

Student's name and surname: Paweł Bałbatun

ID: 193652

Cycle of studies: bachelor's degree studies

Mode of study: full-time studies

Field of study: Automatic Control, Cybernetics and Robotics

Profile: Automation Systems

ENGINEERING DIPLOMA THESIS

Title of the thesis: Design of a measurement card for a laboratory stand for measuring physical quantities in HVAC systems, based on a selected design kit with a TFT graphic display.

Title of the thesis (in Polish): Projekt stanowiska pomiarowego podstawowych wielkości fizycznych wykorzystywanych w systemach HVAC w oparciu o wybrany zestaw uruchomieniowy wyposażony w wyświetlacz graficzny TFT

Supervisor: mgr inż. Piotr Darski

SPIS TREŚCI

1 WSTĘP	2
2 STAN WIEDZY I ZAŁOŻENIA PROJEKTOWE	3
2.1 Sygnał 0–10 V i sterowanie HVAC	3
2.2 Regulacja PI/PID i sekwencje pracy central HVAC	3
2.3 Sterowniki HVAC na mikrokontrolerach i RTOS	3
2.4 Ramy projektowania PCB dla układów mieszanych	4
3 WYMAGANIA PROJEKTU	5
3.0.1 Wymagania funkcjonalne	5
3.0.2 Wymagania niefunkcjonalne	6
4 PROJEKT CZĘŚCI SPRZĘTOWEJ	7
4.1 Wymagania sprzętowe — przegląd	7
4.2 Moduł zasilania	8
4.3 Tor wyjściowy 0–10 V: przetwornik DAC + wzmacniacze operacyjne	9
4.4 Tor wejściowy 0–10 V: interfejs pomiarowy	11
4.5 Projekt PCB i separacja stref	13
5 FIRMWARE	15
5.1 Wybór środowiska uruchomieniowego	15
5.1.1 Dlaczego Zephyr + LVGL?	15
5.2 Struktura projektu firmware	16
5.3 Model konfiguracji HVAC i format JSON	17
5.4 Próby wczytywania konfiguracji z karty SD	20
5.5 Algorytm regulacji PI i sekwencja pracy układu	21
5.6 Interfejs użytkownika (HMI)	22
5.7 Konfiguracja systemu Zephyr	23
5.8 Wątki systemu czasu rzeczywistego i harmonogram zadań	24
5.9 Abstrakcja wejść/wyjść analogowych i integracja z przetwornikami	25
5.10 Diagnostyka i logowanie	25
5.11 Testy i walidacja działania firmware'u	26
5.12 Ograniczenia obecnej implementacji i kierunki rozwoju	26
5.13 Walidacja	27
5.13.1 Metodyka	27
5.13.2 Wyniki	27
5.13.3 Dyskusja	28

Streszczenie

Praca przedstawia projekt oraz weryfikację stanowiska pomiarowego dla wybranych wielkości fizycznych typowych w systemach HVAC (temperatura, wilgotność, ciśnienie, przepływ oraz sygnały sterujące 0 V–10 V). Rozwiązanie bazuje na zestawie uruchomieniowym STM32F746G-Discovery z wyświetlaczem TFT oraz na dedykowanej karcie pomiarowo-wyjściowej 0 V–10 V zaprojektowanej w KiCad. Oprogramowanie wykonano w środowisku Zephyr RTOS z wykorzystaniem sterowników peryferiów i biblioteki LVGL do obsługi interfejsu graficznego. Przedstawiono wymagania funkcjonalne, projekt części analogowej (tory wejściowe z ochroną i skalowaniem, tor wyjściowy 0 V–10 V), architekturę oprogramowania (wątki RTOS, kolejki, sterowniki), a także wyniki walidacji dokładności i powtarzalności pomiarów na podstawie wzorców i porównania z przyrządami referencyjnymi.

1. WSTĘP

Celem pracy jest zaprojektowanie i weryfikacja stanowiska pomiarowego dla podstawowych wielkości fizycznych spotykanych w HVAC oraz interfejsu sterującego 0 V–10 V, z wykorzystaniem zestawu STM32F746G-Discovery (STM32F746NG, Cortex-M7) i dedykowanego modułu pomiarowego PCB. Motywacją jest potrzeba ekonomicznego, dydaktycznego stanowiska do testów i demonstracji algorytmów sterowania.

Omówiono kontekst przemysłowy sygnałów 0 V–10 V, przegląd czujników oraz wymagania co do dokładności i izolacji torów.

2. STAN WIEDZY I ZAŁOŻENIA PROJEKTOWE

Celem projektu jest przede wszystkim zaprojektowanie i uruchomienie stanowiska laboratoryjnego, a nie opracowanie nowego modelu teoretycznego. Z tego powodu przegląd literatury ma charakter praktyczny i koncentruje się na zagadnieniach, które były rzeczywiście potrzebne podczas projektu: standardzie sygnału 0–10 V w HVAC, podstawowych algorytmach regulacji oraz ogólnych zasadach projektowania płytek PCB dla układów mieszanych analogowo–cyfrowych.

2.1. Sygnał 0–10 V i sterowanie HVAC

W automatyce budynkowej jednym z najczęściej spotykanych sygnałów sterujących jest napięcie 0–10 V. W materiałach producentów czujników, przetworników oraz w opracowaniach dotyczących automatyki budynkowej podkreśla się, że sygnał 0–10 V jest powszechnie używany do sterowania silownikami przepustnic i zaworów, falownikami oraz różnymi czujnikami stosowanymi w instalacjach HVAC[4, 14, 15]. Główna zaleta takiego sygnału to prostota: praktycznie każdy sterownik PLC lub system BMS potrafi wygenerować albo zmierzyć napięcie 0–10 V, a jego interpretacja jest intuicyjna (0–100 % odpowiada 0–10 V).

W literaturze naukowej dotyczącej sterowania HVAC większość uwagi poświęca się modelom cieplnym i algorytmom regulacji, a warstwa sprzętowa jest zwykle opisywana dość ogólnie. W artykułach dotyczących energooszczędnego sterowania HVAC stosuje się klasyczne sygnały analogowe (napięciowe lub prądowe), ale szczegóły toru 0–10 V są zazwyczaj schowane wewnątrz sterowników[3, 16]. W niniejszej pracy ten tor został celowo „wyciągnięty na wierzch” w postaci osobnej płytki dydaktycznej, tak aby można było go obserwować i modyfikować podczas zajęć laboratoryjnych.

2.2. Regulacja PI/PID i sekwencje pracy central HVAC

W publikacjach dotyczących sterowania HVAC najczęściej stosuje się regulatory PI lub PID. Do ich strojenia wykorzystuje się zarówno proste metody (np. Ziegler–Nichols), ale również bardziej zaawansowane algorytmy optymalizacyjne[3, 12, 20]. Przykładowo Almabrok i in. prezentują szybką metodę strojenia regulatora PID dla systemu HVAC z użyciem algorytmu optymalizacyjnego[3]. Zminy termodynamiczne w budynkach są na ogół procesem powolnym, dlatego algorytmy przyspieszające optymalne strojenie regulatorów są głównym motywem przewodnim prac badawczo–naukowych w tej dziedzinie.

W przypadku logiki działania całych central wentylacyjnych, często odwołuje się do ustandardowanych sekwencji pracy opisanych w wytycznych ASHRAE, w szczególności w dokumencie Guideline 36[7]. Wytyczne te definiują gotowe sekwencje sterowania nagrzewnicą, chłodnicą, odzyskiem ciepła i bypassem, uwzględniając pasmo martwe oraz warunki bezpieczeństwa. W projekcie zastosowano uproszczony wariant takiej sekwencji: pojedynczy regulator PI temperatury, którego wyjście jest podzielone na przedziały odpowiadające grzaniu, chłodzeniu, odzyskowi ciepła oraz pasmowi martwemu.

2.3. Sterowniki HVAC na mikrokontrolerach i RTOS

W literaturze można znaleźć przykłady implementacji sterowania HVAC z użyciem mikrokontrolerów oraz prostych systemów operacyjnych czasu rzeczywistego. W pracy Fernandesa opisano regulator PID przepływu powietrza w instalacji wentylacyjnej, zaimplementowany na mikrokontrolerze, z lokalnym pomiarem oraz sterowaniem pracą wentylatora[12]. Układ ten jest zbliżony koncepcyjnie do niniejszego projektu, obejmuje jedną pętlę regulacji, czujnik, element wykonawczy i prosty interfejs użytkownika.

W projekcie jako środowisko firmware'u wykorzystano Zephyr RTOS, lekki system operacyjny czasu rzeczywistego przeznaczony do układów wbudowanych, obsługujący wiele architektur i posiadający rozbudowany zestaw sterowników[25, 26]. Dokumentacja Zephyra pokazuje typowe podejście do struktury aplikacji: logika jest podzielona na wątki, a sprzęt (np. magistrale SPI, wyświetlacze) opisuje się w drzewie urządzeń (Devicetree). Do realizacji interfejsu HMI wykorzystano bibliotekę LVGL, czyli popularną otwartą bibliotekę graficzną dla mikrokontrolerów[18, 24]. LVGL udostępnia gotowe widżety (przyciski, wykresy, listy), dzięki czemu można skupić się na logice HVAC, zamiast implementować od podstaw warstwę graficzną.

2.4. Ramy projektowania PCB dla układów mieszanych

Istotnym elementem projektu jest płytka PCB zawierająca zarówno tory analogowe 0–10 V, jak i cyfrowe interfejsy SPI. W literaturze dotyczącej projektowania układów mieszanych powtarza się kilka podstawowych zaleceń: logiczny podział płytki na część analogową i cyfrową, kontrola powrotu prądów w masie oraz rozsądne prowadzenie zasilania[9, 10, 19]. Ott [19] zwraca uwagę, że zamiast dzielić masę na dwie osobne płaszczyzny, korzystniej jest utrzymać jedną wspólną masę i wydzielić część analogową oraz cyfrową głównie geometrycznie oraz odpowiednim prowadzeniem ścieżek.

W notach aplikacyjnych firm Analog Devices i Microchip przedstawiono przykładowe projekty płyt dla układów mieszanych, gdzie pokazano m.in. sposób umieszczania przetworników ADC na granicy stref analog/digital, prowadzenia linii SPI oraz stosowania przelotek łączących pola masy[9, 10]. W niniejszym projekcie przyjęto podobne podejście: tory 0–10 V znajduje się w wydzielonych częściach płytki z osobnym polem masy analogowej GNDA, interfejsy SPI umieszczone bliżej złącza do płyt Discovery, a przetworniki cyfrowo/analogowe i analogowo/cyfrowe pełnią rolę „mostu” pomiędzy częścią analogową a cyfrową[2, 11].

3. WYMAGANIA PROJEKTU

Zanim przejdzie się do szczegółów dotyczących sprzętu i oprogramowania, warto jednoznacznie określić, czego właściwie oczekuje się od projektowanego stanowiska. Wymagania mają tutaj dwa poziomy: funkcjonalny (co stanowisko ma potrafić) oraz niefunkcjonalny (z jaką dokładnością, w jaki sposób i przy jakiej wygodzie użytkowania). Poniższy zestaw założeń był punktem odniesienia przy projektowaniu zarówno płytki PCB, jak i firmware'u opisanych w kolejnych rozdziałach.

3.0.1. Wymagania funkcjonalne

Podstawowym celem jest budowa stanowiska, które pozwala na wygodny pomiar i generację sygnałów 0–10 V typowych dla instalacji HVAC oraz na rozszerzenie działania prostego układu regulacji temperatury. W praktyce sprowadza się to do spełnienia następujących wymagań:

- Płytnka powinna umożliwiać pomiar napięć wejściowych w zakresie 0–10 V z efektywną rozdzielczością nie gorszą niż 10 mV na kanał, tak aby możliwe było rejestrowanie zmian wielkości fizycznych z dokładnością wystarczającą do ćwiczeń laboratoryjnych.
- Układ ma generować na każdym z ośmiu wyjść sygnał 0–10 V o obciążalności co najmniej 5 mA, co pozwala na sterowanie typowymi przetwornikami i siłownikami 0–10 V spotykanyimi w systemach HVAC (zawory, przepustnice, falowniki itp.).
- Do dyspozycji użytkownika powinno być co najmniej osiem wejść i osiem wyjść analogowych, tak aby dało się równolegle podłączyć kilka czujników i elementów wykonawczych oraz odtworzyć uproszczony schemat małej centrali wentylacyjnej.
- Każdy kanał wejściowy ma udostępniać, oprócz samego sygnału, także dedykowaną masę sygnałową oraz zasilanie +24 V, co umożliwia bezpośrednie podłączenie aktywnych czujników przemysłowych bez dodatkowych zasilaczy pomocniczych.
- Na wyświetlaczu TFT powinny być pokazywane aktualne wartości napięć na wejściach i wyjściach 0–10 V, wraz z ich opisem funkcjonalnym (np. „T supply”, „T extract”, „Heater”, „Cooler”), tak aby z poziomu GUI dało się zorientować, jaką rolę pełni dany kanał w danej konfiguracji.
- Interfejs użytkownika ma umożliwiać ustawienie zadanej temperatury oraz granic pasm sekwencji (grzanie, chłodzenie, odzysk ciepła, pasmo martwe), a następnie obserwację wynikowych sygnałów wyjściowych 0–10 V.
- Oprogramowanie powinno realizować co najmniej jedną pętlę regulacji temperatury (regulator PI/PID w zależności od wybranej konfiguracji) z wyjściem wyrażonym w procentach, które są następnie rozdzielane na wyjścia odpowiadające grzałce, chłodniczy, odzyskowi ciepła i wentylatorowi zgodnie z wybraną sekwencją pracy.
- Użytkownik powinien mieć możliwość przełączania się pomiędzy co najmniej dwiema kompletnymi konfiguracjami stanowiska (różne nastawy, parametry regulatora, mapowanie kanałów, progi sekwencji) bez konieczności rekompilacji oprogramowania – konfiguracja ma być wczytywana z opisu tekstopiowego (JSON) w czasie działania systemu.

