Evaluation of network conditions on the performance of an Industrial IoT control and monitoring system

Randolph Bock
orcid.org/0000-0002-4059-4224

Dissertation accepted in fulfilment of the requirements for the degree Master of Engineering in Computer and Electronic Engineering at the North-West University

Supervisor:	Prof K.R. Uren
Co-supervisor:	Prof G. van Schoor

Graduation: June 2021
Student number: 25905570

“The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency.”

Bill Gates

Abstract

Advances in smart manufacturing and Industry 4.0 have drawn the interest of the industry. The rise in popularity of *internet of things* and *cyber-physical systems* in a manufacturing environment meant the broader connection of devices, servers, sensors and actuators within a closed-loop control system. This study aims to examine the impact of different network conditions on the performance of an industrial control process using both local controllers and a remote system controller. Network conditions examined are network node location of the remote controller, latency, packet delay variance (jitter) and packet loss. The study uniquely uses both industrial hardware and communication equipment from Siemens© for the emulated industrial plant, as well as consumer networking equipment for the connection en emulation of a cloud-connected remote controller. An industrial plant is emulated through a system of electric motors. The emulated system uses both local controllers and a remote controller. The local controllers are responsible for direct control and monitoring of a motor, including failsafe features for the motor. A remote controller is then responsible for control over the whole interconnected system.

A considerable increase in system response time is observed when the remote controller is moved from a LAN connection to a WAN connection. A remote controller connected to a LAN connection is described as a fog node, while the remote controller connected with a WAN connection is described as a remote cloud node. Even with delay mitigation implemented on the remote controller connected via WAN, lower system performance is still observed than the remote controller connected via LAN. Applying a double exponential smoothing model as delay mitigation on the WAN connected remote controller improved system performance over not having any delay mitigations. However, performance is better when via LAN compared to WAN with delay mitigations implemented.

Relationships between the average system response time and the different network conditions are made through curve fitting of the measured data. As network conditions worsen, the performance of the system is degraded in a linear and sometimes quadratic fashion. The relationship between system response time and network latency is slightly quadratic, the relationship between jitter and system response time is linear, and the relationship between packet loss and system response time is quadratic.

The performance impact of different network conditions are measured for a system with and without delay mitigations implemented on the remote controller: two different delay mitigation mechanisms where tested, exponential moving average and double exponential smoothing model. The exponential moving average proved ineffective for the emulated system by yielding lower system performance per test point, likely due to exponential moving average being more suited for small changes, and the implementation thereof increased processing time. Implementing a double exponential smoothing model as delay mitigation increased system performance for each test point. Double exponential smoothing model boasts a better prediction ability, making it more suited as for delay mitigation.

Some future work could focus on testing the scalability of an IIoT system, as well as how implementing security protocols could impact the performance of an IIoT system. Tests should be done to determine how scaling the system or adding security protocols to the system with varying network conditions will impact the performance of an IIoT system, as the effects of scalability and security will be compounded along with the performance impact of different networking conditions.

Keywords: IIoT, Cyber-physical systems, latency, jitter, packet loss, Industry 4.0.

Contents

List of acronyms

CPPS	Cyber-Physical Production System
CPS	Cyber-Physical System
CPU	Central Processing Unit
CSV	Comma-separated Values
DESM	Double Exponential Smoothing Model
EDT	Experimental Digital Twin
EMA	Exponential Moving Average
FOC	Field Oriented Control
HMI	Human Machine Interface
I4.0	Industry 4.0
IaaS	Infrastructure as a Service
ICS	Internet-based Control Systems
IEEE	Institute of Electrical and Electronic Engineers
IIoT	Industrial Internet of Things
IoE	Internet of Everything
IoP	Internet of People
IoS	Internet of Services
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
MBSE	Model Based System Engineering
MRO	Maintenance, Repair, Overhaul
NCS	Network Control System
NAS	Network Automation System
PDV	Packet Delay Variance
PLC	Programmable Logic Controller
RFID	Radio-frequency identification
RMSE	Root Mean Square Error
RTT	Round-trip Time
RTP	Real-time Transport Protocol
SaaS	Software as a Service
SCADA	Supervisory Control And Data Acquisition
TCP/IP	Transmission Control Protocol/Internet Protocol
TSN	Time-Sensitive Networking
VFD	Variable Frequency Drive
VSD	Variable Speed Drive
WAN	Wide Area Network

List of Figures

List of Tables

Chapter 1

Introduction

This study is conducted on the topic of Industry 4.0/Smart manufacturing, with the intent of determining the effects of network conditions on the performance of an IIoT control and monitoring system.

This chapter aims to provide background information on Industry 4.0/Smart manufacturing and the relevant challenges of Industry 4.0. The problem of uncertain network communication and the effects of different networking conditions on an IIoT control and monitoring system is identified.

The problem statement is given, followed by the research goals and objectives for this study. The research methodology for each of the research goals is then presented. An overview of the dissertation is then outlined.

1.1 Background

Advancements in new technologies drove the world into its fourth industrial revolution, or as it has come to be known Industry 4.0 (I4.0). The fourth industrial revolution contains a trend of system-wide automation, and the use of cyber-physical systems (CPSs), where the physical world is linked via communication models to a virtual world. Emerging and maturing technologies such as cloud computing and Internet of Things (IoT) bring advancements in data collection, process monitoring and improvements in the intelligence and decision-making abilities of industrial systems [8]. IoT allows for everyday and simple objects to communicate and interact with each other, therefore increasing the number of connected devices. The increase in devices increases the amount of data transfer and collections. Cloud computing can support an increase in the number of users as well as the increased amount of data associated with the increase in users and devices.

The benefits to industry of implementing an IoT and cloud-based industrial control system will be centralised control and cost optimisation of systems. However, moving parts of the control system are not straightforward since it can affect the stability, robustness and reliability of the system [8].

Problems that exist within Industry 4.0 include: Effectively processing and evaluating the large amount of data received from IoT devices; the cyber-security aspects of keeping data safe and private and preventing malicious attacks; and finally the problem explored in this study is the unpredictable network delay introduced in distributed control loops. The communication networks introduce delays. The delays are due to limited bandwidth and overhead in communication nodes and networks. The delays in different systems will be varying randomly. Control systems with varying delays cannot be described as time-invariant. Standard techniques and theory can, therefore, not be used in the design and analysis of distributed control systems [8].

Related work on the performance impact of different network conditions on an Industrial IoT (IIoT) include:
Industrial Automation as a Cloud Service [5] explores the implementation of industrial automation as a cloud service, delay and fault tolerance compensation methods are discussed.
Real-time Control in Industrial IoT [9] explores the impact of network conditions on the real-time control capabilities of an IoT system. This study explores the effects of network conditions on an IoT system with

only a remote controller controlling the system. Delay mitigations for an IIoT system are also discussed. The paper by Tipsuwan et al. [10] explores the network-induced delay effect in a control loop of a networked control system (NCS). The effects and characteristics of network delays are explored, and different NCS control methodologies are presented.

1.2 Problem statement

A problem many of the Industry 4.0 solutions and products has is the use of standard internet technologies as the communications channel to interact with industrial automation systems [11].

The internet protocol (IP) networks as communication channel pose different challenges as the traffic, latency and packet delay variance (jitter) within the communications network is always varying and unpredictable. Therefore, an unpredictable arrival time of network data packets for the Industrial IoT system exists. Industrial automation happens in a real-time environment where low and predictable latencies are requirements.

Cloud computing connects to a system over a Wide Area Network (WAN) using the internet as a communication medium. The research question posed then is as follows: "Can a reliable industrial control system be implemented with IoT and cloud computing as part of it, and how will different network conditions affect the performance of such a system?" An additional question then develops: "How can the effects of the different network conditions be decreased by adding delay mitigation mechanisms to the remote controller of the system?"

1.3 Research goals and objectives

The study focuses on the effect of adding a remote controller to a typical closed-loop control system, where a typical closed-loop control consists of an actuator, process, sensor and local controller. This research study aims to move the control to a remote server. When control is moved to a remote controller, network conditions like latency, jitter and packet loss are introduced.

The research goal is to determine the performance impact of different network conditions on a remote controller *Industrial Internet of Things* system, along with investigating the effect of implementing delay mitigation mechanisms and structures on the remote controller. The tests are uniquely done for determining the impact of network conditions on an *industrial* system using industrial hardware and communication components with local and remote controllers.

It is expected that when the network location of the remote controller is moved from a LAN connection to a WAN connection that a considerable increase in system response time will be observed because of the increased transmission time due to the extra network hops added to the communication system. It is also expected that when network conditions such as latency and jitter are added to the IIoT control and monitoring system that the response times of the system will increase linearly. In contrast, it is expected that when packet losses are introduced in the system that the response time will increase quadratically. Packet loss entails that the packet is not received and that a new packet had to be sent over, drastically increasing the transmission time of packets over the network. The implementation of delay mitigations are expected to increase system performance over the system with no delay mitigations for the same network conditions. Predictions are that the double exponential smoothing model will increase system performance more than exponential moving average as delay mitigation. The double exponential smoothing model boasts better forecasting ability, while the exponential moving average is better suited for minor disturbances.

The research objectives are as follows:

1. **Experimental design**
   Firstly, to gain context of closed-loop control system architectures whether it be typical, networked or internet control loops. A proposed control loop architecture for determining the performance impact of network conditions on an IIoT system is made. A consideration to be made for the proposed architecture is that the IIoT system is to be constructed from Siemens© industrial components in a mechatronics laboratory, made available for this study. Another consideration for the system architecture is the use of a network-attached remote controller within the closed-loop control along with a method to vary the network conditions to allow for system performance impact testing of different network conditions on an emulated IIoT control and monitoring system. The scope of the study does not include security implementations and the impact thereof on such a system.

   Thereafter, a high-level experimental design of the IIoT system and network is created according to the proposed control loop architecture made. The experimental design for the physical system takes into consideration the industrial hardware within a mechatronics laboratory made available for this study. Testing methodologies and test schemes are introduced in the high-level experimental design in order to determine the system performance impact of network conditions on an IIoT system. Tests created are to determine the performance impact of moving the remote controller from a local area network (LAN) to a wide area network (WAN) is measured. Subsequently, different network conditions are tested by using a network emulator to change the network's conditions and behaviours. The network conditions tested for include: network latency, packet delay variance (jitter) and packet loss.

2. **Create a digital simulations model**
   A digital simulations model of the IIoT system is to be constructed to digitally represent the behaviour, information and communication of the physical system. The simulation model supports testing control schemes before the implementing thereof on the physical system. The digital simulations model is also used in determining the system performance impact for different network conditions by applying the same network conditions to the simulation model and the experimental system. The simulation model's behaviour and response are to be verified for an accurate representation of the physical system. The control scheme response and logged motor velocities per timestamp obtained from the model are also verified for its operation. Tests are, therefore, to be conducted to verify that the behaviour and response of the simulation model are as expected.

3. **Construct the experimental system**
   The experimental setup is constructed based on the experimental design created for this study. The system is constructed in the mechatronics laboratory using both industrial hardware and consumer networking equipment made available for this study. The experimental setup consists of the physical connection of hardware components, as well as software creation and implementation to allow for an operational and emulated IIoT system with the ability to test the impact of different network conditions on the system. After the construction of the experimental system, the system's response is to be verified if it is as expected for the given control scheme created for the system.

4. **Results and analysis**
   The tests designed prior are then conducted on the digital model and experimental system to determine the impact of different network conditions on the performance of the systems. The results obtained are then analysed, and relationships between the different network conditions and the performance of the system are calculated. The results obtained from the digital simulation for different networking conditions are then validated with the result obtained from the experimental system subjected to the same network conditions for an accurate representation of the effect of different network conditions on the performance of the emulated IIoT control and monitoring system.

1.4 Research methodology

In this section, the details of the research methodology steps are described for each of the research objectives.

1.) **Perform experimental design**
Different system architectures are examined, and an experimental system architecture is then proposed based on the requirements and constraints for the study. The system's functionalities and sub-parts are defined, along with the interfaces between components.

Considerations for the system architecture is that the architecture must allow for testing the impact of different network conditions on the performance of the system as per the research question. For this study, a requirement for a study in the topic of "smart manufacturing" or "Industry 4.0" is made. A mechatronics laboratory is made available for the study equipped with different stations consisting of Programmable Logic Controllers (PLCs), Variable Speed Drives (VSDs) and electric induction motor setups. The creation of the system architecture takes into account the equipment available for use in this study. The proposed architecture should make use of a remote controller and allows for the network condition between the plant and the remote controller to be altered for testing. The goal is to add a network-connected remote controller to a typical closed-loop control system. Therefore the proposed architecture is that of a bilateral control structure, where both local controllers and a remote controller is used. Technical knowledge and theory obtained for literature are applied to produce the proposed system's architecture for the prototype design.

Delay mitigation techniques are defined and described for implementation in the experiment setup. The research approach for testing the effects of latency, jitter and the addition of delay mitigations on the emulated system is described.

From the systems architecture created prior and the available industrial equipment available for the study in the mechatronics laboratory, an experimental design is created. The experimental system design includes network connections and a network diagram that allows for the modification of network communication from the remote controller to the emulated industrial plant via WANem, a network emulator program. The network architecture is designed to resemble the proposed bilateral closed-loop control architecture The experimental designed to emulate an industrial process of electric motors based on the proposed system architecture. The system emulates an industrial plant with three electric motors. Each motor is controlled locally by a Variable Speed Drive (VSD) and a Programmable Logic Controller (PLC). The individual local controllers are then connected to a remote controller through an industrial IoT gateway. The remote controller is responsible for control over the whole interconnected system. Tests are designed to obtain default system response times. Four tests are required, test one for determining how the network location of the remote controller impacts the performance of the system. Test two for determining the system performance impact of network latency. Test three for determining the effect of network jitter on system performance. Test four to measure the performance impact of packet loss. An average system response time is used to compare results from different tests to the default data test-set.

2.) **Create a digital simulation model**
A digital simulations model of the experimental system is created. The digital simulations model is developed with MATLAB® and Simulink®. The system's simulation model entails behavioural, information and communication sub-models. The combination and integration of behavioural, information and communications sub-models provide a digital representation of the experimental system, as presented in the experimental design. The behavioural sub-models in the simulations model include the induction motors, VSDs, local controllers and remote controller.

The induction motors and VSDs are modelled via electrical Simscape™ sub-models within Simulink®. The sub-models for the induction motors and VSDs are created using electric equivalent circuit parameters calculated from datasheet values or measured by the VSD. The response of the motor and VSD sub-model is verified against the response of the physical system to ensure an accurate representation of the motors and VSDs.

The system controllers are modelled as MATLAB® scripts. The local controllers apply the received set point from the remote controller to its connected motor and VSD. The remote controller receives the measured velocities from each of the induction motor, VSD and local controller sub-system. A set point for each of the motors is then determined by the remote controller based on the pre-determined control scheme. The set points are then sent along to each of the motors for execution. The remote and local controllers are verified for the expected response based on the control scheme created for the system.

The communication model describes the network communication between the remote controller and the local controllers. The network communication is modelled based on a mathematical and statistical approach and is created by combining Simulink® and MATLAB® components. The communication sub-model allows for the alteration of network conditions to determine the effects of different network condition on the system. The communications sub-model's response is verified for an accurate representation of physical network communication responses.

3.) **Experimental system**
The experimental system is then constructed according to the system design and architecture defined in the previous steps. Industrial hardware and consumer networking equipment, made available for this study, is built and then connected according to the network architecture defined in the previous step. The experimental system is constructed and connected in the mechatronics laboratory. Three stations are made, each consisting of an induction motor, a VSD and a PLC. The three stations are connected and to the IoT gateway through an industrial network. The IoT gateway with two separate network connections is also connected to a consumer network that connects to the remote controller of the system. A network modifier is also connected to the same consumer network as the IoT gateway and the remote controller.

The network modifier used is WANem, a software package on a computer connected to the network. WANem allows for the network conditions between the remote controller and the IoT gateway to be changed as desired. Software is created for the different devices in the experimental setup. PLC ladder logic is used to create the automation program for each PLC acting as a local controller. The IoT gateway is programmed in Node-RED, allowing for measured speeds of each motor to be logged as well as relaying the information between the local controller and the remote controller. The remote controller is a Python script, running on a computer connected to the consumer network, executing the control scheme.

The operation and capabilities of the experimental system are tested and evaluated for the requirements set for this study. The response of the experimental system is verified against the expected response according to the control scheme created. The network emulator is also tested to allow for modification of network conditions between the remote- and local-controllers.

4.) **Obtain results and perform analysis**
After the construction and implementation of the digital simulation model and the experimental setup is completed, the system performance impact of different network conditions can then be evaluated. Default test data is created for the system using the average system response times for the best case network conditions.

Tests are then administered with different network conditions applied to the digital and physical systems. The different experimental results obtained from the two systems are compared to each system's control group default results obtained. The result comparisons are then used to calculate relationships between the different network conditions and the performance of the IIoT system. The results obtained from the digital simulations model is validated with the results obtained from the experimental physical system. A conclusion on the impact of different network conditions on the performance of the IIoT system is made.

1.5 Chapter outline

The rest of the dissertation is organised as follows: Chapter 2, gives relevant literature, explains terms, concepts and technologies. Chapter 3 explains different system architectures and presents a proposed system architecture for the prototype system's layout, along with any relevant information and theory to the system and its requirements. Chapter 3 also presents the experimental design of both the system and tests to determine the impact of different network conditions on the performance of an IIoT system. Chapter 4 explains the digital simulation model of the experimental system. Chapter 5 presents the experimental system, both the hardware and software aspects thereof. Chapter 6 shows and discusses the results gathered. Chapter 7 concludes the dissertation.

Chapter 2

Literature Review

2.1 Introduction

From the requirement of *smart manufacturing,* or otherwise known as *Industry 4.0* for the topic of this study and the research question of *how do different network conditions impact the performance of an IIoT system* the following topics for the literature review are presented: I4.0, IoT, CPS, I4.0 related industry problems and different network conditions.

This chapter considers and explains the relevant literature, starting topics on I4.0/Smart manufacturing, industrial automation as a cloud service and how network conditions affect an IoT control and monitoring system. Successive terms and concepts relating to I4.0 are explained. The following subsection describes the technologies used in I4.0, followed by a subsection specifying communication network conditions and considerations. It is essential to understand the different network conditions that can exist and their effects on digital communication before implementing them in industrial applications. This chapter aims to provide that foundation.

2.2 Industry 4.0

I4.0 is a term created in 2011 by industry leaders, mainly from German, implying the fourth industrial revolution [8]. In [1], the concept of I4.0 is defined as a combination of IoT, specifically IIoT and CPS. The fundamental principles of I4.0 are described in [8] as the extensive use of the internet as a communications medium, production flexibility and virtualisation of processes. Most I4.0 ready marked products are marked because of the ability to control the product from a remote application through an internet connection. An attractive feature of I4.0 is production flexibility that allows for easy switching between large batch production and efficient small batch production. Another feature of I4.0 is virtualisation, the universal connection of devices and equipment along with digital models integrated into CPSs, allowing for fully-configurable industrial processes [8].

The first industrial revolution is considered to consist of mechanisation through steam and water power. The second industrial revolution consists of using electricity for manufacturing, introducing mass production and assembly line manufacturing. The third industrial revolution is generally seen as the implementation of computers and electronics, alongside the use of information technology for automation [8]. The first three industrial revolutions can be summarised in terms of their enabling technology, namely mechanisation, electricity and information technology (IT), respectively.

The fourth industrial revolution consists of a combination of the Internet of Things (IoT), specifically Industrial IoT (IIoT), and CPSs in manufacturing environments, as shown in the ven diagram in figure 2.1 from [1]. The term " Internet of Things " includes both the Internet of -Services (IoS) and -People (IoP). The time-line and their enabling technologies for each of the four industrial revolutions are shown in figure 2.2. The fourth industrial revolution, I4.0, is an optimisation of the computerisation of the third industrial revolution. The third industrial revolution focused on the automation of single machines, while the fourth industrial revolution focuses on digitisation and integration of an entire organisation [8]. The main design principles for I4.0 is interconnectivity, information transparency and decentralised decisions [8]. I4.0 is the first industrial revolution to be predicted pre-revolution instead of post-revolution, providing both companies and research institutions to steer the future of I4.0 [12].

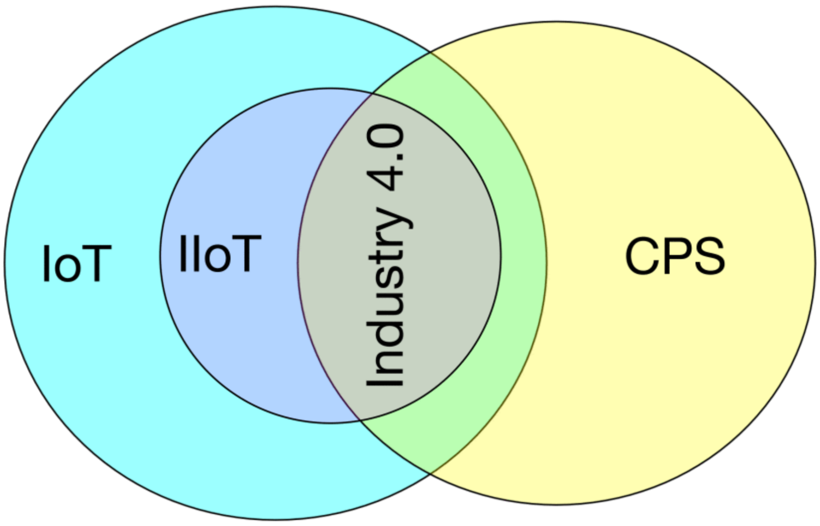


Figure 2.1: IoT, CPS, IIoT, and Industry 4.0 in Venn diagram from [1]

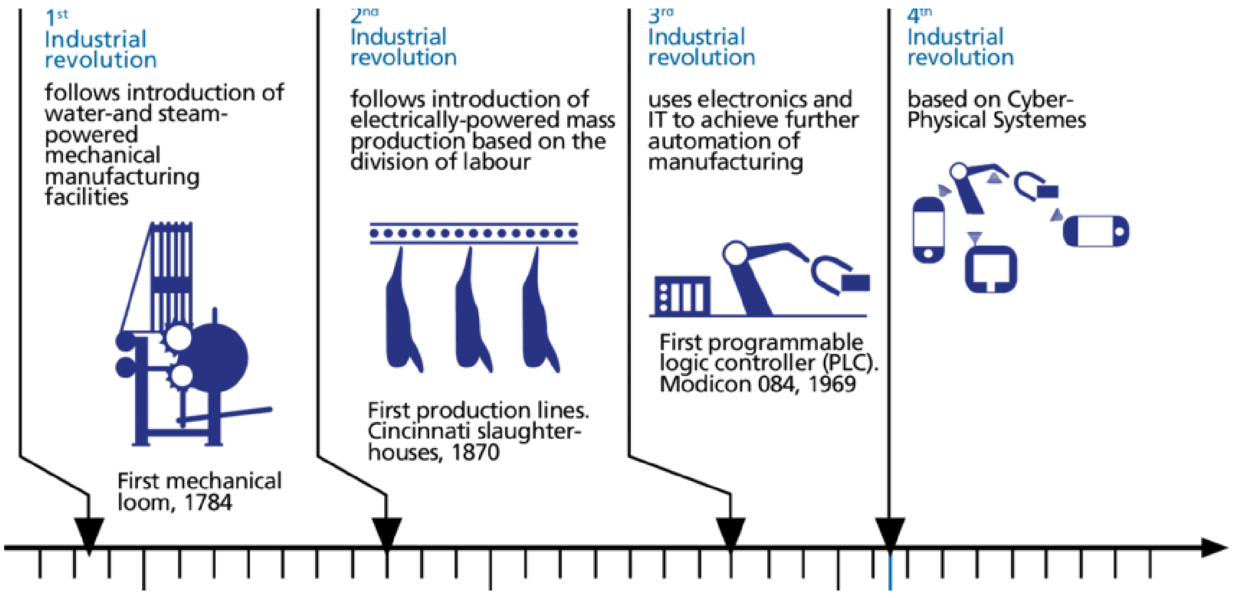

1st Industrial revolution
follows introduction of water-and steam-powered mechanical manufacturing facilities
First mechanical loom, 1784
2nd Industrial revolution
follows introduction of electrically-powered mass production based on the division of labour
First production lines. Cincinnati slaughter-houses, 1870
3rd Industrial revolution
uses electronics and IT to achieve further automation of manufacturing
First programmable logic controller (PLC). Modicon 084, 1969
4th Industrial revolution
based on Cyber-Physical Systemes

Figure 2.2: Industrial Revolution Time-line from (DFKI,2011) [2]

Advocates of I4.0 believe that it will bring core improvements to industrial processes due to the improved communication between humans, machines, and resources [12]. The three main components to I4.0 is *IoT* and *CPSs* within *Smart factories*, discussed in more detail in sections to follow.

I4.0 also focuses on the idea of shifting from centrally controlled to decentralised industrial processes. The ability for the CPSs to perform the task as autonomously as possible, by allowing the systems to make decisions [8]. Information transparency in I4.0 entails a large amount of data collected from machines and sensors. The data gathered is processed into useful information for decision making, increasing performance by simulated optimisation and decreasing downtime by using predictive maintenance [8].

I4.0 relies heavily on the use of the internet and internet services. Most of the “Industry 4.0 ready” marked products are marked as such because of the internet connection to the product and being able to control the product from a smart-phone as example [8].

The use of the internet in I4.0 is not only for a channel to connect devices and sensors but to add additional functionality such as the ability to do predictive maintenance based on gathered data. The data can be collected from different locations and be processed at one single location together. The data gathered from the various sources can be processed and evaluated for advanced diagnostics of all machinery [8].

The manufacturing process is ever increasingly becoming more digitised. Data from the manufacturing processes are collected automatically and in real-time using IoT technologies. The data collected could then be used for different aspects enabling opportunities for smart manufacturing. The manufacturing process data can be analysed with big data analytic to find inefficiencies and apply optimisation to improve said inefficiencies [8]. I4.0 allows businesses and engineering processes to undergo last-minute changes, along with the capability to respond to disruptions and failures rapidly. I4.0 manufacturing processes with incorporated smart machines, storage systems and production facilities can communicate and exchange information. Actions can be triggered based on the information, and elements of the manufacturing process can control each other [13].

I4.0 shares common aspects and goals with terms like: “industrial internet”, “integrated industry”, “smart industry” and “smart manufacturing” [12].

2.3 Components of Industry 4.0

2.3.1 Internet of Things

IoT is described in [1] as “... a computing concept describing the ubiquitous connection to the Internet, turning common objects into connected devices”. Kevin Ashton described the word *Internet of Things* in [14]. He stated that any physical object in the world could be connected to the internet via sensors. The idea of IoT depicts devices or “things” like sensors and embedded devices connected to the internet and using it as a means of communicating information and data. There is, however, no formal definition of the term IoT, and different authors describe IoT differently. Some might focus on the devices’ networking and communications, and others might focus on the embedded devices as things with their limited resources available. IoT can be seen as the most general term of embedded devices and sensors connected to the internet [9].

IoT is a key enabler in I4.0. IoT allows connectivity between different devices, creating a new type of data flow on networks. IoT enables large amounts of devices such as machines, sensors and human-machine-interfaces to connect with more and more being added, while cloud computing supports more significant amounts of data and users [9]. Factory floors are connected, allowing for a centralized platform for data collection from all factories regardless of location [8].

A technological hurdle of IoT is supplying an adequate network connection to an immense number of devices. IoT will add trillions of new devices to the internet [15]. The modern-day internet uses an “end-to-end” principle, where the complexity is dealt with at the endpoints only, and the network is kept very simple. This principle has allowed the internet to be vastly scalable. IoT, however, requires different approaches where the end-to-end policy might not be feasible. IoT has different use cases, from real-time applications where the IP protocol is not suitable due to its unreliability to small devices where the IP protocol can be too complicated for such a system [15].

The “things” referred to in IoT include devices such as radio-frequency identification (RFID), sensors, actuators, mobile phones and many more. The “things” collaborate and with other “smart components” to reach a common goal within a greater system [12].

Along with IoT, a new paradigm of Internet Of People has emerged, where people and their devices are not seen as application end users only but as active aspects within the internet. Machines, devices, sensors, actuators and resources are connected over IoT with People over IoP to create the Internet of everything, IoE [12].

IoT's potential is tremendous in the use of automation due to the connection of a large amount of embedded system, allowing for better utilization of the enormous amount of data they generate and expanding on their limited functionality due to the limited resources of each embedded system. The development of the CPS was an effect of IoT's potential in automation. CPSs are automation systems that connect the physical world to a virtual world. CPSs entails the connection between physical and computer infrastructures [9]. CPSs have discussed further in section 2.3.2

2.3.2 Cyber-physical systems

CPSs are defined in [16] as an automated system with the ability to connect the physical system processes with computing and communication infrastructures. CPSs are therefore seen as the interconnect between a physical world and a digital- or cyber- world.

In [3] by Lee et al., it is stated that "... a CPS consists of two main functional components: (1) the advanced connectivity that ensures real-time data acquisition from the physical world and information feedback from the cyberspace; and (2) intelligent data management, analytics and computational capability that constructs the cyberspace."

CPSs are integrations of computation and physical processes. Industrial embedded automation systems and networks allow for increased physical processes through their access to the cyber world. The origin of CPS comes from mechatronic devices with integrated communication capabilities. CPSs are the fusion between the physical and the virtual world. The connection to the cyber world (computing and communication) enables remote access and control of the physical systems. Remote access to the systems allows for data collection from the system. The data can then be used to decrease the system's downtime by implementing predictive maintenance [16].

A CPS is an embedded system consisting of a control unit, a communication interface, sensors and actuators. The control unit is usually a microcontroller, with the sensors and actuators connect to the physical world. The communication interface is one of the most critical aspects of a CPS, allowing for data exchange between systems. The data from interconnected CPS can be evaluated centrally [16].

A CPS consists of various disciplines. These disciplines consist of control-, software-, mechanical-, and network engineering. A CPS requires a functional interface between embedded software engineering and control engineering. Design considerations and trade-offs are made in CPSs, as a design choice in one domain can negatively impact the other domain [17].

CPSs have a high requirement for reliability as well as predictability. However, the physical world is, as a rule, not entirely predictable. CPSs must therefore be able to work around the unpredictable physical conditions that may occur as well as any subsystem failure that may occur [18].

A common discussion topic regarding CPSs is online optimisation or compensation. Online optimisation works by measuring an actual control system and then adjusting the system's task attributes or schedules to optimise the system [17].

The main difference between IoT and CPS is IoT focusses mainly on the connectivity and communication of devices, while CPS's goal is the integration and cooperation of physical and computational/virtual elements [19]

A CPS is composed of the following two main aspects: The connectivity for the real-time data acquisition from the physical world and reaction from the cyber world, and secondly, the computational abilities, analytics and data management of the cyber world [3]. For the implementation of CPSs, Lee et al. proposed a five-level architecture in [3]. The five levels mentioned can be seen in figure 2.3 along with discretions for each level below:

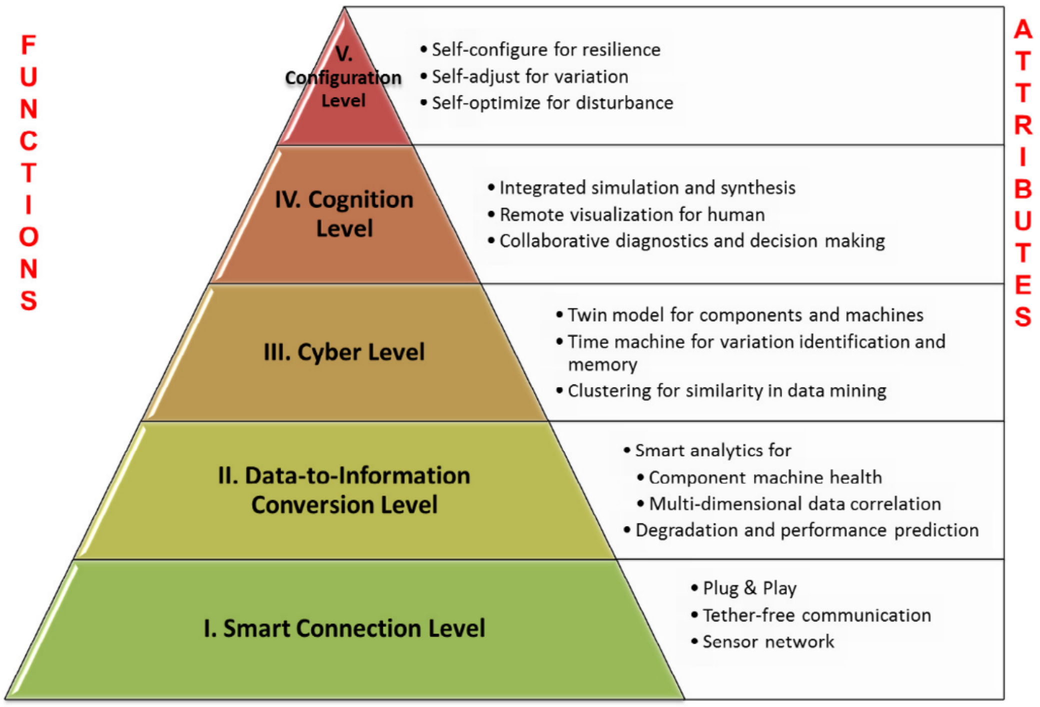

FUNCTIONS
V. Configuration Level
• Self-configure for resilience
• Self-adjust for variation
• Self-optimize for disturbance
IV. Cognition Level
• Integrated simulation and synthesis
• Remote visualization for human
• Collaborative diagnostics and decision making
III. Cyber Level
• Twin model for components and machines
• Time machine for variation identification and memory
• Clustering for similarity in data mining
II. Data-to-Information Conversion Level
• Smart analytics for
• Component machine health
• Multi-dimensional data correlation
• Degradation and performance prediction
I. Smart Connection Level
• Plug & Play
• Tether-free communication
• Sensor network
ATTRIBUTES

Figure 2.3: 5C architecture for CPS implementation [3]

1. Smart connection
   The first level entails the reliable acquisition of data from machines and components. This level involves creating a sensor network by connecting physical devices in order to record data from them. Condition-based monitoring is implemented
2. Data to information conversion
   Using data processing to convert data received from the connected devices. The information can then be processed for predictive maintenance, determining correlations between machines and systems with peer-to-peer comparisons resulting in prognostics and health management (PHM) to predict machines' performance and degradation.
3. Cyber
   The cyber level's goal is to create a central data acquisition point. A central point where the data from all the connected devices to be evaluated. The creation of twin models of the machines and systems will allow performance measurements of individual machines and complete systems.
4. Cognition
   This level involves displaying information acquired at the previous levels to an end-user, enabling diagnosis and decision-making to occur. Visualisation of the information is critical to relay information to an end-user quickly.
5. Configuration
   The final level is the final step from the cyber world back to the physical world. The final level gives CPS the ability to self-configure, adjust, and optimise for variations and disturbances in the system.

2.3.3 Smart factory

A smart factory is a factory that assists both humans and machines in performing their tasks. Background systems within the factory assist both humans and machines with their tasks. The background systems use information from both the physical and virtual worlds. Information from the physical world is, e.g. position and operating conditions of a machine, and information from the virtual world is, e.g. digital documents and simulation models. The information comparison between the physical and virtual world is then used to correct and improve workflows within the factory. A smart factory incorporates the use of IoT and CPSs. IoT is used for information gathering from the physical worlds, and CPS connect the physical and virtual worlds for comparisons, data collection and process optimization. Smart factories are a great example of the implementation of I4.0 as machines and humans are allowed to communicate with each other to perform their expected tasks better [12].

2.3.4 Challenges within Industry 4.0

Challenges existing within smart manufacturing or I4.0 is the lack of computational power to process the vast amounts of data generated efficiently. Stated in [20] is that due to the massive amounts of data IoT will generate, data-centres will face challenges regarding security, consumer privacy, storage management, lack of processing power and data centre networking.

The threat of malicious attacks is ever-present with devices using the internet as a communication medium [8]. Therefore, a cyber-security challenge exists within I4.0 to keep production data safe and private. Cyber-security in the context of I4.0 poses a significant safety concern. The malicious alteration of control signal data sent via the internet could cause damages and even loss of life due to the incorrect operation of industrial equipment.

A technological hurdle of I4.0 is supplying an adequate network connection to an immense number of devices. IoT will add trillions of new devices to the internet [15]. The modern-day internet uses an "end-to-end" principle, where the complexity is dealt with at the endpoints only, and the network is kept very simple. This principle has allowed the internet to be vastly scalable. IoT, however, requires different approaches where the end-to-end policy might not be feasible. IoT has various use cases, from real-time applications where the IP protocol is not suitable due to its unreliability to small devices where the IP protocol can be too complicated for such a system [15].

The challenges that network controlled systems face are network latency, security and multi-user access. The most defining challenge for network control systems (NCSs) is data transmission latency. In comparison, typical systems use private media where the transmission delay can be well modelled. The transmission latency of a public and shared network such as the internet is challenging to model and predict since the data transmission route between two points is not fixed, and the network traffic varies on the different transmission paths [21]. Different network conditions result from different routes and different amounts of network traffic. Changes in network conditions include latency, packet delay variance and packet loss.

Using the internet for communication between controller and actuators and sensors introduces considerable variable delays to the control loop. Network delays are not deterministic as limited bandwidth and overheads in a network yield unpredictable network delays, and variance in packet delays (jitter) are observed [22].

When the system requires a deterministic timing scheme, it may not be achievable by an internet-based control system due to the web-related traffic delay [21].

