T.C.
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

DESIGN, DEVELOPMENT AND EVALUATION OF A STATE-OBSERVER CONCEPT FOR INDUSTRIAL AUTOMATION PLANTS

BAŞAK GENÇER

MASTER OF SCIENCE THESIS
ELECTRICAL ENGINEERING DEPARTMENT
CONTROL VE AUTOMATION PROGRAM

ADVISOR
PROF. DR. GALİP CANSEVER

İSTANBUL, 2011

T.C.

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

DESIGN, DEVELOPMENT AND EVALUATION OF A STATE-OBSERVER CONCEPT FOR INDUSTRIAL AUTOMATION PLANTS

The thesis study prepared by Başak GENÇER has been accepted as **MASTER OF SCIENCE THESIS** in the Graduate School of Natural and Applied Science Electrical Engineering Department on 22/11/2011 by the jury given in the following.

Thesis Advisor

Prof. Dr. Galip Cansever

Yıldız Technical University

Members of the Jury

Prof. Dr. Galip Cansever

Yıldız Technical University ______________________

Ass. Prof. Dr. Kayhan Gülez

Yıldız Technical University ______________________

Ass. Prof. Dr. Semih Sezer

Yıldız Technical University ______________________

This study was prepared and completed in Automation and Information Systems Chair at Munich Technical University.

ACKNOWLEDGMENT

This thesis work is conducted in Automation and Information Systems Chair in Munich Technical University within the ERASMUS MUNDUS program terms of agreement.

The preparation of this work has elevated my academic and theoretical vision, it helped me gain perspective and additional knowledge on fields such as system modelling and fault management, at the same time allowed me to practice my studies on PLCs.

Although I take full possession on the whole study from every line drawn to every word, I have to mention the key people whom have a huge part in the making of the project, and without them I wouldn’t be able to do this.

First of all, I am deeply and forever indebted to my wonderful family, especially my loving and supportive mother, my brilliant brother and my dear Duygu for all the magic spell she has casted on me and my work.

I send out my deepest love to Caner for being there.

I would like to express my vast gratitude to my great friends, Hande and Gürsu for all the motivation, Murat and Emre for all their precious help and inspiration, last but not least Ali Dinçer for opening doors for me and making it happen.

I am deeply grateful to Prof. Dr. Galip Cansever for believing in me all along.

I would like to acknowledge my professor, Prof. Dr. Ing- Birgit Voegel-Heuser. My sincere thanks go to Gülden Bayrak for the efforts she has dedicated to this study. It has been an extraordinary experience and a privilege to be a visiting student in TUM and in beautiful Munich.

August, 2011

Başak GENÇER

TABLE OF CONTENTS

LIST OF SYMBOLS

L	Observer Gain
K_i	Integral Gain
K_p	Proportional Gain
ζ	Controllability Matrix
Q	Observability Matrix
$H(s)$	Transfer Function
$G_c(s)$	Controller System Function
x	System State
$\hat{x}$	Observed System State
e	Observer Error

LIST OF ACRONYMS

SysML	Systems Modelling Language
LTI	Linear Tıme Invariant
PLC	Programmable Logic Controllers
AIS	Automatisierung und Informationssysteme
BDD	Block Definition Diagram
IBD	Internal Block Diagram
PAR	Parametrics Diagram
REQ	Requirements Diagram
FM	Fault Management
FMEA	Failure Mode and Effects Analysis
CFC	Continuous Function Chart
MRA	Multi-Representation Architecture
MDA	Model Driven Approach
P	Proportional Controller
PI	Proportional and Integral Controller

LIST OF FIGURES

Page

ABSTRACT

DESIGN, DEVELOPMENT AND EVALUATION OF A STATE-OBSERVER CONCEPT FOR INDUSTRIAL AUTOMATION PLANTS

Başak GENÇER

Department of Electrical Engineering

MSc. Thesis

Thesis Advisor: Prof. Dr. Galip Cansever

In automation plants it is a necessity to take into account that the sensor devices might experience faults within their lifecycles. Therefore, in order to increase the plant's dependability and efficiency, a compensation for the sensor faults is required. This thesis project presents an approach that provides fault management through replacement of the sensor values with the calculated values obtained from the state-observers in case of an unpredictable fault or hardware malfunction.

Initially, the plant chosen for implementation is going to be modelled in Systems Modelling Language (SysML). The model will include basic diagrams, constructing an idea about how the plant works and demonstrates the interconnections such as fluid exchanges.

The plant is going to be controlled by Programmable Logic Controllers (PLCs). The state-observer is going to be designed and simulated in MatLab/Simulink. The MatLab/Simulink model is going to be converted into the code that the PLC interface uses. In this thesis, it corresponds to TwinCAT, therefore the conversion is going to be made by the MatLab/Simulink code generator. The main fault management function is going to be implemented and realized first in MatLab/Simulink then on PLC software. While modelling approach is based on SysML, PLCs are going to be programmed in IEC 61131-3 languages.

The SysML model includes modelling of the parametric diagrams as well as block definition diagrams and internal block diagrams. The Parametric Diagram is going to be utilized to describe the continuous behaviour of the technical systems, i.e. the relationships between the component parameters.

Fault management is going to be handled by the state-observer. Designing and implementing the observer is going to shape the fault management concept as a whole. Finally the overall concept on the contribution of increased dependability is going to be demonstrated and evaluated on the process engineering part of a hybrid plant.

Key Words: Virtual Sensors, Estimation, State-Observer Model, SysML (Systems Modelling Language), Fault Management

ÖZET

ENDÜSTRİYEL ÜRETİM TESİSLERİ İÇİN BİR DURUM GÖZLEMLEYİCİSİ KONSEPTİ TASARLAMA, GELİŞTİRME VE DEĞERLENDİRİLMESİ

Başak GENÇER

Elektrik Mühendisliği Bölümü

Yüksek Lisans Tezi

Danışman: Prof. Dr. Galip Cansever

Otomasyon tesislerinde sensor cihazlarının kullanım süreleri boyunca bazı hatalar verebileceğini tahmin etmek bir ihtiyaçtır. Bu yüzden, tesislerin güvenilirliklerini ve verimliliklerini artırmak üzere, sensor hatalarının telafi edilmesi gerekmektedir. Bu tez projesi, olası bir hata durumunda ya da bir donanımsal arıza anında sensor değerlerinin durum gözlemleyicisinden alınan değerlerle değiştirilmesi yöntemine dayanan bir hata yönetimi metodu sunuyor.

Öncelikle, sıvı dolum tesisinin Systems Modelling Language (SysML) dilinde modellenmesi yapılacak. Modeldeki temel diyagramlar; tesisin nasıl çalıştığına dair bir fikir inşa edecek ve sıvı akış bağlantılarını sergileyecek.

Tesis Programlanabilir Lojik Kontrolörler'le (PLC) kontrol edilecek. Durum gözlemleyiciler MatLab/Simulink'te tasarlanacak ve simule edilecekler. MatLab/Simulink modeli PLC arayüzü kodu olan TwinCAT'e dönüştürülecek. Aradaki dönüşüm MatLab/Simulink kod üretici ile yapılacak. Hata yönetimi fonksiyonu öncelikle MatLab/Simulink'te sonra ise PLC yazılımda uygulanacak ve gerçeklenecek.

SysML modeli parameter diyagramlarını olduğu kadar blok tanım diyagramlarının ve dahili blok diyagramlarının modellenmesini içeriyor. Parametre diyagramı, teknik sistemlerin örneğin; bileşen parametrelerinin aralarındaki ilişkilerin sürekli davranışını tanımlamak üzere kullanılacaktır.

Hata yönetimi, durum gözlemleyici tarafından idare edilecektir. Gözlemleyicinin tasarımı ve uygulanması ise hata yönetimi konseptini bir bütün olarak şekillendirecektir.

algoritma kısa sürede yeteri kadar iyi sonuçlar vermektedir. Algoritmanın uygulanması için geliştirilen bilgisayar programı, tren dispeçerleri için bir karar destek sistemi olarak da kullanılabilir.

Anahtar Kelimeler: Sanal sensörler, Kestirim, Durum-Gözlemleyici Modeli, SysML (Systems Modelling Language), Hata Yönetimi

CHAPTER 1

INTRODUCTION

In this master thesis, an approach to fault management with “Sensor Fusion” method by the use of state-observers is presented. This approach is going to be implemented in the process engineering part of an experimental hybrid automation plant laboratory (will be called the plant henceforth) in the chair of Automatisierung und Informationssysteme (AIS) in Technische Universität München.

The plant’s basic hardware structure and its hardware behaviour layout are modelled in Systems Modelling Language (SysML) without denoting the controllers.

The details about the hardware components of the plant and their behaviours will be given in following sections to help in understanding how it is controlled. All hardware, regardless of being electrical or mechanical, are modelled in the Block Definition Diagrams (BDD). In the Internal Block Diagrams (IBD) and Parametrics Diagrams (PD), the relationships between inputs and outputs through constraints and mathematical functions are provided, in order to shape the whole system model and denote the extent of their behaviour boundaries and interconnections.

Fault management is implemented by Virtual Sensors method with the help of the state-observers. The virtual sensor method, as the name implies, is able to calculate the sensor output, a sensor can measure even when that particular sensor is not in service; hence can reproduce value, therefore it substitutes for a real sensor. This approach is put to practice through state-observers so that the sensor output can be continuously observed and compared to estimated values whether it is providing accurate measurements or not. If not, it will be able to replace the virtual sensor values with the real measurements.

Sensor fusion is described as combining a number of sensor data to improve performance by providing redundancy. For example, two different sensor data are processed in order to obtain extra information that does not exist as an element on the hardware level, hence the name: “Virtual Sensor”. [1]

Unlike the modelling part, the time behaviour is simulated in Matlab/Simulink and the state-observer is initially designed there in order to be evaluated in a simulation environment prior to the PLC implementation which will be in IEC-61131 programming standards.

The evaluation of the concept is going to be made by using the plant as the demonstrator. The system is expected to run only on the control software, without the real measurements of the sensors that was designed to be redundant. Therefore the state-observers’ functionality is going to be tested by observing whether or not the plant can continue and complete the process without that particular sensor successfully. The ongoing process will be filling and draining the tanks. The real sensor output and the observer outputs are going to be plotted in time domain and then compared to each other in terms of accuracy.

1.1 Literature Review

There are different backgrounds of scientific fields presented in this section. The keywords related to an industrial plant’s controlled behaviour are related in a way that they can often supplement each other although they bear small differences in essence. Efficiency, availability, reliability, stability, dependability and flexibility are differentiated carefully; even compared, for automation plants in the following. In general terms short descriptions for these terms are given below.

Efficiency is a term used when comparing competing experimental designs; with one design being more efficient than another if it can achieve the same precision with fewer resources. [2] Therefore effectiveness can be interpreted by the capability of all the efforts in an experiment, to produce accurate results. Availability is the ratio of the actual result of the effort to the capability.

Reliability is, in the context of this thesis, the resistance of a device/system to failure or the ability to perform and maintain its functions in routine circumstances, also just as good in emergency circumstances.[3]

Reducing the dependency on the physical sensors is made possible through various methods such as virtual sensors. There exists many ways for virtual sensor implementation and one of them is the utilization of the observers in control systems to improve system dependability.

Dependability is defined in many ways but a profound definition is the trustworthiness of a system which allows reliance to be justifiably placed on the service it delivers or operation it conducts. [4]

Dependability may include the meanings of: availability as readiness for correct service or operation and reliability as continuity of correct service or operation.

Flexibility is the ability of a system to respond to potential internal or external changes affecting its value delivery, in a timely and cost-effective manner. In a given perspective, it may be called adaptability to change. [5]

After all, the terms cannot be sliced apart from each other; each one of them contributes to the other’s existence.

The background of studies in “Sensor Fusion” has developed from a more general branch called “Information Fusion”. It is basically the composition of a new data derived from the existent data in a system. [6]

Fault Management (FM) being an older branch than data or sensor fusion has other facets in the literature, such as Network FM but in this work Sensor FM is the corresponding study area. Estimation methods and agent-oriented control systems can be referred to as the study fields in Sensor FM.

The academic works in the following have been reviewed shortly, a summary is derived from the previous works and the matrix of Focus vs. Research follows.

The focus points are covered completely in this project are Model-based System Engineering, SysML, Sensor Fusion, Virtual Sensors, Fault management, Hybrid

Systems, Availability, Dependability, Estimation, State-Observers. Following them, an outlook on future studies is given.

In the works of Andreas Wannagat and Prof. Dr. –Ing. Birgit Vogel-Heuser, the emphasis is on the agent-orientation as an encapsulated software unit with fault management purposes in automation systems.

In [11], a multi-agent PLC based control system has been designed to enable dynamic reconfiguration in a continuous time example. The agent-based system regulates each agent so that they have awareness for the effect of their actions, ability to estimate the significance of a changing environment, also to observe all other parts and to relate values of a real and a virtual sensor therefore enabling superior fault management.

In [12], an approach for run-time component failures has been developed. This approach also utilizes dynamic reconfiguration which is decided by rules from a knowledge-base. This self-adapting agent approach showing an example to the composition of the modular systems along with being beneficial to hybrid systems.

The study; "Reliability study of complex physical systems using SysML" [7] draws the attention closer to integrating reliability in the design process of the complex systems. However the underlying motivation of their research about reliable systems is more about the safety aspects. The main focus is the unification and enhancement of system development for safety critical and complex systems. The authors propose a method that links the functional design phase implemented in SysML with techniques in reliability area i.e., Failure Mode and Effects Analysis (FMEA), dysfunctional models using another software tool.

The new method proposed in this article enables analysis of the models expressed in SysML or UML in terms of obtaining new information about the dependability of the system. It is made possible to automatically compute FMEA from the SysML functional models and creating a dysfunctional behaviour database in SysML that includes failure rates, failure behaviour expressions derived through the failure laws of each FM. The functional model described in SysML is exploitable to identify the functional aspects expected to set up the model for reliability analysis.

It is however stated that there is a drawback of delay and complication in the design process.

In “Empirical Evaluation of SysML through the Modeling of an Industrial Automation Unit” [8], SysML is evaluated as a description language through the modelling of an industrial automation unit. The presentation of the complete plant in SysML is provided through three subsystems as physical, control and supervision.

It was derived that although learning phase costs a considerable time for those that do not have a previous knowledge of UML, the language shows every structure of a system through its diagram views effectively.

“Sensor Fusion Potential Exploitation—Innovative Architectures and Illustrative Applications” [1] defines an ideal sensor fusion. It suggests the benefits of central and local decision makers and their cooperation. It also discusses fusion system architectures’ flexibility then illustrating them with application examples to real-world scenarios. Finally, the approach is proven to be flexible in accommodating various types and variable number of sensors, able to recover from initial sensor errors, and to have rapid track label confirmation in most cases and robust against individual sensor failures.

“Hybrid Modeling and Diagnosis in the Real World: A Case Study” [9] has kept the focus on a supervisory controller performing discrete switching tasks through dynamic reconfiguration of the system components, or switching controllers. It is a case study of an aircraft fuel system, followed by a discussion of the methodologies for “online tracking of system behaviour and performing fault isolation and identification”. The hierarchical component based modelling, specifically a hybrid bond graph modelling approach, was utilized. The hybrid observer uses quantitative state space models to correctly estimate the parameters and match with the observations in order to avoid false alarming.

1.2 Thesis Purpose

The investors and/or business holders together with the engineers in various branches desire many common features in an automation plant. Today amongst those features, there are certain ones that are becoming imperative rather than necessary. The researches in automation and system engineering fields have a strong focus on the methods that would make a plant more reliable, flexible and adaptable for the reasons briefly explained below.

As stated by Gupta and Goyal in [10] "…manufacturing systems that are flexible can utilize the flexibility as an adaptive response to unpredictable situations." Flexibility itself, not only as a means of adaptation but also, alone as a feature is now gaining more attention, due to the result of the reformations in the modelling languages during the recent years. The need for a flexible plant and the growing robustness of the relatively new-established modelling languages such as SysML, feed each other in terms of importance and attention.

As a result of this matter's nature, the evolution of the end-products and the demands of the plant-owner cannot be easily anticipated by system engineers. Due to this reason, Oppermann states in [11] "the dynamics of changing conditions shifts the customisation process of the system's characteristics from the development phase to its usage and operation phase because the time needed for a professional development is too short or the new features are too costly." This derivation is making adaptability one of the major focuses in the field.

"Integrated System Model must address multiple aspects of a system" [12]. The plant's model is going to reflect the component structure as a whole hardware layout. In that structure, their behaviours are represented independent of any methodology therefore enabling better adaptation and flexibility. Eventually the model can be integrated into different methodologies upon user/designer preference and necessity.

The employment of a model-based diagnosis system on an automation plant often requires a sensor network. As stated by Henderson "Multiple sensors in a control system can be used to provide more information, robustness and complementary information." [13] and therefore the sensor network provides the grounds for Fault Management.

1.3 Hypothesis

The hypothesis of this study is to create an estimation mechanism for the continuous process part of a plant with the help of sensor fusion. However not only the estimation of the system states is a means of evaluation but also an extensive model of the complete hybrid automation plant is going to be presented.

The study process is made up of many steps, each one followed by another very closely.

The SysML model of the hybrid automation plant is done and followed by the design of the control loops in the continuous process. Then the candidate virtual sensors are formed before implementation. The different control loops are designed and simulated in Simulink. Among all, the best one was chosen and then modified to be compatible with the Code Generator tool to create an CFC model of the relevant Simulink model. And finally the CFC model in TwinCAT has been modified for virtual sensor ideas and the behaviour of the state-observer.

The major limitation to this project is that the level sensor in the continuous process plant is non-functional therefore it is not possible to prove a real-time evaluation between the observer data and the sensor data. However, instead of the real-time simulation the results will be given in Simulink, showing the output from the normal plant and the observed output.

