APPLICATION OF THE OBJECT ORIENTED MODELLING METHOD FOR MECHATRONIC SYSTEMS USING OPENMODELICA – CASE STUDIES

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

SÜMEYYE BETÜL COŞKUNOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF MECHATRONICS ENGINEERING

FEBRUARY 2016

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. K. İbrahim Akman

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Abdülkadir Erden

Head of Department

This is to certify that we have read the thesis Application of the Object-Oriented Modelling Method for Mechatronic Systems Using OpenModelica – Case Studies submitted by Sümeyye Betül Coşkunoğlu and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Abdülkadir Erden

Supervisor

Examining Committee Members

Prof. Dr. Abdülkadir Erden _____

Asst. Prof. Dr. Zühal Erden _____

Asst. Prof. Dr. Bülent İrfanoğlu _____

Asst. Prof. Dr. Yiğit Taşçıoğlu _____

Asst. Prof. Dr. Çiğdem Turhan _____

Date: 04.02.2016

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Sümeyye Betül Coşkunoğlu

Signature:

ABSTRACT

APPLICATION OF THE OBJECT-ORIENTED MODELLING METHOD FOR MECHATRONIC SYSTEMS USING OPENMODELCA – CASE STUDIES

Coşkunoğlu, Sümeyye Betül

M.S., Mechatronics Engineering Department

Supervisor: Prof. Dr. Abdülkadir Erden

February 2016, 92 pages

Simulation of mechatronics engineering systems is a challenging topic for researchers. Although there are many alternative approaches for simulation models, considering the complex, multidisciplinary and integrated structure of Mechatronic systems, system-level modelling and simulation using object-oriented methods would be useful to understand system requirements and system level integration. In this study, using the object-oriented modelling language Modelica, several case studies are performed using different Mechatronic systems. The mechanical, electrical and information subsystems are examined with respect to input/output and process structure, and then these subsystems are modelled for simulation using the OpenModelica program. Integration of subsystems is realized using OpenModelica, and the simulation of the main systems is performed. Based on these case studies, although OpenModelica platform have some limitations to implement on mechatronic system modelling and simulation, it is understood that Modelica language may be a convenient and useful tool as a modelling language.

Keywords: Mechatronic systems modelling, mechatronic systems simulation, Modelica language, evaluation of Modelica.

ÖZ

NESNE-YÖNELİMLİ MODELLEME YÖNTEMİNİN OPENMODELICA KULLANILARAK MEKATRONİK SİSTEMLER İÇİN UYGULANMASI – VAKA İNCELEMELERİ

Coşkunoğlu, Sümeyye Betül

Yüksek Lisans, Mekatronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Abdülkadir Erden

Şubat 2016, 92 sayfa

Mekatronik mühendislik sistemlerinin benzetimi araştırmacılar için zorlu bir konudur. Benzetim modelleri için birçok alternatif yaklaşım olmasına karşın, Mekatronik sistemlerin karmaşık, çok disiplinli ve tümleşik yapısı göz önüne alındığında, nesne yönelimli modelleme yöntemleri kullanılarak sistem düzeyinde modelleme ve benzetim yapılması sistem gereksinimlerini ve sistem düzeyinde entegrasyonu anlamak açısından faydalı olacaktır. Bu çalışmada, nesne yönelimli bir modelleme dili olan Modelica kullanılarak farklı mekatronik sistemler üzerinde çeşitli vaka incelemeleri yapılmıştır. Mekanik, elektronik ve bilişim altsistemleri öncelikle girdi/çıktı ve işlem yapısına göre incelenmiş ve daha sonra bu altsistemler OpenModelica programı kullanılarak benzetim için modellenmiştir. Alt sistemlerin tümleştirmesi OpenModelica kullanılarak gerçekleştirilmiş ve ana sistemlerin benzetimi yapılmıştır. Bu vaka incelemelerine dayanarak, OpenModelica platformunun mekatronik sistem modelleme ve benzetiminde kısıtlamalar barındırmasına karşın, Modelica dilinin, bir modelleme dili olarak kolaylık sağlayan ve yararlı bir araç olabileceği anlaşılmıştır.

Anahtar sözcükler: Mekatronik sistemlerin modellenmesi, mekatronik sistemlerin benzetimi, Modelica dili, Modelica değerlendirmesi.

To My Mother, Father, Brother and Sisters

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Prof. Dr. Abdülkadir Erden for his guidance and insight throughout the research. Thanks also go to Cahit Gürel and M. Hassan Golmohammadzadeh for their ungrudging share of their study. For her technical support, I appreciate to Meral Aday. For their continuous and precious support and patience, I offer thanks to my friends Yağmur Taylan, Gökçe Nur Aykaç and Handan Kara. Also, I express my sincere thanks to Burak Güney for his invaluable guidance at any moment I am in a quandary.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

LIST OF ABBREVIATIONS

ADAMS – Automatic Dynamic Analysis of Mechanical Systems

CTMS – Control Tutorials for MATLAB and Simulink

HRES – Hybrid Renewable Energy System

HVAC – Heating, Ventilating and Air Conditioning

MATLAB – Matrix Laboratory

POOSL – Parallel Object-Oriented specification Language

UML – Unified Modelling Language

CHAPTER 1

INTRODUCTION

As technology is developing, engineering products make people's life easier. More developed products require more complex structures in engineering. The complex problems make engineering designers' life harder. The complex system design is an important issue, as well as the analysis of the system. The designed systems should be analysed carefully to answer the questions about whether the system meets the needs, the problems about the structure, or whether the optimal design solution is found or not. These questions force the designer to find methods of analysing the systems accurately. An effective solution to answer questions about the system is *system simulation*.

Simulation is a process that enables the designer to find out the system structure. Using the simulation method, a system can be experimented on, without setting up the actual physical system. The required adjustments about the system can be made on the system model used in simulation. System can be questioned about the needs, the requirements, the inputs, outputs and the process. If any process goes wrong, the system structure can be changed and then again it can be experimented. The structure to allow this experimentation is named the system simulation model.

System modelling is a difficult task for designers. The model should cover as many parameters as it can include. An unconsidered parameter may change everything in a simulation and the simulation may result in incorrect deductions about the system. The results of the simulation can change and if an erroneous result is used while installing the real system, there may be huge failures within the system. This situation causes a waste of time and money, which is an unwelcome situation for the engineering area.

Modelling a system has been a research area for years. Improving science and technology allows designers to imagine more and more complex systems. There are some methods developed by researchers to model these systems.

To be able to simulate a system, first of all a simulation model should be established. To model a system, the system parameters should be understood very well. The parameters are covered inside the model to have the same characteristic of the real system. Therefore, the system model can have the same behaviour as the real system. It can be said that the system model carries a system to an abstract level, to find the necessary answers easier. If it is agreed that the model reflects the real system in a sufficient level, the system can be simulated.

On the other hand, it is a painful and problematic issue to model a complex system without losing the specialities. In many modelling methods, especially the physical topology of the real system is lost. Therefore, the characteristic of the real system cannot be sustained. This problem is tried to be solved by the developers using various methods.

1.1.Statement of Problem

In this thesis, modelling of mechatronic systems is realized by using Modelica language. This method is based on object-oriented modelling. This thesis proposes that Modelica language can be used to create a simple and entire model of mechatronic systems to be able to use in simulations in an easy way. Firstly Modelica language is investigated and then an appropriate simulation tool is chosen. After that, modelling is applied on chosen mechatronic systems, which are from Control Tutorial for MATLAB and Simulink (CTMS) website by Michigan University and the Atılım University Mechatronics Laboratory. Finally, the modelled systems are simulated and the results are provided.

1.2.Scope of the Thesis

In the scope of this thesis, the object-oriented modelling method is applied in two mechatronic systems. One of these systems is Inverted Pendulum System from CTMS. Second system is the Rose Harvesting Robot System which is developed by the researchers in the Atılım University Mechatronics Laboratory. In both cases, the real model is simplified before the simulation model is developed. The unnecessary parts are omitted while the method is applied on this model.

1.3.Outline of the Thesis

Outline of this thesis is indicated as follows.

- Chapter 2 introduces main concepts about modelling and simulation, and the researches about the topic. Also, the simulation tools which can be used for mechatronic systems are examined in this part.
- Chapter 3 presents the application of object-oriented modelling method on the chosen systems from CTMS, Inverted Pendulum System, and the Atılım University Mechatronics Laboratory, Rose Harvesting Robot System. This chapter also covers the implementation of the model using Modelica language on a chosen simulation tool and provides the simulation results.
- Chapter 4 represents the discussion on Modelica language by means of mechatronic systems in sight of the case studies and conclusion of the thesis and suggests future work.

CHAPTER 2

LITERATURE SURVEY

Engineering system modelling is a topic which researchers have been studying for a long time. The reason for this duration from past to present is the increasing complexity of engineering systems, and thus new methods for modelling are needed to be developed. In this chapter, the modelling of engineering systems and the place and importance of simulation modelling especially for mechatronic engineering systems will be explained in detail in sight of the literature. Also, the related work previously done about mechatronic system modelling will be given in detail. Finally, as a different point of view of the same topic, main simulation tools which are used by the researchers studying mechatronics engineering area will be listed with their explanations.

2.1. The concepts related to the modelling and simulation

The modelling of the engineering systems is a troublesome issue for designers. System modelling for simulation is difficult. Therefore, there are various definitions and concepts produced to explain the system structure and the need for the modelling and simulation. For the aim of clarifying the study in this thesis, the concepts of simulation, modelling and mechatronics and the relation between them will be provided in this part.

2.1.1. Simulation definition

Simulation is the imitation of the operation of the system over time. There is a main difference between modelling and simulation. Model is the imitation of the real *system itself*; however, simulation is the imitation of the *operation* of this system. That means, while there is one model of the real system, there may be several simulations on this model according to different scenarios.

According to Naylor (1966), “Simulation is a numerical technique for conducting experiments on a digital computer, which involves certain types of mathematical and logical models over extended period of real time.”

Singh (2009) defines system simulation as the technique of solving problems by the observation of the performance, over time, of a dynamic model of the system.

Simulation is generally done on computers. The model of the real system will be constructed in computer medium and the operation of the system is imitated.

Aydın (2013) states that “Simulation enables the production of artifical history of the system and observation of this history to deduce about the characteristic features of the real system.”

The aim of simulation is having the opportunity of observing the system’s behaviour under different conditions; and the model is built for this goal. Inside this model, a set of assumptions about system operation are included and these assumptions are expressed as the mathematical, logical and symbolic relations between the concerned attributes of the system.

2.1.2. Reasons for simulation

Simulation is done on the model of the real system. This model reflects the real system’s properties and functions. After building the model, the system can be experimented with “what if?” questions and answers are collected about the real system.

It is possible to use a system model as an analysis tool or a design tool (Aydın, 2013).

- An analysis tool: The effects of the changes on a current system can be observed using the model.
- A design tool: Prediction about the performance of a new system can be built by using the model.

In simulation, data collected from the real system (if exists) is used to predict the input parameters which are needed to run the system model.

According to Aydın (2013), some goals for using simulation are given as follows:

- **Assesment:** Demonstration of how the proposed system works well according to specified criteria,
- **Comparison:** Comparison of the proposed system design or policy,
- **Estimation:** Estimating the system's performance under the proposed conditions,
- **Sensitivity Analysis:** Determining what factors influence the performance of the system,
- **Optimization:** Determining the combination of factor levels giving the best performance value,
- **Bottleneck Analysis:** To determine the bottlenecks in a system.

Simulation is the process of designing a dynamic model of an actual dynamic system for the purpose either of understanding the behaviour of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system (Ingalls, 2011).

Simulation is the imitation of operation of a real-world process or system over time (Banks, 2001). According to the same author, simulation is an indispensible problem-solving methodology for the solution of many real-world problems. Simulation is used to describe and analyse the behaviour of a system, ask “what if” questions about the real system, and aid in the design of real systems.

The impact of design changes can be observed on a simulation model very quickly, helping to speed up the development cycles and to select an optimal solution as early as possible (Li et al., 2007).

2.1.3. Concepts underlying simulation

Banks (2001) and Carson (2004) clearly defined the concepts underlying simulation. These can be listed as following.

1. **Model:** Representation of an actual system, or representation of a system or process.

2. **State of model:** List of values that are sufficient to define the complete state of the system at any point in time.

3. **Event:** An instantaneous occurrence that changes the model's state.

4. **Primary events:** Events driven by data, occurs at a future time, calculated from data and statistical assumptions.

5. **Secondary events:** Generated events internally by model logic.

6. **Entity:** An object in the model.

7. **Resource:** An entity that provides a service to dynamic entities.

It can be clearly seen that the crucial parts of a simulation are *modelling* and the *flow inside the model*. System states are defined in modelling phase, and the system status is changed by the flow inside the system. The "flow" indicates the relation between two entities. First one of these entities is the *inputs* of the system, and the second important part is obviously the *outputs* of the system. The aim for doing simulation is analysing the relations between inputs and the outputs, as well as the behaviour of the system. Evaluation, comparison and analysis are the key reasons for doing simulation (Carson, 2004). Therefore, as the first step before simulation, the system model and the input and output entities should be clear and well-defined. Before the model is built up, the system structure should be understood deeply. This is the possible way to build up a realistic model of the real system.

2.1.4. Modelling definition

Model is a replica of the real system. Model of a system should reflect the real system, which means, model should have all the properties and functions of the real system. This can be said as an ideal model of the system. If the ideal model is hard to build, the model should include as many properties of the real system as it can have. The aim for modelling is to simulate the replica in a laboratory or in computer medium and find out about the real system.

A simulation model is a descriptive model of a process or system, and usually includes parameters that allow the model to be configurable, that is, to present a number of somewhat different system or process configurations. As a descriptive

model, you can use a simulation model to experiment with, and evaluate and compare, any number of system alternatives (Carson, 2004).

According to Singh (2009), models can be categorized in three groups:

Physical model: This type of model is the physical copy of the real system. The model is scaled down according to the actual system. Example for this category may be the model cars or model aeroplanes. The model can be run according to the scenario for the real system inside a controlled medium.

Mathematical model: This type of model shows the relationships between the attributes and the functions of the actual system as mathematical equations. After the equations are put, some methods are applied and the system can be solved analytically by the researchers. This model type is abstract.

Computer-based model: This model type consists of two parts. First, the physical model of the real system may be built in a virtual medium in an artificial manner. Second part is the mathematical modelling. The mathematical relations can be modelled using computers and numerical solutions can be obtained in a short time. This model type is both abstract and virtual.

These three types are also separated into two groups as static and dynamic.

Static model: This is the model type which does not change with time. Model cars or aeroplanes only reflect the static structure of the real system. In mathematical modelling, modelled systems in equilibrium are said to be static model. In other words, static mathematical models do not include time, t, inside the equations. In computer based models, again time-based models are dynamic where static models do not include the time variable.

Dynamic model: This type of model changes with time. Time variable is included in all dynamic models.

According to Fritzson (2010), the shortcomings of the experimental method lead us over to the model concept. If we make a model of system, this model can be investigated and may answer many questions regarding the real system if the model is realistic enough.

Models are crucial to support, communicate, and document the design activities. Existing modelling practices for mechatronic systems follow the same approaches. Model types and modelling tools used in design are mainly idiosyncratic[1] (Cabrera et al., 2008).

2.1.5. The philosophy of modelling

A study done by Li et al. (2007) mainly defines the modelling philosophy as follows:

From the point of view of system simulation, three core approaches can be distinguished.

1. Physical modelling (Network models): A structurally motivated partitioning of the overall system finally leads to a separation into physical elements (such as springs, dampers, masses etc. in mechanics; volumes, valves, throttles etc. in pneumatics or hydraulics and so on). This way of modelling preserves the physical relationships between the elements in the model structure.

2. Signal modelling: In signal modelling the behaviour of the system under study is implemented into a block structure. Signal modelling is the method of choice for mapping control systems.

3. Equation-based modelling: The function of subsystems and components in a system can also be described directly by differential equations. As for the signal modelling, the underlying physical topology is lost.

As it can be clearly seen, the three modelling approaches can be used in the most appropriate situation. However, these approaches need a different perspective when the mechatronic modelling and simulation is the point in question. In this thesis, this different perspective will be the object at issue. In the conceptual part of a system design, the inputs, outputs and the process flow inside the system will be examined in a different point of view considering mechatronics definitions and requirements. Therefore, the mechatronics needs and definitions will be explained in a short manner.

---

[1] idiosyncratic (adj): unusual and particular to a person or thing (Oxford Learner's Dictionaries, 2016).

2.1.6. Mechatronics definition

Mechatronic system is not just a marriage of electrical and mechanical systems and is more than just a control system; it is a complete integration of all of them (Bolton, 1999). Mechatronic system behaviour is determined by dependencies between different components. Therefore, an integrated and interdisciplinary engineering approach, which contains the various technical fields, is necessary (Scherber and Müller-Schloer, 1999).

According to Bishop (2007), the study of mechatronic systems can be divided into five areas of speciality:

1) Physical systems modelling
2) Sensors and actuators
3) Signals and systems
4) Computers and logic systems
5) Software and data acquisition.

2.1.7. Mechatronic modelling

Design of multi-domain engineering systems, such as mechatronic systems, differs from design of single-domain systems, such as electronic circuits, mechanisms, and fluid power systems, in part because of the need to integrate the several distinct domain characteristics in predicting system behaviour as part of the basic design (Goodman et al., n.d.)

Insight of this explanation, as a multi-domain engineering system, mechatronic systems can be said to be more complex than any other single-domain engineering systems; because of its nature of including different engineering domains at a time. At that point, because of this reason, mechatronic system modelling can be said to be more difficult than any other engineering domains. Also, simulation of mechatronic model carries more importance than the others.

According to Ingalls (2011), the more complex a system is, the longer it takes to model, run and evaluate. When it is considered that mechatronic systems have large complexity, the model will take a long time to build and simulate. Cabrera et al. (2008) claims that mechatronic systems tend to be inherently complex. Also

according to the same study, appropriate modelling and design support tools are essential to deal with system complexity. Therefore, the simulation tool must be strong enough to overcome the complexity problems.

Ghazi and Zarei (2007) say that "The essential characteristic of mechatronics engineering and the key to success in mechatronics is a balance between two sets of skills: modelling/analysing skill and experimentation/hardware implementation skills."

It is obvious that mechatronic modelling should include the modelling approaches that are previously explained. These approaches are physical modelling, signal modelling and equation-based modelling. Based on the definition and the properties of mechatronics discipline, mechatronic modelling has to cover all three approaches inside. The modelling of mechatronic systems requires physical and control modelling.

According to Cao et al. (2011), for mechatronic systems modelling, two specific requirements should be considered. First, the behaviour of the mechatronic system may be continuous, discrete or even hybrid. Therefore, the approach for modelling the three types of behaviour should be provided. The other is that it is difficult to determine if the system is designed correctly to meet different stakeholders' requirements based only on the designed static structure models and behaviour models.

2.1.8. Importance of mechatronic system simulation

Because of the integration of different domains, modelling mechatronic systems may take more time rather than the single-domain systems. If mechatronic systems are examined according to the previously explained reasons for doing simulation, the importance of simulation of mechatronic systems can be understood more clearly.

1. When the system is not appropriate for studying and experimented on: Most of the mechatronic systems are very complex and not appropriate for experimentation. Also, these systems include expensive parts and experimentation on real system may cause high costs.

2. If the system is still in the design phase: Mechatronic systems are expected to be accurate, low-cost and having good performance. Because of the system complexity, these design criteria can be tested on the simulation model in the design phase, to have the necessary information about the optimum system.

3. When the system will be analysed for behaviour: Mechatronic system behaviour may be analysed using simulation. System behaviour is complex due to the different domains and each domain changes the behaviour of the whole system. Therefore, the observation of whole system behaviour is provided by using simulation.

The reasons which Naylor (1966) stated can be considered specific to mechatronic systems as follows:

1. Mechatronic systems have complex internal interactions, between different domain components. Simulation makes it possible to study and experiment with the complex internal interactions of a mechatronic system.

2. Through simulation we can study the effect of certain changes on the operation of a system by making alterations on the model of the system and observing the effects of these alterations on the system's behaviour: Mechatronic systems include different domain components; that means it has much more complex behaviour. The changes on the system may cause huge effects on system behaviour and these effects can be observed on the simulation model rather than the real system. This opportunity gives the researchers the chance of changing the design and result in building more accurate systems having more stable behaviour.

3. Detailed observation of the system being simulated may lead to a better understanding of the system and to suggestion for improving it, suggestions that otherwise would not be apparent: Mechatronic systems should be understood clearly and the developments may be realised using the observations obtained from simulations.

4. Simulation can be used as a pedagogical device for teaching both students and practitioners basic skills in theoretical analysis, statistical analysis, and

decision-making: In mechatronics teaching, some simulations may be helpful to demonstrate system structure to the students.

5. Simulations of complex systems can yield valuable insight into which variables are more important than others in the system and how these variables interact: There are plenty of variables in mechatronic systems, and to be able to solve the relationship between these variables, simulation is an appropriate tool. The most important variables can be determined by experimenting on the simulation model of the real mechatronic system, and their relationships can be tested clearly.

6. Simulation can be used to experiment with new situations about which we have little or no information so as to prepare for what may happen: This opportunity leads the researchers to experience the mechatronic system entirely and gives ideas about the behaviour of the system in reply to unexpected situations. Also, several scenarios may be experimented on the system model, and best fitting situations can be determined using simulation results.

7. Simulation can serve as a "pre-service test" to try out new policies and decision rules for operating a system, before running the risk of experimenting of the real system: If it is considered that mechatronic systems are multi-domain, the risk of experimenting on the real system is higher. Therefore, simulation can be run for a pre service test and the experiments can be done on the simulation model.

8. When new components are introduced into a system, simulation can be used to help foresee bottlenecks and other problems that may arise in the operation of the system: This is an important point for mechatronic systems, because mechatronic systems include several different domain components which should be well-integrated to each other. Therefore, there may be incongruities between some of these components. Using simulation, the incongruent components can be determined and the problems can be seen clearly. Once problems are cleared, the solution can be formed and applied onto the real system.

2.2. Related work

Modelling of mechatronic systems for the aim of simulation is a challenging issue. There are plenty of works on this topic. Also, there are different methods for modelling mechatronic engineering systems. In this part, the main system modelling methods that are used on mechatronic systems in the literature will be explained in detail.

Firstly, the main methods will be mentioned in the literature. Valasek (2010) explains main modelling methods for multidisciplinary systems as follows:

1. Specialized tools for components and interfaces
2. Uniform modelling for uniform language
    i. Equations
    ii. Blocks
    iii. Multipoles
    iv. Bond graphs

The classification of the methods for modelling mechatronic systems can be summarized as explained below. In the literature survey, most of the studies can be classified into the explained headings.

If it is necessary to take a look at the literature insight of the multidisciplinary systems, mathematical modelling has been a preferred type of modelling for a long time. In general, mathematical modelling is used in block-oriented modelling. There are many studies in literature using mathematical modelling and blocks. Al-Sharif et al. (2014) used this approach for building simulation model used in Simulink on an elevator system as a mechatronic engineering modelling. Oh et al. (2009) used the same approach for controlling a model ship. In this point, it is necessary to indicate that, most of the studies using this type approach consider only the control modelling part of the systems. For example, Zhou et al. (2010) used the mathematical modelling and block diagram approach for modelling for Feed Servo Control System, based on torsion dynamics of lead-screw. Also, Drumea et al. (2008) developed a position control system based on a linear variable inductor displacement transducer using the same approach. Kalinski and Buchholz (2014) also used the

same method for the development of a three-wheeled mobile platform via hardware-in-the-loop simulation.

It can be said that the Bond Graph Methodology is known and has been used for mechatronic system modelling from past to present. Using this method, there have been studies in literature in the mechatronics area. To exemplify these studies, Malik and Khurshid (2007) state the Bond Graph Methodology as applicable on the physical systems due to its property of showing the topological and computational structure simultaneously. They used the modelling method on modelling the dynamic of a car crash and simulated the model on software. Kayani and Malik (2007) used the same methodology and genetic programming for automated design of mechatronic systems. Toufighi et al. (2007) modelled a mechatronic actuator system using Bond Graph Methodology. They designed an electro-hydraulic actuator and modelled the design using this methodology.

Coming to another main method, Unified Modelling Language, UML, is a strongly used approach for mechatronic system modelling and simulation. The structure of UML provides the usability for multidisciplinary systems. Also, some other methodologies developed based on the UML are used by designers. Tian and Wang (2010) used UML on a case study and modelled a robot arm. Also, they simulated the model and achieved successful results. The mostly used approaches developed on UML can be said as the SysML and Modelica. Abdul Rahman and Mizukawa states that SysML is a graphical language used to enable systems engineering activities (e.g. requirements analysis). It has been standardized based on UML (2013). They also implemented a SysML model on a mobile robotic system. Using this method, Mhenni et al. (2014) extended the SysML to achieve the detailed model to make safety analysis in mechatronic systems.

Modelica is another approach for researchers. Modelica is an object-oriented modelling language developed in past and came to present and has been popular recently. Elmqvist et al. (1999) states in Modelica the equations are used for modelling the physical phenomena. This property makes Modelica appropriate for mechatronic modelling without losing the physical topology of the real system. Regulin et al. (2013) used Modelica for modelling a dynamic valve-lift control in the automotive area. In this study, the object-oriented method is said to have lower fault

rates by early identification of failure by generation of physical models. The object oriented modelling language enables the build-up of hybrid models which represent physical and logical relations (Regulin et al., 2013). He et al. (2014) implemented a model of a single-zone Heating, Ventilating and Air Conditioning (HVAC) system in Modelica. He et al. states that, "We showed that equation-based object-oriented modeling allows analysing problems that are beyond the capabilities of traditional building simulation programs." Another study is done by Dizqah et al. (2013) who modelled a hybrid renewable energy system (HRES), a Standalone Wind-Solar Plant using Modelica on the platform of OpenModelica. Dizqah et al. state that, the developed mathematical model is declared by the Modelica language that makes it applicable for simulation.

Special tools for physical modelling of mechatronic systems are also used by researchers. Tatu and Alexandru (2012) studied the tracking mechanism for a photovoltaic system. The system is designed using SolidWorks for the solid model, ADAMS/View for the dynamic model, MATLAB/Simulink and ADAMS/Controls for the control system model. According to Tatu and Alexandru, the advantages of the method are reducing the testing time by minimizing the large number of physical prototypes and various measurements can be performed in the virtual environment for different measures (force, motion etc.) in any point of the system. Another research is done by Lesniewska et al. (2006) and restricted element study of the fracture healing by a mechatronic external fixation device was investigated using CATIA physical model and simulation results are obtained.