2.3.5 Industrial automation as a cloud service

In *Industrial automation as a cloud service* by Hegazy et al. [5] industrial automation is introduced as a new cloud service. With the rise in popularity and maturity in cloud computing technologies and services, it is only logical that industrial automation as a cloud service is imminent. The progression of control systems is closely linked to the advancement in computing devices. In the article, a proposed architecture is presented, where the computing function of the automation system is moved into a cloud service. Components such as sensors, actuators and safety/emergency shutdown control cannot be transferred to a cloud service. In the article, digital control algorithms are run on virtual machines instead of using physical hardware in a control room [5].

Using the internet for communication between controller and actuators and sensors introduces considerable variable delays to the control loop. Network delays are not deterministic as limited bandwidth and overheads in a network yield unpredictable network delays, and variance in packet delays (jitter) are observed [22]. Most of the roundtrip delays between the cloud controller and the controlled processes are absorbed into the industrial application sampling period because the sampling period is usually larger than the roundtrip delay. The roundtrip delay varies with tens and a few hundreds of milliseconds, while an average industrial process's sample time ranges between a few hundred milliseconds and several seconds. When the roundtrip delays are smaller than the sampling period, no effect is perceived in the control loop as one action per sampling period will still be executed. Due to the random and varying nature of internet delays, the roundtrip delays seldom go beyond the sampling period and impact the control loop. Delay mitigations are therefore required to combat this effect [5].

In [5], the roundtrip delay in the control loop is simplified to controlling a process with dead-time. A delay compensator is coupled to combat the dead-time. The article presents an experimental evaluation of industrial automation as a cloud service using a real-life plant's physical model. The experimental assessment proved effective to control an industrial plant of a remote connection and switch over a failsafe controller's ability. Result obtained showed effective control of the plant by a remote controller cloud service. Comparing results from the remote cloud controller and local controllers showed that the cloud controller performed similarly to the local controllers, even when varying delays have been injected into the cloud-based controller [5].

2.3.6 Real-time control in IIoT with different network delays

In the thesis *Real-time Control in Industrial IoT* by Didic et al. [9], the impact of implementing cloud control services in a closed-loop control is investigated. An experimental setup is created where a remote controller is used in controlling a process. The controller is implemented with polling and non-polling approaches, with and without delay mitigation techniques. The impact of different network delays and jitter on the system was tested for the various remote controller approaches.

The prototype system tested produced adequate results when the network latencies were smaller than the sampling period. The experimental system used included short sampling periods of around 16 ms. The quick sampling periods made for a system that could not tolerate large changes in response time. The addition of network delays contributed to the large change in response times. The addition of delay mitigations to the remote controller proved useful for small compensations within the system.

2.4 Industry 4.0 enabling technologies

2.4.1 Cloud computing

The `Open Glossary of Edge Computing` by `The Linux Foundation`® [23] defines cloud computing as: "A system to provide on-demand access to a shared pool of computing resources, including network, storage, and computation services. Typically utilizes a small number of large centralized data centres and regional data centres today."

Cloud computing provides a solution for an on-demand processing service for the vast magnitude of data streams created by IoT devices. Many of the IoT applications will have huge data storage requirements along with large amounts of processing power required to enable real-time operation ideal use cases for Cloud computing [20].

Cloud computing is an on-demand service that is elastic, able to be rapidly scaled up or down, and device-independent. Cloud computing can deliver resources such as computers, networks, storage and servers, seen as Infrastructure as a Service (IaaS), as well as options for software through cloud computing, seen as

Software as a Service (SaaS) [20]. SaaS gives consumers the ability to run software on cloud infrastructure. The software is then available to various client devices either through a web browser or a specific application interface [24]. IaaS supplies consumers with various computational resources over a network. Consumers control the applications on the infrastructure, but not the infrastructure itself [24].

Mell and Grance propose five essential characteristics for a cloud computing model in [24]. The model states that cloud computing is an omnipresent and convenient on-demand access to shared computational resources [24].

- *On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- *Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- *Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data-center). Examples of resources include storage, processing, memory, and network bandwidth.
- *Rapid elasticity.* Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
- *Measured service.* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

2.4.2 Fog computing

Cisco® defines fog computing in [25] as *"...a standard that defines how edge computing should work, and [which] facilitates the operation of computing, storage and networking services between end devices and cloud computing data centres."* Fog computing is also described as a combination of peer-to-peer networking and wireless sensor networks (WSN) [7] [9].

Cisco® also defines edge computing in [25] as *"[bringing] processing close to the data source, and it does not need to be sent to a remote cloud or other centralized systems for processing. By eliminating the distance and time it takes to send data to centralized sources, we can improve the speed and performance of data transport, as well as devices and applications on the edge."*

Fog computing enables computational power to the network's edge and facilitates smart sensors and devices in cloud-based applications [9]. The swift increase of IoT platforms increases the use of fog computing by adding fog nodes to increase connected coverage area [9].

Fog computing enables higher computational power for devices with limited computational power, such as mobile devices and embedded systems. Fog servers bring advantages of cloud computing to the edge of the network and benefit from decreased network latency to end devices compared to cloud computing [9]. The main differences between fog and cloud computing are summarised in table 2.1. Security and privacy of data is a notable problem for cloud computing, but less so for fog computing due to data being kept local instead of moved between local and off-site servers [9].

Advantages of fog computing for IoT include scalability, supporting latency-sensitive applications by being no more than one network hop away from the user application, and adding more computational power to the otherwise computational stricken IoT and embedded devices [9].

Table 2.1: Comparison between Cloud and Fog(taken from [7])

Requirement	Cloud computing	Fog computing
Latency	High	Low
Delay Jitter	High	Very Low
Location of server nodes	Within the Internet	At the edge of the local network
Distance between the client and server	Multiple hops	One hop
Security	Undefined	Can be define
Attack on data en route	High probability	Very low probability
Location awareness	No	Yes
Geographical distribution	Centralized	Distributed
Number of server nodes	Few	Very large
Support for Mobility	Limited	Supported
Real-time interactions	Supported	Supported
Type of last mile connectivity	Leased line	Wireless

2.4.3 Cyber security

IoT can increase the productivity of enterprises, but with the risk of security, exploitations treats by hackers and cyber-criminals. IoT devices readily available can contain vulnerabilities [20]. The vulnerabilities are due to lack of encryption of data, vulnerable web interfaces, and deficient software protection [20]. Fog computing is also described as a combination of peer-to-peer networking and wireless sensor networks (WSN) [9]. Fog computing enables computational power to the network's edge [9].

Large interconnected organisation systems pose increased security risks. Cyber-criminals and hackers exploit vulnerabilities in any of the system components. Cyber-criminals' activities can disrupt production and even cause physical damage to systems and resources [13]. A recent cyber-security incident is the “Stuxnet virus” that targeted and attacked vulnerabilities in Supervisor Control, and Data Acquisition (SCADA) systems used in industry [13].

The paradigm of I4.0 with more connected devices increases unexpected security vulnerabilities. Cyber-security must be able to adapt rapidly and with enough agility for use in I4.0 application to not hinder the operation of the systems involved [13].

2.4.4 Big data

Big Data is, in general, described as an extensive amount of data in any structured form that is too vast for typical database software tools. However, Big Data usefulness is in interpreting the data and the ability to uncover hidden information within the vast amounts of data. The main goal of Big Data is not a large volume of data but rather a large amount of valuable information that can to be translated into business advantages. The information from Big Data systems must be analysed promptly to be relevant [26].

There are various sources for big data in the manufacturing process. Data is generated from each stage of the process, such as the design, manufacturing, maintenance, repair and overhaul. Data from the manufacturing stage comes from various sources: environmental, material and products, and factory floors [26].

Four characteristics define big data. The four characteristics are volume, variety, velocity and value. Volume is the enormous amount of data acquisition; variety refers to the different types of data in the datasets;

velocity is the rapid rate at which data is generated; value is the significance of the extracted information derived from the data collected [26].

Industries can not afford to ignore data collections from the manufacturing process. The data collected helps optimise efficiencies within the manufacturing processes, reducing waste and improving quality [26].

2.4.5 Digital twin

The term digital twin refers to a digital replica of a physical/real-world entity. Digital twins operate with data transfer from a physical entity to the virtual space allowing both entities to exist simultaneously [27]. Digital twin allows for improved cyber-physical integration. This cyber-physical integration is regarded as the biggest hurdle or bottleneck in smart manufacturing [26].

A digital twin consists of a virtual replica of a physical entity. The virtual replica contains various models describing different attributes of the physical entity. Models include data, functionality, and communication of the physical entity. The Data model describes structure and geometry attributes, and the functional model describes processes and behaviour. The integration of the models produces the virtual replica of the digital twin [28].

The purpose of a digital twin is to simulate both entities' behaviours by creating a virtual model for a physical object. The virtual models receive sensing data from the physical entity. The virtual model then uses the data received to know the state of the physical entity and analyse any dynamic changes that may occur in the physical entity. The physical entity will then react to any optimisations from the virtual simulation. The cyber-physical closed loop of the digital twin could optimise an entire manufacturing process [26].

The Digital twin consists of three components, as shown in figure 2.4. The components are the virtual model, the physical entity and the connection data between them [26].

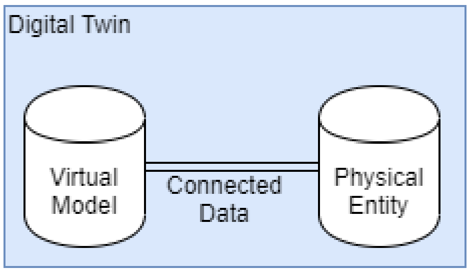


Figure 2.4: Digital twin representation

Manufacturing applications of a Digital twin as stated by Qi and Tao [26]

1. Digital twin Based Production Design
2. Smart Manufacturing in Digital twin Workshop/Factory
3. Product Digital twin for usage Monitoring
4. Digital twin as an enabler for Smart maintenance, repair and overhaul (MRO)

The digital twin is therefore used in the whole lifecycle of manufacturing. The digital twin combines and gathers data from the whole lifecycle of production, promoting efficient and effective manufacturing. A digital twin is used in product design, where the virtual model is used to test and evaluate the designer's expectations. The digital twin allows for rapid changes in the design and advances fault finding within design constraints [26].

Once the product's design has been completed, the design is sent to the "smart factory" to start production. The virtual model of the workshop is used to simulate and evaluate different manufacturing strategies. When the physical production of the product starts, the virtual model is updated to act more accordingly to the physical workshop through data exchange. Changes and faults are then evaluated in the virtual environment, and a new optimised solution is made and pushed to the physical production workshop [26].

After the product has been produced, a digital twin of the product can then be implemented. Data transfer between the virtual and physical product enables additional functionality to the product. This includes Value-added services via software updates to the product and prediction of use cases, and monitoring its performance. The product's remaining lifetime can be estimated along with a predictive maintenance schedule to increase the product's efficiency and lifetime. The digital twin allows for traceability of products and any specific fault to batch production, the data from that batch can then be analysed to estimate the fault/failure and ensure the correction thereof. Feedback from the product and customers can then be used to improve further and develop the product [26].

Because of the monitoring of the production system, a predictive maintenance scheme can be implemented to increase the workshop's lifetime and efficiency and equipment. The digital twin improves the fault-finding capabilities when changes in the data occur. Therefore a digital twin system can aid in the repair process of a workshop. Downtimes of the workshop can be managed, and decreased [26].

According to authors Qi and Tao in [26]: *"Collecting and analysing a large volume of manufacturing data to find laws and knowledge, has become the key to smart manufacturing. Meanwhile, the digital twin breaks the barriers between the physical world and the cyber world of manufacturing."*

2.4.6 Virtual Commissioning

Unlike real commissioning of a manufacturing system, a real production plant and a real controller are required. Virtual commissioning is done with a virtual production plant simulation model and a real controller. Virtual commissioning benefits are reduced efforts in debugging, and correction of real commissioning [4].

Virtual commissioning enables the full verification of a manufacturing system by performing a simulation involving a virtual plant and a real controller. Therefore, a virtual plant model is required that accurately describes the actuators and sensors [4]. With virtual commissioning, design and operational flaws can be identified and addressed before a real plant is commissioned. As shown in [4] and figure 2.5.

1. Real commissioning
   Real plant and real controller
2. Hardware-in-the-loop
   Virtual plant and real controller
3. Reality-in-the-loop
   Real plant and virtual controller
4. Constructive commissioning
   Virtual plant and virtual controller

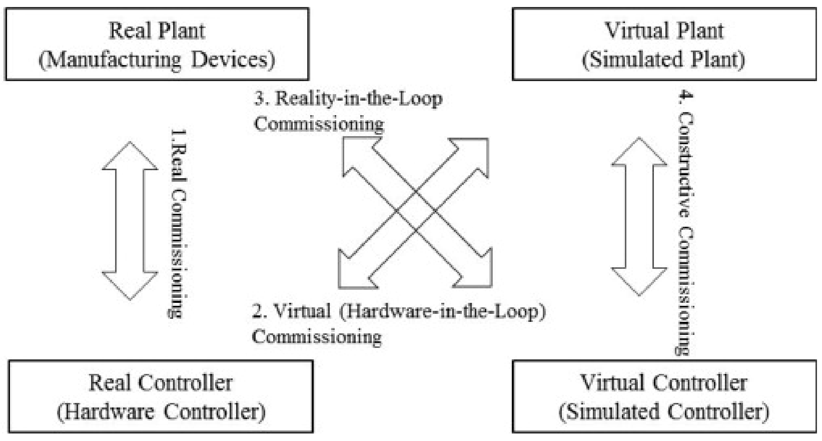


Figure 2.5: Commissioning configurations of a manufacturing system [4]

2.4.7 Electronics

According to Sundmaeker et al. in [29] the electronics contributing to I4.0 and industrial IoT is programmable logic controllers (PLCs), along with their open-source counterparts (OpenPLC).

A usual PLC is comprised of five primary components [30]:

1. Rack
2. Power supply
3. Central Processing Unit (CPU)
4. Inputs
5. Outputs

The rear backplane that connects the PLC components and allows for communication between them is called the **rack**. The **power supply** provides the PLC components with a regulated voltage. The **CPU** processes the information from the inputs models, and according to the program written onto the CPU, determines the appropriate outputs based on the inputs received. The program on the CPU is performed in a loop. **Input modules** read field sensor data to the PLC. Inputs can either be analogue (continuous) or digital (discrete). **Output models** send signals to other devices and actuators. Output values are also either digital or analogue.

For a PLC to have sufficient control over connected devices, a PLC must be real-time. A definition for real-time can be seen as “any information processing activity or system which has to respond to externally generated input stimuli within a finite and specified period” [30]. PLCs follow the IEC 61131-3 standard, which defines the underlying software architecture and programming languages for PLCs [30].

OpenPLC is an open-source Programmable Logic Controller architecture that is based on easy to use the software. OpenPLC is fully open-source and open hardware. Hardware used in OpenPLC is inexpensive, allowing for a flexible PLC for smooth implementation of industrial IoT and I4.0 by small to medium-sized enterprises (SMEs) [29].

The OpenPLC project started by creating a conceptual open-source design architecture. The proposed architecture mimics that of an actual PLC. The concept architecture entails a modular system with an RS-485 bus for communication between components [30]. The concept hardware was done with an AVR ATmega2560 microcontroller as the CPU for its open-source connection with Arduino [30]. The OpenPLC has since expanded to run on other open-source hardware like the Raspberry Pi [31].

2.5 Industrial automation

According to Didic et al. [9] *"Automation or automatic control means using a specialised computer system for controlling physical processes, equipment, machinery and other applications while reducing human intervention"*. Automation is performed through various control loops. An example of a closed-loop control can be seen in figure 3.1. A closed-loop control system receives feedback for error correction while an open control loop does not [9]. The advantages of automation are increased production efficiency resulting in increased quality with better-optimised resource control [9].

The typical automation architecture is divided into five layers, L0 through L4 [5]. A typical control architecture can be seen in figure 2.6 [5]. Level0, L0, at the bottom, is where the actuators and sensors are located. L1 contains control elements such as microcontrollers and PLCs that control the devices in L0. The connection between L1 and L0 is called the field-level network. The next level, L2, is where process variables and control loops are monitored; this is done through Human-Machine Interfaces (HMI) or Supervisory Control And Data Acquisition (SCADA). The control network is the connection between L1 and L2. Level 3, L3, is the manufacturing execution system; this system is responsible for coordinating the whole plant's control. The final and top-level, L4, is the enterprise management level where enterprise resource planning is done. Components in the architecture solely communicate to components on the same or adjacent layers [5].

Industrial automation systems are usually complex, using both legacy and modern subsystems. Legacy systems usually pose the problem of being difficult to integrate with new systems. Industrial automation systems need to adhere to strict requirements for correct industrial operation. [9]. Industrial automation systems are built to be robust, durable, reliable, often fault-tolerant, real-time and safety-critical, relying on strict industrial standards [9].

With the increased use of IoT and cloud computing, the emergence of a cloud-based automation system is sure to follow. The trend is for higher levels, such as supervision and monitoring, to be moved to cloud computing. This shift is logical as L3 and L4 levels have fewer constraints than the lower levels, L1 and L2 [9]. The difficulty with moving lower levels to the cloud is the constraint for the lower levels' real-time operation. The main problem for using cloud computing and an internet connection is introducing unpredictability into systems that require a sense of predictability [9].

However, the properties of fog computing better suits the requirements for industrial manufacturing processes. These properties include low latency, low jitter and improved security. Fog computing allows for a higher sense of predictability compared to cloud computing due to fog computing using local networks and not the internet, thus making fog computing more favourable for industrial process applications than cloud computing, especially for the lower levels of automation [9].

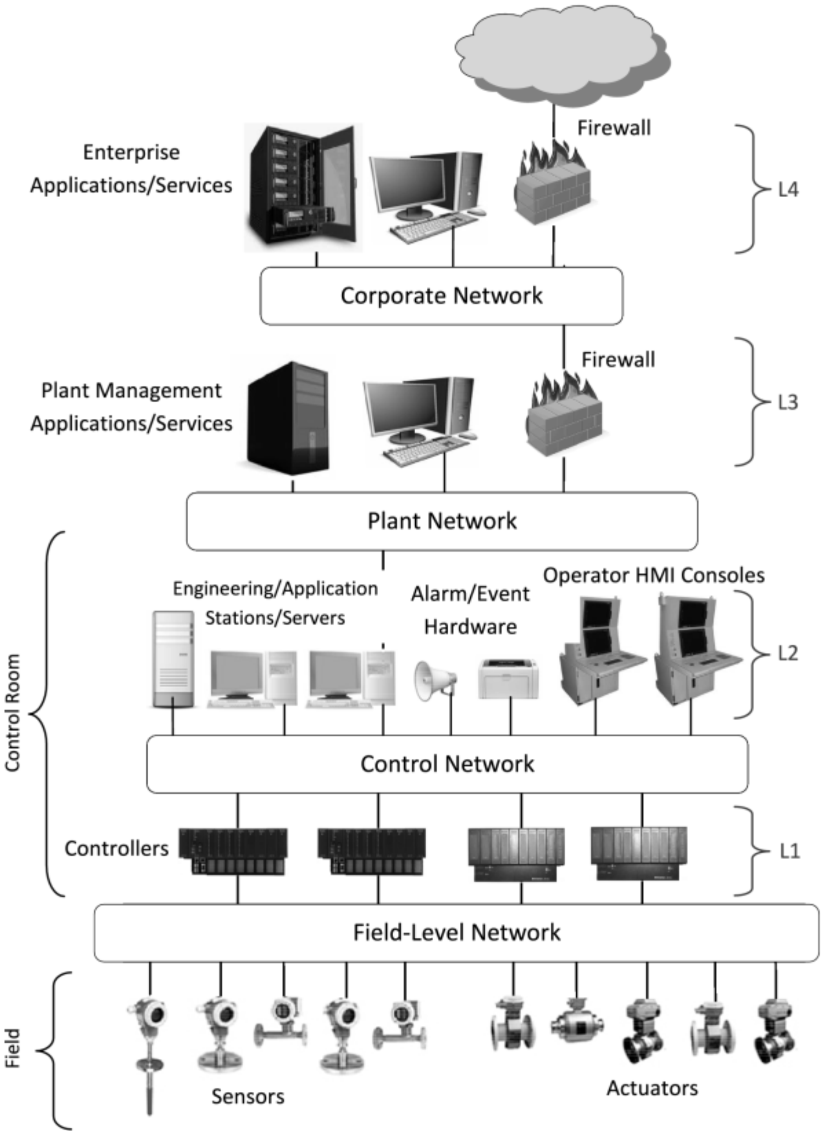


Figure 2.6: Typical automation system architecture [5]

2.6 Network conditions and considerations

Various characteristics can describe a network. Some characteristics of a network are packet latency (`delay`), packet delay variance (`jitter`), packet loss, packet reordering, packet duplication, packet corruption and bandwidth.

A network's delay specifies how long it takes for a bit of data to travel across the network from one node or endpoint to another, usually measured in milliseconds. Jitter is the variance of delays on a network and is described in more detail below. It is also possible for packets to be lost, reordered, duplicated or corrupted on a network. This is due to packet transmission over a network with high amounts of jitter or an unreliable network.

2.6.1 Bandwidth

A characteristic significant to network performance is bandwidth. Bandwidth can be described in two different ways, bandwidth in bit per second and bandwidth in Hertz. As stated in [22]: Bandwidth in bits per second is the number of bits per second that a channel, a link, or even a network can transmit.

Bandwidth in Hertz is the range of frequencies contained in a composite signal or the range of frequencies a channel can pass.

There is a direct relationship between bandwidth in bits per second and bandwidth in Hertz, as the one increases so should the other [22].

2.6.2 Latency (Delay)

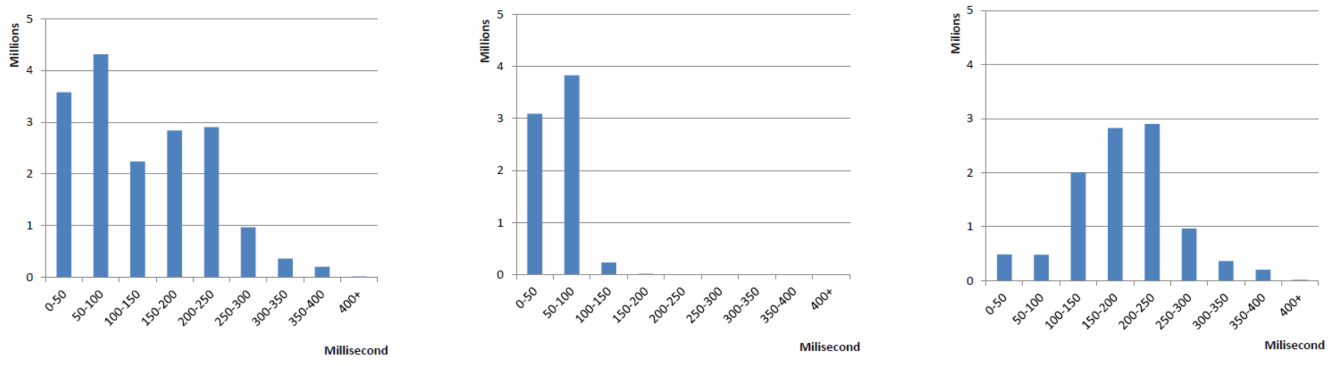
Latency is the time for an entire message to arrive at its destination from the moment that the first bit is sent. Latency is made out of four components, namely propagation time, transmission time, queuing time and processing delay [22].

Latency = propagation time + transmission time + queuing time + processing delay

- **Propagation time** is the time for a bit to travel from its source to its destination.
- **Transmission time** is the time it takes to sent a message. Transmission time = (Message size)/Bandwidth.
- **Queuing time** is the time a device needs to hold a message before it the device can process the message
- **Processing delay** is the time is takes for the device to process the message received

Bandwidth-Delay Product:
The product of bandwidth and delay defines the number of bits that can fill-up a link [22].

A study by Mercan et al. measured packet delay time and variation on the internet in different countries over ten days. General internet packet delay times occurred mostly between 0 and 300 ms. Packets mainly were experiencing a delay of less than 100ms for a large portion of packets travelling in the same country, and delays of between 100-300 ms were measured for packets destined to another country than their origin [6]. The study concluded that delay values were mostly squeezed in a small area. Distribution was unimodal in most cases, but some of them had multimode, asymmetric and right-skewed. Histograms of their measurements can be seen in figure 2.7 [6].



a. General Delay Values b. Intra-national Delay Values c. International Delay Values

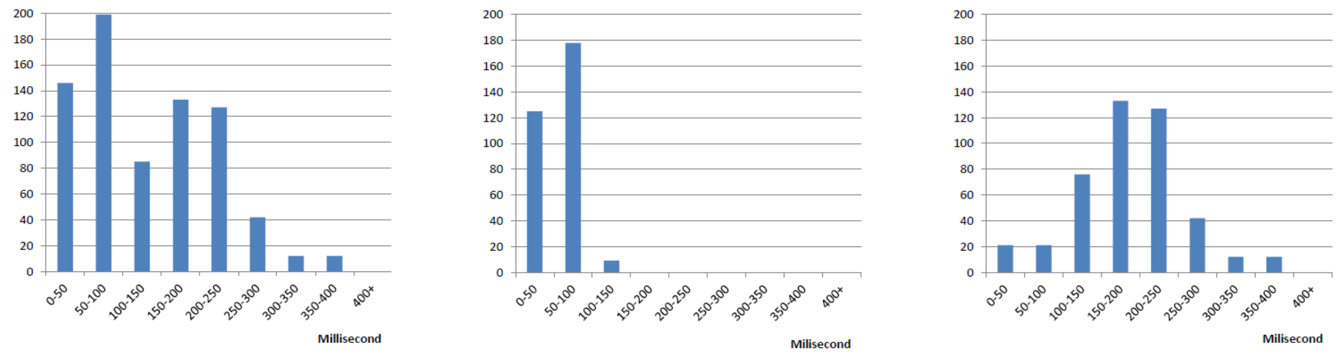


Figure 2.7: Internet packet delays and path averages from [6]

2.6.3 Packet Delay Variance (Jitter)

The “Open Glossary of Edge Computing” created by The Linux Foundation® [23], defines Jitter as “*The variation in network data transmission latency observed over a period of time. They are measured in terms of milliseconds as a range from the lowest to highest observed latency values recorded over the measurement period. A key metric for real-time applications such as VoIP, autonomous driving and online gaming assumes little latency variation is present and sensitive to this metric’s changes.*”

Jitter is recognised as an essential phenomenon that degrades communication performance, particularly in real-time services over a network connection [32]. Jitter can be caused by packets taking different network paths to the destination node. Packets might take different network paths to try and avoid congested or failed network paths. Jitter is, however, usually caused by varying queuing delays introduced at different network nodes [32]. The magnitude of Jitter is affected by the size of a network. Usually, a more extensive network means a more substantial jitter variance.

2.6.4 Packet loss

Packet loss occurs within a network when a router receives multiple packets at once. The packets are stored in a buffer before processing. The buffer has limited space. When the buffer is full, packets received after that are dropped. The impact of packet loss on a network is that the packet needs to be resent. A congested network packet loss can result in more packet loss due to packets being resent [22].

2.6.5 Network emulation

WANem, an open-source WAN (Wide Area Network) Emulator, is a software package developed to provide an authentic experience of a wide area network or internet over a LAN (Local Area Network). WANem allows an application gateway to simulate network characteristics like network delay, jitter packet loss, packet reordering, packet corruption and packet duplication. Documentation, tutorials and source code for WANem can be found at `http://wanem.sourceforge.net/`

2.6.6 Network workloads

Network workloads or traffic can come from a mix of various sources. Different workloads on a network change the network in question's behaviour. Introducing traffic to a network can change the latency and jitter of a network due to bandwidth limitations [33].

A tool such as D-ITG (Distributed Internet Traffic Generator) produces a synthetic realistic network workload to be introduced into a network. The traffic on a network can be increased as desired, and the effect thereof can be measured and analysed [33]. Homepage, documentation and the D-ITG software can be found at `http://www.grid.unina.it/software/ITG/`

The generation of network workloads is used in different network research fields, including evaluating the performance of network and network devices [33].

2.6.7 Real-time protocol (RTP)

The real-time transport protocol is created to manage real-time traffic on the internet. RTP is mainly designed for use in multimedia applications. Some key characteristics of RTP [22]:

- RTP does not have a delivery mechanism
- RTP must be used with User Datagram Protocol (UDP)
- RTP stands between UDP and the intended application
- RTP is located in the application layer and UDP in the transport layer. The socket interface is located between RTP and UDP.

2.6.8 Time sensitive network

Time-Sensitive Networking (TSN) is a communication standard based on Ethernet by the Time-Sensitive Networking task group from the Institute of Electrical and Electronic Engineers (IEEE 802.1) hard real-time communication. The goal is synchronised communication with low latencies and packet delay variances.

The standard provides time synchronisation, latency and bandwidth services for time-sensitive services across bridged and routed LANs. This is done by defining packet format, synchronisation and teardown protocols from Real-Time Transport Protocol (RTP) and IEEE 802.1BA-2011 Audio/video bridging (AVB) protocols.

2.6.9 Profinet IRT

PROFINET IRT is created by Siemens© as part of their PROFINET protocol for Isochronous Real-Time (IRT) communication. PROFINET IRT requires special hardware support on both the master and slave devices. Siemens©t provides the hardware support on controllers and devices such as drives and any infrastructure components requires such as switches [34].

PROFINET IRT uses a scheduling mechanism to set up synchronous and asynchronous parts for a communication cycle. The typology affects how the scheduling is handled, but the detail of how the scheduling of PROFINET IRT operates is proprietary information to Siemens© [34].

2.6.10 EtherCAT

EtherCAT is developed by Beckhoff® to offer a high real-time performance networking protocol. EtherCAT slave devices require specialised hardware for the short packet forwarding required by EtherCAT. The master device, however, can be implemented with standard hardware [34].
EtherCAT works by sending telegrams in a full-duplex manner from the master. The telegrams are reflected at each network segment, allowing all node on the network to read and transmit data to the telegram as it passes by, allowing for a constant delay in each device. EtherCAT can carry standard Ethernet traffic along with the EtherCAT traffic [34].

2.6.11 Requirements of network infrastructure for internet-based control

According to S.H. Yang in [21], the six requirements for the ideal network infrastructure for internet-based control is:

1. Real-time transmission
2. Reasonably reliable transmission
3. Time-out notification
4. Priority based transmission
5. Time synchronisation
6. Penetrating firewall

The ideal network infrastructure stated above takes into account internet transmission behaviours. With the ideal network infrastructure, the control system still requires a different control scheme with time delay mitigations for the application of internet-based control systems.

2.7 Critical review

The literature survey and study discussed in this chapter to gain insights into *smart manufacturing* and *Industry 4.0*. Specific attention has been given to industrial control with IoT and how different network conditions affect the control of the system.

From [5], the feasibility of industrial automation as a cloud service has been positively stated with an experimental study controlling an industrial plant with a remote cloud controller. A statement made in [5] is that any network latency smaller than the sampling period of the process will not affect the control loop because the controlled process will still receive one action per sampling cycle.

In [9], the effects of network conditions on a networked closed-loop control IoT system was evaluated. The experimental setup made use of consumer-grade hardware and a single controller, control loop with a shot sample time. From [9], the delay mitigations used, exponential moving average and double exponential smoothing model are presented and explained. Both delay mitigation proved effective with small network disturbances of the experimental setup. Also presented in [9] is network emulation done through WANem, a network emulation software package. WANem allows for the alteration of network conditions and behaviours between two network nodes.

Network condition assessed for evaluation due to the likelihood of occurrence within an IoT control system is latency, packet delay variance and packet loss. Latency is the time delay from when a message is sent over a network to when it is received at its destination. Jitter or packet delay variance is the variance in latency that can occur in network communication. Packet loss is the loss of network packets that occur when network buffers overflow and new packets to the buffer are discarded, resulting in the packet to be resent [22].

Due to the non-deterministic nature of internet communication and the characteristic of the internet being able to transmit over different network routes, both latency and jitter occur in different degrees, degrading the communication between network nodes [5]. At the same time, packet loss can happen in a network where large amounts of data are transmitted over a network [22]. A large likelihood of packet loss is possible due to a large amount of data being transmitted in an IoT control and monitoring system. A recommendation for good quality of service in network communication made by Cisco®, an industry-leading manufacturer of networking and telecommunication hardware is that latency should remain below 300 ms for a round trip, jitter shouldn't go over 30 ms, and packet loss should be less than 1%. However, the recommendation is made for a good quality of service for consumer services over a network and not industrial services.

2.8 Conclusion

After an analysis of I4.0, its underlying components and different communication network conditions and considerations, it becomes evident that the effects of varying network conditions are an essential factor in the implementation of I4.0. I4.0 promises to enhance current manufacturing environments and methods in the industry. While literature provided a thorough background and understanding of I4.0 and the different network conditions and considerations, a clear understanding of the effects of varying network conditions on I4.0 implementations is unknown. This dissertation aims to provide an understanding of how varying network conditions affects the application of I4.0.

Chapter 3

Experimental design

3.1 Introduction

From chapter two, it is evident that different network conditions can have a noticeable effect on the performance of an I4.0 application using IoT and CPS. The next step is to design an experiment to determine the effects of network conditions on the performance of an IIoT system.

This chapter presents the experimental design for determining just that. An experimental IIoT system is designed to be subjected to different networking conditions in order to measure the performance impact on the system.

Firstly, diverse system architectures are explored of a typical closed-loop control system, a network-based closed-loop control system, and an internet-based closed-loop control system. The proposed system architecture for the experimental setup is then given and explained, followed by the network architecture for the experimental setup based on the proposed system architecture. The goal for the experimental system's architecture is to add a network-attached remote controller to a system of typical closed-loop control systems. Different delay mitigation structures are then defined for use in the experimental setup. The control scheme for the remote-controlled system of motors is then presented. Finally, the research approach is given for testing the effects of varying network conditions on a motor based IoT control and monitoring system, with and without delay mitigations implemented.

3.2 System architecture

3.2.1 Typical closed-loop control

A typical control architecture used is a closed-loop setup, consisting of four parts structured in a centralised control structure, as shown in figure 3.1. The four parts include a controller, an actuator, a process and a sensor. The sensor produces a value based on the process and relays the information to the controller. The controller then analyses the information and provides a control signal to the actuator based on its analysis. The actuator then performs an action based on the control signal received from the controller. The action affects the process, and the loop is repeated. The controller, actuator and sensor must be wired in a point-to-point fashion and be physically located in close proximity. An ideal closed-loop control system presents negligible signal loss with no time delay in the signal transfer but can be expensive to implement.

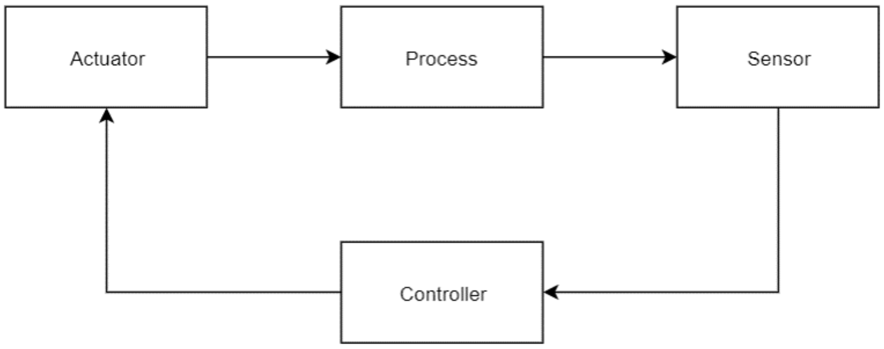


Figure 3.1: Closed-loop control system

3.2.2 Network and internet based closed-loop control systems

A network control system is in principle the same as a closed-loop control. The only difference is the communication to and from the controller, the input and output from the controller is done over a network connection and not a physical connection. NCS enables an efficient but expensive centralized control system [9]. NCS is control *over* a network and not control *of* a network, an important point to take note of [21]. NCS can use existing and shared communication networks, reducing cost and allowing access to other points within the network. The use of an existing and shared network communication adds additional factors for real-time operation such as signal delays and delay variances and packet loss due to over-congestion of the network [21]. A typical network control system can be seen in figure 3.2.

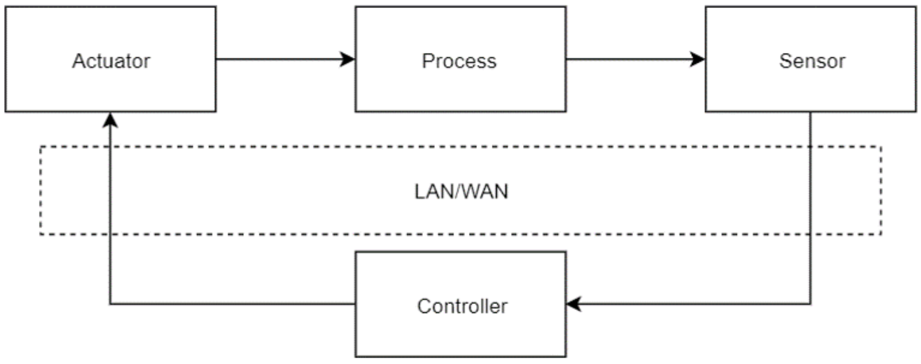


Figure 3.2: Network-based control system

The shared network connections can be either local or global. Internet-based Control Systems (ICS) are NCS that uses a global connection, the internet, as a shared communication network [21].

The main advantage of using a ICS for industrial manufacturing is the ability to place control engineers and specialists in one location and monitor and control the plant in various different locations. The accessibility to the control and monitoring of plants are increased as access points to the internet is vastly available. Internet-based control system's goal is to enhance conventional computer-based control systems and not to replace them. The ICS adds an internet layer to the control system [21].

