Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

Instytut Elektroenergetyki Politechniki Warszawskiej

Praca dyplomowa inżynierska

na kierunku Automatyka i Robotyka Stosowana

Opracowanie w oprogramowaniu WinCC O.A. systemu SCADA (ang. Supervisory Control And Data Acquisition) do obsługi stacji laboratoryjnej poddetektorów FIT (ang. Fast Interaction Trigger) w eksperymencie ALICE (ang. A Large Ion Collider Experiment)

Przemysław Kinasz
numer albumu 319150

promotor
dr hab. inż. Łukasz Kolimas, profesor uczelni,
konsultant
mgr inż. Krystian Rosłon

WARSZAWA 2025

Serdecznie dziękuję mgr inż. Krystianowi Rosłonowi za nieocenioną pomoc i wsparcie udzielone podczas mojego pobytu w CERN oraz w trakcie realizacji niniejszej pracy inżynierskiej.

Opracowanie w oprogramowaniu WinCC O.A. systemu SCADA (ang. Supervisory Control And Data Acquisition) do obsługi stacji laboratoryjnej poddetektorów FIT (ang. Fast Interaction Trigger) w eksperymencie ALICE (ang. A Large Ion Collider Experiment)

Streszczenie

Przedmiotem pracy było opracowanie w oprogramowaniu SIMATIC WinCC Open Architecture Version 3.19 (WinCC OA) systemu Supervisory Control And Data Acquisition (SCADA) do obsługi stacji laboratoryjnej poddetektorów The Fast Interaction Trigger (FIT) w eksperymencie Wielkiego Zderzacza Jonów. A Large Ion Collider Experiment (ALICE) jest jednym z dużych eksperymentów działających na Wielkim Zderzaczu Hadronów w Europejskiej Organizacji Badań Jądrowych CERN w Genewie. Jego głównym celem jest badanie właściwości takiego stanu materii jak plazma kwarkowo-gluonowa. Według współczesnej wiedzy uważa się, że miał on miejsce ułamki sekund po Wielkim Wybuchu. Zespół poddetektorów FIT jest kluczowym systemem eksperymentu ALICE. Składają się na niego trzy systemy detektorowe FT0, FV0 oraz FDD, które pogrupowano w pięć modułów. Informują one mniej dynamiczne detektory w eksperymencie o pojawieniu się kolizji, określają płaszczyznę interakcji i centralność zderzenia. Prace nad rozwojem systemu prowadzone były w CERN dzięki współpracy Wydziału Elektrycznego z Wydziałem Fizyki Politechniki Warszawskiej w lipcu i sierpniu 2024 roku. Autor pracy uczestniczył w opracowywaniu nowej struktury sterowania systemu poddetektorów FIT oraz pełnił rolę ich eksperta dyżurnego on-call. Zakres zadań obejmował modernizację pierwotnego oprogramowania sterującego zespołem poddetektorów, które nie spełniało standardów obowiązujących w eksperymencie. W rezultacie opracowano zgodną z nimi strukturę, której integralną część stanowi opracowana przez autora i opisana w pracy aplikacja SCADA. Umożliwia ona pełną kontrolę nad nowym systemem. W procesie tworzenia systemu zaimplementowano do aplikacji WinCC OA stworzony przez CERN Joint Controls Project Framework (JCOP), dający możliwość komunikacji z niższymi warstwami systemu za pośrednictwem protokołu Distributed Information Management (DIM). Zdefiniowano strukturę przesyłanych ramek danych oraz przygotowano skrypt do ich importu do WinCC, umożliwiający integrację z wewnętrzną bazą danych. Ostatnim etapem było zaprojektowanie i zaprogramowanie interfejsu użytkownika do zarządzania modułami Front-end electronics (FEE): Processing Module (PM) oraz Trigger and Clock Module (TCM). Pierwszym miejscem, gdzie stworzony system został zastosowany, jest stacja laboratoryjna poddetektorów FIT. Umożliwia ona prowadzenie szkoleń dla ekspertów dyżurnych on-call. Każdy z detektorów w eksperymencie ma całodobowo przypisanego dyżurnego, który po wystąpieniu awarii łączy się z systemem sterowania i ją rozwiązuje. Dotychczasowa struktura oprogramowania nie pozwalała na generowanie błędów w warunkach symulacyjnych, co uniemożliwiało przyszłym ekspertom praktyczne szkolenie w ich rozwiązywaniu. Opracowany system docelowo ma zostać zaimplementowany do sterowania zespołem poddetektorów FIT w rzeczywistym eksperymencie.

Słowa kluczowe: CERN, eksperyment ALICE, FIT, automatyka, system SCADA

Development of a SCADA (Supervisory Control And Data Acquisition) system in WinCC O.A. software for the operation of the FIT (Fast Interaction Trigger) laboratory sub-detector station in the ALICE experiment (A Large Ion Collider Experiment)

Abstract

The subject of this thesis was the development of a Supervisory Control And Data Acquisition (SCADA) system in SIMATIC WinCC Open Architecture Version 3.19 (WinCC OA) for the operation of the laboratory station of The Fast Interaction Trigger (FIT) subdetectors in A Large Ion Collider Experiment. A Large Ion Collider Experiment (ALICE) is one of the major experiments conducted at the Large Hadron Collider within the European Organization for Nuclear Research (CERN) in Geneva. Its primary objective is to investigate the properties of a state of matter known as quark-gluon plasma, which according to current scientific knowledge is believed to have existed fractions of a second after the Big Bang. The FIT subdetector system is a key component of the experiment comprising three detector systems FT0, FV0, and FDD grouped into five modules. These detectors provide information to less dynamic detectors in the experiment regarding the occurrence of collisions, determine the interaction plane, and assess collision centrality. The development work was carried out at CERN in collaboration with the Faculty of Electrical Engineering and the Faculty of Physics of the Warsaw University of Technology during July and August 2024. The author of this thesis contributed to the development of a new control structure for the FIT subdetector system and served as an on-call expert. The scope of tasks included modernizing the original software controlling the subdetector system which did not meet the standards required for the experiment. As a result a compliant structure was developed including the SCADA application created by the author and described in this thesis enabling complete control over the new system. During the system's development the CERN Joint Controls Project Framework (JCOP) was implemented into the WinCC OA application allowing communication with lower system layers via the Distributed Information Management (DIM) protocol. The structure of transmitted data frames was defined and a script was prepared for their import into WinCC enabling integration with the internal database. The final stage involved designing and programming a user interface for managing the Front-end Electronics (FEE) modules the Processing Module (PM) and the Trigger and Clock Module (TCM). The first implementation of the developed system was at the detector simulation station in the FIT laboratory which allows for training on-call experts. Each detector in the experiment has an on-call expert assigned around the clock who resolves issues by connecting to the control system in case of a malfunction. The previous software structure did not allow for error generation under simulated conditions which prevented practical training for future experts in troubleshooting. The developed system is ultimately intended for implementation in the control of the FIT in the real ALICE experiment.

Keywords: CERN, ALICE experiment, FIT, control engineering, SCADA system

Spis treści

Rozdział 1

Wstęp - Europejska Organizacja Badań Jądrowych CERN

CERN, czyli Europejska Organizacja Badań Jądrowych (fr. Conseil Européen pour la Recherche Nucléaire), jest jednym z najbardziej prestiżowych i zaawansowanych ośrodków badawczych na świecie, koncentrującym się na prowadzeniu badań podstawowych z obszaru fizyki i rozwoju technologii inżynieryjnych. Organizacja zrzesza 23 państwa członkowskie i jej siedziba znajduje się na granicy Szwajcarii oraz Francji, w pobliżu Genewy. Ośrodek został założony w 1954 roku i od tego czasu odgrywa kluczową rolę w rozwoju nauki, technologii oraz międzynarodowej współpracy badawczej [1]. Polska w 1964 roku jako pierwszy kraj Europy Wschodniej została państwem obserwatorem CERN, a od 1991 roku posiada status państwa członkowskiego organizacji [2].

Badania doświadczalne fizyki wysokich energii polegają na zderzaniu hadronów, których właściwości są dobrze znane. Po rozpędzeniu ich do prędkości relatywistycznych, zbliżonych do prędkości światła, kierowane są one naprzeciw siebie lub w stronę stacjonarnej tarczy. Zgodnie z zasadami szczególnej teorii względności, ich energia kinetyczna ulega przemianie w masę, co prowadzi do powstania wielu nowych cząstek. Ich właściwości są badane przy użyciu specjalistycznych detektorów [3, 4].

Wśród odkryć dokonanych w CERN można wyróżnić odkrycie bozonu Z oraz W, które są nośnikami oddziaływania słabego. W 2012 roku dzięki Wielkiemu Zderzaczowi Hadronów potwierdzono istnienie bozonu Higgsa. Powiązany jest on z mechanizmem nadawania masy cząstkom elementarnym. Im silniej dana cząstka oddziałuje z jego polem, tym staje się cięższa [5, 6]. W 1989 roku w CERN wynaleziono sieć komputerową World Wide Web (WWW). Pierwotnie miała służyć do wymiany informacji między uniwersytetami i ośrodkami badawczymi. Dała jednak początek dostępu do internetu ludziom na całym świecie [7].

1.1 Wielki Zderzacz Hadronów LHC

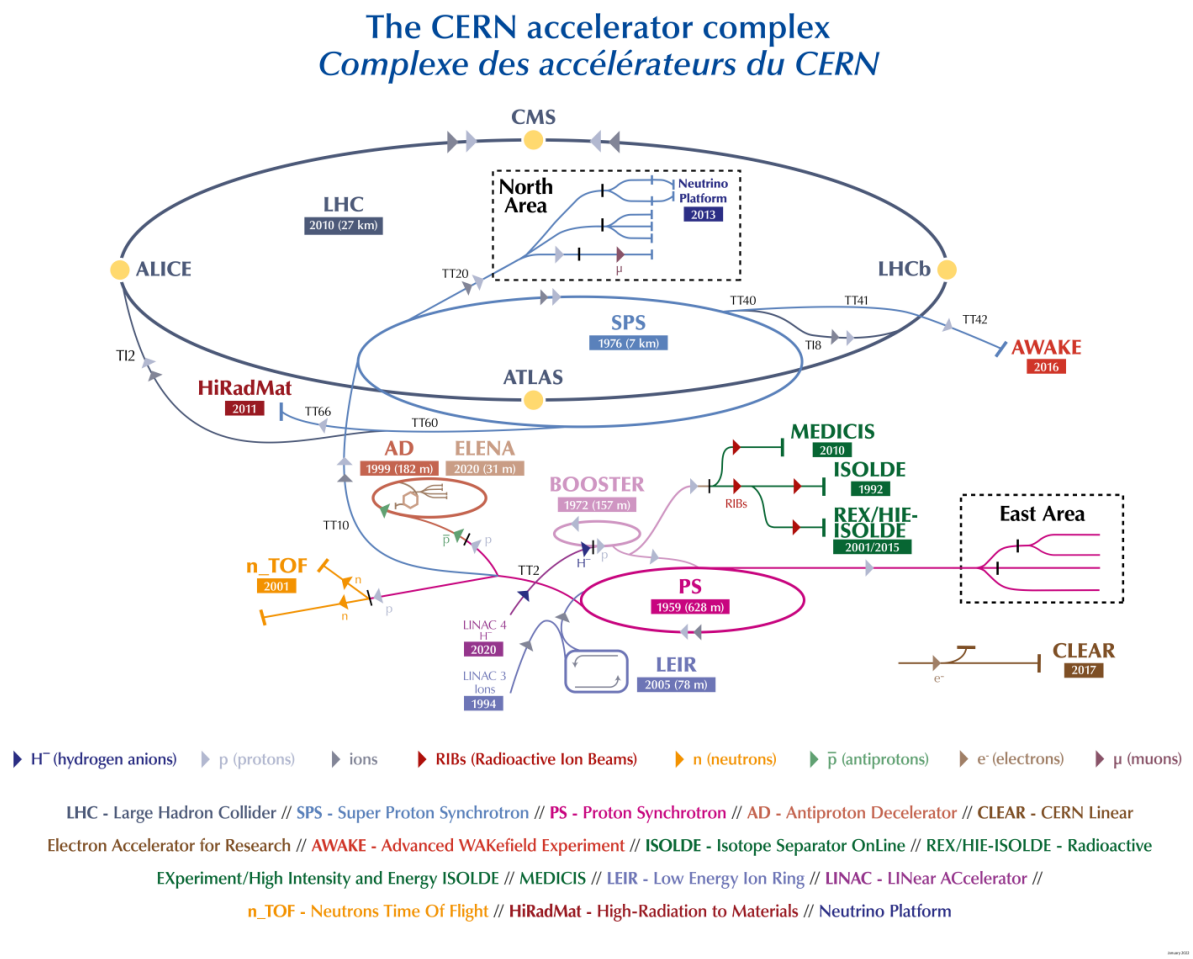
Akceleratory są urządzeniami rozpędzającymi w polu elektrycznym naładowane cząstki, takie jak elektrony, protony lub jony [8]. Wśród nich możemy między innymi wyróżnić akceleratory elektrostatyczne, liniowe, cyklotrony oraz synchrotrony, do którego należy Large Hadron Collider (LHC). W synchrotronach do przyspieszania wiązek i utrzymywania ich w pożądanym torze wykorzystywana jest sekwencja pola elektrycznego i magnetycznego. Prędkość hadronów zwiększana jest w dedykowanych wnękach rezonansowych. Zakrzywianie ich toru ruchu realizuje się za pomocą pola magnetycznego, którego indukcja rośnie wraz z prędkością cząstek, co pozwala na utrzymanie ich w stałej trajektorii [3].

W Europejskiej Organizacji Badań Jądrowych znajduje się zespół dziewięciu akceleratorów i dwóch deakceleratorów cząstek, które przedstawiono na rysunku 1. Największym akceleratorem na świecie jest znajdujący się tam LHC. Został uruchomiony po raz pierwszy 10 września 2008 roku. LHC ma 27 kilometrów długości i jest wysokoenergetycznym synchrotronem znajdującym się na końcu łańcucha pre-akceleratorów wstępnie rozpędzających protony lub ciężkie jony [4, 9].

Protony pozyskiwane są dzięki oddzieleniu od atomów wodoru elektronów, by następnie zostać rozpędzonym w akceleratorze liniowym Linear Accelerator 4 (LINAC4), do uzyskania energii $160\,\mathrm{MeV}$ [10]. Jeden elektronowolt odpowiada energii uzyskanej przez elektron przyspieszony w polu elektrycznym o natężeniu $1\,\mathrm{V}$ [4]. Następnie wstrzykiwane są do akceleratora Proton Synchrotron Booster (PSB), gdzie ich energia wzrasta do $2\,\mathrm{GeV}$. Kolejnym stopniem akceleracji jest Proton Synchrotron (PS), gdzie protony rozpędzane są do $26\,\mathrm{GeV}$. Przed wprowadzeniem wiązki do LHC protony są dodatkowo przyspieszane w synchrotronie Super Proton Synchrotron (SPS), osiągając energię $450\,\mathrm{GeV}$. W LHC, po końcowym etapie przyspieszenia, protony posiadają energię wynoszącą w przybliżeniu $6,8\,\mathrm{TeV}$ [11–13].

W CERN oprócz protonów przyspieszane są również ciężkie jony. Ich przykładem mogą być jony ołowiu, które otrzymywane są ze źródła rezonansu cyklotronowego elektronów (Electron Cyclotron Resonance Source), z podgrzanego do temperatury $550\,^{\circ}\mathrm{C}$ bloku ołowianego wzbogaconego w izotop $^{208}\mathrm{Pb}$, który znajduje się przed akceleratorem liniowym Linear Accelerator 3 (LINAC3) [12]. Następnie pary ołowiu jonizowane są przez strumień elektronów i wysyłane do spektrometru w celu wyselekcjonowania tych, które posiadają pożądany ładunek Pb^{27+}. Po selekcji wielostopniowy system RF przyspiesza wybrane typy jonów do $4,2\,\mathrm{MeV/u}$ i kieruje na folię węglową, która usuwa z nich elektrony, dzięki czemu powstają jony Pb^{53+}. Następnie wiązka kierowana jest z akceleratora LINAC3 do Low Energy Ion Ring (LEIR), gdzie przyspieszana jest do $72\,\mathrm{MeV/u}$ i przekazywana do synchrotronu protonowego PS. PS przyspiesza wiązkę do energii $5,9\,\mathrm{GeV/u}$ i przesyła ją do SPS. Po przepuszczeniu przez drugą folię węglową, gdzie wiązka oddzielana jest od reszty elektronów, otrzymane zostają jony $^{208}\mathrm{Pb}^{82+}$. SPS przyspiesza wiązkę do $450\,\mathrm{GeV/u}$, a następnie wstrzykuje do LHC, gdzie osiąga energię $5,36\,\mathrm{TeV/u}$ [11, 14].

Cząstki akceleratorowe w LHC poruszają się wewnątrz próżniowej rury zwanej również jako *beam pipe*. Składa się ona z dwóch równolegle biegnących do siebie przewodów, krzyżujących



Rysunek 1. Schemat kompleksu akceleracyjnego znajdującego się w Europejskiej Organizacji Badań Jądrowych CERN [13].

się w miejscach, w których znajdują się eksperymenty. Ze względu na chęć uniknięcia kolizji z innymi cząstkami wewnątrz rury utrzymywana jest ultrawysoka próżnia, gdzie ciśnienie wynosi 10^{-13} atmosfery. LHC w obrębie całej swojej długości nie jest stale zakrzywiony. Składa się z ośmiu prostych sekcji oraz ośmiu łuków, na których zamontowano 1232 zakrzywiające magnesy dipolowe. Ich niobowo-tytanowe ($NbTi$) cewki działają w stanie nadprzewodnictwa, generując pole magnetyczne o natężeniu 8,3 T. Chłodzone są ciekłym helem do temperatury 1,9 K ($-271.3°\mathrm{C}$) i przewodzą prąd osiągający 11700 A. Oprócz magnesów dipolowych w LHC działają również magnesy kwadrupolowe, sekstupolowe, oktupolowe, dekapolowe itp. Ich łączna ilość wynosi około 9600 i służą do korygowania oraz optymalizowania trajektorii wiązki [11, 12].

