



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

Główym celem jaki został postawiony jest zrealizowanie budzika który synchronizuje się z wykrozystaniem zewnętrznego serwera czasu. ctd.

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 main goal was to create an alarm clock that synchronizes with the use of using an external time server. ctd.

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	12
2.5 Sytuacja na rynku	12
3 Mikrokontroler ESP32-S3	13
3.1 Moduł RTC	13
3.2 Kontroler USB/JTAG	13
3.3 GPIO	13
3.4 Zasilanie	14
4 Serwery czasu	15
4.1 Protokoły synchronizacji czasu	15
4.2 Struktura serwerów w protokole NTP	16
4.3 Zasada działania protokołu NTP	17
5 Koncepcja układu	19
5.1 Założenia projektowe	19
5.2 Ogólny schemat komunikacji z serwerem	20
6 Realizacja	21
6.1 Szczegółowa koncepcja układu	21
6.1.1 Sterowanie lampami nixie	21
6.1.2 Mikrokontroler	21
6.1.3 Źródło dźwięku	21
6.1.4 Pasek LED	22
6.1.5 Interfejs użytkownika	22
6.1.6 Zasilanie	22
6.1.7 Złącza	23
6.1.8 Obudowa	23
6.1.9 Szczegółowy schemat projektu	23
6.2 Test lamp	24
6.3 Obliczenia mocy	26
7 Realizacja modułów	27
7.1 Sterowanie lampami Nixie	27
7.1.1 Dobór rejestrów	27
7.1.2 Sterowanie rejestrów	27

7.1.3	Sterowanie kropkami dziesiętnymi	28
7.1.4	Dobór rezystorów	28
7.2	Przetwornica 12V na HV	29
7.2.1	Założenia projektowe	29
7.2.2	Wybór układu scalonego	29
7.2.3	Dobór cewki	29
7.2.4	Dobór kondensatorów	30
7.2.5	Dobór diody	31
7.2.6	Dobór tranzystora	31
7.2.7	Ustawienie napięcia wyjściowego	32
7.2.8	Dobór rezystora ograniczającego prąd	33
7.2.9	Schemat	33
7.3	Przetwornica 12V na 5V	34
7.3.1	Założenia projektowe	34
7.3.2	Wybór układu scalonego	34
7.3.3	Dobór komponentów	34
7.3.4	Schemat	35
7.4	Złącze zasilania	36
7.4.1	Dobór złącza	36
7.4.2	Opis podłączenia	36
7.4.3	Zabezpieczenia ESD	36
7.4.4	Schemat	36
7.5	Złącze do programowania	37
7.5.1	Dobór złącza	37
7.5.2	Opis podłączenia	37
7.5.3	Zabezpieczenia ESD	37
7.5.4	Schemat	37
7.6	Buzzer	38
7.7	Encoder	40
7.8	LDO 5V na 3.3V	41
7.9	Złącze do debugowania	42
7.10	Złącze do paska LED	43
7.11	Mikrokontroler ESP32-S3	44
8	Projekt i montaż płytki drukowanej	45
8.1	Projekt płytki drukowanej	45
8.2	Montaż i uruchomienie układu	46
8.3	Uruchomienie układu	47
9	Oprogramowanie	49
9.1	Sterowanie lampami nixie	49
9.2	Połączenie z siecią Wi-Fi	50
9.3	Pobranie czasu z serwera czasu	51
9.4	Obsługa enkodera rotacyjnego	52
9.5	Regulacja i odczyt napięcia	52
9.6	Odtwarzanie dźwięku	53

9.7 Komunikacja z serwerem Home Assistant	54
9.8 Główna logika programu	54
10 Testowanie układu	56
10.1 Testy Przetwornicy HV	56
10.2 Testy układu sterowania	56
10.3 Testy enkodera i buzzera	56
11 Podsumowanie	57
Bibliografia	59
Spis rysunków	60
Spis tabel	61

Wykaz ważniejszych oznaczeń i skrótów

RTC - Real Time Clock, zegar czasu rzeczywistego

NTP - Network Time Protocol, protokół czasu sieciowego

Wi-Fi - Wireless Fidelity, bezprzewodowa łączność

HV - High Voltage, wysokie napięcie

1 Wstęp i cel pracy

1.1 Wstęp

Główym celem jaki został postawiony jest zrealizowanie budzika który synchronizuje się z wykorzystaniem zewnętrznego serwera czasu. ctd.

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 modulem RTC wbudowanym w mikrokontroler.

Projekt zakłada stałe połączenie z siecią Wi-Fi, ale istnieć będzie możliwość działania w trybie offline, z wykorzystaniem modułu RTC wbudowanego w mikrokontroler, chociaż w przypadku tego trybu dokładność będzie mniejsza i nie będzie możliwe ustawienie alarmu.

Urządzenie będzie wykorzystywać aplikację jako interfejs użytkownika, przy zachowaniu części ustawień bezpośrednio na urządzeniu.

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, 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.

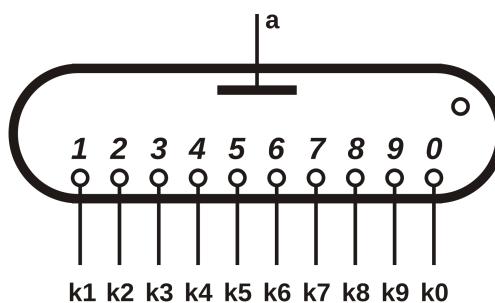
2.1 Zasada działania

Zjawisko zachodzące w lampach zwane jest jako wyładowanie gazowe.[1] 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 nie pełne pokrycie katody warstwą gazową, co powoduje zanikanie wyświetlania cyfr, a czasami skutkuje zanikiem wyświetlania całkowicie.



Rysunek 2.1: Schemat elektryczny lampy nixie ze wspólną anodą[2]

Najczęściej spotykaną konfiguracją podłączenia lampy jest wspólna anoda, gdzie katody są sterowane sygnałem, a anoda jest podłączona do zasilania przez rezystor ograniczający prąd. W przypadku lampy nixie, napięcie zasilanie wynosi od 150 V do 200 V.

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 kropkowania** - zjawisko polegające na wyświetaniu kropełek w miejscach gdzie nie powinny się one znajdować.
- **Zjawisko spalania** - zjawisko polegające na spalaniu się katod, spowodowane zbyt dużym prądem płynącym przez katodę.
- **Zjawisko cienia** - zjawisko polegające na wyświetlanie 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 w szczególności w kontekście lamp używanych w zegarach nixie. Tylko parzyste wyświetlacze działają w optymalny sposób, 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.

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 na przykład poprzez animacje, gdy zmienia się cyfra minut. Można również w godzinach nocnych przez jakiś czas wyświetlać cyklicznie wszystkie cyfry, aby zapobiec zatruciu katodowemu.

W celu zwiększenia trwałości lamp można również tak sterować lampami, aby 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 pinów 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[3]:

- Sterowanie bezpośrednie - każda lampa ma swoje wejście i jest sterowana osobno za pomocą tranzystora HV połączonego z mikrokontrolerem. Wadą jest konieczność posiadania wielu pinów GPIO, co jest nieoptymalne. Sumarycznie wymaga to 66 pinów GPIO oraz 66 tranzystorów HV. Można by w tym porządku zastosować multipleksery co zmniejszyło by ilość wymaganych pinów GPIO do 16, ale zwiększa to skomplikowanie układu.
- Multiplexing - wszystkie lampy są podłączone do jednego drivera, który wybiera katode i załącza odpowiednią anodę. Wymaga to mniej pinów GPIO bo tylko 10, ale multipleksacja powoduje szybsze zużycie lamp i pojawia się artefaktów.
- 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.
- Rejestr przesuwny HV - najbardziej optymalne rozwiązanie, wymaga tylko 3 pinów GPIO. Rejestry HV są ciężko dostępne i dość drogie, muszą też posiadać zatrzask. Wymagane jest również by rejesty miały wyjścia typu Open Drain.
- Połączenie rejestrów przesuwnych z driverami - połączenie rejestrów przesuwnych HV z dedykowanymi driverami, pozwala na zastosowania rejestrów przesuwnego dla niskiego napięcia. Ta kombinacja również wymaga 3 pinów GPIO, przy rejestrze 32 bitowym i 6 driverach. Powoduje jednak to większe skomplikowanie układu oraz układ taki zajmuje więcej miejsca.

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 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 nie wiadomy stan lamp, ponieważ są to produkty one używane.

3 Mikrokontroler ESP32-S3

ESP32-S3 to mikrokontroler firmy Espressif Systems, który jest następcą popularnego ESP32. Jedenego z najpopularniejszych mikrokontrolerów z modułem WiFi, wykorzystanego w wielu projektach IoT[4]. 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
- USB/JTAG kontroler
- ROM: 384 KB
- SRAM: 512 KB
- Wbudowany moduł RTC

3.1 Moduł RTC

RTC (Real Time Clock) to moduł czasu rzeczywistego, który pozwala na śledzenie aktualnego czasu, daty oraz dnia tygodnia. ESP32-S3 posiada wbudowany taki moduł, charakteryzuje się on 16kB pamięci SRAM, wynika z tego nie może on przechowywać daty i czasu w przypadku braku zasilania.

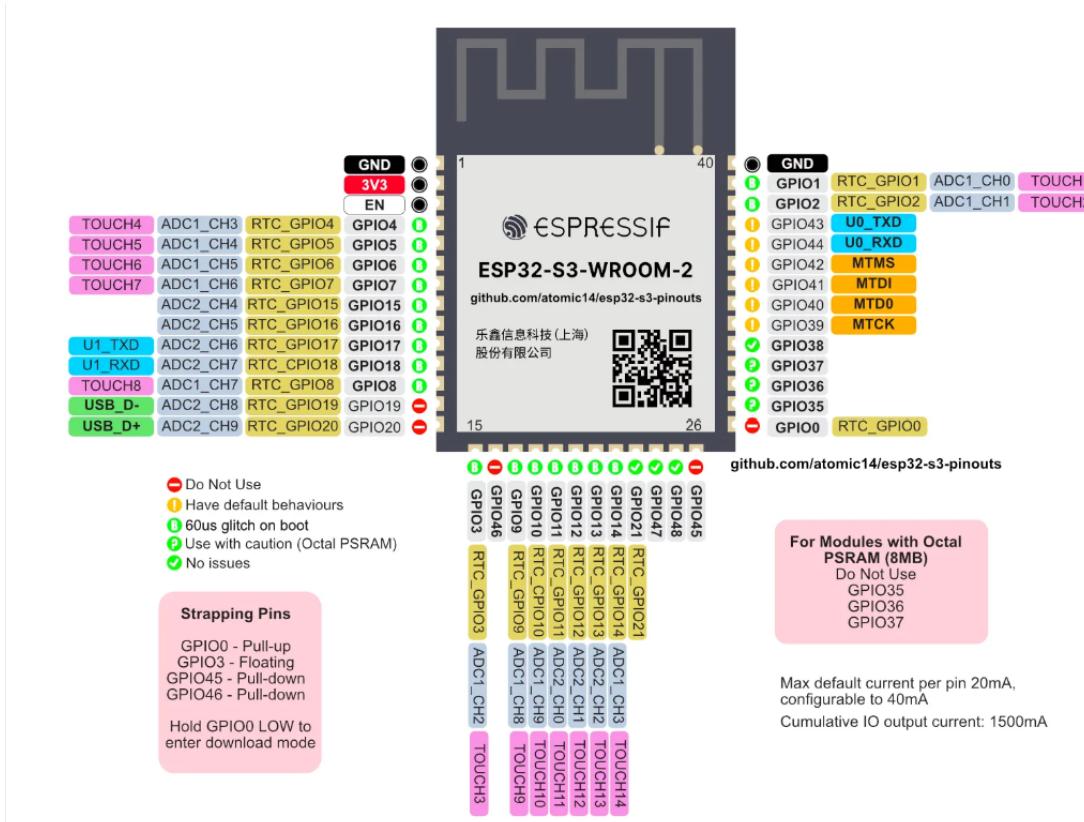
Sam moduł RTC nie jest bardzo dokładny, dlatego zaleca się synchronizację czasu z zewnętrznym serwerem. Istnieje dużo bibliotek do realizacji tego zadania z wykorzystaniem Arduino Framework.

