

Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki
Raport końcowy z Systemów Mikroprocesorowych

PLANTIETM

Autorzy:
KAMIL CHOIŃSKI
OSKAR STABLA

19 stycznia 2020

Spis treści

1	Ogólny opis projektu	2
1.1	Cel i zakres projektu	2
1.2	Kosztorys	2
1.3	Opis podobnych rozwiązań	3
1.3.1	Trutina	3
1.3.2	Arduino Irrigator System	4
2	Szczegółowy opis projektu	4
2.1	Rozwiązania techniczne - Przegląd modułów projektu	4
2.1.1	Czujnik wilgotności i temperatury powietrza: DHT11	4
2.1.2	Czujnik natężenia światła	5
2.1.3	Czujnik wilgotności gleby FC-28	5
2.1.4	Czujnik poziomu wody w zbiorniku	6
2.1.5	Wyświetlacz OLED	6
2.1.6	Przełącznik, Pompka wodna	7
2.1.7	Obudowa	8
2.2	Szczególne rozwiązania warstwy programowej	9
2.2.1	Płytką rozwojowa UNO	9
2.2.2	ESP8266	10
2.2.3	Panel sterowania przez Internet	11
3	Realizacja projektu	12
3.1	Problematyka projektu	12
3.2	Sposób wykonania	12
3.3	Problemy napotkane podczas realizacji	13
3.3.1	Połączenie między UNO, a ESP8266	13
3.3.2	Projektowanie i drukowanie obudowy	13
4	Harmonogram i podział obowiązków	14
4.1	Podział obowiązków	14
4.2	Wstępny schemat blokowy	14
4.3	Końcowy schemat blokowy	15
4.4	Schemat elektryczny	16
4.5	Harmonogram	17
4.5.1	4.11	17
4.5.2	25.11	17
4.5.3	16.12	17
4.5.4	20.01	17
4.6	Zgodność z harmonogramem	18
4.6.1	4.11	18
4.6.2	25.11	18
4.6.3	16.12	18
4.6.4	20.01	18

5	Podsumowanie	19
5.1	Osiągnięte cele	19
5.2	Pomysły na rozwój projektu	19

1 Ogólny opis projektu

1.1 Cel i zakres projektu

Nasz projekt ma na celu pomóc zabieganym ludziom, którzy nie mają czasu na zajmowanie się swoją ukochaną roślinką przez swój częsty brak pobytu w domu. Wystarczy dostęp do internetu, nic więcej.

Powstała koncepcja opiera się na systemie zdalnego zarządzania rośliną. Chcemy mierzyć parametry gleby i otoczenia takie jak wilgotność, nasłonecznienie, a nawet poziom wody w zbiorniku do podlewania. W zależności od odczytanych wartości przez płytkę rozwojową UNO połączoną z modułem ESP8266, będzie możliwe sterowanie pompką wody. Ponieważ koncepcja projektu nie jest rozległa, postanowiliśmy utworzyć panel sterowania na stronie internetowej.

1.2 Kosztorys

Kosztorys przedstawiony w Tabeli nr 1. Większość komponentów projektu została sprowadzona z chin ze względu na ich niską cenę. Niestety, czas oczekiwania na nie wydłużył się i spowodował nieoczekiwane opóźnienia w harmonogramie.

Przedmiot	Cena
Płytką rozwojową UNO	15 PLN
Moduł ESP8266	18 PLN
Czujnik wilgotności i temperatury powietrza	5 PLN
Czujnik wilgotności gleby	5 PLN
Czujnik nasłonecznienia	5 PLN
Czujnik poziomu wody	5 PLN
Przełącznik	5 PLN
Pompa wodna	22 PLN
Bateria 9V	2 PLN
Materiał na obudowę	5 PLN
Przewody, płytka uniwersalna, cyna, klej na gorąco	10 PLN
Suma	97 PLN

Tablica 1: Zakupione produkty

1.3 Opis podobnych rozwiązań

1.3.1 Trutina

‘Trutina’ z Gremon Systems przedstawiony na Rysunku nr 1, pozwala na automatyczne podlewanie danej rośliny, a także na pomiary parametrów za pomocą sensorów wbudowanych w urządzenie takich jak: pyranometer, E-box, GS3 które wysyłając pomiary, kontroluje system i wybiera kiedy podlać roślinę.

System posiada także aplikację na telefon, dzięki której możemy podglądać graficznie mierzone parametry, a także kontrolować system podlewający.

Różni się od naszego projektu aplikacją, gdyż my otrzymujemy wszystkie pomiary na stronę internetową i samoistnym wyborem kiedy będzie włączone podlewanie. Jest to także dużo większy projekt od naszego gdyż jest przystosowany do pracy w warunkach terenowych, nasz zaś do domowych.

