Wireless Event-Triggered Control for Water Irrigation Systems

Jacob Jan Lont

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Wireless Event-Triggered Control for Water Irrigation Systems

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Jacob Jan Lont

August 14, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

The application of optimal control structures for water irrigation systems (WISs) can be enabled by applying wireless event-triggered control (ETC). The term WIS, is used to describe open-water channels, that are mainly used to supply water to farmers all around the world. The water levels in these channels need to be controlled, but because of the large scale of WISs, it is very expensive to create centralized control structures when using wired connections between individual sensors, actuators and a centralized controller. Previously, WISs were typically controlled using individual decentralized (non-communicating) controllers. Applying wireless technologies enables communication between (smart) sensors, actuators and a centralized controller without the expense of installing and maintaining cables over lengths of kilometers. To create such a wireless infrastructure, a network needs to be designed, consisting of multiple nodes that are able to communicate with each other over wireless. Each sensor and actuator will be connected to (or integrated in) a node, just like the centralized controller needs to be connected to a node. To minimize the costs related to creating such an infrastructure, the nodes should have their own power source in order to prevent that a maintenance worker has to change the batteries of the nodes periodically. The nodes could be powered using a solar panel, or by using energy harvesting, which could be done by using a turbine to extract energy from the flow in a water channel. When using such energy sources, it is important to minimize the power consumption of the nodes. Most of the consumed power is used in communication when transmitting information. By applying ETC, the amount of communication between the individual parts of the control system is minimized, while still retaining good closed-loop system control using a centralized controller. In this research, techniques on wireless control, ETC and WIS control are combined and the application of an event-triggered centralized controller is presented using simulations, as well as the achievable reduction in communication compared to regular periodic control. Furthermore, a cyber-physical lab setup is designed and built which makes it possible to test these techniques in the Delft Center for Systems and Control (DCSC) lab.

Table of Contents

List of Figures

List of Tables

Preface and Acknowledgements

During the past twelve months, I have been working on this project in order to bring wireless event-triggered control for water irrigation systems one step closed to being implemented all around the world. All along the project, I have had help and support from several people surrounding me, for which I am very grateful and I would like to thank certain people in particular by dedicating this page of my thesis to them.

In the first place, I would like to thank my daily supervisor ir. Gabriel de Albuquerque Gleizer and supervisor dr. ir. Manuel Mazo Jr. for their great guidance throughout the complete project. I really appreciate the time and effort put in by both supervisors and I cannot thank them enough for the great environment that they have provided during the project, which allowed me to improve myself within multiple disciplines. I would also like to thank all other members of the research group for the interesting discussions and feedback during the weekly meetings and all members of the exam committee for taking the time to read the thesis and for their participation in the defence. For the help on the design and build of the physical lab setup, I would like to thank both lab technicians Wim Wien and Will van Geest. I would like to thank Matteo Trobinger from the University of Trento, Italy to take the time to help me getting started with the communication protocol used to realized wireless ETC for WISs in practice.

Finally, I would like to thank my friends and family, and especially my girlfriend Simone and my father Jaap, who have both greatly supported me to get to the point where I am now.

Delft, University of Technology — Jacob Jan Lont
August 14, 2020

Chapter 1

Introduction

This thesis is part of my graduation project, which is the final project to obtain the MSc degree in Systems and Control at the Delft Center for Systems and Control (DCSC) at Delft University of Technology. The main results and contributions of my work resulting from the research performed as my graduation project are discussed in this document. It also includes the hardware design for a lab setup, and it contains a significant amount of control design theory and application. Most of this theory is explained in this document, however a basic level of knowledge in the field of Systems and Control is expected from the reader. While this thesis is mainly aimed at documenting the progress and findings during my research, its other main purpose is to serve as a document that should reflect my level of skill in the field of Systems and Control. The thesis may also serve as a guideline or as extra documentation for other students that choose to do their graduation project on a similar topic.

Water is one of the main needs of humans to stay alive. Humans are constantly moving in order to improve the water infrastructure all around the world. One specific type of water infrastructure, is called a water irrigation system (WIS). The WISs focused on in this research are so called open-water channels, which are mainly used to supply water to farmers all around the world. These farmers must have water available to be able to grow crops and raise healthy animals. The water levels in WISs must be controlled to be able to supply enough water to the farmers and other water users at any moment. Previously, WISs were mainly controlled using decentralized control structures, which means that individual sections of canals were controlled using individual (non-communicating) controllers [1]. While this type of control can serve the purpose of reliably controlling the water heights, there are some disadvantages. Typically, WISs are modelled as a series of pools, and when there is no communication between the individual controllers, the controllers can not know what is currently happening in the surrounding pools. One could think of large water off-takes for example as a result of a farmer pumping large amounts of water from the canal to his land to water his crops. Studies in this field [2], [3], [4] have shown that an increase in control performance can be gained by the use of communicating controllers. Both the use of distributed control structures and centralized control structures are shown to be suited for this purpose [5]. These optimal control structures cannot be implemented using the previously used infrastructures, that were designed for decentralized control.

The communicating controllers would need an infrastructure that enables the individual parts of the control system like the sensors, actuators and the controllers to send information over large distances. This could be done using cables, which would be very expensive, or using locally powered nodes that would send their measurements over a wireless. A solution would be to use nodes that are powered by solar power or using energy harvesting, so maintenance costs would be minimal and the large expense of installing cables would not be needed. Energy harvesting could be realized using turbines that extract energy from the water flow in a water channel. When using these kinds of energy sources, it is of great importance to minimize the amount of energy consumed by the nodes. Wireless nodes consume most of their energy in radio communication when transmitting information. By applying event-triggered control (ETC) the amount of communication can be minimized, while retaining good closed-loop performance.

When looking at large-scale systems like WISs, in which the time delays are significant [2], there are cases in which there is a lot of redundant measuring. The goal of this project is to design an infrastructure, in which the redundant sampling is taken out by applying ETC, to minimize the power consumption of nodes used in the networked control system (NCS). The NCS is defined as the complete networked control system, consisting of (smart) sensors, actuators and a centralized controller and the corresponding nodes.

In close collaboration of our research group with the D3S Research Group of the University of Trento, a communication protocol called Wireless Control Bus (WCB) was designed and tested. The protocol realizes wireless ETC using a centralized controller in practice. It was based on CRYSTAL [6], [7], which is a protocol that the D3S group developed for event-triggered sampling in wireless sensor networks (WSNs) previously. With the application of this protocol in mind, a cyber-physical lab setup, representing a scaled down WIS, was designed during the project which allows to test the techniques presented in this thesis to be tested in the DCSC-lab.

The main matter of the thesis starts in Chapter 2, which explains the preliminaries on ETC, wireless control, WISs and the communication protocol WCB. In Chapter 3, the architecture of the lab setup is discussed and underpinned. Why specific features are included in the design is justified, and the main design choices are discussed for both the hardware and the software related to the lab setup. Chapter 4 covers the control design techniques used in the project, and explains in detail which steps have been taken in the design process of a robust output feedback controller for a set of plant parameters from literature [5], representing a real-world water channel in Australia. The controller is then tested using simulations, after which the controller is converted to ETC by designing specific triggering functions, which determine when signals are communicated between the sensors of the plant and the controller. In Chapter 5, the results of numerical experiments (simulations) are presented and discussed, which will clearly show the effects of ETC applied to WISs. An explanation on how the worked out techniques of Chapter 4 can be applied to the cyber physical lab setup is presented in Chapter 6. The chapter discusses the identification of parameters of the physical lab setup needed to design a model based robust controller. It subsequently discusses the nonlinear phenomena faced in the real-world implementation and a method on how to handle these nonlinear dynamics in practice. Finally, the thesis is concluded in Chapter 7, in which the individual parts of the thesis are reviewed.

Nomenclature

The set of real numbers is denoted by $\mathbb{R}$. $\mathbb{R}_+$ denotes the set of non-negative real numbers. The set of natural numbers, excluding 0, is denoted by $\mathbb{N}$. The 2-norm of a vector $x \in \mathbb{R}^n$ is denoted by $\|x\| := \sqrt{x^\top x}$. The transpose of a matrix $A \in \mathbb{R}^{n\times m}$ is denoted by $A^\top$. A symmetric square matrix $P \in \mathbb{R}^{n\times n}$ is written as $P \succ 0$ ($P \succeq 0$) if P is positive (semi-)definite. $\|H\|_\infty := \max_\omega \bar{\sigma}(H(j\omega))$ is used to denote the infinity norm of a system H, where $\bar{\sigma}(\cdot)$ denotes the maximum singular value of the matrix argument. The magnitude of a transfer function $T(j\omega)$ at a certain frequency ω is denoted as $|T(j\omega)|$. By $diag(A_1, \ldots, A_N)$, we denote a block-diagonal matrix, with the entries $A_1, \ldots, A_N$ on the diagonal. For a locally integrable signal $w : \mathbb{R}_+ \to \mathbb{R}^n$, $\|x\|_{\mathcal{L}_2} = (\int_0^\infty \|x(t)\|^2 \,\mathrm{dt})^{1/2}$ denotes its $\mathcal{L}_2$-norm, provided the integral is finite. For a signal $w : \mathbb{R}_+ \to \mathbb{R}^n$, the limit from above at time $t \in \mathbb{R}_+$ is denoted by $w^+(t) = lim_{s\downarrow t} w(s)$. Finally, I denotes the identity matrix.

An explicit list of symbols is intentionally not included. The meaning of all symbols is indicated and explained in the neighbourhood of the equations in which the symbols are used. This is done to be able to keep notations as close as possible to the corresponding literature.

Chapter 2

Preliminaries

2-1 Event-triggered control

Nowadays, most control systems are constructed using digital platforms on which a software defined controller is used to steer a physical system in the preferred way. During the last couple of decades, many tools and models have been developed to further improve the field of digital control in all kinds of ways [8]. Before the development and the wide availability of digital platforms, mostly analog controllers were used. Analog controllers are usually continuously measuring and continuously communicating within the system. Digital platforms however, work in a periodic fashion, as each (micro-)processor is designed to work based on clock ticks. The measurement sampling instances are therefore also directly converted to a periodic way of sampling. The rate of communication of signals between the controller and the plant is typically also defined using the same manually chosen sampling period as is used for measurement sampling.

In some cases, faster sampling is needed than in other cases. Faster sampling is mostly used for systems having fast dynamics, which have to be tracked by the controller to be able to steer the system as intended. In other cases, a very small sampling times may be used, resulting in a lot of samples and a lot of communication while this may not be necessary to perform good control in each situation. For example, a system can have very fast dynamics but when the system is controlled into an equilibrium state, it may take fifty samples before any change is detected in the measurements. All these samples still have to be communicated, while communication always consumes energy.

To minimize communication within control systems, event-triggered control (ETC) is introduced [9]. In ETC, a triggering condition is designed, which specifies when signals are communicated from the plant to the controller, as shown in Figure 2-1. In some cases, even an additional triggering condition is designed to minimize the communication of control signals, going from the controller to the plant. Some triggering condition designs seem very trivial at first sight, but they can have a huge impact on the amount of communication. In Section 5 of this thesis, it is presented how the amount of communication is reduced by more than 80% compared to periodic control, using a seemingly simple triggering function.

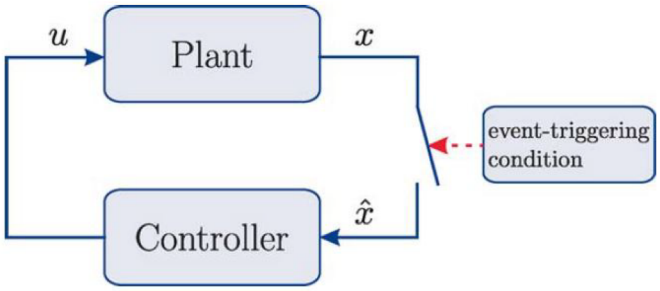


Figure 2-1: ETC loop closing based on an event-triggering condition in which $x \in \mathbb{R}^{n_x}$ is the vector of most recently sampled states, $\hat{x} \in \mathbb{R}^{n_x}$ is the vector of most recently transmitted states, and $u \in \mathbb{R}^{n_u}$ is the vector of control inputs applied to the plant. Figure from [10].

When ETC was first introduced, techniques were developed using continuous sampling of measurements, as in continuous-time control. This version of ETC is now commonly referred to continuous event-triggered control (CETC). Because of the vast majority of digital implementations of controllers nowadays, the focus in this project is on periodic event-triggered control (PETC), which is ETC, with the event-triggering condition being periodically checked. The main reason to use PETC is that the ultimate goal of the project is to be able to apply ETC on a physical lab setup, using a digital implementation of an event-triggered controller.

The measurement (or state) updates, defined by triggering function C, for a controller as in Figure 2-1 can be defined as

$$\hat{x}(t) = \begin{cases} x(t_k), & \text{when } C\left(x(t_k), \hat{x}(t_k)\right) > 0 \\ \hat{x}(t_k), & \text{when } C\left(x(t_k), \hat{x}(t_k)\right) \leq 0 \end{cases}, \quad \text{for } t \in (t_k, t_{k+1}] \tag{2-1}$$

for PETC, in which t_k is the time at which the triggering condition is checked, $x \in \mathbb{R}^{n_x}$ is the most recently sampled set of measurements of the plant and $\hat{x} \in \mathbb{R}^{n_x}$ is the most recently transmitted set of measurements.

All kinds of triggering functions can be composed, but a commonly used triggering condition is defined as

$$\|\hat{x}(t_k) - x(t_k)\| > \sigma \|x(t_k)\|, \tag{2-2}$$

with $\sigma > 0$. A triggering function of comparable form is used later on in the project to simulate the effects of ETC on water irrigation systems (WISs). In that triggering function, the magnitudes of the outputs of the plant and the the magnitudes of the control signals are monitored, instead of the states of the plant.

2-2 Wireless control

Wireless control is the term used to denote control structures using wireless communication between (smart) sensors, controllers, and actuators. Mainly because of the increased computational power on small devices, control systems are increasingly used in large-scale systems and networked infrastructures. A second reason for the use of wireless control is in costs reduction, by not having to use expensive cables anymore to cover large distances

and not having to install and maintain these cables [11] when setting up a networked control system (NCS).

One of the main goals within this project was to create a cyber-physical lab setup to apply ETC on, which enables us to test real-time ETC using a custom designed wireless event-triggered control communication protocol. The power of ETC can really be seen when it is applied in wireless applications. In wireless applications, it is common practice to use battery powered nodes to collect data, like in wireless sensor networks (WSNs). It is of great importance that these battery powered nodes consume the least amount of energy possible, while still being considered reliable. The communication phase of these nodes is a phase in which a lot of energy is consumed, which means that the energy consumption of these nodes can be greatly reduced when reducing the amount of communication, while still being able to perform the needed tasks reliably.

2-3 Water irrigation systems

Water irrigation systems (WISs) are used all over the world to supply water to farmers and others that are in need of water. For farmers, water is an essential ingredient needed to produce crops and raise healthy animals. The products that farmers produce are of great importance to feed people all around the world. To be able to guarantee the required amounts of water, it is needed to perform control on WISs. The control goals on WISs boil down to three measures of interest. The first measure is set-point reference tracking of the water levels in individual sections of the water channel. The second measure is disturbance rejection. These disturbances can come in the form of flow blockage, but also water used by people is treated as a disturbance, because it disturbs the steady state the system is typically in and it is an unforeseen signal influencing the behaviour of the system. These disturbances are typically called off-take load disturbances. The third measure of interest besides set-point tracking and disturbance rejection is minimizing water-level error propagation (WLEP). This is a phenomenon in which water level errors are propagated to upstream pools by requiring water in downstream pools from upstream pools in a non-optimal way. When WLEP is not considered, actuator saturation can occur [2], which can even lead to flooding of pools in certain cases. The goal is to construct a controller that takes all these measures into account, while minimizing the water wastage.

WISs are typically modeled as a series of interconnected pools. A schematic representation of such a pool in shown in Figure 2-2. In this typical representation of a stretch of an open-water channel, the water height y_i, which is measured at the downstream end of pool_i, is controlled by adjusting the gate position p_i at the upstream end of the pool. By adjusting gate position p_i, the flow of pool_{i-1} into pool_i can be controlled, which depends both on the gate position p_i as well as on the difference in water heights between the pools, denoted by h_i. Using the upstream gate (p_i) to control the water height y_i of pool_i is called distant downstream control (DDC). One could also use the downstream gate, denoted by p_{i+1} in Figure 2-2, to control the water height y_i. This strategy will rule out the the transport delays that are introduced by the water having to travel all the way from the upstream gate to the end of the pool, but studies have shown that this strategy leads to more water wastage, compared to DDC [12].

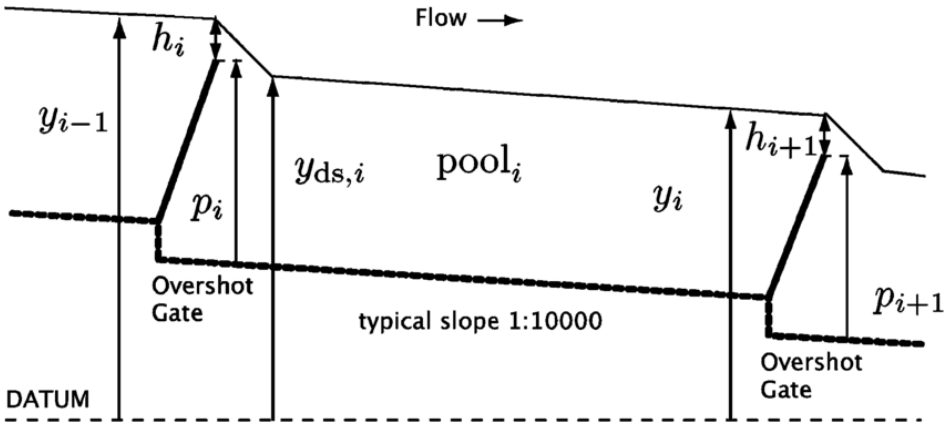


Figure 2-2: Stretch of open-water channel with over-shot gates. Figure from [2].

To test ETC on a real system, a cyber-physical lab setup is created to mimic the behaviour of a WIS. The lab setup consists of a series of pools, of which the water heights are measured using pressure sensors. The water heights are controlled using controllable gates, which are positioned using servo motors. Each gate is connected to a node of the NCS and communicates within the network using the Wireless Control Bus (WCB) protocol. A difference between the schematic representation of a pool shown in Figure 2-2 and the pools of the lab setup is that the lab setup has sluice gates, instead of the overshot gates shown in this figure.

The choice to use a WIS as the system to experiment ETC-techniques on is underpinned by multiple reasons. In studies related to ETC, a commonly used model is that of a WIS [9], [13], because of its large time-delays and slow dynamics, which are properties which usually make that applying ETC can save a lot of resources. A second reason to use a WIS model, is that in a real-world situation the distances between individual sensors and actuators can be multiple kilometers. Because of the large distances that have to be covered, a lot of materials (cables especially) can be saved by using wireless technology instead of wired technologies.

2-4 Wireless Control Bus

At this point, it should be clear why wireless ETC is so interesting for large-scale systems. To show the achievable decrease in terms of communication resources, these techniques are implemented on a cyber-physical system, after being tested using simulations. On the physical system, there will be a number of nodes (small Internet of Things (IoT) boards) that can communicate with each other over wireless. The used nodes are Zolertia Firefly nodes, see Figure 2-3. In cooperation with the D3S Research Group from the University of Trento, Italy, a communication protocol called Wireless Control Bus (WCB) [13] is currently being developed. The protocol will provide the necessary features needed to apply wireless ETC on the setup. It is based on Crystal [7], [6], which is an interference-resilient ultra-low power data collection protocol for WSNs, which was designed by the D3S research group earlier.

Crystal and WCB both have Glossy [14] as underlying communication and time-synchronization primitive. Glossy was designed for efficient network flooding for WSNs. By exploiting constructive interference and the capture effect [15], network-wide flooding is performed by WCB

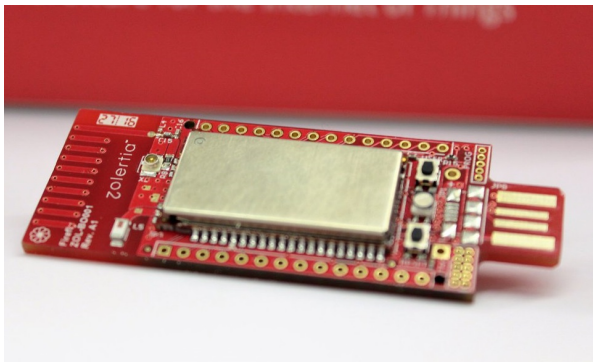


Figure 2-3: Photo of a Zolertia Firefly breakout board.

using Glossy, yielding rapid and reliable packet distribution. The term "network flooding" is used in software architecture to describe routing algorithms in which every incoming packet is sent through to each outgoing link of a node within a computer network, except for the link the packet arrived on. Network flooding is used to make sure each node in the network receives each packet.

The protocol WCB runs inside Contiki-OS [16], which is an open-source operating system (OS) designed to run on low-power micro controllers to make efficient use of available resources. Contiki-OS is used in all kinds of WSN projects, both commercial and non-commercial, ranging from city sound monitoring and networked power meters to alarm systems, remote house monitoring, and more [17].

To let the Firefly nodes make most efficient use of their resources, the high power operations are minimized as much as possible. In this case, that means minimizing communication as much as possible, because transmitting and receiving data are the operations consuming by far the most energy [7]. WCB is designed in a way such that all nodes in the system let their transmission system go to sleep at the exact same moment. This process of turning on and off the transmission system is performed periodically. The period in which the nodes wake up simultaneously is called an epoch, which is depicted in Figure 2-4 using the blue blocks appearing on the timeline periodically.

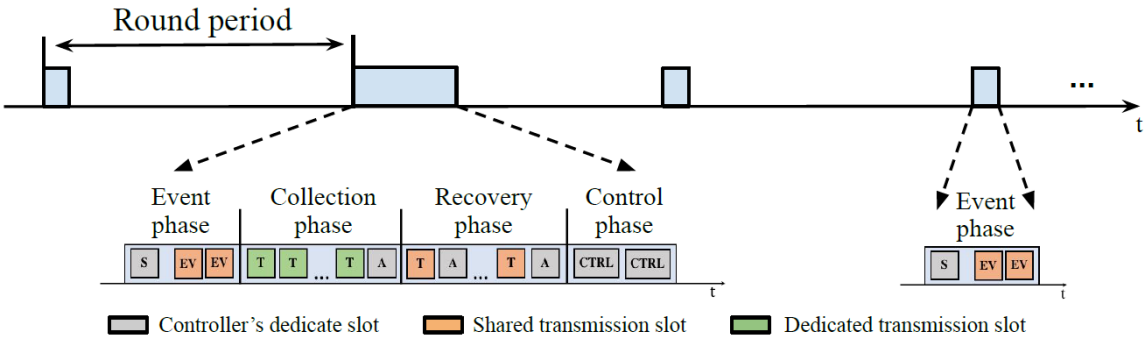


Figure 2-4: Schematic overview of the phases in WCB. Only if there is a trigger in the event phase, then the collection phase, recovery phase and the control phase are initiated. Figure from [13].

From Figure 2-4, it can be seen that not all epochs have equal length. The epochs always start with an event phase, in which it is checked if one of the nodes has triggered based on its triggering condition. If none of the nodes triggered, than the event phase is the only phase in the epoch and all nodes will go to sleep again. This is depicted for the first, third and fourth epoch in Figure 2-4. In the second epoch however, at least one of the nodes in the network has triggered and thus an event occurred, which initiated the collection phase, recovery phase and the control phase. The collection phase start with a series of T (transmission) slots, in which all nodes transmit their most recent sensor measurement. The T-slots are followed by an A (acknowledgement) slot, which is used by the sink node (also referred to as the controller node) to send an acknowledgement package, in which the controller describes which measurements are received (and implicitly which are not). The nodes now check the acknowledgement message in the recovery phase and if the node finds out that its respective measurement was not received by the controller, the node will recover from this by sending the measurement again. After each transmission slot in the recovery phase, an acknowledgement slot is programmed so that in the end of the recovery phase all nodes have received their respective acknowledgement message and have stopped sending their measurements. It is important to acknowledge after each message in the recovery phase, because nodes can overrule messages of other nodes in the network. If there would have been a single acknowledgement slot in the recovery phase, like in the collection phase, it could happen that the signal coming from node_1 is somewhat stronger than the signal of node_2, after which node_2 keeps getting overruled by the signal of node_1. In that case, node_2 would never be able to update its measurements to the controller, which would result in a reliability issue.

Chapter 3

System architecture

The ultimate test for a newly developed product or protocol is to test it in a real-world situation. The most practical way to recreate such a setting including unforeseen disturbances or phenomena without having to interrupt real-world applications, is by creating a physical lab setup. The lab setup used to test the protocol on is a scaled down model of a water irrigation system (WIS). In this chapter the layout of the lab setup will first be discussed in 3-1. The basis of the setup comes from a previously built setup on the faculty of Civil Engineering, but some modifications are made to it, which are elaborated in 3-3. Most of the work needed to get the lab setup ready for applying event-triggered control (ETC) using Wireless Control Bus (WCB) was needed in the electrical domain. The lay-out of the electronics has been changed completely; new printed circuit boards (PCBs), cables and dedicated software has been made, which is all discussed in Section 3-4.

