AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Wizualizacja danych eksperymentalnych w symulowanym środowisku linii pomiarowej w synchrotronie

Michał Ciszewski, Łukasz Dudek, Krystian Mucha

Na przedmiot: Wielopoziomowe struktury sterowania i systemy SCADA

Prowadzący: prof. dr hab inż. Witold Bryski, mgr. inż. Andrzej Latocha

Akademia Górniczo - Hutnicza w Krakowie

10 stycznia 2016 r.

Streszczenie

Niniejszy raport podsumowuje prace nad projektem na przedmiot „Wielopoziomowe struktury sterowania i systemy SCADA" na Akademii Górniczo - Hutniczej im. S. Staszica w Krakowie. Dotyczy przygotowania dwóch sposobów wizualizacji sterowania zespołami silników w symulowanym środowisku linii badawczej w synchrotronie - aplikacji eksperckiej i klienckiej. Zostały omówione podstawy teoretyczne, środowisko programowo - sprzętowe i jego konfiguracja oraz sama realizacja projektu. W ostatnim rozdziale znajduje się instrukcja użytkowania przygotowanych aplikacji.

Słowa kluczowe: SCADA, zespoły silników, Tango Controls, Sardana, Taurus.

Spis treści

I. Podstawa teoretyczna

Systemy SCADA (ang. Supervisory control and data acquisition) to system do zdalnego monitorowania i sterowania procesem techologicznym. Głównymi funkcjami systemów SCADA są:

- zbieranie aktualnych danych (pomiarów)
- wizualizacja danych
- sterowanie procesem
- alarmowanie
- archiwizacja danych

Systemy te dają możliwość współpracy ze sterownikami PLC, regulatorami mikroprocesorowymi i innymi urządzeniami. Pozwalają na realizację zdecentralizowanych systemów automatyki przemysłowej. Systemy SCADA pozwalają na uzyskanie szybkiego wglądu w faktyczny stan urządzeń produkcyjnych i wykonawczych. Są one doskonałym sposobem nie tylko na zamianę języka maszyn na język ludzi, ale także umożliwiają szybką lokalizację alarmów, podstawowe logowanie danych czy też automatyczną reakcję na określone sygnały pochodzące z urządzeń. System SCADA w warstwie graficznej odpowiada za jednoznaczne zaprezentowanie dynamicznie zmieniającej się informacji. Jednocześnie zdefiniowane przez użytkownika algorytmy logiczne przyspieszają i wspomagają operatora w jego pracy. System SCADA jest także podstawowym źródłem danych dla systemów nadrzędnych i przemysłowych baz danych.

II. Koncepcja realizacji

Postawiony w temacie projektu problem jest dwojakiej natury. Po pierwsze, należy zdefiniować system sterowania linii badawczej w synchrotronie, określić w nim rolę silników i zaproponować programowe sposoby wizualizacji ich pracy. Po drugie, jak było wspomniane w początkowym opisie projektu, trzeba zastanowić się nad możliwościami synchronizacji pracy poszczególnych osi takich silników krokowych, które, jak było wspomniane w rozdziale I, ruszają wieloma elementami na liniach badawczych.

Należy również uwzględnić fakt, iż w normalnym środowisku pracy przy linii badawczej w czasie jej pracy mogą nią sterować dwie osoby. Jedna z nich to operator - klient synchrotronu, który przychodzi zrealizować pewne badania na danej linii. Drugą osobą jest ekspert ze strony synchrotronu - nie ma on obowiązku posiadania pełnej znajomości fizyki stojącej za funkcjonowaniem linii, ale powinien dobrze znać jej system sterowania, ponieważ może on mieć wpływ na serce synchrotronu, czyli pierścień akumulacyjny.

Dodatkowo, co jest dość oczywiste, przy tworzeniu projektu autorzy nie mieli dostępu do prawdziwej linii badawczej. W związku z tym wykorzystano środowisko symulacyjne, które jest dostarczane razem z narzędziami do zarządzania nim. Umożliwiło to dokładną konfigurację oraz testowanie różnych zestawów parametrów.

Na podstawie wyżej wymienionych zagadnień sformułowane zostały wymagania stawiane przed projektem:

1. Należy skonfigurować różne urządzenia symulujące pracę rzeczywistych silników z uwzględnieniem takich parametrów, jak prędkość, przyspieszenie lub położenie czujników krańcowych. Liczba takich symulacyjnych napędów powinna wynosić minimum 3.

2. Należy znaleźć sposób synchronizacji przynajmniej dwóch z tych silników. Przez synchronizację rozumie się wykonywanie zaplanowanego ruchu w jednostajny sposób i w jednym czasie. Bez względu na środowisko symulacyjne, zaleca się zastosowanie rozwiązania czysto programowe. Jest to związane z faktem, iż używany zestaw narzędzi do zarządzania systemem sterowania jest obiektowy, co pozwala na hierarchiczność urządzeń. W ten sposób można tworzyć urządzenia nadrzędne, które będą kontrolować rzeczywiste napędy.

3. Należy zaproponować dwa odrębne sposoby wizualizacji danych eksperymentalnych, czyli w tym przypadku pozycji symulowanych silników.

4. Należy przygotować dwie aplikacje: dla eksperta i zwykłego użytkownika. Pierwsza powinna dawać pełną kontrolę nad wszystkimi parametrami, za wyjątkiem stałych parametrów rzeczywistych silników (takich jak prędkość czy przyspieszenie). Druga powinna umożliwiać tylko podgląd stanów poszczególnych silników oraz wprowadzanie drobnych korekt w ich pozycjach.

Taki zestaw założeń projektowych został przyjęty w czasie jego realizacji.