Spełnienie powyższych punktów oznacza, że stanowisko może zostać wykorzystane jako prosty, ale dość elastyczny w dalszym rozwoju układ labolatoryjny, pozwalające zarówno na pomiary statyczne, jak i na podstawowe eksperymenty z regulacją temperatury i sekwencjami działania układu HVAC.

3.0.2. Wymagania niefunkcjonalne

Drugą grupę stanowią wymagania niefunkcjonalne, związane z dokładnością, bezpieczeństwem, czytelnością dydaktyczną oraz możliwością dalszego rozwijania projektu. W pracy przyjęto w szczególności następujące założenia:

- Tor pomiarowy 0–10 V powinien zapewniać rozdzielcość co najmniej 10 mV oraz błąd podstawowy rzędu pojedynczych dziesiątek miliwoltów w całym zakresie, tak aby wyniki pomiarów były powtarzalne i nadawały się do prostych zadań obliczeniowych (np. wyznaczanie charakterystyk statycznych czujników).
- Układ ma być odporny na typowe błędy popełniane w laboratorium: odwrotną polaryzację zasilania, przypadkowe zwarcia oraz krótkotrwałe przekroczenia dopuszczalnego napięcia na wejściach/wyjściach. Wymusza to zastosowanie zabezpieczeń w torze zasilania, ochrony przeciwprzepięciowej oraz podstawowych środków ochrony ESD.
- Projekt PCB powinien uwzględniać dobre praktyki dla układów mieszanych: wydzielenie stref analogowej i cyfrowej, kontrolowany sposób łączenia mas, sensowne prowadzenie ścieżek zasilania i linii szybkich (SPI). Dzięki temu stanowisko ma zachowywać się stabilnie także przy jednoczesnej pracy kilku kanałów i podczas dynamicznych zmian sygnałów.
- Zarówno schemat, jak i sam układ na płytce powinny być czytelne dla studenta. Przepływ sygnału (wejścia 0–10 V – przetwornik A/C – mikrokontroler – przetwornik C/A – wyjścia 0–10 V) powinien dać się łatwo prześledzić wzrokowo, a opisy na warstwie nadruku i w GUI mają pomagać w zrozumieniu roli poszczególnych bloków.
- Oprogramowanie powinno być oparte na otwartych, powszechnie stosowanych narzędziach (w tym przypadku: Zephyr RTOS, LVGL) i napisane w sposób umożliwiający dalszy rozwój: dołożenie kolejnych ekranów HMI, nowych algorytmów regulacji lub innego sposobu wczytywania konfiguracji (np. z karty SD) bez konieczności radykalnej przebudowy kodu.
- Projekt nie zakłada pełnej certyfikacji pod kątem EMC, ale sposób prowadzenia mas, filtracja zasilania i zastosowane zabezpieczenia mają ograniczać podatność na zakłócenia do poziomu akceptowalnego w warunkach laboratorium dydaktycznego.

Tak zdefiniowany zestaw wymagań wyznacza ramy dalszej części pracy. Rozdział dotyczący projektu części sprzętowej pokazuje, w jaki sposób spełniono założenia związane z torami 0–10 V, zasilaniem i PCB, natomiast rozdział o firmware opisuje realizację wymagań dotyczących regulacji, interfejsu użytkownika oraz mechanizmu wczytywania konfiguracji.

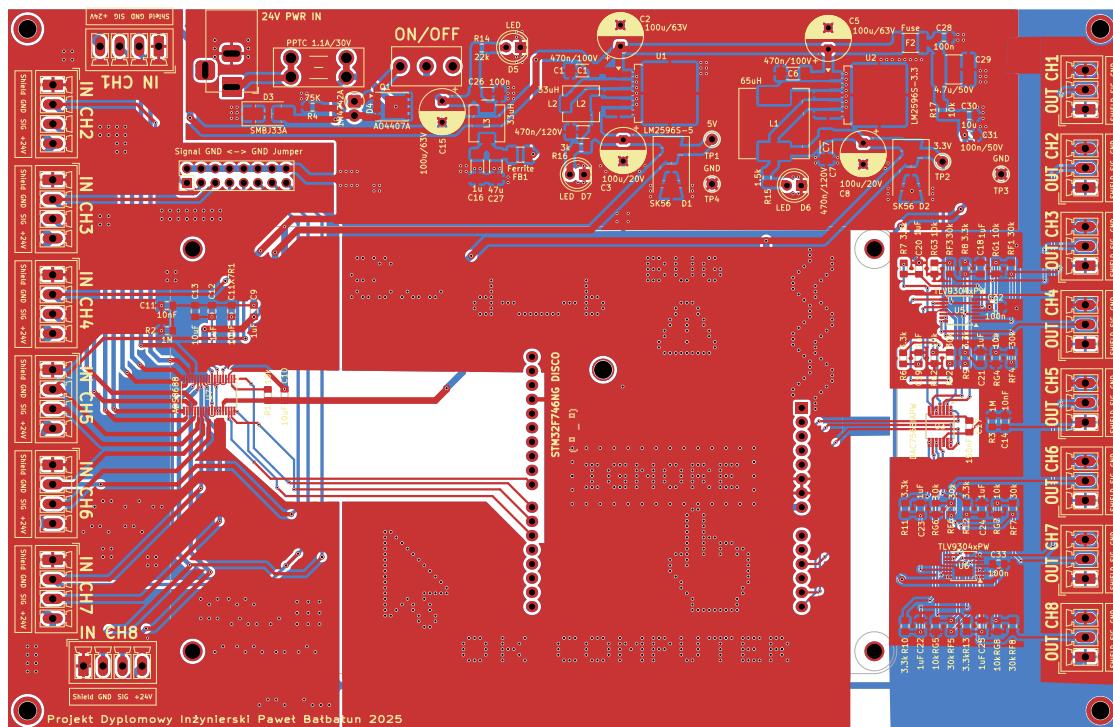
4. PROJEKT CZĘŚCI SPRZĘTOWEJ

4.1. Wymagania sprzętowe — przegląd

Projektowana płytka stanowi jedną, spójną platformę do pomiaru i generacji sygnałów 0–10 V w aplikacjach HVAC[6], współpracującą z zestawem uruchomieniowym STM32F746G-DISCO wyposażonym m.in. w mikrokontroler STM32F7 i panel TFT[21, 22]. Rozwiążanie to ma pełnić rolę uniwersalnego „front-endu” analogowego dla laboratoryjnego sterownika HVAC: umożliwia zarówno rejestrację sygnałów z czujników i przetworników 0–10 V, jak i generację ośmiu niezależnych kanałów 0–10 V do sterowania elementami wykonawczymi (siłowniki przepustnic, zawory mieszające, przetwornice wentylatorów itp.).

Od strony zasilania przewidziano instalacyjne wejście DC (nominalnie 24 V) z podstawowym torem ochronnym (odwrotna polaryzacja, przepięcia, wstępna filtracja), a następnie podział zasilania na osobne gałęzie dla części cyfrowej, analogowej i elementów interfejsowych. Taki układ zmniejsza wpływ zakłóceń na pomiary i stabilizuje pracę torów. Interfejs do świata zewnętrznego obejmuje osiem wejść 0–10 V przygotowanych do bezpiecznego próbkowania przez przetwornik A/C oraz osiem wyjść 0–10 V realizowanych przez przetwornik C/A i wzmacniacze operacyjne.

Istotnym założeniem projektowym było zachowanie kompatybilności elektrycznej i mechanicznej z płytą STM32F746G-DISCO. W centralnej części PCB przewidziano złącze typu goldpin (standard Dupont), którego raster i położenie odpowiada złączu rozszerzeń zestawu Discovery. Dzięki temu całość tworzy układ kanapkowy: płytka z analogowym interfejsem pełni funkcję karty pomiarowej, a zestaw uruchomieniowy zapewnia moc obliczeniową oraz interfejs użytkownika (TFT z panelem dotykowym).



Rysunek 4.1: Widok płytki PCB z zaprojektowanym rozmieszczeniem elementów i złącz. W centralnej części znajduje się obszar montażu płytki STM32F746G-DISCO.

4.2. Moduł zasilania

Układ zasilania płytki został zaprojektowany tak, aby bezpiecznie przyjąć instalacyjne napięcie stałe (do ok. 24 V) i rozdzielić je na dwie stabilne linie robocze: +5 V oraz +3,3 V. Schemat modułu zasilania przedstawiono na rysunku 4.2.

Na wejściu zastosowano gniazdo J1 (DC jack 5,5×2,1 mm), do którego doprowadzane jest napięcie z zewnętrznego zasilacza. Bezpośrednio za złączem znajduje się polimerowy bezpiecznik sa-moresetujący F1 (PPTC 1,1 A/30 V), pełniący rolę zabezpieczenia nadprądowego w przypadku zwarcia na płytce lub błędnego podłączenia odbiorników. Równolegle do wejścia umieszczono diodę TVS D3 (SMBJ33A) tłumiącą przepięcia.

Ochronę przed odwrotną polaryzacją zasilania zrealizowano w oparciu o tranzystor P-MOSFET mocy Q1 (AO4407A w obudowie SO-8 lub równoważny)[5]. Tranzystor włączono w konfiguracji „idealnej diody”: jego źródło jest połączone z wejściem zasilania, dren z resztą układu, a bramka sterowana jest poprzez rezystor R4 i diodę Zenera D4 (1N4742A)[1]. Przy poprawnej polaryzacji tranzystor przewodzi z minimalnym spadkiem napięcia na kanale, a w przypadku odwrotnego podłączenia zasilacza blokuje przepływ prądu i chroni dalsze stopnie zasilania.

Za sekcją ochronną pracuje wyłącznik SW1 odcinający cały moduł zasilania. Dioda LED D5 z rezystorem szeregowym R14 sygnalizuje obecność napięcia po stronie wejściowej; pozwala to na szybką kontrolę stanu zasilania przed przetwornicami.

Kolejnym etapem jest wstępna filtracja C-L-C napięcia wejściowego. Dławik L3 (33 µH) oraz kondensatory C15 (100 µF/63 V), C26 (100 nF), C27 (47 µF) i C16 (1 µF) tworzą filtr typu PI, ograniczający wahania napięcia oraz szpilki prądowe związane z pracą przetwornic impulsowych. Dodatkowy koralik ferrytowy FB1, włączony szeregowo w linii 24 V, poprawia tłumienie zakłóceń o wyższych częstotliwościach, które mogłyby przenikać do dalszych części instalacji.

Konwersję napięcia na poziomy logiczne realizują dwie niezależne przetwornice buck z rodziną LM2596S[17]. Układ U1 (LM2596S-5) generuje linię +5 V. W jego torze znajdują się dławik L2 (33 µH), dioda Schottky’ego D1 (SK56) oraz kondensatory wejściowe C2 (100 µF/63 V) i C1 (470 nF) oraz wyjściowe C4 i C3. Analogicznie układ U2 (LM2596S-3,3) dostarcza linię +3,3 V z użyciem dławika L1 (68 µH), diody D2 (SK56) oraz zestawu kondensatorów C5, C6 po stronie wejściowej i C7, C8 po stronie wyjściowej. Elementy zostały dobrane zgodnie z zaleceniami producenta, tak aby zapewnić stabilność pętli regulacji dla zakładanych obciążień oraz odpowiednio niski poziom tężnień.

Zastosowanie dwóch niezależnych przetwornic zamiast jednego źródła z liniowymi stabilizatorami wtórnymi ma kluczowe znaczenie w kontekście sprawności i wydzielania ciepła. Przy typowych prądach pobieranych przez mikrokontroler, przetwornik A/C i przetwornik C/A, bezpośrednia konwersja z 24 V na 5 V/3,3 V w stabilizatorach liniowych skutkowałaby znaczącym poborem mocy. Przetwornice impulsowe LM2596S pozwalają ograniczyć straty do pojedynczych watów nawet przy pełnym obciążeniu linii 5 V i 3,3 V.

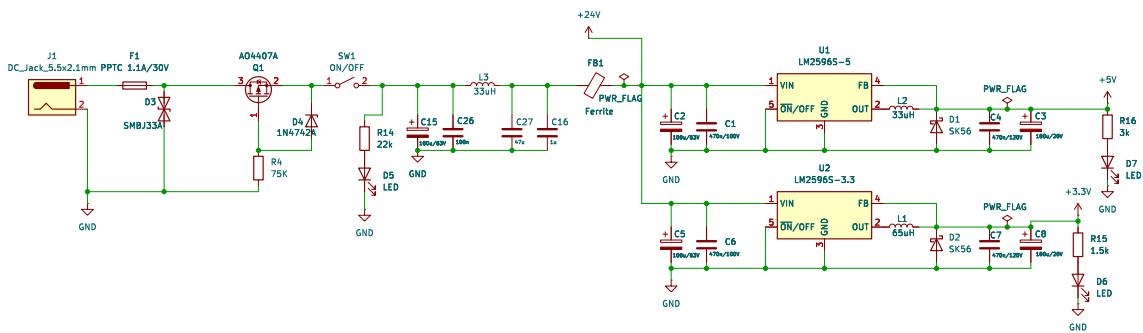
Analogicznie do sygnalizacji obecności napięcia na linii 24 V, stan szyn wyjściowych jest sygnalizowany diodami LED: D7 (dla linii +5 V) z rezystorem R16 oraz D6 (dla linii +3,3 V) z rezystorem R15. Na schemacie umieszczono również znaczniki PWR_FLAG, ułatwiające kontrolę ciągów zasilania w narzędziu CAD i zapobiegające fałszywym ostrzeżeniom o „niezasilonych” sieciach.