Some functional modelling approaches are also well worth mentioning. Cabrera et al. (2008) developed a project named Automatic Generation of Control Software for Mechatronic Systems. In this project, researchers introduce a proposal of a framework of prototype tools that aim to support controller design for mechatronic systems by providing control software generation. The researchers used Functional Behaviour State modelling approach for high level system modelling. In another study, van Beek and Tomiyama (2008) used Function-Behaviour State modelling approach to connect different design aspect views (e.g. functional view, workflow view, requirements view). Function-Behaviour State approach considers three independent concepts of function, behaviour and state. Also, the function modelling provides the connection between high-level requirements and low-level details (van

Beek, 2008). Another study that uses functional modelling is done by Zhang et al. (2003). Zhang et al. proposed an object-oriented development framework for mechatronic systems. In their study, the functional model is based on the flow of information, energy and material. According to their study, the realization of each functional component involves both hardware and software. The study satisfies the requirements for the systems.

There are some approaches that use integration of different methods. To exemplify these types of approaches, Akdağ et al. (2011) proposed an integration of mathematical modelling and physical modelling. The physical modelling is done on SolidWorks platform on a Hexapod robot. The integration gives the advantage of hybrid modelling. In another study, Schneider et al. (2012) used an integration of physical modelling and mathematical modelling in a reduced order, and a behavioural model was obtained. Scherber and Müller-Schloer (1999) proposed an integration platform for different type of modelling and simulation tools to be able to run mechatronic system models. A different integration study is done by Qamar et al. (2009). These researchers proposed to integrate SysML and MATLAB/Simulink for the usage of mechatronic systems. According to this study, the resulting integrated model is more comprehensive for the designer when investigating various design alternatives. They also implement their proposal onto an industrial robot named the semiconductor high speed precision laser pattern generator.

Some studies can be said to propose relatively new approaches for mechatronic modelling. Some examples can be given in this part. El-Manhawy and Hammad (2006) propose a new unified simulation approach for mobile robots control and motion planning. Proposed approach uses VHDL-AMS language and the researchers implemented their proposal onto Hexapod2 robot. Filipescu Jr et al. (2013) used discrete modelling and Petri nets to model a mechatronic system, a flexible teaching line for processing, sorting and storage. According to the researchers, the chosen modelling represents a good solution to accurately highlight the real process and to show different properties of the discrete event system. Miatliuk (2013) proposed a conceptual model for mechatronic design in the basis of hierarchical systems technology. Huang et al. (2007) developed a new design approach for mechatronic systems based on object-orientation. They used Parallel Object-Oriented Specification Language (POOSL). Saleem (2010) developed a component-based

modelling framework for mechatronic systems. Bin et al. (2010) developed a solution representation model based on design catalogue and implemented as a database model.

2.3. Simulation tools

There are plenty of simulation tools that can be used by developers. However, their aims of usage differ as well as their infrastructure. The major seven simulation tools used by the researchers are briefly explained in this chapter. These tools are also criticized on whether they are suitable for mechatronic systems modelling and simulation in a short manner.

2.3.1. ADAMS

ADAMS (Automatic Dynamic Analysis of Mechanical Systems) is a powerful tool for dynamic analysis on complex mechanical models. Main studies show that Adams is used in co-operation with other tools. Wen et al. (2009) used Adams to simulate the Manipulator of Welding Robot, used SolidWorks to create the mechanical model and built the control model in MATLAB/Simulink using block diagrams, and used the co-simulation method to bring the two models together. Generally, the mechanical part of the system is modelled in ADAMS and the dynamic solution can be made in this environment. However, coming to the control simulation, Adams is drawing an inadequate portrait. Although it has an effective coordination with other tools, when the requirements of a mechatronic system simulation are considered, Adams is not sufficient to simulate a mechatronic system; it needs another simulation tool.

2.3.2. COMSOL Multiphysics

Dickinson et al. (2013) expressed the main advantages and disadvantages of COMSOL Multiphysics clearly. COMSOL Multiphysics allows control of physical equations and numerical settings, while providing judicious automated defaults for meshing and solver configuration. There are also disadvantages of COMSOL Multiphysics. Like any discrete numerical method for solving a continuous PDE, the finite element method introduces some numerical error; to minimise this, it is necessary to use an appropriate mesh. However, the main disadvantage of COMSOL Multiphysics is the lack of control modelling. The tool is not suitable for control

design and simulation. Therefore, considering a mechatronic simulation, COMSOL is not a sufficient program, because the mechatronic systems require the simultaneous realization of physical and control modelling and simulation.

2.3.3. MATLAB/Simulink

MATLAB/Simulink is an equation-based/control tool, which is powerful for solving engineering problems. The main reason is the co-simulation power of Simulink. In addition, Simulink also allows the choice of the time step for the simulation and the choice of different types of ordinary differential equation-solvers (Spiryagin et al., 2012). George et al. (2011) used Simulink to implement an Aircraft Landing System. The energy functions used inside the block diagrams, and as it can be understood, they used the block diagrams to model their system.

However, it has a main disadvantage of not having the capability of modelling the whole system displaying the physical entities. MATLAB/Simulink uses mathematical modelling, but in mechatronic modelling and simulation, the physical system and the kinematic part carry a crucial importance. MATLAB/Simulink is a powerful tool for control modelling and simulation, but not for mechanical systems. Physical modelling is needed in mechatronic modelling and simulation, however, MATLAB/Simulink loses the physical topology of the designed system, therefore does not provide the engineers this type of modelling.

2.3.4. AMESim

Guangbu and Ruqiong (2009) modelled a belt conveyor dynamics model with a finite unit model and used AMESim to simulate their model. AMESim is designed to develop and analyse multi-disciplinary system models. Mainly AMESim provides the advantages of simple combining validated components from various libraries covering different physical domains, and providing rapid and accurate solutions, also a good interface with other simulation software. It can be said that the AMESim tool has the physical modelling and signal modelling approaches, as well as the equation-based modelling. Therefore, AMESim is suitable for mechatronic modelling and simulation. However, it should be considered that the performance is an important point and AMESim may need the help of another tool to model and simulate the desired system.

2.3.5. Modelica-based tools

Modelica is a highly flexible, open-source language to meet the needs for modelling and simulation. Thanks to open-interface structure of Modelica, users can develop their own libraries. Another advantage of Modelica is the "reusability" that comes with the object-oriented feature and *acausal* modelling. Acausal modelling means that relationships between equations are not based on cause and effect, and brings direct usage of equations instead of assignment statements. The structure of the model naturally corresponds to the structure of the physical system in contrast to block-oriented modelling tools such as Simulink (Fritzson, 2011).

2.3.5.1. SimulationX

SimulationX is software for valuation of interaction of all components of technical systems. Wei et al. (2011) took the advantage of SimulationX software platform. Based on physical modelling, they built a vehicle powertrain model. The vehicle transmission network-modelling method is based on the physical structures. The model is composed of the connections and components of the powertrain.

It is obvious that the whole system can be modelled together in SimulationX. The mechanical part and the control part can be implemented and realized at the same time; also the simulation of the two is done simultaneously. As it is mentioned before, this is the core of mechatronic simulation. SimulationX tool uses all three approaches of physical modelling, signal modelling and mathematical modelling together. That is the reason for SimulationX is an appropriate tool for mechatronic modelling and simulation providing all the easiness.

2.3.5.2. Dymola

Dymola is a component based modelling and simulation tool that has unique multi-domain modelling capabilities (Dempsey, 2006). Dymola has a wide-range library preference. In Dymola, the physical network can be expressed in the model, and also the control model can be built. Also, according to Dempsey (2006), it is possible to define large, complex systems and re-use them in different simulation problems without having to change the system model itself.

Simic et al. (2007) used Dymola to model a cooler system for an internal-combustion engine. They used object-oriented modelling to implement the physical parts with the mathematical equations. According to Simic et al. (2007), the Modelica language allows the implementation of ordinary and algebraic differential equations of multi-domain physical systems. This point also makes Dymola a really strong tool for mechatronic simulation, as well as involving the physical and control modelling inside.

2.3.5.3. OpenModelica

OpenModelica is currently the major Modelica open source environment. Based on Modelica language, OpenModelica can perform the modelling and simulation of dynamic systems. As it is mentioned in Dymola part, OpenModelica has the major advantages of Modelica. OpenModelica is a powerful tool to perform mechatronic system simulation, because of the flexible structure of Modelica and its own dynamic environment that supports physical modelling and control modelling at the same time. Szolik and Zakova (2012) used OpenModelica to provide the remote control of a thermo-optical plant. They used object-oriented modelling; however, the explained system model is closer to information techniques, therefore includes information flow.

2.4. Tool selection

Tool selection will be done considering the previously explained tools. These tools can be grouped into three, according to their platforms. These groups are named as multibody dynamics tools, block-oriented tools and object-oriented tools. All these have advantages and disadvantages. First, these groups will be explained in detail. After that, the most appropriate group will be chosen as the infrastructure. Then, the tools included by this group will be given shortly. The suitable simulation tool will be chosen for the implementation of the system which has been studied and explained previously.

2.4.1. Multibody dynamics tools

These are mainly based on the mechanical system modelling, and therefore the simulation is done on the physical model. Generally these tools are used for co-simulation with another tool, because of the lack of the control modelling and

simulation. On this type of tools, energy flow can be seen easily. Especially the mechanical energy flow can be seen clearly because of the structure of the model. The physical model is a mechanical system model. However, this is only one part of a mechatronic system.

Mechatronic system includes also the electronic and the information parts. Therefore, the flow of information should be shown on the same model. The mechanical model is insufficient to display the flow of these entities. Therefore, these entities are generally shown inside another simulation tool and after that the results (that means, the output information of the other tool) is given inside the mechanical system model as an input. This lack of information processing and information flow is a handicap for multibody dynamics tools, in the point of mechatronic view.

2.4.2. Block-oriented tools

Blocks are used in modelling of the real system. Blocks can be used to define the entities for information, energy or material, because of the reason the mathematical relations are embedded inside the blocks. However, the block diagram loses the physical topology. The relations between the parts of the system and subsystems cannot be shown clearly; in addition the physical structure is lost. These parts are defined only in an abstract level mathematically. Using block diagrams, the differences and/or similarities between entities cannot be displayed. This is the main disadvantage of the block-oriented tools. Also, the block diagram only shows the mathematical relations that include equations. However, it cannot be seen which kind of input flows into the system and which kind of output is observed. Also, the conversion step cannot be defined clearly. This is an example of physical losses.

2.4.3. Object-oriented tools

This type of tools is the most suitable one for mechatronic modelling and simulation. This is the result of the clear definition of the parts. Whole parts of a mechatronic system can be easily defined using an object-oriented tool. The inputs and outputs can be created independently or relatively. Also, it is not an obligation to define input entities and/or output entities explicitly. An object may include both input and output entities. This type of tools provides the chance of defining all of the parts that a mechatronic system includes. The physical domain can be described as well as the

information domain. Therefore, the information flow and the physical flow can be shown in required relations, using the important parameters, and this situation gives the designer to follow the conversion, consumption and the flow of the information, material and energy. The mechanical part and the control part can be implemented and realized at the same time; also the simulation of the two is done simultaneously.

According to the structure of the systems that is implemented by means of mechatronics, one of these three approaches is chosen as most suitable for mechatronic systems. The most appropriate type of simulation tool for mechatronic applications can be said to be the object-oriented tools. The reason for that is the entities can be implemented easily inside the object-oriented tools. These entities are not obliged to be in the same domain, different domains can be implemented in the same model using object-oriented tools. The relations between different-domain objects can be shown quite easily compared to the block-oriented tools and multibody dynamics tools.

The SimulationX, Dymola and OpenModelica tools are Modelica-based object-oriented tools. They all have advantages and disadvantages. All three tools provide a high opportunity for the implementation of complex systems due to their object-oriented structures that use Modelica modelling language. However, the OpenModelica tool is chosen due to its open source environment. This feature of OpenModelica provides free access to the tool and it is easy to get this tool. Therefore, the implementation of The Rose Harvesting Robot will be performed in OpenModelica tool for this reason.

In OpenModelica, first the objects are implemented. After defining the objects, the system process is attempted to be implemented. The outputs are observed from the system model implementation. The relations between the objects are established and the system simulation is executed. In this chapter, the study on Rose Harvesting Robot using OpenModelica will be explained in detail.

2.5. Conclusion

Simulation is an important tool for testing the systems not on the real system but on the replica of it. The experimentation can be realised on the system model and the results may lead the researchers to more accurate design solutions. Mechatronic

systems are especially complex engineering systems; therefore experimenting on the real system is risky and costly, in addition to the behaviour of the system may not be clear. Also, system structure cannot be clarified easily due to the multi-domain components, and the appropriate components should be chosen for mechatronic systems to provide the accuracy and system performance. These criteria can be measured and clarified by using simulation method. In conclusion, simulation is a significant way to find out, observe and develop a mechatronic system.

CHAPTER 3

IMPLEMENTATION OF CASE STUDIES ON OPENMODELICA

In the scope of this thesis, two chosen mechatronic systems are implemented on OpenModelica tool. These two systems are respectively Inverted Pendulum System and the Rose Harvesting Robot System. These systems will be explained in detail in this chapter. The model constructions will be provided. After that, the implementation on the chosen tool, OpenModelica, will be realised and the obtained results will be provided.

3.1. Inverted Pendulum

Inverted pendulum is a simple mechanism based on a pendulum that is upturned with respect to the x axis, and positioned onto a wheeled cart to be able to control the system. The pendulum must preserve its position constant.

3.1.1. Simple Inverted Pendulum

Inside OpenModelica, using the “writable” modelling, a simple inverted pendulum is modelled and simulated. A simple inverted pendulum can be seen in Figure 3.1.

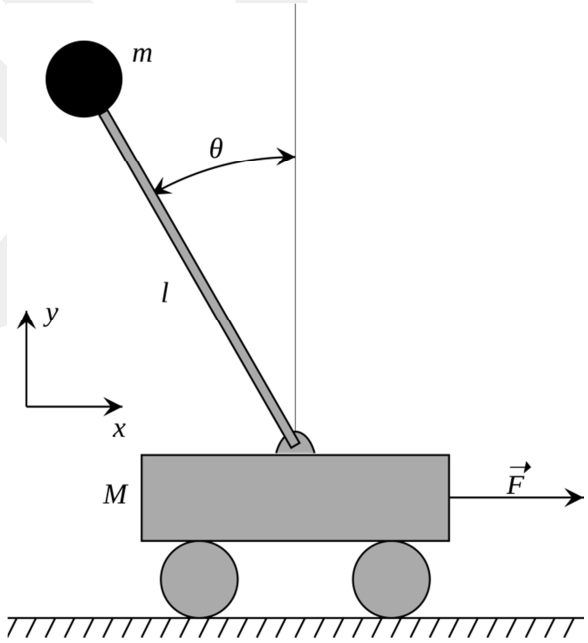


Figure 3.1. Simple inverted pendulum system

The simple inverted pendulum equations of motion are given below (Coller, 2015).

$$mg\sin\theta = m\ddot{x}\cos\theta - ml\ddot{\theta}$$

$$F + ml\ddot{\theta} - ml\dot{\theta}^2\sin\theta = (m+M)\ddot{x}$$

Here,

F: Force acting on the cart

M: Mass of the cart

m: Mass of the pendulum

x: Displacement of the cart

Θ: Angle of the pendulum

l: Length of the pendulum link

g: Gravitational acceleration

The model is set up using OpenModelica "writable" platform. Model listing can be shown in Figure 3.2.

```
model invertedPendul
  constant Real PI = 22 / 7;
  parameter Real m = 0.2, M = 0.5, L = 0.3, g = 9.81;
  Real F;
  Real ref = 0;
  output Real x(start = 0), theta(start = 0.1 * PI /
180);
  output Real vx, vang;
equation
  m * g * sin(theta) = m * der(vx) * cos(theta) - m *
L * der(vang);
  F + m * L * der(vang) - m * L * vang ^ 2 *
sin(theta) = (m + M) * der(vx);
  der(theta) = vang;
  der(x) = vx;
  F = 0.5 * (ref - vang);
end invertedPendul;

```

Figure 3.2. Simple Pendulum model on writable platform of OpenModelica

In this model, a proportional controller is used for the control of the pendulum angle. The signal is given to the system according to the reference angle. Reference angle is given as 0, thinking of the pendulum angle Θ to be 0.

The results for the simulation of this model can be seen in Figure 3.3, Figure 3.4 and Figure 3.5.

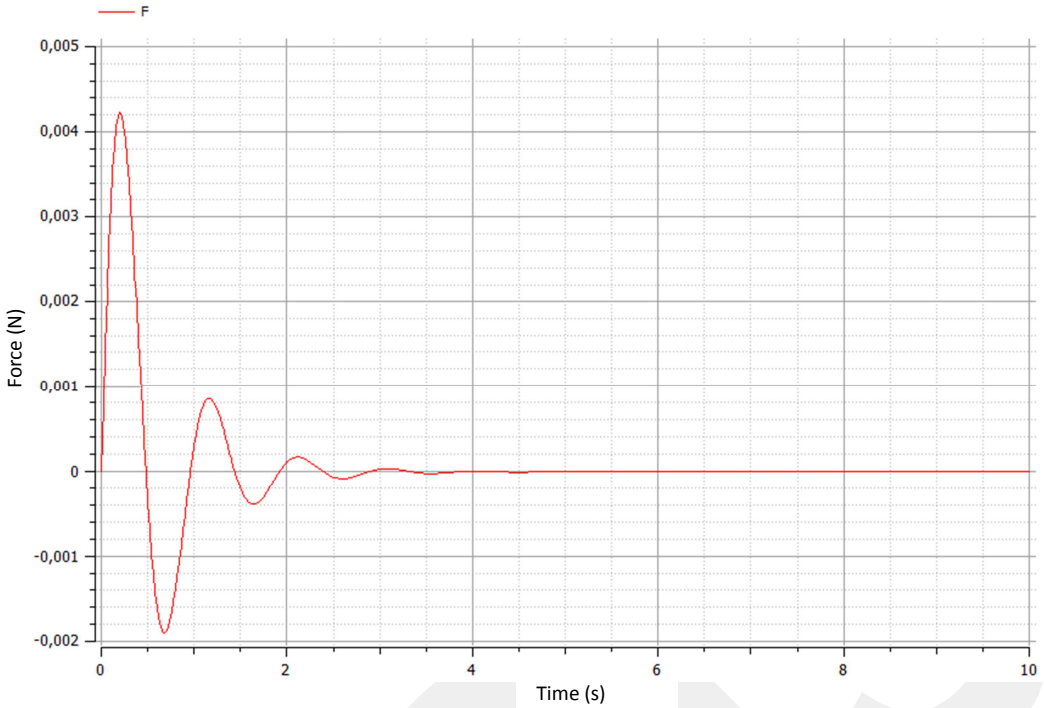


Figure 3.3. The input force given to the system

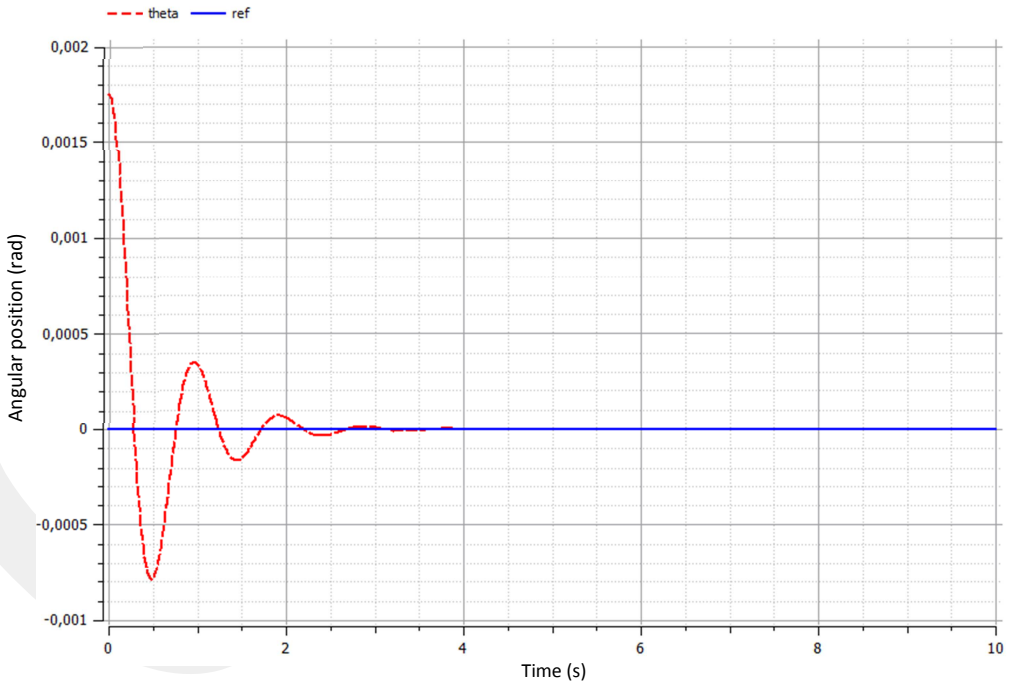


Figure 3.4. Reference angle and actual theta

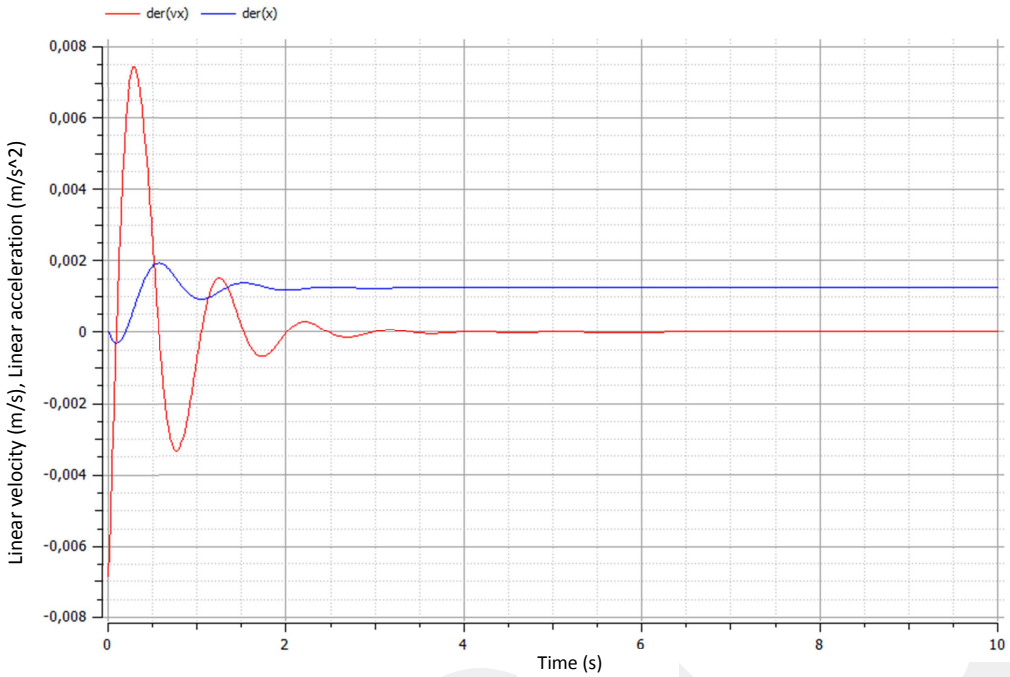


Figure 3.5. Linear speed and acceleration of pendulum

3.1.2. Inverted pendulum with inertia of moment

Another reference for equations of motion for Inverted Pendulum System is the Control Laboratories webpage. These equations include inertia of moment, which means reflect the real physical system better. The system and the free body diagrams can be seen in Figure 3.6.

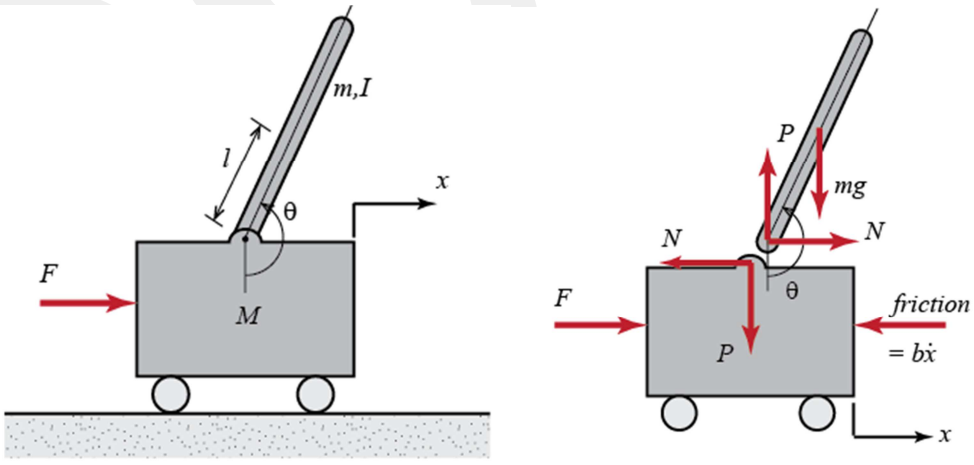


Figure 3.6. Inverted Pendulum and the free body diagram

The equations are derived using the free body diagram.

$$M\ddot{x} + b\dot{x} + N = F \qquad (1)$$

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \quad (2)$$

From (1) and (2), first governing equation:

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \quad (3)$$

and

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \quad (4)$$

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \quad (5)$$

From (4) and (5), second governing equation:

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \quad (6)$$

These equations are linearized about the vertically upward equilibrium position, θ=π. Therefore, the deviation angle, ϕ, should be referenced to 0. That is, θ= π + ϕ. Linearization is done as follows:

$$\cos\theta = \cos(\pi + \phi) \approx -1$$
$$\sin\theta = \sin(\pi + \phi) \approx -\phi$$
$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0$$

After the linearization, the equations for inverted pendulum can be written as follows:

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x}$$
$$(m + M)\ddot{x} + b\dot{x} - ml\ddot{\phi} = F$$

Here, I is the inertia of the cart and b is the coefficient of friction for cart.

The listing for model is shown in Figure 3.7.

```
model InversePendulum
  parameter Real m = 0.2, M = 0.5, b = 0.1, l = 0.3, I = 0.006, g =
9.81;
  Real F, ref = 0, p = 15, d = 0.3;
  output Real x(start = 0.5), phi(start = 0.1 * 3.1415 / 180);
  output Real vx, vang;
equation
  (I + m * l ^ 2) * der(vang) - m * g * l * phi = m * l * der(vx);
  (M + m) * der(vx) + b * der(x) - m * l * der(vang) = F;
  der(phi) = vang;
  der(x) = vx;
  F = p * (ref - phi) + d * der(ref - phi);
end InversePendulum;
```

Figure 3.7. Inverted Pendulum model using inertia inside the equations of motions.

