Cape Peninsula
University of Technology

Design and Implementation of IEC 61499 standard-based controllers in a distributed control environment

by

Kevin Love

Thesis - 100% research project submitted in fulfillment of the requirements for the degree

Master of Technology/Doctor of Technology: **Electrical Engineering**

in the Faculty of **Electrical Engineering**

at the Cape Peninsula University of Technology

Supervisor: **Dr C Kriger**
Co-supervisor: **Dr N Tshemese-Mvandaba**

Bellville, Cape Town

November 2023

DECLARATION

I, Kevin Love, declare that the contents of this thesis represent my unaided work, and that the thesis/dissertation has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

[illegible]	**12/11/2023**
Signed	**Date**

ABSTRACT

The Fourth Industrial Revolution changed how people work, live, and interact with each other and technology with a shift towards automation and data exchange that requires software to be portable, interoperable, configurable, and reusable between multiple Original Equipment Manufacturers (OEMs). The interoperability challenge is overcome by the adherence to governing standards by the producers of the different software programming environments that are used to develop the control systems. PLCOpen function blocks, Codesys integrated development environment, the IEC 61499 Standard, and the EtherCAT network topology, are all examples of software aspects that are used to improve the integration between automation hardware from different vendors.

In this thesis, the MATLAB/Simulink software engineering environment is used to develop a mathematical model of a DC motor control system that is used to control the azimuth and altitude positional movements of a radio antenna dish. A full-state feedback controller is designed to increase the response time of the positional movements to a set point change. Integral control is also added to the system to compensate for the steady-state error caused by using a full-state feedback controller.

The developed simulation model is tested in the Simulink software environment by analysing the results of a step response input to the system. The response of the DC motor open-loop system, a DC motor system with a controller, and a DC motor control system with added integral control, is compared and analysed. The effects of the network-induced delays are also analysed before implementing the controller on the hardware. The effects show that an increase in network delays leads to an increase in system instability.

The thesis findings contribute to detailing the transformation process for the developed controller from the Simulink simulation environment to the TwinCAT 3 programming environment to allow for the real-time implementation of an actual DC motor. The transformed simulation model interacts with the DC motor from a PLC, through an EtherCAT network, to a remote motor controller.

The real-time hardware-in-the-loop implementation results are compared to the results acquired by the simulations done in Simulink. The results show that the effects of network delays are the same in real-time as in the simulation model. The addition of Beckhoff's time compensation feature in TwinCAT 3 reduced the effects of time delays and resulted in a stable system. The control system is also stress-tested to record the limitations of the positional movements.

The thesis findings and deliverables further contribute to the enlarging of the knowledge base in the field of IEC 61499 standard-based control systems and can be used for education to continue further research. The state-space method used in the mathematical model for the design of the controller can be implemented in other similar applications that require a change in angular position. The hardware-in-the-loop test rig can also be used in future research work by postgraduate students at universities or research institutions.

ACKNOWLEDGEMENTS

I wish to thank:

- My supervisor Dr Carl Kriger and co-supervisor Dr Nomzamo Tshemese-Mvandaba for their support, advice, guidance, and contribution to this thesis. Thank you very much for the time you have taken to assist me in this journey.
- My partner Tamika Devon for her support, encouragement, understanding, and patience during my time studying. Thanks for always being by my side.
- My employer Dave Grobler for his generosity gave me the opportunity to do this thesis. He will be missed. Rest in peace.
- My friends at Beckhoff Automation, Dane Potter, and Shaun Potter, for their continuous support and assistance when needed.

DEDICATION

This thesis is dedicated to my mom and dad, Patricia Anne Swanepoel and Anthony Macaulay Swanepoel, for their continuous support and guidance throughout my entire life up until now. I appreciate everything you've done for me. I love you both.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

GLOSSARY OF TERMS

Abbreviations	**Definition/Explanation**
BHS	Baggage Handling System
CAN	Control Area Network
CFBs	Composite Function Blocks
CPU	Central Processing Unit
DALI	Digital Addressable Lighting Interface
DC	Direct Current
DCS	Distributed Control System
DT/DE	Discrete Time/Event Approach
EM	Electromagnetic
emf	Electromotive Force
EtherCAT	Ethernet for Control Automation Technology
FBench	Floating Point Benchmark
FBDK	Function Block Development Kit
FIFO	First-in-first-out
HartRAO	Hartebeesthoek Radio Astronomy Observatory
IEC	International Electrotechnical Commission
I/O	Input/Output
LED	Light Emitting Diode
LMI	Linear Matrix Inequalities
MATLAB	MATrix LABoratory
MPC	Model Predictive Controller
MPS	Modular production system
NCES	Net Condition/Event System
NCS	Networked control system
OEMs	Original Equipment Manufacturers
PC	Personal Computer
PID	Proportional Integral Derivative
PLC	programmable logic controller
UML	Unified Modeling Language
RAM	Random Access Memory
ROM	Read-only memory
RT-UML	Real-time Unified Modeling Language
SIFB	Service Interface Function Block
SPADE	Smart Program Agent Development Environment
TcCom	TwinCAT Component Object Model
TwinCAT	Windows Control and Automation Technology
ViEd	Visual NCES Editor
Vive	Visual Verifier
XAE	eXtended Automation Engineering
XAR	TwinCAT Runtime

LIST OF SYMBOLS

$\vec{F}$	Magnetic Force
$\vec{B}$	Magnetic Field
$\vec{E}$	Electric Field
q	Charge
$\vec{v}$	Velocity of a charged particle
V_a	Armature voltage
R_a	Armature Resistance
i_a	Armature current
L_a	Armature inductance
e_b	Back electromotive force
K_B	Electromotive torque constant
θ_m	Angular position
$\dot{\theta}_m$	Angular velocity
$\ddot{\theta}_m$	Angular acceleration
T_m	Input torque
B_m	Damping coefficient
J_m	Moment of inertia
K_T	Torque constant
J_a	Moment of inertia of the armature
J_L	Moment of inertia of the load
B_a	Motor damping coefficient
B_L	Load damping coefficient
$N1$	Number of gears teeth on the motor side
$N2$	Number of gears teeth on the load side
K	State feedback gain matrix
A_{CL}	Closed loop A matrix
P_C	Controllability matrix
Λ	A constant
I	Identity matrix
K_i	Integral matrix
L	Observer gain matrix
P_O	Observability matrix
T_{sc}	Sensor-to-controller delay
T_{ca}	Controller-to-actuator delay

CHAPTER 1
INTRODUCTION

1.1 Introduction

With the industry moving into its 4th generation, the need for distributed control in industrial control systems is necessary as field devices become more intelligent and data transfer between devices becomes larger. Currently, most systems are controlled centrally with a Programmable Logic Controller (PLC) using the IEC 61131-3 standard. The cyclic nature of these systems prevents adequate control in large systems that consist of multiple Intelligent Electronic Devices (IEDs) networked via Fieldbus communication protocols such as Ethernet/Internet Protocol (Ethernet/IP). To overcome this problem, it is necessary to design systems in a distributed manner.

The IEC 61499 standard-based systems allow for the distribution of control using event-driven function blocks and can therefore be used to distribute control in a decentralized system. The standard also increases the portability of software between different vendors by standardizing the format used by developers when creating software tools. This led to hardware interoperability between devices from different vendors making it possible to communicate between IEDs from different vendors.

An increase in portability, interoperability, configurability, and reusability between different vendor's software tools and hardware devices in industrial control systems decreases the complexity of trying to integrate and communicate between them. Development time will be more focused on the control system and the end product rather than setting up communications. Also, being able to use any product from any vendor will make the market more competitive which will lead to cost savings as vendors will not be able to sell their propriety software at whatever price they like.

This research work focuses on the application of the IEC 61499 standard to show the capabilities of the standard when applied to an industrial control system. The research contributes to the 4th Industrial Revolution that is leading the path to holonic systems in industrial automation.

The following sections describe the awareness of the problem, statement of the problem, the research aim and objectives, research questions and hypothesis, delimitation of the research, deliverables of the project, complete chapter breakdown, and a conclusion.

1.2 Awareness of the problem

The problem with control systems in industry is the use of a centralized point of control. Centralized control is not sufficient in certain industrial practices today as the cyclic nature of how events are handled is not quick enough to control distributed devices. To overcome this problem, it is necessary to design systems in a distributed manner. IEC 61499 standard-based systems allow for the distribution of control using event-driven function blocks and can therefore be used to convert from centralized to decentralized control.

The second problem can be found in networked control systems that are connected via a Fieldbus such as Profinet, Ethernet, Modbus, etc. These networks are affected by network delays that are caused by increased network traffic, the distance between IEDs, and the size of the network system. These network delays can be scaled down when distributing the control as not all the IEDs will have to communicate through a central hub but can instead communicate directly with the intended target. Modeling and analysing these delays allow for better controller design and validation before implementing the control system on a lab-scale plant.

A third problem with automation systems today is the lack of interoperability, portability, and configurability between the software tools and hardware components of the different vendors. Using software packages that are IEC 61499 standard compliant when developing control systems promises a more open environment where switching or communicating between vendors is easier and simpler. An ideal control system allows for the project repositories and software libraries to be portable between any vendors' software environments, and any software tool can configure any vendors' IEDs. This allows for better communication between different vendors' IEDs, increasing interoperability over any Fieldbus network.

1.3 Statement of the problem

The main problem in industry is that centralized PLC control systems cannot sufficiently control intelligent devices on a distributed network system. Distributed systems have become too complex to be controlled from a central point, therefore distributed control is required. This is due to an increase in smart devices that have increased the amount of information that is sent between the controller and the IEDs. The use of a fieldbus to transmit all this extra information has also introduced a new variable of network delays that also needs to be considered when designing a control system. The smart devices can also be from different vendors; therefore another problem is present when trying to communicate with different vendors using one controller from a different manufacturer. These issues are split into sub problems 1 and 2 to be listed and solved separately.

1.3.1 Sub-problem 1

Network delays in distributed control systems are non-deterministic and are therefore it is not possible to calculate or work out. It is necessary to model these delays when designing a system to show what effects these delays have before implementing the system in real-time. Once the effects of a range of time delays are known, the system can be optimized by changing software or hardware aspects to reduce the effects introduced by these delays. This will increase the stability of the controller and increase the system's reaction time to any form and length of network delay.

1.3.2 Sub-problem 2

The portability of software between vendors is a huge issue in automating control systems as switching from one software platform to another is often complex if not impossible. By using software packages that are IEC 61499 standard compliant, the PLC code can easily be transferred between different programming environments. This code includes library repositories and developed function blocks.

1.4 Research Aim and Objectives

1.4.1 Aim

The aim of this research is to design and implement a control system for a radio antenna's azimuth and altitude positional movements by using modern control design methods and utilizing the portability and reusability aspects of the IEC 61499 standard.

1.4.2 Objectives

The main objective of this research is the implementation of the developed control system for the radio antenna's DC motors in a real-time network control system. This can further be sub-divided into theoretical and practical objectives.

1.4.2.1 Objectives: Theoretical analysis

- To conduct a literature review in the area of IEC 61499 standard-based systems
- To conduct a literature review in the area of network control systems
- To investigate radio antennas and the DC motor in different applications
- To develop the formulation of the mathematical model for the plant
- To formulate the design of the controller using modern controller design methods (state-space)

1.4.2.2 Objectives: Practical implementation

- Simulation of the proposed system within the MATLAB/Simulink software environment.
- Analyzing the effects of network delays on the simulation model of the plant.
- Transforming the developed model from the MATLAB/Simulink to the TwinCAT 3 programming environment.
- Configuration of hardware devices for real-time implementation
- Real-time hardware-in-the-loop implementation of the transformed model.
- Testing and validation of the real-time implemented control system with the simulated results by comparing the system responses.
- Performance and conformance testing of the designed control system.

1.5 Research Questions

The research investigations in this thesis attempt to provide solutions to the following questions:

- Can modern control methods be used to achieve the desired response for the positional set points of the azimuth and altitude movements of a radio antenna?
- Can the network delays be modelled and simulated before implementation of the system?
- What length delays reduces the performance of the system?
- What methods can be used to account for the network delays in real-time?
- Is the developed controller interoperable, portable, configurable, and reusable between the different vendor automation software packages?

1.6 Hypothesis

The hypotheses for this research are as follows:

- State space techniques can be used to create a controller that meets the desired system response.
- Network delays can be modelled and analysed using transport delay function blocks in Simulink before real-time implementation.
- Functions within the TwinCAT 3 programming environment can be used to reduce the effects of network induced delays.
- The developed code will be able to port between different vendor’s software packages.

1.7 Delimitation of the research

The delimitations of this research are as follows:

- Literature reviews will only be done on the IEC 61499 standard and distributed network systems.
- State space methods will be used to design the controller
- Simulation of the controller and plant will be done in MATLAB/Simulink.
- Simulation of networked induced delays will be done in MATLAB/Simulink.
- Real-time implementation of the controller and plant will be done using TwinCAT programming environment and Beckhoff hardware solutions.

1.8 Assumptions of the project

- It is assumed that state space methods can be used to control the position of a DC motor as this modern control technique is used in control systems with feedback loops.
- It is assumed that DC motor can be modelled and simulated in MATLAB/Simulink as the programming environment is used for modelling and simulating control systems.
- It is assumed that the MATLAB/Simulink model can be transformed to TwinCAT 3 because both programming environments have IEC 61499 standard capabilities.
- It is assumed that the TwinCAT axis control will be able to change the position of the motor as well as read the feedback from the encoder as the programming environment allows for axis control of a wide range of motors.
- It is assumed that the network delays will effect the position output of the model in simulation and real-time implementation as a delay in transmission of information will cause a delay in receival of information.

1.9 Deliverables of the project

The deliverables for this research are as follows:

- Literature review on the IEC 61499 standard and networked control systems
- Mathematical modelling of the plant and controller
- Simulation of the model of the plant and controller
- Simulation and analysis on the effects of networked induced delays in the developed control system
- Transformation of the Simulink model to TwinCAT 3 programming environment
- Real-time implementation of the developed controller controlling the actual DC motor
- Comparative analysis of the simulated and implemented results

1.10 Chapter breakdown

This thesis has 8 chapters including an introduction, literature review, theory based on antenna and the DC motor, development and simulation of the plant, state feedback controller design, Transformation of Simulink model to TwinCAT object, implementation chapter, and a conclusion.

Chapter 1 presents the problem statement and the different research investigations that are done. The project statement includes the aims and objectives of the research, the statement of the problem including sub problems, hypotheses, delimitations in the research, project assumptions, and project deliverables.

Chapter 2 presents two separate literature reviews related to the IEC 61499 standard and networked control systems. The literature review on the IEC 61499 standard focus on the standard before and after the release of the second edition. Other factors such as execution methods, function block design, and portability are also described. The literature reviewed regarding networked control systems focuses on delays in systems with networks between the sensor and controller, and the controller and actuator.

Chapter 3 presents the plant considered in this thesis. The history of radio antennas, different types of antennas, current control systems for antennas, are all presented in this chapter. Due to radio antennas positional movements being controlled by DC motors, the history, operation, and construction of the DC motor is also described.

Chapter 4 describes the derivation of the mathematical models of the electrical and mechanical components of the DC motor. The transfer function of the complete electromechanical is described and simulated as an open loop system in MATLAB/Simulink. The open loop system is converted to its equivalent state space representation. The step response of the plant in state space form and the plant in transfer function form are simulated and compared to ensure that the conversion to state space is successful.

Chapter 5 describes the state feedback controller design. The pole placement design technique is used to move the poles of the system to allow for a faster response to position set point change. The gain matrix K is determined and used in the simulation of the closed loop system with state feedback control. Integral control is added to the system to reduce the steady state error caused by using a state feedback controller. The integral gain is calculated and added to the system for simulation. Four case studies are considered and compared using a step input to the system to simulate each controller’s step response.

- Case 1: Step response of a DC motor closed-loop system.
- Case 2: Step response of a DC motor closed-loop system with state feedback controller.
- Case 3: Step response of a DC motor closed-loop system with state feedback controller with added integral control.
- Case 4: Step response of a DC motor closed-loop system with state feedback controller with added integral control with a larger set point to show the system has the same response to bigger changes in position set point.

Networked induced delays are added to the system to analyse the effects on the system response. Two network delays are looked at including sensor-to-controller delays, and controller-to-actuator delays. Multiple lengths of these two delays are added to the system using a transport delay function block in Simulink, and the responses are compared in two separate case studies. Firstly, sensor-to-controller delays are considered in 4 cases with different delay times:

- Case 1: Position output with 100ms delay between sensor and controller
- Case 2: Position output with 400ms delay between sensor and controller
- Case 3: Position output with 800ms delay between sensor and controller
- Case 4: Position output with 1200ms delay between sensor and controller

Secondly, controller-to-actuator delays are considered in 4 cases with different delay time:

- Case 1: Position output with 50ms delay between controller and actuator
- Case 2: Position output with 100ms delay between controller and actuator
- Case 3: Position output with 125ms delay between controller and actuator
- Case 4: Position output with 150ms delay between controller and actuator

The chapter concludes with an analysis and discussion of each of the different cases.

Chapter 6 presents the methods used to transform the Simulink model to a TwinCAT object that can be used in the TwinCAT 3 programming environment. A description of each software package is provided including the installation procedures that are described in detail in the Appendix. How to sign Windows drivers to allow for the transformation is also described. A description of the Beckhoff TE1400 transformation tool is described, and the necessary MATLAB code that needs to be run to complete the transformation is provided. A test is done using Beckhoff's Scopeviewer software to prove that the transformation is successful.

Chapter 7 presents the hardware-in-the-loop implementation section of the thesis. All required hardware is described including the Beckhoff controller and motor terminals, the Omron encoder, and the DC motor. A description of the EtherCAT fieldbus technology is also described. A photo and an electrical diagram of the complete test rig is shown. The PLC software in TwinCAT 3, which uses standard programming software environment CodeSYS, and standard motion function block library PLCopen, is described step-by-step. The transformed Simulink model in function block form is described. How to scan for EtherCAT devices as well as how to activate the PLC project on the controller for real-time implementation is described. The complete control system is first analysed with the current networked induced delays caused by the EtherCAT fieldbus. Then, time delay compensation, which is a TwinCAT 3 feature, is added to the system to reduce these delays. A step response is used to test the response of the system without time delay compensation, and then with time delay compensation. The implementation results are compared with the simulation results to ensure the results are correct. Lastly, the developed control system limitations are discussed and implemented in real-time.

Chapter 8 presents the conclusion, deliverables of the thesis, results of the work completed, future work that could add on to this thesis, and the publications emanating from this research. The references and appendices follow Chapter 8.

1.11 Conclusion

This chapter presented the introduction to this thesis including the awareness of the problem, the problem statement, aims and objectives, a hypothesis, possible research questions, delineation of the research, assumptions and deliverables of the project, and a complete chapter breakdown of the research work.

The next chapter focuses on the literature review, which consists of a more in-depth look at the IEC 61499 standard as well as networked control systems.

CHAPTER 2
LITERATURE REVIEW

2.1 Literature Review on the IEC 61499 Standard

2.1.1 Introduction

The International Electrotechnical Commission (IEC) 61499 Standard is used in this thesis to design and implement a controller in a distributed environment. This literature review focuses on the fundamentals of the IEC 61499 Standard from the release of the first edition in 2002 to the current second edition which was released in 2012. This review also focuses on the numerous case studies done by researchers, looking at the advancements and results of their papers which contributed to the refinement of the first edition to the second edition of the IEC 61499 Standard.

Section 2.1.2 describes the keywords used as well as the sources searched to complete the literature review. Section 2.1.3 gives a summary of all the literature reviewed on the IEC 61499 Standard in a tabular form. Two tables are presented: the literature reviewed before the second edition of the standard was released, and the literature reviewed after the second edition of the standard was released. Section 2.1.4 gives a summary of the literature reviewed, focusing on distributed controller design, portability, execution methods, and modeling and verification when using the IEC 61499 Standard. Section 2.1.5 highlights comparisons between the literature reviewed, followed by a conclusion to the chapter in Section 2.1.6.

2.1.2 Literature Search

The main source of literature obtained is the IEEE Xplore database. The following keywords and phrases are used in searching this database: IEC 61499 Standard; distributed control systems; Designing IEC 61499 Standard systems; IEC 61499 Standard implementations; IEC 61131-3 to IEC 61499. A collection of papers regarding these keywords are read and further research papers are found from the references in these articles. References found led to various websites and articles from ResearchGate.

A total of 44 papers are reviewed using the above-mentioned keywords. Figure 2.1 illustrates a bar graph of the number of papers reviewed versus when the papers were published. The paper's publication dates range from 2004 - 2023. More papers were found before 2012 as the IEC 61499 Standard was still new and many authors had comments to make on the first edition. After the release of the second edition in 2012, the articles are more about the implementation of the standard and not about its semantics as all ambiguities in the first edition were corrected. A gap is shown for the years 2019 and 2020 which could be due to the pandemic that occurred globally. To the best of my knowledge, there weren't any articles related to this literature review that could be summarized and described. The years 2021-2023 shows an increase in literature regarding the IEC 61499 Standard with multiple articles being published.

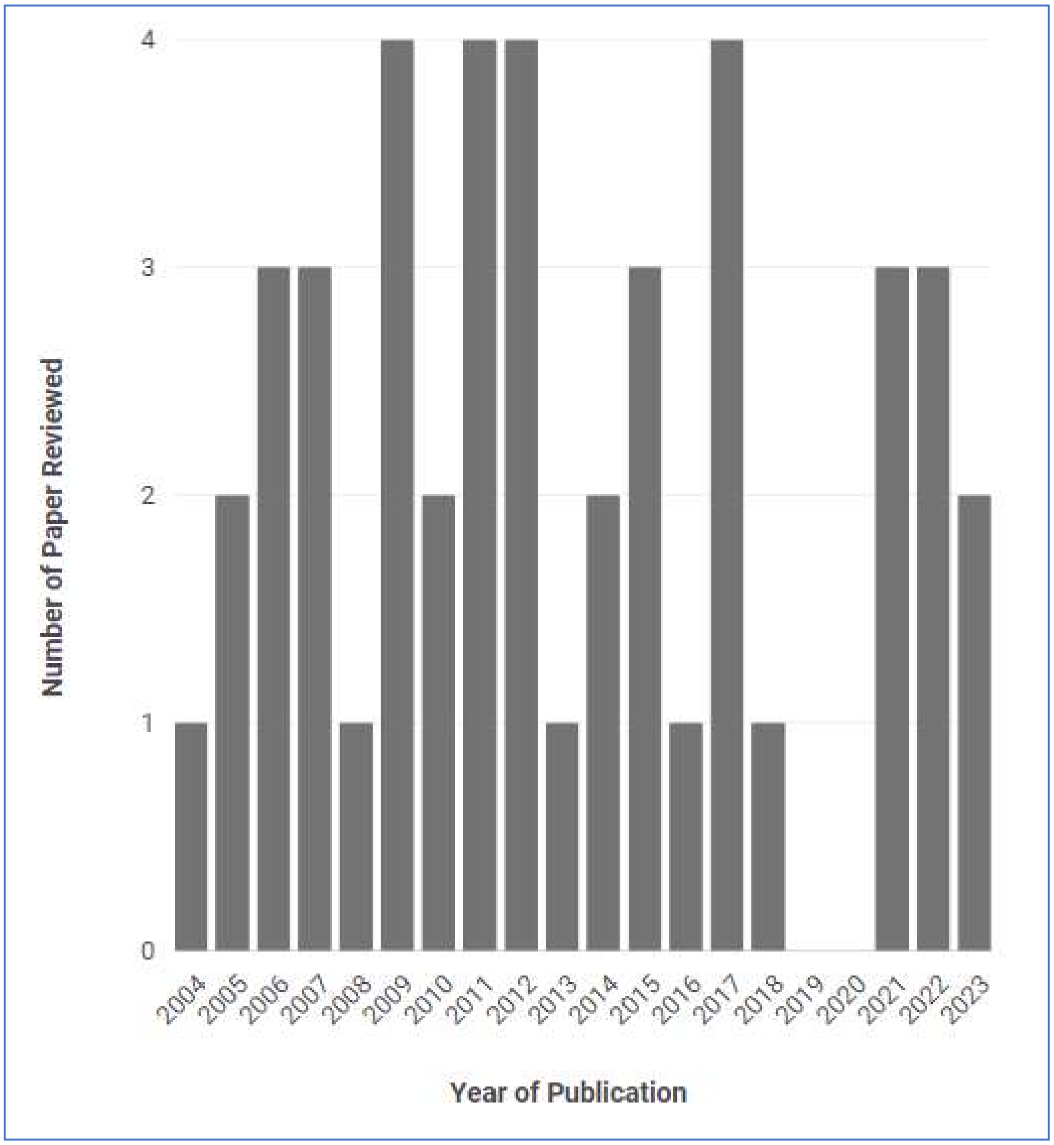


Figure 0.1: **Bar graph showing the number of publications, categorized by year published, reviewed on IEC 61499 Standard in distributed control systems**

The following subsection summarizes the reviewed articles into Table 2.1 and 2.2. The review summary describes articles written on the first and second editions of the IEC 61499 standard focusing on the aim of the project, the system overview, the hardware/software required, and the authors conclusions.

2.1.3 Literature Review Summary

Table 0.1: **Literature Review of IEC61499 Standard first edition**

Paper (Reference)	**Aim of the project**	**System Overview**	**Hardware/Software Required**	**Author's Conclusions**
Design of distributed control systems based on new international standards. (Koziorek, 2004).	To give an overview of a distributed control system design method that is based on the IEC 61499 and IEC 61131 standard. This method can be used to design a new control system or to adapt an existing system.	Investigative Paper	No specific hardware or software is used as the paper depicts a generic control system.	The paper added to the development phase of control systems. The processes described in this paper that are used to design and develop control system models would be demonstrated by the second half of 2004.
Functional design for IEC 61499 distributed control systems using Unified Modeling Language (UML) activity diagrams. (Panjaitan and Frey, 2005).	Use UML Activity diagrams to make designing a distributed control system easier and allow software components to be reusable. The UML diagrams must also increase the flexibility of system design to allow for easier reconfiguration and configuration of old and new systems.	Feeder Station	Function Block Development Kit (FBDK) from Rockwell used for the IEC 61499 function block creation. NETMASTER controller with a TINI microcontroller is used as the hardware.	By using UML activity diagrams, this paper achieved flexibility and reusability when designing distributed control systems. It is shown that it is easier to design and understand distributed control systems when using the IEC 61499 Standard as the creation of function blocks enables a plug-and-play design.
Intuitive control engineering for mechatronic components in distributed automation systems based on the reference model of IEC 61499 (Sunder et al., 2005).	To model a mechatronic component into a single autonomous part that can be used in a distributed control system. This allows the local control device to interact with the component through services while keeping the mechatronic component's control separate.	Control of a linear servo drive through IEC 61499	The hardware consists of a micro-controller and a linear servo drive. No specific software is mentioned but it does have IEC 61499 capabilities.	It is possible to control a linear servo drive through Motion service commands sent from the local controller. This keeps the mechatronic component separate from the main controller, allowing better distribution of control through the system. The mechatronic components model is also reusable if more linear servo drives are needed.
Systematic design and implementation of distributed controllers in industrial automation (Vyatkin et al., 2006).	Comparing different approaches to distributed system design in automation systems.	A distribution station consisting of a magazine and feeder unit is used to transport work pieces from storage	No specific hardware or software is mentioned but it should be IEC 61499 compliant	Function block applications are sufficient when used to implement distributed controllers.
Applying the IEC 61499 model to the shoe manufacturing sector (Colla et al., 2006).	Use IEC 61499 Standard to design an automation system for a plant that contributes to the preservation of shoe-manufacturing activities in Europe.	A manufacturing system that moves semi-finished shoes over innovative	MATrix LABoratory (MATLAB) environment for simulations Custom tool made with Eclipse to import and	The development phase is faster due to IEC 61499 using modular function blocks that can be used reused.

		transport lines from one station to another on a predefined operation schedule.	convert Real-time Unified Modeling Language (RT-UML). specification to IEC 61499 model.	As systems get more complicated, the design approach is not usually identified immediately. Depending on the software used for developing IEC 61499 code, take into consideration the method used for executing control code as this could generate undesired effects.
Formal Modeling of Function Block Applications Running in IEC 61499 Execution Runtime *(*Goran Čengić et al., 2006).	To prove that different logical behaviour of function blocks is shown when different execution methods are used.	Carriage transport system for work pieces.	Simulation and function block application execution by free software Fiber. supports IEC 61499 runtime.	Using different execution run time environments, which use different execution methods, led to the different behaviour of function blocks.
Deployment of IEC 61499 compliant distributed control applications (Hussain and Frey, 2007).	To find feasible and optimal deployment methods for distributed control systems that are IEC 61499 compliant.	The proposed concepts are carried out on two separate systems. 1) FESTO Modular Production System (MPS) didactic model. 2) Lift control system.	No specific hardware or software is mentioned but should be IEC 61499 compliant.	Worst-case scenarios are taken into consideration when doing the feasibility analysis which showed results that are not as feasible as predicted. The algorithm will continue to be tested in future work.
Design and implementation of heterogeneous distributed controllers according to the IEC 61499 Standard - a case study (Hirsch et al., 2007).	Present solutions to re-usability and integration between heterogeneous controllers in a distributed control system Illustrate how to integrate controllers using the IEC 61499 standard into systems previously based on Programmable Logic Controller (PLC) distributed control.	Testbed built up of 4 different stations: distribution, testing, drilling, handling.	Various hardware modules are used to show the hardware independency of the IEC 61499 Standard. 1) Netmaster 2nd series device based on SNAP Java microprocessor. 2) Siemens Simatic S7 PLC. 3) W2-FBC with SNAP microprocessor. FBDK used to design and implement controllers and visualization.	Integration of IEC 61499 standard systems into existing IEC 61131 systems are inevitable in the future as distributed control becomes the norm. The integration and development of these systems and new control technologies do not require as much effort as anticipated by the authors. The modularity of the design allows for the results of smaller systems to be scaled to bigger and more complex industrial systems.
Hierarchical distributed controllers - design and verification (Missal et al., 2007).	Use a multi-layered architecture to design and implement the IEC 61499 Standard distributed control system to enhance the reusability of controllers.	Testbed used to drill holes in work pieces and monitor if the drilled holes	No specific hardware or software is mentioned but should be IEC 61499 compliant.	Designing controllers based on modular and hierarchical design allows for fast redesign and formal verification of distributed control systems.

		are correct. The system is split into multiple modules: table, drilling, and testing.	SESA software tool is used for closed-loop verification.	
A control software development method using IEC 61499 function blocks, simulation, and formal verification (Goran Čengić and Knut Åkesson, 2008).	To use control software to simulate a system without using the real hardware.	The ball sorting process used to sort steel balls through a system of lifts and distributed to different levels based on weight.	Simulation and function block application execution by free software Fiber. supports IEC 61499 runtime.	The control software described allows the development and simulation of the entire system to be tested before deployment to real hardware. This approach leads to shorter development time, and it makes the control system less prone to incorrect behaviour which could destroy equipment in a real process.
A synchronous approach for IEC 61499 function block implementation (Yoong et al., 2009).	To show the feasibility and ease of verification when using a synchronous approach for the development of IEC 61499 function blocks.	Cruise control system.	Esteral Studio and V7 Esteral compiler are used for software. FBDK is used on a test bench to compare the Esteral results with No specific hardware is mentioned but should be IEC 61499 compliant.	There is no need for a runtime environment when using a synchronous approach. Execution speed is increased due to no runtime being used as all decisions are made during compile-time.
The IEC 61499 Standard and its semantics (Vyatkin, 2009).	Investigation of the semantics of IEC 61499 Standard, code portability, as well as function blocks and function block networks.	Pneumatic cylinder test bed with two modes of operation. The modes are linked to two set positions for the pneumatic cylinder to move to as a reference. Safety curtains are also added to the system to prevent movement when triggered	A Central Processing Unit (CPU) that is IEC 61499 compliant with peripheral I/O (Input/Output) including a start and mode button, as well as a 7-segment (Light Emitting Diode) LED display to show which mode is active. ISaGRAF used for function block development	By defining a limited number of models for function block network execution, the investigation has progressed the portability of function block applications in distributed control systems. These differences in execution models can be seen as future research topics.
Closed-loop modeling in future automation system	Describe new methodologies for validating and designing distributed control systems.	Control of a storage system	Floating Point Benchmark (FBench) used to model the	The work done contributes to the grand challenge of completely automating the design and validation process by enabling correct-by-construction

engineering and validation (Vyatkin et al., 2009).	Describe how these techniques are made possible by advanced technologies and the IEC 61499 Standard.		mechatronic system into function blocks. Plant models are created in Visual NCES Editor (ViEd) Visual Verifier (Vive) used to check and analyze models. Simulink used to apply graphical models	models. There is still lots of work to be done to solve the grand challenge, but the author states they will continue their efforts and recommends other researchers to help
Design and implementation of LabVIEW-based IEC 61499 compliant device (Polaków, 2009)	To develop a run-time environment that can be uploaded to a PAC to convert the device to a compliant IEC 61499 device.	Simulation of event generation and dispatching.	The run-time environment is developed in the LabVIEW platform.	Full functionality of the system is achievable theoretically, but the run-time is currently in the development phase
Improving verification and reliability of distributed control systems design according to IEC 61499 (Lapp et al, 2010)	Use Net Condition/Event Systems (NCES) as formal models to improve the verification and reliability of distributed control systems.	Investigative Paper	No specific hardware or software is mentioned but should be IEC 61499 compliant.	By using the approach in this paper to model systems, there is an increase in validity and reliability, which allows the use of IEC 61499 to be used in engineering practices.
Intelligent component-based automation of baggage handling systems with IEC 61499 (Black and Vyatkin, 2010)	Create a decentralized control system using a multi-agent approach for a Baggage Handling System (BHS) using IEC 61499 function blocks.	Baggage handling system, concentrating on creating a single component function block for a conveyor. Testbed using FESTO MPS500 used for final testing	FBDK used for function block design. Netmaster controller specified hardware	Creating these autonomous conveyor sections has led to the easier setup of reconfigurable BHS systems. This system performs the same function as the previous centralized control application but is instead distributed over multiple embedded intelligent devices. Holonic control is achieved which allows for easier setup when there is an environmental change such as layout changes due to baggage flow increase
IEC 61499 Function Block Model - Facts and Fallacies (Thramboulidis, 2010)	Look at the semantics of IEC 61499 function blocks to discuss facts and fallacies to determine why the standard has not been accepted by the industry yet.	Review paper illustrating and analysing multiple case studies	Multiple IEC 61499 compliant hardware and software presented.	Many issues are investigated which led to the conclusion that a major revision of the standard Is needed to allow it to be considered for industrial use. The standard does not apply certain basic engineering practices, which has led to many ambiguities during the initial phase of development.
IEC 61499 as an enabler of distributed and intelligent automation: a state-of-the-art review	Review the current work done and in progress regarding the IEC 61499 Standard in distributed control systems.	Review paper illustrating and analysing multiple case studies.	Multiple IEC 61499 compliant hardware and software is looked presented.	IEC 61499 has become more popular due to the increase in smart devices causing more distributed systems. The standard is being used more with other standards such as IEC 61850 and IEC

(Vyatkin, 2011)				62424 which is enabling a means where the control system is automatically generated by importing design documentation into specific software. This decreases the development time drastically. The IEC 61499 has added great amounts of knowledge regarding distributed control systems that are not previously satisfied by IEC 61131-3.
Distributed execution and cyber-physical design of baggage handling automation with IEC 61499 (Yan and Vyatkin, 2011)	To reduce the design and validation steps by developing efficient design methods for BHS. This results in improved robustness and adaptability in these systems. Also, to prove that designing a fully distributed system, which uses IEC 61499 function blocks, is feasible.	Baggage handling system, consisting of numerous conveyors connected via an Ethernet network of 50 control nodes	48 Netburner devices (embedded controllers) Controller Operating System: μC/OS-II Real-Time Controller Runtime: ISaGRAF runtime	Feasibility is confirmed for implementing BHS control in a distributed system. By using a cyber-physical method, the goal of simple reconfiguration of BHS systems is possible. Future work will look at the automatic generation of function blocks through visual techniques to allow system designers to easily create new BHS systems.
Design and execution issues in IEC 61499 distributed automation and control systems (Strasser et al., 2011)	To discuss and present different execution models of function blocks, Composite Function Blocks (CFBs), and sub-applications, in IEC 61499 devices.	Investigative Paper	No specific hardware or software is mentioned but should be IEC 61499 compliant.	This paper contributes to the ongoing goal of achieving an IEC 61499 distributed control system that is portable, configurable, interoperable, and distributed.
On the use of model-based IEC 61499 controller design (Preuße et al., 2011)	Review existing ways of modeling and verification of IEC 61499 compliant function blocks. State further challenges for formal technique developments of IEC 61499.	Review paper illustrating and analysing multiple case studies	Multiple IEC 61499 compliant hardware and software presented	It is shown that IEC 61499 has reached a point whereby it is realistic to apply in real-life industrial applications. The problem is that there is no pressure on vendors to take advantage of these technologies to develop new control systems. This could lead to all the research being stuck in limbo, only being displayed on test benches and not in the field. Companies must also be willing to migrate from IEC 61131-3 PLC-based automation to new technologies. This will probably be a phased transition, resulting in many heterogeneous systems.
Redesign distributed PLC control using IEC 61499 function blocks (Dai and Vyatkin, 2012)	Propose methodologies for the conversion of PLC control (IEC 61131-3) to event-driven control which uses IEC 61499 function blocks.	Airport baggage handling system	NXTStudio development environment used to create the function blocks.	Distributed control logic is not easily implemented when designing IEC 61131-3 PLC systems. Three approaches are looked at when considering migration from IEC 61131-3 to IEC 61499 function blocks: object-oriented reuse, object-oriented

			No specific hardware is mentioned but should be IEC 61499 compliant.	conversion, and class-oriented approach. Object-oriented conversion: used when a state machine is used for the original code. Object-oriented reuse: used when state machine code isn't easily recovered. Class-oriented approach: use when converting data-intense systems. Reuse of IE C61131-3 code is suggested when migrating to IEC 61499 as this accelerates the transition process.
Distributed control design of medical devices using plug-and-play IEC 61499 function blocks (Sorouri et al., 2012)	Reducing the complexity of current medical devices by using IEC 61499 Standard function blocks to apply distributed control architectures.	A robot to assist people who have lost the ability to move certain limbs due to having a stroke.	Control system design in NXTStudio IDE using IEC 61499 Standard function blocks. Beckhoff CX101 controller used for deployment	Using the IEC 61499 Standard architecture reduced the development time and complexity by using plug-and-play software components. The proposed approach of this simple system can be used for fast implementation on more complex systems.

Table 0.2: **Literature review of IEC61499 Standard second edition**

Paper (Reference)	Aim of the project	System Overview	Hardware/Software Required	Author's Conclusions
The IEC 61499 Function Block Standard: Overview of the second edition (Christensen et al., 2012)	Presents improvements of the IEC 61499 Standard to its second edition which will be released in late 2012.	Investigate Paper	No hardware or software is presented as the entire IEC 61499 Standard is looked at	After listing all the new additions that form the IEC 61499 standard second edition, it is said stated that the standard will be refined to be more clear, unambiguous, and industrially useful. The conclusion is that all vendors should seriously consider developing software tools, runtime platforms, or control hardware so that they can enter a growing market.
The IEC 61499 Function Block Standard: Launch and Takeoff (Strasser et al., 2012)	Discuss the start of the take-off phase of the IEC 61499 Standard. Look at processes that will lead to a successful take-off of the standard.	Investigate Paper	No hardware or software is presented as the entire IEC 61499 Standard is looked at	IEC 61499 take-off phase has started. The standard can serve a great payoff if early adapters focus on addressing complaint profiles and trained personnel as soon as possible.
Virtual Smart Metering in Automation and Simulation of Energy-efficient Lighting System (Pang et al., 2013)	Use function blocks based on IEC 61499 Standard to create virtual smart metering systems that are easily reusable to reduce laborious work when creating building automation and control systems.	Prototype lighting control system to simulate a virtual smart metering system that measures and	SCADA developed in NXTStudio Digital Addressable Lighting Interface (DALI) protocol is used for communication.	The research found that intelligent lighting control, compared to normal lighting control, has much better energy efficiency.

		monitors energy efficiency in a building	IEC 61499 Standard used for function blocks of the DALI Light	
A Portability Study of IEC 61499: Semantics and Tools (Pang et al., 2014)	A study of portability issues in IEC 61499 tools such as FBDK, 4DIAC, ISaGRAF and NXTStudio. The main outline is with the compatibility issues because of different execution semantics.	Investigative Paper	FBDK, 4DIAC, ISaGRAF and NXTStudio are all compared and tested for compatibility	IEC 61499 second edition has solved many of the execution issues since its release. Full portability and interoperability can only be reached once it is possible to formally analyze and validate against runtime platforms.
Cyber-physical Components for Heterogeneous Modeling, Validation, and Implementation of Smart Grid Intelligence (Zhabelova et al., 2014)	To show that using Cyber-physical systems can improve design, validation, and verification in smart grid automation.	Load balancing cyber-physical system	A distributed network is modeled with MATLAB. NXTStudio used for IEC 61499 environment. Cyber-physical systems executed on an ARM board	Load balancing test application shows that IEC 61499 allows for the execution of Cyber-physical systems on industrial hardware applications.
Bridging Service-oriented Architecture and IEC 61499 for Flexibility and Interoperability (Dai et al., 2015)	Describe a method of modeling that uses SOAs in a distributed control system	Section of a baggage handling system where a single infeed conveyor is split into two screening lines	BHS emulator - Glidepath Group FBSRT used for the function block service runtime. Beaglebone Black board used for hardware, with specifications AM335x 1Ghz CPU, 512M DDR3 Random Access Memory (RAM), and 4GB Read-only Memory (ROM). FORTE runtime used as a test reference running a similar implementation.	Using SOAs with the IEC 61499 Standard allows the ability to reconfigure, update or delete FB instances without stopping the system's normal execution runtime.
Formal Verification of IEC61499 Function with Abstract State Machines and SMV - modeling (Patil et al., 2015)	Use abstract state machines to propose rules that can be used to formally model IEC 61499 function blocks	A generic basic function block is used for modeling	FB developed in ISaGRAF No hardware as paper focuses on simulation only	A formal model of IEC61499 is presented and proved that it can be used to verify and simulate using SMV.
Formal Verification of IEC61499 Function with Abstract State Machines	Present the IEC 61499 Standards execution semantics ambiguities as well	Investigative Paper	No hardware or software is presented as the	Presented the model checking for an industrial automation control system.

and SMV - Execution Semantics (Dubini and Vyatkin, 2015)	as describe the rules for SMV model transformation.		entire IEC 61499 Standard is looked at	Demonstrated how their auto generator can support different execution semantics
Complementing Testing of IEC61499 Function Blocks with Model-checking (Glatz et al., 2016)	Use Uppaal model checker to prove the usefulness of an automated translation approach to generate models for IEC 61499 function blocks.	A segment of a building that has an automated system	No specific hardware or software is mentioned but should be IEC 61499 compliant.	The presented approach to automate IEC 61499 function blocks to Uppaal processes, to allow testing with model checking, helps improve verification of these systems.
Open Architecture for Cost-Effective Protection and Control of Power Distribution (Zhabelova et al., 2017)	Demonstrate an execution platform for developing protection functions using open standards. Test the functions and the reaction times. Investigate verification methods for the functions as well as how to do remote upgrades.	Investigate Paper	No hardware or software is presented as the entire IEC 61499 Standard is looked at	
Distributed Home Automation System Based on IEC61499 Function Blocks and Wireless Sensor Networks (Abrishambaf et al., 2017)	Design and implement a fully distributed wireless sensor network with IEC 61499 function blocks as the architecture.	Building automation project which measures the temperature and humidity in various rooms and sends the data to the cloud	Systems simulated in Cooja platform of Contiki-OS TelosB platforms used for implementation. NXTStudio used for function block layout and HMI development. Ubidots used as a cloud service to capture and post data to the Internet	A fully distributed system is developed and implemented to show the flexibility and reconfigurability of using the IEC61499 Standard for function blocks
Estimation, Measurement, and Improvement of Distributed Automation Applications Performance (Väänänen and Vyatkin, 2017)	Determine the performance of IEC 61499 Standard distributed systems to add to the research field of improving end-to-end response times of said systems.	Function block system to determine time delays in I/O latching	4DIAC WCET analysis tool to determine response times. 4DIAC IDE to develop function blocks. No hardware specified but should be 61499 Standard compliant	A mathematical model is presented to determine the performance of distributed automation systems. More research should be done to determine end-to-end response times of systems to improve overall performance
Toward self-Manageable and Adaptive Industrial Cyber-Physical Systems with Knowledge-driven Autonomic Service Management (Dai et al., 2017)	Integrate service-oriented architecture with an automatic service manager in an IEC 61499 Standard distributed control system.	Baggage Handling system	BHS simulator is used for the simulation of the control system. SQWRL language is used to set rules for automatic service managers.	Interoperability and flexibility are achieved when using service-based agents in service-oriented execution environments at the device level. Self-optimization as a self-management feature can improve efficiency when trying to utilize resources.

			No hardware specified but should be 61499 Standard compliant.	
Refactoring of IEC 61499 Function Block Application – A Case Study (Patil et al., 2018)	Propose methods to refactor existing function blocks to increase readability, maintainability, reuse-ability, and debugging.	Festo didactics' distribution station code used	NXTStudio development environment used to create the function blocks. No specific hardware is mentioned but should be IEC 61499 compliant	The paper proves that refactoring is possible on IEC 61499 function blocks. The refactoring methods presented contribute to the field that is not yet extensively researched and used in function block development.
Multi-Agent Modeling of Cyber-Physical Systems for IEC 61499 Based Distributed Automation (Lyu et al., 2021)	Model industrial cyber-physical systems using a two-layer architecture: high level uses a multi-agent computer model, whereas low level uses the IEC 61499 Standard function block model.	Generic System to show the functions of the two-layer architecture	Smart Python Agent Development Environment (SPADE) is used to develop multi-agent models. Eclipse 4diac used to develop the IEC 61499 Standard function blocks. The agents are spread out between a Jetson Nano, Raspberry Pi microcontrollers	Future work needs to be done for both layers in the architecture proposed. Multi-agent modelling requires the development of self-learning and machine learning capabilities. IEC 61499 Standard function block modelling requires an easier method of deploying control applications to microcontrollers such as the Raspberry Pi
Simulation and Control of a Cyber-Physical System under IEC 61499 Standard (Santos and da Silva, 2021)	Use a low-cost device to implement a modular control system.	Festo conveyor project kit	Eclipse 4diac used to develop the IEC 61499 Standard function blocks. Raspberry Pi used as hardware for runtime. Siemens s7-1200 used as plant controller for conveyor	The method used to validate and simulate a cyber-physical system is proven to be successful as a single element or in a network combination
Towards IEC 61499-Based Distributed Intelligent Automation: A Literature Review (Lyu and Brennan, 2021)	Compose a detailed literature review of the IEC 61499 Standard.	Review paper illustrating and analysing multiple case studies	Multiple IEC 61499 compliant hardware and software presented	Detailed literature review sectioned into the three main issues with the IEC 61499 Standard. These issues include how to transition code from IEC 61131-3 to IEC 61499, how the standard has enabled distributed control in industrial systems, and how to implement the standard in engineering environments.
Design of Integrated Energy System Based on	To showcase a method using the IEC 61499 Standard and OPC UA communications to apply cyber-physical	A photovoltaic simulated system connected through	Simulation of the photovoltaic system is	The simulation results show that the integration with OPC UA is successfully applied on the

IEC 61499 and OPC UA (Wang et al., 2022)	system technology in Internet energy systems. The proposed design is modular to allow for reconfigurability, interoperability and reusability.	a gateway to an electricity meter for monitoring.	done in MATLAB/Simulink. Hardware consists of a Raspberry Pi equipped with an embedded Linux operating system. The programming software used to create the function blocks is 4DIAC.	photovoltaic system to communicate with the IEC 61499 function blocks. The author predicts that component-based and modular systems will be the new trend in Internet energy systems by using the IEC 61499 Standard as well as the OPC UA technologies. Future work involves validating the proposed approach as well is applying it to more complex real-world scenarios.
Validating Effect of Refactoring of IEC 61499 Function Block in Distributed Control Systems (Cruz et al., 2022)	Propose methods to refactor IEC 61499 Standard function blocks in an existing material handling system to allow for easier portability and reusability. These methods allow for easier generation of code for automated systems.	Material handling system	Simulation model of the material handling system is created using NXTStudio and designed and controlled using IEC 61499 Standard function blocks.	By refactoring existing function blocks, it is easier to reuse code when creating new projects. The authors conclude that they recommend distributed control design when creating automation platforms and optimizing existing platforms by refactoring existing designs.
Model-aware Simulation of IEC 61499 Designs (Mehlhop and Walter, 2022)	To extract information from IEC 61499 Standard simulated models to analyse system events for verification. This is done by transforming models into SystemC models that can be analysed by third party software for debugging.	D-flip-flop model created with IEC 61499 Standard function blocks	4Diac is used to create the initial IEC 61499 Standard function blocks. SystemC is used for debugging the extracted transform 4DIAC function blocks. FORTE is used as the runtime platform.	The authors methods proved feasible through trials completed on the D-flip-flop system. The function blocks are successfully extracted and transformed to SystemC for analysing of the events. The ability to follow traces of events between function blocks is proven insightful. Future work looks to increase the accuracy of the time model, as well as develop more rigorous test cases.
Structuring Cyber-Physical Systems for Distributed Control with IEC 61499 Standard (Cruz et al., 2023)	To propose a cyber-physical model that reduces the complexity of control software by distributing the software over numerous small devices. The IEC 61499-based model is also validated and tested using a case study.	Material handling system consisting of 3 cylinders controlled by separate controllers that have IEC 61499 Standard capabilities.	NXTStudio IDE is used for development of IEC 61499 Standard function blocks and communication links. Distributed Control System (DCS) mini Schneider controller used as the hardware platform for the deployment of the created function blocks and software.	The proposed methods are proven to be more modular, reusable, and adaptable than the usual requirements for such systems. This is done by doing simple extensions and adaptations that help with reusing the models. Future works looks to add the communication protocol OPC UA to the models to allow for information to be sent to the cloud.

Automatic Information Model Generation for Industrial Edge Applications Based on IEC 61499 and OPC UA (Dai et al., 2023)	The aim of the authors is to present an automated generation model to help reduce the development time when creating information systems that are based on the IEC 61499 Standard. The use of OPC UA architectures is present in the case study used to test the proposed models when transforming between design time and runtime.	A white-body welding simulation line used by an automotive company is used as a case study.	FB Builder is used to create the IEC 61499 Standard function blocks, and FBSRT is used for the runtime.	The case study resulted in an increase in efficiency, flexibility, and interoperability in design time and runtime when automatically generating the information models. The improvement in efficiency shows a significant reduction in time when combining control and information models. Future work consists of improvements in the automatic generation process.

2.1.4 Literature review based on the IEC 61499 Standard

2.1.4.1 Distributed controller design based on the IEC 61499 Standard

Control systems are becoming more distributed due to an increase in smart devices in the industry. Before, all control was done in a centralized PLC using the IEC 61131-3 standard as all the peripheral I/O devices did not have any or had very little intelligence. The IEC 61131-3 standard was not designed for distributed control, and with the increase in the distribution of control devices, the standard is subject to problems with performance, spatial distribution, and ease of integration and reuse (Vyatkin et al., 2006). According to (Dai and Vyatkin, 2012), this traditional way of designing automation control systems with PLCs using the IEC 61131-3 standard is becoming a serious bottleneck when attempting to implement decentralized control.

One of the main advantages of the IEC 61499 Standard is that it allows for distributed controller design. Distributed control allows for a decrease in hardware cost while simultaneously increasing communication power (Thramboulidis, 2010). These systems also require less wiring, have more fault diagnostic information, and allow distributed components to be autonomous. These systems are also less complex compared to centralized control, as the functionality is spread over many sub-processing units (Hirsch et al, 2007). (Hirsch et al, 2007) mentions three different approaches to controller design: Centralized approach, distributed approach with master controller, and pure distributed approach. IEC 61499 Standard fits best with the pure distributed approach as the flexibility and reusability nature of the function blocks allow for control to be easily distributed. (Vyatkin, 2006) suggests a layered approach to distributed controller design as shown in Figure 2.2. The sensors/actuators layer consists of the direct interface with the plant. The operation layer consists of the operations defined to control the interfaced components of the plant. Lastly, the applications layer consists of intelligent code developed on decentralized controllers.

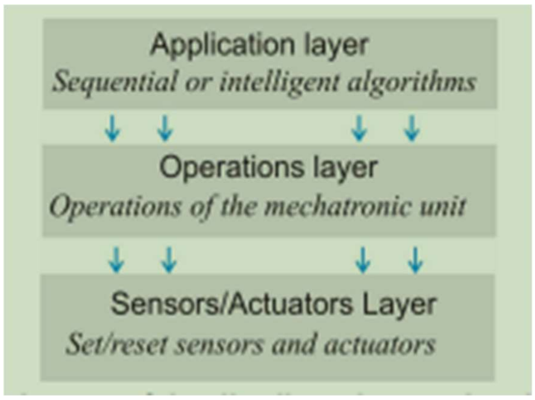

Application layer
Sequential or intelligent algorithms
Operations layer
Operations of the mechatronic unit
Sensors/Actuators Layer
Set/reset sensors and actuators

Figure 0.2: **Layered approach to distributed controller design (Vyatkin, 2006)**

(Missal et al, 2007) approaches distributed controller design by splitting the controllers in multiple master controllers. The result is a composition of distributed controllers over a layered system as shown in Figure 2.3. The physical layer is either a model of a plant or the plant itself. The task layer consists of controllers that only control the specific plant that it is linked to. These controllers do not interact with the rest of the system and therefore work independently. The coordination layer consists of the master controllers that use communication mediums to control the distributed controllers in the task layer.

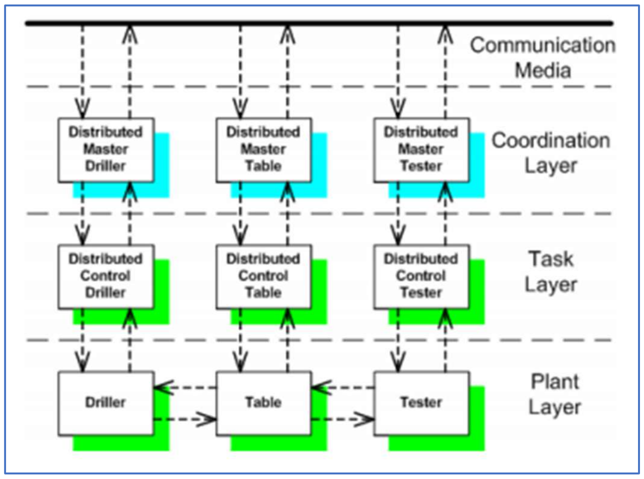

Communication Media
Coordination Layer
Task Layer
Plant Layer
Distributed Master Driller
Distributed Master Table
Distributed Master Tester
Distributed Control Driller
Distributed Control Table
Distributed Control Tester
Driller
Table
Tester

Figure 0.3: **Information flow between a multi-layered networked architecture (Missal et al, 2007)**

Communication between the different layers in any of this distributed system is done through events. The following subsection describes the different execution methods in which these events take place.

