
UiA
University of Agder
Norway

Lecture #2.1

Introduction to Part II

MAS418

Programming for Intelligent Robotics and Industrial systems

Part II: PLC Software Development

Spring 2024

Daniel Hagen, PhD

UiA

Associate Professor

Daniel Hagen

- 35 years old, from **Lyngdal**, lives in **Arendal** with my wife, her 15-year-old daughter, and our four-year-old son
- **BSc** (2012) and **MSc** (2014) in Mechatronics from **UiA** based on my trade certificate as an Automation Mechanic (2009)
- Received my **PhD** with a Specialization in Mechatronics in 2020
- Between **MSc** and **PhD** I worked in Cameron Sense with PLC-based Control Systems (Siemens) for oil drilling equipment
- After the **PhD** I worked in **Red Rock AS** as a Senior R&D Engineer responsible for the PC based Control Systems (Beckhoff) for cranes
- Since March 2022 I have worked full-time at **UiA**: MAS418, MAS514, MAS516, MAS513

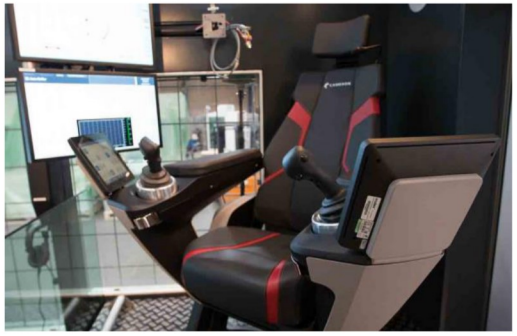
UiA

https://www.uia.no/en/kk/profile/danielh

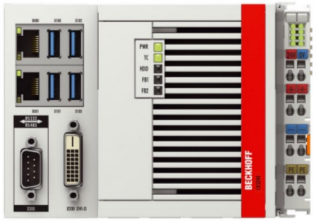
Background

Operator/Human-in-the-loop Control

Human Machine Interface (HMI)



Real-Time Controller (PLC)



Mechatronic Application

Motivation

Autonomous Control Systems

Top-level control

Real-Time Controller (PLC)

Low-level (machine) control

Mechatronic Application

UiA

MAS418 Overview

Distributed Control System

Part III – Project Work in Groups

Part I

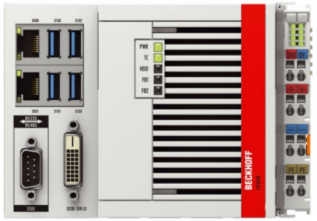
ROS

ubuntu

Top-level control – motion reference, data logging, visualization, etc.

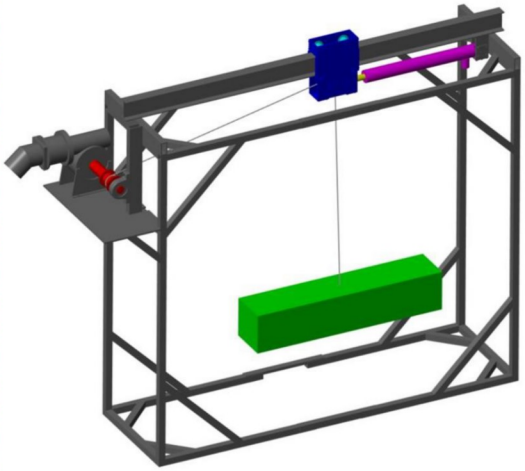
Part II

Real-Time Controller (PLC)



Low-level (machine) control – safety, open-loop (manual) and closed-loop (automatic) motion control

Mechatronic Application



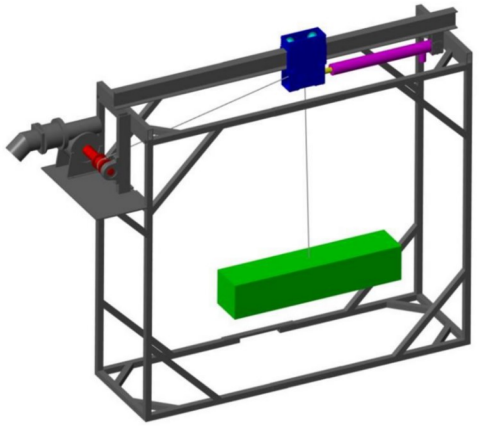
UiA

Part II Overview

- **PLC programming** using Structured Text (ST) in Beckhoff TwinCAT software
- Focus on **object-oriented programming** and **version control**
- **Programming** of a typical industrial mechatronic application (machine) control system
- **Testing** of the program in PLC simulator

Real-Time Controller (PLC)

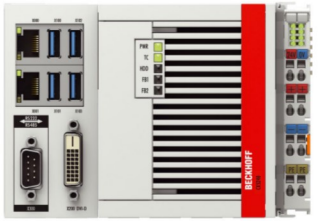
Mechatronic Application

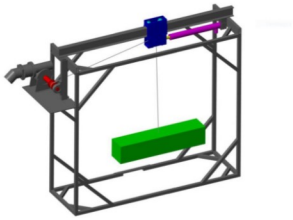


Part III - Project Overview

- **Programming** and **implementation** of a simulator representing the real machine on the PLC and ROS 2
- **Interface** with ROS2
- **Control** the simulated machine using ROS 2 (high level control) and PLC (low level control) and visualization in Rviz2

Real-Time Controller (PLC)





ROS

ubuntu

UiA

Information

Teacher - Part II

Daniel Hagen

Email: daniel.hagen@uia.no

Office: D3-037

Unofficial Discord server for support

In Part III both Daniel and Kristian are involved

Updated information and course materials available in Canvas

Students with a BSc in Mechatronics or Mechanical Engineering

2. sem	MAS409-G Elektriske motordrifter 7,5 sp	MAS414-G Maskintekniske systemer 2 7,5 sp	MAS411-G Industriell IT 7,5 sp	For students with a BSc in Mechatronics or Mechanical Engineering ● Maskintekniske systemer 1 (7,5 sp) ● Programmering for intelligent robotikk og industrielle systemer (7,5 sp)

Students with a BSc in Electronics or Electrical Engineering

2. sem	MAS409-G Elektriske motordrifter 7,5 sp	MAS414-G Maskintekniske systemer 2 7,5 sp	MAS413-G Maskintekniske systemer 1 7,5 sp	For students with a BSc in Electronics or Electrical Engineering ● Industriell IT (7,5 sp) ● Programmering for intelligent robotikk og industrielle systemer (7,5 sp)

UiA

Activities

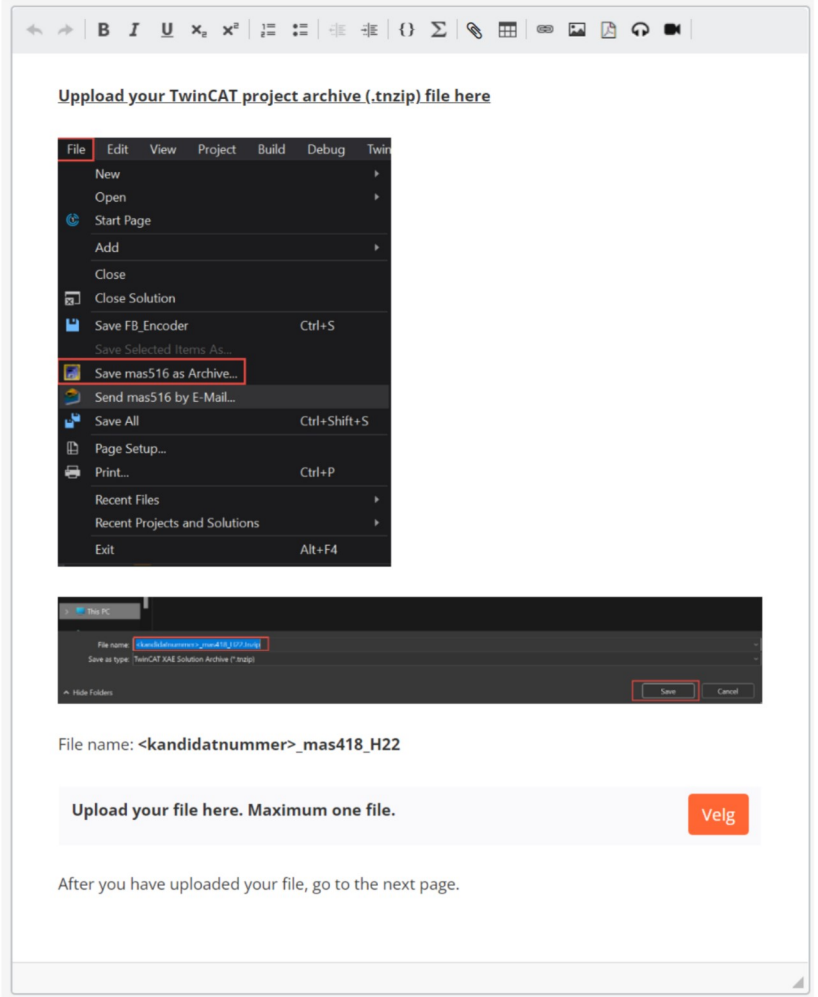
- **Lectures**
  - Four main lectures (weeks 8-11)
  - One lectures/demo related to Part III (week 12)
  - Wednesdays 10:15-12(13):00 (A2-123)
- **Individual Lab Exercises**
  - Four lab exercises (weeks 5-11)
  - Wednesdays 12(13):15-16:00 (D2-036)
- **Project work in Groups**
  - Lab work related Part III (weeks 13-17)
  - Wednesdays 10:15-16:00 (D2-036)
- **Homework**
  - Self-study before lectures
  - Complete exercises that are not finished during the lab exercise
  - 5-7 hours a week is expected

Change of the plan will be announced in Canvas!

[Course plan: MAS418-G 24V Programmering for intelligent robotikk og industrielle systemer (instructure.com)]

Activities

- **Examination Part II**
  - PLC programming task in TwinCAT on lab PCs
  - 3 hours - 35% of total grade
  - Submission of .tnzip file in Canvas assignment
  - Grading (100%):
    - Correct and sensible use of **Enumeration** (10%)
    - Correct and sensible use of **Structure** (10%)
    - Correct and sensible use of **Program** (10%)
    - Correct and sensible use of **Function** (10%)
    - Correct and sensible use of **Instruction** (CASES / IF-ELSE statements) (10%)
    - Correct and sensible use of **Function block** in an object-oriented way (30%)
      - Reuse of a **Function block** (10%)
      - Use of **Method** (10%)
      - Use of **Interface** (10%)
    - A working **simulator** demonstrating that the code works as intended (20%)



Literature

- Optional Book: **PLC Controls with Structured Text (ST), V3**
  - https://www.adlibris.com/no/sok?q=PLC+Controls+with+Structured+Text+%28ST%29%2C+V3+PLC+Controls+with+Structured+Text Links to an external site.
- **Beckhoff Information System**
  - https://infosys.beckhoff.com/index_en.htmLinks to an external site.
- **CODESYS Identifier**
  - https://help.codesys.com/api-content/2/codesys/3.5.12.0/en/_cds_identifiers/Links to an external site.
- **PLC programming using TwinCAT 3**
  - https://www.youtube.com/playlist?list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowOLinks to an external site.
- **TwinCAT 3 Tutorial**
  - http://www.contactandcoil.com/twincat-3-tutorial/ Links to an external site.
- **Learning PLCs with Structured Text**
  - https://www.youtube.com/playlist?list=PLE1CU6EebvTCJCMIUOSWgMseMaW-2k5zH

Overview

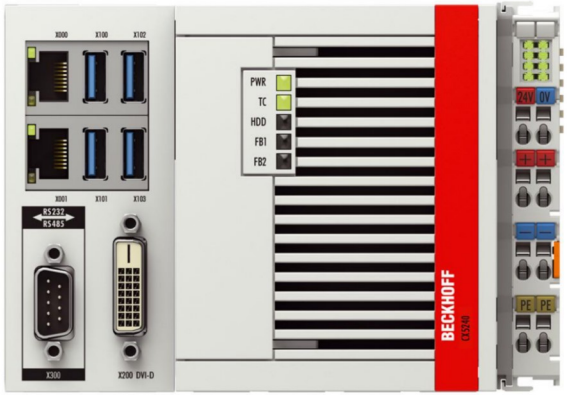
Introduction

Part I: Introduction to PLC programming

Part II: TwinCAT basics

Part III: Simulation in TwinCAT

Summary



UiA

Part I: Introduction to PLC programming

1. SCADA system
2. History
3. IEC61131-3 standard
4. Basic data types
5. Version control

SCADA system

- **Supervisory Control And Data Acquisition**

https://youtu.be/nlFM1q9QPJw?t=46

History - PLC

- Before PLCs, it was possible to automate a process using relay-based control systems
  - Requires a huge number of relays, cabling and space → difficult to maintain
  - Small change in system require rewiring
  - Single wire brake could cause the whole system to not work
- In the end of the **1950**s the first ideas for replacement of these hard-wired relay systems arrived
  - Independently the companies Allen-Bradley, GM and Siemens investigated the electronic replacement, and they created the first PLCs

https://ladderlogicworld.com/relay-logic-vs-ladder-logic/

UiA

https://www.youtube.com/watch?v=0iDn9E0V1iw

History - PLC

- In **1973**, the first recognizes mass-marked PLC to achieve success arrived (Modicon 184)
- **Siemens PLC** history:
  - **1958**: SIMATIC Version G
  - **1973**: SIMATIC S3
  - **1979**: SIMATIC S5
  - 1995: SIMATIC S7
  - **2010**: Total Integrated Automation (TIA) portal

Siemens SIMATIC S-7 1500 (**2012**)

Modicon 184 (**1973**)

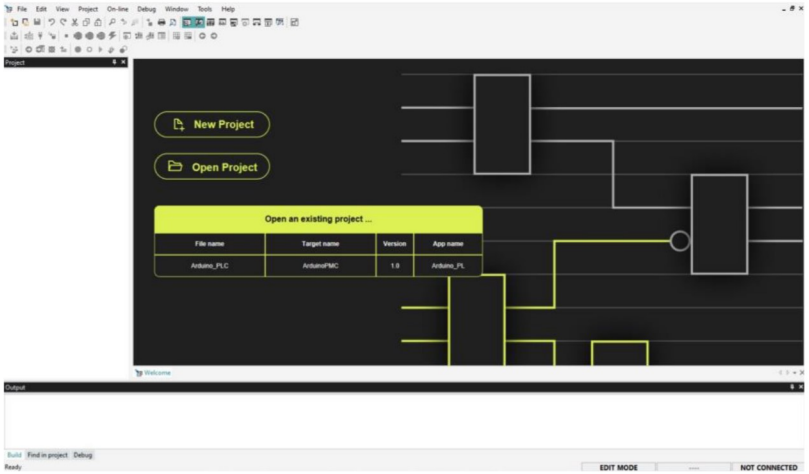
Siemens SIMATIC S-7 300 (**1995**)

https://www.c3controls.com/white-paper/history-of-programmable-logic-controllers/

History - Arduino PLC

- In **2022** Arduino released their own PLC IDE, allowing you to program the Arduino Portenta Machine Control using the 5 programming languages defined by the IEC 61131-3 standard
- In **2023** Arduino will release their own PLC hardware – Arduino Opta, a microPLC with Industrial IoT Capabilities.

Arduino PLC IDE



https://www.arduino.cc/pro/software-plc-ide

Arduino Opta

https://www.arduino.cc/pro/hardware-arduino-opta

https://www.youtube.com/watch?v=mq1xgubaQeA

History - PC-based control (TwinCAT)

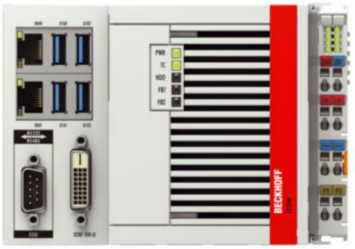
- The adoption of PCs in **1980**s was the start of interfacing PCs with PLCs
  - During **1990**s PLC monitoring software arrived for doing troubleshooting by technicians and that was the start of PC-based HMIs communication with PLC
- In **1980**s Beckhoff started with PC-based control running on standard PC in DOS
  - In **1986** the first industrial PLC was released
- In **1996** Beckhoff released The Windows Control and Automation Technology (TwinCAT) 2, where it was possible to turn a standard PC running Windows into a real-time capable controller (More details in Part 2)

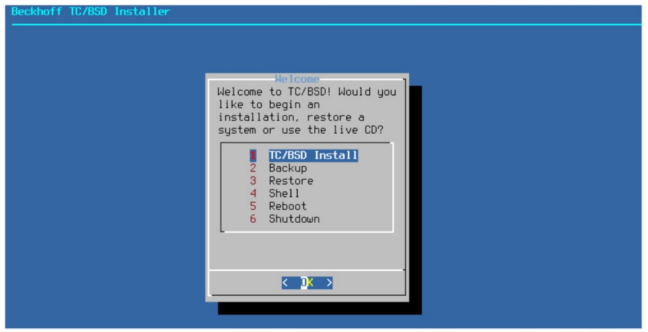
Beckhoff C-4000 (**1986**)

https://www.c3controls.com/white-paper/history-of-programmable-logic-controllers/

History - PC-based control (TwinCAT)

- In **2012** TwinCAT 3 was released where Beckhoff integrated both the development environment and the functionality of the system manager into MS Visual Studio where everything related to configuration, writing code, compiling, publishing, and doing version control could be done from this single Integrated Development Environment (IDE)
- In **2020** there was released a TwinCAT runtime for Beckhoffs FreeBSD called Tc/BSD
  - However, all PLC software development is carried out in a Windows environment (or through virtual machine in Linux/OSx)



