

Politechnika Warszawska

W Y D Z I A Ł E L E K T R Y C Z N Y



Instytut Sterowania i Elektroniki Przemysłowej

Praca dyplomowa magisterska

na kierunku Informatyka Stosowana
w specjalności Inżynieria Oprogramowania

Dotykowy pulpit do monitorowania oraz sterowania pracą
przekształników energoelektronicznych

Adam Chorągwicki

Numer albumu 294358

promotor
dr inż. Piotr Grzejszczak

konsultant części laboratoryjnej
mgr inż. Marek Szymczak

WARSZAWA 2020

DOTYKOWY PULPIT DO MONITOROWANIA ORAZ STEROWANIA PRACĄ PRZEKSZTAŁNIKÓW ENERGOELEKTRONICZNYCH

Streszczenie

Niniejsza praca dotyczy opracowania dotykowego pulpitu do monitorowania oraz sterowania pracą przekształników energoelektronicznych. Dotykowy pulpit ma na celu ułatwienie uruchamiania i testowania przekształników energoelektronicznych dzięki wizualizacji parametrów urządzenia na wygodnym w użyciu panelu dotykowym. Wizualizacja odbywa się zarówno w formie tekstu jak i graficznej. Użytkownik poza odczytem parametrów urządzenia ma również możliwość efektywnego sterowania przekształnikiem poprzez modyfikację jego parametrów pracy z poziomu panelu dotykowego.

Do realizacji pracy w roli panelu dotykowego wykorzystano płytę STM32F469 Discovery z wbudowanym wyświetlaczem dotykowym, zaś w roli urządzenia sterującego przekształnikiem energoelektronicznym, czyli sterownika, zastosowano płytę Nucleo-F411RE. W implementacji oprogramowania płytek pomocny okazał się IAR Embedded Workbench umożliwiający łatwą komplikację i debuggowanie kodu oraz środowisko TouchGFX umożliwiające wygenerowanie graficznego interfejsu użytkownika wyświetlanego na ekranie dotykowym płytki STM32F469 Discovery. Dodatkowo na etapie implementacji, a później testowania systemu przydatna okazała się komputerowa aplikacja testowa stworzona w środowisku QtCreator.

Praca składa się z 7 rozdziałów oraz 4 dodatków, które razem stanowią kompletną dokumentację wykonanego projektu. Rozdział 1 to krótki wstęp wprowadzający do tematyki ekranów dotykowych, opisujący i uzasadniający cel pracy. Rozdział drugi przedstawia cele pracy, założenia projektowe, ograniczenia oraz planowane funkcjonalności systemu. Trzeci rozdział składa się z opisu stanowisk testowych użytych przy realizacji pracy, w tym użytego sprzętu, oprogramowania oraz schematu połączeń. Na czwarty rozdział składa się szczegółowy opis opracowanego modelu komunikacji w systemie w tym schemat pakietu UART, znaczenia jego pól oraz dopuszczalne wartości. Piąty rozdział opisuje zrealizowane funkcjonalności systemu wraz z podaniem przykładowego pakietu UART dla każdej z funkcjonalności. Rozdział szósty stanowi kompletny opis przebiegu testowania komunikacji mikrokontroler-mikrokontroler w systemie, opisuje implementację aplikacji testowej na płytce Nucleo-F411RE, w tym schemat zastosowanych połączeń oraz opis zestawu komend służących do testowania komunikacji między STM32F469 Discovery a Nucleo-F411RE, wraz z opisem procedur użycia komend testowych i rysunkami przedstawiającymi wiadomości diagnostyczne obu płyt przesłane w trakcie testów. Dodatek A opisuje strukturę aplikacji testowej użytej do bieżącego testowania systemu w trakcie implementacji, w tym opis funkcjonalności poszczególnych grup widżetów. Dodatek B zawiera zrzuty ekranów aplikacji testowej na różnych etapach działania. Dodatek C opisuje strukturę oprogramowania płytka STM32F469 Discovery ze szczególnym uwzględnieniem systemu FreeRTOS oraz modelu oprogramowania TouchGFX. Dodatek D stanowi zapis pseudokodu mogącego posłużyć jako szablon pomocny w przeniesieniu kodu aplikacji testowej do dowolnego mikrokontrolera. Tekst niniejszej pracy podsumowuje zestaw wniosków w rozdziale 7 oraz bibliografia składająca się z zestawu odnośników do źródeł użytych w trakcie wykonywania pracy.

Słowa kluczowe: system mikroprocesorowy z STM32, panel dotykowy, monitorowanie przekształników, interfejs sterowania

Chorągwicu Adam

TOUCH SCREEN PANEL FOR MONITORING AND CONTROL OF POWER CONVERTERS

Abstract

The thesis concerns a touch screen panel for monitoring and control of power converters. The touch screen panel is supposed to facilitate testing and usage of power converters by visualizing the device parameters on a convenient to use touch panel. Visualization takes place in both text and graphic form. In addition to reading the device parameters, user can also effectively control the converter by modifying its operating parameters from the touch panel.

For the thesis, STM32F469 Discovery board with a built-in touch display was used as the touch panel, while the Nucleo-F411RE board was used as the digital controller of the power converter. The IAR Embedded Workbench IDE proved to be helpful for compiling and debugging of the code during software implementation phase, as well as the TouchGFX environment enabling generation of a graphical user interface displayed on the touch screen of a STM32F469 Discovery board. Additionally, at the stage of implementation and later testing of the system, a computer test application created in the QtCreator environment proved to be useful.

The thesis consists of 7 chapters and 4 appendices constituting complete documentation of the project. Chapter 1 is a short introduction to a subject of touch screen panels describing and justifying the aim of the thesis. Chapter 2 presents the aim of the thesis, design assumptions, limitations and planned system functionalities. Chapter 3 consists of description of test stands utilized during creation of the thesis, including utilized hardware, software and connection scheme. Chapter 4 presents detailed description of the designed communication model in the system, including schema of the UART packet, meaning of its field and acceptable values. Chapter 5 describes the developed functionalities of the system, including exemplary UART packets for each functionality. Chapter 6 is a complete description of the testing process of microcontroller-microcontroller communication in the system, describes the implementation of the test application on the Nucleo-F411RE board, including the connection diagram used and a description of the set of commands used to test communication between STM32F469 Discovery and Nucleo-F411RE, along with a description of the procedures of using test commands and drawings showing diagnostic messages sent during tests by both boards. Appendix A describes the structure of the test application used for testing during the implementation phase, including a description of the functionality of individual groups of widgets. Appendix B contains screenshots of the test application at various stages of execution. Appendix C presents the structure of the STM32F469 Discovery software with particular emphasis on the FreeRTOS system support and the TouchGFX software model. Appendix D is a record of a pseudocode which can potentially be used as a template for porting application to any microcontroller. The text of the thesis is summarized with the set of conclusions in chapter 7 and the bibliography consisting of a set of references to external sources utilized during development of the thesis.

Keywords: STM32 microprocessor system, touch screen panel, power converters monitoring, control interface

Charynichu Adlam



Politechnika Warszawska

załącznik nr 3 do zarządzenia nr 24/2016 Rektora PW

Warszawa, 30.06.2020
miejscowość i data

Adam Chorągwicki
imię i nazwisko studenta

294358
numer albumu

Informatyka Stosowana
kierunek studiów

OŚWIADCZENIE

Świadomy odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w systemie iSOD są identyczne.

Chorągwicki Adam
czytelny podpis studenta

Spis treści

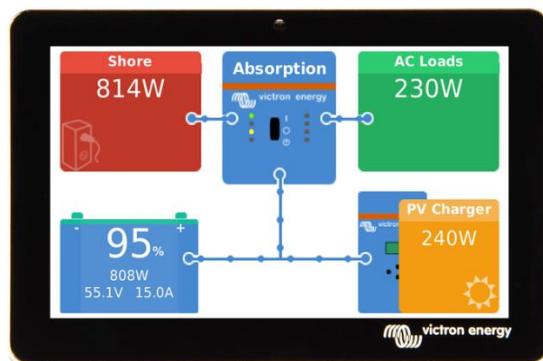
1	Wstęp.....	1
2	Cel pracy.....	4
3	Opis systemu, stanowiska testowego, użytego sprzętu i oprogramowania	6
3.1	Schemat docelowego systemu.....	6
3.2	Prezentacja stanowiska testowego - etap 1.....	6
3.3	Prezentacja stanowiska testowego - etap 2.....	8
3.4	Specyfikacja użytych urządzeń	10
3.5	Lista użytego oprogramowania.....	12
4	Opis komunikacji w systemie.....	14
4.1	Pakiet UART – opis ogólny	14
4.2	Konwersja liczbowo-znakowa.....	14
4.3	Pakiet UART – zakresy wartości	15
4.4	Pakiet UART – znaczenie pól.....	15
4.5	Prędkość transmisji	18
5	Opis funkcji systemu	19
6	Test komunikacji panel-sterownik.....	22
6.1	Strategia testowa	22
6.2	Zmodyfikowana aplikacja testowa	24
6.3	Ekrany pulpitu dotykowego	25
6.4	Przebieg testów.....	30
7	Wnioski	43
	Bibliografia	45
	Dodatek A Aplikacja testowa PC	46
	Dodatek B Ekrany aplikacji testowej.....	51
	Dodatek C Oprogramowanie płytki STM32F469-Discovery	54
	Przygotowanie środowiska	54
	Zadania systemu FreeRTOS	54
	Obsługa modelu TouchGFX MVP	55
	Ekrany TouchGFX	55
	Diagnostyka błędów.....	56
	Dodatek D Pseudokod aplikacji testowej sterownika	58
	Pakiet UART i jego struktura danych	58
	Pseudokod poszczególnych funkcjonalności systemu	59
	Schematy blokowe funkcjonalności systemu	67

1 Wstęp

Przekształtnik energoelektroniczny jest urządzeniem służącym do przekształcania energii elektrycznej poprzez bezpośrednią modyfikację przebiegu czasowego prądu elektrycznego [1]. Przyrządy tego typu można w sposób najbardziej ogólny podzielić na prostowniki, falowniki, przekształtniki prądu stałego na stały oraz przemienniki częstotliwości [2]. Są to urządzenia skomplikowane, wymagające dedykowanych układów sterowania i pomiarów, które powszechnie realizowane są przez procesy sygnałowe (ang. Digital Signal Processing - DSP) [3].

Istnieje wiele opatentowanych, skomplikowanych metod sterowania przekształtnikami energoelektronicznymi np. patent US8736234 [4] będący projektem aparatury sterującej przekształtnika kontrolującego generator prądu zmiennego w samochodach lub patent US8400125 [5] będący projektem sterowania przekształtnikami wielofazowymi. Tego typu standardowe metody monitorowania i sterowania przekształtnikami są mało intuicyjne i niezbyt atrakcyjne z punktu widzenia użytkownika. Z pomocą może tu jednak przyjść rozwój technologii, a konkretnie współczesnych powszechnych metod łączących w sposób przystępny sterowanie i wizualizację danych – ekranów dotykowych.

Istnieją obecnie na rynku rozwiązania komercyjne umożliwiające sterowanie i monitorowanie parametrów innych urządzeń z użyciem panelu dotykowego np. GX Touch 50¹ firmy Victron Energy stanowiący część systemu fotowoltaicznego firmy Victron Energy umożliwiającego pozyskiwanie energii elektrycznej ze światła słonecznego. Nie istnieją jednak obecnie rozwiązania tego typu dedykowane dla przekształtników energoelektronicznych.



Rys. 1.1 Panel dotykowy Victron Energy Gx Touch 50²

Niniejszy dokument stanowi kompletną dokumentację projektu realizującego funkcje sterowania i monitorowania przekształtnika energoelektronicznego z użyciem dotykowego pulpitu. W projekcie założono, że dotykowy pulpit ma umożliwić użytkownikowi uruchomienie i efektywne monitorowanie parametrów pracy przekształtnika w czasie rzeczywistym. Uruchomienie przekształtnika ma wiązać się z inicjalizacją dotykowego pulpitu informacjami o nazwach jak i wartościach liczbowych parametrów roboczych urządzenia. Parametry te mają być na bieżąco odbierane z przekształtnika i wizualizowane zarówno w formie tekstopisowej jak i graficznej - w postaci diagramu rysowanego w czasie rzeczywistym. Dodatkowo użytkownik

¹ <https://www.victronenergy.com/panel-systems-remote-monitoring/gx-touch-50> (25.06.2020 r.).

² https://www.victronenergy.pl/upload/cache/1580982740_upload_documents_775_500-GX-Touch-50-%28front-with-screen%29.png (28.06.2020 r.)

ma mieć możliwość bezpośredniej ingerencji w pracę przekształtnika poprzez modyfikację dowolnego parametru pracy urządzenia z poziomu dotyковego pulpitu. Tego typu modyfikacje parametrów umożliwiają kalibrację systemów pomiarowych czy też konfiguracje zabezpieczeń np. poprzez ustawienie maksymalnych wartości prądu lub napięcia, których urządzenie nie powinno przekraczać. Ponadto kluczowym jest, by użytkownik mógł na bieżąco dobierać nastawy regulatorów wpływających na główną funkcjonalność przekształtnika np. poprzez ustawienie docelowego napięcia, natężenia prądu czy częstotliwości przebiegu czasowego prądu elektrycznego już po jego przekształceniu. Dodatkowo wizualizacja danych w postaci wykresu powinna pomóc użytkownikowi w obserwacji ewentualnych nieprawidłowości w działaniu przekształtnika.

Wszystkie wymienione funkcjonalności mają być realizowane z użyciem pulpitu dotykowego, którego sercem ma być ekran dotykowy. Wyświetlacze dotykowe bardzo powszechnie stosowane są od stosunkowo niedawna, mimo że technologia jako taka znana jest już od kilkudziesięciu lat.

Pierwszy ekran dotykowy zaprojektował i opatentował 7 października 1975 amerykański wynalazca George Samuel Hurst - numer patentu US3911215³. W tamtych czasach zapewne nie podejrzewał on jak rewolucyjnym stanie się jego wynalazek z biegiem lat.

Technologia ekranów dotykowych rozwijała się przez wiele lat stopniowo pojawiając się w coraz większej ilości produktów, jednakże swój szczyt popularności osiągnęła dopiero w ostatniej dekadzie. Obecnie ze względu na dużą dostępność i niską cenę tej technologii ekran dotykowe są wszechobecne – coraz trudniej spotkać użytkownika telefonu korzystającego z urządzenia z klasycznym zestawem klawiszy i niewielkim ekranem.

Fenomen technologii ekranów dotykowych polega na połączeniu możliwości wyświetlania informacji oraz sterowania urządzeniem w obrębie jednej spójnej – większej lub mniejszej – płaskiej powierzchni. Dzięki temu nie ma potrzeby dołączania do każdego wyświetlacza dodatkowego urządzenia sterującego w postaci myszki lub klawiatury, gdyż zarówno operację kliku myszki jak i wyboru klawisza można łatwo zrealizować przy użyciu ekranu dotykowego.



Rys. 1.2 Współczesny ekran dotykowy⁴

³ <https://patents.google.com/patent/US3911215> (2.06.2020 r.).

⁴ <https://i.insider.com/5db8832addee019279924bd86?width=1300&format=jpeg&auto=webp> (2.06.2020 r.).

Szczególnie ciekawą funkcjonalność można osiągnąć łącząc powyższe cechy wyświetlaczy dotykowych z programowalnym urządzeniem – mikrokontrolerem. Można w ten sposób uzyskać niewielkie wielofunkcyjne urządzenie potrafiące wyświetlać dane w atrakcyjny dla użytkownika sposób. Dodatkowo możliwe jest intuicyjne sterowanie urządzeniem w szerokim zakresie wartości nieosiągalne w innych przypadkach bez użycia klasycznej klawiatury czy myszki.

Zakres możliwych do osiągnięcia funkcjonalności znacznie zwiększa się, gdy mikrokontroler z wbudowanym wyświetlaczem dotykowym staje się częścią większego systemu w praktyce sterując innymi wyspecjalizowanymi urządzeniami. Właśnie taki przypadek użycia jest tematem niniejszej pracy.

2 Cel pracy

Praca ma 3 główne cele:

- Opracowanie modelu komunikacji między dotykovym pulpitem i sterownikiem przekształtnika energoelektronicznego
- Implementacja oprogramowania dotyковego pulpitu umożliwiającego interakcję z użytkownikiem, a jednocześnie komunikację ze sterownikiem przekształtnika energoelektronicznego
- Przetestowanie funkcjonalności oprogramowania dotyковego pulpitu w komunikacji z innym układem mikroprocesorowym

W niniejszej pracy dotykowy pulpit stanowić będzie mikrokontroler firmy ST Microelectronics STM32F469NI Discovery. Posiada on wbudowany 4-calowy wyświetlacz dotykowy, bardzo wydajny mikroprocesor Cortex-M4 o maksymalnej częstotliwości taktowania 180Mhz oraz niezbędne peryferia komunikacyjne potrzebne do komunikacji z innymi urządzeniami.

Sterownik przekształtnika energoelektronicznego na potrzeby niniejszej pracy będzie symulować inną płytka rozwojowa – Nucleo-F411RE wyposażona w słabszy mikrokontroler STM32F411RE, niemniej jednak posiadający wystarczającą ilość pamięci RAM, flash oraz interfejsów UART na potrzeby niniejszej pracy.

Komunikacja między urządzeniami w systemie odbywać się będzie przy użyciu UART – Universal Asynchronous Receiver Transmitter. Obydwie płytki mają wbudowane układy scalone realizujące funkcje UART, natomiast po stronie komputera PC funkcje te realizuje konwerter USB-UART.

W pierwszym etapie projektu dotykowy pulpit STM32F469 połączony interfejsem UART bezpośrednio z komputerem PC razem tworzyć będą stanowisko eksperymentalne umożliwiające łatwe i wygodne przetestowanie wszystkich funkcjonalności systemu przy użyciu komputerowej aplikacji testowej. Aplikacja testowa ma za zadanie symulować sterownik przekształtnika energoelektronicznego. Szczegóły etapu pierwszego opisano w podrozdziale 3.2.

W dalszej fazie projektu dotykowy pulpit STM32F469 połączony interfejsem UART z płytą Nucleo-F411RE razem tworzyć będą stanowisko testowe sterowane z poziomu komputera PC poprzez zmodyfikowaną wersję aplikacji testowej. Umożliwi to łatwe i wygodne przetestowanie docelowej komunikacji mikrokontroler-mikrokontroler. Szczegóły etapu drugiego opisano w podrozdziale 3.3.

Planowane funkcjonalności systemu:

- Z poziomu ekranu dotyковego modułu STM32F469:
 - a) Aktywowanie parametru w aplikacji
 - b) Deaktywowanie parametru w aplikacji
 - c) Ustawienie wartości parametru w aplikacji
 - d) Wyświetlenie wartości odebranych z aplikacji w postaci grafu

- Z poziomu aplikacji komputerowej:
 - a) Inicjalizacja i deinicjalizacja połączenia
 - b) Możliwość wysłania pakietu o dowolnej zawartości do modułu STM32F469
 - c) Możliwość odebrania pakietu o dowolnej zawartości od modułu STM32F469
 - d) Logowanie wysłanych i odebranych pakietów w aplikacji testowej z możliwością podglądu
 - e) Możliwość łatwego wysyłania zbioru pakietów danych w podanym zakresie

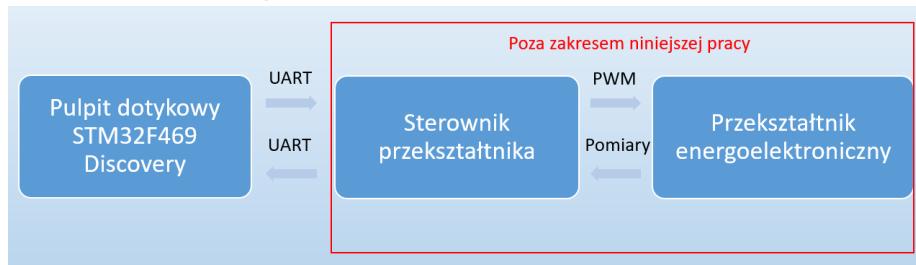
Oprogramowanie modułu STM32F469 Discovery powstałe w toku realizacji niniejszej pracy ma przy użyciu wbudowanego 4-calowego wyświetlacza TFT-LCD realizować wygodny i intuicyjny w obsłudze panel dotykowy umożliwiający użytkownikowi zarówno przesyłanie danych do przekształtnika jak i odczyt danych odebranych z przekształtnika.

Dane odebrane z przekształtnika mogą być wizualizowane na pulpicie dotykowym w postaci tekstowej lub graficznej tj. w postaci grafu rysowanego w czasie rzeczywistym. Wizualizowane dane mogą reprezentować nazwy oraz wartości liczbowe dowolnych parametrów pracy odczytyanych z przekształtnika np. napięcie, prąd, moc, częstotliwość. Dane przesyłane z pulpitu dotykowego do przekształtnika mogą być wartościami parametrów pracy przekształtnika jakie użytkownik chce ustawić w urządzeniu.

Dzięki możliwości wspomnianego dwukierunkowego przesyłania danych użytkownik będzie miał możliwość monitorowania dowolnych parametrów pracy przekształtnika w czasie rzeczywistym jak i sterowania jego pracą poprzez modyfikację wartości dowolnych parametrów co umożliwi np. kalibrację wejść pomiarowych przekształtnika.

3 Opis systemu, stanowiska testowego, użytego sprzętu i oprogramowania

3.1 Schemat docelowego systemu

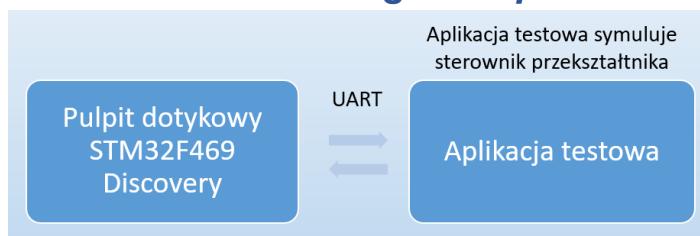


Rys. 3.1 Schemat docelowego systemu

Docelowy system składa się z płytki STM32F469 Discovery połączonej interfejsem UART ze sterownikiem mikroprocesorowym przekształtnika energoelektrycznego, z kolei sterownik ten jest połączony bezpośrednio z przekształtnikiem energoelektrycznym.

Sterownik, jego oprogramowanie jak i sam przekształtnik oraz sposób ich połączenia są poza zakresem niniejszej pracy. W celu zrealizowania projektu elementy zaznaczone na Rys. 3.1 zostały dwuetapowo zastąpione swoimi ekwiwalentami funkcjonalnymi – w etapie nr 1 komputerową aplikacją testową, natomiast w etapie nr 2 płytka Nucleo-F411RE symulującą sterownik przekształtnika.

3.2 Prezentacja stanowiska testowego - etap 1

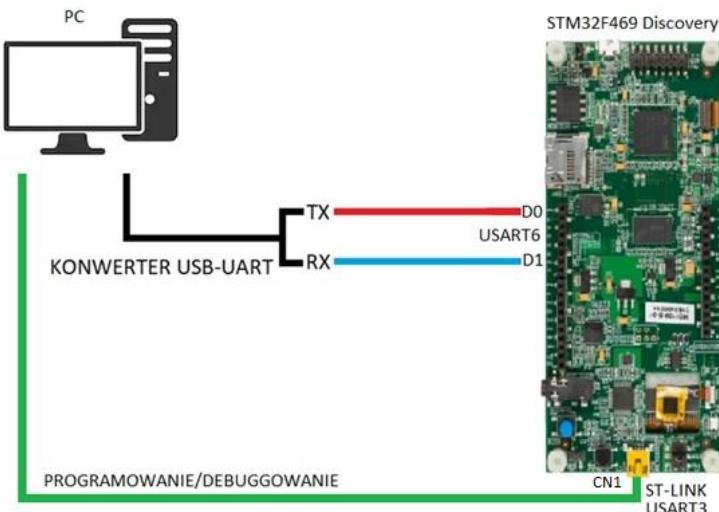


Rys. 3.2 Schemat stanowiska testowego etapu pierwszego

Stanowisko testowe etapu pierwszego składa się z modułu STM32F469 Discovery połączonego z użyciem konwertera USB-UART z komputerem PC na którym uruchomiona jest aplikacja testowa z graficznym interfejsem użytkownika. Aplikacja symuluje sterownik cyfrowy przekształtnika.

Użycie na tym etapie aplikacji testowej znaczco przyspieszyło implementację oprogramowania poprzez ułatwienie procesu testowania komunikacji w systemie. Dodatkowo dzięki funkcji logowania przychodzących i wychodzących pakietów UART (Rys. A.5) aplikacja testowa umożliwiła szybkie dostrzeganie i naprawianie wszelkich nieprawidłowości w komunikacji.

Schemat połączeń



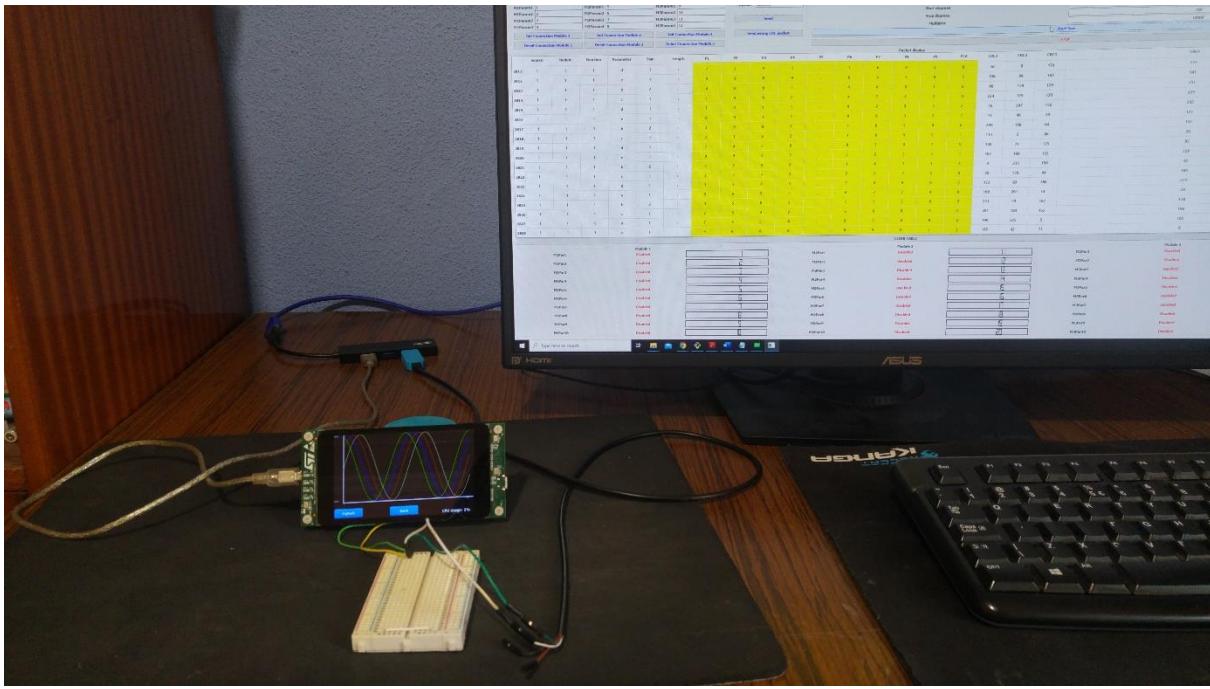
Rys. 3.3 Schemat połączeń stanowiska testowego etapu pierwszego

Komputer PC połączony jest kablem USB (oznaczonym kolorem zielonym) ze złączem CN1 płytki Discovery. Na złączu tym wyprowadzony jest interfejs ST-LINK służący między innymi do programowania płytki.