In this model, a PD controller is used to control the pendulum. Reference angle is set to 0 because of the angle ϕ is the deviation. The results for the simulation of controlled system can be seen in Figure 3.8, Figure 3.9 and Figure 3.10.

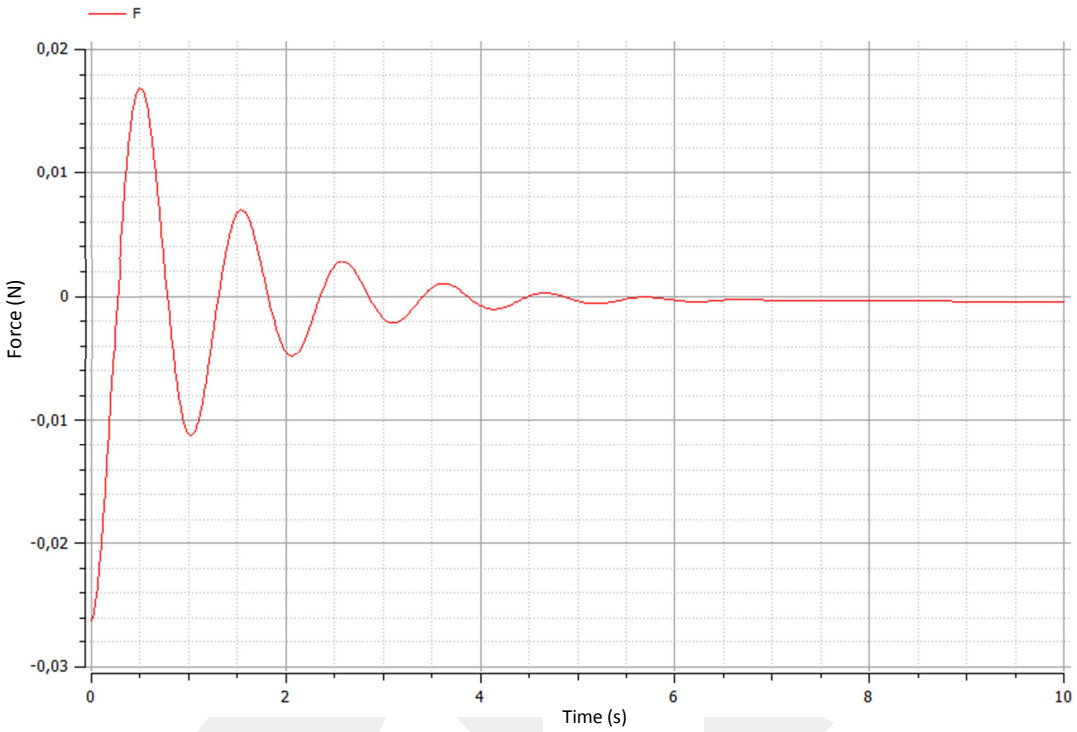


Figure 3.8. The input force given to the system with inertia

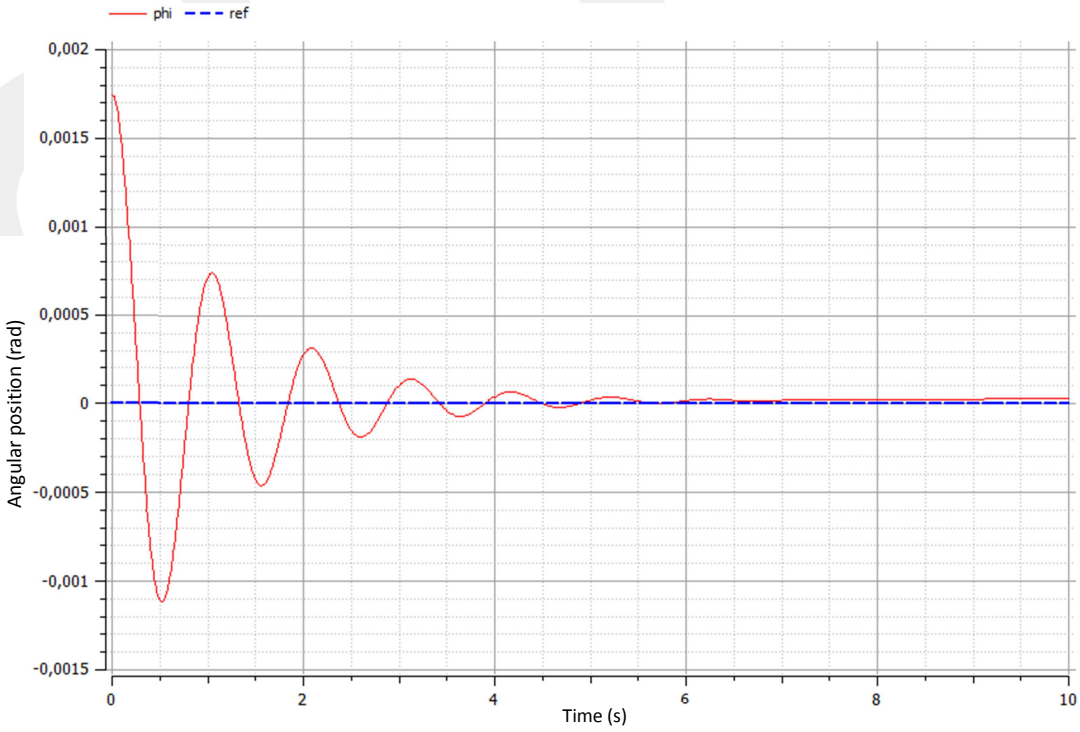


Figure 3.9. Reference angle and actual phi for system with inertia

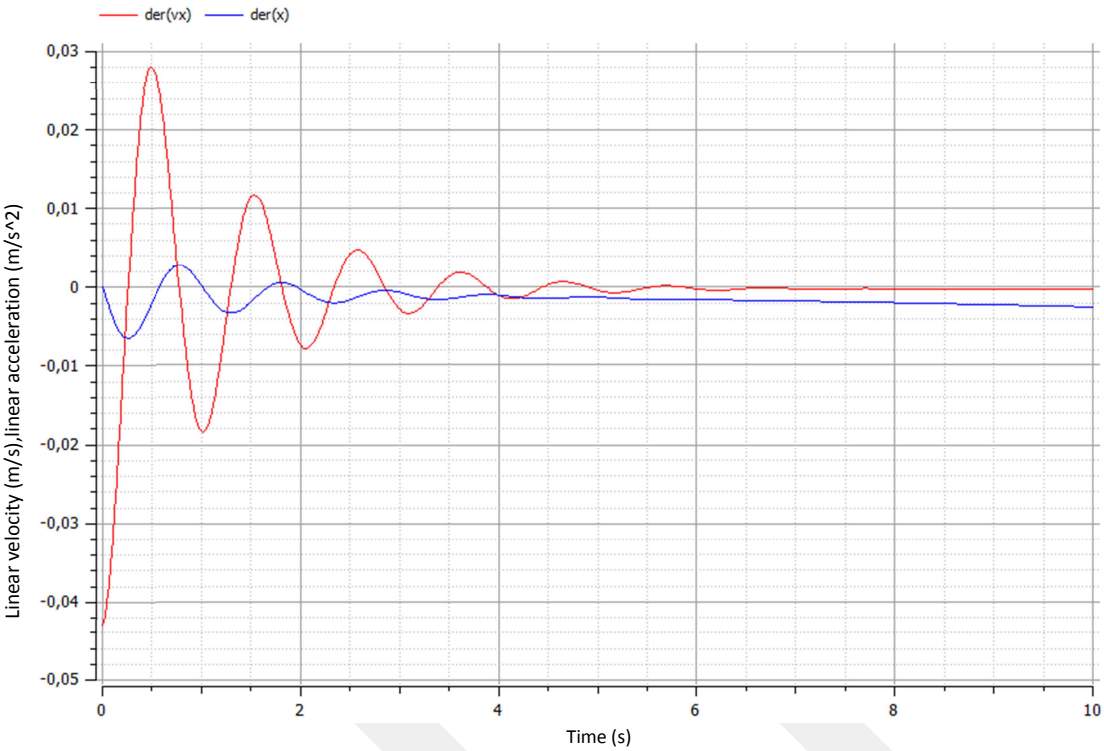


Figure 3.10. Linear speed and acceleration for system with inertia

3.1.3. Modelling Using Object-Orientation and Graphical Interface

The inverted pendulum is also modelled using the graphical interface of OpenModelica.

The pendulum is modelled using seven different classes. These classes are: Cart, Pendulum, Link, Controller, AngularSensor, SensorConnector and the base class InvertedPendulum, which is the class where all objects are connected to each other.

The classes are explained in detail below.

3.1.3.1. Cart

This class includes the simple parameters for the cart part of the inverted pendulum (Figure 3.11). These parameters are given as:

- Input force
- Mass of the cart

- Force between the pendulum and the cart
- Angle between the pendulum and the cart
- Linear position of the cart

```
model Cart
  constant Real pi = 3.1415, g = 9.81;
  Real Mass = 0.5;
  Real Angle(start = 1);
  Real T, x, xdot, xddot;
  Real F;
  //pendulumForce, acceleration
  //Link L;
  //Signal signal;
  Link L annotation(Placement(visible =
{24, 24}}, rotation = 0), iconTransform
40}}, rotation = 0)));
  Signal signal annotation(Placement(vi
{{-22, -22}, {22, 22}}, rotation = 0),
{27, 27}}, rotation = 0)));
equation
  F = signal.signal;
  T = L.Tforce;
  Angle = L.angle;
  xdot = der(x);
  xddot = der(xdot);
  F - T * sin(Angle) = Mass * xddot;
```

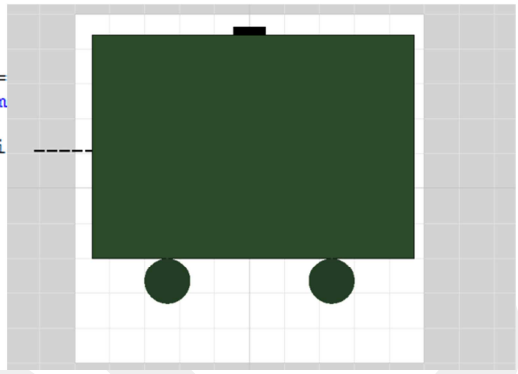


Figure 3.11. The graphical display for the cart

3.1.3.2. Pendulum

The pendulum class can be seen in Figure 3.12, including the parameters of

- Pendulum mass,
- Link length,
- Angle between pendulum and the cart,
- Force between pendulum and the cart,
- Angular acceleration.

```
model Pendulum
  constant Real pi = 3.1415, g = 9.81;
  Real mass = 0.1;
  Real angle;
  Real length = 0.5;
  Real T, a, ax, ay;
  //Link lnk;
  //AngularSensor sensor;
  //pendulumForce, acceleration
  AngularSensor sensor annotation(Placement(v
{{-56, -56}, {56, 56}}, rotation = 0), iconTr
{40, 40}}, rotation = 0)));
  Link lnk annotation(Placement(visible = tru
{41, 41}}, rotation = 0), iconTransformation(
rotation = 0)));
equation
  a = ax + ay;
  T * sin(angle) = mass * ax;
  (-T * cos(angle)) - mass * g = mass * ay;
  lnk.Tforce = T;
  lnk.angle = angle;
  sensor.Angle = angle;
```

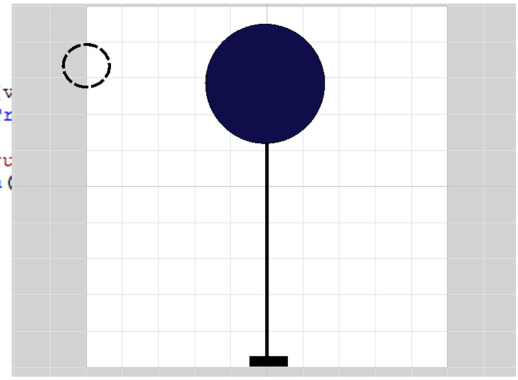


Figure 3.12. The Pendulum listing and the graphical display

3.1.3.3. Link

Link class is a connector type object. This class is the main responsible class for the connection between the pendulum and the cart. Also, the common variables for these two classes, the angle and force between cart and pendulum are shared using this object. Figure 3.13 shows the Link class.

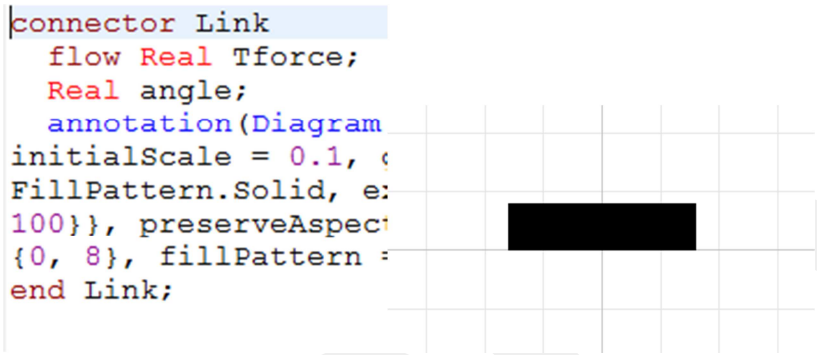


Figure 3.13. The listing for the Link class and the icon used for it.

3.1.3.4. Controller

Controller class is the one that includes the controlling equations. A PD controller is used inside the class. The desired theta angle is defined in this class. Also, this class is connected with the Pendulum class and the Cart class independently. The needed parameter (actual theta) is provided by the Pendulum class using the AngularSensor class, and the desired signal is given to the Cart class via Signal class. The Controller class can be seen in Figure 3.14.

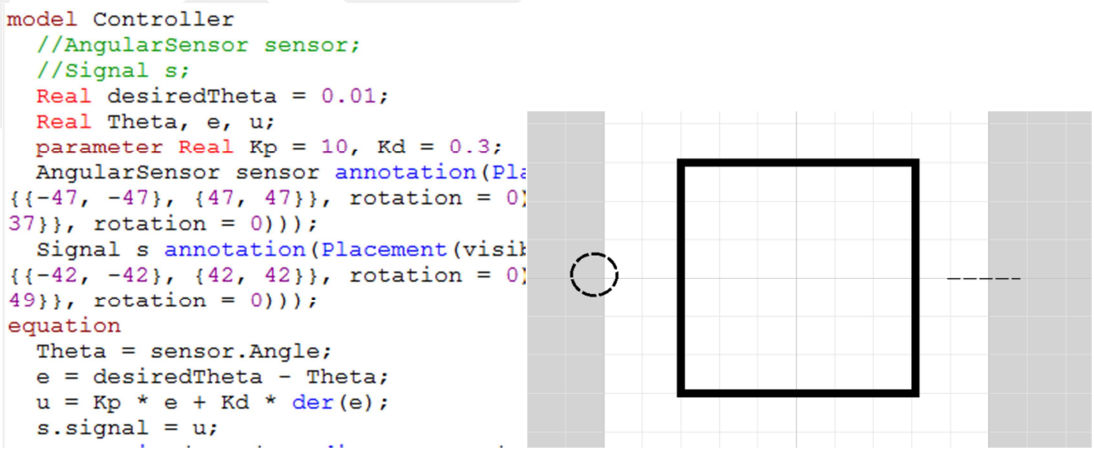


Figure 3.14. The Controller class model and the graphical display.

3.1.3.5. Angular Sensor

AngularSensor class is a connector class between the Pendulum and Controller classes (Figure 3.15). This connector carries the variable “Angle” and gives the opportunity to use it in Controller class.

```
connector AngularSensor
  Real Angle;
  annotation(Icon(coordina
initialScale = 0.1, grid =
lineThickness = 2, extent
{{-100, -100}, {100, 100}}
{Ellipse(origin = {1, 1},
-31}}, endAngle = 360)}));
end AngularSensor;
```

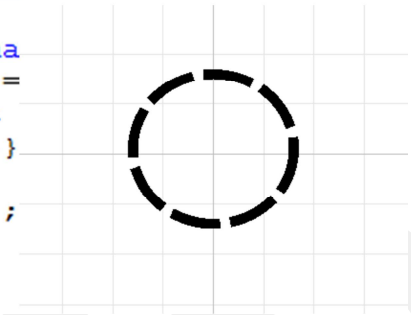


Figure 3.15. Angular sensor class and the icon used for it

3.1.3.6. Signal

The Signal class is mainly responsible for carrying the control signal to the Cart class as the input force. This class is a connector type class and only includes the “signal” parameter. Figure 3.16 shows the listing for the Signal class and its icon used in the graphical modelling.

```
connector Signal
  Real signal;
  annotation(Diagra
initialScale = 0.1,
0}}, pattern = Line
100}}, preserveAspe
points = {{-59, 0},
end Signal;
```


Figure 3.16. The Signal class and the graphical display.

3.1.3.7. Total Inverted Pendulum Model

The inverted pendulum system is set up in this model. The defined classes are connected to each other inside this class. The graphical model setup is shown in Figure 3.17.

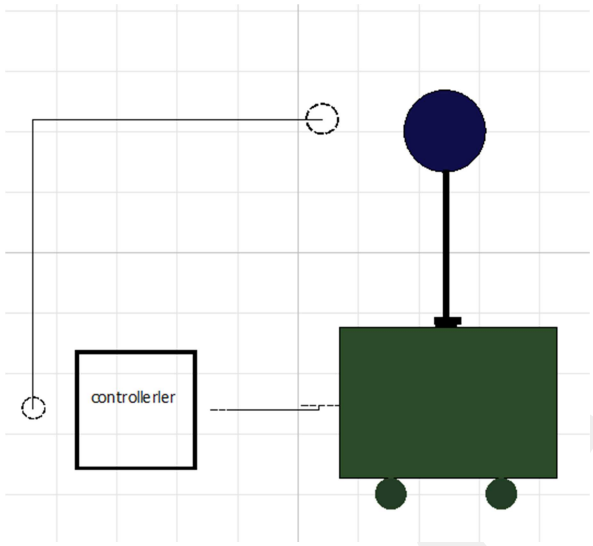


Figure 3.17. The connection between the elements in the system.

After the connections are set up, the related equations are provided in the writable part. Only the equations for motions are given inside this model. All other necessary variables and calculations are made inside the part classes and the graphical connection also provides the needed relation at the backstage of the OpenModelica software. The listing for the Inverted Pendulum model is shown in Figure 3.18, as well as the icon is shown in Figure 3.19.

```
model InvertedPendulum
  constant Real pi = 3.1415, g = 9.81;
  Real mp, mc, theta, thetadot, thetaddot;
  Real x, xdot, xddot;
  Real Lp;
  Real F;
  Cart Cart annotation(Placement(visible = true, transformation(origin = {49
-39}, {39, 39}}, rotation = 0)));
  Pendulum pendulum annotation(Placement(visible = true, transformation(orig
{{-41, -41}, {41, 41}}, rotation = 0)));
  Controller kontrol annotation(Placement(visible = true, transformation(ori
{{-32, -32}, {32, 32}}, rotation = 0)));
equation
  connect(pendulum.sensor, kontrol.sensor) annotation(Line(points = {{8, 44}
-52}}));
  connect(kontrol.s, Cart.signal) annotation(Line(points = {{-23, -52}, {7,
  connect(Cart.L, pendulum.lnk) annotation(Line(points = {{49, -25}, {49, -2
  F = Cart.F;
  Lp = pendulum.length;
  mp = pendulum.mass;
  mc = Cart.Mass;
  theta = Cart.Angle;
  thetadot = der(theta);
  thetaddot = der(thetadot);
  x = Cart.x;
  xdot = der(x);
  xddot = der(xdot);
  -mp * g * sin(theta) = mp * xddot * cos(theta) - mp * Lp * thetaddot;
  F + mp * Lp * thetaddot * cos(theta) - mp * Lp * thetadot ^ 2 * sin(theta)
```

Figure 3.18. The listing for the Inverted Pendulum Class

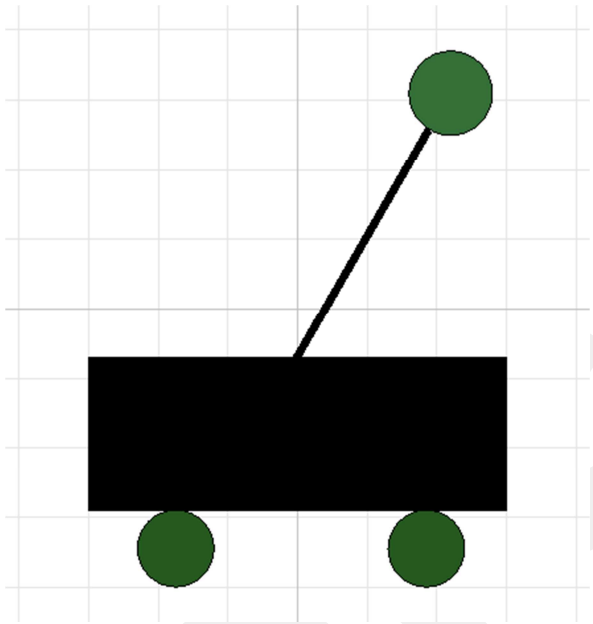


Figure 3.19. Icon for the Inverted Pendulum system.

Results for the system can be seen Figure 3.20, Figure 3.21 and Figure 3.22.

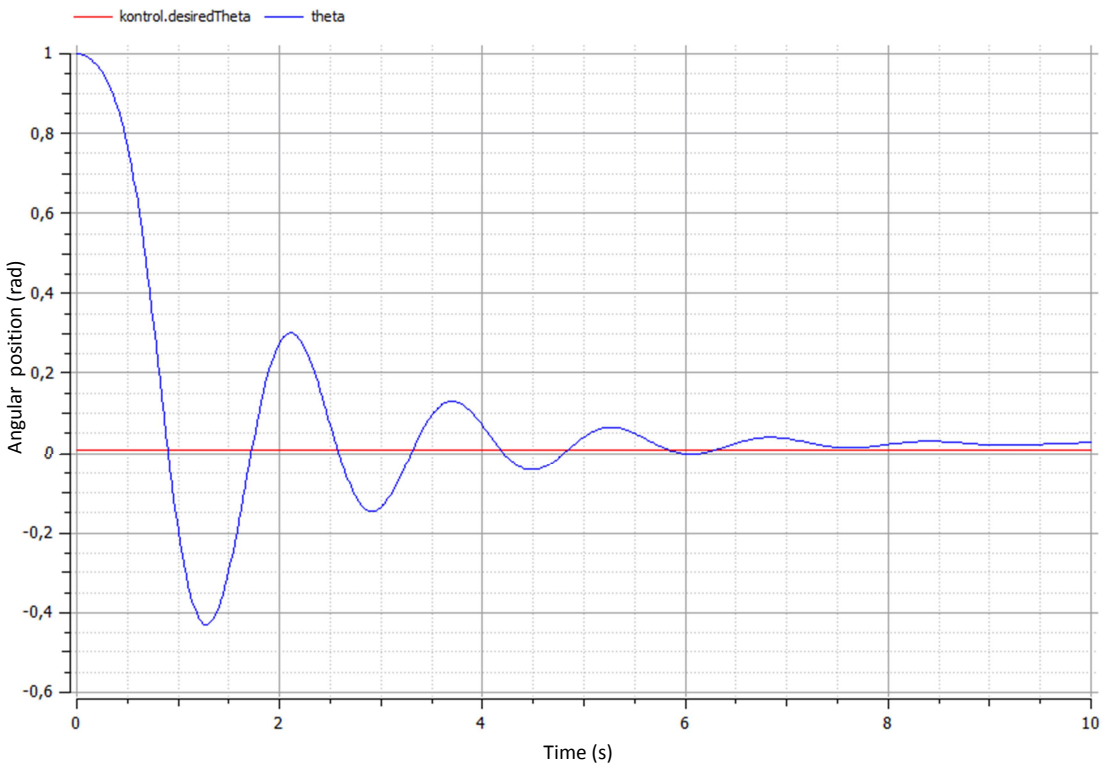


Figure 3.20. Theta and reference angle for object-oriented modelled pendulum

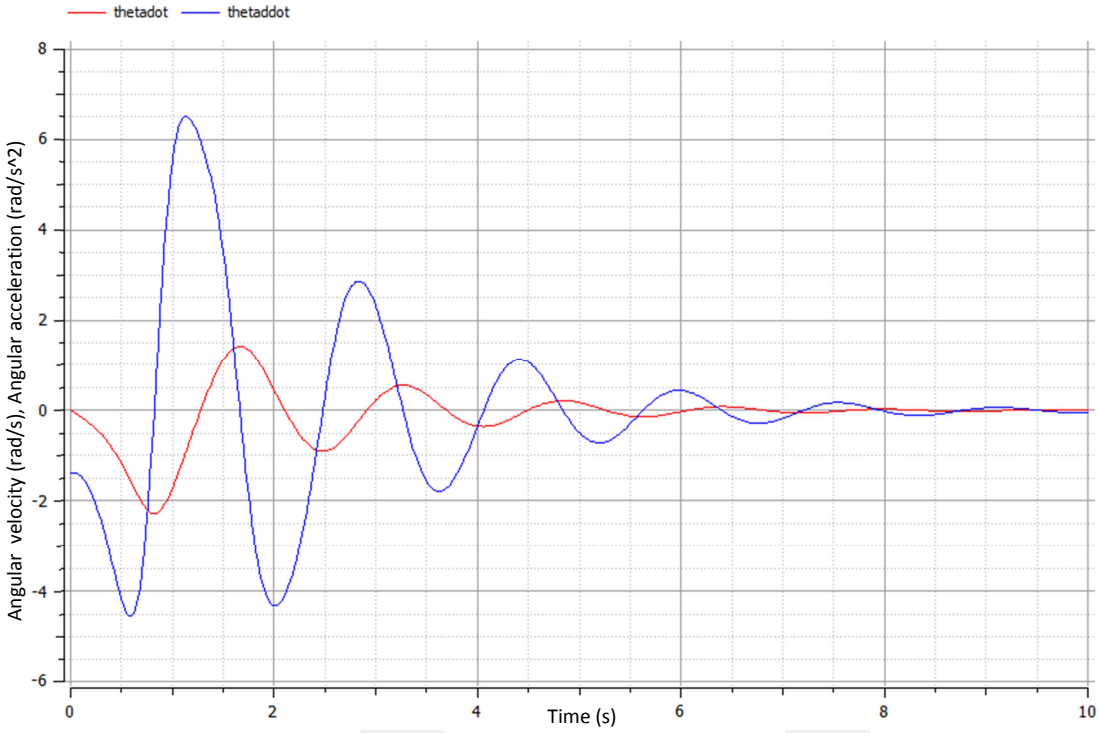


Figure 3.21. Angular speed and acceleration for object-oriented modelled pendulum system

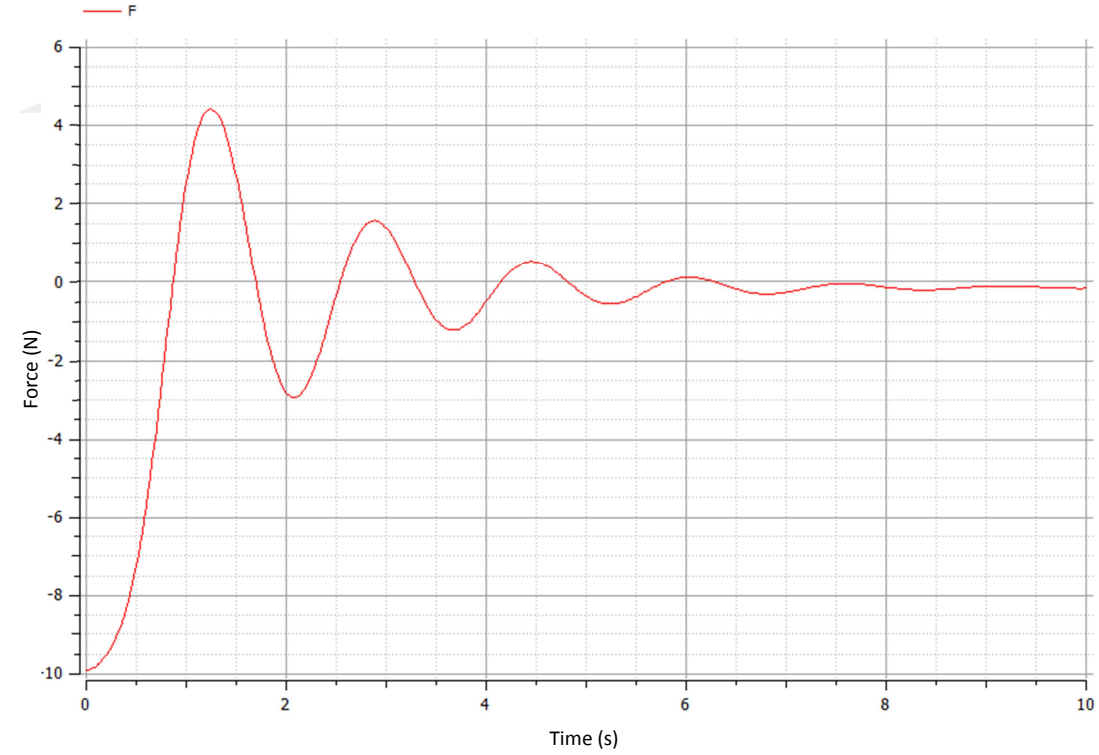


Figure 3.22. Input force for object-oriented modelled pendulum system

3.1.4. Conclusion for Inverted Pendulum System

The inverted pendulum system was modelled in OpenModelica using both writable and graphical platforms. First, the system was modelled in writable platform and equations of motions are used in only one class. After that, a more complex system is modelled again in writable part. Lastly, the system is modelled part by part, using different classes. The integration of the classes are provided inside another class. All three systems follow the desired reference and the control of the inverted pendulum system is successfully modelled and simulated using OpenModelica platform.