3-1 Plant layout

The plant is a scaled down version of a typical water channel, which are used all over the world to supply water to farmers and other people in dry areas. A schematic representation of the lab setup is shown in Figure 3-1. The setup consists of a series of connected tanks (also called pools), in which water can flow from one pool to the next pool. The water flow between the individual pools is regulated using controllable gates, which are depicted using orange rectangles in Figure 3-1. The water height is the measure that is controlled. It is measured using seven individual pressure sensors (type Keller PR-35), which are depicted by the yellow circles in Figure 3-1. The typical flow directions are shown using arrows, and the manual in- and outflow valves are shown using grey bars, on which the black circle is used to depict the valve-handle used to open and close the valve. The real in- and outflow valves are shown in Figure 3-2. Figure 3-3 shows a picture of the lab setup in its current state in the Delft Center for Systems and Control (DCSC) basement.

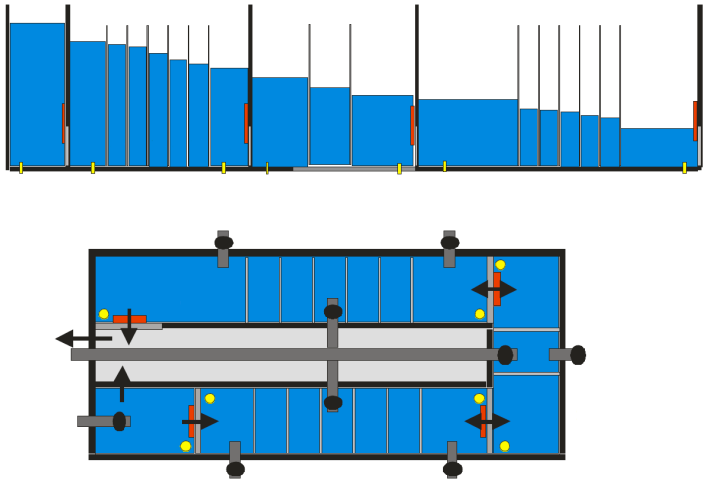


Figure 3-1: Schematic overview of the lab setup. The yellow circles denote pressure sensor positions, while the orange bars are used to depict the movable gates. Original image from [18].

ID	Serial number	Primary IEEE MAC Address	Connected to
FF1	0002625	0x00, 0x12, 0x4b, 0x00, 0x19, 0x4a, 0x51, 0x6c	Gate 1, Sensor 1+2
FF2	0002552	0x00, 0x12, 0x4b, 0x00, 0x19, 0x4a, 0x51, 0x04	Gate 2, Sensor 3+4
FF3	0002601	0x00, 0x12, 0x4b, 0x00, 0x19, 0x32 ,0xe6, 0x91	Gate 3, Sensor 5+6
FF4	0002626	0x00, 0x12, 0x4b, 0x00, 0x19, 0x4a, 0x52, 0x05	Gate 4, Sensor 7
FF5	0002574	0x00, 0x12, 0x4b, 0x00, 0x19, 0x4a ,0x51, 0xa8	- (Repeater node)
FF6	0002591	0x00, 0x12, 0x4b, 0x00, 0x19, 0x4a ,0x51, 0xd7	- (Repeater node)
FF7	0002611	0x00, 0x12, 0x4b, 0x00, 0x19, 0x4a ,0x51, 0xcd	- (Repeater node)
FF8	0002631	0x00, 0x12, 0x4b, 0x00, 0x19, 0x4a ,0x51, 0xb5	- (Repeater node)

Table 3-1: Current node configuration on the lab setup.

3-2 Communication infrastructure

In Section 2-4, the communication protocol WCB was already introduced. Using WCB, ETC can be performed on the cyber physical lab setup over wireless. Each node in the networked control system (NCS) has a specific function and each node must be connected in a specific position in the network, as this structure must correspond to the structure defined in WCB. The details are shown in in Table 3-1. The structure of the nodes is defined in the file *deployment.c* within the files of WCB. In the current configuration, node 5,6,7 and 8 have the function of a repeater node. This means that they are there to support the network flooding in the current configuration. Because of the possibility to reconfigure each actuator and each set of sensors surrounding a gate, the current repeater nodes can later on be used to control a gate for example, when required in case of other control structures than the one that is aimed for now.

Next to the wireless infrastructure, a wired connection is also included in this case because of research purposes. In a real-world implementation the wired connection would not have to be present, but to be able to conduct experiments this wired USB connection is needed to log the experiments. The USB infrastructure is depicted in Figure 3-4. The center of the of

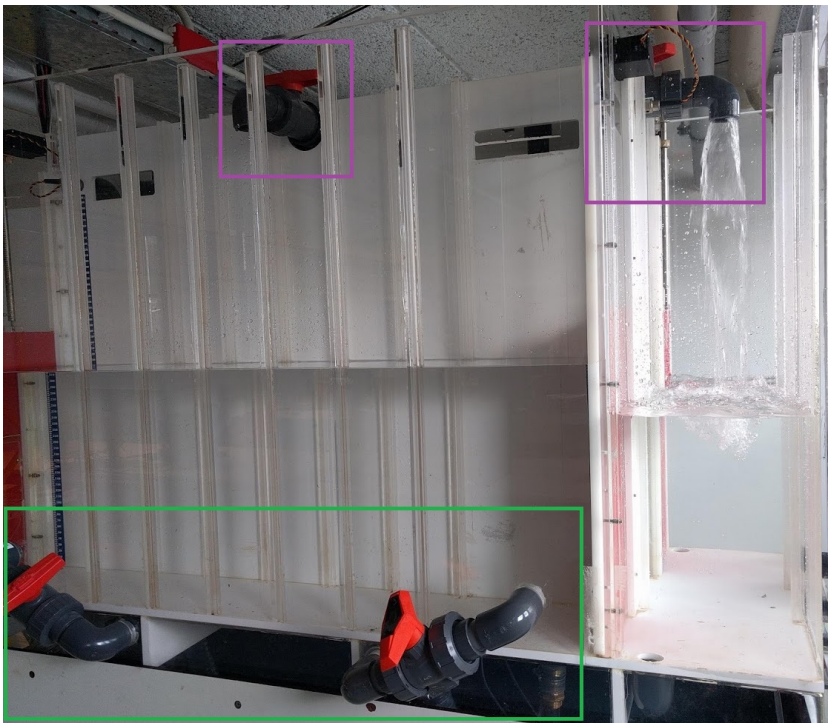


Figure 3-2: The in- and outflow valves shown on the physical lab setup. The inflow valves are shown in the purple boxes in the top of the photo. The outflow valves, which can be used to create off-take load disturbances are shown in the green box in the bottom.

the USB infrastructure is the Raspberry Pi, which is a very small (credit card size) computer running a version of Linux as the operating system (OS). On the Raspberry Pi runs the program to log all signals from the Firefly nodes periodically. To be clear: these signals are not used to perform control, but only to be able to compare the continuous measurements to the zero-order hold (ZOH) signals used in the event-triggering mechanism (ETM). The USB infrastructure is not used to perform control.

A second function of the USB cables is to power the Firefly nodes. Both because the Raspberry Pi does not have 8 USB ports and because the Raspberry Pi cannot supply the current needed by all Firefly nodes combined at peak moments, the (powered) USB hubs are included. The USB hubs can be seen in Figure 3-9.

3-3 Mechanical modifications on the setup

The original lab setup was built approximately twenty years ago. As a result of good care and a sustainable design, it is still in excellent condition. Every aspect of the setup is checked, from which it was clear that new water pumps were needed and some connections had to be re-glued to make sure everything would stay water-tight. The original servo motors still worked, but the DCSC-lab technicians chose to replace the motors in advance of possible failures due to the age of the original servo motors. The new servo motors are the Hitec HS-785HB motors, which is a successor of the originally used motors (Hitec HS-725BB). Both types are sail winches, designed for use in radio-controlled (RC) sail boats.

Figure 3-3: An overview picture of the lab setup in its current state in the DCSC basement.

3-4 Electronics design

The current version of WCB was designed to run on the Zolertia Firefly platform. A wireless control system (WCS) was constructed using a set of Firefly nodes, and to be able to read out the sensors of the plant and to get the actuators moving using signals from the Firefly boards, a completely new set of electronics was designed. A detailed overview of the new set of electronics is included in Appendix B, along with the schematics and designs of the custom PCBs made for the setup. An overview of the new set of electronics, while being mounted on the plant, is shown in Figure 3-5. The insides of the water-proof boxes are shown in Figure 3-6. The individual parts of the electronics are discussed in detail in this section, starting with the Fireboard.

3-4-1 The Fireboard

The Fireboard is one of the main components in the new electronics design of the lab setup, which I designed in close collaboration with DCSC-lab technician Will van Geest. The Fireboard is the name of the PCB holding the Firefly board. It also contains an analog to digital converter (ADC), and additional components and connectors needed to let the Firefly node interface with two sensors and one actuator of the setup. The Fireboard is shown on the left side of Figure 3-7, while on the right side of the same figure the PCB is fully assembled and mounted in the custom 3D-printed enclosure. The schematics for this PCB are included in Appendix B-3.

The lab setup was designed such that it can be used for distributed control experiments. Initially, a design was made in which 8 Fireboards were needed. This design was based on a

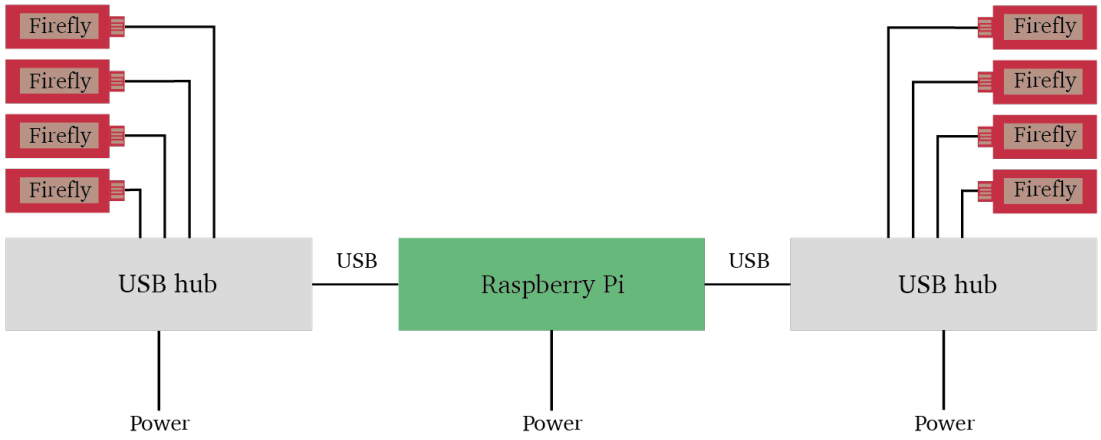


Figure 3-4: The USB infrastructure used to power the Firefly nodes and logging of the input and output signals.

simulation of distant downstream control (DDC), in which a Fireboard would be connected to each sensor at the end of pool (four in total, see Figure 3-1 as a reference), the remaining four Fireboards would be connected to the four movable gates in the system. Later on, this design was changed because the gates act as local flow controllers and to do this, each Fireboard must be able to be connected to one servo motor and two sensors. The PCB can be thought of in two separate circuits, which are both used by the Firefly board; one circuit for reading out two sensors and one circuit to control an actuator.

To control the servo motors on the plant, a $4.8-6.0$ V pulse-width modulation (PWM)-signal is needed. The Firefly operates at 3.3 V, and thus it can only output a 3.3 V PWM-signal. Using a metal–oxide–semiconductor field-effect transistor (MOSFET), the 3.3 V is scaled to a 5 V signal. More details on the construction of this signal are discussed later on in Section 3-7.

One of the demands while designing the new electronics, was that each Firefly must be able to read out two sensors of the lab setup. To be sure of high enough precision, a 16-bit multi-channel ADC is used to read out the sensor values. The sensors can communicate using the RS485 protocol, but in [18] it is recommended to choose for an external ADC to be able to increase the sampling rate, which is exactly what is implemented in the new electronics design. The sensor is supplied with a 9 V input voltage and outputs a $4-20$ mA current. This current is converted to a voltage by using a carefully selected current sensing resistor. The resulting voltage is then supplied to the ADC, which converts the signal to a digital value that is communicated to the Firefly over inter-integrated circuit (I2C), which is a synchronous, serial computer bus commonly used for communication between components on PCBs.

Because the Firefly operates at 3.3 V, I chose to also let the ADC operate on 3.3 V, to be able to create an I2C-bus on which the two boards can communicate without the need of any extra level converters. The ADC is supplied with a regulated 3.3 V source, coming from a linear regulator of type *LF33ABV*, which is on its turn supplied the same 9 V as is used for the sensors connected to the Fireboard. I made the choice to use a linear regulator instead of using a 3.3 V output of the Firefly for two reasons. The first reason was that, in theory, the Firefly was not be able to supply enough current in peak situations, although it was expected

Figure 3-5: Overview of the new electronics on the lab setup. The power supply units and the Raspberry Pi are all located in the white enclosures, preventing any contact with water.

that this situation will never occur in practice. The second reason to use the linear regulator was that the stability of the input voltage of the ADC is directly reflected in the stability of the measurements. By inspection of the schematics of the Firefly, Will and I concluded that the 3.3 V port is also used by on-board components on the Firefly, which implies that there would certainly be fluctuations in this voltage, which would have led to fluctuations in the measurements. To use the linear regulator, some additional capacitors were needed, which are also included in the schematics in Appendix B-3. The connections between the ADC, linear voltage regulator and the Firefly are all shown in the sketch in Figure 3-8, which is very close to the final design.

The input channels of the ADC can only handle a certain maximum voltage (U_{max}), which is relative to the reference voltage supplied to the ADC. To make sure that the input voltages (U_{in}) stay within bounds ($0 < U_{in} < U_{max}$), the input signals are scaled using resistors R3 and R4, shown in Figure 3-8. Because an ADC reference voltage of 3.3 V is used in this case, I chose to scale the sensor signals to the same maximum voltage $U_{max} = 3.3$ V. The sensor output is a $4 - 20$ mA current, which is very common in industry. The ADC can convert voltages, but not currents, which is a second reason to use the current sensing resistors R3 and R4 apart from scaling the signals. Using the resistors, the current is converted to a voltage in the range $0 - U_{max}$ V. The values of the two resistors are the same, as each resistor is used to convert one sensor signal.

The sensor can measure pressures in the range $0 - 200$ mbar $= 0 - 0.2$ bar, which corresponds

Figure 3-6: The insides of the two water proof electronics enclosures on the lab setup, holding the Raspberry Pi and the 5 V and 9 V power supply units used to power the servo motors and sensors respectively.

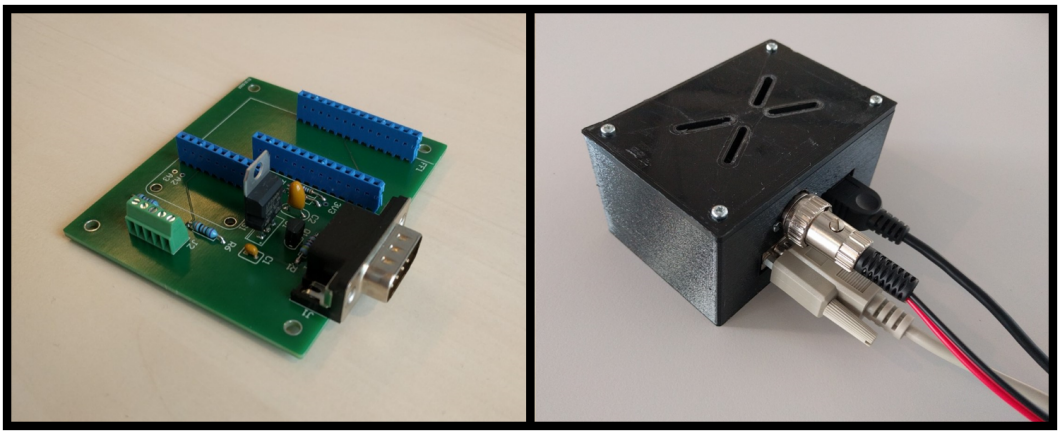


Figure 3-7: The Fireboard PCB without Firefly and ADC on the left. A completely assembled Firebox on the right including a Firefly, ADC-board and all connections.

to a maximum water pressure of 2 mH (meter water column). In our application on the lab setup, there will never be more than 0.7 mH of pressure on the sensor. To be sure we do not exceed U_{max} on any of the input channels of the ADC, I set the maximum water height to $h_{max} = 0.8$ m in the resistor value calculation. The sensor maps its values using a linear mapping from input to output, which implies that the output current of the sensor should never exceed

$$\begin{aligned} I_{max} &= I_{min} + \frac{(I_{max} - I_{min})}{h_{range}} \cdot h_{max} \\ &= 4 \text{ mA} + \frac{(20 - 4) \text{ mA}}{2 \text{ m}} \cdot 0.8 \text{ m} \\ &= 4 + 6.4 = 10.4 \text{ mA}, \end{aligned} \tag{3-1}$$

in our chosen case of $h_{max} = 0.8$ m.

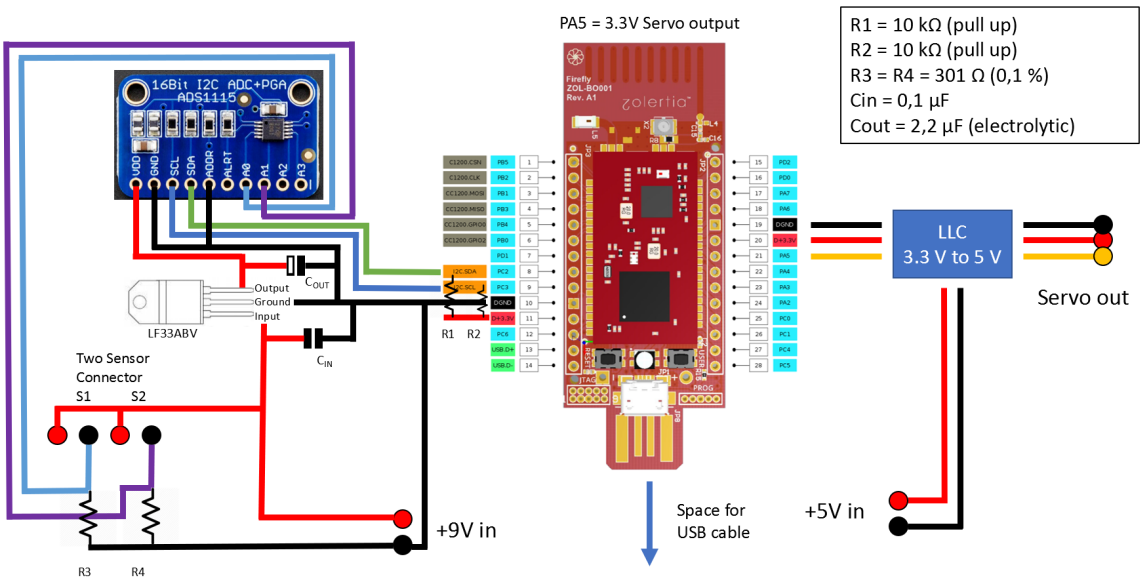


Figure 3-8: An initial sketch of the Fireboard connections, clearly showing the ADC, Firefly and the linear regulator. On the right side, the logic level converter (LLC) is shown, which is the predecessor of the MOSFET in the design.

The optimal resistance of R3 and R4 is then computed as

$$R3 = R4 = \frac{U}{I} = \frac{3.3 \text{ V}}{10.4 \cdot 10^{-3} \text{ A}} \approx 317.3 \ \Omega. \tag{3-2}$$

In close contact with the lab technicians, we chose to use 301 Ω high precision $(0, 1\%)$ resistors. The 301 Ω resistor is the closest to our calculated value, while also being available at the regular suppliers. The resistor value is slightly smaller than the calculated value, but it still results in $U_{in} < U_{max}$ for all water heights on the setup.

The sensors both have a current output as well as a voltage output. While it may seem to make more sense at first to use the voltage output as the ADC can only handle voltages, there are some reasons why I chose to use the current output. The main reason is that the cables from the Power Distribution Box (PDB) to the nodes are 5 meters long, just to be able to locate the nodes one meter apart from each other on the ceiling. Using the voltage output of the sensor, the voltage would most likely drop over this length of wire, and the voltage signal would have been more sensitive to noise over this length. In both cases (current or voltage sensor output), resistors are needed to convert the signal to a signal that is usable by the ADC, which made it clear that there are no benefits in using the voltage output over the current output of the sensor in this case.

3-4-2 Power Distribution Box

The Power Distribution Box (PDB) is the box containing the electronics used to supply power to all servo motors, nodes and sensors (excluding the supplied power over USB to power the Firefly boards). In the PDB, also the servo motor command signals are directed from the

node connectors at the top (see Figure 3-9) to the servo connector, located exactly on the other side of the PDB. During the design phase of the PDB, the option to easily reconfigure the connections from the nodes to the servos is included for the case in which someone wants to apply a different control structure on the setup during future use, for which this option might be needed. The schematics of the Power Distribution Box are included in Appendix B-4.

Figure 3-9: The (white) node connectors at the top of the Power Distribution Box, as well as the USB hubs connecting the the Firefly nodes to the Raspberry Pi.

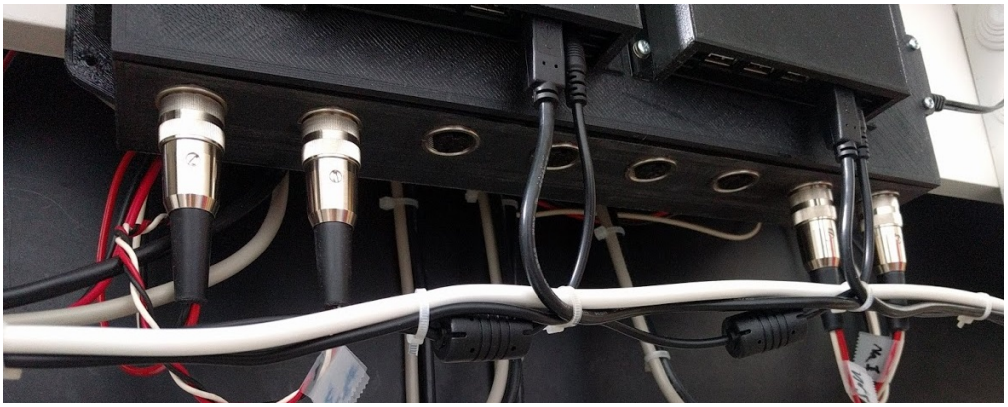


Figure 3-10: The servo motor connections can be reconfigured easily from one node to another by plugging the (shiny) connectors into the female connector corresponding to the node of interest on the bottom side of the Power Distribution Box.

3-5 Sensor testing and corresponding software design

Just as with the actuators, tests were designed and performed for the individual sensors. Because the existing lab-setup was fully revised, new electronics had to be designed and tested.

For the Firefly to communicate with the ADS1115 ADC, an existing (code) library for this ADC was taken as a basis and was modified to work within Contiki-OS. The original library is the Adafruit_ADS1X15[1] library, which was written for the Arduino[2] platform in the C++ programming language. Contiki-OS was programmed in C, which made that the way the original library was structured had to be changed. The language C does not support classes, while the original library made use of classes in a very professional way. A second difference was that Contiki-OS had a predefined structure for additional components like sensors or external ADCs, which had to be used in order to make the interconnection work in the software.

In the end, I had completely changed the structure of the library, while some components of the original library were still used. The license of the original library was still included and the source of the the original library was clearly mentioned in the files. To finally test the sensors, an additional program was created and supplied with a custom Makefile to compile it within the Contiki-OS. To support the open-source community, the custom library and the corresponding program used to read out the ADS1115 ADC values were shared publicly on GitHub [3].

Figure 3-11: Left: The pressure sensor test setup designed for testing individual sensors. Right: The initial 3D-design created in SolidWorks.

The sensors read-outs are first tested on a self-built lab setup, shown in Figure 3-11. The test setup was first designed in SolidWorks. A 3D rendered image of this design is shown on the

[1] https://github.com/adafruit/Adafruit_ADS1X15
[2] https://www.arduino.cc/
[3] https://github.com/jacoblont/ads1115-contiki-os-firefly

right side of Figure 3-11. The pressure sensor is clamped to the wooden board using a custom 3D-printed clamp and the sensor was attached to a transparent hose. In this transparent hose, a water column can be created by filling the hose from the top with water. Next to the transparent hose, a ruler was glued to the board so the digital ADC output values can be easily compared to the height of the water column in meters. After creating the software to read out the sensor values, this same test setup was requested by the DCSC lab technicians to test all the remaining sensors, which was done rapidly using this setup.

