



**AKADEMIA GÓRNICZO-HUTNICZA  
IM. STANISŁAWA STASZICA W KRAKOWIE**

## **Rozbudowa i uaktualnienie systemu GGSS detektora ATLAS TRT**

Arkadiusz Kasprzak  
Jarosław Cierpich

Opiekun pracy: dr hab. inż. Bartosz Mindur, prof. AGH

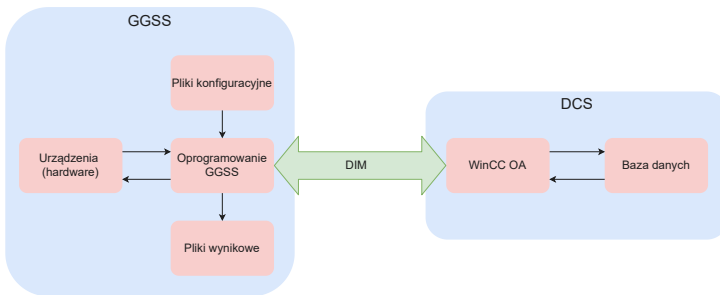


- 1 Wprowadzenie do tematyki pracy**
- 2 Modyfikacja kodu źródłowego projektu**
- 3 Modyfikacja systemu budowania projektu**
- 4 Podsumowanie**



## AGH Wprowadzenie do systemu GGSS

- System Stabilizacji Wzmocnienia Gazowego (GGSS - *Gas Gain Stabilization System*)
- projekt zintegrowany z systemem kontroli detektora ATLAS w CERN
- umożliwia poprawne działanie detektora TRT (*Transition Radiation Tracker*) będącego częścią ATLAS
- składa się z warstwy oprogramowania (aplikacje oraz infrastruktura) oraz sprzętu (zasilacz wysokiego napięcia, multiplekser analogowy, wielokanałowy analizator amplitudy MCA)
- warstwa oprogramowania koordynuje działanie urządzeń i umożliwia sterowanie nimi za pomocą specjalnych komend
- aplikacje i biblioteki napisane z wykorzystaniem języka C++, infrastruktura: CMake, Bash oraz Python



**Rysunek:** Wysokopoziomowa architektura systemu GGSS

- WinCC OA - system system typu SCADA pozwalający na sterowanie i kontrolę działania podsystemów detektora
- DIM - protokół komunikacyjny dla aplikacji rozproszonych



## AGH Cele pracy

- kontynuacja pracy inżynierskiej, skupiającej się na infrastrukturze projektu
- rozbudowa przygotowanych w ramach wspomnianej pracy rozwiązań
- główny nacisk na aplikację *ggss-runner*, stanowiącą trzon systemu
- poprawa jakości kodu źródłowego oraz wprowadzenie nowych funkcjonalności
- rozbudowa infrastruktury pozwalającej na testowanie projektu
- udokumentowanie projektu (dokumentacja kodu źródłowego oraz pliki instruktażowe)
- migracja projektu - nowy komputer docelowy (początkowo planowany wyjazd do CERN, ostatecznie zrealizowana zdalnie)
- konieczność zachowania wysokiej niezawodności systemu - stosowanie praktyk takich jak *code review* i testy automatyczne

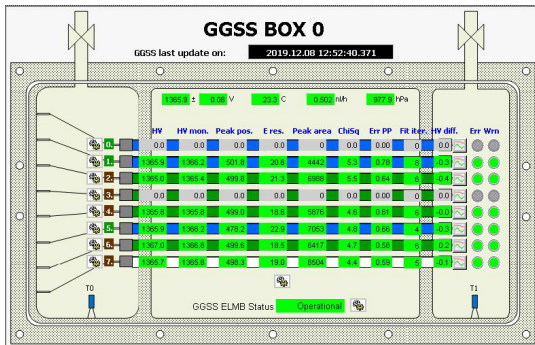


- projekt ściśle związany z infrastrukturą CERN
- aplikacje wchodzące w skład systemu działają na dedykowanym komputerze z dostępem do wymaganych urządzeń
- ograniczenia dotyczące wersji kompilatora języka C++ oraz narzędzia CMake
- ograniczone uprawnienia w środowisku docelowym
- konieczność zachowania kompatybilności wstecznej

