
# Julia

Brain
Computer
**FIELDBUS**
Interface

Ing. Nicola Urbano
email: info@hrk-brk.com
Website: www.hrk-brk.com

# Julia

THE FIRST **BRAIN COMPUTER FIELDBUS INTERFACE** ON THE MARKET

- A universal native *Fieldbus Slave* born with the goal of being used in every sector (e.g. industrial, building automation, medical, etc).
- It collects biomedical signals in a synchronized manner using Ethernet Deterministic Fieldbus.
- Embedded with modularity that allows integration of more than one slave at at time whether on the same network or different networks using synchronized protocols such as PTP 1588, TSN, etc.
- Offers analysis, control, and diagnostics of a single or multi-user scenarios.

# Fieldbus Interface

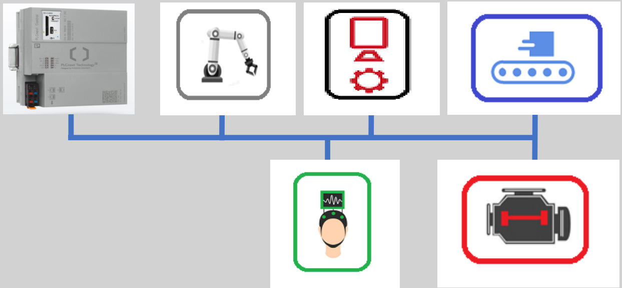
Compatible with industry standards and leading fieldbus technologies such as:

- ✓Profinet
- ✓EtherCAT,
- ✓EtherNetIP
- ✓Sercos III
- ✓PowerLink
- ✓RT-Ethernet, MQTT, OPC UA
- ✓Varan
- ✓CC-Link

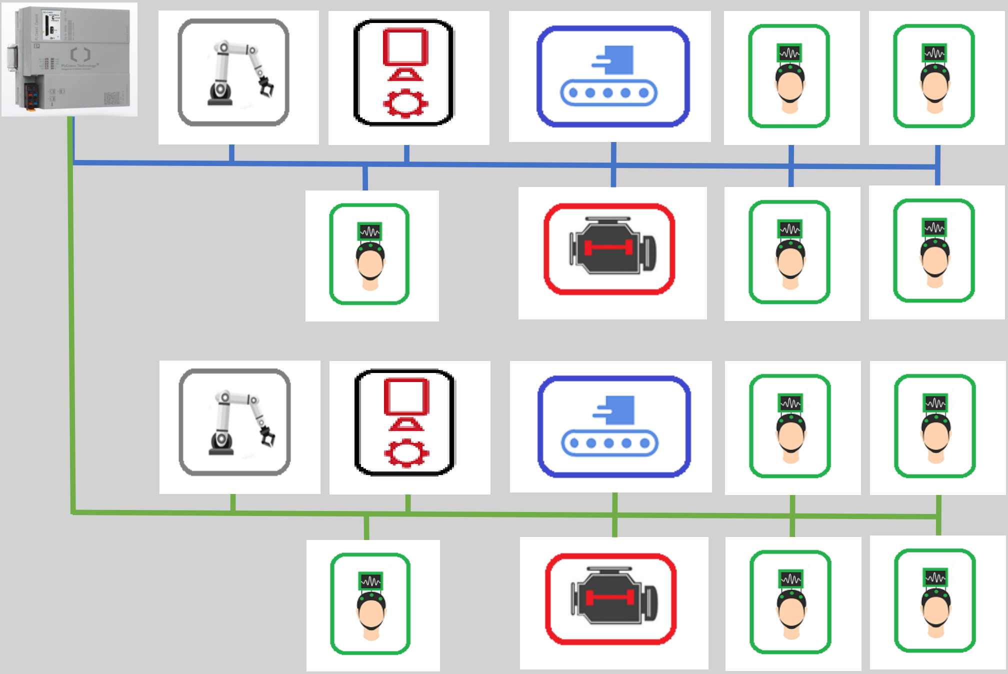
# Fieldbus Interface

Example Scenarios

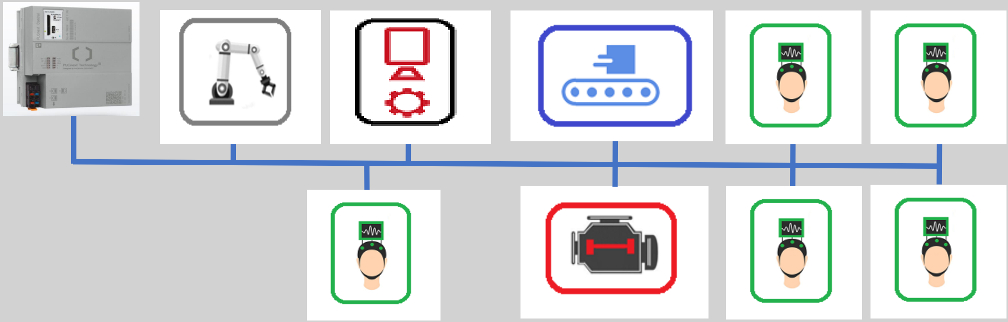
## Controller - Single User - Single Fieldbus



## Controller - Multiuser - Multiple Fieldbus



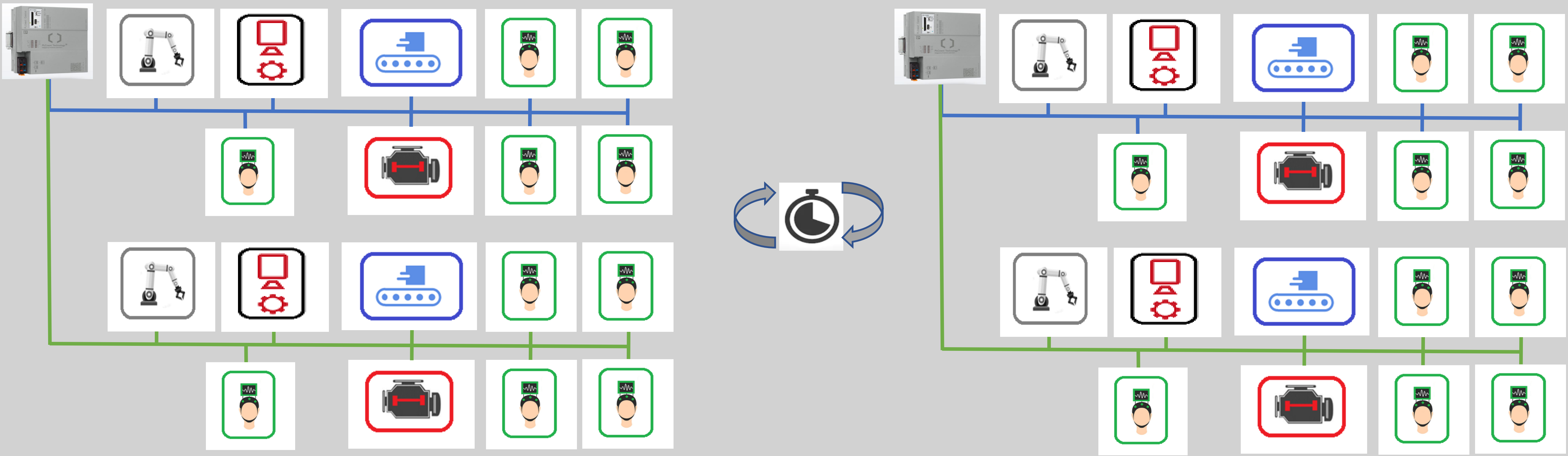
## Controller - Multiuser - Single Fieldbus



