CODEN:LUTEDX/(TEIE-5483)/1-63(2022)

Industrial Electrical Engineering and Automation

Latency Aware and Event-Based Wireless Control for Cloud-Fog Automation

Anna Bengtsson
Sofie Nilsson

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

MASTER THESIS

Latency Aware and Event-Based Wireless Control for Cloud-Fog Automation

JUNE 2022

Anna Bengtsson & Sofie Nilsson

Principal supervisor: Gunnar Lindstedt, IEA

Assistant supervisor: Zhibo Pang, ABB
Assistant supervisor: Alf Isaksson, ABB

Examiner: Ulf Jeppsson, IEA

LUND UNIVERSITY

Abstract

The development seen within industrial automation in recent years enables the emergence of manufacturing sites that utilise technology for improved productivity, quality and safety. Increased communication demands introduced by Industry 4.0 have motivated the development of cloud-fog automation; a technique characterised by wireless communication between lower levels of the automation pyramid. Specifically, the new communication requirements gives rise to the need for efficient wireless communication between input/output devices and controllers.

This thesis presents findings from an investigation of wireless control and communication performance. A cascaded internal model control-based Proportional, Integral, Derivative controller structure is developed to control a time-critical motion system. Various communication protocols and control algorithms are considered to examine the performance behaviour. The results indicate that user datagram protocols, compared to transmission control protocols, are a promising candidate for the target purpose. However, by manipulating the control algorithm sampling interval, transmission control protocol communication yields almost equivalent control performance results.

An exploration of event-based control results in an algorithm that achieves communication reductions of over 80%. Given a consciously chosen sampling interval, control performances comparable to those of the corresponding time-based system can be maintained. The research indicates that wireless control has the potential to accomplish control performances equal to those found in wired systems.

Keywords: *Wireless control, Cloud-fog automation, Latency, PID control, Event-based derivative control, Threshold tuning*

Acknowledgements

This thesis would not have been possible without the great support and guidance from our supervisors at ABB Corporate Research, Zhibo Pang and Alf Isaksson. The time they have spent with us, sharing their knowledge and experience, has been truly invaluable. A sincere and warm thank you should also be directed to all of our colleagues at the ABB Corporate Research Center, our thesis experience would not have been the same without you.

A second enabler for this thesis is the help and support provided by our supervisor at the division of Industrial Electrical Engineering and Automation at LTH, Gunnar Lindstedt. Furthermore, we would like to thank him and our examiner, Ulf Jeppsson, for their inspiring automation courses which introduced us to this highly interesting topic.

Finally, we wish to express our gratitude to our families for their inexhaustible support, not only during this thesis but throughout our entire academic journeys.

Anna & Sofie

Acronyms and Abbreviations

5G	Fifth generation cellular network
AIO Device	Analog Input Output Device
CPU	Central Processing Unit
HMI	Human Machine Interface
IEC 61131-3	International Electrotechnical Commission Standard
IMC	Internal Model Control
I/O	Input Output Device
IP	Internet Protocol
ISA95	International Society of Automation Standard
KPI	Key Performance Indicators
MAE	Mean Absolute Error
MPC	Model Predictive Control
OPC UA	Open Platform Communications United Architecture
OSI	Open Systems Interconnection
PID	Proportional, Integral, Derivative
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Acquisition
TCP	Transmission Control Protocol
UADP	Unified Access Data Plane
UDP	User Datagram Protocol
UE	User Equipment
vPLC	Virtualised PLC

Contents

List of Figures

List of Tables

1
Introduction

This chapter introduces the topics of wireless control and communication which are central concepts in this thesis. The objectives and limitations related to the project are presented as well as the division of labor and the structure of the succeeding content.

1.1 Project Description

The development seen within wireless communication during the past years contributes to the development of automatic control using wireless instead of wired interfaces [15]. This provides an opportunity to unlock the real potential of Industry 4.0 while also enabling cost reductions, design simplifications and increased mobility of the controlled systems. However, a wireless communication strategy also imposes challenges, especially in the control of time-critical processes where wireless communication is sometimes too slow to satisfy the systems needs.

1.2 Objectives

This thesis is part of an extensive project at ABB Corporate Research Center with the objective to create a common test bed for wireless automation. The intended outcome of this individual thesis is to construct a reliable and well-performing wireless control algorithm for a time-critical motion system. The main focus is targeted at reducing the effects of latency introduced by wireless communication on the control. This will be done by first deploying a controller tuning presented in a previous master thesis conducted at ABB, which explicitly takes latency into account [9]. The control algorithm implementation will then be adjusted to reduce the influence of the present latency. The objectives described above are condensed into the following research questions:

1. What is the state-of-the-art of cloud-fog automation and wireless control methods?
2. How do the communication and control strategies affect the performance of a time-critical motion process?
3. How can the performance of these strategies be evaluated?
4. Can the need for communication be reduced by using event-based control?

1.3 Limitations

The project was carried out with means mainly composed of physical experiments conducted on one single physical process. The results were dependent on this equipment, its resolution and the manually tuned and identified variables. Simulations of the process could have strengthened the presented findings and conclusions. Though, the projects time constraints did not allow for this. Additionally, the physical experiments were carried out in a controlled lab environment, meaning that any disturbances potentially present in real industrial settings are eliminated. Another restriction was imposed by the development environment, which bounded the work to the available functionalities and library versions.

1.4 Division of Labor

The work has been conducted in close collaboration between the two authors. The labor was equally divided and occasionally distributed to enhance efficiency. The most significant division was made within the design and development of the controllers. Anna Bengtsson was responsible for the Internal Model Control design and Sofie Nilsson for the event-based control strategy. However, during the implementation of said topics, both authors were equally involved. Pair programming was conducted in all coding tasks to ensure productivity and quality.

1.5 Outline

This report is divided into 6 chapters. Chapter 1 gives a general introduction to the project. Chapter 2 covers the theory and concepts related to the thesis work. Chapter 3 presents the undertaken approach to investigate the scope of the project and arrive at the results presented in chapter 4. The results are then concluded and analyzed in chapter 5. Finally, in chapter 6, future research areas are suggested.

2 Background

This chapter provides the theoretical background required for the thesis. The material gives a thorough introduction to wireless communication, related concepts and the possibilities they enable. The devices and an automation development platform are presented as well as the examined communication protocols and a controller tuning framework. The last sections introduce discrete and event-based control.

2.1 Wireless Communication

In recent years, a rapid development has been seen within wireless communication, computing and control [15]. This has provided opportunities to introduce Internet technologies within industry. Industrial processes can utilise the networked control and manufacturing that characterises Industry 4.0 [19].

2.1.1 Wireless Control Systems

One component of Industry 4.0 is the introduction of wireless control systems. Historically, industrial control systems have been designed using a wired architecture in which sensor values are sent over wired interfaces to a controller where they are used to determine the actuator signals.

The transformation of these control systems, from wired to wireless means that the communication between sensors, controllers and actuators is deployed over a wireless link. There are several advantages of the wireless architecture that motivate the technical shift. Intuitively, wireless communication enables higher flexibility of the system. Additionally, it entails economic savings due to reduced material requirements, simplified design and installation processes, and diminished maintenance needs. Finally, although the wireless systems experience temporary fails at higher frequencies their long-term reliability is higher in comparison to their wired counterparts [15].

The wireless control systems introduce some drawbacks as well, mainly related to their performance. Industrial applications typically require deterministic communication with short delays and high reliability. The amounts of data sent are usually low but require a high communication frequency [15]. These needs have historically not been fulfilled by the available wireless alternatives. The communication time of the wireless system would have resulted in unstable control for many industrial applications [13].

2.1.2 Control-Loop Latency

The time it takes for a measurement from a sensor to result in a control signal realised by an actuator can be referred to as the control-loop latency. A previously stable closed-loop system might become unstable if this latency grows too big. This is what might happen when the communication of a system is transformed from wired to wireless since wireless communication is typically slower than wired. To resolve the potential instability issue, different solutions are available. One of them is to alter the control algorithm to make it suitable for the current system control-loop latency.

2.1.3 5G and Edge Computing

One of the emerging techniques for communication that enables the use of wireless control systems within industry is the fifth generation of cellular networks, 5G. Due to its high communication speed, it might have the capability to reduce the control-loop latency compared to previous generations of cellular networks [13].

With the development of Industry 4.0, the networked industries of the future will consist of large amounts of connected devices. The computing power of individual end-devices are already today complemented by remote data centers in the cloud. The growing number of connected devices on our networks will result in even higher demands on this type of remote computations. However, when data is sent from an end-device to a distant cloud data center, an extra delay is introduced. For a closed-loop control system, that is highly affected by its control-loop latency, the deployment of the calculations in the cloud might result in an unstable system. One possible solution to this problem is to limit the distance to the computing nodes. By deploying the computations in edge data centers, which are smaller data centers located closer to the end-devices, the delay introduced by the communication can be reduced. This can be referred to as fog computation and will be further described in section 2.1.4. Latency-sensitive closed-loop control applications that, for example, can be found in process control and automation are prominent potential test cases for this computational strategy [13].

2.1.4 Cloud-Fog Automation

Cloud-fog automation is a development of the traditional pyramid automation. Historically, factory automation has followed the hierarchical approach defined in ISA95 [7] that can be seen to the left in figure 2.1. The traditional design yields that the layers of the pyramid can only communicate with their neighboring layers. Direct communication with other layers is not possible.

The introduction of cloud automation entails that the two upper layers of the automation pyramid may potentially be deployed in the cloud. This can be seen in the middle of figure 2.1. Furthest to the right in the same figure a representation of cloud-fog automation is shown. Here, the lower-level applications of the pyramid are deployed over local wireless networks. This is illustrated by the second cloud in the figure, which represents the fog and includes the supervisory controllers, the machine controllers and the input/output modules. Advantages of the cloud-fog automation approach are based on its ability to transfer calculations from individual components of the pyramid to the fog or the cloud. Firstly, this entails possibilities to significantly reduce the costs of system components since the computational requirements on these will be reduced. Secondly, performing computations in the cloud and the fog enables the use of more advanced algorithms and machine intelligence since the computation platform has the ability to be more cost-effective and powerful [11].

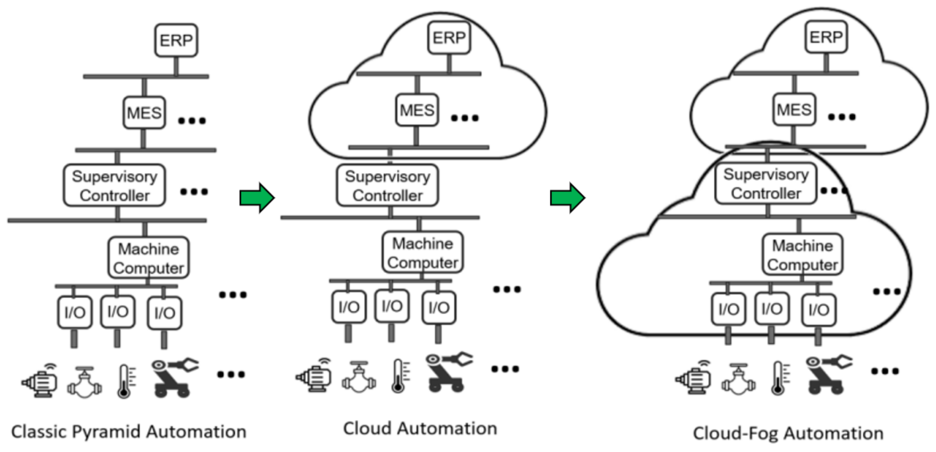


Figure 2.1: The evolution from the classical pyramid automation to cloud-fog automation [11]

2.2 Programmable Logic Controllers

Programmable logic controllers (PLCs) have been widely used in industry since they replaced hard-wired logic circuits of relays in the 1970s. The new solution offers a more efficient and flexible controller with advanced features like communication between PLCs and the possibility to locate the device distanced from the production lines. The international standard IEC 61131-3 defines a number of different programming languages that can be used to program a PLC [17].

2.2.1 Hardware Controllers and I/O Units

A hardware PLC is a physical unit consisting of a central processor, memory, power supply, communication modules and modules to receive and transmit data, also known as Input/Output (I/O) units [17]. The controller actuates the manipulated variables based on sensor inputs and setpoints within the plant or factory process.

2.2.2 Software Controllers

Besides physical PLCs, a PLC can also be ”software-based”, meaning that the control is running from within a PC environment. Advantages of virtualised PLCs (vPLCs) include them being scalable and cheaper compared to hardware ones. In combination with the emerging technologies of cloud-based solutions vPLCs can become of importance for areas such as Industry 4.0 and Industrial Internet. vPLCs can be used for applications and tasks that require more computational power than what is traditionally available in the lower layers of the automation pyramid [16].

2.3 CODESYS

CODESYS is a software platform for industrial automation technology from the CODESYS Group. It provides a development environment system, a programming tool supporting the languages included in the international standard IEC 61131-3. The tool includes several library add-ons to enable control application development. From within the tool, programs can be uploaded to the desired devices by connecting to them. If the connection is kept established, the tool also enables real-time monitoring of variables when the device is in running mode [3].

2.4 Industrial Communication Protocols

The action of sending and receiving information is typically referred to as communication. A communication model is an idealised, systematic representation of such actions [28]. The Open Systems Interconnection (OSI) model is commonly used for industrial communication. It contains seven layers to represent the process of transmitting data from one node to another in a network, as can be seen in figure 2.2. Each layer can only communicate with corresponding layer in another node and the closest preceding and succeeding layers of the same node. The layers of the model represent different abstraction levels and is utilised by different protocols to implement the data exchange [25]. The following sections will therefore be dedicated to briefly introducing two common types of protocols.

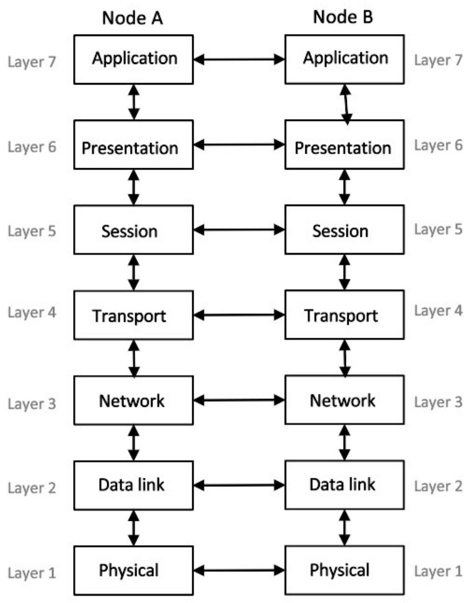


Figure 2.2: An overview of the layers in the OSI reference model for communication between two nodes on different layers

2.4.1 Modbus TCP

Modbus Transmission Control Protocol (TCP) is an application-layer communication protocol for data exchange at level 7 of the OSI model. Modbus was first developed during the 1970s and is a widely used industrial standard even today. It is based on a client/server model that enables real-time communication between devices on an Ethernet network, using the Transmission Control Protocol/Internet Protocol (TCP/IP) stack. The devices can, for example, be a PLC and an I/O device, although several different types of devices are possible, such as Human Machine Interfaces (HMI), Supervisory Control and Data Acquisition (SCADA) applications, drivers and computers. The client sends a request with a function code to initiate the transaction, received as an indication of what kind of action to perform on the server. The server then provides the data by sending a response with the same function code as a confirmation to the client. The function codes range between 1 and 255. Common commands are to read and write values to and from registers [30]. All the data handled via Modbus TCP must be located in the device application memory. The system port 502 is reserved on the TCP/IP stack by default for Modbus communication [21].

2.4.2 CODESYS OPC UA PubSub and CODESYS Network Variables

Open Platform Communications Unified Architecture (OPC UA) was released in 2008 and is a communication protocol that is platform independent which means that it can be used over several hardware platforms and operating systems. An available configuration of it is called Publish-Subscribe (PubSub). It is optimised for communication between several devices. No direct exchange of requests and responses are made when OPC UA PubSub is used. The publishers do, instead, send their messages to a Message Oriented Middleware. Thus, they have no information regarding the subscribers of their messages. Correspondingly, the subscribers subscribe to the data they are interested in and receives it without information of the publisher [2].

CODESYS OPC UA PubSub is a communication protocol implementation released in October 2020 that is used for communication between devices via the User Datagram Protocol/ Internet Protocol (UDP/IP). The communication follows the rules defined for the Unified Access Data Plane (UADP). The data is transferred in a binary format according to the standards defined by the OPC UA Foundation [6]. Another CODESYS communication protocol based on UDP is the CODESYS Network Variables protocol. It enables communication of variables between CODESYS programs running on different devices in the same network [4]. The protocol can be used from version 3.5.14.0 of CODESYS [5], released in December 2018 [3].

CODESYS OPC UA PubSub and CODESYS Network Variables are using UDP which is a protocol defining the fourth layer of the OSI reference model. UDP is said to be connectionless since, in comparison to TCP, it lacks handshaking between the transportation layers of the sender and the receiver. This entails less data flow on the network and enables for higher communication speeds over UDP than the ones achievable with TCP. However, the absence of handshaking also results in the lack of an acknowledgement for the sender to confirm that the data has been received. This is the greatest disadvantage of UDP communication, it does not guarantee the arrival of data at the destination [22].

2.5 IMC Based PID Control

Internal Model Control (IMC) was initially described as a strategy that provides continuous-time control for a single input, single output continuous process. The order of the process that is to be controlled will determine the order of the controller that is designed according to the IMC framework [37]. This specific feature can be applied in the design of Proportional, Integral (PI) and Proportional, Integral, Derivative (PID) controllers. Using the IMC approach for a first or second order system will result in a first or second order controller that can be rewritten as a PI or a PID controller respectively. Hence, IMC provides a PID parameter tuning framework that is logical and comprehensible. Contrary to the classical PID tuning strategies, where optimality is to be extracted from a fixed control structure, the IMC approach delivers the structure as well as the parameters for the controller based on user defined control objectives. An advantage of the IMC design is that the resulting PID parameters have the system's closed-loop time constant as their only tuning parameter [36].

The intent of the IMC approach is to find parameters that are based on functions of the open-loop system parameters and parameters defining the closed-loop system to which one wants to arrive [34]. This section will focus on explaining how it is achieved based on a published tuning framework [36].

Figure 2.3 depicts an open-loop control system. An advantage of this type of system is that it provides fast and accurate setpoint tracking. A stable process and a stable controller yield a stable system in the open-loop case, which is yet another advantage with the system setup.

However, the open-loop systems cannot handle unmeasured disturbances and the performance is highly dependent on the accuracy of the plant model. A control structure as the one depicted in figure 2.3 is also unable to stabilise an unstable system.

