[57] Strona internetowa i dokumentacja PlatformIO, https://docs.platformio.org (dostęp 29.11.2021r.)
[58] Strona internetowa sklepu z rozszerzeniami dla Visual Studio Code, https://marketplace.visualstudio.com (dostęp 29.11.2021r.)
[59] Oficjalna strona z narzędziami i bibliotekami kryptograficznymi dla deweloperów C/C++/C#, https://www.cryptosys.net/manapi/api_aeadalgorithms.html (dostęp 15.02.2022r.)
[60] Oficjalna dokumentacja biblioteki mbed TLS, https://www.trustedfirmware.org/projects/mbed-tls/ (dostęp 28.02.2022r.)
[61] Oficjalna dokumentacja firmy Semtech, https://lora-developers.semtech.com/documentation/tech-papers-and-guides/the-book/packet-size-considerations/ (dostęp 28.02.2022r.)
[62] N.Sornin, A.Yegin, LoRaWAN™ 1.1 Specification, LoRa Alliance™, Beaverton, 2017r.
[63] Dokument techniczny firmy Semtech, *LoRa® and LoRaWAN®:A Technical Overview,* Camarillo,2020r., http://semtech.com/LoRa
[64] Strona internetowa firmy Semtech, https://www.semtech.com, (dostęp 10.02.2022r.)
[65] *Rozporządzenie Ministra Infrastruktury z dnia 19 sierpnia 2011r. zmieniające rozporządzenie w sprawie urządzeń radiowych nadawczych lub nadawczo-odbiorczych, które mogą być używane bez pozwolenia radiowego*, Dziennik Ustaw Nr 188, Poz. 1122, Internetowy System Aktów Prawnych, http://isap.sejm.gov.pl/
[66] *Lora w praktyce(1)*, Elektronika Praktyczna, 10/2017, s.100-104
[67] Chiani M., Elzanaty A.,*On the LoRa Modulation for IoT: Waveform Properties and Spectral Analysis, EEE Internet of Things Journal, 6(5), 8463–8470*
[68] *LoRa™ Modulation Basics*, AN1200.22, Revision 2, Semtech Corporation - Wireless Sensing and Timing Products Division, Camarillo, 2015r., s.1-13
[69] Oficjalna strona internetowa Internet Engineering Task Force (IETF), *Network Time Protocol Version 4: Protocol and Algorithms Specification*, https://datatracker.ietf.org/doc/html/rfc590, (dostęp 12.02.2022r.)
[70] Rybaczyk P., *Expert Network Time Protocol - An Experience in Time with NTP*, Apress, Berkeley, 2006r., s.58-59
[71] Strona internetowa Głównego Urzędu Miar, https://www.gum.gov.pl/pl/uslugi/zegar/524,Zegar.html, (dostęp 12.02.2022r.)
[72] Pulver T., *Hands-On Internet of Things with MQTT*, Packt Publishing, Birmingam, 2019, s.57-59
[73] Oficjalna strona internetowa protokołu MQTT, https://mqtt.org/, (dostęp 13.06.2022r.)
[74] B.Mishra, A.Kertesz, *The Use of MQTT in M2M and IoT Systems: A Survey,* czasopismo IEEE Access, vol. 8, s.201071- 201079
[75] Oficjalna strona internetowa Amazon AWS IOT, https://aws.amazon.com/iot/ (dostęp 13.06.2022r.)
[76] Strona internetowa brokera MQTT HiveMQ, https://www.hivemq.com/solutions/automotive/, (dostęp 13.06.2022r.)
[77] Strona z zasobami grafik, https://www.flaticon.com (dostęp 13.06.2022r.)
[78] Oficjalne repozytorium protokołu MQTT, https://github.com/mqtt/mqtt.org/wiki/Differences-between-3.1.1-and-5.0, (dostęp 13.06.2022r.)
[79] Oficjalne repozytorium protokołu MQTT, https://github.com/topics/mqtt-sn, (dostęp 13.06.2022r.)
[80] Mars T., *JSON at Work, Practical Data Integration for the WEB*, O'Reilly Media, Sebastopol, 2017r. s.3-5
[81] Smith B., *Beginning JSON*, Apress Media, New York, 2015
[82] Oficjalna strona internetowa Internet Engineering Task Force (IETF), *The JavaScript Object Notation (JSON) Data Interchange Format*, https://tools.ietf.org/html/rfc8259, (dostęp 06.02.2022r.)