2.1.4.2 Execution methods

Before the release of the IEC 61499 second edition, there were great ambiguities regarding the execution methods of function blocks when using this standard. This led to runtime environments from different software developers taking different approaches to how the execution of events should be handled (Lapp et al., 2010). With there being no real specification of event execution, all the developed execution models became compliant.

There are three main execution models of IEC 61499 Standard function blocks: sequential, cyclic, and synchronous. Each of these methods has a different way of handling simultaneous input events, as the IEC 61499 Standard syntax does not allow for events to invoke the function block at the same time (Vyatkin, 2011). Sequential execution of events is the method of executing the events in a preserved order from when it invoked the function block (Vyatkin, 2009). The stored events occur on a First-In-First-Out (FIFO) basis and are only active once the previous event has been processed by the function block (Preuße et al., 2011). The cyclic execution model is like the IEC 61131-3 standard in which events cyclically invoke the function blocks (Vyatkin, 2009). Synchronous execution uses global events called ticks to align the execution speeds with a time constant. It is like cyclic execution as it moves away from event-driven to a more event-scan-based control (Preuße et al., 2011). The issue with having different execution methods is that portability between software tools becomes almost impossible. It also influences the case studies performed by researchers as different results are shown depending on which software tools are used as execution methods are different (Čengić et al., 2006). Certain researchers perform their experiments according to more than one execution method just to prove their results conform to the standard (Hussain and Frey, 2007).

2.1.4.3 Portability

In the automation industry, the need to migrate code from one hardware platform to another arises very often (Vyatkin, 2009). With the IEC 61131-3 standard, this is usually a difficult and complex task, and the code is not automatically read in another software platform without adjusting or copying the code line by line. This is normally due to semantics and different syntax between software platforms. The IEC 61499 Standard aims to move away from heterogeneous systems as these systems, consisting of devices from multiple vendors which can become time-consuming and are more prone to errors (Goran Čengić and Knut Åkesson, 2008). Function block systems that are developed using the IEC 61499 Standard should be reusable across all IEC 61499 compliant devices. This is one of the major requirements of the standard and allows for ease of portability, interoperability, configurability, and reusability. Figure 2.4 below shows the relationship between these requirements in a distributed control system.

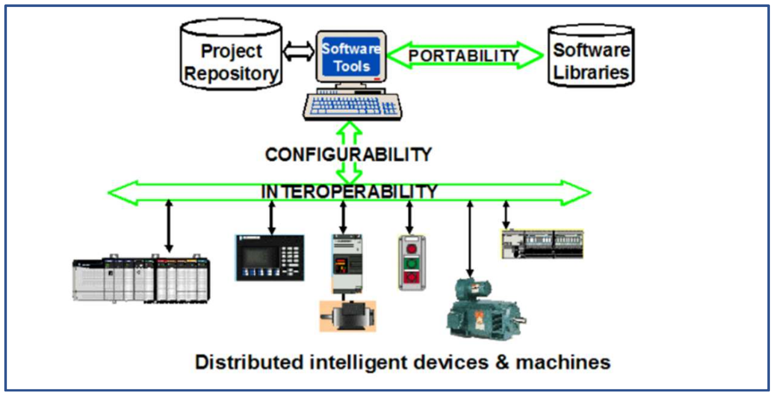


Figure 0.4: **Flow chart showing portability, configurability, and interoperability in a distributed control system (http://www.holobloc.com/papers/iec61499/overview.htm)**

Software is portable when it can be interpreted by multiple different software tools (Strasser et al., 2011). The IEC 61499 Standard eases portability as all the code is encapsulated in function blocks. This makes the code independent from the event command sequences outside of the blocks (Vyatkin, 2011). The configurability of a device depends on if its software components can be configured by other software tools from various vendors (Strasser et al., 2010). Interoperability in distributed control systems is when communication between the embedded devices allows them to perform functions together for distributed applications. Lastly, reusability is the ability to use function blocks in different systems under a different context (Missal et al., 2007). Reusability is achieved by standardizing inputs, outputs, and means of communication of a function block (Hirsch et al., 2007).

2.1.4.3 Modeling and verification

Control systems should be verified before being connected to a plant (Lapp et al, 2010). This is done by modeling the system using various techniques available. Examples of a few formal modeling techniques include Event System, Finite-State Machine, Discrete-Time/Event approach (DT/DE), Timed Net Condition, Time Automata, Petri next, etc. (Santos and da Silva, 2021). Using these methods to formally verify a control system is advantageous as the performance of the system is quantified by estimating end-to-end response times, through simulations, before deployment (Dai et al., 2017).

Formal verification is the most efficient way to prove a system is correct as the given specifications are used to verify the algorithms in the control code (Patil et al., 2015). State-space generation helps identify failures in a system that may prevent the system from being considered safe. Dynamic verification, on the other hand, focuses on monitoring and unit testing only certain devices in a system (Glatz et al., 2016). This form of verification is less time-consuming but could put the system at risk if certain bugs are not found due to multiple devices not being modeled and verified. Verification of IEC 61499 Standard control systems is very important as there is a difference in execution semantics depending on which vendor is used. These systems should be tested for interoperability and portability in different environments to prove or disprove behavior for verification (Patil et al., 2015).

To verify an IEC 61499 distributed control system, it is necessary to convert all devices, resources, Service Interface Blocks (SIFBs), and scheduling functions into formal models (Lapp et al, 2010). By modeling all components of the distributed system, verification and validation are ensured as code is tested, debugged, and simulated. This modeling process is often challenging as all aspects of the system need to be considered, including all control nodes and variable communication times (Zhabelova et al., 2014). Although challenging, modeling needs to be done to ensure the correctness of the control system.

2.1.4.4 The IEC 61499 Standard second edition

The IEC 61499 Standard was updated from the first to the second edition in 2012. The new edition comes after 120 editorials and 40 technical comments were written in response to the first edition's semantics (Christensen et al., 2012). The first edition had many ambiguities which confused developers and programmers who were using the standard. (Thramboulidis, 2010) found the learning curve very steep and mentioned that there were not enough reference implementations to show the advantages of the technology, therefore stating that a major revision should seriously be considered.

(Vyatkin, 2011) stated that by 2011 the use of the IEC 61499 Standard was very low in the industry due to "the lack of mature engineering tools, reliable embedded control hardware, proven design methodologies, and trained engineers". (Vyatkin, 2011) also said that the standard does not accompany the arrival of multiple events after one another, which could lead to a loss of event inputs.

The second edition refined how event inputs to function blocks should be executed. (Christensen et al., 2012) explains that the way around event execution is to prevent the resource from sending more than one event to the input of a function block at the same time. The IEC 61499 Standard systems' interaction with PLCs, the addition of temporary variables in function blocks, as well as the simplification of 'READ' and 'WRITE' commands, are all factors that are changed in the second edition due to the editorial and technical comments made since the release of the first edition (Christensen et al., 2012).

Subsection 2.1.4 summarized and discussed the literature reviewed on the IEC 61499 standard. The next subsection compares the similarities and differences between the articles reviewed.

2.1.5 Comparisons between literature reviewed on the IEC 61499 Standard

After the release of the IEC 61499 Standard first edition, there were many articles written on how to design distributed systems based on the standard. Earlier published articles were more investigative papers, not having many practical examples or case studies. As shown in Table 2.3, Articles 1 and 2 have no hardware or software required as these papers were more theoretical than practical. When looking at more recent articles, such as Article 3, which was published in 2021, a case study is designed and implemented with the specified software and hardware. This shows the improvement in the development of IEC 61499 compliant software tools in the short period from 2004 to 2021.

Table 0.3: **Comparison of papers published on IEC 61499 Standard distributed control systems**

No.	Paper (Reference)	Aim of the project	System Overview	Hardware/Software Required	Author's Conclusions
1	Design of distributed control systems based on new international standards (Koziorek, 2004)	To give an overview of a distributed control system design method that is based on the IEC 61499 and IEC 61131 standards. This method can be used to design a new control system or to adapt an existing system.	Investigate Paper	No specific hardware or software is used as the paper depicts a generic control system.	The paper added to the development phase of control systems. The processes described in this paper that are used to design and develop control system models would be demonstrated by the second half of 2004.
2	Systematic design and implementati on of distributed controllers in industrial automation (Vyatkin et al., 2006)	Comparing different approaches to distributed system design in automation systems.	A distribution station consisting of a magazine and feeder unit is used to transport workpieces from storage.	No specific hardware or software is mentioned but should be IEC 61499 compliant.	Function block applications are sufficient when used to implement distributed controllers.
3	Multi-Agent Modeling of Cyber-Physical Systems for IEC 61499 Based Distributed Automation (Lyu et al., 2021)	Model industrial cyber-physical systems using a two-layer architecture: high level uses a multi-agent computer model, whereas low level uses the IEC 61499 Standard function block model.	Generic System to show the functions of the two-layer architecture.	SPADE is used to develop multi-agent models. Eclipse 4Diac is used to develop the IEC 61499 Standard function blocks. The agents are spread out between a Jetson Nano, and Raspberry Pi microcontrollers.	Future work needs to be done for both layers in the architecture proposed. Multi-agent modeling requires the development of self-learning and machine learning capabilities. IEC 61499 Standard function block modeling requires an easier method of deploying control applications to microcontrollers such as the Raspberry Pi.

Due to ambiguities in the execution methods of IEC 61499 Standard function blocks in the first edition, different software environments handled event inputs differently. Authors from articles 1 and 2 in Table 2.4 concluded that different execution methods cause different results on the same IEC 61499 Standard function blocks. Multiple execution methods led to authors writing articles, such as article 3 in Table 2, comparing different execution methods and the behavior it has on the IEC 61499 Standard function blocks.

These different execution methods prevent portability of IEC 61499 function blocks; therefore, most authors suggested a major revision of the standard is needed. The major revision of the IEC 61499 standard was issued in 2012, fixing most ambiguities noticed by researchers of the standard. The new revisions focus on the initial goal of the IEC 61499 standard which looks to make programming software more portable, interoperable, reusable, and easily configurable.

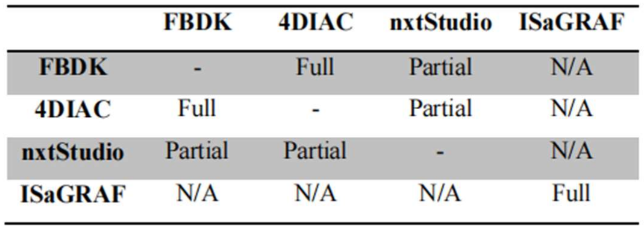
Table 0.4: **Comparison of papers published on executions methods of IEC 61499 Standard function blocks**

No.	Paper (Reference)	Aim of the project	System Overview	Hardware/Software Required	Author's Conclusions
1	Formal Modeling of Function Block Applications Running in IEC 61499 Execution Runtime (*Čengić et al., 2006)*	To prove that different logical behavior of function blocks is shown when different execution methods are used.	Carriage transport system for workpieces.	Simulation and function block application execution by free software Fiber. Supports IEC 61499 runtime.	Using different execution run time environments, which use different execution methods, led to the different behavior of function blocks.
2	IEC 61499 Function Block Model - Facts and Fallacies (Thramboulidi s, 2010)	Look at the semantics of IEC 61499 function blocks to discuss facts and fallacies to determine why the standard has not been accepted by the industry yet.	Review paper illustrating and analyzing multiple case studies.	Multiple IEC 61499 compliant hardware and software are presented.	Many issues are investigated which led to the conclusion that a major revision of the standard Is needed to allow it to be considered for industrial use. The standard does not apply certain basic engineering practices, which has led to many ambiguities during the initial phase of development.
3	Design and execution issues in IEC 61499 distributed automation and control systems (Strasser et al., 2011)	To discuss and present different execution models of function blocks, CFBs, and sub-applications, in IEC 61499 devices.	Investigate Paper	No specific hardware or software is mentioned but should be IEC 61499 compliant.	This paper contributes to the ongoing goal of achieving an IEC 61499 distributed control system that is portable, configurable, interoperable, and distributed.

Many articles are written on the lack of portability between the different IEC 61499 Standard programming environments. Article 1, in Table 2.5, compared the differences and similarities between these software tools by testing their portability between them. Figure 2.5 shows a summary of the results of the study, comparing FBDK, 4DIAC, NXTStudio, and ISAGRAF. The study found that library repositories and function blocks are not fully portable between different vendors, which defeated the point of using the IEC 61499 Standard. This lack of portability is mainly due to the different execution methods used by the different vendors.

Table 0.5: **Summary of a portability study on IEC 61499 Standard software tools published by (Pang et al., 2014)**

No.	Paper (Reference)	Aim of the project	System Overview	Hardware/Software Required	Author's Conclusions
1	A Portability Study of IEC 61499: Semantics and Tools (Pang et al., 2014)	A study of portability issues in IEC 61499 tools such as FBDK, 4DIAC, ISaGRAF, and NXTStudio. The main outline is with the compatibility issues because of different execution semantics.	Investigate Paper	FBDK, 4DIAC, ISaGRAF, and NXTStudio are all compared and tested for compatibility.	IEC 61499 second edition has solved many of the execution issues since its release. Full portability and interoperability can only be reached once it is possible to formally analyze and validate against runtime platforms.



	FBDK	4DIAC	nxtStudio	ISaGRAF
FBDK	-	Full	Partial	N/A
4DIAC	Full	-	Partial	N/A
nxtStudio	Partial	Partial	-	N/A
ISaGRAF	N/A	N/A	N/A	Full

Figure 0.5: **Comparison of IEC 61499 Standard programming environments (Pang et al., 2014)**

The IEC 61499 Standard became more popular after the release of the second edition in 2012. Articles 1 and 2, in Table 2.6, both conclude that the use of the IEC 61499 Standard in the industry will increase, and that vendors and early adaptors should start investing in this means of controlling distributed systems. Both these articles are investigative, focusing on how the standard has improved and become more industry useful.

Table 0.6: **Comparison of papers published on the release of the IEC 61499 Standard second edition**

No.	Paper (Reference)	Aim of the project	System Overview	Hardware/Software Required	Author's Conclusions
1	The IEC 61499 Function Block Standard: Overview of the second edition (Christensen et al., 2012)	Presents improvements of the IEC 61499 Standard to its second edition which will be released in late 2012.	Investigate Paper	No hardware or software is presented as the entire IEC 61499 Standard is looked at.	After listing all the new additions that form the IEC 61499 Standard second edition, it is said stated that the standard will be refined to be more clear, unambiguous, and industrially useful. The conclusion is that all vendors should seriously consider developing software tools, runtime platforms, or control hardware so that they can enter a growing market.
2	The IEC 61499 Function Block Standard: Launch and Takeoff (Strasser et al., 2012)	Discuss the start of the take-off phase of the IEC 61499 Standard. Look at processes that will lead to a successful take-off of the standard.	Investigate Paper	No hardware or software is presented as the entire IEC 61499 Standard is looked at.	IEC 61499 take-off phase has started. The standard can serve a great payoff if early adapters focus on addressing compliant profiles and training personnel as soon as possible.

This subsection discussed the similarities and difference between multiple articles written on the IEC 61499 Standard. The next section of Chapter 2 describes the literature review on networked control system..

2.2 Literature Review on Networked Control Systems

2.2.1 Introduction

The proposed control system in this thesis consists of a Direct Current (DC) motor being controlled via an Ethernet for Control Automation Technology (EtherCAT) fieldbus network. The EtherCAT network is a replacement for the usual hardwired system. Due to this, a literature review on Network Control Systems (NCSs) assists in gaining the necessary knowledge before designing and implementing the system. This review focuses on the advantages and disadvantages of NCSs, as well as the network-induced delays that occur in these real-time networks. Multiple authors' methods to overcome these delays are also discussed.

Section 2.2.2 describes the keywords used as well as the sources searched to complete the literature review. Section 2.2.3 gives a summary of all the literature reviewed on NCSs in a tabular form. Section 2.2.4 gives a summary of the literature reviewed, focusing on an overview of NCSs and the delays that are caused by using a fieldbus in the network. Section 2.2.5 highlights comparisons between the literature reviewed, followed by a conclusion to the chapter in Section 2.2.6.

2.2.2 Literature Search

The main source of literature is obtained from the IEEE Xplore database. The following keywords and phrases were searched in this database: Network control systems; delays in network control systems; distributed network control systems; stability in network control systems. A collection of papers regarding these keywords areread and further research papers are found from the references in these articles. References found led to various websites and articles from ResearchGate.

A total of 20 papers are reviewed on the above-mentioned keywords. Figure 2.6 illustrates a bar graph of the number of papers reviewed verses when the papers were published. The paper's publication dates range from 2005 - 2023.

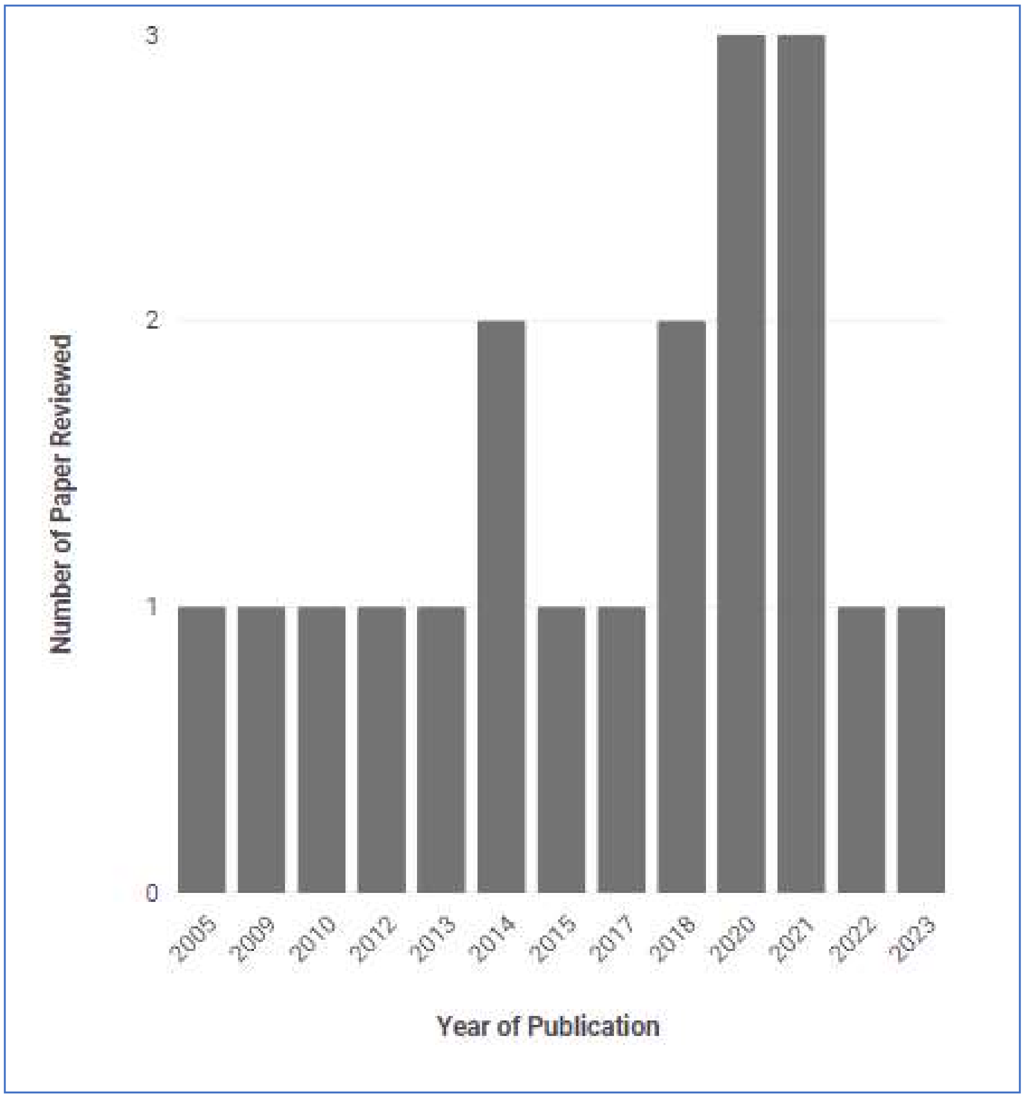


Figure 0.6: **Bar graph showing the number of publications, categorized by year published, reviewed on networked control systems**

2.2.3 Literature Review Summary

Table 0.7: Literature Review of Networked Control Systems

Paper (Reference)	Aim of the project	System Overview	Hardware/Software Required	Author's Conclusions
A New Method for Stabilization of Networked Control Systems with Random Delays (Zhang et al., 2005)	To analyse the stabilization problem of a discrete-time plant with random network-induced delays. To model the two network delays as Markov chains and then design a state-feedback controller to reduce the concern of stability.	A cart with an inverted pendulum example is used as a closed-loop system in the numerical example.	No specific hardware or software is mentioned in this paper, but a graphing-simulating tool is used to display the results of the illustrative example.	Results from the numerical example prove that the closed-loop system is stochastically stable when using the methods developed in this paper.
Output Feedback Stabilization of Networked Control Systems with Random Delays Modeled by Markov Chains (Shi et al., 2009)	To propose methods to guarantee stability and performance in NCSs when designing the controller. Use Markov chains to model network-induced delays.	A cart with an inverted pendulum example is used as a closed-loop system in the numerical example.	No specific hardware or software is mentioned in this paper, but a graphing-simulating tool is used to display the results of the illustrative example.	Results from the simulations of the proposed system example proves that the methods used are effective.
Network Control Systems – Overview and Research Trends (Gupta and Chow, 2010)	To summarize trends and history of networked control systems and the different research areas, such as network delays, real-time network security, allocation of resources, and integrating network components. Future areas of research are also discussed and looked at.	Investigative paper	Investigative paper	The authors research shows the importance of development and research of networked control systems as most systems today are connected via some sort of network. The paper lists all the main research topics that are being investigated, and also challenging problems that haven't been solved for future research.
Effect of Network-induced Delays in Control Systems – DC motor application (Kolla and Mainoo, 2012)	Compare the effects of network induced time delays in different position controllers for a DC motor.	Simulation of a DC motor	MATLAB/Simulink used for	The authors experiments prove that an increase in network delays leads to an increase in network stability. The analysis are completed on two different controllers and both are influenced negatively by these delays, which occur either from the sensor to the controller, or from the controller to the actuator. The linear quadratic regulator performed better that the Proportional Integral Derivative (PID) controller when subjected to network delays.
Modelling and Stabilization for Singular Networked Cascade Control Systems with State Delay	To use Linear Matrix Inequalities (LMI) to model a singular networked cascade controller that is stable.	An example singular networked cascade control system with state	MATLAB LMI toolbox is used to generate the state response graph as well as the desired gains	The authors use Lyapunov stability theory to derive sufficient conditions for the systems. The proposed methods show applicability and usefulness developed and can be used for

(Zhaoping et al., 2013)	Network-induced delays and data packet loss are also considered in the controller design.	delay and disturbance.	for the corresponding systems.	controller design of systems with time delay and packet loss.
Networked Predictive Control for Linear Systems with Unknown Communication Delay (Sun et al., 2014)	To propose a control scheme, for controller design, that can predict unknown communication delays in a linear system.	Two examples are used in this paper to apply the formulas developed. Example 1 is a cart-pendulum system. Example 2 is a servo control system.	No specific hardware or software is mentioned in this paper, but the results are displayed in graph form using a simulation tool.	The author uses switched Lyapunov function method to deal with network-induced delays and achieve stability. Graphs comparing system output and time, of both examples, show the effectiveness of this method.
Experimental Investigations for Distributed Networked Control Systems (Mahmoud and Sabih, 2014)	Experiment and simulate on available software and hardware to create further fields of research for distributed networked systems.	Two PCs are used to simulate the controller and the plant. A data interface is used to gather information from the actual plant.	LabVIEW simulations of a tank level control system distributed over two PCs on an Ethernet network. MATLAB/Simulink	
Fuzzy Speed Control of Networked Motion Control Systems (Zhang et al., 2015)	To model the control of an induction motor in a network control system. To compensate for time delays by designing and using a state predictor. To illustrate the effectiveness of the approach through simulations and experiments.	An NCS that includes an induction motor with a feedback sensor, a controller, an actuator, and a local controller, all connected via a communications network.	Simulations are done using MATLAB/Simulink software tool, utilizing the TrueTime toolbox. Control Area Network (CAN) protocol is used as the Fieldbus for communication.	The authors use the Lyapunov-Krasovskii function to calculate the maximum allowed time delay for the system to be stable. The theorem is also used to calculate the limit of packet dropouts. Feedback time delay is minimized by the state predictor using the feedback time-stamped messages. The fuzzy PI controller works better than the Proportional (P) and PI controller, as this controller shows a better stead-state performance. This controller is more robust to variation in the network Quality of Service (QoS).
Distributed Control of Large-Scale Networked Control Systems with Communication Constraints and Topology Switching (Zhang et al., 2017)	To suggest and prove methods that allow for large-scale networked closed-loop systems to be more stable and have an H∞ disturbance attenuation level. Methods such as event-based communication and logarithmic	Two Continuous Stirred Tank Reactors (CTSRs) connected via a communication network are used	No specific hardware or software is mentioned in this paper, but a graphing-simulating tool is used to display the results of the illustrative example.	To reduce the size of information transmitted in the communication network, methods such as event-based control and logarithmic quantization are introduced. The authors use Lyapunov stability theory and a switched system approach to propose sufficient conditions that allow the closed-loop system

	quantization are tested during controller design.	as an illustrative example.		example to be exponentially stable and have an H∞ disturbance attenuation level. A graphical simulation is created which demonstrates how effective the proposed controller design is.
Stability and H Infinity Performance of nonlinear Fuzzy Network Control Systems with Time Varying Delay (Lu et al., 2018)	To develop a controller for nonlinear models which includes time-varying delays.	Developed theorems are tested on a fuzzy network control system	MATLAB Toolbox for the calculations.	The author completes the proof by using the theorems developed on an example model. The results show the system is stable.
Stability Criterion for Networked Control Systems Based on T-S Model with Time-Varying Delays (Liu and Liu, 2018)	Create methods to proof stability of networked control systems that have time-varying delays.	Developed theorems are tested on a T-S fuzzy network control system	No software or hardware mentioned in the article.	The author completes the proof by using the theorems developed on an example model. The results show the system is stable even though the system is not conventional and pulse free.
Observer Based Incremental Predictive Control of Networked Multi-agent Systems with Random Delays (Pang et al., 2020)	To compare the effects of network induced delays and random packet losses between a networked system without compensation and a system with an incremental networked predictive controller.	A networked multi-motor control test rig	No software mentioned in the article. Siemens control equipment used on the test rig.	The graphed results show the experimental outputs of the system without compensation and the system with incremental networked predictive control. The results show the effectiveness and applicability of the introduction of this type of controller to overcome networked induced delays and random packet dropouts. The methods are tested on 3 different motors to further prove the effectiveness of the developed control.
Delay-dependent Stability Analysis of Networked-controlled DC Motor with Time-invariant Delays (Subramanian and Kokil, 2020)	To analyse the effects of delay margins, phase margins, and gain margins on networked control systems.	A DC motor networked control system is used for analysing.	MATLAB/Simulink is used to simulate the model and graph the output responses.	The simulations show that gain and phase margins influence the system responses, such as rise time and overshoot. The results show that an increase in gain and phase margins decreases the delay margins in a closed-loop system. Using a proportional controller helps tune a controller to be stable when time delays are present.
Networked DC Motor Control with Time-Varying Delays and Application to a Mobile Robot (Xie et al., 2020)	To test the performance and stability of a networked DC motor performing position control while under the effect of time-varying delays.	A DC motor test system consisting of two DC motors with encoder feedback to a controller via a 4G network adapter	M091760 Maxon encoder with a National Instruments myRIO controller for data acquisition. MATLAB/Simulink is used to simulate and graph the results.	The author tested a networked DC motor system with a range of time-varying delays to show the impact on the stability of the system. The pole placement method is used to compensate for these delays to a certain point until the delays are so large that the system remains unstable. Future work will consider disturbances and uncertainties, incorporating predictive control and sliding mode control into the investigations.

Robust Stability of Networked Linear Control Systems with Asynchronous Continuous and Discrete Time Event Triggering Schemes (Xiao et al., 2021)	Analyse and simulate continuous-time and discrete time systems to test event-triggered approaches when transmitting data over networks.	A numerical example is used to test the developed theorems.	No software or hardware mentioned in the article.	The authors methods show that the event-triggered approaches to data transmission reduces the traffic on a network in a networked control system. The methods are tested in a numerical example that applies to many continuous-time or discrete-time systems, as well as time-varying delays.
Event-triggered Optimal Control for the NCSs with Time Delays (Li et al., 2021)	To develop theorems to be used in event-triggered networked control systems that are subjected to network-induced delays.	A numerical example is used to test the developed theorems.	No software or hardware mentioned in the article.	By using an event-trigger generator, the authors reduced the amount of bandwidth used on the network hence reducing the network-induced delays. The rest of the delay is compensated by introducing a network predictive controller.
Observer-based Controller Design for a Class of Networked Control Systems with Transmission Delays and Packet Losses (Asl et al., 2021)	Analyse the effects of network delays, between zero and two times the sampling rate, on an observer-based network control system.	A simulation example is used to test the developed theorems.	No software or hardware mentioned in the article.	By using LMI to determine the controller and observer gains, the proposed methods could stabilize the networked control system as well as improve the performance.
The Effect of Observer Position on Networked Control Systems with Random Transmission Delays and Packet Dropouts (Asl et al., 2022)	To analyse the effects of observer placement in networked control systems that have time-varying delays.	A numerical example is used to test the developed theorems.	No software or hardware mentioned in the article.	Two models are tested and compared. The first model the observer is inserted on the system side, whereas the second model has the observer on the controller side. Both models are tested with a numerical example and the results showed that the second model performed better than the first, even though the first did stabilize the system.
Design of Predictive Controller for Networked Control Systems (Chen and Zhou., 2023)	To analyses the stability and system performance of a networked control system with communication delay as well as data packet loss, not just one of the two.	A simulation example is used to test the developed theorems.	A Dc motor simulated plant is used for testing.	The proposed method of using a model predictive controller resulted in the networked control system having the same response as a closed-loop system without network induced delays.

2.2.4 Literature review based on network control systems

2.2.4.1 Overview of network control systems

A Network Control System (NCS) is a system where all components are connected as nodes via a communications protocol instead of hardwired connections (Kolla and Mainoo, 2012). Unlike classic control systems, communication networks are used to close the control loops instead (Mamoud et al., 2014).

The advantages of these networks include the reduction in wiring costs, weight and space reduction of the system, simple installation and diagnosis for maintenance, and an overall increase in reliability and agility (Zhang et al., 2005), (Shi et al., 2009). (Gupta and Chow, 2010) states that these networks are designed to replace the existing 4-20mA analog signal standard and termed them as "shared networks".

Today, NCSs are becoming more distributed as there is an increase in complexity in modern engineering systems (Zhang et al., 2017). With the sensors, actuators, and controllers being distributed over a communications network, instead of being hardwired, the performance of the system degrades due to network-induced delays (Gupta and Chow, 2010). These delays need to be considered when designing the controller in an NCS to prevent the system from underperforming and becoming unstable (Shi et al., 2009).

2.2.4.2 Delays in networked control systems

One of the biggest issues in NCSs are the network-induced delays between the sensor and controller, as well as between the controller and actuator (Zhang et al., 2005). As shown in Figure 2.7, these delays are random, but can also be constant or time varying. The overall network-induced delays can be calculated by adding the sensor-to-controller delay (τ_k) and the controller-to-actuator delay (d_k). A computation delay is found in systems with multiple controllers (Zhaoping et al., 2013). This delay would exist between the primary and second controller.

There are many causes of such delays in networked systems, including data transmission, protocol conversion, a selected protocol for communication, and distance between nodes (Yu et al., 2014). Any of these constraints increases the time delays and packet dropouts in NCSs, bringing difficulties in the analysis of these systems (Shi et al., 2009). Another cause is the use of multiple sensors in the same NCS as a mixture of different delays are present and need to be considered (Xiao et al., 2021).

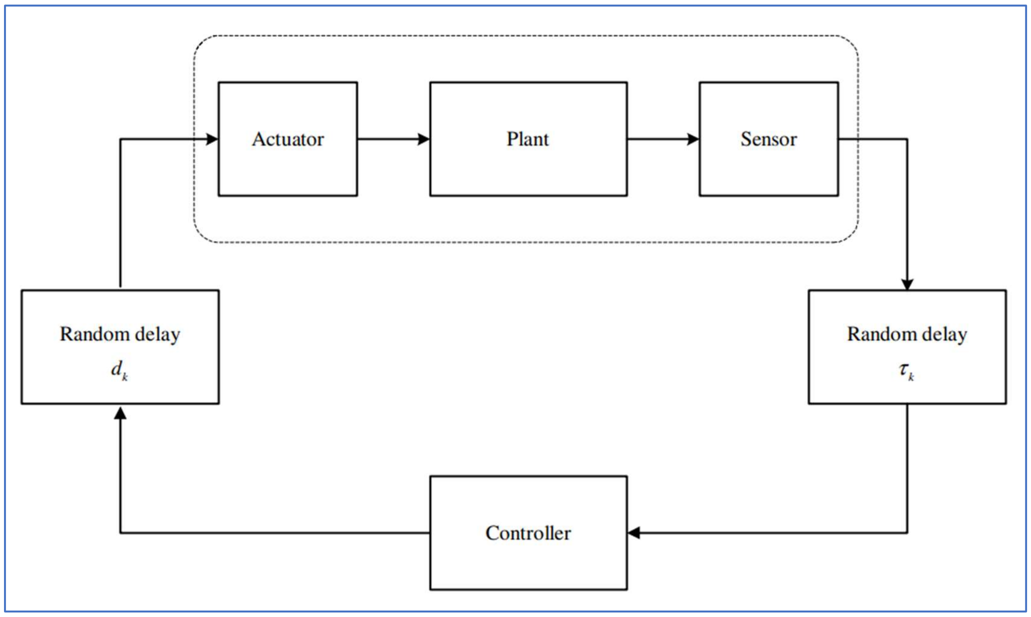


Figure 2.7: **Networked control system with random delays (Zhang et al., 2005)**

Time delays need to be considered when designing controllers to prevent a decrease in stability and performance of the overall system (Yu et al., 2014), (Zhang et al., 2015), (Lu et al., 2018), (Zhaoping et al., 2013). (Chen and Zhou, 2021) state that data loss exists in NCSs and therefore considering these delays during controller design is important. Designs could include some form of time stamping to reduce data loss during communication transmissions.

To prevent such delays, the communications networks need to become part of the control loop (Yu et al., 2014). There are two main methods which researchers use to accomplish this task (Shi et al., 2009). The first method is to design a controller without considering network induced delays and then determining the maximum delays allowed before the system is unstable. The second is to incorporate the network-induced delays using models when initially designing the controller. The latter method, known as Model Predictive Control (MPC), is more efficient as the models can predict the systems behaviours and reduce the effects of delays in real-time. Other methods include the time delay method, robust control method, and stochastic control method.

There are many techniques used by researchers to model network-induced delays. (Zhang et al., 2005) and (Shi et al.,2009) used Markov chains to model the two random delays in NCSs. Once modeled, the controllers are calculated using the iterative Linear Matrix Inequality (LMI) approach. (Yu et al., 2014) used the MPC scheme to compensate for the delays of a test bench which consisted of industrial-grade devices and controllers connected via a communications network. The authors proved that this scheme is a viable solution that can be used in industrial control systems.

(Zhang et al, 2015) used a fuzzy logic controller, with a set of heuristic decision rules, to compensate for network-induced delays on non-linear systems. The authors also used a state predictor between the sensor and controller to alleviate the possibility of time delays during data transmission. Stability is analyzed using the Lyapunov-Krasovskii Theorem to compute the allowed maximum and minimum bounds of the time delays and packet dropouts. (Zhang et al., 2017) also uses the Lyapunov direct method to achieve a closed-loop system which is exponentially stable.

(Asl et al., 2021) designed an observer-based controller to reduce transmission delays and random packet loss. Observers are used in this practical example as not all the states are normally available. (Asl et al., 2021) then analysed the effects of observer-based control when designing NCSs. Two models with different observer locations are tested under the same conditions. The results showed that the model with an observer on the controller side stabilized the system.

There are methods to compensate for network-induced delays that do not require any input in the controller design process. These are related to the network infrastructure of the NCS. Delays can be minimized depending on which network protocol is used in the NCS (Mahmoud et al., 2014). Distributing the control by using an event-triggered approach when programming decreases network traffic as data requests are less (Xiao et al., 2021). (Zhang et al., 2017) also suggests event-based control in NCSs and further adds that signal quantization reduces the communication rate.

This subsection discussed multiple techniques to overcome network delays in NCSs. The next subsection compares literature reviewed on NCSs.

2.2.5 Comparison between literature reviewed on network control systems

Many authors have used similar methods when trying to solve problems in NCSs. These methods and theorems are usually developed years prior to current research but are used on newer issues that researchers come across. These newer issues include phenomena such as network-induced delays and network time delays, which were never a problem before a communication link was added into the control system between the different nodes in an automation system.

In the case of articles 1 and 2, in Table 2.3, both used Markov chains to model the network-induced delays in a network control system. The same example of a cart with an inverted pendulum is used to create a closed-loop system for testing. The articles both illustrate the results of a numerical example using a graphing tool which shows the stability of the system when subjected to network-induced delays. The outcome of each paper showed effective results as both systems remained stable when subjected to random network-induced delays. The results showed no decrease in performance in the system proving that using Markov chains when modeling is beneficial when dealing with these delays.

Table 0.8 **Comparison of papers published on random delays in network control systems**

No.	Paper (Reference)	Aim of the project	System Overview	Hardware/Software Required	Author's Conclusions
1	A New Method for Stabilization of Networked Control Systems with Random Delays (Zhang et al., 2005)	To analyse the stabilization problem of a discrete-time plant with random network-induced delays. To model the two network delays as Markov chains and then design a state-feedback controller to reduce the concern of stability.	A cart with an inverted pendulum example is used as a closed-loop system in the numerical example.	No specific hardware or software is mentioned in this paper, but a graphing-simulating tool is used to display the results of the illustrative example.	Results from the numerical example prove that the closed-loop system is stochastically stable when using the methods developed in this paper.
2	Output Feedback Stabilization of Networked Control Systems with Random Delays Modeled by Markov Chains (Shi et al., 2009)	To propose methods to guarantee stability and performance in NCSs when designing the controller. Use Markov chains to model network-induced delays.	A cart with an inverted pendulum example is used as a closed-loop system in the numerical example.	No specific hardware or software is mentioned in this paper, but a graphing-simulating tool is used to display the results of the illustrative example.	Results from the simulations of the proposed system example proves that the methods used are effective.

Articles 1, 2, 3 and 4, in Table 2.4, all investigate network-induced delays in NCSs. Each author uses a different method to incorporate the delays into the closed-loop system, but all four prove effectiveness in their methods. Stability is achieved in all the examples by using the Lyapunov-Krasovskii method to calculate sufficient conditions for the system. The use of the MATLAB/Simulink software tool is used in Articles 1 and 3 to simulate and illustrate the results of the controller's response to network-induced delays in the control loop.

Table 0.9: **Comparison of papers published on methods to incorporate network-induced delays in NCSs**

No.	Paper	Aim of the project	System Overview	Hardware/Software Required	Author’s Conclusions
1	Fuzzy Speed Control of Networked Motion Control Systems (Zhang et al., 2015)	To model the control of an induction motor in a network control system. To compensate for time delays by designing and using a state predictor. To illustrate the effectiveness of the approach through simulations and experiments.	An NCS that includes an induction motor with a feedback sensor, a controller, an actuator, and a local controller, all connected via a communications network.	Simulations are done using MATLAB/Simulink software tool, utilizing the TrueTime toolbox. CAN network protocol is used as the Fieldbus for communication.	The authors use the Lyapunov-Krasovskii function to calculate the maximum allowed time delay for the system to be stable. The theorem is also used to calculate the limit of packet dropouts. Feedback time delay is minimized by the state predictor using the feedback time-stamped messages. The fuzzy PI controller works better than the Proportional (P) and PI controller, as this controller shows a better steady-state performance. This controller is more robust to variation in the network Quality of Service (QoS).
2	Distributed Control of Large-Scale Networked Control Systems with Communication Constraints and Topology Switching (Zhang et al., 2017)	To suggest and prove methods that allow for large-scale networked closed-loop systems to be more stable and have an H∞ disturbance attenuation level. Methods such as event-based communication and logarithmic quantization are tested during controller design.	Two Continuous Stirred Tank Reactors (CTSRs) connected via a communication network are used as an illustrative example.	No specific hardware or software is mentioned in this paper, but a graphing-simulating tool is used to display the results of the illustrative example.	To reduce the size of information transmitted in the communication network, methods such as event-based control and logarithmic quantization are introduced. The authors use Lyapunov stability theory and a switched system approach to propose sufficient conditions that allow the closed-loop system example to be exponentially stable and have an H∞ disturbance attenuation level. A graphical simulation is created which demonstrates how effective the proposed controller design is.
3	Modelling and Stabilization for Singular Networked Cascade Control Systems with State Delay (Zhaoping et al., 2013)	To use LMI to model a singular networked cascade controller that is stable. Network-induced delays and data packet loss are also considered in the controller design.	An example singular networked cascade control system with state delay and disturbance.	MATLAB LMI toolbox is used to generate the state response graph as well as the desired gains for the corresponding systems.	The authors use Lyapunov stability theory to derive sufficient conditions for the systems. The proposed methods show applicability and usefulness developed and can be used for controller design of systems with time delay and packet loss.
4	Networked Predictive Control for Linear Systems with Unknown Communication Delay (Sun et al., 2014)	To propose a control scheme, for controller design, that can predict unknown communication delays in a linear system.	Two examples are used in this paper to apply the formulas developed. Example 1 is a cart-pendulum system. Example 2 is a servo control system.	No specific hardware or software is mentioned in this paper, but the results are displayed in graph form using a simulation tool.	The author uses switched Lyapunov function method to deal with network-induced delays and achieve stability. Graphs comparing system output and time, of both examples, show the effectiveness of this method.

This subsection discussed the comparisons between the literature reviewed on NCSs. The next subsection concludes Section 2.2.

2.3 Conclusion

The IEC 61499 Standard has become more popular since the second edition was released in 2012. The standard has started the path to Industry 4.0, allowing the development of distributed control systems that are portable, interoperable, configurable, and reusable. These factors allow developed software to run on different hardware platforms from multiple vendors, which relaxes hardware and software dependencies in automation systems (Zhabelova et al., 2017). This allows for feasible alternatives as standard PLCs are overpriced and can be substituted by industrial PCs and embedded controllers distributed in a controlled environment (Vyatkin 2011).

The adoption of the IEC 61499 Standard in the industry lead to heterogeneous control systems. The two standards (IEC 61499 and IEC 61311-3) must co-exist, for now, making the adaptation of existing systems to distributed systems a challenge. This challenge of engineering heterogeneous systems is said to be “not as big as expected” as the two standards are similar in most ways (Hirsch et al, 2007). The PLC code used in existing systems can be reused in IEC 61499 function blocks, leading to a shorter learning curve when migrating from one standard to the other (Dai and Vyatkin, 2012).

The goal of Industry 4.0 is to achieve distributed control and intelligent automation. This can be done by creating software that uses the IEC 61499 Standard at the low-level physical modules and linking these modules with intelligent software systems at a high-level (Lyu et al., 2021). These systems are termed “Cyber-physical" where the physical parts of a system are used to automatically generate the cyber parts (Yan and Vyatkin, 2011).

The IEC 61499 Standard is leading the path to holonic systems in industrial automation. With the rapid growth of technology, the future of self-adapting and self-organizing distributed control systems could be sooner than expected.

Developing control distributing control systems also has some disadvantages that have lead to an increase in research and development topics. Distributed systems are usually controlled via a network instead of hardwired connections. This networked topology has added the effects of networked induced delays into the control system. Many researches have studied and experimented what causes these delays, and how to design control systems that can negate the effects that can cause instability when using a network.

In most reviewed papers, the experimental and simulation results are proven to be effective. (Mahmoud et al., 2014) suggests that researching in a co-experimental simulation environment improves results and contributes to expanding the knowledge base in the community of control systems. For NCSs to become more realistic for the industrial environment, more physical designs need to be built to validate these results. (Yu et at., 2014) mentions that more lab-scale systems with industrial-grade instruments need to be built and tested on, and more focus should be put on what causes these delays in a physical environment.

Although there are many advantages to upgrading an existing control system to be networked, the initial development and conversions are costly, time consuming and inconvenient (Gupta and Chow, 2010). The existing systems must be redesigned, with some devices being replaced, to allow for communication over a data network instead of being hardwired. These initial costs are what causes companies not to progress into smarter, more distributed systems.

NCSs allow for a simpler setup of control systems as devices are becoming plug-and-play, and therefore reducing the complexity of the design and building phases in projects. Even though there are disadvantages in NCSs such as network-induced delays, there are many researched methods that can be used when designing the controller to mitigate these delays. Correct controller design and the correct use of the communication capabilities of networked systems can increase the overall performance of the solution (Zhang et al., 2017).

This chapter reviewed literature based on the IEC 61499 Standard and networked control systems. The following chapter describes the theory based on the plant that is used in this thesis.

CHAPTER 3
THEORY BASED ON ANTENNAS AND THE DC MOTOR

3.1 Introduction

This chapter discusses the plant which has been selected to be controlled in this thesis. The plant is a radio antenna which is used to locate celestial objects in the sky. This thesis focuses on the development of the controller that guides the dish to specific position set points accurately and does not focus on the actual tracking of celestial objects in the sky.

The following sections describe the plant in more detail. The radio antenna is described in Section 3.2. The DC motor is described in section 3.3. Radio antenna and DC motor discussions are described in section 3.4. The chapter is concluded in section 3.5.

3.2 Antenna Theory

An antenna is a component that converts guided waves into free-space waves when transmitting, or free-space waves into guided waves when receiving (Balanis, 1992). This defines an antenna as a transducer because the device converts energy from one form to another. As shown in Figure 3.1, the system consists of a parabolic dish, which collects all the electromagnetic (EM) waves into a focal point, an antenna to convert the received waves into current, and two DC motors for azimuth and altitude positional movements. The motors are coupled to servo drives which receive the necessary positional set points from a Programmable Logic Controller (PLC).

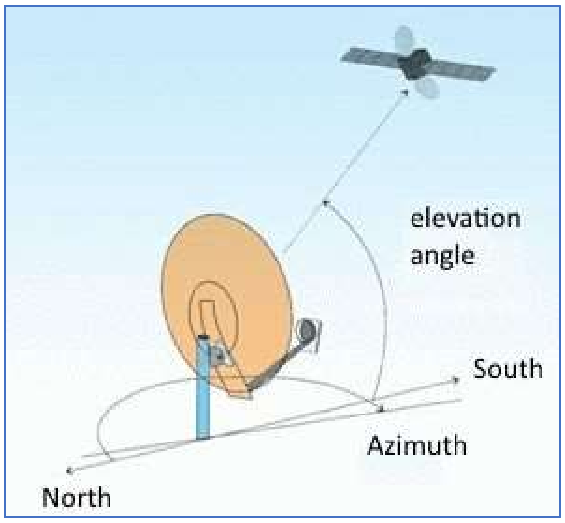


Figure 0.1: **Azimuth and elevation of a radio antenna**

Antennas can either transmit EM waves or receive them, depending on the type. When transmitting, the antenna receives a current from a source device and then radiates the generated waves at specific frequencies through the air toward other antennas. When receiving, the antenna intercepts the waves that are airborne and converts them to a current which is sent to a device for processing. It is possible for antennas to both receive and transmit EM waves by means of a transceiver (Sheldon, 2023).

We need antennas to allow for the transmission of signals where hard-wired connections are not possible. The use of antennas for wireless communications is more feasible for applications where cables are not required (Administrator, 2019). Selection of which antenna to use is application dependant. Figure 3.2 shows several different types of antennas including wire dipole, loop, wire monopole, Yagi-Uda array, horn, microstrip patch, corner reflector, parabolic reflector (dish), and slot.

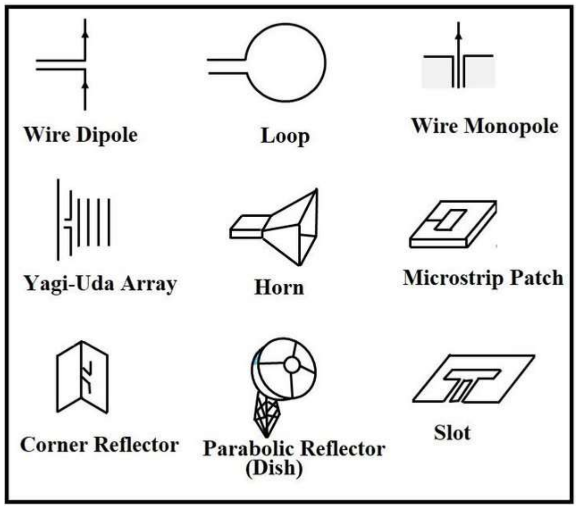


Figure 0.2: **Several types of Antennas**

(Dhillon and Kumar, 2017)

The simplest antenna design is a wire antenna. This is the most used antenna and can be found in cars, ships, buildings, and aircraft. The wire dipole, loop, wire monopole and Yagi-Uda array antennas are all designed based on the same functionality as the wire antenna. One end of each of these antennas is connected to either a receiver or a transmitter, and the rest of the antenna captures the airborne EM signals that are travelling in free space. The Yagi-Uda is the most complex wire antenna but was most popular for receiving television signals before dish antennas were used.

Microstrip patch antennas are small antennas printed into circuit boards to allow for wireless communications in mobile devices (Sheldon, 2023). Corner reflectors reflect EM waves back towards the source, which is useless for calibrations of radar systems. Slot antennas are like dipole antennas but are cut into a surface instead of a wire made up of material.

Figure 3.3 shows a radio antenna system that is used to obtain EM waves from celestial radio sources. The system includes a parabolic reflector (dish) as well as a horn antenna. The parabolic structure of the dish reflects EM waves to a focal point, which causes the waves to bounce back off a sub reflector and into the horn antenna. The horn antenna converts the EM waves into a current signal that is fed into the connected receiver for processing and analyzing.

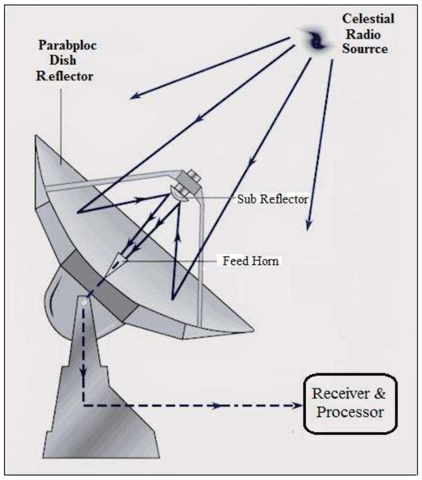


Figure 0.3: **Radio Antenna collecting EM waves from a celestial radio source**

The radio antenna is selected as the plant for this thesis because the azimuth and altitude positional movement control system of the antenna works as a good case study for testing of a multi-variable control system. The following sections describe the radio antenna in more detail, considering the history of radio antennas, types of radio antennas, and current control systems of antennas.

3.2.1 History of Radio Antenna

The history of radio antennas begins with the discovery of EM radiation by Heinrich Hertz in 1886 (Rubin, 2018). The German physicist, Heinrich Hertz, set out to prove Maxwell's theory of electromagnetism by building the test rig shown in Figure 3.4. He made use of a capacitor (C) and an induction coil (T) to generate a spark between a spark gap (S) formed by two spheres. This spark, in turn, generated EM waves, which were then enhanced by an antenna (A). A receiving antenna (RA) was placed across the transmitting antenna to capture the oscillating EM waves that were being produced. The received waves were concentrated into the receiving gap (C) causing a spark, proving that radiation was being detected. Due to the limitations in frequencies, the concepts and proofs developed by Hertz were not used until after World War 2 when microwave frequencies were discovered.

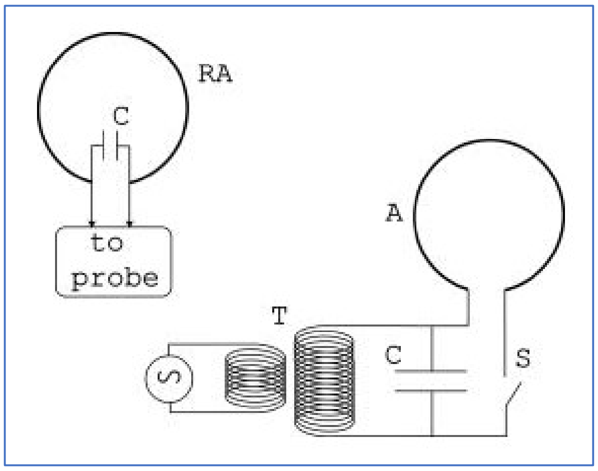


Figure 0.4: Heinrich Hertz's radio antenna experiment

(Faccio et al, 2006)

The first use of radio telescopes to observe EM waves produced from space was Karl Jansky, 1933 (Rahmat-Samii et al ,2009). He built a radio antenna that detected radio waves from the Milky Way but found little support from most astronomers regarding his theories and could not get enough funding for his research. His work was revisited in 1937 by Grote Reber, who built a 9.5m parabolic reflector radio antenna in his backyard. This was the first antenna used for astronomical research, and his sky surveys created the field of radio astronomy (Wielebinki, 2007).

Parabolic antennas became widely used during the 1960's for microwave communication systems. The first radio antenna used for satellite communications was built in 1962 at Goonhilly earth station, in Cornwall, UK. This antenna as the first to transmit live video signals across the world, revolutionised global telecommunications.

Although other types of antennas are used in radio astronomy, dish antennas of parabolic reflectors are the most common type used for radio telescopes today (Mirghani, 2017). Due to the discoveries by Heinrich Hertz, Grote Reber, and Karl Jansky, there are multiple radio telescopes around the world that are being used to better understand the universe by detecting small waves from space.