Inside the Control Tutorials for MATLAB and Simulink webpage, several dynamic systems are clearly explained besides the inverted pendulum system. These systems are also examined and DC motor system is implemented on OpenModelica to clarify the dynamic system modelling and simulation using Modelica language. In order not to destroy the integrity of the thesis, DC motor modelling and simulation results are placed in Appendix part, and if required, can be seen in Appendix.

3.2. Rose Harvesting Robot

Rose Harvesting Robot is a robot which is used in greenhouses for the aim of harvesting rose flowers in order to sell in flower stores. This robot has been developing in Mechatronics Laboratory in Atılım University by researchers. In this chapter, the Rose Harvesting Robot will be examined in detail in functional model and will be modelled. After modelling the system, the model will be adapted to OpenModelica and simulated to observe the inputs and outputs, as well as the system behaviour.

3.2.1. System overall

Rose Harvesting Robot is a kind of robot that receives the pots, which carry the rose plants. After receiving the plants, the robot senses the roses upside from the pot, in other words, the sensors take the image of the rose plant from top view. After taking the image, robot processes this image and detects the rose flowers' positions and ripeness. The ripe roses are determined to be cut. After the decision making, the mechanical part is run. The rose flowers which will be cut are found by the arm. After that, the cutting position is decided after taking the image of the stem and processing the image. Then, the gripper holds the stem and the shears cut the stem.

The rose flower is put into a container. All ripe roses are cut and after harvesting, the trimmed rose plant will be sent to its place.

3.2.2. Inputs and outputs of Rose Harvesting Robot

Like every system, Rose Harvesting Robot has inputs, outputs and a process inside the system. The inputs and outputs of the system will be explained below.

First input of Rose Harvesting Robot is the rose plant. Second input is the *image* of rose plant. The image is taken by the system using sensors. As said before, sensors are chosen as camera for this system. The camera takes the image of the rose plant from top view. After receiving the input information, the related subsystem will process on it. The required information can be defined as follows:

- The presence of rose flowers
- The ripeness of the rose flowers
- The positions of the rose flowers

Processing the image of the rose plant, the system obtains this useful information in two steps. First, the shape and diameter of the rose flowers, the colour of the rose flowers, etc. are determined by the system using the image. In second step, *using* this information the useful information written above is achieved. These are the first outputs of the system.

This process can be named as the information conversion. The beginning input information is processed and converted. The converted information is used again inside the system to make decisions to be able to perform the system task correctly. The ripe rose flowers are detected and their information is used. Correct information enables system to send required signals to mechanical subsystem and the cutting system perform the tasks.

System task is realized in the mechanical part. These tasks can be clarified as follows:

- Finding the ripe rose flower using the position information of the rose flower
- Cutting the rose flower

As it can be seen clearly, the mechanical part uses the input material. The input rose plant possesses the rose flowers which are wanted to harvest. The mechanical arm finds the ripe rose flowers one by one. Then, the shears cut the rose flower from its stem.

Here, a material change is in question. The input material, rose plant, undergoes a change. The possessed rose flowers are cut from the rose plant. Also, there is a material production. There is a new material produced; this is the rose flower. This whole process is named as the material conversion. Inside the system, also another new material produced from the input, this is the trimmed rose plant.

A more detailed system structure can be seen in Figure 3.23. In this figure, Rose Harvesting Robot is shown by means of the information, material and energy flow. The information conversion and energy conversion, and the system process that is the rose harvest can be seen in the same figure.

Also, the inputs of the image of rose plant, electrical energy and rose plant material, and the outputs of trimmed rose plant, rose flowers and feedback information can be seen clearly.

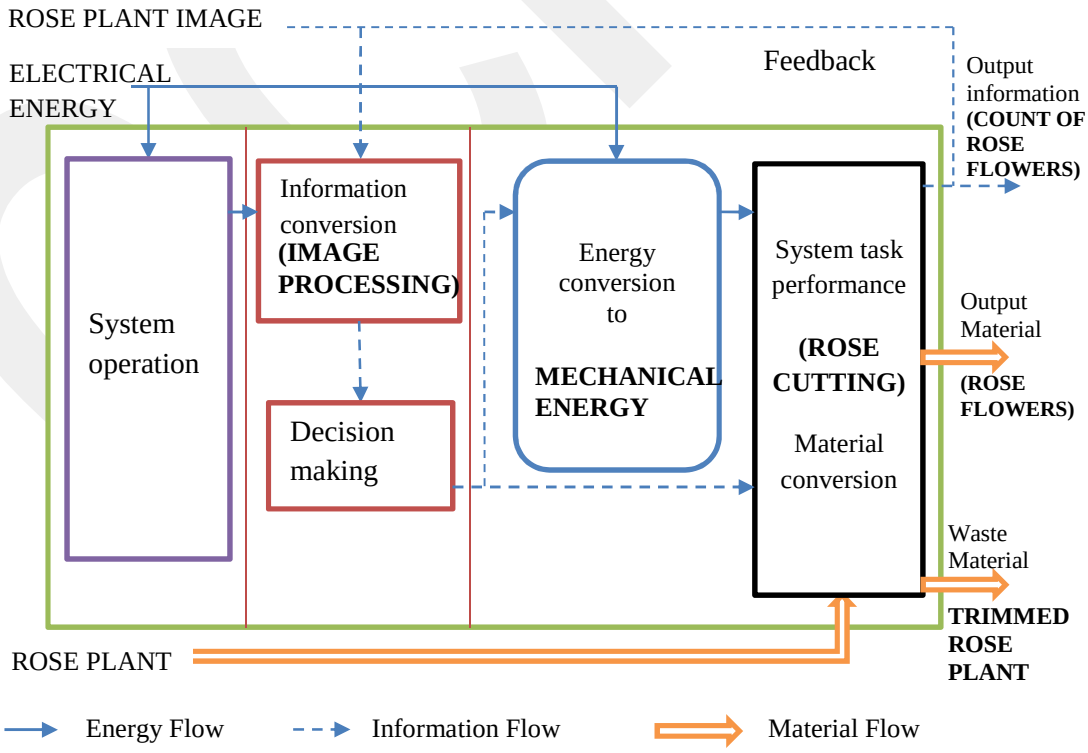


Figure 3.23. The overall system of Rose Harvesting Robot

3.2.3. Rose Harvesting Robot Function Model

The Rose Harvesting Robot system will be modelled using component modelling. This type of modelling eases the understanding about the component structure of the system. Sub models can be easily set up due to the functional model; also the relations between inputs and outputs can be seen clearly.

On the other hand, system will be modelled in a simplified form. The stem grabbing and cutting position determination parts are omitted from the system which will be modelled on the simulation tool to prevent the complexity. Moreover, the environmental objects will not be modelled and implemented on the simulation platform. Only the Rose Harvesting Robot and its components will be the major issue for this study due to the component modelling. It is useful for the writer to indicate that the transportation of the pots is not the subject for this system, especially in the context of this study. Therefore, neither in modelling nor in simulation parts the pot transportation will be explained.

Now that, the Rose Harvesting Robot will be examined in the view of its components.

Rose Harvesting Robot System consists of three subsystems. These are Robot Vision, Operator and Cartesian Robot Arm subsystems. The component model of Rose Harvesting Robot is shown in Figure 3.24.

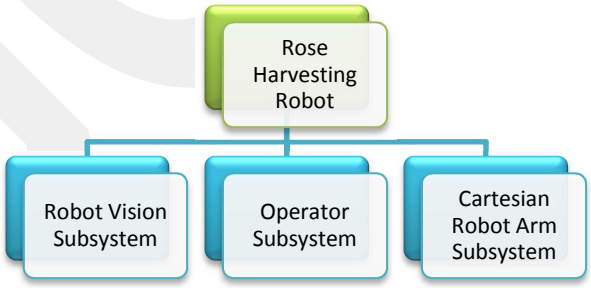


Figure 3.24. Rose Harvesting Robot subsystems

Each subsystem will be examined in detail under separate headings. As preliminary information, the system will be briefly explained in this part.

- **Robot Vision Subsystem:** Is a type of electronic subsystem. The realization of the rose plant will be performed using electronic solutions. In modelling,

functional model will be used. Inside the model, this subsystem is also assumed as the image processing part. Because of the real system uses a digital camera for information gathering, this equipment is met using some algorithms during the implementation. The information about the input is obtained with this method. Therefore, it can be said that the information process starts in this subsystem.

- **Operator Subsystem:** Is a type of Information subsystem. The important decisions are made inside this subsystem as well as operational control. This subsystem is a computer in real system. This computer is used as a bridge between the Robot Vision Subsystem and Mechanical Subsystem. In modelling of Operator Subsystem, it will be in the same position. Functional model will be used for implementation and its main functions will be defined inside the implementation part.
- **Cartesian Robot Subsystem:** Is a type of Mechanical subsystem. This Robot Arm is the processing part of Rose Harvesting Robot. This part performs the harvest. Therefore, this is a crucial part for the Robot. This mechanical subsystem is modelled mathematically first. After that, it is implemented on computer platform.

3.2.4. Rose Harvesting Robot Modelling in OpenModelica

In OpenModelica, first the objects are implemented. After defining the objects, the system process is attempted to be implemented. The outputs are observed from the system model implementation. The relations between the objects are established and the system simulation is executed. In this chapter, the study on Rose Harvesting Robot using OpenModelica will be explained in detail.

In real world, explained three subsystems are combined inside a higher level system (Rose Harvesting Robot). The subsystems are connected to each other in this platform and forms the top-level system. This assembly combines different kinds of systems and composes a mechatronic system. In virtual medium, these three subsystems are implemented and they are connected to each other in a higher level system, establishing a mechatronic system model.

Some assumptions are made during the modelling. These are written below.

RobotVision part will not consider the z-axis. The recognition of the stem is not taken into consideration.

Input images include only one rose flower. Recognition is made accordingly. Also, the images have a maximum size of 25x25 pixels. The images are assumed as taken from the pot having a size of maximum 750x750 mm.

Mechanical part is assumed as showing the displacement of the robot arm. The scissors and cutting procedure is omitted from the mechanical part. Only a CUT command (1 or 0) expresses the cutting process.

The implementation of Rose Harvesting Robot on OpenModelica is made as a class, putting all three subsystem implementations together. Rose Harvesting Robot class can be seen in Figure 3.25. The classes of Operator, RobotVision and Cartesian will be explained in detail in next sections.

```
model RoseHarvesting
  Operator op;
  RobotVision v;
  Cartesian c;
equation
  connect(v.p, op.q);
  connect(op.r, c.p);
end RoseHarvesting;
```

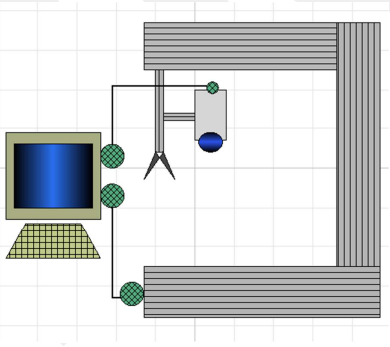


Figure 3.25. Rose Harvesting Robot class and the graphical representation

3.2.5. General Structure of Robot Vision Subsystem

Under the Rose Harvesting Robot, a robotic vision part is included. Robot Vision Subsystem is the information source for Rose Harvesting Robot. The visualization is realised by this part. A simple display for Robot Vision Subsystem is shown in Figure 3.26.

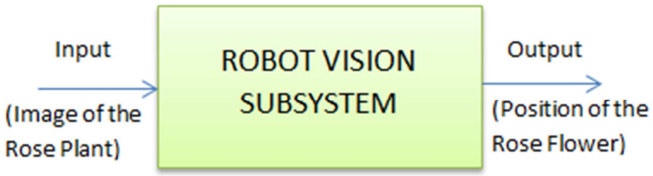


Figure 3.26. Robot Vision Subsystem, input and output

This subsystem should be separated into components. The decomposition model gives the needed objects while using the object-oriented modelling. The separation of the Robot vision Subsystem is shown in Figure 3.27.

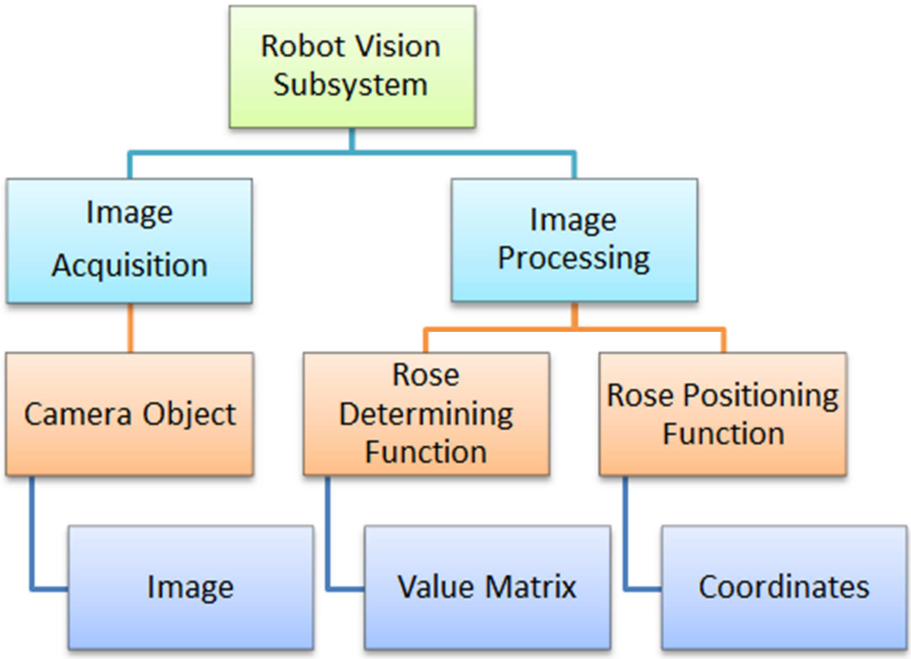


Figure 3.27. The Components included by Robot Vision Subsystem

As it can be seen in Figure 3.27, the Robot Vision Subsystem has a hardware part for image acquisition, and algorithms for two main functions:

- Sensing the rose flower (Recognition)
- Finding the position of the rose flower (Localization)

For meeting these two functions, first the hardware image acquisition part will be setup. Secondly, algorithms for segmentation and localization are used. The hardware and software parts will be separated in this study, because the algorithms to segment the rose flower and positioning will be used inside the Operator, which is the information subsystem of the Rose Harvesting Robot. Inside the Robot Vision Subsystem, chosen hardware and the input information will be explained. After that, the components of Robot Vision Subsystem, the Camera object, Image object and the Function objects, will be defined. After that, the Operator Subsystem will use the

related image processing algorithms and hence The Rose Harvesting Robot will sense and segment the rose flower.

In real system, chosen alternative for image acquisition is using a digital camera (Figure 3.28). The camera used on Rose Harvesting Robot has the properties seen in Figure 3.29.

Figure 3.28. The camera used in real system

Image	
Codec image size (H × V)	1920 × 1440, 1600 × 1200, 1680 × 1056, 1920 × 1080, 1440 × 1280 × 1024, 1024 × 768, 1024 × 576, 800 × 480, 768 × 576, 7 384 × 288, 320 × 240, 320 × 192 (H.264, MPEG-4, JPEG)
Video compression format	H.264, MPEG-4, JPEG
Codec streaming capability	Dual streaming
Maximum frame rate	H264: 20 fps (1920 × 1440) / 30 fps (1920 × 1080) MPEG-4: 15 fps (1920 × 1440) / 20 fps (1920 × 1080) JPEG: 10 fps (1920 × 1440) / 15 fps (1920 × 1080)

Figure 3.29. Specifications of the camera related to image

It is obvious that the input information for image processing algorithm is the photograph of the rose plant in real system. However, based on the conceptual design, while modelling the Robot Vision system, the photograph input set will be created by researcher and passed over to the related function objects for the process of Recognition and Localization.

From this point, the input set should be created carefully. The assumed inputs should be suitable for processing. Therefore, firstly the image properties of the real setup should be known. Image definition should be carefully made. After that, related implementations can be done in a healthy way.

3.2.5.1. Image Definition

As it is known the previous parts, robot vision subsystem obtains the images from real world. In specific, the vision subsystem of Rose Harvesting Robot takes the photograph of the rose plant and image is processed by the Function objects. Therefore, the object *Image* should be clearly defined just before the modelling of the Robot Vision Subsystem. After this step, the other steps for image processing can be done in an easy way.

A digital image is formed of pixels and each pixel has three values named R, G, and B (Figure 3.30). The letters stand for Red, Green and Blue. Each of these has a numerical value ranging from 0 to 255. Therefore, the mixture of these three values expresses secondary colours (Figure 3.31).

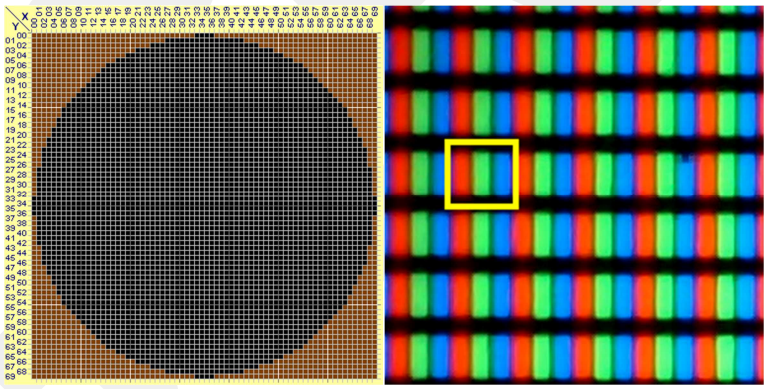


Figure 3.30. Pixels of an image and the RGB displaying of one pixel

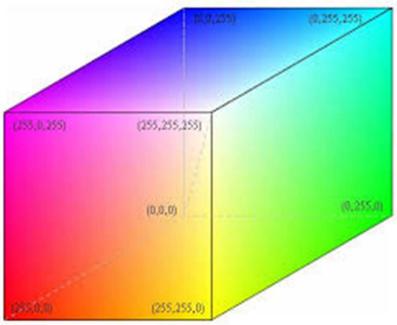


Figure 3.31. RGB colour space

3.2.5.2. Input Image for Rose Harvesting Robot Model

The image taken from the camera will be defined as a matrix of pixels, which have own RGB values and this information will be used during the image processing. In physical system, the real photograph has the resolution of 640x480 pixels. However, in this study the Image object is defined in a simple manner in order to achieve easiness and avoid unclearness. It is assumed that the size of the image taken from the Camera is approximately 20 times reduced. Therefore, the image has a matrix value of less than 32x24. Because of not being possible to use physical sources in modelling, this minimization will help the image production for simulation from a hypothetical resource. Another point is related with OpenModelica software. OpenModelica platform cannot be able to cope with higher resolution images. This issue will be mentioned later.

The actual images used in Rose Harvesting Robot show the rose plant and rose flowers. Therefore, while filtering, the RGB values for rose flowers will be used. To do this, a minimum and a maximum RGB value for a rose flower will be determined. An assumption related to this point may be said as used rose flowers will be a shade of red (Figure 3.32).

Figure 3.32. Rose flower

3.2.5.3. Input data pool for images

A sample input data pool is created using 20 images taken previously. For image process, the RGB matrices of sample images will be used. Because of the lack of necessary functions inside OpenModelica, these RGB matrices are obtained as a

preliminary work by using MATLAB. Therefore, the input space is previously determined by the researcher with the help of MATLAB and arranged with the help of Microsoft Excel before using in OpenModelica. A sample input image which obtained sing the rose flower image seen in Figure 3.32 is shown in Figure 3.33.

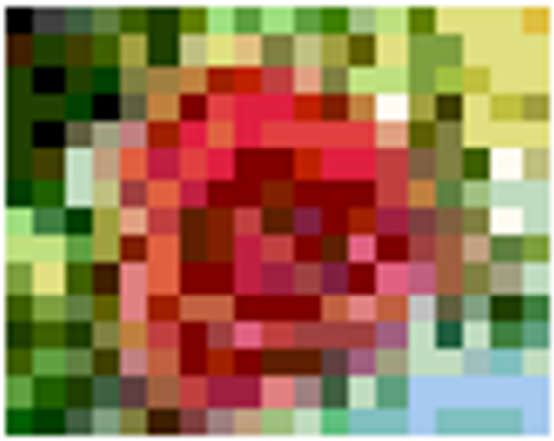


Figure 3.33. The small-size image obtained from a rose flower image.

The RGB matrices of related image are obtained using MATLAB, and the values are seen in Figure 3.34.

R

28,75,58,89,55,37,82,155,88,160,103,78,171,202,93,228,255,255,226
31,30,48,51,148,39,191,228,214,122,176,148,84,230,132,141,255,238,255
35,[illegible],6,129,151,202,150,192,198,235,141,187,206,120,160,189,255,252
29,29,12,1,104,187,139,222,239,233,184,137,176,255,147,57,244,186,168
44,7,99,175,206,159,211,255,234,246,250,229,228,251,103,112,225,255,230
16,59,220,183,209,195,241,224,155,142,178,212,240,201,133,121,72,255,195
0,37,209,176,160,225,182,115,118,139,101,136,148,207,178,111,166,193,222
163,64,11,145,243,192,84,132,176,89,133,117,168,170,132,123,129,231,217
188,199,111,159,123,220,86,134,179,204,108,110,226,139,152,160,180,94,136
137,238,79,38,221,220,116,107,176,174,150,133,101,240,198,146,136,145,220
76,128,88,86,237,163,197,197,120,122,106,186,255,202,227,86,129,23,62
26,78,42,124,202,196,103,164,225,195,160,148,173,170,213,33,193,51,97
60,101,107,56,193,201,150,154,126,94,100,106,144,146,160,162,147,119,157
90,41,21,78,111,168,193,157,159,210,161,77,196,101,156,158,179,166,180
13,3,49,121,116,33,112,167,178,169,171,110,128,119,150,124,126,112,137

G

12,74,86,126,82,68,130,212,146,220,165,128,187,212,118,246,251,255,200
29,37,76,83,171,54,203,230,202,114,189,172,104,251,159,159,253,222,255
56,[illegible],1,32,142,136,139,40,36,51,151,112,212,250,150,176,195,255,250
55,34,34,8,84,124,14,49,24,25,31,41,143,255,162,74,251,191,173
55,18,107,154,129,31,41,81,43,55,57,49,76,154,88,131,230,255,230
36,79,239,162,109,45,66,47,0,0,0,16,42,57,97,136,82,255,201
45,93,255,184,75,95,47,0,3,24,0,0,0,68,133,118,184,216,244
223,124,68,155,161,70,0,24,66,0,38,7,16,34,65,106,141,255,243
216,228,144,150,31,95,0,21,34,66,14,17,89,0,51,103,174,112,152
146,253,103,26,122,92,13,0,25,29,57,46,0,109,107,100,137,171,246
84,146,119,80,140,33,96,95,0,0,13,93,131,86,194,103,168,76,115
46,103,76,122,114,71,0,50,92,71,70,65,69,101,230,94,255,121,167
105,144,143,52,117,82,15,8,0,0,13,52,110,150,224,251,230,197,235
155,98,60,84,69,88,78,32,43,123,123,81,236,171,249,255,255,255,255
95,73,94,147,126,17,60,116,163,187,225,198,250,248,255,226,242,234,255

B

0,44,35,56,1,0,28,107,43,120,66,31,98,116,14,136,140,147,89
8,4[illegible]16,93,0,117,147,128,41,110,88,17,156,56,51,145,111,150
25,[illegible],0,73,71,86,5,21,35,115,54,131,155,54,77,89,156,139
16,12,0,0,33,81,0,45,39,39,26,16,90,189,79,0,155,89,70
15,0,60,111,101,18,44,93,59,73,76,60,71,122,23,49,146,167,140
0,26,183,119,93,46,79,63,11,0,17,38,59,49,47,67,13,193,131
0,18,187,127,54,93,51,0,6,29,0,1,0,65,94,64,136,170,198
137,38,0,94,137,65,0,24,69,0,36,8,28,36,39,63,95,211,196
131,144,65,91,8,91,0,23,49,79,12,12,96,5,43,73,114,38,81
63,170,25,0,101,89,6,0,40,42,52,39,0,117,102,74,79,97,172
0,62,39,20,121,31,88,90,0,0,6,88,145,95,179,71,121,22,61
0,20,0,65,94,69,0,49,97,73,62,59,76,104,211,61,224,84,130
20,64,73,4,101,84,22,19,0,0,6,40,101,136,200,223,202,171,209
71,21,0,38,47,81,83,38,44,114,102,54,209,145,228,239,254,239,255
13,0,35,102,91,0,47,99,132,145,175,147,201,210,239,213,221,209,236

Figure 3.34. R matrix (left), G matrix (right) and B matrix (below).