III. Specyfikacja programowo-sprzętowa

I. Sprzęt

W przypadku tak stricte programowego projektu, jak opisywany w niniejszym raporcie ciężko jest wyróżnić jakąś szczególną specyfikację potrzebnego sprzętu. Do realizacji wykorzystywaliśmy tylko komputer klasy PC z oprogramowaniem opisanym w podrozdziale II.

To, o czym jednak należy tutaj wspomnieć, to używane zwykle w synchrotronach zestawy silników krokowych, które mają specyficzne możliwości konfiguracyjne oraz mogą być prosto zintegrowane z rozproszonym systemem sterowania zarządzanym przy pomocy narzędzi Tango. Najczęściej stosowanym rozwiązaniem jest platforma programowo - sprzętowa IcePAP [4, 5].

II. Platformy programowe

1. **Tango**

   System sterowania TANGO jest darmowym zestawem narzędzi do sterowania dowolnego rodzaju sprzętem lub oprogramowaniem oraz do budowy systemów SCADA (ang. Supervisory Control And Data Acquisition). Jest to oprogramowanie typu open-source wykorzystywane do sterowania synchrotronami, laserami oraz innymi eksperymentami fizycznymi. TANGO jest aktywnie rozwijane przez TANGO Consortium.

   TANGO jest rozproszonym systemem sterowania. Oznacza to, że może działać zarówno na jednej maszynie, jak i na wielu. Wykorzystuje dwa protokoły sieciowe - COBRA oraz Zeromq. Podstawowym modelem komunikacji jest model klient-serwer. Komunikacja pomiędzy klientem i serwerem może być synchroniczna, asynchroniczna (COBRA) oraz sterowana zdarzeniem (Zeromq) [6].

   System TANGO jest opakowaniem (ang. wrapper) dla protokołu COBRA zapewniającym przyjazne dla użytkownika API. Dzięki takiemu podejsciu wiele detali związanych z nawiązywaniem i utrzymywaniem połączenia pomiędzy urządzeniami jest niewidoczna dla użytkownia, co pozwala na szybsze i łatwiejsze rozbudowywanie systemu sterowania.

   TANGO wykorzystuje MySQL jako bazę danych do trzymania informacji. Informacjami takimi mogą byc np. nazwy urządzeń, adresy sieciowe, listy urządzeń itp. MySQL jest relacyjną bazą danych implementującą podzbiór SQLa. System TANGO jest wspierany przez 4 platformy: Linux, Windows NT, Solaris oraz HP-UX.

   (a) **Taurus**

   Taurus jest platformą programistyczną dla Pythona służącą do sterowania oraz akwizycji danych w zastosowaniach naukowych oraz przemysłowych. Pozwala na szybkie i proste tworzenie interfejsów użytkownika. Jest to częśc pakietu programistycznego Sardana. Taurus posiada bogatą bibliotekę dzięki czemu stworzone GUI (ang. Graphical User Interface) może zawierać wiele różnych komponentów takich jak wykresy, tabele, przyciski itp. Celem biblioteki Taurus jest zapewnienie przyjaznego API dla użytkownika oraz przyspieszenie procesu rozwijania aplikacji bazowanych na TANGO.

   (b) **Sardana**

   Sardana to pakiet oprogramowania do nadzoru, kontroli i akwizycja danych w zastosowaniach naukowych. Jej celem jest zredukowanie kosztów oraz czasu potrzebnych do projektowania, rozwijania oraz utrzymywania systemów SCADA. Rozwój

Sardany zapoczątkowany został przy synchrotronie ALBA, a dzisiaj jest wspierany przez wiekszą społeczność, w skład której wchodzą liczne laboratoria oraz inne jednostki (ALBA, DESY, MaxIV, Solaris, ESRF). Sardana jest bazowana na systemie TANGO i wykorzystuje bibliotekę Taurus umożliwiającą programowanie i konfigurację interfejsu użytkownika [7].

(c) **TangoBox9** (maszyna wirtualna)

W związku z polityką udostępniania oprogramowania na otwartych licencjach, konsorcjum Tango wypuściło gotową wirtualną maszynę, która zawiera wszystkie narzędzia pakietu Tango zainstalowane na systemie Ubuntu 14.04 64-bit. Są tam uwzględnione również wszystkie poboczne projekty, które ułatwiają korzystanie z systemu i dostarczają dodatkowych opcji potrzebnych systemowi sterowania, np. archiwizacji, logowania czy tworzenia interfejsów użytkownika. W tym systemie dostarczono w pełni funkcjonalne symulacyjne środowisko systemu sterowania synchrotronem z uwzględnieniem jednej linii badawczej. Obecne są tam również podstawowe aplikacje będące częścią bazowej paczki Tango, takie jak: Astor (służy do zarządzania serwerami urządzeń), Pogo (generator kodu serwerów), Jive czy AtkMoni (umożliwiają konfigurację i testowanie samych urządzeń).

2. **Python**

Python to wysokopoziomowy język programowania ogólnego przeznaczenia. Posiada rozbudowane pakiety bibliotek standardowych. Ideą przewodnią Pythona jest czytelność i klarowność kodu źródłowego. Jego składnia cechuje się przejrzystością i zwięzłością. W Pythonie możliwe jest programowanie obiektowe, programowanie strukturalne i programowanie funkcyjne. Typy sprawdzane są dynamicznie, a do zarządzania pamięcią stosuje się garbage collection. Python rozprowadzany jest na otwartej licencji umożliwiając także zastosowanie do zamkniętych komercyjnych projektów [8].

(a) **PyCharm**

PyCharm to zintegrowane środowisko programistyczne (IDE) dla języku programowania Python firmy JetBrains. Zapewnia m.in.: edycję i analizę kodu źródłowego, graficzny debugger, uruchamianie testów jednostkowych, integrację z systemem kontroli wersji. Wspiera także programowanie i tworzenie aplikacji internetowych w Django.