3.2.2 Types of Radio Antennas

Radio antennas consist of two components, a reflective surface, and a feed antenna to collect EM waves at a focal point (Nikolova, 2016). The reflective surface is normally a parabolic dish made of some reflective material, and the antenna is application dependent, for example a horn antenna could be used. As shown in Figure 3.5, there are four different types of radio antennas, namely axial-feed, off-axis feed, Cassegrain, and Gregorian. The antennas are distinguished by how EM waves travel from free space to guided space, meaning from the air into the antenna.

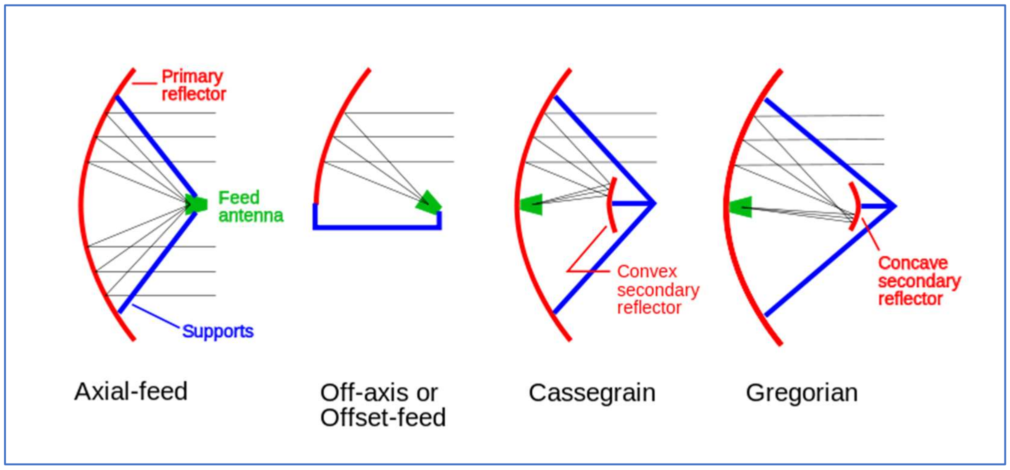


Figure 0.5: **Different types of parabolic radio antennas**

(Stroski, 2019)

The axial-feed parabolic reflector reflects EM waves to focal point which contains a feed antenna to collect the concentrated signals. The main disadvantage with this type of parabolic radio antenna is the support structure blocks a portion of the incoming EM waves. This can be solved by using an off-axis feed system, where the feed antenna is slightly off center of the dish, preventing any blocking of signals from occurring. This type of dish is used mostly for home satellite television systems.

The Cassegrain and Gregorian feed antennas are similar as both have a secondary reflector concentrating the EM waves toward the feed antenna. The only difference is that Cassegrain antennas have a convex secondary reflector, whereas Gregorian antennas have a concave secondary reflector. There are minimal losses in information when the incoming EM waves are passed through either a convex or concave reflector as the feed antenna is mounted physically closer to the receiver (Wolff, 1997). The main disadvantage is the issue of the support structure interfering with the signals being received. An example of a Cassegrain antenna is the Hartebeesthoek Radio Astronomy Observatory (HartRAO) radio antenna found in South Africa as shown in Figure 3.6.

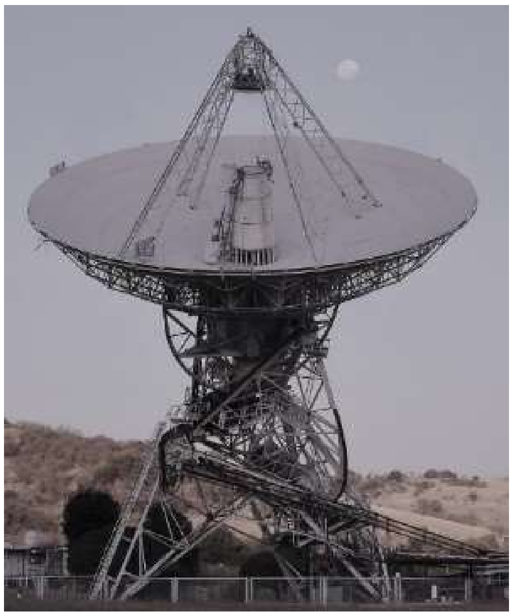


Figure 0.6: **26m dish of HartRAO radio antenna**

(Mirghani, 2017)

3.2.3 Radio Antenna Control Systems

The positional movement of a radio antenna is necessary to locate celestial objects in the sky. To do this, a closed loop control system is required that consists of a positional set point and position feedback. Figure 3.7 shows a simple feedback control loop system. A potentiometer is used to send a required set point to the radio antenna. The difference between the set point and the feedback from the potentiometer connected to gear 3 is amplified and used to turn the motor. Once the error between the potentiometer feedback and input set point is zero, the motor will not rotate and the set position will have been reached (Ahlawat et al, 2019).

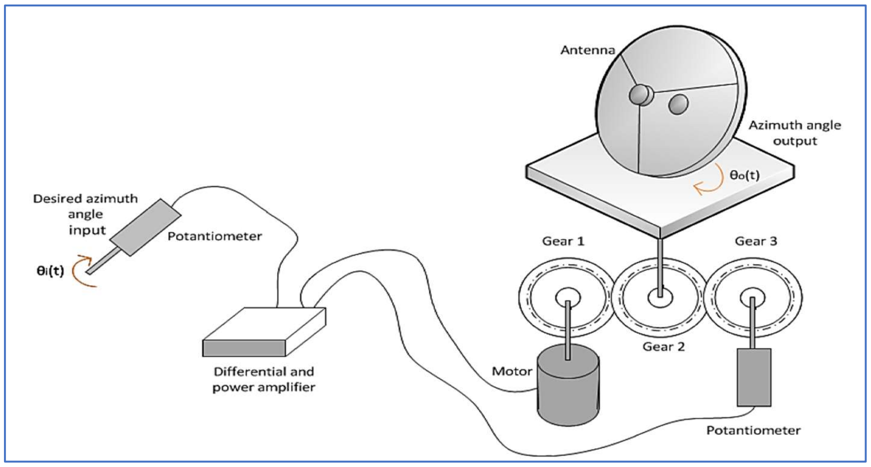


Figure 0.7: **Closed loop position control of an antenna's azimuth angle (Okumus et al, 2012)**

The detailed setup is illustrated in Figure 3.8 as a block diagram. The summing junction shows the subtraction of the input set point and the potentiometer feedback (kpot), before the error is fed into the preamplifier (K). The signal then passes through the power amplifier and into the motor, load, and gears before changing the azimuth angular position of the antenna. This system does not include a controller, and therefore the response cannot be altered to improve the position control performance.

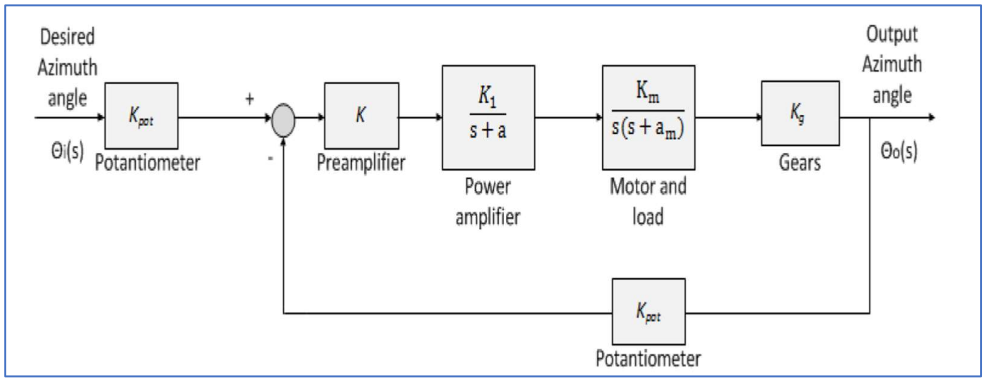


Figure 0.8: **Block Diagram of the closed loop position control of an antenna's azimuth angle (Okumus et al, 2012)**

The problem with not using a controller to optimize the performance of the plant is that the system could experience a non-stable response, causing components to saturate and be damaged leading to massive costs in repairs and replacements (Xuan, 2009). Many authors have simulated different control systems to control the azimuth or altitude position of a radio antenna. Table 3.1 summarizes a few articles regarding these control systems, comparing the software and hardware used, as well as the results that the authors found.

Table 0.1: **Summary of articles written regarding position control of radio antennas.**

Author(s)	Title	Control System	Software/Hardware	Results
Xuan, 2009	Antenna Azimuth Position Control Analysis and Controller Implementation	Control of a DC servo motor with a discrete PID controller.	MATLAB was used for system simulation. No physical system was implemented.	The author's system reaches steady state with some overshoot when using a discrete PID controller and recommends this controller for its response times and stability performance.
Okumus et al, 2012	Antenna Azimuth Position Control with Classical PID and Fuzzy Logic Controllers	Two position controllers for a DC motor are looked at and compared. 1. PID controller 2. Fuzzy logic controller	MATLAB/Simulink was used for system simulation. No physical system was implemented.	The author concludes that the fuzzy logic controller, with several fuzzy rules and functions, gives the most convenient response for the system compared to the classical PID controller.
Temekovski and Achkoski, 2014	Modelling and Simulation of Antenna Azimuth Position Control System	No controller is presented. The actual position of the azimuth position is fed back into the system using a potentiometer. The feedback is subtracted from the input set point and fed into the plant through a power amplifier.	MATLAB/Simulink was used for system simulation. No physical system was implemented.	The author compares the open loop step response with the closed loop step response. The closed loop system overshoots the set point but eventually settles back down with no stead state error, whereas the open loop system never settles as there is no feedback.
Uthman and Sudin, 2018	Antenna Azimuth Position Control System using PID Controller & State-Feedback Controller Approach	Two position controllers for a DC motor are looked at and compared. 1. PID controller 2. State feedback controller with pole placement methods	MATLAB/Simulink was used for system simulation. No physical system was implemented but is mentioned for future work.	The author observed that the state feedback controller performed the best compared to the PID controller. The results showed little settling time and overshoot with not steady state error.
Ahlawat et al, 2019	Antenna Azimuthal Position Control Using Model Predictive Control	Two position controllers for a DC motor are looked at and compared. 1. PID controller 2. MPC	MATLAB was used for system simulation. No physical system was implemented.	The author concluded that the MPC performed better than the PID controller, especially when some transport delay was inflicted on the system. The delays did not affect the response times of the MPC controller as it did with the PID controller.

Temekovski and Achkoski, (2014) approached the antenna control without a controller. Their results show that when there is a change in position set point, the system does stabilize over time leaving no stead state error. With no controller, they are not able to change the response of the system. (Xuan, 2009) instead used a PID controller to control the position of the DC motor that is coupled to the radio antenna. The controller is used to meet the requirements for overshoot and rise time, while achieving zero steady state error.

At first, these authors (Okumus et al, 2012), (Uthman and Sudin, 2018) and (Ahlawat et al, 2019) have utilized a PID controller to control the DC motor's position. This was carried out to provide a point of reference for simulations and conducted on an alternative controller. The authors results described when using a fuzzy logic controller, state feedback controller, and

MPC controller, shows an improvement in response times. All the authors use MATLAB/Simulink to simulate the controllers and display results. As shown by the articles in Table 3.1, the use of a DC motor for the radio antenna position control is very common. The next section will focus on motor theory, which includes a brief history of the DC motor, and a description on the operation and construction of the DC motor.

3.3 Motor Theory

A DC motor is selected to control the rotational angular position of the radio antennas dish in both the azimuth and altitude plane in this thesis. DC motors are often used in industrial applications because of their simplicity, reliability, ease of application and cost effectiveness (Maung et al, 2018). These motors are usually used with gears to provide high torque outputs from the motor shaft to the output of the required gearbox. The high torque provided by DC motors is one of the main reasons why these motors are preferred over other types of motors (Aloo et al, 2016). Another advantage of DC motors is the fast response times to rotational changes as the motors can operate at high speeds (Eze, 2021).

In summary, the DC motor is the most suited motor for use in the control systems of radio antennas due to their high torque capabilities, affordability, simplicity, and ability to provide precise and accurate control of the antennas position when used in a feedback system. The next subsection describes the history, operation, and construction of the DC motor.

3.3.1 History of the DC Motor

The DC motor was designed to convert electrical energy to mechanical energy. DC motors can be found in electrical home appliances, automobiles, and most industrial equipment. Over the period 1820 – 1835, many scientists were involved with the development of the DC motor that is known today. The main contributions came from Christian Oersted, Andre Maria Ampere, Michael Faraday, Joseph Henry, and William Sturgeon (Seale, 2016).

Christian Oersted and Andre Maria Ampere (1820) started the revolution by discovering that an electric current, passing through a conductor produces a magnetic field. In 1821, Michael Faraday wanted to prove and demonstrate this theory. He placed a magnet in a dish of mercury with a wire hanging down into it as shown in Figure 3.9. He then connected the positive side of a battery to the other end of the wire and connected the negative to the mercury portion of the experiment. This caused the wire to rotate around the magnet, creating motion from electrical energy.

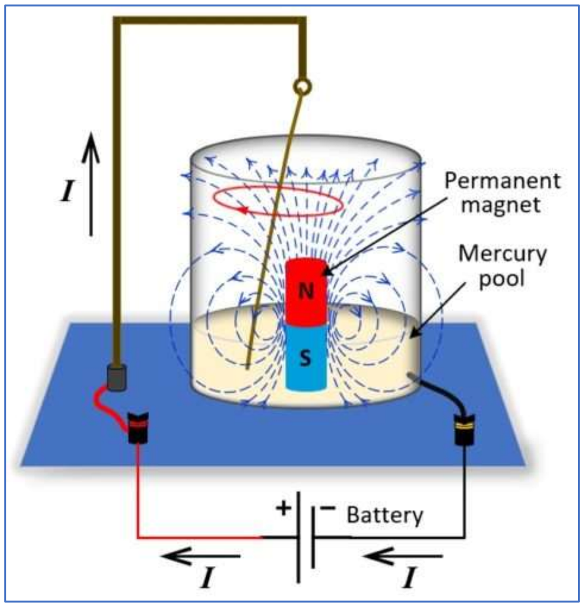


Figure 0.9: **Michael Faraday's DC motor setup**

(Sarkar, 2020)

There was no practical use for Faraday's experiment, but it led to the creation of Joseph Henrys' invention 10 years later (Figure 3.10). The lab experiment used an electromagnet that freely rocked on a reciprocating beam. Once charged with a current, the electromagnet rocked up and down as each coil repels off the North Pole magnetic field. The experiment rocked at a pace of 75 cycles per minute.

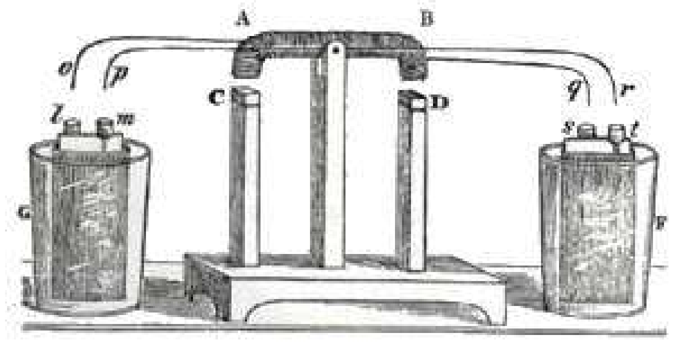


Figure 0.10: **Electro-magnetic engine by Joseph Henry**

(Henry, 1831)

Just a year later in 1832, William Sturgeon was accredited with making the first rotary motor, which is like the DC motor used today. This was made possible when he invented the commutator, a means of constantly supplying a DC voltage to the electromagnet, causing it to rotate. Sturgeon overcame the problem where the motion of the magnets would stop once the poles repelled, leading to a revolutionary invention for all automation.

3.3.2 Operation of a DC Motor

Figure 3.11 shows that a DC motor consists of an armature, stator, commutator, and brushes. The stator, which is the stationary part of the motor, is built up of two magnets. The South pole of one magnet is pointed at the North pole of the other magnet which creates an external magnetic field across the armature (Sarkar, 2020). This magnetic field is permanent and flows from the North to the South pole.

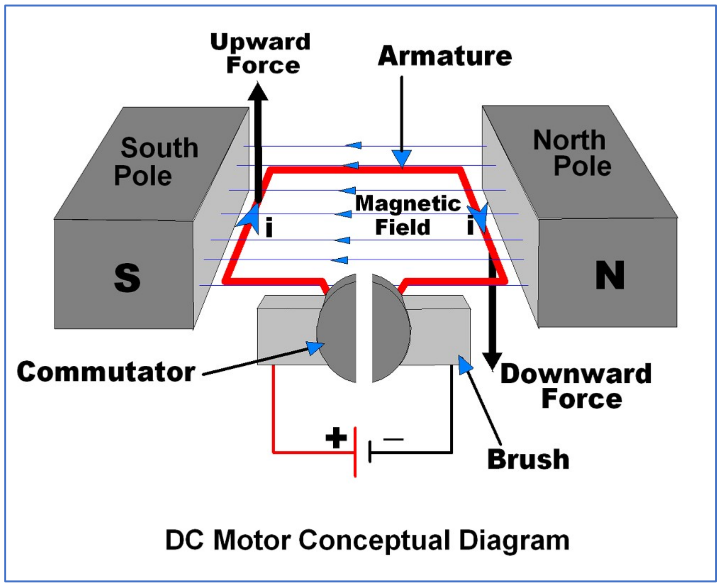


Figure 0.11: **Construction of a DC motor**

(Bzdigian, 2022)

When an electric current passes through the armature coil, it creates an electromagnetic force around the current-carrying conductor. When the magnetic field created by the armature is placed in the external magnetic field created by the stator, the armature starts to rotate. The direction of force is based on Fleming’s left-hand rule; hence the left side of the armature is pushed by an upward force and the right side of the armature moves down due to a downward force. The armature is also known as the rotor, as it is the rotating part of the motor. The electromagnetic force produced by the current flowing through the armature is known as the

Lorentz force (Britannica, 2023). The rotational field is produced by the charged particles traveling through the conductor. The magnitude of the force produced can be calculated with Equation (3.1).

$$\vec{F} = q(\vec{E} + \vec{v} \times \vec{B}) \quad \textbf{(3.1)}$$

Where,

- $\vec{F}$ = Magnetic Force
- $\vec{B}$ = Magnetic Field
- $\vec{E}$ = Electric Field
- q = Charge
- $\vec{v}$ = velocity of the charged particle

Therefore,

- $q * \vec{E}$ = electrical force on the charged particle
- $q * (\vec{v} \times \vec{B})$ = magnetic force of the charged particle

Hence the sum of the electrical force and magnetic force equals the force a charged particle experiences when placed in an electromagnetic field.

3.3.3 Construction of a DC Motor

The internal construction of a DC motor is presented in Figure 3.12. The most important internal components of the DC motor are the armature, brush, commutator, permanent magnet, and a shaft.

Through brushes, the commutator establishes a permanent electrical connection between the armature and the DC power source. This allows the armature to rotate freely without losing connection. The brushes are spring-loaded so that contact with the commutator is always maintained. The commutator also reverses the current direction to the armature to allow it to continue rotating in one direction. A single armature coil has two commutators attached to each respective end in the form of a split ring (Roderick, 2021).

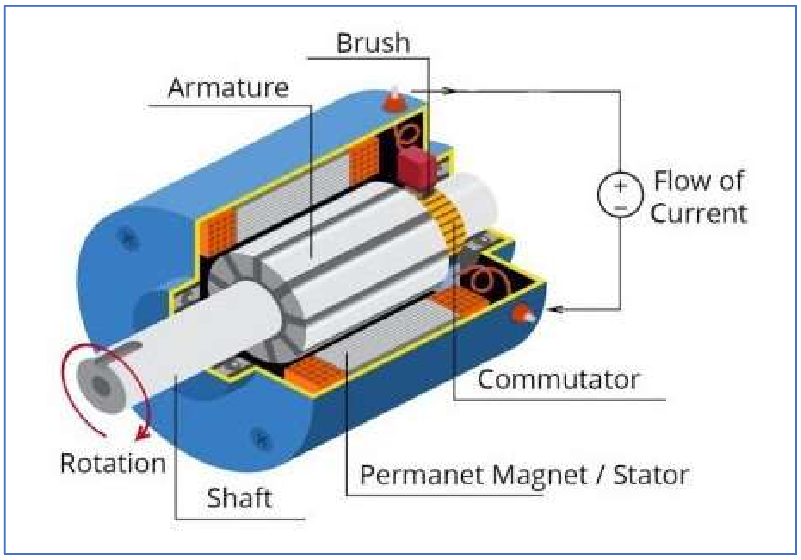


Figure 0.12: **Dissected view of a DC motor (Roderick, 2021)**

The current supplied by the DC supply flows through the wire, through the brush, through the commutator ring, forms a loop within the armature, and out through the second commutator, brush, and wire. As mentioned before, this causes an electromagnetic field across the armature. As the commutator spins, the gap between the commutator rings is reached and the direction of flow of current in the armature is switched, causing the polarity of the electromagnetic field to switch. The commutator ring has the same effect as switching the wires of the input DC supply.

The motor in Figure 3.12 has more than one set of commutator rings. Multiple commutator rings prevent the brushes from meeting halfway between commutator segments which can cause irregularities in the rotation of the armature. The shaft is also shown, which allows for the transfer of the produced mechanical energy to the process that needs to be rotated.

DC motors are very advantageous regarding torque capabilities (Tan Kiong Howe, 2003). The startup torque of these motors is very high, which is needed when speed must be constant and consistent in applications. Also, the relationship curve between speed and torque is more linear than in most other motors. Other advantages include no harmonic effects, quick installation, minimal maintenance, and low cost.

3.4 Discussions on Radio Antennas and DC Motors

This section discusses the use of DC motors being used in radio antennas control systems. Table 3.1 in Section 3.2.2 shows a few articles in which authors have studied different radio antenna control systems. In most cases, a DC motor is used in a feedback control system to allow for positional movements of the azimuth or altitude planes of the dish. Only one author did not use a controller in the feedback loop, but the others used controllers such as model predictive control, PID control, fuzzy logic control and state feedback control.

MATLAB/Simulink was used by all the authors to simulate their models. No physical systems were mentioned or built to compare results but there were a few mentions of future work being done on implementation of the designed systems. The next section will conclude the chapter by discussing and summarizing the main points regarding the theory of antennas and radio antennas.

3.5 Conclusion

This chapter gave a theoretical overview of antennas and the DC motor. Antenna theory is discussed, including a brief history, different types of antennas, and several radio antenna control systems. The multiple articles regarding radio antenna systems are discussed and compared to see which control systems were used more and which has better results. It was shown that in all the articles DC motors were used for the positional control of the radio antenna dish.

The DC motor operation, construction and history is also discussed. Finally, Section 3.4 discusses the correlation between the DC motor and the radio antenna to show the importance of the two for this research work. The use of DC motors in radio antenna systems is shown to be common in this chapter. The high torque-to-inertia ratio, cost effectiveness, and precision are all factors that motivate the use of a DC motor for positional control of the parabolic dish. The plant is considered linear and therefore the disturbance of wind and any other external factors will not be considered. The load, consisting of the dish, will be included in the system calculations by means of a gearbox model.

Chapter Four describes the formulation of the mathematical model of the plant, combining the electrical and mechanical components of the DC motors used for azimuth and altitude positional movements of the antenna dish.

CHAPTER 4
DEVELOPMENT AND SIMULATION OF THE PLANT

4.1 Introduction

The mathematical modelling of the DC motor and radio antenna system are discussed in this chapter. To develop an understanding of the system's behaviour driven by the DC motor, the antenna system model is evaluated. In control systems, it is necessary to first design and simulate the appropriate model of the plant before applying the controller to the actual hardware. The simulations are used to prove the behaviour of the system when parameters are changed. To model a system, it is necessary to find its transfer function (relating the output to the input) and then develop its MATLAB script or a Simulink block diagram. This allows for easier implementation of the practical model as many input sources and graphing tools are available to test the system during the simulation.

Radio antenna systems are non-linear because external factors such as wind usually influence the systems response. In this thesis the radio antenna is considered as linear when calculating the mathematical models, ignoring the external factors. The plant considered in this research has two DC motors controlling the antenna dish's azimuth and altitude positional movements. Therefore, it is necessary to model these DC motors to allow for simulations before applying the controller to the real hardware. A single DC motor is modelled, as the calculations can then be duplicated for the second DC motor.

The mathematical model of a DC motor is obtained in Section 4.2. This section describes the Modelling of a DC motor's electrical and mechanical components. Both models are combined to form the DC motor's electromechanical open loop transfer function. The state space representation of the DC motor is described in Section 4.3. The model of the antenna system is described in 4.4, followed by a conclusion in Section 4.5.

4.2 Modelling of a DC Motor

The electro-mechanical diagram of a DC motor can be seen in Figure 4.1. The system is comprised of a coupled electrical and mechanical subsystem. By using the laws of physics, it is possible to obtain the system equations. These system equations are then used to find the transfer function which is mathematically easier to work with and simulate. Obtaining the transfer function allows for the use of tools and graphing techniques in the MATLAB/Simulink software environment to help better understand the behaviour of the system.

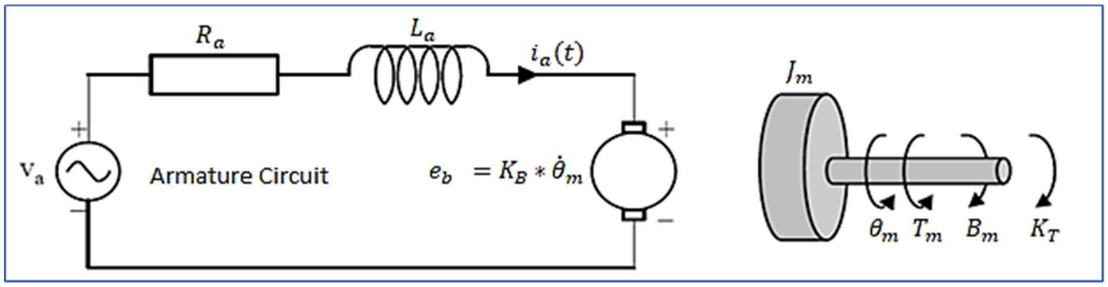


Figure 0.1 **Electro-mechanical diagram of a DC motor with a fixed magnetic field (Mikova et al, 2016)**

4.2.1 Transfer Function of the Electrical Component of a DC Motor

The electrical part of a DC motor is modelled using Kirchhoff's voltage law, which states that the sum of potential differences in a closed circuit is equal to zero (Iswanto et al, 2021). The balance equation is defined by

$$V_a(t) - R_a i_a(t) - L_a \frac{di_a}{dt} - e_b = 0 \qquad \textbf{(4.1)}$$

where V_a is the voltage source, R_a is the resistance of the armature, L_a is the inductance of the armature, e_b is the back Electromotive Force (emf) generated from the load, and $i_a(t)$ is the armature current that flows through the circuit with time.

From Equation 4.1, the voltage source is expressed as

$$\boldsymbol{V_a(t) = R_a i_a(t) + L_a \frac{di_a}{dt} + e_b} \qquad \textbf{(4.2)}$$

The back emf is expressed as

$$e_b \;= K_B * \dot{\theta}_m \qquad \textbf{(4.3)}$$

where K_B is the emf constant of the DC motor, and $\dot{\theta}_m$ is the angular velocity.

Substituting Equation 4.3 into Equation 4.2 yields

$$V_a(t) = R_a i_a(t) + L_a \frac{di_a}{dt} + K_B \dot{\theta}_m \qquad \textbf{(4.4)}$$

To solve for the transfer function of the DC motor, it is necessary to convert all differential equations to the frequency domain which allow for easier algebraic manipulation of equations. The conversion makes it easier to converge the electrical and mechanical equations into one equation and is necessary for conversion to block diagram form in Simulink. Taking the Laplace transform of Equation 4.4, we obtain

$$V_a(s) = R_a I_a(s) + L_a s I_a(s) + K_B s\theta_m(s) \tag{4.5}$$

where $\theta_m(s)$ is the angular position of the DC motor in the s-domain , and s is a complex frequency domain parameter.

Rearranging Equation 4.5 results in Equation 4.6.

$$V_a(s) - K_B s\theta_m(s) \ = I_a(s)(R_a + L_a(s)) \tag{4.6}$$

Equation 4.6 is used to find the transfer function of the electromechanical model of the DC motor. The next section determines the transfer function of the mechanical component of the DC motor.

4.2.2 Mechanical Transfer Function of a DC Motor

It is important to model the mechanical part of the DC motor because the output rotational movement of the shaft is related to the current input from the electrical part. A relationship between the electrical and mechanical parts must be found to calculate the complete transfer function of the system that can be used in the final model.

The electrical current flowing through the circuit causes a fixed magnetic field due to a magnet component in the DC motor. This magnetic field applies a force on the inertial mass that causes an input torque which is defined using Newton's 2nd law. The balance equation is defined by

$$T_m - B_m\dot{\theta}_m = J_m\ddot{\theta}_m \tag{4.7}$$

where T_m is the input torque to the load, B_m is the damping coefficient, $\dot{\theta}_m$ is the angular velocity of the DC motor, $\ddot{\theta}_m$ is the angular acceleration of the DC motor, and J_m is the initial moment of inertia.

The initial torque generated is expressed as

$$T_m = K_T * i_a(t) \tag{4.8}$$

where K_T is the DC motor torque constant, and $i_a(t)$ is the armature current that flows through the circuit with time.

Substituting Equation 4.8 into Equation 4.7 yields

$$K_T i_a(t) - B_m \dot{\theta}_m = J_m \ddot{\theta}_m \qquad \textbf{(4.9)}$$

This substitution is crucial as it allows the like term of $i_a(t)$ to be in both the electrical and mechanical balance equations. Taking the Laplace transform of Equation 4.9, to obtain

$$K_T I_a(s) - sB_m\theta_m(s) = J_m s^2 \theta_m \qquad \textbf{(4.10)}$$

Rearranging Equation 4.10 finds the balance equation of the mechanical part of the DC motor which is defined by

$$K_T I_a(s) = s(J_m s + B_m)\theta_m(s) \qquad \textbf{(4.11)}$$

Equation 4.6 and 4.11 describes the balance equations of the electrical and mechanical parts of the DC motor in the frequency domain. The next section will determine the open loop transfer function of the complete DC motor, consisting of both electrical and mechanical parts.

4.2.3 Open loop transfer function of DC motor without load

It is necessary to obtain the complete electro-mechanical transfer function of a DC motor to simulate the system. The electrical and mechanical parts' system equations, as described in Equations 4.6 and 4.11 respectively, are combined to formulate the full system. This is done by using the like term of $I_a(s)$ which is found in both transfer functions. By making $I_a(s)$ the subject of each formula and then making the two equations equal to each other Equation 4.12 is obtained.

$$V_a(s) - K_B s\theta_m(s) = \frac{(L_a s + R_a)(J_m s + B_m)}{K_T} * s\theta_m(s) \qquad \textbf{(4.12)}$$

Rearranging Equation 4.12 results in,

$$V_a(s) = \left[\frac{(L_a s + R_a)(J_m s + B_m) + K^2}{K_T}\right] * s\theta_m(s) \qquad \textbf{(4.13)}$$

Rearranging Equation 4.13 to find the transfer function for input acceleration to output voltage we obtain

$$\frac{\dot{\theta}_m(s)}{V_a(s)} = \frac{K_B}{(J_m s+B_m)(L_a s+R_a)+K_B*K_T} \qquad \textbf{(4.14)}$$

Since position control of a DC motor is the aim of this thesis, the relationship between position and voltage is considered. This is achieved by integrating both sides of Equation 4.14 to obtain

$$\frac{\theta_m(s)}{V_a(s)} = \frac{K_B}{s[(J_m s+B_m)(L_a s+R_a)+K_B*K_T]} \qquad \textbf{(4.15)}$$

Equation 4.15 is the input-over-output transfer function form that is needed when trying to model a system (voltage being the input and position being the output).

Considering that armature inductance L_a in a fixed motor is negligible,

$$\frac{\theta_m(s)}{V_a(s)} = \frac{K_B}{R_a J_m s^2+s[R_a B_m+K_B*K_T]} \qquad \textbf{(4.16)}$$

Simplifying Equation 4.16 finds the combined transfer function of the electrical and mechanical parts of the DC motor defined by:

$$\frac{\theta_m(s)}{V_a(s)} = \frac{\frac{K_B}{R_a}}{J_m s^2+s[\frac{R_a B_m+K_B*K_T}{R_a}]} \qquad \textbf{(4.17)}$$

To reduce the complexity of the transfer function to allow for easier modelling of the system, new variables K_m and a_m are substituted into Equation 4.17 to obtain

$$\frac{\theta_m(s)}{V_a(s)} = \frac{K_m}{s(s+a_m)} \qquad \textbf{(4.18)}$$

where

$$K_m = \frac{K_B}{R_a J_m} \qquad \textbf{(4.19)}$$

and

$$a_m = \frac{R_a B+K_B*K_T}{R_a J_m} \qquad \textbf{(4.20)}$$

Table 0.1: **DC motor parameters**

Parameters	Symbol	Values	Unit of Measurement
R_a	Armature resistance	14.3	ohm
L_a	Armature inductance	0	henry
K_B	Electromotive force constant	0.425	volt/(rad/sec)
K_T	Torque constant	2.3	N-m/ampere
J_a	Moment of inertia of the armature	0.013	Kg/ m²
J_L	Moment of inertia of the load	0.001	Kg/ m²
B_a	Motor damping coefficient	0.00001	Ns/m
B_L	Load damping coefficient	1	Ns/m
$N1$	Number of gears teeth N1	1	N-m
$N2$	Number of gears teeth N2	270	radians

The motor parameters shown in Table 4.1 is substituted into Equation 4.18 to obtain the complete transfer function for the specified DC motor. The motor used is a 6000 RPM 12V DC motor with a 1:270 gearbox. The load moment of inertia and damping ratios are from the incremental encoder that is mechanically connected to the shaft of the motor through a 1:1 gear ratio.

The total moment of inertia is defined as

$$J_m = J_a + J_L * \left(\frac{N_1}{N_2}\right)^2 \qquad \textbf{(4.21)}$$

where J_a is the moment of inertia of the armature, J_L is the moment of inertia of the load, N_1 is the number of gear teeth on the motor side of the gearbox, and N_2 is the number of gear teeth on the load side of the gearbox. Substituting the parameters from Table 4.1 yields

$$J_m = 0.013 \quad \text{Kg-m}^2\text{/rad} \qquad \textbf{(4.22)}$$

The total damping coefficient is defined as

$$B_m = B_a + B_L * (\frac{N_1}{N_2})^2 \qquad \textbf{(4.23)}$$

where B_a is the motor damping coefficient, and B_L is the load damping coefficient. Substituting the parameters from Table 4.1 yields

$$B_m = 0.00002 \quad \text{N-m/(rad/sec)} \qquad \textbf{(4.24)}$$

The values for the armature resistance, the armature inductance, the electromotive force constant, the torque constant, the moment of inertia of motor and load, and the damping ratio of mechanical system, are substituted into equations 4.19, 4.20 to obtain

$$K_m = \frac{0.425}{14.3 * 0.013} \tag{4.25}$$

$$K_m = 2.2862 \tag{4.26}$$

$$a_m = \frac{14.3 * 0.00002 \quad (0.425 * 2.3)^2}{14.3 * 0.013} \tag{4.27}$$

$$a_m = 5.26 \tag{4.28}$$

Finally, the values for K_m and a_m are substituted into Equation 4.18 to obtain the final DC motor transfer function defined as

$$\frac{\theta(s)}{V_a(s)} = \frac{5.26}{s(s+2.2862)} \tag{4.29}$$

The open loop transfer function is simulated and validated in the next section using MATLAB and Simulink.

4.2.4 Simulation and Validation

The open loop response of the DC motor without load is simulated in MATLAB and Simulink. The calculations to determine the values for K_m and a_m are done in MATLAB and can be seen in Appendix A4.1. The transfer function is configured in Simulink using a transfer function block as shown in Figure 4.2. A step input is used to test the response of the system. The step value is 1 and the time at which the step occurs is at 1 second. Figure 4.3 shows the graphical open loop response from the scope in Simulink.

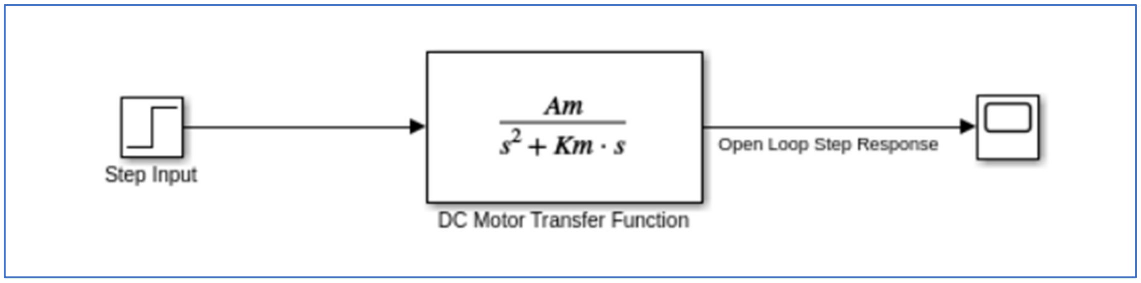


Figure 0.2: **Simulink block diagram of a step input to a DC motor transfer function**

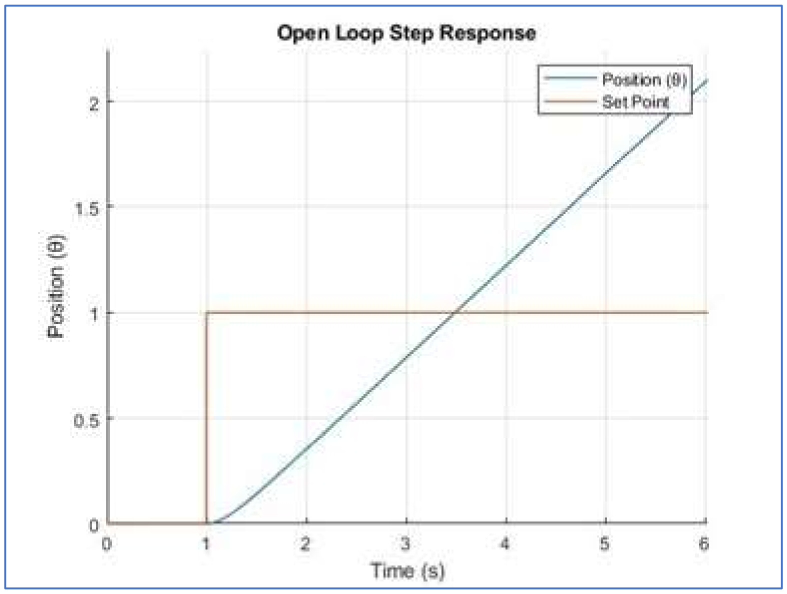


Figure 0.3: **Open loop step response of a DC motor**

The response in Figure 4.2 illustrates an exponential increase in the motor position. This occurs because the measured position is not fed back into the control system. A closed loop system with measured position feedback is discussed in section 4.4. This thesis focuses on modern control theory and therefore the state space representation is required before the controller can be developed. The next section will formulate the state space equations of a DC motor.

4.3 State space representation of open loop system of a DC motor

Working with differential equations and transfer functions when trying to model systems becomes more difficult as the systems become more complex. Therefore, it is necessary to use state space representation to allow for easier design and modelling of systems. State space representation of a physical system consists of a set of inputs, outputs and state variables in a mathematical model which is related by first order differential equations (Rowell, 2002). The states which change with time in a DC motor are position, velocity, and armature current. In this case, only position and velocity are considered. The state space equations of a plant is expressed as

$$\dot{x} = Ax + Bu \tag{4.30}$$

$$y = Cx + Du \tag{4.31}$$

where A is the state matrix, B is the input matrix, C is the output matrix, and D is the feedthrough matrix. The A matrix captures the dynamics of the linear system, which includes how the energy of the system is captured, stored, and moved. The B matrix determines how the system responds to inputs. All four matrices can be calculated by finding the derivatives of the states of the system.

Rearranging Equation 4.18, the angular position is obtained as shown in Equation 4.32.

$$\theta_m(s) = \frac{K_m}{s(s+a_m)} * V_a(s) \quad \textbf{(4.32)}$$

Taking the inverse Laplace transform of Equation 4.32:

$$s^2\theta_m(s) + s\theta_m(s)a_m = K_m V_a(s) \quad \textbf{(4.33)}$$

This transformation is done as the state space representation requires the model of the plant to be in the time domain. Rearranging Equation 4.33 results in the angular acceleration being obtained as shown in Equation 4.34.

$$\ddot{\theta}_m(s) = -\dot{\theta}_m(s)a_m + K_m V_a(s) \quad \textbf{(4.34)}$$

The states of the system are expressed as

$$y = \theta_m = X_1 \quad \textbf{(4.35)}$$

and

$$\dot{y} = \dot{\theta}_m = \dot{X}_1 = X_2 \quad \textbf{(4.36)}$$

and

$$\ddot{y} = \ddot{\theta}_m = \ddot{X}_1 = \dot{X}_2 = -\dot{\theta}_m(s)a_m + K_m V_a(s) \quad \textbf{(4.37)}$$

Where X_1 and X_2 are vector components. These components make it easier to formulate the derivatives of the states. Once all the derivatives are found, it is possible to find the state space equations by making the derivatives the subject of the formula expressed as

$$\dot{X}_1 = X_2 \quad \textbf{(4.38)}$$

and

$$\dot{X}_2 = -X_2 a_m + K_m V_a(t) \tag{4.39}$$

The output equation is expressed as

$$y = X_1 \tag{4.40}$$

Therefore, the matrices of the state space models are expressed as

$$, \quad A = \begin{bmatrix} 0 & 1 \\ 0 & -a_m \end{bmatrix}, B = \begin{bmatrix} 0 \\ K_m \end{bmatrix}, \quad C = [1 \quad 0]\begin{bmatrix} 0 \\ K_m \end{bmatrix}, \quad D = 0$$

Finally, the state space model of the plant is expressed as

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -a_m \end{bmatrix}\begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ K_m \end{bmatrix} u(t) \tag{4.41}$$

$$y(t) = [1 \quad 0]\begin{bmatrix} 0 \\ K_m \end{bmatrix} \tag{4.42}$$

Substitute the values for a_m and K_m for the complete state space equation of an open loop DC motor we obtain

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -5.26 \end{bmatrix}\begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 2.2862 \end{bmatrix} u \tag{4.43}$$

$$y(t) = [1 \quad 0]\begin{bmatrix} 0 \\ 2.2862 \end{bmatrix} \tag{4.44}$$

4.3.1 Simulation and Validation

The open loop response of the DC motor without a load in the state space form is simulated in MATLAB and Simulink. The values for matrix A, B, and C are calculated in MATLAB and can be seen in Appendix A4.2. The state space equations are illustrated in Simulink using a gain block for the matrixes, an integrator to convert between states, and a summing junction that adds matrix A as feedback into the system. A step input is used to test the response of the system. The step value is 1 and the time at which the step occurs is at 1 second. Figure 4.5 shows the graphical open loop response from the scope in Simulink.

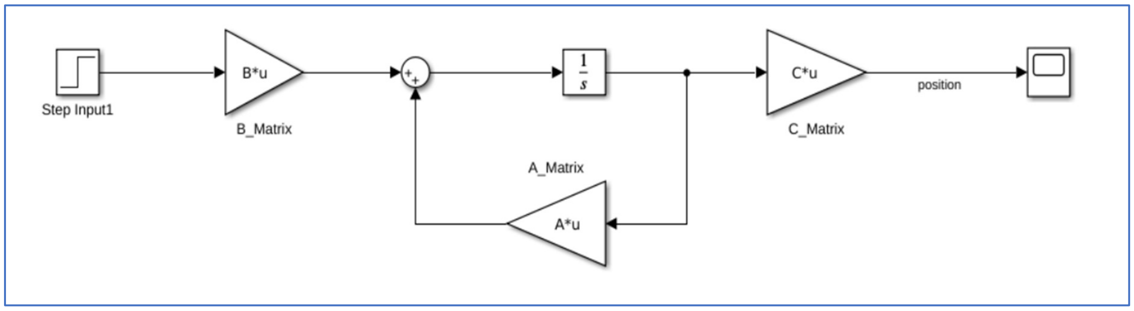


Figure 0.4: **Simulink block diagram of a step input to a DC motor that is represented in state space form**

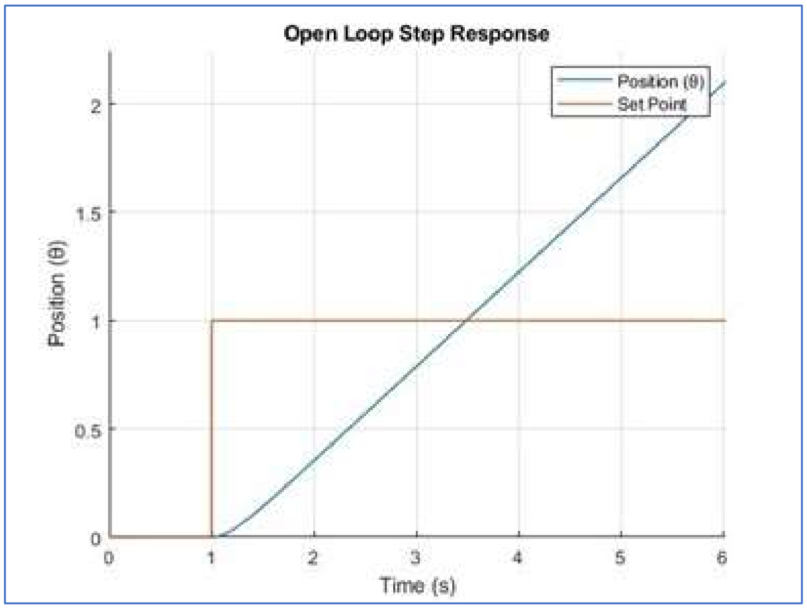


Figure 0.5: Open loop step response of a DC motor that is represented in state space form

The open loop step response of the DC motor in state space form is the same as the open loop step response of the DC motor transfer function shown in Figure 4.3. This verifies that the conversion to state space form is correct and that the state space equations can now be used in future models and simulations. The next section describes the full model of the plant.

4.4 Model of a radio antenna control system

The model for the radio antenna consists of two separate models for the azimuth and altitude positional movements. The load of the antenna is not included in the block diagram as the gear ratios are already included in the transfer function. The addition of the postion output being fed back into the system through a summing junction, as shown in Figure 4.6, represents the encoder position feedback from the motor. The difference between the input and the output is the steady state error, which causes the system to move towards the set point. Equations 4.43 and 4.44 represent the state space model of the closed loop system.

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}-5.26 & -2.2862\\ 1 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}2\\ 0\end{bmatrix}u(t) \quad \textbf{(4.43)}$$

$$y(t) = \begin{bmatrix}0 & 1.1431\end{bmatrix} \quad \textbf{(4.44)}$$

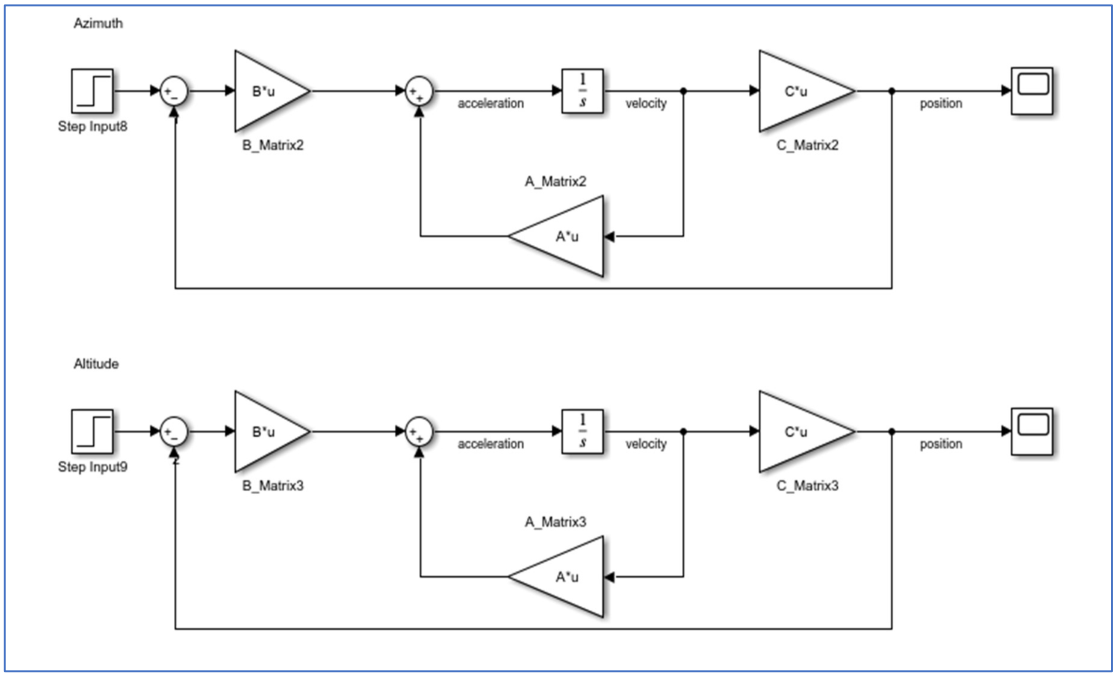


Figure 0.6: **Simulink block diagram of the radio antenna control system**

A step input of 1 with an initial value of 0 is used to see the response of both DC motors. The outputs, showing the angular position of the motor, shows the same results on both scopes as the same motors and gear ratios are used for both the altitude and azimuth movements. As seen in Figure 4.7, the system takes 12 seconds to reach the set position and achieve a steady state.

.

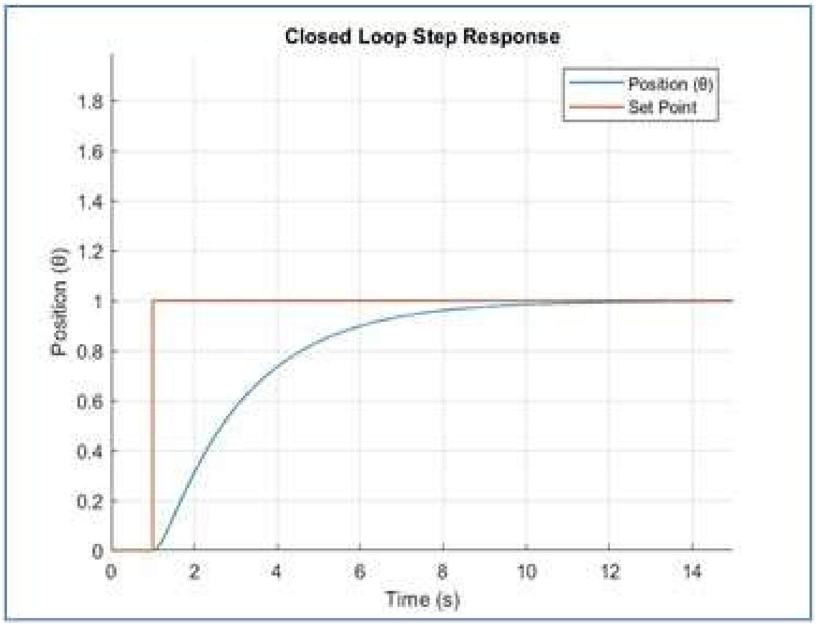


Figure 0.7: **Step response of the radio antenna plant with step value of 1**

Another step input with the value 10 is also used to validate the response of the closed-loop system. As seen in Figure 4.8, the new set point also takes 28 seconds to reach and achieve a steady state. This response is very slow and can be compensated by developing and using a controller to decrease the rise time.

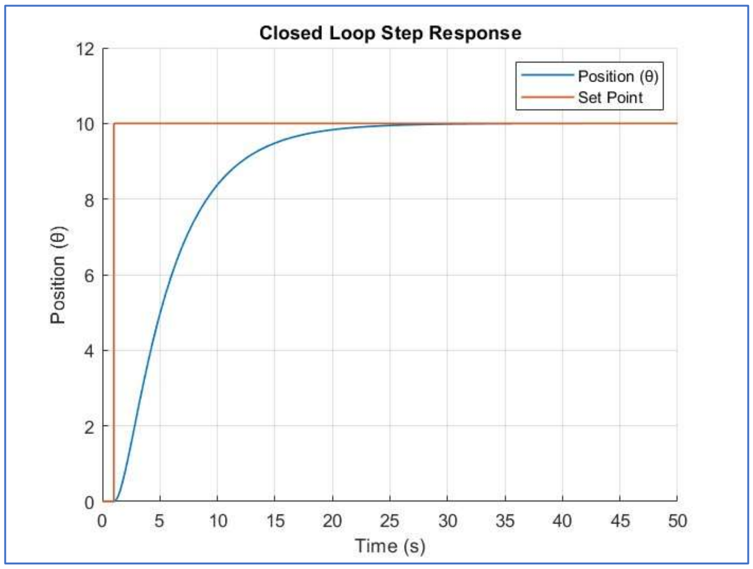


Figure 0.8: **Step response of the radio antenna plant with step value of 10**

4.5 Conclusion

The mathematical model of the DC motors used for the positional movements of the dish is presented in this chapter. The models are derived using the laws of physics, namely Kirchhoff’s voltage law and Newton’s law of motion. These system equations of the electrical and mechanical components of the motor are then substituted and rearranged to derive the electromagnetic model. This model is represented in both the transfer function and the state space form.

The open loop step response of the motor is simulated using MATLAB/Simulink. The response shows an unstable system as the set position is never reached over time. Both the transfer function and state space representation of the open loop step response have the same results which verifies that the state space conversion was correct. The state space representation is required in the next chapter for the controller design.

The radio antenna plant is also modelled and tested by means of a step response. It is shown that the system is stable but that the system performance is very slow. The next chapter focuses on the design of the controller for the plant to decrease the overall rise time. A state feedback gain controller with integral action and observer is proposed. The issue of network induced delays is also investigated.

CHAPTER 5
STATE FEEDBACK CONTROLLER DESIGN

5.1 Introduction

This chapter is based on state feedback controller design techniques. This technique is generally used to shape the system's behavior and dynamics to achieve desired performance and stability. It is used to design control systems that provide control inputs based on the full state information of a dynamic system. A state feedback controller allows the movement of the system poles to any desired location. Moving the poles of a system results in different responses in factors such as rise time, overshoot, frequency, gain, and settling time. Poles can be moved from the right-hand plane to the left, causing the system to become stable. This is done by multiplying each state by a certain gain K and feeding the result back into the system. Many advantages such as reduction in design complexity, reduction in hardware, and overall adaptivity are a direct result of applying a state feedback controller to a system (Ruderman et al, 2018).