# Fieldbus Interface

Example Scenarios

## Multi Controller - Multiuser - Multiple Fieldbus - Same Plant

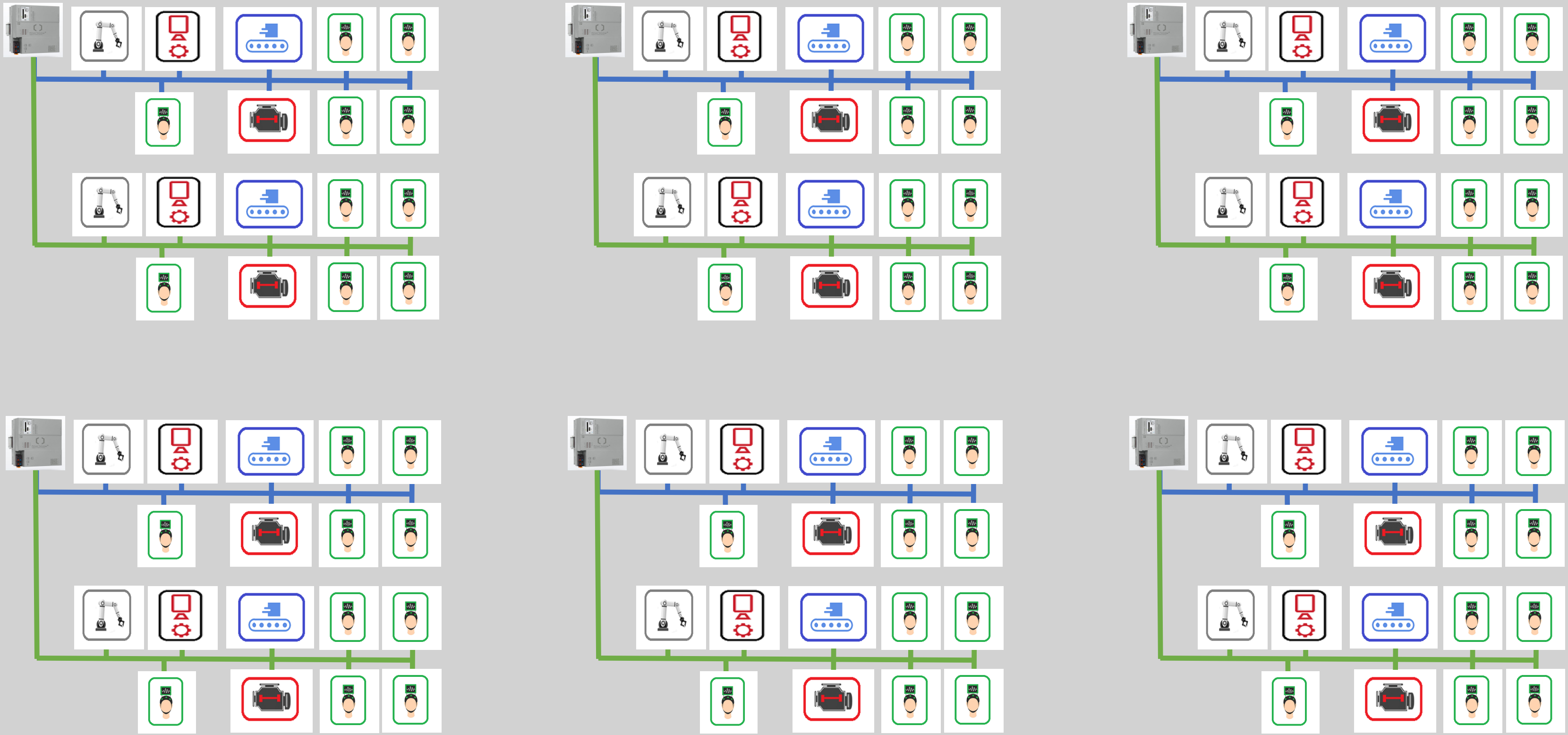


Using PTP 1588 Synchronization

# Fieldbus Interface

Example Scenarios

## Multi Controller - Multiuser - Multiple Fieldbus - Multiple Plants



PTP 1588
Synchronization

# Julia Features

Up to 32 Low-Noise PGAs and 32 High-Resolution Simultaneous-Sampling ADCs

- Input-Referred Noise: 1 µVPP (70-Hz BW)
- Input Bias Current: 300 pA
- Data Rate: 250 SPS to 16 kSPS
- CMRR: –110 dB
- Programmable Gain: 1, 2, 4, 6, 8, 12, or 24
- Unipolar or Bipolar Supplies:
  - Analog: 4.75 V to 5.25 V
  - Digital: 1.8 V to 3.6 V
- Built-In Bias Drive Amplifier, Lead-Off Detection, Test Signals
- Built-In Oscillator
- Internal or External Reference
- Flexible Power-Down, Standby Mode
- Operating Temperature Range: –40°C to +85°C

# Julia Features

- Power Supply 24 V
- Din Rail support
- Debug Connector to monitoring Fieldbus Synchronization Performances such as:
  - ✓ Synchronization ISR
  - ✓ SPI Clock
  - ✓ SPI MOSI
  - ✓ SPI MISO
  - ✓ SPI DATA
- Two plugin DB37 female connectors for Positive and Negative Inputs
  - ✓ 32 pins dedicated to signals
  - ✓ 2 pins as reference SBR1 SBR2
  - ✓ VOUT noise cancelling
- <u>Every single channel is independent and can be parametrized during the runtime</u>

# Julia Features

- Support standard profile for different slave device (e.g. robots, motor drives, vision system, multiple digital signals, etc) and can be integrated to work with Julia.
- Established ATEX and SAFETY levels and no further hardware customizations are needed for operation.
- No gateway required to communicate with other industrial field systems and devices.
- Zero latency added to the communication protocols.
- Suitable for medical applications and can be used to extend them without any limitations and enhance the usage of those applications.

# Julia: Signals

Sample Human Biomedical Signal Analysis

### Electrocardiography

Electrocardiogram
Heart Rate
Baroreflex Sensitivity
Interbeat Interval
Heart Rate Variability
PRQ Interval
QRS Width
QT Interval
R-R Interval
Respiratory Sinus Arrhythmia
Spectral HRV
Time-Domain HRV
Very High Freq Power Band
Very Low Freq Power Band
Cardiac Output
Cardiac Work
Left Ventricular Ejection Time

### Electrodermal Activity

Psychophysiology
Electrodermal Activity (EDA)
Phasic EDA
Skin Conductance Response
Skin Conductance Level

### Electroencephalography

Electroencephalogram
Alpha
Beta
Delta
Gamma
Full-band EEG
Auditory Evoked Potential
Event Related Potential
Sensory Evoked Potential
P50, N100, P200, N200, P300, N300, N400, P600 ERP [N/P] Tests
OEP Olfactory Event Related Potential
SEP Somatosensory Evoked Response
VEP Visual Evoked Potential
VER Visual Evoked Response
EEG Seizure
Cognitive State
Stress

### Electromyography

Electromyogram
Fatigue
Maximum Voluntary Contraction
Total Power
Mean Power
Muscle Activation
Startle Response
Facial EMG
H-reflex Hoffman Reflex
MEPs Motor Evoked Potentials

# Julia: Signals

Sample Human Biomedical Signal Analysis

### Hemodynamics

Arterial Blood Pressure
Cardiac Output
Central Venous Pressure
Mean Arterial Pressure
Central Venous Pressure

### Metabolic Activity

Anaerobic Threshold
Cardiopulmonary Exercise Testing
Lactate Threshold
Respiratory Gas Analysis

### Plethysmography

Blood Volume Pressure
Impedance Plethysmogram
Penile Plethysmogram
Photo Plethysmogram
Pneumo Plethysmogram

### Eye Movements

Electrooculogram
Saccadic Eye Movements
Smooth Pursuit Eye Movements
Spontaneous Nystagmus

### Evoked Response

Brainstem Evoked Potential
Auditory Evoked Response
Visual Evoked Potential
Olfactory Event Related Potential
Somatosensory Evoked Response
Nerve Conduction Study

### Polysomnography

Polysomnogram
Sleep Studies

### Respiratory Activity

Apnea Time
Breaths Per Minute
Pulmonary Compliance
Exhalation Time
Inspiration Time
Pulmonary Function Testing
Respiratory Rate

### Sensory Stimulation

Peripheral Nerve Stimulation
Visual stimulation
Olfactory Stimulation
Tactile stimulation
Pain stimulation
Vagus Nerve Stimulation

# Machine Learning (ML)

Big data collections can be analyzed in real-time or in offline state. And can be easily integrated with leading Machine Learning (ML) tools such as:

# The Automation Architecture

- Julia is a slave Fieldbus that requires Fieldbus Master in order to read human body signals.
- The Fieldbus Master controller needs to be with a deterministic OS.
- It can be a PLC, SoftPLC, embedded PC, or even a standard PC with a Realtime Extension.
- Examples of such in the market are Siemens, Allen Bradley, Phoenix Contact, Omron, B&R, Schneider, Mitsubishi, Beckhoff, LSElectric.
- SoftPLC software with Realtime Extension such as Codesys with IEC 61131,  is the de facto a standard well recognized in Automation Control.
- Other Realtime systems that are ready to use are QNX, VxWorks, Micrium, Green Hills, and Free RTOS.

# The Automation Architecture

- Each Operating System has a set of programming languages to develop software.
- The architecture of Julia is not bound to a specific programming language as long as it integrates with IEC 61131.
- This means the complete structures can be continuously reviewed and updated with different technologies and environments.
- Moreover, complex projects can be modularized with different components that can cohabit and interconnect between them.
- The fieldbus integration allows also to control Robots, drives and every network peripherals following the well know DS402 profile.

# The Automation Architecture

- The area is complex but the main purpose it is to highlight how many new possibilities can be discolored.

- One of the most innovative idea that on the market today it is the PLCnext controllers.

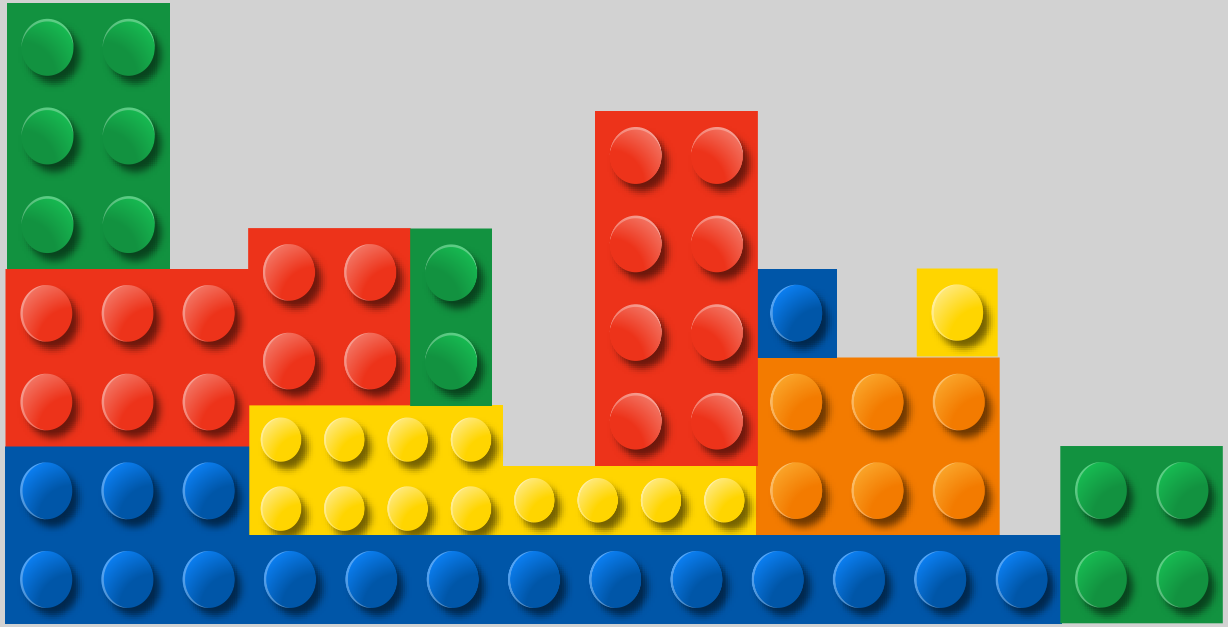
- It can be defined as the ***Fifty Shades of Controller***. It is a new vision to merge every different kind of controllers in one device to develop automation software.

- It is possible to program it as a PLC, a SoftPLC, a standalone Realtime platform in endless opportunities.

- In the next slides, we will describe PLCnext architecture and the way Julia integrates with it.

- For more information about PLCnext, you can refer to the official channel: https://www.plcnext-community.net/en/

# PLCnext System



PLCnext offers a *"Lego Architecture"* of modularity that builds layers and components on top of each other in order to create a solution that fits your exact demand.

