PLCnext Technology

PLCnext Engineer

Selección, nivel básico y ejemplos de lenguaje de alto nivel

Ingeniero Antonio Gordillo Abril 2021

PLCnext Technology Ecosystem

PLCnext Technology

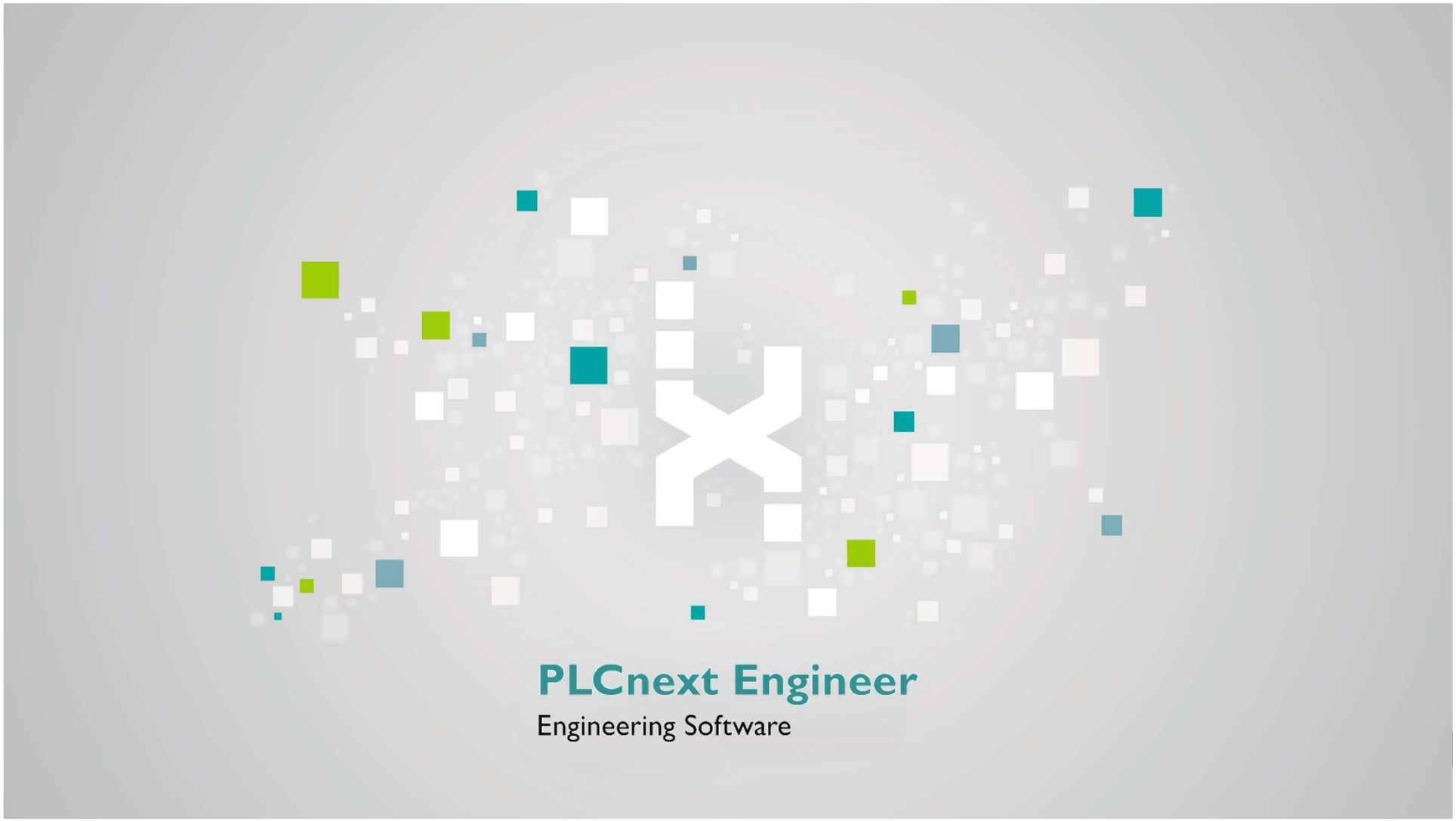
PHOENIX CONTACT
INSPIRING INNOVATIONS

Webinars

Agenda

- Selección
- Nivel Básico PLCnext Engineer
- Nivel Básico PLCnext Engineer HMI
- Lenguajes de Alto Nivel
- Herramientas

PHOENIX CONTACT
INSPIRING INNOVATIONS


PLCnext Engineer
Engineering Software

PLCnext Technology
Designed by PHOENIX CONTACT

PLCnext Engineer

PLCnext Technology Configuration and Engineering

Fast and flexible configuration

- C-Code, Simulink models, function components, IEC61131-3, Safety, HMI

Extendable

- By licensed add-ins like the Viewer for Simulink

Easy handling

- Intuitive user interface
- Clear structures

The software for configuration and engineering

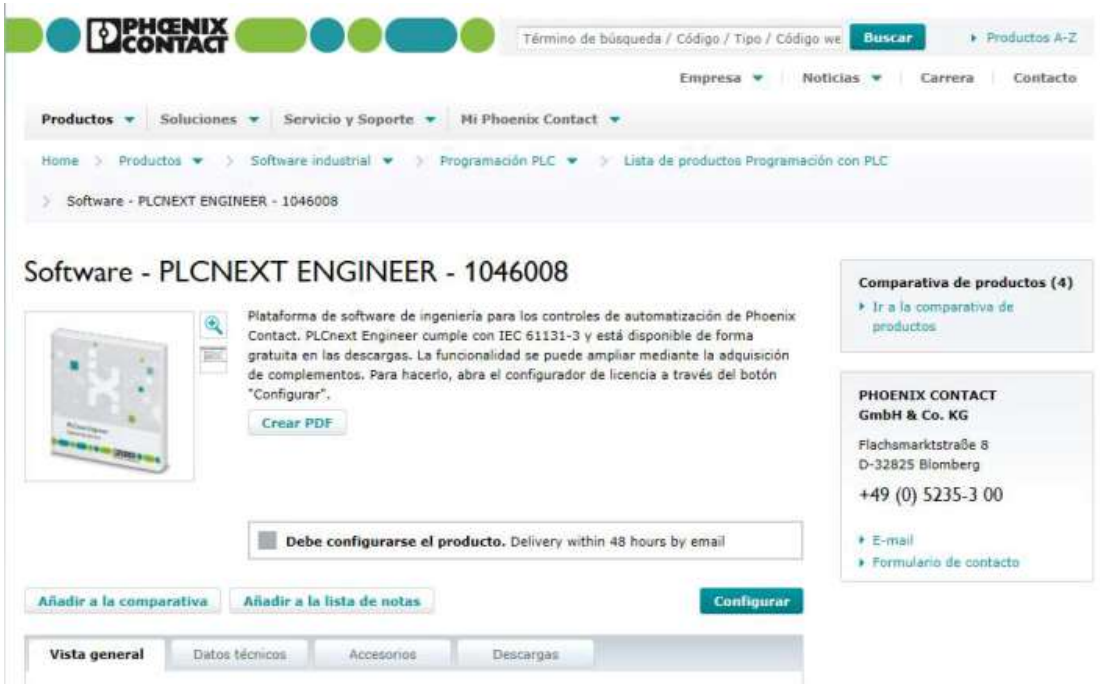
1046008

PLCNEXT ENGINEER

PHOENIX CONTACT
INSPIRING INNOVATIONS

Selección

PLCnext Engineer



INSPIRING INNOVATIONS

Area de descargas

Note Open Source

PLCnext Engineer

Change Notes

Activation Wizard

Industrial Security

Indicaciones de uso

	Descripción	Idioma	Versión
☐	[html, 307 KB] Indicaciones de uso Nota del software Open-Source PLCnext-Engineer-2020-Open-Source-Software-Notice.html	inglés	1
☐	[pdf, 1 MB] Indicaciones de uso Prod.-Id. 107913: AH DE INDUSTRIAL SECURITY Maßnahmen zum Schutz von netzwerkfähigen Geräten mit Kommunikationsschnittstellen, Lösungen und PC-basierter Software vor unberechtigten Zugriffen ah_de_industrial_security_107913_de_03.pdf	alemán	03
☐	[pdf, 142 KB] Indicaciones de uso Prod.-Id. 108455: AH DE PHOENIX CONTACT ACTIVATION WIZARD - CHANGE NOTES Phoenix Contact Activation Wizard - Änderungshinweise im Überblick ah_de_phoenix_contact_activation_wizard_change__108455_de_03.pdf	alemán	03
☑	[pdf, 277 KB] Indicaciones de uso Prod.-Id. 108337: AH DE PLCNEXT ENGINEER - CHANGE NOTES PLCnext Engineer - Änderungshinweise im Überblick ah_de_plcnext_engineer_change_notes_108337_de_14.pdf	alemán	14
☑	[pdf, 266 KB] Indicaciones de uso Prod.-Id. 108337: AH EN PLCNEXT ENGINEER - CHANGE NOTES PLCnext Engineer - Change notes at a glance ah_en_plcnext_engineer_change_notes_108337_en_14.pdf	inglés	14
☑	[pdf, 130 KB] Indicaciones de uso Prod.-Id. 108455: AH EN PHOENIX CONTACT ACTIVATION WIZARD - CHANGE NOTES Phoenix Contact Activation Wizard - Change notes at a glance ah_en_phoenix_contact_activation_wizard_change__108455_en_03.pdf	inglés	03
☑	[pdf, 1 MB] Indicaciones de uso Prod.-Id. 107913: AH EN INDUSTRIAL SECURITY Measures to protect network-capable devices with communication interfaces, solutions, and PC-based software against unauthorized access ah_en_industrial_security_107913_en_03.pdf	inglés	03

Quick Start Guide

- **PLCnext Engineer**

Installing and operating the PLCnext Engineer software

Quick start guide

UM QS EN PLCNEXT ENGINEER

Table of contents

PLCnext Engineer

Quick Start Guide

PHOENIX CONTACT
INSPIRING INNOVATIONS

Quick Start Guide

PLCnext Engineer

- Licenses
  - Button Configure
  - Activation Wizard

1.5 PLCnext Engineer licenses

The basic functions of PLCnext Engineer are available as a free of charge license. Once installed, these functions are available without limitation and free of charge. Further functions can be added for a fee (even at a later stage). The licenses are bound to the hardware of a PC or a USB dongle.
To order further licenses, proceed as follows:

- Log in with your access data at phoenixcontact.net/products or register for the first time.
- Select the PLCNEXT ENGINEER product (Order No. 1046008).
- Select "Configure" on the PLCNEXT ENGINEER product page to configure your personal license.

Once you have sent your order, within 48 hours you will receive an email from Phoenix Contact that contains a ticket ID. You need the ticket ID to activate the license.

The Phoenix Contact Activation Wizard is used for the activation process of licenses for further functions. The Phoenix Contact Activation Wizard is a part of the PLCnext Engineer installation package. In order to start the application you will find an .EXE-file under the installation path (Default path: „C:\Program Files (x86)\PHOENIX CONTACT\Phoenix Contact Activation Wizard\").

The USB dongle ESL STICK USB A (Order No. 1080084) for saving licenses for various software products is delivered without licenses. The Phoenix Contact Activation Wizard is also used for the activation process of USB dongle licenses.

- To activate a license, follow the instructions in the Phoenix Contact Activation Wizard.

Quick Start Guide

Quick UM

- PLCnext Engineer

"COMPONENTS" area

The "COMPONENTS" area contains all the components available for the project. The components can be divided into the following categories based on their function:

- Developing program code ("Data Types", "Programs" and "Functions & Function Blocks")
- Showing all devices available for the "PLANT" area and adding them via GSDML or FDCML ("Devices")
- Editing HMI pages ("HMI")
- Adding libraries such as firmware libraries, IEC user libraries or libraries provided by Phoenix Contact ("References")

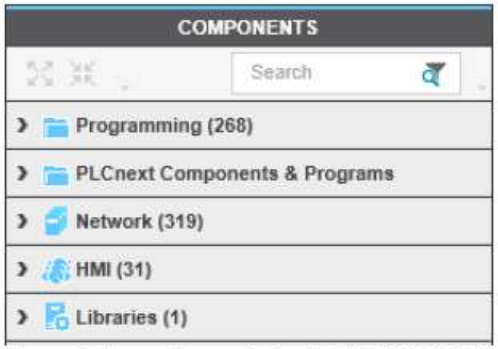


Figure 3-4 Example for the "COMPONENTS" area

Selección Software actual PLCnext Engineer 2020.6.2

PLCnext Engineer

- Ir a página www.phoenixcontact.com/global

- 1046008 PLCNEXT ENGINEER

Software

	Descripción	Idioma	Versión
☐	[exe, 470 MB] **Software** PLCnext Engineer 2020.6.2: PLCnext Engineer es la plataforma de software modular para los sistemas de control de la familia PLCnext Control. Incluye las disciplinas técnicas necesarias para la configuración, el desarrollo y la puesta en servicio de una aplicación de automatización. SHA256 Checksum: 5c381602b353cdd19c4761dc5b58082ea489635ef24d130a3b5470a572cd6bea PLCnext_Engineer_Setup_2020.6.2_64bit.exe	Internacional	2020.6.2
☐	[exe, 467 MB] **Software** PLCnext Engineer 2020.0 LTS Hotfix 1: PLCnext Engineer es la plataforma de software modular para los sistemas de control de la línea PLCnext Control. Incluye las disciplinas técnicas necesarias para la configuración, el desarrollo y la puesta en servicio de una aplicación de automatización. SHA256 Checksum: 0f656daa0a19023070db58cb6f9e13b75cbebe3b5f4c529f269fcc31c2696ec8 PLCnext_Engineer_Setup_2020.0.1_LTS_(64bit).exe	Internacional	2020.0.1 LTS
☐	[zip, 47 MB] **Software** El asistente de activación de Phoenix Contact sirve para activar licencias de software para las que previamente se solicitó un ID de ticket. SHA256 Checksum: 970da4622347af2b2bee4a6184364847a7ed2c396600c77951a643b7c9b64e6f Activation Wizard Setup 1.3.2.zip	Internacional	1.3.2

Selección del software

Selección Software DEMO

PLCnext Engineer

- Ir a página www.phoenixcontact.com/global

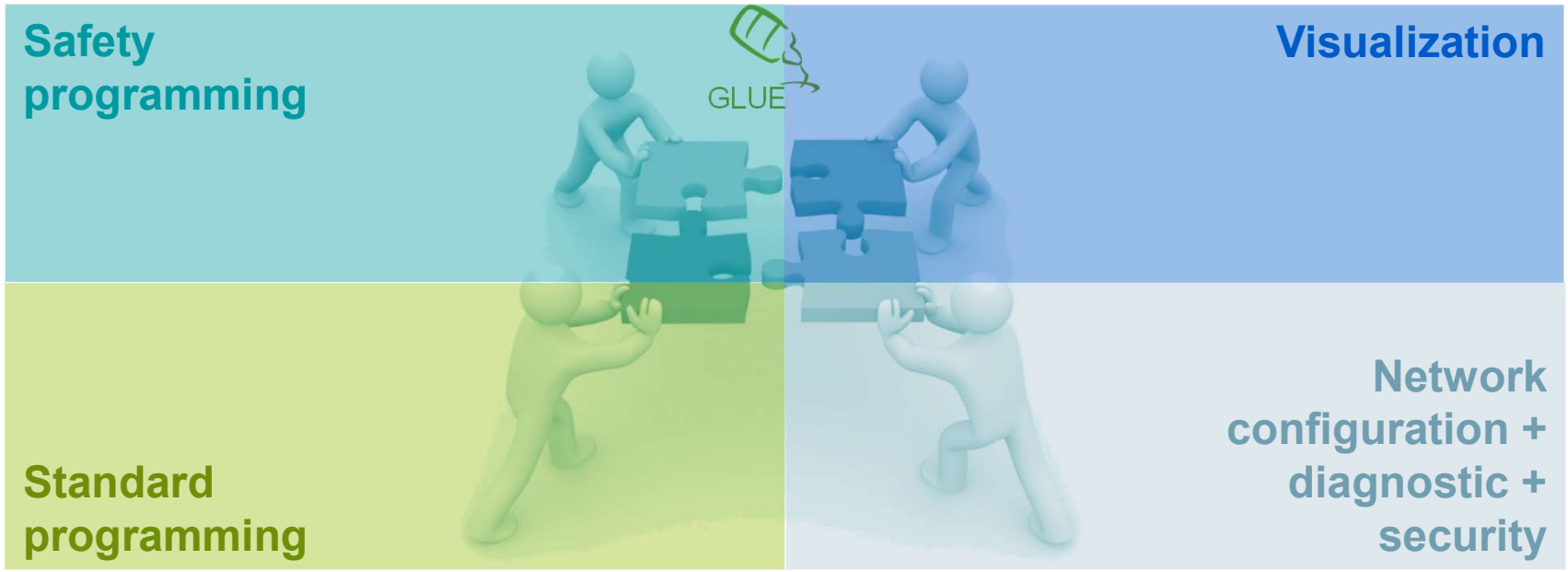
- 1046008 PLCNEXT ENGINEER

Demo-Software (revisions)

	Descripción	Idioma	Versión
☐	[exe, 470 MB] Demo-Software (revisions) PLCnext Engineer 2020.6: PLCnext Engineer es la plataforma de software modular para los sistemas de control de la familia PLCnext Control. Incluye las disciplinas técnicas necesarias para la configuración, el desarrollo y la puesta en servicio de una aplicación de automatización. PLCnext-Engineer-Setup-2020.6-64bit.exe	Internacional	2020.6
☐	[exe, 476 MB] Demo-Software (revisions) PLCnext Engineer 2020.3 Hotfix 1: PLCnext Engineer es la plataforma de software modular para los sistemas de control de la familia PLCnext Control. Incluye las disciplinas técnicas necesarias para la configuración, el desarrollo y la puesta en servicio de una aplicación de automatización. PLCnext-Engineer-Setup-2020.3.1-64bit-.exe	Internacional	2020.3.1
☐	[zip, 476 MB] Demo-Software (revisions) PLCnext Engineer 2020.3: PLCnext Engineer es la plataforma de software modular para los sistemas de control de la familia PLCnext Control. Incluye las disciplinas técnicas necesarias para la configuración, el desarrollo y la puesta en servicio de una aplicación de automatización. PLCnext Engineer Setup 2020.3 (64bit).zip	Internacional	2020.3
☐	[zip, 467 MB] Demo-Software (revisions) PLCnext Engineer 2020.0 LTS: PLCnext Engineer es la plataforma de software modular para los sistemas de control de la línea PLCnext Control. Incluye las disciplinas técnicas necesarias para la configuración, el desarrollo y la puesta en servicio de una aplicación de automatización. PLCnext_Engineer_Setup_64bit_2020.0_LTS.zip	Internacional	2020.0 LTS
☐	[zip, 433 MB] Demo-Software (revisions) PLCnext Engineer 2019.9: PLCnext Engineer es la plataforma de software modular para los sistemas de control de la línea PLCnext Control. Incluye las disciplinas técnicas necesarias para la configuración, el desarrollo y la puesta en servicio de una aplicación de automatización. PLCnext_Engineer_Setup_(64bit)_2019.9.zip	Internacional	2019.9

PLCnext Technology
Designed by PHOENIX CONTACT

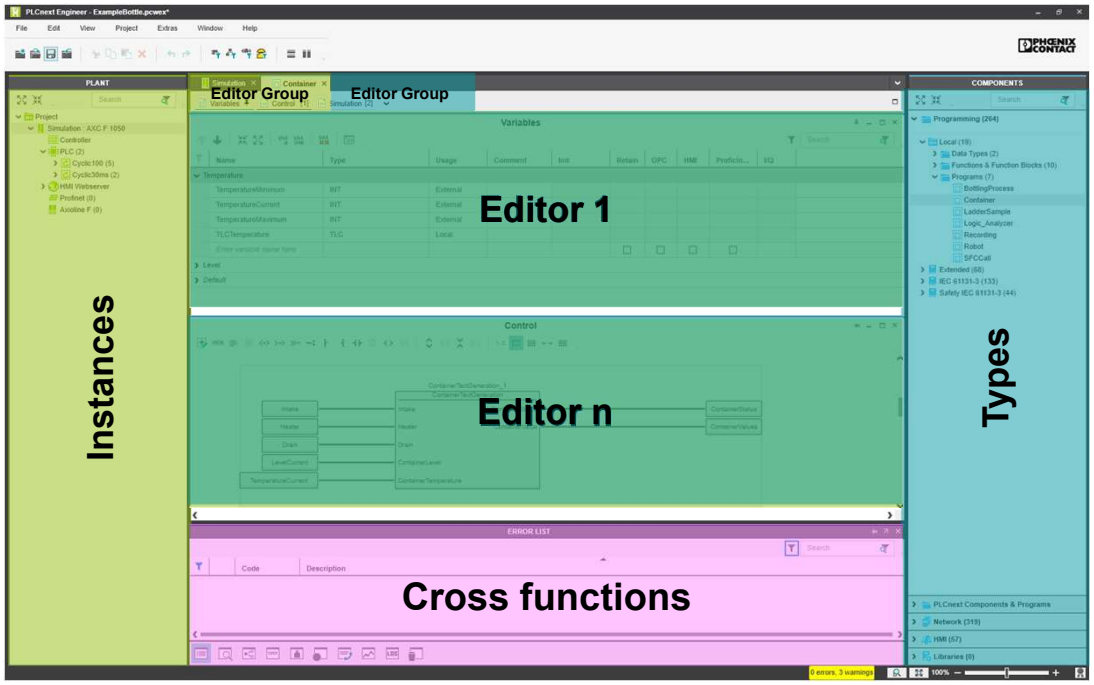
Complete Integrated System



PHOENIX CONTACT
INSPIRING INNOVATIONS

PLCnext Technology
Designed by PHOENIX CONTACT

Information Architecture



PHOENIX CONTACT
INSPIRING INNOVATIONS

PLCnext Engineer

Integrated Visualization Editor

- **Deeply integrated**
  - Based on central handling
- **Scalable**
  - From small scale controllers to IPCs
- **No client installation**
  - Modern web browser
- **Technology-neutral**
  - Screens are stored in neutral format
- **Lightweight**
  - Low resource demands on PLC

PLCnext Engineer

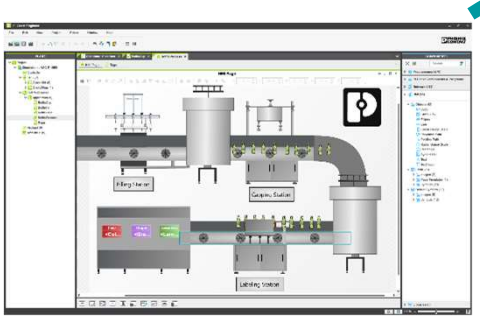
Visualization Runtime Concept

PLCnext Technology
Designed by PHOENIX CONTACT

One project
(PLC + HMI)

HMI application
HMI project

variable
values



PLCnext Engineer
Engineering Software

Client(s)