3-6 Sensor value mapping

Before the sensor values can be used for control, a conversion is needed. Using the new electronics on the lab setup, the measurements will no longer be converted to a value in mH by the sensor directly. This was the case in the previously used configuration, but as we now use an external ADC to read out the sensors for faster sampling, as was recommended by the research group we got the lab setup from, this is not the case anymore and a new mapping from sensor value to mH needs to be implemented. The maximum water-height of the pools on the setup is 0.6 m + 0.05 m, where 0.05 m corresponds to the distance between the sensor and the bottom of the pool. The sensor value corresponding to the pressure of 0.6 m + 0.05 m = 0.65 m water was measured to be 21570.

Because the sensors are not perfectly identical, the measurements differ from sensor to sensor when measuring the base value at atmospheric pressure. The best thing to do is to correct each sensor, but an analysis is done for when this sensor specific correction is not performed. The base values (at atmospheric pressure) have been measured for all sensors and are found to lie between 9604 and 9682. The mean value of these two values is 9643, which makes that the maximum deviation from this mean is 39 steps. From the experiments it is derived that each increment of the sensor value corresponds to an increase of approximately 0.054 mm in water height. This means that the maximum deviation without the sensor specific corrections will be approximately $0.054 \cdot 39 \approx 2.30$ mm for the base level measurements.

To compute the water heights real-time, a linear mapping was made from the measured sensor values to the water height in meters. The heights in this calculation are the heights measured from the bottom of the pools. This is why $h_{max} = 0.60$ m, and the minimum water height is taken as $h_{min} = -0.05$ m, as this is the level of the sensor with respect to the bottom of the pools. In practice, $h = 0.05$ m is the lowest the water can get in the pools, as the outflow valves of the pools are located at this height, however, computing the mean over a larger range generally gives a more accurate result, which is why the sensor height was used to compute the ratio of meter per step instead of the bottom of the pool.

$$\begin{aligned} h_i \text{ [mH]} &= min\left(p - p_{min},\ 0\right) \cdot \frac{h_{max} \text{ [mH]} - h_{min} \text{ [mH]}}{p_{max} - p_{min}} \\ &= min\left((p - 9643),\ 0\right) \cdot \frac{0.60 \text{ [mH]} - (-0.05) \text{ [mH]}}{21570 - 9643}, \end{aligned} \tag{3-3}$$

where p is the current (pressure) measurement output of the ADC. To make sure that no sensor-value will be negative after subtracting the minimum sensor value corresponding to atmospheric pressure, the minimum of the base value is set to 0.

Even though the sensors are relatively expensive industrial grade sensors, I recommend that extra measurements are taken and a comparison is performed to check if this map is accurate for all sensors on the setup, or that a sensor specific map should be implemented for more accurate results in future projects.

3-7 Actuator signal design

Movable gates are used to control the water levels of the individual pools of the plant. The gates can be set to a specific height in order to create the required amount of flow from one pool to the next. The controller of the system computes control signals, but these control signals are not directly suited to control the servo motors that are used to move the gates up and down. Instead, the control signal is mapped to a set of values, which are then used to control the servo motor. In the original manual for the lab setup [18] (written in 2001), it is mentioned that the position of the servo motors can be specified in 256 steps. A quick experiment showed that the servo motors that are currently on the plant have a range of approximately $4\frac{3}{4}$ full circles, which corresponds to a total range of $4\frac{3}{4} \cdot 360^\circ = 1710$ degrees. These values are all acquired experimentally because no documentation can be found online for the servo motors that are currently used. Even contacting the manufacturer of the actuators did not give any results because the company has not produced these exact actuators anymore for some years.

The movable gates have a total travel from top (fully open) to bottom (fully closed) of 22.4 cm. The gates are driven using a rack and pinion construction, in which the pinion is attached to the servo motor and the rack to the gate. Using the acquired numbers, it is possible to determine the precision of this assembly. The actuator precision, denoted by p_a is computed as

$$p_a = \frac{22.4}{256}\frac{\text{cm}}{\text{step}} = 0.0875\frac{\text{cm}}{\text{step}} < 0.9\frac{\text{mm}}{\text{step}}, \tag{3-4}$$

which shows that it should be possible to move the gates by less than 1 mm per step, in theory.

Based on what is written in [18], some friction on the gates should be expected. The structures of the gates are cleaned during the revision of the setup and all gates slide smoothly in the slots. However, the expectation is that there will still be some friction present as a result of the water pressure acting on the gates, which may have an impact on how accurately the gates can be controlled. To which extend this really is a problem, can only be tested in practice.

To make sure that the actuators can operate in their full range, the actual signals coming from the electronics have been checked beforehand in the lab. Using an oscilloscope, the frequency and the PWM duty cycle have been inspected. A 3.3 V PWM-signal is generated on the Firefly, which is converted to a 5 V signal on the Fireboard. In this conversion process, the PWM-signal is inverted, which is counteracted by pre-inverting the signal in the software running on the Firefly. During the inspection of the signals, it was also found that the PWM-duty cycle generated by the servo-library in the Contiki-OS[4] does not follow the general timing standards for servo motors. Because of this, the constants `SERVO_CONF_MIN_VAL` and `SERVO_CONF_MAX_VAL` (defined in the file contiki/platform/zoul/dev/servo.h) have to be

[4] https://github.com/contiki-os/contiki

adjusted and set to 14400 and 33600 respectively in the header (.h) file corresponding to the protocol used for the experiments. These values correspond to minimum and maximum values of approximately 0.9 ms and 2.1 ms of pulse duration each cycle. The theoretical pulse durations for the signal are 1.0 ms and 2.0 ms, but the used values are advised by the DCSC lab technicians because these will make sure that the actual minimum and maximum of the (analog) actuators are reached. An example mapping between motor position and pulse duration is shown in Figure 3-12 for a servo motor with a range of $0° - 180°$. An optional change is to set the constant `SERVO_CONF_MAX_DEGREES` to the previously computed 1710°, in order to do the mapping to servo-positions at the lowest level possible, which should give the smallest rounding errors and thus the most accurate map.

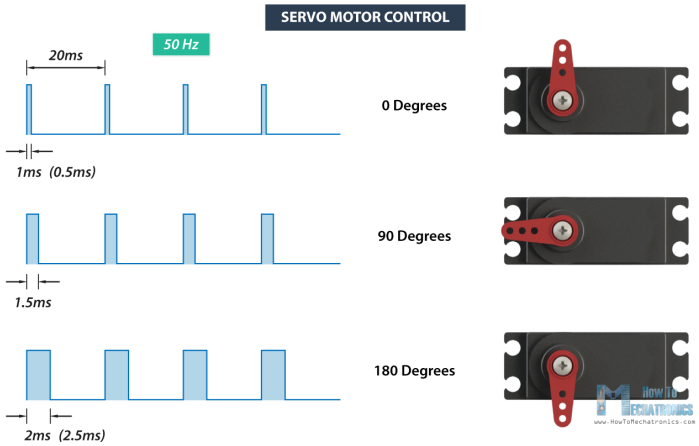


Figure 3-12: Visualization of the PWM-signal for 180°-range servo motor. Image from [19].

An important thing to note is the minimal time between updates of actuator signals. During the lab-tests of the servo motors, it was observed that the servo motors need a significant amount of time to get to the required position. For a full sweep of the range, so basically a step response from $r_{min} = 0°$ to $r_{max} = 1710°$, approximately 10 seconds were needed, when no external forces are acting on the actuator. It is very important to be aware of this time-delay, because fast changes of actuator signals will, because of this time-delay, not always be reflected in the actual gate positions. On the other side, because the system dynamics of WISs are in general not very fast, as can be seen later on from the time-delays in Table 6-1, this actuator time-delay is not expected to be a limiting factor in terms of control performance.

Chapter 4

Event-triggered control design

This chapter explains the control design procedure and the design of the event-triggering mechanism (ETM) for a real-world water irrigation system (WIS), located in Australia, whose parameters are provided in [5]. This set of parameters is used to prove that the constructed models and the created software work as intended. The second reason why these parameters are used, is because the lab setup was not ready yet at the time this part of the project was carried out, because of which it was not possible to do parameter identification at that point for the lab setup. The chapter starts of with the construction of a set of distributed controllers, followed by an explanation on the discrete-time implementation of the controllers. Subsequently, the ETM design is explained and the required math is performed to prove stability properties of the closed loop periodic event-triggered control (PETC) system.

4-1 Distributed control design

Based on the literature survey [9] at the start of this project, it was concluded that it would be most interesting to construct an H_∞-based optimal controller in distributed form for the WIS. The main sources from literature used to actually design the continuous-time output-feedback dynamic controller are [5] and [3]. Using *Theorem 1.* from [5], a set of distributed controllers is constructed using an LMI-based approach to solve the H_∞-problem for a specific value of $\gamma > 0$, where γ denotes the H_∞-gain of the closed-loop system $H(G, \hat{K})$;

$$\gamma = \|H(G, \hat{K})\|_\infty = \sup_{\omega \in \mathbb{R}} \bar{\sigma}(H(G, \hat{K})(j\omega)), \tag{4-1}$$

in which $\bar{\sigma}(\cdot)$ denotes the maximum singular value of a matrix.

$\hat{K}$ denotes the interconnection of distributed controllers $\hat{K} = (\hat{K}_1, \ldots, \hat{K}_N)$ that stabilises the interconnection of weighted generalized pools $G = (G_1, \ldots, G_N)$, following the notation of [5]. The weighted generalized pool models (G_i) are approximately realized with the following

finite-dimensional state-space model, which is constructed using a Padé approximation of the delay associated to each of the specific pools

$$\begin{bmatrix} \dot{x}_i \\ w_i \\ z_i \\ y_i^K \end{bmatrix} = \begin{bmatrix} A_i^{tt} & A_i^{ts} & B_i^{tn} & B_i^{tu} \\ A_i^{st} & A_i^{ss} & B_i^{sn} & B_i^{su} \\ C_i^{tz} & C_i^{sz} & D_i^{zn} & D_i^{zu} \\ C_i^{ty} & C_i^{sy} & D_i^{yn} & D_i^{yu} \end{bmatrix} \begin{bmatrix} x_i \\ v_i \\ n_i \\ u_i^K \end{bmatrix}$$

$$:= \left(\begin{array}{cccc|c|ccc|c} 0 & \frac{1}{\alpha_i} & \frac{-1}{\alpha_i} & 0 & \frac{-1}{\alpha_i} & 0 & \frac{-1}{\alpha_i} & 0 & 0 \\ 0 & \frac{-2}{\tau_i} & \frac{4}{\tau_i} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{\kappa_i\phi_i}{\rho_i} & \frac{\kappa_i\phi_i}{\rho_i} \\ 0 & 0 & 0 & \frac{-1}{\rho_i} & 0 & 0 & 0 & \frac{\kappa_i(\rho_i-\phi_i)}{\rho_i^2} & \frac{\kappa_i(\rho_i-\phi_i)}{\rho_i^2} \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right) \begin{bmatrix} x_i \\ v_i \\ n_i \\ u_i^K \end{bmatrix}, \tag{4-2}$$

where $x_i = (y_i, \Delta_i, u_i, \Omega_i)^T$ is the local state, of which Ω_i corresponds to the loop-shaping pole at $s = -1/\rho_i$ and the sub-state Δ_i corresponds to the pole in the Padé approximation of the delay [5]. The plant output (the water height in pool$_i$) is denoted by y_i, while u_i is the control signal (the output of the local controller K_i, which includes both $\hat{K}_i$ and the shaping weight W_i), and v_i is the outflow of the pool$_i$ into pool$_{i+1}$. The vector of exogenous signals is denoted by $n_i := (r_i, d_i, q_i)^T$, in which q_i is used to model flow uncertainties over gate$_i$, d_i represents the off-take load disturbances and r_i the pool specific reference set-points. The signal u_i^K is the output of the local controller $\hat{K}_i$. The signals are depicted in Figure 4-1. The loop shaping parameters are denoted by κ_i, ϕ_i, and ρ_i, and the specific function of these individual variables is explained later on when considering the structure of the loop-shaping weight, which is used to tune the set of robust controllers. The parameters α_i, τ_i, and φ_i are pool specific parameters, denoting the area, time delay and the most dominant wave frequency, respectively, of pool$_i$. The values for α_i, τ_i, and φ_i used in this case are given in Table 4-1 and are based on [5].

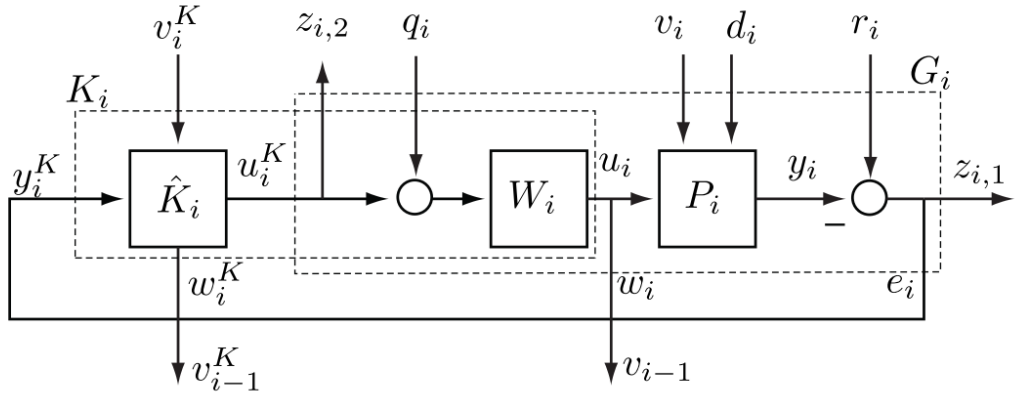


Figure 4-1: Localized portion of a controlled channel. Figure from [5].

i	τ_i	α_i	φ_i
1	4 mins	6492 m^2	0.48 rad/min
2	2 mins	2478 m^2	1.05 rad/min
3	4 mins	6084 m^2	0.48 rad/min
4	4 mins	5658 m^2	0.48 rad/min
5	6 mins	7650 m^2	0.42 rad/min

Table 4-1: Pool specific parameters for simulations for pool$_i$: time delay τ_i, surface area α_i and most dominant wave frequency φ_i, from [5].

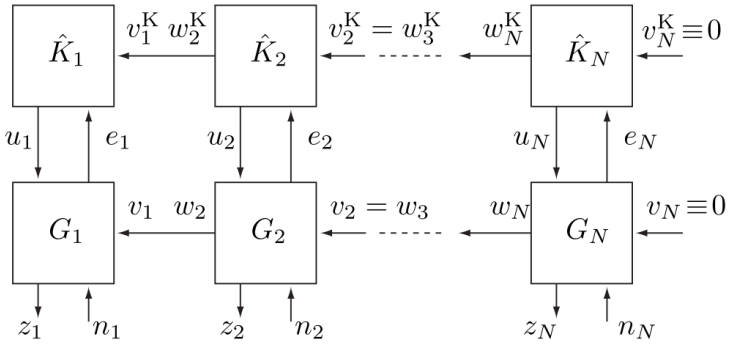


Figure 4-2: Distributed control structure for synthesis. Figure from [5].

The pool interconnection is shown in Figure 4-2 and is characterized by $w_i = v_{i-1}$, which signal is shown in Figure 4-1 in more detail. The interconnection of the distributed controllers ($\hat{K}_i$) is denoted by $w_i^K = v_{i-1}^K$, and is also depicted in these figures. The design trade-off in this case corresponds to managing the effect of the exogenous signals n_i on the vector performance signals z_i, consisting of:

$$z_i := \begin{bmatrix} y_i^K = e_i \\ u_i^K \end{bmatrix} \tag{4-3}$$

The loop-shaping weight W_i has the same structure as the shaping weight in [5], which has the same structure as the fully decentralized controllers used in [2], and it also corresponds to the structure used for optimal centralized control design in [4]. The structure of W_i is

$$W_i(s) = \frac{\kappa_i(1 + s\phi_i)}{s(1 + s\rho_i)}, \tag{4-4}$$

where κ_i is used to set the loop-gain bandwidth. Following [5], the loop-gain bandwidth should be below $1/\tau_i$ rad/min because of the time delay, which is not reflected in the loop-gain $L_i(s) = W_i(s)/s\alpha_i$. Using ϕ_i, phase-lead can be introduced in the cross-over region to reduce the roll-off (the steepness of a transfer function's magnitude plot with frequency) for stability and robustness. Additional roll-off beyond the loop-gain bandwidth is provided by tuning ρ_i to ensure sufficiently low gain at the dominant wave frequency. The weight parameters for the plant used in [5] are shown in Table 4-2. The values for η_i shown in Table 4-2 are used to scale the individual shapes in order to combine them in one plot.

Next, the dynamic output-feedback controller is created by solving the structured H_∞ problem by applying Theorem 1 from [5]. This theorem is a specialization of Theorem 4 in [20], which

i	κ_i	ϕ_i	ρ_i	η_i
1	1.69	113,64	9,97	130
2	6.47	37,17	3,26	223
3	2.37	86.96	7.60	183
4	2.21	96.15	8.47	170
5	1.68	113.64	9.97	153

Table 4-2: Loop-shaping weight parameters.

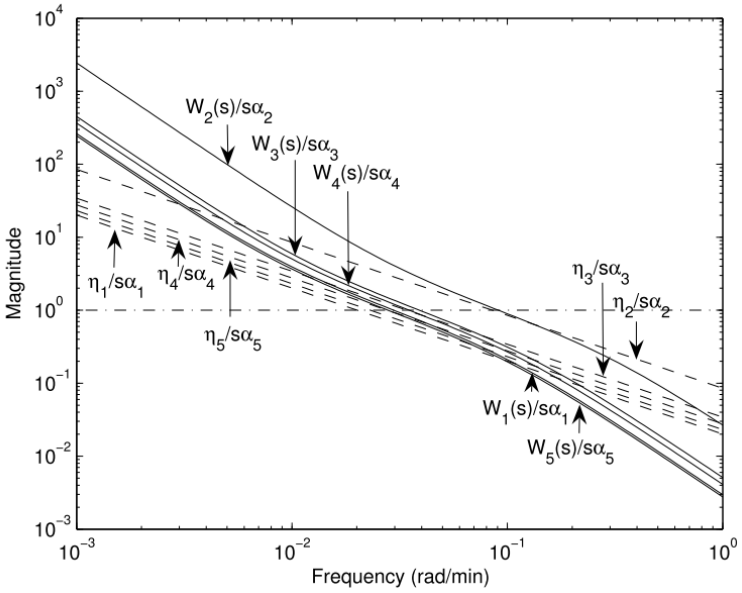


Figure 4-3: The tuned loop gains for the plant from [5].

gives a state-space characterization of the synthesis problem. Application of the theorem leads to the set $\hat{K}$ of distributed controllers with state matrix S_i as shown in Eq. (4-5).

$$\begin{bmatrix} \dot{x}_i^K \\ w_i^K \\ u_i^K \end{bmatrix} = S_i \begin{bmatrix} x_i^K \\ v_i^K \\ y_i^K \end{bmatrix}, \tag{4-5}$$

in which the dimension of x_i^K is equal to the dimension of x_i in Eq. (4-2). The structured H_∞ problem is solved using CVX [21], [22] in MATLAB [23], which is a package for specifying and solving convex programs. The link to the repository with the MATLAB-scripts I created specifically for this purpose is given in Appendix A-1. The H_∞ norm achieved in the end for the controller is $\gamma_{min} = 3.92$. The numerical values of the S_i-matrices are all included in the GitHub repository linked to in Appendix A-1.

4-2 Discrete-time implementation

The main goal of the project in to apply event-triggered control (ETC) to WISs. Based on literature [9] is concluded that PETC [10] is the best choice of ETC implementation for this project, since the controller is implemented on a digital platform and will thus not be continuous-time. To use PETC, a sampling time is chosen and subsequently the periodic (discrete-time) controller is acquired by discretization of the in Section 4-1 designed continuous-time controller. In the following subsections, relevant discretization methods are discussed as well the sampling time selection used for discretization.

4-2-1 Discretization methods

The (distributed) controllers and loop-shaping weights are discretized using the Tustin/bilinear approximation. This method is preferred for discretizing controllers in order to preserve phase properties [8]. The preservation of phase properties is considered important because of the water-level error propagation (WLEP) properties of the set of controllers, which is optimized to handle/avoid specific dominant wave frequencies. The plant models are discretized using the Zero-Order Hold method. All discretizations are performed using the MATLAB function c2d[1].

4-2-2 The sampling period

An important choice in discretization was the sampling period to discretize for, as the result of the discretization depends on the chosen sampling period. A general rule of thumb is to have 3-10 samples per rise-time [8]. The controlled pool having the smallest rise-time was used for sampling-time selection, as this pool has the fastest response. The rule of thumb used in this process, was to aim for approximately 8 samples per rise-time. As the rise-times also differ per controller, the sampling period should be revised or at least be checked each time after the controller is tuned. The exact code used in this process was uploaded to a GitHub repository, which is further elaborated on and linked to in Appendix A-1.

4-2-3 Model selection for sampling-time analyses

Control simulations were done for multiple plants from literature, from which was concluded that the first- and third-order models showed very similar responses. The similarity in behaviour of the first- and third order models was also presented in [24]. In Figure 4-4 these similarities are also directly clear, using the model parameters of the five pools presented in [2]. In [2] was shown that the third order model used in this study corresponds very well to the actual behaviour of WISs. Based on all these findings, it was clear that the first order model captures enough of the water-dynamics to be used for rise-time determination. The first-order model of pool_i, without the corresponding time-delay, has the form

$$P_i = \frac{1}{\alpha_i s}, , \tag{4-6}$$

[1] https://nl.mathworks.com/help/control/ref/c2d.html

in which α_i again denotes the surface area of pool_i in m^2 as given in Table 6-1.

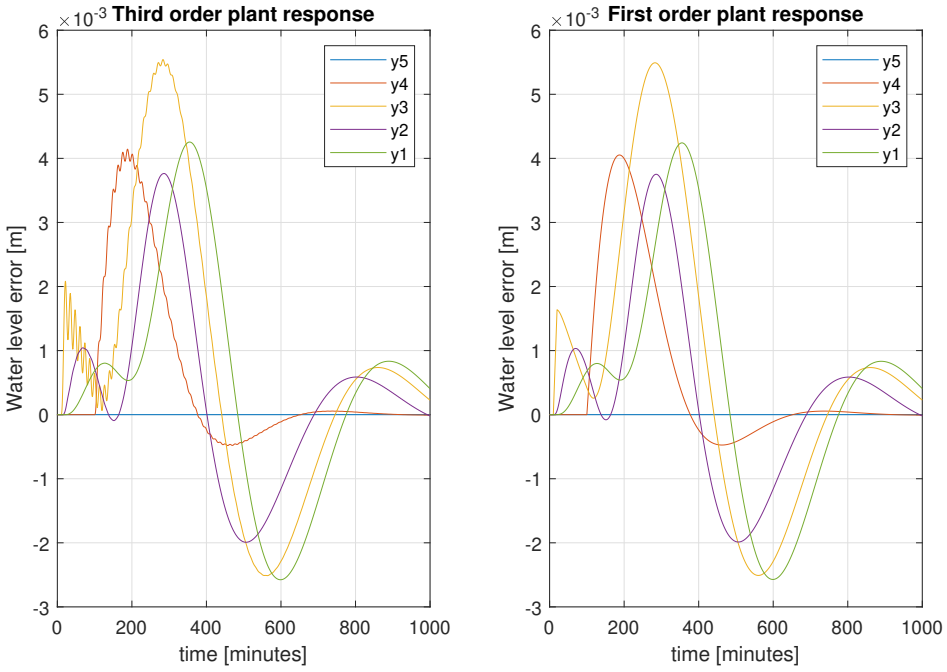


Figure 4-4: Simulation results for both the first and third order model using the five-pool model parameters from [5] clearly shows the similarity in the responses.

4-3 Event-triggered mechanism design

After designing the H_∞-controller based on the ideas presented in [5] and [3], the next step was to combine the constructed optimal controller with ETC. The decentralized ETC design for the dynamic output-based feedback controller was based on [10]. The ETM works as follows: in the network, a distinction was made between sensor nodes and actuator nodes. The sensor nodes periodically request new measurements from their water-height sensor. The new measurement is compared in a predefined way to the most recently transmitted sensor value. The function used for the comparison of the newest sensor value and the most recently transmitted value is called the triggering condition. Each sensor-node has such a local triggering condition, which is checked periodically to see if it is needed for the node to update its states to the controller. When the local triggering condition is satisfied, the node will initiate a network flood in the first available epoch in which all sensor nodes will transmit their most recent measurement to the centralized controller.

Just after the controller has received the current state of the system, the controller will check the control-side triggering function to see if it is needed (in terms of performance and stability) to update the actuator (control) signals. If this is the case, then the controller will initiate a network flood with actuator signal packages for the actuator nodes in the networked control system (NCS).