3.2 Kontroler USB/JTAG

ESP32-S3 posiada wbudowany kontroler USB/JTAG, pozwalający programować układ bez użycia zewnętrznego programatora. Piny 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).

3.3 GPIO

Układ posiada wiele GPIO ogólnego przeznaczenia, 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 pinów GPIO. Posiada on również 20 pinów które mogą obsługiwać wejście analogowe.



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

3.4 Zasilanie

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

Układ można wprowadzić w dwa tryby uśpienia:

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

4 Serwery czasu

Serwerem czasu nazywamy serwer komputerowy, pobierający czas z zewnętrznych źródeł i udostępniający go dla innych urządzeń w sieci[7]. 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 rubinowe,
- 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, 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 - wykorzystuje odbiorniki GPS do synchronizacji zegarów na różnych serwerach. Zapewnia bardzo dokładne dane czasu. 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 implementacje.
- 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. Jedną 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 nie zalecany 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

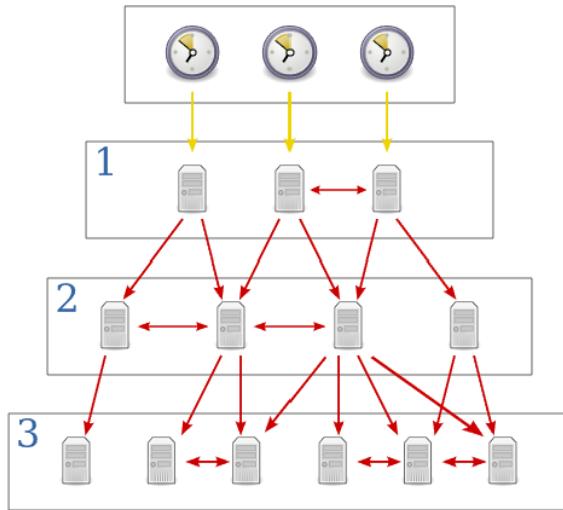
Synchronizacji NTP wykorzystuje uporządkowaną strukturę gałęziową STRATUM[8]. Zasada hierarchii wygląda następująco: urządzenia warstwy STRATUM N mogą być serwerami czasu dla warstwy STRATUM N+1, ale nie na odwrót. Komputery STRATUM N mogą być również klientami urządzeń warstwy STRATUM N-1 itd.

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, ilość 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 za pomocą 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.



Rysunek 4.1: Struktura serwerów czasu w protokole NTP[9]

W przypadku zegara nixie poziom STRATUM nie ma większego znaczenia, ponieważ zegar nie wymaga bardzo precyzyjnego czasu, chociaż oczywiście zależy, jak precyzyjny czas będzie wyświetlane, ale w przypadku zegara na 6 cyfrach, różnica w czasie rzędu kilku milisekund nie będzieauważalna.

4.3 Zasada działania protokołu NTP

NTP różni się od typowego protokołu komunikacyjnego. Nie transmituje on bowiem absolutnej wartości czasu, lecz przekazuje informacje o opóźnieniach i korelacjach czasowych w regularnych odstępach czasu, jakie zachodzą w sieci TCP/IP. Protokół wyróżnia się 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.

Sam pakiet NTP opisany jest w następujący sposób:

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

Tablica 2: NTP – format komunikatu

- 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

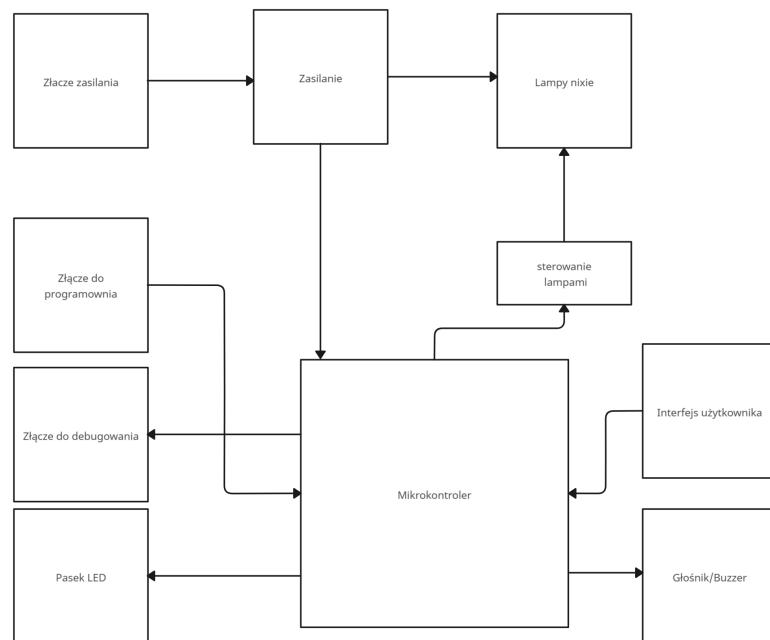
Etap koncepcyjny został podzielony na dwa etapy: koncepcję układu oraz realizację. W tym rozdziale znajduje się ogólna koncepcja działania układu, która nie jest związana z konkretnymi elementami sprzętowymi, a jedynie z funkcjonalnościami, które mają być zrealizowane.

5.1 Założenia projektowe

Zgodnie z celem pracy, określono następujące założenia projektowe:

- Funkcjonalność ustawiania godziny budzika będzie realizowana przez zewnętrzny serwer.
- Na wyświetlaczu Nixie będą wyświetlane godziny, minuty, sekundy.
- Od spodu obudowy będą umieszczone paski LED, które będą podświetlały obudowę i będą wyświetlane animacje podczas alarmu.
- Alarm będzie sygnalizowany dźwiękiem oraz miganiem pasków LED.
- Wyłączanie alarmu będzie możliwe poprzez przycisk na obudowie, aplikację mobilną lub zewnętrzny przycisk połączony z serwerem.
- W przypadku braku połączenia z serwerem, czas będzie mierzony przez RTC wbudowany w mikrokontroler.
- Programowa oraz manualna regulacja jasności wyświetlacza Nixie oraz pasków LED.

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

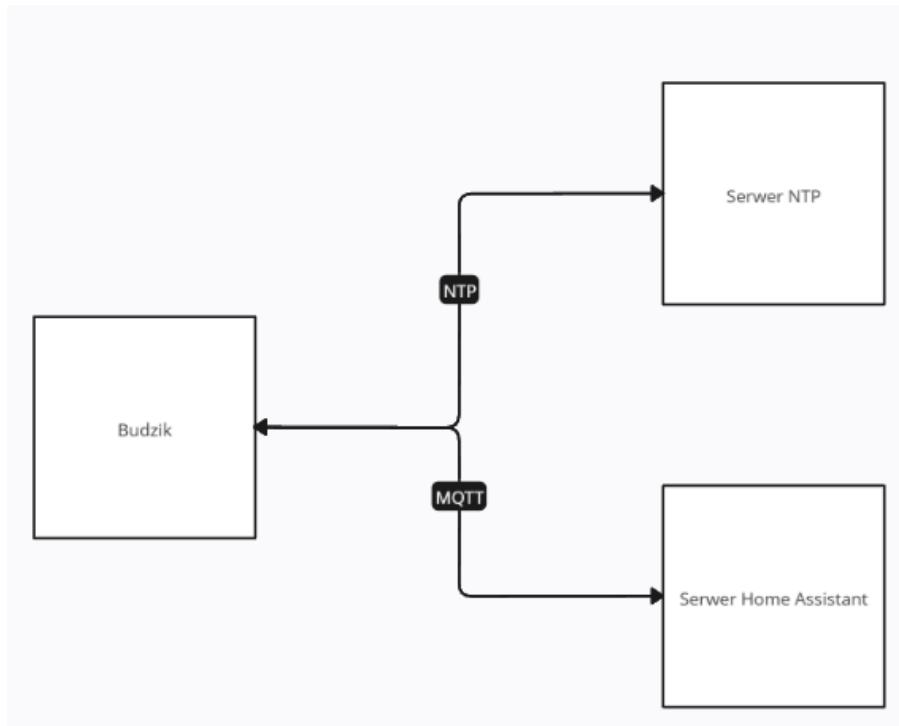


Rysunek 5.1: Ogólna koncepcja układu

Sekcja opisana jako *zasilanie* będzie odpowiedzialna za zasilanie wszystkich elementów układu, w tym lamp Nixie, paska LED, mikrokontrolera oraz głośnika, więc będzie wymagane rozbicie jej na kilka podsekcji, ponieważ będą potrzebne różne napięcia zasilania. Lampy Nixie potrzebują zasilania wysokim napięciem, natomiast pozostałe elementy potrzebują zdecydowanie niższych napięć. 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 interfejsem 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 najbardziej 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 bardzo popularnym protokołem w IoT, co pozwoli na łatwe rozbudowanie funkcjonalności. Połączenia te przedstawione są 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 wymaga rozbicia na poszczególne sekcje układu, ponieważ wybory jednego elementu wpływają na wybór kolejnych elementów.

6.1.1 Sterowanie lampami nixie

Kluczowym jest wybór sterownia lampami nixie, ponieważ na podstawie tego wyboru zostanie zaprojektowany reszta układu. Zgodnie z analizą przeprowadzoną w podrozdziale 2.4, zdecydowano się na sterowanie lampami za pomocą rejestrów przesuwnych HV. Zastosowanie tego rozwiązania pozwala na zredukowanie ilości potrzebnych pinów mikrokontrolera do sterowania lampami oraz jest to rozwiązanie proste w implementacji.

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

Wynika z tego, że potrzebne są 2 rejesty przesuwne HV do sterowania lamp i neonówkami oraz 6 tranzystorów HV do sterowania kropkami dziesiętnymi. Do sterowania rejestrami prawdopodobnie będzie potrzebny konwerter poziomów logicznych, ponieważ mikrokontroler ESP32-S3 pracuje na 3.3V, a rejesty 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 dziesiętnym, 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 dużo bibliotek i przykładów.

6.1.3 Źródło dźwięku

Jako źródło dźwięku wybrano głośnik piezoelektryczny, który jest prosty w implementacji i nie wymaga dodatkowego wzmacniacza, do tego jest mały i tani. Wystarczy jedynie podłączyć go do pinu GPIO mikrokontrolera i za pomocą PWM można 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 i 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 PWM można sterować jasnością.

6.1.5 Interfejs użytkownika

Interfejs użytkownika będzie składał się z enkodera z przyciskiem, który będzie służył do regulacji jasności, a przycisk do wyłączania alarmu. Enkoder 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 jest możliwe zaprojektowanie przetwornicy z napięciem zasilania 5V co by pozwoliło na użycie zasilacza USB. 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. Natomiast wymagało to zastosowania zasilania 12V, co uniemożliwia użycie tylko złącza USB do zasilania układu, natomiast znaczaco upraszcza projektowanie układu.

Wybrano więc przetwornicę typu boost, która będzie zasilana z zasilacza 12V, a wyjście będzie podłączone do anod lamp nixie.

Do tego będzie potrzebne zasilanie 5V dla paska LED oraz 3.3V dla mikrokontrolera. Moduł zasilania 5V będzie zasilał pasek LED, który potrafi pobrać większy prąd, to ze względu na zachowanie wysokiej efektywności, zdecydowano się na przetwornicę typu buck. Moduł zasilania 3.3V będzie zasilaniem mikrokontrolera, więc wystarczy zastosować stabilizator liniowy.

