Join Institute for Nuclear Research
Laboratory of High Energy Physics

School of Engineering
Dubna University

Benefits of a Microkernel Architecture for Slow Control Systems Software in Research Activities

Case Studies from LHEP, JINR

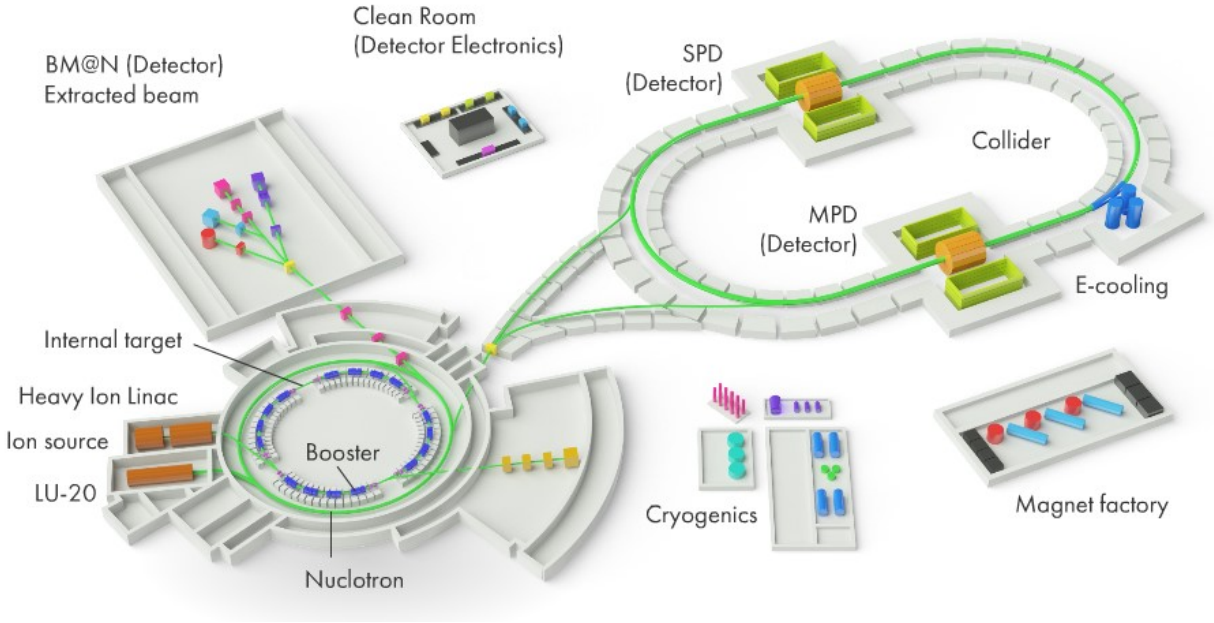
Authors: A. Kozlovsky, A. Bukharin, D. Ivliev, O. Melnikova, A. Noskov, Y. Smirnova, I. Shirikov, I. Zhabin

Oral: Alexey Kozlovsky

AYSS-2025

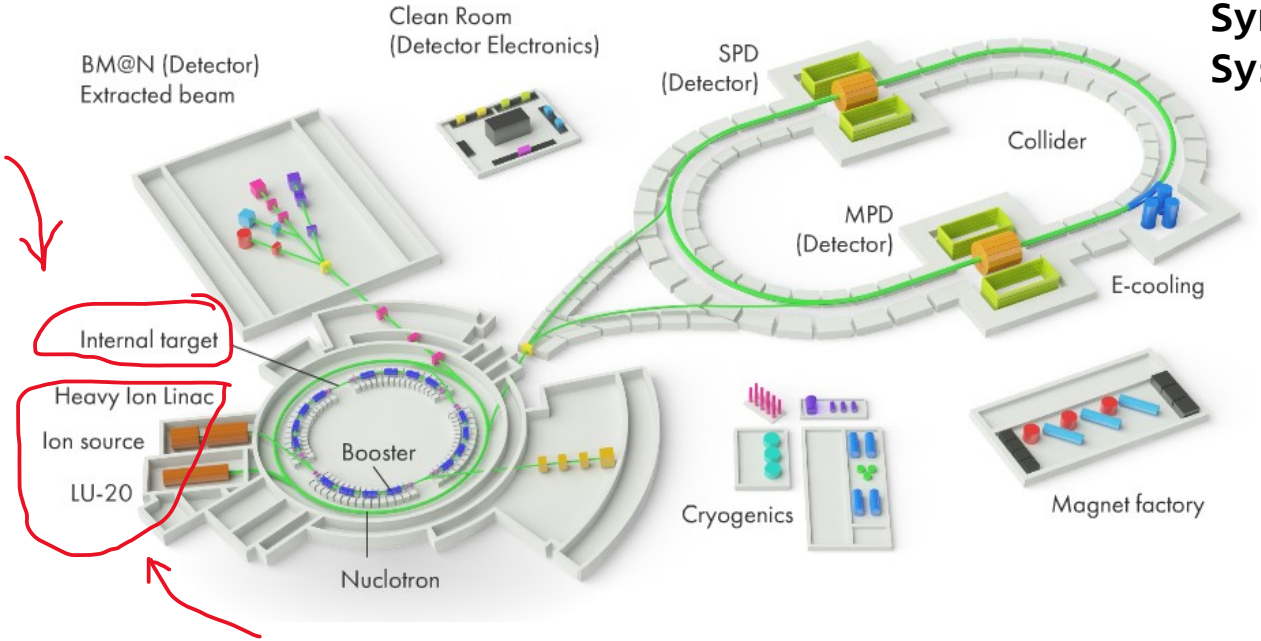
Context

Our team is developing and maintaining hardware and software for various systems at LHEP JINR



Context

Our team is developing and maintaining hardware and software for various systems at LHEP JINR