Dodatkowo, w bezpośrednim sąsiedztwie przetwornic buck przewidziano punkty testowe dla linii +5 V oraz +3,3 V. Każdy punkt testowy ma obok wyrowadzoną również masę, co umożliwia wygodny pomiar napięć roboczych sondą oscyloskopową lub multimetrem podczas uruchamiania i diagnostyki układu. Lokalizacja tych punktów przy samych przetwornicach pozwala na obserwację rzeczywistych

tętnień i zachowania regulatorów bez dodatkowego wpływu rezystancji i indukcyjności ścieżek zasilających.

Całość tworzy spójny tor: *wejście i zabezpieczenia* → *filtracja wstępna* → *konwersja 24 V na +5 V/+3,3 V* → *dystrybucja i sygnalizacja*, co przekłada się na stabilną pracę układu.

Na poziomie PCB cały moduł zasilania został umieszczony w górnej części płytki (rysunek 4.1), możliwie blisko gniazda wejściowego oraz oddalony od wrażliwych części analogowych. Ścieżki prowadzące prądy impulsowe z przetwornic LM2596S zaprojektowano jako szerokie i możliwie krótkie, z lokalnymi polami masy minimalizującymi powierzchnię pętli prądowych. Dzięki temu ograniczono emisję zakłóceń przewodzonych i promieniowanych oraz uproszczono separację pomiędzy strefą mocy a strefą pomiarową.



Rysunek 4.2: Moduł zasilania płytki z wejściem instalacyjnym 24 V, torem ochrony przed przepięciami i odwrotną polaryzacją oraz przetwornicami step-down LM2596 generującymi linie +5 V i +3,3 V.

4.3. Tor wyjściowy 0–10 V: przetwornik DAC + wzmacniacze operacyjne

Tor wyjściowy generujący sygnały 0–10 V oparto na ośmiokanalowym przetworniku cyfrowo-analogowym U3 (DAC7568IAPW)[11] współpracującym z dwoma czterokanałowymi wzmacniaczami operacyjnymi U5 i U6 (TLV9304xPW)[23]. Dzięki temu możliwe jest niezależne sterowanie wszystkimi ośmioma wyjściami analogowymi. Schemat toru wyjściowego przedstawiono na rysunku 4.3.

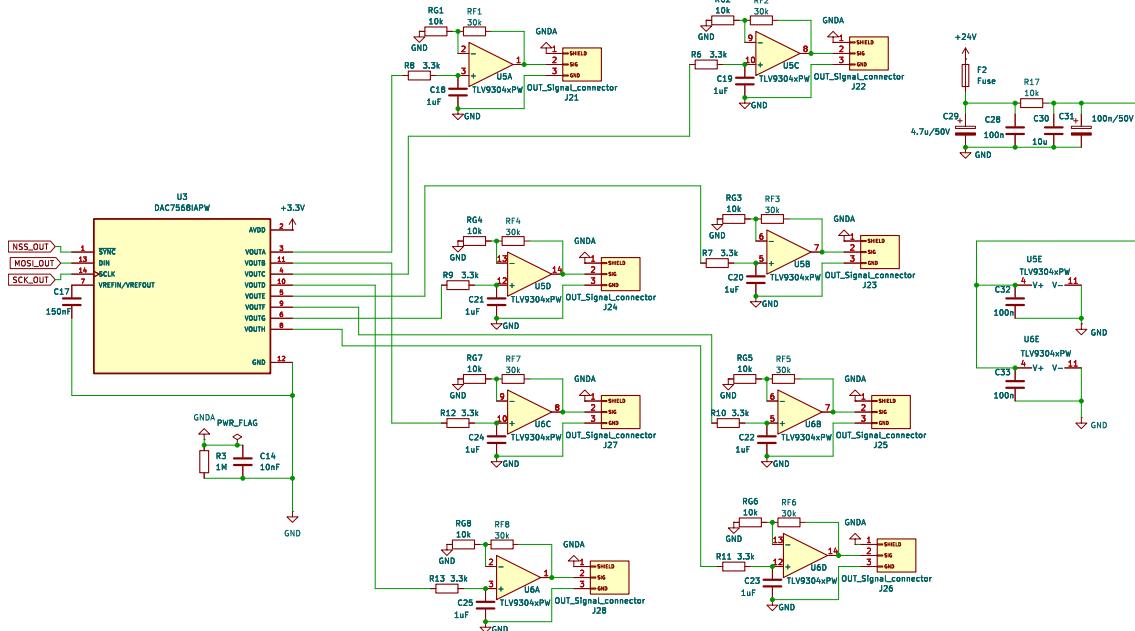
Przetwornik DAC7568 jest zasilany z linii +3,3 V. Komunikację z płytą STM32F746G-DISCO realizuje poprzez magistralę SPI: linie NSS_OUT, MOSI_OUT i SCK_OUT zostały wyprowadzone z MCU i doprowadzone do odpowiednich pinów układu U3. Z uwagi na charakter przetwornika (układ typu „write-only”) nie przewidziano linii MISO; konfiguracja rejestrów i aktualizacja wyjść odbywa się wyłącznie poprzez wysyłanie ramek danych z mikrokontrolera.

Wyprowadzenie VREFIN/VREFOUT służy do ustalenia napięcia referencyjnego. W projekcie wykorzystano wewnętrzne źródło odniesienia przetwornika, dlatego pin został odsprzęgnięty kondensatorem C17 (150 nF) umieszczonym możliwie blisko wyprowadzeń, zgodnie z zaleceniami producenta[11]. Masę części analogowej doprowadzono do masy analogowej GNDA, która na PCB prowadzona jest jako wydzielona wyspa z kontrolowanym połączeniem do wspólnej masy GND poprzez elementy R2/C11 (opisane szerzej w podrozdziale o torze wejściowym).

Każdy z ośmiu kanałów wyjściowych DAC (VOUTA–VOUTH) jest dalej kształtowany przez prosty filtr dolnoprzepustowy RC na wejściu wzmacniacza: rezistor szeregowy (R6–R13, typowo 3,3 kΩ) oraz kondensator do masy (C18, C21–C25, 1 μF). Wyznacza to częstotliwość odcięcia rzędu

$$f_c \approx \frac{1}{2\pi RC} \approx \frac{1}{2\pi \cdot 3,3 \text{ k}\Omega \cdot 1 \mu\text{F}} \approx 48 \text{ Hz}, \quad (4.1)$$

co skutecznie tłumii szum oraz poszarpanie przebiegu pochodzące z aktualizacji DAC, a jednocześnie



Rysunek 4.3: Schemat toru wyjściowego 0–10 V wraz z wzmacniaczami operacyjnymi.

jest w pełni wystarczające dla powolnych procesów w systemach HVAC.

Wzmacniacze U5A–U5D oraz U6A–U6D pracują w konfiguracji nieodwracającej i są zasilane z linii +24 V (wg producenta maksymalne napięcie zasilania to 40 V DC). W ramach zabezpieczenia na torze zasilającym idącym do wzmacniaczy znajduje się polimerowy bezpiecznik samoresetujący F2 oraz filtr dolnoprzepustowy C-R-C składający się z rezystora R17, kondensatorów elektrolitycznych C29 i C31 oraz kondensatorów ceramicznych C28 i C30. Filtr usuwa wszelkie niechciane zakłócenia, które mogą występować w torze zasilania.

Zastosowanie wzmacniaczy o szerokim zakresie napięć zasilania pozwala uzyskać odpowiedni zapas napięciowy dla wyjść 0–10 V bez konieczności stosowania dodatkowych przetwornic podwyższających. Dla każdego kanału zastosowano identyczną sieć sprzężenia zwrotnego: rezistor do masy RG ($10\text{ k}\Omega$) oraz rezistor w pętli sprzężenia RF ($30\text{ k}\Omega$). Wzmocnienie napięciowe pojedynczego toru wynosi więc

$$A_v = 1 + \frac{R_F}{R_G} = 1 + \frac{30\text{ k}\Omega}{10\text{ k}\Omega} = 4. \quad (4.2)$$

Przy referencji DAC rzędu 2,5 V umożliwia to uzyskanie pełnego zakresu 0–10 V na wyjściu wzmacniacza, z zapasem na niewielkie tolerancje i błędy kalibracji.

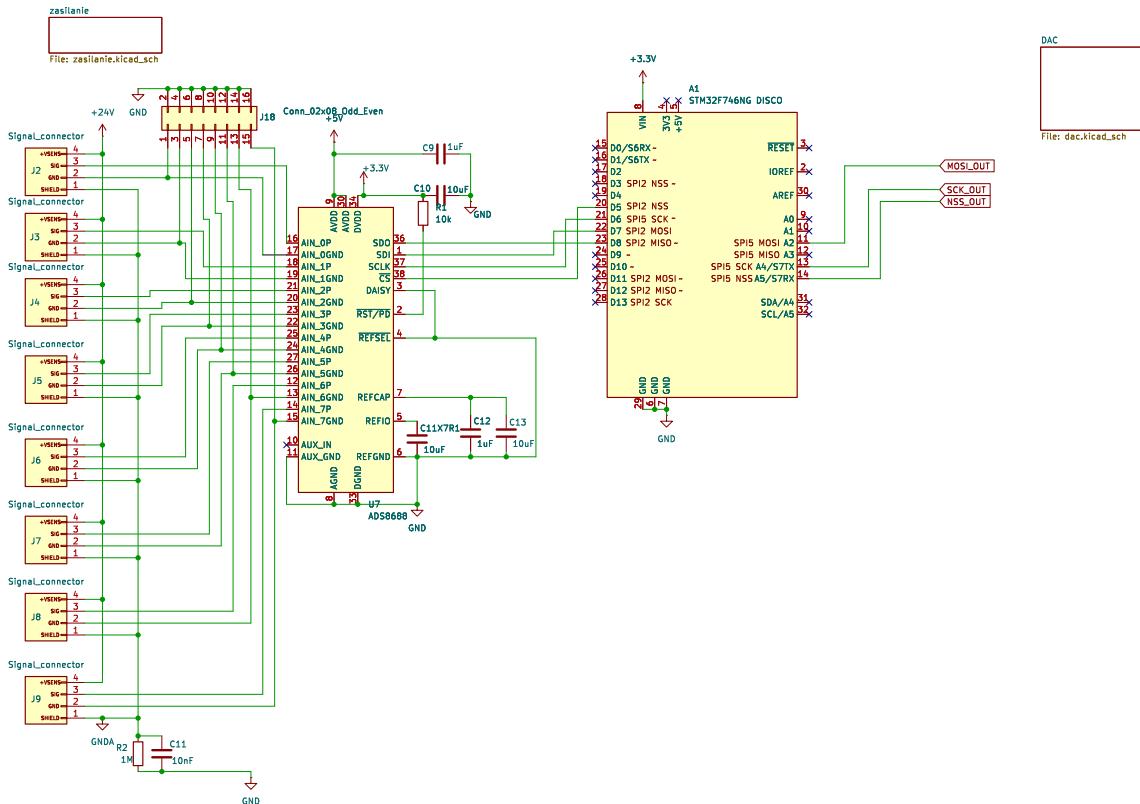
Wyjścia poszczególnych wzmacniaczy są wyprowadzone na uniwersalne złącza J21–J28. Każde złącze udostępnia linię sygnałową 0–10 V, odniesienie SG (signal ground) oraz pin SHIELD przeznaczony do ekranowania przewodów.

Na płytce PCB (rysunek 4.1) złącza wyjściowe zostały rozmieszczone wzdłuż prawej krawędzi w regularnym rastrze, co ułatwia prowadzenie przewodów i daje poczucie „przepływu” sygnałów przez sterownik (od lewej do prawej). Ścieżki sygnałowe pomiędzy wzmacniaczami a złączami są możliwe krótkie i prowadzone nad ciągłą płaszczyzną masy, co redukuje indukcyjność pętli i przesłuchy pomiędzy kanałami.

Podsumowując, tor wyjściowy ma strukturę: *STM32 → DAC7568 → filtr RC → wzmacniacz nieodwracający o wzmocnieniu 4 → złącze sygnałowe z ekranem i zasilaniem pola*, co zapewnia zarówno elastyczność sterowania, jak i zgodność z powszechnie stosowanym standardem 0–10 V.

4.4. Tor wejściowy 0–10 V: interfejs pomiarowy

Tor wejściowy odpowiada za obsługę czujników przemysłowych (pasywnych i aktywnych) oraz bezpieczne doprowadzenie sygnałów 0–10 V do wielokanałowego przetwornika A/C ADS8688[2]. Schemat tej części układu przedstawiono na rysunku 4.4. Przetwornik ten integruje w sobie przełączany multiplexer wejściowy, programowalne zakresy napięciowe oraz wewnętrzny front-end zabezpieczający, co pozwala uprościć zewnętrzny tor analogowy.



Rysunek 4.4: Schemat toru wejściowego 0–10 V wraz z przetwornikami ADC oraz połączenia do mikrokontrolera.

Po lewej stronie schematu rozmiastiono osiem identycznych złączy wejściowych (oznaczonych jako Signal connector, J2–J9). Na każdym złączu wyprowadzono cztery piny:

- SIG — linia sygnałowa 0–10 V,
 - SG — dedykowany powrót sygnałowy (signal ground) dla danego kanału,
 - SHIELD — ekran przewodu, przeznaczony do podłączenia oplotu lub ekranu kabla,
 - +24 V — zasilanie czujnika/konwertera z instalacji 24 V.

Takie ustandaryzowane złącza pozwalają na łatwą zamianę czujników pomiędzy kanałami oraz ograniczają liczbę pomyłek przy okablowaniu.

Linia SIG każdego złącza jest prowadzona bezpośrednio do odpowiedniego pinu wejściowego przetwornika (AIN_xP), natomiast powrót SG trafia do przypisanego ujemnego wejścia odniesienia kanału (AIN_xGND). W ten sposób tworzony jest układ wejściowy o charakterze pseudo-różnicowym, w którym każdy kanał ma swój lokalny powrót odniesiony do tej samej masy analogowej GNDA. Taka topologia kompensuje spadki napięć i przesłuchy na przewodzie powrotnym oraz poprawia odporność

na zakłóczenia wspólnie przy dłuższych odcinkach okablowania, co jest kluczowe w instalacjach HVAC rozproszonych na dużej przestrzeni.