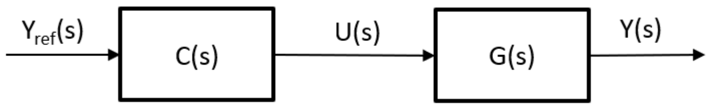


Figure 2.3: Block diagram of the open-loop control system where $C(s)$ is the controller, $G(s)$ is the process, $Y_{ref}(s)$ is the reference signal, $U(s)$ is the control signal and $Y(s)$ is the measurement signal

Closed-loop feedback systems can, contrary to open-loop systems, handle model mismatches and unmeasured disturbances well. However, tuning them becomes more complicated due to the closed-loop stability problem. An example of a closed-loop system can be seen in figure 2.4.

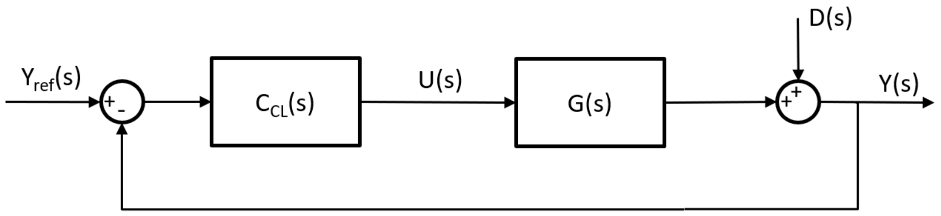


Figure 2.4: Block diagram of the closed-loop feedback control system where $C_{CL}(s)$ is the closed-loop controller and $D(s)$ are disturbances acting on the system

The IMC structure can be seen in figure 2.5. In the rare occasions of having a perfect process model, meaning that $\tilde{G}(s) = G(s)$, and no disturbances are acting on the system, $D(s) = 0$, feedback is not needed and the IMC system becomes structurally the same as the open-loop system. This is positive since it entails simplifications in the controller design process whilst the IMC system still retains the benefits of being a closed-loop feedback system.

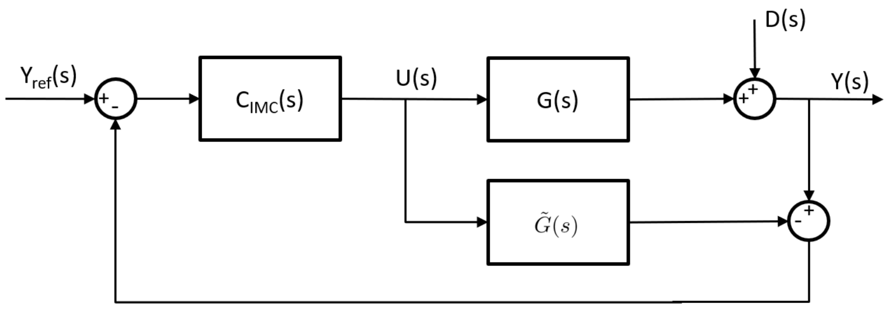


Figure 2.5: Block diagram of the IMC system where $C_{IMC}(s)$ is the IMC controller and $\tilde{G}(s)$ is the model of the process

Given the circumstances described above, the theoretical closed-loop transfer function of the system becomes

$$G_{sys} = C_{IMC}(s)G(s) \tag{2.1}$$

If the IMC controller is chosen to be $C_{IMC}(s) = F(s)\tilde{G}(s)^{-1}$ and $\tilde{G}(s) = G(s)$, it is possible to arrive at a system that is simply described by the user-defined filter $F(s)$

$$G_{sys} = C_{IMC}(s)G(s) = F(s)\tilde{G}(s)^{-1}G(s) = F(s) \tag{2.2}$$

However, if $G(s)$ is non-invertible due to it being unstable or having a time delay, some alterations are required. The transfer function $G(s)$ is then factored according to

$$G(s) = G_{-}(s)G_{+}(s) \tag{2.3}$$

where $G_{+}(s)$ contains all time delays and instability-creating elements. Thus, $G_{-}(s)$ is stable and can safely be inverted. The IMC controller is now instead

$$C_{IMC}(s) = F(s)\tilde{G}_{-}(s)^{-1} \tag{2.4}$$

Through block diagram calculations and a comparison between the closed-loop system in figure 2.4 and the IMC system in figure 2.5, it can be seen that the relation between the IMC controller, $C_{IMC}(s)$, and the closed-loop controller, $C_{CL}(s)$, can be described as

$$C_{CL}(s) = \frac{C_{IMC}(s)}{1 - \tilde{G}(s)C_{IMC}(s)} \tag{2.5}$$

By combining this with the expression for $C_{IMC}(s)$ found in (2.4), the final description of the closed-loop controller becomes

$$C_{CL}(s) = \frac{F(s)\tilde{G}_{-}(s)^{-1}}{1 - F(s)\tilde{G}_{+}(s)} \tag{2.6}$$

2.6 Discrete Control

Realising continuous-time controllers and handling sampled systems in computers require discrete time approximations. If the sampling frequency is sufficiently high compared to the frequency content of the control signal the discrete- and continuous-time controllers will be of close correspondence. The continuous transfer functions are transformed to pulse transfer functions by replacing the differential operator by a difference approximation. Common approximation methods include backward and forward difference, also referred to as Euler's methods, and Tustin's approximation. The methods map the stability regions in slightly different ways, where Tustin's method offers the advantage of mapping the left-half s-plane into the unit disc. [32]

2.6.1 Discrete PID Control

The standard PID controller described in the time domain is given by

$$u(t) = K_p e(t) + K_i \int e(t)\, dt + K_d \frac{de}{dt} \tag{2.7}$$

The control signal u is sent to the actuator, the error e is the difference between the setpoint for the process, y_{ref}, and the measured output value, y. The tuning parameters K_p, K_i and K_d are the respective gains for each controller part. K_p contributes with a proportional part related to the control error acting as a pure gain while the integral part K_i ([$1/s$]) reduces the occurrence of static process errors by acting on the integral of the error. The derivative part K_d ([s]) predicts the future behavior of the process and one of its purposes is the reduction of overshoots and oscillations [12]. The derivative part is often implemented in combination with a filter since a pure derivative could largely amplify measurement noise. Additionally, it is common to take derivative action only on the process value and not the error [32]. This realisation means that a change in the setpoint would not affect the derivative action, which otherwise could spike undesirably on those occasions.

Transformed into the Laplace domain, the controller is described by

$$U(s) = \frac{K_d s^2 + K_p s + K_i}{s} \tag{2.8}$$

An advantage of the PID controller is that the proportional, integral and derivative parts can be discretised separately and then combined into the control signal. The proportional part does not require any approximation since it is purely static

$$u_{P,k} = K_p e_k \tag{2.9}$$

where u_k is an expression used to describe the discrete control signal $u(kT_s)$, where k is the discrete time instance and T_s is the sampling interval. The integral action $u_{I,k}$ can be expressed using an Euler backward approximation as

$$u_{I,k} = u_{I,k-1} + \frac{T_s}{T_i} e_k \tag{2.10}$$

Lastly, the derivative part can be approximated according to

$$u_D(t) = -\dot{y}(t) T_d \implies \tag{2.11}$$

$$\implies \int_{t-T}^{t} u(\tau)\, d\tau = -\int_{t-T}^{t} \frac{d}{d\tau} y(\tau)\, d\tau \tag{2.12}$$

where both sides are integrated. The left-hand side of 2.12 is in addition approximated as

$$T_s \frac{u_D(t - T_s) + u_D(t)}{2} = -(y(t) - y(t - T_s))T_d \tag{2.13}$$

Then the $q-$operator is inserted as

$$\begin{aligned} T_s \frac{u_D(t - T_s) + u_D(t)}{2} &= -(y(t) - y(t - T_s)T_d \implies \\ \implies \frac{T_s}{2}(1 + q^{-1})u_D(t) &= -(1 - q^{-1})T_d y(t) \end{aligned} \tag{2.14}$$

This is in fact the derivation of Tustin's approximation and the derivative part can, thus, be implemented as

$$u_{D,k} = -u_{D,k-1} + \frac{2T_d}{T_s}(y_k - y_{k-1}) \tag{2.15}$$

The full control law is then given by

$$u_k = u_{P,k} + u_{I,k} + u_{D,k} \tag{2.16}$$

2.7 Event-Based Control

2.7.1 General Overview

Research in and development of automatic control is most of the time conducted considering periodic sampling of continuous-time signals. Though, there might be control applications where an event-based triggering scheme for the sampling is more suitable. Such control systems are often aperiodic or asynchronous. The underlying idea is to not disturb the process with a new control actuation unless a significant deviation from the reference requires so. Communication is then executed when a predefined event occurs rather than when a specified amount of time has passed. The event trigger can be based on various happenings, such as a measurement value exceeding a certain threshold, the arrival of a data package to a network node or something entirely different, depending on what is best suited for the controlled process. Another reason for the calculation of a new control signal is the event of a setpoint change [33].

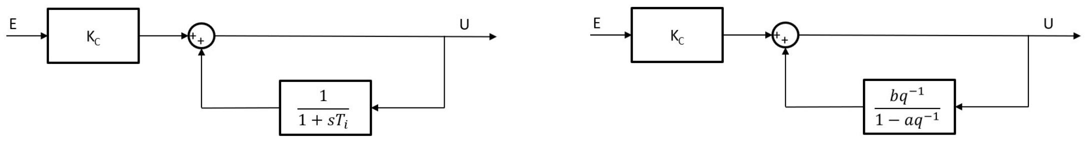
An effect of event-based data transmission, as opposed to time-driven, is the potential reduction in resource utilisation, both in terms of Central Processing Unit (CPU) cycles for controller calculations and communication bandwidth. In order to benefit from this effect, it is important that the chosen threshold is sufficiently high such that unnecessary sampling due to measurement noise is prevented. Event-triggered sampling introduces some complexity in the stability analysis. It also entails difficulties to the verification of worst-case performances using scheduling theory due to the non-uniform sampling. The interval variations can be compensated for by including the sampling time in the measurement transmission and let it be taken into account by the control algorithm [33].

2.7.2 Discrete Event-Based Control

In conjunction to what is mentioned in section 2.6, event-based controllers can be subject to discretisation. The foundation of event-based control is less communication, hence, the experienced elapsed time between incoming measurements will naturally increase. When the control is no longer executed on a periodic basis the discretised integral and derivative parts may not perform well anymore. The response to setpoint changes and disturbances occurring between measurement updates may be delayed and cause spikes in the control signal. To counteract this behavior of the classic PID controller and thereby reduce its effects, some modifications to the controller can be made [18].

Event-based control has previously been investigated by ABB, however, a PI controller without derivative action was used [27]. When derivative action is desired a well-known adaption of the PID controller, called PIDPLUS, can be used [29]. The integrator is replaced by a filter that is updated when new measurements arrives. The event-based version of the derivative part is similar to the regular one, being updated with the full sampling interval between measurements. Thus, the derivative action will become smaller the longer the time intervals are. The full control law is updated once new measurements arrive and the actuator holds the last communicated value between the instances [26].

The discretised integral part can be derived by examining figure 2.6.



(a) Controller in Laplace domain (b) Discretised controller

Figure 2.6: Block schemes representing a PI controller

The feedback block of figure 2.6a is representing a PI controller, which can be realised by the block calculation in (2.17). The transfer function from the error E to the control signal U is a classic PI controller. The controller output will be piecewise constant and can therefore be exactly discretised already in the sampling as seen in the corresponding block of figure 2.6b, where $a = e^{\frac{-T_s}{T_i}}$ and $b = 1 - a$.

$$U(S) = \frac{1}{1+sT_i}U + K_cE \implies U(S) = K_c(1 + \frac{1}{sT_i})E \tag{2.17}$$

The control signal in (2.17) can also be expressed as

$$u_k = K_c(e_k + i_k) \tag{2.18}$$

The block calculation for figure 2.6b yields u_k according to

$$u_k = \frac{bq^{-1}}{1-aq^{-1}}u_k + K_ce_k \implies u_k = K_c\frac{1-aq^{-1}}{1-q^{-1}}e_k \tag{2.19}$$

In order to separate the P- and I-part of the controller, it can be further simplified to

$$u_k = K_c\frac{1-q^{-1}+(1-a)q^{-1}}{1-q^{-1}}e_k = K_c(1+\frac{bq^{-1}}{1-q^{-1}})e_k \tag{2.20}$$

The integral part can then be identified by applying the $q-$operator as

$$u_{I,k} = K_c\frac{be_{k-1}}{1-q^{-1}} \implies u_{I,k} = u_{I,k-1} + K_cbe_{k-1} \tag{2.21}$$

By comparison to the time-based (2.10), it can be noted that the event-based (2.21) uses the previous error and not the current.

A way to prevent the integral action from building up while not receiving any new measurements is to simply limit the time used to calculate the integral part. This can be achieved by setting T_s to the maximum value of itself and T_i, so that $\frac{T_s}{T_i}$ does not exceed 1, which is depicted in figure 2.7.

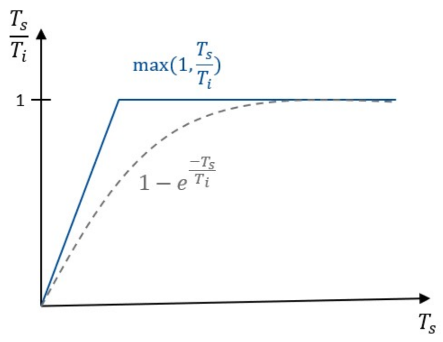


Figure 2.7: Schematic plot of the integral action-limiting mechanism

3
Methodology

The methodology used to arrive at the results is presented in this chapter. A graphical overview of the approach can be seen in the flowchart in figure 3.1. Each of the following sections corresponds to one of the steps in the chart.

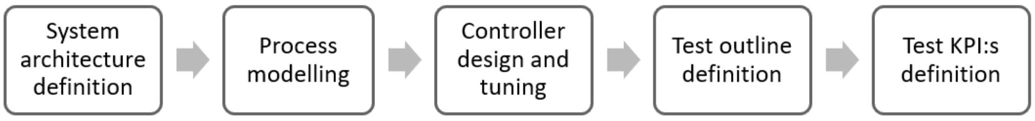


Figure 3.1: Flowchart showing the different methodology steps

3.1 System Architecture

In this thesis, a ball-and-beam process, figure 3.2, is used to represent a time-critical motion system where the performance of the proposed control algorithms can be evaluated. A ball is located on the beam and can roll freely along its full length. The servo motor that is connected to the beam can control the ball's position by adjusting the angle of the beam. A vPLC of the type CODESYS Control RTE V3 Version 3.5.17.10, realised within a Windows machine, is used to calculate the desired input voltage to the servo motor. This is translated into an angular velocity in the motor and is, thus, determining the speed at which the beam angle is changing. Because of this, the input voltage to the servo motor will henceforth be called the control signal. The calculation of it is dependent on the ball position and the beam angle, which are the measurement signals from the ball-and-beam process to the controller.

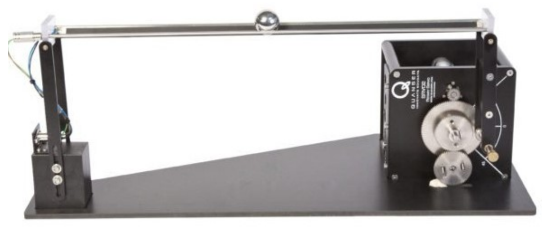


Figure 3.2: The ball-and-beam process used to evaluate the performance of the proposed control algorithms [8]

The measurement signals and the control signal are communicated to and from the process through the use of an analog I/O device. It is implemented in a hardware PLC of the type Kunbus RevPi core 3+ which has two analog I/O modules connected to it. This device will henceforth be referred to as the AIO device. A conceptual sketch of the process, the AIO device, the vPLC and an arbitrary communication link can be seen in figure 3.3. Comparisons between this system architecture and the cloud-fog automation pyramid seen in figure 2.1 conclude that the ball-and-beam process is a device in the bottom of the pyramid while the AIO device and the vPLC are located in the fog. The following section will be devoted to descriptions of the different communication link alternatives between the vPLC and the AIO device relevant to this thesis.

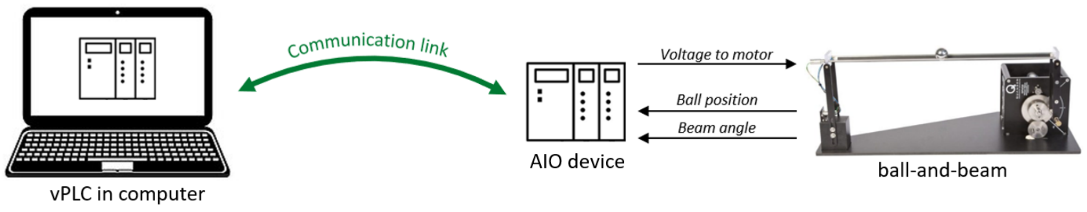


Figure 3.3: The general system architecture

3.1.1 Communication Link Alternatives

The communication link between the vPLC and the AIO device has two main characteristics, the communication medium and the communication protocol. Three different media are investigated in this thesis:

- **Ethernet:** By using a standard Ethernet cable, wired communication between the vPLC and the AIO device can be achieved. The cable is connected to the RJ45 ports on the AIO device and the computer on which the vPLC is located. The maximum speed for communication over Ethernet cable is 400 Gb/s [14].
- **WiFi 5:** One wireless alternative for connection between the devices is WiFi 5, henceforth referred to as WiFi. It has a theoretical maximum transmission speed of 7 Gb/s [20]. When WiFi is used as communication medium, the computer that hosts the vPLC is connected via an Ethernet cable to a WiFi router and the AIO device is connected via an Ethernet cable to a WiFi access point. Thus, there is no wired connection between the computer and the AIO device. They are instead dependent on the wireless connection between the WiFi router and access point to exchange information.
- **5G:** The third and final communication alternative is 5G. This wireless communication is achieved by connecting the computer to a 5G base station with an Ethernet cable. The AIO device is connected via an Ethernet cable to a 5G User Equipment (UE). The wireless connection between the 5G UE and the 5G base station entails communication between the computer with the vPLC and the AIO device. A peak data rate of 20Gb/s can be achieved using 5G [10].

The second characteristic of the communication link is the communication protocol. In this thesis, a total of three CODESYS-compatible communication protocols are investigated. It is possible to freely combine them with either of the investigated communication media described above. The three communication protocols are:

- **Modbus TCP:** Following the standards of the Modbus communication protocol, the vPLC is configured as the client and the AIO device is configured as the server of the communication. Packages are sent in both directions between the devices but the vPLC, being the client, defines how this communication should be conducted. This means that data is only sent from the AIO device as a result of a preceding request for it from the vPLC.