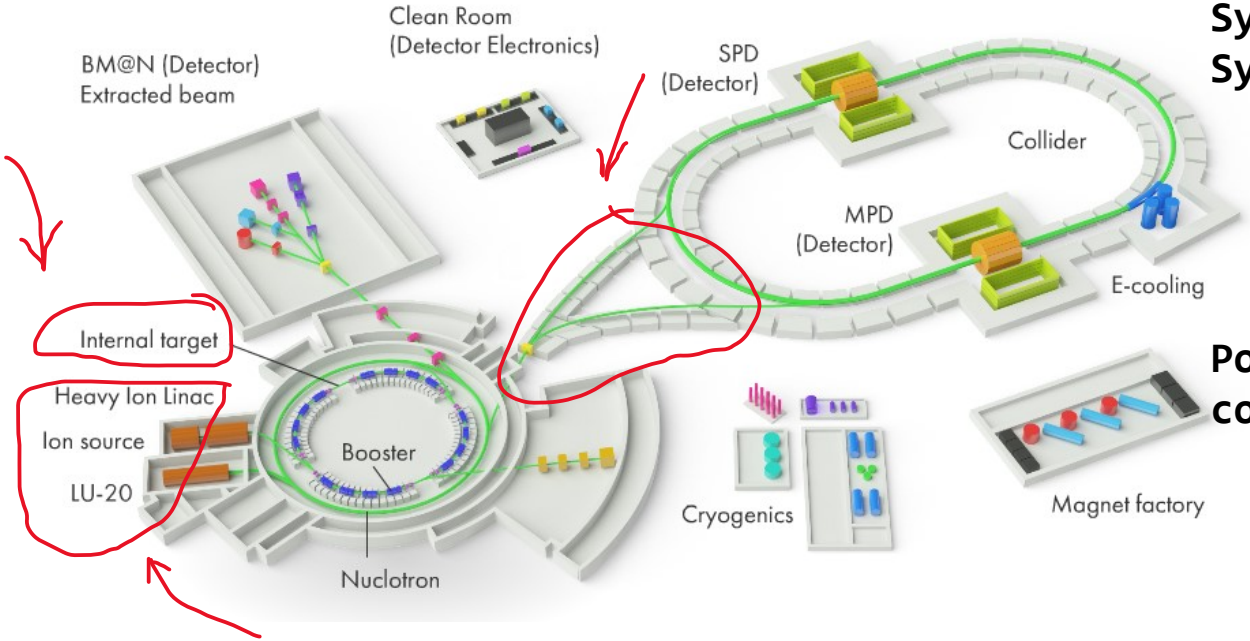


Synchronization Systems

- KRION
- Heavy Ion Linac
- Source of Polarized Ions
- Internal Nuclotron Target

Context

Our team is developing and maintaining hardware and software for various systems at LHEP JINR



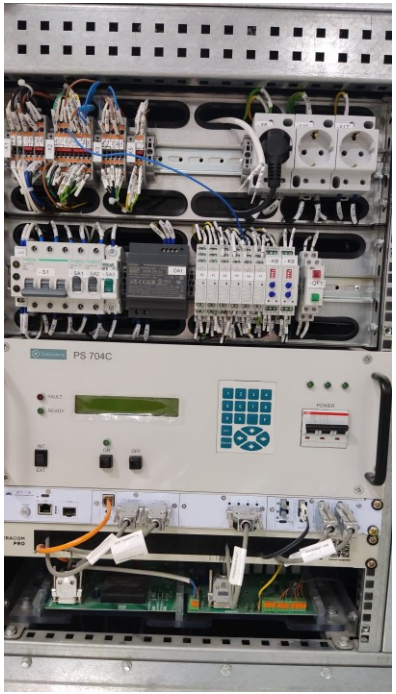
Synchronization Systems

- KRION
- Heavy Ion Linac
- Source of Polarized Ions
- Internal Nuclotron Target

Power supply controllers

- Beam Transport Channel
- Nuclotron-Collider

Context



The **hardware** is **various** according to the aim, **but** a majority of modules are built on the **unified** platform

Software should be unified as well

Problem

Research activities **require specific hardware** maintained by staff hardware developers

The **complexity** of modern systems arouses the question of the need for the **bespoke software**

Both software and **hardware** need to be **flexible** to adapt to various constantly-changing requirements from end-users: **physicists**

Slow control systems

Slow control systems

Despite the fact that there is no general definition for a **slow control system** it can be described as the **opposite** of a **Real-Time system**

Slow control systems

Despite the fact that there is no general definition for a **slow control system** it can be described as the **opposite** of a **Real-Time system**

We have a lot of tasks related to slow control systems at LHEP

Slow control systems

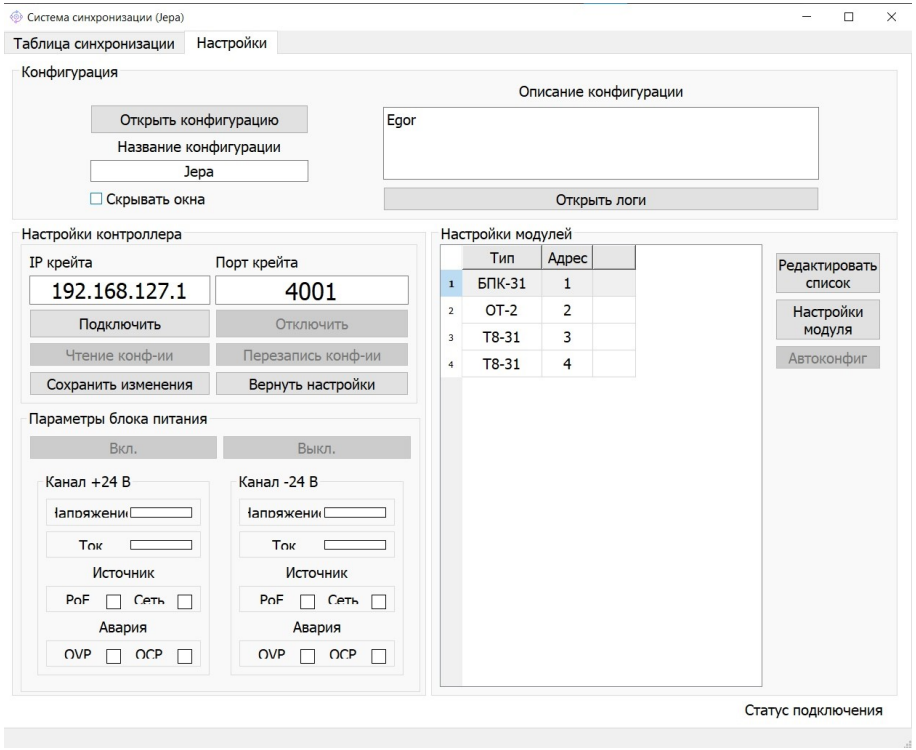
Despite the fact that there is no general definition for a **slow control system** it can be described as the **opposite** of a **Real-Time system**

We have a lot of tasks related to slow control systems at LHEP

Often we used to build the standalone software for such kind of systems from scratch

But can we unify the software developing process in such context?