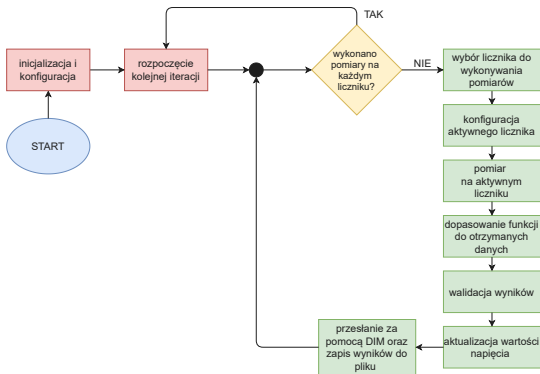


## AGH Aplikacja ggss-runner

- wielowątkowa aplikacja napisana z wykorzystaniem C++ i pakietu Boost
- zadanie: cykliczne gromadzenie danych w postaci widma poprzez komunikację z wielokanałowym analizatorem amplitudy oraz wyznaczanie na ich podstawie wartości napięcia, komunikacja z systemem WinCC OA



Rysunek: Panel dostępny w ramach systemu WinCC OA



Rysunek: Uproszczony schemat działania aplikacji





## AGH Modernizacja i poprawa jakości kodu źródłowego

- pierwotnie kod wykorzystywał standard C++03 oraz elementy C++11
- przeprowadzono migrację do standardu C++11 (pętle zakresowe, silne typy wyliczeniowe itd.)
- migracja niepełna z uwagi na ograniczenia kompilatora
- ujednolicono konwencje stosowane w kodzie (formatowanie, nazewnictwo)
- zlikwidowano niewykorzystywane lub wykomentowane fragmenty kodu
- usunięto nieliczne błędy (biblioteki *xml-lib* oraz *log-lib*)
- znaczne zmiany w 12 z 14 bibliotek wchodzących w skład systemu

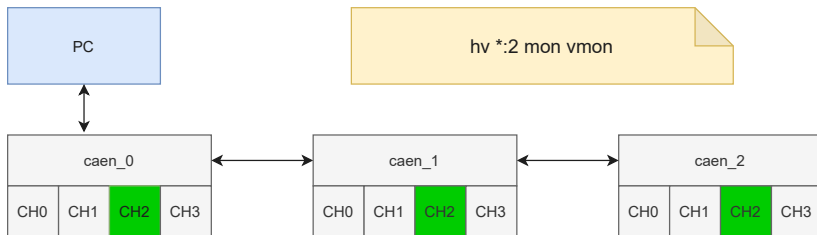


- dodanie obsługi zaawansowanych komend dla zasilaczy wysokiego napięcia
- rozbudowa biblioteki odpowiedzialnej za dopasowanie krzywej do zebranych danych
- dodanie możliwości aktualizacji parametrów i zebranych danych na żądanie
- dodanie zabezpieczenia przed przepełnieniem bufora urządzenia MCA
- dodanie możliwości przywracania domyślnej kolejności liczników słomkowych



## AGH    Przykład: nowe komendy dla zasilaczy

- GGSS wykorzystuje 3 zasilacze wysokiego napięcia CAEN N1470
- do tej pory użytkownik mógł przeprowadzić z nimi interakcję za pomocą systemu komend wspieranego przez producenta, np.:  
`$BD:00,CMD:SET,CH:0,PAR:VSET,VAL:1368.0`
- format niewygodny i nieintuicyjny z punktu widzenia użytkownika
- ponadto nie pozwala na odczyt danych lub wprowadzenie zmian dla kilku zasilaczy jednocześnie
- przygotowano więc nowy, przyjazny użytkownikowi format komend
- implementacja jako biblioteka statyczna - możliwość wielokrotnego wykorzystania
- przykład: `hv *:0 set vset 1368.0`

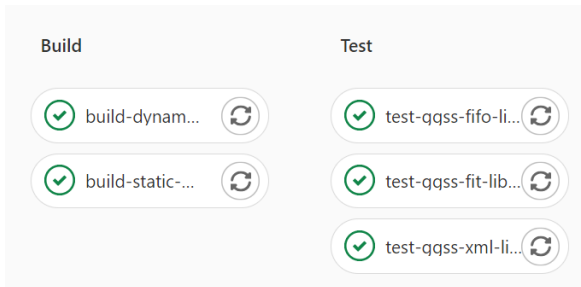


**Rysunek:** Schemat przybliżający sposób działania nowego formatu komend.



## AGH Testy automatyczne

- w celu zapewnienia niezawodności i szybkiego wykrywania błędów wykorzystano testy automatyczne i metodykę TDD (*Test Driven Development*)
- testy przygotowane z wykorzystaniem biblioteki Boost.Test
- uruchamiane automatycznie po wdrożeniu zmian do repozytoriów projektu



