

Politechnika Opolska

Wydział Elektrotechniki, Automatyki i Informatyki

Katedra Informatyki

PRACA DYPLOMOWA

Studia pierwszego stopnia
niestacjonarne

Kierunek studiów
Informatyka

**PROJEKT I WYKONANIE SYSTEMU MONITOROWANIA I
NAWADNIANIA ROŚLINNOŚCI Z APLIKACJĄ MOBILNĄ**

Promotor:
dr inż. Michał Podpora

Pracę wykonał:
mgr inż. Jakub Mońska
nr albumu: 94571

Opole, wrzesień 2020

PROJEKT I WYKONANIE SYSTEMU MONITOROWANIA I NAWADNIANIA ROŚLINNOŚCI Z APLIKACJĄ MOBILNĄ

S t r e s z c z e n i e

W obecnych czasach kładzie się coraz większy nacisk na optymalne zarządzanie wodą. Dotyczy to przede wszystkim rolnictwa, ale także małych przydomowych ogrodów. Typowe komercyjne systemy są bardzo drogie, co uniemożliwia przeciętnemu użytkownikowi wdrożenie tego typu systemu. Przedstawione rozwiązanie wykorzystuje ideę Internetu Rzeczy, pozwalając na monitorowanie i kontrolowanie nawodnienia za pomocą Arduino, bazy danych oraz aplikacji mobilnej a przy tym mieści się w zasięgu cenowym każdego pracującego użytkownika. Oprócz zaprojektowania w pełni funkcjonalnego systemu określono cel pracy, wprowadzono do tematu Internetu Rzeczy oraz przygotowano przegląd dostępnych rozwiązań.

DESIGN AND IMPLEMENTATION OF A VEGETATION AND IRRIGATION MONITORING SYSTEM WITH A MOBILE APPLICATION

S u m m a r y

Nowadays, more and more emphasis is placed on optimal water management. This applies primarily to agriculture, but also to small home gardens. Typical commercial systems are very expensive, making it impossible for the average person to implement this type of system. The presented solution uses the idea of the Internet of Things, allowing to monitor and control hydration using the Arduino, database and mobile application and at the same time it is within the price range of each working user.

Spis treści

1.	Wprowadzenie.....	4
2.	Cel i zakres pracy.....	5
3.	Wstęp teoretyczny.....	7
3.1.	Przegląd literatury	7
3.2.	Przegląd istniejących rozwiązań	9
4.	Wybór i prezentacja technologii	15
4.1.	Arduino.....	15
4.2.	Qt	16
5.	Projekt	18
5.1.	Zalożenia projektowe	18
5.2.	Wizualizacja systemu w przestrzeni	21
5.3.	Projekt prototypu.....	22
6.	Implementacja modułu sterującego	26
6.1.	Komunikacja z czujnikiem BMP280	28
6.2.	Komunikacja z czujnikami wilgotności.....	28
6.3.	Komunikacja z bazą danych	29
7.	Implementacja aplikacji mobilnej.....	31
7.1.	Interfejs użytkownika.....	32
7.2.	Komunikacja z bazą danych	35
8.	Testowanie	38
9.	Podsumowanie i możliwości dalszego rozwoju.....	40
10.	LITERATURA.....	41

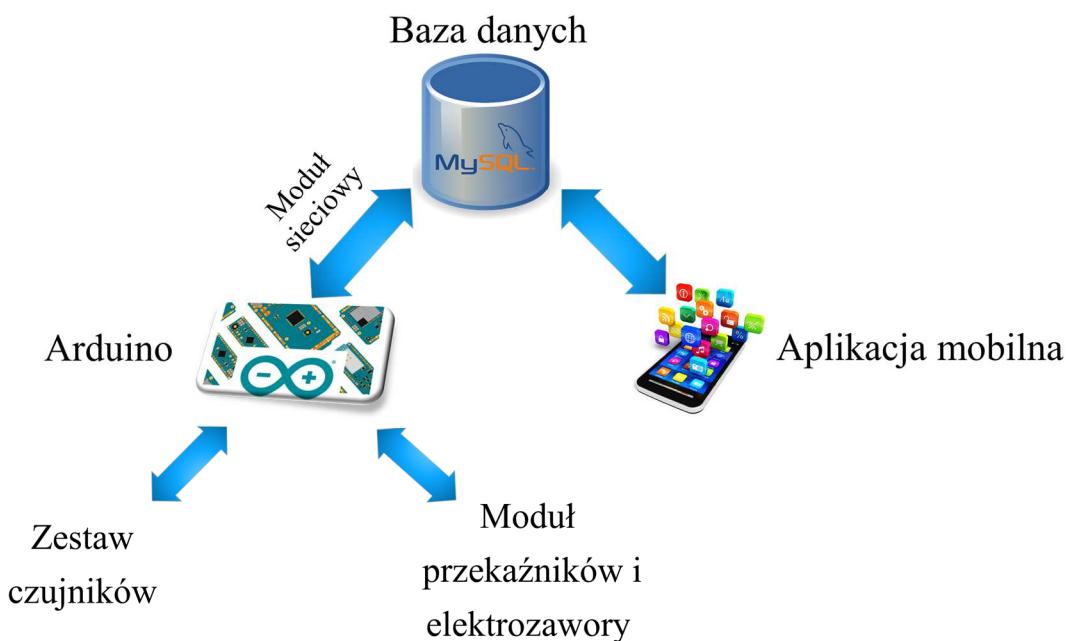
1. Wprowadzenie

Początkowo internet został zaprojektowany jako narzędzie komunikacji. W toku rozwoju stał się wszechobecny, dając prawie nieograniczone możliwości. Nie jest łatwo przewidzieć jak w przyszłości rozwinie się jego zastosowanie. Dowodem tej tezy mogą być słynne wypowiedzi znanych autorytetów z przeszłości. W latach siedemdziesiątych ubiegłego wieku założyciel i długotrwały prezes firmy Digital Equipment Corporation- Ken Olson publicznie stwierdził, że „nie ma najmniejszego powodu, by ktokolwiek odczuwał potrzebę instalowania komputera w swoim domu”. Dekadę później, obecnie chyba najbardziej znany informatyk na świecie miał powiedzieć „640 kB pamięci powinno wystarczyć każdemu”. Mowa o Billu Gatesie, który oficjalnie zaprzeczył temu sformułowaniu. Prawdopodobnie można przytoczyć jeszcze wiele takich cytatów, potwierdzając, że nawet najwybitniejsi przedstawiciele branży komputerowej jeszcze w trakcie swojego życia nie spodziewali się takiego postępu. Dziś prawie każdy człowiek na świecie posiada w zasięgu wzroku komputer, który za pomocą internetu przesyła różnego rodzaju dane.

Każdego dnia znacząco rośnie liczba urządzeń mogących łączyć się i komunikować ze sobą. Niemal każdy przedmiot może zostać podłączony z Internetem, nawet jeśli nie został wyprodukowany z myślą o IoT (ang. Internet of Things, pol. Internet rzeczy). Rozszerzenie jego funkcjonalności nie stanowi problemu nawet w przypadku butów, które przykładowo mogą zliczać ilość kroków i wyświetlać rezultaty na wykresie. Spowodowane jest to chęcią poprawienia komfortu życia oraz gromadzenia i analizowania danych. Powstało już wiele rozwiązań i aplikacji IoT, które przetwarzają tysiące petabajtów danych dziennie. Internet rzeczy aktualnie wdrażany jest w każdej dziedzinie życia. Właśnie te, wydaje się nieograniczone możliwości, były inspiracją do zrealizowania projektu w ramach przedstawionej pracy inżynierskiej.

2. Cel i zakres pracy

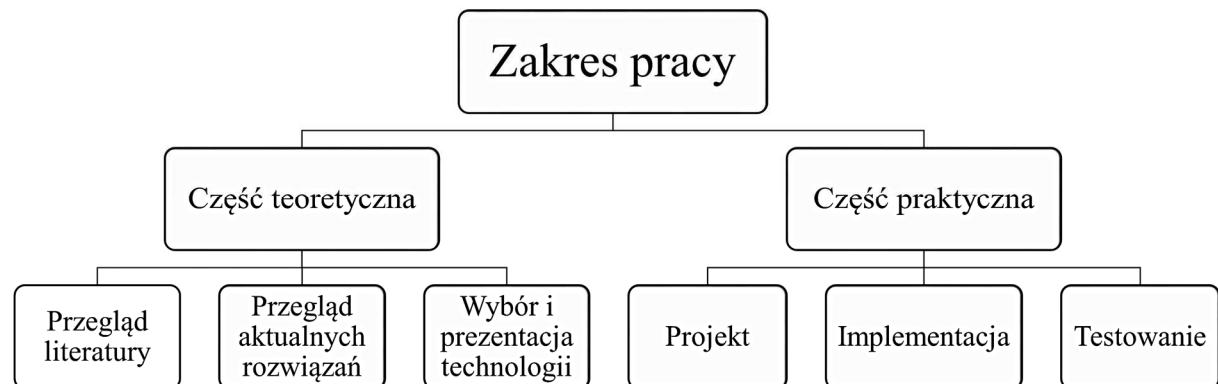
Celem pracy inżynierskiej było zaprojektowanie oraz wykonanie w pełni automatycznego systemu służącego do monitorowania i nawadniania roślinności. Do tego celu wykorzystano płytę z mikrokontrolerem ATmega 2560, odpowiednich czujników, modułu sieciowego, modułu przekaźnikowego oraz elektrozaworów. Kompletny wykaz urządzeń stanowi załącznik nr 1 do niniejszej pracy. Dzięki czujnikom system rejestruje dane takie jak temperatura, wilgotność, ciśnienie i wysyła je do bazy danych. Aby uzyskać optymalne warunki do uprawy roślin napisano aplikację mobilną, za pomocą której można monitorować oraz sterować systemem. Ogólny uproszczony schemat działania systemu przedstawiono na rys. 2.1.



Rys. 2.1. Uproszczony schemat funkcjonowania systemu

Wałąną cechą systemu jest jego prostota i uniwersalność, zapewniając możliwość kontrolowania sytuacji z każdego miejsca na Ziemi, niemal ze wszystkich urządzeń podłączonych do Internetu. Bez trudności dokładając kolejne czujniki lub linie kodu można rozszerzyć jego funkcjonalność. Dodatkowo system powinien minimalizować koszty zużycia wody oraz jako całość mieścić się w zasięgu cenowym każdego pracującego użytkownika.

Niniejszą pracę podzielono na dwie części zgodnie z rys. 2.2. W pierwszej części pracy – części teoretycznej – przedstawiono aktualny stan wiedzy oraz przykłady podobnych komercyjnych i niekomercyjnych systemów. Opisano również wykorzystaną technologię.



Rys. 2.2. Zakres pracy

Drugą stanowi część praktyczna, w której przedstawiono powstawanie całego systemu od projektu, poprzez implementację, aż do testowania.

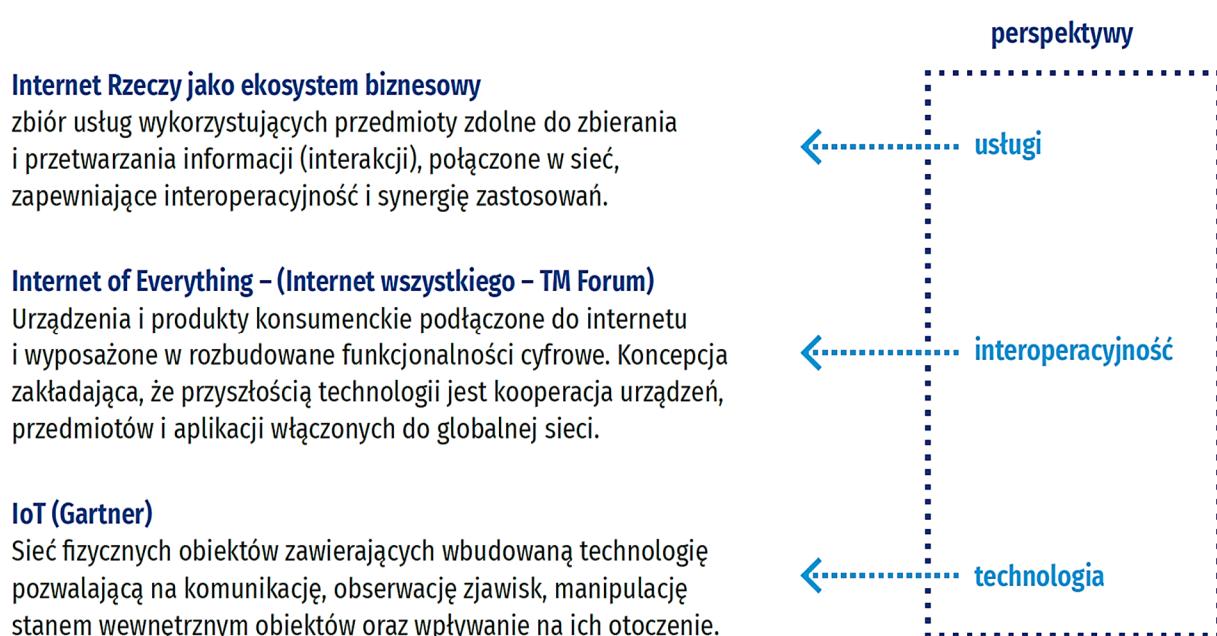
3. Wstęp teoretyczny

3.1. Przegląd literatury

Koncepcja IoT nie jest nowa, ponieważ została zaproponowana na prezentacji K. Ashtona dla koncernu Procter & Gamble (P&G) jeszcze w 1999 r. [2] Termin ten został wtedy powiązany z wykorzystaniem Radiowych Systemów Automatycznej Identyfikacji w łańcuchach dostaw P&G. Oczywiście, aby zapewnić odpowiednią chronologię na osi czasu należy cofnąć się wiele lat wstecz i poszukiwać genezy IoT wraz z pojawiением się Internetu przez ARPANET [8], a nawet przy powstawaniu pierwszej elektroniki takiej jak telegraf elektromagnetyczny. Choć często za pierwszą rzeczą z rodziny Internetu Rzeczy uważa się toster podłączony do Internetu za pomocą sieci TCP/IP stworzony przez Johna Romkey'a i Simona Hacketta w 1990 roku [18], to jednak pierwszym oficjalnym urządzeniem internetowym była maszyna do sprzedawy napojów Coca-Cola na Carnegie Mellon University. Powstała w 1982 roku i korzystający z sieci programiści mieli możliwość sprawdzenia stanu maszyny oraz tego, czy w automacie czeka na nich zimny napój.

W czasie ostatnich dwóch dekad zaproponowano wiele definicji IoT i obejmowały coraz szerszy zakres zastosowań, takich jak opieka zdrowotna, usługi komunalne, transport, itd. [7]. Bliskość spojrzenie rozbiija ten termin na dwa ważne filary: „Internet” i „Rzeczy”, lecz te wyrazy nie wyjaśniają zawiłości terminu. Wydaje się, że każdy obiekt zdolny do połączenia się z globalną siecią będzie należeć do kategorii „Rzeczy”. Ma on bardziej ogólny sens – odnosi się bowiem do inteligentnych sensorów, urządzeń i wszelkich obiektów, które są w stanie komunikować się z innymi podmiotami w dowolnym czasie i miejscu [3]. „Rzeczą” w IoT może być zarówno wszczepiony implant do monitorowania serca, zwierzę gospodarskie z transponderem biochip, pojazd z wbudowanymi czujnikami ostrzegającymi kierowcę, gdy ciśnienie w oponie jest niskie lub jakimkolwiek innym naturalnym lub wymyślonym przez człowieka obiektem, któremu można przypisać adres IP i jest w stanie przesyłać dane przez sieć [10]. Wszechobecna łączność jest kluczowym wymogiem. Aby go spełnić aplikacje muszą być wspierane przez całą gamę urządzeń i protokołów komunikacyjnych, od małych przyrządów pomiarowych raportujących parametry, przez urządzenia sieciowe takie jak router, aż do potężnych back-endowych serwerów.

Podczas zorganizowanej przez Ministerstwo Cyfryzacji konferencji „Internet Rzeczy – Polska Przyszłość” (2 lipca 2019 r.) zaprezentowano raport „IoT w polskiej gospodarce” [11]. Nad rapportem od 24 sierpnia 2018 r. pracowali eksperci wywodzący się ze środowisk biznesowych, akademickich oraz instytucji państwowych. Podeszli do definicji IoT z trzech perspektyw, które w skróconej wersji można zobaczyć na rysunku 3.1.