Złącze CN1 obsługuje również wirtualny port komunikacyjny – Virtual COM Port. W praktyce jest to interfejs USART3 widoczny z poziomu komputera PC jako komunikacyjny port szeregowy. Port ten używany jest do wysyłania wiadomości diagnostycznych z płytka do komputera.

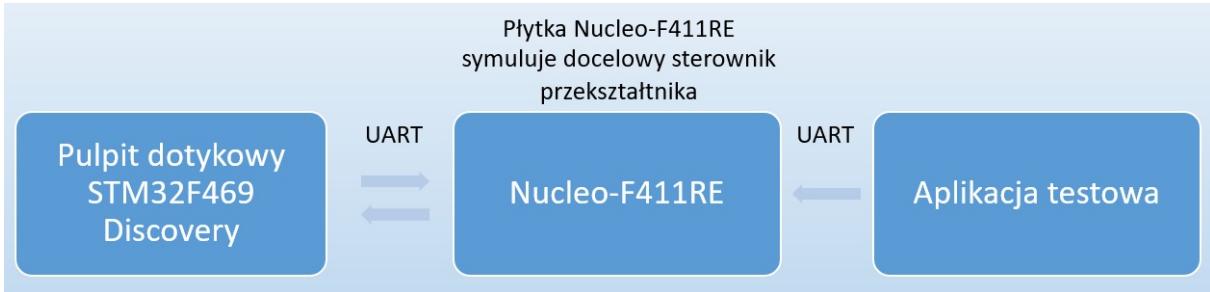
Do komputera PC podłączony jest również konwerter USB-UART rozgałęziający się na dwa kable – TX oraz RX – które służą odpowiednio do nadawania oraz odbierania sygnału. Kable te połączone są z cyfrowymi pinami żeńskimi D0 oraz D1 płytka. Piny te są skonfigurowane jako piny RX i TX interfejsu USART6.

Rys. 3.4 przedstawia stanowisko testowe etapu pierwszego. Widoczny jest ekran komputera PC z uruchomioną aplikacją testową wysyłającą dane do dotykowego pulpitu. Można zaobserwować również ekran dotykowego pulpitu, a konkretnie ekran grafu (Rys. 6.6) rysujący w czasie rzeczywistym odebrane dane.



Rys. 3.4 Stanowisko testowe etapu pierwszego

3.3 Prezentacja stanowiska testowego - etap 2



Rys. 3.5 Schemat stanowiska testowego etapu drugiego

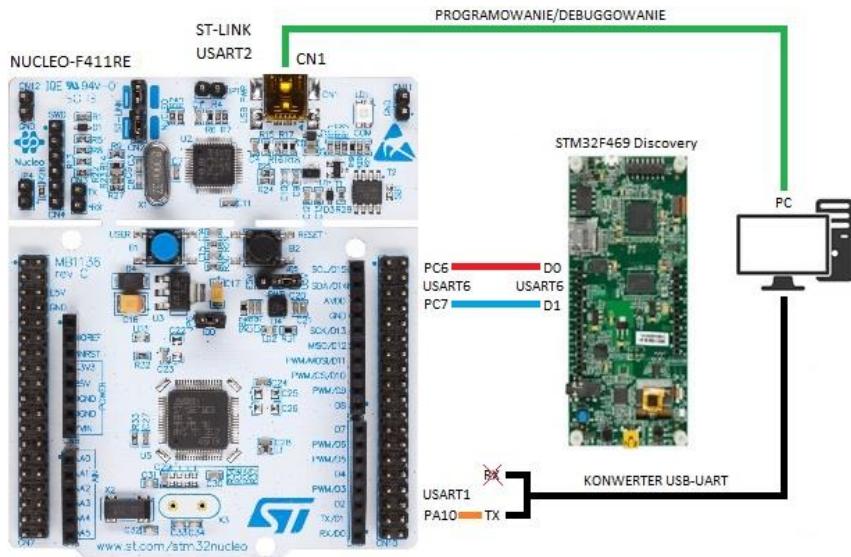
Stanowisko testowe etapu drugiego składa się z modułu Nucleo-F411RE połączonego interfejsem UART z modułem STM32F469 Discovery, a jednocześnie – poprzez konwerter USB-UART - z komputerem PC.

Złącze CN1 obsługuje również wirtualny port komunikacyjny – Virtual COM Port. W praktyce jest to interfejs USART2 widoczny z poziomu komputera PC jako komunikacyjny port szeregowy. Port ten używany jest do wysyłania wiadomości diagnostycznych z płytki do komputera.

Testowanie systemu w tej konfiguracji polega na przesyłaniu komend z komputera PC do płytki Nucleo-F411RE. Każda poprawna komenda po odpowiedniej interpretacji przez mikrokontroler uruchamia konkretną funkcjonalność co powoduje przesłanie stosownych pakietów UART do pulpitu dotykowego. Przy użyciu odpowiedniej komendy możliwy jest również odczyt danych wcześniejszych odebranych z pulpitu dotykowego. Komendy testowe wysyłane są z poziomu interfejsu graficznego zmodyfikowanej aplikacji testowej (Rys. 6.1).

Lista komend testowych, ich parametrów oraz szczegółowa procedura testowa opisane zostały w rozdziale 6.

Schemat połączeń

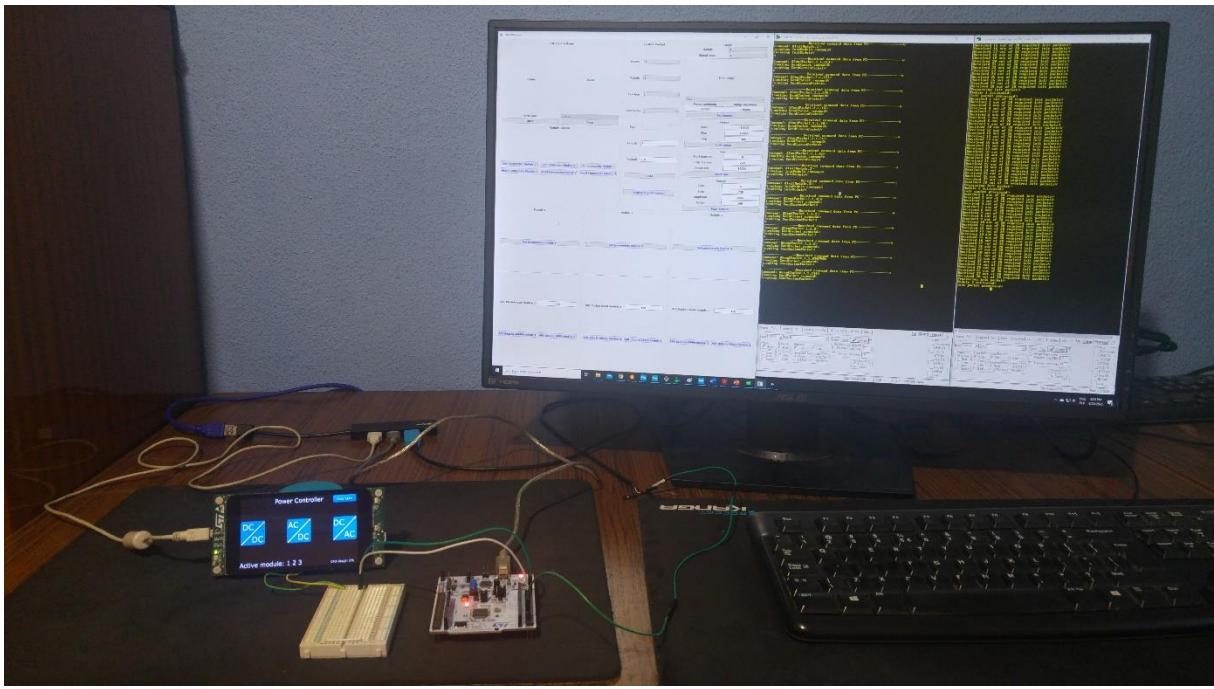


Rys. 3.6 Schemat połączeń stanowiska testowego etapu drugiego

Komputer PC połączony jest kablem USB ze złączem CN1 płytki Nucleo-F411RE. Na złączu tym wyprowadzony jest interfejs ST-LINK służący między innymi do programowania płytki.

Do komputera PC podłączony jest również konwerter USB-UART rozgałęziający się na dwa kable – TX oraz RX. Kabel TX podłączony jest do pinu PA10 modułu USART1 płytki. Tą drogą przesyłane są komendy. Kabel RX pozostaje niepodłączony.

Dodatkowo piny PC6 i PC7 modułu USART6 połączone są z pinami D0 i D1 modułu USART6 płytki STM32F469 Discovery. Tą drogą odbywa się komunikacja między mikrokontrolerami.



Rys. 3.7 Stanowisko testowe etapu drugiego

Rys. 3.7 przedstawia stanowisko testowe etapu drugiego. Można zaobserwować ekran główny dotykowego pulpitu w stanie po zainicjalizowaniu wszystkich 3 modułów. Widoczny jest również ekran komputera PC podzielony na 3 części.

Po lewej stronie widać uruchomioną zmodyfikowaną aplikację testową umożliwiającą użytkownikowi wpisanie lub wybranie z listy odpowiednich danych i wysłanie stosownej komendy do płytki Nucleo-F411RE poprzez kliknięcie przycisku reprezentującego pożądaną funkcjonalność.

Po prawej stronie ekranu widoczne są 2 okna programu RealTerm służącego do obsługi komunikacji szeregowej między urządzeniami. Okna te umożliwiają śledzenie danych odebranych z płyt, odpowiednio Nucleo-F411RE i STM32F469. Dane te informują użytkownika jakie operacje są wykonywane przez mikrokontroler każdej z płyt w danym momencie i jaka jest treść danych wysłanych lub odebranych. W szczególności zaobserwować można jakie komendy zostały odebrane przez Nucleo-F411RE i jakie pakiety UART zostały odebrane przez STM32F469.

3.4 Specyfikacja użytych urządzeń

Płytki rozwojowa STM32F469NI Discovery

Płytki rozwojowa STM32F469NI Discovery odgrywająca w pracy rolę dotykowego pulpitu to mikrokontroler produkowany przez firmę ST Microelectronics. Tabela 3.1 przedstawia kluczowe cechy mikrokontrolera wykorzystane w niniejszej pracy.

Tabela 3.1 Kluczowe elementy specyfikacji płytki STM32F469 Discovery⁵⁶

Ekran dotykowy	Wyświetlacz 4" LCD typu TFT i rozdzielczości 800x480
Mikroprocesor	Cortex M4
Maksymalna częstotliwość	180Mhz
Pamięć RAM	324kB
Wewnętrzna pamięć flash	2MB
Zewnętrzna pamięć flash	128-Mbit Quad-SPI NOR
Konektory zewnętrzne	Zestaw konektorów typu Arduino UNO V3
Diody sygnalizacyjne	4 kolorowe diody umieszczone obok ekranu dotykowego
Liczba wykorzystanych wejść/wyjść cyfrowych	2 piny typu Arduino UNO V3 (D0, D1) oraz złącze CN1 do zasilania płytki, programowania i debuggowania
Interfejsy komunikacyjne	4 konfigurowalne USART, 4 konfigurowalne UART
Napięcie zasilania	5V lub 3.3V
Przetwornik AC	12 bit

Interfejsem służącym do programowania oraz debuggowania oprogramowania płytki jest wbudowany ST-LINK w wersji V2-1.

Dodatkową istotną cechą jest obecność wirtualnego portu komunikacyjnego USB współdzielącego złącze z interfejsem ST-LINK. Wspomniany port został wykorzystany do łatwego przesyłania wiadomości diagnostycznych z płytki do komputera PC.

Płytnka rozwojowa Nucleo-F411RE

Płytnka rozwojowa Nucleo-F411RE odgrywająca w niniejszej pracy rolę sterownika przekształtnika to mikrokontroler produkowany przez firmę ST Microelectronics. Tabela 3.2 przedstawia kluczowe cechy płytki wykorzystane w niniejszej pracy.

Tabela 3.2 Kluczowe cechy specyfikacji płytki Nucleo-F411RE⁷⁸

Mikrokontroler	STM32F411RE
Maksymalna częstotliwość	100Mhz
Liczba modułów UART	3
Pamięć RAM	128kB
Pamięć flash	512kB
Konektory zewnętrzne	Zestaw konektorów typu Arduino UNO V3
Diody sygnalizacyjne	1 zielona dioda użytkownika
Liczba wykorzystanych wejść/wyjść cyfrowych	4 piny (PA9, PA10, PC6, PC7) oraz złącze CN1 do zasilania płytki, programowania i debuggowania
Interfejsy komunikacyjne	3 konfigurowalne USART
Napięcie zasilania	5V lub 3.3V
Przetwornik AC	12 bit

Interfejsem służącym do programowania oraz debuggowania oprogramowania płytki jest wbudowany ST-LINK w wersji V2-1.

⁵ https://www.st.com/resource/en/data_brief/32f469idiscovery.pdf (2.06.2020 r.).

⁶ https://www.st.com/resource/en/user_manual/dm00218846-discovery-kit-with-stm32f469ni-mcu-stmicroelectronics.pdf (2.06.2020 r.).

⁷ https://www.st.com/resource/en/data_brief/nucleo-f411re.pdf (24.06.2020 r.).

⁸ https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf (24.06.2020 r.).

Dodatkową istotną cechą jest obecność wirtualnego portu komunikacyjnego USB współdzielącego złącze z interfejsem ST-LINK. Wspomniany port został wykorzystany do łatwego przesyłania wiadomości diagnostycznych z płytki do komputera PC.

Konwerter USB-UART

Urządzenie podłączane do portu USB komputera mające na celu konwersję sygnału szeregowego do sygnału rozpoznawalnego przez UART.

3.5 Lista użytego oprogramowania

Interfejs graficzny dotykowego pulpitu został zaprojektowany, a następnie wygenerowany przy użyciu środowiska TouchGFX [9]. W ten sposób automatycznie utworzony został zestaw folderów zawierających kompletne predefiniowane projekty gotowe do uruchomienia, edycji i komplikacji w wybranych środowiskach programistycznych np. IAR Embedded Workbench lub Keil uVision. Dodatkowo, wspomniane foldery zawierały również biblioteki kodu niezbędne do obsługi systemu czasu rzeczywistego FreeRTOS oraz biblioteki HAL (Hardware Abstraction Layer) przydatne do wygodnego programowania płytka rozwojowej bez potrzeby pisania kodu odwołującego się do układu pinów i rejestrów procesora specyficznych dla danej płytki i mikrokontrolera.

IAR Embedded Workbench

Głównym narzędziem pracy nad oprogramowaniem obu płytka rozwojowych było zintegrowane środowisko programistyczne IAR Embedded Workbench 8.40 [10]. Jest ono wspierane przez środowisko TouchGFX 4.10 dzięki czemu możliwe było szybkie i łatwe komplikowanie kodu predefiniowanego projektu, programowanie pamięci mikrokontrolera oraz debuggowanie.

TouchGFX 4.10

Szkielet kodu, na podstawie którego zrealizowany został projekt wygenerowano przy użyciu środowiska TouchGFX. Jest to oprogramowanie służące do łatwego tworzenia graficznego interfejsu użytkownika wyświetlanego na ekranie dotykowym dowolnej wspieranej płytki produkowanej przez ST Microelectronics. Tworzenie interfejsu graficznego polega na prostym przeciąganiu i umieszczaniu na ekranie widgetów np. przycisków, suwaków, pól tekstowych. W ten sposób możliwe jest stworzenie atrakcyjnego interfejsu w bardzo krótkim czasie. Ponadto środowisko umożliwia szybkie testowanie stworzonego interfejsu przy użyciu wygenerowanego projektu Microsoft Visual Studio jeszcze przed wgraniem go na płytę. Dodatkowym atutem oprogramowania jest możliwość komplikacji kodu oraz wgrania programu do pamięci mikrokontrolera bezpośrednio z ekranu tworzenia interfejsu graficznego. Wygenerowany kod źródłowy oparty jest na systemie FreeRTOS.

FreeRTOS 7.6.0

Kod generowany przez środowisko TouchGFX oparty jest na darmowym systemie FreeRTOS w wersji 7.6.0 [11]. Jest to system czasu rzeczywistego umożliwiający efektywne dzielenie czasu procesora w układach jednordzeniowych między różne zdefiniowane wcześniej zadania co odbywa się poprzez zastosowanie różnych narzędzi synchronizujących zadania np. semaforów, mutexów, kolejek danych itd. W przypadku niniejszej pracy dzięki zastosowaniu systemu czasu rzeczywistego płytka może odbierać, nadawać i przetwarzać dane w międzyczasie zachowując responsywność pulpitu dotykowego.

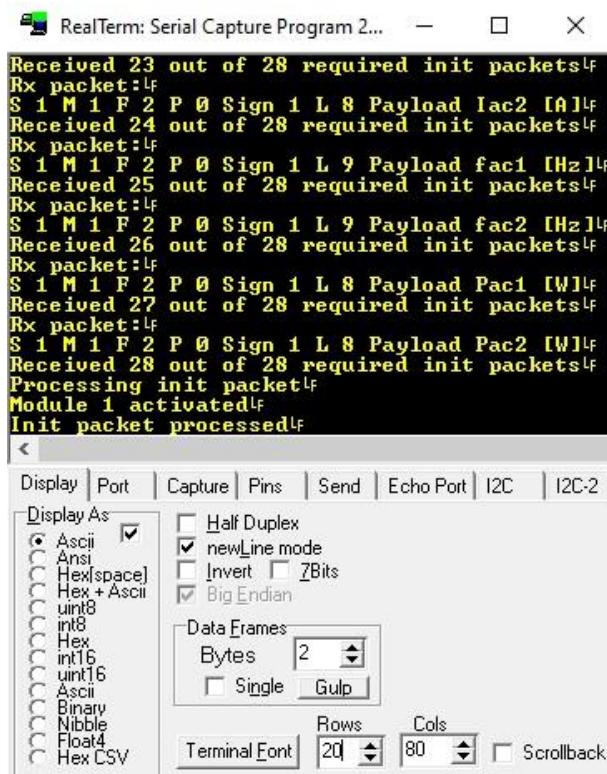
QtCreator 4.11.2 IDE + biblioteki Qt 5.13.0

Komputerowa aplikacja testowa zaimplementowana została w środowisku QtCreator [12]. Jest to oprogramowanie służące głównie do tworzenia programów zawierających niezależne od platformy graficzne interfejsy użytkownika. Dodatkowo dzięki zastosowaniu narzędzi dostępnych w obszernej bibliotece Qt możliwa jest realizacja bardziej zaawansowanych programów np. do obróbki audio lub danych pochodzących z sieci internetowych.

W przypadku niniejszej pracy biblioteki Qt posłużyły do stworzenia zaawansowanego, obsługującego komunikację szeregową przez interfejs UART, wygodnego w użyciu interfejsu graficznego, przydatnego do testowania komunikacji w systemie.

RealTerm 2.0.0.70

Do celów obserwacji komunikacji szeregowej między urządzeniami jak i również wysyłania komend testowych w etapie drugim implementacji projektu, zastosowany został program RealTerm. Zapewnia on pełną kontrolę nad portami komunikacyjnymi oraz sposobem wyświetlania i interpretacji danych. Możliwe jest przeglądanie danych odebranych zarówno w postaci kodów ASCII jak i w postaci wartości liczbowych co okazało się bardzo przydatne podczas implementacji komunikacji w niniejszym systemie.



Rys. 3.8 Dane diagnostyczne - wirtualny port komunikacyjny

Rys. 3.8 przedstawia zrzut ekranu programu RealTerm wyświetlającego dane diagnostyczne odebrane z wirtualnego portu komunikacyjnego pulpitu dotykowego. Dzięki tym danym możliwe jest bieżące monitorowanie w czasie rzeczywistym wykonywanych operacji oraz danych procesowanych przez dotykowy pulpit.

4 Opis komunikacji w systemie

Komunikacja między aplikacją testową i płytą Discovery polega na wymianie wiadomości w postaci pakietów. Każdy pakiet składa się z 20 jednobajtowych ramek, gdzie wartość każdej z ramek reprezentuje jedno pole pakietu. Każdy pakiet składa się z nagłówka, payloadu oraz sumy kontrolnej CRC.

4.1 Pakiet UART – opis ogólny

Nagłówek, payload oraz CRC składają się odpowiednio z 6, 10 i 4 pól. Opis roli jaką odgrywa każde pole w pakiecie przedstawia tabela 4.1.

Tabela 4.1 Opis pakietu UART

	Pole	Opis
Nagłówek	Source	Definiuje z/do którego urządzenia zaadresowany jest pakiet
	Module	Definiuje z/do którego modułu zaadresowany jest pakiet
	Function	Definiuje jaką jest funkcja danego pakietu
	Parameter	Definiuje parametr, którego wartość modyfikuje/przesyła dany pakiet
	Sign	Definiuje znak payloadu
	Length	Definiuje długość payloadu
	Payload	Zbiór 10 pól w których zapisywana jest przesyłana wartość
CRC	CRC	Zbiór 4 pól w których zapisywana jest suma kontrolna 32 bit CRC

4.2 Konwersja liczbowo-znakowa

Wartość każdego pola pakietu jest czytelną wprost dla człowieka wartością znakową (char) kodowaną zgodnie ze standardem ASCII np. cyfra 3 przechowywana jest jako kod liczbowy znaku '3' czyli 51. Taka decyzja projektowa umożliwiła łatwiejsze programowanie oraz diagnostykę wyświetlania i przetwarzania znaków.

Z drugiej zaś strony podczas transportu każda ramka UART przesyłana i odbierana jest jako 8-bitowa liczba całkowita bez znaku (uint8_t). Z tego względu komunikacja wymusza ciągłą konwersję z postaci liczbowej do postaci znakowej i odwrotnie.

Zamiana postaci liczbowej na znakową odbywa się poprzez dodanie do liczby kodu znaku '0' czyli 48. Zamiana odwrotna odbywa się poprzez odjęcie od kodu znaku liczby kodu znaku '0'.

$$3 + '0' \Leftrightarrow 3 + 48 \Leftrightarrow 51 \Leftrightarrow '3'$$

Tabela 4.2 Konwersja wartości liczbowych i znakowych

Wartość liczbową	Wartość znakowa	Kod liczbowy ASCII
0	'0'	48
1	'1'	49
2	'2'	50
3	'3'	51
4	'4'	52
5	'5'	53
6	'6'	54
7	'7'	55
8	'8'	56
9	'9'	57
10	:	58

4.3 Pakiet UART – zakresy wartości

Tabela 4.3 przedstawia długość oraz zakres poprawnych wartości dla każdego pola pakietu. W przypadku każdego pola wartość liczbową 0 oznacza błędą wartość co oznacza, że cały pakiet jest niepoprawny.

Tabela 4.3 Długości i zakresy wartości pól pakietu

Pole	Długość[bajt]	Zakres wartości [ASCII]
Source	1	'1'
Module	1	'1' - '3'
Function	1	'1' - '9'
Parameter	1	'0' - '9' oraz 'a' - 'e'
Sign	1	'1' - '2'
Length	1	'0' - ':'
Payload	10	'0000000000' do '9999999999'
CRC	4	4 pola liczbowe

4.4 Pakiet UART – znaczenie pól

Każde pole przenosi określona informację w pakiecie. Informacje te zostały szczegółowo opisane poniżej.

Source

Pole definiuje z którego lub do którego urządzenia adresowany jest pakiet. W obecnej implementacji występuje tylko jedno urządzenie w związku z czym pole to zawsze ma wartość '1', jednakże istnieje możliwość dodania do systemu i adresacji większej ilości urządzeń.

Module

Pole definiuje moduł, z którego lub do którego adresowany jest pakiet. Każde urządzenie może mieć wiele niezależnych modułów. W obecnej implementacji urządzenie obsługuje 3 moduły, jednakże istnieje możliwość dodania do systemu i adresacji większej ilości modułów.

Function

Pole definiuje funkcję danego pakietu. Obsługiwanych jest 9 funkcji opisanych w tabeli 4.4.

Tabela 4.4 Opis pola Function pakietu

Funkcja	Wartość znakowa	Opis
DATA_PACKET	'1'	Przesył danych w celu ich wyświetlenia na grafie
INIT_PACKET	'2'	Otwarcie połączenia z modułem
DEINIT_PACKET	'3'	Zamknięcie połączenia z modułem
ENABLE_PARAMETER_PACKET	'4'	Aktywacja parametru
DISABLE_PARAMETER_PACKET	'5'	Deaktywacja parametru
SET_PARAMETER_PACKET	'6'	Ustawienie wartości parametru
SET_GRAPH_RANGE_MIN	'7'	Ustawienie dolnego zakresu grafu
SET_GRAPH_RANGE_MAX	'8'	Ustawienie górnego zakresu grafu
SET_GRAPH_TIME_RANGE	'9'	Ustawienie zakresu czasowego grafu

Parameter

Pole definiuje parametr, którego dotyczy pakiet. Obsługiwanych jest 14 parametrów opisanych w tabeli 4.5.