4-3-1 Output-based PETC description

The considered system is a continuous-time linear time-invariant (LTI) system, of which only the outputs of the system are available for control purposes. The system dynamics can be recast in the form

$$\begin{aligned} \dot{x}^p(t) &= A^p x^p(t) + B^p \hat{u}(t) + B^w w(t) \\ y(t) &= C^p x^p(t), \end{aligned} \tag{4-7}$$

where $\dot{x}^p(t) \in \mathbb{R}^n$ is vector of state derivatives , $x^p(t) \in \mathbb{R}^{n_p}$ is the state vector, $u(t) \in \mathbb{R}^n_u$ denotes the vector of (control) inputs applied to the plant, $w(t) \in \mathbb{R}^{n_w}$ represents the vector of external, unknown disturbances and $y \in \mathbb{R}^{n_y}$ the output of the plant. The matrices A^p, B^p and B^w denote the state, input, and disturbance matrices of the plant respectively. The matrix C^p denotes the plant output matrix. The superscript p is added to distinguish between the states of the plant and the states of the controller, which are indicated by the superscript c. Following the exact notation of [10], the plant is controlled using a discrete-time LTI controller

$$\begin{aligned} x^c_{k+1} &= A^c x^c_k + B^c \hat{y}_k \\ u_k &= C^c x^c_k + D^c \hat{y}_{k-1}, \end{aligned} \tag{4-8}$$

where $x^c_k \in \mathbb{R}^{n_c}$ is the state of the dynamic controller at time-step k, $\hat{y} \in \mathbb{R}^{n_y}$ denotes the input of the controller, and $u \in \mathbb{R}^{n_u}$ is used to denote the output of the dynamic output-feedback controller. The sampling times are defined as $t_k = kh, k \in \mathbb{N}$, with $h > 0$ being the sampling interval. At each sampling time t_k, the (decentralized) triggering condition will check the sampled values and determine if the newly sampled values contained by $y(t_k)$ and $u(t_k)$ will be transmitted, or not transmitted. When the triggering condition is satisfied and the values are transmitted, then the most recently transmitted (and thus the most recently received) values $\hat{u}$ and $\hat{y}$ are updated (assuming a 100% succes rate in data transmission). The updates of these signals are depicted in Figure 4-5.

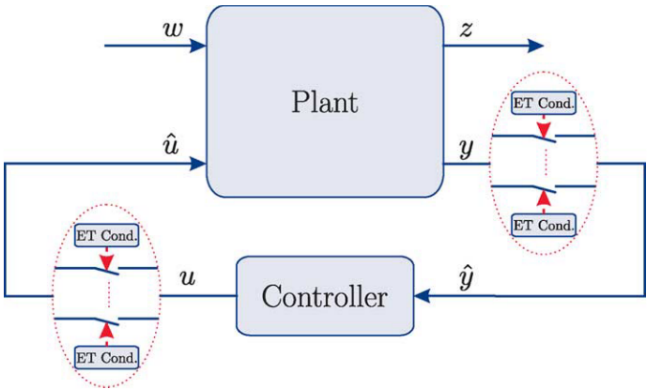


Figure 4-5: Decentralized event-triggered control schematic. Figure from [10].

The controller's state update x^c_{k+1} is based on $\hat{y}_k$. This state update should in practice occur in the interval $(t_k, t_{k+1}]$, $k \in \mathbb{N}$; however, in the model the convention was adopted that for $(t_k, t_{k+1}]$, $k \in \mathbb{N}$ it holds that

$$x^c(t) = x^c_{k+1} = A^c x^c_k + B^c \hat{y}_k, \tag{4-9}$$

in which the updates of the controller state takes place right after t_k, $k \in \mathbb{N}$, where it should be observed that x^c is a left-continuous signal [10]. The control signal $u(t_k) = u_k$ at time t_k is computed based on $\hat{y}_{k-1}$, which will be equal to $\hat{y}(t_k)$, being the most recently transmitted output of the plant, as we define for $t \in (t_k, t_{k+1}]$

$$\hat{y}(t) = \hat{y}_k \quad \text{and} \quad \hat{u}(t) = \hat{u}_k. \tag{4-10}$$

It should be clear that both $\hat{u}$ and $\hat{y}$ are also left-continuous, $\hat{y}_k := \lim_{t \downarrow t_k} \hat{y}(t)$, and $\hat{u}_k := \lim_{t \downarrow t_k} \hat{u}(t)$. Because of this,

$$u(t_k) = C^c x^c(t_k) + D^c \hat{y}(t_k), \quad k \in \mathbb{N}. \tag{4-11}$$

4-3-2 Decentralized event-triggering conditions

In this case, not all states of the plant are measurable. In fact, only the outputs can be measured, which implies than it is not possible to implement a triggering condition that uses the full state of the system. The available signals are the control signals contained in u_k and the output signals, y_k. These signals are the ones that will be transmitted at time t_k when the triggering condition is satisfied. We define

$$v = \begin{bmatrix} y \\ v \end{bmatrix} \in \mathbb{R}^{n_v} \quad \text{and} \quad \hat{v} = \begin{bmatrix} \hat{y} \\ \hat{v} \end{bmatrix} \in \mathbb{R}^{n_v}, \tag{4-12}$$

where $n_v := n_y + n_u$. The entries in v and $\hat{v}$ must be grouped into N nodes, such that the entries in v and $\hat{v}$ corresponding to node $j \in \{1, \dots, N\}$ are denoted by v^j and $\hat{v}^j$, respectively.

In cases where full state information is available, a triggering condition of the form

$$\|\hat{x}(t_k) - x(t_k)\| > \sigma \|x(t_k)\| \tag{4-13}$$

is often used. An elaboration on this type of triggering conditions is included in the literature survey [9], which I have written in the first phase of this project. For the decentralized output-feedback case, [10] proposes the following description of the triggering condition and the update of the signals contained in $\hat{v}$:

$$\hat{v}^j(t) = \begin{cases} v^j(t_k), & \text{if } \left\| v^j(t_k) - \hat{v}^j(t_k) \right\| > \sigma_j \left\| v^j(t_k) \right\| \\ \hat{v}^j(t_k), & \text{if } \left\| v^j(t_k) - \hat{v}^j(t_k) \right\| \leq \sigma_j \left\| v^j(t_k) \right\|, \end{cases} \tag{4-14}$$

for $t \in (t_k, t_{k+1}], k \in \mathbb{N}$, and $\sigma_j \geq 0, j \in \{1, \dots, N\}$ are constants which are tuned later on in the process, in which it is made sure that stability of the (closed loop) PETC system is preserved. Eq. (4-14) expresses that at each sampling time t_k, $k \in \mathbb{N}$, each node computes the difference $v^j(t_k) - \hat{v}^j(t_k)$. If the absolute value of this difference is too large (based on the value of σ_j), node j must transmit its signals contained in $v^j(t_k)$ and subsequently $\hat{v}^j$ is updated using the corresponding values immediately after time t_k. To implement decentralized triggering conditions, each node checks its own local triggering condition, which is in this case defined as

$$\left\| v^j(t_k) - \hat{v}^j(t_k) \right\| > \sigma_j \left\| v^j(t_k) \right\|. \tag{4-15}$$

When a local triggering condition is satisfied, the corresponding node will initiate a network flood in the first available epoch after the time the trigger happened. During this network flood, all nodes will update their current state to the centralized controller. It is possible that multiple nodes trigger before the same epoch, but this is no problem when using Wireless Control Bus (WCB), as the nodes will register when an epoch happens and the variable used to flag that a network flood has to be initiated will be cleared when a node receives the message that a network flood is already initiated by an other node in the network. WCB is designed such that all nodes are able to send their updates to the controller before the epoch is over.

The local triggering condition shown in Eq. (4-15) can be rewritten to fit the form of a quadratic triggering condition as

$$\xi^\top(t_k)Q_j\xi(t_k) > 0, \tag{4-16}$$

where ξ is defined as

$$\xi = \begin{bmatrix} x^p \\ x^c \\ \hat{v} \end{bmatrix} = \begin{bmatrix} x^p \\ x^c \\ \hat{y} \\ \hat{u} \end{bmatrix}, \tag{4-17}$$

using a properly chosen Q_j, $j \in \{1, \dots, N\}$.

In our case, we mainly use this theory for output-based PETC from [10] in order to be able to prove global exponential stability (GES) and to compute the $\mathcal{L}_2$-gain of the closed loop system from the disturbance signals $\bar{w}$ to the outputs y, by creating one artificial node, containing all actual nodes of our system. This method simplifies the analysis of the interconnection, after which [25] is used to decentralize the actual triggering condition. By choosing to combine the nodes to one artificial node, using all signals contained in the vector $v = [y^\top u^\top]^\top$, we are in essence setting the number of nodes to $N = 1$ artificially.

In [10], a set of diagonal matrices $\Gamma_\mathcal{J} \in \mathbb{R}^{n_v \times n_v}$ is defined for $\mathcal{J} = \{1, \dots, N\}$ to be able to write 4-15 in the form of 4-17 as

$$\Gamma_\mathcal{J} = \text{diag}(\gamma_\mathcal{J}^1, \dots, \gamma_\mathcal{J}^{n_y+n_u}) = \text{diag}(\Gamma_\mathcal{J}^y, \Gamma_\mathcal{J}^u), \tag{4-18}$$

with $\Gamma_\mathcal{J}^y \in \mathbb{R}^{n_y \times n_y}$ and $\Gamma_\mathcal{J}^u \in \mathbb{R}^{n_u \times n_u}$. The set $\mathcal{J}$ defines the set of nodes that will perform a measurement update to the controller in case of a trigger. Because all nodes send an update in WCB when a trigger happens, it is sound to use one artificial node in this approach. In case of a trigger, with $N = 1$, $\Gamma_\mathcal{J}$ is equal to

$$\Gamma_1 = \begin{bmatrix} \Gamma_1^y & 0 \\ 0 & \Gamma_1^u \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}, \tag{4-19}$$

with $I \in \mathbb{R}^{5\times5}$ being the identity matrix. This is the case because all the signals contained in the vector $v = [y^\top u^\top]^\top$ correspond to the single node. By setting $N = 1$, the set $\mathcal{J} = \{1, \dots, N\}$ reduces to $\mathcal{J} = \{1\}$. This may seem very trivial, especially since the set of nodes is equal to $\{1\}$, but what may be not immediately clear is that there is also the possibility of $\Gamma_\emptyset$, in case none of the nodes in the set triggers. As there are no signal updates corresponding to this non-triggering-event, $\Gamma_\emptyset$ is defined as

$$\Gamma_\emptyset = \begin{bmatrix} \Gamma_\emptyset^y & 0 \\ 0 & \Gamma_\emptyset^u \end{bmatrix} = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{(n_u+n_y)\times(n_u+n_y)}, \tag{4-20}$$

By defining the matrices C and D as

$$C := \text{diag}(C^p, C^c), \text{ and } D := \begin{bmatrix} 0 & 0 \\ D^c & 0 \end{bmatrix}, \tag{4-21}$$

we can write

$$\begin{aligned} \left\| v^j(t_k) \right\| &= \left\| \Gamma_j [C \;\; D] \xi(t_k) \right\|, \textit{and} \\ \left\| v^j(t_k) - \hat{v}^j(t_k) \right\| &= \left\| \Gamma_j [C \;\; D - I] \xi(t_k) \right\| \end{aligned} \tag{4-22}$$

for $k \in \mathbb{N}$, where Γ_j is used to define which inputs and outputs defined in the vector v belong to node j. In this case all inputs and all outputs belong to the single artificial node, because of which

$$\Gamma_j = I, \tag{4-23}$$

with $I \in \mathbb{R}^{n_v \times n_v}$ being the identity matrix. Using these definitions, Eq. (4-15) can be written in the quadratic form of Eq. (4-17) with

$$Q_j = Q := \begin{bmatrix} (1-\sigma_j)C^\top C & (1-\sigma_j)C^\top D - C^\top \\ (1-\sigma_j)D^\top C - C & (D-I)^\top (D-I) - \sigma_j D^\top D \end{bmatrix}. \tag{4-24}$$

The subscript j corresponding to the index of the node can be dropped from Q_j as there is only one node in our approach. Using the definition of $\Gamma_\mathcal{J}$, the updates of $\hat{v}$ just after time t_k can be compactly written as

$$\begin{aligned} \hat{v}^+(t_k) &= \Gamma_{\mathcal{J}(\xi(t_k))} v(t_k) + (I - \Gamma_{\mathcal{J}(\xi(t_k))}) \hat{v}(t_k) \\ &= [\Gamma_{\mathcal{J}(\xi(t_k))} C \quad \Gamma_{\mathcal{J}(\xi(t_k))} D + I - \Gamma_{\mathcal{J}(\xi(t_k))}] \xi(t_k), \end{aligned} \tag{4-25}$$

which in case of a trigger, with $N = 1$, becomes

$$\hat{v}^+(t_k) = [C \quad D] \xi(t_), \tag{4-26}$$

where for $\xi \in \mathbb{R}^{n_\xi}$

$$\mathcal{J}(\xi) := \{ j \in \{1\} \mid \xi^\top Q_j \xi > 0 \}. \tag{4-27}$$

Note that, just as with $\Gamma_\mathcal{J}$, there is also the possibility of $\mathcal{J}(\xi) = \mathcal{J}_\emptyset$, in case none of the nodes in the set triggers.

4-4 The impulsive system model

In order to use the impulsive system approach, as presented in Section V.B. of [10], the PETC system needs to be recast in impulsive system form. To obtain an impulsive system model of

the system, one should observe that for the definition of $\mathcal{J}(\xi)$ in Eq. (4-27), for $k \in \mathbb{N}$, we have $\mathcal{J}(\xi(t_k)) = \mathcal{J}$ if and only if

$$\xi(t_k)^\top Q \xi(t_k) > 0, \;\; j \in \mathcal{J} \quad \text{and} \quad \xi(t_k)^\top Q \xi(t_k) \leq 0, \;\; j \in \mathcal{J}^c, \tag{4-28}$$

where $\mathcal{J}^c := \{1, \ldots, N\} \backslash \mathcal{J}$ denotes the complement of any arbitrary set $J \subseteq \{1, \ldots, N\}$. Using this definition, the impulsive system model can be obtained as

$$\begin{bmatrix} \dot{\xi} \\ \dot{\tau} \end{bmatrix} = \begin{bmatrix} \bar{A}\xi + \bar{B}w \\ 1 \end{bmatrix}, \quad \text{when } \tau \in [0, h] \tag{4-29a}$$

$$\begin{bmatrix} \xi^+ \\ \tau^+ \end{bmatrix} = \begin{bmatrix} J_{\mathcal{J}}\xi \\ 0 \end{bmatrix}, \quad \text{when } \tau = h, \quad \xi^\top Q \xi > 0, \;\; j \in \mathcal{J} \quad \text{and} \quad \xi^\top Q \xi \leq 0, \;\; j \in \mathcal{J}^c \tag{4-29b}$$

$$z = \bar{C}\xi + \bar{D}w, \tag{4-29c}$$

where $z \in \mathbb{R}^{n_z}$ is a performance output, defined by properly chosen matrices $\bar{C} \in \mathbb{R}^{n_z \times n_\xi}$ and $\bar{D} \in \mathbb{R}^{n_z \times n_w}$. The performance outputs are chosen based on the performance outputs of the generalized plant as

$$z = \begin{bmatrix} y \\ \hat{u} \end{bmatrix} = \begin{bmatrix} C_y^p x^p \\ \hat{u} \end{bmatrix} \tag{4-30}$$

which is realized by defining the matrices $\bar{C}$ and $\bar{D}$ as

$$\bar{C} = \begin{bmatrix} C_y^p & 0 & 0 \\ 0 & 0 & I \end{bmatrix}, \quad \bar{D} = \begin{bmatrix} 0 \end{bmatrix} \tag{4-31}$$

Where the zero blocks in the middle column of $\bar{C}$ are $0 \in \mathbb{R}^{n_z \times (n_{x^c} + n_{\hat{y}})}$. The identity matrix is in this case defined as $I \in \mathbb{R}^{n_{\hat{u}} \times n_{\hat{u}}}$, as it should only pass through the signals contained in the vector $\hat{u} \subset \xi$. The matrix C_y^p is defined as

$$C_y^p = diag(C_{y1}^p, \ldots, C_{y5}^p), \quad \text{where } C_{yi}^p = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \tag{4-32}$$

as the first state of the plant state vector $x_i^p \subset \xi$ is equal to y_i.

The matrices $\bar{A} \in \mathbb{R}^{n_\xi \times n_\xi}$, $\bar{B} \in R^{n_\xi \times n_w}$ and $J_\mathcal{J} \in R^{n_\xi \times n_\xi}$ are defined as

$$\bar{A} = \begin{bmatrix} A^p & 0 & 0 & B^p \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4-33a}$$

$$\bar{B} = \begin{bmatrix} B_w \\ 0 \\ 0 \\ 0 \end{bmatrix}, \text{ and} \tag{4-33b}$$

$$\begin{aligned} J_\mathcal{J} &= \begin{bmatrix} I & 0 & 0 & 0 \\ B^c \Gamma^y_\mathcal{J} C^p & A^c & B^c(I - \Gamma^y) & 0 \\ \Gamma^y_\mathcal{J} C^p & 0 & (I - \Gamma^y_\mathcal{J}) & 0 \\ 0 & \Gamma^u_\mathcal{J} C^c & \Gamma^u_\mathcal{J} D^c & I - \Gamma^u_\mathcal{J} \end{bmatrix} \\ J_1 &= \begin{bmatrix} I & 0 & 0 & 0 \\ B^c C^p & A^c & 0 & 0 \\ C^p & 0 & 0 & 0 \\ 0 & C^c & D^c & 0 \end{bmatrix}, \quad J_\emptyset = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & A^c & B^c & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}. \end{aligned} \tag{4-33c}$$

4-4-1 Combined controller description

To apply the impulsive system approach described in [10], the set of distributed controllers is combined into a single (discrete-time) state-space description of the form described by Eq. (4-8). The set of models for the individual pools should also be combined to a single state-space formulation, described by Eq. (4-7). The distributed dynamic output-feedback controllers were initially created in the form

$$\begin{bmatrix} \dot{x}^K_i \\ w^K_i \\ u^K_i \end{bmatrix} = S_i \begin{bmatrix} x^K_i \\ v^K_i \\ y^K_i \end{bmatrix}, \tag{4-34}$$

of which the matrix S_i can be partitioned as

$$S_i = \left[\begin{array}{c|c} S^A_i & S^B_i \\ \hline S^C_i & S^D_i \end{array}\right]. \tag{4-35}$$

Next, these sub-matrices were then used to create a continuous-time state-space model:

$$\begin{aligned} \dot{x}^K_i(t) &= S^A_i \cdot x^K_i(t) + S^C_i \cdot \begin{bmatrix} v^K_i(t) \\ y^K_i(t) \end{bmatrix} \\ \begin{bmatrix} w^K_i(t) \\ u^K_i(t) \end{bmatrix} &= S^C_i \cdot x^K_i(t) + S^D_i \cdot \begin{bmatrix} v^K_i(t) \\ y^K_i(t) \end{bmatrix}, \end{aligned} \tag{4-36}$$

which is subsequently discretized using the Tustin method (as explained in Section 4-2-1) to obtain the discrete-time version of the individual controllers. The discrete-time version of the individual controllers can be described by

$$\begin{aligned} x_i^K(k+1) &= S_i^{A_k} \cdot x_i^K(k) + S_i^{B_k} \cdot \begin{bmatrix} v_i^K(k) \\ y_i^K(k) \end{bmatrix} \\ \begin{bmatrix} w_i^K(k+1) \\ u_i^K(k) \end{bmatrix} &= S_i^{C_k} \cdot x_i^K(k) + S_i^{D_k} \cdot \begin{bmatrix} v_i^K(k) \\ y_i^K(k) \end{bmatrix}. \end{aligned} \tag{4-37}$$

The impulsive system description requires that only the outputs of the plant are used as inputs for the controller. At this point, both the plant outputs $y_i^K(k)$ and the interconnection signals $v_i^K(k) = w_{i+1}^K$ are inputs of the individual controllers. The output $w_i^K(k)$ can be added as a discrete-time state by splitting up the matrices of this state-space description as

$$S_i^{B_k} = \begin{bmatrix} S_i^{Bv_k} & S_i^{By_k} \end{bmatrix}, \quad S_i^{C_k} = \begin{bmatrix} S_i^{Cw_k} \\ S_i^{Cu_k} \end{bmatrix}, \quad S_i^{D_k} = \begin{bmatrix} S_i^{Dvw_k} & S_i^{Dyw_k} \\ S_i^{Dvu_k} & S_i^{Dyu_k} \end{bmatrix}, \tag{4-38}$$

which allows us to define the controllers having only u_i^K as the output and having v_i^K as a state, by defining

$$\begin{aligned} \begin{bmatrix} x_i^K(k+1) \\ w_i^K(k+1) \end{bmatrix} &= A_i^c \cdot \begin{bmatrix} x_i^K(k) \\ v_i^K(k) \end{bmatrix} + B_i^c \cdot y_i^K(k) \\ u_i^K(k) &= C_i^c \cdot \begin{bmatrix} x_i^K(k) \\ v_i^K(k) \end{bmatrix} + D_i^c \cdot y_i^K(k), \end{aligned} \tag{4-39}$$

Because the interconnection between the distributed controllers is defined as $v_i^K = w_{i+1}^K$, $v_N^K := 0$, these signals still need to be present when combining the individual controllers in order to create one centralized controller. In the centralized case, the sub-controllers are now able to access the signals $v_i^K = w_{i+1}^K$ from the combined state vector, in which the superscript K is from now on dropped for ease of notation,

$$x^c(k) = x_k^c = \begin{bmatrix} x_k^1 \\ w_k^1 \\ x_k^2 \\ w_k^2 \\ \vdots \\ x_k^5 \\ w_k^5 \end{bmatrix}, \quad \text{where } x_k^i := \begin{bmatrix} x_k^{i1} \\ x_k^{i2} \\ x_k^{i3} \\ x_k^{i4} \end{bmatrix} = x_i^K(k), \text{ for i} = \{1, \ldots, N\} \tag{4-40}$$

instead of receiving these signals as an input. In this equation, x_k^c denotes the state of the combined controller at time-step k and $w_i^K(k) = w_k^i$. Because $w_1^k = v_0$ is not used as an input and because the signal $v_N := 0$, I chose to use w_1^K as input $v_N = v_5^k = w_k^6$ for the sub-controller related to pool_5, by setting $w_1^K = 0 \ \forall \ x_1^K, v_1^K, y_1^K$. I chose to do this to preserve a consistent structure in the combined state vector.

The combined controller can now be written in the form of Eq. (4-8)

$$\begin{aligned} x^c_{k+1} &= A^c x^c_k + B^c \hat{y}_k \\ u_k &= C^c x^c_k + D^c \hat{y}_{k-1}, \end{aligned}$$

where the combined controller matrices A^c, B^c, C^c, and D^c are defined as

$$\begin{aligned} A^c &= \begin{bmatrix} A^c_{row1} \\ \vdots \\ A^c_{row5} \end{bmatrix}, \quad B^c = diag(B^c_1, \dots, B^c_5), \\ C^c &= \begin{bmatrix} C^c_{row1} \\ \vdots \\ C^c_{row5} \end{bmatrix}, \quad D^c = diag(D^c_1, \dots, D^c_5). \end{aligned} \tag{4-41}$$

Where $B^c_i = S_i^{B_{y_k}}$, and $D^c_i = S_i^{D_{yu_k}}$. Based on the structure of the state vector x^c_k, the block rows of the matrix A^c are defined as

$$\begin{aligned} A^c_{rowi} &= \begin{bmatrix} 0_{i1} & S_i^{A_k} & 0_{i2} & S_i^{B_{v_k}} & 0_{i3} \\ 0_{i4} & S_i^{C_{w_k}} & 0_{i5} & S_i^{D_{vw_k}} & 0_{i6} \end{bmatrix}, \text{ for } i = \{1, \dots, 4\}, \\ A^c_{row5} &= \begin{bmatrix} 0_{i7} & S_5^{B_{v_k}} & 0_{i3} & S_5^{A_k} & 0_{i8} \\ 0_{i9} & S_5^{D_{vw_k}} & 0_{i6} & S_5^{C_{w_k}} & 0_{i10} \end{bmatrix} \end{aligned} \tag{4-42}$$

where $0_{i1} \in \mathbb{R}^{n_{x^i} \times (i-1)\cdot(n_{x^i}+n_{w^i})}$, $0_{i2} \in \mathbb{R}^{n_{x^i} \times (n_{x^i}+n_{w^i})}$, $0_{i3} \in \mathbb{R}^{n_{x^i} \times (n_{A^c}-p)}$, where p is equal to the cumulative width of all other columns in the row. Furthermore, $0_{i4} \in \mathbb{R}^{n_{w^i} \times (i-1)\cdot(n_{x^i}+n_{w^i})}$, $0_{i5} \in \mathbb{R}^{n_{w^i} \times (n_{x^i}+n_{w^i})}$, $0_{i3} \in \mathbb{R}^{n_{w^i} \times (n_{A^c}-p)}$. The remaining zero-blocks corresponding only to A^c_{row5} are defined as $0_{i7} \in \mathbb{R}^{n_{x^i} \times n_{A^c_1}}$, $0_{i8} \in \mathbb{R}^{n_{x^i} \times n_{w^i}}$, $0_{i9} \in \mathbb{R}^{n_{w^i} \times n_{A^c_1}}$, $0_{i10} \in \mathbb{R}^{n_{w^i} \times n_{w^i}}$. In a similar way the rows of C^c are defined as

$$\begin{aligned} C^c_{rowi} &= \begin{bmatrix} 0_{i4} & S_i^{C_{u_k}} & 0_{i5} & S_i^{D_{vw_k}} & 0_{i6} \end{bmatrix}, \text{ for } i = \{1, \dots, 4\}, \\ C^c_{row5} &= \begin{bmatrix} 0_{i9} & S_5^{D_{vu_k}} & 0_{i6} & S_5^{C_{w_k}} & 0_{i10} \end{bmatrix} \end{aligned} \tag{4-43}$$

4-4-2 Combined plant description

As a basis for the impulsive system description, the generalized plant description (Eq. (4-2)) is taken as a basis for the combined plant model. The goal is to write the dynamics of the set of pools into the following form (initially presented in Eq. (4-7))

$$\begin{aligned} \dot{x}^p(t) &= A^p x^p(t) + B^p \hat{u}(t) + B^w \bar{w}(t) \\ y(t) &= C^p x^p(t), \end{aligned} \tag{4-44}$$

where the matrix $A^p \in \mathbb{R}^{n_p \times n_p}$ is the state evolution matrix of the plant (indicated by the superscript p), $\dot{x}^p \in \mathbb{R}^{n_p}$ is the state derivative, while $x^p \in \mathbb{R}^{n_p}$ is the state vector. The matrix $B^p \in \mathbb{R}^{n_p \times n_{\hat{u}}}$ is the input matrix, $\hat{u} \in \mathbb{R}^{n_{\hat{u}}}$ is the vector of inputs, $B^w \in \mathbb{R}^{n_p \times n_{\bar{w}}}$ is the disturbance input matrix, and $\bar{w} \in \mathbb{R}^{\bar{w}}$ is the vector of external disturbances. The vector $y \in \mathbb{R}^{n_y}$ denotes the vector of outputs and $C^p \in \mathbb{R}^{n_y \times n_{x^p}}$ is the plant output matrix.