The challenges that NCS and ICS face are network latency, security and multi-user access. The most defining challenge for NCS and ICS is data transmission latency. In comparison, typical systems use private media where the transmission delay can be modelled accurately. The transmission latency of a public and shared network such as the internet is challenging to model and predict due to the data transmission route between two points are not fixed, and the network traffic varies on the different transmission paths [21]. Different network conditions are produced by different routes and different amounts of network traffic. Changes in network conditions include latency, packet delay variance and packet loss.

Usually, a system consists of components working in unison to achieve a common objective. A control system's goal is to maintain a relationship between inputs and outputs of the system with regards to different disturbances in the system [21].

A requirement specification is needed for the design of a system. The requirement specification entails the objective of the system as well as any constraints on the system. Constraints can either be physical or safety-related. Constrains will limit the amount of design options and a trade-off should be made between any functional goals and constrains if the constraints hinder the objective of the system. An internet-based control system should include process monitoring along with control objectives of the system [21].

When the system requires a deterministic timing scheme, it may not be achievable by an internet-based control system due to the web-related traffic delay [21].

3.2.3 Proposed system architecture

The goal is to add a remote controller to a typical closed-loop control system as shown in figure 3.1. A closed-loop control where the controller is connected through a network is shown in figure 3.2. When a remote controller is used latencies, jitter and packet loss is introduced between the actuator, sensors and controller. The proposed architecture is shown in figure 3.3. The remote controller is either connected through a local (LAN) or a global (WAN) network. In the case of a local network connection, the remote controller is seen as a local cloud or fog node.

The control architecture proposed is that of a bilateral controller, as seen in figure 3.4, where local controllers are placed at the process/plant side, and a remote controller is placed at the operator side with the controllers connected through a network connection. The local controller is responsible for the regular operation of the process, including fail-safes of the process. The remote controller is then used for monitoring and changing parameters of the operations and control across interdependent systems or processes.

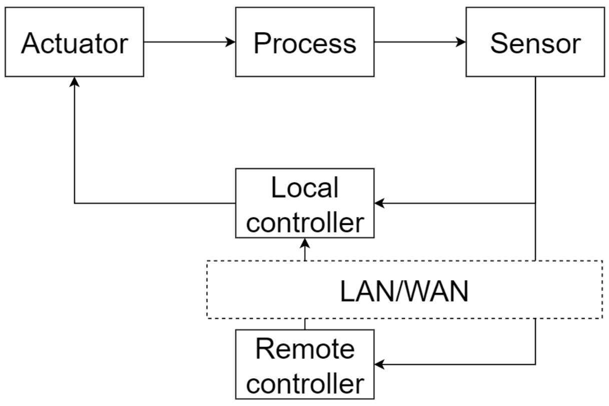


Figure 3.3: Proposed remote controller closed-loop control system

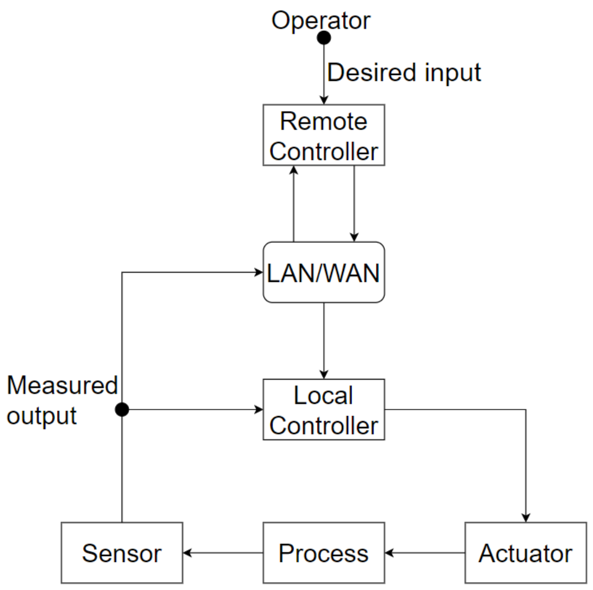


Figure 3.4: Bilateral controller control structure

3.3 Network architecture

The network architecture is designed to resemble the proposed bilateral closed-loop control architecture shown in figure 3.3. The system emulates an industrial plant with three electric motors. Each motor is controlled locally by a Variable Speed Drive (VSD) and a Programmable Logic Controller (PLC). The individual local controllers are then connected to a remote controller through an industrial IoT gateway. The remote controller is responsible for control over the whole interconnected system. The system's networking diagram can be seen in figure 3.5. Measured motor velocities by the VSDs are sent via an industrial ethernet connection to the PLCs. The PLCs, in turn, send the velocity data to the IoT gateway. The remote controller receives the velocity data from the IoT gateway, processes the data and generates speed set points for each motor. The reference set points are sent back to the IoT gateway to be relayed to the PLCs and VSDs for execution on the relevant electric motor. The IoT gateway used allows for two separate network connections. The IoT gateway's first connection is to an industrial ethernet connecting to the PLCs and VSDs. The second connection is to a consumer network connected to the remote controller. In order to evaluate the effects of different network conditions on the performance of the system, a computer acting as a network emulator by running the WANem (short for WAN emulator) software is connected to the consumer network, allowing for the network conditions between the IoT gateway and the remote controller to be altered.

WANem allows for the control and monitoring of network conditions. WANem is a software tool developed to interface with incoming network traffic and introduce the desired network behaviour. WANem can add the following to a network: delays, jitter, packet loss, packet reorder, bandwidth limit, duplication, corruption and disconnection.

The WANem software package is installed on a device connected to the same network as the production plant and plant controller. A computer has been set-up with a Linux operating system with the WANem software running on a virtual machine. The computer is then attached to the same network as the physical and virtual systems. WANem will be used to change network characteristics to test the effect thereof on the Industrial IoT system.

Network packets flow from one host to another via WANem, with WANem producing the desired network conditions for testing. The packet flow and setup of the network emulator can be seen in figure 3.6.

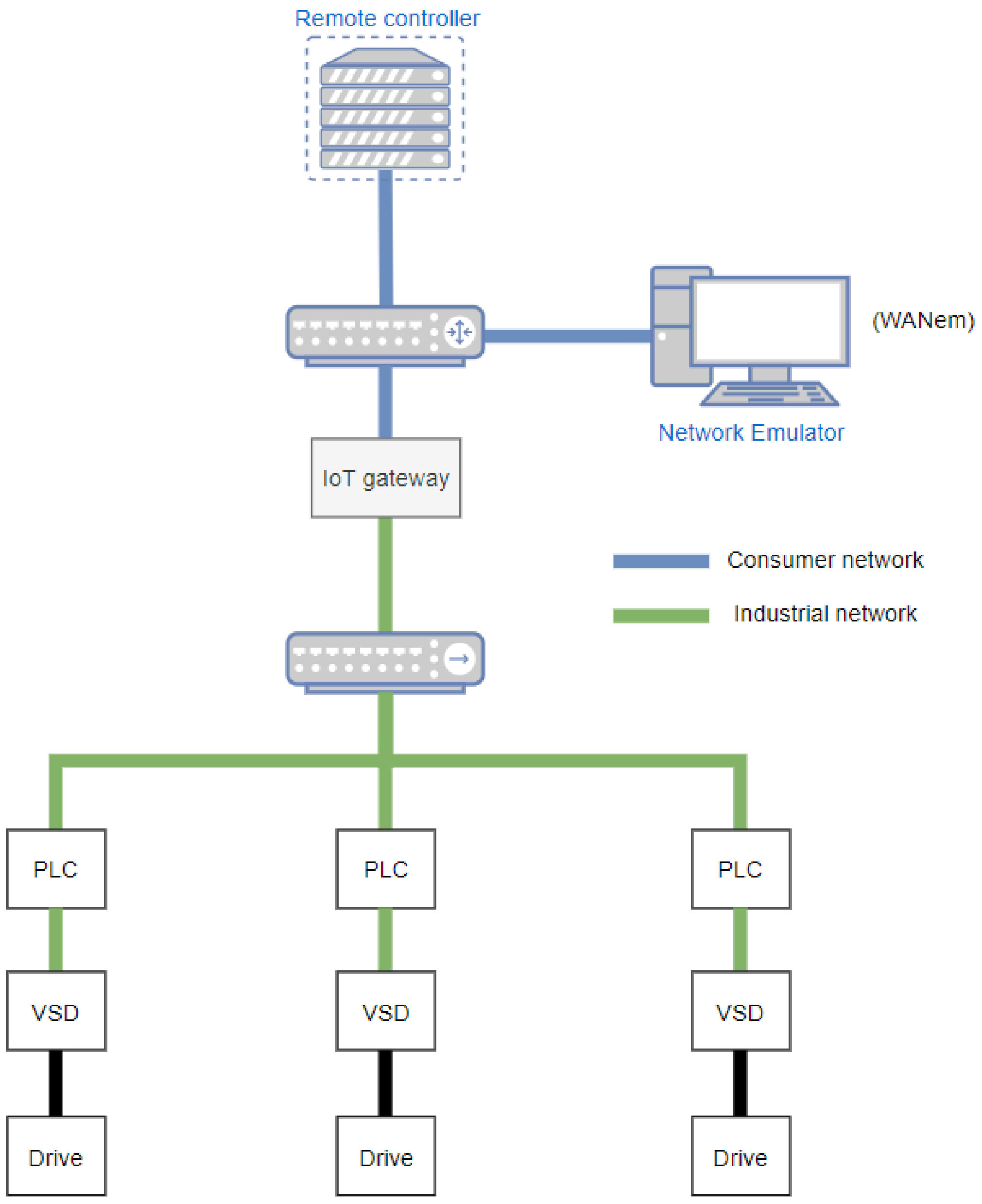


Figure 3.5: Physical systems and networking diagram

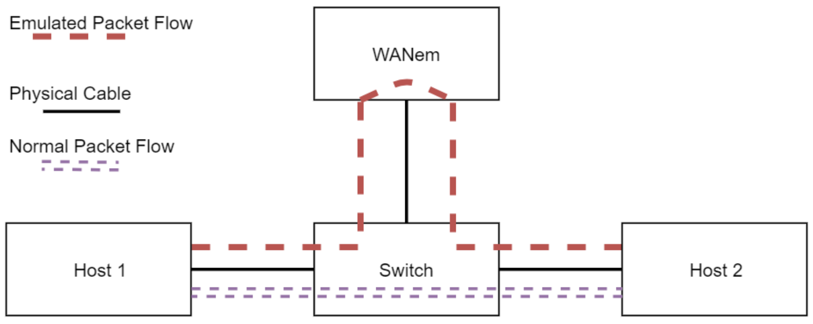
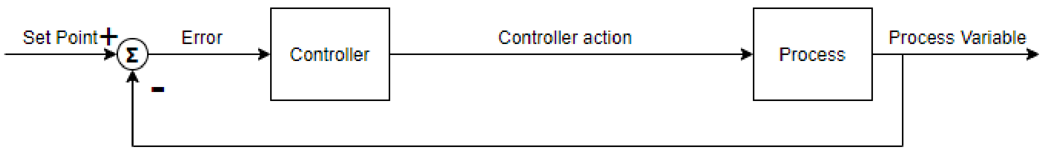


Figure 3.6: WANem packet flow

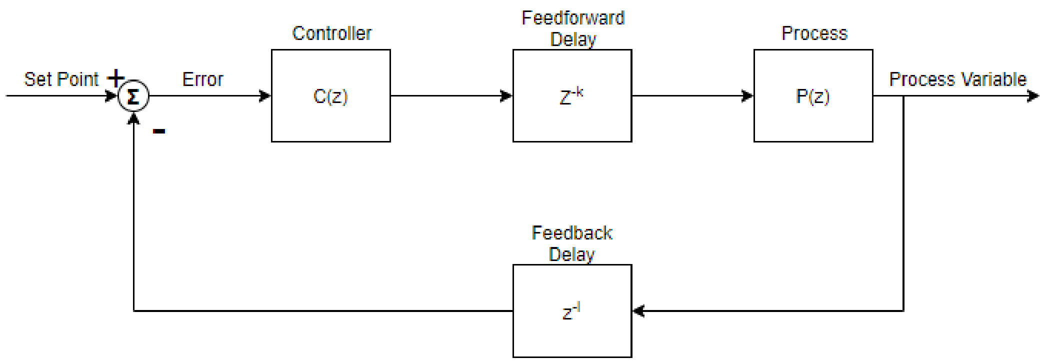
3.4 Delay mitigation mechanisms

Due to the controller being offloaded, and the added delay effect of network conditions to the closed-loop control system, it is beneficial to add a local delay mitigation mechanism or structure to help combat the effect of the different network conditions.

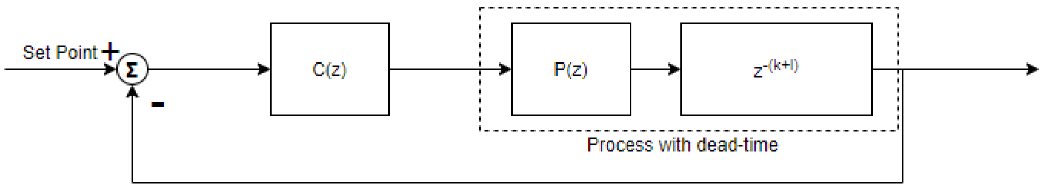
A tradition feedback control loop can be seen is fig. 3.7a. By moving the control to a remote server delays are added in both directions this is shown in fig. 3.7b. For the model $C(z)$ denotes the transfer function of the controller, and $P(z)$ the transfer function of the controlled process; z^{-k} is the feed-forward delay and z^{-l} the feedback delay. The model with delays can then be simplified to a model resembling a process with dead-time shown in fig. 3.7c.



(a) Traditional feedback control loop



(b) Feedback control loop model with delay



(c) System equivalent to process with dead-time

Figure 3.7: Feedback control loops for delay mitigation

For the typical feedback control loop as shown in figure 3.7a the closed-loop transfer function $T(z)$ is as follows:

$$T(z) = \frac{C(z)P(z)}{(1 + C(z)P(z))} \tag{3.1}$$

A controller $\bar{C}(z)$ is then derived for a process with dead-time, $P(z)z^{-(k+l)}$, the transfer function for a closed loop with a dead-time process is $\bar{T}(z) = T(z)z^{-(k+l)}$, yielding (3.2) to be solved for $\bar{C}(z)$

$$\frac{\bar{C}(z)P(z)z^{-(k+l)}}{1+\bar{C}(z)P(z)z^{-(k+l)}} = \frac{z^{-(k+l)}C(z)P(z)}{1+C(z)P(z)} \tag{3.2}$$

The controller $\bar{C}(z)$ is then given as:

$$\bar{C}(z) = \frac{C(z)}{1+(1-z^{-(k+l)})C(z)P(z)}. \tag{3.3}$$

Another approach to implement a local delay mitigation mechanism, as shown in [9], is to add a time-out to the controller. The controller will wait for a response, and if a response is not received within the time frame given, then the controller will use a predicted response value instead. Any late responses received will then be buffered and can then either be used or be discarded. This is a form of dynamic offloading where the controller will use prediction models to estimate changes rather than continuously checking for changes. The prediction models considered for delay mitigation are the Exponential Moving Average (EMA) and the Double Exponential Smoothing Model (DESM).

Exponential moving average

The exponential moving average is a weighted moving average of data values. The mathematical expression is given in (3.4) [9] with s_t the statistical value, and s_{t-1} the previous statistical value calculated. x_t is the current observed value and A the smoothing value, with $0 < A < 1$.

$$s_t = Ax_t + (1-A)s_{t-1} \tag{3.4}$$

The forecast for the next value predicted is given by:

$$F_{t+1} = s_t. \tag{3.5}$$

The initial value for the exponential moving average is described as:

$$s_1 = x_0. \tag{3.6}$$

Double exponential smoothing model

The double exponential smoothing model introduces a term for taking the change of slope or trend into account based on the *Holt model*. The mathematical expression is given in (3.7) with b_{t-1} the trend calculated in (3.8). A new smoothing factor B is introduced, to weigh the trend of the slope, where $0 < B < 1$ [9].

$$s_t = Ax_t + (1-A)(s_{t-1} + b_{t-1}) \tag{3.7}$$

$$b_t = B(s_t - s_{t-1}) + (1-B)b_{t-1} \tag{3.8}$$

A prediction can then be made for x_{t+m}, with $m > 0$. The prediction is then calculated as shown in (3.9).

$$F_{t+m} = s_t + mb_t \tag{3.9}$$

The initial values for the double exponential smoothing model are $s_1 = x_0$ and $b_1 = x_1 - x_0$.

The smoothing values A and B are between 0 and 1, while there is no formal calculation, the values can be calculated using optimization techniques for the lowest error between the data and the predicted vales. The values used in [9] are $A = 0.8$ and $B = 0.9$.

3.5 Control scheme

The emulated control scheme for the three motor experimental setup is as follows: A emulated plant-operator controls one motor. The second motor's speed set point is based on the measured speed of the first motor. The third motor's speed set point is then in turn based on the measured speed of the second motor. The control scheme for the motors is shown in figure 3.8 for an ideal and theoretical system with zero latencies within the whole closed-loop control.

For consistent testing, a startup scheme for the first motor is created to be executed for each test. The motor starts at 0 r/min; at 0.4 seconds a setpoint for 1500 r/min is given and at 7.0 seconds a set point for -1500 r/min is provided. The control scheme has a rate limiter of 1500 r/min over a second implemented for each motor.

The second motor's set point is determined from the speed measurements received from the first motor. The second motor setpoints are to match the first motor but in the opposite rotational direction.

The third motor's set point is determined from the speed measurements received from the second motor. The third motor aims to do two-thirds of the second motor's measured speed in the opposite direction. Shown below is the control scheme expression for each of the three motor setup.

$$\begin{aligned}
Motor_1 &= 0 && t < 0.4\\
&= 1500 && 0.4 < t < 7.0\\
&= -1500 && 7.0 < t\\[1em]
Motor_2 &= -1(Motor_1 MeasuredSpeed)\\[1em]
Motor_3 &= -2/3(Motor_2 MeasuredSpeed)
\end{aligned}$$

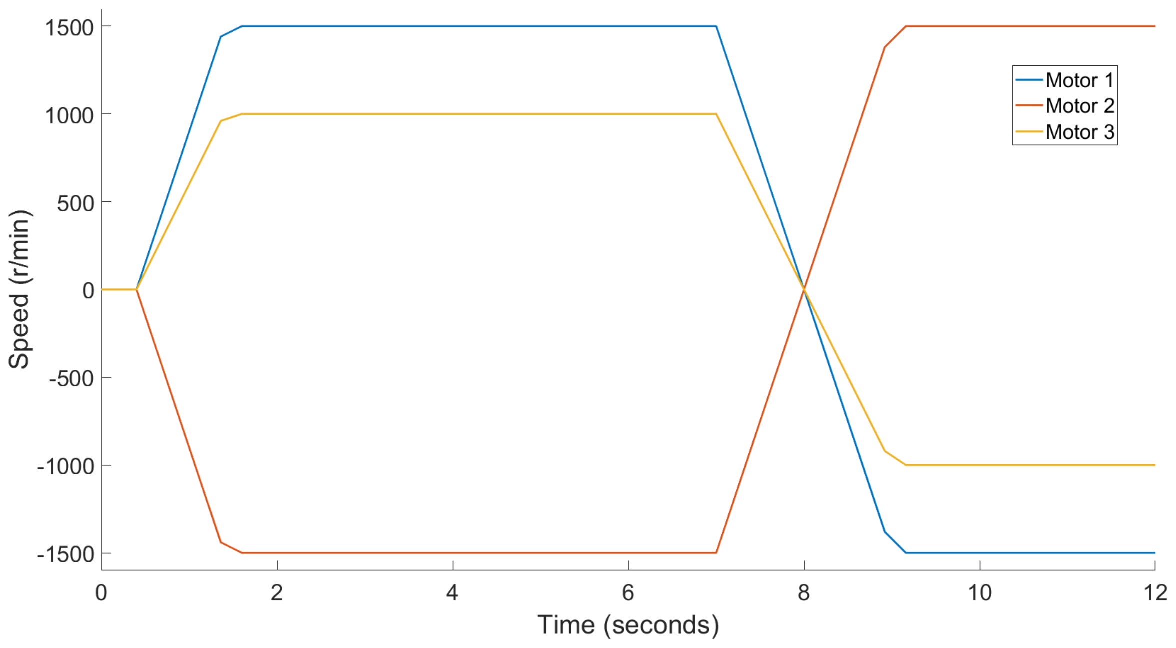


Figure 3.8: Speed graph for system of motors - startup control scheme

3.6 Research approach

After the construction of the physical system and the digital simulation model in SIMULINK®, the simulation model is validated for the physical system. The simulation model must represent the behaviour, information and communication of the physical system accurately to be used for comparisons and remote controller evaluations.

In the emulated system, the reference speed for each motor is determined by the remote controller. Speed measurements at discrete-time intervals are sent from the motor's local controller to the remote controller through the modifiable communications channel. The remote controller uses the speed information received from one motor to determine the speed reference of another based on the mission scheme for the emulated system. The reference speed is then sent from the remote controller to the relevant motor's local controller, through the same modifiable communications channel.

A baseline control group set of data is created for comparison. The control group data set constitutes the speed of the motors at discrete time intervals from the physical system. An average system response time is then determined for the difference in speed at each time interval between the motors in the systems.

The average system response time is obtained using the average time difference between settling times between the motor being controlled and the motor used to determine the set point of the controlled motor based on each step for the control scheme created. Response times are determined between motor two and motor one, as well as between motor three and motor two. The measured speed of motor one is used to determine the set point for motor two, and the measured speed of motor two is used to determine the set point for motor three. The response time is then the time it takes between settling times between the two motor pairs for each step of the control scheme.

The control group data set of average system response times are established with the best possible network conditions and without any delay mitigation mechanisms implemented. The average system response time gathered from the control group is used as a reference when compared to datasets from experiment runs where network delays, jitter and packet loss are added.

Tests are conducted to determine the effects of network latency, jitter and packet loss on the emulated system. The test will show the impact on an industrial application on a commercial network and how the network requirements change to fit the industrial use. The tests are repeated with different network conditions with and without delay mitigations implemented on the remote controller. The data received for each test are then compared against the control data set, and error values for each test are determined.

Test one determines the system performance impact of moving the network location of the remote controller from LAN to WAN. This is achieved by implementing the remote controller's script on the IoT gateway for emulating a LAN connected remote controller. Next, the remote controller is then implemented on the remote server connected via a WAN connection. The system performance is measured and then compared for a LAN and WAN connected remote controller.

Test two explores the effects of network latency by adding only network communication delays to the system. Delays from 0 to 500 ms are introduced in intervals to the system measuring the performance impact thereof on the system.

Test three explores the effects of both delays and jitter on the system. As before, delays are introduced in intervals from 0 to 500 ms. At each latency-interval additional jitter is added to network communication in intervals from 0 to 50 ms. The performance impact of jitter is then measured for each interval.

Test four explores the performance impact of packet loss on the system. Packet loss, measured in percentage, is introduced to the network communication in intervals from 0 to 10%. The performance impact of packet loss is then measured for each interval.

Once the performance impact of latency, jitter and packet loss on the system has been determined, the tests are re-run with delay mitigation implemented on the remote controller. The system performance is measured to determine how the effects of network conditions can be decreased when delay mitigations are implemented.

The experiment system diagram for the study is shown in figure 3.9. The system diagram describes both the physical and virtual systems.

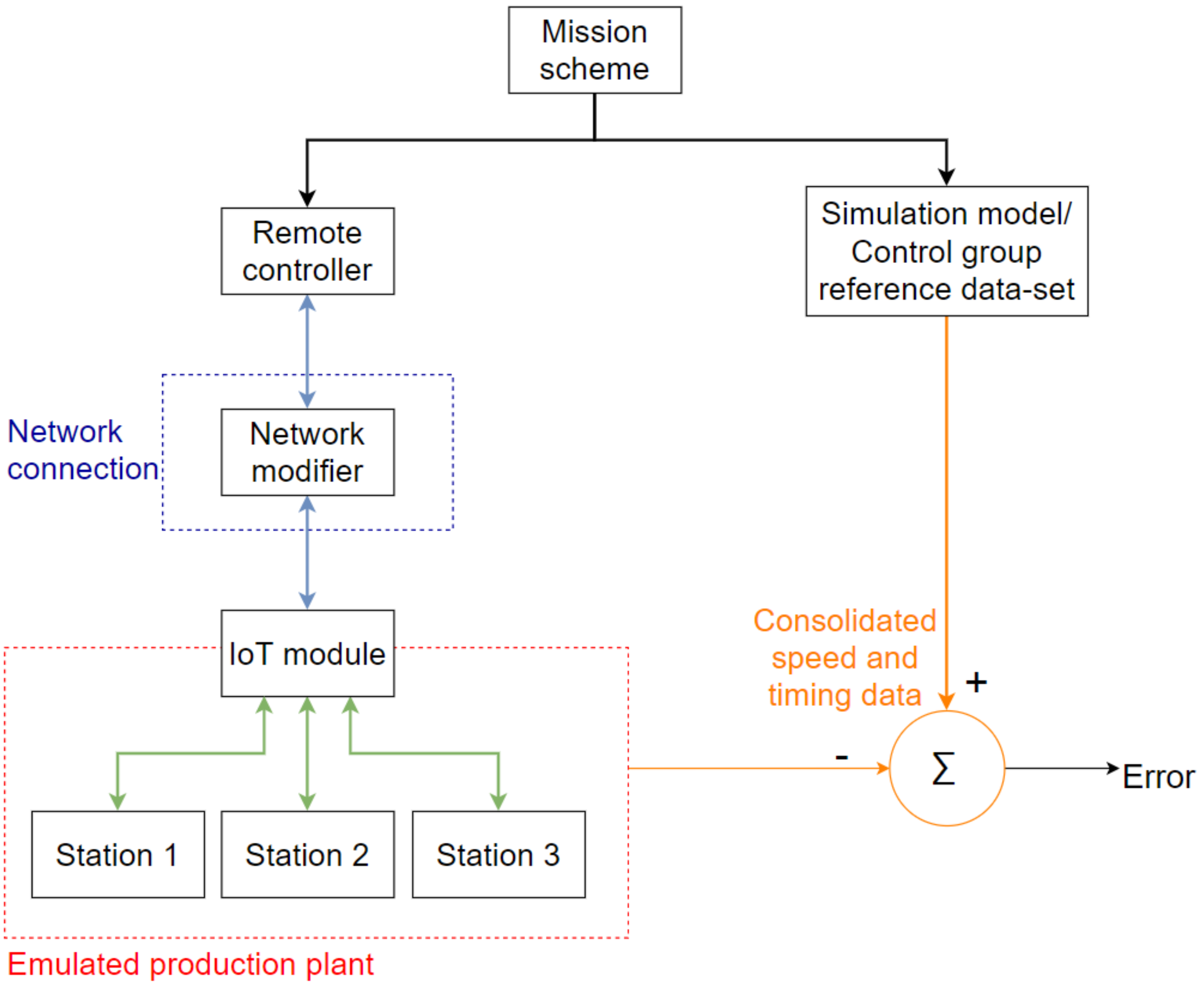


Figure 3.9: Experimental system diagram

3.7 Conclusion

Typical and networked/Internet-based control architectures were explored, and it was decided to opt for a bilateral architecture. A bilateral architecture was chosen because the architecture uses both remote and local controllers. With the addition of a network modifier, the network conditions between the remote controller and the local controllers can be altered, allowing for testing the performance impact of different network conditions on the performance of such an IIoT system. Due to the goal of adding a remote controller to a typical closed-loop control system, the proposed architecture uses both local controllers in a typical closed-loop control and a network-attached remote controller. The network architecture based on the proposed bilateral control loop architecture is created with an added connection for a network emulator. The proposed bilateral control structure and network architecture are implemented on the IIoT control and monitoring system.

The network emulator allows the manipulation of network conditions between the IoT gateway and the remote controller. The network conditions can be varied as desired to determine how different network conditions between the remote controller and the local controller affect the performance of the system.

Delay mitigations for implementation on the remote controller is discussed and explained. Delay mitigation implementations are tested on the experimental system with the expectation of mitigating the effects of different network conditions on the experimental system.

A control scheme is created for the system of motors, where the reference speed point of one motor is determined by applying the control scheme to another motor's measured speed. The control scheme allows for changes in system performance when different network conditions are applied to the experimental system. The changes can then be measured and compared to evaluate the performance impact of different network conditions on an IIoT control and monitoring system.

Four tests are designed to determine the impact of different network conditions on the performance of the experimental IIoT control and monitoring system. The tests make use of the network emulator to change the network conditions. The tests will determine how the network location of the remote controller, network latency, network jitter and packet loss will impact the performance of the system. The tests are repeated with delay mitigations implemented on the remote controller. The effects of adding delay mitigations are observed with the hopes of improving the performance of the IIoT for certain network conditions.

Chapter 4

Simulation model

4.1 Introduction

Previously the experimental design and research approach was given to determine the performance impact of different network conditions on an IIoT control and monitoring system. Stated in the chapter is that the development and assembly of a simulations model of the experimental system is to be made.

The digital simulation model is developed and deployed, and the details thereof discussed in this chapter. The simulation model of the experimental system is used to test control schemes as well as to aid in determining the performance impact of different network conditions on the system.

This chapter describes the digital simulation model representing the three motor industrial IoT control and monitoring system. The section starts with a general overview of the simulation model layout. The system model is divided into subsections describing the behaviour, information and communication models. Each section of the simulation is described and then validated for an accurate representation of the experimental system. The chapter presents sections on the behaviour, information and communication sub-models, as well as verification testing of the simulation model, followed by a conclusion for the section.

4.2 Simulation modelling

The system model residing in MATLAB® and Simulink® resembles the physical three-motor, remote control system. The simulation model is used to test multiple control schemes before implementing the control schemes on the physical system. The system model is made out of various functional simulation sub-models describing the process and behaviour of the system. The system functional model implemented is shown in figure 4.1 with five distinct aspects: input, output, behaviour, information and communication.

The input for the digital system model is the same as for the physical system. The mission scheme for the system is the input to both systems. The mission scheme includes reference speeds and timing schemes for the different motors. The mission scheme and system-wide control are located within the remote controller.

The output is made out of timing data and the measured speeds of the motors compared to their reference speed set points.

The core of the digital system model is the sub-models describing the behaviour of the system. The response of the motor and local control subsystems are described with electrical Simscape™ models. The Simscape models describe the response of the local controllers, variable speed drives (VSD) and induction motors for a given control signal input from the remote controller. The output is measured speeds from the motors at discrete time intervals. The control model for the control of the system of motors (Remote controller) is modelled to receive measured motor speeds and output speed set point back to the motors based on the control scheme for the system.

Information within the system collected for comparisons and logging is the reference speed set point as determined by the remote controller based on the measured motor speeds and the measured motor speeds for each motor within each discrete timeframe.

The communication sub-model aims to model the network communication between the remote controller and each motor's local controller. The communication sub-model describes network communication latencies and jitter. The communication model receives an unaltered signal, either measured speeds from the local controllers or speed set points from the remote controller, and outputs the received signal with network latencies and jitter applied.

The digital system model is shown in figure 4.2. The figure shows the three motor sub-models, each with a motor speed set point as input and a measured motor speed value as an output. The reference speeds set points are calculated by the remote controller based on the measured motor speed inputs and the mission scheme for the motors. The reference speed set point signals are combined and sent through the communications models to add network latencies and jitter to the signal. After the network conditions have been applied to the reference speed set point signal, it is divided and sent to the appropriate motor model for execution.

Each motor sub-model produces measured speed values that are combined and sent through the communications model to add the same network conditions as before, to the remote controller. The remote controller then uses the received values to produce the reference speed set point values as dictated by the mission scheme. The process is then repeated until the mission scheme is completed.

A log is created of reference speed set points for each motor and the measured speeds of each motor at discrete time intervals. The log is used to calculate average system response times that are used for comparing system performance for different network conditions. Comparisons are also made between the digital simulation model and the physical system's results.

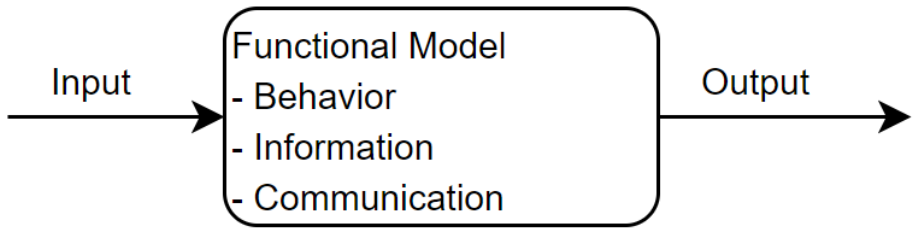


Figure 4.1: System simulation functional modelling diagram

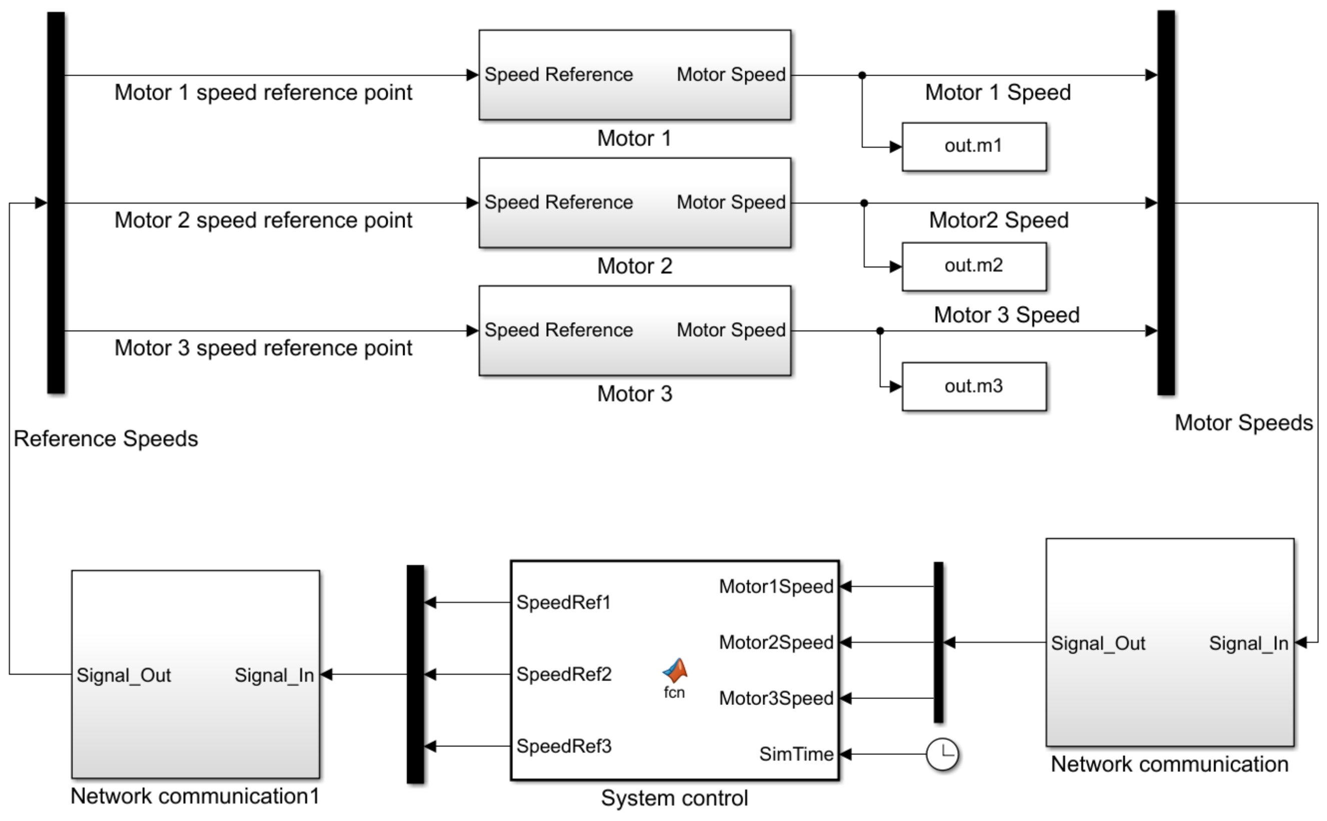


Figure 4.2: Digital system simulation model overview

4.3 Behavioral models

A subsystem is created for each motor, containing MATLAB®, Simulink® and electrical Simscape™ components for the variable frequency drives, the induction motor and the local controller. One such subsystem can be seen in figure 4.3. The sub-model describes a squirrel cage induction motor with a variable frequency drive (VFD). The VFD/motor model is controlled utilising vector control. The vector control implemented is field-oriented control (FOC).

The subsystem is adapted from a Simulink®/Simscape™ example for field-oriented control of a squirrel cage three-phase induction motor. The model is modified to fit the operation of the physical system by substituting the example motor parameters with the equivalent circuit parameters of the physical motor used in the mechatronics laboratory, as shown in table 4.1. The parameters in table 4.1 contains both rated values as per the motor's datasheet and equivalent phase circuit values calculated and measured by the attached VSD.