Jest oprogramowaniem wieloplatformowym pracującym na platformach systemowych: Microsoft Windows, Linux oraz OS X. Wydawany jest w wersji Professional Edition, które jest oprogramowaniem zamkniętym oraz w wersji darmowej Community Edition, które pozbawione jest jednak niektórych funkcjonalności w porównaniu z wersją komercyjną [9].

3. QtDesigner

IV. Opis realizacji

Realizacja zadań projektowych opisanych w rozdziale II została przeprowadzona przy użyciu wszystkich technologii, które są wymienione w rozdziale III. Można ją podzielić na trzy główne części, które zostaną przedstawione w kolejnych podrozdziałach.

I. Konfiguracja wirtualnego środowiska

Pierwszym elementem, który należało opanować było uruchomienie wirtualnej maszyny „TangoBox9", a następnie konfiguracja wszystkich elementów niezbędnych do realizacji postawionego celu. Kolejne kroki postępowania zostały przedstawione poniżej:

1. Sprawdzenie funkcjonowania systemu Tango: głównej bazy danych, poszczególnych „serwerów urządzeń" (ang. device servers) oraz samych urządzeń. W tym celu użyto dwóch aplikacji stanowiących pakiet do zarządzania systemem Tango: Astor i Jive. Weryfikacja poprawności polegała na uruchomieniu obu aplikacji i wizualnym sprawdzeniu stanów interesujących z punktu widzenia projektu komponentów systemu.

2. Sprawdzenie funkcjonowania dodatków do systemu Tango: archiwizacji, bibliotek Taurus i Sardana. Należało uruchomić odpowiednie aplikacje („jhdbconfigurator" w przypadku archiwizacji, „taurusgui" - Taurusa i „Sardemo" - Sardany) i sprawdzić, czy nie wyrzucają jakiś błędów.

3. Uruchomienie serwerów urządzeń zarządzających silnikami. Klasa „Motors" jest częścią oprogramowania „Pool", które należy do pakietu Sardana. Program Astor umożliwia włączenie serwera.

4. Konfiguracja silników. Jej część jest automatycznie wprowadzana w czasie uruchomienia serwera urządzeń, jednak pewne parametry trzeba ustawić samemu. Zostały ustawione następujące elementy:

   (a) urządzenie „motor/motctrl01/1":
      i. zakres wartości położenia: -120, 120,
      ii. progi alarmowe: -110, 110,
      iii. progi ostrzeżeń: -100, 100,
      iv. pozycje czujników krańcowych - takie, jak zakresy wartości położenia,
      v. prędkość: 10,
      vi. przyspieszenie: 0,5.

   (b) urządzenie „motor/motctrl01/2":
      i. zakres wartości położenia: -120, 120,
      ii. progi alarmowe: -110, 110,
      iii. progi ostrzeżeń: -100, 100,
      iv. pozycje czujników krańcowych - takie, jak zakresy wartości położenia,
      v. prędkość: 100,
      vi. przyspieszenie: 40.

   (c) urządzenie „motor/motctrl01/3":
      i. zakres wartości położenia: -50, 50,

ii. progi alarmowe: -45, 45,
iii. progi ostrzeżeń: -40, 40,
iv. pozycje czujników krańcowych - takie, jak zakresy wartości położenia,
v. prędkość: 10,
vi. przyspieszenie: 0,1.

(d) urządzenie „motor/motctrl01/4”:

i. zakres wartości położenia: -90, 140,
ii. progi alarmowe: -85, 130,
iii. progi ostrzeżeń: -80, 125,
iv. pozycje czujników krańcowych - takie, jak zakresy wartości położenia,
v. prędkość: 100,
vi. przyspieszenie: 1.

5. Dodanie odpowiednich parametrów do archiwizacji. Wirtualna maszyna „TangoBox9” zawiera system archiwizacji „HDB++”, który został przygotowany przez zespół synchrotronu Elettra we Włoszech. Dostarczone są do niego dwie aplikacje: „jhdbconfigurator” (służąca do wprowadzania zmian w konfiguracji archiwizacji) oraz „jhdbviewer” (wykorzystywana do podglądu archiwizowanych atrybutów). Za pomocą pierwszej z nich ustawiono archiwizację położeń i pozycji czujników krańcowych każdego silnika. Druga aplikacja została zaprezentowana na rysunku 1.

6. Następnie dokonano sprawdzenia działania urządzeń nadrzędnych nad zwykłymi silnikami - tzw. „pseudo-silników” (ang. PseudoMotors). System Tango wspiera hierarchiczność urządzeń, a więc synchronizację pracy poszczególnych silników najprościej jest zrealizować właśnie w takiej formie. Użyto domyślnych ustawień, obecnych w symulacji środowiska linii pomiarowej, jakie jest dostarczone z pakietem Sardana.

   Skonfigurowano dwa pseudo-silniki: „gap” i „offset”. Oba synchronizują pracę pierwszych dwóch silników. Pierwszy może być interpretowany jako ustawienie wielkości szczeliny, przez którą pada światło, a drugi służy do realizowania przesunięcia między dwoma symetrycznymi elementami, którymi te silniki poruszają.

II. Opracowanie aplikacji eksperckiej i operatorskiej

Druga część realizacji zagadnień projektowych obejmowała utworzenie obu wymaganych aplikacji realizujących dwa odrębne sposoby wizualizacji wartości położeń wałów silników. Dodatkowo, w pracy nad nimi wykorzystano dwa możliwe sposoby tworzenia aplikacji w pakiecie TaurusGUI.