Also the state-observers, by definition, need the input and the output information of a system to calculate its state-variables. Therefore the estimation of a sensor value in real time as an output is not completely possible only by the deployment of a state-observer.

Another limitation to the observer implemented components is that the discrete parts of the plant are not suitable for observer implementation; this is again by the definition that observers are not functional in discrete time models. An example to this is the temperature control loop that is mounted within the Workstation.

Posed questions are given in the following. The answers to those are going to be stated in the Discussion and Recommendations chapter along with the path forward.

- Is sensor fault management possible in run-time systems?
- Is the virtual sensor concept feasible for the hybrid automation plants?
- Is the virtual sensor concept applicable for the hybrid automation plants?

- Can sensor fusion be applied for all the plant components?
- Can we use these models in an integrated manner with automated consistency checking and in a predictable way for future product development?
- Are distributed systems as dependable and efficient as the centralized solutions?
- Is the model driven approach an effective method of system availability and fault management?

CHAPTER 2

MODEL DEVELOPMENT

2.1 Introduction to Model Development and Systems Modeling Language

The Systems Modeling Language (SysML) is defined as: "The OMG systems Modeling Language (OMG SysML™) is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. In particular, the language provides graphical representations with a semantic foundation for modeling system requirements, behaviour, structure, and parameters, which is used to integrate with other engineering analysis models." It offers a multi-representation architecture (MRA) [14] and also Model Driven Approach (MDA). MRA is relatively a new field but there is more to it than what meets the eye. The idea behind MRA is to offer alternatives to data mapping between different hierarchies of programs. The common purpose of MRA and MDA usage in system models can be summarized with this phrase: to make something once and reusing it several times.

The baseline is, with SysML the structures of an element or a system can be described in a single unified concept therefore it enables engineers from different branches to be able to work and design together on a single model in the correlations of their fields of expertise.

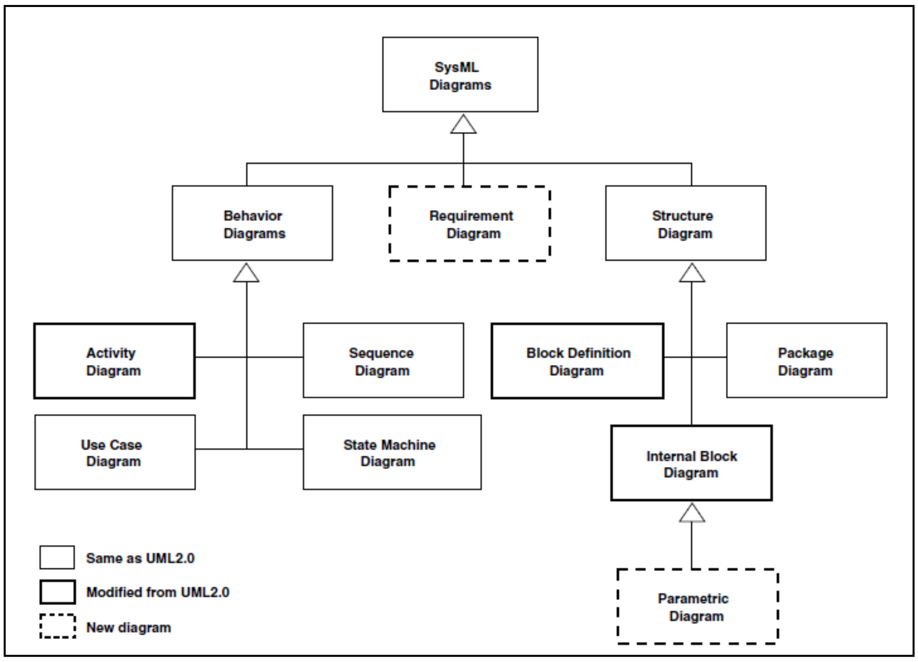


Figure 2.1 SysML Representations

As seen in the Figure 2-1, a SysML model features many diagrams i.e., Behaviour Diagrams, Requirement Diagrams and Structure Diagrams. In the thesis, Block Definition Diagrams (BDD) and Internal Block Diagrams (IBD) are going to be utilized for the structural system model and the Parametric Diagrams are going to be introduced to simply model the mathematical relationships and later compared with a simulation based model of the same system made in Simulink.

The Model-Based systems engineering approach is defined in the book "A Practical Guide to SysML" [15] through two phases which are not abstract, on the contrary blended together throughout the development phase as a whole. First, the model has to be organized and the libraries have to be identified by the following[16]:

1. Modelling behaviour and elaborating as necessary, similar to the problem statement
2. Modelling structure by capturing implied inputs and outputs, the system flows
    - Identifying structural components and their interconnections
    - Allocating behavioural flow onto interconnections
    - The requirements and the assumptions have to be made clear.
    - Capturing and evaluating parametric constraints.

Modelling a system in SysML will enable certain capabilities. It allows increased knowledge capture, traceability and modularity. When a system engineer needs knowledge about the system for back-tracing or interoperation planning, (s)he can easily obtain it from the SysML model rather than complex mathematical simulations or bare flow-chart schematics of the system. SysML is an efficient modelling methodology between methods that are complex in detail or else merely austere.

While manufacturing systems need to interact in various and complex ways, SysML could satisfy the need for concurrency and provide simultaneousness in a modelling tool.

As another benefit, SysML also reduces modelling efforts and therefore saves the system engineer valuable time and costs. [17] It is possible to cross connect the structure and behavioural aspects in SysML with requirements and parameters belonging to the structure in question.

What is done with the model is essentially the mapping of the components with their responsibilities or their functions within the system or subsystem level requirements.

The allocation relationship of SysML provides an effective means to capture this assignment and allow the navigation between the system models by establishing cross-cutting relationships among them. [18]

2.2 Modeling Constucts in SysML

In this section, the basic elements used in the diagrams, namely package structures, as well as the diagrams themselves will be described.

The Block Definition Diagram (BDD) in SysML, defines features of blocks and relationships between blocks such as associations, generalizations, and dependencies. It captures the definition of blocks in terms of properties and operations and also relationships such as a system hierarchy or a system classification tree.

The Internal Block Diagram (IBD) in SysML, captures the internal structure of a block in terms of properties and connectors between properties. A block can include properties to specify its values, parts, and references to other blocks. [19]

BDD, as the name implies, gives information about the definition or type of a structure/element and captures its properties, whereas IBD depicts usage, the element’s role in the system.

The Requirements Diagram (REQ) depicts what an element requires for the operations they are constrained to do. It specifies the capability or condition that should be satisfied, traced and/or allocated by the system.[8] For example in a PI controller, the system would require the controller to obtain the gain parameters and the transfer function to be able to deliver the correct plant input also meaning the controller output. All these interactions are listed and symbolized in a REQ to fully describe what a system needs during its operation.

The Parametrics Diagram (PAR) enables the designer to enclose the mathematical equations between the parameters of the system. It does not yet provide a simulation environment; however there is a path-forward for various studies on tool development for the type of conversion that would enable a correctly coded PAR to be transformed into an enhanced simulation environment such as Matlab or a software where it is possible to work on the system’s time behaviour.

However before all, the structure has to be laid out using the blocks as the fundamental element. The cornerstone of a diagram is a block. It describes a part, without any specific definition of the structure. [16] An element belonging to a block can be typed by another block. The element defines a local usage of its defining block within the specific context to which it belongs i.e., the valves in the system. This is how instances for these elements are created and used. For example a manual valve structure is defined once in the main BDD and then named differently after other different valves in the system therefore creating just as many different instances that can have different values. This is also the same case with the following model constructs.

A Flow Port describes an interaction point of a block. It is used by the block to interact with its environment and objects can flow into and out of the block over this port. In general, ports specify types of interactions allowed between blocks, in this case a flow port may symbolize i.e., signal, information, liquid flow etc.

A constraint is a condition that always has to be met, and which restricts the semantics of model elements. Therefore, a constraint block demonstrates the boundaries of operations by providing statements in the description part of the block.

Eventually with this model, an engineer can distinguish both between model and diagram and between structure and behaviour. The area of the elements to which neither structure nor behaviour can be allocated is much larger in SysML than it is in UML.[16]

2.3 Hybrid Automation Plant and its SysML Model

In the plant, there are two identical Filling Stations and one Workstation which is placed in the middle of the two Filling stations.

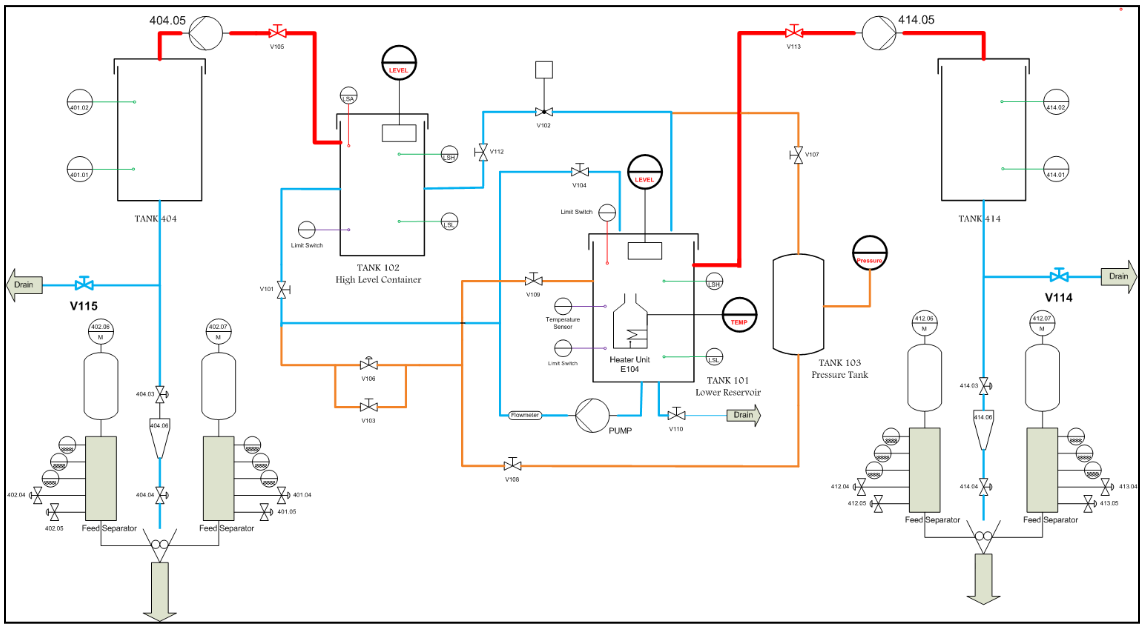


Figure 2.2 Plant schematics

The general view of the Hybrid Process Model is as given below. The Filling Station and Workstation are connected to each other with two aggregation lines originating from Workstation to create 2 different instances of the Filling Station. With this association FS1 and FS2 instances, for Filling Stations left and right respectively, are created.

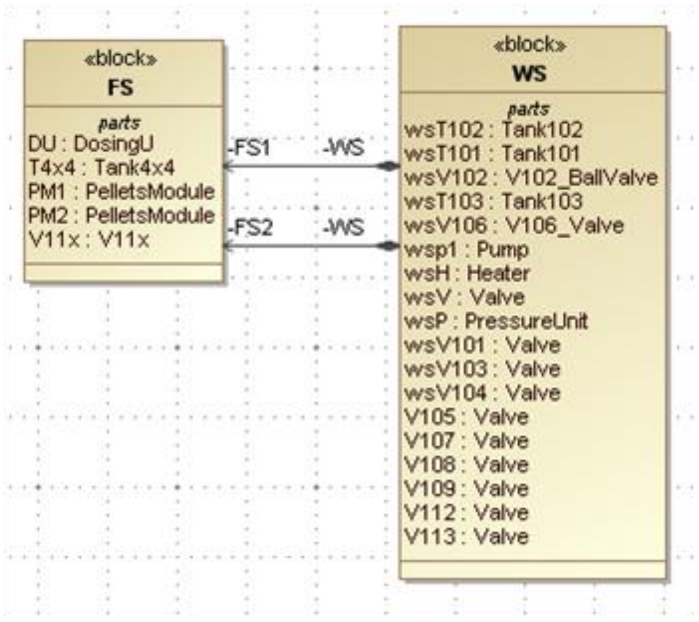


Figure 2.3 BDD of the hybrid process model

2.4 The Workstation Subsystem's Structure and its SysML Model

The hardware components and the functionalities that they provide are given in this section. It is followed by the SysML model of the complete workstation including all the hardware that an engineer should be wary of.

2.4.1 Components and Functionality

The main mechanical components are 2 containers (water tanks; high-level container and lower reservoir), pressure tank, valves and plug-in piping system.

The electrical components are 2 capacitive sensors, 2 float switches, ultrasound sensor, flow sensor, pressure sensor and PT100 temperature sensor. The specific valves are proportional directional control valve and ball valve which manipulates the flow direction between high level container (Tank 102) and lower reservoir (Tank 101) with pneumatic drive.

The Programmable Logic Controller (PLC) used to control the plant is Beckhoff CX1100-0004.

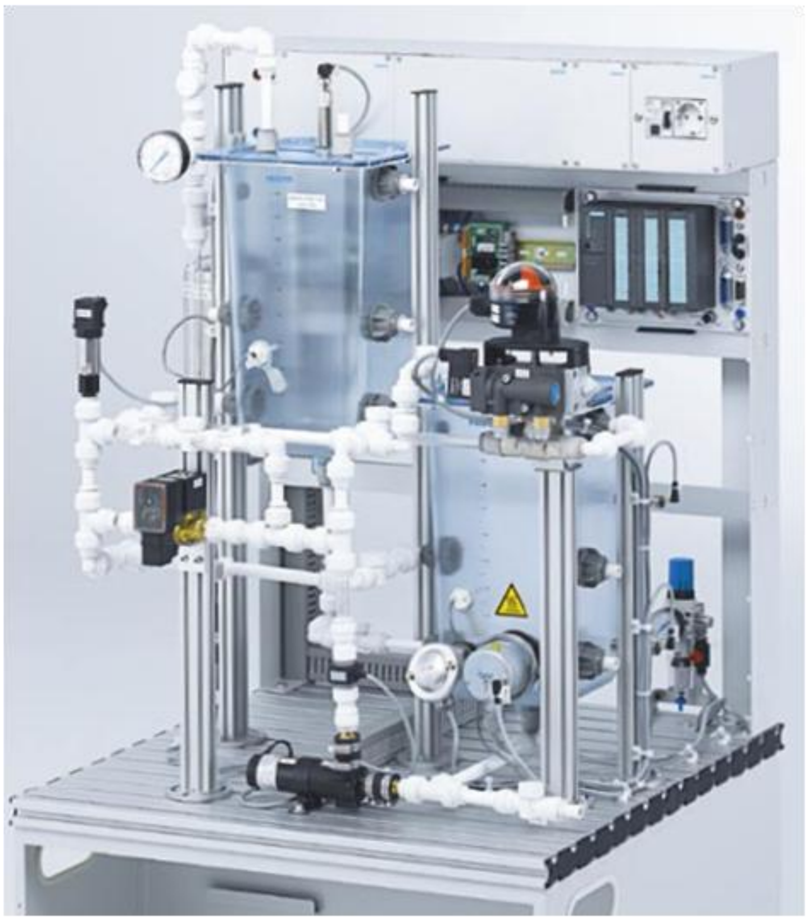


Figure 2.4 A Representative Image of the Workstation [20]

The active components necessary for the level, flow-rate, pressure and temperature controlling are the pump, proportional directional control valve, ball valve with pneumatic drive, heater and pressure tank along with the other electrical measurement devices.

If the controller design is made accordingly, the level and flow rate control systems can be cascaded together. As a result of its convenience, P controller is chosen for implementation here, among the other possibilities such as PI or PD controllers and also 2 point control.

Temperature control in Tank 101, flow-rate (level) control between Tank 102 and Tank 101 and Pressure control in Tank 103 are other implementation possibilities that belong to the model.

The level control loop is in charge of keeping the water level in Tank 102 on a certain level. It takes the flow-rate measurements from the relevant sensor mounted on the output of the pump motor, which pumps the water from Tank 101. The maximum level sensor on the Tank 102 provides the signal that the tank is full and this feedback turns of the pump motor.

The temperature control loop uses the characteristics of the PT100 temperature sensor controlled by the internal micro-controller of the heater unit. This loop can not start unless Tank 101 has enough water. The floating switch is activated when the heater unit is submerged into the water. The heater starts heating until the measured temperature has reached 50-60 °C and then turned off. If the heater controlling boolean is still set to true, it starts again when the temperature drops below 45 °C. There is also a semiconductor temperature limit to the heater unit’s operation. Once the semiconductor’s temperature is above 90 °C it is switched off internally until it cools down to 85 °C.

The loop for level control is functional for Tank 101 and Tank 102. They are filled in turns, when Tank 101 is full, the heater starts and water gets heated. Then when the heater stops, the motor pumps the water up to the Tank 102. Once the Tank 102 is full, the ball valve releases the water to the Tank 101 without the help of the pump.