Tabela 4.5 Opis pola Parameter pakietu

Nazwa parametru	Wartość znakowa	Opis
NULL_PARAMETER	'0'	Pusty parametr – parametr nie ma znaczenia
PARAMETER1	'1'	Parametr 1
PARAMETER2	'2'	Parametr 2
PARAMETER3	'3'	Parametr 3
PARAMETER4	'4'	Parametr 4
PARAMETER5	'5'	Parametr 5
PARAMETER6	'6'	Parametr 6
PARAMETER7	'7'	Parametr 7
PARAMETER8	'8'	Parametr 8
PARAMETER9	'9'	Parametr 9
PARAMETER10	'a'	Parametr 10
GRAPH_PARAMETER1	'b'	Parametr nr 1 rysowany na grafie
GRAPH_PARAMETER2	'c'	Parametr nr 2 rysowany na grafie
GRAPH_PARAMETER3	'd'	Parametr nr 3 rysowany na grafie
GRAPH_PARAMETER4	'e'	Parametr nr 4 rysowany na grafie

Sign

Pole definiuje znak payloadu pakietu. Obsługiwane są wartości podane w tabeli 4.6.

Tabela 4.6 Opis pola Sign pakietu

Znak	Wartość znakowa	Opis
POSITIVE_SIGN	'1'	Znak dodatni
NEGATIVE_SIGN	'2'	Znak ujemny

Length

Pole definiuje długość payloadu pakietu. Obsługiwany jest zakres od 1 do 10 oraz pusty payload jak opisano w tabeli 4.7.

Tabela 4.7 Opis pola Length pakietu

Długość	Wartość znakowa	Opis
NO_PAYLOAD	'0'	Pakiet nie ma payloadu
1	'1'	Długość 1
2	'2'	Długość 2
3	'3'	Długość 3
4	'4'	Długość 4
5	'5'	Długość 5
6	'6'	Długość 6
7	'7'	Długość 7
8	'8'	Długość 8
9	'9'	Długość 9
10	:	Długość 10

Payload

Zbiór 10 pól definiujących payload pakietu. Dopuszczalne wartości znajdujące się w payloadzie to znaki od '0' do '9' oraz kropka ułamkowa '.' umożliwiająca umieszczenie w payloadzie liczby zmiennoprzecinkowej. Z tego względu możliwy do uzyskania zakres wartości wynosi od 0 do 9999999999 oraz każda liczba zmiennoprzecinkowa składająca się z 10 znaków w tym kropka ułamkowa.

CRC

Zbiór 4 pól definiujących 32-bitową sumę kontrolną CRC pakietu. Każde 8-bitowe pole zawiera wartość liczbową w zakresie od 0 do 255, ale dla celów obliczeniowych wszystkie 4 pola interpretowane są łącznie jako jedna 32-bitowa liczba całkowita reprezentująca sumę kontrolną CRC pakietu.

4.5 Prędkość transmisji

Interfejsy UART w systemie zostały skonfigurowane by transmitować dane z prędkością 115200 Baudów (symboli) na sekundę. Jako że w niniejszym systemie 1 bit koduje 1 symbol, a 8 bitów to 1 bajt, ta prędkość odpowiada bezpośrednio liczbę 115200 bitów na sekundę co daje 14400 bajtów na sekundę.

Każdy pakiet komunikacyjny UART składa się z 20 jednobajtowych pól – każde przesyłane osobną ramką UART - co daje łącznie 20 bajtów. Dodatkowo przesyłanie każdej ramki UART rozpoczyna się od bitu startu oraz bitu stopu. Dodanie tych dwóch niezbędnych bitów do każdej z 20 ramek sprawia, że łącznie każdy pakiet komunikacyjny UART w systemie ma długość 200 bitów.

Biorąc pod uwagę powyższe informacje, można łatwo obliczyć, że skoro system potrafi przesłać 115200 bitów w ciągu sekundy to przesłanie 200-bitowego pakietu zajmuje około 1.736ms. Jest to wartość potrzebna do samego fizycznego przesłania pakietu i nie jest tutaj brany pod uwagę czas potrzebny na jego dalsze przetworzenie. Z tego względu realna prędkość transmisji w systemie jest mniejsza.

5 Opis funkcji systemu

W systemie wyróżnić można 9 różnych funkcji, gdzie każda z nich realizowana jest przy użyciu innego rodzaju pakietu. Każda funkcjonalność uruchamiana jest przy użyciu dedykowanego przycisku w aplikacji testowej bądź też na jednym z ekranów interfejsu graficznego płytki Discovery. Ekrany dotykowego pulpitu zostały zaprezentowane i szczegółowo opisane w podrozdziale 6.3.

Dla każdej funkcjonalności podany został przykładowy pakiet realizujący daną funkcję w postaci tabeli pól. Wartość w każdym polu jest wartością znakową np. cyfra 1 to znak '1' przesyłany w systemie jako kod ASCII równy 49. Wartość pola '-' oznacza, że pole nie zostało zainicjalizowane tzn. jego wartość jest równa 0 – ASCII 0. Pola od 17 do 20 zostały pominięte, gdyż zawierają sumę CRC.

Przesył danych

Pakiet zaklasyfikowany jako przesył danych (funkcja '1') to najprostszy w obsłudze pakiet w systemie. Służy wyłącznie do przekazywania danych zapisanych w payloadzie między urządzeniami bądź też między aplikacją testową, a płytka Discovery. Pakiet taki nie spełnia żadnych dodatkowych funkcji.

Tabela 5.1 Pakiet przesyłu danych

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	1	3	1	5	2	3	4	.	5	6	-	-	-	-

Tabela 5.1 przedstawia pakiet przesyłający do lub z modułu nr 1 wartość parametru nr 3 o długości 5 znaków wynoszącą +234.56.

Inicjalizacja połączenia

Otwarcie połączenia między aplikacją, a modułem urządzenia polega na wysłaniu przy użyciu aplikacji testowej 28 ramek inicjalizacyjnych (funkcja '2'). Każda z tych ramek zawiera w swoim payloadzie po jednej nazwie lub wartości parametru inicjującego. Parametry te wczytywane są z pliku przez aplikację testową, a następnie po wysłaniu i odebraniu przez płytke Discovery są one wyświetlane na stosownych ekranach płytki.

Bez inicjalizacji połączenia do modułu nie ma możliwości wejścia do danego modułu, gdyż przycisk jest nieaktywny. Aktywuje się on dopiero po otrzymaniu 28 ramek inicjalizacyjnych.

Każdy moduł przechowuje w pamięci własne parametry inicjalizacyjne oraz musi być aktywowany osobno.

Tabela 5.2 Pakiet inicjalizacji połączenia

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	2	0	0	9	P	a	r	a	m	e	t	r	1	-

Tabela 5.2 przedstawia jeden z pakietów inicjalizujących połączenie z modułem nr 1 przesyłający nazwę lub wartość parametru, w tym przypadku nazwę „Parametr1” o długości 9 znaków.

Deinicjalizacja połączenia

Zamknięcie połączenia między aplikacją testową, a modułem płytka polega na wysłaniu 1 ramki deinicjalizacyjnej (funkcja '3'). Po odebraniu i przetworzeniu tej ramki płytka przechodzi do ekranu głównego, a moduł i przycisk wejścia do modułu staje się nieaktywny.

Tabela 5.3 Pakiet deinicjalizacji połączenia

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	3	0	0	0	-	-	-	-	-	-	-	-	-	-

Tabela 5.3 przedstawia pakiet deinicjalizujący połączenie z modułem nr 1. Pola parametru, znaku i długości są w tym przypadku nieistotne i ustawione jako wartość znakowa '0'. Pola payloadu są niezainicjalizowane – równe 0.

Aktywacja parametru

Aktywacja parametru polega na przesłaniu 1 pakietu aktywacyjnego (funkcja '4'). Obecny stan parametru widoczny jest na ekranie aplikacji testowej.

Tabela 5.4 Pakiet aktywacji parametru

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	4	5	0	0	-	-	-	-	-	-	-	-	-	-

Tabela 5.4 przedstawia pakiet aktywujący parametr nr 5 modułu nr 1. Pola znaku i długości są w tym przypadku nieistotne i ustawione jako wartość znakowa '0'. Pola payloadu są niezainicjalizowane – równe 0.

Deaktywacja parametru

Aktywacja parametru polega na przesłaniu 1 pakietu deaktywacyjnego (funkcja '5'). Obecny stan parametru widoczny jest na ekranie aplikacji testowej.

Tabela 5.5 Pakiet deaktywacji parametru

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	5	5	0	0	-	-	-	-	-	-	-	-	-	-

Tabela 5.5 przedstawia pakiet deaktywujący parametr nr 5 modułu nr 1. Pola znaku i długości są w tym przypadku nieistotne i ustawione jako wartość znakowa '0'. Pola payloadu są niezainicjalizowane – równe 0.

Ustawianie wartości parametru

Ustawienie wartości parametru polega na przesłaniu 1 pakietu (funkcja '6'). Wartość przesyłana w payloadzie pakietu jest wartością ustawianą dla parametru. Obecna wartość parametru widoczna jest na ekranie aplikacji testowej.

Tabela 5.6 Pakiet ustawienia wartości parametru

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	6	5	2	7	1	2	3	.	4	5	6	-	-	-

Tabela 5.6 przedstawia pakiet ustawiający wartość parametru nr 5 modułu nr 1. Przesyłana wartość składa się z 7 znaków i jest ujemna – wynosi -123.456.

Ustawianie zakresu dolnego grafu

Ustawienie dolnego zakresu grafu polega na przesłaniu 1 pakietu (funkcja '7'). Wartość przesyłana w payloadzie pakietu jest wartością ustawianą dla dolnego zakresu grafu.

Tabela 5.7 Pakiet ustawienia zakresu dolnego grafu

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	7	0	2	6	1	5	0	0	0	0	-	-	-	-

Tabela 5.7 przedstawia pakiet ustawiający dolny zakres grafu. Pole parametru jest w tym przypadku nieistotne i ustawione jako wartość znakowa '0'. Przesyłana wartość składa się z 6 znaków i jest ujemna – wynosi -150000.

Ustawianie zakresu górnego grafu

Ustawienie górnego zakresu grafu polega na przesłaniu 1 pakietu (funkcja '8'). Wartość przesyłana w payloadzie pakietu jest wartością ustawianą dla górnego zakresu grafu.

Tabela 5.8 Pakiet ustawienia zakresu górnego grafu

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	8	0	1	8	2	5	0	0	0	0	0	0	-	-

Tabela 5.8 przedstawia pakiet ustawiający górny zakres grafu. Pole parametru jest w tym przypadku nieistotne i ustawione jako wartość znakowa '0'. Przesyłana wartość składa się z 8 znaków i jest dodatnia – wynosi +25000000.

Ustawianie zakresu czasowego grafu

Ustawienie zakresu czasowego grafu polega na przesłaniu 1 pakietu (funkcja '9'). Wartość przesyłana w payloadzie pakietu jest wartością ustawianą dla zakresu czasowego grafu.

Tabela 5.9 Pakiet ustawienia zakresu czasowego grafu

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
1	1	9	0	1	4	3	6	0	0	-	-	-	-	-	-

Tabela 5.9 przedstawia pakiet ustawiający zakres czasowy grafu. Pole parametru jest w tym przypadku nieistotne i ustawione jako wartość znakowa '0'. Przesyłana wartość składa się z 4 znaków – wynosi +3600.

6 Test komunikacji panel-sterownik

Po zaimplementowaniu aplikacji na dotykowym pulpicie STM32F469 Discovery i dogłębnym przetestowaniu komunikacji w systemie z użyciem aplikacji GUI przyszła pora na test praktyczny systemu. Polegał on na przetestowaniu komunikacji między dotykowym pulpitem STM32F469 Discovery i mikrokontrolerem symulującym sterownik przekształtnika.

Do tego celu w niniejszej pracy wybrana została płytka rozwojowa Nucleo-F411RE, której częścią jest relatywnie tani i mniej wydajny mikrokontroler STM32F411RE dysponujący mniejszą ilością pamięci RAM i Flash niż mikrokontroler STM32F469 co musiało zostać wzięte pod uwagę w trakcie implementacji aplikacji testowej. Kod aplikacji testowej zaimplementowano bazując na pseudokodzie i schematach blokowych każdej z funkcjonalności opisanych w dodatku D, podczas gdy opis struktury oprogramowania panelu dotykowego realizującego funkcjonalności systemu zawarty został w dodatku C.

6.1 Strategia testowa

Testowanie komunikacji między dwoma kontrolerami wymaga bezpośredniej kontroli nad każdym z nich. W przypadku płytki STM32F469 Discovery jest to bardzo proste, gdyż wszystkimi funkcjami steruje się przy pomocy interfejsu graficznego wyświetlanego na wbudowanym ekranie dotykowym. W przypadku płytki Nucleo-F411RE wymagana była implementacja dodatkowych funkcjonalności.

Sterowanie funkcjonalnościami płytki Nucleo-F411RE odbywa się poprzez przesyłanie do jej modułu USART1 komend testowych, które uruchamiają odpowiednie funkcjonalności płytki. Komendy to łańcuchy znakowe w których każdy wyraz rozpoczyna się dużą literą. Każda komenda rozpoczyna się od znaku „#” i jest uzupełniona o dodatkowe argumenty podawane po przecinkach np. „#SendPacket,1,1,500”. Ta komenda spowoduje wysłanie do parametru nr 1 modułu nr 1 wartości 500. Wartość ta zostanie następnie wyświetlona na odpowiednim ekranie (Rys. 6.3) płytki STM32F469 Discovery.

Oprogramowanie płytki Nucleo-F411RE interpretuje przesłane komendy po czym realizuje podstawową walidację liczby i rodzaju przesłanych argumentów.

Należy zaznaczyć, że płytka Nucleo-F411RE przechowuje w swojej pamięci zarówno wartości inicjalizacyjne parametrów jak i stan oraz wartości parametrów każdego modułu. Pełną listę stanów i wartości parametrów każdego z modułów można wyświetlić dzięki komendzie #GetParameters podając jako argument numer modułu np. „#GetParameters,1”. Wyświetlanie z punktu widzenia użytkownika odbywa się na zasadzie przesłania danych parametrów pożądanego modułu do komputera PC przez interfejs diagnostyczny USART2 płytki Nucleo-F411RE. Jest to interfejs współdzierający złącze z ST-LINK w związku z czym operacja nie wymaga dodatkowego okablowania.

Komunikacja między płytami STM32F469 Discovery i Nucleo-F411RE - zainicjalizowana z poziomu komputera PC odpowiednią komendą - odbywa się w identyczny sposób jak przetestowana wcześniej bezpośrednią komunikację między płytą STM32F469 Discovery, a komputerową aplikacją GUI tzn. płyta wysyłająca dla każdego przesyłanego pakietu oblicza, a następnie dołącza do pakietu sumę CRC32, a płyta odbierająca sprawdza zgodność wspomnianej sumy CRC32 zanim przystąpi do dalszego procesowania pakietu.

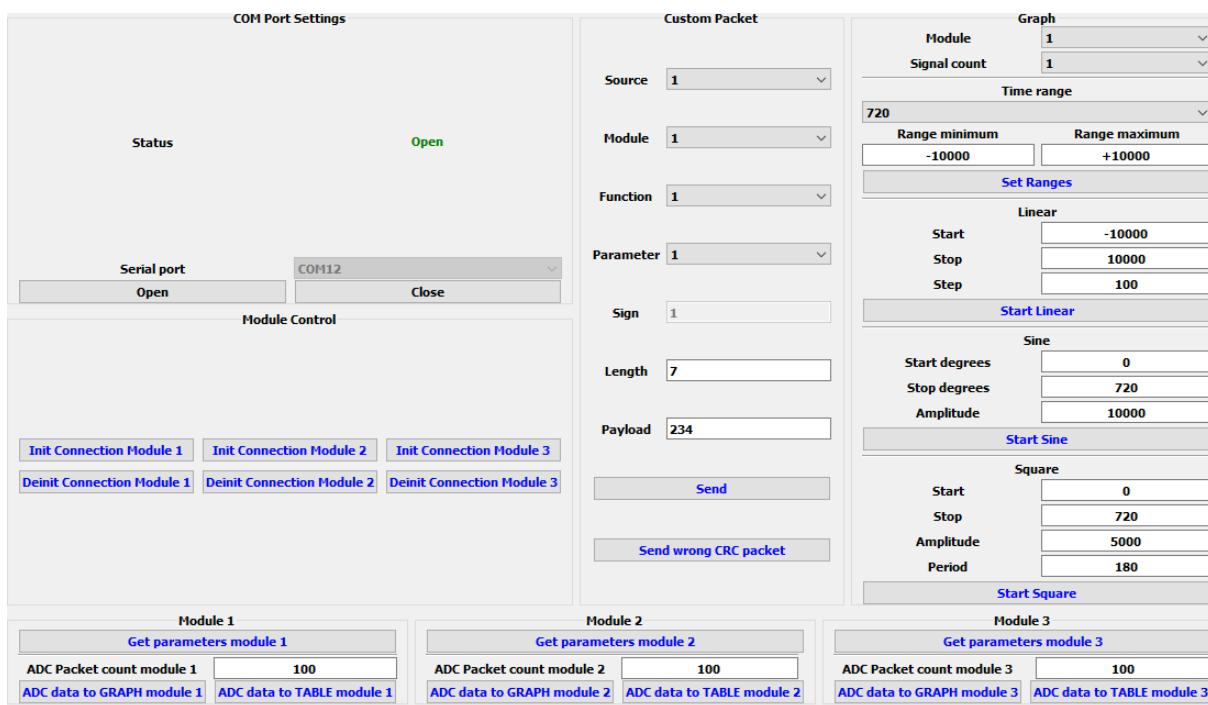
Tabela 6.1 Lista dostępnych komend testowych

Treść komendy	Liczba argumentów	Argumenty	Funkcja
#InitModule	1	Numer modułu	Inicjalizacja modułu parametrami inicjalizacyjnymi zapisanymi w pamięci płytki
#DeinitModule	1	Numer modułu	Deinicjalizacja modułu
#SetGraphMin	2	Numer modułu, Wartość liczbową (liczba całkowita)	Ustawienie wartości jako dolny zakres grafu modułu
#SetGraphMax	2	Numer modułu, Wartość liczbową (liczba całkowita)	Ustawienie wartości jako górny zakres grafu modułu
#SetGraphTime	2	Numer modułu, Wartość liczbową (liczba całkowita od 360 do 3600, podzielna przez 360)	Ustawienie wartości jako zakres czasowy grafu modułu
#GraphLinear	5	Numer modułu, liczba sygnałów, start, stop, krok	Generacja i przesyłanie danych wykresu funkcji liniowej o podanych parametrach
#GraphSine	5	Numer modułu, liczba sygnałów, start (stopnie), stop (stopnie), amplituda	Generacja i przesyłanie danych wykresu funkcji sinusoidalnej o podanych parametrach
#GraphSquare	6	Numer modułu, liczba sygnałów, start, stop, amplituda, okres	Generacja i przesyłanie danych wykresu przebiegu prostokątnego o podanych parametrach
#SendWrongCrc	1	Numer modułu	Celowe przesłanie pakietu z błędą sumą CRC w celach testowych
#GetParameters	1	Numer modułu	Wyświetlenie w konsoli stanów i wartości parametrów modułu
#Adc1	2	Numer modułu, liczba pakietów	Wczytanie z przetwornika ADC wartości napięcia oraz przesłanie jej do ekranu grafu pulpitu dotykowego
#Adc2	2	Numer modułu, liczba pakietów	Wczytanie z przetwornika ADC wartości napięcia oraz przesłanie jej do ekranu tabeli danych pulpitu dotykowego
#SendPacket	3	Numer modułu, Numer parametru, Wartość parametru (liczba rzeczywista)	Wysłanie pakietu o określonej wartości do parametru modułu

Tabela 6.1 przedstawia pełną listę obsługiwanych komend wraz z liczbą i rodzajem przyjmowanych argumentów oraz krótkim opisem funkcji każdej komendy.

6.2 Zmodyfikowana aplikacja testowa

W celu ułatwienia i przyspieszenia procedury testowania komunikacji mikrokontroler-mikrokontroler zaadaptowano odpowiednio komputerową aplikację testową używaną w pierwszym etapie projektu. Adaptacja polegała na usunięciu wszelkich nieużytecznych na tym etapie funkcjonalności aplikacji np. wczytywanie i wyświetlanie parametrów z pliku, wyświetlanie wartości parametrów oraz logowanie historii pakietów. Każdą z tych odpowiedzialności przejęła na tym etapie płytki Nucleo-F411RE. Dodatkowo dodano widgety – przyciski i pola edycji tekstuowej – odpowiedzialne za nową funkcjonalność testową tj. pobieranie i przesyłanie do dotykowego pulpitu danych z przetwornika ADC płytka Nucleo-F411RE. Rys. 6.1 przedstawia ekran aplikacji testowej (Rys. B.1) zmodyfikowanej na potrzeby testów komunikacji panel-sterownik.

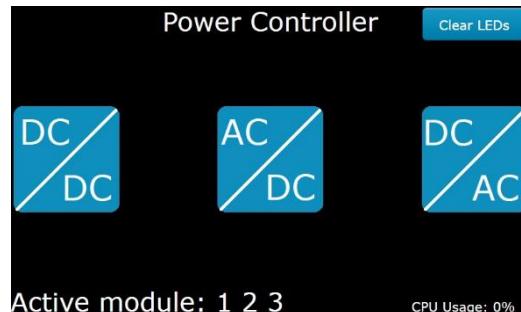


Rys. 6.1 Ekran zmodyfikowanej aplikacji testowej

6.3 Ekrany pulpitu dotykowego

Niniejszy podrozdział prezentuje ekrany dotykowego pulpitu zainicjalizowane przykładowymi wartościami wygenerowane w programie TouchGFX.

Ekran Main Menu



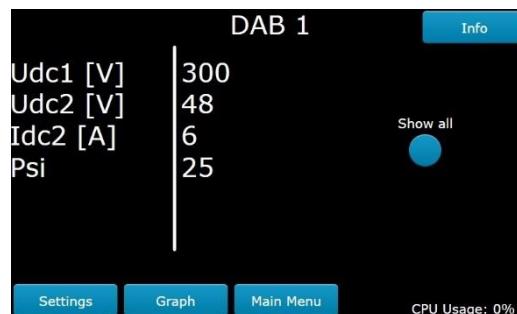
Rys. 6.2 Ekran Main Menu

Ekran główny jest pierwszym ekranem widocznym po włączeniu płytki Discovery. Umożliwia podgląd które z modułów są aktywne w danym momencie oraz przejście do poszczególnych modułów pod warunkiem, że dany moduł został wcześniej aktywowany.

Aktywne moduły wymienione są w polu tekstowym na dole ekranu, podczas gdy przyciski nieaktywnych modułów są lekko zaciemnione co symbolizuje, że nie są one dostępne w danym momencie.

Dodatkowo naciśnięcie przycisku „Clear LEDs” powoduje zgaszenie wszystkich 4 diod sygnalizacyjnych znajdujących się obok ekranu dotykowego. Zaświecone diody sygnalizują, że co najmniej jeden z pakietów odebranych przez płytę był błędny i został odrzucony – miał niepoprawną sumę CRC.

Ekran Module Data



Rys. 6.3 Ekran Module Data

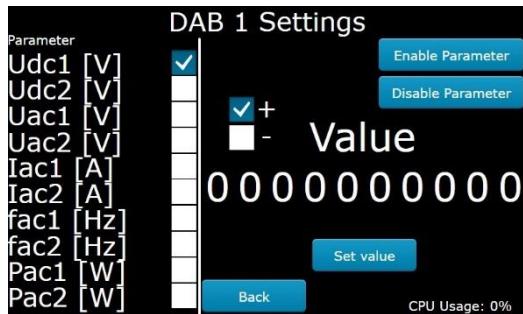
Ekran Module Data umożliwia – po lewej - podgląd nazw zainicjalizowanych parametrów oraz ich wartości – po prawej. Naciśnięcie pola tekstowego nazwy parametru powoduje ukrycie tej nazwy oraz wartości danego parametru. W ten sposób można ukryć każdy parametr. Naciśnięcie niebieskiego przycisku w kształcie kropki powoduje ponowne pokazanie wszystkich parametrów.

Po zainicjalizowaniu modułu, wartości początkowe pól tekstowych nadpisywane są odpowiednio nazwami i wartościami inicjalizacyjnymi parametrów.

W celu przesłania nowej wartości do jednego z 4 parametrów należy przesyłać pakiet typu Przesył danych (funkcja '1') o numerze parametru odpowiednio od '1' do '4' i umieścić w jego payloadzie pożądaną wartość. Najlepiej skorzystać w tym celu z grupy Custom Packet (Rys. A.3) aplikacji testowej.

Przyciski Settings, Graph, Main Menu oraz Info powodują przejście do odpowiednich ekranów.

Ecran Module Settings



Rys. 6.4 Ecran Module Settings

Jest to rozbudowany ekran umożliwiający ustawienie, aktywację oraz deaktywację parametrów w aplikacji testowej z poziomu płytka Discovery. Jest to również jedyny ekran nieodbierający danych z zewnątrz ze względu na wyłączanie przetwarzania przerwań odbioru UART podczas inicjalizacji ekranu. Działanie to jest wykonywane w celu łatwego uniknięcia konfliktów jednoczesnego przesyłu i odbioru na jednym interfejsie UART. Wspomniane przerwania są uruchamiane ponownie w przypadku przejścia do innego ekranu.

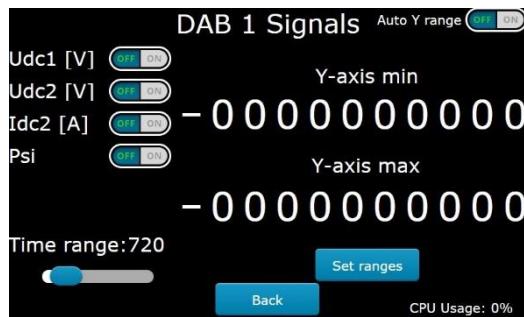
Ekran zawiera 10 pól tekstowych zainicjalizowanych odpowiednimi nazwami parametrów widocznymi również w aplikacji testowej. Każdy z parametrów ma przyporządkowany własny przycisk jednokrotnego wyboru (ang. radio button), którego zaznaczenie sprawia, że wszystkie inne dane ustawiane na tym ekranie dotyczą tego parametru. Logika przycisku jednokrotnego wyboru zapewnia, że zawsze aktywny jest jeden i tylko jeden parametr.

