Politechnika Śląska

PROJEKT INŻYNIERSKI

System wizualizacji danych webowego interfejsu Maszyny W z użyciem mikrokontrolera ESP32

Bartosz FARUGA
Nr albumu: 305985

Kierunek: Informatyka
Specjalność: Bazy danych i inżynieria systemów

PROWADZĄCY PRACĘ
dr. inż. Tomasz Rudnicki
KATEDRA SYSTEMÓW CYFROWYCH
Wydział Automatyki, Elektroniki i Informatyki

OPIEKUN, PROMOTOR POMOCNICZY
⟨**stopień naukowy imię i nazwisko**⟩

Gliwice 2025

Tytuł pracy

System wizualizacji danych webowego interfejsu Maszyny W z użyciem mikrokontrolera ESP32

Streszczenie

(Streszczenie pracy – odpowiednie pole w systemie APD powinno zawierać kopię tego streszczenia.)

Słowa kluczowe

(2-5 slow (fraz) kluczowych, oddzielonych przecinkami)

Thesis title

Thesis title in English

Abstract

(Thesis abstract – to be copied into an appropriate field during an electronic submission – in English.)

Key words

(2-5 keywords, separated by commas)

Spis treści

Rozdział 1

Wstęp

1.1 Cel pracy

Celem niniejszej pracy jest opracowanie systemu wizualizacji danych pochodzących z webowego interfejsu Maszyny W z wykorzystaniem mikrokontrolera ESP32-S3.

Opracowane w ramach pracy dyplomowej urządzenie umożliwia odwzorowanie w czasie rzeczywistym wartości liczbowych oraz sygnałów sterujących symulatorem Maszyny W na adresowalnych diodach RGB. Ponadto system pozwala na zdalne sterowanie funkcjami symulatora zarówno za pomocą przycisków fizycznych, jak i interfejsu internetowego. System obsługuje hosting interfejsu webowego bezpośrednio na mikrokontrolerze oraz zapewnia komunikację pomiędzy ESP32 a interfejsem. Wykorzystany w tym celu protokół WebSocket umożliwia natychmiastową synchronizację zmian między stroną internetową a wyświetlaczem, co pozwala użytkownikowi na intuicyjny i interaktywny monitoring oraz kontrolę stanu Maszyny W.

1.2 Wprowadzenie do Maszyny W

Współczesne systemy komputerowe charakteryzują się wysokim stopniem złożoności zarówno pod względem architektury sprzętowej, jak i warstwy programowej. Pomimo tego, podstawowe zasady ich działania pozostają zgodne z klasycznymi założeniami architektury von Neumanna, sformułowanymi w połowie XX wieku.

W celu ułatwienia zrozumienia tych zasad w procesie dydaktycznym stosowane są uproszczone modele komputerów, których zadaniem jest eksponowanie kluczowych mechanizmów przetwarzania informacji przy jednoczesnym ograniczeniu liczby elementów składowych. Jednym z takich modeli jest Maszyna W.

Maszyna W została zaprojektowana w latach siedemdziesiątych XX wieku na Politechnice Śląskiej przez zespół kierowany przez prof. Stefana Węgrzyna jako uproszczony

model komputera, przeznaczony do nauki podstaw architektury komputerów oraz zasad działania procesora. [1]

Jej konstrukcja stanowi świadome uproszczenie rzeczywistych jednostek obliczeniowych, przy jednoczesnym zachowaniu kluczowych cech architektury von Neumanna, takich jak wspólna pamięć dla danych i rozkazów oraz sekwencyjne wykonywanie instrukcji programu.

Zgodnie z przyjętymi założeniami, Maszyna W składa się z trzech podstawowych bloków funkcjonalnych: pamięci operacyjnej, jednostki arytmetyczno-logicznej oraz układu sterującego. Pamięć operacyjna przechowuje zarówno dane, jak i rozkazy programu. Jednostka arytmetyczno-logiczna (JAML) realizuje operacje arytmetyczne i logiczne (m.in. dodawanie, odejmowanie), wykorzystując akumulator jako główny rejestr roboczy, natomiast układ sterujący odpowiada za pobieranie, dekodowanie oraz wykonywanie rozkazów.

Istotnym elementem Maszyny W jest sposób sterowania przepływem danych pomiędzy jej komponentami. Komunikacja realizowana jest za pomocą magistrali adresowej i magistrali słowa, przebieg operacji kontrolowany jest przez zestaw sygnałów mikrosterujących (np. czyt, wys, wei, il). Sygnały te, umożliwiają precyzyjne sterowanie przesyłami międzypamięciowymi i międzypamięciowo-rejestrowymi w kolejnych taktach zegara. [1] Dzięki temu możliwe jest szczegółowe śledzenie procesu realizacji pojedynczego rozkazu na poziomie mikrooperacji.

Z perspektywy dydaktycznej szczególnie istotna jest możliwość analizy pełnego cyklu rozkazu, obejmującego fazy pobrania, dekodowania oraz wykonania instrukcji. W Maszynie W faza pobrania i dekodowania rozkazu jest identyczna dla wszystkich instrukcji, co pozwala na wyraźne oddzielenie części wspólnej cyklu rozkazu od operacji charakterystycznych dla konkretnego typu instrukcji. Takie podejście sprzyja lepszemu zrozumieniu mechanizmów sterowania procesorem oraz zależności pomiędzy rejestrami i pamięcią.

Maszyna W wykorzystywana jest obecnie głównie w postaci programowego symulatora, który umożliwia implementację i uruchamianie własnych rozkazów asemblerowych. W ramach zajęć laboratoryjnych studenci projektują rozkazy, definiując sekwencje mikrosygnałów realizujących zadane operacje [2]. Proces ten wymaga nie tylko znajomości struktury Maszyny W, lecz także umiejętności analizy algorytmicznej oraz świadomego zarządzania stanem rejestrów i pamięci.

Rozdział 2

Przykładowa realizacja systemu wizualizacji

2.1 Założenia realizacyjne

Pierwszym etapem budowy systemu wizualizacji danych jest projektowanie, które powinno być poprzedzone przemyśleniami dotyczącymi możliwych realizacji oraz wyborem tych, które są optymalne i spełniają przyjęte założenia. Założenia dodatkowe nie są bezwzględnie wymagane, ale mogą poszerzyć tworzone rozwiązanie o alternatywne sposoby działania.

Założenia podstawowe:

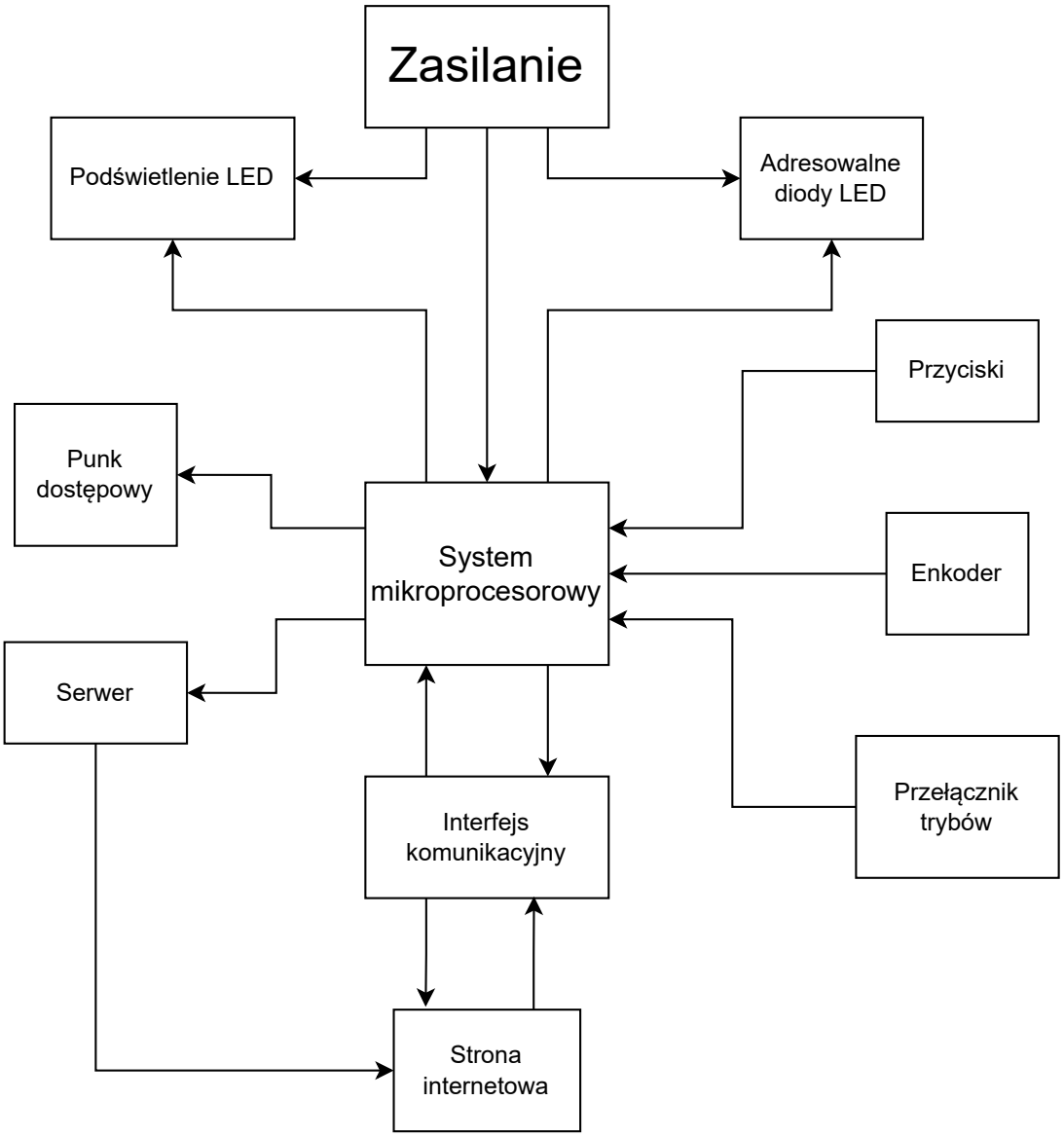
- dedykowany układ dla mikrokontrolera z odpowiednimi wyprowadzeniami,
- dedykowany układ dla LED-owych wyświetlaczy siedmiosegmentowych,
- konstrukcja kompletnego układu elektronicznego z układów mikrokontrolera, wyświetlaczy LED oraz przycisków,
- hosting strony internetowej bezpośrednio na urządzeniu wraz z przesyłem danych pomiędzy urządzeniem a stroną,
- wizualizacja informacji pobieranych ze strony internetowej na adresowalnych diodach LED,
- przesył informacji o wciśniętych przyciskach do strony internetowej,
- obsługa podświetlenia modelu Maszyny W diodami LED.

Założenia dodatkowe:

- lokalna symulacja Maszyny W na mikrokontrolerze,
- obsługa enkodera do wprowadzania wartości liczbowych.