[83] ECMA Technical Committee 39, *The JSON Data Interchange Syntax, Standatd ECMA-404*, Ecma International 2017

[84] Strona internetowa JSON Editor Online, http://jsoneditoronline.org (dostęp 08.02.2022r.)
[85] Blanchon B., *Mastering ArduinoJson - Efficient JSON serialization for embedded C++*, https://arduinojson.org/book/, (dostęp 06.02.2022r.)
[86] Internetowy serwis i sklep dotyczący nawadniania https://www.nawodnienia.eu/ (dostęp 25.05.2022r.)
[87] Basarat S.,*Beginning Node.js,* Apress Media, Nowy Jork, 2014r., s.1-65
[88] Algorytm Obserwatorium Marynarki Wojennej Stanów Zjednoczonych do obliczania wschodów i zachodów słońca, http://www.edwilliams.org/sunrise_sunset_algorithm.htm (dostęp 31.05.2022r.)
[89] Oficjalna strona internetowa i dokumentacja biblioteki JavaScript ChartJS do tworzenia wykresów, https://www.chartjs.org/ (dostęp 12.05.2022r.)
[90] Wróbel M., Wykład 'Zarządzanie projektem informatycznym', kurs 'Planowanie projektu' str.15 , rok akadem. 2020/2021, Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki
[91] Oficjalna strona internetowa oprogramowania ProjectLibre do zarządzania projektami, https://www.projectlibre.com/ (dostęp 13.04.2022r.)
[92] Poradnik użytkownika oprogramowania ProjectLibre https://project-management.com/projectlibre-tutorial-part-1/ (dostęp 13.04.2022r.)
[93] Strona internetowa producenta układów elektronicznych Heltec, https://heltec.org/project/wifi-lora-32/, (dostęp 20.02.2022r.)
[94] Strona internetowa agencji prasowej dostarczające informacji na temat rynków finansowych https://www.bloomberg.com/graphics/2021-semiconductors-chips-shortage/
[95] Bezpłatna encyklopedia internetowa, https://en.wikipedia.org/wiki/Year_2038_problem (dostęp 1.06.2022r.)
[96] Aplikacja do projektowania obwodów elektronicznych, https://easyeda.com (dostęp 24.11.2021r.)
[97] Strona internetowa firmy zajmującej się zaawansowaną analiza i modelowaniem zakłóceń elektromagnetycznych https://www.emisoftware.com/calculator/coplanar-capacitance/
[98] Oficjalna strona internetowa projektu Arduino, https://www.arduino.cc/en/Tutorial/Foundations/CapacitanceMeter (dostęp 01.06.2022r.) https://www.arduino.cc/reference/en/language/variables/variable-scope-qualifiers/volatile/
[99] Serwis internetowy magazynu Elektronika Praktyczne, https://serwis.avt.pl/files/kurs_c/25_KursAVR_cz04.pdf (dostęp 02.06.2022r.)
[100] Roux J., Escriba Ch. i in., *A new bi-frequency soil smart sensing moisture and salinity for connected sustainable agriculture,* Journal of Sensor Technology, Scientific Research, 2019, 9 (3), s.1-2
[101] Abdelaziz O., Hasnaa I., *Designing Low-Cost Capacitive-Based Soil Moisture Sensor and Smart Monitoring Unit Operated by Solar Cells for Greenhouse Irrigation Management,* Sensors 2021, 21, s.4
[102] Karta katalogowa rodziny układów Si5351, https://cdn-shop.adafruit.com/datasheets/Si5351.pdf
[103] Harris, N., Stonard A., *A Printed Capacitance Sensor for Soil Moisture Measurement*. Proceedings. vol. 2., 2018r., s.4
[104] Zypman F., *Mathematical expression for the capacitance of coplanar strips,* Journal of Electrostatics, vol. 101, 2019r., s.4
[105] Chen A., Darhube A., *Capacitive sensing of droplets for microfluidic devices based on thermocapillary actuation, Lab Chip vol. 4, 2004r., s.473*
[106] Campbell G., Campbell C., Cobos. D., *How install soil moisture sensors faster, better, and higher accuracy*, https://www.metergroup.com/en/meter-environment/expertise-library/how-install-soil-moisture-sensors-faster-better-and-higher, (dostęp 05.06.2022r.)