Aktywny parametr może być aktywowany lub deaktywowany w aplikacji testowej po naciśnięciu przycisku, odpowiednio Enable Parameter i Disable Parameter.

Po prawej stronie ekranu znajduje się zbiór 10 przewijalnych kół (ang. scroll wheel). Są to widgety umożliwiające pionowe przewijanie wartości w zakresie cyfr od 0 do 9 oraz dodatkowo kropka przecinkowa. Dzięki takiemu mechanizmowi możliwe jest łatwe i szybkie wybranie dowolnej 10-cyfrowej wartości, zarówno liczby całkowitej jak i ułamkowej. Znak wybranej w ten sposób liczby ustala się wybierając odpowiedni przycisk jednokrotnego wyboru znajdujący się nieco wyżej, domyślnie jest to znak dodatni.

Wartość wybrana przy użyciu zbioru przewijalnych kół może zostać wysłana do aplikacji testowej poprzez naciśnięcie przycisku Set value. Dotknięcie przycisku Back skutkuje przejściem do ekranu Module Data.

Ekran Signals



Rys. 6.5 Ekran Signals

Jest to rozbudowany ekran umożliwiający kontrolę nad wartościami wyświetlonymi na innej warstwie – grafie.

Obsługiwane funkcje to:

- kontrola wyświetlania sygnałów na grafie tj. dowolny sygnał może być ustawiony jako widoczny lub niewidoczny
- ustawienie zakresu czasowego grafu jako wielokrotności liczby 360 stopni
- ustawienie dolnego i górnego zakresu danych grafu
- aktywacja i deaktywacja automatycznego ustawiania zakresu osi pionowej

Wybieranie wartości dolnego i górnego zakresu grafu odbywa się podobnie jak w przypadku ekranu Settings – z użyciem przewijalnych kół, jednakże w tym przypadku nie jest dostępna kropka ułamkowa, ponieważ wartość zakresu grafu musi być liczbą całkowitą.

Wybór znaku wartości odbywa się poprzez kliknięcie pola reprezentującego znak plus/minus. Pole to nie jest polem tekstowym, a przyciskiem przełącznika (ang. toggle button) o zmodyfikowanej grafice.

Należy zaznaczyć, że powyższe obiekty służące do ręcznego ustawiania zakresu grafu widoczne są na ekranie tylko wtedy, kiedy automatyczne ustawianie zakresu Y jest wyłączone. Uaktywnienie tej opcji powoduje ukrycie powyższych elementów ekranu i włączenie w aplikacji logiki ustawiającej zakres grafu automatycznie w zależności od odbieranych wartości.

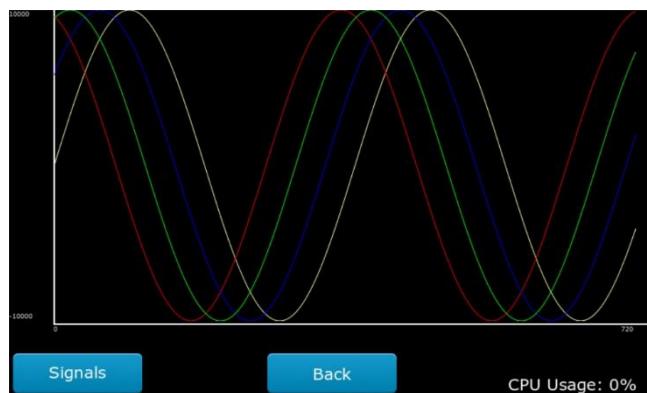
W celu wybrania zakresu czasowego należy ustawić odpowiednią wartość suwakiem znajdującym się w lewym dolnym rogu ekranu. Dostępne wartości to wielokrotności liczby 360 w zakresie od 360 do 3600.

Po wybraniu pożądanego zakresu czasowego oraz osi Y należy wcisnąć przycisk Set ranges w celu zapisania zmian. W przeciwnym razie zakresy nie zostaną zapamiętane.

Próba wybrania dolnego zakresu, który jest większy lub równy zakresowi górnemu zostanie odrzucona.

Należy tu podkreślić, że ustawienie zakresu czasowego oraz górnego i dolnego zakresu osi Y możliwe jest również zdalnie, z poziomu aplikacji testowej. Obydwie opcje są sobie funkcjonalnie równoważne.

Ekran Graph



Rys. 6.6 Ekran Graph

Ekran umożliwiający narysowanie grafu na podstawie wartości odbieranych przez płytę. Jest on w pełni sterowany z poziomu omówionego już ekranu – Signals. Możliwe jest również wspomniane wcześniej zdalne zarządzanie zakresami grafu z poziomu aplikacji testowej, również w trakcie rysowania grafu. Zdalna zmiana zakresów grafu wiąże się z jego wyczyszczeniem i rozpoczęciem rysowania od początku.

Graf rysowany jest w zakresie od 0 do wybranego zakresu czasowego – wielokrotności liczby 360 w przedziale od 360 do 3600. Wybór liczby 360 jako podstawy czasowej grafu jest nieprzypadkowy. Liczba ta jednocześnie odpowiada 360 stopniom w sensie trygonometrycznym co oznacza, że np. zakres czasowy 360 umożliwia obserwację na jednym ekranie jednego cyklu sinusa. Odpowiednio wybranie zakresu wynoszącego 3600 umożliwia obserwację 10 cykli sinusa na jednym ekranie. Wybranie większej liczby nie jest możliwe bez dalszych modyfikacji grafu ze względu na ograniczenie wielkości pamięci stosu aplikacji, w której przechowywany jest bufor zawierający wszystkie widoczne na ekranie punkty grafu.

Wartość zakresu czasowego odpowiada liczbie punktów rysowanych w danym cyklu grafu potrzebnych by graf wizualnie zajął całą dostępną mu przestrzeń od lewej do prawej. Po osiągnięciu zakresu czasowego pole grafu jest czyszczone, podstawa czasowa się zeruje, a cykl rozpoczyna się od nowa.

Ekran ten pozornie wydaje się dość prosty, lecz jest on równocześnie bardzo elastyczny w kwestii danych które może z powodzeniem wyświetlać. Ta elastyczność możliwa jest dzięki relatywnie skomplikowanej logice i przekształceniom umieszczonym w kodzie grafu, zwłaszcza w partiach kodu odpowiedzialnych za rysowanie i skalowanie grafu działającym na niskim poziomie abstrakcji, czyli operujących na prostych klasach typu malarz (PainterRGB565) budujących i rysujących graf jako zbiór punktów.

Wspomniana część kodu pobrana została z oficjalnego wymienionego w bibliografii repozytorium projektu TouchGFX. Należy zaznaczyć jednak, że oprogramowanie TouchGFX jest wciąż rozwijane. Z tego względu graf jako widget – przy całej swojej użyteczności i wygodzie użycia - nie jest jeszcze do końca dopracowany przez twórców przez co nie jest jeszcze częścią oficjalnego pakietu widgetów dostępnych w środowisku TouchGFX. Na chwilę obecną przede wszystkim nie obsługuje on liczb zmiennoprzecinkowych oraz obsługuje bezbłędnie tylko niewielki - określony empirycznie - zakres liczb całkowitych. Przypadkowe wyjście poza ten zakres wywołuje nieprzewidywalne działanie grafu. Dodatkowo biorąc pod

uwagę ogólne duże możliwości obliczeniowe płytki Discovery z mikroprocesorem z rodziną Cortex-M4 sama wydajność rysowania grafu wydaje się pozostawać wiele do życzenia.

Z wymienionych względów efektywne użycie grafu wiąże się z dodatkowymi działaniami polegającymi na skalowaniu każdej wartości przychodzącej tak by nie przekroczyła ona minimalnej oraz maksymalnej wartości obsługiwanej poprawnie przez graf, a jednocześnie by zachowana została właściwa struktura grafu. Wspomniane minimalne i maksymalne wartości to odpowiednio -2500 oraz +2500. Każda wartość, która ma być umieszczona na grafie musi zostać przeskalowana by ostatecznie zawierać się w tym zakresie.

Stosunkowo kiepska wydajność rysowania grafu spowodowana jest opóźnieniami w odbiorze danych przez interfejs UART. Jest to efekt zastosowania bibliotek HAL, które bardzo ułatwiają pracę z kodem, jednakże kosztem ich uniwersalności jest spory narzut czasowy wykonania ich kodu. Stwierdzono empirycznie, że dotykowy pulpit jest w stanie odebrać, przetworzyć i narysować na grafie zawartość pakietu w około 15 milisekund, co daje prędkość około 65 pakietów na sekundę. Przekroczenie tej bariery powoduje występowanie błędów typu UART Overrun co oznacza, że ramki UART na siebie nachodzą tzn. nowa ramka dociera do bufora nim jeszcze poprzednia została z niego odczytana i usunięta. Łatwe wykrycie i klasyfikacja tego typu błędów jest niewątpliwą zaletą biblioteki HAL.

Ecran Module Info

DAB 1 Info	
fs [kHz]	100
Uref [V]	48
Iref [A]	16
Umax [V]	60
Imax [A]	25

[Back](#) CPU Usage: 0%

Rys. 6.7 Ecran Module Info

Jest to prosty ekran wyświetlający 10 pól tekstowych. Po lewej wyświetlane są nazwy poszczególnych parametrów np. napięcie, prąd, nazwa modelu, prędkość czy wersja oprogramowania, a po prawej wyświetlana jest wartość przypisana danemu parametrowi. Wspomniane parametry pojawiają się w aplikacji podczas inicjalizacji modułu.

6.4 Przebieg testów

W niniejszym podrozdziale przedstawiono przebieg testów każdej z funkcjonalności na przykładowych danych np. moduł nr 1, parametr nr 1 itp. Należy zaznaczyć, że na potrzeby demonstracyjne obydwie płytki mają w kodzie źródłowym aktywowaną flagę trybu debuggowego tzn. wyświetlają zawartość każdego wysłanego jak i odebranego pakietu. Tego typu operacje wyświetlania przetwarzanych danych są kosztowne w sensie czasowym i znaczaco obniżają ogólną wydajność płytki zabierając czas procesora.

Do każdej funkcjonalności dołączono zrzut ekranu programu RealTerm przedstawiający informacje diagnostyczne wysiane przez każdą z płyt po wysłaniu lub odebraniu odpowiednich pakietów oraz zdjęcia ekranów dotykowego pulpitu po wykonaniu każdej z komend. Zawartość pól pakietów wiadomości diagnostycznych jest zakodowana zgodnie z tabelą 6.2.

Tabela 6.2 Znaczenie kodów w pakiecie UART

Kod	Rozwinięcie	Znaczenie
S	Source	urządzenie źródłowe/docelowe pakietu
M	Module	moduł źródłowy/docelowy pakietu
F	Function	funkcja pakietu
P	Parametr	parametr, którego dotyczy pakiet
Sign	Sign	znak payloadu pakietu
L	Length	długość payloadu pakietu
Payload	Payload	payload pakietu

Początek serii wiadomości diagnostycznych w przypadku płytki Nucleo-F411RE to 4 linie tekstu o następującym znaczeniu:

Linia 1: Informacja o tym, że odebrano komendę z komputera PC („Received command data from PC”)

Linia 2: Pełna treść komendy i odebranych argumentów,

Linia 3: Krótki zapis informujący, która komenda została rozpoznana i wywołana

Linia 4: Krótki zapis informujący, która funkcjonalność płytki została uruchomiona

W przypadku przesłania nieistniejącej w systemie komendy linia nr 3 poinformuje o tym stosownym komunikatem diagnostycznym.

Iinicjalizacja połączenia

Iinicjalizacja połączenia z modułem nr 1 wywoływana jest komendą #InitModule1. Powoduje to przesłanie 28 pakietów inicjalizacyjnych z płytki Nucleo-F411RE do STM32F469. Łańcuchy znakowe parametrów inicjalizacyjnych i ich początkowych wartości zapisane są w pamięci płytki Nucleo-F411RE.

Nucleo-F411RE

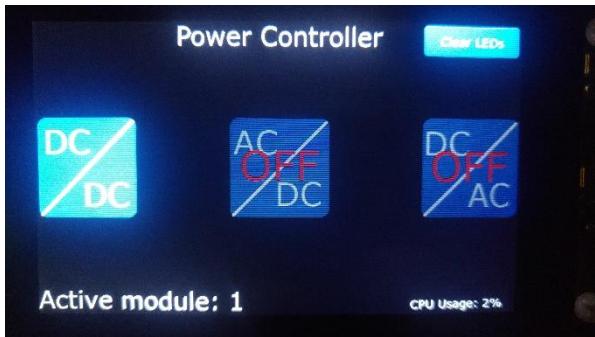
```
-----Received command data from PC-----
Command: #InitModule,1
Invoking InitModule command
Invoking InitModule
Sending init packet 1: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload fs [kHz]
Sending init packet 2: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Uref [V]
Sending init packet 3: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Iref [A]
Sending init packet 4: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Umax [V]
Sending init packet 5: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Imax [A]
Sending init packet 6: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 3 Payload 100
Sending init packet 7: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 48
Sending init packet 8: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 16
Sending init packet 9: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 60
Sending init packet 10: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 25
Sending init packet 11: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Udc1 [V]
Sending init packet 12: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Udc2 [V]
Sending init packet 13: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Idc2 [A]
Sending init packet 14: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 3 Payload Psi
Sending init packet 15: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 3 Payload 300
Sending init packet 16: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 48
Sending init packet 17: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 1 Payload 6
Sending init packet 18: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 25
Sending init packet 19: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Udc1 [V]
Sending init packet 20: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Udc2 [V]
Sending init packet 21: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Vac1 [V]
Sending init packet 22: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Vac2 [V]
Sending init packet 23: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Iac1 [A]
Sending init packet 24: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Iac2 [A]
Sending init packet 25: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 9 Payload fac1 [Hz]
Sending init packet 26: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 9 Payload fac2 [Hz]
Sending init packet 27: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Pac1 [W]
Sending init packet 28: Init Packet is: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Pac2 [W]
```

Rys. 6.8 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#InitModule,1"

STM32F469 Discovery

```
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload fs [kHz]
Received 1 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Uref [V]
Received 2 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Iref [A]
Received 3 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Umax [V]
Received 4 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Imax [A]
Received 5 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 3 Payload 100
Received 6 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 48
Received 7 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 16
Received 8 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 60
Received 9 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 25
Received 10 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Udc1 [V]
Received 11 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Udc2 [V]
Received 12 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Idc2 [A]
Received 13 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 3 Payload Psi
Received 14 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 3 Payload 300
Received 15 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 48
Received 16 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 1 Payload 6
Received 17 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 2 Payload 25
Received 18 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Udc1 [V]
Received 19 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Udc2 [V]
Received 20 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Vac1 [V]
Received 21 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Vac2 [V]
Received 22 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Iac1 [A]
Received 23 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Iac2 [A]
Received 24 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 9 Payload fac1 [Hz]
Received 25 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 9 Payload fac2 [Hz]
Received 26 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Pac1 [W]
Received 27 out of 28 required init packets
Received packet: S 1 M 1 F 2 P 0 Sign 1 L 8 Payload Pac2 [W]
Received 28 out of 28 required init packets
Processing init packet
Module 1 activated
Init packet processed
```

Rys. 6.9 Wiadomości diagnostyczne STM32F469 - Komenda "#InitModule,1"



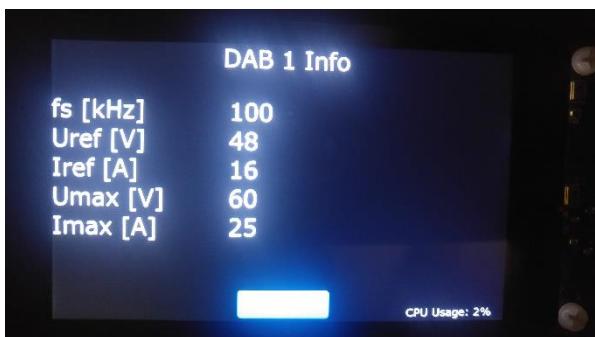
Rys. 6.10 Ekran główny po inicializacji modułu 1 komendą "#InitModule,1"



Rys. 6.11 Ekran Data po inicializacji komendą "#InitModule,1"



Rys. 6.12 Ekran Settings po inicializacji komendą "#InitModule,1"



Rys. 6.13 Ekran Info po inicializacji komendą "#InitModule,1"

Deinicjalizacja połączenia

Deinicjalizacja połączenia z modułem nr 1 wywoływana jest komendą #DeinitModule1. Powoduje to przesłanie pakietu deinicjalizacyjnego z płytki Nucleo-F411RE do STM32F469.

Nucleo-F411RE

```
-----Received command data from PC-----
Command: #DeinitModule,1
Invoking DeinitModule command
Invoking DeinitModule
Deinit Packet is: S 1 M 1 F 3 P 0 Sign 1 L 0 Payload
```

Rys. 6.14 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#DeinitModule,1"

STM32F469 Discovery

```
Received packet: S 1 M 1 F 3 P 0 Sign 1 L 0 Payload
Deinit packet received
Module 1 deactivated
```

Rys. 6.15 Wiadomości diagnostyczne STM32F469 - Komenda "#DeinitModule,1"



Rys. 6.16 Ekran główny po deinicjalizacji modułu 1 komendą "#DeinitModule,1"

Ustawienie zakresów grafu

Wartość minimalna: 0

Wartość maksymalna: 100

Zakres czasowy: 360

Ustawienie powyższych zakresów grafu modułu nr 1 wiąże się ze zbiorczym wysłaniem 3 komend, odpowiednio: „SetGraphMin,1,0”, po czym „SetGraphMax,1,100”, a następnie „SetGraphTime,1,360”

Nucleo-F411RE

```
-----Received command data from PC-----
Command: #SetGraphMin,1,0
Invoking SetGraphMin command
Invoking SetGraphLowerRange
SetGraphLowerRange Packet is: S 1 M 1 F 7 P 0 Sign 1 L 1 Payload 0

-----Received command data from PC-----
Command: #SetGraphMax,1,100
Invoking SetGraphMax command
Invoking SetGraphUpperRange
SetGraphUpperRange Packet is: S 1 M 1 F 8 P 0 Sign 1 L 3 Payload 100

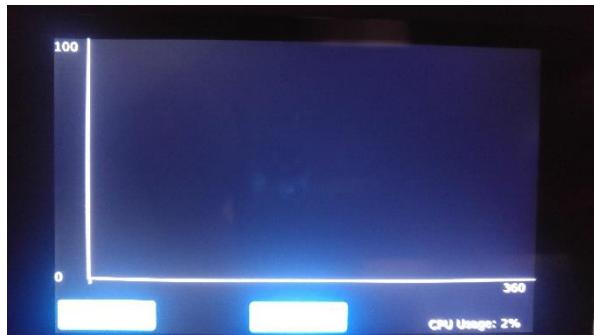
-----Received command data from PC-----
Command: #SetGraphTime,1,360
Invoking SetGraphTime command
Invoking SetGraphTimeRange
SetGraphTimeRange Packet is: S 1 M 1 F 9 P 0 Sign 1 L 3 Payload 360
```

Rys. 6.17 Wiadomości diagnostyczne Nucleo-F411RE - Komendy zakresów czasowych

STM32F469 Discovery

```
Received packet: S 1 M 1 F 7 P 0 Sign 1 L 1 Payload 0
Set graph range minimum packet received
Received packet: S 1 M 1 F 8 P 0 Sign 1 L 3 Payload 100
Set graph range maximum packet received
Received packet: S 1 M 1 F 9 P 0 Sign 1 L 3 Payload 360
Set graph time range packet received
```

Rys. 6.18 Wiadomości diagnostyczne STM32F469 - Komendy zakresów czasowych



Rys. 6.19 Ekran grafu po zmianie zakresów na 0, 100, 360

Generacja przebiegu liniowego

Start: -5

Stop: +5

Krok: 2

W celu wygenerowania 4 przebiegów liniowych o powyższych parametrach na grafie modułu nr 1 należy przesłać komendę „#GraphLinear,1,4,-5,5,2”.

Nucleo-F411RE

```
-----Received command data from PC-----
Command: #GraphLinear,1,4,-5,5,2
Invoking GraphLinear command
Invoking GenerateLinearGraph
```

Rys. 6.20 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#GraphLinear,1,4,-5,5,2"

STM32F469 Discovery

```
Received packet: S 1 M 1 F 1 P b Sign 2 L 8 Payload 5.000000
Received packet: S 1 M 1 F 1 P c Sign 2 L 8 Payload 3.750000
Received packet: S 1 M 1 F 1 P d Sign 2 L 8 Payload 2.500000
Received packet: S 1 M 1 F 1 P e Sign 2 L 8 Payload 1.250000
Received packet: S 1 M 1 F 1 P b Sign 2 L 8 Payload 3.000000
Received packet: S 1 M 1 F 1 P c Sign 2 L 8 Payload 2.250000
Received packet: S 1 M 1 F 1 P d Sign 2 L 8 Payload 1.500000
Received packet: S 1 M 1 F 1 P e Sign 2 L 8 Payload 0.750000
Received packet: S 1 M 1 F 1 P b Sign 2 L 8 Payload 1.000000
Received packet: S 1 M 1 F 1 P c Sign 2 L 8 Payload 0.750000
Received packet: S 1 M 1 F 1 P d Sign 2 L 8 Payload 0.500000
Received packet: S 1 M 1 F 1 P e Sign 2 L 8 Payload 0.250000
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 1.000000
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 0.750000
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.500000
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 0.250000
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 3.000000
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 2.250000
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 1.500000
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 0.750000
```

Rys. 6.21 Wiadomości diagnostyczne STM32F469 - Komenda "#GraphLinear,1,4,-5,5,2"



Rys. 6.22 Przykładowe przebiegi liniowe narysowane na grafie dotykowego pulpitu – Komenda „#GraphLinear,1,4-10,10,0.1”

Generacja przebiegu sinusoidalnego

Start (stopnie): 0

Stop (stopnie): +5

Amplituda: 10

W celu wygenerowania 4 przebiegów sinusoidalnych o powyższych parametrach na grafie modułu nr 1 należy przesłać komendę „#GraphSine,1,4,0,5,10”. Sinusoidy są przesunięte względem siebie o 120 stopni.

Nucleo-F411RE

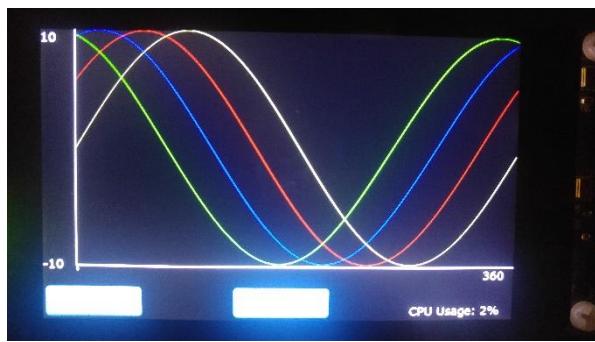
```
----- Received command data from PC -----
Command: #GraphSine,1,4,0,5,10
Invoking GraphSine command
Invoking GenerateSineGraph
```

Rys. 6.23 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#GraphSine,1,4,0,5,10"

STM32F469 Discovery

```
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 0.000000
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 5.806112
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 9.454452
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 9.589157
Received packet: S 1 M 1 F 1 P h Sign 1 L 8 Payload 0.174524
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 5.947322
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 9.509868
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 9.538186
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 0.348995
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 6.086720
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 9.562388
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 9.484309
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 0.523359
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 6.224264
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 9.611995
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 9.427543
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 0.697564
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 6.359912
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 9.658674
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 9.367906
```

Rys. 6.24 Wiadomości diagnostyczne STM32F469 - Komenda "#GraphSine,1,4,0,5,10"



Rys. 6.25 Przykładowe przebiegi sinusoidalne narysowane na grafie dotykowego pulpitu - Komenda "#GraphSine,1,4,0,360,10"

Generacja przebiegu prostokątnego

Start: 0

Stop: 10

Amplituda: 5

Okres: 90

W celu wygenerowania 2 przebiegów prostokątnych o powyższych parametrach na grafie modułu nr 1 należy przesyłać komendę „#GraphSquare,1,4,0,10,5,90”. Sygnały prostokątne są przesunięte względem siebie o pół okresu.

Nucleo-F411RE

```
-----Received command data from PC-----
Command: #GraphSquare,1,4,0,10,5,90
Invoking GraphSquare command
Invoking GenerateSquareGraph
```

Rys. 6.26 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#GraphSquare,1,4,0,10,5,90"

STM32F469 Discovery

Rys. 6.27 Wiadomości diagnostyczne STM32F469 - Komenda "#GraphSquare, 1,4,0,10,5,90"



Rys. 6.28 Przykładowe przebiegi prostokątne narysowane na grafie dotykowego pulpitu – Komenda "#GraphSquare, 1,4,0,360,5,180"

Przesłanie pakietu o błędny CRC

Przesłanie pakietu o błędny CRC do modułu nr 1 wywołuje się komendą #SendWrongCrc,1".

Nucleo-F411RE

```
-----Received command data from PC-----  
Command: #SendWrongCrc,1  
Invoking SendWrongCrc command  
Invoking SendWrongCrc  
SendWrongCrc packet is:S 1 M 1 F 1 P 0 Sign L Payload
```

Rys. 6.29 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#SendWrongCrc, 1"

STM32F469 Discovery

WRONG_CRC S 1 M 1 F 1 P 0 Sign L Payload CRC1 0 CRC2 0 CRC3 0 CRC4 0

Rys. 6.30 Wiadomości diagnostyczne STM32F469 - Komenda "#SendWrongCrc, 1"



Rys. 6.31 Zapalona czerwona dioda po odebraniu pakietu o błędnej sumie CRC

Pobranie wartości i stanów parametrów

W celu pobrania wartości i stanów parametrów modułu nr 1 z pamięci płytki Nucleo-F411RE należy przesyłać komendę „#GetParameters,1”. W odpowiedzi płytki przesyła w postaci wiadomości diagnostycznej wartości liczbowe i stany wszystkich 10 parametrów. W przypadku stanów parametrów wartość 1 oznacza, że parametr jest aktywny, natomiast wartość 0 oznacza, że parametr jest nieaktywny.

