WSKAZÓWKI DLA PROJEKTANTA
I DO DALSZEGO ROZWOJU SYSTEMU
STEROWNIKA NAWANIANIA

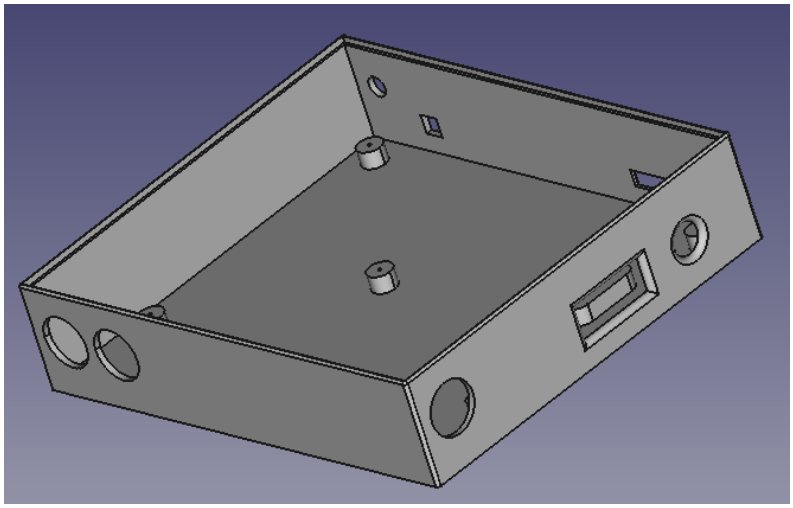
OPTI-SERV

Z BEZPRZEWODOWYMI CZUJNIKAMI ŚRODOWISKA

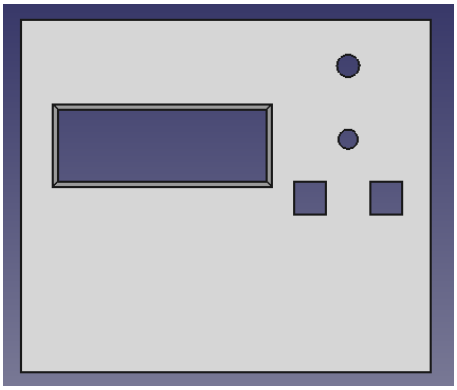
Wersja dokumentu: 1.0

1. Sterownik nawadniania

1) Dokończenie projektu obudowy sterownika, aktualna wersja jest dostępna w załączonych plikach w formacie .FCStd (projekt został wykonany w programie FreeCAD 0.20)



Rysunek 1: Obudowa sterownika – dolna część



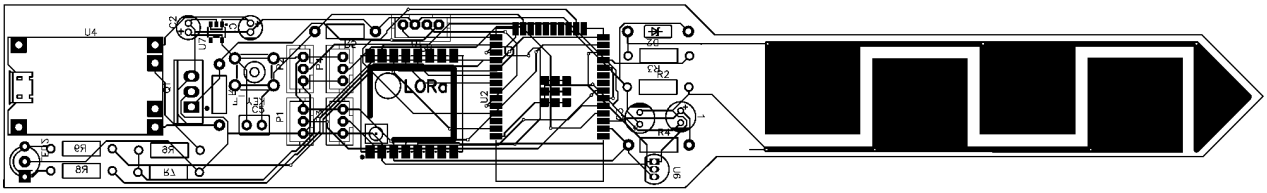
Rysunek 2: Obudowa sterownika – część górna

2 okrągłe otwory z rys. nr 1 (dolna część obudowy) oraz 1 z prawej części przeznaczone są do wprowadzenia przewodów przez dławicę. Prostokątny wyfrezowany otwór w lewej prawej części to miejsce na gniazdo ekspandera stref. Podążając dalej ku górze, wyfrezowany otwór to wlot powietrza dla czujnika wilgotności i temperatury. W górnej części obudowy znajdują się kolejno od prawej do lewej otwory na kartę pamięci, dostęp do przycisku reset oraz wyprowadzenie anteny LoRa. Opis otworów z rysunku nr 2 znajduje się w instrukcji obsługi.

2) Na dolnej listwie barierowej (rysunek nr 3, instrukcja obsługi) znajdują się aktualnie niewykorzystane zaciski S01 i S02. Możliwa jest integracja z dostępnymi czujnikami przewodowymi wymagających zasilania 3.3V lub 5V, które to jest wyprowadzone wraz z masą na znajdujące się obok zaciski (złącze Z6). Jednakże ze względu na specyfikację układu ESP32 sygnał wyjściowy z czujników musi zostać dostosowany do logiki 3.3V. Zaciski S01 i S02 są połączone liniami odpowiednio do wyprowadzeń ESP32 GPIO34 (tylko wejście) oraz GPIO33 (wejście i wyjście). W oprogramowaniu należy obsłużyć sygnały dostarczane przez integrowane urządzenia. Docelowo zaleca się stworzenie listy kompatybilnych urządzeń i umożliwienie ich wyboru z poziomu aplikacji, definiując ich funkcjonalności w oprogramowaniu układowym sterownika.

3) Pomimo możliwości wskazanych w powyższym punkcie nr 2, zaleca się rozwój czujników lub innych peryferiów w oparciu o moduły bezprzewodowe LoRa, mając na uwadze ich zasięg

(rzędu kilometrów) oraz stosunkowo nieduże wymagania energetyczne. Jednym z takich czujników, wymagających weryfikacji i sprawdzenia jest doniczkowa wersja czujnika wilgotności gleby (plik projektowy *miniSoilSS.gerber*), którego obraz PCB przedstawiono na poniższym rysunku nr 3.