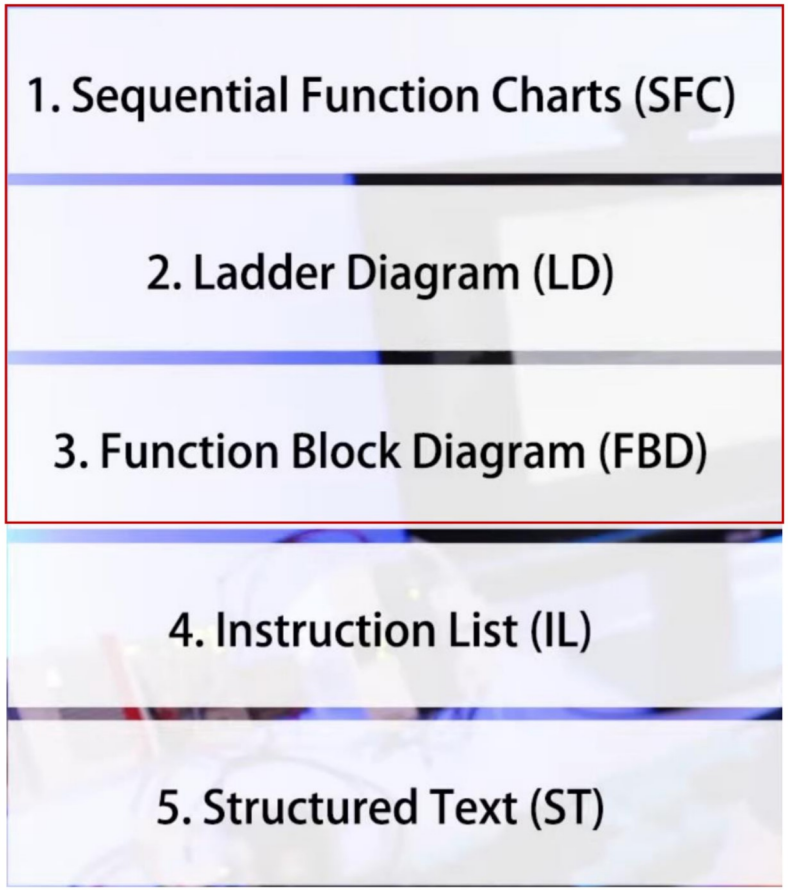


https://www.youtube.com/watch?v=0iDn9E0V1iw

IEC61131-3 standard

International standard for PLC programming

- One of the most significant milestones in PLC history was the introduction of the International Electrotechnical Commission (IEC) specification in **1982**
- It was the standard by which PLC software being developed was to be held against. It became published in **1993** as IEC 61131 International Standard for Programmable Controllers
- The introduction of the IEC 61131-3 (**2013**) was necessary as it brought consistency to all the software products on the market
  - This allows engineers and technicians to easily understand logic and program flow from any PLC software



https://en.wikipedia.org/wiki/IEC_61131-3

IEC61131-3 standard

- To be able to do PLC software in Visual Studio, Beckhoff relies on a compiler made by **CODESYS**
  - German manufacturer of IEC61131-3 automation software use by several PLC manufacturer e.g., Rexroth, WAGO, etc.
  - There is even a runtime for Raspberry Pi!
  - If you have worked with CODESYS you will find many similarities when using TwinCAT eXtended Automation Engineering (XAE) for PLC software development

https://www.codesys.com/

https://www.youtube.com/watch?v=0iDn9E0V1iw

Basic data types

- **What is a variable?**
  - Variables are used to store information to be references and manipulated in a computer program
  - They provide a way of labeling data with a descriptive name, so our programs can be understood more clearly by the developers
- Think of variables as container that hold information
  - Their purpose is to label and store data in memory
  - This data can then be used throughout your program
- When creating a variable, you need to define what data it should have from the IEC61131-3 standard

All variables are declared between:

```
1  PROGRAM TestProgram
2
3  VAR
4      bTestVariable : BOOL;
5      fTestVariable : REAL;
6      nTestVariable : INT;
7  END_VAR
```

Giving good names to variables is a task that should not be underestimated - see:
https://alltwincat.com/2019/02/11/plc-naming-conventions/

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Basic data types

- **Booleans**

TYPE	VALUES	MEMORY CONSUMPTION	EXAMPLE
BOOL	FALSE (0) TRUE (1)	8 bits (1 byte)	1 2 3 bJustAVariable : BOOL := TRUE;

1. Name
2. Datatype
3. Optional assigned (initial) value
   - If not assigned, the default value for a boolean is FALSE

UiA

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Basic data types

- **Integer**

TYPE	LOWER BOUND	UPPER BOUND	MEMORY CONSUMPTION	EXAMPLE
INT	-32768	32767	16 bits (2 bytes)	`nThisIsANumber : INT := -4232;`
UINT	0	65535	16 bits (2 bytes)	`nThisIsANumber : UINT := 44000;`
SINT	-128	127	8 bits (1 byte)	`nThisIsANumber : SINT := -3;`
USINT	0	255	8 bits (1 byte)	`nThisIsANumber : USINT := 241;`
DINT	-2147483648	2147483647	32 bits (4 bytes)	`nThisIsANumber : DINT := 1362992311;`
UDINT	0	4294967295	32 bits (4 bytes)	`nThisIsANumber : UDINT := 3453331141;`
LINT	-9223372036854775808	9223372036854775807	64 bits (8 bytes)	`nThisIsANumber : LINT := -2233720368547758O7;`
ULINT	0	18446744073709551615	64 bits (8 bytes)	`nThisIsANumber : ULINT := 13246311073709551615;`

UiA

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Basic data types

Base Converter:
https://www.rapidtables.com/convert/number/base-converter.html

- **Integer**

TYPE	LOWER BOUND	UPPER BOUND	MEMORY CONSUMPTION	EXAMPLE
BYTE	0	255	8 bits (1 byte)	`nSomeVariable : BYTE := 2#1011_0011;`
WORD	0	65535	16 bits (2 bytes)	`nSomeVariable : WORD := 16#02AE;`
DWORD	0	4294967295	32 bits (4 bytes)	`nSomeVariable : DWORD := 16#02AE_FFFF;`
LWORD	0	18446744073709551615	64 bits (8 bytes)	`nSomeVariable : LWORD := 10#18446744073709551615;`

```
nSomeVariable : BYTE := 2#1011_0011;        nSomeVariable.3;
```

UiA

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Basic data types

- **Floating point**

TYPE	LOWER BOUND	UPPER BOUND	MEMORY CONSUMPTION	EXAMPLE
REAL	-3.4×10^{38}	3.4×10^{38}	32 bits (4 bytes)	`fPi : REAL := 3.14159265359;`
LREAL	$-1.7976931348623158^{308}$	1.7976931348623158^{308}	64 bits (8 bytes)	

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Basic data types

- **String**

TYPE	MEMORY CONSUMPTION	EXAMPLE	
STRING	80 (+1) bytes	`sHelloWorldString : STRING := 'Hello World!';`	**ASCII-coded**
STRING(300)	300 (+1) bytes	`sThisIsAStringVariable : STRING(300) := 'This is a string';`	**ASCII-coded**

UiA

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Basic data types

ASCII-character Table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Basic data types

- **String**

TYPE	MEMORY CONSUMPTION	EXAMPLE	
STRING	80 (+1) bytes	`sHelloWorldString : STRING := 'Hello World!';`	ASCII-coded
STRING(300)	300 (+1) bytes	`sThisIsAStringVariable : STRING(300) := 'This is a string';`	ASCII-coded
WSTRING	80 (+1) words 160 (+2) bytes	`wsHelloWorld : WSTRING := "Hello World!";`	UNICODE-coded
WSTRING(300)	300 (+1) words 600 (+2) bytes	`wsJustAVariable : WSTRING(300) := "ĠHäöæå";`	UNICODE-coded

Å Æ Ø

UiA

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Basic data types - Type conversion

USINT

1 byte

USINT UPPER BOUND
255

UINT

2 bytes

UINT UPPER BOUND
65535

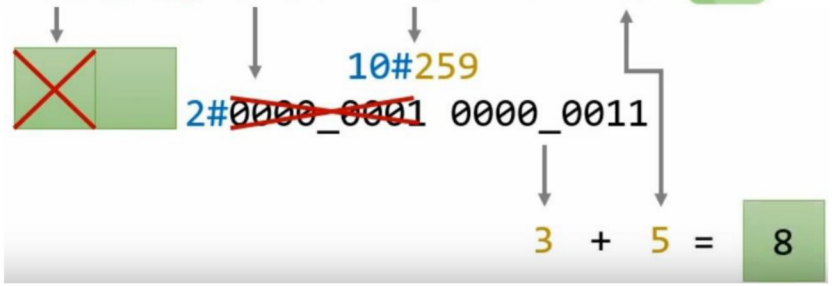
```
VAR
    nOne_Usint : USINT;
    nOne_Uint : UINT := 259;
END_VAR
```

```
nOne_Uint := nOne_Usint + 5;
```

```
nOne_Usint := nOne_Uint + 5;
```

Cannot convert type 'UINT' to type 'USINT'

```
nOne_Usint := UINT_TO_USINT(nOne_Uint) + 5;
```



https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=5

Basic data types

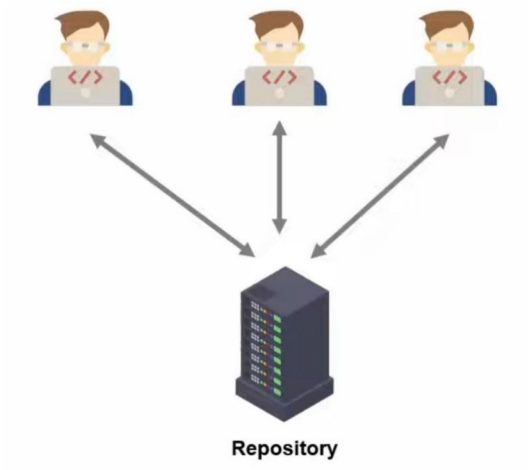
- There are a few other data types available in the IEC standard and TwinCAT3, but these together with the ones in the next lecture are the most common ones
  - Se Beckhoff documentation: https://infosys.beckhoff.com/english.php?content=..%2Fcontent%2F1033%2Ftc3_plc_intro%2F2529388939.html&id
- In TwinCAT3, and in PLCs in general, all type declaration is done at compile time
  - You have to declare the data types of your variable before you use them, and the type cannot be changed during the execution of the program
  - Implicitly trying to convert from one type to another will generally result in a compiler warning or error
  - In PLCs you generally allocate memory upfront
  - Dynamic memory allocation the way it can be done in for example C++ can be done in TwinCAT, although it is generally avoided

UiA

https://www.youtube.com/watch?v=JSFvSu8uB9U&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=4

Version control

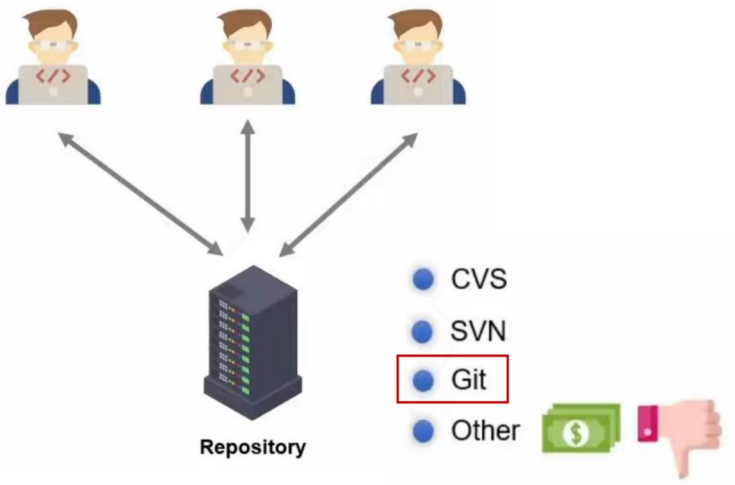
- **Why do version control?**
  - Long-term history for every file
  - Branching and merging
  - Traceability



https://www.youtube.com/watch?v=1g6eYnlzKtA&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=15

Version control

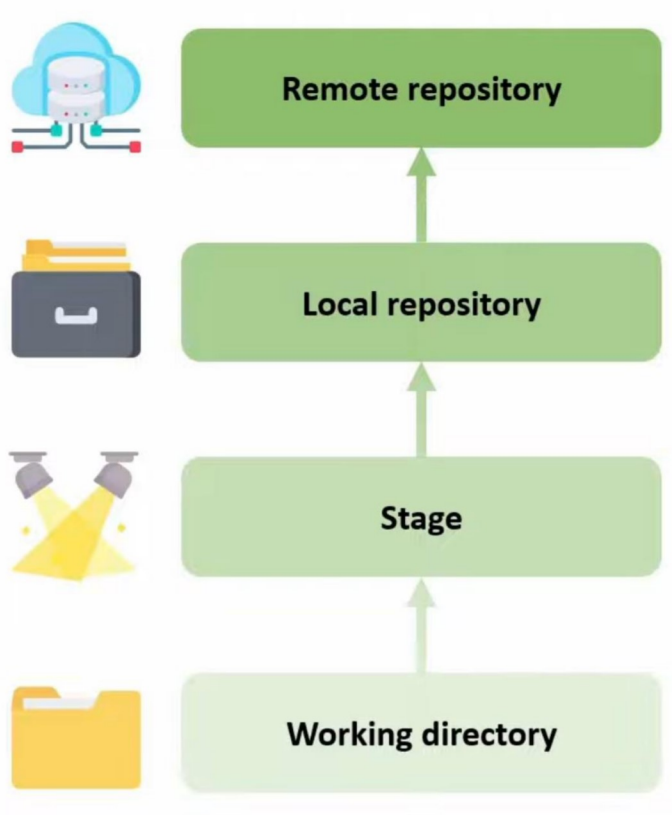
- **Git**
  - The worlds most popular version control system
  - A Distributed version control system and differs from traditional centralized version control system (SVN)
  - Uses multiple repositories including a centralized repository and server, as well as some local repositories
  - Does not rely on a central server
    - Can work locally (offline)
  - The central server can be self hosted or using third-party hosting service such as GitHub, GitLab, etc.



https://www.youtube.com/watch?v=1g6eYnlzKtA&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=15

Version control

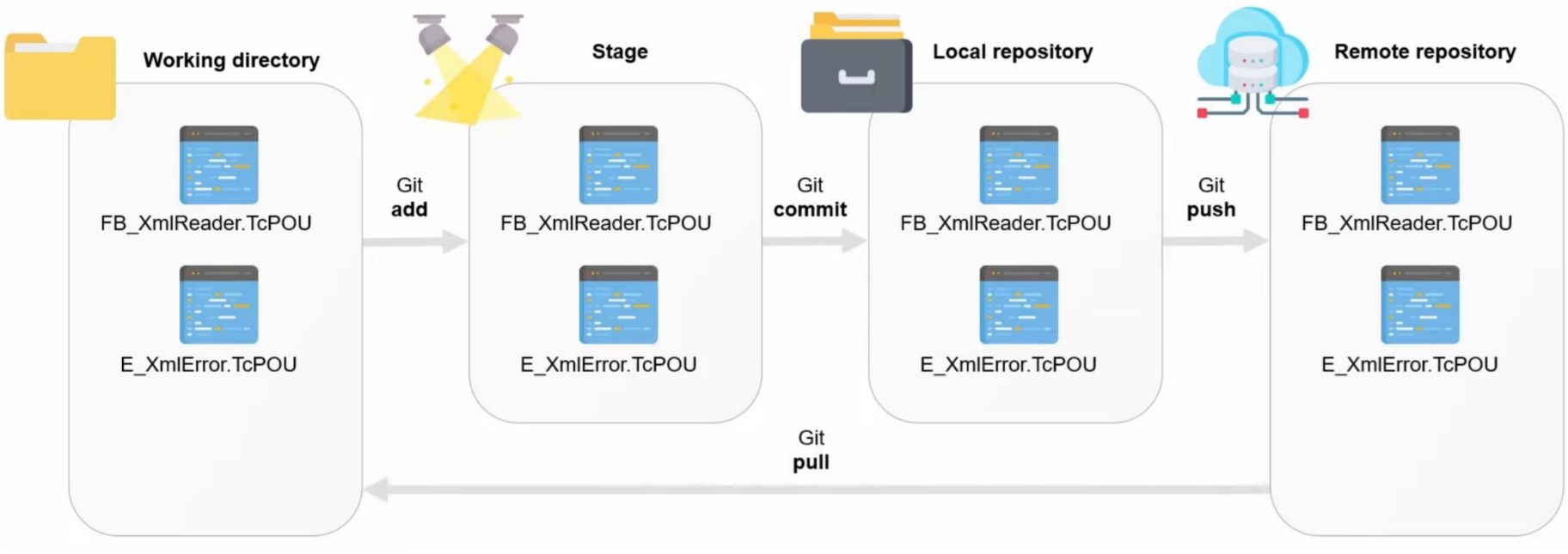
- **Git**
  - The main components



https://www.youtube.com/watch?v=1g6eYnlzKtA&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=15

Version control

- **Git**
  - Workflow



https://www.youtube.com/watch?v=1g6eYnlzKtA&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=15

Version control

- **Git**
  - Git ignore (.gitignore)
    - TwinCAT - https://github.com/github/gitignore/blob/main/TwinCAT3.gitignore
    - VisualStudio - https://github.com/github/gitignore/blob/main/VisualStudio.gitignore

https://www.youtube.com/watch?v=1g6eYnlzKtA&list=PLimaF0nZKYHz3I3kFP4myaAYjmYk1SowO&index=15

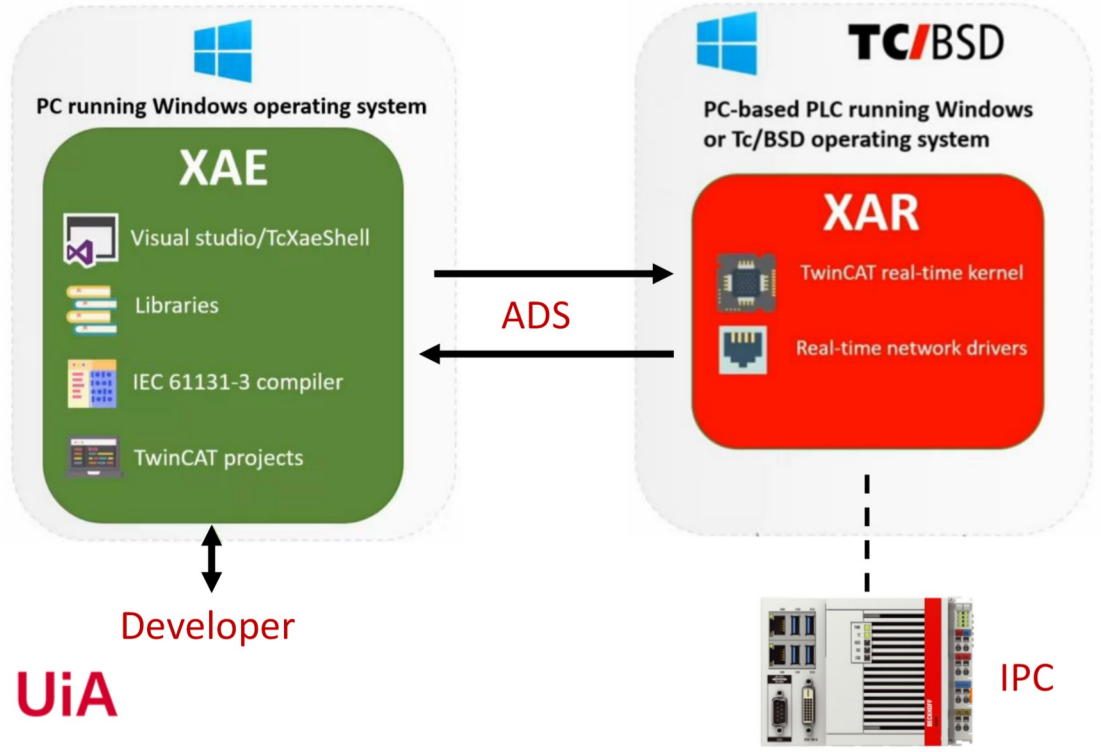
Part II: TwinCAT basics

1. Introduction to TwinCAT
2. Difference to SW running in OS (self-study)

Introcution to TwinCAT

Consist of two main parts:

- TwinCAT eXtended Automation Engineering (XAE)
- TwinCAT eXtended Automation Runtime (XAR)



XAE:

- IDE based on Visual Studio
- Libraries for various functionalities
- The IEC61131-3 compiler itself
- Used to create, edit, and compile your PLC (TwinCAT 3) projects
- Always installed on a Windows machine

XAR:

- TwinCAT real-time kernel which provides the deterministic properties of the PLC SW
- Drivers for industrial fieldbus protocols such as EtherCAT
- Can be installed on both Windows and Beckhoff's own Tc/BSD

Communication protocol:

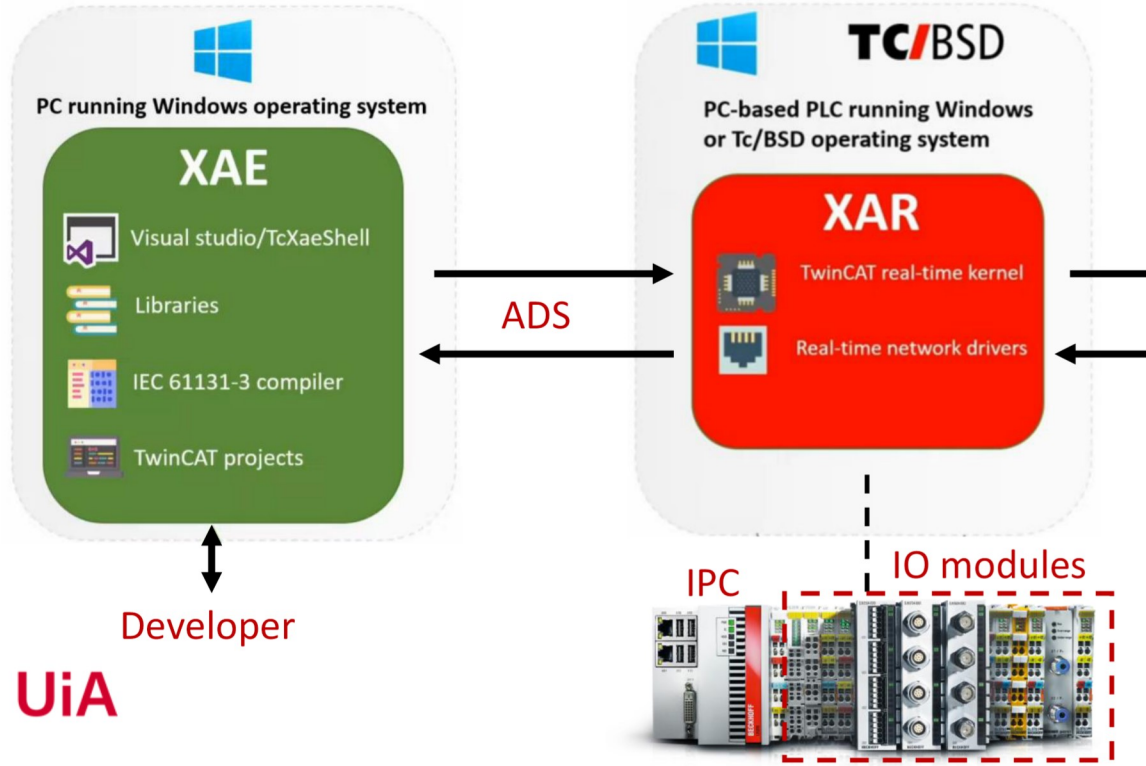
- Automation Device Specification (ADS)
- Upload new SW
- Read & Write variables (e.g., Visualizer and Measurement)

https://www.youtube.com/watch?v=0iDn9E0V1iw

Introcution to TwinCAT

Consist of two main parts:

- TwinCAT eXtended Automation Engineering (XAE)
- TwinCAT eXtended Automation Runtime (XAR)



Fieldbus interfaces:

- Communication with various sensors and actuators
- Remote Input/Output (RIO)

BACnet CANopen CC-Link DALI DMX DeviceNet EIB/KNX EtherCAT EtherNet/IP IEEE 1588 IO-Link LON M-Bus MP-BUS Modbus PROFINET RS232 RS485 sercos SMI

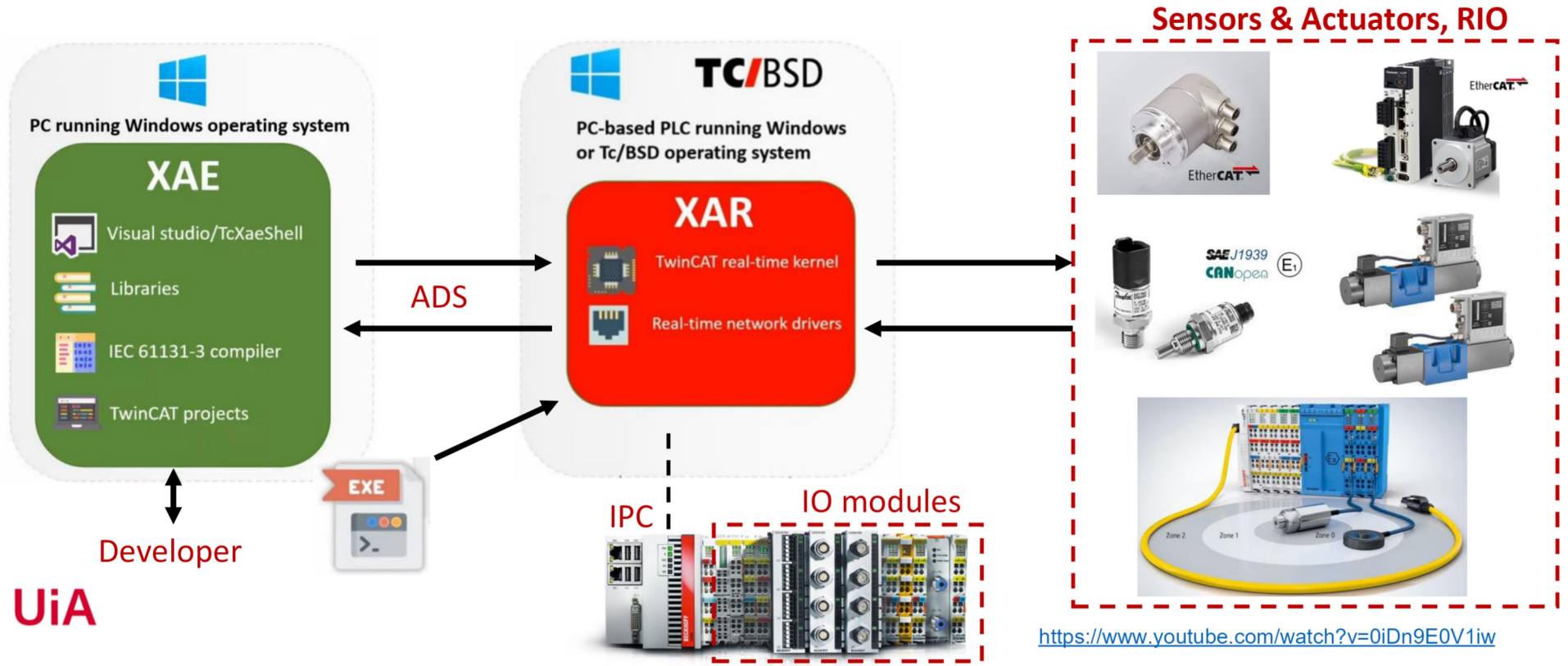
https://www.beckhoff.com/en-en/products/i-o/overview-fieldbus-systems/

https://www.youtube.com/watch?v=0iDn9E0V1iw

Introcution to TwinCAT

Consist of two main parts:

- TwinCAT eXtended Automation Engineering (XAE)
- TwinCAT eXtended Automation Runtime (XAR)



Compilation:

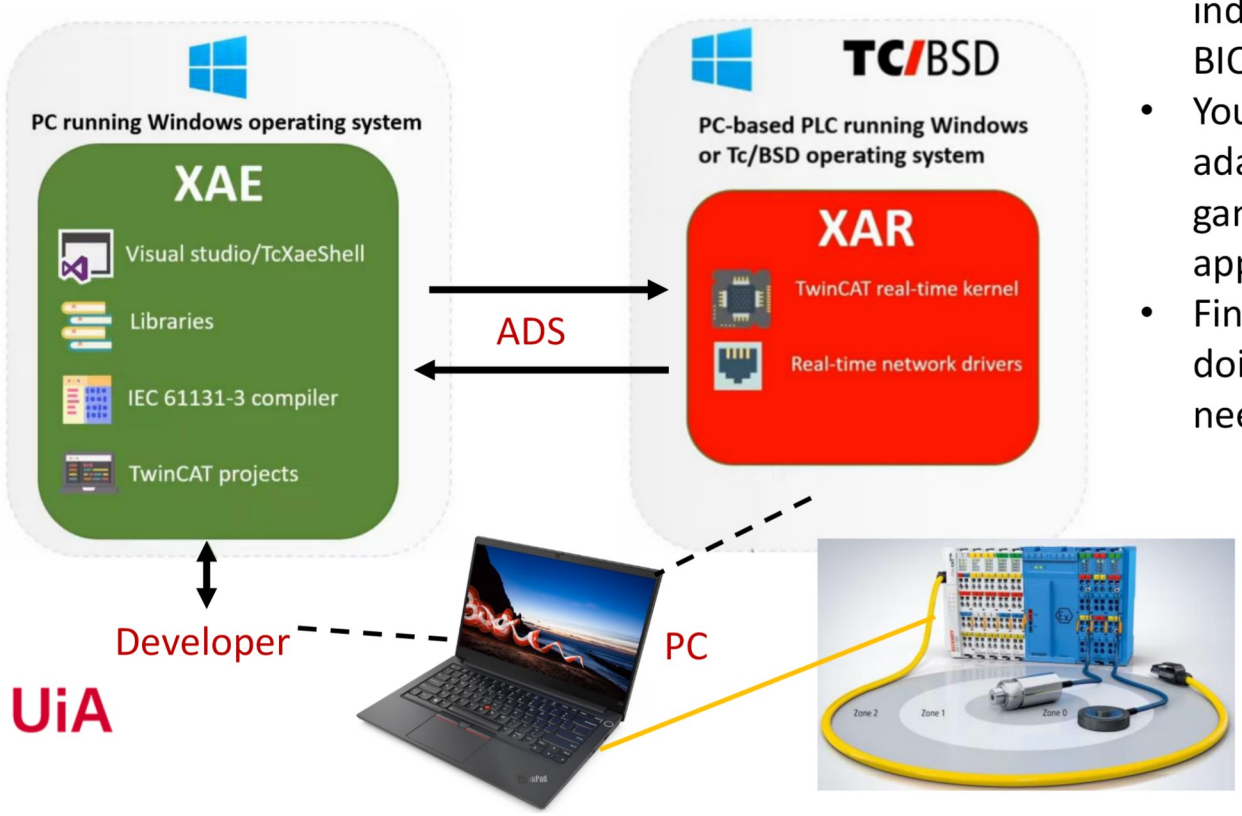
- Executable binary is created, the program can be transferred to the XAR and executed so it is running in the real-time environment of TwinCAT 3 on a PC HW (e.g. Beckhoff IPC)

https://www.youtube.com/watch?v=0iDn9E0V1iw

Introcution to TwinCAT

Consist of two main parts:

- TwinCAT eXtended Automation Engineering (XAE)
- TwinCAT eXtended Automation Runtime (XAR)



Installation:

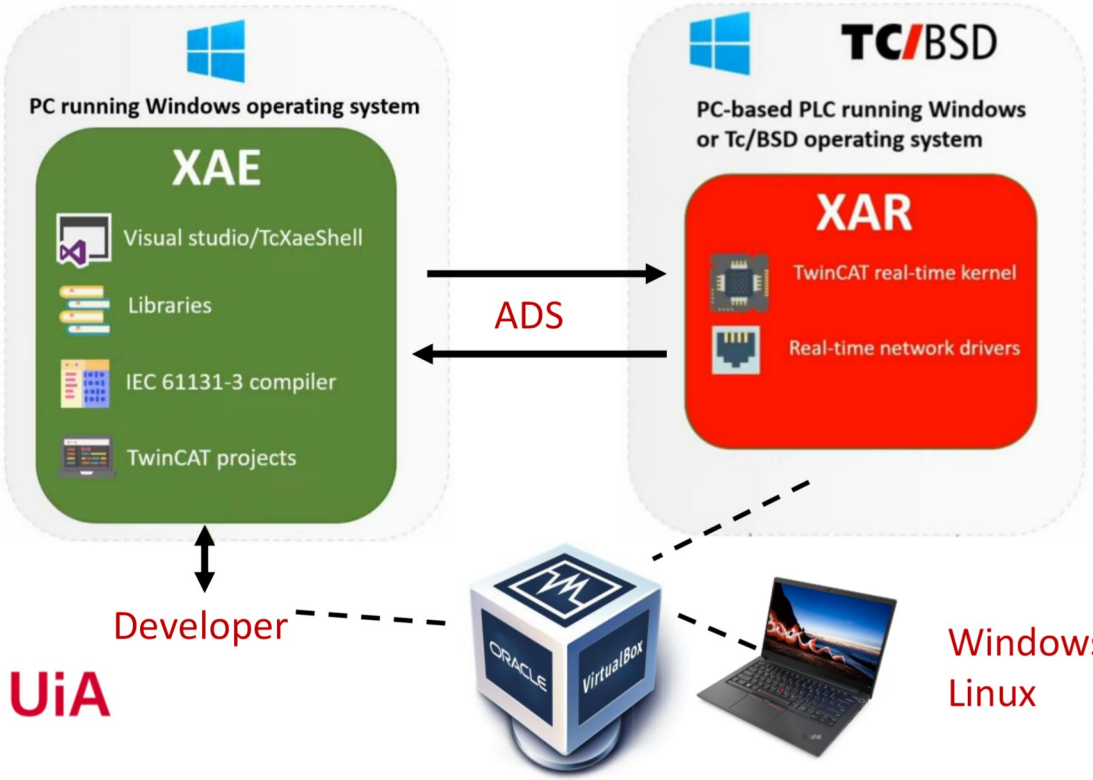
- You can choose to install either **XAE** or **XAR**
- But if you install **XAE** you automatically get **XAR** and can run your TwinCAT software locally on your development PC
- So, you don't need a separate PLC/IPC to try your SW! ☺
- Beckhoff "PLCs" are just ordinary PCs with industrial HW (24v supply etc.) and optimized BIOS-settings for real-time behavior
- You can even connect a RIO to your Ethernet adapter if you have a proper Intel chipset (not gaming PC → Realtek) and control real applications with sufficient real-time behavior
- Finally, you don't need to pay for license for doing software development. But the trial license need to be updated once every week.

https://www.youtube.com/watch?v=0iDn9E0V1iw

Introcution to TwinCAT

Consist of two main parts:

- TwinCAT eXtended Automation Engineering (XAE)
- TwinCAT eXtended Automation Runtime (XAR)



TwinCAT integration:

- You can choose if you want to use your own PC and directly install the **XAE** or use the lab PC's
- You can also use virtual machine (VirtualBox). For the fist four LAB exercises we don't care about low jitter and real-time behavior
- In the project you will work in groups using real Beckhoff IPSc (CX2033) running TC/BSD

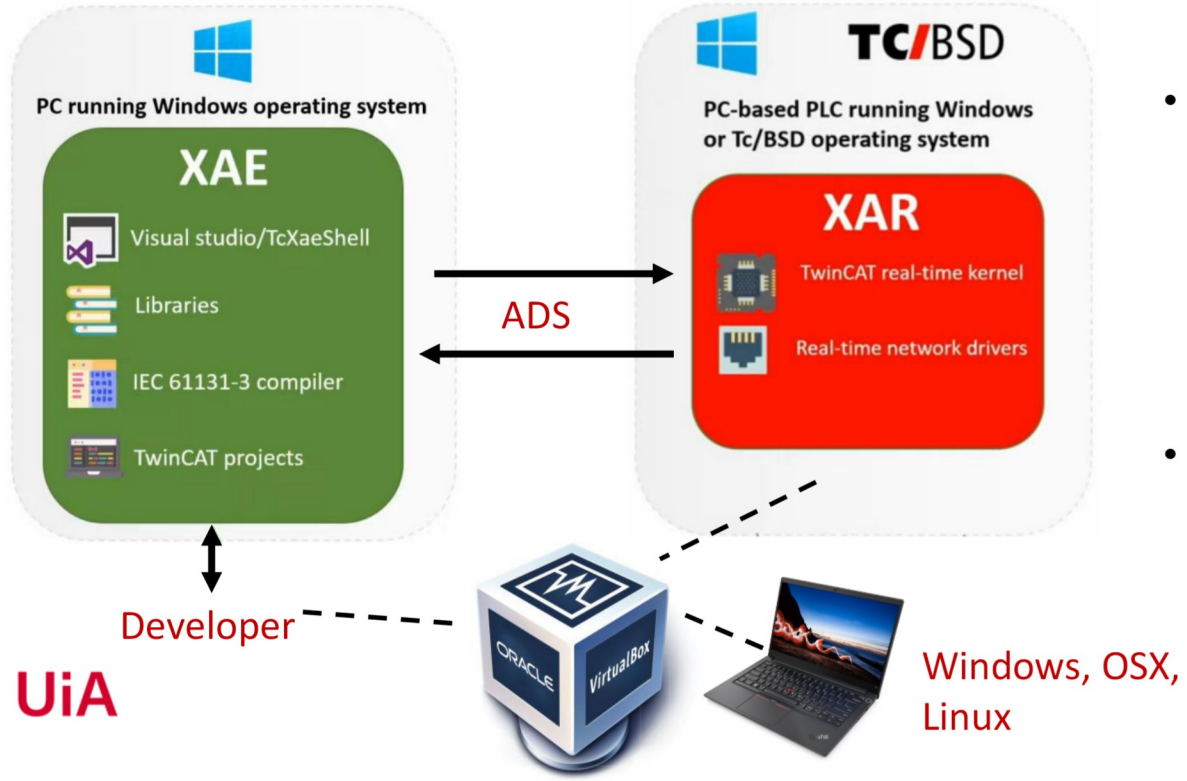
Technical data	CX2033
Processor	AMD Ryzen™ V1202B 2.3 GHz
Number of cores	2
Flash memory	slot for CFast card
Main memory	8 GB DDR4 RAM (expandable ex factory to 16 GB)

https://www.youtube.com/watch?v=0iDn9E0V1iw

Introcution to TwinCAT

Consist of two main parts:

- TwinCAT eXtended Automation Engineering (XAE)
- TwinCAT eXtended Automation Runtime (XAR)



XAE integration:

- When you install TwinCAT 3 **XAE** you get the TwinCAT XAE Shell development environment, which is based on Visual Studio (**VS**)

- The TcXaeShell is basically a version of **VS** where the possibility for all other language such as C#/C++ have been removed, but all the other functionality of VS is left intact
  - Build command, search and replace function, integrated version control with Git, etc.
- However, if you are already using VS (2013, 2015, 2017, or 2019) you can integrate the **XAE** installation with **VS**. But I will recommend all to use the Shell in this course.

https://www.youtube.com/watch?v=0iDn9E0V1iw

Difference to SW running in OS

- Difference between traditional “IT” software running as a process in an operating system compared to writing software that runs in TwinCAT 3

