



WYDZIAŁ ELEKTRONIKI,
TELEKOMUNIKACJI
I INFORMATYKI

Imię i nazwisko studenta: Wojciech Paderewski

Nr albumu: 184823

Poziom kształcenia: Studia pierwszego stopnia

Forma studiów: stacjonarne

Kierunek studiów: Elektronika i telekomunikacja

Profil: Komputerowe systemy elektroniczne

PRACA DYPLOMOWA INŻYNIERSKA

Tytuł pracy w języku polskim: Budzik synchronizowany przez WiFi

Tytuł pracy w języku angielskim: Alarm clock synchronized via WiFi

Opiekun pracy: dr hab. inż. Paweł Wierzba

Streszczenie

W pracy przedstawiono proces projektowania zegara synchronizowanego z internetowym serwerem czasu. Zegar ten posiada funkcję budzika, współpracuje z serwerem Home Assistant, zaś wizualizacja czasu dokonywana jest za pomocą lamp Nixie. Przedstawiono niezbędne zagadnienia teoretyczne potrzebne do zrozumienia etapu projektowania układu. Następnie przedstawiono koncepcję układu oraz podzielono go na moduły, które zostały opisane szczegółowo. Omówiono proces projektowania i wykonania obwodów drukowanych oraz montażu układu. Zaprezentowano oprogramowanie stworzone do obsługi zegara. Opisano proces uruchomiania i testowania układu, w którym potwierdzono spełnienie przez układ wszystkich wymagań projektowych. Sformułowano wnioski dotyczące kierunków dalszych prac.

Słowa kluczowe: Nixie, ESP32, Wi-Fi, Home Assistant, NTP

Dziedzina nauki i techniki, zgodne z wymogami OECD: nauki inżynierijno-techniczne: automatyka, elektronika, elektrotechnika i technologie kosmiczne

Abstract

The paper presents the process of designing a synchronized clock with an internet time server. This clock has an alarm clock function, works with the Home Assistant server, and time visualization is performed using Nixie tubes. Necessary theoretical issues are presented needed to understand the system design stage. Then the concept of the system was presented and it was divided into modules, which have been described in detail. The design and implementation process was discussed printed circuits and system assembly. The software was presented created to operate the clock. The process of starting and testing the system is described, which confirmed that the system meets all requirements design. Conclusions were formulated regarding directions for further work.

Keywords: Nixie, ESP32, Wi-Fi, Home Assistant, NTP

Spis treści

Wykaz ważniejszych oznaczeń i skrótów	8
1 Wstęp i cel pracy	9
1.1 Wstęp	9
1.2 Cel pracy	9
2 Lampy nixie	10
2.1 Zasada działania	10
2.2 Problemy związane z wykorzystaniem lamp nixie	11
2.3 Rozwiązania problemów związanych z lampami nixie	11
2.4 Sterowanie lampami	11
2.5 Sytuacja na rynku	12
3 Mikrokontroler ESP32-S3	13
3.1 Moduł RTC	13
3.2 GPIO	13
3.3 Kontroler USB/JTAG	14
3.4 Zasilanie	14
3.4.1 Środowisko programistyczne	14
4 Serwery czasu	16
4.1 Protokoły synchronizacji czasu	16
4.2 Struktura serwerów w protokole NTP	17
4.3 Zasada działania protokołu NTP	18
5 Koncepcja układu	20
5.1 Założenia projektowe	20
5.2 Ogólny schemat komunikacji z serwerem	21
6 Realizacja	22
6.1 Szczegółowa koncepcja układu	22
6.1.1 Sterowanie lampami nixie	22
6.1.2 Mikrokontroler	22
6.1.3 Źródło dźwięku	23
6.1.4 Pasek LED	23
6.1.5 Interfejs użytkownika	23
6.1.6 Zasilanie	23
6.1.7 Szczegółowy schemat projektu	24
6.2 Test lamp	25
6.3 Obliczenia mocy	27
7 Realizacja modułów	28
7.1 Sterowanie lampami Nixie	28
7.1.1 Dobór rejestrów	28
7.1.2 Sterowanie rejestrów	28
7.1.3 Sterowanie kropkami dziesiętnymi	29

7.1.4	Dobór rezystorów	29
7.2	Przetwornica 12V na wysokie napięcie	30
7.2.1	Założenia projektowe	30
7.2.2	Wybór układu scalonego	30
7.2.3	Dobór cewki	30
7.2.4	Dobór kondensatorów	32
7.2.5	Dobór diody	32
7.2.6	Dobór tranzystora	33
7.2.7	Ustawienie napięcia wyjściowego	33
7.2.8	Dobór rezystora ograniczającego prąd	34
7.3	Przetwornica 12V na 5V	35
7.3.1	Założenia projektowe	35
7.3.2	Wybór układu scalonego	35
7.3.3	Dobór komponentów	35
7.4	Złącze zasilania	37
7.4.1	Dobór złącza	37
7.4.2	Opis podłączenia	37
7.4.3	Zabezpieczenia ESD	37
7.5	Złącze do programowania	38
7.5.1	Dobór złącza	38
7.5.2	Opis podłączenia	38
7.5.3	Zabezpieczenia ESD	38
7.6	Buzzer	39
7.7	Enkoder obrotowy	41
7.8	LDO 5V na 3.3V	42
7.9	Złącze do uruchamiania układu	43
7.10	Złącze do paska LED	44
7.11	Mikrokontroler ESP32-S3	45
8	Projekt i montaż płytki drukowanej	46
8.1	Projekt płytki drukowanej	46
8.2	Montaż i uruchomienie układu	47
8.3	Uruchomienie układu	48
9	Oprogramowanie	50
9.1	Sterowanie lampami Nixie	50
9.2	Połączenie z siecią Wi-Fi	51
9.3	Pobranie czasu z serwera czasu	52
9.4	Obsługa enkodera rotacyjnego	53
9.5	Reguluja i odczyt napięcia	53
9.6	Odtwarzanie dźwięku	54
9.7	Komunikacja z serwerem Home Assistant	55
9.8	Główna logika programu	55
9.9	Interfejs użytkownika	57

10 Testowanie układu	58
10.1 Testy przetwornicy wysokiego napięcia	58
10.2 Testy poboru mocy	60
11 Podsumowanie	62
Bibliografia	64
Spis rysunków	65
Spis tabel	66

Wykaz ważniejszych oznaczeń i skrótów

NTP - Network Time Protocol, protokół czasu sieciowego

Wi-Fi - Wireless Fidelity, bezprzewodowa łączność

1 Wstęp i cel pracy

1.1 Wstęp

Potrzeba pomiaru czasu istnieje od początku istnienia ludzkiej cywilizacji. Ludzie od zawsze starali się mierzyć czas, aby zorganizować swoje życie w sposób bardziej efektywny.

Najprostsze i najstarsze metody pomiaru czasu to obserwacja zjawisk astronomicznych. Metody te, czyli np. obserwacja ruchu słońca, księżyca czy gwiazd, były stosowane przez tysiące lat, jednak była to metoda mało precyzyjna, która pchała wynalazców do poszukiwania nowych metod pomiaru czasu.

Pierwszym zegarem stworzonym przez człowieka był zegar słoneczny, który wskazywał czas, bazując na cieniu rzucanym przez słońce na tarczę zegara. Jednak metoda była zależna od długości dnia, pory roku i nie pozwalała na pomiar czasu nocą. Jako rozwiązanie zaczęto stosować urządzenia, w których upływający czas wyznaczał stały i ciągły przepływ substancji ciekłej lub sypkiej.

Wraz z postępem technologicznym zaczęto stosować zegary mechaniczne, oparte na mechanicznych oscylatorach, takich jak wahadło czy sprężyna. Jednak ich precyzja była ograniczona do skali sekundowej, a do tego pojawił się problem stabilności oscylatorów, które były podatne na zmiany temperatury i wilgotności. Pod koniec XIX wieku stabilność zegarów wahadłowych (ok. 0,01–0,001 sekundy na dobę) zaczynała graniczyć z tą, dla której wpływ zmian g związkanych z kształtem Ziemi miał już znaczenie [1]. Było to impusem do poszukiwania metod opartych o rozwiązania elektroniczne.

Powstały zegary kwarcowe, które miały właściwości niezależne od temperatury, były stabilne mechanicznie i chemicznie, jednak ich stabilność nie była wystarczająca. Zegar po miesiącu mógł się spóźniać o kilka sekund. Z powodu szybkiego rozwoju potrzebny był bardziej precyzyjny zegar o większej stabilności.

Rozwiązaniem okazał się zegar atomowy, który bazuje na zjawisku przejścia pomiędzy dwoma poziomami energetycznymi w atomie. Wraz z pojawiением się komputerów pojawiła się potrzeba ich synchronizacji, co doprowadziło do powstania serwerów czasu, które pozwalają na dokładną synchronizację czasu na całym świecie. Powstała również możliwość bezprzewodowej synchronizacji czasu. Mimo zaawansowania technologicznego cały czas występuje potrzeba opracowywania nowych rozwiązań.

1.2 Cel pracy

Celem pracy jest zaprojektowanie i wykonanie budzika opartego o lampy Nixie, który będzie synchronizowany z serwerem czasu. Mechanizm ten będzie wspierany modułem zegara czasu rzeczywistego wbudowanym w mikrokontroler. Czas pobrany z serwera będzie wyświetlany na lampach Nixie. Odtwarzanie alarmu będzie realizowane za pomocą źródła dźwięku umieszczonego na urządzeniu. Powinna być też możliwość wyłączenia alarmu za pomocą przycisku.

Urządzenie będzie wykorzystywać aplikację realizującą interfejs użytkownika, przy zachowaniu części ustawień bezpośrednio na urządzeniu. Aplikacja interfejsu użytkownika będzie umożliwiała ustawienie godziny alarmu i będzie on hostowana na zewnętrznym serwerze. Projekt zakłada stałe połączenie z siecią Wi-Fi.

2 Lampy nixie

Lampy nixie są to szklane lampy wyświetlające cyfry, litery lub inne symbole. Pojedyncza lampa składa się z katod w kształcie cyfr, liter lub innych symboli oraz anody w kształcie siatki otaczającą katody, wszystkie te elementy zamknięte są w szklanej bańce wypełnionej gazem szlachetnym. Wyświetlenie danej cyfry odbywa się poprzez wysterowanie odpowiedniej katody.

Nixie były używane w latach 50-70 XX wieku w różnych urządzeniach pomiarowych, licznikach, zegarach, itp. Obecnie nie są używane ze względu na konieczność zasilania wysokim napięciem, skomplikowanie układu sterującego i duży koszt produkcji. Mają one natomiast, ponadczasowe walory estetyczne, co jest powodem ich popularności pośród elektroników hobbytów.

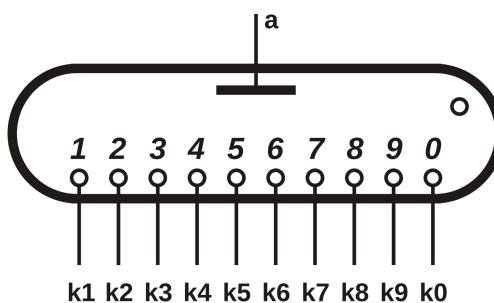
2.1 Zasada działania

Zjawisko zachodzące w lampach zwane jest jako wyładowanie gazowe [2]. Naładowane elektrycznie cząstki (elektrony), poprzez wysokie napięcie osiągają dużą energię kinetyczną. W momencie zderzenia z atomami gazu, elektrony w atomie gazu są wzbudzane do wyższych stanów energetycznych, a następnie wracają do stanu podstawowego emitując foton światła. Barwa światła zależy od gazu:

- jony neonu - czerwono-pomarańczowe,
- wodór - niebieskawo-fioletowy,
- azot - fiolet,
- krypton - biało-niebiesko.

Najczęściej stosowana jest mieszanka neonu i argonu pod małym ciśnieniem. Dodawana jest również rtęć, która ma za zadanie zwiększyć trwałość lampy, minimalizując tak zwane zatrucie katodowe. Efekt ten powoduje niepełne pokrycie katody warstwą gazową, co powoduje zanikanie wyświetlania cyfr, a czasami skutkuje zanikiem wyświetlania całkowicie.

Najczęściej spotykana konfiguracją lampy jest wspólna anoda przedstawiona na rysunku 2.1. Taki sposób podłączenia polega na wysterowaniu wybranej katody, podczas gdy anoda jest połączona do zasilania przez rezystor ograniczający prąd. Napięcie zasilania wynosi od 150 V do 200 V.



Rysunek 2.1: Schemat elektryczny lampy nixie ze wspólną anodą [3]

Gdzie:

- a - anoda
- k_n - katoda
- n - numer katody

2.2 Problemy związane z wykorzystaniem lamp nixie

Lampy są podatne na wiele problemów, które mogą wystąpić podczas użytkowania, takie jak:

- **Zatrucie katodowe** - zanikanie wyświetlania cyfr, spowodowane nie pełnym pokryciem katody warstwą gazową.
- **Zjawisko spalania** - zjawisko polegające na spalaniu się katod, spowodowane zbyt dużym prądem płynącym przez katodę.
- **Zjawisko cienia** - zjawisko polegające na wyświetlaniu niebieskiej poświaty, spowodowane zbyt dużym napięciem na anodzie.
- **Zjawisko zaniku** - zjawisko polegające na zanikaniu wyświetlania cyfr, spowodowane zbyt małym napięciem na anodzie.

Największym problemem jest zatrucie katodowe, które często pojawia się w zegarach wykorzystujących lampy nixie. Tylko parzyste wyświetlacze działają w odpowiednich warunkach, ponieważ używają wszystkich cyfr, a więc wszystkie cyfry zużywają się równomiernie. Problem pojawia się w przypadku nieparzystych katod, gdzie niektóre cyfry są używane znacznie rzadziej niż inne.

Na przykład, lampa po skrajnej lewej stronie wykorzystuje cyfry 0, 1, 2 do wyświetlania części dziesiątej godziny. Trzecia lampa (pozycja dziesiątek minut) używa cyfr 0, 1, 2, 3, 4, 5. Gdy konkretna cyfra nie jest używana przez długi czas (np. cyfra 8 na najbardziej po lewej stronie lampy), cyfra ta jest pokryta osadem metalu uwolnionym z innych aktywowanych cyfr. Te konkretne cyfry ostatecznie będą miały defekty w świeceniu, jeżeli nie zostaną użyte przez długi czas [4].

2.3 Rozwiązania problemów związanych z lampami nixie

Rozwiązaniem problemu zatrucia katodowego jest sterowanie lampami w taki sposób, aby wszystkie cyfry były używane. Można to osiągnąć przykładowo poprzez animacje, gdy zmienia się cyfra minut. Można również w godzinach nocnych przez jakiś czas wyświetlać cyklicznie wszystkie cyfry, aby zapobiec zatruciowi katodowemu.

W celu zwiększenia trwałości lamp można zmniejszyć prąd pracy lampy. Ogranicza to zjawisko spalania katod, natomiast prąd musi być wystarczająco duży aby lampa świeciła jasno.

2.4 Sterowanie lampami

Sterowanie lampami nixie jest trudne ze względu na konieczność zastosowania wysokiego napięcia. Przy szacowaniu potrzebnych wyprowadzeń mikrokontrolera i elementów założono użycie 6 lamp nixie, każda lampa ma 10 katod z cyframi od 0 do 9 oraz kropkę dziesiątną, co daje 11 sygnałów sterujących na lampa. Istnieje kilka sposobów sterowania lampami nixie [4]:

- Sterowanie bezpośrednie - każda lampa jest sterowana osobno, wykorzystując tranzystor wysokiego napięcia wysterowanego przez mikrokontroler. Wadą jest konieczność posiadania wielu pinów GPIO, co jest nieoptymalne. Sumarycznie wymaga to 66 pinów GPIO oraz 66

tranzystorów wysokiego napięcia. Można by w tym porządku zastosować multipleksery co zmniejszyłoby ilość wymaganych pinów GPIO do 16, ale zwiększyło skomplikowanie układu.

- Wykorzystanie dedykowanych driverów - istnieją specjalne układy scalone, które są przeznaczone do sterowania lampami nixie, niestety one również wymagają wielu pinów GPIO po 4 na każdą lampa, co daje 24 wymagane piny GPIO.
- Multiplexing - katody każdej lampy są podłączone do jednego drivera, który wybiera katodę i załącza odpowiednią anodę. Wymaga to mniej pinów GPIO bo tylko 10, ale multipleksacja powoduje szybsze zużycie lamp i pojawienie się artefaktów.
- Rejestr przesuwny wysokiego napięcia - rozwiązanie wymaga tylko 3 pinów GPIO. Rejestry wysokiego napięcia są ciężko dostępne i drogie. Rejestry muszą posiadać zatrzask. Wymagane jest również aby rejstry miały wyjścia typu Open Drain.
- Połączenie rejestrów przesuwnych z driverami - połączenie rejestrów przesuwnych wysokiego napięcia z dedykowanymi driverami, pozwala na zastosowania rejestrów przesuwnego na niskie napięcie. Ta kombinacja również wymaga 3 pinów GPIO, przy rejestrze 32 bitowym i 6 driverach. Powoduje jednak to większe skomplikowanie układu oraz wzrost objętości zajmowanego miejsca.

Podsumowanie sposobów sterowania lampami nixie przedstawia ??.

Nazwa	Ilość pinów	Cena	Trudność implementacji	Objętość
Sterowanie bezpośrednie	66	niska-średnia	niska-średnia	duża
Multiplexing	10	niska	wysoka	mała
Dedykowane drivery	24	średnia	niska	średnia
Rejestr przesuwny HV	3	wysoka	niska	średnia
Rejestr przesuwny + drivery	3	wysoka	średnia	duża

Tablica 2.1: Tabela opłacalności sposobów sterowania lampami nixie

2.5 Sytuacja na rynku

Obecnie dostępność lamp nixie jest ograniczona, gdyż nie są one już produkowane masowo. Istnieją małe firmy, które zajmują się produkcją, ale są to bardzo duże lampy w małych ilościach, co powoduje ich wysoką cenę. Dostępne są również lampy używane, ale w tym przypadku również duże lampy są drogie, a małe lampy są trudne do znalezienia. Problemem jest również niewiadomy stan lamp, ponieważ są to produkty w większości używane.

3 Mikrokontroler ESP32-S3

ESP32-S3 to mikrokontroler firmy Espressif Systems, który jest następcą popularnego ESP32. Jedenego z najpopularniejszych mikrokontrolerów z modulem WiFi wykorzystanego w wielu projektach IoT. Układ ten posiada następujące cechy odczytane z karty katalogowej [5]:

- 2 rdzenie Xtensa LX7 o taktowaniu 240 MHz
- 2,4 GHz WiFi 4 (802.11 b/g/n)
- Bluetooth 5.0 LE
- dwa 12-bitowe przetworniki ADC do 20 kanałów
- 14 pinów do obsługi dotykowego ekranu
- 45 programowalnych GPIO - część z nich ma specjalne funkcje
- kontroler USB/JTAG
- ROM: 384 KB
- SRAM: 512 KB
- Wbudowany moduł RTC

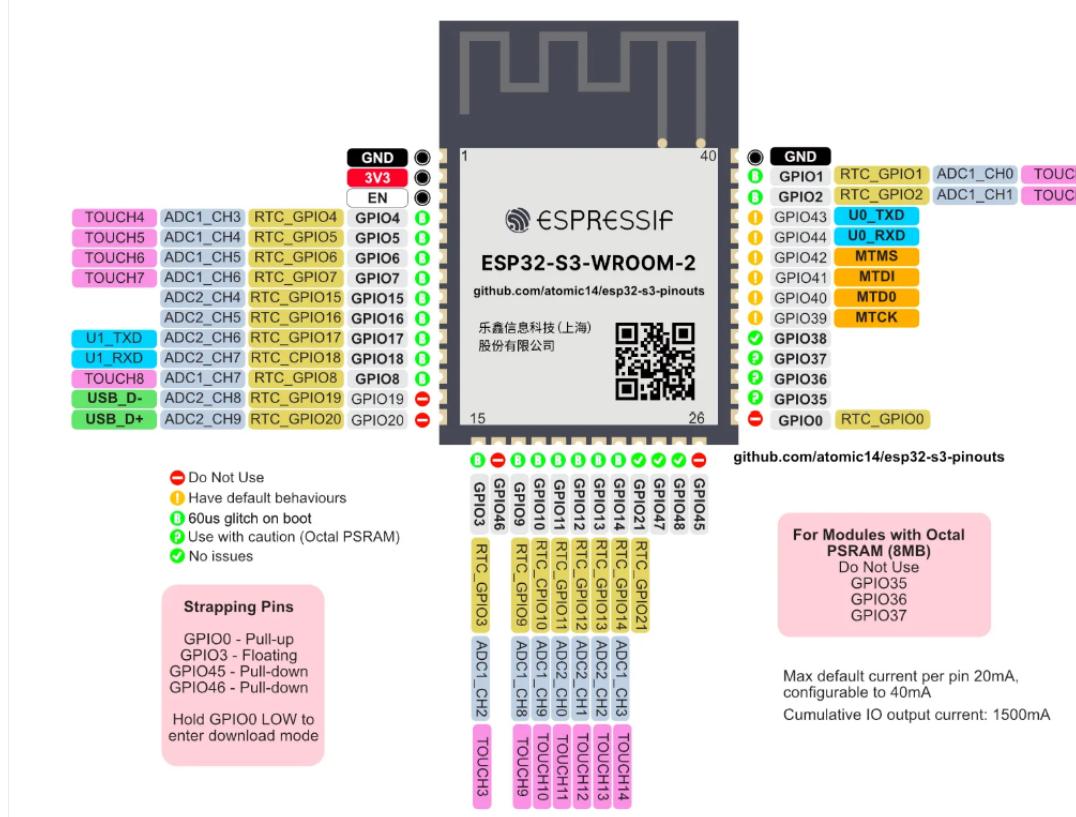
3.1 Moduł RTC

RTC (Real Time Clock) to moduł zegara czasu rzeczywistego, który pozwala na śledzenie aktualnego czasu, daty oraz dnia tygodnia. Moduł RTC w ESP32-S3 wykorzystuje 16 kB pamięci SRAM, co oznacza, że nie może on przechowywać daty i czasu w przypadku braku zasilania.