Nucleo-F411RE

```
--Received command data from PC-----
Command: #GetParameters,1
Invoking GetParameters command
Invoking GetParameters

Module 1 parameters:

Parameter 1 state: 0
Parameter 2 state: 1
Parameter 3 state: 1
Parameter 4 state: 0
Parameter 5 state: 1
Parameter 6 state: 0
Parameter 7 state: 1
Parameter 8 state: 1
Parameter 9 state: 0
Parameter 10 state: 1

Parameter 1 value: -36392.719000
Parameter 2 value: 8092.719000
Parameter 3 value: -592.719000
Parameter 4 value: 2.908719
Parameter 5 value: -1692.719000
Parameter 6 value: 12345.678900
Parameter 7 value: 0.719000
Parameter 8 value: -18092.719700
Parameter 9 value: 3.929719
Parameter 10 value: 409087.900000
```

Rys. 6.32 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#GetParameters,1"

Przesłanie wartości ADC do grafu dotykowego pulpitu

W celu przesyłania do grafu modułu nr 1, 20 pakietów reprezentujących dane z 4 kanałów konwertera ADC płytki Nucleo-F411RE (po 5 pakietów na każdy sygnał) należy użyć komendy „#Adc1,1,5”. Sygnały ADC przedstawione na Rys. 6.35 są odpowiednio przeskalowane tzn. do każdego z nich dodano wartość odpowiednio 0, +3, -3 i +6, by uniknąć efektu nakładania się sygnałów.

Nucleo-F411RE

```
-----Received command data from PC-----
Command: #Adc1,1,5
Invoking Adc1 command
Invoking SendAdcToGraphScreen

Module 1, generating and sending 5 ADC packets per each of 4 signals

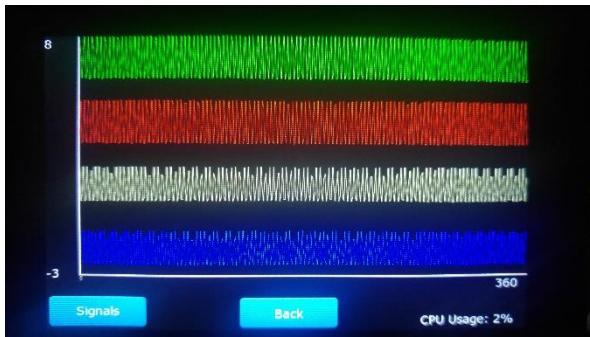
ADC raw value is 1869 out of 4096 = 1.506U
Sending packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 1.505786
ADC raw value is 1244 out of 4096 = 1.002U
Sending packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 1.002246
ADC raw value is 426 out of 4096 = 0.343U
Sending packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.343213
ADC raw value is 2530 out of 4096 = 2.038U
Sending packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 2.038330
ADC raw value is 437 out of 4096 = 0.352U
Sending packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 0.352075
ADC raw value is 2509 out of 4096 = 2.021U
Sending packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 2.021411
ADC raw value is 473 out of 4096 = 0.381U
Sending packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.381079
ADC raw value is 524 out of 4096 = 0.422U
Sending packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 0.422168
ADC raw value is 2526 out of 4096 = 2.035U
Sending packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 2.035107
ADC raw value is 68 out of 4096 = 0.055U
Sending packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 0.054785
ADC raw value is 2720 out of 4096 = 2.191U
Sending packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 2.191406
ADC raw value is 93 out of 4096 = 0.075U
Sending packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 0.074927
ADC raw value is 2664 out of 4096 = 2.146U
Sending packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 2.146289
ADC raw value is 572 out of 4096 = 0.461U
Sending packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 0.460840
ADC raw value is 424 out of 4096 = 0.342U
Sending packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.341602
ADC raw value is 2538 out of 4096 = 2.045U
Sending packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 2.044775
ADC raw value is 428 out of 4096 = 0.345U
Sending packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 0.344824
ADC raw value is 2540 out of 4096 = 2.046U
Sending packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 2.046387
ADC raw value is 450 out of 4096 = 0.363U
Sending packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.362549
ADC raw value is 2235 out of 4096 = 1.801U
Sending packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 1.800659
```

Rys. 6.33 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#Adc1,1,5"

STM32F469 Discovery

```
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 1.505786
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 1.002246
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.343213
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 2.038330
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 0.352075
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 2.021411
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.381079
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 0.422168
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 2.035107
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 0.054785
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 2.191406
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 0.074927
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 2.146289
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 0.460840
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.341602
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 2.044775
Received packet: S 1 M 1 F 1 P b Sign 1 L 8 Payload 0.344824
Received packet: S 1 M 1 F 1 P c Sign 1 L 8 Payload 2.046387
Received packet: S 1 M 1 F 1 P d Sign 1 L 8 Payload 0.362549
Received packet: S 1 M 1 F 1 P e Sign 1 L 8 Payload 1.800659
```

Rys. 6.34 Wiadomości diagnostyczne STM32F469 - Komenda "#Adc1,1,5"



Rys. 6.35 4 sygnały reprezentujące szum zbierany przez 4 kanały przetwornika ADC narysowane na grafie dotykowego pulpitu

Przesłanie wartości ADC do widoku tabeli dotykowego pulpitu

W celu przesłania do widoku tabeli modułu nr 1, 20 pakietów reprezentujących dane z 4 kanałów konwertera ADC płytki Nucleo-F411RE (po 5 pakietów na każdy sygnał) należy użyć komendy „#Adc2,1,5”.

Nucleo-F411RE

```
-----Received command data from PC-----
Command: #Adc2,1,5
Invoking Adc2 command
Invoking SendAdcToDataScreen

Module 1, generating and sending 5 ADC packets per each of 4 signals

ADC raw value is 429 out of 4096 = 0.346U
Sending packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 0.345630
ADC raw value is 130 out of 4096 = 0.105U
Sending packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 0.104736
ADC raw value is 1578 out of 4096 = 1.221U
Sending packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 1.271338
ADC raw value is 1223 out of 4096 = 0.985U
Sending packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 0.985327
ADC raw value is 427 out of 4096 = 0.344U
Sending packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 0.344019
ADC raw value is 2532 out of 4096 = 2.048U
Sending packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 2.039941
ADC raw value is 444 out of 4096 = 0.358U
Sending packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 0.357715
ADC raw value is 2486 out of 4096 = 2.003U
Sending packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 2.002881
ADC raw value is 500 out of 4096 = 0.403U
Sending packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 0.402832
ADC raw value is 901 out of 4096 = 0.726U
Sending packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 0.725903
ADC raw value is 2240 out of 4096 = 1.805U
Sending packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 1.804688
ADC raw value is 81 out of 4096 = 0.065U
Sending packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 0.065259
ADC raw value is 2514 out of 4096 = 2.025U
Sending packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 2.025440
ADC raw value is 114 out of 4096 = 0.092U
Sending packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 0.091846
ADC raw value is 2477 out of 4096 = 1.996U
Sending packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 1.995630
ADC raw value is 434 out of 4096 = 0.406U
Sending packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 0.406055
ADC raw value is 425 out of 4096 = 0.342U
Sending packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 0.342407
ADC raw value is 2497 out of 4096 = 2.012U
Sending packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 2.011743
ADC raw value is 438 out of 4096 = 0.353U
Sending packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 0.352881
ADC raw value is 2524 out of 4096 = 2.033U
Sending packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 2.033496
```

Rys. 6.36 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#Adc2,1,5"

STM32F469 Discovery

```
Received packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 0.345630
Received packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 0.104736
Received packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 1.271338
Received packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 0.985327
Received packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 0.344019
Received packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 2.039491
Received packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 0.357715
Received packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 2.002881
Received packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 0.402832
Received packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 0.725903
Received packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 1.804688
Received packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 0.065259
Received packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 2.025440
Received packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 0.091846
Received packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 1.995630
Received packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 0.406055
Received packet: S 1 M 1 F 1 P 1 Sign 1 L 8 Payload 0.342407
Received packet: S 1 M 1 F 1 P 2 Sign 1 L 8 Payload 2.011743
Received packet: S 1 M 1 F 1 P 3 Sign 1 L 8 Payload 0.352881
Received packet: S 1 M 1 F 1 P 4 Sign 1 L 8 Payload 2.033496
```

Rys. 6.37 Wiadomości diagnostyczne STM32F469 - Komenda "#Adc2,1,5"



Rys. 6.38 Ekran Data wyświetlający dane z przetwornika ADC

Przesłanie pakietu o dowolnej wartości

W celu przesłania do parametru nr 1 modułu nr 1 pakietu o wartości payloadu 123456.789 należy przesyłać komendę „#SendPacket,1,1,123456.789”.

Nucleo-F411RE

```
-----Received command data from PC-----
Command: #SendPacket,1,1,123456.789
Invoking SendPacket command
Invoking SendCustomPacket
SendCustomPacket packet is:S 1 M 1 F 1 P 1 Sign 1 L : Payload 123456.789
```

Rys. 6.39 Wiadomości diagnostyczne Nucleo-F411RE - Komenda "#SendPacket,1,1,123456.789"

STM32F469 Discovery

```
Received packet: S 1 M 1 F 1 P 1 Sign 1 L : Payload 123456.789
```

Rys. 6.40 Wiadomości diagnostyczne STM32F469- Komenda "#SendPacket,1,1,123456.789"



Rys. 6.41 Ekran Data po przesłaniu komendy "#SendPacket,1,1,123456.789"

7 Wnioski

Ekran dotykowy mikrokontrolera STM32F469NI-Discovery zgodnie z oczekiwaniami okazał się bardzo dobrym interfejsem kontrolnym zdolnym zarówno do efektywnego wyświetlania danych odbieranych jak i również łatwego ustawiania wartości wysyłanych następnie do z innego mikrokontrolera – w tym przypadku Nucleo-F411RE.

Mimo wielu trudności udało się zrealizować bardzo elastyczną funkcjonalność grafu umożliwiającą równoległe rysowanie do 4 wykresów w zadowalającym tempie oraz w zakresie wybieranym przez użytkownika z poziomu samej płytki Discovery lub zewnętrznie z poziomu płytki Nucleo-F411RE. Wartości zakresu grafu ograniczone są jedynie przez zakres możliwych wartości minimalnych i maksymalnych, które można przesyłać w pakiecie.

Wspomniana funkcjonalność stała się dostępna po zaadaptowaniu kodu udostępnionego przez twórców TouchGFX gdzie punktem wyjściowym był graf, który potrafił rysować jedynie dodatnie liczby całkowite w przedziale od 0 do 1000.

Istotną funkcjonalnością do zrealizowania było łatwe ustawianie z poziomu ekranu dotykowego wartości liczbowych o długości do 10 znaków włącznie na dość ograniczonej powierzchni ekranu. Zastosowano zbiór widgetów - kół przewijalnych (ang. scroll wheel) – dzięki czemu nie było potrzeby implementacji typowej klawiatury dotykowej.

Można mieć zastrzeżenia do szybkości rysowania grafu, jednakże po bliższym przyjrzeniu się można stwierdzić, że jest to wina zbyt wolnego odbierania danych z interfejsu UART. Niestety biblioteka HAL nie jest idealna i jej uniwersalność oraz wygoda użycia okupiona jest sporem narutem czasowym wykonania jej kodu. Sam mikroprocesor Cortex-M4 jest bardzo wydajną jednostką zapewniającą płynne i responsywne działanie graficznego interfejsu graficznego mimo taktowania 180Mhz, które de facto jest o rząd wielkości niższe niż częstotliwości mikroprocesorów współczesnych telefonów komórkowych sięgających i przekraczających wielkości 2GHz.

Framework TouchGFX okazał się być bardzo wygodnym, darmowym narzędziem do tworzenia graficznego interfejsu użytkownika dla płytki Discovery. Stworzenie podstawowego interfejsu użytkownika z użyciem tego oprogramowania sprowadza się do stworzenia projektu, intuicyjnego umieszczenia na ekranie kilku widgetów i wciśnięcia przycisku, który automatycznie buduje i flashuje projekt na płytce Discovery.

UART jest stosunkowo prostym i mocno ograniczonym funkcjonalnie i wydajnościowo interfejsem komunikacyjnym w porównaniu choćby do komunikacji realizowanej przy użyciu protokołu SPI czy I2C, ale jak się okazuje po odpowiednim skonfigurowaniu i zabezpieczeniu przed błędami może być z powodzeniem użyty nawet w bardziej skomplikowanych projektach. W przypadku niniejszej pracy krytyczna jest prędkość przesyłu danych, w związku z czym protokół SPI byłby trudniejszym w użyciu, ale jednak lepszym rozwiązaniem niż UART, gdyż biorąc pod uwagę wymienione wcześniej protokoły SPI jest z nich najszybszym. Nie ma w tym przypadku znaczenia brak możliwości sprawdzania błędów transferu danych w protokole SPI (np. dodatkowy bit parzystości obecny w UART) gdyż sprawdzanie poprawności przesłanych pakietów w niniejszym systemie odbywa się na zasadzie obliczania i weryfikacji sumy kontrolnej CRC.

System zaimplementowany i testowany początkowo z użyciem wyłącznie płytki STM32F469 Discovery i aplikacji testowej PC działa również poprawnie - bez wprowadzania żadnych modyfikacji po stronie płytki – po zastąpieniu aplikacji testowej PC niskobudżetową płytą Nucleo-F411RE. Fakt ten dowodzi, że aplikacja STM32F469 Discovery jest w stanie komunikować się poprawnie z dowolnym mikrokontrolerem wyposażonym w moduł UART. Co więcej, system zaimplementowano z myślą o sterowaniu i monitorowaniu przekształników energoelektronicznych, jednakże okazuje się on być tak uniwersalny, że może z powodzeniem zostać wykorzystany również do sterowania innymi systemami np. oświetlenia, ogrzewania.

Niniejszy praca opisuje komunikację dotyковego pulpitu z jednym sterownikiem przekształtnika, jednakże projekt może zostać rozwinięty o możliwość obsługi kilku sterowników przekształników, gdyż pierwsze pole pakietu komunikacyjnego UART o nazwie „Source” identyfikuje urządzenie nadawcze lub docelowe dla każdego z pakietów co umożliwia adresację wielu urządzeń. Wymagałoby to jednak zmiany protokołu komunikacyjnego z UART na I2C lub SPI, gdyż w przeciwieństwie do UART wspierają one adresację urządzeń.

Kolejnym ulepszeniem niniejszego systemu może być zwiększenie prędkości procesowania danych - a co za tym idzie prędkości rysowania sygnałów na grafie – poprzez optymalizację kodu aplikacji dotyковego pulpitu. Można tego dokonać zmieniając sposób obsługi interfejsu UART, a dokładniej rezygnując z użycia biblioteki HAL na rzecz pracy bezpośrednio na rejestrach mikroprocesora znajdującego się na płytce STM32F469 Discovery. Prawdopodobnie przypieszy to znaczaco przetwarzanie danych w systemie, jednakże należy liczyć się z faktem, że po takiej modyfikacji w przypadku wystąpienia błędów UART w systemie nie będą one już tak łatwo zauważalne, gdyż większość operacji wykonywanych przez HAL w kwestii UART to właśnie sprawdzanie wszelkich flag błędów komunikacji i raportowanie o nich.

Bibliografia

- [1] M. P. Kaźmierkowski, J. T. Matysik, *Wprowadzenie do elektroniki i energoelektroniki*, Warszawa 2005, s. 277.
- [2] M. Nowak, R. Barlik, *Poradnik inżyniera energoelektronika* s. 26.
- [3] M. P. Kaźmierkowski, J. T. Matysik, *Wprowadzenie do elektroniki i energoelektroniki*, Warszawa 2005, s. 279.
- [4] K. Nakajima, *Power converter control apparatus*,
https://www.researchgate.net/publication/302740541_Power_converter_control_apparatus (dostęp 25.06.2020)
- [5] P. Markowski, *Multiphase power converters, controllers and control methods*,
https://www.researchgate.net/publication/302617445_Multiphase_power_converters_controllers_and_control_methods (dostęp 25.06.2020)
- [6] Amazon Web Services, *The FreeRTOS Reference Manual*, Amazon.com 2017,
https://www.freertos.org/wp-content/uploads/2018/07/FreeRTOS_Reference_Manual_V10.0.0.pdf (dostęp 3.06.2020)
- [7] R. Barry, *Mastering the FreeRTOS Real Time Kernel*, Real Time Engineers, 2016,
https://www.freertos.org/wp-content/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf (dostęp 3.06.2020)
- [8] ST Microelectronics, Oficjalne repozytorium oprogramowania TouchGFX w serwisie GitHub, <https://github.com/touchgfx/touchgfx-open-repository> (dostęp 3.06.2020)
- [9] ST Microelectronics, Repozytorium kodu źródłowego grafu dotykowego pulpitu będący częścią oficjalnego repozytorium oprogramowania TouchGFX w serwisie GitHub, <https://github.com/touchgfx/touchgfx-open-repository/tree/master/widgets/Graph> (dostęp 3.06.2020)
- [10] ST Microelectronics, Dokumentacja środowiska TouchGFX,
<https://support.touchgfx.com/docs/introduction/welcome> (dostęp 3.06.2020)
- [11] TouchGFX 4.13.0, ST Microelectronics, <https://www.touchgfx.com> (dostęp 3.06.2020)
- [12] IAR Embedded Workbench 8.40, IAR Systems, <https://www.iar.com/iar-embedded-workbench> (dostęp 3.06.2020)
- [13] FreeRTOS 7.6.0, Real Time Engineers Ltd., <https://www.freertos.org> (dostęp 3.06.2020)
- [14] QtCreator 4.11.2, The Qt Company, <https://www.qt.io> (dostęp 3.06.2020)

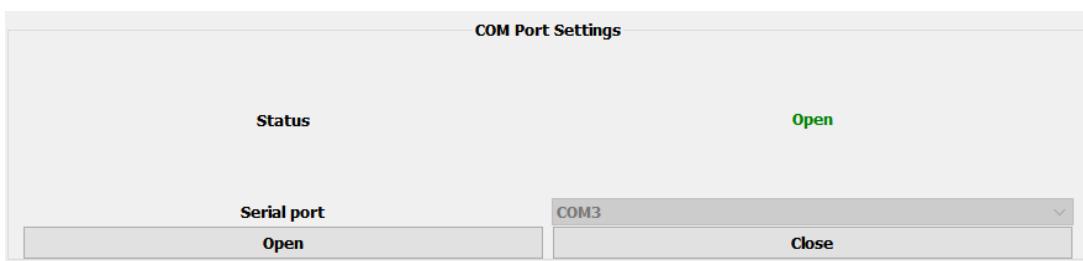
Dodatek A Aplikacja testowa PC

Aplikacja testowa uruchomiona na komputerze PC stanowi interfejs graficzny składający się z kilkudziesięciu widgetów pogrupowanych w mniejsze funkcjonalne zespoły. Część widgetów to kontrolki umożliwiające użytkownikowi wprowadzanie danych wysyłanych następnie do płytki Discovery. Pozostałe widgety realizują funkcję podglądu obecnego statusu i wartości poszczególnych parametrów oraz podglądu zawartości pakietów wysyłanych do i odbieranych z płytki Discovery.

Grupy widgetów

Poszczególne grupy widgetów można traktować jako niezależne funkcjonalnie moduły interfejsu graficznego. Każdy z nich ma własny layout i zakres odpowiedzialności.

Grupa COM Port Settings



Rys. A.1 Grupa COM Port Settings

Grupa odpowiadająca za kontrolę portów szeregowych komputera PC w tym:

- Wyświetlanie listy dostępnych portów szeregowych
- Umożliwienie wyboru portu szeregowego
- Otwarcie portu
- Zamknięcie portu
- Wyświetlanie statusu portu

Bezpośrednio po uruchomieniu aplikacji testowej grupa COM Port Settings jest jedyną aktywną grupą. Poprawne otwarcie dowolnego portu szeregowego wymagane jest by uzyskać dostęp do pozostałych grup interfejsu użytkownika.

W przypadku zamknięcia portu szeregowego dostęp do pozostałych grup interfejsu graficznego jest blokowany do czasu poprawnego ponownego otwarcia dowolnego portu szeregowego.

Grupa Module control

Module Control									
M1InitPar1	M1Value1	M2InitPar1	M2Value1	M3InitPar1	M3Value1				
M1InitPar2	M1Value2	M2InitPar2	M2Value2	M3InitPar2	M3Value2				
M1InitPar3	M1Value3	M2InitPar3	M2Value3	M3InitPar3	M3Value3				
M1InitPar4	M1Value4	M2InitPar4	M2Value4	M3InitPar4	M3Value4				
M1InitPar5	M1Value5	M2InitPar5	M2Value5	M3InitPar5	M3Value5				
Parameter 1 name	1	Parameter 1 name	1	Parameter 1 name	1				
Parameter 2 name	2	Parameter 2 name	2	Parameter 2 name	2				
Parameter 3 name	3	Parameter 3 name	3	Parameter 3 name	3				
Parameter 4 name	4	Parameter 4 name	4	Parameter 4 name	4				
Init Connection Module 1			Init Connection Module 2			Init Connection Module 3			
Deinit Connection Module 1			Deinit Connection Module 2			Deinit Connection Module 3			

Rys. A.2 Grupa Module control

Grupa odpowiadająca za funkcjonalności dotyczące inicjalizacji i deinicjalizacji połączenia z modułami płytka Discovery w tym:

- Wczytywanie nazw i wartości parametrów inicjalizacyjnych z pliku tekstowego
- Wyświetlanie nazw parametrów inicjalizacyjnych oraz ich wartości dla każdego z modułów
- Inicjalizacja modułu tj. zarządzanie wysyłaniem oraz zawartością 28 ramek inicjalizacyjnych przekazywanych do modułu płytka Discovery
- Deinicjalizacja modułu tj. zarządzanie wysyłaniem oraz zawartością ramki deinicjalizacyjnej przekazywanej do modułu płytka Discovery

Grupa Custom packet

Custom Packet	
Source	<input type="text" value="1"/>
Module	<input type="text" value="1"/>
Function	<input type="text" value="1"/>
Parameter	<input type="text" value="1"/>
Sign	<input type="text" value="1"/>
Length	<input type="text" value="7"/>
Payload	<input type="text" value="123.456"/>
Send	
Send wrong CRC packet	

Rys. A.3 Grupa Custom packet

Grupa udostępniająca użytkownikowi możliwość zbudowania i wysłania pojedynczego pakietu. Większość pól grupy nagłówkowej wybierana jest z listy rozwijanej. Pola długości i

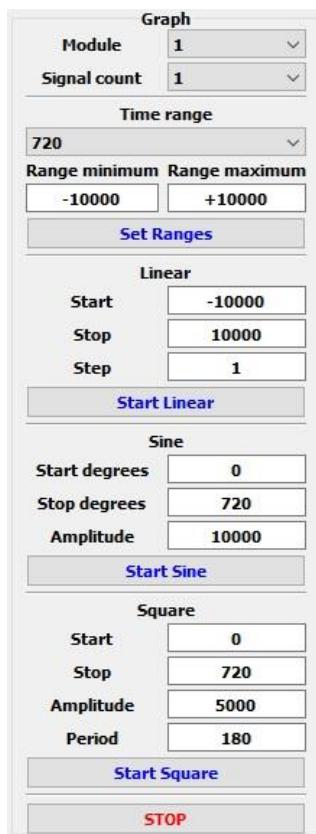
znaku są ustawiane przez aplikację testową automatycznie na podstawie analizy wartości wpisanej do pola payload.

Pole payload umożliwia wpisanie maksymalnie 10 znaków, które program interpretuje jako liczbę i umieszcza w payloadzie pakietu.

Suma CRC jest obliczana i umieszczana w pakiecie po wciśnięciu przycisku Send.

W obrębie grupy Custom packet istnieje możliwość celowego wysłania pakietu z błędnią wartością CRC w celach diagnostycznych. Większość pól takiego pakietu wypełniona jest wartościami zerowymi. Przesłanie takiego pakietu powinno spowodować zaświecenie wszystkich 4 LED sygnalizacyjnych na płytce Discovery co oznacza odebranie ramki o błędnej sumie CRC. W celu zgaszenia diód należy wcisnąć opisany wcześniej przycisk Clear LEDs znajdujący się na ekranie głównym interfejsu graficznego płytki Discovery.

Grupa Graph



Rys. A.4 Grupa Graph

Grupa Graph składa się z widgetów umożliwiających sterowanie i testowanie funkcjonalności grafu. Dostępne są następujące funkcje:

- Wybór modułu, do którego pakiety grafu mają być wysyłane
- Wybór ilości sygnałów, które użytkownik jednocześnie chce rysować na grafie, możliwy zakres to od 1 do 4.
- Możliwość wyboru zakresu czasowego grafu z listy rozwijanej. Możliwe wartości to wielokrotności liczby 360 w zakresie od 360 do 3600
- Możliwość wyboru dolnego oraz górnego zakresu danych

- Możliwość zdalnego ustawienia w płytce Discovery zakresów grafu przy użyciu przycisku Set Ranges
- Możliwość wysłania zestawu pakietów rysujących na ekranie grafu wykres liniowy z możliwością ustawienia wartości startowej, stopu oraz kroku
- Możliwość wysłania zestawu pakietów rysujących na ekranie grafu wykres sinusoidalny z możliwością ustawienia wartości startowej i stopu w postaci liczby stopni. Możliwe jest również ustawienie amplitudy obliczanej i przesyłanej sinusoidy. W przypadku rysowania więcej niż jednego przebiegu sinusoidalnego każdy sygnał jest przesunięty w fazie o 120 stopni.
- Możliwość wysłania zestawu pakietów rysujących na ekranie grafu do 2 wykresów przebiegu prostokątnego z możliwością ustawienia wartości startowej i stopu. Możliwe jest również ustawienie amplitudy oraz okresu obliczanego sygnału. W przypadku rysowania dwóch przebiegów prostokątnych jednocześnie, są one przesunięte względem siebie o pół okresu.
- Możliwość zatrzymania wysyłania pakietów grafu w dowolnym momencie poprzez naciśnięcie przycisku Stop

Wszystkie pola wejściowe interfejsu użytkownika przyjmujące wartości bezpośrednio do użytkownika są zabezpieczone validatorami uniemożliwiającymi np. wpisanie litery do pola oczekującego wyłącznie wartości liczbowych.