Rysunek 3: Rzut z projektu doniczkowego czujnika wilgotności gleby

4) Projekt i wykonanie wspomnianego w pracy ekspandera wyprowadzeń. Projekt elektryczny można oprzeć o układy wykonawcze dla stref Z01-Z08 ze schematu elektrycznego sterownika (rys. 63 z pracy dyplomowej). Sygnały sterujące mają być podane przez 10-pinowy port z prawej części płytki laminowanej i jej obudowy (oznaczenie G4). Zasilanie 24V AC należy doprowadzić z dedykowanej listwy zaciskowej Z7 (skrajnie prawe położenie). Obsługa stref 7-16 jest już zaimplementowana w oprogramowaniu układowym w sterownika w wersji v1.0.

5) Jedną z istotniejszych funkcjonalności jaką należy wprowadzić jest implementacja aktualizacji oprogramowania układowego przez użytkownika. Może to zostać wykonane stosunkowo niedużym nakładem pracy przy wykorzystaniu funkcjonalności użytej biblioteki AsyncWebServer, obsługującej metodę HTTP POST z typem danych *multipart/form-data*. Przykładowa implementacja jest dostępna w repozytorium[i]. Aktualnie podział partycji pamięci flash jest podziałem domyślnym[ii].

2. Bezprzewodowy czujnik wilgotności i temperatury gleby i powietrza

1) Zaprojektowanie szczelnej obudowy dla element wykonawczego czujnika wilgotności gleby. Podczas badań i testów systemu wykorzystana była taśma samowulkanizująca. W płytce PCB przewidziano dwa otwory przeznaczone na zamocowanie obudowy (rys. 52, praca dyplomowa)

2) W projekcie elektrycznym pozostawiono wolną linię oznaczoną EMP doprowadzoną do GPIO39 (tylko wejście) do wykorzystania. Jeśli to okazałoby się niewystarczające, można wykorzystać linie SQ4 i SS4 wyprowadzone do gniazda P4 (JST-PH3). Port ten został zaprojektowany jako rezerwowy dla P1-P3, jednakże nic nie stoi na przeszkodzie aby zmienić jego przeznaczenie, nie zmniejszając funkcjonalności urządzenia.

3) Działaniem wartym podjęcia jest dostosowanie istniejącego ogniwa słonecznego z uchwytem, lub wykonanie dedykowanego uchwytu w celu podłączenia tego ogniwa do ładowarki TP4056

zastosowanej w module bateryjnym MB(rys. 58, praca dyplomowa). Doładowywanie akumulatora 18650 znacząco wydłuży bezobsługową pracę czujnika wilgotności.

4) Niezwykle istotnym jest podkreślenie konieczności dobrego zabezpieczenia krawędzi płytki PCB będącej elementem wykonawczym czujnika wilgotności. W badaniach oraz w trakcie testów zabezpieczenie stanowił lakier izolacyjny PVB 16[iii]

3. Aplikacja mobilna

1) Pożądany kierunkiem rozwoju jest implementacja aplikacji mobilnej na urządzenia przenośne firmy Apple (np. Iphone, Ipad). Działanie to będzie o tyle łatwiejsze, że wymaga jedynie osadzenie istniejącej aplikacji www do obsługi sterownika w komponencie przeglądarki w obrębie aplikacji. Wymagane będzie obsługa wyjątków dotyczących połączeń sieciowych oraz implementacja trwałego zapisu w pamięci urządzenia danych dostępowych do ogrodu/ogrodów (poniższy punkt nr 2)

2) W obecnej wersji v1.0 aplikacji mobilnej na system Android przewidziano i przygotowano obsługę wielu ogrodów poprzez wykorzystanie bazy danych SQLite. Obecnie obsługa ograniczona jest programowo tylko do jednego. Poprzez implementację listy oraz operacji CRUD na odpowiadających jej elementom rekordach bazy, uzyskana zostanie opisywana funkcjonalność.

3) Niezwykle użyteczną dla użytkownika funkcją mogłaby stać się graficzna prezentacja historii nawadniania w aplikacji mobilnej. Obecnie możliwe jest przeglądanie historii zdarzeń w postaci tabeli logów. Problem jaki pozostaje do rozwiązania to uzyskanie informacji np. z 7 dni wstecz z plików tekstowych zapisanych na karcie pamięci. Precyzując, dotyczy to rozpoznanego ograniczenia w postaci odczytu do około 250 zdarzeń na dzień wynikającego z asynchronicznej obsługi zdarzeń oraz współdzielenie magistrali SPI przez moduły kart pamięci i LoRa. Rozwiązaniem mogłoby stać się filtrowanie zdarzeń i przygotowywaniem ich w dedykowanym pliku tekstowym, a następnie przetwarzaniem go do postaci graficznej w aplikacji.

4) Wartym rozwagi jest implementacja klienta MQTT bezpośrednio w aplikacji mobilnej (np. przy wykorzystaniu biblioteki programistycznej Paho[iv]) w celu odbierania powiadomień z systemu Optiserv. Tu jednak po uwagę należy wziąć konieczność pracy w tle i utrzymywaniem połączenia sieciowego, co w ostatecznym rozrachunku może spowodować, że powinna być to opcja do aktywowania w aplikacji, a nie jej domyślnie włączony składnik.

i https://github.com/lbernstone/asyncUpdate/blob/master/AsyncUpdate.ino

ii https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/partition-tables.html

iii https://nowyelektronik.pl/index.php?id_product=99764&controller=product

iv https://github.com/eclipse/paho.mqtt.android