Można, więc podzielić zasilanie na 3 pod moduły:

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

6.1.7 Złącza

W związku z tym, że będzie potrzebne zasilanie z zasilacza 12V, zdecydowano się na zastosowanie złącza DC jack, które jest powszechnie stosowane w zasilaczach. Do zasilania mikrokontrolera oraz programowania, zdecydowano się na złącze USB-C, które jest obecnie najbardziej uniwersalnym rozwiązaniem. Zostanie również dodane złącze Goldpin, które będzie służyło jako złącze debugowe, co pozwoli na łatwe debugowanie układu, na etapie tworzenia oprogramowania.

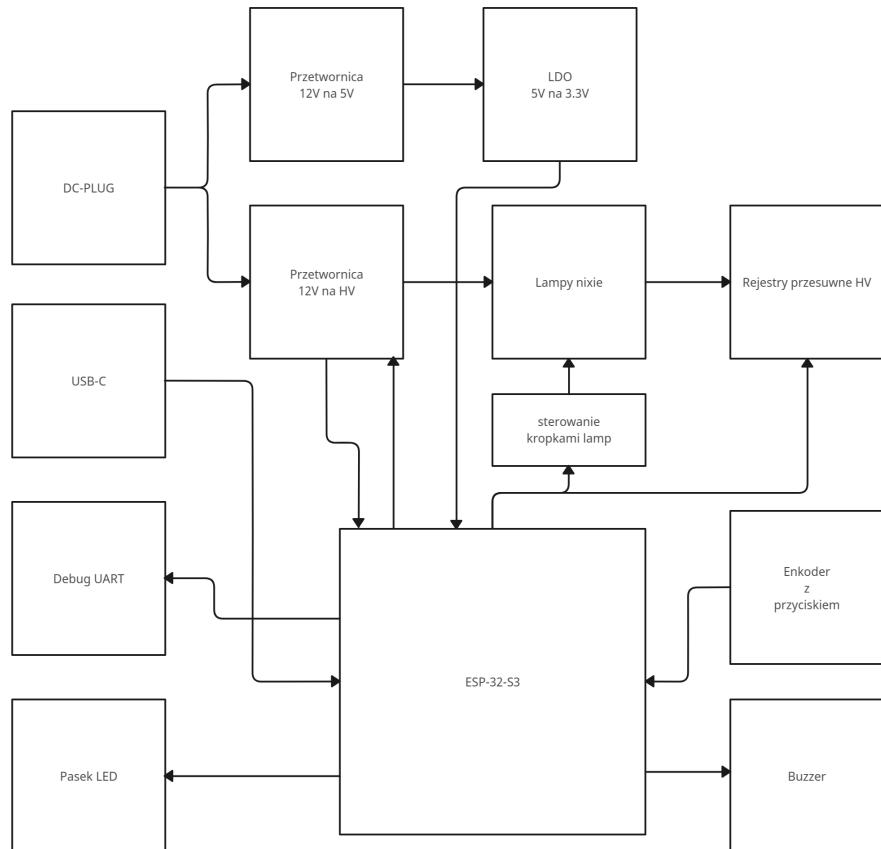
Urządzenie będzie posiadać 3 złącza:

- Złącze USB-C do programowania mikrokontrolera
- Złącze DC do zasilania układu
- Złącze Goldpin jako złącze debugowe

6.1.8 Obudowa

By zachować wygląd retro, zdecydowano się na zastosowanie obudowy drewnianej. Od góry będzie znajdowało się szkło akrylowe, które będzie służyło jako osłona przed kurzem i jednocześnie będą widoczne elementy elektryczne.

6.1.9 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[10] 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 HV z regulowanym napięciem wyjściowym zakupiony w sklepie internetowym.

Zakupiona przetwornica HV ma następujące parametry:

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

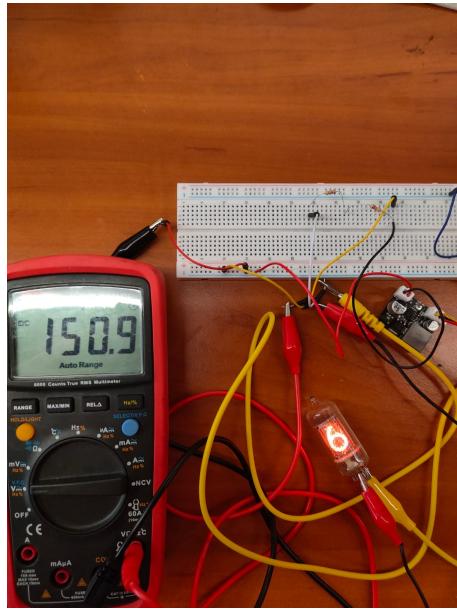
W celu sprawdzenia działania lampy należy najpierw dobrać rezystor ograniczający prąd katodowy. Zakładając, że napięcie zasilania wynosi maksymalnie $U_{\max} = 220$ V, a napięciu pracy lampy $U_{\text{pr}} = 150$ V, przy prądzie katodowym $I_{\text{kat}} = 2$ mA, rezystor ograniczający prąd katodowy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

$$R = \frac{U_{\max} - U_{\text{pr}}}{I_{\text{kat}}} = \frac{220\text{ V} - 150\text{ V}}{2\text{ mA}} = 35\text{ k}\Omega \quad (1)$$

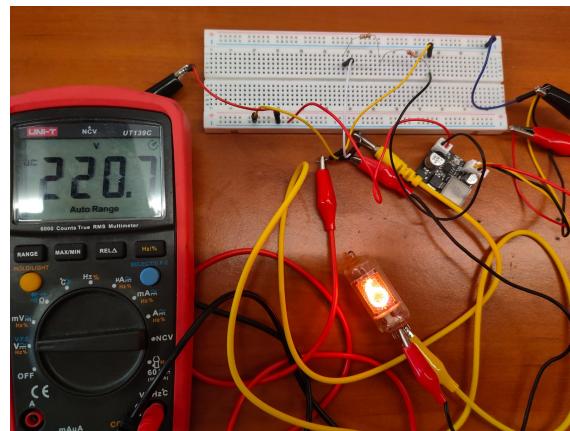
W nocy katalogowej lampy nixie Z570M producent podaje, że zalecany rezystor ograniczający prąd katodowy powinien mieć wartość $33\text{ k}\Omega$ dla napięcia zasilania 200 V, więc wartość $35\text{ k}\Omega$ dla napięcia 220 V wydaje się obliczona prawidłowo, taki też rezystor ma zostać użyty w faktycznym 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, ale ustalono, że lampa nawet przy napięciu 150 V była w stanie zaplonąć 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 zaplonąć i świecić przy napięciu wygaszania 150 V.
- Zakres regulacji napięcia na zasilacz HV 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:

$$P_{\text{LED}} = 18 \text{ W m}^{-1} \cdot 0.3 \text{ m} = 5.4 \text{ W} \quad (2)$$

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

$$I_{\text{LED}} = \frac{P_{\text{LED}}}{U_{\text{LED}}} = \frac{5.4 \text{ W}}{5 \text{ V}} = 1.08 \text{ A} \quad (3)$$

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:

$$P_{\text{ESP32}} = 340 \text{ mA} \cdot 3.3 \text{ V} = 1.122 \text{ W} \quad (4)$$

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

$$I_{\text{Nixie}} = 6 \cdot 2 \text{ mA} = 12 \text{ mA} \quad (5)$$

Następnie obliczono moc potrzebną do zasilania lampy nixie, przy napięciu 220 V oraz prądzie wszystkich lamp 12 mA, zakładając sprawność przetwornicy na poziomie 70 %:

$$P_{\text{Nixie}} = \frac{U_{\text{Nixie}} \cdot I_{\text{Nixie}}}{\text{Sprawność}} = \frac{220 \text{ V} \cdot 12 \text{ mA}}{0.7} = 3.43 \text{ W} \quad (6)$$

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

$$P_{\text{całkowita}} = P_{\text{LED}} + P_{\text{ESP32}} + P_{\text{Nixie}} = 5.4 \text{ W} + 1.122 \text{ W} + 3.43 \text{ W} = 9.952 \text{ W} \quad (7)$$

Szacunkowa moc potrzebna do zasilania całego układu wynosi około 10 W,

7 Realizacja modułów

Po określeniu 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 rejestyry przesuwne HV firmy microchip o numerze HV5530, o następujących parametrach[11]:

- Rejestr 32 bitowy
- Maksymalne napięcie na wyjściu - 315V
- Maksymalna częstotliwość pracy - 8MHz
- napięcie zasilania - od 10.8V do 13.6V
- stan wysoki - Napięcie zasilania - 2V

7.1.2 Sterowanie rejestrów

Sterowanie rejestrem realizowane jest za pomocą następujących pinów:

- CLK - sterowanie zegarem rejestru
- LE - załadowanie danych do rejestru(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 rejestru

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 3.3V, a rejestr operuje na napięciu około 12V.

Zastosowano konwerter poziomów logicznych CD40109B-Q1 firmy Texas Instruments[12]. Konwerter jest 4 kanałowy, co pozwala na podłączenie 4 sygnałów, więc wybrano połączenia CLK, LE, DATA_IN, BL. Konwerter pracuje w zakresie napięć od 3V do 20V, 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 [11]:

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

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.

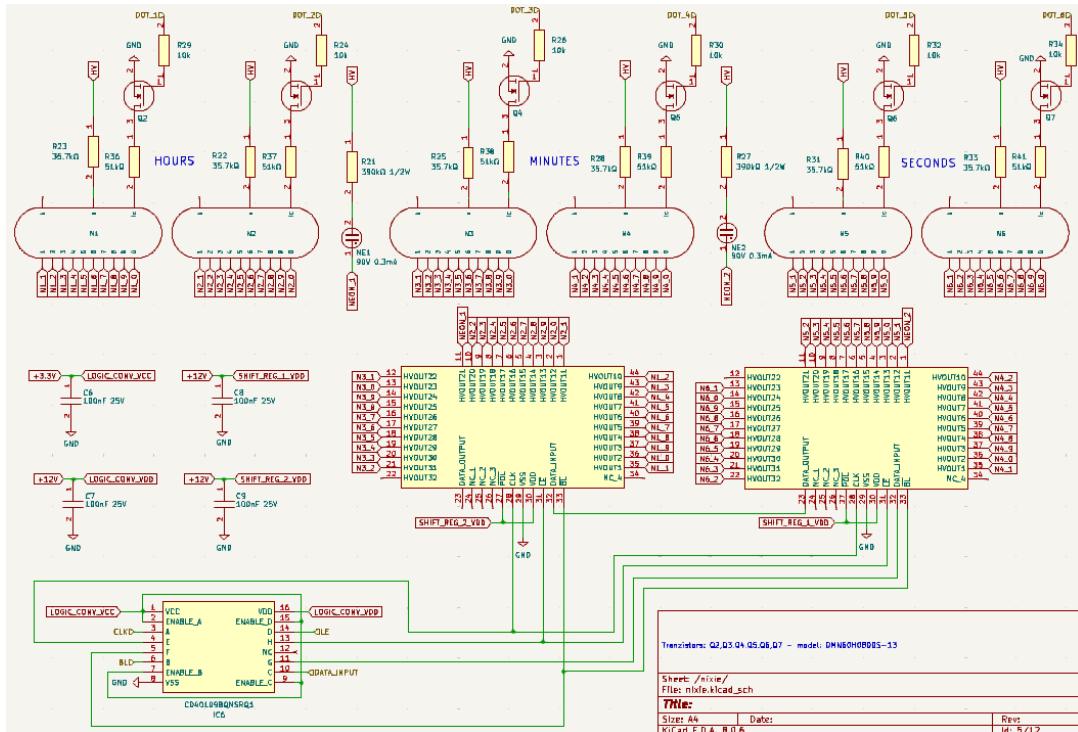
Lampy neonowe mają zdecydowanie mniejszy prąd pracy oraz mniejsze napięcie pracy. W sklepie internetowym sprzedający deklarował następujące parametry:

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

Rezystor potrzebny do zabezpieczenia lampy neonowej przy napięciu zasilania 220V powinien mieć wartość około $433\text{k}\Omega$.

$$R = \frac{U}{I} = \frac{220V - 90V}{0.3mA} \approx 433k\Omega \quad (8)$$

Zdecydowano się na użycie rezystora o wartości $390\text{k}\Omega$, ze względu na dostępność w sklepie internetowym.