The induction motor has a rating of 0.55 kW, [230 V delta/400 V Y] and a rated current of [2.2 A delta/1.26 A Y]. The induction motor is classified as an IEC class motor. The motor characterisation is done from datasheet values as well as parameter values determined by the VSD's control unit. The VSD performs stationary measurements followed by parameter calculations of the motor to produce resistance and leakage inductance values for the stator and rotor as well as the magnetising inductance. The inductive reactance can be calculated with (4.1), where X_L is the inductive reactance in Ohm, f is the frequency in Hertz and L the inductance in Henry (H). The inductive reactance is calculated for the rated electrical frequency of 50 Hz.

$$X_L = 2\pi f L \tag{4.1}$$

The base values calculated for the motor is done with the following equations. Base voltage, V_{base} is calculated as a peak value, line to neutral in volts, using the rated electrical voltage, V_n.

$$V_{base} = \frac{V_n\sqrt{2}}{\sqrt{3}} \tag{4.2}$$

The base current, I_{base} is calculated as a peak value in amps using the base voltage and the rated nominal power P_n in Watts.

$$I_{base} = \frac{P_n}{1.5(V_{base})} \tag{4.3}$$

Base resistance, Z_{base}, in Ohm is calculted next.

$$Z_{base} = \frac{V_{base}}{I_{base}} \tag{4.4}$$

The base electrical radial frequency, ω_{base} in radians per second is calculated, using the rated elctrical frequency, f_n, in Hertz.

$$\omega_{base} = 2\pi f_n \tag{4.5}$$

The equation for base torque, T_{base}, in Newton meters is as follows, where p is the number of pole pairs in the induction machine.

$$T_{base} = \frac{P_n}{\omega_{base}/p} \tag{4.6}$$

Nominal flux, *psin*,of the induction machine is calculated as follows.

$$psin = \frac{V_{base}}{\omega_{base}} \tag{4.7}$$

Additional motor characterisation is done by performing a stator resistance test, a blocked-rotor test and a no-load test on a three-phase induction motor.

Stator resistance test:
The purpose of this test is to determine the resistance of each phase winding of the stator. This test is done by measuring the resistance between any two terminals of the motor. The purpose of this test is to determine the per phase resistance R of the stator. This value can then be used to determine the value for R_1 in the approximate equivalent per phase circuit diagram.

$$R_1 = 3.58773\,\Omega \tag{4.8}$$

Blocked-rotor test:
The purpose of this test is to determine the equivalent rotor resistance. The test is also known as the locked rotor test, and it is very similar to the short circuit test of a transformer.

For the test, the rotor is locked, allowing no movement of the rotor. The stator is connected to a variable voltage three-phase source. The voltage is slowly increased until the rated current is achieved. Voltage, current and power measurements are taken.

The equivalent resistance R_e can be determined by the follwing equation, where V_{br} is the applied voltage in volts, I_{br} is the rated current in amps and P_{br} is the power input measured in Watt.

$$R_e = \frac{P_{br}}{I_{br}^2} \tag{4.9}$$

And since R_1 in known (equation 4.8) R_2 can be calculated:

$$R_2 = R_e - R_1 \tag{4.10}$$

However, the series impedance is:

$$Z_e = \frac{V_{br}}{I_{br}} \tag{4.11}$$

Therefore,

$$X_e = \sqrt{Z_e^2 - R_e^2}. \tag{4.12}$$

Since it is rather difficult to isolate the leakage reactance X1 and X2, they are assumed to be equal for all practical purposes.

$$X_1 = X_2 = 0.5X_e \tag{4.13}$$

The total series impedance, Z_e, can be written as:

$$Z_e = R_1 + R_2 + j(X_1 + X_2) = R_e + jX_e \tag{4.14}$$

No-load test:
The purpose of this test is to determine the power of the induction motor. The no-load test is very similar to the open-circuit test in transformers, and it is where the motor can run freely without any load connected to it. In this test, the slip of the motor is almost nothing. The core loss of the equivalent resistor can be calculated by subtracting the windage and friction loss from the input power.

The rated voltage is applied upon the stator windings, and the motor is operated freely without any load. The slip is almost zero, and thus the rotor impedance is nearly infinite (hence the open circuit characteristics).

For this test, the following parameters are measured and calculated on a per phase basis.

- V_{oc} - Rated applied voltage in volts
- I_{oc} - Input current in amps
- W_{oc} - Power input in watts

The power loss due to the core (core loss Poc) is represented with a core resistance, R_c. The friction and windage loss ($P_{fw\Phi}$) can be measured. The core loss can be calculated by subtracting the friction and windage loss from the power input.

$$P_{oc} = W_{oc} - P_{fw\Phi} \tag{4.15}$$

From equation 4.15, the core resistance can be determined:

$$R_c = \frac{V_{oc}^2}{P_{oc}} \tag{4.16}$$

The power factor under no-load is:

$$\cos\theta_{oc} = \frac{W_{oc}}{V_{oc}I_{oc}} \tag{4.17}$$

The magnetic reactance is:

$$X_m = \frac{V_{oc}}{I_{oc}\sin\theta_{oc}} \tag{4.18}$$

The magnetisation reactance can also be determined by:

$$S_{oc} = V_{oc}I_{oc} \tag{4.19}$$

$$Q_{oc} = \sqrt{S_{oc}^2 - W_{oc}^2} \tag{4.20}$$

and

$$X_m = \frac{V_{oc}^2}{Q_{oc}} \tag{4.21}$$

The modified subsystem created to represent the physical system's motors, VSDs and local controllers is expanded to add the remote controller functionality. The motor model is placed within a subsystem with an input and an output. The subsystem receives a speed reference set point as an input. The speed reference set point is then executed by the field orientated controlled induction motor. The output from the motor model is split by a *demux* function to extract the relevant information from the motor. The subsystem then outputs the speed calculated for the motor by the motor model.

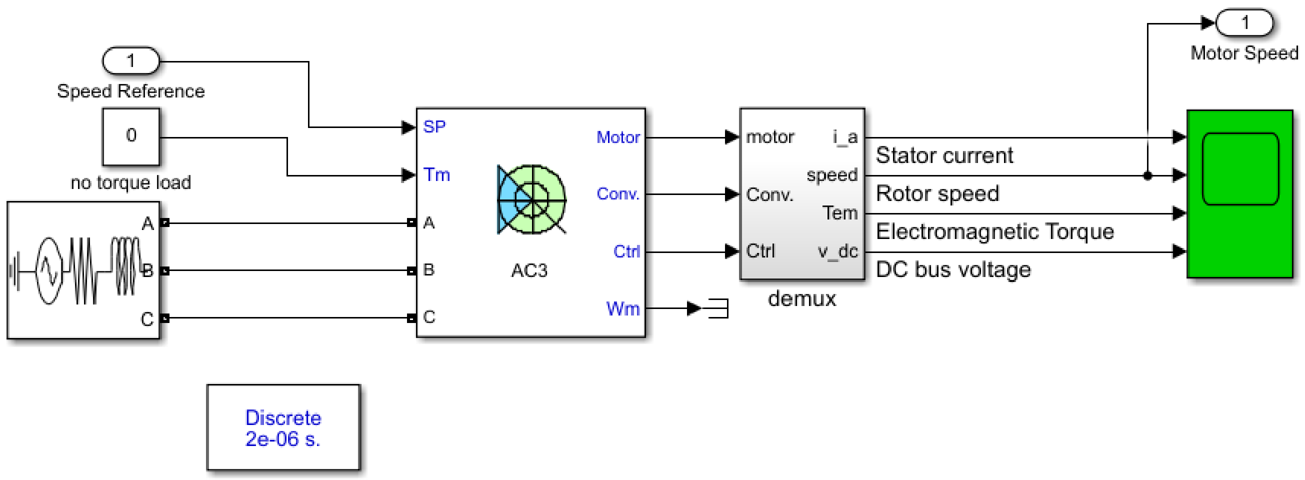


Figure 4.3: Induction motor, VSD and local controller - digital sub-system model

Table 4.1: Motor simulation model parameters

Rated data from datasheet	
Rated apparent power[kW]	0.55
Rated Voltage[V]	230Δ/400Y
Rated electrical frequency[Hz]	50
Number of pole pairs	2
Rated motor speed[r/min]	1440
Rated motor current[A]	2.2Δ/1.26Y
Moment of inertia[kg m^2]	0.0021
Calculated base parameters	
Base voltage, peak, line-to-neutral[V]	187.7942
Base current, peak[A]	1.9525
Base resistance[Ω]	96.1818
Base radial frequency[rad/s]	314.1593
Base torque[Nm]	3.5014
Nominal flux	0.5978
Equivalent circuit parameters	
Stator resistance[Ω]	3.58773
Stator leakage inductace[mH]	14.60895
Stator leakage reactance[Ω]	8.177267
Rotor resistance[Ω]	2.36502
Rotor leakage inductance[mH]	15.37453
Rotor leakage reactance[Ω]	7.19507105
Magnetising inductance[mH]	293.37772
Magnetising reactance[Ω]	92.16732899

4.4 Information model

The information model in the system simulation model resembles the physical system. Speed measured from the motors is sent together in a single packeted message frame to the remote controller. The remote controller sends a packeted message frame with the reference speeds for the motors for interpretation by the local controllers. Simulation time is used for timestamp data in the message frames.

The speed measurements of each motor are multiplexed into a single signal. The signal is then sent through the communication model (discussed in the next section) and then demultiplexed for the remote controller. The same occurs for reference speed set points from the remote controller to each motor subsystem. The three reference speeds are multiplexed into one signal. The signal is then sent through the communication model and demultiplexed for each motor subsystem to receive their unique, relevant reference speed set point.

4.5 Communication model

The communications sub-models describes time delays for packets/information sent over a network/internet communication channel. The internet time delay $T_d(k)$ at the instant k can be described as follows (Han et al. 2001):

$$T_d(k) = \sum_{i=0}^{n} \left[\frac{l_i}{c} + t_i^R + t_i^L(k) + \frac{M}{b_i} \right], \tag{4.22}$$

$$T_d(k) = \sum_{i=0}^{n} \left(\frac{l_i}{c} + t_i^R + \frac{M}{b_i} \right) + \sum_{i=0}^{n} t_i^L(k),$$

$$T_d(k) = d_N + d_L(k),$$

where l_i is the ith length of the network link, c the speed of light, t_i^R the routing speed of the ith node, $t_i^L(k)$ the delay caused by the ith node's load, M the amount of data, and b_i the bandwidth of the ith link. d_N is a term, which is independent of time, and $d_L(k)$ is a time-dependent term. **Because of the time-dependent term $d_L(k)$, it is somewhat unreasonable to model the Internet time delay for accurate prediction at every instance.**

In the system simulation model, signals to and from the remote controller are altered with the communication sub-model shown in figure 4.4. The communication sub-model adds latency (packet delays) as well as jitter (packet delay variance) to the signal to represent network delays and jitter. The communications sub-model represents only one way of communication. Two identical models are used to describe communication in both directions, to and from the remote controller.

The input signal passes through a variable transport block adding a varying delay to the signal. The output to the system is the signal with the varying network delays created based on the input signal.

The varying delay contains elements of both a constant delay and a varying delay. A constant number block represents the constant delay, in the case of the figure shown is 100 ms. The variance in the delay is created by a normally distributed random number that is clamped by a MATLAB® function block. The MATLAB® function block ensures that the distributed random number is between the desired jitter value. The sum of the constant delay and jitter produces a shifted gamma distribution of communication delay values.

The t_i input of the variable transport block receives the sum of the constant and variable components used to produce and add the varying communication delays to the input signal.

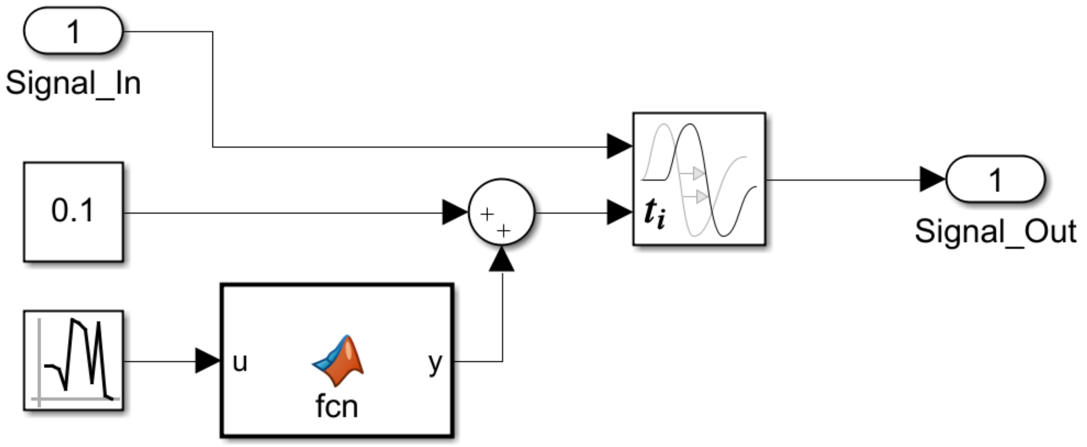


Figure 4.4: System simulation model - communication sub-model

4.6 Verification

The response from the simulation model is verified for an accurate representation of the experimental system. Firstly the sub-model of the induction motor, VSD and local controller is verified. This is done by comparing the response of the sub-model to that of the physical sub-system. A start-up sequence and a speed reduction sequence is executed on both systems and the response thereof compared. For the response of the simulation sub-model to be verified, it is expected to be similar to the response of the physical system.

The response of the simulation's communication sub-model is verified by comparing it mathematically and statistically to the response observed of physical network communication. The response of the communication sub-model is determined verified if a strong mathematical and statistical relationship between simulation and physical measurements can be made.

The control and operation of the simulated experimental setup are tested by implementing the control scheme created for the experiment system. The response of the simulation model is examined when the control scheme is implemented. The expected response of the system is that the motors will act as stated in the control scheme for the system. It is also expected for the response of the motors to incorporate the additional delay due to the 100 ms sample rate implemented in the system.

4.6.1 Motor sub-system

The response of the induction motor, VSD and local controller sub-model was verified for an accurate representation of the response from the physical induction motor, VSD and PLC system used in the experimental setup. The simulation model's response is compared to the physical system as set up in the mechatronics laboratory. The physical system is described in section 5.3.

The response of the physical system and the digital simulation is shown in figure 4.5. The figure shows start-up sequence response values of the motors from 0 r/min to a speed set point of 1500 r/min. Both systems use a rate limiter of 1500 r/min over 1 second, resulting in a 1 second ramp-up time for a speed set point of 1500 r/min.

The speed reduction response of the physical system and the digital simulation is shown in figure 4.6. The speed reduction response sequence shown is from a starting value of 1500 r/min to a set point of 0 r/min. The rate limiter of 1500 r/min over 1 second of both systems is also applied to the speed reduction sequence.

From the figures, it can be seen that the motor model is an accurate representation of the physical system. A calculated average delta of 0.1% can be observed between the responses of the simulation and the physical

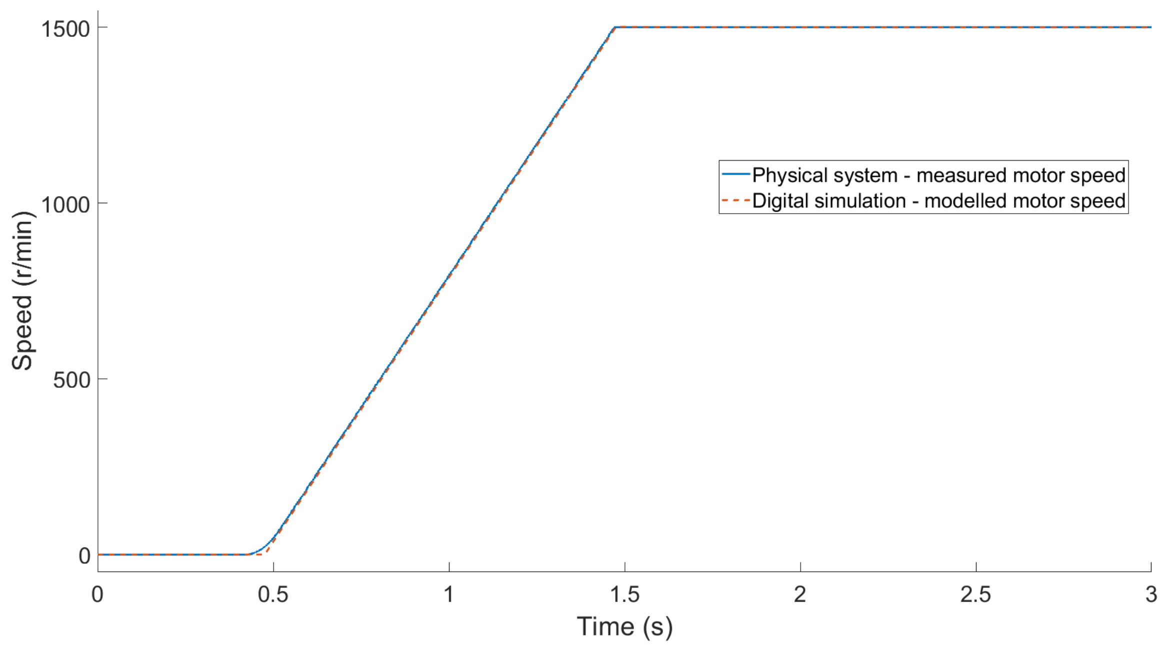


Figure 4.5: Motor speed graph - physical vs modelled motor startup response

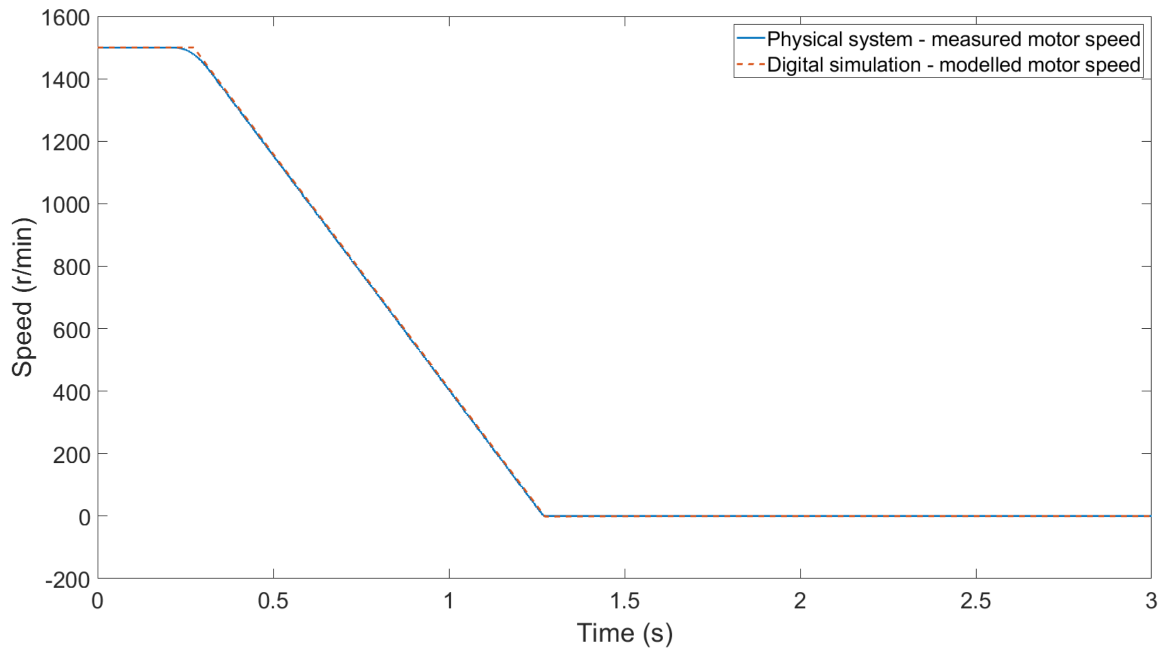


Figure 4.6: Motor speed graph - physical vs modelled motor speed reduction response

systems. The response of the simulation model closely resembled the response of the physical system. The response is, therefore, as expected from the simulation model.

4.6.2 Communication sub-model

The response of the simulation's communication sub-model is designed to resemble network and internet communication, by adding the appropriate network conditions to the communications signal. The response is verified against a measured network response in a mathematical and statistical sense.

The communications sub-model is described in (4.23), where $u(t)$ represents the input signal, $y(t)$ the output siganl, t_0 the constant delay added by the transport delay block, and $\tau(t)$ represents the delay variance; varried by an uniform random number (jitter).

$$y(t) = u(t - t_0 - \tau(t)) \tag{4.23}$$

Using (4.22), presented prior, that describes internet time delay as $T_d(k) = d_N + d_L(k)$ at the instance k the time independant term d_N is described in (4.23) as $d_N = t_0$ and the time dependat term $d_L(k)$ is described as $d_L(t) = \tau(t)$ therefore the equation can be described as (4.24):

$$y(t) = u(t - T_d(t)) \tag{4.24}$$

As stated by Vern Paxson, internet paths and thus delays are modelled using a shifted gamma distribution. The parameters of the distribution are dependant on the time of day and vary from the path to path [35].

Figure 4.7 shows a histogram of measured communication delays created by WANem. Conditions set for WANem is a round-trip time (RTT) of 300 ms and 30 ms jitter per communication direction. Figure 4.8 shows a histogram of the modelled communication delays of the simulation model. The same conditions of a 300 ms round-trip time (RTT) and 30 ms jitter per communication direction is set for the communications model.

Both figures show a right-skewed, shifted gamma distribution; consistent with the findings of Vern Paxson in [35]. The communications model can then be described as an accurate representation of internet delays observed based on a mathematical and statistical approach. The response of the communication sub-model is mathematically and statistically as expected and therefore verified and accurate representation of physical network communication.

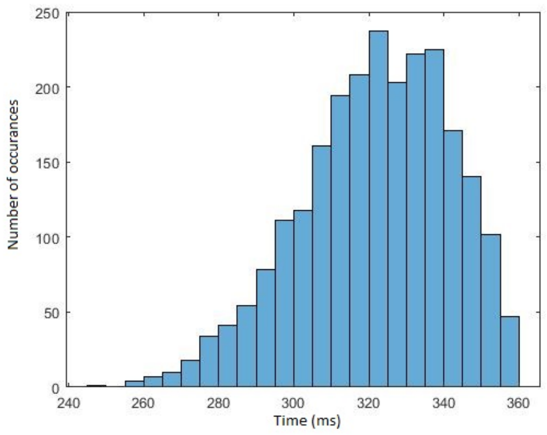


Figure 4.7: Histogram graph - measured communication delays (300 ms RTT; 30 ms Jitter)

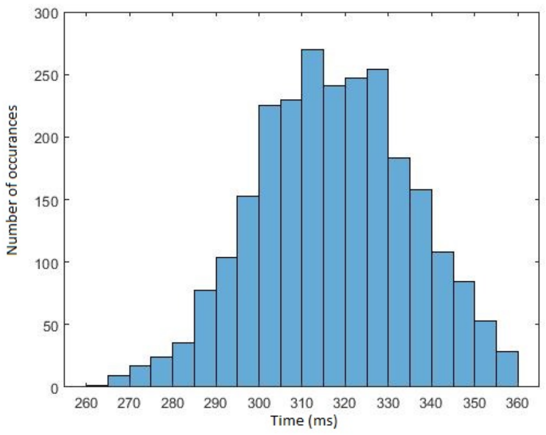


Figure 4.8: Histogram graph - modelled communication delays (300 ms RTT; 30 ms Jitter)

4.6.3 Simulation model

The response of the simulation model when subjected to the control scheme, as described in the experimental design chapter, is verified. The expected response is that described with the control scheme, shown in figure 3.8, but with the additional delays between the motor outputs. The delays are due to the sampling rate of 100 ms, one way, within the system.

The control scheme is implemented directly on the simulation model. The first test of applying the control scheme to the simulation model is done without any network conditions applied to the model. A difference between the control scheme and the response of the model should be observed due to the 100 ms, one way, sampling rate of the system. The sampling rate on the system will result in a delay between the motors' outputs. The response of applying the control scheme to the simulation model is shown in figure 4.9.

Examining the response of the simulation model obtained shows a response to the control scheme but with the added delay between the motors' outputs due to the sampling rate of the system. The response of the simulation model for the control scheme implemented is as expected.

Next, the response of the simulation model is tested when implementing the control scheme, along with the addition of network delays. The response is expected to look similar to the response of the system without network conditions but with additional delays between the outputs of the motors.

Figure 4.10 shows the response of the simulation model with network latencies of 200 ms round trip time and jitter of 10 ms. The response is similar to the response of the system with no network latencies, but except for the additional delays between the outputs of the motors. The additional delays are expected for the response due to the effects of the network conditions on the system.

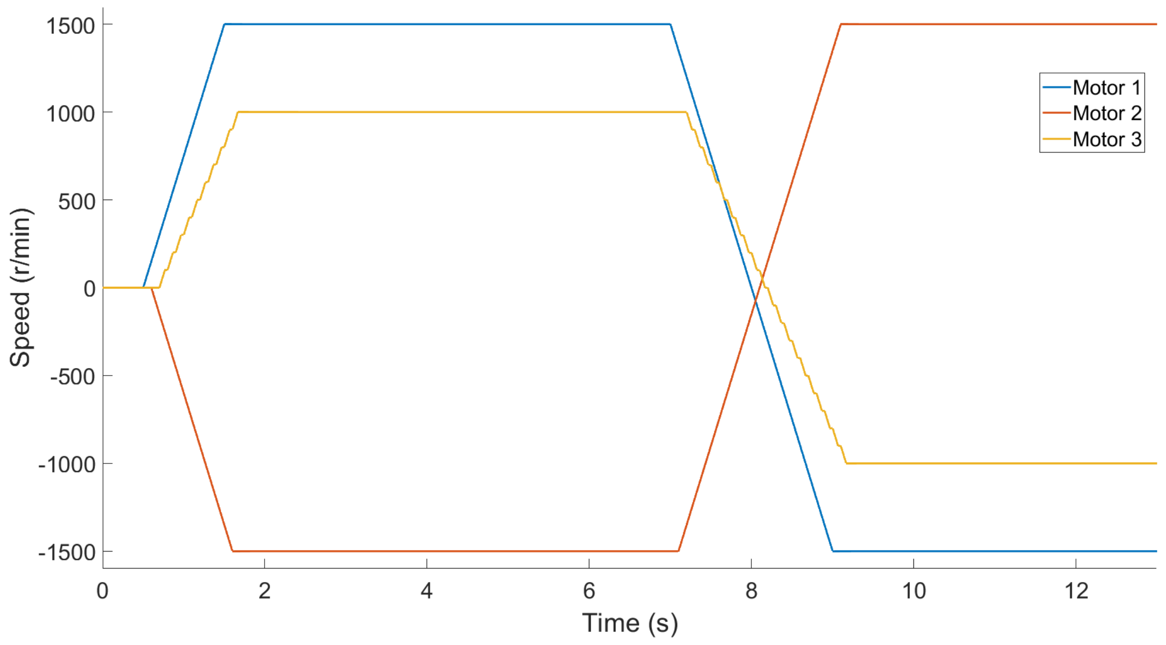


Figure 4.9: Speed graph - Simulation model control scheme response

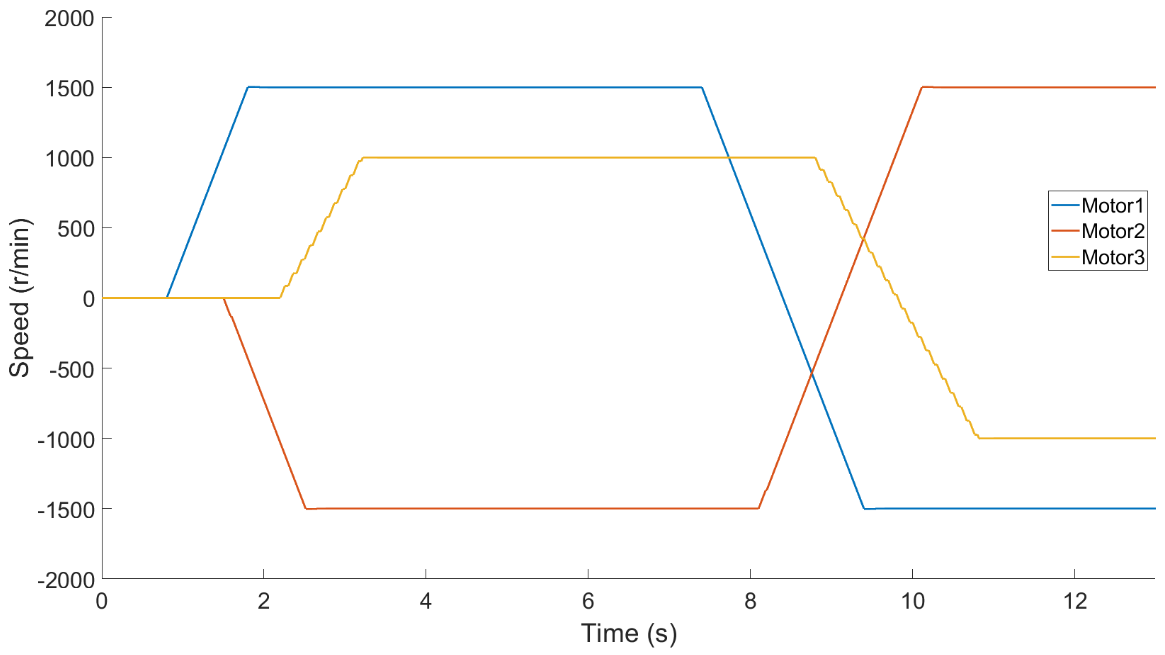


Figure 4.10: Speed graph - Simulation model control scheme response, with network delays

4.7 Conclusion

In this chapter, the digital system simulation model for the experimental setup is created and described. The digital system model represents the physical system in five distinct aspects, namely: input, output, behaviour and communication. The input of the system is the control scheme, and output of the system is the measured simulation speeds of the system of motors for discrete time intervals. Behaviour is modelled using MATLAB®, Simulink® and Simscape™ models for the induction motors, VSDs, local controllers and remote controller. Information models describe the flow of measured speeds from the motors, as well as reference speed set point from the remote controller. The communication model describes the network communication between the local controllers and the remote controller. The communication model is based on a statistical representation of network latencies and jitter values. The simulation model uses parameters extracted or calculated from datasheets and physical measurements for accurate modelling of the experimental system.

The goal of the simulations model is to aid in evaluating the performance impact of various network conditions on an IIoT control and monitoring system. The information received from the simulation model is used to compare and draw comparisons based on the impact different network condition have on the performance of an IIoT system. The simulation model is tested to validate and verify an accurate representation of the experimental system. The system proved to accurately represent the behaviour, information and communication of the experimental system. The control scheme created for the research approach was tested on the simulation model before the implementation of the experimental system.

Chapter 5

Experimental system

5.1 Introduction

Following the development and implementation of the simulation model for the experimental system, the physical system is created and tested.

The design building and testing of the physical system to emulate an IIoT control and monitoring system is shown in this chapter. The physical system is to be subjected to different networking conditions to measure the performance impact on the system.

Firstly a brief overview is given of the experimental setup. The experimental system is created within a mechatronics laboratory with Siemens© industrial equipment that was made available for this study. The experimental system is created based on the experimental design discussed in chapter 3. The hardware setup and components are discussed, followed by the device connections and interfaces. Next, the software for the various components is explained. The control scheme created for the experimental system is tested on the physical system and presented. Finally, some conclusions are drawn with respect to the experimental system.

5.2 Experimetal setup overview

The laboratory setup consists of a network-attached remote plant controller and different stations simulating an interdependent production plant. The network controller is responsible for system-wide control of three subsystem stations. Each subsystem consists of a local controller and an electrical motor. The local controller receives an input from the remote controller via the IoT gateway, performs local control on the electrical motor, measures the speed of the motor and sends the speed data back to the remote controller through the IoT gateway.

The experimental setup, done in a mechatronics laboratory, will emulate a production system where motors are used at different stations. The plant controller sends control and timing data to the various stations within the production plant through a network connection. Each of the stations within the production plant is control and timing-dependent, but do not perform the same operation.

5.3 Hardware Setup

Hardware used for the experimental setup includes the following:

- Two computers with integrated network interface cards.
  (One computer is used for running the network emulation and monitoring software, while the other is

used to run the remote controller application.)

- Consumer networking equipment.
  (A TP-LINK® 10/100Mbps router, model no.: TL-WR741ND, with cat 5e networking cable).
- Siemens© IoT2040 industrial intelligent gateway, Order no.: 6ES7647-0AA00-1YA2.
- Industrial networking equipment
  (Siemens© Compact Switch Module CSM 1277 10/100Mbit/s unmanaged switch, order no.: 6GK7277-1AA10-0AA0 and Siemanes Profinet networking cables).
- Three Siemens© S7-1200 PLCs (One for each of the three local stations)
  PLC information: CPU 1214C DC/DC/DC, Order no. 6ES7 214-1AG40-0XB0, Version: V4.0.
- Three Siemens© KTP700 Basic PN 7 inch HMI panels, Order no.: 6AV2 123-2GB03-0AX0, Version: V13.0.0.0.
- Three Motor stations consisting of:
  - Power inverter, Siemens© Power module PM240-2 Order no.: 6SL32101PB138UL0.
  - VFD, Siemens© G120 Control unit, capable of PROFINET support of vector control. Order no. 6SL32460BA221FA0.
  - Siemens© Sinamics control unit CU250s-2 PN, Order no.:6sl3246-0BA22-1FA0, Version: V4.7.10.
  - Three-phase squirrel-cage 4 pole induction motor, Siemens© 1AV3082B Order no.: 1LE1003-0DB22-2AB4.
  - Rotary encoder, (G11) Kübler© Sendix 5020 Order no.: 8.5020.0064.1024.S222.

In each station, the PLC is seen as the local controller, controlling the VSD, power inverter and connected induction motor. An induction motor is also known as an asynchronous motor. It is an AC machine that uses the electromagnetic induction from the magnetic field (stator) to produce the electric current in the rotor. Induction motors are mainly used for constant speed applications in conjunction with variable frequency drives.

The power inverter and VSD apply speed control of the three-phase induction, as well as providing the power connections required for the operation of the motor. Speed control is achieved with vector control by the VSD. Speed measurements obtained by the VSD through means of a rotary encoder attached to the motor is sent to the PLC through an industrial network connection.

As per the network diagram shown in figure 3.5, the PLCs are connected to the IoT gateway and motor stations via the industrial network.

The IoT gateway with two separate physical network connections allows the gateway to connect to both the industrial and consumer networks at the same time. The consumer network connects the IoT gateway and the two other computers running the network emulation software and the remote controller applications, respectively.

The constructed setup of the Siemens© S7-1200 PLC frame and electric motor drive in the mechatronics laboratory can be seen in figure 5.1 and figure 5.2 respectively.

5.3.1 Hardware construction

The Siemens© industrial hardware had to be assembled and connected before it could be used in this study. Two different setups were created to assemble and connect the required Siemens© industrial hardware.

Firstly a PLC setup was created for use. Originally a setups of Siemens© S7-1500 PLCs and Siemens© TP700 HMI panels was planned for construction and use in this study. However, due to time constrains

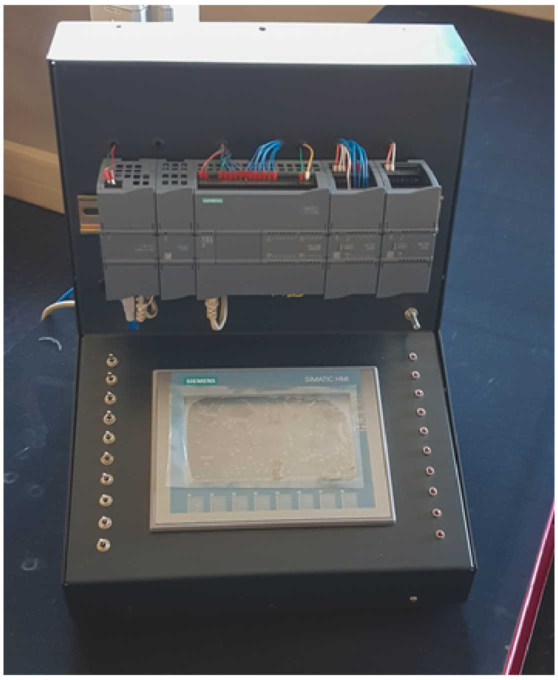


Figure 5.1: Siemens© S7-1200 PLC frame set-up

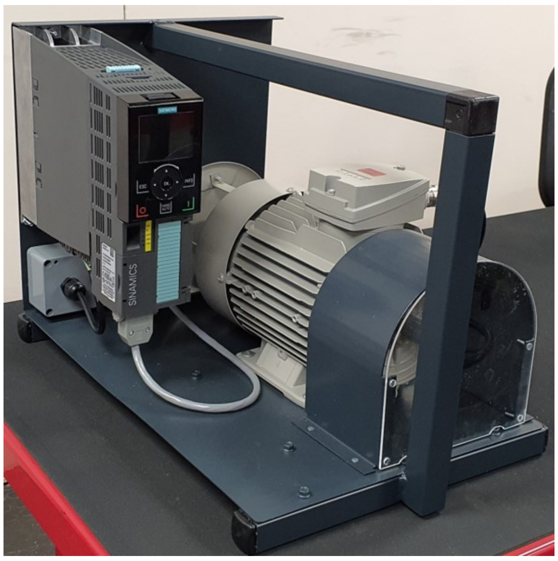


Figure 5.2: Electrical Motor physical set-up

setups of Siemens© S7-1200 PLCs and Siemens© KTP700 Basic HMI panels were used as they were already constructed and tested before this study and no performance or functionality was lost for implementation in this study. The design for a frame setup for the S7-1500 PLCs and Siemens© TP700 HMI panels was still done before the decision to change to the already completed PLC frame setup. The design renders and some design files of the S7-1500 PLCs and Siemens© TP700 HMI panels frame setup is shown with design files in appendix A.

The induction motor, VSD and power module from Siemens© have to be designed for construction and integration in one Motor frame setup. The design thereof is shown in appendix B. The motor frame was constructed, tested and used in the experimental system.

5.4 Device connections

5.4.1 Interfaces

Three interfaces exist in the experimental setup, between the motor's VSD control unit and PLC. The PLC and the IoT gateway; and the IoT gateway and the remote controller. All of the interface connections are ethernet based and described in table 5.1.