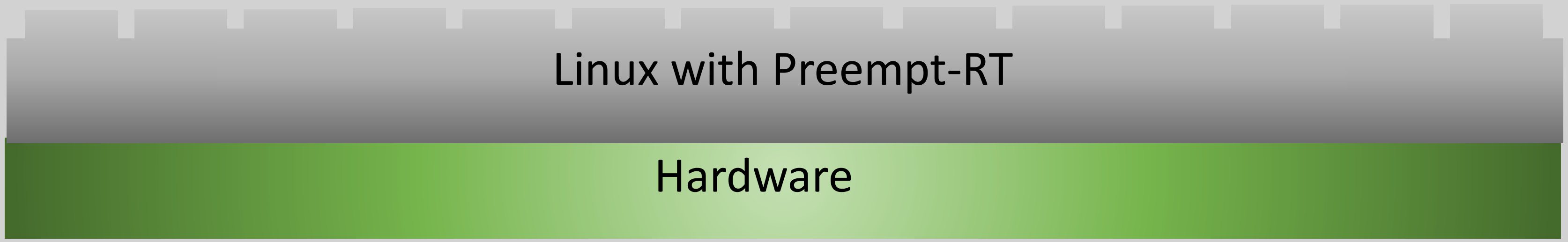
# PLCnext System Architecture

It is a scalable system device with different controllers that share the same software features