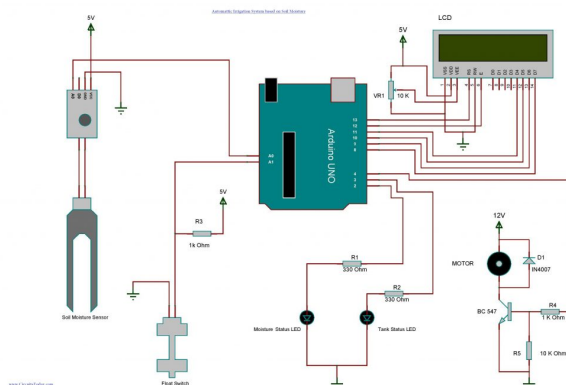


Rysunek 1: Trutina [1]

1.3.2 Arduino Irrigator System

Ten projekt przedstawiony na Rysunku nr 2, bazuje na płytce rozwojowej Arduino UNO i ma na celu automatyzację procesu pomiarów parametrów rośliny i jej podlewania. System bada wilgotność gleby i włącza pompę wodną jeśli wilgotność spadnie poniżej pewnego poziomu. Kiedy system wykryje wartość powyżej ustalonej wyłączy pompę. System posiada także wyświetlacz LCD 16x2 na którym są wyświetlane: poziom wody w zbiorniku, status czy pompka jest włączona, wilgotność gleby.

Projekt różni się od naszego sposobem wyświetlania danych - różnica wyświetlaczy, a także brakiem zapisywania pomiarów i połączeniem z siecią Wi-Fi.



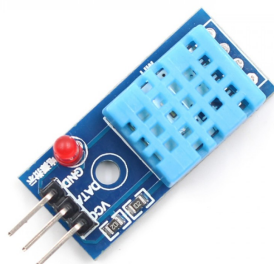
Rysunek 2: Arduino Irrigator System [2]

2 Szczegółowy opis projektu

2.1 Rozwiązania techniczne - Przegląd modułów projektu

2.1.1 Czujnik wilgotności i temperatury powietrza: DHT11

Pomiary wilgotności i temperatury powietrza są wykonywane z użyciem czujnika DHT11. Do implementacji w ArduinoIDE została wykorzystana biblioteka DHT-sensor-library, aby pomiary wykonywały się poprawnie, dzięki czemu ręczna kalibracja nie była potrzebna. Czujnik przedstawiony na Rysunku nr 3.



Rysunek 3: Czujnik DHT11

2.1.2 Czujnik natężenia światła

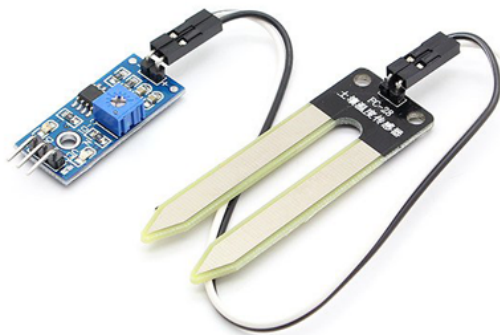
Pomiary natężenia aktualnego oświetlenia w otoczeniu rośliny są wykonywane z użyciem czujnika natężenia światła. Jego parametry były skonfigurowane na podstawie własnych doświadczeń: 0 procent - czujnik zakryty ciemnym materiałem, 100 procent - czujnik oświetlony mocnym źródłem światła. Czujnik przedstawiony na Rysunku nr 4.



Rysunek 4: Czujnik natężenia światła

2.1.3 Czujnik wilgotności gleby FC-28

Pomiary wilgotności gleby naszej rośliny są wykonywane z użyciem czujnika wilgotności gleby FC-28. Jego parametry były skonfigurowane na podstawie własnych doświadczeń: 0 procent - czujnik w powietrzu - brak powierzchni styku, 100 procent - czujnik zanurzony w wodzie. Czujnik przedstawiony na Rysunku nr ??.



Rysunek 5: Czujnik FC-28

2.1.4 Czujnik poziomu wody w zbiorniku

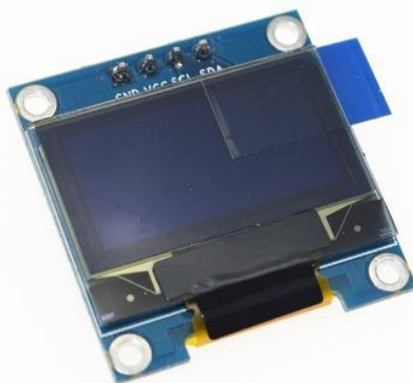
Pomiary aktualnego poziomu wody w zbiorniku z wodą do podlewania są wykonywane z użyciem czujnika poziomu wody. Jego parametry były skonfigurowane na podstawie własnych doświadczeń: 0 procent - czujnik suchy, nie zanurzony w wodzie, 100 procent - czujnik w całości zanurzony w wodzie. Czujnik przedstawiony na Rysunku nr 6.



Rysunek 6: Czujnik poziomu wody

2.1.5 Wyświetlacz OLED

Zewnętrzne wyświetlanie pomiarów wilgotności gleby i powietrza jest obsługiwane przez wyświetlacz OLED zamontowany na przodzie obudowy, aby była możliwość monitorowania jej stanu, gdy przebywamy w jej otoczeniu bez potrzeby połączenia internetowego. Do poprawnego wyświetlania danych na ekranie zostały wykorzystane biblioteki AdafruitGFX.h i AdafruitSSD1306.h. Wyświetlacz przedstawiony na Rysunku nr 7.