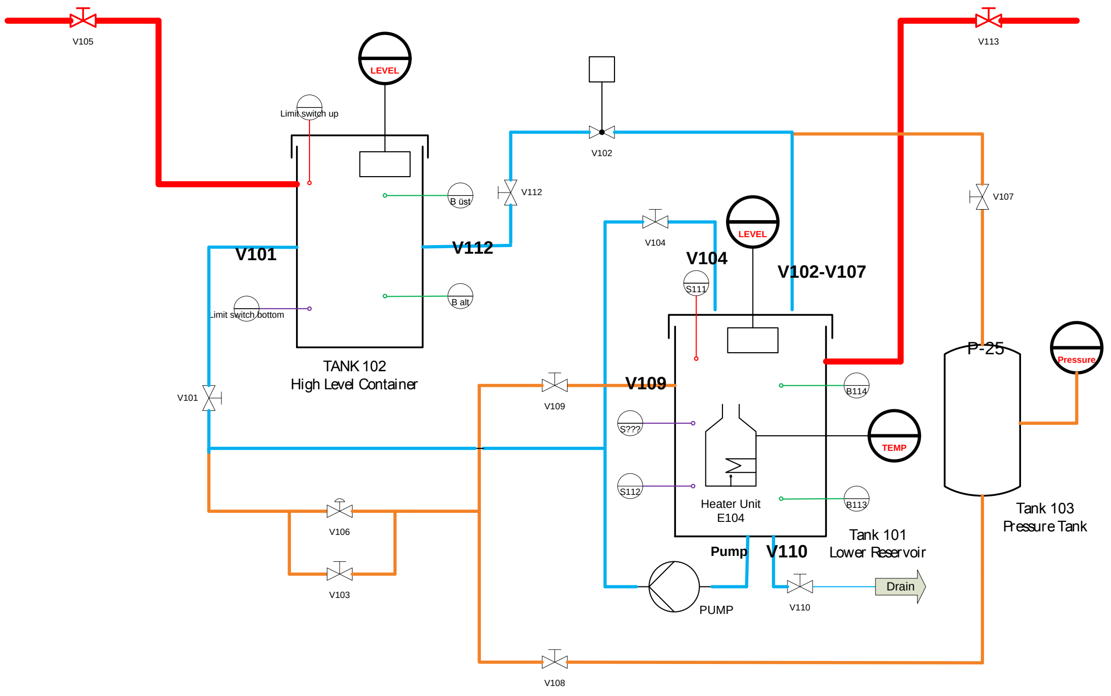


Figure 2.5 Workstation process diagram

The process diagram of the Workstation is given above.

2.4.2 The SysML Model of the Workstation

In the general view of the workstation BDD, it is possible to see all the hardware elements that embody the workstation as a whole; these include the water tanks, the pump, the pressure unit, the heater and different blocks for the different types of valves. The PI controller is not one of the hardware components therefore modelled as a requirement block to point out the difference.

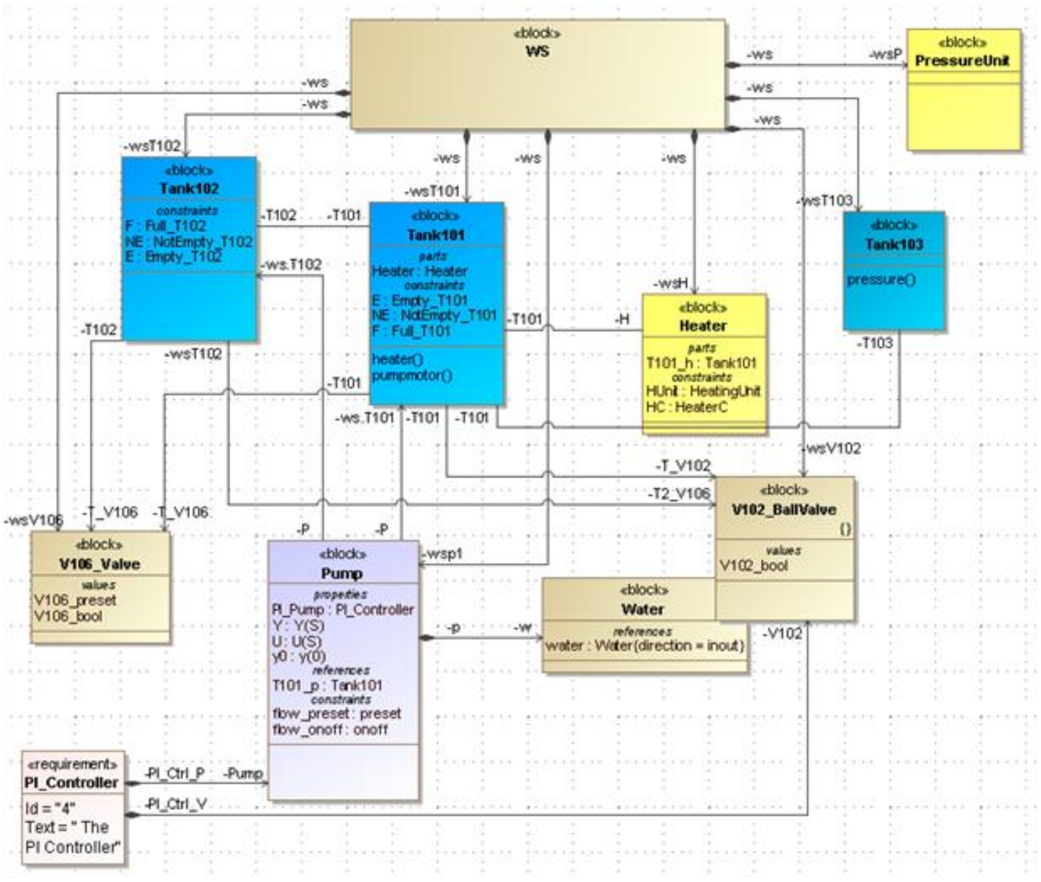


Figure 2.6 BDD of workstation

The BDD's and IBD's are modelled for main elements such as Tank 101, Tank102 and so on, also with the sensors they have accordingly. The sensors are represented as flow ports; they are categorized into two types such as digital and analogue in this model. In Fig 2-5 and Fig 2-6, the BDD's of Tank 101 and Tank 102 are given with the constraint blocks associated to each one of them. There are components with attached sensors and actuators or the components that the liquid flows through. Those are Tank 101, Tank 102, Tank 103, Heater, Pump and Pressure Unit. There are some limitations to their processes. These limitations correspond to constraints, which are modelled in the constraint blocks in each component's Block Definition Diagrams and Internal Block Diagrams.

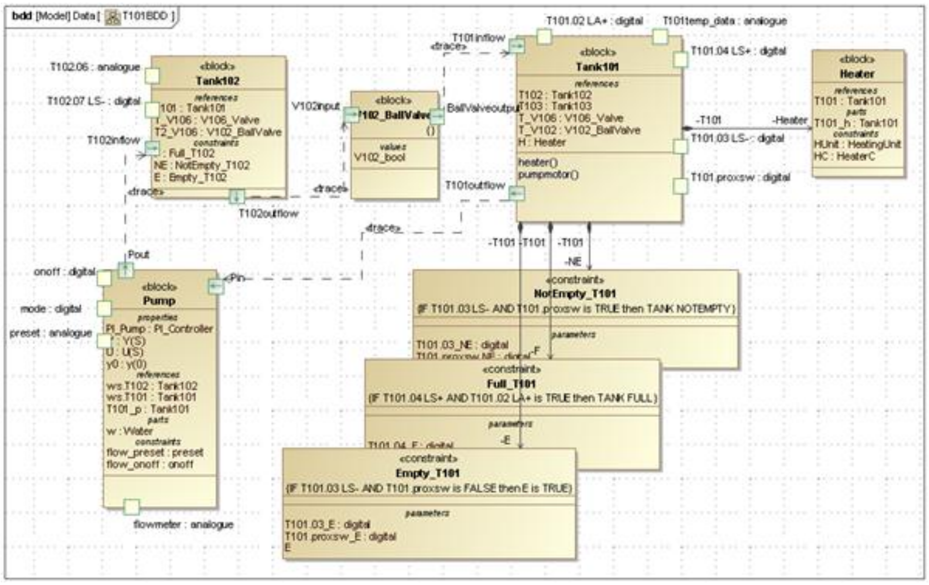


Figure 2.7 BDD of Tank 101

The Tank 101 has 3 constraint blocks showing 3 different conditions related with the water level in the tank, as NotEmpty_T101, FullT101 and Empty_T101.

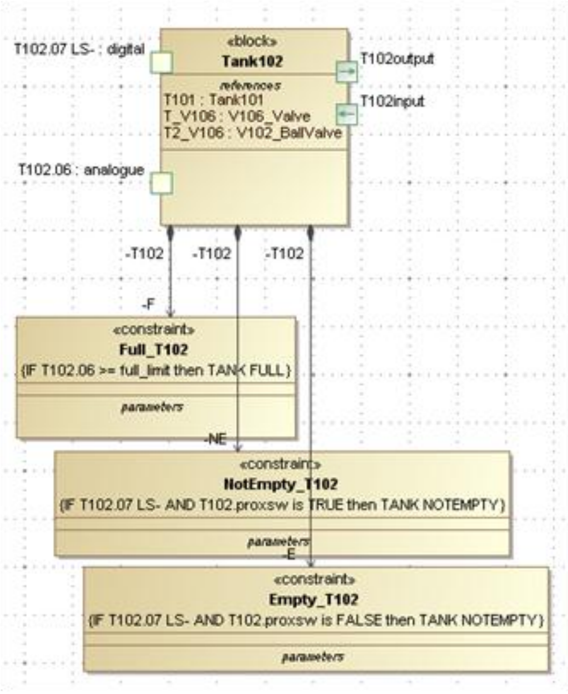


Figure 2.8 BDD of Tank 102

In the Tank 102's constraint blocks, the tank conditions are given with the constraints related to them. For example, the provided statement for "Full_T102" is that the level sensor on the tank measures the predefined level that corresponds to full.

The Pump System BDD shows the interactions between the pump and the tanks. From this diagram, it can be seen that the tanks are common elements for the pump's operation and the pipes are common between the tanks therefore are also utilized for the pump's operation.

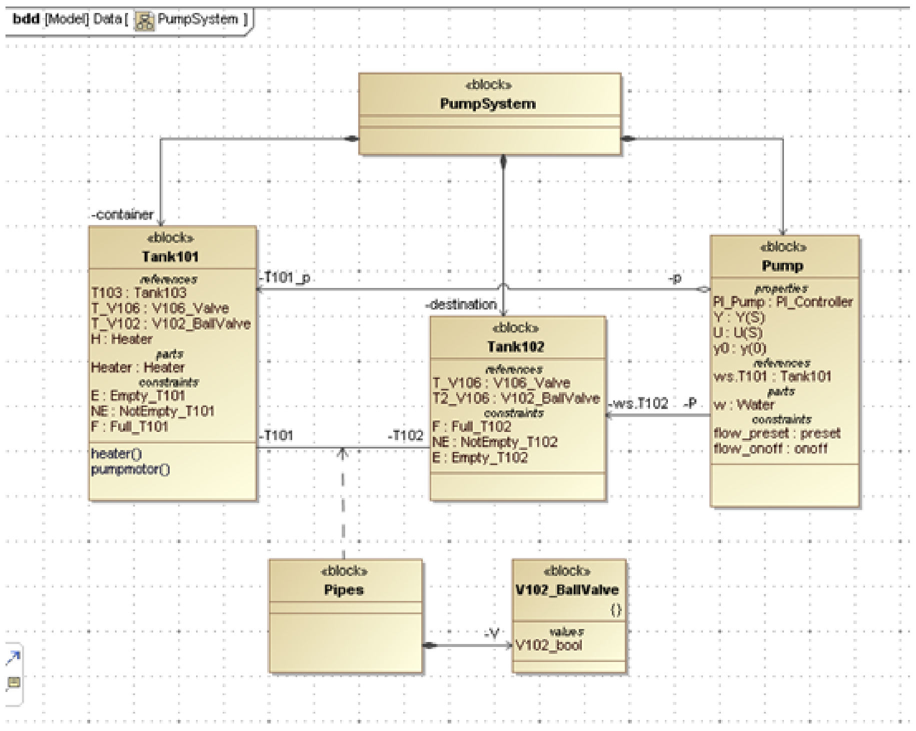


Figure 2.9 BDD of the pump system

The Pump BDD shows the constraint blocks modelling what the pump needs to operate or what are the limitations to its operation. There are two constraints, "onoff" and "preset" that state the two different control conditions of the pump, such as on-off control digitally with a boolean and entering a preset value for the pump to operate.

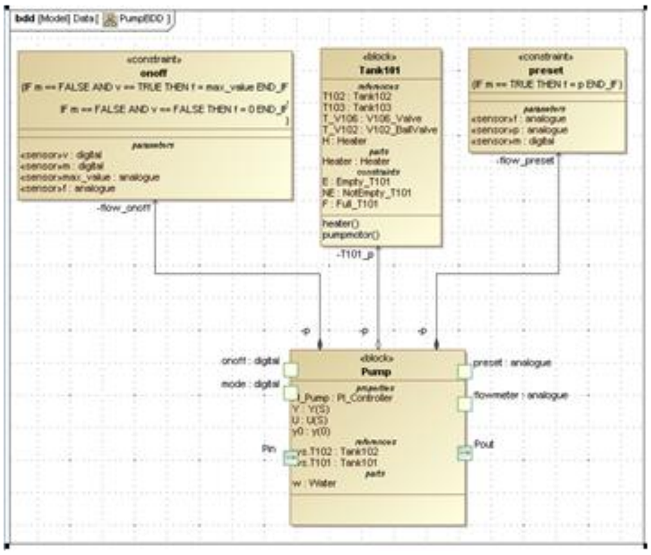


Figure 2.10 BDD of the pump

In the IBD of the pump it is possible to see the item flow between defined blocks; in this case a liquid flowing between Tank 101 and Tank 102. This provides a component specific view, augmenting the interconnections with other components.

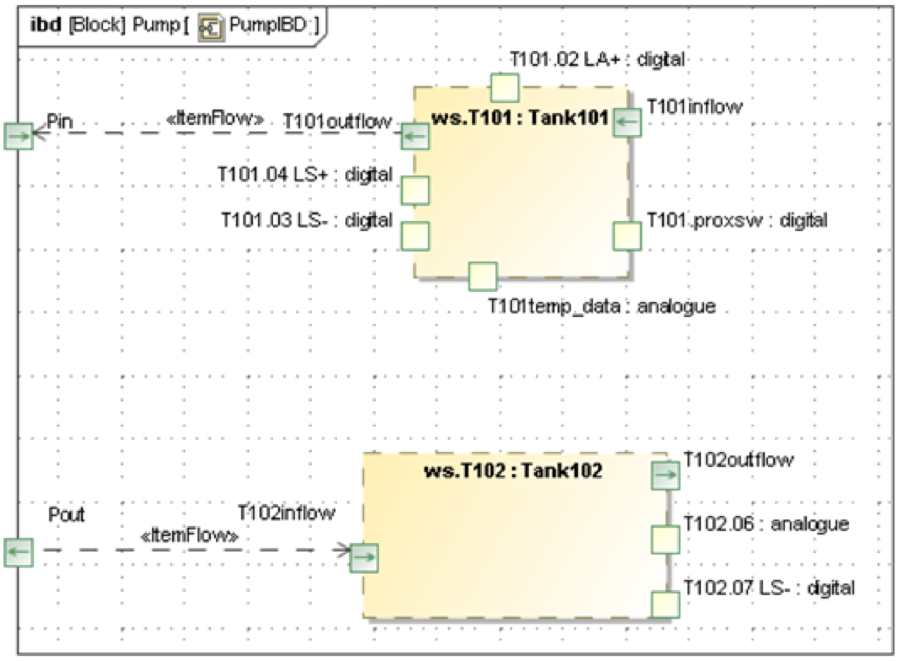


Figure 2.11 IBD of the pump

The BDD designed for the Control System is responsible for a specific loop that is implemented for the purpose of this thesis. It includes the PI controller block, showing the signal originations. From this view it is possible to see what the control loop interacts with and in which terms.

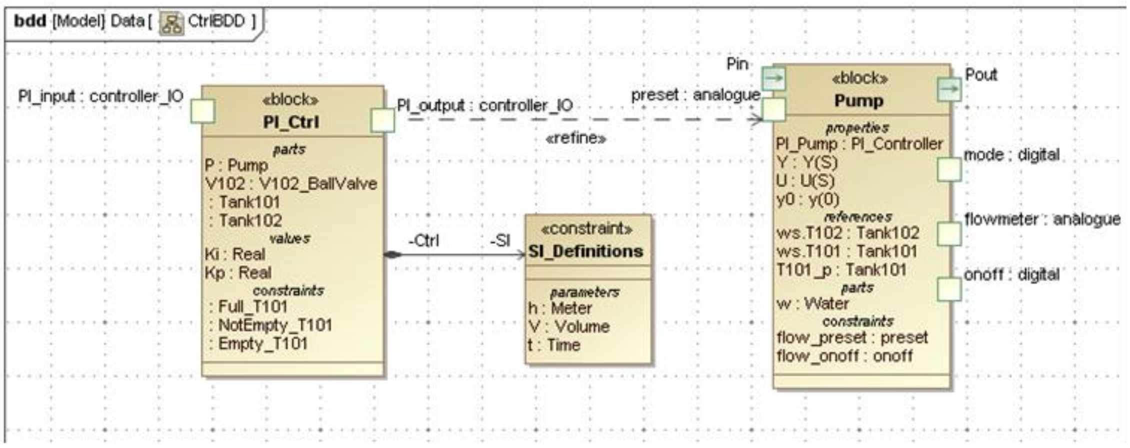


Figure 2.12 Control system BDD

The detailed view of the pump constraints are modelled in the Parametrics Diagram. The constraints of the pump block determine either the pump is working on digital mode or analogue. If it is working in analogue mode, the preset value is used to adjust the flow-rate.