Our past experience



We have started with the maintaining of **KRION** synchronization system software in 2021

But it was not scalable enough

The idea of unifying the software and implementing it into other systems came to mind

Solution

Solution

We decide to build our own platform

Solution

We decide to build our own platform

1. Integration

Solution

We decide to build our own platform

1. Integration
2. Evolutionary

Solution

We decide to build our own platform

1. Integration
2. Evolutionary
3. Simplicity

Solution

We decide to build our own platform

According to Neal Ford and Mark Richards, Microkernel Architecture would satisfy our quality attribute requirements

1. **Integration**
2. **Evolutionary**
3. **Simplicity**

Architecture characteristic	Star rating
Partitioning type	Domain and technical
Number of quanta	1
Deployability	★★★
Elasticity	★
Evolutionary	★★★
Fault tolerance	★
Modularity	★★★
Overall cost	★★★★★
Performance	★★★
Reliability	★★★
Scalability	★
Simplicity	★★★★
Testability	★★★

Solution

We decide to build our own platform

According to Neal Ford and Mark Richards, Microkernel Architecture would satisfy our quality attribute requirements

1. **Integration**
2. **Evolutionary**
3. **Simplicity**

Microkernel Architecture == Plugin Architecture

Architecture characteristic	Star rating
Partitioning type	Domain and technical
Number of quanta	1
Deployability	★★★
Elasticity	★
Evolutionary	★★★
Fault tolerance	★
Modularity	★★★
Overall cost	★★★★★
Performance	★★★
Reliability	★★★
Scalability	★
Simplicity	★★★★
Testability	★★★

Solution

We decide to build our own platform

According to Neal Ford and Mark Richards, Microkernel Architecture would satisfy our quality attribute requirements

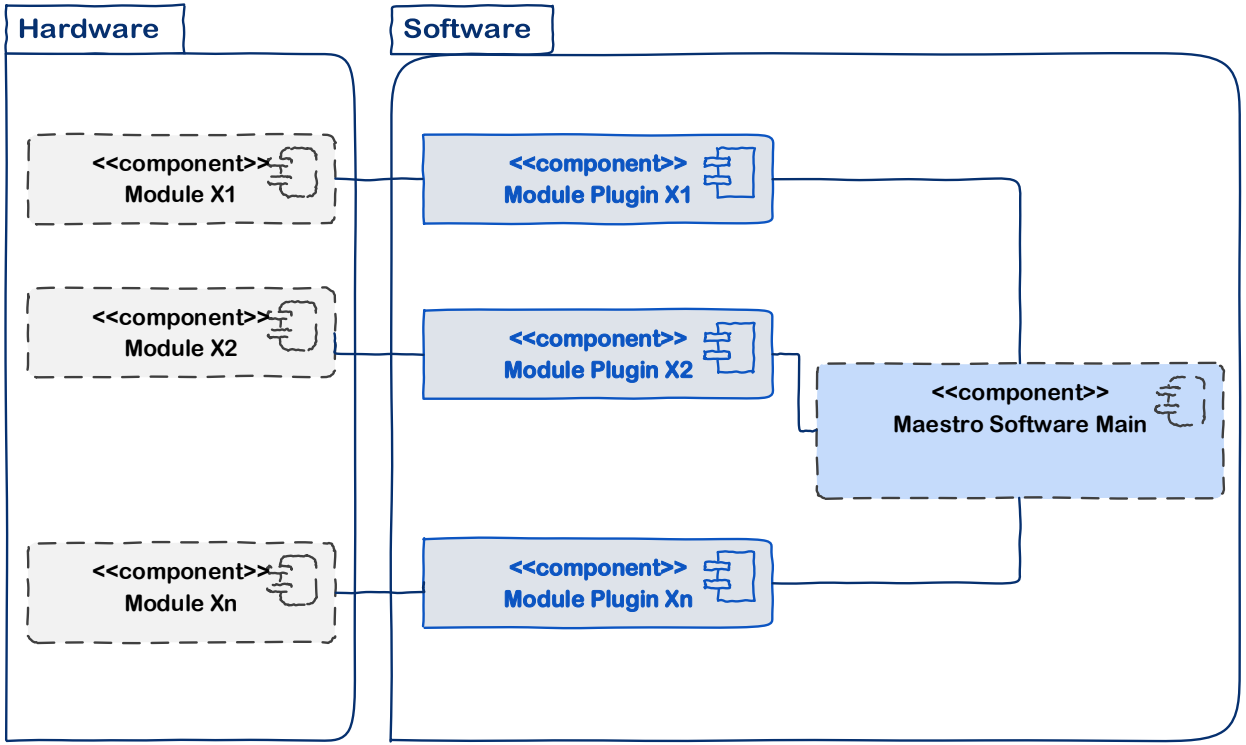
1. **Integration**
2. **Evolutionary**
3. **Simplicity**