PHOENIX CONTACT
INSPIRING INNOVATIONS

PLCNEXT ENGINEER

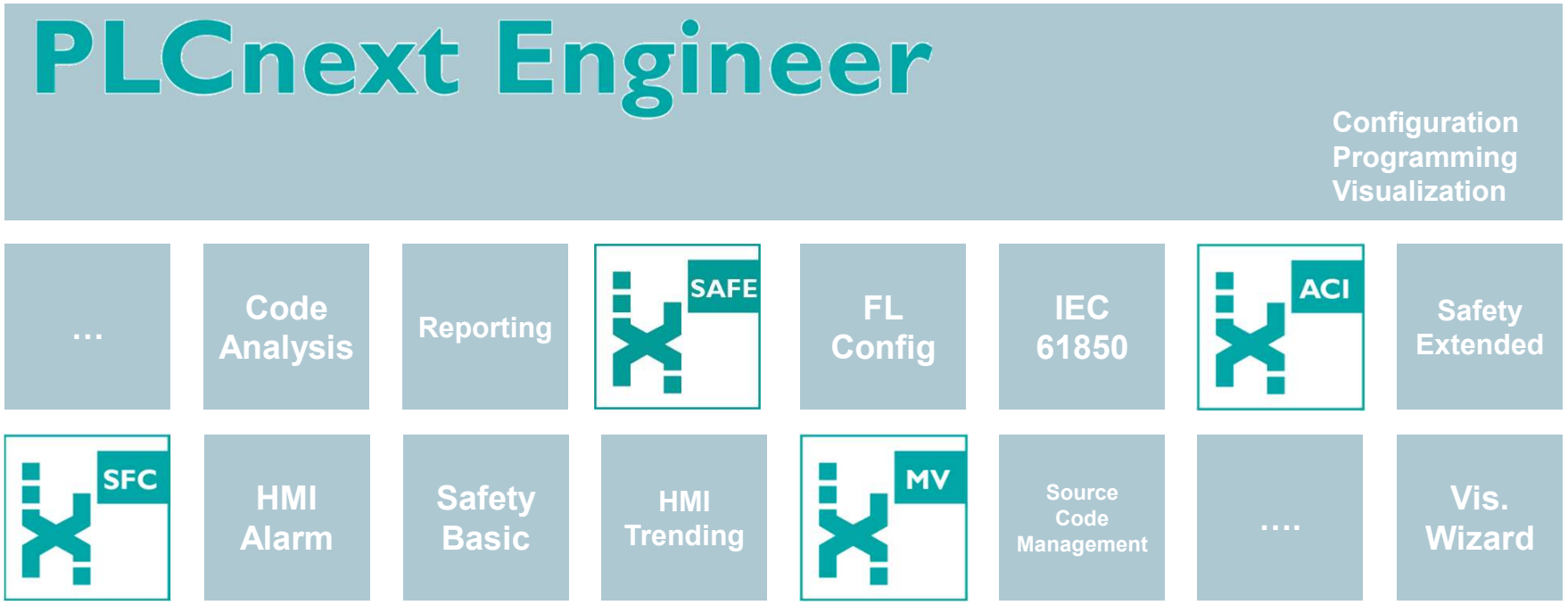
Página Internet Internacional

- **Configurar**

PHOENIX CONTACT
INSPIRING INNOVATIONS

PLCnext Technology
Designed by PHOENIX CONTACT

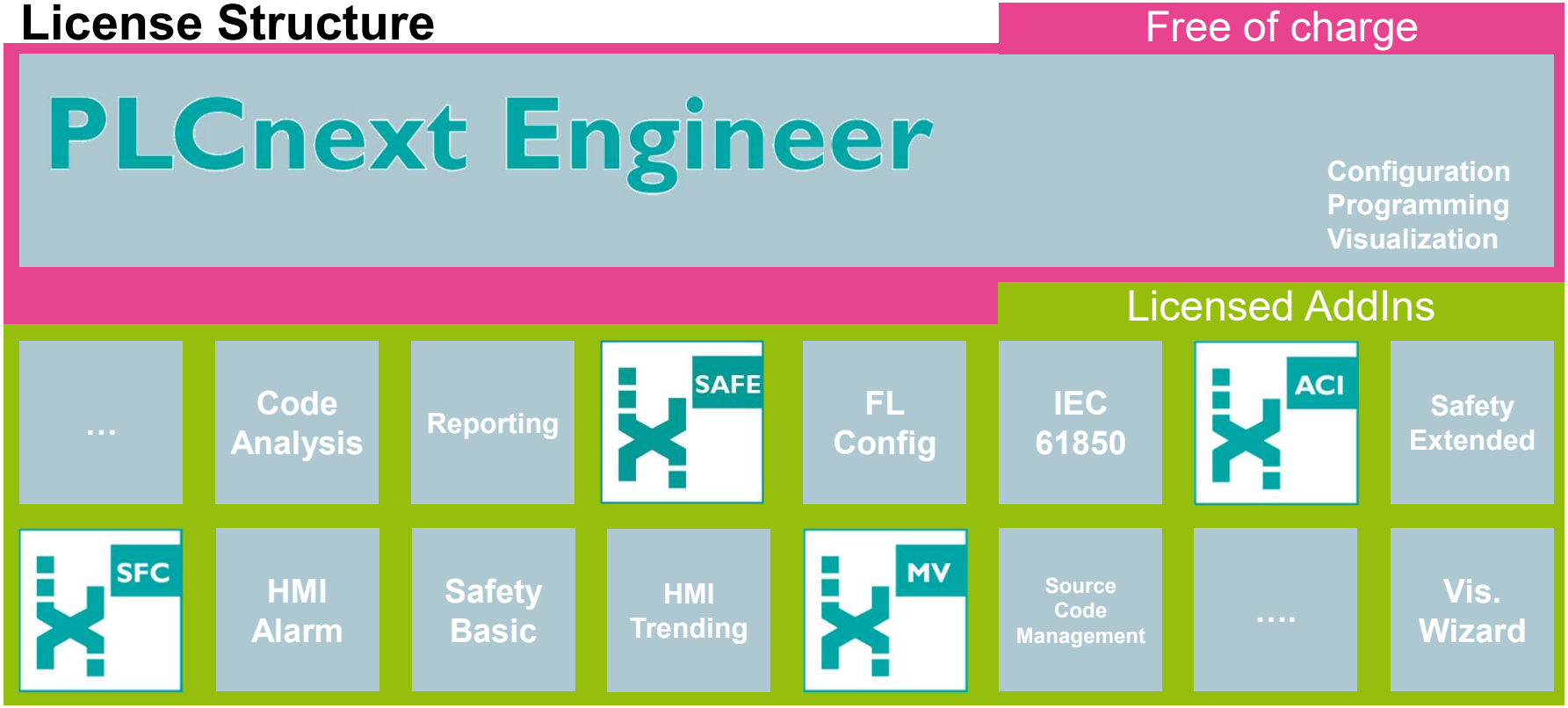
License Structure



PHOENIX CONTACT
INSPIRING INNOVATIONS

PLCnext Technology
Designed by PHOENIX CONTACT

License Structure



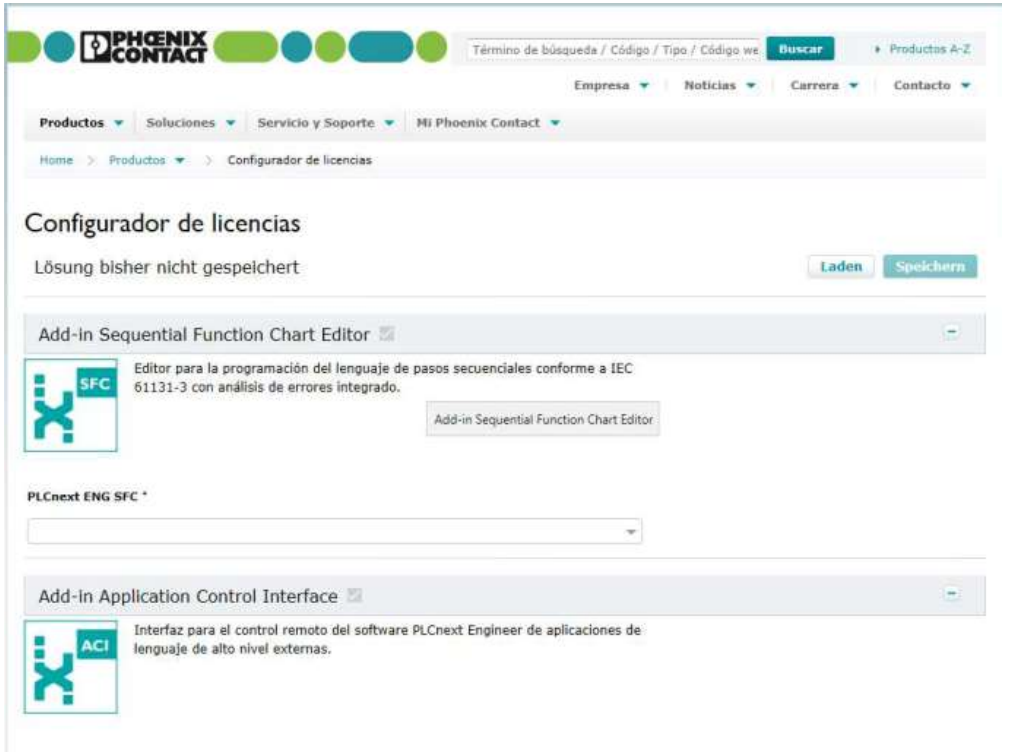
Icon = available Add-In

No icon = Idea about future Add-Ins

PHOENIX CONTACT
INSPIRING INNOVATIONS

Opciones

PLCnext Engineer 1046008

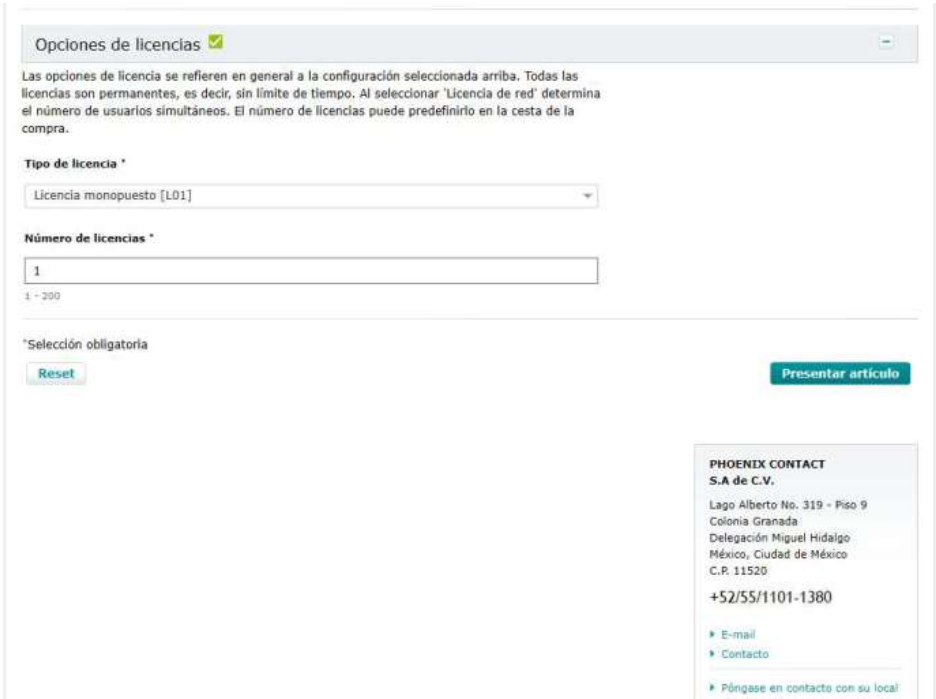


Opciones

PLCnext Engineer 1046008

Opciones

PLCnext Engineer 1046008

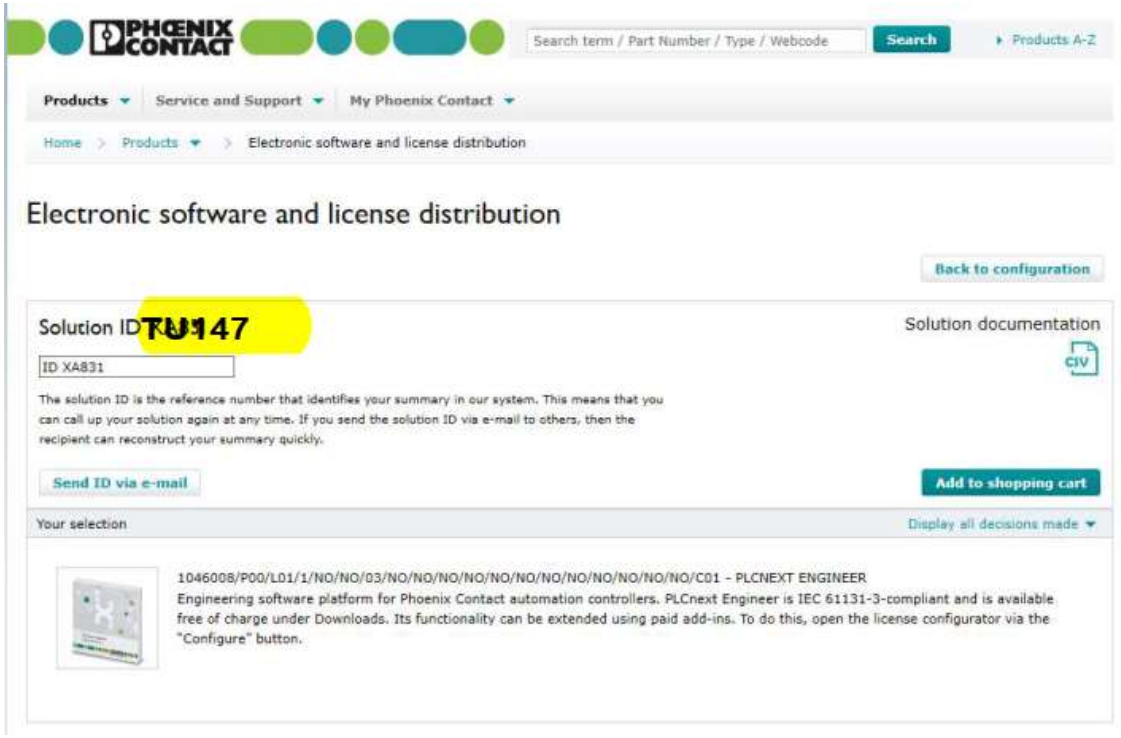


Opciones

PLCnext Engineer 1046008

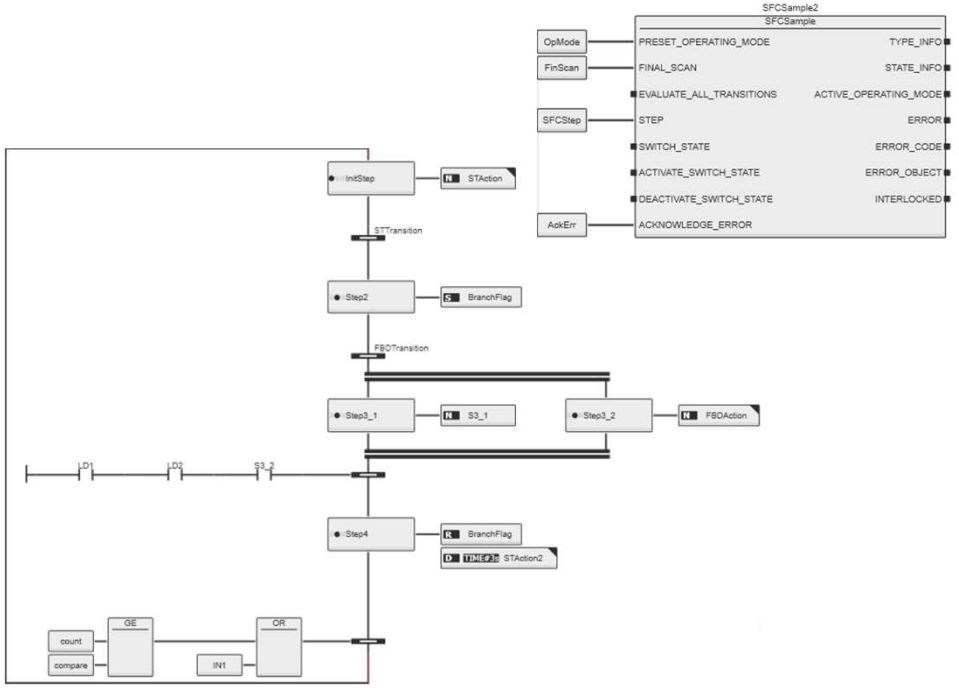
Opciones

PLCnext Engineer 1046008



Sequential Function Chart – SFC

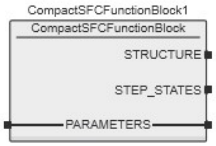
- Represented as a function block
- Automatic generated TypeInfo and StateInfo structure
- Error handling
- Directly connected transitions
- Transitions in separate worksheets (FBD, ST, LD)
- Operation modes:
  Automatic, Manual Step, Halted

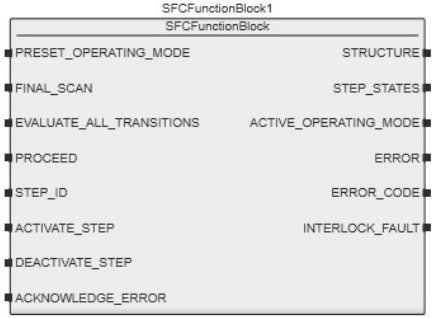


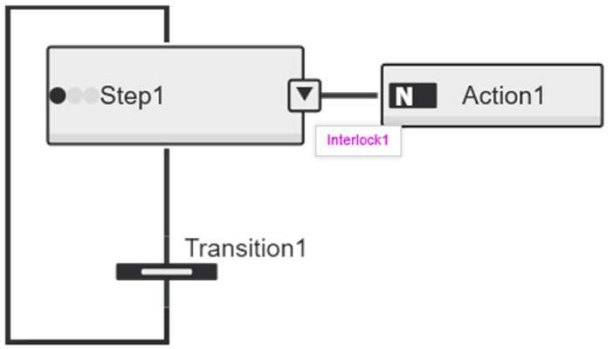
PLCnext Engineer

Sequential Function Chart – SFC

- Compact SFC
- STEP Interlock
  can be used to control the execution of actions associated to a step
- Pre-Execute worksheet
- Post-Execute worksheet





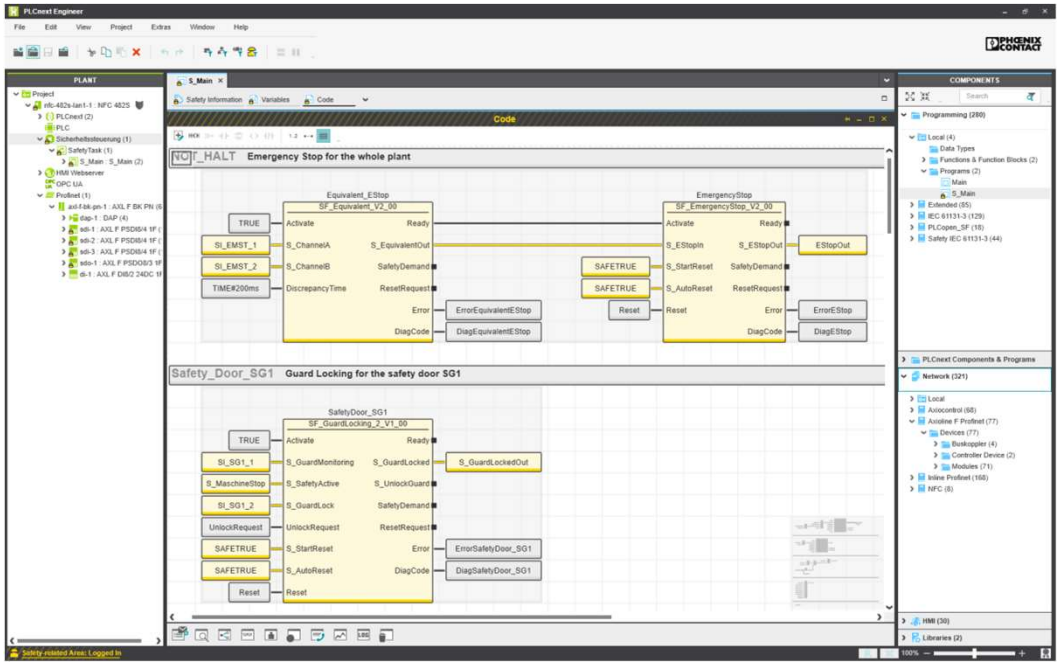


PLCnext Engineer

Functional Safety Programming

Fully integrated Safety Programming

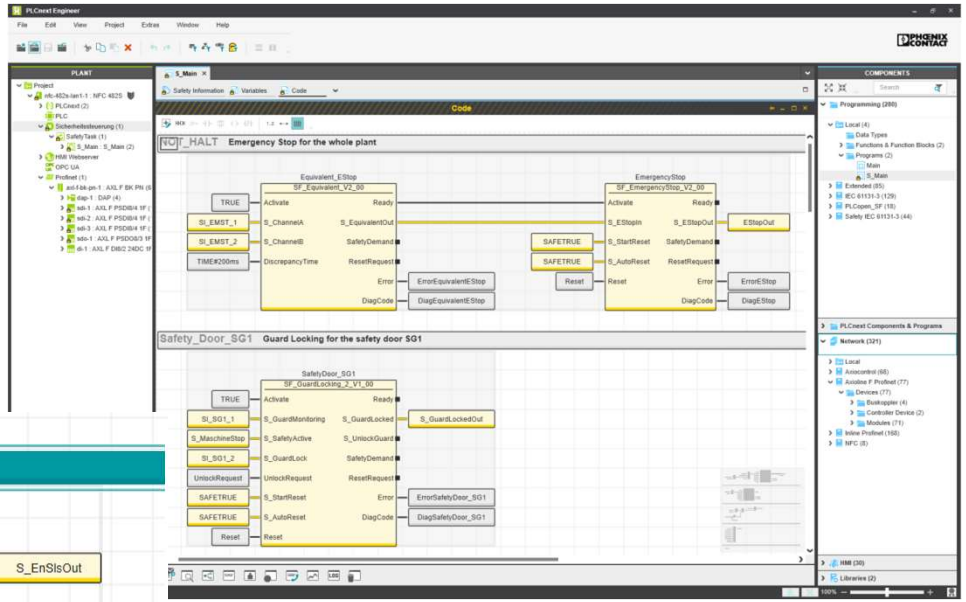
- TÜV Rheinland certified according to IEC 61508
- Editor with common behavior as known from standard FBD or LD editor
- Low Variability Language support
- Network granular CRC checksums
- PROFIsafe Support

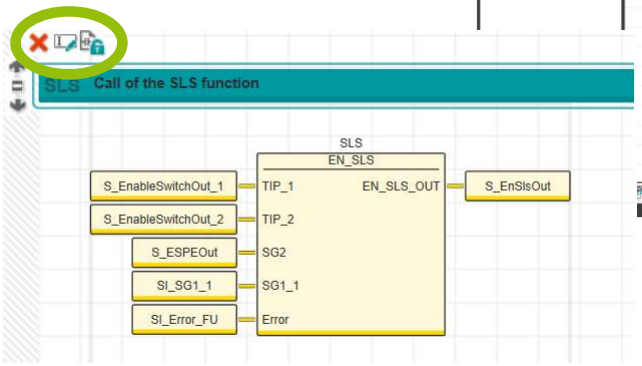


Functional Safety Programming

Fully integrated Safety Programming

- Individual safety functions can be protected by a verification function
- Background signal path analysis
- Background safe semantic analysis
- Diversely-redundant code generator

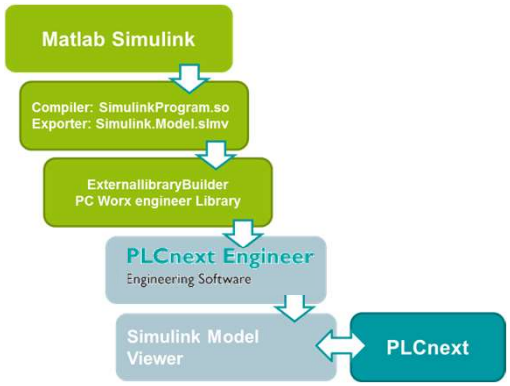


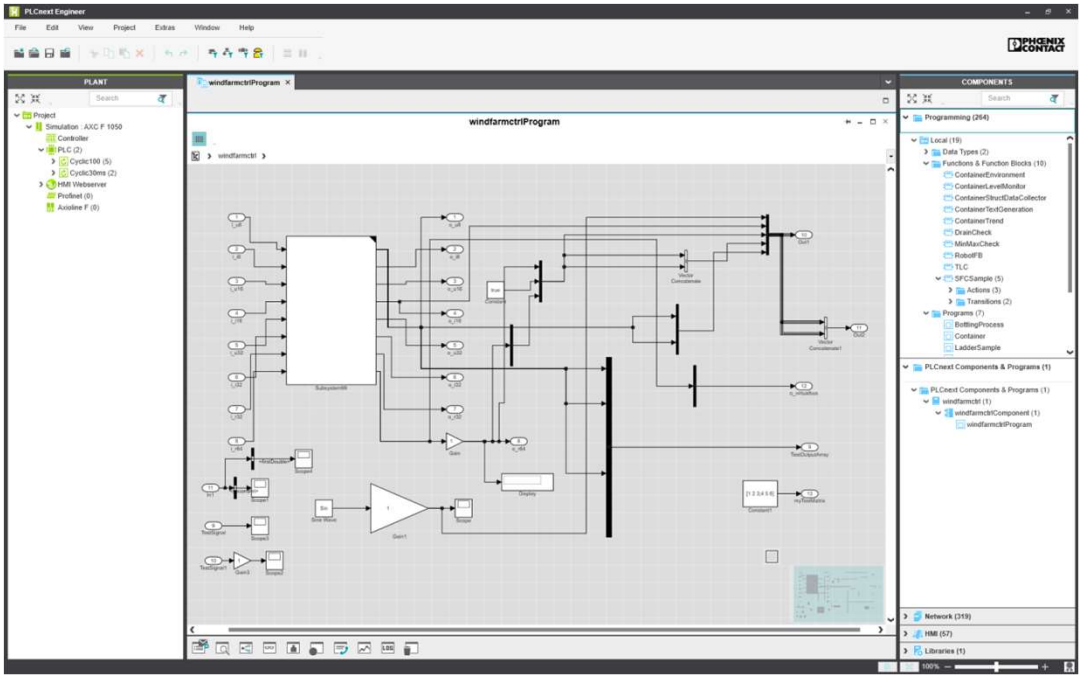


PLCnext Engineer

Viewer for Simulink

- Model export as part of a PLCnext library
- Drill-down into sub-models
- Online-values for In- and Out-Ports

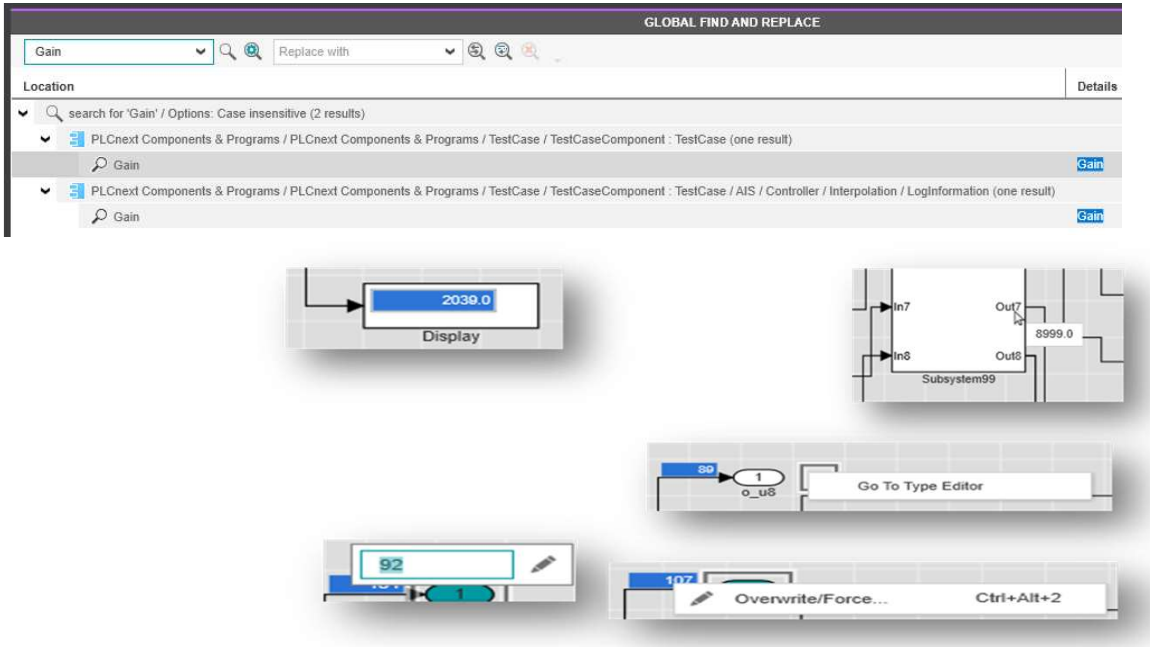




PLCnext Engineer 2019.0

Viewer for Simulink

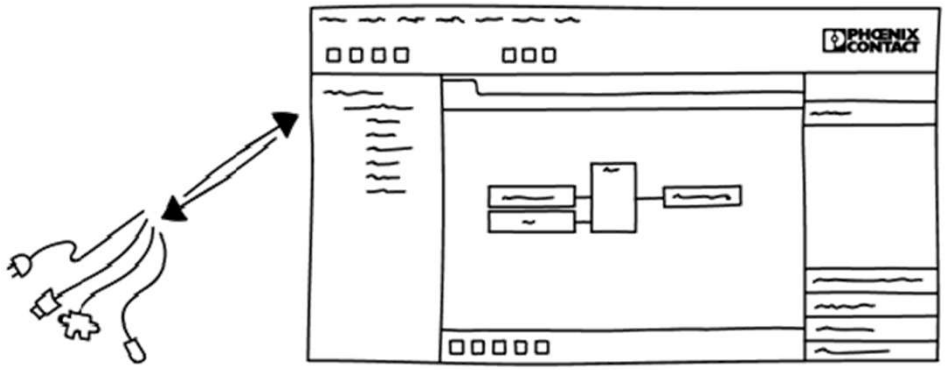
- Global / Local Search
  - Jumpable objects selected
- Display block
  with online values
- Overwrite of GDS ports
- Jump to Type Model from Instance
- Online Indication on lines
  for boolean in /out ports