- **CODESYS Network Variables:** When utilising the CODESYS Network Variables protocol, both the vPLC and the AIO device are configured to be senders and receivers, which enables mutual sharing of data. The sending side of the channels defines the characteristics of the communication. As a result, the AIO device is responsible for the communication channel where it sends data to the vPLC and the vPLC is responsible for the channel where it sends data to the AIO device.
- **CODESYS OPC UA PubSub:** Similarly to the CODESYS Network Variables protocol, the CODESYS OPC UA PubSub protocol utilises two separate communication channels to send data between the vPLC and the AIO device. The sending side is controlling the communication, which is defined as a separate sub-task in the code. To ensure continuous transmission and reception of data, cyclic calls to the channels are used to trigger communication. Since it is not desired for the communication task to reduce the performance of the ball position control, the communication task is given a lower priority than the control task in CODESYS.

3.2 Time-Related Measurements

3.2.1 Latency

The control-loop latency, as described in section 2.1.2, will simply be referred to as the latency in the subsequent parts of this report.

To achieve desired control performance, the latency of the system has to be measured. This is accomplished by using the system clock of the AIO device. With each ball and beam position sent from the AIO device, a timestamp with the value of the AIO's current system clock value is sent. When a set of ball and beam position data is used in the vPLC to calculate a new control signal, the timestamp that came with the two measurements is retained. This timestamp is then sent back to the AIO device along with the newly calculated control signal. The roundtrip latency can then be calculated by comparison of the received timestamp and the current value of the AIO's system clock. The proposed method to measure the latency consequently includes both communication and computation in one non-separable measure. In figure 3.4, the latency is conceptually defined as the blue arrow that travels from the AIO device, to the vPLC and back.

3.2.2 Time Between Data Packages

Another time-related variable measured in this project is the frequency at which data packages arrives to the vPLC with new measurement signals. A conceptual sketch of this variable is illustrated by the red arrow in figure 3.4. All data packages include their individual timestamps used for the latency calculations described in the previous section. These stamps can also be utilised in the calculations of the elapsed time between data package arrivals. A simple comparison between the timestamp for the newly arrived data package and the timestamp for the previous data package results in satisfactory calculations of the difference variable $diff$ according to

$$diff = timestamp_k - timestamp_{k-1} \tag{3.1}$$

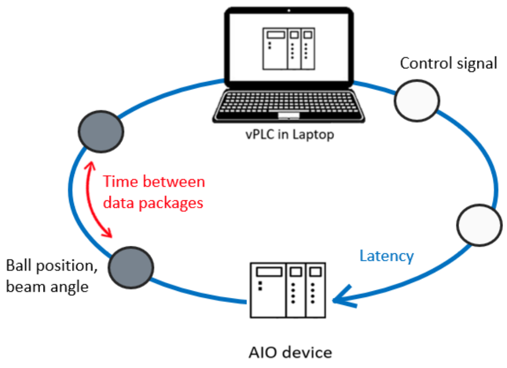


Figure 3.4: Conceptual image of the latency and the time between data packages

3.3 Process Modelling

To succeed with the control of the ball's position on the beam, the two underlying processes of the ball-and-beam setup must be identified. An open-loop representation of them can be seen in figure 3.5 where $G_{sm}(s)$ is the transfer function of the servo motor and $G_{bb}(s)$ is the transfer function of the ball position. Moreover, $U(s)$ is the control signal of the ball-and-beam process, $Y_{ma}(s)$ is the measurement signal of the servo motor angle and $Y_{bp}(s)$ is the measurement signal of the ball's position on the beam.

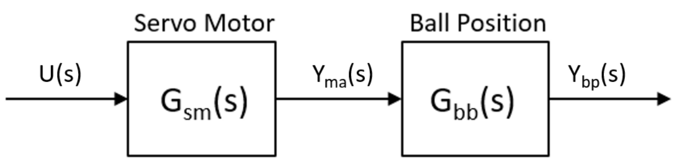


Figure 3.5: An open-loop block scheme of the two physical processes in the ball-and-beam setup

The two processes can be decoupled and treated as sub-plants of the system. Each of them has one measurable value, the angle of the servo motor and the position of the ball, which are both read by the AIO device. The subsequent parts of this section will be devoted to the identification of the two sub-processes' transfer functions, $G_{sm}(s)$ and $G_{bb}(s)$, based on documentation for them provided by the manufacturer [24] and ideas outlined in an earlier thesis work [9].

3.3.1 Servo Motor

The sub-plant representing the servo motor consists of a measurement variable for the motor angle and a manipulated variable for the voltage signal to control the beam. By utilising first-order principles and combining a set of mechanical and electrical equations, the transfer function that describes the servo motor can be expressed as

$$G_{sm} = \frac{K_{sm}}{s(\tau s + 1)} \tag{3.2}$$

where τ represents the time constant of the system and its value is given as 0.0248 s by the manufacturer.

Due to the nature of the ball-and-beam setup, the servo motor can not use the full range of the disc it is mounted on due to a smaller disc mounted in front of it. The system gain, K_{sm}, therefore needs to be scaled from the given manufacturer value. This was done by estimating the obstructed interval range, which can be seen in figure 3.6. Since every angle of the beam can be achieved from two motor angle orientations, only half of the disc was considered. Additionally, taking into account that the voltage signal is converted to a percentage in the chosen programming environment, the final system gain is estimated to $K_{sm} = 5.90\ \frac{\%_{rad/s}}{\%_V}$.

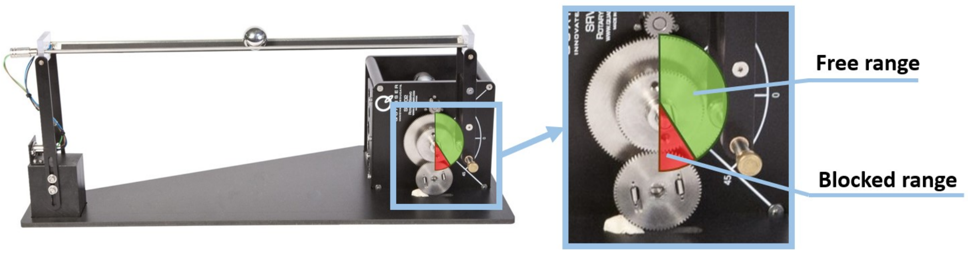


Figure 3.6: An overview of the operational motor angles

3.3.2 Ball Position

The second sub-plant relates to the ball's position on the beam. Similarly to the servo motor sub-plant, it consists of a manipulated variable and a measurement value. The manipulated variable is the output from the servo motor sub-plant and the measurement variable is the ball's position. The transfer function $G_{bb}(s)$ is, hence, describing the relation between the motor angle and the ball position.

From Newton's Law of Motion, a balance of the forces acting on the ball can be obtained. Neglecting friction and using mathematical and mechanical relations gives the transfer function for small motor angles

$$G_{bb} = \frac{K_{bb}}{s^2} \tag{3.3}$$

The constant K_{bb} is the gain of the sub-plant, which has to be found in order to fully define the transfer function. By conducting a physical experiment where the beam angle was controlled and the ball position was measured, K_{bb} was experimentally obtained according to strategies explained further in [9]. Since the tuning of the motor angle was already completed according to the previous section of the thesis, it was now possible to control it. To collect the measurement data of the ball position, the motor angle was controlled to be $\pm10\%$ of its maximum value in a series of steps. The result of this can be seen in figure 3.7.

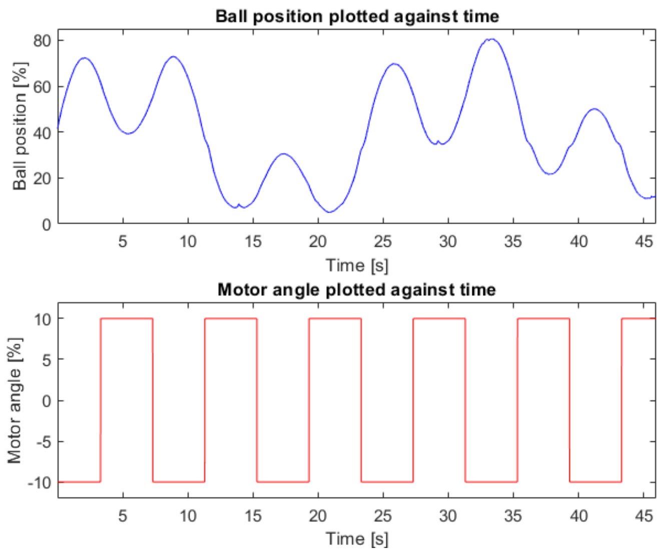
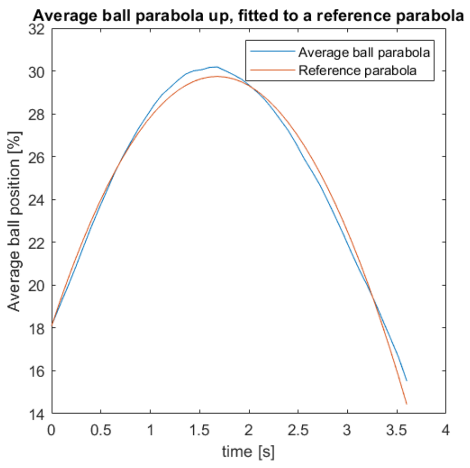
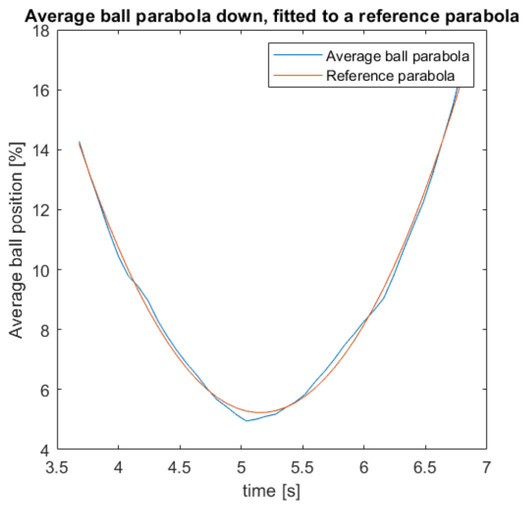


Figure 3.7: A plot of the K_{bb} calibration experiment showing the motor angle and the corresponding ball movements

From the data obtained in this test, three ball position parabolas going upwards and three parabolas going downwards were selected. These were all fitted to a reference parabola using the least-squares method and the gain that it required was calculated. The average value of the required gain for the upward and downward parabolas then became $\overline{K}_{bb,up} = 1.0037\frac{\%y_{bp}}{\%y_{ma}}$, $\overline{K}_{bb,down} = 1.5484\frac{\%y_{bp}}{\%y_{ma}}$. The average ball position parabolas and their corresponding reference parabolas can be seen in figure 3.8.



(a) The average ball parabola for negative motor angles and the corresponding reference parabola



(b) The average ball parabola for positive motor angles and the corresponding reference parabola

Figure 3.8: The ball parabolas for estimation of the gain K_{bb}

The ball moves continuously up and down on the beam when the process is running. This makes it undesirable to use two different values of K_{bb} based on the current angle of the beam since it can create inconsistencies in the control signal. To resolve this, it was decided that the mean of the two values of K_{bb} should be used. This was calculated according to

$$\overline{\overline{K}}_{bb} = \frac{\overline{K}_{bb,up} + \overline{K}_{bb,down}}{2} = \frac{1.0037 + 1.5484}{2} = 1.276 \frac{\%y_{bp}}{\%y_{ma}} \tag{3.4}$$

The mean value, $\overline{\overline{K}}_{bb}$, is used as the value for K_{bb}. With the gain constant defined, the transfer function (3.3) is fully known. From the block diagram in figure 3.5, the full transfer function of the two sub-plants is calculated according to

$$G_{processes} = G_{sm}G_{bb} = \frac{K_{sm}K_{bb}}{s^3(\tau s + 1)} \tag{3.5}$$

3.4 Controller Design

In order to control the target process described in the previous section, a controller structure needs to be chosen. Given the fact that the system signals consist of two measurement variables and one manipulated variable, a cascaded control approach is a suitable choice. The foundation of cascaded control is de facto that there exists an intermediate variable that can be measured and some actuation point that can be used to control it. This is well represented by the motor angle and its torque. In addition, cascaded control brings the benefit of disturbance rejection in the inner loop to prevent degraded performance of the outer loop. Another benefit entailed by this controller structure is the reduction of nonlinear effects. However, the complexity introduced by adding the second control-loop complicates the controller tuning. To guarantee stability, the inner loop should be faster than the outer loop [31].

Figure 3.9 illustrates the cascaded control scheme adapted to the target process of this project. The ball position setpoint, $Y_{ref,bp}$, goes into the outer controller, C_{out}. The output from this controller then acts as the motor angle setpoint, $Y_{ref,ma}$, for the inner controller, C_{in}, which consecutively produces the control signal, $U(s)$, that is sent to the plant. The measured process values for the motor angle and ball position, Y_{ma} and Y_{bp}, are then fed back into their own loops respectively.

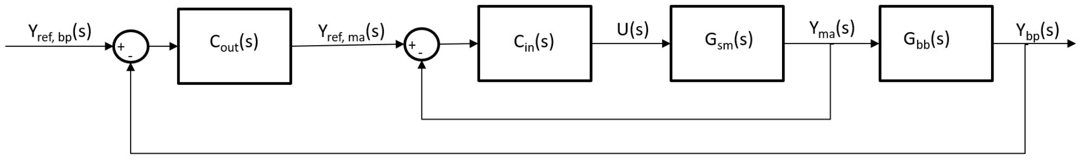


Figure 3.9: Cascaded control block diagram for the ball-and-beam process

After choosing the control structure to use, the cascaded controllers need to be designed. The most widely used controller in industrial settings is the PID controller [34]. Because of this, the outer controller was chosen to be of that type. Due to the simpler dynamics of the inner process, a P controller was deemed to be sufficient for the inner controller. The control parameters are tuned based on the framework presented in section 2.5. The approach includes the latency in the controller synthesis, which facilitates easy parameter adjustment if said variable changes.

3.4.1 Inner Controller

To commence the design of the inner controller, a modification of the transfer function for the servo motor, (3.2), is required. It is re-written as

$$G_{sm}(s) = \frac{K_{sm}e^{-sL}}{s(\tau s + 1)} \tag{3.6}$$

where L is the system delay, in this thesis referred to as the latency. The IMC method requires e^{-sL} to be approximated, which can be achieved with various methods. Since the aim for the inner controller is a plain P controller, a first-order Taylor series approximation

$$e^{-sL} \approx 1 - sL + ... \tag{3.7}$$

is satisfactory and works well with the IMC framework. Combining (3.7) with the servo motor model leads to

$$G_{sm,app}(s) = \frac{K_{sm}(1 - sL)}{s(\tau s + 1)} \tag{3.8}$$

Since the aim is to design a proportional controller for the inner loop, $G_{sm,app}$ is reduced to a first-order model by finding a reduced polynomial for the denominator

$$\hat{A}(s) = \frac{1}{2}(\hat{A}_1(s) + \hat{A}_2(s)) \tag{3.9}$$

The requirements of (3.9) is that the components, $\hat{A}_1(s)$ and $\hat{A}_2(s)$, should preserve the slowest pole and the coefficient of the lowest order of (3.8) [34].

Since this corresponds to s for both requirements, the reduced model for the servo motor can be described by

$$G_{sm,red}(s) = \frac{K_{sm}(1 - sL)}{s} \tag{3.10}$$

As previously discussed, the controller is given by (2.6). Equation (3.10) is therefore split into the two expressions $G_{sm_+}(s)$ and $G_{sm_-}(s)$ according to

$$G_{sm,red}(s) = G_{sm_+}(s)G_{sm_-}(s) \tag{3.11}$$

The latency, here represented by the zero located in the right-half plane, is placed in $G_{sm_+}(s)$. Consequently, $G_{sm_-}(s)$ only contains the (marginally) stable parts.

$$\begin{aligned} G_{sm_-}(s) &= \frac{K_{sm}}{s} \\ G_{sm_+}(s) &= 1 - sL \end{aligned} \tag{3.12}$$

Lastly, the filter $F(s)$ is chosen as

$$F(s) = \frac{1}{\lambda s + 1} \tag{3.13}$$

Inserting (3.12) and (3.13) into (2.6) generates the inner proportional controller

$$C_{in}(s) = \frac{1}{K_{sm}(\lambda + L)} \tag{3.14}$$

where λ is a pole placement parameter that can be altered by the designer.