Rysunek 7.1: Schemat układu sterowania lampami

7.2 Przetwornica 12V na HV

Przetwornica impulsowa HV jest najtrudniejszym do zaprojektowania elementem układu, do tego 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.

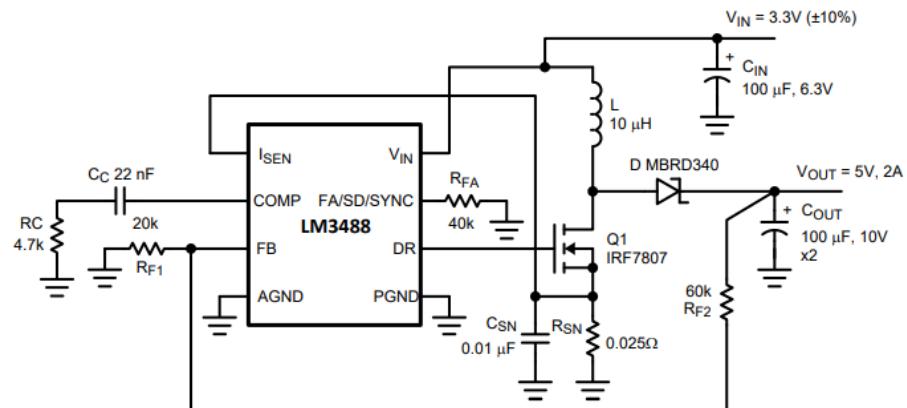
7.2.1 Założenia projektowe

- Napięcie wejściowe: 12V
- Napięcie wyjściowe: 130-220V
- Prąd wyjściowy: 20mA
- 0.1V 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[13].

Jako początkowe założenie przyjęto częstotliwość przełączania 400kHz zgodnie z domyślną wartością w nocy katalogowej układu, ale możliwe jest zwiększenie częstotliwości do 1MHz, 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.



Rysunek 7.2: Schemat układu z karty katalogowej[13]

7.2.3 Dobór cewki

Wartość prądu cewki oszacowano na podstawie założonego prądu wyjściowego:

$$I_l = \frac{V_{out} \cdot I_{out}}{V_{in}} = \frac{220 \cdot 0.02}{12} \approx 0.36A \quad (9)$$

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

$$I_{peak} = (1 + 0.3) \cdot I_l = 1.3 \cdot 0.36 \approx 0.468A \quad (10)$$

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

Obliczono wypełnienie PWM, na podstawie wzoru:

$$D_{220} = \frac{V_{out} - V_{in}}{V_{out}} = \frac{220 - 12}{220} \approx 0.945 \quad (11)$$

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

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

$$L > \frac{D(1 - D)V_{in}}{2f_{sw}I_{out}} \quad (13)$$

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

$$I_{out} = 0.3 \cdot 20\text{mA} = 6\text{mA} \quad (14)$$

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

$$L_{220} > \frac{0.945 \cdot (1 - 0.945) \cdot 12}{2 \cdot 400000 \cdot 0.006} \approx 128.9\mu\text{H} \quad (15)$$

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

$$L_{130} > \frac{0.908 \cdot (1 - 0.908) \cdot 12}{2 \cdot 400000 \cdot 0.006} \approx 209.5\mu\text{H} \quad (16)$$

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 0.9A, co jest wystarczającym zapasem prądowym.

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

$$L_{220} > \frac{0.945 \cdot (1 - 0.945) \cdot 12}{2 \cdot 500000 \cdot 0.006} \approx 103.1\mu\text{H} \quad (17)$$

$$L_{130} > \frac{0.908 \cdot (1 - 0.908) \cdot 12}{2 \cdot 500000 \cdot 0.006} \approx 167.6\mu\text{H} \quad (18)$$

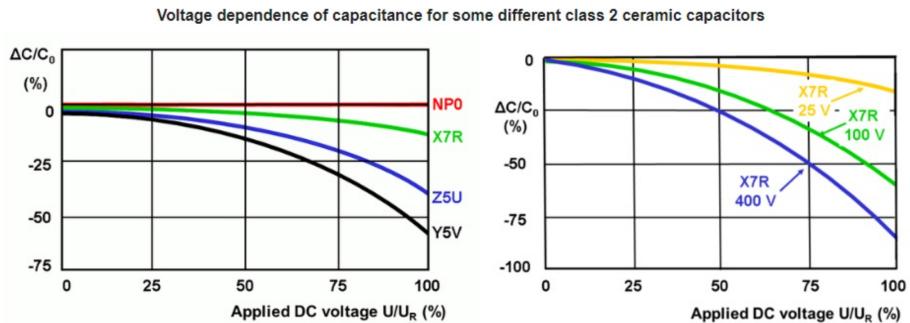
Z obliczeń wynika, że zwiększenie częstotliwości pozwala na zmniejszenie wartości cewki, co pozwala na zastosowanie cewki o wartości 180 μH .

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 μF i napięciu pracy do 250V.

W obliczeniach należy uwzględnić spadek pojemności kondensatora wraz ze wzrostem napięcia. Wartości spadku pojemności dla kondensatora ceramicznego odczytano z następującego wykresu:



Rysunek 7.3: Spadek pojemności kondensatora ceramicznego wraz ze wzrostem napięcia

Wybrany kondensator jest klasy 2, wykonany z materiału X7R, co oznacza, że jego pojemność spadnie w przybliżeniu o 30% dla napięcia 220V. 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:

$$C_{220} = 2 \cdot (1 - 0.3) \cdot 2.2\mu\text{F} = 3.08\mu\text{F} \quad (19)$$

Obliczono tętnienia dla dobranych wartości kondensatorów:

$$\Delta V_{220} = \frac{V_{out}}{2 \cdot \frac{V_{out}}{I_{out}} \cdot C} \cdot \frac{D}{f_{sw}} = \frac{220}{2 \cdot \frac{220}{0.02} \cdot 3.08\mu\text{F}} \cdot \frac{0.945}{500000} \approx 6.1\text{mV} \quad (20)$$

$$\Delta V_{130} = \frac{V_{out}}{2 \cdot \frac{V_{out}}{I_{out}} \cdot C} \cdot \frac{D}{f_{sw}} = \frac{130}{2 \cdot \frac{130}{0.02} \cdot 3.08\mu\text{F}} \cdot \frac{0.908}{500000} \approx 5.89\text{mV} \quad (21)$$

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

7.2.5 Dobór diody

Oszacowano prąd diody na podstawie wzoru z karty katalogowej:

$$I_d = \frac{I_{out}}{1 - D} + \Delta I_{out} = \frac{0.02}{1 - 0.945} + 0.006 \approx 0.37\text{A} \quad (22)$$

Dioda powinna mieć prąd przewodzenia większy niż 0.4A oraz być szybką diodą shottky'ego, by zminimalizować straty w układzie.

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

7.2.6 Dobór tranzystora

Można założyć, że prąd tranzystora to prąd cewki, czyli 0.468A. Zgodnie z kartą katalogową tranzystor powinien mieć następujące parametry:

- Napięcie minimalnie drain-source: 250V
- Jak najmniejszy $R_{DS(on)}$
- Prąd przewodzenia większy niż 0.5A

- Niskie napięcie progowe V_{TH}
- Jak najmniejszy ładunek bramki
- Wymaganego prąd bramki mniejszego niż 1A

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

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

Prąd potrzebny na załączenie tranzystora można obliczyć na podstawie wzoru:

$$I_g = Q_g \cdot f_{sw} = 22\text{nC} \cdot 500000 = 11\text{mA} \quad (23)$$

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:

$$P_{mos} = I_l^2 \cdot R_{DS(on)} = 0.468^2 \cdot 0.044 \approx 0.01\text{W} \quad (24)$$

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 10kΩ, co pozwala na uzyskanie odpowiedniego zakresu ustawień napięcia wyjściowego.

Zgodnie z kartą katalogową napięcie na pinie FB powinno wynosić 1.26V, 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 10kΩ:

$$V_{out} = 1.26 \cdot \left(1 + \frac{R_{fb1}}{R_{fb2} + R_{pot}}\right) = 1.26 \cdot \left(1 + \frac{2.49M\Omega}{14.39k\Omega + 10k\Omega}\right) \approx 130.4V \quad (25)$$

Obliczone napięcie na wyjściu dla potencjometru z nastawą 0Ω:

$$V_{out} = 1.26 \cdot \left(1 + \frac{R_{fb1}}{R_{fb2} + R_{pot}}\right) = 1.26 \cdot \left(1 + \frac{2.49M\Omega}{14.39k\Omega}\right) \approx 220.6V \quad (26)$$

Uzyskano zakres napięcia wyjściowego od 130.4V do 220.6V, 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:

$$ISW_{limit} = \left(\frac{I_{out}}{1 - D} + \frac{D \cdot V_{in}}{2 \cdot f_{sw} \cdot L} \right) = \left(\frac{0.02}{1 - 0.945} + \frac{0.945 \cdot 12}{2 \cdot 500000 \cdot 180\mu H} \right) \approx 0.426A \quad (27)$$

Następnie obliczono wartość rezystora ograniczającego prąd zgodnie z kartą katalogową:

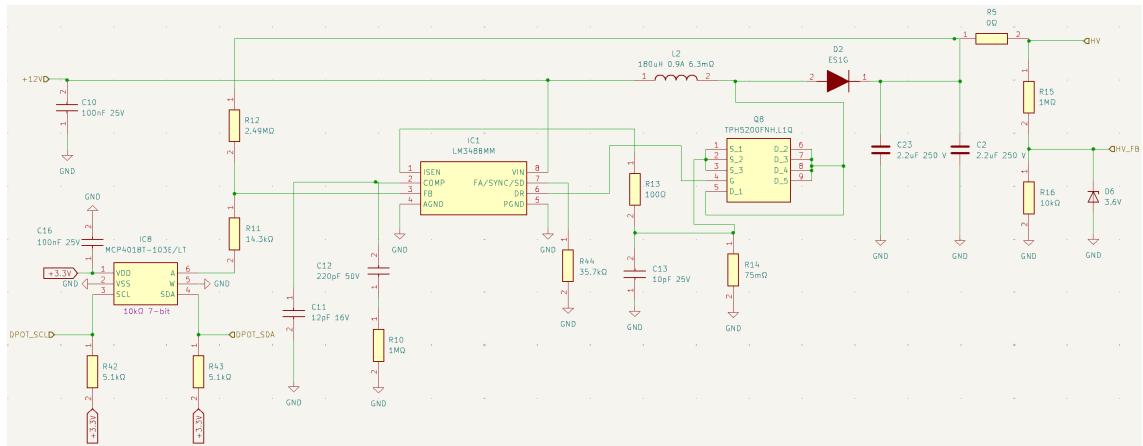
$$R_{sense} = \frac{V_{SENSE} - (D \cdot V_{SENSE} \cdot V_{SL-ratio})}{ISW_{limit}} = \frac{156mV - (0.945 \cdot 156mV \cdot 0.49)}{0.426} \approx 74m\Omega \quad (28)$$

Następnie sprawdzono warunek na maksymalną wartość rezystora ograniczającego prąd:

$$R_{sense} < \frac{2 \cdot V_{SL} \cdot f_{sw} \cdot L}{V_{out} - (2 \cdot V_{IN})} = \frac{2 \cdot 92mV \cdot 500000 \cdot 180\mu H}{220 - (2 \cdot 12)} \approx 84m\Omega \quad (29)$$

Zdecydowano się na zastosowanie rezystora o wartości 75mΩ, który spełnia wszystkie założenia. Został również dodany kondensator o wartości 10pF w celu zminimalizowania tętnień napięcia na rezystorze, oraz rezystor kompensujący 100Ω.