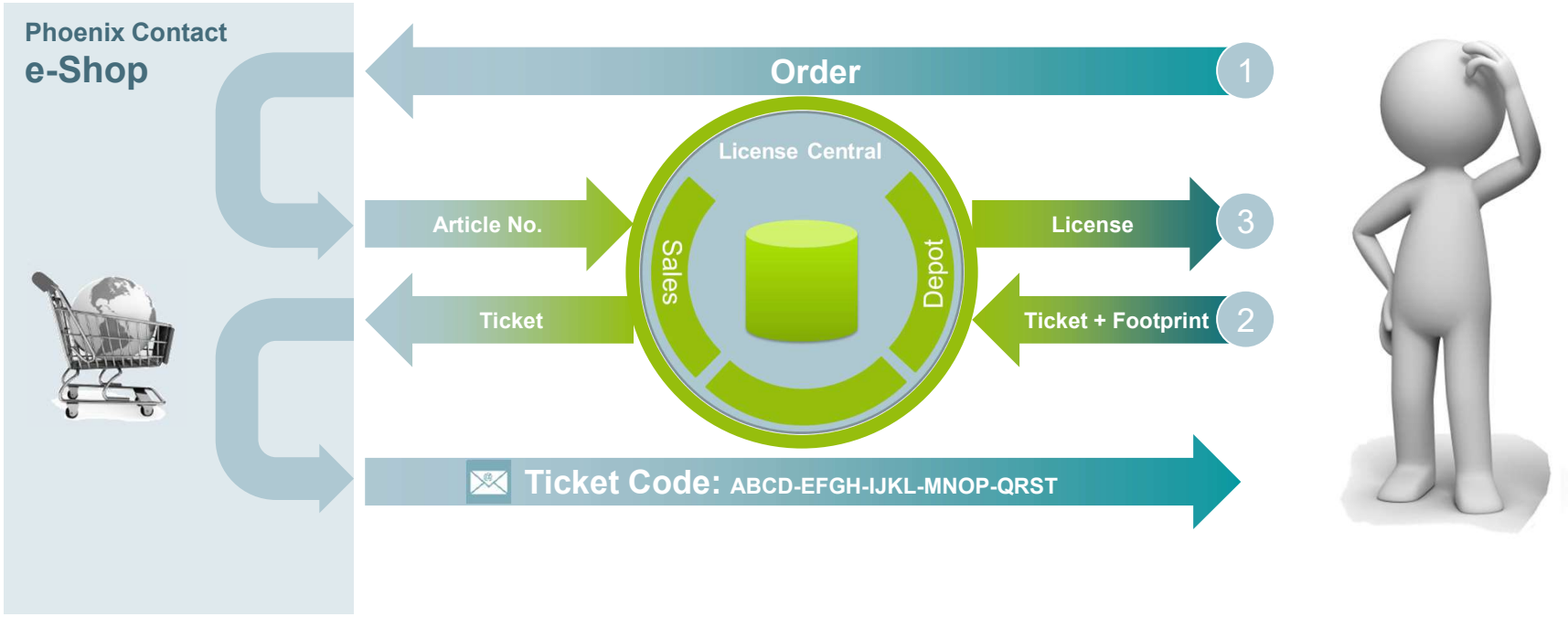
Application Control Interface (ACI)

Remote Control of the software:

- Application.BuildPath (property)
- Application.OpenProject (method)
- Application.ProjectOpened (event)
- Project.Close (method)
- Project.Save (method)
- Project.SaveAs (method)
- Project.Closed (event)
- ……



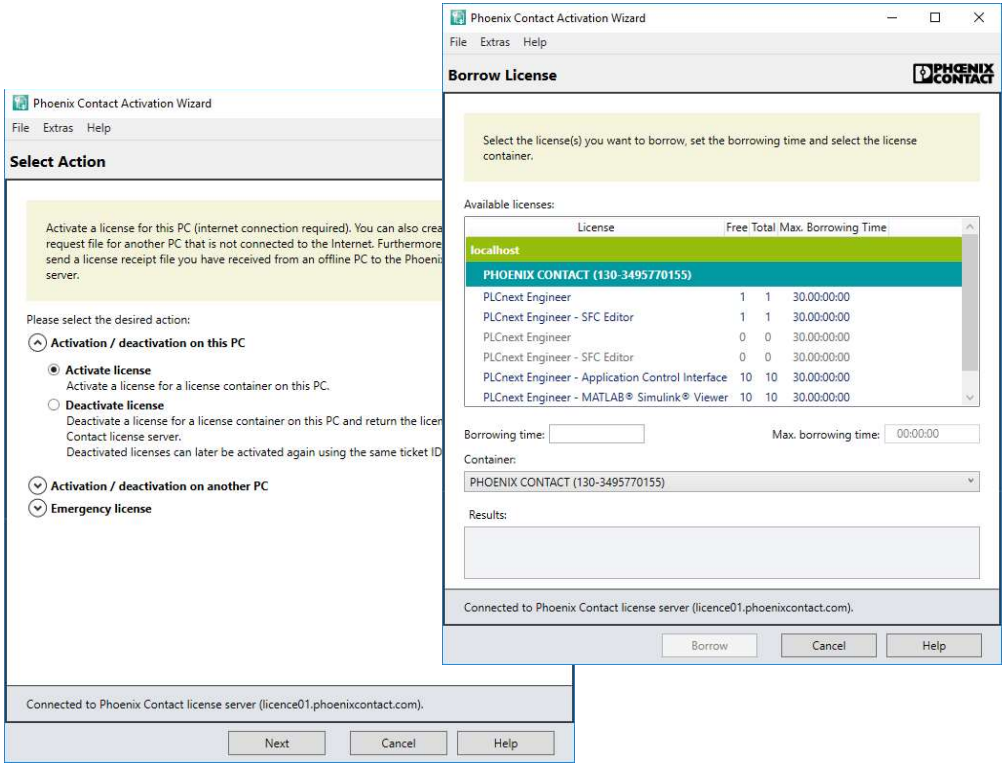
Software License Distribution



Licensing

Activation Wizard

- Version 1.1 HMI 2018
  - Deactivating / Moving licenses
- Version 1.2 SPS 2018
  - Network server for licenses
  - Server list; authentication
  - Borrowing of licenses
    (can be returned to pool)



PLCnext Engineer

Electronic Software License on USB A

PLCnext Technology
Designed by PHOENIX CONTACT

Software dongle - ESL STICK USB A - 1080084

CmDongle for saving licenses for various software products

Generate product PDF

current article

Add to product comparison | Add to part list | Add to shopping cart

Overview | Technical data | Downloads

Product Description

Up to 2000 licenses with different license models for various software tools can be stored on the license dongle.
Licenses can be used flexibly by moving the dongle from one computer to another.
Use of a license dongle is recommended when using virtual machines. This means that licenses can still be used after virtual machines are copied or even if settings are changed on the virtual machine.
Using the "Activation Wizard" software tool, activate and deactivate licenses on the license dongle. Or use the "Activation Wizard" to migrate licenses from a PC hard drive to a license dongle (or vice versa), for example.
No additional drivers are required to operate the dongle. After it is connected to a computer, it can immediately be used without administrator rights.

RoHS

IF Design Award 2019

PHOENIX CONTACT
INSPIRING INNOVATIONS

PLCnext Engineer

Versioning

PLCnext Technology
Designed by PHOENIX CONTACT

PLCnext Engineer
2020.0 LTS

January
2020

PLCnext Engineer
2020.3

March
2020

PLCnext Engineer
2020.6

June
2020

PLCnext Engineer
2020.9

September
2020

PHOENIX CONTACT
INSPIRING INNOVATIONS

LTS Version

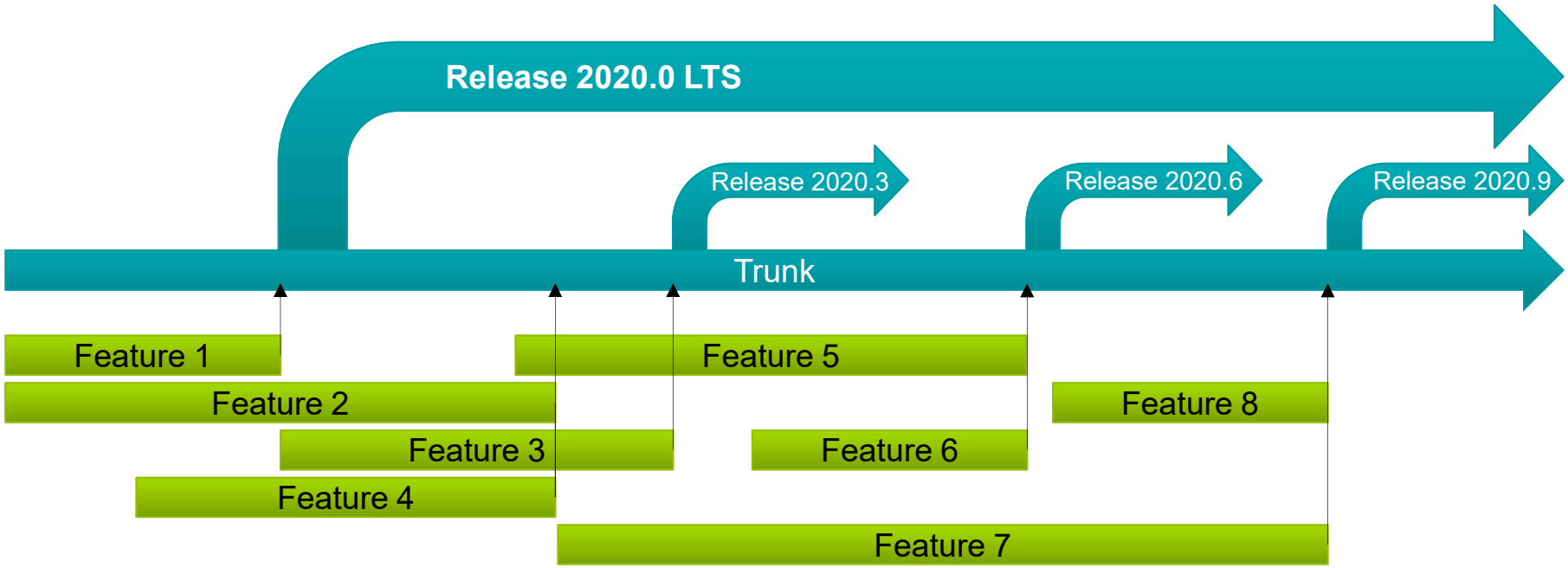
Wikipedia:

Long-term support (**LTS**) …
… is a product lifecycle management policy in which a stable release of computer software is maintained for a longer period of time than the standard edition. The term is typically reserved for open-source software, where it describes a software edition that is supported for months or years longer than the software's standard edition.

Source 2019/01: https://en.wikipedia.org/wiki/Long-term_support

PLCnext Technology
Designed by PHOENIX CONTACT

Feature-Driven Development



PHOENIX CONTACT
INSPIRING INNOVATIONS

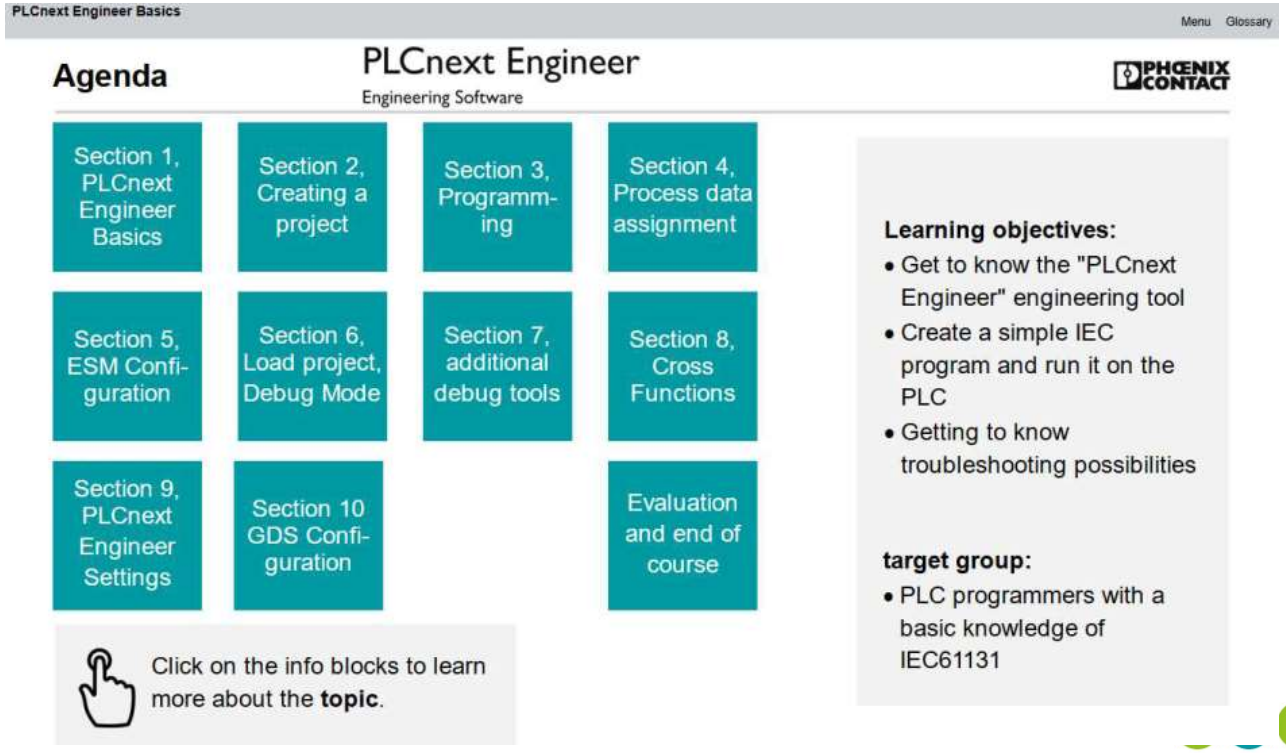
PLCnext Engineer

- **Nivel Básico PLCnext Engineer**

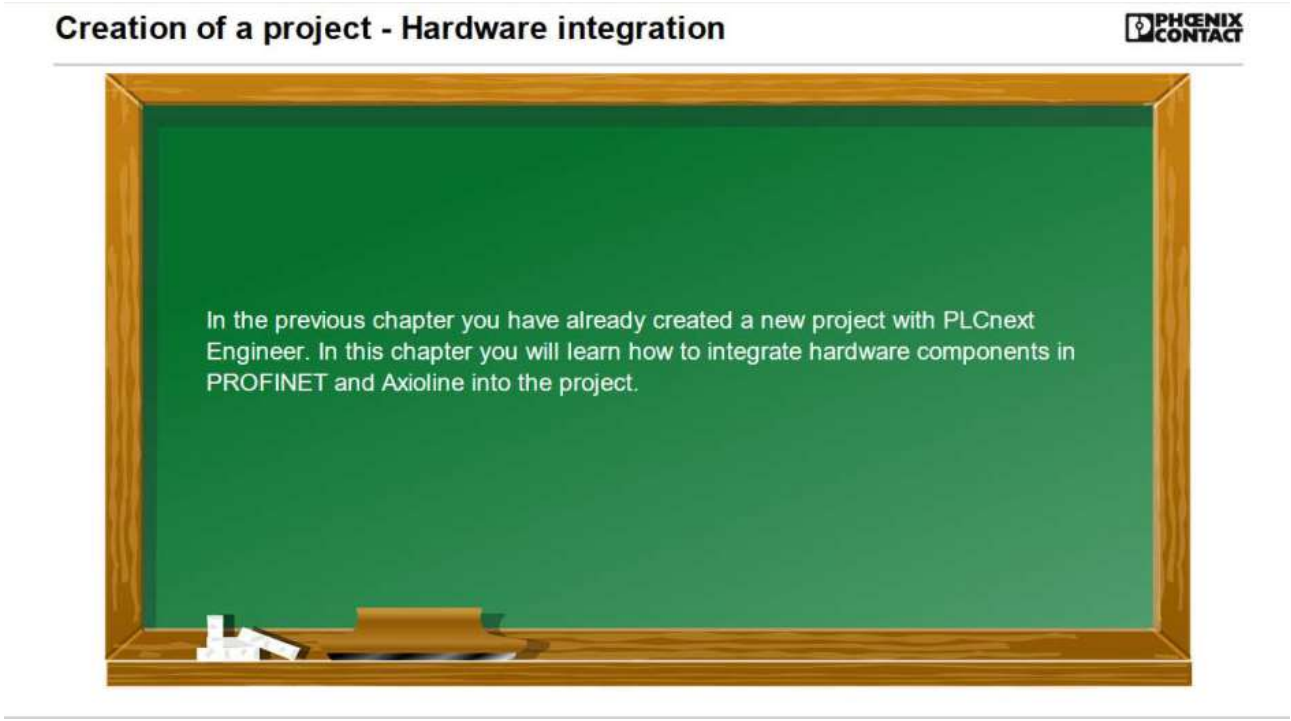
PHOENIX CONTACT
INSPIRING INNOVATIONS

PLCnext Community

E-Learning PLCnext Engineer Basics



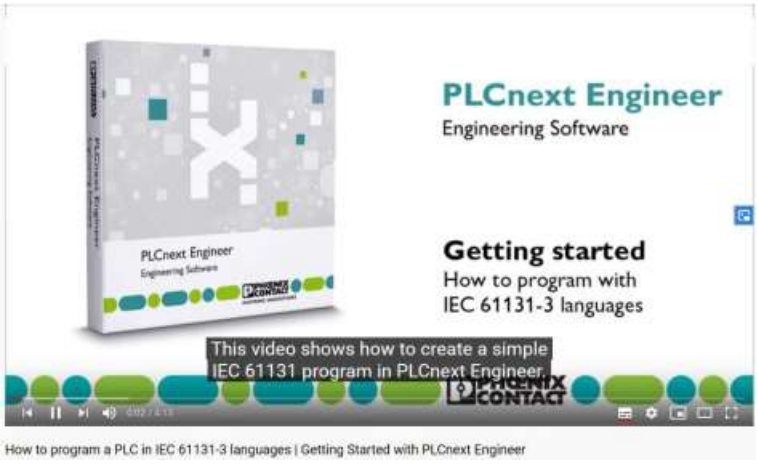
Chapter 2 Creating a Project



Video Youtube

PLCnext Engineer Tutorial(s)

How to set up a new project | Getting Started with PLCnext Engineer



How to program a PLC in IEC 61131-3 languages | Getting Started with PLCnext Engineer

Video Youtube

PLCnext Engineer Tutorial(s)

[1] PLCnext Engineer | Comenzando con PLCNext - Phoenix Contact

Video Youtube

PLCnext Engineer Tutorial(s)

PHOENIX CONTACT
INSPIRING INNOVATIONS

Video Youtube

PLCnext Engineer Tutorial(s)

PLCnext Lesson 4 - Programming the Controller's External Digital IOs

PLCnext Lesson 5 - Programming the Controller's External Analog IOs

PHOENIX CONTACT
INSPIRING INNOVATIONS

Video Youtube

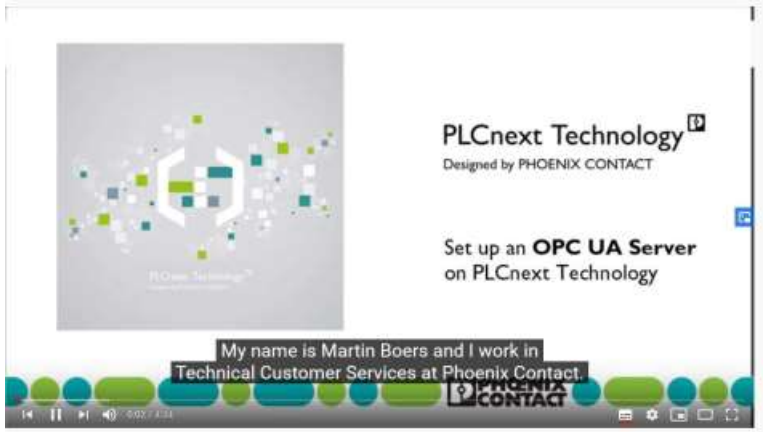
PLCnext Engineer Tutorial(s)

How to create a library out of many device descriptions | Getting started with PLCnext Engineer

How to import device descriptions and devices libraries | Getting started with PLCnext Engineer

Video Youtube

PLCnext Engineer Tutorial(s)



OPC UA with PLCnext Technology | How to set up a server connection with PLCnext Engineer

OPC UA with PLCnext Technology | How to transfer a file to the PLCnext Control via OPC UA Server

Video Youtube

PLCnext Engineer Tutorial(s)

How to debug using Breakpoints | Debugging with PLCnext Engineer

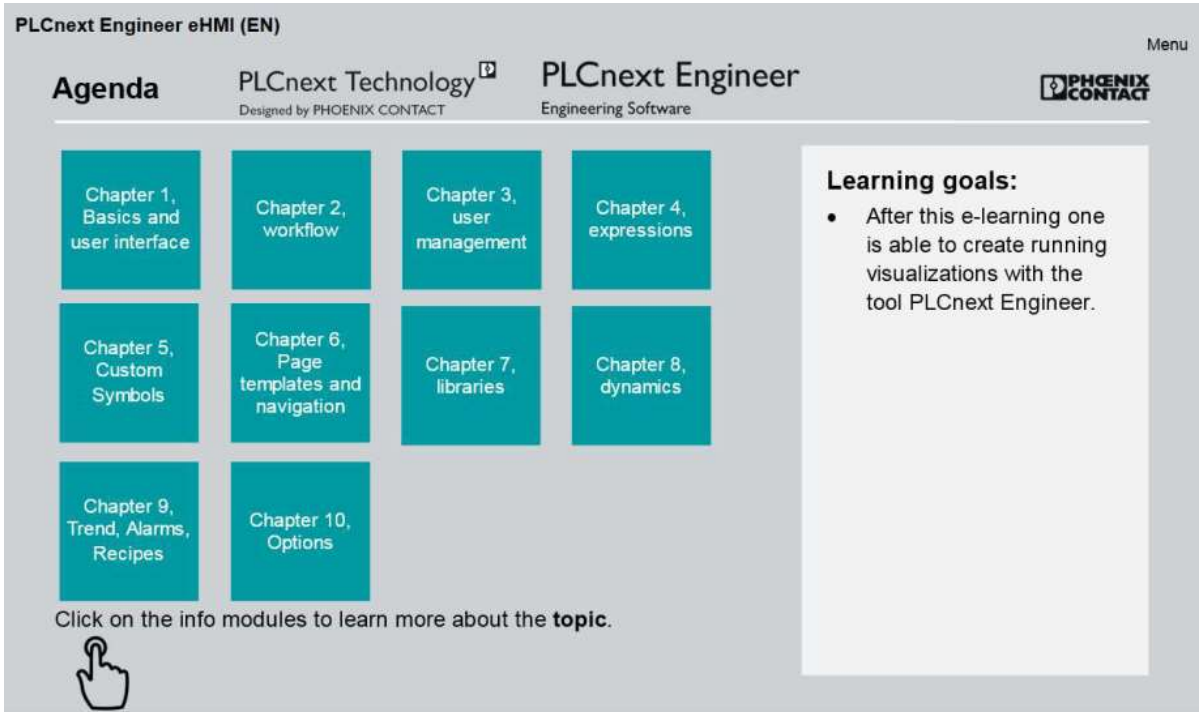
How to use Wireshark and WinSCP in debugging PLC installations | Working with PLCnext Technology

PLCnext Engineer HMI

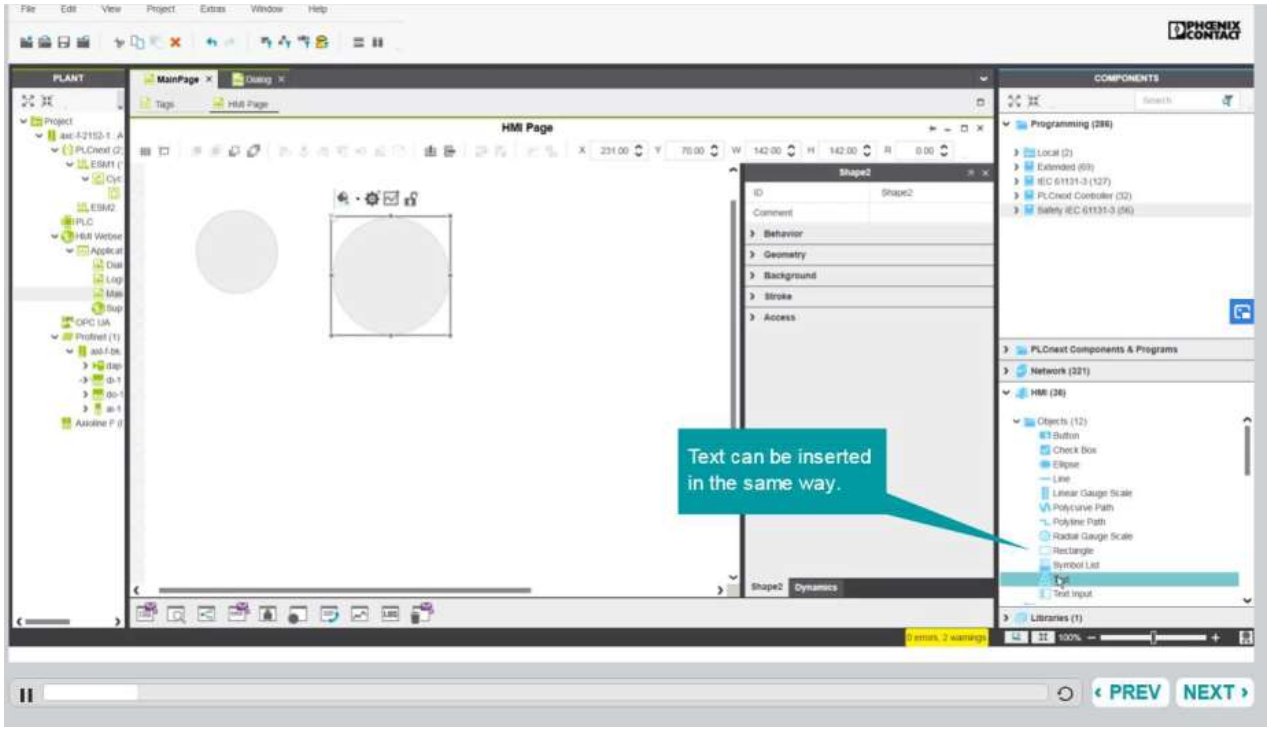
- **Nivel Básico PLCnext Engineer HMI**

PHOENIX CONTACT
INSPIRING INNOVATIONS

E-Learning PLCnext Engineer HMI Basics



Chapter 2



Video Youtube

PLCnext Engineer Tutorial(s)

How to create a basic visualization with PLCnext Engineer

PHOENIX CONTACT
INSPIRING INNOVATIONS

PLCnext Engineer

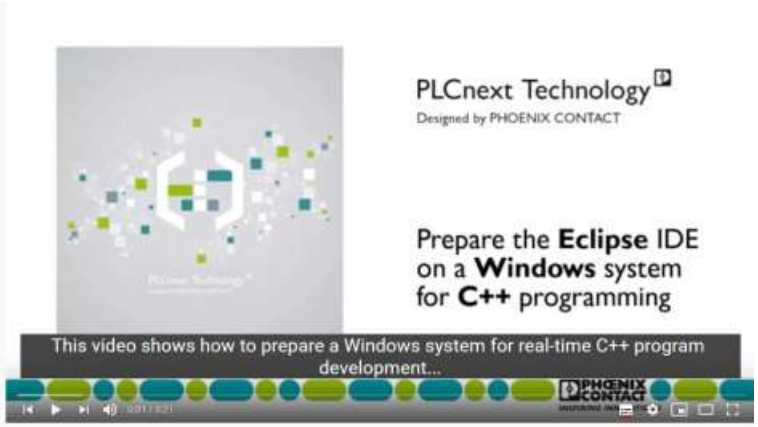
Tutorials Videos

Videos PLCnext Technology Eje Eléctrico SMC gobernado por un Google Home

PHOENIX CONTACT
INSPIRING INNOVATIONS

Video Youtube

PLCnext Engineer Tutorial(s)