W torze wejściowym nie stosuje się dodatkowych dzielników ani zewnętrznych filtrów antyalasingowych; skalowanie i zabezpieczenie wejść realizuje wewnętrzny front-end przetwornika ADS8688[2], który oferuje kilka przełączanych zakresów pomiarowych oraz obwody ograniczające prądy przy przeięciach.

Piny SHIELD wszystkich wejść są połączone do wspólnego ekranu wylanego pod ścieżkami sygnału i dołączonego względem masy przez filtr RC: rezystor R2 ($1 \text{ M}\Omega$) równolegle z kondensatorem C11 (10 nF) pomiędzy GNDA i GND. Zapewnia to upływ ładunków statycznych oraz tłumienie składowych o wysokiej częstotliwości, a jednocześnie ogranicza stałoprądowe prądy pętli masy i chroni ekran przed „przeciąganiem” potencjału przez inne urządzenia podłączone do tej samej instalacji. Rozwiązanie to stanowi jednocześnie kontrolowany punkt połączenia masy analogowej i cyfrowej — GNDA jest używana w torach wejściowych i wyjściowych, natomiast GND stanowi referencję dla logiki cyfrowej i zasilania 3,3 V.

Część analogowa przetwornika ADS8688[2] jest zasilana z linii +5 V (kondensator C9 $1 \mu\text{F}$ blisko pinów AVDD/AGND), natomiast część cyfrowa z linii +3,3 V (kondensator C10 $10 \mu\text{F}$ przy DVDD/DGND). Linia RST/PD jest podcięgnięta rezystorem $10 \text{ k}\Omega$ do +3,3 V i może być sterowana z mikrokontrolera, co umożliwia programowe resetowanie przetwornika oraz przechodzenie w tryb uśpienia w stanie bezczynności.

Odniesienie napięciowe przetwornika realizowane jest wewnętrznie; piny REFCAP, REFIO i REFGND są odsprzęgnięte kondensatorami klasy X7R (C12 i C13 po $1 \mu\text{F}$) umieszczonymi możliwie blisko wyprowadzeń, zgodnie z zaleceniami producenta[2]. Zapewnia to niskoszumowe, stabilne napięcie odniesienia, co bezpośrednio przekłada się na rozdzielcość efektywną przetwornika.

Na schemacie przewidziano również dodatkowe złącze J18 w postaci listwy goldpin z możliwością założenia zworek. Złącze to służy do konfigurowania połączenia pomiędzy masą sygnałową (SG), wykorzystywaną jako powrót dla wejść 0–10 V, a ogólną masą układu (GND). Poprzez odpowiednie ustawienie zworek użytkownik może zdecydować, czy masa sygnałowa ma pozostać możliwie „czysta” i odniesiona głównie do GNDA (praca z czujnikami pasywnymi), czy też powinna zostać zwarta z masą zasilania w celu zasilania czujników aktywnych, wymagających wspólnego potencjału odniesienia dla toru zasilania i sygnału.

Takie rozwiązanie ma kilka zalet:

- umożliwia elastyczną konfigurację toru wejściowego w zależności od typu podłączonych czujników (pasywne z odseparowaną masą sygnałową vs. aktywne wymagające wspólnego potencjału zasilania i sygnału),
- poprawia kompatybilność z typowymi przetwornikami 0–10 V, które zakładają wspólną masę zasilania i wyjścia napięciowego,
- pozwala na łatwe eksperymentowanie i diagnostykę w warunkach laboratoryjnych — zmiana konfiguracji sprowadza się do przełożenia zworek, bez konieczności modyfikacji PCB ani rozcinania ścieżek.

Komunikacja z mikrokontrolerem odbywa się po magistrali SPI: linie SDI, SDO, SCLK i CS zostały połączone z odpowiednimi pinami MCU. STM32F7 wyposażony jest w dwa oddzielne interfejsy SPI. Dzięki temu oba przetworniki (ADS8688 i DAC7568) mogą komunikować się z MCU niezależnie od siebie; każdy z nich korzysta z własnej magistrali, co upraszcza konfigurację sprzętową i pozwala w przyszłości obsługiwać je w osobnych wątkach oprogramowania.

Na poziomie PCB (rysunek 4.1) przetwornik ADS8688 oraz złącza wejściowe zostały umieszczone w lewej części płytki, w niewielkiej odległości od siebie. Ścieżki sygnałowe SIG/SG prowadzone są nad pełną płaszczyzną masy GNDA, z zachowaniem odstępów pomiędzy kanałami, co ogranicza pojemnościowe sprzężenia krzyżowe. Ekrany SHIELD tworzą osobną, częściowo wylaną strefę połączoną z GNDA poprzez elementy R2/C11.

4.5. Projekt PCB i separacja stref

Widok płytki PCB przedstawiono na rysunku wygenerowanym z narzędzia CAD (rysunek 4.1). Płytki została zaprojektowana jako dwuwarstwowa, z rozległymi polami masy w obu warstwach. Główna powierzchnia zajmowana jest przez prostokątny obrys odpowiadający formatowi płytki Discovery, w którego obrębie przewidziano wycięty obszar (strefę bez elementów i ścieżek) stanowiący miejsce montażu zestawu STM32F746G-DISCO. W czterech narożach oraz w pobliżu złączy umieszczono otwory montażowe umożliwiające sztywne zamocowanie całości do płyty bazowej lub obudowy.

Pod względem funkcjonalnym płytę można podzielić na trzy główne strefy:

1. **Strefa mocy** — w górnej części, obejmująca gniazdo zasilania J1, układ ochrony (F1, D3, D4, Q1, L3, FB1) oraz przetwornice LM2596S. Ścieżki o dużych prądach i impulsowych zmianach prądu prowadzone są lokalnie, nad fragmentarnymi polami masy, co ogranicza emisję zakłóceń.
2. **Strefa analogowa** — przy lewej i prawej krawędzi, obejmująca przetwornik ADS8688, przetwornik DAC7568, wzmacniacze TLV9304 oraz złącza wejściowe i wyjściowe. W tych częściach płytki zastosowano wydzieloną masę GNDA, która jest łączona z ogólną masą GND w jednym punkcie poprzez elementy R2/C11.
3. **Strefa cyfrowa** — w centralnej części, wokół rastru GPIO mikrokontrolera, gdzie prowadzone są linie SPI. Tutaj masa GND jest wylana jako osobna wyspa, a przejścia do masy analogowej są kontrolowane.

Istotnym elementem projektu PCB jest częste uwspólnianie potencjałów masy pomiędzy warstwami i strefami. W torach zasilania oraz w części cyfrowej zastosowano gęsto rozmieszczone przelotki łączące pola GND na obu warstwach, co skraca ścieżkę powrotu prądu i zmniejsza indukcyjność pętli. Analogicznie, w obszarach torów analogowych rozmieszczone przelotki spinające pola GNDA, dzięki czemu masa analogowa tworzy zwartą, dobrze przewodzącą referencję dla sygnałów pomiarowych i wyjściowych. Połączenie GNDA z GND pozostaje jednak zrealizowane w jednym, kontrolowanym punkcie (R2/C11), co łączy zalety wspólnego potencjału odniesienia z ograniczeniem pętli mas i przesłuchów pomiędzy strefą analogową a cyfrową.

Rozdzielenie tych stref w przestrzeni płytki ogranicza przesłuchy pomiędzy torami, a jednocześnie pozwala na intuicyjną analizę układu podczas uruchamiania i diagnostyki. Szerokości ścieżek dobrano zgodnie z przewidywanymi prądami (najszerze dla linii 24 V, 5 V i 3,3 V, węższe dla linii sygnałowych). Przy przejściach pomiędzy warstwami stosowane są przelotki w „gęstych” grupach, tak aby zapewnić zwarty powrót prądu między płaszczyznami masy.

Dodatkowym elementem ułatwiającym pracę z płytą jest rozbudowana warstwa opisowa (silkscreen), na której zaznaczono nazwy złączy, kierunki numeracji pinów, oznaczenia kanałów wejściowych i wyjściowych oraz podstawowe kierunki przepływu sygnału (strzałki). Dzięki temu użytkownik może korzystać z płytki w warunkach laboratoryjnych praktycznie bez konieczności sięgania do schematu.

Opisany w niniejszym rozdziale projekt części sprzętowej stanowi bazę dla dalszych rozważań

dotyczących implementacji algorytmów sterowania i architektury oprogramowania w rozdziałach poświęconych części programowej pracy.

5. FIRMWARE

Część programowa projektu pełni rolę interfejsu pomiędzy opracowaną platformą sprzętową a docelowym użytkownikiem – studentem realizującym ćwiczenia. Jej zadaniem jest nie tylko poprawne sterowanie wyjściami analogowymi i akwizycja sygnałów 0–10 V z przetworników, ale również prezentacja stanu układu w czytelnej formie graficznej, umożliwienie zmiany parametrów regulatora oraz wygodne przełączanie konfiguracji laboratoryjnych bez konieczności rekompilacji oprogramowania.

Z tego powodu firmware nie został zbudowany jako prosta aplikacja „bare–metal” ani w oparciu o szablon generatora STM32CubeMX, lecz jako aplikacja systemu Zephyr RTOS, z wykorzystaniem biblioteki LVGL do budowy interfejsu użytkownika. W kolejnych podrozdziałach opisano motywację tego wyboru, strukturę aplikacji, opis modelu konfiguracji systemu HVAC w oparciu o JSON, logikę regulacji oraz działanie interfejsu HMI.

5.1. Wybór środowiska uruchomieniowego

Projekt wymaga równoległego i deterministycznego wykonywania kilku zadań: odświeżania HMI na wyświetlaczu TFT, generacji sygnałów 0–10 V (komunikacja SPI z przetwornicą DAC) oraz obsługa drugiej magistrali SPI do komunikacji z przetwornicą ADC. Wszystkie te funkcje muszą być realizowane przy zachowaniu powtarzalnych czasów reakcji. Z tych powodów jako środowisko uruchomieniowe wybrano Zephyr RTOS zintegrowany z biblioteką graficzną LVGL.

Zastosowanie systemu czasu rzeczywistego zamiast prostego „bare–metal” pozwala w prosty sposób rozdzielić zadania i przypisać je do osobnych wątków: wątku interfejsu użytkownika, wątków akwizycji i przetwarzania danych oraz wątków odpowiedzialnych za komunikację ze sprzętem. Dzięki temu możliwe jest jednoczesne utrzymanie płynnego odświeżania UI, bezpiecznej obsługi przerwań z przetworników oraz terminowego odświeżania wyjść analogowych sterujących urządzeniami HVAC.

5.1.1. Dlaczego Zephyr + LVGL?

Zephyr udostępnia deterministyczny kernel o prostym, ale wystarczająco elastycznym modelu współbieżności. Wątki o priorytetach, kolejki *workqueue*, przerwania oraz timery wysokiej rozdzielczości umożliwiają zdefiniowanie jasnego podziału zadań i ich wzajemnych zależności. W projekcie wykorzystano tę możliwość do wydzielenia ścieżki krytycznej czasowo (akwizycja i generacja sygnałów) od zadań mniej wrażliwych na opóźnienia (obsługa UI, diagnostyka, ładowanie konfiguracji). Tryb *tickless* ogranicza narzucona związek z tykaniem systemowym, a jednocześnie pozwala zachować stałe okresy odświeżania UI oraz stabilne czasy próbkowania.

Istotną zaletą Zephyra jest spójny ekosystem sterowników oraz wspólna warstwa konfiguracji sprzętu. Standardowe API dla interfejsów SPI, I²C, ADC, DAC/PWM, GPIO czy DMA, uzupełnione mechanizmami DeviceTree i Kconfig, redukuje do minimum ilość kodu „klejącego” między aplikacją a warstwą sprzętową. Dla mikrokontrolerów z rodziny STM32 dostępne są gotowe drivery oraz integracja z biblioteką HAL producenta, co znacząco przyspiesza uruchomienie peryferiów, takich jak kontroler SPI współpracujący z zewnętrznymi przetwornikami A/C i C/A. Zmiana konfiguracji sprzętu (np. inny pin CS, dodatkowy kanał) sprowadza się najczęściej do modyfikacji pliku *.overlay* DeviceTree, bez ingerencji w logikę aplikacyjną.

Zephyr dostarcza ponadto oficjalny subsystem LVGL, odpowiedzialny za integrację biblioteki graficznej z systemem operacyjnym. W praktyce oznacza to gotową obsługę zegara *tick*, wątku renderując-

cego, przydziału pamięci oraz sterowników wyświetlaczy i urządzeń wejściowych. Konfiguracja odbywa się poprzez parę Kconfig+DeviceTree, gdzie określa się m.in. rozmiary buforów ekranu, sposób odświeżania (*flush callback*) oraz mapowanie wejścia dotykowego. Dzięki temu warstwa HMI może zostać zaimplementowana w całości w LVGL, bez konieczności ręcznego "sklejania" sterownika wyświetlacza, sterownika dotyku i logiki zadań systemowych.

Wybór Zephyra jest również uzasadniony dostępnością narzędzi deweloperskich oraz ekosystemu okołosystemowego. Jednolity system budowania oparty na CMake i menedżerze west, wbudowany logger, konsola shell, obsługa trwałych ustawień (NVS), różne systemy plików (FAT, LittleFS) oraz zintegrowany framework testowy (twister) ułatwiają utrzymanie projektu oraz jego automatyzację. W kontekście pracy inżynierskiej ważna jest także skalowalność: Zephyr oferuje stosey sieciowe (Ethernet, BLE, IP), jak również bootloader MCUBoot, co otwiera drogę do przyszłego rozszerzenia urządzenia o zdalne aktualizacje czy komunikację sieciową bez zmiany fundamentów projektu.