Microkernel Architecture == Plugin Architecture

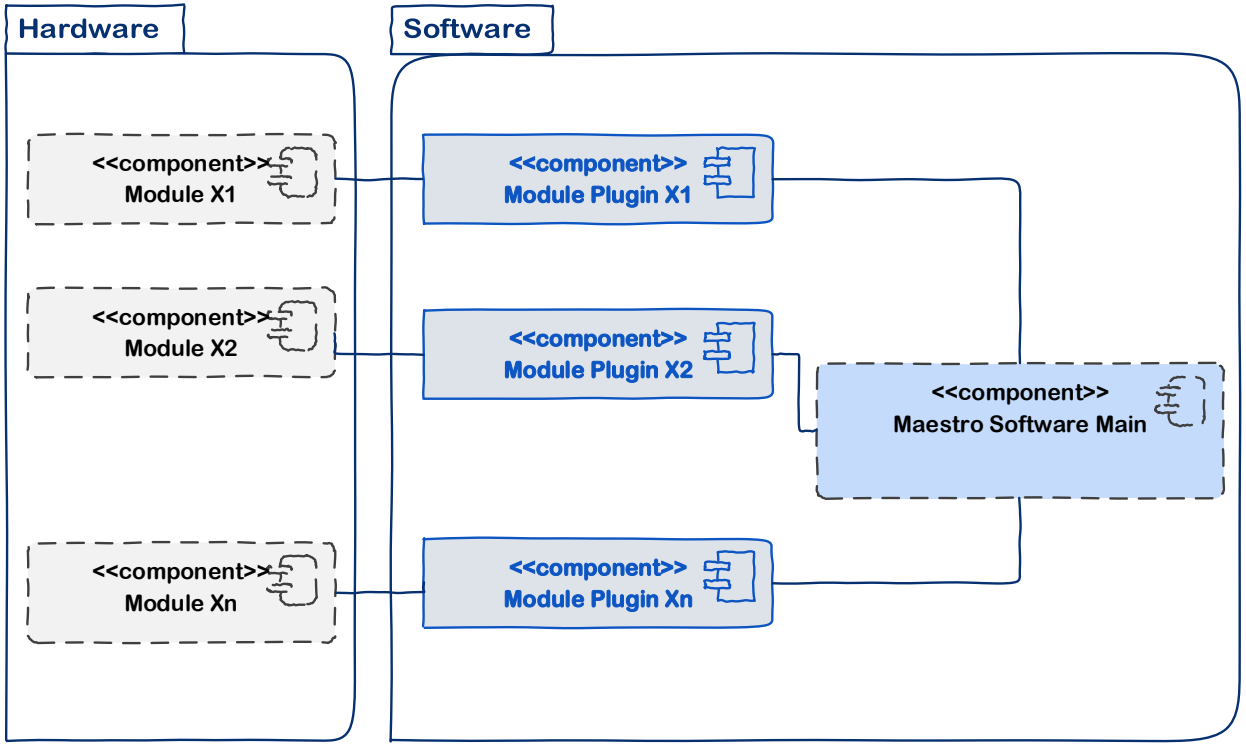
Software module for controlling hardware device == Plugin

Architecture characteristic	Star rating
Partitioning type	Domain and technical
Number of quanta	1
Deployability	★★★
Elasticity	★
Evolutionary	★★★
Fault tolerance	★
Modularity	★★★
Overall cost	★★★★★
Performance	★★★
Reliability	★★★
Scalability	★
Simplicity	★★★★
Testability	★★★

Implementation – Maestro Software

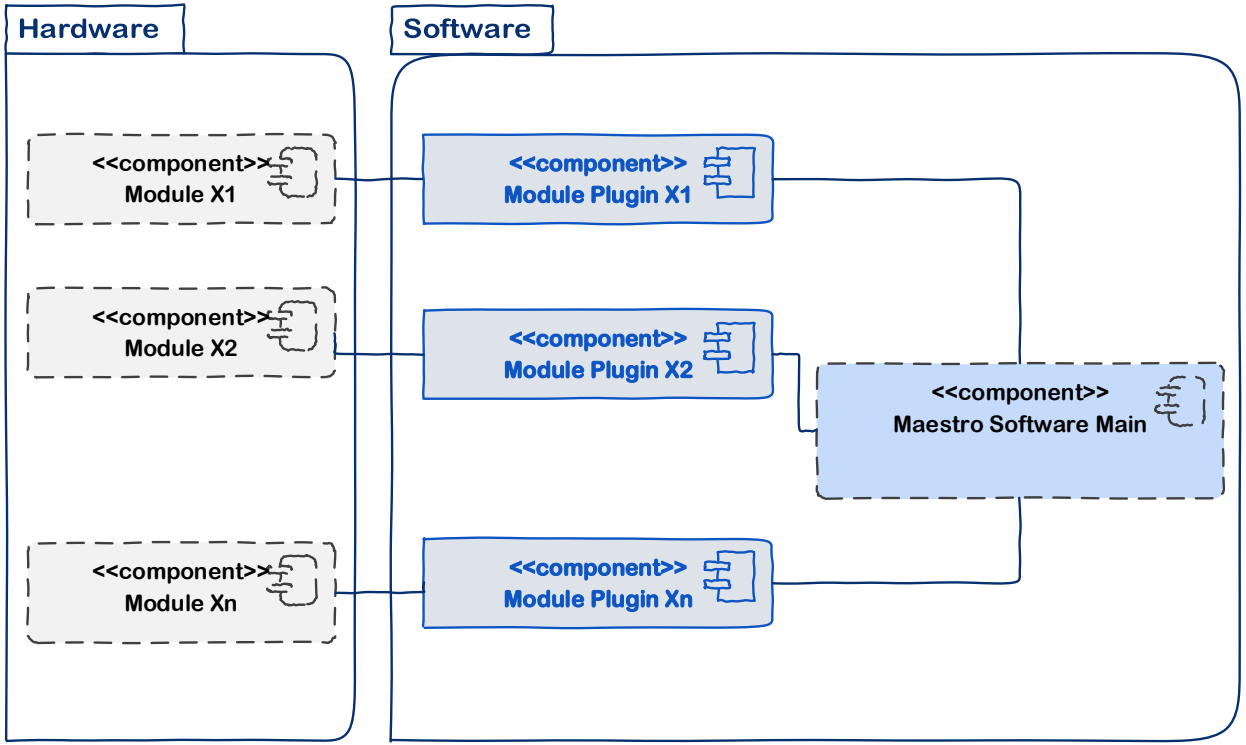


Implementation – Maestro Software



Every plugin is a dynamic library (.so or .dll)