3.2.5.4. Implementation on OpenModelica

As previously explained, the Robot Vision Subsystem will be implemented on OpenModelica platform, including the classes Camera, Image, Recognition and Localization and Robot Vision itself, as an upper level class. In this part, these classes will be implemented respectively.

a. Camera Class

Appropriately, in modelling, a Camera class is created (Figure 3.35). Inside the Camera class, there is an Image object which will be used as an input. Also, the size of the image is kept as attributes m and n, taking the information by access of the attributes of Image object *foto*.

```
model Camera
  Image foto;
  Integer m, n;
equation
  m = foto.rowsize;
  n = foto.colsize;
end Camera;
```

Figure 3.35. The Camera class

b. Image Class

Input image is held in Image class (Figure 3.36). As explained previously, an image has three components as R, G and B. Therefore, inside the Image class, three matrices are created as Red, Green and Blue. These matrices are composed of the data obtained by the preliminary work told before. An Image object will also have the size information in terms of pixels.

```
model Image
  parameter Integer Red[15, 19] = {... ....};
  parameter Integer Green[15, 19] = {... ....};
  parameter Integer Blue[15, 19] = {... ....};
  Integer rowsize, colsize;
equation
  rowsize = size(Red, 1);
  colsize = size(Red, 2);
end Image;
```

Figure 3.36. The Image class

c. Recognition Class

This class is a kind of Function class in OpenModelica. The flowchart related to this function can be seen in Figure 3.37. The implementation can be seen in Figure 3.38.

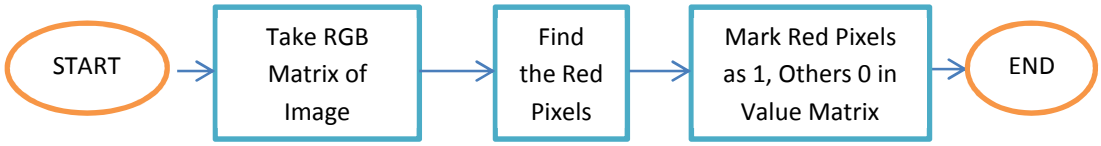


Figure 3.37. Flowchart for Recognition function

```
function Recognition
  input Integer Red[:, :];
  input Integer Green[:, :];
  input Integer Blue[:, :];
  input Integer m;
  input Integer n;
  output Integer Val[m, n];
protected
  Integer Value[m, n];
algorithm
  for i in 1:m loop
    for j in 1:n loop
      if Red[i, j] > 100 and Red[i, j] < 255 then
        if Green[i, j] > 0 and Green[i, j] < 80 then
          if Blue[i, j] > 0 and Blue[i, j] < 80 then
            Value[i, j] := 1;
          else
            Value[i, j] := 0;
          end if;
        else
          Value[i, j] := 0;
        end if;
      else
        Value[i, j] := 0;
      end if;
    end for;
  end for;
  Val := Value;
end Recognition;
```

Figure 3.38. Class definition for Recognition function

The RGB matrix is actually is in dimension [m, n, 3]. Here, m and n stands for the size (resolution, pixel number) of the image. The third variable shows that this image consists of three matrices named as Red, Green and Blue. Each matrix has values range from 0 to 255.

To be able to process the image with respect to Red, Green and Blue components, some treshold values must be determined first. Since the red component is significant for this study, the values greater than 100 in Red matrix will be taken into

consideration in concurrence with the values inside the Green and Blue matrices, which have the numbers less than 80. That means, the pixels which have Red value greater than 100 and Green and Blue values less than 80 are subject for rose flower. These pixels show the position of the rose flower and will be kept inside a *Value* matrix, having a value 1. Other pixels will be kept as 0. This process is named as digitizing. The Value matrix for the sample rose can be seen in Figure 3.39.

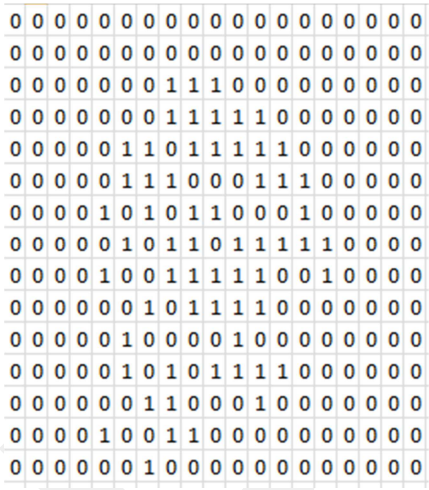


Figure 3.39. The digits 1 shows the red pixels.

d. Localization Class

The implementation of Localization Function is done by using two parts as XCoord and YCoord functions. After digitizing process, the coordinates of the rose flower are found by using the Xcoord and Ycoord functions. These functions simply take the Value matrix as an input and find the minimum and maximum values of the positions of 1's and take the average for x-coordinate and y-coordinate (Figure 3.40). The resulting value shows the position of the rose flower with respect to x and y axes. The implementation can be seen in Figure 3.41.

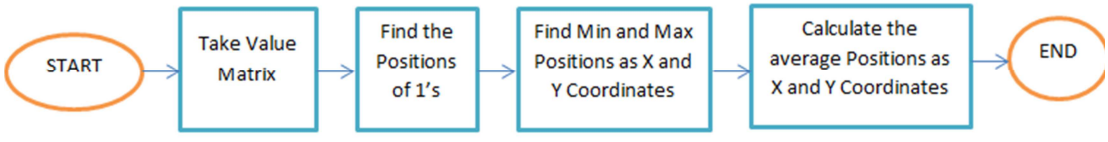


Figure 3.40. The flowchart for Localization Function

```
function Xcoord
  input Integer Value[:, :];
  output Real xCoord;
protected
  parameter Integer m = size(Value, 1);
  parameter Integer n = size(Value, 2);
  Integer a[n];
  Integer x[n];
  Integer flag, point, isaret;
  Integer min, max;
  Real sum, ave[m];
algorithm
  sum := 0;
  isaret := 0;
  for i in 1:m loop
    flag := 100;
    min := 0;
    point := 0;
    max := 0;
    a[:] := Value[i, :];
    for j in 1:n loop
      if a[j] == 1 then
        x[j] := j;
        if x[j] < flag and x[j] > 0
          min := x[j];
          flag := j;
        end if;
        if x[j] > point then
          max := x[j];
          point := j;
        end if;
      else
        x[j] := 0;
      end if;
    end for;
    ave[i] := (min + max) / 2;
  end for;
  for i in 1:m loop
    if ave[i] > 0 then
      sum := sum + ave[i];
      isaret := isaret + 1;
    end if;
  end for;
  xCoord := sum / isaret;
end Xcoord;
```

Figure 3.41. The implementation of XCoord Localization Class

If it is necessary to look in a level deeper, Xcoord function takes each row of Value matrix and finds the 1's positions. At the same time, keeps the minimum position and the maximum position of 1's. Then, takes the average of the two positions. For each row, this process is done. Then, summation of all averages is made and the final average value is found for x-axis.

The same process is done for y-axis by using the columns of the Value matrix. This is a simple way for finding the x and y coordinates of the rose flower. These two functions are taken as Localization Object together.

e. Rose Area Class

This is a function class that returns the 1's number inside the Value matrix. This number implies the area of the rose flower. The listing is shown in Figure 3.42.

```
function Rosarea
  input Integer Value[:, :];
  output Integer area;
protected
  parameter Integer m = size(Value, 1);
  parameter Integer n = size(Value, 2);
  Integer a[n];
  Integer sum;
algorithm
  sum := 0;
  for i in 1:m loop
    a[:] := Value[i, :];
    for j in 1:n loop
      if a[j] == 1 then
        sum := sum + 1;
      else
        sum := sum + 0;
      end if;
    end for;
  end for;
  area := sum;
end Rosarea;
```

Figure 3.42. Area Class

f. Robot Vision Class

This class is an upper-level class, representing the whole of Robot Vision Subsystem. The Camera (and hence, the Image) object, function objects Recognition and Localization are combined in this frame class. The implementation of Robot Vision Class can be seen in Figure 3.43.

```
model RobotVision
  Camera cam;
  Integer m, n;
  Integer Value[15,19];
  Integer R[:,:] = cam.foto.Red;
  Integer G [:,:]= cam.foto.Green;
  Integer B[:,:] = cam.foto.Blue;
  Real XC;
  Real YC;
equation
  m = cam.m;
  n = cam.n;
  Value  = Recognition(R,G,B,m,n);
  XC = Xcoord(Value);
  YC = Ycoord(Value);
end RobotVision;
```

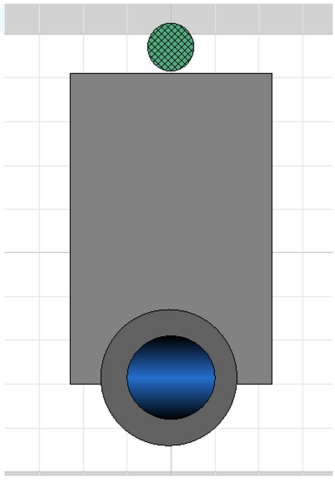


Figure 3.43. The Robot Vision Subsystem Class and the graphical representation

3.2.5.5. Simulation results

There are three images for each sample. These are the real image, the small-size of the real image and the digitized image. Each rose flower is also positioned by the algorithm written in OpenModelica.

Sample 1

First image:

Smalled to the size of 8x13:

After RGB Filter:

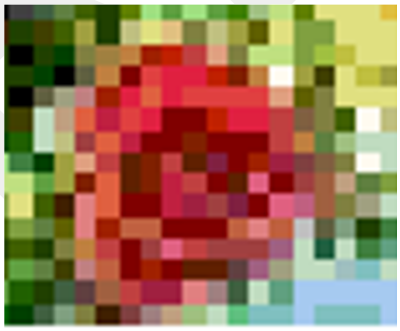
X-coordinate: 6.75

Y-coordinate: 5.25

Sample 2

First image:

Small-size image (19x15):



After RGB Filter:

X-coordinate: 9.19231

Y-coordinate: 8.54545

Sample 3

First image:

Small-size image (22x17):

Digitized image:

Found coordinates:

X-coordinate: 8.42857

Y-coordinate: 6

While transferring the pixel coordinates into physical world, it is assumed that the size of the pot is maximum 750x750 mm, and because the image is downscaled 30 times, pixel coordinates are multiplied by 30.

Using OpenModelica, the Robot Vision Subsystem is implemented. The algorithm to detect the rose flower and the algorithm to find the coordinates of that rose are developed. However, OpenModelica is insufficient with image processing, because it cannot manage big-sized matrices. Bigger matrices cause more time to calculate, and sometimes program gives error and shut down before the process is completed. To overcome this situation, MATLAB is used as a helpful tool. Consequently,

OpenModelica can be said as not very suitable for image processing; however, it can provide results at average.

3.2.6. Operator Subsystem

Operator Subsystem is core part of Rose Harvesting Robot. Operator subsystem is the part responsible for the information process. That means, in fact, the developed algorithms related to Mechanical Subsystem and Robot Vision Subsystem are run on this subsystem. It may look simple; however, this subsystem is elementary. The connection between the Robot Vision and Mechanical Subsystems is set by Operator. This subsystem decides the ripeness of rose flower and sends the related information to Cartesian Robot Arm (Figure 3.44).

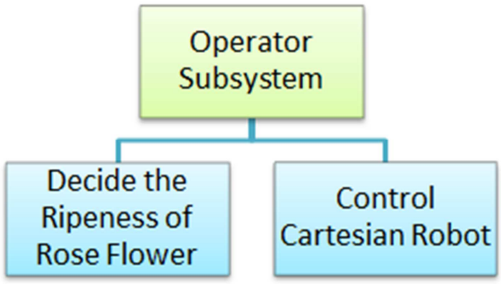


Figure 3.44. Operator Subsystem functions

In real system, the Operator is a computer. In modelling, this part will be shown as a class in OpenModelica. As the real system, this Operator Class will connect the Robot Vision Class to Cartesian Class. Previous studies show that Cartesian Class needs input coordinates to go that position and cut the rose. This coordinate information is generated by Robot Vision Class. Generated coordinates are updated inside the Operator to have the correct proportion for Cartesian Robot. After that, the coordinates are submitted to the Cartesian Robot Arm. Therefore, Cartesian Robot arm can reach the desired position viewed by Robot Vision part. The input and output relation of Operator Subsystem can be seen in Figure 3.45.

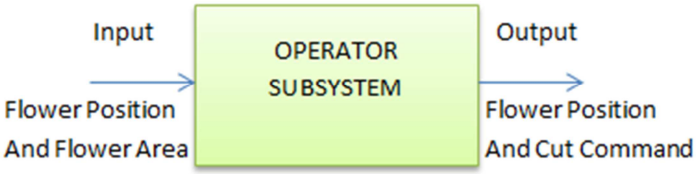


Figure 3.45. Input/output relation of Operator Subsystem

The rose cutting is also covered in Operator Subsystem. The related algorithm can be seen in Figure 3.46. Area information is provided by again Robot Vision Class. If the rose has Area more than 15 pixels, this flower will assumed as ripe. This treshold value is only and assumption to be able to reach the main aim. Treshold value can be easily changed according to desired values.

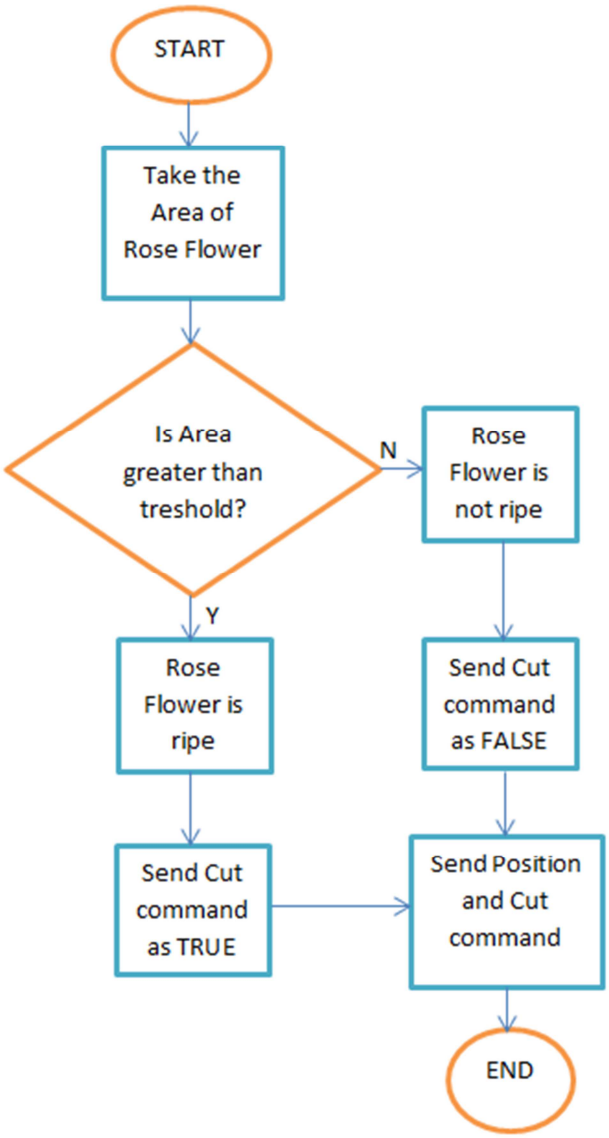


Figure 3.46. Determination of ripe roses to be cut

3.2.6.1. Implementation of Operator Subsystem on OpenModelica

Inside OpenModelica, an Operator class is created (Figure 3.47). This class has two variables for Coordinates of the rose flower and a variable to keep the rose flower's area. The graphical representation of Operator class can also be seen in Figure 3.47.

```
model Operator
  Port q, r;
  Real cut, tru;
  Real xpos, ypos;
equation
  cut = q.cut;
  xpos = q.x;
  ypos = q.y;
algorithm
  if cut >= 30 then
    tru := 1;
  else
    tru := 0;
  end if;
equation
  r.x = xpos * 30;
  r.y = ypos * 30;
  r.cut = tru;
end Operator;
```

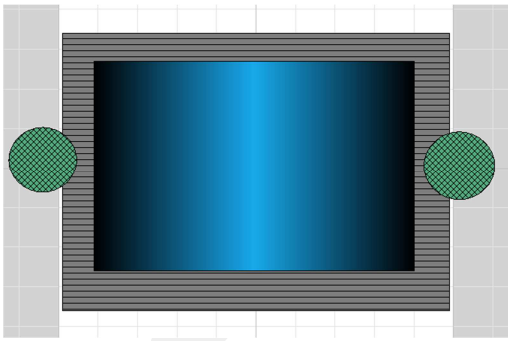


Figure 3.47. Operator Class definition and the graphical representation

Also, a Port class, which is a connector type, is created to be able to connect the other classes to Operator class (Figure 3.48). The objects of Port class, q and r, can be seen in Figure 3.47.

```
connector Port
  Real x;
  Real y;
  Real cut;
end Port;
```

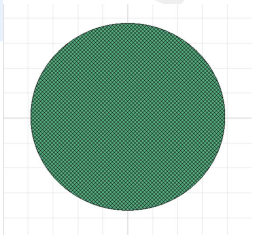


Figure 3.48. The Port Class and the graphical representation

The algorithm section inside the Operator class is deciding the ripeness of the rose. The output value will be sent to the Cartesian Robot Arm together with the Position information.

In conclusion, Operator class is implemented on OpenModelica, to be a bridge between the Robot Vision class and Cartesian class. The information transfer is realized via Operator class. This class also uses the taken information and makes an important decision for Cartesian class, which the information of the rose is ripe enough for harvest. This decision controls the Cartesian Robot Arm.

3.2.7. Cartesian Robot Arm Subsystem

The mechanical part of the Rose Harvesting Robot consists of simply a three-dimensional cartesian robot. In this report, the modelling and simulation of a three-dimensional cartesian robot arm will be demonstrated. To be able to do this, the implementation will be made in OpenModelica and the simulation results will be provided.

3.2.7.1. Mechanical Subsystem Implementation of Rose Harvesting Robot

Using OpenModelica, the mechanical arm is modelled and the equations of motions are embedded in this model. Specifications of the servo motors on the Cartesian Robot are given below and can be seen in Figure 3.49.

$n_N = 4100$ rpm, $M_N = 2.29$ Nm, $U_N = 360$ V

$M_0 = 2.56$ Nm, $I_0 = 2.96$ A, $I_{max} = 10$ A

Nominal Speed: 4100 rpm

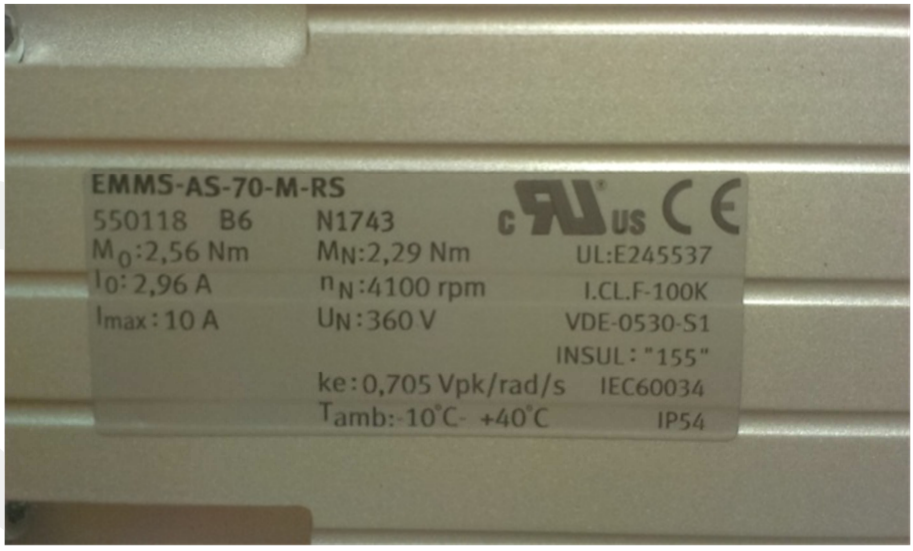


Figure 3.49. Servo motor on the cartesian robot (Festo®)

Also, used 3-axes cartesian robot arm's linear speed for each axis can be seen in Figure 3.50 (Festo Specifications Manual for 3D Cartesian Robot Arm).

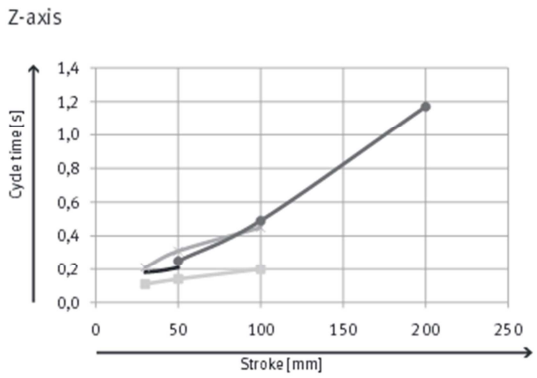


Figure 3.50.a. Z-axis linear position change

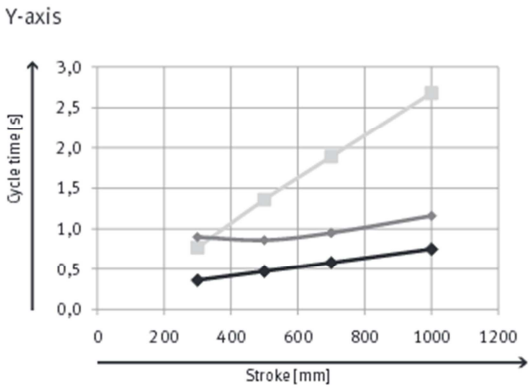


Figure 3.50.b. Y-axis linear position change

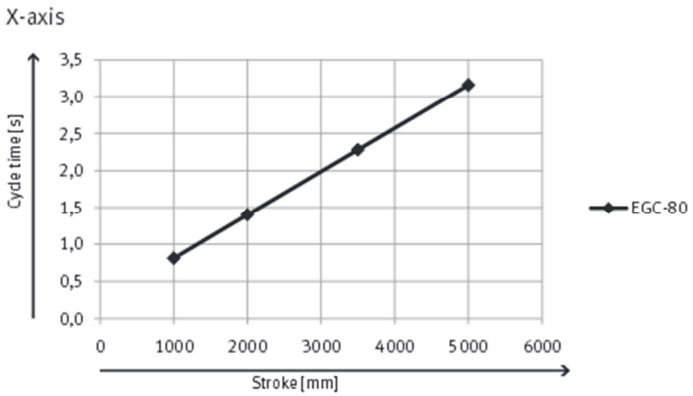


Figure 3.50.c. X-axis linear position change

Calculation for linear equations:

Linear speed for z-axis:

$$\frac{linear\ displacement}{time} = \frac{200-25}{1.2-0.2} = \frac{175}{1}$$

$$= 175\ mm/s$$

Linear speed for x-axis:

$$\frac{linear\ displacement}{time} = \frac{1000-300}{0.75-0.4} = \frac{700}{0.35}$$

$$= 2000\ mm/s$$

Linear speed for y-axis:

$$\frac{linear\ displacement}{time} = \frac{5000-1000}{3.2-0.8} = \frac{4000}{2.4}$$

$$= 1160\ mm/s$$

Using these calculations, the axis for the cartesian robot could be modelled using OpenModelica. The results for linear speeds for x, y and z axes are used inside the model of Cartesian robot. The equations of motion for the axes can be shown using the relation between position and velocity:

*CurrentPosition=Linear velocity*time*

For x axis: *CurrentPosition=1160mm/s*time*

For y axis: *CurrentPosition=1160mm/s*time*

For z axis: *CurrentPosition=1160mm/s*time*

a. The class Axis:

First, the Axis class is defined to be able to use in the upper class Cartesian. The three axes x, y and z share the common properties, therefore one class of all axes can be used to calculate the movement and simulate the motion of Robot Arm. After the creation of Axis class, the three axes x, y and z are instantiated as objects inside the upper class Cartesian.