Biblioteka LVGL, wykorzystana jako warstwa HMI, jest naturalnym wyborem dla mikrokontrolerów klasy STM32F7 z kolorowym TFT. Zapewnia rozbudowany zestaw widżetów (przyciski, suwaki, listy rozwijane, wykresy) oraz mechanizmy układu obiektów (flex, grid), co pozwala zbudować zarówno ekran diagnostyczny, jak i bardziej złożony panel sterowania. LVGL jest aktywnie rozwijana, dobrze udokumentowana i szeroko stosowana w systemach wbudowanych, a jej integracja z Zephyrem jest oficjalnie wspierana. Użycie powszechnie znanego stosa Zephyr+LVGL zmniejsza ryzyko problemów z utrzymaniem projektu w przyszłości.

Nie bez znaczenia pozostaje kwestia licencjonowania i wsparcia społeczności. Zephyr jest projektem rozwijanym pod licencją Apache-2.0, a LVGL pod licencją MIT. Obie licencje są permisywne i sprzyjają wykorzystaniu w projektach komercyjnych oraz akademickich. Aktywne społeczności użytkowników oraz profesjonalne wsparcie firm współtworzących te projekty ułatwiają rozwiązywanie problemów oraz zapewniają długoterminową stabilność stosa programowego.

5.2. Struktura projektu firmware

Kod aplikacji został zorganizowany w możliwie prosty sposób, z myślą o łatwym uruchomieniu i debogowaniu na płytce STM32F746G-DISCO. Zasadnicza część logiki znajduje się w pojedynczym pliku `main.c`, natomiast zasoby graficzne (ikony i wykresy sekwencji) są przechowywane w osobnych plikach źródłowych wygenerowanych przez konwerter obrazów LVGL. Taka organizacja jest w pełni wystarczająca na etapie prototypu stanowiska laboratoryjnego.

Wejściem do aplikacji jest funkcja `main`, w której inicjalizowany jest sterownik wyświetlacza oraz uruchamiany jest stos LVGL. W tym samym miejscu tworzone są wszystkie ekranы interfejsu: ekran główny (dashboard), wizualizator I/O, ekran wczytywania konfiguracji oraz ekran podglądu sekwencji („Sequence Viewer”). Po utworzeniu obiektów HMI wybierany jest ekran startowy, a wyświetlacz zostaje odblokowany.

Pętla główna ma dwie podstawowe odpowiedzialności. Po pierwsze, cyklicznie wywołuje mechanizmy odświeżania LVGL, dzięki czemu interfejs użytkownika reaguje na zdarzenia i jest dostatecznie płynny. Po drugie, w zadany okresie (w aktualnej implementacji co około 1 s) wykonywana jest aktualizacja wartości kanałów analogowych prezentowanych na ekranie wizualizatora I/O. Do odmierzania czasu wykorzystywany jest systemowy licznik czasu rzeczywistego, a pętla została uzupełniona krótkim opóźnieniem, tak aby nie obciążać niepotrzebnie rdzenia CPU i pozostawić zapas mocy obliczeniowej na dalszy rozwój funkcjonalności.

Niezależnie od pętli głównej działa również oddzielny wątek odpowiedzialny za regulator PI. Wątek ten pracuje z krótkim okresem (około 100 ms), na bieżąco oblicza przyrost czasu między kolejnymi

iteracjami i na tej podstawie wykonuje krok algorytmu regulacji. Rozdzielenie obsługi HMI (wątek główny) od pętli sterowania (osobny wątek RTOS) zapewnia dobrą responsywność interfejsu nawet wtedy, gdy w przyszłości logika regulacji lub komunikacja z przetwornikami zostaną znacząco rozbudowane.

Na początku pliku `main.c` zdefiniowano globalne struktury i zmienne opisujące stan regulatora oraz konfigurację stanowiska. Kluczową rolę pełni struktura konfiguracji HVAC, która zawiera nastawę temperatury, parametry regulatora PI, mapowanie analogowych kanałów wejść i wyjść oraz opis sekwencji grzania i chłodzenia (progi podziału na strefy: chłodzenie, martwą strefę, grzanie, ewentualnie odzysk ciepła/chłodu). Dodatkowe pole tekstowe przechowuje nazwę wariantu sekwencji wykorzystywaną m.in. do wyboru odpowiedniej grafiki na ekranie „Sequence Viewer”. Aktualnie obowiązująca konfiguracja jest przechowywana w jednej globalnej strukturze, a stan wewnętrzny regulatora w osobnej. Wczytanie nowej konfiguracji polega na skopiowaniu jej do struktury globalnej, zresetowaniu stanu regulatora oraz odświeżeniu odpowiednich elementów interfejsu (dashboard, wizualizator I/O, podgląd sekwencji).

Istotnym elementem są również funkcje abstrakcyjne I/O, które odpowiadają za odczyt napięć z wejść analogowych oraz zapis wartości na wyjściach 0–10 V. W obecnej wersji prototypowej funkcje te pełnią rolę atrap: zwracają wartości stałe lub ignorują parametry, co pozwala rozwijać i testować logikę regulatora oraz interfejs LVGL bez konieczności pełnej implementacji obsługi zewnętrznych przetworników A/C i C/A. W docelowej wersji projektu funkcje te będą opakowywać rzeczywistą obsługę urządzeń (ADC i DAC sterowane magistralami SPI) oraz odpowiadać za normalizację napięć 0–10 V na fizyczne wielkości (np. temperaturę, przepływ).

Cała warstwa HMI oparta jest na obiektach LVGL. W kodzie zadeklarowano zestaw wskaźników na potrzeby poszczególnych ekranów: dashboardu, wizualizatora I/O, ekranu wyboru konfiguracji oraz podglądu sekwencji. Dodatkowe tablice wskaźników na etykiety przechowują obiekty tekstowe reprezentujące kolejne kanały wejść i wyjść analogowych na ekranie wizualizatora. Ekran dashboardu korzysta z etykiety wyświetlającej bieżącą nastawę oraz z obiektów reprezentujących pasma sekwencji, którym przypisane są odpowiednie ikony i przyciski do zmiany progów. Ekran „Sequence Viewer” wykorzystuje natomiast pojedynczy obiekt typu obraz, na którym prezentowany jest wykres sekwencji w postaci bitmapy LVGL.

Zasoby graficzne zostały wygenerowane za pomocą konwertera obrazów LVGL i są zapisane w oddzielnym pliku źródłowym. Obejmują one m.in. ikonę nastawy temperatury, ikonę płatka śniegu symbolizującą chłodzenie, ikonę grzałki, ikonę wymiennika ciepła (odzysk) oraz kilka wykresów sekwencji grzania/chłodzenia odpowiadających różnym wariantom konfiguracji. Każdy z tych plików definiuje strukturę opisu obrazu oraz tablicę danych w formacie wymaganym przez LVGL. W `main.c` obrazy są jedynie deklarowane i wiązane z obiektami interfejsu; logika HMI operuje więc na wskaźnikach do gotowych zasobów graficznych, a same bitmapy pozostają w osobnych modułach, co upraszcza ich ewentualną podmianę lub rozszerzanie zestawu ikon.

5.3. Model konfiguracji HVAC i format JSON

Aby umożliwić wygodne przełączanie konfiguracji układu sterowanego bez ponownej komplikacji programu, przygotowano prosty model systemu HVAC zapisany w formacie JSON. Opis ten obejmuje przede wszystkim nastawę temperatury, wzmocnienia regulatora PI, mapowanie kanałów wejść/wyjść analogowych oraz granice pasm sekwencji grzania, chłodzenia, odzysku ciepła i martwej strefy (dead-band). Dzięki temu jedna, tekstowa definicja konfiguracji w pliku JSON wystarcza do pełnego opisania sposobu pracy stanowiska.

W przytoczonym przykładzie struktura zawiera komplet informacji potrzebnych do uruchomienia prostej centrali nawiewno-wywiewnej: nastawę temperatury komfortu (21 °C), parametry regulatora PI

```

static char hvac_config1_json[] =
{
    "{" +
        "  \"setpoint\": 21," +
        "  \"pid\": {" +
            "    \"kp\": 10," +
            "    \"ki\": 2," +
            "    \"kd\": 1" +
        "  }," +
        "  \"seq\": {" +
            "    \"cooling\": {" +
                "      \"from_percent\": -100," +
                "      \"to_percent\": -30" +
            "    }," +
            "    \"deadband\": {" +
                "      \"from_percent\": -30," +
                "      \"to_percent\": 30" +
            "    }," +
            "    \"heating\": {" +
                "      \"from_percent\": 30," +
                "      \"to_percent\": 100" +
            "    }," +
            "    \"heat_recovery\": {" +
                "      \"from_percent\": 0," +
                "      \"to_percent\": 0" +
            "    }" +
        "  }," +
        "  \"sequence_type\": \"cool_dead_heat\"," +
        "  \"io\": {" +
            "    \"t_supply_ai\": 0," +
            "    \"t_extract_ai\": 1," +
            "    \"t_exhaust_ai\": 2," +
            "    \"t_outdoor_ai\": 3," +
            "    \"frost_ao\": -1," +
            "    \"bypass_ao\": -1," +
            "    \"fan_vfd_ao\": 1," +
            "    \"heater_ao\": 2," +
            "    \"cooler_ao\": 3" +
        "  }"
    "}";
}

```

Przykładowa deklaracja kodu JSON opisującego konfigurację HVAC

(kp, ki, kd), opis pasm sekwencji (cooling, heating, heat_recovery, deadband) oraz mapowanie kanałów analogowych. Poszczególne pasma sekwencji są zdefiniowane jako przedziały from_percent/to_percent w zakresie od -100 % do 100 %, wyrażające fragment skali wyjścia regulatora, w którym dane urządzenie ma być aktywne. Mapowanie I/O wskazuje, które fizyczne kanały wejściowe odpowiadają temperaturom nawiewu, wyciągu, wyrzutu i czerpną (t_supply_ai, t_extract_ai, t_exhaust_ai, t_outdoor_ai), a także które kanały wyjściowe sterują nagrzewnicą, chłodnicą, falownikiem wentylatora czy ewentualnym bypassem. Wartość -1 oznacza, że dane wejście lub wyjście jest w danej konfiguracji nieużywane.

Po stronie firmware'u wszystkie te informacje są zebrane w jednej nadrzędnej strukturze konfiguracji, która grupuje nastawę, parametry regulatora, mapowanie I/O, granice pasm sekwencji oraz łańcuch znaków określający wariant sekwencji. Jej uproszczony fragment pokazano poniżej:

```
struct hvac_config {
    int32_t           setpoint;
    struct hvac_pid_cfg pid;
    struct hvac_io_cfg io;
    struct hvac_seq_cfg seq;
    const char        *sequence_type;
};
```

Fragment struktury opisującej konfigurację HVAC w firmware

Parametry regulatora PI (kp, ki, kd) są przechowywane w strukturze konfiguracji jako liczby całkowite. Przy obliczeniach są one rzutowane na typ zmiennoprzecinkowy, co upraszcza implementację i jest w pełni wystarczające w zastosowaniach laboratoryjnych. W aktualnej wersji wykorzystuje się jedynie człony proporcjonalny i całkujący (regulator PI), natomiast pole odpowiadające kd pozostaje zarezerwowane na ewentualną późniejszą rozbudowę do pełnego regulatora PID.

Analogicznie, część odpowiedzialna za mapowanie kanałów analogowych wykorzystuje te same nazwy, które pojawiają się w konfiguracji JSON. Dzięki temu łatwo powiązać zapis tekstowy (np. "t_supply_ai": 0) z interpretacją w programie, a jeden firmware może obsługiwać wiele wariantów okablowania stanowiska. Zmiana przypisania kanału sprowadza się do edycji odpowiedniej liczby w pliku JSON, bez konieczności ingerencji w kod.

Dekodowaniem konfiguracji zapisanej w formacie JSON zajmuje się dedykowana funkcja pomocnicza, której interfejs przedstawiono na listingu ???. Funkcja otrzymuje wskaźnik na łańcuch JSON, jego długość oraz wskaźnik na strukturę konfiguracji, którą należy wypełnić.

```
int hvac_load_config_from_json(const char *json,
                                size_t json_len,
                                struct hvac_config *out_cfg);
```

Interfejs funkcji wczytującej konfigurację z JSON

Wewnątrz funkcji wykorzystywana jest biblioteka JSON dostępna w Zephyr RTOS. Najpierw łańcuch z konfiguracją jest kopowany do lokalnego bufora o ustalonej długości, a następnie parser JSON mapuje poszczególne pola na elementy struktury konfiguracji. Odczytywane są m.in. nastawa temperatury, parametry regulatora PI, opis pasm sekwencji oraz mapowanie kanałów wejść i wyjść analogowych. Funkcja zwraca 0 w przypadku powodzenia, natomiast wartość ujemną w razie błędu (np. gdy przekazany JSON jest zbyt długi lub niekompletny).

Na potrzeby testów i demonstracji przygotowano dwie gotowe konfiguracje, zapisane obecnie jako stałe łańcuchy JSON w pliku `main.c`. Pierwsza z nich (pokazana na listingu ??) ustawia nastawę na 21 °C, definiuje wyraźne pasmo martwej strefy w okolicach zera oraz szerokie pasma dla grzania i chłodzenia. Druga konfiguracja wykorzystuje inną nastawę (23 °C) oraz zmienione granice pasm i wzmacnienia regulatora, co pozwala studentowi porównać zachowanie dwóch różnych nastaw bez wgrywania nowego firmware'u.

Po udanym zdekodowaniu JSON-a tymczasowa struktura konfiguracji jest przekazywana do funkcji, która aktualnia globalne parametry sterownika, resetuje stan regulatora PI oraz odświeża odpowiednie elementy interfejsu graficznego. W przypadku błędu użytkownik otrzymuje czytelny komunikat na ekranie: „Error loading configuration [nr]”. Sam mechanizm pozostaje jednak taki sam: logika sterowania i UI korzystają z jednej, spójnej struktury konfiguracji, natomiast sposób jej wprowadzania (bezpośrednio w kodzie czy z zewnętrznej pamięci) można w przyszłości zmieniać bez potrzeby modyfikowania pozostałych części programu. W dalszej części pracy omówiono próby przeniesienia plików JSON na kartę SD oraz potencjalne kierunki rozwoju tego rozwiązania.