3.4.2 Outer Controller

In order to design the outer controller, a similar approach is followed. Firstly, adding the latency into the process transfer function (3.3) gives

$$G_{bb}(s) = \frac{K_{bb}e^{-sL}}{s^2} \tag{3.15}$$

For this sub-plant, the latency can be approximated by a first-order Padé approximation [38] according to

$$e^{-sL} \approx \frac{\frac{-L}{2}s+1}{\frac{L}{2}s+1} \tag{3.16}$$

which leads to the third-order system

$$G_{bb}(s) = \frac{K_{bb}}{s^2}\frac{(\frac{-L}{2}s+1)}{(\frac{L}{2}s+1)} \tag{3.17}$$

To arrive at a PID controller, the model should be of second order and (3.17) is therefore reduced in the same way as described above. Following the same reasoning regarding the reduced denominator polynomial, the reduced model $G_{bb,red}(s)$ is found to be

$$G_{bb,red}(s) = \frac{K_{bb}(\frac{-L}{2}s+1)}{s^2} \tag{3.18}$$

Continuing to follow the same approach as for the inner controller, (3.18) can be divided into

$$\begin{aligned} G_{bb_-}(s) &= \frac{K_{bb}}{s^2} \\ G_{bb_+}(s) &= \frac{-L}{2}s+1 \end{aligned} \tag{3.19}$$

The double integrator in the denominator of the reduced model, (3.18), has to be canceled out by the outer controller to avoid marginally stable process behaviors. In order to aid the cancelling, the filter is split into a numerator and a denominator

$$F(s) = \frac{F_n(s)}{F_d(s)} \tag{3.20}$$

Based on the controller presented in (2.6), the denominator of the outer controller can be written as

$$1 - \frac{F_n(s)}{F_d(s)}G_{bb_+}(s) \tag{3.21}$$

The term that the denominator needs to cancel out is s^2, which implies that (3.21) shall at least contain this term in its numerator. If an additional s is included, the controller will be able to achieve the beneficial behavior of integral action. Thus, the filter can for example be chosen as a low-pass filter of the fourth order, which then requires the numerator of (3.21) to be $s^3(t_1s+t_0)$. Multiplying (3.21) with $F_d(s)$ yields the Diophantine [1] equation

$$F_d(s) - F_n(s)\left(\frac{-L}{2}s+1\right) = s^3(t_1s+t_0) \tag{3.22}$$

The equation defining the poles of the filter denominator is chosen as

$$F_d(s) = (\lambda_1 s+1)(\lambda_2 s+1)(\tau_0^2 s^2 + 2\zeta\tau_0 s+1) \tag{3.23}$$

The selected design strategy implies that the poles of (3.23) can be placed by the designer to achieve desired control. The parameters for the two real poles are λ_1 and λ_2 and the two complex-conjugated poles are placed according to τ_0 and ζ. Assuming that a second-order polynomial is sufficient for the filter nominator, $F_n(s)$, it becomes

$$F_n(s) = f_2 s^2 + f_1 s + 1 \tag{3.24}$$

where f_1 and f_2 can be altered to arrive at the desired controller. Combining the expressions for the filter nominator, (3.24), and denominator, (3.23), with (3.22) yields

$$(\lambda_1 s + 1)(\lambda_2 s + 1)(\tau^2 s^2 + 2\zeta\tau_0 s + 1) - (f_2 s^2 + f_1 s + 1)(\frac{-L}{2}s + 1) = s^3(t_1 s + t_0) \tag{3.25}$$

By identifying each order of s separately in (3.25), the system of equations for f_1, f_2, t_0 and t_1 results in

$$\begin{cases} f_1^* = \lambda_1\lambda_2 + 2\zeta\tau_0 + \frac{L}{2} \\ f_2^* = \tau_0^2 + 2(\lambda_1 + \lambda_2)\zeta\tau_0 + \lambda_1\lambda_2 + f_1^*\frac{L}{2} \\ t_0^* = 2\lambda_1\lambda_2\zeta\tau_0 + (\lambda_1 + \lambda_2)\tau_0^2 + f_2^*\frac{L}{2} \\ t_1^* = \lambda_1\lambda_2\tau_0^2 \end{cases} \tag{3.26}$$

The system has a unique solution for each set of the selectable parameters $\lambda_1, \lambda_2, \tau_0$ and ζ, which confirms that a second order $F_n(s)$ meets the requirements.

Inserting (3.19), (3.20) into (2.6) results in the outer PID controller, with an included filter

$$C_{out}(s) = \frac{1}{K_{bb}} \frac{f_2^* s^2 + f_1^* s + 1}{s(t_1^* s + t_0^*)} \tag{3.27}$$

In order to find the individual P-, I- and D-parts from this expression, the filter needs to be separated from the full equation. By selecting the filter constant β, to be $\frac{t_1^*}{t_0^*}$, a low-pass filter dependent on the roundtrip latency L can be described as

$$F_{out}(s) = \frac{1}{\beta s + 1} = \frac{1}{\frac{t_1^*}{t_0^*} s + 1} \tag{3.28}$$

Separating this filter from (3.27) results in

$$C_{out}(s) = \frac{f_2^* s^2 + f_1^* s + 1}{K_{bb} t_0^* s} \frac{1}{\frac{t_1^*}{t_0^*} s + 1} \tag{3.29}$$

A comparison between (2.8) and the part of (3.29) that does not contain $F_{out}(s)$ results in the individual PID parameters

$$\begin{cases} K_{p,out} = \frac{f_1^*}{K_{bb} t_0^*} \\ K_{i,out} = \frac{1}{K_{bb} t_0^*} \\ K_{d,out} = \frac{f_2^*}{K_{bb} t_0^*} \end{cases} \tag{3.30}$$

In line with the design ambition, the two cascaded controllers are then dependent on the full control-loop latency, L, and their parameters can therefore be tuned according to the measured latency of the different test setups.

3.5 Discretisation and Implementation

3.5.1 Controllers

When evaluating time-based control in this thesis, the above derived controllers are implemented according to the discretisation methods presented in the background section 2.6.1 with one exception. When the I-part of the time- and event-driven PID controllers are calculated, the current error is used instead of the old error. The decision to do this is based on a desire to use the most recent measurement data available in order to enhance the accuracy of the calculations. Additionally, the event-based controller differs from the proposed PIDPlus alterations described in section 2.7.2. The cycle time of the vPLC is used as the sampling interval in the event-based controller calculations instead of the actual time that has passed since the previous communication was conducted.

3.5.2 Filters

In similarity with the controllers, the two filters $F_{out}(s)$ and $F_{ref}(s)$ in the developed control application require discretisation before they can be implemented in the programming environment. The continuous-time description of the filter that is separated from the outer PID controller found in (3.29) is presented in (3.28). It filters the measurement signal of the ball's position on the beam, Y_{bp}. A Tustin approximation is used to discretise the continuous filter. Hence, s is re-written as

$$s = \frac{2}{T_s}\frac{(1-q^{-1})}{(1+q^{-1})} \tag{3.31}$$

Inserting this in (3.28) yields

$$F_{out,k} = \frac{1}{\frac{2\beta}{T_s}\frac{(1-q^{-1})}{(1+q^{-1})}+1}y_{bp,k} = \frac{T_s(1+q^{-1})}{2\beta(1-q^{-1})+T_s(1+q^{-1})}y_{bp,k} \tag{3.32}$$

Further simplifications lead to the final, discrete filter

$$(2\beta + T_s - (2\beta - T_s)q^{-1})F_{out,k} = T_s(1+q^{-1})y_{bp,k}$$

$$\implies F_{out,k} = \frac{(2\beta - T_s)F_{out,k-1} + T_s(y_{bp,k} + y_{bp,k-1})}{2\beta + T_s} \tag{3.33}$$

The sampling interval, T_s, used by this filter will be decided based on the control algorithm that is implemented.

The reference signal for the ball position, $Y_{ref,bp}$, comes from a square-wave generator implemented in the vPLC. The reference signal is passed through a low-pass filter to facilitate smooth reactions to the rising and falling edges of the wave. The filter is of the same type as the ball position filter. Thus, it is described in continuous time as

$$F_{ref} = \frac{1}{\tau_{ref}s + 1} \tag{3.34}$$

where τ_{ref} is the time constant of the filter. Based on physical experiments and evaluations it is defined to be 0.3 s. The filter is discretised in the same way as the filter for the ball position and the resulting discrete representation is

$$F_{ref,k} = \frac{(2\tau_{ref} - T_{s,ref})F_{ref,k-1} + T_{s,ref}(y_{ref,bp,k} + y_{ref,bp,k-1})}{2\tau_{ref} + T_{s,ref}} \tag{3.35}$$

Since the reference signal is generated within the vPLC, it is processed once every cycle. Thus, the sampling interval of this filter, $T_{s,ref}$, is always set to be the cycle time of the vPLC application.

3.5.3 The Complete Outer Controller

A schematic figure of the full outer PID controller implementation strategy can be seen in figure 3.10. As presented in the figure, the filter F_{ref} is used to smoothen the reference signal. The ball position signal, Y_{bp}, is filtered by F_{out}. The P- and I-parts of the control signal are calculated according to the descriptions in section 2.6.1. The difference between the two filtered signals, also referred to as the error, is used in the proportional and integral calculations. As described in section 2.6.1, the D-part acts on the process value and not the error. Hence, it has in this schematic figure been placed in a separate block, not connected to the reference signal, $Y_{ref,bp}$. The control signal from the outer PID controller serves as the reference signal to the inner loop, $Y_{ref,ma}$ and is calculated by combining the P-, I- and D-parts.

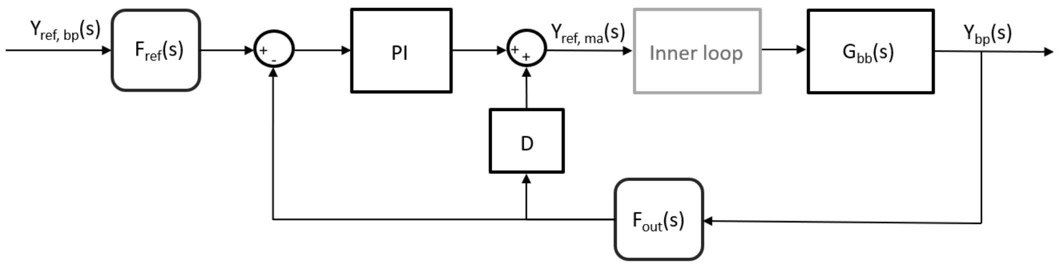


Figure 3.10: A block scheme presenting the complete implementation structure of the outer PID controller

3.6 Controller Tuning

3.6.1 Outer Controller

As described in section 3.4, the continuous-time outer controller is dependent on five design variables. The two variables λ_1 and λ_2 define the real poles, and the two variables τ_0 and ζ define the two complex-conjugated poles, all according to (3.23). The fifth and final design variable is the latency L.

The pole placement is performed through a combination of theory and practical tests. To ensure stability, all of the continuous-time poles are placed in the left-half of the complex plane. The real poles are placed closest to the imaginary axis based on a desire for them to be dominating. The two complex-conjugated poles are placed close to the real axis, preventing them from contributing to additional oscillations in the system. Having these poles furthest away from the imaginary axis means that they are the fastest poles of the outer loop. The final placement of the four poles that gives satisfactory control can be seen in figure 3.11. This yields the design values $\lambda_1 = 5.81$, $\lambda_2 = 4.61$, $\tau_0 = 0.49$ and $\zeta = 1$.

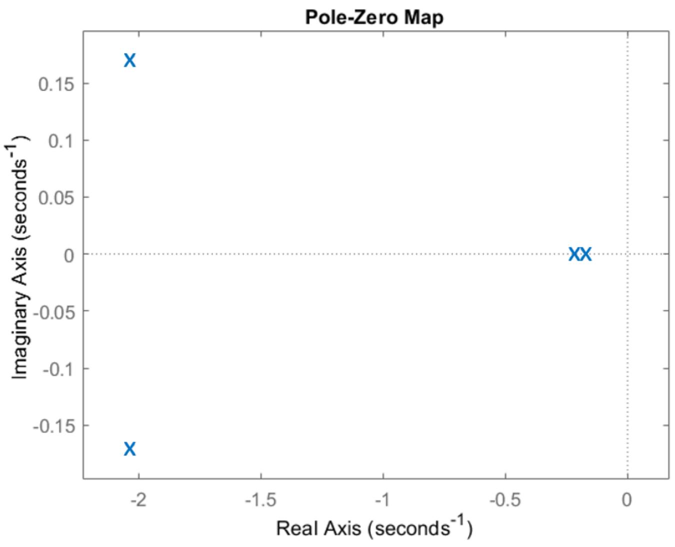


Figure 3.11: The placement of the continuous-time outer controller poles in the complex plane

The last required design variable is the latency, L. The outer controller is tuned according to the actual latency that is measured on the specific communication link currently in use. In most cases the latency is a fluctuating variable, thus, a careful tuning approach was selected. This means that the controller is usually tuned for a latency slightly above the average of what it is measured to be.

With all five design variables defined, the final PID parameters for the outer controller can be found by combining (3.26) and (3.30).

3.6.2 Inner Controller

The inner controller, defined in (3.14), is tuned according to the general guideline which states that an inner control-loop should be faster than the outer control-loop in cascaded structures. This is fulfilled by the choice of $\lambda = 0.051$, which places the inner controller pole on the real axis at $\frac{-1}{\lambda} = -19.77$. Based on (3.14), the calculation of $K_{p,in}$ is then conducted according to

$$K_{p,in} = \frac{1}{K_{sm}(\lambda + L)} \tag{3.36}$$

which now only depends on the latency L. The inner controller will be given the same latency as the outer controller when tuning is performed. This means that the latency is chosen in the same way as described above in section 3.6.1.

3.7 Test Outline

To facilitate the investigation of the research questions presented in section 1.2, a test outline needs to be established. Section 3.1.1 of this chapter presents various communication media and protocols, where several setups are possible. By identifying prosperous combinations that aids quantification of the research questions, a set of hypotheses were established to base the following tests upon.

3.7.1 Hypotheses

0. Reference tests

The tests labeled as references were performed in order to have a baseline to compare subsequent tests with. The reference tests were all executed with a wired Ethernet connection. The control algorithm that was used does not consider potential delays between received packages, implicating that it can run the control loop based on the same measurement values more than once. The sampling interval, T_s, which is used for the ball position filter and the I- and D-parts of the outer controller, is set to the vPLC's cycle time.

1. Protocol matters

This hypothesis aims at investigating the impact that the communication protocol might have on wireless control. The same control algorithm as for the reference tests was used with the three different protocols, namely Modbus TCP, OPC UA PubSub and Network Variables. This and subsequent hypotheses were investigated in a wireless manner over either WiFi or 5G.

2. Control implementation matters

As previously indicated, it was suspected that taking the time between data packages into account for the control calculations might improve the performance. Suggested alterations to the algorithm, based on this suspicion, are:

- Using the time between data packages described in section 3.2.2 instead of the application cycle time as the sampling interval T_s.
- Using the timestamps accompanying every measurement to verify that the incoming measurements are more recent than the ones previously used. This also serves as a natural way to handle variables that could overflow due to size restrictions.
- If no new measurements have arrived since the last application cycle, the controllers should not be run in order to prevent incorrect values of the I- and D-parts. As a result, the control signal has a constant value until the next set of new measurements arrives to the vPLC.

3. Event-based control saves communication

Based on the philosophy presented in section 2.7.1, it is unnecessary to run the controller unless the setpoint for the process changes or disturbances occurs. This hypothesis aims at investigating how much communication bandwidth an event-based controller can save. Seeing that UDP based protocols represent a promising candidate for implementation of event-based control, only Network Variables was realised due to feasible employment and its large similarities to OPC UA PubSub.

The event thresholds were manually tuned for each of the transmitted signals; the control signal, the ball position measurement and the beam angle measurement. In addition, a keep-alive-trigger was implemented with the value of 10 s. The purpose of this trigger is to force the sensor to occasionally communicate its latest measurements even in stationary states. This ensures the client that the connection is still alive even if no measurements have been sent for a long period of time. A pseudocode example of how the thresholds and the keep-alive-trigger are managed in the sensor can be seen below.

```
 timeDelta = time since values were last sent to the vPLC;

IF(timeDelta > keep-alive-trigger){
    sendValues = true;
}

IF(currentSetpoint != oldSetpoint){
    setpointChanged = true;
}

IF(deviation from last sent value >= threshold OR sendValues OR setpointChanged){
    send new values to the vPLC;
}
```

4. Event thresholds can be adapted at the sensor

Further improving and generalising the previous hypothesis, it was investigated whether the event threshold for each of the measurements can be adapted in the sensors themselves. However, two key limitations were established to make the threshold tuning algorithm as generic as possible. The background for this decision is a desire to create an algorithm that will be suitable for a broader range of applications than just the ball-and-beam process and the specific system architecture used in this project. The two limitations are:

- **Separated sensors and actuators:** In the examined setups, the AIO device serves as both sensor and actuator since it provides the controller with the measurement signals but also forwards the control signal to the process. For generalisation purposes, an imaginary system architecture where it is assumed that the ball and beam sensors are separated from each other and from the actuator is adopted. This entails two sensors that are unaware of the measurement signal from the other sensor and the control signal that is sent to the actuator.
- **One-way communication:** The second limitation that is established with the aim to create a generic threshold tuning algorithm restricts the communication between controller and sensor. It is assumed that the sensor only acts as a measurement device, providing the controller with data. Communication is, hence, imagined to only travel from the sensor to the controller and not back.

Considering the two limitations described, the only data that is left to use for a threshold tuning algorithm performed in the two sensors are their own individual measurements. Additionally, the ball and the beam sensors are unaware of the reference values for the respective measurement.

The proposed tuning approach is based on the idea that the user can configure the desired amount of communication to save. The threshold tuning consists of running the system time-driven for a sufficiently long period of time in order to capture the full behavior of the controlled process and its surroundings. For the process used in this project, 2 minutes are chosen since it captures enough steps to reflect the usual states. During this period, the sensor stores the difference between every current and previous sample. The stored values can be seen as derivatives. Once the configured time has elapsed, the derivatives are sorted in ascending order in a vector which can be divided in portions according to the indicated desired saved amount of communication. The neighboring derivative value to the right of the split is set as the event threshold.

This threshold selection can be illustrated as in figure 3.12, where n corresponds to a signal in the system. The simple example consists of 10 derivative values stored in ascending order in a vector, meaning that $\Delta_{10} \geq \Delta_9 \geq \Delta_8$... The user has configured a 60% communication saving. Thus, the value stored at the 7:th vector position, highlighted with a bold red border in the figure, will be the threshold since it is the first derivative that is larger than 60% of the stored derivatives. Consequently, when the threshold has been established, a newly sampled measurement will be compared to the most recently transmitted measurement. If the difference between them is less than the threshold value, the new sample will not be communicated. The threshold therefore contributes to communication savings, represented by the shaded six first elements in the vector in the figure.

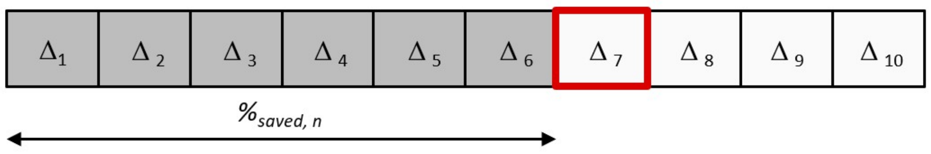


Figure 3.12: An illustration of the event threshold selection

The method acts as a high-pass filter for the derivatives. In cases where the combination of the configured saved communication percentage and the recorded measurements results in a zero threshold, the minimum non-zero derivative value is chosen as the threshold to secure that the tuned system will indeed be event-based.

5. The method is generic for other wireless media

The final hypothesis was derived with the purpose of proving genericness, demonstrating that the preceding results are independent of the type of wireless link that is used. The event-driven implementation, including adapting thresholds at the sensors presented in the previous hypothesis, was run using a 5G network.

3.7.2 General Test Definition

The hypotheses described above are summarised in table 3.1. All tests were run for 6 minutes with controller latency parameters individually tuned to achieve best performance for each test case. Both the vPLC and the AIO device were configured to have a cycle time of 4 ms throughout all tests. It should be noted that these cycles are running asynchronously, meaning that there are no guarantees that they will align at every time instance. They will rather phase in and out of sync, which consequently affects the experienced latency and time between data packages. The reference signal for the ball position, $Y_{ref,bp}(s)$ in figure 3.9, is a square wave that has a period of 50 ms and an amplitude of 30% of the beam length. The performance will be evaluated based on the mean behaviour of the up and down steps.