Sam moduł zegara czasu rzeczywistego nie jest bardzo dokładny; dlatego zaleca się synchronizację czasu z zewnętrznym serwerem. Istnieje wiele bibliotek do realizacji tego zadania z wykorzystaniem Arduino Framework.

3.2 GPIO

Układ posiada wiele wyprowadzeń wejścia/wyjścia ogólnego przeznaczenia (GPIO). Nie posiada on dedykowanych pinów do obsługi interfejsów takich jak I2C, można je skonfigurować na dowolnych pinach GPIO. Generacja sygnałów PWM jest również możliwa na większości wyprowadzeń. Posiada on również 20 pinów które mogą obsługiwać wejście analogowe. Rozkład pinów mikrokontrolera ESP32-S3 przedstawiono na 7.18.



Rysunek 3.1: Rozkład pinów mikrokontrolera ESP32-S3 [6]

3.3 Kontroler USB/JTAG

ESP32-S3 posiada wbudowany kontroler USB/JTAG, pozwalający programować układ bez użycia zewnętrznego programatora. Wyprowadzenia do programowania układu za pośrednictwem interfejsu USB to GPIO19(D-) oraz GPIO20(D+). Istnieje również możliwość programowania układu za pomocą protokołu UART korzystając z pinów GPIO1(TX) oraz GPIO3(RX).

3.4 Zasilanie

Według dokumentacji producenta, ESP32-S3 może być zasilany napięciem od 3 V do 3.6 V, zalecane napięcie zasilania to 3.3 V. Jego maksymalny pobór prądu wynosi 340 mA, jednak w praktyce jest on znacznie mniejszy, zależy to od wykorzystywanych funkcji i periferii.

Układ można wprowadzić w dwa tryby uśpienia:

- Light Sleep - pobór prądu wynosi około 240 μ A, w tym odłączany jest moduł WiFi a wszystkie piny GPIO są w stanie wysokiej impedancji.
- Deep Sleep - pobór prądu wynosi około 8 μ A, jedynie zasilany jest moduł zegara czasu rzeczywistego, wszystkie inne funkcje są wyłączone.

3.4.1 Środowisko programistyczne

ESP32-S3 jest na tyle rozbudowanym mikrokontrolerem, że wymaga pozwala on na programowanie w wielu językach programowania, takich jak C, C++, MicroPython, Lua oraz Rust. Najpopularniejszym językiem programowania dla ESP32-S3 jest C++, jest on bardzo wydajny, a przy tym

pozwala na wysokopoziomowe programowanie, co znacząco przyspiesza pisanie kodu oraz jego czytelność, również dla tego języka występuje najwięcej bibliotek i przykładów.

W przypadku ESP32-S3 najczęściej wykorzystuje się Arduino Framework, które jest najbardziej popularnym rozwiązaniem dla mikrokontrolerów ESP32. Można również korzystać z ESP-IDF, które jest oficjalnym środowiskiem programistycznym dla mikrokontrolerów rodziny ESP32, ale jest ono bardziej skomplikowane i wymaga więcej czasu na naukę. Dwoma najpopularniejszymi środowiskami programistycznymi dla ESP32-S3 są PlatformIO oraz Arduino IDE.

4 Serwery czasu

Serwerem czasu nazywamy serwer komputerowy, pobierający czas z zewnętrznych źródeł i udostępniający go dla innych urządzeń w sieci. Udostępniane są bardzo precyzyjne dane czasowe; dokładność zależy od źródła czasu, z którego serwer korzysta. Serwer czasu może być używany jako lokalny lub internetowy.

Serwery wykorzystują różne źródła zewnętrzne do synchronizacji czasu, takie jak:

- zegary atomowe,
- odbiorniki czasu GNSS (Global Navigation Satellite System),
- oscylatory rubidowe,
- oscylatory cezowe,
- zegary wodorowe

Są to zegary o bardzo dużej precyzyji, rzędu nanosekund, co pozwala na synchronizację czasu w sieciach komputerowych, telekomunikacyjnych itp.

4.1 Protokoły synchronizacji czasu

Serwery te wykorzystują różne protokoły sieciowe do synchronizacji czasu, takie jak:

- NTP (Network Time Protocol) - wysyła okresowo pakiet zawierający aktualne opóźnienie w odniesieniu do czasu UTC (Universal Time Coordinated), na podstawie których klient kalibruje swoje zegary. Jest to najpopularniejszy protokół synchronizacji czasu w sieciach komputerowych, jest on wspierany przez większość systemów operacyjnych.
- PTP (Precision Time Protocol) - jest bardziej precyzyjną alternatywą NTP i jest używany w systemach o wysokiej precyzyji. Najczęściej stosowany w sieciach przemysłowych oraz przy badaniach naukowych. Jest w stanie osiągnąć dokładność synchronizacji zegarów poniżej mikrosekundy.
- Algorytm Berkeley - to algorytm synchronizacji czasu opracowany na Uniwersytecie Kalifornijskim w Berkeley. Jego działanie polega na pomiarze szybkości dryfowania zegara między serwerami, często jest łączony z protokołem NTP.
- GPS (Global Positioning System) - wykorzystuje odbiorniki GPS do synchronizacji zegarów na różnych serwerach. Zapewnia bardzo dokładne dane o aktualnym czasie. Czas ten można wykorzystać do synchronizacji zegarów serwerów podłączonych do tego samego odbiornika GPS.

Każdy z tych protokołów ma swoje następujące wady i zalety:

- NTP - Główną zaletą jest niezawodność i dokładność, co sprawia, że nadaje się do szerokiego zakresu zastosowań. Jednak NTP nie jest tak dokładny jak PTP i może synchronizować zegary z dokładnością do kilku milisekund. W związku z tym, że jest to leciwy protokół, nie jest najbardziej bezpiecznym rozwiązaniem, może być podatny na niektóre rodzaje ataków, takie jak ataki typu man-in-the-middle. Protokół istnieje już bardzo długo, więc dobrze znany i istnieje wiele rozwiązań ułatwiających implementację.
- PTP - porównując do NTP, jest bardziej precyzyjny i może synchronizować zegary z dokładnością do kilku mikrosekund. Jednak ma zdecydowanie większe wymagania sprzętowe (specjalistyczny sprzęt) i konfiguracyjne, co sprawia, że jest bardziej skomplikowany w użyciu.
- Algorytm Berkeley - może być używany w połączeniu z NTP. Jedeną z głównych zalet tego algorytmu jest to, że może synchronizować zegary z dokładnością do kilku mikrosekund, dzięki czemu nadaje się do wielu zastosowań. Podobnie jak w PTP wymaga on specjalistycznego sprzętu, co sprawia, że jest bardziej skomplikowany w użyciu i droższy.
- GPS - najbardziej precyzyjny z wymienionych protokołów, może synchronizować zegary z dokładnością do kilku nanosekund. Jest jednak niezalecany do zastosowań wewnętrznych pomieszczeń, ze względu na konieczność widoczności satelitów GPS i wymaga odbiornika GPS.

Z wyżej wymienionych protokołów, NTP jest najczęściej stosowany w sieciach komputerowych, dlatego też wydaje się być najlepszym wyborem do synchronizacji zegara nixie. Alternatywnym rozwiązaniem może być wykorzystanie własnego serwera który by zwracał czas wykorzystując REST API, ale wymaga to posiadania własnego serwera i jest zależne od jego działania.

4.2 Struktura serwerów w protokole NTP

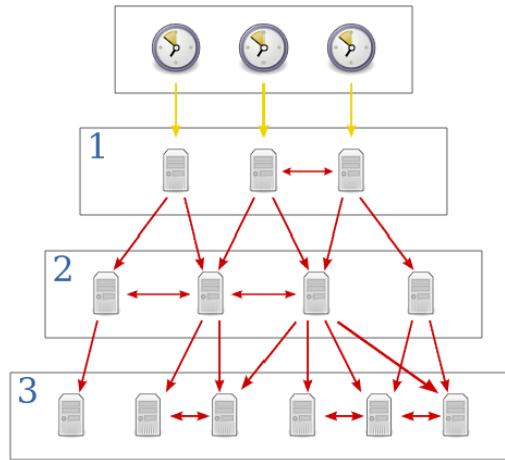
Synchronizacji NTP wykorzystuje uporządkowaną strukturę gałęziową STRATUM [7]. Zasada hierarchii wygląda następująco: urządzenia warstwy STRATUM N mogą być serwerami czasu dla warstwy STRATUM N+1, ale nie na odwrót. Komputery STRATUM N mogą być również klientami urządzeń warstwy STRATUM N-1 itd.

Struktura ta ma na celu uporządkowanie i wprowadzenie hierarchii priorytetów urządzeń, zgodnie z ich rzeczywistym przeznaczeniem i funkcją. Aby nie powodować nadmiernego skomplikowania systemu i związanych z tym opóźnień, liczba warstw została ograniczona do 16(STRATUM 0 - STRATUM 15).

Niektóre warstwy mają specjalne właściwości. Warstwa STRATUM 0 służy wyłącznie dla wzorców czasu, czyli zegarów atomowych, satelitarnych, itp. będących faktycznym źródłem czasu. Połączenie ze źródłem nie jest sieciowe, a zazwyczaj odbywa się używając specjalnych interfejsów sprzętowych.

STRATUM 1 oraz STRATUM 2 stanowią najwyższe warstwy NTP i powinny być wykorzystane w przypadku dużych serwerów wysokiej jakości, superkomputerów lub sprzętowych serwerów czasu. Pozostałe warstwy są przeznaczone dla urządzeń lokalnych, takich jak komputery, routery, itp.

Numer STRATUM mówi jak daleko od wzorca czasu znajduje się dany serwer. Im niższy numer, tym bliżej źródła czasu. W rozbudowanych sieciach poziom STRATUM nie ma znaczącego wpływu na jakość synchronizacji i precyzję uzyskiwanego czasu. Strukturę serwerów czasu w protokole NTP przedstawiono na rysunku 4.1.



Rysunek 4.1: Struktura serwerów czasu w protokole NTP [8]

W przypadku zegara nixie poziom STRATUM nie ma większego znaczenia, ponieważ zegar nie wymaga bardzo precyzyjnego synchronizowania czasu, oczywiście zależy to od dokładności z jaką czas będzie wyświetlany, ale w przypadku zegara na 6 lampach, różnica w czasie rzędu kilku milisekund nie będzieauważalna.

4.3 Zasada działania protokołu NTP

NTP różni się od typowego protokołu komunikacyjnego. Nie transmisuje on bowiem absolutnej wartości czasu, lecz przekazuje informacje o opóźnieniach i korelacjach czasowych w regularnych odstępach czasu, jakie zachodzą w sieci TCP/IP. Protokół zyskuje większą dokładność dopiero przy stosowaniu wielu źródeł czasu jednocześnie, wykorzystuje od wtedy algorytm analizy statystycznej czasu oparty na metodzie DTS (Dynamic Time Scales).

NTP wykorzystuje pakiety UDP o długości 72 bajtów na porcie 123, które są okresowo wymieniane co 2^τ sekund, gdzie τ wynosi od 4 (16s) do 17 (36h). Pozwala to klientom serwera, wyliczać opóźnienie względem idealnego czasu UTC. Znając aktualne opóźnienie w odniesieniu do czasu UTC, klient NTP sam kalibruje swój zegar lokalny, która polega na płynnym przyspieszaniu lub spowalnianiu pracy lokalnego zegara programowego. Przy różnicach czasu przekraczających 128ms, stosowana jest metoda step, która polega na skokowym przesunięciu zegara o określoną wartość. Dzięki temu każdy z klientów, asymptotycznie zmierza do czasu pochodzącego z wzorcowego zegara czasu UTC. Pakiet NTP ma strukturę przedstawioną w 4.3.

LI	VN	Mode	Stratum	Poll	Precision
Root Delay					
Root Dispersion					
Reference Identifier					
Reference Timestamp					
Originate Timestamp					
Receive Timestamp					
Transmit Timestamp					
Authenticator					

Tablica 4.2: NTP – format komunikatu [8]

- LI – wskaźnik sekund przestępnych
- VN – (Version Number) numer wersji protokołu
- Mode – tryb pracy
- Stratum – warstwa, w której funkcjonuje komputer będący nadawcą komunikatu
- Poll interval – okres pomiędzy kolejnymi aktualizacjami czasu
- Precision – określenie dokładności zegara komputera wysyłającego dany komunikat
- Root Delay – opóźnienie pomiędzy nadawcą a serwerem warstwy 1
- Root Dispersion – maksymalny błąd pomiędzy zegarem lokalnym a serwera warstwy 1
- Reference Identifier – identyfikator źródła czasu, względem którego następuje synchronizacja
- Reference Timestamp – pole zawierające pomocnicze informacje o czasie poprzedniej synchronizacji
- Originate Timestamp – pole zawierające czas wysłania żądania przez klienta
- Receive Timestamp – czas odebrania komunikatu od klienta
- Transmit Timestamp – czas wysłania odpowiedzi do klienta
- Authenticator – informacje uwierzytelniające zarówno klienta, jak i serwer czasu
- Root Dispersion – maksymalny błąd pomiędzy zegarem lokalnym a serwera warstwy 1
- Reference Identifier – identyfikator źródła czasu, względem którego następuje synchronizacja
- Reference Timestamp – pole zawierające pomocnicze informacje o czasie poprzedniej synchronizacji
- Originate Timestamp – pole zawierające czas wysłania żądania przez klienta
- Receive Timestamp – czas odebrania komunikatu od klienta
- Transmit Timestamp – czas wysłania odpowiedzi do klienta
- Authenticator – informacje uwierzytelniające zarówno klienta, jak i serwer czasu

5 Koncepcja układu

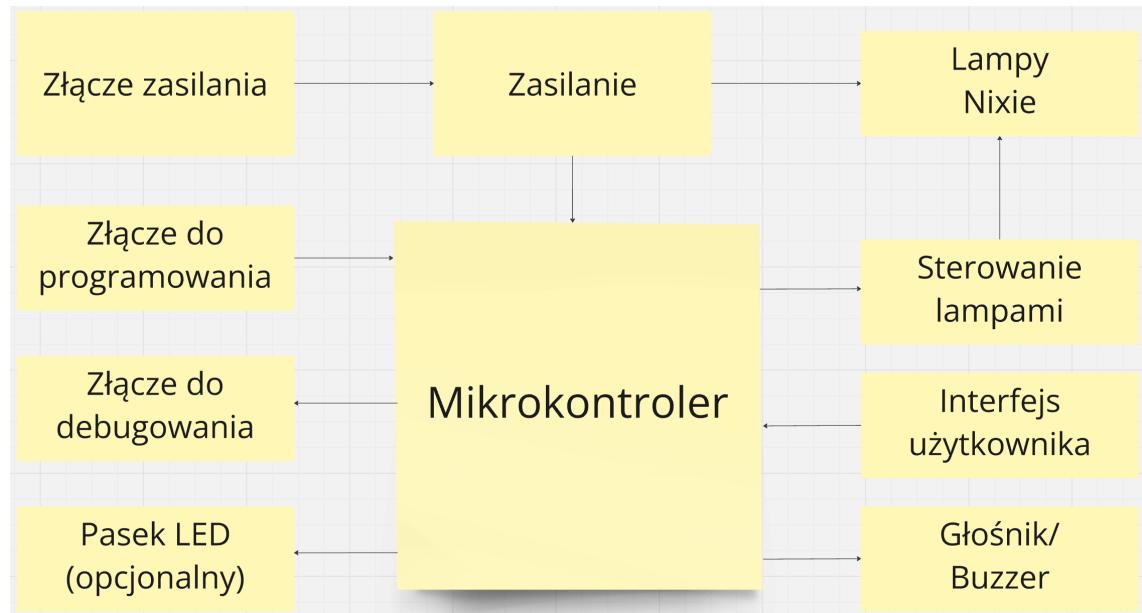
Proces koncepcyjny został podzielony na dwa etapy: koncepcję układu oraz realizację. W tym rozdziale znajduje się ogólna koncepcja działania układu, która nie jest związana z konkretnymi elementami sprzętowymi, a jedynie z funkcjonalnościami, które mają być zrealizowane.

5.1 Założenia projektowe

Zgodnie z celem pracy, określono następujące założenia projektowe:

- Funkcjonalność ustawiania godziny budzika będzie realizowana przez aplikację hostowaną na zewnętrznym serwerze.
- Na wyświetlaczu Nixie będą wyświetlane godziny, minuty, sekundy.
- Alarm będzie sygnalizowany dźwiękiem oraz animacją.
- Wyłączanie alarmu będzie możliwe poprzez przycisk na obudowie.
- Możliwość regulacji jasności wyświetlacza Nixie.
- Dodatkową funkcjonalnością jest możliwość podpięcia paska LED, od dołu budzika dla dodatkowego efektu wizualnego.

Powyższe założenia powodują podzielenie projektu na poszczególne moduły realizujące poszczególne funkcje, które będą opisane w dalszej części pracy. Ogólna koncepcja układu przedstawiono na rysunku 5.1.



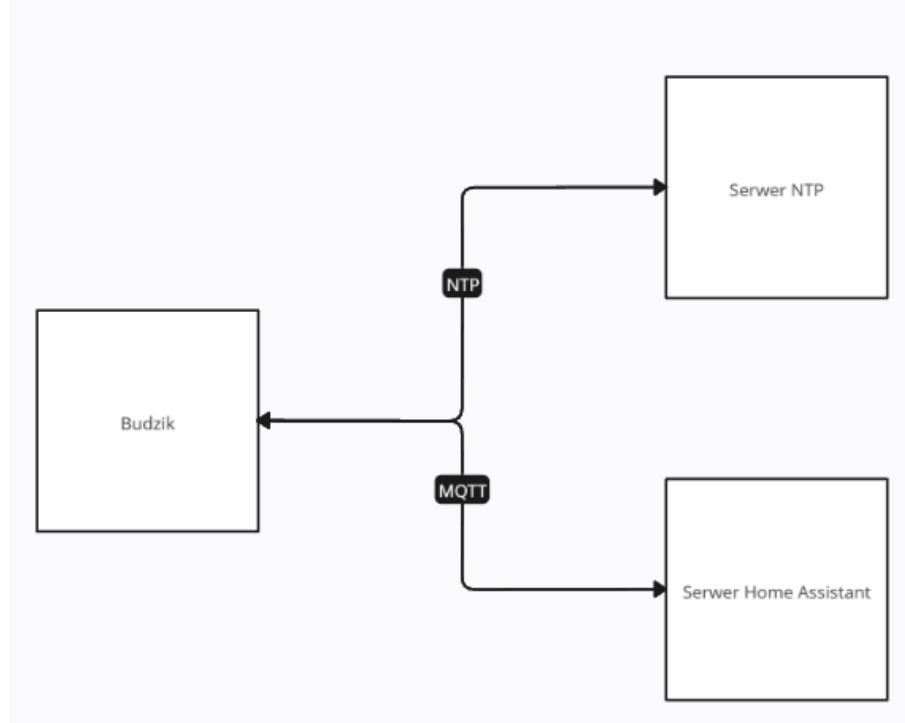
Rysunek 5.1: Ogólna koncepcja układu

Sekcja opisana jako *zasilanie* będzie odpowiedzialna za zasilanie wszystkich elementów układu, w tym lamp Nixie, paska LED, mikrokontrolera oraz głośnika, więc będzie wymagane podzielenie jej na kilka podsekcji, ponieważ będą potrzebne różne napięcia zasilania. Lampy Nixie potrzebują zasilania wysokim napięciem, natomiast pozostałe komponenty układu potrzebują zasilania

niższym napięciem. Blok *sterowanie lampami Nixie* będzie odpowiedzialny za wyświetlanie odpowiednich cyfr na lampach. Sekcja *Interfejs użytkownika* będzie odpowiedzialna za interakcję z użytkownikiem, w tym regulacja jasności oraz wyłączania alarmu, ważne by interfejs był intuicyjny i jak najbardziej rozwojowy na potencjalne przyszłe funkcje.

5.2 Ogólny schemat komunikacji z serwerem

Urządzenie będzie musiało komunikować się z serwerem czasu, a także z serwerem Home Assistant, który będzie realizował interfejs użytkownika. Komunikacja z serwerem czasu będzie odbywała się poprzez protokół NTP, ponieważ inne protokoły opisane w rozdziale 4 służą do zapewnienia większej dokładności czasu, co nie jest wymagane w tym projekcie. Inne protokoły wymagają też większej ilości zasobów lub specjalistycznego sprzętu. Kolejną zaletą wyboru NTP jest to, że jest to najbardziej popularny protokół do synchronizacji czasu w sieciach komputerowych, co sprawia, że jest on przetestowany i stabilny. Posiada on wiele implementacji, które są dostępne na wielu platformach, w tym na platformę ESP32. Komunikacja z serwerem Home Assistant będzie odbywała się poprzez protokół MQTT, który jest popularnym protokołem w IoT, co pozwoli na łatwe rozbudowanie funkcjonalności. Strukturę połączeń przedstawiono na rysunku 5.2.



Rysunek 5.2: Ogólny schemat komunikacji z serwerem

6 Realizacja

W tym rozdziale opisano szczegółową koncepcję układu, która została opracowana na podstawie analizy przeprowadzonej w rozdziałach 2 i 3 oraz po zapoznaniu się z ofertą sklepów elektronicznych. Zostały określone konkretne rozwiązania projektowe, które będą wykorzystane, oraz wykonano niezbędne obliczenia, które pozwoliły na wybór odpowiednich komponentów na dalszym etapie projektowania.

6.1 Szczegółowa koncepcja układu

Szczegółowa koncepcja układu została podzielenia na poszczególne sekcje układu.