2.2 Schemat blokowy

Na rysunku poniżej (Rys. 2.1) przedstawiono przykładowy schemat blokowy systemu wizualizacji interfejsu maszyny (SWIM). Schemat uwzględnia zarówno bloki spełniające założenia podstawowe, jak i dodatkowe.



Rysunek 2.1: Schemat blokowy SWIM

Rozdział 3

Część projektowa

3.1 System mikroprocesorowy

Ze względu na wymaganą komunikację sieciową projektu, wysoką złożoność rozwiązania oraz zapotrzebowanie na wiele portów wejścia/wyjścia należało wybrać system mikroprocesorowy zapewniający poniższe możliwości:

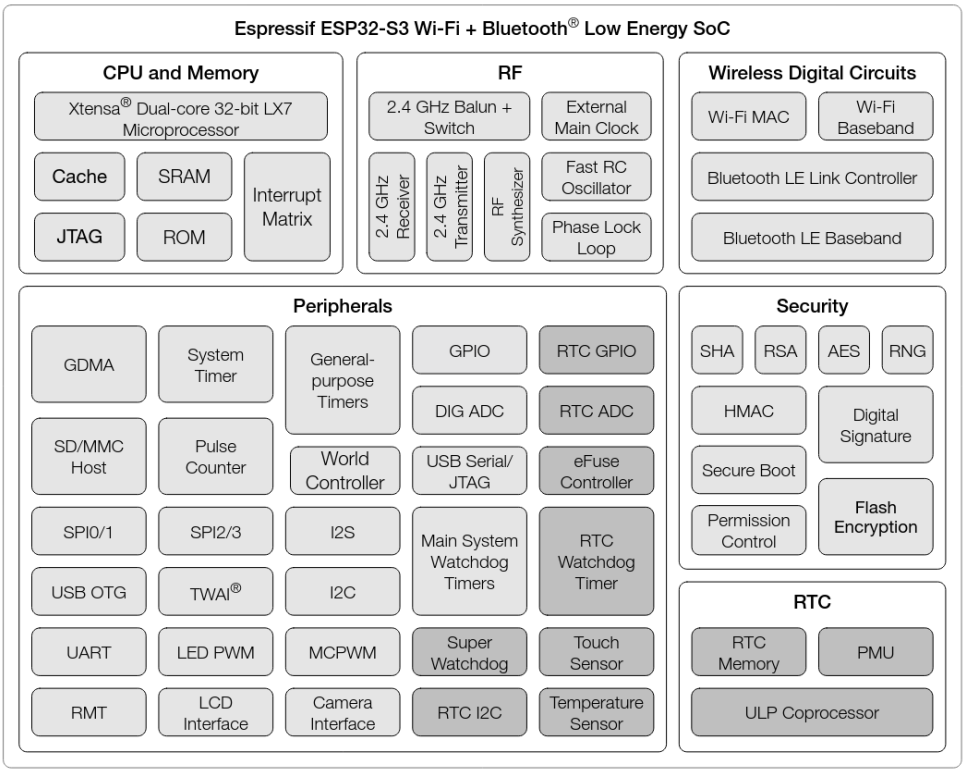
- obsługa dwóch kanałów dla adresowalnych diod LED RGB,
- obsługa 17 przycisków wraz z enkoderem,
- zapewnienie punktu dostępu internetu,
- przechowywanie danych w pamięci nieulotnej,
- serwowanie strony internetowej.

Zwracając uwagę na powyżej wymienione wymagania rozważono dwie możliwości: mikrokomputer z rodziny *Raspberry Pi* lub mikrokontroler z rodziny *ESP32*. Zarówno mikrokomputer jak i mikrokontroler spełniają wszystkie ww. założenia. Urządzenia z rodziny *Raspberry Pi* są proste w użytku, posiadają system operacyjny oparty na jądrze *Linux'a* oraz pozwalają na programowanie w językach *Python* oraz *C++*. System operacyjny pozwala również na łatwe przechowywanie danych w pamięci nieulotnej urządzenia z uwagi na obecny w nim syWyświetlacze segmentostem plików. Obecny na urządzeniach *Raspberry Pi* system operacyjny *Raspbian* będący pochodną popularnej dystrybucji *Debiana* pozwala na tworzenie własnych *daemonów* (procesy działające w tle).

Mikrokontrolery z rodziny *ESP32* nie posiadają systemu operacyjnego, a ich programowanie jest możliwe przy użyciu języków takich jak *C* oraz *C++*. Służą do tego specjalnie przygotowane środowiska programistyczne takie jak *Visual Studio Code* z rozszerzeniem *PlatformIO* a także *Arduino IDE*. Ich przewagą nad mikrokomputerami są: brak systemu operacyjnego, niższa cena, mniejsza złożoność oraz mniejsze zużycie energii.

Po rozważeniu obu opcji decydowano się na użycie mikrokontrolera z rodziny ESP32. Wybór został podyktowany pozwalającym na pełną kontrolę urządzenia językiem *C++*, niższą ceną, brakiem systemu operacyjnego, dostępnością oraz bogatą dokumentacją. Z katalogu firmy *Espressif Systems* wybrano urządzenie *ESP32-S3-WROOM-1*.

Poniższy rysunek (Rys. 3.1) z dokumentacji producenta przedstawia schemat funkcjonalny wybranego układu



Rysunek 3.1: Schemat blokowy funkcjonalny układu ESP32-S3

Własności wybranego układu zapożyczone z dokumentacji firmy *Espressif Systems*:

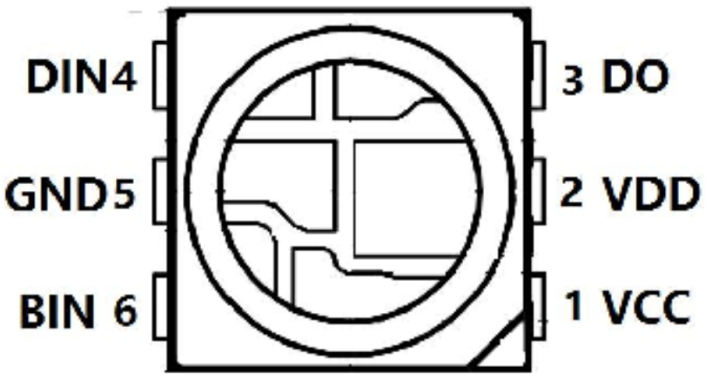
- dwurdzeniowy 32-bitowy mikroprocesor *Xtensa LX7*,
- 16MB pamięci flash, 16MB pamięci PSRAM,
- 512MB pamięci SRAM,
- 36 portów GPIO,
- komunikacja 2.4GHz WiFi (802.11b/g/n) oraz Bluetooth 5,
- wbudowana antena,
- wsparcie dla: I2C, LED PWM, USB Serial,
- maksymalny pobór prądu 300-400mA,
- zasilanie napięciem 3.3V.

3.2 Adresowalne diody LED

Zakładano wizualizację interfejsu Maszyny W na tablicy złożonej z diod LED RGB. Adresowalne diody LED RGB pozwalają na sekwencyjne zadanie kolorów każdej z 3 diod (czerwonej, zielonej oraz niebieskiej) elementu, połączonych w ciągłą linię. Przewidywana liczba pojedynczych diod LED podłączonych do SWIM przekracza 600. Znaczna liczba elementów może doprowadzać do wpływających na poprawne działanie układu spadków napięcia.

Spośród aktualnie dostępnych na rynku wersji elementu wyróżniły się dwa modele: *WS2812B* oraz *WS2815B* firmy *WORLDSEMI*. Podczas wstępnych testów obu elementów zaobserwowano następujące cechy. Diody *WS2812B* zasilane napięciem 5V wykazywały zauważalny spadek jakości odwzorowania kolorów przy liczbie elementów przekraczającej 100 (w ciągłej linii). Wykorzystanie ww. elementów niosłoby za sobą potrzebę dodatkowego wprowadzania napięcia do linii diod. Diody *WS2815B* zasilane napięciem 12V nie wykazały zauważalnych spadków napięcia przy tej samej liczbie elementów. Cecha ta zadecydowała o ostatecznym wyborze modelu *WS2815B-V1*, jego jedyną wadą jest nieznacznie większa cena w porównaniu do 5V odpowiednika.

Poniższy rysunek (Rys. 3.2) z dokumentacji producenta przedstawia schemat wyprowadzeń elementu.



Rysunek 3.2: Schemat wyprowadzeń adresowalnej diody LED RGB *WS2815B-V1*

Własności elementu:

- napięcie pracy 12V,
- pobór prądu 15mA przy maksymalnej jasności każdej z 3 kolorowych diod,
- montaż SMD,
- częstotliwość odświeżania 4kHz.

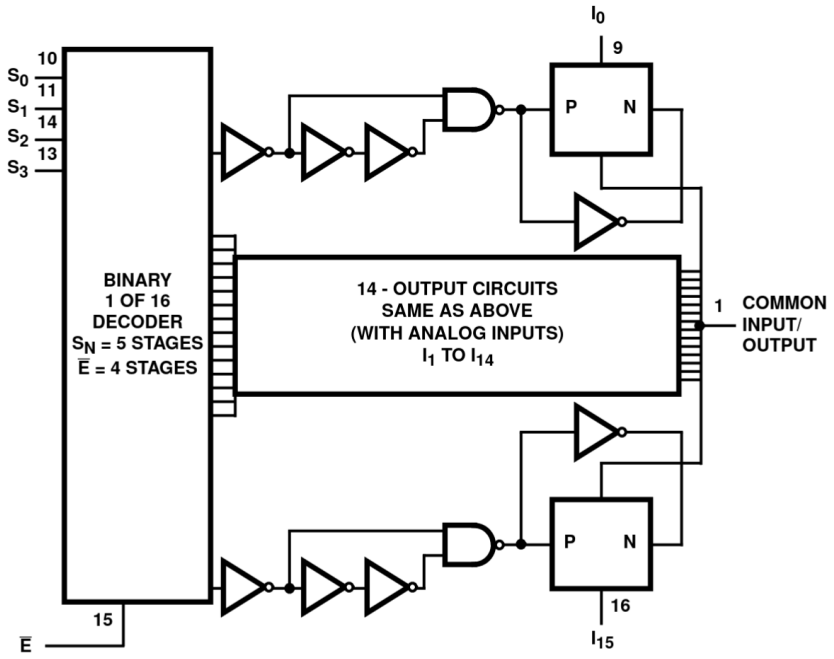
3.3 Przyciski

W celu zapewnienia pełnej obsługi Maszyny W potrzebne jest wykorzystanie 17 przycisków. Symulator w wersji podstawowej udostępnia użytkownikowi 16 możliwych sygnałów: *czyt, wys, wei, il, wyad, weak, pisz, przep, wel, wyl, dod, ode, wes, weja, wyak, wea*. Do wymienionych sygnałów należy również wliczyć przycisk *takt*, który zapewni możliwość wykonania zadanego rozkazu.