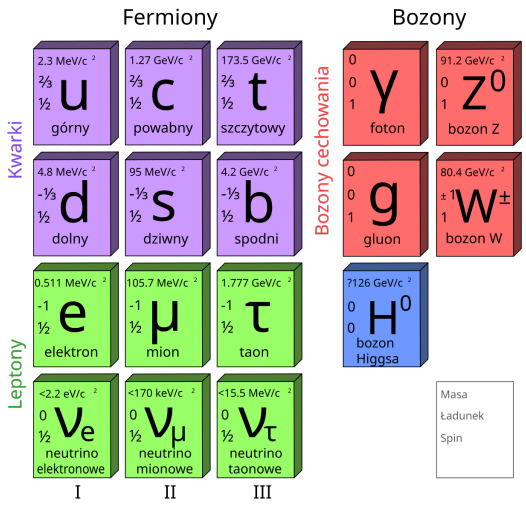
1.2 Eksperymenty działające w obrębie LHC

W ramach Wielkiego Zderzacza Hadronów realizowane są zaawansowane projekty badawcze, znane również jako eksperymenty. Wykorzystują wysoce precyzyjne i innowacyjne detektory cząstek, które umożliwiają obserwację i analizę zdarzeń zachodzących na poziomie subatomowym. Każdy z nich ma inne cele i skupia się na specyficznych aspektach fizyki cząstek elementarnych. Wśród największych z nich możemy wyróżnić ALICE, A Toroidal LHC Apparatus (ATLAS), Compact Muon Solenoid (CMS) oraz Large Hadron Collider beauty experiment (LHCb).

1.2.1 Eksperyment ALICE

ALICE jest jednym z głównych eksperymentów prowadzonych w Europejskiej Organizacji Badań Jądrowych. Jego celem jest ujawnienie właściwości takiego stanu materii jak plazma kwarkowo-gluonowa. Jest to zagadnienie z zakresu chromodynamiki kwantowej, będącej teorią oddziaływań silnych. Uważa się, że taki stan miał miejsce przez pierwsze mikrosekundy po Wielkim Wybuchu [15, 16].

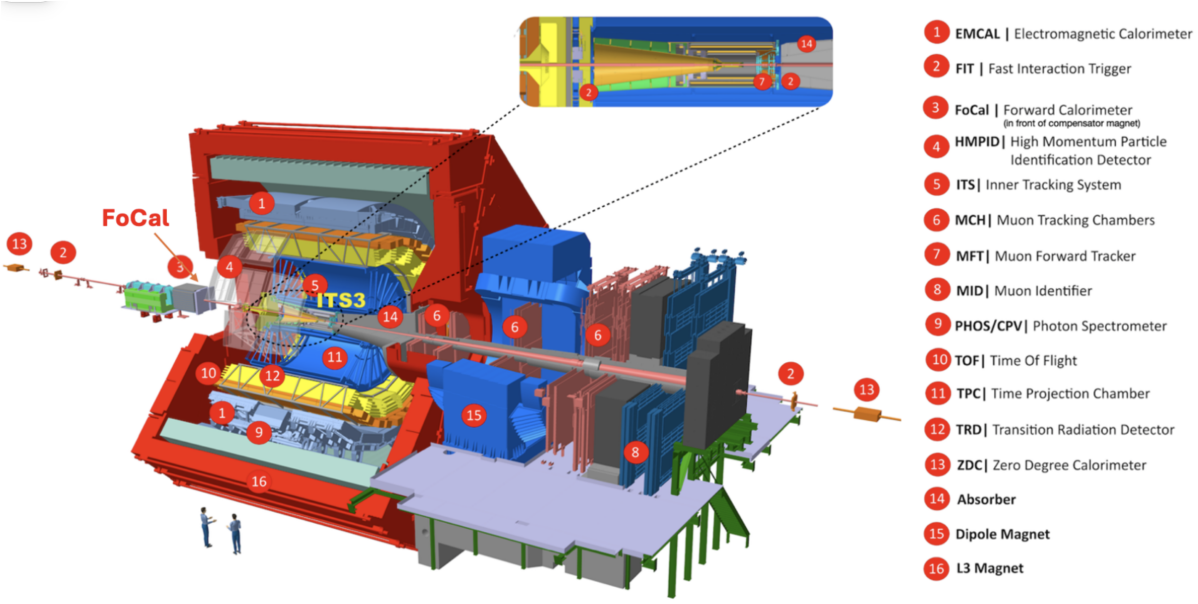
Materia jaką znamy zbudowana jest z jąder atomowych, na których powłokach znajdują się elektrony. Protony i neutrony, które tworzą jądra atomowe, składają się z kwarków. Istnieje sześć rodzajów kwarków, które przedstawiono na rysunku 2, takich jak: górny, dolny, powabny, dziwny, szczytowy oraz spodni. Kwarki połączone są dzięki silnemu oddziaływaniu przenoszonemu przez gluony. W zwykłej materii kwarki znajdują się wyłącznie wewnątrz hadronów [17, 18].



Rysunek 2. Cząstki elementarne modelu standardowego [19].

W eksperymencie ALICE dzięki rozpędzaniu w LHC ciężkich jonów ołowiu $^{208}\mathrm{Pb}^{82+}$ do ultra-relatywistycznych prędkości, zbliżonych do prędkości światła, a następnie ich zderzeniu udaje się uzyskać środowisko zbliżone do tego, jakie miało miejsce chwilę po Wielkim Wybuchu. Kwarki i gluony na ułamek sekundy stają się "wolne". Powstała plazma kwarkowo-gluonowa jest niezwykle gęsta, a

jej temperatura sięga 100000-krotności temperatury słońca. Stan ten jest jednak nietrwały i kwarki z gluonami łączą się, tworząc znów materię hadronową. Proces ten jest nieustannie obserwowany i rejestrowany przez zespół detektorów eksperymentu ALICE [15, 17].



Rysunek 3. Schemat eksperymentu ALICE [20].

ALICE ma wymiary 16x16x26 m^3 i waży około 10000 ton. W jego skład wchodzi zespół detektorów przedstawionych na rysunku 3 w punktach od 1 do 13. Oparte są one o technologie półprzewodnikowe, gazowe, scyntylatorowe, czy detektory promieniowania Czerenkowa. Pozwalają identyfikować typy powstałych cząstek, mierzyć ich energię, pęd, trajektorię jak i zachowanie w polu magnetycznym generowanym przez dwa magnesy konwencjonalne: dipolowy oraz L3. W obszarze badań eksperymentu ALICE oprócz zderzeń ciężkich jonów ołowiu $^{208}\mathrm{Pb}^{82+}$ dokonywane są również zderzenia proton-proton, proton-jądro oraz jądro-jądro [15].

Pakiety cząstek z LHC przechodzą przez eksperyment ALICE z częstotliwością dochodzącą do 40 MHz. W celu umożliwienia synchronizacji wszystkich detektorów, w ostatnim czasie zaimplementowano między innymi nowy zespół poddetektorów FIT [21]. Zintegrowano również system przetwarzania danych, mogący przesyłać je z prędkością dochodzącą do 1 Tb/s. Pozwoliło to na zwielokrotnienie wydajności w porównaniu do poprzedniej konfiguracji eksperymentu [22].

1.2.2 Pozostałe duże eksperymenty działające w CERN

Eksperyment ATLAS

Eksperyment ATLAS jest jednym z multidetektorów uniwersalnego zastosowania. Jego głównym zadaniem jest badanie podstawowych składników materii i fundamentalnych sił przyrody poprzez rejestrowanie zderzeń wysokoenergetycznych protonów oraz jonów. Jego nazwa nawiązuje to konstrukcji

systemu nadprzewodzących magnesów toroidalnych. Eksperyment ma 46 metrów długości, 25 metrów średnicy i waży aż 7 000 ton. Ma symetryczną budowę pozwalającą pokryć niemal 4π kąta bryłowego. Składa się z kilku kluczowych systemów, takich jak wewnętrzny detektor śledzący, mierzący trajektorię i pęd naładowanych cząstek, kalorymetry elektromagnetyczne i hadronowe mierzące ich energię oraz spektrometr mionowy służący do pomiaru pędu mionów. Detektor został zaprojektowany do pracy w ekstremalnych warunkach wysokiej liczby kolizji. Największym odkryciem dokonanym w ATLAS było eksperymentalne potwierdzenie istnienia bozonu Higgsa w 2012 roku [23–26].

Eksperyment CMS

CMS tak jak i ATLAS jest eksperymentem zastosowania uniwersalnego, jednak jego cele naukowe skupiają się na bardzo dokładnym badaniu mionów. Został zaprojektowany do badania zderzeń proton-proton oraz jądro-jądro z energią do $14\,\mathrm{TeV}$ przy zderzeniu centralnym. Multidetektor CMS ma zwartą i złożoną konstrukcję mierzącą 21x15x15 m^3 i ważącą około 12500 t. Pokrywa niemal 4π kąta bryłowego wokół punktu kolizji. Jego sercem jest cylindryczny magnes działający w stanie nadprzewodnictwa, który może generować pole o natężeniu 4 T [27]. Pierwszym systemem, na który trafiają cząstki powstałe w wyniku zderzenia jest detektor półprzewodnikowy, umożliwiający rejestrację pozycji cząstek oraz pomiaru ich pędu. Otaczają go kalorymetry: elektromagnetyczny Electromagnetic Calorimeter (ECAL) do pomiaru energii elektronów i fotonów oraz hadronowy Hadronic Calorimeter (HCAL) do detekcji cząstek składających się z kwarków. Jarzmo wykorzystanego tam magnesu zatrzymuje wewnątrz wszystkie cząstki oprócz neutrin i mionów, które identyfikowane są przez czterowarstwowy detektor mionowy, będący ostatnią warstwą detektora. Neutrina nie są w nim widoczne, lecz ich obecność można wnioskować na podstawie deficytu energii powstałej w wyniku zderzenia [28].

Eksperyment LHCb

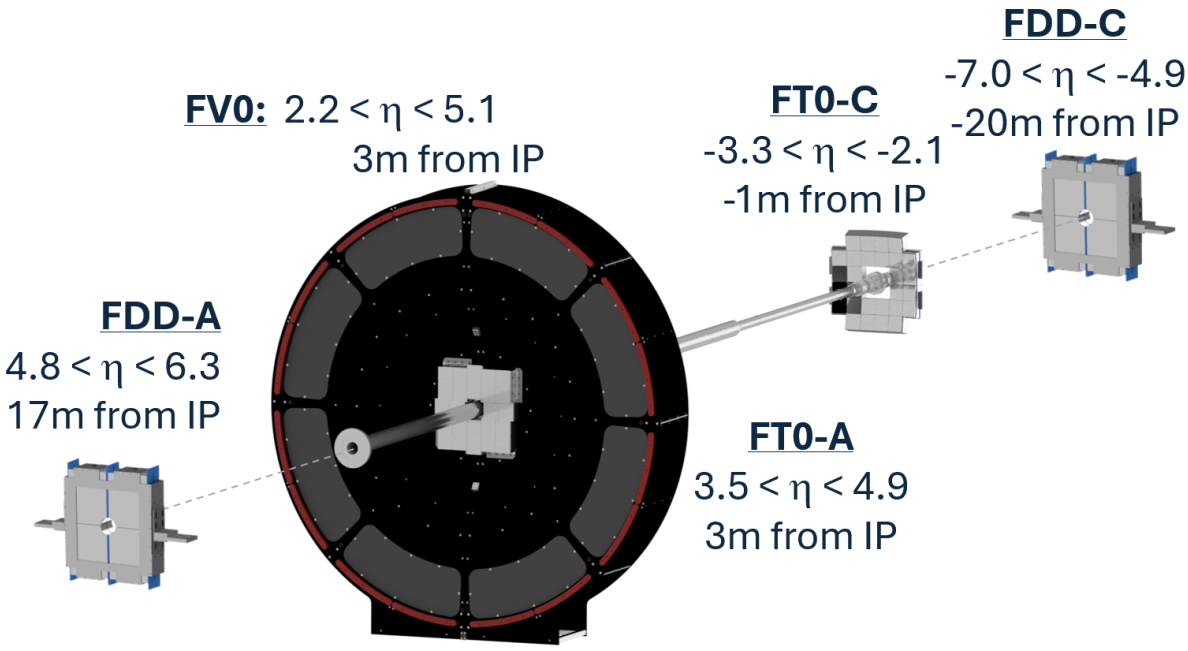
Celami naukowymi eksperymentu LHCb jest badanie różnic między materią i antymaterią poprzez obserwację kwarków b (pięknych). LHCb w przeciwieństwie do pozostałych opisanych multidetektorów, ma konstrukcję otwartą, skupiającą się na badaniu cząstek emitowanych w jednym kierunku od miejsca kolizji. Detektor ma wymiary 21x10x13 m^3 i waży około 5600 t [12]. Na jego konstrukcję składają się: krzemowy detektor wierzchołka zderzenia (VELO), który precyzyjnie określa miejsce zderzenia cząstek, dwa detektory promieniowania Czerenkowa (RICH 1 i RICH 2) umożliwiające identyfikację rodzaju cząstek na podstawie ich pędu, zaawansowany system śledzenia torów cząstek oraz zestaw kalorymetrów służących do pomiaru ich energii. Kluczową rolę odgrywa magnes dipolowy, który generuje pole magnetyczne pozwalające na zakrzywienie torów ruchu naładowanych cząstek. Dodatkowo, system mionowy, składający się z pięciu stacji detekcyjnych, jest odpowiedzialny za identyfikację mionów, na podstawie ich zdolności do przenikania przez poprzednie warstwy detekcyjne [29].

1.3 System ALICE-FIT

Jednym z systemów detektorowych działających w eksperymencie ALICE jest przedstawiony na rysunku 4 FIT. Został wprowadzony do działania podczas drugiej, długiej przerwy modernizacyjnej LHC w latach 2018-2022. Jest kluczowym systemem dla całego eksperymentu. Między innymi służy jako wyzwalacz interakcji, informując inne, wolniejsze detektory o pojawieniu się kolizji. Określa jego miejsce i wskazuje płaszczyznę. Umożliwia identyfikację cząstek na podstawie czasu ich lotu. Potrafi zwracać do systemu LHC informacje o ilości zderzeń oraz ich świetlności, a także monitorować tło w trakcie eksperymentów. Dodatkowo mierzy przekroje procesów dyfrakcyjnych [30]. Świetlność $\mathcal{L}$ zderzacza jest zdolnością do wytwarzania wymaganej liczby kolizji i miarą proporcjonalności między liczbą zderzeń na sekundę $\frac{dR}{dt}$, a ich przekrojem σ_p. Określana jest wzorem [31]:

$$\frac{dR}{dt} = \mathcal{L} \cdot \sigma_p.$$

FIT składa się z trzech systemów detektorowych FT0, FV0 oraz FDD, które pogrupowano w pięć modułów. Tylko niektóre z detektorów w eksperymencie ALICE pracują w trybie ciągłym (ZDC, TOF, MCH, ITS, MFT, MID, TPC). Część z nich (HMPID, EMCAL, TRD, CPV, DCAL, PHOS) wymaga zastosowania zewnętrznego wyzwolenia, który zapewnia zespół poddetektorów FIT [30, 32].



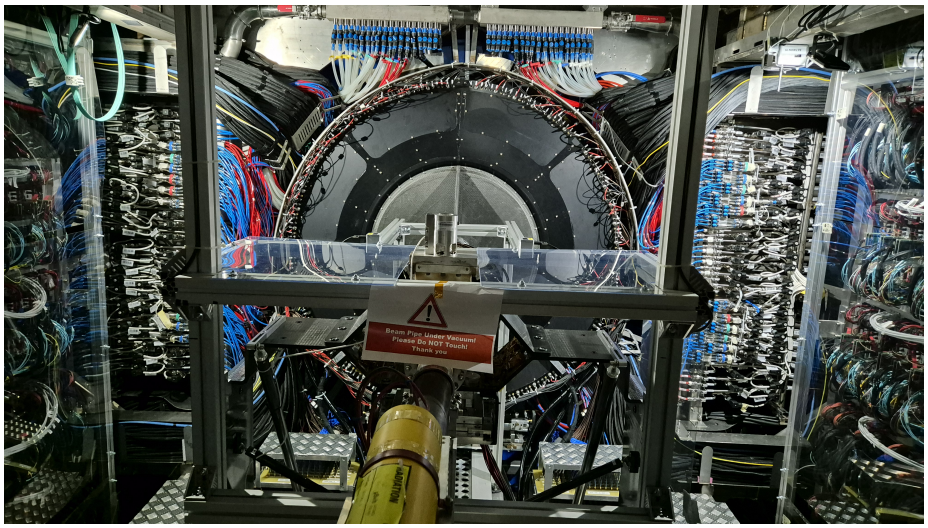
Rysunek 4. Wizualizacja zespołu poddetektorów The Fast Interaction Trigger [33].

1.3.1 Detektor FT0

Detektor Forward Time Zero (FT0) jest najszybszym z detektorów FIT. Umożliwia niemal natychmiastową identyfikację kolizji oraz pomiar świetlności. Składa się z dwóch jednostek: FT0-A umieszczonego trzy metry od punktu zderzenia oraz FT0-C znajdującego się metr za nim. Jego rozdzielczość wynosi 208 modułów [32]. Oparty jest on o kryształy kwarcowe, w których emitowane jest promieniowanie Czerenkowa. Zjawisko to polega na wyemitowaniu fotonów w formie stożka czerenkowa, podczas przejścia wysokoenergetycznej cząstki przez ośrodek dieelektryczny, z prędkością większą od prędkości fazowej światła w tym ośrodku [34]. Do wykrywania fotonów emitowanych w kryształach wykorzystuje się zmodyfikowane fotosensory Planacon MCP-PMT [30, 35].

1.3.2 Detektor FV0

Największym detektorem systemu FIT jest przedstawiony na rysunku 5 detektor Forward V0 (FV0). Służy on do określenia krotności, centralności oraz płaszczyzny zderzeń. Składa się z macierzy scyntylatora umieszczonej na pięciu okręgach i w sześciu sektorach. Sygnały z każdego z sektorów odczytywane są za pomocą dwóch fotosensorów [32]. Efekt scyntylacji polega na zjawisku emitowania wytraconej energii podczas przejścia promieniowania przez materię w postaci promieniowania elektromagnetycznego [36].



Rysunek 5. Autorska fotografia z wnętrza eksperymentu ALICE przedstawiająca obudowę macierzy detektora FV0.