Rys. 3.1. Definicje IoT z różnych perspektyw [11]

Na całym świecie w roku 2018 istniało około 22 miliardów urządzeń podłączonych w ramach IoT. Prognozy sugerują, że do 2030 roku na całym świecie liczba urządzeń wzrośnie do około 50 miliardów [14]. Urządzenia te wykorzystywane są w wielu różnych branżach, gdzie liderami wzrostów i sumy wydatków w obszarze IoT są: sektor konsumencki, transport oraz przemysł. W tabeli 3.1 zestawiono wydatki w poszczególnych branżach.

Tabela 3.1. Wydatki w obszarze IoT w poszczególnych branżach [mln \$], czerwiec 2018 [11]

Sektor	Rok					
	2017	2018	2019	2020	2021	2022
Sektor konsumencki	287.8	345.2	412.5	492.5	581.5	677.6
Transport	338.9	367.4	404.2	450.3	511.4	585.2
Produkcja ciągła	208.5	234.9	270.3	313.9	366.6	429.0
Media użytkowe	258.4	269.9	284.3	301.3	320.8	343.6
Produkcja dyskretna	196.4	224.6	260.5	307.4	357.0	421.3
Handel detaliczny	126.5	147.0	173.1	201.9	237.1	279.0
Usługi dla biznesu	106.6	124.4	149.9	178.6	215.0	269.7
Władze krajowe/lokalne	72.8	81.6	91.6	104.3	119.9	140.0
Usługi dla ludności	56.2	64.9	77.4	95.9	118.3	149.6
Sprzedaż hurtowa	56.1	64.3	77.3	92.8	108.6	127.2
Władze federalne/centralne	58.0	66.1	76.9	89.3	101.7	118.3
Ochrona zdrowia	62.2	66.8	73.2	80.5	90.7	104.9
Edukacja	29.8	35.0	43.8	54.1	65.1	77.6
Budownictwo	31.8	35.8	41.0	48.5	56.6	65.9
Telekomunikacja	28.1	31.9	36.9	43.2	50.0	57.9
Media	26.4	29.9	34.8	42.1	49.5	58.6
Przemysł wydobywczy	17.0	19.7	22.8	26.6	31.1	36.4
Ubezpieczenia	8.7	10.6	13.3	17.4	22.1	28.4
Papiery wartościowe i usługi inwestycyjne	6.6	7.2	8.2	9.2	10.7	12.3
Bankowość	3.4	4.2	5.0	5.9	7.1	8.1
Suma końcowa	1980.1	2231.1	2557.2	2955.8	3420.4	3991.7

Choć prognozy analityków i firm badawczych mogą się różnić, każdy z nich jest zgodny z faktem, iż wartość rynku Internetu Rzeczy bardzo szybko wzrasta i nie zamierza, póki co, spowalniać. Bezpośrednie korzyści finansowe, wynikające z szerokiego zastosowania Internetu Rzeczy odnaleźć można na trzech poziomach dojrzałości [6]:

- „Data To Discovery”, gdzie analizując otrzymane dane odkryto zjawiska o których istnieniu dotąd nie wiedziano np. przyczynę choroby odkrytą dzięki szczegółowym danym pochodzący z urządzeń medycznych,
- „Data To Decisions”, gdzie na bazie otrzymanych danych można podjąć decyzję lub decyzja wykonywana jest automatycznie. Przykładem może być komunikat dla użytkownika lub wyłączenie silnika w przypadku awarii,
- „Data To Dollars-Dividends”, gdzie z połączenia dwóch wcześniejszych umiejętności wyłania się nowa korzyść finansowa lub szansa rozwoju biznesu. Są to innowacyjne produkty bądź usługi, które powstały dzięki idei Internetu Rzeczy np. „Smart Home”

Tak szybki rozwój technologii otwiera drogę na coraz bardziej nowatorskie i zaawansowane rozwiązania ułatwiające życie. Nowe systemy skutecznie radzą sobie z podejmowaniem decyzji w czasie rzeczywistym na szybkich i dużych strumieniach danych.

3.2. Przegląd istniejących rozwiązań

Istnieją różne opinie na temat skutków i istnienia zmiany klimatu na Ziemi. Nie ma jednak wątpliwości, że wiele państw, firm i zwykłych podatników zwiększa swoją świadomość dotyczącą zrównoważonej konsumpcji. Wpływa to nie tylko na ich poglądy na temat marnowania zasobów, ale także doprowadza do powstania wielu urządzeń oraz publikacji naukowych dotyczących między innymi zmniejszenia zużycia wody przeznaczonej do nawadniania. Co więcej, w coraz bardziej intensywnym życiu ciężko jest znaleźć czas na tak proste zadanie jak podlewanie. Oczekuje się, że nowoczesne automatyczne systemy z rodziny IoT staną się rozwiązaniem tego problemu. Wszystko to jest możliwe dzięki postępowi, jaki dokonał się w dziedzinie elektroniki, co w rezultacie doprowadziło do powstania wielu małych, energooszczędnego czujników i urządzeń komunikacyjnych.

Zarówno w projektowaniu jak i programowaniu systemów nawadniania nie można znaleźć jednej najlepszej ścieżki. Poszukując odpowiedzi w książkach można natknąć się na bardzo dużą ilość przydanych informacji i przykładów, lecz zebraną wiedzę należy łączyć w całość małymi krokami. Przykładowo budując system nawadniania oparty o IoT nie znajdzie się gotowego idealnego przepisu. Stopniowo trzeba wglądać się w realizowany temat zaczynając od budowy płytka z mikrokontrolerem, poprzez odczytywanie napięcia z czujników analogowych, aż po komunikację sieciową. Następnie w oparciu o zdobytą wiedzę, sumując wady i zalety, należy wybrać optymalne rozwiązanie. Co innego można natomiast znaleźć w Internecie. W rzetelnych źródłach informacji, takich jak artykuły naukowe, jest wiele podobnych systemów lub nawet bazujących na dokładnie tych samych elementach, lecz są to na ogół ogólne informacje pokazujące same funkcjonalności. Najwięcej gotowych rozwiązań przedstawia społeczność hobbystów, którzy dzielą się swoimi

projektami, dokładnie omawiając sposób działania i niejednokrotnie udostępniając kod źródłowy z odpowiednimi komentarzami.

Jako że problem nawadniania towarzyszy nam już od starożytności, wielowiekowy rozwój doprowadził do powstania różnych systemów irygacji. W omawianych przykładach nie przedstawiono pomysłowych metod nawadniania, natomiast skupiono się nad rozwiązaniami sterującymi. Obecnie oferowanych jest już wiele rozwiązań, które pozwalają w mniejszym lub większym stopniu, oraz przy różnym nakładzie pracy monitorować i kontrolować system nawadniania. Przeglądając te rozwiązania wybrały kilka, aby pokazać co oferują, na jakiej zasadzie działają oraz jak je obsługiwać. Ze względu na znacznie odbiegające od siebie pod względem wizualnym oraz cenowym rozwiązania te podzielono na dwie grupy.

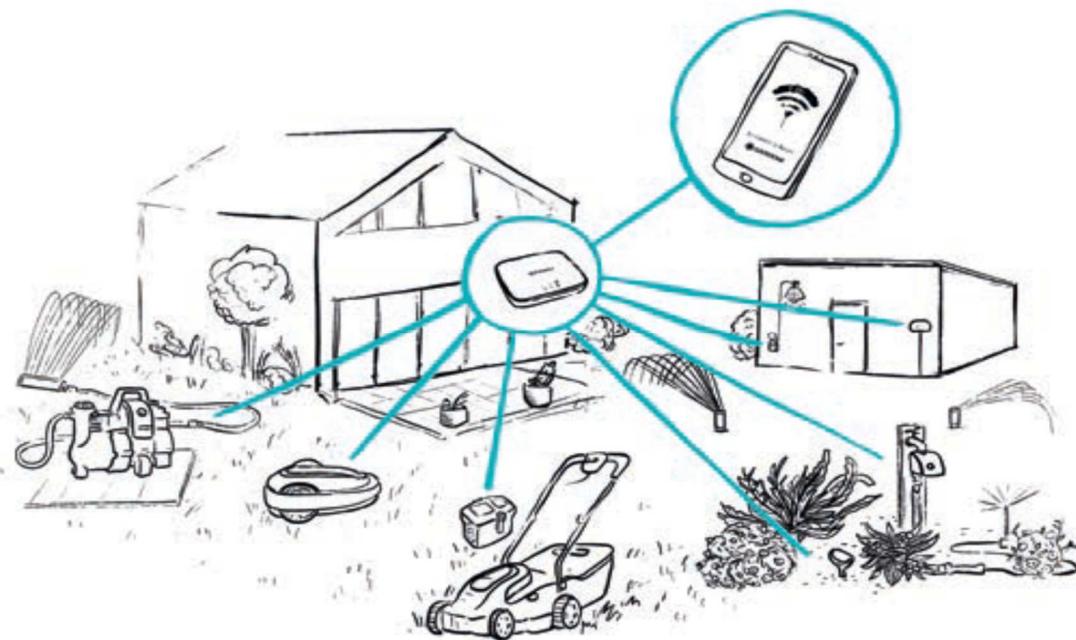
3.2.1. Rozwiązania komercyjne

Spora część osób zainteresowanych tematem nawadniania podejmuje się go na własną rękę. Nic dziwnego, tak jak wspomniano, można znaleźć coraz więcej informacji, filmików i poradników dotyczących tworzenia takich systemów. Ma to swoje wady i zalety. Korzystając z usług firm z długim stażem oczekuje się wiedzy i doświadczenia, które przełoży się na zadowalające wyniki. Najczęściej sam projekt wliczony jest w cenę instalacji systemu a tworzony jest po pierwszej wizycie specjalistów lub po przesłaniu planu działki z zaznaczonymi źródłami wody i niezbędnymi informacjami na temat parametrów tych źródeł. Oczywiście projekt jest przygotowywany indywidualnie na podstawie wymagań użytkownika. Zaprojektowany system może być zainstalowany przez wykonawcę bądź dostarczony razem z instrukcją do samodzielnego montażu. Jeśli montaż został zamówiony w profesjonalnej firmie, na działanie instalacji otrzyma się gwarancję oraz można również liczyć na kwalifikowany serwis i darmowe przygotowanie systemu do zimy w pierwszym roku. Trzeba liczyć się jednak z faktem, iż gotowe rozwiązania nie należą do najtańszych.

Należy zaznaczyć, że różnych systemów i poszczególnych ich elementów jest bardzo wiele. Każdy rodzaj systemu ma swoje zastosowanie oraz swoje wersje. Przy małych projektach bardzo często stosuje się sterowniki oparte tylko o czasowe otwarcie zaworów, zależnie od zaprogramowanego harmonogramu. Niekiedy można dokupić do takiego sterownika czujnik opadu deszczu, aby nie uruchamiać nawadniania dodatkowo w trakcie opadów. Takie rozwiązanie jest stosunkowo tanie, lecz nie zapewnia optymalnego nawadniania a tym bardziej nie pozwala na kontrolę z każdego miejsca na ziemi. Dlatego poniżej wybrano tylko niektóre, wydające się przez autora tekstu, ciekawe rozwiązania systemów opartych o Internet Rzeczy.

3.2.1.1. Gardena – smart system

System ten daje duże możliwości, ponieważ dołączona aplikacja umożliwia dostęp do zamontowanego na stałe systemu nawadniania, ma możliwość sterowania swoim robotem koszącym oraz oświetleniem zewnętrznym. Działa na smartfonach z systemem iOS i Android oraz w przeglądarce internetowej. W pakiecie jest sterowanie, analiza danych oraz źródło wiedzy na tematy ogrodnicze. Na rysunku 3.2. przedstawiono schemat działania tego systemu.



Rys. 3.2. Schemat działania systemu Gardena [9]

Sercem tego zestawu jest router zapewniający bezprzewodową komunikację sieciową pomiędzy urządzeniami „smart Gardena”, a domowym Internetem. Reszta urządzeń wybierana jest na podstawie preferencji użytkownika. W każdej chwili można rozbudować system o kolejne urządzenia takie jak:

- Jednokanałowy sterownik nawadniania, który jest podłączony bezpośrednio do kranu. Pozwala nawadniać jeden obszar. Może współpracować ze „smart Sensorem” (koszt 732 zł),
- Wielokanałowy sterownik nawadniania, który pozwala nawadniać maksymalnie 6 obszarów za pomocą zaworów 24 V. Również może współpracować ze „smart Sensorem” (koszt 1288 zł),
- Smart Sensor, który pozwala na bezprzewodowy pomiar światła, temperatury powietrza i wilgotności gleby w wyznaczonym miejscu. Aplikacja może wykorzystywać wyniki pomiarów oraz inne parametry do sterowania nawadnianiem za pomocą sterownika nawadniania lub sterownika wielokanałowego (koszt - 463 zł),
- Hydrofor elektryczny, który działa jako programowalna pompa ogrodowa. Umożliwia przesyłanie informacji do aplikacji na temat ciśnienia przepływu i ewentualnych usterek (koszt 2355 zł),
- Wtyczka elektryczna, która służy do integracji innych urządzeń elektrycznych z „Gardena smart system” (koszt 245 zł).

3.2.1.2. Rain Bird

System ClimateMinder™ firmy Rain Bird to rozwiązanie przeznaczone do monitorowania i sterowania w rolnictwie. Informacje z systemu można odbierać na telefonie komórkowym, na komputerze lub w systemach sterowania. ClimateMinderTM umożliwia:

- Monitorowanie pola i warunków wzrostu upraw w czasie rzeczywistym - temperatury i wilgotności otoczenia, wilgotności gruntu, napięcia, temperatury gruntu, przewodności elektrycznej, promieniowania słonecznego, wilgotności liści, ciśnienia wody, przepływu wody, wiatru i opadów deszczu,
- Sterowanie pompą nawadniającą, zaworami i wtryskiwaczami nawozu za pomocą telefonu, komputera lub tabletu.
- Natychmiastowe powiadamianie o zmianie warunków w terenie.
- Odbieranie i wysyłanie dziennych lub tygodniowych raportów dotyczących warunków terenowych.

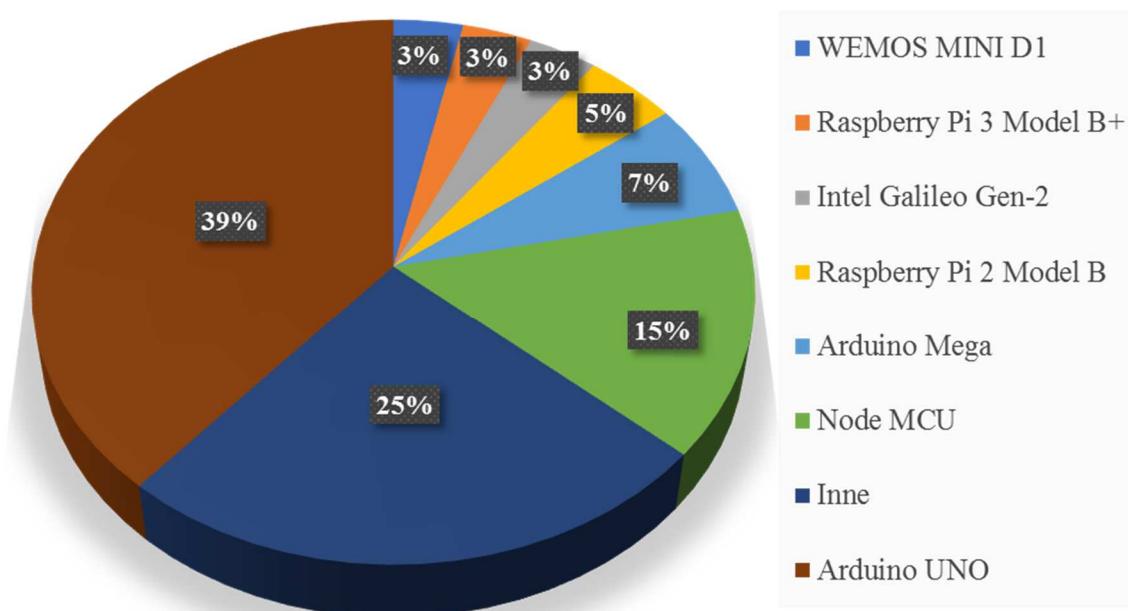


Rys. 3.2. System ClimateMinder™ firmy Rain Bird [15]

System opiera się o sieć bezprzewodową umożliwiającą współpracę czujników terenowych i elementów sterowania. Obejmuje automatycznie regulującą się bezprzewodową sieć węzłów (z czujnikami lub sterownikami, które można podłączać do każdego węzła) oraz bramki komunikującej się z serwerem internetowym ClimateMinder™. Jest to miejsce, w którym przechowywane są wszystkie dane oraz odbywa się sterowanie systemu.