This study focuses on modern controller design techniques using a state feedback controller approach. The designed parameters are later used to control the position movement of a radio antennas azimuth plane. The step response of the closed-loop system as discussed in the previous chapter is fundamentally used to prove the requirement for the designed controller. The stability of the system is proven, as there is no steady-state error, but the time taken by the system to get to a steady state is too long. To decrease the rise time, it is necessary to introduce a controller, hence the introduction of this chapter Each set of motors will have a state feedback controller; therefore, this chapter only discusses the controller design for the motor used for Azimuth position movement. The same controller is then used for the Altitude motor.

A state feedback controller is designed in Section 5.2. The addition of integral control to the controller is implemented in Section 5.3. Observer control for the current controller is described in Section 5.4. The effects of network delays in the developed control system are discussed in Section 5.5, followed by the conclusion in Section 5.6.

5.2 State Feedback Controller Design Using Pole-Placement Approach

The pole placement technique is used to change the behaviour of a system by moving its poles to a desired location. As indicated in the previous chapter the current system poles are situated at a position that causes the system to have a slow rise time. The graphical record based on the step response as presented in Chapter 4 are the results of the poles of the system that are situated at -0.486 and -4.78 as shown in Figure 5.1. To achieve a faster rise time, it is necessary to move these poles further into the left-hand side of the s-plane. Figure 5.1 shows the MATLAB output of the root locus of the DC motor closed-loop system described by the state space equations that are formulated in the previous chapter.

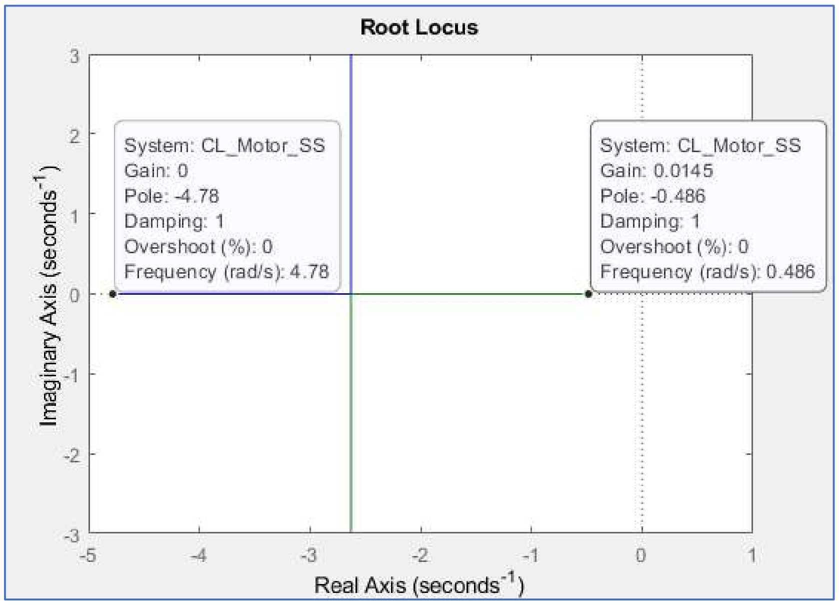


Figure 0.1: **Root Locus of the closed-loop system**

The following subsections describe the process involved in designing a state feedback controller with pole placement feedback. The control law for state feedback is first described followed by a test to see if the system is controllable. The state feedback gain is then calculated and tested through simulations.

5.2.1 Control Law

To determine how a system responds to input signals and regulates its output signals to achieve its desired objective, it is necessary to determine its control law. The control law is a mathematical representation that shows the behaviour of a control system. Developing a control law is very beneficial when designing a system, performing optimization, analysing stability, testing model compatibility, and preventing rework. Determining the control law before designing the control system establishes a solid foundation for the overall system performance. The state space equations for a plant is defined as

$$\dot{x} = Ax + Bu \quad \textbf{(5.1)}$$

$$y = Cx + Du \quad \textbf{(5.2)}$$

where A is the state matrix, B is the input matrix, C is the output matrix, and D is the feedthrough matrix.

The first step is to determine the control law for a full-state feedback controller. Because the system is regarded as linear, it can be said that the C matrix is equal to the identity matrix, and the D matrix is equal to zero. Therefore, by inserting this logic into Equation 5.2,

$$y(t) = x(t) \tag{5.3}$$

Therefore, the control law for a full state feedback controller is given by

$$u(t) = -Kx(t) \tag{5.4}$$

where K is the gain matrix.

Substituting Equation 5.4 into equation 5.1 results in Equation 5.5.

$$\dot{x}(t) = Ax(t) + B(-Kx(t)) \tag{5.5}$$

Rearranging Equation 5.5 the closed-loop state space representation of the system is obtained as shown in Equation 5.6.

$$\dot{x}(t) = (\mathrm{A} - BK)x(t) \tag{5.6}$$

The A matrix in a closed loop system is defined as

$$A_{CL} = A - BK \tag{5.7}$$

where A_{CL} is the closed-loop A matrix

Substituting Equation 5.7 into Equation 5.6 yields

$$\dot{x}(t) = A_{CL}.x(t) \tag{5.8}$$

This closed-loop A matrix governs the behaviour of the system as changing the value of K results in the movement of the eigenvalues. In the next subsection, the first task is to evaluate the controllability of the control system before using the state feedback controller to optimize the response of the system.

5.2.2 Controllability

A DC motor system model needs to be in a controllable state to use a state feedback controller with pole placement (Iswanto et al, 2021). To check controllability, it is necessary to test if the system model is already in controllable canonical form. The test for controllability is expressed as

$$P_C = [A \quad AB] \tag{5.9}$$

where P_C is the controllability matrix.

Substituting the A and B matrices of the closed-loop system into Equation 5.9 yields

$$P_C = \left[\begin{bmatrix} -5.26 & -2.2862 \\ 1 & 0 \end{bmatrix} \quad \begin{bmatrix} -5.26 & -2.2862 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix}\right] \tag{5.10}$$

Simplifying Equation 5.10 results in Equation 5.11.

$$P_C = \begin{bmatrix} 2 & -10.5201 \\ 0 & 2 \end{bmatrix} \tag{5.11}$$

As shown in Equation 5.11, the second row of the matrix P_C is not dependent on the first row and therefore the rank is 2, proving that the system is controllable.

The DC motor system model is proven to be controllable and therefore a controller can be designed to change the response of the system. The next section uses the state feedback control law and the positions of the desired poles to solve for the gain matrix K.

5.2.3 Solve for Gain Matrix K

This section describes how to solve for the gain matrix K when designing a state feedback controller. The value of the gain matrix K is used to move the poles of a system to a desired location. It is necessary to move the poles of the system to decrease the current system's rise time. The gain matrix K is multiplied by the input matrix B, and then subtracted from the state matrix A. This shows that the value of K has a direct effect on the states of the system.

Before moving the poles to a desired location, it is necessary to find the characteristic equation of the closed loop system with state feedback control. This is done by finding the value of matrix A_{CL} and then finding the eigenvalues of the closed-loop matrix. The values for matrices A, B, and K are substituted into Equation 5.7 to obtain

$$A_{CL} = \begin{bmatrix} -5.26 & -2.2862 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix} [K_1 \quad K_2] \tag{5.12}$$

where K_1 and K_2 are vectors of the matrix K.

Simplifying Equation 5.12 through matrix multiplication Equation 5.13 is obtained.

$$A_{CL} = \begin{bmatrix} -5.26 & -2.2862 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 2K_1 & 2K_2 \\ 0 & 0 \end{bmatrix} \tag{5.13}$$

Simplifying Equation 5.13 through matrix subtraction results in Equation 5.14.

$$A_{CL} = \begin{bmatrix} -5.26 - 2K_1 & -2.2862 - 2K_2 \\ 1 & 0 \end{bmatrix} \tag{5.14}$$

The next step is to find the eigenvalues of matrix A_{CL}. The formula used to calculate the eigenvalues of a system is expressed as

$$0 = \det(\lambda I - A_{CL}) \tag{5.15}$$

where I is a 2x2 identity matrix, and λ is a mathematical constant.

Substituting Equation 5.14 into 5.15 yields

$$0 = \det\left(\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} -5.26 - 2K_1 & -2.2862 - 2K_2 \\ 1 & 0 \end{bmatrix}\right) \tag{5.16}$$

Simplifying Equation 5.16,

$$0 = \det\left(\begin{bmatrix} \lambda - (-5.26 - 2K_1) & -(-2.2862 - 2K_2) \\ -1 & \lambda \end{bmatrix}\right) \tag{5.17}$$

Simplifying Equation 5.17,

$$0 = \det\left(\begin{bmatrix} \lambda + 5.26 + 2K_1) & 2.2862 + 2K_2) \\ -1 & \lambda \end{bmatrix}\right) \tag{5.18}$$

Finding the determinant of Equation 5.18 results in Equation 5.19.

$$0 = (\lambda + 5.26 + 2K_1)\lambda - (-1((2.2862 + 2K_2)) \quad \textbf{(5.19)}$$

Simplifying Equation 5.19 obtains the characteristic equation of the closed-loop system and is expressed as

$$0 = \lambda^2 + 5.26\lambda + 2K_1\lambda + 2.2862 + 2K_2 \quad \textbf{(5.20)}$$

To achieve a faster rise time, the poles -2 and -6 are selected. These poles are known as the desired poles and is expressed as the desired characteristic equation

$$0 = \lambda^2 + 8\lambda + 12 \quad \textbf{(5.21)}$$

The values for K_1 and K_2 are solved by equating the coefficients of the like terms in the closed loop and desired characteristic equations shown in Equations 5.20 and 5.21. The equated characteristic equations are expressed as

$$5.26 + 2K_1 = 8 \quad \textbf{(5.22)}$$

and

$$2.2862 + 2K_2 = 12 \quad \textbf{(5.24)}$$

Substituting K_1 and K_2 into the gain matrix K yields

$$K = [1.37 \quad 4.8569] \quad \textbf{(5.25)}$$

Substituting the gain matrix K back into Equation 5.14, the closed-loop A matrix is expressed as

$$A_{CL} = \begin{bmatrix} -8 & -12 \\ 1 & 0 \end{bmatrix} \quad \textbf{(5.26)}$$

The state space model of the closed-loop system with full state feedback controller is expressed as

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -8 & -12 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} u \quad \textbf{(5.27)}$$

$$y(t) = [0 \quad 1.1431] \quad \textbf{(5.28)}$$

The new closed loop system shown in Equations 5.27 and 5.28 has poles at the desired location to achieve a faster rise time. The next section presents the simulation and validation of the behaviour and response of the closed loop system using MATLAB/Simulink.

5.2.4 Simulation and Validation

This section presents the simulation and performance evaluation of the DC motor control system being controlled with a state feedback controller. It is necessary to simulate and validate the state space equations that are developed in the previous section to prove that the equations are correct before applying the controller to a real-world system.

The MATLAB software environment is used to define and calculate the closed loop state space representation of the system. The software is also used to prove controllability, define the desired poles, calculate the values of K using Ackerman's formula, and lastly determining the new state space model of the system. Appendix A5.1 presents the MATLAB code used to find the new state space model or gain control model parameters. The new root locus response can be seen in Figure 5.2, showing the desired poles at -2 and -6. Therefore, it is proven that the calculated values of the gain matrix K are correct.

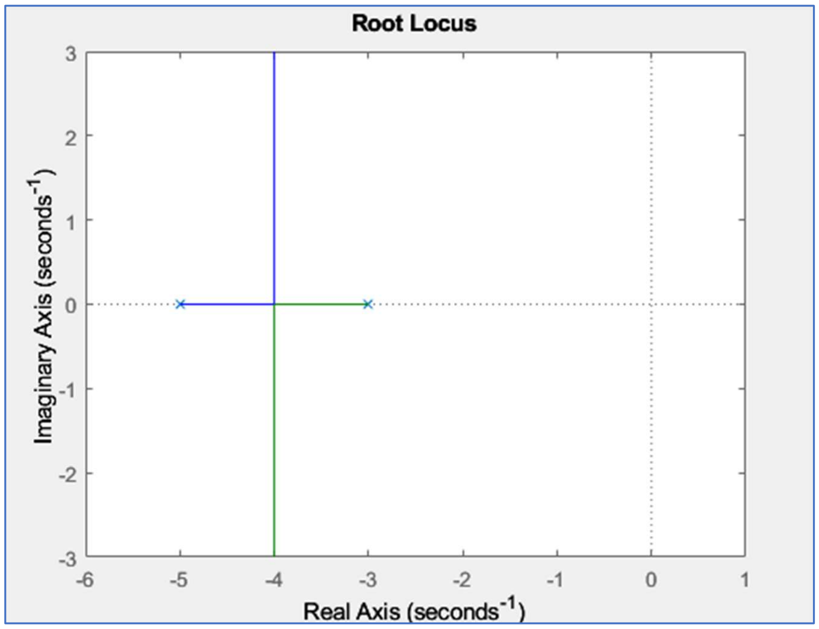


Figure 0.2: Root locus of closed loop system with new desired poles

The Simulink block diagram can be seen in Figure 5.3. The state gain matrix K is shown being multiplied by the output of the integrator, which is also the states of the system. The states multiplied by K are then subtracted from the input and multiplied by the closed loop matrix B_{CL}.

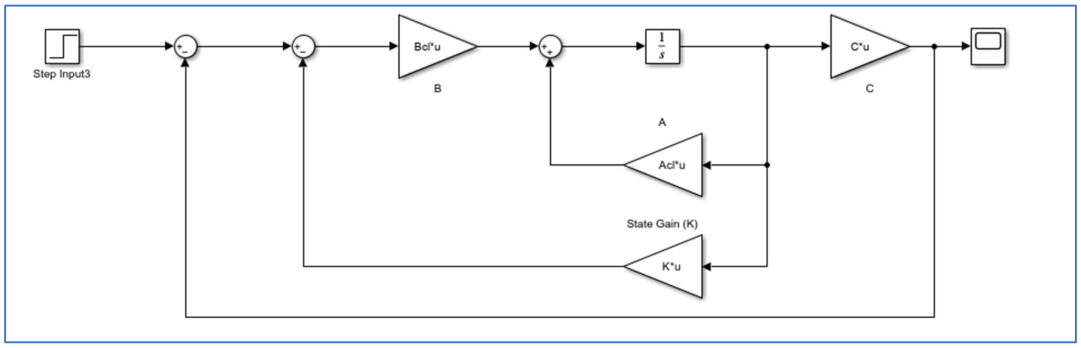


Figure 0.3: **Simulink block diagram of closed loop system with state feedback controller**

The considered closed loop system with state feedback controller is simulated and the step response is shown in Figure 5.4. The rise time has decreased from 28 seconds to 4 seconds. Hence the state feedback controller made the system react 7 times faster than the open loop system. Note that the position reached at a steady state is not the same as the unit step input which has the value of 1. The steady-state error that is present is the main disadvantage of using a full-state feedback controller.

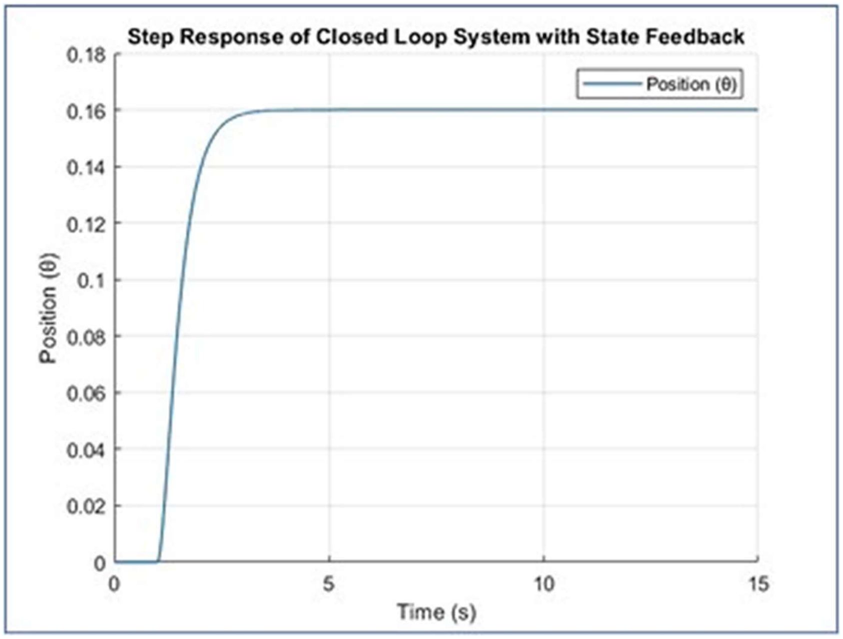


Figure 0.4: **Step response for closed loop system with state feedback controller**

The next section shows how to compensate for the error in position by applying integral control to the controller.

5.3 State Feedback with Integral Control

This section focuses on designing a controller that can compensate for the steady-state error encountered in the results in Section 5.2. The step response shown in Figure 5.4 shows that the introduction of full state feedback control has the disadvantage of having a huge steady state error. Steady state error can cause low accuracy as the set point of the controller is never reached. The system considered in this research requires zero steady state error; therefore, it is necessary to add integral control to the controller to reduce the current steady state error.

The following subsections describe the process to be followed when designing a state feedback controller with integral control. The new control law for state feedback is first described. The state feedback gain and integral gain is then calculated and tested through simulations.

5.3.1 Control Law

Integral control feeds the error between the input $(r(t))$ and output $(-Cx(t))$ of the plant back into the system through an integrator. This new control structure adds an additional state, x_i, to the system due to the output from the integrator (Equation 5.32). The new state is then multiplied by a gain K_i and subtracted from the full-state feedback controller's input. The equation for the new state is shown is expressed as

$$\dot{x}_i = -Cx(t) + r(t) \tag{5.29}$$

where $r(t)$ is the input to the controller and $\dot{x}_i$ is the input state of the integrator.

The new control law is expressed as

$$u(t) = -Kx(t) + K_i x_i(t) \tag{5.30}$$

where K_i is the integral gain and $x_i(t)$ is the output state of the integrator

Substituting Equation 5.30 into the output equation of a state space system, the state space model of the plant with full statefeedback and integral control is expressed as

$$\dot{x} = Ax(t) + B(-Kx(t) + K_i x_i(t)) \tag{5.31}$$

Simplifying Equation 5.31,

$$\dot{x} = (A - BK)x(t) + BK_i x_i(t) \tag{5.32}$$

Therefore, the state space model of a closed-loop system with a full state feedback controller and integral control is expressed as

$$\begin{bmatrix} \dot{x} \\ x_n \end{bmatrix} = \begin{bmatrix} A - BK & BK_i \\ -C & 0 \end{bmatrix} \begin{bmatrix} x \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(t) \tag{5.33}$$

$$y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ x_n \end{bmatrix} \tag{5.34}$$

The next subsection solves for the new state feedback gain K and the integral control gain K_i to allow for the movement of the system poles to a desired location and remove the steady state error. Once the values for the gains are solved, the system is tested in MATLAB/Simulink.

5.3.2 Solve for Gains K and K_i

The previous controller design without integral control is not sufficient as the result showed a huge steady state error. Therefore, it is necessary to add integral control into the system to decrease the current steady state error. Integral control is added to the system by integrating and multiplying the input matrix B with a gain value K_i.

It is necessary to solve for K and K_i to achieve the desired transient response of the DC motor system. This is done by moving the poles to desired locations. Before moving the poles, it is necessary to find the characteristic equation of the closed loop system with state feedback and integral control. This is done by finding the value of Acl, Bcl and C, then substituting these into the state space model.

Acl is expressed as

$$Acl = A - BK = \begin{bmatrix} -5.26 - 2K_1 & -2.286 - 2K_2 \\ -1 & 0 \end{bmatrix} \tag{5.35}$$

Bcl is expressed as

$$Bcl = BK_i = \begin{bmatrix} 2 \\ 0 \end{bmatrix} * K_i \tag{5.36}$$

C is expressed as

$$C = [0 \quad 1.1431] \tag{5.37}$$

Substituting Equations 5.35, 5.36, and 5.37 into 5.33 yields

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ x_i \end{bmatrix} = \begin{bmatrix} -5.26 - 2K_1 & -2.286 - 2K_2 & 2K_i \\ 1 & 0 & 0 \\ 0 & 1.1431 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_i \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(t) \tag{5.38}$$

The next step is to find the eigenvalues of matrix A_{CL}. Substituting into Equation 5.15 results in Equation 5.39.

$$0 = det\left(\begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} - \begin{bmatrix} -5.26 - 2K_1 & -2.286 - 2K_2 & 2K_i \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\right) \tag{5.39}$$

Simplifying Equation 5.39,

$$0 = det\left(\begin{bmatrix} \lambda + 5.26 + 2K_1 & 2.286 + 2K_2 & -2K_i \\ -1 & \lambda & 0 \\ 0 & -1.1431 & \lambda \end{bmatrix}\right) \tag{5.40}$$

Finding the determinant of Equation 5.40 yields

$$0 = \lambda^3 + 5.26\lambda^2 + 2\lambda^2 K_1 + 2.286\lambda + 2\lambda K_2 - 2.2862K_i \tag{5.41}$$

The order of the system has increased due to the addition of another state after the integrator output. Therefore, another pole must be added when calculating the desired characteristic equation. The poles -2, -6 and -8 are chosen to achieve a faster rise time. The desired characterstic equation is expressed as

$$0 = \lambda^3 + 16\lambda^2 + 76\lambda + 96 \tag{5.42}$$

The values for K_1, K_2 and K_i are solved by equating the coefficients of the like terms in the closed loop and desired characteristic equations shown in Equations 5.41 and 5.42. The equated characteristic equations are expressed as

$$16 = 5.26 + 2K_1 \tag{5.43}$$

and

$$76 = 2.286 + 2K_2 \quad \textbf{(5.44)}$$

and

$$96 = -2K_i \quad \textbf{(5.45)}$$

Therefore the values for K and gain K_i are expressed as

$$K = \begin{bmatrix} 5.37 \\ 36.8569 \end{bmatrix} \quad \textbf{(5.46)}$$

$$K_i = -41.9915 \quad \textbf{(5.47)}$$

Substituting the gain matrix K and gain K_i back into Equation 5.38 to get the state space model of the closed loop system with full state feedback and integral control yields

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \theta_i \end{bmatrix} = \begin{bmatrix} -5.26 & -2.2862 & 0 \\ 1 & 0 & 0 \\ 0 & -1.1431 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \theta_i \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} r(t) \quad \textbf{(5.48)}$$

$$y(t) = \begin{bmatrix} 0 & 1.1431 & 0 \end{bmatrix} \begin{bmatrix} x \\ x_n \end{bmatrix} \quad \textbf{(5.49)}$$

The new closed loop system shown in Equations 5.48 and 5.49 has poles at the desired location to achieve a faster rise time and no steady state error. The next section presents the simulation and validation of the behaviour and response of the closed loop system using MATLAB/Simulink.

5.3.3 Simulation and Validation

This section presents the simulation and performance evaluation of the DC motor control system being controlled with a state feedback controller with integral control. It is necessary to simulate and validate the state space equations that are developed in the previous section to prove that the equations are correct before applying the controller to a real-world system. MATLAB functions are used to define and calculate the closed loop state space representation of the system. The software is also used to determine the desired poles, calculate the values of gain matrix K and gain K_i using Ackerman's formula, and lastly display the root locus of the system. The root locus response can be seen in Figure 5.5, showing the desired poles at -2, -6 and -8. Therefore, it is proven that the calculated values of gain matrix K and gain K_i are correct.

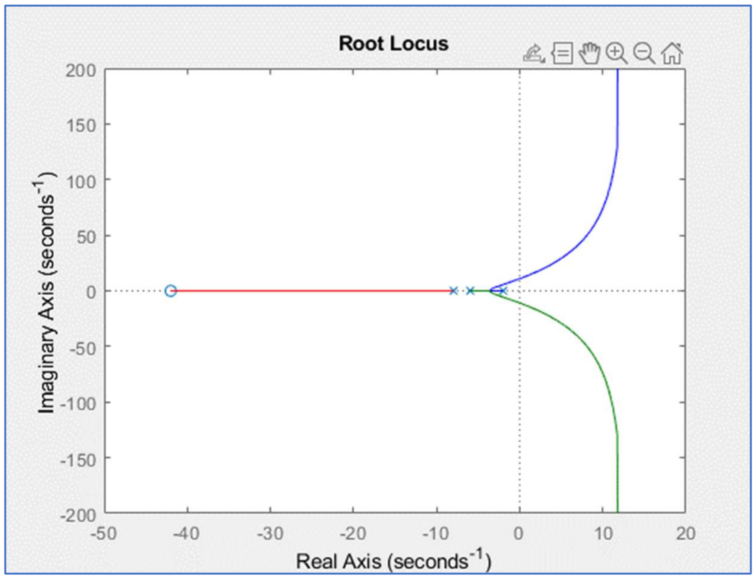


Figure 0.5: **Root locus of closed loop system with desired poles at -3, -5, and -20**

The Simulink block diagram can be seen in Figure 5.6. The state gain matrix K is shown being multiplied by the output of the integrator, which is also the states of the system. The states multiplied by K are then subtracted from the input and multiplied by closed loop matrix B_{CL}. The addition of the integral gain is shown multiplied by an integrator before the summing junction.

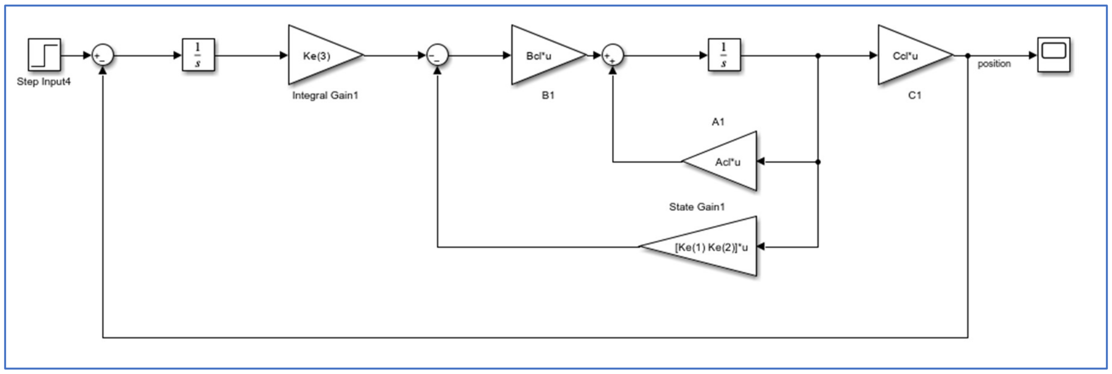


Figure 0.6: **Simulink block diagram of a state feedback controller with integral control**

The step response of the plant with full state feedback and integral control is shown in Figure 5.7. The step input of value 1 is reached in the same rise time as the state feedback controller without integral control but with zero steady state error. The resultant controller has achieved zero steady state error; therefore, the addition of integral control has worked.

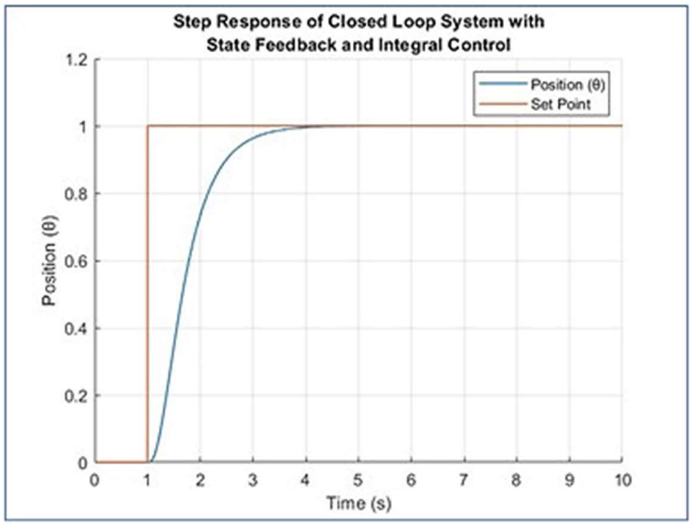


Figure 0.7: **Step response state feedback controller with integral control**

The next subsection focuses on a case study testing the response of the system. Response factors such as rise time, settling time, steady state-state error and overshoot are compared between different system inputs to test the performance and stability of the developed full state feedback controller with integral action.

5.3.4 Case Study

This subsection is composed of a case study comparing the different systems, with and without controllers, that have been developed in this research so far. The controllers are the same for both the azimuth and altitude position systems and therefore the results can be used for either. This case study only looks at the control of one DC motor to prevent duplication of results.

Four cases are discussed and compared based on the different system responses and characteristics. Case 1 describes the step response of the closed loop DC motor system that is developed in Section 4.4. Case 2 describes the step response of the closed loop DC motor system with state feedback control that is developed in Section 5.2. Case 3 and 4 describes the step response of the DC motor system with state feedback and integral control. The difference between case 3 and 4 is the change in the input set point to compare the controller response to multiple inputs.

Figure 5.8 shows the step input responses for cases 1, 2, 3 and 4. In each case, the set point is compared to the angular position feedback of the DC motor. As shown, Case 1 has the slowest rise time as there is no addition of a controller to the system. Adding a full state feedback controller causes Case 2 to have a faster rise time than Case 1, but the steady-state error is huge. The addition of integral control to the system reduces the steady state error to 0 which can be seen in Case 3 and 4.

The output graphs of the 4 cases are developed in MATLAB with the code found in Appendix A5.3. The 'To-Workspace' function block is used to extract the positions and set points from Simulink to MATLAB to allow for custom plotting of the results.

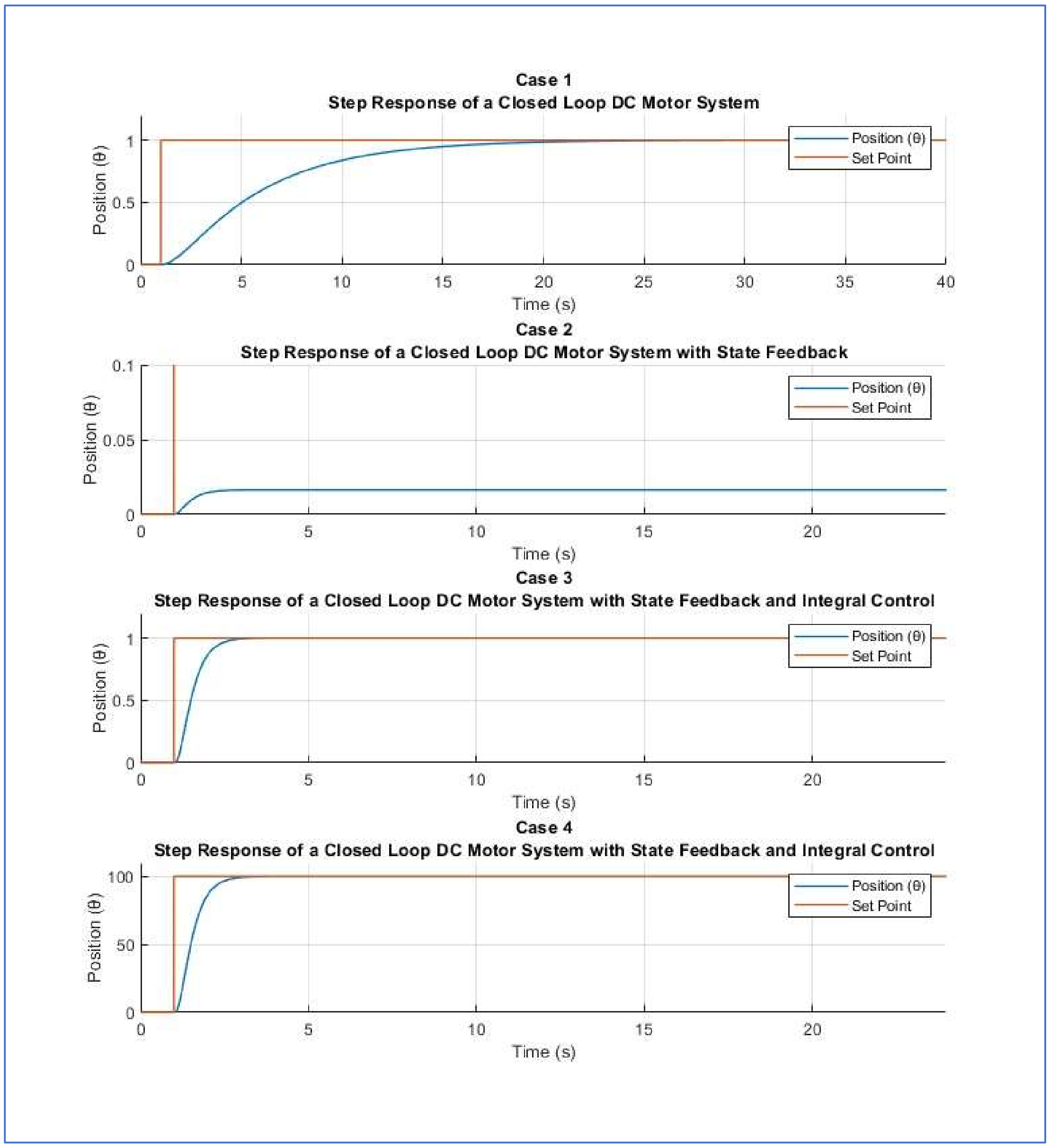


Figure 0.8: Developed controllers' response to step inputs

The response characteristics of each case is shown and compared in Table 5.1. The results show the difference in step input, rise time, settling time, steady-state error, and overshoot.

Table 0.1: **Characteristics of the step response of the different DC motor control systems**

	Description	**Step Input**	**Rise time**	**Settling time**	**Steady-state error**	**Overshoot**
Case 1	Closed loop DC motor system without a controller	1	4.013s	9.164s	0	0%
Case 2	Closed loop DC motor system full state feedback control	1	0.966s	2.714s	9.84	0%
Case 3	Closed loop DC motor system full state feedback and integral control with a small input set point	1	1.244s	3.303s	0	0%
Case 4	Closed loop DC motor system full state feedback and integral control with a large input set point	100	1.244s	3.303s	0	0%

The results prove that adding a full state feedback controller to a closed loop system does decrease the rise time as shown by the difference between Case 1, which has a rise time of 4.013s, and Case 2, which has a rise time of 0.966s. The decrease in rise time results in a decrease in settling time and improves the overall response of the system. To reduce the 9.84 steady-state error shown in case 1, integral control is used for Case 3 and 4. This results in similar rise and settling time, but zero steady state error. A step input of 100 is input into Case 4 to prove that the system reacts the same to larger values inputs. This is shown by Case 3 and 4 having the same rise time and settling time responses.

The final test done to show the full potential of the designed controller is shown in Figure 5.9. A random number generator is used as an input to the model to see how the system responds when the set point changes over time. The results shown has proven that the system is stable and can respond to changes in position set point with the same settling time as Case 3 and 4.

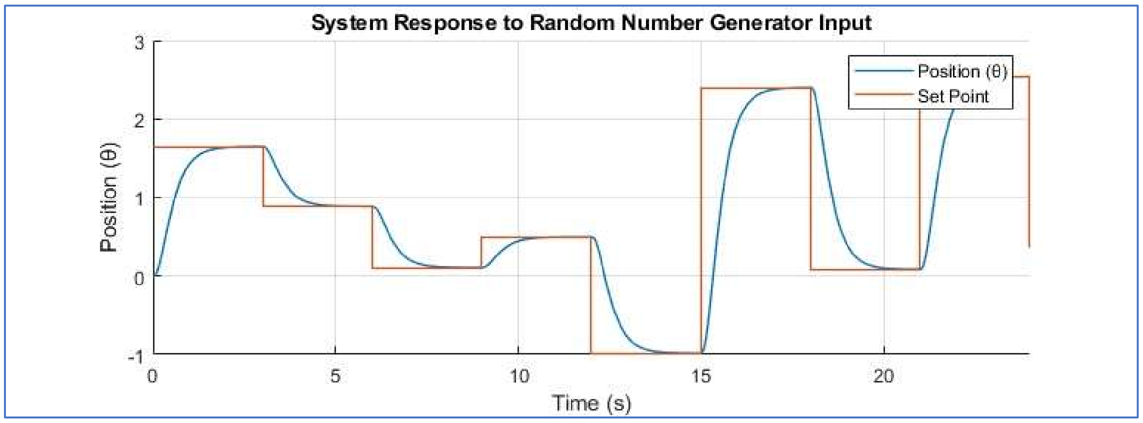


Figure 0.9: **Final Control System Response to Random Number Generator**

To use state feedback control, it is necessary to measure every state in the system. This can become very costly as additional measuring hardware is required. To overcome this issue, it is possible to add an observer to the control system to estimate the values of certain states. The next section simulates the effect of network induced delays on the DC motor control system.

5.4 Communication Delays

This subsection simulates the effect of network delays on the DC motor control system developed in Chapter 5.3. Network delays need to be considered when designing a system which has some fieldbus separating different components. This hardware implementation uses the EtherCAT fieldbus protocol to communicate between the controller and the plant. The network delays associated with this configuration can be implemented in the Simulink model to analyse the effects on the overall system.

The two important delays which occur in networked control systems is the sensor-to-controller time (T_{sc}) delay and the controller-to-actuator time (T_{ca}) delay. These delays are added to the Simulink block diagram by using a transport delay block as shown in Figure 5.12. The T_{sc} delay is added in the feedback line to simulate a delay in the encoder position feedback from the motor to the controller. A value of 200ms is used as an example. The T_{ca} delay is added between the controller and the input to the B matrix of the state space form of the motor. A value of 100ms is used as an example. This simulates a delay in command transmission from the controller to the plant.

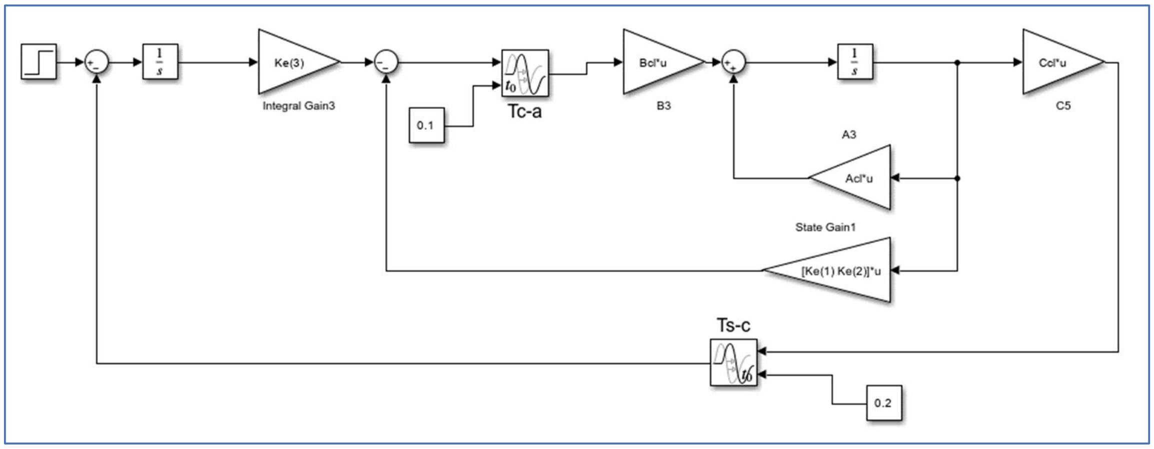


Figure 0.10: Simulink block diagram with added transport delay blocks

Simulating the delays in Simulink help understand the limits of the system when designing network-controlled systems. These delays can be analysed beforehand which can lead to a change in controller design, or a change in the hardware used. Critical delays which can cause the system to become unstable can also be found and dealt with before implementation. The following subsections describe the effects of the T_{sc} and T_{ca} delays on the system through case studies. Various delays are added to the system and compared to the model which does not have delays.

5.5.1 Sensor to controller delays

T_{sc} delays can occur in closed-loop control systems depending on the hardware and fieldbus protocol used. In this thesis, these delays occur due to the time taken for the encoder feedback to register in the motor terminal controller. Depending on the magnitude of the delay, the system could become unstable. In other cases, the delay might not be critical, causing the system to decrease in performance, but remain stable. Table 5.2 compares the system with no delays to four cases of systems with different sensor to controller delays.

The results show that an increase in T_{sc} leads to a reduction in rise time. This happens due to the controller trying to compensate for the delay in feedback by commanding the motor to run faster. The motor climbs to the set point quicker, but this can cause an overshoot as seen in Case 2 and 3. The settling time also increases as T_{sc} increases due to the long duration that the system takes to reach the set point. The results in Case 4 show an unstable system as the set point is never reached and therefore the settling time, steady state error and overshoot is infinite.

Table 0.2: **Comparison between different sensor to controller delays in DC motor system**

	T_{sc}	Rise Time	Settling Time	Steady-State	Over-shoot
No delays	0ms	1.24s	3.30s	0	0%
Case 1	100ms	0.99s	2.74s	0	0%
Case 2	400ms	0.70s	5.56s	0	25.48%
Case 3	800ms	0.53s	9.91s	0	128.55%
Case 4	1200ms	6.96s	Infinite	Infinite	Infinite

The tabular results are graphed using Simulink and shown in Figure 5.13 using a step input of 10mm. Each case is compared to the system with zero delay between the sensor and controller (T_{sc}). The overshoot described for Cases 2 and 3 can clearly be seen. The first three cases are shown to eventually reach steady state and achieve the desired set point. The position in Case 4 will continue to increase and decrease at an unstable rate, never reaching steady state.

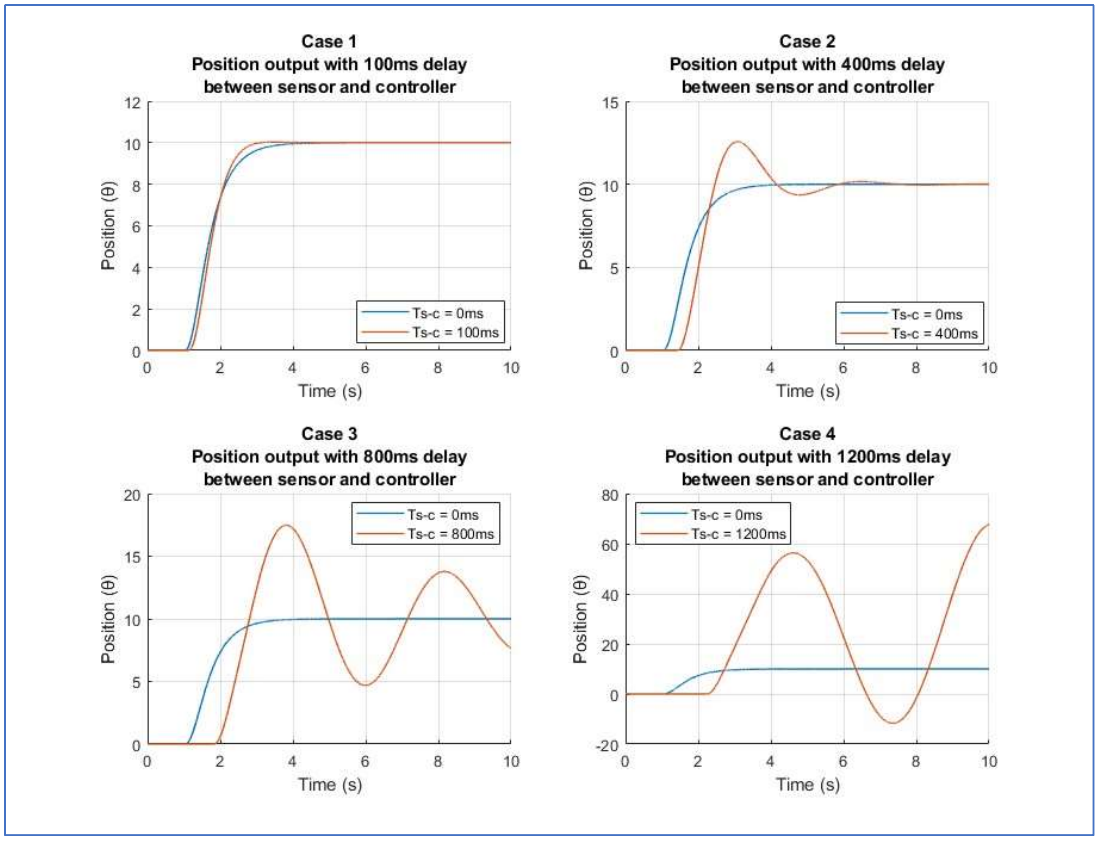


Figure 0.11: **Case Study on the effects of sensor-to-controller delays**

The effects of T_{sc} delays in the DC motor control system have been analysed in this subsection. The following subsection shows the simulation results of the system with different controller to actuator, T_{ca} delays.

5.5.2 Controller to actuator delays

Delays from the controller to the actuator can occur in closed-loop control systems. In this system these delays occur between the controller and the DC motor. Like T_{sc} delays, an increase in T_{ca} delays cause stability and performance issues in a control system. Table 5.3 shows the results when an increase in T_{ca} delays is added to the DC Motor Simulink model using a transport delay block. Noticeably, small T_{ca} delay examples are used for simulation as even a small increase in this type of delay has major negative effects on the system.

Table 0.3: **Comparison between different controller to actuator delays in DC motor system**

	T_{ca}	Rise Time	Settling Time	Steady-State	Over-shoot
No delays	0ms	1.244s	3.303s	0	0%
Case 1	50ms	1.225s	3.314s	0	0%
Case 2	100ms	1.175s	3.331s	0	0%
Case 3	125ms	1.227s	7.467s	0	2.464%
Case 4	150ms	1.867s	Infinite	Infinite	Infinite

The larger the T_{ca} delay, the longer the system takes to reach the set point as shown by the increase in rise time values between Case 1 to 4. A small overshoot is present when the delay is 125ms, but steady state is still reached over time. Case 4 shows that a T_{ca} delay of 150ms or more causes the system to become unstable, preventing the motor from ever reaching steady state. The motor continuously turns forward and then reverse and then forward in an ever-increasing endless cycle.

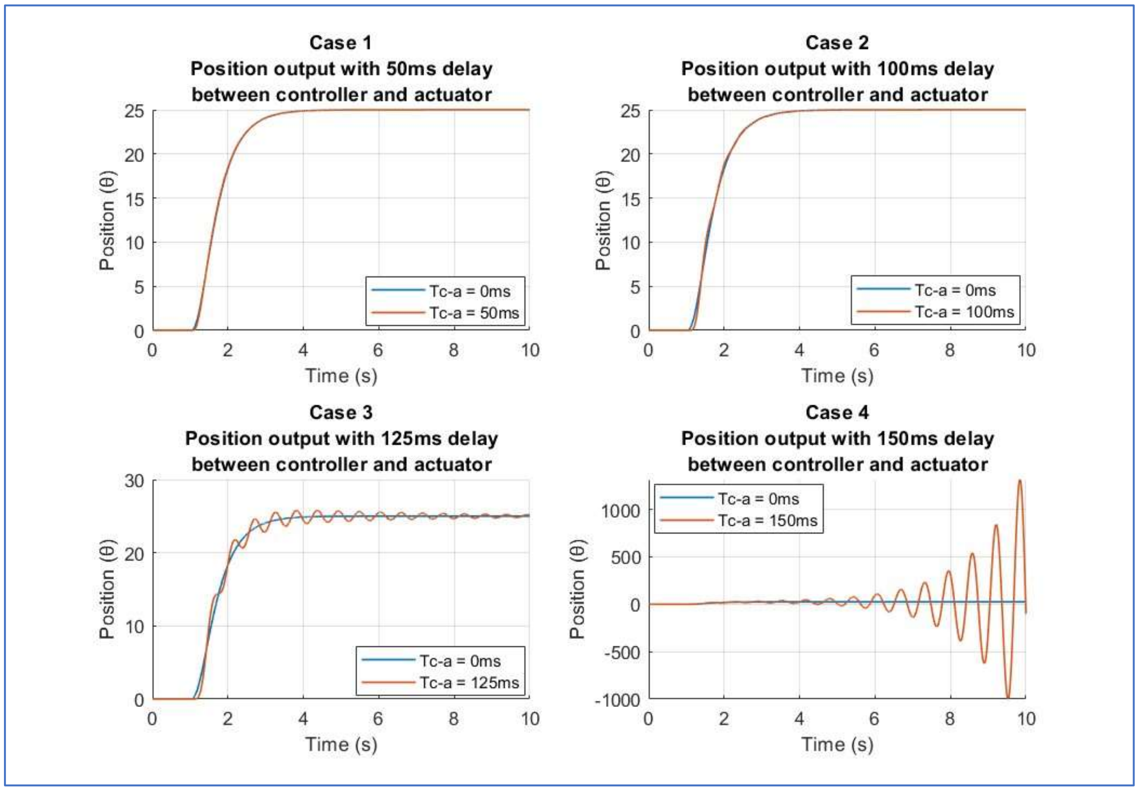


Figure 0.12: **Case Study on the effects of controller-to-actuator delays**

This section analysed the effects of T_{sc} and T_{ca} delays in the developed control system. It is shown that an increase in either delay leads to instability and performance decrease. In some cases, the system can recover, but at certain critical delays the system becomes completely unstable. It is also not good practice to have a system overshoot the set point and recover, depending on the desired characteristic response. The next section concludes the chapter on controller design.

5.5 Discussions and analysis

The simulation results of the designed state feedback controller for the DC motor closed-loop system is proven to be effective. The aim of adding the controller was to decrease the rise time of the step response of the system. This would cause the DC motor to react quicker to changes in angular position when commanded. The controller reduced the rise time response to a quarter of the rise time that the system with no controller performed.

The disadvantage is that a huge steady state error is present when using state feedback. To overcome this, integral control is added to the system. Integral control reduced the steady state error from 9.84mm to zero. This is done by multiplying the integrated input by a gain big enough to have zero steady state. There is also no overshoot present and the rise time is slightly affected but still much faster than the closed-loop system without a controller. The system is tesed using a random set point generator over time to test the robustness. The controller performed as desired, showing stability and validated the controller design.

Since a fieldbus protocol will be used in the implementation to create the communication link between the controller and plant, the imposed delays were analysed in a case study. This is done to have results that show the effects of sensor-to-controller and controller-to-actuator delays on the response of the system. The results help identify if different hardware or control techniques should be considered before implementation studies can commence.

The case study proved that the network induced delays do have an effect on the performance and stability of the system. It is shown through Simulation simulations that an increase in either sensoer-to-controller or controller-to-actuator delays leads to a decrease in system performance. The delays in position feedback causes the controller to overshoot the set point. In some cases the plant eventually reaches steady state, but for longer delays then system becomes unstable.

5.6 Conclusion

This chapter focused on the design of a full state feedback controller for the closed-loop DC motor control system. Pole placement technique is used to shift the poles of the system by using a gain K in the feedback loop. Shifting the poles is important as it allows for the change in system response to a desired response. In this section the desired response was to have a faster rise time than the DC motor closed-loop system with no controller while still having good performance and stability.

A step-by-step process is described showing the control law, testing controllability, and showing how to solve for the gain matrix K. The addition of state feedback causes a huge steady state error in the output of the control system, therefore integral control is also added. The integral control law is used to solve for the gains K and Ki that speed up the response of the system while also achieving a faster rise time, no overshoot, and zero steady state error.

A case study is completed comparing the step response of the closed loop system without a controller, the system with state feedback control, and the system with state feedback and integral control. The full state feedback with added integral controller showed the best response by achieving the desired position output. A random set point generator is used to test the agility and robustness of the controller to rapid changes in position set point. The effect of network delays is also analysed and discussed.

The control system has been simulated and analysed and is therefore ready for implementation. The next section shows how to transform the developed Simulink block diagram to the Windows Control and Automation Technology (TwinCAT) programming environment to allow for real-time implementation of the control system.

CHAPTER 6

TRANSFORMATION FROM SIMULINK MODEL TO TWINCAT OBJECT

6.1 Introduction

This chapter describes the transformation of Simulink models into objects that are used in the Beckhoff TwinCAT 3 software programming environment. The application of the transformation is necessary to enable the utilization of the controller formulated in Chapter 5 for the regulation of the actual DC motor that is connected to the Beckhoff EtherCAT remote interface. The Simulink block diagram is converted to C++ code which can be read and used by TwinCAT. The transformation assesses the portability of the code from Simulink to TwinCAT, encompassing the transition from a continuous-time application to an active real-time control environment. A successful transformation is in-line with the IEC 61499 standard which aims to increase the portability, interoperability, and reusability of software components between vendors.

Initially, a comprehensive description detailing the characteristics of every necessary software package is provided, alongside a step-by-step manual explaining the installation process for each development environment. The software required includes MATLAB/Simulink, TwinCAT 3, Visual Studio, and the TE1400 Target. Once all software packages are installed, a step-by-step guide is shown on how to use and configure the installed tools to achieve a successful transformation from the Simulink to TwinCAT 3 software environments. An overview of the common installation errors is examined, and corresponding solutions are supplied to rectify these issues. To validate the success of the transformation, a real-time numerical generator is formulated within TwinCAT. This generator uses identical inputs as the Simulink random number generator output shown in Figure 5.9 within Section 5.3.4, facilitating a comparative assessment of the system's behaviors. The object’s response to the random set points is plotted using Scope Viewer, which is a plotting tool included in the TwinCAT 3 package.

Section 6.2 describes all the necessary software packages needed to set up the engineering computer for development. Section 6.3 describes how to export the developed controller model in the Simulink software environment. Section 6.4 describes how to import the Simulink model into the TwinCAT programming environment. In Section 6.5 the Simulink model and TwinCAT objects are simulated and compared to confirm that the transformation is successful. Lastly, the discussion and conclusion to the chapter is presented in Section 6.6.

6.2 Software Descriptions

This section describes the installation of the prerequisite software that is needed for the transformation of Simulink models to the TwinCAT 3 programming environment. Table 6.1 shows the versions, with download links, of the software that is used to complete the implementation section of this research. All the listed software packages are installed on the same Personal Computer (PC) which is referred to as the engineering PC in this thesis. Windows 11 Home version 22H2 is the installed operating system on the engineering PC.

Table 0.1: **Software installation download links**

No.	Software	Download Link
1	Visual Studio 2019 Community	https://visualstudio.microsoft.com/vs/community
2	TwinCAT 3 XAE v3.1.4024.47	https://www.beckhoff.com/en-en/support/download-finder/software-and-tools/
3	MATLAB/Simulink 2023a	https://www.mathworks.com/products/matlab.html
4	TE1400 Target	https://www.beckhoff.com/en-us/products/automation/twincat/texxxx-twincat-3-engineering/te1400.html

The following subsections give a summary of each software environment that is required to be installed on the engineering PC. The summaries include information regarding the software packages, as well as a guide on certain parts of the individual installations.