1.3.3 Detektor FDD

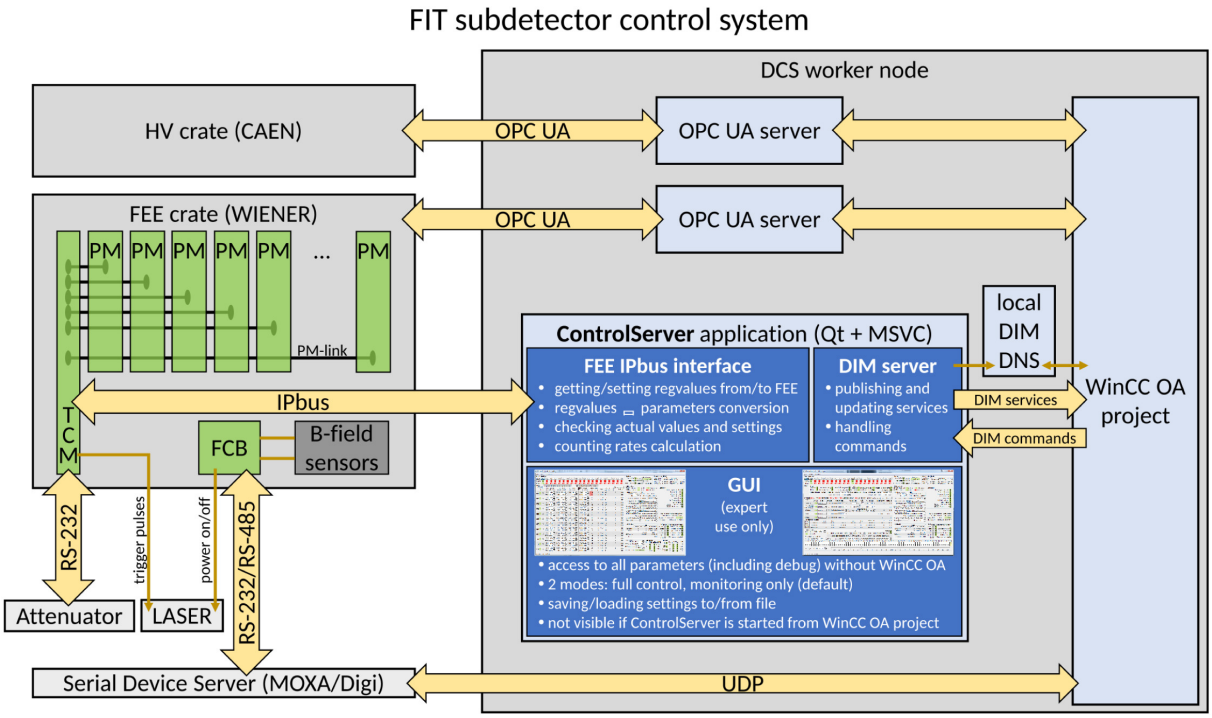
Detektor Forward Diffractive Detector (FDD) składa się z dwóch jednostek: FDD-A oraz FDD-C. Znajdują się one siedemnaście metrów przed miejscem kolizji i dwadzieścia metrów za nim. Tak jak detektor FV0 oparty jest o macierz scyntylatora. Posiada łączną rozdzielczość wynoszącą 16 modułów detekcyjnych. Do jego zadań należy monitorowanie zjawisk dyfrakcyjnych oraz tła [32].

Rozdział 2

Modernizacja struktury sterowania poddetektorów FIT

2.1 Struktura dotychczasowego systemu

Struktura sterowania systemu poddetektorów FIT składała się z trzech głównych elementów: węzła roboczego systemu sterowania *DCS worker node*, zasilacza wysokiego napięcia *HV Crate (CAEN)* oraz jednostki zawierającej moduły kondycjonujące sygnały PM i TCM *FEE Crate (WIENER)*, które przedstawiono na rysunku 6.



Rysunek 6. Struktura kontoli i sterowania detektora w eksperymencie ALICE [32].

Najwyższą warstwę systemu stanowi oprogramowanie SCADA. Komunikuje się ono z układem kondycjonowania sygnałów oraz zasilania za pośrednictwem serwera OPC Unified Architecture (OPC UA). Dodatkowo, WinCC wykorzystuje lokalny serwer DIM do komunikacji z serwerem kontroli (Control Server). Serwer kontroli zapewnia kompleksowe zarządzanie FEE. Posiada zintegrowane środowisko graficzne, pozwalające na monitorowanie i kontrolę wszystkimi parametrami modułów PM oraz TCM. Opracowany został z wykorzystaniem środowiska QT. Komunikuje się z FEE wykorzystując zrozumiałe dla układów FPGA połączenie IPbus, które oparto na protokole User Datagram Protocol (UDP) [37].

2.2 Opis struktury systemu po modernizacji

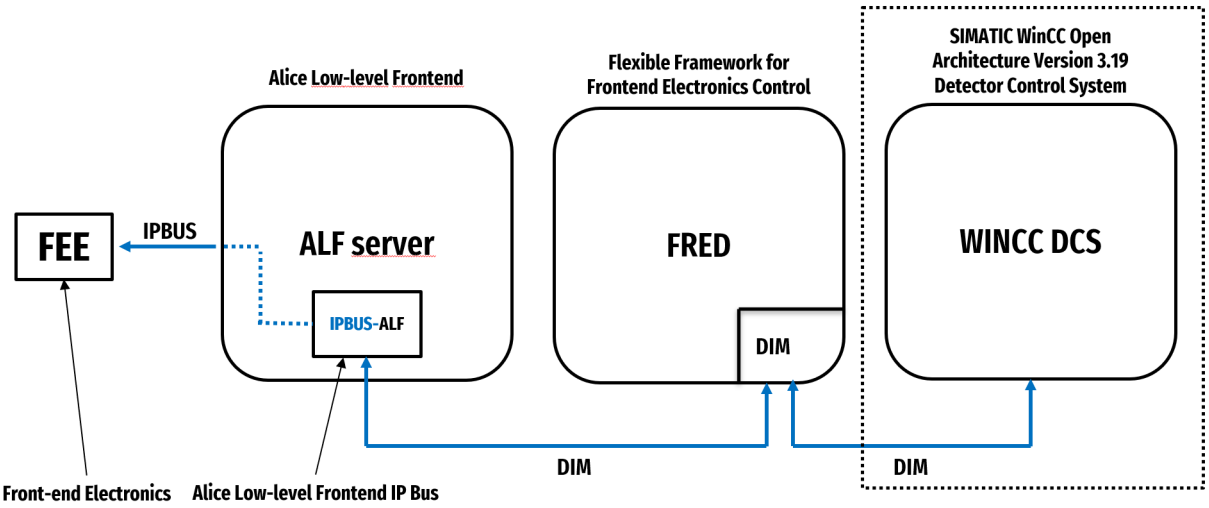
Serwer kontroli przedstawiony na rysunku 6 stworzony został przez naukowców ze Zjednoczonego Instytutu Badań Jądrowych w Dubnej. Zaimplementowano go jako dedykowaną aplikację wraz z wprowadzeniem FIT do eksperymentu ALICE. Wśród jego wad można wyróżnić:

- Brak kompatybilności ze strukturami systemów sterowania wykorzystywanymi w eksperymencie ALICE.
- Wysoki poziom skomplikowania struktury.
- Brak możliwości rekonfiguracji.
- Osadzenie całego systemu sterowania na jednej jednostce *localhost*.
- Brak wsparcia technicznego od grudnia 2024 roku, w związku z opuszczeniem przez Rosjan Europejskiej Organizacji Badań Jądrowych.

Aby rozwiązać opisane problemy, podjęto decyzję o modernizacji systemu sterowania zespołem poddetektorów FIT. Dotychczasowy serwer kontroli zastąpiono frameworkiem ALFRED, którego górną warstwę stanowi opisany w niniejszej pracy system SCADA (rysunek 7). Struktura ta jest szeroko wykorzystywana w eksperymencie ALICE. Framework ALFRED składa się z dwóch głównych komponentów: ALICE low level frontend (ALF) oraz Flexible Framework for Frontend Electronics (FRED), które ściśle współpracują, zapewniając stabilną integrację systemów niskiego i wysokiego poziomu [37].

ALF pełni rolę interfejsu łączącego elektronikę detektorów z systemami wyższego poziomu. Jego zadaniem jest umożliwienie kontroli detektorów z wykorzystaniem protokołu DIM. Wykorzystując komponent IPBUS-ALF, tłumaczy polecenia otrzymane z warstwy FRED na takie, zrozumiałe dla FEE, przesyłane za pośrednictwem magistrali IPBus. Magistrala ta oparta jest na modelu klient-serwer, gdzie FEE jest serwerem, a ALF klientem. Umożliwia odczyt i zapis wartości rejestrów, a także dokonywanie operacji bitowych takich jak RMWbits - selektywnego zapisywania oraz odczytywania bitów w rejestrze oraz RMWsum - dodawanie na wartościach rejestru [38].

FRED pełni rolę interfejsu odpowiedzialnego za tłumaczenie pakietów danych wysokiego poziomu, na niskopoziomowe, odpowiednie dla ALF. Rozpakowuje i konwertuje również dane otrzymywane od ALF, przed przesłaniem ich do WinCC OA. Pozwala przetwarzać informacje z detektora na użyteczne



Rysunek 7. Struktura systemu sterowania FEE po modernizacji.

parametry fizyczne. Umożliwia łatwą rekonfigurację systemu oraz dostosowanie go do działania z innymi detektorami [37].

Pierwszym krokiem implementacji nowego systemu sterowania jest uruchomienie go w laboratorium FIT. Opracowana wersja ma również posłużyć uruchomieniu stacji treningowej dla ekspertów dyżurnych on-call poddetektorów FIT.

Rozdział 3

Stacja treningowa dla ekspertów dyżurnych on-call systemu poddetektorów FIT

Podczas funkcjonowania detektora, gdy zbierane są dane ze zderzeń w LHC, obsługa całodobowo czuwa nad procesem działania eksperymentu. W pomieszczeniu kontrolnym przebywa lider zmiany, dyżurny obsługi detektorów, dyżurny obsługi eksperymentu oraz osoba kontrolująca jakość zbieranych danych. Jednocześnie każdy z detektorów posiada eksperta dyżurnego on-call, który dostępny jest zdalnie przez całą dobę. W przypadku zauważenia błędu przez obsługę stacjonarnie nadzorującą eksperyment, lider zmiany wykonuje telefon do dyżurnego detektora, w którym wystąpił błąd. Ekspert dyżurny on-call loguje się zdalnie do systemu kontroli, a następnie diagnozuje i rozwiązuje problem.

Aktualnie proces szkolenia ekspertów dyżurnych w systemie poddetektorów FIT odbywa się w formie krótkiego kursu wzbogaconego przez dostarczoną instrukcję tekstową. Aktualna struktura systemu sterującego stacją laboratoryjną oraz detektorem uniemożliwiała prowadzenie szkoleń w zakresie obsługi i rozwiązywania błędów systemu. Dzięki przeprowadzeniu modernizacji osoba prowadząca szkolenie może ręcznie wyzwalać błędy warstwie FRED, co daje możliwość rozwiązywania ich przez przyszłego eksperta dyżurnego.

3.1 Laboratorium systemu poddetektorów FIT

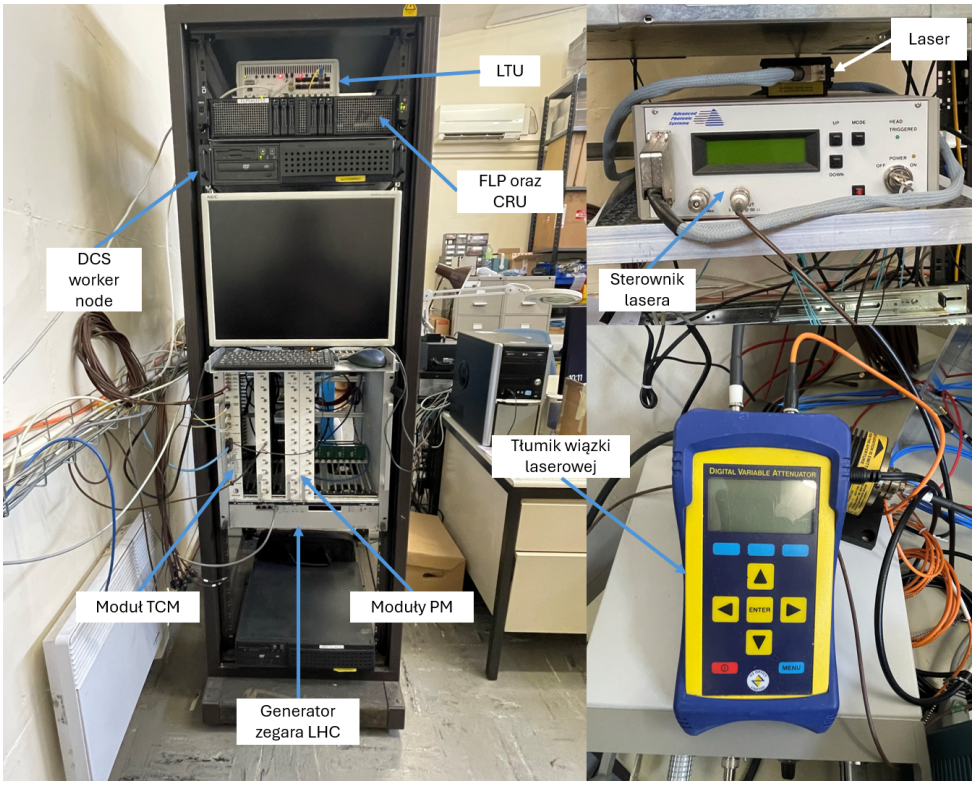
W Europejskiej Organizacji Badań Jądrowych CERN zbudowano laboratorium poddetektorów systemu Fast Interaction Trigger, które odgrywa kluczową rolę w przygotowaniu i weryfikacji systemów detekcyjnych jeszcze przed użyciem w pełnoskalowym eksperymencie ALICE. Opracowano uniwersalny model stacji symulacyjnej, który umożliwia odwzorowanie reakcji rzeczywistego detektora w warunkach laboratoryjnych. W tym procesie moduły detekcyjne pobudzane są przez układ laserowy, zastępujący wysokoenergetyczne cząstki. Wygenerowane sygnały rozpoznawane są za pomocą fotopowielaczy, a następnie trafiają do modułów odpowiedzialnych za kondycjonowanie

sygnałów. Na rysunku 8 przedstawiono obudowę stacji, w której znajdują się moduły detekcyjne, wyzwalane wiązką laserową.

Rysunek 8. Obudowa stacji symulacyjnej detektora, wewnątrz której znajdują się fotopowielacze MCP-PMT.

W laboratorium znajdują się również systemy kondycjonowania i wstępnego przetwarzania sygnałów. Spośród aparatury laboratoryjnej przedstawionej na rysunku 9, można wyróżnić narzędzia przedstawione w poniższej liście.

- **Układ wyzwalania Local Trigger Unit (LTU)**
  Urządzenie to jest częścią systemu wyzwalającego detektory w eksperymencie ALICE. Posiada dwa tryby pracy, którymi są tryby: emulacji i autonomiczny. W trybie emulacji LTU działa niezależnie od funkcjonowania centralnego systemu wyzwalania. Pozwala to na prowadzenie testów i kalibracji niezależnie od funkcjonowania systemu. Może również generować błędy, w trybie losowym jak i na żądanie. W trybie autonomicznym LTU współpracuje z centralnym systemem wyzwalania. Odbiera z niego sygnały, a następnie przekształca w format odpowiedni dla detektorów [39].
- **Moduły PM**
  Moduły PM (rysunek 10 oraz 11) odpowiadają za kondycjonowanie analogowych sygnałów, które dostarcza detektor. Każdy z nich umożliwia podłączenie do 12 kanałów detektora. Każdy z jego kanałów wyposażony jest w dwa 12-bitowe przetworniki analogowo-cyfrowe, które aby zapewnić ciągłość zbierania danych pracują naprzemiennie. Gdy jeden z nich zbiera dane, drugi jest odczytywany. Rozdzielczość czasowa okna zbierania danych modułu wynosi 13,02 ps [32].
- **Moduł TCM**
  TCM (rysunek 10 oraz 11) w systemie elektroniki FIT pełni funkcję centralnej jednostki koordynacyjnej. Wszystkie moduły PM należące do poszczególnego detektora podłączone są do jednego TCM. Kondycjonuje on dane zebrane przez moduły procesujące i dystrybuuje je do systemu akwizycji danych za pośrednictwem szybkiego łącza optycznego. Jego przepustowość wynosi 4,8 Gb/s. Dodatkowo zapewnia synchronizowanie zbieranych danych z zegarem otrzymywanym z systemów LHC [32].

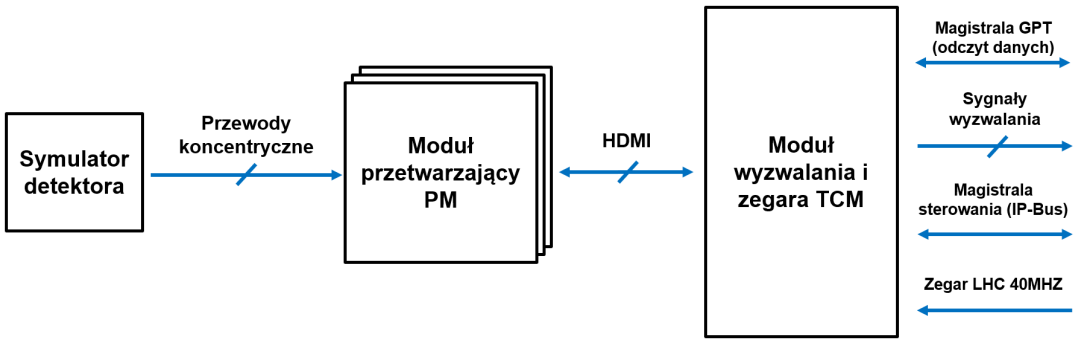


Rysunek 9. Aparatura, pozwalająca na odwzorowanie oprzyrządowania FIT znajdującego się w rzeczywistym eksperymencie ALICE.

- **Jednostka wspólnego odczytu Common Readout Unit (CRU)**
  CRU jest kluczowym elementem systemu zbierania danych w eksperymencie ALICE. Jest interfejsem pomiędzy elektroniką detektora FEE, a jednostkami obliczeniowymi First Level Processor (FLP). Jedno CRU wspiera podłączenie, aż 24 łączy Gigabit Transceiver (GBT). Jego głównym zadaniem jest odbieranie danych z detektorów przesyłanych za pomocą protokołu GBT, a następnie wstępne ich przetwarzanie i buforowanie. Dodatkowo, CRU współpracuje z systemem wyzwalania, zapewniając synchronizację danych z różnych detektorów i ich precyzyjne oznaczenie czasowe. CRU przesyła dane do FLP za pomocą inferfejsu PCI Express (PCIe) (x16) [40, 41].
- **Układ wstępnego przetwarzania FLP**
  System FLP ma za zadanie wstępnie przetwarzać dane napływające z systemów niższego poziomu. Wstępnie filtruje, agreguje oraz kompensuje dane, co pozwala na znaczne zmniejszenie ich objętości przed przesłaniem do centralnej infrastruktury obliczeniowej. FLP odgrywa kluczową rolę w przetwarzaniu danych w czasie rzeczywistym, szczególnie w trybie ciągłego odczytu, gdzie dane nie są dzielone na pojedyncze zdarzenia, lecz przesyłane jako stały strumień. Dzięki wstępnej obróbce realizowanej przez FLP, systemy centralne mogą skupić się na bardziej zaawansowanych operacjach, takich jak pełna rekonstrukcja zdarzeń i ich analiza [40, 42].

