Gliwice, 9.01.2021

Sprawozdanie z projektu PBL

Rejestracja i analiza ruchu przemysłowej sieci Ethernet

Katedra Systemów Rozproszonych i Urządzeń Informatyki

Wydział Automatyki Elektroniki i Informatyki

Skład zespołu:

Jonatan Chrobak

Adam Puchała

Witold Smaga

Jarosław Tumula

Kierownictwo projektem:

dr inż. Jacek Stój

Projekt zrealizowany dzięki dofinansowaniu przyznanemu w ramach I konkursu finansowania projektów studenckich kół naukowych w ramach programu „Inicjatywa Doskonałości – Uczelnia Badawcza”

Spis treści

1. Temat i zakres projektu

W roku akademickim 2020/21 członkowie Międzywydziałowego Koła Naukowego Przemysłowych Zastosowań Informatyki „Industrum”, istniejącego w ramach Politechniki Śląskiej, pracowali nad projektem dotyczącym praktycznego wykorzystania urządzenia TAP (ang. Test Access Point) w postaci karty NetFPGA do analizy i diagnostyki ethernetowych sieci przemysłowych. Zakres projektu objął wykorzystanie dotychczasowych osiągnięć członków SKN “Industrum” na polu wykorzystania karty FPGA do przechwytywania ramek sieciowych, ich rozszerzenie i zebranie w bardziej przystępną dla użytkownika końcowego formę aplikacji komputerowej i komputerowej stacji badawczej.

2. Podstawa opracowania projektu

Projekt ten jest kontynuacją i rozszerzeniem prac realizowanych w ramach koła Industrum oraz pracy magisterskiej o tytule “Sprzętowy system analizy i monitorowania ruchu sieciowego” autorstwa mgr. inż. Łukasza Nowoka. Był on realizowany dzięki I konkursowi finansowania projektów studenckich kół naukowych w ramach programu Inicjatywa Doskonałości - Uczelnia Badawcza. Temat został zaproponowany przez kierownika projektu dr. Inż. Jacka Stója w ramach zaliczenia przedmiotu Projektowanie Systemów Cybernetyczno-Fizycznych i był on kontynuowany przez zespół poprzez nauczanie PBL (Project Based Learning).

3. Motywacja realizacji projektu

Będąc u zarania czwartej rewolucji przemysłowej, komputerowe systemy przemysłowe wymagają nowych rozwiązań pozwalających na zautomatyzowanie procesów analitycznych, diagnostycznych i monitorujących. Nadzór ludzki nad stanami systemów powinien pełnić coraz mniejszą rolę. Z trybu manualnego powinien przechodzić w tryb automatyczny realizowany poprzez systemy wbudowane i komputerowe. Istnieje zatem potrzeba nie tylko opracowania nowych rozwiązań dla monitorowania i diagnostyki przemysłu, ale również ich implementacji w już istniejących systemach przy minimalizacji kosztów i ingerencji w strukturę systemów.

Cechy układów FPGA są mocną podstawą do badań nad możliwością ich wykorzystania w ww. kontekście. Układy te charakteryzują się znaczną szybkością w przetwarzaniu danych co pozwala na umieszczanie ich wewnątrz systemowych kanałów komunikacyjnych bez wprowadzania znaczących dla systemu opóźnień. Z kolei wykorzystanie danych uzyskanych dzięki

FPGA przez odpowiednio zaprojektowaną aplikację może pozwolić na wygodne otwarcie tychże danych na komputerową analizę w czasie rzeczywistym.

4. Założenia projektowe

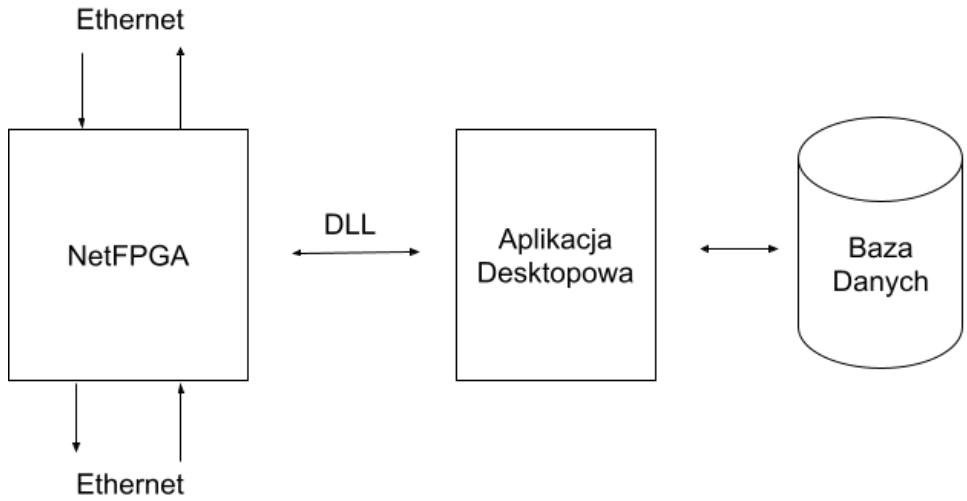
Założeniem projektu było zaprojektowanie działającego rozwiązania pozwalającego użytkownikowi przechwytywanie danych użytecznych z przychodzących w czasie rzeczywistym ramek wysyłanych przez sterownik lub inne urządzenie zewnętrzne, bez konieczności z korzystania z aplikacji firm trzecich. Dodatkowo aplikacja powinna mieć możliwość zapisywania danych w lokalnej lub zewnętrznej bazie danych oraz przełączania się pomiędzy filtracją sprzętową z użyciem FPGA i programową. Projekt ten skupia się przede wszystkim na wprowadzaniu nowych funkcjonalności do obecnego systemu, więc przystępność dla użytkownika była kwestią drugoplanową.

5. Rozwiązanie

W rozdziale tym opisano zaimplementowane w ramach projektu oprogramowanie. Omówiono architekturę systemu oraz opis jego najważniejszych części. Przedstawiono również sposób działania układu FPGA na karcie netFPGA oraz sposoby filtracji.

5.1 Architektura

Rozwiązanie składa się z trzech komponentów. Uproszczony schemat architektury systemu przedstawia rysunek 5.1.



Rys 5.1 *Architektura systemu Ethernet Tap*

W systemie można wyróżnić dwa najważniejsze komponenty:

- **NetFPGA –** karta PCIE z układem FPGA nasłuchującym ruch sieciowy w sieci Ethernet. Układ umożliwia również filtrację przechwyconych ramek Ethernet. Komunikuje się z oprogramowaniem za pośrednictwem biblioteki DLL.
- **Aplikacja desktopowa –** program komputerowy umożliwiający komunikację z kartą netFPGA. Odpowiada za wyodrębnianie i filtrację danych użytecznych pochodzących z przechwyconych ramek. Umożliwia wysyłanie uzyskanych danych do zewnętrznej bazy danych bądź zapis ramek zawierających dane użyteczne do pliku pcap. Udostępnia również interfejs użytkownika służący do konfigurowania parametrów przechwytywania i filtrów sprzętowych.

5.2 NetFPGA

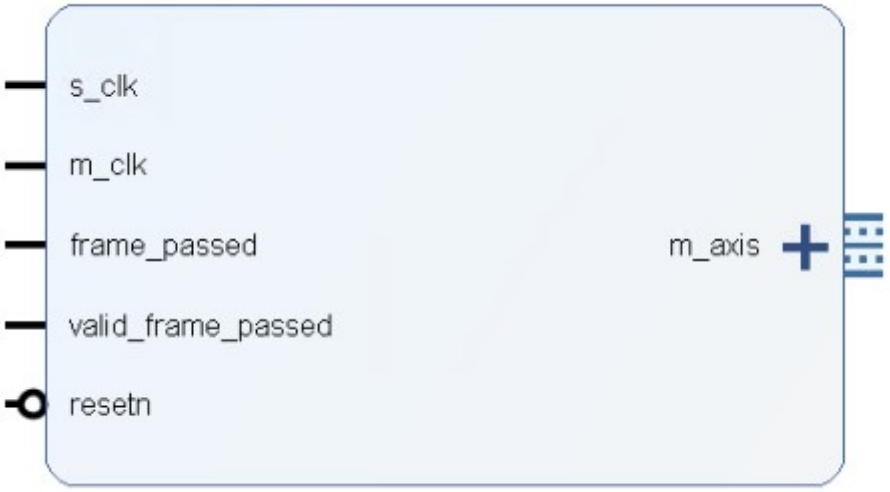
Urządzenie NetFPGA włączane jest pomiędzy dwa urządzenia sieciowe komunikujące się ze sobą. Jego zadaniem jest przechwytywanie ramek sieciowych w celu umożliwienia ich przechowywania oraz dalszej analizy.

Urządzenie wpinane jest do portu PCIe komputera. Daje to możliwość komunikacji z utworzoną w ramach projektu aplikacją. Za pomocą połączenia możliwy jest odczyt oraz zapis danych do urządzenia.

Karta posiada cztery porty Ethernet. Daje to możliwość wpięcia czterech urządzeń. Z poziomu urządzenia możliwe jest stworzenie połączenia między portami pierwszym i drugim oraz trzecim i czwartym w celu ustanowienia komunikacji pomiędzy wybranymi stacjami. Działanie portów zarządzane jest przez układ FPGA, w którym zaimplementowane zostały bloki mdio ustawiający tryb i parametry pracy portów oraz blok kontrolujący połączenia między portami i przechwytywanie przepływających danych.