6.2.1 MATLAB/Simulink software description

MATLAB/Simulink is a software environment that combines programming, simulation, and visualization tools to facilitate the development and analysis of complex engineering and scientific systems. The MATLAB/Simulink programming environment has already been used in Chapters 4 and 5 to develop the block diagram of the full state feedback controller with integral control that needs to be exported to TwinCAT 3. The installation procedure for MATLAB/Simulink is shown in Appendix B6.1. It is recommended to select all products when installing to allow for the full functionality of the MATLAB and Simulink programming environments.

The MATLAB and Simulink Coders, which are automatically installed, are used to convert the script files and block diagrams to C++ code for TwinCAT 3 to understand. This conversion process is described in Section 6.3. The rest of the installation of MATLAB/Simulink software is straightforward and is completed by clicking next until the products start downloading and installing. The next subsection describes how to install Visual Studio 2019 with all its necessary components.

6.2.2 Visual Studio 2019 software description

Older versions of the TwinCAT software required the installation of Visual Studio to serve as the programming shell for the software. TwinCAT 3 includes shell software called TwinCAT eXtended Automation Engineering (XAE) where the user can program and interact with the software tools. Unfortunately, this new shell program does not include the development tools needed to run C++ objects. Therefore, it is needed to install Visual Studio 2019 Community to add the desktop development kit. This is done during the installation of Visual Studio 2019. Appendix B6.2 describes the installation procedure for Visual Studio 2019. This software is important as the Simulink block diagram is converted to C++ code when transformed to TwinCAT and therefore the necessary tools to read the C++ language are required.

It is important to note that Visual Studio must be installed before the TwinCAT software to integrate the two software programs together. If TwinCAT 3 is already installed on the engineering PC, it is necessary to uninstall it, then install Visual Studio, and then reinstall TwinCAT 3 software. The rest of the Visual Studio 2019 software installation process is straightforward and is followed by the installation of TwinCAT 3 software which is described in the next subsection.

6.2.3 TwinCAT 3 software description

The Windows Control and Automation Technology (TwinCAT) 3.1 software is Beckhoff's integrated programming platform that enables the development, configuration, and control of automation systems, offering real-time control, data processing and visualization capabilities. This software package is used as the gateway between the Simulink model and the hardware components. The installation of the TwinCAT software is straightforward and assistance is provided by the installation guide on the Beckhoff website if necessary.

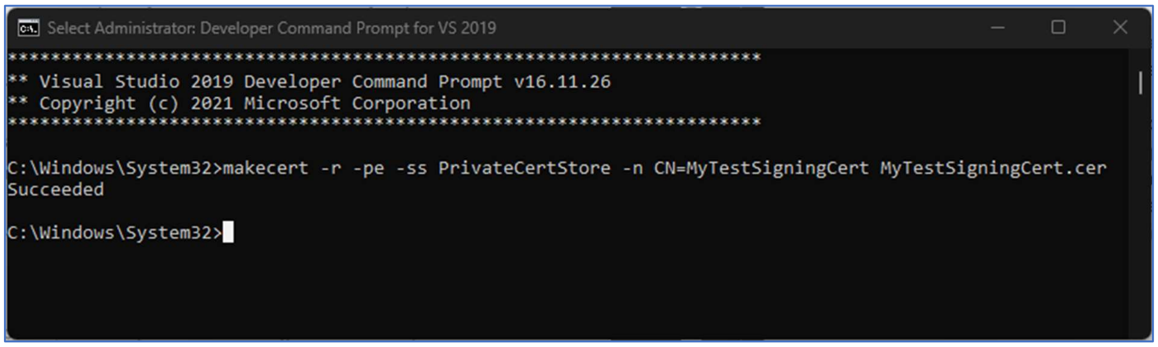
The full version of TwinCAT is installed on the engineering PC, which includes the engineering environment as well as the run time software. The installation procedure for TwinCAT 3 is described in Appendix B6.3. Once the installation is complete it is required to restart the engineering PC to allow for all the new system drivers to activate. The next subsection describes the creation of self-signed certificates to register the engineering PC drivers to enable the use of the TwinCAT software tools.

6.2.4 Drive signing

TwinCAT systems use a process called drive signing to ensure that device drivers are authentic by providing a trusted and secure environment for communication between the hardware and software components. This is done by creating certificates that are used during the TwinCAT build process to authenticate the necessary drivers. Usually, a driver needs to be co-signed by Beckhoff through a certificate creation process, but for testing purposes it is possible to create a test certificate that is self-signed.

Below are the steps for successful signing of drivers on Windows.

1) Create a test certificate that is used by TwinCAT to allow for projects to be authenticated when building. Open Developer Command Prompt 2019 as an administrator and execute the command 'makecert -r -pe -ss PrivateCertStore -n CN=MyTestSigningCert MyTestSigningCert.cer' as shown in figure 6.1. After executing, the message "Succeeded" shows if the certificate creation was successful. The command prompt is found in the Windows search bar and should be run as an administrator otherwise the certificate will not be created. Developer Command Prompt 2019 is installed automatically when installing Visual Studio 2019.



```
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.11.26
** Copyright (c) 2021 Microsoft Corporation
**********************************************************************

C:\Windows\System32>makecert -r -pe -ss PrivateCertStore -n CN=MyTestSigningCert MyTestSigningCert.cer
Succeeded

C:\Windows\System32>
```

Figure 0.1: **Developer Command Prompt 2019 Interface**

2) Check if the certificate was created successfully. Search 'mmc' with the Windows run function to open Microsoft Management Console (MMC). When the software opens, navigate to 'File' and then 'Add/Remove Snap-in' as shown in Figure 6.2, labelled 1 and 2 respectively. Shortcut keys Ctrl+M can also be used for this step.

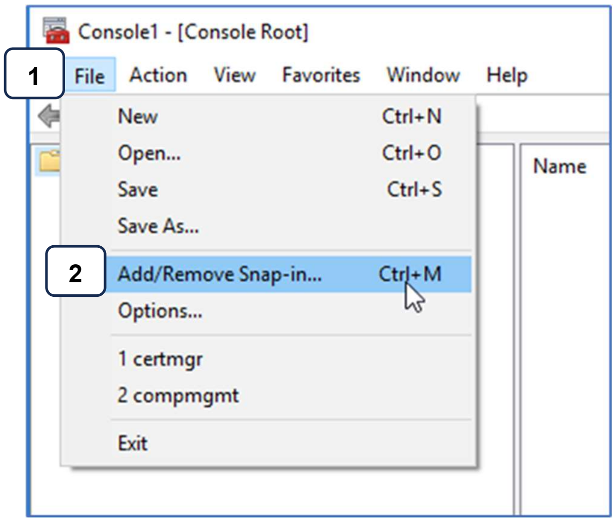
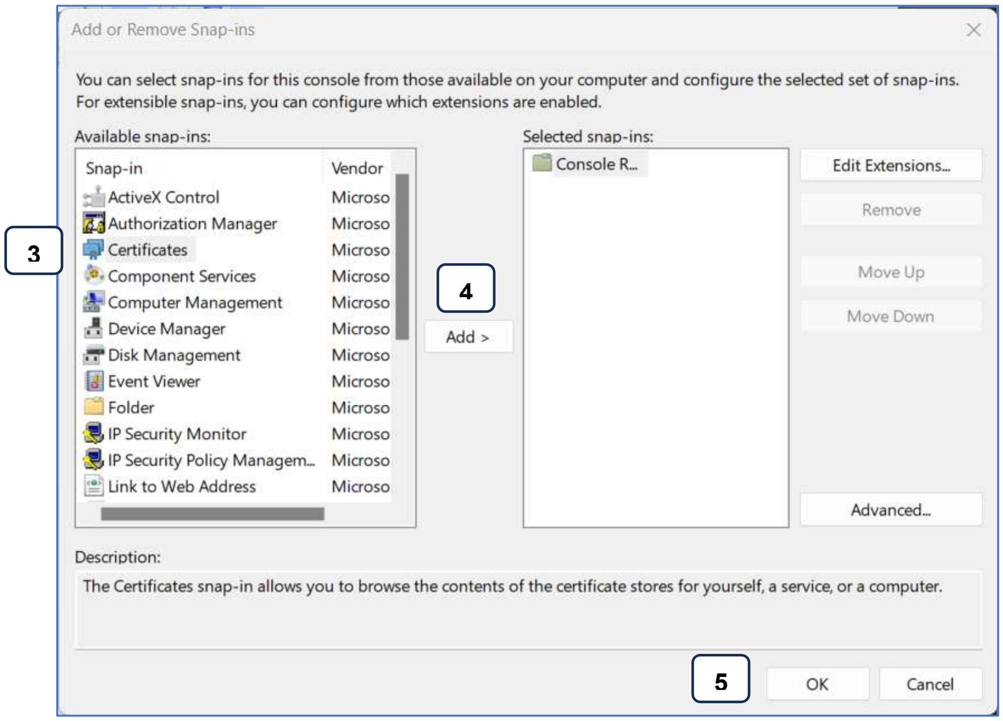


Figure 0.2: MMC - Add/Remove Snap-in

The menu shown in Figure 6.3 will open. Click certificates (3), and then add to add console root (4) to the selected snap-ins. Click OK (5) to confirm selection and the menu shown in Figure 6.4 will appear. Select the radio button for my user accounts (6) and click finish (7). Lastly, check that 'Certificates – Current User' option is available under Console Root (8-9), as shown in Figure 6.5. Click OK (10) and the configured settings open a new MMC window to view all created certificates as shown in Figure 6.6. 'MyTestSigningCert' should be created under submenu 'PrivateCertStore – Certificates' (11-12).


Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microso
Authorization Manager	Microso
Certificates	Microso
Component Services	Microso
Computer Management	Microso
Device Manager	Microso
Disk Management	Microso
Event Viewer	Microso
Folder	Microso
IP Security Monitor	Microso
IP Security Policy Managem...	Microso
Link to Web Address	Microso

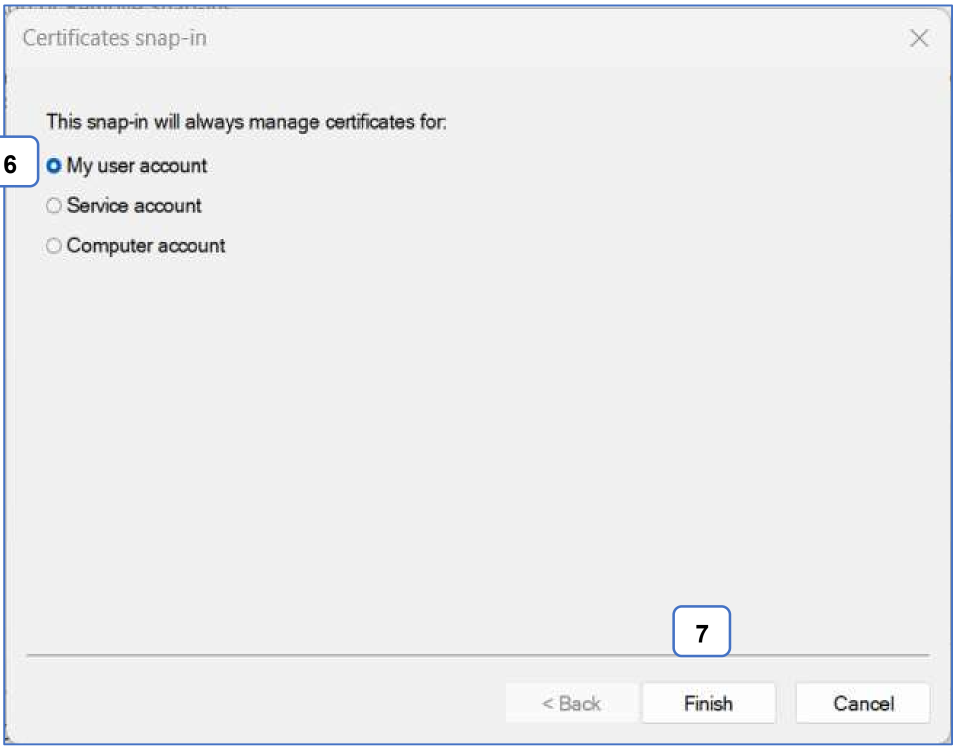
Selected snap-ins:

Console R...

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

Figure 0.3: MMC - Add/Remove Snap-in Step 2


Certificates snap-in

This snap-in will always manage certificates for:

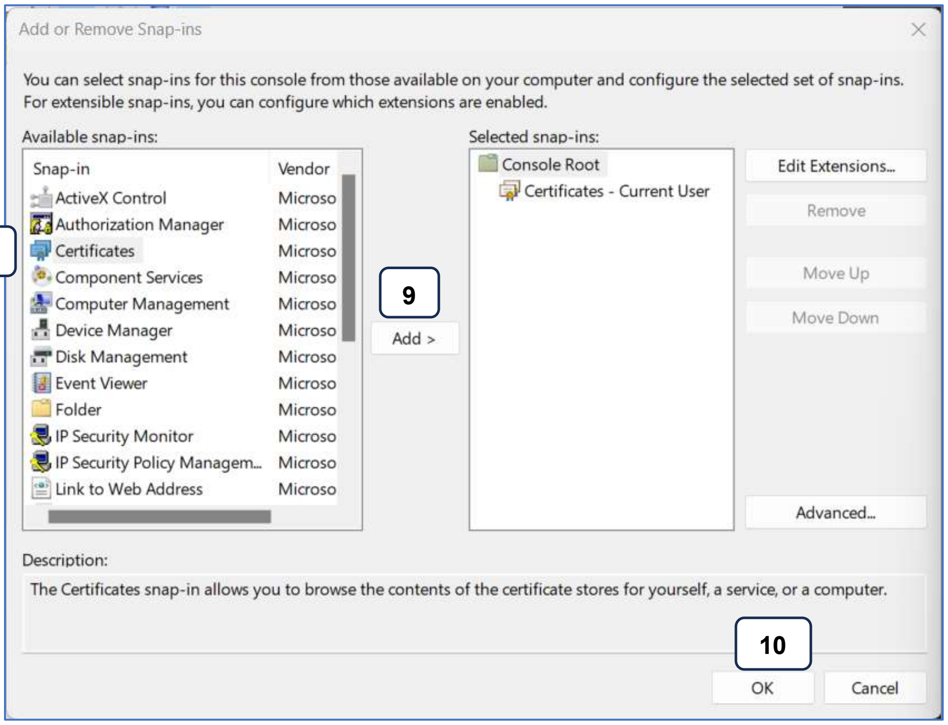
6

[✓ My user account]

Service account

Computer account

Figure 0.4: MMC - Add/Remove Snap-in Step 3


Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microso
Authorization Manager	Microso
Certificates	Microso
Component Services	Microso
Computer Management	Microso
Device Manager	Microso
Disk Management	Microso
Event Viewer	Microso
Folder	Microso
IP Security Monitor	Microso
IP Security Policy Managem...	Microso
Link to Web Address	Microso

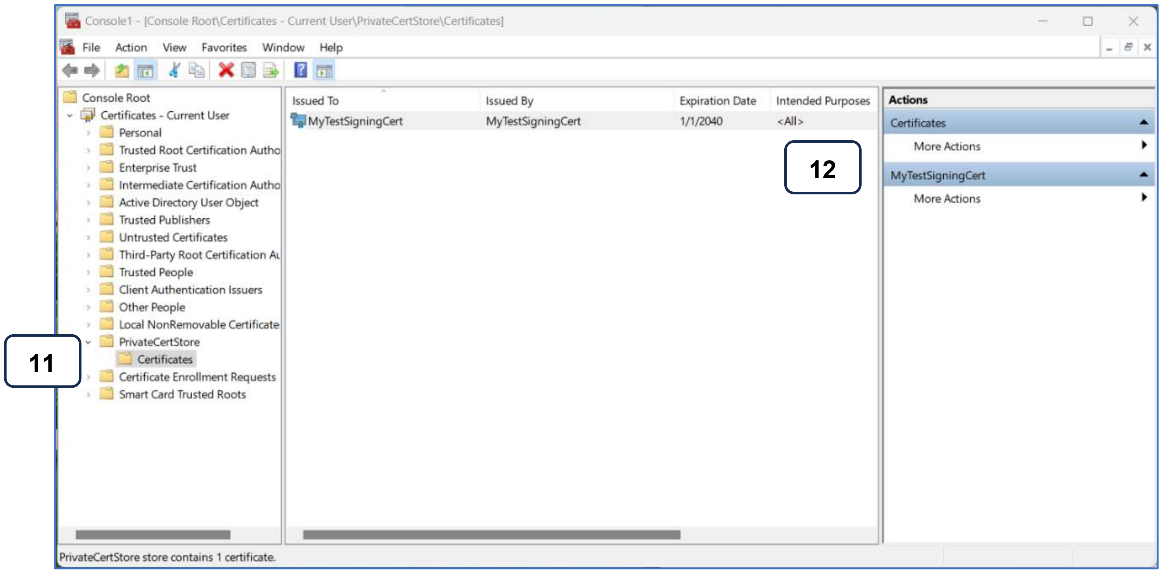
Selected snap-ins:

Console Root
Certificates - Current User

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

Figure 0.5: MMC - Add/Remove Snap-in Step 4


Console1 - [Console Root\Certificates - Current User\PrivateCertStore\Certificates]
File Action View Favorites Window Help
Console Root
Certificates - Current User
Personal
Trusted Root Certification Autho
Enterprise Trust
Intermediate Certification Autho
Active Directory User Object
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Au
Trusted People
Client Authentication Issuers
Other People
Local NonRemovable Certificate
PrivateCertStore
Certificates
Certificate Enrollment Requests
Smart Card Trusted Roots
Actions
Certificates
More Actions
MyTestSigningCert
More Actions
11
12

Issued To	Issued By	Expiration Date	Intended Purposes
MyTestSigningCert	MyTestSigningCert	1/1/2040	<All>

PrivateCertStore store contains 1 certificate.

Figure 0.6: MMC - Certificates

3) Start Windows operating system in Test Mode. Self-signed certificates are only active when running Windows in Test Mode. To change to Test Mode, enter the following command in Windows command prompt: ‘bcdedit /set testsigning yes’. It is necessary to run Windows command prompt as an administrator to allow for the successful transition to Test Mode. Windows command prompt is accessed by searching ‘cmd’ using the Windows search function. After restarting the PC, Windows should be in Test Mode. To confirm, there should be white text in the bottom right corner of the Windows desktop stating that that Windows is now in Test Mode. To revert to the normal operating mode, use the command ‘bcdedit /set testsigning no’ in Windows command prompt and then restart the PC.

This section configured the necessary self-signed certificates needed to authenticate the x64 drivers for TwinCAT. The next subsection describes the installation of the TE1400 target that must be installed for the transformation of Simulink block diagrams to TwinCAT objects.

6.2.5 TE1400 Target

This section describes the installation of the TE1400 target that is installed in Simulink. Beckhoff Automation has developed targets such as the TE1400 to allow for a seamless transition between two software development environments. In this research, the target allows for the efficient integration of the developed Simulink model into the TwinCAT automation system by converting the block diagrams to C++ code. The C++ code is automatically converted and imported into TwinCAT when running a series of MATLAB commands that uses the target functionality. Using the TE1400 target to generate TwinCAT objects from Simulink block diagrams requires a license. The license must be installed on the engineering PC where the module generation takes place. It is possible to use the target without a license but with limited features. These limitations are on the design and complexity of the Simulink models that are transformed. Only models with less than 100 blocks, 5 input signals, and 5 output signals can be transformed without a license.

The TE1400 target is added to the TwinCAT functions after installing the installation file that can be downloaded from the link of item number 4 in Table 6.1. The target function can be accessed in the MATLAB software by navigating to the C: > TwinCAT > Functions > TE14xx-ToolsForMatlabAndSimulink path (1) and running the setup file (2) as shown in Figure 6.7. The installation takes a few seconds and if successful, the message shown in Figure 6.8 is displayed in the MATLAB command window (3). Instructions on how to use the target as well as a few sample projects are also displayed in the same MATLAB command window (4).

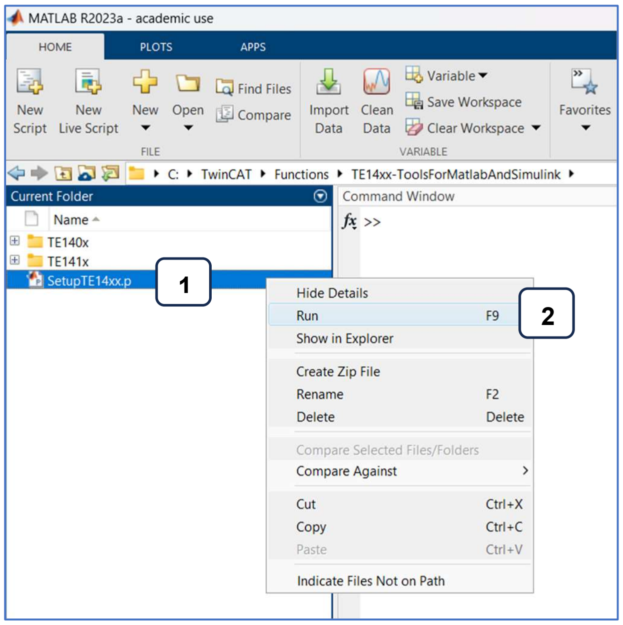
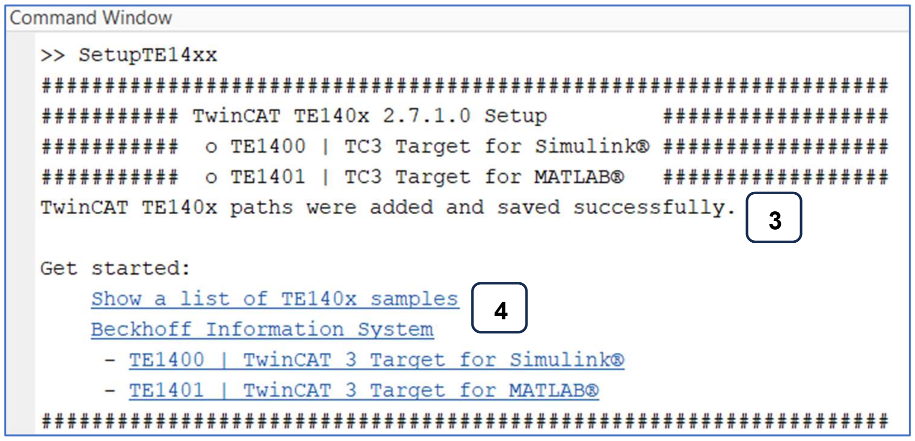


Figure 0.7: **Navigation to TE1400 TwinCAT function**

Section 6.2 described the installation of all the prerequisite software required for the transformation of Simulink block diagrams to TwinCAT objects. The installation of MATLAB/Simulink, Visual Studio 2019 Community, TwinCAT 3, and the TE1400 target are all discussed. The use of certificates to self-sign Windows drivers for authentication with TwinCAT to allow for successful compiling of TwinCAT projects is also described.


Command Window

```
>> SetupTE14xx
####################################################################
########### TwinCAT TE140x 2.7.1.0 Setup          ##################
###########  o TE1400 | TC3 Target for Simulink® ##################
###########  o TE1401 | TC3 Target for MATLAB®   ##################
TwinCAT TE140x paths were added and saved successfully.

Get started:
    Show a list of TE140x samples
    Beckhoff Information System
     - TE1400 | TwinCAT 3 Target for Simulink®
     - TE1401 | TwinCAT 3 Target for MATLAB®
####################################################################
```

Figure 0.8: **Status of TwinCAT TE140x installation from the MATLAB Command Window**

Now that the engineering PC is set up and all necessary software is installed, the software tools are used to start the transformation process. The next section describes the steps taken in Simulink for the transformation to take place, followed by Section 6.4 which describes the steps taken in TwinCAT to simulate and test that the transformation is successful.

6.3 Simulink Configuration

This section shows how to generate TwinCAT objects from the developed Simulink block diagrams. A revised iteration of the comprehensive Simulink diagram established in Chapter 5 is shown in Figure 6.9. The variables for state feedback gain, integral gain, and the A, B and C matrices, which were previously calculated in a MATLAB script, are all hard coded into the Simulink block diagram. This is done because the transformation process from Simulink to TwinCAT only uses the Simulink blocks and cannot include the MATLAB script files. Lastly, the step and scope Simulink blocks are removed and replaced with an input labelled “set point” and an output labelled “position” which is used in TwinCAT to interface with the model.

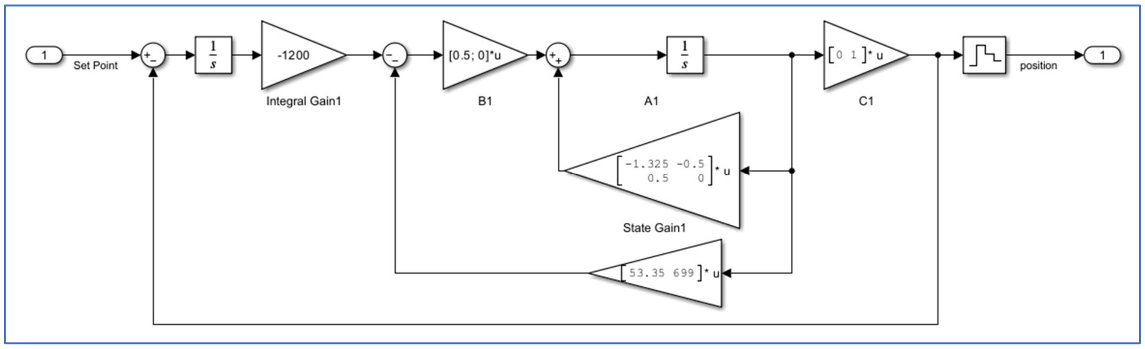
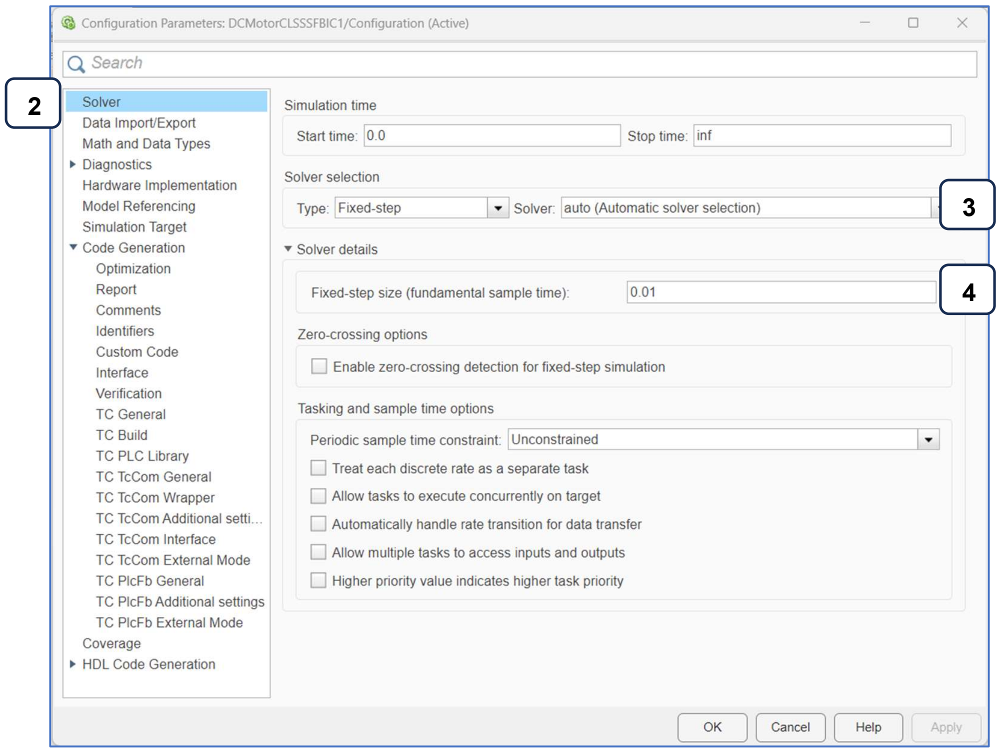


Figure 0.9: **Updated controller block diagram with added input and output blocks**

Appendix A6.1 contains code developed that assists with the generation of the TwinCAT objects. When run, the code performs a few tasks which changes the Simulink models' parameters and then saves and builds the project using the Simulink coder. The model parameters can also be set up manually beforehand. These parameters include changing the solver type, the system target file, and the TwinCAT 3 object name. All the parameters that need to be changed are found in the Simulink model which can be opened by using the keyboard shortcut Cntrl-E. Figures 6.10, 6.11, and 6.12 are all screenshots from the model settings page.

Figure 6.10 shows the settings page for the type of solver used (2). A fixed-step type is required (3), with a fixed-step size of 0.01ms (4). The solver itself is set to automatic or to whichever solver is preferred. Figure 6.11 changes the target system file to "TwinCatGrt.tlc" (4) which is the TE1400 target installed in section 6.2.5. Lastly, Figure 6.12 shows the naming conventions of the object that is created once transformed to TwinCAT 3 (7). As mentioned before, if the MATLAB script in Appendix A6.1 is run, all these settings are automatically updated, and the object is immediately available for use in TwinCAT 3.


Configuration Parameters: DCMotorCLSSSFBIC1/Configuration (Active)

Simulation time
Start time: 0.0   Stop time: inf

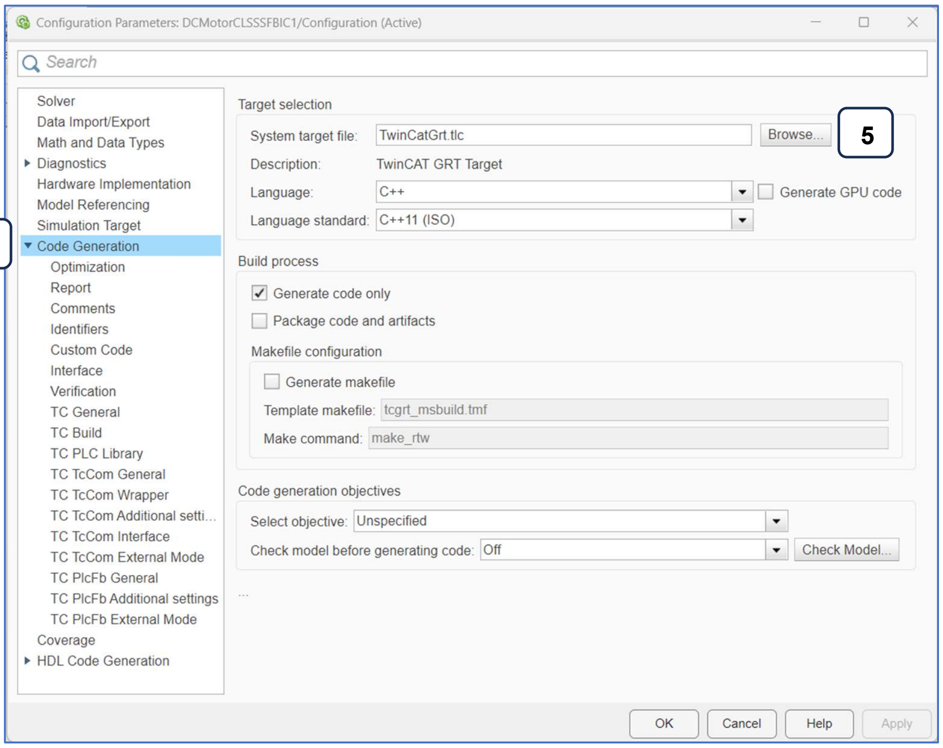
Solver selection
Type: Fixed-step   Solver: auto (Automatic solver selection)

Solver details
Fixed-step size (fundamental sample time): 0.01

Zero-crossing options
☐ Enable zero-crossing detection for fixed-step simulation

Tasking and sample time options
Periodic sample time constraint: Unconstrained
☐ Treat each discrete rate as a separate task
☐ Allow tasks to execute concurrently on target
☐ Automatically handle rate transition for data transfer
☐ Allow multiple tasks to access inputs and outputs
☐ Higher priority value indicates higher task priority

Figure 0.10: **Model configuration - Solver menu**



Target selection

System target file: TwinCatGrt.tlc

Description: TwinCAT GRT Target

Language: C++ ☐ Generate GPU code

Language standard: C++11 (ISO)

Build process

☑ Generate code only

☐ Package code and artifacts

Makefile configuration

☐ Generate makefile

Template makefile: tcgrt_msbuild.tmf

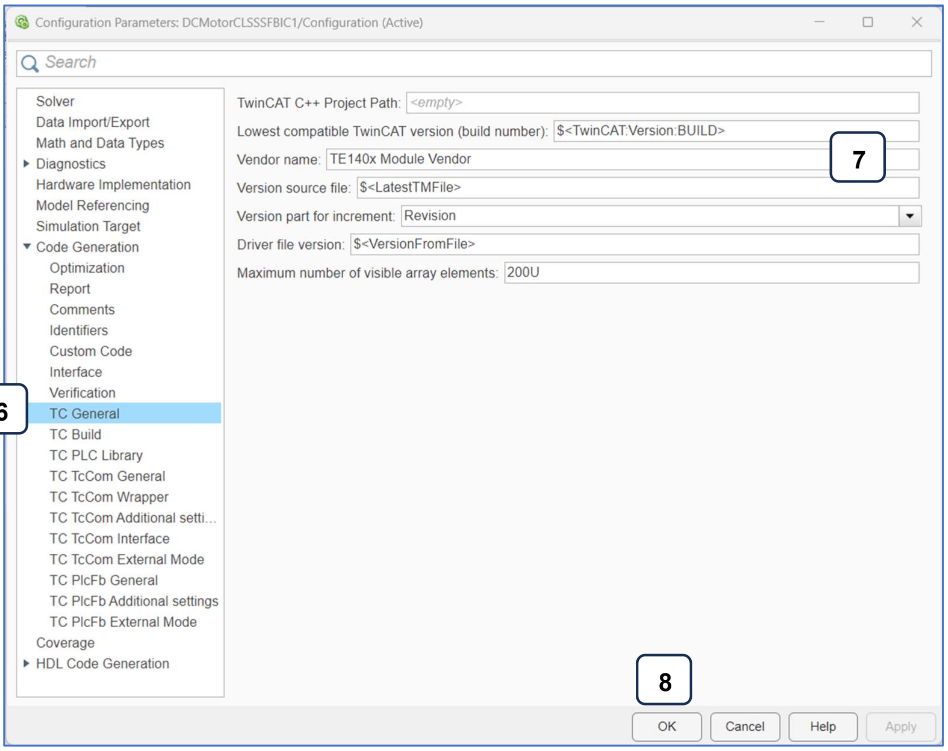
Make command: make_rtw

Code generation objectives

Select objective: Unspecified

Check model before generating code: Off

Figure 0.11: Model configuration - Code Generation submenu



TwinCAT C++ Project Path: <empty>

Lowest compatible TwinCAT version (build number): $<TwinCAT.Version.BUILD>

Vendor name: TE140x Module Vendor

Version source file: $<LatestTMFile>

Version part for increment: Revision

Driver file version: $<VersionFromFile>

Maximum number of visible array elements: 200U

Figure 0.12: Model configuration - TC General submenu

Table 6.2 shows a list of errors and solutions to issues found when trying to generate the TwinCAT objects from Simulink. The errors are found in the MATLAB diagnostics window when trying to build the code found in Appendix A6.3. Once all the errors are cleared using the supplied solution, the build is successful and the TwinCAT object is available for use in the TwinCAT XAE programming environment.

Table 0.2: **Possible errors during Simulink Coder compiling**

Error	Solution
Simulink Real-Time model build cannot use a file path with spaces for model build directory	The file location of the Simulink model on Windows had a space in one of the folder names. Delete all spaces and replace them with underscores to prevent this error.
Publish Procedure Failed	When running the certificate creation task in the command prompt, the command is run twice so two certificates are created. The two certificates are clashing as the same name is used. Make sure there is only one certificate in the mcc to prevent this error.
Error using tlc_new	This occurs when trying to compile the Simulink block diagram with the Simulink Coder without configuring the steps shown in Figures 6.13, 6.14, and 6.15.

6.4 TwinCAT Configuration

This section describes the process of adding the generated TwinCAT object from Section 6.3 into the TwinCAT XAE environment. The new object which consists of the Simulink block diagram is used to control the DC motors that are connected to the Beckhoff terminal card. Therefore, it is necessary to first develop and test the software before mapping the inputs and outputs of the object to the hardware.

Figure 6.13 shows the menu that is used to configure a new project with the TwinCAT 3 programming environment. The “TwinCAT XAE Project” is selected as a base starting project for development. Click on the required starting project (1) and then click the OK button (2) to start developing.

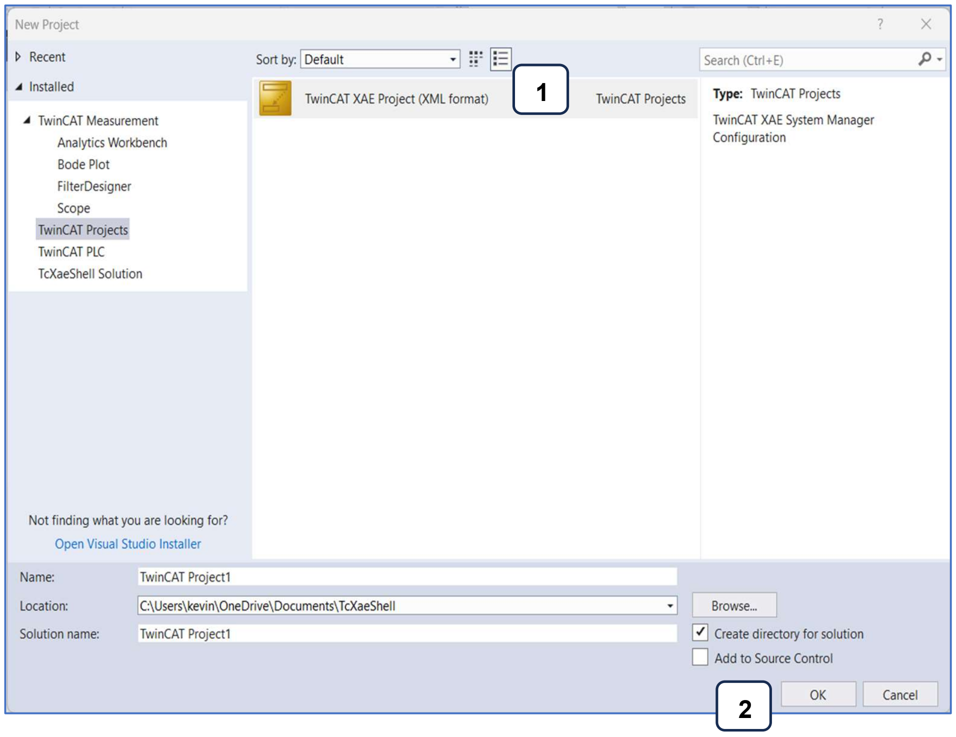


Figure 0.13: TwinCAT 3 new project window

Once a new project is created, the converted Simulink block diagram is added. Figure 6.14 shows how to open the list of TwinCAT Component Object Model (TcCom) Objects that can be added to the project for interfacing (3). This list is accessed from the Solutions Explorer tab in TwinCAT 3 programming environment. A new item can also be added by using the insert button on the keyboard (4).

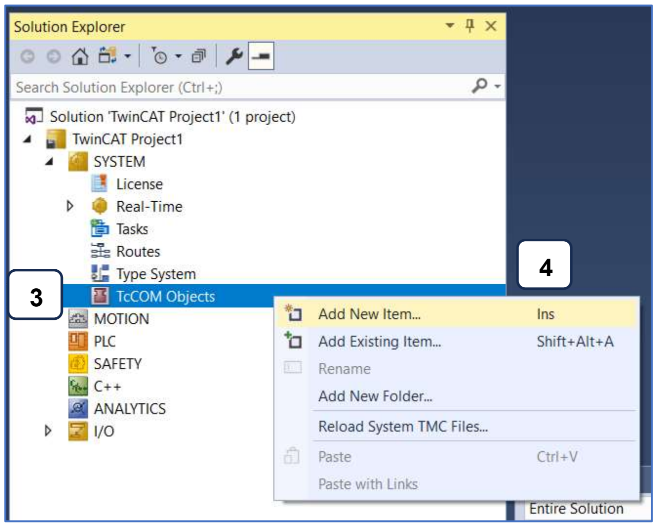
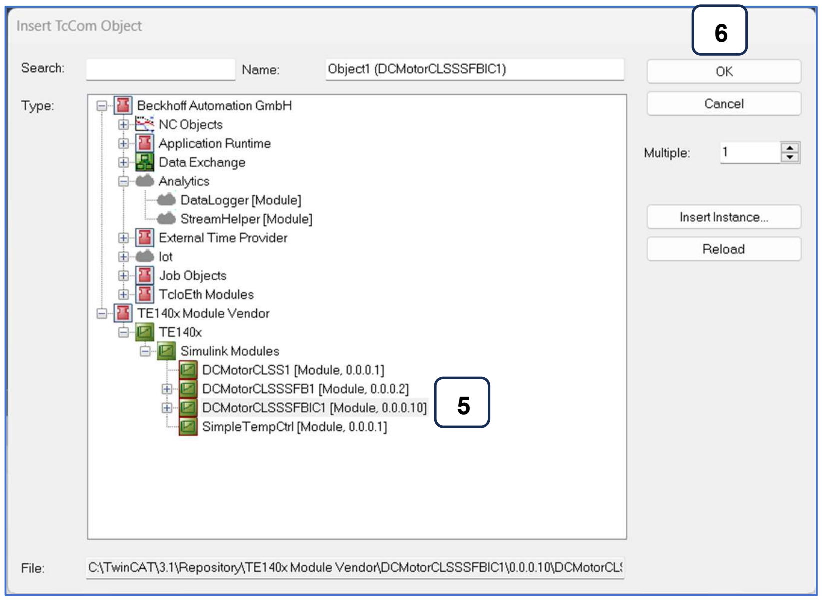


Figure 0.14: How to add a new TcCom object

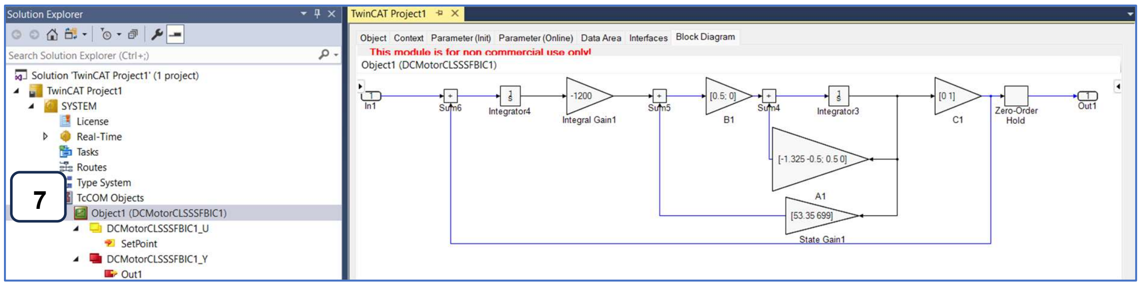
A new window, shown in Figure 6.15, appears when trying to add a new item. Navigate to TE140x Module Vendor (5) to find all the objects that are generated using the TE1400 target for Simulink. If a Simulink block diagram is generated twice, the version will update and be displayed next to the object name. Older versions are accessed via the small plus sign next to the object name. Once an object is selected, click OK (6) to add the item to the current TwinCAT project. The item is added to the solutions explorer under TcCom objects as shown in Figure 6.16 (7). The objects' inputs and outputs are now accessible by the rest of the TwinCAT project. The object's block diagram can also be viewed under the block diagram submenu.


Insert TcCom Object

Search:
Name: Object1 (DCMotorCLSSSFBIC1)
Type:
- Beckhoff Automation GmbH
  - NC Objects
  - Application Runtime
  - Data Exchange
  - Analytics
    - DataLogger [Module]
    - StreamHelper [Module]
  - External Time Provider
  - Iot
  - Job Objects
  - TcIoEth Modules
- TE140x Module Vendor
  - TE140x
    - Simulink Modules
      - DCMotorCLSS1 [Module, 0.0.0.1]
      - DCMotorCLSSSFB1 [Module, 0.0.0.2]
      - DCMotorCLSSSFBIC1 [Module, 0.0.0.10]
      - SimpleTempCtrl [Module, 0.0.0.1]

File: C:\TwinCAT\3.1\Repository\TE140x Module Vendor\DCMotorCLSSSFBIC1\0.0.0.10\DCMotorCL

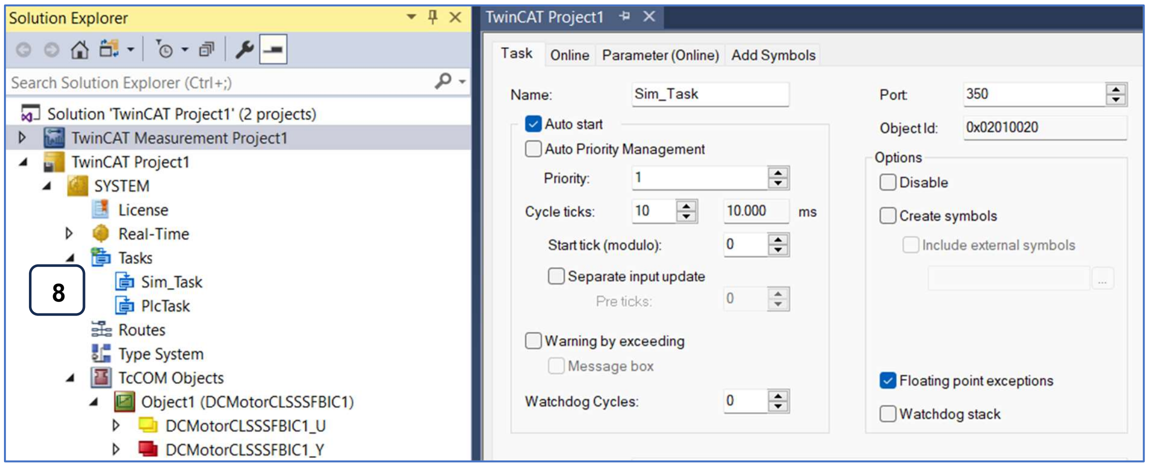
Figure 0.15: List of available TcCom objects


Solution Explorer
Search Solution Explorer (Ctrl+;)
- Solution 'TwinCAT Project1' (1 project)
  - TwinCAT Project1
    - SYSTEM
      - License
      - Real-Time
      - Tasks
      - Routes
      - Type System
      - TcCOM Objects
        - Object1 (DCMotorCLSSSFBIC1)
          - DCMotorCLSSSFBIC1_U
            - SetPoint
          - DCMotorCLSSSFBIC1_Y
            - Out1

TwinCAT Project1
Object | Context | Parameter (Init) | Parameter (Online) | Data Area | Interfaces | Block Diagram
This module is for non commercial use only!
Object1 (DCMotorCLSSSFBIC1)

Figure 0.16: Simulink block diagram successfully transformed to TwinCAT 3

Before activating the created TcCom object it is necessary to create and assign a task to set the cycle at which the object is processed. A task is added by right-clicking on 'Tasks' in the solution explorer. The only changes needed to be made to the default task are the name and the cycle ticks. Because the fixed step cycle time is set to 0.01 seconds in Simulink, a cycle ticks value of 10 is necessary. This creates a task cycle time of 0.01 seconds as shown in Figure 6.17 (8).



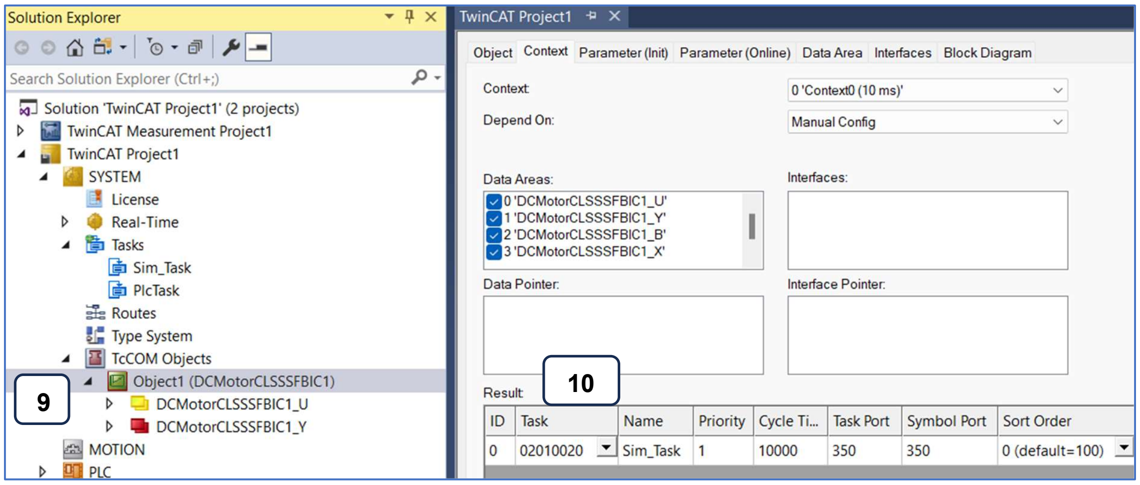
Solution 'TwinCAT Project1' (2 projects)
- TwinCAT Measurement Project1
- TwinCAT Project1
  - SYSTEM
    - License
    - Real-Time
    - Tasks
      - Sim_Task
      - PlcTask
    - Routes
    - Type System
    - TcCOM Objects
      - Object1 (DCMotorCLSSSFBIC1)
        - DCMotorCLSSSFBIC1_U
        - DCMotorCLSSSFBIC1_Y

Name: Sim_Task
[✓] Auto start
[ ] Auto Priority Management
Priority: 1
Cycle ticks: 10 10.000 ms
Start tick (modulo): 0
[ ] Separate input update
Pre ticks: 0
[ ] Warning by exceeding
[ ] Message box
Watchdog Cycles: 0

Port: 350
Object Id: 0x02010020
Options
[ ] Disable
[ ] Create symbols
[ ] Include external symbols
[✓] Floating point exceptions
[ ] Watchdog stack

Figure 0.17: Adding a new task

The next step is to assign the new task to the created TcCom object. This is done under the object properties, under the context submenu as shown in Figure 6.18 (9). Simply click on the drop-down menu under task (10) and select the new task that is created. The task name updates as an indication that the object will now cycle at the same speed as the set cycle of the task. The object is now completely set up and ready for activation.



Solution 'TwinCAT Project1' (2 projects)
- TwinCAT Measurement Project1
- TwinCAT Project1
  - SYSTEM
    - License
    - Real-Time
    - Tasks
      - Sim_Task
      - PlcTask
    - Routes
    - Type System
    - TcCOM Objects
      - Object1 (DCMotorCLSSSFBIC1)
        - DCMotorCLSSSFBIC1_U
        - DCMotorCLSSSFBIC1_Y
  - MOTION
  - PLC

Context: 0 'Context0 (10 ms)'
Depend On: Manual Config

Data Areas:
[✓] 0 'DCMotorCLSSSFBIC1_U'
[✓] 1 'DCMotorCLSSSFBIC1_Y'
[✓] 2 'DCMotorCLSSSFBIC1_B'
[✓] 3 'DCMotorCLSSSFBIC1_X'

Interfaces:

Data Pointer:

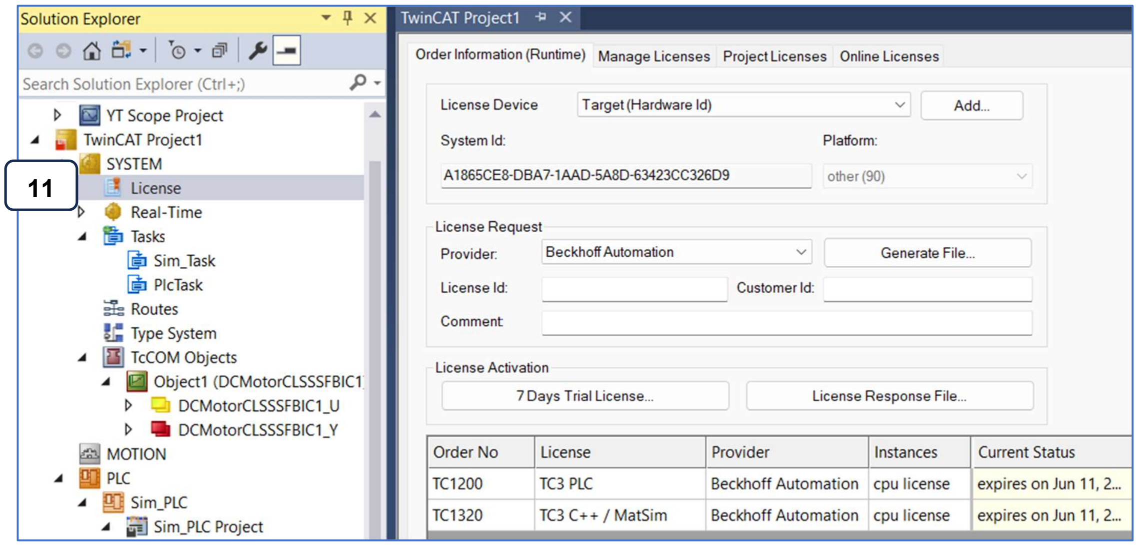
Interface Pointer:

Result:

ID	Task	Name	Priority	Cycle Ti...	Task Port	Symbol Port	Sort Order
0	02010020	Sim_Task	1	10000	350	350	0 (default=100)

Figure 0.18: Assigning a task to a TcCom Object

The last step to finish setting up the project is to activate the required licenses. Figure 6.19 shows the TwinCAT licensing menu (11) and the submenu which shows the current installed licenses. Licenses are removed or added under the manage licenses menu. The license called TC1220 allows the usage of the PLC, MATLAB, Simulink, and C++ functions. A trial license can be run for 7 days if development needs to be started before receiving the actual license. Once the license is activated it is possible to run the TwinCAT project.



License Device: Target (Hardware Id)

System Id: A1865CE8-DBA7-1AAD-5A8D-63423CC326D9

Platform: other (90)

Provider: Beckhoff Automation

License Id:

Customer Id:

Comment:

Order No	License	Provider	Instances	Current Status
TC1200	TC3 PLC	Beckhoff Automation	cpu license	expires on Jun 11, 2...
TC1320	TC3 C++ / MatSim	Beckhoff Automation	cpu license	expires on Jun 11, 2...

Figure 0.19: TwinCAT 3 Licensing Menu

Table 6.3 shows a list of errors and solutions to issues found when trying to activate the developed TwinCAT project. The errors are found in the TwinCAT error list. Once all the errors are cleared using the supplied solution, the project is active and in a running state.