Inside the Axis class, the linear equation of motion is written using the calculated speeds. Also, the axis can go to the desired position that taken from the upper class Cartesian. The algorithm is written inside the Axis class and after the desired position is reached, the related arm stops. The Axis class can be seen in Figure 3.51.

```
model Axis
  Real vel;
  Real t;
  Real desired;
  Real pos;
  Real d_pos;
  Real current_position;
  Real velocity;
equation
  d_pos = desired + time * 0;
  pos = vel * time;
algorithm
  if pos < d_pos then
    current_position := pos;
    velocity := der(pos);
  else
    current_position := d_pos;
    velocity := der(d_pos);
  end if;
  when current_position == d_pos then
    t := time;
  end when;
end Axis;
```

Figure 3.51. The description for one axis

b. The class Cartesian:

The x-axis, y-axis and z-axis of the cartesian robot are created using the Axis class. All axes are combined to each other inside the upper class named Cartesian. The model for 3-axes Cartesian Robot is created as Cartesian class. The variables needed inside the Axis objects x-axis, y-axis and z-axis are defined inside the Cartesian class. Also, the conditions for motion are included inside this class. The listing for the Cartesian class can be seen in Figure 3.52. This class is used to have the simulation for the mechanical system.

```
model Cartesian
  Axis x, y, z;
  Real xpos, ypos, zpos;
  Real xvel, yvel, zvel;
  Real desired_x = 3000;
  Real desired_y = 2500;
  Real desired_z = 30;
equation
  xvel = 1160;
  yvel = 2000;
  zvel = 175;
  x.vel = xvel;
  y.vel = yvel;
  z.vel = zvel;
  x.desired = desired_x;
  y.desired = desired_y;
  z.desired = desired_z;
  xpos = x.current_position;
  ypos = y.current_position;
  zpos = z.current_position;
end Cartesian;
```

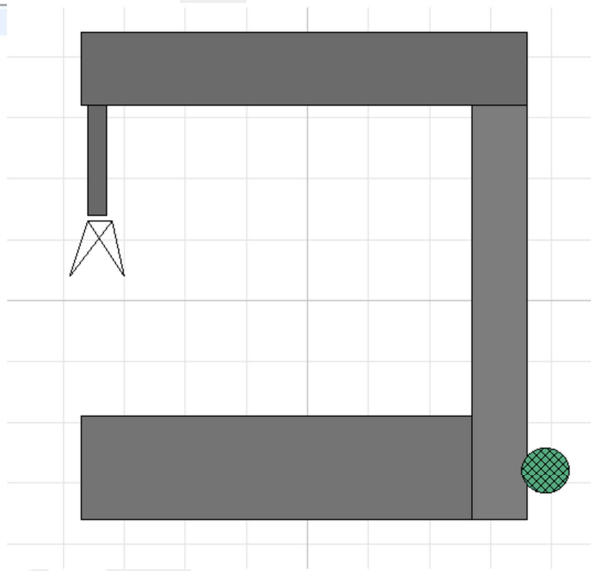


Figure 3.52. The Cartesian class for Cartesian Robot Arm and the graphical representation

3.2.7.2. The results for the simulation of Cartesian Robot Arm

The desired position defined in Cartesian class is turned into a line for easy readability of the current position of the robot arm. The desired x, y, z positions for the robot arm are parameterized inside the Cartesian class. Therefore, desired positions can be changed several times after one simulation of Cartesian. The parameters that can be changed are seen at the right side of Figure 3.53.

After the other components of Rose Harvesting Robot are modelled, the position information will be provided by the Operator. Therefore this Class will be updated with simple changes while linking the components of Rose Harvesting Robot to each other.

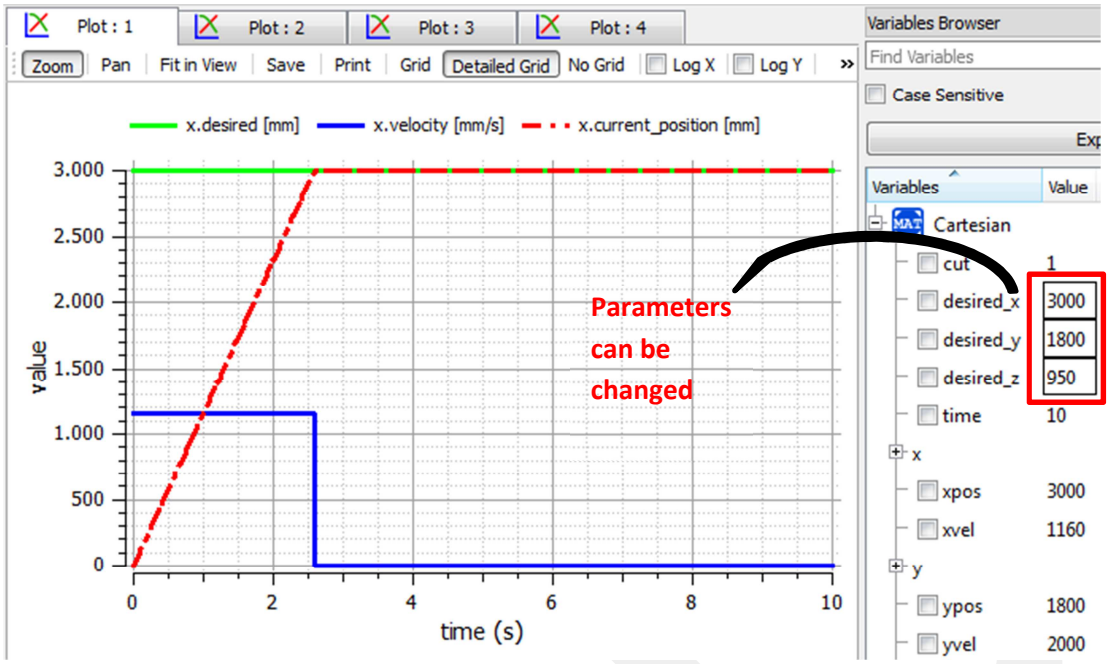


Figure 3.53. Desired position line, velocity and current position for x-axis of Robot Arm

As it can be seen in Figure 3.53, Figure 3.54 and Figure 3.55 the 3-axis robot arm can be implemented in OpenModelica using the linear motion equations for the real system. The velocity is constant during the operation of displacement. After the arm is stopped, the velocity will be zero. Also, the displacement line can be seen clearly.

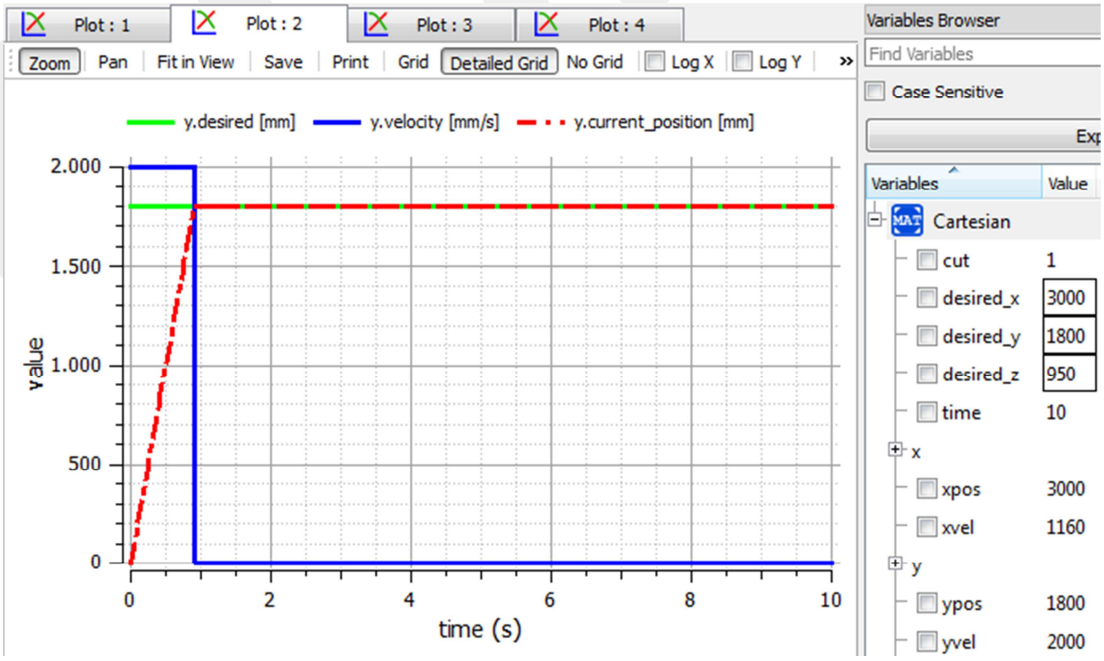


Figure 3.54. Desired position line, velocity and current position for y-axis of Robot Arm

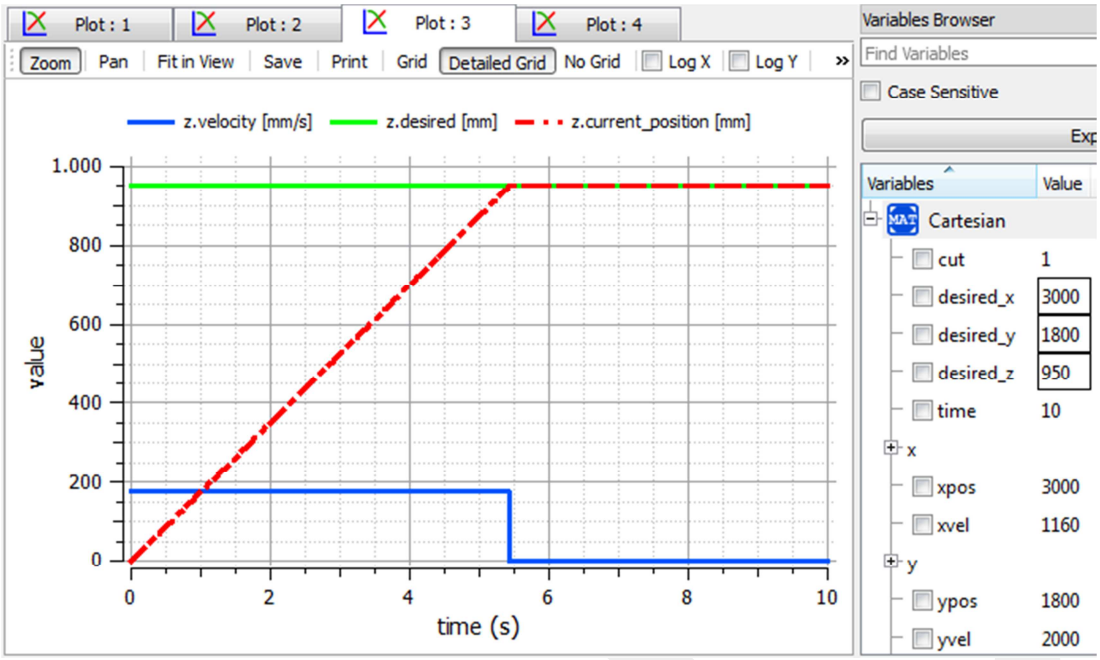


Figure 3.55. Desired position line, velocity and current position for z-axis of Robot Arm

The results after changing the values for desired_x, desired_y and desired_z can be seen in Figure 3.56, Figure 3.57 and Figure 3.58.

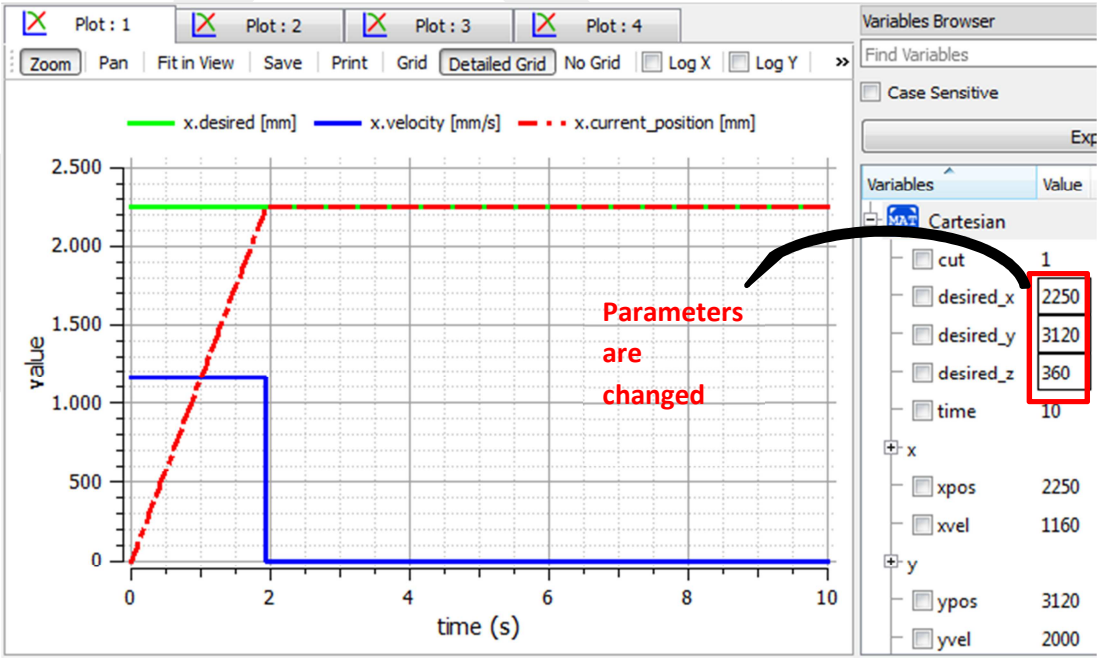


Figure 3.56. X-axis values after input change

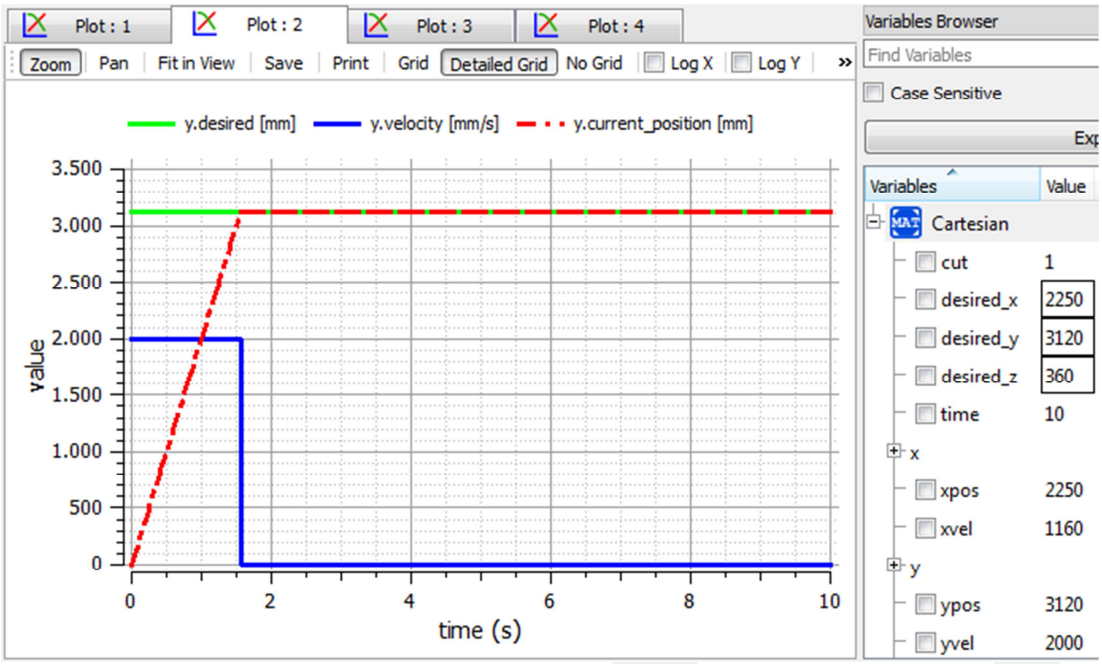


Figure 3.57. Y-axis values after input change

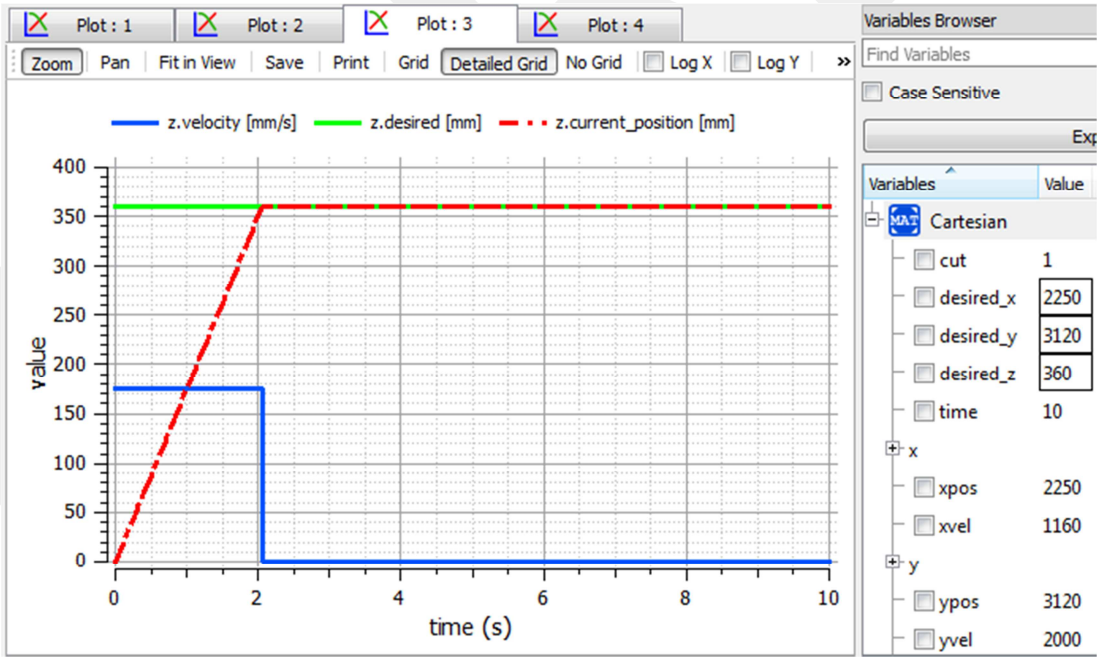


Figure 3.58. Z-axis values after input change

3.2.8. Connection of the classes in an upper-level class: RoseHarvest

After the three subsystems of Rose Harvesting Robot is defined, the graphical model can be setup inside an upper level class. This model is named as RoseHarvest inside

OpenModelica. The graphical icons of Camera, Operator and the Cartesian Robot Arm can be dragged and dropped from the object tree inside the OpenModelica. Then, the Ports can be connected to each other respectively. This means, it is not necessary to define anything inside the RoseHarvest class; OpenModelica does it instead of the researcher. Also, the physical topology of the Rose Harvesting Robot can be easily seen in this model. Using this method, the graphical representation and the coded implementation of Rose Harvesting Robot displayed in Chapter 3.2.4 is reached readily. The object tree window in OpenModelica and the graphical model are shown in Figure 3.59.

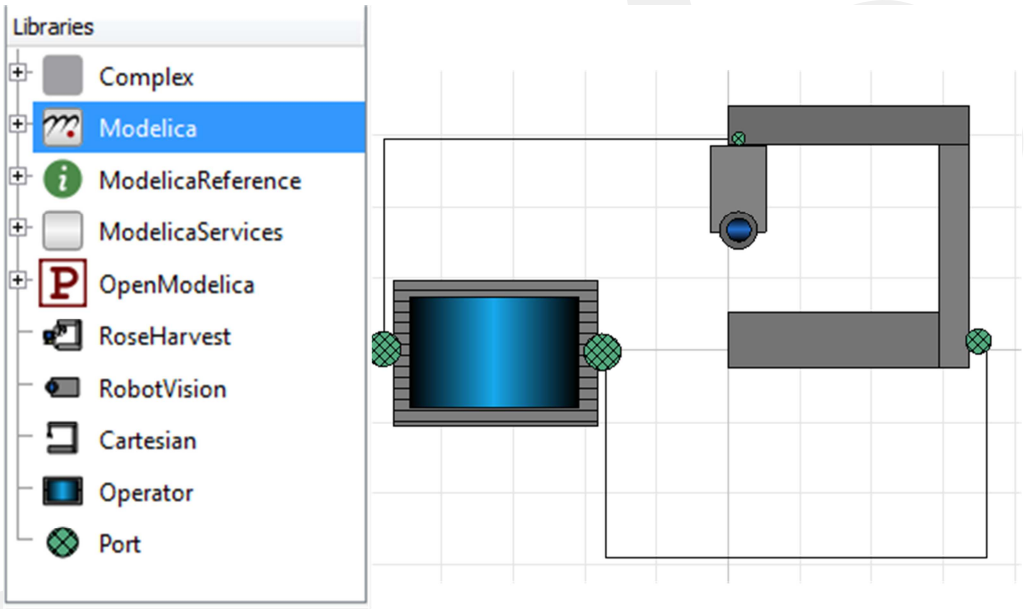


Figure 3.59. The object tree in OpenModelica and connection of different classes on an upper-level class

3.2.9. Simulation for Rose Harvesting Robot

After the Simulation model is established, the simulation setup is done on OpenModelica. The start time, simulation duration and the method of the solution can be chosen in this setup. Also, other necessary advanced adjustments can be easily chosen in the simulation setup such as determining the step size of the simulation, changing the launch type, or saving the simulation settings. Also, there are some other beneficial tabs to determine the output properties and simulation flags, and access to the previous simulation archive. The dialog box of simulation setup can be seen on Figure 3.60.

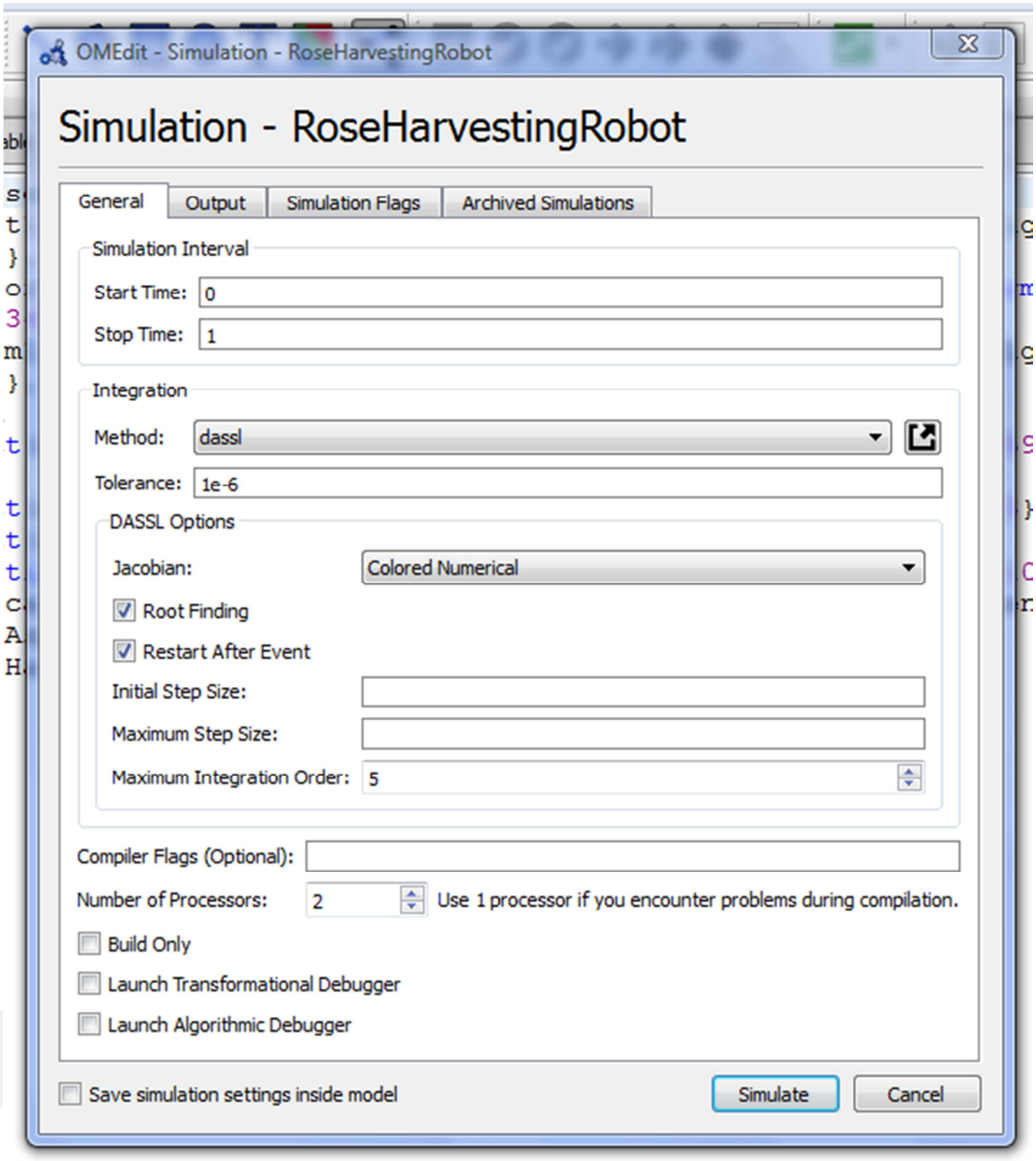


Figure 3.60. The Simulation setup dialog box in OpenModelica

For the simulation of Rose Harvesting Robot, the default options are used in simulation setup and the simulation is run. Running simulation can be seen in Figure 3.61 and Figure 3.62.

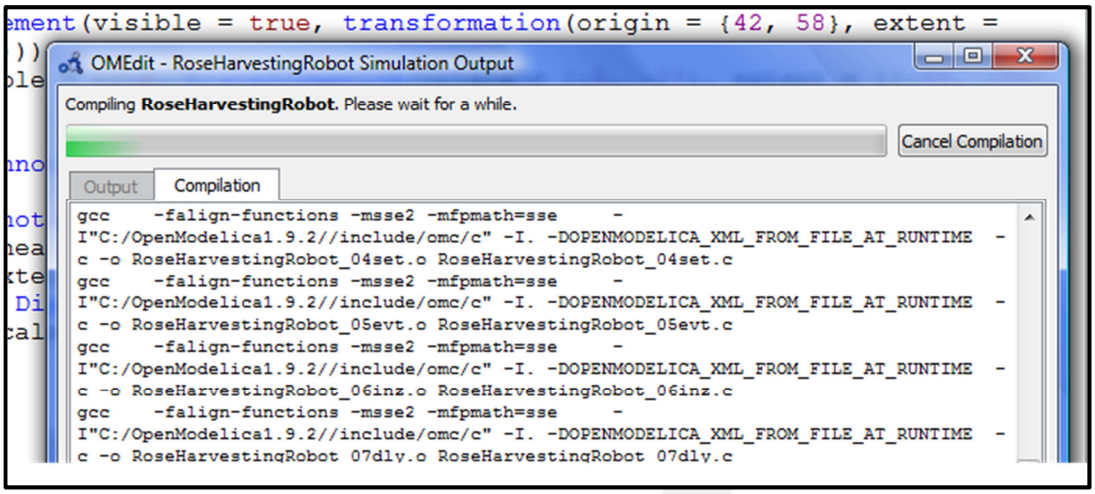


Figure 3.61. The simulation is running by OpenModelica

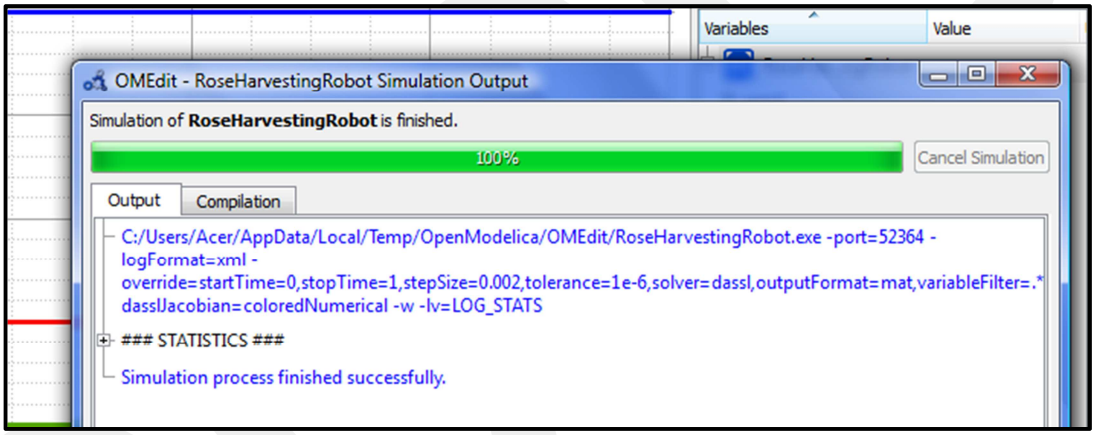


Figure 3.62. The simulation run by OpenModelica is complete.