Hardware

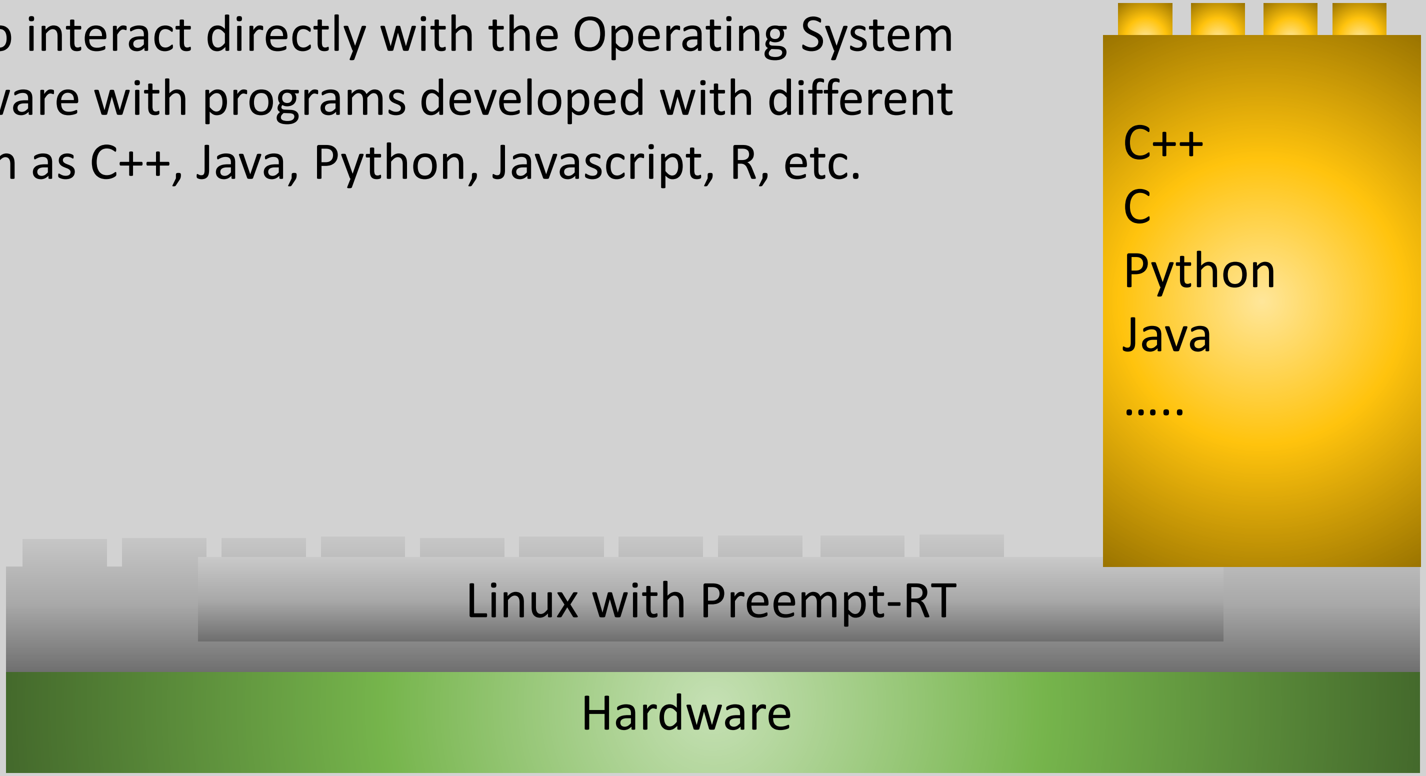
# PLCnext System Architecture

Operating System is based on Linux plus **RealTime Preempt–RT patch** extension

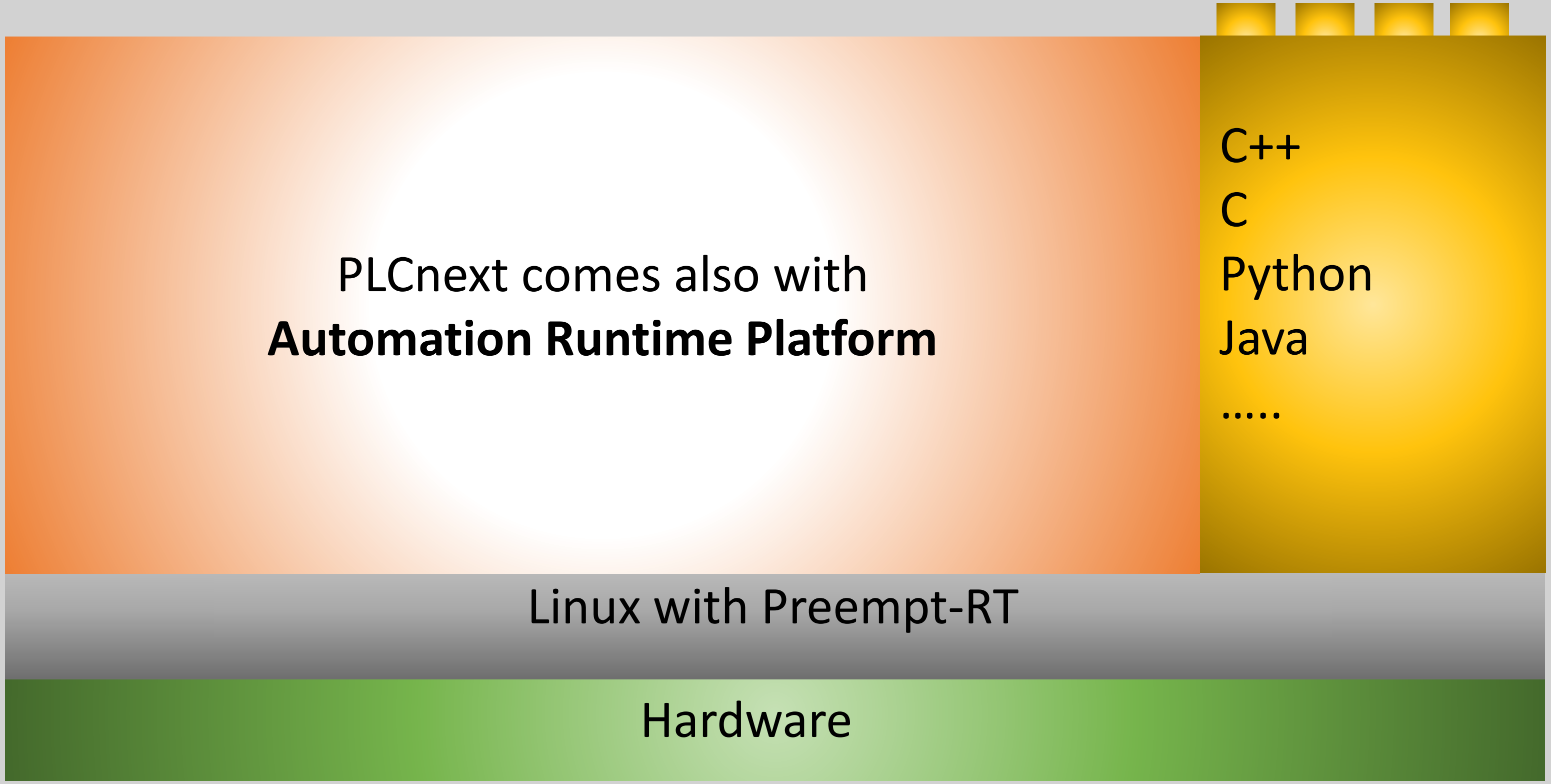


# PLCnext System Architecture

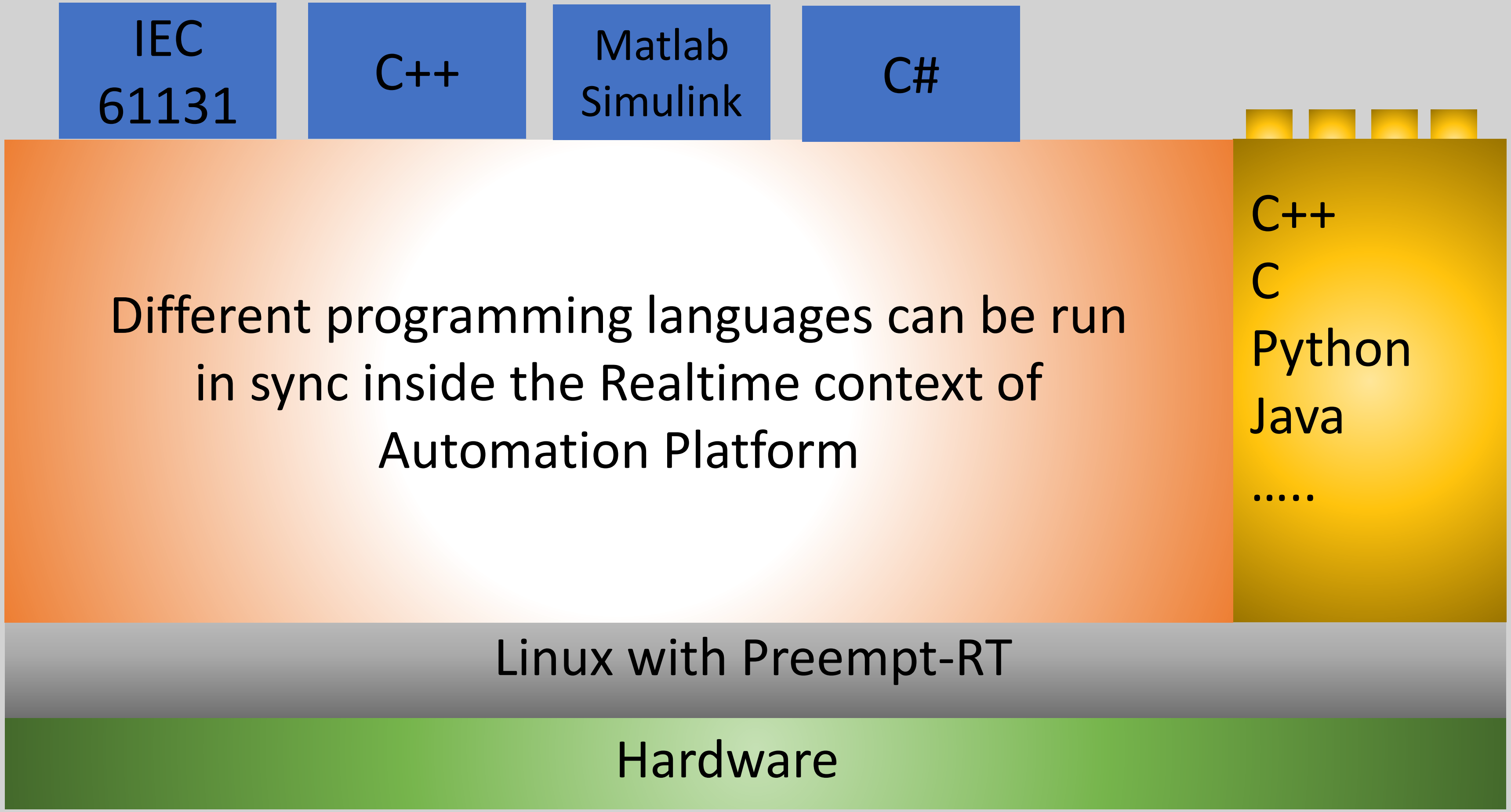
It is possible to interact directly with the Operating System and the Hardware with programs developed with different languages such as C++, Java, Python, Javascript, R, etc.



# PLCnext System Architecture



# Automation Runtime Platform



# PLCnext Engineer

The roles of PLCnext Engineer are:

- The reference and native tool to develop IEC 61131 programs
- The tool allows to import different program’s components and to execute them in the Realtime context of PLCnext Automation Framework

# PLCnext Engineer

With the plugins provided by PLCnext Automation Platform it is possible to implement code written with different tools as Visual Studio, Eclipse, Matlab and Simulink as shown below and imported ad components

| |  | | MATLAB & SIMULINK | eclipse 2020-09 |
|---|---|---|---|---|
| IEC 61131 | X | | | |
| C++ | | | | X |
| C# | | X | | |
| Matlab Simulink | | | X | |

# Automation Runtime Platform

IEC 61131

C++

Matlab Simulink

C#

ESM

When the different Programs are downloaded, they are synchronized and scheduled by ESM (Execution Synchronization Manager) component.

The different Programs don’t live isolated in their own context, the GDS (Global Data Space) allows to share variables between them in a consistent manner with the use of Ports mechanisms (Input/Output).

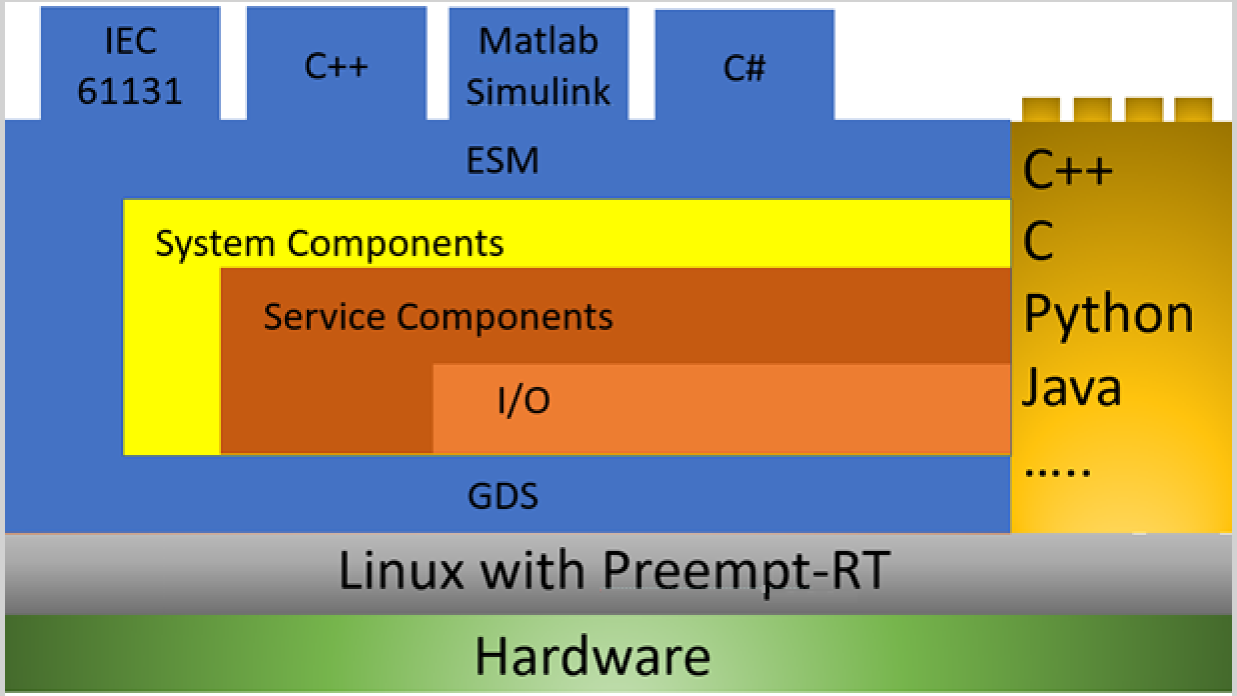
GDS

# Automation Runtime Platform

It is a galaxy of components and services to open communication with every device and system as:

- ✓ I/O access
- ✓ System Components
- ✓ Service Components
- ✓ OPC UA
- ✓ MQTT
- ✓ etc

It is only a question **how** and **who** communicates.



# Test Benchmark

- The benchmark test is based on the following architecture :
  a) The PLCnext controller : AXC F 3152
  b) A Julia Device with EtherCAT Interface
  c) A bus coupler EK1100
  d) EL2202 triggered by Sync Manager
  e) EL2202-0100 with DC
  f) PC to store and visualize the data powered with Ubuntu Operating System
- An EtherCAT Master was developed in order to read/write the data from Julia

Two (2) different test benches were conducted:

1. The first test was executed as an external program without any interaction with the PLCnext Automation Platform.
2. The second test was conducted by integrating it as an internal component and controlled by ESM of the PLCnext Automation Platform.

# Test Benchmark

Independent from the implementation, the EtherCAT Master always manages the network in the same way. The network parametrization shows many interesting discussion points:

- ❑ Julia maps 148 Inputs Bytes and 4 Outputs Bytes
- ❑ The configuration has a utilization factor of 4.82%
- ❑ The Frame size is 277 bytes and the duration is 24.08 µs
- ❑ The Cycle Time of the Task is set to 500 µs

Sync Manager:

| SM | Size | Type | Fla... |
|---|---|---|---|
| 0 | 128 | MbxOut | |
| 1 | 128 | MbxIn | |
| 2 | 4 | Outputs | |
| 3 | 148 | Inputs | |

PDO List:

| Index | Size | Name | Flags | SM | SU |
|---|---|---|---|---|---|
| 0x1A00 | 148.0 | Module Tx | MF | 3 | 0 |
| 0x1600 | 4.0 | Module Rx | MF | 2 | 0 |

| Frame | Cmd | Addr | Len | WC | Sync Unit | Cycle (ms) | Utilization (%) | Size / Duration (µs) |
|---|---|---|---|---|---|---|---|---|
| 0 | NOP | 0x0000 0x0900 | 4 | | | 0.500 | | |
| 0 | ARMW | 0xfffe 0x0910 | 4 | | | 0.500 | | |
| 0 | LRD | 0x09000000 | 1 | | | 0.500 | | |
| 0 | LWR | 0x01000000 | 1 | 1 | EI2202 | 0.500 | | |
| 0 | LWR | 0x01000800 | 1 | 1 | EI2202-DC | 0.500 | | |
| 0 | LWR | 0x01001000 | 4 | 1 | Julia | 0.500 | | |
| 0 | LRD | 0x01001800 | 148 | 1 | Julia | 0.500 | | |
| 0 | BRD | 0x0000 0x0130 | 2 | 4 | | 0.500 | 4.80 | 277 / 24.08 |
| | | | | | | | 4.82 | |

- Box 1 (Julia 32)
  - Module Tx
    - Status Registers
    - Status 1
    - Value_1
    - Value_2
    - Value_3
    - Value_4
    - Value_5
    - Value_6
    - Value_7
    - Value_8
    - Status 2
    - Value_9
    - Value_10
    - Value_11
    - Value_12
    - Value_13
    - Value_14
    - Value_15
    - Value_16
    - Status 3
    - Value_17
    - Value_18

# Test Benchmark

- A complex configuration with nine Julia devices will need one Ethernet Frame, 1468 Bytes and a Duration of 119.36 µs
- There is still room to extend this to many devices and possibly run with multicore or multi controllers.