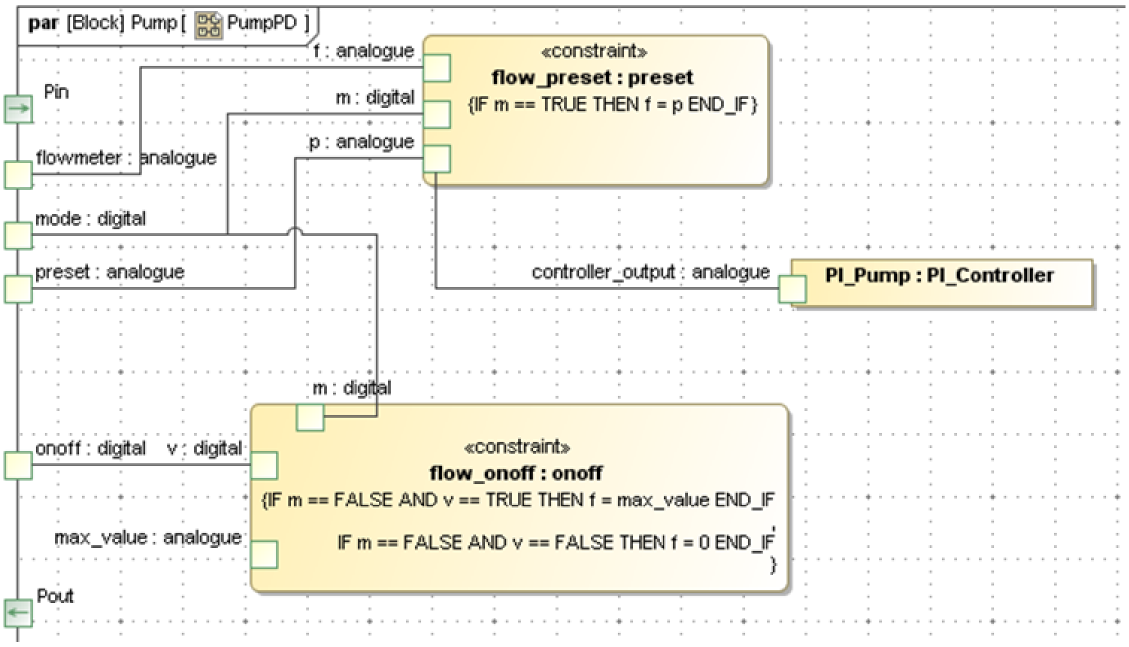


Figure 2.13 PD of the pump

2.5 The Filling Station Subsystem Structure and its Model

The filling stations are located on each sides of the workstation. One of the each outflow valves of the Tank 101 and Tank 102 are connected to the left and right filling stations respectively.

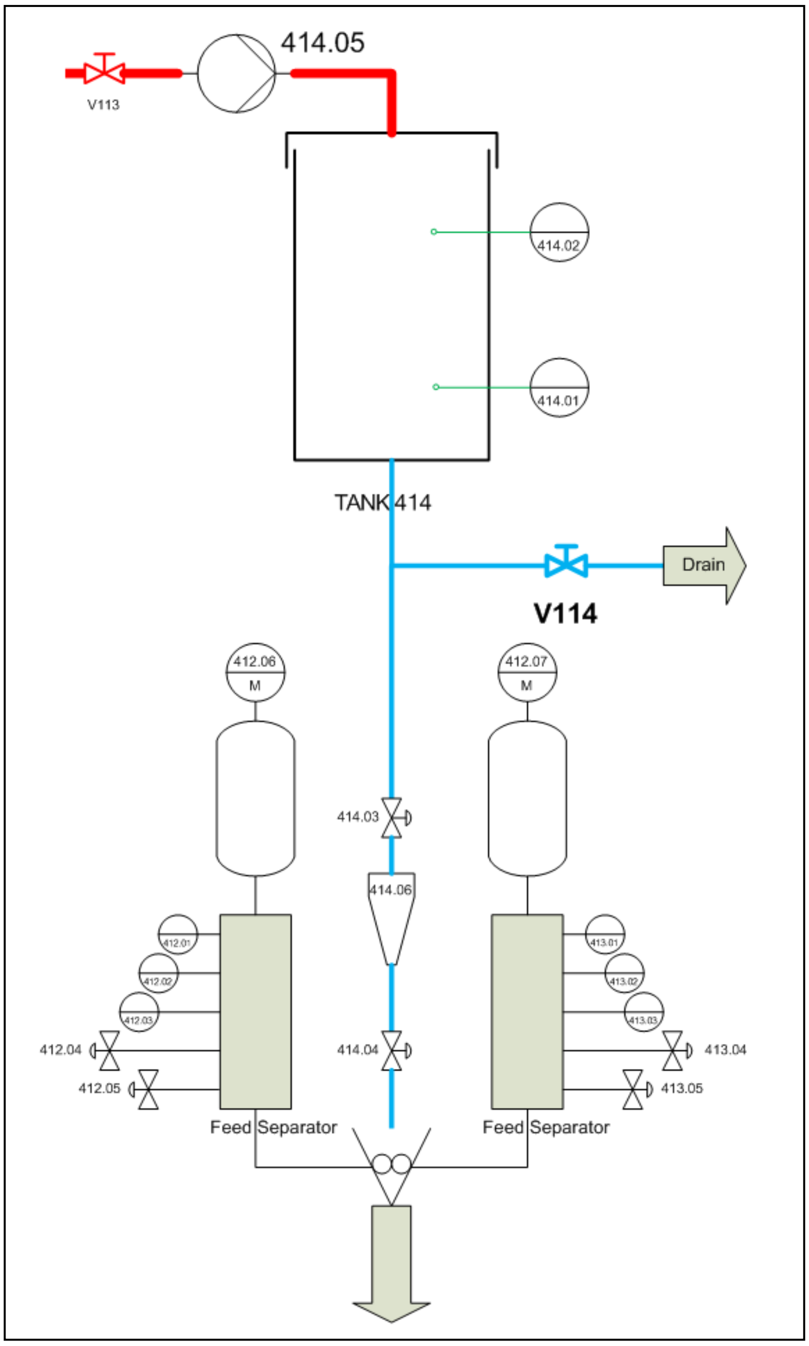


Figure 2.14 Filling Station process diagram (Right)

2.5.1 Components and Functionality

Since the two subsystems are the same, one of them will be described here. The two subsystems can be differentiated by the naming conventions in the model, i.e., for the left side Tank 404 and for the right Tank 414.

There is a water container which has two digital level sensors as in Tank 101. It has a pump connected to it to pump the water from the connected tank in the workstation and a drain pipe with a manual valve to empty the tanks. The other outflow from this tank that supplies flow to another process is through the electromagnetic valves and a dosing unit which is a type of valve as well.

Underneath this group of equipments, there are two identical pellet modules for each filling station. In each module, there is a container with a pump to mix the ingredients. There is a feed separator that receives the work pieces from the container; it has 5 different valves installed.

2.5.2 The SysML Model of the Filling Station

The filling station components are modelled in the Block definition diagrams and Internal Block diagrams as below. Since there are two identical Filling Stations in the system, it is sufficient to model one of them in a BDD and then create two instances by giving different names and in this case FS1 and FS2 as already given in the beginning of this section.

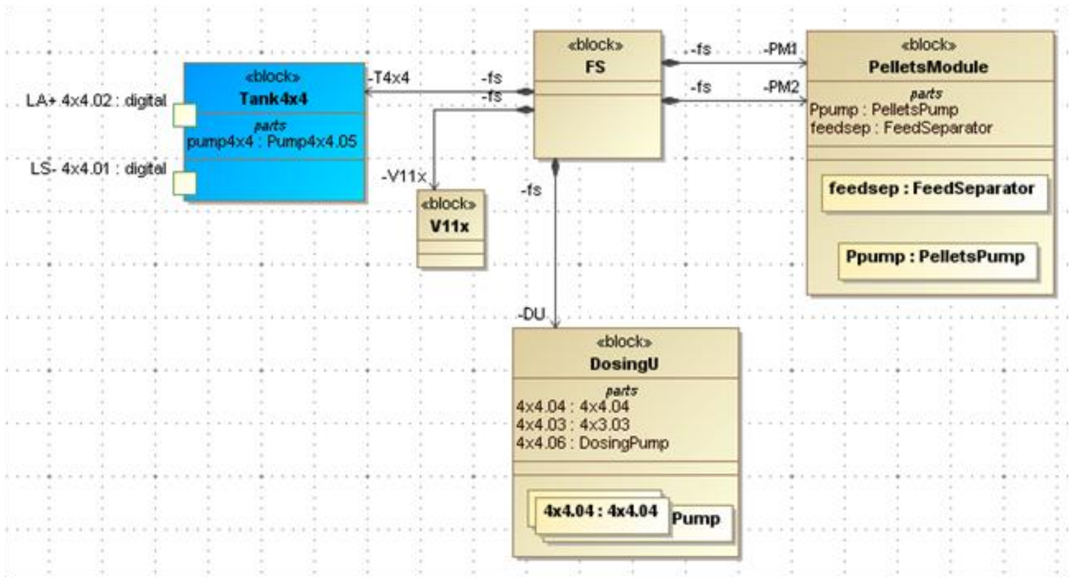


Figure 2.15 BDD of the filling station

The BDD of the Filling Station includes a dosing unit, one tank, valves and two identical “Pellets Modules” which include a Feed Separator and a pump for the particles.

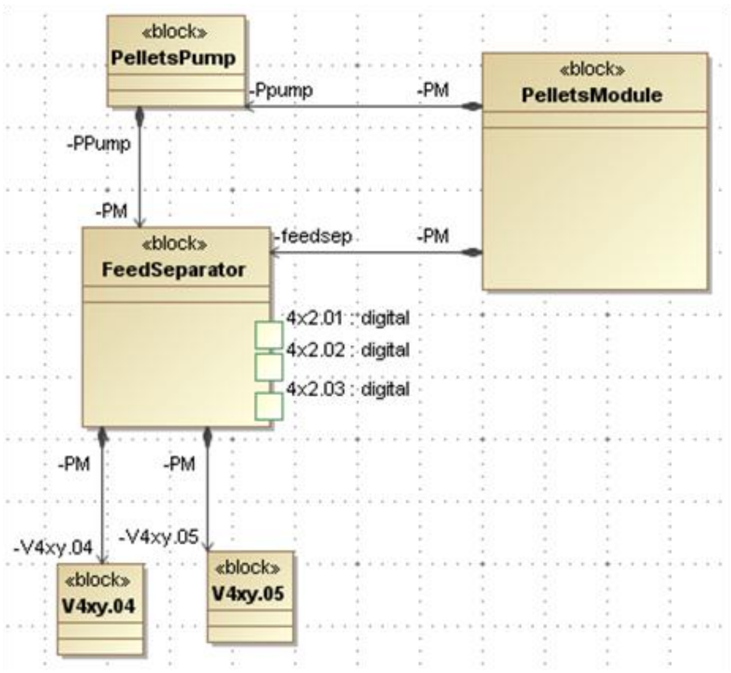


Figure 2.16 BDD of feed separator

Feed Separator has 3 digital sensors, 2 electromechanical valves, while the pump and the Pellets module are connected to it in order to create instances for the Feed Separator.

CHAPTER 3

IMPLEMENTATION

A part of the system is modelled in Matlab/Simulink in order to obtain simulation results for the controller and the state-observer implemented in the mentioned part which includes the high level container (T102), the lower reservoir (T101), the actuators between them such as the ball valve (V102), the water pump and various sensors in the workstation.

The desired process for this part of the plant is an automated fluid circulation between tanks and level control. The implemented methods are going to be presented in this section, starting with the introduction of the sensor fusion method, then controllers and the state-observers.

Different methods have been designed for the level control process. PI, PD and P controllers are designed and simulated for the same purpose in Matlab/Simulink. Then only for two of the controller models, the state-observers are designed and simulated to be able to evaluate their time-behaviours. The best fit for the system is chosen according to the simulation results. The chosen controller and observer combination is converted into TwinCAT CFC diagram with the help of the Matlab/Simulink Code Generator tool [21].

3.1 Sensor Fusion

According to (Ellis, G.2002) [22] to enhance reliability, reducing the cables connections and component count is proven to work where electrical contacts are the least reliable elements in a system today.

Sensor Fusion in theory has different levels of implementation such as data, feature and decision levels, given in an increasing order of abstraction. The different levels serve to different functional purposes in a variety of areas from digital signal processing to artificial intelligence. [1]

The virtual sensor, in the context of this thesis, is a soft interface for an ultrasonic level sensor that normally supplies the feedback signal to the control system. The sensor data is crucial to complete the closed loop of the controller and it also enables the observer to be able to estimate the necessary system states. In short words it is an artificial formation of the measurement, in order to be used when the sensor is out of order as it is in the Hybrid Automation Plant that this study is conducted.

3.1.1 Implementation of the Virtual Sensor in the Process

As previously stated in Limitations section, the aforementioned level sensor is not functional. Therefore the rest of the sensors in the system that provide relevant data, are used to create a virtual sensor good enough to replace the level measurement data provided by the real sensor.

The level data closest to reality could be calculated with the help of the flow-meter output $p(t)$, combined with the manually given value of the initial water level (y_0) in the tank. Since the flow-meter outputs a value in litre per minute, it will be integrated to obtain a volume unit and then added to the initial water level in the tank that is given by the user.

$$y(t) = y_0 + \int p(t).dt \qquad (3.1)$$

The system will start with the initial water level of 45 mm in the Tank 102 which corresponds to the higher capacitive sensor (B114) in the Tank 101. Therefore the equation is going to be as given below.

$$y(t) = 45 + \int p(t).dt \qquad (3.2)$$

It is also possible to assume that 45 mm is the minimum level of the Tank 102 and directly declare that point as zero in the TwinCAT code. The reason will be explained later in the controller selection process in details that it is possible to declare the minimum water level, which corresponds to 45 mm, zero in the control software.

3.2 System Parameters and Equations for Level Control

System's output parameter is $\mathrm{y(t)}$ which is the liquid in the tank in liters. There is $\mathrm{p(t)}$, the water volume per minute pumped in to the tank (l/min) as $\mathrm{v_{in}}$. System's input parameter is $\mathrm{v(t)}$ which is the liquid volume per second that leaves the tank (l/min) as $\mathrm{v_{out}}$. To be able to calculate how many liters of liquid there is in the tank, we need the initial volume which is denoted by $\mathrm{y_0}$, the liquid already in the tank in liters. Therefore, the system functions in time domain is formulated and the block diagram is shaped as below.

$$y(t) = y_0 + \int p(t).dt - \int v(t).dt \tag{3.3}$$

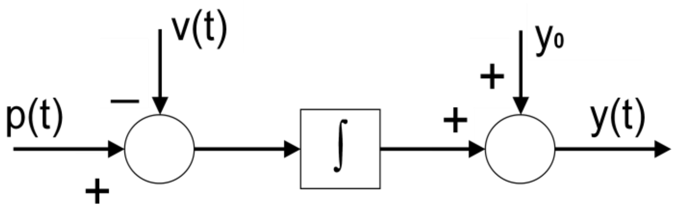


Figure 3.1 Block diagram of the system parameters

If the Laplace transformation is applied to the above equation, the system function is derived in s domain.

$$Y(s) = \frac{y_0}{s} + \frac{P(s)}{s} - \frac{V(s)}{s} \tag{3.4}$$

$$s.Y(s) = y_0 + P(s) - V(s) \tag{3.5}$$

$$s.Y(s) - y_0 = P(s) - V(s) \tag{3.6}$$

If the initial conditions are assumed to be zero, then the function would be as given below. According to the naming conventions, the system inputs are named after u and the outputs are y.

$$y_0 = 0\ ,\ V(s) = U(s) \tag{3.7}$$

$$s.Y(s) = P(s) - U(s) \tag{3.8}$$

3.3 PI Controller

According to the definition, proportional and integral controllers use the proportional and integral gain to multiply with the error and then add them together to provide an input to the plant.