Rysunek 10. Moduły PM oraz TCM znajdujące się w laboratorium poddetektorów FIT w CERN.

- **Sterownik, moduł oraz tłumik wiązki laserowej**
  Układ ten symuluje działanie detektora w warunkach laboratoryjnych. Tworzone impulsy świetlne odwzorowują te, generowane w rzeczywistym eksperymencie przez wysokoenergetyczne cząstki.
- **Generator zegara LHC**
  Generator sygnału zegarowego pełni rolę zegara synchronizacyjnego, naśladując sygnał otrzymywany z systemów LHC.
- **Zasilacz CAEN SY4527**
  Służy do zasilania układu detekcyjnego fotopowielaczy. Z dołączonym modułem wysokiego napięcia CAEN A7030DP może zapewniać zasilanie o napięciu od 0 do $3\,\mathrm{kV}$ i mocy maksymalnej wynoszącej $1.5\,\mathrm{W}$ [37].



Rysunek 11. Schemat elektroniki w laboratorium systemu poddetektorów FIT w CERN.

Rozdział 4

System SCADA

Rozdział ten dedykowany jest przedstawieniu właściwości systemów SCADA, ich zastosowań oraz historii. Przedstawione zostanie również wykorzystywane w Europejskiej Organizacji Badań Jądrowych oprogramowanie SIMATIC WinCC Open Architecture Version 3.19 oraz opracowany na potrzeby modernizacji system SCADA.

4.1 Zastosowanie systemów SCADA

Systemy SCADA są przeznaczone do zdalnego monitorowania, nadzorowania oraz sterowania procesami przemysłowymi. Ich głównym zadaniem jest gromadzenie danych z różnorodnych urządzeń, takich jak czujniki, sterowniki PLC czy dedykowane urządzenia elektroniczne, a następnie ich przetwarzanie i prezentacja w sposób przystępny dla operatorów, najczęściej za pomocą interfejsów graficznych [43].

Dzięki bieżącej obserwacji wszystkich parametrów procesu przemysłowego (lub jak w przypadku CERN eksperymentu fizycznego), operatorzy wykorzystując prosty panel graficzny mogą decydować o jego przebiegu oraz diagnozować ewentualne problemy. Nowoczesne systemy SCADA pozwalają na zdalną kontrolę, co eliminuje konieczność fizycznej obecności operatora w miejscu realizacji procesu. Współpracują również z zaawansowanymi bazami danych, co umożliwia ich przechowywanie przykładowo w celu wykorzystania narzędzi analitycznych do prognozowania i minimalizowania ryzyka wystąpienia awarii [44].

4.2 Historia systemów SCADA

W przemyśle systemy kontroli pojawiają się od lat 60. XX wieku. Wówczas stosowane były rozwiązania analogowe, które pozwalały operatorom monitorować i regulować parametry procesów technologicznych, takich jak temperatura czy ciśnienie, bezpośrednio z pomieszczenia kontrolnego. W tamtych czasach operatorzy ręcznie sterowali ustawieniami, dokonując zmian za pomocą przycisków

i pokręteł. Systemy SCADA jako osobna technologia zaczęły wyróżniać się dopiero w latach 70. XX wieku. Wraz z rozwojem komputerów pozwalały na coraz większą automatyzację kontroli i obsługi.

Systemy pierwszej generacji umożliwiały łączenie tylko i wyłącznie ze zdalnymi terminalami. Każdy z systemów działał samodzielnie i praktycznie nie umożliwiał łączenia z innymi. Procesy monitorowano i sterowano z jednego miejsca, co zapewniało spójną bazę danych, lecz rodziło problemy ze skalowalnością oraz znaczną podatność na awarie. Wykorzystywane wówczas protokoły komunikacyjne były opracowywane przez dostawców sprzętu i były dedykowane do wykorzystania z danym systemem. W celu zwiększenia stabilności wykorzystywano systemy redundantne, łącząc dwa identyczne systemy jedną magistralą. Podejście takie minimalizowało ryzyko awarii, umożliwiając ciągłość działania nawet w przypadku uszkodzenia jednego z systemów. Znacząco jednak zwiększało to jego koszty [43–45].

Od lat 80. można wyróżnić drugą generację systemów SCADA. Rozwój technologii umożliwił poprzez implementację sieci lokalnych Local Area Network (LAN) stworzenie architektury rozproszonej. Dzięki temu pojawiła się możliwość integracji dużej liczby stacji operatorskich dysponujących własną mocą obliczeniową, co podniosło niezawodność i elastyczność systemów. Jednocześnie pojawiały się nowe protokoły komunikacji takie jak Modbus, czy Profibus [43–45].

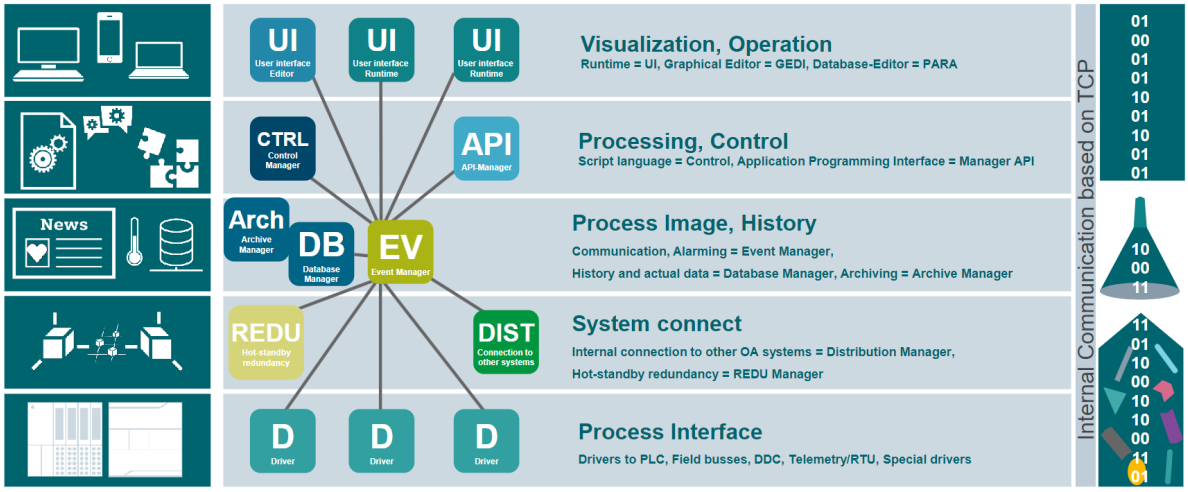
Generacja trzecia pojawiła się na początku XXI wieku. Jej główną zaletą wykorzystanie otwartych standardów i protokołów komunikacji, zamiast zastrzeżonych przez dostawców. Pozwala to na łatwą integrację systemów z urządzeniami od różnych dostawców. Zaczęto również wykorzystywać technologie internetowe, co umożliwia łączenie się z instalacjami rozproszonymi bez ponoszenia wysokich kosztów infrastruktury. Nowoczesne systemy SCADA wprowadziły również rozbudowane środowiska baz danych, które pozwalają na gromadzenie, przetwarzanie i analizowanie ogromnych ilości danych w czasie rzeczywistym [43–45].

4.3 Oprogramowanie SIMATIC WinCC Open Architecture

WinCC OA znane również jako SIMATIC WinCC Open Architecture jest wykorzystywanym w CERN oprogramowaniem umożliwiającym tworzenie systemów wizualizujących i nadzorujących procesy przemysłowe. Dawną nazwą oprogramowania jest PVSS [46]. WinCC OA ma charakter modułowy i jego funkcjonalności realizowane są przez poszczególne komponenty zwane menadżerami (rysunek 12). Komunikacja pomiędzy wewnętrznymi warstwami przebiega w jednolity sposób za pomocą protokołu TCP/IP. W poprawnie stworzonym systemie, w przypadku braku zmian w poszczególnych menadżerach, dane nie są wymieniane, co niemal całkowicie ogranicza wewnętrzny ruch pakietów.

Najniższy poziom aplikacji stanowi interfejs modułowy. Umożliwia on wymianę danych ze sterownikami i urządzeniami zewnętrznymi, poprzez tłumaczenie ich protokołów komunikacyjnych na format zrozumiały dla WinCC OA. Pozwala na wykorzystanie protokołów takich jak Profibus, ModbusTCP, OPC czy Ethernet IP [47].

Głównym elementem WinCC OA jest menadżer zdarzeń Event Manager (EV). Stanowi centrum komunikacyjne oprogramowania. Przechowuje w swojej pamięci aktualny stan wszystkich procesów,



Rysunek 12. Struktura oprogramowania SIMATIC WinCC Open Architecture Version 3.19 [48].

w komórkach zwanych Data Point (DP). W przypadku, gdy inna jednostka chce uzyskać informację o stanie drugiej, ma możliwość pobrania jej z pamięci EV, bez potrzeby komunikowania się z jednostką docelową. Menadżer ten ma również możliwość samodzielnego wykonywania obliczeń oraz w przypadku wystąpienia problemów sygnalizowania alarmów [48].

Równolegle do menadżera zdarzeń działa również menadżer bazy danych. Wszystkie zapytania kierowane do bazy danych nie są kierowane do niej bezpośrednio, lecz pośredniczy im menedżer. Zarządza danymi konfiguracyjnymi umieszczonymi w bazie, zapisuje w niej wartości zmiennych oraz ewentualnych alarmów. W WinCC dane procesowe powstałe w procesie sterowania lub wizualizacji archiwizowane są w sposób chronologiczny w postaci serii plików. Dostęp do nich realizowany jest za pomocą języka Structured Query Language (SQL) [48].

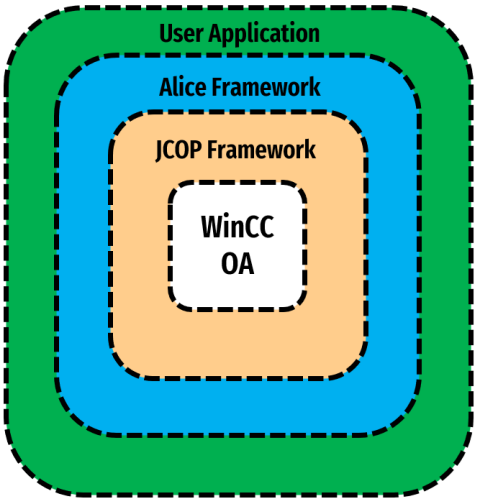
Interfejs użytkownika jest obsługiwany przez UI Manager. Obejmuje on między innymi edytor wizualizacji (GEDI), edytor bazy danych (PARA) oraz moduł wizualizacyjny (VISION). Moduł zawiera bibliotekę funkcji, pozwalając użytkownikom dodawać własne panele oraz wtyczki. Posiada również wbudowany edytor skryptów. Dzięki stworzonym interfejsom graficznym operatorzy w przystępny sposób mogą nadzorować parametry procesów technologicznych, sterować urządzeniami czy generować raporty. Oddzielenie warstwy wizualizacji od warstwy logiki przetwarzania sprawia, że system jest czytelny i łatwy w utrzymaniu [48].

WinCC OA oferuje również możliwość tworzenia własnych skryptów. Służy do tego dedykowany język programowania Control (CTRL), oparty na składni C Programming Language (ANSI C), wzbogaconej o elementy ułatwiające pracę z obiektami systemu. Pozwala on na szybką implementację dodatkowych modułów mogących importować zewnętrzne bazy danych, dokonywać obliczeń, czy rozwijać funkcjonalność interfejsu użytkownika [48].

W celu integracji WinCC OA z innymi systemami w CERN wykorzystuje się narzędzie JCOP [49].

4.4 Struktura narzędzia JCOP

JCOP jest pakietem oprogramowania, które ma za zadanie uprościć i ujednolicić strukturę systemów sterowania w CERN. Przed jego wprowadzeniem systemy tworzone przez naukowców z krajów członkowskich organizacji, rozwijane były według różnych standardów, co znacznie utrudniało ich integrację. JCOP stanowi ujednoliconą warstwę (rysunek 13) pomiędzy systemami SCADA, protokołami komunikacyjnymi takimi jak OPC [50], DIP [51], DIM [52] oraz innymi rozwiązaniami programistycznymi, w tym narzędziami do tworzenia maszyn stanów skończonych (FSM).



Rysunek 13. Struktura systemów sterowania wykorzystywanych w eksperymencie ALICE w CERN.

JCOB Framework ma konstrukcję modułową. Pozwala na instalację tylko tych komponentów, które są potrzebne w danym projekcie. Pierwszym elementem oprogramowania, jaki można wyróżnić, jest narzędzie instalacyjne. Odpowiada za importowanie, aktualizowanie oraz usuwanie dodawanych do WinCC OA modułów. Nadzoruje również ich pracę, umożliwiając prawidłową integrację wszystkich elementów systemu.

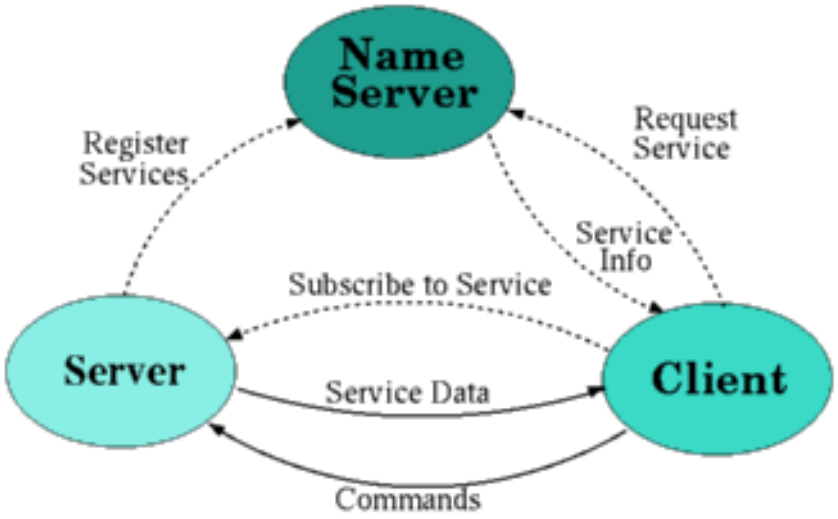
Komponent główny (Core Component) jest podstawowym elementem funkcjonowania JCOB framework. Wśród jego funkcji można wyróżnić możliwość ustawiania hierarchii urządzeń, ich dodawania i konfigurowania oraz definiowania alarmów.

Kolejnym zbiorem komponentów dostarczanych przez JCOB Framework są komponenty narzędziowe (Tool Components). Pozwalają one na przetwarzanie danych poprzez zapewnienie protokołów komunikacji, umożliwienie prezentowania wykresów, pobieranie i zapisywania danych w systemach bazodanowych itd.

Inną grupą są komponenty dedykowane do konkretnych urządzeń wykorzystywanych w CERN. Wśród nich możemy wyróżnić wykorzystywane w CERN zasilacze wysokiego napięcia, elektronikę detekcyjną oraz systemy akwizycji danych [49].

4.5 Protokół komunikacji Distributed Information Management System (DIM)

DIM jest opracowanym w CERN protokołem służącym do publikowania informacji, transferu danych oraz komunikowania się między procesami. Opiera się na powszechnie stosowanym paradygmacie wydawcy-subskrybenta (publisher-subscriber) przedstawionym na rysunku 14. Jest podstawowym protokołem wymiany danych w systemach rozproszonych eksperymentu ALICE. W protokole DIM wydawcy udostępniają subskrybującym serwisy. Ich nazwa może przybierać dowolny format. Gdy klient rozpoczyna subskrypcję danej usługi, wydawca automatycznie w określonych odstępach czasu aktualizuje przypisane do niej dane. W celu zapewnienia przejrzystości systemu, wprowadzono koncepcję serwera nazw. Serwer nazw utrzymuje aktualny katalog dostępnych serwerów i usług w systemie, co umożliwia dynamiczną konfigurację systemu [52].



Rysunek 14. Struktura protokołu DIM [52].

W przypadku, gdy serwer przesyła informacje do klienta, wysyłany jest DIM serwis. Natomiast w sytuacji, gdy klient przesyła ramkę danych do serwera, wysyłana jest DIM komenda. W opisywanym w niniejszej pracy przypadku rolę serwera pełni warstwa tłumacząca FRED, a klientem jest stworzony przez autora pracy system SCADA.

4.6 Opracowany system SCADA

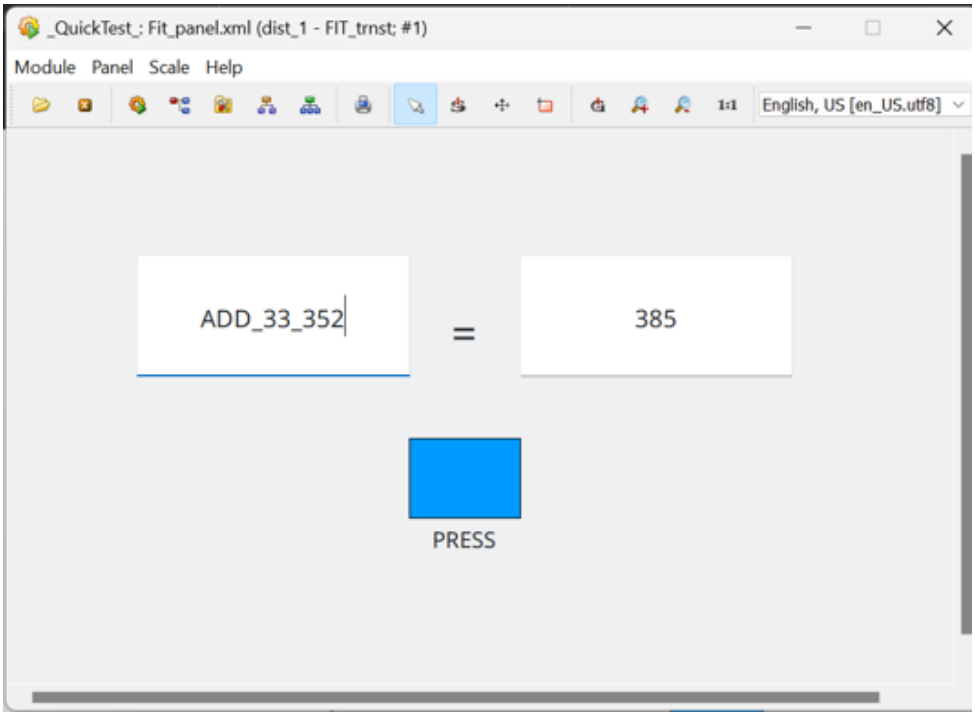
System SCADA będący przedmiotem niniejszej pracy został opracowany na podstawie funkcjonalności dotychczas wykorzystywanej aplikacji serwera kontroli. Oprogramowanie stanowi najwyższą warstwę modernizowanego systemu sterowania detektorami FIT w eksperymencie ALICE. Jego pierwotnym zadaniem jest stworzenie środowiska treningowego dla ekspertów dyżurnych, umożliwiając prowa-

dzenie szkoleń w realistycznych warunkach symulacyjnych. W przyszłości, system zostanie zaimplementowany do sterowania rzeczywistymi detektorami. Zgodnie ze standardami wykorzystywanymi w CERN system oparto o oprogramowanie SIMATIC WinCC Open Architecture wraz z dołączonym narzędziem JCOP. Pozwoliło to na opracowanie uniwersalnej aplikacji, która z łatwością może zostać dostosowana do sterowania różnymi detektorami FIT.