3.2.2. Rozwiązania niekomercyjne

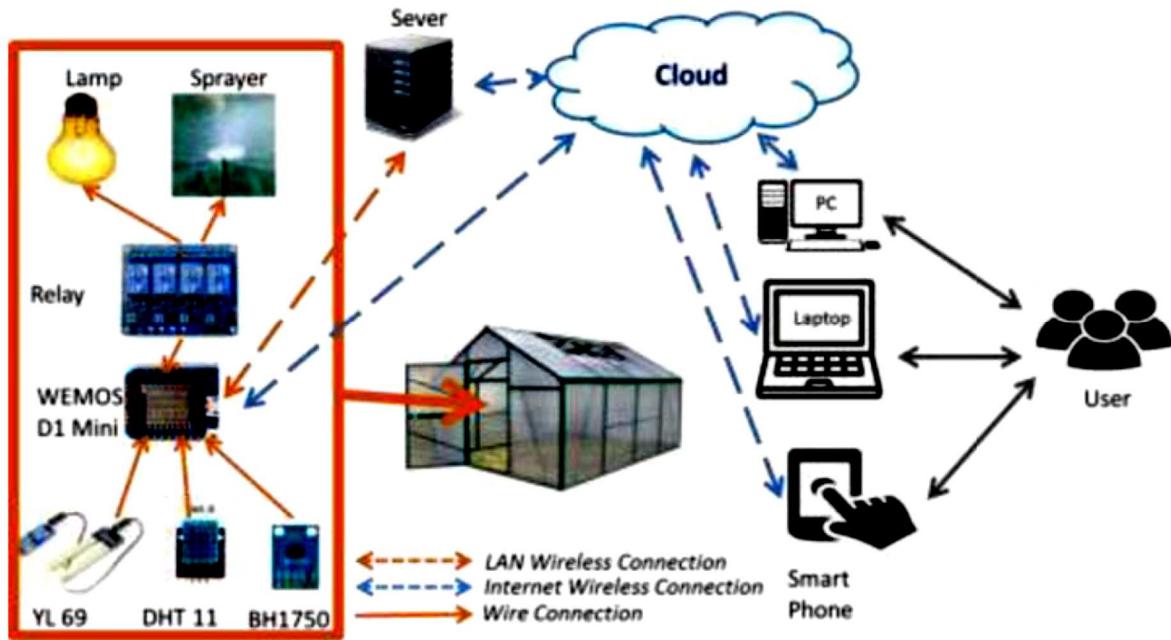
Jak już wcześniej wspomniano najwięcej gotowych rozwiązań podobnych systemów przedstawia społeczność hobbystów, którzy dzielą się swoimi projektami. Są to tak zwane projekty DIY (ang. - do it yourself), które wykonane są zwykle na własne potrzeby bez pomocy specjalistów. Dzięki tym twórczym hobbystom powstała nieskończona ilość systemów przeznaczonych do nawadniania. W artykule o nazwie „Inteligentne systemy nawadniania oparte na IoT: przegląd najnowszych trendów w zakresie czujników i systemów IoT do nawadniania w rolnictwie precyzyjnym”[5] przedstawiono najczęściej wykorzystywane elementy do realizacji systemów nawadniających opartych na Internecie Rzeczy. Autorzy tekstu sporządzili statystyki na podstawie znalezionych artykułów naukowych. Poniższy rysunek przedstawia najczęściej wykorzystywane płytki z mikrokontrolerem do realizacji tych systemów.



Rys. 3.3. Najczęściej wykorzystywane produkty do wdrażania systemów nawadniania [5]

Wiele też projektów nie wykorzystuje gotowych płyt z mikrokontrolerem, takich jak te przedstawione powyżej i decyduje się na zbudowanie systemu dobierając każdy najmniejszy element samemu. Ma to swoje zalety, ponieważ daje to możliwość opracowania własnych rozwiązań dopasowanych do szczegółowych wymagań.

Jednym z niekomercyjnych przykładów do nawadniania może być projekt inżynierów z Indonezji, którzy zaprojektowali system do sterowania, nawadniania i monitoringu przy uprawie chryzantemy. Łączenie się ze sterownikiem realizuje się za pomocą telefonu komórkowego bądź komputera dzięki Wemos D1 mini oraz modułowi ESP8266. Na rysunku 3.4. zobrazowano końcowy efekt i sposób działania.



Rys. 3.4. Przykład automatycznego systemu [4]

W tym projekcie za pomocą odpowiednich czujników mierzono w szklarni: temperaturę, wilgotność gleby, wilgotność powietrza i moc naświetlania lamp. Dane z czujników przesyłano w czasie rzeczywistym do bazy danych. Ponadto system był w stanie zdalnie sterować oświetleniem i pompami wody.

4. Wybór i prezentacja technologii

4.1. Arduino

Arduino powstało jako pomoc dydaktyczna dla włoskich studentów, lecz szybko stało się bazą do nauki elektroniki i programowania dla wielu osób na całym świecie. Nastąpiło to na skutek faktu, iż ta platforma typu open source jest oparta na sprzęcie i oprogramowaniu łatwym w obsłudze, a koszt jaki należy ponieść przy zakupie jest stosunkowo niewielki. Aby zaprogramować swój układ nie jest wymagana szczególna wiedza na temat mikrokontrolerów, ponieważ do zaimplementowania nawet skomplikowanego programu wystarczy skorzystać z ogólnodostępnych bibliotek napisanych w języku zbliżonym do C/C++. Takie podejście nazywane jest „czarną skrzynką” – nie uczy zasad działania mikrokontrolera i podłączonych do niego elementów, a jedynie ich wykorzystanie. Jest to idealny sposób do szybkiego budowania i testowania prototypów. Duża ilość użytkowników również ułatwia zadanie w znalezieniu pomocy w razie kłopotów. Kolejną zaletą jest to, że nie wymaga zewnętrznego programatora, a bez najmniejszych kłopotów tworzy spójną całość z dedykowanym kompilatorem. Oprogramowanie to jest dystrybuowane na licencji GPL (ang. General Public License, pl. Powszechna Licencja Publiczna), oznacza to, że aby móc legalnie z niego korzystać, trzeba cały napisany kod publikować za darmo.

Wadą środowiska jest brak wbudowanego debuggera, a jedyną formą służącą do analizy jest wykorzystywanie funkcji wyświetlania tekstu za pomocą interfejsu szeregowego. Dodatkowo brakuje w nim systemów sprawdzania składni, autouzupełniania i innych przydatnych funkcji, które można znaleźć w znanych środowiskach IDE (ang. Integrated Development Environment, pl. Zintegrowane Środowisko Programistyczne). Natomiast gotowe biblioteki są uniwersalne - działają na każdej płytce, przez co są słabo prezentują się pod kontem wydajności. Oczywiście istnieje możliwość edycji lub tworzenia własnych bibliotek, lecz wtedy poziom trudności znaczaco wzrasta. Mimo, że można znaleźć zastosowanie Arduino w projektach komercyjnych, rzadko kiedy są stosowane ze względu na brak testów i certyfikatów.

Do płytki bazowej mogą być podłączone liczne moduły oraz płytki rozszerzające, zawierające elementy zróżnicowane pod względem poziomu skomplikowania, takie jak czujniki, elementy wykonawcze, sterowniki magistral itp. Na rynku oprócz oryginalnych produktów Arduino jest dostępnych wiele „klonów”, które w mniejszym lub większym stopniu odzwierciedlają oryginał. Są znacznie tańsze, lecz trzeba liczyć się z tym, że część z nich może mieć zmieniony schemat lub wykonane są z gorszych jakościowo materiałów.

4.2. Qt

Qt jest wieloplatformowym frameworkiem napisanym w języku C++ i posiadającym w pełni obiektową architekturę. Raz napisaną aplikację można bez większych problemów zbudować na Windowsie, Linuksie, urządzeniu z Androidem, iOS lub innym. Qt w pakiecie instalacyjnym posiada ułatwiające pracę narzędzia takie jak:

- Qt Creator – zintegrowane środowisko programistyczne,
- Qt Designer – aplikacja do definiowania graficznego interfejsu użytkownika,
- UIC (User Interface Compiler) – narzędzie tworzące pliki nagłówkowe (.h) na podstawie pliku .ui w formacie XML, który zawiera definicję interfejsu użytkownika tworzonego w ramach modułu Qt Widgets,
- MOC (Meta Object Compiler) – specjalny preprocesor, który na podstawie plików nagłówkowych (.h) generuje dodatkowe pliki źródłowe (.cpp),
- qmake – narzędzie do zarządzania procesem komplikacji narzędzie oraz automatycznego generowania plików makefile, na których podstawie odbywa się komplikacja. Wykorzystuje moc oraz uic,
- Qt Assistant – rozbudowany system pomocy dla programistów,
- Qt Linguist – narzędzie wykorzystywane do tłumaczenia aplikacji.

Z uwagi na szeroką gamę zastosowań platformy programistycznej do projektu dochodzi się jedynie realnie wykorzystywane moduły realizujące poszczególne funkcjonalności. W tabeli 4.1 zestawiono podstawowe moduły odpowiedzialne za obsługę.

Tabela 4.1. Qt Essentials - podstawowe moduły wykorzystywane w aplikacjach [15]

Moduł	Opis
Qt Core	Podstawowe klasy niegraficzne używane przez inne moduły.
Qt GUI	Podstawowe klasy dla komponentów interfejsu graficznego. Obsługa OpenGL
Qt Multimedia	Klasy do obsługi audio, wideo, radia czy aparatu.
Qt Multimedia Widgets	Klasa rozszerzająca możliwości modułów Qt Multimedia oraz Qt Widgets
Qt Network	Klasy ułatwiające implementację komunikacji sieciowej.
Qt Qml	Klasy wspierające języki Qml oraz JavaScript
Qt Quick	Framework umożliwiający tworzenie interfejsu użytkownika wykorzystując język Qml
Qt Quick Controls	Dostarcza wygodnych w użyciu typów do wykorzystania podczas tworzenia interfejsu w języku Qml
Qt Quick Dialogs	Dostarcza typów do obsługi systemowych okien dialogowych w aplikacjach opartych na Qt Quick
Qt Quick Layouts	Dostarcza typy umożliwiające ułożenie w prosty sposób innych elementów interfejsu graficznego.
Qt Quick Test	Framework do testów jednostkowych dla aplikacji opartych na Qt Quick
Qt SQL	Klasy wykorzystywane do integracji z bazami SQL
Qt Test	Klasy do testów jednostkowych aplikacji napisanych w oparciu o Qt, ale też o czysty język C++.
Qt Widgets	Klasy rozszerzające moduł Qt GUI o widżety

Wymienione powyżej moduły to zaledwie mała część nazywana Qt Essentials, która jest najczęściej wykorzystywane na wszelakich platformach. Pozostałe moduły określane są jako Qt Add-Ons, a ich pełną listę można zobaczyć w naprawdę obszernej i dobrze napisanej dokumentacji. Od daty premiery Qt w roku 1995 środowisko wprowadziło wiele zmian, jeżeli chodzi o możliwości pisania w innych językach programowych. Z framework'a korzystać można w językach takich jak QML, Python, C#, Rust, Go oraz wielu innych, lecz z ich wsparciem bywa różnie [13]. Kolejną zaletą jest możliwość korzystania z Qt w wersji darmowej typu Open Source. Qt wydane jest na kilku licencjach w tym: GPL3, GPL2 i GPLv3. Zwłaszcza drugi typ licencji jest godny uwagi, ponieważ pozwala na zachowanie kodu źródłowego w tajemnicy przy jednoczesnym zarabianiu na projekcie.

4.2.1. QML oraz Qt Quick

QML jest językiem deklaratywnym, który umożliwia opisywanie interfejsów użytkownika pod względem ich elementów wizualnych oraz ich interakcji i relacji ze sobą. Jest to bardzo czytelny język, który został zaprojektowany w celu umożliwienia dynamicznego łączenia komponentów, a także umożliwia łatwe, ponowne wykorzystanie i dostosowywanie ich korzystając z interfejsu graficznego. Korzystając z QtQuick projektanci i programiści mogą łatwo tworzyć płynne animowane interfejsy użytkownika w QML i mają możliwość połączenia tych interfejsów użytkownika z dowolnymi bibliotekami z zaplecza C++ [1].

Moduł Qt Quick jest standardową biblioteką do pisania aplikacji QML. Podczas gdy moduł Qt QML zapewnia silnik QML i infrastrukturę językową, moduł Qt Quick zapewnia wszystkie podstawowe typy niezbędne do tworzenia interfejsów użytkownika z QML. Zapewnia wizualną kanwę i zawiera typy do tworzenia i animowania komponentów wizualnych, odbierania danych wejściowych od użytkownika, tworzenia modeli danych i widoków oraz tworzenia instancji obiektów [16].

4.2.2. Niezbędne narzędzia do pisania oprogramowania w QT pod system Android

Istotną częścią procesu tworzenia aplikacji pod system Android jest wybór odpowiedniej wersji QT wraz z komponentami podczas instalacji. W razie braków któregoś z komponentów istnieje możliwość skorzystania z „Qt Maintenance Tool” bez konieczności ponownej instalacji QT. Dalsza konfiguracja sprowadza się do zainstalowania kilku niezbędnych narzędzi do budowy aplikacji takich jak:

- JDK (Java Developer Kit),
- Android SDK (Android Software Development Kit),
- Android NDK (Android Native Development Kit),
- Gradle, dla Qt w wersji niższej niż 5.9 (wyższe instalują się automatycznie).

5. Projekt

5.1. Założenia projektowe

W tym rozdziale opisano wymagania funkcjonalne i niefunkcjonalne tak, aby precyzyjnie ustalić jakie usługi ma oferować system, jak ma się zachować w określonych sytuacjach oraz jakie posiada ograniczenia.

5.1.1. Wymagania ogólne

System przeznaczony jest do sterowania nawadnianiem zależnie od parametrów zdefiniowanych przez użytkownika. Do jego funkcji powinno należeć gromadzenie danych i udostępnianie ich użytkownikowi. System powinien posiadać budowę modułową, zapewniającą łatwość montażu i demontażu a także umożliwiać przystępna wymianę jego elementów, bez konieczności użycia specjalistycznych narzędzi.

5.1.2. Rejestrowanie, transmisja oraz przetwarzanie danych

System powinien umożliwiać okresowy tryb rejestracji danych, o częstotliwości zapisu możliwej do ustalenia indywidualnie dla każdego czujnika, z interwałem czasowym od 1 sekundy do 24 godzin. System powinien mieć możliwość tworzenia raportów na potrzeby dalszej edycji danych. Rejestrowane w czasie lokalnym dane powinny być zapisywane w bazie danych umieszczonej na serwerze. Zebrane dane powinny być weryfikowane pod kątem wartości granicznych w celu zapobiegania rejestracji wartości nierealnych, zgodnie z lokalnymi warunkami klimatycznymi. Ponadto, system powinien zapewniać możliwość wykonania obliczeń dla dowolnej zmiennej, a przy prezentacji danych możliwość wyboru jednostek pomiarowych (np. °C, °F).

5.1.3. Interfejsy urządzeń

System powinien posiadać co najmniej sześć (6) wejść analogowych z 10-bitowym przetwornikiem ADC oraz czternaście (14) cyfrowych kanałów wejść i wyjść a ponadto interfejs portu szeregowego. Funkcja komunikacji musi umożliwiać odbieranie danych z czujników oraz kontrolę działania systemu (w celach diagnostycznych). Minimalne interfejsy komunikacyjne:

- UART (ang. - Universal Asynchronous Receiver-Transmitter),
- I²C/TWI (ang. - Inter-Integrated Circuit/Two Wire Interface),
- SPI (ang. - Serial Peripheral Interface).