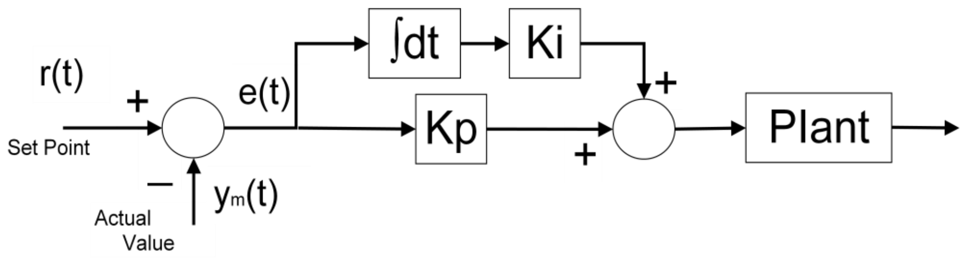


Figure 3.2 Block diagram of a PI controller

The PI controller for the system is formulated as can be seen in discrete form in the block diagram.

$$G_c(s) = K_p + \frac{K_i}{s} \tag{3.9}$$

$P(s)$ is placed in the s domain system function and initial condition is assumed to be zero.

$$Y(s).K_p + \frac{Y(s).K_i}{s} - s.Y(s) = U(s) \tag{3.10}$$

$$s.Y(s).K_p + Y(s).K_i - s^2.Y(s) = s.U(s) \tag{3.11}$$

$$Y(s)(s.K_p + K_i - s^2) = s.U(s) \tag{3.12}$$

From here, we derive the Transfer Function of this controlled system as below:

$$\frac{Y(s)}{U(s)} = \frac{-s}{s^2 - s.K_p - K_i} \tag{3.13}$$

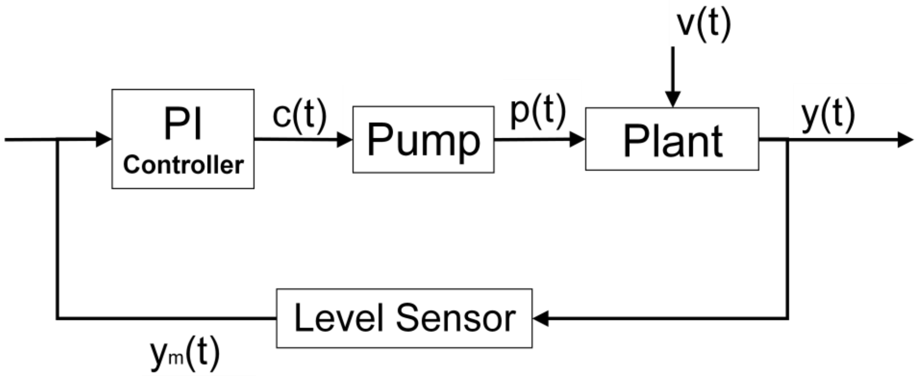


Figure 3.3 Block diagram of the PI controlled system

By doing manual tuning, the K_p and K_i values for the PI controller have been derived after several iterations for faster implementation as below:

$$K_p = -1 \quad K_i = -10 \tag{3.14}$$

The transfer function is changed by adding the gain parameters into the equation.

$$\frac{Y(s)}{U(s)} = \frac{-s}{s^2 + s + 10} \tag{3.15}$$

$$s^2 . Y(s) + s . Y(s) . K_p + Y(s) . K_i = -s . U(s) \tag{3.16}$$

The inverse Laplace of the unit step response of the function is calculated in Matlab using the `ilaplace(f)` function.

$$y(t) = \frac{-10}{\sqrt{{K_p}^2 + 4K_i}} . e^{\left(\frac{1}{2K_p} + \frac{1}{2\sqrt{{K_p}^2 + 4K_i}}\right) . t} - e^{\left(\frac{1}{2K_p} + \frac{1}{2\sqrt{{K_p}^2 + 4K_i}}\right) . t} \tag{3.17}$$

3.3.1 The State-Space Notation of the PI Controller

The general notation of the state-space matrices are as seen below.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{3.18}$$

It is directly written from the already derived transfer function without the gain values.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} K_p & K_i \\ 1 & 0 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.19}$$

$$\boldsymbol{y} = [\mathbf{1} \quad \mathbf{0}] \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{bmatrix} \tag{3.20}$$

With the gain values filled, the matrices will be as below:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -1 & -10 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.21}$$

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.22}$$

3.3.2 Observability Check for the System

To check whether the system is observable or not, the observability matrix is formed and checked for singularity. Controllability matrix is $\zeta = [B \ \vdots \ AB]$ and observability matrix is $Q = [C^* \ \vdots \ A^*C^*]$.

$$\zeta = [B \ \vdots \ AB] = \left[\begin{bmatrix} 1 \\ 0 \end{bmatrix} \vdots \begin{bmatrix} -1 & -10 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right] = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \tag{3.23}$$

$$Q = [C^* \ \vdots \ A^*C^*] = \left[\begin{bmatrix} 1 \\ 0 \end{bmatrix} \vdots \begin{bmatrix} -1 & -10 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right] = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \tag{3.24}$$

In this case, as seen from the results, the matrix is non-singular for both and therefore the plant is controllable and observable at the same time.

3.3.3 Matlab/Simulink Model for PI Controller

The first Simulink model had the conventional PI block from Simulink library. The system is designed in s domain. In this model, the output of the controller block is the flow-rate of the liquid flow into the Tank 102, which is the pump output measured by the flow-meter. Before the controller block, the measured flow-rate is subtracted from the tank's outflow which is symbolized as a step input. The result of this subtraction is integrated from litre/min to litres. When added with the initial liquid volume in the tank, it equals to the tank's present volume value.

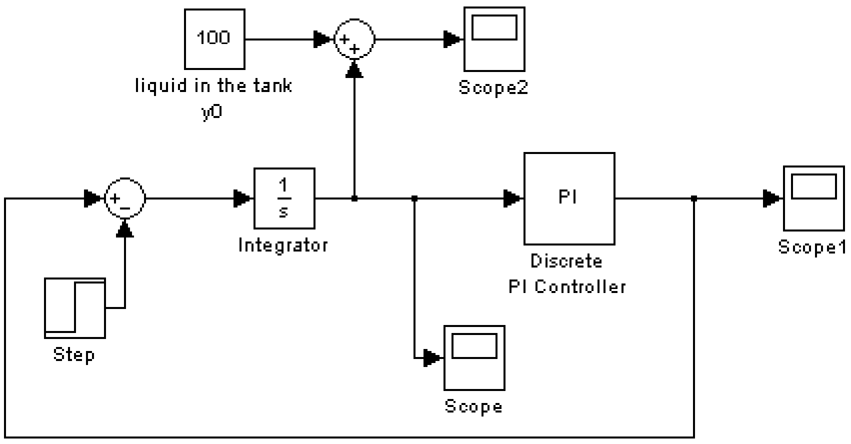


Figure 3.4 First PI Simulink model without observer

The modified system is as given below, after the discretization process (Tustin, $t_s = 0.01$) and the addition of the observer as a subsystem.

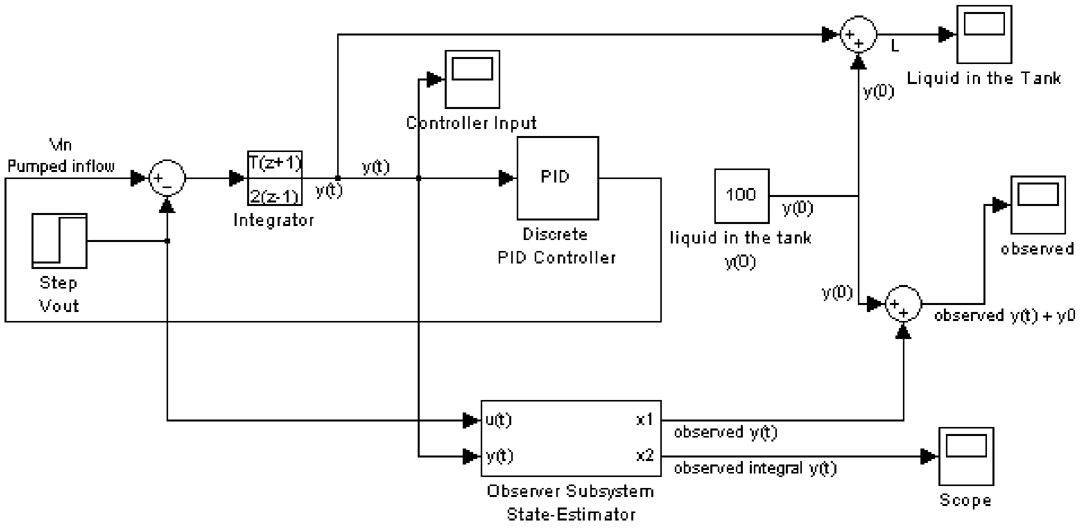


Figure 3.5 PI Simulink model with observer

The Controller block is changed to the TwinCAT function block of the PI Controller by loading the special TwinCAT control library, to be made compatible for conversion into CFC by Code Generator. The Observer subsystem is identical with the previous one.

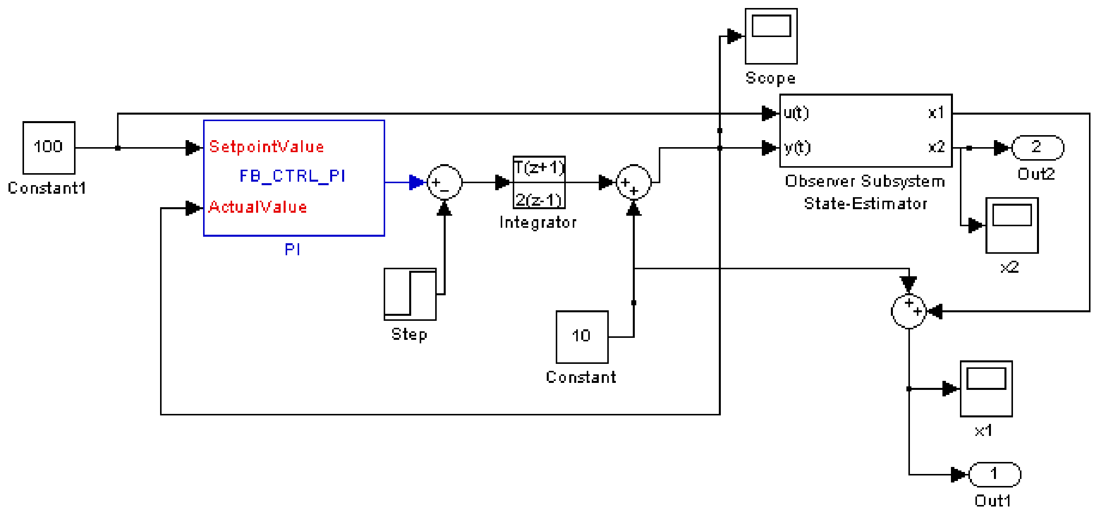


Figure 3.6 PI Simulink model with TwinCAT lib element and observer

3.4 The PD and P Controller

The PD controllers are mostly chosen where the controlled plant has an initial value and needs a fast ramp up to the set point from that inital value.

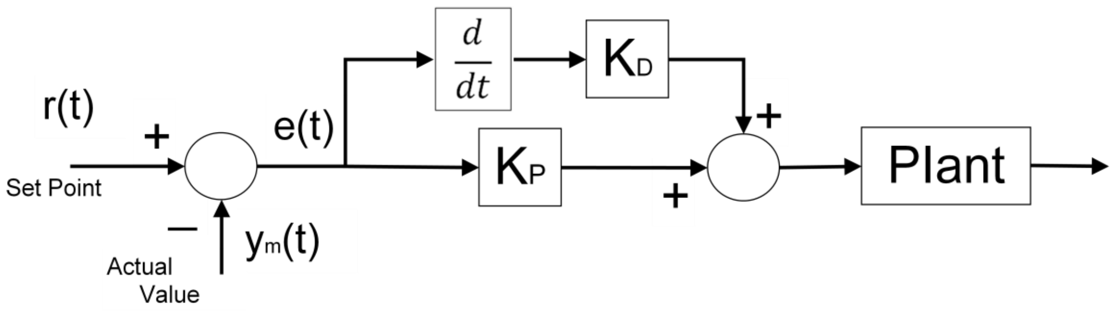


Figure 3.7 PD Controller Diagram

As will be given later in this section, for the level control application in this study, the outflow and the inflow will never concide; therefore PD is a better fit than PI controller because the PI controller is more dependent upon the outflow from the system.

$$G_c(s) = K_D.s + K_P \tag{3.25}$$

The mathematical equation of the desired output signal for the level control system is as given below.

$$y(t) = A.u(t) - [A.u(t) - y_0]e^{-\propto t} \tag{3.26}$$

When Laplace conversion is applied, the functions below are obtained.

$$Y(s) = \frac{A}{s} - \frac{A - y_o}{s + \propto} \tag{3.27}$$

$$Y(s) = \left[\frac{A}{s} - Y(s)\right].H(s).\frac{1}{s} \tag{3.28}$$

$$H(s) = \left[\frac{A \propto}{A - y_o} + \frac{y_o s}{A - y_o}\right] \tag{3.29}$$

Error is formulated as

$$\mathrm{E}(s) = \frac{A}{s} - Y(s) \tag{3.30}$$

$$Y(s) = \left[\frac{A}{s} - Y(s)\right].H(s).\frac{1}{s} \tag{3.31}$$

$$\frac{A}{s}.H(s) - Y(s).H(s) = s.Y(s) \tag{3.32}$$

$$\frac{A}{s}.H(s) - \left(\frac{A}{s} - \frac{A - y_o}{s + \propto}\right).H(s) = s.\frac{A}{s} - s.\frac{A - y_o}{s + \propto} \tag{3.33}$$

$$\frac{A}{s}.H(s) - \frac{A}{s}.H(s) + \left(\frac{A - y_o}{s + \propto}\right).H(s) = A - s.\frac{A - y_o}{s + \propto} \tag{3.34}$$

$$\frac{A.(s + \propto) - s.(A - y_o)}{s + \propto} = H(s).\frac{A - y_o}{s + \propto} \tag{3.35}$$

PD Controller’s transfer function is dericed as below.

$$H(s) = \frac{A \propto}{A - y_o} - \frac{y_o\, s}{A - y_o} \tag{3.36}$$

As it can be clearly seen from the transfer function that if the initial conditions are zero $(\mathrm{y_o} = 0)$, then the derivative term meaning the s variant would be null so the controller will become an only P controller. In the implementation it is formulated for $\mathrm{y_o} = 0$ condition.

$$H(s) = \frac{A \propto}{A} = \propto \tag{3.37}$$

The gain parameters for the PD controller are formulated as below.

$$K_P = \frac{A \propto}{A - y_0} \tag{3.38}$$

$$K_D = \frac{y_0}{A - y_0} \tag{3.39}$$

A transfer function for the controlled system is derived out of gain parameters and the PD controller transfer function.

$$[U(s) - Y(s)] \,.\, (K_P + K_D\, s\,).\frac{1}{s} = Y(s) \tag{3.40}$$

$$[U(s) - Y(s)].\, (K_P + K_D\, s\,) = s.\, Y(s) \tag{3.41}$$

$$U(s).\, K_P + U(s).\, K_D\, s - Y(s).\, K_P - Y(s).\, K_D \tag{3.42}$$

$$U(s).\, (K_P + K_D\, s) = Y(s).\, (K_P + K_D\, s + s) \tag{3.43}$$

$$\frac{Y(s)}{U(s)} = \frac{K_P + K_D\, s}{K_P + K_D\, s + s} \tag{3.44}$$

The derivative term will be ineffective under the terms assumed for the initial conditions as mentioned earlier. The further equations will be derived following this assumption.

3.4.1 The State-Space Notation of the P Controller

The system has a 1st degree transfer function. The state-space matrices are derived as below.

$$\dot{x} = -K_P x + K_P\, u$$

$$y = x \tag{3.45}$$

$$A = [-K_P], \qquad B = [\,K_P], \qquad C = 1$$

The state-observer implementation is given in Observers section.

3.4.2 Matlab/Simulink Model for Controllers

The Set Point for the PD controller is set to 100 for simulation purposes and the initial conditions (the liquid in the tank) is given with a constant block, again for simulation purposes set to 50. The set point is subtracted from the initial liquid height, resulting in the information of how much liquid the tank needs to get to the set point.

Subtracting this value from the system's output (the integrated flow-meter value), we obtain the error that is going to form the controller's input.

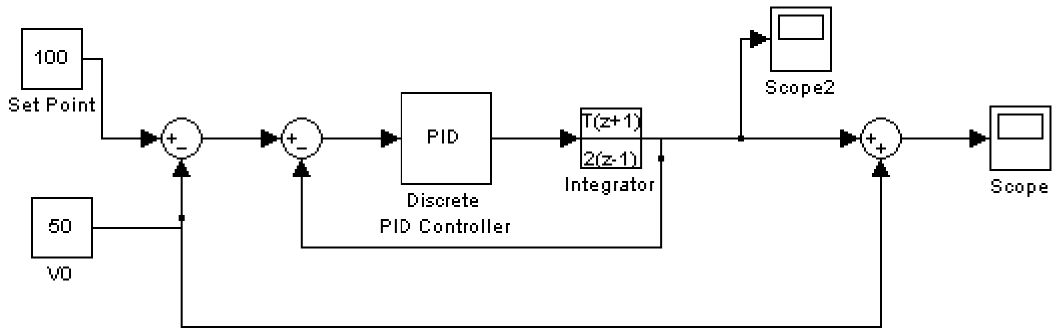


Figure 3.8 PD Controller Simulink model

The observer is implemented by using the system output, the error input. It estimates a flow-rate from these terms.

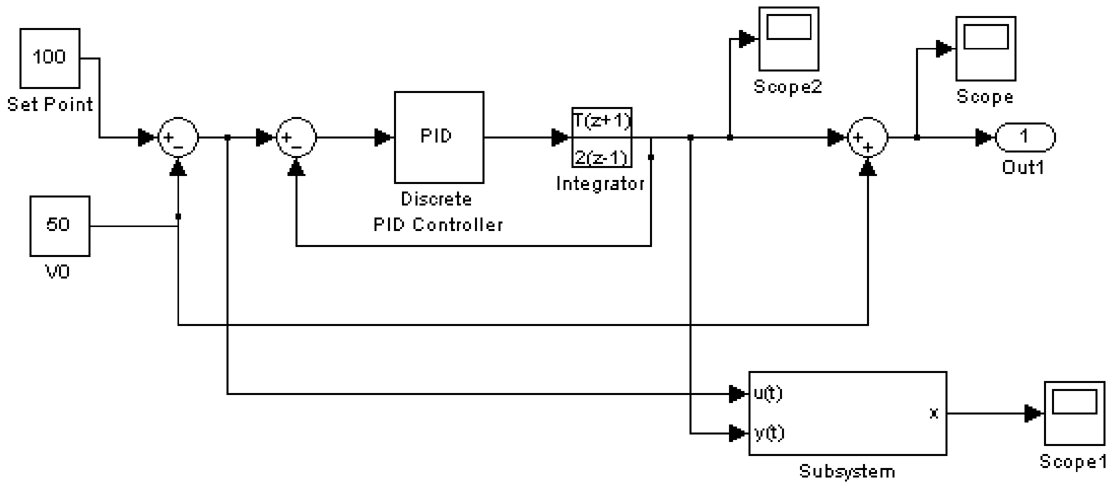


Figure 3.9 PD Controller Simulink model with the observer

The P controlled model of the system is modified using the Function Block of P control from TwinCAT library.