Table 0.3: Possible errors when activating a TwinCAT project

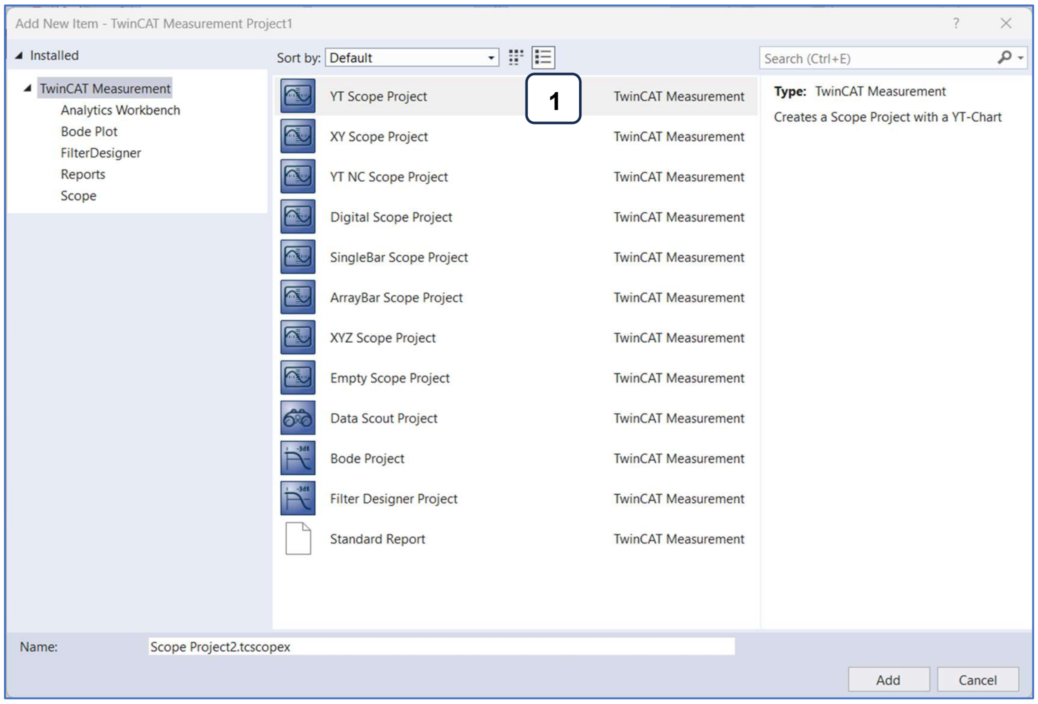
Error	Solution
Task assignment to context 0 is not valid	No task is added to the created TcCom object. Figure 6.20 shows how to add a task and Figure 6.21 shows how to assign a task to an object.
No certificate found in PE image file	This error occurs if the operating system is not in test mode, or if the driver signing process is not completed correctly. Either reboot in test mode or go through the driver signing process again to make sure no steps were skipped.
AdsWarning: 4115 (0x1013, RTIME: System clock setup fails.)	Hyper-V needs to be disabled as the service permanently runs in the background and crashes with certain TwinCAT drivers. Deactivate the service in the windows features menu.

This section describes the configuration of the TwinCAT project. The TcCom objects are inserted into the project, a new cycle task is created, the project is licensed, and all possible errors were looked at that prevented the project from activating. Now that the project is active and running, the following section describes how to set up the Scope Viewer to monitor the real-time response of the radio antenna control system.

6.5 Scope Viewer

This section describes the configuration of the Beckhoff Scope Viewer to create graphs to show the response of the TcCom object created in section 6.4. The graphical response is necessary to compare with the Simulink response obtained in Chapter 5 to prove that the transformation from Simulink to TwinCAT is successful. If the responses are the same, then the controller can be connected to the physical system for testing as the simulations are successful. A license is required to use all the features of Scope Viewer, but for this research, it is not necessary, as a demo license is used.

A new scope view can be created by selecting 'YT Scope Project' (1) when creating a new project, as shown in Figure 6.23. A YT scope graph shows the relationship between two variables across the Y and X axis. The two variables that need to be plotted are time and position. Time is on the X-axis and position is on the Y-axis.


Add New Item - TwinCAT Measurement Project1

Installed
- TwinCAT Measurement
  - Analytics Workbench
  - Bode Plot
  - FilterDesigner
  - Reports
  - Scope

Sort by: Default

YT Scope Project	TwinCAT Measurement
XY Scope Project	TwinCAT Measurement
YT NC Scope Project	TwinCAT Measurement
Digital Scope Project	TwinCAT Measurement
SingleBar Scope Project	TwinCAT Measurement
ArrayBar Scope Project	TwinCAT Measurement
XYZ Scope Project	TwinCAT Measurement
Empty Scope Project	TwinCAT Measurement
Data Scout Project	TwinCAT Measurement
Bode Project	TwinCAT Measurement
Filter Designer Project	TwinCAT Measurement
Standard Report	TwinCAT Measurement

Search (Ctrl+E)

Type: TwinCAT Measurement
Creates a Scope Project with a YT-Chart

Name: Scope Project2.tcscopex

Figure 0.20: Create a new scope viewer project

To confirm that the transformation from Simulink to TwinCAT is successful, a PLC program is used to generate the same random numbers as a set point input as the random number generator indicated in Section 5.3.5 in Chapter 5. Figure 6.21 shows the response in Simulink, whereas Figure 6.22 shows the response in TwinCAT Scope Viewer. Characteristics such as rise time, overshoot, settling time, and steady state error of the responses of both graphs are the same and therefore the transformation is successful.

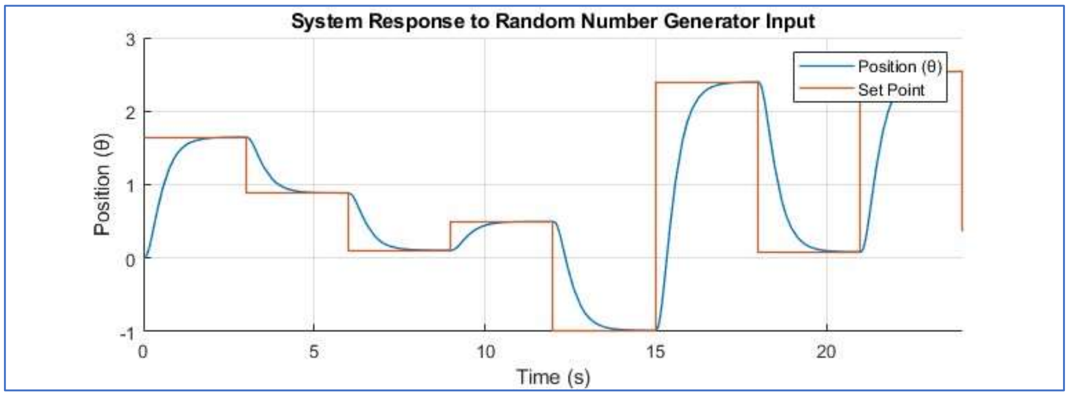


Figure 0.21: **Random set point generator in Simulink**

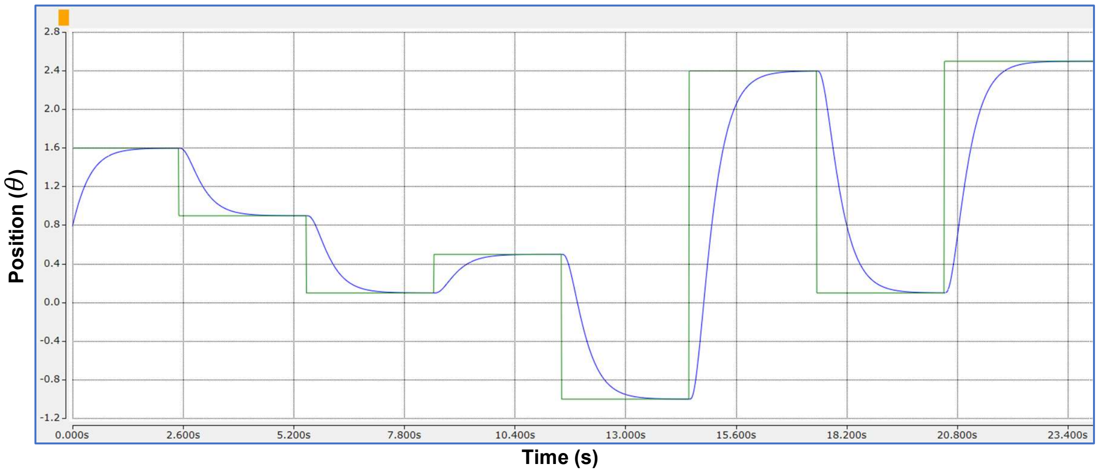


Figure 0.22: **Random set point generator in Scope Viewer**

This section described how to configure Scope Viewer to view real-time values of different variables. The section also proved that the transformation of Simulink block diagrams to TwinCAT is possible by comparing the response of the controller in both programming environments. The next section concludes the chapter, discussing the results that this chapter has produced.

6.6 Discussions

This section discusses the process, challenges, and significance of the successful transformation of the Simulink model to TwinCAT object. The transformation of components from one software vendor to another is a major factor to consider in terms of one of the deliverables in this thesis and therefore this chapter is very important. From installing the software to testing the response of the model with Scopeviewer, the entire process had some challenges.

Once all the requisite software is installed, the challenge is finding all the necessary tools required to do the transformation. Software packages such as MATLAB/Simulink and TwinCAT have an abundance of tools so finding the correct features is the difficult part. Features such as the TE1400 target have minimal documentation and therefore using the functionality takes time as more effort is put into figuring out how to use the tool than using the tool itself.

A big challenge is finding the correct MATLAB commands that need to be run to start the transformation process (Appendix A6.1). If a script is not used, it is necessary to set up each model to be in the correct format required for the TwinCAT software environment. When running the script, the code automatically uses the Simulink code builder, instead of the default MATLAB code builder. Also, each time a model is generated, a new version number is added to the name atuomatically, which could lead to confusion when trying to revert back to older versions.

Licensing is a major issue when trying to use features in the TwinCAT programming environment. A license is required for most features, but a 7 day trial is activated when using features that are unlicensed. The main license issue is that when using the TE1400 target, a signed certificate from Beckhoff is required to run Simulink models on the PLC. It is possible to do driver signing, as discussed in this chapter, to create a test certificate, but this will not allow the downloading of Simulink models to the PLC.

Finding support for the errors found while trying to compile software in MATLAB/Simulink and TwinCAT was not easy. Both software packages have information systems that show how to use most of their features, but lacks documentation on trouble shooting and error handling.

Even with the challenges faced, the results are significant as it is demonstrates that portability and interoperability between different software vendors is possible. Any developed Simulink model can now be used in TwinCAT 3 for real-time implementation purposes. An added advantage is using the Scopeviewer to see results of the real-time graphs.

6.7 Conclusion

This chapter described how to transform Simulink block diagrams into TwinCAT 3 TcCom objects to allow for real-time simulation in the TwinCAT programming environment. All the prerequisite software tools needed for this transformation are discussed, including installation guidelines and configuration steps. These software platforms include Visual Studio 2019, MATLAB/Simulink v2023a, TwinCAT 3.1 Runtime (XAR), and the TE1400 Target. Each one is installed on the same engineering PC, including the TwinCAT run time software which automatically installs when installing the TwinCAT XAR software.

Once the PC is set up, the Simulink block diagram is altered to allow for inputs and outputs for interfacing with the TwinCAT programming environment after the transformation is complete. The TE1400 Target software is used to transform the updated Simulink block diagram, and automatically add the item to the TcCom objects list in TwinCAT by running the set of MATLAB commands shown in Appendix A6.1.

A task is created to allow the TwinCAT object to execute during run time. The task is allocated under the objects properties and executes once the run time is activated after a successful build. The TC1220 license is required to build and run the TwinCAT project as this license allows the usage of the PLC, MATLAB, Simulink, and C++ functions.

Lastly, the transformation is tested by comparing the response of both the models in TwinCAT and Simulink by using the same random number set point generator as the input to each system. The graph outputs compared in Section 6.5 prove that the transformation is successful and that the setup guide is accurate.

The transformation from one programming platform to the other resonates with parts of the IEC 61499 standard requirements such as portability and interoperability. Another factor of the IEC 61499 standard shown is the reusability of software components between platforms without any modifications of the configuration files needed.

The next section describes how to implement the transformed Simulink model to control an actual DC motor in real time. The input and output variables of the Simulink model is mapped to a motor terminal card which controls and monitors the DC motor as required.

CHAPTER 7
HARDWARE-IN-LOOP IMPLEMENTATION FOR IEC 61499 COMPATIBILITY

7.1 Introduction

This chapter presents the real-time implementation of the transformed Simulink model. The model is used to control an actual DC motor. The motor is connected to a remote motor terminal card which communicates to the PLC over an EtherCAT network. The controller (transformed model) runs on the PLC and controls the plant (DC motor) by mapping the input and output variables for monitoring and control. The hardware equipment required for the control system is described before presenting the complete test rig used for implementation.

The CodeSYS PLC software which connects the transformed model, and the actual hardware is discussed. The code uses the PLCopen software function blocks to command the DC motor to change its position. PLC code that is programmed using a CodeSYS programming environment can be used in other vendors software packages. This means that Beckhoff's TwinCAT 3 software is IEC 61499 compliant as the PLC code can be reused and ported to other vendors. This is very advantageous as reprogramming is not necessary when using a different PLC.

How to activate the developed code onto the PLC runtime is also shown. The code cyclically runs on the PLC, waiting for a change in position set point via push buttons. Navigating and setting up the axis controller is shown to link the CodeSYS PLC software with the actual DC motor. The Beckhoff NC control features are used to configure the axis and the parameters of the encoder, drive, and controller. A full list of all configured parameters is provided for the successful implementation of the DC motor control system.

The closed loop system implemented on the hardware is analyzed with an input step response. The transient response and the effects of network delays of the system is discussed and compared to the same system with added time compensation.

Section 7.2 describes the hardware involved with the implementation of the DC motor position control. Section 7.3 describes the PLC code and interface with the Simulink controller module and the hardware. Section 7.4 describes the motion interface of the TwinCAT programming environment and how to set up the required axis for motion control. Section 7.5 presents the analysis of the implementation results of the closed-loop system.

7.2 Hardware Description

This section describes the hardware used to implement the designed controller. The hardware consists of a Beckhoff C6015 PLC, Beckhoff EK1100 EtherCAT remote module, Beckhoff EL7342 terminal motor controller, a DC motor, and an Omron incremental encoder for position feedback. Each item has a crucial role in the closed loop system. Once each component of the system is described, the complete test rig is shown. An electrical diagram is also provided for the setup.

7.2.1 Beckhoff C6015

The Beckhoff C6015 Industrial Personal Computer (IPC) is a Windows based PLC that is used for real-time control. The IPC hosts a 2 core Intel Atom processor which makes it capable of managing resource intensive tasks. Some of these tasks include axis control, short cycle times, high-volume data handling, hosting Human Machine Interface (HMI) applications, Internet-of-Things (IoT) connectivity, and a range of automation configurations (Beckhoff, 2023).

The IPC uses an Ethernet port for communication to other Ethernet based devices such as a programming PC or HMI interface. The second Ethernet port is configured to be used as an EtherCAT fieldbus system to communicate with other EtherCAT devices. In this case the IPC is the master EtherCAT device, and the other devices are the slaves. Ethernet port 1 and 2 are shown in Figure 7.1 as X102 and X103 respectively. Table 7.1 summarizes the interface ports that are used to interact with the IPC.

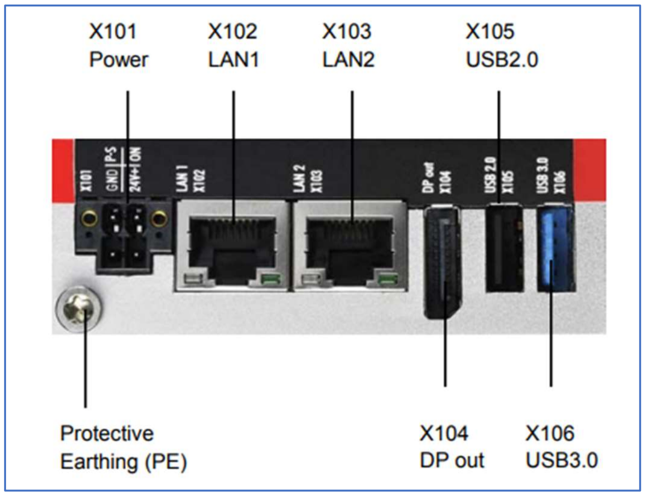


Figure 0.1: **Beckhoff C6015 PLC front view (Beckhoff, 2023)**

Table 0.1: **Beckhoff C6105 interface list and functional purpose**

Terminal Number	**Description**	**Purpose in test rig**
X101	Power supply	Terminal connection to connect 24-volt supply to power the PLC
X102	Network LAN connection 1	Ethernet port for connection of programming PC for downloading and online viewing of the runtime
X103	Network LAN connection 2	EtherCAT port to communicate to EtherCAT devices on the network
X104	Display port for video feed	Not used
X105	USB 2.0 port	Not used
X106	USB 3.0 port	Not used

In this work, the Beckhoff IPC hosts the TwinCAT 3.1 software configuration by using the Beckhoff automation run-time software. The IPC is also the gateway between the transformed Simulink model and the remote EtherCAT terminals which creates the connection to the DC motor and encoder feedback. PLC code is written to bridge the model and hardware together to close the control system loop. This PLC code runs on the IPC during run-time.

7.2.2 EtherCAT

EtherCAT is a fieldbus protocol that is based on a master-slave architecture. One device is the system serves as the master and is the only devices which can send an EtherCAT data frame over the network. The slave devices then read the necessary information from the data frame according to the hardware and software requirements, and then attach necessary information onto the data frame. This transaction is known as “processing on the fly” and is a great advantage for the protocol as it reduces network delays and network traffic (Stubbs, 2011).

EtherCAT uses a distributed clock system that allows the master to determine the time taken for the transmission of information to the slave devices. This is done when the slave devices timestamp the data frame every time it receives data. The master can then calculate the transmission rates to each slave to help with real time operation and avoid delays.

Other advantages of EtherCAT are the rapid response times, minimal data requirements for each device, and low cost of implementation. Another positive part is that no external Ethernet switch is required as EtherCAT device have two Ethernet ports available. One port is used for receiving the data frame from the master and the other port is used to transmit the data frame to the next EtherCAT device. The test rig built for this research work only has one master and one slave device. The slave device connected in a line topology as the Master only has one EtherCAT port. This slave device is described in the next subsection.

7.2.3 EK1100

The Beckhoff EK1100 is a gateway device between the EtherCAT master and the terminal cards that are connected on the same backplane when mounted on the slot next to it. This device takes the EtherCAT telegrams sent from the EtherCAT master and converts it to whatever signal is required from the connected terminal cards. This conversion also works the other way for when the terminal card needs to transmit data back to the EtherCAT master. Examples of terminal cards that can slot onto the EK1100 backplane are digital inputs, digital outputs, analogue inputs, analogue outputs, etc. This module allows for remote connection of any device which needs information sent back to a central IPC.

As shown in Figure 7.2, the EtherCAT master connects to the input EtherCAT port. The EtherCAT signal output port can be used to create a daisy chain to the next device. The green and orange lights on the left of the RJ45 ports (1) indicate if the link is active to another device when flashing. These lights are off if no connection is present. The status of the remote interface is shown on the Light Emitting Diodes (LEDs) and the power connections can connect to the red plus (24V) and blue minus (0V) marked terminals. The module is earthed via the PE labelled terminals.

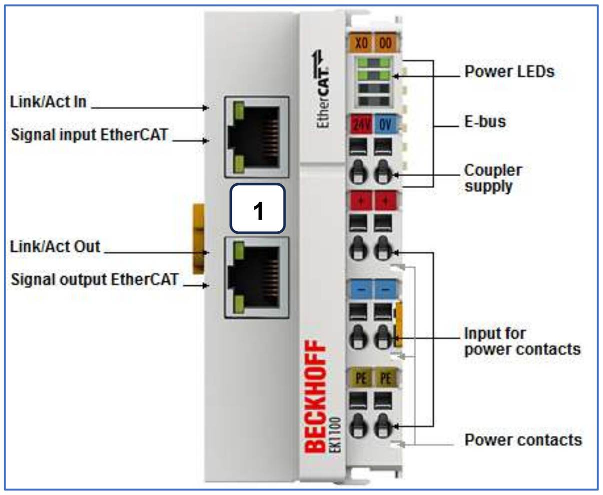


Figure 0.2: Beckhoff EK1100 interface ports and indication lights (Beckhoff, 2023)

In this research project, the EK1100 is used as an interface between the IPC and the remote terminal modules that control the motor and feedback signals. The EtherCAT gateway allows the programming software to control the DC motor over a networked control system. The next subsection describes the terminal card use for motor control.

7.2.4 Beckhoff EL7342

The Beckhoff EL7342 is part of Beckhoff's compact drive technology range which allows for the direct integration of motors in the Input/Output (I/O) system. The range of terminal control cards caters for most types of motors, such as DC, AC, stepper, and servo motors. The EL7342 specifically controls DC motors and is therefore used in this research work for the control of the two DC servo motors for azimuth and altitude control of a radio antenna dish.

The EL7342 can operate DC motors with a voltage up to 50 Vdc and a current of up to 3.5 A. The motors are connected directly onto terminals A1 and A2, or B1 and B2, as shown in Figure 7.3. Incremental encoders can be connected to the terminal card to realize a simple servo axis for each motor. In this research, an incremental encoder is used to send feedback of the position count of the actuated DC motor.

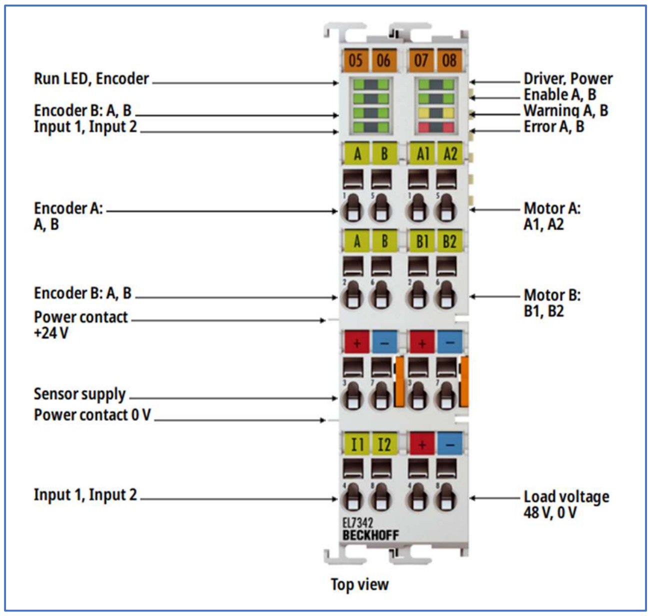


Figure 0.3: **Beckhoff EL7342 interface ports and indication lights (Beckhoff, 2023)**

7.2.5 DC Motor

As described in Chapter 3, DC motors are used to control the movements of a radio antenna. The DC motor used in Chapter 4 for modelling has part number GB37Y360. The DC motor is rated at 12 Vdc and 1.6 A. The rated speed of the motor is 6000 RPM, but with the gearbox ratio of 1:270, the motors shaft output rated speed is 270 RPM.

7.2.6 Omron Encoder

An incremental encoder is used to measure the angular position of the shaft of the DC motor. Specifically, an Omron E6C2-CWZ5B 100 pulse encoder is used. The wiring diagram for the encoder is shown in Figure 7.4, where the blue, brown, black, and white wires are connected directly to the EL7342 which is described in Section 7.2.4. The EL7342 does not have the functionality to measure the Z phase of an incremental encoder and therefore the orange wire is not connected.

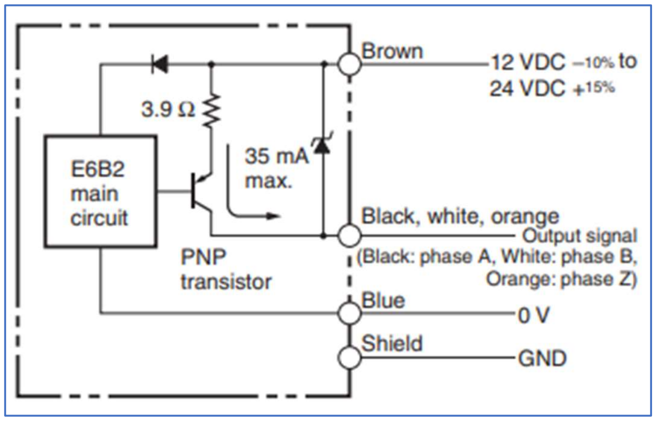

Brown
12 VDC −10% to 24 VDC +15%
3.9 Ω
35 mA max.
E6B2 main circuit
PNP transistor
Black, white, orange
Output signal (Black: phase A, White: phase B, Orange: phase Z)
Blue
0 V
Shield
GND

Figure 0.4: **Omron encoder wiring diagram (Omron, 2023)**

An incremental encoder sends pulses through the A and B phase wires to the inputs of the EL7342. The A and B channels pulse at an offset to determine which way the motor is rotating. For example, if the A phase pulses before the B phase, then the motor is turning in one direction, whereas if B pulses before A, then the motor is turning in the opposite direction. The terminal controller then counts the rising edge of the pulses to determine how far the motor has rotated and displays a scaled angular position value in the IPC. The scaled value is based on how many pulses equal one rotation of the motor.

For this research work, the two phases are viewed as a quadrature encoder by counting the rising and falling edges of both A and B phases. This increases the resolution of the encoder by four times. Therefore, since the encoder pulses 100 times per revolution, the scaled value is calculated by a pulse rate of 400 pulses per revolution. The higher the resolution of the encoder, the more accurate the position feedback is.

The previous subsections discussed all the hardware components for the test rig that is built for this research work. The next subsection describes the complete test rig.

7.2.5 Complete Test Rig

Figure 7.5 shows the complete test rig used to implement the controller and plant. The rig is built up of all the previously described hardware components. Each component is numbered and described in the list below. A wiring diagram for the test rig is shown in Appendix C7.1.

1- AC voltage distribution breakers
2- 24 Vdc power supply
3- 12 Vdc power supply
4- Beckhoff C6015 IPC
5- Beckhoff EK1100 EtherCAT coupler
6- GB37Y360 12V DC motor
7- Omron E6C2-CWZ5B incremental encoder
8- Push buttons

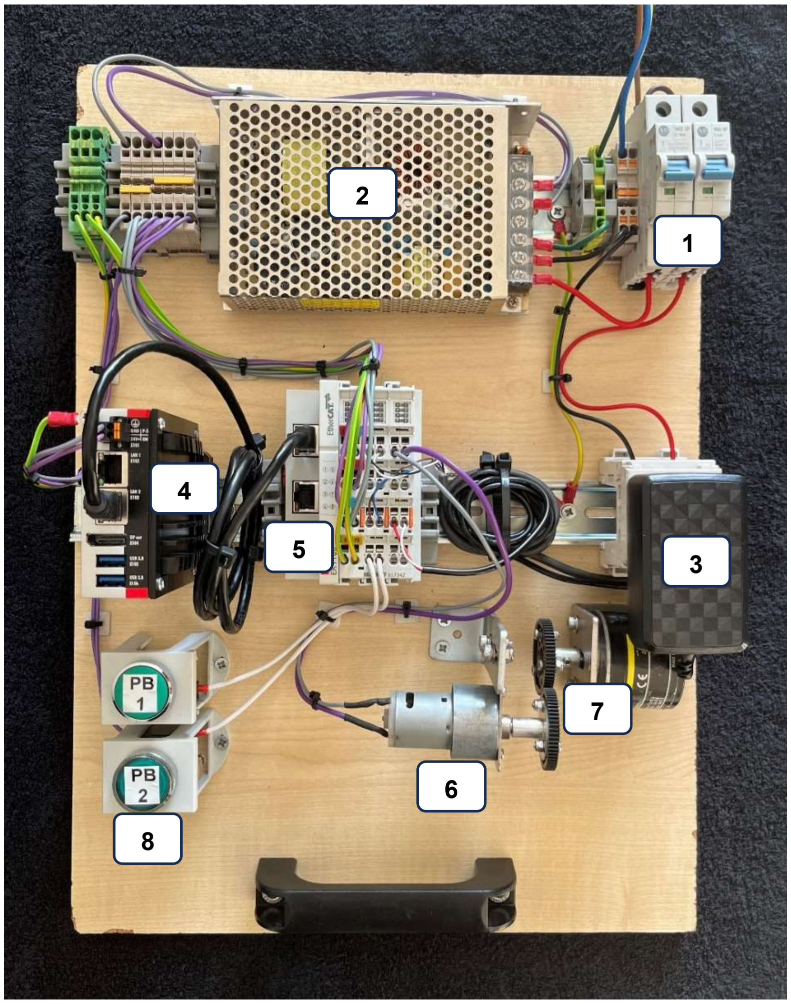
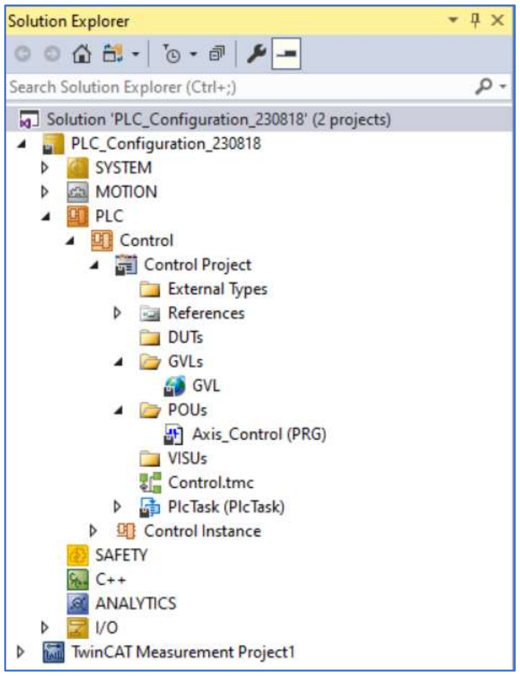


Figure 0.5: DC Motor Control System Test Rig

7.3 PLC software

This section describes the PLC code which interfaces the transformed Simulink model and the actual DC motor. A PLC configuration is added to the TwinCAT XAE programming environment by right clicking on PLC in the solutions explorer window and adding a new item as shown in Figure 7.6. The PLC item is made up of various subfolders that can be used for configuring the project, such as Data Unit Types (DUTs), Global Variable Lists (GVLs), and Program Organization Units (POUs). This project only uses GVLs and POUs.


Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'PLC_Configuration_230818' (2 projects)
  PLC_Configuration_230818
    SYSTEM
    MOTION
    PLC
      Control
        Control Project
          External Types
          References
          DUTs
          GVLs
            GVL
          POUs
            Axis_Control (PRG)
          VISUs
          Control.tmc
          PlcTask (PlcTask)
        Control Instance
    SAFETY
    C++
    ANALYTICS
    I/O
  TwinCAT Measurement Project1
```

Figure 0.6: PLC configuration in Solution Explorer

The GVL is where all the PLC variables are declared by name and datatype. These variables can be used in any part of the PLC code as they are listed as global. The PLC code is found in the POU called “Axis_Control”. This is the main and only POU of the program and runs cyclically according to the PLC task period.

Subsection 7.3.1 describes the PLC programming environment used, whereas subsection 7.3.2 describes the function block libraries that is used to control the DC motor. Subsection 7.3.3 describes the flow and functionality of the developed PLC code found in POU “Axis_Control”. Subsection 7.3.4 describes the converted controller function block. Subsection 7.3.5 describes how to activate and download the code to the IPC for real-time implementation.

7.3.1 CodeSYS

Controller Development System (CodeSYS) is a programming environment used to program, configure, and manage PLCs and other automation devices. This proprietary development platform is used is used by various PLC manufacturers making it more flexible and versatile than any other programming environment. Suppliers such as Beckhoff Automation, WAGO, Phoenix Contact, Eaton, and Turck, all use CodeSYS as the basis of their PLC programming software.

The premise of CodeSYS is in line with the IEC 61499 standard as code can be ported between the various vendors' programming software packages without any changes being made. The benefits of this portability allows developers to easily port software between different hardware components. If a vendor does not have stock of a certain PLC, the developer can use any other PLC that is Codesys compliant. Other advantages include the robustness of the PLC environment, the cross-platform compatibility, inclusion of visualization for HMI development in the same environment, possibility for simulation, and the means to integrate third-party libraries and components.

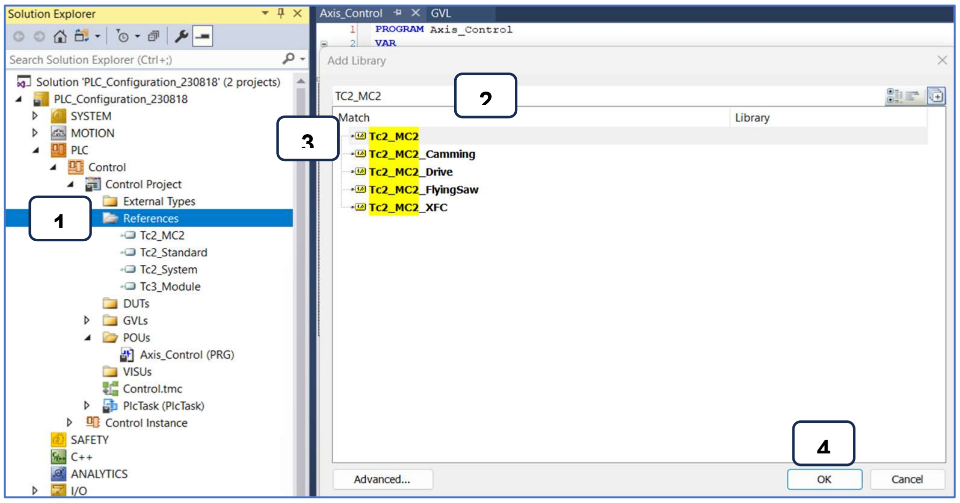
The TwinCAT programming environment uses the CodeSYS environment as a basis for PLC development. The next subsection describes the library that needs to be installed to allow for servo motion control of the DC motor.

7.3.2 PLCopen Motion Control

The DC motor is controlled as a servo motor via the TwinCAT software environment. This is done using PLCopen motion control blocks to enable, reset, move, and stop the axis which controls the DC motor. The PLCopen motion control library is an open-source standardized repository that is used by most PLC suppliers for motion control. The library is interoperable between these suppliers, therefore a change in code is not necessary when porting from one software environment to another. The library is used to simplify PLC code by encapsulating all the necessary code for various motion commands into easy-to-use function blocks. Other advantages of using the PLCopen motion control library include safety assurance as the function blocks have been tested thoroughly, there is lots of online support, and scalability as the function blocks can be used on as many axes as the controller can support.

It is necessary to install the required libraries to TwinCAT 3 to use the functionalities. Figure 7.7 shows how to add a library as a reference in a PLC project in TwinCAT 3. The PLCopen library used in this project is called TC_MC2. As shown in Figure 7.7, right click on references

(1) and add a library. Then search for the required library (2) and highlight it to approve selection. Click OK to add the library as a reference. The library and its functionality are now available for use with the PLC project.



```
PROGRAM Axis_Control
VAR
```

TC2_MC2

Figure 0.7: **Adding a library reference to TwinCAT 3**

The PLC programming environment and all necessary libraries are now installed and therefore the PLC code development can start. The next subsection describes the flow and functionality of the developed PLC code to control the position of the DC motor via the mapped variables from the transformed Simulink model.

7.3.2 PLC code functionality

The PLC code runs cyclically according to the PLC task time. To prevent code from executing incorrectly, a sequence step is used to only run sections of the program at once until certain conditions are met. Figure 7.8 illustrates a flow chart of how the PLC program executes and what conditions are required before moving to the next step in the DC motor position control sequence. The sequence starts from the initial start of the PLC and continuous through the step cycle during runtime.

Four steps are used for the position control. The first step, step 0, rehomes the axis to zero the position. Step 1 waits for a change in set point before executing the PLCopen relative move function blocks in step 2. Steps 3 and 4 are used to stop and autocorrect the position respectively. Steps 0 – 4 are described in more detail in subsection 7.3.2.1 to 7.3.2.4. All PLC code associated with the different steps are described and analysed in detail to show the functionality of the angular position control of the DC motor.

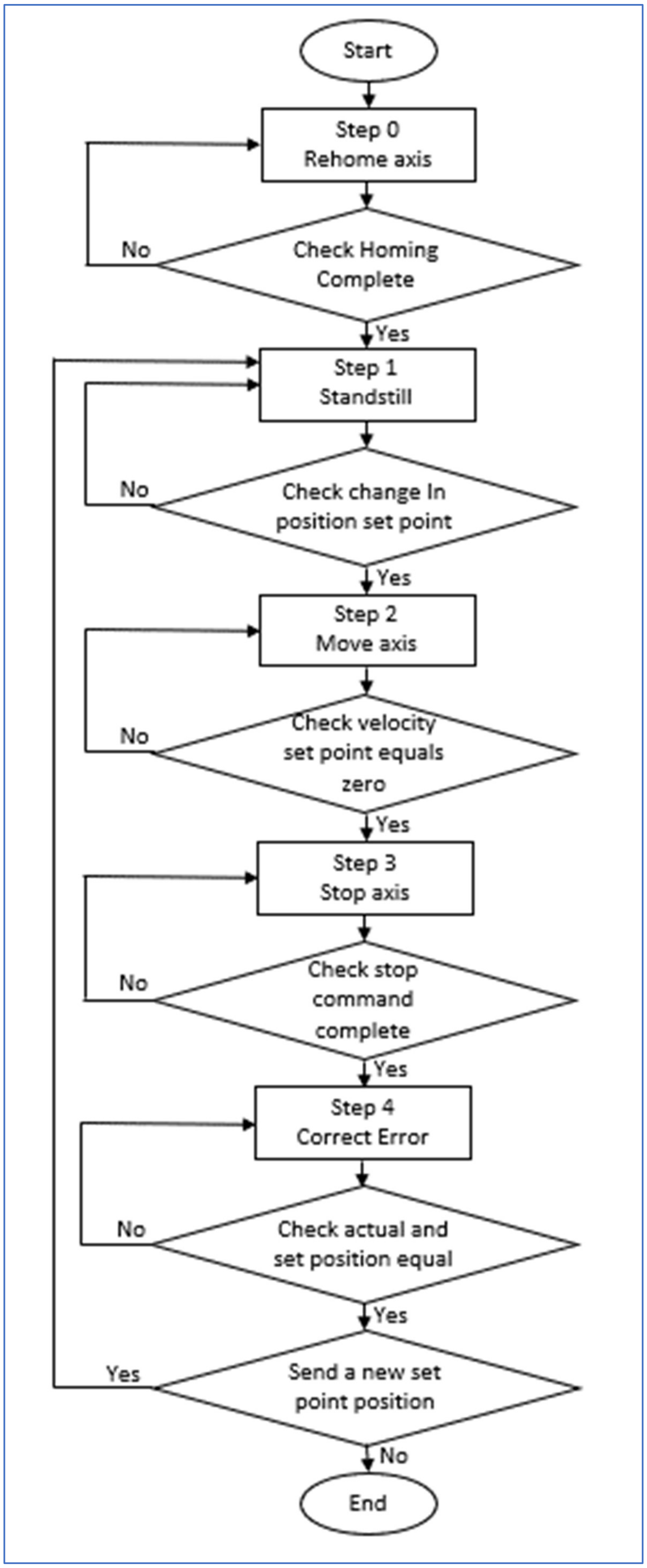

Start
Step 0
Rehome axis
No
Check Homing Complete
Yes
Step 1
Standstill
No
Check change In position set point
Yes
Step 2
Move axis
No
Check velocity set point equals zero
Yes
Step 3
Stop axis
No
Check stop command complete
Yes
Step 4
Correct Error
No
Check actual and set position equal
Yes
Yes
Send a new set point position
No
End

Figure 0.8: PLC step sequence flow chart

7.3.2.1 Step 0: Homing the axis

Step 0 is the initial step that is active when the PLC goes into run mode. This is because the variable Sequence_Step is not retentive and therefore cannot save a value when the PLC switches off or goes into stop mode, resetting the variable to zero. In Figure 7.9, the sequence step is equal to zero and therefore the MC_SetPosition block is executed. This PLCopen standard block sets the position of the referenced axis to whatever position set point is allocated. In this case, the axis position is set to zero to force the axis to take the current position as its home. Once the MC_SetPosition done bit is activated, a value of one is moved into sequence step, as shown in line 12. The axis is now at a zero point and ready for positional movements based on an angular position set point change.

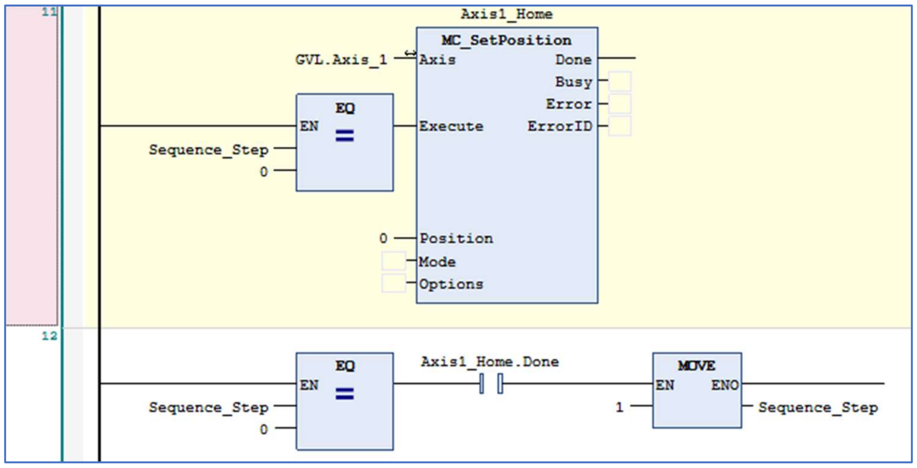


Figure 0.9: Ladder logic for step 0

7.3.2.2 Step 1: Waiting for a change in position set point

In step 1, shown in Figure 7.10, the axis is at a standstill and waiting for a change in position set point. The condition to go to the next step is for the set point to be different from the axis actual position feedback. On startup both the set point and feedback are at zero due to the homing of the axis in the previous step. By pushing either the forward or reverse push buttons on the test rig, the contact for input1_enable or input2_enable switches. The causes the set point to either increase by 10, or decrease by 10, depending on which direction the motor needs to travel. An R_TRIG function is used to only allow the change in set point to occur on a rising edge of the button input. This is done to prevent any debounce or multiple additions from the ADD block. If either button is pushed, the set point value is not equal to the actual value, and therefore the condition to move to step 3 is met. The next step moves the actual axis to the new set point value.

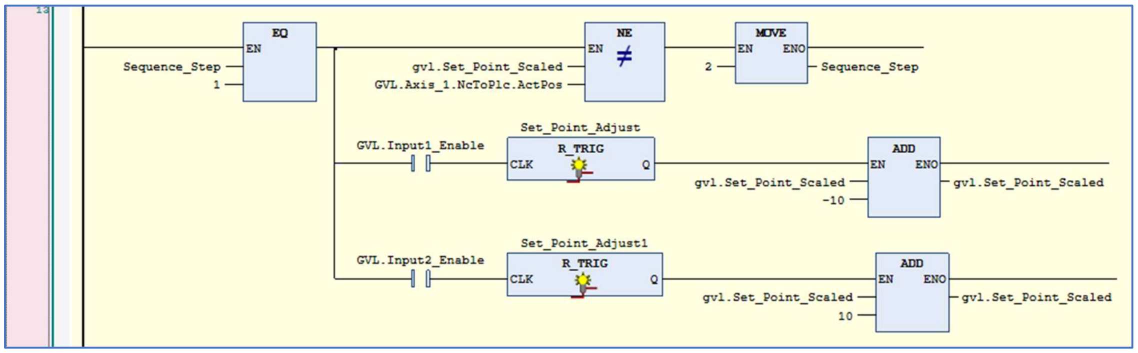


Figure 0.10: Ladder logic for step 1

7.3.2.3 Step 2: Moving the axis

When the sequence is at step 2, the axis begins to move by commands from two MC_MoveRelative function blocks, as shown in Figure 7.11. The two blocks are alternated by means of a toggle bit which activates every 20 milliseconds. The busy bit of each relative move block is used as a condition to execute the latter to prevent both moves from executing together.

The velocity, acceleration, deceleration, and jerk inputs are all mapped from the outputs of the transformed Simulink model to allow for the axis to respond in the exact same way as the model. The distance input is a scaled value calculated by subtracting the actual position of the DC motor from the set point. As the DC motor moves closer to the set point, the distance required to travel decreases and therefore the distance input decreases. As the relative blocks are toggled, the input variables constantly update to change the response of the motor. The motor continues to turn until the velocity input reaches zero, preventing the motor from running as a zero input means the motor must be at a standstill position.

The MC_MoveRelative block is not able to execute when any of the input variables are negative values. Therefore, the inverted values are used as inputs, which are calculated by multiplying the actual values by -1 if the Simulink model outputs any negative variables. This makes sure that the velocity, acceleration, deceleration, and jerk are always positive numbers. A quick time delay is added before checking for zero velocity as there is a delay before the relative move blocks initiate a response on the actual motor.

The condition to go to the next step is having a zero-velocity output from the Simulink model. This occurs once the model has reached the angular positional set point and therefore the axis are also at standstill and have reached the set point.

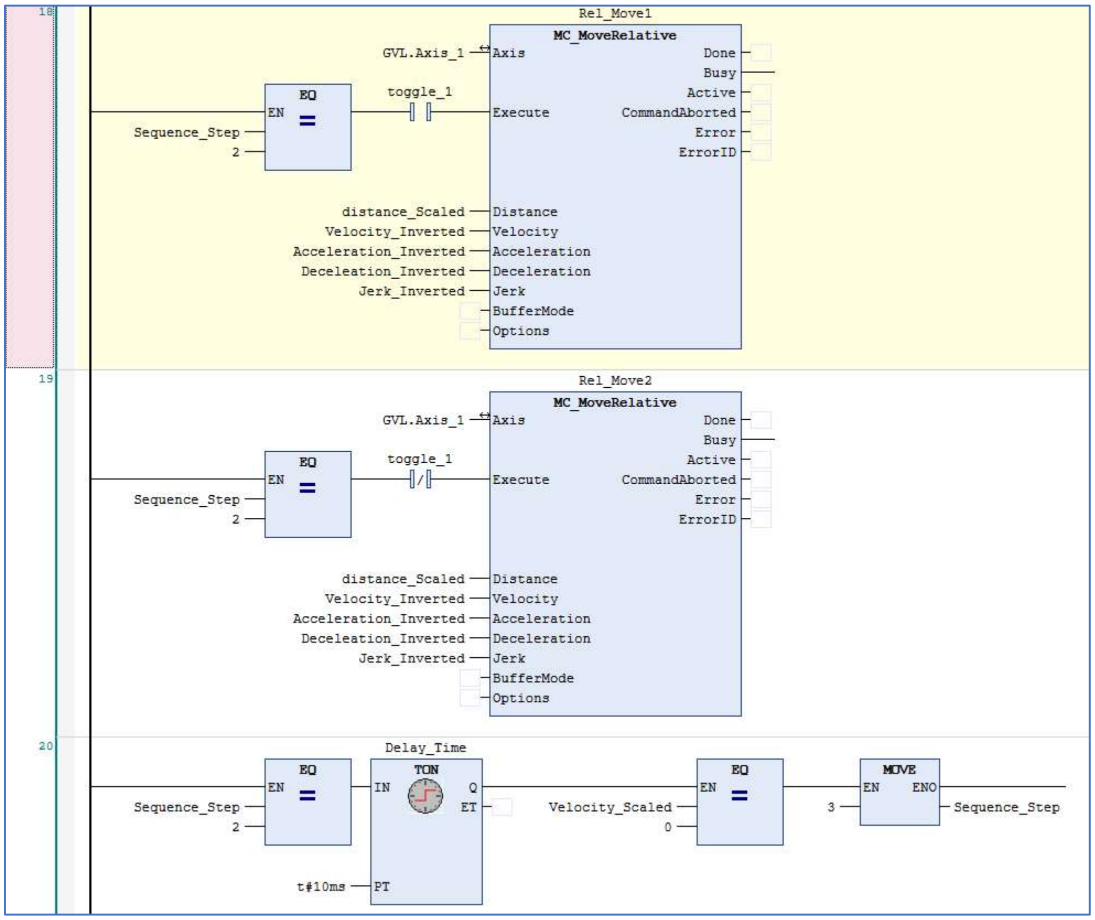


Figure 0.11: Ladder logic for step 2

7.3.2.4 Step 3 and 4: Stop the axis and autocorrect position

Steps 3 and 4 occur after the motor has reached the position set point. These two steps are added for safety in the event of any oscillation or stability issues with the motor. Step 3 forces the motor into a stop mode to prevent any further movement of the motor. Step 4 then moves the current axis position into the set point variable using a move function block. This is done to compensate for any minor deviation of the actual position and the set point that could have occurred due to any nonlinearities during the movement of the DC motor.

A short time delay is added to make sure there is no deviation between the actual position feedback and the position set point before moving back to step 1 as shown in Figure 7.12 line 24. Since in step 1 the condition to continue is a difference between the actual position and the set point, the code is looped back to step 1 and waits for another change in the set point. A change in set point can be achieved by pushing either of the push buttons for forward or reverse movement as previously discussed in step 1.

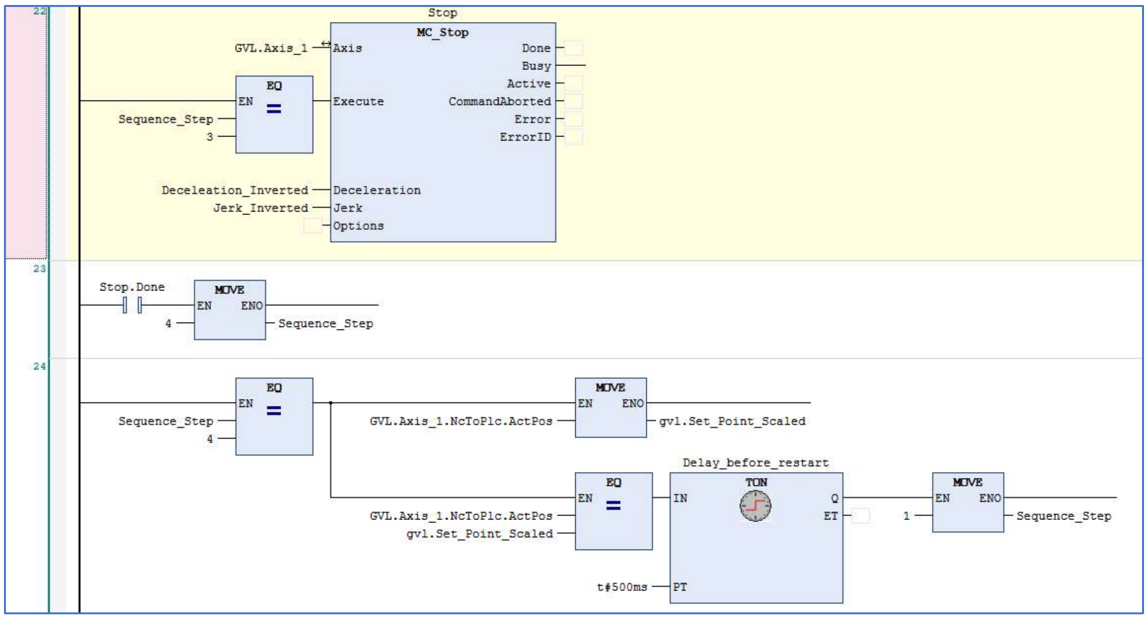


Figure 0.12: Ladder Logic for steps 3 and 4

Subsection 7.3.2 describes the PLC code that interfaces between the transformed Simulink model and the actual DC motor. The next subsection describes how to use the transformed Simulink model in the CodeSYS programming environment.

7.3.4 Interfacing with the transformed Simulink model

The Simulink model described in Chapter 5 is used in the CodeSYS environment to control the DC motor. The DC motor angular position set point is measured in millimetres and therefore the Simulink model is updated to include the conversion of all the set points and feedback signals from radians per second to millimetres per second. This conversion is shown in Figure 7.13, where the position set point, encoder feedback, actual position, actual velocity, actual acceleration, actual deceleration, and actual jerk are all scaled as necessary. Equation 7.1 is used to convert the set points to radians, whereas Equation 7.2 is used to calculate the feedback signals back to millimetres per second.

$$radians\ per\ second = \left(\frac{millimeters\ per\ second}{1000 * gear\ radius}\right) \qquad \textbf{(7.1)}$$

$$millimeters\ per\ second = radians\ per\ second * 1000 * gear\ radius \qquad \textbf{(7.2)}$$

Another change to this model is a saturation block (1) on the actual velocity signal to prevent the controller from trying to turn the motor at a velocity that is not possible. The DC motor used has a maximum velocity of 565 mm/s. The acceleration, deceleration, and jerk are derived from the output velocity by using the derivative blocks (2) within Simulink.

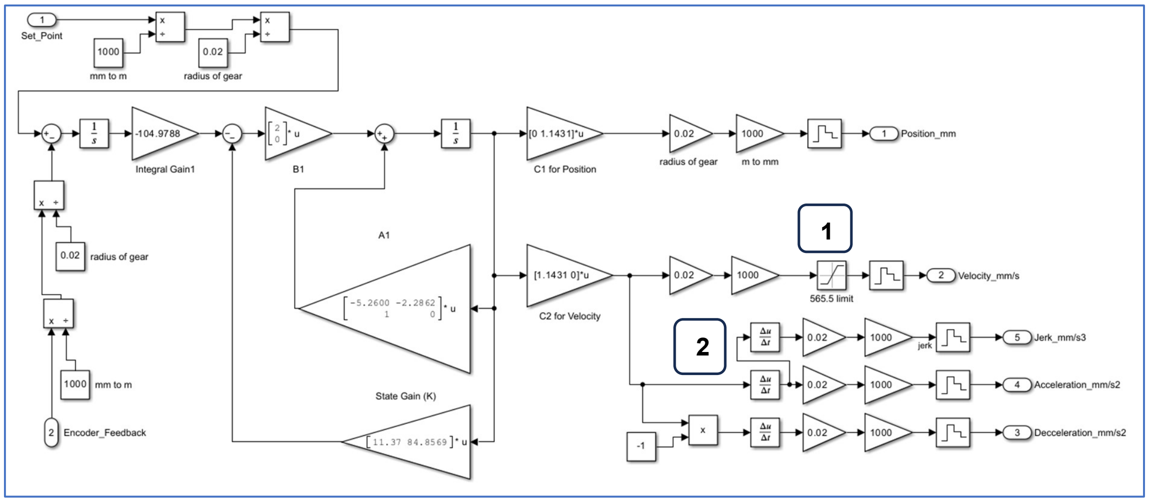


Figure 0.13: Updated Simulink block diagram of the DC motor with full state feedback and integral control

Figure 7.14 shows the transformed Simulink block diagram in CodeSYS function block form. The function block has the same inputs, outputs, and functionality as the Simulink block diagram in Figure 7.13. All that is required is to map the necessary PLC variables to the pins of the block to create a link between the Simulink function blocks and the PLC code within the Beckhoff IPC. The PLC variables created in the GVL are used, as well as the actual position that is directly linked to axis 1 for encoder feedback.

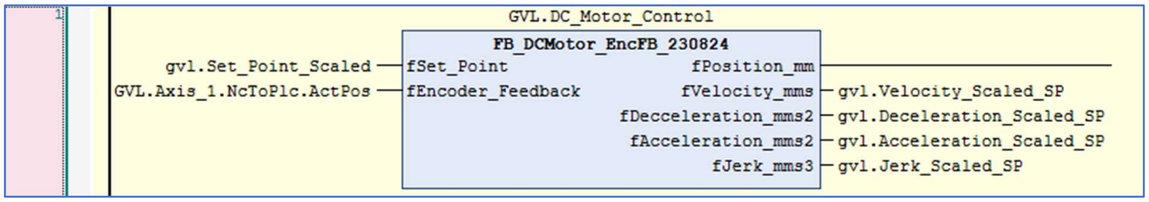