System powinien posiadać port, do którego można podłączyć komputer/laptop w celu przeprowadzenia ponownej konfiguracji, aktualizacji oprogramowania, odczytania zarejestrowanych danych i monitorowania poszczególnych modułów. Czynności te mogą zakłócać rejestrację danych. System powinien mieć interfejs sieciowy w standardzie Ethernet lub WIFI, który zapewnia dwukierunkową łączność przez sieć TCP/IPv4 bezpośrednio przez port Ethernet lub poprzezłączony modem. Dodatkowo powinien wspierać co najmniej protokół HTTP z obsługą żądań GET protokołu w wersji 1.0 i 1.1.

5.1.4. Środowisko pracy

System przeznaczony jest do pracy w warunkach letnich i przejściowych tzn. nie jest przeznaczony do pracy w warunkach zimowych, gdy temperatura powietrza spada poniżej 0°C. W celu zminimalizowania wpływu warunków atmosferycznych na niezawodność sprzętu oraz jakość danych system powinien być zdolny do pracy w co najmniej poniższych warunkach:

- temperatura powietrza: 0°C ÷ 50°C,
- zakres wilgotności powietrza: 0% ÷ 100% RH,
- prędkość wiatru min. do 50m/s,
- klasa obudowy min. IP44

System musi być zdolny do pracy ciągłej w powyższych warunkach tj. 24h na dobę.

5.1.5. Obudowa

Elementy elektroniczne systemu powinny być umieszczone w szczelnej obudowie z łatwym dostępem do wszystkich elementów. Obudowa powinna charakteryzować się co najmniej parametrami:

- klasa szczelności IP44,
- brak możliwości powstawania pary wodnej wewnętrz obudowy,
- wykonana z materiału odpornego na korozję oraz promieniowanie UV

5.1.6. Zasilanie

System powinien mieć możliwość zasilania z sieci energetycznej, akumulatorów lub paneli słonecznych.

5.1.7. Oprogramowanie

Oprogramowanie dołączone do systemu powinno mieć interfejs graficzny, umożliwiać zmianę ustawień systemu oraz pracować między innymi pod systemem operacyjnym Windows oraz Android.

5.1.8. Czujniki

System powinien obsługiwać każdy czujnik osobno oraz mieć możliwość rozpoznawania statusu podłączonych czujników, by w przypadku awarii jednego z nich, nie wpływało to na pomiary z pozostałych.

System powinien być wyposażony w czujnik temperatury powietrza, który spełnia co najmniej wymagania znajdujące się w tabeli 5.1.

Tabela 5.1. Minimalne wymagania stawiane czujnikom temperatury powietrza

Parametr	Wymaganie
Zakres pomiarowy	0°C ÷ 50°C
Dokładność pomiaru	±1.0°C w całym zakresie temperatur pracy
Temperatura pracy	0°C ÷ 50°C
Rozdzielcość	0.1°C

System powinien być wyposażony w czujniki wilgotności gleby, które spełniają co najmniej wymagania znajdujące się w tabeli 5.2.

Tabela 5.2. Minimalne wymagania stawiane czujnikom wilgotności gleby

Parametr	Wymaganie
Typ czujnika	analogowy, pojemnościowy
Napięcie robocze	DC 3.3 ÷ 5.0 V
Napięcie wyjściowe	DC 0 ÷ 3.0 V
Zastosowany materiał	Odporny na korozję

System powinien być wyposażony w czujnik ciśnienia atmosferycznego, który spełnia co najmniej wymagania znajdujące się w tabeli 5.3.

Tabela 5.3. Minimalne wymagania stawiane czujnikom ciśnienia atmosferycznego

Parametr	Wymaganie
Zakres pomiarowy	300 ÷ 1100 hPa
Dokładność pomiaru	±1 hPa w całym zakresie temperatur pracy
Temperatura pracy	0°C ÷ 50°C
Rozdzielcość	0.16 Pa
Napięcie zasilania	3.3 V lub 5.0 V DC

System powinien być wyposażony w 8-kanałowy przekaźnik z optyczną separacją, który spełnia co najmniej wymagania znajdujące się w tabeli 5.4.

Tabela 5.4. Minimalne wymagania stawiane przekaźnikom

Parametr	Wymaganie
Typ modułu	przekaźnikami z optyczną separacją
Napięcie zasilania przekaźnika, cewek	5 V DC
Aktywacja przekaźnika	stanem wysokim > 3,3 V DC
Maksymalne obciążenie	10 A przy 230 V AC oraz 10 A przy 30 V DC.

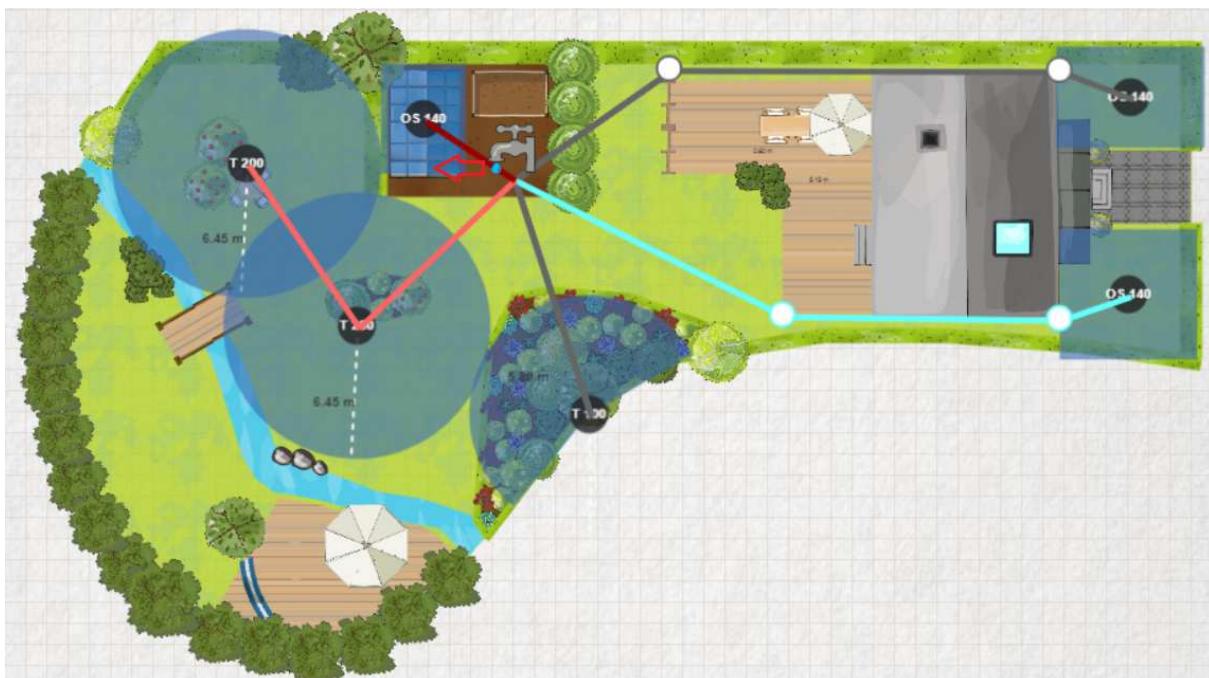
System powinien być wyposażony w więcej niż 2 zawory elektromagnetyczne, które spełniają co najmniej wymagania znajdujące się w poniższej tabeli.

Tabela 5.4. Minimalne wymagania stawiane zaworom elektromagnetycznym

Parametr	Wymaganie
Środowisko pracy	woda
Ciśnienie pracy	od 0,02 MPa do 0,8 MPa
Wytrzymałość na ciśnienie statyczne	do 2 MPa
Temperatura cieczy	od 0 °C do 60 °C
Napięcie zasilania	12 ÷ 24 V
Gwint połączeniowy	1/2 ÷ 3/4 "

5.2. Wizualizacja systemu w przestrzeni

Przyjęto, że system umiejscowiony będzie przy niewielkim domu jednorodzinnym z dużym ogrodem. Nawadnianie powinno objąć i monitorować obszary takie jak: szklarnia, skalniak ogrodowy, trawnik przy szklarni oraz frontowy dziedziniec. Schemat przedstawiono na rysunku 5.1. Serce systemu (urządzenie sterujące) oraz skrzynia z elektrozaworami zostaną umiejscowione w odrębnej części szklarni, co dodatkowo zabezpieczy urządzenia przed warunkami atmosferycznymi. Czujniki wilgotności zostaną umiejscowione na obszarach zaznaczonych kolorem niebieskim.

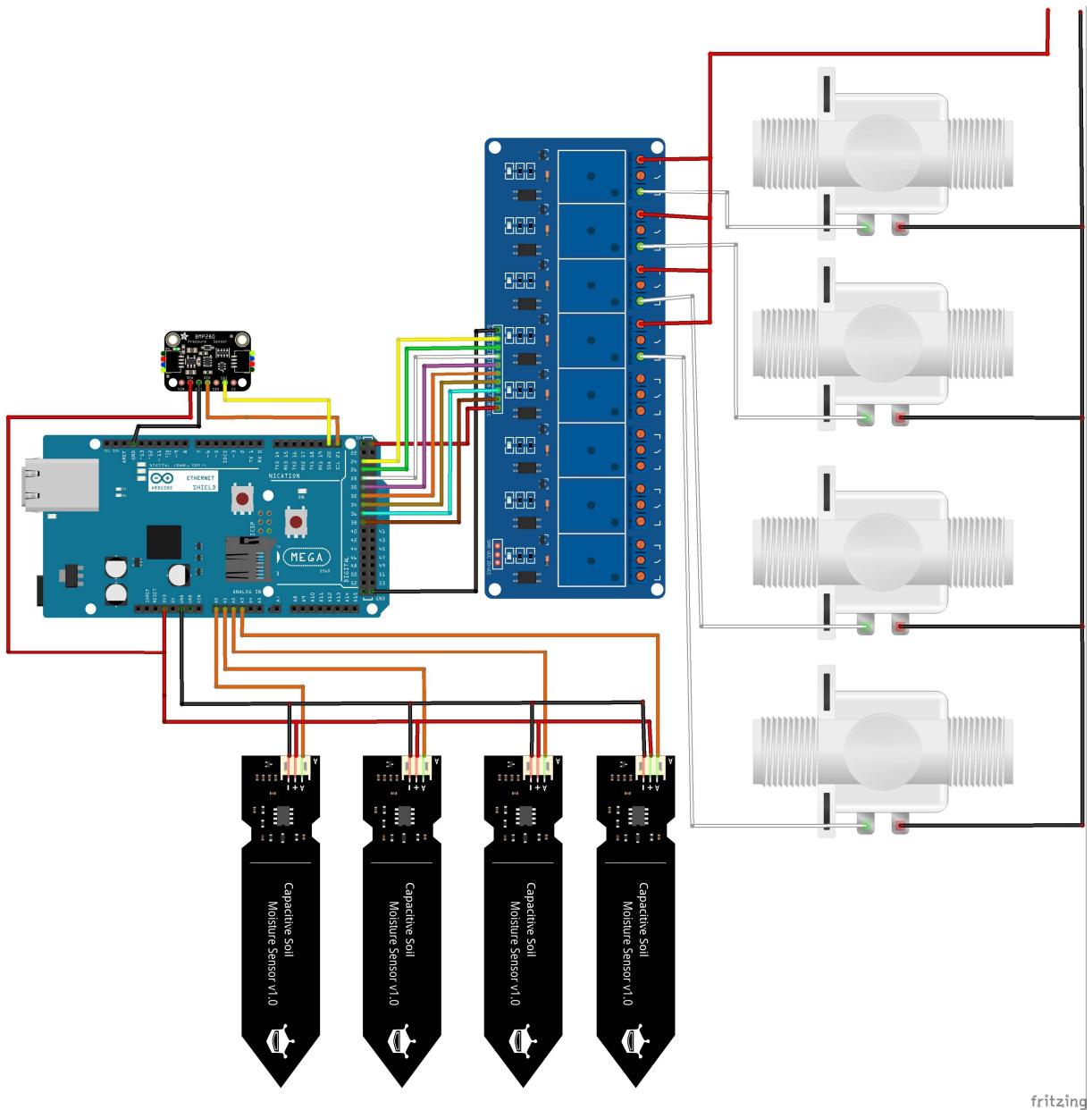


Rys. 5.1. Schemat usytuowania systemu

Wizualizacja została zrealizowana za pomocą darmowej aplikacji internetowej do planowania ogrodów „GARDENA myGarden”. Wbudowane narzędzia aplikacji pozwalają na dobranie obiektów zarówno do wizualizacji ogrodu, jak i odpowiednich elementów do systemu nawadniania. Po zakończeniu edycji istnieje możliwość pobrania listy potrzebnych elementów do montażu takiego systemu – oczywiście tylko z puli przedmiotów firmy Gardena.

5.3. Projekt prototypu

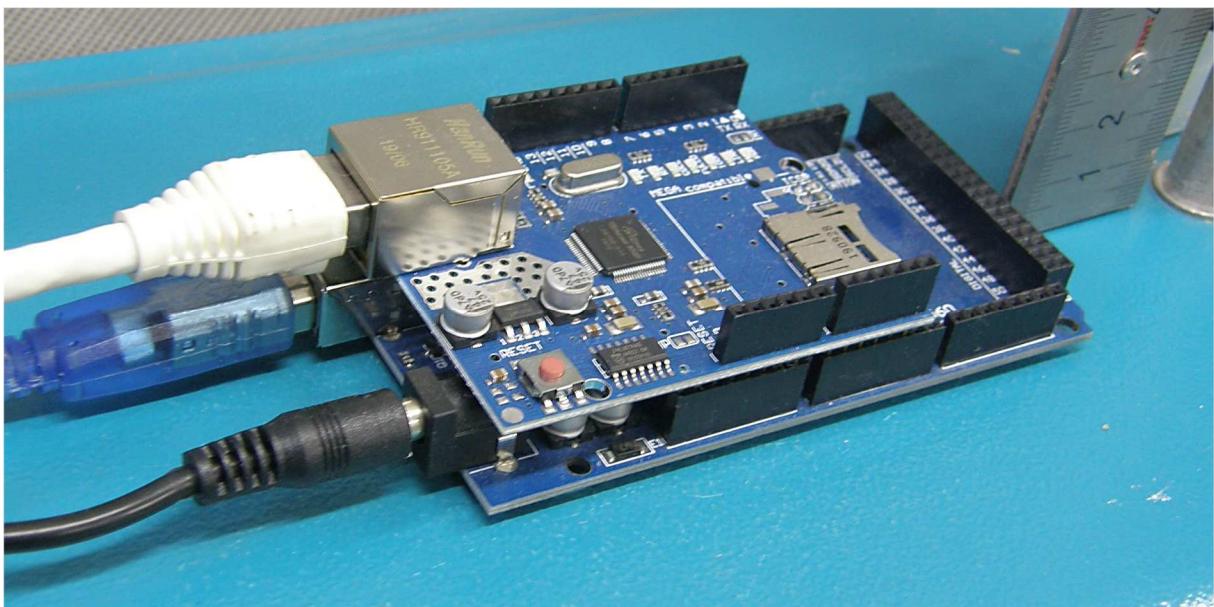
Zgodnie z przedstawionymi założeniami skonstruowano prototyp urządzenia. Dobrano minimalną ilość elementów potrzebnych do realizacji systemu. Zaletą takiego rozwiązania jest zminimalizowanie kosztów, jednak ze względu na brak redundancji czujników zmniejszono możliwość zabezpieczenia na wypadek uszkodzenia jednej z części systemu. Proponowany prototyp wygląda tak jak pokazano na rysunku 5.2.