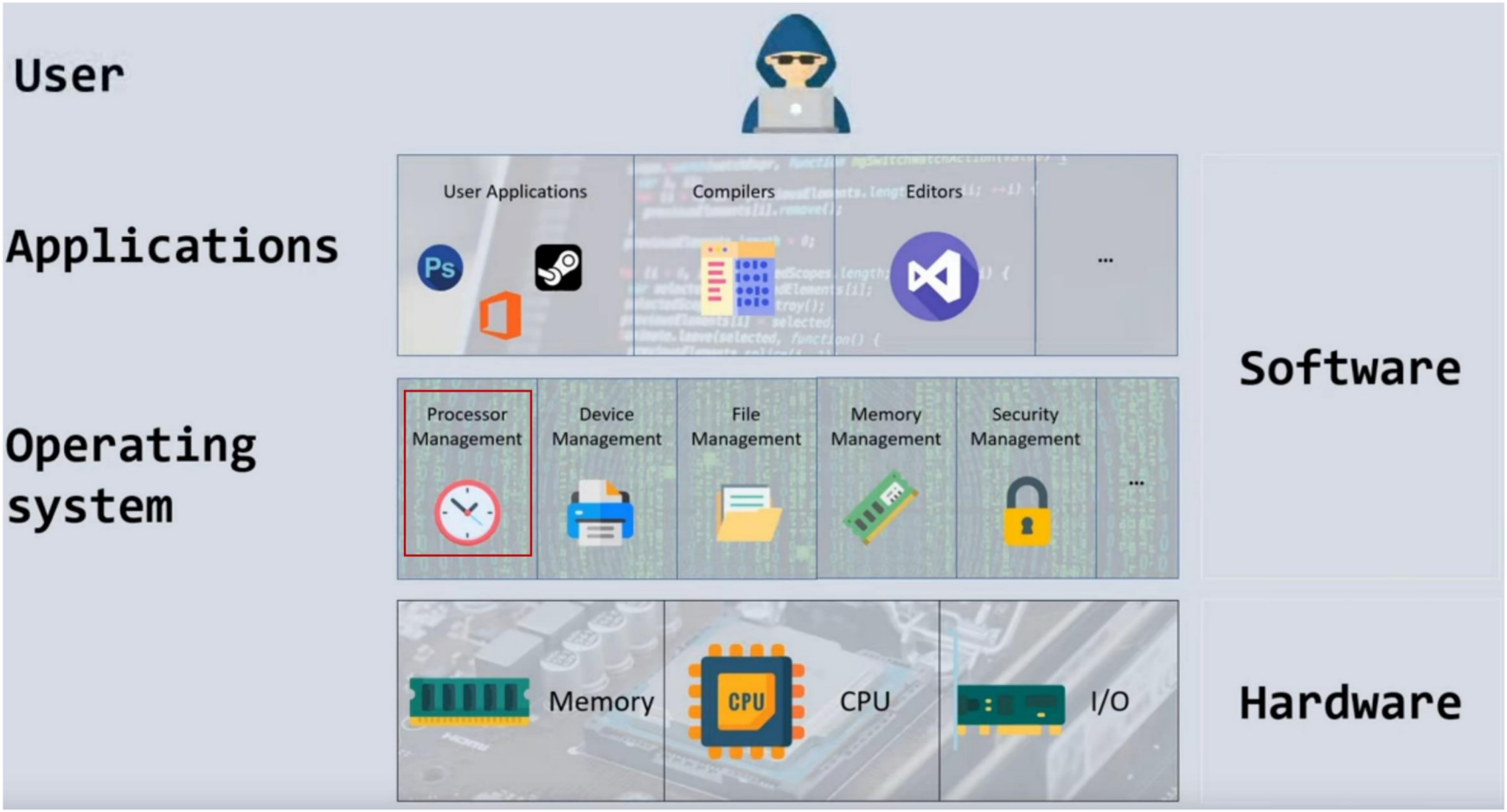
- **Self-study:**
  - **PLC programming using TwinCAT 3 - Tasks, programs & “Hello world” (Part 3/18)**

IT software

Automation software

Difference to SW running in OS

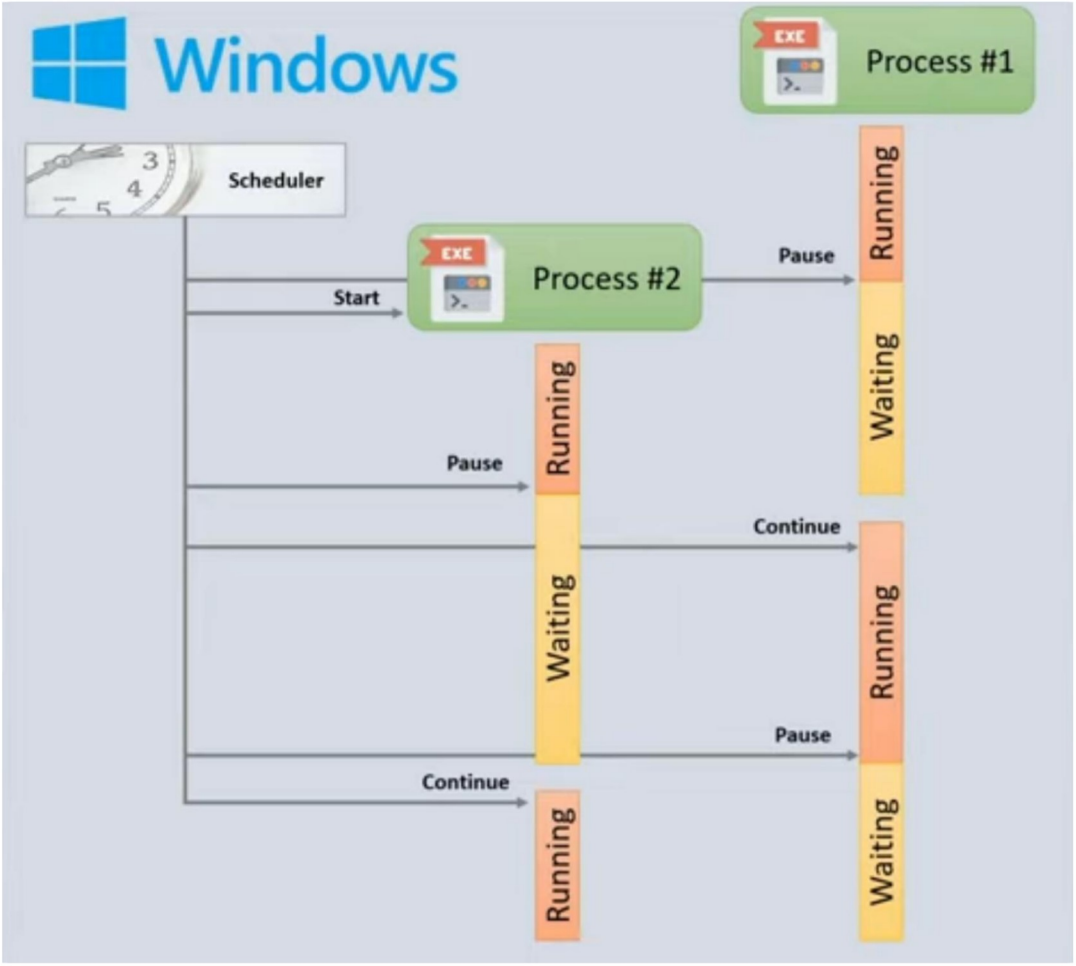
- Basics



https://www.youtube.com/watch?v=P9uUgT8EhUM

Difference to SW running in OS

- Processor management (Windows)



Scheduling policies

- First in, first out (FIFO)
- Shortest job first
- Round-robin
- Fixed priority pre-emtive scheduling
- Manual scheduling
- Priority scheduling
- …