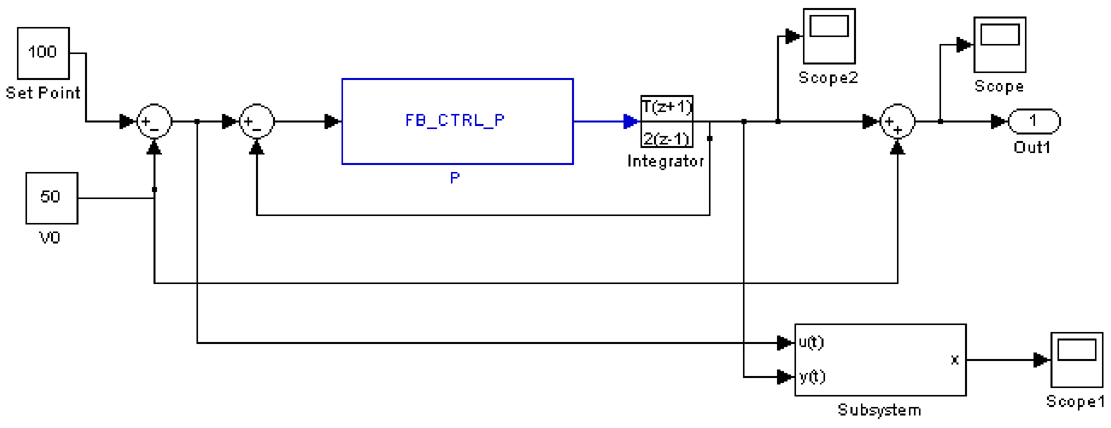


Figure 3.10 P Controller Simulink model with the observer

3.5 The State-Observers

As said in Observers in Control Systems [22] by Ellis, G. the observer technology is not a panacea but it provides a very good opportunity for the systems that are eligible to increase their reliability and dependability either by replacement of the signal or its enhancement. By eligibility it is pointed that not every system is observable, therefore it is not possible for some certain types of systems to implement and employ state-observers, such as the systems whose parameters at $t = 0$ are not possible to calculate.

The aim is to develop a model that relates the inputs with the measured outputs that are desired to be regulated. The relationship between the output and the input of the device that is desired to be controlled is formulated as differential equations. Then the derived differential equation is transformed into transfer functions of the observers using the Laplace Transformation.

The State-Observer selection is directly related to the system dynamics, whether the coefficients change over time or not. Since this is a linear continuous time system without any noise or interferences, chosen observer is linear, time invariant (LTI) State-Observers. The components of the studied system are designed close to being LTI, and therefore the non LTI behaviour could be avoided.

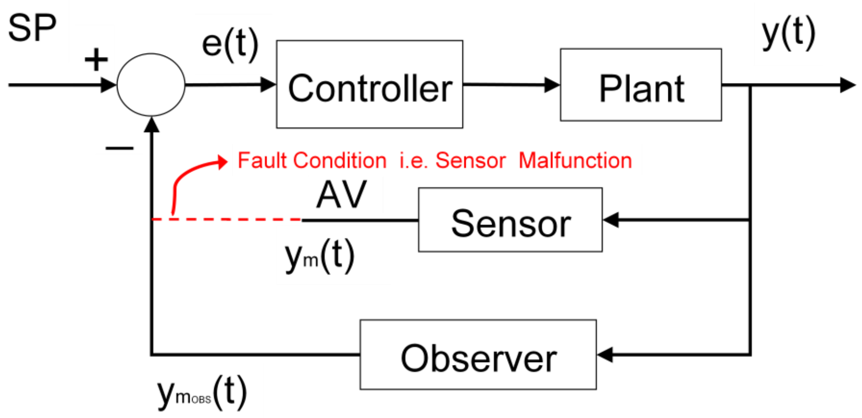


Figure 3.11 The State-Observer Model

For a generic system modelled here, the state vector (n-vector) is x, the control vector (r-vector) is u and the output vector (m-vector) is y with A, B, C, (in this case $D = 0$) matrices.

$$\dot{x} = Ax + Bu, \qquad y = Cx \tag{3.46}$$

The observer gain is symbolized with L.

$$\dot{\hat{x}} = A\,\hat{x} + Bu + L(y - C\hat{x}) \tag{3.47}$$

The observer error is the difference of x and the observed state of x.

$$e = \hat{x} - x \tag{3.48}$$

$$\dot{e} = (A - LC)e \tag{3.49}$$

3.5.1 State-Observer for the PI controlled system

The estimated states are designed to be $x_1 = y(t)$ and $x_2 = \int y(t).dt$ .Therefore the level in the tank and its integral value are estimated.

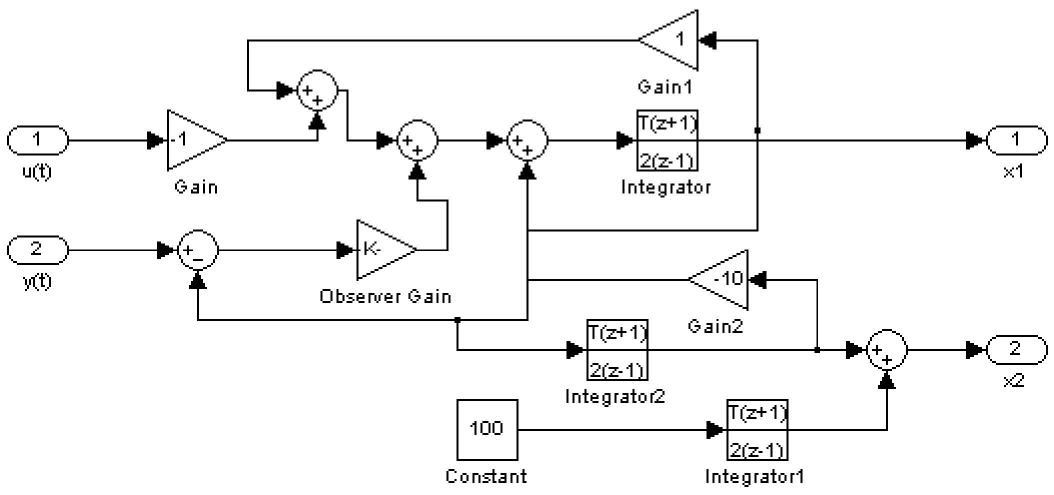


Figure 3.12 PI Simulink model observer subsystem

3.5.2 State-Observer for the P controlled system

The observer's gain (L) is chosen to be 100, the proportional gain is inherited as 2 from the main model. The output estimated the water level in the tank by observing the input and the outputs of the plants.

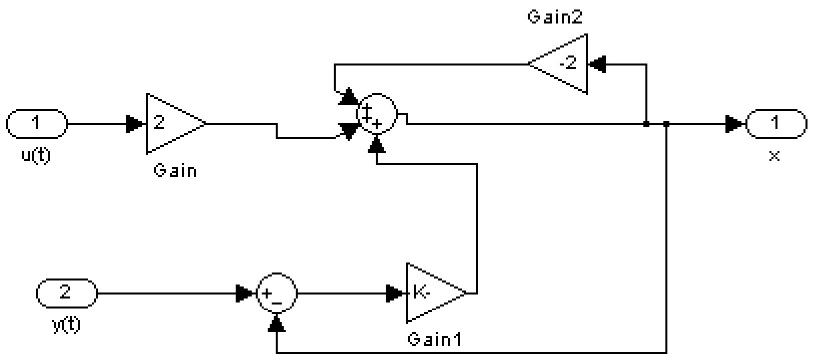


Figure 3.13 P Simulink model observer subsystem

3.6 The Operation of the Pump

The hydraulic pump in the plant operates at 24 Volts and uses 26 Watts of maximum power. The current and the flowrate of the pump are assumed to be proportional. By using the curve fitting tool in Matlab, the function is derived to be a linear polynomial.

x is the current (A) that feeds the pump as the input and the output y is the flowrate (l/min).

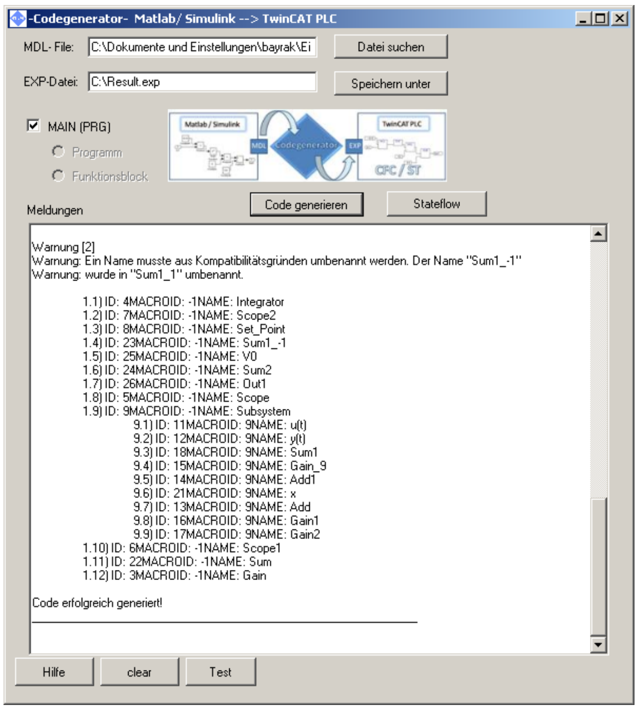
$$\mathrm{f(x)} = \mathrm{y} = 47{,}5\mathrm{x} - 26{,}5 \tag{3.50}$$

3.7 Code-Conversion Tool between Simulink and TwinCAT

The control system is modelled first in s-domain and then in continuous time domain but the model has to be in discrete-time for code generation process from Simulink to TwinCAT. Therefore a conversion will be made in Matlab model from continuous time domain to discrete time domain using the model discretizer tool with a sampling period of $T = 0.01\,sec$ before transferring the model to the conversion tool. The Tustin transform method is chosen in order to avoid problems about compatibility in code conversion.

Matlab/Simulink is chosen to implement the controller and estimator elements. Although Matlab/Simulink is an extensive program for simulation, the simulated and checked model has to be carried into the program environment in IEC-61131 language. The Simulink models would have to be built from zero to adjust into TwinCAT program if the Matlab/Simulink Code Generator [21] was not utilized.

Matlab/Simulink Code Generator enables code generation from a Simulink model into a Continuous Function Chart (CFC) under TwinCAT. It performs a 1:1 graphical transformation of the model into a CFC in IEC-61131 code using the similarity between Simulink models and CFC.



MDL- File: C:\Dokumente und Einstellungen\bayrak\Ei

EXP-Datei: C:\Result.exp

[✓] MAIN (PRG)

```
Warnung [2]
Warnung: Ein Name musste aus Kompatibilitätsgründen umbenannt werden. Der Name "Sum1_-1"
Warnung: wurde in "Sum1_1" umbenannt.

        1.1) ID: 4MACROID: -1NAME: Integrator
        1.2) ID: 7MACROID: -1NAME: Scope2
        1.3) ID: 8MACROID: -1NAME: Set_Point
        1.4) ID: 23MACROID: -1NAME: Sum1_-1
        1.5) ID: 25MACROID: -1NAME: V0
        1.6) ID: 24MACROID: -1NAME: Sum2
        1.7) ID: 26MACROID: -1NAME: Out1
        1.8) ID: 5MACROID: -1NAME: Scope
        1.9) ID: 9MACROID: -1NAME: Subsystem
                9.1) ID: 11MACROID: 9NAME: u(t)
                9.2) ID: 12MACROID: 9NAME: y(t)
                9.3) ID: 18MACROID: 9NAME: Sum1
                9.4) ID: 15MACROID: 9NAME: Gain_9
                9.5) ID: 14MACROID: 9NAME: Add1
                9.6) ID: 21MACROID: 9NAME: x
                9.7) ID: 13MACROID: 9NAME: Add
                9.8) ID: 16MACROID: 9NAME: Gain1
                9.9) ID: 17MACROID: 9NAME: Gain2
        1.10) ID: 6MACROID: -1NAME: Scope1
        1.11) ID: 22MACROID: -1NAME: Sum
        1.12) ID: 3MACROID: -1NAME: Gain

Code erfolgreich generiert!
```

Figure 3.14 A Screenshot image from the Matlab/Simulink Code Generator

3.8 The TwinCAT Implementation

TwinCAT PLC Software utilizes IEC-61131 Language as its backbone. It is has a broad selection of methods for programming such as using function blocks (FB), ladder diagrams (LD), continuous function chart (CFC) and so on.

CFC, also used here, enables connecting continuous time elements in a diagram and it is possible to create feedback loops as it is in Simulink, therefore it is an appropriate choice for real time implementation, such as the state-observer.

3.8.1 Continuous Function Charts

The CFCs are generated for the PI and P controlled systems, as given in the following. There are some tiny problems with the connections and below figures are obtained they are cut out

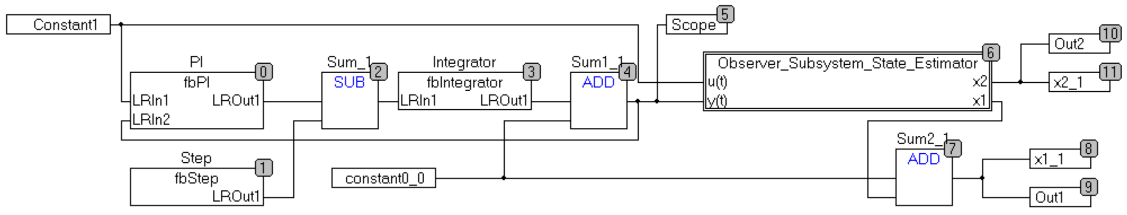


Figure 3.15 Diagram of PI Controlled System

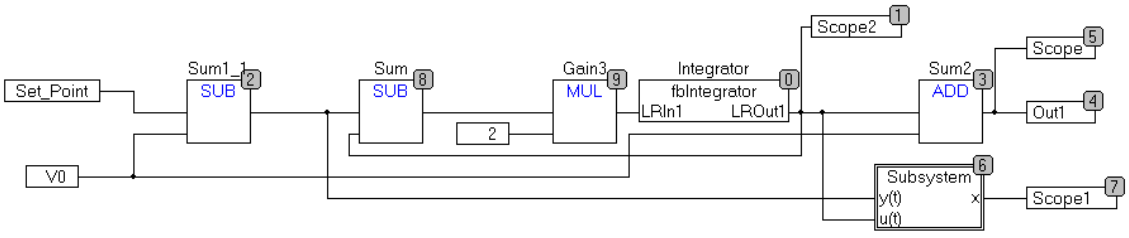


Figure 3.16 CFC Diagram of P Controlled System

The subsystems in each figure symbolize the observer blocks. The detailed view of the observers for P and PI systems are given in the following.

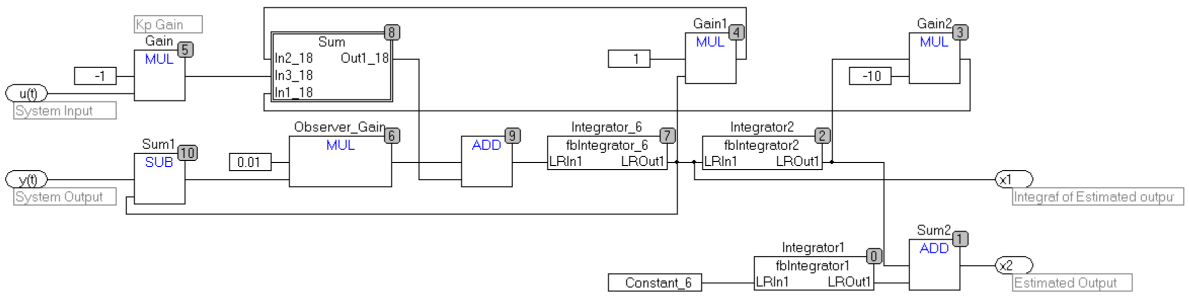


Figure 3.17 CFC Diagram of the PI Controlled System Observer

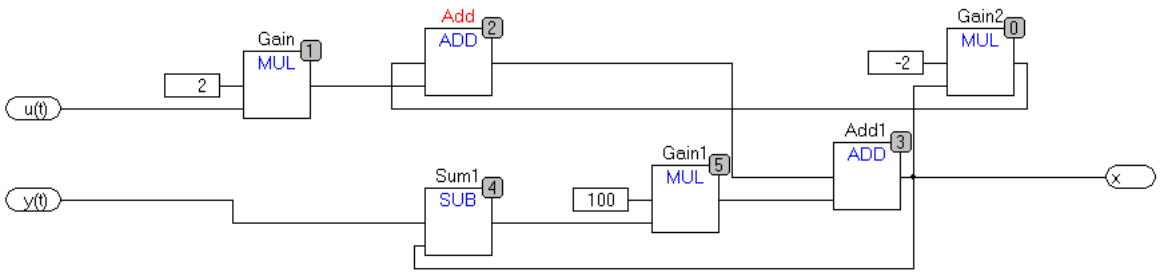


Figure 3.18 CFC Diagram of the P Controlled System Observer

3.8.2 The Functionality and Time Behaviour in TwinCAT

The aim for developing a CFC in TwinCAT is to implement the observer in the level control loop to be able to demonstrate the compensation of the observer through the system dynamics.

The conversion from Simulink to TwinCAT has automatically generated a CFC which is ready to implement with a few adjustments. One of those adjustments is the timing difference between the Simulink model response and TwinCAT CFC model response. Time behaviour is inherited by the generator from Simulink, however the problem lies in the feedback loops where there is backwards coupling. The reason for the difference between time behaviours lies in how Simulink operates. The intern functions of the Simulink solves this cyclic time behaviour issue however PLCs do not have such a mechanism therefore the converted code in this model needed extra Unit Delay blocks for each back coupled loop such that the first mathematical operation can acquire valid values.

CHAPTER 4

EVALUATION

The observer based fault management concept is simulated in the continuous process part of a hybrid automation plant to be evaluated.