Rys. 5.2. Wizualizacja prototypu przy użyciu programu fritzing

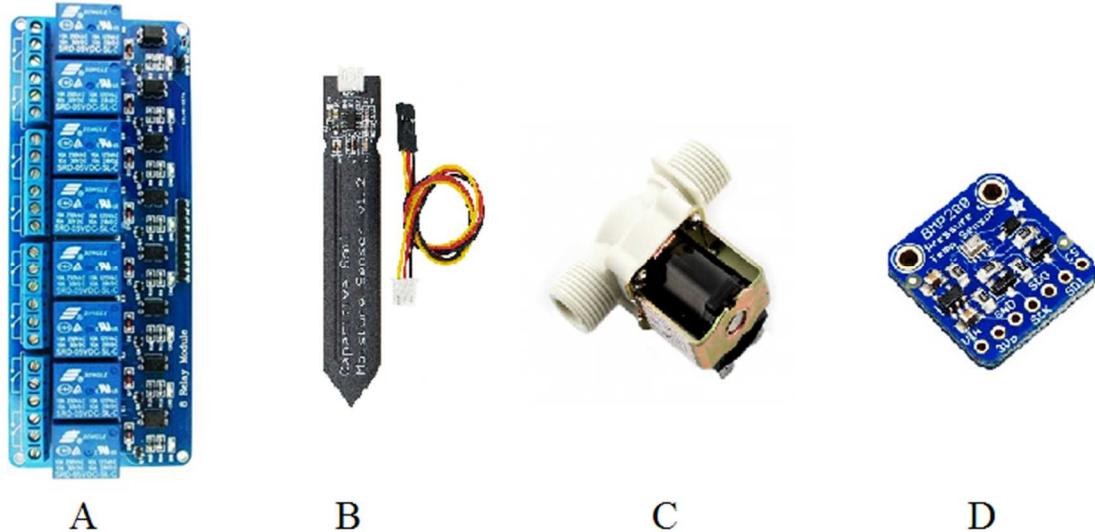
Najważniejszym elementem systemu jest moduł Arduino Mega 2560 Rev3 z mikrokontrolerem AVR ATmega2560. Element ten umożliwia kontrolę nad całym systemem pomiarowo-sterującym. Jest jedną z „najbogatszych” wersji Arduino, która

wypożyczona jest w 54 cyfrowe wejścia/wyjścia z czego 15 PWM (ang. Pulse-Width Modulation) oraz 16 analogowych. Układ taktowany jest sygnałem zegarowym o częstotliwości 16 MHz. Posiada 256 kB pamięci programu Flash oraz 8 kB pamięci operacyjnej SRAM. W przypadku tego prototypu płytki obsługuje wszystkie czujniki oraz umożliwia sterowanie systemem poprzez zadane parametry, które mogą być zmieniane dzięki stałemu połączeniu z bazą danych. Na rysunku poniżej pokazano schemat połączenia zasilania do płytki wraz z modelem rozszerzającym (ekspanderem wyprowadzeń) zwanym potocznie „shieldem”. Ten moduł służy do połączenia Arduino z Internetem. Nakładka posiada także czytnik kart pamięci microSD, który umożliwia przechowywanie większej ilości danych.



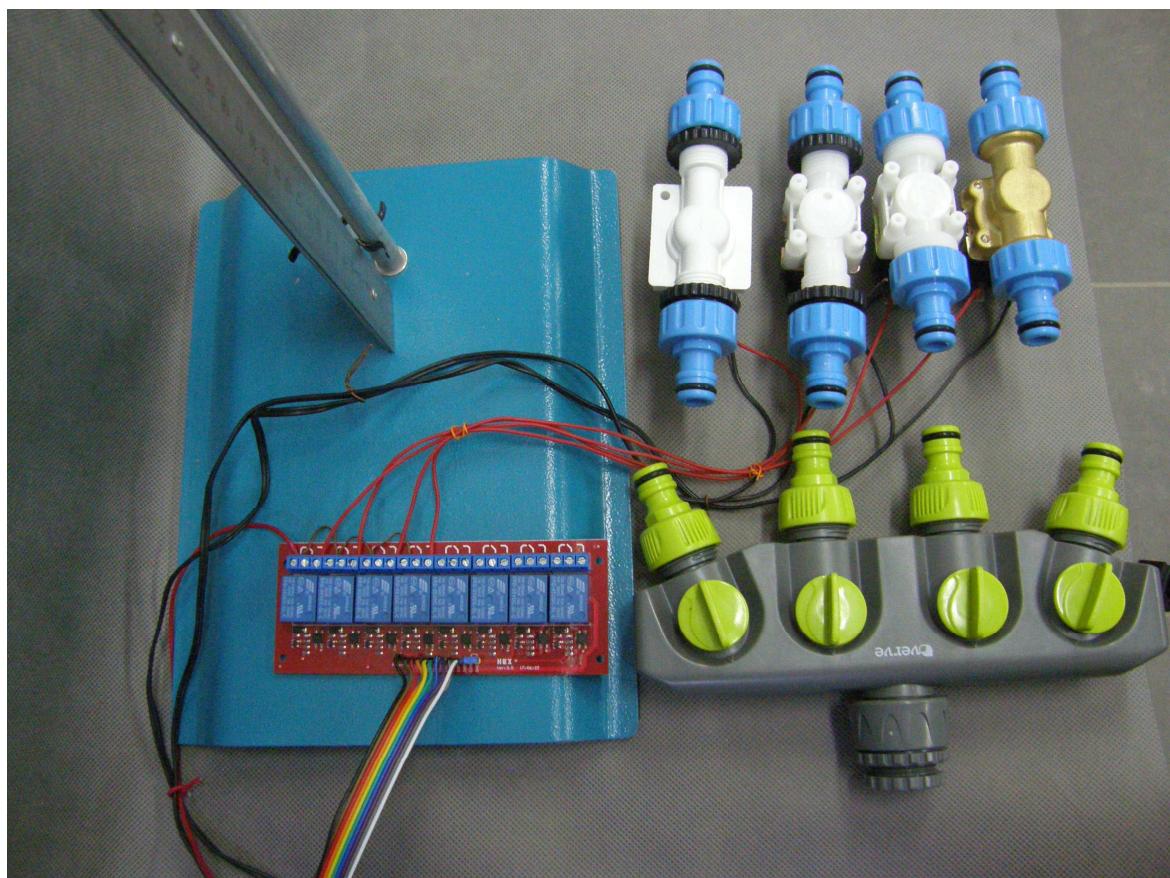
Rys. 5.3. Schemat podłączenia modułu do płytki prototypowej

Tak jak wspomniano wcześniej moduł Arduino obsługuje czujniki umożliwiające gromadzenie danych. Wszystkie wykorzystane czujniki przedstawiono na rysunku 5.4. Najmniejszym elementem prototypu jest cyfrowy czujnik ciśnienia atmosferycznego, który również posiada również wbudowany sensor temperatury. Umożliwia pomiar w zakresie od 300 do 1100 hPa z dokładnością do 1 hPa oraz pomiar temperatury z zakresu od -40 do 85°C. Na rynku istnieją różne wersje tego czujnika działające na logice 3,3 lub 5 V. Moduł komunikuje się przez magistralę I2C lub SPI. Następnym czujnikiem jest pojemościowy czujnik wilgotności gleby M335. Zasadę działania czujnika opisano w rozdziale 5.3.1. na etapie kalibracji, lecz należy dodać, że w porównaniu z czujnikami rezystancyjnymi, czujniki pojemościowe nie wymagają bezpośredniej ekspozycji metalowych elektrod (wykonane są z materiałów odpornych na korozję), co znacznie wydłuża żywotność. Kolejne elementy systemu służą do sterowania nawadnianiem. Kontrolę nad przepływem cieczy sprawuje moduł 8 przekaźników oraz zawory elektromagnetyczne. Zawory działają pod ciśnieniem 0,8 MPa, co pozwoli na połączenie systemu do instalacji wodociągowej budynku. Są zamknięte do momentu podłączenia zasilania. Ze względu na to, że działają przy napięciu wyższym niż napięcie zasilania płytki z mikrokontrolerem należy zastosować moduł z optyczną separacją.



Rys. 5.4. Zestawienie elementów składowych; A- Moduł 8 przekaźników, B- Czujnik wilgotności gleby M335, C- Zawór elektromagnetyczny, D- czujnik ciśnienia i temperatury BMP280

Dzięki odseparowaniu układu sterującego od przekaźników, nie występuje ryzyko uszkodzenia mikrokontrolera na wskutek przebicia. Zastosowany przekaźnik aktywowany jest stanem wysokim (z zakresu 3,3÷12 V), co minimalizuje ryzyko otwarcia zaworu w przypadku uszkodzenia mikrokontrolera. Na rysunku 5.5. pokazano w jaki sposób elektrozawory podłączone są do przekaźnika i w jaki sposób pozwalają rozdzielić oraz kontrolować przepływ wody.



Rys. 5.5. Sposób podłączenia modułu przekaźnika z elektrozaworami magnetycznymi

5.3.1. Kalibracja czujników wilgotności

Określenie wilgotności podłoża za pomocą wykorzystanych czujników odbywa się na podstawie pomiaru przenikalności elektrycznej gleby, która uzależniona jest głównie od zawartości wody w ośrodku. Przy dokonywaniu pomiarów, w celu wyeliminowania zakłóceń w odczycie, należy uważać, aby sonda ściśle przylegała do próbki pomiarowej. Ze względu na zasadę działania czujnika wpływ na odczyty może mieć zasolenie gleby, dlatego aby uzyskać jak najbardziej precyzyjne pomiary należy wykonać skalowanie (kalibrację) czujników dla gleby występującej na danym obszarze.

W celu dokonania kalibracji układu wyznaczono empirycznie wartości wyjściowe dla dwóch skrajnych przypadków. Sonda mierzy przenikalność elektryczną gleby, co oznacza, że gdy wilgotność gleby wzrasta, wartość wyjściowa maleje i odwrotnie - gdy wilgotność maleje, wartość wyjściowa wzrasta. Proces kalibracji zobrazowano na rysunku 5.5.

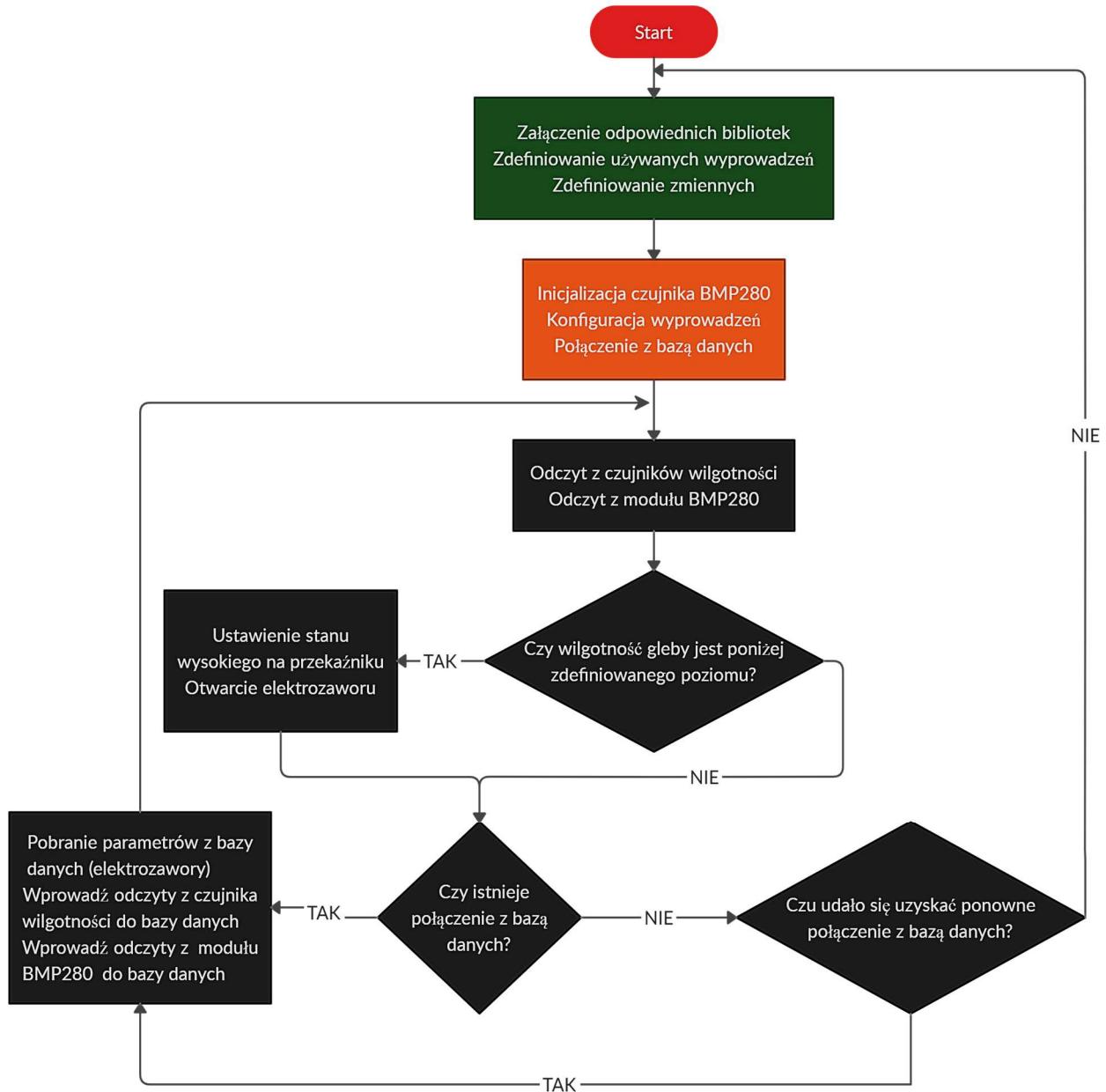


Rys. 5.6. Proces kalibracji czujników wilgotności

Aby wyznaczyć dolną granicę pobrano próbki gleby, które poddano etapowi suszenia w temperaturze 105°C aż do chwili ustabilizowania się ich masy zgodnie z normą PN-EN ISO 17892-1. Następnie po ostudzeniu próbek do temperatury pokojowej zanurzono sondę w suchym podłożu i odczytano wartość na wyjściu czujników. Z kolei w celu wyznaczenia górnej wartości te same próbki maksymalnie nasączone wodą destylowaną. Procentowa zawartość wilgotności gleby obliczana jest na poziomie oprogramowania urządzenia pomiarowego, co opisano na etapie implementacji.

6. Implementacja modułu sterującego

Oprogramowanie wgrywane do pamięci mikrokontrolera zostało napisane w środowisku programistycznym Arduino 1.8.13. Jak sama nazwa wskazuje jest to środowisko dedykowane płytkom z rodziną Arduino. Na rysunku 6.1. zobrazowano sposób działania programu. Ze względu na obszerny kod źródłowy przedstawiono jedynie najważniejsze fragmenty programu. Nie bez przyczyny poszczególne etapy zaznaczono różnymi kolorami. Kolor zielony reprezentuje dołączone biblioteki oraz zdefiniowanie stałych, zmiennych i funkcji. Dla każdego z podzespołów wykorzystano gotowe biblioteki dostępne w menadżerze środowiska.