| Frame | Cmd | Addr | Len | WC | Sync Unit | Cycle (ms) | Utilization (%) | Size / Duration (µs) |
|---|---|---|---|---|---|---|---|---|
| 0 | NOP | 0x0000 0x0... | 4 | | | 0.500 | | |
| 0 | ARMW | 0x0000 0x0... | 4 | | | 0.500 | | |
| 0 | LRD | 0x09000000 | 2 | | | 0.500 | | |
| 0 | LWR | 0x01000000 | 36 | 9 | <default> | 0.500 | | |
| 0 | LRD | 0x01000800 | 1332 | 9 | <default> | 0.500 | | |
| 0 | BRD | 0x0000 0x0... | 2 | 9 | | 0.500 | 23.86 | 1468 / 119.36 |
| | | | | | | | 23.87 | |

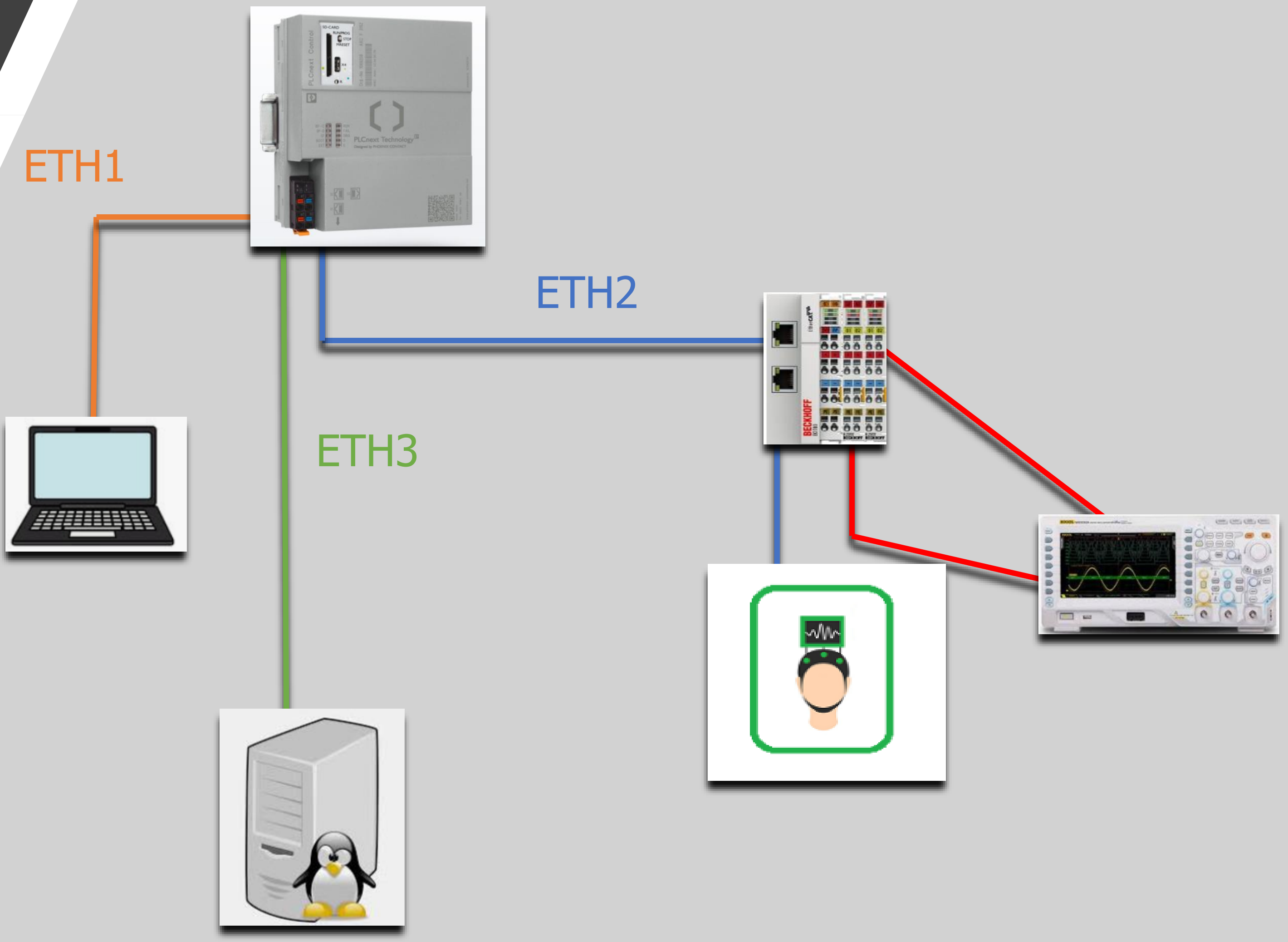
# PLCnext AXC F 3152

| | |
|---|---|
| **Processor** | Intel® Atom™ E3930 Dual Core – 2 x 1,3 GHz |
| **Operating system** | Linux |
| **RAM** | 2048 Mbyte |

| | |
|---|---|
| **Amount of process data** | max. 8192 Bit (per station) |
| | max. 4096 Bit (Axioline F local bus (input)) |
| | max. 4096 Bit (Axioline F local bus (output)) |
| **Number of supported devices** | max. 63 (per station) |
| **Number of local bus devices that can be connected** | max. 63 (observe current consumption) |
| **Program memory** | 16 Mbyte |

| | |
|---|---|
| **Engineering tool** | PLCnext Engineer |
| | Eclipse |
| **Program memory** | 16 Mbyte |
| **Mass storage** | 32 Mbyte |
| **Retentive mass storage** | 1 Mbyte |
| **Realtime clock** | Yes |

# The Network Topology




The AXC F 3152 has three independent Ethernet Ports. They were implemented in this way

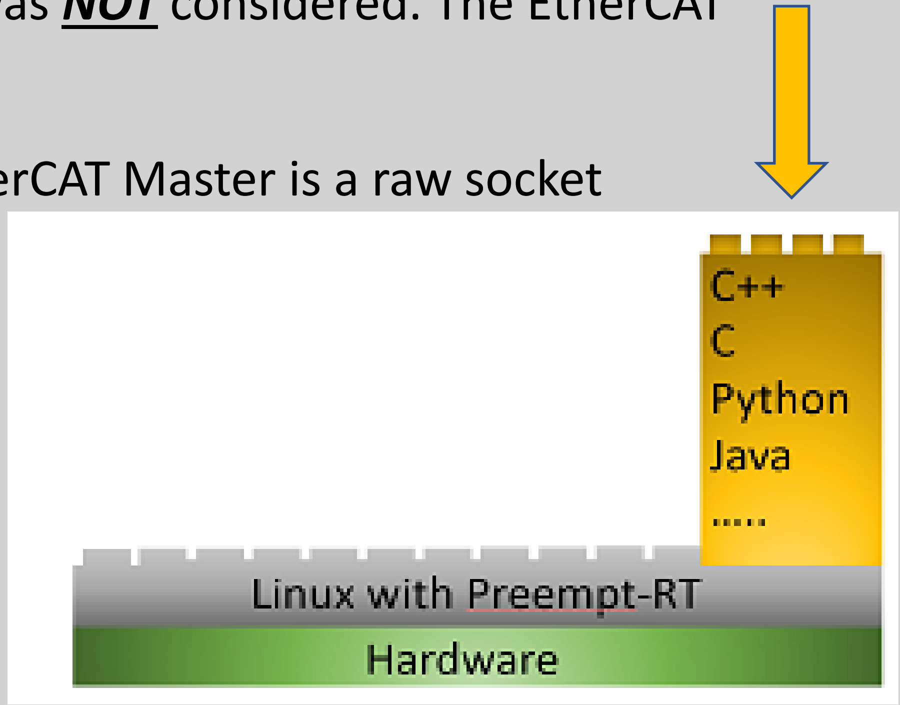
ETH1 : local services as SSH connection, FTP client, web server

ETH2 : UDP communication to read all the process data collected by the EEG's probes and delivered to PC Supervisor

ETH3 : EtherCAT Master to control the whole fieldbus networks

# External Process

- The first test benchmark concerns to use an **External Process** running in Linux Preempt-RT context.
- In this particular case the PLCNext automation Framework was ***NOT*** considered. The EtherCAT Master used was an executable and standalone component.
- In order to execute it, we must have root privileges. The EtherCAT Master is a raw socket communication channel.
- The raw socket are do not follow a standard protocol such as UDP or TCP.
- The figure shows the application is in yellow context.
- The scheduling is based on the POSIX schema.



# External Process

From the remote SSH shell :

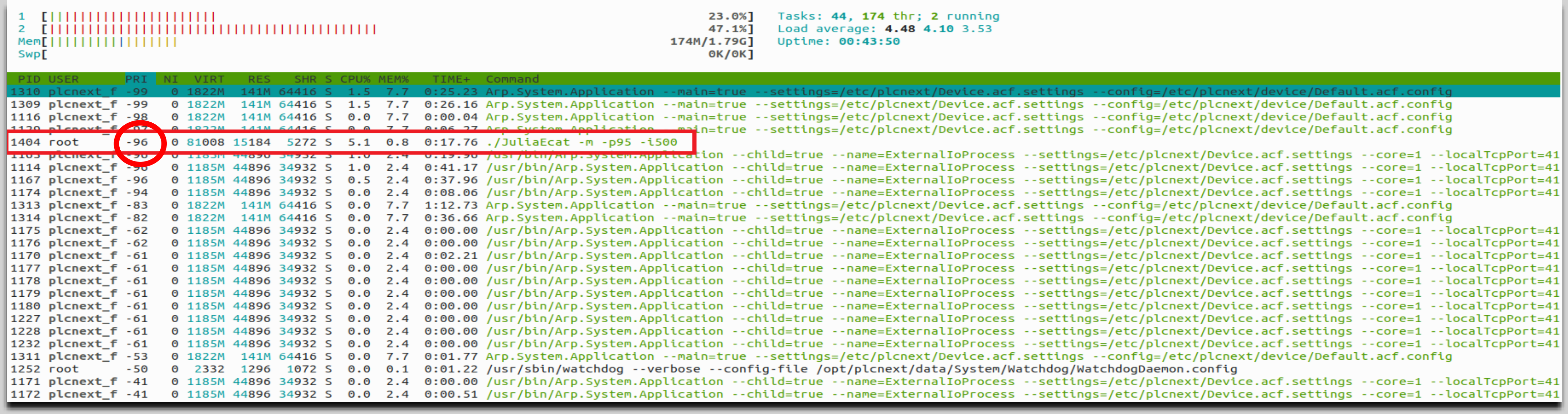
**root@axcf3152: /opt/nc/# ./JuliaEcat  -m –p95 –i500 – M**

- The command to execute the process it is also filled with some startup parameters
- There are many options to improve the performance according the startup values.

| | |
|---|---|
| p95 | sets the priority level, in this case for test purpose a high priority (95) was selected |
| m | lock current and future memory allocations |
| i500 | base interval of thread in µs |
| M | delay updating the screen until a new max latency is hit |

# External Process

- To verify the real performances of the application we used different methods: software and hardware.
- The software performances results were verified to meet the expectations.
- ***htop*** was used to check to the scheduling position of the process mapping in the operating system.