How to install Eclipse IDE tools on Windows 10 | C++ with PLCnext Technology

How to program C++ on a PLCnext control | C++ with PLCnext Technology

PHOENIX CONTACT
INSPIRING INNOVATIONS

Video Youtube

PLCnext Engineer Tutorial(s)

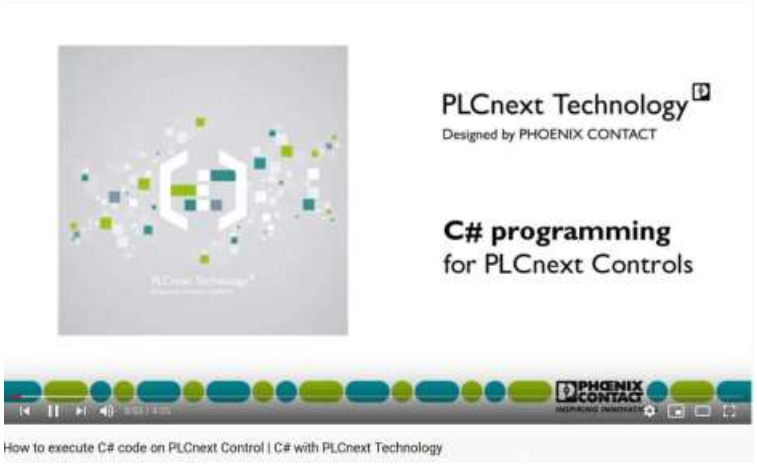
How to install Eclipse IDE tools on Linux | C++ with PLCnext Technology

How to add or remove programs and components in C++ programming | PLCnext Technology

PHOENIX CONTACT
INSPIRING INNOVATIONS

Video Youtube

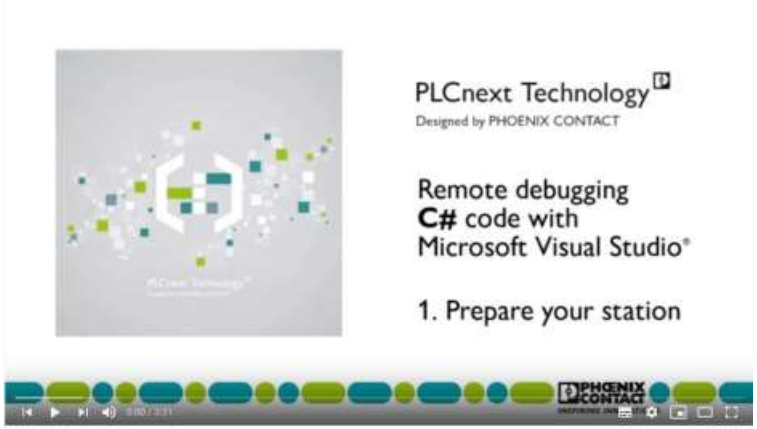
PLCnext Engineer Tutorial(s)



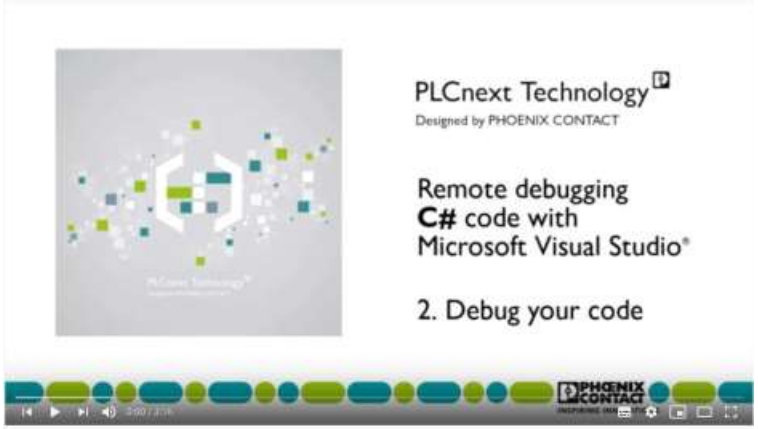
PHOENIX CONTACT
INSPIRING INNOVATIONS

Video Youtube

PLCnext Engineer Tutorial(s)



How to prepare your station for remote debugging C# code on PLCnext Controls | PLCnext Technology

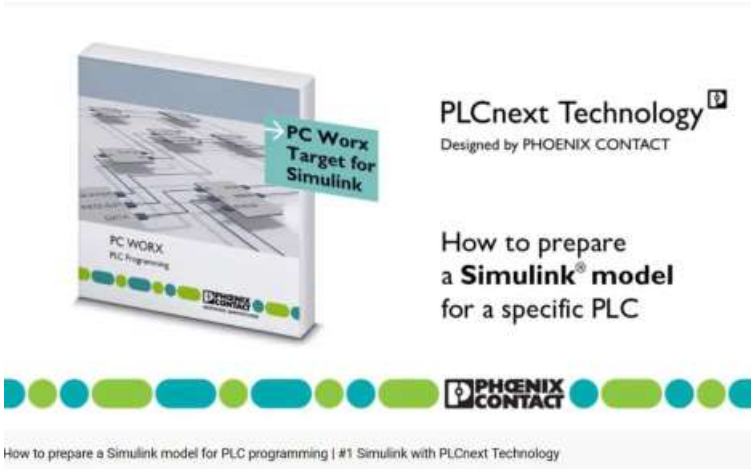


How to debug C# code with Visual Studio development tools | PLCnext Technology

PHOENIX CONTACT
INSPIRING INNOVATIONS

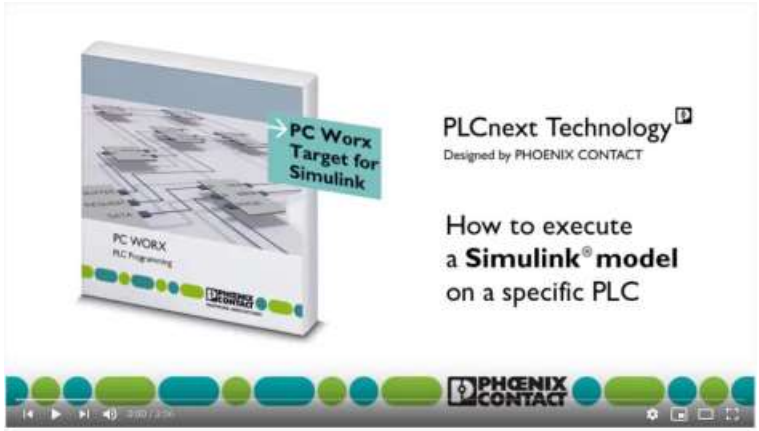
Video Youtube

PLCnext Engineer Tutorial(s)

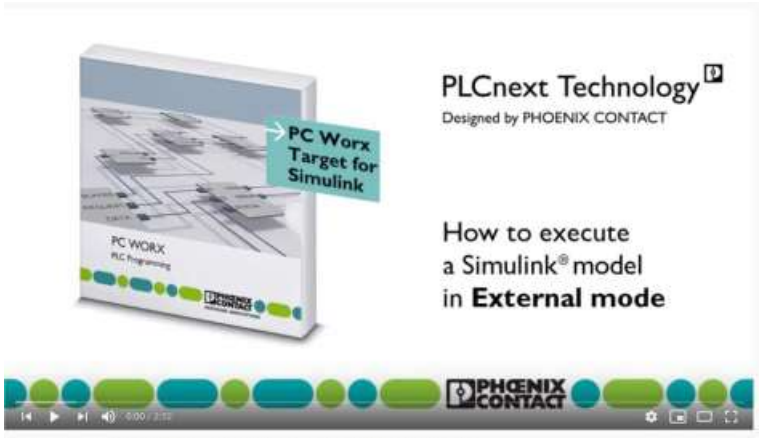


Video Youtube

PLCnext Engineer Tutorial(s)



How to execute a PLC program based on a Simulink model | #2 Simulink with PLCnext Technology



How to set up an External Mode execution | #3 Simulink with PLCnext Technology

INSPIRING INNOVATIONS

Video Youtube

PLCnext Engineer Tutorial(s)

Video Youtube

PLCnext Engineer Tutorial(s)

PLCnext Lesson 8 - Designing the HMI for Analog Outputs

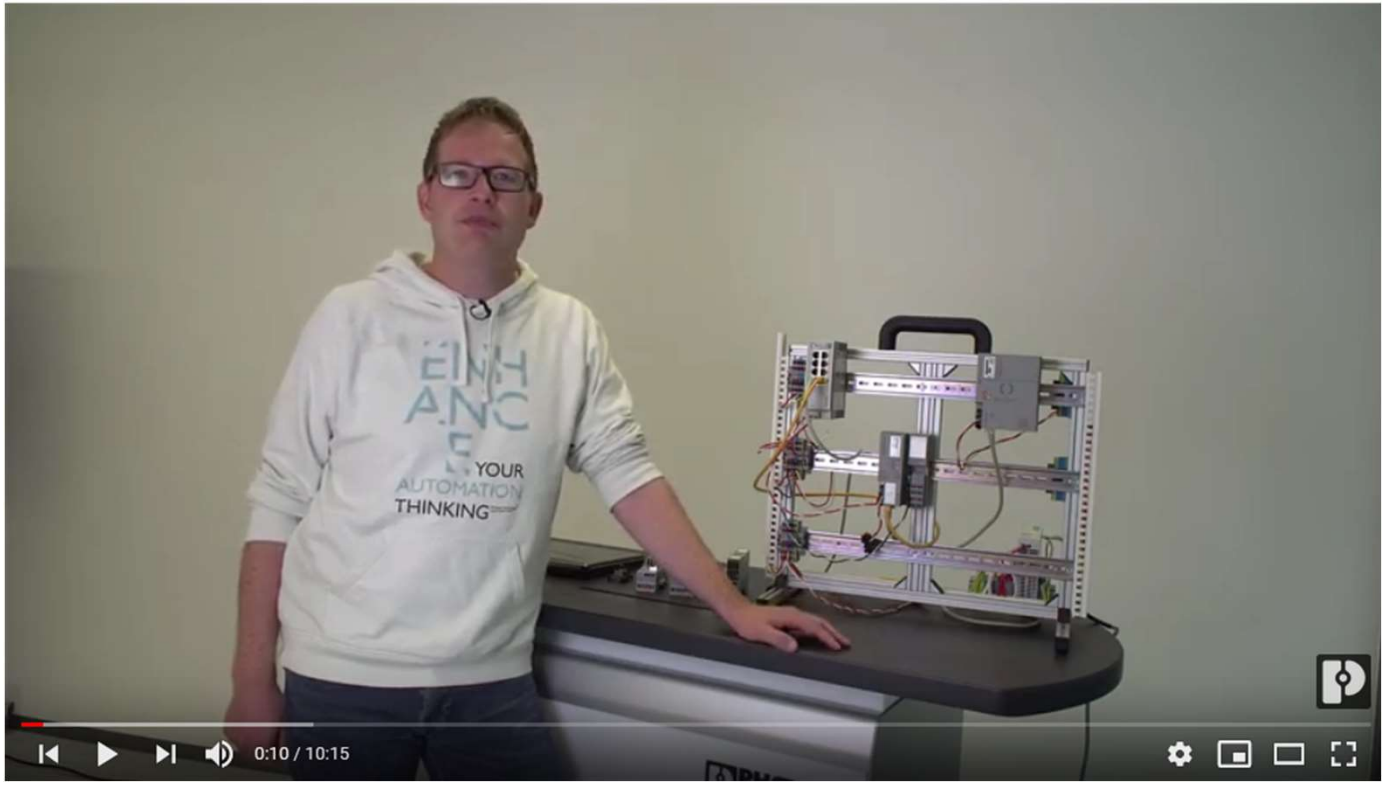
PLCnext Lesson 9 - Monitoring and controlling HMI in browser

PHOENIX CONTACT
INSPIRING INNOVATIONS

Video Youtube

PLCnext Engineer Tutorial(s)

PHOENIX CONTACT
INSPIRING INNOVATIONS



How to set up an additional network card to your PLC with an AXC F XT ETH 1TX | PLCnext Control

PHOENIX CONTACT
INSPIRING INNOVATIONS

High Level Languages

- Phyton
- C++
- C++ Components and Programs
- C#

The new era of automation

PLCnext Technology – Python examples

Python examples

Agenda

- General introduction to Python
- Hello World project
- Installation of packages
  - Manual installation
  - Installation via PIP
- Example 1: Modbus TCP with Python
- Example 2: MQTT with Python
- Q&A

PHOENIX CONTACT
INSPIRING INNOVATIONS

Python examples

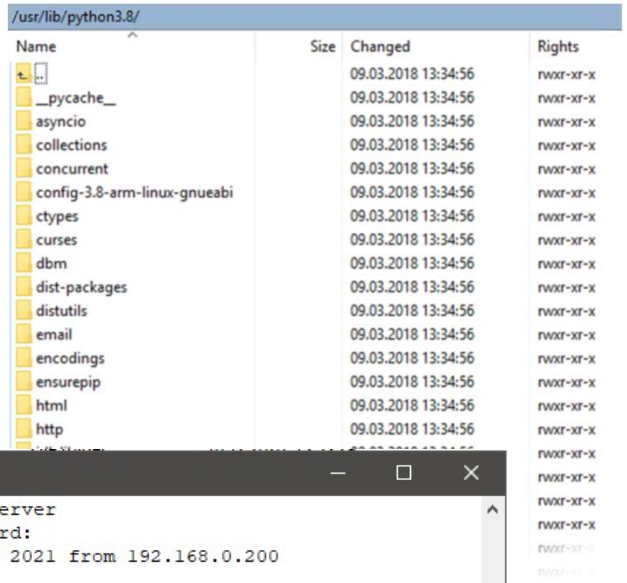
General introduction to Python

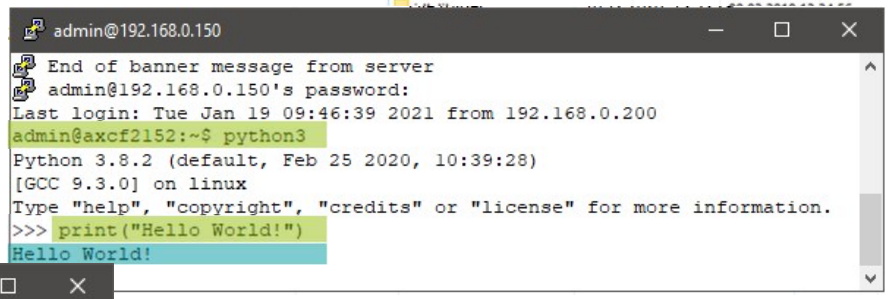
- is an interpreted, high-level programming language
- supports object-oriented and structured programming
- was developed in the early 1990s by Guido van Rossum as a follow-up to the language ABC
- Python is …
  - platform independent
  - characterized by its readability and shortness (for example by using spaces instead of curly brackets)
  - designed to keep the fun in programming and that's why the name "Python" was chosen as a tribute to the comedian group Monty Python
  - executed line by line and converted into low level machine code

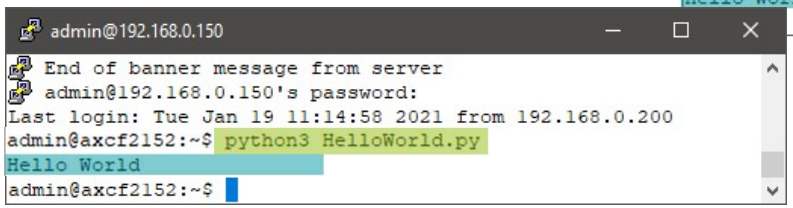
Python examples

Python on PLCnext Control

- Python 3.8 is pre-installed on PLCnext Controls
- Pre-installed packages can be found in /usr/lib/python3.8
- Python code can be executed:
  - directly within the command line
  - as **.py* script







PHOENIX CONTACT
INSPIRING INNOVATIONS

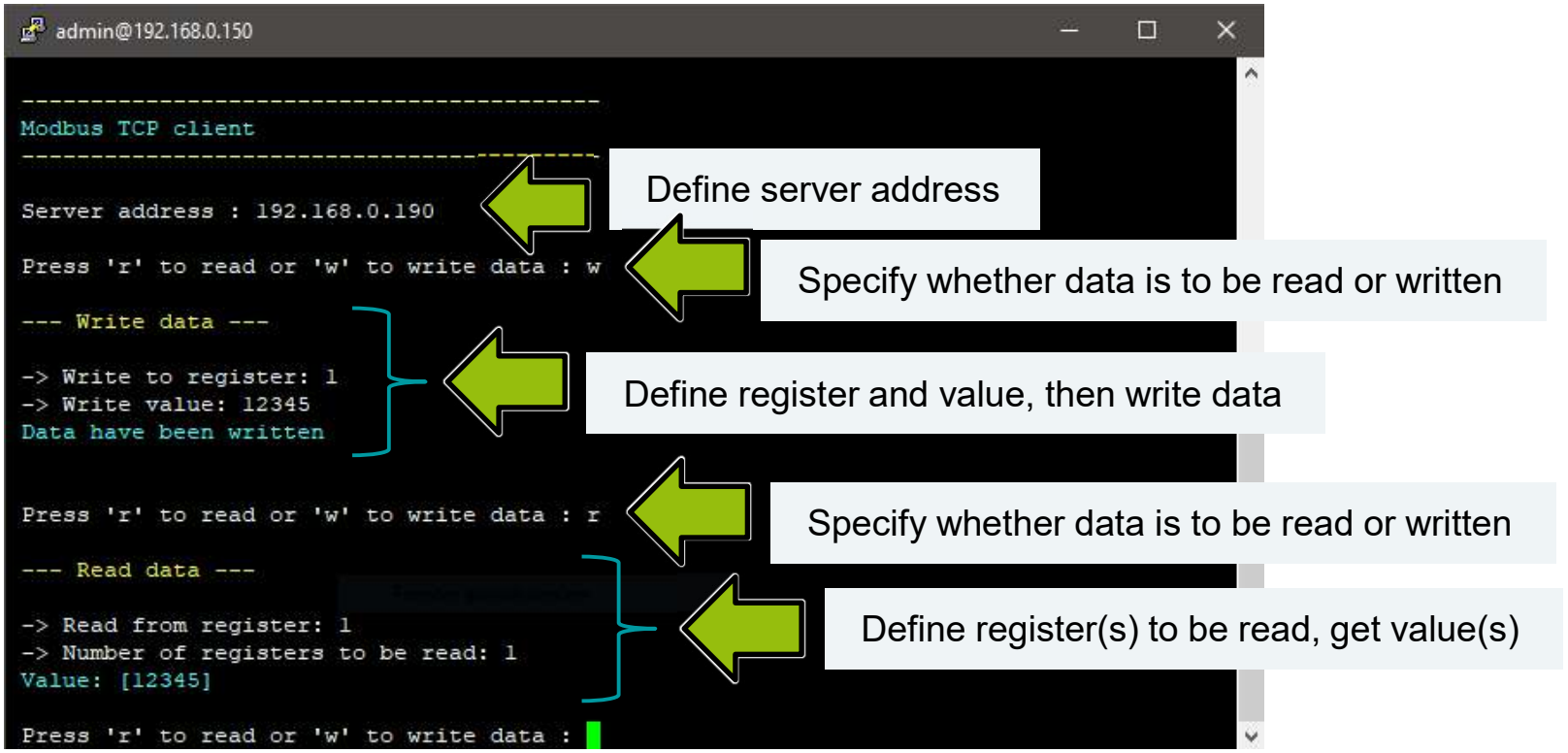
Python examples

Hello World project

Demo

PHOENIX CONTACT
INSPIRING INNOVATIONS

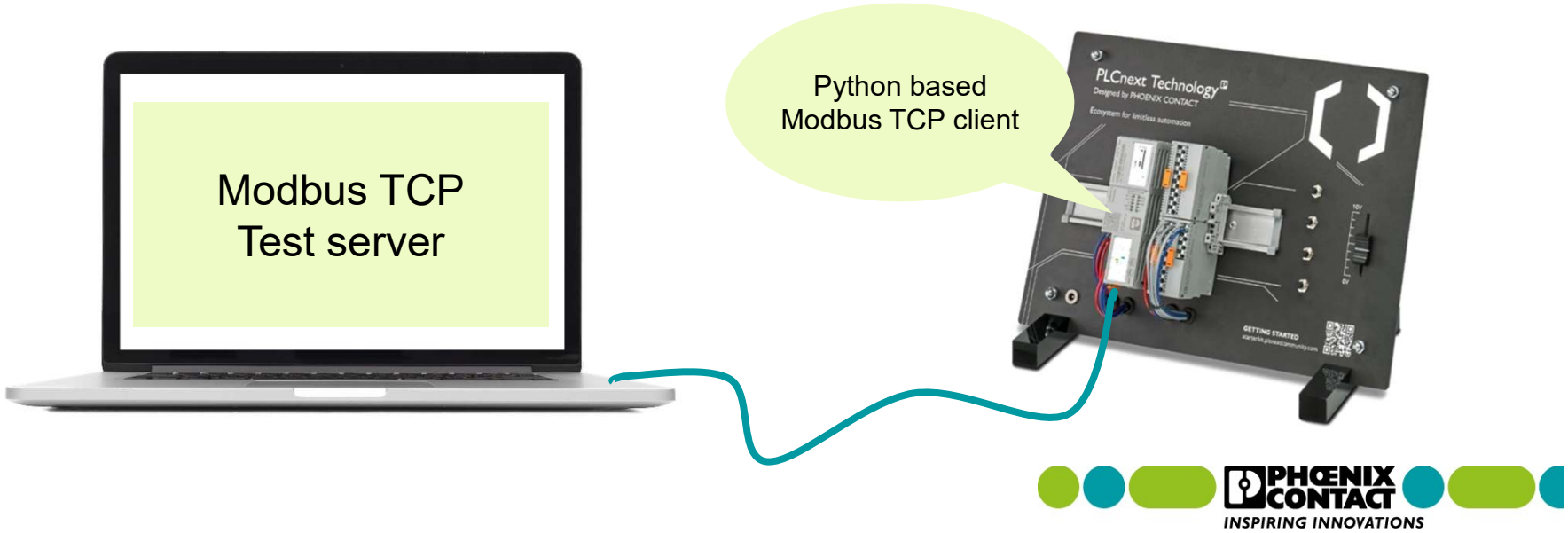
Python examples



Python examples

Modbus TCP with Python

Demo



Python examples

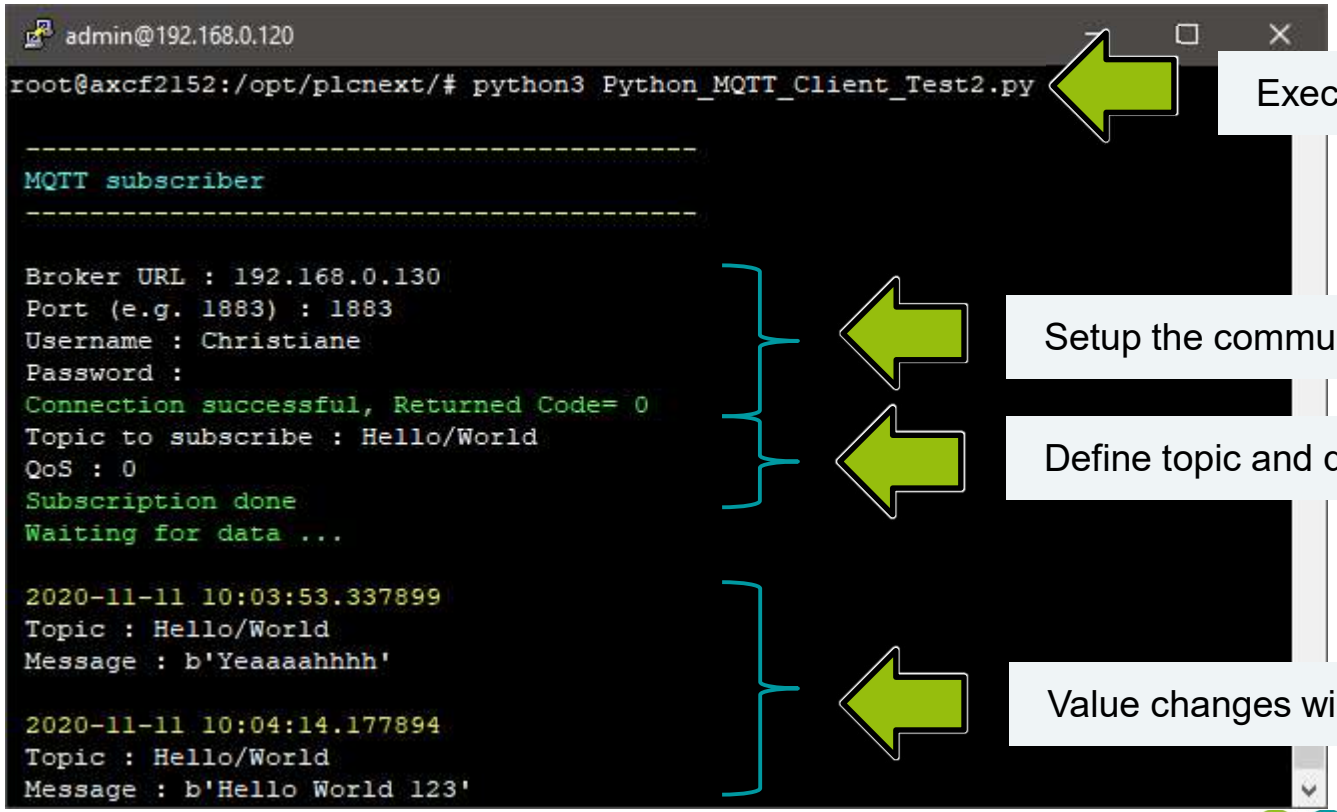
Python code creation

```
import paho.mqtt.client as mqtt
import time
import getpass

print("\033[1;33;40m \n ----------------------------------------")
print("\033[1;36;40m MQTT publisher")
print("\033[1;33;40m ------------

# function definition
def on_connect(client, userdata,
    if rc==0:
        int("\0 ;40m Conn
    e
        t("\033 0m Conn

# create
client = i ient()

# get us t for  ion e
broker_ tr(inpu 3[1;37
broker_ int(i 033[1;3

user t("\03 40m Usern
pswd = getpass.getpass("\033[1;37

# try to connect to broker
client.username_pw_set(user, pswd
client.connect(broker_url, broker_port)
client.on_connect = on_connect

# repeat subscription until keyboard interrupt
while True:
    try:
        client.loop_start()
        time.sleep(1)
        topic = str(input("\033[1;37;40m Topic: "))
        qos_level = int(input("\033[1;37;40m QoS : "))
        retain = bool(input("\033[1;37;40m Retain (True/False) : "))
        payload = str(input("\033[1;37;40m Message : "))
```

Useful information and examples can be found here, for example:

- https://pypi.org/project/paho-mqtt/
- https://github.com/eclipse/paho.mqtt.python/tree/master/examples