4.6.1 Początek rozwoju systemu

Pierwszym etapem prac było zainstalowanie oprogramowania SIMATIC WinCC Open Architecture Version 3.19 oraz JCOP Framework. W celu implementacji do WinCC narzędzia JCOP Framework wykorzystano dedykowane oprogramowanie instalacyjne JCOP. Podczas procesu instalacji do WinCC OA dodano rozszerzenie fwCore oraz fwDIM. Pozwalają one na komunikację między systemami, wykorzystując sieć lokalną w CERN.

Ze względu na to, że systemy ALF oraz FRED były tworzone od podstaw, od momentu przystąpienia autora niniejszej pracy do projektu, pierwszym opracowanym narzędziem był prosty kalkulator, przedstawiony na rysunku 15.



Rysunek 15. Kalkulator opracowany w celu walidacji połączenia WinCC OA z FRED.

WinCC OA przesyłało do serwera w warstwie FRED ramkę danych w formacie tekstowym *string* z poleceniem. Serwer przetwarzał dane i zwracał wynik w postaci serwisu z obliczoną wartością, który następnie był wyświetlany na panelu użytkownika. Jego implementacja pozwoliła na zweryfikowanie poprawności działania serwera DIM w warstwie FRED oraz na przetestowanie stabilności i skuteczności połączenia, stanowiąc solidny fundament dla dalszego rozwoju systemu.

Następnie przystąpiono do tworzenia docelowego systemu SCADA, skryptów oraz paneli interfejsu użytkownika.

4.6.2 Skrypt importujący DIM serwisy i komendy

Problemem jaki wystąpił, było importowanie listy DIM serwisów i komend. Na początku, w miarę rozwoju warstwy FRED wykonywano to ręcznie. Z biegiem czasu okazało się jednak, że rozwiązanie to jest nieoptymalne. Importowanie każdej pojedynczej zmiennej wymagało ręcznego stworzenia w bazie danych rekordu, wprowadzenia nazwy serwisu lub komendy do menadżera DIM oraz połączenia go z rekordem w bazie danych. Ostateczna liczba importowanych komend i serwisów wyniosła ponad jedenaście tysięcy. Podjęto decyzję o opracowaniu skryptu importującego nazwy komend i serwisów z pliku tekstowego lub bazy danych.

W WinCC OA własne skrypty dodaje się w katalogu *Scripts*, który pojawia się po rozwinięciu drzewa projektu w interfejsie graficznym GEDI. Następnie w panelu menadżerów dodano nowy menadżer, uruchamiający stworzony skrypt podczas inicjalizacji projektu. Wśród funkcji skryptu można wyróżnić:

- Importowanie danych wejściowych z pliku tekstowego lub bazy danych.
- Tworzenie rekordów w bazie danych WinCC OA.
- Importowanie listy DIM serwisów oraz komend do JCOP Framework.
- Sprawdzanie błędów w strukturze danych wejściowych.

W toku prac nad rozwojem nowego systemu przyjęto następującą strukturę DIM serwisów oraz komend. Jako przykład można przytoczyć serwis:

FIT_LAB/LAB/READOUTCARDS/TCM0/DELAY_A_ANS

FIT_LAB jest nazwą własną systemu SCADA. Nawiązuje do laboratorium systemu poddetektorów FIT. W drugim członie nazwy określona jest nazwa detektora. W tym przypadku *LAB* jest stacją symulacyjną. Następnie *READOUTCARDS/TCM0/DELAY_A_ANS* jest nazwą własną danego serwisu lub komendy.

W pliku źródłowym oprócz nazwy serwisu znajdują się również jego parametry. Rozdzielone są one średnikami. Jego format można opisać w następujący sposób, widoczny w tabeli 1. Wartość

nazwa DIM serwisu lub komendy;	wartość bazowa;	typ zmiennej;	serwis/komenda

Tabela 1. Format danych wyjściowych skryptu do importowania DIM serwisów oraz komend.

bazowa jest wartością przypisywaną w bazie danych jako wartość domyślna. Następnie określany jest typ zmiennej. Wśród nich można wyróżnić takie typy jak int, string, float, bit32 czy char. Ostatnim parametrem jest wskazanie czy dany rekord jest serwisem lub komendą.

Pierwszym krokiem działania skryptu jest otworzenie pliku wejściowego (w tym przypadku w formacie csv) i odczytanie go. W tabeli 2 przedstawiono wyciąg ośmiu przykładowych wierszy pliku wejściowego. Każdy wiersz dzielony jest na fragmenty, na podstawie umieszczonych w nim średników

i przepisywany do tabeli, która ma cztery kolumny oraz tyle wierszy, ile rekordów znajduje się w pliku. W przypadku gdy liczba członów danego wiersza jest różna od czterech, oznaczana jest flaga błędu i skrypt zostaje zatrzymany.

LAB/PM/PMC0/18VPOWER_ANS;0;string;s
LAB/PM/PMC0/1VPOWER_ANS;0;string;s
LAB/PM/PMC0/ADC_DELAY_1_ANS;0;string;s
LAB/PM/PMC0/ADC_DELAY_2_ANS;0;string;s
LAB/PM/PMC0/ADC_DELAY_3_ANS;0;string;s
LAB/READOUTCARDS/TCM0/LASER_DIVIDER_ANS;0;string;s
LAB/READOUTCARDS/TCM0/LASER_DIVIDER_REQ;0;string;c

Tabela 2. Tabela przykładowych danych wejściowych skryptu importującego DIM serwisy oraz komendy.

Następnie tablica ta jest sortowana alfabetycznie z wykorzystaniem algorytmu sortowania bąbelkowego. Algorytm sortowania bąbelkowego polega na porównywaniu wartości danych sąsiadujących w parach i odwracaniu ich kolejności w przypadku, gdy pierwsza jest większa od drugiej. Złożoność tego algorytmu wynosi $\mathcal{O}(n^2)$ [53]. Kod funkcji przedstawiono w listingu 1. Następne kroki działania algorytmu będą miały na celu zbudowanie tablic danych wejściowych w formacie oczekiwanym przez JCOP.

```
int filetosortarray()
{
    f=fopen("D:/inputdim.csv","r");
    flong = 1;
    max = 0;
    while (feof(f)==0)
    {
        fgets(dummy,150,f);
        partstemp[flong] = dummy;
        parts = strsplit(partstemp[flong], ";");
        if (dynlen(parts) != 4)
        {
            errorflag=1;
            DebugN("ERROR: error in input file in line: "+flong);
            continue;
        }
        dpparts = strsplit(parts[1], "/");
        tabsize=dynlen(dpparts);
        if (max<tabsize)
            max=tabsize;
        devpart[flong] = strsplit(partstemp[flong], ";");
        flong++;
    }
    rewind(f);
    fclose(f);
```

```
26      for (int i=1; i <=dynlen(devpart);i++)
27          for (int j = 1; j <= dynlen(devpart)-i; j++)
28              if (devpart[j][1].trimmed()>devpart[j+1][1].trimmed())
29              {
30                  dyn_string temp = devpart[j];
31                  devpart[j] = devpart[j + 1];
32                  devpart[j + 1] = temp;
33              }
34      return max;
35  }
```

Listing 1. Funkcja importująca i sortująca dane z pliku wejściowego.

Następnie algorytm z listingu 2 przepisuje wartości z tablicy przechowującej posortowane dane do zmiennych pomocniczych. Jednocześnie stosowana jest funkcja *trimmed()*, która usuwa ewentualnie pojawiające się puste znaki. Rozpoznawany jest typ zmiennej z pliku źródłowego i przypisywany poprawny, zgodny ze standardem języka Control. Dane są sprawdzane i w przypadku wykrycia błędu ustawiana jest jego flaga. Nazwy DIM serwisów i komend dzielone są na części według ukośnika.

```
1       main()
2   {
3       fwDim_deleteConfig(CONFIG_NAME);
4       fwDim_createConfig(CONFIG_NAME);
5       firstIteration = 1;
6       int indexJ=1;
7       dyn_string vars;
8       max= filetosortarray();
9       for(int g=1;g<flong;g++)
10      {
11          parts = (devpart[g]);
12          dp = parts[1].trimmed();
13          value = parts[2].trimmed();
14          type = parts[3].trimmed();
15          dimtype=parts[4].trimmed();
16          if ((dimtype!="s")&&(dimtype!="c"))
17          {
18              errorflag=1;
19              DebugN("ERROR: DIM type error in line: "+g);
20          }
21          if (type=="int") vartype=DPEL_INT;
22          else if (type=="string") vartype=DPEL_STRING;
23          else if (type=="float") vartype=DPEL_DYN_FLOAT;
24          else if (type=="b32") vartype=DPEL_BIT32;
25          else if (type=="char") vartype=DPEL_CHAR;
26          else
27
28          {
```

```
29              errorflag=1;
30              DebugN("ERROR: DIM type error in line: "+g);
31          }
32          parts[1]=parts[1].trimmed();
33          dpparts = strsplit(parts[1], "/");
34          tabsize=dynlen(dpparts);
```

Listing 2. Funkcja resetująca konfigurację i rozpoznająca typy zmiennych.

Następnie skrypt z listingu 3 ma za zadanie zbudować dwie tablice danych, które zostaną wykorzystane do wczytania konfiguracji. JCOP framework oczekuje formatu danych przedstawionych w tabelach 3 i 4. Dzięki temu, że tablice zostały wcześniej posortowane, można z łatwością usunąć powtarzające się dane.

Zmienne konstrukcyjne			Zmienne strukturalne
LAB			
	PM		
		PMC0	
			18VPOWER_ANS
			1VPOWER_ANS
			ADC_DELAY_1_ANS
			ADC_DELAY_2_ANS
			ADC_DELAY_3_ANS
	READOUTCARDS		
		TCM0	
			LASER_DIVIDER_ANS
			LASER_DIVIDER_REQ

Tabela 3. Struktura tablicy DIM serwisów i komend, opracowana na podstawie danych z tabeli 2.

1	-	-	-
0	1	-	-
0	0	1	-
0	0	0	25
0	0	0	25
0	0	0	25
0	0	0	25
0	0	0	25
0	1	-	-
0	0	1	-
0	0	0	25
0	0	0	25

Tabela 4. Struktura tablicy określającej typ DIM serwisów oraz komend, opracowana na podstawie danych z tabeli 2.

W tabeli 4 przedstawiono tablicę danych wejściowych potrzebną do wygenerowania struktury bazy danych. Cyfra 1 odpowiada zmiennej strukturalnej *DPEL_STRUCT*, 0 braku danej w określonym miejscu, a 25 *DPEL_STRING*.

```
string gen;
int cont=1;
int indexJp;
       if (firstIteration == 1) {
    for (int j = 1; j <= max; j++) {
        dyn_string tempArr;
        int tempZ = 1;
        for (int z = 1; z <= j; z++) {
            tempArr[z] = "";
            tempZ = z;
            if (cont == 1 || z == j) {
                xxdepei[indexJ][z] = DPEL_STRUCT;
            } else {
                xxdepei[indexJ][z] = 0;
            }
            cont = 2;
            size = z;
            indexJp = indexJ;
        }
        tempArr[tempZ] = dpparts[j].trimmed();
        for (int z = tempZ + 1; z <= max; z++) {
            tempArr[z] = "";
        }
        xxdepes[indexJ] = tempArr;
        indexJ++;
    }
    xxdepei[indexJp][size] = vartype;
    firstIteration = 0;}
else {
    int startIndex = 1;
    int notFound = 1;
    for (int j = 1; j <= ((dynlen(dpparts) > dynlen(vars)) ? dynlen(vars
   ) : dynlen(dpparts)); j++) {
        startIndex = j;

        if (dpparts[j].trimmed() != vars[j].trimmed()) {
            notFound = 0;
            break;
        }
    }
    if (startIndex == 1) {
        dpTypeChange(xxdepes, xxdepei);
```

```
42
43          if (dpExists(xxdepes[1][1])) {
44              DebugN("WARNING: Already exists!");
45          } else {
46              dpCreate(xxdepes[1][1], xxdepes[1][1]);
47          }
48          xxdepes.clear();
49          xxdepei.clear();
50          indexJp = 1;
51          indexJ = 1;
52      }
53      vars = dpparts;
54      for (int j = startIndex; j <= dynlen(dpparts); j++) {
55          dyn_string tempArr;
56          int tempZ = 1;
57          for (int z = 1; z <= j; z++) {
58              tempArr[z] = "";
59              tempZ = z;
60              if ((cont == 1 || z == j) && (notFound == 1)) {
61                  xxdepei[indexJ][z] = DPEL_STRUCT;
62              } else {
63                  xxdepei[indexJ][z] = 0;
64              }
65              if ((z == j) && (notFound == 0)) {
66                  xxdepei[indexJ][z] = DPEL_STRUCT;
67              } else {
68                  xxdepei[indexJ][z] = 0;
69              }
70              cont = 2;
71              size = z;
72              indexJp = indexJ;
73          }
74          tempArr[tempZ] = dpparts[j].trimmed();
75          for (int z = tempZ + 1; z <= max; z++) {
76              tempArr[z] = "";
77          }
78          xxdepes[indexJ] = tempArr;
79          indexJ++;
80      }
81      xxdepei[indexJp][size] = vartype;
82  }
83  vars = dpparts;
84  genDIM();
```

Listing 3. Struktura funkcji tworzącej tablice wejściowe.

Następnie wywoływana jest funkcja subskrybująca DIM serwisy i komendy. Jej kod przedstawiono w listingu 4.

```
void genDIM()
{
    dpdot=dp;
    dpdot.replace("/",".");
    if (dimtype=="s")
        fwDim_subscribeService(CONFIG_NAME,dp,WINCC_project_name+":"+
   dpdot.trimmed(), value);
    else if (dimtype=="c")
        fwDim_subscribeCommand(CONFIG_NAME,dp,WINCC_project_name+":"+
   dpdot.trimmed(), value);
    else
    {
        errorflag=1;
        DebugN("ERROR: error in input file in line: "+flong);
        continue;
    }
}
```

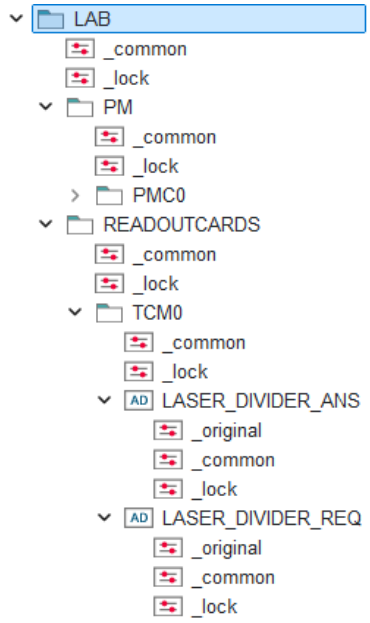
Listing 4. Funkcja subskrybująca DIM serwisy i komendy.

W przypadku, gdy nie wystąpiły błędy, konfiguracja jest aplikowana. Skrypt przekazuje również informację o wyniku operacji do konsoli. Przedstawiono go w listingu 5.

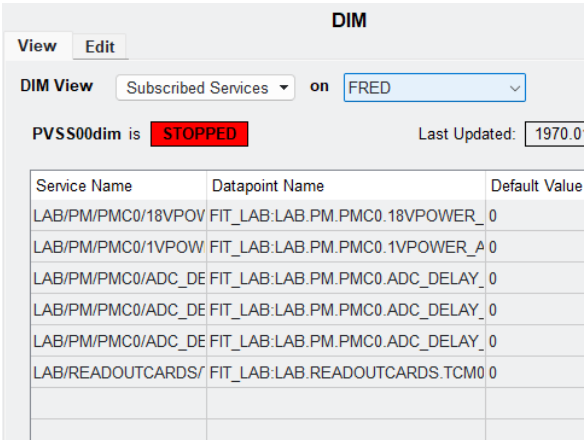
```
int dp_create_status;

if (errorflag == 0)
    dp_create_status = dpTypeChange(xxdepes, xxdepei);

if (dp_create_status == 0 && errorflag == 0)
    DebugN("INFO: new configuration correctly loaded");
else
    DebugN("ERROR: error loading configuration");
}
```

Listing 5. Kod aplikujący konfigurację.

Wynikiem działania skryptu jest poprawne skonfigurowanie rozszerzenia JCOP fwDIM, wygenerowanie struktury bazy danych w WinCC (rysunek 16) oraz połączenie DIM serwisów i komend z przynależnymi rekordami przedstawionymi na rysunku 17.



Rysunek 16. Wygenerowana struktura w bazie danych PARA na podstawie danych z tabeli 2.