Pierwszy sposób to podejście konfiguracyjne. Użytkownik nie musi pisać kodu aplikacji - jest on generowany niejawnie za niego. Tak powstała aplikacja ekspercka: uruchomiono kreator GUI poleceniem: „`taurusgui -new`” i ustawiono konkretne panele, które miały się w niej znajdować. Następnie ustawiono odpowiednią perspektywę (o nazwie „ExpertGUI”) i dodatkowe panele, które użytkownik może sam dołączyć do aplikacji w czasie jej działania.

Kolejnym krokiem była konfiguracja panelu z wykresem oraz zapisanie ustawień, które zawierały między innymi: odpowiednie sygnały, ich nazwy, kolory na wykresie i skalę. Ostatnią czynnością było utworzenie i zapisanie przykładowej sekwencji makr. Oba wspomniane wyżej pliki konfiguracyjne zostały dołączone do plików projektowych i są opisane w rozdziale V.

Drugi możliwy sposób tworzenia interfejsów użytkownika przy pomocy pakietu Taurus to podejście programistyczne. Najpierw stworzono szkielet wyglądu aplikacji przy pomocy

programu QtDesigner. Wygenerowany plik z rozszerzeniem .ui (z ang. user interface - interfejs użytkownika) można albo bezpośrednio zaimportować do aplikacji w Pythonie, albo skonwertować na kod w tym języku. Wybrany został drugi sposób postępowania, jako że importowany plik .ui nie zawierał wszystkich potrzebnych elementów. Przede wszystkim, stworzenia wymagał drugi sposób wizualizacji, czyli klasa realizująca wyświetlanie w czasie rzeczywistym wartości położenia silników. Pisanie kodu i testowanie prototypów aplikacji w znaczącym stopniu ułatwiło korzystanie z zaawansowanego środowiska programistycznego, jakim jest PyCharm.

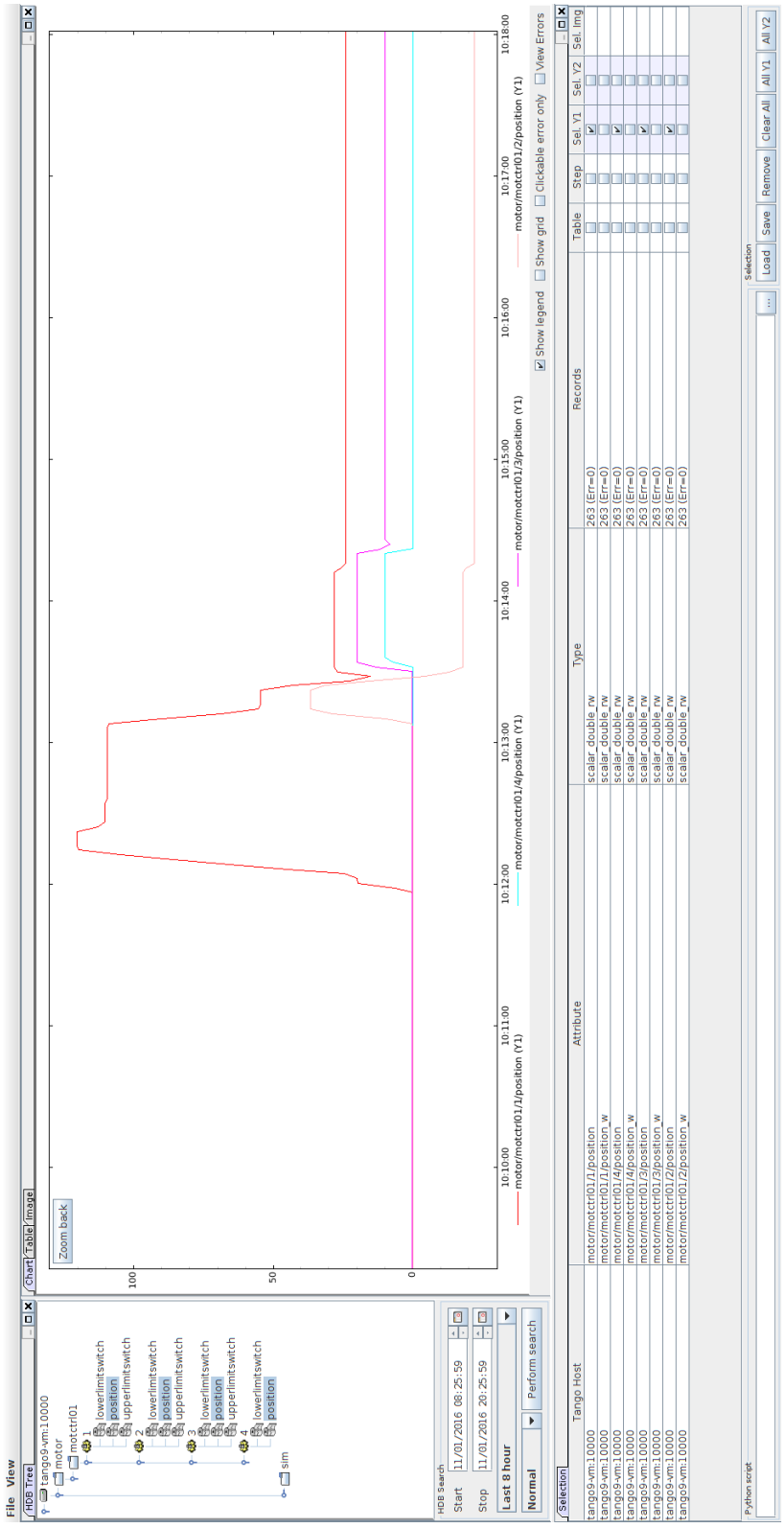
Dokładny opis obu aplikacji znajduje się w rozdziale VI.