Table 5.1: Experimental setup interfaces

	Connection between	Connection type	Communication protocol
IF1	VSD control unit & PLC	Industrial	Profinet interface - Standard telegram
IF2	PLC & IoT gateway	Industrial	Siemens© PUT/GET S7
IF3	IoT gateway & remote controller	Consumer	TCP/IP

The IoT gateway, with two network connections, connects to both the industrial network and the consumer computer network, adhering to both standards of communication and interpreting message from one network to the other.

5.4.2 Information flow

Information flow from the various stations' PLCs to the remote controller is described in figure 5.3 (a). Measured speed values are sent from the PLCs to the IoT gateway through an industrial network, the IoT gateway then assembles the different values into a single message that is sent to the remote controller over a standard computer network with WAN emulation integrated.

Information flow back from the remote controller to each stations' PLC is described in figure 5.3 (b). Reference speeds for each stations' motor determined by the remote controller are sent in a single message to the IoT gateway through the standard computer network with WAN emulation implemented. The IoT gateway then sends each station it's applicable reference speed for implementation over an industrial network connection.

The IoT gateway multiplexes the speed measurement values from each of the PLCs in the systems and sends it to the remote controller. The remote controller transmits the reference speeds for each motor in a single message to the IoT gateway. The IoT gateway then demultiplexes the message containing the reference speeds. It relays the reference speed to each of the appropriate PLCs.

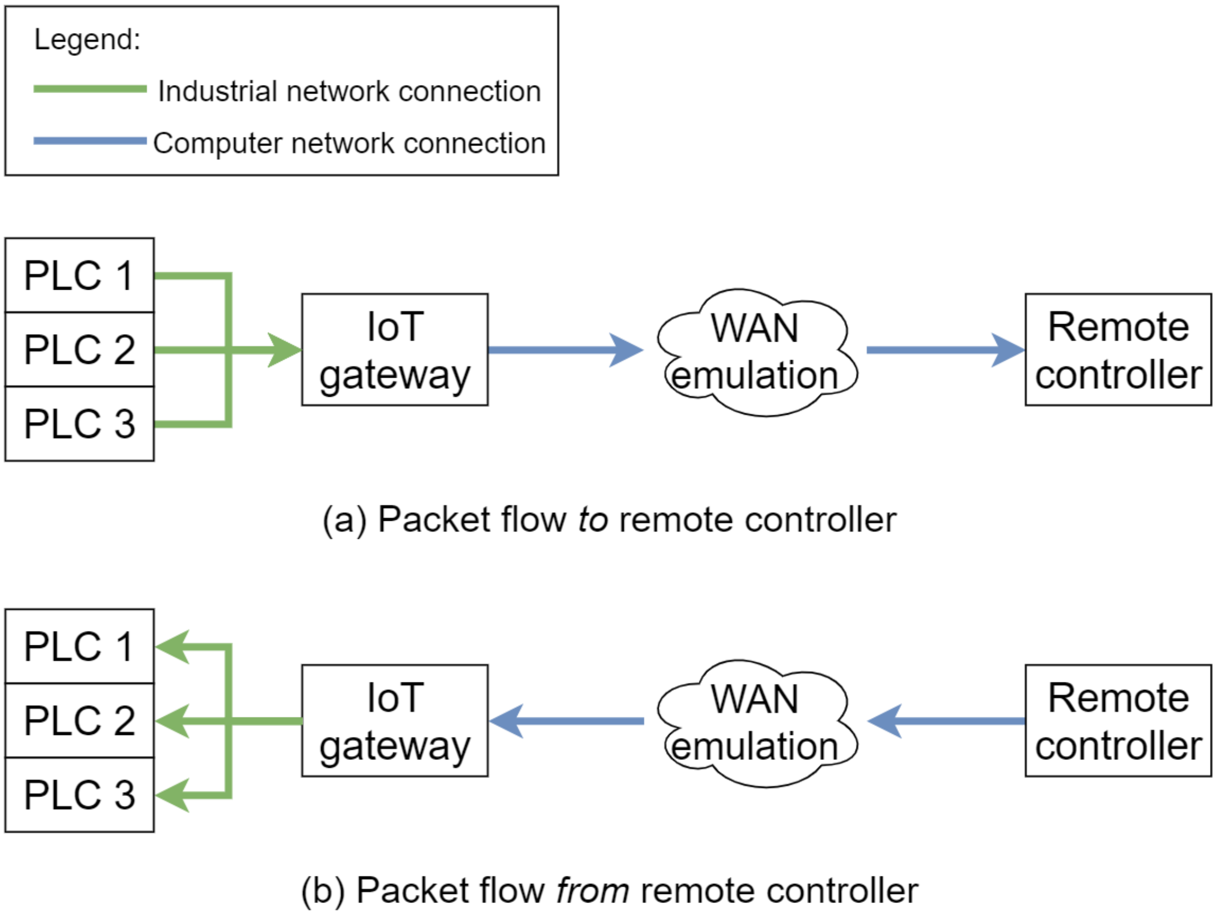

Legend:
Industrial network connection
Computer network connection
PLC 1
PLC 2
PLC 3
IoT gateway
WAN emulation
Remote controller
(a) Packet flow *to* remote controller
PLC 1
PLC 2
PLC 3
IoT gateway
WAN emulation
Remote controller
(b) Packet flow *from* remote controller

Figure 5.3: Information packet flow diagram

5.5 Software

This section describes the software and programming configurations of the PLCs, IoT gateway model and the remote controller server.

5.5.1 PLC automation program

The program for the PLCs to control the motors based on a set point received and sending the measured velocities of each motor to a remote controller is written in ladder logic in Siemens© TIA® (Totally Integrated Automation) portal software.

The automation program allows for remote speed control of three induction motors. Each motor is attached to a power module and a VSD. The VSD is in turn connected to a PLC through a *Profinet* (industrial ethernet standard) connection. The network diagram in TIA for the automation program is shown in figure 5.4. The PLCs are connected to the IoT gateway through an industrial ethernet connection.

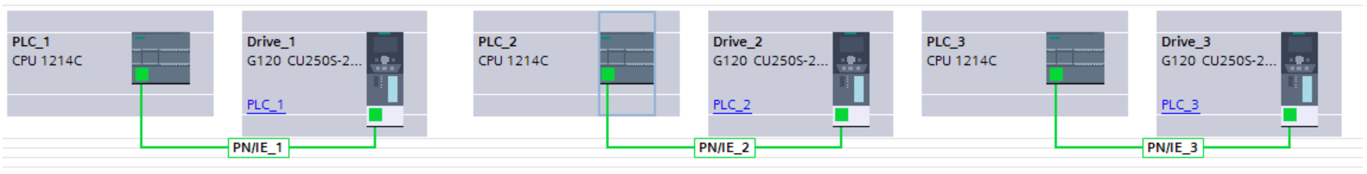


Figure 5.4: Automation program's network view

The automation program is structured to receive a set point for the motor attached to the PLC from a remote controller via an IoT gateway running Node-RED. The set point is sent from the PLC to the VSD and motor for execution. The VSD sends measured velocities of the attached motor to the PLC. The PLC relays the speed to the remote controller through the IoT gateway.

Two interfaces with the PLC exist. The first interface for the PLC is the *Profinet* connection to the VSD and motor setup. The Profinet link achieved by an industrial ethernet cable from Siemens©. The automation program uses the *Drive_Lib_S7_1200_1500* library include with Siemens© *Startdrive*[1]. The *SINA_SPEED* function block from the library is used to send speed set points to the VSD and to receive velocity measurements back from the VSD.

The second interface is the connection between the IoT gateway running Node-Red and the PLC. The IoT gateway and each PLC is linked with an industrial ethernet connection. Communication between the PLC and the IoT gateway is done through Siemens© *PUT/GET S7 protocol*. Preconditions for communication between the PLC and Node-Red on the IoT gateway is the following: The values should be stored in non-optimised data blocks on the PLC, and PUT/GET communication on the PLC must be enabled.

PLC values to read and write into by Node-Red is stored as variables in a non-optimised data block, "S7_connection_Values". The variables and their function is shown in table 5.2.

Table 5.2: Communication variable between the PLC and the IoT gateway

Variable	Data Type	Function
I_On	Boolean	Switches opearation on or off
Set Point	Real	Motor velocity set point
ActVelocity	Real	Measured motor velocity

The PLC's operation can start by two different means. The first is by switching a toggle switch on the PLC frame to 'on'. The second is by sending the value 'true' from the IoT gateway to the PLC using variable 'I_On' described in table 5.2. The network in the automation program in the main organization block to switch on the operation of the PLC is shown in figure 5.5.

Once the PLC's operation is started, the PLC sends a received set point to the VSD and motor for execution. The VSD and motor setup sends measured velocities of the motor to the PLC.

The PLC receives motor set points from a remote controller via the IoT gateway. The PLC transmits

[1]Siemens© Startdrive is commissioning software required for the integration of drives in automation within TIA Portal.

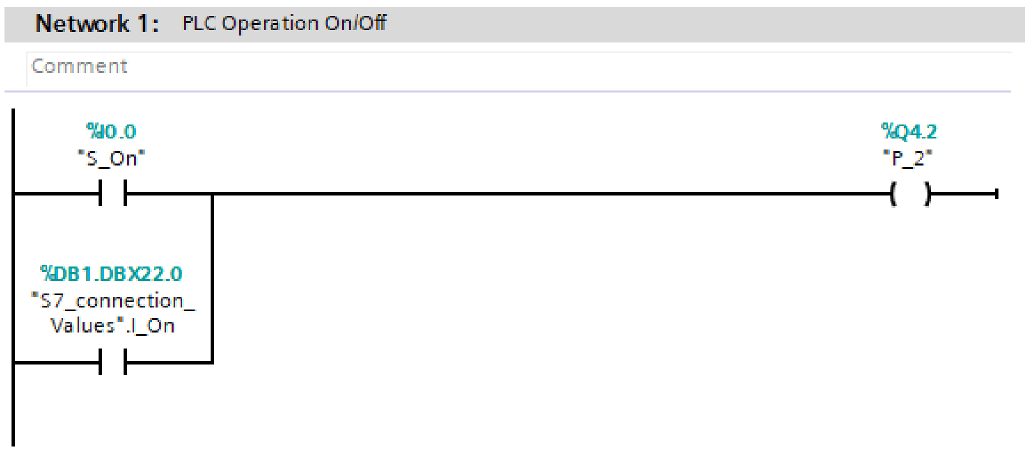


Figure 5.5: PLC automation program - PLC operations switch

measured velocities back to the remote controller via the IoT gateway. Variables used for the transmission of set points and measured speeds are explained in table 5.2. The variables used are stored, accessed and transmitted through data block “S7_connection_Values”. Changes to the variables are transmitted in 100ms intervals. The remote controller then does control of the interconnected three-motor system. The automation program’s network for speed control of the motors is shown in figure 5.6.

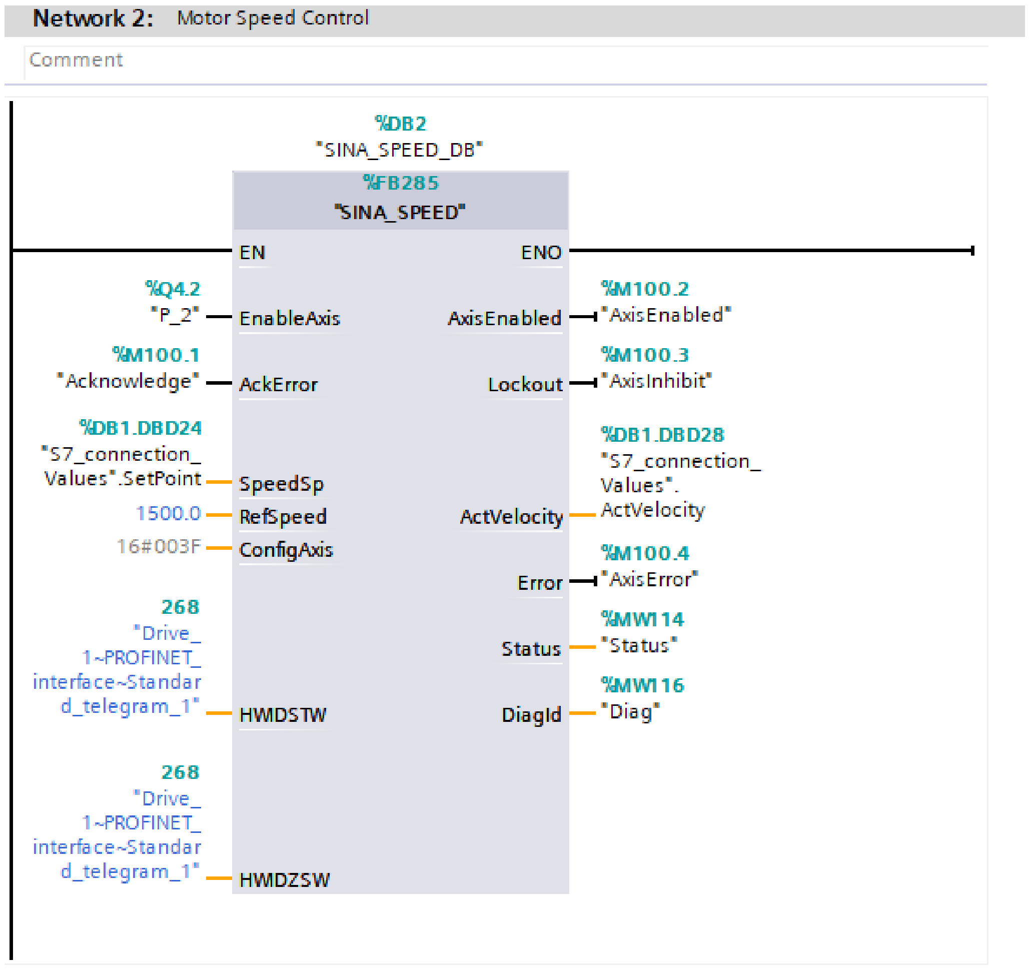


Figure 5.6: PLC automation program - PLC motor speed control

5.5.2 IoT gateway module

The Siemens© IoT2040 uses the Yocto Linux distribution for its operating system. The Node-RED application runs as a service on the module. Node-RED is a programming environment for event-driven applications.

Two interfaces to the IoT gateway exists. The first interface is an industrial ethernet connection to the PLCs using the Siemens© PUT/GET S7 protocol. The second is a consumer ethernet connection to the remote controller server, using the TCP/IP protocol. The IoT gateway, with it's two separate network connections, connects to both an industrial network and a consumer network for communication.

The main objective of the IoT gateway is to gather measured motor speed from the PLCs and send it to the remote server over a consumer network. The IoT gateway also receives control data from the remote server that is split by the IoT gateway and sent to the appropriate PLC. The flow created in Node-RED for the IoT gateway's operation is shown in figure 5.7.

The IoT gateway receives information from the PLCs using the '*node-red-contrib-s7*' node, communicating with the Siemens© PUT/GET S7 protocol over the industrial network. Values to read and write into by Node-Red is stored in a non-optimised data block. Changes to the values are updated every 50 ms.

The values are sent to Node-Red through the PUT/GET S7 protocol. The collected data from the PLCs is first formated to add the PLC number to the appropriate value. The formatted values from the PLCs are then concatenated into a CSV (Comma-separated values) line. The CSV line is then sent to the remote server via a TCP/IP (port: 100000) connected over a consumer network. Each motor's received velocity measurements are logged locally to CSV files for extraction and analysis of each test run.

The IoT gateway receives control information from the remote server as a CSV line through the TCP/IP connection. The CSV line contains set points for each motor determined by the remote controller. The CSV line is split, and the relevant control information is sent to its respective PLC, using the Siemens© PUT/GET S7 protocol, for execution.

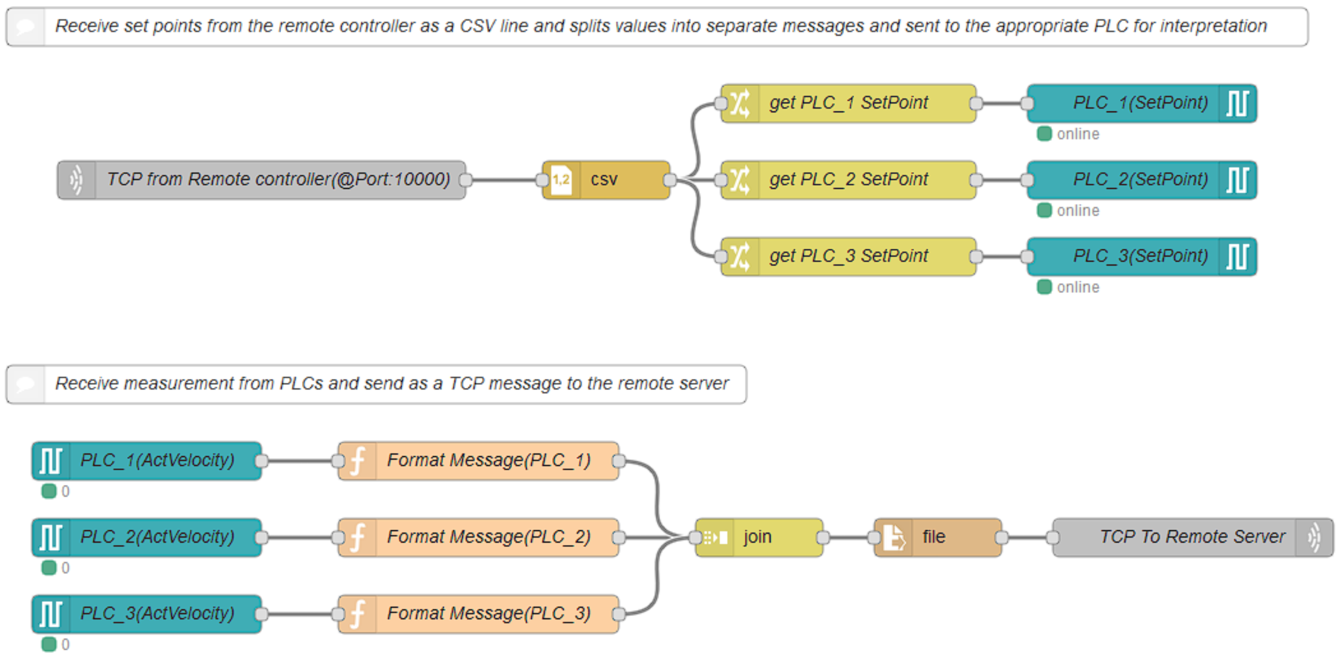


Figure 5.7: Node-RED program flow diagram

5.5.3 Remote controller software

The remote control algorithm runs as a Python^{TM} (version 3.8.5) script, receives motor speed values, via an IoT gateway, from the local controllers attached to the motors. The remote controller interprets the values and calculates set points based on the information received and a pre-programmed control scheme. The reference speed set points are sent back to the motor's local controllers, via the IoT gateway for execution. The process is then repeated until the control scheme is finished. The flow diagram for the remote controller's operation is shown in figure 5.8. The phyton script code for the remote controller is shown in appendix C.

The remote controller program starts by initialising variables and functions required for its operation. The remote controller creates a TCP/IP socket connection to the IoT gateway and sends 0 r/min set point to reset the operation of each motor.

The transmission function to send set point from the remote controller to the IoT gateway takes three arguments, one for each motors set point. The set points are then concatenated in a CSV formatted line and formated into a byte-literal string required for TCP/IP communication. The formatted line is then sent through the TCP/IP connection to the IoT gateway.

Data received from the IoT gateway through the TCP/IP connection is done through a receiving function. The function decodes the received data from a byte-literal string to a string. The PLC numbers and speed values are then extracted from the string using regular expressions and stored into an array. The array, therefore, contains the PLC and consequently the motor's number as well as the motor's measured speed.

The set point is then calculated based on the speed of each motor received and the control scheme. Once the set points are estimated, they are sent to the IoT gateway through the transmission function.

Timer interrupts are used to implement the control scheme, as described in the control scheme section, for motor 1. The first interrupt at 0.4 seconds sets the motor 1 set point to 1500 r/min. The second interrupt at 3.0 seconds sets the motor 1 set point to -1500 r/min. The third and final interrupt at 7.0 second concludes the test, copy motor speed log files from the IoT gateway, and closes the program.

The tests for determining the performance impact of different network conditions on the IIoT system is done for system implementations with and without delay mitigations. Two different delay mitigations are implemented and tested. When delay mitigation is implemented on the remote controller, the received values from the IoT gateway is sent through either an 'exponential moving average' or a 'double exponential smoothing model'. The predicted value from the implemented delay mitigation is then used in the control scheme.

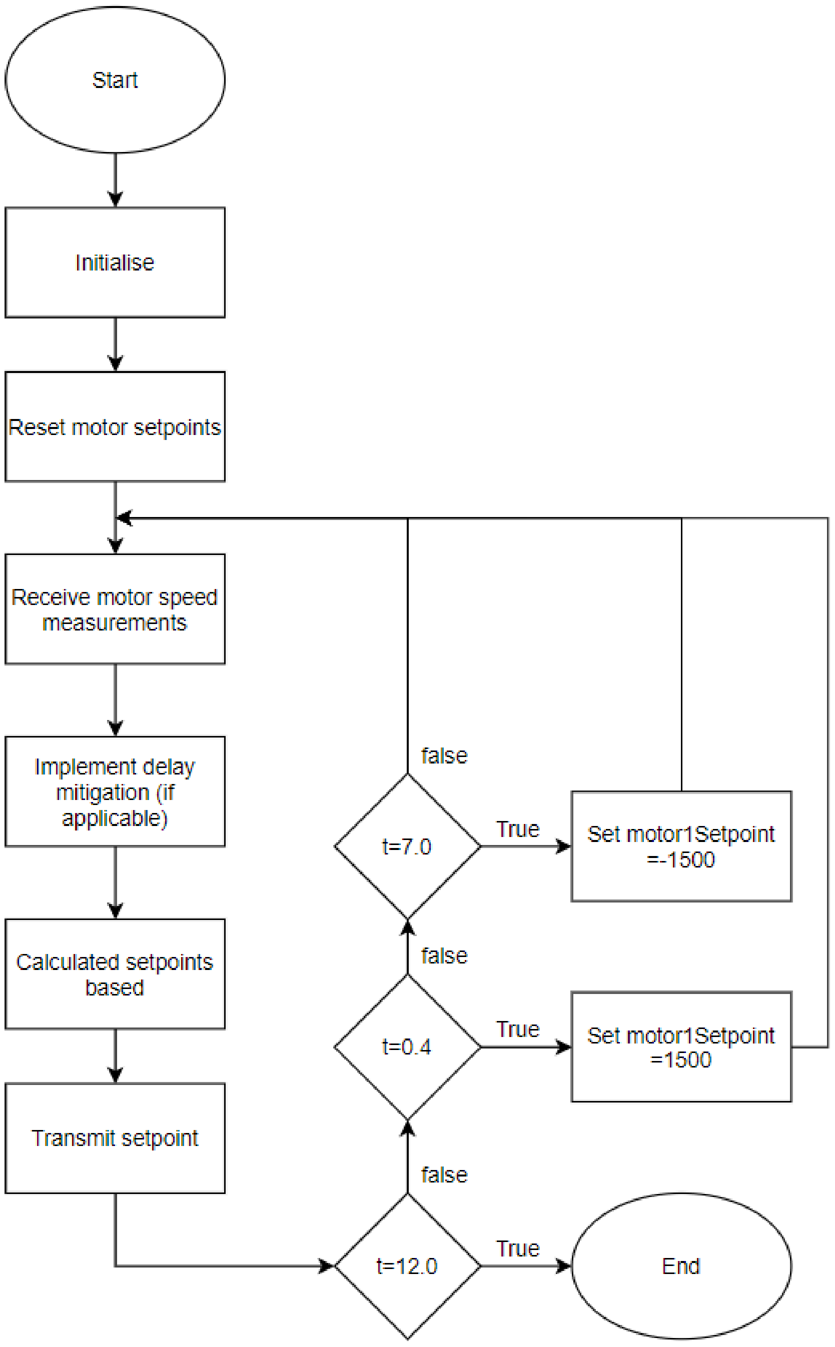

Start
Initialise
Reset motor setpoints
Receive motor speed measurements
Implement delay mitigation (if applicable)
Calculated setpoints based
Transmit setpoint
t=12.0
True
End
false
t=0.4
True
Set motor1Setpoint =1500
false
t=7.0
True
Set motor1Setpoint =-1500
false

Figure 5.8: Remote controller program flow diagram

5.6 Physical system control scheme response

As before the control scheme, as described in the experimental design chapter, is tested on the experimental system. The response expected from the system is as stated with the control scheme, but with a delay between the outputs of the motors. The addition of the sampling period in the system results in the delay between the motor outputs. An additional increase in delay between the motor outputs can also be expected due to network conditions and overheads present on the experimental system. The motor speed responses of the system, as measured in the system by applying the control scheme, is shown in figure 5.9. The experimental system uses a sampling period of 100 ms. The test of the control scheme on the experimental system was done with best-case network conditions. The figure indicates that the measured response of the experimental system is as expected when subjected to the control scheme for the system.

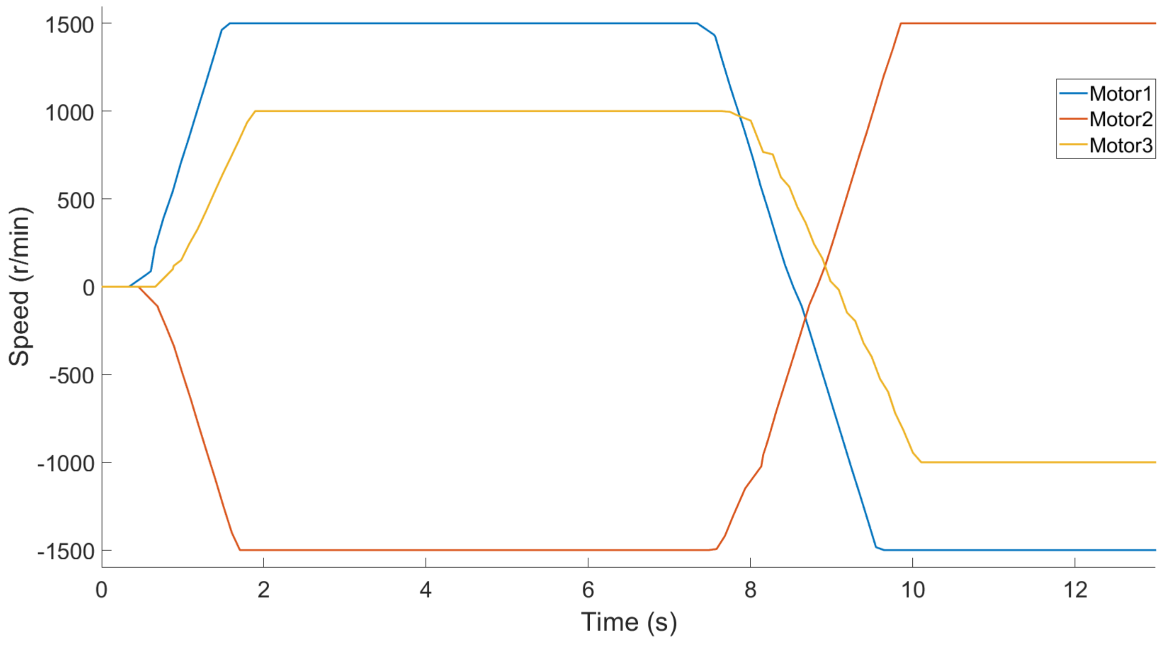


Figure 5.9: Speed graph - Experimental system control scheme response

5.7 Conclusion

In this chapter, the experimental system setup is described in terms of its hardware, inter-device connections, software and overall operation. The chapter describes the Siemens© industrial equipment and the setup thereof in the mechatronics laboratory. The experimental system was constructed based on the design done for the experiment. The goal of the experimental system is to determine and evaluate the effects of different network conditions on the performance of an IIoT control and monitoring system. The control scheme was then implemented on the experimental system to ensure the correct operation of the experimental system. The system performed as expected when the control scheme was implemented. The experimental setup is used with the network modifier to conduct the four tests, described in the experimental design, to determine the effects of network condition on such an IIoT based system. The results obtained for each test is logged and stored for comparison and evaluation.

Chapter 6

Results and analysis

6.1 Introduction

The completion of both the simulations model and the experimental physical system allows for performance impact testing of network conditions on the systems. The systems are subjected to different network conditions, as stated in the research approach in chapter three.

This chapter explores, examines and analyses the results gathered on the effects of network conditions on the performance of the IIoT control and monitoring system.

The results are presented in the order of the tests. The first section shows the impact of moving the remote controller of the system from a LAN to a WAN. The second section describes the impact of delays on the IIoT system with and without delay mitigations implemented. The third section shows the effects of both delays and jitter on the performance of the IIoT system. The fourth section displays the results of network packet loss on the performance of the system. The fifth section describes the effects of latency and jitter on the performance of the IIoT system's simulation model. The results obtained from the simulation model are discussed first followed by the results from the experimental physical system. Validation of the results between the two systems is then analysed. A discussion on the results obtained is then given, followed by a final section providing a conclusion on the chapter. The average system time for the two systems and the various tests is shown in appendix D.

6.2 Simulation model's reaction to different network conditions

The digital simulation model of the IIoT system is subjected to different simulated network conditions to determine the performance impact of different network conditions on the average system time of the IIoT experimental system. The system simulation model is implemented with and without delay mitigations. The simulation model allows for the addition, modification and testing of network conditions such as network location, latency and jitter. The simulation model does not take into account any additional overheads that may occur within the system.

6.2.1 Performance impact based on the network location of the remote controller.

Firstly the impact of moving the system controller from local to remote is measured and determined. The remote controller is tested with simulated LAN and WAN connections. The performance impact of the position of the system's controller is shown as a bar graph in figure 6.1. The figure from left to right show the average system response time for a local connection, a LAN connection, a WAN connection, a WAN connection with an exponential moving average (EMA) delay mitigation, and finally a WAN connection with double exponential smoothing model delay mitigation (DESM).

The value shown for the response time of a local systems controller is due to the system's sampling rate of 100 ms, resulting in one action every 100 ms. The LAN connected remote controller adds additional delays (0.08 ms or 80%) due to the addition of network and routing latencies. The move from a LAN to a WAN connection introduce even more substantial network and routing latencies, thus increasing the system response times by 0.1836 ms or 102%.

Two different delay mitigations are tested on the WAN connected remote controller in order to try and mitigate the effects introduced by the network and routing of communication over the network. The first delay mitigation, EMA, proved ineffective for the system and resulted in a larger average system response time of 0.01 ms or 3% and decreased the performance of the system compared to the system with no delay mitigations. The second delay mitigation, DESM, proved effective and capable by improving system performance and decreasing system response times, a decrease of 0.15 ms or 41%.

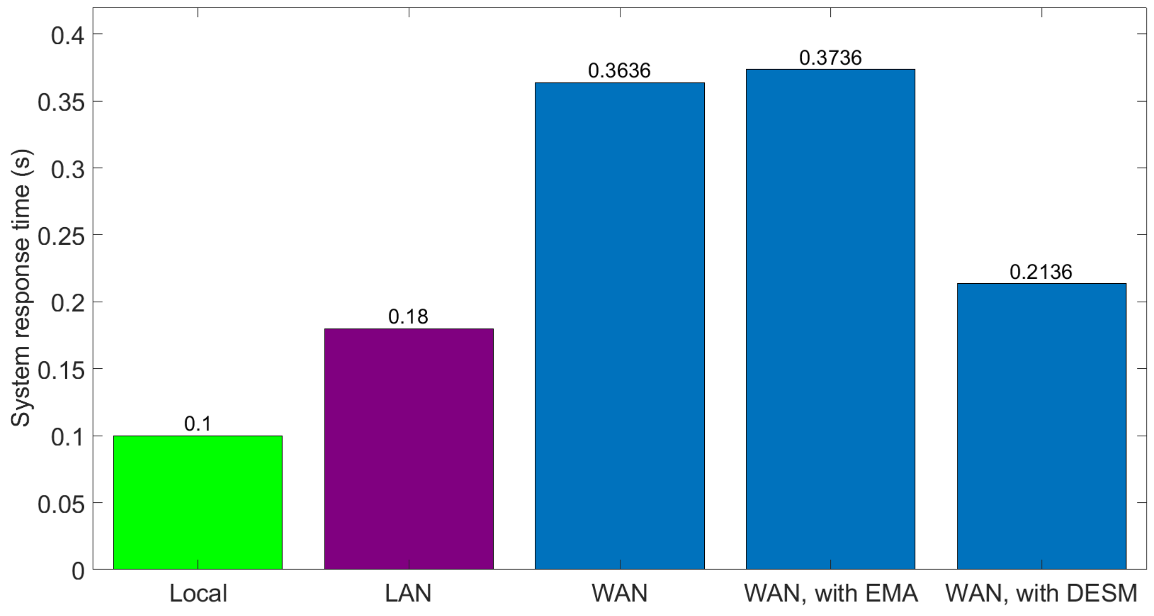


Figure 6.1: Average simulation model system response time bar chart - Impact of remote controller's network position

6.2.2 Performance impact of network latency

The second test measures and determines the effect of network latency on the performance and response time of the experimental system. The result obtained for the simulation model is presented in figure 6.2 and shows the impact of different network latencies on the average system response times. The figure shows the average system response times for the system with and without the two different delay mitigations implemented. The figure shows a clear linear relationship between network latency and the average system response time, as network latency increase, so does the system response time.

The two delay mitigations are implemented to try and curb the network effects. The first delay mitigation proved ineffective and caused a decrease in system performance compared to the system with no delay mitigations. The second delay mitigation, DESM, improved system performance for each of the tested network latencies compared to the system without delay mitigations. The system response times were on average 0.15 ms or 24% faster for the system with DESM as delay mitigation and on average 0.021 ms or 2% slower for the system with EMA as delay mitigation when compared to the system with no delay mitigations.

Both delay mitigation implementations on the simulation model of the experimental system also indicate a positive linear or slightly quadratic relationship between network latency and the average system response time. This is confirmed by applying polynomial quadratic and linear curve fitting to the results obtained on the system response time without delay mitigations implemented. The quadratic curve fitting and linear curve fitting is shown in (6.1) and (6.2) respectively, where τ is the network latency round trip time (RTT) in milliseconds, and $f(\tau)$ is the systems response time of the system for the given network latency is seconds. The root mean square error (RMSE) for the polynomial fitting is 0.02867 s and 0.04509 s for the linear curve fitting. The curve fitting indicates with the linear fitting and small quadratic term on the polynomial fitting that the effects of network delays on the performance of the IIoT system are mostly linear.

$$f(\tau) = 1.387^{-6}\tau^2 + 0.001709\tau + 0.3346 \tag{6.1}$$

$$f(\tau) = 0.002398\tau + 0.2877 \tag{6.2}$$

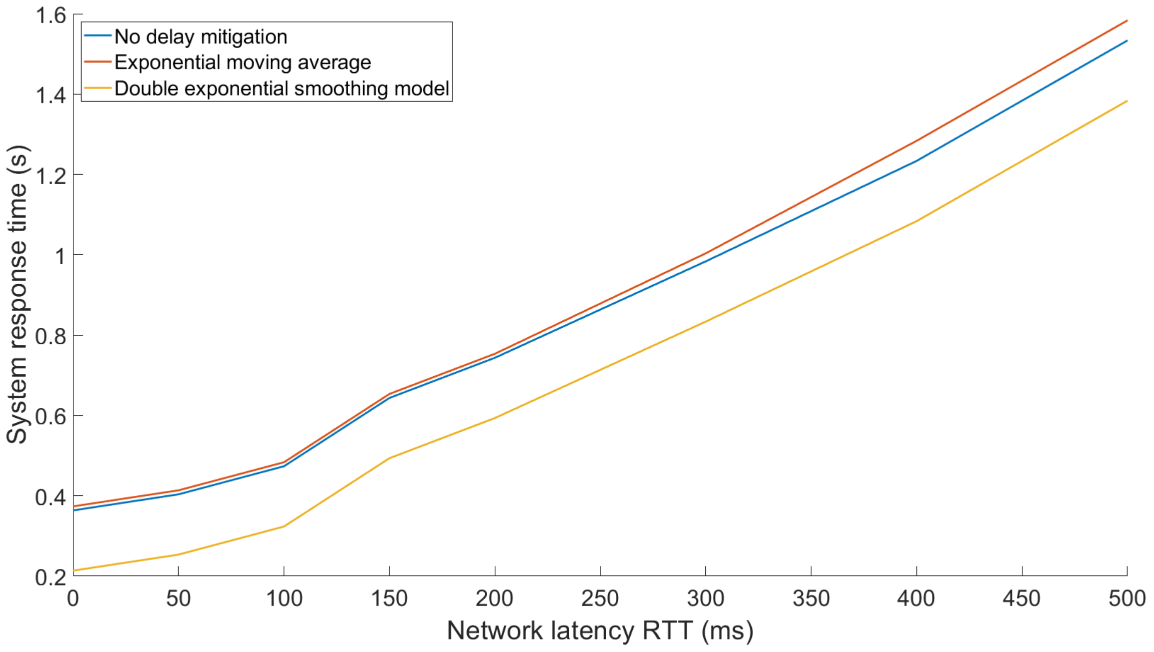


Figure 6.2: Average simulation model system response time line graph - Impact of network latency

6.2.3 Performance impact of network jitter

The third test for measuring and determining the average system performance impact of network jitter on the experimental IIoT system is done on the simulation model.