Windows 10

- Multilevel feedback queue

https://www.youtube.com/watch?v=P9uUgT8EhUM

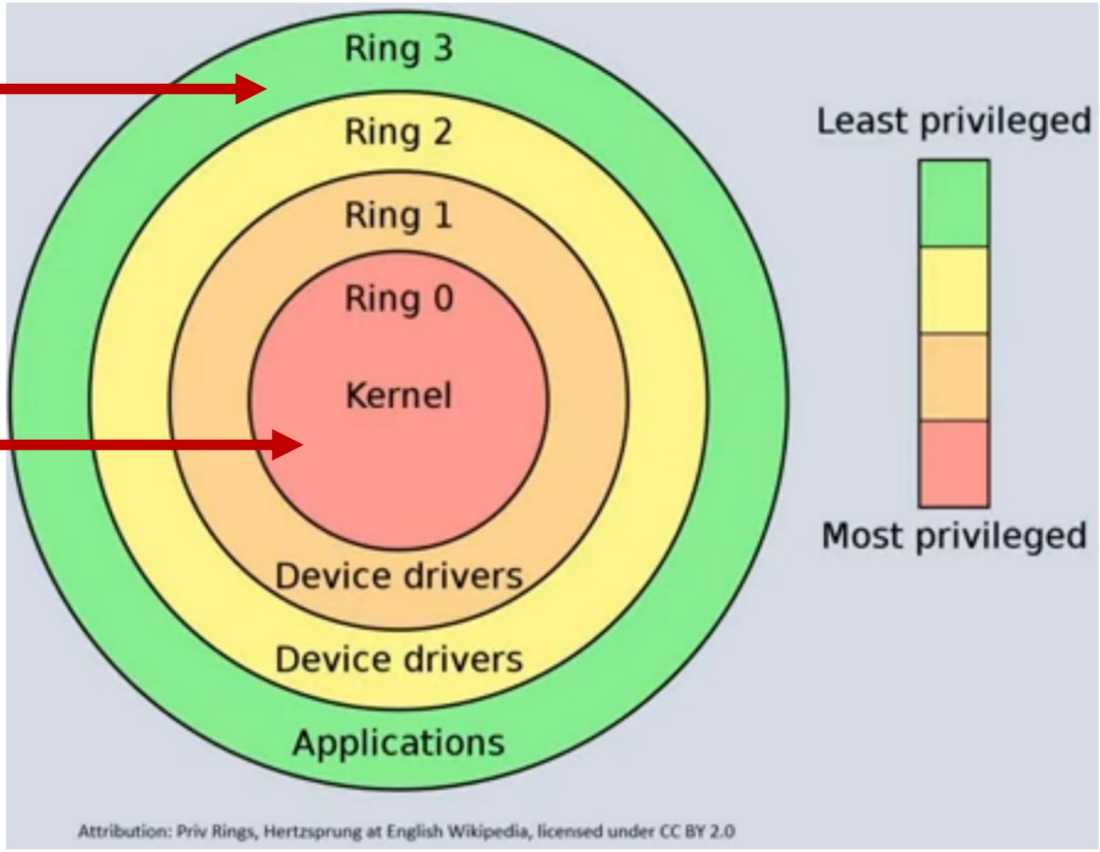
Difference to SW running in OS

Windows applications

Windows networking
Applications (processes)
TwinCAT HMI
…

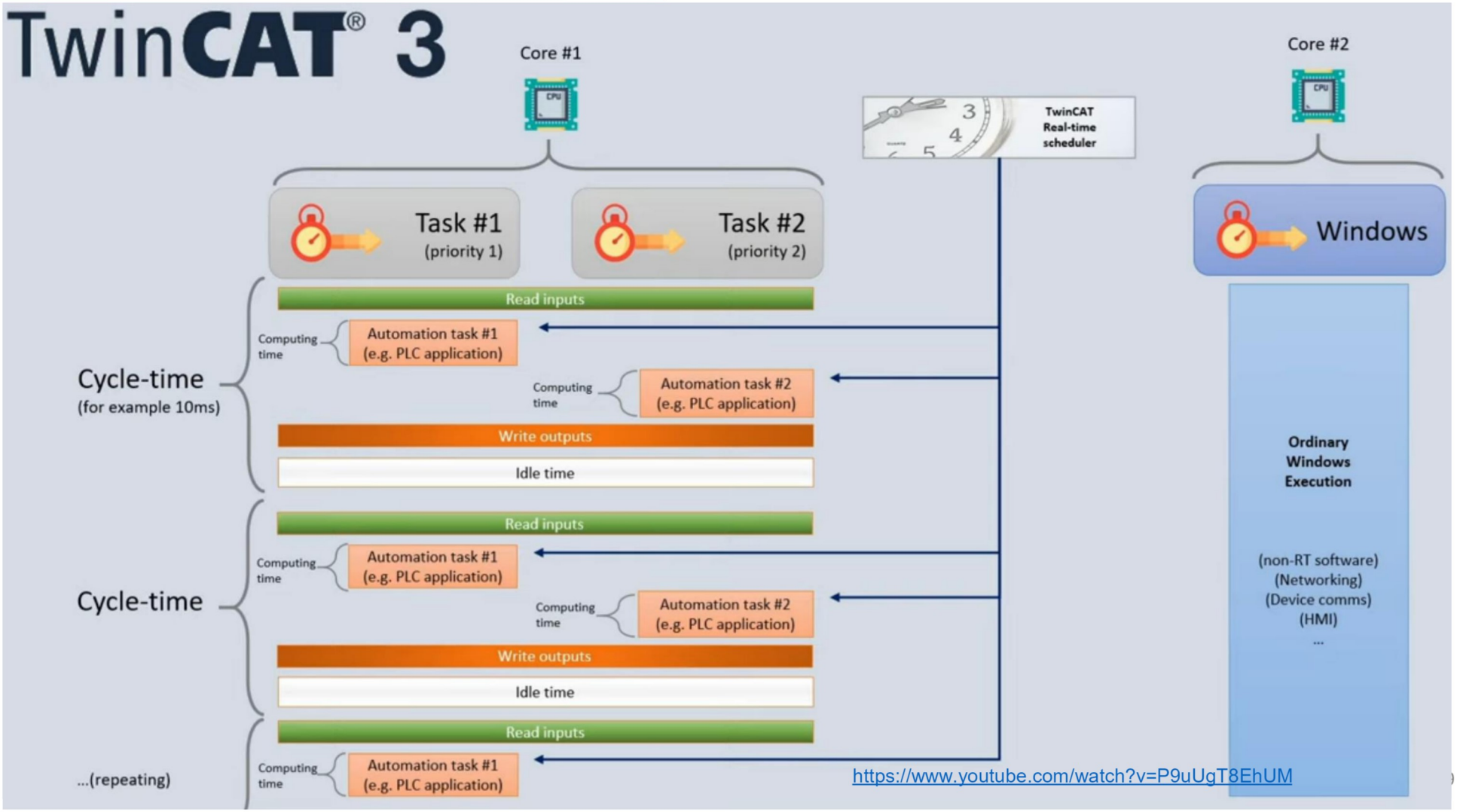
Windows kernel & TwinCAT runs here

TwinCAT scheduler
I/O, fieldbuses (EtherCAT…)
Motion Control
…



https://www.youtube.com/watch?v=P9uUgT8EhUM

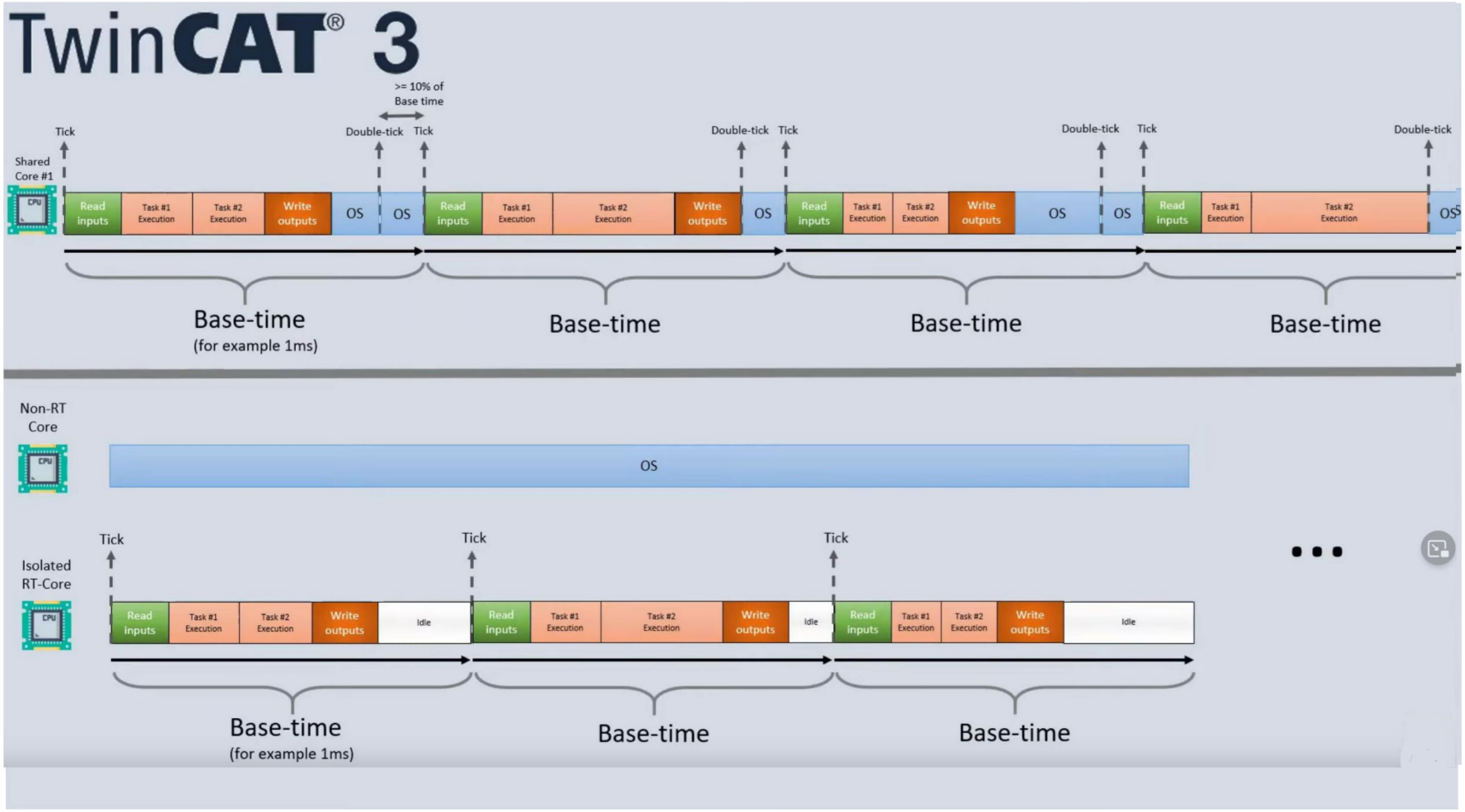
Difference to SW running in OS



https://www.youtube.com/watch?v=P9uUgT8EhUM

UiA

Difference to SW running in OS

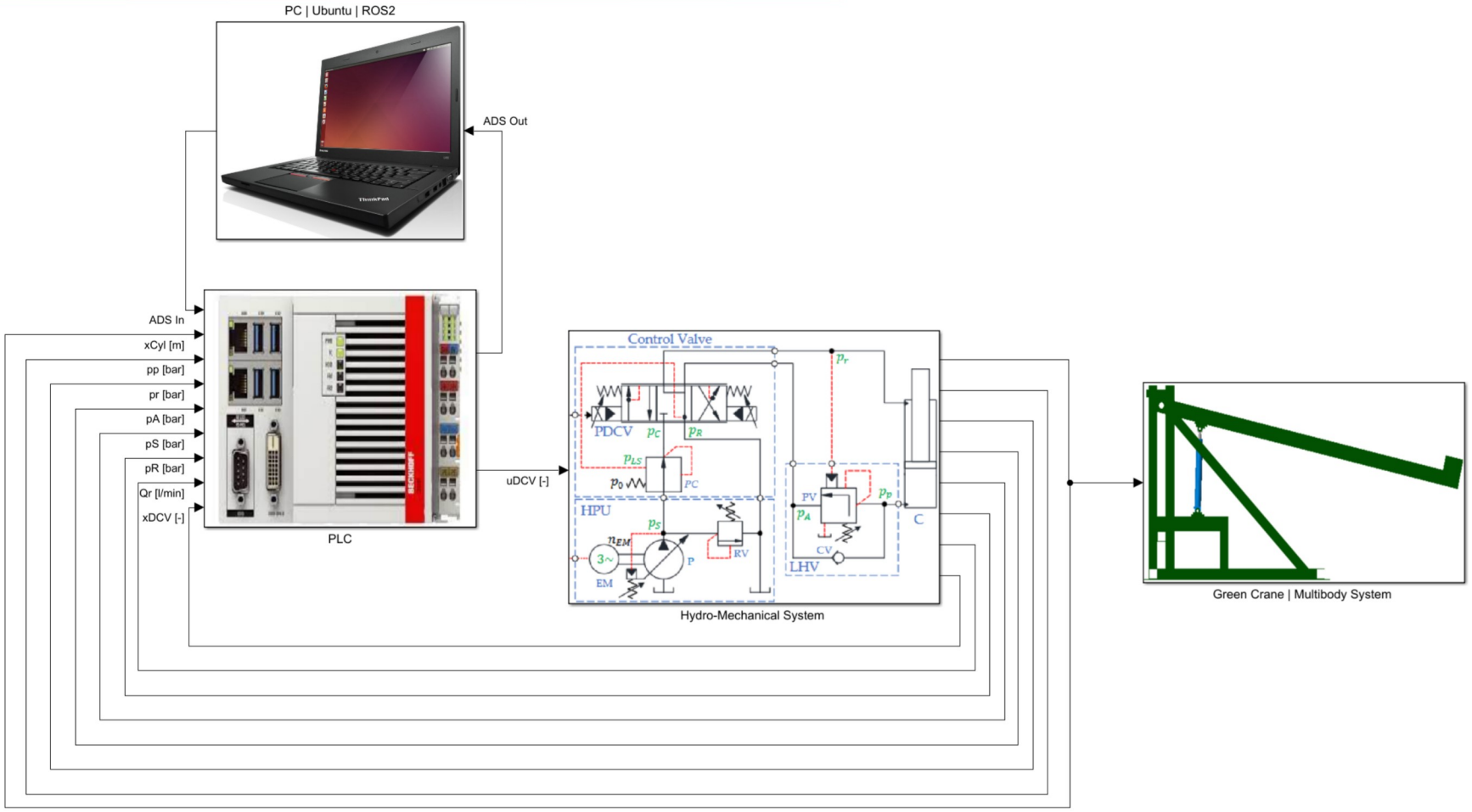


https://www.youtube.com/watch?v=P9uUgT8EhUM

Part III: Simulation in TwinCAT

1. Simulink Example of Green Crane
2. Green Crane simulator example
3. Simplified hydro-mechanical model
4. Time-domain simulation
5. Create new task in TwinCAT
6. Simulink PLC Coder

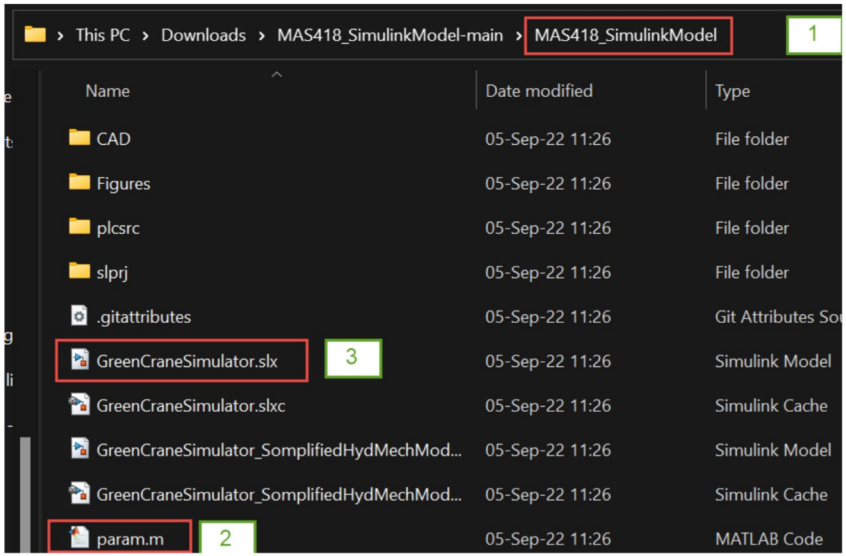
Simulink Example of Green Crane



See Simulink model for details: GreenCraneSimulator.slx - https://github.com/DrDanielh/MAS418_SimulinkModel

Simulink Example of Green Crane

- The model includes the motion ref generator, closed loop control system, hydraulics, and multi-body mechanical system model for the green crane arm.
- In the **PC | Ubuntu | ROS2** block you can adjust the settings for the control system, monitoring signals and visualization based on position feedback.
- The motion ref generator you will find under the PLC block. The controller is missing anti-dead-band-compensator.
- You can clone or download the model from Github: https://github.com/DrDanielh/MAS418_SimulinkModel
- Before running the Simulink model you must do the following:
  - Make sure that the folder name is MAS418_SimulinkModel (sometimes -main is added, but that will cause problems since we are linking to folder with CAD files and pictures in the model).
  - Open and Run the param.m file first to add parameters to the workspace and to define the correct path (the model uses CAD files and pictures located in the folder).
  - Start the GreenCraneSimulator.slx model or the simplified model: GreenCraneSimulator_SomplifiedHydMechModel.slx
  - The other .slx models are examples of using PLC_coder etc.



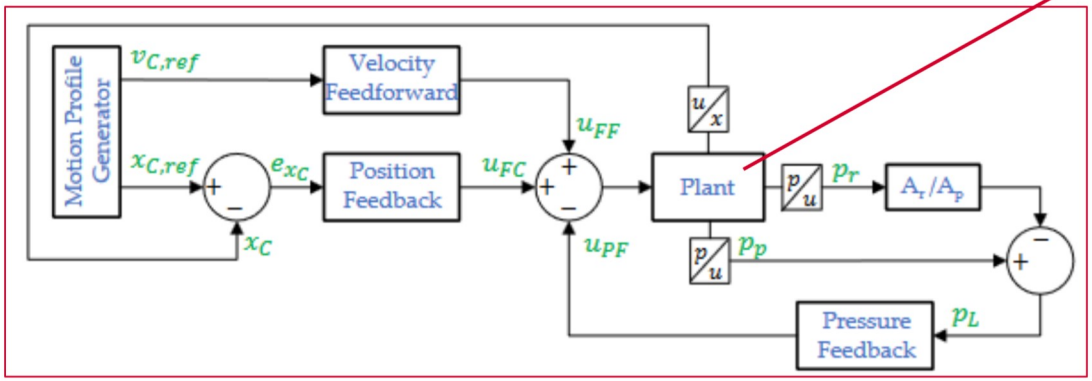
Green Crane simulator example

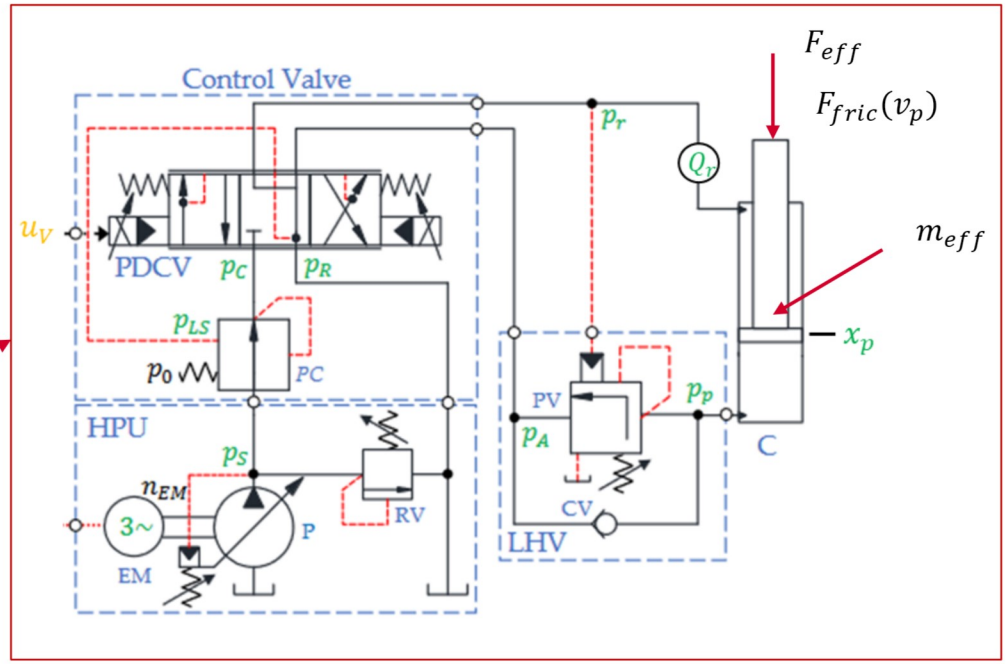
Task1 (10ms)

MAIN
(P_CraneControl)

Task2 (0.1ms)

P_Crane Simulator





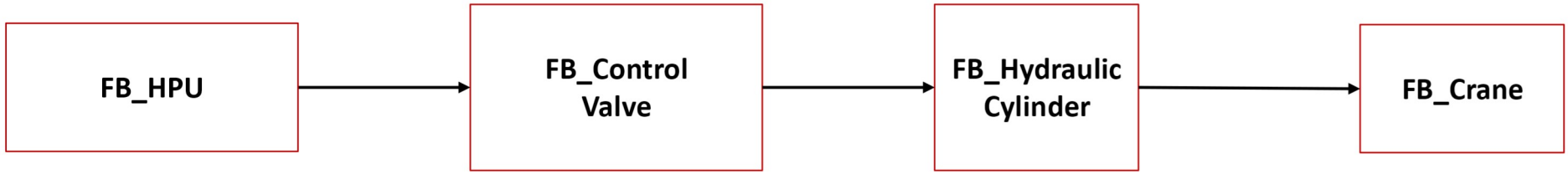
UiA

See Simulink model for details: GreenCraneSimulator.slx - https://github.com/DrDanielh/MAS418_SimulinkModel

Green Crane simulator example

P_Crane Simulator

- **Overview**



Out:
- Ps=180 [bar]
- Pr=1 [bar]

In:
- Ps=180 [bar]
- Pr=1 [bar]
- uValve

Out:
- Qa
- Qb

In:
- Qa
- Qb
- xCyl
- vCyl

Out:
- Fhyd

In:
- Fhyd

Out:
- xCyl
- vCyl

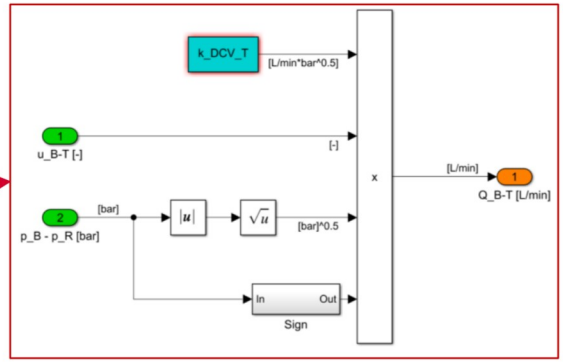
UiA

See Simulink model for details: GreenCraneSimulator.slx - https://github.com/DrDanielh/MAS418_SimulinkModel

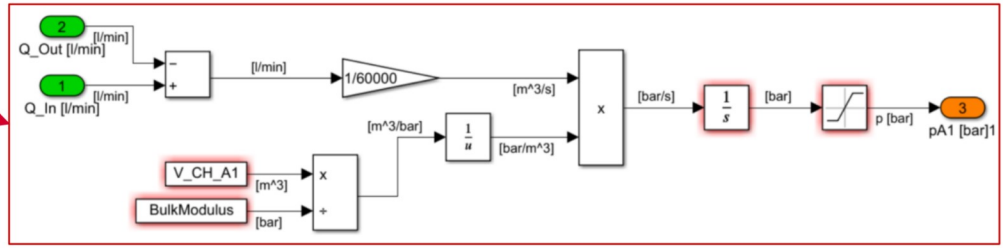
Green Crane simulator example

- Other **functions** / **function blocks** you have to make and reuse withing the main FBs

FB_ValveFlow



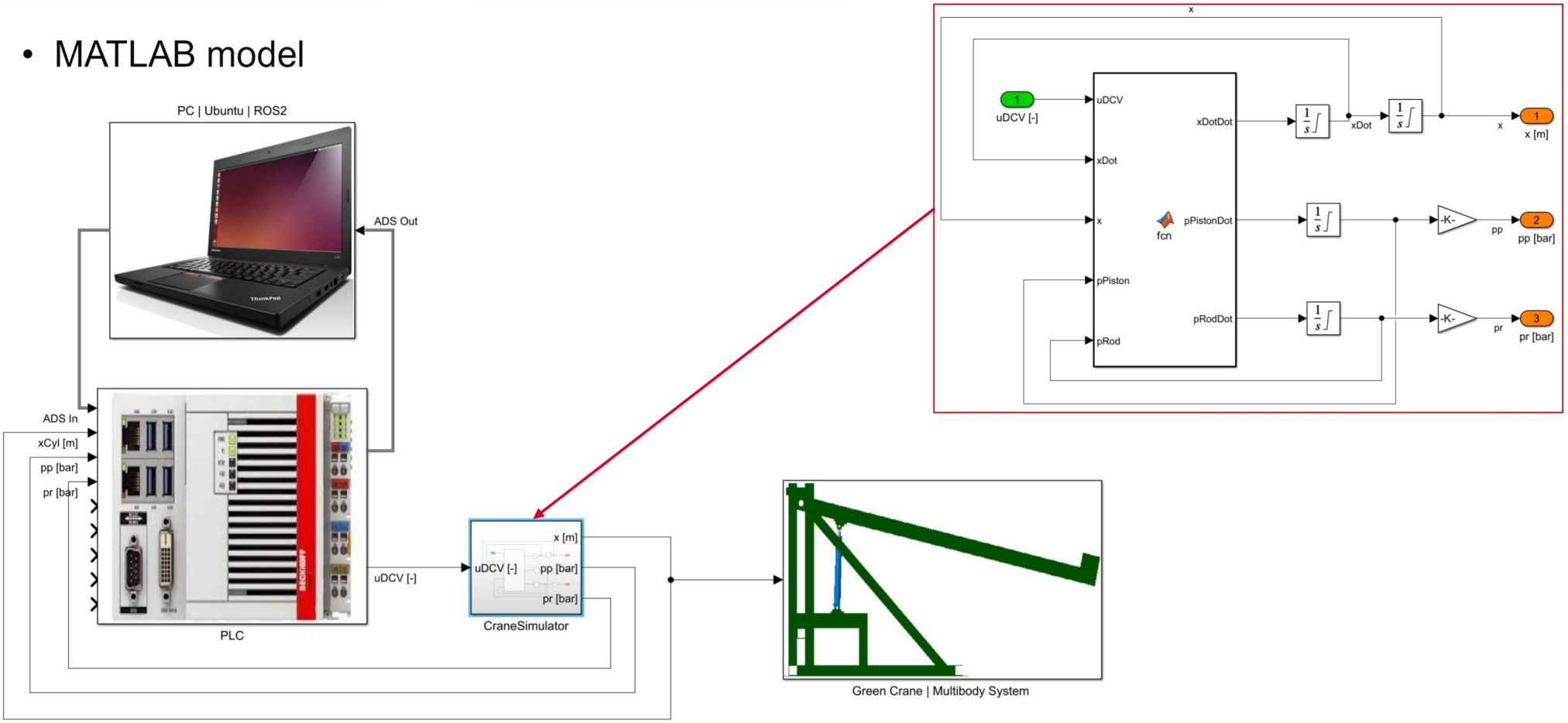
FB_PressureBuildup



See Simulink model for details: GreenCraneSimulator.slx - https://github.com/DrDanielh/MAS418_SimulinkModel

Simplified hydro-mechanical model

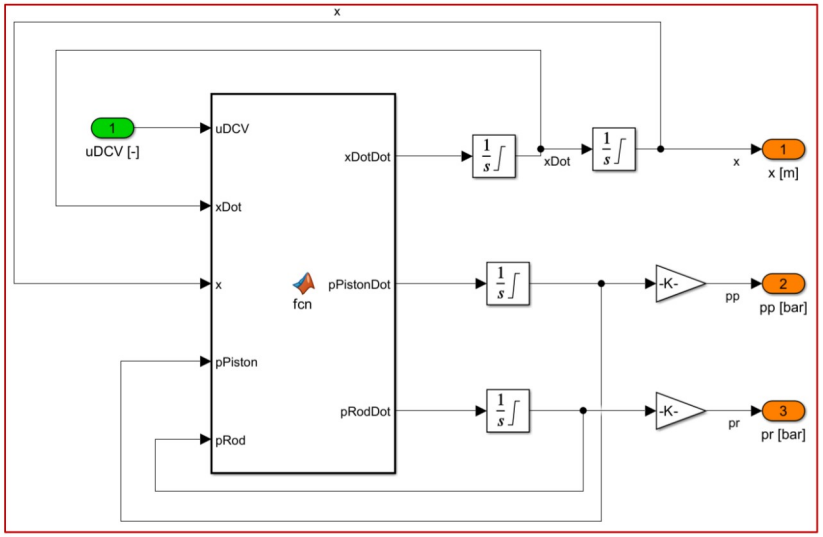
- MATLAB model



See Simulink model for details: GreenCraneSimulator_SomplifiedHydMechModel.slx - https://github.com/DrDanielh/MAS418_SimulinkModel

Simplified hydro-mechanical model

- MATLAB model

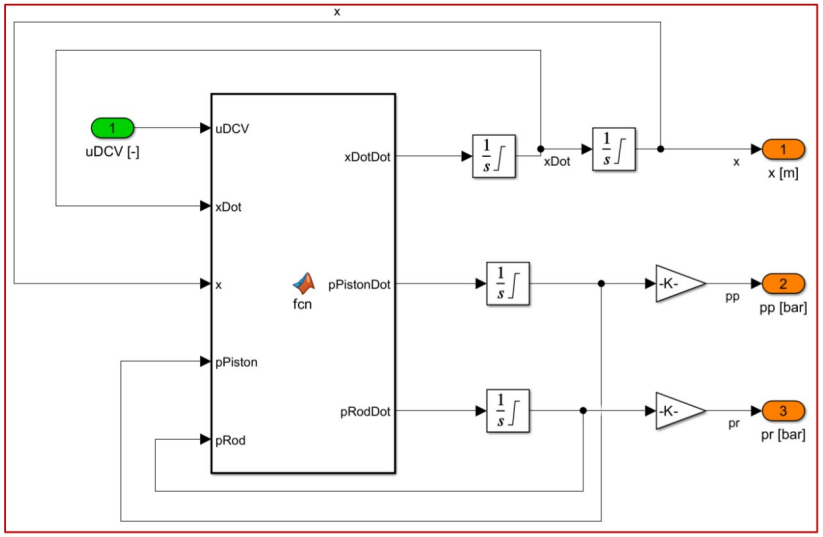



```
function [xDotDot, pPistonDot, pRodDot] = fcn(uDCV, xDot, x, pPiston, pRod)
%% Parameters
Fexternal = 1.3365e+04; % [N]
mEffective = 1.3365e+04;  % [kg]
kFric = 4000; % [N/(ms)]
Apiston = 0.0033; % [m^2]
Aannulus = 0.0023; % [m^2]
beta = 1.0e9; % [Pa]
pSupply = 180e5; % [Pa]
pReturn = 1.1e5; % [Pa]
MaxStroke = 0.5; % [m]
VCyl_A = 1.0e-03;
VCyl_B = 1.25e-03;
p_nom_DCV = 7e5;
Q1_ref_DCV = 20/6e4;
kv1_DCV = Q1_ref_DCV/sqrt(p_nom_DCV);
p0 = 7e5; %compensator crack pressure
```