The simulation results for RoseHarvesting class not only include the rose flower information, but also provide the results about the system such as the position of the rose, the ripeness information, the position of robot arm and the process result. This system can perform an entire process from sensing the rose flower to the cutting. There are some results for this process from start to finish (Table 3.1).

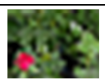
ROSE	SIZE	RECOGNIZED	ROSE AREA	RIPE	XPOS AS PIXEL	YPOS AS PIXEL	CALCULATED XPOS	CALCULATED YPOS	CUT
	22*17	YES	14	NO	8.4	6	252	180	NO
	19*15	YES	61	YES	9.19	8.54	275.7	256.2	YES
	8*13	YES	10	NO	6.75	5.25	202.5	157.5	NO
	14*20	YES	24	YES	10	7,81	300	234.375	YES
	17*23	YES	50	YES	18.91	9.86	567.5	295.909	YES
	17*24	YES	30	YES	18.7	7.444	561	223.333	YES
	18*24	YES	57	YES	10.961	7.571	328.846	227.143	YES
	20*20	YES	61	YES	10.6	10.5	318	315	YES
	18*24	YES	8	NO	3.75	13.5	112.5	405	NO
	23*16	NO	0	NO	0	0	0	0	NO

Table 3.1. The results for various runs for different rose flowers

Graphical results for some roses are provided below. In Figure 3.63, the simulation result for 17x24-sized input image is seen as 561 for x-position and 223 for y-position, meaning the rose flower is in these coordinates. The rose is decided as ripe, that results in CUT command as 1. Also the movement of Cartesian robot arm can be seen clearly in the chart. Similarly, Figure 3.64 shows the simulation result for 18x24 sized input image with the same variables, with the values of 328 and 227 for x and y positions, and the CUT command as 1. Figure 3.65 displays the x and y positions as 318 and 315 and the rose is ripe and can be cut (cut=1) for the input image sized 20x20.

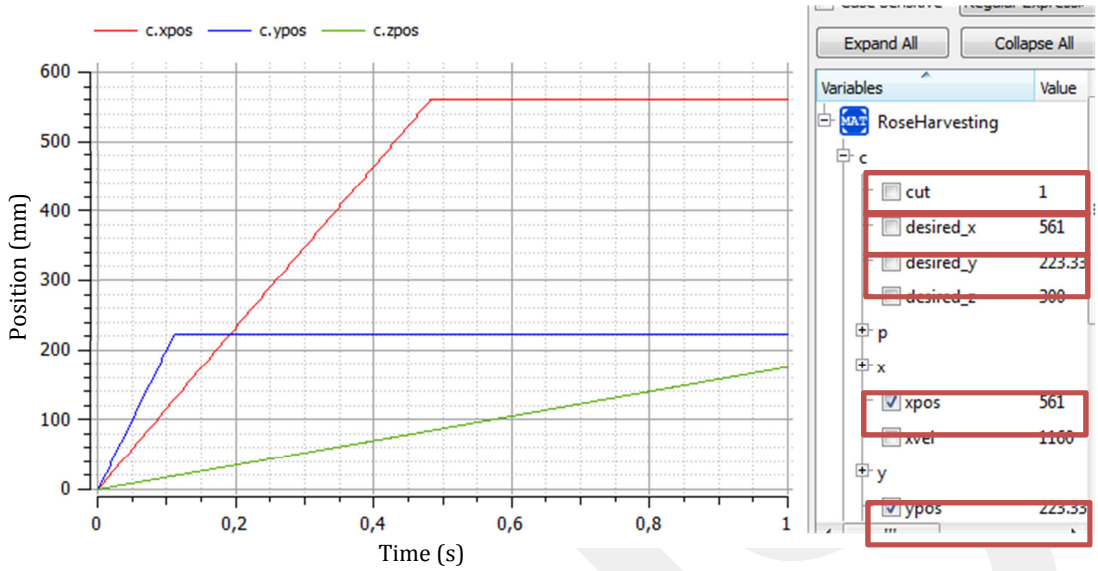


Figure 3.63. The simulation result for the input image 17x24

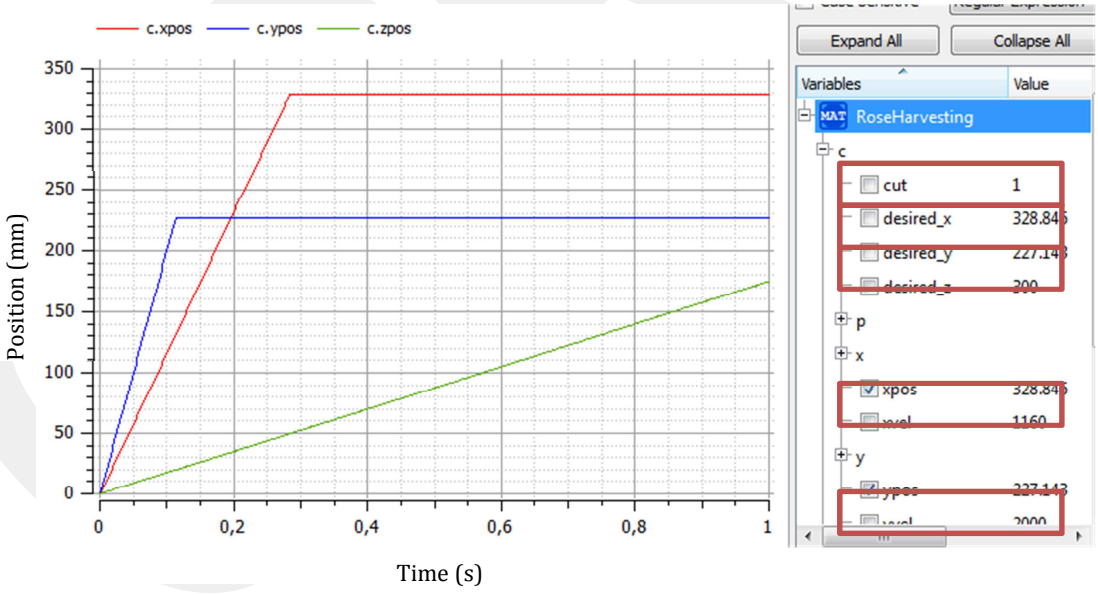


Figure 3.64. Simulation result for the input image 18x24

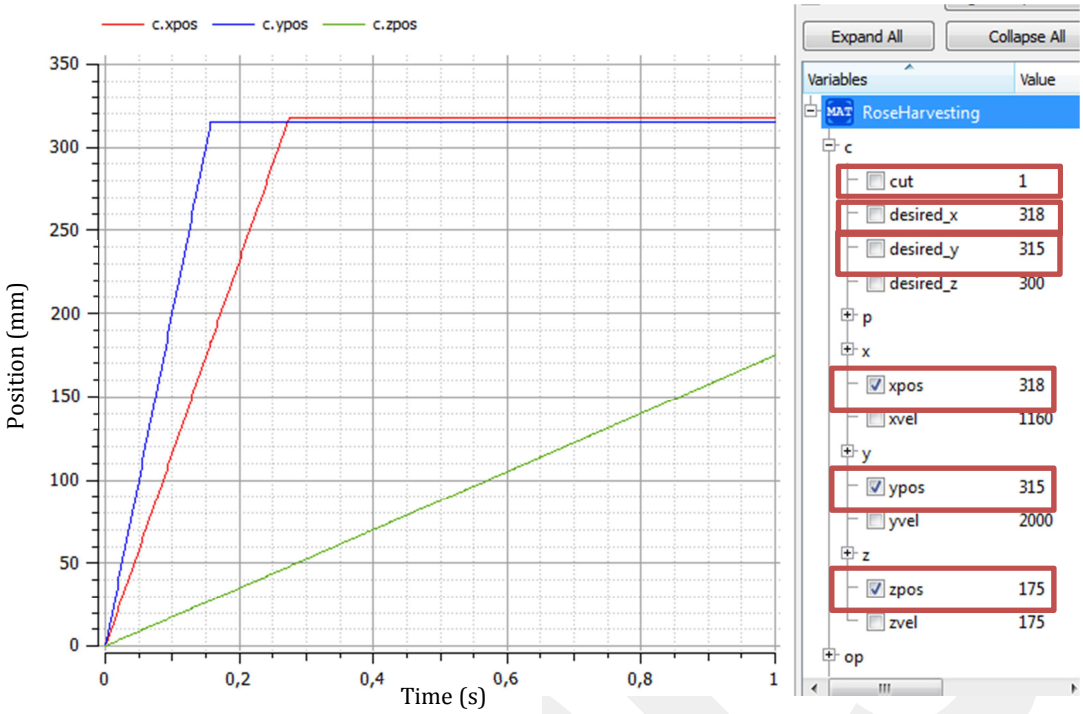


Figure 3.65. Simulation result for input image sized as 20x20

Coming to the examples for not ripe or not recognized roses, the simulation results for different input images can be shown as suitable samples. An 18x24 image is given to the system as an input and it actually has one rose flower. However, the decision mechanism determined this rose as not ripe. Therefore, in spite of finding the coordinates as 112 and 405 for x and y axes, the CUT command is decided as 0, and the Cartesian robot does not go to the calculated positions and the rose is not cut. The result can be seen in Figure 3.66.

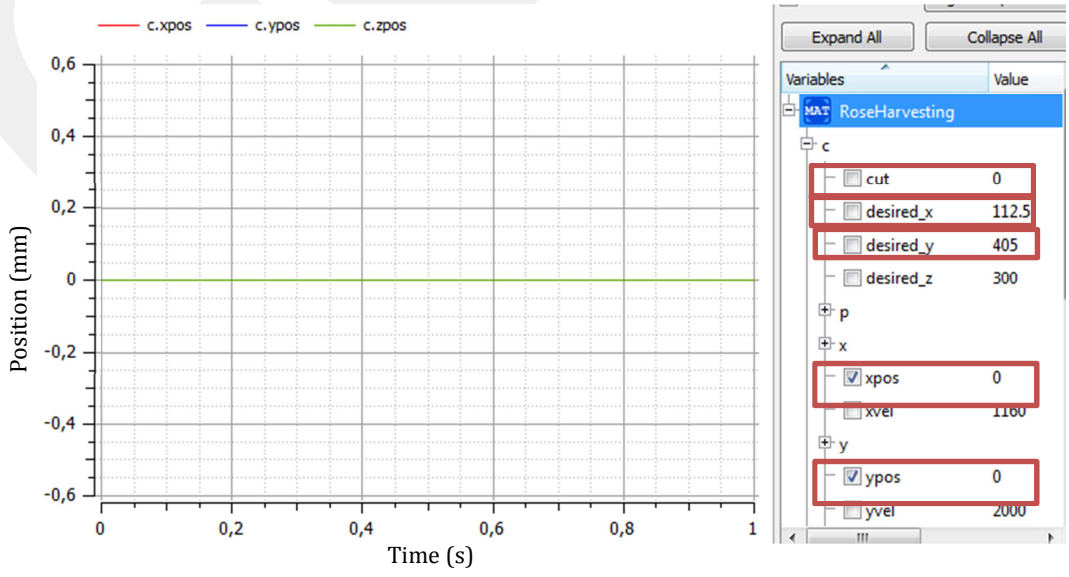


Figure 3.66. Simulation result for the input image having an unripe rose

The 23x16-sized input image has a rosebud which cannot be recognized easily. The shape of the bud is not a circle and there are not sufficient red pixels inside the picture. This situation makes the rose flower unrecognizable for the system. The simulation results can be seen in Figure 3.67. Because the rose flower cannot be recognized by the system, the coordinates are not found and the cut command remained as 0. Therefore, Cartesian robot arm does not move.

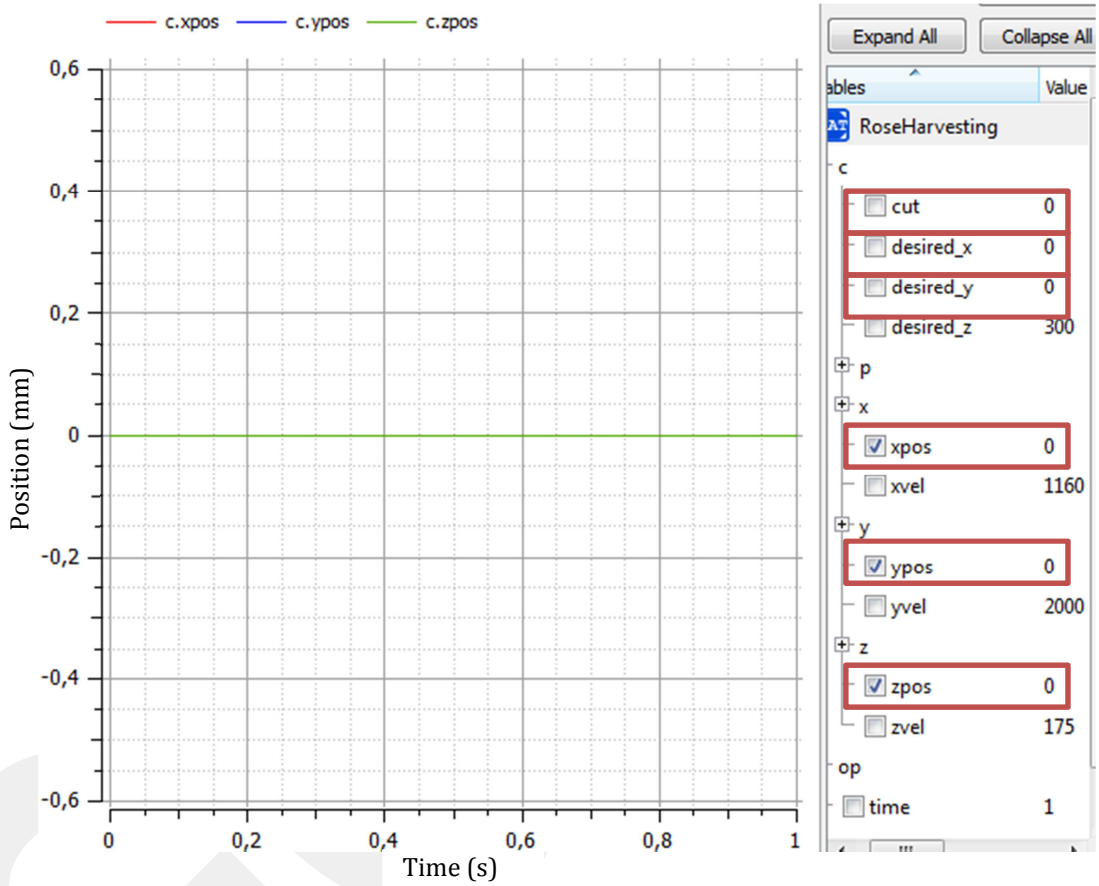


Figure 3.67. Simulation result for input image having an unrecognizable rose flower

3.2.10. Conclusion for Rose Harvesting Robot System

The Rose Harvesting Robot system was modelled in OpenModelica using both writable and graphical platforms. First, the subsystems were modelled in writable platform as different classes using subclasses. The integration of the classes are provided inside another class both in writable platform and the graphical platform. All three subsystems follow their desired function and the entire system is successfully modelled and simulated using OpenModelica platform.

CHAPTER 4

DISCUSSION AND CONCLUSION

In this study, object-oriented simulation models of chosen mechatronic systems are developed. The related language for modelling is studied before the application. Also, the simulation tools are investigated and an appropriate simulation tool is decided and the developed models are implemented with this simulation tool. The results of the simulations are gathered and the modelling and the simulations are evaluated. In this chapter, the discussion about the study is given and the thesis is concluded including the future work suggestions.

4.1. Discussion

In this study, Modelica Language is used via OpenModelica Platform. Therefore, the evaluation of Modelica Language for this study will be specific to OpenModelica platform. First, the evaluation of the language will be done in sight of the case studies. Difficulties and simplicities during modelling the Inverted Pendulum and Rose Harvesting Robot Systems will be explained. Then, the evaluations will be generalized to the Mechatronic Systems. The advantages and disadvantages will be provided and the study will be concluded.

4.1.1. Modelica Language specific to case studies

During the modelling, there have been many problems. Most of the problems were able to be solved. However, some of the problems are necessary to mention. Also, some of the advantages are well worth saying.

Inverted pendulum is modelled using mathematical equations. Differential equations of motion are used to model the pendulum. First, the main advantage of Modelica is the ease of use. As part of object-oriented structure, the variables and motion equations are directly placed inside a class. OpenModelica uses the equations without distinction of order. This *acausal* structure makes the modelling easier than conventional block diagram method or causal (reason-result) assignment statements.

Also, the system has easily been designed as objects. The cart, pendulum and controller were able to be designed and combined in a straightforward way.

On the other hand, a PD controller is designed for Inverted Pendulum. The equation for controller is placed inside the controller class. However, the values for related coefficients are given manually. There is no automatic tuning, which is an important property for complex systems.

Rose Harvesting Robot is a more complex system compared to Inverted Pendulum. This system is modelled using component method. The components (subsystems) of the system are modelled as objects. These subsystems use appropriate methods inside the classes.

First, the mechanical subsystem is modelled using mathematical equations. These are linear equations for motion. These simple equations are placed into the class and the movement of the Robot arm is shown. This modelling is quite easy. Reusability of created classes and multiple instantiation ease the modelling, for example, for each axis only one class is used. Also, simulation is quick and reliable.

Secondly, the Operator Subsystem is modelled. While modelling this subsystem, function modelling is considered. Although the object model is simple, this class had some problems. Because this class combines two other systems to each other, there have been connection problems. Two different types of systems may not be compatible with each other. However, it is difficult to understand the reasons for the errors. Some of the error messages are difficult to understand, because they do not indicate the exact place where the error occurs. Even if the system is modelled regularly, the background solution platforms may be incompatible and simulation crashes.

Third, the Robot Vision Subsystem is modelled considering function modelling. This system modelling is crucial for this study. There are really serious difficulties during this modelling. One of the main problems is the lack of a specialized tool for Image Processing inside OpenModelica. This shortage causes wasting time and may be the reason for unclear results. For example, there is a need for image reading and image writing, however there are no such ready functions. Another disadvantage related to

this problem is the lack of special image processing commands such as RGB value obtaining, RGB to binary or vice versa.

Second main problem can be said that data import and data export cannot be done at file level. Therefore, the required matrices were inputted by the researcher manually, and the output matrices were written in file by researcher herself. This may cause mistakes and data losses.

The third point related to this system is about matrix processing. Matrix processing is rather slow. Therefore, the matrix inputs are limited to small indexed ones. In this study, maximum processed matrix is a size of 26x18, processed in approximately 10 minutes. Another reason for limitation of the matrix size is memory allocation. OpenModelica cannot allocate bigger sized matrices while processing, and exits the program giving memory error. The simulation cannot be done. Therefore, OpenModelica can be said to be not suitable for matrix processes. In other words, OpenModelica does not have a proper platform for matrices.

In this class, not dependent to particularly this subsystem, there is one problem that function classes cannot return more than one value. In official tutorial sources, it is said to be, however, despite using indicated syntax, it was not able to get more than one return value. This problem caused the writing one function for each desired value. If a large-scale model is done on OpenModelica, this will lead to redundant classes, duplications and complexity troubles. On the other hand, if this problem is overcome, the function structure is seriously beneficial. In Modelica, functions are specialized classes, that is to say, functions are also objects, which use needed algorithm sections inside. These properties bring easy writing and usage. Function classes are practical, especially if the mathematical equations and algorithms need to be separated; because, algorithms are needed even in object-oriented modelling and Modelica provides this advantage.

4.1.2. Advantages and disadvantages of OpenModelica special to mechatronic systems

In sight of these studies, some advantages and disadvantages of Modelica can be deduced for Mechatronic systems.

First advantage is the easy use of equations in modelling. In general, mathematically modelled systems can easily be implemented in OpenModelica, due to the “acausal” modelling. That means, equations are not based on cause and effect relationship, and can be written directly into the classes without considering the order of equations. Therefore, mathematically implemented systems, mechanical or electrical, can be modelled using the Modelica language.

Second advantage of the Modelica language is the chance of using different domain models in one platform. The example for this can be said as mechanical systems using controller systems. Controller systems can be applied to mechanical systems which are operated by embedded systems with software or electronic systems.

Algorithms are the third advantage of Modelica on Mechatronic systems. To develop mechatronic systems, decision making mechanisms should be included. In Modelica, algorithms can easily be developed and used by mechanical, electrical or information systems. Decision making is covered within the systems together.

Another benefit of Modelica language is the wide standard library. There are ready libraries and tools consisting of various ready components and objects. Also, the researchers are free to develop their own components and classes, not being dependent on the ready tools. Thinking of mechatronic systems, this flexibility provides customizing standard components in line with the requirements of a mechatronic system. The special goals of a mechatronic system may be met by customized classes, components or libraries; or a needed library is able to written from beginning to end.

Last but not least, Modelica has the advantage of object-orientation to be able to apply top-down approach for Mechatronic systems. The hierarchical structure of Modelica allows the application of top-down approach easily. Low-level classes can be merged in higher-level classes due to the object orientation, and the top level system can be decomposed until the endpoint systems. Different domain systems can be implemented in lower level classes and can be combined in a top-level class, which is the most characteristic property of Mechatronic systems as a multidisciplinary approach.

Free from any doubt, Modelica language also has disadvantages. Some drawbacks of the Modelica language should be written considering Mechatronic systems. First of all, there are some ready tools of Modelica, providing ease of use and wide application area. However, these tools are not sufficient. Some of the tools such as the ones for Image Processing are absent. This is important for Mechatronic systems, because robot vision plays an important role in many mechatronic systems and image processing is a widely used method for robotic vision. After all, Modelica does not provide even fundamental image processing functions yet.

Modelica has another disadvantage. Mechatronic systems consist of systems from different disciplines. While modelling different systems, Modelica can easily be used and each system works excellently in itself. On the other hand, combining such systems may cause compatibility and robustness issues on OpenModelica. Properly working systems may start to give errors based on background differences. This problem may be relevant to combining the varied solution methods of OpenModelica for solving different-discipline systems in one platform. Therefore, while modelling large-scale mechatronic systems in Modelica, each component increases the complexity and the risks arise. Therefore, while developing such systems, developers should be careful about this risk.

In conclusion, Modelica is a highly-effective and flexible modelling language. It is simple and powerful for modelling Mechatronic systems. There are many advantages of Modelica in Mechatronic modelling. Even if it has disadvantages, the favours outweigh the drawbacks. The ease of use, the multi-disciplinary usability, the ability to solve critical problems, easy accessible platforms as OpenModelica, flexibility of development make Modelica a powerful language for modelling Mechatronic systems. In case of elimination of deficiencies, Modelica is a quite convenient language for Mechatronic systems modelling.

4.2. Conclusion

The first step of this study is the modelling method clarification. The proposed method for this thesis is the modelling based on object-oriented modelling language Modelica. This basic method is shown that it can be applied on Mechatronic systems. To do this, two case studies are performed. Two selected systems are modelled and implemented using Modelica Language.

First system, Inverted Pendulum was modelled using the mathematical characteristics of each component. The component objects are defined individually and the characteristic features are implemented in these objects.

Next, the second mechatronic system, Rose Harvesting Robot is applied with the same procedure. That means the system of the robot is decomposed into its components, which means, objects of the system. These subsystems are also modelled individually, considering their characteristics. The Rose Harvesting Robot model is easily established via this method. There is a simple and effective model obtained using component decomposition method.

Finally, the developed models for the two selected systems are implemented on the chosen simulation tool, OpenModelica. The reason for choosing this tool is the Modelica-based infrastructure of OpenModelica. Therefore, the developed models of Inverted Pendulum and the Rose Harvesting Robot are easily implemented in terms of class representation using the Modelica language. The components of the systems are defined in the simulation model. The simulation is run in the OpenModelica platform and the results are achieved.

The results of the simulations show that, the object-oriented modelling method is implemented on the mechatronic systems in an easy way. The results of the evaluation of Modelica language show that Modelica is a convenient language for mechatronic systems. However, there were some problems during the implementation, which are explained in the evaluation section. On the other hand, most of these problems were not related to the model structure or the implementation. The reason for the problems is the OpenModelica Environment. Due to its restricted platform about the acausal programming, the implementation may take a long time. Also, the tool has some problems and shortages in itself. The reason for these problems is that OpenModelica is an open-source environment, developing platform and it is still developing.

System modelling and implementation on simulation environments are important parts for engineering design. Therefore, the system modelling in an effective way is crucial. The simulation tools use the developed models to make simulations and aim to result in correct deductions. Therefore, designers can set up desired real physical systems in an accurate way.

Main goal of this thesis is applying the effective modelling method known as object-oriented modelling on the mechatronic systems and using the developed models in a chosen simulation tool via using Modelica modelling language. The study shows that the method can be easily applied on mechatronic systems and simulation modelling; and Modelica is an appropriate language for mechatronic systems.

5.3. Future work

In this study, the mechatronic systems are shown that can be implemented by object-oriented modelling method using Modelica. The future work related with this study may be the change on the simulation tool specific to mechatronic systems examined in this study. The developed model of any mechatronic system can be implemented by changing simulation tools as Dymola or AMESim based on again Modelica Language. The change on the simulation tool enables the researchers to have the opportunity to compare the tools with each other, as well as compare the applicability of Modelica language in different simulation platforms. Especially, other tools based on Modelica can be investigated in terms of the libraries such as Image Processing. The future work also will show the advantages and disadvantages of the presented method, depending on the advantages and disadvantages of the simulation tools while simulating mechatronic models.