Python examples



Execute the python script

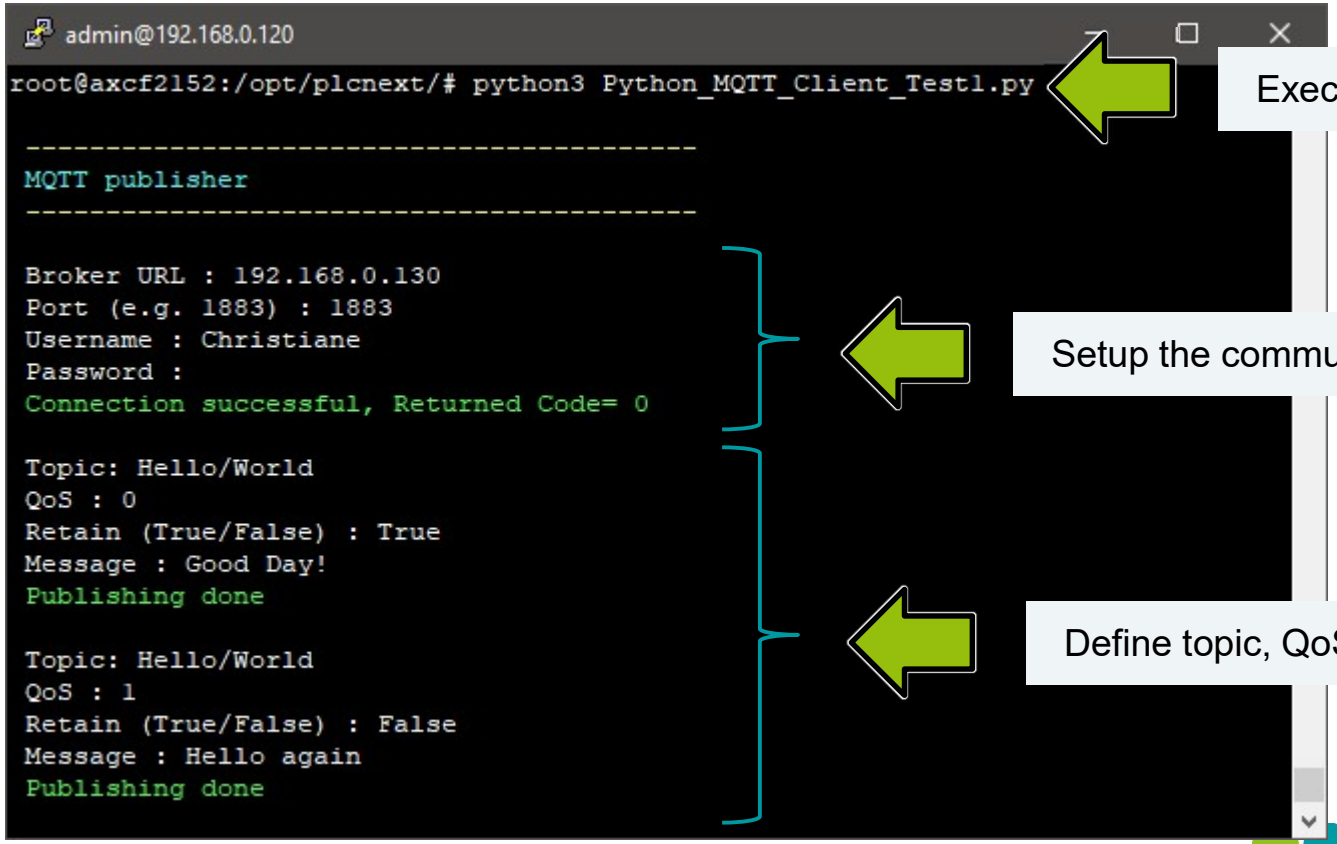
Setup the communication to broker

Define topic and quality of service level

Value changes will be displayed, incl. time

PHOENIX CONTACT
INSPIRING INNOVATIONS

Python examples



Execute the python script

Setup the communication to broker

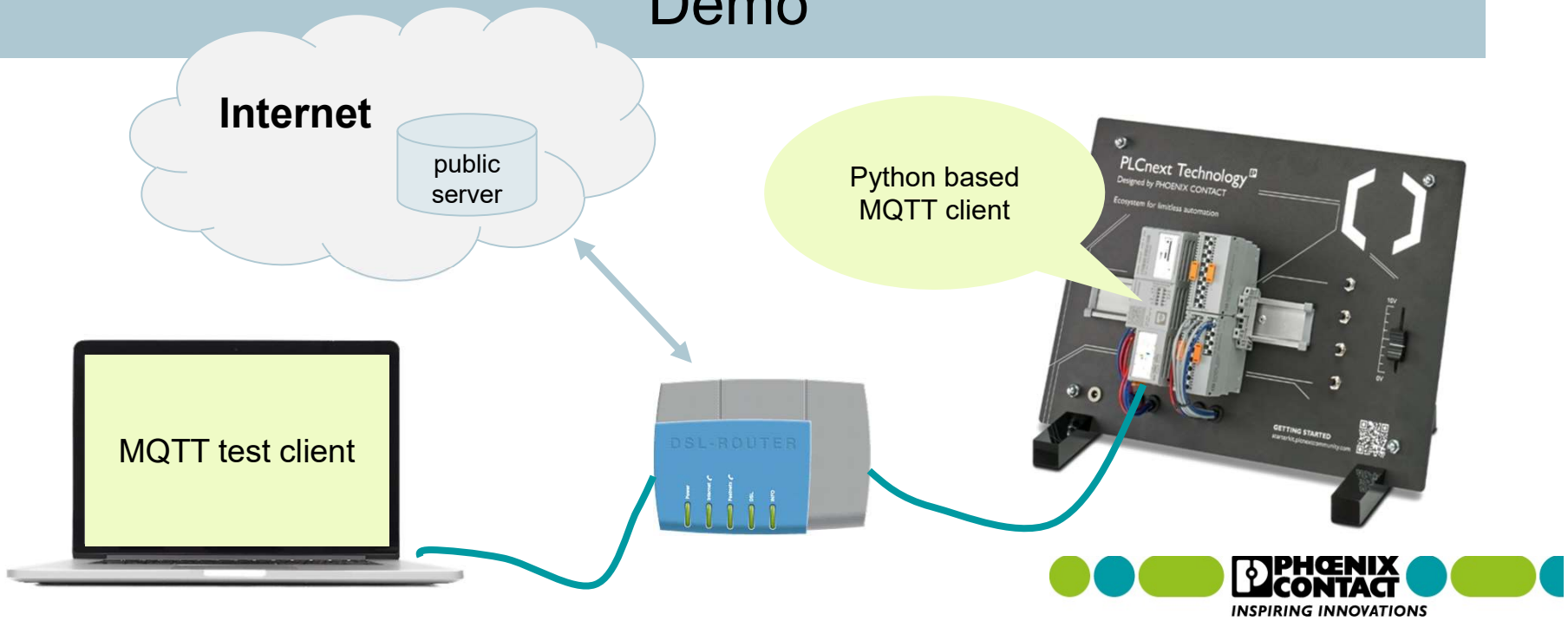
Define topic, QoS, retain and payload

PHOENIX CONTACT
INSPIRING INNOVATIONS

Python examples

MQTT with Python

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

Further information and examples

- Python in Industrial Automation (plcnext-community.net)
- Modbus TCP with Python on AXC F 2152 (plcnext-community.net)
- OpenCV - Python, Red Light detection on PLCnext (plcnext-community.net)
- Machine Learning on PLCnext (plcnext-community.net)

The new era of automation

PLCnext Technology – C++ Components and RSC services

C++ Components

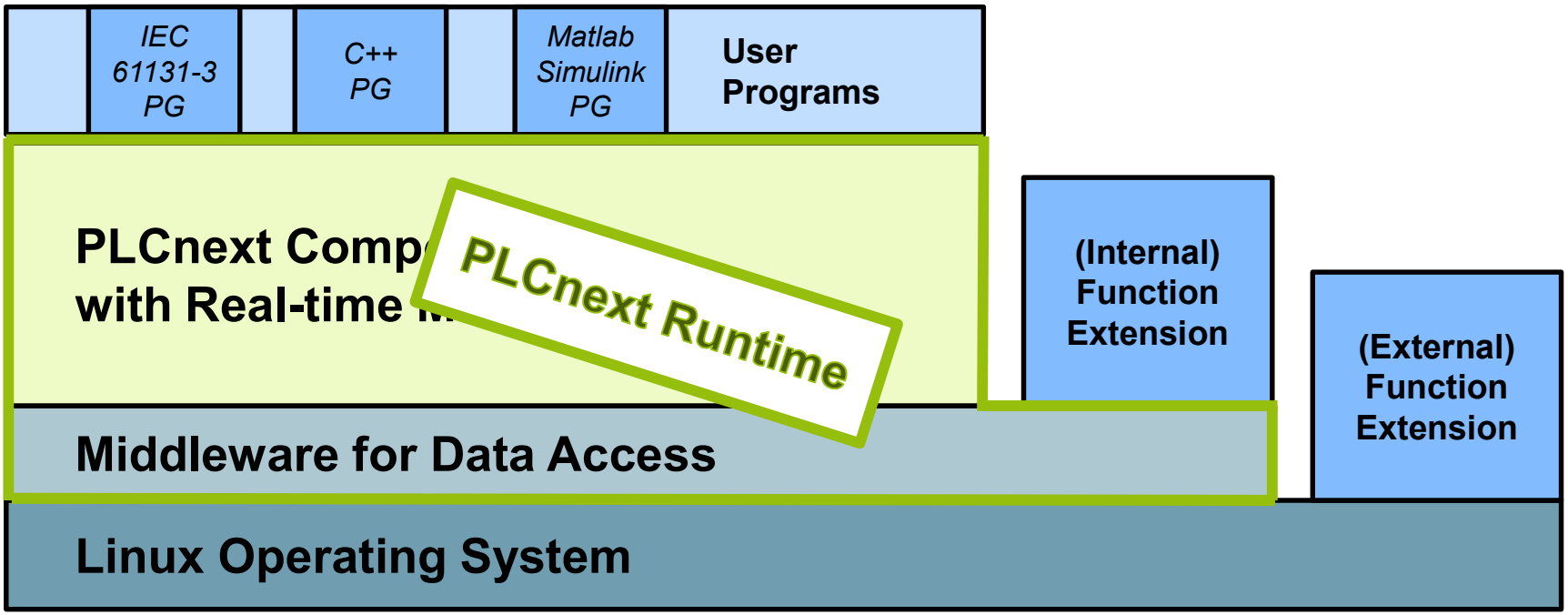
Agenda

- General introduction
- Required software for C++ Programming
- C++ project structure
- C++ Component functions
- C++ Worker thread
- C++ Component interfaces (GDS ports)
- C++ Component instantiation
- C++ Remote Service Calls

PHOENIX CONTACT
INSPIRING INNOVATIONS

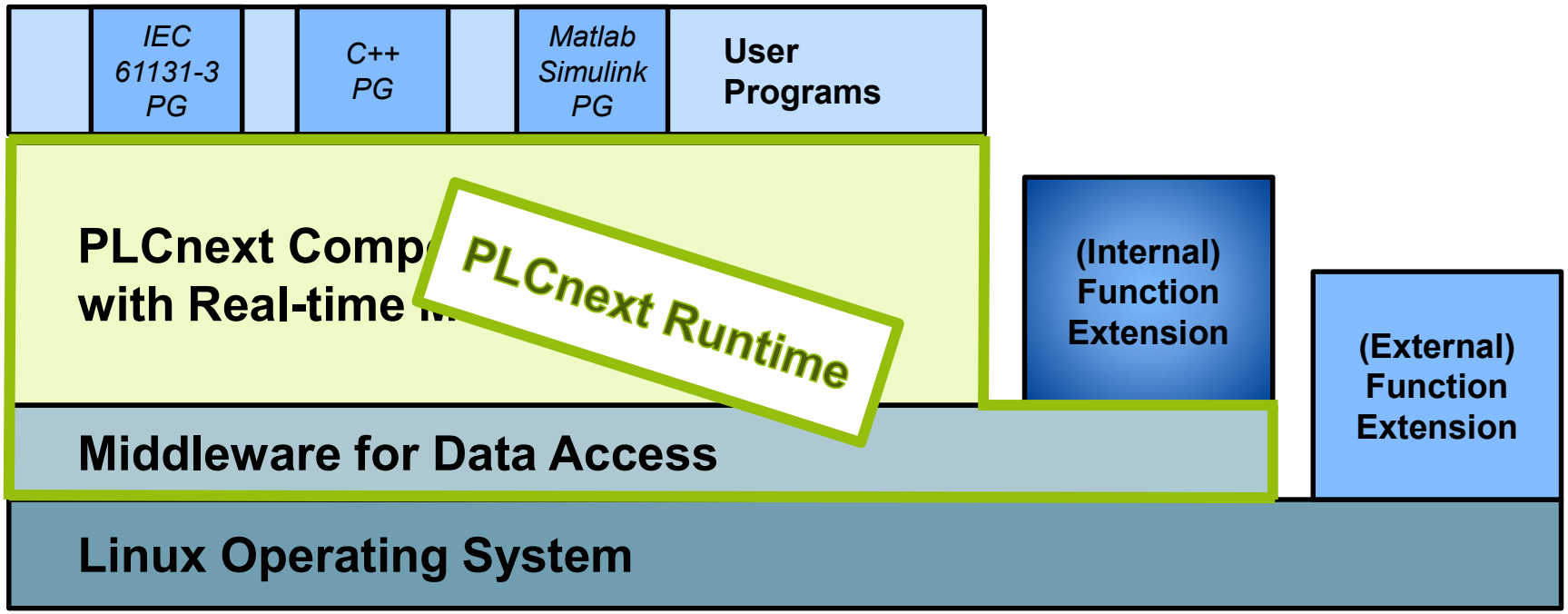
C++ Components

PLCnext Control - System architecture



C++ Components

PLCnext Control - System architecture



PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Agenda

- General introduction
- Required software for C++ Programming
- C++ project structure
- C++ Component functions
- C++ Worker thread
- C++ Component interfaces (GDS ports)
- C++ Component instantiation
- C++ Remote Service Calls (Overview)

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Required software

C++ with Eclipse	C++ with Visual Studio	C++ with any further editor
- Eclipse ≥ Neon	- Visual Studio 2019	- Preferred C++/text editor
- Command-Line Interface (PLCnCLI) - Software Development Kit (SDK) - PLCnext Plugin for Eclipse	- Command-Line Interface (PLCnCLI) - Software Development Kit (SDK) - PLCnext Plugin for Visual Studio	- Command-Line Interface (PLCnCLI) - Software Development Kit (SDK)

Available as one bundle on the Phoenix Contact homepage

More information: https://www.plcnext.help/te/Programming/Cpp/Cpp_programming/Required_Installations.htm

PHOENIX CONTACT
INSPIRING INNOVATIONS

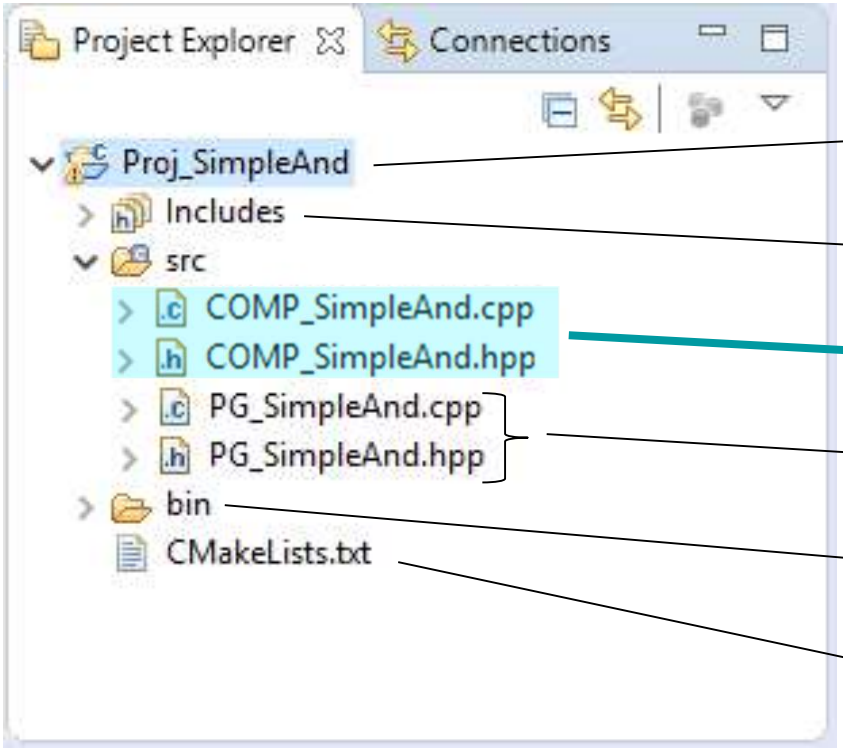
C++ Components

Agenda

- General introduction
- Required software for C++ Programming
- C++ project structure
- C++ Component functions
- C++ Worker thread
- C++ Component interfaces (GDS ports)
- C++ Component instantiation
- C++ Remote Service Calls (Overview)

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ project structure



- Project name
- Includes – Pre-included classes
- Component files
- Program files
- Folder with compilation output
- CMake list for cross-compilation

C++ Components

Templates for *.cpp and *.hpp

COMP_SimpleAnd.hpp

```hpp
#pragma once
#include "Arp/System/Core/Arp.h"
#include "Arp/System/Acf/ComponentBase.hpp"
#include "Arp/System/Acf/IApplication.hpp"
#include "Arp/Plc/Commons/Esm/ProgramComponentBase.hpp"
#include "COMP_SimpleAndProgramProvider.hpp"
#include "Arp/Plc/Commons/Meta/MetaLibraryBase.hpp"
#include "Arp/System/Commons/Logging.h"

namespace LIB_SimpleAnd
{

using namespace Arp;
using namespace Arp::System::Acf;
using namespace Arp::Plc::Commons::Esm;
using namespace Arp::Plc::Commons::Meta;

//#component
class COMP_SimpleAnd : public ComponentBase, public ProgramComponentBase, private Loggable<COMP_SimpleAnd>
{
public: // typedefs

public: // construction/destruction
    COMP_SimpleAnd(IApplication& application, const String& name);
    virtual ~COMP_SimpleAnd() = default;

public: // IComponent operations
    void Initialize() override;
    void LoadConfig() override;
    void SetupConfig() override;
    void ResetConfig() override;

public: // ProgramComponentBase operations
    void RegisterComponentPorts() override;
```

COMP_SimpleAnd.cpp

```cpp
#include "COMP_SimpleAnd.hpp"
#include "Arp/Plc/Commons/Esm/ProgramComponentBase.hpp"
#include "Proj_SimpleAndLibrary.hpp"

namespace LIB_SimpleAnd
{

COMP_SimpleAnd::COMP_SimpleAnd(IApplication& application, const String& name)
: ComponentBase(application, ::LIB_SimpleAnd::Proj_SimpleAndLibrary::GetInstance(), name, ComponentCategory::Cust
, programProvider(*this)
, ProgramComponentBase(::LIB_SimpleAnd::Proj_SimpleAndLibrary::GetInstance().GetNamespace(), programProvider)
{
}

void COMP_SimpleAnd::Initialize()
{
    // never remove next line
    ProgramComponentBase::Initialize();

    // subscribe events from the event system (Nm) here
}

void COMP_SimpleAnd::LoadConfig()
{
    // load project config here
}

void COMP_SimpleAnd::SetupConfig()
{
    // never remove next line
    ProgramComponentBase::SetupConfig();

    // setup project config here
}
```

COMP_SimpleAnd.hpp

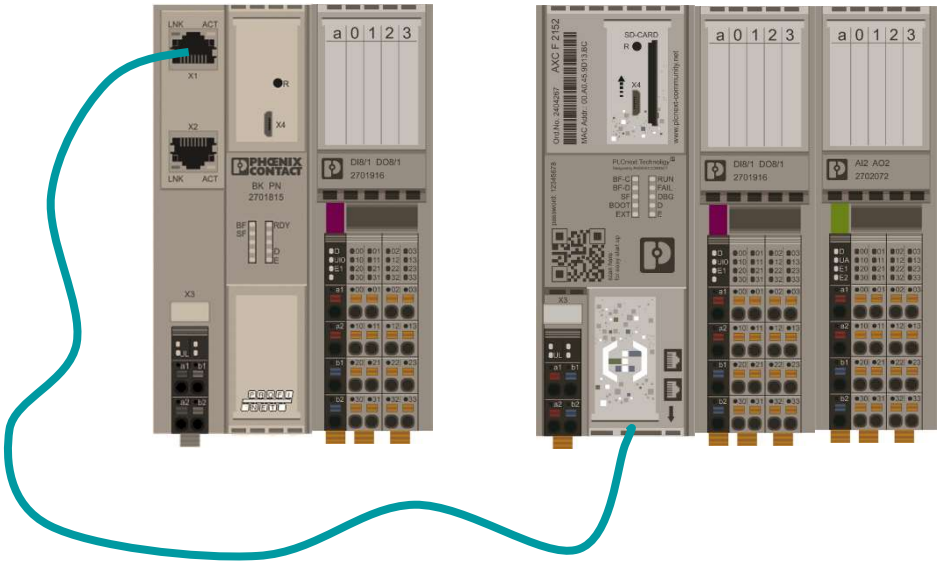
COMP_SimpleAnd.cpp

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Project creation

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

Demo 1: Project creation

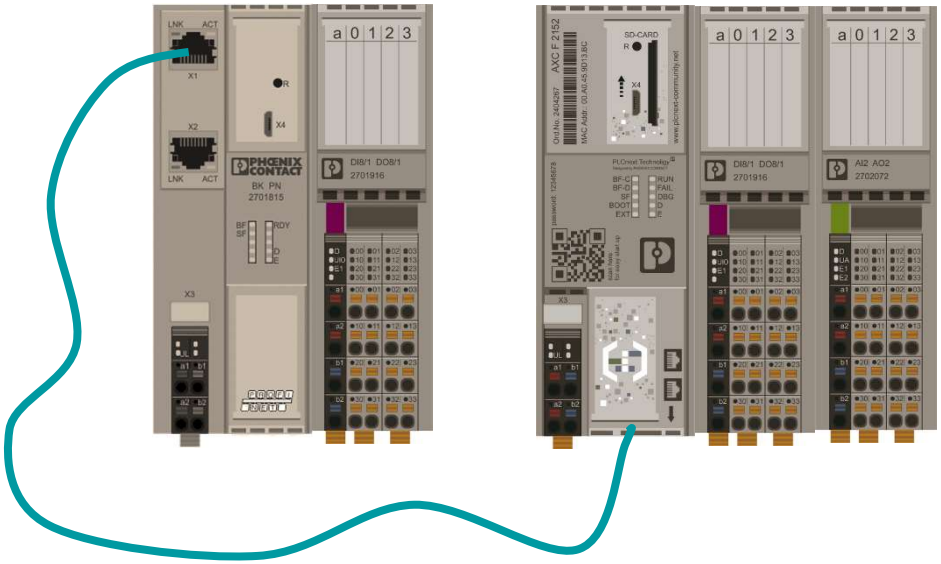
Create a new C++ project and use the following names in it:

Project name:	Proj_AddTwoValues
Component name:	COMP_AddTwoValues
Program name:	PG_AddTwoValues
Project namespace:	LIB_AddTwoValues

C++ Components

Real-time execution via user programs

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

Demo 2: Real-time execution via user programs

1. Program within the user program that two values provided via ports can be added. The result should be provided by the C++ program.
2. Save and compile the project.
3. Insert the created library into your PLCnext Engineer project.
4. Call the program within a cyclic task and assign the ports.
5. Download the project to the PLCnext Control and check whether the result can be calculated correctly.
6. Then switch back to your C++ project, open the component files and gain an overview about the content therein.

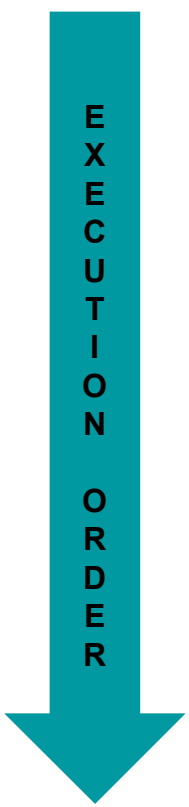
C++ Components

Agenda

- General introduction
- Required software for C++ Programming
- C++ project structure
- C++ Component functions
- C++ Worker thread
- C++ Component interfaces (GDS ports)
- C++ Component instantiation
- C++ Remote Service Calls (Overview)

C++ Components

Pre-defined functions



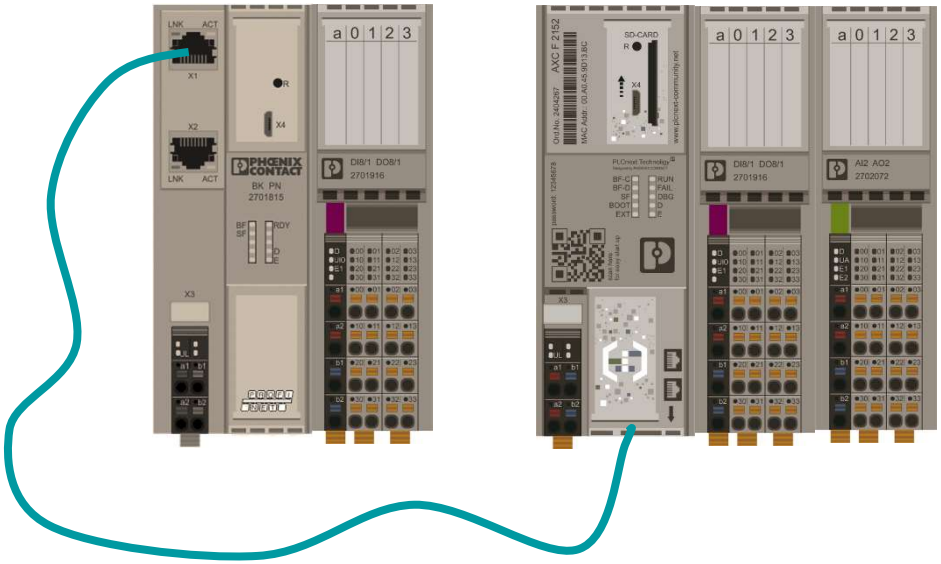

```
void Initialize()
void SubscribeServices()
void LoadSettings(const String& SettingsPath)
void SetupSettings()
void PublishServices()
void LoadConfig()
void SetupConfig()
void ResetConfig()
void Dispose()
```

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Pre-defined functions

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

Demo 3: Pre-defined functions

1. Look at the functions declared in the component header file template. Add "Dispose".
2. Change to the **.cpp* file of the component. Place a programming here, which creates a different log file entry in each function.
3. Save and compile the C++ project.
4. Download the synchronized PLCnext Engineer project to the PLCnext Control.
5. Establish an SFTP connection, e.g. via WinSCP and open the log file.
6. Analyze the order and number of calls of the C++ functions. Also restart the firmware processes via command-line (`sudo /etc/init.d/plcnext restart`) and see which function is called when.