Table 3.1: An overview of the conducted tests

	Time-based			Event-based
	Modbus TCP	OPC UA PubSub	Network Variables	Network Variables
Ethernet	0.	0.	0.	
WiFi	1. 2.	1.	1. 2.	3. 4.
5G				5.

3.8 Performance Evaluation

3.8.1 Control

To evaluate the performance of the different control approaches for the process, a series of step response tests were conducted. Data from these tests were used to generate plots of the steps. To facilitate and quantify the comparison of different tests from a control perspective, a set of four key performance indicators (KPI:s) were also defined.

- **Rise Time** t_r [ms]: This KPI relates to the time domain characteristics of the controller and is defined as the time required for the ball to go from 10% to 90% of its new setpoint value on the beam [23]. If the ball never reaches 90% of the new setpoint, t_r is given the value of the full step length. In this thesis, t_r is calculated as a mean value from a series of steps.

- **Settling Time** t_s [ms]: The settling time highlights the time required for the ball to reach within a predefined band of its reference position [23]. Similarly to t_r, this KPI also relates to the time domain characteristics of the process. The band was decided to be $\pm 10\%$ of the step amplitude. With the chosen step-definition this becomes $\pm 3\%$ of the beam length. If the ball never reaches and stays within the band, t_s is given the value of the full step length. The t_s's presented were calculated as mean values from a series of steps.

- **Mean absolute error** MAE: This KPI displays the magnitude of the step response's static error. It is based on calculations of the integral of the absolute error, IAE, which is a commonly used controller KPI, calculated according to the equation below. The error $e(t)$ is defined as the difference between the actual ball position and the desired ball position [35].

$$IAE = \int_0^\infty |e(t)|\, dt \tag{3.37}$$

  Since measurement values of this error are recorded in discrete time, based on the cycle time of the application, IAE is not entirely suitable to use as a KPI for the experiments conducted. To facilitate calculations using discrete values, a sum is used instead of an integral. Shorter cycle times of the application results in more measurement values than longer cycle times. To ensure comparability between tests with different cycle times, the sum mentioned above is divided by the number of measurements, N. These alterations result in the calculation of MAE as

$$MAE = \frac{1}{N}\sum_{i=0}^{N} |e(t)| \tag{3.38}$$

  Equation (3.38) is valid also for the event-based tests since the most recently received ball position is used for the error calculations. During static periods with no communication, the same error is simply collected every cycle until a new ball position is received. As a result, the MAE calculations from the event-based tests are comparable to their time-based counterparts although ball movements below the threshold might not be captured in the event-based MAEs. However, the narrow thresholds used during the tests reduce the effects of such errors.

- **Mean absolute control signal** $|u|$: It is desirable to avoid excessive control action [23], hence, this KPI is utilised to highlight the control action needed to achieve the current process control. The measurement is important to display since two different controllers can appear to perform equally well from a control perspective but one of them might require larger amounts of power. From an energy perspective this is of course negative and might increase the wear and tear of the components in the controlled process. $|u|$ is calculated according to (3.39) where u is the control signal and N is the number of control signal samples.

$$|u| = \frac{1}{N}\sum_{i=0}^{N}|u_i| \tag{3.39}$$

3.8.2 Communication

The performance of the different communication protocols used in the thesis is assessed through the use of two KPI:s connected to the time characteristics of the communication. These are useful when time-based control is evaluated. However, since event-based control strives to reduce the communication on the network by avoiding transmission of unnecessary measurements, these KPI:s are not suitable to use for that control approach. The main reason for this is that it is difficult to determine if longer communication times originate from poor communication performance or simply from the nature of event-based control.

- **Mean Latency** L [ms]: The latencies in the tests are found according to the description in section 3.2.1. The presented L's are mean values of all measured latency values. The latency KPI is calculated according to (3.40), where L is the latency and N is the number of latency measurements.

$$L = \frac{1}{N}\sum_{i=0}^{N}L_i \tag{3.40}$$

- **Mean Time between data packages** $diff$ [ms]: The $diff$ is found according to the description in section 3.2.2. Through measurements of this variable for longer periods of time and calculations of the resulting mean value according to (3.41), the presented $diff$'s were found.

$$diff = \frac{1}{N}\sum_{i=0}^{N}diff_i \tag{3.41}$$

3.8.3 Event-Based

When assessing event-based control, an interesting variable to study is the reduced amount of communication. The retained communication can be computed as the number of transmissions in the event-driven case divided by the number of transmissions that would take place if the system would be controlled in a time-driven manner, according to

$$\%_{saved,n} = 1 - \frac{n_{event}}{n_{time}} \tag{3.42}$$

Since there are several signals sent in the system based on different events, these are separated into three different KPI:s to aid isolated analysis of the ball and beam measurements and the control signal. Additionally, this decision generalises the system such that the sensors could have been physically separated from each other.

- **Saved Control Signal Transmissions** $\%_{saved,Control}$
- **Saved Ball Measurement Transmissions** $\%_{saved,Ball}$
- **Saved Beam Measurement Transmissions** $\%_{saved,Beam}$

4
Results

This chapter presents the results of the tests conducted according to the test plan established in section 3.7. Further descriptions of the communication media and protocols that are mentioned can be found in section 3.1.1. A deeper discussion and analysis of the results presented in this chapter can be found in chapter 5

4.1 Reference Tests

The first tests conducted were the reference tests. As described in section 3.7, these were all conducted with Ethernet as a wired communication medium. The results will be used as a baseline for future comparison to subsequent tests. The latency L, the sampling interval T_s, the PID parameters and the filter variable β that were used for the individual tests can be found in table 4.1.

Table 4.1: Parameters for the reference tests

Parameter\Communication Protocol	Modbus TCP	Network Variables	OPC UA PubSub
L [ms]	24	16	38
T_s [ms]	4	4	4
$K_{p,out}$	0.7481	0.7519	0.7415
$K_{i,out}$	0.02690	0.02710	0.02670
$K_{d,out}$	1.011	1.013	1.007
β	0.2208	0.2220	0.2188
$K_{p,in}$	2.271	2.544	1.912

4.1.1 Modbus TCP

The first reference test was run with Modbus TCP as communication protocol. The step responses, latency and time between data packages that this resulted in can be seen in figure 4.1. It can be noted in this plot that the measurement value does not follow the reference value closely, there is a static error. A possible explanation to this is that the calculations of the integral part of the PID controller is not correct. In chapter 5 this topic will be discussed further.

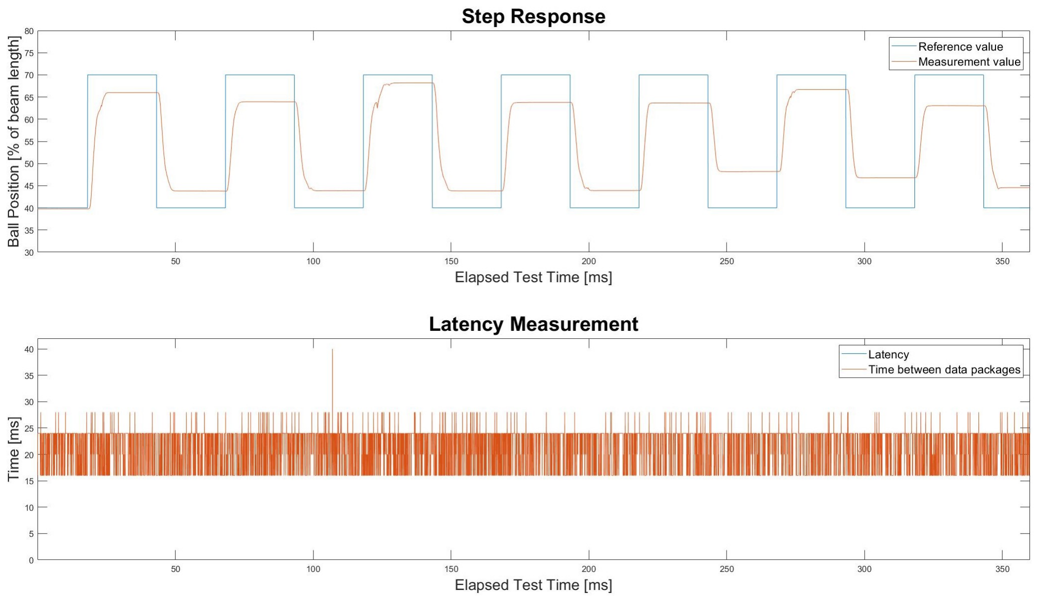


Figure 4.1: The step responses, latency and time between data packages from the reference test over Ethernet using Modbus TCP

4.1.2 Network Variables

The second reference test was run with Network Variables as communication protocol. The step responses, latency and time between data packages that this resulted in can be seen in figure 4.2.

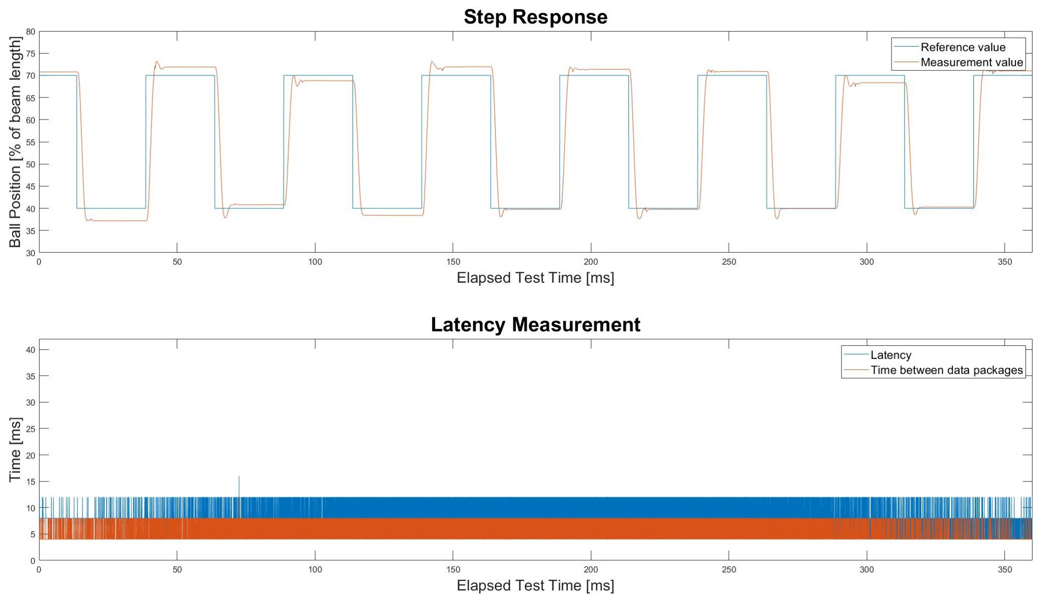


Figure 4.2: The step responses, latency and time between data packages from the reference test over Ethernet using Network Variables

4.1.3 OPC UA PubSub

The final communication protocol that was investigated during the reference testing was OPC UA PubSub. The step responses, latency and time between data packages that this resulted in can be seen in figure 4.3.

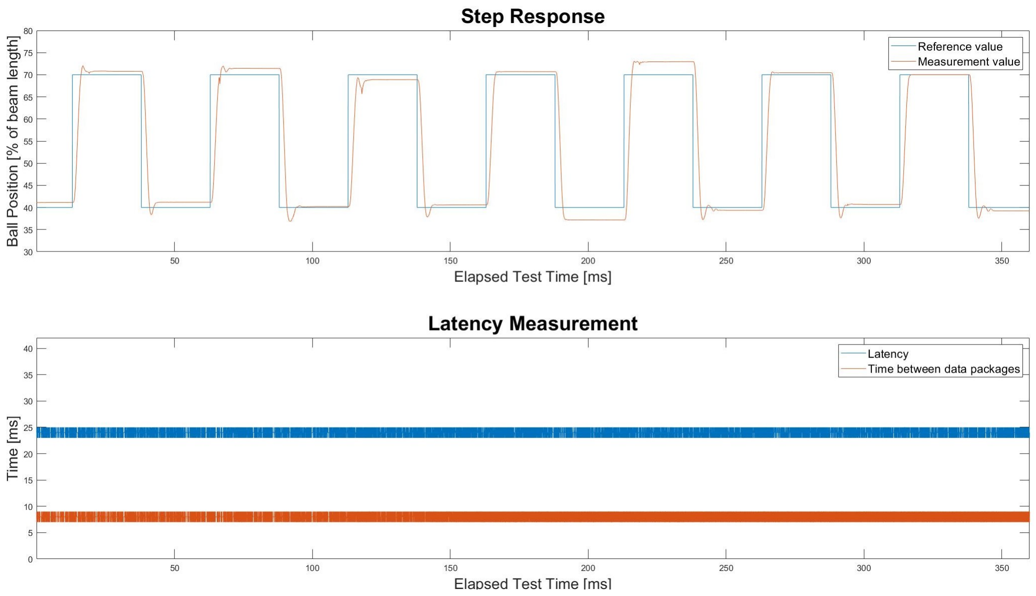


Figure 4.3: The step responses, latency and time between data packages from the reference test over Ethernet using OPC UA PubSub

The KPI:s for the three communication protocols can be seen in table 4.2. Modbus TCP had noticeably longer rise and settling times compared to the other protocols. This was a result of poor performance that led to the specified requirements not being fulfilled for these KPI:s during some of the steps. Another result that followed the poor control performance of Modbus TCP was the significantly higher error KPI. The rise and settling time and error KPI for Network Variables and OPC UA PubSub were similar. The protocol that used the least amount of energy was OPC UA PubSub. The L and $diff$ had the same value for Modbus TCP which was negative for the control performance. The overall fastest protocol was Network Variables, that had both L and $diff$ close to the application cycle time, 4 ms. OPC UA PubSub had a low $diff$ but a L comparable to the Modbus TCP L.

Table 4.2: KPI:s for the reference tests

KPI \ Communication Protocol	Modbus TCP	Network Variables	OPC UA PubSub
Rise Time t_r [s]	23.23	1.667	1.685
Settling Time t_s [s]	23.32	2.910	3.215
MAE	6.972	3.309	3.241
Mean Absolute Control Signal $\|u\|$	388.8	448.3	375.9
Mean Latency L [ms]	21.37	8.746	24.01
Mean Time Between Data Packages $diff$ [ms]	21.37	5.387	7.986

4.2 Protocol Matters

The tests that were run to evaluate the effects of the communication protocol on the control performance used the wireless communication medium WiFi. The same three communication protocols as used in the previous tests were investigated. The latency L, the sampling interval T_s, the PID parameters and the filter variable β that were used for the individual tests can be found in table 4.3.

Table 4.3: Parameters for the tests investigating performance differences between communication protocols used to communicate over WiFi

Parameter\Communication Protocol	Modbus TCP	Network Variables	OPC UA PubSub
L [ms]	28	16	38
T_s [ms]	4	4	4
$K_{p,out}$	0.7462	0.7519	0.7415
$K_{i,out}$	0.02690	0.02710	0.02670
$K_{d,out}$	1.010	1.013	1.007
β	0.2203	0.2220	0.2188
$K_{p,in}$	2.155	2.544	1.912

4.2.1 Modbus TCP

The plot showing the step responses and measured latency of the Modbus TCP protocol can be seen in figure 4.4.

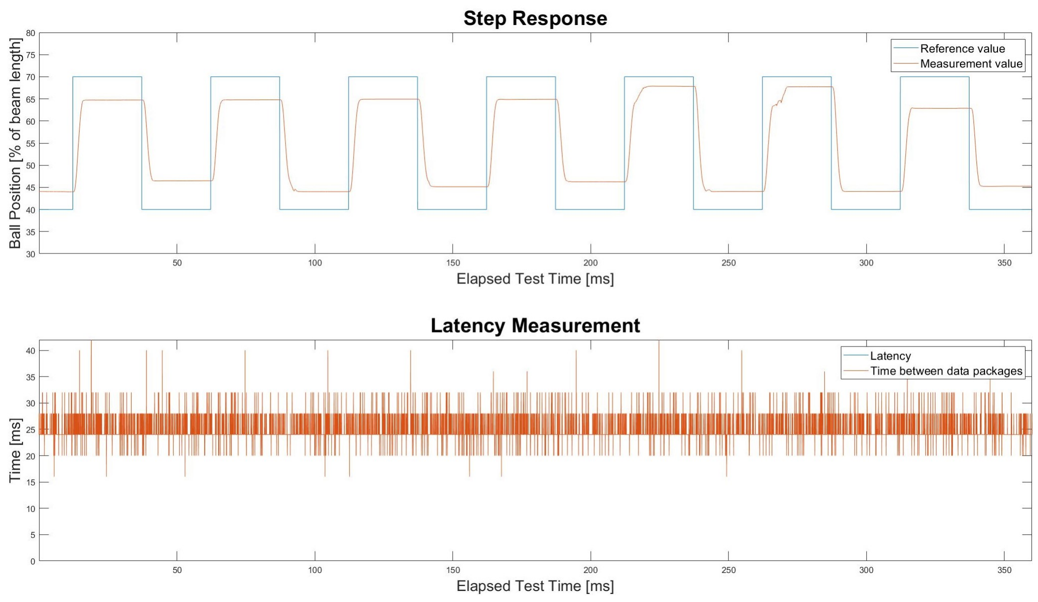


Figure 4.4: The step responses, latency and time between data packages from the test conducted to evaluate protocol dependency using WiFi and Modbus TCP

4.2.2 Network Variables

The plot showing the performance of the Network Variables protocol can be seen in figure 4.5.