6.1.1 Sterowanie lampami nixie

Kluczowym jest wybór sterownia lampami nixie, ponieważ na podstawie tego wyboru zostanie zaprojektowany pozostała część układu. Zgodnie z analizą przeprowadzoną w podrozdziale 2.4, zdecydowano się na sterowanie lampami za pomocą rejestrów przesuwnych wysokiego napięcia. Zastosowanie tego rozwiązania pozwala na zredukowanie ilości potrzebnych wyprowadzeń mikrokontrolera do sterowania lampami. Ten typ sterowania jest łatwy w implementacji, wymaga jedynie wgrania odpowiednich danych do rejestrów przesuwnych, a następnie zatrzaśnięcie wyjścia, co pozwala na wyświetlenie odpowiedniej cyfry na lampie.

Niezależnie od wyboru lamp każda ma 10 katod z cyframi i jedną katode od kropki dziesiątej, więc wymagane jest 11 wyjść na każdą lampa. Dostępne w sprzedaży są rejstry 32 bitowe, co pozwala na sterowanie 3 lampami nixie bez kropek i jedną neonówką która będzie służyć jako separator między godzinami a minutami oraz między minutami a sekundami. Do sterownia kropkami dziesiętnymi zostaną użyte tranzystory wysokiego napięcia podłączone do wyjść mikrokontrolera, ponieważ nie opłacalnym jest dodawanie kolejnego rejestru przesuwnego wysokiego napięcia tylko do sterowania kropkami dziesiętnymi.

Wynika z tego, że potrzebne są 2 rejstry przesuwne wysokiego napięcia do sterownia lamp i neonówkami oraz 6 tranzystorów wysokiego napięcia do sterowania kropkami dziesiętnymi. Do sterowania rejestrami prawdopodobnie będzie potrzebny konwerter poziomów logicznych, ponieważ mikrokontroler ESP32-S3 zasilany jest napięciem 3.3 V, a rejstry prawdopodobnie będą operować na wyższym napięciu.

6.1.2 Mikrokontroler

Wybór sposobu sterowania lampami Nixie wpłynął na wybór mikrokontrolera, ponieważ musi on posiadać odpowiednią ilość pinów GPIO oraz musi być w stanie generować sygnał zegarowy. Potrzebne jest 9 pinów GPIO do pełnego sterowania lampami oraz kropkami dziesiętnymi, do tego trzeba pamiętać o zapasie pinów na pozostałe funkcje. W związku z tym wybrano mikrokontroler ESP32-S3, który posiada 45 programowalnych GPIO, co pozwala na swobodne zaprojektowanie reszty układu. Ma też on dużą zaletę w postaci kontrolera USB/JTAG, dzięki czemu nie jest potrzebny dodatkowy programator do programowania układu. Jest to też popularny mikrokontroler, dla którego istnieje wiele bibliotek i przykładów.

6.1.3 Źródło dźwięku

Jako źródło dźwięku wybrano głośnik piezoelektryczny, który jest prostym elementem i nie wymaga dodatkowego wzmacniacza. Jego zaletami są niski pobór prądu, małe rozmiary i niska cena. Sterowanie odbywa się za pomocą sygnału PWM, który pozwala generować proste melodie. Głośnik piezoelektryczny jest wystarczająco głośny, aby być słyszalnym w pomieszczeniu, w którym będzie znajdował się budzik.

6.1.4 Pasek LED

Pasek LED będzie służył jako dodatkowe źródło światła, które będzie sygnalizować alarm, oraz jako element estetyczny. Pasek ten musi zawierać w sobie adresowane diody LED, które pozwolą na wyświetlanie różnych kolorów (RGB). Rozwiążanie to jest proste w implementacji, wystarczy podłączyć go do pinu GPIO mikrokontrolera i za pomocą sygnału PWM można sterować jasnością.

6.1.5 Interfejs użytkownika

Interfejs użytkownika będzie składał się z enkodera obrotowego z przyciskiem, który będzie służył do regulacji jasności, a przycisk do wyłączania alarmu. Enkoder obrotowy jest też na tyle uniwersalnym rozwiązaniem, które pozwala na mnogość kombinacji sterowania, ale wygodniejsze jest korzystanie z aplikacji mobilnej.

6.1.6 Zasilanie

Kluczowe jest zaprojektowanie przetwornicy wysokiego napięcia do zasilania lamp Nixie, ponieważ jest to najbardziej wymagający element układu.

Możliwe są dwa rozwiązania:

- Przetwornica typu flyback
- Przetwornica typu boost

Przetwornica typu flyback ma zaletę w postaci izolacji galwanicznej między wejściem a wyjściem oraz umożliwia zaprojektowanie przetwornicy z napięciem zasilania 5 V. Wadą jest to, że jest potrzebny transformator, który jest drogi i trudno dostępny, do tego jest to bardziej skomplikowane rozwiązanie na etapie projektowania.

Ze względu na duży problem ze znalezieniem transformatora, zdecydowano się na przetwornicę typu boost, która jest prostsza w implementacji i tańsza. Wadą jest to, że nie ma izolacji galwanicznej między wejściem a wyjściem, ale w tym przypadku nie jest to wymagane. Następnym krokiem było wybranie jednej z dwóch konfiguracji układu:

- USB-C jako zasilanie zewnętrzne, przetwornica typu boost z zasilacza 5 V na 12 V i przetwornica typu boost z zasilacza 12 V na wysokie napięcie
- Złącze DC jako zasilanie zewnętrzne, przetwornica typu boost z zasilacza 12 V na wysokie napięcie oraz przetwornica typu buck z zasilacza 12 V na 5 V

Wadą pierwszej konfiguracji jest to, że niemożliwym będzie jednoczesne programowanie i zasilanie układu, ponieważ złącze USB-C podłączone do komputera ma niską wydajność prądową. Dla USB 2.0 wynosi ona 500mA, a dla USB 3.0 wynosi 900mA, co jest niewystarczające dla zasilania

układu. Brak możliwości jednoczesnego zasilania i programowania było by dużym problemem na etapie pisania oprogramowania, więc zdecydowano się na drugą konfigurację.

Wybrano więc przetwornicę typu boost, która będzie zasilana z zasilacza 12 V, a wyjście będzie podłączone do anod lamp nixie. Do tego będzie potrzebne zasilanie 5 V dla paska LED oraz 3.3 V dla mikrokontrolera. Moduł zasilania 5 V będzie zasilał pasek LED, który potrafi pobrać większy prąd, to ze względu na zachowanie wysokiej efektywności, zdecydowano się na przetwornicę typu buck. Moduł zasilania 3.3 V będzie zasilaniem mikrokontrolera, więc wystarczy zastosować stabilizator liniowy.

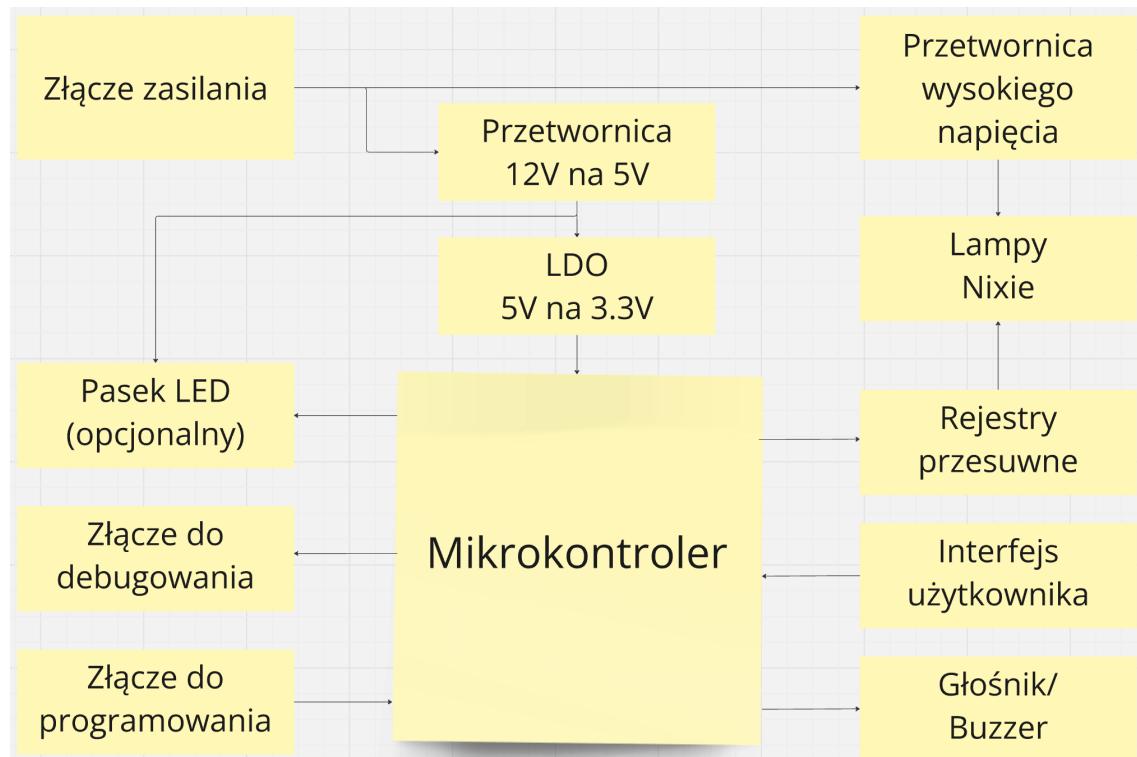
Można, więc podzielić zasilanie na 3 pod moduły:

- Przetwornica typu boost z zasilacza 12 V na wysokie napięcie
- Przetwornica typu buck z zasilacza 12 V na 5 V
- Stabilizator liniowy z zasilacza 5 V na 3.3 V

Wynika z tego, że urządzenie będzie posiadać 3 złącza:

- Złącze USB-C do programowania mikrokontrolera
- Złącze DC do zasilania układu
- Złącze typu goldpin jako złącze do komunikacji szeregowej wykorzystywane w procesie uruchamiania układu

6.1.7 Szczegółowy schemat projektu



Rysunek 6.1: Schemat blokowy projektu

6.2 Test lamp

Pierwszą rzeczą, która została wykonana to zakup lamp Nixie, które będą wykorzystane w projekcie. Przez coraz mniejszą dostępność lamp Nixie, zdecydowano się na zakup małych lamp Z570M, które zostały zakupione w ilości 6 sztuk. Lampy te mają 10 cyfr oraz kropkę dziesiątną. Lampy są używane, ale wszystkie lampy zostały sprawdzone i działają poprawnie. Kluczowe parametry zastosowanych lamp nixie odczytane z karty katalogowej[9] to:

- Napięcie zapłonu: 170 V
- Napięcie wygaszania: 120 V
- Napięcie pracy: 150 V
- Prąd katodowy średni: 2 mA

W celu zweryfikowania działania lamp nixie i sprawdzenia parametrów zasilania, został wykonyany prototyp układu z jedną lampą nixie, wykorzystujący zasilacz impulsowy wysokiego napięcia z regulowanym napięciem wyjściowym zakupiony w sklepie internetowym.

Zakupiona przetwornica wysokiego napięcia ma następujące parametry:

- Napięcie wejściowe: 5 – 12 V
- Napięcie wyjściowe: 150 – 220 V
- Prąd wyjściowy: 20 mA

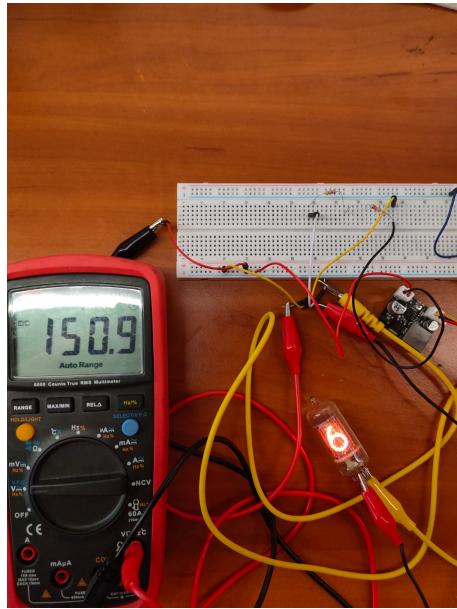
W celu sprawdzenia działania lampy należy najpierw dobrać rezystor ograniczający prąd katodowy. Zakładając, że napięcie zasilania wynosi maksymalnie $U_{\max} = 220$ V, a napięciu pracy lampy $U_{\text{pr}} = 150$ V, przy prądzie katodowym $I_{\text{kat}} = 2$ mA, rezystor ograniczający prąd katodowy można obliczyć ze wzoru:

$$R = \frac{U_{\max} - U_{\text{pr}}}{I_{\text{kat}}} = \frac{220 \text{ V} - 150 \text{ V}}{2 \text{ mA}} = 35 \text{ k}\Omega \quad (6.1)$$

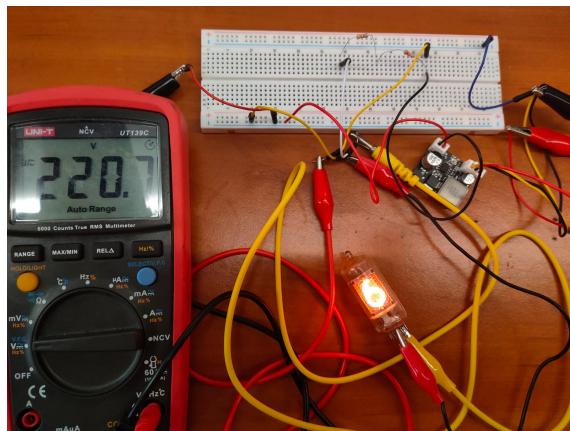
W nocy katalogowej lampy nixie Z570M producent podaje, że zalecany rezystor ograniczający prąd katodowy powinien mieć wartość $33 \text{ k}\Omega$ dla napięcie zasilania 200 V, więc wartość $35 \text{ k}\Omega$ dla napięcia 220 V wydaje się obliczona prawidłowo, taki też rezystor ma zostać użyty w faktycznym układzie.

Zatem rezystor ograniczający prąd katodowy powinien mieć wartość około $35 \text{ k}\Omega$. Do testu użyto rezystora o wartości $22 \text{ k}\Omega$ oraz rezystora o wartości $10 \text{ k}\Omega$, połączonych szeregowo, co daje wartość $32 \text{ k}\Omega$, co jest wartością zbliżoną do obliczonej.

Z testów wynika, że lampy Nixie działają poprawnie, a dobrany rezystor ograniczający prąd katodowy jest odpowiedni. Przy napięciu zasilania 150 V lampa świeci słabiej, ale jest to zgodne z oczekiwaniami. Natomiast przy napięciu 220 V lampa świeci jasno i pojawiają się lekko niebieskie refleksy wewnętrz lampy, co jest zgodne z oczekiwaniami.



Rysunek 6.2: Prototyp układu z lampą Nixie przy napięciu zasilania 150 V



Rysunek 6.3: Prototyp układu z lampą Nixie przy napięciu zasilania 220 V

Nie sprawdzono napięcia wygaszania, ponieważ zakres pracy zasilacza okazał się za mały na takie napięcie. Ustalono jednak, że lampa nawet przy napięciu 150 V była w stanie zaplonić i świecić poprawnie.

Z testów można wyciągnąć następujące wnioski:

- Lampy Nixie działają poprawnie przy napięciu zasilania 150 V oraz 220 V.
- Dobrany rezystor ograniczający prąd katodowy jest odpowiedni.
- Lampa Nixie Z570M jest w stanie zaplonić i świecić przy napięciu wygaszania 150 V.
- Zakres regulacji napięcia na zasilaczku wysokiego napięcia powinien być większy, np. 130 – 250 V, by lampa mogła być jeszcze słabiej podświetlona. Może się to okazać przydatne w nocy.

6.3 Obliczenia mocy

By móc zaprojektować odpowiednie zasilanie dla całego układu, należy obliczyć szacunkową moc potrzebną do zasilania wszystkich komponentów. Poza lampami Nixie, najbardziej obciążającym elementem będzie pasek LED oraz mikrokontroler; pozostałe elementy będą pobierały znikome ilości prądu.

Założono maksymalną długość paska LED na 30 cm. Z deklaracji producenta paska LED wynika, że moc na metr wynosi 18 W, co daje:

$$P_{\text{LED}} = 18 \text{ W m}^{-1} \cdot 0.3 \text{ m} = 5.4 \text{ W} \quad (6.2)$$

Następnie obliczono prąd potrzebny do zasilenia paska LED przy napięciu 5 V:

$$I_{\text{LED}} = \frac{P_{\text{LED}}}{U_{\text{LED}}} = \frac{5.4 \text{ W}}{5 \text{ V}} = 1.08 \text{ A} \quad (6.3)$$

Następnie obliczono moc potrzebną do zasilania mikrokontrolera ESP32-S3. Według producenta maksymalny pobór prądu wynosi 340 mA, co przy napięciu zasilania 3.3 V daje:

$$P_{\text{ESP32}} = 340 \text{ mA} \cdot 3.3 \text{ V} = 1.122 \text{ W} \quad (6.4)$$

Następnie policzono prąd pobierany przez wszystkie lampy, których jest 6 sztuk, przy prądzie katodowym 2 mA każda, co daje:

$$I_{\text{Nixie}} = 6 \cdot 2 \text{ mA} = 12 \text{ mA} \quad (6.5)$$

Jednak należy dodać jeszcze prąd potrzebny do zasilania kropek oraz separatorów, oszacowano prąd kropki na 1 mA oraz separatora na 1 mA, co daje:

$$I_{\text{Nixie}} = 6 \cdot 2 \text{ mA} + 6 \cdot 1 \text{ mA} + 2 \cdot 1 \text{ mA} = 20 \text{ mA} \quad (6.6)$$

Następnie obliczono moc potrzebną do zasilania lampy nixie, przy napięciu 220 V oraz prądzie 20 mA, zakładając sprawność przetwornicy na poziomie 70 %:

$$P_{\text{Nixie}} = \frac{U_{\text{Nixie}} \cdot I_{\text{Nixie}}}{\text{Sprawność}} = \frac{220 \text{ V} \cdot 20 \text{ mA}}{0.7} = 6.29 \text{ W} \quad (6.7)$$

Pozostałe komponenty będą pobierały znikome ilości prądu, więc nie będą brane pod uwagę w obliczeniach. Szacunkowa moc potrzebna do zasilania całego układu wynosi:

$$P_{\text{całkowita}} = P_{\text{LED}} + P_{\text{ESP32}} + P_{\text{Nixie}} = 5.4 \text{ W} + 1.122 \text{ W} + 6.29 \text{ W} = 12.812 \text{ W} \quad (6.8)$$

Szacunkowa moc potrzebna do zasilania całego układu wynosi około 13 W,

7 Realizacja modułów

Po określaniu szczegółowej koncepcji układu przystąpiono do realizacji poszczególnych modułów, które zostały opisane w rozdziale 6. Każdy z modułów ma opisany dobór komponentów do realizacji, funkcjonalności modułu oraz sposób podłączenia.

7.1 Sterowanie lampami Nixie

Moduł jest odpowiedzialny za sterowanie wyświetlaniem cyfr na lampach Nixie oraz załączanie lamp neonowych. Sterowanie lampami i neonówkami odbywa się za pomocą rejestrów przesuwnych, które są sterowane przez mikrokontroler. Natomiast kropki na lampach Nixie są sterowane za pośrednictwem tranzystorów.

7.1.1 Dobór rejestrów

Wybrano rejestyre przesuwne HV firmy microchip o numerze HV5530, o następujących parametrach [10]:

- Rejestr 32 bitowy
- Maksymalne napięcie na wyjściu - 315 V
- Maksymalna częstotliwość pracy - 8 MHz
- napięcie zasilania - od 10.8 V do 13.6 V
- stan wysoki - $V_{cc} - 2$ V gdzie V_{cc} to napięcie zasilania

7.1.2 Sterowanie rejestrów

Sterowanie rejestrem realizowane jest za pomocą następujących pinów:

- CLK - sterowanie zegarem rejestrów
- LE - załadowanie danych do rejestrów(Latch Enable)
- POL - ustawienie polaryzacji wyjścia
- DATA_IN - wejście danych
- BL - wyjście blanking(ustawianie wszystkich wyjść na zadany stan logiczny)
- DATA_OUT - wyjście danych dla następnego rejestrów

Do sterowania wystarczą jedynie 3 linie CLK, LE, DATA_IN, ponieważ BL i POL można ustawić na stałe. Rejestry można połączyć ze sobą dzięki czemu wymagana jest tylko jedna linia danych. Sterowanie wymaga użycia konwertera poziomów logicznych, ponieważ mikrokontroler pracuje na napięciu 3.3 V, a rejestr operuje na napięciu około 12 V.

Zastosowano konwerter poziomów logicznych CD40109B-Q1 firmy Texas Instruments [11]. Konwerter jest 4 kanałowy, co pozwala na podłączenie 4 sygnałów, więc wybrano połączenia CLK, LE, DATA_IN, BL. Konwerter pracuje w zakresie napięć od 3 V do 20 V, więc spełnia wymagania.