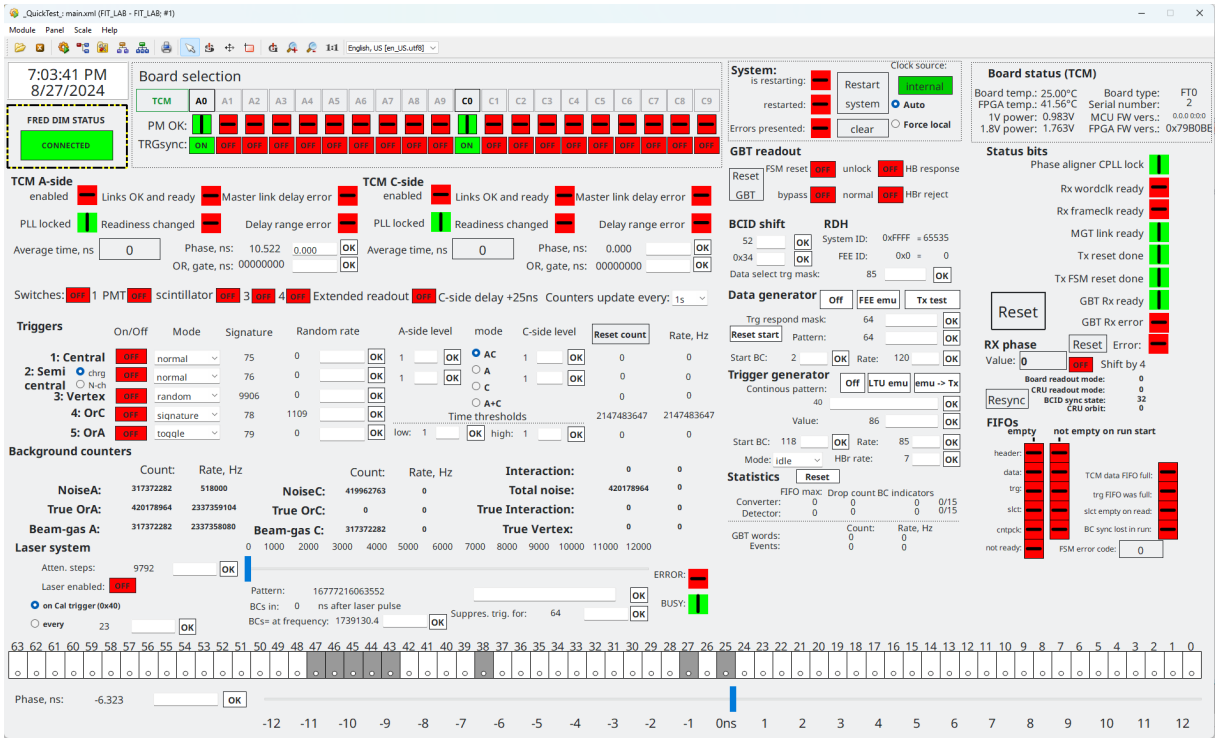
Rysunek 17. Poprawnie zaimportowane DIM serwisy.

4.6.3 Interfejs użytkownika stworzonego systemu SCADA

Na rysunku 18 zaprezentowano panel interfejsu użytkownika opracowanego systemu SCADA. Wykorzystując intuicyjne środowisko graficzne, operator ma możliwość kompleksowego monitorowania i sterowania parametrami modułów FEE.

Na panelu możemy wyróżnić następujące moduły:

- **Wyboru sterowanej płyty *Board selection***
  Służy do wyboru płyty FEE, którą chcemy sterować. Na ilustracji 18 wybrana została płyta TCM. W jego obrębie znajduje się również ikona wskazująca na aktualny status modułu. Zmienia ona swój stan w przypadku wystąpienia błędu. Możemy również włączyć synchronizowanie modułu PM z zegarem TCM poprzez przełącznik *TRGSync*.
- **Statusu połączenia protokołem DIM z warstwą FRED *DIM Status***
  Pozwala na sprawdzenie połączenia z serwerem DIM na warstwie FRED oraz monitorowanie statusu jego działania.
- **Zegara**
  Moduł wyświetla aktualną datę i godzinę. Gdy ekspert dyżurny on-call przystępuje do naprawy detektora, zaraz po uruchomieniu panelu, wykonuje zrzut ekranu. Dołączany jest on do raportu z przeprowadzonej interwencji, co umożliwia łatwe odczytanie parametrów, przy których doszło do awarii detektora.

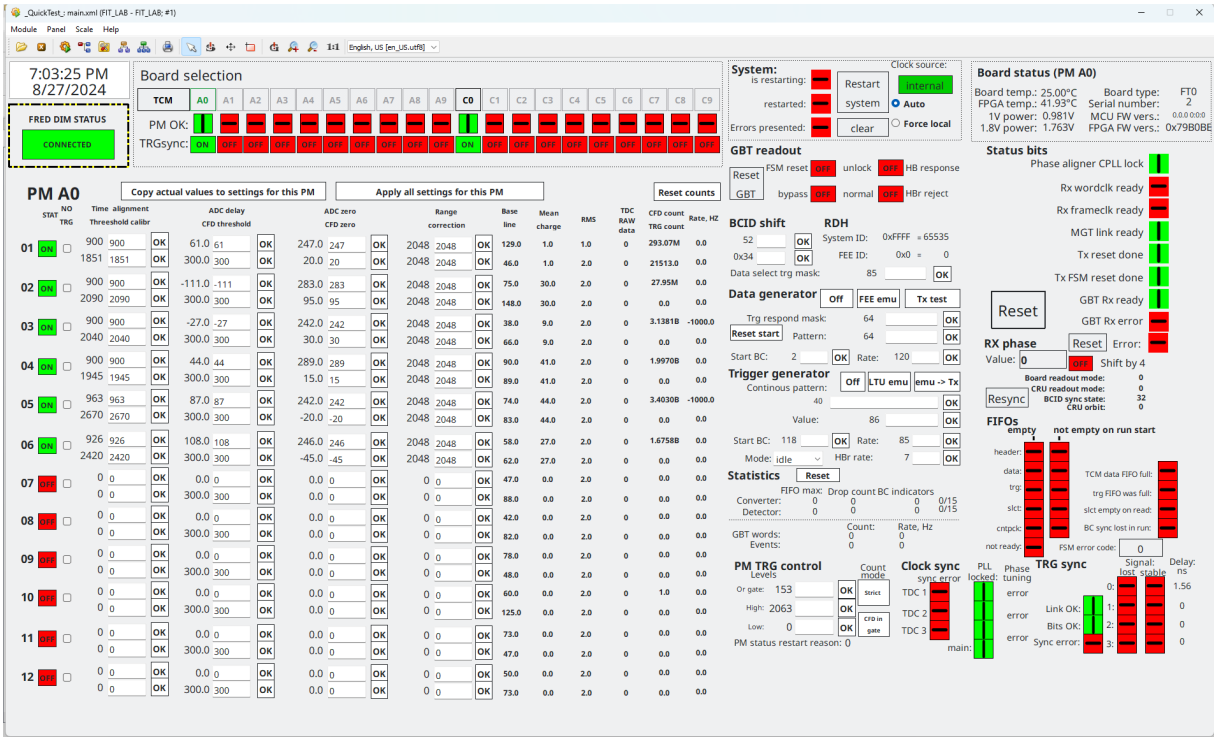


Rysunek 18. Panel opracowanego systemu SCADA, z wybraną opcją sterowania płytami TCM.

- **Ustawień i statusów TCM**
  Panel kontroluje statusy modułów TCM (np. status połączeń, synchronizację oraz błędy opóźnienia sygnału). Moduł wskazuje również, czy wszystkie połączenia są poprawne (status *Links OK and ready*). Dodatkowo umożliwia synchronizowanie z Central Trigger Processor (CTP).
- **Ustawień wyzwalania *Triggers***
  Służy do konfiguracji parametrów wyzwalania.
- **Liczników *Background counters***
  Moduł umożliwia monitorowanie liczników zdarzeń tła, które mogą być wykorzystane do analizy diagnostycznej oraz statystycznej.
- **Systemu laserowego *Laser system***
  Panel odpowiada za kontrolę i monitorowanie systemu laserowego, który wykorzystywany jest do kalibracji oraz testowania detektora.
- **Systemu sterowania detektorem *System***
  Umożliwia restartowanie systemu, zmianę źródła zegara (*Clock source*), a także wyświetla status procesu restartu, błędów oraz źródło zegara detektora.
- **Odczytu magistralą GBT *GBT readout***
  Moduł ten pozwala na zarządzanie magistralą GBT, umożliwiając resetowanie Finite State Machine (FSM), odblokowywanie odczytu oraz wyświetlanie błędów transmisji (*GBT Rx error*).

- **BCID oraz RDH**
  Odpowiada za zarządzanie numerami identyfikacyjnymi wiązek oraz nagłówkami danych. Ustawienia te pozwalają na synchronizację detektora z systemem akwizycji danych eksperymentu.
- **Generowania danych *Data generator***
  Sekcja umożliwia symulowanie danych wejściowych detektora w celu testowania jego funkcji.
- **Generowania wyzwalania *Trigger generator***
  Moduł ten umożliwia testowanie funkcji wyzwalania poprzez generowanie sygnałów wyzwalających w trybie ciągłym lub zdefiniowanym przez użytkownika wzorcem.
- **Statystyki *Statistics***
  Wyświetla dane statystyczne, takie jak liczba przetworzonych zdarzeń, liczba błędów bufora FIFO oraz inne kluczowe wskaźniki diagnostyczne.
- **Statusu płyty *Board status***
  Prezentuje fizyczne parametry płyty, takie jak temperatura, napięcie, numer seryjny, czy wersję oprogramowania. Pozwala to na szybkie sprawdzenie stanu technicznego modułu.
- **Statusu operacji *Status bits***
  Wyświetla kluczowe statusy operacyjne, takie jak fazy synchronizacji oraz błędy pracy.
- **Fazy *Rx phase***
  Umożliwia monitorowanie i regulację fazy sygnału, która może być zmieniana w celu synchronizacji sygnału z systemem CTP.
- **Statusu bufora *FIFOs***
  Panel monitoruje stan buforów wejścia-wyjścia (*FIFO*) w module TCM, wskazując, czy dane zostały poprawnie odebrane lub przesłane.

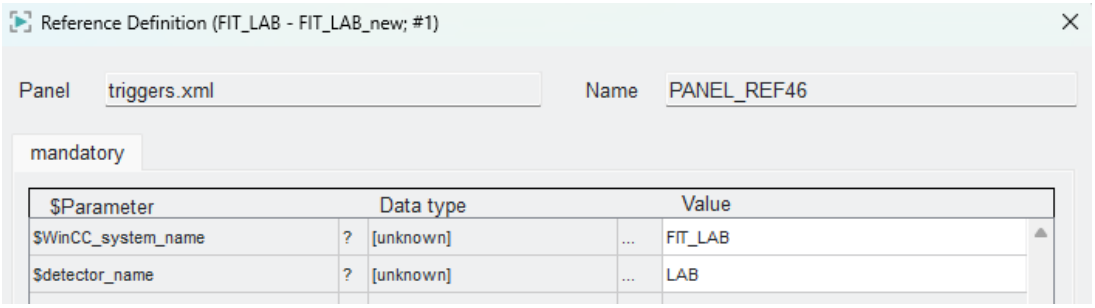
Na ilustracji 19 zaprezentowano wygląd panelu SCADA z wybranym modułem PMA0. Środkowa część panelu umożliwia ustawianie i obserwację parametrów jego wszystkich dwunastu kanałów. Prawa część panelu, choć wygląda bardzo podobnie do tej w module TCM, służy wyłącznie do obsługi wybranego modułu PM.



Rysunek 19. Panel opracowanego systemu SCADA, z wybraną opcją sterowania płytą PMA0.

Budowa interfejsu użytkownika

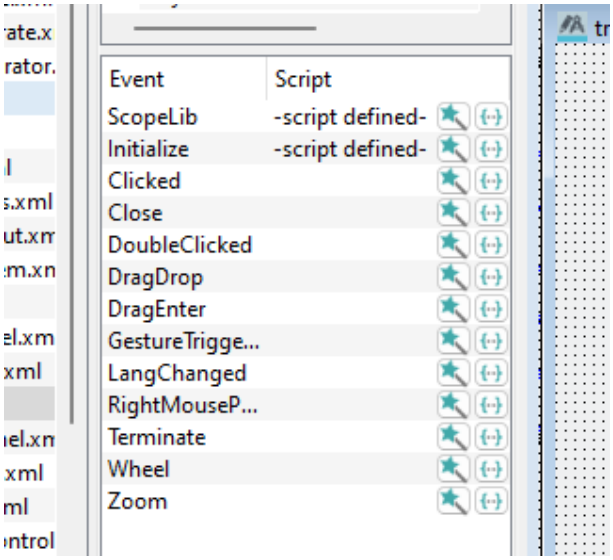
Panel składa jest z modułów, w które zaimplementowano możliwość parametryzacji. Dzięki wykorzystaniu zunifikowanych nazw DIM serwisów oraz komend, tworząc system do obsługi nowego detektora, nie ma potrzeby modyfikacji zawartego w nim kodu. W trybie deweloperskim podczas dodawania poszczególnych modułów pojawia się okno dialogowe, przedstawione na rysunku 20, w którym uzupełniamy parametry takie jak nazwa systemu SCADA oraz detektora.



Rysunek 20. Okno dialogowe parametryzacji panelu *triggers.xml*.

Po dodaniu modułu do panelu wyświetla się menu (rysunek 21), w którym można zaprogramować reakcje na różne zdarzenia. Podczas tworzenia systemów obrębie modułów wykorzystano dwie funkcje: *ScopeLib* oraz *Initialize*. *ScopeLib* jest miejscem, w którym można tworzyć funkcje wywoływane w obrębie poszczególnego modułu. Wykorzystywany jest do obsługi zdarzeń wywoływanych przez

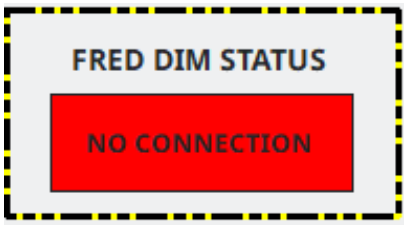
interakcję operatora z panelami. W przypadku, gdy użytkownik naciśnie dany przycisk zawarty w module, wywoływana jest w nim funkcja zdeklarowana właśnie w tym panelu. *Initialize* jest uruchamiana już podczas pierwszego włączenia modułu. Zawiera skrypty obsługujące DIM serwisy. Jeśli w jednym z tych serwisów nastąpi zmiana wartości, polecenie *dpConnect* w *Initialize* wywołuje funkcję, która automatycznie uaktualnia tę wartość na panelu użytkownika. Oprócz nich WinCC OA pozwala na wywoływanie funkcji po przykładowo naciśnięciu panelu, najechaniu na niego kursorem itd.



Rysunek 21. Menu programowania zdarzeń.

DIM status

Pierwszym przykładowym modułem systemu jest przedstawiony na rysunku 22 panel statusu połączenia DIM z jednostką FRED. Jego status uzależniony jest od sumy logicznej dwóch parametrów: statusu zmiennej JCOP wskazującej na status połączenia z serwerem Domain Name System (DNS) *FRED.ApiInfo.manState:_online.._value* oraz wartości serwisu kontrolnego wystawianego przez FRED *LAB/READOUTCARDS/TCM0/WORK_STATUS_ANS*, który wskazuje na jego poprawne działanie. W WinCC OA wykorzytując polecenia języka Control istnieje możliwość zmiany parametrów bloków graficznych umieszczonych na panelach.



Rysunek 22. Moduł statusu połączenia z serwerem FRED DIM.

Wykorzystano komendę *.text* (zmieniającą tekst obiektu) oraz *.color* (zmieniającą kolor obiektu). W przypadku nawiązania połączenia, ikona statusu zmienia kolor na zielony, a w ramce wyświetla się tekst *CONNECTED*. Jeśli połączenie zostanie zerwane, co dwie sekundy następuje automatyczna próba nawiązania ponownej komunikacji z serwerem DNS *server.dyndns.cern.ch*. Dodatkowo skrypt zawarty w panelu publikuje informacje o statusie w konsoli WinCC OA, której zawartość jest automatycznie zapisywana. Poniżej w listingu 6 zaprezentowano kod panelu umieszczony w zakładce *Initialize*.

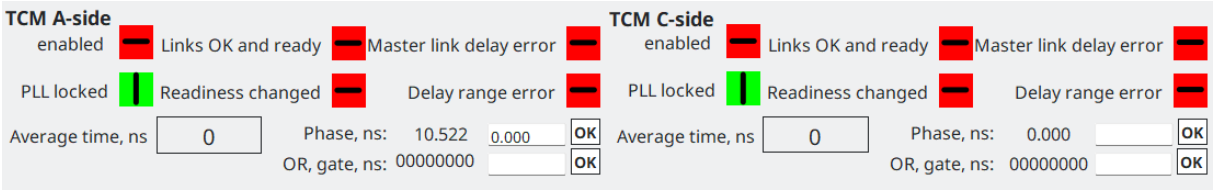
```
//deklaracja zmiennych lokalnych
int DIMstate, DIMstate_old, i, stat;
bit32 bits;
string DIMConfig="FRED";
string DIMdnsNode="server.dyndns.cern.ch";
string dpa, system_name,detector_name;

main()
{
//parametryzacja
  system_name = $WinCC_system_name;
  detector_name = $detector_name;
  dpa = system_name+":"+detector_name+".READOUTCARDS.TCM0.
   WORK_STATUS_ANS";
//wywolanie funkcji dp w przypadku zmiany wartosci serwisow
  dpConnect("dp",dpa);
  dpConnect("dp","FRED.ApiInfo.manState:_online.._value");
}
void dp(string statusDp, int statusValue)
{
//pobranie wartosci serwisow
dpGet(dpa,stat);
dpGet("FRED.ApiInfo.manState:_online.._value", DIMstate);
check();
}
//sprawdzenie statusu polaczenia i jego prezentacja
void check()
{
  if(DIMstate&&(stat==1)){
       dim_stat_text.text="CONNECTED";
       dim_status.color="green";
       DebugN("INFO: Successfully connected to the DIM FRED server");
    }
  else{
    dim_stat_text.text="NO CONNECTION";
    dim_status.color="red";
    DebugN("ERROR: No connection to the DIM FRED server");
  }
```

```
38  if (DIMstate==0){
39    fwDim_start(DIMConfig,2, DIMdnsNode);
40    DebugN("INFO: I am trying to reconnect with DIM FRED...");
41    delay(2);
42  }}
```

Listing 6. Kod panelu wyświetlającego status połączenia z serwerem DIM.

TCM status

Kolejnym opisywanym elementem jest przedstawiony na rysunku 23 panel statusów TCM. Ze względu na fakt, że detektor FT0, na którego systemie sterowania oparto stację laboratoryjną, składa się z dwóch modułów, system kontroli TCM został zdublowany.



Rysunek 23. Moduł kontroli TCM A-side oraz C-side.