The average system response time for different network latencies and jitter values is shown in figure 6.3. The figure shows the system response time for different network latencies, and each line in the figure represents an additional applied network jitter value for a system without any delay mitigations implemented. An increase in system response time is observed for each latency test point when jitter is introduced and increased, as evident in the figure where each line in the line graph represents a different jitter value. A higher jitter value results in a higher line, and thus a larger system response time. An increase in system response times means a decrease in system performance.

The average system response measured for a jitter value over the range of tested network latencies is shown in figure 6.4. The figure shows the response of the system for a jitter value by taking the average system response time over the range of network latencies. The figure shows the response for a system without delay mitigation and the systems with the two different delay mitigations implemented. The system with EMA implemented proved that EMA is ineffective for use in the system as it increased system response time and therefore decreased system performance at each tested jitter interval. The system with EMA as delay mitigation performance on average 3% worse than the system without any delay mitigations The system with DESM as delay mitigation decreased system performance by a significant margin for each jitter test point. On average, the system with DESM as delay mitigation performed 19% better than the system with no-delay mitigation.

The results obtained for the performance impact of net jitter on the experimental IIoT system, shown in figure 6.4, indicates a significant increase in the average system response time when jitter is introduced to the system. After that, a linear relationship is observed between network jitter and the average system response time. When network jitter is increased in the system, the average system response time is also increased.

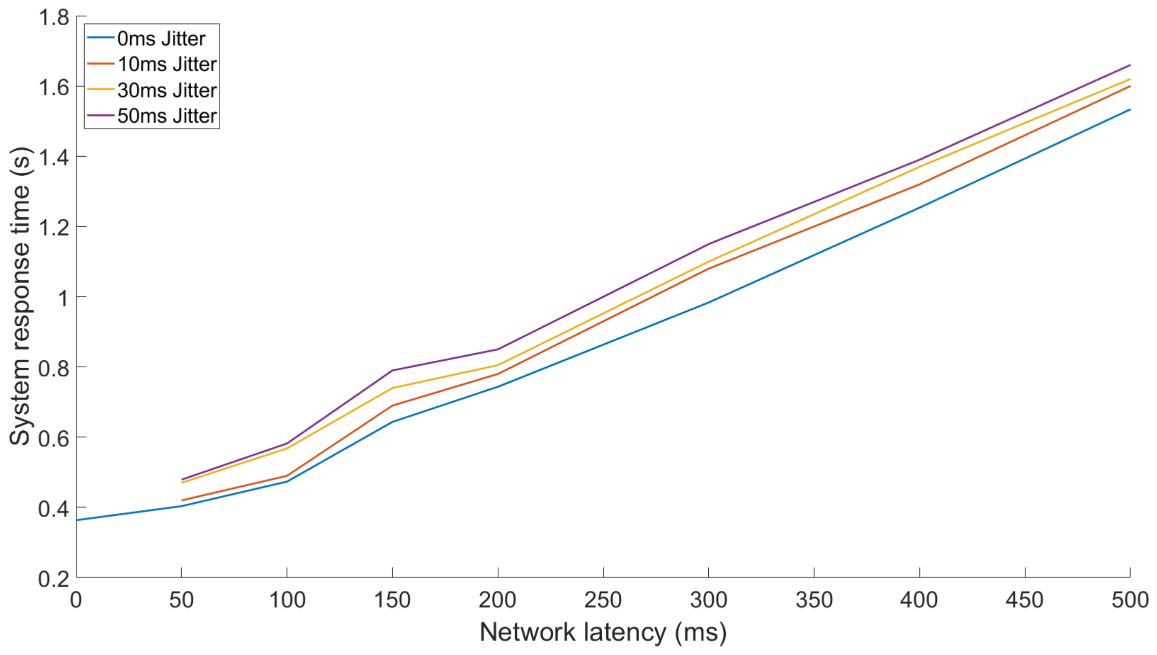


Figure 6.3: Average simulation model system response time line graph - Impact of network latency and jitter

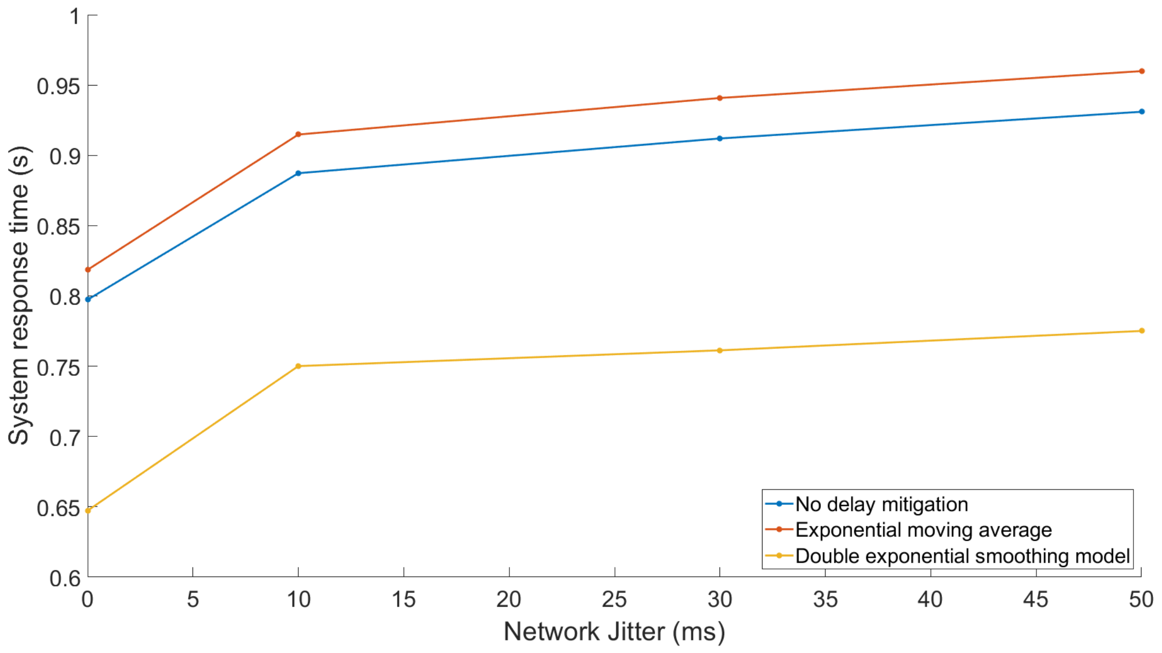


Figure 6.4: Average system response time line graph - Impact of network jitter

6.3 Experimental system's response to different networking conditions

The physical experimental IIoT system is subjected to different emulated network conditions to determine the performance impact of the different network conditions on the average system time and performance of the IIoT system. The experimental system can be deployed with and without delay mitigations, where EMAs and DESMs are used for delay mitigations The experimental system along with the network emulator and modifier, WANem, allows for the addition and alteration of network conditions such as network location, latency, jitter and packet loss. The experimental system is subjected to different network conditions, as stated by the tests created in the research approach to determine the effects.

6.3.1 Performance impact based on the network location of the remote controller

Figure 6.5 shows the results obtained from **Test one**, described in section 3.6, presenting the impact of moving the remote controller of the system from a local area network (LAN) to a wide area network (WAN). The left bar in the bar graph indicates the system response time when the remote controller is implemented natively on the IoT gateway, emulating a remote controller on a LAN connection. The bars to the right show the response time of the system when the remote controller is connected via a WAN connection. The effects on system performance are shown when no delay mitigations are implemented as well as when delay mitigations are implemented. The different delay mitigations implemented are exponential moving average (EMA) and double exponential smoothing model (DESM).

An increase of 0.187 seconds or 102% in system response time is observed when moving the remote controller from a LAN to a WAN connection without any delay mitigations implemented on the remote controller. An increase of 0.199 seconds or 108% is observed when an exponential, moving average is implemented. When the double exponential smoothing model is implemented on the remote controller, an increase of only 0,047 seconds or 26% in system response time is observed when the remote controller is moved from a LAN to a WAN connection.

When comparing the different algorithms on a WAN connected IIoT system, the following was observed. The exponential moving average as delay mitigation algorithm performed 3% (0.012 seconds) slower than the system with no delay mitigations implemented. However, the system with double exponential smoothing model implemented as delay mitigation algorithm performed 38% (0.140 seconds) faster.

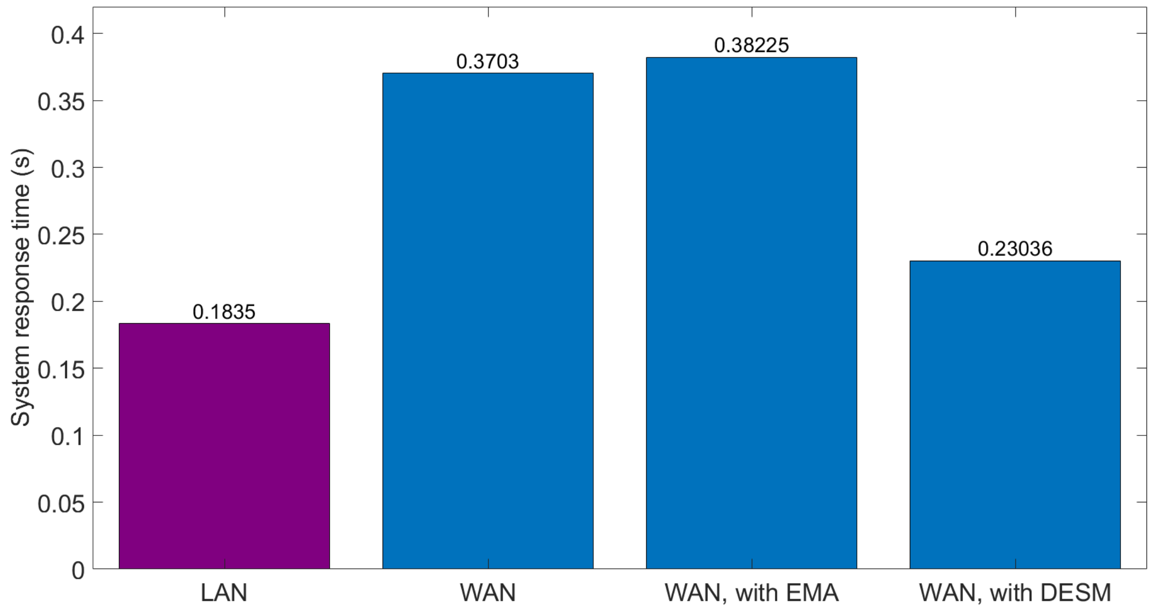


Figure 6.5: Average system response time bar chart - Impact of remote controller's network position

6.3.2 Performance impact of network latency

The results obtained from **Test two**, described in section 3.6, is presented in figure 6.6 showing the average system response time for different network latencies. The figure shows the effect of network latency on the response time of the system with and without delay mitigations implemented. Delay mitigations used are Exponential Moving Average and Double Exponential Smoothing Model. From the graph, it is evident that as network latencies increase so does the system response time. It is also apparent from the graph that the first delay mitigation algorithm, exponential moving average, yields lower performance than that of the system with no delay mitigations implemented. In contrast, the performance is increased when the second delay mitigation algorithm is implemented.

Applying polynomial and linear curve fitting to the effects that network latency has on the system response time without delay mitigations implemented is shown in (6.3) and (6.4), respectively, where τ is the network latency round trip time (RTT) in milliseconds, and $f(\tau)$ is the systems response time of the system for the given network latency is seconds. The root mean square error (RMSE) for the polynomial fitting is 0.0293 seconds and 0.0429 seconds for the linear curve fitting. The curve fitting indicates with the linear fitting and small quadratic term on the polynomial fitting that the effects of network delays on the performance of the IIoT system are mostly linear. As network delays are increased, so does the system response time linearly or slightly quadratic.

$$f(\tau) = 1.266e^{-6}\tau^2 + 0.00177\tau + 0.3415 \tag{6.3}$$

$$f(\tau) = 0.002399\tau + 0.2987 \tag{6.4}$$

The average system response times in milliseconds for the different delay mitigation systems for each network delay RTT in milliseconds is shown in table 6.1. Additionally, the difference between not implementing delay mitigation and implementing the additional delay mitigations is indicated as a percentage value. A positive percentage value indicates an increase in system response time and therefore, a decrease in system performance, while a negative percentage value indicates a decrease in system response time and an increase in system performance. Abbreviations used in the table are as follows:

- No-DM - No delay mitigation algorithm
- EMA - Exponential moving average delay mitigation algorithm
- DESM - Double exponential smoothing model delay mitigation algorithm
- No-DM Vs. EMA - No delay mitigation compared to Exponential moving average delay mitigation algorithm
- No-DM Vs. DESM - No delay mitigation compared to Double exponential smoothing model delay mitigation algorithm

The values show that exponential moving average as delay mitigation is on average 4% slower than no delay mitigations implemented, with values ranging between 1 and 9% slower. While double exponential smoothing model as delay mitigation algorithm yields on average 23% better performance with values ranging between 9 and 39% better performance.

Table 6.1: Impact values of network delays on the IIoT system

Delay(ms)	No-DM (s)	EMA (s)	DESM (s)	No-DM Vs. EMA (%)	No-DM Vs. DESM (%)
0	0.3703	0.3823	0.2304	3	-38
50	0.4114	0.4245	0.2515	3	-39
100	0.4882	0.5018	0.3435	3	-30
150	0.6526	0.6579	0.4920	1	-25
200	0.7625	0.7740	0.6402	2	-16
300	1.0033	1.0211	0.7911	2	-21
400	1.2345	1.3488	1.1248	9	-9
500	1.5455	1.6368	1.4026	6	-9
Averages	0.8085	0.8434	0.6595	4	-23

An average system response time of 0.3703 seconds as per the default system time with no added network conditions and no delay mitigation implemented is used to determine the performance cost of network latency on the IIoT system. The performance cost compared to network latency is shown in table 6.2, where a positive value represented a decrease in system performance and vice versa for a negative value.

Table 6.2: System performance cost of network delay

Delay(ms)	No-DM (%)	EMA (%)	DESM (%)
50	11	15	-32
100	32	36	-7
150	76	78	33
200	106	109	73
300	171	176	114
400	233	264	204
500	317	342	279

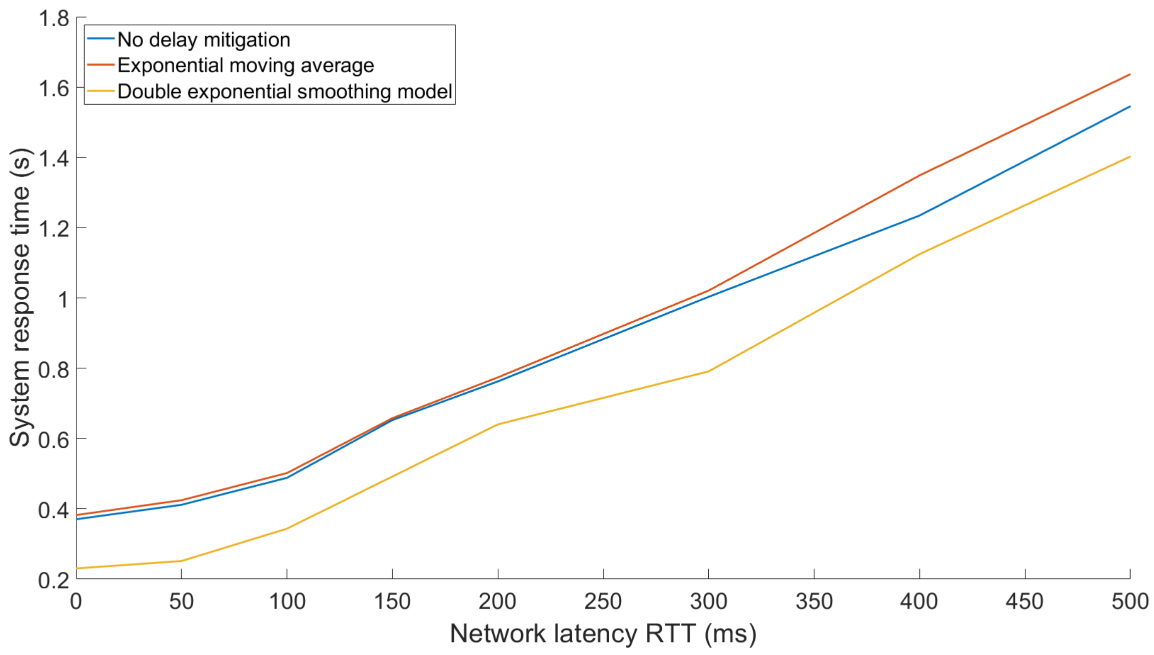


Figure 6.6: Average system response time line graph - Impact of network latency

6.3.3 Performance impact of network jitter

Jitter or packet delay variance, measured in milliseconds, is a variance in packet delays. Results from **Test three**, described in section 3.6, are presented in figure 6.7 showing the average effect of jitter on average system response time. The average value represented for each jitter test point is determined as the moderate impact of that jitter test point for all the tested latency intervals. The figure shows the effects of jitter on a system without delay mitigations as well as the system with delay mitigations implemented.

As before, an increase in system time is observed when jitter within the network is increased. Reasonable jitter test values have been chosen for testing based on jitter values that commonly occur in practice. Jitter values are smaller than the sampling period, 100 ms, of the experimental system, but have been applied to latencies larger than the sampling period of the IIoT system. A substantial increase of 15% (0.1238 seconds) in system response time is observed when moving from 0 ms jitter to 10 ms jitter. After that, the effects of jitter on the system response time can be approximated by (6.5) with an RMSE of 0.0008 seconds, obtained by applying linear curve fitting to the data points. In (6.5) $f(\tau)$ represents the system response time in milliseconds and τ the network jitter value in milliseconds.

Figure 6.7 also indicates that the exponential moving average delay mitigation algorithm performed worse than without any delay mitigations. In contrast, the system with the double exponential smoothing model increased performance by decreasing the system response time. On average exponential moving average performed 2% worse than the system without delay mitigations and double exponential smoothing model yielded on average 19% better performance than the system with no delay mitigations.

$$f(\tau) = 0.001282\tau + 0.9192 \tag{6.5}$$

Figure 6.8 shows the effect of jitter, for various network latencies, on the IIoT system's response times without any delay mitigations implemented. Each line on the line graph represents a different jitter value as described on the graph's legend. From the graph, an overall increase is observed when jitter is increased for a given latency.

With an average system response time of 0.3703 ms as per the default system time with no added network

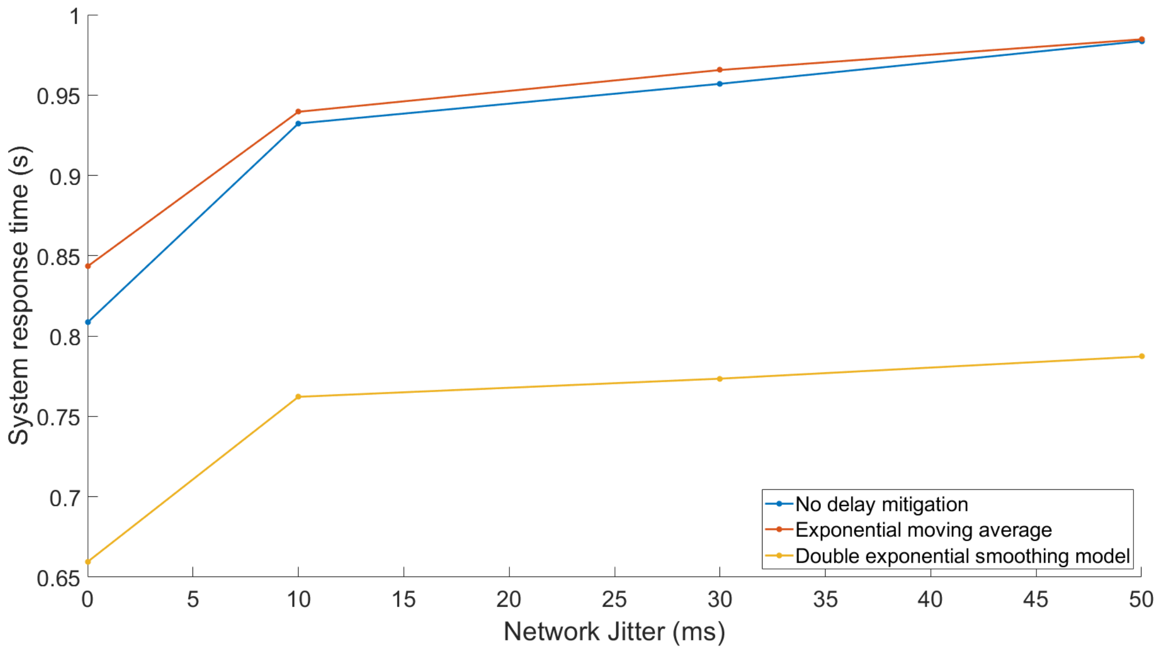


Figure 6.7: Average system response time line graph - Impact of network jitter

conditions and no delay mitigation implemented is used to determine the performance cost of jitter for various latencies on the IIoT system without any delay mitigations implemented. The performance cost calculated is shown in table 6.3, where a positive value represented a decrease in system performance and vice versa for a negative value.

Table 6.3: Performance cost of network jitter, per latency on the IIoT system withour delay mitigations

Jitter (ms) / Delay (ms)	0	10	30	50
50	11%	16%	28%	29%
100	32%	33%	58%	58%
150	76%	95%	100%	114%
200	106%	113%	114%	133%
300	171%	196%	200%	196%
400	233%	268%	272%	278%
500	317%	342%	339%	351%

6.3.4 Performance impact of packet loss

Packet loss represented as a percentage of packet transmissions that are lost during transmission. The impact of network packet loss on the performance of the IIoT system, as results obtained from **Test four**, is shown in figure 6.9 for the system with no delay mitigations implemented as well as when exponential moving average and double exponential smoothing model is implemented as delay mitigation algorithms. The impact of packet loss increased the system response times of the IIoT system in a quadratic manner. Applying a polynomial Quadratic curve-fitting on the system without delay mitigations yields (6.6) with an RMSE of 0.00154 seconds, that can be used to approximate the system response time based on a given

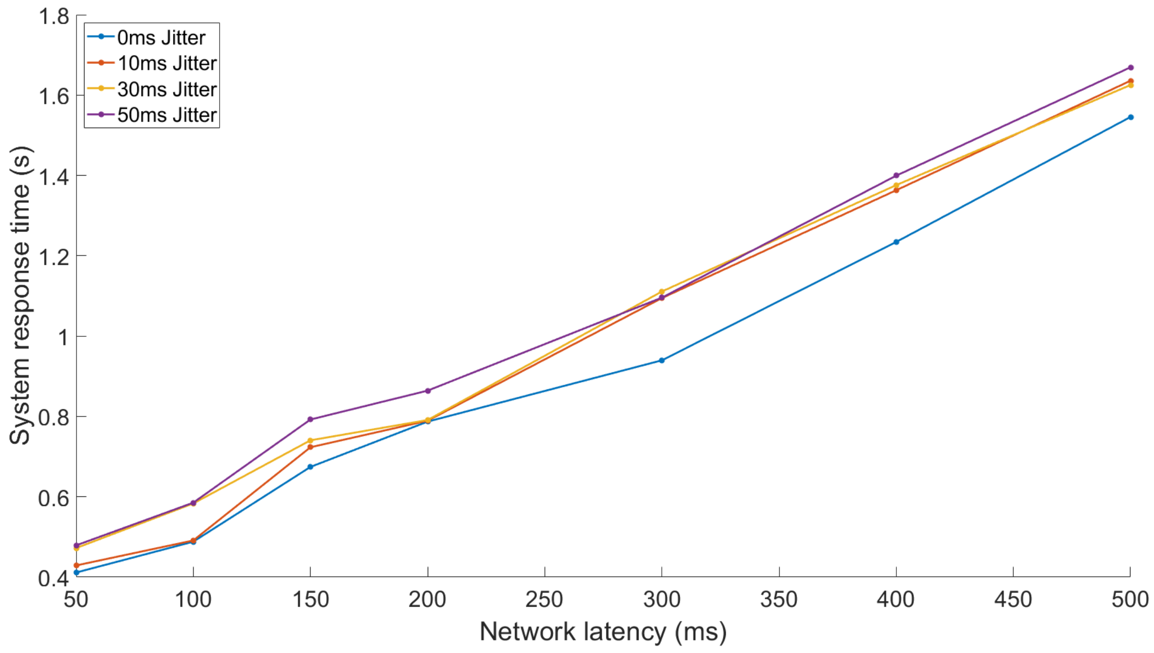


Figure 6.8: Average system response time line graph - Impact of jitter for various latencies

percentage of network packet loss.

$$f(\tau) = 0.001169\tau^2 + 0.01115\tau + 0.3698 \tag{6.6}$$

The average system response time for each tested network packet loss percentage is shown in table 6.4. The difference between not implementing any delay mitigations on the remote controller and the different delay mitigations tested is shown in as a percentage representation. A positive percentage value dictates an increase in system response time and thus a decrease in system performance and vice versa for a negative value .Abbreviations used in the table are as follows:

- No-DM - No delay mitigation algorithm
- EMA - Exponential moving average delay mitigation algorithm
- DESM - Double exponential smoothing model delay mitigation algorithm
- No-DM Vs. EMA - No delay mitigation compared to Exponential moving average delay mitigation algorithm
- No-DM Vs. DESM - No delay mitigation compared to Double exponential smoothing model delay mitigation algorithm

The table shows that exponential moving average as delay mitigation is on average 12% slower than no delay mitigations implemented with regards to packer loss. The double exponential smoothing model as delay mitigation algorithm yields on average 25% better performance when dealing with packet loss over no delay mitigations implemented.

An average system response time of 0.3703 ms as per the default system time with no added network conditions and no delay mitigation implemented is used to determine the performance cost of packet loss on the IIoT system. The performance cost compared to network packet loss is shown in table 6.5, where a positive value represented a decrease in system performance and vice versa for a negative value.

Table 6.4: Impact values of network delays on the IIoT system

Packet loss(%)	No-DM (s)	EMA (s)	DESM (s)	No-DM Vs. EMA (%)	No-DM Vs. DESM (%)
0	0.3703	0.3823	0.2304	3	-38
1	0.3811	0.4253	0.2430	12	-36
4	0.4345	0.5055	0.3203	16	-26
7	0.5039	0.5768	0.4212	14	-16
10	0.5985	0.6694	0.5366	12	-10
Average	0.4576	0.5119	0.3503	12	-25

Table 6.5: System performance cost of network packet loss

Packet loss(%)	No-DM (%)	EMA (%)	DESM (%)
1	3	15	-34
4	17	37	-14
7	36	56	14
10	62	81	45

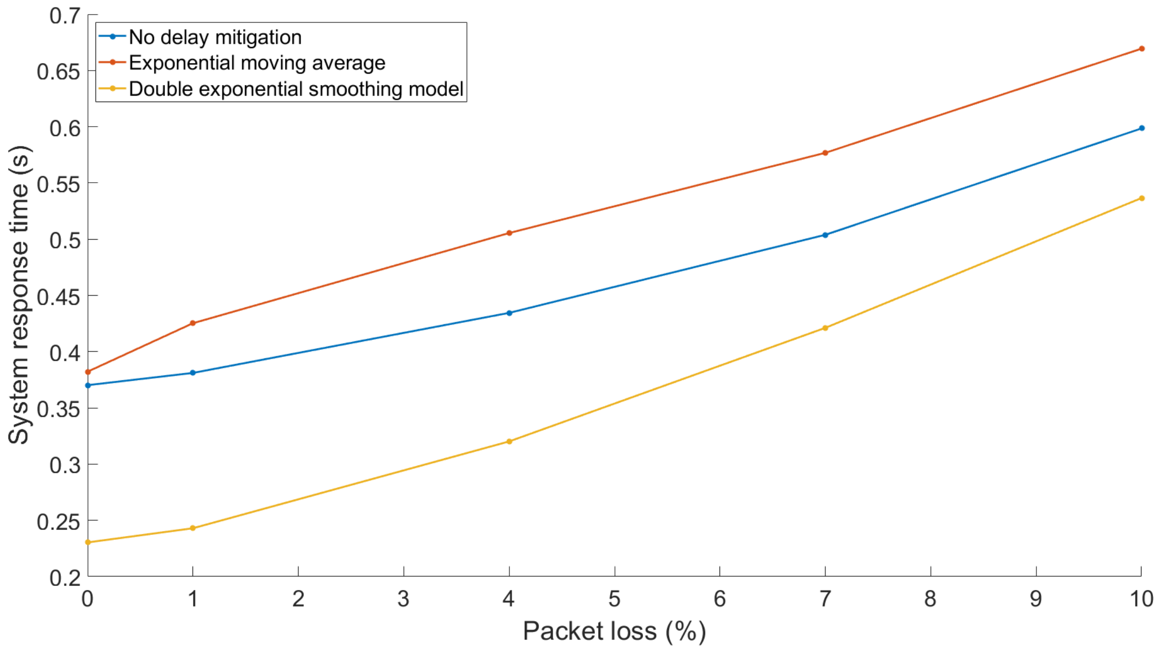


Figure 6.9: Average system response time line graph - Impact of network packet loss

6.4 Validation

The simulation model was adapted to simulate the network conditions of the physical system. When comparing the result obtained from the simulation model and the physical system, the effects of network latency show the same trend. The same relationship between the different network conditions and the average system response time can be made for both the simulation model and the physical system. The data gathered from both systems have a maximum average delta of 2% or 0.011 ms. The difference can be attributed to different communication overheads and the non-deterministic nature of network communication. The pseudo-random rightly skewed distribution of the simulated network delays and jitter can not be fully representative of actual networking delays and jitter. Actual networking delays and jitter is time dependant and challenging to accurately model.

The result obtained by measuring the performance impact of the network location of the remote controller on the performance of the IIoT system yielded similar results, within 2% or 0.006 seconds. The simulation model could also provide a theoretical average system response time of 0.1 seconds if the system had zero network and communication delays. This is, however, only a theoretical response time and not possible for the experimental system due to network and communication delays. The different delay mitigation tested show similar results, with EMA not performing as expected and producing a decreasing in system performance. At the same time, DESM proved effective in delay mitigation and improved system performance when the remote controller is connected via a WAN connection.

The digital and physical systems are subjected to different and increasing network delays. The response of the systems both shows a positive linear or slightly quadratic relationship between network latency and the average system response time for both systems. An increase in network latency increases in the average system response time, as expected.

Both systems also responded similarly to delay mitigation implementations. When EMA is implemented on both system, the performance is decreased, and the system response times are increased for network latency test points. DESM, on the other hand, had a positive effect on the performance of the system by decreasing the average system response times for the various network latency test points. The delta between the simulation model and the physical system with and without delay mitigations implemented for different network latencies is shown in table 6.6. An average of between 1% and 2% difference is observed between the two systems, with and without delay mitigations implemented.

Table 6.6: Delta between experimental system and simulation model with different delay mitigations

Delay (ms)	No-DM (ms)	EMA (ms)	DESM (ms)
0	0.007	0.008	0.010
50	0.007	0.011	-0.002
100	0.014	0.018	0.02
150	0.009	0.004	-0.001
200	0.019	0.021	0.046
300	0.020	0.017	-0.0425
400	0.030	0.065	0.041
500	0.012	0.053	0.019
Average (ms)	0.015	0.024	0.011

The simulation model and the physical system are then subjected to both network latencies and jitter. The response of the simulation model shows the same trend and relationship as the response of the physical system with the same network conditions applied, a nearly linear increase in system response time is seen when latencies are increased. A more predictable increase in system response times is observed when jitter values are increased compared to the physical system. The pseudo-random distribution of the simulated jitter compared to the unpredictable physical network jitter results in a more predictable effect of jitter on the simulation model.

Comparing results for the effects of network latency and jitter between the experimental physical system and the digital simulation model yields a small difference for each test point between the two systems. The differences expressed as a percentage for each network delay and jitter test point is shown in table 6.7. An average of 1% (0.01 seconds) difference is observed between the two systems, with an average deviation of 1% (0.012 Seconds).

The relationship between jitter and average system response time obtained from examining the results from both the simulation model and the physical system show the pattern. A large increase in system response is observed when jitter is introduced to the system, after that, as jitter increase, so does the system response time linearly. As before when delay mitigations are implemented on the systems, the same responses are observed. The EMA as delay mitigation negatively impacted the system response times and decreased the system's performance. Implementing DESM as delay mitigation on both systems improved system performance by decreasing the average system response times at each network condition test interval. The delay mitigations had similar effects of both the simulation model and the physical system.

Table 6.7: Delta between experimental system and simulation model

Delay (ms)\Jitter (ms)	0	10	30	50
0	-2%	N/A	N/A	N/A
50	-2%	-2%	0%	0%
100	-3%	0%	-1%	-1%
150	-1%	-5%	0%	0%
200	-2%	-1%	2%	-2%
300	-2%	-1%	-1%	5%
400	-3%	-3%	0%	-1%
500	-1%	-2%	0%	-1%
Average	2%	2%	0%	0%
Std deviation	1%	1%	1%	2%

6.5 Discussion

Test one indicates that moving the remote controller from a LAN to a WAN network connection degrades the performance of the IIoT system. However, by implementing DESM as delay mitigation on the WAN connected remote controller improves performance, but not quite to the extent when the remote controller is connected via LAN. When EMA is implemented on the WAN connected remote controller as delay mitigation, the performance is decreased over not using any delay mitigations. EMA negatively impact the performance of the IIoT system and is not suited for used in the system. Possible explanations for the decrease in performance is the increased processing time when EMA is implemented, as well as EMA only being useful for smaller disruptions to the system. The simulation model could produce a theoretical system response time equal to the sampling rate, 100ms, of the experimental system. The value is, however, only theoretical as absolutely zero communication, routing and networking delays have to be present in the system and not possible for the experimental system.

Test two shows a linear or slightly quadratic relationship between system response time and network latency. A linear relationship is expected, but increased network overhead with increased latency results in a somewhat quadratic relationship. Network latencies lower than the sampling rate of the system affected the system less than the latencies higher than the sampling rate. Network latencies that are lower than the sampling rate are somewhat absorbed into the sampling rate affecting the system to a lesser degree. Applying DESM to the remote controller improved system performance by decreasing system response times. EMA proved to be ineffective and decreased system performance.

Test three indicates a considerable increase in system response time when jitter is introduced. A linear relationship between network jitter and system performance is observed. A linear relationship is seen, because of the right-skewed, shifted gamma distribution of jitter values and the use of average system response times for results. As before applying DESM improved system performance at each of the tested jitter intervals, while EMA failed and increased system performance.

Test four, conducted on the experimental physical system, shows a quadratic relationship between system response time and network packet loss. An increase in network overhead occurs when packet loss is introduced, as packets that are lost are retransmitted, increasing network traffic. Once again DESM as delay mitigation improved system performance, while EMA decreased system performance.

Comparing results for the effects of network latency and jitter between the physical experimental system and the digital simulation model yields a small difference for each test point between the two systems. The differences expressed as a percentage for each network delay and jitter test point is shown in table 6.6 and 6.7. An average of 1% (0.01 seconds) difference is observed between the two systems, with an average deviation of 1% (0.012 Seconds). Similar performance impacts were observed between the simulation model and the experimental system for different network delay and jitter test points. An average difference of 1% is observed between the two systems. The difference can be attributed to the differences in network overheads, and the pseudo-random nature of jitter values in the simulation model compared to the non-deterministic values observed in the experimental system. The same trends and relationships are observed in both systems between the average system response times and the various network conditions tested.

EMA proved ineffective when implemented on the remote controller as delay mitigation for both the physical system and the simulation model of the experimental system. This is possibly due to the increased processing time required for its implementation, as well as EMA, being more suited for small system disturbances and not the excessive disturbances caused by the different networking conditions. EMA as delay mitigation negatively impacted the performance of the IIoT system for each test point. The implementation of EMA is therefore not suitable for the specific use in the experimental system. DESM, on the other hand, proves effective as a delay mitigation technique. DESM improved system performance over not having any delay mitigations implemented by positively impacting the average system response times, by consistently lowering the system response time for each of the test points. DESM boast better prediction ability, making it more suited as a delay mitigation technique.

6.6 Conclusion

The impact of varying network conditions on the performance of the IIoT system is tested, presented and analysed. The network location of the remote controller, latency, jitter and packet loss are tested to determine the performance impact thereof on the experimental system. Simulated latency and jitter are tested for performance impact on the system's digital simulation model.

By moving the network node of the remote controller from a LAN to a WAN connection reduced the performance of the IIoT system. The impact of latency on the performance of the system is a slightly quadratic, and mostly linear increase in system response time. When jitter is introduced to a system, a large increase in system response time is observed, after that, as jitter is increased so is the average system response in a linear fashion. Packet loss impacts the system response times in a quadratic relationship, as network packet loss is increased in the system, so does the system response time in a quadratic fashion. The experimental system and the simulation model yielded similar results, within 1%, for the performance impact on network latency and jitter on the system. The same trends are observed between the simulation model and the physical system for different networking conditions. The same relationships between the various tested networking conditions and the average system response times could be made for both systems.

Different delay mitigation structures, such as exponential moving averages and double exponential smoothing model, were tested for the different network conditions. The exponential moving average implemented on the system impacted the average system response time for each of the test point negatively and yielded lower system performance values than the system with no delay mitigations implemented. EMA was not suited for the experimental system, as it is more suited for small system disruptions. In contrast, the system with double exponential smoothing model implemented for delay mitigation allowed for increased system performance for each of the tested network conditions over the system with no delay mitigations. DESM positively impacted the system response times at each test point and produced lower average response times over the system with no delay mitigations subjected tot the same networking conditions. DESM was well suited for the specific application in the experimental IIoT system.

Chapter 7

Conclusion

7.1 Introduction

This chapter provides a conclusion of some of the essential aspects and findings for this study.