Zdecydowano się na wykorzystanie przełączników *MX Blue* firmy *Cherry*. Według dokumentacji producenta wyżej wymieniony przełącznik klawiaturowy pozwala na 50 milionów aktywacji. Pozwoli to na długoletnią, niezawodną działalność. Wykorzystano przyciski typu *Blue* oferujące słyszalny odgłos kliknięcia przy wciskaniu. Daje to użytkownikowi informację zwrotną o stanie przełącznika.

Aby ograniczyć liczbę potrzebnych wyprowadzeń dla przycisków zdecydowano wykorzystać multiplekser. W tym celu wybrano 16 kanałowy multiplekser z wyprowadzeniem *COM*. Urządzenie to pozwala na odczyt stanu przycisku wprowadzając na wejście multipleksera 4-bitowego sygnału. Cykliczny odczyt stanu wyprowadzenia *COM* wraz z odpowiednią kombinacją 4-bitowego wejścia pozwala na odczyt aktualnego stanu przycisków. Zmiana kombinacji 4-bitów wejścia oraz odczyt stanu wyprowadzenia *COM* co cykl mikroprocesora pozwala na odczyt aktualnego stanu przycisków. Z uwagi na niską cenę oraz niewielki wymiar wybrano model *CD74HC4067M* firmy *Texas Instruments*.

Poniższy rysunek (Rys. 3.3) z dokumentacji producenta przedstawia diagram funkcjonalny układu.



Rysunek 3.3: Diagram funkcjonalny multipleksera *CD74HC4067M*

Poniższa tabela (Tab. 3.1) prawdy przedstawia odczyt stanu przycisków.

Tabela 3.1: Tablica prawdy multipleksera

S0	**S1**	**S2**	**S3**	**COM**	**Kanał**
0	0	0	0	1	0
1	0	0	0	1	1
0	1	0	0	1	2
1	1	0	0	1	3
0	0	1	0	1	4
1	0	1	0	1	5
0	1	1	0	1	6
1	1	1	0	1	7
0	0	0	1	1	8
1	0	0	1	1	9
0	1	0	1	1	10
1	1	0	1	1	11
0	0	1	1	1	12
1	0	1	1	1	13
0	1	1	1	1	14
1	1	1	1	1	15

3.4 Podświetlenie LED

Chcąc zapewnić przyjazny wizualnie wygląd modelu SWIM zdecydowano się na podświetlenie następujących elementów:

- licznik rozkazów,
- akumulator,
- rejestr instrukcji,
- rejestry A oraz S,
- logo Politechniki Śląskiej.

Wykorzystano do tego 12V linie diod LED. Aby uniknąć zauważalny dla użytkowników spadków jasności podświetlenia zdecydowano na podłączenie bezpośrednio do zasilacza 12V.

3.5 Enkoder

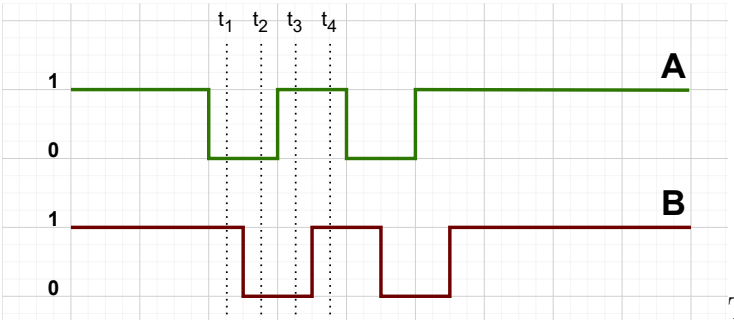
W celu zapewnienia możliwości wprowadzania wartości do: licznika rozkazów, akumulatora oraz rejestrów w lokalnym trybie działania urządzenia zastosowano enkoder obrotowy z przyciskiem.

Zamierzone działanie enkodera:

- zwiększanie/zmniejszanie wartości liczbowych w zależności od kierunku obrotu,
- aktywacja trybu wprowadzania danych poprzez przytrzymanie przycisku,
- przełączanie między polami wprowadzania danych przy użyciu szybkich kliknięć.

Enkoder pozwala użytkownikowi w prosty i intuicyjny sposób sterować parametrami urządzenia bez korzystania z interfejsu sieciowego. Cykliczny odczyt stanów dwóch kanałów enkodera umożliwiał stwierdzenie strony rotacji.

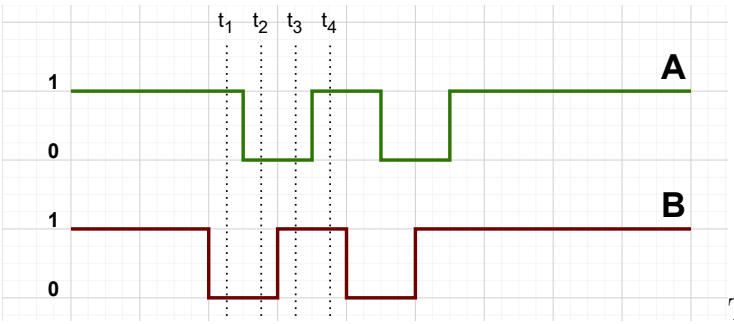
Aby określić kierunek obrotu elementu należy odczytać zmianę stanów na wyprowadzeniach urządzenia. Poniższe wykresy przedstawiają sposób odczytu obrotu w prawo (Rys. 3.4, Tab. 3.2) oraz w lewo (Rys. 3.5, Tab. 3.3).



Rysunek 3.4: Wykres stanów przy obrocie w prawo

Krok (t)	A	B
1	0	1
2	0	0
3	1	0
4	1	1

Tabela 3.2: Tablica prawdy dla obrotu w prawo



Rysunek 3.5: Wykres stanów przy obrocie w lewo

Krok (t)	A	B
1	1	0
2	0	0
3	0	1
4	1	1

Tabela 3.3: Tablica prawdy dla obrotu w lewo

3.6 Przełącznik trybów

Jednym z dodatkowych założeń projektu jest możliwość przełączania SWIM między trybami lokalnym oraz sieciowym. Aby zapewnić taką możliwość bezpośrednio z poziomu modelu wykorzystano przełącznik bistabilny. Podłączenie przełącznika bezpośrednio do systemu mikroprocesorowego pozwala na wykorzystanie przerwań w celu natychmiastowej zmiany trybu pracy urządzenia. Korekta drgań styków urządzenia została zapewniona z poziomu oprogramowania.

3.7 Serwer

Na potrzebę modernizacji narzędzia dydaktycznego, jakim jest Maszyna W, została stworzona strona internetowa (rozdział **??**) symulująca działanie modelu komputera. Projekt oraz wykonanie ww. strony internetowej nie jest częścią poniższej pracy dyplomowej. Sieciowy symulator Maszyny W został wykonany przez (AUTORZY !!!). Zadamiem SWIM jest serwowanie ww. strony internetowej na lokalnej sieci utworzonej przez mikroprocesor.

Mikroprocesor wybrany w rozdziale 3.1 umożliwia serwowanie złożonych stron internetowych z poziomu urządzenia. Aby było to możliwe planowano wykorzystać biblioteki takie jak:

- *WiFi* - obsługa połączeń i konfiguracji interfejsu sieci bezprzewodowej,
- *AsyncTCP* - asynchroniczna, nieblokująca komunikacja TCP,
- *AsyncWebServer* - asynchroniczna obsługa serwera HTTP,
- *LittleFS* - system plików dla pamięci flash mikrokontrolera (przechowywanie plików strony)
- *DNSServer* - lokalna obsługa i przekierowywanie zapytań serwera nazw domen.

Wykorzystanie owych bibliotek pozwala na:

- kontrolę połączeń z użytkownikami oraz ograniczenie ich liczby,
- komunikację przy wykorzystaniu protokołu sieciowego TCP,
- przechowywanie złożonej strony internetowej Maszyny W bezpośrednio w pamięci flash mikroprocesora,
- wykorzystanie portalu przechwytującego zapytania DNS w celu przekierowania użytkowników na wybraną stronę.

3.8 Punkt dostępowy

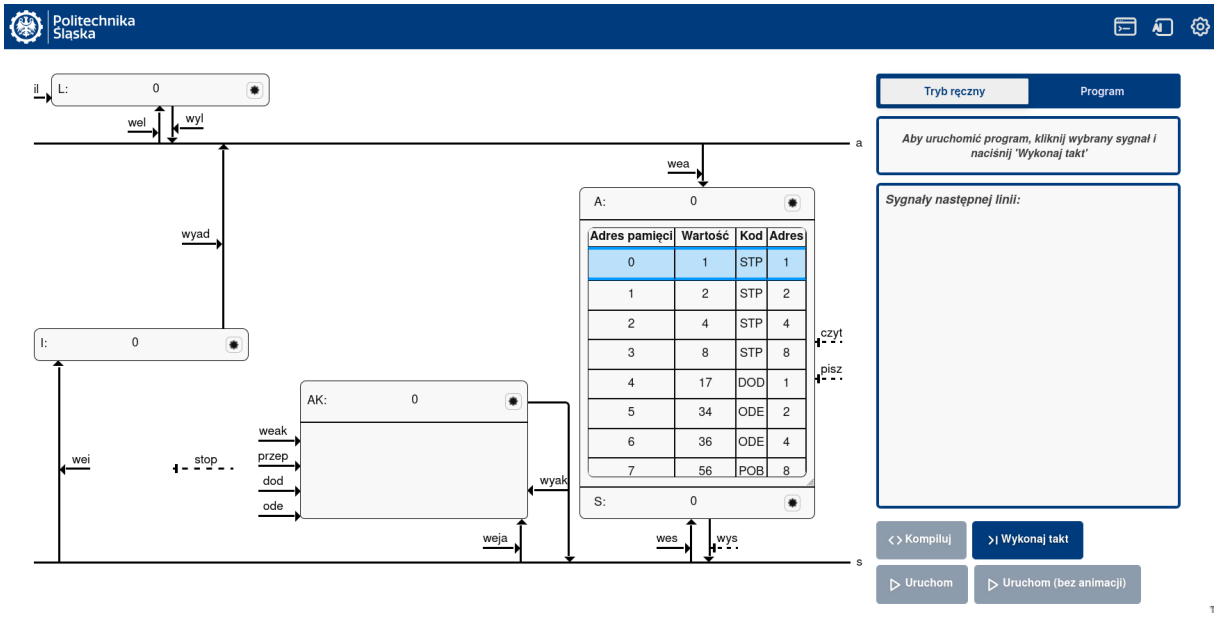
Kreacja urządzenia sieciowego wymaga zabezpieczenia rozwiązania przed niechcianym dostępem oraz atakami. Spośród dwóch możliwych rozwiązań: serwowaniu strony internetowej w globalnej sieci lub sieci lokalnej zdecydowano na rozwiązanie w pełni lokalne. Aby było to możliwe mikroprocesor tworzy własną sieć internetową na której następnie serwuje stronę internetową.

Tworzenie własnego punktu dostępu oraz izolacja go od sieci globalnej zapewnia bezpieczeństwo oraz kontrolę dostępu dla urządzenia. Bibioteki wymienione w rozdziale 3.7 umożliwiają zadanie hasła punktowi dostępu oraz ograniczenie liczby połączonych na raz użytkowników.