5.4. Próby wczytywania konfiguracji z karty SD

Naturalnym uzupełnieniem modelu konfiguracji opartego na JSON było przeniesienie tych plików z kodu programu do zewnętrznej pamięci, tak aby można było dodawać nowe scenariusze ćwiczeń bez rekompilacji firmware'u. Najprostszym kandydatem jest karta SD, do której użytkownik może łatwo wgrać kolejne pliki `.json` z poziomu komputera. Przykładem takiego podejścia są drukarki 3D i maszyny CNC, które korzystają z kart SD do przechowywania plików *G-code* opisujących zadania do wykonania.

W ramach pracy podjęto pierwsze próby w tej kierunku. Koncepcja zakładała wykorzystanie kontrolera karty SD dostępnego na płytce STM32F746G-DISCO i jego opis w *DeviceTree*, tak aby *Zephyr* mógł zainicjalizować urządzenie przy starcie systemu. W warstwie aplikacyjnej planowano dodać prostą procedurę montowania systemu plików FAT na karcie oraz funkcje, które otwierają pliki JSON o z góry ustalonych nazwach, wczytując ich zawartość do bufora i przekazując ją do istniejącej funkcji dekodującej konfigurację. Z punktu widzenia interfejsu HMI ekran „Config Loader” miałby zyskać dodatkowe przyciski pozwalające na wybór pliku z listy dostępnych konfiguracji.

Praktyczna realizacja tego pomysłu okazała się bardziej problematyczna, niż początkowo zakładano. Konfiguracja kontrolera SD wraz z odpowiednim zestawem pinów i zegarów w *DeviceTree* wymagała dokładnego przeanalizowania schematu płytki, a próby montowania systemu plików FATFS kończyły się niestabilnym działaniem: sporadycznymi błędami inicjalizacji i problemami z poprawnym odczytem danych z karty. Ze względu na ograniczony czas oraz fakt, że głównym celem pracy było uruchomienie kompletnego toru sterowania HVAC z interfejsem HMI, zdecydowano się w finalnej wersji firmware'u pozostawić konfiguracje JSON w postaci wbudowanych łańcuchów znakowych w kodzie.

Mimo że obsługa karty SD nie została doprowadzona do w pełni działającego stanu, prace wykonane na tym etapie nie są całkowicie stracone. Przygotowana konfiguracja systemu plików w `prj.conf`, wstępne wpisy w *DeviceTree* oraz szkice funkcji montujących FATFS i otwierających pliki stanowią dobry punkt wyjścia do dalszego rozwoju stanowiska. W przyszłości możliwe będzie powrócenie do tego tematu, dopracowanie konfiguracji kontrolera SD i dokonanie integracji w taki sposób, aby pliki konfiguracyjne były traktowane analogicznie do plików *G-code* w drukarkach 3D czy maszynach CNC: jako zewnętrzny opis zadania, który można łatwo przygotować i wymienić bez ingerencji w oprogramowanie urządzenia.

5.5. Algorytm regulacji PI i sekwencja pracy układu

Logika sterowania w projekcie jest podzielona na dwie warstwy. Pierwsza to prosty regulator PI, który na podstawie błędu temperatury wylicza sygnał sterujący w postaci wartości procentowej z zakresu od -100 % do 100 %. Druga warstwa to sekwencja grzanie/chłodzenie/odzysk, która zamienia ten sygnał na trzy osobne „kanały” procentowe, przypisane do grzałki, chłodnicy i obiegu odzysku ciepła. Dopiero te wartości są skalowane do napięć 0–10 V i wysyłane na wyjścia analogowe.

Parametry regulatora PI są przechowywane w strukturze konfiguracji (wzmocnienia k_p , k_i oraz obecnie niewykorzystywane k_d), natomiast stan całki jest zapamiętywany w osobnej strukturze stanu. Przy starcie sterowania stan całkujący jest zerowany, a w każdej iteracji pętli regulatora obliczany jest nowy sygnał sterujący na podstawie aktualnego błędu i upływu czasu od poprzedniego kroku. Całkowanie błędu zrealizowano w najprostszej postaci dyskretnej: w każdym kroku do stanu całki dodawany jest iloczyn $k_i \cdot \text{error} \cdot \Delta t$, co odpowiada aproksymacji całki metodą prostokątów (metodą Eulera w przód)[13]. Implementacja jako całość odpowiada klasycznemu regulatorowi PI w formie równoległej, z ograniczeniem sygnału do przedziału -100 %...100 %, zgodnie z typowymi zaleceniami literaturowymi dla regulatorów cyfrowych[8].

Ograniczenie tej wartości do zakresu -100 %...100 %, zapobiega narastaniu całki w sytuacji, gdy wyjście regulatora osiągnęło już fizyczne nasycenie i błąd utrzymuje się przez dłuższy czas. Taki prosty „clamping” stanu całkującego jest najczęściej spotykaną, elementarną formą mechanizmu anti-windup w regulatorach PI implementowanych programowo[8]. Dzięki temu po powrocie układu do obszaru, w którym sygnał sterujący nie jest już ograniczany, regulator nie musi przez długi czas „odrabiać” przeintegrowanego błędu.

Błąd regulacji zdefiniowano jako różnicę między zadaną temperaturą a temperaturą wyciągu:

$$\text{error} = \text{setpoint} - T_{\text{extract}}$$

Dzięki takiemu zapisowi dodatni błąd oznacza, że powietrze w pomieszczeniu jest za zimne i trzeba grzać, a ujemny – że jest za ciepłe i należy włączyć chłodzenie. W prototypie zastosowano uproszczony model czujnika: napięcie na wejściu analogowym jest liniowo przeliczane na temperaturę (przyjęto, że 1 V odpowiada 5 °C). Pozwala to uruchomić i przetestować algorytm sterowania, zanim zostanie zaimplementowana obsługa konkretnych czujników temperatury stosowanych w rzeczywistej instalacji HVAC.

Wyjście regulatora PI w postaci wartości procentowej trafia następnie do bloku sekwencji. Granice poszczególnych pasm (chłodzenie, grzanie, odzysk ciepła, martwa strefa) są zdefiniowane w konfiguracji jako przedziały `from_percent/to_percent`. Algorytm sekwencji najpierw sprawdza martwą strefę: jeżeli wyjście regulatora mieści się w tym zakresie, wszystkie trzy kanały (grzanie, chłodzenie, odzysk) są wyłączone. Zapobiega to oscylowaniu układu w pobliżu punktu pracy i ogranicza liczbę przełączeń urządzeń.

Jeżeli sygnał wyjdzie poza deadband, sprawdzane są kolejno pozostałe pasma. Dla pasma grzania im bliżej górnego końca przedziału, tym większy procentowy sygnał sterujący kierowany na nagrzewnicę; dla pasma chłodzenia zależność jest analogiczna, ale z przeciwniej strony skali (największe wysterowanie przy najbardziej ujemnych wartościach wyjścia PI). Pasmo odzysku ciepła wykorzystuje wartość bezwzględną sygnału: istotne jest wyłącznie odchylenie od nastawy. Pozwala to traktować odzysk jako funkcję „siły” odzysku, niezależnie czy jest to odzysk ciepła czy chłodu.

W ostatnim etapie logiki sterowania każdy z trzech kanałów sekwencji jest przeliczany z procentów na napięcie 0–10 V zgodnie ze wzorem:

$$V = (\text{pct} / 100) * 10 \text{ V}$$

z dodatkowymi ograniczeniami do zakresu 0...10 V. Jeżeli w konfiguracji dane wyjście jest oznaczone jako nieużywane (np. wartością -1) albo jego numer wykracza poza zakres dostępnych kanałów, sygnał nie jest w ogóle wysyłany na przetwornik C/A. Dla wyjścia sterującego falownikiem wentylatora przyjęto prostą zasadę: jeżeli którekolwiek z trzech głównych wyjść (grzanie, chłodzenie, odzysk) ma wartość większą od zera, wentylator otrzymuje stałe napięcie 10 V, w przeciwnym przypadku 0 V.

Cały opisany łańcuch obliczeń jest okresowo wykonywany w osobnym wątku systemu czasu rzeczywistego. Wątek mierzy odstęp czasu między kolejnymi iteracjami, tak aby krok regulacji wykonywać co około 100 ms. Pomiędzy obliczeniami dodano krótkie uśpienie, dzięki czemu procesor ma czas na obsługę innych zadań, w szczególności odświeżanie interfejsu graficznego. Taki prosty harmonogram jest w pełni wystarczający dla stanowiska laboratoryjnego, a jednocześnie na tyle czytelny, że studenci mogą na jego podstawie prześledzić cały przepływ informacji: od pomiaru temperatury, przez regulator PI i sekwencję, aż po napięcia 0–10 V na wyjściach.

5.6. Interfejs użytkownika (HMI)

Interfejs użytkownika został zbudowany w całości w bibliotece LVGL i podzielony na kilka logicznych ekranów: dashboard z nastawami i pasmami sekwencji, wizualizator wejść/wyjść analogowych, prosty „loader” konfiguracji oraz ekran z poglądowym wykresem wybranej sekwencji. Wszystkie ekranы mają wspólny pasek nawigacyjny u góry z czterema przyciskami: „Dashboard”, „I/O”, „Sequence” oraz „Config Loader”. Niezależnie od tego, który ekran jest aktualnie widoczny, użytkownik zawsze ma do dyspozycji ten sam układ przycisków służących do przełączania widoków.

Najważniejszym ekranem z punktu widzenia ćwiczenia jest dashboard. W jego górnej części umieszczono wiersz z przyciskami „Screen 1” i „Screen 2”, które przełączają widok między dwiema podstronami. Pierwsza prezentuje nastawę temperatury oraz pasma sekwencji dla chłodzenia i grzania, druga – pasma martwej strefy i, jeżeli jest w konfiguracji, odzysku ciepła. Taki podział pozwala zmieścić wszystkie elementy w czytelny sposób, bez przeładowania jednego ekranu zbyt dużą liczbą kontrolek.

Nastawa temperatury jest prezentowana w postaci wyraźnej etykiety, poprzedzonej ikoną symbolizującą parametr komfortu cieplnego. Po jej bokach znajdują się przyciski „-” i „+”, którymi użytkownik może zmniejszać lub zwiększać wartość zadanej temperatury. Zmiana wykonana na ekranie jest od razu zapisywana w aktualnej konfiguracji i wykorzystywana przez regulator PI, dzięki czemu wartości widoczne na dashboardzie odpowiadają rzeczywiście używanym parametrom sterowania.

Poniżej nastawy znajdują się wiersze opisujące poszczególne pasma sekwencji. Każdy wiersz składa się z ikony (np. płatka śniegu dla chłodzenia, symbolu grzałki dla ogrzewania czy wymiennika ciepła dla odzysku) oraz kontrolek pozwalających na zmianę progów `from` i `to` w danym paśmie. Progi te są edytowane bezpośrednio z poziomu interfejsu za pomocą przycisków „-” i „+”, a informacje o tym, który przycisk zmienia jaki próg i w jakim paśmie, przechowywane są w małej strukturze kontekstu powiązanej z danym przyciskiem. Algorytm modyfikujący wartości pilnuje, aby progi nie wyszły poza zakres -100 %...100 % i aby dolna granica nie przekraczała górnej. Po każdej zmianie odświeżane są etykiety z wartościami progów.

Następny ekran to wizualizator wejść i wyjść układu. W jego górnym wierszu znajdują się przyciski „Inputs (AI)” i „Outputs (AO)”, które przełączają widoczność dwóch kontenerów z danymi. Widok wejść pokazuje listę kanałów analogowych 0...7, każdy w osobnym wierszu z numerem kanału, aktualną wartością napięcia oraz opisem roli wynikającym z konfiguracji (np. „T supply”, „T extract” albo „Unused”). Widok wyjść jest zbudowany podobnie, lecz dla lepszej czytelności wyjścia podzielono na dwie kolumny, tak aby wszystkie osiem kanałów zmieściło się jednocześnie na ekranie. Aktualne napięcia na wejściach i wyjściach są odświeżane cyklicznie (co około sekundę), dzięki czemu wizualizator

pełni rolę prostego, graficznego miernika pracy przetworników A/C i C/A. Po zmianie konfiguracji role kanałów aktualizują się automatycznie, ponieważ opisy są wyznaczane na podstawie bieżącego mapowania z modelem konfiguracji.

Trzeci ekran, oznaczony w interfejsie jako „Config Loader”, służy do szybkiego przełączania się między przygotowanymi wcześniej konfiguracjami JSON. W centralnej części ekranu umieszczono dwa duże przyciski, odpowiadające dwóm przykładowym konfiguracjom, oraz etykietę z tekstem statusu. Naciśnięcie przycisku powoduje wywołanie funkcji wczytującej konfigurację w formacie JSON (opisanej w poprzedniej sekcji), a następnie zastosowanie nowego zestawu parametrów w globalnej strukturze konfiguracji. Jeżeli operacja się powiedzie, etykieta statusu informuje użytkownika, że dana konfiguracja została załadowana pomyślnie; w przeciwnym razie pojawia się komunikat o błędzie parsowania. W ten sposób student może w trakcie ćwiczenia łatwo przełączyć się między dwoma scenariuszami bez ponownego programowania mikrokontrolera.