**Rysunek:** Pipeline CI/CD wykonujący testy automatyczne po wdrożeniu zmian



## AGH Modyfikacja systemu budowania projektu

- system oparty o narzędzie CMake, przygotowany w ramach pracy inżynierskiej
- jego zadaniem jest obsługa projektu o hierarchicznej strukturze
- wsparcie dla testów automatycznych oraz generowania dokumentacji za pomocą narzędzia Doxygen
- system oparty o tzw. szablony - pliki *.cmake* zawierające często wykorzystywane fragmenty kodu
- zmiana sposobu implementacji szablonów - wykorzystanie funkcji i makr

```
ggss_build_static_library(  
    TARGET_NAME "thread"  
    DEPENDENCY_PREFIX "${CMAKE_CURRENT_SOURCE_DIR}/.." "  
    DEPENDENCIES "sigslot" "ggss-util-lib/log"  
)
```



- wszystkie podstawione wymagania zostały spełnione
- stosowanie nowoczesnych praktyk takich jak testy automatyczne pozytywnie wpłynęło na proces rozwoju oprogramowania
- liczne testy w środowisku docelowym potwierdzają poprawność wprowadzonych zmian
- możliwy dalszy rozwój projektu - modernizacja kodu, zmiany na poziomie architektonicznym

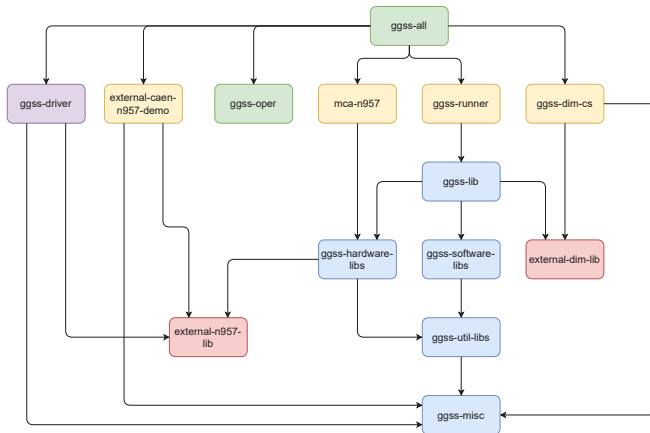


- 5 Modyfikacja architektury projektu**
- 6 Modyfikacja infrastruktury projektu**
- 7 Aplikacje do testów urządzeń**
- 8 Testy projektu**
- 9 Podsumowanie**



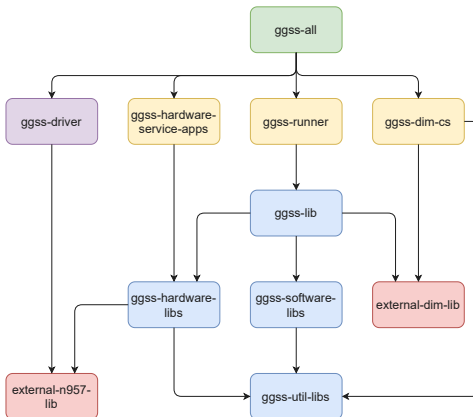
# Modyfikacja architektury projektu

## Poprzednia architektura projektu



**Rysunek:** Architektura projektu na podstawie modułów oraz ich powiązań

- Zmniejszenie złożoności poprzez zmniejszenie ilości modułów
- Pozbycie się modułów, które, w trakcie prac, okazały się niepotrzebne (na przykład: *ggss-misc*)
- Wydzielenie powielonego kodu do nowej biblioteki zawartej w ramach repozytorium *ggss-hardware-libs*
- Pozostawienie możliwości szybkiego powrotu do początkowej architektury oraz wersji kodu źródłowego poprzez zastosowanie gałęzi *legacy* oraz archiwizacji projektów



**Rysunek:** Architektura projektu na podstawie modułów oraz ich powiązań



## AGH Mechanizm ciągłej integracji i dostarczania

- Mechanizm pozwalający na automatyczne testowanie, budowanie oraz przygotowywanie kodu do dostarczania do środowiska docelowego
- Pierwsza wersja została utworzona w ramach pracy inżynierskiej, następnie została ona rozwinięta w ramach pracy magisterskiej
- Do mechanizmu zostało dodane tworzenie artefaktów, a wraz z wprowadzeniem testów automatycznych został dodany krok *test*
- Automatyzacji poddane zostało również wersjonowanie projektu



## AGH Automatyizacja i centralizacja wersjonowania projektu

- Automatyizacja oparta o rozszerzenia:
  - plików `.cmake` w celu wsparcia parametru `version`
  - skryptu `build.py`, który jest nakładką narzędzia CMake
  - mechanizmu ciągłej integracji w celu automatycznego generowania wersji na podstawie wiadomości zawartych w ramach rewizji
- Rozwiązanie oparte o standard *semantic-versioning*, bibliotekę *semantic-release* oraz konwencję oznaczania wiadomości zgodną z *eslint-semantic-release*