[107] Karta katalogowa oscylatora kwarcowego, https://datasheet.lcsc.com/lcsc/1912111437_HELE-Harmony-Elec-SSI030000I3CHE-T_C254241.pdf, (dostęp 01.06.2022r.)
[108] Karta katalogowa przewodu UTP kat. 5e produkowanych przez Bitner: https://bitner.com.pl/data/t4txt_photos/121758210119_1_BiTLAN_U_UTP_cat_5e_200_MHz.pdf
[109] Strona internetowa przedsiębiorstwa wytwarzającego obwody drukowane PCB https://www.nanotech-elektronik.pl/index.php/en/info/materials, (dostęp 03.03.2022r.)
[110] Strona internetowa poświęcona czujnikom glebowym z artykułami i referencjami naukowymi https://soilsensor.com/articles/dielectric-permittivity/ (dostęp 03.03.2022r.)
[111] Strona www z samouczkami z elektroniki analogowej, https://www.electronics-tutorials.ws/pl/dioda/diody-mocy.html,(dostęp 04.02.2022r.)

[112] Karta katalogowa rodziny układów 30XX, https://www.onsemi.com/pdf/datasheet/moc3023m-d.pdf, (dostęp 01.02.2022r.)
[113] Irmak S., *Perspectives and Considerations for Soil Moisture Sensing Technologies and Soil Water Content- and Soil Matric Potential-Based Irrigation Trigger Values,* Institute of Agriculture and Natural Resources at the University of Nebraska, Lincoln, 2019r., s.1-6
[114] Dokumentacja języka PHP, *PHP Manual,* https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php (dostęp 23.10.2021r.)
[115] Klamkowski K.,Treder W., *Metody pomiaru wilgotności gleby*, Instytut Sadownictwa i Kwiaciarstwa w Skierniewicach, Hasło Ogrodnicze nr 06/2010, s.12-14
[116] Bilskie J., *Soil water status: content and potential*, Campbell Scientific, Inc., https://s.campbellsci.com/documents/us/technical-papers/soilh20c.pdf (dostęp 22.10.2021r.)
[117] Pawłowska M., Wysocka A., *Laboratorium:Gleboznawstwo i rekultywacja*, Wydział Inżynierii Środowiska, Lublin, http://wis.pol.lublin.pl/pliki/gleby.pdf
[118] Karta katalogowa ekspander wyprowadzeń MCP23017, https://ww1.microchip.com/downloads/en/devicedoc/20001952c.pdf (dostęp 26.01.2022r.)
[119] Oficjalna strona internetowa wytwórcy obwodów drukowanych JLCPCB, https://jlcpcb.com/, (dostęp 10.05.2022r.)
[120] Portal dla deweloperów aplikacji internetowych, https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication, (dostęp 20.05.2022r.) https://datatracker.ietf.org/doc/html/rfc7235, (dostęp 20.05.2022r.)
[121] Dokumentacja dla deweloperów Android dotycząca WebView, https://developer.android.com/develop/ui/views/layout/webapps/webview (dostęp 20.05.2022r.)
[122] Centrum Zasad dla deweloperów, https://play.google.com/about/developer-content-policy/, (dostęp 20.05.2022r.)
[123] Wytyczne dotyczące recenzji w AppStore dotyczące minimalnej funkcjonalności aplikacji, https://developer.apple.com/app-store/review/guidelines/#minimum-functionality, (dostęp 20.05.2022r.)
[124] Oficjalna dokumentacja dla deweloperów Android dotycząca pliku manifest, https://developer.android.com/guide/topics/manifest/uses-sdk-element (dostęp 20.05.2022r.)
[125] Dokument RFC dotyczący *HTTP Authentication*, https://www.rfc-editor.org/rfc/rfc2617, (dostęp 24.05.2022r.)
[126] Oficjalna dokumentacja dotycząca formatu SVG ze strony World Wide Web Consortium W3C, https://www.w3.org/TR/SVG2/, (dostęp 22.05.2022r.)
[127] Oficjalna strona www biblioteki programistycznej JavaScript, https://jquery.com/, (dostęp 22.05.2022r.)
[128] Strona www dotycząca frameworka do testów jednostkowych Unity, https://docs.platformio.org/en/stable/advanced/unit-testing/frameworks/unity.html, (dostęp 25.04.2022r.)
[129] Oficjalna dokumentacja programu Postman, https://learning.postman.com/docs/getting-started/, (dostęp 26.04.2022r.)