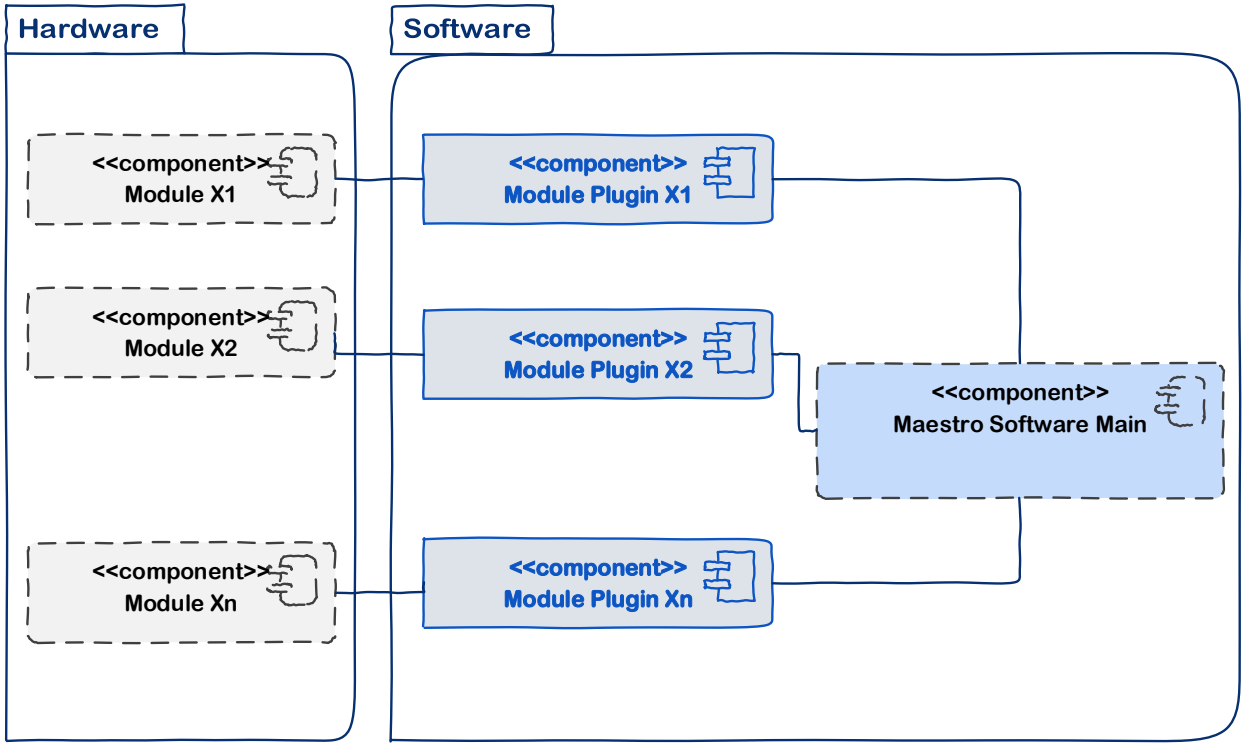
Implementation – Maestro Software



Every plugin is a dynamic library (.so or .dll)

For **each** hardware module we have its software **adaptor**

Implementation – Maestro Software

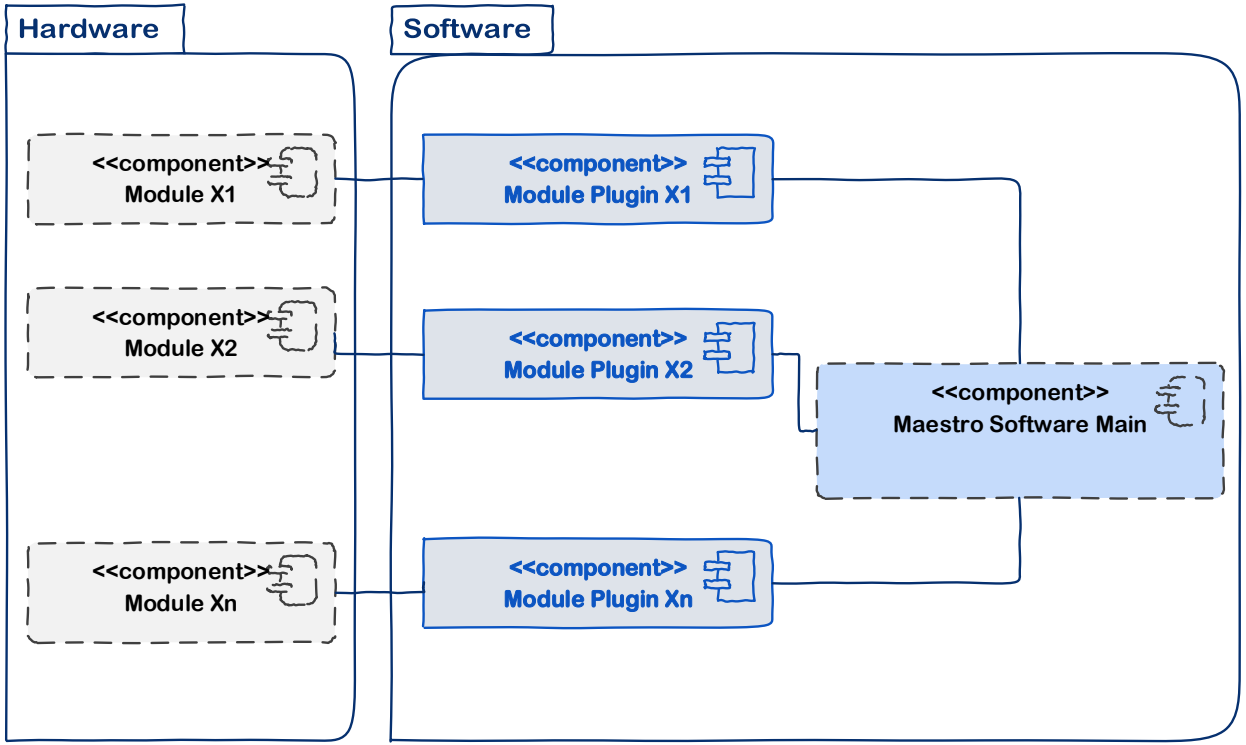


Every plugin is a dynamic library (.so or .dll)