The state derivative vector $\dot{x}^p$ and the state vector x^p are defined in a similar way as the vector x^c is defined for the controller, as

$$\dot{x}^p = \begin{bmatrix} \dot{x}_1^p \\ \dot{x}_2^p \\ \vdots \\ \dot{x}_5^p \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \dot{x}_{11}^p \\ \dot{x}_{12}^p \\ \dot{x}_{13}^p \\ \dot{x}_{14}^p \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \dot{x}_{51}^p \\ \dot{x}_{52}^p \\ \dot{x}_{53}^p \\ \dot{x}_{54}^p \end{bmatrix} \end{bmatrix}, \qquad x^p = \begin{bmatrix} x_1^p \\ x_2^p \\ \vdots \\ x_5^p \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} x_{11}^p \\ x_{12}^p \\ x_{13}^p \\ x_{14}^p \end{bmatrix} \\ \vdots \\ \begin{bmatrix} x_{51}^p \\ x_{52}^p \\ x_{53}^p \\ x_{54}^p \end{bmatrix} \end{bmatrix}, \tag{4-45}$$

The vectors $\hat{u}$ and $\bar{w}$ are defined as

$$\hat{u} = \begin{bmatrix} \hat{u}_1^K \\ \vdots \\ \hat{u}_5^K \end{bmatrix}, \quad \bar{w} = \begin{bmatrix} \bar{w}_1 \\ \vdots \\ \bar{w}_5 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} d_1 \\ q_1 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} d_5 \\ q_5 \end{bmatrix} \end{bmatrix}, \tag{4-46}$$

where d_i models an off-take load-disturbance and q_i is used to model uncertainty in the flow over gate$_i$ as in [5]. The interconnection between the individual pools is very similar to the interconnection of the distributed controllers as shown in the previous subsection.

To be able to define the plant dynamics in the required form, it is needed to split up certain matrices used in the description of the generalized plant,

$$\begin{bmatrix} \dot{x}_i \\ w_i \\ z_i \\ y_i^K \end{bmatrix} = \begin{bmatrix} A_i^{tt} & A_i^{ts} & B_i^{tn} & B_i^{tu} \\ A_i^{st} & A_i^{ss} & B_i^{sn} & B_i^{su} \\ C_i^{tz} & C_i^{sz} & D_i^{zn} & D_i^{zu} \\ C_i^{ty} & C_i^{sy} & D_i^{yn} & D_i^{yu} \end{bmatrix} \begin{bmatrix} x_i \\ v_i \\ n_i \\ u_i^K \end{bmatrix}, \quad \text{where } n_i = \begin{bmatrix} r_i \\ d_i \\ q_i \end{bmatrix},$$

which is fully introduced in Eq. (4-2).

Because only the entries d_i and q_i are used from the vector n_i, the matrices related to the vector n_i are split up as follows

$$\begin{aligned} B_i^{tn} = \begin{bmatrix} B_i^{tnr} & B_i^{tnd} & B_i^{tnq} \end{bmatrix} &= \begin{bmatrix} B_i^{tnr} & B_i^{tnw} \end{bmatrix} \\ B_i^{sn} &= \begin{bmatrix} B_i^{snr} & B_i^{snw} \end{bmatrix} \\ D_i^{zn} &= \begin{bmatrix} D_i^{znr} & D_i^{znw} \end{bmatrix} \\ D_i^{yn} &= \begin{bmatrix} D_i^{ynr} & D_i^{ynw} \end{bmatrix} \end{aligned} \tag{4-47}$$

Expanding the state update equation for pool$_i$ gives

$$\dot{x}_i = A_i^{tt} x_i + A_i^{ts} v_i + B_i^{tnw} \bar{w}_i + B_i^{tu} u_i^K, \tag{4-48}$$

where $v_i = w_{i+1}$, of which w_{i+1} is computed as

$$\begin{aligned} w_i &= A_i^{st} x_i + A_i^{ss} v_i + B_i^{sn} n_i + B_i^{su} u_i^K \\ w_i &= A_i^{st} x_i \end{aligned} \tag{4-49}$$

as can be seen by looking at the numerical description of the generalized plant in Eq. (4-2). This means that x_i can be computed using only the entries from the combined state-vector and the input $u_i^K = \hat{u}_i$. To do this, the matrix A^p is defined as

$$A^p = \begin{bmatrix} A_1^{tt} & [0\ 0\ A_1^{ts}\ 0] & 0 & 0 & 0 \\ 0 & A_2^{tt} & [0\ 0\ A_2^{ts}\ 0] & 0 & 0 \\ 0 & 0 & A_3^{tt} & [0\ 0\ A_3^{ts}\ 0] & 0 \\ 0 & 0 & 0 & A_4^{tt} & [0\ 0\ A_4^{ts}\ 0] \\ 0 & 0 & 0 & 0 & A_5^{tt} \end{bmatrix}, \tag{4-50}$$

where

$$[0\ 0\ A_i^{ts}\ 0] = A_i^{ts} \cdot A_{i+1}^{st}, \quad A_{i+1}^{st} = A_i^{st} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \ \forall\ i \in [1, \ldots, N-1]. \tag{4-51}$$

The input matrix B^p is defined as

$$B^p = diag(B_1^{tu}, B_2^{tu}, B_3^{tu}, B_4^{tu}, B_5^{tu}). \tag{4-52}$$

Splitting up the matrices as in Eq. (4-47) immediately enables us to compose the input matrix B^w as

$$B^w = diag(B_1^{tnw}, B_2^{tnw}, B_3^{tnw}, B_4^{tnw}, B_5^{tnw}). \tag{4-53}$$

As is stated before, the first state of the vector x_i^p corresponds to the water height y_i of pool_i. Because of this and because of the structure of the combined plant state vector, defined in Eq. (4-44), the output matrix C^p is defined as

$$C^p = diag(C_{y1}^p, C_{y2}^p, C_{y3}^p, C_{y4}^p, C_{y5}^p), \quad \text{where } C_{yi}^p = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \text{ for } i = 1, \ldots, 5 \tag{4-54}$$

to finally get the vector of water heights

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_5 \end{bmatrix} \tag{4-55}$$

as the output of the combined plant description.

4-5 Implementing decentralized event-triggered control

To prove stability all nodes of the system were combined in one artificial node. It was possible to do this, because in WCB a trigger of one node, initiates a network flood, which is in essence the same as all nodes triggering at the same time. However, in practice, there are multiple individual nodes on the setup and these nodes do not have access to the current measurements

of their neighbouring nodes. This makes that it is not possible to use a centralized triggering condition in physical implementation.

The centralized triggering condition

$$\|v(t_k) - \hat{v}(t_k)\| > \sigma \|v(t_k)\|, \tag{4-56}$$

in which $v = [y^\top u^\top]^\top$ and $\hat{v} = [\hat{y}^\top \hat{u}^\top]^\top$ at sampling time t_k and σ is a triggering parameter, can be decentralized using the method described in Section 2.4 of [13], which was originally proposed in [25]. Eq. (4-56) can be written in the form

$$\sum_{i=1}^{n} (v_i(t_k) - \hat{v}_i(t_k))^2 - \sigma^2 v_i^2(t_k) > 0, \tag{4-57}$$

which implies

$$\bigvee_{i=1}^{n} \left((v_i(t_k) - \hat{v}_i(t_k))^2 - \sigma^2 v_i^2(t_k) > \theta_i \right), \tag{4-58}$$

as long as the condition $\sum_{i=1}^{n} \theta_i = 0$ is satisfied. The $\vee$ in the equation denotes the 'logical or', and is thus used to set the equation true whenever one of the nodes triggers, after which all nodes will send their most recent measurement to the controller using a network flood. The set of θ_i's can both be designed offline, but it can also be a dynamically adjusted online. Design strategies for both cases are presented in [25].

4-6 Stability of the closed-loop PETC system

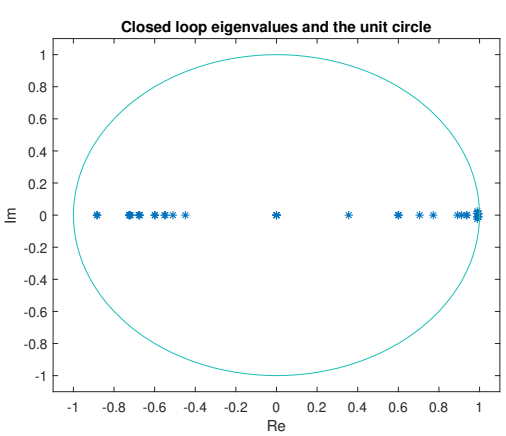
4-6-1 Nominal stability of the system with periodic control

The ultimate goal in terms of stability analysis is to prove GES for the closed loop system. As a start of the stability analysis, nominal stability is checked. In theory the robust-controller should provide nominal stability by design, but the closed loop system can become unstable when discretizing the controller. Because of this, nominal stability is checked after discretization using a carefully selected sampling period.

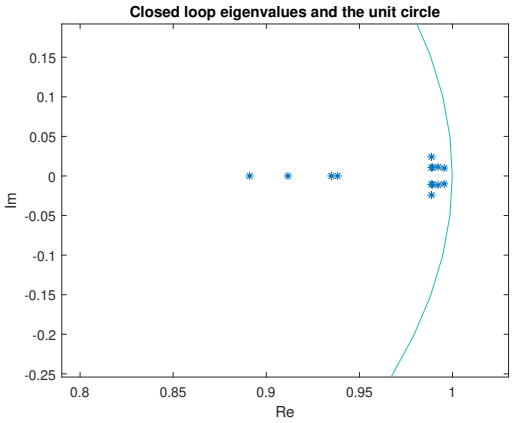
Nominal stability of the closed loop system is checked before including the ETM in the loop. As mentioned earlier, it provides a starting point of stability analysis to make sure stability is preserved in the process of discretization. The main goal of the project is to apply a discrete-time controller to a continuous-time plant, but to check nominal stability, the continuous-time plant model is discretized using the same sampling period h, after which the eigenvalues of the closed-loop system are analyzed. The closed-loop eigenvalues are shown in Figure 4-6a. Clearly all eigenvalues are inside the open unit circle, which is a necessary and sufficient condition for nominal stability and GES.

4-6-2 Exponential stability and $\mathcal{L}_2$-gains

It is possible to show GES and a finite $\mathcal{L}_2$-gain for event-triggered controlled systems in certain cases. A detailed description on how to prove this for output-feedback controlled systems, as is the case here, is presented in [10]. Now that we have the impulsive system description,



(a) Closed loop eigenvalues with respect to the unit circle.



(b) Zoomed in view of the eigenvalues at the right side of the unit circle.

Figure 4-6: The closed loop eigenvalues of the discrete-time system without ETC.

derived in Section 4-4, *Theorem V.2* of [10] can be applied to try and guarantee GES and provide the $\mathcal{L}_2$-gain from the disturbance signals w to the output signals z of the closed loop system. Unfortunately applying the theorem did not provide the existence of the positive definite matrices and parameters described in the theorem in [10]. This does not necessarily mean that the system is not GES or that a finite $\mathcal{L}_2$-gain does not exist for this system, but we are not able to prove it using this theorem. The theorem states that if one can find a solution to the set of linear matrix inequalities (LMIs), then the closed-loop PETC system is GES with a guaranteed convergence rate that is greater or equal than the value of the input parameter ρ, and the $\mathcal{L}_2$-gain is smaller or equal than the value of the input parameter for a specified value of triggering parameter σ. The input parameters were tuned manually, and a lot of parameter sets are tried, but I was not able to get a satisfactory result. I performed experiments with the parameter ρ ranging from 10^{-3} to 10^{-13}, σ ranging from 0.0001 to 0.05, and λ ranging from $2 \cdot 10^5$ to $2 \cdot 10^{12}$. At the start of this series of tests, it seemed like it was possible to get a solution, but when looking at the equations closely, Gabriel and I concluded that I forgot to include one LMI constraint that was related to the empty set. From that point on, I have not been able to get a solution for the set of LMIs anymore. We tried both the standard CVX solver SDPT3, as well as SeDuMi, which is a second semi-definite programming solver that comes with CVX.

At the point when the second LMI was not yet implemented, Gabriel and I did notice that the resulting matrices that seemed to be solving the problem were ill-conditioned and when substituting the results into the constraint equations manually, the matrices did not result in positive-definite matrices, which was a requirement specified using the set of LMIs. This outcome was most-likely due to numerical issues, that may have to do with the large number of variables and the combination of very large and very small numbers used in the problem, which possibly makes that the problem is hard to scale by the solvers.

Chapter 5

Numerical experiments

In this chapter, the results of the numerical experiments are discussed. The chapter starts with a discussion on disturbance rejection, which shows the effects of small modifications on the triggering condition to improve the the behaviour of the event-triggering mechanism (ETM) around the origin. The plant parameters are the same as presented in Chapter 4, representing a real water irrigation system (WIS), and the triggering conditions used in the experiments are of the exact same form as they should be implemented on the lab setup when using Wireless Control Bus (WCB). The behaviour of the closed-loop periodic event-triggered control (PETC) system is tested using a set-point disturbance of 0.1 m on pool_4 for multiple values of the triggering parameters σ and ϵ, as is explained in Section 5-1 .The chapter is finalized with the presentation and the discussion of numerical experiments for decentralized event-triggered control (ETC). The experiments show the achievable amount of reduction in communication very clearly by comparing the number of triggering instants to the amount of communications instants that would have been needed when periodic control would have been applied. The results also show the reduction in control performance as a result of the reduction in communication.

5-1 Disturbance rejection

5-1-1 Change of coordinates

An assumption that is not always clearly indicated in literature, is the change of coordinates in PETC. PETC was designed to stabilize a system to the origin. This implies that when a system is being controlled to a reference set-point other than the origin (zero), a coordinate change has to be applied to make set this reference set-point the origin in the triggering function, which is explained in Section 2.5 of [13]. In terms of implementation, this implies that one should use the set-point error as an argument for the triggering function, instead of the actual output of the plant. It is important to implement this change of coordinates to prevent the water-levels with a relative large magnitude reference set-point from dominating

the triggering function. Not implementing the change of coordinates results in constant oscillations of outputs corresponding to reference set-points having a relatively small magnitude, compared to the other reference set-points, in centralized triggering functions. In Figure 5-1 these oscillations can be clearly observed, whereas these oscillations were not present in the exact same experiment but with the change of coordinates, which is shown in Figure 5-2. The figures show results for a centralized triggering case, using $\sigma = 0.10$. Although the oscillations are not present anymore, the downside is that there twice as much triggers. This is most likely due to the small norms of the signals around the origin.

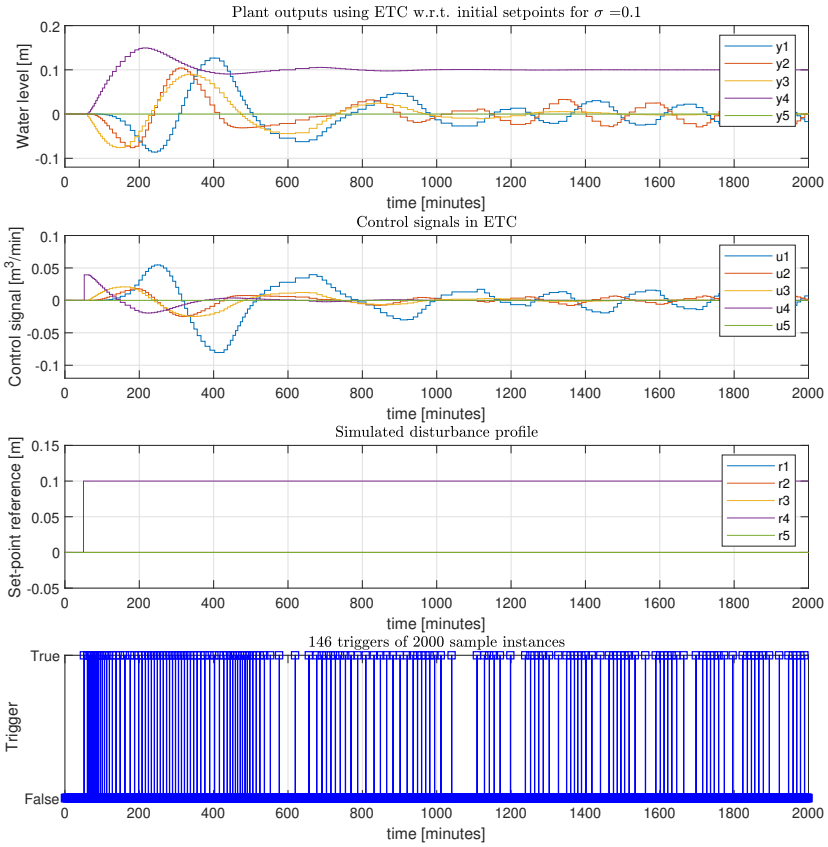


Figure 5-1: The plant outputs, control inputs, disturbance profile and the triggering instants using centralized event-triggered control for a set-point disturbance on pool 4 for $\sigma = 0.10$ using the actual plant outputs in the triggering function, leading to constant oscillations.

5-1-2 Dead band around the origin for the plant outputs

Initially, the local triggering condition for each of the sensor nodes was implemented as

$$\|y_j(t_k) - \hat{y}_j(t_k)\| > \sigma \|y_j(t_k)\|, \tag{5-1}$$

in which $y_j(t_k)$ is the water height of pool_j at time t_k and σ is a triggering parameter. Because a lot of triggers occurred around the origin, a dead band was included, like it was proposed in [13]. In this dead-band oscillations with minimal magnitudes (small waves in the water) can occur around the set-point, while the ETM is not constantly triggered. In practice, these

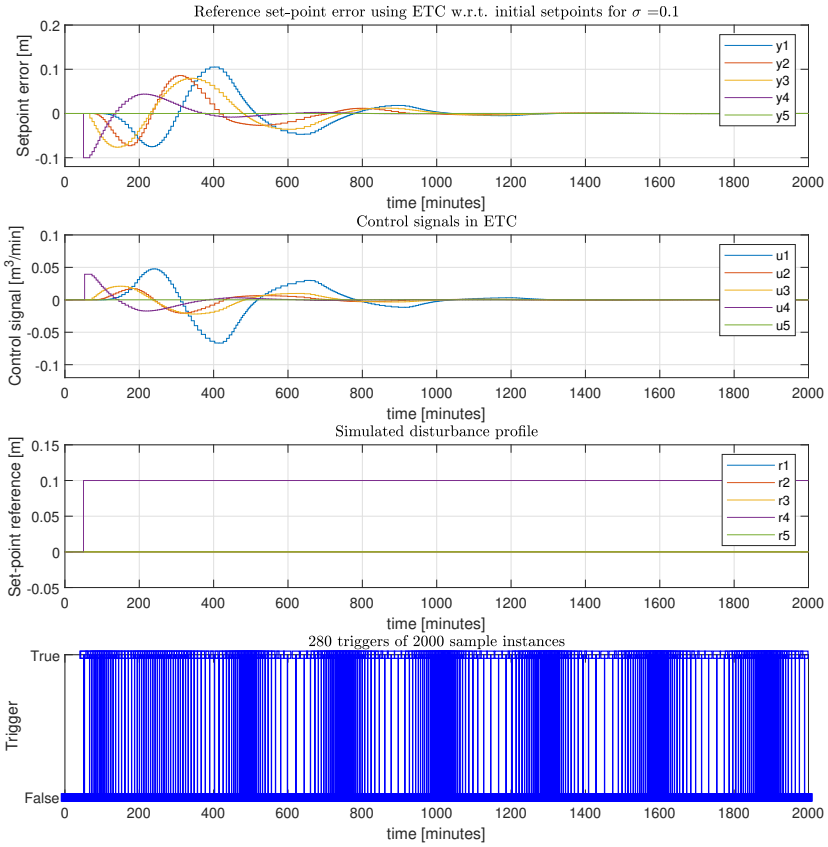


Figure 5-2: The plant outputs, control inputs, disturbance profile and the triggering instants using centralized event-triggered control for a set-point disturbance on pool 4 for $\sigma = 0.10$.

small waves could also be formed by the wind blowing over the water channel. The dead-band around the origin was implemented as

$$\|y_j(t_k) - \hat{y}_j(t_k)\| - \epsilon > \sigma \|y_j(t_k)\|, \tag{5-2}$$

which can be rewritten as

$$\|y_j(t_k) - \hat{y}_j(t_k)\| - \sigma \|y_j(t_k)\| > \epsilon, \tag{5-3}$$

to clearly show the dead band of $\pm\epsilon$ in which the triggering function is not satisfied.

The dead band was not implemented for the controller triggering function. Experiments showed that a dead band on the control signals resulted in a decrease in control performance and more triggers, while the dead band on the plant outputs improved control performance and reduced the number of triggers. I chose to implement the control triggering function for these experiments as

$$\|u(t_k) - \hat{u}(t_k)\| > \sigma \|u(t_k)\|, \tag{5-4}$$

in which

$$u = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}, \quad \hat{u} = \begin{bmatrix} \hat{u}_1 \\ \vdots \\ \hat{u}_N \end{bmatrix}, \tag{5-5}$$

with $N = 5$ denoting the number of pools.

5-2 Decentralized triggering experiments

Because of the large distances in WISs, it is not possible to use a centralized ETM, as the individual signals (the water heights and the control signals) cannot be centralized without the use of communication. Because communication is exactly what needs to minimized, each sensor node has its own ETM. When one of these ETMs trigger, a network flood is initiated and all nodes will send a measurement update to the centralized controller. The exact triggering functions used in these numerical experiments are explained in Section 5-1. In the following subsections, the results of the experiments are shown and discussed. In each case, the results are shown using a plot in which the first subplot shows the water level errors with respect to the corresponding set-points. The second subplot shows the magnitude of the control signals, and the third subplot shows the disturbance profile applied in the experiment. The fourth and final subplot shows the triggering instants at which both the measurements and the control signals are updated. The actual number of triggers is shown in the title of this fourth subplot.

5-2-1 Decentralized triggering: $\sigma = 0.005$

The first experiment is the case with $\sigma = 0.005$ and $\epsilon = 1 \cdot 10^{-6}$, of which the results are shown in Figure 5-3. The water level errors were within 0.1 m and also the magnitude of the control signals were relatively small. There was a total number of triggers of 692 based on 1600 measurements instants, which already gives a 57% reduction in communication.

The number of triggers can be decreased by increasing the dead band parameter ϵ. The effects of increasing ϵ from $\epsilon = 1 \cdot 10^{-6}$ to $\epsilon = 1 \cdot 10^{-4}$, while still using $\sigma = 0.005$ can be clearly seen by comparing Figure 5-3 to Figure 5-4. A decrease in control performance can be observed, but the change did lead to a reduction in communication of 66% compared to the periodic control case by only having 554 triggers.

5-2-2 Decentralized triggering: $\sigma = 0.01$

By increasing σ to $\sigma = 0.01$, the number of triggers is reduced to 498. The behaviour of the system corresponding to this value is shown in Figure 5-5. The magnitude of the signals did increase as a result of increasing σ, however a reduction in communication of 69% is achieved.