Rysunek 7: 0.96 calowy wyświetlacz OLED

2.1.6 Przełącznik, Pompka wodna

Do układu została zamontowana samochodowa pompka do spyskiwaczy firmy TO-PRAN, gdyż było to tańsze rozwiązanie niż inne rodzaje pompek. Jej napięcie znamionowe wynosi 12 [V], jednak w projekcie do jej zasilania używamy baterii 9 [V], które w zupełności spełnia wymagania stawiane przez dotyczące jej wydajności. Okresowe zasilanie jest realizowane za pomocą przełącznika, do którego z jednej strony wpięta jest bateria i pompka, a z drugiej zasilanie 5 [V] z płytki rozwojowej UNO zasilające sam przełącznik, GND i sygnał sterujący. Komponenty przedstawione na Rysunku nr 8.



(a) Przełącznik [6]



(b) Pompka wodna [9]

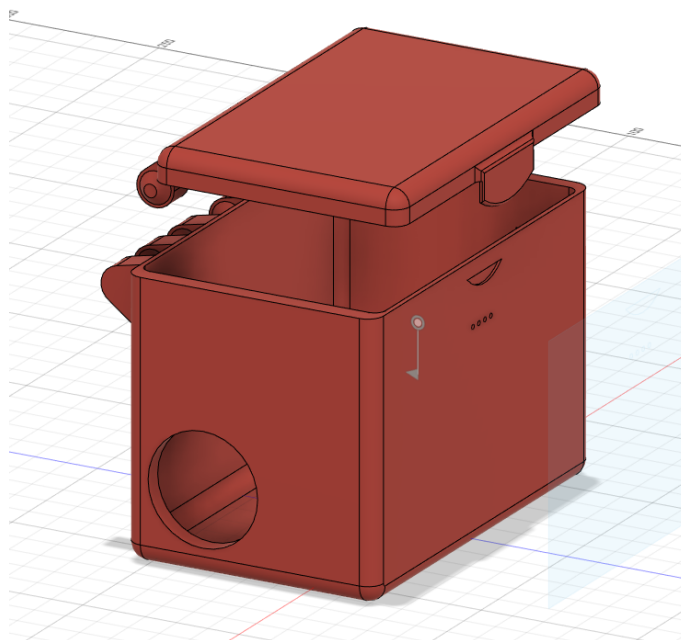


(c) Bateria 9V [10]

Rysunek 8: Komponenty pompki

2.1.7 Obudowa

Obudowa na komponenty naszego projektu została zaprojektowana w programie Fusion360. Model ten został "pocięty" (przygotowany do druku) przy użyciu programu Cura i wydrukowany na drukarce 3D Ultimaker 2+. Został użyty materiał do drukowania o nazwie PLA (Polilaktyd) i dysza drukująca wielkości 0.8mm, co jak na standardy drukowania 3D jest dosyć dużo, jednak nie była wymagana duża dokładność. Prędkość drukowania elementów była ustawiona na 40 [mm/s] i chłodzenie na 100 [%] Czas drukowania pokrywki i pudełka wyniósł 12 godzin, nie licząc nieudanych druków i czasu spędzonego na dostrajanie drukarki. Model obudowy przedstawiony na Rysunku nr 9.



Rysunek 9: Model obudowy (Fusion360)

2.2 Szczególne rozwiązania warstwy programowej

2.2.1 Płytki rozwojowa UNO

Realizacja oczekiwania płytki rozwojowej UNO na sygnał sterujący z ESP8266 została wykonana poprzez ciągłe sprawdzanie czy jej bufor przechowuje jakąś informację. Gdy wykryjemy, że bufor nie jest pusty to zczytujemy jego zawartość. Dane, które są odbierane będą zawarte we wskaźnikach początku "<" i końca ">" danej komendy. Dzięki temu rozwiązaniu możemy wykryć kiedy zaczyna się nowa instrukcja i wyczyścić pozostałości, jakie mogły pozostać w buforze, a także wykryć jej zakończenie. Zapobiega to "gubieniu danych" i otrzymywaniu powielonych instrukcji.

Płytki rozwojowa UNO przedstawiona na Rysunku nr 10.

Część kodu płytki rozwojowej UNO odbierająca dane

```
while (Serial.available())
{
    char serialChar = Serial.read();
    if (serialChar == '<')
    {
        serialMessage = "";
        serialMessage += serialChar;
    }
    else if (serialChar == '>')
    {
        stringComplete = true;
        serialMessage += serialChar;
    }
    else
    {
        if ((serialChar >= 48 && serialChar <= 122) || serialChar == ',')
            serialMessage += serialChar;
    }
}
if( stringComplete == true)
```



Rysunek 10: Płytki rozwojowa UNO

2.2.2 ESP8266

Połączenie płytki ESP8266 z Internetem jest realizowane z pomocą biblioteki ESP8266WiFi.h, która ułatwia nam połączenie poprzez dodanie klasy WiFiClient przyjmującej argumenty nazwy sieci Wi-Fi i hasła oraz łączącej się automatycznie z Internetem. Będąc w tej samej sieci Wi-Fi, łączymy się z naszym serwerem na wcześniej zadeklarowanym IP i porcie za pomocą metody stworzonej z użyciem biblioteki WebSocketClient.h. Aby ułatwić testowanie, po każdym nieudanym połączeniu restartujemy ESP, aby próbowało cały czas się połączyć.

Tak jak w płytce rozwojowej UNO, odbieranie i wysyłanie danych jest realizowane we wskaźnikach początku "<" i końca ">".

ESP8266 przedstawiona na Rysunku nr 11.