C++ Components

Agenda

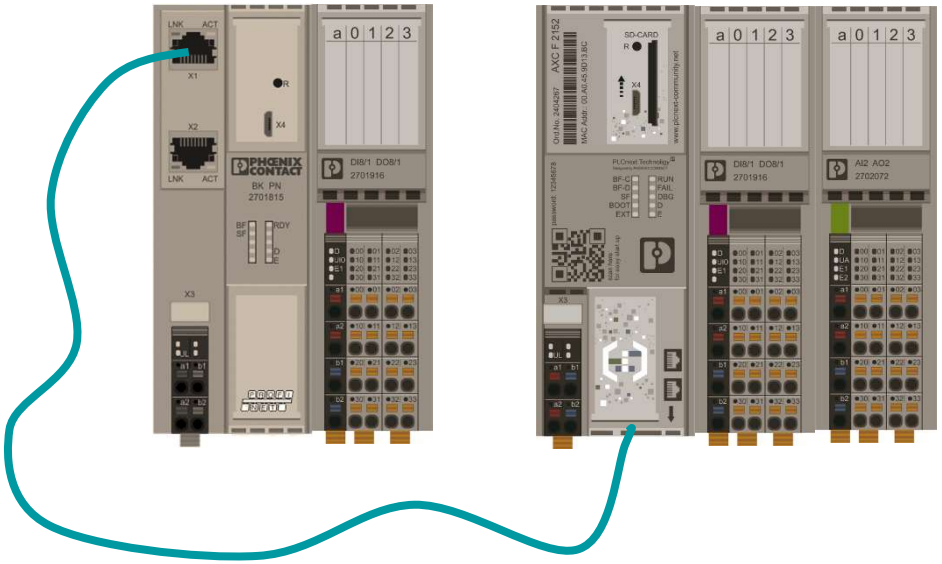
- General introduction
- Required software for C++ Programming
- C++ project structure
- C++ Component functions
- C++ Worker thread
- C++ Component interfaces (GDS ports)
- C++ Component instantiation
- C++ Remote Service Calls (Overview)

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Worker thread

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Demo 4: Worker thread

1. Include the needed header for working with worker threads within the header file.
2. Add the namespace as used namespace within this project.
3. Create a function declaration for the thread function to be processed cyclically.
4. Declare an instance name for the thread.
5. Save your changes and open the **.cpp* file. Here configure the thread.
6. The thread must be started with the call of “LoadConfig” and stopped with “ResetConfig”.
7. Add the thread function and program that a log file entry can be created cyclically.
8. Save and compile the C++ project. Then send the PLCnext Engineer project to the PLC.
9. Now open the log file and check whether you can see the log entries.

C++ Components

Agenda

- General introduction
- Required software for C++ Programming
- C++ project structure
- C++ Component functions
- C++ Worker thread
- C++ Component interfaces (GDS ports)
- C++ Component instantiation
- C++ Remote Service Calls (Overview)

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Port definition within the header file

COMP_SimpleAnd.hpp

```
45
46  public: /* Ports
47              =====
48              Component ports are defined in the following way:
49
50              //#attributes(Hidden)
51              struct Ports
52              {
53                  //#name(NameOfPort)
54                  //#attributes(Input|Retain|Opc)
55                  Arp::boolean portField = false;
56                  // The GDS name is "<componentName>/NameOfPort" if the struct is declared as Hidden
57                  // otherwise the GDS name is "<componentName>/PORTS.NameOfPort"
58              };
59
60              //#port
61              Ports ports;
62
63              Create one (and only one) instance of this struct.
64              Apart from this single struct instance, there must be no other Component variables declared with the #port comment.
65              The only attribute that is allowed on the struct instance is "Hidden", and this is optional.
66              The struct can contain as many members as necessary.
67              The #name comment can be applied to each member of the struct, and is optional.
68              The #name comment defines the GDS name of an individual port element. If omitted, the member variable name is used as the GDS name.
69              The members of the struct can be declared with any of the attributes allowed for a Program port.
70          */
71  };
```

PHOENIX CONTACT
INSPIRING INNOVATIONS

Exemplary port configurations

```
public:

 //#attributes(Hidden)
 struct Ports
 {
   //#name(OUTPORT_xLED)
   //#attributes(Output|Opc)
   Arp::boolean LED = false;
 };

 //#port
 Ports ports;
```

LED is used as variable name within the C++ code, but ***OUTPORT_xLED*** is the port name

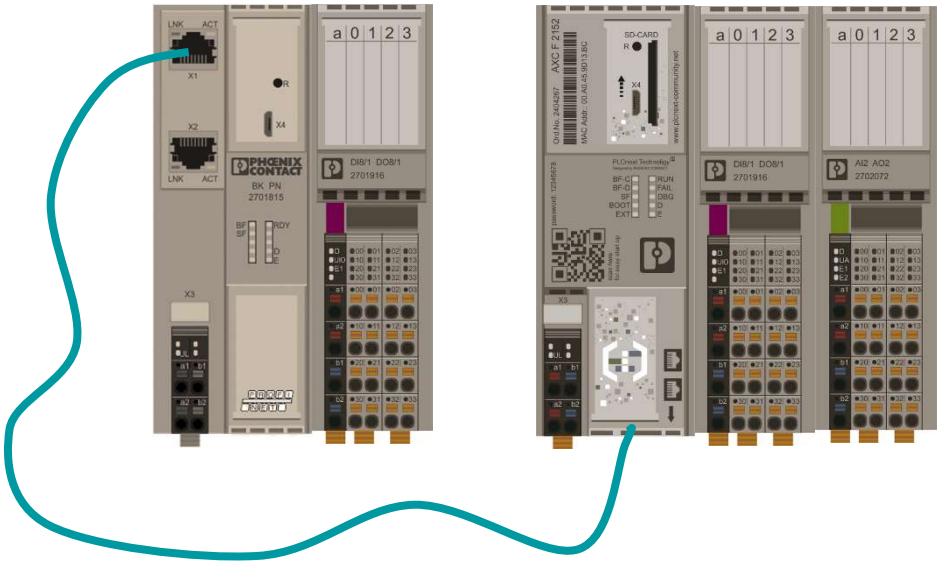
```
public:

 //#attributes(Hidden)
 struct Ports
 {
   //#attributes(Output|Opc)
   Arp::boolean OUTPORT_xLED = false;
 };

 //#port
 Ports ports;
```

OUTPORT_xLED is to be used in the code and as port name

C++ Components

C++ Component ports

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

Demo 5: C++ Component ports

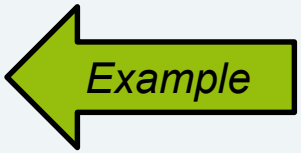
1. Open the **.hpp* file again. Here, create a component port structure with three input port elements (two for the summands, one for the result)
2. Save the changes and open the **.cpp* file of the component. Then insert a programming for the worker thread function. The function should make it possible that whenever the result value changes, the new equation is written to the log file.
3. Save and compile the C++ project.
4. The port assignment can be done within the general port list in PLCnext Engineer.
5. Save and download the project to the PLC.
6. Change the summands several time, e.g. via overwrite in debug mode, and check whether the corresponding entries appear in the log file.

Manual GDS configuration for Component Ports (1)

- Custom GDS file can be stored anywhere on the PLC but must be linked in the default GDS configuration file

→ */opt/plcnext/projects/Default/Plc/Gds/Default.gds.config*

```
<?xml version="1.0" encoding="utf-8" ?>
<GdsConfigurationDocument
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.phoenixcontact.com/schema/gdsconfig"
  schemaVersion="1.0" >

  <Includes>
    <Include path="$ARP_PROJECT_CURRENT_DIR$/Plc/Gds/*.gds.config" />
    <Include path="/opt/plcnext/projects/SimpleAnd/SimpleAnd.gds.config" />
  </Includes>
</GdsConfigurationDocument>
```



Manual GDS configuration for Component Ports (2)

- Custom GDS configuration needs to be done in XML

```
<?xml version="1.0" encoding="utf-8“ ?>
<GdsConfigurationDocument xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemaVersion="1.0"
          xmlns="http://www.phoenixcontact.com/schema/gdsconfig">
  <ComponentTaskRelations />
  <Connectors>
    <Connector startPort="Arp.Io.AxlC/0.IN00" endPort="COMP_SimpleAnd_Test1/INPORT_xVar1" />
    <Connector startPort="Arp.Io.AxlC/0.IN01" endPort="COMP_SimpleAnd_Test1/INPORT_xVar2" />
    <Connector startPort="COMP_SimpleAnd_Test1/OUTPORT_xResult" endPort="Arp.Io.AxlC/0.OUT00" />
  </Connectors>
</GdsConfigurationDocument>
```

C++ Components

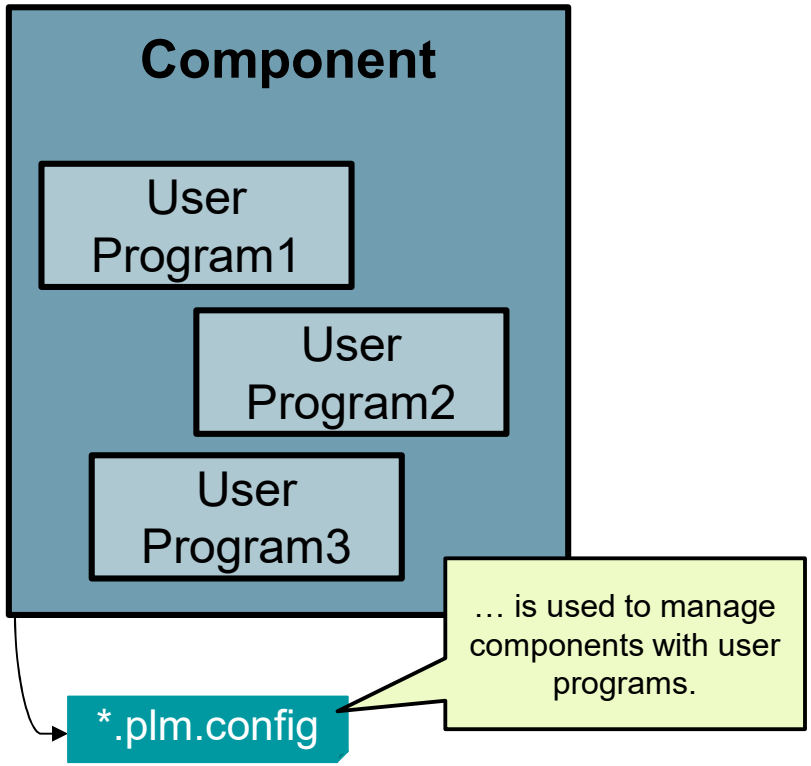
Agenda

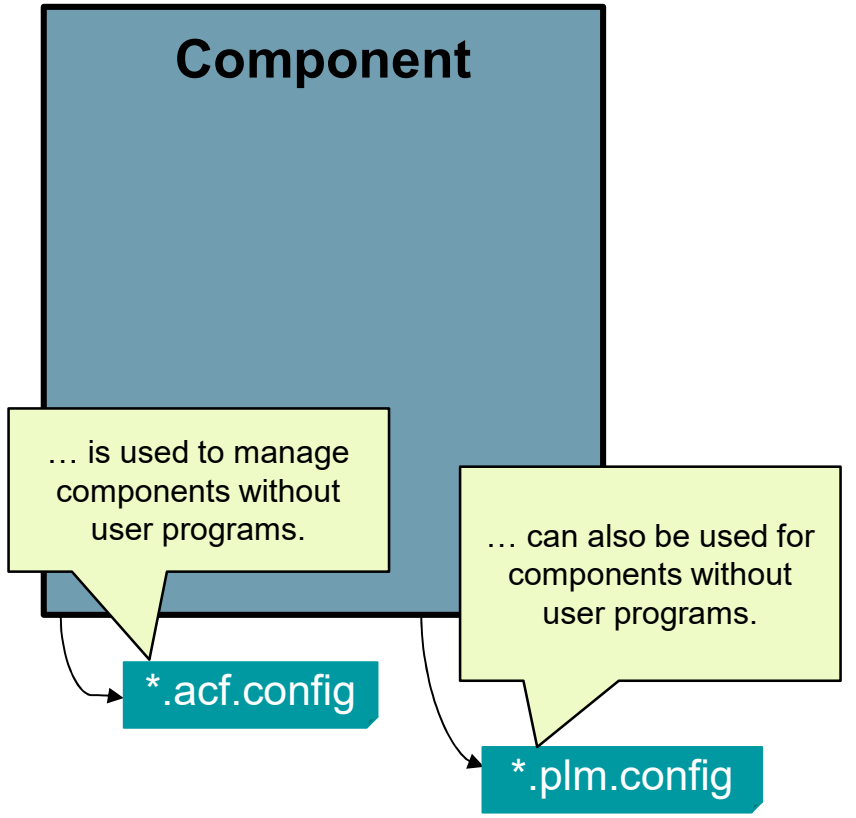
- General introduction
- Required software for C++ Programming
- C++ project structure
- C++ Component functions
- C++ Worker thread
- C++ Component interfaces (GDS ports)
- C++ Component instantiation
- C++ Remote Service Calls (Overview)

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Component instantiation





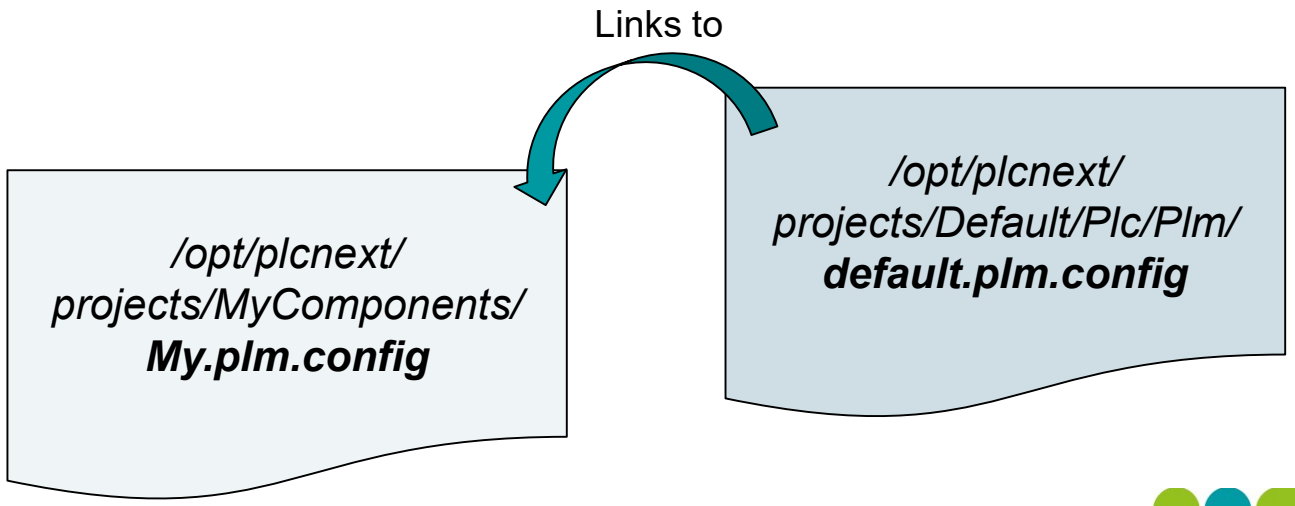
PHOENIX CONTACT
INSPIRING INNOVATIONS

Comparison of PLM and ACF instantiation

Program Library Manager (PLM)	Application Component Framework (ACF)
Both use XML as language within the configuration file, and the XML structure is very similar	
Only components managed by the PLM can also provide programs that can be instantiated in ESM tasks	ACF components can not provide user programs.
Only components that are managed by the PLM can be stopped, changed and started by downloading from the PLCnext Engineer. This also applies to ESM tasks and the programs instantiated therein.	For components managed by the ACF, the firmware must be stopped, started, or restarted. So, components managed by the ACF will persist even if the PLC program is stopped, deleted or started.
Access to the GDS can be done by using ports.	

Manual PLM configuration files

- Custom PLM file can be stored anywhere on the PLC but must be linked within the default PLM configuration file
- **Example:**



C++ Components

ACF configuration files

→ */opt/plcnext/projects/Default/<constum name>.acf.config*

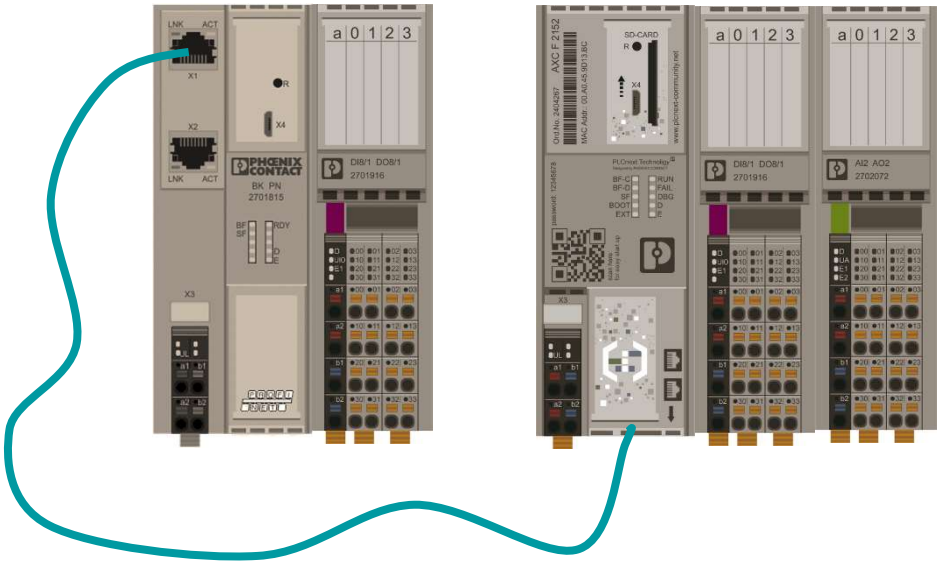
```
<?xml version="1.0" encoding="UTF-8"?>
<AcfConfigurationDocument
  xmlns="http://www.phoenixcontact.com/schema/acfconfig"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.phoenixcontact.com/schema/acfconfig.xsd"
  schemaVersion="1.0" >
  <Processes>
    <Process name="Proj_RSC_ServicesProcess" settingsPath="$ARP_ACF_SETTINGS_FILE$" />
  </Processes>
  <Libraries>
    <Library name="LIB_RSC_Services.Proj_RSC_ServicesLibrary" binaryPath="/opt/plcnext/projects/MyComponents/libProj_RSC_Services.so" />
  </Libraries>
  <Components>
    <Component name="COMP_RSC_Services1" type="LIB_RSC_Services::COMP_RSC_Services“
               library="LIB_RSC_Services.Proj_RSC_ServicesLibrary" />
  </Components>
</AcfConfigurationDocument>
```

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Creation of an ACF Component

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

Demo 6: Creation of an ACF component

1. Create a new C++ project, but now choose „PLCnext ACF project“ as project template.
2. Use the following names:

Project name:	Proj_RscServices
Component name:	COMP_RscServices
Project namespace:	RscServices

3. Then create a worker thread like before for the PLM component.
4. Save and compile the project.
5. Copy the shared object (**.so*) and the adjusted ACF configuration file to the PLC.
6. Restart the PLCnext processes via command.
7. Open the log file to check if the ACF component and the worker thread can be executed .

C++ Components

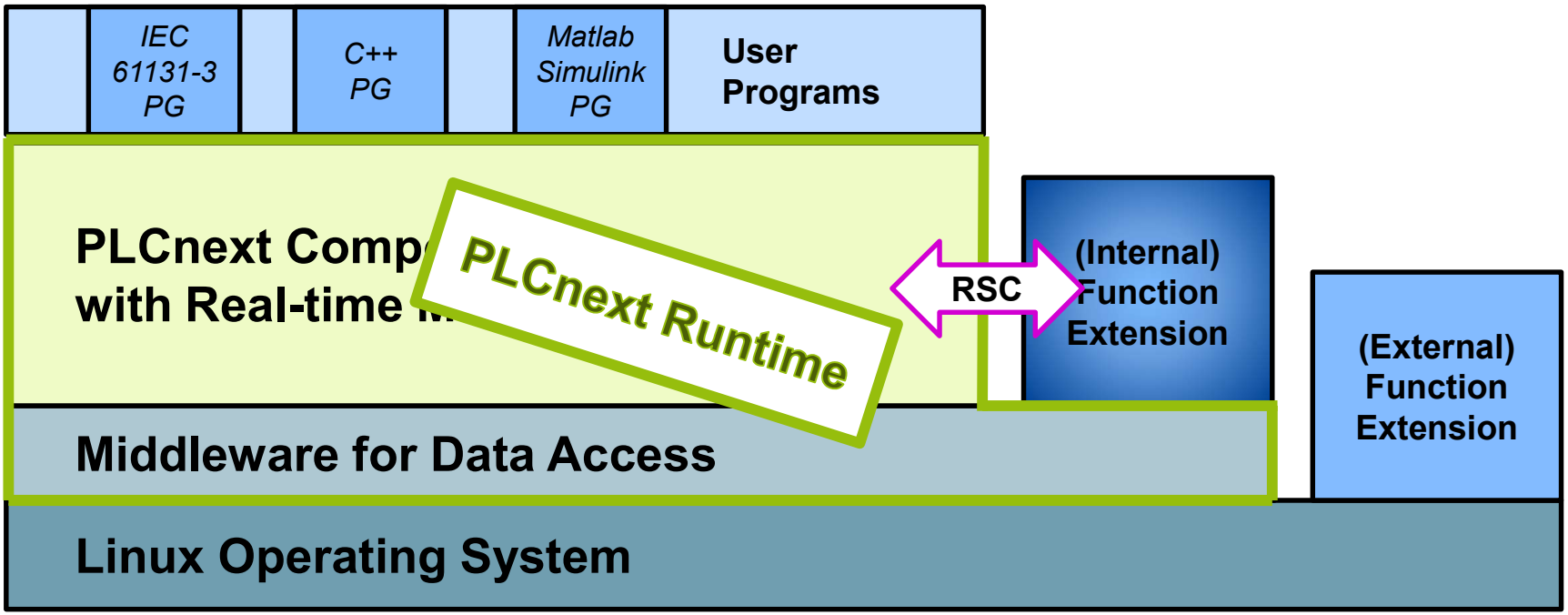
Agenda

- General introduction
- Required software for C++ Programming
- C++ project structure
- C++ Component functions
- C++ Worker thread
- C++ Component interfaces (GDS ports)
- C++ Component instantiation
- C++ Remote Service Calls (Overview)

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Remote Service Calls (RSC services)



C++ Components

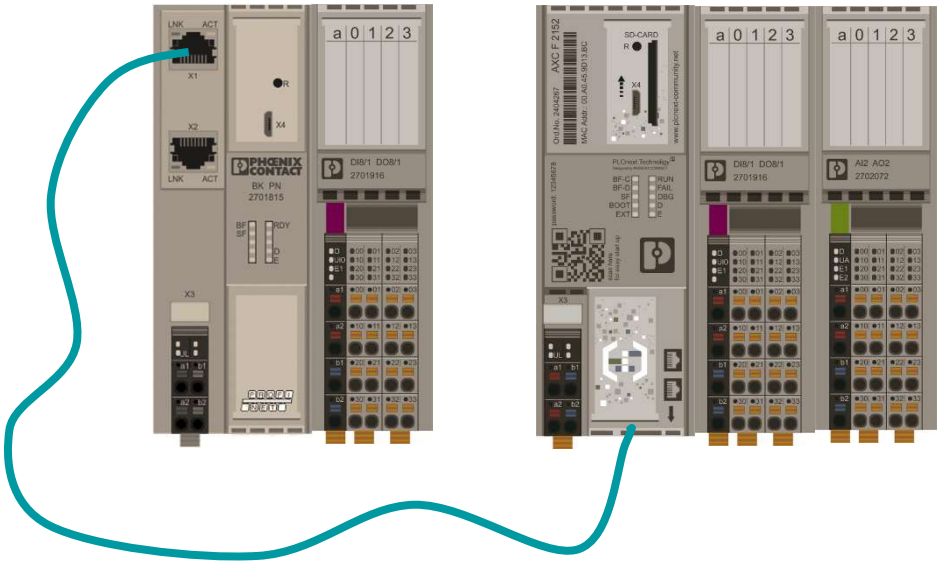
Services

Service	Namespace	Functions
IAcyclicCommunicationService	Arp/Io/**Axioline**/Services	PdiRead, PdiWrite
IAcyclicCommunicationService	Arp/Io/**ProfinetStack**/Controller/Service	RecordRead, RecordWrite
IDeviceInfoService	Arp/**Device**/Interface/Services	GetItem, GetItems
IDeviceStatusService	Arp/**Device**/Interface/Services	GetItem, GetItems
IDataAccessService	Arp/Plc/**Gds**/Services	Read, ReadSingle, Write, WriteSingle

C++ Components

RSC service for Axioline

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Demo 7: RSC services for Axioline

1. Open the header file of this ACF component and include the needed header file.
2. Add the namespace for the RSC service manager as used namespace.
3. Define variables:

```
Arp::Io::Axioline::Services::IAcyclicCommunicationService::Ptr pAxioAcyclicCommunicationService;
Arp::Io::Axioline::Services::PdiParam AxioPdiParameters;
std::vector<uint8> vAxioPdiData;
bool xProductNameWritten = false;
```

4. Save the changes and open the **.cpp* file. Here, subscribe the RSC service and read the **product name**. The result of this read process has to be written to the log file.
5. Save and compile the C++ project.
6. Overwrite the shared object on the PLC
7. Restart the firmware processes and check if you can see the product name in the log file.

C++ Components

Demo 7: RSC services for Axioline

Manual for AXL F DI8/1 DO8/1 1H : 8670_en_03

Index (hex)	Object name	Object type	Data type	A	L	Rights	Meaning	C
Manufacturer								
0001	VendorName	Var	Visible String	1	16	R	Manufacturer name	Pl
0002	VendorID	Var	Visible String	1	7	R	Manufacturer identification	0(
0003	VendorText	Var	Visible String	1	49	R	Comment on the manufacturer	C sy tri
0012	VendorURL	Var	Visible String	1	30	R	URL of the manufacturer	ht co
Module - general								
0004	DeviceFamily	Var	Visible String	1	19	R	Device family	I/(
0006	ProductFamily	Var	Visible String	1	33	R	Product family	A: sp
000E	CommProfile	Var	Visible String	1	4	R	Communication profile	6:
000F	DeviceProfile	Var	Visible String	1	5	R	Device profile	0010
0011	ProfileVersion	Record	Visible String	2	11; 20	R	Device profile version	2011-12-07; Basic Profile V2.0
003A	VersionCount	Array	Unsigned 16	4	4 * 2	R	Version counter	e.g., 0007 0001 0000 0000
Module - special								
0005	Capabilities	Array	Visible String	1	8	R	Properties	Nothing
0007	ProductName	Var	Visible String	1	21	R	Product designation	AXL F DI8/1 DO8/1 1H
0008	SerialNo	Var	Visible String	1	11	R	Serial number	xxxxxxxxxx (e. g.

Diagnostics state (0018_{hex}: DiagState)

This object is used for a structured message of an error.