7.1.3 Sterowanie kropkami dziesiętnymi

Sterowanie kropkami dziesiętnymi odbywa się za pomocą tranzystorów HV firmy Diodes Industries o numerze DMN60H080DS, o następujących parametrach [10]:

- maksymalne napięcie dren-źródło - 600 V
- maksymalny prąd drenu - 80 mA
- napięcie progowe - ok. 2 V

7.1.4 Dobór rezystorów

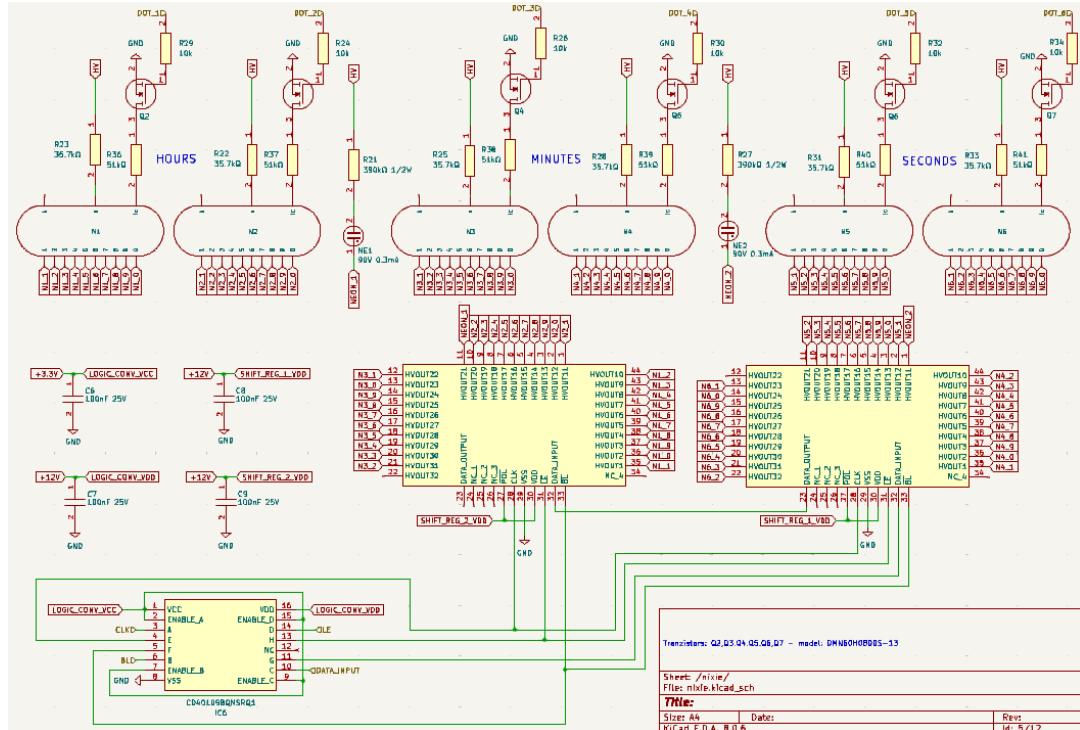
Wartość rezystorów anodowych dla zastosowanych lamp zostały obliczone w rozdziale 2. Kropki wymagają mniejszego prądu. Producent jednak nie podaje dokładnej wartości, więc przyjęto wartość $51\text{ k}\Omega$. Dobór rezystora zostanie oceniony empirycznie podczas testowania gotowego układu.

Lampy neonowe mają zdecydowanie mniejszy prąd pracy oraz mniejsze napięcie pracy. W sklepie internetowym sprzedający deklarował następujące parametry:

- wymagane napięcie - 90 V
- prąd pracy - 0.3 mA

$$R = \frac{U}{I} = \frac{220\text{ V} - 90\text{ V}}{0.3\text{ mA}} \approx 433\text{ k}\Omega \quad (7.1)$$

Rezystor potrzebny do zabezpieczenia lampy neonowej przy napięciu zasilania 220 V powinien mieć wartość około $433\text{ k}\Omega$. Zdecydowano się na użycie rezystora o wartości $390\text{ k}\Omega$ ze względu na dostępność w sklepie internetowym. Schemat elektryczny zaprojektowanego modułu przedstawiono na rysunku 7.1.



Rysunek 7.1: Schemat elektryczny modułu sterowania lampami

7.2 Przetwornica 12V na wysokie napięcie

Przetwornica impulsowa wysokiego napięcia jest najtrudniejszym do zaprojektowania elementem układu. Ma ona posiadać regulowane programowo napięcie wyjściowe, co dodatkowo komplikuje projekt. Nie może być również użyta zbyt duża cewka, ponieważ celem jest stworzenie niskiego urządzenia.

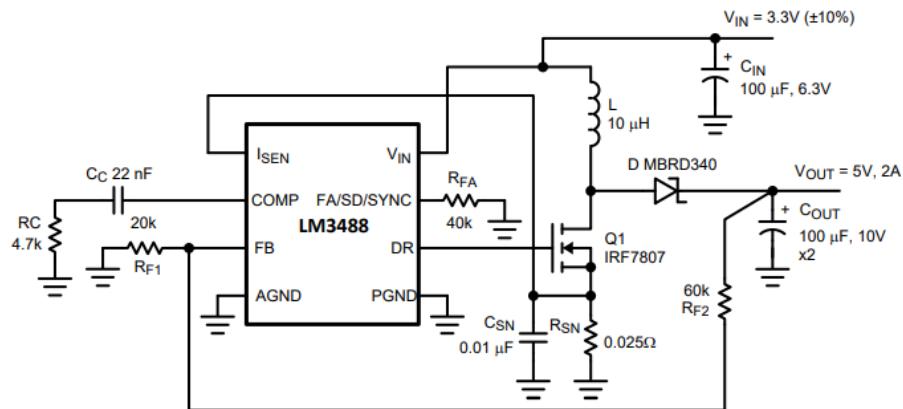
7.2.1 Założenia projektowe

- Napięcie wejściowe: 12 V
- Napięcie wyjściowe: 130-220 V
- Prąd wyjściowy: 20 mA
- 0.1 V tężnienia napięcia wyjściowego

7.2.2 Wybór układu scalonego

Wybrano układ LM3488 produkcji Texas Instruments, który jest układem przeznaczonym do budowy przetwornic typu Boost oraz Flyback. Jest to układ high efficiency, co jest powodem dla którego został wybrany [12].

Jako początkowe założenie przyjęto częstotliwość przełączania 400 kHz zgodnie z domyślną wartością w nocy katalogowej układu, ale możliwe jest zwiększenie częstotliwości do 1 MHz, co pozwala na zmniejszenie rozmiarów cewki oraz kondensatorów, natomiast może to pogorszyć sprawność układu, ponieważ według karty katalogowej wraz ze wzrostem częstotliwości spada wzmacnienie układu, co przekłada się na mniejszą sprawność. Podczas projektowania wzorowano się na schemacie z karty katalogowej, który przedstawiono na rysunku 7.2.



Rysunek 7.2: Schemat układu z karty katalogowej [12]

7.2.3 Dobór cewki

Wartość prądu cewki oszacowano na podstawie założonego prądu wyjściowego:

$$I_l = \frac{V_{out} \cdot I_{out}}{V_{in}} = \frac{220 \cdot 0.02}{12} \approx 0.36 \text{ mA} \quad (7.2)$$

Dodatkowe 30% prądu wyjściowego zostało dodane jako tężnienia prądu, co pozwala oszacować wartość maksymalnego prądu cewki:

$$I_{peak} = (1 + 0.3) \cdot I_l = 1.3 \cdot 0.36 \approx 0.468 \text{ A} \quad (7.3)$$

Wynika z tego, że potrzebna jest cewka o prądzie przewodzenia większym niż 0.5 A.

Obliczono wypełnienie sygnału PWM, na podstawie wzoru:

$$D_{220} = \frac{V_{out} - V_{in}}{V_{out}} = \frac{220 - 12}{220} \approx 0.945 \quad (7.4)$$

$$D_{130} = \frac{V_{out} - V_{in}}{V_{out}} = \frac{130 - 12}{130} \approx 0.908 \quad (7.5)$$

Zgodnie z kartą katalogową układu LM3488, cewka powinna mieć wartość określana wzorem:

$$L > \frac{D(1 - D)V_{in}}{2f_{sw}I_{out}} \quad (7.6)$$

Według dokumentacji I_{out} podczas obliczeń powinno stanowić 30% minimalnej wartości prądu wyjściowego:

$$I_{out} = 0.3 \cdot 20 \text{ mA} = 6 \text{ mA} \quad (7.7)$$

Dla napięcia wyjściowego 220 V oraz napięcia wejściowego 12 V oraz częstotliwości 400 kHz otrzymano wartość cewki:

$$L_{220} > \frac{0.945 \cdot (1 - 0.945) \cdot 12}{2 \cdot 400000 \cdot 0.006} \approx 128.9 \mu\text{H} \quad (7.8)$$

Natomiast dla napięcia wyjściowego 130 V oraz napięcia wejściowego 12 V oraz częstotliwości 400 kHz otrzymano wartość cewki:

$$L_{130} > \frac{0.908 \cdot (1 - 0.908) \cdot 12}{2 \cdot 400000 \cdot 0.006} \approx 209.5 \mu\text{H} \quad (7.9)$$

Napotkano problem z doborem cewki, ponieważ nie udało się znaleźć w sklepie cewki o wartości powyżej 200 μH w rozsądnej cenie i odpowiednich rozmiarach. Dlatego zdecydowano się na zastosowanie cewki o wartości 180 μH , która jest najbliższą wartością dostępną w sklepie. Wartość graniczna prądu cewki wynosi 0.9 A, co jest wystarczającym zapasem. Wybrana cewka ma rezystancję 6.3 m Ω . Moc strat w cewce wynosi:

$$P_l = I_{peak}^2 \cdot R = 0.468^2 \cdot 0.0063 \approx 1.4 \text{ mW} \quad (7.10)$$

W celu osiągnięcia założonego zakresu napięcia wyjściowego z użyciem wybranej cewki, zdecydowano się na zwiększanie częstotliwości przełączania do 500 kHz. Obliczono wartość cewki dla napięcia wyjściowego 130 V oraz napięcia wejściowego 12 V oraz częstotliwości 500 kHz:

$$L_{220} > \frac{0.945 \cdot (1 - 0.945) \cdot 12}{2 \cdot 500000 \cdot 0.006} \approx 103.1 \mu\text{H} \quad (7.11)$$

$$L_{130} > \frac{0.908 \cdot (1 - 0.908) \cdot 12}{2 \cdot 500000 \cdot 0.006} \approx 167.6 \mu\text{H} \quad (7.12)$$

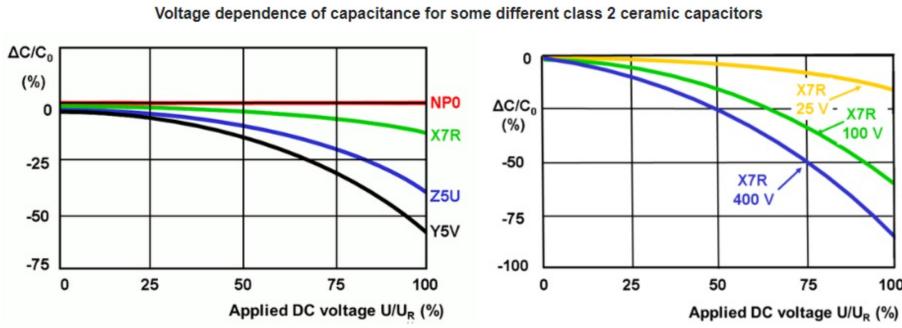
Z obliczeń wynika, że zwiększanie częstotliwości pozwala na zmniejszenie wartości cewki, co pozwala na zastosowanie cewki o wartości 180 μH .

7.2.4 Dobór kondensatorów

Według zaleceń z karty katalogowej w przetwornicy powinny być zastosowane kondensatory o jak najniższym ESR, dlatego odrzucono kondensatory elektrolityczne na rzecz kondensatorów ceramicznych, które mają bardzo niski ESR, tak mały że producent nie podaje tej wartości w notach katalogowych, gdyż jest ona zbyt mała by miała znaczenie.

Jednak problemem jest znalezienie kondensatorów ceramicznych pracujących przy wysokim napięciu, mimo tego został znaleziony kondensator ceramiczny o wartości $2.2 \mu\text{F}$ i napięciu pracy do 250 V .

W obliczeniach należy uwzględnić spadek pojemności kondensatora wraz ze wzrostem napięcia. Wartości spadku pojemności dla kondensatora ceramicznego odczytano z [st:ceramic-capacitor].



Rysunek 7.3: Spadek pojemności kondensatora ceramicznego wraz ze wzrostem napięcia [13]

Wybrany kondensator jest klasy 2, wykonany z materiału X7R, co oznacza, że jego pojemność spadnie w przybliżeniu o 30% dla napięcia 220 V . Zdecydowano się na zastosowanie 2 kondensatorów o wartości $2.2 \mu\text{F}$ połączonych równolegle w celu zminimalizowania tętnień napięcia wyjściowego. Ostatecznie pojemność oszacowana na:

$$C_{220} = 2 \cdot (1 - 0.3) \cdot 2.2 \mu\text{F} = 3.08 \mu\text{F} \quad (7.13)$$

Obliczono tętnienia dla dobranych wartości kondensatorów:

$$\Delta V_{220} = \frac{V_{out}}{2 \cdot \frac{V_{out}}{I_{out}} \cdot C} \cdot \frac{D}{f_{sw}} = \frac{220}{2 \cdot \frac{220}{0.02} \cdot 3.08 \mu\text{F}} \cdot \frac{0.945}{500000} \approx 6.1 \text{ mV} \quad (7.14)$$

$$\Delta V_{130} = \frac{V_{out}}{2 \cdot \frac{V_{out}}{I_{out}} \cdot C} \cdot \frac{D}{f_{sw}} = \frac{130}{2 \cdot \frac{130}{0.02} \cdot 3.08 \mu\text{F}} \cdot \frac{0.908}{500000} \approx 5.89 \text{ mV} \quad (7.15)$$

Wartości tętnień jest mniejsza niż założone 0.1 V , co oznacza, że dobrano odpowiednie wartości kondensatorów.

7.2.5 Dobór diody

Oszacowano prąd diody na podstawie wzoru z karty katalogowej:

$$I_d = \frac{I_{out}}{1 - D} + \Delta I_{out} = \frac{0.02}{1 - 0.945} + 0.006 \approx 0.37 \text{ A} \quad (7.16)$$

Dioda powinna mieć prąd przewodzenia większy niż 0.4 A oraz być diodą szybko przełączającą, by zminimalizować straty w układzie.

Zdecydowano się na zastosowanie diody ES1G firmy Onsemi, która jest diodą super szybką, o prądzie przewodzenia 1 A , co jest wystarczające dla tego zastosowania. Dioda ta ma maksymalne napięcie wsteczne 400 V , co pozwala na bezpieczne zastosowanie w tym układzie.

7.2.6 Dobór tranzystora

Można założyć że prąd tranzystora to prąd cewki, czyli 20 mA. Zgodnie z kartą katalogową tranzystor powinien mieć następujące parametry:

- Napięcie minimalnie drain-source: 250 V
- Jak najmniejszy $R_{DS(on)}$
- Prąd przewodzenia większy niż 0.5 A
- Niskie napięcie progowe V_{TH}
- Jak najmniejszy ładunek bramki
- Wymaganego prąd bramki mniejszego niż 1 A

Zdecydowano się na zastosowanie tranzystora N-Channel MOSFET TPH5200FNH firmy Toshiba, który ma następujące parametry:

- Napięcie drain-source: 250 V
- $R_{DS(on)}$: 44 mΩ
- Prąd przewodzenia: 26 A
- Napięcie progowe: 2 V
- Ładunek bramki: 22 nC
- Rozpraszana moc: 2.5 W

Średni prąd bramki potrzebny do załączenia tranzystora można obliczyć na podstawie wzoru:

$$I_g = Q_g \cdot f_{sw} = 22 \text{ nC} \cdot 500000 = 11 \text{ mA} \quad (7.17)$$

Tranzystor ten spełnia wszystkie założenia, a także ma bardzo niskie $R_{DS(on)}$, co pozwala na zminimalizowanie strat w układzie. Moc wydzielana na tranzystorze można obliczyć na podstawie wzoru:

$$P_{mos} = I_l^2 \cdot R_{DS(on)} = 0.468^2 \cdot 0.044 \approx 0.01 \text{ W} \quad (7.18)$$

Moc jest bardzo niska, co oznacza, że tranzystor nie będzie się nagrzewał, a także nie będzie wymagał radiatora.

7.2.7 Ustawienie napięcia wyjściowego

Napięcie wyjściowe będzie regulowane, dlatego zdecydowano się na zastosowanie potencjometru cyfrowego włączonego w obwód sprzężenia zwrotnego. Potencjometr cyfrowy będzie się komunikował z mikrokontrolerem za pomocą magistrali I2C, co pozwoli na programowe ustawianie napięcia wyjściowego, by regulować jasność lamp.

Wybrano potencjometr cyfrowy MCP4018T-103E/LT firmy Microchip, który ma 128 poziomów ustawień, co pozwala na dokładne ustawienie napięcia wyjściowego, wybrano wartość 10 kΩ, co pozwala na uzyskanie odpowiedniego zakresu ustawień napięcia wyjściowego.

Zgodnie z kartą katalogową napięcie na pinie FB powinno wynosić 1.26 V, napięcie to oznacza, że napięcie wyjściowe jest odpowiednie. Po przetestowaniu kilku kombinacji zdecydowano się na zastosowanie następujących rezystorów:

- $R_{fb1} = 2.49 \text{ M}\Omega$
 - $R_{fb2} = 14.39 \text{ k}\Omega$

Obliczone napięcie na wyjściu dla potencjometru z nastawą $10\text{k}\Omega$:

$$V_{out} = 1.26 \cdot \left(1 + \frac{R_{fb1}}{R_{fb2} + R_{pot}}\right) = 1.26 \cdot \left(1 + \frac{2.49 \text{ M}\Omega}{14.39 \text{ k}\Omega + 10 \text{ k}\Omega}\right) \approx 130.4 \text{ V} \quad (7.19)$$

Obliczone napięcie na wyjściu dla potencjometru z nastawą 0Ω :

$$V_{out} = 1.26 \cdot \left(1 + \frac{R_{fb1}}{R_{fb2} + R_{pot}}\right) = 1.26 \cdot \left(1 + \frac{2.49 \text{ M}\Omega}{14.39 \text{ k}\Omega}\right) \approx 220.6 \text{ V} \quad (7.20)$$

Uzyskano zakres napięcia wyjściowego od 130.4 V do 220.6 V, co jest zgodne z założeniami projektowymi.

7.2.8 Dobór rezystora ograniczającego prąd

Układ ma możliwość ustawienia limitu prądu jaki będzie płynąć przez tranzystor, co jest dodatkowym zabezpieczeniem przed uszkodzeniem tranzystora.

Najpierw obliczono wartość limitu szczytowego prądu przełączania zgodnie z kartą katalogową:

$$ISW_{limit} = \left(\frac{I_{out}}{1-D} + \frac{D \cdot V_{in}}{2 \cdot f_{sw} \cdot L} \right) = \left(\frac{0.02}{1-0.945} + \frac{0.945 \cdot 12}{2 \cdot 500000 \cdot 180 \mu\text{H}} \right) \approx 0.426 \text{ A} \quad (7.21)$$

Następnie obliczono wartość rezystora ograniczającego prąd zgodnie z kartą katalogową:

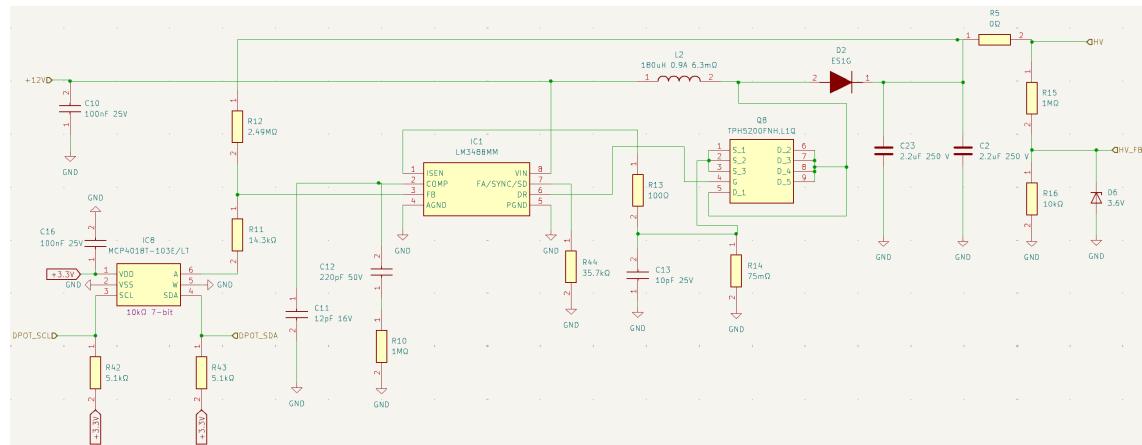
$$R_{sense} = \frac{V_{SENSE} - (D \cdot V_{SENSE} \cdot V_{SL-ratio})}{ISW_{limit}} = \frac{156 \text{ mV} - (0.945 \cdot 156 \text{ mV} \cdot 0.49)}{0.426} \approx 74 \text{ m}\Omega \quad (7.22)$$

Następnie sprawdzono warunek na maksymalną wartość rezystora ograniczającego prąd:

$$R_{sense} < \frac{2 \cdot V_{SL} \cdot f_{sw} \cdot L}{V_{out} - (2 \cdot V_{IN})} = \frac{2 \cdot 92 \text{ mV} \cdot 500000 \cdot 180 \text{ }\mu\text{H}}{220 - (2 \cdot 12)} \approx 84 \text{ m}\Omega \quad (7.23)$$

Zdecydowano się na zastosowanie rezystora o wartości $75\text{ m}\Omega$, który spełnia wszystkie założenia.

Został również dodany kondensator o wartości 10 pF w celu zminimalizowania tętnień napięcia na rezystorze oraz rezystor kompensujący 100Ω . Schemat elektryczny przetwornicy przedstawiono na rysunku 7.4.



Rysunek 7.4: Schemat elektryczny przetwornicy 12 V na wysokie napięcie

7.3 Przetwornica 12V na 5V

7.3.1 Założenia projektowe

- Napięcie wejściowe: 12 V
- Napięcie wyjściowe: 5 V
- Prąd wyjściowy: 2 A
- 50 mV tętnienia napięcia wyjściowego

7.3.2 Wybór układu scalonego

Zdecydowano się na układ TPS563219ADDFR produkcji Texas Instruments, który jest przetwornicą impulsową z wbudowanym tranzystorem mocy oraz zapewniającym prąd wyjściowy do 3 A przy napięciu wyjściowym do 7 V. Układ jest też w obudowie na tyle dużej, by móc go polutować ręcznie.

Układ posiada soft-start oraz wyjście power good (potwierdzające start przetwornicy), co nie jest potrzebne w tym zastosowaniu, tak samo nie jest to najmniejszy układ, ale zapewnia to łatwość montażu co jest ważne w tym przypadku.