W układzie FPGA zaimplementowane zostały także cztery bloki przechwytujące i filtrujące przychodzące ramki Ethernet, odpowiednio po jednym bloku na każdy port. Nastawy filtrów odbierane są z portu PCIe i wpisywane do rejestrów w odpowiednich blokach filtrujących. Jeśli ramka poprawnie przejdzie filtrację, jest ona dodawana do kolejki FIFO, z której może następnie zostać odczytana przez aplikację desktopową.

W ramach projektu PBL w układzie zaimplementowana została nowa funkcjonalność. Dodana została możliwość zliczania ramek przez kartę NetFPGA. Blok zliczający (**rys 5.2**) jest ściśle połączony z blokiem filtrującym, ponieważ jego zadaniem jest nie tylko zliczanie przechwyconych ramek, lecz także zliczanie ilości ramek, które pomyślnie przeszły filtrację.



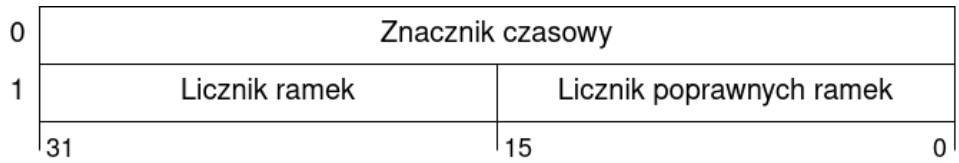
Rys 5.2 Blok zliczający ramki

W module filtrującym zadeklarowane zostały dwa dodatkowe sygnały. Pierwszy sygnał mówi o pojawieniu się nowej ramki, natomiast drugi informuje o pomyślnym przejściu filtracji przez ramkę. Utworzony został także nowy proces, który zmienia stan sygnałów. Stan sygnałów zmieniany jest zaraz przed rozpoczęciem procesu filtracji kolejnej ramki, w momencie, w którym wiadomo już, że obecna ramka przeszła filtrację, lecz nie zostały jeszcze zresetowane flagi.

Tak jak w przypadku bloków filtrujących, stworzone zostały cztery moduły zliczające, co odpowiada ilości portów Ethernet. Wejścia bloków zliczających stanowią dwa nowe sygnały wychodzące z modułu filtracji, dwa wejścia zegara (o częstotliwości 62,5 MHz oraz 125 MHz) oraz aktywny stanem niskim sygnał reset. Moduł taktowany jest wolniejszym zegarem, natomiast szybszy zegar służy do odczytu danych z kolejki FIFO. Dane odczytywane są przy użyciu interfejsu AXIS4.

Sygnały informujące o obecności nowych ramek próbkowane są na narastającym zboczu zegara. Jeśli w tym momencie sygnał informujący o pojawieniu się nowej ramki znajduje się w stanie wysokim, licznik ramek zwiększany jest o jeden. Tak samo dzieje się w przypadku sygnału informującego o pomyślnym przejściu przez ramkę filtracji, inkrementowany jednak jest licznik ramek, które pomyślnie przeszły filtrację.

Dodatkowo na każdym narastającym zboczu zegara zwiększany jest znacznik czasowy. Znacznik czasowy także dodawany jest do kolejki FIFO razem z licznikami. W pierwszej kolejności dodawany do kolejki zostaje 32-bitowy znacznik czasowy, a następnie dwa 16-bitowe liczniki. Struktura danych przedstawiona jest na rysunku 5.2. Pozwala to na dokładne określenie ilości ramek otrzymanych w danym okresie czasu, niezależnie od tego, w jakim przedziale prowadzone było zliczanie.



Rys 5.3 kolejność danych w kolejce FIFO

Dane do kolejki FIFO dodawane są periodycznie co 1,005 ms. W momencie dodawania danych do kolejki liczniki ramek są resetowane. Aby dowiedzieć się jaka ilość ramek odebrana została w danym czasie, wystarczy dodać do siebie odczytane z kolejki stany liczników oraz odjąć stan znacznika czasowego z początku eksperymentu od stanu znacznika z jego końca.

5.3 Aplikacja Desktopowa

Program okienkowy stanowi rozszerzenie oprogramowania konsolowego stworzonego w ramach pracy magisterskiej. Program konsolowy został przekształcony na bibliotekę DLL dzięki czemu jego funkcjonalności mogą być wykorzystywane przez inne aplikacje.

Aplikacja desktopowa została zaimplementowana w technologii .Net WPF oraz języku C#. Architektura aplikacji została oparta o wzorzec MVVM (ang. Model-View-View-Model). W kodzie programu wyodrębniono klasy odpowiadające za komunikację z urządzeniem za pośrednictwem biblioteki DLL, przetwarzanie przechwyconych ramek oraz eksport danych użytecznych do zewnętrznej bazy.

W bibliotece DLL została zaimplementowana funkcja przyjmująca callback - wskaźnik na funkcję wywoływaną w aplikacji desktopowej. Po przechwyceniu ramki wywoływana funkcja, która rozsyła ją do wszystkich obserwatorów (Wzorzec publikowanie-subskrybowanie). W ten sposób poszczególne części aplikacji desktopowej reagują na przechwycone dane – inkrementuje się licznik przechwyconych ramek, ramka jest analizowana w celu wyłuskania danych użytecznych.

Po uruchomieniu programu w pierwszej kolejności odczytywana jest konfiguracja z pliku config.xml. Odczytywane są wartości takie jak numer portu, na którym nasłuchuje urządzenie oraz rozmiar bufora, w którym przechowywane są przechwycone ramki. Numer portu oraz rozmiar bufora ustawiane są jednokrotnie w cyklu życia aplikacji. Proces komunikacji z urządzeniem oraz nasłuchiwanie ramek odbywa się w oddzielnym wątku aplikacji. Przechwytywanie uruchamiane jest z domyślnymi ustawieniami – tryb auto negocjacji przepustowości sieci oraz wyłączony EEE. Program umożliwia nasłuchiwanie na jednym porcie, który określany jest w ustawieniach przechwytywania.

5.3.1 Filtracja programowa

Filtracja w aplikacji desktopowej została zrealizowana poprzez zastosowanie równości poszczególnych bitów wraz ze wskazaniem bitów, w których zapisane są dane użyteczne. Możliwe jest ustalenie wartości progu lub przedziału, dopiero po których przekroczeniu ramka zostanie zarejestrowana. Na chwilę obecną filtracja obsługuje ramki typu Ethernet i EtherCAT. Filtry zapisywane są w formacie XML. Przykładowy filtr znajduje się na rysunku 5.2.

```
<Filter>
    <Condition>And([22]=1e,Or([23]=40,[23]=41))</Condition>
    <Targets>
        <Target>
            <Id>0</Id>
            <Bytes>70,71</Bytes>
            <Type>string</Type>
            <Name>temperature</Name>
            <Threshold>
                <Type>gt</Type>
                <Value>10000</Value>
            </Threshold>
        </Target>
    </Targets>
</Filter>
```

Rys 5.2 *Przykładowy filtr*

Każdy filtr musi być zawarty w tagu Filter i musi zawierać tagi Condition oraz Targets z co najmniej jednym Targetem. W tagu Condition zawieramy warunek jaki ramka ma spełniać, aby została uznana za użyteczną. Warunek ten zapisany jako kombinacja operacji logicznych AND i OR oraz założeń równości poszczególnych bajtów ramki (przykładowo warunek *And([0]=12, [1]=1e)* zakłada, iż pierwszy bajt ramki będzie równy 0x12 i drugi bajt będzie równy 0x1e). Dla danego filtru może być tylko jeden warunek, ale poszczególne operacje logiczne można dowolnie zagnieżdżać.

W tagu Targets zawarte są informacje co do danych użytecznych dla przefiltrowanych ramek. Każdy tag Target zawiera informacje co do jednej zmiennej, a jeden filtr może mieć dowolną liczbę tagów Target (minimum jeden). Każdy tag Target musi zawierać tagi Id, Name, Type i Bytes. Tagi te zawierają informacje użyteczne dla dalszego eksportu danych takie jak id zmiennej, nazwa i typ zmiennej oraz poszczególne bajty, na których zapisana jest zmienna. Aktualnie obłsugiwane typy danych to *boolean*, *int16*, *int32*, *int64*, *string*, *float*, *double* i *byte array*. Dodatkowo opcjonalnym tagiem jest Threshold, który zawiera informacje o progu lub przedziale który zmienna ma przekroczyć, aby została zarejestrowana. Jeden Target może zawierać maksymalnie jeden Threshold.