3.9 Strona internetowa

Poniższa praca dyplomowa wykorzystuje wykonaną na potrzeby projektu PBL, którego SWIM również był częścią, stronę modernizującą symulator Maszyny W. Niezwykle ważnym jest zaznaczyć, iż projekt oraz wykonanie ww. strony internetowej nie są częścią poniższej pracy dyplomowej. Została ona stworzona przez (AUTORZY STRONY !!!). Mając na uwadze autorów tej części projektu należy zrozumieć jej działanie, aby w odpowiedni sposób wdrożyć ją do projektu.

Strona umożliwia naukę działania Maszyny W w intuicyjny oraz łatwo dostępny sposób. Na Rys. 3.6 widnieje strona główna symulatora.



Rysunek 3.6: Strona główna sieciowego symulatora Maszyny W

Zakłada się wizualizację obecnych na powyższym rysunku elementów symulatora, takich jak: licznik, akumulator, czy pamięć operacyjna przy pomocy wymienionych w rozdziale 3.2 wyświetlaczy. Kontrolę sygnałów takich jak: czyt, wys, wei, itd. umożliwią opisane w rozdziale 3.3 przyciski.

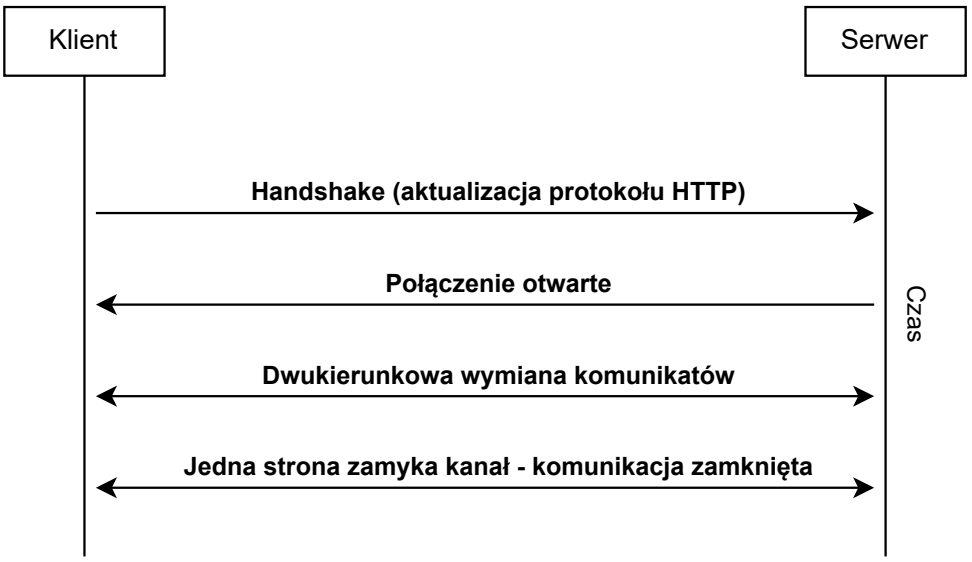
3.10 Interfejs komunikacyjny

Niezbędnym elementem poniższego projektu jest możliwość komunikacji pomiędzy mikroprocesorem, a serwowaną na nim stroną w czasie rzeczywistym. Wyjściowo, nie jest możliwe przesyłanie informacji o aktualnie znajdujących się wartościach w polach tekstowych strony internetowej, lub o aktualnie wybranych sygnałach.

Standardowa implementacja serwera posiada jedynie możliwość komunikacji za pomocą protokołu *HTTP*. Wykorzystanie takiego rozwiązania wymagałoby wysyłania cyklicznych żądań do serwera co powodowałoby wysokie zużycie zasobów oraz opóźnienia w działaniu. W celu zapewnienia wydajnej oraz szybkiej komunikacji pomiędzy stroną internetową a serwerem wykorzystano technologię *Websocketów*.

Websocket jest dwukierunkowym protokołem komunikacyjnym. Działa zarówno w warstwie aplikacji (strona internetowa) oraz na serwerze (mikroprocesor). Umożliwia wymianę danych w czasie rzeczywistym, bez konieczności każdorazowego inicjowania żądania przez użytkownika. Wykorzystuje asynchroniczną komunikację, co pozwala na nieblokujące odbieranie i nadawanie złożonych informacji.

Poniższy schemat ilustruje przebieg komunikacji *WebSocket*, pomiędzy klientem (stroną internetową), a serwerem.



Rysunek 3.7: Schemat komunikacji przy wykorzystaniu protokołu *Websocket*

3.11 Zasilanie

Mając na uwadze wszystkie wymienione w powyższym rozdziale elementy SWIM, dokonano analizy wymagań energetycznych układu w celu dobrania odpowiedniego zasilacza. Adresowalne diody LED oraz podświetlenie wymagały zasilania napięciem 12V, natomiast mikrokontroler ESP32 oraz multiplekser korzystały z napięcia 3,3V.

Przeprowadzono szacunkowe obliczenia maksymalnego poboru prądu:

- dla linii diod LED WS2815B-V1, przy założeniu pełnej jasności każdej z trzech diod RGB oraz wykorzystaniu 600 elementów, przewidziano maksymalny pobór prądu na poziomie około 9A,
- dla podświetlenia LED przewidziano pobór prądu rzędu 0,5A, uwzględniając wszystkie elementy podświetlane jednocześnie,
- dla mikrokontrolera ESP32-S3 oraz układów towarzyszących przewidziano pobór prądu około 600mA przy pełnym obciążeniu interfejsów peryferyjnych i aktywnym WiFi.

Na podstawie powyższych obliczeń dobrano zasilacz impulsowy o napięciu wyjściowym 12V i mocy minimalnej 330W, co zapewniało bezpieczny margines względem przewidywanego maksymalnego poboru prądu. Zastosowanie zasilacza o większej mocy umożliwiało również stabilne działanie układu przy chwilowych skokach poboru prądu przez linie LED.

Zasilanie mikrokontrolera ESP32 oraz innych układów niskonapięciowych realizowano przy użyciu przetwornicy DC-DC obniżającej napięcie z 12V do 3,3V. Rozwiązanie to zapewniało izolację układów niskonapięciowych od wysokoprądowych linii LED oraz stabilne napięcie zasilania, eliminując spadki i zakłócenia w pracy mikrokontrolera. Dodatkowo, w celu ochrony układu przed odwrotną polaryzacją, należy zapewnić odpowiednią ochronę. Takie podejście zapewniło bezpieczne i niezawodne działanie całego systemu SWIM.

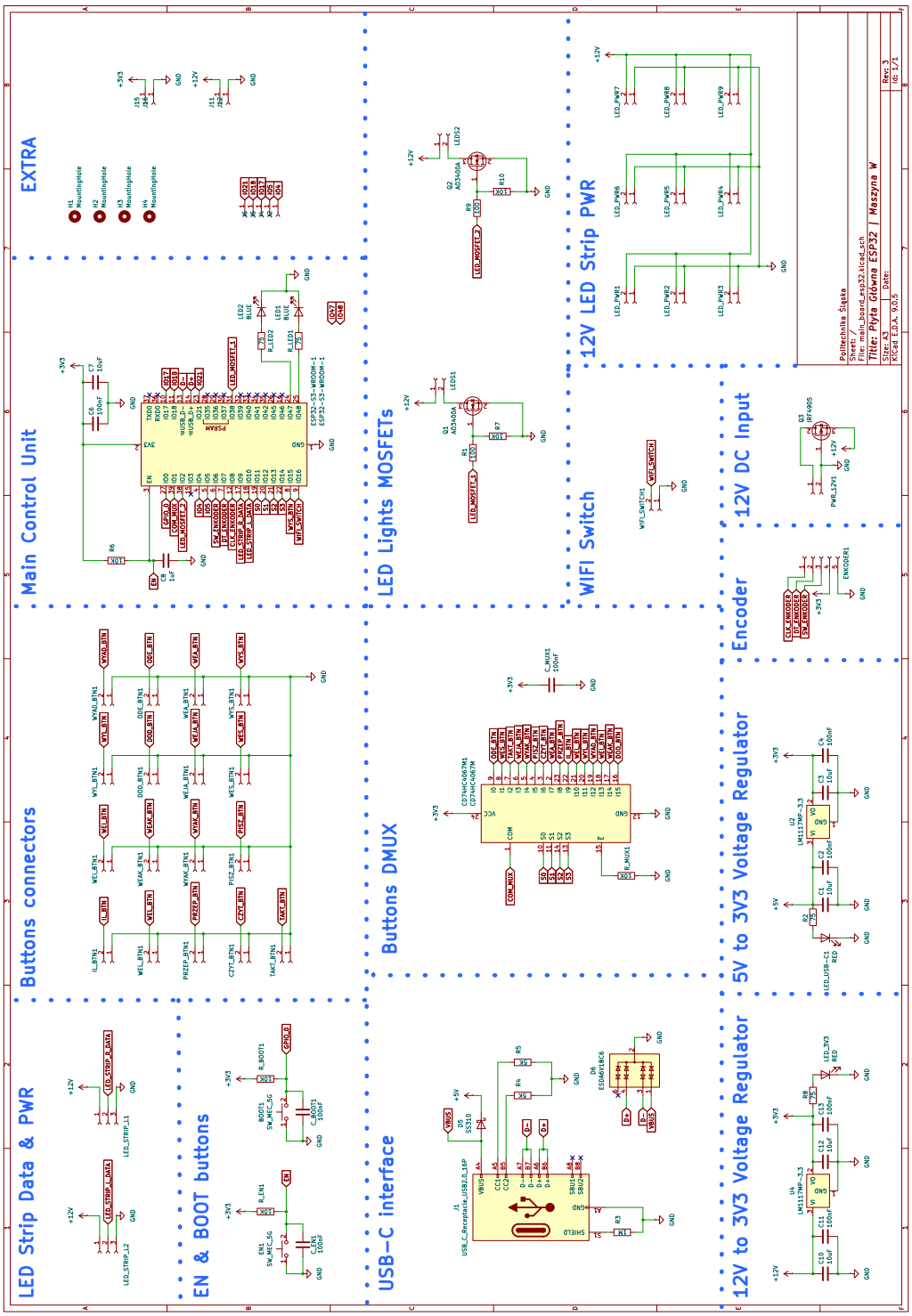
Rozdział 4

Schematy bloków systemu wizualizacji

Poniższy rozdział ma na celu prezentację schematów elektronicznych dla systemu wizualizacji stanu interfejsu Maszyny W. Schematy zostały wykonane przy użyciu otwartego oraz darmowego programu *KiCAD*. Rysunki zostały sformatowane w widoczny sposób z uwagi na ich znaczący rozmiar. Rysunki zostały przygotowane w formacie papieru A3.