See Simulink model for details: GreenCraneSimulator_SomplifiedHydMechModel.slx - https://github.com/DrDanielh/MAS418_SimulinkModel

Simplified hydro-mechanical model

- MATLAB model

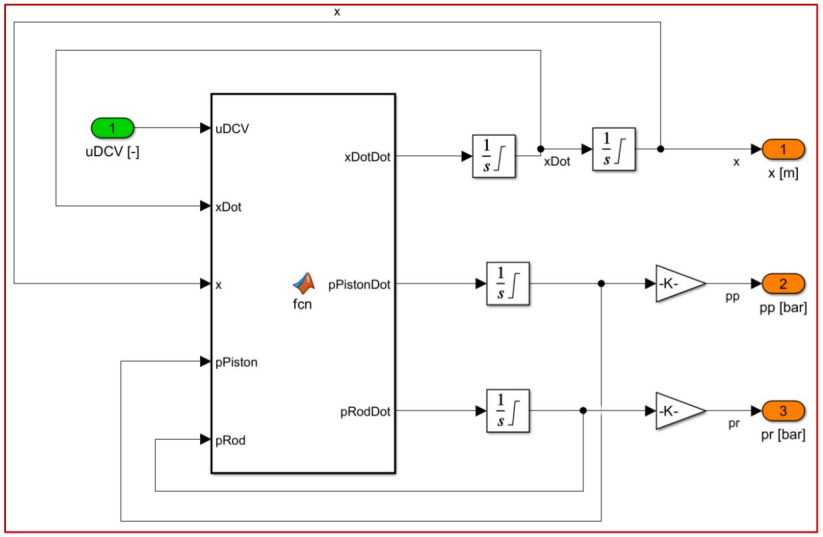



```
%% Directional Control Valve - Lifting Cylinder
% LS Pressure
if uDCV > 0
     pLS = pPiston;
 elseif uDCV < 0
     pLS = pRod;
 else
     pLS = 0;
end

if pLS<0
    pLS = 0;
end

% Pressure Compensator
pPC = 0;

if p0 <= (pSupply - pLS)
      pPC = pLS+p0;
  elseif 0 <= (pSupply - pLS) < p0
       pPC = pSupply-pLS;
  elseif (pSupply - pLS) <= 0
      pPC = pLS;
end

 if pPC<0
     pPC = 0;
 end
% Orifice Equations
if uDCV > 0
QDCV_in =  kv1_DCV*uDCV*sign(pPC-pPiston)*sqrt(abs(pPC-pPiston));
QDCV_out = kv1_DCV*uDCV*sign(pRod-pReturn)*sqrt(abs(pRod-pReturn));
elseif uDCV < 0
QDCV_out =  kv1_DCV*uDCV*sign(pPC-pRod)*sqrt(abs(pPC-pRod));
QDCV_in =  kv1_DCV*uDCV*sign(pRod-pReturn)*sqrt(abs(pRod-pReturn));
else
QDCV_in = 0;
QDCV_out = 0;
end
```

See Simulink model for details: GreenCraneSimulator_SomplifiedHydMechModel.slx - https://github.com/DrDanielh/MAS418_SimulinkModel

Simplified hydro-mechanical model

- MATLAB model




```
%% Pressure Gradients
VA = VCyl_A + x*Apiston;
VB =  VCyl_B + (MaxStroke - x)*Aannulus;
VA_dot = Apiston*xDot;
VB_dot = -Aannulus*xDot;
pPistonDot = beta/VA*(QDCV_in - VA_dot);
pRodDot = beta/VB*(-QDCV_out - VB_dot);

% Viscouse Friction
Ffric = kFric*xDot;
```

```
%% Hydraulic Force
Fhyd = pPiston*Apiston - pRod*Aannulus;
```

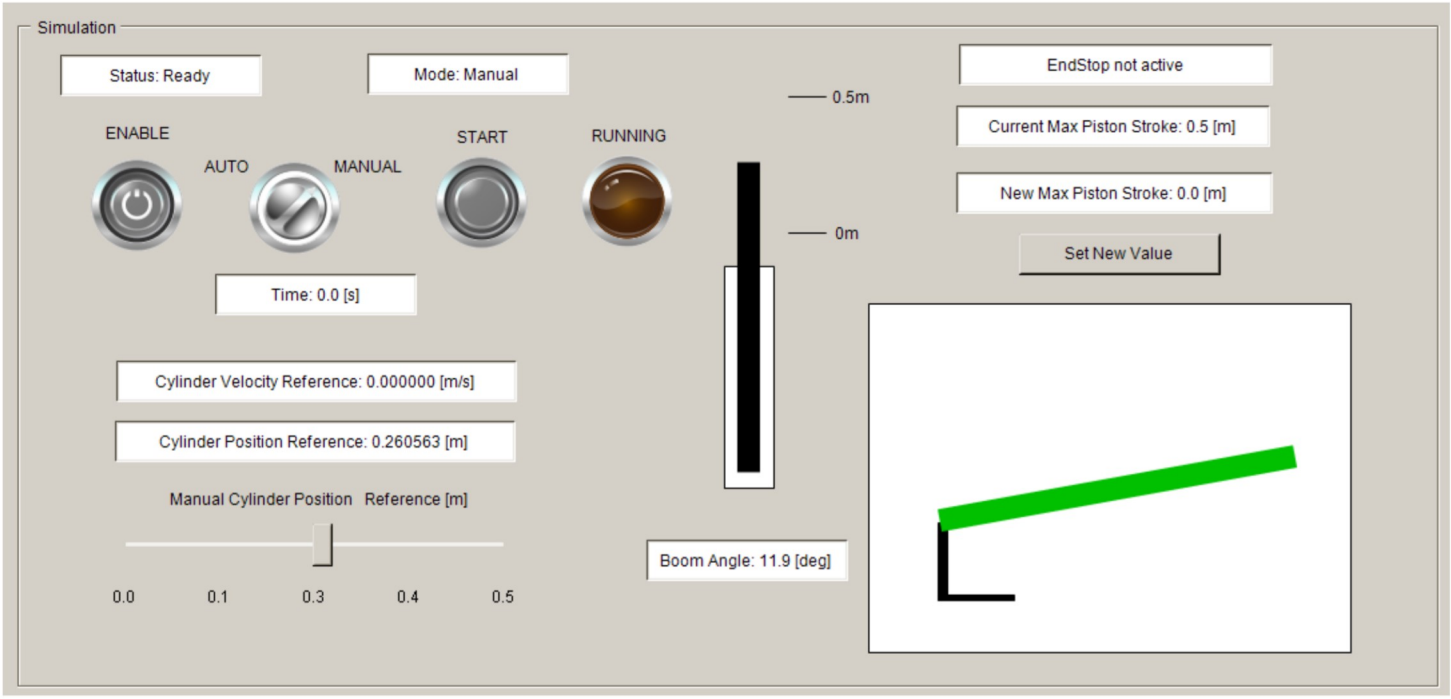
```
%% Newton Second Law of Motion
xDotDot = (Fhyd - Ffric - Fexternal)/mEffective;
```

UiA

See Simulink model for details: GreenCraneSimulator_SomplifiedHydMechModel.slx - https://github.com/DrDanielh/MAS418_SimulinkModel

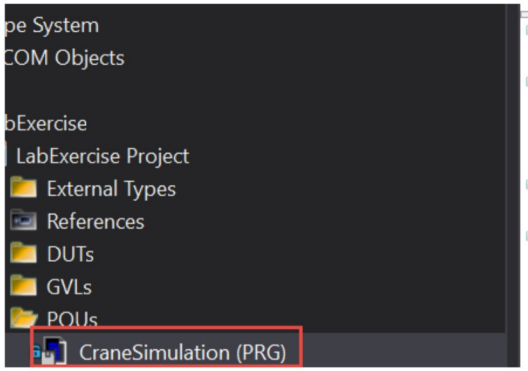
Time-domain simulation

Example of Green Crane



Time-domain simulation

Example of Green Crane

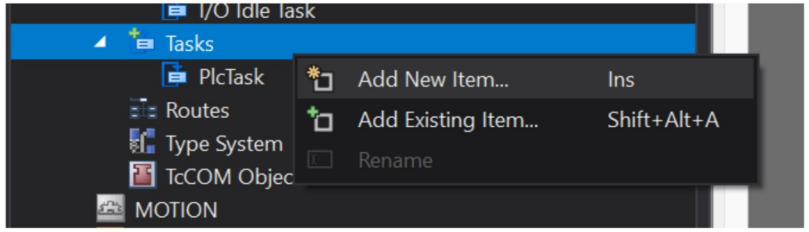


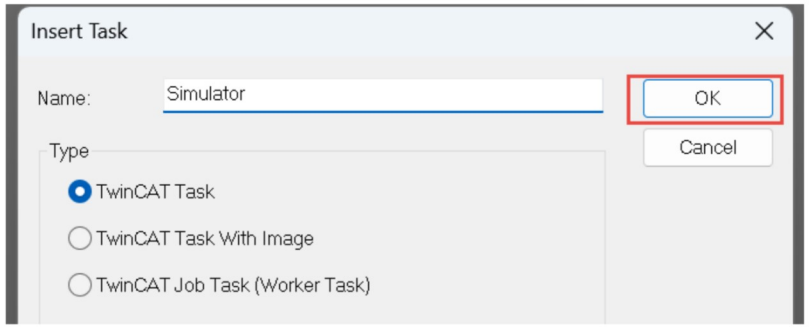

```
IF Inputs.bStartSimulation AND Inputs.bAutoMode  THEN
    fTime := fTime + Parameters.fCyclicTime;
ELSE
    fTime := 0;
END_IF

IF fTime > 0 AND Inputs.bAutoMode THEN
    bRunning := TRUE;
ELSE
    bRunning := FALSE;
END_IF

fPositionAutoMode := Inputs.fCylinderVelocityReference * 1/2*(fTime*fTime) + fInitialVelocity * fTime + fInitialPosition;
```

UiA

Create new task in TwinCAT

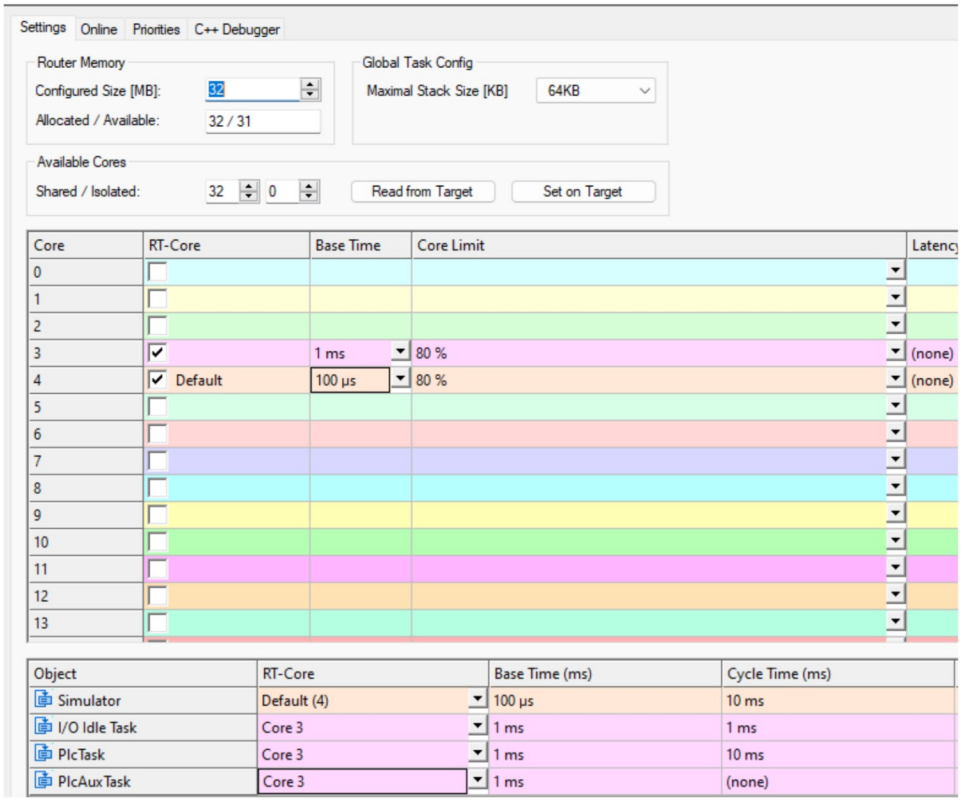
1. **Create new task** and name it





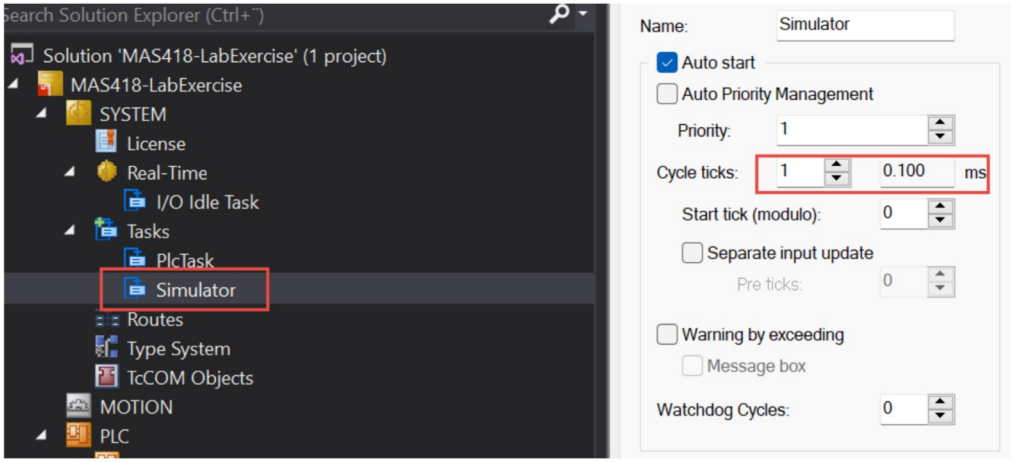
Create new task in TwinCAT

1. **Create new task** and name it
2. In real time window – **assign a new core** with base time 100 microseconds



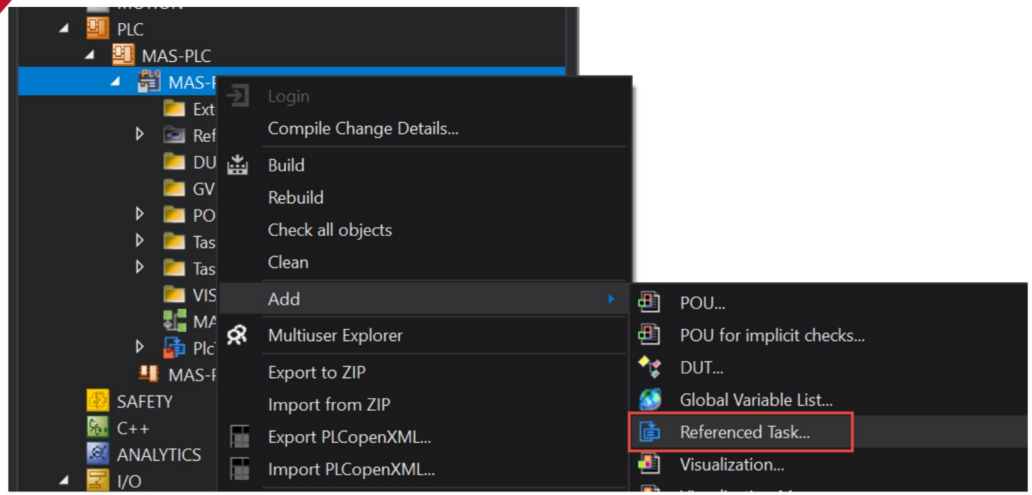
Create new task in TwinCAT

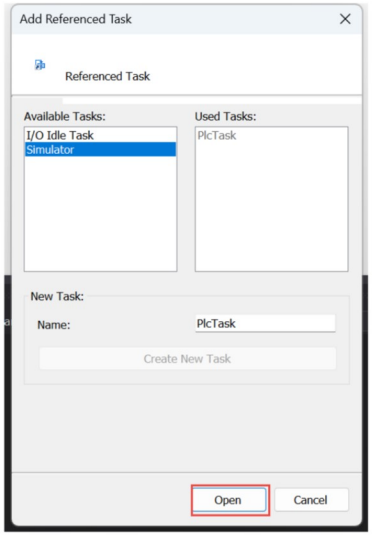
1. **Create new task** and name it
2. In real time window – **assign a new core** with base time 100 micro seconds
   - must be isolated if using the virtual machine
3. Adjust cycle time (ticks) to 0.1ms



Create new task in TwinCAT

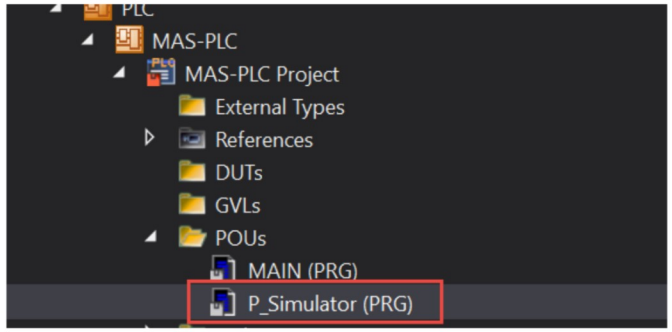
1. **Create new task** and name it
2. In real time window – **assign a new core** with base time 100 micro seconds
   - must be isolated if using the virtual machine
3. Adjust cycle time (ticks) to 0.1ms
4. Add mew Referenced Task and select the new task