Każdy Threshold musi zawierać tag Type oraz Value i w przypadku niektórych typów również tag Value2. Tag Type określa rodzaj progu, a tagi Value i Value2 zawierają informację o wartościach granicznych progu. Progi obsługiwane są tylko dla typów numerycznych (int16, int32, int64, float i double) i w przypadku innego typu zostaną zignorowane. Aktualnie obsługiwane typy progów to:

- gt - wartość zmiennej musi przekroczyć wartość Value
- ge - wartość zmiennej musi przekroczyć lub być równa wartości Value
- lt - wartość zmiennej musi być poniżej wartości Value
- le - wartość zmiennej musi być równa lub poniżej wartość Value
- InOpen – wartość zmiennej musi zawierać się w przedziale otwartym (Value;Value2)
- InClosed – wartość zmiennej musi zawierać się w przedziale domkniętym <Value;Value2>
- OutOpen – wartość zmiennej musi zawierać się w sumie przedziałów obustronnie otwartych (-∞;Value) ∪ (Value2;∞)
- OutClosed – wartość zmiennej musi zawierać się w sumie przedziałów jednostronnie domkniętych (-∞;Value> ∪ <Value2;∞)

5.3.2 Filtracja sprzętowa

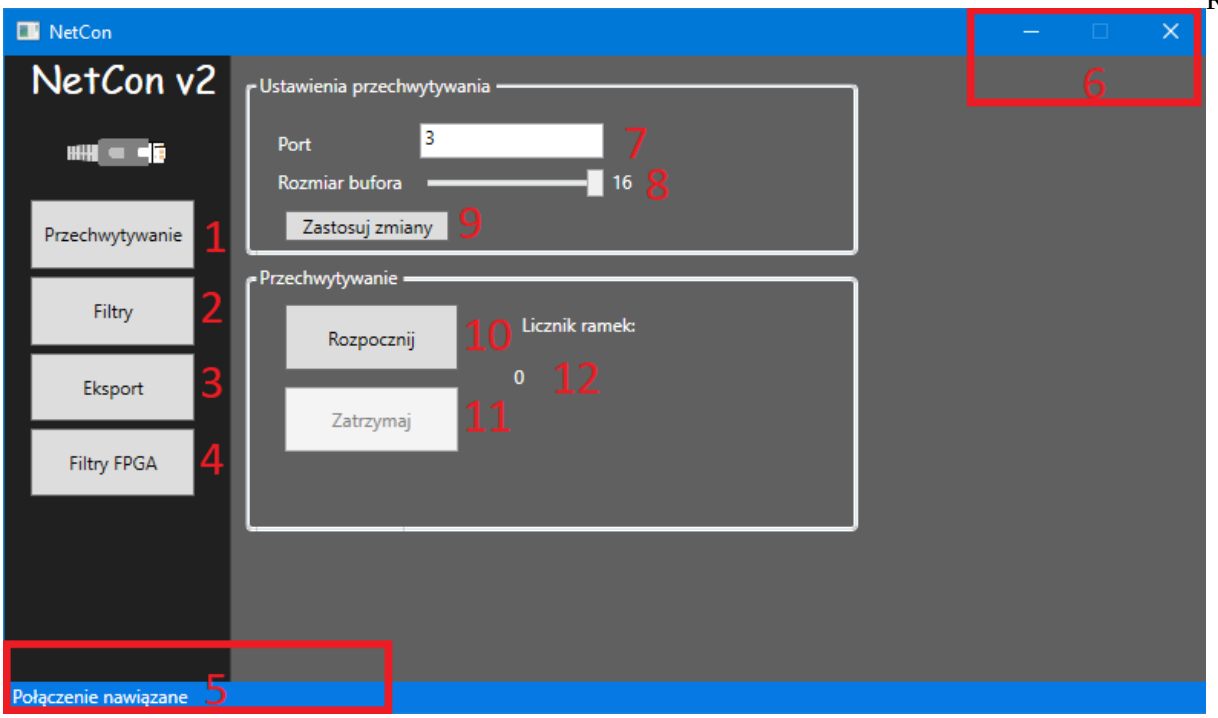
Aplikacja umożliwia ustawianie filtrów sprzętowych poprzez bibliotekę DLL niezależnie od tego czy proces przechwytywania działa. Minimalny rozmiar ramki określa się dla wszystkich bloków filtracyjnych jednocześnie. Filtry definiowane są w sposób jak na przykładzie:

- 1 123 125 – wartość bajtu ramki na pozycji 1 ma być w przedziale <123,125>
- 3 40 – wartość bajtu ramki na pozycji 3 ma być równa 40

Filtracja sprzętowa nie została zmieniona i działa na tej samej zasadzie co w pierwotnej aplikacji konsolowej. Więcej informacji odnośnie definiowania filtrów można znaleźć w pracy magisterskiej “Sprzętowy system monitorowania i analizy ruchu sieciowego” w podrozdziale 3.3.4.

6. Obsługa aplikacji

Okno aplikacji przedstawia rysunek 6.1.



Rys. 6.1. *Okno główne aplikacji desktopowej*

Po lewej stronie okna znajduje się panel nawigacji pozwalający przechodzić po poszczególnych częściach aplikacji:

- Przechwytywanie (nr. 1, Rys. 6.1.) – domyślna strona widoczna po uruchomieniu aplikacji. Pozwala konfigurować parametry przechwytywania oraz aktywować/dezaktywować przechwytywanie;
- Filtry (nr. 2, Rys. 6.1.) – strona odpowiedzialna za ustawianie filtrów programowych;
- Eksport (nr. 3, Rys 6.1.) – część aplikacji pozwalająca skonfigurować eksport danych do zewnętrznej bazy oraz pliki pcap;
- Filtry FPGA (nr. 4, Rys 6.1.) – strona, która pozwala ustawić filtry sprzętowe.

Informacje o błędach oraz innych zdarzeniach, które nastąpiły w aplikacji widoczne są na pasku oznaczonym numerem 5 na rysunku 6.1. W przypadku wyświetlania błędów pasek zmieni swój kolor na pomarańczowy.

W celu określenia portu należy wpisać żądaną wartość w polu tekstowym oznaczonym numerem 7 na rysunku 6.1. Rozmiar bufora regulowany jest suwakiem oznaczonym numerem 8 na

rysunku 6.1. W celu zapisania zmian należy kliknąć w przycisk oznaczony numerem 9. Zmiany zostaną zatwierdzone po ponownym uruchomieniu aplikacji.

Aby rozpocząć przechwytywanie należy kliknąć w przycisk oznaczony numerem 10 na rysunku 6.1. Do zatrzymywania przechwytywania służy przycisk oznaczony numerem 11. Liczba aktualnie przechwyconych ramek widoczna jest w miejscu oznaczonym numerem 12 na rysunku.

W celu zamknięcia lub zminimalizowania aplikacji należy skorzystać z przycisków oznaczonych numerem 6 na rysunku 6.1.

Okno ustawień eksportu zostało zaprezentowane na rysunku 6.2.

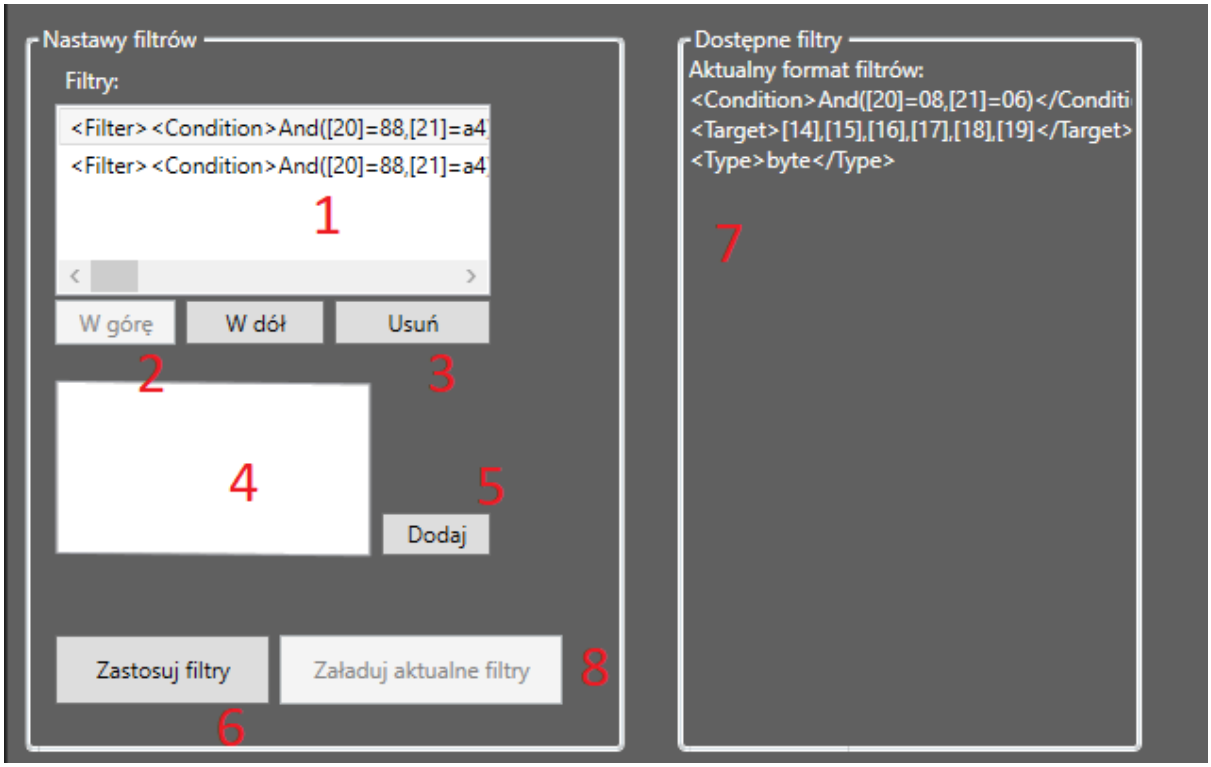
Rys.