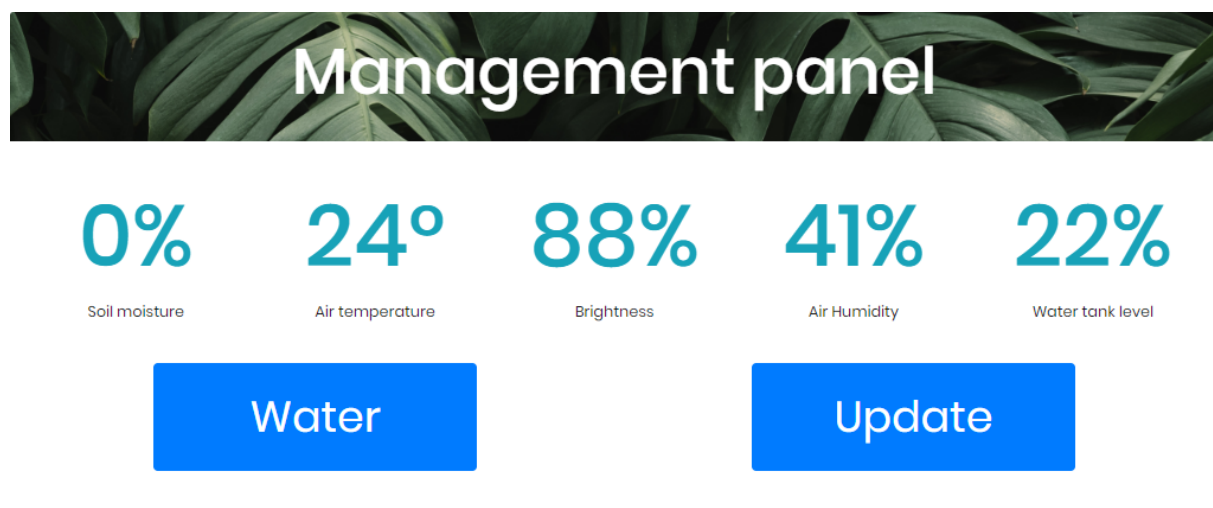
Rysunek 11: ESP8266

2.2.3 Panel sterowania przez Internet

Panel sterowania został napisany z wykorzystaniem języka HTML i stylu CSS wraz z różnego rodzaju frameworkami takimi jak Bootstrap. Zależało nam na prostocie, możliwości łatwej rozbudowy i szybkim dostępie do wszystkich potrzebnych parametrów danej roślinki. Pozwala on na sterowanie nawadnianiem i innymi, możliwymi do implementacji czynnościami poprzez kliknięcie przycisku na stronie z nazwą funkcji nawet przez osoby, które z internetem mają mało wspólnego.

Dzięki skryptom, strona automatycznie pobiera informacje z płytki ESP8266, obrazuje je w postaci tabeli, a następnie tworzy wykres w oparciu o wszystkie zebrane dotychczas dane. Wstępny okres między aktualizacjami wynosi 5 sekund, jednak w razie zaistniałej potrzeby jest możliwość bardzo łatwej zmiany tego parametru nawet w edytorze tekstowym.

Panel sterowania przedstawiony na Rysunku nr 12 wraz z przykładowymi pomiarami.



Rysunek 12: Panel sterowania

3 Realizacja projektu

3.1 Problematyka projektu

Przed rozpoczęciem prac nad projektem utworzyliśmy tabele w programie internetowym Trello, na których zapisaliśmy rzeczy potrzebne do zrobienia, a także podział kto jakimi elementami projektu się zajmie. Najtrudniejszą częścią wydawała nam się implementacja połączeń pomiędzy płytą rozwojową UNO, ESP8266 oraz serwerem, więc postanowiliśmy zacząć od tego, a dopiero później przejść do następnych aspektów. Wyzwaniem dla nas była również współpraca nad tak rozległym projektem, w którym wymagane było poprawne i integralne działanie zarówno części hardware'u, jak i tej internetowej. Zdecydowanie z pomocą przyszedł nam GitHub, ułatwiający zarówno równoległą pracę, jak i ewentualne poprawy błędów.

3.2 Sposób wykonania

Postanowiliśmy połączyć płytę rozwojową UNO obsługującą pompkę i monitorującą wartości naszych czujników z ESP8266, aby była możliwość sterowania poprzez Internet.

Płyta rozwojowa UNO cały czas oczekuje na przychodzące dane które będą zawarte we wskaźnikach początku "<" i końca ">" danej komendy, aby wykonać określoną czynność lub odesłać pomiary wykonane przez czujnik wilgotności i temperatury do ESP8266, które jest połączone z UNO poprzez piny RX i TX. UNO odsyła swoje komendy również w wskaźnikach początku i końca, aby zapobiec "gubieniu" danych.

ESP8266 jest naszym tzw. "mostem" i w każdym momencie czeka na dane czy to z UNO czy ze strony internetowej i w zależności od kogo dane otrzymuje przetrzuca je do danego odbiorcy. Program na ESP jest zrealizowany z użyciem bibliotek ESP8266WiFi, WebSocketClient, SoftwareSerial, które kolejno służą do: umożliwieniu połączenia się ESP do sieci WiFi zadeklarowanej w naszym kodzie, połączeniu się poprzez websocket jako klient do naszego serwera postawionego na laptopie, połączenia się poprzez wyprowadzenie RX i TX do płytki rozwojowej UNO.