5-2-3 Decentralized triggering: $\sigma = 0.05$

The next experiment was done using $\sigma = 0.05$. To keep reasonable control performance, it was needed to decrease the dead band parameter ϵ to $\epsilon = 1 \cdot 10^{-7}$, for which the results are shown in Figure 5-6. The signals did not really increase in magnitude as a result of using these parameters, however the reduction in communication increased to 75%. At this point, a lower density of the triggering instant plot is very clear from the figure.

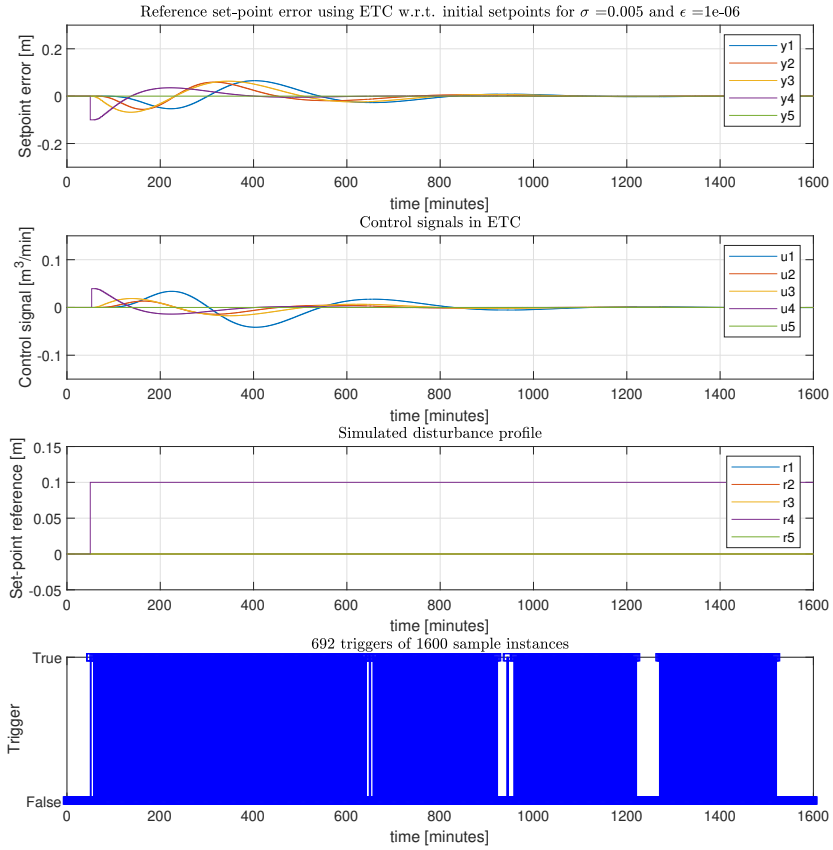


Figure 5-3: The plant outputs, control inputs, disturbance profile and the triggering instants using decentralized event-triggered control for a set-point disturbance on pool 4 for $\sigma = 0.005$ and $\epsilon = 1 \cdot 10^{-6}$.

5-2-4 Decentralized triggering: $\sigma = 0.10$

The final experiment was conducted after selecting $\sigma = 0.10$ and $\epsilon = 1 \cdot 10^{-7}$, for which the results are shown in Figure 5-7. The plot clearly shows that the magnitude of both the outputs of the plant, as well as the magnitude of the control signals increased significantly after I increased the value for σ to $\sigma = 0.10$. While the reduction in communication is at 83%, the plot shows that oscillations start to form again around the steady state, while already having tuned the dead band.

5-3 Conclusions

It is clear from the experiments that a lot of communication resources can be saved, compared to periodic control, by applying decentralized ETC on WISs. By reducing communication between the plant and the controller, we are reducing the amount of information that is available to the controller. This typically leads to reduced control performance, which is also clear from these experiments. In general, the magnitudes of the signals get larger as the amount of communication instants is reduced. The decrease in control performance can also be seen by inspecting the time it takes to stabilize the system, which also grows when

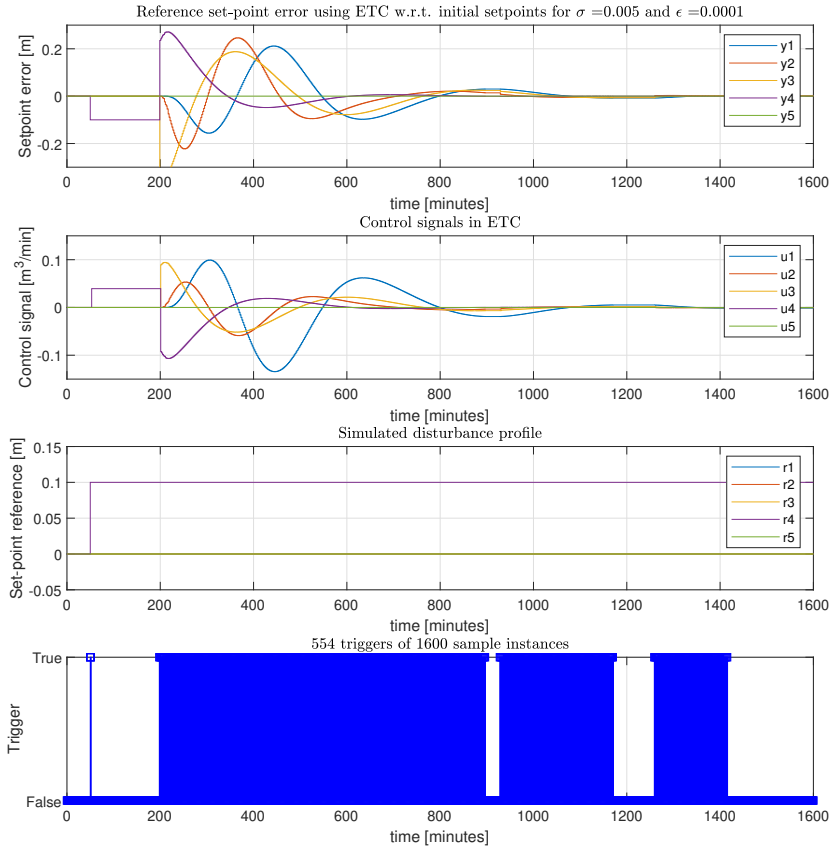


Figure 5-4: The plant outputs, control inputs, disturbance profile and the triggering instants using decentralized event-triggered control for a set-point disturbance on pool 4 for $\sigma = 0.005$ and $\epsilon = 1 \cdot 10^{-4}$.

the number of communications instants is reduced. Tuning ETC always presents a trade-off between the amount of communication and the control performance. Where the optimal point of this trade-off is, depends on the application and the requirements of the application.

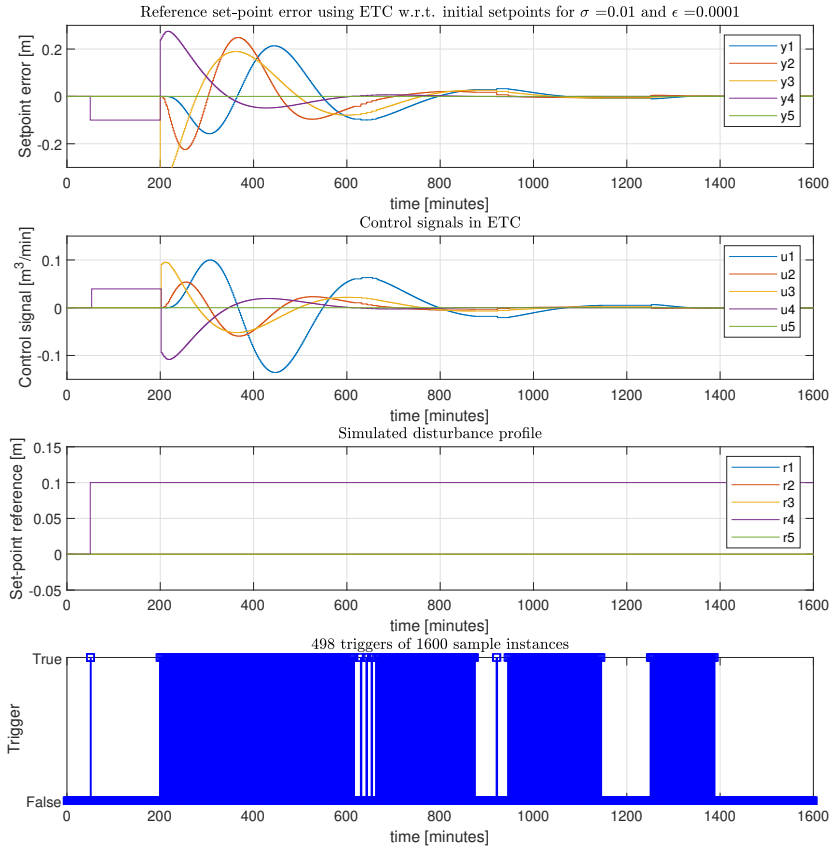


Figure 5-5: The plant outputs, control inputs, disturbance profile and the triggering instants using decentralized event-triggered control for a set-point disturbance on pool 4 for $\sigma = 0.01$ and $\epsilon = 1 \cdot 10^{-4}$.

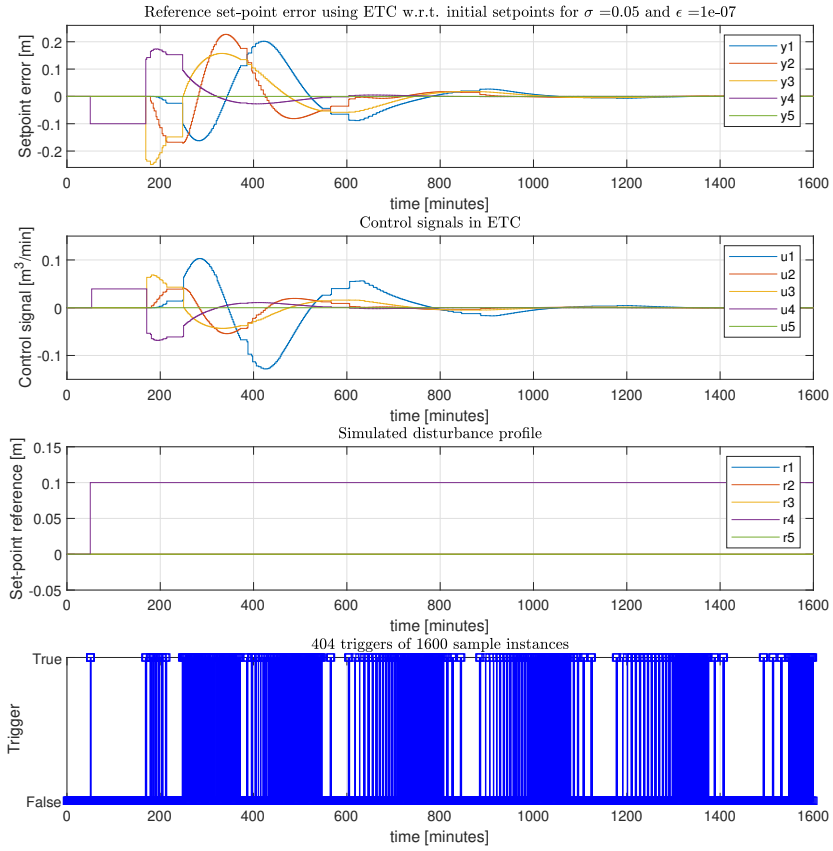


Figure 5-6: The plant outputs, control inputs, disturbance profile and the triggering instants using decentralized event-triggered control for a set-point disturbance on pool 4 for $\sigma = 0.05$ and $\epsilon = 1 \cdot 10^{-7}$.

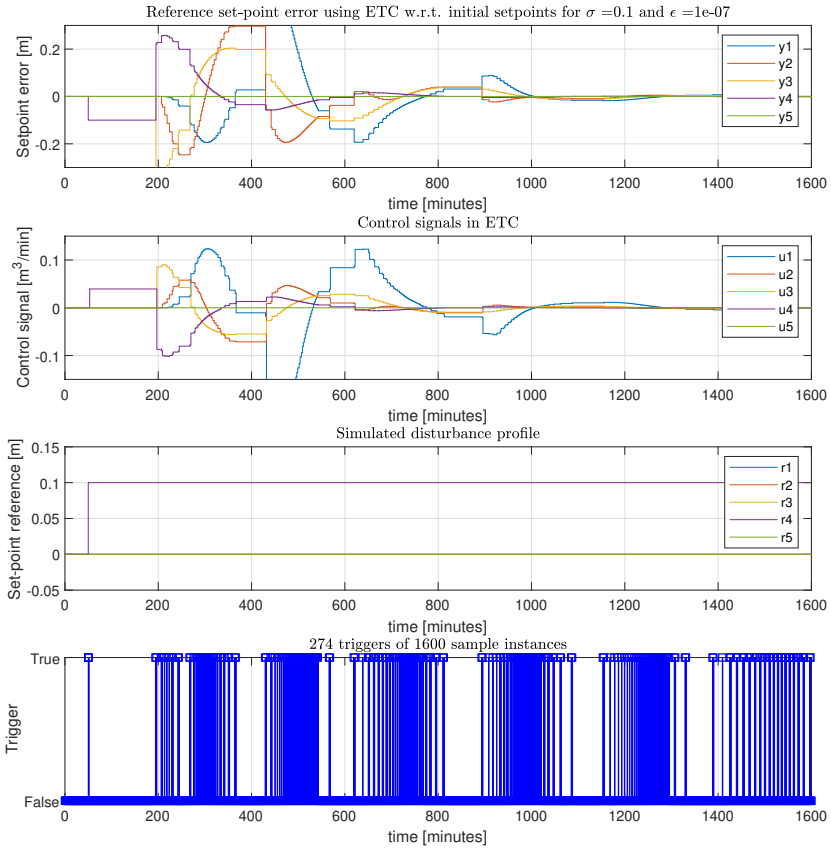


Figure 5-7: The plant outputs, control inputs, disturbance profile and the triggering instants using decentralized event-triggered control for a set-point disturbance on pool 4 for $\sigma = 0.10$ and $\epsilon = 1 \cdot 10^{-7}$. The scale of the axes is intentionally not scaled to the signals, to be able to compare this plot to previous results.

Chapter 6

The cyber-physical implementation

To be able to design a model-based controller for the lab setup, as is done in Chapter 4 for that set of pool parameters, the model parameters τ_i, α_i, and φ_i of the cyber-physical setup are needed. The parameters denote the transport delay, the pool surface area and the most dominant wave frequency, respectively. The model parameters still have to be identified using system identification, which is explained in the first section of this chapter. Subsequently is explained how a set of robust controllers can be designed in a similar fashion as is done in Chapter 4, as well as two alternative control design approaches. In Section 6-3, the implementation of the controller on the lab setup, using the Firefly nodes is explained. The chapter is finalized by an explanation on how the nonlinear flow dynamics around the controllable gates are handled.

6-1 Parameter identification

The parameter τ_i is identified by applying a step-input to each of the pools individually and visually inspecting the time-delay shown in the response. The values for α_i (the surface area of pool$_i$) are simply computed by multiplying the corresponding length and width of each pool, which are shown in Figure 6-1. The parameter φ_i, denoting the (most dominant) wave frequency of pool$_i$ can be identified by running open loop experiments and inspecting the responses of the plant. The plant parameters that are available at this point for the lab setup are shown in Table 6-1.

i	τ_i	α_i	φ_i
1	- minutes	0.1853 m^2	- rad/min
2	- minutes	0.1187 m^2	- rad/min
3	- minutes	0.2279 m^2	- rad/min

Table 6-1: Identified parameters per pool$_i$: delay (τ_i), surface area (α_i), and wave frequency (φ_i)

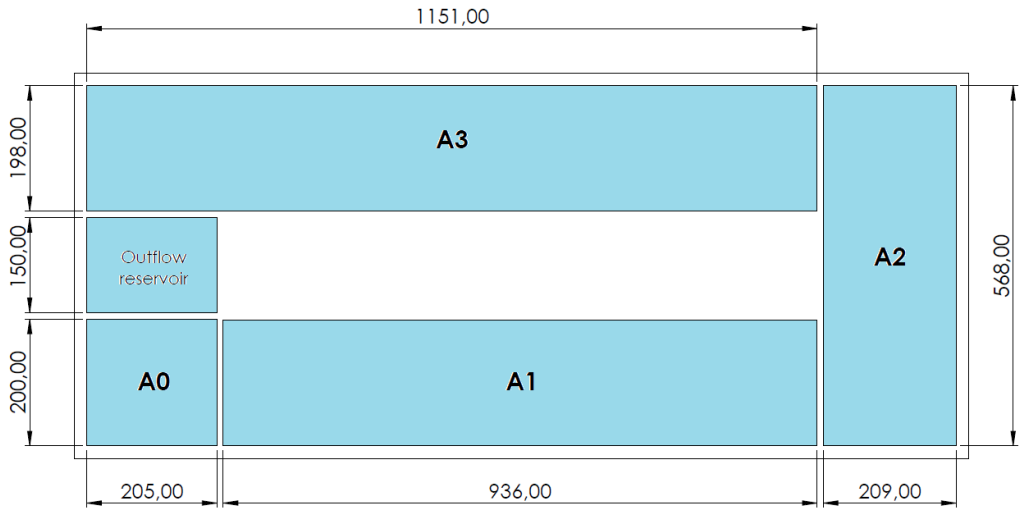


Figure 6-1: Schematic top view of the lab setup including the pool dimensions. All dimensions are in millimeters. The blocks A1, . . . , A3 indicate the areas corresponding to $\text{pool}_1, \ldots, \text{pool}_3$ respectively.

6-2 Robust control design

After obtaining the model parameters, a set of distributed controllers can be constructed for the lab setup, just as is done in Chapter 4 for the simulated plant. The controllers are again constructed by solving a set of linear matrix inequalities (LMIs), after which the state-space defining A, B, C, and D matrices are extracted and the dynamic output-feedback controllers are constructed in state-space form. The same structure is used as for the controllers designed in Chapter 4, based on [5]. To code that I created to do this is provided in Appendix A-1.

6-3 Implementing the controller

When a set of controllers is designed, it can be implemented on the lab setup quite easily. First, a sampling period has to be chosen and the continuous-time set of controllers needs to be discretized. The individual controllers can be centralized based on the techniques explained in Subsection 4-4-1, for which the code is all available in the repository provided in Appendix A-1. Then, the individual state update and output equations can be extracted from the constructed state-space model and these simple equations can be implemented for the controller node (called the sink node within the protocol code).

The sensor sampling period must be equal to the epoch period, and the measurements should be taken just before the radio up-time of the node starts. If the measurement would be taken just after the up-time, it could be the case that a trigger is delayed by one epoch period, compared to the first strategy.

6-4 Smart gates

The considered models for water irrigation systems (WISs), all use a linearized signal for the flow over the gates. However, this linearization has to be counteracted to implement a controller on a physical plant. The flow over the gates can be modeled using a nonlinear relation. To be able to still use a linear model and a control signal corresponding to that, it is needed to handle the (static) nonlinear relations of the system using sub-controllers called smart gates. These lower-level actuator controllers receive the regular control signal (u_i) from the controller as their local reference, after which (faster) reference tracking is performed locally (on the node) to get the required flow over a gate. The actual flow is expected to not be perfect because the flow is controlled using two pressure sensors and a servo motor and not using a calibrated flow regulator. Uncertainties in flow over the gates are thus to be expected and an extra disturbance is included in the model used for control design to account for this uncertainty as much as possible.

6-4-1 Modeling the nonlinearity

In previous studies, a Simulink model was used to simulate the behaviour of the lab setup that is now re-used. This Simulink model includes a lot of properties and sub-models used to model the plant and because of this it is very well suited to be reverse-engineered to acquire the parameters and calculations needed to describe flow over the so called sluice gates.

The main variables in computing flow over a gate, are the water heights before and after the gate. These variables are denoted by h_{in} and h_{out}, respectively. Other important values describe the sill height and the height of the gate (top of the opening), which are denoted by h_{sill} and h_{top} respectively. These variables are all depicted in Figure 6-2. The controlled variable is h_{top}, which is controlled by changing the position of the servo motor connected to the gate.

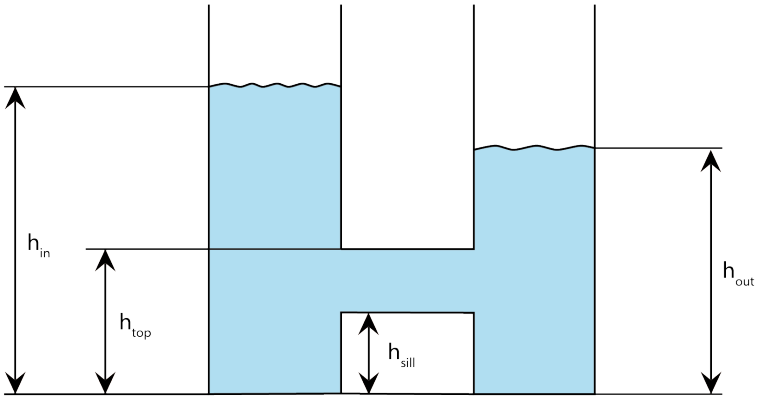


Figure 6-2: Variables used in the flow equations, where h_{top} is the controlled variable corresponding to the height from the bottom to the gate.

The first operation performed in the Simulink model is the determination of the highest and the lowest water height, denoted by h_1 and h_2 respectively. Note that h_1 does not necessarily

Parameter	Value	Units
b	0.01	m
Ce	0.95	-
μ	1.00	-

Table 6-2: Gate parameters; b denotes the width of the opening, C_e is the flow efficiency over the gate and μ denotes the flow-depth constant.

correspond to h_{in} in Figure 6-2. When h_{out} is higher than h_{in}, then h_1 corresponds to h_{out}, while it corresponds to h_{in} when $h_{in} > h_{out}$.

$$h_1 = \max(h_{in} - h_{sill},\ \ h_{out} - h_{sill},\ \ 0) \tag{6-1}$$

The minimum value of h_1 is zero, as can be seen in Eq. (6-1), as there is no flow when both water heights are smaller than the sill height. The sill height is zero for all regular gates in the lab setup, except for the last gate which is basically a variable height sill. The value of h_2 is computed as

$$h_2 = \min(h_{in} - h_{sill},\ \ h_{out} - h_{sill}). \tag{6-2}$$

The flow Q over a gate (in $\frac{m^3}{s}$) is computed as

$$Q = b \cdot C_e \cdot \sqrt{q} \cdot y_{jet} \cdot sign(h_{in} - h_{out}) \cdot \sqrt{2 \cdot g}, \tag{6-3}$$

where b is the width of the opening, C_e is the flow efficiency of the opening and g denotes gravitational acceleration (taken as $g = 9.81\ \frac{\mathrm{m}}{\mathrm{s}^2}$). All gates in the setup have identical dimensions, except for the last gate, which is called the sill gate from this point on, and this sill gate should not be considered as a regular gate. The parameters of the gates are given in Table 6-2. The value of q, used in Eq. (6-3), is computed as

$$q = h_1 - \max\left[h_2,\ \ \min\left(h_1 \cdot \frac{2}{3},\ \ \mu \cdot (h_{top} - h_{sill})\right)\right], \tag{6-4}$$

of which the term $h_1 \cdot \frac{2}{3}$ is referred to as the critical depth. The constant μ is related to flow depth. The term $\mathrm{sign}(h_{in} - h_{out})$ is used to indicate the direction of the flow in Eq. (6-3), where the flow Q is positive when it flows in the direction from h_{in} to h_{out}. The term $\mathrm{sign}(h_{in} - h_{out})$ of Eq. (6-3) is defined as

$$\mathrm{sign}(x) = \begin{cases} 1, & \text{if } x > 0 \implies h_{out} < h_{in} \\ 0, & \text{if } x = 0 \implies h_{out} = h_{in} \\ -1, & \text{if } x < 0 \implies h_{out} > h_{in} \end{cases} \tag{6-5}$$

Finally, the term y_{jet} of Eq. (6-3) is defined as:

$$y_{jet} = \min\left[\mu \cdot (h_{top} - h_{sill}),\ \ \max\left(h_2,\ \ h_1 \cdot \frac{2}{3}\right)\right] \tag{6-6}$$

Using these equations, the flow over each gate can be computed using the current position of the gate and the two measured water pressures, which are directly related to the water heights.

More information about possible flow cases around the gates is described in [26]. A full study on calibration of submerged multi-sluice gates is described in [27], which may provide interesting insights if problems are encountered in calibrating the flow control mechanism.

6-4-2 Feed-forward flow control

The nonlinear equations described in Subsection 6-4-1 describe a set of dynamics that can be interpreted as a hybrid (switching) system, which is mainly due to the minimum and maximum operators used in a number of the equations. These cases are all worked out and split up based on the inputs of the equations, being h_{in}, h_{out} and Q_{ref}. Q_{ref} denotes the reference flow as computed by the main (centralized) controller. This reverse engineered is called the feedforward map from now on. The structure of the implementation of this feedforward map is depicted in Figure 6-3.