7.2.9 Schemat



Rysunek 7.4: Schemat przetwornicy 12V na HV

7.3 Przetwornica 12V na 5V

7.3.1 Założenia projektowe

- Napięcie wejściowe: 12V
- Napięcie wyjściowe: 5V
- Prąd wyjściowy: 2A
- 50mV 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 3A przy napięciu wyjściowym do 7V. 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].

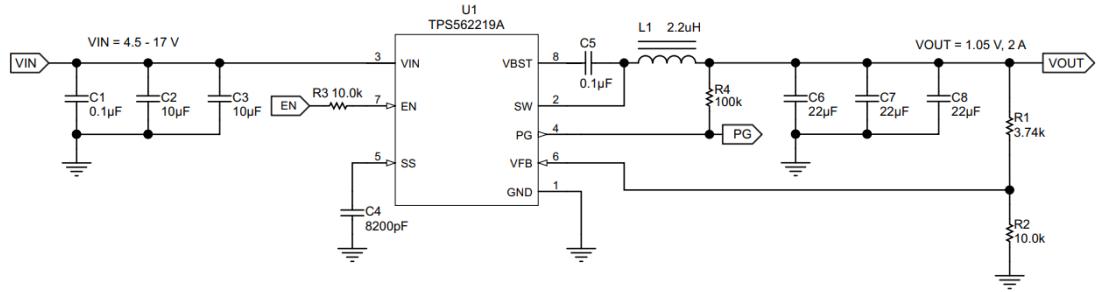
Table 4. TPS563219A Recommended Component Values

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]

Na podstawie tabeli dobrano następujące wartości komponentów:

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



Rysunek 7.6: Typowe połaczenie układu TPS563219ADDFR[14]

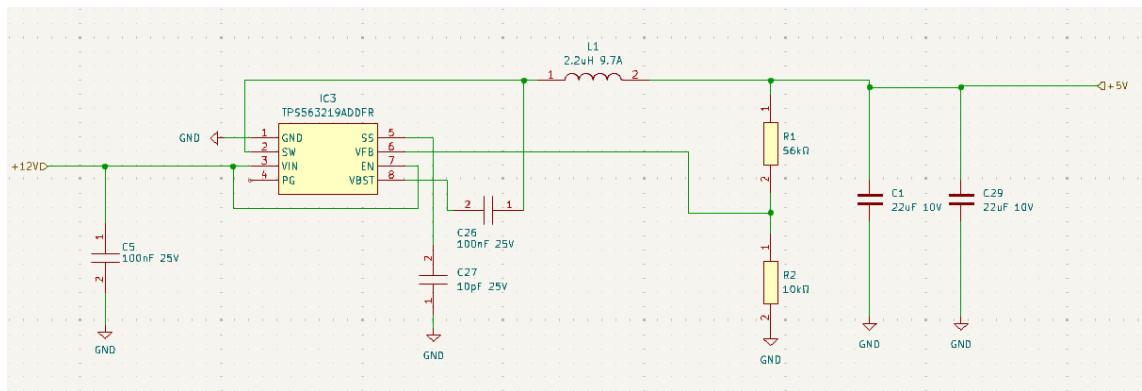
Kondensator dołączony do wyprowadzenia SS(soft start) oraz kondensator dołączony do pinu VBST, został skopiowany z układu z noty katalogowej, gdyż nie jest to krytyczny element i nie ma potrzeby doboru wartości pod kątem zastosowania w zegarze.

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, więc ESR kondensatora ma większe znaczenie, 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 dla znaczenia.

Cewka dobrano biorąc pod uwagę jest stosunek oporu do ceny, zdecydowano się na cewkę o oporze 14.5mΩ, gdyż jest to najniższa wartość jako udało się znaleźć w sklepach elektronicznych w sensownej cenie i dość małej obudowie.

Dodano również kondensator filtrujący 100nF na pinie VCC, by zredukować szumy z linii zasilania.

7.3.4 Schemat



Rysunek 7.7: Schemat złącza DC-Plug

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 12V daje prąd 0.83A. 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 0.65x2.35mm, 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 100uF w celu zminimalizowania zakłóceń niskich częstotliwości, oraz 1 kondensator o pojemności 100nF w celu zminimalizowania zakłóceń wysokich częstotliwości.

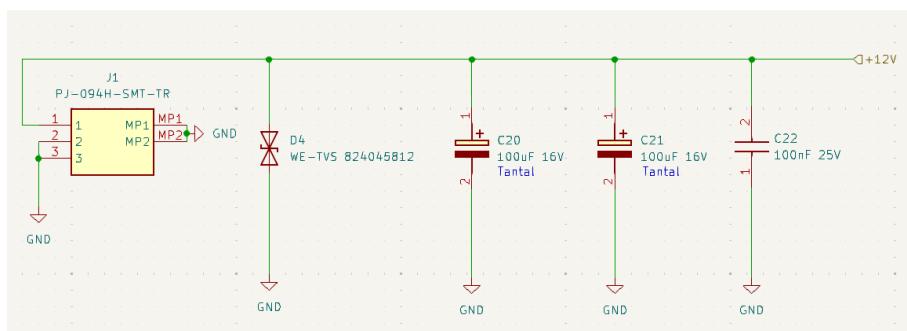
Kondensatory 100uF są kondensatorami tantalowymi, a kondensator 100nF jest kondensatorem ceramicznym. Wykorzystano te rodzaje kondensatorów, ponieważ są to kondensatory o długiej żywotności, a także mają one małe rozmiar w przeciwieństwie do kondensatorów elektrolitycznych.

7.4.3 Zabezpieczenia ESD

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

Dioda ta została wybrana gdyż nie udało się znaleźć diody TVS o napięciu stabilizacji 13V. 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.

7.4.4 Schemat



Rysunek 7.8: Schemat złącza DC-Plug

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 3.3V, 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 diody TVS PUSB3AB4Z firmy Nexperia. Diody te mają wystarczająco duże opakowanie by dało się je przylutować ręcznie. Napięciem roboczym jest 3.3V, a napięcie stabilizacji wynosi 5V.

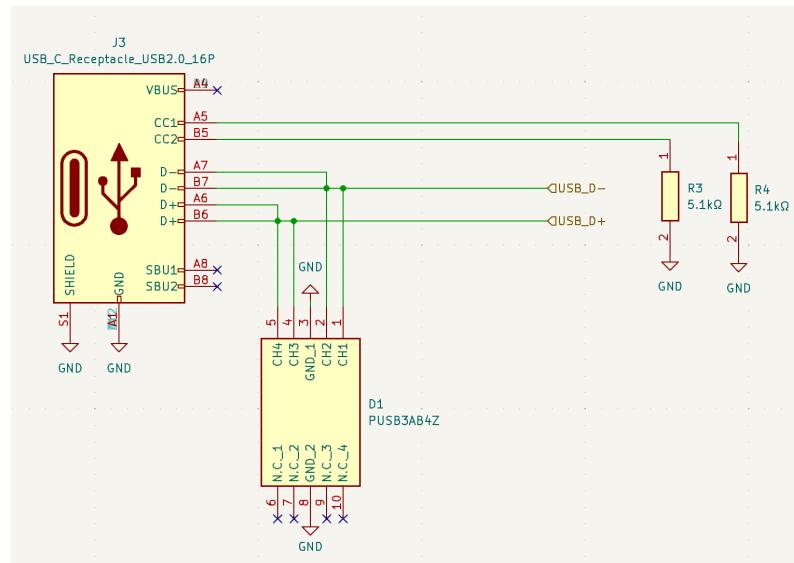
Mimo że jest to napięcie wyższe niż napięcie zasilania ESP32-S3, to nie powinno to stanowić problemu, ponieważ napięcie to pojawi się na krótki czas, a sam esp32-s3 ma również wbudowane zabezpieczenia przed przepięciami.

Wewnętrzne zabezpieczenia według katalogowej ESP32-S3:

- Test Standard JS-001; HBM (Human Body Mode) $\pm 2000 \text{ V}$
- Test Standard JS-002; CDM (Charged Device Model) $\pm 1000 \text{ V}$

Wynika z tego, że złącze USB-C w dość dobry sposób jest zabezpieczone przed przepięciami.

7.5.4 Schemat



Rysunek 7.9: Schemat złącza USB-C do programowania

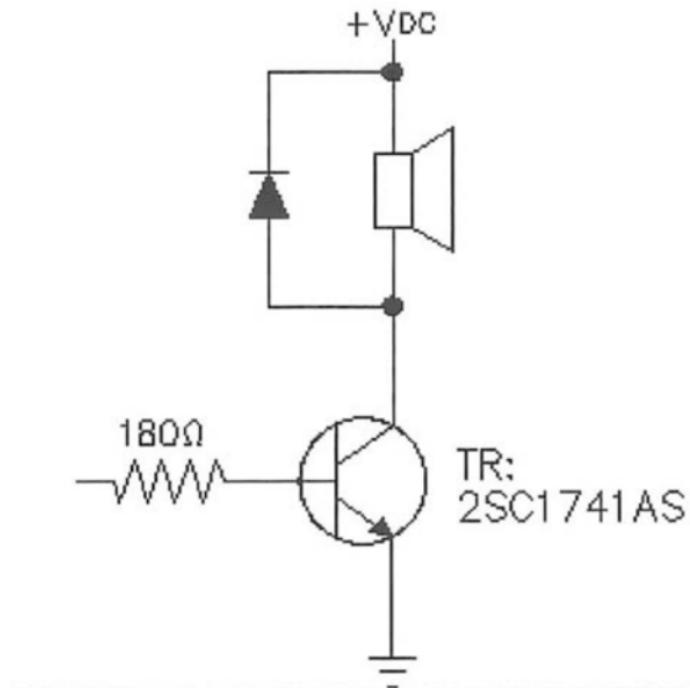
7.6 Buzzer

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

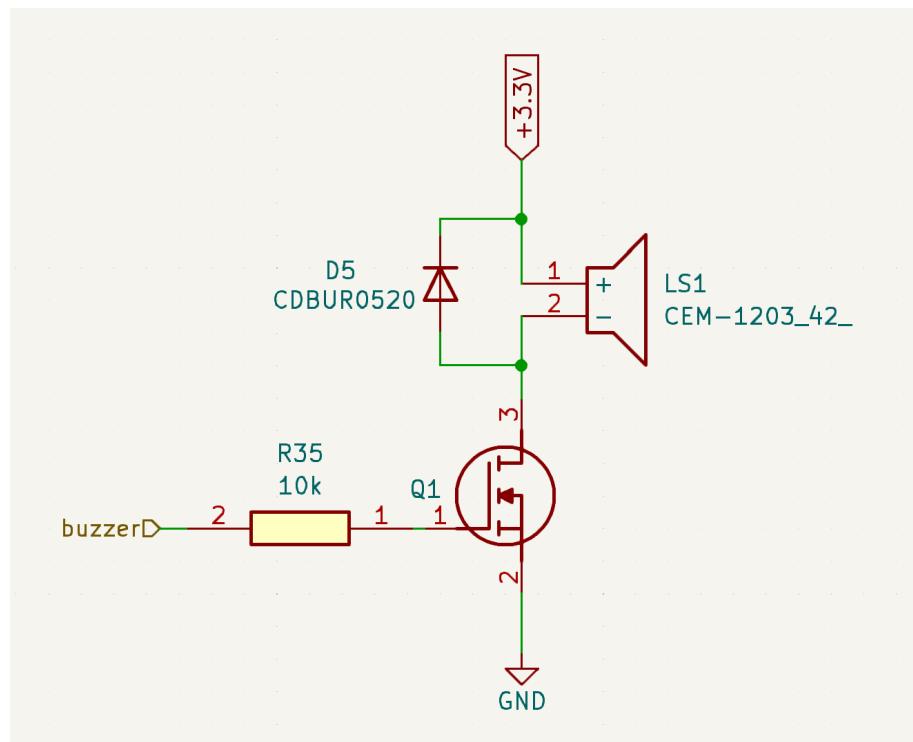
- Napięcie zasilania: 3-5V
- Max prąd zasilania: 35mA
- Częstotliwość: 2,048kHz
- Głośność: 85dB - 95dB
- Średnica: 12mm, wysokość: 8mm
- Rezystancja: 42Ω
- Typ montażu: THT

Buzzer został podłączony według schematu zalecanego przez producenta. 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 80mA, co jest wystarczające do sterowania buzzerem.