The effects of different network conditions on the performance of an emulated IIoT system, with and without delay mitigations are concluded. Reflections on the different objectives for this study, as stated in chapter one, is made. The chapter gives the conclusion for the dissertation as well as a section for recommendations useful for future work. An accepted conference contribution based on this study is outlined, followed by a final closure for this study is then presented.

7.2 Reflecting on the objectives

The performance impact of different network conditions between the remote controller and the local controllers on an emulated system with a bilateral control loop architecture controlling a system of interconnected electric motors was investigated. The system uniquely made use of both industrial hardware and communication equipment (Siemens$^{©}$ PLCs, VSDs, electric motors, scalance modules and IoT gateway), as well as consumer networking equipment.

The *first* objective entails the experimental design, where an experiment was designed to determine the effects of different network conditions on the performance of an IIoT control and monitoring system. Different system control loop architecture was examined, and a proposed system architecture for the experiment was designed. For the proposes system, a bilateral system architecture was developed with both local controllers and a remote controller. The local controller is responsible for the regular operation of a sub-system, while the remote controller is used for monitoring and changing parameters of the interconnected system of locally controlled sub-systems. The use of both local controllers and a network-attached remote controller allows for the manipulation of network conditions between the remote and local controllers. The Architecture was developed based on the industrial equipment in a mechatronics laboratory made available for this study. The experimental design of the system was created based on the proposed bilateral system. The experimental design included the use of industrial hardware and communication equipment, as well as consumer networking equipment. The experimental design allowed for industrial plant emulation of a system containing electrical motors. A network diagram was created that allowed for the network conditions to be changed between the remote controller and the local controller through a network emulation program, WANem. Different delay mitigations, obtained from literature, are discussed for implementation to help curb the effect of network conditions on the IIoT system. Four tests were designed to determine the system performance impact of the network location of the remote controller, network latency, jitter and packet loss.

The implementation of the experimental design allowed for results on the performance impact of different network conditions on the emulated IIoT control and monitoring system. The delay mitigations implemented that were obtained from literature yielded contrasting results. By implementing an exponential moving average for delay mitigation decreased system performance instead of increasing performance over a system with no delay mitigation implemented for different networking conditions. The decrease in performance is most probably due to the algorithm being suitable for small disruptions only. On the other hand implementing double exponential smoothing model for delay mitigation improved system performance, as expected. The algorithm allowed for system response prediction improving the performance of the system when implemented over the system with no delay mitigations.

The *second* objective is to create a digital simulation model of the experimental system. The digital simulation model of the experimental system was constructed to aid in testing control schemes and to aid in the evaluation of the performance impact of different network conditions on an IIoT system. The simulation model digitally represented the functional, behavioural and communication sub-models of the experimental system. The simulation model digitally described the response of the experimental system. The simulation is subjected to the same control scheme as the experimental system with the response of the motors at discrete time intervals logged and then output for evaluation. Due to the irregular and time-dependent nature of the internet and consumer network communication, the accurate representation of network communication proved difficult. The network communication behaviour in the simulation was modelled in a statistical and mathematical sense. The simulations model is verified in terms of having an accurate and representing response of the experimental physical system and a representing network communication response. The reaction of the simulation model is also verified with the control scheme, designed in the experimental design, implemented. The response of the simulation model was as expected, and as planned, the simulation model created is therefore verified for use in the experimental system.

The *third* objective was constructing the experimental system in the mechatronics laboratory using the industrial equipment and the consumer network components made available for this study. The Siemens© industrial hardware and consumer networking equipment were built and connected based on the experimental design and the network architecture. The connected components were programmed in their respective programming environment to be able to perform their task, as stated in the experimental design.

The original sampling period for the experimental system was meant to be 50 ms, the fastest sampling period capable by the IoT gateway module used. When two or fewer PLCs are connected to the IoT gateway module, the sampling period of 50 ms performed without errors. However, when the third PLC is attached to the IoT gateway module with a sampling period of 50 ms, errors occurred within the system. The sampling period was then increased to 100 ms to prevent system errors. The system with three PLCs and the sampling period of 100 ms performed as required. This raises the question of scalability for such a scenario, as devices are added to the system, so does the amount of data and communication overheads.

The experimental system was tested for its operation and its response to the control scheme of the system. The response of the system when the control scheme was implemented was as expected, and the functionality of being able to modify network communication conditions allowed for the tests to be executed.

The *fourth* objective entails the testing, evaluation and analysis of the performance impact of the different network conditions on the experimental IIoT control and monitoring system. The following results were obtained from examining the test to determine the performance impact of different networking conditions on the IIoT system, as stated in the research approach in section 3.6.

In the first test, a 102% increase in average system response time is observed when the remote controller is moved from a LAN connection to a WAN connection with best-case network conditions. When the remote controller is connected to the system via a LAN connection it is seen as a local cloud or fog node, and when connected via a WAN connection it is seen as a remote cloud node.

The results from tests two, three and four indicate that for a system with no delay mitigations implemented that the relationship between the average system response time and the tested network condition appears to be slightly quadratic for latency, linear for jitter, and quadratic for packet loss. As the various network conditions (latency, jitter, packet loss) increase, so does the system response time, thus decreasing the system performance.

The system with delay mitigations implemented had contrasting results. The first delay mitigation method, exponential moving average, caused a decrease in performance as the algorithm is best suited for small compensations and the implementation thereof increased processing time. However, the double exponential smoothing model consistently increased the system's performance when compared to the results of the system with no delay mitigation at the same networking conditions. The double exponential smoothing model algorithm's better system performance is attributed to its better forecasting ability.

An increase of 21% in system response time is still observed when the remote controller is moved from a local fog node to a remote cloud node using the double exponential smoothing model for delay mitigation. The best possible performance for the emulated system is therefore observed when the remote controller is implemented on a fog node on a local network connection.

Similar results, within an average of 1%, are obtained from the digital simulation model of the IIoT system when latency and jitter are introduced to the system. Differences can be explained by the time-dependent and random nature of network communication. The results from the simulation model for different network latencies and jitter show a linear relationship between the average system response time with latency and jitter, respectively.

Therefore it can be concluded that using a local cloud or fog node is a better solution for the emulated system than cloud nodes. Remote cloud nodes are best kept for less critical soft real-time tasks that can handle delays with more flexibility. For a remote cloud implementation, the remote controller with double exponential smoothing model as delay mitigation and favourable network conditions proved quite capable for the emulated system.

7.3 Future work

Many possibilities are left open in smart manufacturing and the use of IoT and cloud services in industrial processes, but further investigations in the reliability, performance, security and scalability are required before placing full confidence in such a system. These include testing the effects of network conditions on processes with different timing requirements and different processing complexities. Testing should also be done where system and server loads vary. A system of different but interconnected sub-processes and sub-system with varying loads should be tested. The use of delay compensators as described in [5] could be investigated for implementation on an IIoT system with different network conditions.

Testing should be done on the scalability of such an IIoT system, as well as how implementing stricter security protocols on the system will impact system performance. Adding more devices and protocols to a network increases overhead. The increased overheads and complexities in communication routing are compounded with worsened networking conditions, possibly resulting in a more considerable impact on system performance. The scalability and hardware limitations of larger and more complex IIoT control and monitoring systems should be investigated, as hardware limitations already impacted the sampling period of the experimental system used in this study.

7.4 Conference contribution

A conference contribution was made for SUAPEC/RobMech/PRASA from the findings of this study outlined in this dissertation. The conference is held from 27 to 29 January 2021 in Potchefstroom, South Africa. The accepted article for the conference is `R. Bock, K.R. Uren, and G. van Schoor, ‘‘Performance impact of network conditions on an IIoT system." SUAPEC/RobMech/PRASA conference,` 2021 and is shown in appendix E.

7.5 Closure

The objective of determining the effects of different network conditions on specifically an IIoT with and without delay mitigations was achieved. A relationship between the different network conditions and the average system response time and thus, system performance could be established. The implementation of delay mitigations in the system was tested for different network conditions when implementing a double exponential smoothing model as delay mitigation. The system performance improved for each of the tested network conditions over the system with no delay mitigations implemented. DESM proved capable as delay mitigation for small network degrading conditions. More testing is required, especially with regards to scalability and security of such an IIoT system.

Bibliography

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[2] DFKI, "Deutsches Forschungszentrum für Künstliche Intelligenz (German Research Center for Artificial Intelligence)," 2011. [Online]. Available: https://www.dfki.de/web/

[3] J. Lee, B. Bagheri, and H. A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.mfglet.2014.12.001

[4] C. G. Lee and S. C. Park, "Survey on the virtual commissioning of manufacturing systems," *Journal of Computational Design and Engineering*, vol. 1, no. 3, pp. 213–222, 2014. [Online]. Available: http://dx.doi.org/10.7315/JCDE.2014.021

[5] T. Hegazy and M. Hefeeda, "Industrial Automation as a Cloud Service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2750–2763, 2015.

[6] S. Mercan and A. I. Zreikat, "Statistical analysis of packet delay time and variation on the internet," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019*, pp. 695–700, 2019.

[7] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, p. 27–32, Oct. 2014. [Online]. Available: https://doi.org/10.1145/2677046.2677052

[8] L. Bassi, "Industry 4.0: Hope, hype or revolution?" in *RTSI 2017 - IEEE 3rd International Forum on Research and Technologies for Society and Industry, Conference Proceedings*, 2017, pp. 1–6.

[9] A. Didic and P. Nikolaidis, "Real-time control in industrial IoT," 2015. [Online]. Available: http://mdh.diva-portal.org/smash/get/diva2:821335/FULLTEXT01.pdf

[10] Y. Tipsuwan and M. Y. Chow, "Control methodologies in networked control systems," *Control Engineering Practice*, vol. 11, no. 10, pp. 1099–1111, 2003.

[11] A. Faul, N. Jazdi, and M. Weyrich, "Approach to interconnect existing industrial automation systems with the Industrial Internet," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2016-Novem, pp. 1–4, 2016.

[12] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," *Proceedings of the Annual Hawaii International Conference on System Sciences*, vol. 2016-March, pp. 3928–3937, 2016.

[13] T. Pereira, L. Barreto, and A. Amaral, "Network and information security challenges within Industry 4.0 paradigm," *Procedia Manufacturing*, vol. 13, pp. 1253–1260, 2017. [Online]. Available: https://doi.org/10.1016/j.promfg.2017.09.047

[14] Ashton K., "That 'Internet of Things' Thing," *RFID Journal*, p. 4986, 2009.

[15] R. van Kranenburg and A. Bassi, "IoT Challenges," *Communications in Mobile Computing*, vol. 1, no. 1, pp. 1–5, 2012.

[16] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," in *Proceedings of 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2014*. IEEE, 2014, pp. 1–4.

[17] P. Derler, E. A. Lee, M. Törngren, and S. Tripakis, "Cyber-physical system design contracts," in *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2013, pp. 109–118.

[18] E. A. Lee, "Cyber Physical Systems: Design Challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369.

[19] C. J. du Plessis, "A framework for implementing Industrie 4 .0," no. March, 2016.

[20] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.bushor.2015.03.008

[21] H. Shuang Yang, *Internet-based Control Systems.* Springer, 2011.

[22] B. A. FOROUZAN, *Data Communications AND Networking*, 5th ed. McGraw-Hill, 2013.

[23] "Open Glossary of Edge Computing," 2019. [Online]. Available: https://www.lfedge.org/projects/openglossary/

[24] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," 2011. [Online]. Available: http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

[25] D. Linthicum, "Edge computing vs. fog computing: Definitions and enterprise uses." [Online]. Available: https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html

[26] Q. Qi and F. Tao, "Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison," *IEEE Access*, vol. 6, pp. 3585–3593, 2018.

[27] A. El Saddik, "Digital Twins: The Convergence of Multimedia Technologies," *IEEE Multimedia*, vol. 25, no. 2, pp. 87–92, 2018.

[28] M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann, "Experimentable Digital Twins—Streamlining Simulation-Based Systems Engineering for Industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1722–1731, 2018.

[29] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, *Vision and challenges for realizing the internet of things*, Brussels,, 2010, vol. 1, no. March 2010.

[30] T. R. Alves, M. Buratto, F. M. De Souza, and T. V. Rodrigues, "OpenPLC: An open source alternative to automation," *Proceedings of the 4th IEEE Global Humanitarian Technology Conference, GHTC 2014*, pp. 585–589, 2014.

[31] "OpenPLC," 2020. [Online]. Available: https://www.openplcproject.com/reference

[32] E. Daniel, C. White, and K. Teague, "An interarrival delay jitter model using multistructure network delay characteristics for packet networks," pp. 1738–1742, 2004.

[33] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2012.02.019

[34] G. Prytz, “A performance analysis of EtherCAT and PROFINET IRT Gunnar Prytz Abstract,” *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, no. 1396, pp. 408–415, 2008.

[35] V. Paxson, “End-to-end Internet packet dynamics,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 277–292, 1999.

Appendices

Appendix A

PLC frame design

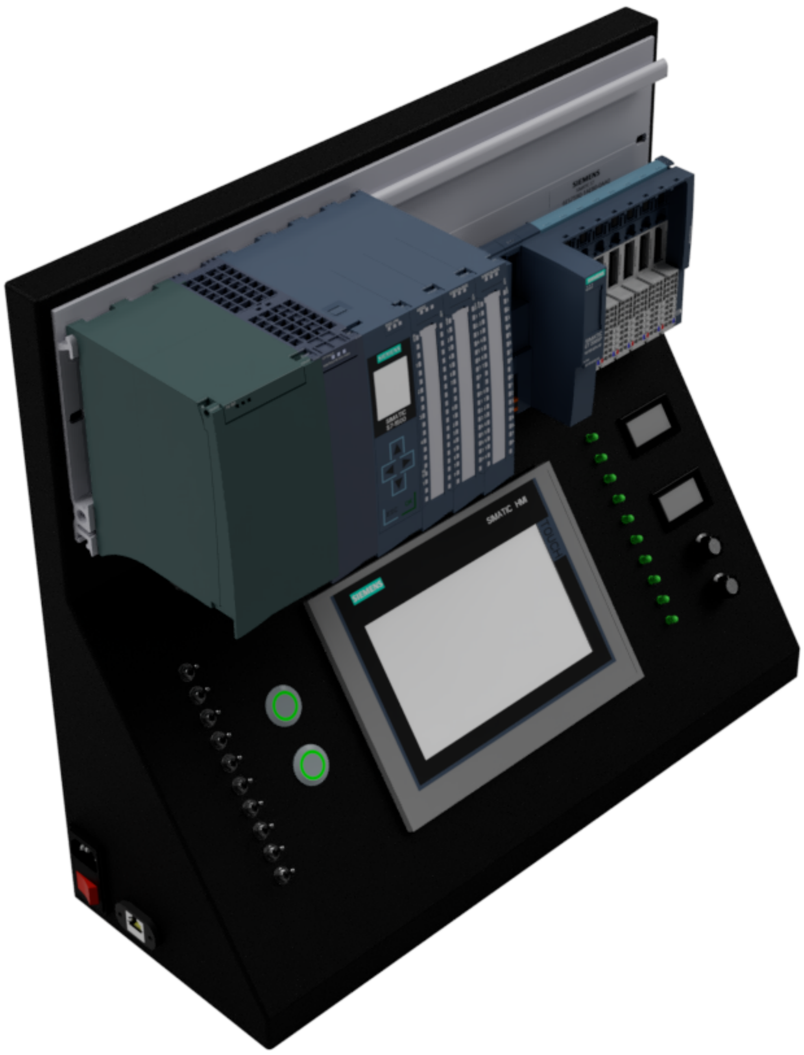


Figure A.1: S7-1500 PLC and Siemens© TP700 HMI panel frame design render

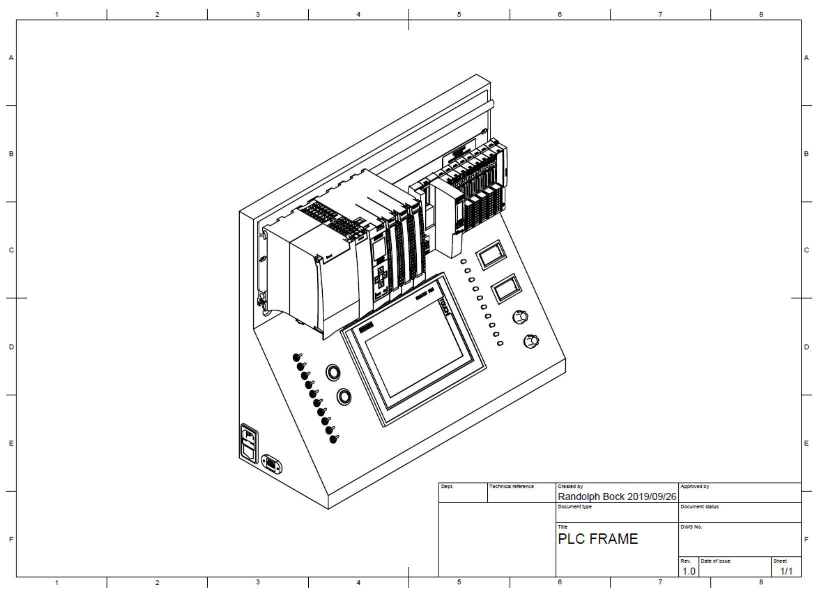


Figure A.2: S7-1500 PLC and Siemens© TP700 HMI panel frame design drawing

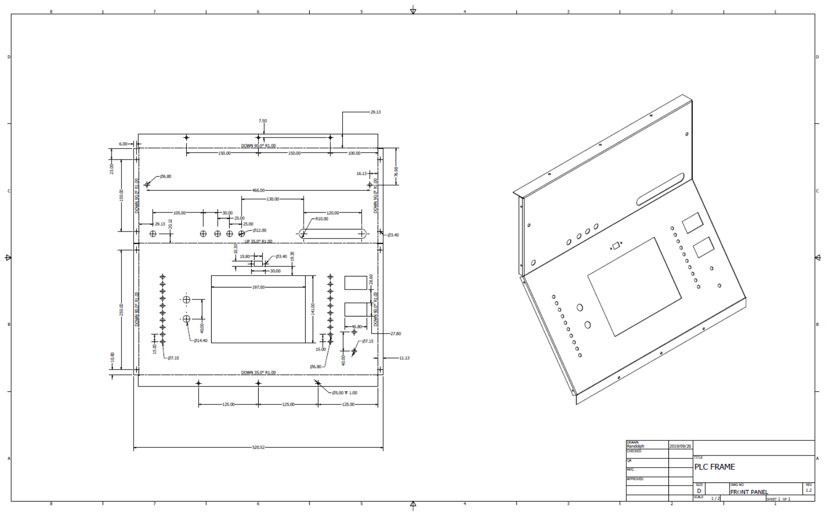


Figure A.3: S7-1500 PLC and Siemens© TP700 HMI panel frame front plate design drawing

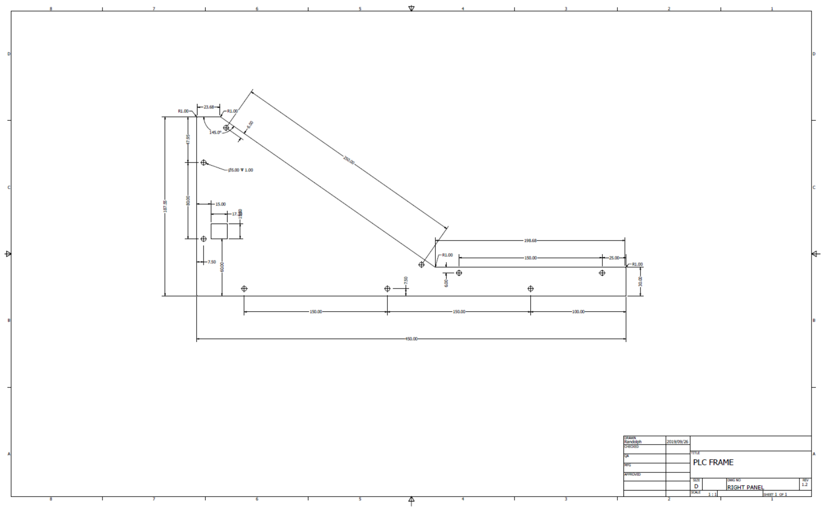


Figure A.4: S7-1500 PLC and Siemens© TP700 HMI panel frame side plate design drawing

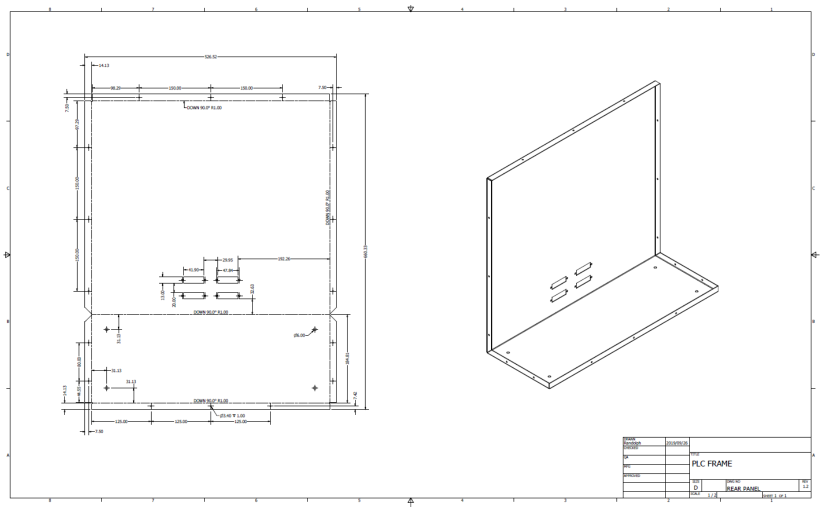


Figure A.5: S7-1500 PLC and Siemens© TP700 HMI panel frame back plate design drawing

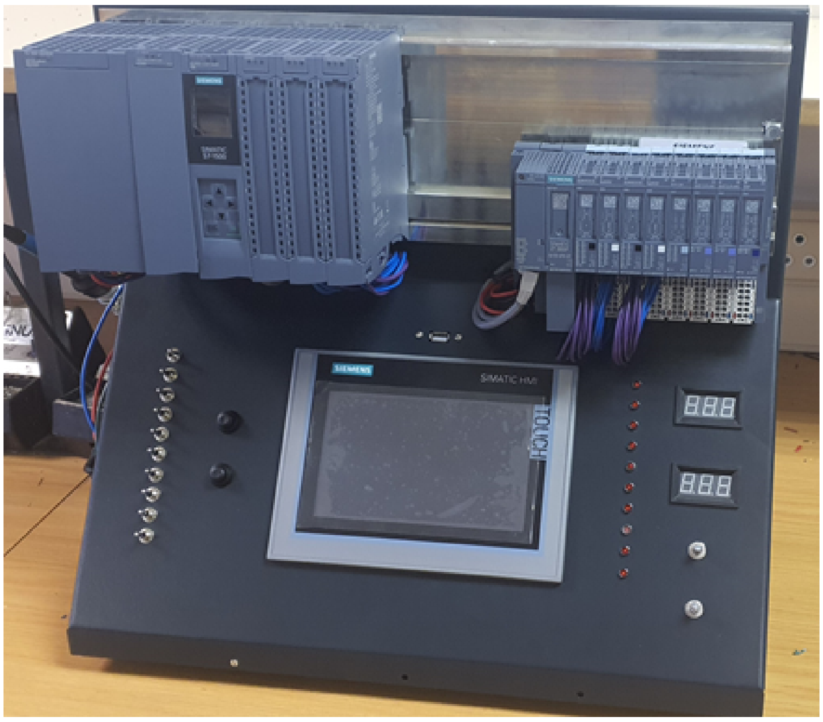


Figure A.6: S7-1500 PLC and Siemens© TP700 HMI panel frame setup constructed

Appendix B

Motor frame design

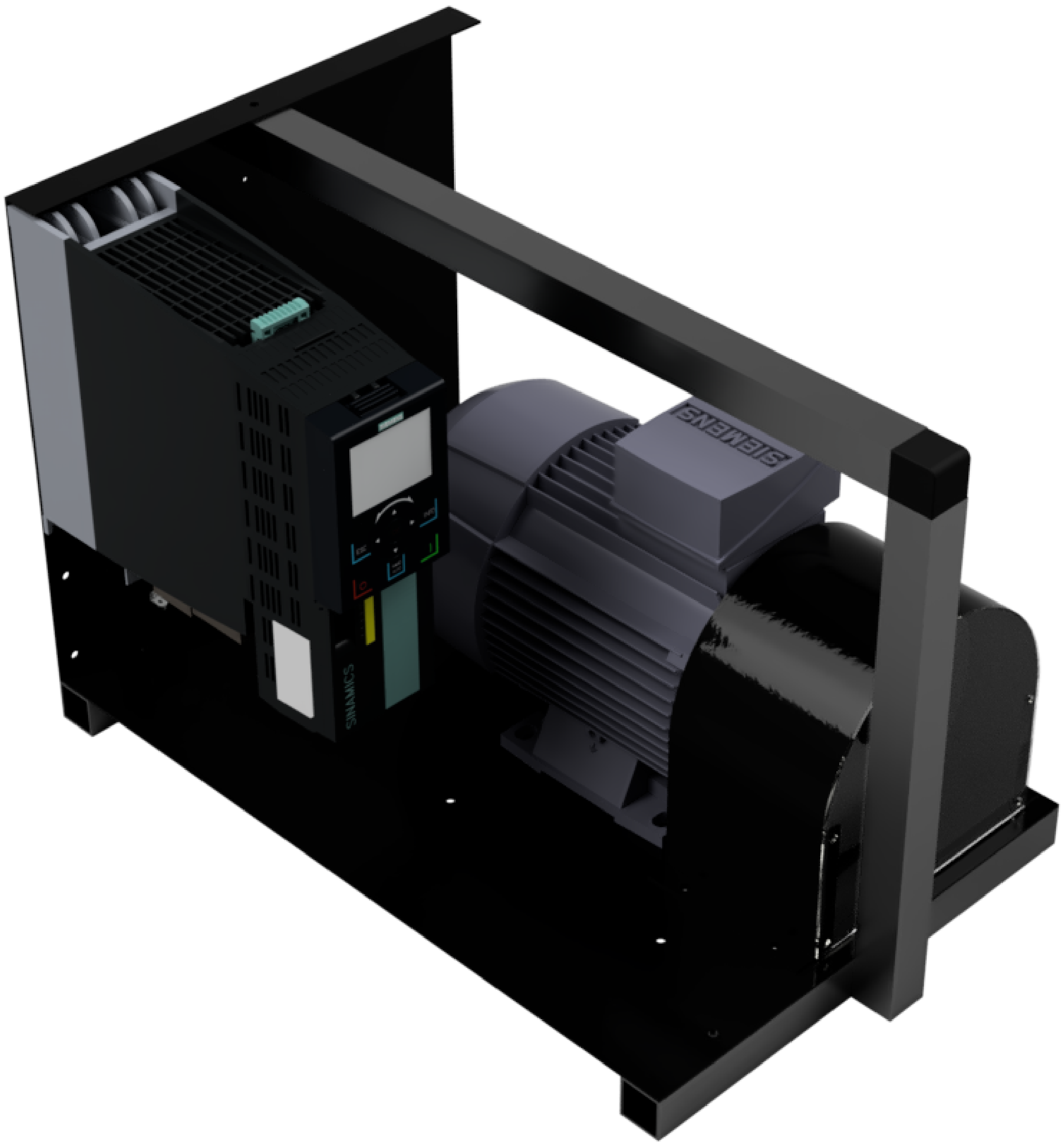


Figure B.1: Siemens© induction motor, VSD and Power module - frame design render

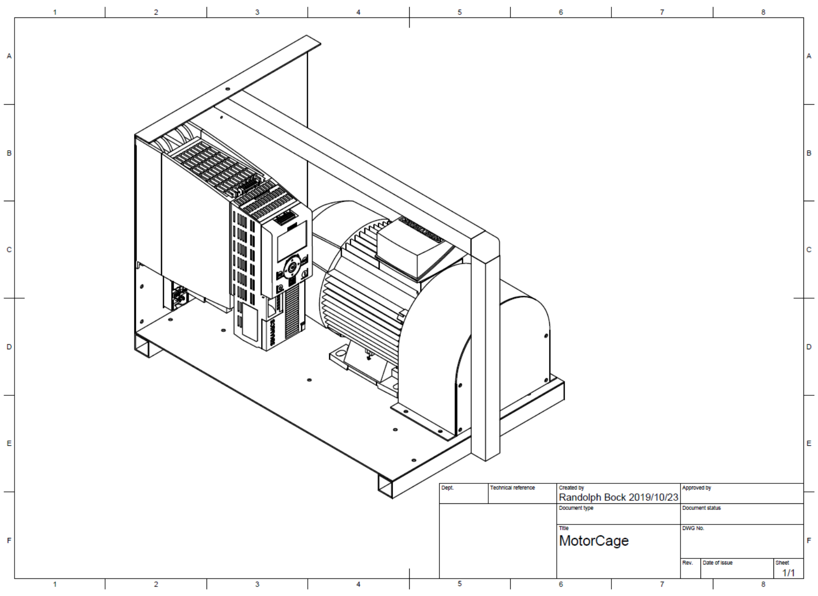


Figure B.2: Siemens© induction motor, VSD and Power module - frame design drawing

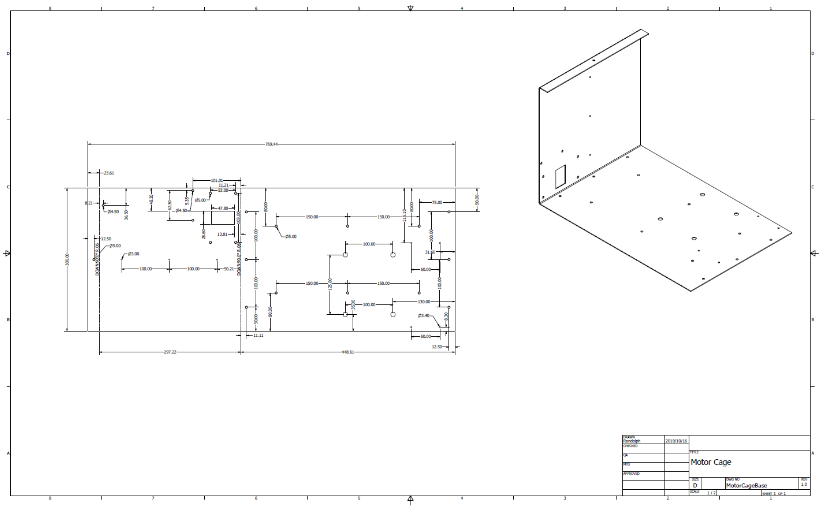


Figure B.3: Siemens© induction motor, VSD and Power module - frame base design drawing

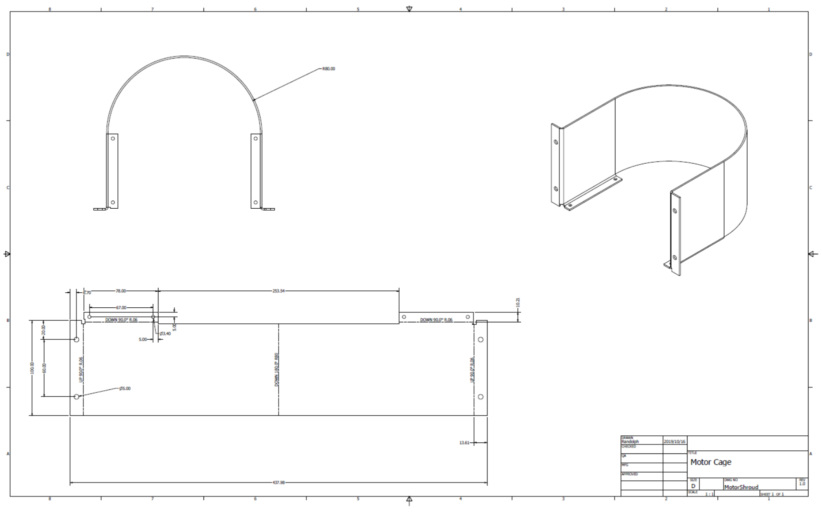


Figure B.4: Siemens© induction motor, VSD and Power module - frame shroud design drawing

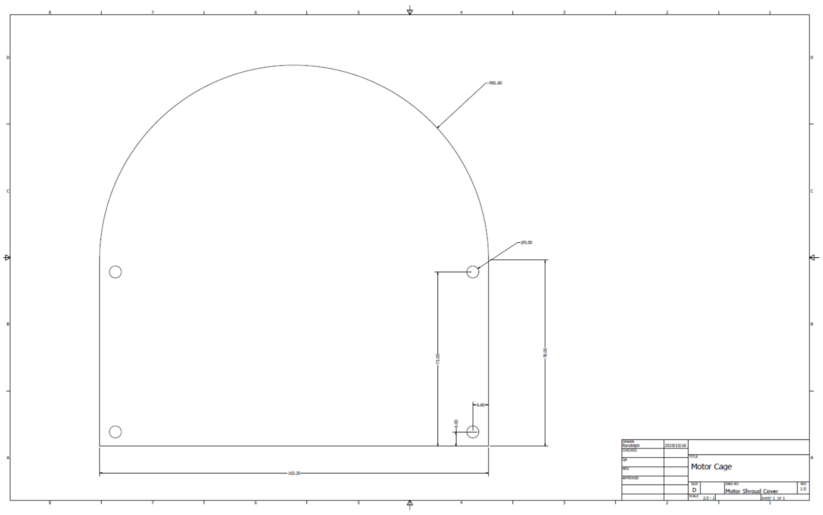


Figure B.5: Siemens© induction motor, VSD and Power module - frame shroud cover design drawing

Appendix C

Remote controller script

```python
#! python3
#Import all required library and modules
import socket, re, threading, subprocess, time

re_PLC = re.compile(r'''PLC(\d):(-?\d+\.?\d*)''')#Regular expresion to extract PLC numbers
     and their measured motor velocities
TCP_IP = '192.168.111.20'#Industrial network address
#TCP_IP = '192.168.0.4'#Consumer network address
TCP_PORT = 10000#TCP/ip connection port
BUFFER_SIZE = 1024#TCP/IP Message buffer size
server_address = (TCP_IP,TCP_PORT)

#Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Bind the socket to the port
print('Starting up on %s port %s' % server_address)
sock.connect((TCP_IP, TCP_PORT))

def sendSetpoints(SP1,SP2,SP3):#Format and send set points to the IoT gateway
    #format and encode message
    try:
        msg = str(SP1)+','+str(SP2)+','+str(SP3)+'$'
        msg = str.encode(msg)
    except:
        print('Message string encoding error!')
    #Send mesage over TCP to IoT_Gateway
    try:
        #print('Sending %s' %msg)
        sock.send(msg)
    except:
        print('TCP transmision error!')

#Receive and extract TCP message data
def receiveActVelocity():
    #receive TCP Data
    try:
        TCP_data = sock.recv(BUFFER_SIZE)
        TCP_data = TCP_data.decode('utf-8')
        #print(TCP_data)
        ValTuple = re_PLC.findall(TCP_data)#Extract PLC number and PLC's velocity into a
    Tuple
    except:
        print('Error while receiving and extracting TCP data')
    return ValTuple

```

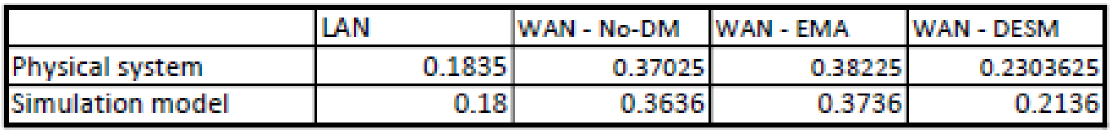
```
def step1():#First step for motor1 @ 0.4Seconds
    global SetPoint1
    SetPoint1 = 1500
    sendSetpoints(SetPoint1,0,0)

def step2():#Second step for motor1  @ 3.0Seconds
    global SetPoint1
    SetPoint1 = -1500

def end():#Copy measurement log files from IoT gateway to Remote server for result analysis @ 7.0Seconds
    print("Test done!!!!!!!!!!!!!")
    #subprocess.call([r'C:\Users\Randolph\Desktop\CopyMeasurementsFromNR.bat'])#Copy velocity log files from IoT gateway to remote server computer (Industrial network address)
    #subprocess.call([r'C:\Users\Randolph\Desktop\CopyMeasurementsFromNR_ConNet.bat'])#Copy velocity log files from IoT gateway to remote server computer (Consumer network address)

#Create interrupts required for step1, step2 and end functions
t1 = threading.Timer(0.4, step1)
t2 = threading.Timer(7.0, step2)
t3 = threading.Timer(12.0, end)

try:
    global SetPoint1, SetPoint2, SetPoint3#Setpoint variables
    global ActVelocity1, ActVelocity2, ActVelocity3#Measurement value variables
    start = False# test start variable
    SetPoint1, SetPoint2, SetPoint3, ActVelocity1, ActVelocity2, ActVelocity3 = 0,0,0,0,0,0#Initialize variables
    sendSetpoints(SetPoint1,SetPoint2,SetPoint3)#Send 0 r/min setpoint to reset any setpoint in the motors
    time.sleep(0.2)
    print('Test starting!!!!!!')
    t1.start()
    t2.start()
    t3.start()
    while True:
        #receive and interpret motor measurements received
        ActVelocityTuple = receiveActVelocity()
        for i in ActVelocityTuple:
            PLC_num = i[0]
            try:
                SpeedVal = float(i[1])
            except:
                print('ActVelocity type error!')
            if PLC_num == '1':
                ActVelocity1 = SpeedVal
            elif PLC_num == '2':
                ActVelocity2 = SpeedVal
            elif PLC_num == '3':
                ActVelocity3 = SpeedVal
            #Calculate setpoints according to mission scheme
            SetPoint2 = -ActVelocity1
            SetPoint3=(-2/3)*ActVelocity2
            #print("Received value from PLC %s: " % PLC_num, SpeedVal)
        sendSetpoints(SetPoint1,SetPoint2,SetPoint3)#Send set points
        #time.sleep(0.01)

finally:
    sock.close()#Close connection socket
    print('Closing socket')
```