III. Testy

Procesowi testowania został poddany każdy fragment wymienionej w poprzednich dwóch podrozdziałach konfiguracji. Korzystając ze stworzonych aplikacji jako punktów wejścia do systemu, poddano próbie ustawienia silników oraz same aplikacje. Sprawdzono poprawność konfiguracji archiwizacji (co jest widoczne na rysunku 1).

Na wykresach można zauważyć wpływ różnych ustawień prędkości i przyspieszeń poszczególnych silników na ogólne funkcjonowanie całego ich zbioru - przede wszystkim jest to odczuwalne w przypadku poruszania pierwszymi dwoma silnikami za pomocą pseudo-motorów.

Funkcjonalność synchronizacji ruchu została zapewniona przez zastosowanie tych właśnie nadrzędnych pseudo-silników. Co warte zaznaczenia, mimo różnic w prędkościach i przyspieszeniach między silnikami pierwszym i drugim (połączonymi przez owe nadrzędne urządzenia), ich położenia zmieniają się według takich samych krzywych.



Rysunek 1: *Zrzut ekranu przedstawiający aplikację wyświetlającą dane archiwalne położenia silników.*

V. Opis załączonego kodu

Załączone do niniejszego raportu archiwum w formacie ZIP, stanowiące techniczno - implementacyjną część projektu stanowi zrzut repozytorium, które zostało założone na potrzeby projektu. Zawiera następujące elementy:

1. Katalog „OperatorGUI”:
   (a) Plik Pythona „editable_mano_meter.py” zawierający definicję klasy wykresów kołowych użytych w aplikacji operatora.
   (b) Plik Pythona „operator_gui.py” zawierający definicję całej aplikacji operatora.
   (c) Plik „operator_gui.ui” będący podstawą aplikacji operatora utworzoną przy pomocy programu QtDesigner.
2. Katalog „Raport” zawierający niniejszy raport oraz jego pliki źródłowe w języku LaTeX. Ten opis nie zawiera jego dokładnej zawartości, jako że nie jest on częścią kodu napisanego na potrzeby projektu.
3. Plik Pythona „__init__.py” potrzebny Pythonowi, aby odpowiednio rozpoznał pliki w tym języku znajdujące się wewnątrz tego katalogu.
4. Pliki konfiguracyjne aplikacji eksperckiej: „config.xml” oraz „config.py”.
5. Skrypt „ExpertGUI” służący do uruchomienia aplikacji eksperckiej.
6. Plik „macro_sequence_motors_showcase.xml” stanowiący zapis sekwencji makr.
7. Plik „motors_positions.pck” stanowiący zapis konfiguracji wykresu w aplikacji eksperckiej.
8. Plik „README.md” opisujący skrótowo zawartość katalogu projektowego.
9. Plik „wizard.log” zawierający podsumowanie generacji ustawień aplikacji eksperckiej.
10. Pliki konfiguracyjne repozytorium.

VI. Instrukcja użytykownika

Jak zostało wcześniej wspomniane, możliwość interakcji z systemem silników została przewidziana w formie dwóch aplikacji. Niniejszy rozdział został poświęcony opisowi możliwych schematów wykorzystania przygotowanego oprogramowania. Podstawą obu jest opisywana w rozdziale III biblioteka Taurus, a konkretniej jej część (o nazwie „TaurusGUI") umożliwiająca programowanie bądź konfigurację głównych okien interfejsów użytkownika.

I. Aplikacja ekspercka

Pierwsza z utworzonych aplikacji to interfejs ekspercki - został on przedstawiony na rysunku 2. Umożliwia on sprawowanie pełnej kontroli nad procesem sterowania silnikami, a więc również nad przeprowadzaniem eksperymentów na linii badawczej. Została ona podzielona na 4 części oraz dwa menu:

1. Część w lewym górnym rogu zawiera dwa panele. Pierwszy o nazwie „Macros", widoczny na rysunku przedstawiającym aplikację ekspercką, to panel umożliwiający użytkownikowi wybór odpowiedniej, pojedynczej komendy spośród obfitego zestawu dostępnych, ustawienie jej parametrów i uruchomienie. Wybór odbywa się poprzez kliknięcie odpowiedniej pozycji na rozwijanej liście. Uruchomić makro można przyciskiem „Play" - obok niego znajdują się również przyciski do wstrzymywania i przerywania działania komendy. Dioda w ostatnim wierszu pokazuje stan urządzenia, które zajmuje się wykonywaniem makr - kolory odpowiadają standardom Tango, opisanym w rozdziale III.

   Drugi panel - „Macro Description" - zawiera opis wybranego makra. Przedstawia on skrótowy opis działania komendy oraz znaczenie i zakres wszystkich jej parametrów - zarówno wejściowych, jak i wyjściowych (jeśli dane makro cokolwiek zwraca). Znajdują się tam również czasem dodatkowe informacje lub fakty, na które warto zwrócić uwagę. Wyboru można dokonywać zarówno na poprzednim panelu w tej części aplikacji, jak i na panelu „Sequence", opisanym niżej.

2. Część w prawym górnym rogu zawiera aż 4 panele. Pierwszy z nich, widoczny na rysunku, to panel „Sequence". Zawiera on opcje pozwalające na wykonywanie, zapisywanie i wczytywanie sekwencji makr. Pierwsza z tych operacji jest aktywowana przyciskiem „Play", podobnie, jak na panelu „Macros" (jak również przyciski umożliwiające wstrzymanie i przerwanie działania makra). Możliwość tworzenia nowej sekwencji, otwierania istniejącej i zapisywania bieżącej dają (w takiej właśnie kolejności) przyciski, które znajdują się w górnym lewym rogu panelu.