7.3.3 Dobór komponentów

Dobór komponentów wykonano na podstawie sugerowanych wartości z noty katalogowej układu TPS563219ADDFR [14]. Tabelę z wartościami komponentów z noty katalogowej przedstawiono na rysunku 7.5.

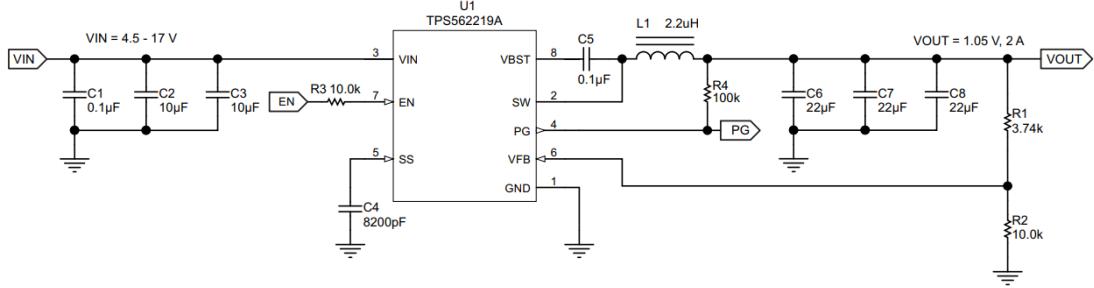
Table 4. TPS563219A Recommended Component Values

Output Voltage (V)	R2 (kΩ)	R3 (kΩ)	L1 (μH)			C6 + C7 + C8 (μF)
			MIN	TYP	MAX	
1	3.09	10.0	1.0	1.5	4.7	20 - 68
1.05	3.74	10.0	1.0	1.5	4.7	20 - 68
1.2	5.76	10.0	1.0	1.5	4.7	20 - 68
1.5	9.53	10.0	1.0	1.5	4.7	20 - 68
1.8	13.7	10.0	1.5	2.2	4.7	20 - 68
2.5	22.6	10.0	1.5	2.2	4.7	20 - 68
3.3	33.2	10.0	1.5	2.2	4.7	20 - 68
5	54.9	10.0	2.2	3.3	4.7	20 - 68
6.5	75	10.0	2.2	3.3	4.7	20 - 68

Rysunek 7.5: Tabela doboru komponentów z noty katalogowej [14]

Projekt wzorowano na schemacie z noty katalogowej, który przedstawiono na rysunku 7.6. Na podstawie tabeli dobrano następujące wartości komponentów:

- C1: 22 μF 10 V
- C29: 22 μF 10 V
- L1: 2.2 μH 9.2 A 14.5 mΩ
- R1: 56 kΩ
- R2: 10 kΩ



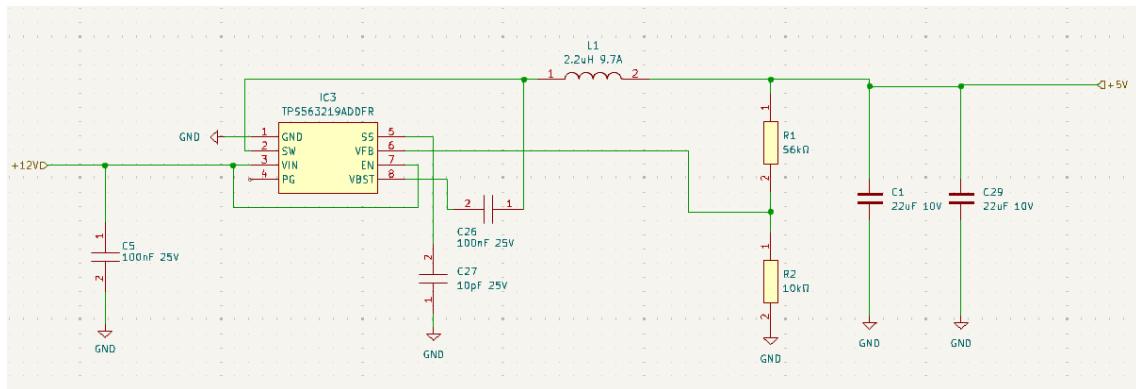
Rysunek 7.6: Typowe połączenie układu TPS563219ADDFR [14]

Kondensator dołączony do wyprowadzenia SS (soft start) oraz kondensator dołączony do pinu VBST został skopiowany z układu z noty katalogowej, gdyż nie jest to krytyczny element i nie ma potrzeby doboru wartości pod kątem zastosowania w zegarze.

Zdecydowano się na użycie kondensatorów ceramicznych, gdyż są one mniejsze i mają lepszy ESR niż elektrolityczne, co ma znaczenie przy przetwornicach impulsowych, gdzie mamy wyższe częstotliwości przełączania. Przy zbyt dużym ESR kondensatora może on się nagrzewać, co prowadzi do jego uszkodzenia. Kondensatory ceramiczne natomiast cechują się tak małym ESR, że producent nie podaje tej wartości w notach katalogowych, gdyż jest ona zbyt mała, by miała znaczenie.

Cewkę dobrano, biorąc pod uwagę jej stosunek oporu do ceny. Zdecydowano się na cewkę o oporze $14.5\text{ m}\Omega$, gdyż jest to najwyższa wartość, jaką udało się znaleźć w sklepach elektronicznych w sensownej cenie i dość małej obudowie.

Dodano również kondensator filtrujący 100 nF na pinie VCC, aby zredukować szумy z linii zasilania. Schemat elektryczny zaprojektowanego modułu przedstawiono na rysunku 7.7.



Rysunek 7.7: Schemat elektryczny modułu przetwornicy z 12 V na 5 V

7.4 Złącze zasilania

7.4.1 Dobór złącza

W doborze złącza kluczowa była jego wielkość oraz prąd, który jest w stanie przewodzić. Teoretyczna moc układu to 10W, co przy napięciu 12 V daje prąd 0.83 A. Złącze musi być w stanie przewodzić prąd 1.5A, by zapewnić bezpieczeństwo.

Wybrano złącze firmy Same Sky o symbolu PJ-094H-SMT-TR o styku 0.65x2.35 mm, które jest w stanie przewodzić prąd 2.5A, czyli więcej niż wystarczająco. Złącze jest bardzo niskie, jego wysokość to 3.5mm, co pozwala na zminimalizowanie wysokości zegara.

7.4.2 Opis podłączenia

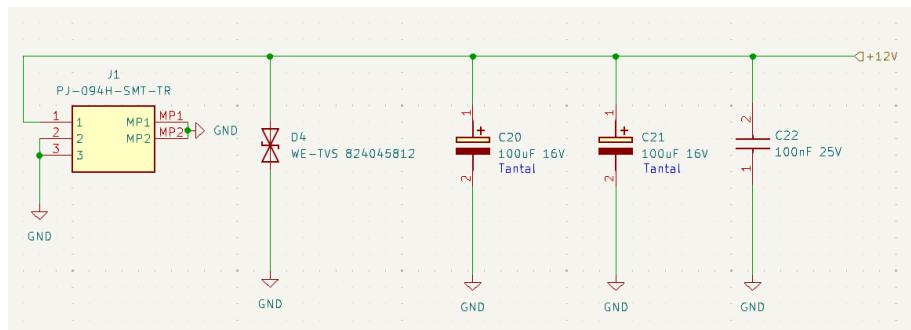
Jako kondensatory filtrujące wykorzystano, 2 kondensatory o pojemności 100 μ F w celu zminimalizowania zakłóceń niskich częstotliwości, oraz 1 kondensator o pojemności 100 nF w celu zminimalizowania zakłóceń wysokich częstotliwości.

Kondensatory 100 μ F są kondensatorami tantalowymi, a kondensator 100 nF jest kondensatorem ceramicznym. Wykorzystano te rodzaje kondensatorów, ponieważ są to kondensatory o długiej żywotności, a także mają one małe rozmiar w przeciwieństwie do kondensatorów elektrolitycznych.

7.4.3 Zabezpieczenia ESD

W celu zabezpieczenia linii przed przepięciami, wykorzystano diodę Transila firmy Wurth Elektronik o symbolu 824045812. Dioda ta jest diodą TVS o napięciu przebicia 13.3 V, oraz napięciu stabilizacji 15 V. Dioda musi mieć jak najmniejsze napięcie przebicia, by skok napięcia nie uszkodził rejestrów przesuwnych HV, które są wrażliwe na napięcia powyżej 13.2 V.

Dioda ta została wybrana gdyż nie udało się znaleźć transila o napięciu stabilizacji 13 V. Ma ona również dużą pojemność, co powoduje, że nie jest ona zalecana do zastosowań z wysokimi częstotliwościami, jednak w tym przypadku nie jest to problemem, gdyż jest to tylko złącze zasilania o stałym napięciu. Schemat złącza zasilania przedstawiono na rysunku 7.8.



Rysunek 7.8: Schemat elektryczny złącza zasilania

7.5 Złącze do programowania

7.5.1 Dobór złącza

Złącze USB musi posiadać przynajmniej 12 wyprowadzeń, ponieważ dopiero w takim układzie na złączu są linie D+ i D-, czyli linie danych. Wybrano złącze z 16 wyprowadzeniami, ponieważ takie było dostępne w sklepie.

7.5.2 Opis podłączenia

By ustawić napięcie komunikacji USB-C na 3.3 V, zastosowano rezystory podciągające R1 i R2 o wartości $5.1\text{ k}\Omega$. Do podłączenia wykorzystano parę różnicową, by połączyć linie D+ i D- z ESP32-S3, w celu zminimalizowania zakłóceń CMN (Common Mode Noise).

7.5.3 Zabezpieczenia ESD

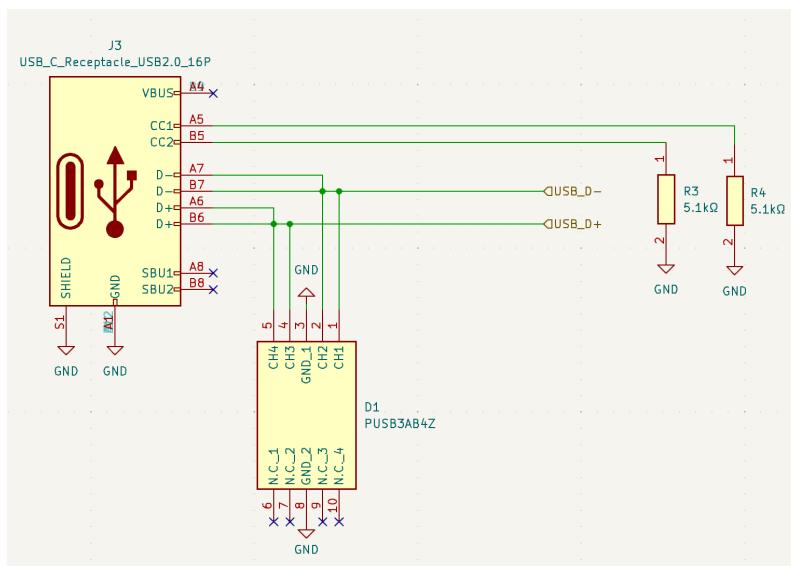
W celu zabezpieczenia linii przed przepięciami, zastosowano transil PUSB3AB4Z firmy Nexperia. Diody te mają wystarczająco duże opakowanie, by dało się je przylutować ręcznie. Napięciem roboczym jest 3.3 V, a napięcie stabilizacji wynosi 5 V.

Mimo że jest to napięcie wyższe niż napięcie zasilania ESP32-S3, to nie powinno to stanowić problemu, ponieważ napięcie to pojawi się na krótki czas, a sam mikrokontroler ma również wbudowane zabezpieczenia przed przepięciami.

Wewnętrzne zabezpieczenia według karty katalogowej ESP32-S3:

- Test Standard JS-001; HBM (Human Body Mode) ± 2000 V
 - Test Standard JS-002; CDM (Charged Device Model) ± 1000 V

Wynika z tego, że złącze USB-C jest dość dobrze zabezpieczone przed przepięciami. Schemat elektryczny złącza USB-C przedstawiono na rysunku 7.9.



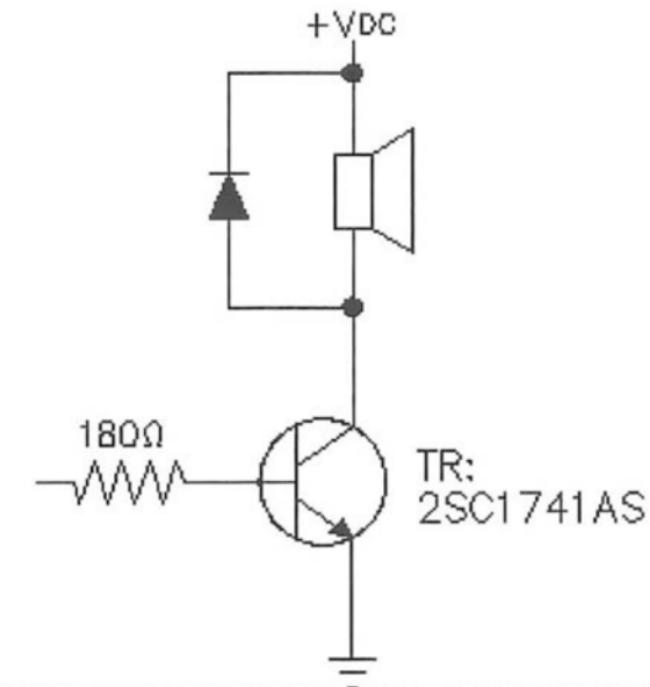
Rysunek 7.9: Schemat elektryczny złącza USB-C

7.6 Buzzer

Kryterium wyboru buzzera było jego napięcie zasilania, głośność oraz jak najmniejszy rozmiar. Wybrano buzzer CEM120342, który jest buzzerem piezoelektrycznym. Z noty katalogowej [15] wypisano użyteczne parametry na etapie projektowania:

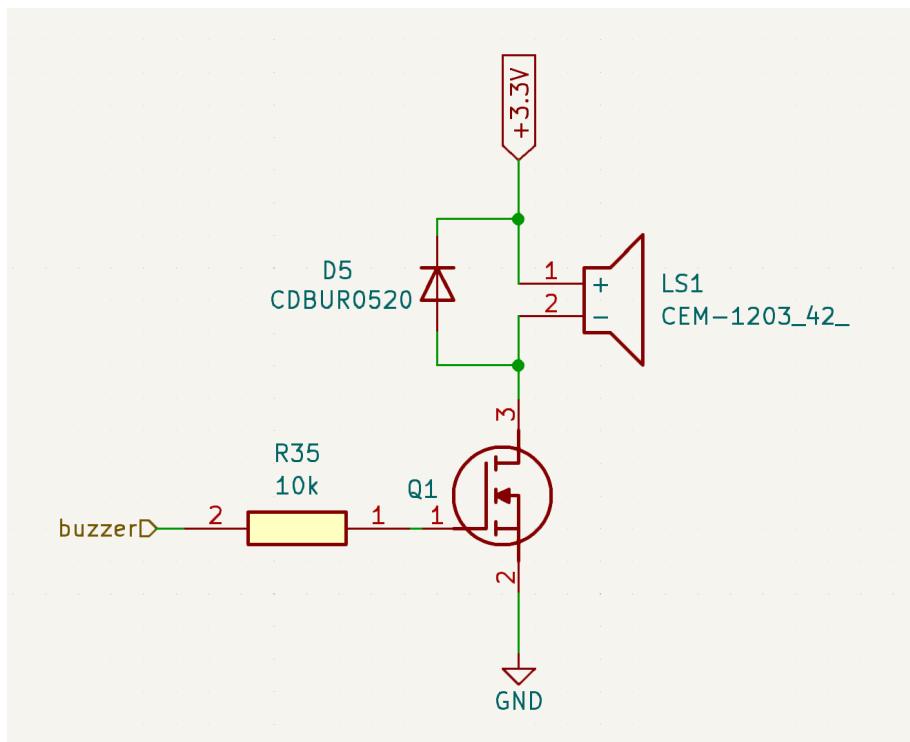
- Napięcie zasilania: 3-5 V
- Max prąd zasilania: 35 mA
- Częstotliwość: 2,048 kHz
- Głośność: 85 dB - 95 dB
- Średnica: 12 mm, wysokość: 8 mm
- Rezystancja: 42Ω
- Typ montażu: THT

Buzzer został podłączony według schematu zalecanego przez producenta przedstawionego na rysunku 7.10.



Rysunek 7.10: Schemat zalecany przez producenta [15]

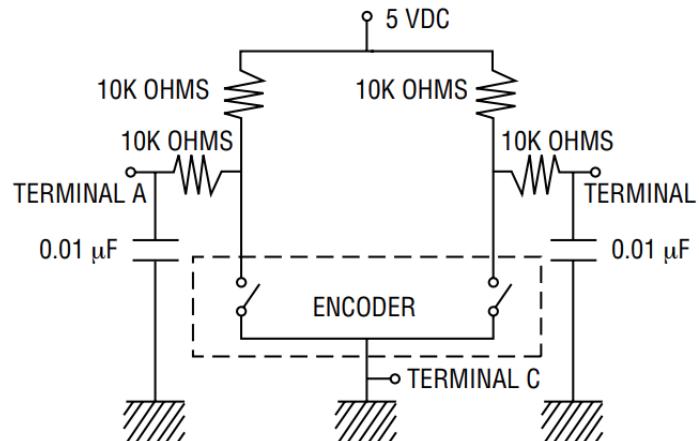
Do sterowania wykorzystano inny tranzystor. Użyto tranzystora, który służy do sterowania katodami kropek w lampach Nixie, by nie dodawać kolejnego elementu do projektu. Tranzystor ten ma prąd kolektora 80 mA, co jest wystarczające do sterowania buzzerem. Buzzer jest dostatecznie mały, jedyną wadą jest montaż THT, który ogranicza miniaturyzację płytki drukowanej. Schemat elektryczny złącza z buzzerem przedstawiono na rysunku 7.11.



Rysunek 7.11: Schemat elektryczny podłączenia buzzera

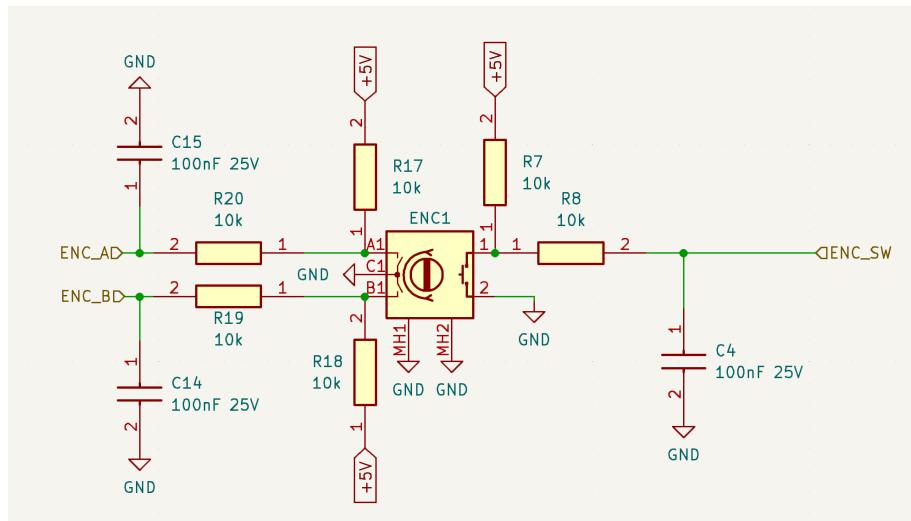
7.7 Enkoder obrotowy

Jednymi kryteriami wyboru enkodera obrotowego były napięcie zasilania oraz to, by posiadał on przycisk. Wybrano encoder PEC12R41BBFS0012 firmy Bourns [16]. Jest to 2-kanałowy encoder o rozdzielczości 12 impulsów na obrót. Jego napięcie robocze wynosi 5 V. Producent w karcie katalogowej informuje również, jakie filtry zastosować, aby zapobiec zakłóceniom. Schemat zalecany przez producenta przedstawiono na rysunku 7.12.



Rysunek 7.12: Schemat filtrów zalecany przez producenta [16]

Dodatkowo dodano filtr dolnoprzepustowy przy przycisku enkodera, aby zapobiec drganiom styków i nie musieć implementować debouncingu w oprogramowaniu. Schemat elektryczny podłączenia enkodera obrotowego przedstawiono na rysunku ??.

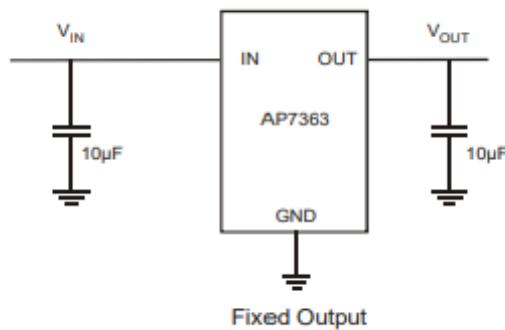


Rysunek 7.13: Schemat elektryczny enkodera obrotowego

7.8 LDO 5V na 3.3V

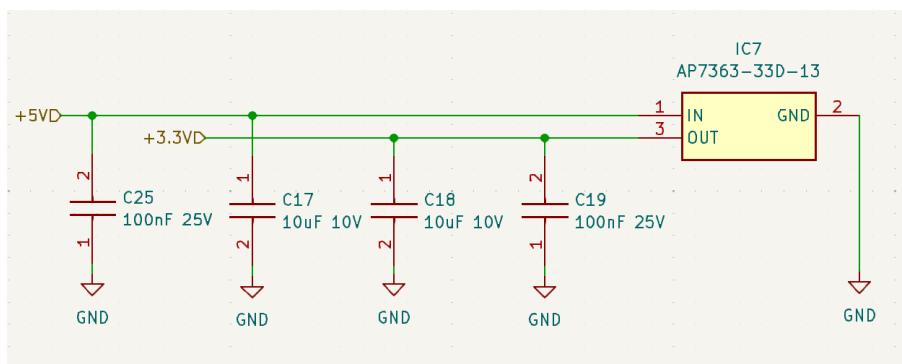
Wymagania, jakie musi spełnić LDO, to zapewnienie zasilania mikrokontrolera ESP32 oraz potencjometru cyfrowego. Zgodnie z dokumentacją ESP32 [5] pobiera on maksymalnie 340 mA prądu, a potencjometr cyfrowy [17] 100 mA, co daje łącznie 440 mA. Układ LDO ma być zasilany 5 V, a na wyjściu ma być 3.3 V.