6.2. *Okno eksportu*

Aby aktywować eksport do bazy danych należy zaznaczyć pole oznaczone numerem 1 na rysunku 6.2

W celu nawiązania połączenia należy skonfigurować parametry połączenia. Adres (nr. 2, Rys. 6.2) oraz nazwę bazy danych (nr. 3, Rys.6.2.). Aby przetestować połączenie dla podanych parametrów należy kliknąć w przycisk oznaczony numerem 4 na rysunku. Wynik testu zostanie wyświetlony w oknie dialogowym. W celu zapisania zmian należy kliknać przycisk oznaczony nr. 5 na rysunku.

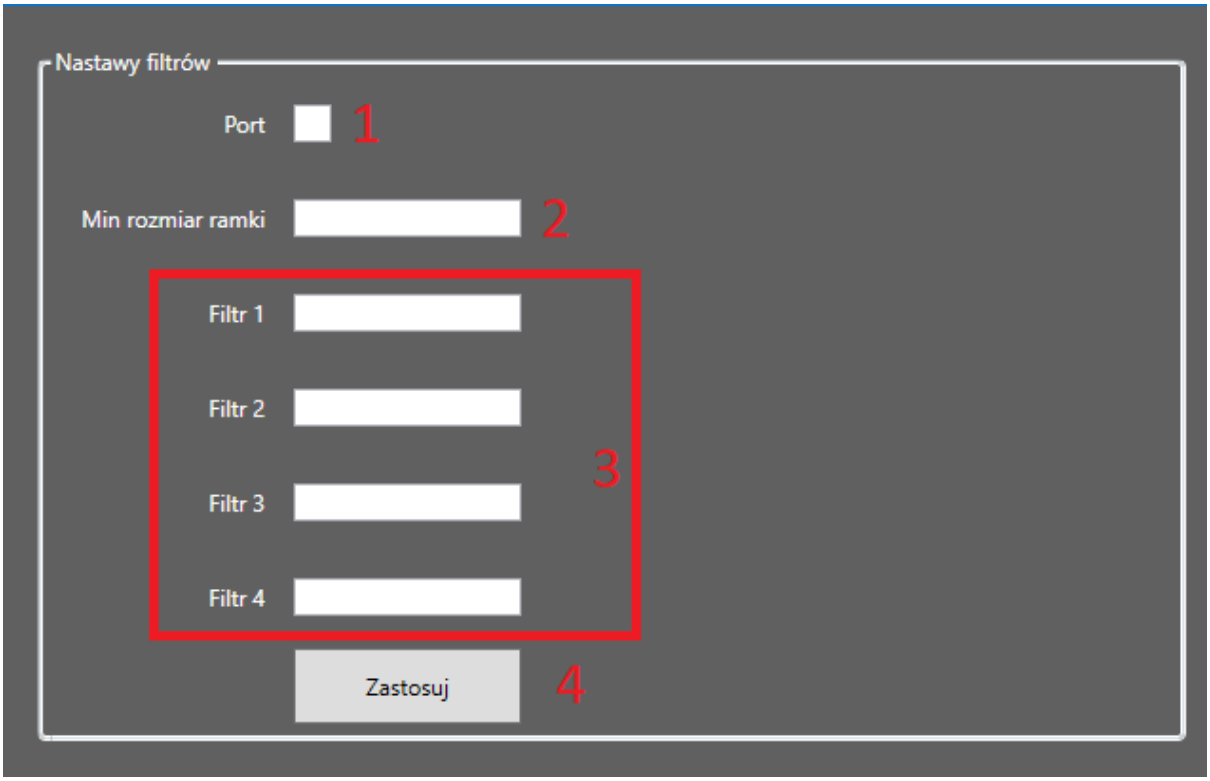
Program umożliwia również eksport przechwyconych ramek do pliku pcap. Po zaznaczeniu pola oznaczonego numerem 6 na rysunku 6.2 przechwycone ramki będą zapisywane w pliku “myFrames.pcap” w katalogu z plikiem wykonywalnym aplikacji.



Rys. 6.3. *Okno konfiguracji filtrów*

W celu konfiguracji filtrów należy przejść do okna konfiguracji filtrów. Lista aktualnie dostępnych filtrów widoczna jest w miejscu oznaczonym numerem 1 na rysunku 6.3. Aby dodać filtr należy wpisać jego definicję w odpowiednim formacie w polu oznaczonym numerem 4 na rysunku oraz kliknąć przycisk oznaczony numerem 5 na rysunku. Istnieje możliwość zmiany pozycji zaznaczonego filtra w zestawie filtrów przy pomocy przycisków oznaczonych numerem 2 na rysunku. Aby usunąć zaznaczony filtr należy kliknąć w przycisk oznaczony numerem 3 na rysunku. W celu zastosowania zmian należy kliknąć w przycisk oznaczony numerem 6 na rysunku. Istnieje możliwość załadowania ostatnich zapisanych filtrów przy pomocy przycisku oznaczonego numerem 8 na rysunku. W miejscu oznaczonym numerem 7 na rysunku wyświetlany jest poprawny format filtrów.

Aby ustawić filtry sprzętowe należy przejść do strony “Filtry FPGA”. Strona konfiguracji filtrów sprzętowych prezentuje się jak na rysunku 6.4.



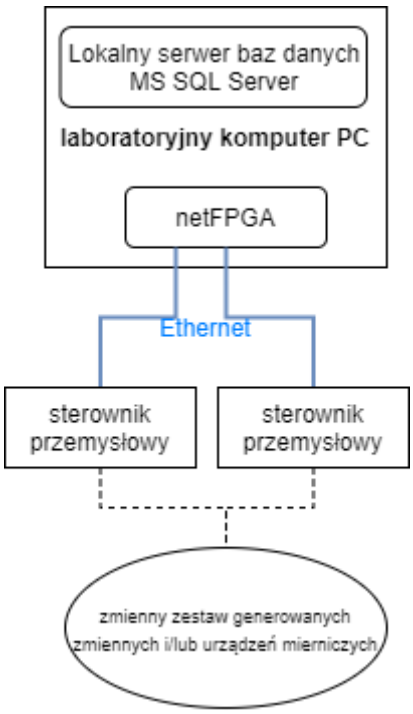
Rys 6.4. *Okno konfiguracji filtrów sprzętowych*

Aby ustawić filtry sprzętowe należy wprowadzić numer portu (nr. 1 na rysunku 6.4), minimalny rozmiar ramki (nr. 2 na rysunku) oraz przynajmniej jeden z filtrów (pola oznaczone ramką nr. 3 na rysunku). Po wprowadzeniu wymaganych danych należy kliknąć przycisk “Zastosuj” (nr. 4 na rysunku).

7. Testowanie i uruchamianie

Projekt był uruchamiany i testowany na bieżąco w trakcie rozwoju, w dwóch różnych środowiskach testowych – laboratoryjnym i wirtualnym, symulowanym.

Środowisko laboratoryjne przygotowane przez opiekunów koła znajdowało się na terenie Politechniki Śląskiej i składało się z komputera PC z systemem Windows, z podpiętą kartą netFPGA do magistrali PCIe. Do portów karty podpięte zostały sterowniki przemysłowe przesyłające między sobą przykładowe ramki EtherCAT. W celu testowania eksportu danych na komputerze uruchomiono lokalny serwer MS SQL Server:



Schemat 7.1 - stanowisko laboratoryjne wykorzystywane do testów i badań projektu

Wirtualne środowisko testowe opierało się na uruchamianiu projektu i serwera baz danych na dowolnym komputerze osobistym z systemem Windows, aby umożliwić testowanie projektu na prywatnych komputerach członków koła w przypadku braku dostępu zdalnego do prawdziwego stanowiska laboratoryjnego. W aplikacji projektowej zaimplementowana została możliwość pobierania ramek z przygotowanego pliku pcap i symulowanie wysyłania ich z dużymi prędkościami (nawet do 1000 razy na sekundę) do pozostałych komponentów aplikacji tak jakby pochodziły one z magistrali PCIe.