   Po wybraniu makra z rozwijanej listy można je dodać do edytowanej sekwencji przyciskiem „+" znajdującym się w prawym górnym rogu panelu. Przyciski poniżej „+" umożliwiają odpowiednio usunięcie makra z sekwencji, przesunięcie go w górę lub w dół na liście obrazującej kolejność wykonywania oraz zmianę poziomu wykonywania makra (dotyczy tylko niektórych, powiązanych komend). Po wybraniu makra na liście sekwencji uaktywnia się dolna lista na panelu, która zawiera konfigurację wszystkich argumentów wejściowych makra.

   Pozostałe panele w tej części aplikacji eksperckiej - „DoorOutput", „DoorDebug" oraz „DoorResult" - zawierają informacje zwrotne uzyskiwane po uruchomieniu makra (bądź ich sekwencji). Znajdują się tam odpowiednio komunikaty, wiadomości dotyczące odpluskwiania makra (jeśli uruchomione jest ono w takim trybie) oraz wyniki ewentualnych obliczeń. W przygotowanej na potrzeby sekwencji makr te panele nie są wykorzystywane.

3. Część znajdująca się w lewym dolnym rogu zawiera tylko jeden panel - „Positions". Jest on interaktywnym wykresem, aktualizowanym na bieżąco w czasie rzeczywistym (okres oczekiwania na odświeżenie wynosi 1 do 3 sekund). Zmiennymi, których wartości przedstawia są położenia wszystkich czterech silników, stanowiących podstawę niniejszego projektu. Ten panel daje użytkownikowi bardzo dużo możliwości - większość z nich jest dostępna poprzez kliknięcie prawym przyciskiem myszy na wykres. Pokazuje się wtedy menu kontekstowe, które zawiera takie opcje, jak:

   - dostosowywanie ustawień osi (wybór zakresów, zmiana trybu z liniowego na logarytmiczny, określenie kolorów odpowiadających poszczególnym sygnałom, itp.),
   - ustawienie zmiennych prezentowanych na wykresie,
   - zapisanie i wczytanie aktualnych ustawień wykresu,
   - zmianę tytułu,
   - dostosowanie skal osi (oraz opcję autoskalowania),
   - włączanie/wyłączanie pokazywania wartości minimalnej/maksymalnej oraz legendy,
   - obliczanie większej ilości parametrów statystycznych.

   Dodatkowo, klikanie na nazwy atrybutów w legendzie pozwala wyłączać pokazywanie odpowiadającej im krzywej na wykresie lub przełączanie się między prawą i lewą osią Y.

4. W prawym dolnym rogu aplikacji eksperckiej znajdują się dwa panele - „/slit" oraz „/mirror". Oba umożliwiają bezpośrednie poruszanie silnikami bądź grupami silników. Każdy z nich zawiera pola do wpisywania wartości zadanych oraz monitorowania aktualnego położenia silników. Przycisk „Stop", znajdujący się między polami każdego silnika pozwala zatrzymać ruch. Po lewej stronie, zaraz za nazwą silnika, znajdują się przyciski „-" oraz „+". Pozwalają one na wykonanie pojedynczego kroku odpowiednim silnikiem.

   Kluczową sprawą dla użytkowania aplikacji jest zwracanie uwagi na format wpisywanych w polach edycyjnych danych. Normalna czcionka oznacza brak różnic między wartością zadaną a aktualnym położeniem. W takim stylu są też wyświetlane wartości po uruchomieniu aplikacji. Kolor niebieski i pogrubienie czcionki znamionują obecność takich zmian - można je zatwierdzić, używając klawisza Enter. Użytkownik może również chcieć zaaplikować wartość wyświetlaną w normalnym stylu (na czarno) - w tym celu przewidziano kombinację klawiszy Ctrl + Enter. Żółty kolor czcionki oznacza wartość, która przekroczyła poziom ostrzeżenia. Kolor żółty oraz czerwona ramka naokoło pola edytowalnego - wartość przekraczającą poziom alarmowy.

5. Dodatkowymi elementami aplikacji są 2 menu oraz dioda znajdująca się w prawym dolnym rogu. Jej ciągłe miganie to efekt „bicia serca" (ang. hearbeat), czyli proste sprawdzenie, czy aplikacja działa, czy się zacięła.

   Pierwsze menu znajduje się na samej górze aplikacji. Składa się z dwóch części - jedna to standardowe menu aplikacji, w którym znajdziemy takie opcje jak zapisanie perspektywy, zmianę widoku, opcje wyświetlania i ukrywania paneli, itp. Poniżej znajdują się przyciski menu narzędziowego - umożliwiają one szybki dostęp do maksymalizacji okna, wczytania perspektywy (czyli aktualnie wyświetlanej konfiguracji paneli) oraz dodanie nowego panelu.

   Drugie menu znajduje się w pasku po prawej stronie aplikacji. Obecne tam 4 ikony to odpowiednio - logo organizacji, logo aplikacji, przycisk paniki (zatrzymuje wszystkie aktualnie wykonywane operacje) oraz tryb inspekcji.

Przykładowe użycie aplikacji mogłoby obejmować następujące kroki:

1. Uruchom aplikację poprzez wywołanie skryptu: `./ExpertGUI`.
2. Wczytaj zapisaną perspektywę, sekwencję makr i ustawienia wykresów.
3. Sprawdź, czy wszystko jest gotowe do przeprowadzenia eksperymentu (w szczególności należy zadbać o właściwą kolejność makr i odpowiednie dla nich argumenty).
4. Włącz sekwencję makr.
5. Monitoruj położenie silników na wykresie.
6. W razie potrzeby, dopasuj ręcznie pozycję (lub zostaw to operatorowi).

II. Aplikacja kliencka

Druga z utworzonych aplikacji to interfejs klienta lub operatora - jest on przedstawiony na rysunku 3. Pozwala on tylko na poruszanie konkretnymi silnikami w - operator może skorygować w niewielkim zakresie wartość położenia każdego elementu ruchomego i zobaczyć aktualny stan poszczególnych silników.