Rys. 6.1. Schemat działania programu zaimplementowanego do mikrokontrolera

Następnym etapem jest wykonanie funkcji inicjalizującej parametry początkowe – void setup(). Inicjalizacja (kolor pomarańczowy) wykonywana jest jeden raz i ma miejsce po

każdym uruchomieniu urządzenia. Ma na celu aktywowanie niezbędnych do działania programu funkcji początkowych oraz stanów na wyjściach czujników. W funkcji setup(), na rysunku 6.2, zdefiniowano wyjścia przekaźników i ustawiono początkowy stan niski, co zabezpiecza system przed niekontrolowanym uruchomieniem zaworu otwierającego przepływ wody. Kolejno zainicjalizowano połączenie się modułu sieciowego z Internetem i z bazą danych oraz rozpoczęto komunikację z sensorem BMP280. W komentarzach zaznaczono biblioteki jakie należy dołączyć, aby można było wykonać polecenie. Aby komunikować się z monitorem szeregowym i wyświetlać rezultaty użyto funkcji Serial.begin(). W nawiasie określono szybkość transmisji danych w bitach na sekundę (zwaną baud-rate).

```

void setup() {
    pinMode(relay1, OUTPUT);
    digitalWrite(relay1, LOW);
    pinMode(relay2, OUTPUT);
    digitalWrite(relay2, LOW);
    pinMode(relay3, OUTPUT);
    digitalWrite(relay3, LOW);
    pinMode(relay4, OUTPUT);
    digitalWrite(relay4, LOW);
    Serial.begin(115200);           //while (!Serial);
    Ethernet.begin(mac_addr);      //##include <Ethernet.h>
    dns_client.begin(Ethernet.dnsServerIP()); //##include <Dns.h>
    dns_client.getHostName(hostname, server_ip);
    Serial.println(server_ip);
    Serial.println("Łączenie...");
    if (conn.connect(server_ip, 3306, user, password)) { //##include <MySQL_Connection.h>
        Serial.println("Połączono");
        delay(1000);
    }
    else
        Serial.println("Brak połączenia");
    if (bmp280.initialize()) Serial.println("BMP 280- znaleziono czujnik");
    else Serial.println("BMP 280- NIE znaleziono czujnika"); //##include "i2c.h"
    bmp280.setEnabled(0); //##include "i2c_BMP280.h"
    bmp280.triggerMeasurement(); //##include <Wire.h>
}

```

Rys. 6.2. Funkcja inicjalizująca - void setup()

Kolorem czarnym na rysunku 6.1. zaznaczono główną część programu, która przy prawidłowym połączeniu i braku awarii będzie zapętlona w nieskończoność - void loop(). Działanie tego kodu polega na wykonywaniu wszystkich instrukcji z pętli, linijka po linijce. W danym momencie mikroprocesor może zająć się jedynie jedną operacją. Taki sposób wykonywania działań może wydawać się dość ograniczający, lecz należy pamiętać, że mikroprocesor pracuje z częstotliwością 16 MHz, co w dużym uproszczeniu oznacza, że jest w stanie wykonać 16 milionów operacji na sekundę. Jest to na tyle dużo, że ludzkie oko nie jest w stanie zarejestrować niektórych zmian, chyba że specjalnie wprowadzi się np. opóźniające polecenie delay(). W tej części programu wykonują się trzy główne zadania – odczytanie wartości z czujników, sterowanie nawadnianiem oraz komunikowanie się z bazą danych.

6.1. Komunikacja z czujnikiem BMP280

Pomiar temperatury oraz ciśnienia odbywa się za pomocą czujnika BMP280, który podłączono do interfejsu komunikacyjnego I²C. Czujnik może zasadniczo działać w dwóch trybach: normalnym oraz „forced mode”. Wykorzystano ten drugi, ponieważ w tym trybie wykonywany jest pojedynczy pomiar, a następnie czujnik powraca do trybu uśpienia. Jest to powszechnie stosowana metoda w aplikacjach, w których wymagana jest niska częstotliwość próbkowania. Poniżej na rysunku 6.3. przedstawiono fragment odpowiedzialny za pobranie wartości z czujnika.

```
void bmp280_read() {
    bmp280.awaitMeasurement();
    bmp280.getTemperature(temperature);
    bmp280.getPressure(pascal);
    hPa = pascal / 100;
    bmp280.triggerMeasurement();
    Serial.print(" Ciśnienie atmosferyczne: ");
    Serial.print(hPa);
    Serial.print(" hPa; Temperatura: ");
    Serial.print(temperature);
    Serial.println(" C");
}
```

Rys. 6.3. Fragment odpowiedzialny za obsługę BMP280

Aby polecenia poprawnie się wykonały należy dołączyć dedykowane pliki nagłówkowe: „i2c_BMP280.h” oraz bibliotekę do obsługi magistrali I2C <Wire.h>

6.2. Komunikacja z czujnikami wilgotności

Do odczytania wartości z czujników wilgotności wykorzystano przetwornik ADC (ang. Analog-Digital Converter), którego zadaniem jest próbkowanie napięcia podanego na wyjściu układu i przetworzenie go na postać cyfrową. W Arduino Mega zastosowano przetwornik

10-bitowy, co oznacza, że zależnie od napięcia wyjściowego z czujnika jego wartość może mieścić się w zakresie od 0 do 1023 (napięcie wejściowe waha się w zakresie 0-5 V). Na poniższym rysunku pokazano w jaki sposób odczytane wartości przekształcono na procentową zawartość wilgotności w glebie.

```

void moisture_sensor_read() {
    soilSensorRead1 = analogRead(A0);
    Serial.println(soilSensorRead1);
    soilMoisturePercentage1 = map(soilSensorRead1, DryValue1, WaterValue1, 0, 100);
    if (soilMoisturePercentage1 > 100)
    {
        Serial.println("error > 100 %");
    }
    else if (soilMoisturePercentage1 < 0)
    {
        Serial.println("error < 0 %");
    }
    else if (soilMoisturePercentage1 > 0 && soilMoisturePercentage1 < 100)
    {
        Serial.print("Moisture sensor nr 1: ");
        Serial.print(soilMoisturePercentage1);
        Serial.println("%");
    }
}

```

Rys. 6.4. Fragment odpowiedzialny za pomiar z czujnika wilgotności

Na początku wykorzystano funkcję analogRead(), która odpowiedzialna jest za pobranie wartości cyfrowej z przetwornika ADC. Aby można było przekształcić otrzymaną wartość na wilgotność gleby należy dokonać kalibracji czujnika, tak jak opisano to w rozdziale 5.3.1 a następnie użyć funkcji map() , która przeskalałatwuje podaną wartość między odczytem dla próbki suchej a próbki w pełni nasyconej do wartości z zakresu od 0 do 100. Przykładowo, dla pierwszego czujnika wartość odczytu „A0” dla próbki suchej wynosiła 661, natomiast dla próbki w pełni nasączanej wodą 346. Dalsze instrukcje programu służą jedynie do sprawdzenia (wyświetlenia), czy odczyty mieszczą się w pożądanym zakresie i jaka jest zawartość procentowa wilgotności gleby.

6.3. Komunikacja z bazą danych

Aby rozpocząć proces komunikacji z bazą danych należy spełnić kilka warunków. W pierwszej kolejności potrzebne jest połączenie z Internetem oraz zbiór odpowiednich plików nagłówkowych. W tym celu do obsługi modułu sieciowego wykorzystano bibliotekę Ethernet.h, natomiast do komunikacji z bazą danych: MySQL_Connection.h, Dns.h oraz MySQL_Cursor.h. Oczywiście, do utworzenia bazy danych jak i połączenia się z nią konieczna jest obecność serwera SQL. W tym przypadku użyto hostingu bazy danych – MySQL ze strony <https://hosting.domena.pl>. Struktura bazy danych jest bardzo prosta. Składa się ona z trzech tabel, z których dwie służą do gromadzenia danych, natomiast trzecia przechowuje jedynie parametry konfiguracyjne do sterowania nawadnianiem. Sam proces komunikacji opiera się na dwóch poleceniach: SELECT (wyświetlanie rekordów) oraz INSERT (wstawianie rekordów). Poniżej przedstawiono fragmenty kodu dotyczący każdego z nich. Zanim jednak wykona się poszczególne polecenia należy zdefiniować zmienne przechowujące zapytania.

```

char select_setting1[] = "SELECT moisture1 FROM inzynierka.Settings";
char select_setting2[] = "SELECT moisture2 FROM inzynierka.Settings";
char select_setting3[] = "SELECT moisture3 FROM inzynierka.Settings";
char select_setting4[] = "SELECT moisture4 FROM inzynierka.Settings";

char bmpInsertData[] = "INSERT INTO inzynierka.BMP280 (temperature, pressure) VALUES (%s,%s)";
char bmpQuery[128];
char bmpTemperature[10];
char bmpHPa[10];

```

Rys. 6.5. Przykład zdefiniowania zmiennych do zapytań bazy danych

Zapytanie SELECT wykonano za pomocą dynamicznego kursora, który po wykonaniu zapytania w pierwszej kolejności pobiera kolumny, a następnie stopniowo pobiera po jednym wierszu używając funkcji get_next_row, aż zwróci wartość „NULL”. Na koniec, aby zwolnić przydzieloną pamięć należy skorzystać z funkcji delete().

```

void settings() {

    MySQL_Cursor *cur_mem1 = new MySQL_Cursor(&conn);
    cur_mem1->execute(select_setting1);
    column_names *columnNames1 = cur_mem1->get_columns();
    do {
        row = cur_mem1->get_next_row();
        if (row != NULL) {
            setting1 = atol(row->values[0]);
        }
    } while (row != NULL);
    delete cur_mem1;
}

```

Rys. 6.6. Wykonanie zapytania SELECT

Analogicznie wykonywane jest polecenie INSERT, z tą różnicą, że przed wysłaniem zapytania należy odpowiednio przygotować polecenie konwertując wszystkie zmienne na jeden typ znakowy.

```

void bmp280_insert() {

    MySQL_Cursor *bmp_cur_mem = new MySQL_Cursor(&conn);
    dtostrf(temperature, 5, 2, bmpTemperature);
    dtostrf(hPa, 7, 2, bmpHPa);
    sprintf(bmpQuery, bmpInsertData, bmpTemperature, bmpHPa);
    bmp_cur_mem->execute(bmpQuery);
    delete bmp_cur_mem;
    Serial.println("BMP Data recorded.");
}

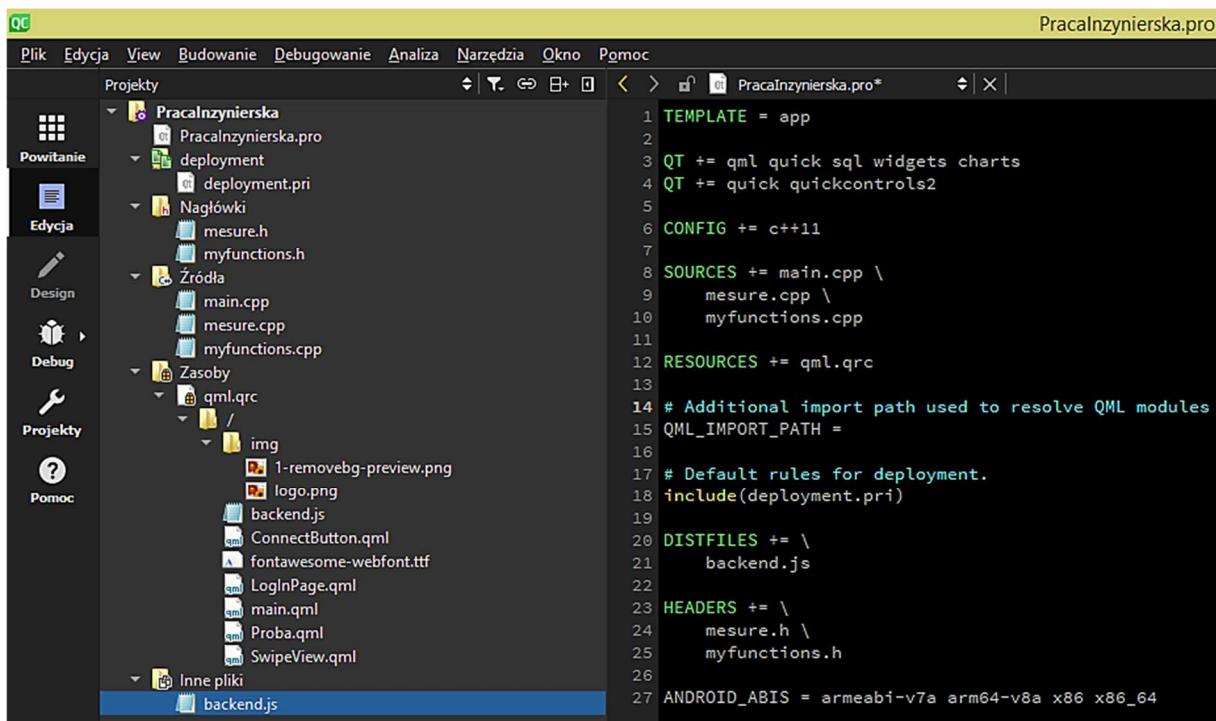
```

Rys. 6.7. Wykonanie polecenia INSERT

W tym celu wykorzystano funkcję dtostrf(), która konwertuje liczbę zmiennoprzecinkową na tablicę znaków. Przykładowo dtostrf(temperature, 5, 2, bmpTemperature) zamienia zmienną typu float na tablice znaków bmpTemperature przy maksymalnej długości 5 znaków, zachowując 2 miejsca po przecinku. Następnie funkcja sprintf() łączy ciąg znaków w jedno zapytanie o nazwie bmpQuery.

7. Implementacja aplikacji mobilnej

Aplikację mobilną opracowano za pomocą najnowszych dostępnych technologii od Qt, a mianowicie Qt Quick oraz języka QML. Dla uściślenia danych skorzystano z wersji Qt 5.15.0 i Qt Creator 4.13.0. Projekt składa się z pliku projektu (.pro) oraz 3 głównych katalogów - katalogu z plikami nagłówkowymi, katalog z plikami źródłowymi oraz katalog przechowujący zasoby. Jak można zauważyć na poniższym rysunku część plików dotycząca interfejsu (katalog zasoby) stanowi odrębną część, dzięki czemu istnieje możliwość rozdzielenia pracy między projektantów zajmujących się częścią graficzną, a zespołem deweloperów, którzy tworzą logikę aplikacji.



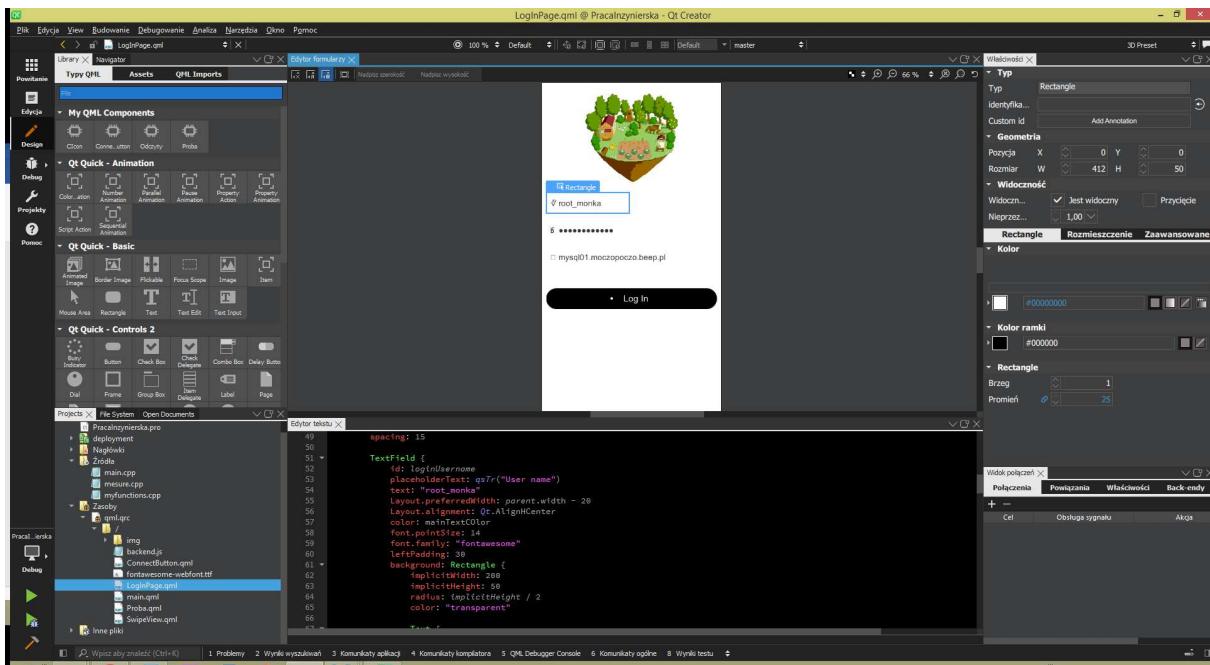
The screenshot shows the Qt Creator interface with the following details:

- Project Tree (Left):** Shows the project structure:
 - PracaInzynierska (selected)
 - deployment (with deployment.pri)
 - Nagłówki (with mesure.h, myfunctions.h)
 - Źródła (with main.cpp, mesure.cpp, myfunctions.cpp)
 - Zasoby
 - qml.qrc
 - / (empty folder)
 - img
 - 1-removebg-preview.png
 - logo.png
 - backend.js
 - ConnectButton.qml
 - fontawesome-webfont.ttf
 - LoginPage.qml
 - main.qml
 - Proba.qml
 - SwipeView.qml
 - Inne pliki (with backend.js)
- Code Editor (Right):** Displays the content of the PracaInzynierska.pro file:

```
1 TEMPLATE = app
2
3 QT += qml quick sql widgets charts
4 QT += quick quickcontrols2
5
6 CONFIG += c++11
7
8 SOURCES += main.cpp \
9     mesure.cpp \
10    myfunctions.cpp
11
12 RESOURCES += qml.qrc
13
14 # Additional import path used to resolve QML modules
15 QML_IMPORT_PATH =
16
17 # Default rules for deployment.
18 include(deployment.pri)
19
20 DISTFILES += \
21     backend.js
22
23 HEADERS += \
24     mesure.h \
25    myfunctions.h
26
27 ANDROID_ABIS = armeabi-v7a arm64-v8a x86 x86_64
```