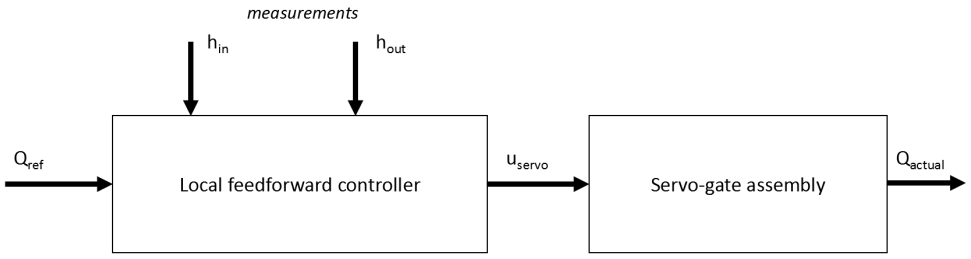


Figure 6-3: Feed-forward block scheme using Q_{ref} in m^3/s as input reference value. The feedforward controller computes a (servo motor) control signal, denoted by u_{servo}, based on the water levels h_{in} and h_{out} (in meters), such that the actual flow Q approaches Q_{ref} as much as possible.

The feedforward flow map is simulated for a set of random reference flow values within a reachable range, using MATLAB. The corresponding code used to create this feedforward control function, as well as the code used to perform this validation, is included in the distributed control GitHub repository, for which an explanation is included in Appendix A-1. The results of this validation simulation are shown in Figure 6-5. This is a validation of the reverse map, which only serves as a check to see if the map is inverted correctly. To determine in which flow region the gate is in at a certain point, the maximum achievable flow is computed for the most recent measurements h_{in} and h_{out}. A 3D-map showing these maximum flow values is shown in Figure 6-4.

The map is not yet validated on the lab setup. To validate the flow map on the lab setup, extra tests have to be designed, which could be done by analysing the increase in water height over time for a certain pool by setting Q_{ref} to a fixed value for example. Then the increase in water height over time can be compared to the expected flow, which will show how the feedforward controller is performing.

An alternative way to implement the reversed map is to use a lookup table that is generated beforehand, but the Firefly nodes have only a limited amount of memory available and it may take long to process the table on the fly, but please note that this is not tested yet and this assumption may be wrong.

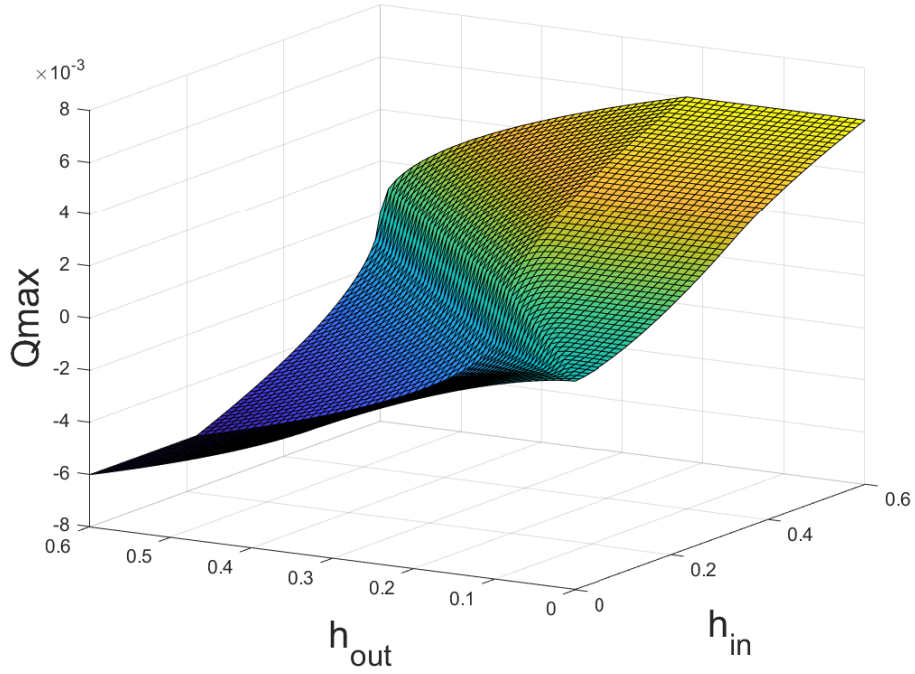


Figure 6-4: 3D map showing the maximum flow values Q in m^3/s for different water levels of h_{in} and h_{out} in meters.

In the process of creating the feedforward map, one case is encountered which does not have an explicit solution. Even without an explicit solution, the feedforward controller should still come up with a gate position that leads to an amount of flow that is as close as possible to the reference flow Q_{ref}. To do this, a bisection optimization algorithm is programmed to find a good gate position corresponding to the reference flow Q_{ref} in a finite number of bisection steps. The maximum number of steps (denoted by N_{max} in the corresponding code) is currently set to 15.

6-4-3 Testing the gates

The gates are all individually tested and the position control of the gates works as expected when controlling them using the Firefly nodes. An example program to control servo motors of the Contiki-OS was modified to have the right timing for the servo motors on the setup to perform this test, which immediately showed that everything works. The hardware is designed and assembled in such a way that the gate is closed when the servo motor is in its limit position on one side. When the servo motor is in the other limit position, there are no physical constraints. This is done such that it is not needed to limit the servo motor positions in the software, as this could have presented complications in future use of the lab setup.

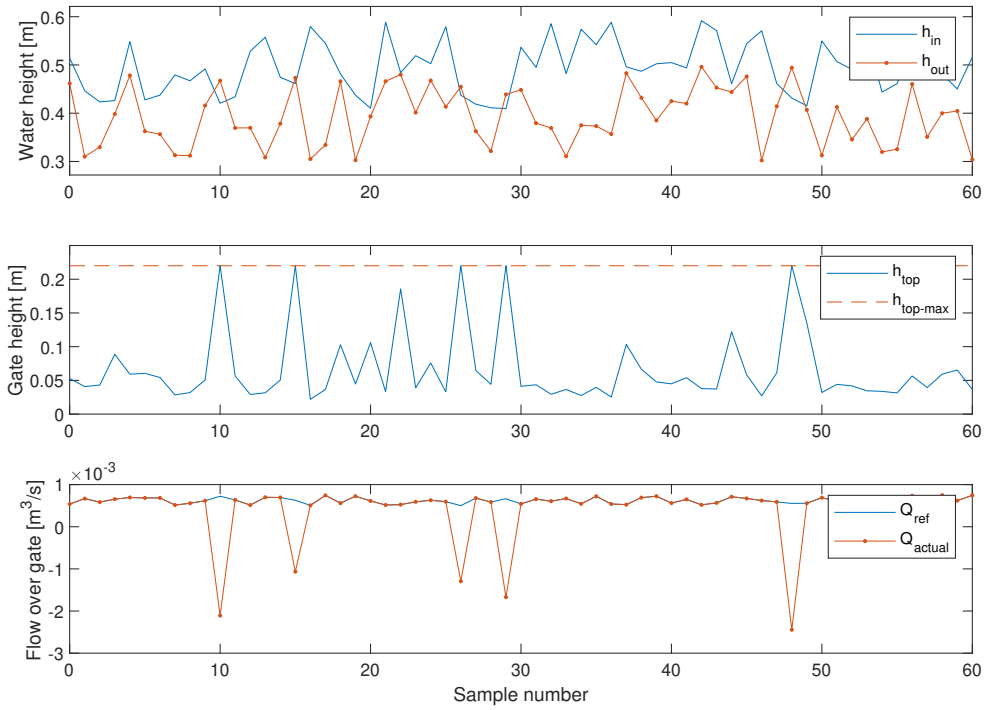


Figure 6-5: Plot showing a validation of the feedforward map for random flow reference values within a certain range, and random values for h_{in} and h_{out}. It can be observed that the feedforward mapping works perfectly as long as $h_{in} > h_{out}$, which can be explained by noting that the reference flow values are in this case always positive and a positive flow value can only be obtained when $h_{in} > h_{out}$, which should also be the case when using distant downstream control (DDC) techniques.

Chapter 7

Conclusions

In this chapter, the conclusions regarding the thesis are discussed. The chapter is finalized by a section discussing possible future work that could be performed as a follow-up of this project.

7-1 Conclusions

The thesis started with a plan to combine wireless control and event-triggered control (ETC) and apply the combination to water irrigation systems (WISs), to eventually enable the use of centralized control structures in this field, while minimizing the expenses needed to realize this structure. The main goal could be realized by designing an infrastructure that would allow the use of wireless ETC, as this has all the features in terms of needed communication and it does not require the expensive install and maintenance of cables over lengths of multiple kilometers.

To test the combination of wireless control and ETC in practice, a communication protocol was designed and tested, which resulted from close collaboration of our research group and the D3S Research group of the University of Trento. The protocol is called Wireless Control Bus (WCB) and was designed to perform all functionalities needed to apply ETC on a cyber-physical lab setup. The design of WCB was not part of this project, but a lot of testing of previous versions of WCB was done to make sure that we could use WCB in our lab and to make sure that the integration of the hardware on the lab setup was actually possible in WCB.

A cyber-physical lab setup was designed and built, with the use of WCB in mind, to eventually conduct real-time periodic event-triggered control (PETC) experiments on a physical setup. The lab setup consists of a series of pools, which can be used to mimic the dynamics of a WIS. A wireless communication infrastructure is present to perform control on the plant using WCB and a universal serial bus (USB) infrastructure was made to log all signals of experiments performed on the lab setup. This data can be analyzed during experiments or can be processed afterwards. Logging can be done using the Raspberry Pi that is included

in the lab setup, which can be also used to flash the software to the nodes in the networked control system (NCS) in case of updated software for the nodes. The actuator and sensor signals were all analyzed and unit test software was created to test the individual parts of the physical setup.

Next, the control design phase was started by designing a set of state-of-the-art distributed output-feedback dynamic controllers for WISs. The controller was implemented in centralized form eventually by recasting the set of individual controllers into a combined impulsive system description. During the project, I chose to centralize this set of controllers to try to proof stability for the closed loop system, and besides that, the centralized controller description very well suits the current set of features included in WCB, which allows the use of PETC techniques on the lab setup.

It was proven to be hard to prove global exponential stability (GES) and a finite $\mathcal{L}_2$-gain (from the disturbance signals $\bar{w}$ to the outputs y) for larger systems than typically given in the examples in literature [10]. In the case of the plant used in this project, the closed-loop system has 55 states, 5 outputs and 5 control signals. The applied theorem (Theorem V.2 in [10]) states that if one can find a set of matrices satisfying the set of linear matrix inequalities (LMIs) in the theorem, then the closed loop system is GES and there is a finite $\mathcal{L}_2$-gain, which is smaller than some chosen value of γ, which thus denotes an upper bound on the $\mathcal{L}_2$-gain. Using CVX [21], [22] and MATLAB, I managed to get a solution to the set of LMIs once, using extremely conservative bounds on the decay rate ρ and γ, but I was not able to reproduce this result. These kinds of problems are most likely due to numerical errors, which may be less of a problem when having a reduced number of states.

Numerical experiments were conducted to prove the achievable reduction in communication by applying ETC on WISs in Chapter 5. A reduction in communication of more than 80% was achieved using the exact same decentralized triggering structure as could be used for a real-world WIS. The triggering function was tuned, and the typical ETC trade-off between the amount of communication on one side and the reduction in control performance on the other side was clearly illustrated by presenting and discussing the results of the numerical experiments.

Finally, in Chapter 6 was discussed how the methods that have been used in the preceding chapters can be used to eventually implement the proposed control structure on the lab setup. First, system identification should be performed, which is now possible since all sensors are working at this point. Next, the created MATLAB scripts, included in Appendix A-1, can be used to construct a centralized controller. Then, this controller can be implemented in WCB, as well as the corresponding triggering functions which are presented in Chapter 5 for this exact type of system. The nonlinear part, related to the smart gates, was also already programmed in MATLAB and is thus also ready to be implemented in WCB, apart from some adaptations that most likely have to be performed because of the differences between the programming languages MATLAB and C.

To conclude, the infrastructure needed to perform wireless ETC on WISs was designed, and a plan has been presented on how the individual parts can be combined to test wireless ETC on the lab setup. However, there is still some work that needs to be carried out before actual experiments can be conducted on the lab setup as is discussed in the previous paragraph. The final section presents ideas on future work related to the presented work.

7-2 Future work

7-2-1 (Software) improvements for the lab setup

An idea that was already present from the start of the project, is to create a (web) interface on which the water levels and the gate positions are visualized real-time. Besides that, a possible more valuable addition to the current code base, in terms of actual research, may be to create experiment analysis code. The D3S Research Group already created scripts to analyse experiment results for their test-bed. Similar code for our lab setup, designed to quickly parse the data resulting from experiments, could be very valuable in the future.

In order to be able to do precise reference-tracking, A structure could be created in the code of WCB to calibrate the sensors. By implementing it in WCB, both the controller node as the logging code will receive calibrated sensor values. This could help, as I have noticed during the unit tests that not all sensors present the same sensor values for equal water heights. By also defining sensor-specific values for the minimum and maximum water heights, a sensor specific map can be defined in the code, which will increase the accuracy of the water level measurements.

A second addition to WCB would be the ability to send the reference set-point values (the required water heights of the individual pools) to the nodes. Using the value of the reference set-point, a change of coordinates can be performed such that the triggering condition always has its origin at the reference set-point. This is not possible currently, as only the control signals are being transmitted to the nodes at this point. This is important as most theory on PETC [10],[28] is based on the assumption that the controlled system should converge to the origin. The effects of this modification are shown in Subsection 5-1-1, and these were found to be quite significant in terms of PETC performance.

An extra improvement to conduct experiments on the lab setup would be to have calibrated flow controlling off-take valves, that can be read-out or maybe even controlled using software. By adding these valves, flow validation experiments can be performed and an additional improvement will be that exact disturbances can be depicted when presenting experiment results. Connecting these to the setup will not be too hard, as the manual off-take valves are connected using threaded PVC connectors, which can be changed easily.

7-2-2 Control design

Because WCB is designed to use a centralized controller, it may be interesting to design a centralized controller for the lab setup instead of first designing a set of distributed controllers and combining them later on to create a centralized controller. Directly designing a centralized controller will simplify the analysis, because in that case it will be easier to write the controller matrices in the form required by [10] for the impulsive system approach for dynamic output feedback controllers, used to prove GES and an upper bound on the $\mathcal{L}_2$-gain. Another important aspect is that a set of distributed controllers will have more structure than a centralized controller, which can be interpreted as it having more constraints, which implies that in theory the optimal (best possible) set of distributed controllers can never outperform the optimal (best possible) centralized controller.

A second topic I recommend to inspect to improve the control performance is to do an experiment that can be used to validate the flow over a gate computed by the presented nonlinear flow model. By doing this, a comparison between the actual flow and the flow computed using the gate position and the two water heights surrounding the gate can be performed. A future addition to the smart gate could be to include some sort of (Kalman) observer, using for example a differential-based model to get a more accurate estimate of the flow, which could then be used to add a feedback loop to the control structure shown in Figure 6-3 in order to improve the local flow controllers.

7-2-3 Research

The current LMI condition presented in [10] for GES and a finite $\mathcal{L}_2$-gain for output feedback controlled systems (Theorem V.2) was proven to be very hard to solve for a closed loop system having 55 states. It is numerically unstable, which makes it very hard to come up with a right set of parameters σ, ρ, and γ, and even when a solution is there, it was not reproducible. The expectation is, that this has mainly to do with the dimensions of this system, but this is just an assumption. It could be very interesting to test more of these theorems and create an overview that can be used as a guideline in trying to prove stability for (output-feedback) PETC systems. For example, Theorem V.3 of [10] could be tested for the current system.

As follow-up work on this thesis, it could be interesting to create simulations for decentralized event-triggering conditions, using the already available building blocks, using identified lab parameters. This would allow one to compare simulations and actual experiments for this kind of systems and show what the differences are between the two approaches.

Appendix A

Software and git repositories

In this appendix, the contributions in terms of software are listed with short explanations, as well as links to the GitHub repositories the code is in. Instead of including the actual code in the report, the choice was made to make a GitHub repository, so the code can be reused more easily. The code was divided over a set separate GitHub repositories to keep the repositories aimed at a specific use. The second reason was that not all of the software was intended to be spread around in the same way, while I would of course never mind that people use the things I created for their respective projects as long as a reference to me is kept in the code at all times.

Do not hesitate to contact me by email on lontjacob@hotmail.com in case of any issues.

A-1 MATLAB code for the robust control problem

This appendix shortly explains the MATLAB code used to solve the H_∞ problem described in Chapter 4, as well as the code to subsequently construct the output-feedback dynamic control matrices. The controller was created in the file 'cantoni_LMI.m', which uses the optimization package CVX to solve the robust control problem. The function 'createsi' creates the actual state-space matrices of the set of distributed controllers, while the function 'splitup_S_matrix' splits up the created S-matrices into A, B, C, and D matrices for use in simulations and to for use in the combined plant model, which is constructed in Subsection 4-4-2.

The repository is located at:

https://github.com/jacoblont/grad_distributed_control_design

A-2 ADS1115 library for Contiki-OS

The code used to combine the Zolertia Firefly nodes and the Adafruit ADS1115 analog to digital converter (ADC) using inter-integrated circuit (I2C) is stored in the following reposi-

tory:

https://github.com/jacoblont/ads1115-contiki-os-firefly

Additional explanations regarding the use of this code are included in the GitHub page, including a guide on how to compile the code and how to clone the repository, including the underlying 'submodule' repository, which is used to always include the most recent files of Contiki-OS.

A-3 Current version of Wireless Control Bus

The current version of the protocol used on the lab setup was stored in a private repository. The repository was kept private, since the creators of the protocol did not yet share this version (the CC2538 version) themselves on their GitHub page[1]. This version was still under the name Crystal[2], instead of Wireless Control Bus (WCB), and it was not yet event-triggered, but instead, it was set up to perform periodic control. The periodic control version was intended to show correct periodic control and to check that the hardware and software is working as expected, without already adding the event-triggered control (ETC) part.

There is a newer version of WCB available now, which includes the ETC part. This version can be obtained by contacting ir. Gabriel de Albuquerque Gleizer of the Mazo research group and together contacting Matteo Trobinger of the D3S Research Group of the University of Trento, Italy.

[1] https://github.com/d3s-trento
[2] https://github.com/d3s-trento/crystal

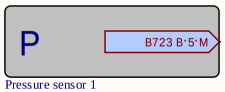
Appendix B

Electronics schematics

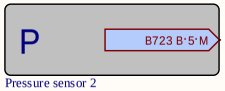
This appendix includes all the electronics schematics used for the design of the lab setup. It includes both the schematics as the printed circuit board (PCB) production drawings.

B-1 Overview interconnections

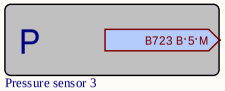
This attached document shows the connections between the individual parts on the lab setup and its main purpose is to clearly document the required connections and corresponding connectors. In the preceding sections, the designs of the underlying components are elaborated.



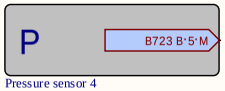
Pressure sensor 1



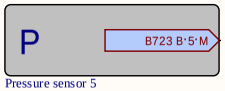
Pressure sensor 2



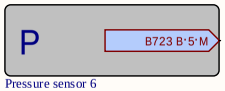
Pressure sensor 3



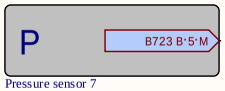
Pressure sensor 4



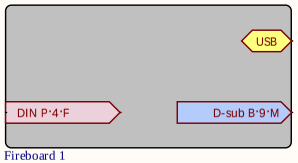
Pressure sensor 5



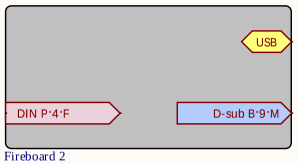
Pressure sensor 6



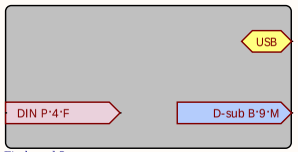
Pressure sensor 7



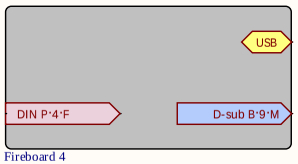
Fireboard 1



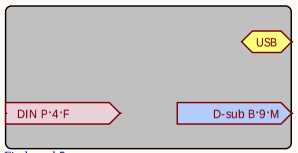
Fireboard 2



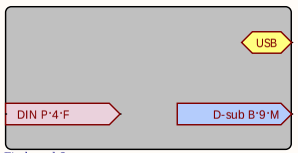
Fireboard 3



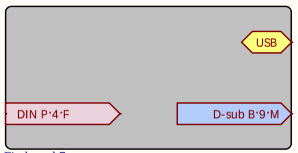
Fireboard 4



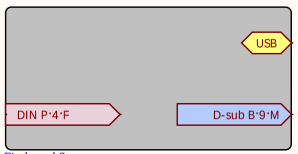
Fireboard 5



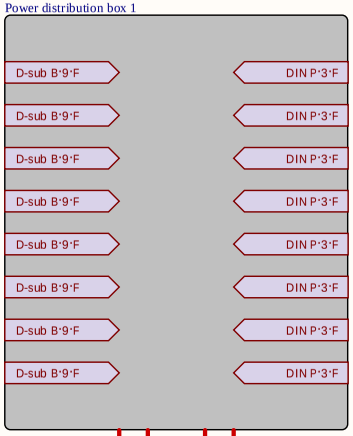
Fireboard 6

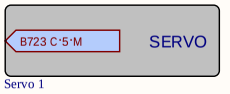


Fireboard 7

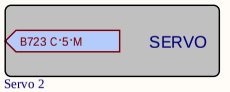


Fireboard 8

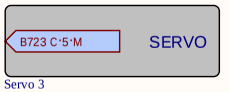




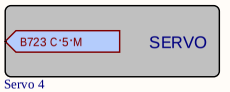
Servo 1



Servo 2

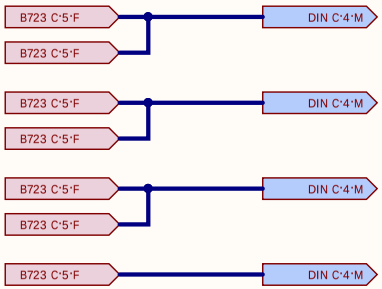


Servo 3

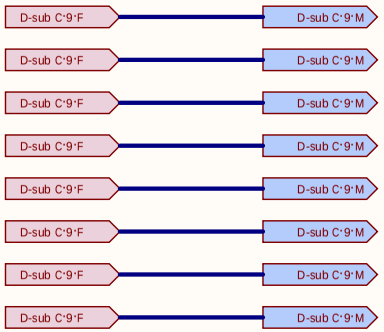


Servo 4

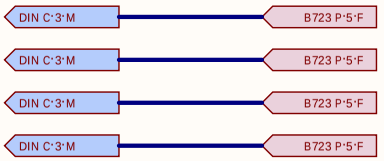
Pressure sensor <-> Fireboard



Fireboard <->Power distribution box



Power distribution box <-> Servo




```
Pressure sensor connector:
    B723 B·5·M:  Binder series 723, builtin, 5 pin male
Pressure sensor <-> Fireboard connectors:
    B723 C·5·F:  Binder series 723, cable mount, 5 socket female
    DIN C·4·M:   DIN, cable mount, 4 pin male
Fireboard connectors:
    DIN P·4·F:   DIN, panel mount, 4 socket female
    D-sub B·9·M: D-sub, PCB mount, 9 pin male
Fireboard <> Power distribution box connectors:
    D-sub C·9·F: D-sub, cable mount, 9 socket female
    D-sub C·9·M: D-sub, cable mount, 9 pin male
Power distribution box connectors:
    D-sub B·9·F: D-sub, PCB mount, 9 socket female
    DIN P·3·F:  DIN, panel mount, 3 socket female
Power distribution box <-> Servo connectors:
    DIN C·3·M:  DIN, cable mount, 3 pin male
    B723 P·5·F: Binder series 723, panel mount, 5 socket female
Servo connector:
    B723 C·5·M: Binder series 723, cable mount, 5 pin male
```

Title: *Overview interconnections*		
Project: Water Testbed	Revision: 1	*Will van Geest TU Delft DCSC Mekelweg 2 2628CD Delft*
Date: 8-4-2020 12:50:00	Sheet 1 of *	
File: Overview2.SchDoc		

B-2 Connector and power supply details

This appendix includes the most recent list of ordered connectors and power supply units. The list includes all https://nl.rs-online.com/web/-numbers, so they can be found when needed in future use.