```
GVL.DC_Motor_Control
FB_DCMotor_EncFB_230824
gvl.Set_Point_Scaled ── fSet_Point                    fPosition_mm ──
GVL.Axis_1.NcToPlc.ActPos ── fEncoder_Feedback        fVelocity_mms ── gvl.Velocity_Scaled_SP
                                          fDecceleration_mms2 ── gvl.Deceleration_Scaled_SP
                                          fAcceleration_mms2 ── gvl.Acceleration_Scaled_SP
                                                  fJerk_mms3 ── gvl.Jerk_Scaled_SP
```

Figure 0.14: Transformed Simulink model in CodeSYS function block form

This subsection describes the functionality of the PLC code for positional control of a DC motor. The PLC code is ready to be deployed to the PLC for real-time testing. The next section describes how to activate the PLC code of the Beckhoff C6015 IPC.

7.3.5 Activating the PLC Configuration

Deploying PLC code to the Beckhoff C6015 is called “Activating the Configuration” in the TwinCAT 3 software environment. This is done once the PLC code is complete and ready for testing in the real-time environment. To activate the current configuration onto the PLC, click the first symbol on the task bar after the current build description, as shown in Figure 7.15. It is also possible to click project and then activate this configuration. The configuration is downloaded, and the green gear symbol activates showing that the run time is now active.

Figure 7.15 also shows the taskbar status when there is no connection to the PLC. The symbols shown at (2) from left to right are the login, start PLC, stop PLC, and logout buttons. The login button is available to interact with which means that the programming environment is not currently logged in to the PLC. Figure 7.16 shows the status of the connection at (3) once the login button is pushed. The connection is successful as the option to put the PLC in run mode is available. If the run button is clicked, the run button fades and the stop button becomes available as shown in Figure 7.17 at (4).

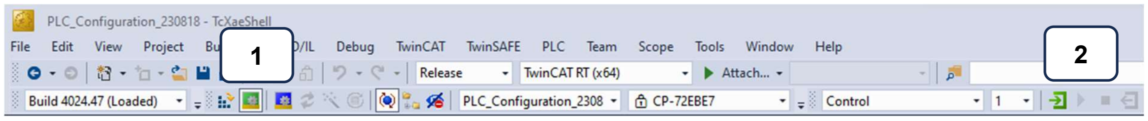

PLC_Configuration_230818 - TcXaeShell
File Edit View Project Bu /IL Debug TwinCAT TwinSAFE PLC Team Scope Tools Window Help
1
2
Release
TwinCAT RT (x64)
Attach...
Build 4024.47 (Loaded)
PLC_Configuration_2308
CP-72EBE7
Control
1

Figure 0.15: **Taskbar when not connected to the PLC**

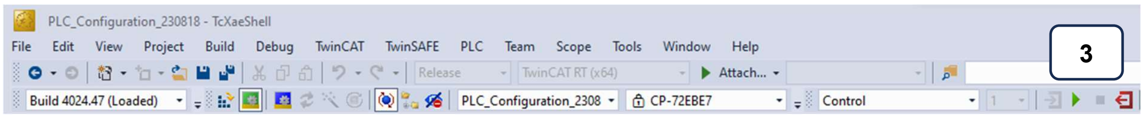

PLC_Configuration_230818 - TcXaeShell
File Edit View Project Build Debug TwinCAT TwinSAFE PLC Team Scope Tools Window Help
3
Release
TwinCAT RT (x64)
Attach...
Build 4024.47 (Loaded)
PLC_Configuration_2308
CP-72EBE7
Control
1

Figure 0.16: **Taskbar when connected to the PLC which is in stop mode**

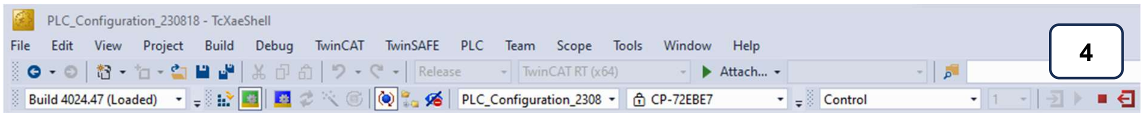

PLC_Configuration_230818 - TcXaeShell
File Edit View Project Build Debug TwinCAT TwinSAFE PLC Team Scope Tools Window Help
4
Release
TwinCAT RT (x64)
Attach...
Build 4024.47 (Loaded)
PLC_Configuration_2308
CP-72EBE7
Control
1

Figure 0.17: **Taskbar when connected to the PLC which is in run mode**

The PLC code is now active on the IPC in real-time. When the PLC is in run mode, the code interacts with all the mapped variables that were created. These variables include the interfacing variables to the Simulink model as well as the variables linked to the axis through the terminals on the EtherCAT coupler. The Simulink model and PLC side have been configured, but the link to the axis has not. The next section describes the configuration of the axis using Beckhoff’s motion control technology.

7.4 Motion Control

Beckhoff's motion control in the Twincat 3 software environment is used to configure the axis that is used to interface with the DC motor. Motion control allows for precise and synchronized motion of the DC motor according to a specific set point. Since the DC motor follows the transformed Simulink model, motion control is necessary to achieve the same response.

First, it is necessary to scan the EtherCAT network to find and configure the slave hardware device that interfaces with the DC motor and encoder feedback. Subsection 7.4.1 describes how to scan for EtherCAT devices on the network. Subsection 7.4.2 describes the navigation of the motion portion in the solution explorer, as well as the important parameters that are set to control the current DC motor. Subsection 7.4.3 describes the linking of the configured axis to the PLC software which is shown in Section 7.3.

7.4.1 Scanning for EtherCAT device

The EtherCAT coupler and motor terminal controller discussed in sections 7.2.3 and 7.2.4 must be added to TwinCAT 3 to allow for interfacing with the modules. This can be done by completing a network scan shown in Figure 7.18 (1) through the EtherCAT master. Figure 7.19 shows the EK1100 (2) and EL7342 (3) appear after doing a network scan. These devices can now be configured as well as used in the rest of the PLC project. By scanning and adding the EL7342, TwinCAT automatically adds two new axes to the motion tab. These are discussed in the next subsection.

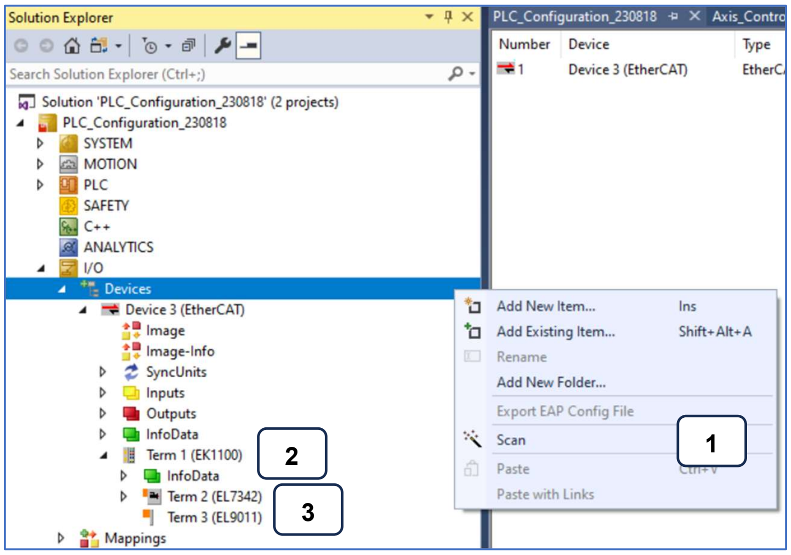


Figure 0.18: Menu navigation to scan the EtherCAT network

7.4.2 Navigation and parameter settings

The Motion section in the Solution Explorer section in TwinCAT 3 software is used to configure the axes that interface with the DC motors. As shown in Figure 7.19, each axes has menus to configure the axis itself, the encoder, the drive, and the control mode. Each heading has multiple parameters that can be configured depending on the type of DC motor used, the type of encoder, and the method of control that is needed. The inputs and outputs associated with each axis which can be linked to the rest of the TwinCAT project are also shown.

Settings for the two tasks that run the axes are also available under the heading NC-Task Sentence Execution (SAF) and Sentence Preparation (SVB). The SAF task handles the cyclic communication with the drives and executes the servo blocks. The SVB task prepares the execution commands to allow for them to be executed quickly by the SAF task. Both tasks are set to 1 millisecond in this project.

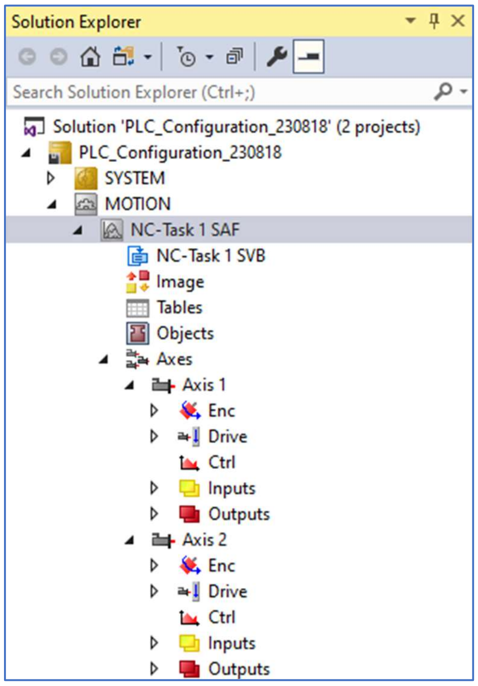


Figure 0.19: **TwinCAT 3 motion navigation in solution explorer**

Table 7.2 shows a summary of the parameters that are set for this project to allow for accurate position control of a DC motor through axis 1. A description is given for each parameter that is needed as well as the value that the parameters are set to. These values are calculated from the motor parameters as well as the type of encoder used.

Table 0.2: **Parameters changes for Axis 1**

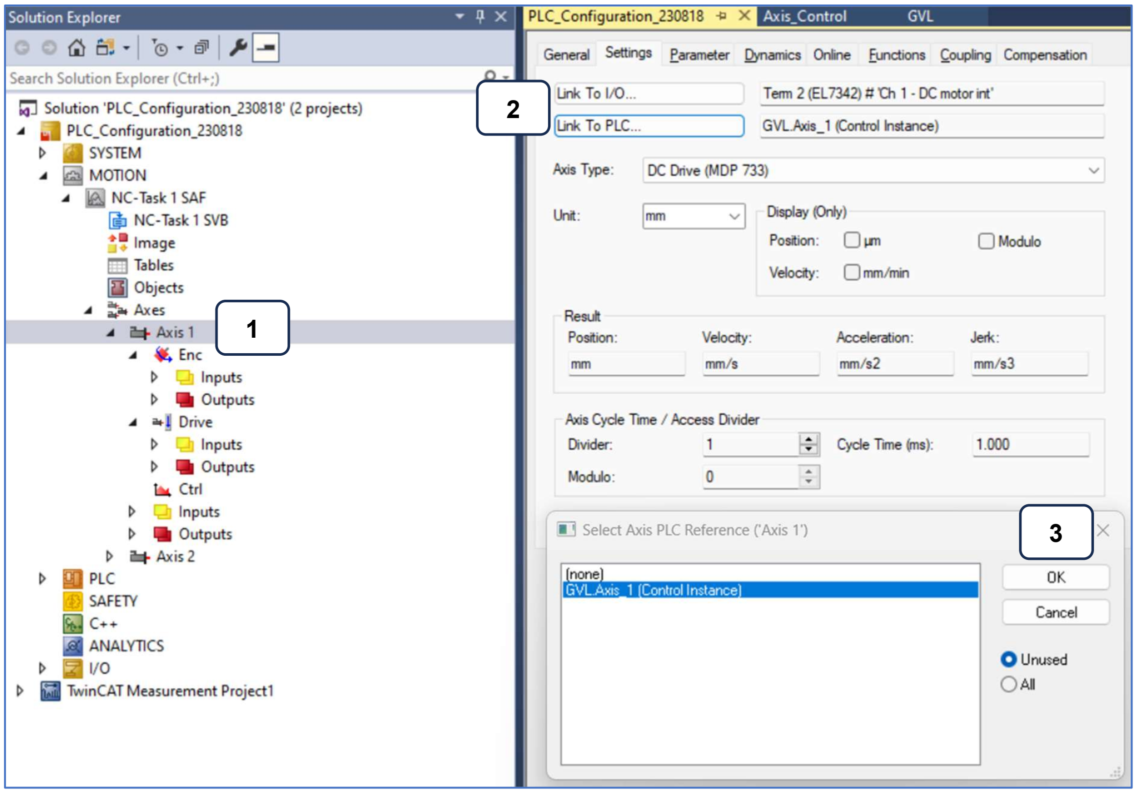
Heading	Parameter	Description	Value
Axis 1	Reference Velocity	Rated speed of the motor	84.78
Axis 1	Maximum Velocity	Maximum allowed speed for the motor	84.78
Encoder	Scaling Factor Numerator	Numerator for the scaled encoder value	18.84
Encoder	Scaling Factor Denominator	Denominator for the scaled encoder value	400.0
Encoder	Filter Time for Actual Position	Time between samples of the actual position	0.001
Encoder	Filter Time for Actual Velocity	Time between samples of the actual velocity	0.001
Encoder	Filter Time for Actual Acceleration	Time between samples of the actual acceleration	0.001
Ctrl	Proportional Factor Kv (standstill)	Proportional gain factor of the P-controller at standstill	20.0
Ctrl	Proportional Factor Kv (moving)	Proportional gain factor of the P-controller while moving	15.0
Ctrl	Proportional Factor Ka	Proportional gain factor for acceleration	0.15

The axis is configured and can be linked to the PLC code developed in section 7.3 The next subsection describes how to create this link.

7.4.3 Linking axis to PLC code

Now that the axis is set up it is required to link the axis to the PLC code. Figure 7.21 shows the configuration menu used to link an axis to a peripheral inside the TwinCAT 3 project. Select axis 1 as shown in Figure 7.20 (1) and open the settings tab in the submenu. It is possible to configure the link to I/O and the link to the PLC (2) under this tab. The axis is linked to the EL7342 EtherCAT remote terminal and is also linked to a variable in the PLC code called GVL.Axis_1. This variable is defined in the GVL that is described in Section 7.3. Once the link is made, any change in variables mapped to the EL7342 motor terminal, or the PLC variable, will affect the axes. For example, using a velocity move on the PLC variable causes the axes to move, in turn causing the motor linked to the hardware to move.

It is now possible to control the actual DC motor from the PLC code which is linked to the transformed Simulink model. The following section describes the implementation of the full-state feedback controller and analyses the effects of the network induced delays. The simulation results and implementation results of the actual DC motor are compared and discussed.



Link To I/O... Term 2 (EL7342) # 'Ch 1 - DC motor int'

Link To PLC... GVL.Axis_1 (Control Instance)

Axis Type: DC Drive (MDP 733)

Unit: mm

Display (Only)
Position: ☐ µm ☐ Modulo
Velocity: ☐ mm/min

Result
Position: mm
Velocity: mm/s
Acceleration: mm/s2
Jerk: mm/s3

Axis Cycle Time / Access Divider
Divider: 1
Cycle Time (ms): 1.000
Modulo: 0

[✓ Unused] All

Figure 0.20: Linking an axis to peripherals

7.5 Implementation of closed loop system with controller

This section provides the implementation of the developed control systems on the hardware components that are described in sections 7.2, 7.3 and 7.4. A simplified block diagram of the control system is shown in Figure 7.21 showing the information sent between the different hardware components. The position set point is sent from the CodeSYS PLC software to the Simulink transformed model to initiate a change in position of the DC motor. The model returns the velocity, acceleration, deceleration, and jerk values that are needed for the actual DC motor to achieve the same response times as the simulation. These values are scaled and sent to the NC axis control where the data is packaged and sent over EtherCAT to the EK1100 motor terminal controller. The motor terminal controller outputs an analog voltage value to the DC motor which causes the motor to rotate. The encoder counts the rotations in pulses and feeds this information back into the EK1100. The EK1100 converts the pulses into EtherCAT data packages and sends them back to the C6015 IPC. Lastly, the IPC converts the EtherCAT data packages into a scaled position feedback value which is used by the PLC code to determine how far the motor has travelled. Once the motor reaches the set point, the system is at standstill until another positional set point command is sent to the Simulink transformed model.

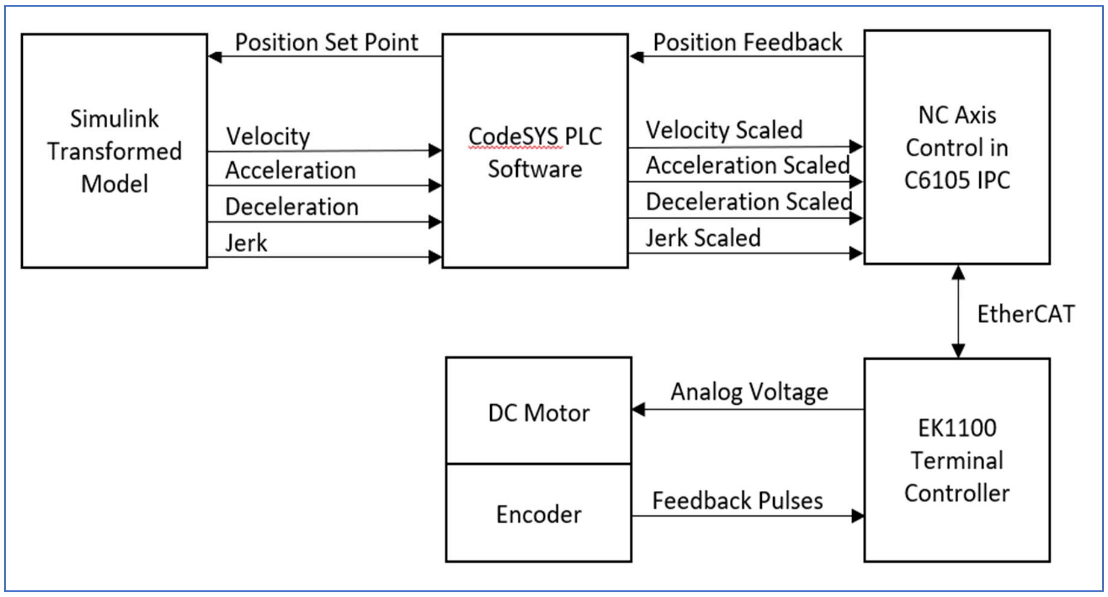


Figure 0.21: **Block diagram showing the information transmitted between the components of the DC motor control system**

The following subsections implements and analyses the DC motor control system with and without time delay compensation. The section is concluded with subsection 7.5.3 which compares the simulated results from Chapter 5 with the implementation results from this chapter.

7.5.1 Analysis of the control system with delays

An EtherCAT fieldbus connection is used to connect the controller and the plant in this research project. It is necessary to compare and analyse the effects of this type of topology to make sure that the system can handle the effects of network delays. In some cases, these delays can cause instability and a decrease in system performance. If the delays cause issues in the system, steps need to be taken to reduce the effects, such as slowing the controller down, decreasing traffic on the network, or considering using a different type of network and hardware devices.

Figure 7.22 shows the response of the actual DC motor to a position set point change of 10 mm. The blue line represents the scaled set point output from the Simulink model. The green line represents the actual position set point sent from the NC terminal controller to the DC motor. The pink line represents the actual position of the DC motor which is fed back by the encoder. The response is recorded using Scopeviewer within the TwinCAT 3 programming environment in real-time.

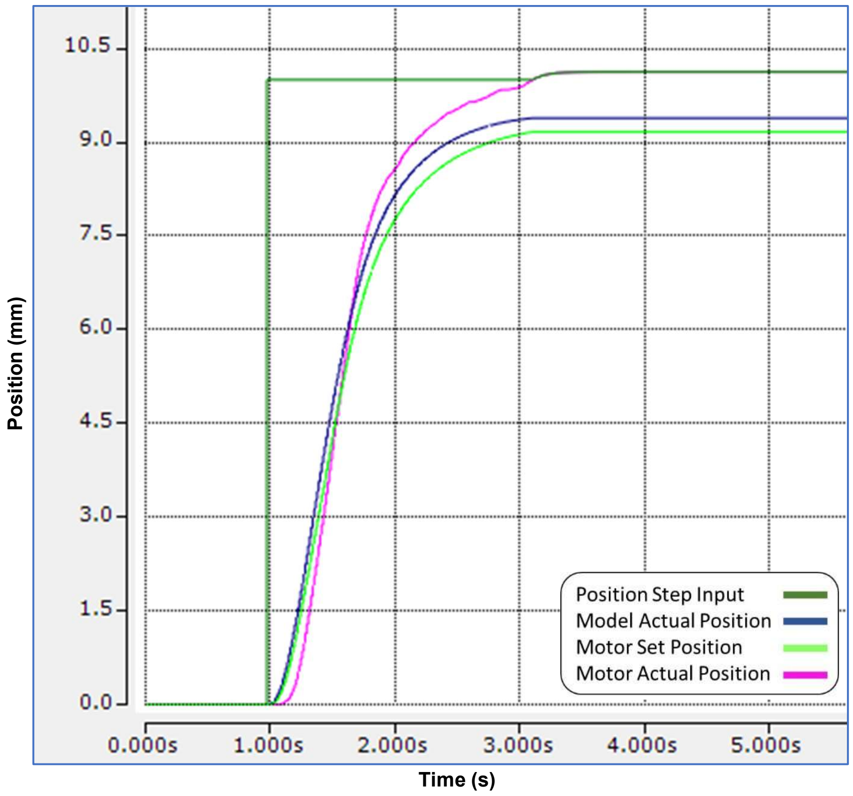


Figure 0.22: Step response of actual DC motor control system

Figure 7.23 shows a panned image of the DC motor step response shown in Figure 7.22. A delay of 75 ms is shown between the Simulink positional command and the actual position of the DC motor. This delay is caused by the delay in the feedback from the encoder to the EK1100 motor terminal controller as well as for the transmission of the data back to the IPS over the EtherCAT network. This delay can be considered as the sensor-to-controller delay.

There is also a 5 ms delay between the Simulink model output set point command and the NC command sent to the DC motor. This delay is considered as the controller-to-actuator delay and is caused by cycle times in the IPC as well as the rate of transmission of the commands over EtherCAT from the IPC to the EK1100. The combined 75 ms and 5 ms delays equate to a combined 80 ms delay in the overall system.

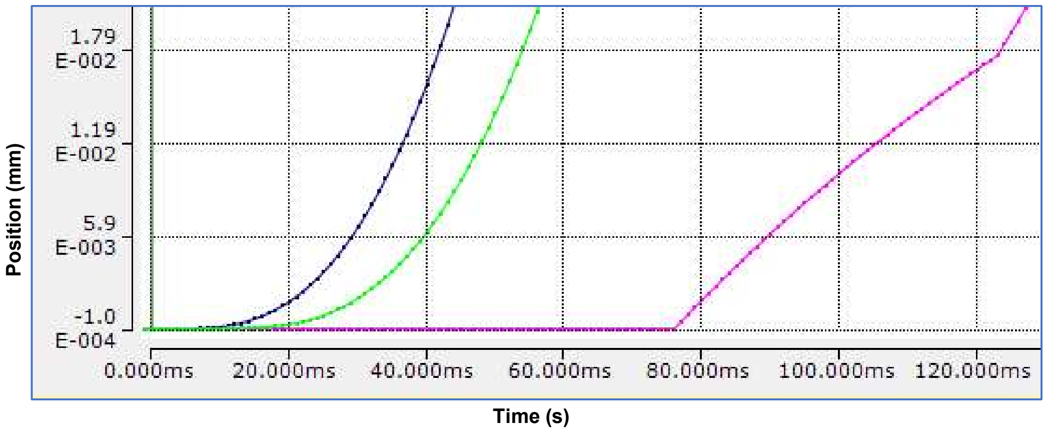


Figure 0.23: **Panned in X and Y axis of the actual DC motor step response**

The effects of these delays can be seen in Figure 7.22 where the actual motor position tries to catch up to the set point resulting in an overshoot. The controller then decreases the position set point to compensate for the fast rise time of the position of the actual DC motor. The delay is small enough to not cause the controller to become unstable, but the initial simulated response is not achieved and therefore the controller does not perform with the desired characteristics as designed. Figure 7.24 shows the simulated response of the DC motor with an 80 ms delay. As shown, the response is like the implementation results where the actual motor position has an increased rise time and overlaps the desired response.

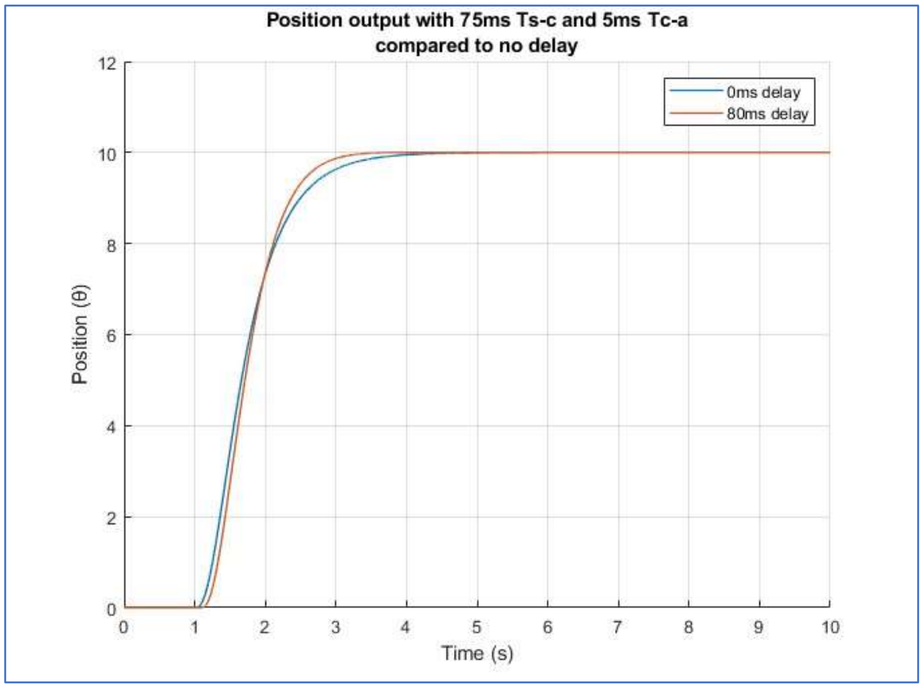


Figure 0.24: **Simulink step response with 80 ms network delay**

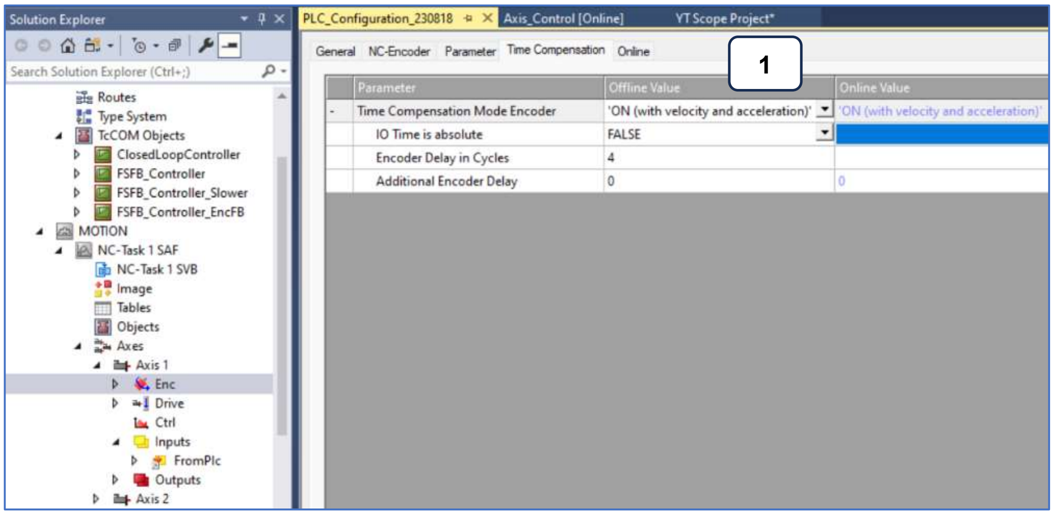
It is not possible to increase or decrease the delay shown in Figure 7.22 as this is an EtherCAT-based system, and it is not possible to influence the data transfer on the network as there is a direct connection between the IPC and the remote EtherCAT coupler. If however, an Ethernet system is used, an additional Ethernet port is available to allow for an external source such as PC to flood the system with data to increase the delay times by using up the available bandwidth.

This subsection presents the network-induced delays caused by the EtherCAT fieldbus in the DC motor control system. The effects are compared with the Simulink response with added sensor-to-controller and controller-to-actuator delays. The next subsection describes how to minimize the network delays in the system by using features provided within the TwinCAT 3 programming environment.

7.5.2 Analysis of the control system with time delay compensation

The TwinCAT 3 software has features that can be used to decrease the delays shown in the results discussed in subsection 7.5.1. These delays that affect the system1's response are referred to as dead time in the TwinCAT programming environment. The software measures the system's response time by monitoring a control input and the corresponding feedback response. Specific algorithms are used to predict the desired output using the system's characteristics and adjusts the control signals to account for the dead time in real-time. This improves the accuracy and stability of the system and is crucial in systems that require high precision and synchronization.

There are two methods to add time compensation to the system to decrease the effects of delays caused by transmission rates in the network. The encoder method uses time compensation to compensate for the conversion of all the feedback data from the terminal controller to the PLC, such as the actual position of the DC motor. The drive method uses time compensation to compensate for the conversion of command signals sent from the PLC to the terminal controller, such as the set position, velocity, acceleration, deceleration, and jerk. In this project, both time and drive compensation are used to predict the velocity and acceleration delayed responses. Figure 7.25 shows where to activate the time compensation mode for the encoder (1). The same method can be used to activate the setting for the drive.


Solution Explorer

Search Solution Explorer (Ctrl+;)

Routes
Type System
TcCOM Objects
ClosedLoopController
FSFB_Controller
FSFB_Controller_Slower
FSFB_Controller_EncFB
MOTION
NC-Task 1 SAF
NC-Task 1 SVB
Image
Tables
Objects
Axes
Axis 1
Enc
Drive
Ctrl
Inputs
FromPlc
Outputs
Axis 2

PLC_Configuration_230818 | Axis_Control [Online] | YT Scope Project*

General | NC-Encoder | Parameter | Time Compensation | Online

Parameter	Offline Value	Online Value
Time Compensation Mode Encoder	'ON (with velocity and acceleration)'	'ON (with velocity and acceleration)'
IO Time is absolute	FALSE	
Encoder Delay in Cycles	4	
Additional Encoder Delay	0	0

Figure 0.25: Activating time compensation in TwinCAT 3

Figure 7.26 shows a panned screenshot of the step response of the DC motor control system with time compensation for the drive and encoder. As in Figure 7.24, the blue line represents the scaled set point output from the Simulink model, the green line represents the actual position set point sent from the NC terminal controller to the DC motor, and the pink line represents the actual position of the DC motor which is fed back by the encoder. Unlike in Figure 7.2.4, the actual motor position is not delayed as the time compensation feature has predicted what the motors response would have been if there were no delays in the network. This prevents the motor from trying to run at a faster rate on startup as the feedback is where it should be in normal conditions without delays.

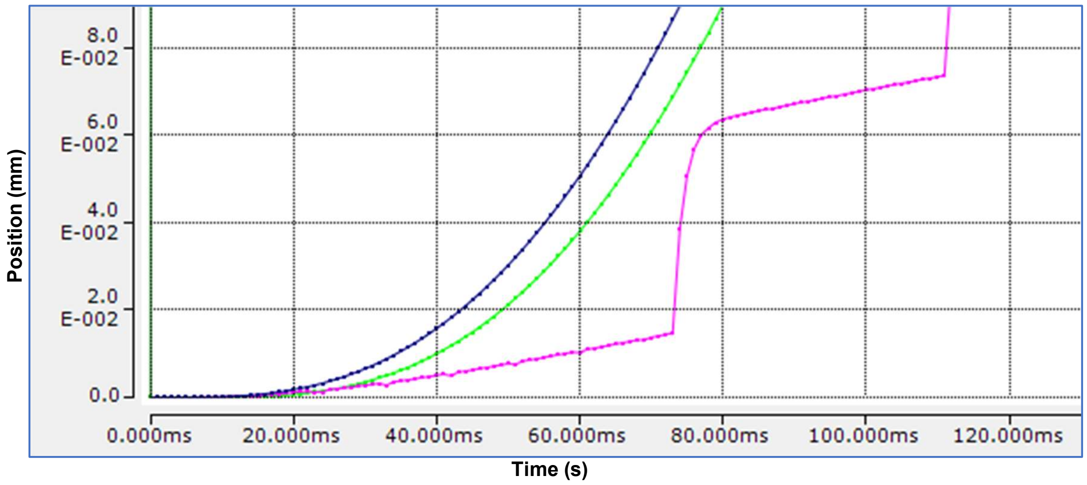


Figure 0.26: Step response with time compensation

Figure 7.27 shows the full graph of the step response with time compensation for an input position set point change of 10 mm. The actual feedback does not overlap the position command output as seen when there are delays present. This prevents the system from overshooting the position set point and allows the response to be the same as the system with no delays. The addition of time compensation has prevented an unwanted response and reduced the effects of delays caused by the EtherCAT network and the signal processing between the controller and the motor terminal card.

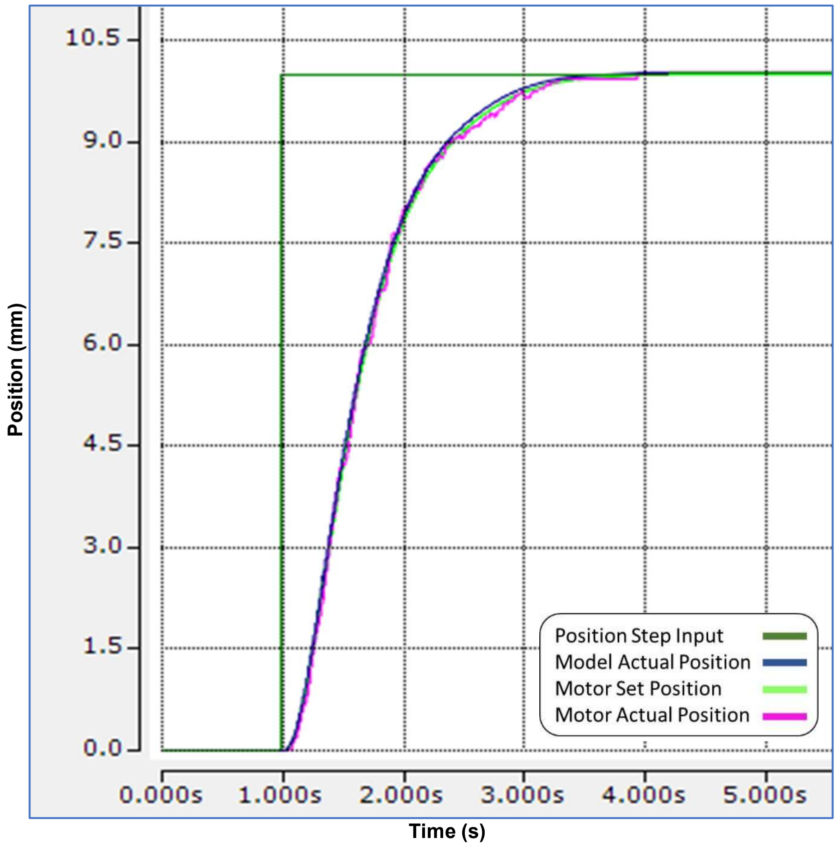


Figure 0.27: Step response with time compensation

A ripple effect is seen for the feedback response of the DC motor position. This effect is caused by running the motor at a low speed to get to the position set point at the rate that the Simulink model sets. A much higher current for the motor to overcome initial torque is needed to turn the gearbox but the EK1100 is only rated at 3.5 A and therefore a higher current output is not possible. To overcome oscillations due to mechanical and electrical constraints, a high-efficiency DC motor can be used as running at low speeds is possible. An encoder with a higher resolution can also be considered to allow for the motor to run at low speeds as sufficient feedback is returned.

7.5.3 Positional Set Point Limitations

A positional set point change of 10 mm has been used in this work for testing and simulation purposes. Depending on the application, a higher or lower set point change would be required. For this work, the DC motors are used to control the azimuth and altitude movements of a radio antenna dish. The dish would generally only need to turn at a 270-degree angle for azimuth and a 90-degree angle for altitude. If a full rotation of the motor is 18.41 mm then the motor would only need to move within a 13.81 mm range for azimuth and a 4.6 mm range for altitude. Figure 7.28 shows the DC motor system response to a set point change of 5 mm. Due to factors mentioned in the previous section such as a low efficiency motor, low pulse encoder, and low current output from the controller, the motor's feedback ripple effect can clearly be seen when trying to rotate at lower speeds. Since the range of the altitude movement is less than 5 mm, the DC motor oscillates for any change in position. To overcome this problem, a high efficiency DC motor should be used with a higher resolution encoder.

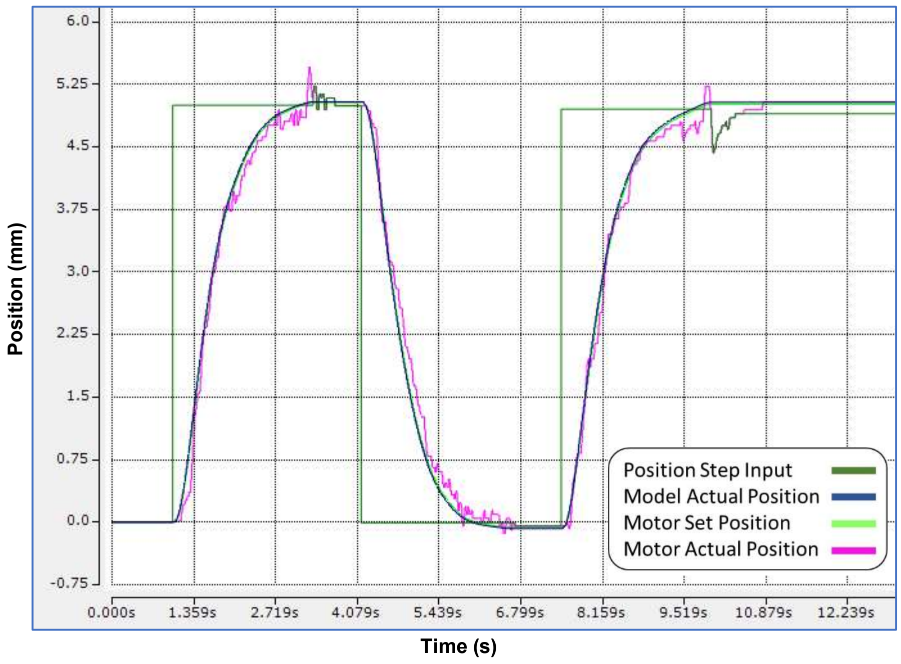


Figure 0.28: **System response to position set point changes of 5mm**

Figure 7.29 shows the response of the control system to a position set point that is double the rotation of the motor. The results show that the DC motor still reaches the position while remaining stable with no oscillations. Once the system is stable at double the necessary position set point, an additional 2:1 gearbox can be used for the altitude movement. This allows the motor to run at twice the speed, minimizing the ripple effect while still allowing the motor to move at set points lower than 5 mm. Another advantage is cost savings as there is no need for a better motor or encoder any longer.

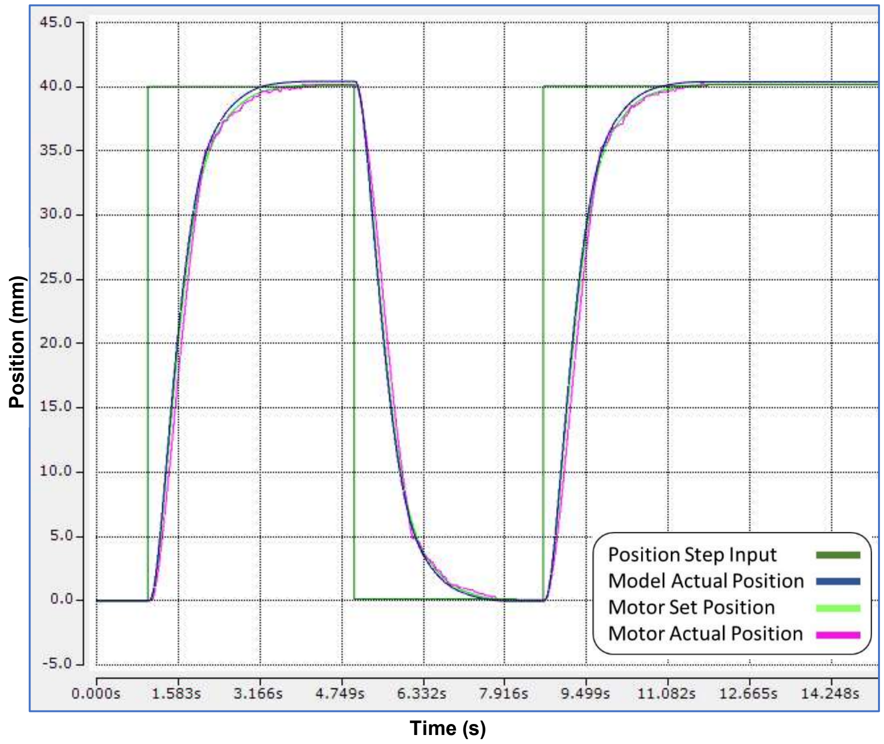


Figure 0.29: System response to position set point changes of 40mm

Figure 7.30 shows the response of the control system to a change of 18.41 mm, which is a full 360 degree turn of the motor shaft. The system is stable and the set point reaches steady state in the forward and reverse direction. Initially, the actual motor position and the Simulink model's positions are in sync as the motor does not need to turn at a low velocity. Closer to the set point the system starts decelerating and causing minor ripples as described in Section 7.5.2. The results show that a full 360-degree range of the motor is possible, and therefore the 270-degree range for azimuth, and 90-degree range for altitude is possible.

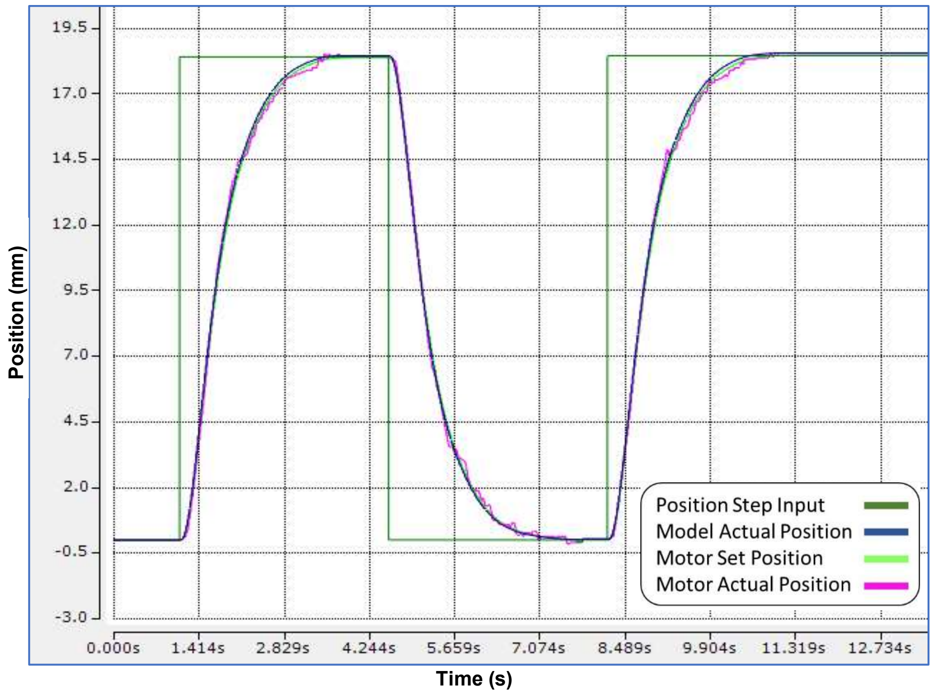


Figure 0.30: **System response to position set point changes of 18.41mm**

A final test to show the positional change limit is shown in Figure 7.31 where the set point of 60 mm is input to the control system. As shown, the motor is not able to accelerate fast enough and therefore the controller overshoots the set point to compensate for the position feedback. The DC motor eventually reaches a steady state, but the overshoot would cause the radio antenna dish to move past the required position and then move back to the set point. This is not ideal for the real-life implementation. The motor is limited to a maximum angular position set point change of 720 degrees and therefore cannot reach a set point that is higher without leading to instability. To overcome this issue, a limit on the allowed positional change should be set.

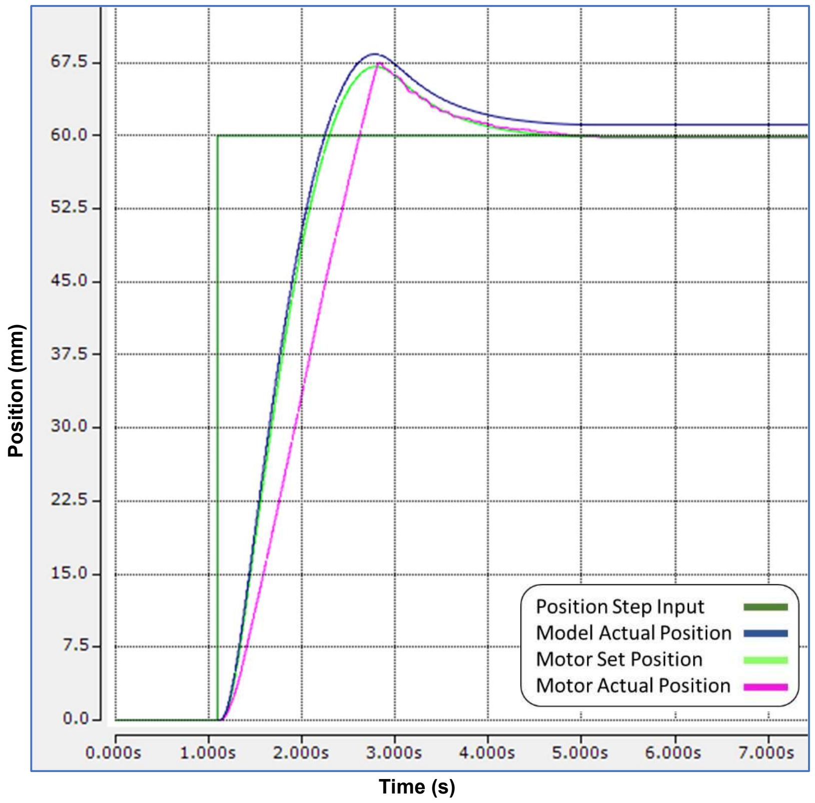


Figure 0.31: **System response to position set point change of 60mm**

This subsection discussed the results of the real-time implementation of the DC motor control system. The next subsection discusses the results and findings of the work completed in this thesis.

7.5 Discussions

The main objective of the hardware-in-the-loop implemnentaion is to use the Simulink model of the controller to control an actual DC motor via the Beckhoff IPC. This objective aims to prove that portability of software components from one vendor's programming environment to another is possible. The PLC code developed using CodeSYS programming environment and PLCopen function blocks also allows for reusability of the code with different vendor's hardware. The factors of portability and reusability show that the transformation is successful and that aspects of the IEC 61499 standard are possible.

The DC motor response to a position change is exactly like the simulation of the modelled motor in Simulink. This is possible by using Beckhoff's time compensastion feature to overcome the challenge of networked induced delays. Without this feature, the real-time system with 80 ms network induced delays reacted the same as the Simulink model with delays in the loop. This proves that the modelling of the system is correct as both the simulated and real-time systems respond the same under certain conditions.

The system’s limitations and robustness to position set point changes are tested. A very small change in position is sent to the motor, but due to the motor not being very efficient and the current limitations of the motor terminal card, the motor tends to oscillate at low speeds. A test is done to run the motor for two revolutions in one set point change and the results show that the control system is still stable. Therefore if a smaller positional movement is required for the DC motor, as is needed for the altitude movements, then the motor can be run at twice the speed using a 2:1 gearbox.

Other factors of the IEC 61499 standard can be looked at for further improvements of the control system. The use of IEC 61499 standard function blocks will allow the PLC code to be interoperable between different software vendors. The TwinCAT 3 programming environment does not support IEC 61499 standard function blocks or event-based execution methods of PLC code at this time, and therefore this option was not explored. Other software packages such as nxtStudio and 4DIAC could be used to create a similar system to compare with the current design.

The work done in this thesis creates a basis for future work and development projects that can contribute to the ongoing growth of the IEC 61499 standard in the industry. The test rig can be used to test portability, reusability, configurability, and interoperability between different software vendors by adding other IEC 61499 compliant hardware to the system. Distributed the plant and controller over different hardware platforms can also be explored to prove the benefits of using the IEC 61499 standard.

This section discussed the objectives, challenges, outcomes, and contributions of the work done in this thesis. The next section concludes the chapter.

7.6 Conclusion

This chapter presented the real-time implementation of the DC motor control system. All the hardware and software components used to control the DC motor using the built test rig are described. The hardware includes the C6015 IPC, EK1100 EtherCAT coupler, EL7342 motor terminal controller, DC motor, Omron encoder, and the electrical components used to connect everything. The software includes the CodeSYS PLC code, Beckhoff's motion interface to control the axis, the PLCopen blocks to command the axis, and the transformed Simulink block of the modelled DC motor and controller. An electrical diagram and photo of the complete test rig are also shown.

The control system is tested by giving the DC motor a new set point position. The results are recorded using Scopeviewer, and compared with simulation results from MATLAB/Simulink. The system is first tested with no delay compensation techniques. A delay of 80 ms is present due to the EtherCAT fieldbus connection between the IPC and the motor terminal controller. This delay is shown to cause the motor to run faster than the intended set point, leading to an overshoot. The MATLAB/Simulink response shows the same results when influenced by the same delay.

Time compensation is used in the Beckhoff programming environment to predict the response of the motor when no feedback has been received. This allows the system to reduce the delays to zero by simulating a feedback from the motor that has not yet been received by the IPC through the EtherCAT network. This method of adding time compensation reduces the overshoot and allows the DC motor to respond exactly like the simulation results that do not have network delays included in the model. Lastly, the limitations of the system are tested regarding changes in positional set point.

The results in this chapter show that the system is stable when given a change in position and time compensation is used to negate the network delays. The next chapter concludes the thesis.

CHAPTER 8

CONCLUSION AND FUTURE DIRECTION OF RESEARCH

8.1 Introduction

Distributed control systems are inevitable in the industry for the foreseeable future. With smart devices adding huge amounts of data traffic to networked systems, these networks must be considered when designing control systems. These smart devices should also be easy to import into any vendors' software packages to allow for quick integration and simple communication. Standards such as the IEC 61499 standard, PLCopen function blocks, and CodeSYS programming environment, are all good examples of a future of software in automation that is portable, interoperable, and reusable. Using these standards to create control systems that are distributed and coupled by communication networks instead of hardwired systems provided the motivation for this study. The purpose of this research is to design and implement a control system for a radio antenna's azimuth and altitude positional movements by using modern control design methods and utilizing the portability and reusability aspects of the IEC 61499 standard.

This chapter presents the deliverables and the conclusion to the thesis.. Section 8.2 provides the deliverables and the objectives that were achieved. Section 8.3 shows table of developed software programmes to achieve the results of the completed work. Section 8.4 presents possible applications of the research work in the industrial and academic fields. Section 8.5 presents possible future work directions for the content of this research. Section 8.6 details the publications emanating from this research. Section 8.7 concludes this work.

8.2 Thesis Deliverables

This section describes the thesis deliverables that have been achieved.

8.2.1 Literature Review

Two literature reviews encompassing the IEC 61499 standard and networked control systems is completed.

The literature review on the IEC 61499 standard spans over 19 years, from 2004 to 2023. The review covers articles written on the first edition of the standard, as well as the the second edition which was published in 2012. The first review covers distributed controller design based on the standard, the different types of execution methods of standard function blocks, the portability capabilities of software tools which are IEC 61499 standard compliant, and modelling and verification of systems that use the standard. The changes in the second edition that saw an increase of use of the standard in the industry, and comparisons between different articles regarding the standard, are also focused on in the review.

Due to the nature of the distributed control systems being networked, a second literature review is conducted in the area of networked control systems. The review provides an overview of networked control systems, describes the issue of delays induced by using networks, and compares articles from multiple authors on how these delays were dealt with and which methods were more effective.

8.2.2 Mathematical Modelling of the Plant and Controller

In this thesis, the model of the closed-loop DC motor is derived using Kirchhoff and Newton's laws of physics. The balance equations of the electrical and mechanical parts are combined to describe an electro-mechanical system. The load of the dish is also added to the system by means of a gearbox model. A simplified transfer function showing the relationship between angular position and armature voltage is described.

The transfer function is converted to state space representation to reduce the complexity when trying to model the system. Converting to state space also allows for the addition of a state feedback controller which is based on modern control theory. The state feedback controller is designed for a faster response time of the DC motor to a change in angular position. An integrator is added before the controller to reduce the steady state error of the response. The complete transfer function of the DC motor with statefeedback and integral control is modelled and simulated in MATLAB/Simulink.

8.2.3 Simulation of the Model of the Plant and Controller

The open-loop model of the DC Motor without a controller is simulated and verified using MATLAB/Simulink software tools. The step responses of the model shows that with no feedback, the motor continues to turn with time. Because of this, a feedback loop is added for the motor to reach steady state.

The closed-loop system of the DC motor without a controller, with a state feedback controller, and with integral control added, are all simulated in MATLAB/Simulink. The step responses of each system is compared and validated. The results show that the system with statefeedback and integral control had the best transient response compared to the other systems. The control system has zero overshoot and reaches steady state within 3 seconds. These simulated results are compared to the real-time implementation results later.

8.2.4 Simulation and Analysis on the Effects of Networked Induced Delays

The effects of networked induced delays are analyzed using the developed controller. Two network induced delays are looked at: sensor-to-controller and controller-to-actuator delays. Different magnitudes of time delays are tested using a delay function block in Simulink. The effects of each delay can clearly be seen negatively effecting the performance and stability of the control system. The greater the delay, the more unstable the system becomes when trying to reach an angular position set point.

Sensor-to-controller delays of 100ms, 400ms, 800ms, 1200ms are simulated and the results are graphed using MATLAB scripts. The results for the controller-to-actuator delays of 50ms, 100ms 125ms, and 150ms, are also tabulated and graphed.

8.2.5 Transformation of the Simulink model to TwinCAT 3

The Simulink model is successfully transformed to TwinCAT 3 using the Beckhoff TE1400 target. The model is converted to a TcCom object that is used within the CodeSYS PLC software environment to interact with the developed PLC code and PLCopen axis function blocks. A MATLAB script is developed to allow for an easy transformation by just compiling and running the code. Descriptions of the software required and installation procedures is also presented in this thesis.

8.2.6 Real-time Hardware-in-the-loop Test-bed Implementation of the Developed Controller

A complete test bench is built for the real-time implementation of the simulated controller and plant. The test rig components allow for the interaction between the transformed Simulink model and the actual DC motor. A Beckhoff IPC hosts the run-time application and serves as the EtherCAT hub for the system. A Beckhoff EK1100 allows for the conversion of EtherCAT signals sent from the IPC to the EL7342 motor terminal card. The motor terminal card controls the voltage supplied to the motor, and returns the encoder feedback through the EtherCAT back to the IPC for position monitoring.

The test rig has a 24V DC power supply to power the Beckhoff components, and a 12V DC power supply for the motor supply. Two push buttons are added to the test rig to allow for an easier interface to change the angular position set point. The motor and encoder are connected via a mechanical 1:1 gearbox. The test rig requires 230V AC supply in order to power all the components.

8.2.7 Comparisons between the Simulated and Implemented Results

The first test on the real-time system is to see the system's response to a change in position set point. The first results showed that there is a delay due to the EtherCAT network being used. The response of the system is proven to be the exact same as the simulation results which showed that a delay causes some overshoot. Beckhoff's Scopeviewer feature is used to view the real-time response of the DC motor control system.

Beckhoff's delay time compensation feature is activated in the system and the same test is carried out. Delay time compensation predicts the feedback of the motor and estimates an initial response which counters the starting delay of the actual position. This feature prevents the actual position from overshooting the position of the simulation model. The real-time system and the model simulation results are the same for a change in angular position.

The control system's limitations are also tested such as maximum change in set point allowed, minimum velocity of the DC motor, and the system response to consecutive changes. The results show that the system is stable for a full 720 degree set point change. This range covers the initial requirements for the 90 degee and 270 degree turns of the azimuth and altitude positional movements respectively.

8.3 Software Development

Table 0.1: **Summary of the software programmes developed in this research**

Number	Filename	Application Description	Appendix
1	App_A_41.m	DC motor parameters	Appendix A4.1
2	App_A_42.m	Open loop system in state space form	Appendix A4.2
3	App_A_51.m	Full State Feedback Controller	Appendix A5.1
4	App_A_52.m	Full State Feedback Control with Integral Gain	Appendix A5.2
5	App_A_53.m	Case study 5.3.4 graph outputs for step responses	Appendix A5.3
6	App_A_54.m	Observer gain	Appendix A5.4
7	App_A_551.m	Sensor to controller delay graph outputs	Appendix A5.5.1
8	App_A_552.m	Controller to actuator delay graph output	Appendix A5.5.2
9	App_A_61.m	TwinCAT transformation commands	Appendix A6.1
10	App_A_71.m	Position output of control system with and without delay	Appendix A7.1

8.4 Application of the Developed Algorithms and Methods

The control system developed for the DC motors can be used for both academic and industrial applications as indicated.

8.4.1 Industrial Applications

- Using the IEC 61499 standard for a DC motor control system that operates an antenna dish azimuth and altitude positional movements of a radio antenna dish
- Any other application that requires movement within a 360 degree rotation
- Using the IEC 61499 standard in applications that require software that has increased modularity, scalability, and interoperability to allow for faster development times.
- Applications that require fast response times as the IEC 61499 standard application function blocks allow for distributed control which has faster response times than central controllers.

8.4.2 Academic Applications

This research can be used to train undergraduate students or retrain researchers at the postgraduate level.

- How to mathematically model DC motors
- How to convert from transfer function to state space equations
- How to apply a state feedback controller with integral gain
- The constructed test rig can be used for future research work by postgraduate students.

8.5 Future Work

- Methods could be applied to reduce the network induced delays in the model therefore not using time compensation in TwinCAT 3.
- To distribute the plant and controller over two different PLC vendors that are both IEC 61499 Standard compliant.
- To change the manual set point push buttons to a automatic tracking system that can track celestial objects in the sky
- To build the actual radio antenna with two DC motors for azimuth and altitude movements to continue testing.

8.6 Publications Related to the Thesis

- Love K., Kriger C., Tshemese-Mvandaba N. (2023) ‘Design and Simulation of a Full State Feedback Controller for a DC Motor’, International Journal of Electrical and Electronic Engineering & Telecommunications (IJEEET) (submitted for publication).
- Love K., Kriger C., Tshemese-Mvandaba N. (2023) ‘Hardware-in-the-loop real-time simulation and implementation for a DC motor’, Submitted to the Southern African Universities Power Engineering Conference (SAUPEC 2024).

8.7 Conclusion

This chapter described all the deliverables that were proposed and achieved. The applications where this work can be implemented in industry and academia is described. A list of the developed software is provided. Possible future work that could arise from this project are described. Submitted publications emanating from this research are also listed.

REFERENCES

Abrishambaf, R. Bal, M. and Vyatkin, V. (2017) ‘Distributed home automation system based on IEC61499 function blocks and wireless sensor networks, 2017 IEEE International Conference on Industrial Technology (ICIT), pp. 1354-1359. DOI: 10.1109/ICIT.2017.7915561.

Administrator (2019) Different Types of Antennas & Characteristics of Antenna. Available at https://www.electronicshub.org/types-of-antennas/ (Accessed: 18 April 2023)

Aloo, L. (2016) ‘DC Servomotor-based Antenna Positioning Control System using Hybrid PID-LQR Controller’, European International Journal of Science and Technology, 5(2), pp. 17-31.

Antoneko, D. (2022) Centralized vs Decentralized vs Distributed Networking Explained. Available at: https://www.businesstechweekly.com/operational-efficiency/computer-networking/centralized-vs-decentralized-vs-distributed-networking-explained/ (Accessed: 3 August 2022)

Asl, E. M. Hashemzadeh, F. Baradarannia, M. Bagheri, P. (2021) ‘Observer-based Controller Design for a Class of Networked Control Systems with Transmission Delays and Packet Loss’, 2021 7th International Conference on Control, Instrumentation and Automation (ICCIA), pp. 1-6, DOI: 10.1109/ICCIA52082.2021.9403532.

Asl, E. M. Hashemzadeh, F. Baradarannia, M. Bagheri, P. (2022) ‘The Effect of Observer Position on Networked Control Systems with Random Transmission Delays and Packet Dropouts’, 2022 8th International Conference on Control, Instrumentation and Automation (ICCIA), pp. 1-6, DOI: 10.1109/ICCIA54998.2022.9737196

Balanis, C. A. (1992) ‘Antenna Theory: A Review’, Proceedings of the IEEE, 80(1), pp. 7-23. doi: 10.1109/5.119564.

Black, G. and Vyatkin, V. (2010) ‘Intelligent Component-Based Automation of Baggage Handling Systems With IEC 61499’, IEEE Transactions on Automation Science and Engineering, 7(2), pp. 337–351. DOI: 10.1109/TASE.2008.2007216.

Brennan, R.W., Lyu, G. (2019). ‘IEC 61499 and the Promise of Holonic Systems. In:, et al. Industrial Applications of Holonic and Multi-Agent Systems,‘ HoloMAS 2019. Lecture Notes in Computer Science, 11710(0). DOI.org/10.1007/978-3-030-27878-6_1.

Britannica, The Editors of Encyclopaedia (2023) Lorentz Force. Available at: https://www.britannica.com/science/Lorentz-force (Accessed: 18 April 2023)

Bzdigian, A. (2022) 10. Output devices. Available at: https://fabacademy.org/2022/labs/dilijan/students/ashod-bzdigian/Assignments/week11/ (Accessed: 18 April 2023)

Čengić, G. and Åkesson, K. (2008) ‘A Control Software Development Method Using IEC 61499 Function Blocks, Simulation and Formal Verification’, IFAC Proceedings Volumes 41(2), pp. 22-27. DOI.org/10.3182/20080706-5-KR-1001.00003.

Čengić, G. Ljungkrantz, O. and Akesson, K. (2006) ‘Formal Modeling of Function Block Applications Running in IEC 61499 Execution Runtime’, 2006 IEEE Conference on Emerging Technologies and Factory Automation, pp. 1269-1276. DOI: 10.1109/ETFA.2006.355187.

Chen F. and Zhou, X. (2023) ‘Design of Predictive Controller for Networked Control Systems’, 2023 IEEE 6th Information Technology, Networking, Electronics and Automation Control Conference (ITNEC), pp. 1544-1547, DOI: 10.1109/ITENC56291.2023.10082208.

Christensen, J. H. (2022) A Standard for Software Reuse in Embedded, Distributed Control Systems. Available at: https://holobloc.com/papers/iec61499/overview.htm (Accessed: 15 July 2022)

Christensen, J. Strasser, T. Valentini, A. Vyatkin, V. and. Zoitl, A. (2012) ‘The IEC 61499 function block standard: Overview of the second edition’, ISA Autom. Week, 6, pp. 6–7.

Colla, M. Brusaferri, A. and Carpanzano, E. (2006) ‘Applying the IEC-61499 Model to the Shoe Manufacturing Sector’, 2006 IEEE Conference on Emerging Technologies and Factory Automation, pp. 1301-1308. DOI: 10.1109/ETFA.2006.355422.

Cruz, E. M. Carrillo, L. R. Salazar, L. A. C. (2023) ‘Structuring Cyber-Physical Systems for Distributed Control with IEC 61499’, IEEE Latin America Transactions, 21(2), pp. 251-259. DOI: 10.1109/TLA.2023.10015217.

Cruz, E. M. Carrillo, L. R. G. Patil, S. Ceron, P. Ceron, J. F. (2022) ‘Validating effect of Refactoring of IEC 61499 Function Block in Distributed Control Systems’, 2022 IEEE International Conference on Automation/XXV Congress of the Chilean Association of Automatic Control (ICA-ACCA). DOI: 10:1109/ICA-ACCA56767.2022.10005950.

Dai, W. and Vyatkin, V. (2012) ‘Redesign Distributed PLC Control Systems Using IEC 61499 Function Blocks’, in IEEE Transactions on Automation Science and Engineering, 9(2), pp. 390-401. DOI: 10.1109/TASE.2012.2188794.

Dai, W. Vyatkin, V. Christensen, J. H. and Dubinin, V. N. (2015) ‘Bridging Service-Oriented Architecture and IEC 61499 for Flexibility and Interoperability, IEEE Transactions on Industrial Informatics, 11(3), pp. 771-781. DOI: 10.1109/TII.2015.2423495.

Dai, W. Dubinin, V. N. Christensen, J. H. Vyatkin, V. and Guan, X. (2017) ‘Toward Self-Manageable and Adaptive Industrial Cyber-Physical Systems With Knowledge-Driven Autonomic Service Management’, in IEEE Transactions on Industrial Informatics, vol. 13(2), pp. 725-736. DOI: 10.1109/TII.2016.2595401.

Dai, W. Zhang, Yi. Zhang, Yu. Kang, J. Huang, D. (2023) ‘Automatic Information Model Generation for Industrial Edge Applications Based on IEC 61499 and OPC UA’, IEEE Transactions on Industrial Informatics, 19(4), pp. 6093-6104. DOI: 10.1109/TII.2022.3191365.

Dhillon, M. Kumar, S. (2017) ‘Performance Analysis and Comparison of various Rectenna based RF Energy Harvesting System’, Introduction to Mobile Communications.

Du, Z. Hu, S. and Li, J. (2013) ‘Modeling and stabilization for singular networked cascade control systems with state delay’, Proceedings of the 32nd Chinese Control Conference, pp. 6704-6709.

Eze, P. C. Ugoh, A. C. (2021) ‘Positioning Control of DC Servomotor-Based Antenna Using PID Tuned Compensator’, Journal of Engineering Sciences, 8(1), pp. E9-E16. doi: 10.21272/jes.2021.8(1).e2.

Faccio, D. Clerici, M. Tambuchi, D. (2006) ‘Revisiting the 1888 Hertz Experiment’, American Journal of Physics, 74, pp. 992-994. doi: 10.1119/1.2238886.

Glatz, B. Cleary, F. Horauer, M. Schuster, H. and Balog, P. (2016) ‘Complementing testing of IEC61499 function blocks with model-checking’, 2016 12th IEEE/ASME International Conference on

Mechatronic and Embedded Systems and Applications (MESA), pp. 1-7. DOI: 10.1109/MESA.2016.7587151.

Goonhilly (2022) 60 years ago today, on 11th July 1962, Goonhilly made world history. Available at https://www.goonhilly.org/60th-anniversary-of-a-world-first (Accessed: 20 April 2023).

Gupta, R. A. and Chow, M. (2010) ‘Networked control system: overview and research trends’, IEEE Transactions on Industrial Electronics, 57(7), pp. 2527-2535. DOI: 10.1109/TIE.2009.2035462.

Hirsch, M. Gerber, C. Hanisch, H. and Vyatkin, V. (2007) ‘Design and Implementation of Heterogeneous Distributed Controllers According to the IEC 61499 Standard - A Case Study’, 2007 5th IEEE International Conference on Industrial Informatics, pp. 829-834. DOI: 10.1109/INDIN.2007.4384881.

Hussain, T. and Frey, G. (2007) ‘Deployment of IEC 61499 compliant distributed control applications’, 2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007), pp. 502-505. DOI: 10.1109/EFTA.2007.4416810.

Iswanto. Raharja, N. M. Ma’arif, A. Ramadhan, Y. Rosyady, P. A. (2021) ‘Pole Placement Based State Feedback for DC Motor Position Control’, Annual Conference on Science and Technology Research (ACOSTER) 2020. doi: 10.1088/1742-6596/1783/1/012057.

Kolla, S. R. and Mainoo, J. (2012) ‘Effect of network-induced delays in control systems: application to dc motor control’, 2012 International Conference on Advances in Power Conversion and Energy Technologies (APCET), pp. 1-6. DOI: 10.1109/APCET.2012.6302075.

Koziorek, J. (2004) ‘Design of Distributed Control Systems Based on New International Standards’, IFAC Proceedings Volumes, 37(19), pp. 313-318. DOI: 10.1016/S1474-6670(17)30703-6.

Lapp, H. Gerber, C. and Hanisch, H. (2010) ‘Improving verification and reliability of distributed control systems design according to IEC 61499’, 2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010), pp. 1-8. DOI: 10.1109/ETFA.2010.5641247.

Lu, H. Guo, M. Hu, Y. and Guo, C. (2018) ‘Stability and H∞ performance of nonlinear fuzzy network control systems with time varying-delay’, 2018 37th Chinese Control Conference (CCC), pp. 253-256. DOI: 10.23919/ChiCC.2018.8482806.

Lyu, G. and Brennan, R. W. (2021) ‘Towards IEC 61499-Based Distributed Intelligent Automation: A Literature Review’, in IEEE Transactions on Industrial Informatics, 17(4), pp. 2295-2306. DOI: 10.1109/TII.2020.3016990.

Lyu, G. Fazlirad, A. Brennan, R.W. (2020) ‘Multi-agent modeling of cyber-physical systems for IEC 61499 based distributed automation’, Procedia Manuf, 51, pp. 1200–1206. DOI: 10.1016/j.promfg.2020.10.168.

Li, G. Wang, L. Yang, R. (2021) ‘Event-triggered Optimal Control for the NCSs with Time Delays’, Proceedings of the 40th Chinese Control Conference, pp. 4815-4818, DOI: 10.23919/CCC523632021.9550753.

Liu, T. and Liu, X. (2018) ‘Stability criterion for networked control systems based on T-S model with time-varying delays’, 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS), pp. 1085-1088. DOI: 10.1109/DDCLS.2018.8516056.

Mahmoud, M. S. and Sabih, M. (2014) ‘Experimental investigations for distributed networked control systems’, IEEE Systems Journal, 8(3), pp. 717725. DOI: 10.1109/JSYST.2012.2228122.

Mehlhop, S. Walter, J. (2022) ‘Model-aware Simulation of IEC 61499 Designs’, 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-4. DOI: 10.1109/ETFA52439.2022.9921452.

Mikova, L. Virgala, I. Kelemen, M. (2016) ‘Speed Control of DC Motor’, American Journal of Mechanical Engineering, 4(7), pp. 380-384. doi. 10.12691/ajme-4-7-27

Mirghani, M. (2017) ‘Reuse of Obsolete Dish Antenna in Radio Astronomy’.

Niklova, N. K. (2016) ‘Lecture 19: Reflector Antennas’, Department of Electrical Engineering, McMaster University.

Maung, M. M. Latt, M. M. New, C. M. (2018) ‘DC Motor Angular Position Control using PID Controller with Friction Compensation’, International Journal of Scientific and Research Publications, 8(11), pp. 149-155. doi: 10.29322/IJSRP.8.11.2018.p8321

Missal, D. Hirsch, M. and Hanisch, H. (2007) ‘Hierarchical distributed controllers - design and verification’, 2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007), pp. 657-664. DOI: 10.1109/EFTA.2007.4416832.

Pang, C. Patil, S. Yang, C. Vyatkin, V. and Shalyto, A. (2014) ‘A portability study of IEC 61499: Semantics and tools’, 2014 12th IEEE International Conference on Industrial Informatics (INDIN), pp. 440-445. DOI: 10.1109/INDIN.2014.6945553.

Pang, C. Vyatkin, V. Deng, Y. and Sorouri, M. (2013) ‘Virtual smart metering in automation and simulation of energy-efficient lighting system,’ 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), pp. 1-8. DOI: 10.1109/ETFA.2013.6648040.

Pang, Z. Luo, W. Liu, G. Han, Q. (2021) ‘Observer-Based Incremental Predictive Control of Networked Multi-Agent Systems with Random Delays and Packet Dropouts’, IEEE Transactions on Circuits and Systems II: Express Briefs, 68(1), pp. 426-430, DOI: 10.1109/TCSII.2020.2999126.

Panjaitan, S., Frey, G. (2005) ‘Functional design for IEC 61499 distributed control systems using UML activity diagrams’, Int. Conf, Instrumentation, Communication and Information Technology, pp. 64–70.

Patil, S. Drozdov, D. Zhabelova, G. and Vyatkin, V. (2018) ‘Refactoring of IEC 61499 function block application — A case study’, 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pp. 726-733. DOI: 10.1109/ICPHYS.2018.8390797.

Patil, S. Dubinin, V. and Vyatkin, V. (2015) ‘Formal Verification of IEC61499 Function Blocks with Abstract State Machines and SMV -- Modelling’, 2015 IEEE Trustcom/BigDataSE/ISPA, pp. 313-320. DOI: 10.1109/Trustcom.2015.650.

Patil, S. Dubinin, V. and Vyatkin, V. (2015) ‘Formal modeling and verification of IEC 61499 function blocks with abstract state machines and SMV-execution semantics’, International Symposium on Dependable Software Engineering: Theories, Tools, and Applications. Springer pp. 300–315.

Polaków, G., Metzger, M. (2009) Design and Implementation of LabVIEW-Based IEC61499 Compliant Device.’, Holonic and Multi-Agent Systems for Manufacturing. HoloMAS 2009. Lecture Notes in Computer Science, 5696 (0). DOI.org/10.1007/978-3-642-03668-2_18.

Preuße, S. Missal, D. Gerber, C. Hirsch, M. and Hanisch, H. M. (2011) ‘On the use of model-based IEC 61499 controller design’, International Journal of Discrete Event Control Systems, 1(1), pp. 115–128

Rahmat-Samii, Y. Densmore, A. (2009) ‘A History of Reflector Antenna Development: Past, Present and Future’, 2009 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC), pp. 17-23. doi: 10.1109/IMOC.2009.5427640

Roderick, A. (2021) Direct Current Generator Components. Available at https://eepower.com/technical-articles/direct-current-generator-components/# (Accessed: 18 April 2023)

Rowell, D. (2002) ‘State-Space Representation of LTI Systems’, 2.14 Analysis and Design of Feedback Control Systems.

Rubin, J. (2018) Heinrich Hertz, The Discovery of Radio Waves. Available at https://www.juliantrubin.com/bigten/hertzexperiment.html (Accessed: 20 April 2023).

Santos, A. A. and da Silva, A. F. (2021) ‘Simulation and Control of a Cyber-Physical System under IEC 61499 Standard’, Procedia Manufacturing, 55, pp. 72-79. DOI: 10.1016/j.promfg.2021.10.011.

Sarkar, R. (2020) Basics of DC Motor. Available at: https://www.electronicsforu.com/resources/dc-motor-basics (Accessed: 18 April 2023)

Seale, E. (2016) DC Motors. Available at: http://solarbotics.net/starting/200111_dcmotor/200111_dcmotor.html (Accessed: 20 March 2023).

Sorouri, M. Vyatkin V. and Xie S. (2012) ‘Distributed control design of medical devices using plug-and-play IEC 61499 Function Blocks’, 2012 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), pp. 450-455.

Sheldon, R. (2023) antenna. Available at https://www.techtarget.com/searchmobilecomputing/definition/antenna (Accessed: 18 April 2023)

Shi, Y. and Yu, B. (2009) ‘Output feedback stabilization of networked control systems with random delays modeled by Markov Chains’, IEEE Transactions on Automatic Control, 54(7), pp. 1668-1674. DOI: 10.1109/TAC.2009.2020638.

Strasser, T. J. Christensen, H. Valente, A. Chouinard, J. E. Carpanzano, A. Valentini, et al., (2012) ‘The IEC 61499 Function Block Standard: Launch and Takeoff’, presented at the ISA Automation Week 2012, Orlando, US.

Strasser, T. J. Zoitl, A. Christensen, J. H. and Sünder, C. (2011) ‘Design and Execution Issues in IEC 61499 Distributed Automation and Control Systems’, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 41(1), pp. 41-51. DOI: 10.1109/TSMCC.2010.2067210.

Stroski, P. N. (2019) Type of antennas (Part 2, reflecting antennas). Available at https://www.electricalelibrary.com/en/2019/04/10/type-of-antennas-part-2-reflecting-antennas/ (Accessed: 20 April 2023)

Stubbs, J. (2011) ‘Ethernet protocol, EtherCAT, processes on the fly. Available at https://www.controleng.com/articles/ethernet-protocol-ethercat-processes-on-the-fly/ (Accessed: 30 April 2023)

Subramanian, M. and Priyanka, K. (2020) ‘Delay-dependent Stability Analysis of Network-controlled DC Motor with Time-invariant Delays’, 2020 IEEE 17th India Council International Conference (INDICON), pp. 1-7, DOI: 10.1109/INDICON49873.2020.9342554.

Sun, J. Chen, J. Dou, L (2014) ‘Networked Predictive Control for Linear Systems with Unknown Communications Delay’, 2014 UKACC International Conference on Control, pp. 668-672. DOI: 10.1109/CONTROL.2014.6915219.

Sunder, C. Zoitl, A. Strasser, T. and Favre-Bulle, B. (2005) ‘Intuitive control engineering for mechatronic components in distributed automation systems based on the reference model of IEC 61499’, INDIN ‘05 2005 3rd IEEE International Conference on Industrial Informatics, 2005, pp. 50-55. DOI: 10.1109/INDIN.2005.1560351.

Thramboulidis, K. (2009) “IEC61499 Function Block Model: Facts and Fallacies,” IEEE Industrial Electronics Magazine, 3(4), pp. 7-26.

Väänänen, E. and Vyatkin, V. (2017) ‘Estimation, measurement and improvement of distributed automation applications performance’, IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, pp. 5426-5431. DOI: 10.1109/IECON.2017.8216940

Vyatkin, V. (2011) ‘IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review’, IEEE Transactions on Industrial Informatics, 7(4), pp. 768-781. DOI: 10.1109/TII.2011.2166785.

Vyatkin, V. Hanisch, H. Pang, C. and Yang, C. (2009) ‘Closed-Loop Modeling in Future Automation System Engineering and Validation’, in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 39(1), pp. 17-28. DOI: 10.1109/TSMCC.2008.2005785.

Vyatkin, V. Hirsch, M. and Hanisch, H. (2006) ‘Systematic Design and Implementation of Distributed Controllers in Industrial Automation’, 2006 IEEE Conference on Emerging Technologies and Factory Automation, pp. 633-640. DOI: 10.1109/ETFA.2006.355448

Vyatkin, V. (2009) ‘The IEC 1499 standard and its semantics’, IEEE Industry Electron Magazine., Vol 3(4), pp. 40-48. DOI: 10.1109/OJIES.2021.3138537

Wang, Y. Guo, N. Yan, G. Liu, J. (2022) ‘Design of Integrated Energy System Based on IEC 61499 and OPC UA’, Proceedings of the 41st Chinese Control Conference, pp 4270-4275. DOI: 10.23919/CCC55666.2022.990135.

Wolff, C. (1997) Gregory Antenna. Available at https://www.radartutorial.eu/06.antennas/Gregory%20Antenna.en.html (Accessed: 20 April 2023).

Wielebinski, R., Kellermann, K.I., and Orchiston, W., eds. (2007). ‘The Early History of European Radio Astronomy’, Astronomische Nachrichten. 328(5), 375–446.

Wikipedia. (2023) Karl Guthe Jansky. Available at https://en.wikipedia.org/wiki/Karl_Guthe_Jansky (Accessed: 20 April 2023)

Xiao, F. Shi, Y. and Chen, T. (2021) ‘Robust stability of networked linear control systems with asynchronous continuous- and discrete-time event-triggering schemes’, IEEE Transactions on Automatic Control, 66(2), pp. 932-939. DOI: 10.1109/TAC.2020.2987649.

Xie, H. Zheng, J. Wang, M. Chai, R. (2020) ‘Networked DC Motor Control with Time-Varying Delays and Application to a Mobile Robot’, 2020 IEEE 16th International Conference on Control & Automation (ICCA), pp. 171-176, DOI: 10.1109/ICCA51439.2020.9264587.

Yan, J. and Vyatkin, V. (2011) ‘Distributed execution and cyber-physical design of Baggage Handling automation with IEC 61499’, 2011 9th IEEE International Conference on Industrial Informatics, 2011, pp. 573-578. DOI: 10.1109/INDIN.2011.6034942.

Yoong, L. H. P. Roop, S. Vyatkin, V. and Salcic, Z. (2009) ‘A Synchronous Approach for IEC 61499 Function Block Implementation’, in IEEE Transactions on Computers, 58(12), pp. 1599-1614. DOI: 10.1109/TC.2009.128.

Yu, X. and Jiang, J. (2014) ‘Analysis and compensation of delays in FF H1 fieldbus control loop using model predictive control’, IEEE Transactions on Instrumentation and Measurement, 63(10), pp. 2432-2446. DOI: 10.1109/TIM.2014.2310093.

Zhabelova, G. et al., (2014) ‘Cyber-physical components for heterogeneous modeling, validation and implementation of smart grid intelligence’, 2014 12th IEEE International Conference on Industrial Informatics (INDIN), pp. 411-417. DOI: 10.1109/INDIN.2014.6945548.

Zhabelova, G. Yang, C. Vyatkin, V. Etherden N. and Christoffersson, L. (2017) ‘Open architecture for cost-effective protection and control of power distribution networks’, 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 729-735. DOI: 10.1109/SmartGridComm.2016.7778848.

Zhang, D. Nguang, S. K. and Yu, L. (2017) ‘Distributed control of large-scale networked control systems with communication constraints and topology switching’, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(7), pp. 1746-1757. DOI: 10.1109/TSMC.2017.2681702.

Zhang, L. Shi, Y. Chen, T. and Huang, B. (2005) ‘A new method for stabilization of networked control systems with random delays’, Proceedings of the 2005, American Control Conference, 2005, 1, pp. 633-637. DOI: 10.1109/ACC.2005.1470028.

Zhang, S. Zhao, D. Li, C. and Stobart, R. (2015) ‘Fuzzy speed control of networked motion control systems’, Journal of Computational and Nonlinear Dynamics, 10(6), pp. 061013/1-061013/9. DOI:10.1115/1.4029903.

Zhaoping, D.U. Songlin, H.U. Jianzhen, L.I. (2013) ‘Modeling and Stabilization for Singular Networked Cascade Control Systems with State Delay’, Proceedings of the 32nd Chinese Control Conference, pp. 6704-6709.

APPENDICES

APPENDIX A: MATLAB SCRIPT FILES

Appendix A4.1: DC Motor Parameters

Calculates all the motor parameters needed for the Simulink block diagrams. The code calculates the state space matrices from the motor parameters.