Rys. 7.1. Struktura projektu

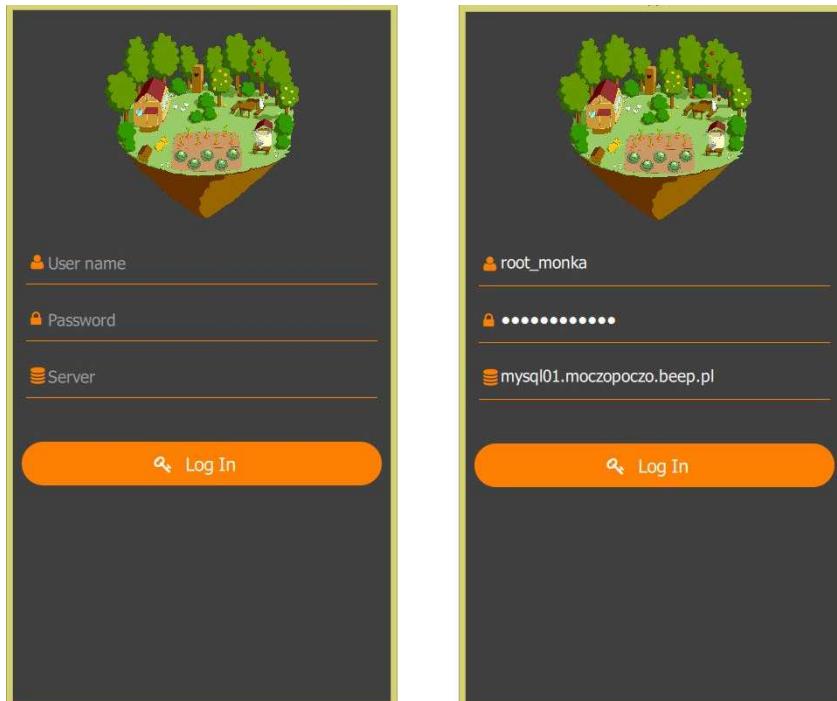
Interfejs graficzny aplikacji wykonano w trybie „Design” przy użyciu edytora tekstu oraz gotowych elementów i komponentów wbudowanych w Qt Creator. Użycie tego trybu znacznie przyspiesza pracę, a przede wszystkim pozwala na bieżąco monitorować wprowadzone zmiany. Widok płotna oraz edytor tekstu można edytować równocześnie, co zobrazowano na rysunku 7.2. Dobrą praktyką z perspektywy czasu okazało się tworzenia aplikacji rozpoczynając od szkieletu interfejsu aplikacji, gdyż w przeciwnym wypadku pomija się wiele istotnych elementów logiki oprogramowania.



Rys. 7.2. Budowa szaty graficznej aplikacji

7.1. Interfejs użytkownika

Widok ekranu logowania jest pierwszym widokiem, który pojawia się po uruchomieniu aplikacji. Aby móc zalogować się użytkownik musi posiadać login identyfikujący, hasło oraz adres serwera bazy danych. Aplikacja w tym przypadku nie ma możliwości zakładania konta lub przypominania hasła. Na poniższym rysunku pokazano widok ekranu logowania.



Rys. 7.3. Widok okna startowego – logowanie

W chwili wciśnięcia przycisku „Log in” wywoływana jest funkcja sprawdzająca dane na podstawie wprowadzonych wartości. Jeżeli dane są poprawne aplikacja pozwoli na wejście we właściwą część dotyczącą systemu nawadniania i warunków panujących w ogrodzie.

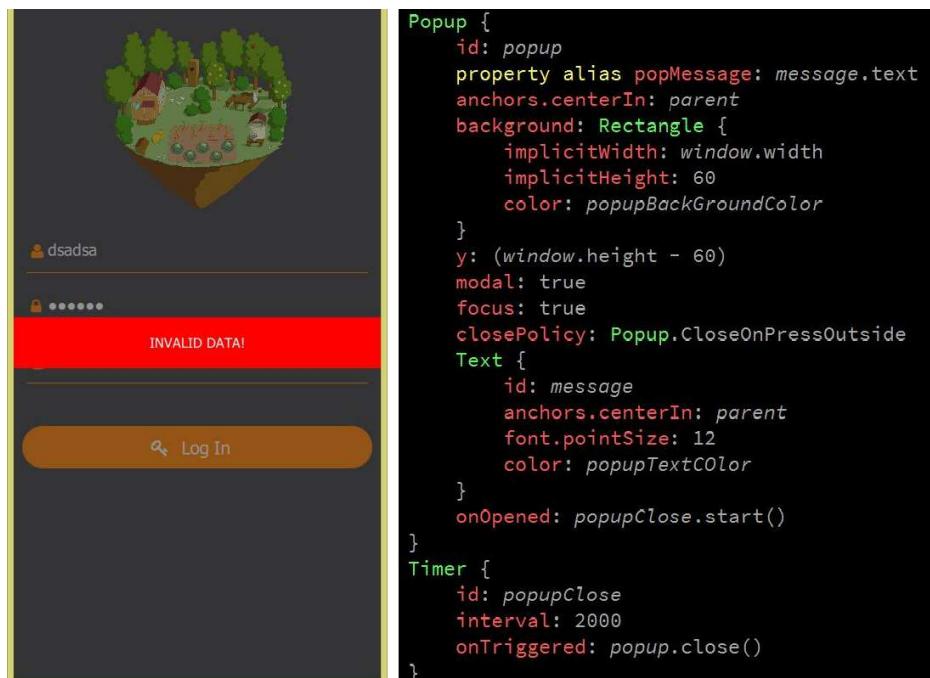
```
function loginUser(serv, uname, pword)
{
    var ret = Backend.validateUserCredentials(serv,uname, pword)
    var message = ""
    if(ret)
    {
        message = "Missing credentials!"
        popup.popMessage = message
        popup.open()
        return
    }

    Myfunction.connectDB(serv, uname, pword)

    var checkconnection= Myfunction.errorConnection
    if (checkconnection==false){
        console.log("Login Success!")
        showSwipe() //onClicked: stackView.replace("qrc:/SwipeView.qml")
    }
    else {
        message = "INVALID DATA!"
        popup.popMessage = message
        popup.open()
        return}
}
```

Rys. 7.4. Fragment programu dotyczący funkcji logowania

W przeciwnym wypadku istnieją dwie inne możliwości, w których użytkownik albo nie wprowadził żadnych wartości, albo pomylił się przy ich wprowadzaniu. W celu komunikacji z użytkownikiem skorzystano z typu Popup. W kontekście C++ Qt dla „klas” QML stosuje nazewnictwo „Type”, np. Popup QML Type. Jest to typ, który powoduje pojawienie np. komunikatu, który można zobaczyć na rysunku 7.5.



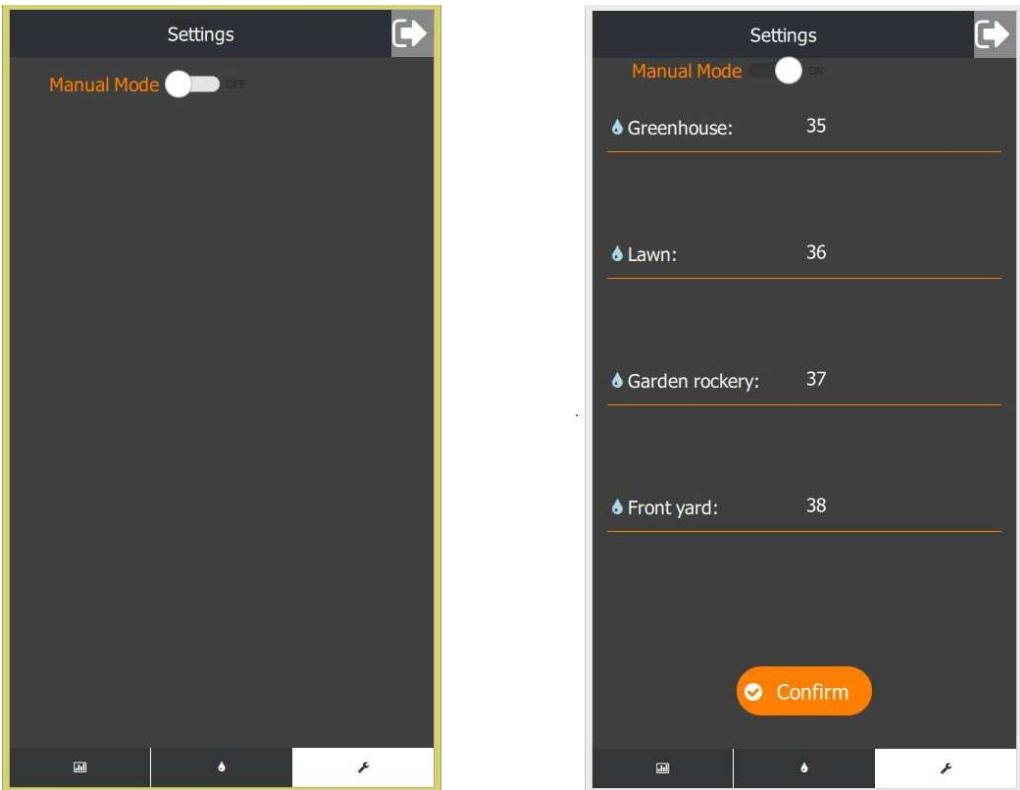
Rys. 7.5. Przykład użycia typu Popup

Zalogowany użytkownik w dalszej części aplikacji ma do dyspozycji trzy podstrony z modułu nawigacji. Pierwsza z nich dotyczy warunków atmosferycznych, które wyświetlane są w formie wykresu oraz w postaci aktualnie występujących wartości pokazanych w dolnej części aplikacji. Sposób pobierania i wyświetlania wartości z bazy danych opisany jest w rozdziale 5.5.2. Na wykresach wyświetlane są średnie wartości występujące w ciągu ostatnich pięciu dni. Oczywiście wykresy mogą być definiowane różnie, zależnie od indywidualnych potrzeb użytkownika. Druga podstrona dotyczy procentowej zawartości wilgotności gleby. W tym przypadku wyświetlane są jedynie aktualne panujące warunki w ogrodzie. W celach polepszających wizualną część podstrony wstawiono mapę ogrodu ukazującą kluczowe jej części.



Rys. 7.6. Widok dwóch pierwszych podstron

Ostatnia podstrona aplikacji służy do zmiany parametrów sterujących. Aby aktywować panel należy przesunąć przełącznik do pozycji „ON”. W tym momencie z bazy danych pobierane są ostatnie wprowadzone zmiany. Podane wartości (procentowa zawartość wilgotności w glebie) ustawiają moment włączenia się zaworów elektromagnetycznych. Po wprowadzeniu nowych wartości, aby zatwierdzić proces przesyłania danych należy nacisnąć przycisk „Confirm”. Widok panelu sterującego przedstawiono na rysunku 7.8.



Rys. 7.8. Panel sterujący

Jak widać na powyższych rysunkach na każdej z podstron w prawym górnym rogu znajduje się ikonka służąca do wylogowania użytkownika. Przycisk ten kończy komunikację z bazą danych oraz przenosi użytkownika do pierwszej strony logowania. Zastosowane ikony i symbole to tak naprawdę szczególny rodzaj czcionki – Font Awesome.

7.2. Komunikacja z bazą danych

Do utworzenia połączenia z bazą danych SQL w Qt służy klasa QSqlDatabase, która napisana jest w języku C++. Niestety połączenie z bazą danych uruchomioną na lokalnym lub globalnym serwerze wymagany jest odpowiedni sterownik. W przypadku MySQL, jak i Microsoft SQL Server sterownik ten nie jest domyślnie dostępny. Oznacza to, że aby możliwe było realizowanie połączenia z bazą należy pobrać ze strony producenta odpowiednią paczkę ze sterownikami (zwracając uwagę na wersję Qt Creator). Czasem jednak i to nie wystarcza. W przypadku autora tekstu ostatecznym rozwiązaniem było pobranie dodatkowych plików od innych użytkowników z serwisu GitHub. W celu przesyłania informacji między C++, a QML stworzono dwie klasy dziedziczące po klasie QObject o nazwie „Myfunction” oraz „Messure”. Zawartość klasy znajduje się w pliku nagłówkowym .h oraz w pliku źródłowym .cpp. Proces komunikacji zostanie przedstawiony na podstawie samego połączenia się z bazą danych. Analogicznie wykonuje się to tak samo dla innych poleceń.

Aby połączyć się z bazą danych w pliku nagłówkowym załączono odpowiednie definicje klas modułów, zdefiniowano zmienne i funkcje oraz makro Q_PROPERTY (służy do przekazywania właściwości), Dzięki Q_INVOKABLE istnieje możliwość wykonywania

metod z języka C++ w języku QML. Na rysunku 7.9 pokazano przykładowe użycie metody na etapie logowania.

```
#ifndef MYFUNCTIONS_H
#define MYFUNCTIONS_H
#include <QObject>
#include <QString>
#include <QtSql/QtSql>
#include <QSqlDatabase>
#define Nameof_DB QString("inzynierka")

class Myfunctions : public QObject
{
    Q_OBJECT
    Q_PROPERTY(bool errorConnection MEMBER errorConnection NOTIFY errorChanged)

public:
    explicit Myfunctions(QObject *parent = nullptr);
    Q_INVOKABLE bool connectDB(QString server, QString userName, QString password);

    Q_INVOKABLE bool errorConnection;

private:
    QSqlDatabase db;

signals:
    void errorChanged();
};

#endif // MYFUNCTIONS_H
```

Rys. 7.9. Fragment pliku nagłówkowego „myfunction.h” załączonego w celu połączenia się z bazą danych

Następnie w pliku źródłowym „myfunction.cpp” umieszczono metodę odpowiedzialną za połączenie się z bazą danych i emisję sygnału w przypadku zmiany parametru (errorChanged()).

```
bool Myfunctions::connectDB(QString server, QString userName, QString password)
{
    qDebug() << Q_FUNC_INFO << QSqlDatabase::drivers();
    db = QSqlDatabase::addDatabase("QMYSQL");

    qDebug() << Q_FUNC_INFO << server << userName << password;
    db.setHostName(server);
    db.setUserName(userName);
    db.setPassword(password);
    db.setDatabaseName(Nameof_DB);
    db.setPort(3306);
    if(db.open())
    {
        errorConnection = false;
        qDebug() << QString::fromUtf8("Success !");
    }
    else
    {
        errorConnection = true;
        qDebug() << QString::fromUtf8("Connection error !");
    }

    emit errorChanged();

    return errorConnection;
}
```

Rys. 7.10. Fragment pliku źródłowego „myfunction.cpp” załączonego w celu połączenia się z bazą danych

Kolejnym krokiem jest załączenie obiektu tej klasy do QML. W tym celu dokonano kilku wpisów do pliku main.cpp przy wykorzystaniu silnika QQmlApplicationEngine. Oprócz załączenia plików nagłówkowych zarejestrowano kontekst (wystawiono dane) obiektu Myfunction w głównym kontekście silnika – pozwala to na użycie QML obiektu w taki sposób jak pokazano na rysunku 7.11.

```
//#include <QGuiApplication>
#include<QApplication>
#include <QQmlApplicationEngine>
#include <QQuickView>
#include <QIcon>
#include <QQmlContext>

#include <myfunctions.h>
int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QApplication::setApplicationName("MyItrygationApplication");
    QApplication app(argc, argv);
    QIcon::setThemeName("MyApplicationDBCharts");
    QQmlApplicationEngine engine;
    engine.rootContext()->setContextProperty("Myfunction", new Myfunctions());
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

    return app.exec();
}
```

Rys. 7.11. Zawartość pliku „main.cpp” po rejestracji obiektu „Myfuntion”

Najważniejszą częścią tego programu jest silnik QQmlApplicationEngine, który ładuje interfejs aplikacji, a także pozwoli wystawić do QML obiekty klas C++. Przydatną funkcją jest tryb skalowania na ekranach o dużym DPI (ang. dots per inch) co zastosowano dzięki `QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling)`.