Overview Connectors Lab Setup Water Irrigation System **Jacob Lont** **Most recent revision: 02-07-2020**

	# pins	Gender	To	Qty	RS online nr	Type	Position	Eur/Piece (excl VAT)	Total cost
Sensors	4 pin	male	cable	4	7863448	RS PRO B-8003 Series	Sensorkabel to Fireboard	2,97	11,88
		female	panel	8	7863429	RS PRO B-2002 Series	Fireboard PS connector	2,88	23,04
	5 pin (2 used)	female	cable	7	2616023	BINDER 723	Sensor to sensorcable	20,62	144,34
Servos	3 pin	male	cable	4	Use existing	5-pin DIN	Servo to SRVkabel	0	0
		female	panel	4	Use existing	5-pin DIN	SRVkabel wallmounted connector	0	0
		male	cable	4	Use existing	DIN EN 60529	Wallmount-cable to Powerbox	0	0
		female	panel	8	7406095	DIN EN 60529	Powerbox to Servocable	4,77	38,16
Signals	9 pin	male	panel + pcb	8	1748821	D-sub 9, 2.77mm pitch	Fireboard SIG connector	1,152	9,216
		female	panel + pcb	8	1748825	D-sub 9, 2.77mm pitch	In power distribution box	1,282	10,256
D-sub	-			32	252119		D-sub Hex standoffs UNC4-40 Outer + Inner threads, L = 8,30 mm	0,475	15,2
Power	2 pin			2	1895966	Screw terminal block	5V 25A Terminal connector	1,284	2,568
	2 pin			0	Same as for 5V	Screw terminal block	9V		0

Subtotal ex. VAT
254,66

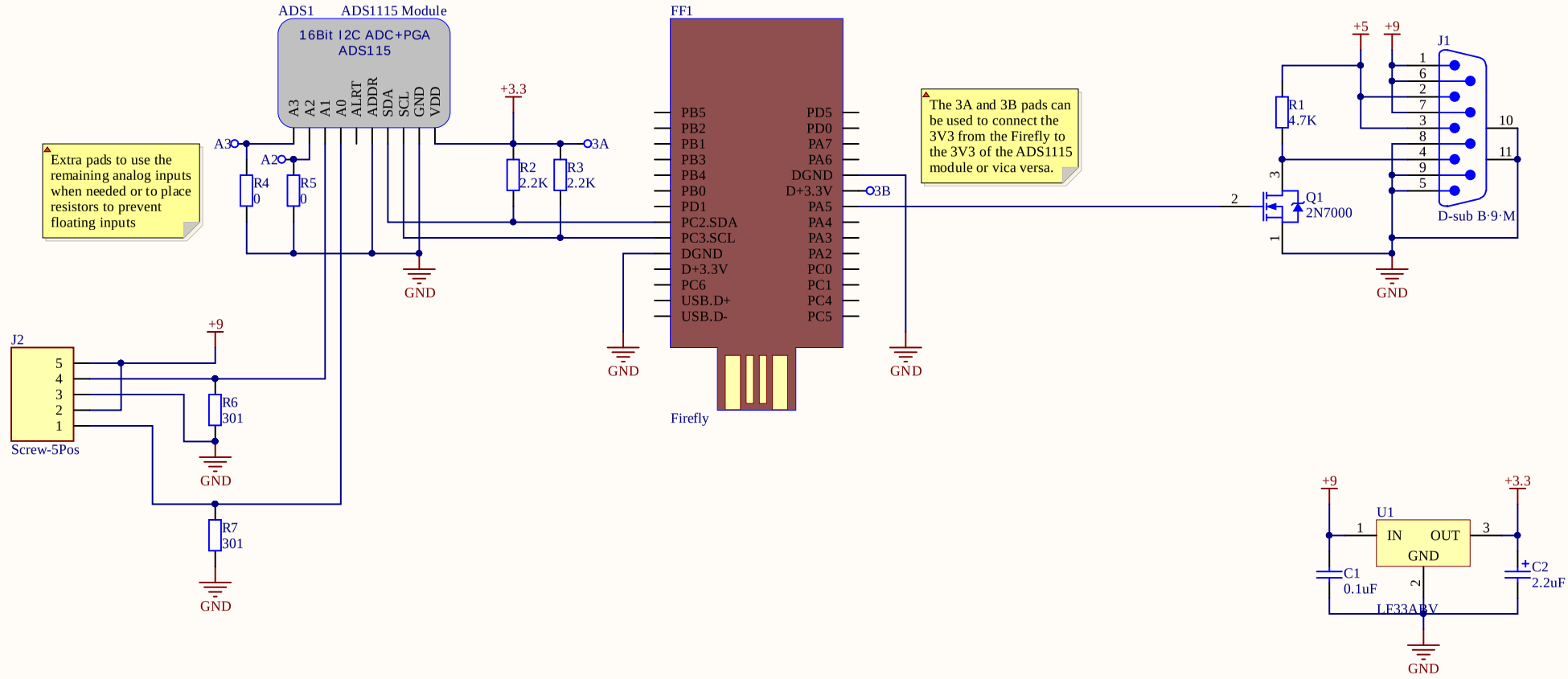
Power Supply Units	Voltage	Required Current	Max current of PSU	Qty	RS online nr	Type	Extra description	Eur/Piece (excl VAT)	Total cost
Servos	5V	25A	26A	1	6447023	Mean Well RS-150-5	5V DC Switch Mode Power Supply	32,79	32,79
Sensors	9V	4A	5A	1	1759086	XP Power AKM45US09	IEC C13 Inlet connector	29,9	29,9
				1	7769122	IEC 60320 C13	9V adapter-input plug	2,83	2,83

Subtotal ex. VAT
65,52

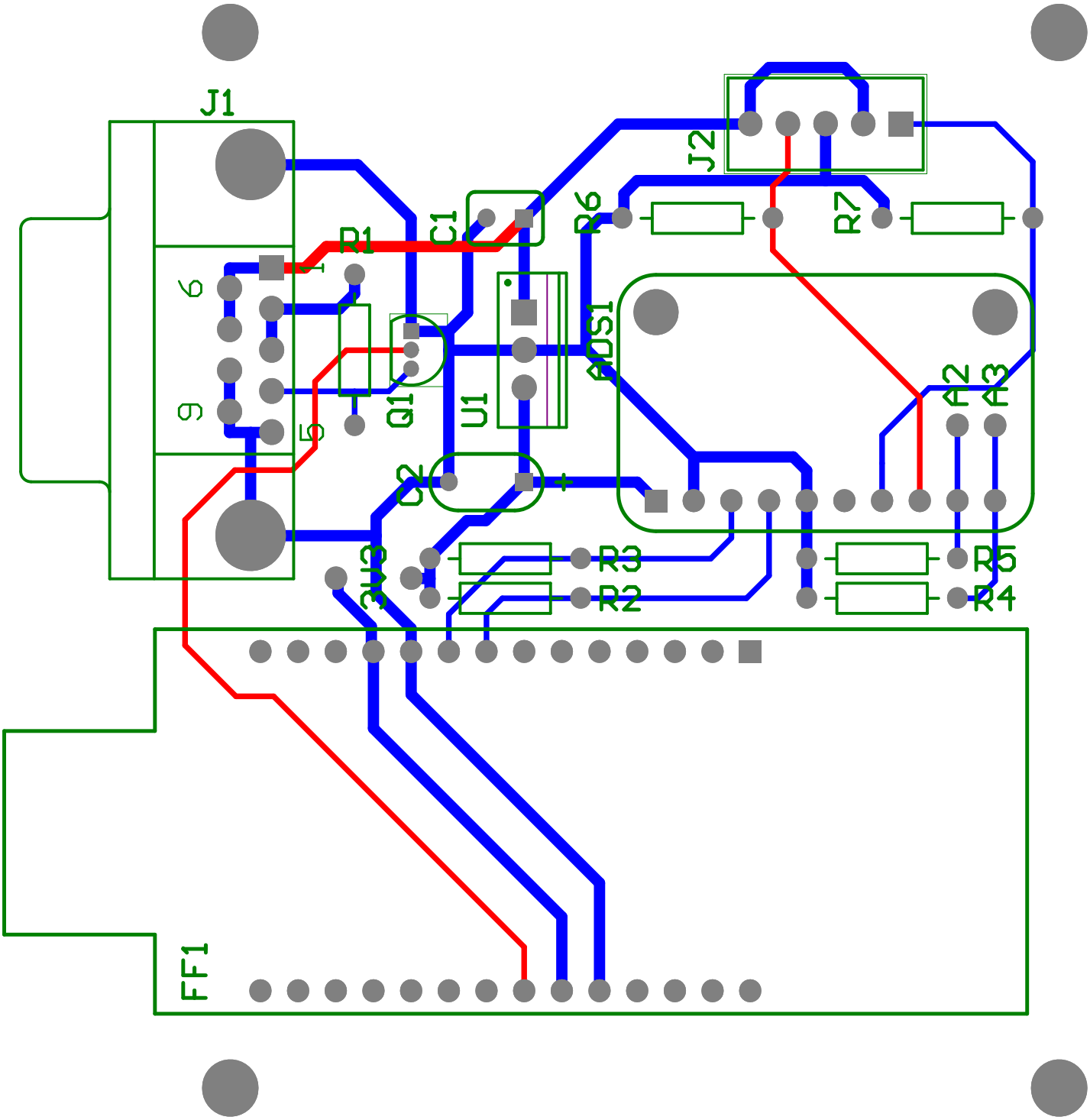
Total cost ex. VAT
320,18

B-3 Fireboard documentation

The Fireboard is the PCB designed to hold a Firefly board and an ADS1115 analog to digital converter (ADC) board. More information on the functionalities of the board are described in Chapter 3.



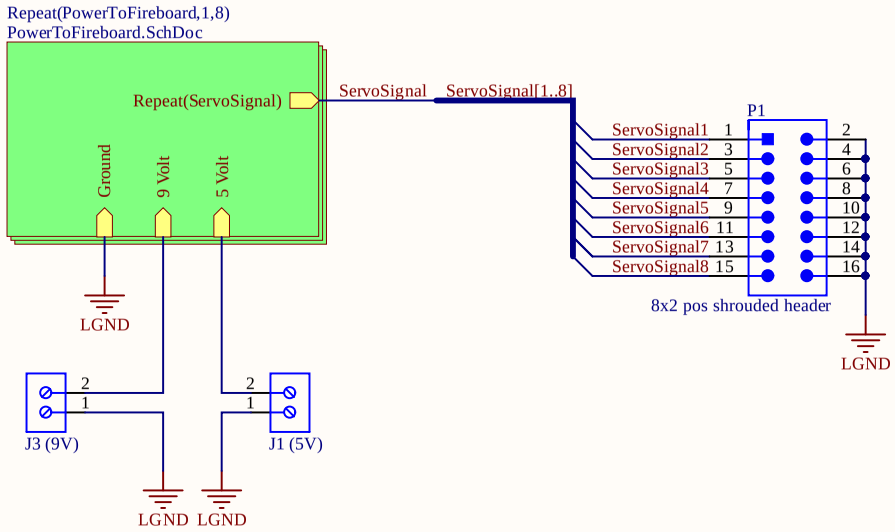
Title: ***Fireboard***		
Project: Water Testbed	Revision: 1	*Will van Geest* *TU Delft* *DCSC* *Mekelweg 2* *2628CD Delft*
Date: 26-5-2020 12:38:17	Sheet 1 of *	
File: Fireboard.SchDoc		

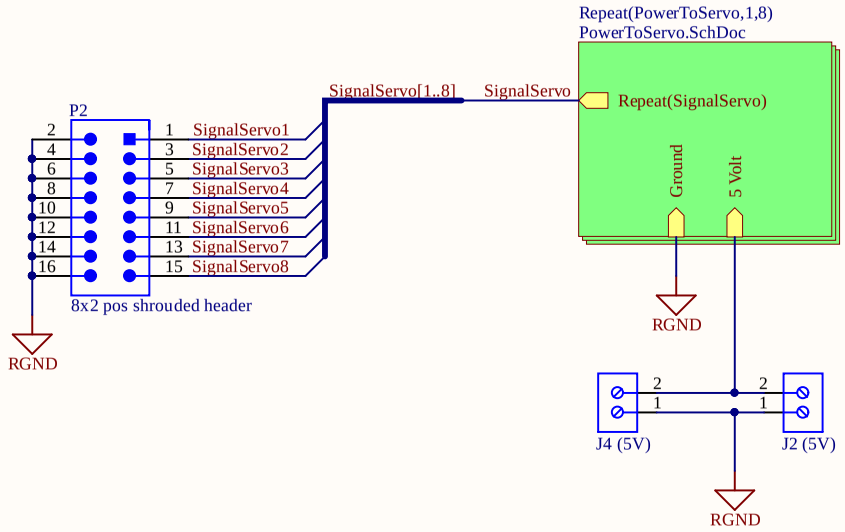

J1
J2
R1
C1
R6
R7
6
9
1
5
Q1
U1
ADS1
A2
A3
C2
3V3
R3
R2
R5
R4
FF1

Designator	Description	Quantity	Manufacturer 1	Manufacturer Part Number 1	Supplier 1	Supplier Part Number 1
ADS1	ADS1115 16-Bit ADC breakout, 4 Channel with Programmable Gain Amplifier	1	Adafruit	1085	Antratek	ADA-1085
C1	Capacitor (Radial)	1	TDK-Lambda	C320C104K5R5TA	RSComponents	5381310
C2	Polarized Capacitor (Radial)	1	KEMET	T356C225K035AT	RSComponents	5382060
FF1		1	Zolertia	Firefly A2		
J1	D-sub Plug Assembly, 9 Position, Right Angle	1	NorComp	182-009-113R531	RSComponents	1748821
J2	Serie MPT 0.5/5-2.54, 5way PCB terminal strip	1	Phoenix Contact	1725685	RSComponents	2204298
Q1	Small Signal MOSFET, 200 mA, 60 V, N-Channel, 3-Pin TO-92, Bulk Box	1	NXP Semiconductors	2N7000	Farnell	9845178
R1	Resistor	1	Vishay BCcomponent	SFR2500004701JA500	RSComponents	1650993
R2, R3	Resistor	2	Vishay	MRS16000C2201FCT00	RSComponents	6832819
R4, R5	Resistor	2	Vishay BCcomponent	SFR25000Z0000ZA500	RSComponents	1885290
R6, R7	Resistor	2	TE Connectivity	YR1B301RCC	RSComponents	7549060
U1	500mA, Very Low Drop (0.45V) Voltage Regulator, 3.3V, 3-Pin TO-220	1	STMicroelectronics	LF33ABV	RSComponents	6869710

B-4 Power Distribution Box design

This appendix includes the electronic design of the power distribution box. More information on the functionalities of the power distribution box are described in Chapter 3.

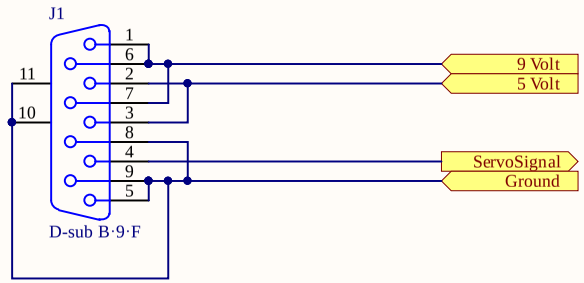




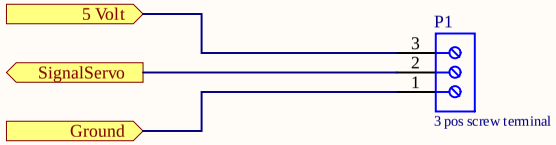
The electronics in the Power distribution box is divided over 2 PCB's. One PCB carries the Dsub connectors and should be mounted on the "left" side of the enclosure. The other PCB holds the Binder connectors and should be mounted on the "right" side. The PCB's are connected to each other with a simple flat-cable. Each PCB can be mounted without putting mechanical stress on the other, even if dimensions are not completely matching.

Connect the power supplies to J2 and J3. Use J1 and J4 to pass the 5 volt to the "left" PCB.

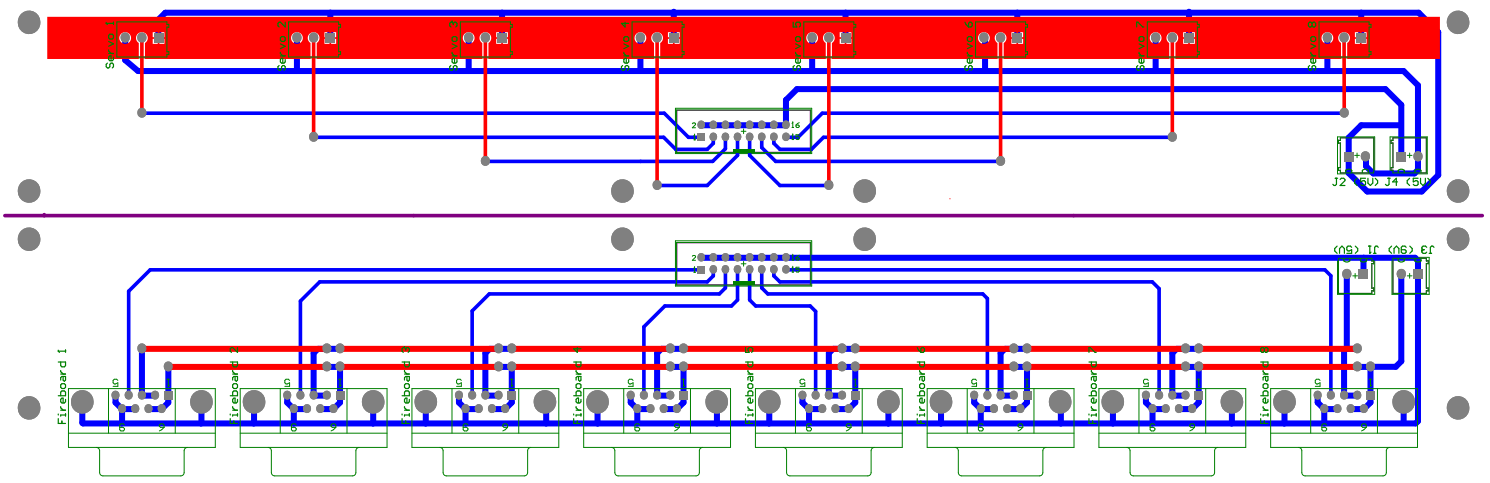
Title: ***Power distribution box***		
Project: Water Testbed	Revision: 1	*Will van Geest* *TU Delft* *DCSC* *Mekelweg 2* *2628CD Delft*
Date: 18-6-2020 10:12:37	Sheet 1 of *	
File: PowerDistributionBox.SchDoc		



Title:	***Power to Fireboard***	
Project: Water Testbed	Revision: 1	*Will van Geest* *TU Delft* *DCSC* *Mekelweg 2* *2628CD Delft*
Date: 18-6-2020 10:12:38	Sheet 2 of *	
File: PowerToFireboard.SchDoc		



Title:	***Power to Servo***		
Project: Water Testbed		Revision: 1	*Will van Geest* *TU Delft* *DCSC* *Mekelweg 2* *2628CD Delft*
Date: 18-6-2020 10:12:38		Sheet 3 of *	
File: PowerToServo.SchDoc			


Servo 1
Servo 2
Servo 3
Servo 4
Servo 5
Servo 6
Servo 7
Servo 8
J2 (5V) J4 (5V)
J3 (5V) J1 (5V)
Fireboard 1
Fireboard 2
Fireboard 3
Fireboard 4
Fireboard 5
Fireboard 6
Fireboard 7
Fireboard 8

Note	Description	Quantity	Manufacturer 1	Manufacturer Part Number 1	Supplier 1	Supplier Part Number 1
	Serie 2141 - 3.50 mm Horizontal Entry Modular with Rising Cage Clamp WR-TBL, 2 p	4	Wurth Electronics	691214110002	RSComponents	8267295
	D-sub Receptacle Assembly, 9 Position, Right Angle	8	NorComp	182-009-213R531	RSComponents	1748825
	Male Box Header WR-BHD, THT, Vertical, pitch 2.54 mm, 16 pins	2	Wurth Electronics	61201621621	RSComponents	7718357
3 pos screw terminal	Serie 2141 - 3.50 mm Horizontal Entry Modular with Rising Cage Clamp WR-TBL, 3 p	8	Wurth Electronics	691214110003	RSComponents	8267298

Bibliography

[1] E. Weyer, "Decentralised pi control of an open water channel," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 95 – 100, 2002. 15th IFAC World Congress.

[2] M. Cantoni, E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan, "Control of large-scale irrigation networks," *Proceedings of the IEEE*, vol. 95, pp. 75–91, Jan 2007.

[3] Y. Li and B. D. Schutter, "Stability and performance analysis of an irrigation channel with distributed control," *Control Engineering Practice*, vol. 19, no. 10, pp. 1147 – 1156, 2011.

[4] Y. Li, M. Cantoni, and E. Weyer, "Design of a centralized controller for an irrigation channel using H_∞ loop-shaping." -, Jan 2004.

[5] Y. Li and M. Cantoni, "Distributed controller design for open water channels," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10033 – 10038, 2008. 17th IFAC World Congress.

[6] M. Trobinger, T. Istomin, A. L. Murphy, and G. P. Picco, "Competition: CRYSTAL clear: Making interference transparent," in *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks, EWSN 2018. Madrid, Spain, February 14-16, 2018*, pp. 217–218, 2018.

[7] T. Istomin, A. L. Murphy, G. P. Picco, and U. Raza, "Data prediction + synchronous transmissions = ultra-low power wireless sensor networks," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, SenSys '16, (New York, NY, USA), pp. 83–95, ACM, 2016.

[8] K. J. Åström and B. Wittenmark, *Computer-controlled Systems (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.

[9] J. J. Lont, "(Literature survey on) Event-Triggered Control of Water Irrigation Systems," Oct. 2019. Carried out in preparation of the graduation project for the Master in Systems and Control at Delft University of Technology.

[10] W. P. M. H. Heemels, M. C. F. Donkers, and A. R. Teel, "Periodic event-triggered control for linear systems," *IEEE Transactions on Automatic Control*, vol. 58, pp. 847–861, April 2013.

[11] J. Araújo, A. Anta, M. Mazo, J. Faria, A. Hernandez, P. Tabuada, and K. H. Johansson, "Self-triggered control over wireless sensor and actuator networks," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–9, June 2011.

[12] X. Litrico and V. Fromion, "Advanced control politics and optimal performance for an irrigation canal," in *2003 European Control Conference (ECC)*, pp. 820–825, Sep. 2003.

[13] M. Trobinger, T. Istomin, G. Gleizer de Alberquerque, M. Mazo Jr., A. Murphy, and G. Picco, "The Wireless Control Bus: Enabling Efficient Multi-hop Event-Triggered Control with Concurrent Transmissions." ACM Transactions on Cyber-Physical Systems (to be submitted), 2020.

[14] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 73–84, April 2011.

[15] K. Leentvaar and J. Flint, "The capture effect in fm receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 531–539, 1976.

[16] "Contiki-OS website." https://www.contiki-ng.org/. Accessed: 2020-07-30.

[17] "Contiki-OS github page." https://github.com/contiki-os/contiki. Accessed: 2020-08-11.

[18] J. Schram, *Handleiding Watermodel*. Delft University of Technology, Jan 2001.

[19] "Source of Figure 3-12; How To Mechatronics." https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/. Accessed: 2020-06-29.

[20] C. Langbort, R. Chandra, and R. D'Andrea, "Distributed control design for systems interconnected over an arbitrary graph," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1502–1519, Sept. 2004.

[21] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1." http://cvxr.com/cvx, Mar. 2014.

[22] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control* (V. Blondel, S. Boyd, and H. Kimura, eds.), Lecture Notes in Control and Information Sciences, pp. 95–110, Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[23] The MathWorks, Inc., "MATLAB, version r2019a." https://www.mathworks.com/, June 2020.

[24] E. Weyer, "System identification of an open water channel," *Control Engineering Practice*, vol. 9, no. 12, pp. 1289 – 1299, 2001.

[25] M. Mazo and P. Tabuada, "Decentralized event-triggered control over wireless sensor/actuator networks," *IEEE Transactions on Automatic Control*, vol. 56, pp. 2456–2461, Oct 2011.

[26] G. Spaan, R. Nooyen, B. Graaff, and R. Brouwer, "Discharge formulas of crump-de gruyter gate-weir for computer simulation," *Journal of Irrigation and Drainage Engineering-asce - J IRRIG DRAIN ENG-ASCE*, vol. 129, 08 2003.

[27] M. F. Sauida, "Calibration of submerged multi-sluice gates," *Alexandria Engineering Journal*, vol. 53, no. 3, pp. 663 – 668, 2014.

[28] V. Dolk, D. Borgers, and W. Heemels, ""output-based and decentralized dynamic event-triggered control with guaranteed $\mathcal{L}_p$-gain performance and zeno-freeness"," *IEEE Transactions on Automatic Control*, vol. 62, pp. 34–49, 1 2017.

Glossary

List of Acronyms

ADC	analog to digital converter
CETC	continuous event-triggered control
DCSC	Delft Center for Systems and Control
DDC	distant downstream control
ETC	event-triggered control
ETM	event-triggering mechanism
GES	global exponential stability
I2C	inter-integrated circuit
IoT	Internet of Things
LLC	logic level converter
LMI	linear matrix inequality
LTI	linear time-invariant
MOSFET	metal–oxide–semiconductor field-effect transistor
NCS	networked control system
OS	operating system
PCB	printed circuit board
PDB	Power Distribution Box
PETC	periodic event-triggered control
PWM	pulse-width modulation
RC	radio-controlled
USB	universal serial bus
WCB	Wireless Control Bus
WCS	wireless control system

WLEP water-level error propagation

WIS water irrigation system

WSN wireless sensor network

ZOH zero-order hold