4.1 Płyta główna SWIM

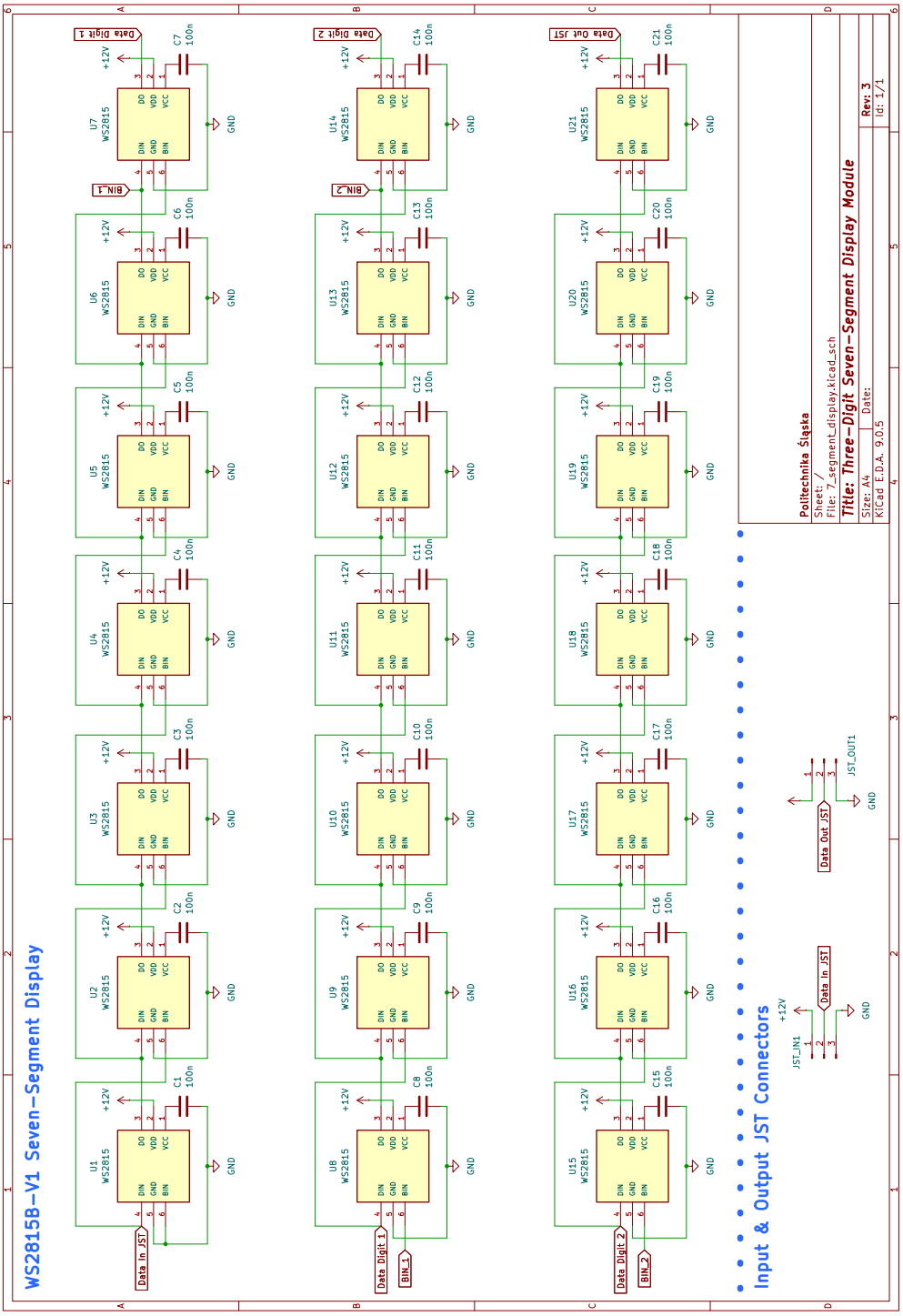
Na rysunku (Rys. 4.1) przedstawiono schemat płyty głównej systemu wizualizacji interfejsu Maszyny W.



Rysunek 4.1: Schemat płyty głównej z mikroprocesorem ESP32-S3

4.2 Wyświetlacz siedmiosegmentowy trzycyfrowy

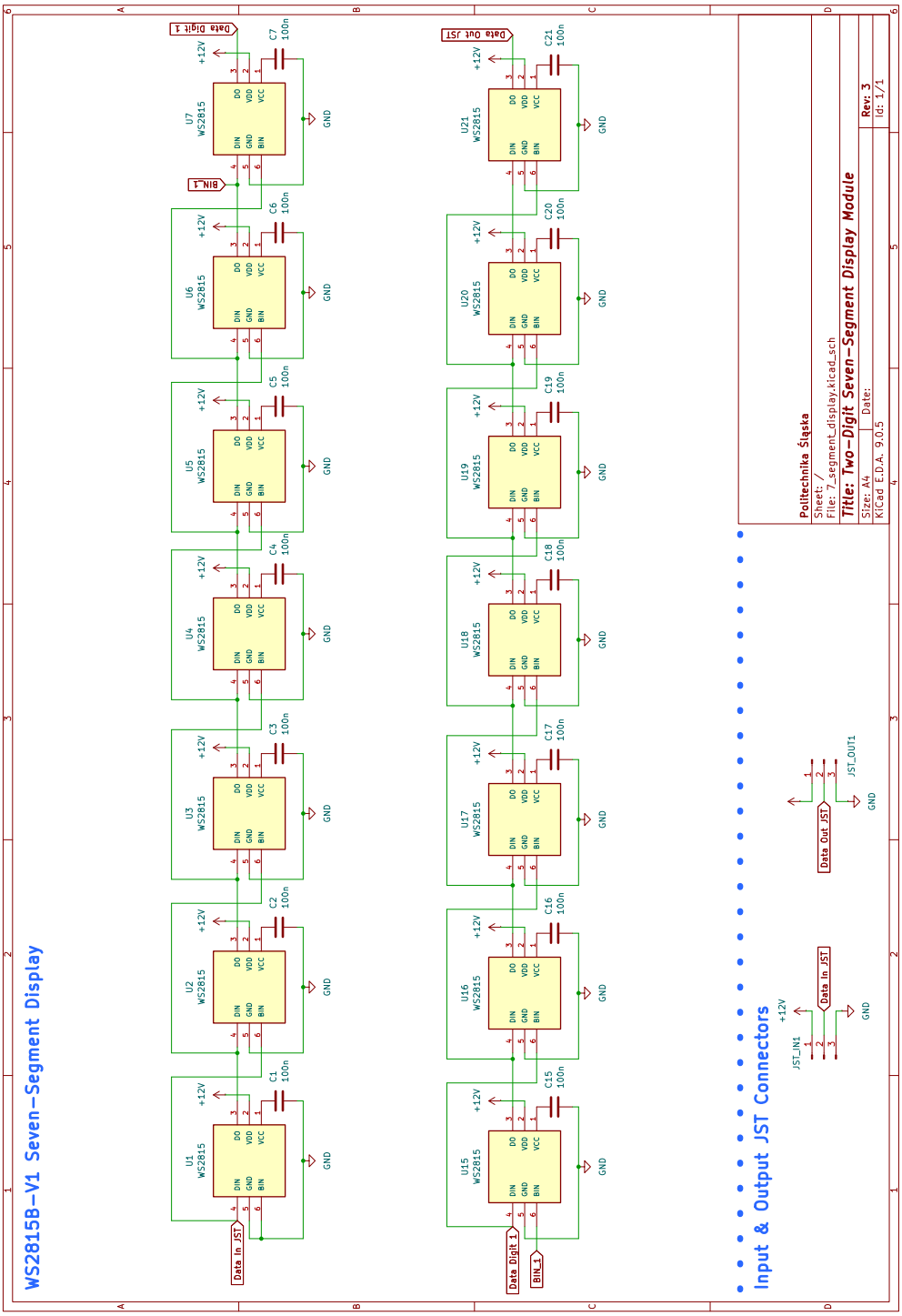
Na rysunku (Rys. 4.2) przedstawiono schemat wyświetlacza trzycyfrowego skonstruowanego z połączonych ze sobą w linii ciągłej diod LED.



Rysunek 4.2: Wyświetlacz siedmiosegmentowy trzycyfrowy

4.3 Wyświetlacz siedmiosegmentowy dwucyfrowy

Na rysunku (Rys. 4.3) przedstawiono schemat wyświetlacza dwucyfrowego skonstruowanego z połączonych ze sobą w linii ciągłej diod LED.



Rysunek 4.3: Wyświetlacz siedmiosegmentowy trzycyfrowy

4.4 Połączenia pomiędzy blokami

TODO !!!

Rozdział 5

Część konstrukcyjna

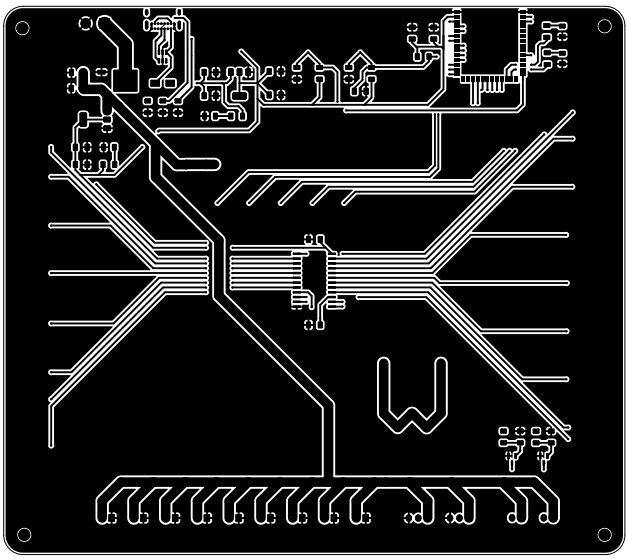
5.1 Obwody drukowane

Obwody drukowane zostały wykonane na specjalne zamówienie przez zewnętrzną firmę *JLCPCB*. Mozaikę ścieżek płytek drukowanych zaprojektowano na bazie schematów z rozdziału 4. Wykorzystano w tym celu program *KiCAD*. Odpowiednie rysunki przedstawiające warstwę górną oraz dolną płytek drukowanych przedstawiono w podrozdziałach 5.1.1 oraz 5.1.2.

5.1.1 Płyta główna

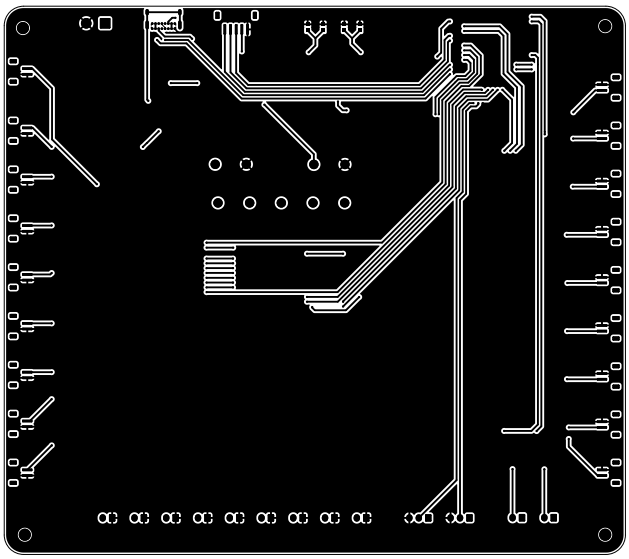
Na rysunkach 5.1 oraz 5.2 przedstawiono mozaiki obwodów drukowanych płyty głównej SWIM.