Prawidłowość eksportu danych do pliku pcap i do bazy danych testowana była za pomocą programów Wireshark oraz Microsoft SQL Server Management Studio:

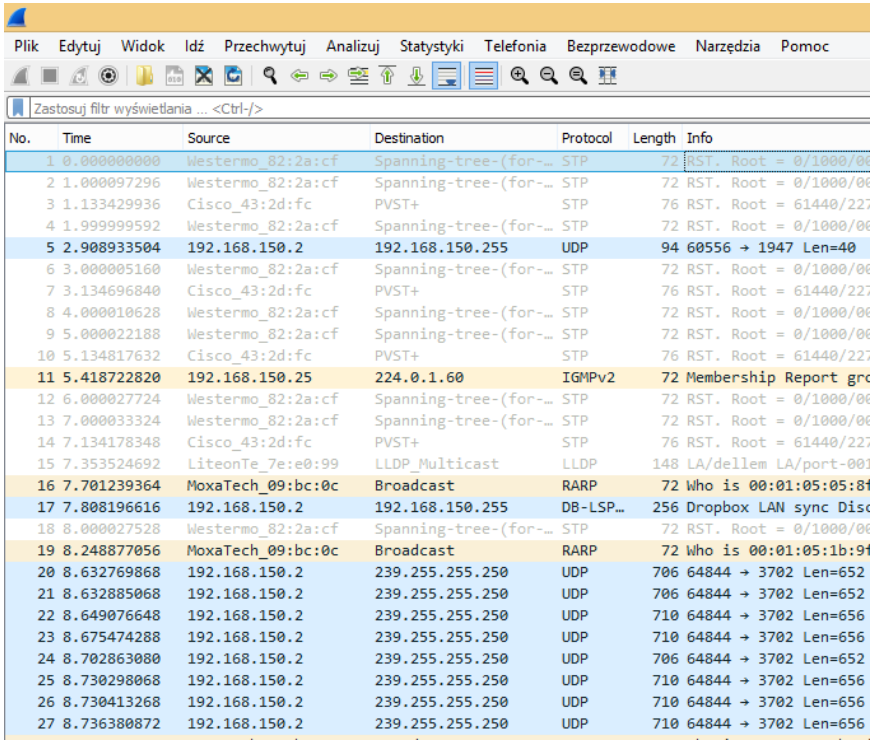


Fig. 7.2 - widok z programu Wireshark przedstawiające przechwycone i zapisane do pliku pcap ramki

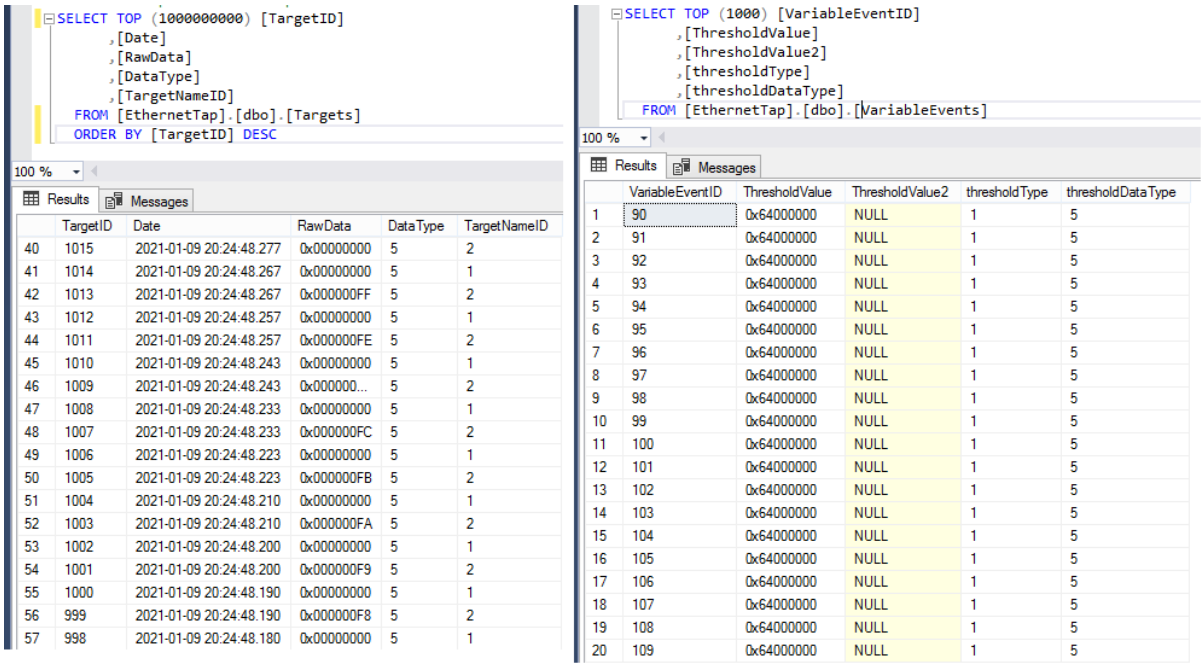


Fig. 7.3 - Widoki z programu Microsoft SQL Server Management Studio przedstawiające wpisy w bazie danych z zapisanymi wartościami dwóch zmiennych i informacjami o przekroczeniu przez zmienną nr. 2 wartości 100

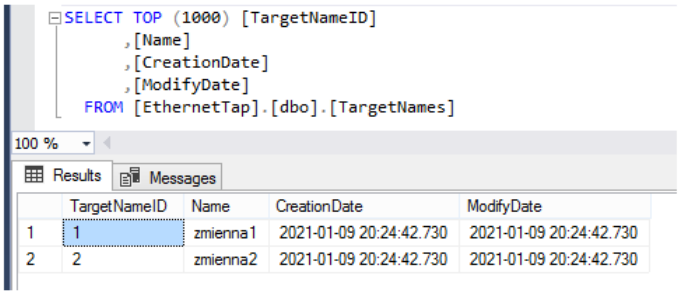


Fig. 7.4 - Widok z programu Microsoft SQL Server Management Studio przedstawiające wpisy w bazie danych z nazwami i ID zapisanych zmiennych

8. Badania na bazie wykonanego projektu

Z pomocą projektu możliwe jest zorganizowanie stacji badawczej do badań nad możliwościami układów FPGA w kontekście diagnostyki systemów przemysłowych oraz nad analizą przechwyconych i zapisanych danych przez algorytmy klasyfikacyjne. W trakcie prac nad projektem członkowie Industrum przeprowadzili badania nad generacją danych diagnostycznych przez układ FPGA oraz nad diagnostyką na podstawie klasteryzowanych danych uzyskanych przez aplikację.

8.1 Generowanie dodatkowych danych diagnostycznych na poziomie układu FPGA

Możliwe jest generowanie informacji diagnostycznych z poziomu układu FPGA, co może znacząco przyspieszyć proces reagowania na anomalie. Początkowo konieczne było wysłanie odebranych danych przez aplikację komputerową i ich dalsza analiza w celu stwierdzenia wystąpienia anomalnego zachowania, a następnie podejmowana była decyzja o przesyłaniu danych dalej.

Dzięki generacji danych diagnostycznych pominięty może zostać krok analizy danych przez aplikację desktopową, co zmniejszy czas od wystąpienia awarii do reakcji na nią, a w dalszych stadiach projektu możliwe jest zaimplementowanie w układzie FPGA nie tylko możliwości analizy danych, lecz także modułu szybkiej reakcji na nagłe zdarzenia.

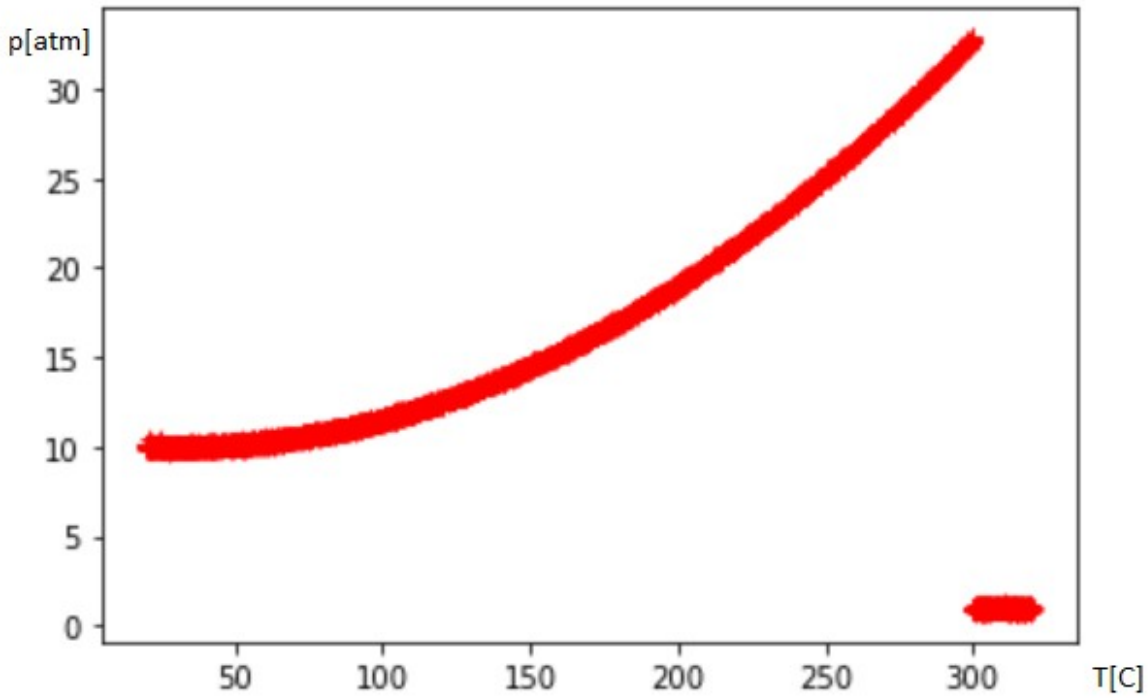
8.2 Wykrywanie i predykcja anomalii w systemie na podstawie klasteryzowanych danych uzyskanych poprzez aplikację

Odpowiednia konfiguracja filtrów w aplikacji desktopowej umożliwia de facto rejestrowanie setek lub tysięcy odczytów wybranych wartości z ramek sieciowych w czasie rzeczywistym, a także

informacji o osiąganiu wybranych wartości i zakresów granicznych. Analiza tychże danych może pozwolić na wykrywanie i predykcję stanów systemu, w tym sytuacji anormalnych i awaryjnych.