Ostatni ekran to „Sequence Viewer”. Jego zadaniem jest graficzna prezentacja sekwencji pracy układu w postaci statycznego obrazka LVGL. Na ekranie znajduje się jeden obiekt typu obraz, a wybrana bitmapa zależy od pola sequence_type w aktualnej konfiguracji. Dla przykładowej konfiguracji o typie "cool_dead_heat" prezentowany jest wykres z wyraźnym pasmem martwej strefy między obszarem chłodzenia i grzania, natomiast dla konfiguracji z odzyskiem ciepła ("cool_rec_dead_rec_heat") dodatkowo pokazane jest pasmo pracy wymiennika. Ekran ten nie jest jeszcze interaktywny, ale pozwala intuicyjnie powiązać liczbowe wartości progów from/to ustawiane na dashboardie z ich graficzną reprezentacją. W planach rozwoju przewiduje się rozszerzenie tego widoku o możliwość dynamicznego reagowania na zmiany progów sekwencji, tak aby wykres odzwierciedlał faktyczną sekwencję oraz podświetlanie aktualnie używanej sekwencji w zależności od pracy regulatora.

5.7. Konfiguracja systemu Zephyr

Oprócz samej logiki sterowania istotnym elementem projektu jest konfiguracja środowiska Zephyr, która decyduje o tym, jakie komponenty systemu operacyjnego są w ogóle dostępne dla aplikacji. W praktyce sprowadza się to do dwóch warstw: pliku `prj.conf`, w którym włącza się odpowiednie moduły i ustawia ich parametry, oraz plików `DeviceTree`, które opisują sprzęt dostępny na płytce STM32F746G-DISCO i sposób jego podłączenia.

W pliku `prj.conf` aktywowano przede wszystkim podsystem graficzny LVGL oraz mechanizmy niezbędne do jego poprawnej pracy: zegar systemowy, obsługę przerwań, wątki, kolejki zdarzeń i logger. Wybrano konfigurację z jądrem `tickless`, która zmniejsza liczbę przerwań od zegara, a jednocześnie pozwala precyzyjnie planować momenty wybudzania wątków. Dla wątku odpowiedzialnego za regulację PI oraz dla wątku obsługującego LVGL ustawiono większe rozmiary stosu niż wartości domyślne, tak aby uniknąć problemów z przepeleniem przy bardziej złożonym interfejsie HMI. Włączono również system logowania umożliwiając diagnostykę działania programu podczas uruchamiania i testów na płytce.

Drugim filarem konfiguracji jest `DeviceTree`. W przypadku STM32F746G-DISCO większość podstawowych peryferiów ma już gotowe opisy dostarczone przez społeczność Zephyra: kontroler wyświetlacza, magistrale SPI, linie GPIO czy kontroler pamięci. W projekcie wykorzystano tą bazową konfigurację i zostały podjęte próby rozszerzenia jej o obsługę karty SD. Przypisano również konkretną magistralę SPI do planowanego przetwornika C/A generującego sygnały 0–10 V oraz drugą magistralę do przetwornika A/C mierzącego napięcia wejściowe. W pliku `overlay` określono piny sygnałów MISO, MOSI, SCK oraz linie CS dla tych urządzeń, tak aby były zgodne ze schematem zaprojektowanej płytki.

Konfiguracja `DeviceTree` określa również parametry wyświetlacza oraz mapowanie wejścia dotykowego, jeżeli w danym wariantie sprzętowym jest ono dostępne. Dzięki temu kod aplikacji nie od-

wołuje się nigdzie do numerów pinów czy rejestrów; korzysta wyłącznie z nazw logicznych urządzeń, które Zephyr wiąże z odpowiednimi driverami na podstawie opisu w DeviceTree. Ewentualna zmiana sprzętu, na przykład przejście na inny przetwornik A/C lub zmianę linii CS, wymaga więc jedynie modyfikacji pliku `.overlay`, a nie przepisywania całej logiki regulacji czy interfejsu HMI.

Taki podział ról pomiędzy `prj.conf` a DeviceTree dobrze wpisuje się w charakter stanowiska laboratoryjnego. Z jednej strony konfiguracja systemu pozostaje zwięzła i daje się utrzymać w jednym repozytorium wraz z kodem aplikacji, z drugiej pozwala dość swobodnie eksperymentować z warstwą sprzętową bez ryzyka przypadkowego popuszczenia logiki programu.

5.8. Wątki systemu czasu rzeczywistego i harmonogram zadań

Użycie RTOS narzuca pewien sposób myślenia o aplikacji: zamiast jednej, monolitycznej pętli program składa się z kilku współbieżnych wątków, z których każdy ma własną odpowiedzialność. W projekcie stanowiska HVAC wykorzystano tę możliwość w dość oszczędnny sposób, ale z wyraźnym podziałem ról.

Podstawowym elementem jest wątek główny, który uruchamia się po starcie mikrokontrolera. To w nim wykonywana jest inicjalizacja wyświetlacza, start strosu LVGL, utworzenie wszystkich ekranów HMI oraz wybór ekranu startowego. Po zakończeniu konfiguracji interfejsu wątek główny przechodzi w prostą pętlę, w której cyklicznie wywoływane są funkcje odpowiedzialne za odświeżanie LVGL oraz aktualizację etykiet wizualizatora I/O. Aktualny okres odświeżania parametrów wejścia/wyjścia jest rzędu jednej sekundy, co zapewnia dobrą czytelność dla użytkownika, a jednocześnie nie obciąża niepotrzebnie procesora. Pomiędzy kolejnymi iteracjami pętli wątek główny usypia się na ustalony czas, pozostawiając rdzeń do dyspozycji innych zadań.

Równolegle działa drugi, osobny wątek, odpowiedzialny za pętlę regulacji PI i sekwencję grzania/chłodzenia. Wątek ten działa z krótszym okresem, rzędu 100 ms. Każde przebudzenie rozpoczyna od odczytania aktualnego czasu systemowego i obliczenia odstępu od poprzedniej iteracji, tak aby krok całkowania w regulatorze był wykonywany z poprawnym δt . Następnie wątek odczytuje zmierzona temperaturę z wybranego kanału wejściowego, przelicza ją na stopnie Celsjusza, wyznacza błąd względem nastawy, wykonuje krok regulacji PI oraz oblicza wynikowe sygnały sterujące dla grzania, chłodzenia i odzysku ciepła. Na końcu wątek skaluje te wartości do zakresu 0–10 V i, o ile dane wyjścia są aktywne w konfiguracji, przekazuje je do warstwy I/O odpowiedzialnej za docelowe przetworniki.

Priorytet wątku regulacji został ustawiony tak, aby miał on pierwszeństwo przed zadaniami mniej wrażliwymi na opóźnienia, takimi jak aktualizacja etykiet na ekranie. Dzięki temu nawet przy intensywniejszym korzystaniu z interfejsu graficznego, przewijaniu list czy zmianie konfiguracji, podstawowa logika sterowania zachowuje w miarę stabilny okres próbkowania. Z drugiej strony częstotliwości próbkowania dobrano świadomie na dość niskim poziomie, typowym dla układów HVAC, w których stałe czasowe są długie i nie ma potrzeby aktualizowania sterowania co kilka milisekund. Taki kompromis upraszcza harmonogram zadań i ogranicza ryzyko problemów wynikających z niedoszacowania obciążień.

W obecnej wersji nie wprowadzano dodatkowych wątków do obsługi przetworników A/C i C/A czy innej komunikacji. Ich obsługa zostanie w przyszłości w naturalny sposób włączona do wątku regulacji (dla ścieżki sterowania) oraz wątku głównego (dla wizualizacji). Jeżeli w toku dalszego rozwoju okaże się, że to za mało, Zephyr pozwala stosunkowo łatwo wydzielić kolejne zadania do osobnych wątków, nadać im priorytety i dołożyć proste mechanizmy synchronizacji.

5.9. Abstrakcja wejścia/wyjścia analogowych i integracja z przetwornikami

Jednym z założeń projektowych było rozdzielenie logiki sterowania od szczegółów sprzętowych związanych z przetwornikami A/C i C/A. Z punktu widzenia regulatora PI i warstwy HMI interesujące są jedynie wielkości fizyczne: temperatura, procent wysterowania grzałki, wartość napięcia na wyjściu sterującym falownikiem. To, czy odczyt temperatury pochodzi z konkretnego kanału przetwornika A/C konkretnego producenta, powinno pozostać ukryte w jednej, spójnej warstwie pośredniej.

W firmware przyjęto prosty model abstrakcji wejścia/wyjścia. Dla każdego kanału wejściowego AI dostępna jest funkcja zwracająca napięcie 0–10 V w postaci liczby zmiennoprzecinkowej, a dla każdego kanału wyjściowego AO funkcja przyjmująca żąданie napięcie w tym samym zakresie. Kod regulatora posługuje się wyłącznie tymi prostymi interfejsami. Przeliczenie napięć na temperaturę i odwrotnie jest zrealizowane w jednym miejscu, według obecnie przyjętego liniowego modelu (1 V odpowiada 5 °C). Zmiana typu czujnika temperatury w przeszłości, na przykład na przetwornik z wyjściem 0–10 V o innej charakterystyce, wymaga więc tylko modyfikacji tej funkcji, a nie całego algorytmu sterowania.

W aktualnej wersji prototypowej funkcje odczytu i zapisu napięć są celowo uproszczone. Zwierają wartości stałe lub pomijają parametry, dzięki czemu można było równolegle rozwijać interfejs LVGL, model JSON i logikę regulacji, nie czekając na pełne uruchomienie warstwy sprzętowej. W docelowej wersji projektu ich implementacja zostanie rozszerzona o wywołania driverów korzystających z magistral SPI skonfigurowanych w DeviceTree. Dla przetwornika A/C będzie to typowy schemat: rozpoczęcie konwersji, odczyt próbki, przeliczenie na napięcie oraz ewentualna filtracja; dla przetwornika C/A odpowiednio: przeskalowanie żadanego napięcia na kod cyfrowy i przesłanie go interfejsem SPI.

Takie podejście ma dwie zalety. Po pierwsze, studenci analizujący kod firmware'u nie są od razu konfrontowani z mało czytelnymi sekwencjami konfiguracji rejestrów przetwornika; mogą skupić się na logice HVAC, a warstwę niskopoziomową potraktować jako osobny, zamknięty moduł. Po drugie, ułatwia to utrzymanie projektu: ewentualna wymiana przetwornika na inny model, lepiej dostępny na rynku, będzie wymagała jedynie modyfikacji warstwy I/O, bez ingerencji w pozostałe części programu.

5.10. Diagnostyka i logowanie

Przy pracy nad oprogramowaniem na mikrokontroler równie ważne, co sama logika sterowania, jest możliwość sprawdzenia, co dokładnie dzieje się w środku. W projekcie wykorzystano w tym celu wbudowany system logowania Zephyra, który pozwala wysyłać komunikaty tekstowe z programu na interfejs konsoli (w tym przypadku przez UART).

Logowanie zostało włączone w `prj.conf`, a dla modułu aplikacji ustwiono poziom pozwalający na wyświetlanie komunikatów informacyjnych i ostrzeżeń. W kluczowych miejscach programu wstawiono komunikaty sygnalizujące między innymi start i zakończenie inicjalizacji, załadowanie konfiguracji JSON, treść aktualnie wczytanej konfiguracji (nastawa, wzmacniania regulatora, granice pasm sekwencji, przypisania kanałów I/O) oraz ewentualne błędy parsowania. Dzięki temu podczas uruchamiania na płytce można na bieżąco obserwować, czy firmware rzeczywiście interpretuje konfigurację zgodnie z oczekiwaniemi.

Komunikaty logowania okazały się szczególnie przydatne w trakcie prac nad mechanizmem przełączania konfiguracji. W początkowej wersji występował problem polegający na tym, że wybrany JSON dawało się wczytać poprawnie tylko raz, a kolejne próby kończyły się błędem. Dzięki loggerowi udało się zawęzić obszar poszukiwań do fragmentu kodu odpowiedzialnego za buforowanie łańcucha JSON i zarządzanie jego długością. Po poprawkach mechanizm stał się stabilny, a logi służą do weryfikacji, którą konfigurację użytkownik wybrał i jakie dokładnie wartości parametrów zostały w niej ustawiione.

Zastosowanie standardowego loggera Zephyra ma tę zaletę, że w przyszłości łatwo można przełączyć backend na inny, na przykład na zapis do pliku na karcie SD lub przesyłanie logów przez sieć. W kontekście stanowiska laboratoryjnego otwiera to możliwość rejestrowania przebiegu ćwiczenia: zapisywania historii zmian nastawy, odpowiedzi układu i ewentualnych komunikatów o błędach, co może być wartościowym uzupełnieniem materiałów dydaktycznych.

5.11. Testy i walidacja działania firmware'u

Opis logiki sterowania i interfejsu HMI nie byłby pełny bez chociażby podstawowej weryfikacji, że oprogramowanie pracuje zgodnie z założeniami. Z uwagi na charakter projektu oraz ograniczony czas testy miały głównie charakter manualny, ale zostały przeprowadzone w sposób uporządkowany, tak aby sprawdzić kluczowe funkcje stanowiska.

Pierwszym etapem była walidacja modelu konfiguracji JSON. Przygotowano kilka wariantów konfiguracji różniących się nastawą, wzmacnieniami regulatora oraz granicami pasm sekwencji. Dla każdego wariantu sprawdzano, czy funkcja wczytująca JSON poprawnie wypełnia strukturę konfiguracji, a logi systemowe pokazują dokładnie te wartości, które zostały zapisane w pliku. Testowano także reakcję na typowe błędy: brak któregoś z pól, błędne typy danych czy zbyt dużą zawartość. W takich przypadkach aplikacja powinna zasygnalizować problem komunikatem na ekranie „Config Loader” i nie zmieniać obowiązującej konfiguracji.

Kolejnym krokiem były testy algorytmu regulacji PI i sekwencji. Ponieważ wejścia analogowe są w prototypie uproszczone, symulowano różne wartości temperatury poprzez odpowiednie ustawienie zwracanego napięcia w funkcjach odczytu. Dla kilku charakterystycznych scenariuszy, na przykład „temperatura dużo poniżej nastawy”, „temperatura w martwej strefie”, „temperatura powyżej nastawy”, obserwowano wartości sygnału sterującego oraz wynikowe napięcia na kanałach grzania, chłodzenia i odzysku. Weryfikowano, czy sygnał grzania pojawia się tylko przy ujemnym błędzie, chłodzenie przy dodatnim, a w martwej strefie wszystkie wyjścia są wyłączone. Dodatkowo sprawdzano, czy włączenie któregoś z głównych kanałów powoduje uaktywnienie wyjścia wentylatora z wartością 10 V, zgodnie z przyjętym uproszczonym modelem.