Wybrano układ LDO AP7363 firmy Diodes Incorporated [18]. Jest to układ LDO o napięciu wyjściowym 3.3 V i prądzie wyjściowym 1.5 A, co daje duży zapas wydajności prądowej. Układ ten jest dostępny w obudowie TO252, co pozwala na łatwy montaż na płytce drukowanej. Producent zaleca podłączenie układu zgodnie z rysunkiem 7.14.



Rysunek 7.14: Schemat podłączenia LDO zalecany przez producenta [18]

Względem schematu zaproponowanego przez producenta zdecydowano się na zastosowanie dodatkowego kondensatora ceramicznego o pojemności 100 nF, by zminimalizować zakłócenia. Dodano po kondensatorze na wejście i wyjście układu równolegle do kondensatorów 10 µF zaproponowanych przez producenta. Schemat elektryczny układu LDO przedstawiono na rysunku 7.15.



Rysunek 7.15: Schemat elektryczny układu LDO

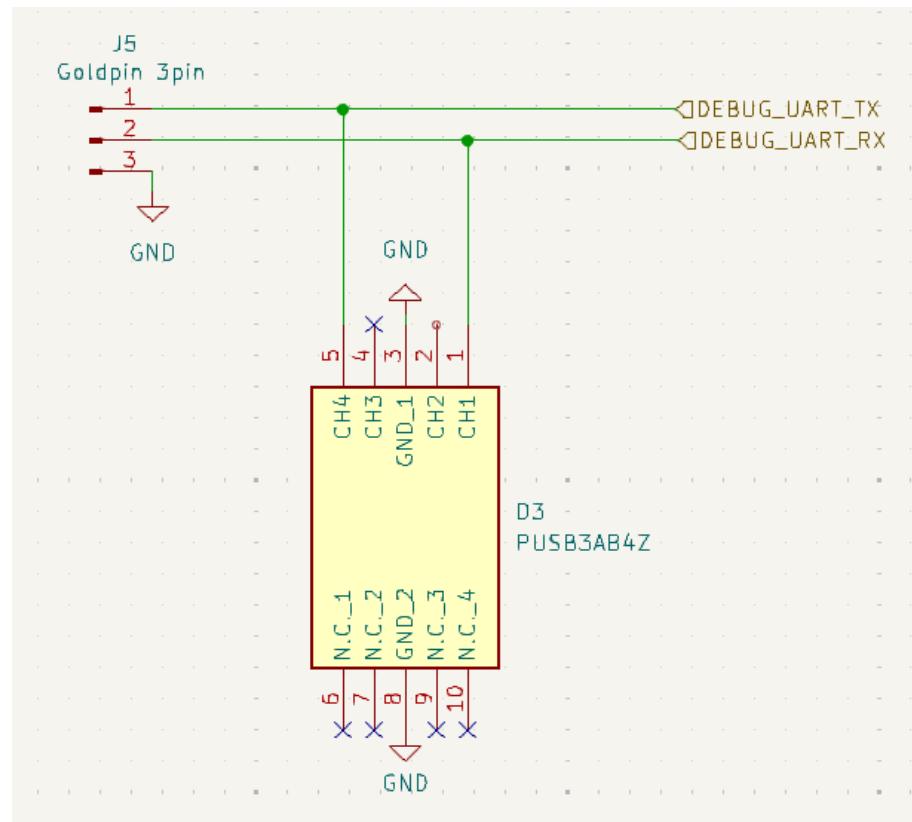
7.9 Złącze do uruchamiania układu

Wybrano horyzontalne złącze gold pin, ponieważ łatwo jest podłączyć do niego uniwersalne przewody żeńskie, często wykorzystywane w prototypowaniu. Horyzontalna wersja złącza została wybrana, by nie zwiększać wysokości płytki. Dodatkową zaletą tego rozwiązania jest ewentualna możliwość programowania mikrokontrolera za pomocą tego złącza w przypadku, gdyby nie udało się tego zrobić przez USB.

Złącze będzie wystawiać następujące sygnały:

- RX - odbiór danych
- TX - wysyłanie danych
- GND - masa

W celu zabezpieczenia złącza przed ESD zastosowano transil, analogiczny jak w podrozdziale 7.5. Schemat złącza przedstawiono na rysunku 7.16.



Rysunek 7.16: Schemat elektryczny złącza UART

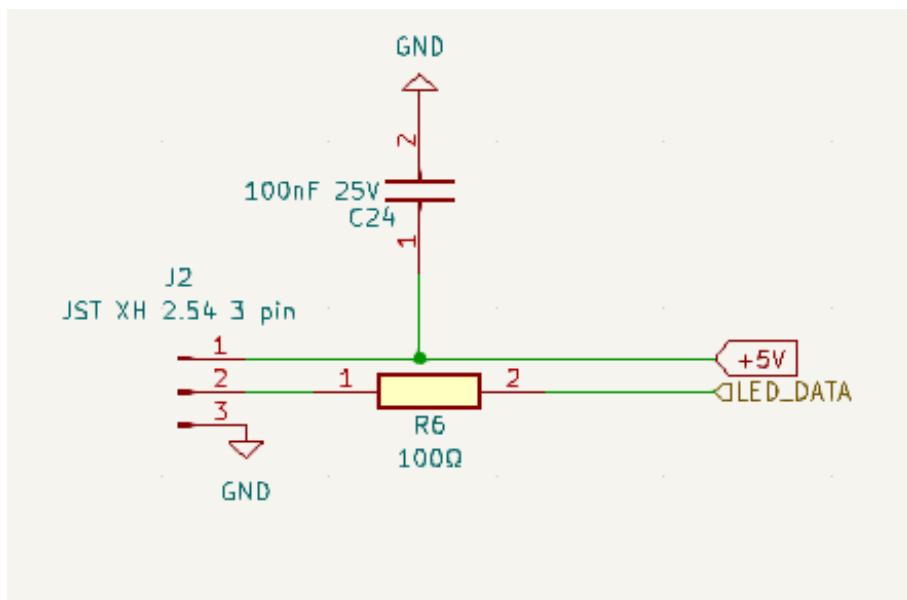
7.10 Złącze do paska LED

Wybrany pasek LED to adresowalny pasek zawierający diody LED o gęstości 60 LED na metr. Do każdej diody dodany jest chip sterujący WS2812B. Pasek LED to dodatkowy moduł, którego głównym celem jest aspekt wizualny, dlatego jedynie dodane zostało złącze na pasek LED. Moduł LED będzie wydrukowanym oddzielnym elementem wsadzanym od dołu obudowy. Zawierał będzie osłonę rozpraszającą światło dla lepszego efektu wizualnego.

Złącze będzie wystawiać następujące sygnały:

- 5 V - Zasilanie paska LED
- data - Sterowanie paskiem LED
- GND - masa

Wybrano złącze typu JST w orientacji horyzontalnej w celu zaoszczędzenia miejsca. Dodatkowo na linie danych dodano zabezpieczający rezystor 100 Ω . Zastosowano również kondensator odsprzęgający 100 nF. Schemat złącza przedstawiono na rysunku 7.17.

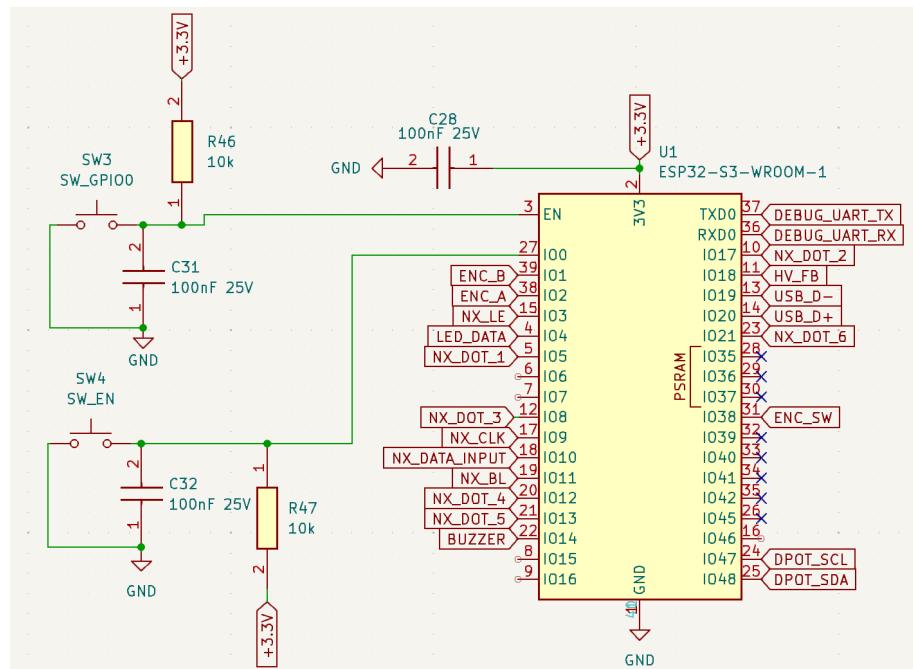


Rysunek 7.17: Schemat złącza paska LED

7.11 Mikrokontroler ESP32-S3

ESP32-S3 wymaga jedynie dodania przycisków do resetowania i wprowadzenia w tryb programowania. Dodano więc 2 przyciski, jeden do pinu EN (Enable), a drugi do pinu IO0, który jest pinem programowania. Każdy z przycisków ma dodatkowo podłączony kondensator ceramiczny o pojemności 100 nF, by zminimalizować zakłóczenia oraz rezystor podciągający o wartości 10 kΩ.

Dodano również kondensator odsprzęgający 100 nF na zasilanie ESP32-S3. W związku z tym, że większość pinów ESP32-S3 może pełnić dowolne funkcje, istniała duża dowolność w podłączeniu pinów, co pozwoliło na łatwiejsze rysowanie ścieżek na płytce drukowanej. Schemat podłączenia ESP32-S3 przedstawiono na rysunku 7.18.



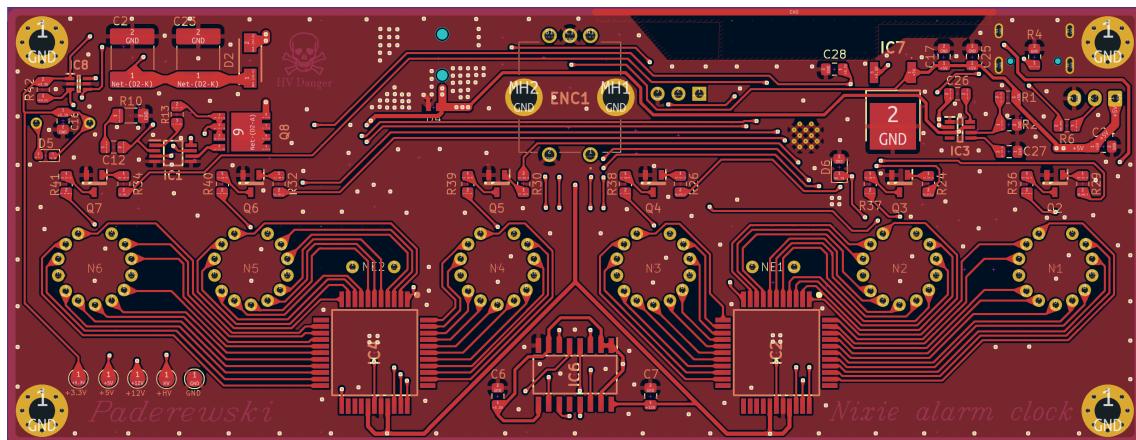
Rysunek 7.18: Schemat podłączenia ESP32-S3

8 Projekt i montaż płytki drukowanej

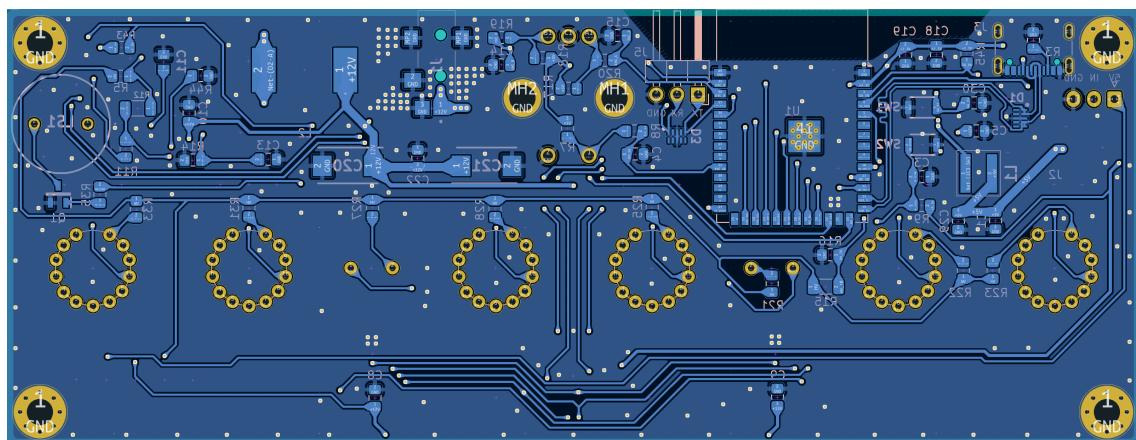
8.1 Projekt płytki drukowanej

Do zaprojektowania płytki drukowanej wykorzystano open-source'owy program KiCad. Zdecydowano się na zastosowanie dwustronnej płytki drukowanej. Ostatecznie udało się uzyskać płytę o wymiarach 138x53 mm oraz szacowaną wysokość wraz z komponentami na poziomie 1 cm, co oznacza, że udało się osiągnąć cel wykonania jak najmniejszej płytki drukowanej.

W celu poprawy aspektów wizualnych płytki zamówiona została pozłacana płyta drukowana. Uzyskano też pozłacaną ramkę, która powstała przez nienakładanie soldermaski na krawędzie płytki. W taki sam sposób uzyskano pozłacane napisy. Płytkę posiada cztery otwory montażowe na śruby M3. Otwory montażowe płytki oraz otwory montażowe enkodera obrotowego są podłączone do masy, co pozwala na lepsze prowadzenie masy między warstwami płytki. W celu uzyskania jak najlepszego połączenia masy między warstwami użyto dużo przelotek między warstwami. Zabieg ten pozwala na lepsze ekranowanie ścieżek, co powinno przyczynić się do zmniejszenia zakłóceń elektromagnetycznych. Na rysunkach ?? oraz ?? przedstawiono warstwy górna i dolna zaprojektowanej płytki drukowanej.



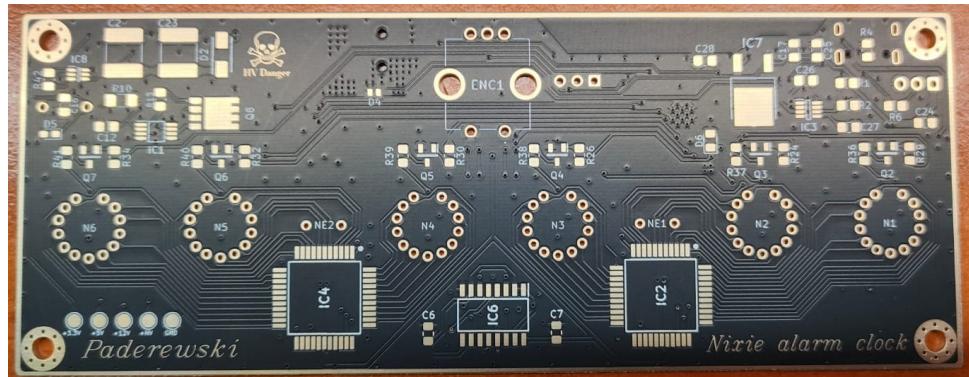
Rysunek 8.1: Góra warstwa płytki drukowanej



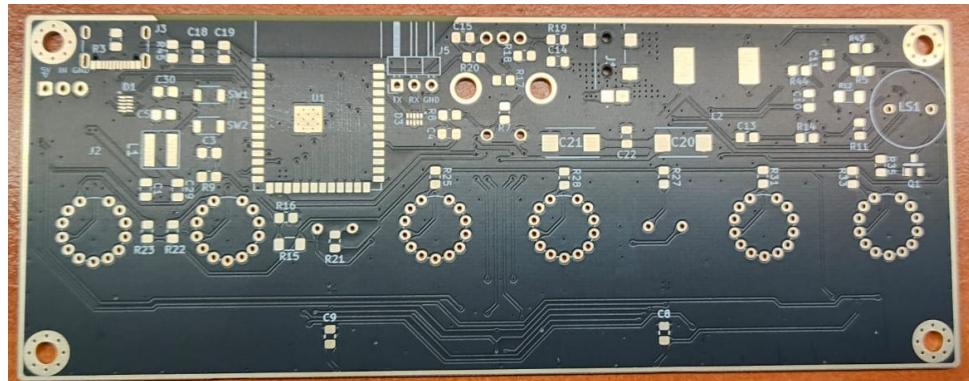
Rysunek 8.2: Dolna warstwa płytki drukowanej

8.2 Montaż i uruchomienie układu

Kolejnym etapem realizacji projektu był montaż elementów na płytce drukowanej. Przed rozpoczęciem lutowania wszystkie pozłacane elementy które ostały dodane jak elementy estetyczne, zostały zabezpieczone przed przypadkowym kontaktem z cyną, w wypadku gdyby cyna dostała się na pozłacane elementy, nie udałoby się jej usunąć. Do zabezpieczenia wykorzystano taśmę kaptonową, która jest odporna na wysokie temperatury.



Rysunek 8.3: Płytkę drukowaną - widok od góry

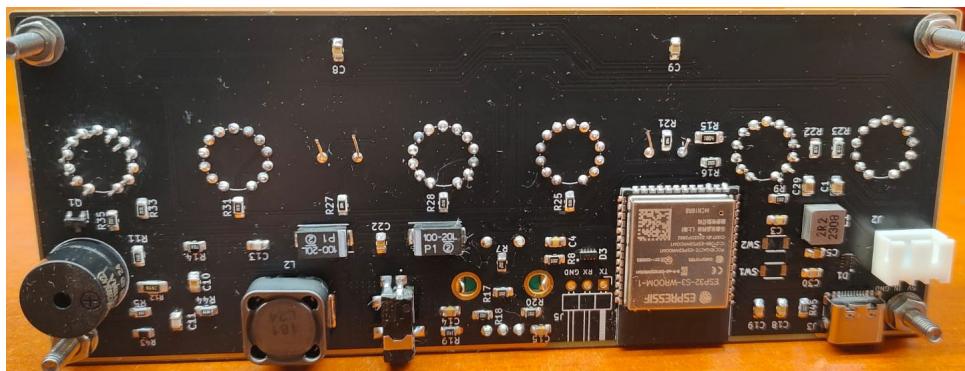


Rysunek 8.4: Płytkę drukowaną - widok od spodu

Lutowanie rozpoczęto od elementów najmniejszych SMD (elementów montowanych powierzchniowo). W celu łatwego lutowania elementów SMD, użyto pasty lutowniczej (flux) oraz stacji lutowniczej na gorące powietrze (hot air). Po zlutowaniu elementów SMD, wyczyszczono płytę z nadmiaru topnika, korzystającą z alkoholu izopropylowego. Następnym krokiem było lutowanie elementów przewlekanych (THT). Ostatnim mocowanym elementem były lampy Nixie, które trzeba było odpowiednio wypoziomować, by wszystkie lampy były na tej samej linii. Lampy mają duże tolerancje produkcyjne, co powoduje że niektóre z lamp mogą być wyżej lub niżej niż pozostałe. W celu wypoziomowania lamp, wykorzystano kątownik. Ostatnim etapem montażu było końcowe czyszczenie płytki z nadmiaru topnika. Zmontowane płytki drukowane przedstawiono na rysunkach 8.5 oraz 8.6.



Rysunek 8.5: Zmontowana płytka drukowana - widok od góry



Rysunek 8.6: Zmontowana płytka drukowana - widok od spodu

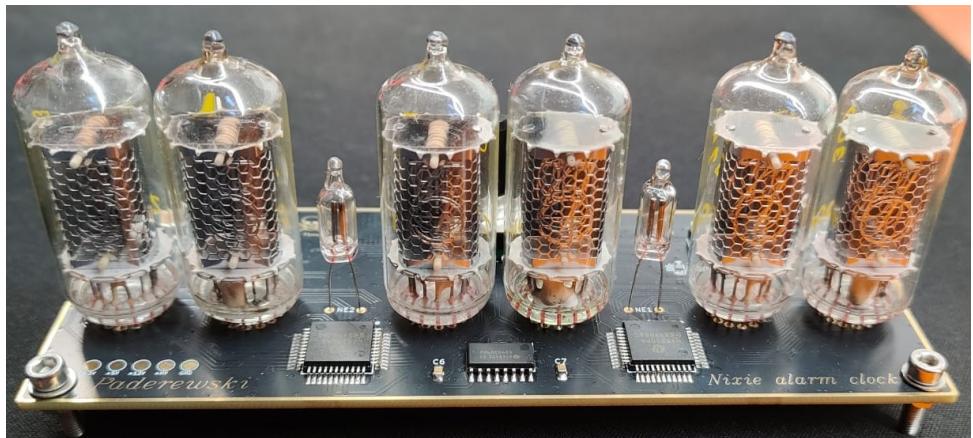
8.3 Uruchomienie układu

Przetwornica wysokiego napięcia jako najtrudniejszy moduł układu, została odizolowana na etap pierwszego uruchomiania, w celu zminimalizowania ryzyka uszkodzenia lamp Nixie. Zrealizowano izolacje poprzez nie przylutowanie rezystora $0\ \Omega$ na linii łączącej przetwornicę z lampami Nixie.