# External Process

- PR is the priority level. The lower the PR, the higher the priority of the process will be given.

- PR is calculated as follows:

| | |
|---|---|
| Normal Processes | 20 + NI (NI is the value of nice and ranges from -20 to 19) |
| Real Time Processes | PR = - 1 - real_time_priority (real_time_priority ranges from 1 to 99) |

According to the formula **PR = -1 - 95 = -96** , the value, showed by ***htop***, matches the startup settings shown in the previous slide.

# External Process

The snippet below shows the values copied during the runtime:

- The Minimum Cycle Time is 5 µs
- The Average Cycle Time is 8 µs
- The Max Cycle Time is 70 µs
- The Number of executed cycles were 238033324 => 3.3h of continuous running

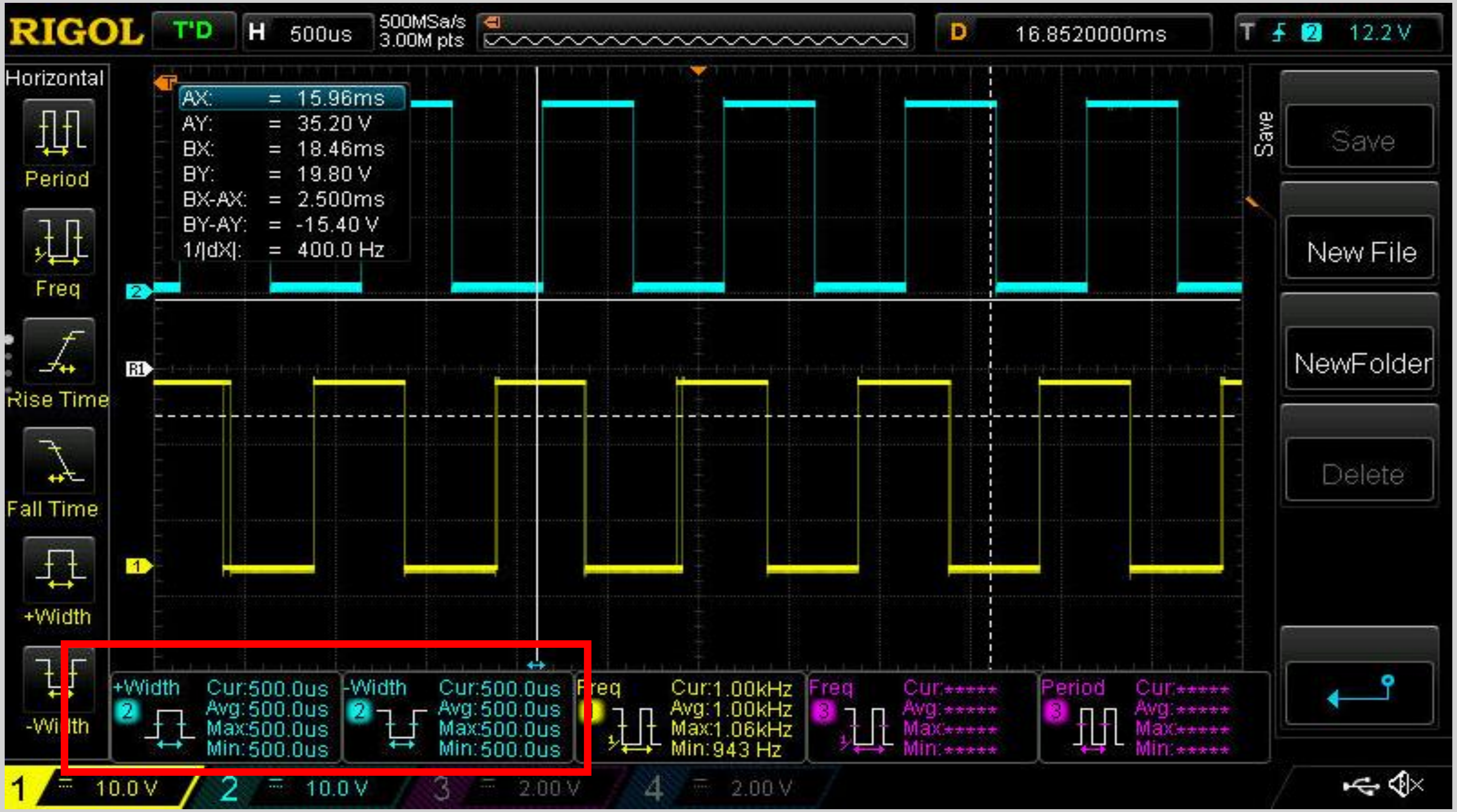
```
T: 0 ( 1307) P:95 I:500 C:23803324 Min:      5 Act:    8 Avg:    8 Max:      70
```

# External Process

The Oscilloscope compares the El2202 and DC Clock of Julia. The Yellow Line (CH1) is relative to EL2202 and the Blue Line (CH2) EL2202-0100

- CH1 shows jitter values of 70 µs, these values are in accordance with the software value shown before.
- There is a gap between the CH1 and CH2, but this is intentionally set during the startup. It is possible to specify a DC offset for each single slave in order to avoid that the controller's jitter that can influence the data consistency.

# External Process



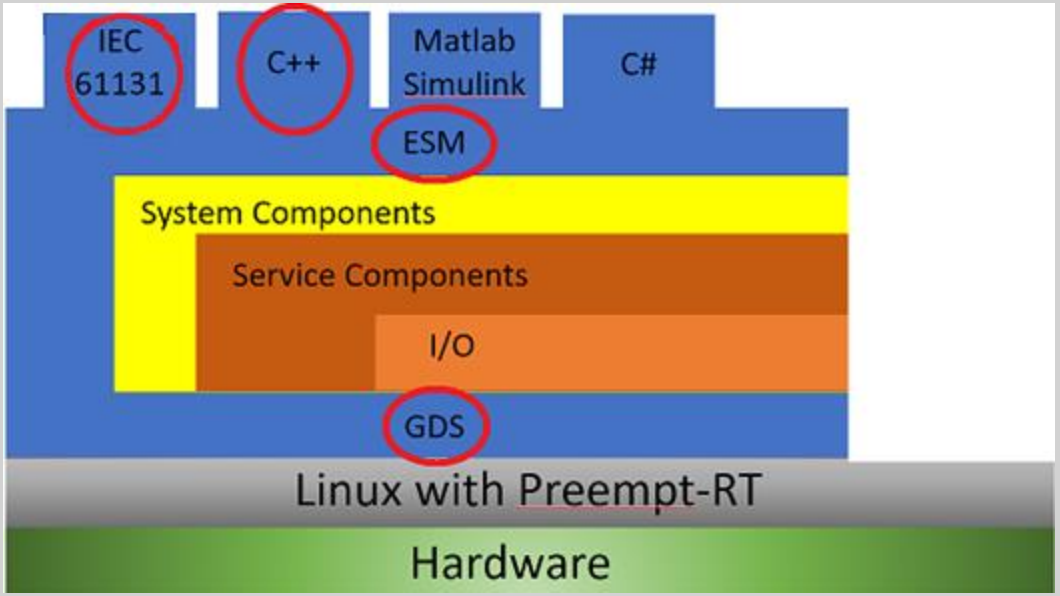

To visualize these values the Program toggles both digital Outputs every cycle time. In this way the waves are generated in square visuals and can be compared easily

# Internal Program

- The second test benchmark was implemented by importing (Eclipse module) as an internal PG unit (C++).
- PLCnext Engineering developed the whole logic to control the scheduling of tasks.
- The programs are scheduled with two different tasks, because the requirements are also different.
- The first Task is set to 500 µs and controlled by the first CPU's core, it manages the EtherCAT Master module, and it has in charge the copy of the data from the Julia device to PLCnext.

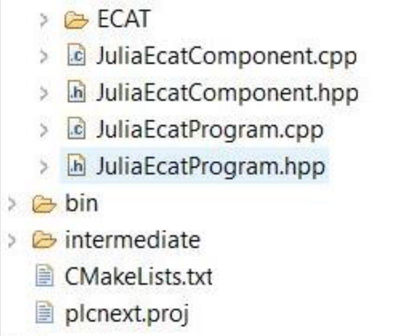
The second task was set to 4ms and triggered by the second CPU's core.