Serwer do którego łączy się ESP8266 działa na laptopie. Połączenie jest realizowane z użyciem websocketów. Dzięki takiemu rozwiązaniu możemy wpisać adres na którym działa nasz serwer do przeglądarki i ujrzeć stronę pozwalającą na kontrolowanie naszego urządzenia. Sterowanie odbywa się za pomocą przycisków umieszczonych na głównym panelu, które po naciśnięciu wysyłają daną komendę do podłączonych klientów - w tym przypadku naszego ESP8266. Dane pomiarowe przychodzące do serwera są zapisywane w stworzonej na potrzeby projektu bazie danych i przedstawiane graficznie w postaci wykresów na stronie.

3.3 Problemy napotkane podczas realizacji

3.3.1 Połączenie między UNO, a ESP8266

Próbowaliśmy różnych typów połączeń jak I2C, albo SPI jednak poprawność testowych danych jakie otrzymywaliśmy była nikła - albo dostawaliśmy tylko część danych, albo były one w różny sposób powielone. Rozwiązaliśmy ten problem poprzez użycie wskaźników początku "<" i końca ">" przy wysyłaniu danej komendy, aby wykonać określoną czynność lub odesłać pomiary.

3.3.2 Projektowanie i drukowanie obudowy

Dużym problemem była niewiedza związana z rozmiarem, jaki powinna mieć obudowa, jednak zdecydowaliśmy, że będzie ona większa niż nasze potrzeby, aby umożliwić łatwe rozbudowanie projektu w przyszłości.

Przy początkach drukowania popełniliśmy błąd złej początkowej wysokości druku co spowodowało że model był drukowany w powietrzu a co za tym idzie druk się nie powiódł. Kalibracja drukarki 3D naprawiła nasz problem i druk został wykonany poprawnie

Nieudany druk pokrywki przedstawiony na Rysunku nr 13.



Rysunek 13: Nieudany druk pokrywki

4 Harmonogram i podział obowiązków

4.1 Podział obowiązków

Kamil zajął się:

- Uruchomieniem i szczegółowym zaznajomieniem się z płytką UNO wraz z modułem ESP8266,
- Testowaniem i implementacją czujników oraz innych komponentów elektrycznych,
- Implementacją połączenia między płytką rozwojową, a ESP8266,
- Zaprojektowaniem układu elementów na płytce PCB wraz ze stworzeniem obudowy metodą druku 3D,
- Przeniesieniem projektu na płytkę i włożeniem do obudowy

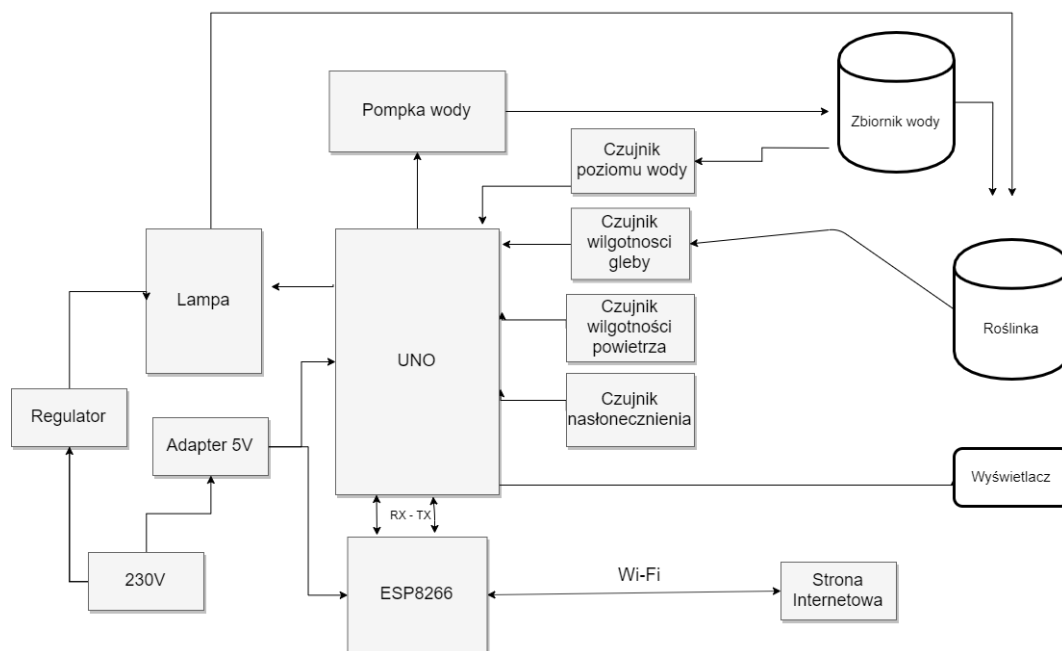
Oskar zajął się:

- Dostarczeniem potrzebnych materiałów i komponentów,
- Stworzeniem serwera i panelu sterowania,
- Implementacją połączeń między ESP8266, a serwerem,
- Stworzeniem bazy danych wraz z graficznym przedstawieniem odczytów,
- Testowaniem

4.2 Wstępny schemat blokowy

Przed rozpoczęciem pracy nad projektem stworzyliśmy schemat za pomocą programu draw.io, którego staraliśmy się trzymać, jednak ze względu na pewne komplikacje w trakcie realizacji pewne aspekty zostały zmienione, co zostało przedstawione na finalnym schemacie blokowym.