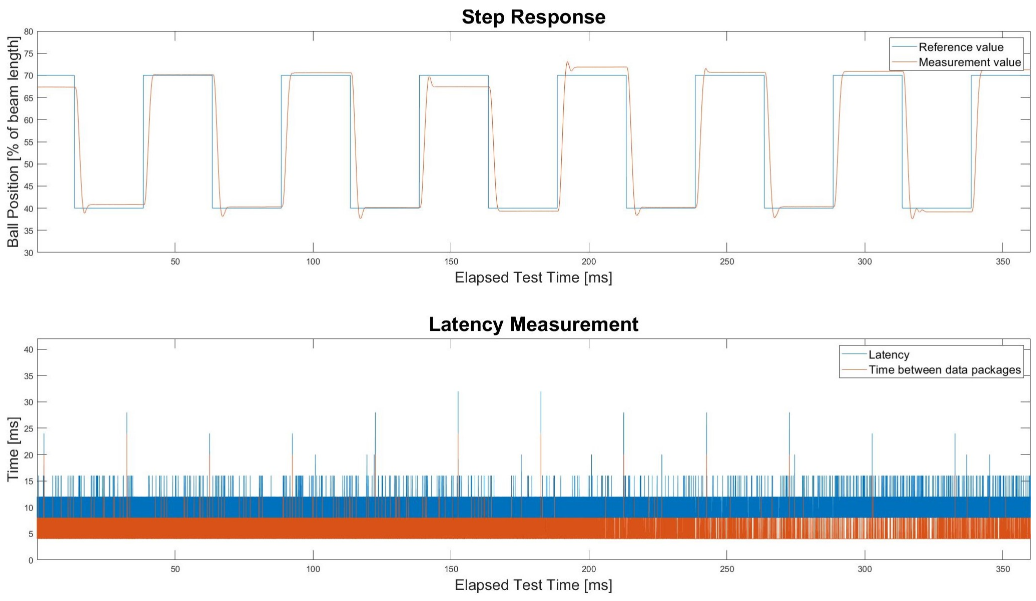


Figure 4.5: The step responses, latency and time between data packages from the test conducted to evaluate protocol dependency using WiFi and Network Variables

4.2.3 OPC UA PubSub

The plot showing the performance of OPC UA PubSub can be seen in figure 4.6.

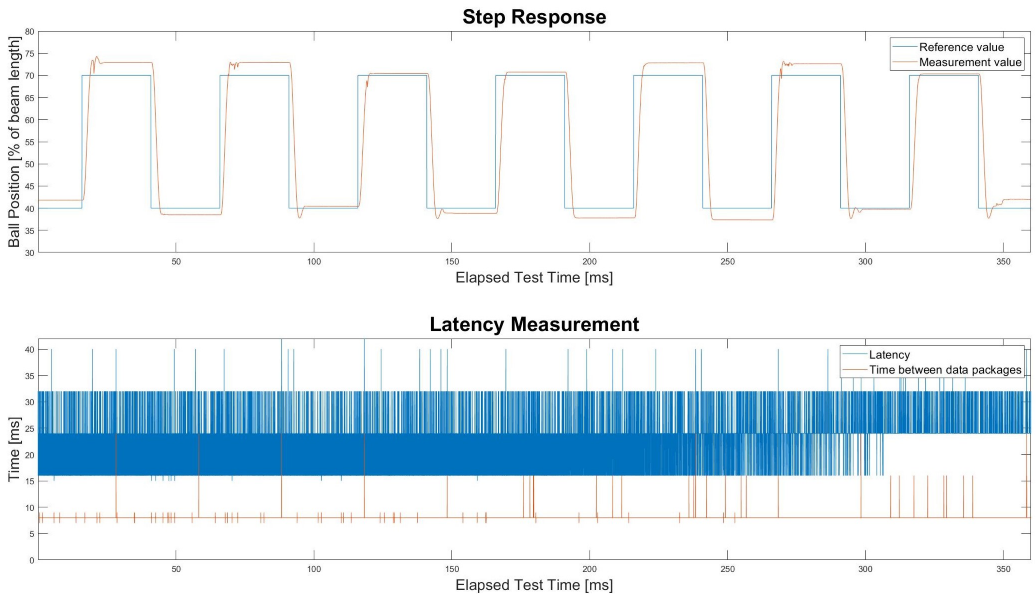


Figure 4.6: The step responses, latency and time between data packages from the test conducted to evaluate protocol dependency using WiFi and OPC UA PubSub

The resulting KPI:s from the tests conducted to evaluate the effects of the communication protocol for wireless control performance can be seen in table 4.4. The observed overall pattern and differences between the protocols closely followed the results of the reference tests. The largest difference between this and the reference test was the energy consumption for OPC UA PubSub, that experienced a large increase when transferred to a wireless network. It is possible that the longer latency was the reason for this since it might create the need for greater control signals in order to achieve the desired control performance.

Table 4.4: KPI:s for the tests conducted to evaluate protocol dependency of the control performance using the communication medium WiFi

KPI \ Communication Protocol	Modbus TCP	Network Variables	OPC UA PubSub
Rise Time t_r [s]	21.97	1.834	1.643
Settling Time t_s [s]	21.97	2.830	3.187
MAE	6.493	2.853	3.814
Mean Absolute Control Signal $\|u\|$	390.3	346.7	459.2
Mean Latency L [ms]	24.75	9.423	22.34
Mean Time Between Data Packages $diff$ [ms]	24.75	4.920	8.007

4.3 Control Implementation Matters

To evaluate the effects of the controller implementation on the control performance, two tests were conducted using a new, more consciously chosen control algorithm. This was designed according to the descriptions in section 3.7.1. The algorithm was tested on the Modbus TCP and Network Variables protocols and the latency L, the sampling interval T_s, the PID parameters and the filter variable β that were used for the tests can be found in table 4.5.

Table 4.5: Parameters for the tests investigating the performance effects of a new control implementation over WiFi

Parameter\Communication Protocol	Modbus TCP	Network Variables
L [ms]	32	16
T_s [ms]	$diff$	$diff$
$K_{p,out}$	0.7443	0.7519
$K_{i,out}$	0.02680	0.02710
$K_{d,out}$	1.009	1.013
β	0.2197	0.2220
$K_{p,in}$	2.051	2.544

4.3.1 Modbus TCP

The test that was run to confirm that control implementation can be of importance when using Modbus TCP for wireless control can be seen in figure 4.7.

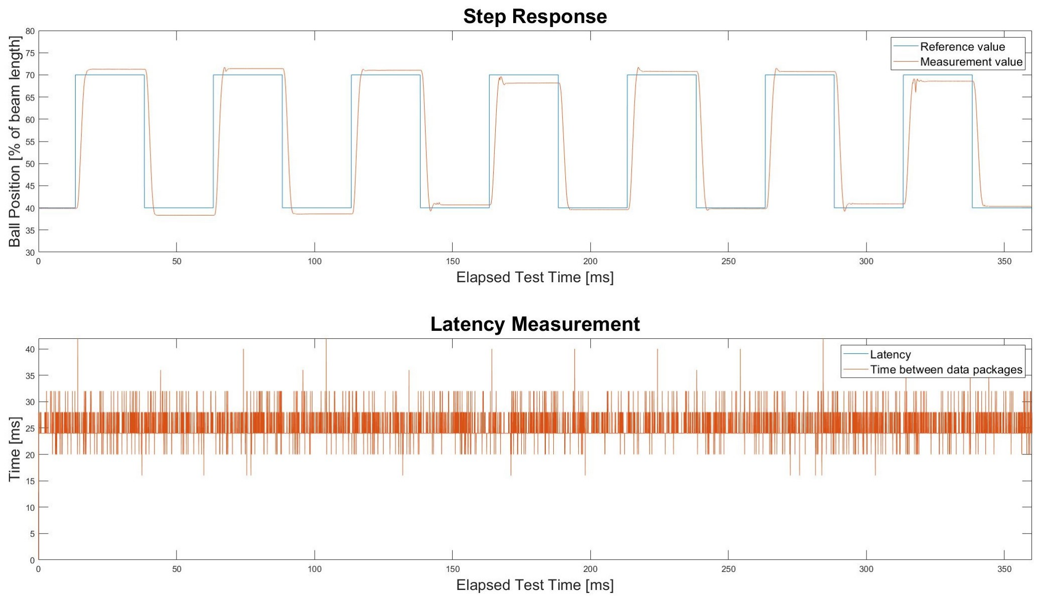


Figure 4.7: The step responses, latency and time between data packages from the test conducted to evaluate control algorithm dependency using Modbus TCP and WiFi

4.3.2 Network Variables

The test that was run to evaluate the performance effects of the control implementation using the Network Variables protocol can be seen in figure 4.8.

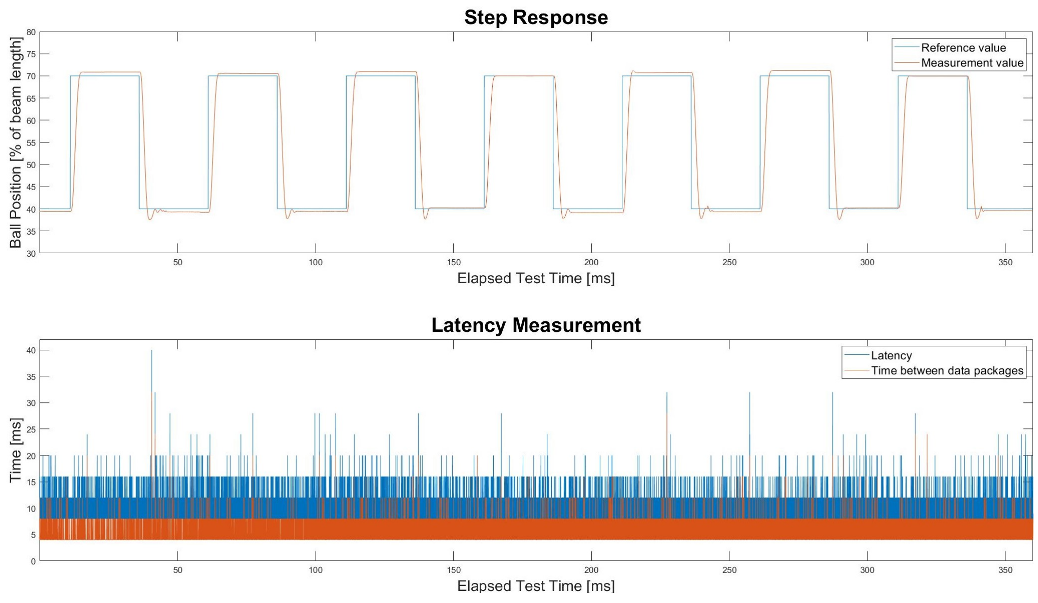


Figure 4.8: The step responses, latency and time between data packages from the test conducted to evaluate control algorithm dependency using Network Variables and WiFi

The KPI:s from the tests can be seen in table 4.6. The performance of Modbus TCP for the control KPI:s improved substantially compared to the original implementation. Its performance thereby became comparable to that of the Network Variables protocol. The Network Variables protocol barely changed with the improved control algorithm.

Table 4.6: KPI:s for the tests conducted to evaluate the dependency of the control implementation on the control performance when using WiFi as communication medium

KPI \ Communication Protocol	Modbus TCP	Network Variables
Rise Time t_r [s]	1.857	1.701
Settling Time t_s [s]	3.018	2.695
MAE	3.348	2.788
Mean Absolute Control Signal $\|u\|$	413.9	350.7
Mean Latency L [ms]	24.77	9.920
Mean Time Between Data Packages $diff$ [ms]	24.74	5.355

4.4 Event-Based Control Saves Communication

To evaluate the performance and the communication savings of the event-based control algorithm, the test seen in figure 4.9 was conducted. It was performed over WiFi using the Network Variables protocol. The latency L, the sampling interval T_s, the PID parameters, the filter variable β and the thresholds that were used for the test can be found in table 4.7. The KPI:s that the test resulted in can be seen in table 4.8. The control KPI:s slightly increased compared to the time-driven system and the energy consumption was also increased. This trade-off, which enabled communication savings of above 75% is put in perspective in section 5.4.

Table 4.7: Parameters for the test investigating the communication savings of event-based control over WiFi

Parameter\Communication Protocol	Network Variables
L [ms]	16
T_s [ms]	4
$K_{p,out}$	0.7519
$K_{i,out}$	0.02710
$K_{d,out}$	1.013
β	0.2220
$K_{p,in}$	2.545
Control Signal Threshold [% of the previous control signal]	5
Ball Movement Threshold [% of beam length]	0.3964
Beam Movement Threshold [% of motor angle range]	0.4200

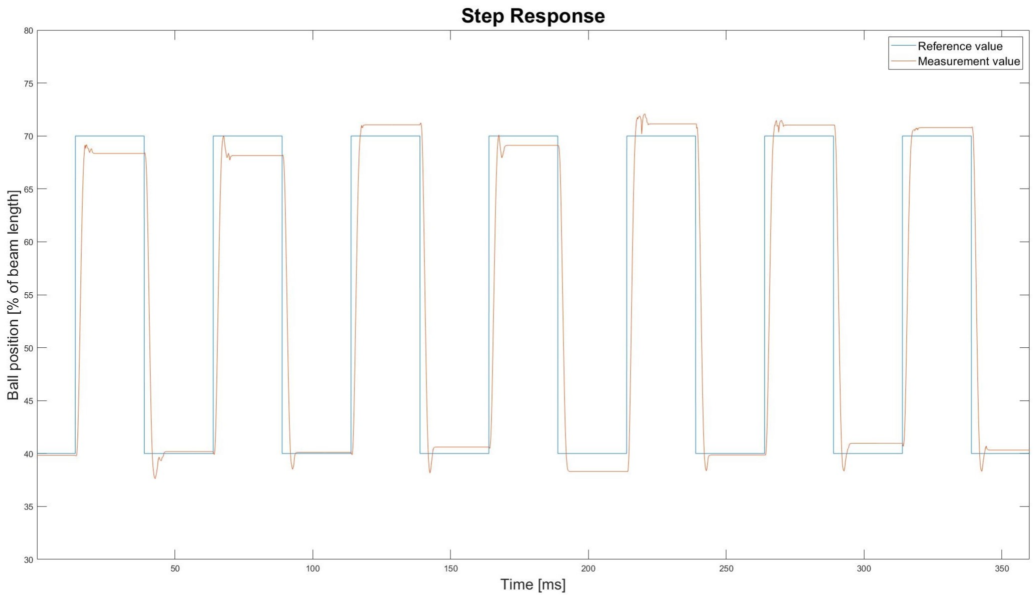


Figure 4.9: The step responses from the test conducted to evaluate the effects of an event-based control strategy when controlling wirelessly over WiFi and using the Network Variables protocol

Table 4.8: KPI:s for the tests conducted to evaluate the communication savings of event-based control over WiFi

KPI \ Communication Protocol	**Network Variables**
Rise Time t_r [s]	1.772
Settling Time t_s [s]	2.720
MAE	3.022
Mean Absolute Control Signal $\lvert u\rvert$	428.9
Saved Control Signal Transmissions $\%_{saved,Control}$	0.8153
Saved Ball Measurement Transmissions $\%_{saved,Ball}$	0.9870
Saved Beam Measurement Transmissions $\%_{saved,Beam}$	0.9824

Figure 4.10 illustrates the controller behavior during step responses. It can be seen in the lower part of the figure that the controller was fully running during the step transients and was off for most of the time after the ball had settled. Note that this figure does not directly translate to the amount of saved communication since the controller would always run while the setpoint changes but the threshold for the control signal can prevent transmission of every value.

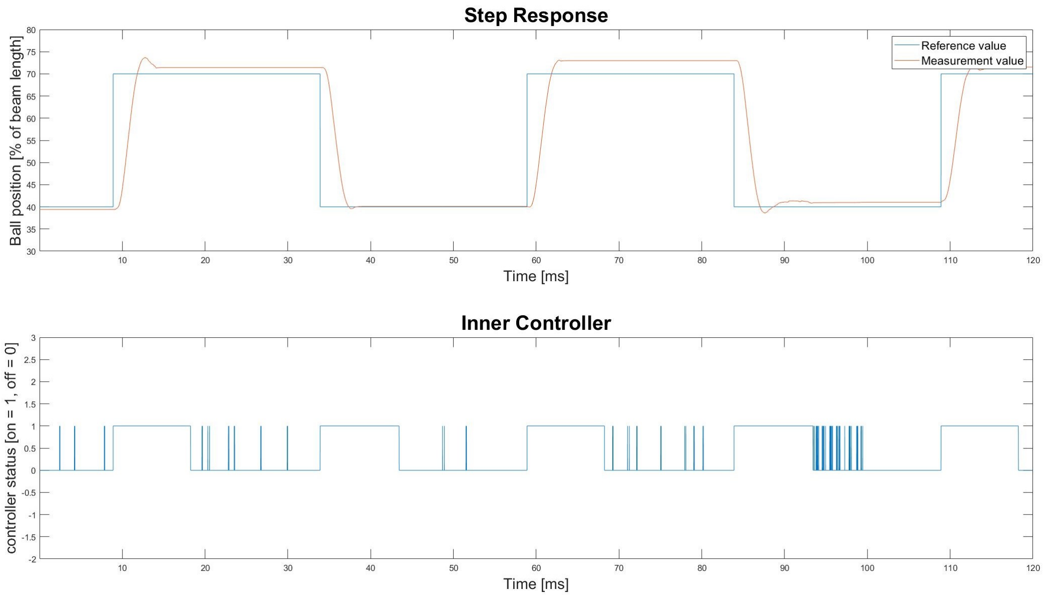


Figure 4.10: Demonstration of the correspondence between the inner controller status and step responses when wireless event-based control was investigated using WiFi

4.5 Event Thresholds can be Adapted at the Sensor

The resulting step responses for automatic tuning of the event thresholds in the sensor according to section 3.7.1 can be found in figure 4.11. This test was, in similarity to the previous test, run over WiFi using the Network Variables protocol. The latency L, the sampling interval T_s, the PID parameters, the filter variable β and the thresholds that were used for the test can be found in table 4.9. The resulting KPI:s were to the largest extent similar to the previous event-based results and are presented in table 4.10. The self-tuning threshold algorithm, configured to save 95% of the measurement signal communications, resulted in less harsh values then what was manually tuned in the previous test case. Thus, lower communication savings were experienced.