Buzzer jest dostatecznie mały, jedyną wadą jest montaż THT, który ogranicza miniaturyzację płytki drukowanej.



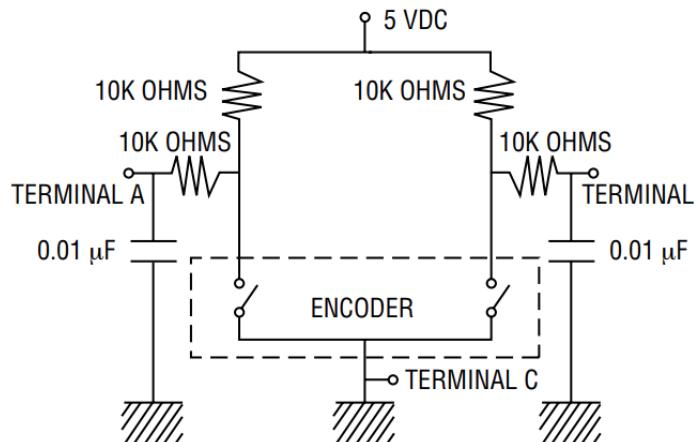
Rysunek 7.10: Schemat zalecany przez producenta[15]



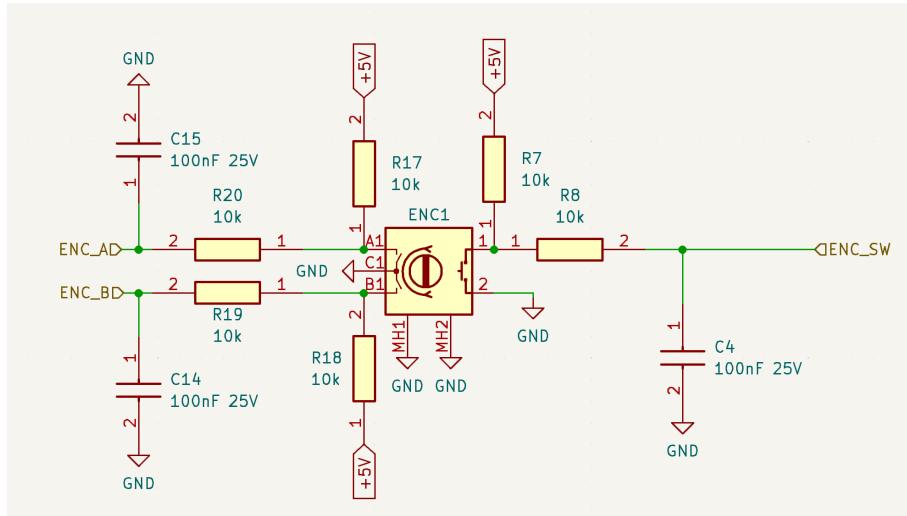
Rysunek 7.11: Gotowy układ z buzzerem

7.7 Encoder

Jednymi kryteriami wyboru encodera były napięcie zasilania oraz by posiadał on przycisk. Wybrano encoder PEC12R41BBFS0012 firmy Bourns[16], jest to 2 kanałowy encoder o rozdzielcości 12 impulsów na obrót. Jego napięcie robocze wynosi 5V. Producent w karcie katalogowej informuje również, jakie filtry zastosować, by zapobiec zakłóceniom. Dodatkowo dodano filtr dolno-przepustowy, przy przycisku encodera, by zapobiec drganiom styków i nie musieć implementować debouncingu w oprogramowaniu.



Rysunek 7.12: Schemat filtrów zalecany przez producenta[16]

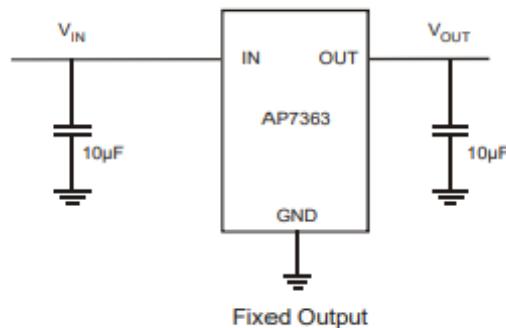


Rysunek 7.13: Gotowy układ encodera

7.8 LDO 5V na 3.3V

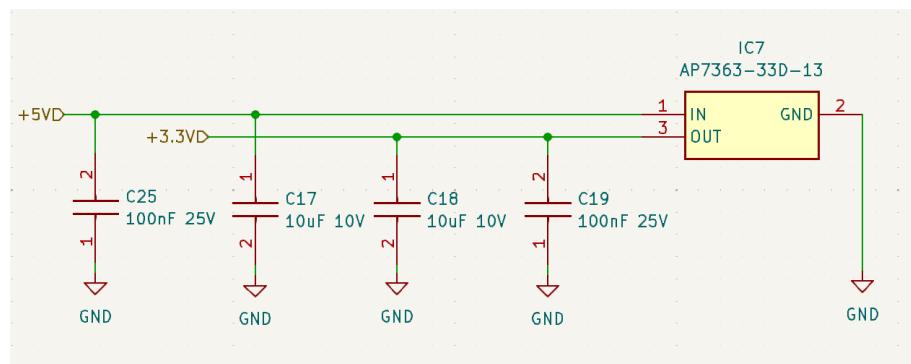
Wymagania jakie musi spełnić LDO to zapewne zasilania mikrokontrolera ESP32 oraz potencjometru cyfrowego. Zgodnie z dokumentacją ESP32[5] pobiera on maksymalnie 340mA prądu, a potencjometr cyfrowy[17] 100mA, co daje łącznie 440mA. Układ LDO ma być zasilany 5V, a na wyjściu ma być 3.3V.

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



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

Względem schematu zaproponowanego przez producenta, zdecydowano się na zastosowanie dodatkowo kondensator ceramiczny o pojemności 100nF, by zminimalizować zakłócenia. Dodano po kondensatorze na wejście i wyjście układu równolegle do kondensatorów 10uF zaproponowanych przez producenta.



Rysunek 7.15: Gotowy układ LDO

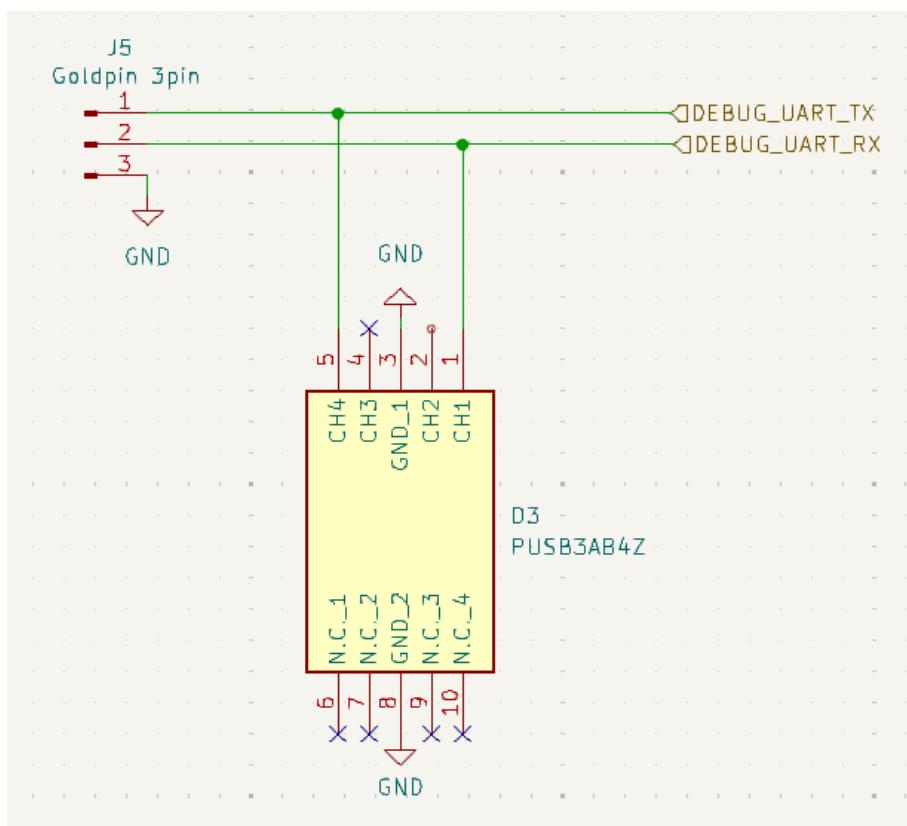
7.9 Złącze do debugowania

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 wystawia następujące sygnały:

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

W celu zabezpieczenia złącza przed ESD, zastosowano diodę TVS, analogiczną co w podrozdziale 7.5.



Rysunek 7.16: Gotowy układ do debugowania przez UART

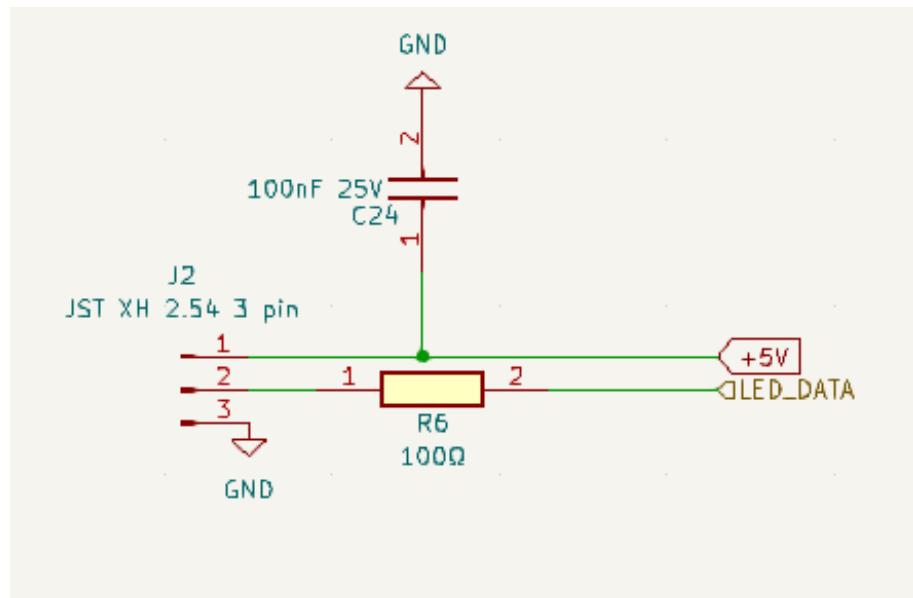
7.10 Złącze do paska LED

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

Złącze będzie wystawia następujące sygnały:

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

Wybrano złącze typu JST w orientacji horyzontalnej w cel zaoszczędzenie miejsca. Dodatkowo na linie danych dodano zabezpieczający rezystor 100Ω w celu. Zastosowano również kondensator odsprzęgający $100nF$.