Dla zobrazowania i zbadania potencjału diagnostycznego przygotowano dwie hipotetyczne sytuacje: załóżmy system przemysłowy odczytujący i przesyłający odczyty parametrów podgrzewanego zbiornika na płyn. Moduł filtracji aplikacji desktopowej można skonfigurować do cyklicznej rejestracji wartości ciśnienia i temperatury w zbiorniku, przesyłanych jako wartości odpowiednich zmiennych systemowych wewnątrz ramek sieciowych biorących udział w komunikacji systemu:

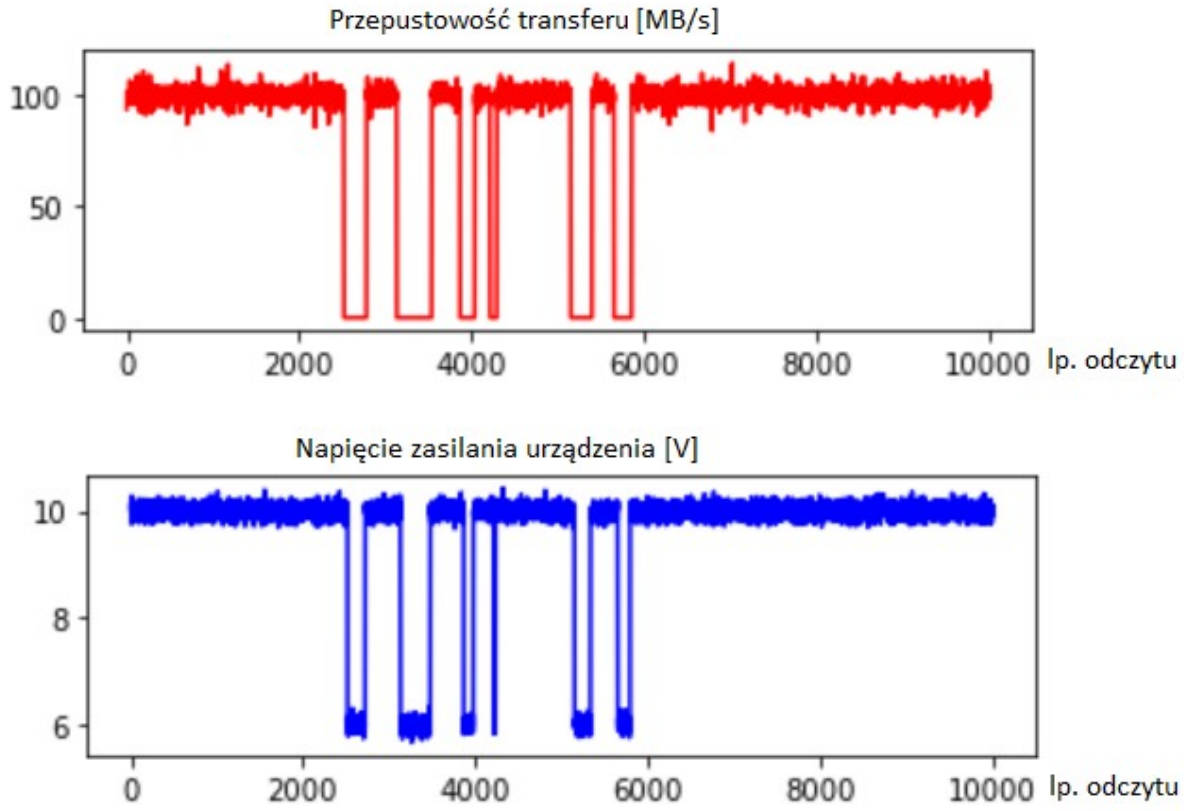
Wyk.



8.2.1 - pomiary ciśnienia i temperatury w zbiorniku płynu

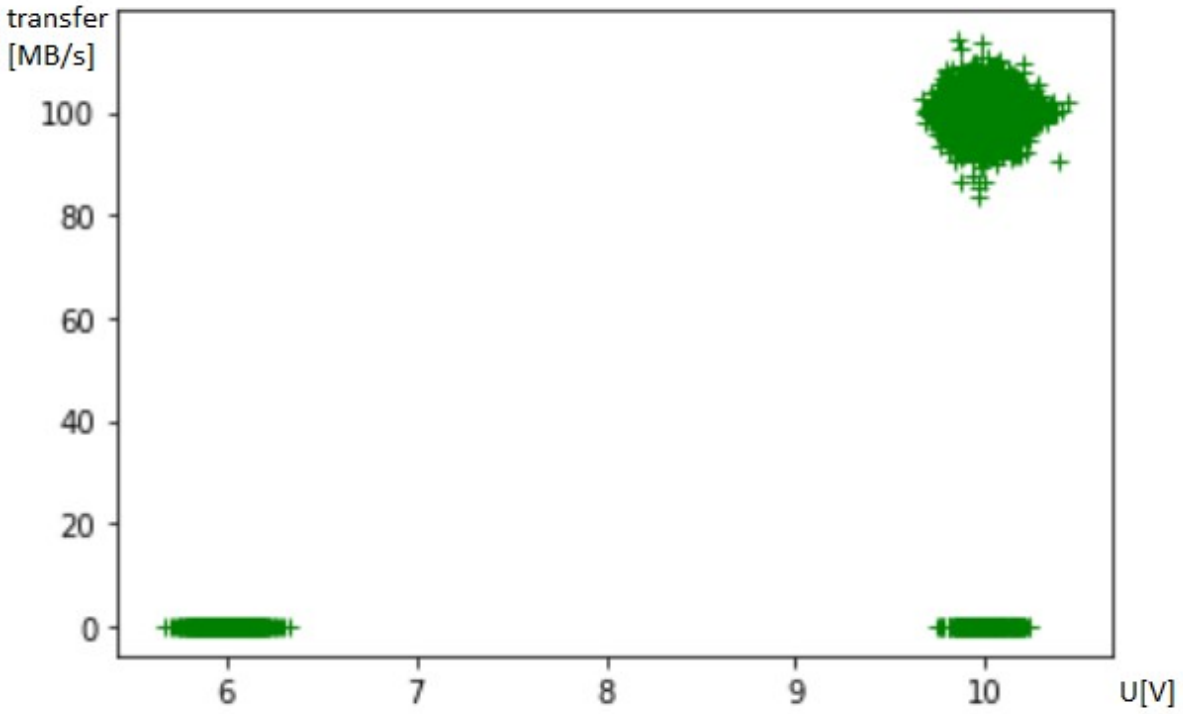
Na wykresie zobaczyć można odczyty zebrane w pary ciśnienie-temperatura. Podgrzewanie zbiornika skutkuje wzrostem ciśnienia w komorze. Po przekroczeniu pewnej wartości krytycznej następuje zdarzenie awaryjne – rozszczelnienie zbiornika. Występuje anomalia wartości ciśnienia, które nagle osiąga stałe wartości jednej atmosfery. Zajść może potrzeba wykrycia i/lub predykcji takiej awaryjnej sytuacji.

Załóżmy również pomiary parametrów pracy pewnego elementu komunikacyjnego w systemie, który jest zasilany napięciem nominalnym o wartości 10V i przeprowadza pewien transfer o mierzalnej przepustowości:



8.2.2 - pomiary prędkości transferu i napięcia zasilania urządzenia komunikacyjnego