Grupa Packet display

Packet display																				
Source	Module	Function	Parameter	Sign	Length	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	CRC1	CRC2	CRC3	CRC4	
23	1	1	2	0	1	6	M	1	P	a	r	9	e	5	r	5	237	241	36	174
24	1	1	2	0	1	7	M	1	P	a	r	1	0	5	r	5	211	43	207	2

Rys. A.5 Grupa Packet display

Grupa składająca się z widoku tabeli (ang. table view) oraz przycisku czyszczącego zawartość tabeli na żądanie. Tabela ma 20 opisanych kolumn, po jednym dla każdego pola pakietu.

Każdorazowe przesłanie lub odebranie pakietu przez aplikację testową powoduje dodanie do tabeli wiersza zawierającego zawartość każdego pola pakietu.

Każdy wiersz tabeli opisujący pakiet wysłany do płytka Discovery ma białe tło, podczas gdy dla rozróżnienia wiersze opisujące pakiety odebrane z płytka Discovery mają tło zielone.

Grupy Module 1/2/3

	Module 1	
M1Par1	Disabled	1
M1Par2	Disabled	2
M1Par3	Disabled	3
M1Par4	Disabled	4
M1Par5	Disabled	5
M1Par6	Disabled	6
M1Par7	Disabled	7
M1Par8	Disabled	8
M1Par9	Disabled	9
M1Par10	Disabled	10

Rys. A.6 Grupy Module 1/2/3

W dolnej części ekranu aplikacji testowej znajdują się 3 grupy widgetów - Module 1, Module 2 oraz Module 3.

Każda z grup zorganizowana jest w formie 3-kolumnowej tabeli. Każda kolumna reprezentuje odpowiednio:

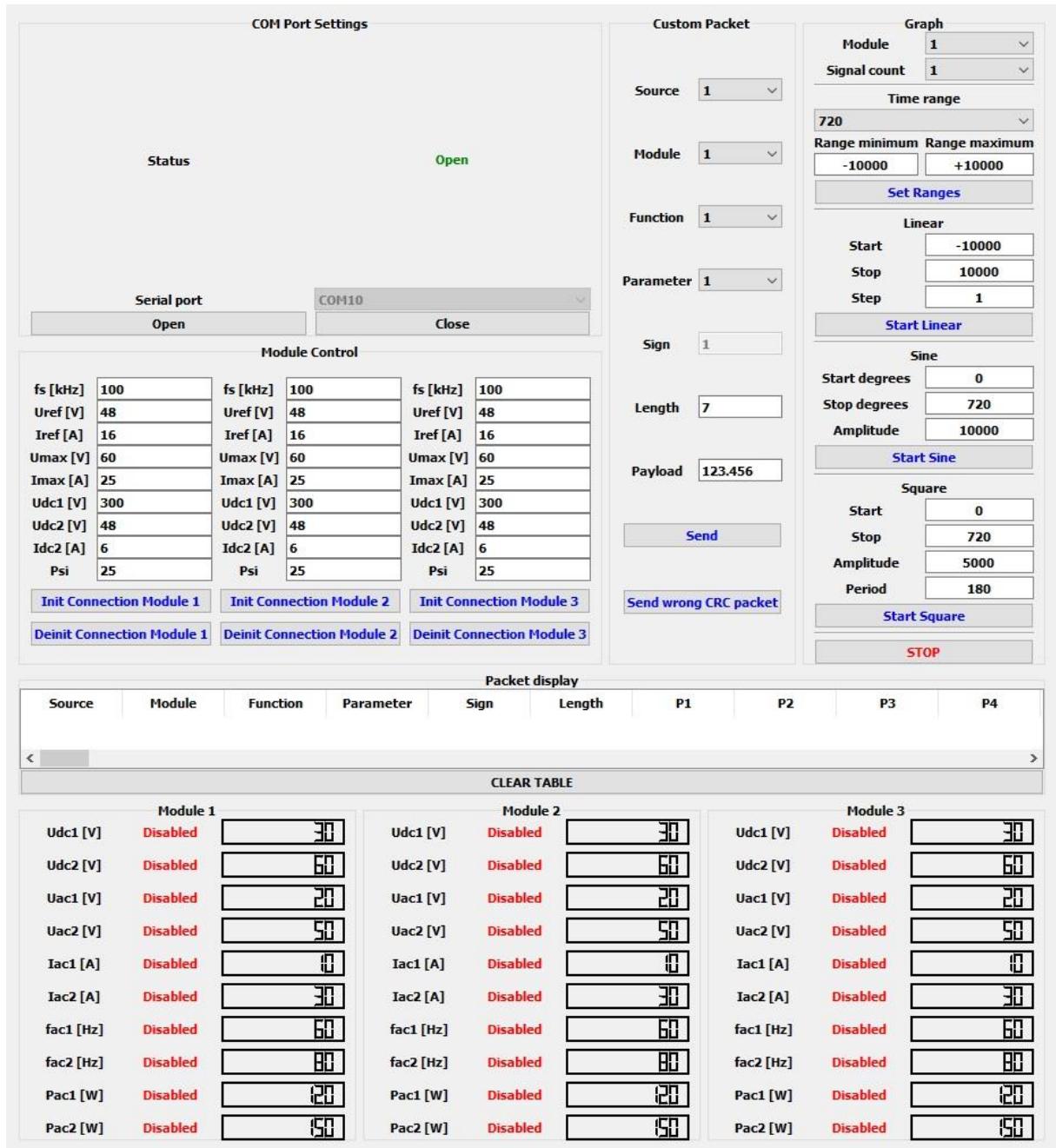
- Nazwy poszczególnych parametrów zainicjalizowane z pliku tekstowego
- Stan parametru – włączony lub wyłączony
- Wartość parametru

Każda z tabel umożliwia wygodny podgląd obecnego stanu, wartości parametrów oraz ich zmian w wyniku przetwarzania pakietów konfiguracyjnych odbieranych z płytki Discovery.

Dodatek B Ekrany aplikacji testowej

Niniejszy dodatek prezentuje - dla celów poglądowych - zrzuty ekranów aplikacji testowej będącej na różnych etapach działania.

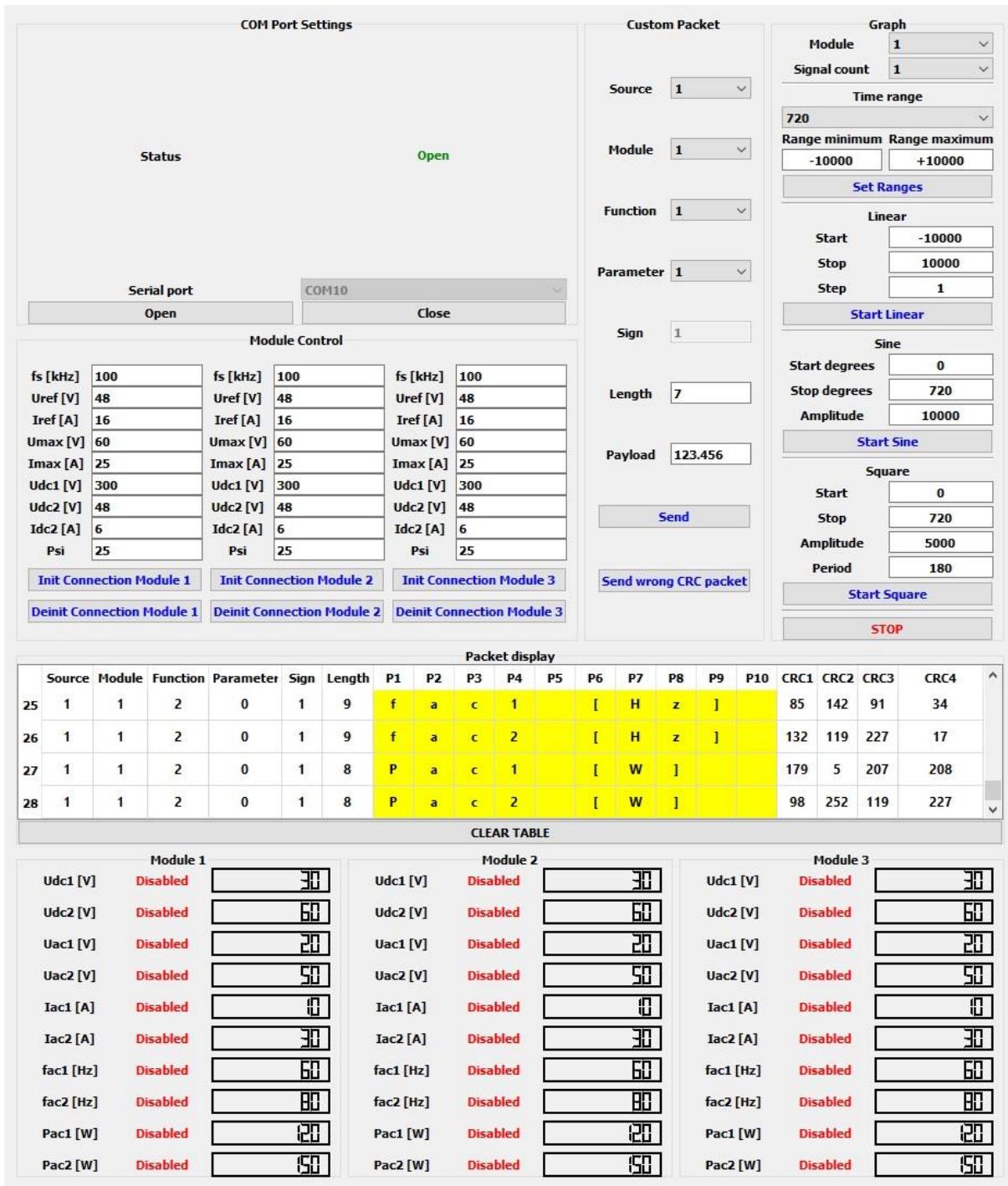
Aplikacja testowa po otwarciu portu komunikacyjnego



Rys. B.1 Zrzut ekranu aplikacji testowej po otwarciu portu komunikacyjnego

Rys. B.1 przedstawia zrzut ekranu graficznego interfejsu aplikacji testowej tuż po otwarciu portu komunikacyjnego. Na tym etapie wszystkie grupy widgetów są aktywne i aplikacja jest gotowa do wysyłania i odbierania pakietów.

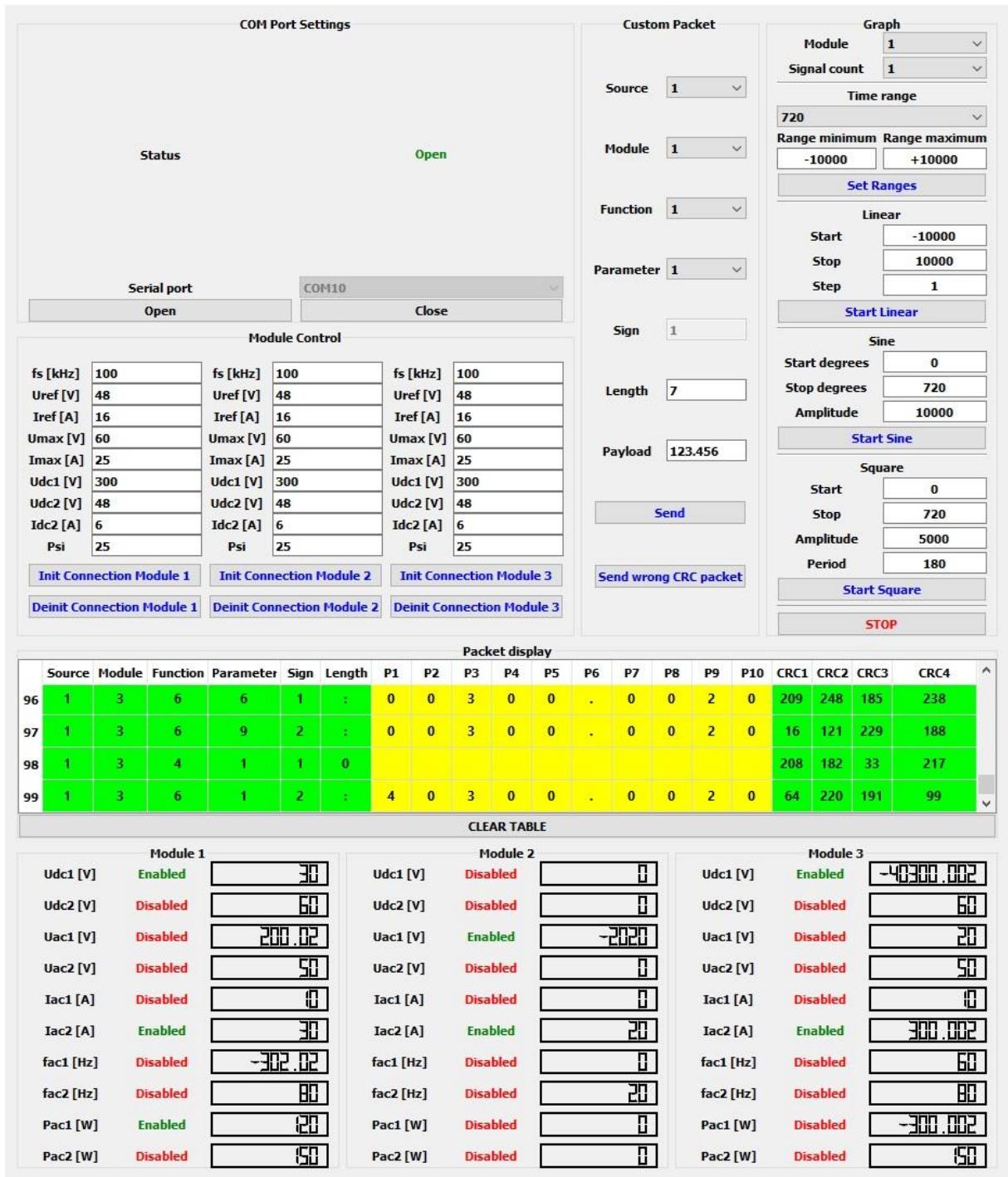
Aplikacja testowa po inicjalizacji połączenia



Rys. B.2 Zrzut ekranu aplikacji testowej po inicjalizacji połączenia

Rys. B.2 przedstawia zrzut ekranu graficznego interfejsu aplikacji testowej po inicjalizacji połączenia z modułem nr 3. Widok tabeli z grupy widgetów Packet display wyświetla zawartość każdego z pól poszczególnych pakietów. Możliwe jest przewinięcie widoku tabeli w górę co umożliwia podgląd historii odebranych i wysłanych pakietów. Widoczne jest 4 spośród 28 zarejestrowanych pakietów inicjalizacyjnych zawierających nazwy i wartości parametrów inicjalizacyjnych. Zastosowano żółte tło dla pól od P1 do P10 dla uwidocznienia payloadu pakietów.

Aplikacja testowa po aktywacji i ustawieniu części parametrów



Rys. B.3 Widok aplikacji testowej po aktywacji i ustawieniu części parametrów

Rys. B.3 przedstawia zrzut ekranu graficznego interfejsu aplikacji testowej po aktywacji i ustawieniu części parametrów każdego z modułów. Można zaobserwować, że część parametrów z każdego z modułów została aktywowana – tekst Enabled w kolorze zielonym – oraz części z parametrów została przypisana wartość. Zielone tło w tabeli podglądu pakietów oznacza, że dany pakiet jest pakietem przychodzącym.

Dodatek C Oprogramowanie płytki STM32F469-Discovery

Aplikacja uruchomiona na płytce Discovery składa się z 3 głównych elementów:

- Przygotowanie środowiska
- Definicje zadań (ang. tasks) systemu FreeRTOS
- Obsługa modelu MVP oprogramowania TouchGFX
- Definicje ekranów oprogramowania TouchGFX

Przygotowanie środowiska

Przed właściwym uruchomieniem systemu FreeRTOS oraz inicjalizacją bibliotek TouchGFX potrzebne jest przygotowanie środowiska. W tym celu należy wykonać kilka różnorodnych akcji.

- Inicjalizacja podzespołów płytki Discovery odpowiadających za obsługę ekranu dotykowego
- Inicjalizacja części oprogramowania TouchGFX odpowiadającej za bezpośrednią komunikację z obsługą sprzętową ekranu dotykowego
- Inicjalizacja interfejsów USART tj. USART3 – diagnostycznego - oraz USART6 – komunikacyjnego
- Inicjalizacja bufora Canvas służącego do przechowywania obiektów narysowanych w danym momencie na ekranie dotykowym
- Stworzenie zdefiniowanych wcześniej zadań systemu FreeRTOS wraz z przydzieleniem każdemu z nich priorytetu oraz rozmiaru dostępnej pamięci stosu
- Aktywacja przerwania interfejsu USART6 wraz z definicją rutyny obsługującej to przerwanie
- Inicjalizacja semaforów i kolejek na potrzeby synchronizacji odbioru i nadawania pakietów oraz przesyłania danych między poszczególnymi zadaniami FreeRTOS
- Przedefiniowanie funkcji przetwarzającej wyjście znakowe (fputc) w celu przekierowania wiadomości diagnostycznych na interfejs USART3
- Aktywacja planisty (ang. scheduler) FreeRTOS

Zadania systemu FreeRTOS

W ramach aplikacji zdefiniowane są 3 zadania, którym system FreeRTOS przydziela czas procesora w zależności od statusu gotowości danego zadania oraz jego priorytety.

Zadanie guiTask

Jest to zadanie domyślnie zdefiniowane w kodzie wygenerowanym przez środowisko TouchGFX. W ramach tego zadania wykonywane są wszelkie czynności związane z obsługą ekranu dotykowego takie jak:

- Rysowanie obiektów
- Odświeżanie ekranu z częstotliwością 60Hz
- Przetwarzanie sygnałów wejścia w postaci dotyku ekranu dotykowego
- Transfer danych od poziomu widoku do poziomu modelu oraz od poziomu modelu do poziomu widoku zgodnie z koncepcją Model-View-Presenter. Dane dostępne na poziomie widoku (View) są gotowe do wyświetlenia na ekranie, podczas gdy dane

dostępne na poziomie modelu (Model) są gotowe do przekazania z lub do innych zadań systemu FreeRTOS.

Zadanie uartRxTask

Zadanie służące do obsługi danych przychodzących do płytki Discovery z aplikacji testowej. Jest ono aktywowane z poziomu rutyny obsługującej przerwanie USART6 każdorazowo po odebraniu 20 ramek UART.

Zadanie to ma na celu zebranie odebranych wcześniej 20 ramek UART w 1 pełny pakiet, a następnie sprawdzenie jego poprawności poprzez porównanie sumy CRC pakietu z nową sumą CRC obliczoną na podstawie reszty pól pakietu.

W przypadku pozytywnej weryfikacji poprawności pakietu jest on przekazywany z użyciem kolejki do zadania guiTask. W przeciwnym razie pakiet jest odrzucany wraz z przesłaniem odpowiedniej wiadomości diagnostycznej na interfejsie USART3.

Zadanie uartTxTask

Zadanie służące do obsługi danych wychodzących z płytki Discovery w kierunku aplikacji testowej. Ma ono najwyższy priorytet i aktywowane jest w przypadku naciśnięcia przez użytkownika stosownego przycisku na ekranie Settings.

Zgodnie z modelem MVP na podstawie danych wynikających ze stanu poszczególnych obiektów na ekranie (View) budowany jest pakiet przekazywany poprzez prezentera (Presenter) aż do modelu (Model) z którego ostatecznie przekazywane są z użyciem kolejki do zadania uartTxTask.

Obsługa modelu TouchGFX MVP

Obsługa modelu MVP oprogramowania TouchGFX polega na wydzieleniu dla każdego ekranu klasy Presenter i klasy View oraz jednej wspólnej klasy Model. Dzięki takiemu podziałowi możliwe jest swobodne rozdzielenie odpowiedzialności pomiędzy różne klasy.

Tabela C.1 Opis odpowiedzialności modelu MVP

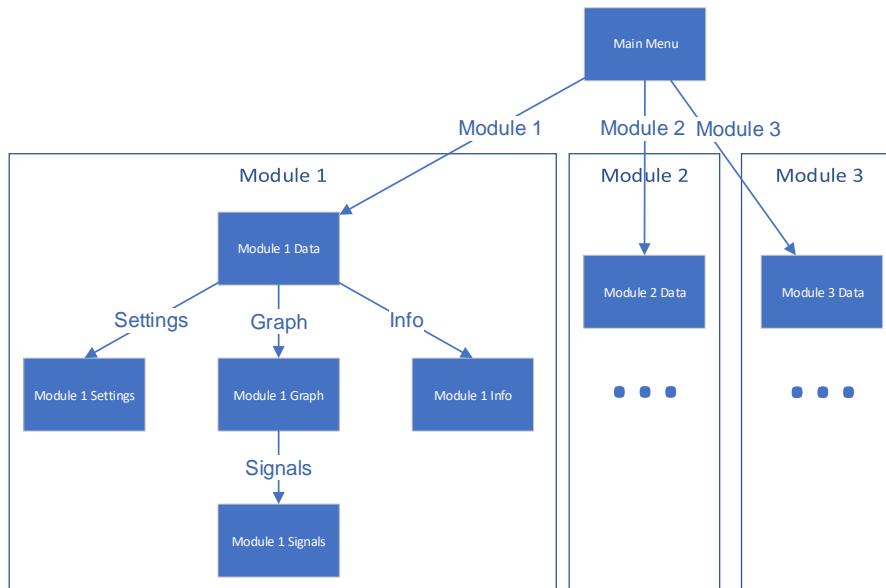
Klasa	Odpowiedzialność
View	Wyświetlanie i zarządzanie elementami widocznymi na ekranie dotykowym
Presenter	Zawiera logikę inicjalizacyjną oraz deinicjalizacyjną ekranu Służy jako łącznik w transferze danych na linii Model-View
Model	Przechowuje informacje interfejsu użytkownika Łączy graficzny interfejs użytkownika z peryferiami płytki Discovery oraz innymi zadaniami systemu FreeRTOS

Ekrany TouchGFX

Aplikacja obsługuje łącznie 16 ekranów (warstw) składających się na graficzny interfejs użytkownika płytki Discovery. Jeden z ekranów - ekran główny - jest wspólny dla wszystkich 3 modułów, a pozostałe 5 ekranów jest unikalne dla każdego z modułów tj. każdy z nich posiada własną przestrzeń adresową, a co za tym idzie przechowuje inne dane. Każdy z ekranów wyświetla inne dane oraz udostępnia inne funkcjonalności. Ponadto każdy ekran w prawym dolnym rogu prezentuje obecny procent obciążenia procesora. Ekrany tworzą hierarchię

przedstawioną na Rys. C.1. Nazwy którymi opisane są strzałki to etykiety przycisków znajdujących się na danym ekranie pozwalające przejść do innego ekranu.

Dodatkowo każdy ekran oprócz głównego posiada przycisk o etykiecie „Back”, który pozwala przejść do ekranu znajdującego się o poziom wyżej w hierarchii.



Rys. C.1 Diagram hierarchii ekranów

Dodatek A do niniejszej pracy zawiera zestaw rysunków poglądowych prezentujących ekran główny oraz każdy z ekranów modułu 1 zainicjalizowany przykładowymi danymi wraz z krótkimi opisami co użytkownik może zrobić i zobaczyć z poziomu każdego z ekranów.

Diagnostyka błędów

Komunikacja w systemie opiera się na założeniu, że urządzenia przesyłają między sobą pakiety UART o ścisłe określonej długości i zawartości. Każdy pakiet ma długość 20 bajtów, a każde pole pakietu może zawierać jedną ze zdefiniowanych wcześniej dozwolonych wartości. Każde odstępstwo od tych reguł będzie stanowić błąd transmisji i w każdym przypadku błędny pakiet zostanie odrzucony i nie będzie procesowany.

Błędy CRC

Zakłócenia i szумy elektryczne mogą wywołać zmianę wartości pojedynczych bajtów pakietów UART. System jest zabezpieczony przed tego typu błędami dzięki zastosowaniu mechanizmu obliczania i weryfikacji sumy kontrolnej CRC.

Każde urządzenie nadające buduje pakiet UART – 6 bajtów nagłówka i 10 bajtów payloadu po czym oblicza i dołącza do pakietu UART 4-bajtową sumę CRC wyliczoną na podstawie pierwszych 16 bajtów danego pakietu. Następnie każde urządzenie odbierające pakiet UART oblicza sumę CRC pierwszych 16 pól odebranego pakietu i porównuje z sumą CRC odczytaną z odebranego pakietu. W przypadku wystąpienia niezgodności sumy CRC obliczonej z sumą otrzymaną pakiet jest odrzucony i nie jest dalej procesowany. Dodatkowo zapalane są

wszystkie 4 diody płytka Discovery. Diody te można zgasić wciskając przycisk z napisem „Clear LEDs” znajdujący się w prawym górnym rogu ekranu głównego Main Menu.

Z poziomu aplikacji testowej błąd CRC powoduje wyświetlenie ostrzegawczego okna dialogowego.

Niedozwolone wartości pól

Inne możliwe błędy są mniej widoczne dla użytkownika. Z poziomu płytka Discovery zdiagnozować można je wyłącznie poprzez interfejs diagnostyczny USART3. W przypadku gdy którykolwiek pole pakietu UART ma wartość niedozwoloną na wcześniej wymienionym interfejsie diagnostycznym pojawia się wiadomość informująca użytkownika które pole powoduje problem.

Z poziomu aplikacji testowej niedozwolona wartość pola w odebranym pakiecie powoduje wyświetlenie ostrzegawczego okna dialogowego.

Dodatek D Pseudokod aplikacji testowej sterownika

Aplikacja testowa PC stworzona w ramach niniejszej pracy jest niezbędna do efektywnego testowania każdej funkcjonalności aplikacji płytki Discovery. Umożliwia ona odbiór pakietów odebranych poprzez interfejs UART, a następnie intuicyjny odczyt tych danych wyświetlonych w postaci tabeli na graficznym interfejsie użytkownika.