Table 4.9: Parameters for the test investigating the communication savings of event-based control with automatically tuned thresholds and WiFi as communication medium

Parameter\Communication Protocol	Network Variables
L [ms]	16
T_s [ms]	4
$K_{p,out}$	0.7519
$K_{i,out}$	0.02710
$K_{d,out}$	1.013
β	0.2220
$K_{p,in}$	2.544
Control Signal Threshold [% of the previous control signal]	5
Ball Movement Threshold [% of beam length]	0.07930
Beam Movement Threshold [% of motor angle range]	0.2500

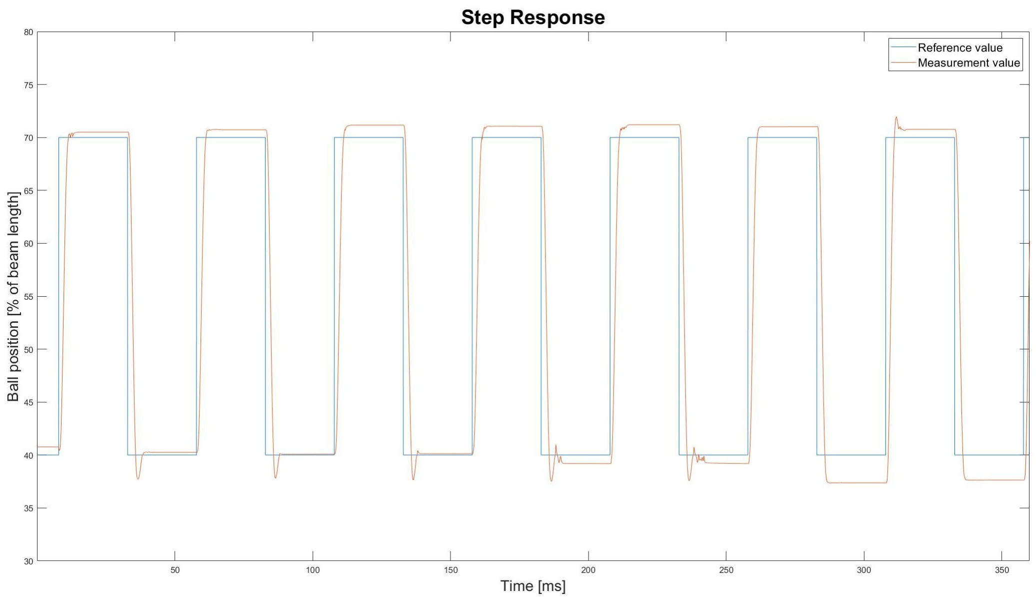


Figure 4.11: The step responses from the test conducted to evaluate the effects of automatic threshold tuning in the sensor using WiFi as communication medium

Table 4.10: KPI:s for the tests conducted to evaluate the communication savings of event-based control with automatically tuned thresholds and WiFi as communication medium

KPI \ Communication Protocol	**Network Variables**
Rise Time t_r [s]	1.709
Settling Time t_s [s]	2.721
MAE	3.084
Mean Absolute Control Signal $\lvert u \rvert$	424.2
Saved Control Signal Transmissions $\%_{saved,Control}$	0.7627
Saved Ball Measurement Transmissions $\%_{saved,Ball}$	0.9515
Saved Beam Measurement Transmissions $\%_{saved,Beam}$	0.9451

4.6 The Method is Generic for Other Wireless Media

One final test was performed to evaluate if the proposed event-based control approach with automatically tuned thresholds in the sensor is generic and feasible also over other wireless media. It was conducted using 5G and the Network Variables protocol. The latency L, the sampling interval T_s, the PID parameters, the filter variable β and the thresholds that were used for the test can be found in table 4.11. To ensure comparability between this and the previous test, the thresholds were chosen to be the same. The step responses from the test can be seen in figure 4.12 and the corresponding KPI:s are displayed in table 4.12. The KPI:s were similar to the ones of the corresponding test conducted over WiFi. This indicates that the hypothesis was successful, an indication further discussed in section 5.1.5.

Table 4.11: Parameters for the test conducted to investigate the effects of using 5G as the wireless communication medium

Parameter\Communication Protocol	Network Variables
L [ms]	24
T_s [ms]	4
$K_{p,out}$	0.7481
$K_{i,out}$	0.02690
$K_{d,out}$	1.011
β	0.2208
$K_{p,in}$	2.271
Control Signal Threshold [% of the previous control signal]	5
Ball Movement Threshold [% of beam length]	0.07930
Beam Movement Threshold [% of motor angle range]	0.2500

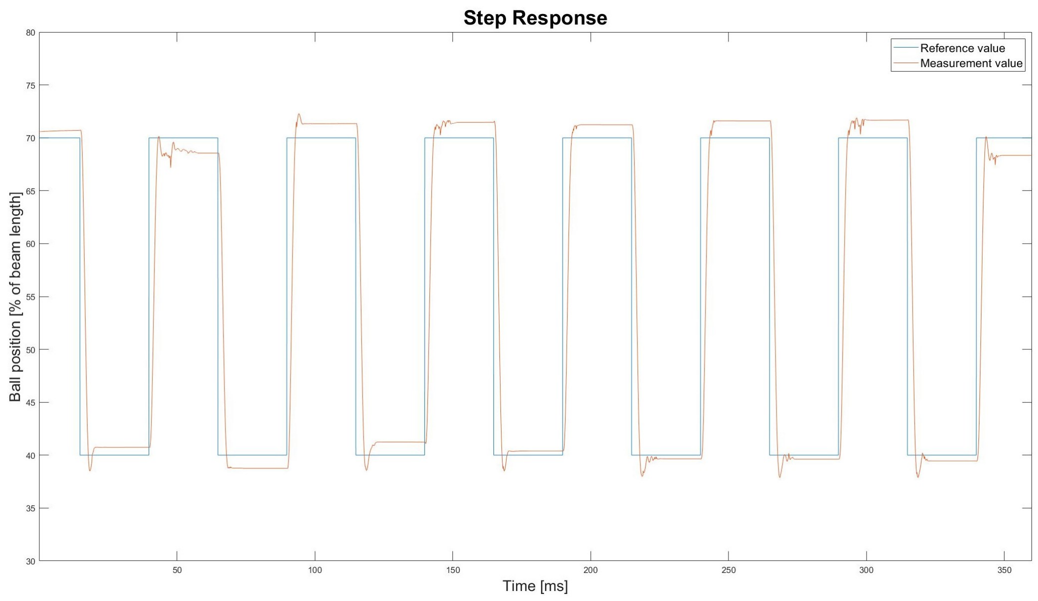


Figure 4.12: The step responses from the test conducted to evaluate the effects of using 5G as communication medium for wireless control with automatic threshold tuning in the sensor

Table 4.12: KPI:s for the tests conducted to evaluate the communication savings of event-based control over 5G with automatically tuned thresholds

KPI \ Communication Protocol	**Network Variables**
Rise Time t_r [s]	1.739
Settling Time t_s [s]	2.714
MAE	3.136
Mean Absolute Control Signal $\lvert u\rvert$	461.9
Saved Control Signal Transmissions $\%_{saved,Control}$	0.7624
Saved Ball Measurement Transmissions $\%_{saved,Ball}$	0.9427
Saved Beam Measurement Transmissions $\%_{saved,Beam}$	0.9422

5 Discussion and Conclusions

This chapter will be devoted to discussions and conclusions based on results found in the thesis. It commences with a section structured according to the five hypotheses. More general discussion topics with connections to the results in a broader sense will be covered at the end of the chapter.

5.1 Results Analysis

A topic of discussion relevant for all of the results is the choice and tuning of the controllers. It is possible that better control performance could have been achieved using a controller design method different to IMC. As described in sections 3.6.1 and 3.6.2, the tuning of the controllers was performed based on theory and experiments. It is possible that a tuning approach with a stronger theoretical anchoring would have resulted in increased controller performance.

The tests were, as described in section 3.7, run for 6 minutes each. Longer test times might be beneficial since it could have reinforced the belief that the gathered results were indeed the true results and not a product of coincidence. Another way of increasing the result reliability could be to run the same tests several times.

5.1.1 Protocol Matters

One protocol specific detail, highlighted already in section 4.1, is that the latency, L, and the time between the data packages, $diff$, were the same for the tests run over Modbus TCP. This is, however, in line with the expected behavior of a TCP protocol. As stated in section 2.4.1, the client requests data from the sensor and then waits for the sensor to provide the data. A new request is therefore not considered until the previous one has been completed. This strategy of communication results in the same values for $diff$ and L.

The two UDP protocols, Network Variables and OPC UA PubSub displayed characteristic UDP behaviors. Their $diff$'s were low, for Network Variables at times as low as the AIO device's cycle time of 4 milliseconds. This can be explained by the UDP protocol's lack of handshaking. New measurements are sent from the sensor as a result of user configurations rather than a request from the client. Thus, the $diff$ converged towards the cycle time of the server in these tests since it was configured to send new measurement values every cycle. An interesting difference between Network Variables and OPC UA PubSub is how the L's differ between figure 4.5 and figure 4.6. With both protocols being UDP based and the communication running over WiFi in both tests, many latency-affecting factors were the same. One explanation for the difference is that their communication configurations and implementation in CODESYS might differ. Another explanation, possibly connected to the first one, is the novelty of the OPC UA PubSub CODESYS library. As it was launched in October 2020, it might not be as mature and efficient as the older Network Variables protocol.

Concerning the control performances that were observed in the tests, it was clear that the Modbus TCP protocol displayed the poorest performance. The Network Variables protocol and the OPC UA PubSub protocol gave similar and better results. If an assessment of the control performance is made with L and $diff$ taken into consideration, one possible conclusion that can be drawn is that the L is of less importance for the performance than the $diff$. The basis for this would be that the OPC UA PubSub communication, seen in figure 4.6, displays similar values of L as the Modbus TCP communication, seen in figure 4.4, whilst still producing step responses alike the ones of the Network Variables protocol seen in figure 4.5. Another similarity between the two well performing protocols was that they presented lower values for the $diff$ than for the L, which made the conclusion presented above possible.

Finally, a comparison between the tests mentioned in this discussion section and the wired reference tests displayed few significant differences. However, a notable divergence between them is the measured L and $diff$ for the OPC UA PubSub protocol tests. The one using WiFi displayed a larger distribution of the values. Despite the variance difference of the L and the $diff$ in the wired and wireless case, a comparison of their KPI:s shows small differences between the tests since the KPI:s are representing mean values. One explanation for this behavior might be that the cycle times of the client and the server aligned differently in the two cases. Another one could be that the OPC UA PubSub protocol was somehow affected by the communication medium being WiFi. In order to confirm or refute these explanations, additional tests are needed.

5.1.2 Control Implementation Matters

The altered control implementation did not have much impact on the L and the $diff$ of the two tests, which can be seen in figures 4.7 and 4.8. This result was expected since a new control algorithm should theoretically not affect the communication between the devices.

The greatest difference introduced by the new implementation was the Modbus TCP protocol performance. The step responses from this test yielded a performance close to the one of the Network Variables protocol. One of the included algorithm changes was the use of $diff$ in the control calculations instead of the vPLC cycle time, as described in section 3.7. Thus, the value of the variable T_s for the Modbus TCP protocol went from 4 ms to the value of $diff$. According to the KPI table 4.6, this was on average equal to 25 ms. This sampling interval, equal to the value of $diff$, will henceforth be referred to as the communication sampling interval. It was used in the calculations of the I- and D-parts of the discretised outer controller and the filter for the ball position, F_{out}, as described in sections 2.6.1 and 3.5.2. When comparing these Modbus TCP tests to the previous ones conducted, it can be concluded that the use of the communication sampling interval affected those variables significantly. The step responses in figure 4.7 confirms the effects on the I-part since these steps present a noticeable reduction of static errors.

A similar reasoning as the one in previous paragraphs provides an explanation to the unchanged behaviors of the Network Variables protocol. In similarity to Modbus TCP, it previously used the cycle time of 4 ms as T_s. With the new control implementation where T_s is equal to the value of the variable $diff$, it can be seen from the $diff$ KPI in table 4.6 that it on average had a value of 5 ms. The difference between the previously used T_s and the communication sampling interval that is now used is, thus, much smaller for the Network Variables protocol, which can serve as an explanation for its practically unchanged control behavior.

A final and concluding observation regarding the tests connected to hypothesis 1 and 2 is that the Modbus TCP protocol still had longer L and $diff$ than the Network Variables protocol. However, as a result of the new control implementation, the Modbus TCP protocol now resulted in control performance KPI:s comparable to those produced by the Network Variables protocol.

5.1.3 Event-Based Control Saves Communication

The event-based control application showed promising results. As mentioned in section 4.4, a remarkable amount of communication could be saved by accepting a slight degradation of the control performance. The implemented application imposed a threshold not only for the sensor measurements but also for updates of the communicated control signal. These thresholds were all manually configured. As with many other non-automatic tuning tasks, intuition and insight regarding the process and its behavior are crucial. The thresholds should be set sufficiently high to avoid unnecessary communication due to measurement noise. Meanwhile, awareness of accepted control performance degradation is of equally high importance. Additionally, as mentioned in the corresponding results section, the control signal threshold is the reason for the discrepancy between the saved communication KPI, found in table 4.8, and the controller status seen in figure 4.10.

The initiation to run the controllers was triggered based on two causes, either a change in setpoint or a receival of new measurement values. If the flag was set due to the first cause, both the outer and inner controller would run. However, if the reason was the second cause, both controllers would not necessarily be run, depending on which sensor generated the measurement. This approach was undertaken for two reasons, the first one being to simulate a real industrial situation, where sensors are delocalised. The second reason was the desire to optimise algorithm efficiency. Since measurement values are sent individually from the sensors, there is a possibility that only one of them is updated. If a new beam measurement is received while the ball measurement remains unchanged, only the inner controller has to run. The cascaded control structure provides the explanation for this. The ball measurement is the input to the outer controller, whose output is the reference for the inner controller. When no new ball measurement has arrived, the reference signal from the outer controller to the inner controller will not change. A new beam measurement therefore only requires the inner controller to run. Hence, computational occupancy can be prevented.

5.1.4 Event Thresholds can be Adapted at the Sensor

The event-based application was further enhanced by the feature of adaptive threshold tuning in the sensor. The examined process displayed stationary characteristics between the step changes, entailing recurrent recordings of identical measurements. Processes with this behavior can therefore achieve great communication savings with relatively small thresholds, as long as poor sensor resolution is considered.

The proposed tuning method is based on the operator configuring the desired percentage of communication savings, a setting that might not be entirely intuitive. A more conventional implementation could be based on the accepted control performance deviation. With that implementation, the operator would instead configure the desired maximum diversion from the control setpoint. To manage tuning of event thresholds guaranteeing this desired control performance, a more sophisticated control algorithm with prediction ability would be required. The tuning task would then also have to be of a more complex character. To achieve this, one alternative is to move it from the sensor to the controller. Another alternative is to enable two-way communication between the nodes, which is further discussed in section 6.2.

Self-contained applications are an attractive product that minimises the need for manual maintenance and interference. In the case of changes to the surrounding environment, an advantage of the self-tuning algorithm is that it can be run again to adapt the thresholds to the new situation. This functionality might be suitable to incorporate in the HMI of the application.

5.1.5 The Method is Generic for Other Wireless Media

The justification for not using 5G as the primary wireless medium in this project is its lack of repeatability due to the current network implementation. However, it was well-suited as a final test to confirm that earlier findings hold also over this promising medium. The test acknowledges the control method's scalability, which will be an advantage as 5G usage is presumed to expand within industries, as elaborated upon in section 2.1.3.

5.2 Communication

The segment of this thesis devoted to investigation of different communication protocols has illustrated that the choice of protocol could carry great importance for control applications. This especially applies to wireless control targets due to the increased delays imposed by cellular networks, described in section 2.1.2. An influential aspect of the protocol performance is the process characteristics. A time-critical motion process, alike the one used in this work, could suffer severely by a poor protocol choice if, for example, the lack of new measurements is not compensated for by a conscious control algorithm. This concept is further discussed in section 5.3. A process with slower dynamics and a cycle time closer to the minimum communication time that the protocol can provide might not experience any performance drawbacks. This further emphasises the need for good process knowledge in control engineering. However, the choice of protocol is often not completely free but depends on other factors, such as company standards and compatibility with other network interfaces.

As brought up already in section 5.1.1, the CODESYS OPC UA PubSub library was released quite recently and the version used in this thesis is the first one. It is therefore suspected that subsequent versions will be more stable and unlock the true potential of a unified architecture protocol. OPC UA PubSub offers the advantage of not being bound to the CODESYS software in the same way as the Network Variables protocol is. Additionally, it has the great advantage of being compatible with several device manufacturers.

A general conclusion regarding Ethernet and WiFi as communication media is that the performance of WiFi exhibits similar results to those of Ethernet for the tested protocols. This is despite WiFi being wireless, indicating a promising future for wireless control.

5.3 Control Performance

This thesis has shown that latency awareness can be used as a tool in a control algorithm to counteract poor control performance due to slow communication protocols. The theoretical control algorithm does not work very well when there is a large discrepancy between the expected and actual interval of incoming measurements. This performance degradation can be rectified by considering the increased interval as proposed in section 3.7.1. The reasoning for this is that the change in measurement value is assumed to have occurred during the entire sampling interval rather than the application cycle time. Thus, the accumulated integral term will become greater, resulting in more accurate process control. Another effect of using the longer sampling interval is that the derivative term will decrease. In other words, a correctly designed and tuned controller can relax the requirements on the communication protocol.

The suggested improvement additionally consists of keeping a constant control signal between the arrival of measurements. This was implemented to prevent the I- and D-parts of the outer controller from using the same measurement values multiple times, accumulating the values of these terms wrongly. The effects that this alteration and the timestamp management, mentioned in section 3.7, impose are challenging to assert. Implementing the suggested improvements incrementally would have helped to isolate the effects of each change. Tests aiming to study the likeliness of messages arriving in incorrect order would also have provided value for this analysis.

As acknowledged above, observations of the L and $diff$ KPI:s provide valuable insights regarding the control task performance. One thing that could be investigated further is the robustness of the improved control algorithm. Specifically, it might be of interest to examine the existence and magnitude of a maximum handleable difference between the vPLC cycle time and the measured $diff$, used as the sampling interval. However, it may be that the limiting factor for sufficient control instead depends on other process specific characteristics.

5.4 Event-Based Control

The event-based controller was implemented as described in section 3.5.1. Using the current error instead of the old ball position error did probably not impair the control performance too much. In fact, it can be argued that the use of a newer error is beneficial for the controller since it entails usage of a more accurate value.

The second difference to the implementation of the PIDPlus controller had more noticeable effects. As described in section 2.7.2, the proposed method is to use the actual elapsed time between two communications as the sampling interval when calculating the I- and D-part of the control signal. This type of sampling interval, based on the frequency of the communication, was introduced as the communication sampling interval in section 5.1.2. It was successfully used in hypothesis 2 where it entailed improved control performance for the Modbus TCP protocol. However, when utilising the communication sampling interval for the event-based controller on the ball-and-beam process in hypothesis 3, the control performance was significantly reduced. The step responses from a hypothesis 3 test using the communication sampling interval can be seen in figure 5.1.