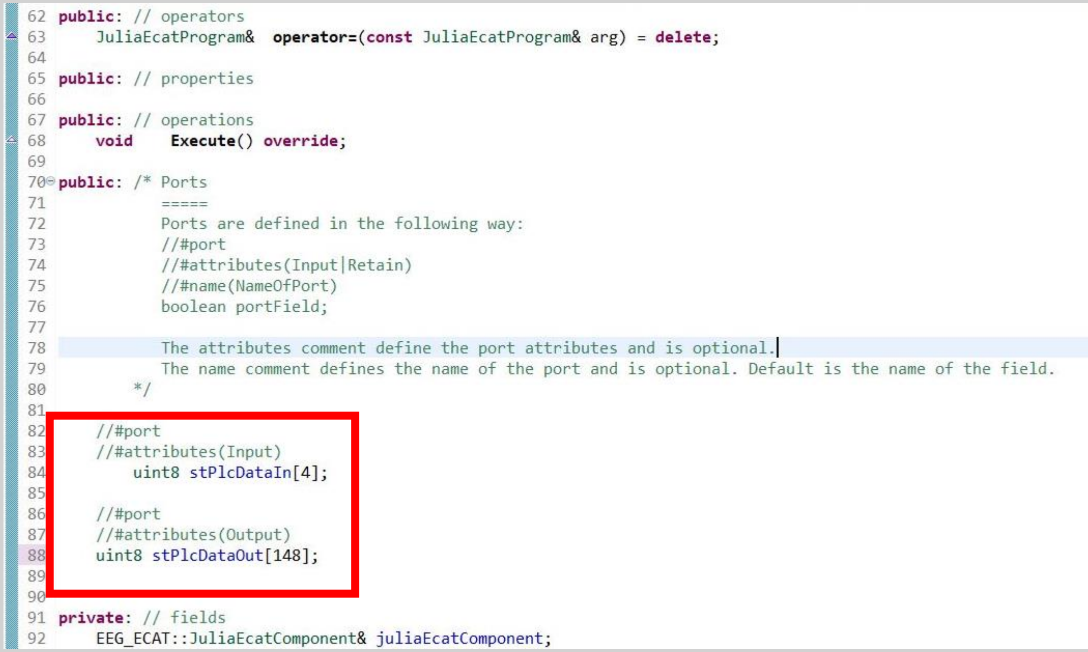
The priority is lower than first Task and it controls the UDP communication.



# Internal Program

- The program was developed with Eclipse and the module generated was imported to the PLCnext Engineer.
- The EtherCAT variables mapped by Julia were declared as Ports
- There are 148 Inputs Bytes and 4 Output Bytes




# Internal Program

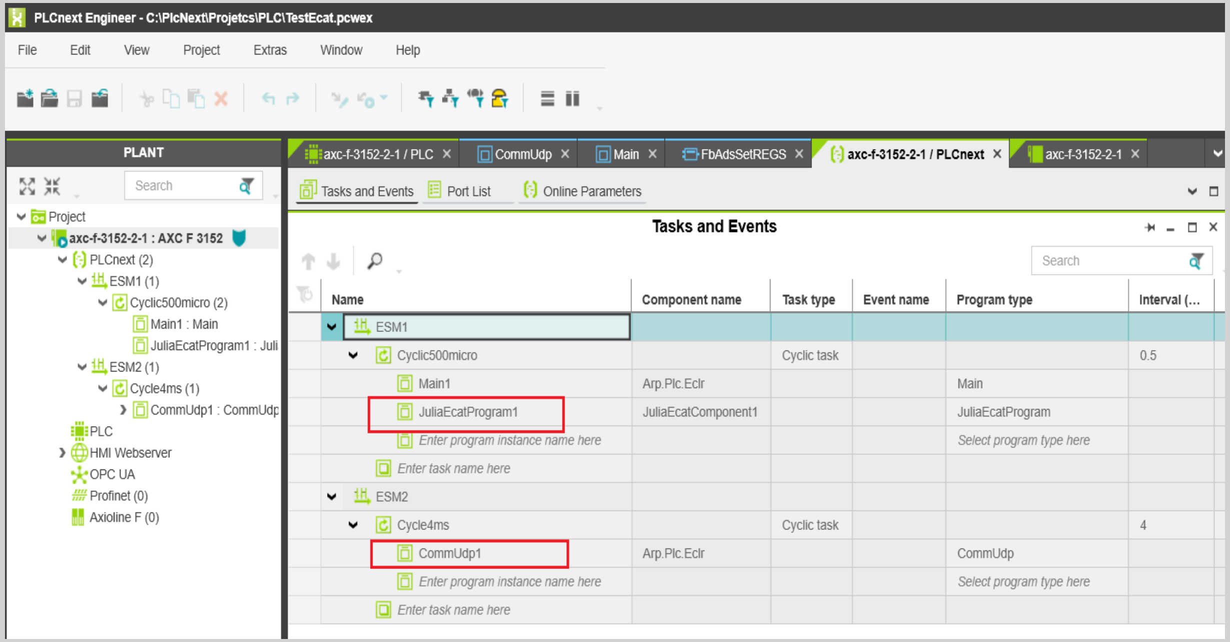
- The same mapping values were also present in PLCnext Engineer and were declared with the Ports attributes.
- Note that Port Inputs are wired with Ports Output and vice-versa.

```
public: // operators
    JuliaEcatProgram&  operator=(const JuliaEcatProgram& arg) = delete;

public: // properties

public: // operations
    void    Execute() override;

public: /* Ports
           =====
           Ports are defined in the following way:
           //#port
           //#attributes(Input|Retain)
           //#name(NameOfPort)
           boolean portField;

           The attributes comment define the port attributes and is optional.
           The name comment defines the name of the port and is optional. Default is the name of the field.
        */

    //#port
    //#attributes(Input)
        uint8 stPlcDataIn[4];

    //#port
    //#attributes(Output)
    uint8 stPlcDataOut[148];


private: // fields
    EEG_ECAT::JuliaEcatComponent& juliaEcatComponent;
```

```
ADS_Ch : STRUCT
 Status: DWORD;
 Channel: ARRAY[0..7] OF DINT;
END_STRUCT


ECAT_SLAVE_ADS : STRUCT
  TimeStamp1:DWORD;
  TimeStamp2:DWORD;

  GlobalStatusRegisters:DWORD;
  Module: ARRAY[0..3] OF ADS_Ch;
END_STRUCT


ChannelADS: ARRAY[0..147] OF USINT;
CtrlADS: ARRAY[0..3] OF USINT;
EEGDATA :array [0..7]of ECAT_SLAVE_ADS;
```

# Internal Program

Below is the review of the configurations with ESMs, priorities and cycle times




| Program Instance | ESM1 | ESM2 | Priority | Cycle Time |
|---|---|---|---|---|
| | | | | |
| Main1 | X | | 0 | 500 µs |
| JuliaEcatProgram1 | X | | 0 | 500 µs |
| CommUDP1 | | X | 1 | 4 ms |

# Internal Program

CommUdp1 opens and binds the socket with ETH2 (172.16.17.200) and listens on port 1500.

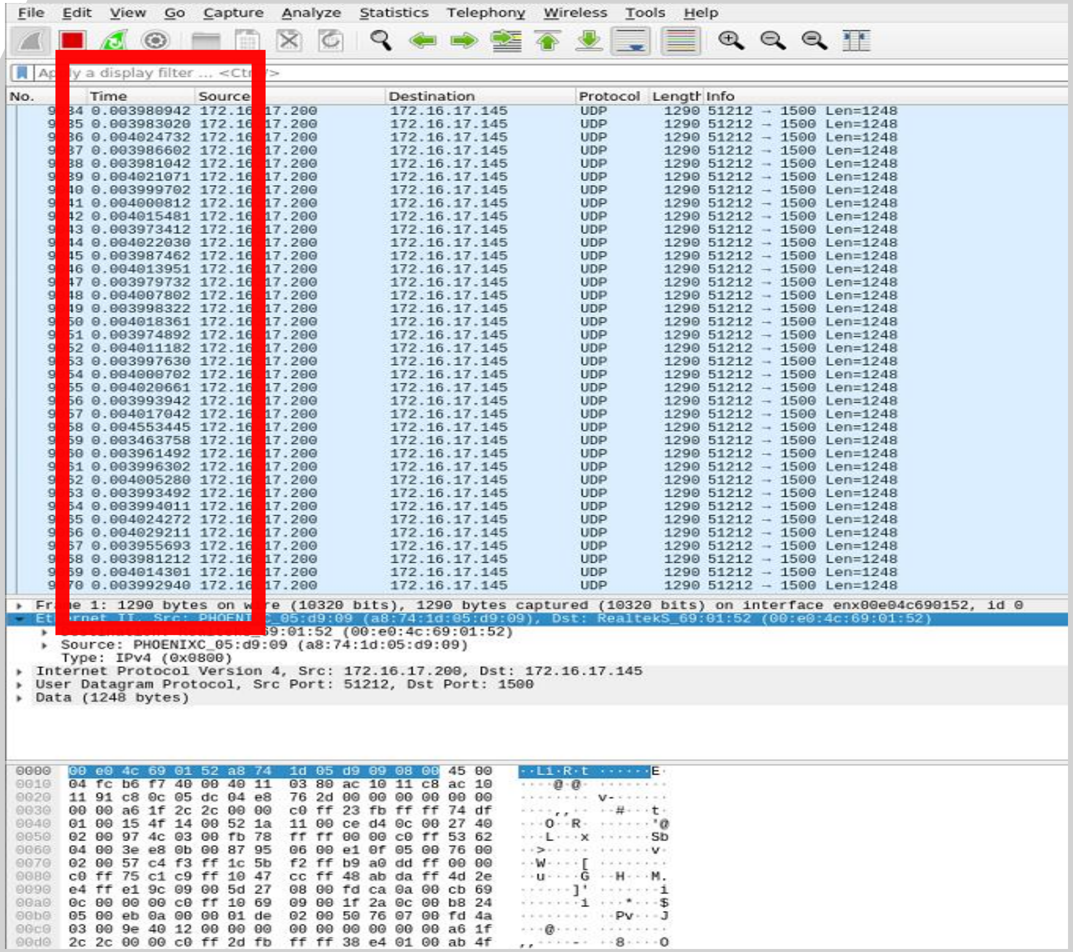
This is the Supervisor PC's ethernet address

```
(*UDP OPEN and bind*)

UDP_SOCKET1(ACTIVATE := bStartComm,
            BIND_IP := '172.16.17.200',
            BIND_PORT := 1500,
            HANDLE => wHandleUDP,
            ACTIVE => bActive,
            BUSY => bBusy,
            ERROR => bError,
            STATUS => wStatus,
            USED_PORT => uPortUsed);
```

```
if wHandleUDP > 0
THEN
UDP_SEND1(REQ := bSenUdp,
            HANDLE := wHandleUDP,
            DEST_IP := '172.16.17.145',
            DEST_PORT := 1500,
            DATA_CNT := SIZEOF(UDPOutDataAds),
            DATA := UDPOutDataAds);
            bSenUdp := FALSE;
```

If the socket is opens successfully, then it will start to deliver the data acquired by Julia and forward the Supervisor PC.

# Internal Program




- The Supervisor PC collects the data with the programmed cycle times as shown by the Wireshark's trace.
- The time column display the current value in seconds versus the previous frame.
- There is a difference of about 4ms between two consecutives UDP frames.