Z drugiej zaś strony aplikacja testowa umożliwia łatwe przesyłanie danych do płytki pozwalając na uruchomienie każdej funkcjonalności poprzez wybranie odpowiednich wartości z list rozwijanych lub bezpośrednie wpisanie ich w pola tekstowe, a następnie wcisnięcie dedykowanego przycisku na graficznym interfejsie użytkownika. W tym przypadku - podobnie jak dla danych odebranych – kluczowa jest możliwość podglądu na żywo danych wysyłanych do płytki Discovery. Dane te również są dostępne do podglądu w wspomnianej już wcześniej tabeli.

Taki sposób testowania aplikacji płytki Discovery jest nieporównywalnie łatwiejszy i wygodniejszy niż bezpośrednią implementację i testowanie komunikacji interfejsem UART między dwoma mikrokontrolerami w związku z czym stworzenie aplikacji testowej było kluczowe dla projektu i w zasadzie nieuniknione.

Należy jednak wziąć pod uwagę, że docelowy system ma składać się z płytki Discovery komunikującej się bezpośrednio ze sterownikiem przekształtnika energoelektrycznego, który w praktyce jest mniej lub bardziej zaawansowanym mikrokontrolerem. Z tego względu pomocnym może się okazać zapisanie kluczowych funkcjonalności aplikacji testowej w postaci pseudokodu stanowiącego szablon ułatwiający napisanie właściwego kodu dla docelowego mikrokontrolera.

Pakiet UART i jego struktura danych

Pakiet UART przechowywany jest w pamięci w postaci 20 kolejnych bajtów. Poniższy kod przedstawia implementację wspomnianego pakietu jako strukturę w języku C, gdzie każdy bajt ma odpowiednią nazwę ułatwiającą pracę z pakietem – jego zapis oraz odczyt. Ta forma pakietu stosowana jest przy realizacji wszystkich funkcjonalności systemu.

Kod w języku C:

```
struct UartPacket
{
    uint8_t Source;
    uint8_t Module;
    uint8_t Function;
    uint8_t Parameter;
    uint8_t Sign;
    uint8_t Length;
    uint8_t Payload[10];
    uint8_t Crc[4];
};
```

Pseudokod poszczególnych funkcjonalności systemu

Nadawanie danych

Mikrokontroler przesyła dane do płytki Discovery np. w celu ich wizualizacji z użyciem grafu zgodnie z poniższym schematem. Założono tutaj, że chcemy wysłać pakiet przedstawiający parametr 'b' do modułu nr 1 urządzenia nr 1.

Pseudokod w języku C:

```
/*Argument value to liczba zmiennoprzecinkowa o długości maksymalnie 10 znaków*/
void SendData(double value)
{
    /*Inicjalizacja struktury przechowującej pola pakietu zerami*/
    struct UartPacket uartPacket = {0};

    /*Przypisanie odpowiednich wartości znakowych kolejnym bajtom pakietu*/
    uartPacket.Source = '1';
    uartPacket.Module = '1';
    uartPacket.Function = '1';
    uartPacket.Parameter = 'b';

    /*Sprawdzenie znaku przesyłanej liczby i przypisanie do pola pakietu odpowiedniej
    wartości*/
    if(value >= 0)
    {
        uartPacket.Sign = '1'
    }
    else
    {
        uartPacket.Sign = '2'
    }

    /*Tablica znakowa przechowująca liczbę w postaci łańcucha znaków*/
    char valueAsString[10];

    /*Konwersja liczby na postać znakową*/
    ConvertNumericToString(value, valueAsString);

    /*Przypisanie długości łańcucha do zmiennej*/
    int stringLength = length(valueAsString);

    /*Konwersja długości łańcucha do postaci znakowej i przypisanie do odpowiedniego
    bajtu tablicy pakietu*/
    uartPacket.Length = ToCharacter(stringLength);

    /*Wpisanie znaków liczby w postaci łańcucha znaków do kolejnych odpowiednich pól
    tablicy pakietu*/
    for(int i=0; i < stringLength; i++)
    {
        uartPacket.Payload[i] = valueAsString[i];
    }

    /*Obliczenie sumy kontrolnej CRC pakietu*/
    crc32 = CalculateCrc32(uartPacket);

    /*Zapisanie obliczonej sumy CRC w czterech ostatnich bajtach pakietu*/
    AppendCrc32ToPacket(uartPacket, crc32);

    /*Przesłanie pakietu poprzez UART*/
    SendPacketViaUart(uartPacket);
}
```

Odbiór danych

Mikrokontroler odbiera dane z płytki Discovery za pośrednictwem modułu UART. Następnie otrzymany pakiet jest walidowany tzn. sprawdzane jest czy suma CRC obliczona na podstawie zawartości pól pakietu jest zgodna z sumą CRC otrzymaną w pakiecie. Na tej podstawie podejmowana jest decyzja czy pakiet jest poprawny i powinien być dalej przetwarzany.

Pseudokod w języku C:

```
struct UartPacket PacketReceived()
{
    /*Inicjalizacja struktury przechowującej pola pakietu zerami*/
    struct UartPacket uartPacket = {0};

    /*Skopiowanie 20 bajtów odebranych przez moduł UART do struktury reprezentującej
     *pakiet*/
    memcpy(uartPacket, RECEIVED_PACKET, 20);

    /*Obliczenie 32-bitowej sumy kontrolnej CRC pakietu*/
    int crc32 = CalculateCrc32(uartPacket);

    /*Sprawdzenie zgodności obliczonej sumy CRC z sumą odebraną w pakiecie*/
    if(CheckCrc(uartPacket, crc32))
    {
        /*Pakiet poprawny i gotowy do dalszego procesowania*/
        return uartPacket;
    }
    else
    {
        /*Pakiet jest niepoprawny i zostaje odrzucony*/
    }
}
```

Inicjalizacja połączenia

Mikrokontroler inicjalizuje połączenie poprzez przesłanie do płytki Discovery 28 pakietów inicjalizacyjnych zgodnie z poniższym schematem. Założono tutaj, że chcemy zainicjalizować moduł nr 1 urządzenia nr 1.

Pseudokod w języku C:

```
char initParameters[28][10]    /*Bufor zawierający dane dla 28 pakietów
inicjalizacyjnych, do 10 znaków każdy*/

void InitModule(initParameters)
{
    /*Inicjalizacja struktury przechowującej pola pakietu zerami*/
    struct UartPacket uartPacket = {0};
    /*Przypisanie odpowiednich wartości znakowych kolejnym bajtom pakietu*/
    uartPacket.Source = '1';
    uartPacket.Module = '1';
    uartPacket.Function = '2';
    uartPacket.Parameter = '0';
    uartPacket.Sign = '1';

    /*Znak ':' oznacza długość 10*/
    uartPacket.Length = ':';

    /*Przetwarzanie wszystkich 28 wartości inicjalizacyjnych*/
    for(int i=0; i < 28; i++)
    {
        /*Kopiowanie parametru inicjalizacyjnego do pól Payload pakietu*/
        memcpy(uartPacket.Payload, initParameters[i], 10);
    }
}
```

```

/*Obliczanie sumy CRC pakietu*/
int crc32 = CalculateCrc32(uartPacket);

/*Zapisanie sumy CRC w 4 ostatnich bajtach pakietu*/
AppendCrc32ToPacket(uartPacket, crc32);

/*Przesłanie pakietu poprzez UART*/
SendPacketViaUart(uartPacket);
}
}

```

Deinicjalizacja połączenia

Mikrokontroler deinicjalizuje połączenie poprzez przesłanie do płytki Discovery 1 pakietu deinicjalizacyjnego zgodnie z poniższym schematem. Założono tutaj, że chcemy zdeinicjalizować moduł nr 1 urządzenia nr 1.

Pseudokod w języku C:

```

void DeinitModule ()
{
    /*Inicjalizacja struktury przechowującej pola pakietu zerami*/
    struct UartPacket uartPacket = {0};

    /*Przypisanie odpowiednich wartości znakowych kolejnym bajtom pakietu*/
    uartPacket.Source = '1';
    uartPacket.Module = '1';
    uartPacket.Function = '3';
    uartPacket.Parameter = '0';
    uartPacket.Sign = '1';
    uartPacket.Length = '0';

    /*Obliczanie sumy CRC pakietu*/
    int crc32 = CalculateCrc32(uartPacket);

    /*Zapisanie sumy CRC w 4 ostatnich bajtach pakietu*/
    AppendCrc32ToPacket(uartPacket, crc32);

    /*Przesłanie pakietu poprzez UART*/
    SendPacketViaUart(uartPacket);
}

```

Aktywacja parametru

Mikrokontroler aktywuje parametr, gdy odbierze od płytki Discovery 1 pakiet aktywacyjny zgodnie z poniższym schematem. Założono tutaj, że poprzez moduł UART odebrany został pakiet aktywujący parametr nr 1 modułu nr 1 urządzenia nr 1.

Pseudokod w języku C:

```
void ActivateParameter ()  
{  
    /*Inicjalizacja struktury przechowujacej pola pakietu danymi z odebranego pakietu*/  
    struct UartPacket uartPacket = PacketReceived();  
  
    /*Aktywacja parametru nr 1, którego stan przechowywany jest w zmiennej w pamięci*/  
    if(uartPacket.Module == '1')  
    {  
        if(uartPacket.Function == '4')  
        {  
            if(uartPacket.Parameter == '1'  
            {  
                parameter1Enabled = true;  
            }  
        }  
    }  
}
```

Deaktywacja parametru

Mikrokontroler deaktywuje parametr, gdy odbierze od płytki Discovery 1 pakiet deaktywacyjny zgodnie z poniższym schematem. Założono tutaj, że odebrany zostanie pakiet deaktywujący parametr nr 1 modułu nr 1 urządzenia nr 1.

Pseudokod w języku C:

```
void DeactivateParameter ()  
{  
    /*Inicjalizacja struktury przechowujacej pola pakietu danymi z odebranego  
    pakietu*/  
    struct UartPacket uartPacket = PacketReceived();  
  
    /*Deaktywacja parametru nr 1, którego stan przechowywany jest w zmiennej w  
    pamięci*/  
    if(uartPacket.Module == '1')  
    {  
        if(uartPacket.Function == '5')  
        {  
            if(uartPacket.Parameter == '1'  
            {  
                parameter1Enabled = true;  
            }  
        }  
    }  
}
```

Ustawianie wartości parametru

Mikrokontroler ustawia wartość parametru, gdy odbierze od płytki Discovery 1 pakiet ustawiający, zgodnie z poniższym schematem. Założono tutaj, że odebrany zostanie pakiet ustawiający wartość parametru nr 1 modułu nr 1 urządzenia nr 1.

Pseudokod w języku C:

```
/*Argument value to liczba zmiennoprzecinkowa o długości maksymalnie 10 znaków*/
void SetParameterValue (double value)
{
    /*Inicjalizacja struktury przechowującej pola pakietu danymi z odebranego
    pakietu*/
    struct UartPacket uartPacket = PacketReceived();

    if(uartPacket.Module == '1')
    {
        if(uartPacket.Function == '6')
        {
            if(uartPacket.Parameter == '1')
            {
                /*Ustawienie wartości parametru nr 1 na value. Wartość ta
                przechowywana jest w zmiennej w pamięci*/
                parameter1Value = value;
            }
        }
    }
}
```

Ustawianie zakresu dolnego grafu

Mikrokontroler zdalnie ustawia dolny zakres grafu rysowanego na ekranie Graph płytki Discovery poprzez przesłanie 1 ramki ustawiającej zgodnie z poniższym schematem. Założono tutaj, że chcemy ustawić zakres dolny grafu modułu nr 1 urządzenia nr 1.

Pseudokod w języku C:

```
/*Argument value to liczba zmiennoprzecinkowa o długości maksymalnie 10 znaków która
ma stanowić dolny limit grafu*/
void SetGraphRangeMinimum (double value)
{
    /*Inicjalizacja struktury przechowującej pola pakietu zerami*/
    struct UartPacket uartPacket = {0};

    /*Przypisanie odpowiednich wartości znakowych kolejnym bajtom pakietu*/
    uartPacket.Source = '1';
    uartPacket.Module = '1';
    uartPacket.Function = '7';
    uartPacket.Parameter = '0';

    /*Sprawdzenie znaku przesyłanej liczby i przypisanie do pola pakietu
    odpowiedniej wartości*/
    if(value >= 0)
    {
        uartPacket.Sign = '1'
    }
    else
    {
        uartPacket.Sign = '2'
    }

    /*Tablica znakowa przechowująca liczbę w postaci łańcucha znaków*/
    char valueAsString[10];
```

```

/*Konwersja liczby na postać znakową*/
ConvertNumericToString(value, valueAsString);

/*Przypisanie długości łańcucha do zmiennej*/
int stringLength = length(valueAsString);

/*Konwersja długości łańcucha do postaci znakowej i przypisanie do
odpowiedniego bajtu tablicy pakietu*/
uartPacket.Length = ToCharacter(stringLength);

/*Wpisanie znaków liczby w postaci łańcucha znaków do kolejnych odpowiednich
pól tablicy pakietu*/
for(int i=0; i < stringLength; i++)
{
    uartPacket.Payload[i] = valueAsString[i];
}

/*Obliczenie sumy kontrolnej CRC pakietu*/
crc32 = CalculateCrc32(uartPacket);

/*Zapisanie obliczonej sumy CRC w czterech ostatnich bajtach pakietu*/
AppendCrc32ToPacket(uartPacket, crc32);

/*Przesłanie pakietu poprzez UART*/
SendPacketViaUart(uartPacket);
}

```

Ustawianie zakresu górnego grafu

Mikrokontroler zdalnie ustawia górny zakres grafu rysowanego na ekranie Graph płytki Discovery poprzez przesłanie 1 ramki ustawiającej zgodnie z poniższym schematem. Założono tutaj, że chcemy ustawić zakres górnego grafu modułu nr 1 urządzenia nr 1.

Pseudokod w języku C:

```

/*Argument value to liczba zmiennoprzecinkowa o długości maksymalnie 10 znaków która
ma stanowić górny limit grafu*/

void SetGraphRangeMaximum (double value)
{
    /*Inicjalizacja struktury przechowującej pola pakietu zerami*/
    struct UartPacket uartPacket = {0};

    /*Przypisanie odpowiednich wartości znakowych kolejnym bajtom pakietu*/
    uartPacket.Source = '1';
    uartPacket.Module = '1';
    uartPacket.Function = '8';
    uartPacket.Parameter = '0';

    /*Sprawdzenie znaku przesyłanej liczby i przypisanie do pola pakietu
    odpowiedniej wartości*/
    if(value >= 0)
    {
        uartPacket.Sign = '1'
    }
    else
    {
        uartPacket.Sign = '2'
    }

    /*Tablica znakowa przechowująca liczbę w postaci łańcucha znaków*/
    char valueAsString[10];

    /*Konwersja liczby na postać znakową*/
    ConvertNumericToString(value, valueAsString);
}

```

```

/*Przypisanie długości łańcucha do zmiennej*/
int stringLength = length(valueAsString);

/*Konwersja długości łańcucha do postaci znakowej i przypisanie do odpowiedniego bajtu tablicy pakietu*/
uartPacket.Length = ToCharacter(stringLength);

/*Wpisanie znaków liczby w postaci łańcucha znaków do kolejnych odpowiednich pól tablicy pakietu*/
for(int i=0; i < stringLength; i++)
{
    uartPacket.Payload[i] = valueAsString[i];
}

/*Obliczenie sumy kontrolnej CRC pakietu*/
crc32 = CalculateCrc32(uartPacket);

/*Zapisanie obliczonej sumy CRC w czterech ostatnich bajtach pakietu*/
AppendCrc32ToPacket(uartPacket, crc32);

/*Przesłanie pakietu poprzez UART*/
SendPacketViaUart(uartPacket);
}

```

Ustawianie zakresu czasowego grafu

Mikrokontroler zdalnie ustawia zakres czasowy grafu rysowanego na ekranie Graph płytka Discovery poprzez przesłanie 1 ramki ustawiającej zgodnie z poniższym schematem. Założono tutaj, że chcemy ustawić zakres czasowy grafu modułu nr 1 urządzenia nr 1.

Pseudokod w języku C:

```

/*Argument value to liczba całkowita w zakresie od 360 do 3600 stanowiąca wielokrotność liczby 360 mająca stanowić zakres czasowy grafu*/

void SetGraphTimeRange(int value)
{
    /*Inicjalizacja struktury przechowującej pola pakietu zerami*/
    struct UartPacket uartPacket = {0};

    /*Przypisanie odpowiednich wartości znakowych kolejnym bajtom pakietu*/
    uartPacket.Source = '1';
    uartPacket.Module = '1';
    uartPacket.Function = '9';
    uartPacket.Parameter = '0';
    uartPacket.Sign = '1'

    /*Tablica znakowa przechowująca liczbę w postaci łańcucha znaków*/
    char valueAsString[10];

    /*Konwersja liczby na postać znakową*/
    ConvertNumericToString(value, valueAsString);

    /*Przypisanie długości łańcucha do zmiennej*/
    int stringLength = length(valueAsString);

    /*Konwersja długości łańcucha do postaci znakowej i przypisanie do odpowiedniego bajtu tablicy pakietu*/
    uartPacket.Length = ToCharacter(stringLength);

    /*Wpisanie znaków liczby w postaci łańcucha znaków do kolejnych odpowiednich pól tablicy pakietu*/
    for(int i=0; i < stringLength; i++)
    {
        uartPacket.Payload[i] = valueAsString[i];
    }
}

```

```
/*Obliczenie sumy kontrolnej CRC pakietu*/
crc32 = CalculateCrc32(uartPacket);

/*Zapisanie obliczonej sumy CRC w czterech ostatnich bajtach pakietu*/
AppendCrc32ToPacket(uartPacket, crc32);

/*Przesłanie pakietu poprzez UART*/
SendPacketViaUart(uartPacket);
}
```

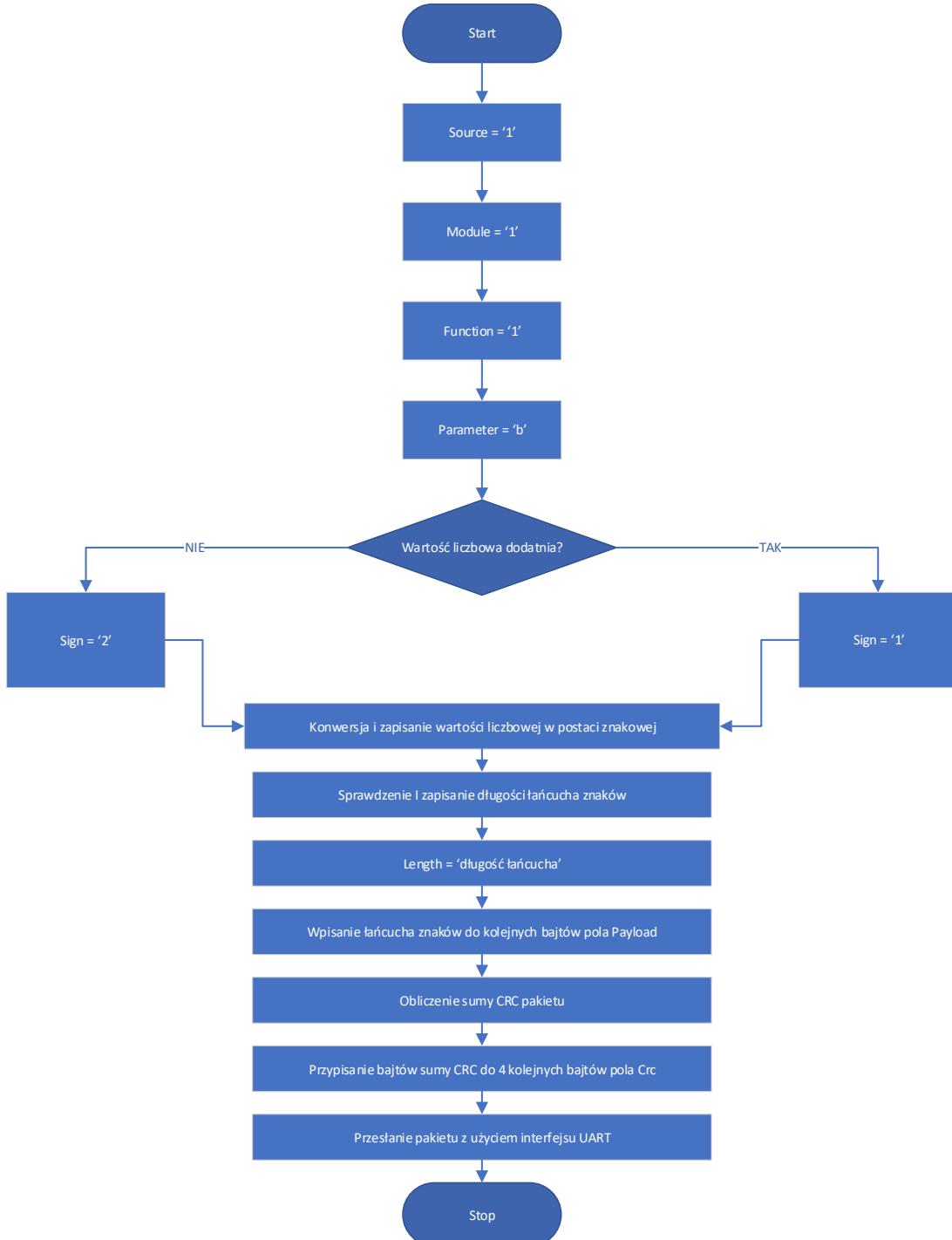
Schematy blokowe funkcjonalności systemu

Niniejszy podrozdział prezentuje schematy blokowe funkcjonalności realizowanych w systemie.

Nadawanie danych

Warunki początkowe:

- Określona jest liczba zmiennoprzecinkowa o długości maksymalnie 10 znaków

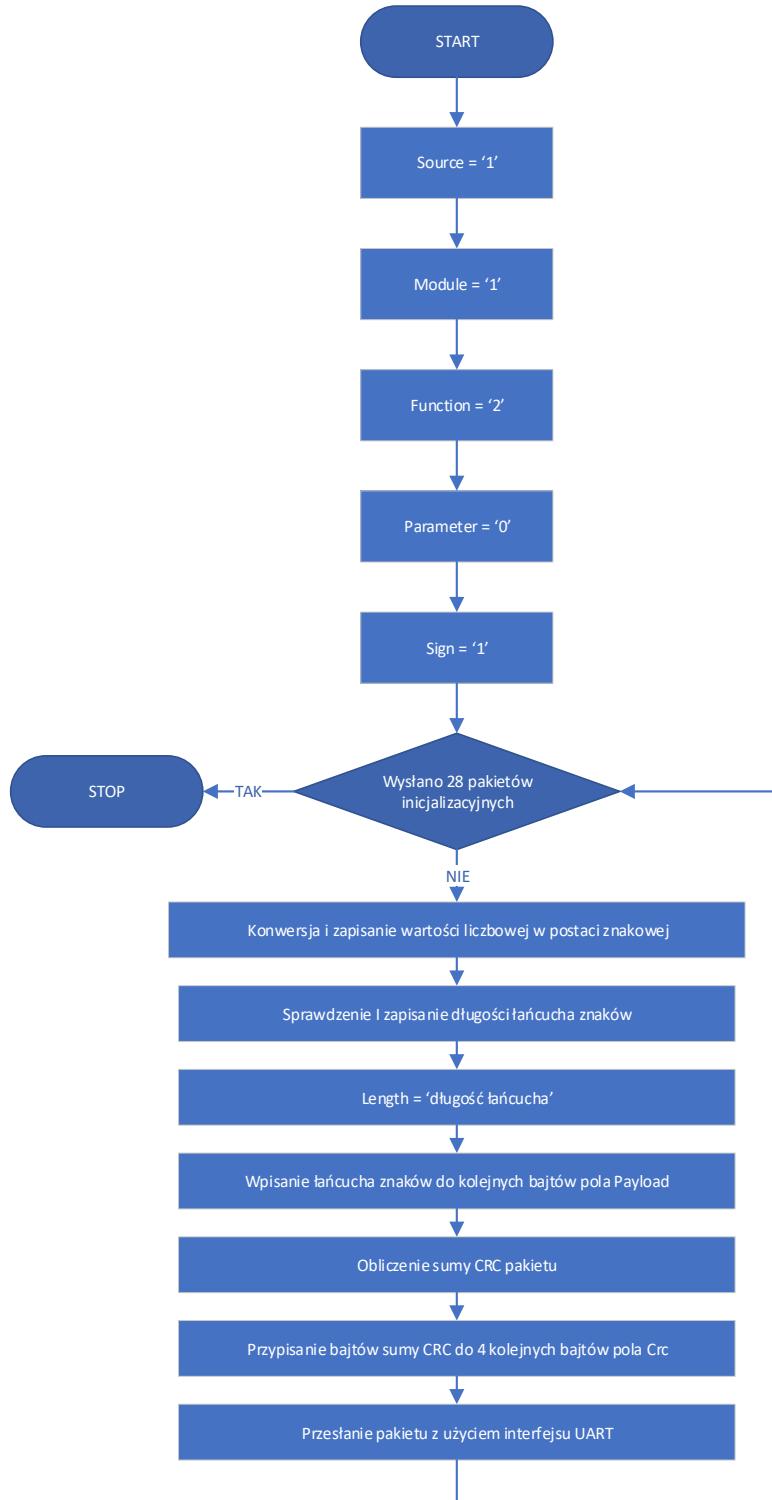


Rys. D.1 Schemat blokowy funkcjonalności Nadawanie danych

Iinicjalizacja połączenia

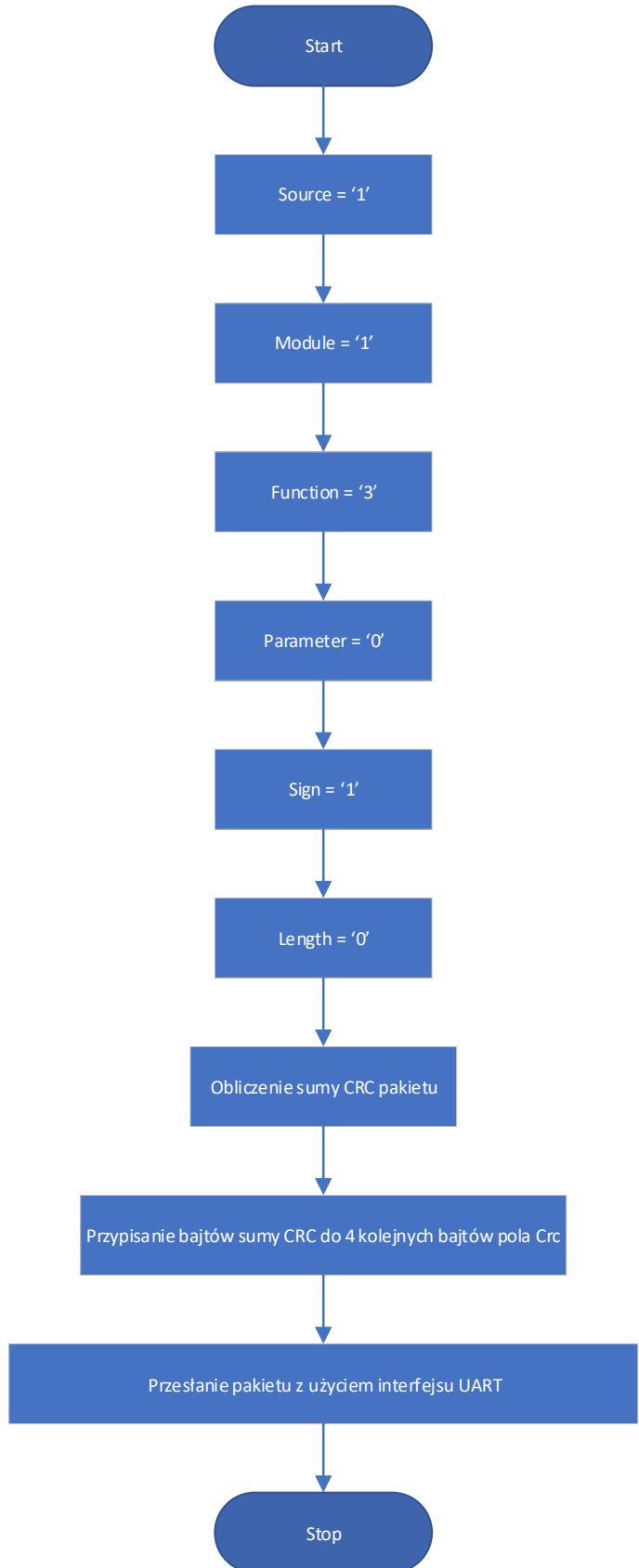
Warunki początkowe:

- Nazwy i wartości parametrów inicjalizacyjnych wczytane z dowolnego źródła i obecne w pamięci



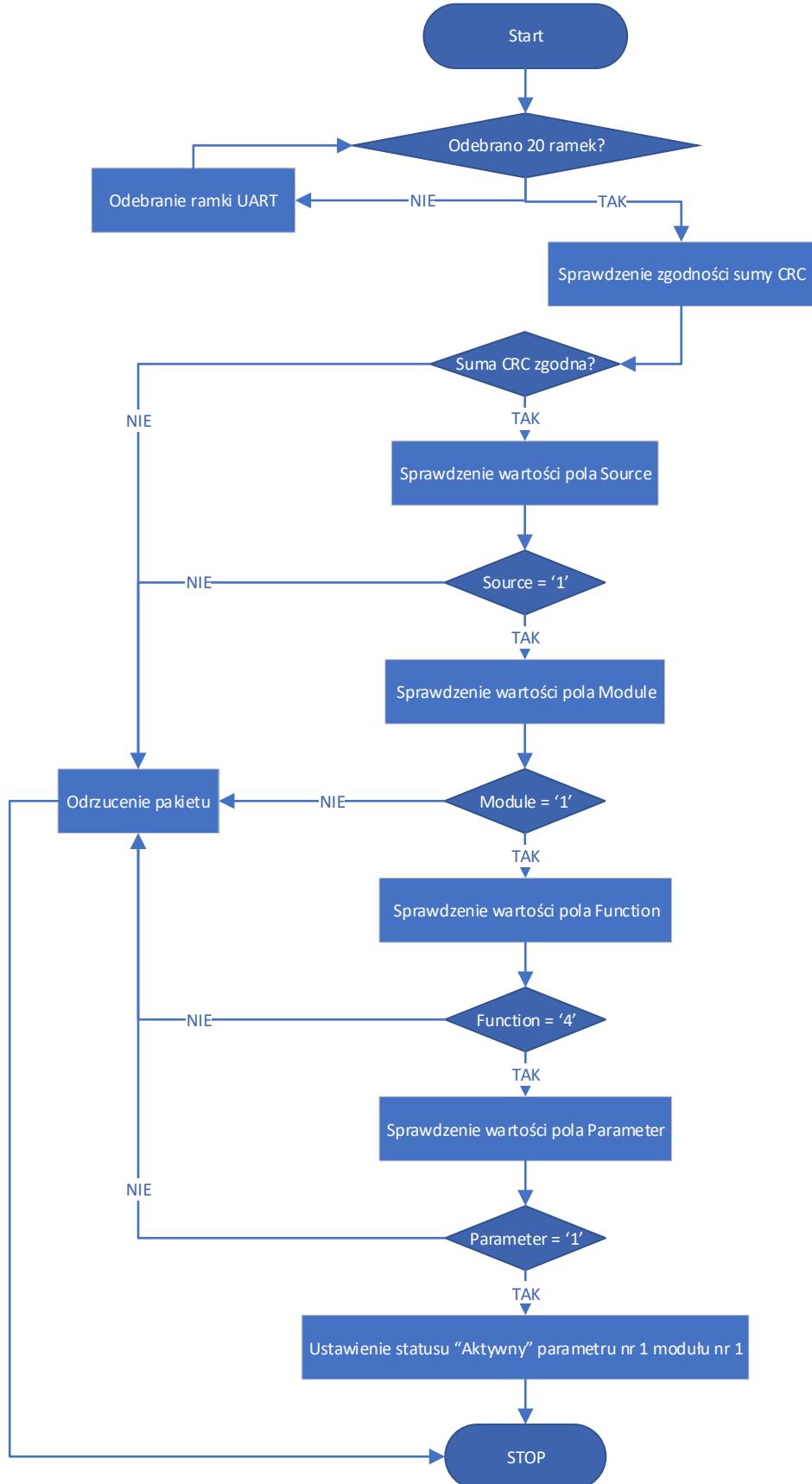
Rys. D.2 Schemat blokowy funkcjonalności inicjalizacja połączenia

Deinicjalizacja połączenia



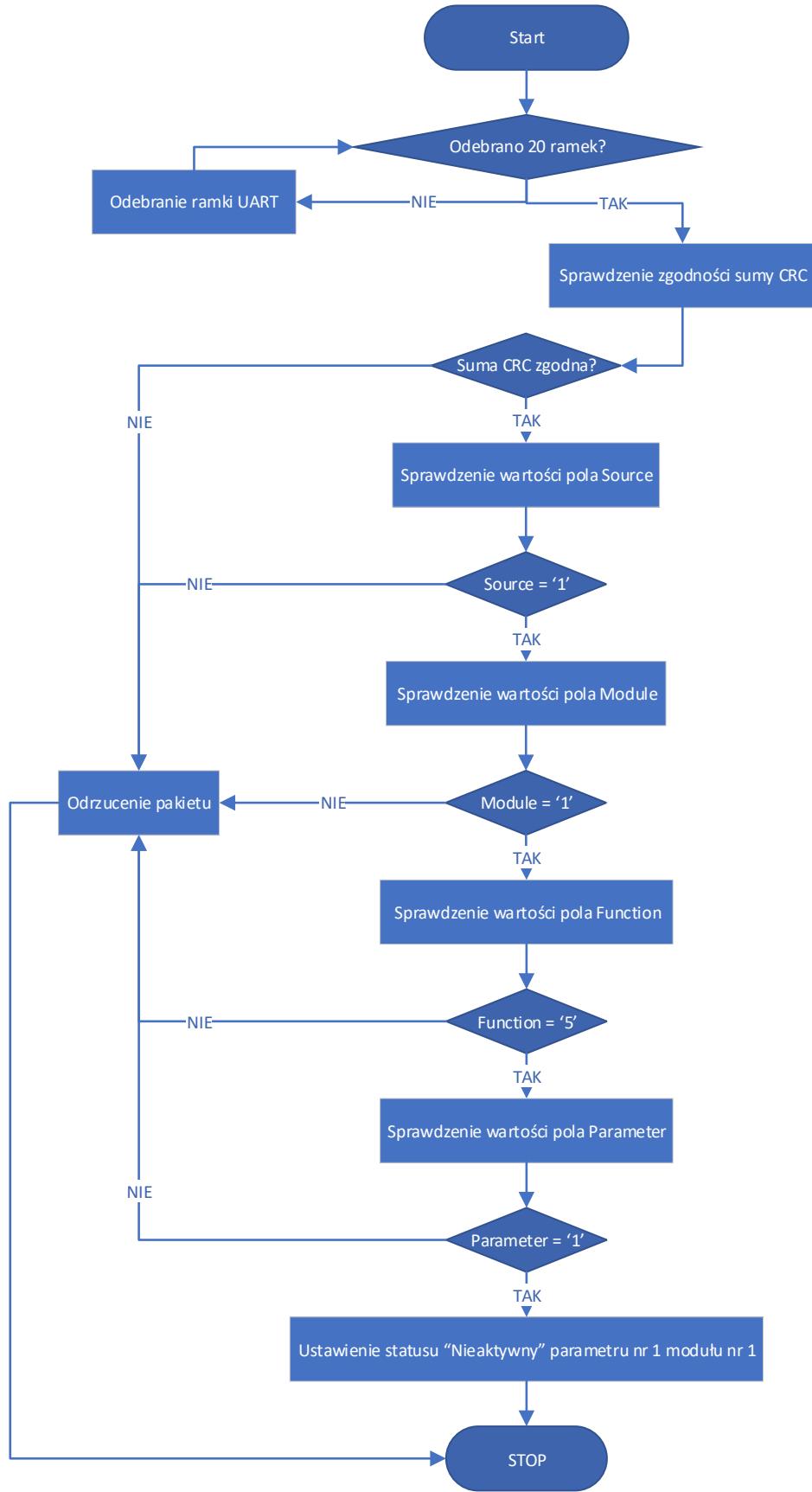
Rys. D.3 Schemat blokowy funkcjonalności Deinicjalizacja połączenia

Aktywacja parametru



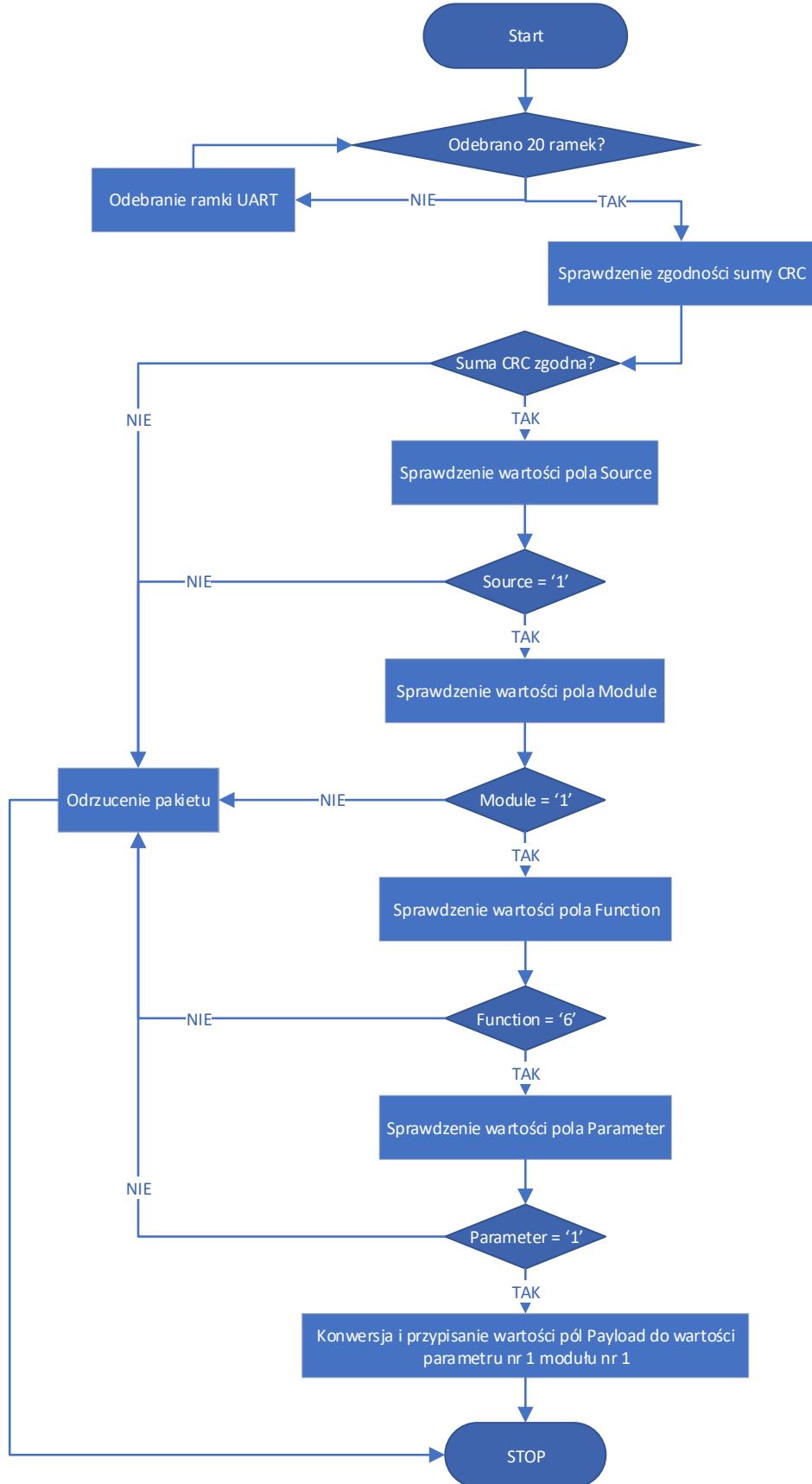
Rys. D.4 Schemat blokowy funkcjonalności Aktywacja parametru

Deaktywacja parametru



Rys. D.5 Schemat blokowy funkcjonalności Deaktywacja parametru

Ustawianie wartości parametru

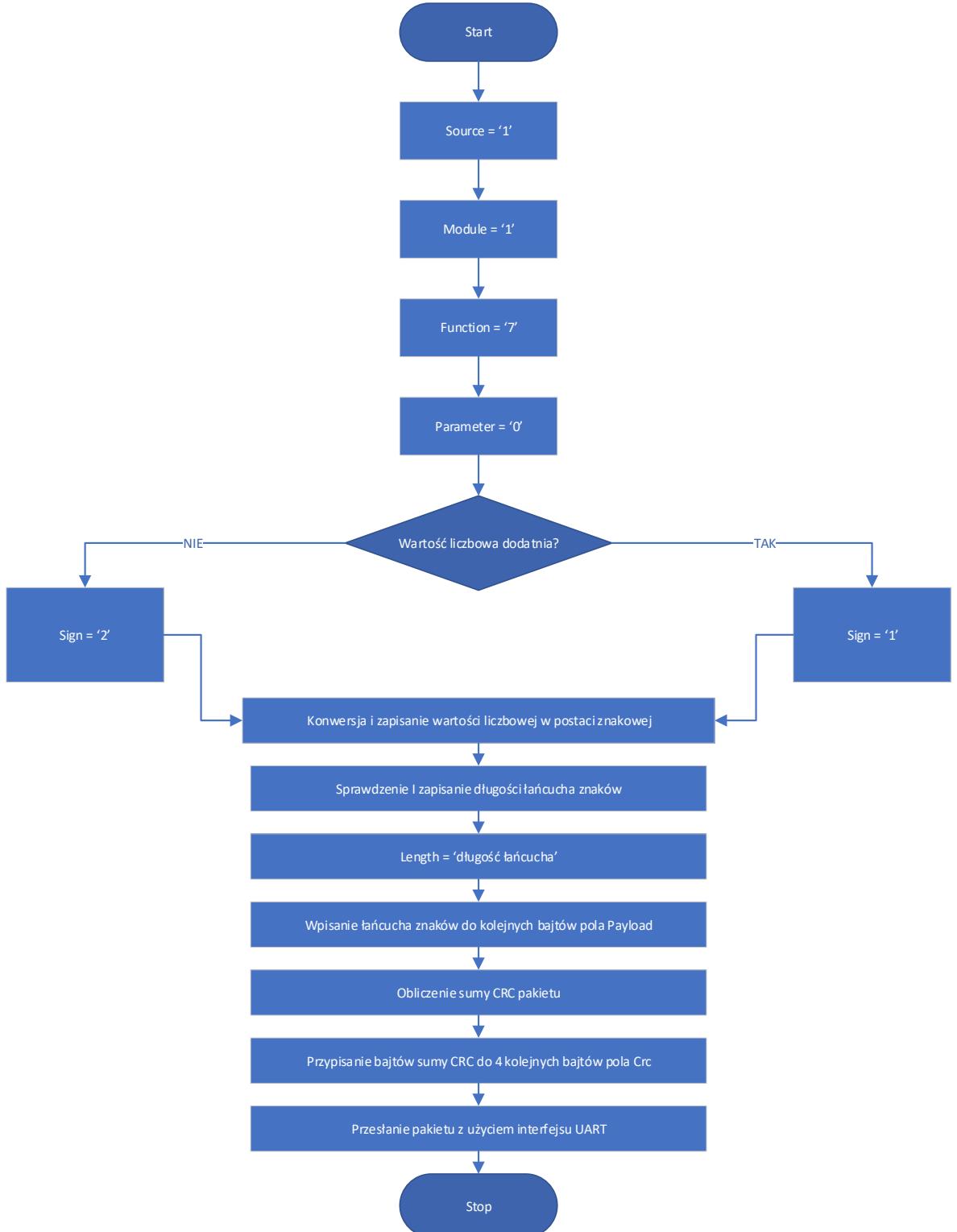


Rys. D.6 Schemat blokowy funkcjonalności Ustawianie wartości parametru

Ustawianie dolnego zakresu grafu

Warunki początkowe:

- Określona jest liczba całkowita składająca się z maksymalnie 10 znaków mająca stanowić wartość dolnego zakresu grafu

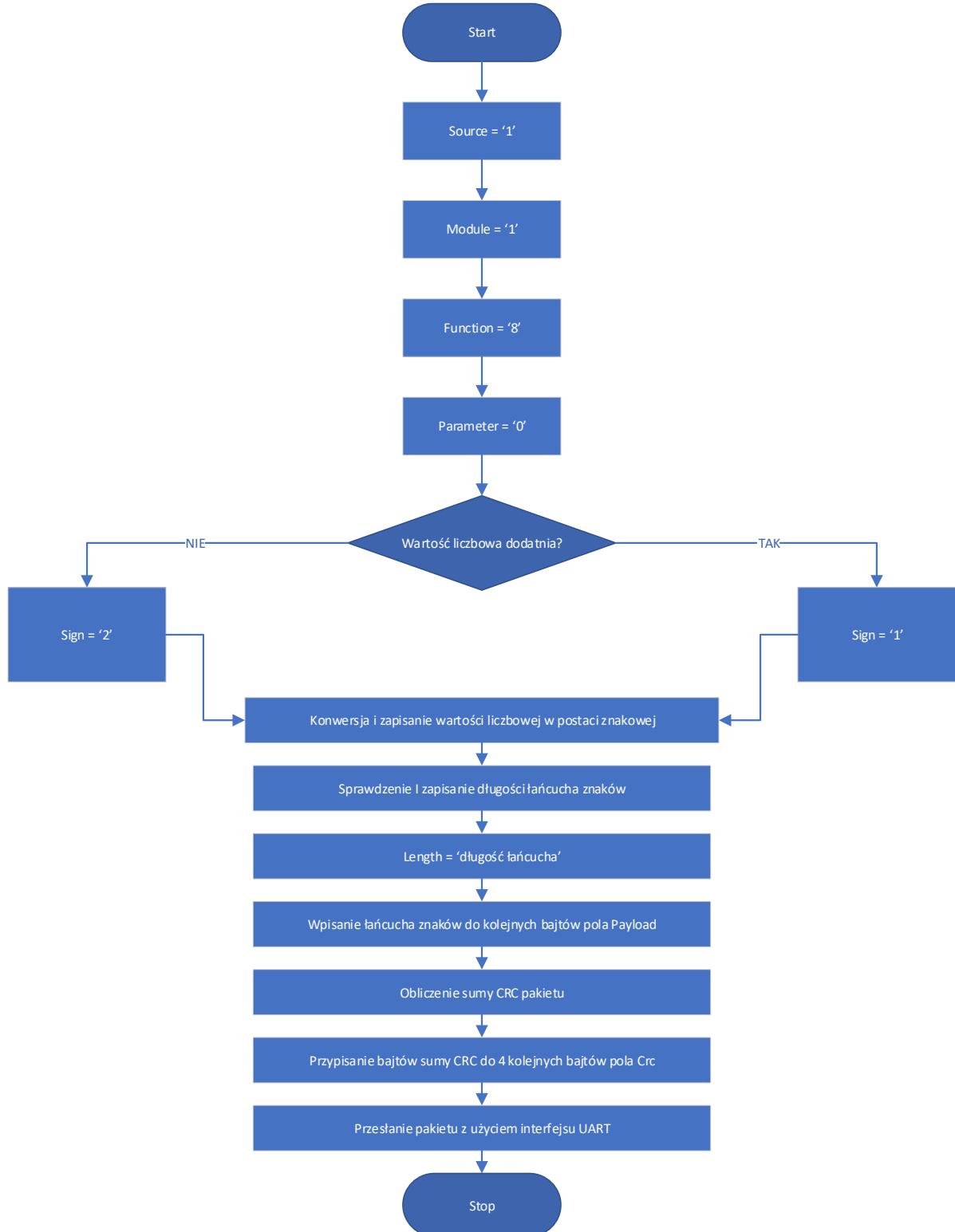


Rys. D.7 Schemat blokowy funkcjonalności Ustawianie dolnego zakresu grafu

Ustawianie górnego zakresu grafu

Warunki początkowe:

- Określona jest liczba całkowita składająca się z maksymalnie 10 znaków mająca stanowić wartość górnego zakresu grafu

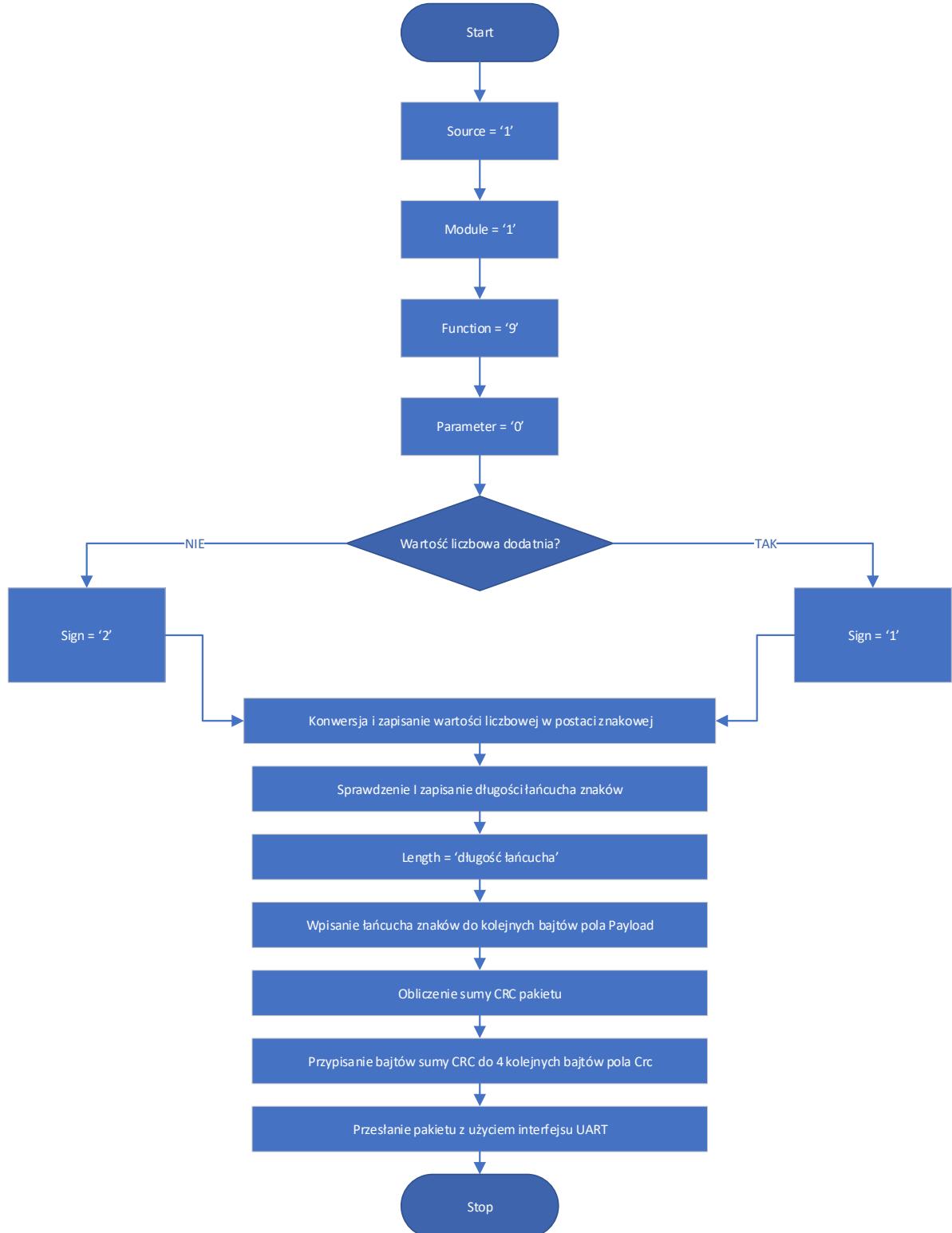


Rys. D.8 Schemat blokowy funkcjonalności Ustawianie górnego zakresu grafu

Ustawianie zakresu czasowego grafu

Warunki początkowe:

- Określona jest liczba całkowita w zakresie od 360 do 3600 będąca wielokrotnością liczby 360 mająca stanowić zakres czasowy grafu



Rys. D.9 Schemat blokowy funkcjonalności Ustawianie zakresu czasowego grafu