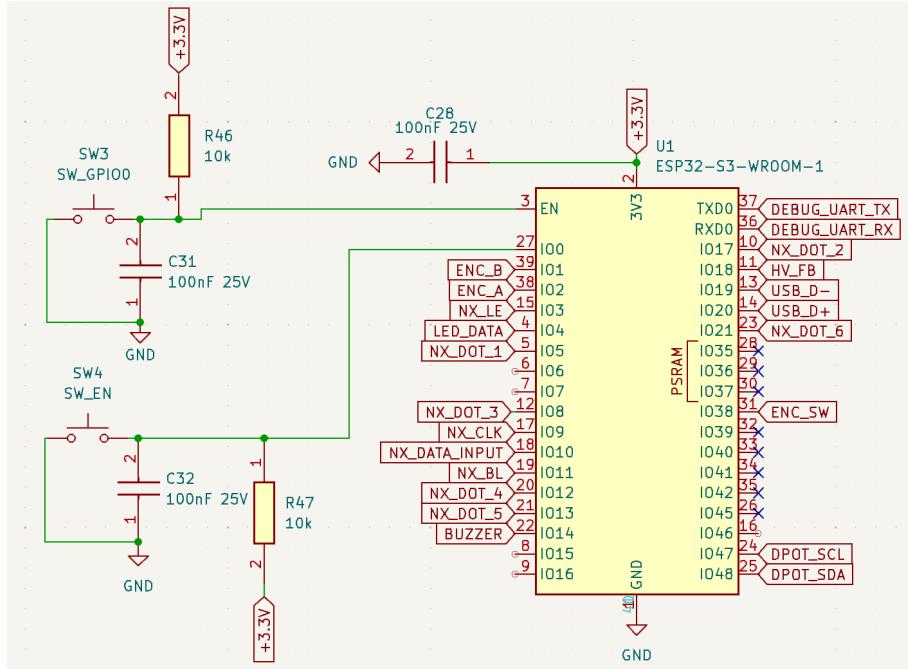


Rysunek 7.17: Gotowy układ złącza

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 100nF, by zminimalizować zakłócenia oraz rezystor podciągający o wartości 10kΩ.

Dodano, również kondensator odsprzęgający 100nF 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.



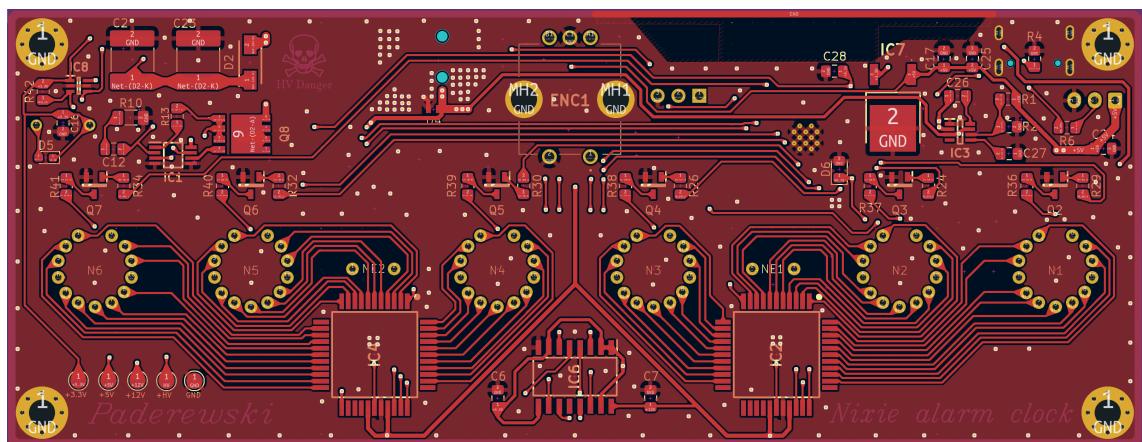
Rysunek 7.18: Schemat podłączenia ESP32-S3

8 Projekt i montaż płytki drukowanej

8.1 Projekt płytki drukowanej

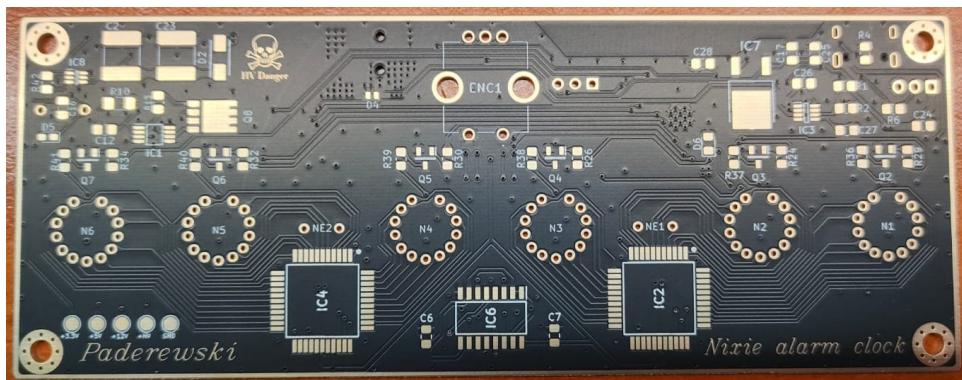
Do zaprojektowania płytki drukowanej wykorzystano open-source'owy program KiCad. Zdecydowano się na zastosowanie dwustronnej płytki drukowanej. Ostatecznie udało się uzyskać płytę o wymiarach 138x53mm oraz szacowana wysokość wraz z komponentami na poziomie 1cm, co oznacza, że udało się osiągnąć cel wykonania jak najmniejszej płytki drukowanej.

W celu poprawy aspektów wizualnych płytki, zamówiona została pozłacana płyta drukowana. Uzyskano też pozłacaną ramkę, która powstała przez nie nakładanie soldermaski na krawędzie płytki, w taki sam sposób uzyskano pozłacane napisy. Płytki posiada 4 otwory montażowe ma śruby M3. Otwory montażowe oraz otwory montażowe encodera są podłączone do masy, co pozwala na lepsze prowadzenie masy między warstwami płytki. W celu poprawy 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.

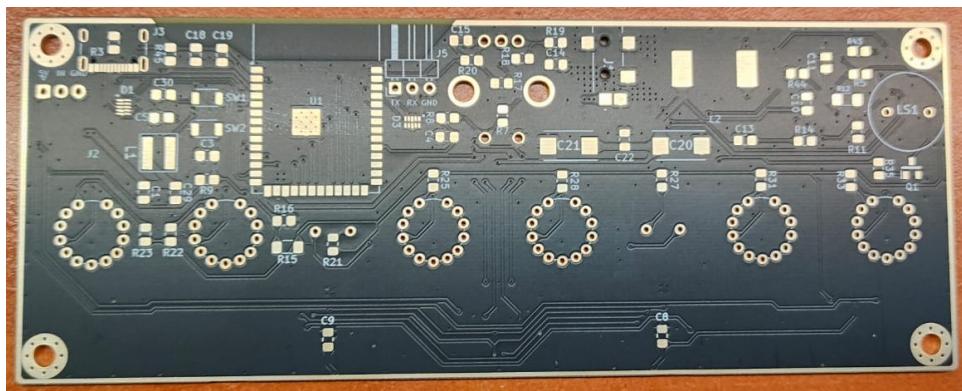


8.2 Montaż i uruchomienie układu

Kolejnym etapem realizacji projektu był montaż elementów na płytce drukowanej. Zamówione płytki widzimy na rysunkach 8.3 oraz 8.4. Przed rozpoczęciem lutowania wszystkie pozłacane elementy które ostały dodane jak elemnet estetyczny, 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: Płytki drukowane - widok od góry

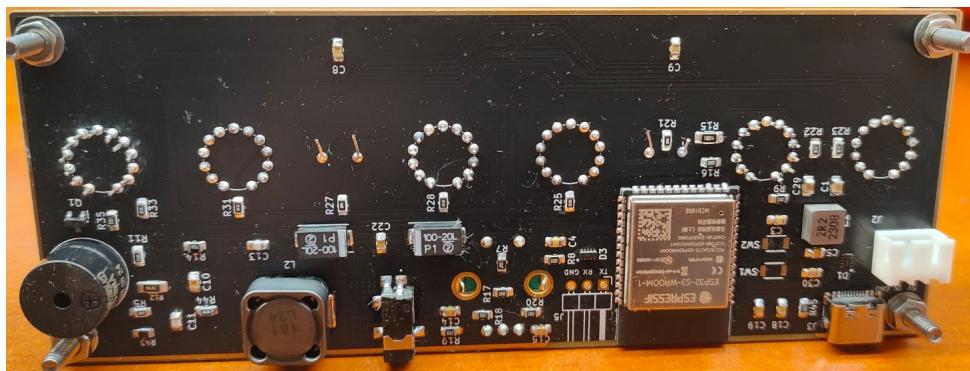


Rysunek 8.4: Płytki drukowane - widok od spodu

Lutowanie rozpoczęto od elementów najmniejszych SMD(montaż powierzchniowy). 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 linie. Lampy mają duże tolerancje produkcyjne, co powoduje że niektóre z lamp mogą być wyżej lub niżej niż pozostałe. W celu wypoziomowania lamp, wykorzystano katownik. Ostatnim etapem montażu było końcowe czyszczenie płytki z nadmiaru topnika.



Rysunek 8.5: Zmontowana płytka drukowana - widok od góry



Rysunek 8.6: Zmontowana płytka drukowana - widok od spodu

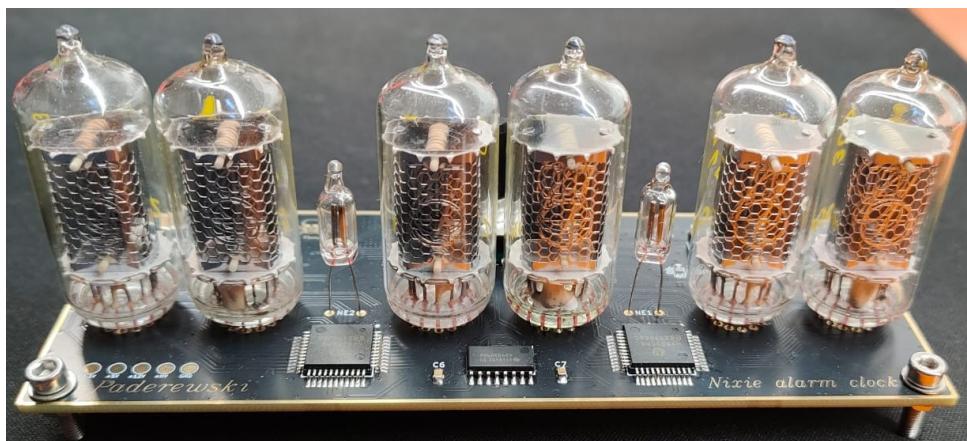
8.3 Uruchomienie układu

Przetwornica HV jako najtrudniejszy moduł układu, została odizolowana na etap pierwszego uruchomiania, w celu zminimalizowania ryzyka uszkodzenia lamp Nixie. Zrealizowano izolacje poprzez nie przylutowanie rezystora 0Ω 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 5V - 5.081V
- linia 3.3V - 3.306V
- linia 12V - 12.01V
- linia Wysokie napięcie - 162.1V

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



Rysunek 8.7: Zmontowany układ

9 Oprogramowanie

Do napisanie oprogramowania budzika wykorzystano język wykorzystano środowisko PlatformIO, używając języka C++. Program musi spełniać następujące wymagania:

- Obsługa rejestrów przesuwnych w celu wyświetlenia cyfr na lampach nixie.
- Połączenie z siecią Wi-Fi.
- Pobranie czasu z serwera czasu.
- Obsługa enkodera rotacyjnego.
- 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 następnym etapie projektowania oprogramowania.

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 łączącym cyfrę z konkretnym bitem w rejestrze przesuwnym.
- Klasę do animacji wyświetlacza.