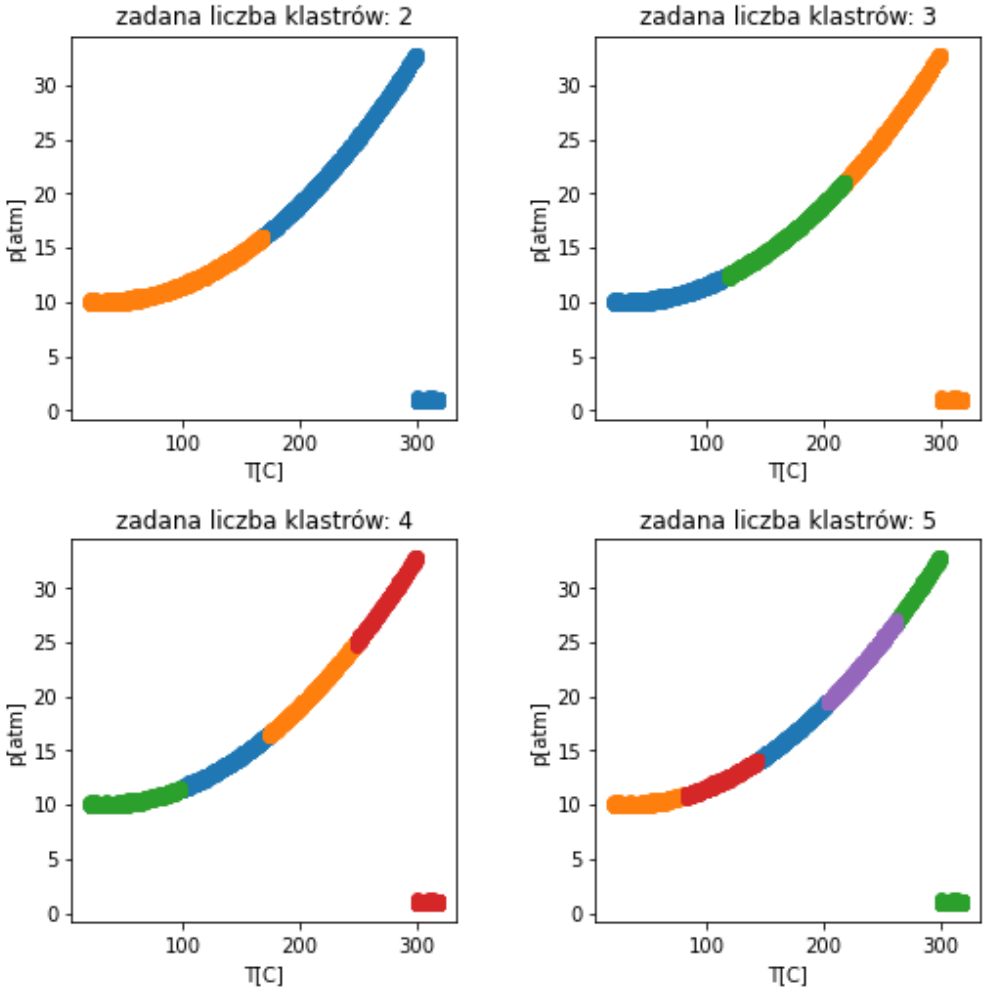
Zasilanie urządzenia jest szwankujące i niestabilne. Sporadyczne spadki napięcia poniżej nominalnych 10V skutkują zerwaniem transferu, który następnie stabilizuje się z opóźnieniem po powrocie stabilnego napięcia. Wyróżniają się trzy grupy odczytów - praca normalna przy napięciu 10V, zerwanie transferu podczas spadku napięcia i chwilowy brak transferu po powrocie napięcia, przed ponowną stabilizacją:



8.2.3 - pomiary prędkości transferu i napięcia zasilania urządzenia komunikacyjnego

Takie i podobne zestawy pomiarów n zmiennych można traktować jako zbiory danych o n atrybutach, które w celu analizy można poddać np. klasteryzacji. Powyższe zbiory zostały poddane działaniom czterech przykładowych algorytmów klasteryzacji o różnych ustawieniach hiperparametrów: K-Means, Mean-Shift, DBSCAN i grupowanie aglomeracyjne. Przykładowo dla algorytmu K-Means o różnej docelowej liczbie klastrów uzyskano następujące struktury:

KMeans - rozkład klastrów(rozdzielonych osobnymi kolorami)
w zależności od ich zadanej liczby



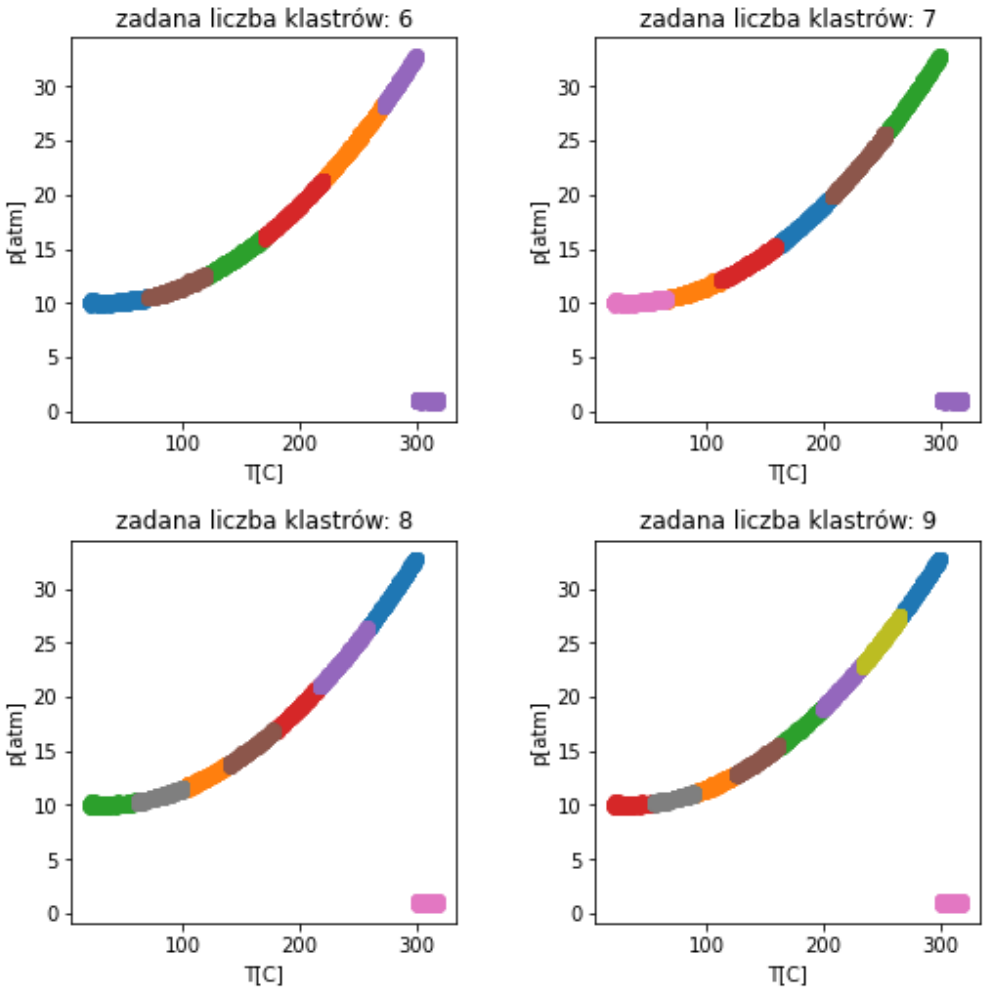
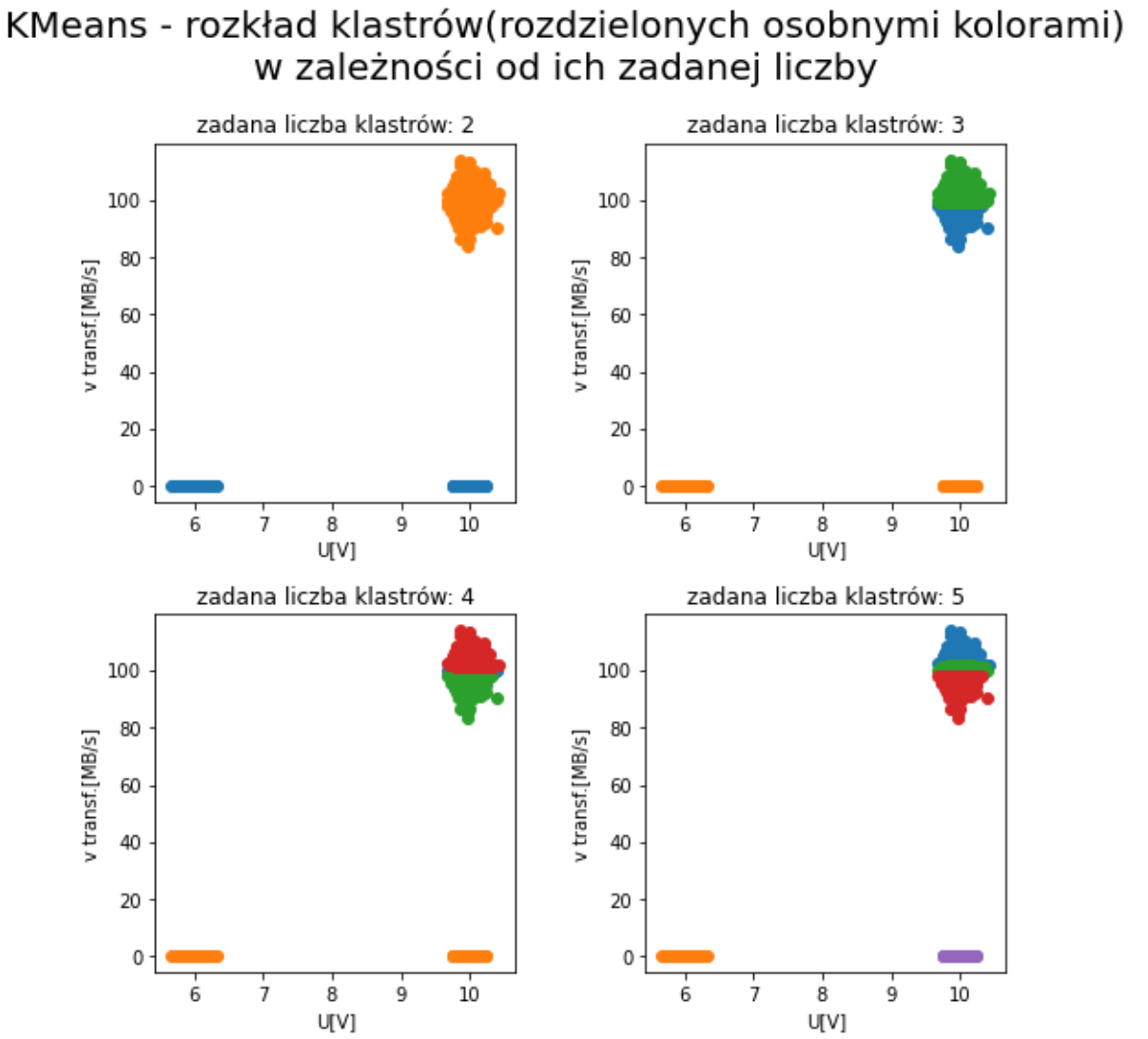


Fig. 8.2.4 - struktura klastrów algorytmu K-Means w zależności od zadanej ilości klastrów dla pomiarów ciśnienia i temperatury

Przykładowo dla siedmiu zadanych klastrów w zbiorze pojawiają się grupy opisujące wiele interesujących stanów zbiornika, szczególnie stan rozszczelnienia, ale też stan tuż przed rozszczelnieniem oraz stan tuż przed największym przyspieszeniem wartości ciśnienia - u “nasady” wykresu wykładniczego. Również wiele stanów pośrednich w normalnym cyklu pracy występuje w strukturze klastrów, które dla analityka mogą stanowić interesujące wskazówki co do kondycji systemu.