Wstępny schemat blokowy przedstawiony na Rysunku nr 14.

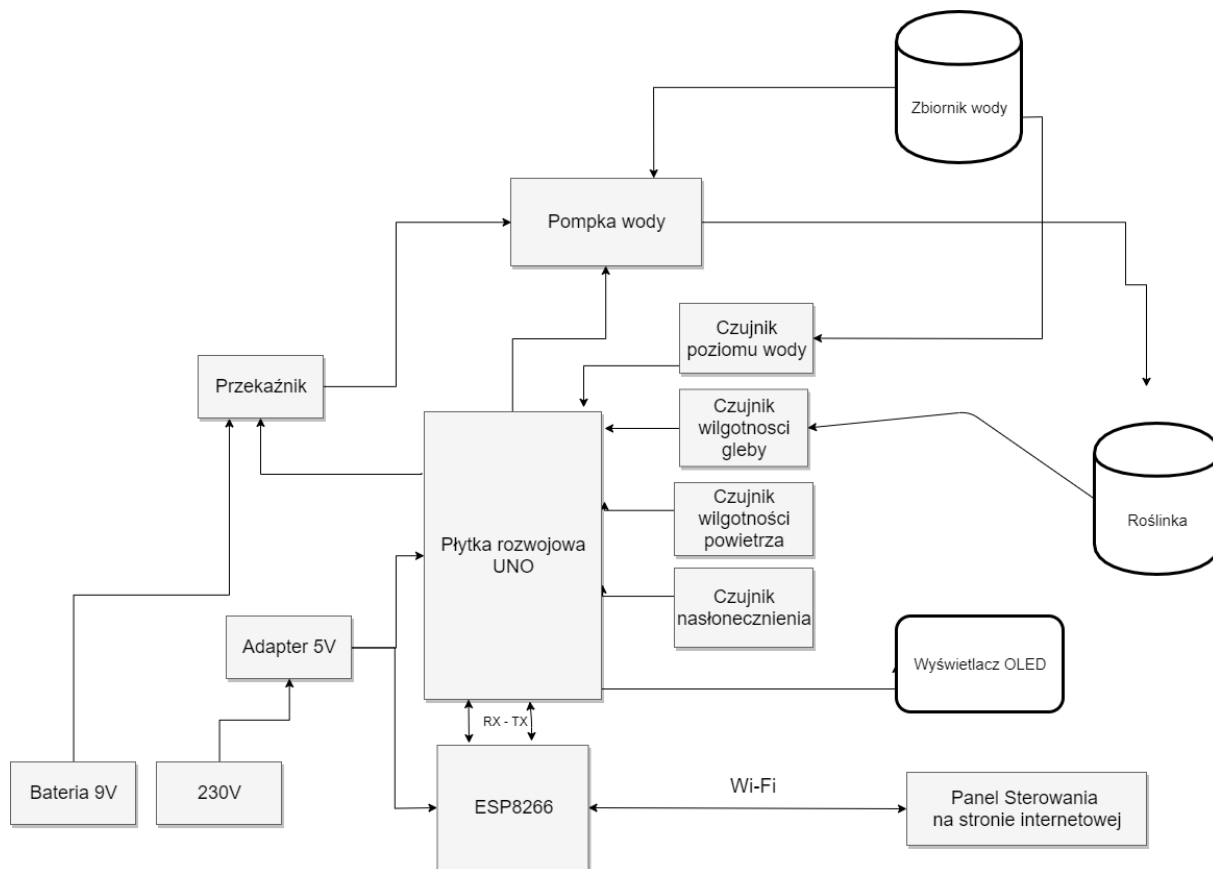


Rysunek 14: Wstępny schemat blokowy

4.3 Końcowy schemat blokowy

Można zauważyć, że nasz schemat zmienił się, jednak różnice nie są duże, a koncepcja projektu została zachowana. Zmiany zostały wprowadzone w trakcie budowy projektu ze względu na problemy z implementacją oświetlenia, a także poprzez decyzję o zasilaniu pompki wodnej z użyciem baterii 9 [V], gdyż płyta rozwojowa UNO nie była w stanie dostarczyć dostatecznie dużego natężenia prądu.