For **each** hardware module we have its software **adaptor**

The plugins should simply be placed into a specific directory

Implementation – Maestro Software

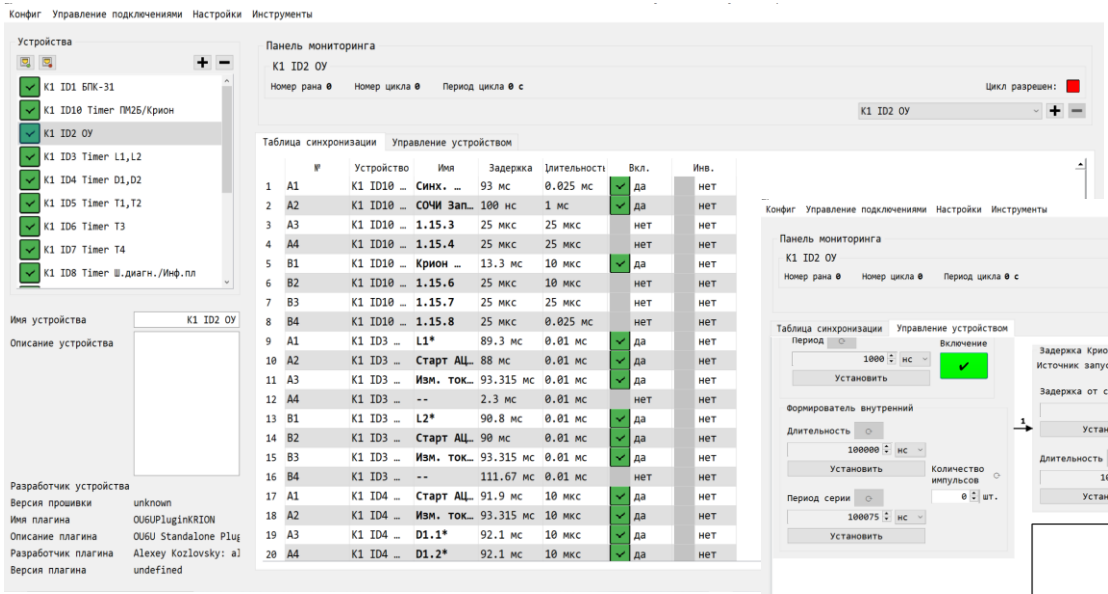


Every plugin is a dynamic library (.so or .dll)

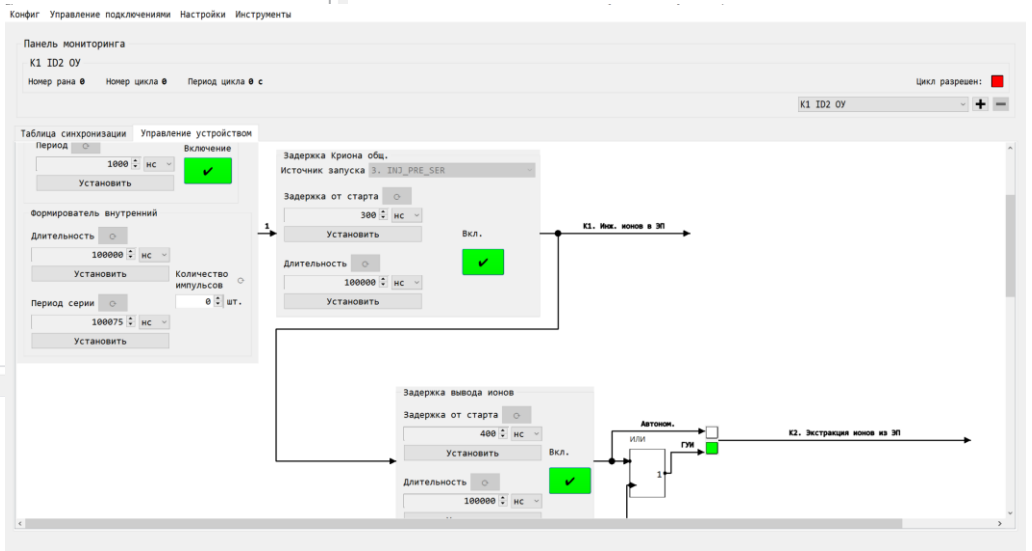
For **each** hardware module we have its software **adaptor**

The plugins should simply be placed into a specific directory

Implementation – **Maestro Software**



Software examples from KRION and HILAC Synchronization Systems

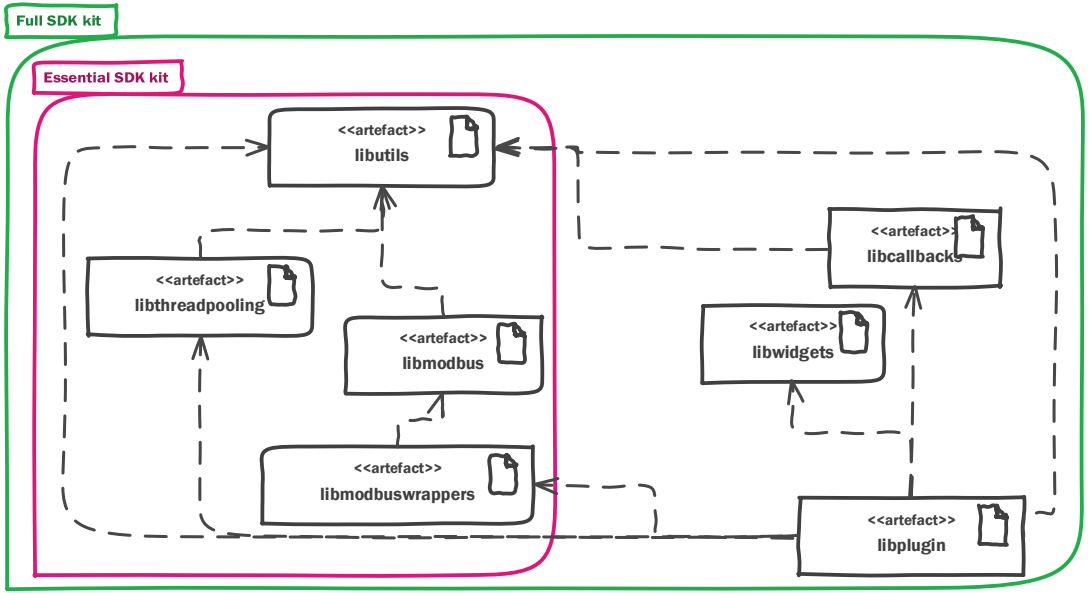


Challenges

As we built **our SDK** with **7** libraries there were an issue with the complexity of maintaining which consumes a lot of time resources

For solving this problem we implemented our **own** self-hosted conan **package manager***