Front płytki zapewnia możliwość przymocowania elementów urządzenia wyłączając wyprowadzenia oraz porty. Rysunek 5.1 przedstawia płytkę pomniejszoną dwukrotnie względem wymiarów rzeczywistych.



Rysunek 5.1: Widok warstwy górnej płyty głównej

Rewers płytki zapewnia możliwość przymocowania wyprowadzeń oraz portów urządzenia. Umożliwia również podłączenie dodatkowych elementów przy wykorzystaniu dostępnych wyprowadzeń dodatkowych. Rysunek 5.1 przedstawia płytkę pomniejszoną dwukrotnie względem wymiarów rzeczywistych.

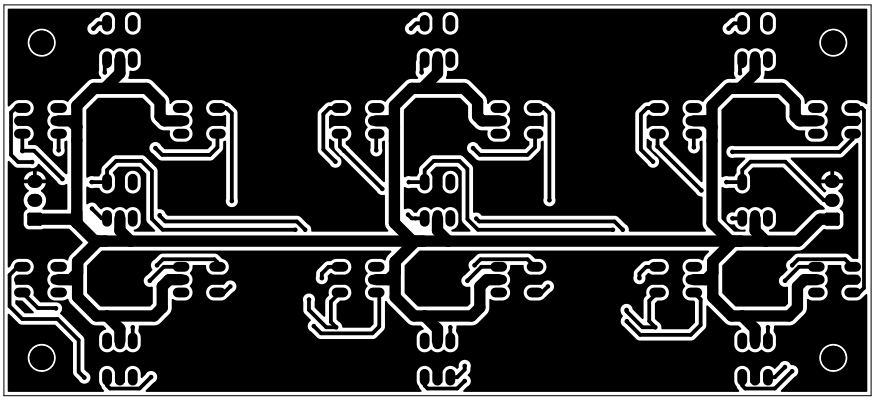


Rysunek 5.2: Widok warstwy dolnej płyty głównej

5.1.2 Wyświetlacze siedmiosegmentowe

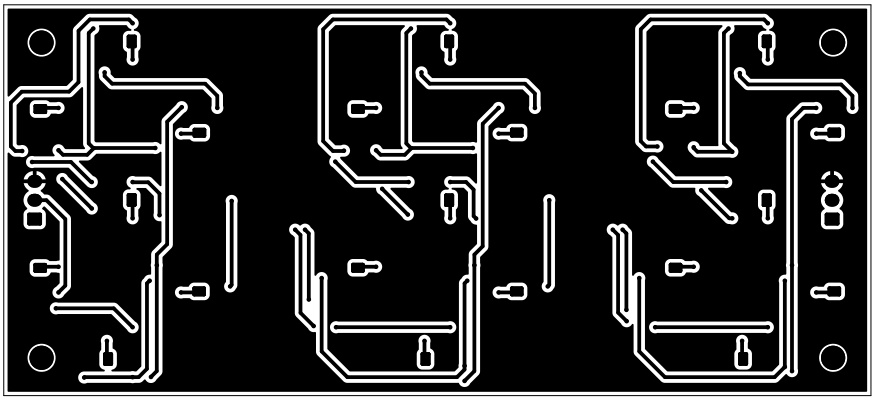
Na rysunkach 5.3 oraz 5.4 przedstawiono mozaiki obwodów drukowanych wyświetlacza trzycyfrowego.

Front płytki zapewnia możliwość przymocowania diod LED. Rysunek 5.3 przedstawia płytkę w rzeczywistych wymiarach.



Rysunek 5.3: Widok warstwy górnej wyświetlacza trzycyfrowego

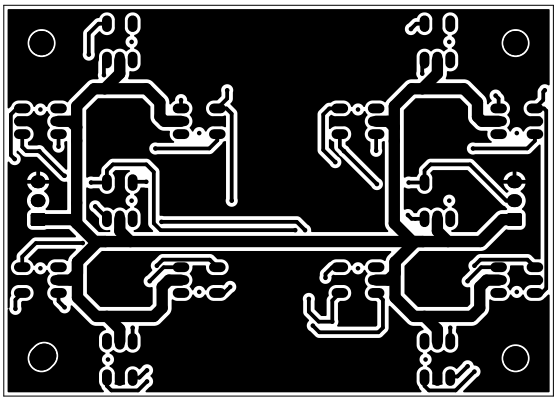
Rewers płytki zapewnia możliwość przymocowania kondensatorów filtrujących. Rysunek 5.4 przedstawia płytkę w rzeczywistych wymiarach.



Rysunek 5.4: Widok warstwy dolnej wyświetlacza trzycyfrowego

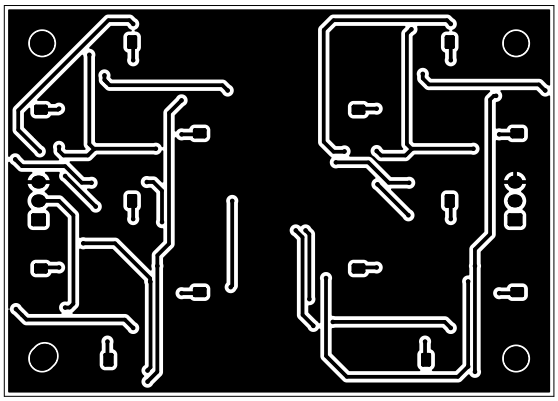
Na rysunkach 5.5 oraz 5.6 przedstawiono mozaiki obwodów drukowanych wyświetlacza dwucyfrowego.

Front płytki zapewnia możliwość przymocowania diod LED. Rysunek 5.5 przedstawia płytkę w rzeczywistych wymiarach.



Rysunek 5.5: Widok warstwy górnej wyświetlacza dwucyfrowego

Rewers płytki zapewnia możliwość przymocowania kondensatorów filtrujących. Rysunek 5.6 przedstawia płytkę w rzeczywistych wymiarach.



Rysunek 5.6: Widok warstwy dolnej wyświetlacza dwucyfrowego

5.2 Opis złącz

5.2.1 Płyta główna

Złącze zasilania

Urządzenie jest wyposażone w 2-wyprowadzeniową listwę zaciskową typu *ARK*. Złącze pełni funkcję głównego wejścia zasilania i przyjmuje napięcie 12V. Układ elektryczny wyposażono w zabezpieczenie przed odwrotną polaryzacją, uniemożliwiające uszkodzenie płytki w przypadku błędnego podłączenia przewodów.

Złącze USB-C

W celu umożliwienia łatwego programowania i komunikacji z mikroprocesorem, płytkę wyposażono w złącze USB typu C (tryb USB 2.0 Device). Wyprowadzone są jedynie linie *D+* oraz *D−*. Linie zasilania *VBUS* są wykorzystywane do zasilania sekcji USB oraz wykrywania podłączenia komputera. Aby zapewnić zabezpieczenie przed prądem wstecznym wykorzystano diodę *Schottky'ego* połączoną w kierynku zaporowym na linii zasilania. Pozostałe wyprowadzenia USB-C niewykorzystywane przez układ zostały pozostawione niepodłączone, zgodnie ze specyfikacją.

Złącze enkodera

Do podłączenia impulsatora zastosowano 5-wyprowadzeniowe złącze typu *JST-ZH* o rastrze 1.5mm. Wyprowadzone sygnały:

- *+3V3* – zasilanie elementu,
- *GND* – masa układu,
- *CLK* – pierwszy kanał kwadraturowy enkodera,
- *DT* – drugi kanał kwadraturowy enkodera,
- *SW* – złącze przycisku enkodera

Sygnały prowadzone są do wejść cyfrowych mikroprocesora. Redukcja drgań styków zapewniono z poziomu oprogramowania.

Złącza przycisków

Każdy przycisk posiada dedykowane 2-wyprowadzeniowe złącze typu *JST-ZH* o rastrze 1.5mm. Jedno wyprowadzenie każdego złącza podłączone jest do masy *GND*, drugie – do wejścia cyfrowego mikroprocesora. Linie sygnałowe wyposażono w rezystor polaryzujący do zasilania z poziomu wbudowanych w mikroprocesor rezystorów. Filtrację sygnałów zapewniono od strony oprogramowania.

Złącze przełącznika trybów

Przełącznik trybów posiada dedykowane 2-wyprowadzeniowe złącze typu *JST-ZH* o rastrze 1.5mm. Jedno wyprowadzenie złącza podłączone jest do masy *GND*, drugie – do wejścia cyfrowego mikroprocesora wykorzystującego wbudowany w mikroprocesor rezystor polaryzujący do zasilania. Filtrację sygnałów zapewniono od strony oprogramowania.

Złącza linii LED

Dwa wyprowadzenia linii LED wykorzystują 3-wyprowadzeniowe złącza typu *JST-XH* o rastrze 2.5mm, prowadzące sygnały do linii diod LED typu *WS2815B-V1*. Wyprowadzenia:

- *+5V* – zasilanie diod,
- *DATA* – linia danych,
- *GND* – masa układu.

Dodatkowo zastosowano kondensator filtrujący do masy na każdej z adresowalnych diod obecnych na płytkach drukowanych, zgodnie z dokumentacją producenta.

Złącza zasilania podświetlenia modelu

Płytka zawiera dodatkowe dwa 2-wyprowadzeniowe złącza typu *JST-XH* o rastrze 2.5mm. Każde złącze wspiera kontrolę zasilania 12V z poziomu mikroprocesora. W tym celu zostały wykorzystane tranzystory typu *MOSFET*.

Złącza dodatkowego wprowadzania zasilania

Projekt układu przewiduje potrzebę dodatkowego wprowadzania zasilania na linie adresowalnych diod LED. W tym celu płytkę wyposażono w 9 złącz typu *JST-XH* o rastrze 2.5mm. Każde złącze zapewnia zasilanie 12V.

5.2.2 Wyświetlacze segmentowe

Złącza wejścia/wyjścia

Każdy z wyświetlaczy segmentowych (zarówno trzycyfrowe, jak i dwucyfrowe) posiadają dwa złącza *JST_IN* oraz *JST_OUT* typu *JST-XH* o rastrze 2.5mm. Złącza umożliwiają łączenie wyświetlaczy w ciągłe linie. Wyprowadzenia:

- *+5V* – zasilanie diod,
- *DATA_IN/DATA_OUT* – wejście/wyjście linii danych,
- *GND* – masa układu.