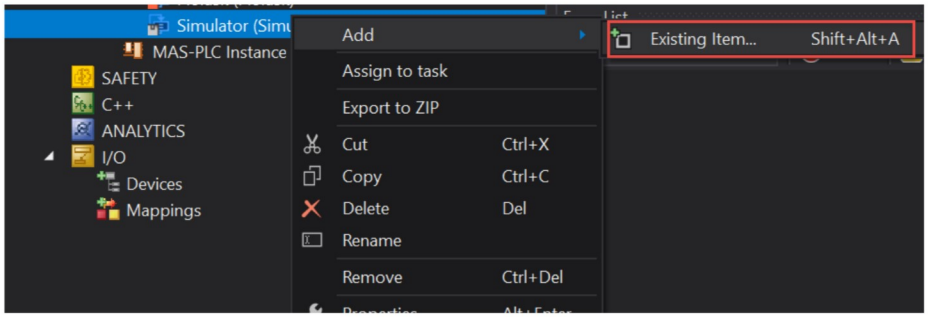
Create new task in TwinCAT

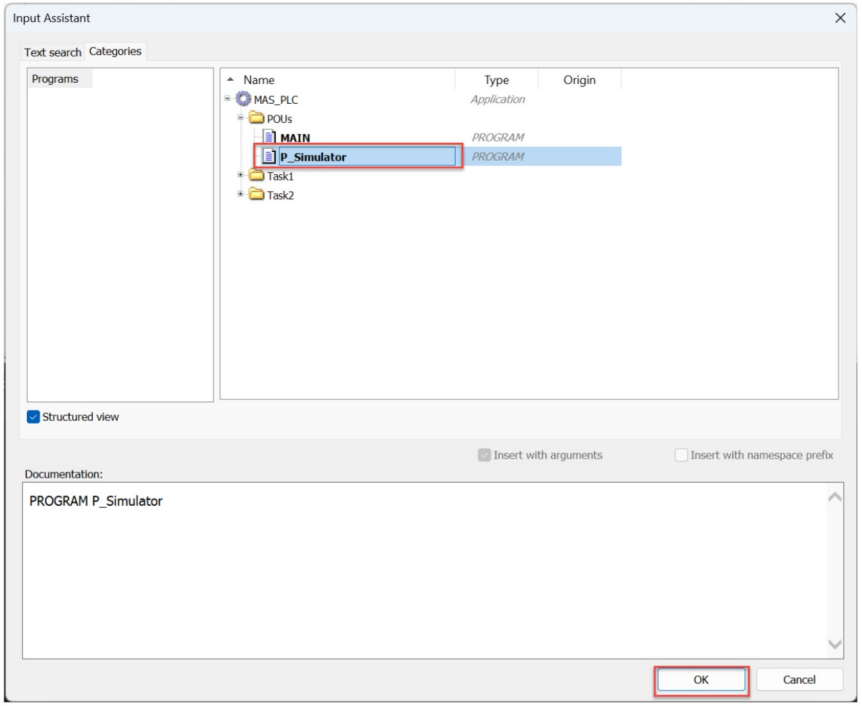
1. **Create new task** and name it
2. In real time window – **assign a new core** with base time 100 micro seconds
   - must be isolated if using the virtual machine
3. Adjust cycle time (ticks) to 0.1ms
4. Add mew Referenced Task and select the new task
5. Create a new Program



Create new task in TwinCAT

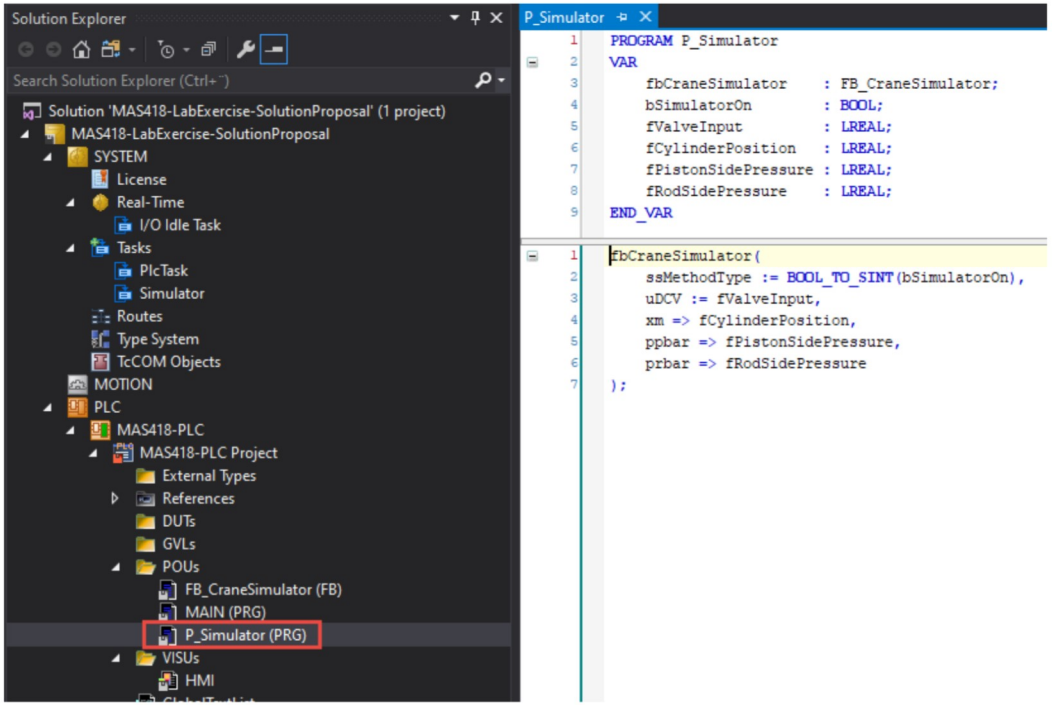
1. **Create new task** and name it
2. In real time window – **assign a new core** with base time 100 micro seconds
   - must be isolated if using the virtual machine
3. Adjust cycle time (ticks) to 0.1ms
4. Add mew Referenced Task and select the new task
5. Create a new Program
6. Assign the new Program to the task





Create new task in TwinCAT

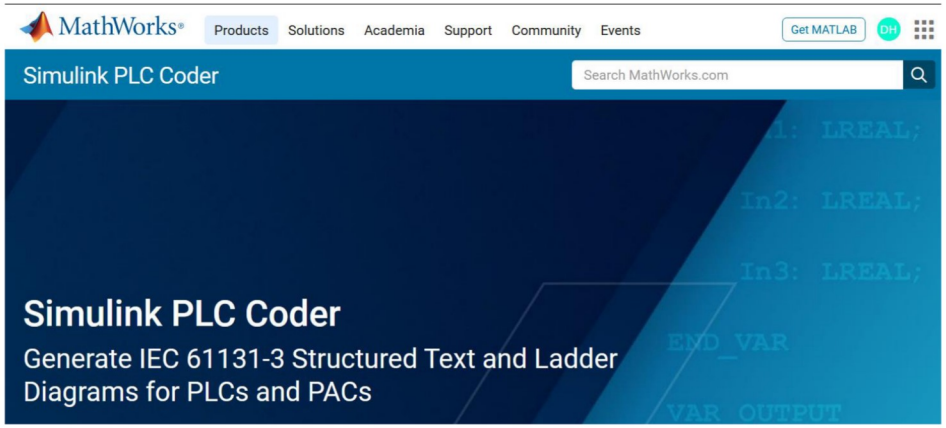
1. **Create new task** and name it
2. In real time window – **assign a new core** with base time 100 micro seconds
   - must be isolated if using the virtual machine
3. Adjust cycle time (ticks) to 0.1ms
4. Add mew Referenced Task and select the new task
5. Create a new Program
6. Assign the new Program to the task
7. Create a FB(s) for the simulator (or implement from Simulink PLC coder), instantiate it in the new program



Simulink PLC Coder

Procedure

- Use a separated Simulink project specific settings
- Use supported Simulink blocks (i.e. discrete integrators etc.)
- When building new code, delete previous generated folder
- The blocks need to be in a subsystem



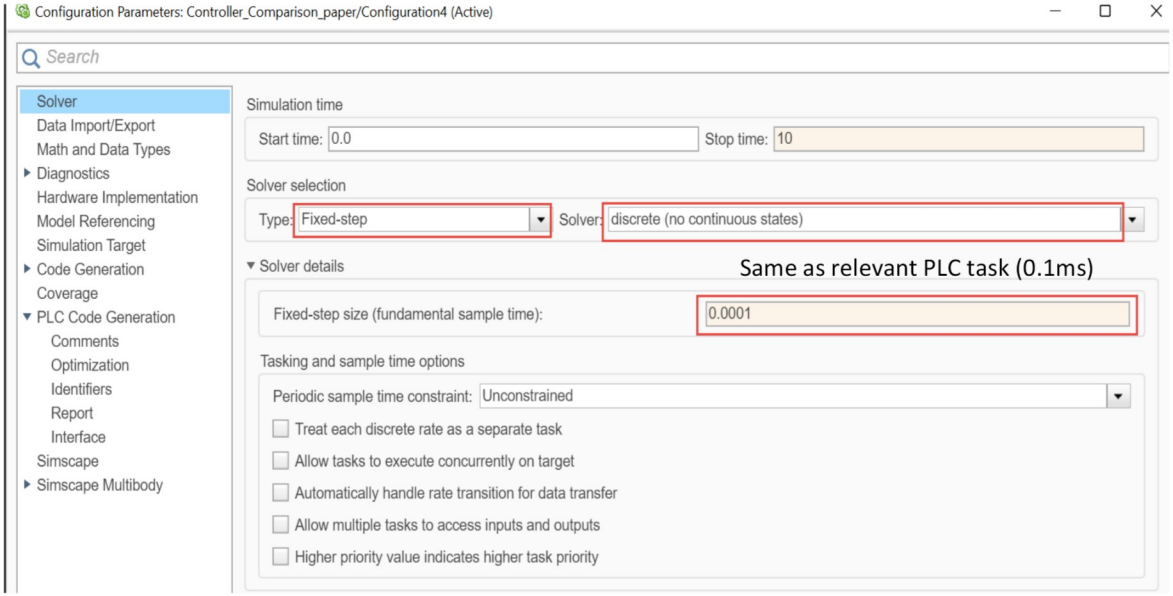
https://se.mathworks.com/products/simulink-plc-coder.html

UiA

Simulink PLC Coder

Simulink settings

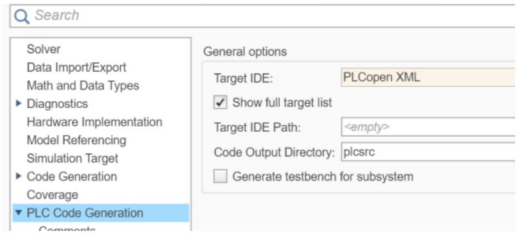
- Solver

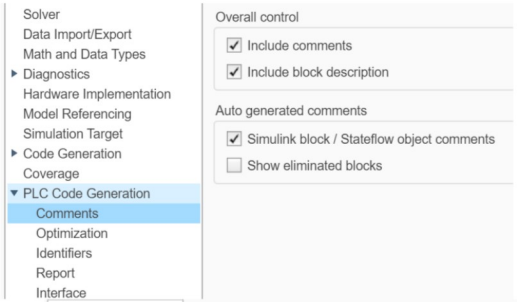


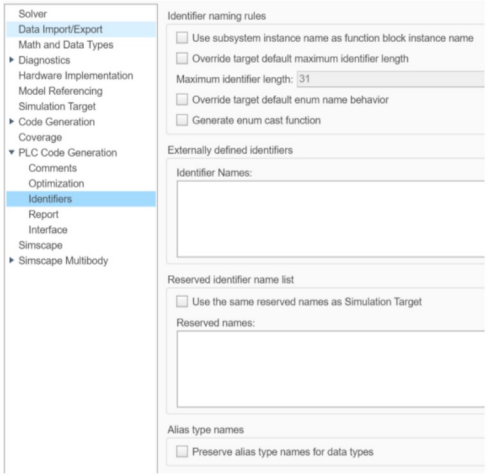
Simulink PLC Coder

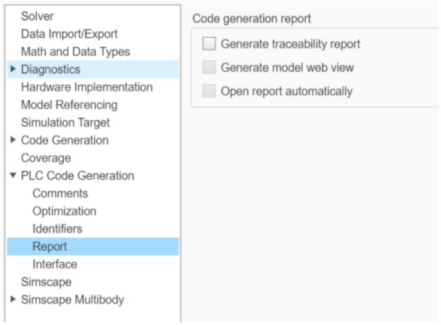
Simulink settings

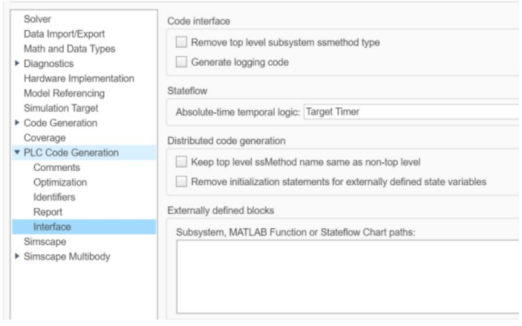
- Solver
- PLC coder options
  - Play with them