W ogólności schemat działania skryptów obsługujących DIM serwisy wszystkich paneli można opisać w następujących krokach:

1. Deklaracja zmiennych lokalnych.
2. Deklaracja paramertryzacji panelu.
3. Wygenerowanie zmiennych zawierających nazwę właściwych DIM serwisów na podstawie nazwy systemu, nazwy detektora i nazwy własnej serwisu.
4. Wywołanie komendy *dpConnect*, która wyzwala funkcję wskazaną w parametrze w przypadku zmiany zawartości komówki w bazie danych przypisanej do DIM serwisu.
5. Funkcja wywołana przez `dpConnect` pobiera z bazy danych wartość wskazanej zmiennej, a następnie (w zależności od potrzeby) wykonuje bitową sumę odpowiednich bitów lub konwertuje ją bezpośrednio i wyświetla na ekranie.

W DIM serwisach często w obrębie jednej 32-bitowej ramki, przesyłanych jest wiele niezależnych danych. W takim przypadku przed przedstawieniem danej zmiennej dokonywana jest jej suma bitowa. Jako przykład można przytoczyć przedstawiony w tabeli 5 serwis *LAB/READOUTCARD-S/TCM0/AVERAGE_TIME_ANS*. Od zerowego do piętnastego bitu zawiera on wartość *Average time TCM A-side*, a od piętnastego do trzydziestego pierwszego *Average time TCM C-side*.

X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Average time A																Average time C															

Tabela 5. Zawartość serwisu AVERAGE_TIME_ANS.

Na poniższym listingu 7 przedstawiono kod odpowiedzialny za wyświetlanie danych w tym panelu.

```
//deklaracja zmiennych
string system_name,detector_name, tcm_a_phase1_dp, tcm_c_phase1_dp,
stats_dp,stat_a_dp,stat_c_dp,average_time_dp;
main()
{
//parametryzacja panelu
 system_name = $WinCC_system_name;
 detector_name = $detector_name;

//generowanie nazw DP
 tcm_a_phase1_dp = system_name+":"+detector_name+".READOUTCARDS.TCM0.
   DELAY_A_ANS";
 tcm_c_phase1_dp = system_name+":"+detector_name+".READOUTCARDS.TCM0.
   DELAY_C_ANS";
 stats_dp = system_name+":"+detector_name+".READOUTCARDS.TCM0.
   BOARD_STATUS_ANS";
 stat_a_dp=system_name+":"+detector_name+".READOUTCARDS.TCM0.
   SIDE_A_STATUS_ANS";
 stat_c_dp=system_name+":"+detector_name+".READOUTCARDS.TCM0.
   SIDE_C_STATUS_ANS";
 average_time_dp=system_name+":"+detector_name+".READOUTCARDS.TCM0.
   AVERAGE_TIME_ANS";
//wywolanie funkcji dp w przypadku zmiany wartosci serwisow
 dpConnect("stat_a",stat_a_dp);
 dpConnect("stat_c",stat_c_dp);
 dpConnect("stats",stats_dp);
 dpConnect("tcm_a_phase1",tcm_a_phase1_dp);
 dpConnect("tcm_c_phase1",tcm_c_phase1_dp);
 dpConnect("average_timef",average_time_dp);
}
//funkcja bloku average time
void average_timef(string statusDp, int statusValue)
{
    int temp, sum;
    dpGet(average_time_dp,temp);
    bit32 x=(bit32)temp;
        for (int i=0; i<=15;i++){
        if (getBit(x,i)==true){sum=sum+(pow(2,i));}}
    av_time_a.text=sum;
        for (int i=16; i<=31;i++){
```

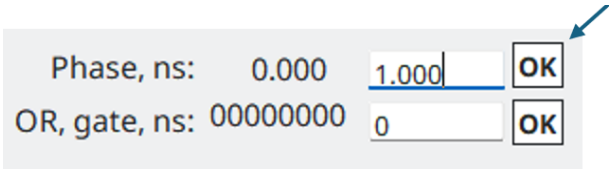
```
35          if (getBit(x,i)==true){sum=sum+(pow(2,i-16));}}
36      av_time_c.text=sum;
37  }
38  //wyswietlanie tcm_a_phase
39  void tcm_a_phase1(string statusDp, int statusValue){
40  string tcm_a_phase_f;
41  dpGet(tcm_a_phase1_dp,tcm_a_phase_f);
42  tcm_a_phase.text=tcm_a_phase_f;
43  }
44  //wyswietlanie tcm_c_phase
45  void tcm_c_phase1(string statusDp1, int statusValue1){
46  string tcm_c_phase_f;
47  dpGet(tcm_c_phase1_dp,tcm_c_phase_f);
48  tcm_c_phase.text=tcm_c_phase_f;
49  }
50  //wyswietlanie ikon statusow
51  void stats(string statusDp, int statusValue){
52  int tcm_int;
53  bit32 tcm_bits;
54  dpGet(stats_dp,tcm_int);
55  tcm_bits=(bit32)tcm_int;
56
57  //lock_c
58  if (getBit(tcm_bits,0)!=true) {tcmc_4.color="red"; tcmc_l4.visible=0;
        tcmc_n4.visible=1;}
59  if (getBit(tcm_bits,0)==true) {tcmc_n4.visible=0; tcmc_4.color="green";
        tcmc_l4.visible=1;}
60
61  //lock_a
62  if (getBit(tcm_bits,1)!=true) {tcma_4.color="red"; tcma_l4.visible=0;
        tcma_n4.visible=1;}
63  if (getBit(tcm_bits,1)==true) {tcma_n4.visible=0; tcma_4.color="green";
        tcma_l4.visible=1;}
64  }
65
66  void stat_a(string statusDp, int statusValue)
67  {
68  bit32 tcm_bits;
69  dpGet(stat_a_dp,tcm_bits);
70  //TCM_A SIDE
71  //master link delay error
72  if (getBit(tcm_bits,27)!=true) {tcma_3.color="red"; tcma_l3.visible=0;
        tcma_n3.visible=1;}
73  if (getBit(tcm_bits,27)==true)
74  {tcma_n3.visible=0; tcma_3.color="green";tcma_l3.visible=1;}
75  //side a enabled
```

```
76 if (getBit(tcm_bits,28)!=true) {tcma_1.color="red"; tcma_l1.visible=0;
       tcma_n1.visible=1;}
77 if (getBit(tcm_bits,28)==true)
78 {tcma_n1.visible=0; tcma_1.color="green";tcma_l1.visible=1;}
79 //delay range error
80 if (getBit(tcm_bits,29)!=true) {tcma_6.color="red"; tcma_l6.visible=0;
       tcma_n6.visible=1;}
81 if (getBit(tcm_bits,29)==true)
82 {tcma_n6.visible=0; tcma_6.color="green";tcma_l6.visible=1;}
83 //readiness changed
84 if (getBit(tcm_bits,30)!=true) {tcma_5.color="red"; tcma_l5.visible=0;
       tcma_n5.visible=1;}
85 if (getBit(tcm_bits,30)==true)
86 {tcma_n5.visible=0; tcma_5.color="green";tcma_l5.visible=1;}
87 //links and ready ok
88 if (getBit(tcm_bits,31)!=true) {tcma_2.color="red"; tcma_l2.visible=0;
       tcma_n2.visible=1;}
89 if (getBit(tcm_bits,31)==true)
90 {tcma_n2.visible=0; tcma_2.color="green";tcma_l2.visible=1;}
91 }
92
93 void stat_c(string statusDp, int statusValue)
94 {
95 bit32 tcm_bits;
96 dpGet(stat_c_dp,tcm_bits);
97 //TCM_C SIDE
98 //master link delay error
99 if (getBit(tcm_bits,27)!=true) {tcmc_3.color="red"; tcmc_l3.visible=0;
       tcmc_n3.visible=1;}
100 if (getBit(tcm_bits,27)==true)
101 {tcmc_n3.visible=0; tcmc_3.color="green";tcmc_l3.visible=1;}
102 //side a enabled
103 if (getBit(tcm_bits,28)!=true) {tcmc_1.color="red"; tcmc_l1.visible=0;
       tcmc_n1.visible=1;}
104 if (getBit(tcm_bits,28)==true)
105 {tcmc_n1.visible=0; tcmc_1.color="green";tcmc_l1.visible=1;}
106 //delay range error
107 if (getBit(tcm_bits,29)!=true) {tcmc_6.color="red"; tcmc_l6.visible=0;
       tcmc_n6.visible=1;}
108 if (getBit(tcm_bits,29)==true)
109 {tcmc_n6.visible=0; tcmc_6.color="green";tcmc_l6.visible=1;}
110 //readiness changed
111 if (getBit(tcm_bits,30)!=true) {tcmc_5.color="red"; tcmc_l5.visible=0;
       tcmc_n5.visible=1;}
112 if (getBit(tcm_bits,30)==true)
113 {tcmc_n5.visible=0; tcmc_5.color="green";tcmc_l5.visible=1;}
```

```
114 //links and ready ok
115 if (getBit(tcm_bits,31)!=true) {tcmc_2.color="red"; tcmc_l2.visible=0;
    tcmc_n2.visible=1;}
116 if (getBit(tcm_bits,31)==true)
117 {tcmc_n2.visible=0; tcmc_2.color="green";tcmc_l2.visible=1;}
118 }
```

Listing 7. Kod modułu kontroli TCM A-side oraz C-side.

W obrębie panelu zawarte są również cztery pola tekstowe umożliwiające wprowadzanie takich wartości jak *Phase (ns)* oraz *OR, gate (ns)*. Użytkownik systemu, po wpisaniu żądanej wartości do pola tekstowego, akceptuje ją, naciskając ikonę *OK* przedstawioną na rysunku 24. W jego wewnętrznej zakładce *Clicked*, wyzwalanej po naciśnięciu obszaru przycisku, znajduje się przedstawione w listingu 8 wywołanie funkcji *event* ze *ScopeLib* panelu, zaprezentowanej w listingu 9.



Rysunek 24. Przycisk akceptujący wprowadzone dane.

```
1 main(mapping event)
2 {
3   event(1);
4 }
```

Listing 8. Wyzwolenie funkcji event przycisku OK Phase TCM A-side.

```
1  public void event(int d)
2  {
3  string system_name,detector_name,ans,a_dp,c_dp, ora_dp,orc_dp;
4  system_name = $WinCC_system_name;
5  detector_name = $detector_name;
6  //generowanie nazw komend
7  a_dp=system_name+":"+detector_name+".READOUTCARDS.TCM0.DELAY_A_REQ";
8  c_dp=system_name+":"+detector_name+".READOUTCARDS.TCM0.DELAY_C_REQ";
9  ora_dp=system_name+":"+detector_name+".READOUTCARDS.TCM0.OR_A_REQ";
10 orc_dp=system_name+":"+detector_name+".READOUTCARDS.TCM0.OR_C_REQ";
11 string value;
12   if(d==1){value=phasea.text(); dpSet(a_dp,value+",1");}
13   if(d==2){value=phasec.text(); dpSet(c_dp,value+",1");}
14   if(d==3){value=ora.text(); dpSet(ora_dp,value+",1");}
15   if(d==4){value=orc.text(); dpSet(orc_dp,value+",1");}
16 }
```

Listing 9. Funkcja umieszczona w zakładce ScopeLib modułu *TCM.xml*.

W trakcie projektowania warstwy pośredniczącej FRED, ujednolicono format odbieranych DIM komend. Ustalono, że każda wysyłana ramka, będzie miała następującą strukturę:

- wartość,1 - zapis na całym rejestrze
- dowolna_wartosc,2 - zerowanie rejestru
- wartość,numer_bitu,3 - ustawianie pojedynczego bitu

W tym przypadku, każdy z rejestrów jest indywidualnie przypisany do danej wartości, więc wykorzystano następujący format zapisu: *dpSet(a_dp,value+",1");*.

Panel BCID shift

Jako przykład optymalizacji struktury kodu zawartego w systemie można zaprezentować panel BCID Shift. Panel ten występuje w takiej samej formie zarówno dla modułu TCM, jak i dwudziestu PM. Indywidualne tworzenie kodu, dla każdego z nich wiązałoby się ze znacznym obciążeniem systemu. Z uwagi na zunifikowaną strukturę DIM serwisów i komend, która jest taka sama zarówno dla PM oraz TCM, można było wykorzystać jeden panel do obsługi wszystkich urządzeń. Podczas wyboru sterowanego urządzenia w panelu *Board selection* DP *FIT_LAB:PM_index.PM_index* przyjmuje wartości od 0 do 9 dla PMA, 10-19 dla PMC oraz 100 dla TCM. Skrypt na tej podstawie automatycznie tworzy nazwę DIM serwisu, z którego pobiera dane. Część z wartości wyświetlanych w obrębie panelu prezentowanych jest w formacie heksadecymalnym. Wartości konwertowane są z użyciem struktury *sprintf(hexValue,"0x%X",sum)*. W listingu 10 zaprezentowano skrypt odpowiedzialny za wyświetlanie danych w opisanym panelu.

```
1  //deklaracja zmiennych
2  string system_name,detector_name, rdh_fields_dp,data_select_dp,
       bcid_offset_dp,pm_num_dp;
3  public void event(int d)
4  {
5  main()
6      {
7          //parametryzacja panelu
8          system_name = $WinCC_system_name;
9          detector_name = $detector_name;
10         //generowanie nazwy DP
11         pm_num_dp= system_name+":"+"PM_index.PM_index";
12         //pobranie wartosci DIM serwisu odpowiedzialnego za wskazanie
       wybranego panelu
13         dpConnect("boardTypef",pm_num_dp);
14     }
15 void boardTypef(string statusDp, int statusValue)
16 {
17   int x;
18   string temp_index,boardType;
19   dpGet(pm_num_dp,x);
20
```

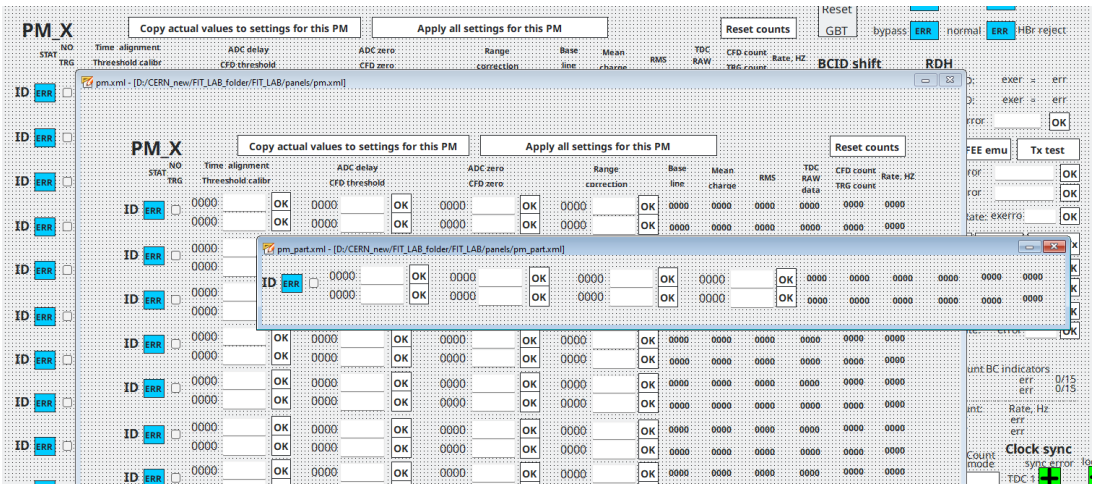
```
if (x<10)
  {temp_index="A"; boardType=".PM.PM"+temp_index+x+".";}
if (x>=10&&x<=20)
  {x=x%10; temp_index="C"; boardType=".PM.PM"+temp_index+x+".";}
if (x==100)
  {boardType=".READOUTCARDS.TCM0.";}
//generacja nazwy DP powiazanego z DIM komenda
rdh_fields_dp = system_name+":"+detector_name+boardType+"RDH_FIELDS_ANS"
    ;
data_select_dp = system_name+":"+detector_name+boardType+"
    DATA_SEL_TRG_MASK_ANS";
bcid_offset_dp = system_name+":"+detector_name+boardType+"
    BCID_OFFSET_ANS";
//wywolanie funkcji pobierajacych wartosci z DP w przypadku zmiany ich
    wartosci
 dpConnect("rdh_fields_dpf",rdh_fields_dp);
 dpConnect("data_select_dpf",data_select_dp);
 dpConnect("bcid_offset_dpf",bcid_offset_dp);
}
//wyswietlanie danych na panelach
//rdh systemid and fee id
void rdh_fields_dpf(string statusDp, int statusValue)
{
    string hexValue;
    int temp;
    int sum;
    dpGet(rdh_fields_dp,temp);
    bit32 x=(bit32)temp;

    for (int i=0; i<=15;i++)
    {if (getBit(x,i)==true){sum=sum+(pow(2,i));}}
    ri1.text=sum;
    sprintf(hexValue, "0x%X", sum);
    rh1.text=hexValue;
    sum=0;

    for (int i=16; i<=23;i++)
    {if (getBit(x,i)==true){sum=sum+(pow(2,i-16));}}
    ri2.text=sum;
    //konwersja na format heksadecymalny
    sprintf(hexValue, "0x%X", sum);
    rh2.text=hexValue;
}
```

```
63 //bcid offset
64 void bcid_offset_dpf(string statusDp, int statusValue)
65 {
66     string hexValue;
67     int temp;
68     long sum;
69     dpGet(bcid_offset_dp,temp);
70     bit32 x=(bit32)temp;
71     for (int i=0; i<=12;i++)
72     {if (getBit(x,i)==true){sum=sum+(pow(2,i));}}
73     b1.text=sum;
74     sprintf(hexValue, "0x%X", sum);
75     b2.text=hexValue;
76 }
77 //data select trg mask text
78 void data_select_dpf(string statusDp, int statusValue)
79 {
80     string temp;
81     dpGet(data_select_dp,temp);
82     ds1.text=temp;
83 }
```

Listing 10. Skrypt odpowiedzialny za wyświetlanie danych z płyt TCM oraz PM w panelu BCID shift.

Panel PM

Jako przykład optymalizacji struktury systemu można zaprezentować również przedstawiony na rysunku 25 panel PM. W ramach jego implementacji, ze względu na konieczność sterowania



Rysunek 25. Sparametryzowany moduł, który został powielony dwunastokrotnie w celu sterowania wszystkimi kanałami płyty PM.

dwunastoma identycznymi kanałami w obrębie jednego modułu, zastosowano podejście modularne,

polegające na zagnieżdżeniu w nim kolejnego modułu, który następnie powielono dwunastokrotnie. W ramach parametryzacji oprócz nazwy systemu i detektora, wprowadza się również numer kanału. Pozwala to na łatwe zarządzanie parametrami oraz wspiera skalowalność systemu. Dzięki takiemu podejściu kod jest bardziej czytelny, a ewentualne zmiany lub ulepszenia w strukturze jednego kanału automatycznie odzwierciedlają się w pozostałych.