Rozdział 6

Opis oprogramowania

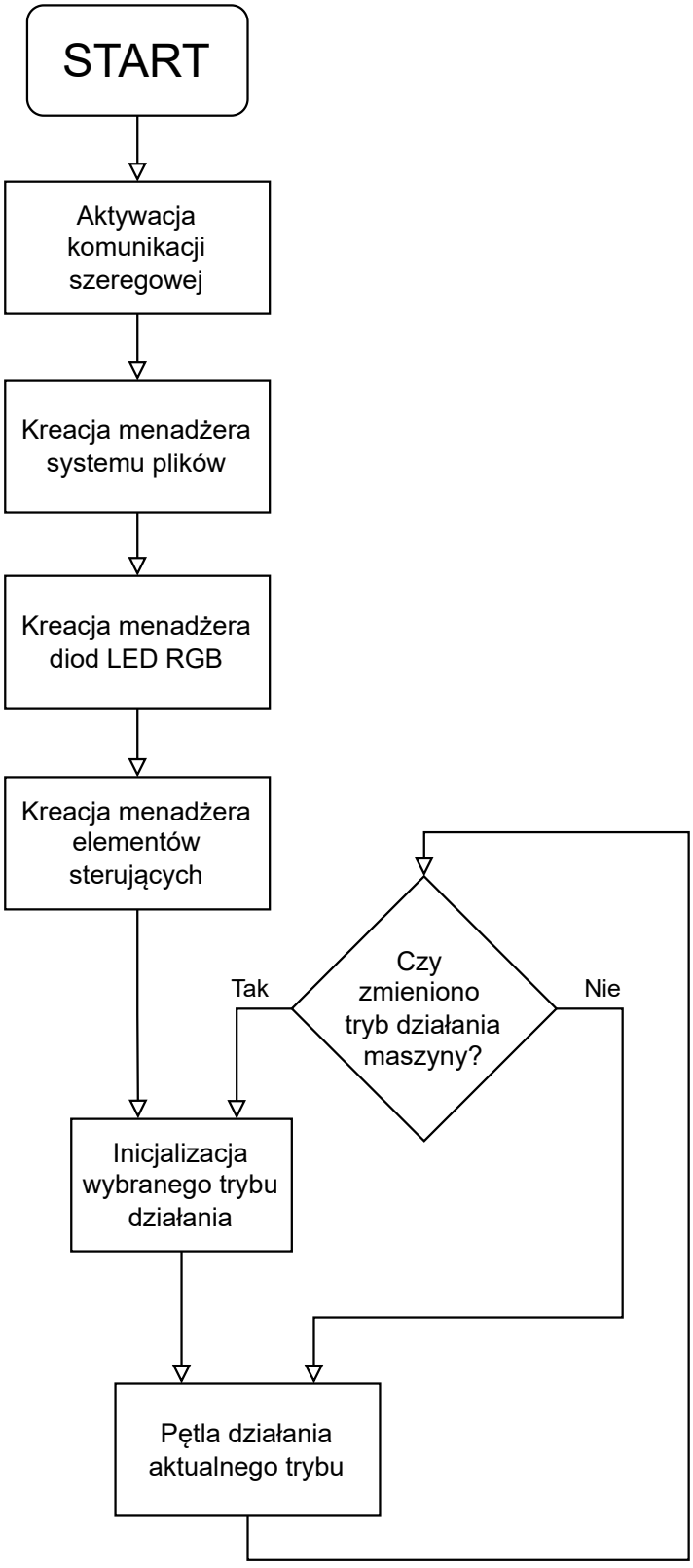
Program mikrokontrolera ESP32 został napisany w języku *C++* za pomocą aplikacji *Visual Studio Code* wykorzystującej, dedykowane programowaniu mikrokontrolerów, rozszerzenie *PlatformIO*. Kod wynikowy zajmuje około 900kB. Mikrokontroler ma do zrealizowania następujące zadania:

- obsługa adresowalnych diod LED,
- obsługa przycisków, enkodera oraz przełącznika trybów,
- obsługa pamięci nieulotnej/systemu plików,
- obsługa trybu sieciowego SWIM,
- obsługa trybu lokalnego SWIM,
- kontrola podświetlenia LED.

Kompletny kod źródłowy mikrokontrolera wraz z instrukcją instalacji został zamieszczony w dodatku (TODO Dodatek z kodem źródłowym).

Poniższe rozdziały opisują w szczegółowy sposób działanie oprogramowania urządzenia. Rozdziały 6.1, 6.3 oraz 6.4 opisują działanie elementów oprogramowania funkcjonujących zarówno w trybie sieciowym, jak i lokalnym urządzenia. Opis działania owych elementów jest kluczowy do zrozumienia funkcjonowania obu trybów. Metody zarządzania ww. elementami oraz sposób działania obu trybów SWIM zostały omówione w rozdziałach 6.5 oraz 6.6.

Poniższy schemat (Rys. 6.1) przedstawia punkt wejściowy programu urządzenia. Rozszerzony opis działania widocznych na nim elementów znajduje się w ww. rozdziałach.



Rysunek 6.1: Ogólny diagram sekwencji działania oprogramowania

6.1 Obsługa adresowalnych diod LED

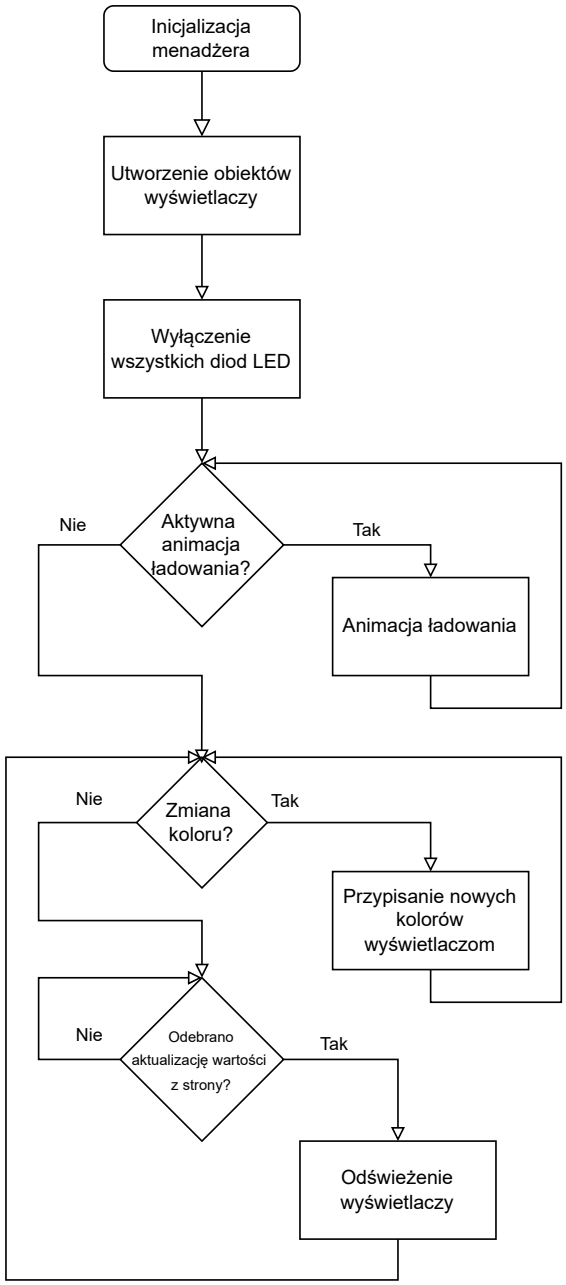
Stworzenie wydajnego oraz łatwego w rozbudowie oprogramowania wymagało zaprojektowania odpowiedniej architektury kodu. W tym celu stworzono obiekt *DisplayManager*. Jest on odpowiedzialny za zarządzanie wszystkimi adresowalnymi diodami LED urządzenia. Implementacja obejmuje następującą funkcjonalność:

- tworzenie/usuwanie wyświetlaczy,
- zmianę koloru poszczególnych elementów,
- zadawanie wartości do wyświetlenia elementom,
- odświeżanie koloru wszystkich elementów naraz,
- wyświetlanie animacji testowych/ładowanie.

Podczas tworzenia implementacji dla obsługi diod zdecydowano się na abstrakcyjne rozwiązanie problemu wielu różnych typów wyświetlanych wartości. Powstała w tym celu klasa wirtualna *LedElement* pozwala na proste odnalezienie elementu w linii diod LED, dynamiczne zarządzanie pamięcią urządzenia oraz tworzenie klas pochodnych dla specyficznych wartości takich jak: sygnał, magistrala, czy pamięć operacyjna. Sposób wyświetlania wartości liczbowych przy użyciu diod LED opisano w rodziale 6.2.

Wykorzystanie abstrakcyjnego podejścia przy kreacji architektury oprogramowania pozwoliło w łatwy sposób odizolować i wykryć błędy. Pozwala na łatwą rozbudowę lub modyfikację układu wyświetlaczy.

Implementacja menadżera w ostatecznej wersji oprogramowania wywołuje metodę odświeżającą wszystkie diody LED zgodnie z poniższym schematem blokowym (Rys. 6.2). Widoczny na rysunku schemat ma na celu prezentację podstawowych funkcji zarządzania diodami LED.

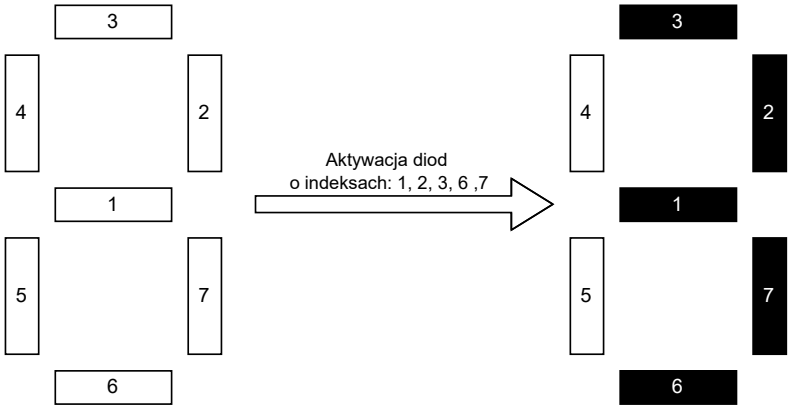


Rysunek 6.2: Diagram sekwencji działania menadżera diod LED

6.2 Obsługa wyświetlaczy segmentowych

Stworzenie czytelnego oraz łatwego w obsłudze oprogramowania wymagało specjalnego obiektu zarządzającego diodami LED wyświetlającymi wartości liczbowe. Model Maszyny W został skonstruowany z 9 wyświetlaczy 3-cyfrowych oraz 8 wyświetlaczy 2-cyfrowych. Zarządzanie 43 cyframi wymagało stworzenia obiektu przechowującego indeksy diod odpowiedzialnych za daną wartość liczbową.

Powyżej opisany problem należało rozwiązać na etapie konstrukcji schematów elektrycznych wyświetlaczy (rozdziały 4.2 oraz 4.2). Rysunek 6.3 przedstawia kolejność połączenia diod w wyświetlaczu segmentowym. Wyjście ostatniej diody segmentu jest podłączone do wejścia pierwszej diody kolejnego segmentu lub elementu wyświetlacza.