Przed podłączeniem zasilania, sprawdzono czy nie ma zwarcia na żadnej linii zasilania. Po podłączeniu zasilania, sprawdzono napięcia na każdej z sekcji zasilania. Pomierzone napięcia były następujące:

- linia 5 V - 5.081 V
- linia 3.3 V - 3.306 V
- linia 12 V - 12.01 V
- linia wysokiego napięcie - 162.1 V

Wszystkie napięcia są zgodne z wartościami oczekiwanyimi. Napięcia są stabilne i nie obserwuje się żadnych skoków napięcia. Ostatnim krokiem było dolutowanie rezystora $0\ \Omega$ łączącego przetwornicę z lampami Nixie. Na 8.7 został przedstawiony zmontowany układ.



Rysunek 8.7: Zmontowany układ

9 Oprogramowanie

Bazując na informacjach z podrozdziału 3.4.1, wybrano środowisko programistyczne PlatformIO, które jest wtyczką do Visual Studio Code. Zaletą tego rozwiązania nad Arduino IDE są podpowiedzi składniowe, co znaczco przyspiesza pisanie kodu. PlatformIO pozwala również na łatwe zarządzanie bibliotekami, komunikację z użyciem portu szeregowego oraz możliwość debugowania kodu.

Określono wymagania, które musi spełniać oprogramowanie:

- Obsługa rejestrów przesuwnych w celu wyświetlenia cyfr na lampach Nixie.
- Połączenie z siecią Wi-Fi.
- Pobranie czasu z serwera czasu.
- Obsługa enkodera obrotowego.
- Komunikacja z potencjometrem cyfrowym w celu zmiany jasności wyświetlacza.
- Odczyt obecnego napięcia na przetwornicy wysokiego napięcia.
- Odtwarzanie dźwięku za pomocą głośnika piezoelektrycznego.
- Komunikacja z serwerem Home Assistant przez protokół MQTT.

W celu zrealizowania powyższych wymagań program został podzielony na moduły, które są odpowiedzialne za poszczególne funkcjonalności. Taki podział pozwolił też na testowanie poszczególnych funkcjonalności niezależnie od siebie i połączenie ich w pliku `main.cpp`.

9.1 Sterowanie lampami Nixie

W celu wyświetlenia cyfr na lampach Nixie moduł podzielono na 4 warstwy abstrakcji:

- Klasę samego rejestru przesuwnego, który pozwala na wysłanie 64 bitów danych do rejestru przesuwnego.
- Klasę wyświetlacza, który pozwala na wyświetlenie cyfry na konkretnym wyświetlaczu Nixie.
- Zdefiniowane struktury łączące cyfrę z konkretnym bitem w rejestrze przesuwnym.
- Klasę do animacji wyświetlacza.

Wysyłanie danych do rejestrów odbywa się za pomocą funkcji `send` w klasie `ShiftRegisterExpander`.

```
void send() {  
    for (size_t i = 0; i < this->outputCount; i++) {  
        digitalWrite(regClkPin, HIGH);  
        if (this->outputs[this->outputCount - i - 1]) {  
            digitalWrite(regDataPin, HIGH);  
        } else {  
            digitalWrite(regDataPin, LOW);  
        }  
        digitalWrite(regClkPin, LOW);  
    }  
}
```

```

    esp_rom_delay_us(1);
}

// latch toggle
digitalWrite(regLePin, HIGH);
esp_rom_delay_us(1);
digitalWrite(regLePin, LOW);
esp_rom_delay_us(1);
}

```

Animacja cyfr polega na zapaleniu każdej z cyfr pomiędzy 0 a obecnie wyświetlana cyfrą, w momencie zmiany danej cyfry na 0. Funkcjonalność ta jest zrealizowana w klasie **AnimationDriver** w metodzie **update**.

```

void update() {
    if (this->desiredDigit == this->currentDigit) {
        return;
    }

    if (!this->animating) {
        this->nixie.setDigit(this->desiredDigit);
        this->currentDigit = this->desiredDigit;
        return;
    }

    uint8_t animationStep = (millis() - this->startTime) / 50;

    if (animationStep > 9) {
        this->animating = false;
        this->currentDigit = this->desiredDigit;
        this->nixie.setDigit(this->desiredDigit);
        return;
    }

    uint8_t animatedDigit = 9 - animationStep;
    if (animatedDigit < this->currentDigit) {
        this->nixie.setDigit(animatedDigit);
        this->currentDigit = animatedDigit;
        return;
    }
}

```

9.2 Połaczenie z siecią Wi-Fi

W celu połączenia z siecią Wi-Fi, wykorzystano bibliotekę **WiFi.h** dostępną w środowisku PlatformIO. Dodatkowo by nie trzymać hasła do sieci na repozytorium, ze względów bezpieczeństwa,

stworzono oddzielny plik `secret.hpp`, w którym przechowywane są dane do połączenia z siecią. Do inicializacji połączenia z siecią służy funkcja `WIFI_Init`.

```
void WIFI_Init() {
    Serial.println("[wifi] Starting WiFi");

    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    Serial.print("[wifi] Connecting to network ssid: ");
    Serial.print(ssid);
    Serial.print(" password: ");
    Serial.println(password);

    while (WiFi.status() != WL_CONNECTED) {
        Serial.println("[wifi] Waiting for connection...");
        delay(1000);
    }

    Serial.print("[wifi] Connected to network with IP: ");
    Serial.println(WiFi.localIP());
}
```

9.3 Pobranie czasu z serwera czasu

W celu pobrania czasu z serwera czasu, wykorzystano bibliotekę `NTPClient.h` oraz `WiFiUdp.h` dostępne w środowisku PlatformIO. Do komunikacji z serwerem czasu służą następujące funkcje:

```
void NTP_Init() {
    timeClient.begin();
    timeClient.setTimeOffset(3600 * 1);
}

void NTP_Update() {
    timeClient.update();
}

uint32_t NTP_GetTime() {
    return timeClient.getEpochTime();
}

uint8_t NTP_GetHour() {
    return timeClient.getHours();
}

uint8_t NTP_GetMinute() {
    return timeClient.getMinutes();
```

```

    }

    uint8_t NTP_GetSecond() {
        return timeClient.getSeconds();
    }
}

```

9.4 Obsługa enkodera rotacyjnego

Moduł enkodera rotacyjnego został zaimplementowany w klasie `Encoder`. Klasa ta zlicza impulsy z enkodera oraz zapisuje kierunek obrotu enkodera. Klasa ta bazuje na mechanizmie callbacków, które są wywoływanie w momencie zmiany stanu pinów enkodera. Inne moduły mogą reagować na zmianę stanu enkodera poprzez zarejestrowanie callbacka. Dostępne są następujące callbacki:

```

std::function<void()> rightCallback;
std::function<void()> leftCallback;
std::function<void()> switchPressCallback;
std::function<void()> switchReleaseCallback;

```

9.5 Reguluja i odczyt napięcia

Komunikacja z potencjometrem cyfrowym odbywa się za pomocą interfejsu I2C. W celu komunikacji z potencjometrem, wykorzystano bibliotekę `Wire.h`. Wykorzystany potencjometr jest 128-stopniowy, z informacji z karty katalogowej [17] wynika, że ramka danych ma się składać z adresu urządzenia (podanego w dokumentacji) oraz wartości korku w zakresie od 0 do 127. Na postawienie tych informacji w klasie `HVConverter` zaimplementowano funkcje do ustawienia wartości potencjometru `sendPotSteps` oraz funkcję do przeliczania zadanego napięcia na wartość potencjometru `voltageToSteps`.

```

#define MAX_VOLTAGE 210
#define MIN_VOLTAGE 134
#define POT_STEPS 127
#define POT_ADDR 0x2F
#define OFFSET 4

void sendPotSteps(uint8_t steps) {
    Wire.beginTransmission(POT_ADDR);
    Wire.write(steps);
    Wire.endTransmission();
}

int voltageToSteps(uint8_t v) {
    return (v + OFFSET - MIN_VOLTAGE) * POT_STEPS / (MAX_VOLTAGE - MIN_VOLTAGE);
}

```

Klasa `HVConverter` ma również zaimplementowaną funkcję do odczytu napięcia na przetwornicy, która bazuje na pomiarze napięcia na dzielниku napięcia za pomocą wbudowanego 12 bitowego

przetwornika ADC. Funkcja ta zwraca wartość napięcia w mV, co następnie jest przeliczane na wartość napięcia na przetwornicy, za pomocą funkcji liniowej.

```
#define R1 10 000
#define R2 1 000 000

uint16_t readVoltage() {
    uint16_t rawValueMilliVolts = analogReadMilliVolts(voltageMeasurePin);
    uint16_t rawVoltageInVolts = rawValueMilliVolts / 1000;
    return rawVoltageInVolts * (R1 + R2) / R2;
}
```

9.6 Odtwarzanie dźwięku

Do odtwarzania dźwięku wykorzystano funkcje `tone` oraz `noTone` dostępne w środowisku, do generowania sygnału o zadanej częstotliwości. Same melody są zdefiniowane w pliku `melody.hpp`, a poszczególne tony w pliku `tones.hpp`. Same melody są zdefiniowane jako tablice tonów oraz tablice długości tonów. W celu lepszej czytelności kodu, stworzono strukturę `Melody`, która przechowuje tablice tonów oraz długości tonów.

```
struct Melody {
    String name;
    std::vector<int> tones;
    std::vector<int> noteDurations;
};

// Przykładowa melodia
Melody happyBirthdayMelody = {
    .name = "Happy Birthday",
    .tones = {
        NOTE_C4, NOTE_C4, NOTE_D4, NOTE_C4, NOTE_F4, NOTE_E4,
        NOTE_C4, NOTE_C4, NOTE_D4, NOTE_C4, NOTE_G4, NOTE_F4,
        NOTE_C4, NOTE_C4, NOTE_C5, NOTE_A4, NOTE_F4, NOTE_E4, NOTE_D4,
        NOTE_AS4, NOTE_AS4, NOTE_A4, NOTE_F4, NOTE_G4, NOTE_F4},
    .noteDurations = {400, 400, 400, 400, 400, 200, 400, 400, 400, 400, 200, 400, 400, 400, 400, 400,
```

Samo odtwarzanie dźwięku dzieje się w klasie `Buzzer`, pozwala ona na ustawienie zadanej melodii, odtworzenie melodii oraz zatrzymanie odtwarzania. Samo odtwarzanie musi być asynchroniczne, aby nie blokować innych funkcjonalności. W tym celu napisano funkcję `update`, która jest wywoływaną w pętli głównej programu.

```
void update() {
    if (!isPlaying) {
        return;
    }
    unsigned long currentTime = millis();
    int adjustedNoteDuration = melody.noteDurations[currentNote];
```

```

        if (currentTime - lastNoteTime >= adjustedNoteDuration * 1.30) {
            noTone(buzzerPin);
            currentNote++;
            if (currentNote >= melody.tones.size()) {
                stop();
                return;
            }
            lastNoteTime = currentTime;
        }
        if (melody.tones[currentNote] != 0 &&
            currentTime - lastNoteTime < adjustedNoteDuration) {
            tone(buzzerPin, melody.tones[currentNote]);
        }
    }
}

```

9.7 Komunikacja z serwerem Home Assistant

Komunikacja z serwerem Home Assistant odbywa się za pomocą protokołu MQTT. W celu komunikacji z serwerem Home Assistant, wykorzystano bibliotekę `PubSubClient.h`. Realizację komunikacji z serwerem Home Assistant zaimplementowano w klasie `MQTT`. Klasa ta pozwala na inicjalizacji połączenia z serwerem, publikowanie wiadomości oraz obsługę callbacków z serwera. Informacje potrzebne do połączenia z serwerem są przechowywane w pliku `secret.hpp`.

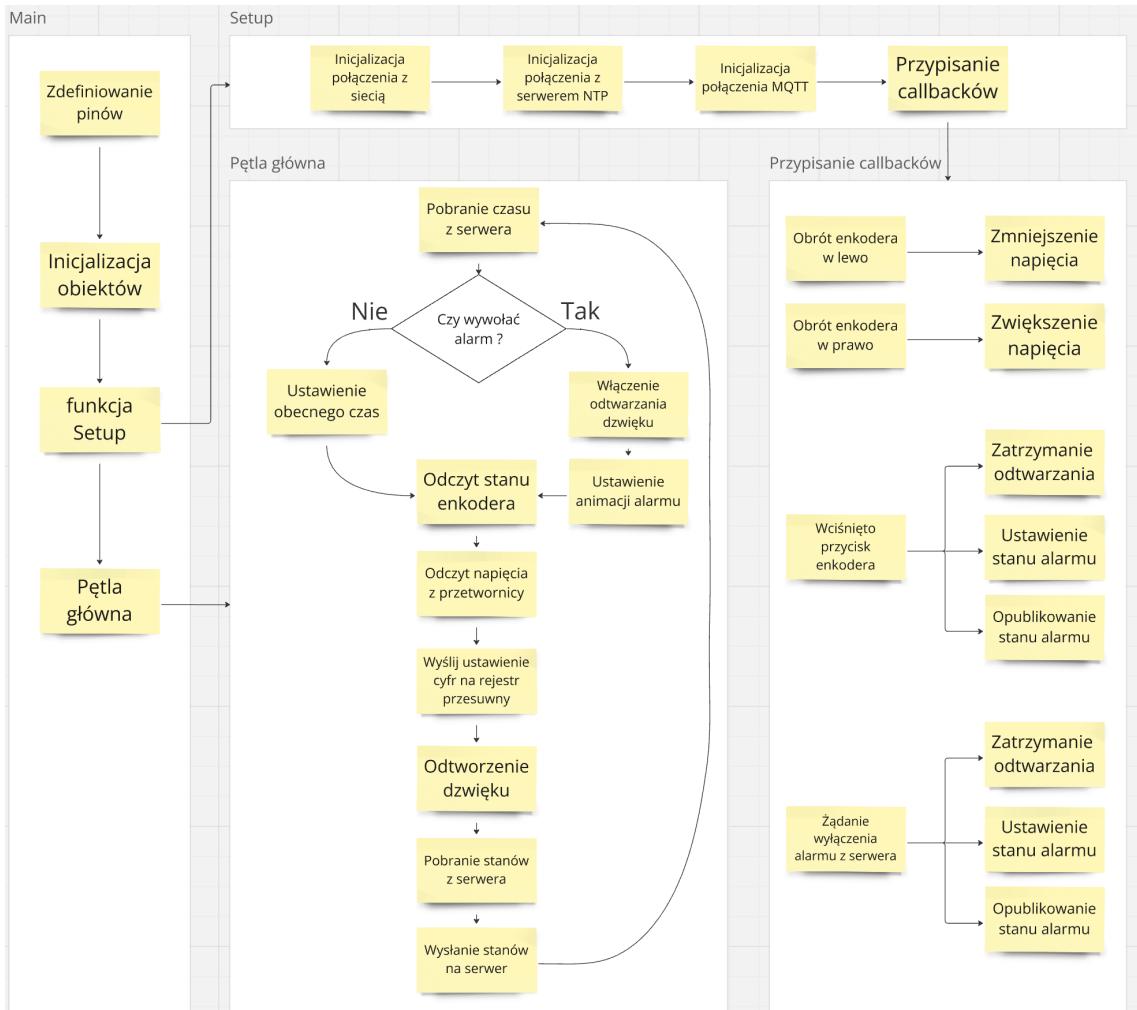
W tej klasie również przechowywane są dane żądanego stanu przez serwer, który jest urządzeniem nadziedzonym. Są to następujące dane:

- `alarmDesiredHour` - żądana godzina alarmu.
- `alarmDesiredMinute` - żądana minuta alarmu.
- `isAlarmEnabled` - czy alarm jest aktywny.
- `desiredBrightness` - żądana jasność wyświetlacza.
- `desiredMelody` - żądana melodia.

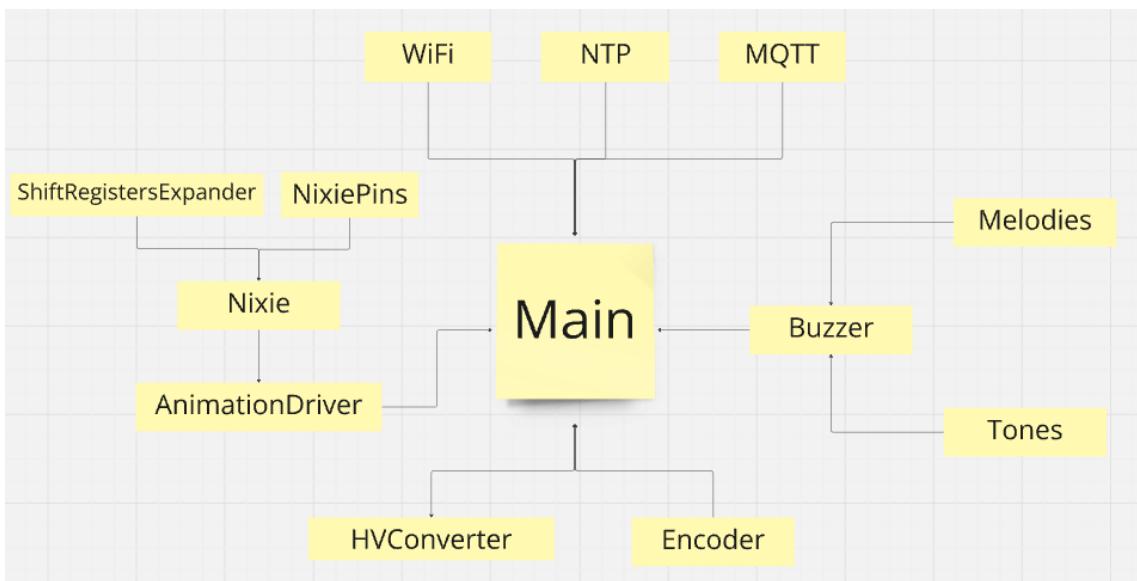
Dodawano również oddzielny topic który informuje zegarek że alarm został wyłączony, tak samo zegarek publikuje informacje do serwera o stanie alarmu. Na podstawie tych zmiennych będą wykonywane działania w pętli głównej programu.

9.8 Główna logika programu

Główna logika programu znajduje się w pliku `main.cpp`. Następuje tu inicjalizacji wszystkich obiektów oraz definicja wszystkich pinów i przekazanie ich do obiektów odpowiednich klas. W funkcji `setup` inicjalizowane są moduły oraz funkcje co muszą być wykonane tylko raz. Następnie w pętli głównej programu wywoływane są funkcje update z poszczególnych klas, które są odpowiedzialne za obsługę poszczególnych funkcjonalności. Schemat działania programu przedstawiony jest na rysunku 9.1. Na schemacie 9.2 przedstawiona jest struktura programu oraz relacje pomiędzy poszczególnymi modułami.



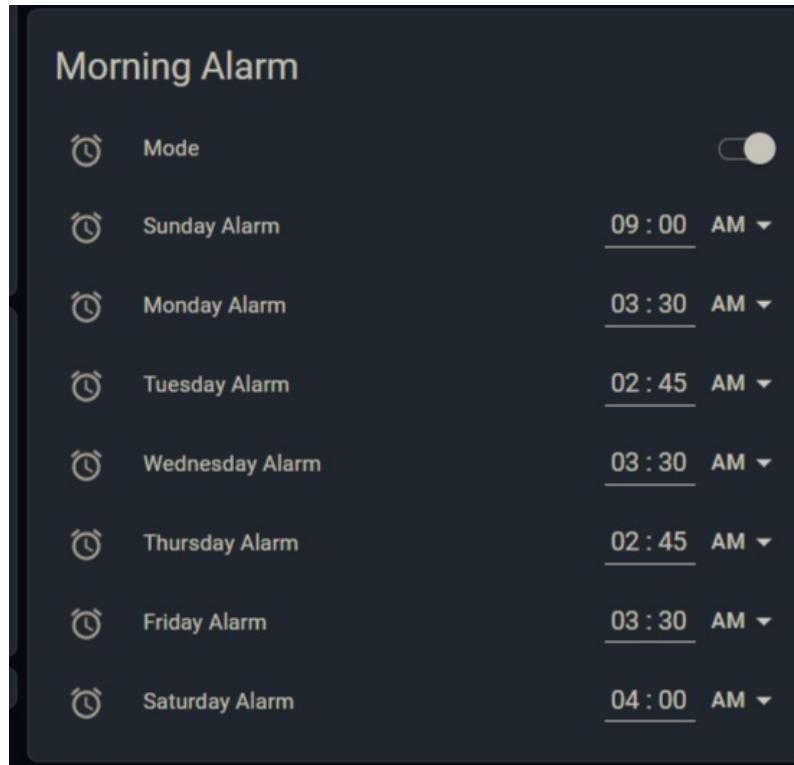
Rysunek 9.1: Schemat działania programu



Rysunek 9.2: Struktura programu

9.9 Interfejs użytkownika

Jako interfejs użytkownika służy widget dodany w aplikacji Home Assistant. Środowisko Home Assistant pozwala na tworzenie własnych widgetów, które są wyświetlane na pulpicie głównym. Widget ten pozwala na ustawienie godziny alarmu na wybrane dni oraz włączenie lub wyłączenie funkcji alarmu. Widok interfejsu użytkownika przedstawiony jest na rysunku 9.3. Pozostałe funkcje takie jak zmiana jasności wyświetlacza i wyłączanie alarmu są dostępne wyłącznie na zegarku. W przyszłości planowane jest dodanie dodatkowych funkcji takich jak zmiana melodi alarmu oraz obsługa paska LED.



Rysunek 9.3: Widget w aplikacji Home Assistant

10 Testowanie układu

W tym rozdziale opisano testowanie poszczególnych modułów oraz całego układu, które pozwoliło na sprawdzenie poprawności działania oraz zidentyfikowanie błędów.