8. Testowanie

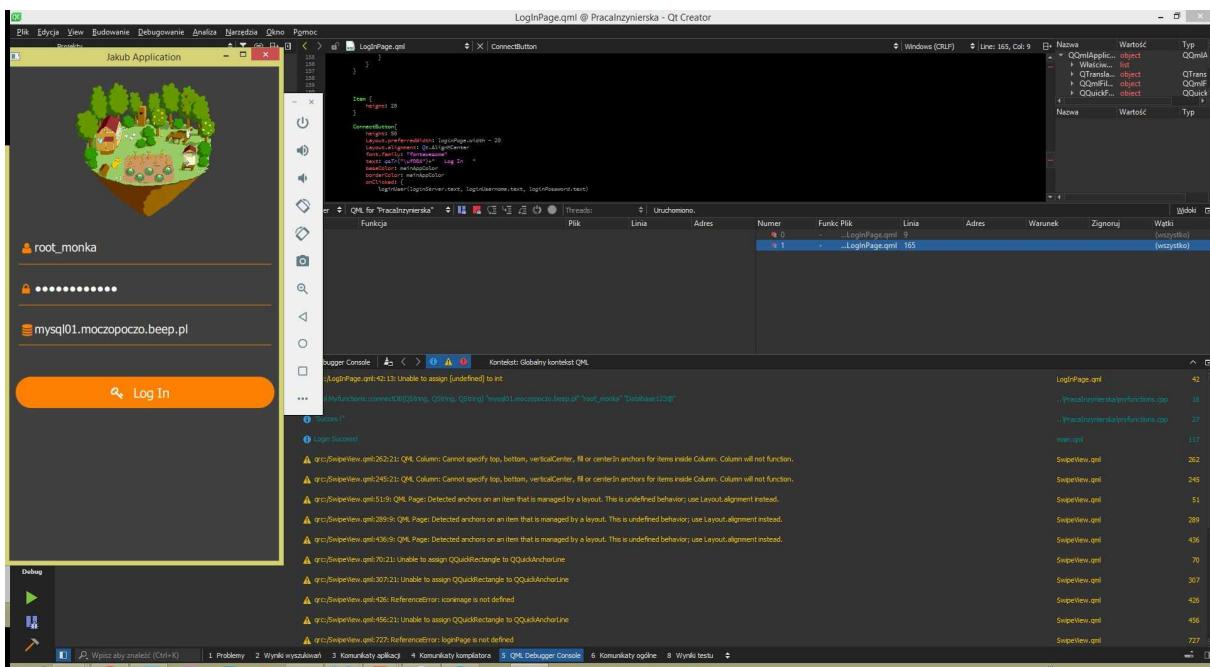
Aby sprawdzić poprawność działania systemu przeprowadzono szereg testów. W pierwszej kolejności wykonano testy jednostkowe weryfikując poprawność działania pojedynczych elementów programu. Po stronie wykonanego prototypu polegało to na komunikowaniu się z poszczególnym elementem systemu za pomocą interfejsu szeregowego UART wykorzystując komputer. Aby transmisja przebiegła prawidłowo należało ustawić taką samą prędkość przesyłania danych na obu układach. Diagnostyka i sygnalizacja różnych zdarzeń odbyła się za pomocą funkcji przesyłania ciągu znaków - Serial.println(). Rezultat użycia tej funkcji pokazano na rysunku 8.3. Oprócz sprawdzenia poprawności wykonania się polecen z poziomu portu szeregowego, sprawdzono również wartości przesyłane z każdego czujnika w bazie danych.

	ID	temperature	pressure	bmpTime
<input type="checkbox"/> Edytuj	99	26.1	995.07	2020-08-26 23:03:45
<input type="checkbox"/> Edytuj	100	26.41	995.02	2020-08-26 23:03:47
<input type="checkbox"/> Edytuj	101	26.4	995.06	2020-08-26 23:03:49
<input type="checkbox"/> Edytuj	102	26.46	995.04	2020-08-26 23:03:51
<input type="checkbox"/> Edytuj	103	26.52	995	2020-08-26 23:04:01
<input type="checkbox"/> Edytuj	104	26.46	995.03	2020-08-26 23:04:03
<input type="checkbox"/> Edytuj	105	26.46	995.01	2020-08-26 23:04:05
<input type="checkbox"/> Edytuj	106	26.46	994.98	2020-08-26 23:04:07
<input type="checkbox"/> Edytuj	107	26.46	994.95	2020-08-26 23:04:10
<input type="checkbox"/> Edytuj	108	26.47	994.98	2020-08-26 23:04:12
<input type="checkbox"/> Edytuj	109	26.47	994.97	2020-08-26 23:04:14
<input type="checkbox"/> Edytuj	110	26.48	995.02	2020-08-26 23:04:16
<input type="checkbox"/> Edytuj	111	26.47	995.05	2020-08-26 23:04:18
<input type="checkbox"/> Edytuj	112	26.47	995	2020-08-26 23:04:21
<input type="checkbox"/> Edytuj	113	26.47	995	2020-08-26 23:04:24
<input type="checkbox"/> Edytuj	114	26.48	995.05	2020-08-26 23:04:26

Rys. 8.1. Sprawdzenie wartości przesłanych do bazy danych

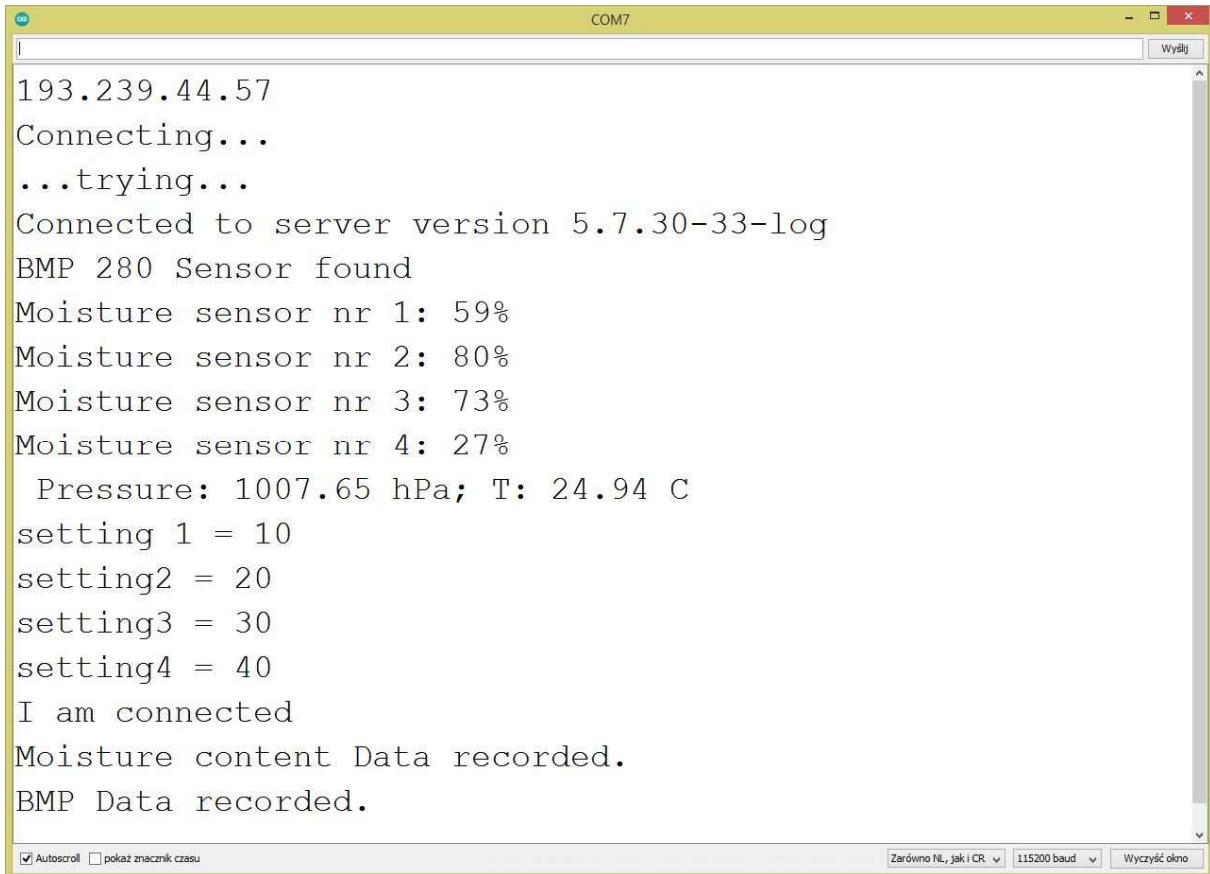
Dla części systemu związanego z Arduino wykonano również testy strukturalne (nazywane testami białej skrzynki). Polegały one na wykonaniu pomiarów elektrycznych za pomocą multimetra oraz poprzez podawanie na wejściu spreparowanych danych w celu przetestowania reakcji i poprawności wykonania.

Testy jednostkowe dla aplikacji mobilnej wykonano pod nadzorem debuggera wbudowanego w Qt Creator. Kontrolowanie fragmentu kodu polegało na zatrzymaniu aplikacji w kluczowych momentach przy użyciu breakpointów. Jest to bardzo przydatna funkcja, ponieważ w części aplikacji, która jest napisana w języku QML, nie ma możliwości wyłapywania błędów na etapie komplikacji. Błędy takie widoczne są dopiero w trakcie działania programu.



Rys. 8.2. Etap szukania błędów (debuggowania) w Qt Creator

Na końcu wykonano testy funkcjonalne (zwane testami czarnej skrzynki) oraz systemowe. Polegały one na sprawdzeniu systemu jako całości poprzez udostępnienie aplikacji osobie nie związańej z tematem pracy. Dodatkowo dla końcowego rezultatu wykonano ponownie test wykorzystując port szeregowy co pokazano na poniższym rysunku.



Rys. 8.3. Komunikacja interfejsem szeregowym UART

9. Podsumowanie i możliwości dalszego rozwoju

W niniejszej pracy dyplomowej przedstawiono proces tworzenia systemu do nawadniania roślin wraz z aplikacją mobilną, dzięki której istnieje możliwość sterowania przepływem wody oraz, w czasie rzeczywistym, wyświetlania parametrów panujących w ogrodzie takich jak: wilgotność gruntu, temperatura, ciśnienie atmosferyczne. Do cyklicznego gromadzenia odczytywanych parametrów użyto bazy danych, która przechowuje również wartości służące do sterowania systemem. Stanowi ona swoiste centrum dowodzenia, umożliwiając połączenie pomiędzy panelem operatorskim (Komputer PC, telefon z Androidem) a modułowym urządzeniem z mikrokontrolerem. Aplikację mobilną napisano w środowisku Qt, które daje możliwość budowy cross-platformowych aplikacji, interfejsów graficznych lub paneli operatorskich. Zaprojektowane urządzenie bazuje na powszechnie dostępnych, tanich podzespołach, które wiążą się również z niskimi kosztami utrzymania. Wszystkie główne cele i założenia projektowe zostały spełnione, za wyjątkiem zbudowania specjalnej obudowy chroniącej urządzenie w warunkach wykonywanej pracy.

Procesowi projektowania i powstawania systemu nieustannie towarzyszyło poczucie ogromnych możliwości jakie on daje. Wciąż możliwe są dalsze modyfikacje i perspektywy rozwoju poprzez zwiększenie liczby modułów elektronicznych lub przystosowanie urządzenia do nowych zadań. Zależnie od indywidualnych potrzeb można również dostosować aplikację mobilną dodając kolejne funkcjonalności lub rozszerzając aktualne – przykładowo wprowadzając możliwość sortowania wyświetlanych danych, bądź porównań statystycznych. Bardzo ciekawym urozmaiceniem zrealizowanego w ramach niniejszej pracy systemu byłoby dodanie zdolności do prawidłowego interpretowania danych pochodzących z zewnątrz za pomocą metod sztucznej inteligencji.

10. LITERATURA

- [1] Aplikacje QML – dokumentacja QT, <https://doc.qt.io/qt-5/qmlapplications.html> (czerwiec 2020)
- [2] Ashton K., That “Internet of Things” Thing, RFID Journal, 22(7), s.97-114, 2009
- [3] Buyya R., Dastjerdi A. V., Internet of Things: Principles and Paradigms, Morgan Kaufmann, 2016
- [4] Fajrin N., Taufik I., Ismail N., Kamelia L., Ramdhani M.A., On the Design of Watering and Lighting Control Systems for Chrysanthemum Cultivation in Greenhouse Based on Internet of Things, Materials Science and Engineering 288(1), 2018
- [5] García L., Parra L., Jimenez J.M., Lloret J., Lorenz P., IoT-Based Smart Irrigation Systems: An Overview on the Recent Trends on Sensors and IoT Systems for Irrigation in Precision Agriculture, Sensors (Basel) 20(4), 1042, 2020
- [6] Grodner M., Kokot W., Kolenda P., Krejtz K., Legoń A., Rytel P., Wierzbiński R., Internet Rzeczy w Polsce – raport, 2015
- [7] Gubbi J., Buyya R., Marusic S., Palaniswami M., Internet of Things (IoT): a vision, architectural elements, and future directions, Future Generation Computer Systems, 29(7), s.1645–1660, 2013
- [8] Hafner K., Lyon M., Where wizards stay up late: the origins of the Internet, Simon and Schuster, Nowy Jork, 1998.
- [9] Inteligentne systemy nawadniania – katalog Gardena, www.gardena.com (czerwiec 2020)
- [10] Internet rzeczy w domu i biznesie, www.mariusz-czarnecki.pl/internet-rzeczy-w-domu-i-biznesie (marzec 2020)
- [11] IoT w polskiej gospodarce – Raport grupy roboczej do spraw Internetu Rzeczy przy Ministerstwie Cyfryzacji, www.gov.pl/web/cyfryzacja/polska-przyszlosci-to-polska-z-internetem-rzeczy (marzec 2020)
- [12] Javed A., Building Arduino Projects for the Internet of Things: Experiments with Real-World Applications, Apress, Illinois 2016
- [13] Kosiński Ł., Aplikacje okienkowe w C++, <https://binarnie.pl/kurs-qt-aplikacje-okienkowe-w-cpp> (maj 2020)
- [14] Number of internet of things (IoT) connected devices worldwide in 2018, 2025 and 2030 www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology (luty 2020).
- [15] Rain Bird, <https://www.rainbird.pl/>, (czerwiec 2020)
- [15] Qt Essentials – dokumentacja QT, <https://doc.qt.io/qt-5/qtmodules.html> (maj 2020)
- [16] Qt Quick – dokumentacja QT, <https://doc.qt.io/qt-5/qtquick-index.html> (czerwiec 2020)
- [18] Timeline of the Internet of Things, www.sutori.com/story/timeline-of-the-internet-of-things--XAgMmRfEmPBq979T3kecdM13 (marzec 2020)

Załącznik nr. 1.

Tabela nr 1. Wykaz wykorzystanych elementów wraz z kosztorysem

Nazwa	Ilość [-]	Cena [zł] (allegro 06.2020)
ARDUINO Mega 2560 R3	1	42,00
Moduł sieciowy LAN W5100	1	30,49
Moduł 8 przekaźników	1	35,00
Moduł czujnik ciśnienia Bosch BMP280	1	6,99
Czujnik wilgotności gleby M335	4	23,60
Zawór elektromagnetyczny 12V - 1/2"	4	183,30
Zasilacz sieciowy AC-DC 9V/2A/18W	1	14,50
Zasilacz AC-DC 12V/5A/60W	1	22,90
Przewody, kable, zworki	69	6,90
Rozdzielnica- czwórnik	1	21,90
Przyłącze na kran węża BLACK LINE 3/4x1/2	10	18,90
Szybkozłączka złączka węża 1/2 Standard	13	49,27
Wąż ogrodowy 1/2" Green 223	130 m	143,00
Skrzynka zaworowa prostokątna standard R	1	58,00
Zraszacz Rain Bird 1804 Statyczny Wynurzalny	5	55,50
Mikrozraszacz - zraszacz podwieszany z głowicą 120	1	10,00
Suma		722,25