Rysunek 6.3: Wizualizacja schematu działania wyświetlaczy segmentowych

Obiekt *Segment* obecny w oprogramowaniu SWIM jest odpowiedzialny za zarządzanie diodami LED wchodzącymi w skład segmentu. Przechowuje indeksy diod, oraz zawiera implementację metody wyświetlania cyfr.

6.3 Obsługa przycisków, enkodera, diod płyty głównej oraz przełącznika trybów

Podobnie jak w przypadku obsługi diod LED (rozdział 6.1), przygotowano rozwiązanie umożliwiające zarządzanie wszystkimi elementami sterującymi. W tym celu utworzono obiekt *HumanInterface* nakładający warstwę abstrakcji na niskopoziomowe operacje związane z obsługą wejść urządzenia.

Obiekt *HumanInterface* jest odpowiedzialny za ustawienie wyprowadzeń mikroprocesora. Przy inizjalizacji ustawia wyprowadzenia jako wyjście dla multipleksera, przycisku *takt*, przełącznika trybów oraz enkodera. Inicjalizowane również są wyprowadzenia dla dwóch diod LED obecnych na płycie głównej.

Obiekt posiada dedykowane metody do obsługi następujących funkcji urządzenia:

- obrót enkodera,
- zwracanie stanu przycisku enkodera,
- multipleksowanie przycisków,
- zwracanie stanu przycisków,
- zwracanie stanu przełącznika trybów,
- eliminacja drgań styków,
- kontrola diod LED płyty głównej,
- kontrola podświetlenia LED modelu.

6.4 Obsługa pamięci nieulotnej/systemu plików

Rozmiar plików strony internetowej serwowanej przez urządzenie oraz potrzeba aktualizacji strony wymagała stworzenia sposobu na zapis plików w pamięci nieulotnej urządzenia. Dodatkowo wykorzystanie diod LED RGB pozwala na personalizację kolorów poszczególnych elementów SWIM. Strona internetowa posiada opcję ustawienia koloru dla: wyświetlaczy segmentowych, sygnałów oraz magistrali. Aby zapewnić możliwość przechowywania wybranej konfiguracji również należało zapisać wartości w pamięci nieulotnej urządzenia.

Wyżej wymienione powody wymagały wykorzystania zewnętrznej biblioteki *LittleFS*. Pozwala ona stworzyć osobną partycję w pamięci flash urządzenia dla systemu plików. Zapisane w ten sposób dane nie ulegają usunięciu po odłączeniu zasilania urządzenia.

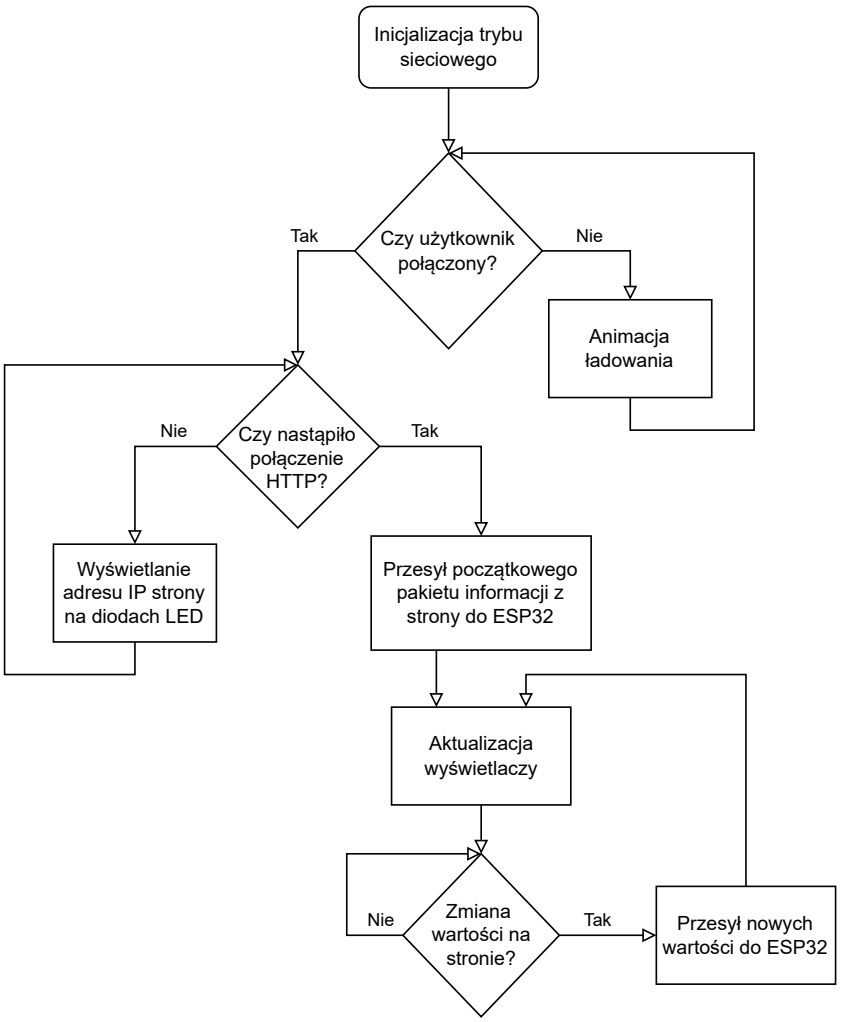
Urządzenie przechowuje w pamięci plik z konfiguracją kolorów w formacie *json*. Podczas uruchamiania urządzenia konfiguracja jest odczytywana, a kolory zostają odpowiednio ustawione. Użytkownik ma możliwość nadpisania pliku z konfiguracją wykorzystując odpowiednią opcję w ustawieniach strony internetowej serwowanej przez urządzenie.

6.5 Obsługa trybu sieciowego SWIM

Głównym trybem działania urządzenia jest wizualizacja interfejsu sieciowego symulatora Maszyny W. Powstały w tym celu obiekt *W_Server* zajmuje się kreacją oraz zarządzaniem następującymi funkcjami:

- tworzenie punktu dostępowego,
- tworzenie serwera DNS,
- serwowanie strony internetowej,
- obsługa komunikacji pomiędzy stroną internetową, a serwerem,

Poniższy schemat blokowy (Rys. 6.4) przedstawia uproszczony sposób działania trybu sieciowego.



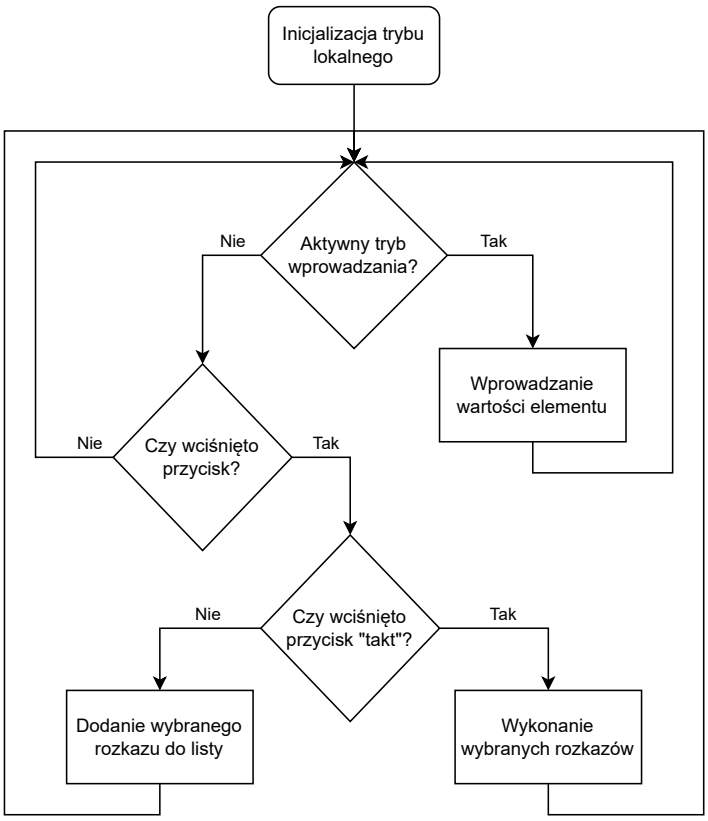
Rysunek 6.4: Uproszczony diagram sekwencji działania trybu sieciowego

Sposób działania pomija szczegóły zwiazane z inicjalizacją poszczególnych komponentów takich jak: tworzenie serwera DNS czy serwowanie strony internetowej.

6.6 Obsługa trybu lokalnego SWIM

Dodatkowym trybem działania urządzenia jest symulacja działania Maszyny W bezpośrednio na urządzeniu. Powstały w tym celu obiekt *W_Local* działa analogicznie do obiektu *W_Server*. Podobnie jak menadżer trybu sieciowego wykorzystuje obiekty *DisplayManager*, *HumanInterface* oraz *FileSystem*.

Poniższy schemat blokowy (Rys. 6.5) prezentuje uproszczony sposób działania trybu sieciowego.



Rysunek 6.5: Uproszczony diagram sekwencji działania trybu lokalnego

Sygnały wybierane poprzez naciśnięcie odpowiednich przycisków zostają dodane do listy sygnałów. Obiekt posiada funkcję wykrywającą, czy Maszyna W pozwala na daną kombinację rozkazów. Przykładem wykluczających się wzajemnie rozkazów są: czyt/pisz, wyak/wys lub wyl/wyad. W przypadku wykrycia wciśnięcia niedozwolonego przycisku urządzenie nie doda rozkazu do listy. Po wciśnięciu przycisku *takt* obiekt wykona rozkazy z listy w kolejności dodania.

W celu zapewnienia poprawnego działania trybu lokalnego wprowadzono zabezpieczenie przed przekroczeniem wartości możliwej do wyświetlenia na wyświetlaczach dwu oraz trzycyfrowych. W przypadku przekroczenia zadanego limitu liczba zostanie wyrównana do największej/najmniejszej możliwej do wyświetlenia.

Rozdział 7

Obsługa

Tutaj trzeba będzie opisać jak działa maszyma i jak ją można obsługiwać

Bibliografia

[1] Robert Tutajewicz Alina Momot. „Maszyna W – jak zaprojektować prosty rozkaz”. W: *MINUT* (2019), s. 24–35.

[2] Robert Brzeski. „Prawidłowe tworzenie rozkazów asemblerowych dla Maszyny W cz.2”. W: *MINUT* (2020), s. 136–148.

Dodatki

Spis skrótów i symboli

SWIM	system wizualizacji interfejsu maszyny
RGB	red green blue
LED	light emitting diode
HTTP	Hypertext Transfer Protocol
DNS	Domain Name Server
TCP	Transmission Control Protocol
USB	universal serial bus
ARK	Anschlus Reihenklemme Klemme
GND	ground
CAD	computer-aided design

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2     throw std::string ("unknown number of clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4     throw std::string ("You should set a maximal number of
          iteration or minimal difference — epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6     throw std::string ("Both number of iterations and minimal
          epsilon set — you should set either number of iterations
          or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

Spis tabel