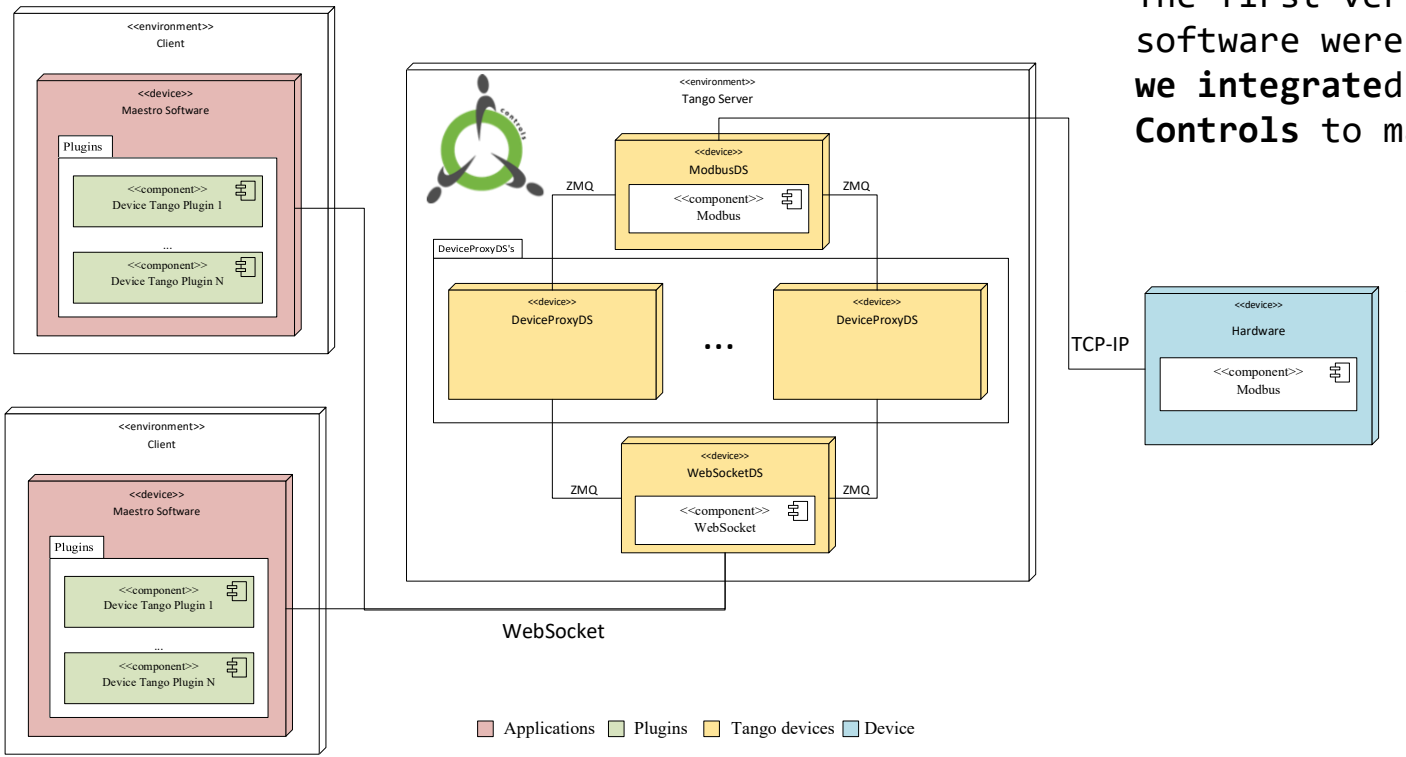
* See the talk of Denis Ivliev



JFrog

CONAN 2.0
C/C++ Package Manager

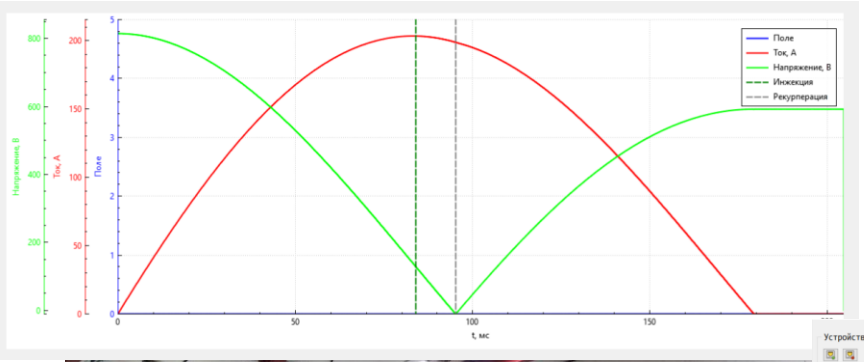
Challenges



The first versions of our software were standalone, but **we integrated** it **with Tango Controls** to make it **distribute**

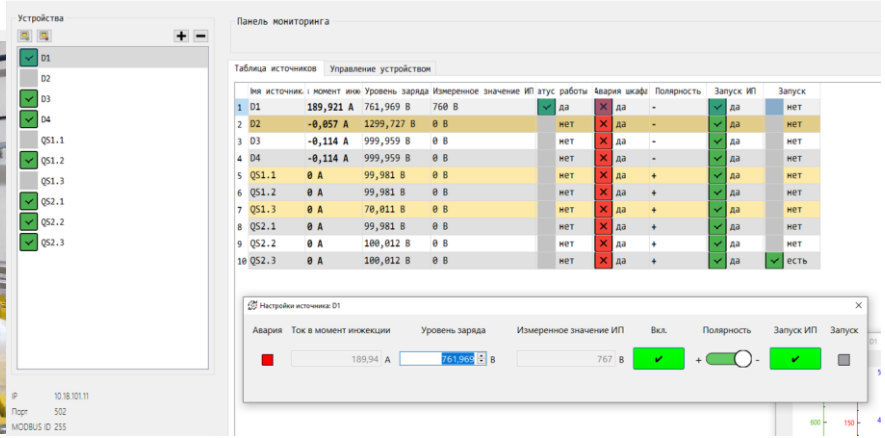
* See the talk of Alexey Bukharin

What about other systems?



Despite the initial orientation towards the synchronization system, our unified platform can be customized to fit the aims of controlling the power supplies of the **Nuclotron-Collider Beam Transport Channel**

* See the talk of Artyom Noskov



Have we satisfied our QA?

Have we satisfied our QA?

Integration 

We have integrated our software with Tango Controls, and it may be used with various hardware

Have we satisfied our QA?

Integration 

We have integrated our software with Tango Controls, and it may be used with various hardware

Evolutionary 

We have adapted the software for Nuclotron Collider Beam Transport Channel Power Supplies control and our system has grown functional

Have we satisfied our QA?

Integration 

We have integrated our software with Tango Controls, and it may be used with various hardware

Evolutionary 

We have adapted the software for Nuclotron Collider Beam Transport Channel Power Supplies control and our system have grown functionally

What about SIMPLICITY?