Wysyłanie danych do rejestru 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 wyświetlaną cyfrą, w momencie zmiany gdy dana cyfra ma wyświetlić 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 funkcje do przeliczania zadanej 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 dzieliku 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 melodye są zdefiniowane w pliku `melody.hpp`, a poszczególne tony w pliku `tones.hpp`. Same melodye 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, 400, 400, 400, 200, 400, 400, 400},
```

Samo odtwarzanie dzwięku 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 wywoływana 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`.

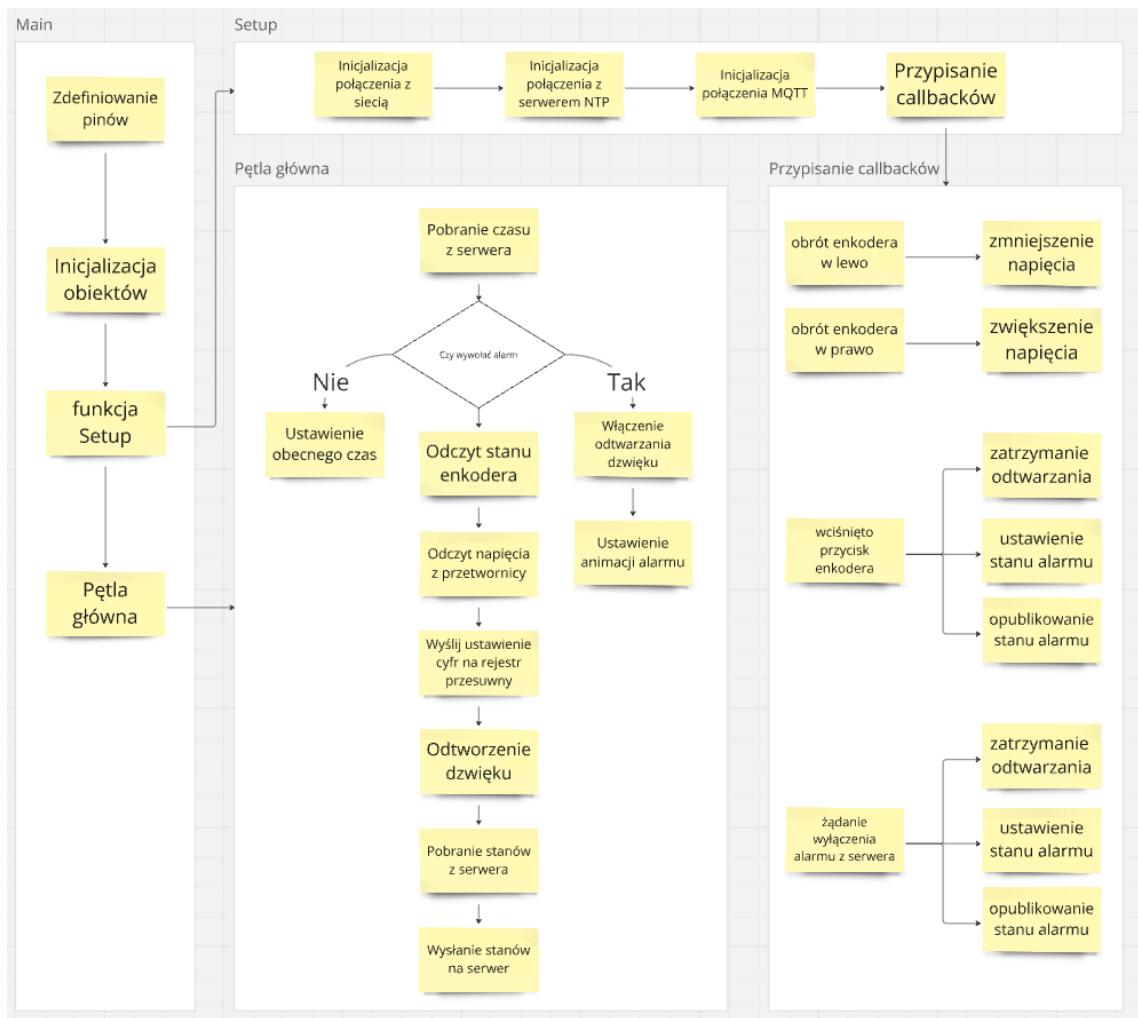
W tej klasie również przechowywane są dane żądanego stanu przez serwer, który jest urządzeniem nadziedzonym. Są to następujące dane:

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

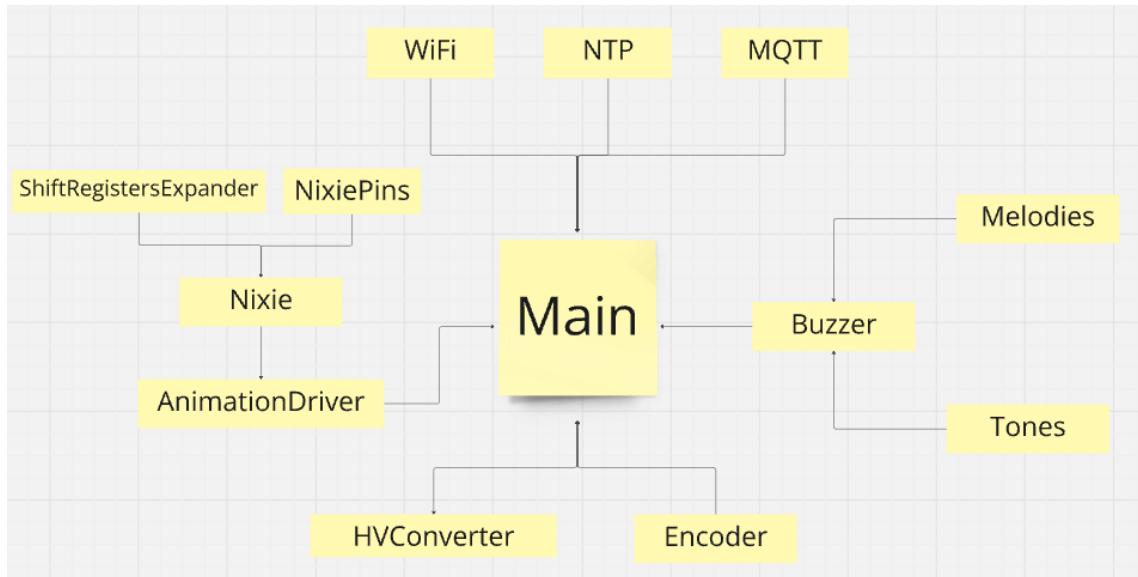
Dodawano również oddzielny topic który informuje zegarek że alarm został włączony, tak samo zegarek publikuje informacje do serwera o stanie alarmu. Na podstawie tych zmiennych będą wykonywane działania w pętli głównej programu.

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

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 HV

Przetwornica HV jako najtrudniejszy moduł układu, została odizolowana na etap pierwszego uruchomiania rezystorem 0Ω w celu zminimalizowania ryzyka uszkodzenia lamp Nixie. Po podłączeniu zasilania przetwornica działa z napięciem wyjściowym ok 160V, co jest wartością oczekiwana. Napięcie to jest stabilne, nie obserwuje się żadnych skoków napięcia.

Po napisaniu testowego programu do komunikacji z potencjometrem, przetestowano działanie regulacji. Regulacja działa poprawnie, faktyczny zakres regulacji wynosi ??? i ???;

10.2 Testy układu sterowania

Po pełnym złożeniu układu napisano prosty program który zapala i gasi wszystkie cyfry po kolei by przetestować działanie sterowanie lampami nixie.

10.3 Testy enkodera i buzzera

Najpierw napisano prosty program do obsługi enkodera i przetestowano jego działanie. Działa wykrywanie obrotu oraz przycisk. Analogicznie wgrano przykładowy program do odtwarzania dźwięków i przetestowano działania buzzera. Udało się zagrać melodie, ale buzzer okazał się cichszy niż zakładano, ale nie powinno to stanowić problemu.

11 Podsumowanie

Bibliografia

- [1] *Zegar Nixie - nostalgiczna elegancja.* 4/2024 Kwiecień (16). Zrozumieć Elektronikę z Piotrem Góreckim. Kw. 2024. URL: <https://piotr-gorecki.pl/wp-content/uploads/2024/03/ZE2404.pdf>.
- [2] *Lampa cyfrowa. Schemat elektryczny lampy NIXIE.* Wikipedia. 2023. URL: https://pl.wikipedia.org/wiki/Lampa_cyfrowa.
- [3] *Kompendium wiedzy o lampach Nixie.* Rev. 2. rafalbartoszak. Kw. 2019. URL: <https://rafalbartoszak.pl/kompendium-wiedzy-o-lampach-nixie/>.
- [4] *Internet of Things Projects with ESP32. Build exciting and powerful IoT projects using the all-new Espressif ESP32.* Agus Kurniawan. Mar. 2019.
- [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] *Expert Network Time Protocol: An Experience in Time with NTP (Expert's Voice).* 9781590594841. Peter Rybaczyk. 2005.
- [8] *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>.
- [9] *Network Time Protocol. Struktura warstw STRATUM 0-15.* Wikipedia. 2024. URL: https://pl.wikipedia.org/wiki/Network_Time_Protocol.
- [10] *RFT tube data book and translation.* Z570M/Z5700M. RFT electronic. URL: <https://www.tube-tester.com/sites/nixie/data/z5730m/z5730m.html>.
- [11] *32-Channel Serial-to-Parallel Converter With Open Drain Outputs.* DS20005851A. Microchip. Paź. 2017. URL: <https://ww1.microchip.com/downloads/en/DeviceDoc/20005851A.pdf>.
- [12] *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>.
- [13] *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.
- [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] *Part No: CEM-1203(42) Description: 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] *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] *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[2]	10
3.1	Rozkład pinów mikrokontrolera ESP32-S3[6]	14
4.1	Struktura serwerów czasu w protokole NTP[9]	17
5.1	Ogólna koncepcja układu	19
5.2	Ogólny schemat komunikacji z serwerem	20
6.1	Schemat blokowy projektu	23
6.2	Prototyp układu z lampą nixie przy napięciu zasilania 150 V	25
6.3	Prototyp układu z lampą nixie przy napięciu zasilania 220 V	25
7.1	Schemat układu sterowania lampami	28
7.2	Schemat układu z karty katalogowej[13]	29
7.3	Spadek pojemności kondensatora ceramicznego wraz ze wzrostem napięcia	31
7.4	Schemat przetwornicy 12V na HV	33
7.5	Tabela doboru komponentów z noty katalogowej[14]	34
7.6	Typowe połączenie układu TPS563219ADDFR[14]	35
7.7	Schemat złącza DC-Plug	35
7.8	Schemat złącza DC-Plug	36
7.9	Schemat złącza USB-C do programowania	37
7.10	Schemat zalecany przez producenta[15]	38
7.11	Gotowy układ z buzzerem	39
7.12	Schemat filtrów zalecany przez producenta[16]	40
7.13	Gotowy układ encodera	40
7.14	Schemat podłączenia LDO zalecany przez producenta[18]	41
7.15	Gotowy układ LDO	41
7.16	Gotowy układ do debugowania przez UART	42
7.17	Gotowy układ złącza	43
7.18	Schemat podłączenia ESP32-S3	44
8.1	Górna warstwa płytki drukowanej	45
8.2	Dolna warstwa płytki drukowanej	45
8.3	Płytnica drukowana - widok od góry	46
8.4	Płytnica drukowana - widok od spodu	46
8.5	Zmontowana płytka drukowana - widok od góry	47
8.6	Zmontowana płytka drukowana - widok od spodu	47
8.7	Zmontowany układ	48
9.1	Schemat działania programu	55
9.2	Struktura programu	55

Spis tablic

1	Tabela opłacalności sposobów sterowania lampami nixie	12
2	NTP – format komunikatu	18