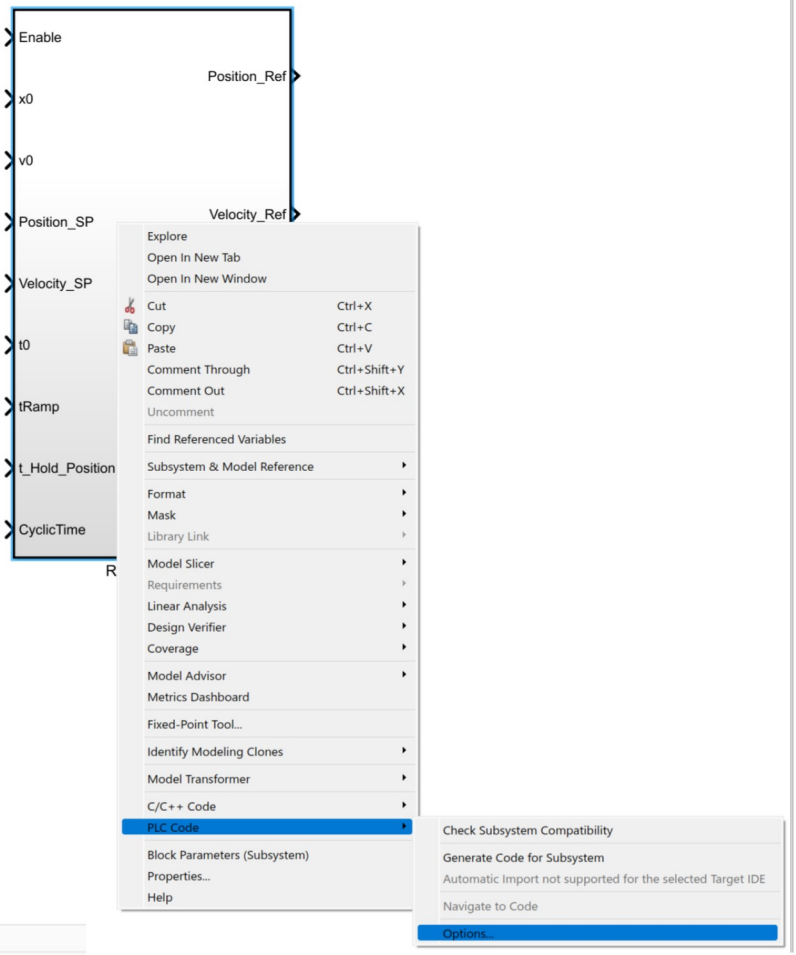








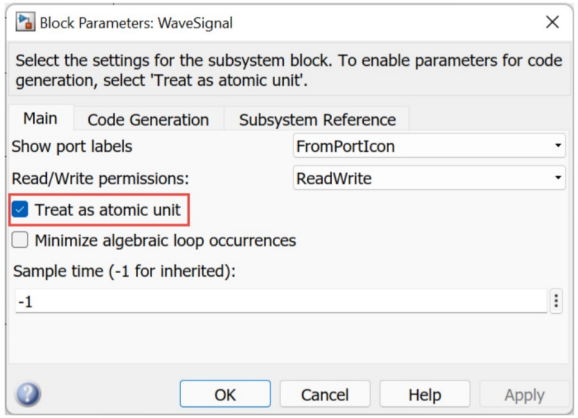


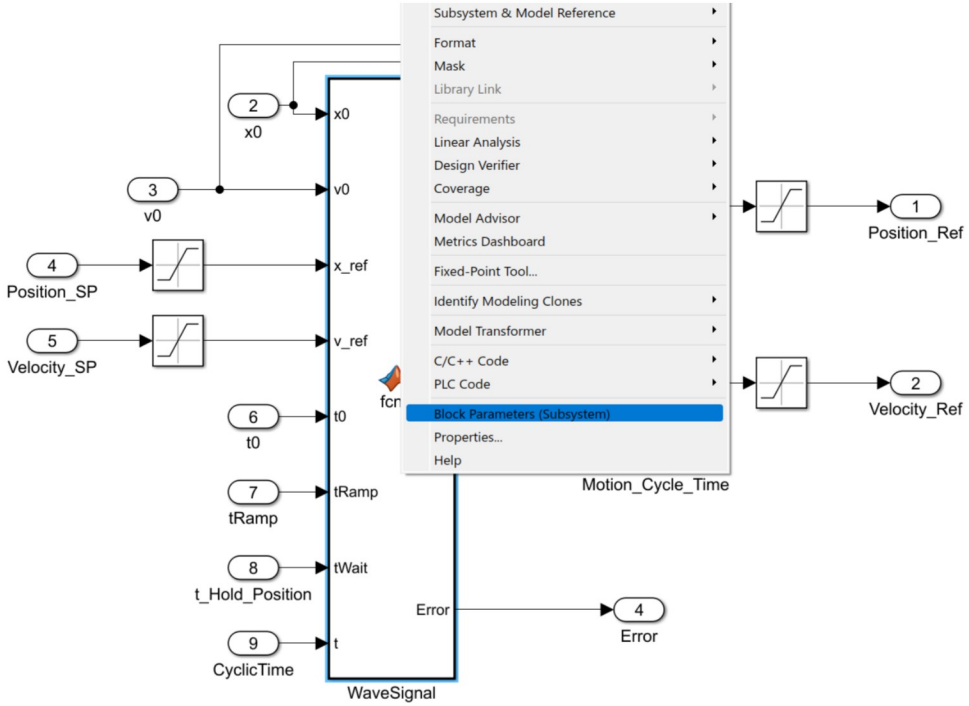


Simulink PLC Coder

Simulink settings

- Solver
- PLC coder options
  - Play with them
- MATLAB function settings

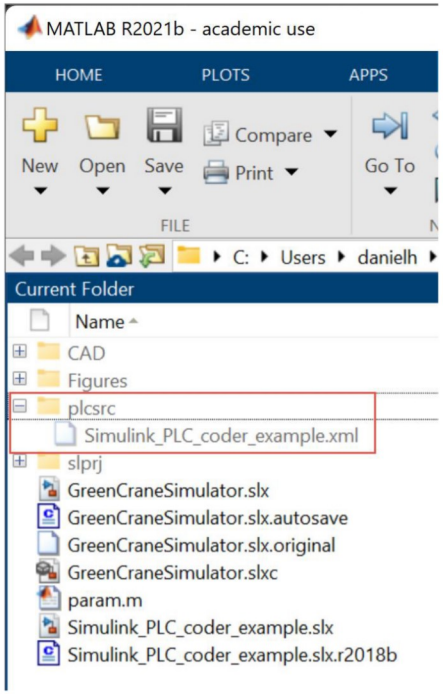


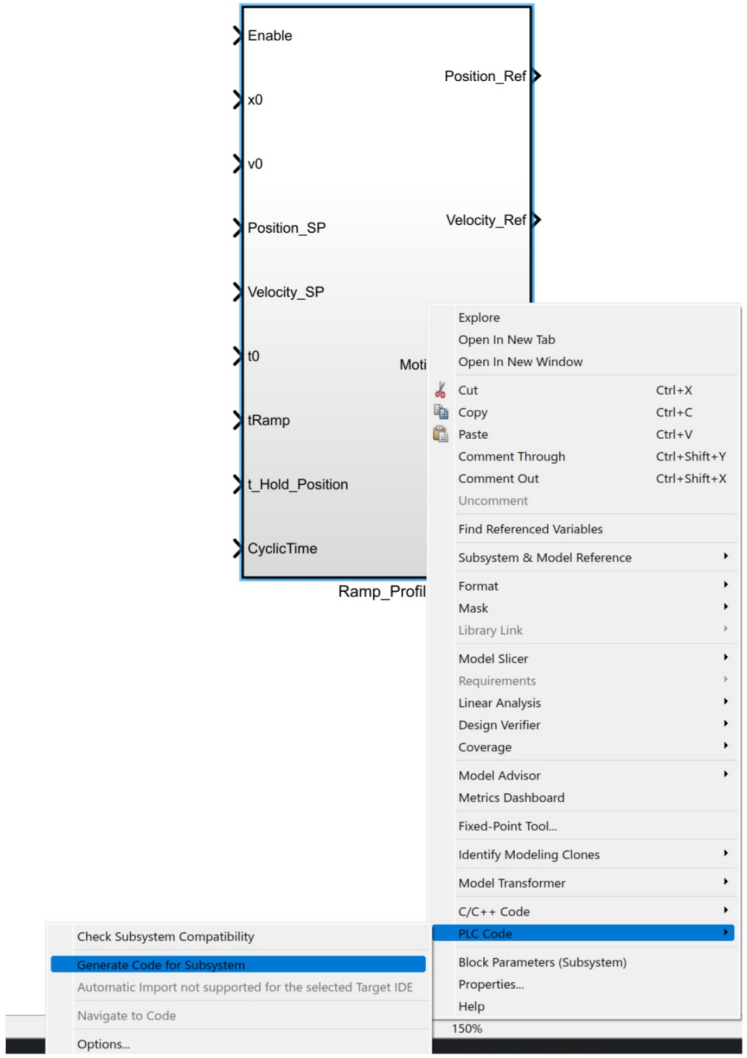


See Simulink model for details: https://github.com/hagenmek/MAS418_SimulinkModel

Simulink PLC Coder

Code generation

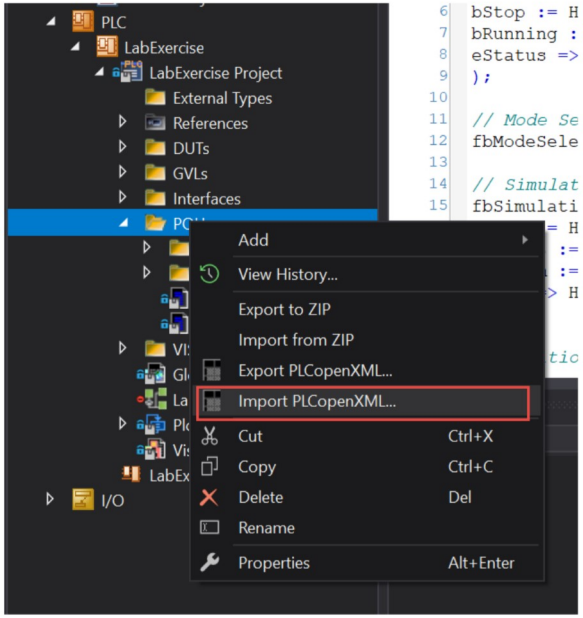


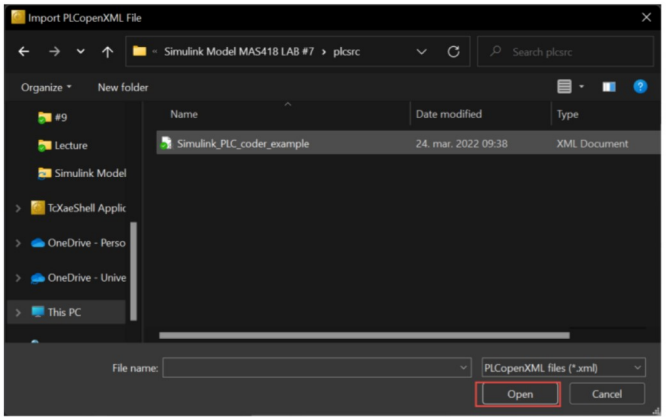


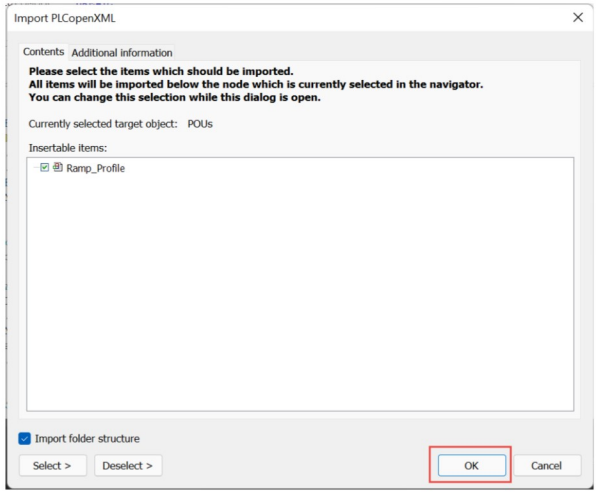
See Simulink model for details: https://github.com/hagenmek/MAS418_SimulinkModel

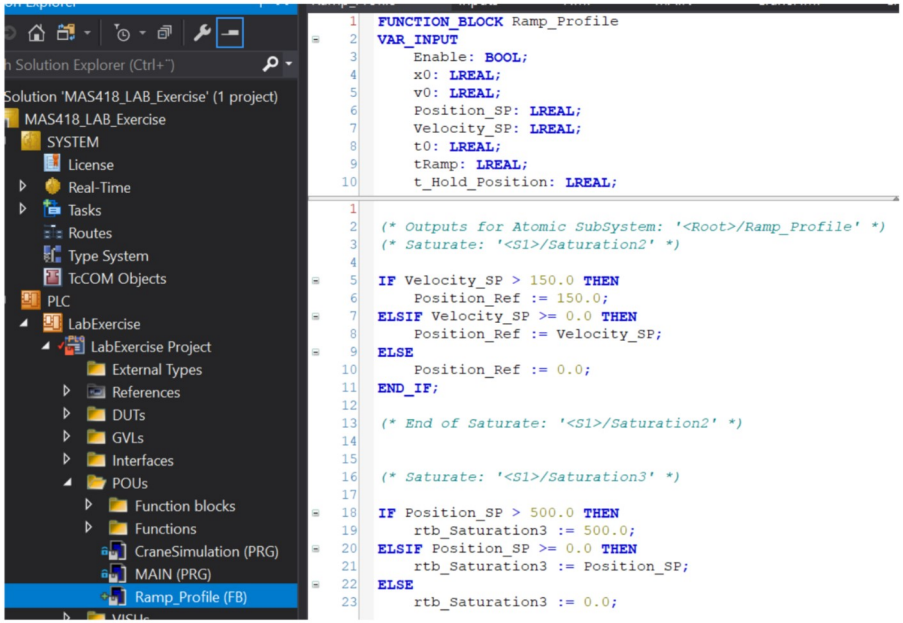
Simulink PLC Coder

Implement code in TwinCAT









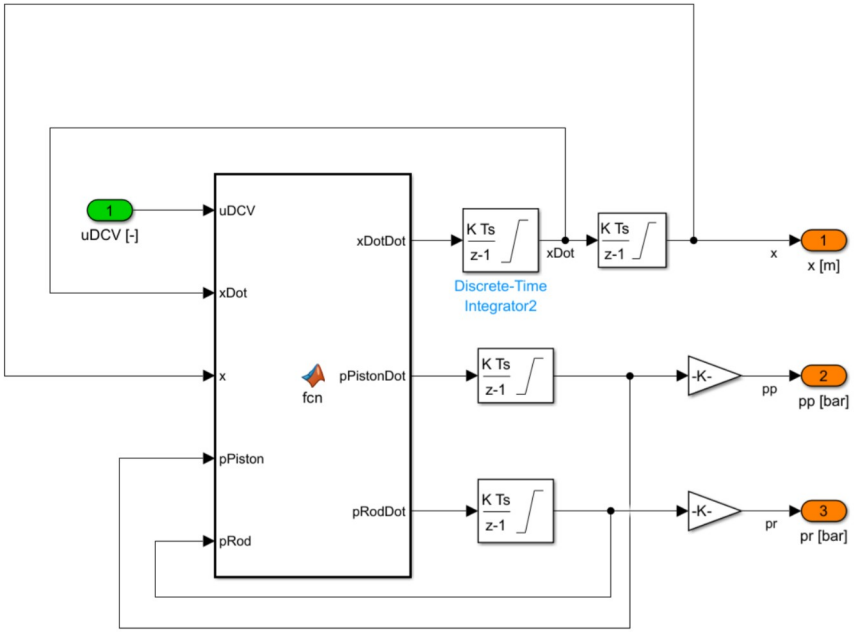

```
FUNCTION_BLOCK Ramp_Profile
VAR_INPUT
    Enable: BOOL;
    x0: LREAL;
    v0: LREAL;
    Position_SP: LREAL;
    Velocity_SP: LREAL;
    t0: LREAL;
    tRamp: LREAL;
    t_Hold_Position: LREAL;
```

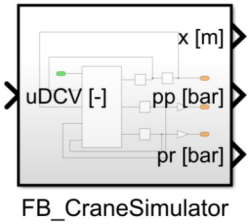
```
(* Outputs for Atomic SubSystem: '<Root>/Ramp_Profile' *)
(* Saturate: '<S1>/Saturation2' *)

IF Velocity_SP > 150.0 THEN
    Position_Ref := 150.0;
ELSIF Velocity_SP >= 0.0 THEN
    Position_Ref := Velocity_SP;
ELSE
    Position_Ref := 0.0;
END_IF;

(* End of Saturate: '<S1>/Saturation2' *)


(* Saturate: '<S1>/Saturation3' *)

IF Position_SP > 500.0 THEN
    rtb_Saturation3 := 500.0;
ELSIF Position_SP >= 0.0 THEN
    rtb_Saturation3 := Position_SP;
ELSE
    rtb_Saturation3 := 0.0;
```

UiA

Simulink PLC Coder

Simplified hydro-mechanical Simulator

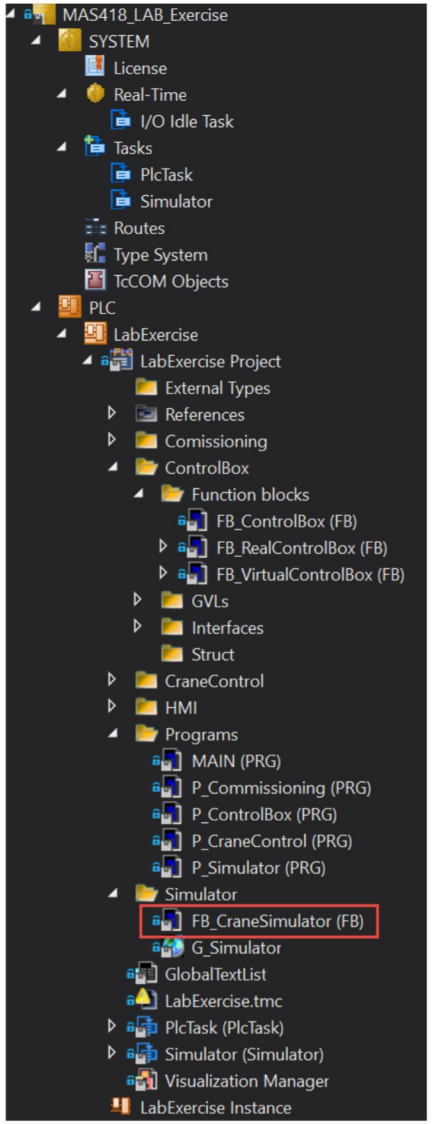




See Simulink model for details:

Simulink_PLC_coder_example_LAB_11_simulator.slx

https://github.com/DrDanielh/MAS418_SimulinkModel

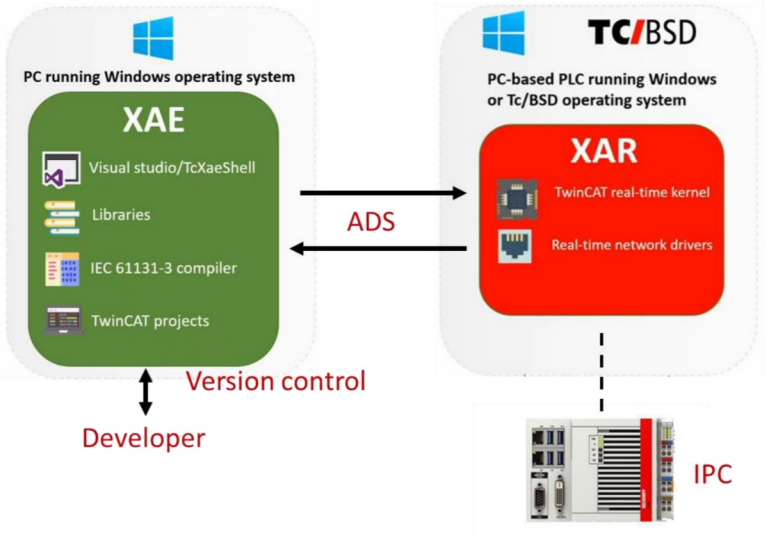


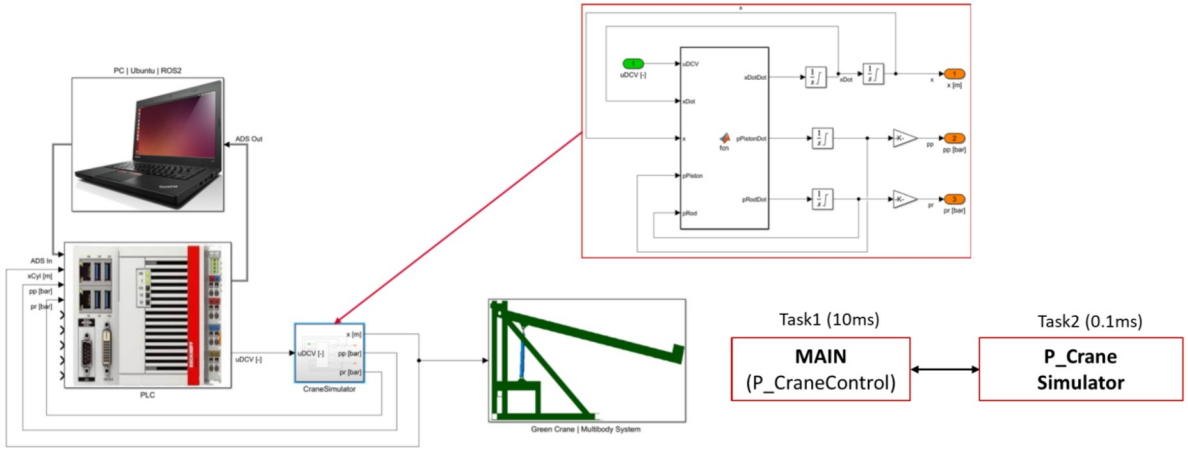
Summary

UiA

Summary

- **Course Introduction**
- **Introduction to PLC programming**
  - SCADA system
  - History
  - IEC61131-3 standard
  - Basic data types
  - Version-control
- **TwinCAT basics**
  - Introduction to TwinCAT
  - Difference to SW running in OS (self-study)
- **Simulation in TwinCAT**
  - Simulink Example of Green Crane
  - Green Crane simulator example
  - Simplified hydro-mechanical model
  - Time-domain simulation
  - Create new task in TwinCAT
  - Simulink PLC Coder





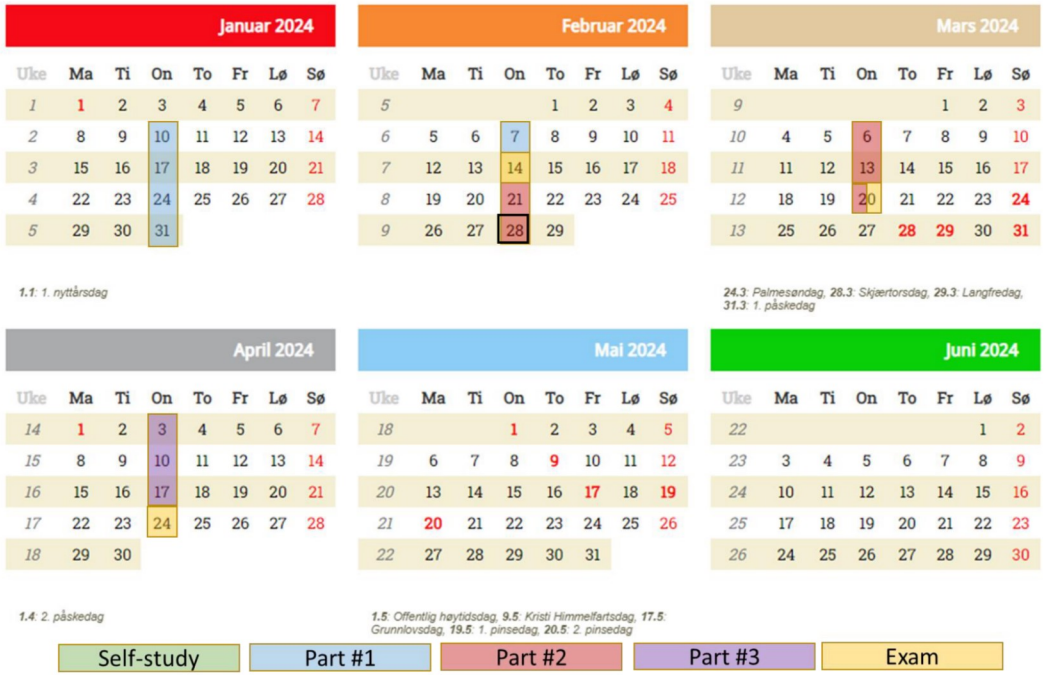
Next Lecture

Procedural-oriented PLC programming:

I. **Data types cont. & Std. library**
II. **Structures & functions**
III. **Instructions**

Homework:

- PLC programming using TwinCAT 3 - Data types & arrays (Part 4/18) (youtube.com)
- PLC programming using TwinCAT 3 - Structures & functions (Part 5/18) (youtube.com)
- PLC programming using TwinCAT 3 - Tc2_Standard (Part 8/18) (youtube.com)
- PLC programming using TwinCAT 3 - Tc2_Standard (Part 7/18) (youtube.com)



Course plan: MAS418-G 24V Programmering for intelligent robotikk og industrielle systemer (instructure.com)

Lab exercise

Lab Exercise #2.1 – Basic PLC programming