School of Engineering for LHEP

School of Engineering
Dubna University

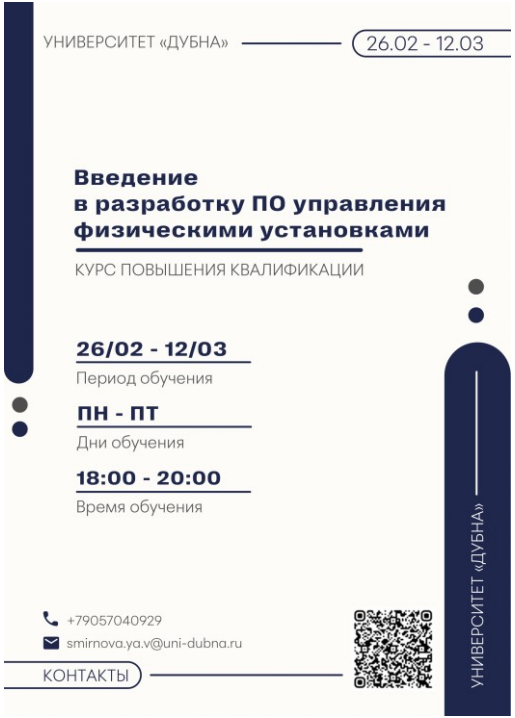
We have employed three students from the School of Engineering 1.5 year ago and now they are **indispensable** members of our team

School of Engineering for LHEP

School of Engineering
Dubna University

We have employed three students from the School of Engineering 1.5 year ago and now they are **indispensable** members of our team

Educational course has been developed for engaging the students into the process of slow control system software development



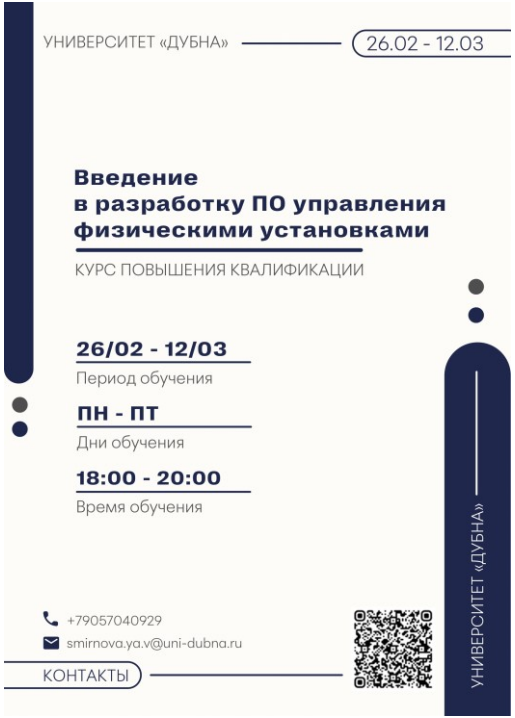
School of Engineering for LHEP

School of Engineering
Dubna University

We have employed three students from the School of Engineering 1.5 year ago and now they are **indispensable** members of our team

Educational course has been developed for engaging the students into the process of slow control system software development

Maestro Software Platform architecture has been built on such principles that have made it possible to describe the future plugin easy even for students



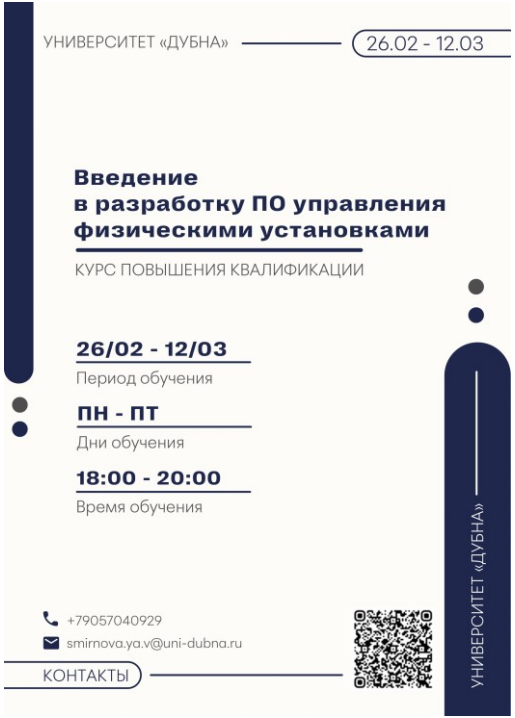
School of Engineering for LHEP

School of Engineering
Dubna University

We have employed three students from the School of Engineering 1.5 year ago and now they are **indispensable** members of our team

Educational course has been developed for engaging the students into the process of slow control system software development

Maestro Software Platform architecture has been built on such principles that have made it possible to describe the future plugin easy even for students



It has become possible to delegate certain tasks of developing software for hardware module control to a student!

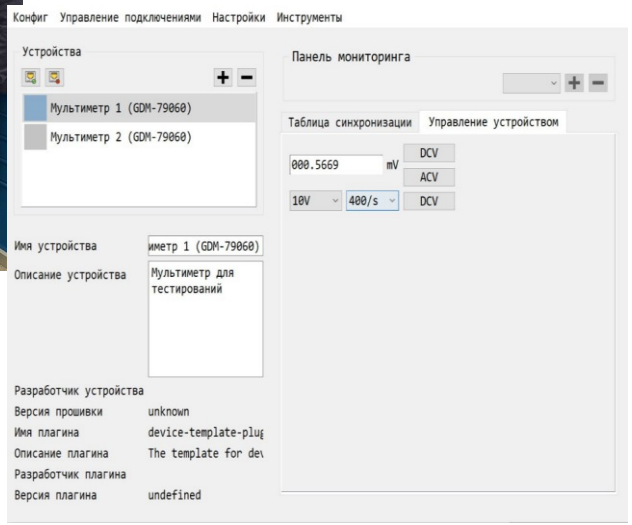
School of Engineering for LHEP

School of Engineering
Dubna University

The new generation of SoE students now is working on the software for Measure Unit Hardware control

School of Engineering for LHEP

School of Engineering
Dubna University



The new generation of SoE students now is working on the software for Measure Unit Hardware control

They are developing the **Maestro Software plugins** with **SCPI** commands support by **themselves**

Results

The unified platform for client or standalone software for slow control system management has been developed and implemented in different parts of NICA Complex

The software has been integrated with Tango Controls

The system of Maestro Software Plugins **description** has been composed and successfully tested within the framework of the **School of Engineering**

School of Engineering
Dubna University

Thank you for your attention

Alexey Kozlovsky kozlovsky@jinr.ru