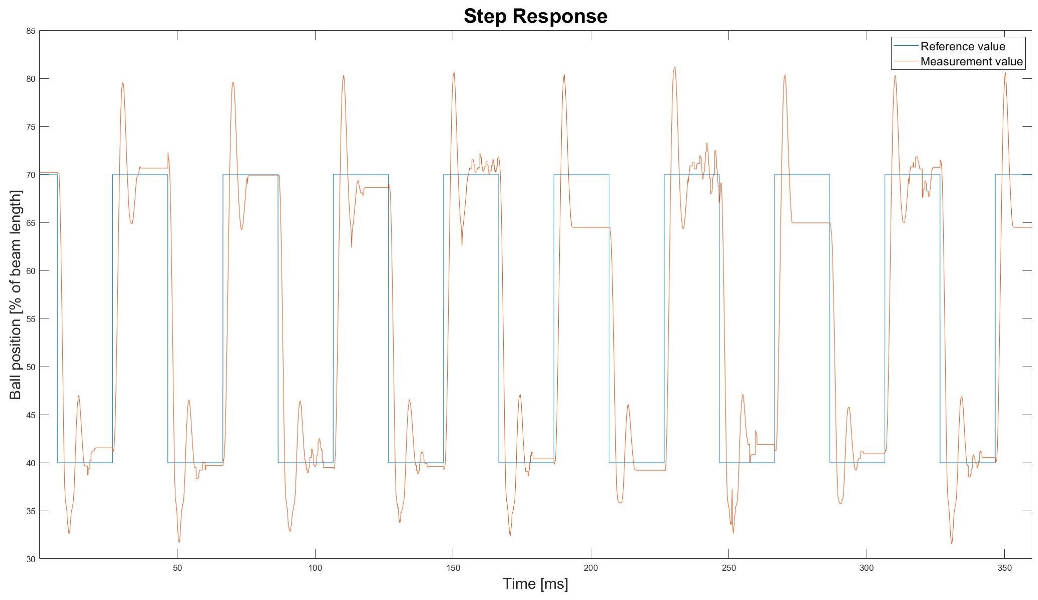


Figure 5.1: Event-based control step responses using the communication cycle time

When comparing these results with the ones presented in figure 4.9, it becomes clear that the desired control performance was achieved when the cycle time of the vPLC was used as the sampling interval. In a PID controller, the proportional part is not dependent on the sampling interval. The integral part of the outer controller used in this thesis is derived using the communication sampling interval. Hence, the only difference between the two hypothesis 3 tests is the sampling interval that was used to calculate the derivative part of the outer controller.

As described in section 2.6.1, the D-part of a PID controller is calculated as a derivative and can be seen as the part of the controller that tries to predict the future process behavior. In this thesis, the discretised D-part was calculated according to (2.15). In the equation, it can be seen that the calculation involves a division by the variable T_s. Since the communication sampling interval is equal to or longer than the vPLC cycle time, the use of a T_s equal to the communication sampling interval results in a smaller derivative part.

Another purpose of the D-part in a PID controller is its contribution to the reduction of overshoots of the controlled variable, as described in section 2.6.1. In figure 5.1, the step responses using the communication sampling interval displays large overshoots and oscillations. One possible explanation to this is that the D-part becomes too small when the communication sampling interval is used, as elaborated upon in the previous paragraph. This can reduce the controller's ability to dampen process oscillations. The ball-and-beam specific process behaviors can aid in the search for an explanation to the highly oscillatory step responses seen in figure 5.1. Small thresholds in combination with friction that obstructs the ball-translation indicate that movements of the ball are rapid when they occur. Hence, it is unlikely for the ball to move linearly with a constant speed when translation of it is initiated. This is, though, assumed when the communication cycle time is used to calculate the derivative part of the control signal. A graphical example of it can be seen in figure 5.2 where D_1 is calculated using a T_s equal to the communication sampling interval and D_2 is calculated with T_s equal to the sensor cycle time. From this figure it is clear that D_2 results in a derivative much closer to the true derivative of the ball's movement than D_1 does. The communication sampling interval reduces the size of the D-part dramatically which, most likely, is the reason for the poor control performance seen in the step responses introduced in this section.

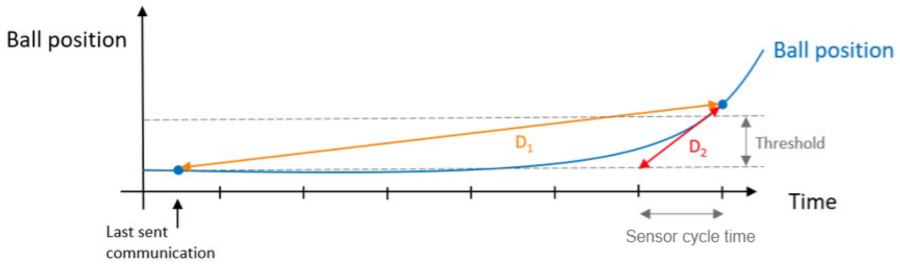


Figure 5.2: A comparison of the D-part calculated using the communication sampling interval, D_1, or the controller sampling interval, D_2

A question that naturally arises in light of the previous discussion is why the communication sampling interval resulted in improved control performance in hypothesis 2 and worsened control performance for the latter, event-based, hypotheses. One possible explanation to this is that the time-based controller, used in hypothesis 2, did not receive new measurements every controller cycle due to the utilised communication protocol. When the communication sampling interval was used in hypothesis 2, the calculations were based on the actual performance of the communication link meaning that they gave more accurate results. In comparison, the goal of the event-based controller was reduced communication so it was not necessarily bad for it to not receive new measurement values every cycle. It might simply be a result of little to no ball movements. Longer communication times were therefore, in this case, not caused by poor communication link performance. Further explanations to the behavior that the different values of T_s gave can come from the fact that it was the I-part that showed improved behavior when the communication cycle time was implemented in the time-based controller. The event-based controller kept this implementation of the integral part but altered the one of the derivative part to suit the new, event-based, situation. Thus, using the vPLC cycle time to calculate the derivative part of the outer controller can be seen as an add-on to the improved algorithm rather than a return to the old strategies.

When implementing event-based control, the overall goal is to save communication. However, there is often a trade-off to the savings that can be seen in the control performance. This compromise can be subject to discussion and requires case-by-case evaluation of each potential control target. Some processes might have very strict error tolerances or swiftly changing setpoints that can be difficult to satisfy or follow using event-based control. As a result, it is of high importance to consider the suitability of event-based control for the intended control target. Event-based control was seen as a promising strategy for the ball-and-beam process due to the relatively long stationary periods of the ball position between reference changes. However, the unstable and fast dynamics of the process imposed difficulties.

To conclude the tests conducted in this thesis, event-based control shows promising performance results even for unstable and fast processes like the ball-and-beam. The proposed alteration to the D-part calculations of the event-based controller is probably best suited for processes with fast dynamics. Other attributes that might make a process fitting for the alteration are large but few setpoint changes or disturbances. A final and general conclusion to draw is that the full potential of the event-based PID controller is unlocked only when its calculations are altered to suit and reflect the behavior of the designated target process.

6 Future Work

This chapter proposes future research recommendations for the thesis topics and presented methods. It also debates the chosen implementation strategies and suggests modifications to these.

6.1 General Improvements

Evaluating the thesis methodology in retrospect, a few further improvements would be recommended to investigate. One of these could be to expand the event-based area in favor of implementing an automatic tuning of the control signal threshold in similarity to the measurement thresholds. The thesis might also benefit from a more refined approach for calculating the amount of saved communication for deeper understanding and analysis of the phenomena. Another improvement could be to increase the robustness such that the controllers and thresholds would readapt their parameters in case of a substantial change of conditions, making the application more autonomous and independent of the settings at launch.

To enhance the legitimacy of the proposed methods and found results, the tests could have been run on an additional process to confirm the results from the ball-and-beam. Additionally, a common strategy to confirm physical results in research contexts is through simulations of the tests. However, the delays experienced in the discussed setup and event-triggered scheme could complicate building such simulation environments.

6.2 Intelligent Sensor

6.2.1 Two-way Communication

As discussed in section 3.7.1, this project assumes that no communication is possible from the controller to the sensor. The advantage of this assumption is that the presented findings and methods could be realised in a wide range of applications where the architecture enforces such constraints. Allowing communication in both directions would, however, enable more complex approaches. A sensor capable of both receiving and processing information from the controller will hereafter be referred to as an intelligent sensor. One suggested way to exploit the two-way communication is to let the intelligent sensor be aware of the process setpoint. This scenario enables the possibility to calculate the integral term already in the sensor. The benefit of this is that the integral would be updated with the correct, consistent, sampling interval and then it would be communicated to the controller. This approach could be feasible for both time- and event-driven applications.

6.2.2 The Threshold Tuning Algorithm

Another feature of the intelligent sensor is that a more complex threshold tuning for event-based control could be developed through the use of the additional accessible information. The operator would then, instead of configuring desired saved amount of communication, insert an error tolerance for the setpoint. A controller with prediction ability could be used to adapt the communication thresholds. The adaptation would be based on anticipating the minimum distance between the controlled variable and its tolerance bound where the control structure still manages to readjust the controlled variable and retain it from exceeding the tolerance bound. The event trigger is then set to the process variable crossing this minimum distance, which sparks the communication and consequently initiates the control algorithm.

6.3 Prospective outlook

The wireless control task in this thesis project opens up for a number of complex approaches that would be interesting to investigate. These include, but are not limited to, a shift of the controller into another structure, for example model predictive control (MPC), or an incorporation of advanced data analysis with machine learning. A similarity between those alternatives is the computational requirements they impose. It cannot be expected of all industrial end-devices to possess sufficient power to meet those. However, the utilisation of wireless control entails opportunities to relocate the demanding operations, instead conducting them in the edge or the cloud. Connecting back to the wireless communication being the cause for the need of more sophisticated controllers, it can indeed be concluded that wireless control is the demander as well as the enabler for the alterations. This might entail doubts regarding if they are in fact truly needed. Relocations of industrial computation tasks could, though, prove to be a beneficial investment. Explorations within this field of technology might facilitate a prosperous industrial future.

7
References

[1] The Editors of Encyclopaedia Britannica. *Diophantine equation.* Accessed: 2022-05-30. URL: https://www.britannica.com/science/Diophantine-equation.

[2] OPC Foundation. *Unified architecture.* Accessed: 2022-04-14. URL: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[3] CODESYS Group. *CODESYS Development System V3.* Accessed: 2022-04-29. URL: https://store.codesys.com/en/codesys.html.

[4] CODESYS Group. *CODESYS Network Variables.* Accessed: 2022-04-14. URL: https://help.codesys.com/api-content/2/codesys/3.5.13.0/en/_cds_f_networkvariables/.

[5] CODESYS Group. *CODESYS Network Variables.* Accessed: 2022-04-14. URL: https://forge.codesys.com/prj/codesys-example/network-variabl/home/Home/.

[6] CODESYS Group. *CODESYS OPC UA PubSub SL.* Accessed: 2022-04-14. URL: https://store.codesys.com/en/opc-ua-pubsub-sl.html.

[7] *ISA95, Enterprise-Control System integration - ISA.* Accessed: 2022-04-14. URL: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95.

[8] Quanser. *Ball and Beam Image.* Accessed: 2022-04-29. URL: https://www.quanser.com/products/ball-and-beam/.

[9] E. Lindahl and M. Wallberg. *Towards latency-aware control using 5G and Edge-based control architectures.* 2022. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-467541.

[10] A. Nordrum and K. Clark. "Everything you need to know About 5G". In: *IEEE Spectrum* (Sept. 2021). URL: https://spectrum.ieee.org/everything-you-need-to-know-about-5g.

[11] Kang B Lee, Richard Candell, Hans-Peter Bernhard, Dave Cavalcanti, Zhibo Pang, Inaki Val et al. "Reliable, High-Performance Wireless Systems for Factory Automation". In: *NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology* (Sept. 2020). Gaithersburg, United States. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=930630.

[12] T. Hägglund. *Reglerteknik AK Föreläsningar.* 2019. URL: https://www.control.lth.se/fileadmin/control/Education/EngineeringProgram/FRTF05/forel.pdf.

[13] K. E. Årzén, P. Skarin, W. Tärneberg and M. Kihl. "Control over the Edge Cloud - An MPC Example". In: *1st International Workshop on Trustworthy and Real-time Edge Computing for Cyber-Physical Systems* (Dec. 2018). Nashville, United States. URL: https://cps-vo.org/group/TREC4CPS_conference.

[14] IEEE Standards Association. *IEEE SA - IEEE Standard for ethernet.* Accessed: 2022-06-21. Aug. 2018. URL: https://standards.ieee.org/ieee/802.3/7071/.

[15] M. Luvisotto, Z. Pang and D. Dzung. “Ultra High Performance Wireless Control for Critical Applications: Challenges and Directions”. In: *IEEE Transactions on Industrial Informatics* 13.3 (2017), pp. 1448–1459. DOI: 10.1109/TII.2016.2617459.

[16] T. Goldschmidt, M. K. Murugaiah, C. Sonntag, B. Schlich, S. Biallas and P. Weber. “Cloud-Based Control: A Multi-tenant, Horizontally Scalable Soft-PLC”. In: *2015 IEEE 8th International Conference on Cloud Computing* (2015). New York, United States, 27 June-2 July, pp. 909–916. DOI: 10.1109/CLOUD.2015.124.

[17] D. H. Hanssen. *Programmable Logic Controllers: A Practical Approach to IEC 61131-3 using CoDeSys*. J. Wiley & Sons, Hoboken, United States, Nov. 2015. ISBN: 9781118949221. URL: https://books.google.se/books?id=My-PCgAAQBAJ.

[18] T. Blevins, M. Nixon and W. Wojsznis. “PID control using wireless measurements”. In: *2014 American Control Conference* (2014). Portland, United States, 4-6 June, pp. 790–795. DOI: 10.1109/ACC.2014.6858597.

[19] R. Drath and A. Horch. “Industrie 4.0: Hit or Hype? [Industry Forum]”. In: *IEEE Industrial Electronics Magazine* 8.2 (2014), pp. 56–58. DOI: 10.1109/MIE.2014.2312079.

[20] IEEE Standards Association. *IEEE SA - IEEE Standard for Information Technology–Telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications–amendment 4: Enhancements for very high throughput for operation in bands below 6 ghz.* Accessed: 2022-06-21. Dec. 2013. URL: https://standards.ieee.org/standard/802_11ac-2013.html.

[21] *Modbus application protocol specification*. Tech. rep. V 1.1b3. Modbus Organization, Andover, United States, Apr. 2012.

[22] E. Oki, R. Rojas-Cessa, M. Tatipamula and C. Vogt. *Advanced Internet Protocols, Services, and Applications*. J. Wiley & Sons, Hoboken, United States, 2012. ISBN: 0470499036.

[23] D. E. Seborg, T. F. Edgar, D. A. Mellichamp and F. J. Doyle III. *Process Dynamics and control*. J. Wiley & Sons, Hoboken, United States, 2011.

[24] *Stundent Workbook - Ball and Beam experiment for MATLAB/SIMULINK Users*. Quanser Inc. Markham, Canada, 2011.

[25] A. S. Tanenbaum and D. Wetherall. *Computer networks, 5th Edition*. Pearson, London, United Kingdom, 2011. ISBN: 0132553171. URL: https://www.worldcat.org/oclc/698581231.

[26] O. Kaltiokallio, L. M. Eriksson and M. Bocca. “On the performance of the PIDPLUS controller in wireless control systems”. In: *18th Mediterranean Conference on Control and Automation, MED’10* (2010). Marrakech, Morocco, 23-25 June, pp. 707–714. DOI: 10.1109/MED.2010.5547788.

[27] T. Norgren and J. Styrud. *Non-periodic sampling schemes for control applications*. 2010. URL: https://www.diva-portal.org/smash/record.jsf?pid=diva2%5C%3A420850&dswid=5319.

[28] Sabah Al-Fedaghi, Alsaqa Ala’a and Fadel Zahra’a. “Conceptual Model for Communication”. In: *International Journal of Computer Science and Information Security* 6 (Dec. 2009).

[29] D. Chen, M. Nixon, T. Blevins, W. Wojsznis, J. Song and A. Mok. “Improving PID Control with Unreliable Communications”. In: *ISA Expo Technical Conference* (2006). Houston, United States, 16-19 October, pp. 17–19.

[30] *Modbus messaging on TCP/IP implementation guide*. Tech. rep. V 1.0b. Modbus Organization, Andover, United States, Oct. 2006.

[31] JO Flower and EA Parr. “13 - Control Systems”. In: *Electrical Engineer’s Reference Book (Sixteenth Edition)*. Ed. by M.A. Laughton and D.J. Warne. Sixteenth Edition. Oxford: Newnes, 2003, pp. 13-1-13–63. ISBN: 978-0-7506-4637-6. DOI: https://doi.org/10.1016/B978-075064637-6/50013-7. URL: https://www.sciencedirect.com/science/article/pii/B9780750646376500137.

[32] B. Wittenmark, K. E. Årzén and K. J. Åström. *Computer Control: An Overview*. IAFC Professional Brief. International Federation of Automatic Control, 2002.

[33] K. E. Årzén. “A simple event-based PID controller”. In: *IFAC Proceedings Volumes* 32.2 (1999). 14th IFAC World Congress 1999, Beijing, China, 5-9 July, pp. 8687–8692. ISSN: 1474-6670. DOI: https://doi.org/10.1016/S1474-6670(17)57482-0. URL: https://www.sciencedirect.com/science/article/pii/S1474667017574820.

[34] A. J. Isaksson and S. F. Graebe. “Analytical PID parameter expressions for higher order systems”. In: *Automatica* 35.6 (1999), pp. 1121–1130. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(99)00009-6. URL: https://www.sciencedirect.com/science/article/pii/S0005109899000096.

[35] A. P. Swanda and D. E. Seborg. “Controller performance assessment based on setpoint response data”. In: *Proceedings of the 1999 American Control Conference* 6 (1999). San Diego, United States, 2-4 June, 3863–3867 vol.6. DOI: 10.1109/ACC.1999.786240.

[36] D. E. Rivera, M. Morari and S. Skogestad. “Internal model control: PID controller design”. In: *Industrial & engineering chemistry process design and development* 25.1 (1986), pp. 252–265. URL: https://doi.org/10.1021/i200032a041.

[37] C. E. Garcia and M. Morari. “Internal model control. A unifying review and some new results”. In: *Industrial & Engineering Chemistry Process Design and Development* 21.2 (1982), pp. 308–323. URL: https://doi.org/10.1021/i200017a016.

[38] G A Baker Jr. “The theory and application of the Pade approximant method”. In: *pp 1-58 of Advances in Theoretical Physics. Vol. I. Brueckner, Keith A. (ed.). New York, United States, Academic Press, 1965.* (Oct. 1967). URL: https://www.osti.gov/biblio/4454325.