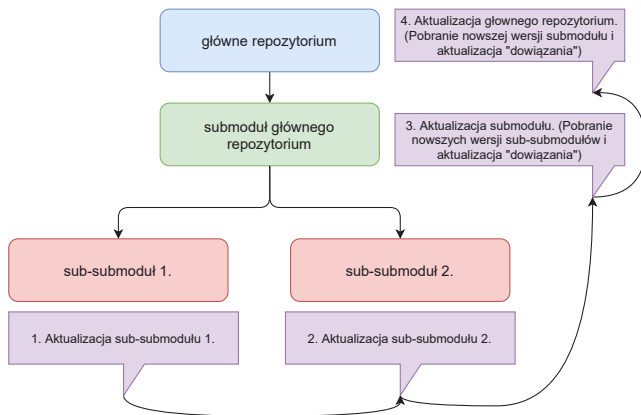


Fix: fix gitlabci paths

Jaroslav Piotr Cierpich authored 1 day ago



**Rysunek:** Rewizja zgodna z konwencją *eslint-semantic-release*



**Rysunek:** Opis kroków jakie należy podjąć, aby wprowadzone zmiany były widoczne w całym projekcie.



- w celu automatyzacji wcześniej ukazanego procesu przygotowany został skrypt *gitio* w języku *Python* oparty o bibliotekę *GitPython*
- skrypt działa na zainicjalizowanym repozytorium wraz z jego submodułami:
  - odtwarza w pamięci całą strukturę repozytoriów korzystając z metadanych *git*
  - sprawdza, czy repozytoria nie zawierają zmian, które nie zostały opublikowane w ramach rewizji
  - aktualizuje wersje wskazywane przez submoduły, na najnowsze dostępne w ramach odpowiadających im repozytoriów
  - publikuje zmiany wskazywanych wersji poprzez utworzenie nowej rewizji oraz przekazanie jej na zdalne repozytorium



- zaktualizowane zostały skrypty operacyjne, czyli skrypty obsługujące projekt GGSS w środowisku docelowym
- nowy skrypt monitorujący zasoby wykorzystywane przez główną aplikację projektu GGSS został utworzony
- przygotowana została aplikacja *high-voltage-killer*, która zabezpiecza system poprzez wyłączenie zasilania na zasilaczu wysokiego napięcia
- dokonano modyfikacji struktury katalogów oraz nazw w środowisku docelowym





- w ramach pracy magisterskiej przygotowane zostały aplikacje służące do testowania urządzeń ze względu na wysokie wymagania dotyczące niezawodności systemu oraz potrzebę sprawnych, poprawnie skonfigurowanych urządzeń w celu jego działania
- w projekcie istniały rozwiązania pozwalające na testowanie urządzeń, natomiast ich możliwości były mocno ograniczone, a ich interfejs znacznie się między nimi różnił
- przygotowane, w ramach pracy magisterskiej, rozwiązanie pozwala na dogłębne przetestowanie urządzeń z wykorzystaniem dwóch trybów: interaktywnego oraz scenariuszowego
- w celu dokonania odpowiednich testów wymagana jest jedynie wiedza domenowa. Nie jest potrzebna wiedza na temat dokładnego działania aplikacji



## Listing 1: Przykładowy scenariusz dla zasilacza wysokiego napięcia.

```
HighVoltageTestScenario:  
- hv module1:3 set vset 10.0  
- sleep 20000  
- assert hv module1:3 mon vset 10.0  
- assertTol hv module1:3 mon vmon 10.0 1.0  
- hv module1:3 set vset 0  
- sleep 20000  
- assert hv module1:3 mon vset 0  
- assert hv module1:3 mon vmon 0
```



- podczas dokonywania zmian w systemie GGSS w ramach pracy magisterskiej cyklicznie dokonywano testy nowych funkcjonalności w środowisku docelowym
- podczas pracy magisterskiej odbyła się migracja projektu GGSS do nowego środowiska. Udział autorów w migracji ograniczył się do weryfikacji poprawności konfiguracji nowego środowiska, oraz do przeprowadzenia testów systemowych z wykorzystaniem ostatniej, stabilnej wersji aplikacji
- udział autorów w projekcie GGSS został zwieńczony całościowymi testami zarówno wprowadzonych, jak i wcześniej używanych funkcjonalności. Monitorowaniu poddane zostały również zasoby wykorzystywane przez główną aplikację z wykorzystaniem wcześniej wymienionego skryptu, czy też narzędzia Valgrind



- przygotowany system automatyzacyjny pozwala na łatwe oraz szybkie wprowadzanie nowych zmian w projekcie
- wszystkie rozwiązania cechują się stosowaniem ogólnie przyjętych standardów co w połączeniu z dużym naciskiem na odpowiednią dokumentację znacząco zmniejsza próg wejścia
- w celu zapewnienia poprawnego działania wszystkie zmiany zostały przetestowane w środowisku docelowym