Ta aplikacja zawiera tylko jeden typ panelu - taki sam dla wszystkich czterech silników, którymi operator może sterować. Każdy z tych paneli zawiera następujące elementy:

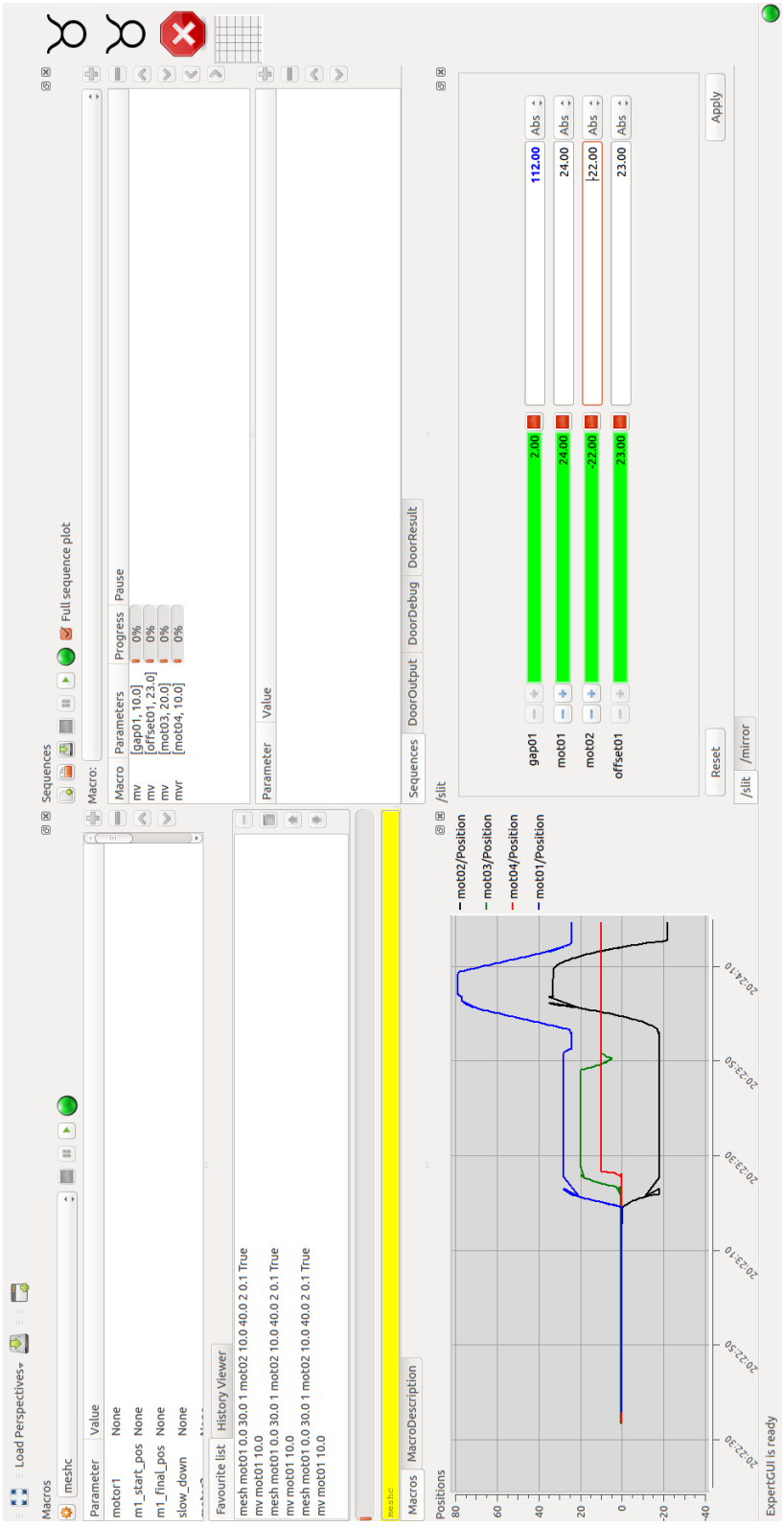
1. Nazwę silnika (według konwencji Tango).
2. Diodę określającą stan urządzenia (odpowiadającą standardowej kolorystyce wykorzystywanej w Tango). Na rysunku 3 są widoczne trzy możliwe stany urządzenia związanego z silnikiem.
3. Duże, zielone pole statusu. Jego rozmiar jest warunkowany najdłuższym możliwym komunikatem, który jest w nim wyświetlany. Jak widać na rysunku 3, może prezentować różne rodzaje informacji statusowej.
4. Wykres kołowy aktualizowany na bieżąco, który pokazuje pozycję silnika.
5. Pole edytowalne służące do wpisywania pożądanej wartości zadanej położenia silnika. Podlega dokładnie takim samym regułom, jak to w aplikacji eksperckiej, opisane w podrozdziale I.

Najechanie kursorem na dowolny element powyższej listy (oprócz nazwy silnika) powoduje pokazanie się widocznej na rysunku 3 podpowiedzi dotyczącej atrybutu dołączonego do danego elementu aplikacji. W przypadku pozycji pokazują się tam takie informacje, jak zakres, poziomy ostrzeżenia i alarmu.