Listing C.1: Remote controller phyton script

Appendix D

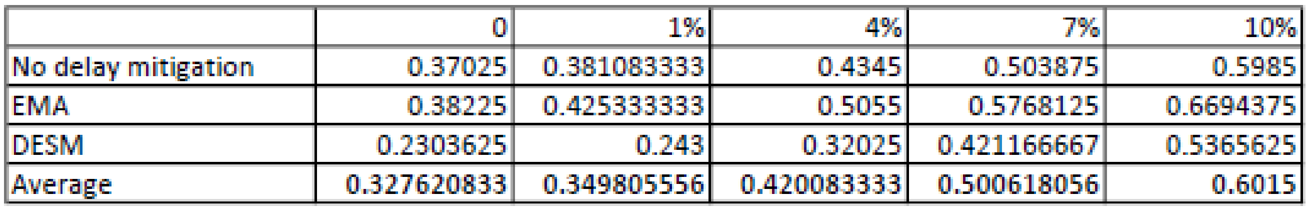
Tests raw results

Performance impact of network location



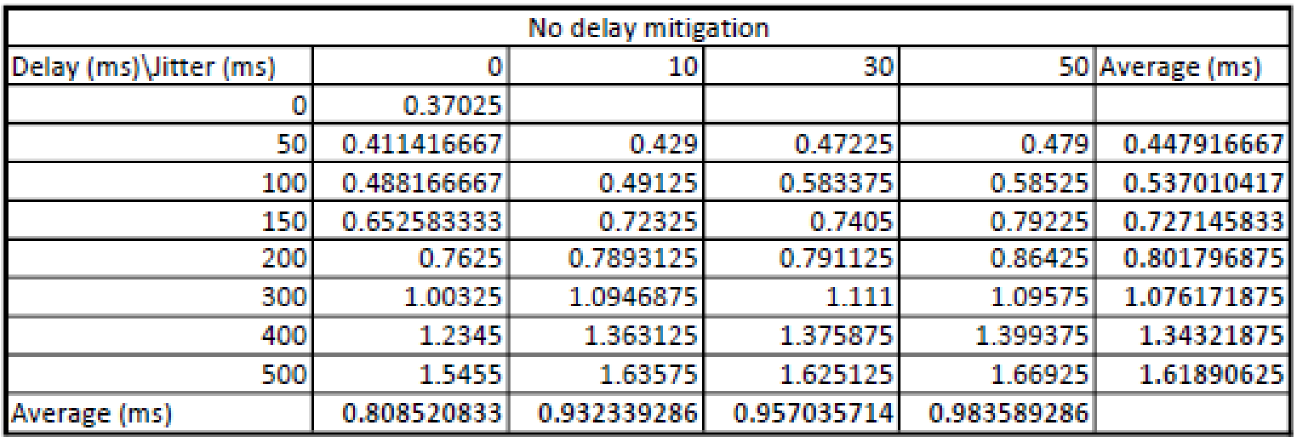
	LAN	WAN - No-DM	WAN - EMA	WAN - DESM
Physical system	0.1835	0.37025	0.38225	0.2303625
Simulation model	0.18	0.3636	0.3736	0.2136

Performance impact of network packet loss on physical system

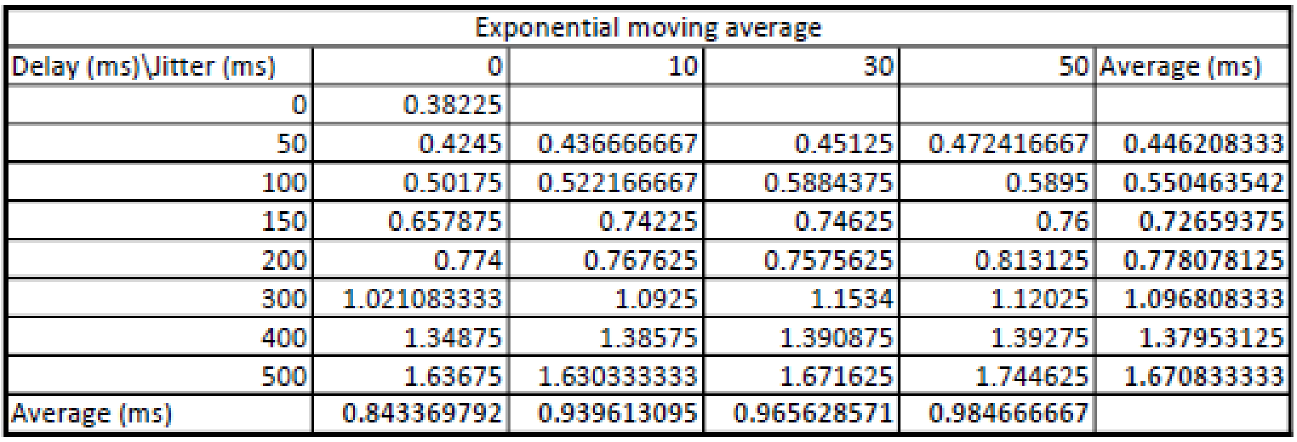


	0	1%	4%	7%	10%
No delay mitigation	0.37025	0.381083333	0.4345	0.503875	0.5985
EMA	0.38225	0.425333333	0.5055	0.5768125	0.6694375
DESM	0.2303625	0.243	0.32025	0.421166667	0.5365625
Average	0.327620833	0.349805556	0.420083333	0.500618056	0.6015

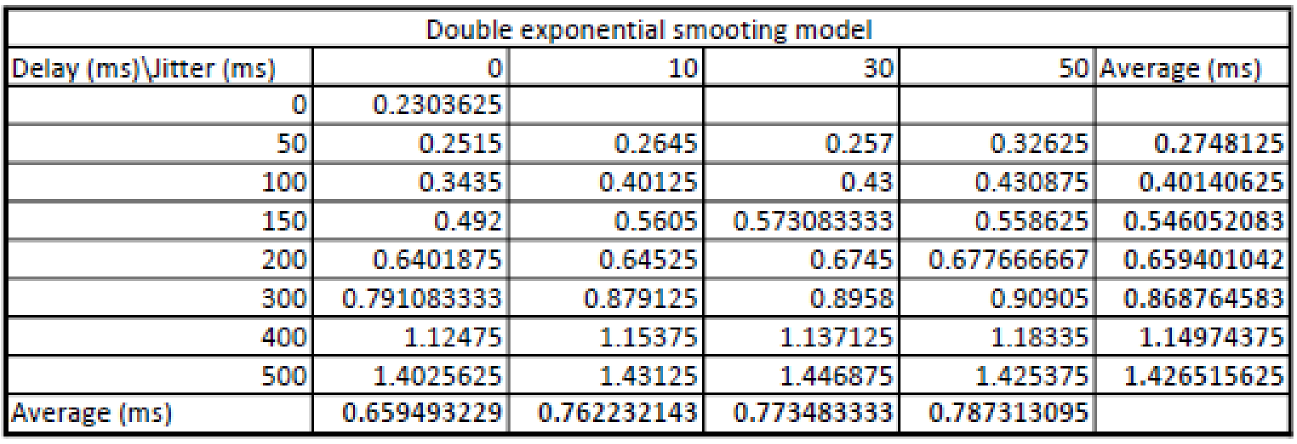
Performance impact of network latency and jitter on physical experimental system



No delay mitigation					
Delay (ms)\Jitter (ms)	0	10	30	50	Average (ms)
0	0.37025				
50	0.411416667	0.429	0.47225	0.479	0.447916667
100	0.488166667	0.49125	0.583375	0.58525	0.537010417
150	0.652583333	0.72325	0.7405	0.79225	0.727145833
200	0.7625	0.7893125	0.791125	0.86425	0.801796875
300	1.00325	1.0946875	1.111	1.09575	1.076171875
400	1.2345	1.363125	1.375875	1.399375	1.34321875
500	1.5455	1.63575	1.625125	1.66925	1.61890625
Average (ms)	0.808520833	0.932339286	0.957035714	0.983589286	



Exponential moving average					
Delay (ms)\Jitter (ms)	0	10	30	50	Average (ms)
0	0.38225				
50	0.4245	0.436666667	0.45125	0.472416667	0.446208333
100	0.50175	0.522166667	0.5884375	0.5895	0.550463542
150	0.657875	0.74225	0.74625	0.76	0.72659375
200	0.774	0.767625	0.7575625	0.813125	0.778078125
300	1.021083333	1.0925	1.1534	1.12025	1.096808333
400	1.34875	1.38575	1.390875	1.39275	1.37953125
500	1.63675	1.630333333	1.671625	1.744625	1.670833333
Average (ms)	0.843369792	0.939613095	0.965628571	0.984666667	



Double exponential smooting model					
Delay (ms)\Jitter (ms)	0	10	30	50	Average (ms)
0	0.2303625				
50	0.2515	0.2645	0.257	0.32625	0.2748125
100	0.3435	0.40125	0.43	0.430875	0.40140625
150	0.492	0.5605	0.573083333	0.558625	0.546052083
200	0.6401875	0.64525	0.6745	0.677666667	0.659401042
300	0.791083333	0.879125	0.8958	0.90905	0.868764583
400	1.12475	1.15375	1.137125	1.18335	1.14974375
500	1.4025625	1.43125	1.446875	1.425375	1.426515625
Average (ms)	0.659493229	0.762232143	0.773483333	0.787313095	

Performance impact of network latency and jitter on simulation model

No delay mitigation

Delay (ms)\Jitter (ms)	0	10	30	50	Average (ms)
0	0.3636				
50	0.4036	0.42	0.47	0.479	0.44315
100	0.4735	0.49	0.578	0.582	0.530875
150	0.6435	0.69	0.74	0.79	0.715875
200	0.7435	0.78	0.805	0.85	0.794625
300	0.9836	1.08	1.1	1.15	1.0784
400	1.2036	1.32	1.37	1.39	1.3209
500	1.5336	1.6	1.62	1.66	1.6034
Average (ms)	0.7935625	0.911428571	0.954714286	0.985857143	

Exponential moving average

Delay (ms)\Jitter (ms)	0	10	30	50	Average (ms)
0	0.3736				
50	0.4136	0.445681101	0.471696577	0.490734673	0.455428088
100	0.4835	0.515581101	0.541596577	0.560634673	0.525328088
150	0.6535	0.685581101	0.711596577	0.730634673	0.695328088
200	0.7535	0.785581101	0.811596577	0.830634673	0.795328088
300	1.0036	1.035681101	1.061696577	1.080734673	1.045428088
400	1.2837	1.315781101	1.341796577	1.360834673	1.325528088
500	1.5836	1.615681101	1.641696577	1.660734673	1.625428088
Average (ms)	0.818575	0.914223958	0.940239435	0.95927753	

Double exponential smooting model

Delay (ms)\Jitter (ms)	0	10	30	50	Average (ms)
0	0.2136				
50	0.2536	0.294695565	0.305946756	0.319776518	0.29350471
100	0.3235	0.364595565	0.375846756	0.389676518	0.36340471
150	0.4935	0.534595565	0.545846756	0.559676518	0.53340471
200	0.5935	0.634595565	0.645846756	0.659676518	0.63340471
300	0.8336	0.874695565	0.885946756	0.899776518	0.87350471
400	1.0837	1.124795565	1.136046756	1.149876518	1.12360471
500	1.3836	1.424695565	1.435946756	1.449776518	1.42350471
Average (ms)	0.647325	0.75038128	0.76163247	0.775462232	

Appendix E

Conference contribution

Performance impact of network conditions on an IIoT system.

R Bock
School of Electrical, Electronic Computer Engineering
North-West University (NWU)
Potchefstroom, South Africa
ORCID: 000-0002-4059-4224

KR Uren
School of Electrical, Electronic and Computer Engineering
North-West University (NWU)
Potchefstroom, South Africa
ORCID: 0000-0002-0561-0735

G van Schoor
School of Electrical, Electronic and Computer Engineering
North-West University (NWU)
Potchefstroom, South Africa
ORCID: 0000-0001-5702-1812

***Abstract*—Advances in smart manufacturing and industry 4.0 have drawn the interest of the industry. The rise in popularity of *internet of things* and *cyber-physical systems* in a manufacturing environment meant the broader connection of devices, servers, sensors and actuators within the closed-loop control. The use of existing communication networks such as computer networks and the internet to connect controllers, sensors, and actuators introduces different communication behaviours to the typical closed-loop control. This article examines the impact of different network conditions on the performance of a control process using a remote system controller. Network conditions examined are network node location of the remote controller, latency, packet delay variance (jitter) and packet loss.**

An industrial plant is emulated through a system of electric motors. The emulated system uses both local controllers and a remote controller. A local controller is responsible for direct control and monitoring of a motor including failsafe features for the motor. The remote controller is responsible for control over the whole interconnected system.

The performance impact of different network conditions is measured for a system with and without delay mitigations implemented. As network conditions worsen, the performance of the system degrades in a linear and sometimes quadratic fashion. The relationship between system response time and network latency is slightly quadratic, the relationship between jitter and system response time is linear, and the relationship between packet loss and system response time is quadratic.

***Index Terms*—IIoT, Industrial control systems, latency, jitter, packet loss, industry 4.0.**

I. Introduction

Advancements in new technologies drove the world into its fourth industrial revolution, or as it has come to be known Industry 4.0 (I4.0). The fourth industrial revolution contains a trend of system-wide automation, and the use of cyber-physical systems, where the physical world is linked via communication models to a virtual world. Emerging and maturing technologies such as cloud computing and Internet of Things (IoT) bring advancements in data collection, process monitoring and improvements in the intelligence and decision-making abilities of industrial systems [1]. IoT allows for everyday and simple objects to communicate and interact with each other, therefore increasing the number of connected devices. The increase in devices increases the amount of data transfer and collections. Cloud computing can support an increase in the number of users as well as the increased amount of data associated with the increase in users and devices.

The benefits to industry of implementing an IoT and cloud-based industrial control system will be centralised control and cost optimisation of systems. However, moving parts of the control system are not straightforward since it can affect the stability, robustness and reliability of the system [1].

Problems that exist within Industry 4.0 include: effectively processing and evaluating the large amount of data received from IoT devices; the cyber-security aspects of keeping data safe and private and preventing malicious attacks; and finally the problem explored in this study is the unpredictable network delay introduced in distributed control loops. The communication networks introduce delays. The delays are due to limited bandwidth and overhead in communication nodes and networks. The delays in different systems will be varying randomly. Control systems with varying delays cannot be described as time-invariant. Standard techniques and theory can, therefore, not be used in the design and analysis of distributed control systems [1].

This paper presents the performance impact of different network conditions on an *Industrial Internet of Things* (IIoT) system. The tests are uniquely done for determining the impact of network conditions on an *industrial* system using industrial hardware and communication components. The IIoT system monitors and controls an emulated system containing three electric motors. The performance impact of moving the remote controller from a local area network (LAN) to a wide area network (WAN) is measured. Subsequently, different network conditions are tested by using a network emulator to change the network's conditions and behaviours. The network conditions tested for include: network latency, packet delay variance (jitter) and packet loss.

The paper is organised as follows: Section II provides background on Industry 4.0 and the network control challenges that exist. Section III gives the experimental design, broken into control system architectures, experiment overview, the testing methodology and delay mitigation. Section IV presents the results obtained, describing the effects of varying network conditions on the performance of the IIoT system. The paper is concluded in section VI.

II. Industry 4.0 network control challenges

Industry 4.0 is a reference to the fourth industrial revolution originating in 2011 from a technology strategy by the German federal government.

The fourth industrial revolution constitutes a combination of IoT, specifically Industrial IoT, and Cyber-physical systems(CPSs) in manufacturing environments [2].

A. *Internet of Things*

The idea of IoT depicts devices or *things* like sensors and embedded devices connected to the internet and using it as a means of communicating information and data [3]. There is, however, no formal definition of the term IoT, and different authors describe IoT differently. Some might focus on the networking and communications of the devices, and others might focus on the embedded devices as things with their limited resources available. IoT can be seen as the most general term of embedded devices and sensors connected to the internet [4].

IoT's potential is tremendous in the use of automation due to the connection of a large number of embedded systems allowing for better utilisation of the enormous amount of data they generate as well as to expand on their limited functionality due to limited resources of the individual embedded system. The development of the CPS was an effect of IoT's potential in automation. CPSs are automation systems that connect the physical world to a virtual world.

B. *Cyber-physical systems*

CPSs are an integration of computation and physical processes, industrial embedded automation systems and networks, allow for increased functionality of physical processes through their access to the cyber world. CPSs originate from mechatronic devices with integrated communication capabilities.

The remote access and control enabled by the connection in CPSs to the cyber world allow for data collection from the system. The data can then be used to decrease the downtime of the system by implementing predictive maintenance [5].

A CPS is implemented as an embedded system consisting of a control unit, a communication interface, sensors and actuators. The sensors and actuators are the connection to the physical world. The communication interface is one of the most critical aspects of a CPS, allowing for data exchange between a system of systems. The data from interconnected CPS can be stored and processed centrally on the connected cyber world integration [5].

C. *Network control challenges*

I4.0 relies heavily on the use of the internet as a communication medium. Most of the *I4.0 ready* marked products are marked as such because of the internet connection to the product and its ability to control the product from a smartphone as example [1].

A technological hurdle of IoT is supplying an adequate network connection to an immense number of devices. IoT will add trillions of new devices to the internet [6]. The modern-day internet uses an "end-to-end" principle, where the complexity is dealt with at the endpoints only, and the network is kept very simple. This principle has allowed the internet to be vastly scalable. IoT, however, requires different approaches where the end-to-end policy might not be feasible. IoT has various use cases, from real-time applications where the IP protocol is not suitable due to its unreliability, to small devices where the IP protocol can be too complicated for such a system [6].

The challenges that network controlled systems face are network latency, security and multi-user access. The most defining challenge for network control systems (NCSs) is data transmission latency. In comparison, typical systems use private media where the transmission delay can be well modelled. The transmission latency of a public and shared network such as the internet is challenging to model and predict since the data transmission route between two points is not fixed, and the network traffic varies on the different transmission paths [7]. Different network conditions result from different routes and different amounts of network traffic. Changes in network conditions include latency, packet delay variance and packet loss.

When the system requires a deterministic timing scheme, it may not be achievable by an internet-based control system due to the web-related traffic delay [7].

III. Experimental Design

A. *Control system architecture*

A typical control architecture used is closed-loop control, consisting of four parts structured in a centralised control structure, as shown in figure 1. The four parts include a controller, an actuator, a process and a sensor. The sensor produces a value based on the process and relays the information to the controller. The controller then analyses the information and provides a control signal to the actuator based on its analysis. The actuator then performs an action based on the control signal received from the controller. The action affects the process, and the loop is repeated. The controller, actuator and sensor must be wired in a point to point fashion and be physically located in close proximity. A typical closed-loop control system presents negligible signal loss with no time delay in the signal transfer but can be expensive to implement.

A NCS is in principle the same as a closed-loop control. The only difference is the communication to and from the controller, is now a network connection and not a physical connection. NCS enables an efficient but expensive centralised control system [4]. Referring to a NCS implies here control

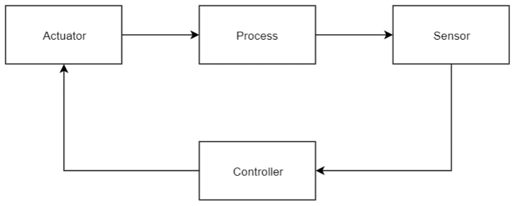


Fig. 1. Typical closed-loop control diagram.

over a network. It does not refer to the control of a network. This is an important distinction to be made [7].
NCS can use existing and shared communication networks, reducing cost and allowing access to other points within the network. The use of existing and shared network communication infrastructure adds additional network conditions such as signal delays, delay variances and packet loss due to over-congestion of the network [7]. The shared network connections can be either local or global. Internet-based control systems are NCSs that use a global connection, the internet, as a shared communication network [7].

B. Experimental overview

The goal is to add a network-connected remote controller to a typical closed-loop control system. When a remote controller is used latencies, jitter and packet loss are introduced between the actuator, sensors and controller. The proposed architecture is shown in figure 2. The remote controller is either connected through a Local Area Network (LAN) or a Wide Area Network (WAN). In the case of a LAN connection, the remote controller is seen as a local cloud or fog node.

The control architecture proposed is that of a bilateral controller, where local controllers are placed at the process/plant side, and a remote controller is placed at the operator side with the controllers connected through a network connection. The local controller is responsible for the regular operation of the process, including fail-safes of the process. The remote controller is then used for monitoring and changing parameters of the operations and control across interdependent systems or processes.

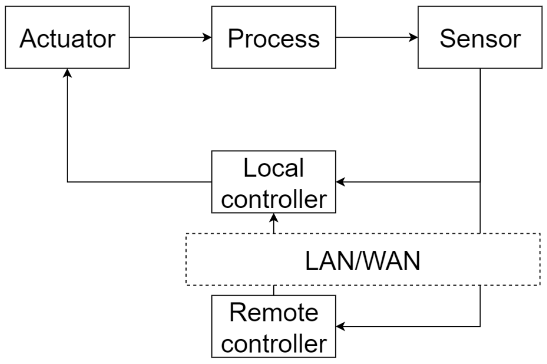


Fig. 2. Proposed bilateral closed-loop control architecture.

The network architecture is designed to resemble the proposed bilateral closed-loop control architecture shown in figure 2. The system emulates an industrial plant with three electric motors. Each motor is controlled locally by a Variable Speed Drive (VSD) and a Programmable Logic Controller (PLC). The individual local controllers are then connected to a remote controller through an industrial IoT gateway. The remote controller is responsible for control over the whole interconnected system. The system's networking diagram can be seen in figure 3. Measured motor velocities by the VSDs are sent via an industrial ethernet connection to the PLCs. The PLCs, in turn, send the velocity data to the IoT gateway. The remote controller receives the velocity data from the IoT gateway, processes the data and generates speed set-points for each motor. The reference set-points are sent back to the IoT gateway to be relayed to the PLCs and VSDs for execution on the relevant electric motor. The IoT gateway used, allows for two separate network connections. The IoT gateway's first connection is to an industrial ethernet connecting to the PLCs and VSDs. The second connection is to a consumer network connected to the remote controller. To evaluate the effects of different network conditions on the performance of the system, a computer acting as a network emulator by running the WANem (short for WAN emulator) software is connected to the consumer network, allowing for the network conditions between the IoT gateway and the remote controller to be altered.

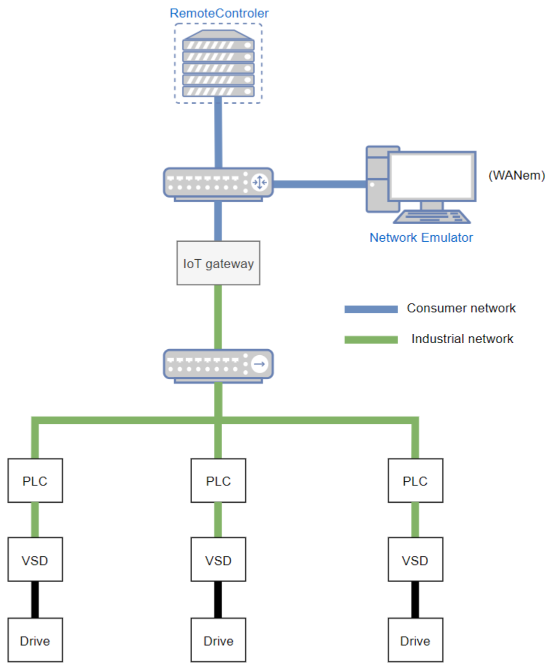


Fig. 3. Experimental system network diagram

C. Testing methodology

In the emulated system, the reference speed for each motor is determined by the remote controller. Speed measurements at discrete-time intervals are sent from the motor's local controller to the remote controller through the modifiable communications channel. The remote controller uses the speed information received from one motor to determine the speed reference of another based on the mission scheme for the emulated system. The reference speed is then sent from the remote controller to the relevant motor's local controller, through the same modifiable communications channel.

A baseline control group set of data is created for comparison. The control group data set constitutes the speed of the motors at discrete time intervals from the physical system. An average system response time is then determined for the difference in speed at each time interval between the motors in the systems.

The average system response time is obtained using the average time difference between a motor's measured speed and the expected speed of that motor based on the control scheme for the system.

The control group data set of average system response times are established with the best possible network conditions and without any delay mitigation mechanisms implemented. The average system response time gathered from the control group is used as a reference when compared to datasets from experiment runs where network delays, jitter and packet loss are added.

Tests are conducted to determine the effects of network latency, jitter and packet loss on the emulated system. The test will show the impact on an industrial application on a commercial network and how the network requirements change to fit the industrial use. The test is re-run with different network conditions with and without delay mitigations implemented on the remote controller. The data received for each test are then compared against the control data set, and error values for each test are determined.

Test one determines the system performance impact of moving the network location of the remote controller from LAN to WAN. This is achieved by implementing the remote controller's script on the IoT gateway for LAN. The remote controller is then implemented on the remote server connected via a WAN. The system performance is measured and then compared for a LAN and WAN connected remote controller.

Test two explores the effects of network latency by adding only network communication delays to the system. Delays from 0 to 500 ms are introduced in intervals to the system measuring the performance impact thereof on the system.

Test three explores the effects of both delays and jitter on the system. As before, delays are introduced in intervals from 0 to 500 ms. At each latency-interval additional jitter is added to network communication in intervals from 0 to 50 ms. The performance impact of jitter is then measured for each interval.

Test four explores the performance impact of packet loss on the system. Packet loss, measured in percentage, is introduced to the network communication in intervals from 0 to 10%. The performance impact of packet loss is then measured for each interval.

Once the performance impact of latency, jitter and packet loss on the system has been determined, the tests are re-run with delay mitigation implemented on the remote controller. The system performance is measured to determine how the effects of network conditions can be decreased when delay mitigations are implemented.

D. Delay mitigation

An approach to implement a local delay mitigation mechanism, as shown in [4] is to add a time-out to the controller. The controller will wait for a response, and if a response is not received within the time frame given, then the controller will use a predicted response value instead. Any late responses received will then be buffered and can then either be used or be discarded. This is a form of dynamic offloading where the controller will use prediction models to estimate changes rather than continuously checking for changes.

The prediction models considered for delay mitigation are the Exponential Moving Average (EMA) and the Double Exponential Smoothing Model (DESM).

1) Exponential moving average: The exponential moving average is a weighted moving average of data value. The mathematical expression is given in (1) [4] with s_t the statistical value and s_{t-1} the previous statistical value calculated. x_t is the current observed value and A the smoothing value, with $0 < A < 1$.

$$s_t = Ax_t + (1 - A)s_{t-1} \tag{1}$$

The forecast for the next value predicted is given by:

$$F_{t+1} = s_t \tag{2}$$

The initial value for the exponential moving average is described as $s_1 = x_0$.

2) Double exponential smoothing model: The double exponential smoothing model introduces a term for taking the change of slope or trend into account based on the *Holt model*. The mathematical expression is given in (3) with b_{t-1} the trend calculated in (4). A new smoothing factor B is introduced, to weigh the trend of the slope, where $0 < B < 1$ [4].

$$s_t = Ax_t + (1 - A)(s_{t-1} + b_{t-1}) \tag{3}$$

$$b_t = B(s_t - s_{t-1}) + (1 - B)b_{t-1} \tag{4}$$

A forecast can then be made for x_{t+m}, with $m > 0$. The forecast is then calculated as shown in (5).

$$F_{t+m} = s_t + mb_t \tag{5}$$

The initial values for the double exponential smoothing model are $s_1 = x_0$ and $b_1 = x_1 - x_0$.

IV. Results

A. performance impact based on the network location of the remote controller.

Figure 4 shows the impact of moving the remote controller of the system from a local area network (LAN) to a wide area network (WAN), from test one. The left bar in the bar graph indicates the system response time when the remote controller is implemented natively on the IoT gateway, emulating a remote controller on a LAN connection. The next bars to the right indicates the response time of the system when the remote controller is connected via a WAN connection. The effects on system performance are shown when no delay mitigations are implemented as well as when delay mitigations are implemented.

An increase of 0,187 s or 102% in system response time is observed when moving the remote controller from a LAN to a WAN connection without any delay mitigations implemented on the remote controller. An increase of 0,199 s or 108% is observed when EMA is implemented. When the DESM is implemented on the remote controller, an increase of only 0,047 s or 26% in system response time is observed when the remote controller is moved from a LAN to a WAN connection.

When comparing the different algorithms on a WAN connected IIoT system: The exponential moving average as delay mitigation algorithm performed 3% (0.012 s) slower than the system with no delay mitigations implemented. However, the system with double exponential smoothing model implemented as delay mitigation algorithm performed 38% (0.140 s) faster.

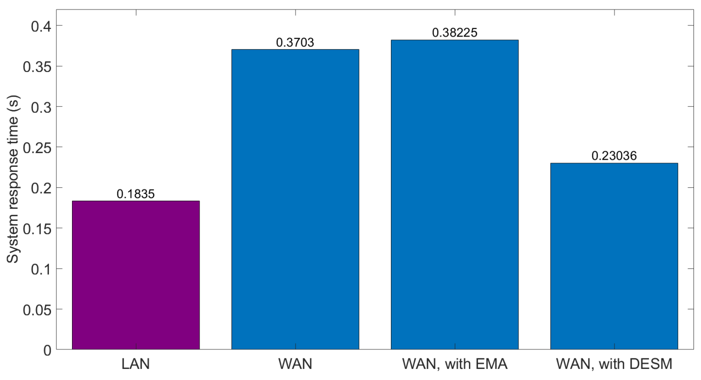


Fig. 4. Average system response time bar chart - Impact of controller's network position

B. Performance impact of network latency

Figure 5 shows the system response time for different network latencies obtain from test two. The figure shows the effect of network latency on the response time of the system with and without delay mitigations implemented. From the graph, it is evident that as network latencies increase so does the system response time. It is also apparent from the graph that the first delay mitigation algorithm, EMA, yields lower performance than that of the system with no delay mitigations implemented. In contrast, the performance is increased when the second delay mitigation algorithm, DESM, is implemented.

Applying a polynomial quadratic curve fitting to the effects that network latency has on the system response time without delay mitigations implemented is shown in (6). Where τ is the network latency round trip time (RTT) in milliseconds, and $f(\tau)$ is the systems response time of the system for the given network latency is seconds. The root mean square error

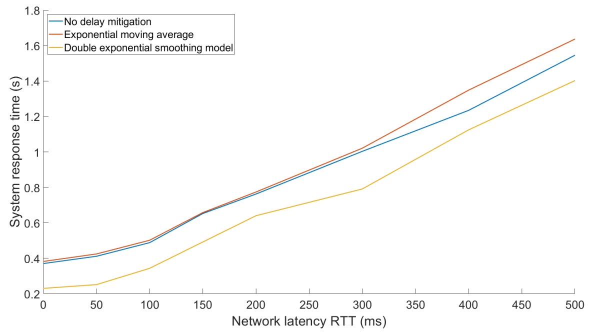


Fig. 5. Average system response time line graph - Impact of network latency

(RMSE) for the polynomial fitting is 0.0293 s. The curve fitting indicates with the small quadratic term on the curve fitting that the relationship between network latency and the performance of the IIoT system are mostly linear and slightly quadratic.

$$f(\tau) = 1.266e^{-06}\tau^2 + 0.00177\tau + 0.3415 \tag{6}$$

C. Impact of network jitter

Shown in figure 6 is the average effect of jitter on system response time defined by test three. The average value represented for each jitter test point is determined as the moderate impact of that jitter test point for all the tested latency intervals. The figure shows the effects of jitter on a system without delay mitigations as well as the system with delay mitigations implemented.

As before, an increase in system time is observed when jitter within the network is increased. A substantial increase of 15% (0.1238 s) in system response time is observed when moving from 0 ms jitter to 10 ms jitter. After that, the effects of jitter on the system response time can be approximated by (7) with an RMSE of 0.0008 s, obtained by applying linear curve fitting to the data points. In (7) $f(\tau)$ represents the system response time in milliseconds and τ the network jitter value in milliseconds. The curve fitting indicates a positive linear relationship between jitter and system response time.

$$f(\tau) = 0.001282\tau + 0.9192 \tag{7}$$

D. Impact of packet loss

Packet loss represented as a percentage of packet transmissions that are lost during transmission. The impact of network packet loss on the performance of the IIoT system, as determined by test four, is shown in figure 7 for the system with and without delay mitigations implemented. The impact of packet loss increased the system response times of the IIoT system in a quadratic manner as shown by applying a polynomial quadratic curve-fitting on the system without delay mitigations that yield (8) with an RMSE of 0.00154 s, that can be used to approximate the system response time based on a given percentage of network packet loss.

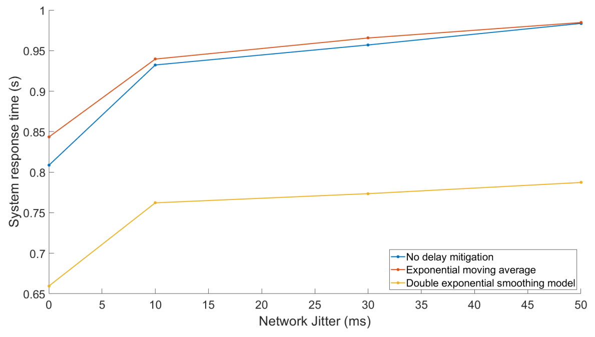


Fig. 6. Average system response time line graph - Impact of network jitter

$$f(\tau) = 0.001169\tau^2 + 0.01115\tau + 0.3698 \tag{8}$$

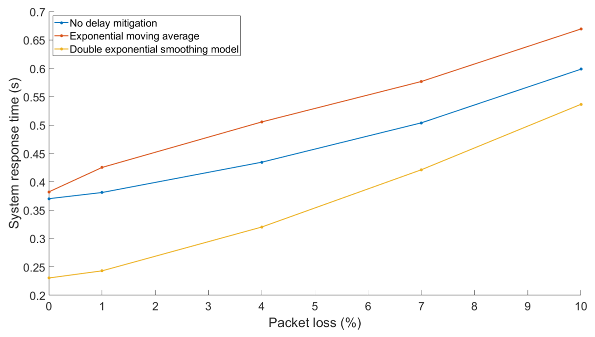


Fig. 7. Average system response time line graph - Impact of network packet loss

V. Discussion

Test one indicates that moving the remote controller from a LAN to a WAN connection degrades the performance of the IIoT system. Implementing DESM as delay mitigation on the WAN connected remote controller improves performance, but not to the extent when the remote controller is connected via LAN.

Test two shows a linear or slightly quadratic relationship between system response time and network latency. A linear relationship is expected, but increased network overhead with increased latency results in a somewhat quadratic relationship.

Test three indicates a considerable increase in system response time when jitter is introduced. A linear relationship between network jitter and system performance is observed. A linear relationship is seen, because of the right-skewed, shifted gamma distribution of jitter values and the use of average system response times for results.

Test four shows a quadratic relationship between system response time and network packet loss. An increase in network overhead occurs when packet loss is introduced, as packets that are lost are retransmitted, increasing network traffic.

Applying EMA as delay mitigation yielded lower performance as the algorithm is better suited for small system changes. The implementation thereof also introduced an increase in processing time. DESM, as delay mitigation improves system performance through its improved prediction ability.

VI. Conclusion

The performance impact of different network conditions between the remote controller and the local controllers on an emulated system of electric motors was investigated. The results indicated that for a system with no delay mitigations implemented that the relationship between the system response time and the tested network condition appear to be linear or slightly quadratic for latency and jitter, and quadratic for packet loss. As the various network conditions (latency, jitter, packet loss) increase, so does the system response time, thus decreasing the system performance.

A 102% increase in system response time is observed when the remote controller is moved from a LAN connection to a WAN connection with best-case network conditions. Therefore it can be concluded that using a local cloud or fog node is a better solution for the emulated system than cloud nodes. Remote cloud nodes are best kept for less critical soft real-time tasks that can handle delays with more flexibility.

The system with delay mitigations implemented had contrasting results. The first delay mitigation method, exponential moving average, caused a decrease in performance as the algorithm is best suited for small compensations. However, the double exponential smoothing model consistently increased the system's performance when compared to the results of the system with no delay mitigation at the same networking conditions. The double exponential smoothing model algorithm's better system performance is attributed to its better forecasting ability.

An increase of 21% in system response time is still observed when the remote controller is moved from a fog node to cloud node using the double exponential smoothing model for delay mitigation. The best possible performance for the emulated system is therefore observed when the remote controller is implemented on a fog node on a local network connection.

References

[1] L. Bassi, "Industry 4.0: Hope, hype or revolution?" in *RTSI 2017 - IEEE 3rd International Forum on Research and Technologies for Society and Industry, Conference Proceedings*, 2017, pp. 1–6.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[3] Ashton K., "That 'Internet of Things' Thing," *RFID Journal*, p. 4986, 2009.

[4] A. Didic and P. Nikolaidis, "Real-time control in industrial IoT," 2015. [Online]. Available: http://mdh.diva-portal.org/smash/get/diva2:821335/FULLTEXT01.pdf

[5] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," in *Proceedings of 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2014*. IEEE, 2014, pp. 1–4.

[6] R. van Kranenburg and A. Bassi, "IoT Challenges," *Communications in Mobile Computing*, vol. 1, no. 1, pp. 1–5, 2012.

[7] H. Shuang Yang, *Internet-based Control Systems*. Springer, 2011.