# Internal Program

| | | |
|---|---|---|
| [ ] ESM_INFOS | [...] | ESM_IN |
| ESM_INFOS[1] | (...) | ESM_IN |
| TASK_COUNT | 1 | UINT |
| TICK_COUNT | 0 | UDINT |
| TICK_INTERVAL | 0 | UDINT |
| [ ] TASK_INFOS | [...] | TASK_IN |
| TASK_INFOS[1] | (...) | TASK_IN |
| INTERVAL | 500 | LINT |
| PRIORITY | 0 | INT |
| WATCHDOG | 0 | LINT |
| LAST_EXEC_DURATION | 35 | LINT |
| MIN_EXEC_DURATION | 17 | LINT |
| MAX_EXEC_DURATION | 344 | LINT |

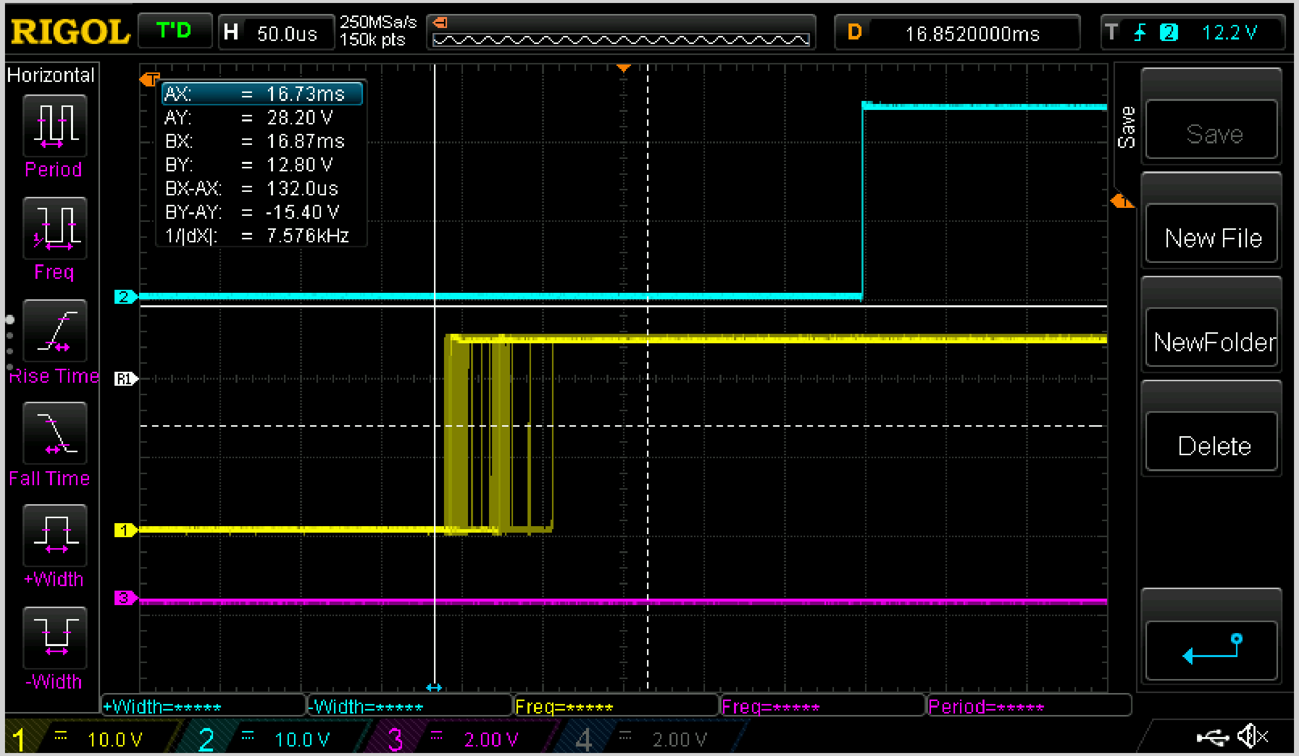
- Task2 has an average about 20 µs

- Here we show the global data structures of the Task's activities.

- Task1 has an average cycle time execution of 30 µs

| | | |
|---|---|---|
| ESM_INFOS[2] | (...) | ESM_INFO |
| TASK_COUNT | 1 | UINT |
| TICK_COUNT | 0 | UDINT |
| TICK_INTERVAL | 0 | UDINT |
| [ ] TASK_INFOS | [...] | TASK_INFO_AR |
| TASK_INFOS[1] | (...) | TASK_INFO |
| INTERVAL | 4000 | LINT |
| PRIORITY | 1 | INT |
| WATCHDOG | 100000 | LINT |
| LAST_EXEC_DURATION | 87 | LINT |
| MIN_EXEC_DURATION | 26 | LINT |
| MAX_EXEC_DURATION | 1730 | LINT |
| LAST_ACTIVATION_DELAY | 14 | LINT |
| MIN_ACTIVATION_DELAY | 9 | LINT |
| MAX_ACTIVATION_DELAY | 103 | LINT |

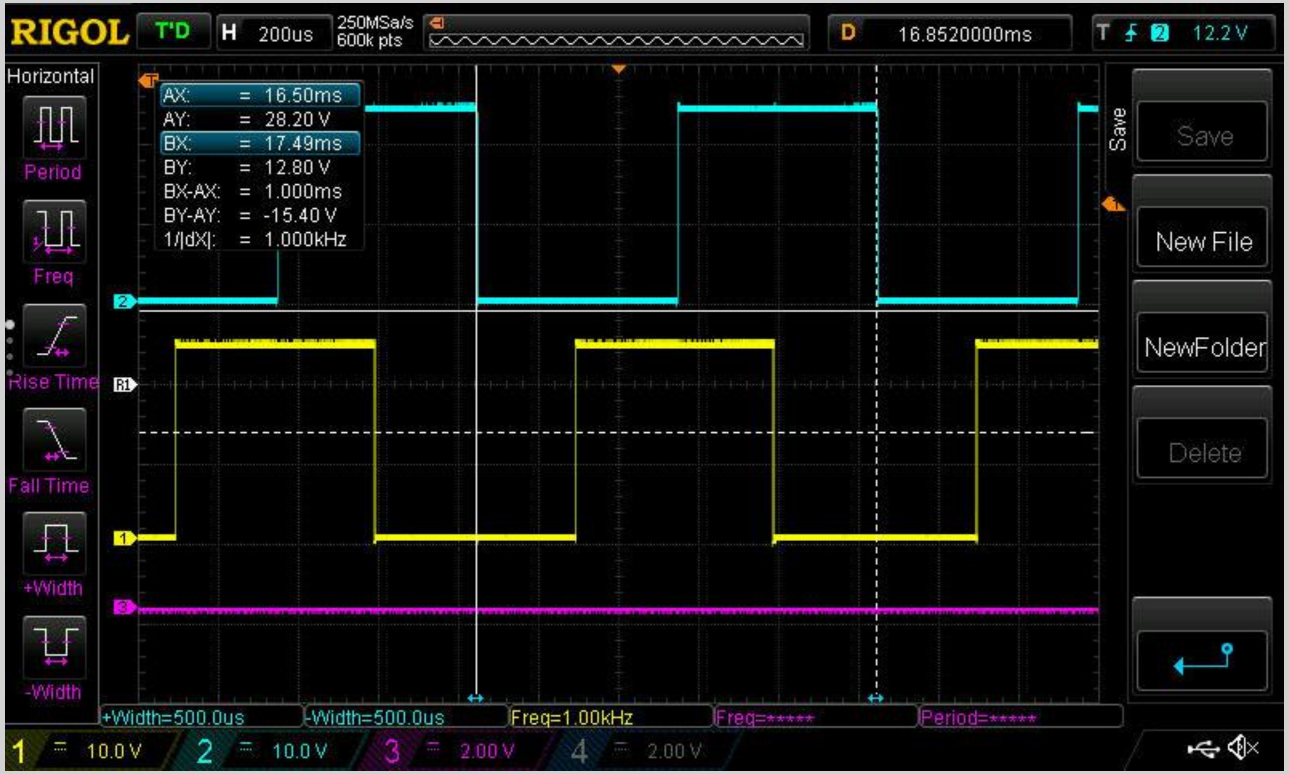
# Internal Program




- The jitter is equal to External Program applications
- These measurements compare the EL2202 Synchronized by SM and the DC of the EL2202-0100
- The jitter value was around +/- 25 µs

- The gap between EL2202 and EL2202-0100 is an offset of 250µs added during the startup phase.

# Internal Program




The Two signals show a cycle time of 500µs as planned and controlled by the task options.

# Conclusions

- This first test benchmark showed how to integrate, and measure data collected by Julia device.
- It was justified using the selection of the AXC F 3152, PLCnext by Phoenix Contact.
- The two different approches used proved the same results.
- The first implemented an EtherCAT Master as a standalone application and the second as a Program Unit controlled by the PLCnext Automation Platform.
- Credit to the Controller (AXC F 3152) that allowed many eclectics and powerful options.
- Both results fit the timing and quality required and choosing either is a matter of preference to the intended solution.

# Conclusions

- There are many different possible solutions related to the controller, but in the area of automation it is also important to select a partner that guarantees long term support and products.
- There are open source hardware platforms, but one of the cons is that for every component installed or integrated dedicate time and energy is needed.
- PLCnext has a strong community that shares different software products and experiences.
- Many controllers can be integrated with Julia, however, the risk is always there once that controller is updated or a new version was released.

# Conclusions

THE FIRST **BRAIN COMPUTER FIELDBUS INTERFACE** ON THE MARKET

- Julia is the ***FIRST DEVICE IN THE WORLD*** that it is cable to integrate user(s) and machines.
- Some papers or trials work with some motors/signals but they are limited as potential and not comparable with Julia.
- Julia is ready to operate with all fieldbus devices available on the market.
- It is more than a laboratory to analyze the human behaviors.
- It is the plug between many split worlds: Humans and Machines.

- Coming Next => … will change the mind


HRK-BRK


HRK-BRK

الحركة بركة

Ing. Nicola Urbano
email: info@hrk-brk.com
Website: www.hrk-brk.com