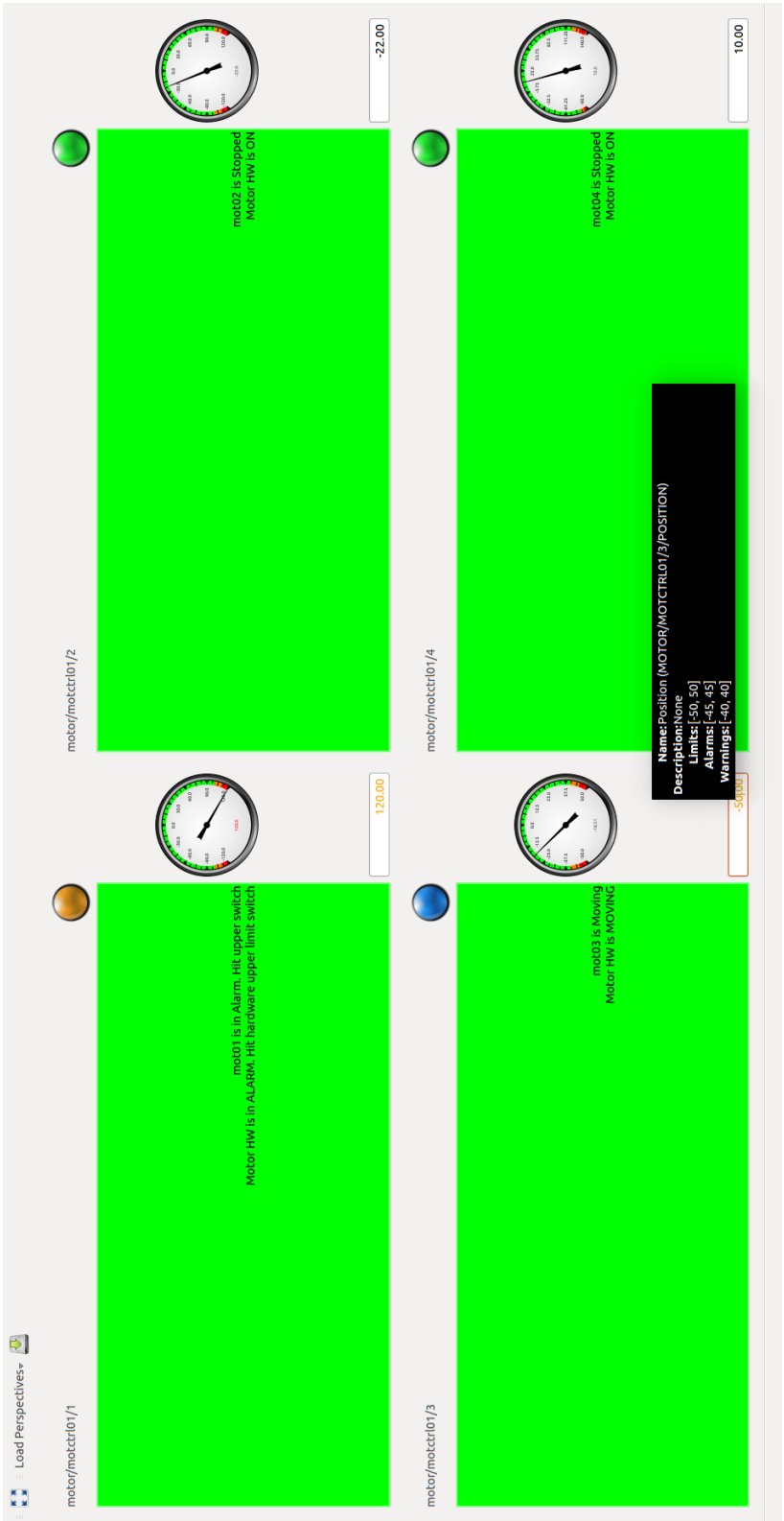
Aplikacja operatora, tak jak opisywane wcześniej oprogramowanie eksperta, posiada menu bazowe aplikacji opartej o TaurusGUI, a więc umożliwiające m.in. konfigurację widoku oraz tworzenie perspektyw.

Przykładowy proces korzystania z aplikacji może być przedstawiony w następujących krokach:

1. Uruchom aplikację poleceniem: `python OperatoGUI/operator_gui.py`
2. Sprawdź, czy stany wszystkich silników umożliwiają poruszenie nimi.
3. Dokonaj odpowiednich korekt w położeniu.



Rysunek 2: *Zrzut ekranu pokazujący aplikację eksperecką.*



Rysunek 3: *Zrzut ekranu pokazujący aplikację operatora.*

VII. Bibliografia

[1] Strona internetowa Tango Controls. http://www.tango-controls.org/ . Zarządzane przez: Stowarzyszenie Tango. Stan na: 10.01.2016 r.

[2] Dokumentacja zestawu narzędzi Taurus. http://www.taurus-scada.org/en/stable/ . Zarządzane przez: ALBA Synchrotron. Stan na: 10.01.2016 r.

[3] Dokumentacja zestawu narzędzi Sardana. http://sardana.readthedocs.org/en/stable/ . Zarządzane przez: ALBA Synchrotron. Stan na: 10.01.2016 r.

[4] Strona internetowa poświęcona zestawowi silników IcePAP. http://www.esrf.eu/Instrumentation/DetectorsAndElectronics/icepap . Zarządzane przez: ESRF. Stan na: 10.01.2016 r.

[5] Janvier, N., Clement, J. M., Fajardo, P., *IcePAP: An Advanced Motor Controller for Scientific Applications in Large User Facilities*. W materiałach: ICALEPCS 2013, San Francisco, Kalifornia, USA.

[6] System TANGO https://en.wikipedia.org/wiki/TANGO . Stan na: 13.01.2016 r.

[7] Sardana Home Page http://www.sardana-controls.org/en/stable/ . Stan na: 13.01.2016 r.

[8] Python https://pl.wikipedia.org/wiki/Python . Stan na: 13.01.2016 r.

[9] PyCharm https://pl.wikipedia.org/wiki/PyCharm . Stan na: 13.01.2016 r.