Pozostałe panele w projekcie zrealizowano w analogiczny sposób do już opisanych. Wykorzystują one lokalnie działające skrypty, które interpretują wartości bitowe DIM serwisów, wyświetlając je na ekranie i wysyłają DIM komendy w oparciu o wartości ustawione przez użytkowników systemu SCADA. W celu ułatwienia sterowania podczas tworzenia interfejsu graficznego wykorzystywane zostały interaktywne przyciski, listy rozwijane, suwaki itd. Wszystkie panele zostały sparametryzowane, co umożliwia ich łatwe dostosowanie do sterowania innymi poddetektorami systemu FIT w eksperymencie ALICE.

Rozdział 5

Podsumowanie

Przedmiotem niniejszej pracy było opracowanie w oprogramowaniu WinCC OA systemu SCADA, stanowiącego integralną część przeprowadzanej modernizacji systemu sterowania poddetektorów FIT w eksperymencie ALICE. W ramach prowadzonych prac, we współpracy z zespołem obejmującym pracowników CERN, członków kolaboracji ALICE, a także pracowników i studentów Wydziału Fizyki oraz Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej, oraz Akademii Górniczo-Hutniczej w Krakowie, opracowano strukturę ALICE low level frontend and Flexible Framework for Frontend Electronics (ALF-FRED) wraz z opracowaną w całości przez autora pracy aplikacją SCADA do sterowania systemem. Prace odbywały się w lipcu i sierpniu 2024 roku w Europejskiej Organizacji Badań Jądrowych w Genewie. Projekt zrealizowano, tworząc środowisko mające funkcjonalność serwera kontroli, jednocześnie będące zgodnym ze standardami wykorzystywanymi w eksperymencie ALICE. W rezultacie przeprowadzonych badań, pod kierownictwem mgr. inż. Krystiana Rosłona opublikowano artykuł *Integrating IPbus ALFRED into the ALICE-FIT setup* [33]. Autor pracy będąc członkiem kolaboracji ALICE został współautorem publikacji.

Pierwszym miejscem, gdzie system został zastosowany jest stacja laboratoryjna poddetektorów FIT. Implementacja nowego systemu sterowania pozwoliła na stworzenie stacji treningowej dla ekspertów dyżurnych detektora. Dotychczas proces ich szkolenia przebiegał na podstawie krótkiego instruktażu, wzbogaconego instrukcją tekstową. Autor pracy miał okazję poznać specyfikę obowiązków eksperta dyżurnego, pełniąc jego funkcję przez 240 godzin w sierpniu 2024 roku. Dzięki przejściu, z systemu opartego na jednej jednostce, na taki oparty na warstwach odpowiedzialnych za komunikację z elektroniką ALF oraz warstwy tłumaczącej na jednostki fizyczne FRED, struktura systemu sterowania stała się klarowna. Pozwala na łatwe wyzwalanie błędów, które przyszły ekspert dyżurny może rozwiązywać w stworzonym systemie SCADA. Pozwoli to znacznie lepiej przygotować się do pełnienia tej funkcji. Planowane, jest również stworzenie skryptu w MatLab, który mógłby w sposób automatyczny symulować zachowanie elektroniki oraz generować problemy do rozwiązania. Jednocześnie struktura systemu jest walidowana i ulepszana. Docelowo stworzony system ma zostać wdrożony do sterowania rzeczywistym zespołem poddetektorów FIT, przyczyniając się do zwiększenia jego efektywności i niezawodności.

Bibliografia

[1] CERN, *About CERN*, Uzyskano dostęp: 2025-01-06, 2024. adr.: https://home.cern/about.

[2] CERN, *CERN International Relations*, Uzyskano dostęp: 2025-01-06, 2024. adr.: https://international-relations.web.cern.ch/stakeholder-relations/states/poland.

[3] William Moebs Samuel J. Ling, J. S., *Fizyka dla szkół wyższych. Tom 3*, 3 wyd. OpenStax, 2018. adr.: https://d3bxy9euw4e147.cloudfront.net/oscms-prodcms/media/documents/Fizyka_dla_szko%C5%82_wyzszych_Tom_3_v3.29_hq.pdf.

[4] CERN, *Accelerators*, Uzyskano dostęp: 2025-01-06, 2025. adr.: https://home.cern/science/accelerators.

[5] CERN, *Fundamental reaserch*, Uzyskano dostęp: 2025-01-06, 2025. adr.: https://home.cern/about/what-we-do/fundamental-research.

[6] CERN, *Higgs boson*, Uzyskano dostęp: 2025-01-06, 2025. adr.: https://home.cern/science/physics/higgs-boson.

[7] CERN, *History of WWW*, Uzyskano dostęp: 2025-01-06, 2025. adr.: https://home.cern/science/computing/birth-web/short-history-web.

[8] International Atomic Energy Agency (IAEA), *What are Particle Accelerators?* Uzyskano dostęp: 15-01-2025, 2023. adr.: https://www.iaea.org/newscenter/news/what-are-particle-accelerators.

[9] CERN, *The Large Hadron Collider*, Uzyskano dostęp: 2024-12-19, 2024. adr.: https://home.cern/science/accelerators/large-hadron-collider.

[10] Vretenar, M. i in., *Linac4 Design Report* (CERN Yellow Reports: Monographs), Vretenar, M., red. Geneva, Switzerland: CERN, 2020, t. CERN-2020-006, ISBN: 978-92-9083-579-0. DOI: 10.23731/CYRM-2020-006. adr.: https://doi.org/10.23731/CYRM-2020-006.

[11] Hermes, P. D., „Heavy-Ion Collimation at the Large Hadron Collider: Simulations and Measurements," Munster U., 2016. adr.: https://cds.cern.ch/record/2241364.

[12] Europejska Organizacja Badań Jądrowych (CERN), *LHC – Wielki Zderzacz Hadronów (polski podręcznik)*, Uzyskano dostęp: 2025-01-06, 2016. adr.: https://ppss.ifj.edu.pl/materials_2016/LHC-booklet-Polish.pdf.

[13] CERN, *CERN's accelerator complex*, Uzyskano dostęp: 2025-01-07, 2025. adr.: https://home.cern/science/accelerators/accelerator-complex.

[14] CERN. „Heavy-Ion Run of the LHC Begins." Uzyskano dostęp: 2025-01-16. (2024), adr.: https://home.cern/news/news/experiments/heavy-ion-run-lhc-begins.

[15] Collaboration, T. A., „The ALICE experiment at the CERN LHC," *Journal of Instrumentation (JINST)*, t. 3, S08002, 2008. DOI: 10.1088/1748-0221/3/08/S08002. adr.: https://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08002/pdf.

[16] Pluta, J., *Akceleratory cząstek - Klasyfikacja cząstek*, Uzyskano dostęp: 2025-01-07, 2025. adr.: https://www.if.pw.edu.pl/~pluta/pl/dyd/lekcje/lekcja15/segment4/main.htm.

[17] Grebieszkow, K., „Relatywistyczne zderzenia ciężkich jonów jako narzędzie w badaniu diagramu fazowego silnie oddziałującej materii," *Politechnika Warszawska*, 2024. adr.: https://pti.fizyka.pw.edu.pl/HIC.pdf.

[18] Ministerstwo Edukacji i Nauki. „Zintegrowana platrofma edukacyjna." Uzyskano dostęp: 2025-01-16. (), adr.: https://zpe.gov.pl/a/przeczytaj/DGHDMZNN1.

[19] Barabasz, A., *Cząstki elementarne modelu standardowego*, Uzyskano dostęp: 2025-01-07, 2012. adr.: https://pl.wikipedia.org/wiki/Model_standardowy#/media/Plik:Czastki_elementarne_modelu_standardowego.svg.

[20] CERN ALICE Collaboration, *ALICE Detector schematic*, Uzyskano dostęp: 2024-12-20, 2024. adr.: https://ep-news.web.cern.ch/sites/default/files/inline-images/ALICE-detector-ITS.png.

[21] Collaboration, A., „Technical Design Report for the Upgrade of the ALICE Inner Tracking System," CERN, spraw. tech. ALICE-TDR-015, 2014. adr.: https://cds.cern.ch/record/1603472/files/ALICE-TDR-015.pdf.

[22] ALICE Collaboration, „Upgrade of the Online - Offline computing system: Technical Design Report," CERN, Technical Design Report ALICE-TDR-019, 2015. adr.: https://cds.cern.ch/record/2011297/files/ALICE-TDR-019.pdf.

[23] CERN, *ATLAS Experiment*, Uzyskano dostęp: 2025-01-07, 2025. adr.: https://www.home.cern/science/experiments/atlas.

[24] Collaboration, T. A., „The ATLAS Experiment at the CERN Large Hadron Collider," *Journal of Instrumentation*, t. 3, nr. 08, S08003, 2008. DOI: 10.1088/1748-0221/3/08/S08003. adr.: https://dx.doi.org/10.1088/1748-0221/3/08/S08003.

[25] May, A., *What is the ATLAS experiment?* Uzyskano dostęp: 2025-01-07, 2022. adr.: https://www.livescience.com/cern-atlas-experiment.

[26] The ATLAS Collaboration, „A detailed map of Higgs boson interactions by the ATLAS experiment ten years after the discovery," *Nature*, t. 607, nr. 7917, s. 52–58, 2022. DOI: 10.1038/s41586-022-04893-w. adr.: https://doi.org/10.1038/s41586-022-04893-w.

[27] Chatrchyan, S. i in., „The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment," *JINST*, t. 3, S08004, 2008, Also published by CERN Geneva in 2010. DOI: 10.1088/1748-0221/3/08/S08004. adr.: https://cds.cern.ch/record/1129810.

[28] CERN, *The CMS Experiment*, Uzyskano dostęp: 2024-12-19, 2024. adr.: https://cms.cern/detector.

[29] Antunes-Nobrega, R. i in., *LHCb reoptimized detector design and performance: Technical Design Report* (Technical design report. LHCb). Geneva: CERN, 2003. adr.: https://cds.cern.ch/record/630827.

[30] Władysław Henryk Trzaska. „New ALICE Fast Interaction Trigger." Uzyskano dostęp: 2024-12-20. (2023), adr.: https://ep-news.web.cern.ch/content/new-alice-fast-interaction-trigger.

[31] Herr, W. i Muratori, B., „Concept of luminosity," *CERN*, 2006. DOI: 10.5170/CERN-2006-002.361. adr.: https://cds.cern.ch/record/941318.

[32] Slupecki, M., „Fast Interaction Trigger for ALICE upgrade," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, t. 1039, s. 167 021, 2022, ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.167021. adr.: https://www.sciencedirect.com/science/article/pii/S0168900222004466.

[33] Roslon, K., „Integrating IPbus ALFRED into the ALICE-FIT setup," Politechnika Warszawska, spraw. tech., 2025. arXiv: 2501.04685. adr.: https://cds.cern.ch/record/2921236.

[34] Lesiak, T., „Fizyka ciężkich barionow przy pomocy LEP," Instytut Fizyki Jądrowej im. Henryka Niewodniczańskiego w Krakowie, spraw. tech. INP-1848/PH, 2002. adr.: https://www.osti.gov/etdeweb/servlets/purl/20130917.

[35] Melikyan, Y. i in., „Characteristic properties of Planacon MCP-PMTs," *Journal of Instrumentation*, t. 16, nr. 12, P12032, 2021. DOI: 10.1088/1748-0221/16/12/P12032. adr.: https://dx.doi.org/10.1088/1748-0221/16/12/P12032.

[36] Tsoulfanidis, N., *Measurement and Detection of Radiation*, 2nd. Taylor & Francis, 1995. adr.: https://faculty.kfupm.edu.sa/phys/aanaqvi/NaqEmploy_files/Nicholas_Tsoulfanidis-2-Ed.pdf.

[37] Ramirez, S. A. R., „Design of the Forward Diffractive Detector's Control System for CERN-LHC Run 3," CERN-THESIS-2023-081, prac. dokt., Popular University of Puebla, 2023. adr.: https://cds.cern.ch/record/2862850.

[38] Frazier, R. i in., *IPbus Protocol Specification, Version 2.0*, CERN, 2014. adr.: https://ipbus.web.cern.ch/doc/user/html/_downloads/d251e03ea4badd71f62cffb24f110cfa/ipbus_protocol_v2_0.pdf.

[39] Krivda, M., „The ALICE Trigger Electronics," adr.: https://mkrivda.web.cern.ch/The%20Alice%20trigger%20electronics.pdf.

[40] The ALICE Collaboration, „Technical Design Report for the Upgrade of the Online–Offline Computing System," CERN, Geneva, Switzerland, Technical Design Report ALICE-TDR-019, 2015, Available under Creative Commons Attribution License (CC-BY-3.0). adr.: https://cds.cern.ch/record/2011297.

[41] Mitra, J., Khan, S., Mukherjee, S. i Paul, R., „Common Readout Unit (CRU) - A new readout architecture for the ALICE experiment," *Journal of Instrumentation*, t. 11, nr. 03, s. C03021, 2016. DOI: 10.1088/1748-0221/11/03/C03021. adr.: https://dx.doi.org/10.1088/1748-0221/11/03/C03021.

[42] Barroso, V. i in., „The new ALICE data acquisition system (O^2/FLP) for LHC Run 3," *EPJ Web Conf.*, t. 295, s. 02 029, 2024. DOI: 10.1051/epjconf/202429502029. adr.: https://cds.cern.ch/record/2919265.

[43] Antón, S. D., Fraunholz, D., Lipps, C., Pohl, F., Zimmermann, M. i Schotten, H. D., „Two decades of SCADA exploitation: A brief history," w *2017 IEEE Conference on Application, Information and Network Security (AINS)*, 2017, s. 98–104. DOI: 10.1109/AINS.2017.8270432. adr.: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8270432.

[44] Eric Newman. „The History and Future of SCADA." Uzyskano dostęp: 26-12-2024. (), adr.: https://www.racoman.com/blog/history-and-future-of-scada?srsltid=AfmBOooVCY_ztAFmOYODHTjHT4Ax95ANtIBE4p9PBIMv0RCMnV0mgZe4.

[45] Process Solutions. „A brief history of the SCADA system." Uzyskano dostęp: 26-12-2024. (), adr.: https://processsolutions.com/a-brief-history-of-the-scada-system.

[46] CERN. „PVSS Introduction." Uzyskano dostęp: 2025-01-17. (2025), adr.: https://lhcb-online.web.cern.ch/ecs/pvssintro.htm.

[47] AG, S., *WinCC OA: Getting Started Guide*, Uzyskano dostęp: 27-12-2024. adr.: https://www.winccoa.com/documentation/WinCCOA/latest/en_US/GettingStarted/GettingStarted-79.html.

[48] AG, S., *WinCC OA: Documentation*, Uzyskano dostęp: 27-12-2024. adr.: https://www.winccoa.com/documentation/WinCCOA/3.18/en_US/GettingStarted/GettingStarted-06.html.

[49] Holme, O., González-Berges, M., Golonka, P. i Schmeling, S., „The JCOP Framework," CERN, Geneva, spraw. tech., 2005. adr.: https://cds.cern.ch/record/907906.

[50] Ludwig, M., Arroyo Garcia, J., Bengulescu, M., Farnham, B., Gonzalez Jimenez, P. i Varela, F., „Developing and Validating OPC-UA Based Industrial Controls for Power Supplies at CERN," WEP04, 2019. DOI: 10.18429/JACoW-PCaPAC2018-WEP04. adr.: https://cds.cern.ch/record/2836174.

[51] Copy, B., Mandilara, E., Prieto Barreiro, I. i Varela, F., „Monitoring of CERN's Data Interchange Protocol (DIP) system," THPHA162, 2018. DOI: 10.18429/JACoW-ICALEPCS2017-THPHA162. adr.: https://cds.cern.ch/record/2305320.

[52] CERN, *Distributed Information Management (DIM): Introduction*. adr.: https://dim.web.cern.ch/dim_intro.html.

[53] Algorytm.edu.pl. „Sortowanie bąbelkowe — Algorytmy maturalne.” Uzyskano dostęp: 28-12-2024. (), adr.: https://www.algorytm.edu.pl/algorytmy-maturalne/sortowanie-babelkowe.html.

Wykaz skrótów i symboli

ALF ALICE low level frontend 20, 32

ALF-FRED ALICE low level frontend and Flexible Framework for Frontend Electronics 55

ALICE A Large Ion Collider Experiment 5, 14, 23, 31, 54, 55

ANSI C C Programming Language 29

ATLAS A Toroidal LHC Apparatus 14–16

CMS Compact Muon Solenoid 14, 16

CRU Common Readout Unit 25

CTP Central Trigger Processor 41, 42

DIM Distributed Information Management 5, 20, 31–33, 35, 40, 43, 46, 51

DNS Domain Name System 44, 45

DP Data Point 29, 51

ECAL Electromagnetic Calorimeter 16

EV Event Manager 28, 29

FDD Forward Diffractive Detector 18

FEE Front-end electronics 5, 20, 25, 40

FIT The Fast Interaction Trigger 5, 18, 21, 24, 54, 55

FLP First Level Processor 25

FRED Flexible Framework for Frontend Electronics 20, 23, 32, 33, 40, 44, 51, 55, 67

FSM Finite State Machine 41

FT0 Forward Time Zero 18, 46

FV0 Forward V0 18

GBT Gigabit Transceiver 25, 41

HCAL Hadronic Calorimeter 16

JCOP Joint Controls Project Framework 5, 29, 30, 32–34, 36, 39, 44

LAN Local Area Network 28

LEIR Low Energy Ion Ring 12

LHC Large Hadron Collider 12, 15, 23, 24, 26

LHCb Large Hadron Collider beauty experiment 14, 16

LINAC3 Linear Accelerator 3 12

LINAC4 Linear Accelerator 4 12

LTU Local Trigger Unit 24

OPC UA OPC Unified Architecture 20

PCIe PCI Express 25

PM Processing Module 5, 24, 40, 51

PS Proton Synchrotron 12

PSB Proton Synchrotron Booster 12

SCADA Supervisory Control And Data Acquisition 5, 20, 27, 28, 33, 43, 55

SPS Super Proton Synchrotron 12

SQL Structured Query Language 29

TCM Trigger and Clock Module 5, 24, 40–42, 46, 51, 67

UDP User Datagram Protocol 20

WinCC OA SIMATIC WinCC Open Architecture Version 3.19 5, 20, 28–30, 32, 33, 44, 45, 55, 67

WWW World Wide Web 11

$\frac{dR}{dt}$ Zmienność szybkości reakcji w czasie. 17

$\mathcal{L}$ Świetlność detektora. 17

σ_p Przekrój czynny procesu. 17

Spis rysunków

Spis tabel