Trzecia grupa testów dotyczyła interfejsu użytkownika. Sprawdzano, czy zmiana nastawy na dashboardzie jest natychmiast odzwierciedlana w strukturze konfiguracji i w kolejnych obliczeniach regulatora, czy przyciski służące do modyfikacji progów pasm sekwencji poprawnie aktualizują wyświetlane wartości i nie pozwalają wyjść poza zakres -100 %...100 %. Dla wizualizatora I/O weryfikowano, czy po przełączeniu się między ekranami wejścia i wyjścia zachowana jest czytelność układu (kanały AO w dwóch kolumnach), a opisy roli każdego kanału zgadzają się z mapowaniem z aktualnej konfiguracji.

Ostatnim krokiem była ocena ogólnej responsywności systemu. W trakcie intensywnego przełączania ekranów, zmiany nastawy i progów sekwencji oraz wielokrotnego ładowania konfiguracji obserwowano, czy interfejs zachowuje płynność, a wątki regulacji nie ulegają widocznemu „zamrożeniu”. Dzięki rozdzieleniu zadań na osobne wątki oraz umiarkowanym częstotliwościami odświeżania udało się osiągnąć dostatecznie stabilne działanie.

5.12. Ograniczenia obecnej implementacji i kierunki rozwoju

Zastosowany w projekcie firmware spełnia podstawowe założenia: pozwala wczytać konfigurację HVAC z opisu JSON, przeprowadzić regulację PI z sekwencyjnym podziałem na pasma grzania, chłodzenia i odzysku oraz zobrazować bieżący stan układu na czytelnym interfejsie HMI. Jednocześnie w aktualnej formie zawiera kilka świadomych ograniczeń, wynikających zarówno z zakresu pracy inżynierskiej, jak i z dostępnego czasu.

Najbardziej widocznym kompromisem jest brak pełnej integracji z rzeczywistymi przetwornikami A/C i C/A. Warstwa I/O została przygotowana w postaci abstrakcyjnych funkcji operujących na napięciach 0–10 V, ale w omawianej wersji zwracają one wartości testowe lub ignorują parametry wyjściowe. Pozwala to prezentować i analizować całą logikę sterowania i interfejs użytkownika, jednak nie daje jeszcze możliwości bezpośredniego podłączenia stanowiska do fizycznych czujników i elementów wykonawczych. Podobnie uproszczony jest model czujników temperatury: liniowa zależność $1\text{ V} = 5\text{ }^{\circ}\text{C}$ jest wystarczająca do celów dydaktycznych, ale w rzeczywistej instalacji HVAC należałoby ją zastąpić charakterystyką konkretnego przetwornika lub czujnika.

Drugim istotnym ograniczeniem jest niedokończona obsługa karty SD i systemu plików FAT. Choć przygotowano wstępную konfigurację i podjęto próby montowania systemu plików, finalna wersja firmware'u korzysta wciąż z wbudowanych łańcuchów JSON. Konfiguracje są więc komplikowane razem z programem, a dodanie nowego scenariusza ćwiczenia wymaga przebudowania firmware'u. Z punktu widzenia użytkownika końcowego, którym ma być student na stanowisku laboratoryjnym, ważniejsze było jednak to, aby samo stanowisko działało stabilnie.

Kolejnym obszarem, w którym świadomie uproszczono rozwiązanie, jest warstwa wizualizacji sekwencji. Ekran „Sequence Viewer” korzysta z przygotowanych statycznych obrazów, dobranych na podstawie typu sekwencji zapisanego w konfiguracji. Nie reaguje jeszcze dynamicznie na zmiany progów from/to wprowadzane na dashboardzie, choć od strony algorytmicznej nic nie stoi na przeszkodzie, aby taką funkcjonalność dopisać.

Wszystkie te ograniczenia można potraktować jako naturalne punkty wyjścia do dalszego rozwoju stanowiska. W pierwszej kolejności planowane jest dokończenie integracji z rzeczywistymi przetwornikami A/C i C/A, tak aby stan wizualizowany na ekranie odpowiadał rzeczywistym napięciom na zaciskach płytka. Równolegle warto wrócić do koncepcji wczytywania konfiguracji z karty SD: docelowo pliki JSON mogłyby być przygotowywane i wgrywane podobnie jak pliki konfiguracyjne do drukarek 3D czy maszyn CNC, co ułatwiłoby prowadzącemu zajęcia szybką zmianę scenariusza ćwiczeń.

Interesującym kierunkiem rozwoju jest także uczynienie widoku sekwencji w pełni interaktywnym. Zamiast statycznej bitmapy LVGL mógłby on dynamicznie rysować wykres w oparciu o aktualne wartości progów zapisane w konfiguracji, a nawet pokazywać bieżący punkt pracy regulatora. Takie rozwiązanie byłoby bardziej przejrzyste dydaktycznie i mogłoby pomóc studentom lepiej zrozumieć zależności pomiędzy nastawami a zachowaniem układu. W dalszej perspektywie możliwe jest również dołączenie prostego interfejsu sieciowego, który pozwoliłby na podgląd i modyfikację nastaw z poziomu przeglądarki lub zewnętrznej aplikacji, korzystając z wbudowanych w Zephyra stosów komunikacyjnych.

Podsumowując, obecna wersja firmware'u realizuje trzon funkcjonalności potrzebny do prowadzenia ćwiczeń laboratoryjnych i stanowi dobrą bazę pod dalszą pracę. Zaprojektowana struktura kodu, rozdzielenie warstw abstrakcji i wybór stosu Zephyr+LVGL sprawiają, że rozwijanie stanowiska o kolejne funkcje powinno być możliwe bez gruntownej przebudowy istniejących modułów.

5.13. Walidacja

5.13.1. Metodyka

Źródła wzorcowe (kalibrator napięcia), obciążenie dla wyjścia 0–10 V, środowisko testowe.

5.13.2. Wyniki

Tabele dokładności, histogram odchyлеń, niepewność typu A/B, budżet niepewności.

5.13.3. Dyskusja

Ograniczenia, dryft temperaturowy, histereza, propozycje ulepszeń.

BIBLIOGRAFIA

- [1] *1N4728A Thru 1N4764A 1 W Zener Diodes*. Diody Zenera serii 1N47xxA stosowane w torze ochrony wejścia zasilania (D4). GOOD-ARK Electronics. 2024. URL: <https://www.goodark.com/specification/1N4728%20thru%201N4764.pdf> (cyt. na s. 8).
- [2] *ADS8688 12-Bit, 500-kSPS, 8-Channel Data Acquisition System*. Zewnętrzny ADC U7. Texas Instruments. 2021. URL: <https://www.ti.com/lit/ds/symlink/ads8688.pdf> (cyt. na s. 4, 11, 12).
- [3] Abdoalnasir Almabrok, Marios Psarakis i Anastasios Dounis. „Fast Tuning of the PID Controller in an HVAC System Using the Big Bang–Big Crunch Algorithm and FPGA Technology”. W: *Algorithms* 11.10 (2018), s. 146. doi: 10.3390/a11100146. URL: <https://doi.org/10.3390/a11100146> (cyt. na s. 3).
- [4] *Ambient Temperature Sensor-PT1000 with 0-10V Output*. Czujnik temperatury otoczenia z przetwornikiem PT1000 na sygnał 0–10 V. Seven Sensor Solutions. 2023. URL: <https://www.sevensensor.com/ambient-temperature-sensor-pt1000-u> (cyt. na s. 3).
- [5] *AO4407A P-Channel Enhancement Mode Field Effect Transistor*. Tranzystor P-MOSFET mocy do ochrony przed odwrotną polaryzacją (Q1). Alpha i Omega Semiconductor. 2024. URL: https://www.aosmd.com/res/data_sheets/AO4407A.pdf (cyt. na s. 8).
- [6] *Application Guidelines for 0–10 V Control Interfaces in HVAC*. ASHRAE. 2019 (cyt. na s. 7).
- [7] ASHRAE. *ASHRAE Guideline 36-2021: High-Performance Sequences of Operation for HVAC Systems*. Guideline 36-2021. American Society of Heating, Refrigerating i Air-Conditioning Engineers. 2021. URL: <https://www.ashrae.org/technical-resources/bookstore/guideline-36-2018> (cyt. na s. 3).
- [8] Karl J. Astrom i Tore Hagglund. *PID Controllers: Theory, Design, and Tuning*. 2 wyd. Research Triangle Park, NC: Instrument Society of America, 1995 (cyt. na s. 21).
- [9] Bonnie C. Baker. *Analog and Interface Guide – Volume 1: A Compilation of Technical Articles and Design Notes*. Zbiór not aplikacyjnych o projektowaniu układów analogowych i PCB. Microchip Technology Inc. 2005. URL: <https://ww1.microchip.com/downloads/en/devicedoc/00924b.pdf> (cyt. na s. 4).
- [10] *Considerations for Mixed Signal Circuit Board Design (AN-404)*. Application Note AN-404. Analog Devices, Inc. 2000. URL: <https://www.analog.com/media/en/technical-documentation/application-notes/495266810an-404.pdf> (cyt. na s. 4).
- [11] *DAC7568 12-Bit, Octal, Buffered Voltage Output DAC*. Zewnętrzny DAC U3. Texas Instruments. 2017. URL: <https://www.ti.com/lit/ds/symlink/dac7568.pdf> (cyt. na s. 4, 9).
- [12] A. C. Fernandes i in. „Control of airflow in ventilation systems using embedded systems on microcontrollers”. W: *Microsystem Technologies* 25.9 (2019), s. 3597–3608. doi: 10.1007/s00542-019-04407-1. URL: <https://doi.org/10.1007/s00542-019-04407-1> (cyt. na s. 3).
- [13] Finn Haugen. *Discretization of Simulator, Filter, and PID Controller*. Spraw. tech. Online article. TechTeach, 2010. URL: <https://www.mic-journal.no/PDF/ref/Haugen2010.pdf> (cyt. na s. 21).

- [14] *HD & HO Series Deluxe Humidity Transmitters*. Aktywne czujniki wilgotności/temperatury z wyjściem 0–5 V / 0–10 V. Veris Industries. 2014. URL: https://www.veris.com/ASSETS/DOCUMENTS/ITEMS/EN/HD_HO_d0321.pdf (cyt. na s. 3).
- [15] *High Accuracy Pressure Transmitter, 0–10 V Output*. Nadajnik ciśnienia z wyjściem napięciowym 0–10 V do systemów automatyki. Jakar Electronics. 2020. URL: <https://www.jakar.cz/en/p/high-accuracy-pressure-transmitter/4103> (cyt. na s. 3).
- [16] V. Kirubakaran i in. „Energy efficient model based algorithm for control of building HVAC systems”. W: *Ecotoxicology and Environmental Safety* 121 (2015), s. 236–243. doi: [10.1016/j.ecoenv.2015.03.027](https://doi.org/10.1016/j.ecoenv.2015.03.027) (cyt. na s. 3).
- [17] *LM2596 SIMPLE SWITCHER 3-A Step-Down Voltage Regulator*. Regulatory zasilania U1 (5 V) i U2 (3.3 V). Texas Instruments. 2020. URL: <https://www.ti.com/lit/ds/symlink/lm2596.pdf> (cyt. na s. 8).
- [18] *LVGL – Light and Versatile Embedded Graphics Library*. <https://lvgl.io/>. Oficjalna strona projektu LVGL, dostęp: 2025-12-03. 2025 (cyt. na s. 4).
- [19] Henry W. Ott. „Partitioning and Layout of a Mixed-Signal PCB”. W: *Printed Circuit Design* 18.6 (2001). Klasyczny artykuł o podziale mas i prowadzeniu ścieżek w układach mieszanych, s. 16–26. URL: https://www.hottconsultants.com/pdf_files/mixed-signal.pdf (cyt. na s. 4).
- [20] Guangji Qu i Mohammed Zaheer-uddin. „Real-time tuning of PI controllers in HVAC systems”. W: *International Journal of Energy Research* 28.15 (2004), s. 1313–1327. doi: [10.1002/er.1030](https://doi.org/10.1002/er.1030). URL: <https://doi.org/10.1002/er.1030> (cyt. na s. 3).
- [21] *STM32F746xG Datasheet*. MCU zastosowany na płytce STM32F746G-Discovery. STMicroelectronics. 2023. URL: <https://www.st.com/en/microcontrollers-microprocessors/stm32f746ng.html> (cyt. na s. 7).
- [22] STMicroelectronics. *32F746GDISCOVERY—Discovery kit with STM32F746NG MCU*. 2025. URL: <https://www.st.com/en/evaluation-tools/32f746gdiscovery.html> (term. wiz. 10.10.2025) (cyt. na s. 7).
- [23] *TLVx9304 Low-Noise, Rail-to-Rail Output Operational Amplifiers*. Bufor analogowy U5, U6. Texas Instruments. 2023. URL: <https://www.ti.com/lit/ds/symlink/tlv9304.pdf> (cyt. na s. 9).
- [24] *Zephyr Integration Guide – LVGL Documentation*. <https://docs.lvgl.io/9.1/integration/os/zephyr.html>. Opis integracji LVGL z Zephyr RTOS, dostęp: 2025-12-03. 2024 (cyt. na s. 4).
- [25] *Zephyr Project Documentation*. <https://docs.zephyrproject.org/latest/>. Dokumentacja systemu Zephyr RTOS, dostęp: 2025-12-03. 2025 (cyt. na s. 4).
- [26] *Zephyr Project RTOS*. <https://www.zephyrproject.org/>. Dostęp: 2025-12-03. 2025 (cyt. na s. 4).