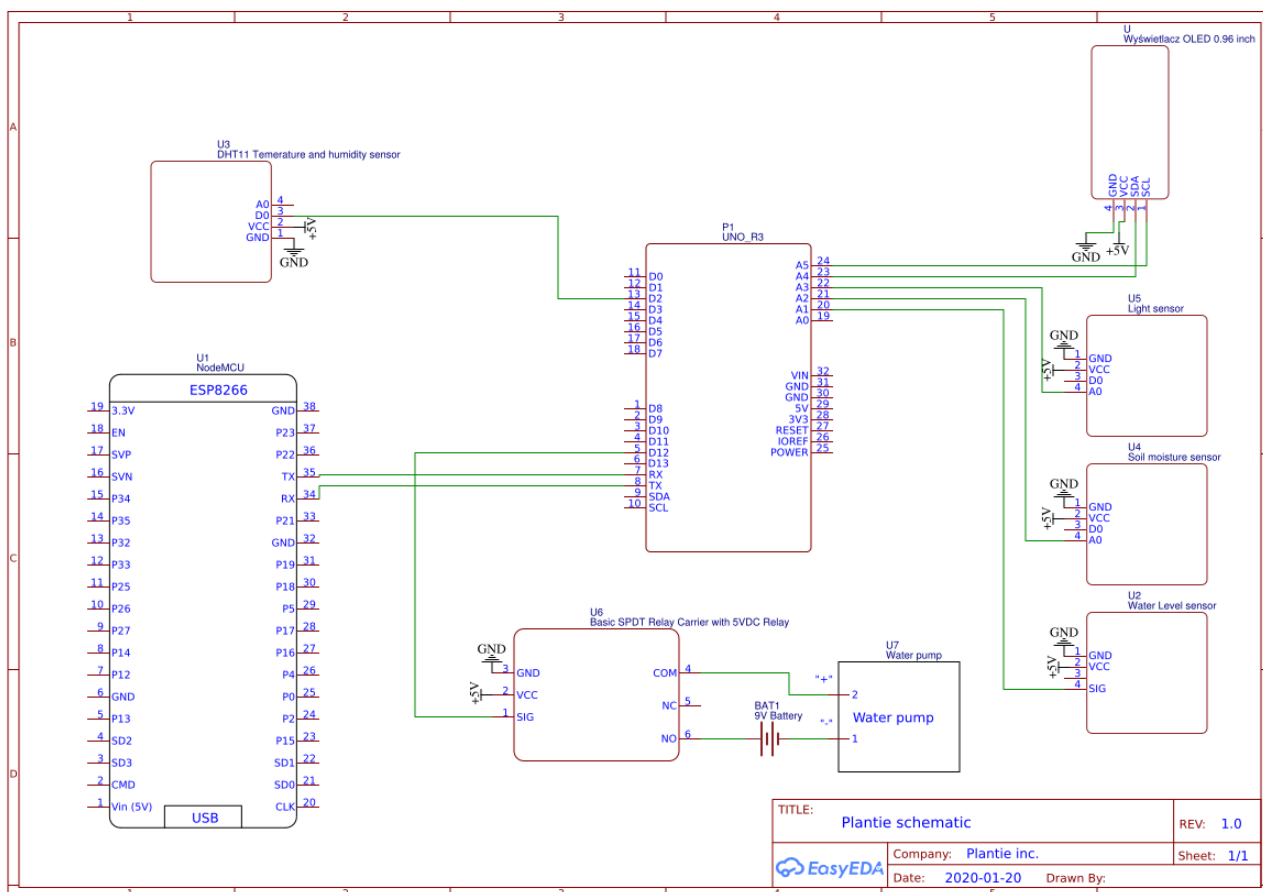
Końcowy schemat blokowy przedstawiony na Rysunku nr 15.



Rysunek 15: Końcowy schemat blokowy

4.4 Schemat elektryczny

Z użyciem programu EasyEDA został utworzony schemat elektryczny projektu. Schemat elektryczny przedstawiony na Rysunku nr 16.



Rysunek 16: Końcowy schemat blokowy

4.5 Harmonogram

4.5.1 4.11

Przeanalizowanie schematów płytki rozwojowej UNO, modułu komunikacyjnego ESP8266, czujnika wilgotności gleby, czujnika wilgotności powietrza, czujnika nasłonecznienia oraz rozwiązanie techniczne doświetlania rośliny. Stworzenie schematu elektrycznego gotowego projektu. Testowanie poprawności działania posiadanych czujników w warunkach domowych. Implementacja komunikacji z modułem ESP8266. Dopasowywanie czasów działania. Dokupienie brakujących komponentów gotowego projektu.

4.5.2 25.11

Realizacja połączeń elektrycznych na płytce stykowej i ostateczne testowanie poprawności działania. Przeniesienie projektu na płytkę uniwersalną. Przygotowanie ewentualnej obudowy i realizacja montażu systemu doświetlania. Stworzenie prezentacji na zajęcia

4.5.3 16.12

Wstępna prezentacja gotowego projektu i ocena błędów.

4.5.4 20.01

Ostateczna prezentacja z poprawką błędów.

4.6 Zgodność z harmonogramem

Okazało się, że założony harmonogram był zbyt optymistyczny. Założyliśmy zbyt wiele rzeczy do zrobienia w początkowej części harmonogramu, a nie ustaliliśmy dużo na końcową część, jednak dzięki temu mieliśmy czas na dokończenie zaplanowanych aspektów projektu i nie przekroczeniu danego czasu na ukończenie go.

4.6.1 4.11

Rozpoczęliśmy prace nad projektem. Postanowiliśmy skorzystać z aplikacji internetowej Trello do utworzenia tabel, które umożliwiły lepsze zarządzanie podziałem prac.

Przeanalizowaliśmy schematy płytki rozwojowej UNO, modułu komunikacyjnego ESP8266, czujnika wilgotności gleby, czujnika wilgotności powietrza oraz czujnika nasłonecznienia. Przetestowaliśmy również działanie każdego z czujników poprzez wykonanie przykładowych kodów na płytce rozwojowej UNO. Dowiedzieliśmy się, jakich komponentów projektu nam brakuje i zamówiliśmy je w wybranych sklepach. Stworzyliśmy wstępny schemat blokowy projektu, aby ukazać działanie poszczególnych elementów zawartych w naszym projekcie. Na ten moment postanowiliśmy nie tworzyć schematu elektrycznego, gdyż nie wiedzieliśmy jak dokładnie zrealizujemy poszczególne połączenia.