REFERENCES

Abdul Rahman, M.A., Mizukawa, M., 2013, "Modeling and design of mechatronics system with SysML, Simscape and Simulink," *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp.1767-1773, 9-12 July 2013. doi: 10.1109/AIM.2013.6584353

Akdağ, M., Karagülle, H., Malgaca, L., 2011, "An integrated approach for simulation of mechatronic systems applied to a hexapod robot", *Mathematics and Computers in Simulation*, Vol. 82, no. 5, January 2012, pp. 818–835.

Al-Sharif, L., Tutunji, T. A., Ragab, D., and Kayfi, R., 2014, "Using elevator system modelling and simulation for integrated learning in mechatronics engineering", *15th International Workshop on Research and Education in Mechatronics (REM)*, pp.1-8, 9-11 Sept. 2014.

Aydın, İ., 2013. *Introductıon to Simulation and Modelling.* (In Turkish). *[BMÜ421: Benzetim ve Modelleme]* Fırat University. Access date: 21.10.2015. http://web.firat.edu.tr/iaydin/Introduction_to_simulation_and_modeling_lecture_1.pdf

Banks, J., Carson, J., Nelson, B., Nicol, D., 2001. Discrete-Event System Simulation. 3rd ed: Prentice Hall, Mishawaka, IN, U.S.A.

Bin, H., Feng L. H., Zhen L. W., 2010, "A Principle Solution Representation Model Based on Design Catalogue in Mechatronics Product Design," *2010 Second International Conference on Computer Engineering and Applications (ICCEA)*, vol.2, pp.434-438, 19-21 March 2010. doi: 10.1109/ICCEA.2010.262

Bishop, R.H., 2007, Mechatronic Systems, Sensors and Actuators: Fundamentals and Modeling. The Mechatronics Handbook, 2nd ed: CRC Press.

Bolton, W., 1999, Mechatronics: Electrical Control Systems in Mechanical and Electrical Engineering, 2nd Ed: Addison Wesley Longman, England.

Cabrera, A.A.A., Erden, M.S., Foeken, M.J., Tomiyama, T., 2008, "High Level Model Integration for Design of Mechatronic Systems," *IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, 2008. MESA 2008,* pp.387-392. doi: 10.1109/MESA.2008.4735736

Carson II, J.S., 2004, "Introduction to modeling and simulation," *Simulation Conference, 2004. Proceedings of the 2004 Winter*, vol.1, pp., 16. doi: 10.1109/WSC.2004.1371297

Cao, Y., Liu, Y., Paredis, C. J., 2011. System-level model integration of design and simulation for mechatronic systems based on SysML. *Mechatronics*, vol.21, no.6, pp.1063-1075.

Coller, Briano, 2015. *Classic Inverted Pendulum - Equations of Motion.* Available at: https://www.youtube.com/watch?v=5qJY-ZaKSic (Access date: 27.08.2015)

Control Tutorials for MATLAB and Simulink, 2015. Avaliable at: http://ctms.engin.umich.edu/CTMS/index.php?aux=Home (Access date: 27.08.2015)

Dempsey, M., 2006, "Dymola for Multi-Engineering Modelling and Simulation," *Vehicle Power and Propulsion Conference, 2006. VPPC '06. IEEE*, pp.1-6. doi: 10.1109/VPPC.2006.364294

Dickinson, E.J.F., Ekström, H., Fontes, E., 2013, "COMSOL Multiphysics: Finite Element Software for Electrochemical Analysis. A Mini Review", *Electrochemistry Communications*, vol. 40, pp. 71-74.

Dizqah, A.M., Maheri, A., Busawon, K., Fritzson, P., 2013, "Acausal Modelling and Dynamic Simulation of the Standalone Wind-Solar Plant Using Modelica," *2013 UKSim 15th International Conference on Computer Modelling and Simulation (UKSim),* pp.580-585, 10-12 April 2013 doi: 10.1109/UKSim.2013.145

Drumea, A., Vasile, A., Svasta, P., Ilie, I., 2008, "Modelling and simulation of simple mechatronic system - position control solution based on linear variable

inductor displacement transducer," *2nd Electronics System-Integration Technology Conference, ESTC 2008*, pp.225-230, 1-4 Sept. 2008
doi: 10.1109/ESTC.2008.4684354

Elmqvist, H., Mattsson, S.E., Otter, M., 1999, "Modelica - a language for physical system modeling, visualization and interaction," *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design, 1999.*, pp. 630-639, 1999 doi: 10.1109/CACSD.1999.808720

FESTO, Servo Motors EMMS-AS, available at: https://www.festo.com/net/SupportPortal/Files/10273/EMMS-AS_ENUS.pdf (Access date: 25.11.2015)

FESTO, Three-Dimensional Gantries, available at: https://www.festo.com/cat/fr_fr/data/doc_engb/pdf/en/dhsr1_en.pdf (Access date: 25.11.2015)

Filipescu Jr., A, Filipescu, A., Petrea, G., Filipescu, S., Minca, E., 2013 "Discrete modelling based control of a processing/reprocessing mechatronics line served by an autonomous robotic system," *4th International Symposium on Electrical and Electronics Engineering (ISEEE), 2013*, pp.1-6, 11-13 Oct. 2013 doi: 10.1109/ISEEE.2013.6674360

Fritzson, P., 2010, Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, Wiley-IEEE Press.

Fritzson, P., 2011, "Modelica — A cyber-physical modeling language and the OpenModelica environment," *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pp.1648, 1653. doi: 10.1109/IWCMC.2011.5982782

George, V. I., D'Souza, J., Kurian, C. P., and Thirunavukkarasu, I., 2011, "A Simulink Model for an Aircraft Landing System Using Energy Functions", *7th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2012,* pp. 355-360.

Ghazi, S.S.M., Zarei, A., 2007, "A Mechatronic System Design Case Study: Adaptive Modeling and Control of an Inverted Pendulum," *2007 IEEE International Conference on Networking, Sensing and Control*, pp.291-296, 15-17 April 2007. doi: 10.1109/ICNSC.2007.372793

Guangbu, L., and Ruqiong, L., 2009, "Belt Conveyor Modeling and Performance Simulation Based on AMESim", *Second International Conference on Information and Computing Science, 2009*, pp. 304-307. doi: 10.1109/ICIC.2009.387

He, D., Xiong, Q., Shi, W., Shi, X., Zhang, J., 2014, "Building energy and control system modeling and simulation using Modelica," *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pp.2794-2797, May 31 2014-June 2 2014 doi: 10.1109/CCDC.2014.6852648

Huang, J., Voeten, J., Groothuis, M., Broenink, J., Corporaal, H., 2007, "A model-driven design approach for mechatronic systems," *Seventh International Conference on Application of Concurrency to System Design, 2007. ACSD 2007*, pp.127-136, 10-13 July 2007. doi: 10.1109/ACSD.2007.40

Ingalls, R.G., 2011, "Introduction to simulation," *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pp.1374-1388. doi: 10.1109/WSC.2011.6147858

Kalinski, K. J., Buchholz, C., 2014, "Mechatronic design of strongly nonlinear systems on a basis of three wheeled mobile platform", *Mechanical Systems and Signal Processing*, vol.52-53, pp. 700-721.

Kayani, S.A., Malik, M.A., 2007, "Automated Design of Mechatronic Systems using Bond-Graph Modeling and Simulation and Genetic Programming," *International Bhurban Conference on Applied Sciences & Technology, IBCAST 2007*, pp.104-110, 8-11 Jan. 2007 doi: 10.1109/IBCAST.2007.4379917

Khurshid, A., Malik, M.A., 2007, "Bond Graph Modeling and Simulation of Impact Dynamics of a Car Crash," *International Bhurban Conference on Applied Sciences & Technology, 2007. IBCAST 2007*, pp. 63-67, 8-11 Jan. 2007
doi: 10.1109/IBCAST.2007.4379910

Lesnicwska, A., Choromanski, W., Deszezynski, J., Dobrzynski, G., 2006, "Modeling and simulation of physical performance of an External Unilateral Mechatronic Orthopaedic Fixator - Bone system," *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2006. EMBS '06,* pp.1533-1536, Aug. 30 2006 - Sept. 3 2006 doi: 10.1109/IEMBS.2006.259328

Li, W., Abel, A., Todtermuschke, K., Zhang, T., 2007, "Hybrid Vehicle Power Transmission Modeling and Simulation with SimulationX", *International Conference on Mechatronics and Automation, 2007. ICMA 2007,* pp.1710-1717. doi: 10.1109/ICMA.2007.4303808

Manhawy, W.E., Hammad, S., 2006, "Behavioral Modeling and Simulation of Hexapod2 Gait System," *Canadian Conference on Electrical and Computer Engineering, 2006. CCECE '06,* pp.398-401, May 2006.
doi: 10.1109/CCECE.2006.277396

Mhenni, F., Choley, J.-Y., Nga Nguyen, 2014, "Extended mechatronic systems architecture modeling with SysML for enhanced safety analysis," *2014 8th Annual IEEE Systems Conference (SysCon),* pp.378-382, March 31 2014-April 3 2014 doi: 10.1109/SysCon.2014.6819284

Miatliuk, K., 2013, "Conceptual model for mechatronic design in the basis of hierarchical systems technology," *2013 14th International Carpathian Control Conference (ICCC),* pp.243-244, 26-29 May 2013. doi: 10.1109/CarpathianCC.2013.6560546

Naylor, T.J., Balintfy, J.L., Burdick, D.S., and K. Chu, 1966, Computer Simulation Techniques, Wiley, NY.

Nielsen, K., Joergensen, K.A., and Petersen, T.D., 2009, "Mechatronics and Mass Customization", *5th World Conference on Mass Customization & Personalization, MCPC 2009,* pp. 1-8.

Oh, S., Sun, J., Li, Z., Celkis, E. A., Parsons, D., 2010, "System Identification of a Model Ship Using a Mechatronic System," *IEEE/ASME Transactions on Mechatronics,* vol.15, no.2, pp.316-320, April 2010. doi: 10.1109/TMECH.2009.2024308

Oxford Learner's Dictionary, 2016. Available at: http://www.oxfordlearnersdictionaries.com/ (Access date: 23.02.2016)

Qamar, A., During, C., Wikander, J., 2009, "Designing mechatronic systems, a model-based perspective, an attempt to achieve SysML-Matlab/Simulink model integration," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2009. AIM 2009*, pp.1306-1311, 14-17 July 2009 doi: 10.1109/AIM.2009.5229869

Regulin, D., Krooß, C., Rehberger, S., Vogel-Heuser, B., 2013, "Efficient modeling of mechatronic systems regarding variety and complexity in the field of automotive," *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pp.3505-3510, 10-13 Nov. 2013 doi: 10.1109/IECON.2013.6699692

Rosenberg, Ronald C., and Baihai Zhang. (N.D.) "Automated Design Methodology for Mechatronic Systems Using Bond Graphs and Genetic Programming." Access date: 21.10.2015. Available at: http://garage.cse.msu.edu/papers/GARAGe02-01-01.pdf

Saleem, A., 2010, "Component-based modeling framework for mechatronic systems," *2010 7th International Symposium on Mechatronics and its Applications (ISMA)*, pp.1-5, 20-22 April 2010.

Scherber, S., Müller-Schloer, C., 1999, "An efficient and flexible methodology for modelling and simulation of heterogeneous mechatronic systems," *Design, Automation and Test in Europe Conference and Exhibition 1999. Proceedings*, pp.784-785. doi: 10.1109/DATE.1999.761230

Schneider, P., Bayer, C., Einwich, K., Köhler, A., 2012, "System level simulation — A core method for efficient design of MEMS and mechatronic systems," *9th International Multi-Conference on Systems, Signals and Devices (SSD), 2012*, pp.1,6, 20-23 March 2012 doi: 10.1109/SSD.2012.6198066

Simic, D., Haumer, A., Bauml, T., Pirker, F., 2007, "Modeling, Simulation and Evaluation of a Cooler Model in Modelica using Dymola," *Vehicle Power and Propulsion Conference, 2007. VPPC 2007. IEEE*, pp.623-628. doi: 10.1109/VPPC.2007.4544198

Singh, V.P., 2009, System Modeling and Simulation. Daryaganj, Delhi, IND: New Age International. ProQuest ebrary. Web. 26.10.2015.

Spiryagin, M., Simson, S., Cole, C., and Persson, I., 2012, "Co-simulation of a Mechatronic System Using Gensys and Simulink", *Vehicle System Dynamics*, vol. 50, no. 3, pp. 495-507.

Szolik, L., Zakova, K., 2012, "OpenModelica based remote control of thermo-optical plant," *9th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pp.1-4, doi: 10.1109/REV.2012.6293167

Tatu, N.-I., Alexandru, C., 2012, "Modeling and simulation of the tracking mechanism for a PV string," *IEEE International Conference on Automation Quality and Testing Robotics (AQTR)*, pp.428-433, 24-27 May 2012. doi: 10.1109/AQTR.2012.6237748

Tian shi-xiang, Wang sheng-ze, 2010, "The conceptual design and simulation of mechatronic system base on UML," *2nd International Conference on Computer Engineering and Technology (ICCET), 2010*, vol.6, pp.V6-188, V6-192, 16-18 April 2010 doi: 10.1109/ICCET.2010.5486322

Toufighi, M.H., Sadati, S.H., Najafi, F., 2007, "Modeling and Analysis of a Mechatronic Actuator System by Using Bond Graph Methodology," *2007 IEEE Aerospace Conference*, pp. 1-8, 3-10 March 2007 doi: 10.1109/AERO.2007.353093

Valasek, M., 2010, "Highly efficient models and simulations – the basis of design of mechatronic systems", *Metalurgija*, Vol. 49, Issue 2, pp. 127-137.

van Beek, T.J.; Tomiyama, T., 2008, "Connecting Views in Mechatronic Systems Design, a Function Modeling Approach," *IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, 2008. MESA 2008*, pp.164, 169, 12-15 Oct. 2008 doi: 10.1109/MESA.2008.4735676

Wei, G., Xiangyang, X., Yongxin, C., and Yang, Y., 2011, "Simulation of Powertrain and Dynamics of Automobile Based on SimulationX", *6th IEEE Conference on Industrial Electronics and Applications*, pp. 2326-2330. doi: 10.1109/ICIEA.2011.5975981

Wen, G., Xu, L., He, F., Zhang, X., 2009, "Kinematics Simulation to Manipulator of Welding Robot Based on ADAMS," *International Workshop on Intelligent Systems and Applications, 2009. ISA 2009,* pp.1-4. doi: 10.1109/IWISA.2009.5072932

Wikipedia, *Inverted pendulum*. Available at: https://en.wikipedia.org/wiki/Inverted_pendulum (Access date: 27.08.2015)

Zhang, M., Fisher, W., Webb, P., Tarn, T., 2003, "Functional model based object-oriented development framework for mechatronic systems," *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*, vol.2, pp.2153-2158 vol.2, 14-19 Sept. 2003 doi: 10.1109/ROBOT.2003.1241912

Zhou, Y., Peng, F., Li, B., Li, Y., 2010, "Mechatronic Modeling and Analyzing for Feed Servo Control System Based on Torsion Dynamics of Lead-Screw," *2010 International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, vol.2, pp.632-635, 13-14 March 2010. doi: 10.1109/ICMTMA.2010.337

APPENDIX

Some of the systems in CTMS webpage are simple and important examples of mechatronic systems. These systems consist of the electronic parts, mechanical parts and controller parts. Therefore, three main components of mechatronic systems are covered inside these simple systems. One of these systems, Inverted Pendulum, was chosen and modelled in the scope of the thesis. However, in a brief manner, one of these systems are chosen and modelled on OpenModelica platform; and the results of the simulation are shared. This study is valuable for discussion on Modelica language if it is suitable for mechatronic systems.

All information related to system models of the following are derived from CTMS webpage.

A.1. Motor Speed System and Motor Position System

These two systems are actually the same system, DC Motor. The only difference is one of them controls the speed and the other does the position of the motor. Therefore, the physical system is in common, and it has an internal electrical circuit and a rotor (Figure A.1).

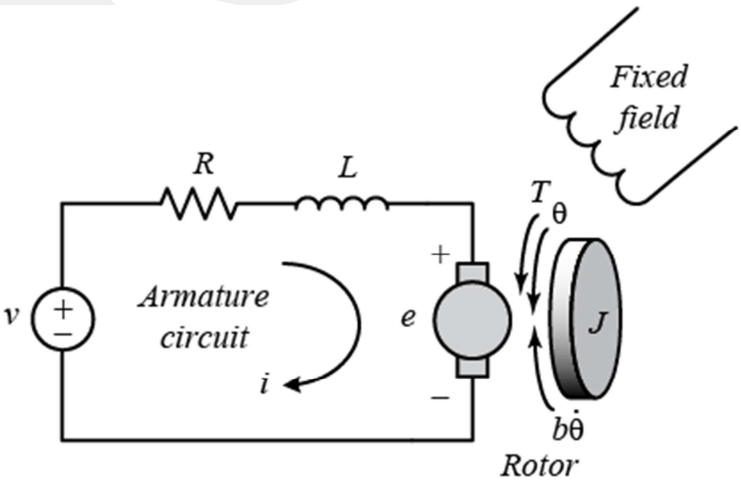


Figure A.1. DC Motor physical system

Parameters of the system are given as follows:

- J: moment of inertia of the rotor
- b: motor viscous friction constant
- K_e (electromotive force constant) = K_c (motor torque constant) = K
- R: electric resistance
- L: electric inductance

The equations can be derived and represented using Newton's 2nd law and Kirchoff's voltage law.

$$J\ddot{\theta} + b\dot{\theta} = Ki$$

$$L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

Motor Speed System is a DC motor which is controlled by the input voltage; and the output of the system is the rotational speed.

The input and output relation can be seen in Figure A.2.

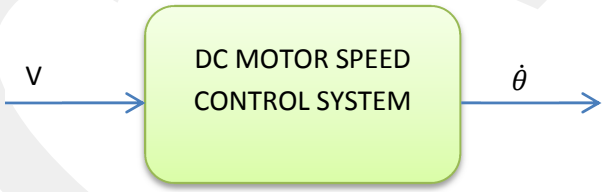


Figure A.2. The input and the output of DC Motor Speed Control system

Motor Position System is a DC motor which is controlled by the input current; and the output of the system is the angular position. The input and output of the system can be seen in Figure A.3.

Figure A.3. The input and the output of DC Motor Position Control system

For both systems, the modelling in OpenModelica can be done in three ways. First way of implementation of this system is using the mathematical model directly inside the writable platform and creating a DC_Motor object. Inside this system, again it is really simple to add a controller to the system.

Second method is implementation of the electrical and mechanical parts as separate objects and placing the related differential equations inside related classes, and also designing a controller object discretely.

Third way is using the ready libraries inside Modelica (Figure A.4.). Modelica provides the electrical and electromechanical elements inside the standard library, which means the physical topology can be represented easily by using the elements.

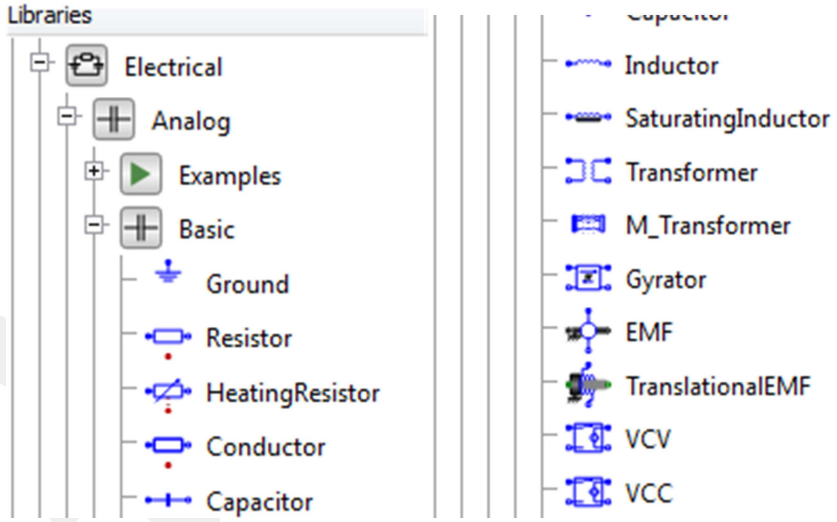


Figure A.4. Electrical library in OpenModelica

A.2. Implementation on OpenModelica

A model is constructed using writable platform of OpenModelica (Figure A.5).

```
model DC_Motor
  Real J = 3.228 * 10 ^ (-6);
  Real K = 0.0274;
  Real L = 2.75 * 10 ^ (-6);
  Real R = 4;
  Real b = 3.5077 * 10 ^ (-6);
  Real i, di;
  Real th, dth, ddth;
  Real V;
  parameter Real p, ref;
equation
  dth = der(th);
  ddth = der(dth);
  di = der(i);
  J * ddth + b * dth = K * i;
  L * di + R * i = V - K * dth;
  V = p * (ref - dth);
end DC_Motor;
```

Figure A.5. DC Motor Speed Control

This system can also be implemented using the ready libraries inside OpenModelica and the model can be seen in Figure A.6.

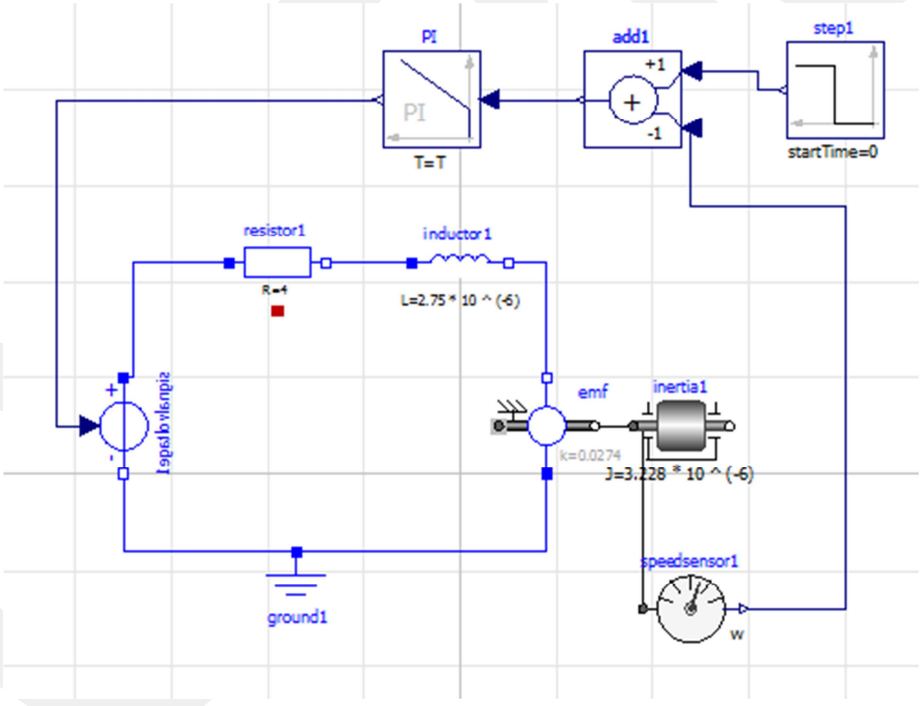


Figure A.6. Graphical model for DC Motor Speed Control System

A.3. Simulation Results

It can be clearly seen that both writable class DC_Motor and connection class DCMOT reaches the desired values and speed can be controlled easily. The simulation results for the two classes are given in Figure A.7.

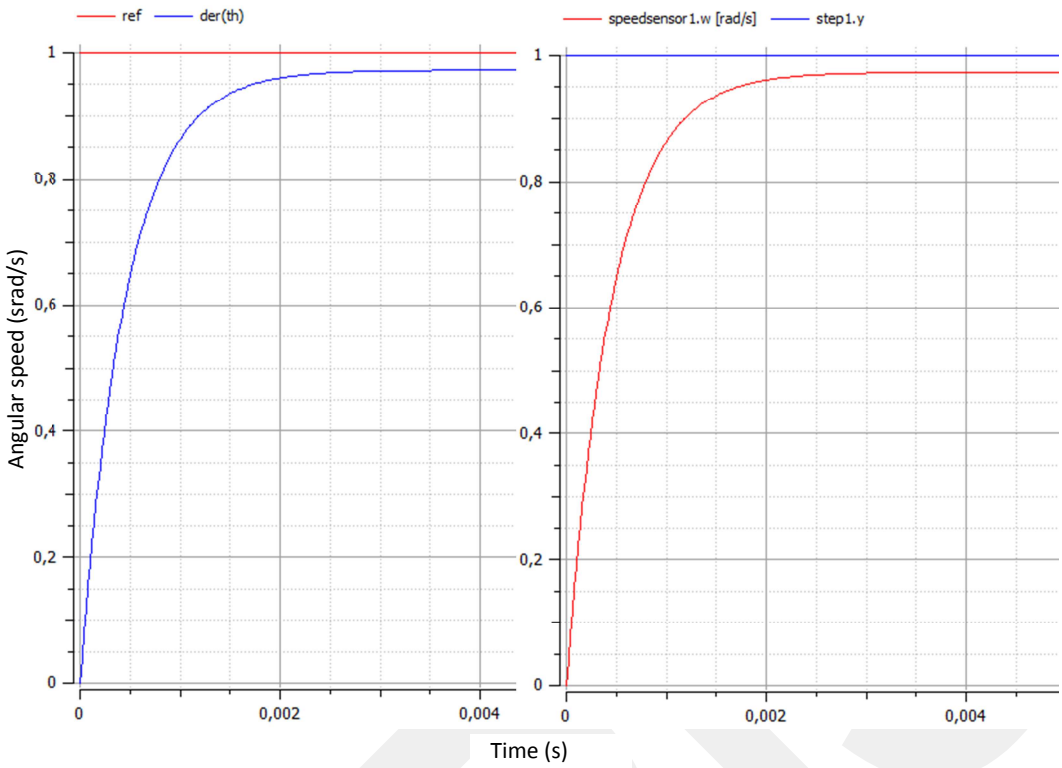


Figure A.7. Speed result for writable model (leftside) and for graphical model (rightside) for proportional controller constant k = 1 and step input reference

A.4. Conclusion for dynamic systems

In this study, one system provided in CTMS webpage is briefly evaluated in terms of applicability on Modelica. The study shows that this dynamic system can be implemented using Modelica easily. To illustrate the applicability, this system is implemented in OpenModelica in two different ways. The results indicate that OpenModelica is suitable for dynamic system modelling and simulation. Also, different domain systems can be set up easily.