However instead, the evaluation has been done on simulations by the plots from Matlab/Simulink and scopes taken from ScopeView TwinCAT interface, because the state-observer designed for this plant, does not operate as it was expected without the level sensor.

The evaluation is going to be done partially by the plots from Matlab/Simulink and scopes taken from TwinCAT interface. The implemented code is expected to control the plant which will operate by following the loop described in the flowchart.

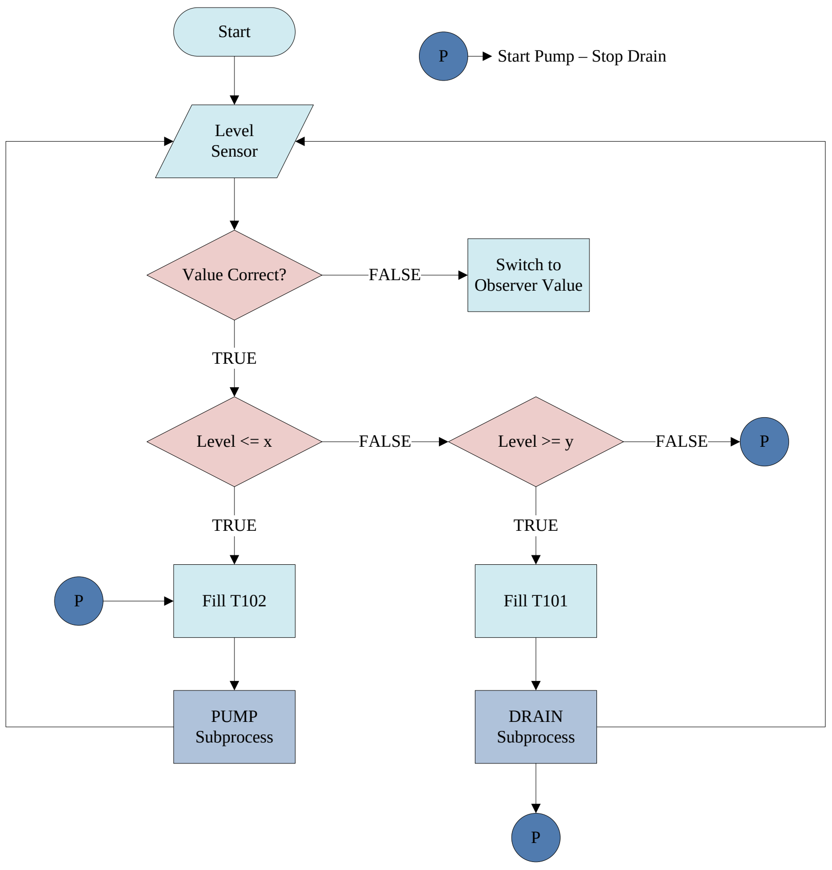


Figure 4.1 Flowchart of the desired conditions for operation

Levels x and y correspond to the water heights in tank101, the decisions are made on the level of tank 101 and then filling tank 102.

There are two actuators that are in charge of all the flow in this example, they are pump and ball valve and obviously their subprocesses are called “Pump Subprocess” and “Drain Subprocess” respectively. The essence is that the two actuators will not work simultaneously; once the pumping stops, the draining will start and vice versa.

The Simulink models and TwinCAT CFCs were previously presented in the Implementation chapter. In this section the Matlab/Simulink plots and scopes taken from TwinCAT Scope View.

4.1 Matlab/Simulink Plot Analysis

The observer output versus plant output will be plotted and given in scopes in this chapter.

The system is considered to be error-free therefore the results for the steady-state conditions were obtained %100 accurately.

4.1.1 PI Controller's Output and Observed Output

The output obtained from the model shows the drop of the level in $t = 1$ (when the draining valve is opened), and then the systems settles on the desired level of 100.

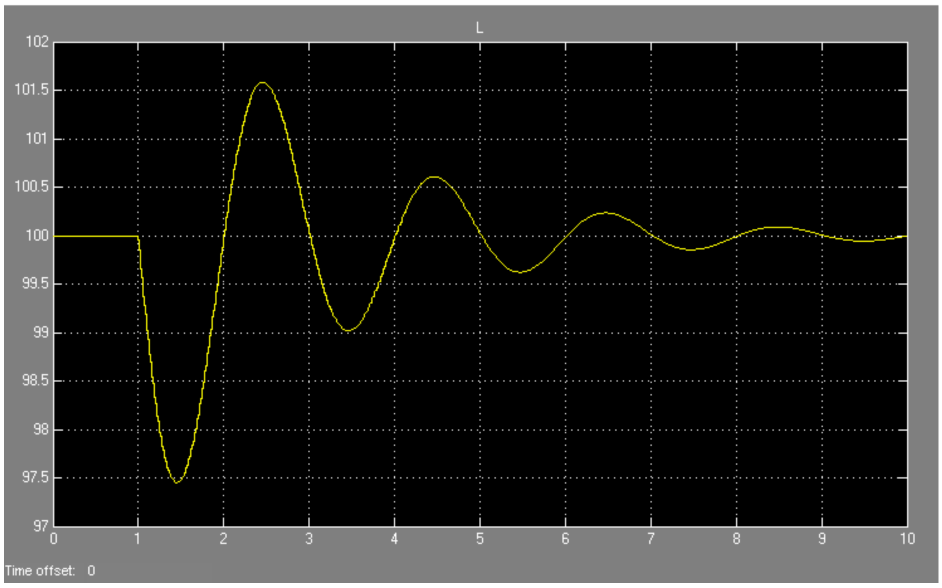


Figure 4.2 Modelled output

Observed output is plotted in the same way and it shows very similar results to normal output.

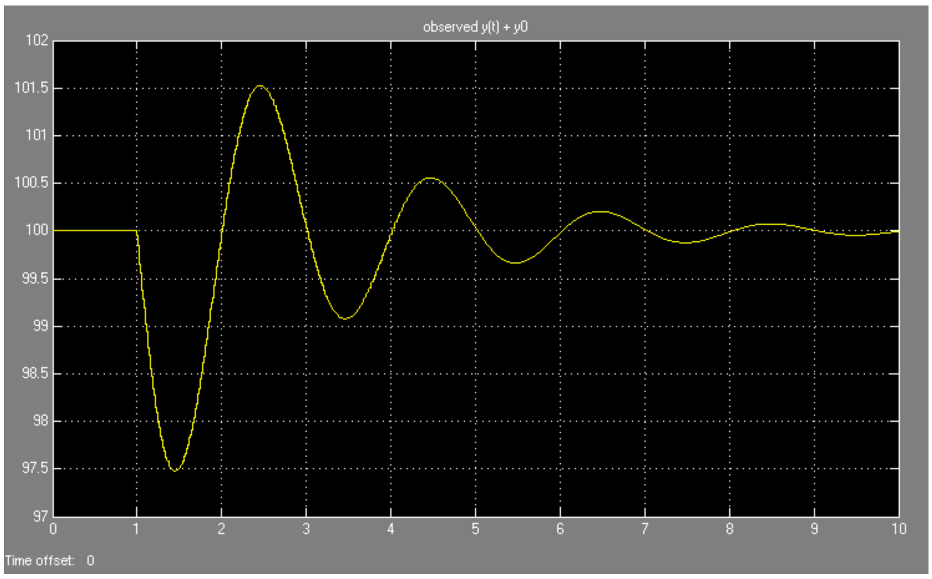


Figure 4.3 Observed output

The same simulations have been made using the same models on s domain. The results are significantly close to the previous results of normal and observed outputs.

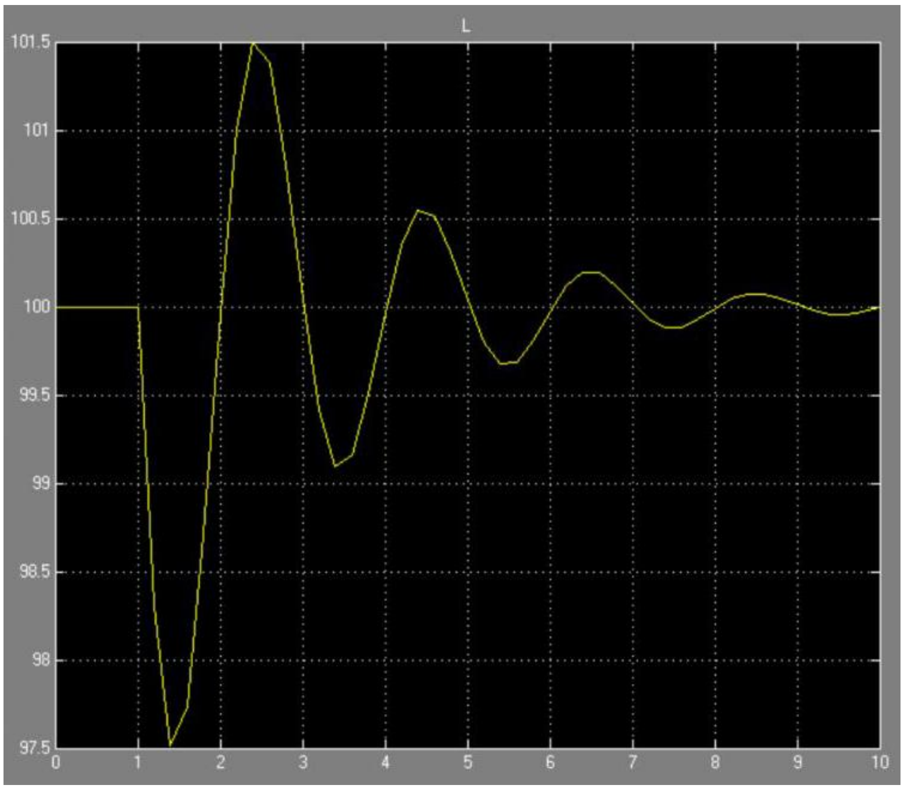


Figure 4.4 Modelled output

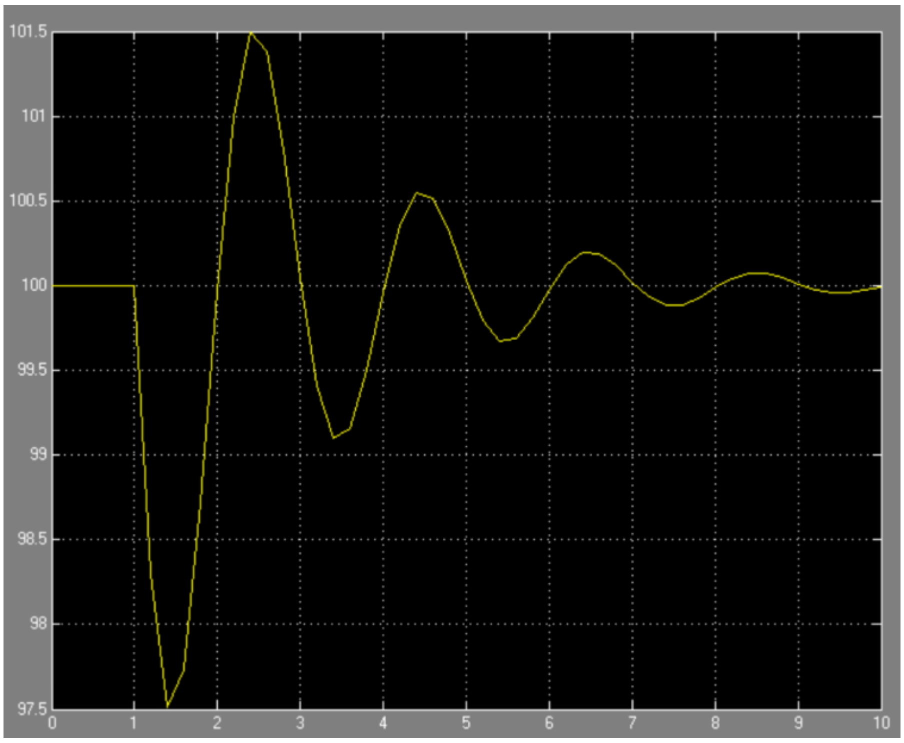


Figure 4.5 Observed output

4.1.2 P Controller's Output and Observed Output

The P controller regulates the level by using the error between the set point and the actual value. The initial water level is given to be 50 and the desired level stays 100. The pump supplies a level of 50 to reach the 100 as desired value.

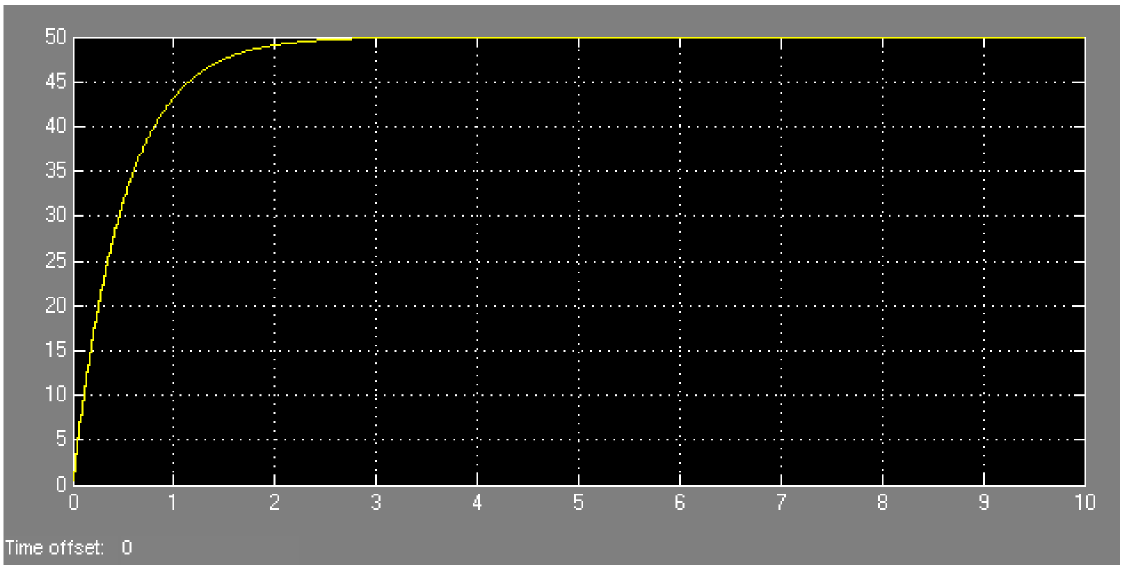


Figure 4.6 Modelled output

Observed output is plotted as below, showing again very similar results to above.

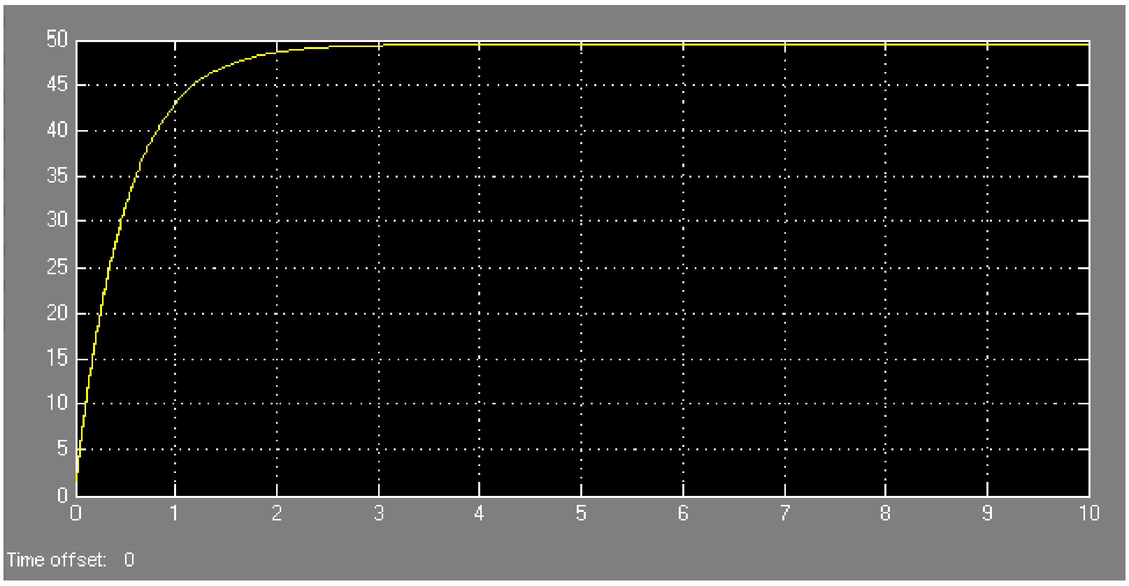


Figure 4.7 Observed Output

4.2 TwinCAT Plot Analysis and Model Validity

The plot analysis is only done for the P controlled system's observer. All the datafills are the same as it is in the Matlab/Simulink model: the level of the tank is given 50 and the desired level is set to 100. The plots show respectively the final level in the tank, the observed output and the output

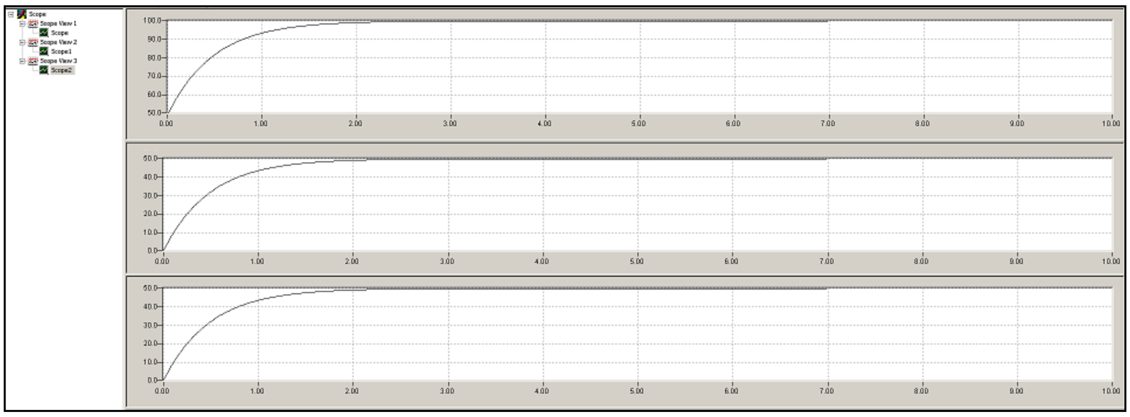


Figure 4.8 TwinCAT Scope View Output

Provided the fact that the codes were transferred from Simulink, the results demonstrate harmony with the Simulink scope plots.