```
%Calculate DC motor open loop variables
%Kevin Love – 2022

%Define motor and plant parameters
Kb = 0.425;                             %Electromotive force constant
Kt = 2.2;                               %Torque constant
Ra = 14.3;                              %Armature resistance
Ja = 0.013;                             %Moment of inertia of the armature
Jl = 0.001;                             %Moment of inertia of the load
Ba = 0.0001;                            %Motor damping coefficient
Bl = 1;                                 %Load damping coefficient
N1 = 1;                                 %Number of gears teeth N1
N2 = 270;                               %Number of gears teeth N2

%Calculate motor and load moment of inertia
Jm = Ja+Jl*(N1/N2)^2

%Calculate motor and load damping coefficient
Bm = Ba+Bl*(N1/N2)^2

%Substitute parameters into Km and Am
Km = Kb/(Ra*Jm)
Am = ((Ra*Bm)+(Kb*Kt))/(Ra*Jm)
```

Appendix A4.2: Open Loop System in State Space Form

```
%Calculate DC motor open loop state space matrices
%Kevin Love - 2022

%Transfer Function of Motor
num = [Am];                     %Declare numerator of open loop transfer function
den = [1 Km 0];                 %Declare denominator of open loop transfer function
Motor_TF = tf(num,den);         %Declare complete transfer function


%State Space Representation of Motor
Motor_SS = ss(Motor_TF);        %Convert transfer function to state space

%Create variables A,B,C,D from state space reference
A = Motor_SS.A;
B = Motor_SS.B;
C = Motor_SS.C;
D = Motor_SS.D;
```

Appendix A5.1: Full State Feedback Controller

Calculates the root locus as well as the state space equations of the closed loop system. Includes the test for controllability as well as the calculation to work out the gain matrix.

```
%Develop full state feedback controller
%Kevin Love - 2023

%Root locus of open loop system
rlocus(OL_Motor_SS);

%Closed loop transfer function of motor
CL_motor_TF = feedback(OL_motor_TF,1);

%Closed loop state space
CL_Motor_SS = ss(CL_motor_TF);

%Declare closed loop matrices
Acl = CL_Motor_SS.A;
Bcl = CL_Motor_SS.B;
Ccl = CL_Motor_SS.C;
Dcl = CL_Motor_SS.D;

%Check for controllability
control=ctrb(Acl,Bcl);
Rank=rank(control)

%Desired poles
J = [-2 -6];

%Acker formula to determine K gain matrix
K = acker(Acl,Bcl,J);

%Add state feedback controller
A_ClosedLoop = Acl - Bcl*K;
eig(A_ClosedLoop);

syscl = ss(A_ClosedLoop,Bcl,Ccl,Dcl);

%Root locus of closed loop system with full state feedback controller
rlocus(syscl)
```

Appendix A5.2: Full State Feedback Controller with Integral Gain

Calculates the gain matrix and integral gain for the new closed loop system. A root locus is also done to test if the poles are at the correct position.

```
%Develop full state feedback controller with integral control
%Kevin Love - 2023
```

```
%Determine new closed loop system with integrator
Aint=[Acl zeros(2,1);-Ccl 0];
Bint=[Bcl;0];
Cint=[Ccl 0];

%Set new desired poles
Pint= [-2 -6 -8];

%Calculate integrator matrix Ke
Ke =acker(Aint,Bint,Pint);

%New closed loop system
Anew = Aint-Bint*Ke;
Br=[0;0;1];
Bnew=Bint+Br;
sysclosed = ss(Anew,Bnew,Cint,Dcl);
rlocus(sysclosed)
```

Appendix A5.3: Case Study 5.3.4 Graph Outputs

Plots all the responses that are developed in Simulink. These responses are for the closed-loop system without controller, closed-loop system with state feedback, and closed-loop system with state feedback with integral control. The plot for the random number generator is also included in this code.

```
%%Case Study 5.3.4 Plots
%Kevin Love - 2023

%Case 1
figure('Name', 'Case Study 5.3.4');
subplot(2,2,1)
xlim([0 20])
ylim([0 1.2])
hold on
plot(out.Time, out.ClosedLoopStepResponse, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 1" + newline + "Step Response of a Closed Loop DC Motor System")
plot(out.Time, out.ClosedLoopStepResponse_SP, 'LineWidth',1)
legend('Position (θ)', 'Set Point')
grid on
hold off

%Case 2
subplot(2,2,2)
xlim([0 20])
ylim([0 0.2])
hold on
plot(out.Time, out.ClosedLoopWithStateFeedback, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
```

```
title("Case 2" + newline + "Step Response of a Closed Loop DC Motor System with State
Feedback")
plot(out.Time, out.ClosedLoopWithStateFeedback_SP, 'LineWidth',1)
legend('Position (θ)', 'Set Point')
grid on
hold off


%Case 3
subplot(2,2,3)
xlim([0 20])
ylim([0 1.2])
hold on
plot(out.Time, out.AddedIntegral1, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 3" + newline + "Step Response of a Closed Loop DC Motor System with State
Feedback and Integral Control")
plot(out.Time, out.AddedIntegral1_SP, 'LineWidth',1)
legend('Position (θ)', 'Set Point')
grid on
hold off

%Case 4
subplot(2,2,4)
xlim([0 20])
ylim([0 110])
hold on
plot(out.Time, out.AddedIntegral2, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 4" + newline + "Step Response of a Closed Loop DC Motor System with State
Feedback and Integral Control")
plot(out.Time, out.AddedIntegral2_SP, 'LineWidth',1)
legend('Position (θ)', 'Set Point')
grid on
hold off

%Extra Case Testing robustness of controller
figure ('Name', 'Response Test')
xlim([0 24])
hold on
plot(out.Time, out.RandomGenerator, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title('System Response to Random Number Generator Input')
plot(out.RandomGenerator_SP, 'LineWidth',1)
legend('Position (θ)', 'Set Point')
grid on
hold off

%Generate charateristics of the different cases for Table 5.1
stepinfo(out.ClosedLoopStepResponse,out.Time)
stepinfo(out.ClosedLoopWithStateFeedback,out.Time)
stepinfo(out.AddedIntegral1,out.Time)
stepinfo(out.AddedIntegral2,out.Time)
```

Appendix A5.4: Observer Gain

Checks that the system is observable and calculates the observer gain.

```
%Calculate Observer Gain
%Kevin Love - 2023


%Check for observability
observe=obsv(Acl,Ccl)
Rank=rank(observe)

%State Feedback Observer Gain Matrix
L = acker(Acl',Ccl',J*4)'
```

Appendix A5.5.1: Sensor to Controller Delay Graph Outputs

MATLAB code that creates the graphs for section 5.5.1 showing the effects of sensor to controller network delays on the step responses of the different control systems developed

```
%Section 5.5.1 – sensor to controller delays
%Kevin Love - 2023

%Case 1
figure('Name', 'Case Study 5.5.1');
subplot(2,2,1)
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 1" + newline + "Position output with 100ms delay"+ newline +"between sensor
and controller")
plot(out.Time, out.Position_Delayed_100ms, 'LineWidth',1)
legend('Ts-c = 0ms', 'Ts-c = 100ms')
grid on
hold off


%Case 2
subplot(2,2,2)
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 2" + newline + "Position output with 400ms delay"+ newline +"between sensor
and controller")
plot(out.Time, out.Position_Delayed_400ms, 'LineWidth',1)
legend('Ts-c = 0ms', 'Ts-c = 400ms')
grid on
hold off
```

```
%Case 3
subplot(2,2,3)
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 3" + newline + "Position output with 800ms delay"+ newline +"between sensor
and controller")
plot(out.Time, out.Position_Delayed_800ms, 'LineWidth',1)
legend('Ts-c = 0ms', 'Ts-c = 800ms')
grid on
hold off

%Case 4
subplot(2,2,4)
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 4" + newline + "Position output with 1200ms delay"+ newline +"between sensor
and controller")
plot(out.Time, out.Position_Delayed_1200ms, 'LineWidth',1)
legend('Ts-c = 0ms', 'Ts-c = 1200ms')
grid on
hold off

%Generate characteristics of the different cases for Table 5.1
stepinfo(out.Position_Delayed_100ms,out.Time)
stepinfo(out.Position_Delayed_400ms,out.Time)
stepinfo(out.Position_Delayed_800ms,out.Time)
stepinfo(out.Position_Delayed_1200ms,out.Time)
```

Appendix A5.5.2: Controller to Actuator Delay Graph Outputs

MATLAB code that creates the graphs for section 5.5.2 showing the effects of controller to actuator network delays on the step responses of the different control systems developed

```
%Section 5.5.2 – controller to actuator delays
%Kevin Love - 2023

%Case 1
figure('Name', 'Case Study  5.5.2');
subplot(2,2,1)
%xlim([0 20])
%ylim([0 1.2])
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 1" + newline + "Position output with 50ms delay"+ newline +"between controller
and actuator")
plot(out.Time, out.Position_Delayed_50ms, 'LineWidth',1)
```

```
legend('Tc-a = 0ms', 'Tc-a = 50ms')
grid on
hold off

%Case 2
subplot(2,2,2)
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 2" + newline + "Position output with 100ms delay"+ newline +"between controller
and actuator")
plot(out.Time, out.Position_Delayed_100ms, 'LineWidth',1)
legend('Tc-a = 0ms', 'Tc-a = 100ms')
grid on
hold off

%Case 2
subplot(2,2,3)
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 3" + newline + "Position output with 125ms delay"+ newline +"between controller
and actuator")
plot(out.Time, out.Position_Delayed_125ms, 'LineWidth',1)
legend('Tc-a = 0ms', 'Tc-a = 125ms')
grid on
hold off

%Case 2
subplot(2,2,4)
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 4" + newline + "Position output with 150ms delay"+ newline +"between controller
and actuator")
plot(out.Time, out.Position_Delayed_150ms, 'LineWidth',1)
legend('Tc-a = 0ms', 'Tc-a = 150ms')
grid on
hold off

%Generate characteristics of the different cases for Table 5.2
stepinfo(out.Position_Delayed_50ms,out.Time)
stepinfo(out.Position_Delayed_100ms,out.Time)
stepinfo(out.Position_Delayed_125ms,out.Time)
stepinfo(out.Position_Delayed_150ms,out.Time)
```

Appendix A6.1: TwinCAT Transformation Commands

MATLAB code to be executed in the MATLAB command window to create a TwinCAT object from the Simulink block diagram in the current folder path.

```
%%Convert Simulink block diagram to TwinCAT object
%Kevin Love - 2023

%Open the model that needs to be converted
modelName = 'DCMotorCLSSSFBIC1';
open_system(modelName);

%Change the solver type to fixed step
TwinCAT.ModuleGenerator.Simulink.ModelExportConfig.ShowModelParam(modelName,'Sol
verType','Fixed-step');

%Change the system target file to TwinCatGrt
TwinCAT.ModuleGenerator.Simulink.ModelExportConfig.ShowModelParam(modelName,'Sy
stemTargetFile','TwinCatGrt.tlc');

%Change the vendoer name to the project name underscore VendorName
TwinCAT.ModuleGenerator.Simulink.ModelExportConfig.ShowModelParam(modelName,'Tc
Project_VendorName');

%Save project and build using Simulink Coder
save_system(modelName);
slbuild(modelName);
```

Appendix A7.5.1: Position output of control system with and without delay

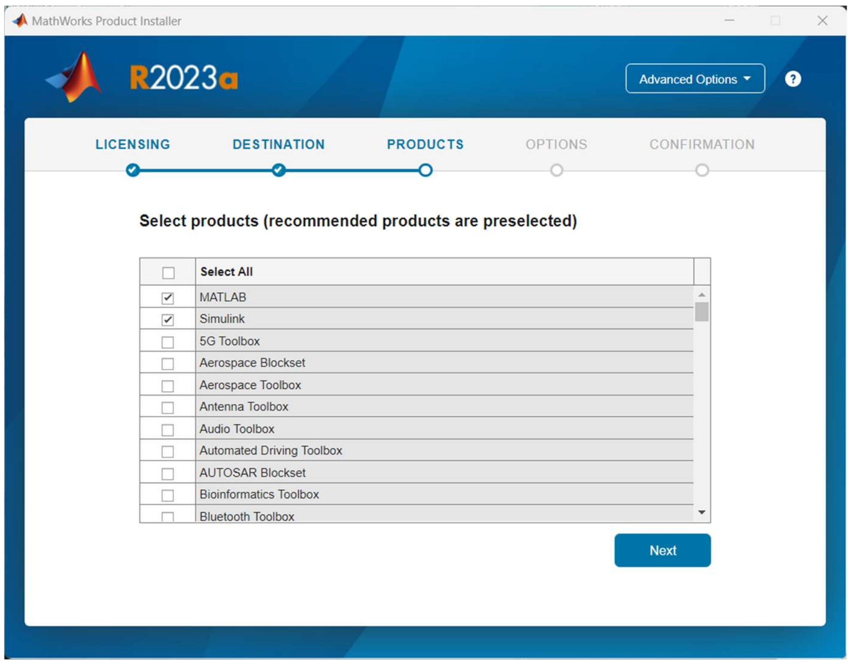
MATLAB code to graph the outputs of the closed-loop control system with delay vs without delay.

```
%Chapter 7 Graph Ouputs
%Kevin Love 2023

%Delay vs No Delay Graph
figure('Name', 'Delay vs No Delay');
hold on
plot(out.Time, out.Position, 'LineWidth',1)
xlabel('Time (s)')
ylabel('Position (θ)')
title("Case 1" + newline + "Position output without delay verse with delay"+ newline)
plot(out.Time, out.Position_Delayed, 'LineWidth',1)
legend('Position (θ)', 'Position (θ) with delay')
grid on
hold off
```

Appendix B6.1: MATLAB/Simulink installation procedure

1. Download the setup file from https://www.mathworks.com/products/matlab.html
2. Save the setup file to the desired location on the PC.
3. Run MATLAB setup file from the selected location.
4. The setup extracts and the product installer start.
5. Enter an email address linked to a MathWorks account, then click on “Next”.
6. Enter password for the email address used, then click on “Next”.
7. Accept the terms of the license agreement, then click on “Next”.
8. Configure required licensing and confirm user, then click on “Next”.
9. Select the default destination for file storage, then click on “Next”.
10. Select MATLAB and Simulink products to be installed.
11. Also select any MATLAB and Simulink add-ons, then click on “Next”.



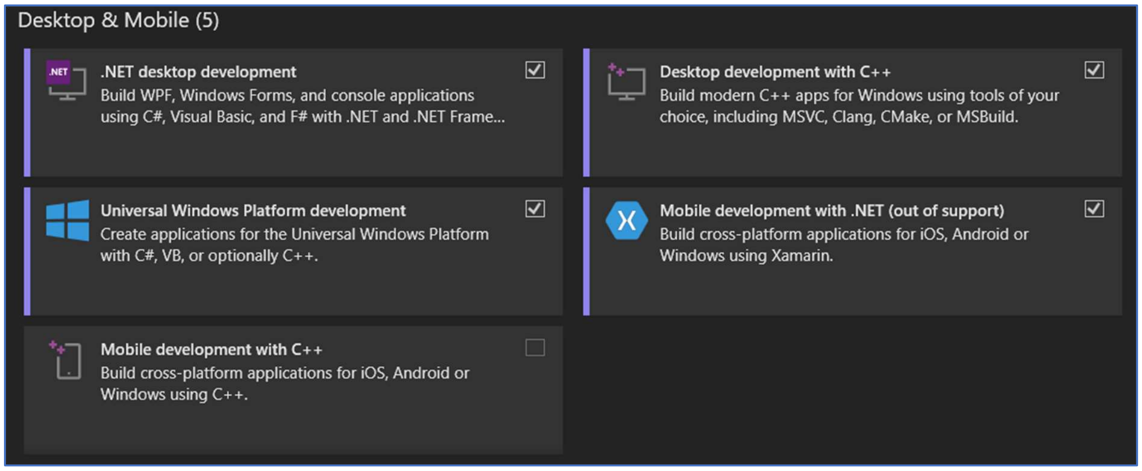
Select products (recommended products are preselected)

☐	Select All
☑	MATLAB
☑	Simulink
☐	5G Toolbox
☐	Aerospace Blockset
☐	Aerospace Toolbox
☐	Antenna Toolbox
☐	Audio Toolbox
☐	Automated Driving Toolbox
☐	AUTOSAR Blockset
☐	Bioinformatics Toolbox
☐	Bluetooth Toolbox

12. The installation manager installs all the selected software.
13. Click on “Close” once the installation is complete.

Appendix B6.2: Visual Studio 2019 installation procedure

1. Download the setup file from https://www.mathworks.com/products/matlab.html
2. Save the setup file to the desired location on the PC.
3. Run Visual Studio 2019 setup file from the selected location
4. The setup extracts and the Visual Studio Installer starts.
5. Once extraction completes, select the below software packages by ticking the box



Desktop & Mobile (5)

.NET desktop development ✓
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

Desktop development with C++ ✓
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Universal Windows Platform development ✓
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

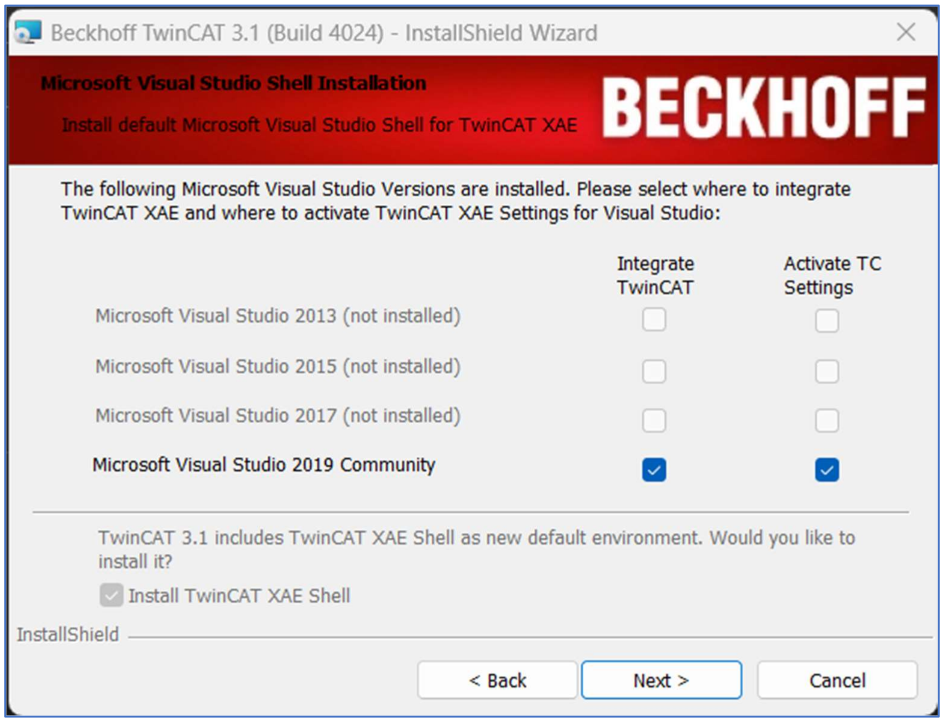
Mobile development with .NET (out of support) ✓
Build cross-platform applications for iOS, Android or Windows using Xamarin.

Mobile development with C++ ☐
Build cross-platform applications for iOS, Android or Windows using C++.

6. Click on "Install" once all the required packages are selected
7. The installer will download and install the packages automatically
8. Restart the PC after installation is complete

Appendix B6.3: TwinCAT 3 installation procedure

1. Download he setup file from https://www.beckhoff.com/en-en/support/download-finder/software-and-tools/
2. Save the setup file to the desired location on the PC.
3. Run TwinCAT 3 setup file from the selected location as an administrator
4. Accept the terms of the license agreement, then click on “Next”.
5. Accept the terms of the 2nd license agreement, then click on “Next”.
6. Select “Complete” installation setup, then click on “Next”.
7. Click on “Install” to begin installing all the necessary software.
8. Click on “Finish” once the installation is complete
9. Restart the PC after installation is complete
10. The TwinCAT 3 set file automatically runs again
11. Select the version of Visual Studio installed on the PC for integration
12. Click on “Next” to being the Visual Studio Shell installation


Beckhoff TwinCAT 3.1 (Build 4024) - InstallShield Wizard

Microsoft Visual Studio Shell Installation

Install default Microsoft Visual Studio Shell for TwinCAT XAE

The following Microsoft Visual Studio Versions are installed. Please select where to integrate TwinCAT XAE and where to activate TwinCAT XAE Settings for Visual Studio:

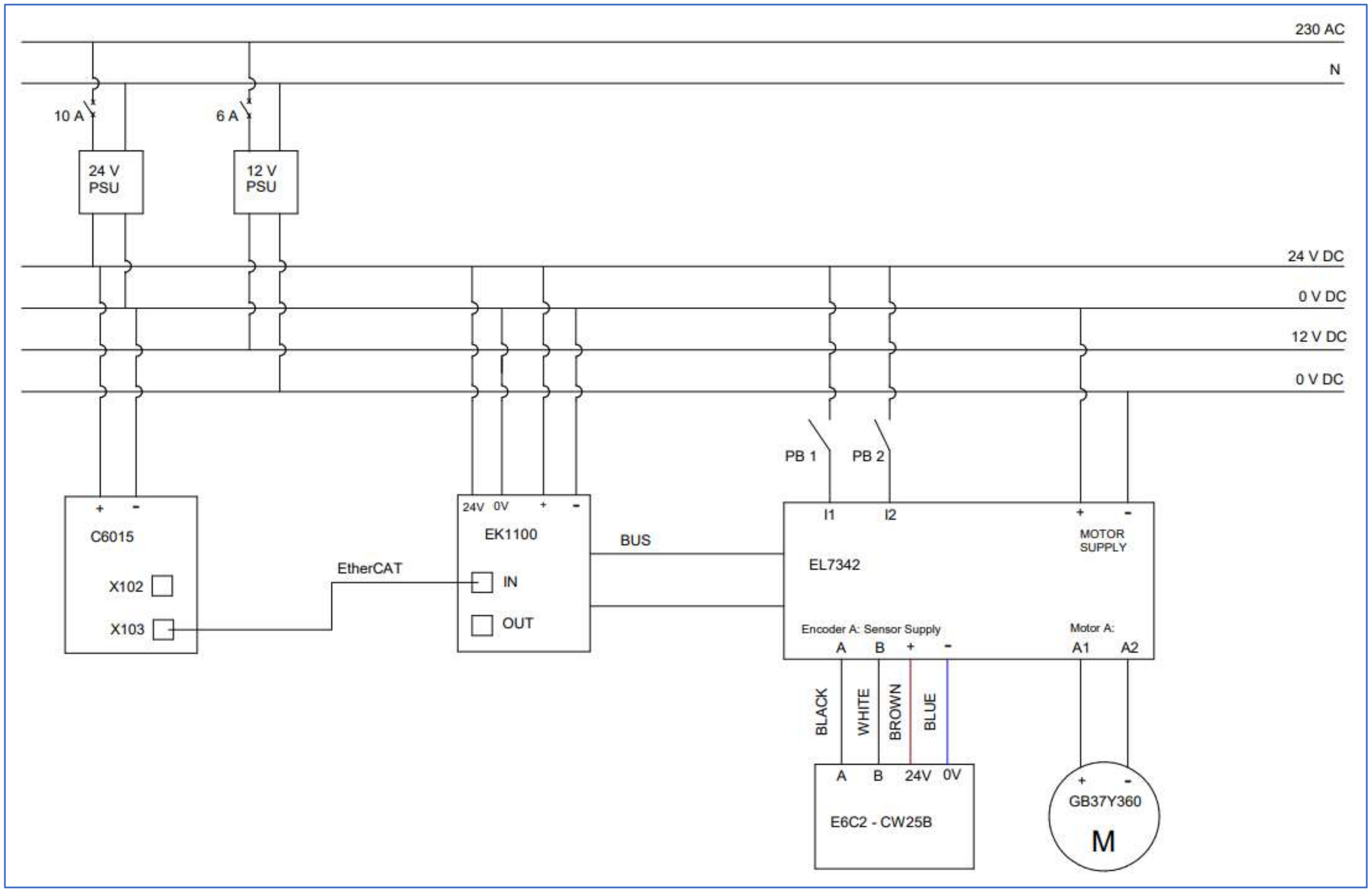
	Integrate TwinCAT	Activate TC Settings
Microsoft Visual Studio 2013 (not installed)	☐	☐
Microsoft Visual Studio 2015 (not installed)	☐	☐
Microsoft Visual Studio 2017 (not installed)	☐	☐
Microsoft Visual Studio 2019 Community	✓	✓

TwinCAT 3.1 includes TwinCAT XAE Shell as new default environment. Would you like to install it?

✓ Install TwinCAT XAE Shell

13. Click on “Next” to being the Visual Studio Shell installation
14. Restart the PC after installation is complete.

Appendix C7.1: Test rig wiring diagram


230 AC
N
10 A
6 A
24 V PSU
12 V PSU
24 V DC
0 V DC
12 V DC
0 V DC
PB 1
PB 2
C6015
X102
X103
EtherCAT
24V 0V + -
EK1100
IN
OUT
BUS
I1
I2
EL7342
MOTOR SUPPLY
Encoder A: Sensor Supply
A B + -
Motor A:
A1 A2
BLACK
WHITE
BROWN
BLUE
A B 24V 0V
E6C2 - CW25B
GB37Y360
M