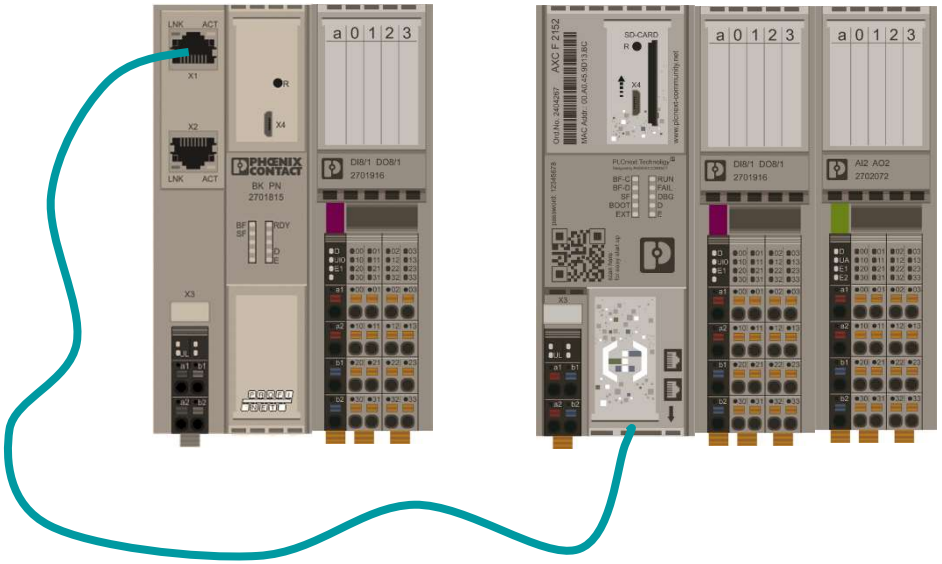
0018_{hex}: DiagState (Read)					
Subindex	**Data type**	**Length in bytes**	**Meaning**	**Contents**	
0	Record	8	Diagnostic state	Complete diagnostics information	
1	Unsigned 16	2	Error number	0 ... 65535_{dec}	
2	Unsigned 8	1	Priority	00_{hex}	No error
				01_{hex}	Error
				02_{hex}	Warning
				81_{hex}	Error removed
				82_{hex}	Warning eliminated
3	Unsigned 8	1	Group	00_{hex}	No error
				FF_{hex}	entire device
4	Unsigned 16	2	Error code	See table below	
5	Unsigned 8	1	More information follows	00_{hex} (not supported)	
6	Visible String	1	Text	00_{hex} (not supported)	

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

RSC service for Profinet

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Demo 8: RSC services for Profinet

1. Open the header file of the ACF component and include the needed header file.
2. Define variables:

```
Arp::Io::ProfinetStack::Controller::Services::IAcyclicCommunicationService::Ptr pPnioAcyclicCommunicationService;
Arp::Io::ProfinetStack::Controller::Services::RecordParam PnioRecordParameters;
std::vector<uint8> vPnioRecordData;
bool xPnioDiagDataWritten = false;
```

3. Save the changes and open the **.cpp* file. Here, subscribe the RSC service and read the **PN diagnostic data**. The result of this read process has to be written to the log file.
4. Save and compile the C++ project.
5. Overwrite the shared object on the PLC
6. Restart the firmware processes and check if you can see the data in the log file.

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Demo 8: RSC services for Profinet

Manual for AXL F BK PN : *105731_en_05*

The diagnostic status register is located in byte 0 and byte 1.

In the example, the following response is stored in variable RECORD:

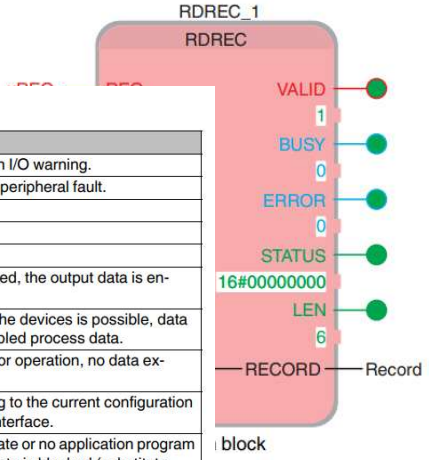
Record	
[1]	16#00
[2]	16#E0
[3]	16#00
[4]	16#00
[5]	16#00
[6]	16#00
[7]	16#00
[8]	16#00
[9 ... 15]	16#00

In this example, the diagnostic status register has the value $00E0_{hex}$ ($0000\ 0000\ 1110\ 0000_{bin}$). Bits 5, 6, and 7 are TRUE. This means that data cycles are being exchanged, the configuration is active, and the local bus master is ready.

The diagnostic status register can be read using asynchronous services (index 2210_{hex}, 8720_{hex}).

Structure of the diagnostic status register

Bit	Designation	Meaning	
00	F_PW_BIT	I/O warning	At least one device indicates an I/O warning.
01	F_PF_BIT	Peripheral fault	At least one device indicates a peripheral fault.
02	F_BUS_BIT	Bus error	A bus error has occurred.
03	-	Reserved	
04	-	Reserved	
05	F_RUN_BIT	Run	Data cycles are being exchanged, the output data is enabled.
06	F_ACTIVE_BIT	Active	Configuration is active, PDI to the devices is possible, data exchange with invalid/non-enabled process data.
07	F_READY_BIT	Ready	The local bus master is ready for operation, no data exchange over the bus.
08	F_BD_BIT	Bus different	A device which does not belong to the current configuration has been detected at the last interface.
09	F_BASP_BIT	SYS_FAIL	The controller is in the STOP state or no application program has been loaded. The output data is blocked (substitute value behavior is active).
10	F_FORCE_BIT	Force Mode	Force mode (startup tool/I/O check is active).
11	F_SYNC_BIT	Synchronization	Synchronization between higher-level system and local bus master has failed.
12	F_PARA_REQ	Module error	At least one device is requesting parameters.
13 ... 15	-	Reserved	



Further information:

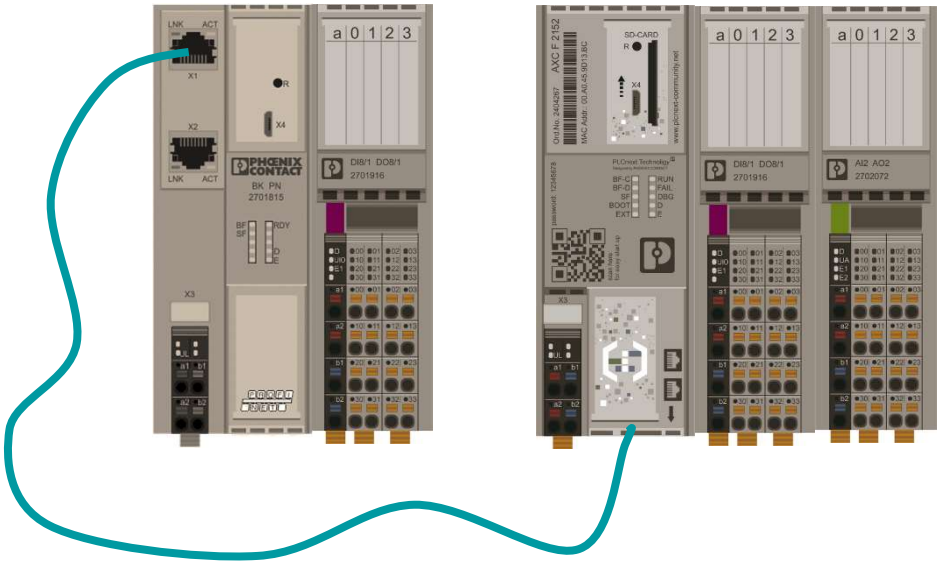
https://www.plcnext.help/te/Service_Components/Remote_Service_Calls_RSC/RSC_PROFINET_Services.htm

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

RSC services for device interface data

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Demo 9: RSC services for device interface data

1. Open the header file of the ACF component and include the needed header file.
2. Define the needed variables:

```
Arp::Device::Interface::Services::IDeviceInfoService::Ptr pDeviceInfoService;
RscString<512> Parameter;
RscVariant<512> DeviceInterfaceServiceData;
bool xDeviceInterfaceDataWritten = false;
```

3. Save the changes and open the **.cpp* file. Here, subscribe the RSC service and read the **firmware status**. The result of this read process has to be written to the log file.
4. Save and compile the C++ project.
5. Overwrite the shared object on the PLC
6. Restart the firmware processes and check if you can see the firmware in the log file.

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Component

Demo 9: RSC services for device interface data

The following parameters are available in the `IDeviceInfoService` RSC interface for calling of information:

Parameter	Data type	Description
General.DeviceClass	UInt32	The `DeviceClass` parameter specifies the device class. At the moment, only "ProgrammableLogicController" is supported. 0: Undefined 1: ProgrammableLogicController 2. BusCoupler 3: Switch
General.VendorName	String	The `VendorName` parameter indicates the name of the manufacturer.
General.ArticleName	String	The `ArticleName` parameter indicates the device name.
General.ArticleNumber	String	The `ArticleNumber` parameter indicates the order number of the device.
General.SerialNumber	String	The `SerialNumber` parameter indicates the serial number of the device.
General.Firmware.Version	String	The `FirmwareVersion` parameter indicates the firmware version of the device. Here, the 5-level notation (Major, Minor, Patch, Build, Status) is used.
General.Firmware.VersionMajor	Byte	The firmware version year is indicated without the first two digits. E.g., "2019" is indicated as "19".
General.Firmware.VersionMinor	Byte	`FirmwareVersionMinor`
General.Firmware.VersionPatch	Byte	`FirmwareVersionPatch`
General.Firmware.VersionBuild	UInt32	`FirmwareVersionBuild`
General.Firmware.VersionStatus	String	`FirmwareVersionStatus`
General.Firmware.BuildDate	String	`FirmwareBuildDate` ISO 8601 format <YYYY>-<MM>-<DD>
General.Firmware.BuildTime	String	`FirmwareBuildTime` ISO 8601 format <hh>:<mm>:<ss>
General.Hardware.Version	String	The `HardwareVersion` parameter indicates the hardware version of the device.
General.Fpga.Version	String	The `FPGAVersion` parameter indicates the FPGA version of the device. Here, the 3-level notation (Major, Minor, Patch) is

Further information:

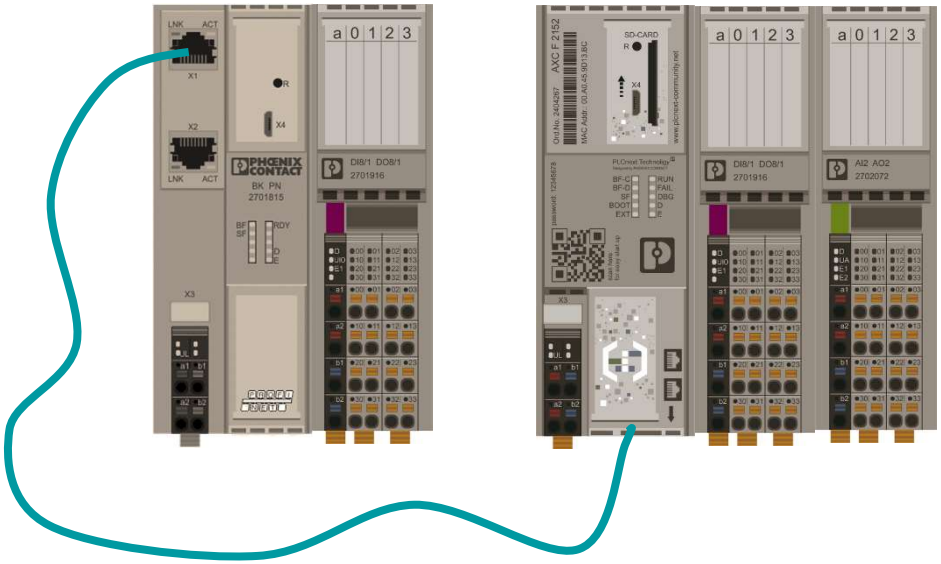
https://www.plcnext.help/te/Service_Components/Remote_Service_Calls_RSC/RSC_device_interface_services.htm

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

RSC service for GDS access

Demo



PHOENIX CONTACT
INSPIRING INNOVATIONS

Demo 10: RSC services for GDS access

1. Open the header file of the ACF component and include the needed header file.
2. Define the needed variables:

```
Arp::Plc::Gds::Services::IDataAccessService::Ptr pDataAccessService;
RscString<512> PortName;
Arp::Plc::Gds::Services::ReadItem PortData;
Arp::Plc::Gds::Services::ReadItem prevPortData;
```

3. Save the changes and open the **.cpp* file. Here, subscribe the RSC service and read the **result of the add operation**. The result has to be written to the log file.
4. Save and compile the C++ project.
5. Overwrite the shared object on the PLC
6. Restart the firmware processes and check if you can see the result in the log file.

Further information

E-Learning to ACF Component and Axioline RSC service	>> Link
Program Library Manager (PLM)	>> Link
Application Component Framework (ACF)	>> Link
General information to RSC services	>> Link1 >> Link2
Axioline RSC service	>> Link
Profinet RSC service	>> Link
Device Information Service	>> Link
GDS access via RSC service	>> Link

Tools

- Node RED
- MQTT
- DOCKER

Node RED

PHOENIX CONTACT
INSPIRING INNOVATIONS

The new era of automation

PLCnext Technology – MQTT examples

C++ Components

Agenda

- General introduction to MQTT
- MQTT with Node-RED
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT with Python
  - Installation of the paho-mqtt library
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT in IEC61131 (preview)
- PLCnext Control as MQTT broker
- Q&A

PHOENIX CONTACT
INSPIRING INNOVATIONS

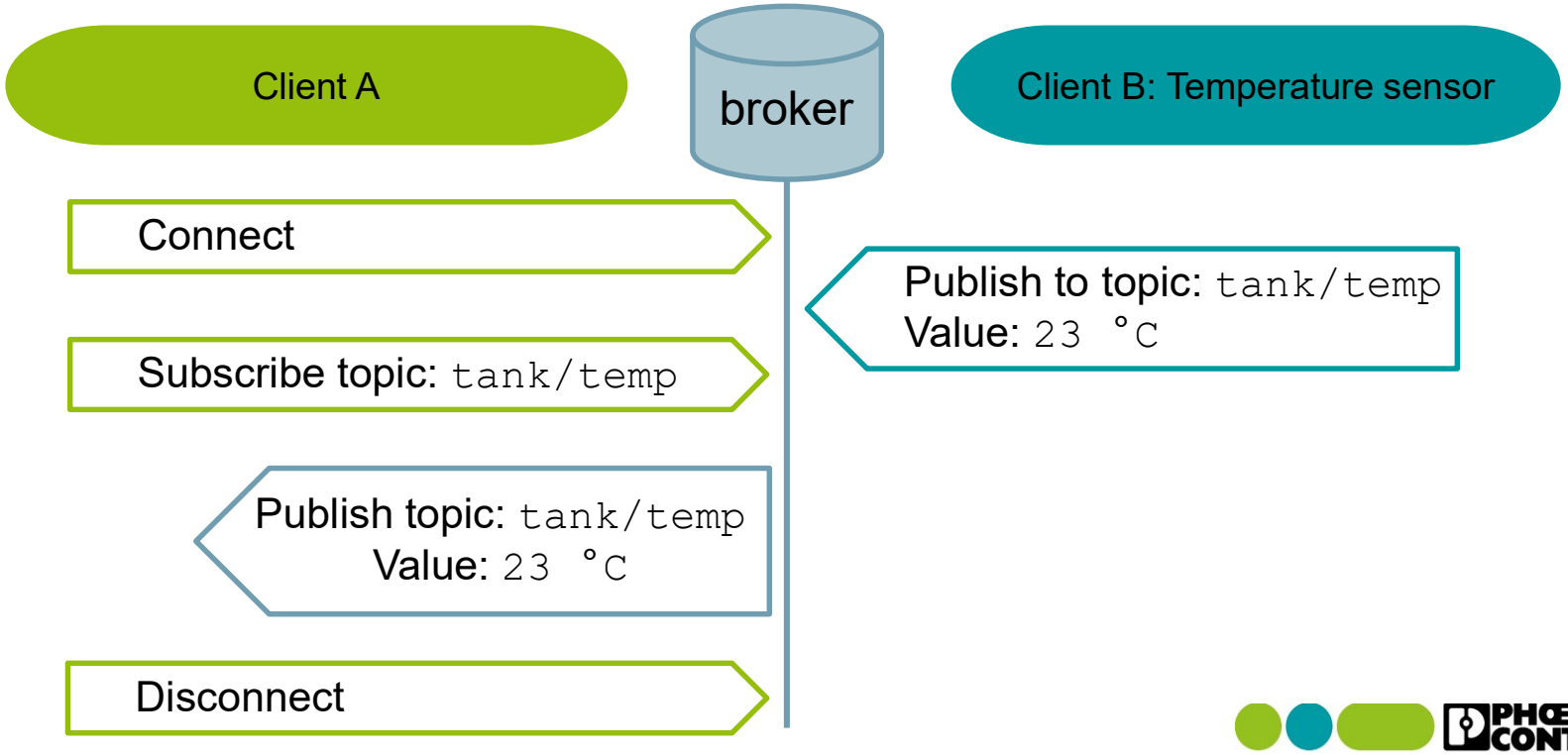
MQTT examples

MQTT (Message Queuing Telemetry Transport)

- TCP/IP message protocol for machine-to-machine communication
- Published in 1999 by IBM
- Standard since 2014
- Uses a publish-subscribe messaging pattern
- Client connects to a server which is called „broker“
- Client can publish or subscribe information
- The broker distributes a message to any client with a subscription.

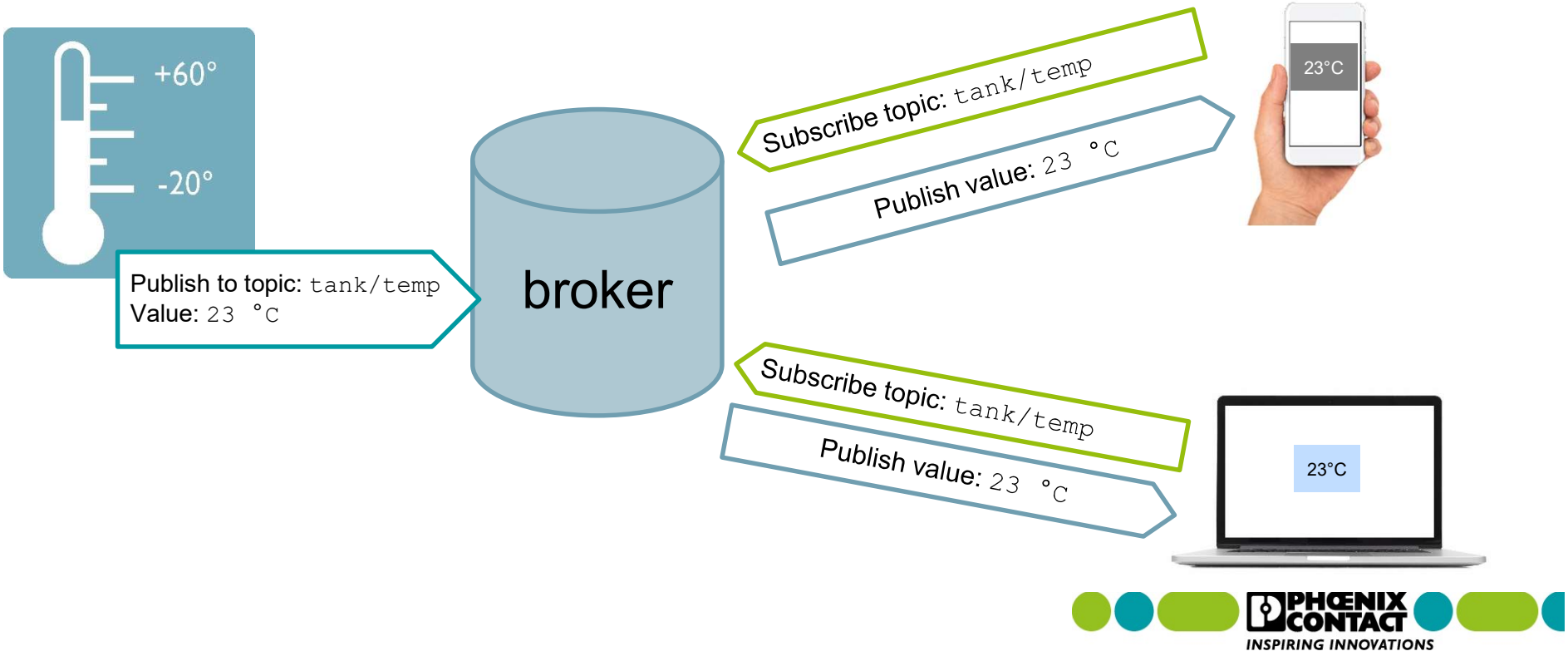
PHOENIX CONTACT
INSPIRING INNOVATIONS

Example of an MQTT connection



MQTT examples

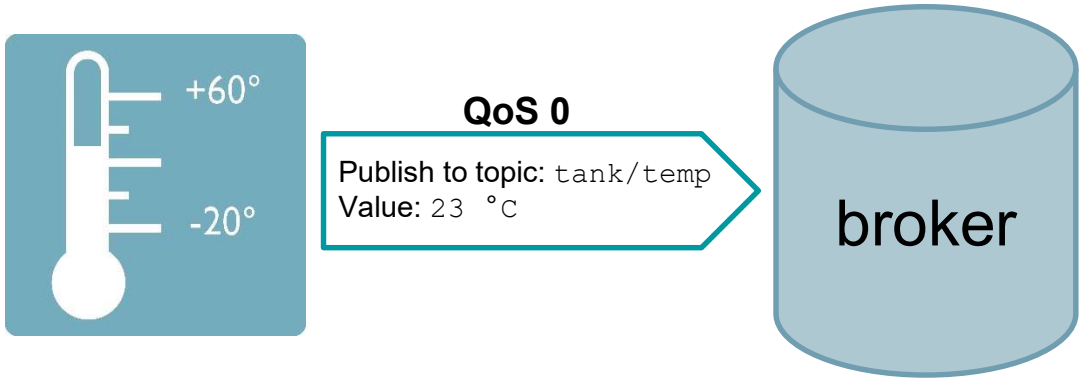
Multiple subscriptions



PHOENIX CONTACT
INSPIRING INNOVATIONS

Quality of Service (QoS)

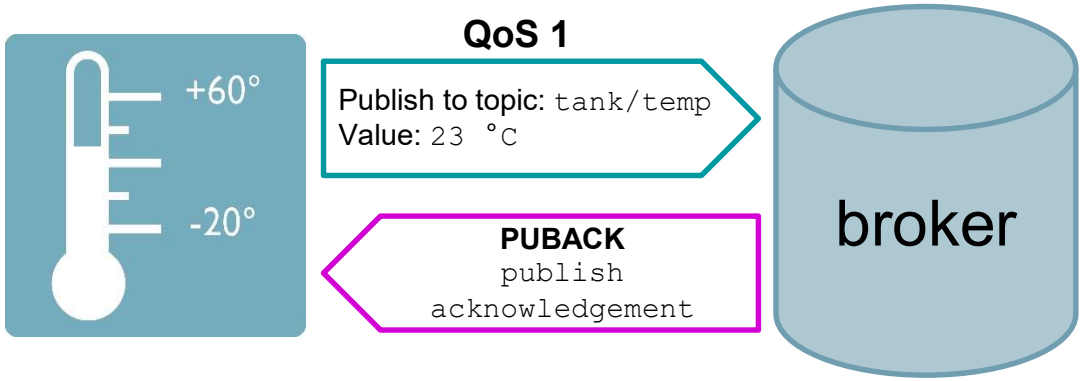
- Determines how the message is sent
  - 0: „at most once“ → message is sent without acknowledgement of the receiver



MQTT examples

Quality of Service (QoS)

- Determines how the message is sent
  - 0: „at most once“ → message is sent without acknowledgement of the receiver
  - 1: „at least once“ → message is sent at least once, with acknowledgement

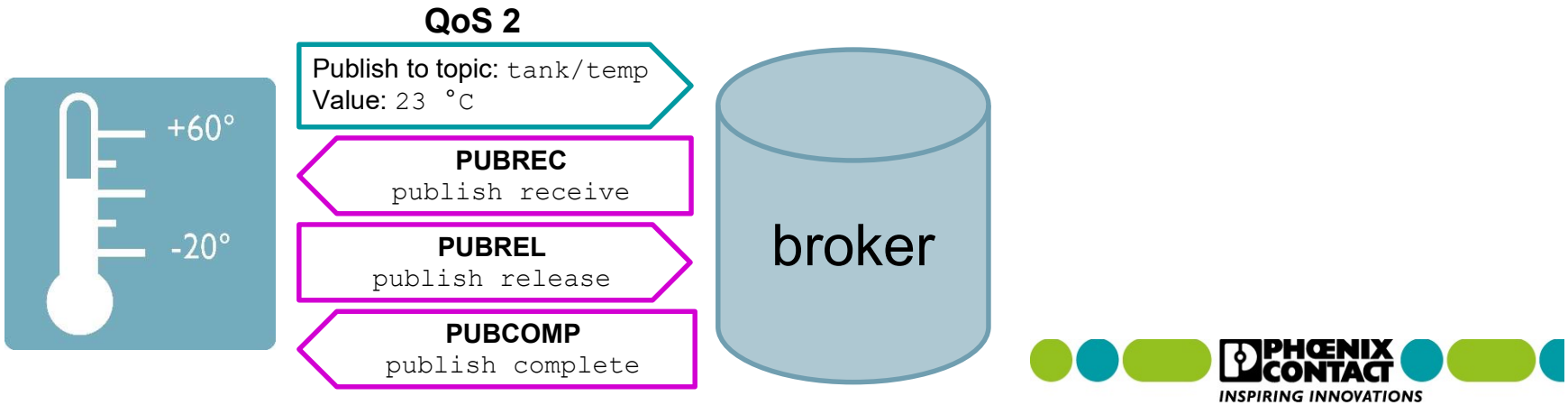


PHOENIX CONTACT
INSPIRING INNOVATIONS

MQTT examples

Quality of Service (QoS)

- Determines how the message is sent
  - 0: „at most once“ → message is sent without acknowledgement of the receiver
  - 1: „at least once“ → message is sent once, with acknowledgement
  - 2: „exactly once“ → 2-level handshake between sender and receiver to ensure that only one message is received



PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Agenda

- General introduction to MQTT
- MQTT with Node-RED
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT with Python
  - Installation of the paho-mqtt library
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT in IEC61131 (preview)
- PLCnext Control as MQTT broker
- Q&A

PHOENIX CONTACT
INSPIRING INNOVATIONS

Required hardware and installations

- PLCnext Control with Node-RED
  - Steps for installation via docker:

    https://www.plcnext-community.net/en/hn-makers-blog/481-node-red-and-getting-started-with-docker.html

    https://www.plcnext-community.net/en/hn-makers-blog/482-node-red-with-docker-tips-and-best-practice.html
  - Steps for offline installation:

    https://www.plcnext-community.net/en/hn-makers-blog/418-install-node-red-and-pm2-offline.html

CK3

MQTT examples

MQTT with Node-RED

Demo

PHOENIX CONTACT
INSPIRING INNOVATIONS

Slide 138

CK3 Christiane Kownatzki, 10/12/2020

Example: Publish and subscribe topics in one flow

- Task: In this example of a Node-RED flow, a string value should be subscribed and published via MQTT.