The scopes from TwinCAT and the Simulink model plots have proven that the state-observer model can be applied to a continuous process plant and it can be used to obtain state variable estimates by using the input and output values of the system.

Although it is able to estimate the state variables, it does not translate to that it can estimate the output during its absence. In this implementation it was aimed for the observer to be able to estimate the sensor value as the sensor breaks down but the level sensor value in this plant model is the only sensor feedback value that supplies the output value for this system. Therefore it was not possible to accomplish such a compensation act with the state-observers in this implementation.

CHAPTER 5

DISCUSSION

The questions suggest the discussed topics through questions or arguments, followed right after them, are the answers and/or further derivation regarding the study.

Was the state-observer efficient in estimation?

In the context of this study which is conducted on a small scale level control process, it was clear that the observer data was not sufficient to replace the level sensor in the system by the observer itself. However, it can be possible to utilize it as a correcting factor for the controller input together with the sensor feedback. In this system virtual sensors can be implemented but it can’t be done only dependant on state-observers.

Why weren’t any estimation mechanisms designed for the system’s other loops?

Temperature control loop was based on on/off control therefore it is not a convenient process to implement an observer. It doesn’t have the particular state-variables for an estimation operation.

For the pressure control, it is possible to use the PI controller designed for the level control issue because they share the same principle. The pump control with PID family controllers is the essence of pressure control in this plant. Only by making the necessary adjustments for the manual valves and the

What types of systems are suitable for state-observers?

The state-observers are popular for their elimination of unwanted effects in the control loops. A major example to this type of problems is the phase lag, which can be removed by state-observers to stabilize the loop. There are different types of observers for specific purposes such as filter, predictor-corrector and so on. The type of implementation should be chosen regarding the plant's requirements; observers can be utilized to prevent the sensor signal from deteriorating by tuning the most suitable gain parameter for the system. This is an unwilling compromise of stability and response time. [22] In case the application requires faster responses, it is a perfect choice to use observer to rule out the phase lag from the loop.

It is important to take a closer look to the sensor in the system. Every sensor has a deviation between the measured and the real values, what needs to be evaluated is the type of the error that particular sensor has. It may be the case that the signal could have lag, it can have a cyclical error or a stochastic or non-deterministic error which is not predictable such as high-frequency noise.

CHAPTER 6

OUTLOOK

Due to the rising need on interfaces between different hierarchical programs, there is a certain demand for the conversion tools in system engineering field and directly in the automation area. It is a serious trend on a wide research and development ground. The success of the new generation conversion tools are directly in relation with the efficiency in system designs. It is also possible to foresee that as common as these interfaces are, as usual will be better reliability as a consequence of the ability to cross check between different hierarchical designs.

Within the scope of this thesis, the similarities between the SysML Parametrics diagram and Simulink model is evaluated.

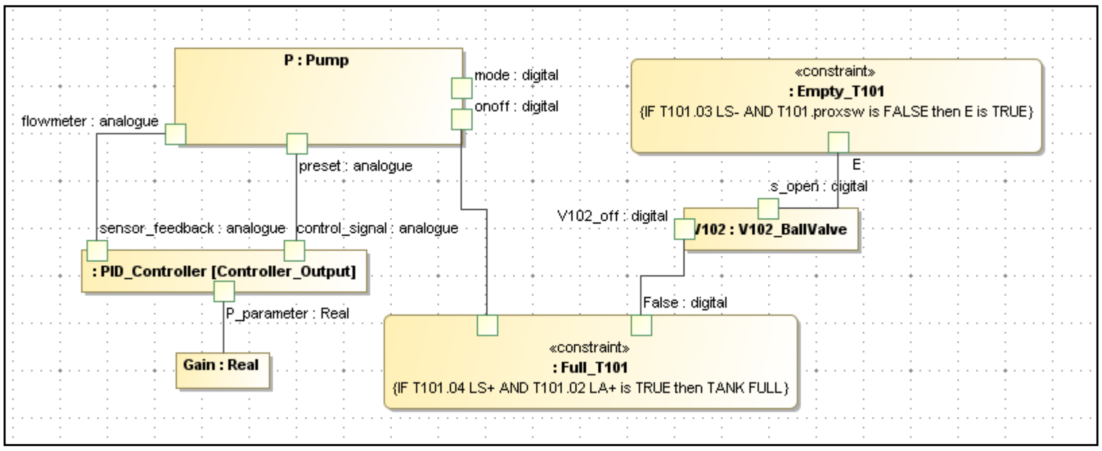


Figure 6.1 Parametrics diagram of Controller

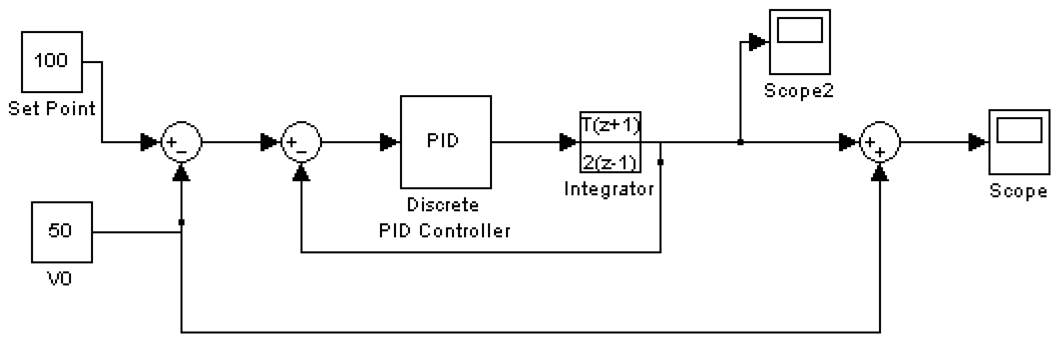


Figure 6.2 The P Controller of the System

When compared, the blocks appear similar at first sight, yet there exists major differences such as time behaviour simulation capabilities. Matlab/Simulink is capable of creating simulations and calculating time-responses of a system, however SysML is a very visual and intuitive modelling tool. It is very difficult to be able to relate these two but not impossible. For example, it should be considered to generate XML files from Matlab/Simulink in the exact format to be exported to define the SysML constraint blocks. In the same way it is possible to take it further and create an interface that is capable of generating a .m file from the SysML model of a system.

A SysML model is most helpful for laying out hardware diagrams and interconnections without any boundary of a strict notation obligation. However due to the limitations of time-behaviour investigation in SysML as in any other system modelling tools, Matlab/Simulink was chosen to implement and simulate the transfer functions and system parametrics in order to correctly analyse the system dynamics.

After the analysis of the system reactions and choosing the most appropriate alternative for the system, the design process has come to the final step where the observer implementation is to be realized on the PLC software which is TwinCAT in this case.

As much as it is tried to bring the three together at the same time, it has not been applied so far because these approaches don’t have a lot in common. Although they are not very distinct in sharing a lot of properties in groups of two, it is still difficult to build up a solitary environment for systems regardless of the fact that they are continuous, discrete or hybrid.

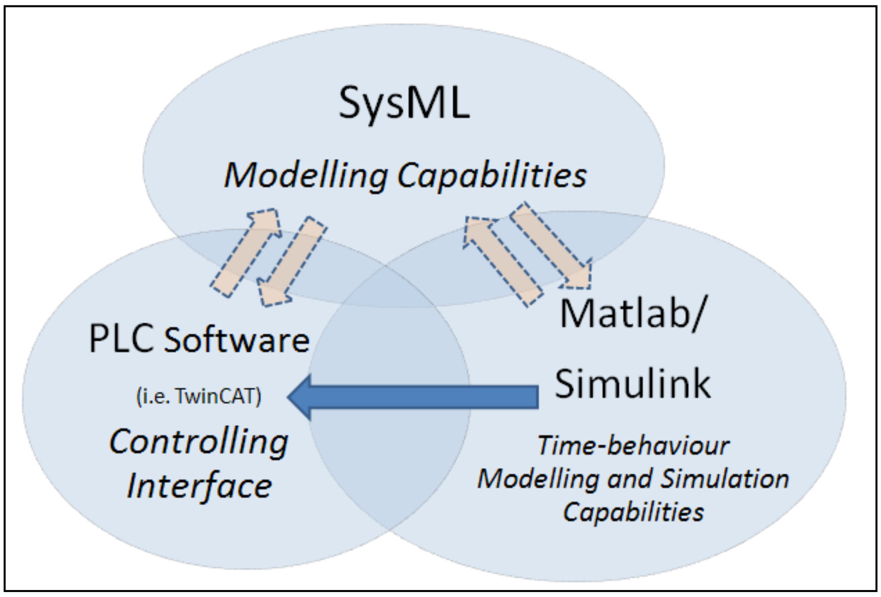


Figure 6.3 Model conversion possibilities

The diagram states that against many differences, they are used to serve the system therefore the same goal, they share many constraints and specifications. Future studies on combining different platforms are important because of the reasons given below.

In engineering, being able to make something once and to reuse it several times is a huge simplification of the design process and as a result a great deal of time saving. The main idea is the flexibility to utilize the design in multiple environments rather than re-defining it to fit into all these new interfaces or defining design independent details to match the interface specific requirements. It is logical that on a perceptive level, the technique behind a system engineer stay same to the eye however has to be told in different domains, different notations to obtain different responses.

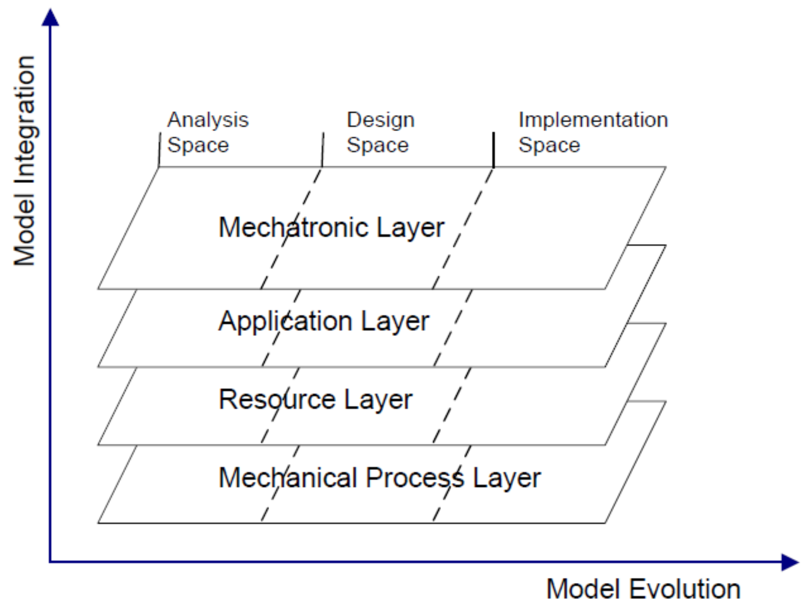


Figure 6.4 Model Integration vs. Evolution [18]

The graph represents the steps that system engineers are involved while designing a system. There are many layers i.e., mechanical, applications etc. and each one of them has their own spaces for analysis, design and implementation. Diversities for each section on a different layer is large but these layers are not abstract. As the model evolves, implementation matters most on each integration level. There can be a suited approach to combine the software environments, performing efficient enough to capture the needs of the designer all the way through.

For instance, in TwinCAT, it is possible to convert the PLC controlling code into other environments. As the circle is getting smaller between these interfaces given above, it is obvious that these languages that seem extremely apart do not have to be so abstract anymore. Even a slight increase in the compatibility would save many extra hours that is put to do the same engineering on different platforms. The more attention is drawn on the conversion tools, the more engineers will be provided the diversity they need in a single dish such that it will lower the costs and man/hour efforts put to the development projects.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

A SysML model is most helpful for laying out hardware diagrams and interconnections without any boundary of a strict notation obligation.

However due to the limitations of time-behaviour investigation in SysML as in any other system modelling tools, Matlab/Simulink was chosen to implement and simulate the transfer functions and system parametrics in order to correctly analyse the system dynamics. Further work on code generation and conversion between SysML, Matlab/Simulink and IEC-61131 languages is encouraged.

The state-observers were found inefficient to estimate system output in case either one of the input or output values are taken out of the system because the observer mechanism, by definition, is not designed for operation without $y(t)$ and/or $u(t)$ values as described in details before.

At steady-state, it was seen that the linear state-observers have performed with 100% percent accuracy in estimating the state variables while the error value converges to zero.

It is proven that it is not a feasible method for the particular systems that have less number of state variables to implement state-observers, as it was in the experimental plant of this thesis project.

It is only possible for the state-observer’s to estimate and, if implemented accordingly; to correct a certain distortion in the sensor data by two ways. First way is by ruling out the phase lag of the sensor signal during the controller is providing high gain values for the plant. As stated earlier, this is used in highly responsive applications, such as velocity control. The second way is to decouple the disturbances by observing them and simply subtracting from the plant’s response.

Therefore, a state-observer alone, is not a feasible method of solution for total sensor replacement because it cannot produce the feedback output that the sensor produces to keep the loop operating.

RESOURCES

[1] Dasarathy, B. V., (1997), "Sensor fusion potential exploitation-innovative architectures and illustrative applications", Proceedings of the IEEE, 85: 24-38.

[2] Everitt, B. S. and Skrondal, A., (2010),The Cambridge Dictionary of Statistics, Cambridge University Press.

[3] Coetze, E., (2008), "Security in Industrial Control Systems:Getting Back to the Basics", Mining, Manufacturing & Process Conference.

[4] Laprie, J.-C., et al., (1988), "WG 10.4 On Dependable Computing And Fault Tolerance", International Federation For Information Processing,

[5] Group, E. F., (2010), "Fundamental Limitations of Current Internet and the path to Future Internet", Fundamental Limitation of Current Internet and the path to Future Internet

[6] Buede, D. M. and Waltz, E. L., (2001),Data Fusion, John Wiley & Sons, Inc.

[7] Pierre David, V. I., Frederic Kratz, (2009), "Reliability study of complex physical systems using SysML ",

[8] Linhares, M. V., et al., Empirical Evaluation of SysML through the Modeling of an Industrial Automation Unit",

[9] Sriram Narasimhan, et al., (2002), "Hybrid Modeling and Diagnosis in the Real World: A Case Study",

[10] Gupta, Y. P. and Goyal, S., (1989), "Flexibility of manufacturing systems: Concepts and measurements", European Journal of Operational Research, 43: 119-135.

[11] Oppermann, R., (2005), "From User-adaptive to Context-adaptive Information Systems (Von benutzeradaptiven zu kontextadaptiven Informationssystemen)", i-com, 4: 4-14.

[12] Friedenthal, S., et al., (2008), "OMG Systems Modeling Language (OMG SysML™) Tutorial", INCOSE Systems Engineering for the Planet,

[13] Henderson, T. C., et al., (1998),Sensor Fusion.

[14] Peak, R. S., et al., (1998), "Integrating engineering design and analysis using a multi-representation approach", Engineering with Computers, 14: 93-114.

[15] Friedenthal, S., et al., (2009),A Practical Guide to SysML: The Systems Modeling Language, Elsevier Science & Technology.

[16] Weilkiens, T., (2007),Systems engineering with SysML/UML: modeling, analysis, design, Morgan Kaufmann OMG Press/Elsevier.

[17] Peak, R. (2010, Nov 9, 2010). A SysML Model-Based Approach for M&S VV&A.

[18] Thramboulidis, K. and Buda, A., (2010), "3+1 SysML view model for IEC61499 Function Block control systems" in Industrial Informatics (INDIN), 2010 8th IEEE International Conference on, 175-180.

[19] Object Management Group Copyright © 2003-2006, A. S. C., (2010),OMG Systems Modeling Language (OMG SysML™), OMG Document Number: formal/2010-06-01.

[20] "Process automation MPS® PA Compact Workstation Manual," F. D. GmbH, Ed., 08/2001 ed: Festo, 2001.

[21] Bayrak, G. and Vogel-Heuser, B., (2007), "Model-Based Development Concept for Hybrid Systems to Support Process Engineers in Thermo-Mechanical Process Development",

[22] Ellis, G., (2002),Observers in Control Systems - A Practical Guide, Elsevier.

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname	: Başak GENÇER
Date and Place of Birth	: 06.09.1985, İstanbul
Foreign Languages	: English, German
E-mail	: basakgencer@gmail.com

EDUCATION STATUS

Degree	**Major**	**School/University**	**Year of Graduation**
MSc.	Kontrol ve Otomasyon	YTÜ	
BSc.	Elektrik Mühendisliği	YTÜ	2008
High School	Fen-Matematik	Kadiköy Anadolu Lisesi	2004

WORK EXPERIENCE

Years	**Firm/Company**	**Occupation**
2008-2010	Nortel Netaş Networks	Global Product Support Engineer

Projects

1. Xplore 2008, Window Cleaner – İstanbul 2008

2. CoTeSys, Cognitive Cane – Munich 2010

AWARDS

1. Xplore 2008 (Germany) – Building Category 2nd