Ze względu na obszerność testowania i planowanie realizacji planu działania na nadchodzące tygodnie, nie zajęliśmy się komunikacją z modułem ESP8266 ani problemem związanym z doświetlaniem rośliny.

4.6.2 25.11

Do tego dnia zajmowaliśmy się komunikacją między płytką rozwojową UNO, ESP8266, a stroną internetową. Napotkaliśmy wiele problemów co nieplanowanie wydłużyło nam pracę na tym etapie projektu. Udało nam się uzyskać stabilne połączenie między naszymi urządzeniami. Zrealizowaliśmy połączenia UNO, ESP8266, a także podłączyliśmy do układu czujnik wilgotności i temperatury powietrza i na płytce stykowej po czym przetestowaliśmy poprawność działania.

Przeniesienie projektu na płytkę uniwersalną okazało się niepraktyczne na tym etapie projektu, gdyż nie próbowaliśmy jeszcze połączyć ze sobą wszystkich modułów projektu i finalny kod do wgrania na UNO i ESP8266 nie był gotowy. Z powodu niewiedzy jak dokładnie będzie wyglądało nasze urządzenie na płytce uniwersalnej nie przygotowaliśmy planowanej obudowy. Nie zrealizowaliśmy również systemu doświetlania.

4.6.3 16.12

Podłączyliśmy ze sobą wszystkie czujniki na płytce uniwersalnej i testowaliśmy poprawność działania. Zajęliśmy się projektowaniem obudowy w programie Fusion360, abyśmy mogli ją wydrukować na drukarce 3D i włożyć do obudowy komponenty naszego projektu. Wydrukowaliśmy obudowę i wsadziliśmy tam wszystkie moduły projektu.

4.6.4 20.01

Stworzyliśmy techniczno-reklamową prezentację projektu, a także rozległy raport końcowy.

5 Podsumowanie

5.1 Osiągnięte cele

Ku naszemu zdziwieniu, projekt zajął zdecydowanie więcej czasu niż mogliśmy się spodziewać. Brak wiedzy skutkował tym, że nie wiedzieliśmy nawet gdzie i czego szukać. Na szczęście internet w dzisiejszych czasach jest bardzo obszerny w różnego rodzaju poradniki i źródła, które pomogły nam zrealizować cele i wyjaśnić czemu poprzednie próby musiały zakończyć się fiaskiem.

Podsumowując, jesteśmy bardzo zadowoleni z ogólnego rezultatu. Niestety, nie wszystko poszło zgodnie z planem, lecz połączenie rozwiązań technicznych i cyfrowych oraz sprawienie, by działały wspólnie spowodowały powstanie ciekawego dla nas projektu oraz zdobycie bardzo cennej wiedzy. Realizacja własnego projektu mikroprocesorowego, który zmusza do samodzielnego myślenia, swoboda w jego tworzeniu oraz doprowadzenie całości do finalnego produktu jest z pewnością bardzo satysfakcjonująca i pomimo czasu, który trzeba w to włożyć, powinno być ich więcej.

5.2 Pomysły na rozwój projektu

Pomysłów na dalszy rozwój projektu jest wiele, ponieważ w trakcie prac przychodziło nam do głowy ich dużo jednak postanowiliśmy je tymczasowo zignorować i spróbować zrealizować postawione przez nas na początku cele. Nie zrealizowaliśmy również jednego aspektu projektu co też stało się naszym planem na przyszłość.

Oświetlenie roślinki przy wykryciu za małej ilości światła.

Automatyzacja procesu podlewania po odczytaniu ustawionej dolnej granicy czujnika wilgotności gleby.

Aplikacja umożliwiająca podgląd parametrów roślinki i zdalne sterowanie.

Domena i działanie projektu w chmurze bez potrzeby włączania serwera na laptopie. Do tego było by potrzebne przeniesienie naszej bazy danych na bazę danych w chmurze (np. Azure), a także implementacja ESP8266 jako serwera.

Literatura

- [1] Trutina z Gremon Systems gremonsystems.com/blog-en/things-you-didnt-know-about-automatic-watering-systems/
- [2] Arduino Irrigator System <http://www.circuitstoday.com/arduino-irrigation-plant-watering-using-soil-moisture-sensor>
- [3] Ivan Grokhotkov *ESP8266 Arduino Core's documentation*. 2017
- [4] Christian Klippel, Peter Andersson, Peter Lerup *ESP8266 core for Arduino*. 2017
- [5] *w3schools.com*. 1999-2019
- [6] Relay Module opencircuit.shop
- [7] Pompa spryskiwacza TOPRAN , www.autoczescionline24.pl/topran-2723484.html
- [8] Dokumentacja ArduinoIDE , forum.arduino.cc
- [9] Testowanie obwodów elektrycznych , www.circuitlab.com
- [10] Bateria alkaliczna 9V *VARTA*, elfadistrec.pl/pl/alkaliczne-bateria-alkaliczna-9v-6lr61-varta-industrial-9v/p/16901614