- The topic name is: ***MyHome/LivingRoom/Light***.

- The payload string can be: ***“1” or “0”***.

C++ Components

Agenda

- General introduction to MQTT
- MQTT with Node-RED
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT with Python
  - Installation of the paho-mqtt library
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT in IEC61131 (preview)
- PLCnext Control as MQTT broker
- Q&A

PHOENIX CONTACT
INSPIRING INNOVATIONS

Installation of paho-mqtt

1) Download the paho-mqtt library: https://pypi.org/project/paho-mqtt/#files

2) Transfer the*.*tar.gz* file to your PLCnext Control, e.g. via WinSCP.

3) Extract the file with the command:

```
tar –xf paho-mqtt-<version>.tar.gz
```

4) Login as root user.

5) Move the source files to the Python3.8 library folder:

```
mv paho-mqtt-<version>/src/paho /usr/lib/python3.8/
```

Alternative installation of paho-mqtt

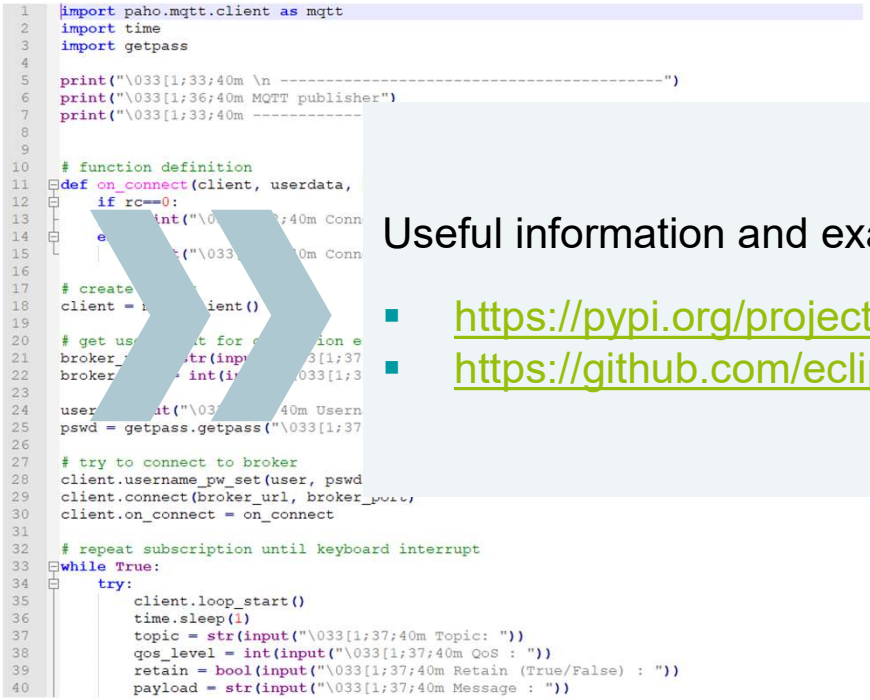
1) Install the Python package manager PIP as described here:

   https://www.plcnext-community.net/en/hn-makers-blog/425-installing-pip-without-ipkg.html

2) User the following command to install paho-mqtt via PIP: `pip install paho-mqtt`

MQTT examples

Python code creation



```
import paho.mqtt.client as mqtt
import time
import getpass

print("\033[1;33;40m \n ----------------------------------------")
print("\033[1;36;40m MQTT publisher")
print("\033[1;33;40m ------------

# function definition
def on_connect(client, userdata,
    if rc==0:
        int("\0      ;40m Conn
    e
        t("\033      0m Conn

# create
client = m      ient()

# get us      t for      ion e
broker_      tr(inp      3[1;37
broker_      int(i      033[1;3

user      t("\03      40m Usern
pswd = getpass.getpass("\033[1;37

# try to connect to broker
client.username_pw_set(user, pswd
client.connect(broker_url, broker_port,
client.on_connect = on_connect

# repeat subscription until keyboard interrupt
while True:
    try:
        client.loop_start()
        time.sleep(1)
        topic = str(input("\033[1;37;40m Topic: "))
        qos_level = int(input("\033[1;37;40m QoS : "))
        retain = bool(input("\033[1;37;40m Retain (True/False) : "))
        payload = str(input("\033[1;37;40m Message : "))
```

Useful information and examples can be found here, for example:

- https://pypi.org/project/paho-mqtt/
- https://github.com/eclipse/paho.mqtt.python/tree/master/examples

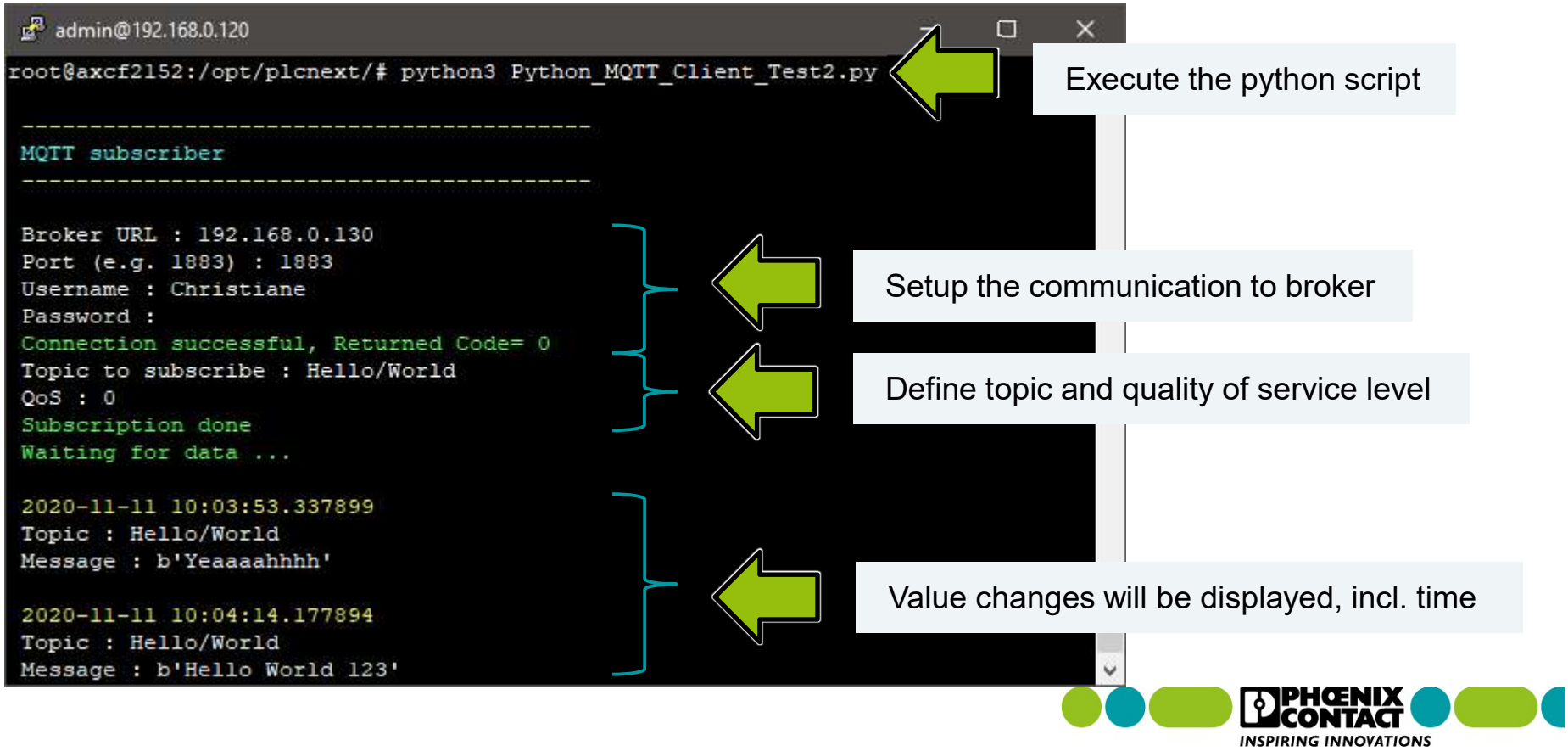
MQTT examples

MQTT with Python

Demo

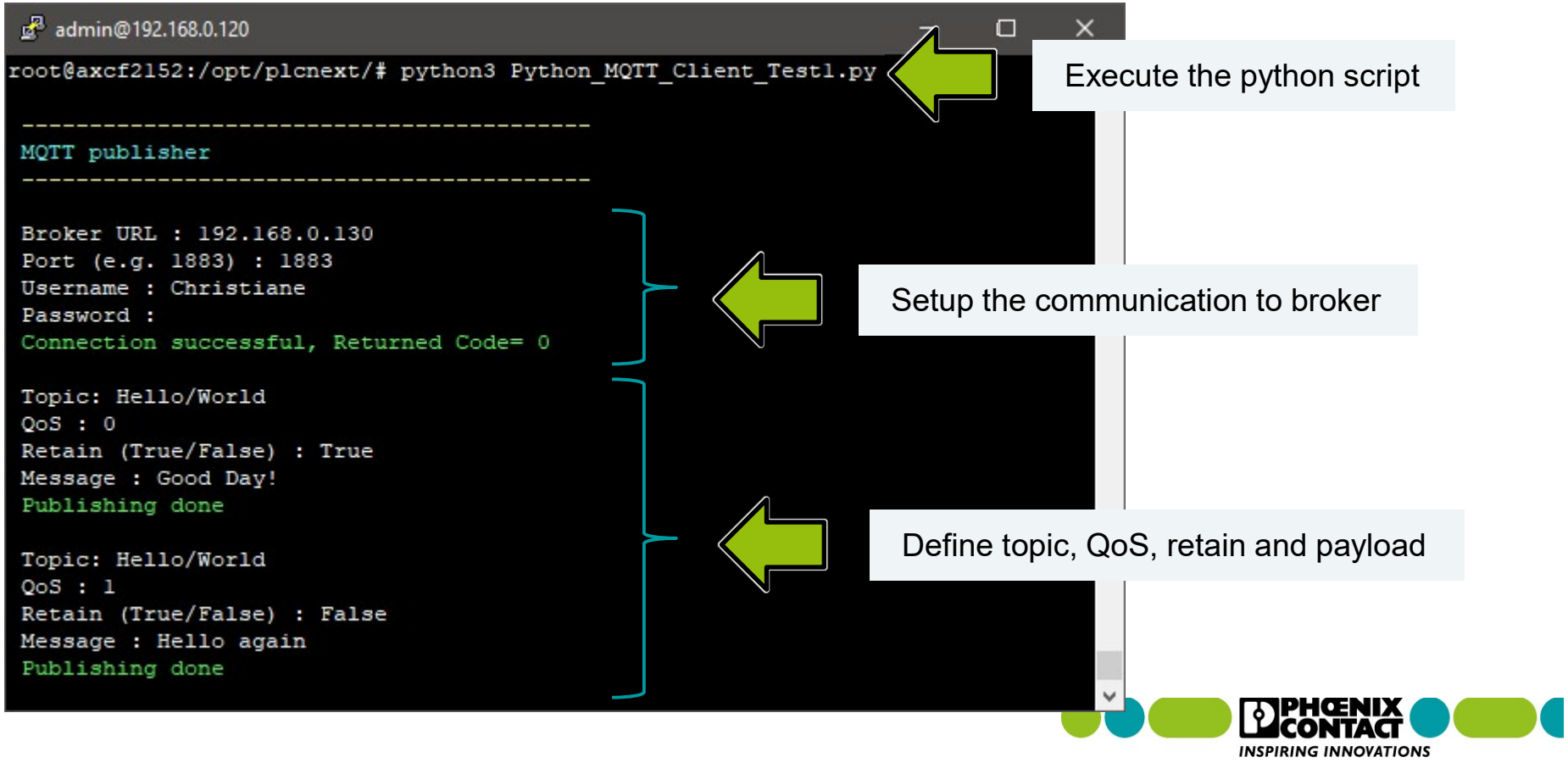
PHOENIX CONTACT
INSPIRING INNOVATIONS

MQTT examples



PHOENIX CONTACT
INSPIRING INNOVATIONS

MQTT examples



PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Agenda

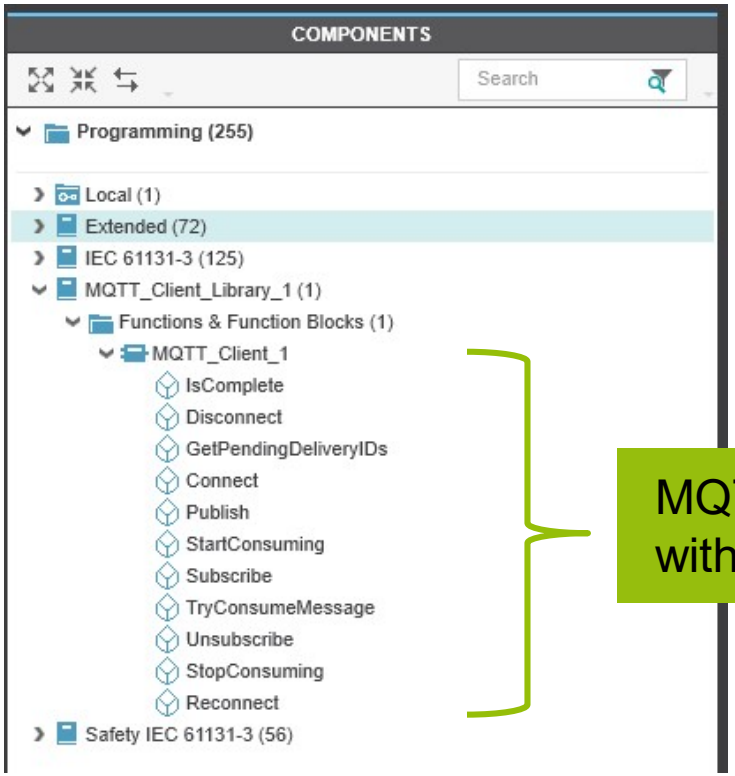
- General introduction to MQTT
- MQTT with Node-RED
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT with Python
  - Installation of the paho-mqtt library
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT in IEC61131 (preview)
- PLCnext Control as MQTT broker
- Q&A

PHOENIX CONTACT
INSPIRING INNOVATIONS

Preview

MQTT examples

Preview IEC61131 MQTT library



MQTT function block with different methods

PHOENIX CONTACT
INSPIRING INNOVATIONS

Preview

MQTT examples

Preview IEC61131 MQTT library

Demo

PHOENIX CONTACT
INSPIRING INNOVATIONS

C++ Components

Agenda

- General introduction to MQTT
- MQTT with Node-RED
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT with Python
  - Installation of the paho-mqtt library
  - Subscribe data from a public broker
  - Publish data to a public broker
- MQTT in IEC61131 (preview)
- PLCnext Control as MQTT broker
- Q&A

PHOENIX CONTACT
INSPIRING INNOVATIONS

MQTT examples

Mosquitto broker

- lightweight open source message broker
- implements MQTT versions 3.1.0, 3.1.1 and version 5.0
- free of charge for everyone (and business-friendly licensing thanks to EPL/EDL)
- Available for Linux and Windows systems

Documentation: https://mosquitto.org/documentation/
Github: https://github.com/eclipse/mosquitto

PHOENIX CONTACT
INSPIRING INNOVATIONS

Installation of the mosquitto broker via docker

1) Create a PuTTY session
2) Login as root
3) Use the following command to install *mosquitto:*

```
balena-engine run –it ––name mosquitto –p 1883:1883 eclipse-mosquitto
```

4) Restart *balenaEngine*:

```
/etc/init.d/balena stop
```

```
/etc/init.d/balena start
```

5) Start *mosquitto:*

```
balena-engine start mosquitto
```

MQTT examples

Creation of a new user

1) Open the container console: `balena-engine exec –it mosquitto /bin/sh`
2) Change to the mosquitto directory: `cd /mosquitto`
3) Create a new password file and your first user:

   `mosquitto_passwd -c passwordfile <Username>`
4) Enter the password twice (It will be stored within the file in encrypted form)

* To create further users enter: `mosquitto_passwd -b passwordfile <Username> <Password>`

Make password file known

1) Change the directory: `cd /config`
2) Open the *mosquitto.config* file in the *vi* editor: `vi mosquitto.conf`
3) Press “G” to jump to the file end (⇧ + G)
4) Add the following two lines:

```
allow_anonymous false
password_file /mosquitto/passwordfile
```

5) Safe and close the file with: Esc , then : + w + q , then Enter
6) Close the container console with: `exit`
7) Restart the mosquitto broker

MQTT examples

PLCnext Control as MQTT broker

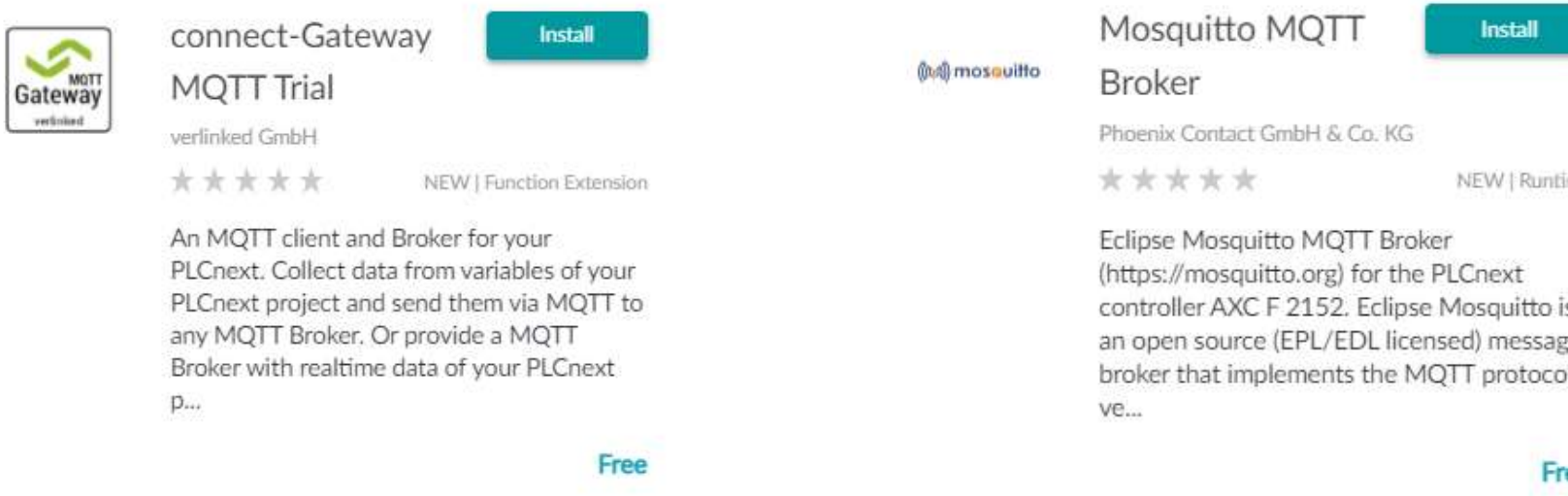
Demo

PHOENIX CONTACT
INSPIRING INNOVATIONS

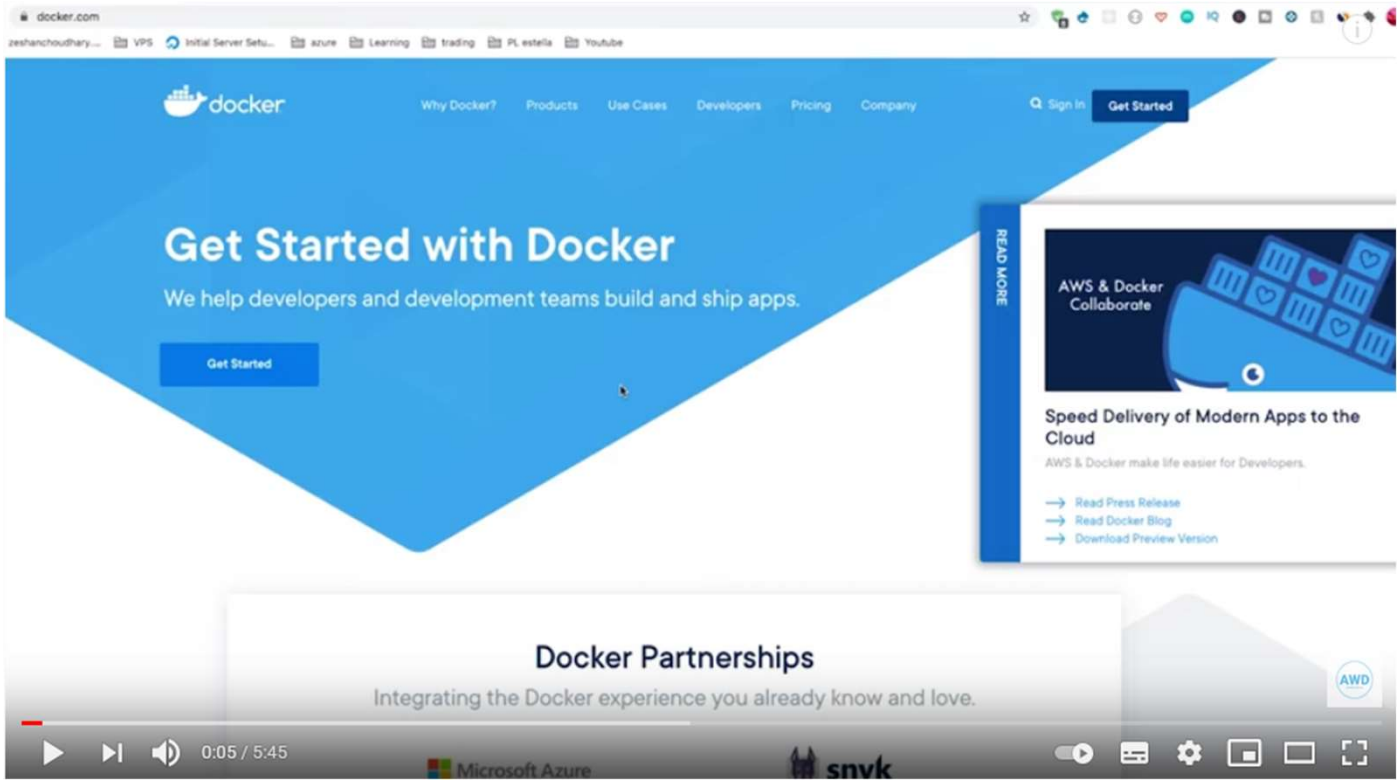
MQTT examples

Some alternatives

See: https://www.plcnextstore.com/#/



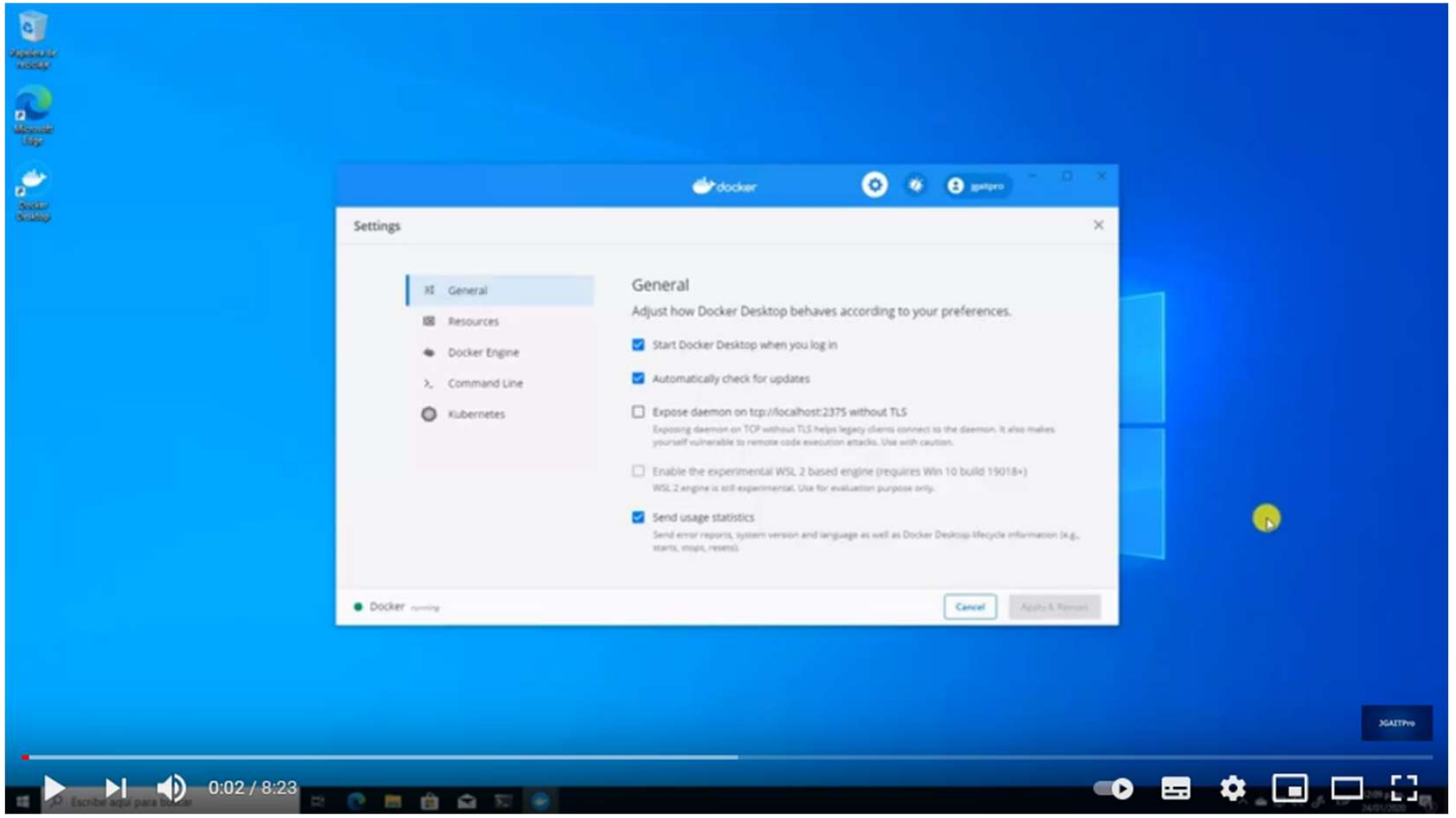
INSPIRING INNOVATIONS



Docker Docker.com Balena engine ¿Qué es?

PHOENIX CONTACT
INSPIRING INNOVATIONS

01.- ¿Que es Docker? Y ¿Realmente lo necesito? 🤔 [Tutorial en Español]



Crear contenedores Windows y Linux en Docker Desktop

PHOENIX CONTACT
INSPIRING INNOVATIONS

Selección y nivel Básico

PLCnext Engineer

Elevator Control System based on PLCnext Technology

PLCnext Technology PROJECTS …. Applications of Products

PHOENIX CONTACT
INSPIRING INNOVATIONS

Thank you

PLCnext Technology
Designed by PHOENIX CONTACT

PHOENIX CONTACT
INSPIRING INNOVATIONS