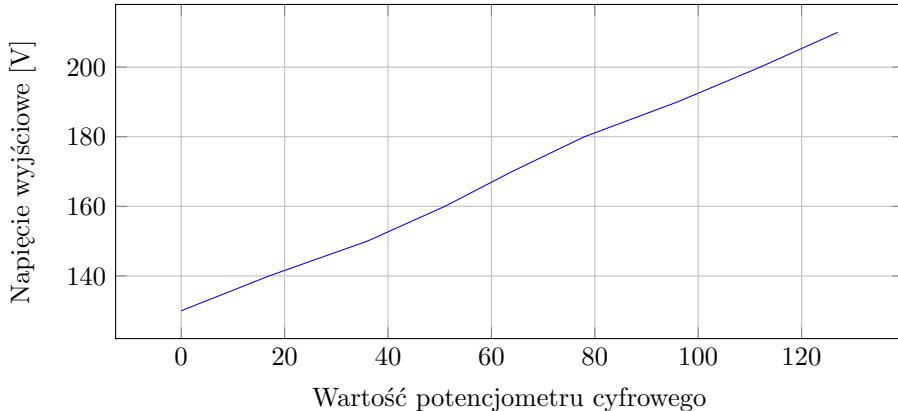
10.1 Testy przetwornicy wysokiego napięcia

Po napisaniu i wgraniu oprogramowania do mikrokontrolera przystąpiono do testów przetwornicy wysokiego napięcia. Sprawdzono działanie regulacji napięcia za pomocą potencjometru cyfrowego. Zakładany zakres napięcia wyjściowego to 130-220 V.

W celu weryfikacji pomierzono zależność napięcia wyjściowego od podanej wartości na potencjometrze cyfrowym. Do pomiarów użyto multymetru cyfrowego UNI-T UT139C. Pomiary zamieszczono w tabeli 10.3.

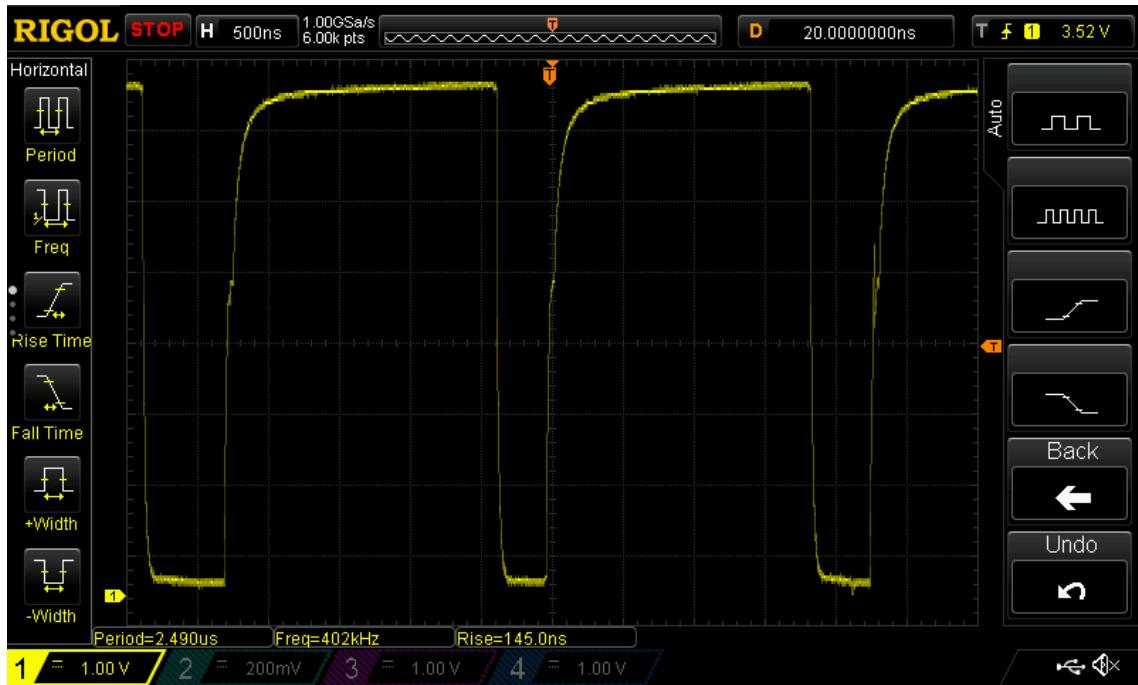
Napięcie wyjściowe [V]	Wartość potencjometru cyfrowego
130	0
140	17
150	36
160	51
170	64
180	78
190	96
200	112
210	127

Tablica 10.3: Zależność napięcia wyjściowego od wartości potencjometru cyfrowego

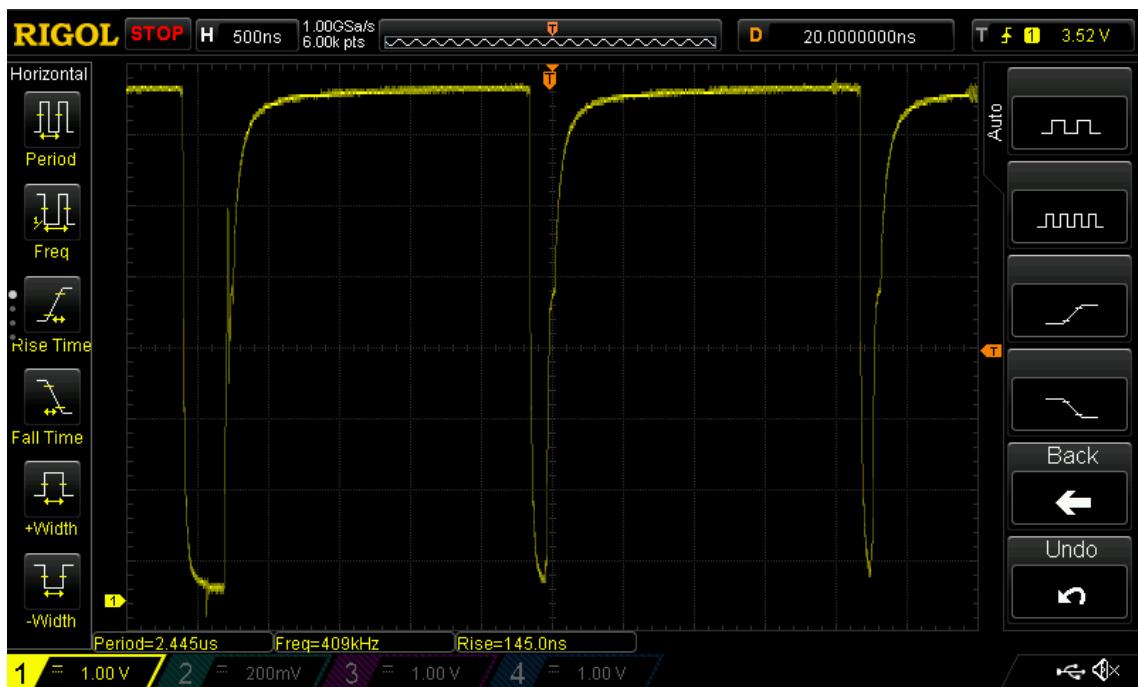


Rysunek 10.1: Wykres zależności napięcia wyjściowego od wartości potencjometru cyfrowego

Na wykresie widać, że zależność jest liniowa, co oznacza, że regulacja napięcia działa poprawnie. Jednak według założeń, napięcie wyjściowe powinno wynosić od 130 V do 220 V. Rozbieżność może wynikać z ograniczenia przetwornicy do generowania dużego wypełnienia sygnału sterującego. By sprawdzić tą hipotezę, zmierzono wypełnienie sygnału sterującego dla napięć wyjściowych 130 V, 210 V. Do pomiarów użyto oscyloskopu Rigol DS1054Z, o paśmie przenoszenia 50Mhz i częstotliwości próbkowania 1 GSa/s, punktem pomiarowym była bramka tranzystora sterującego przetwornicą. Pomiary zamieszczone na rysunku 10.2 i rysunku 10.3.



Rysunek 10.2: Wypełnienie sygnału sterującego przy napięciu wyjściowym 130 V



Rysunek 10.3: Wypełnienie sygnału sterującego przy napięciu wyjściowym 210 V

Jak widać na rysunku 10.3, przy wysokim wypełnieniu sygnału sterującego, tranzystor nie do końca się zamyka, co powoduje, że napięcie wyjściowe jest mniejsze niż zakładane, a do tego tranzystor się nagrzewa i zwiększa się pobór mocy.

Może to wynikać ze źle dobranych elementów kompensujących w układzie przetwornicy. Zmniejszony zakres w jakim można regulować napięcie wyjściowe nie wpłynie na działanie całego układu, ponieważ górny limit napięcia wyjściowego przetwornicy nie jest krytyczny dla działania budzika.

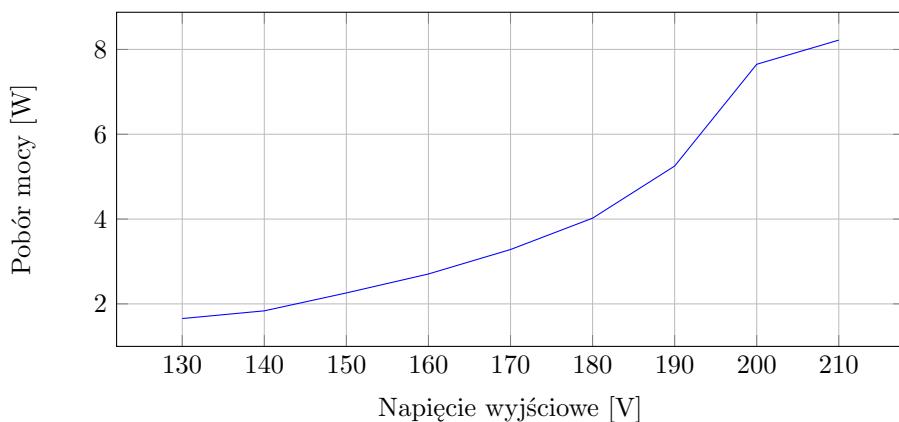
10.2 Testy poboru mocy

Zaczęto od pomiarów mocy pobieranej przez cały układ. Pobór mocy układu zmierzono za pomocą zasilacza laboratoryjnego PS-305DF. Zmierzono pobór mocy układu dla różnych napięć wyjściowych przetwornicy. Pobór mocy zawiera też w sobie pobór mocy pozostałych modułów, ale jest on stały i niezależny od napięcia wyjściowego przetwornicy, więc sam kształt charakterystyki poboru mocy jest zależny od przetwornicy. W podrozdziale 6.3 założono, że pobór mocy układu nie przekroczy 13 W, a sama przetwornica będzie pobierać około 6.5W. Pomiary poboru mocy zamieszczone w tabeli 10.4.

Napięcie wyjściowe [V]	Pobór mocy [W]
130	1.654
140	1.836
150	2.258
160	2.705
170	3.282
180	4.021
190	5.25
200	7.65
210	8.22

Tablica 10.4: Zależność poboru mocy od napięcia wyjściowego

Wykreślono zależność poboru mocy od napięcia wyjściowego na wykresie 10.4.



Rysunek 10.4: Wykres zależności poboru mocy od napięcia wyjściowego

Po dokonaniu pomiaru poboru mocy, zaobserwowano, że pobór mocy jest większy niż zakładano i wynosi ponad 8 W przy napięciu wyjściowym 210 V. Na charakterystyce pobór mocy okazał się nieliniowy przy wyższych napięciach wyjściowych. Wynika to ze spadku wydajności przetwornicy przy wyższych napięciach, z powodu niezamykania się tranzystora przełączającego. Jednak układ domyślnie ma pracować na niższych napięciach, więc zdecydowano się na programowe ograniczenie napięcia wyjściowego przetwornicy do 190 V, gdzie układ pracuje z mocą zakładaną na etapie projektowania. Dolny limit napięcia wyjściowego również został ograniczony do 140 V, ponieważ przy 130 V lampy nie zapalają się w pełni, co może prowadzić do ich uszkodzenia. Jako domyślne napięcie wyjściowe wybrano 150 V, przy którym pobór mocy wynosi 2.358 W, co jest akcepto-

walnym wynikiem. Porównując ten pobór mocy z innymi zegarkami dostępnymi na rynku, jest on porównywalny.

Poza zmniejszonym zakresem pracy przetwornicy wszystkie pozostałe założenia projektowe zostały spełnione. Układ pobiera czas z serwera czasu, wyświetla go na lampach Nixie i działa jako budzik. Wszystkie funkcje działają poprawnie.

11 Podsumowanie

Celem niniejszej pracy inżynierskiej było opracowanie i wykonanie budzika synchronizowanego przez Wi-Fi z wykorzystaniem lamp Nixie.

W pracy omówiono zasadę działania lamp Nixie, mikrokontrolera ESP32-S3 oraz różnych serwerów czasu. Wyjaśniono, jakie są wady i zalety lamp Nixie, jakie są ich zastosowania. Przeanalizowano różne metody sterowania wyświetlacząmi Nixie w celu wybrania najlepszej dla projektu. Wyjaśniono możliwości mikrokontrolera ESP32-S3 w zastosowaniach IoT oraz jego środowisko programistyczne. Przedstawiono różne serwery czasu, ich zasady działania, wady i zalety. Szczegółowo opisano protokół NTP, który jest wykorzystywany w projekcie.

Na podstawie karty projektu i analizy wymagań oraz wykonania poszukiwań przedstawiono założenia projektowe. W oparciu o nie powstała ogólna koncepcja układu, a z niej powstał szczegółowy schemat wykonania projektu oraz jego podział na moduły. Następnie na podstawie testu zakupionych lamp Nixie określono konkretne parametry, które muszą spełniać wybrane elementy. Bazując na tym podziale, przystąpiono do wyboru elementów, które będą wykorzystane w projekcie. Następnie stworzono schematy elektryczne poszczególnych modułów i opisano sposób ich połączenia z uwzględnieniem niezbędnych obliczeń.

W kolejnym etapie przystąpiono do implementacji projektu. Zaprojektowano i wykonano płytę PCB z uwzględnieniem aspektów wizualnych oraz funkcjonalnych. Następnie zmontowano układ i przystąpiono do testów. W trakcie testów napotkano problem z przetwornicą wysokiego napięcia, która nie osiągnęła zakładanego przedziału regulacji i miała mniejszą sprawność niż założono, co powodowało nagrzewanie się przy górnej granicy zakresu. Problem został rozwiązyany poprzez zwężenie przedziału regulacji napięcia wyjściowego. Pozostałe testy przebiegły pomyślnie, a układ działał zgodnie z założeniami.

Dalsze prace nad opracowanym urządzeniem powinny obejmować przede wszystkim przeprojektowanie układu przetwornicy wysokiego napięcia w celu uniknięcia problemów z nagrzewaniem się i poszerzenia zakresu regulacji. Można również dodać dodatkowe funkcje interfejsu użytkownika, takie jak obsługa paska LED, regulacja napięcia wyjściowego z poziomu aplikacji, dodanie możliwości ustawienia alarmu w trybie offline. Można również wykonać obudowę dla układu, aby zabezpieczyć go przed uszkodzeniami mechanicznymi oraz kurzem.

Bibliografia

- [1] *Krótką historią pomiaru czasu, czyli od zegara słonecznego do zegara atomowego.* Jarosław Koperski Instytut Fizyki UJ. 2003. URL: <https://foton.if.uj.edu.pl/documents/12579485/0fafdf2cb-ad07-4e6f-bf1b-88301daf8e4d>.
- [2] *Zegar Nixie - nostalgiczna elegancja.* 4/2024 Kwiecień (16). Zrozumieć Elektronikę z Piotrem Góreckim. Kw. 2024, s. 23–26. URL: <https://piotr-gorecki.pl/wp-content/uploads/2024/03/ZE2404.pdf>.
- [3] *Lampa cyfrowa. Schemat elektryczny lampy NIXIE.* Wikipedia. 2023. URL: https://pl.wikipedia.org/wiki/Lampa_cyfrowa.
- [4] *Kompendium wiedzy o lampach Nixie.* Rev. 2. rafalbartoszak. Kw. 2019. URL: <https://rafalbartoszak.pl/kompendium-wiedzy-o-lampach-nixie/>.
- [5] *ESP32-S3 Series Datasheet.* v1.9. Espressif Systems. URL: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf.
- [6] *ESP32-S3: Which Pins Should I Use?* Chris Greening. Paź. 2023. URL: <https://www.atomic14.com/2023/11/21/esp32-s3-pins>.
- [7] *eSezam 1.0/Synchronizacja w telekomunikacji/Podręcznik 2. Synchronizacja czasu.* Ośrodek Kształcenia na Odległość Politechniki Warszawskiej. URL: <https://esezam.okno.edu.pl/mod/book/view.php?id=76&chapterid=1632>.
- [8] *Network Time Protocol. Struktura warstw STRATUM 0-15.* Wikipedia. 2024. URL: https://pl.wikipedia.org/wiki/Network_Time_Protocol.
- [9] *RFT tube data book and translation.* Z570M/Z5700M. RFT electronic. URL: <https://www.tube-tester.com/sites/nixie/data/z5730m/z5730m.html>.
- [10] *HV5530 32-Channel Serial-to-Parallel Converter With Open Drain Outputs.* DS20005851A. Microchip. Paź. 2017. URL: <https://ww1.microchip.com/downloads/en/DeviceDoc/20005851A.pdf>.
- [11] *CD40109B cmos quad low-to-high voltage shifter.* SCHS380. Texas Instruments. Czer. 2010. URL: <https://www.ti.com/lit/ds/symlink/cd40109b-q1.pdf?ts=1732708893044>.
- [12] *LM3488/-Q1 Automotive High-Efficiency Controller for Boost, SEPIC and Fly-Back DC-DC Converters.* SNVS089O. Texas Instruments. Lip. 2015. URL: https://www.ti.com/lit/ds/symlink/lm3488.pdf?ts=1732111856668&ref_url=https%253A%252F%252Fwww.ti.com%252Fpower-management%252Facdc-dcdc-controllers%252Fproducts.html.
- [13] *Ceramic capacitor.* Wrz. 2024. URL: https://en.wikipedia.org/wiki/Ceramic_capacitor#Voltage_dependence_of_capacitance.
- [14] *TPS56x219A 4.5-V to 17-V Input, 2-A, 3-A Synchronous Step-Down Voltage Regulator in 8 Pin SOT-23.* SLVSDT2. Texas Instruments. Paź. 2016. URL: <https://www.ti.com/lit/ds/symlink/tps563219a.pdf?ts=1732740548889>.
- [15] *CEM-1203(42) magnetic buzzer.* CUI INC. List. 2006. URL: [https://componentsearchengine.com/Datasheets/1/CEM-1203\(42\).pdf](https://componentsearchengine.com/Datasheets/1/CEM-1203(42).pdf).
- [16] *PEC12R - 12 mm Incremental Encoder datasheet.* Bourns. Maj 2018. URL: <https://www.mouser.pl/datasheet/2/54/PEC12R-777795.pdf>.

- [17] *MCP4017/18/19 7-Bit Single I²C™ Digital POT with Volatile Memory in SC70*. Microchip. Mar. 2009. URL: <https://datasheet.datasheetarchive.com/originals/distributors/Datasheets-DGA14/624353.pdf>.
- [18] *AP7363 1.5A low quiescent current, fast transient ultra-low dropout linear regulator*. DS35059. Rev. 10 - 2. Diodes Incorporated. Paź. 2021. URL: <https://www.diodes.com/assets/Datasheets/AP7363.pdf>.

Spis rysunków

2.1	Schemat elektryczny lampy nixie ze wspólną anodą [3]	10
3.1	Rozkład pinów mikrokontrolera ESP32-S3 [6]	14
4.1	Struktura serwerów czasu w protokole NTP [8]	18
5.1	Ogólna koncepcja układu	20
5.2	Ogólny schemat komunikacji z serwerem	21
6.1	Schemat blokowy projektu	24
6.2	Prototyp układu z lampą Nixie przy napięciu zasilania 150 V	26
6.3	Prototyp układu z lampą Nixie przy napięciu zasilania 220 V	26
7.1	Schemat elektryczny modułu sterowania lampami	29
7.2	Schemat układu z karty katalogowej [12]	30
7.3	Spadek pojemności kondensatora ceramicznego wraz ze wzrostem napięcia [13]	32
7.4	Schemat elektryczny przetwornicy 12 V na wysokie napięcie	34
7.5	Tabela doboru komponentów z noty katalogowej [14]	35
7.6	Typowe połączenie układu TPS563219ADDFR [14]	36
7.7	Schemat elektryczny modułu przetwornicy z 12 V na 5 V	36
7.8	Schemat elektryczny złącza zasilania	37
7.9	Schemat elektryczny złącza USB-C	38
7.10	Schemat zalecany przez producenta [15]	39
7.11	Schemat elektryczny podłączenia buzzera	40
7.12	Schemat filtrów zalecany przez producenta [16]	41
7.13	Schemat elektryczny enkodera obrotowego	41
7.14	Schemat podłączenia LDO zalecany przez producenta [18]	42
7.15	Schemat elektryczny układu LDO	42
7.16	Schemat elektryczny złącza UART	43
7.17	Schemat złącza paska LED	44
7.18	Schemat podłączenia ESP32-S3	45
8.1	Górna warstwa płytka drukowanej	46
8.2	Dolna warstwa płytka drukowanej	46
8.3	Płytki drukowane - widok od góry	47
8.4	Płytki drukowane - widok od spodu	47
8.5	Zmontowana płytka drukowana - widok od góry	48
8.6	Zmontowana płytka drukowana - widok od spodu	48
8.7	Zmontowany układ	49
9.1	Schemat działania programu	56
9.2	Struktura programu	56
9.3	Widget w aplikacji Home Assistant	57
10.1	Wykres zależności napięcia wyjściowego od wartości potencjometru cyfrowego	58
10.2	Wypełnienie sygnału sterującego przy napięciu wyjściowym 130 V	59
10.3	Wypełnienie sygnału sterującego przy napięciu wyjściowym 210 V	59
10.4	Wykres zależności poboru mocy od napięcia wyjściowego	60

Spis tabelic

2.1 Tabela opłacalności sposobów sterowania lampami nixie	12
4.2 NTP – format komunikatu [8]	18
10.3 Zależność napięcia wyjściowego od wartości potencjometru cyfrowego	58
10.4 Zależność poboru mocy od napięcia wyjściowego	60