W przypadku pomiarów urządzenia komunikacyjnego:



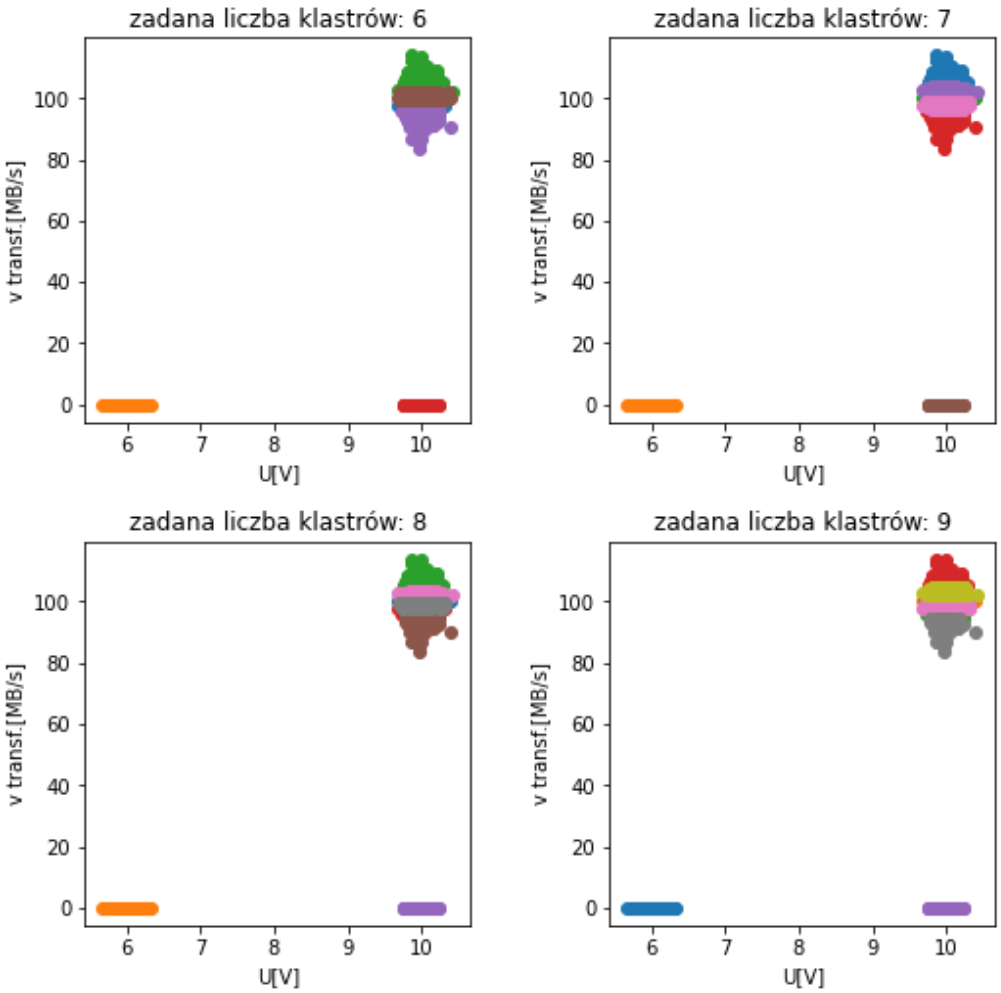


Fig. 8.2.5 - struktura klastrów algorytmu K-Means w zależności od zadanej ilości klastrów dla pomiarów transferu i napięcia

...również można dostrzec interesujące struktury klastrów, np. dla zadanej liczby sześciu i wzwyż wyodrębnione są stany zerwania transferu przy spadku napięcia, niestabilnego transferu po powrocie napięcia, stany normalne z wysokimi i z niskimi transferami oraz parę stanów pośrednich.

Dla pozostałych algorytmów i ustawień hiperparametrów również uzyskiwane są zróżnicowane struktury klastrów. Odpowiedni dobór algorytmów i parametrów mocno zależeć będzie od danej sytuacji i systemu jednak zdecydowanie możliwe jest wyciąganie konstruktywnych wniosków z uzyskiwanych w projekcie danych diagnostycznych. Do dalszych badań pozostają z pewnością możliwości zaangażowania innych mechanizmów analizy danych, w tym inteligentna analiza danych przez sieci neuronowe. Zmieniająca się w czasie rzeczywistym struktura klastrów i/lub wartości danych mogą dawać wyraźne przesłanki do rychłego wystąpienia danego zdarzenia w systemie, co przewidzieć może odpowiednio wytrenowana sieć.

9. Możliwości dalszego rozwoju projektu

Projekt ten może być rozwijany na szereg różnych sposobów. Stronę aplikacji można rozbudować o dodatkowe funkcjonalności oraz usprawnień szczególnie od strony dostępności dla użytkownika. Moduł filtracji można rozszerzyć o dodatkowe typy filtrowanych ramek oraz dodatkowe typy zmiennych.

Biblioteka DLL, która powstała z przekształcenia oprogramowania stworzonego w ramach pracy magisterskiej zawiera funkcje, w których występuje wiele efektów ubocznych np. Wysyłanie sygnału “SIGTERM”. Przez te efekty w aplikacji desktopowej należało pójść na wiele kompromisów np. restart aplikacji po zmianie ustawień przechwytywania lub przechwytywanie tylko na jednym porcie. Usunięcie efektów ubocznych zajęłoby zdecydowanie zbyt wiele czasu, przez co niemożliwe by było zrealizowanie innych założeń. Oczyszczenie kodu biblioteki umożliwiłoby implementację wielu przydatnych funkcjonalności.

Aplikacja desktopowa może być rozwinięta o dynamiczną zmianę parametrów przechwytywania (portu i rozmiaru bufora) bez konieczności jej ponownego uruchamiania. Dodatkowo nastawy filtrów sprzętowych mogłbyby być zapisywane do pliku XML. Dzięki temu nie byłoby konieczności ustawiania ich na nowo. Do aplikacji można dodać również możliwość konfiguracji dodatkowych ustawień urządzenia takich jak przepustowość port czy EEE.

Układ FPGA może zostać zmodyfikowany o kolejne moduły generujące informacje diagnostyczne. Przykładem takiego modułu może być obliczanie średniej wartości wybranego bajtu w ramce. Możliwe jest także zdefiniowanie modułu pozwalającego na wybranie kilku bajtów i traktowanie ich jak zmienną, a następnie wykonywanie operacji arytmetycznych. Obecnie istnieje tylko możliwość wysłania zebranych danych przez port PCIe, jednak w dalszym stadium rozwoju projektu może zostać on wzbogacony o wysyłanie ramek Ethernet zawierających informacje diagnostyczne do zdalnej stacji roboczej.

Moduł eksportu danych może być w przyszłości rozszerzony o automatyczną konfigurację - na zasadzie podobnej do obecnego automatycznego tworzenia i konfigurowania bazy danych na wybranym serwerze - mechanizmów zabezpieczania danych przed ich utratą, takich jak replikacja serwera SQL. Wstępna analiza dokumentacji frameworka .NET i serwerów MS SQL Server sugeruje, że taka konfiguracja z poziomu kodu jest możliwa, jednak nietrywialna.

Celem eksportu danych mogą być też w przyszłości inne, bardziej zaawansowane systemy gromadzenia danych takie jak chmury obliczeniowe, gdzie możliwa będzie od razu implementacja procesów analizujących na bieżąco przychodzące dane.

10. Podsumowanie i wnioski

Projekt pozwolił poszerzyć wiedzę w zakresie przemysłowych sieci Ethernet, protokołów sieciowych, programowania układów FPGA, tworzenia oprogramowania współpracującego ze sprzętem, analizy danych, baz danych oraz rozwijania istniejących projektów. Mimo wielu trudności związanych z warunkami epidemiologicznymi udało się zrealizować założenia projektu. Implementacja modułu typu “mock” do pozornego generowania ramek, znacząco ułatwiła prace nad rozwojem projektu bez dostępnego sprzętu.